Robust Human Action Modelling

Lei Wang

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

October 2023

Declaration. I hereby declare that this thesis represents my original research, and I am the sole author of this work. The experimental work is entirely my own, with clear acknowledgment of any collaborative contributions. Proper references have been provided for all supporting literature and resources. This thesis has not been submitted for any other degree or professional qualification.

Lei Wang
6 October 2023

To my beloved family
and my forever father,

for their love, understanding, and support.

Acknowledgments

This thesis could not have been written without the support of numerous individuals, for whom I am greatly appreciative.

Firstly, I would like to sincerely thank my primary supervisor, Dr. Piotr Koniusz (Data61/CSIRO and ANU), who dedicated tremendous effort and time to supervising my research work. His patience, guidance, and support have helped me grow as a researcher. He generously provided regular and pertinent feedback, which has steadily improved my research skills. Dr. Piotr Koniusz also offered invaluable suggestions for my daily life and future research career development, and I will always remember his contributions.

Secondly, I extend my sincere gratitude to Prof. Stephen Gould (ANU), the chair of the panel, who provided valuable feedback and suggestions when I embarked on my PhD journey. I hold him in high regard, and his guidance in research direction made my work both meaningful and rewarding. Prof. Stephen Gould also played a significant role in advancing my research career. Thank you.

Thirdly, I thank my associate supervisor, Dr. Liang Zheng (ANU), who joined the supervisory panel as I was completing my PhD. I had insightful discussions with him, and I thoroughly enjoyed exploring new ideas and research directions with him. Dr. Zheng also encouraged me to pursue the seemingly impossible. It is a great honor to work as a Research Fellow under his guidance, focusing on data-centric problems.

I want to express my gratitude to my former supervisor, A/Prof. Du Huynh from the Department of Computer Science and Software Engineering in the School of Physics, Mathematics, and Computing at The University of Western Australia. Her guidance and support during my master's degree and the beginning of my PhD encouraged me to work diligently and remain dedicated to my research interests.

I would also like to thank Mr. Graeme Woods (iCetana's former Head of Product) and Dr. Moussa Reda Mansour (iCetana's former Research & Development Lead). Their support and guidance in shaping my research career have been invaluable. Under their mentorship, I have (i) grown professionally, gaining a deeper understanding of business and further exposure to research and development; (ii) developed a critical mindset regarding techniques in machine learning and computer vision; (iii) enhanced my self-management skills; and (iv) become more receptive to new ideas and approaches. I also extend my gratitude to my former colleagues at iCetana, Francis Williams, Vaughan Harman, and Blair Davidson, with whom I had the pleasure of collaborating on industrial research projects over the past two years.

Finally, I would like to express my appreciation to my family for their unwavering encouragement and support throughout my research career. I also want to thank my wife, Xiaomei Xu, and my son, Ziyuan Wang, for their love, support, and understand-

ing. This thesis is also a gift to my son, Ziyuan Wang, for the future, reminding him that nothing comes easy, but hard work pays off.

The research works were supported by the Data61 PhD Scholarship, Data61 Top-up Scholarship, and ANU HDR Fee Remission Merit Scholarship. I also extend my thanks for the technical support provided by CSIRO Scientific Computing and NVIDIA for GPUs.

Abstract

Action recognition has numerous valuable applications, including human-computer interaction, smart video surveillance, sports, and healthcare. These applications have driven extensive research in this field in recent years. Various studies have shown that the performance of action recognition depends significantly on the extracted features and action representations. Despite the abundance of research in the literature, numerous challenges persist, such as geometric variations (*e.g.*, different camera viewpoints, rotations, visual appearances, and body sizes), photometric distortions (*e.g.*, lighting conditions, camera noise, and blur), and varying action execution speeds. Additional issues include partial occlusions of human subjects by objects in the scene and self-occlusion of human subjects.

This thesis concentrates on two primary tasks: (1) action classification, which predicts action concepts, and (2) few-shot learning for understanding the similarity between pairs of action sequences. The contributions of this work are threefold: (i) Demonstrating that combining data from different modalities, such as optical flow and object/saliency detection information, for action recognition outperforms single-modal approaches, and integrating feature representations or finding view-invariant action features from multiple camera viewpoints enhances the robustness of action recognition. (ii) Showing that higher-order feature representations and emerging Transformers, primarily based on self-attention modules, achieve state-of-the-art performance in various action recognition benchmarks, including fine-grained action recognition tasks. (iii) Addressing temporal and geometric variations as well as photometric distortions for few-shot learning through advanced Dynamic Time Warping (DTW) techniques, enhancing the similarity learning of matching action sequences.

Numerous benchmark datasets are available for action recognition, each with its own characteristics, such as view angles and modalities. An analysis of ten recent state-of-the-art algorithms reveals that most methods perform better on cross-subject action recognition than on cross-view action recognition. Skeleton-based features prove to be more robust for cross-view recognition than depth-based features, while deep learning features are suitable for large datasets.

The comparison of recent techniques demonstrates a transition in action recognition from handcrafted features to deep learning models, thanks to the availability of large-scale datasets. However, some domain-specific information remains challenging for deep learning models to capture, which is where traditional methods like Improved Dense Trajectories (IDT) encoded by Bag-of-Words (BoW) / Fisher Vectors (FV) can be beneficial. Therefore, modern CNNs are often fused with IDT encoded by BoW/FV to improve performance. Additionally, global video descriptors like IDT-based BoW/FV can be learned through dedicated CNN-streams during

training and synthesized during testing. Different modalities, such as I3D Optical Flow Features (OFF) and IDT-based BoW/FV, are highly complementary. To simplify the action recognition pipeline, an end-to-end trainable network with streams that learn IDT-based BoW/FV representations during training is proposed. The model can use synthesized BoW/FV representations during testing, simplifying the pipeline and achieving state-of-the-art results.

Building upon previous work on self-supervision, two powerful descriptors are designed and hallucinated: one using popular object detectors applied to training videos and the other using image- and video-level saliency detectors. These descriptors, combined with higher-order statistics, enhance the performance of action recognition, even in fine-grained action recognition tasks.

Inspired by the use of multi-modal inputs and higher-order statistics, tensor representations are introduced to capture the interplay between spatial features and their temporal dynamics. These representations leverage positive definite kernels and higher-order tensors to capture complex action dynamics and relations effectively. The proposed sequence compatibility kernel (SCK) and dynamics compatibility kernel (DCK) capture higher-order correlations and action dynamics, leading to state-of-the-art performance on various benchmarks.

The limitations of current Graph Convolutional Network (GCN)-based models in handling non-connected body joints in action recognition are addressed by proposing a hypergraph model. This model leverages hyper-edges to capture higher-order motion patterns of groups of body joints and achieves state-of-the-art results compared to existing models.

The thesis also discusses the challenge of geometric distortion in matching temporal sequences under varying camera viewpoints. An advanced variant of Dynamic Time Warping (DTW) is proposed, which considers the projective camera geometry to achieve alignment in both temporal and simulated camera viewpoint spaces. A similarity-based loss encourages the alignment of sequences of the same class while preventing the alignment of unrelated sequences, resulting in state-of-the-art results on multiple datasets.

In conclusion, the innovations presented in this thesis address several challenging aspects of action recognition, including geometric and photometric distortions, action dynamics, temporal dependencies, and variations across subjects. These contributions advance the field of human action modeling.

Key words: Action recognition, Self-supervision, Multi-modal fusion, Tensor representations, Transformer, Few-shot learning, Similarity learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Videos capture human actions such as answering a phone call, putting on a hat, throwing a ball, or picking up something from the door, *etc*. Advances in computer vision and machine learning have made it possible to automatically recognize these actions based on the motion of the human body over time. However, robust action recognition remains challenging due to issues such as (i) geometric distortions, *e.g.*, different camera viewpoints, rotations, visual appearances, and variations in human body sizes; (ii) photometric distortions, *e.g.*, scene clutter, occlusions, self-occlusion, scene dynamics, lighting conditions, camera noise, and blur; (iii) dynamics of actions, *e.g.*, varying action speeds; (iv) long-term and short-term motion dependencies; and (v) cross-subject variations.

Despite these challenges, there has been a significant increase in the number of papers on human action recognition in recent years. Action recognition finds applications in various fields, including video surveillance, human-computer interaction, sports analysis, smart homes, and healthcare [Su and Wen, 2022; Qin et al., 2022; Li et al., 2022; Song et al., 2022; Zhang et al., 2021d; Hashiguchi and Tamaki, 2022; Wei et al., 2021b; Zhu et al., 2022; Kong et al., 2022; Jiang et al., 2022; Wang and Koniusz, 2022b,a; Kang et al., 2023; Broomé et al., 2023; Lee et al., 2023; Zhu et al., 2023; Shah et al., 2023; Du et al., 2023; Seon et al., 2023; Noguchi and Tanizawa, 2023; Ahn et al., 2023; Wang and Koniusz, 2023].

This thesis addresses these challenges and contributes to robust human action modeling in the following orthogonal directions: (1) handling geometric and photometric distortions, (2) exploring video-based and skeleton-based feature representations, (3) improving action recognition/classification and few-shot learning for action sequences, (4) investigating self-supervised, supervised, and unsupervised action recognition methods, and (5) comparing uni-modal versus multi-modal approaches in human action recognition. The contributions of this thesis can be summarized as follows:

1. We address temporal variations, geometric distortions, and photometric distortions (*e.g.*, different camera viewpoints of human skeletons, camera noise, *etc*.) by proposing advanced Dynamic Time Warping (DTW) techniques for supervised and/or unsupervised similarity learning when matching pairs of action

Table 1.1: All the benchmark datasets used in this thesis are for human action recognition.

Datasets	Year	Classes	Subjects	#views	#video clips	Sensor	Modalities
UTKinect-Action3D [Xia et al., 2012]	2012	10	10	-	199	Kinect v1	RGB+Depth+3DJoints
Florence3D-Action [Seidenari et al., 2013]	2013	9	10	1	215	Kinect v1	RGB+Depth+3DJoints
MSRAction3D [Li et al., 2010]	2010	20	10	1	567	Kinect v1	Depth+3DJoints
3D Action Pairs [Oreifej and Liu, 2013]	2013	12	10	1	360	Kinect v1	RGB+Depth+3DJoints
CAD-60 [Sung et al., 2011]	2011	14	4	–	68	Kinect v1	RGB+Depth+3DJoints
UWA3D Activity [Rahmani et al., 2014b]	2014	30	10	1	701	Kinect v1	RGB+Depth+3DJoints
UWA3D Multiview Activity II [Rahmani et al., 2016b]	2015	30	9	4	1,070	Kinect v1	RGB+Depth+3DJoints
MPII Cooking Activities [Rohrbach et al., 2012]	2012	64	12	1	3,748	-	RGB
HMDB-51 [Kuehne et al., 2011]	2011	51	-	-	6,766	-	RGB
EPIC-Kitchens [Damen et al., 2018]	2018	149	32	-	39,594	-	RGB+Flow
NTU RGB+D [Shahroudy et al., 2016a]	2016	60	40	80	56,880	Kinect v2	RGB+Depth+3DJoints
Charades [Sigurdsson et al., 2016]	2016	157	-	-	66,500	-	RGB+Flow
NTU RGB+D 120 [Liu et al., 2019a]	2019	120	106	155	114,480	Kinect v2	RGB+Depth+3DJoints
Kinetics-skeleton [Yan et al., 2018]	2017	400	-	-	260,232	-	2DJoints
Kinetics [Kay et al., 2017]	2018	400	-	-	$\sim$ 300,000	-	RGB

sequences. Our methods can be viewed as metric learning-inspired few-shot learning techniques and can be extended to non-episodic learning problems[1].

2. We demonstrate that fusing data from different and complementary modalities, such as optical flow, object/saliency detection information, *etc*., for self-supervised action recognition outperforms uni-modal approaches. Additionally, combining feature representations or finding view-invariant action features from different camera viewpoints enhances the robustness of both video-based and skeleton-based action recognition.

3. To capture short-term and long-term motion dependencies and different dynamics of actions, we model human body kinematics as a hypergraph. We propose a new Transformer architecture that leverages multiple modes, including temporal blocks, channels, and hyper-edges, as well as multiple orders of tensor representations. These innovations result in state-of-the-art performance on several benchmarks, including fine-grained action recognition[2].

In the following sections, we describe existing action recognition benchmarks and evaluation protocols used in this thesis, related works, and the significance of our research.

1.2 Action Recognition Benchmarks and Evaluations

Table 1.1 displays the publicly available benchmark datasets used in the experiments of this thesis for 3D action recognition. Since the introduction of the Microsoft Kinect camera [Zhang, 2012], various research groups have collected datasets for conducting

[1]Episodic training involves organizing training into a series of learning problems, each relying on small support and query sets to simulate the few-shot scenarios encountered during evaluation.

[2]The goal of fine-grained action recognition is to successfully discriminate between action categories with subtle differences, for example, slicing a cucumber *vs*. slicing a tomato.

research on human action recognition and evaluating different algorithms in this field. The Kinect camera is capable of capturing real-time RGB and depth videos, and a publicly available toolkit exists for computing the human skeleton model from each frame of a depth video. Consequently, numerous papers on 3D human action recognition using the Kinect camera have been published [Li et al., 2010; Yang and Tian, 2012; Oreifej and Liu, 2013; Rahmani et al., 2014c, 2016b; Rahmani and Mian, 2016; Wang, 2017; Wang et al., 2019b]. One advantage of using depth videos over conventional RGB videos is the ease of segmenting the foreground human subject even in cluttered scenes. Since depth videos lack color information, the color of the clothing worn by the human subject does not affect the segmentation process. This allows researchers in action recognition to focus more on developing robust feature descriptors for describing actions rather than on low-level segmentation. The algorithm used for computing the 3D joint positions of the human skeletal model with the Kinect toolkit is based on the human skeleton tracking framework (OpenNI) developed by Shotton et al. [2011]. Besides providing access to real-time depth video streams, this tracking framework has also opened up the research area of skeleton-based action recognition [Vemulapalli et al., 2014; Shahroudy et al., 2016a; Vemulapalli and Chellappa, 2016; Ke et al., 2017a,b; Rahmani and Bennamoun, 2017; Wang, 2017; Wang et al., 2019b].

Early benchmark datasets such as MSRAction3D [Li et al., 2010], 3D Action Pairs [Oreifej and Liu, 2013], CAD-60 [Sung et al., 2011], and UWA3D Activity Dataset [Rahmani et al., 2014b] suffer from several limitations: (i) They have a limited number of action classes, and each action class can be easily distinguishable by a simple motion pattern or even the appearance of an interacted object when only a very small number of action classes are available. (ii) These datasets have a limited number of performing subjects and a very narrow range of performers' ages, which limits the intra-class variation of actions. (iii) The camera views in these datasets are highly restricted, with almost all samples captured from a front view with a fixed camera viewpoint. Other small datasets, such as the UWA3D Multiview Activity II dataset [Rahmani et al., 2016b], are captured using multiple cameras simultaneously, but the views are limited to fixed front and side views. (iv) These datasets also have a highly limited number of video samples, which prevents the application of more advanced deep learning methods in this field. To address these limitations, many large-scale benchmark datasets have been introduced, including NTU RGB+D [Shahroudy et al., 2016a] and NTU RGB+D 120 [Liu et al., 2019a].

However, the research community is currently addressing only a part of the overall action recognition problem, and several limiting factors still exist in benchmark datasets: (i) While large-scale datasets with a high number of actions/activities do exist, the typical inter-class variability is often high, which can be rather unrealistic for surveillance or elderly care applications, especially when we need to differentiate between very similar activities. (ii) Many of the considered actions or activities are rather coarse-grained, mainly focusing on full-body activities, such as 'open a box' and 'throw up cap/hat' in NTU RGB+D 120. These activities may appear atypical for many applications where we aim to differentiate between more fine-grained actions

or activities, such as 'cut', 'squeeze', 'peel', *etc.* (iii) Many benchmarks solely address action/activity classification without delving into more realistic and challenging actions/activities in a continuous data stream. To address these shortcomings, some fine-grained action/activity recognition datasets have been released, including MPII Cooking Activities [Rohrbach et al., 2012] and EPIC-Kitchens [Damen et al., 2018].

To further facilitate research in human action recognition, more large-scale, high-quality datasets have been introduced, including Charades [Sigurdsson et al., 2016] and Kinetics [Kay et al., 2017]. These datasets cover a diverse range of human actions, with: (i) Considerable camera motion/shake, illumination variations, shadows, background clutter, *etc*. (ii) A wide variety of performers with differences in how the action is performed, including speed, clothing, body shape, camera framing, viewpoint, *etc*. Although these large-scale datasets provide only raw videos and do not include the modality of 3D human body joints/human skeletons, numerous skeleton pose estimation methods are available, such as the Kinect toolkit with the OpenNI framework [Shotton et al., 2011] and OpenPose [Cao et al., 2017], which can be used for 3D and 2D skeleton estimation. For instance, Kinetics-skeleton [Yan et al., 2018] in 2D was produced using OpenPose with 18 human body joints. With the availability of these large-scale datasets, new generations of neural network architectures, and architectures with attention mechanisms, are being developed. This includes the use of multiple streams of information, such as RGB, optical flow, human poses/skeleton sequences, object category recognition, *etc*.

As different problem definitions have been formulated in various application fields, action recognition techniques are evaluated in different ways. Our work focuses on action recognition, which involves recognizing action concepts and few-shot learning (*e.g.*, learning similarities between pairs of action sequences). For the task of action classification or few-shot action recognition, the recognition accuracy of an algorithm for any given action class is defined as the proportion of correct class labels returned by the algorithm. To demonstrate the recognition accuracies of an algorithm for all the action classes, a confusion matrix is often used. The overall performance of an algorithm on a given dataset is evaluated using the average recognition accuracy. Below, we discuss the commonly used evaluation metrics.

Early action recognition methods, such as those in [Oreifej and Liu, 2013; Rahmani et al., 2014c; Vemulapalli and Chellappa, 2016; Rahmani et al., 2016b; Wang, 2017; Wang et al., 2019b], are usually evaluated on small-scale datasets, such as MSRAction3D, 3D Action Pairs, CAD-60, and UWA3D Activity Dataset. The confusion matrix is typically adopted to display the recognition accuracy of each action category. For some challenging datasets, such as MPII Cooking Activities, EPIC-Kitchens, and Charades, the action classification performance is assessed using the standard mean average precision (mAP) measure. Additionally, the average precision, which approximates the area under the precision-recall curve, is calculated for each individual action class.

Most deep learning-based approaches, such as those in [Simonyan and Zisserman, 2014; Tran et al., 2015; Carreira and Zisserman, 2017; Yan et al., 2018], are generally evaluated on datasets like HMDB-51, NTU RGB+D, and NTU RGB+D 120. Therefore, they can only report the overall recognition performance (*i.e.*, the average recognition

accuracy) on each dataset.

The strong performance of deep learning models, however, heavily relies on training a neural network with abundant labeled instances that exhibit diverse visual variations. For example, thousands of samples for each new class are often required, and pre-training on large-scale datasets with base classes is often necessary. Moreover, the human annotation cost, as well as the scarcity of data in some classes (*e.g.*, very rare actions), significantly limit the applicability of the current vision system to efficiently learn new visual concepts. Inspired by the human visual system's ability to recognize new classes with extremely few labeled samples, few-shot learning/classification, which aims to generalize to new classes with a limited amount of labeled samples for each novel class, has attracted considerable attention in many areas, including few-shot action recognition.

The most common/standard settings used in few-shot action recognition are 5-way 1-/5-shot classification. In these settings, either 1 or 5 labeled instances are available from each novel action class. This is commonly used on datasets like HMDB-51 and Kinetics. An one-shot evaluation protocol was introduced in [Liu et al., 2019a] for NTU RGB+D 120. The full dataset is split into the auxiliary set and the evaluation set, with no overlaps of action classes between these two sets. The evaluation set consists of the novel action classes for one-shot evaluation, and one sample from each novel class is chosen as the exemplar, while the remaining samples are used to test the performance. Based on this evaluation protocol, we created similar one-shot protocols for NTU RGB+D (also known as NTU-60) and some small-scale datasets (MSRAction3D, 3D Action Pairs, and UWA3D Activity Dataset) in Appendix B.

Below, we review action recognition models in the literature related to our work.

1.3 Action Recognition on Videos

There has been significant progress in action recognition, spanning from traditional methods, such as DT [Wang et al., 2011] and IDT [Wang et al., 2013b], to the latest advancements employing deep learning techniques, such as two-stream networks [Simonyan and Zisserman, 2014], 3D spatio-temporal features [Tran et al., 2015], and spatio-temporal ResNet models [Feichtenhofer et al., 2016a]. Deep learning, renowned for its scalability and capacity to efficiently process vast amounts of data, has revolutionized video classification by enabling the utilization of large-scale datasets and powerful models to establish standard representations for video understanding tasks.

At the forefront of video understanding lies action recognition, the primary task of which is to identify various actions within videos. Key to successful action recognition are spatio-temporal and motion information, where the former captures spatial relationships across different video frames, while the latter encapsulates the motion between consecutive frames of objects or human subjects. Recent years have witnessed significant advancements in action recognition performance. Many existing methods, such as [Ji et al., 2013; Carreira and Zisserman, 2017; Wang et al., 2018b; Xie

et al., 2018; Wang et al., 2019], approach this problem as a generic classification task. The distinguishing factor from ImageNet is that the input is now a video sequence. Consequently, substantial efforts have been directed towards harnessing temporal information. Human activities, in contrast to objects in ImageNet, involve complex concepts characterized by factors like human body movement, temporal dynamics, and human-object interactions. This complexity implies that the modeling of visual and temporal features often results in confusion between action classes that share similar visual appearances and motion dynamics. A common modeling approach involves the widely-used classifier head consisting of global average pooling[3] and a single linear classifier, mirroring the setup of object recognition models on ImageNet. For fine-grained action recognition, distinguishing confused classes may necessitate capturing finer-grained motion patterns.

Early deep learning-based action recognition algorithms can be categorized into two main groups: two-stream neural networks [Feichtenhofer et al., 2016b, 2017a; Wang et al., 2017] and 3D convolutional networks (3D CNNs) [Tran et al., 2015; Carreira and Zisserman, 2017]. Two-stream neural networks comprise a spatial stream, which takes RGB frames as input, and a flow stream that uses optical flow as input. The spatial stream models appearance features without considering temporal information, while the flow stream (often referred to as a temporal stream) is designed to capture temporal information. However, the flow stream primarily represents motion features between consecutive frames and shares a nearly identical structure with the spatial stream, utilizing 2D CNNs. This leads to a lack of long-term temporal relationship modeling in the flow stream, not to mention the computational cost of optical flow calculation.

On the other hand, 3D CNNs are designed to capture spatio-temporal features by combining spatial and temporal features using 3D convolutions. While they can capture long-term temporal relationships, they also introduce significant computational costs. Many methods still integrate an independent optical flow stream to further enhance performance with motion information. Thus, these two sources of information complement each other in action recognition. However, these methods often result in a high number of parameters, leading to computational burdens. Some approaches [Sun et al., 2015; Qiu et al., 2017; Tran et al., 2018; Xie et al., 2018] aim to reduce costs by decomposing 3D convolutional kernels into separate spatial and temporal components.

Nevertheless, these methods suffer from two main limitations: (i) they require the pre-computation of optical flow, which is costly, and (ii) the features learned and the final predictions from multiple streams are simply fused using weighted or average sums, making them inferior to temporal relationship modeling. Almost all these works retain the same final layers of the network, which consist of global average pooling followed by a fully connected layer.

Recent advanced action recognition models include the two-stream I3D network [Carreira and Zisserman, 2017], SlowFast networks [Feichtenhofer et al., 2019],

[3] A pooling operation computes the average value for patches of a feature map and utilizes this information to generate a downsampled or pooled feature map.

AssembleNet [Ryoo et al., 2020b], its extended version AssembleNet++ [Ryoo et al., 2020a], video masked autoencoders (VideoMAE) [Tong et al., 2022], Intern-Video [Wang et al., 2022], and transformer-based models such as multiscale vision transformers (MViT) [Fan et al., 2021], video vision transformer (ViViT) [Arnab et al., 2021], TimeSFormer [Bertasius et al., 2021], Recurrent Vision Transformer (RViT) [Yang et al., 2022a], Uncertainty-Guided Probabilistic Transformer (UGPT) [Guo et al., 2022], DirecFormer [Truong et al., 2022], and Video swin transformer (VideoSwin) [Liu et al., 2022]. Vision-language models like CLIP [Radford et al., 2021], X-CLIP [Ni et al., 2022], and ViT-L [Wu et al., 2023] have also played a significant role.

In this thesis, our work explores the self-supervision capacity of valuable yet costly auxiliary descriptors. We integrate them with existing state-of-the-art action recognition architectures to further enhance performance, robustness, and the simplification of modern action recognition pipelines. This integration is achieved in an end-to-end trainable manner.

1.4 Action Recognition on Skeletons

Diverging from video-based action recognition methods, which predominantly concentrate on modeling spatial and temporal features from RGB frames and/or optical flow videos, skeleton-based action recognition methods offer immunity against challenges like background clutter, illumination fluctuations, and appearance variations. The 3D skeleton data comprises a collection of 3D coordinates corresponding to human body joints, providing a robust representation that facilitates the modeling of distinctive temporal aspects of human actions. The accessibility of real-time video streams [Shotton et al., 2011] and advanced human pose estimation algorithms such as OpenPose [Cao et al., 2017] have made it significantly more feasible to acquire skeleton data. Skeleton sequences are built upon two fundamental principles: (i) the structural connectivity of a skeletal graph and (ii) the temporal continuity of each 3D body joint as it evolves over time. While the temporal evolution of individual body joints yields highly informative data, the embeddings of individual joints may fail to capture relationships between 3D body joints effectively. Additionally, while modeling connections between adjacent 3D body joints based on structural connectivity proves informative, these pairs of body joints often exhibit strong correlations in terms of their temporal evolution.

Early attempts in skeleton-based action recognition often encoded all human body joint coordinates in each frame into a feature vector for pattern learning [Wang et al., 2019b]. The advent of real-time video streams and advanced human estimation algorithms also paved the way for handcrafted skeletal action recognition research [Rahmani et al., 2014c; Vemulapalli et al., 2014; Shahroudy et al., 2016a; Vemulapalli and Chellappa, 2016; Koniusz et al., 2016a]. These models typically did not explore the internal motion dependencies between body joints, resulting in the neglect of abundant and rich actional motion dynamics. To address this limitation, recent methods construct a skeleton graph with joints as vertices and bones as edges,

employing deep learning-based models to extract correlated features [Yan et al., 2018; Cheng et al., 2020b; Shi et al., 2021a].

Graph-based models represent one of the state-of-the-art approaches for skeleton-based action recognition, leveraging their effectiveness in handling graph-structured data [Wu et al., 2020]. Existing graph-based models primarily vary in their treatment of temporal information, with Graph Neural Networks (GNNs) encoding spatial neighborhood information followed by aggregation using Recurrent Neural Networks (RNNs). Alternatively, Graph Convolutional Networks (GCNs) perform spatio-temporal convolutions within the neighborhood of each node. Spectral GCNs[Kipf and Welling, 2017] operate in the spectral domain, while spatial GCNs conduct convolutions within a one- or few-hop radius of each node. For instance, the spatio-temporal GCN model known as ST-GCN [Yan et al., 2018] captures the spatio-temporal vicinity of each 3D body joint.

However, ST-GCN applies convolution along structural connections (edges between body joints), overlooking joints that are structurally distant, which may hold crucial patterns for action recognition. ST-GCN can aggregate even larger neighborhoods as the number of layers increases, but this can lead to oversmoothing issues. Moreover, convolution along structural links does not consider dependencies between physically distant 3D body joints.

Recent GCN-based models encompass the Attention-enhanced Graph Convolutional LSTM network (AGC-LSTM) [Si et al., 2019], Actional-Structural GCN (AS-GCN) [Li et al., 2019], Dynamic Directed GCN (DDGCN) [Korban and Li, 2020], Decoupling GCN with DropGraph module [Cheng et al., 2020a], Shift-GCN [Cheng et al., 2020b], Semantics-Guided Neural Networks (SGN) [Zhang et al., 2020c], AdaSGN [Shi et al., 2021a], Context-Aware GCN (CA-GCN) [Zhang et al., 2020e], Channel-wise Topology Refinement Graph Convolution (CTR-GC) [Chen et al., 2021b], InfoGCN [Chi et al., 2022], and a family of Efficient GCN (EfficientGCN-Bx) [Song et al., 2022].

To capture more complex spatio-temporal motion dynamics, some models represent the human body as a hypergraph by modeling larger groups of 3D body joints as hyper-edges. A hypergraph was introduced in [Liu et al., 2020a] to represent 3D human body joints, exploiting kinematic constraints among adjacent and non-adjacent joints with a semi-dynamic hypergraph neural network. This approach captures richer information than traditional GCNs. Hypergraph GNNs [Hao et al., 2021] capture both spatio-temporal information and higher-order dependencies for skeleton-based action recognition.

Recent transformer-based models for skeletal action recognition include Space-Time Transformer [Zhang et al., 2021b], spatial and temporal transformer (ST-TR) [Plizzari et al., 2021], spatial-temporal specialized transformer (STST) [Zhang et al., 2021e], Hyperformer [Zhou et al., 2022], graph-aware transformer (GAT) [Zhang et al., 2022b], Focal and Global Spatial-Temporal Transformer network (FG-STFormer) [Gao et al., 2022], multi-scale temporal transformer (MTT) [Kong et al., 2022], and graph skeleton transformer network (GSTN) [Jiang et al., 2022].

In this thesis, our approach leverages higher-order tensor representations and multi-order hyper-edge features extracted from the skeletal hypergraph. We employ

a newly proposed multi-order multi-mode transformer to capture higher-order relationships between features, long-term and short-term temporal dependencies, and spatio-temporal correlations, enabling robust action recognition.

1.5 Multi-modal and Multi-view Action Recognition

Action recognition from multiple data modalities. Multi-modal learning, a pivotal area of research, revolves around the process of acquiring knowledge from diverse data modalities. This includes visual data such as images and videos, along with their fusion with other modalities like audio, speech, and text. This fusion, accomplished through sophisticated processing algorithms, aims to enhance performance [Koniusz et al., 2016a,b; Wang et al., 2019b,d; Koniusz et al., 2020; Wang and Koniusz, 2021]. Multi-modal learning capitalizes on the synergies between these different modalities, often surpassing the capabilities of single-modal (or uni-modal) learning in various real-world applications, including human action recognition [Radford et al., 2021; Ni et al., 2022; Wu et al., 2023]. In recent times, multi-task learning (MTL) [Caruana, 1997] has gained substantial traction within the computer vision community. MTL, while distinct, shares a conceptual framework closely aligned with multi-modal learning. Unlike single-task learning, MTL seeks to develop a shared representation capable of addressing multiple tasks, ultimately leading to improved generalizability. Both multi-modal learning and MTL capitalize on the idea of concurrently learning the structural aspects that benefit all tasks, thereby enhancing performance. Typically, MTL leverages knowledge gained from auxiliary tasks to bolster the performance of the primary task [Wang et al., 2019d; Wang and Koniusz, 2021].

Several early action recognition methods [Rahmani et al., 2014c; Shahroudy et al., 2016b; Rahmani and Bennamoun, 2017; Elmadany et al., 2018] have explored the integration of complementary data modalities. Some handcrafted methods combine depth and skeleton features to achieve robust human action recognition. For instance, Rahmani et al. [2014c] proposed a fusion of four types of local features extracted from both depth images and 3D joint positions to address local occlusions and enhance recognition accuracy. Shahroudy et al. [2016b]'s approach combines Local Occupancy Patterns (LOP), HON4D [Oreifej and Liu, 2013], and skeleton-based features with hierarchical mixed norms to regularize weights across modality groups for different body parts. Rahmani and Bennamoun [2017] utilized an end-to-end deep learning model for learning body part representations from skeletal and depth images, further enhancing their model with a bilinear compact pooling layer for generated depth and skeletal features. Elmadany et al. [2018] employed canonical correlation analysis to maximize feature correlations from various sensors, including bag-of-angles from skeleton data, depth motion maps from depth video, and optical flow from RGB video. They learned a shared subspace for all these features, and average pooling yielded the final feature descriptor. Recent deep learning-based models, such as the two-stream I3D [Carreira and Zisserman, 2017], AssembleNet [Ryoo et al., 2020b], and its extended version AssembleNet++ [Ryoo et al., 2020a], integrate RGB and optical flow inputs and

leverage late fusion (next to the classifier) with low-level representations like Improved Dense Trajectory (IDT) descriptors [Wang and Schmid, 2013]. These models benefit from the highly complementary nature of these representations[Fernando and Gould, 2016; Cherian et al., 2017b, 2018; Wang and Cherian, 2018; Choutas et al., 2018; Wang et al., 2019d; Wang and Koniusz, 2021]. Recent advancements in action recognition include multi-modal CNN-GCN models and multi-modal feature representations for improved relational action predictions [Shi et al., 2021b], multi-modal domain adaptation for fine-grained action recognition [Munro and Damen, 2020], cross-modal knowledge for domain-adaptive action recognition [Yang et al., 2022b], and vision-language models like CLIP [Radford et al., 2021], X-CLIP [Ni et al., 2022], ViT-L [Wu et al., 2023], Video-Audio-Text Transformer (VATT) [Akbari et al., 2021], and Modality Mixer (M-Mixer) [Lee et al., 2023].

In this thesis, we delve into the realm of multi-modal information utilization, building upon the self-supervision concept. Here, we take RGB frames as input and employ a multi-task learning (MTL) framework to predict action concepts and complementary auxiliary descriptors (*e.g.*, IDT-based Bag of Words/Fisher Vectors, optical flow features, object/saliency detection features). This approach enhances robust action recognition.

Action recognition from multiple camera viewpoints. In practice, capturing human actions from arbitrary camera viewpoints presents a significant challenge for the development of efficient action recognition techniques. A major obstacle in action recognition stems from the substantial variations in action representations when observed from different viewpoints [Wang, 2017; Wang et al., 2019b]. These variations arise due to flexible camera viewpoints that result in markedly different feature representations, even for the same scene. Additionally, actors may execute actions in various orientations, dynamically altering their positions over time. Consequently, the appearance and dynamics of an action can differ significantly when viewed from distinct angles.

Addressing viewpoint variance has been a dynamic area of research, especially within human action recognition. Furthermore, it has become increasingly crucial in the context of representation learning. Large-scale datasets such as Sports1M [Karpathy et al., 2014], Kinetics [Carreira and Zisserman, 2017], Something-something [Goyal et al., 2017], ActivityNet [Caba Heilbron et al., 2015], and Charades [Sigurdsson et al., 2016], incorporating multiple modalities like RGB videos, depth videos, skeleton sequences, optical flow videos, among others, have fueled progress in multi-view action recognition [Zhang et al., 2017b; Wang et al., 2019a; Vyas et al., 2020; Zhang et al., 2019d; Shah et al., 2023]. A prevailing approach in these methods revolves around the learning of view-invariant feature representations. Several works leverage multiple views of the subject performing the action [Shahroudy et al., 2016a; Liu et al., 2019a; Zhang et al., 2019d; Wang et al., 2019d] to mitigate viewpoint variations in action recognition tasks with extensive datasets.

For video-based action recognition, researchers have devised various view-invariant techniques [Bashir et al., 2006; Junejo et al., 2008; Weinland et al., 2010; Liu et al., 2011a; Li and Zickler, 2012; Wu and Jia, 2012; Mahasseni and Todorovic, 2013; Zhang

et al., 2013; Rahmani and Mian, 2015; Wu et al., 2015]. Early methods explored view-invariant features such as self-similarity descriptors or descriptors based on trajectory curvature [Bashir et al., 2006; Junejo et al., 2008]. However, these descriptors, when presented in another domain, might lose valuable information from the original video data. Some approaches [Weinland et al., 2010; Mahasseni and Todorovic, 2013; Wu et al., 2015] require multiple views to train a 'panorama' model, for instance, using the 3D histogram of oriented gradients based on Bag of Words (BoW) to enhance robustness against viewpoint changes [Weinland et al., 2010]. Unfortunately, capturing videos from numerous camera viewpoints is cost-prohibitive. Other approaches employ knowledge transfer-based models to find a latent space for direct comparison of features from different viewpoints [Li and Zickler, 2012; Liu et al., 2011a; Rahmani and Mian, 2015; Zhang et al., 2013]. Nevertheless, these methods necessitate substantial human effort in designing the view-independent latent space. Zhang et al. [2019d] proposed adaptive neural networks featuring view adaptation modules. These modules learn the most suitable observation viewpoints, enabling the transformation of skeletons to those viewpoints for end-to-end action recognition.

In skeleton-based action recognition, achieving view-invariance often involves frame-level preprocessing [Xia et al., 2012; Vemulapalli et al., 2014; Du et al., 2015; Liu et al., 2016; Shahroudy et al., 2016a; Vemulapalli and Chellappa, 2016; Zhu et al., 2016; Liu et al., 2017; Li et al., 2017b; Song et al., 2017]. However, this approach may result in the partial loss of relative motion information by transforming per-frame human skeleton data into a fixed upper-body orientation. On the other hand, sequence-level preprocessing applies the same transformation to all frames, ensuring motion invariance to the initial body position and orientation, thereby preserving motion information. Yet, defining the body plan (*e.g.*, using hip, shoulder, and neck joints) may not always be suitable for orientation alignment, especially when a sequence lacks an upright pose [Wang et al., 2012]. Recent methods employ either view adaptation models to learn suitable camera viewpoints and transform human skeletons to these views for each sequence [Zhang et al., 2019d] or advanced Dynamic Time Warping techniques to simultaneously model the best alignment in both temporal and simulated camera viewpoint spaces for end-to-end learning [Wang and Koniusz, 2022a].

Recent models for multi-view action recognition encompass the Generative Multi-View Action Recognition (GMVAR) framework [Wang et al., 2019a], an unsupervised representation learning framework [Vyas et al., 2020], a Cross-view Contrastive Learning framework (CrosSCLR) [Li et al., 2021a] for unsupervised 3D skeleton-based action representation, and a supervised contrastive learning framework [Shah et al., 2023].

In this thesis, our approach tackles the simultaneous temporal and simulated viewpoint alignment within an end-to-end meta-learning framework. This novel paradigm centers around similarity learning of support-query pairs rather than conventional class concept learning.

1.6 One- and Few-shot Action Recognition

Traditional supervised learning methods rely on extensive labeled data for training, and the test set typically consists of samples from the same categories as the training set, following a similar statistical distribution [Carreira and Zisserman, 2017; Ryoo et al., 2020b,a]. However, emerging few-shot learning offers compelling solutions to address these challenges: (i) Few-shot learning leverages only a limited number of labeled samples, avoiding the need for massive volumes of costly labeled data during model training. This approach enhances generalization capabilities. (ii) One of its advantages is that it eliminates the requirement to retrain a model entirely when extending it to new categories of data, resulting in significant computational savings. (iii) Few-shot learning empowers models to gain insights into rare categories of data with exposure to only limited prior information. (iv) It exhibits versatility by extending its application to other data domains, as long as the data in the support and query sets exhibit coherence, even when the model has been pre-trained on a statistically different data distribution.

The principles of one- and few-shot learning find applications in various domains, including natural language processing and computer vision [Miller et al., 2000; Li et al., 2002; Fink, 2005; Bart and Ullman, 2005; Fei-Fei et al., 2006; Lake et al., 2011; Mishra et al., 2018; Xu et al., 2018; Guo et al., 2018; Dwivedi et al., 2019; Zhang et al., 2020a; Cao et al., 2020; Guo et al., 2020; Dvornik et al., 2020; Wang et al., 2020; Lichtenstein et al., 2020; Fei et al., 2020; Guan et al., 2020; Li et al., 2020b; Elsken et al., 2020; Cao et al., 2020; Tang et al., 2020; Koniusz and Zhang, 2020; Simon et al., 2020b; Yu et al., 2020; Luo et al., 2021; Zhang et al., 2021a, 2022a; Zhu and Koniusz, 2022; Lu and Koniusz, 2022; Wang and Koniusz, 2022b; Zhang et al., 2022d,c]. One of the most promising frontiers in video processing tasks is the realm of meta-learning, where the aim is to acquire transferable knowledge from a diverse array of sampled tasks or episodes. This approach is invaluable in enhancing generalization capabilities and mitigating the risks of over-fitting. Drawing inspiration from metric learning principles, several established methods in the realm of few-shot video classification have emerged. These methods assess the similarity between different videos within the feature space, thereby facilitating classification tasks. Prominent examples of these methods include MatchingNet [Vinyals et al., 2016], ProtoNet [Snell et al., 2017], and RelationNet [Sung et al., 2018]. MatchingNet, for instance, gauges the cosine distance between the query feature and each support feature and subsequently computes the average cosine distance for each class. ProtoNet, on the other hand, measures the Euclidean distance between query features and the class mean of support features. While RelationNet shares a similar underlying concept, it distinguishes itself by replacing distance calculations with a learnable relation module. Furthermore, an innovative approach known as absolute-relative learning [Zhang et al., 2021a] has been proposed. This approach incorporates both similarity and class concept learning to harness the full potential of label information in both supervised and unsupervised scenarios, particularly in the context of image classification.

Video-based. In the realm of video-based tasks, a fundamental distinction between

images and videos lies in the additional temporal dimension. Consequently, representing an entire video as a singular feature vector, for instance, through temporal pooling, proves inadequate. To address this challenge, various temporal alignment methods have been devised to capture the critical temporal information. Cao et al. [2020] introduced an ordered temporal alignment module designed to learn a profound distance metric for the query video concerning novel class proxies within the support set, thereby establishing an alignment path. Zhang et al. [2021c] proposed an implicit temporal alignment network grounded in a self-attention mechanism. This innovative approach integrates spatial and feature channel context, enabling more effective modeling of intra-class variations. Bishay et al. [2019] devised a temporal attentive relation network that capitalizes on attention mechanisms for temporal alignment. This network is instrumental in acquiring a deep-distance measure at the video segment level, catering to zero- and few-shot action recognition scenarios. In [Li et al., 2021c], a two-stage action alignment network was introduced. In the first stage, it identifies the action by learning a temporal affine transform, while the second stage orchestrates the query feature to align with the spatio-temporal action evolution of the support by employing temporal rearrangement and spatial offset prediction. Another noteworthy approach, detailed in [Li et al., 2021b], employs event boundary information to guide the temporal alignment of features in the context of few-shot action recognition. In [Wang et al., 2021a], a temporal relation-based attentive prototype network was proposed for few-shot action recognition within videos. This method incorporates a spatio-temporal motion enhancement module to emphasize object motions in videos. Subsequently, a temporal relation module captures both short- and long-term temporal scales for relations, and an attentive prototype is introduced to assign greater weight to discriminative samples. Furthermore, a neural graph matching network [Guo et al., 2018] was developed to leverage the inherent structure of 3D data through graph representation, specifically for few-shot 3D action recognition within videos. In this approach, nodes in the interaction graph are derived from sources such as human-annotated object and pose detections or pre-trained object and pose detectors. Additionally, edge generation is learned in conjunction with the graph matching metric to ensure differentiability and robustness in the task.

A comprehensive examination of the limitations in representation learning within the domain of metric-based methods is available in [Chen et al., 2019]. Intriguingly, their findings reveal that a straightforward classifier-based baseline, devoid of any temporal alignment, can outperform the meta-learning approaches. In addition to these methodologies, alternative models have been proposed. For example, temporal attention vectors, as described by Bo et al. [2020], adapt to videos of varying lengths while preserving the temporal context of the entire video. Furthermore, Zhang et al. [2020a] introduced permutation-invariant pooling techniques tailored for action recognition videos, accommodating diverse action lengths and long-range temporal dependencies. An innovative approach involves the utilization of the CrossTransformer attention mechanism, as presented by Perrett et al. [2021], to define class prototypes based on relevant subsequences extracted from all support videos. This contrasts with conventional methods that rely on class averages or single best matches.

Most existing approaches follow a meta-learning paradigm with episodic training. In each episode, a small subset of samples is partitioned into support and query sets. These sets are subsequently employed to construct a classifier, which is then assessed using a query-centered loss for model refinement. However, two significant limitations persist. Firstly, there is a deficiency in data efficiency attributable to the design of a query-centered loss[4], as outlined in [Zhu et al., 2021b]. Secondly, these methods struggle to handle outlier samples, such as those involving unconventional viewpoints or occlusions, as well as challenges posed by overlapping inter-class distributions. For example, training samples from distinct classes may exhibit similar backgrounds or visual characteristics in the support set. Consequently, Zhu et al. [2021b] proposed a prototype-centered attentive learning model to tackle these aforementioned limitations within the context of few-shot action learning. Moreover, a recent advancement introduced an action-appearance alignment meta-adaptation module, as outlined in [Patravali et al., 2021]. This module is designed to concentrate on action-oriented video features in relation to appearance features through explicit few-shot episodic meta-learning over carefully selected episodes. It aims to enhance the performance of unsupervised few-shot action recognition.

Skeleton-based. Skeleton-based action recognition has seen significant advancements, driven by pose representations that offer a robust foundation for understanding human actions [Wang, 2017; Wang et al., 2019b]. Nevertheless, downstream algorithms reliant on pose information continue to grapple with unreliable estimations, especially in fine-grained actions characterized by subtle differences between action categories, such as distinguishing between actions like cut and peel. With a scarcity of labeled data, tasks like fine-grained action recognition demand models that can discern actions with subtle discrepancies while accommodating noisy inputs.

Few-shot learning holds the potential to swiftly adapt to novel classes even with limited annotations. However, its application to skeleton-based action recognition remains relatively unexplored due to several challenges, including photometric and geometric distortions. While a few models venture into few-shot action recognition using 3D skeletons [Liu et al., 2017; Liu et al., 2019a; Memmesheimer et al., 2020, 2021], each takes a unique approach. For instance, the Global Context-Aware Attention LSTM [Liu et al., 2017] selectively focuses on informative joints. The Action-Part Semantic Relevance-aware (APSR) model [Liu et al., 2019a] leverages semantic relevance between body parts and action classes at the distributed word embedding level. Signal Level Deep Metric Learning (DML) [Memmesheimer et al., 2020] and Skeleton-DML [Memmesheimer et al., 2021] encode signals into images, extract features using CNNs, and employ multi-similarity miner losses for one-shot learning on datasets like NTU RGB+D 120. Sabater et al. [2021] propose a robust motion representation capable of handling varying kinematic conditions in skeletons for one-shot action recognition. In

[4]When you have K samples available for each of the N classes in the support set, the initial $N \times K$ samples are condensed into N prototypes. Following this, a loss term is calculated for each query sample individually, based on its distances to these prototypes. However, this approach doesn't take into account how the samples in the query set as a whole should be distributed. Consequently, it fails to fully leverage the limited training data available in each episode.

another approach, Hong et al. [2021] introduce a weakly supervised video pose distillation method for fine-grained sports action recognition, utilizing a teacher-student network where the teacher network generates view-invariant pose representations and additional motion signals to guide the student network in generating rich visual patterns from color video frames and optical flow. Guo et al. [2018] present Neural Graph Matching (NGM) networks, designed to recognize previously unseen 3D action classes with only a few examples by harnessing the inherent structure of 3D data through a graphical representation.

Recent advancements in one- and few-shot skeletal action recognition encompass a part-aware prototypical representation [Chen et al., 2022] for one-shot skeletal action recognition, models addressing Few-Shot Open-Set learning for sequences of 3D skeletons [Berti et al., 2022], the Disentangled and Adaptive Spatial-Temporal Matching (DASTM) model [Ma et al., 2022] for few-shot action recognition, the self and mutual adaptive matching network (SMAM-Net) [Li et al., 2023] for both one- and few-shot action recognition, a novel transformer-based model for Skeleton-based One-shot Action Recognition (Trans4SOAR) [Peng et al., 2023], and an Adaptive Local-Component-aware Graph Convolutional Network (ALCA-GCN) [Zhu et al., 2023].

Nevertheless, these methods often overlook critical considerations: (i) accounting for observation noise and uncertainty in temporal sequences (frame-wise or temporal block-wise feature representations) stemming from photometric distortions like camera noise, and (ii) addressing geometric distortions such as varying camera viewpoints, rotations, and visual appearances. Dynamic Time Warping (DTW) [Cuturi, 2011], a method popular in forecasting time series evolution, estimating Fréchet mean of time series, or classifying actions, proves adept at overcoming issues related to temporal alignment in matching action sequence pairs for few-shot learning. Key to DTW is its sequence matching transportation plan, enabling matched sequences to progress at varying speeds, both globally and locally in the temporal domain. While DTW is non-differentiable, a differentiable soft variant known as soft-DTW [Cuturi and Blondel, 2017] enables backpropagation.

In this thesis, we build upon soft-DTW, proposing advanced DTW techniques for alignment-based few-shot skeletal action recognition. The advanced DTW techniques proposed in this thesis extend beyond temporal alignment to account for (i) the uncertainty in observations of temporal sequences due to photometric distortions like camera noise and (ii) the alignment of temporal and camera viewpoints to handle geometric distortions such as variations in camera perspectives and rotations.

In contrast, non-alignment methods often employ approaches like average pooling along the temporal dimension, which may result in the loss of temporal information [Liu et al., 2017; Liu et al., 2019a], or encoders that capture short-term dependencies while disregarding long-term ones [Memmesheimer et al., 2020, 2021; Ben-Ari et al., 2021]. Recognizing the importance of temporal alignment in addressing non-linear temporal variations, recent works introduce various alignment-based models, such as permutation-invariant spatio-temporal attention reweighted distance in ARN [Zhang et al., 2020a], a variant of DTW utilized in OTAM [Cao et al., 2020], a

temporal attentive relation network [Mina et al., 2019], a two-stage temporal alignment network (TA2N) [Li et al., 2021c], a temporal CrossTransformer [Perrett et al., 2021], and a learnable sequence matching distance named TAP [Su and Wen, 2022].

1.7 Thesis Outline and Contributions

In this section, we provide a brief description of each thesis chapter. Chapter 2 offers a comparative review of recent state-of-the-art action recognition algorithms. Chapter 3 and 4 present the hallucination of IDT-based BoW/FV representations and object/saliency detection features for self-supervised action recognition, respectively. Chapter 5 discusses tensor representations for capturing higher-order relationships between visual features and temporal dynamics, while Chapter 6 presents the aggregation of embeddings of multi-order hyper-edges of skeletal hypergraph using Multi-order Multi-mode Transformer (3Mformer). Chapter 7 covers temporal alignment with uncertainty modeling, and Chapter 8 addresses joint temporal-viewpoint alignment for few-shot skeletal action recognition. Finally, our work concludes in Chapter 9. Our contributions in this thesis are listed below:

—**Chapter 2: A Comparative Review.** This chapter conducts a comparative review of 10 recent state-of-the-art action recognition algorithms. It specifically focuses on comparing the effectiveness of using handcrafted features versus deep learning features and skeleton-based features versus depth-based features. Additionally, we evaluate the cross-view versus cross-subject (single-view) performance of these algorithms. For the multiview datasets, we investigate the impact of the camera view on human action recognition, considering whether the features used are depth-based, skeleton-based, or depth+skeleton-based. To the best of our knowledge, such a comparison and evaluation have not been performed before.

—**Chapter 3: Hallucinating IDT Descriptors and I3D Optical Flow Features.** In this chapter, we propose for the first time that old-fashioned IDT-based Bag of Words (BoW) and Fisher Vector (FV) global video descriptors can be learned through dedicated Convolutional Neural Network (CNN) streams during the training stage and then simply hallucinated for classification using a CNN action recognition pipeline during testing. We also demonstrate that even the I3D optical flow stream can be easily hallucinated from the I3D RGB stream. Our pipeline leverages self-supervision, where IDT-based BoW/FV descriptors provide easily obtainable self-information about videos. Additionally, it utilizes Multi-task Learning (MTL), known for boosting generalization and preventing overfitting of CNNs due to task-specific losses.

—**Chapter 4: Statistical Moment and Subspace Descriptors.** In this chapter, we propose utilizing object and human detectors to enhance the performance of action recognition pipelines. We design two types of compact descriptors called Object Detection Features (ODF) and Saliency Detection Features (SDF) for use in action recognition pipelines. These descriptors are statistically motivated high-order representations.

—**Chapter 5: Tensor Representations.** In this chapter, we design sequence and dynamics compatibility kernels to capture the spatio-temporal evolution of 3D skeleton body-joints. We derive linearizations of these kernels using tensors. We extend these kernels to aggregate over multiple subsequences and CNN classifier scores. Additionally, we conduct a novel theoretical analysis of Tensor Power Normalization, connecting it to subspace methods. We are the first to conduct a theoretical analysis of higher-order pooling with Tensor Power Normalization, using it for generic action recognition as well as fine-grained action recognition.

—**Chapter 6: Multi-order Multi-mode Transformer (3Mformer).** In this chapter, we model the skeleton data as a hypergraph of orders 1 to r, where human body joints serve as nodes. Higher-order Transformer (HoT) embeddings of these formed hyper-edges represent various groups of 3D body joints and capture various higher-order dynamics important for action recognition. As HoT embeddings represent individual hyper-edge order and block, we introduce a novel Multi-order Multi-mode Transformer (3Mformer) with two modules: Multi-order Pooling and Temporal block Pooling. Their goal is to form coupled-mode tokens, such as 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' and 'channel-only', and perform weighted hyper-edge aggregation and temporal block aggregation.

—**Chapter 7: Uncertainty-DTW.** In this chapter, we introduce the uncertainty-DTW, known as uDTW, which takes into account the uncertainty of frame-wise (or block-wise) features by selecting the path that maximizes Maximum Likelihood Estimation (MLE). We use parameters like variance from a distribution (*e.g.*, the Normal distribution) within MLE (and uDTW) to model uncertainty. We provide several pipelines that utilize uDTW for (1) forecasting time series evolution, (2) estimating the Fréchet mean of time series, (3) supervised few-shot action recognition, and (4) unsupervised few-shot action recognition.

—**Chapter 8: Temporal-Viewpoint Transportation Plan.** In this chapter, we propose a few-shot action recognition approach for learning on skeleton-based articulated 3D body joints via JEANIE. It performs joint alignment of temporal blocks and simulated viewpoint indexes of skeletons between support-query sequences to select the smoothest path without abrupt jumps in matching temporal locations and view indexes. Warping jointly temporal locations and simulated viewpoint indexes helps meta-learning with limited samples of novel classes. We propose a simple similarity-based loss to encourage the alignment of within-class sequences and prevent the alignment of between-class sequences.

—**Chapter 9: Summary and Future Work.** In the final chapter of this thesis, we summarize our contributions and outline and discuss future research directions.

1.8 Publications

The following publications are associated with the research in this thesis and have not been used to obtain any other degree from another university or institution.

- **Lei Wang***, Piotr Koniusz* and Du Q. Huynh. "Hallucinating IDT Descriptors and I3D Optical Flow Features for Action Recognition with CNNs". *International Conference on Computer Vision (ICCV, **A***)*, pp. 8698-8708, 2019.
- **Lei Wang**, Du Q. Huynh and Piotr Koniusz. "A Comparative Review of Recent Kinect-based Action Recognition Algorithms". *IEEE Transactions on Image Processing (TIP, **IF: 11.041**)*, vol. 29, pp. 15-28, 2020.
- **Lei Wang** and Piotr Koniusz. "Self-supervising Action Recognition by Statistical Moment and Subspace Descriptors". *The 29th ACM International Conference on Multimedia (ACM MM, **A***)*, pp. 4324–4333, 2021.
- **Lei Wang*** and Piotr Koniusz*. "Uncertainty-DTW for Time Series and Sequences". *The 17th European Conference on Computer Vision (ECCV, **A***, oral)*, pp. 176–195, 2022.
- **Lei Wang** and Piotr Koniusz. "Temporal-Viewpoint Transportation Plan for Skeletal Few-shot Action Recognition". *The 16th Asian Conference on Computer Vision (ACCV, oral, **Sang Uk Lee Best Student Paper Award**)*, pp. 4176-4193, 2022.
- Piotr Koniusz, **Lei Wang** and Anoop Cherian. "Tensor Representations for Action Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI, **IF: 24.314**)*, vol. 44, no. 2, pp. 648-665, 2022.
- **Lei Wang** and Piotr Koniusz. "3Mformer: Multi-order Multi-mode Transformer for Skeletal Action Recognition". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR, **A***)*, pp. 5620-5631, 2023.

During my PhD, I also co-authored the following paper on action recognition, which is not discussed in this thesis.

- Zhenyue Qin, Yang Liu, Pan Ji, Dongwoo Kim, **Lei Wang**, R.I. (Bob) McKay, Saeed Anwar and Tom Gedeon. "Fusing Higher-Order Features in Graph Neural Networks for Skeleton-Based Action Recognition. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS, **IF: 14.255**)*, 2021.

* indicates equal contribution.

Chapter 2

A Comparative Review

Video-based human action recognition is currently one of the most active research areas in computer vision. Various research studies indicate that the performance of action recognition is highly dependent on the type of features being extracted and how the actions are represented. Since the release of the Kinect camera, a large number of Kinect-based human action recognition techniques have been proposed in the literature. However, there still does not exist a thorough comparison of these Kinect-based techniques under the grouping of feature types, such as handcrafted versus deep learning features and depth-based versus skeleton-based features. In this chapter, we analyze and compare ten recent Kinect-based algorithms for both cross-subject action recognition and cross-view action recognition using six benchmark datasets. In addition, we have implemented and improved some of these techniques and included their variants in the comparison. Our experiments show that the majority of methods perform better on cross-subject action recognition than cross-view action recognition, that skeleton-based features are more robust for cross-view recognition than depth-based features, and that deep learning features are suitable for large datasets.

2.1 Introduction

Human action recognition has many useful applications such as human computer interaction, smart video surveillance, sports and health care. These applications are one of motivations behind much research work devoted to this area in the last few years. However, even with a large number of research papers found in the literature, many challenging problems, such as different viewpoints, visual appearances, human body sizes, lighting conditions, and speeds of action execution, still affect the performance of these algorithms. Further problems include partial occlusion of the human subjects by other objects in the scene and self-occlusion of human subjects themselves. Among the human action recognition papers presented in the literature, some of the early techniques focus on using conventional RGB videos [Bobick and Davis, 2001; Dollár et al., 2005; Blank et al., 2005; Ke et al., 2007; Laptev et al., 2008; Liu and Shah, 2008; Bregonzio et al., 2009; Liu et al., 2009]. While these video-based techniques gave promising results, their recognition accuracy is still relatively low,

even when the scene is free of clutter.

The Kinect camera introduced by Microsoft in 2001 was an attempt to broaden the 3D gaming experience of the Xbox 360's audience. However, as the Kinect camera can capture real-time RGB and depth videos, and there is a publicly available toolkit for computing the human skeleton model from each frame of a depth video, many research papers on 3D human action recognition using the Kinect camera have emerged. One advantage of using depth videos than the conventional RGB videos is that it is easier to segment the foreground human subject even when the scene is cluttered. As depth videos do not have colour information, the colour of the clothes worn by the human subject has no effect on the segmentation process. This allows action recognition researchers to focus their effort more on getting robust feature descriptors to describe the actions rather than on low level segmentation. Numerous representative methods for 3D action analysis using depth videos include [Li et al., 2010; Yang and Tian, 2012; Oreifej and Liu, 2013; Rahmani et al., 2014c, 2016b; Rahmani and Mian, 2016]. These methods employ advanced machine learning techniques for which good results have been reported. Of course, depth images are also vulnerable to noise due to various factors [Mallick et al., 2014]. Thus, using depth images does not always guarantee good action recognition performance [Li et al., 2010]. The algorithm used for computing the 3D joint positions of the human skeletal model by the Kinect toolkit is based on the human skeleton tracking framework (OpenNI) of Shotton et al. [2011]. In addition to the availability of the real-time depth video stream, this tracking framework also opens up the research area of skeleton-based human action recognition [Vemulapalli et al., 2014; Shahroudy et al., 2016a; Vemulapalli and Chellappa, 2016; Ke et al., 2017a,b; Rahmani and Bennamoun, 2017].

Human action recognition methods using the Kinect data can be classified into two categories, based on how the feature descriptors are extracted to represent the human actions. The first category is handcrafted features. Action recognition methods using handcrafted features require two complex hand-design stages, namely *feature extraction* and *feature representation*, to build the final descriptor. Both the feature extraction stage and feature representation stage differ from one method to another. The feature extraction stage may involve computing the depth (and/or colour) gradients, histogram, and other more complex transformations of the video data. The feature representation stage may involve simple concatenation of the feature components extracted from the previous stage, a more complex fusion step of these feature components, or even using a machine learning technique, to get the final feature descriptor. These methods usually involve a number of operations that require researchers to carry out careful feature engineering and tuning. Kinect-based human action recognition algorithms using handcrafted features reported in the literature include [Li et al., 2010; Shotton et al., 2011; Yang and Tian, 2012; Oreifej and Liu, 2013; Rahmani et al., 2014c, 2016b; Vemulapalli et al., 2014; Vemulapalli and Chellappa, 2016; Koniusz et al., 2016a; Zhang et al., 2018b].

The second category is deep learning features. With the huge advance in neural network research in the last decade, deep neural networks have been used to extract high-level features from video sequences for many different applications, including

3D human action analysis. Deep learning methods reduce the need for feature engineering; however, they require a huge amount of labelled training data, which may not be available, and a long time to train. For small human action recognition datasets, deep learning methods may not give the best performance. Recent Kinect-based human action recognition algorithms are: [Shahroudy et al., 2016a; Rahmani and Mian, 2016; Rahmani and Bennamoun, 2017; Rahmani et al., 2015; Ke et al., 2017a,b; Yan et al., 2018; Li et al., 2018b,a; Wang and Wang, 2017; Tang et al., 2018; Liu et al., 2017; Huang et al., 2017; Lee et al., 2017; Si et al., 2018; Wang and Wang, 2018; Liu et al., 2018; Tanfous et al., 2018; Zheng et al., 2018].

Research contributions. Although both handcrafted and deep learning features have been used in human action recognition, to the best of our knowledge, a thorough comparison of recent action recognition methods for these two categories is not found in the literature. Our contributions in this chapter are twofold:

- We evaluate the performance of 10 recent state-of-art human action recognition algorithms, with specific focus on comparing the effectiveness of using handcrafted features versus deep learning features and skeleton-based features versus depth-based features. We believe that there is a lack of such a comparison in the literature on human action recognition.
- Furthermore, we evaluate the cross-view versus cross-subject performance of these algorithms and, for the multiview datasets, the impact of the camera view for both small and large datasets on human action recognition with respect to whether the features being used are depth-based, skeleton-based, or depth+skeleton-based. To the best of our knowledge, such evaluation has not been performed before.

The chapter is organized as follows. Section 2.2 gives a brief review on recent human action recognition techniques. Section 2.3 covers the details of the 10 algorithms being compared in this chapter. In Section 2.4, we describe our experimental setting and the benchmark datasets. Sections 2.5 and 2.6 summarize our experimental results, comparison, and discussions. The last section concludes the chapter.

2.2 Related Work

Action recognition methods can be classified into three categories based on the type of input data: colour-based [Bobick and Davis, 2001; Blank et al., 2005; Dollár et al., 2005; Fishkin et al., 2005; Hodges and Pollack, 2007; Ke et al., 2007; Liu and Shah, 2008; Laptev et al., 2008; Liu et al., 2009; Bregonzio et al., 2009; Buettner et al., 2009; Sung et al., 2011, 2012; Koppula et al., 2013; Gupta et al., 2014; Bilen et al., 2016; Shahroudy et al., 2016b; Feichtenhofer et al., 2017a; Kar et al., 2017; Cherian et al., 2017b,a; Shahroudy et al., 2018; Carreira and Zisserman, 2017; Feichtenhofer et al., 2016b; Hu et al., 2018], depth-based [Li et al., 2010; Yang and Tian, 2012; Wang et al., 2012; Oreifej and Liu, 2013; Xia and Aggarwal, 2013; Rahmani et al., 2014c,b,a; Yang and Tian, 2014b; Wang et al., 2015b; Rahmani et al., 2015; Wang et al., 2016c; Rahmani

et al., 2016a,b; Rahmani and Mian, 2016; Shahroudy et al., 2016b; Zhang et al., 2017a; Rahmani and Bennamoun, 2017; Hu et al., 2018; Shi and Kim, 2017; Zhang et al., 2018b], and skeleton-based [Shotton et al., 2011; Xia et al., 2012; Vemulapalli et al., 2014; Du et al., 2015; Veeriah et al., 2015; Zhu et al., 2016; Liu et al., 2016; Koniusz et al., 2016a; Vemulapalli and Chellappa, 2016; Amor et al., 2016; Shahroudy et al., 2016a; Rahmani and Bennamoun, 2017; Ke et al., 2017b,a; Hu et al., 2018; Wang and Wang, 2017, 2018; Si et al., 2018; Lee et al., 2017; Huang et al., 2017; Liu et al., 2017; Tang et al., 2018; Li et al., 2018a; Liu et al., 2018; Tanfous et al., 2018; Zheng et al., 2018; Elmadany et al., 2018; Zhang et al., 2018b; Si et al., 2019; Li et al., 2019; Shi et al., 2019a,b]. In this section, we will focus on reviewing recent methods using the last two types of features.

Depth-based action recognition. Action recognition from depth videos [Li et al., 2010; Yang and Tian, 2012; Wang et al., 2012; Oreifej and Liu, 2013; Xia and Aggarwal, 2013; Yang and Tian, 2014b; Rahmani et al., 2014b,a, 2016b,a] has become more popular because of the availability of real-time cost-effective sensors. Most existing depth-based action recognition methods use global features such as space-time volume and silhouette information. For example, Oreifej and Liu [2013] captured the discriminative features by projecting the 4D surface normals obtained from the depth sequence onto a 4D regular space to build the *Histogram of Oriented 4D Normals* (HON4D). Yang and Tian [2014b] extended HON4D by concatenating local neighbouring hypersurface normals from the depth video to jointly characterize local shape and motion information. More precisely, they introduced an adaptive spatio-temporal pyramid to subdivide the depth video into a set of space-time cells for more discriminative features. Xia and Aggarwal [2013] proposed to filter out the noise from the depth sensor so as to get more reliable spatio-temporal interest points for action recognition. Although these methods have achieved impressive performance for frontal action recognition, they are sensitive to changes of viewpoint. One way to alleviate this viewpoint issue is to directly process the pointclouds, as reported in the paper by Rahmani et al. [2016b].

Apart from the methods mentioned above which use handcrafted features, the use of deep learning features [Wang et al., 2015b; Rahmani et al., 2015; Wang et al., 2016c; Shahroudy et al., 2016b; Rahmani and Mian, 2016; Rahmani and Bennamoun, 2017; Zhang et al., 2017a; Shi and Kim, 2017] in human action recognition is on the rise. For example, Wang et al. [2015b] used a *Hierarchical Depth Motion Maps* (HDMMs) to extract the body shape and motion information and then trained a 3-channel deep *Convolutional Neural Network* (CNN) on the HDMMs for human action recognition. In the following years, Rahmani and Mian [2016] proposed to train a single *Human Pose Model* (HPM) from real motion capture data to transfer the human pose from different unknown views to a view-invariant feature space, and Zhang et al. [2017a] used a multi-stream deep neural networks to jointly learn the semantic relations among action attributes.

Skeleton-based action recognition. Existing skeleton-based action recognition methods can be grouped into two categories: joint-based methods and body part based methods. Joint-based methods model the positions and motion of the joints (either individual or a combination) using the coordinates of the joints extracted by the OpenNI

tracking framework. For instance, a reference joint may be used and the coordinates of other joints are defined relative to the reference joint [Yang and Tian, 2012; Vemulapalli et al., 2014; Ke et al., 2017b,a], or the joint orientations may be computed relative to a fixed coordinate system and used to represent the human pose [Xia et al., 2012], *etc*. For the body part based methods, the human body parts are used to model the human's articulated system. These body parts are usually modelled as rigid cylinders connected by joints. Information such as joint angles [Vemulapalli and Chellappa, 2016], temporal evolution of body parts [Amor et al., 2016; Shahroudy et al., 2016a; Koniusz et al., 2016a; Yan et al., 2018], and 3D relative geometric relationships between rigid body parts [Vemulapalli et al., 2014; Vemulapalli and Chellappa, 2016; Yan et al., 2018] has all been used to represent the human pose for action recognition.

The method proposed by Vemulapalli et al. [2014] falls into the body part based category. They represent the relative geometry between a pair of body parts, which may or may not be directly connected by a joint, as a point in $SE(3)$. Thus, a human skeleton is a point of the Lie group $SE(3) \times ... \times SE(3)$ where each action corresponds to a unique evolution of such a point in time. The approach of Ke et al. [2017a] relies on both body parts and body joints. The human skeleton model was divided into 5 body parts. A specific joint was selected for each body part as the reference joint and the coordinates of other joints were expressed as vectors relative to that reference joint. Various distance measures were computed from these vectors to yield a feature vector for each video frame. The features vectors from all video frames were finally appended together and scaled to form a handcrafted greyscale image descriptor fed into a CNN. Somewhat related approach [Tas and Koniusz, 2018] uses kernels formed over body joints to obtain feature maps fed into a CNN for simultaneous action recognition and domain adaptation.

Recent human action recognition papers favour deep learning techniques to perform human action recognition. Apart from the CNN-based approaches [Ke et al., 2017a; Tas and Koniusz, 2018], *Recurrent Neural Networks* (RNNs) have also been popular [Du et al., 2015; Veeriah et al., 2015; Shahroudy et al., 2016a; Zhu et al., 2016; Liu et al., 2016; Li et al., 2018b; Shi and Kim, 2017; Wang and Wang, 2018; Zheng et al., 2018]. Since *Long Short-term Memory* (LSTM) [Hochreiter and Schmidhuber, 1997] can model temporal dependencies as RNNs and even capture the co-occurrences of human joints, LSTM networks have also been a popular choice in human action recognition [Liu et al., 2017; Lee et al., 2017; Tanfous et al., 2018; Liu et al., 2018; Si et al., 2019]. For instance, Zhu et al. [2016] presented an end-to-end deep LSTM network with a *dropout* step, Shahroudy et al. [2016a] proposed a *Part-aware Long Short-term Memory* (P-LSTM) network to learn the long-term patterns of the 3D trajectories for each grouped body part, and Liu et al. [2018] introduced the use of *trust gates* in their spatio-temporal LSTM architecture.

Action recognition via a combination of skeleton and depth features. Combining skeleton and depth features together helps overcome situations when there are interactions between human subject and other objects or when the actions have very similar motion trajectories. Various action recognition algorithms [Rahmani et al., 2014c; Shahroudy et al., 2016b; Rahmani and Bennamoun, 2017; Elmadany et al., 2018] that

Table 2.1: Ten state-of-the-art action recognition methods evaluated in this chapter.

Algorithms	Year	Short descriptions	Kinect data used	Feature dimension
HON4D [Oreifej and Liu, 2013]	CVPR 2013	handcrafted (global descriptor)	depth	[17880, 151200]
HDG [Rahmani et al., 2014c]	WACV 2014	handcrafted (local + global descriptor)	depth+skeleton	[1662, 1819]
LARP-SO [Vemulapalli and Chellappa, 2016]	CVPR 2016	handcrafted (Lie Group)	skeleton	$3 \times 3 \times$ #frames
HOPC [Rahmani et al., 2016b]	TPAMI 2016	handcrafted (local descriptor)	depth $\rightarrow$ pointcloud	depending on #STKs[†]
SCK+DCK [Koniusz et al., 2016a]	ECCV 2016	handcrafted (tensor representations)	skeleton	$\sim$ 40k
P-LSTM [Shahroudy et al., 2016a]	CVPR 2016	deep learning (LSTM)	skeleton	#joints $\times 3 \times 8$[‡]
HPM+TM [Rahmani and Mian, 2016]	CVPR 2016	deep learning (CNN)	depth	4096
Clips+CNN+MTLN [Ke et al., 2017b]	CVPR 2017	deep learning (pre-trained VGG19, MTLN)	skeleton	7168
IndRNN [Li et al., 2018b]	CVPR 2018	deep learning (independently RNN)	skeleton	512
ST-GCN [Yan et al., 2018]	AAAI 2018	deep learning (Graph ConvNet)	skeleton	256

[†]STK stands for *spatio-temporal keypoint*. [‡]The P-LSTM features include 8 video segments, each of which is composed of a number of 3D joints.

use both depth and skeleton features for robust human action recognition have been reported in recent years. For example, Rahmani et al. [2014c] proposed to combine 4 types of local features extracted from both depth images and 3D joint positions to deal with local occlusions and to increase the recognition accuracy. We refer to their method as *HDG* from hereon. Another example is the approach of Shahroudy et al. [2016b] where *Local Occupancy Patterns* (LOP), HON4D, and skeleton-based features are combined with hierarchical mixed norms which regularize the weights in each modality group of each body part. Recently, Rahmani and Bennamoun [2017] used an end-to-end deep learning model to learn the body part representation from both skeletal and depth images. To improve the performance of the model, they adopted a bilinear compact pooling [Gao et al., 2016] layer for the generated depth and skeletal features. Elmadany et al. [2018], on the other hand, proposed to use canonical correlation analysis to maximize the correlation of features extracted from different sensors. The features investigated in their paper include bag of angles extracted from skeleton data, depth motion map from depth video, and optical flow from RGB video. The subspace shared by all the features was learned and average pooling was used to get the final feature descriptor.

2.3 Analyzed and Evaluated Algorithms

We chose ten action recognition algorithms shown in Table 2.1 for our comparison and evaluation as they are recent action recognition methods (from 2013 onward) and they use skeleton-based, depth-based, handcrafted, and/or deep learning features. The technical details of these algorithms are summarized below.

HON4D. Oreifej and Liu [2013] presented a global feature descriptor that captures the geometry and motion of human action in the 4D space of spatial coordinates, depth and time. To form the HON4D descriptor, the space was quantized using a 600-cell polychoron with 120 vertices. The vectors stretching from the origin to these vertices were used as projection axes to obtain the distribution of normals for each video sequence. To improve the classification performance, random perturbations were added to those projectors. The dimensions of HON4D features (Table 2.1) vary

across different datasets.

HDG. In this algorithm [Rahmani et al., 2014c], each depth sequence was firstly divided into small subvolumes; the histograms of depth and depth derivatives were computed for each subvolume. For each skeleton sequence, the torso joint was used as a stable reference joint for computing the histograms of joint position differences. In addition, the variations of each joint movement volume were incorporated into the global feature vector to form spatio-temporal joint features. Two *Random Decision Forests* (RDFs) were trained in this algorithm, one for feature pruning and one for classification. More details about feature dimensions of HDG and the feature pruning applied by us will be given in Section 2.4.4.

HOPC. Approach [Rahmani et al., 2016b] models depth images as 3D pointclouds. The authors used two types of support volume, namely, so-called spatial support volume and spatio-temporal support volume. The HOPC descriptor was extracted from the pointcloud falling inside the support volume around each point, which may be classified as a *spatio-temporal Keypoint* (STK) if the eigenvalue ratios of the pointcloud around it are larger than some predefined threshold. For each STK, the algorithm further projected eigenvectors onto the axes of the 20 vertices of a regular dodecahedron. The final HOPC descriptor for each STK is a concatenation of 3 small histograms, each of which captures the distribution of an eigenvector of the pointcloud within the support volume.

LARP-SO. Vemulapalli and Chellappa [2016] extended their earlier work [Vemulapalli et al., 2014] by the use of Lie Algebra Relative Pairs via $SO(3)$ for action recognition. We follow the convention adopted in [Rahmani and Bennamoun, 2017] and name this algorithm as LARP-SO. In this algorithm, the rolling map, which describes how a Riemannian manifold rolls over another one along a smooth rolling curve, was used for 3D action recognition. Each skeleton sequence was firstly represented by the relative 3D rotations between various human body parts, and each action was then modelled as a curve in the Lie Group. Since it is difficult to perform the classification of action curves in a non-Euclidean space, the curves were unwrapped by the logarithm map at a single point while a rolling map was used to reduce distortions. The *Fourier Temporal Pyramid* (FTP) representation [Wang et al., 2012] was used in the algorithm to make the descriptor more robust to noise and less sensitive to temporal misalignments.

SCK+DCK. Koniusz et al. [2016a] used tensor representations to capture the higher-order relationships between 3D human body joints for action recognition. They applied two different RBF kernels which they referred to as *Sequence Compatibility Kernel* (SCK) and *Dynamics Compatibility Kernel* (DCK). The former kernel captures the spatio-temporal compatibility of joints while the latter models the action dynamics of a sequence. An SVM was then trained on linearized feature maps of such kernels for action classification.

HPM+TM. Approach [Rahmani and Mian, 2016] employs a dictionary containing representative human poses from a motion capture database. A deep CNN architecture which is a modification of [Gupta et al., 2014] was then used to train a view-invariant human pose model. Real depth sequences were passed to the learned model frame-by-frame to extract high-level view-invariant features. Similarly to the LARP-SO

algorithm above, the FTP was used to capture the temporal structure of the action videos. The final descriptor for each video sequence is a collection of the Fourier coefficients from all the segments.

P-LSTM. Approach [Shahroudy et al., 2016a] proceeds by transforming 3D coordinates of the body joints from the camera coordinate system to the body coordinate system with the origin set at the spine. The 3D coordinates of all other body joints were then scaled based on the distance between the 'hip centre' joint and the 'spine' joint. A P-LSTM model was built by splitting the memory cells from the LSTM model into body part based sub-cells. For each video sequence, the pre-processed human joints were grouped into 5 parts (torso, two hands, and two legs) and the video was divided into 8 equal-sized video segments. Then, for a randomly selected frame per video segment, 3D coordinates of the joints inside each grouped part were concatenated and passed as input to the P-LSTM network to learn common temporal patterns of the parts and combine them into a global representation.

Clips+CNN+MTLN. Ke et al. [2017b] presented a skeletal representation referred to as *clips*. The method proceeds by transforming the Cartesian coordinates of human joints (per skeleton sequence) into the cylindrical coordinates to generate 3 clips, with each clip corresponding to one channel of the cylindrical coordinates. To encode the temporal information for the whole video sequence, four stable joints (left shoulder, right shoulder, left hip and right hip) were selected as reference joints to produce 4 coordinate frames. The pre-trained VGG19 network [Simonyan and Zisserman, 2015] was used as a feature extractor to learn the long-term spatio-temporal features from intermediate images formed from the 4 coordinate frames. Moreover, approach [Ke et al., 2017b] also employs the *Multi-task Learning Network* (MTLN) proposed by [Caruana, 1997] to incorporate the spatial structural information from the CNN features.

IndRNN. Li et al. [2018b] proposed a new RNN method, an Independently Recurrent Neural Network, for which neurons per layer are independent of each other but they are reused across layers. Finally, multiple IndRNNs were stacked to build a deeper network than the traditional RNN.

ST-GCN. The spatio-temporal graph representation for skeleton sequences proposed by Yan et al. [2018] is an extension of *Graph Convolutional Networks* (GCN) [Bruna et al., 2014; Defferrard et al., 2016; Kipf and Welling, 2017] tailored to perform human action recognition. Firstly, the spatio-temporal graph is constructed by inserting edges between neighbouring body joints (nodes) of the human body skeleton as well as along the temporal direction. Subsequently, GCN and a classifier are applied to infer dependencies in the graphs (a single graph corresponds to a single action sequence) and perform classification.

2.4 Experimental Setting

To perform experiments, we obtained off-the-shelf codes for HON4D [Oreifej and Liu, 2013], HOPC [Rahmani et al., 2016b], LARP-SO [Vemulapalli and Chellappa,

Table 2.2: Six publicly available benchmark datasets used in our experiments for 3D action recognition.

Datasets	Year	Classes	Subjects	#Views	#videos	Sensor	Modalities	#joints
MSRAction3D [Li et al., 2010]	2010	20	10	1	567	Kinect v1	Depth+3DJoints	20
3D Action Pairs [Oreifej and Liu, 2013]	2013	12	10	1	360	Kinect v1	RGB+Depth+3DJoints	20
CAD-60 [Sung et al., 2011]	2011	14	4	–	68	Kinect v1	RGB+Depth+3DJoints	15
UWA3D Activity Dataset [Rahmani et al., 2014b]	2014	30	10	1	701	Kinect v1	RGB+Depth+3DJoints	15
UWA3D Multiview Activity II [Rahmani et al., 2016b]	2015	30	9	4	1070	Kinect v1	RGB+Depth+3DJoints	15
NTU RGB+D Dataset [Shahroudy et al., 2016a]	2016	60	40	80	56880	Kinect v2	RGB+Depth+3DJoints	25

(The number of views is not stated in the CAD-60 dataset.)

2016], HPM+TM [Rahmani and Mian, 2016], IndRNN [Li et al., 2018b] and ST-GCN [Yan et al., 2018] from the respective authors' websites. For SCK+DCK [Koniusz et al., 2016a], HDG [Rahmani et al., 2014c], P-LSTM [Shahroudy et al., 2016a] and Clips+CNN+MTLN [Ke et al., 2017b], we used our own Matlab implementations given that codes for these methods are not publicly available. Moreover, we employed ten variants of the HDG [Rahmani et al., 2014c] representation so as to evaluate the performance with respect to different combinations of its individual descriptor types. We also implemented the traditional RNN and LSTM as baseline methods, and added four variants of P-LSTM to evaluate the impact of using different numbers of video segments for skeletal representation as well as different numbers of hidden neurons.

2.4.1 Benchmark Datasets

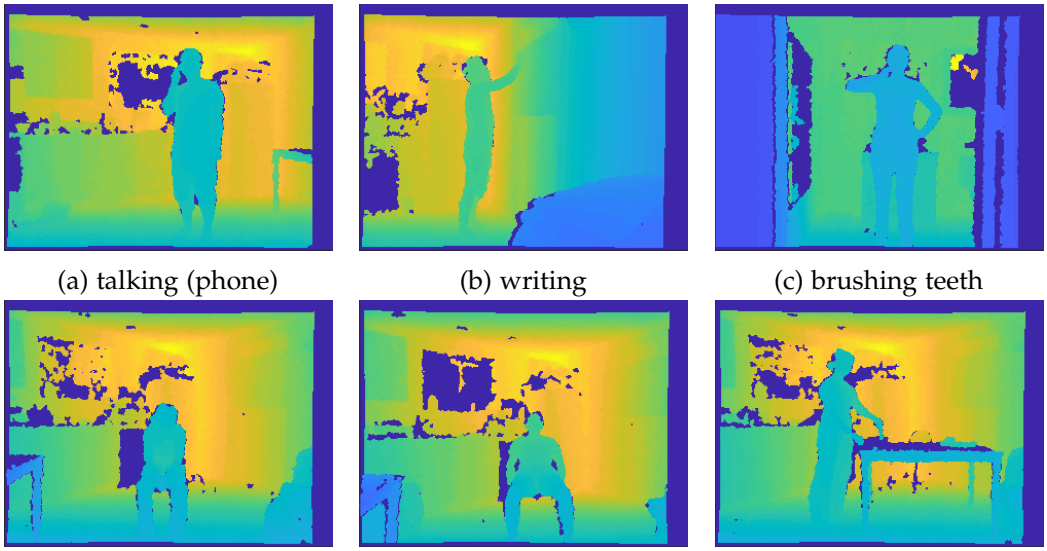
Listed in Table 2.2 are six benchmark datasets used in our evaluation, each of which is detailed below.

MSRAction3D [Li et al., 2010] is one of the earliest action datasets captured with the Kinect depth camera. It contains 20 human sport-related activities such as *jogging*, *golf swing* and *side boxing*. Each action in this dataset was performed 2 or 3 times by 10 people. This dataset is challenging because of high inter-action similarities.

3D Action Pairs [Oreifej and Liu, 2013] contains 6 selected pairs of actions that have very similar motion trajectories, *e.g.*, *put on a hat* and *take off a hat*; *pick up a box* and *put down a box*; *stick a poster* and *remove a poster*. Each action was performed 3 times by 10 people. There are two challenging aspects of this dataset: (i) the actions in each pair have similar motion trajectories; (ii) the object that is interacted by the subject in each video is only present in the RGB-D data but not the skeleton data.

Cornell Activity Dataset (CAD) [Sung et al., 2011] comprises two sub-datasets, CAD-60 and CAD-120. Both sub-datasets contain RGB-D and tracked skeleton video sequences of human activities captured by a Kinect sensor. In this chapter, only CAD-60 was used in the experiments. Fig. 2.1 illustrates depth images from the CAD-60 dataset and demonstrates that this dataset exhibits high levels of noise in its depth videos.

UWA3D Activity Dataset [Rahmani et al., 2014b] contains 30 actions performed by 10 people of various height at different speeds in cluttered scenes. This dataset has high



(a) talking (phone) (b) writing (c) brushing teeth

(d) talking (couch) (e) relaxing (couch) (f) cooking (stirring)

Figure 2.1: Sample depth images from the CAD-60 dataset.

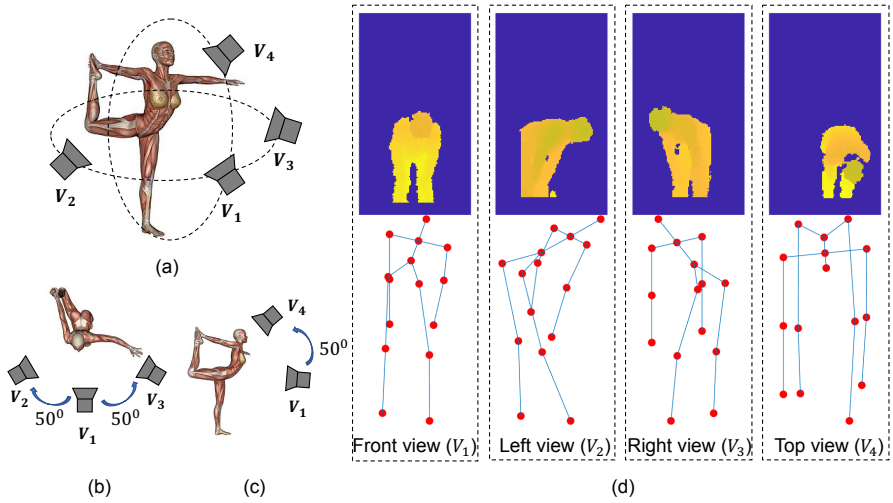


Figure 2.2: (a) A perspective view of the camera setup in the UWA3D Multiview Activity II dataset. The views V_1, V_2 and V_3 are at the same height. (b) and (c) show the top and side views of the setup. The angles between V_1 and V_2, between V_1 and V_3, and between V_1 and V_4 are all approximately 50 degrees [Rahmani et al., 2014b]. (d) An example video frame of the depth and skeleton data for the *bending* action.

between-class similarity and contains frequent self-occlusions.

UWA3D Multiview Activity II [Rahmani et al., 2016b] contains 30 actions performed by 9 people in a cluttered environment. In this dataset, the Kinect camera was moved to different positions to capture the actions from 4 different views (see Fig. 2.2(a)-(c)): front view (V_1), left view (V_2), right view (V_3), and top view (V_4). This dataset is

therefore more challenging than the previous four datasets. Fig. 2.2(d) shows sample video frames from this dataset.

NTU RGB+D Dataset [Shahroudy et al., 2016a] is so far the largest Kinect-based action dataset which contains 56,880 video sequences and over 4 million frames. There are 60 action classes performed by 40 subjects captured from 80 views with 3 Kinect v.2 cameras. This dataset has variable sequence lengths for different sequences and exhibits high intra-class variations.

Dataset usage. Below we detail how the above six datasets were used in our experiments.

The MSRAction3D, 3D Action Pairs, CAD-60 and UWA3D Activity datasets were used for cross-subject (single-view) experiments. For every dataset, we used half of the subjects' data for training and the remaining half for testing. We tested all the possible combinations of subjects for the training and testing splits to obtain the average recognition accuracy of each algorithm. For example, for 10 subjects in the MSRAction3D dataset, $\binom{10}{5} = 252$ experiments were carried out.

The UWA3D Multiview Activity II dataset was used for cross-view experiments, with two views of the samples being used for training and the remaining views for testing. There were 12 different view combinations in the experiments.

The NTU RGB+D dataset was used in both cross-subject and cross-view experiments. Despite indications that this dataset has 80 views of human action recognition, the data samples were grouped according to three camera sets. For cross-view action recognition, we used the video sequences captured by two cameras as our training data and the remaining sequences for testing. A total of 3 different camera combinations were experimented with.

2.4.2 Evaluation Settings

Below we detail the experimental settings of the algorithms.

HON4D. According to [Oreifej and Liu, 2013], HON4D has the frame size of 320×240 and each video is divided into $4 \times 3 \times 3$ (width$\times$height$\times$#frames) spatio-temporal cells. In our evaluations, we used these same settings for all the datasets.

HOPC. Rahmani et al. [2016b] used different spatial and temporal scales for different datasets. In this chapter, a constant temporal scale and spatial scale were used for all the datasets. For the MSRAction3D and 3D Action Pairs datasets, the temporal and spatial scales were set to 2 and 19, respectively. For the remaining datasets, we used 2 for the temporal scale and 140 as the spatial scale. Moreover, we divided each depth video into $6 \times 5 \times 3$ spatio-temporal cells (along the X, Y and *time* axes) to extract features.

LARP-SO. The desired number of frames [Vemulapalli and Chellappa, 2016] used for computing skeletal representation varies depending on the datasets used in the experiments. The desired frame numbers for the UWA3D Activity, UWA3D Multiview Activity II, and NTU RGB+D datasets were all set to 100. For the MSRAction3D, 3D Action Pairs datasets, and CAD-60, they were set to 76, 111, and 1,000, respectively.

SCK+DCK. We followed the experimental settings described in [Koniusz et al., 2016a, 2020] for all the datasets and we used authors' newest model which aggregates over subsequences (not just sequences). For SCK, we normalized all human body joints with respect to the hip joints across frames as well as the lengths of all body parts. For DCK, we used the unnormalized body joints, and assumed that the displacements of body joint coordinates across frames captured their temporal evolution.

HDG. According to [Rahmani et al., 2014c]), the number of used subvolumes has no significant effect to the discriminative features, we divided each video sequence into $10\times10\times5$ subvolumes (along X, Y and *time*) for computing the histograms of depth as well as the depth gradients. For the joint movement volume features, each joint volume was divided into $1\times1\times5$ cells (along X, Y and *time*). There are four individual feature representations encapsulated by HDG:

(i) histogram of depth (`hod`),

(ii) histogram of depth gradients (`hodg`),

(iii) joint position differences (`jpd`),

(iv) joint movement volume features (`jmv`).

We follow [Wang, 2017] and evaluate the performance of the 10 variants of HDG in our experiments.

HPM+TM. We followed [Rahmani and Mian, 2016] and set the number of Fourier Pyramid levels to 3 and the number of low frequency Fourier coefficients to 4 for all datasets. We used the human pose model trained by Rahmani and Mian [2016] to extract view-invariant features from each depth sequence. We also compared the recognition accuracies of this algorithm given Average Pooling (AP) versus Temporal Modelling (TM) used for extraction of CNN features.

P-LSTM. We applied the same normalization preprocessing step as in [Shahroudy et al., 2016a] for the skeletal representation. In our experiments, the number of video segments and the number of hidden neurons for 1-layer RNN, 2-layer RNN, 1-layer LSTM, 2-layer LSTM, 1-layer P-LSTM and 2-layer P-LSTM were all set to 8 and 50, respectively. We also evaluated the performance of different numbers of video segments and different numbers of hidden neurons in P-LSTM. The learning rate and the number of epochs in our experiments were set to 0.01 and 300, respectively.

Clips+CNN+MTLN. The learning rate was set to 0.001 and the batch size was set to 100 for MTLN. We selected four different experimental settings from [Ke et al., 2017b] to compare the performance of recognition: Frames+CNN, Clips+CNN+Pooling, Clips+CNN+Concatenation, and Clips+CNN+MTLN.

IndRNN. We used the Adam optimizer with the initial learning rate 2×10^{-4} and applied the decay by 10 once the evaluation accuracy did not increase. For cross-subject and cross-view experiments, the dropout rates were set to 0.25 and 0.1, respectively.

ST-GCN. For the convolution operations, we used the optimal partition strategy according to the ablation study in [Yan et al., 2018]. As different datasets have

different numbers of body joints (see Table 2.2), we reconstructed the spatio-temporal skeleton graphs. For NTU RGB+D dataset, we used the same experimental settings as described in [Yan et al., 2018] (*e.g.*, we work with up to two human subjects per sequence). For the remaining 5 datasets, we used a different setting as only one performing subject was present per video.

Moreover, we performed extra experiments for IndRNN and ST-GCN: instead of using 3D skeleton sequences as inputs, we used `jpd` features which redefine the 3D skeleton joint positions by translating them to be centred at the torso (or 'spine') joint.

2.4.3 Evaluation Measure

The recognition accuracy A_c of an algorithm for any given action class c is defined as the proportion of correct class c labels returned by the algorithm:

$$A_c = \#\text{correct\_class\_}c\text{\_labels} \,/\, \#\text{actual\_class\_}c\text{\_labels}\,. \tag{2.1}$$

To show the recognition accuracies of an algorithm for all the action classes, a confusion matrix is often used. The overall performance of an algorithm on a given dataset is evaluated using the average recognition accuracy $\bar{A}$ defined as: $\bar{A} = \frac{1}{C}\sum_{c=1}^{C} A_c$, where C is the total number of action classes in a given dataset.
To show the overall performance of each algorithm on M datasets, we first rank the performance of each algorithm from 1 to 5 (a lower rank value represents a better performance) based on the recognition accuracy so each algorithm has its own rank value r_i given the i^{th} dataset. We then compute the Average Rank (AVRank) as follows:

$$\text{AVRank} = \frac{1}{M}\sum_{i=1}^{M} r_i. \tag{2.2}$$

2.4.4 Optimisation of Hyperparameters for HDG

There are 3 hyperparameters in the HDG algorithm. The first hyperparameter is the number of subvolumes, which we set to the same value as in [Rahmani et al., 2014c]. The second and third hyperparameters, which are the number of trees N_{trees} used in training and the threshold θ used in feature pruning, were optimised during our experiments (Table 2.3). As the length of the combined features in the HDG algorithm is large (*e.g.*, the length of the HDG-all features for the MSRAction3D dataset is 13,250), we trained one RDF to select the feature components that have high importance values. This helped to increase the processing speed without compromising the recognition accuracy.

We evaluated the effect of hyperparameters N_{trees} and θ on the HDG algorithm for different combinations of individual HDG features using the MSRAction3D dataset (for single-view) and the UWA3D Multiview Activity II dataset (for cross-view). Table 2.3 shows the optimal values for N_{trees} and θ obtained from the grid search. The corresponding dimensions of different HDG combined features before and after

Table 2.3: Optimal hyperparameter values and feature dimensions before and after pruning for the HDG combined features.

Combination of individual features	MSRAction3D				UWA3D Multiview Activity II			
	dimensions before pruning	optimal θ $\times 10^{-3}$	optimal N_{trees}	dimensions after pruning	dimensions before pruning	optimal θ $\times 10^{-3}$	optimal N_{trees}	dimensions after pruning
HDG-hod	2,500	1.5	100	1,442	2,500	2.7	60	1,231
HDG-hodg	10,000	20.9	200	550	10,000	2.4	60	3,891
HDG-jpd	150	11.5	80	148	150	48.1	120	142
HDG-jmv	600	3.4	140	571	450	3.1	60	449
HDG-hod+hodg	12,500	17.0	180	786	12,500	4.7	140	3,987
HDG-jpd+jmv	750	2.0	80	690	600	6.8	100	581
HDG-hod+hodg+jpd	12,650	8.2	160	2,221	12,650	29.4	80	239
HDG-hod+hodg+jmv	13,100	7.4	180	2,189	12,950	25.0	100	135
HDG-hodg+jpd+jmv	10,750	8.3	180	1,711	10,600	15.2	140	456
HDG-all features	13,250	13.3	120	1,013	13,100	19.0	100	300

pruning are also indicated. Compared to other individual features of HDG, jpd and jmv are small-sized features, thus when used alone in both datasets, their dimensions are not reduced by much during feature pruning. However, when either or both of them are combined with other individual features in cross-view action recognition, their importance values are significantly higher. This allows a large reduction in feature dimension after pruning (see the last 4 rows of the table). Our experiments in the next section also confirm that skeleton-based features deal better with the view-invariance than depth-based features.

The optimal values of N_{trees} and θ shown in Table 2.3 were used to prune the HDG features for all datasets. As different datasets have different numbers of body joints (see Table 2.2), the dimensions of these HDG features after pruning across datasets are not the same.

2.5 Experimental Results

2.5.1 MSRAction3D, 3D Action Pairs, CAD-60, and UWA3D Activity Datasets

Table 2.4 summarizes the results for the single-view action recognition on these datasets. The algorithms with the highest recognition accuracy for the handcrafted feature and deep learning feature categories are highlighted in bold.

Among the methods that use handcrafted features, SCK+DCK outperformed all other methods on all the datasets. The use of RBF kernels to capture higher-order statistics of the data and complexity of action dynamics demonstrates its effectiveness for action recognition. For the 3D Action Pairs and UWA3D Activity datasets, HON4D and HOPC are, respectively, the second top performers. The poorer performance of HDG was due to noise in the depth sequences which affected the HDG-hod and HDG-hodg features, even though the human subjects were successfully segmented. In general, HDG-hodg outperformed HDG-hod, and HDG-jmv outperformed HDG-jpd. We also found that concatenating more individual features in HDG (see Table 2.4, row HDG-hod+hodg+jpd to row HDG-all features) helped improve action recognition. We note that our results for HDG are different from those reported in [Rahmani et al.,

Table 2.4: Comparison of average cross-subject action recognition accuracies (percentage) for the four single-view datasets (*i.e.*, $M = 4$ in Eq. (2.2)). Each block of rows shows the performance of one method and its variants. The best algorithm for each dataset is highlighted in bold. The last column of the table shows the average rank of the best performing algorithm in each block. The final rank values are computed using Eq. (2.2), where top performing methods have smaller rank values. Other poorer performing methods in the same block are not considered for their rank values, so their final ranks are marked as '–'.

	Method	MSRAction3D	3D Action Pairs	CAD-60	UWA3D Activity	AVRank
Hand-crafted features	HON4D [Oreifej and Liu, 2013] (Depth)	82.15	**96.00**	72.70	48.89	3.25
	HOPC [Rahmani et al., 2016b] (Depth)	85.49	92.44	47.55	60.58	3.25
	LARP-SO-logarithm map [Vemulapalli and Chellappa, 2016] (Skel.)	88.69	92.96	69.12	51.96	–
	LARP-SO-unwrapping while rolling [Vemulapalli and Chellappa, 2016] (Skel.)	88.47	94.09	69.12	53.05	–
	LARP-SO-FTP [Vemulapalli and Chellappa, 2016] (Skel.)	89.40	94.67	76.96	50.41	2.50
	HDG-hod [Wang, 2017] (Depth)	66.22	81.20	26.47	44.35	–
	HDG-hodg [Wang, 2017] (Depth)	70.34	90.98	50.98	54.23	–
	HDG-jpd [Wang, 2017] (Skel.)	55.54	53.78	46.08	40.88	–
	HDG-jmv [Wang, 2017] (Skel.)	62.40	84.87	41.18	51.02	–
	HDG-hod+hodg [Wang, 2017] (Depth)	71.81	90.96	51.96	55.17	–
	HDG-jpd+jmv [Wang, 2017] (Skel.)	65.57	84.93	49.02	55.57	–
	HDG-hod+hodg+jpd [Wang, 2017] (Depth + Skel.)	72.06	90.72	51.47	56.41	–
	HDG-hod+hodg+jmv [Wang, 2017] (Depth + Skel.)	75.41	92.27	49.51	58.80	–
	HDG-hodg+jpd+jmv [Wang, 2017] (Depth + Skel.)	75.00	92.28	52.94	59.82	–
	HDG-all features [Wang, 2017] (Depth + Skel.)	75.45	92.13	51.96	60.33	4.00
	SCK+DCK [Koniusz et al., 2016a] (Skel.)	**89.47**	**96.00**	**89.22**	**61.52**	1.00
Deep learning features	Frames+CNN [Ke et al., 2017b] (Skel.)	60.73	73.71	58.82	46.47	–
	Clips+CNN+Pooling [Ke et al., 2017b] (Skel.)	67.64	74.86	58.82	46.47	–
	Clips+CNN+Concatenation [Ke et al., 2017b] (Skel.)	71.27	78.29	61.76	53.85	–
	Clips+CNN+MTLN [Ke et al., 2017b] (Skel.)	73.82	79.43	67.65	54.81	2.25
	HPM+AP [Rahmani and Mian, 2016] (Depth)	56.73	56.11	44.12	42.32	–
	HPM+TM [Rahmani and Mian, 2016] (Depth)	72.00	**98.33**	44.12	54.78	2.50
	1-layer RNN [Shahroudy et al., 2016a] (Skel.)	18.02	32.76	54.90	14.27	–
	2-layer RNN [Shahroudy et al., 2016a] (Skel.)	27.80	56.13	54.91	35.36	–
	1-layer LSTM [Shahroudy et al., 2016a] (Skel.)	62.26	67.14	61.77	50.81	–
	2-layer LSTM [Shahroudy et al., 2016a] (Skel.)	65.33	73.72	63.24	46.78	–
	1-layer P-LSTM [Shahroudy et al., 2016a] (Skel.)	70.50	70.86	61.76	**55.16**	–
	2-layer P-LSTM [Shahroudy et al., 2016a] (Skel.)	69.35	72.00	67.65	50.81	2.75
	Our implementation with modified hyperparam. values:					
	1-layer LSTM (8 segments, 100 hidden neurons) (Skel.)	64.75	73.14	58.82	52.58	–
	2-layer P-LSTM (10 segments, 50 hidden neurons) (Skel.)	66.09	75.43	67.65	50.00	–
	2-layer P-LSTM (10 segments, 100 hidden neurons) (Skel.)	67.43	71.43	54.41	50.32	–
	2-layer P-LSTM (20 segments, 50 hidden neurons) (Skel.)	73.18	71.43	52.94	49.68	–
	2-layer P-LSTM (20 segments, 100 hidden neurons) (Skel.)	70.50	71.43	58.82	53.55	–
	IndRNN (4 layers) [Li et al., 2018b] (Skel.)	71.50	90.05	51.72	44.63	–
	IndRNN (6 layers) [Li et al., 2018b] (Skel.)	72.91	89.53	57.03	42.66	–
	Our improved results:					
	IndRNN (4 layers, with jpd) (Skel.)	76.34	82.66	**84.69**	52.09	–
	IndRNN (6 layers, withjpd) (Skel.)	**77.47**	86.88	80.16	51.34	2.00
	ST-GCN* [Yan et al., 2018] (Skel.)	27.64 (69.09)	20.00 (77.14)	23.53 (70.59)	22.12 (45.83)	–
	Our improved results:					
	ST-GCN* (with jpd) (Skel.)	18.18 (64.00)	54.16 (96.57)	26.47 (67.65)	36.54 (70.51)	4.75

*For ST-GCN, the numbers inside the parentheses denote the top-5 accuracy.

2014c] because we used $1\times1\times5 = 5$ cells [Wang, 2017] instead of $2\times2\times5 = 20$ cells to store the joint motion features.

For deep learning methods, the 1-layer P-LSTM method (8 video segments, 50 hidden neurons) outperformed others on the UWA3D Activity dataset. In general, a 1-layer LSTM with more hidden neurons performed better than a 1-layer LSTM with fewer neurons, and P-LSTM performed better than the traditional LSTM and RNN (see the last column in Table 2.4). The 2-layer P-LSTM (20 video segments, 50 hidden neurons) achieved the best recognition accuracy on the MSRAction3D dataset and HPM+TM outperformed on the 3D Action Pairs dataset. Comparing the last 5 rows of Table 2.4 for different variants of 2-layer P-LSTM shows that having more video segments and/or hidden neurons does not guarantee better performance. The reasons are: (i) more video segments have less averaging effect and so it is likely that noisy video frames with unreliable skeletal information would be used for feature

representation; (ii) having too many hidden neurons would cause overfitting in the training process.

Using the `jpd` features instead of the raw 3D joint coordinates boosts the recognition accuracies for both IndRNN and ST-GCN on almost all the datasets. For the CAD-60 and MSRAction3D datasets, the 4-layer IndRNN and 6-layer IndRNN using `jpd` features are the top two performers. Compared to using the raw 3D joint coordinates, the improvement due to the use of `jpd` is 4.56% for IndRNN (6 layers) on the MSRAction3D dataset and 32.97% for IndRNN (4 layers) on the CAD-60 dataset.

The last column of Table 2.4 computed using Eq. (2.2) shows that SCK+DCK obtains average rank 1 score, followed closely by the 6-layer IndRNN with `jpd` with average rank 2 score. The ST-GCN method uses a more complex architecture having 9 layers of spatio-temporal graph convolutional operators, which require a large dataset for training. As all the datasets in Table 2.4 are quite small, its poor performance (average rank 4.75 score) is not unexpected.

2.5.2 NTU RGB+D Dataset

Table 2.5 summarizes the evaluation results for cross-subject and cross-view action recognition on the NTU RGB+D dataset, where methods are grouped into the handcrafted feature and deep learning feature categories. Top performing methods are highlighted in bold.

Among the methods using handcrafted features, SCK+DCK performed the best for both cross-subject action recognition and cross-view action recognition. Similarly to results on the four datasets shown in Table 2.4, combining more individual features in HDG resulted in higher recognition accuracy for both the cross-subject and cross-view action recognition.

Among deep learning methods, ST-GCN and the 6-layer IndRNN combined with the `jpd` features outperformed other deep learning methods in both cross-subject and cross-view action recognition. In particular, with `jpd`, ST-GCN became the top performer (achieving 83.36%) for cross-subject action recognition and IndRNN (6 layers) achieved the highest accuracy (89.0%) for cross-view action recognition. Compared to using the raw 3D skeleton joint coordinates, using the `jpd` features helps improve the recognition accuracies of both IndRNN and ST-GCN. For example, with `jpd` features, both the top-1 and top-5 accuracies of ST-GCN increased by 1.79% and 0.61% in the cross-subject experiment.

For the other deep learning methods, the recognition accuracies of 2-layer RNN, 2-layer LSTM and 2-layer P-LSTM are higher than those of 1-layer RNN, 1-layer LSTM and 1-layer P-LSTM. Similar to the results in Table 2.4, P-LSTM performed better than the traditional LSTM and RNN. HPM+AP and HPM+TM, on the other hand, did not perform so well. The reason is that both of these methods were trained given only 339 representative human poses from a human pose dictionary whereas the 60 action classes of NTU RGB+D dataset include many more human poses of higher complexity.

Table 2.5: Comparison of average recognition accuracies (percentage) for both cross-subject and cross-view action recognition on the NTU RGB+D Dataset.

	Method	Cross-subject	Cross-view
Hand-crafted features	HON4D [Oreifej and Liu, 2013] (Depth)	30.6	7.3
	HOPC [Rahmani et al., 2016b] (Depth)	40.3	30.6
	LARP-SO-FTP [Vemulapalli and Chellappa, 2016] (Skel.)	52.1	53.4
	HDG-`hod` [Wang, 2017] (Depth)	20.1	13.5
	HDG-`hodg` [Wang, 2017] (Depth)	23.0	25.2
	HDG-`jpd` [Wang, 2017] (Skel.)	27.8	35.9
	HDG-`jmv` [Wang, 2017] (Skel.)	38.1	50.0
	HDG-`hod+hodg` [Wang, 2017] (Depth)	24.6	26.5
	HDG-`jpd+jmv` [Wang, 2017] (Skel.)	39.7	51.9
	HDG-`hod+hodg+jpd` [Wang, 2017] (Depth + Skel.)	29.4	38.8
	HDG-`hod+hodg+jmv` [Wang, 2017] (Depth + Skel.)	39.0	57.0
	HDG-`hodg+jpd+jmv` [Wang, 2017] (Depth + Skel.)	41.2	57.2
	HDG-all features [Wang, 2017] (Depth + Skel.)	43.3	58.2
	SCK+DCK [Koniusz et al., 2016a] (Skel.)	**72.8**	**74.1**
Deep learning features	Frames+CNN [Ke et al., 2017b] (Skel.)	75.7	79.6
	Clips+CNN+Concatenation [Ke et al., 2017b] (Skel.)	77.1	81.1
	Clips+CNN+Pooling [Ke et al., 2017b] (Skel.)	76.4	80.5
	Clips+CNN+MTLN [Ke et al., 2017b] (Skel.)	79.6	84.8
	Clips+CNN+MTLN‡ [Ke et al., 2017b] (Skel.)	79.54	84.70
	HPM+AP [Rahmani et al., 2016b] (Depth)	40.2	42.2
	HPM+TM [Rahmani et al., 2016b] (Depth)	50.1	53.4
	1-layer RNN [Shahroudy et al., 2016a] (Skel.)	56.0	60.2
	2-layer RNN [Shahroudy et al., 2016a] (Skel.)	56.3	64.1
	1-layer LSTM [Shahroudy et al., 2016a] (Skel.)	59.1	66.8
	2-layer LSTM [Shahroudy et al., 2016a] (Skel.)	60.7	67.3
	1-layer P-LSTM [Shahroudy et al., 2016a] (Skel.)	62.1	69.4
	2-layer P-LSTM [Shahroudy et al., 2016a] (Skel.)	62.9	70.3
	2-layer P-LSTM‡ [Shahroudy et al., 2016a] (Skel.)	63.02	70.39
	IndRNN (4 layers) [Li et al., 2018b] (Skel.)	78.6	83.8
	IndRNN (6 layers) [Li et al., 2018b] (Skel.)	81.8	88.0
	Our improved results:		
	IndRNN (4 layers, with `jpd`)(Skel.)	79.5	84.5
	IndRNN (6 layers, with `jpd`)(Skel.)	83.0	**89.0**
	ST-GCN* [Yan et al., 2018] (Skel.)	81.57 (96.85)	88.76 (98.83)
	Our improved results:		
	ST-GCN* (with `jpd`) (Skel.)	**83.36** (97.46)	88.84 (98.87)

‡Our implementations for reproducing original authors' experiment results.
* For ST-GCN, the numbers inside the parentheses denote the top-5 accuracy.

2.5.3 UWA3D Multiview Activity II Dataset

The ten algorithms were compared on the UWA3D Multiview Activity II dataset using cross-view action recognition. Table 2.6 summarizes the results. The top 2 action recognition algorithms are highlighted in bold in each column for the handcrafted feature and deep learning feature categories.

For the methods using handcrafted features, the HDG-all features performed the best for cross-view action recognition (the last column in Table 2.6) followed by other HDG variants that use skeletons and depth. Among skeleton-only methods, HDG-`jpd+jmv` was the second best performer followed by SCK+DCK, HDG-`jmv`, and LARP-SO-FTP, which all performed better than depth-based features such as HON4D, HDG-`hod`, HDG-`hodg` and HDG-`hod+hodg`. According to the table, using one or both skeleton-based features (HDG-`jpd` and/or HDG-`jmv`) in HDG improved its results.

For deep learning methods, HPM+TM and HPM+AP achieved the highest results. Although Clips+CNN+MTLN performed better than other 'Clips' variants, it did not perform as good as HPM+TM and HPM+AP, due to the limited number of video samples in the dataset. P-LSTM performed better than the traditional LSTM and RNN. We noticed that stacking more layers for IndRNN or using `jpd` features for

Table 2.6: Comparison of average recognition accuracies (percentage) for cross-view action recognition on the UWA3D Multiview Activity II dataset.

	Training view	V_1 & V_2		V_1 & V_3		V_1 & V_4		V_2 & V_3		V_2 & V_4		V_3 & V_4		Average
	Testing view	V_3	V_4	V_2	V_4	V_2	V_3	V_1	V_4	V_1	V_3	V_1	V_2	
Hand-crafted features	HON4D [Oreifej and Liu, 2013] (Depth)	31.1	23.0	21.9	10.0	36.6	32.6	47.0	22.7	36.6	16.5	41.4	26.8	28.9
	HOPC [Rahmani et al., 2016b] (Depth)	25.7	20.6	16.2	12.0	21.1	29.5	38.3	13.9	29.7	7.8	41.3	18.4	22.9
	Holistic HOPC* [Rahmani et al., 2016b] (Depth)	32.3	25.2	27.4	17.0	38.6	38.8	42.9	25.9	36.1	27.0	42.2	28.5	31.8
	Local HOPC+STK-D* [Rahmani et al., 2016b] (Depth)	52.7	51.8	**59.0**	**57.5**	42.8	44.2	58.1	38.4	63.2	43.8	66.3	48.0	52.2
	LARP-SO-logarithm map [Vemulapalli and Chellappa, 2016] (Skel.)	48.2	47.4	45.5	44.9	46.3	52.7	62.2	46.3	57.7	45.8	61.3	40.3	49.9
	LARP-SO-unwrapping while rolling [Vemulapalli and Chellappa, 2016] (Skel.)	50.4	45.7	44.0	44.5	40.8	49.6	57.4	44.4	57.6	47.4	59.2	40.8	48.5
	LARP-SO-FTP [Vemulapalli and Chellappa, 2016] (Skel.)	54.9	55.9	50.0	54.9	48.1	56.0	66.5	**57.2**	62.5	54.0	68.9	43.6	56.0
	HDG-`hod` [Wang, 2017] (Depth)	22.5	17.4	12.5	10.0	19.6	20.4	26.7	13.0	18.7	10.0	27.9	17.2	18.0
	HDG-`hodg` [Wang, 2017] (Depth)	26.9	34.2	20.3	18.6	34.7	26.7	41.0	29.2	29.4	11.8	40.7	28.8	28.5
	HDG-`jpd` [Wang, 2017] (Skel.)	36.3	32.4	31.8	35.5	34.4	38.4	44.2	30.0	44.5	33.7	44.4	34.0	36.6
	HDG-`jmv` [Wang, 2017] (Skel.)	57.2	59.3	**59.3**	54.3	56.8	50.6	63.4	52.4	65.7	53.7	67.7	56.9	58.1
	HDG-`hod+hodg` [Wang, 2017] (Depth)	26.6	33.6	17.9	19.3	34.4	26.2	40.5	27.6	28.6	11.6	38.4	29.0	27.8
	HDG-`jpd+jmv` [Wang, 2017] (Skel.)	**61.0**	61.8	**59.3**	56.0	60.0	57.4	68.8	54.2	**71.1**	**57.2**	69.7	59.0	**61.3**
	HDG-`hod+hodg+jpd` [Wang, 2017] (Depth + Skel.)	31.0	43.5	25.7	21.4	45.9	31.1	53.2	35.7	38.0	11.6	49.7	38.3	35.4
	HDG-`hod+hodg+jmv` [Wang, 2017] (Depth + Skel.)	59.0	**62.2**	58.1	52.0	62.5	57.1	66.0	54.2	67.7	52.7	70.3	**61.1**	60.2
	HDG-`hodg+jpd+jmv` [Wang, 2017] (Depth + Skel.)	58.2	61.8	54.8	47.6	**63.5**	**58.7**	**69.0**	52.3	64.9	47.1	67.2	59.4	58.7
	HDG-all features [Wang, 2017] (Depth + Skel.)	**60.9**	**64.3**	57.9	54.6	**62.6**	**59.2**	68.9	55.8	69.8	**55.2**	**71.8**	**62.6**	**61.9**
	SCK+DCK [Koniusz et al., 2016a] (Skel.)	56.0	61.8	50.0	**61.8**	54.1	56.7	**71.4**	**56.9**	**72.9**	49.6	**71.7**	44.0	58.9
Deep learning features	Frames+CNN [Ke et al., 2017b] (Skel.)	30.4	29.8	27.8	23.8	31.7	27.2	39.2	27.8	31.4	23.2	38.8	28.6	30.0
	Clips+CNN+Pooling [Ke et al., 2017b] (Skel.)	31.6	33.3	30.6	27.8	35.3	29.6	44.3	31.3	35.3	23.2	43.1	31.0	33.0
	Clips+CNN+Concatenation [Ke et al., 2017b] (Skel.)	33.2	36.9	32.5	29.8	36.9	31.2	46.3	31.0	39.2	23.6	45.5	31.7	34.8
	Clips+CNN+MTLN [Ke et al., 2017b] (Skel.)	36.4	38.9	34.1	30.6	37.7	33.2	46.7	31.3	38.8	25.6	49.8	33.7	36.4
	HPM+AP [Rahmani and Mian, 2016] (Depth)	**68.3**	51.7	**60.2**	**62.2**	38.7	**50.0**	**58.7**	37.8	**70.6**	**61.6**	**74.0**	**55.6**	**57.5**
	HPM+TM [Rahmani and Mian, 2016] (Depth)	**81.7**	**76.4**	**74.1**	**78.7**	**57.9**	**69.4**	**75.8**	**62.9**	**81.4**	**79.9**	**83.3**	**73.7**	**74.6**
	1-layer RNN [Shahroudy et al., 2016a] (Skel.)	11.4	11.1	10.0	14.9	11.9	13.3	11.1	15.2	10.3	12.2	9.8	12.4	12.0
	2-layer RNN [Shahroudy et al., 2016a] (Skel.)	23.4	21.6	27.3	22.7	26.3	20.3	21.7	19.5	20.7	24.8	27.0	21.5	23.1
	1-layer LSTM [Shahroudy et al., 2016a] (Skel.)	28.9	12.8	19.2	15.5	20.0	33.1	48.2	16.5	43.3	15.5	38.7	16.4	25.7
	2-layer LSTM [Shahroudy et al., 2016a] (Skel.)	29.7	16.3	20.4	12.8	26.8	30.5	53.0	11.0	37.0	17.1	47.4	20.0	26.8
	1-layer P-LSTM [Shahroudy et al., 2016a] (Skel.)	26.8	23.5	22.4	22.7	24.4	31.3	49.6	20.3	38.3	19.5	45.9	17.6	28.5
	2-layer P-LSTM [Shahroudy et al., 2016a] (Skel.)	30.1	24.7	25.2	21.5	24.4	37.4	51.0	22.7	43.1	21.1	47.8	17.2	30.5
	Our implementation with modified hyperparam. values:													
	2-layer P-LSTM (10 segments, 50 hidden neurons) (Skel.)	23.5	24.3	22.4	25.5	25.6	30.9	48.6	21.1	41.5	19.5	43.9	15.6	28.5
	2-layer P-LSTM (10 segments, 100 hidden neurons) (Skel.)	21.1	22.7	24.8	21.1	22.4	33.7	47.8	21.1	45.1	22.0	49.4	17.6	29.1
	2-layer P-LSTM (20 segments, 50 hidden neurons) (Skel.)	25.2	18.3	24.8	21.5	26.4	30.5	49.0	19.1	41.5	24.4	43.5	14.8	28.3
	2-layer P-LSTM (20 segments, 100 hidden neurons) (Skel.)	27.6	24.3	24.8	21.9	28.4	32.9	44.3	24.3	44.3	20.3	45.9	15.6	29.6
	IndRNN (4 layers) [Li et al., 2018b] (Skel.)	34.3	**53.8**	35.2	42.5	**39.1**	38.9	49.2	**42.5**	46.0	27.1	48.6	30.9	40.7
	IndRNN (6 layers) [Li et al., 2018b] (Skel.)	30.7	47.2	32.2	36.0	38.8	35.4	44.5	37.9	40.6	23.9	39.2	25.2	36.0
	IndRNN (4 layers, with `jpd`) (Skel.)	33.5	40.2	26.9	40.9	30.4	41.1	50.7	36.7	46.1	24.6	49.0	23.0	36.9
	IndRNN (6 layers, with `jpd`) (Skel.)	29.4	36.3	23.9	38.2	26.0	36.8	45.4	33.0	41.2	20.5	44.7	18.6	32.8
	ST-GCN [Yan et al., 2018] (Skel.)	36.4	26.2	20.6	30.2	33.7	22.4	43.1	26.6	16.9	12.8	26.3	36.5	27.6
	ST-GCN (with `jpd`) (Skel.)	29.6	21.8	15.5	19.1	17.1	18.8	35.3	14.3	31.0	14.8	13.3	15.5	20.5

*This result is obtained from [Rahmani et al., 2016b] for comparison.

both IndRNN and ST-GCN did not help increase the recognition accuracy. This is due to the lack of representative training videos (compared to the results on the NTU RGB+D dataset in cross-view action recognition).

Fig. 2.3 shows a confusion matrix for the HDG-all representation on the UWA3D Multiview Activity II dataset when V_3 and V_4 views were used for training and V_1 was used for testing. According the figure, the algorithm was confused by a few actions which have similar motion trajectories. For example, *one hand waving* is similar to *two hand waving* and *two hand punching*; *walking* is very similar to *irregular walking*; *bending* and *putting down* have very similar appearance; *sneezing* and *coughing* are very similar actions.

2.6 Discussions

2.6.1 Single-view versus cross-view

Table 2.7 summarizes the performance of each algorithm for both single-view action recognition and cross-view action recognition. One can observe from the table that some algorithms such as HON4D, LARP-SO-FTP, Clips+CNN+MTLN, P-LSTM, and SCK+DCK performed better for single-view action recognition but performed slightly worse in cross-view action recognition. For cross-subject action recognition,

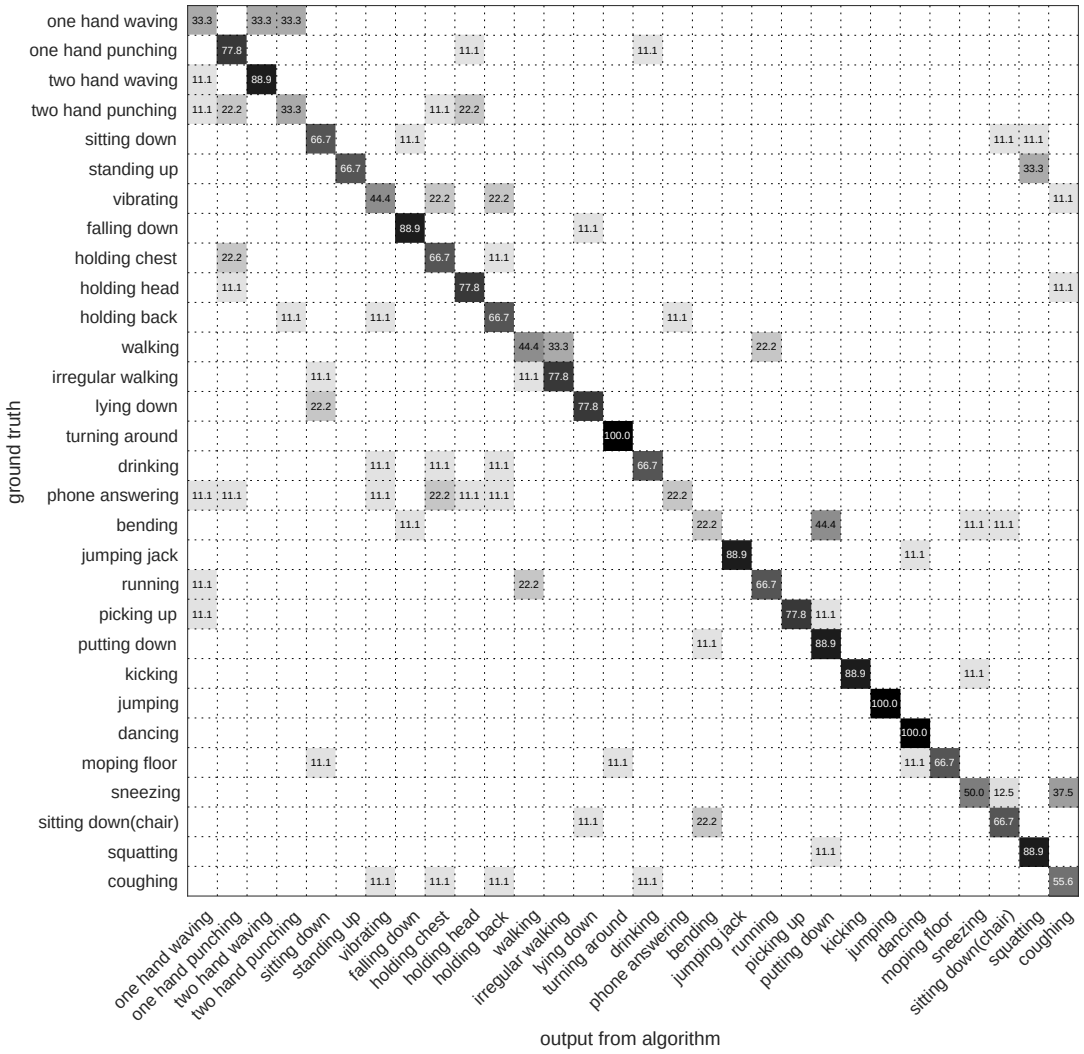
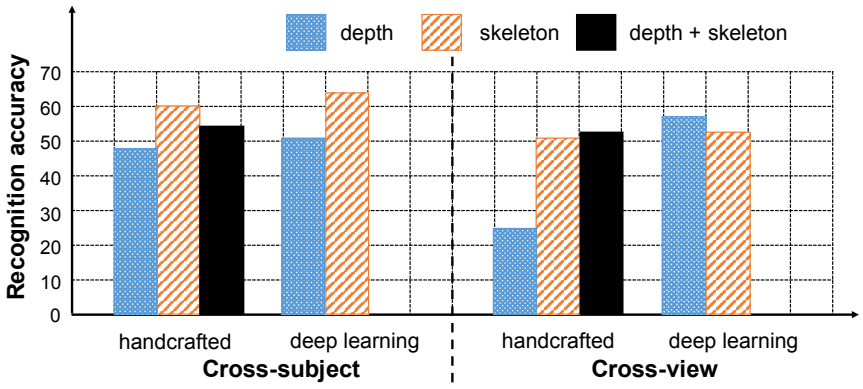


Figure 2.3: Confusion matrix for HDG-all features on the UWA3D Multiview Activity II dataset when V_3 and V_4 were used for training and V_1 was used for testing. For each action class along the diagonal, the darker is the colour, the higher is the recognition accuracy.

SCK+DCK outperformed all other algorithms; for cross-view action recognition, HDG-all, SCK+DCK and our improved IndRNN with `jpd` features all performed well (having the same average rank values).

Fig. 2.4 summarizes the average accuracy of all the algorithms grouped under the handcrafted and deep learning feature categories. On average, we found that the recognition accuracy for cross-view action recognition is lower than cross-subject action recognition using handcrafted features, with the poorest performance from depth-based features. Combining both depth- and skeleton-based features together helps improve cross-view action recognition and gives a similar performance for cross-subject action recognition. Overall, the performance on cross-view recognition



(a) (b)

Figure 2.4: The average recognition accuracy (in percentage) of methods using handcrafted and deep learning features for cross-subject and cross-view recognition. Numbers of methods using handcrafted (i) depth-based features: 7; (ii) skeleton-based features: 7; (iii) depth+skeleton-based features: 4. Numbers of methods using deep learning (i) depth-based features: 2; (ii) skeleton-based features: 20; (iii) depth+skeleton-based features: 0 (see Tables 2.4–2.6).

Table 2.7: The average recognition accuracies / [AVRank] of all the ten algorithms for both cross-subject and cross-view action recognition.

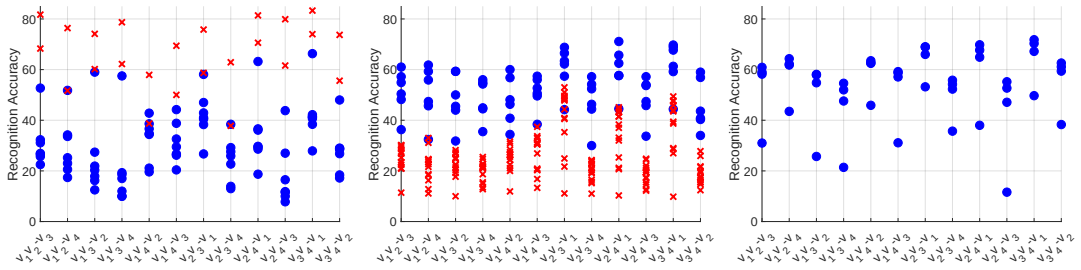
Algorithms	Cross-subject†	Cross-view‡
HON4D [Oreifej and Liu, 2013] (Depth)	52.8 / [3.6]	18.1 / [5.0]
HDG-all [Wang, 2017] (Depth + Skel.)	56.6 / [3.8]	60.1 / **[1.5]**
HOPC [Rahmani et al., 2016b] (Depth)	55.9 / [3.4]	41.4 / [4.0]
LARP-SO-FTP [Vemulapalli and Chellappa, 2016] (Skel.)	65.0 / [2.4]	54.7 / [3.0]
SCK+DCK [Koniusz et al., 2016a] (Skel.)	78.7 / **[1.0]**	66.5 / **[1.5]**
HPM+TM [Rahmani and Mian, 2016] (Depth)	58.7 / [3.0]	64.0 / [3.0]
Clips+CNN+MTLN [Ke et al., 2017b] (Skel.)	74.3 / [2.4]	60.6 / [3.0]
P-LSTM [Shahroudy et al., 2016a] (Skel.)	64.0 / [3.0]	50.4 / [4.0]
IndRNN [Li et al., 2018b](Skel.)	72.6 / [2.4]	62.1 / [2.0]
IndRNN with `jpd` (Skel.)	77.6 / **[2.0]**	60.8 / **[1.5]**
ST-GCN [Yan et al., 2018](Skel.)	52.4 / [4.2]	58.2 / [3.5]
ST-GCN with `jpd` (Skel.)	57.7 / [4.0]	54.7 / [3.0]

†For cross-subject action recognition, the performance of each algorithm is computed by averaging all the recognition accuracies / rank values on the MSRAction3D, 3D Action Pairs, CAD-60, UWA3D Activity and NTU RGB+D datasets. ‡For cross-view action recognition, they are computed over the UWA3D Multiview Activity II and NTU RGB+D datasets.

is lower than cross-subject recognition, except for the deep learning depth-based feature methods. It should be noted that there are only two methods (HPM+AP and HPM+TM) falling into this category and they performed well on the UWA3D Multiview Activity II dataset and reasonably well on the NTU RGB+D dataset.

2.6.2 Influence of camera views in cross-view evaluation

To analyze the influence of using different combinations of camera views in training and testing, we classified all the algorithms in Table 2.6 into 3 feature types:



(a) Depth-based (b) Skeleton-based (c) Depth+Skeleton-based

Figure 2.5: Scatter plots showing the performance of cross-view action recognition on the UWA3D Multiview Activity II dataset. The blue dots and red crosses, respectively, represent methods using handcrafted features and deep learning features. On the horizontal axis of each plot, we use the notation V_iV_j-V_k to denote views i and j being used for training and view k being used for testing.

depth-based, skeleton-based, and depth+skeleton-based representations. Scatter plots showing the performance of all the algorithms on these three feature types are given in Fig. 2.5, where the blue dots and red crosses denote, respectively, algorithms using handcrafted features and deep learning features.

According to plots in Fig. 2.5 the recognition accuracy was high when the left view V_2 and right view V_3 were used for training and front view V_1 for testing (see axis tick mark V_2V_3-V_1). This is obvious as the viewing angle of the front view V_1 is between V_2 and V_3, as shown in Fig. 2.2. However, it is surprising that the recognition accuracy was even slightly higher for V_2V_4-V_1 and V_3V_4-V_1 (see Figs. 2.5(b) and 2.5(c)).

One can also notice that for depth-based methods which use handcrafted features, the recognition accuracy dropped the most when the left view V_2 and top view V_4 were used for training, and the right view V_3 was used for testing. The main reason for this behavior is that the visual appearances of an action in V_2 and V_3 are different, and it is very difficult to find the same view-invariant features from these views. Other combinations of views, such as V_1V_3-V_4, V_1V_3-V_2, and V_1V_2-V_4, also led to a lower performance (see the distribution of the blue dots in Fig. 2.5(a)).

2.6.3 Depth-based features versus skeleton-based features

Cross-subject action recognition. It seems that for the cross-subject action recognition, skeleton-based features outperformed depth-based features for both the handcrafted and deep learning feature categories (Fig. 2.4(a)). However, it is surprising that adding the depth-based features to skeleton-based features actually led to a slight decrease in the action recognition accuracy. The main reason is the background clutter and noise (for instance, see Fig. 2.1) in the depth sequences, making the depth-based features less representative for robust action recognition.

On the other hand, some skeleton-based algorithms such as LARP-SO-FTP and its variants performed well in human action recognition because they only extracted reliable features from those human body joints which have high confidence values.

The only exception is the HOPC algorithm which uses depth-based features and which performed better than skeleton-based features such as HDG-`jpd+jmv` in the cross-subject action recognition (see the average performance in the last column of Table 2.4). One must note that the HOPC algorithm is different from other depth-based algorithms as it treats the depth images as a 3D pointcloud. Such an approach allows the HOPC algorithm to estimate the orientations of the local surface patches on the body of the human subject to more robustly deal with viewpoint changes.

Cross-view action recognition. In Fig. 2.5(b), for the cross-view action recognition, methods using handcrafted features (blue dots) performed better than those using deep learning features (red crosses) for skeleton-based features. For depth-based features (Fig. 2.5(a)), no clear conclusion can be drawn for the handcrafted versus deep learning features.

Comparing the handcrafted features (blue dots) in both Figs. 2.5(a) and 2.5(b), we can see that algorithms using skeleton-based features produced better results than depth-based features. This is evident from Fig. 2.5(b) as the blue dots are well above the red crosses, whereas in Fig. 2.5(a) the blue dots are more spread out and they occupy mainly the lower region of the plot. Our pruning process of the HDG algorithm also proved that skeleton-based features are more robust than the depth-based algorithms in the cross-view action recognition.

For the plot in Fig. 2.5(c), the algorithms in our experiments did not use deep learning to compute depth+skeleton features, thus no comparison on the performance between using handcrafted and deep learning features is possible.

2.6.4 Handcrafted features versus deep learning features

As expected, our experiments confirm that deep learning methods (*e.g.*, Clips + CNN + MTLN, IndRNN, and ST-GCN) performed better on large datasets such as the NTU RGB+D dataset, which has more than 56,000 video sequences, but achieved a lower recognition accuracy on the other smaller datasets. The main reason for this behavior is the reliance of deep learning methods on large amount of data for training to avoid overfitting (in contrast to handcrafted methods). However, most existing action datasets have small numbers of performing subjects and action classes, and very few camera views due to high cost of collecting/annotating data.

We also found that the performance of most handcrafted methods is highly dependent on the datasets. For example, HON4D and HOPC performed better only on some specific datasets, such as MSRAction3D and 3D Action Pairs, but achieved lower performance on the NTU RGB+D dataset. This means that features that have been handcrafted to work on one dataset may not be transferable to another dataset. Moreover, as handcrafted methods prevent overfitting well, they may lack the capacity to learn from big data. With larger benchmark datasets becoming available to the research community, future research trend is more likely to shift to using deep learning features with efforts being put into tuning their hyperparameters.

2.6.5 'Quo Vadis, action recognition?'

Based on the methods evaluated in this chapter, we see the following trends in human action recognition. Handcrafted features progressed from global descriptors (*e.g.*, HON4D in 2013) to local descriptors (*e.g.*, HOPC in 2016) and combinations of global and local feature representation (*e.g.*, HDG in 2014). More recent works focus on designing robust 3D human joint-based representations (*e.g.*, LARP-SO-FTP and SCK+DCK in 2016). Moreover, researchers have adopted features extracted from skeleton sequences as they are easier to process and analyze than depth videos. We also note that deep learning representations are evolving from basic neural networks (*e.g.*, traditional RNN and LSTM) to adapted and/or dedicated networks that rely on a pre-trained network (*e.g.*, HPM+AP, HPM+TM, and Clips+CNN+MTLN). Researchers also modify basic neural network models to improve their learning capability for action recognition (*e.g.*, P-LSTM and IndRNN). Recently, new models such as ST-GCN were devised to robustly model the human actions in large datasets. Given the surge of large datasets and powerful GPU machines, the newest methods learn representations in an end-to-end manner.

2.7 Conclusion

We have presented, analyzed and compared 10 state-of-the-art algorithms in 3D action recognition on six benchmark datasets. These algorithms cover the use of handcrafted and deep learning features computed from depth and skeleton video sequences. We believe that our comparison results will be useful for future researchers interested in designing robust representations for recognizing human actions from videos captured by the Kinect camera and/or similar action sequence capturing sensors.

In our study, we found that skeleton-based features are more robust than depth-based features for both cross-subject action recognition and cross-view action recognition. Handcrafted features performed better than deep learning features given smaller datasets. However, deep learning methods achieved very good results if trained on large datasets. While accuracy as high as 90% has been achieved by some algorithms on cross-subject action recognition, the average accuracy on cross-view action recognition is much lower.

From our literature review and comprehensive experiments, we see that many research papers on human action recognition have already attempted to tackle the challenging issues mentioned in the introduction of the chapter. Examples include using a skeleton model or treating depth maps as 3D pointcloud to overcome the changes of viewpoint; using Fourier temporal pyramid or dynamic time warping to deal with different speeds of execution of actions; using depth videos to overcome changes in lighting conditions and eliminate visual appearances that may be irrelevant to the actions performed. While these papers all report promising action recognition results, new and more robust action recognition algorithms are still required, given that there is a pressing demand for using these algorithms in real and new environments.

Chapter 3

Hallucinating IDT Descriptors and I3D Optical Flow Features

In this chapter, we revive the use of old-fashioned handcrafted video representations for action recognition and put new life into these techniques via a CNN-based hallucination step. Despite of the use of RGB and optical flow frames, the I3D model (amongst others) thrives on combining its output with the Improved Dense Trajectory (IDT) and extracted with its low-level video descriptors encoded via Bag-of-Words (BoW) and Fisher Vectors (FV). Such a fusion of CNNs and handcrafted representations is time-consuming due to pre-processing, descriptor extraction, encoding and tuning parameters. Thus, we propose an end-to-end trainable network with streams which learn the IDT-based BoW/FV representations at the training stage and are simple to integrate with the I3D model. Specifically, each stream takes I3D feature maps ahead of the last 1D conv. layer and learns to 'translate' these maps to BoW/FV representations. Thus, our model can hallucinate and use such synthesized BoW/FV representations at the testing stage. We show that even features of the entire I3D optical flow stream can be hallucinated thus simplifying the pipeline. Our model saves 20–55h of computations and yields state-of-the-art results on four publicly available datasets.

3.1 Introduction

Action Recognition (AR) pipelines have transitioned from the use of handcrafted descriptors [Dalal et al., 2006; Scovanner et al., 2007; Kläser et al., 2008; Wang et al., 2011, 2013b; Wang and Schmid, 2013] to CNN models such as the two-stream network [Simonyan and Zisserman, 2014], 3D spatio-temporal features [Tran et al., 2015], spatio-temporal ResNet [Feichtenhofer et al., 2016a] and the I3D network pre-trained on Kinetics-400 [Carreira and Zisserman, 2017]. Such CNNs operate on RGB/optical flow videos thus failing to capture some domain-specific information which sophisticated low-level representations capture by design. One prominent example are Improved Dense Trajectory (IDT) descriptors [Wang and Schmid, 2013] which are typically encoded with Bag-of-Words (BoW) [Sivic and Zisserman, 2003; Csurka et al., 2004] or Fisher Vectors (FV) [Perronnin and Dance, 2007; Perronnin et al., 2010] and

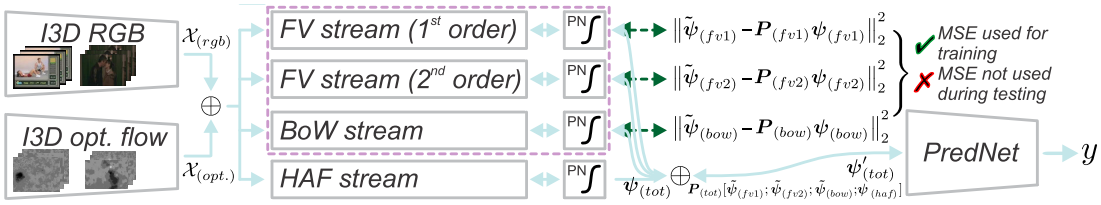


Figure 3.1: The overview of our pipeline. We remove the prediction and the last 1D conv. layers from I3D RGB and optical flow streams, concatenate ($\oplus$) the 1024×7 feature representations $\mathcal{X}_{(rgb)}$ and $\mathcal{X}_{(opt.)}$, and feed them into our *Fisher Vector* (*FV*), *Bag-of-Words* (*BoW*), and *the High Abstraction Features* (*HAF*) streams followed by the *Power Normalization* (*PN*) blocks. The resulting feature vectors $\tilde{\boldsymbol{\psi}}_{(fv1)}$, $\tilde{\boldsymbol{\psi}}_{(fv2)}$, $\tilde{\boldsymbol{\psi}}_{(bow)}$ and $\boldsymbol{\psi}_{(haf)}$ are concatenated ($\oplus$) and fed into our *Prediction Network* (*PredNet*). By ✓, we indicate that the three Mean Square Error (MSE) losses are only applied at the training stage to train our FV (first- and second-order components) and BoW hallucinating streams (indicated in dashed red). By ✗, we indicate that the MSE losses are switched off at the testing stage. Thus, we hallucinate $\tilde{\boldsymbol{\psi}}_{(fv1)}$, $\tilde{\boldsymbol{\psi}}_{(fv2)}$ and $\tilde{\boldsymbol{\psi}}_{(bow)}$, and pass them to PredNet together with $\boldsymbol{\psi}_{(haf)}$ to obtain labels y. The original training FV and BoW feature vectors (used only during training) are denoted by $\boldsymbol{\psi}_{(fv1)}$, $\boldsymbol{\psi}_{(fv2)}$ and $\boldsymbol{\psi}_{(bow)}$, while $\boldsymbol{P}$ are count sketch projecting matrices (see text for details).

fused with CNNs [Fernando and Gould, 2016; Cherian et al., 2017b, 2018; Wang and Cherian, 2018; Choutas et al., 2018] at the classifier which improves results due to several sophisticated steps of IDT: (i) camera motion estimation, (ii) motion descriptor modeling along motion trajectories estimated by the optical flow, (iii) pruning inconsistent matches, (iv) focusing on human motions via a human detector, (v) combination of IDT with powerful and highly complementary to each other video descriptors such as Histogram of Oriented Gradients (HOG) [Freeman and Roth, 1994; Kläser et al., 2008], Histogram of Optical Flow (HOF) [Dalal et al., 2006] and Motion Boundary Histogram (MBH) [Wang et al., 2013b] *e.g.*, HOF and MBH contain zero- and first-order motion statistics [Wang and Schmid, 2013].

However, extracting dense trajectories and corresponding video descriptors is costly due to several off-line/CPU-based steps. Motivated by this shortcoming, we propose simple trainable CNN streams on top of a CNN network (in our case I3D [Carreira and Zisserman, 2017]) which learn to 'translate' the I3D output into IDT-based BoW and FV global video descriptors. We can even 'translate' the I3D RGB output into I3D Optical Flow Features (OFF). At the testing stage, our so-called BoW, and FV and OFF streams (on top of I3D) are able to hallucinate such global descriptors which we feed into the final layer preceding a classifier. We show that IDT/OFF representations can be synthesized by our network thus removing the need of actually computing them which simplifies the AR pipeline. With a handful of convolutional/FC layers and basic CNN building blocks, our representation rivals sophisticated AR pipelines that aggregate features frame-by-frame *e.g.*, HOK [Cherian et al., 2017b] and rank-pooling [Fernando and Gould, 2016; Cherian et al., 2018; Wang and Cherian, 2018; Choutas et al., 2018]. Below, we detail our contributions:

I. We are the first to propose that old-fashioned IDT-based BoW and FV global video descriptors can be learned via simple dedicated CNN-streams at the training stage and simply hallucinated for classification with a CNN action recognition pipeline during testing.

II. We show that even the I3D optical flow stream can be easily hallucinated from the I3D RGB stream.

III. We study various aspects of our model *e.g.*, the count sketch [Pham and Pagh, 2013] of features to avoid overfitting when fusing several streams and Power Normalization [Koniusz et al., 2013b, 2016c, 2018b] to prevent so-called burstiness in BoW, FV and CNNs, and we perform several experiments on four datasets.

Sections 3.2 and 3.3 introduce the background, notations and concepts. Sections 3.4 and 3.5 present our method and results.

3.2 Related Work

Below, we describe handcrafted spatio-temporal video descriptors and their encoding strategies, optical flow, and deep learning pipelines for video classification.

Handcrafted video representations. Early AR relied on spatio-temporal interest point detectors [Laptev, 2005; Dollár et al., 2005; Chakraborty et al., 2012; Willems et al., 2008; Li et al., 2014; Wang et al., 2011] and spatio-temporal descriptors [Dalal et al., 2006; Scovanner et al., 2007; Uijlings et al., 2014; Wang et al., 2011, 2013b; Wang and Schmid, 2013] which capture various appearance and motion statistics.

Spatio-temporal interest point detectors were developed for the task of identifying spatio-temporal regions of videos rich in motion patterns relevant to classification, thus providing sampling locations for local descriptors. The number of sampling points had a significant influence on the processing speed due to the volumetric nature of videos. Harris3D [Laptev, 2005], one of the earliest detectors, performs a search for extreme points in the spatio-temporal domain via the so-called structure tensor and the determinant-to-trace ratio test. Cuboid [Dollár et al., 2005], a faster detector, applies Gaussian and Gabor filters in spatial and temporal domains, respectively. Selective STIP [Chakraborty et al., 2012] extracts initial key-point candidates with the Harris corner detector followed by the candidate suppression with a so-called surround suppression mask. Hes-STIP, a more recent detector, uses integral videos and Hessian matrix to search the scale-space for local maxima of the signal. Evaluations and further reading on spatio-temporal detectors can be found in surveys [Gauglitz et al., 2011; Wang, 2017; Wang et al., 2019b].

One drawback of spatio-temporal interest point detectors is the sparsity of key-points and inability to capture long-term motion patterns. Thus, a Dense Trajectory (DT) [Wang et al., 2011] approach densely samples feature points in each frame to track them in the video (via optical flow). Then, multiple descriptors are extracted along trajectories to capture shape, appearance and motion cues. As DT cannot compensate for the camera motion, the IDT [Wang and Schmid, 2013; Wang et al.,

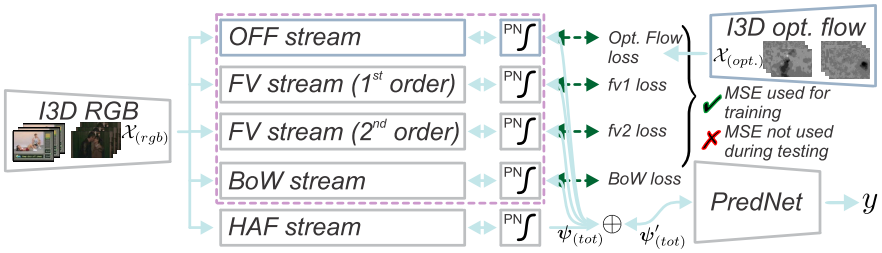


Figure 3.2: Hallucinating the Optical Flow Features (OFF).

2013b] estimates the camera motion to remove the global background motion. IDT also removes inconsistent matches via a human detector.

For spatio-temporal descriptors, IDT employs HOG [Freeman and Roth, 1994], HOF [Dalal et al., 2006] and MBH [Wang et al., 2013b]. HOG [Freeman and Roth, 1994] contains statistics of the amplitude of image gradients w.r.t. the gradient orientation. Thus, it captures the static appearance cues while its close cousin, HOG-3D [Kläser et al., 2008], is designed for spatio-temporal interest points. In contrast, HOF [Dalal et al., 2006] captures histograms of optical flow while MBH [Wang et al., 2013b] captures derivatives of the optical flow, thus it is highly resilient to the global camera motion whose cues cancel out due to derivatives. Thus, HOF and MBH contain the zero- and first-order optical flow statistics. Other spatio-temporal descriptors include SIFT3D [Scovanner et al., 2007], SURF3D [Willems et al., 2008] and LTP [Yeffet and Wolf, 2009].

In this work, we follow the standard practice, that is, we use the Improved Dense Trajectories [Wang et al., 2011; Cherian et al., 2017b; Choutas et al., 2018] and we encode them together with HOG, HOF, and MBH descriptors via BoW [Sivic and Zisserman, 2003; Csurka et al., 2004] and FV [Perronnin and Dance, 2007; Perronnin et al., 2010] which we describe below.

Descriptor encoding. BoW [Sivic and Zisserman, 2003; Csurka et al., 2004], a global image representation, is likely the oldest encoding strategy for local descriptors. It consists of (i) clustering with k-means for a collection of descriptor vectors from the training set to build so-called visual vocabulary, (ii) assigning each descriptor to its nearest cluster center from the visual dictionary, and (iii) aggregating the one-hot assignment vectors via average pooling. Similar models such as Soft Assignment (SA) [van Gemert et al., 2010; Koniusz and Mikolajczyk, 2011] and Localized Soft Assignment (LcSA) [Lingqiao et al., 2011; Koniusz et al., 2013b] use the Component Membership Probability (CMP) of GMM to assign each descriptor with some probability to visual words followed by average or non-linear pooling [Koniusz et al., 2013b; Wang et al., 2019c].

In this chapter, we chose the simplest BoW model [Csurka et al., 2004] with Power Normalization [Koniusz et al., 2013b] detailed in Section 3.3. BoW can be seen as zero-order statistics of FV [Perronnin and Dance, 2007; Perronnin et al., 2010], thus we also employ FV to capture first- and second-order statistics of local descriptors. FV builds a visual dictionary from training data via GMM. Then, a displacement/square

displacement of each descriptor vector w.r.t. each GMM component center is taken, normalized by its GMM standard deviation/variance to capture the first/second-order terms, and then soft-assigned via CMP to each GMM component.

Optical flow. As a key concept in AR from videos, optical flow is the distribution of velocities of movement of brightness pattern across frames [Horn and Schunck, 1981] such as the pattern of motion of objects, surfaces and edges in a visual scene caused by the relative motion between an observer and a scene [Hunter, 1980]. Early optical flow coped with small displacements via energy minimization [Horn and Schunck, 1981; Papenberg et al., 2006]. However, to capture informative motions of subjects/objects, optical flow needs to cope with large displacements [Alvarez et al., 2000]. As energy-based methods suffer from the local minima, local descriptor matching is used in Large Displacement Optical Flow (LDOF) [Brox and Malik, 2011]. Recent methods use non-rigid descriptor matching [Weinzaepfel et al., 2013], segment matching [Braux-Zin et al., 2013] or even edge-preserving interpolation [Revaud et al., 2015].

In this work, we are not concerned with the use of the newest possible optical flow. Thus, we opt for LDOF [Papenberg et al., 2006].

CNN-based action recognition. The success of AlexNet [Krizhevsky et al., 2012] and ImageNet [Russakovsky et al., 2015] sparked studies into AR with CNNs. Early models extracted per-frame representations followed by average pooling [Karpathy et al., 2014] which discards the temporal order. To fix such a shortcoming, frame-wise CNN scores were fed to LSTMs [Donahue et al., 2015]. Two-stream networks [Simonyan and Zisserman, 2014] compute representations per RGB frame and per 10 stacked optical flow frames. However, a more obvious extension is to model spatio-temporal 3D CNN filters [Ji et al., 2013; Tran et al., 2015; Feichtenhofer et al., 2016a; Varol et al., 2018].

The recent I3D model [Carreira and Zisserman, 2017] draws on the two-stream networks, 'inflates' 2D CNN filters pre-trained on ImageNet to spatio-temporal 3D filters, and implements temporal pooling across the inception module. In this chapter, we opt for the I3D network but our proposed layers are independent of the CNN design. We are concerned with 'absorbing' the old yet powerful IDT representations and/or optical flow features into CNN and hallucinating them at the test time.

Temporal aggregation. While two-stream networks [Simonyan and Zisserman, 2014] discard the temporal order and others use LSTMs [Donahue et al., 2015], many AR pipelines address the spatio-temporal aggregation. Rank pooling [Fernando et al., 2015; Fernando and Gould, 2016] projects frame-wise feature vectors onto a line such that the temporal order of vectors is preserved along the line. Subspace and kernel rank pooling [Cherian et al., 2018; Wang and Cherian, 2018] use projections into the RKHS in which the temporal order of frames is preserved. Another aggregation family captures second- or higher-order statistics [Cherian et al., 2017b; Koniusz et al., 2016a,c; Fang et al., 2019].

In this chapter, we are not concerned with temporal pooling. Thus, we use a 1D convolution (as in I3D [Carreira and Zisserman, 2017]).

Power Normalization family. BoW, FV and even CNN-based descriptors have to deal with the so-called burstiness defined as *'the property that a given visual element appears more times in an image than a statistically independent model would predict'* [Jégou et al., 2009], a phenomenon also present in video descriptors. Power Normalization [Koniusz et al., 2013b, 2021] is known to suppress the burstiness, and it has been extensively studied in the context of BoW [Koniusz et al., 2013b, 2021, 2016c, 2018b]. Moreover, a connection to Max-pooling was found in survey [Koniusz et al., 2013b] which also shows that the so-called MaxExp pooling is in fact a detector of *'at least one particular visual word being present in an image'*. According to papers [Koniusz et al., 2013b, 2018b], many Power Normalization functions are closely related. We outline Power Normalizations used in our work in Section 3.3.

3.3 Background

In our work, we use BoW/FV (training stage), as well as Power Normalization [Koniusz et al., 2013b, 2016c] and count sketches [Weinberger et al., 2009].

Notations. We use boldface uppercase letters to express matrices *e.g.*, $\boldsymbol{M}, \boldsymbol{P}$, regular uppercase letters with a subscript to express matrix elements *e.g.*, P_{ij} is the $(i,j)^{\text{th}}$ element of $\boldsymbol{P}$, boldface lowercase letters to express vectors, *e.g.* $\mathbf{x}, \boldsymbol{\phi}, \boldsymbol{\psi}$, and regular lowercase letters to denote scalars. Vectors can be numbered *e.g.*, $\boldsymbol{m}_1, ..., \boldsymbol{m}_K$ or $\mathbf{x}_n$, *etc.*, while regular lowercase letters with a subscript express an element of vector *e.g.*, m_i is the i^{th} element of $\boldsymbol{m}$. Operators ';' and $\oplus$ concatenate vectors *e.g.*, $\oplus_{i\in\mathcal{I}_K}\mathbf{v}_i = [\mathbf{v}_1; ...; \mathbf{v}_K]$ while $\mathcal{I}_d$ denotes an index set of integers $\{1, ..., d\}$.

3.3.1 Descriptor Encoding Schemes

Bag-of-Words [Sivic and Zisserman, 2003; Csurka et al., 2004] assigns each local descriptor $\boldsymbol{x}$ to the closest visual word from $\boldsymbol{M} = [\boldsymbol{m}_1, ..., \boldsymbol{m}_K]$ built via k-means. In order to obtain mid-level feature $\boldsymbol{\phi}$, we solve:

$$\begin{aligned} &\boldsymbol{\phi} = \arg\min_{\boldsymbol{\phi}'} \left\|\boldsymbol{x} - \boldsymbol{M}\boldsymbol{\phi}'\right\|_2^2, \\ &s.\,t.\ \ \boldsymbol{\phi}' \in \{0,1\}, \mathbb{1}^T\boldsymbol{\phi}' = 1. \end{aligned} \tag{3.1}$$

Fisher Vector Encoding [Perronnin and Dance, 2007; Perronnin et al., 2010] uses a Mixture of K Gaussians from a GMM used as a dictionary. It performs descriptor coding w.r.t. to Gaussian components $G(w_k, \boldsymbol{m}_k, \boldsymbol{\sigma}_k)$ which are parametrized by mixing probability, mean, and on-diagonal standard deviation. The first- and second-order features $\boldsymbol{\phi}_k, \boldsymbol{\phi}'_k \in \mathbb{R}^D$ are :

$$\boldsymbol{\phi}_k = (\boldsymbol{x} - \boldsymbol{m}_k)/\boldsymbol{\sigma}_k, \ \ \boldsymbol{\phi}'_k = \boldsymbol{\phi}_k^2 - 1. \tag{3.2}$$

Concatenation of per-cluster features $\boldsymbol{\phi}^*_k \in \mathbb{R}^{2D}$ forms the mid-level feature $\boldsymbol{\phi} \in \mathbb{R}^{2KD}$:

$$\boldsymbol{\phi} = [\boldsymbol{\phi}_1^*; ...; \boldsymbol{\phi}_K^*], \quad \boldsymbol{\phi}_k^* = \frac{p(\boldsymbol{m}_k|\boldsymbol{x}, \theta)}{\sqrt{w_k}} \left[\boldsymbol{\phi}_k; \boldsymbol{\phi}_k'/\sqrt{2}\right], \tag{3.3}$$

where p and θ are the component membership probabilities and parameters of GMM, respectively. For each descriptor $\boldsymbol{x}$ of dimensionality D (after PCA), its encoding $\boldsymbol{\phi}$ is of $2KD$ dim. as $\boldsymbol{\phi}$ contains first- and second-order statistics.

3.3.2 Pooling a.k.a. Aggregation

Traditionally, pooling is performed via averaging mid-level feature vectors $\boldsymbol{\phi}(\mathbf{x})$ corresponding to (local) descriptors $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ from a video sequence $\boldsymbol{\mathcal{X}}$, that is $\boldsymbol{\psi} = \text{avg}_{\mathbf{x}\in\boldsymbol{\mathcal{X}}} \boldsymbol{\phi}(\mathbf{x})$, and (optionally) applying the ℓ_2-norm normalization. In this chapter, we work with either sequences $\boldsymbol{\mathcal{X}}$ (for which the above step is used) or subsequences.

Proposition 1. *For subsequence pooling, let $\boldsymbol{\mathcal{X}}_{s,t} = \boldsymbol{\mathcal{X}}_{0,t} \setminus \boldsymbol{\mathcal{X}}_{0,s-1}$, where $\boldsymbol{\mathcal{X}}_{s,t}$ denotes a set of descriptors in the sequence $\boldsymbol{\mathcal{X}}$ counting from frame s up to frame t, where $0 \leq s \leq t \leq \tau$, $\boldsymbol{\mathcal{X}}_{0,-1} \equiv \emptyset$, and τ is the length of $\boldsymbol{\mathcal{X}}$. Moreover, let us compute an integral mid-level feature $\boldsymbol{\phi}_t' = \boldsymbol{\phi}_{t-1}' + \sum_{\mathbf{x}\in\boldsymbol{\mathcal{X}}_{t,t}} \boldsymbol{\phi}(\mathbf{x})$ which aggregates mid-level feature vectors from frame 0 to frame t, and $\boldsymbol{\phi}_{-1}'$ is an all-zeros vector. Then, the pooled subsequence is given by:*

$$\boldsymbol{\psi}_{s,t} = (\boldsymbol{\phi}_t' - \boldsymbol{\phi}_{s-1}') / (\|\boldsymbol{\phi}_t' - \boldsymbol{\phi}_{s-1}'\|_2 + \epsilon), \tag{3.4}$$

where $0 \leq s \leq t \leq \tau$ are the starting and ending frames of subsequence $\boldsymbol{\mathcal{X}}_{s,t}' \subseteq \boldsymbol{\mathcal{X}}$ and ϵ is a small constant. We normalize the pooled sequences/subseq. as described next.

3.3.3 Power Normalization

As alluded to in Section 3.2, we apply Power Normalizing functions to BoW and FV streams which hallucinate these two modalities (and HAF/OFF stream explained later). We investigate three operators $g(\boldsymbol{\psi}, \cdot)$ detailed by Remarks 1–3.

Remark 1. *AsinhE function [Koniusz et al., 2018b] is an extension of a well-known Power Normalization (Gamma) [Koniusz et al., 2018b] defined as $g(\boldsymbol{\psi}, \gamma) = \text{Sgn}(\boldsymbol{\psi})|\boldsymbol{\psi}|^\gamma$ for $0 < \gamma \leq 1$ to the operator with a smooth derivative and a parameter γ'. AsinhE is defined as the normalized Arcsin hyperbolic function:*

$$g(\boldsymbol{\psi}, \gamma') = \text{arcsinh}(\gamma'\boldsymbol{\psi}) / \text{arcsinh}(\gamma'). \tag{3.5}$$

Remark 2. *Sigmoid (SigmE), a Max-pooling approximation [Koniusz et al., 2018b], is an extension of the MaxExp operator defined as $g(\boldsymbol{\psi}, \eta) = 1 - (1 - \boldsymbol{\psi})^\eta$ for $\eta > 1$ to the operator with a smooth derivative, a response defined for real-valued $\boldsymbol{\psi}$ (rather than $\boldsymbol{\psi} \geq 0$), a parameter η' and a small const. ϵ':*

$$g(\boldsymbol{\psi}, \eta') = \frac{2}{1 + e^{-\eta'\boldsymbol{\psi}/(\|\boldsymbol{\psi}\|_2 + \epsilon')}} - 1. \tag{3.6}$$

Remark 3. *AxMin, a piece-wise linear form of SigmE [Koniusz et al., 2018b], is given as $g(\boldsymbol{\psi}, \eta'') = \text{Sgn}(\boldsymbol{\psi}) \min(\eta''\boldsymbol{\psi} / (\|\boldsymbol{\psi}\|_2 + \epsilon'), 1)$ for $\eta'' > 1$ and a small constant ϵ'.*

Despite the similar role of these three pooling operators, we investigate each of them as their interplay with end-to-end learning differs. Specifically, $\lim_{\psi \to \pm\infty} g(\psi, \cdot)$ for AsinhE and SigmE are $\pm\infty$ and ± 1, resp., thus their asymptotic behavior differs. Moreover, AxMin is non-smooth and relies on the same gradient re-projection properties as ReLU.

3.3.4 Count Sketches

Sketching vectors by the count sketch [Cormode and Hadjieleftheriou, 2008; Weinberger et al., 2009] is used for their dimensionality reduction which we use in this chapter.

Proposition 2. *Let d and d' denote the dimensionality of the input and sketched output vectors, respectively. Let vector $\boldsymbol{h} \in \mathcal{I}_{d'}^{d}$ contain d uniformly drawn integer numbers from $\{1, ..., d'\}$ and vector $\boldsymbol{s} \in \{-1, 1\}^d$ contain d uniformly drawn values from $\{-1, 1\}$. Then, the sketch projection matrix $\boldsymbol{P} \in \{-1, 0, 1\}^{d' \times d}$ becomes:*

$$P_{ij} = \begin{cases} s_i & \text{if } h_i = j, \\ 0 & \text{otherwise}, \end{cases} \tag{3.7}$$

and the sketch projection $p : \mathbb{R}^d \to \mathbb{R}^{d'}$ is a linear operation given as $p(\boldsymbol{\psi}) = \boldsymbol{P}\boldsymbol{\psi}$ (or $p(\boldsymbol{\psi}; \boldsymbol{P}) = \boldsymbol{P}\boldsymbol{\psi}$ to highlight $\boldsymbol{P}$).

Proof 1. *It directly follows from the definition of the count sketch* e.g., *see Definition 1 [Weinberger et al., 2009].*

Remark 4. *Count sketches are unbiased estimators: $\mathbb{E}_{\boldsymbol{h},\boldsymbol{s}}(p(\boldsymbol{\psi}, \boldsymbol{P}(\boldsymbol{h}, \boldsymbol{s})), p(\boldsymbol{\psi}', \boldsymbol{P}(\boldsymbol{h}, \boldsymbol{s}))) = \langle \boldsymbol{\psi}, \boldsymbol{\psi}' \rangle$. As variance $\mathbb{V}_{\boldsymbol{h},\boldsymbol{s}}(p(\boldsymbol{\psi}), p(\boldsymbol{\psi}')) \leq \frac{1}{d'} \left(\langle \boldsymbol{\psi}, \boldsymbol{\psi}' \rangle^2 + \|\boldsymbol{\psi}\|_2^2 \|\boldsymbol{\psi}'\|_2^2 \right)$, we note that larger sketches are less noisy. Thus, for every modality we compress, we use a separate sketch matrix $\boldsymbol{P}$. As video modalities are partially dependent, this implicitly leverages the unbiased estimator and reduces the variance.*

Proof 2. *For the first and second property, see Appendix A of paper [Weinberger et al., 2009] and Lemma 3 [Pham and Pagh, 2013].*

3.4 Approach

Our pipeline is illustrated in Figure 3.1. It consist of (i) the Fisher Vector and Bag-of-Words hallucinating streams denoted as FV and BoW (shown in dashed red), respectively, (ii) the High Abstraction Features stream denoted as HAF, and (iii) the Prediction Network abbreviated as PredNet.

The role of BoW/FV streams is to take I3D intermediate representations generated from the RGB and optical flow frames and learn to hallucinate BoW/FV representations. For this purpose, we use the MSE loss between the ground-truth BoW/FV and the outputs of BoW/FV streams. The role of the HAF stream is to further process I3D

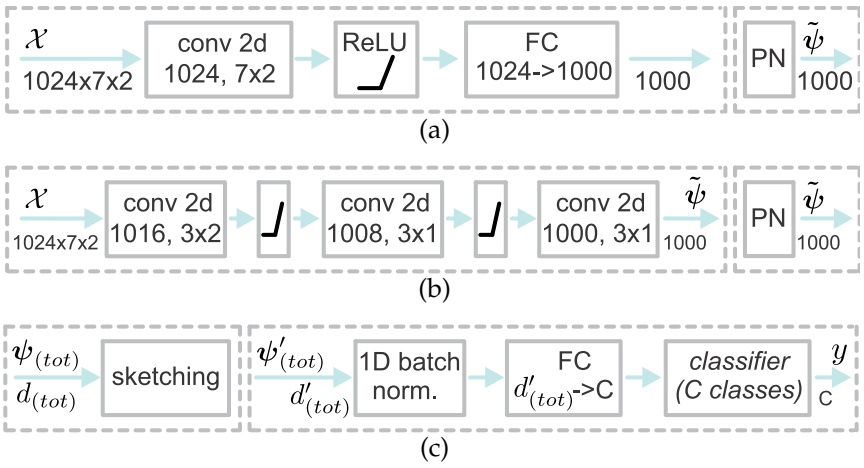


Figure 3.3: Stream types used in our network. Figures 3.3a and 3.3b show *Fully Connected* and *Convolutional* variants used for the practical realization of the FV, BoW, OFF and HAF streams. Figure 3.3c shows our PredNet. Note that we indicate the type of operation and its parameters in each block *e.g.*, *conv2d* and its number of filters/size, or *Power Normalization* (*PN*). Beneath arrows, we indicate the size of input, intermediate or output representation.

intermediate representations before they are concatenated with hallucinated BoW/FV. PredNet fuses the concatenated BoW/FV/HAF and learns class concepts. Figure 3.2 shows our pipeline for hallucinating the OFF representation (I3D optical flow). Below, we describe each module in detail.

3.4.1 BoW/FV Hallucinating Streams

BoW/FV take as input the I3D intermediate representations $\mathcal{X}_{(rgb)}$ and $\mathcal{X}_{(opt.)}$ of size 1024×7 which were obtained by stripping the classifier and the last 1D conv. layer of I3D pre-trained on Kinetics-400. The latter dimension of $\mathcal{X}_{(rgb)}$ and $\mathcal{X}_{(opt.)}$ can be thought of as the temporal size. We concatenate $\mathcal{X}_{(rgb)}$ and $\mathcal{X}_{(opt.)}$ along the third mode and obtain $\mathcal{X}$ which has dimensionality $1024\times7\times2$. As FV contains the first- and second-order statistics, we use a separate stream per each type of statistics, and a single stream for BoW. For the practical choice of BoW/FV pipelines, we use either a Fully Connected (FC) unit shown in Figure 3.3a or a Convolutional (Conv) pipeline in Figure 3.3b. Thus, we investigate the following hallucinating stream combinations: (i) BoW-FC and FV-FC, (ii) BoW-Conv and FV-FC, or (iii) BoW-Conv and FV-Conv. Where indicated, we also equip each stream with Power Normalization (PN). For specific PN realizations, we investigate AsinhE, SigmE, and AxMin variants from Remarks 1, 2 and 3. Below we detail how we obtained ground-truth BoW/FV.

Ground-truth BoW/FV. To train Fisher Vectors, we computed 256 dimensional GMM-based dictionaries on descriptors resulting from IDT [Wang and Schmid, 2013] according to steps described in Sections 3.2 and 3.3.1. We applied PCA to trajectories (30 dim.), HOG (96 dim.), HOF (108 dim.), MBHx (96 dim.) and MBHy (96 dim.), and

we obtained the final 213 dim. local descriptors. We applied encoding as in Eq. (3.2) and (3.3), the aggregation from Section 3.3.2 and Power Normalization from Section 3.3.3. Thus, our encoded first- and second-order FV representations, each of size $256 \times 213 = 54528$, had to be sketched to 1000 dimensions. To this end, we followed Section 3.3.4, prepared matrices $\boldsymbol{P}_{(fv1)}$ and $\boldsymbol{P}_{(fv2)}$ as in Proposition 2, and fixed both of them throughout experiments. The sketched first- and second-order representations $\boldsymbol{\psi}'_{(fv1)} = \boldsymbol{P}_{(fv1)}\boldsymbol{\psi}_{(fv1)}$ and $\boldsymbol{\psi}'_{(fv2)} = \boldsymbol{P}_{(fv2)}\boldsymbol{\psi}_{(fv2)}$ can be readily combined next with the MSE loss functions detailed in Section 3.4.5.

For BoW, we followed Section 3.3.1 and applied k-means to build a 1000 dim. dictionary from the same descriptors which were employed to pre-compute FV. Then, the descriptors were encoded according to Eq. (3.1), aggregated according to steps described in Section 3.3.2 and normalized by Power Normalization from Section 3.3.3. Where indicated, we used 4000 dim. dictionary and thus applied sketching on such BoW to limit its vector size to 1000 dim.

We note that we use ground-truth BoW/FV descriptors only at the training stage to train our hallucination streams.

3.4.2 High Abstraction Features

High Abstraction Features (HAF) take as input the I3D intermediate representations $\mathcal{X}_{(rgb)}$ and $\mathcal{X}_{(opt.)}$. Practical realizations of HAF pipelines are identical to those of BoW/FV/OFF. Thus, we have a choice of either FC or Conv units illustrated in Figures 3.3a and 3.3b. We simply refer to those variants as HAF-FC and HAF-Conv, respectively. Similar to BoW/FV/OFF streams, the HAF representation also uses Power Normalization and it is of size 1000.

3.4.3 Optical Flow Features

For pipeline in Figure 3.2, the I3D intermediate representation $\mathcal{X}_{(rgb)}$ only is fed to hallucination/HAF streams. I3D Optical Flow Features $\mathcal{X}_{(opt.)}$ are pre-computed as the training ground-truth for the OFF layer (the MSE loss is used).

3.4.4 Combining Hallucinated BoW/FV/OFF and HAF

Figure 3.1 indicates that FV (first- and second-order), BoW and HAF feature vectors $\tilde{\boldsymbol{\psi}}_{(fv1)}$, $\tilde{\boldsymbol{\psi}}_{(fv2)}$, $\tilde{\boldsymbol{\psi}}_{(bow)}$ and $\boldsymbol{\psi}_{(haf)}$ are concatenated (via operator $\oplus$) to obtain $\boldsymbol{\psi}_{(tot)}$ and subsequently sketched (if indicated so during experiments), that is, $\boldsymbol{\psi}'_{(tot)} = \boldsymbol{P}_{(tot)}\boldsymbol{\psi}_{(tot)}$ which reduces the size of the total representation from $d = 4000$ to $500 \leq d' \leq 2000$. Matrix $\boldsymbol{P}_{(tot)}$ is prepared according to Proposition 2 and fixed throughout experiments. As sketching is a linear projection, we can backpropagate through it with ease. When also hallucinating OFF as in Figure 3.2, we additionally concatenate $\boldsymbol{\psi}_{(off)}$ with other feature vectors to obtain $\boldsymbol{\psi}_{(tot)}$.

PredNet. The final unit of our overall pipeline, PredNet, is illustrated in Figure 3.3c. On input, we take $\boldsymbol{\psi}_{(tot)}$ (no sketching) or $(\boldsymbol{\psi}'_{(tot)})$ if sketching is used, pass it via the

batch normalization and then an FC layer which produces a C dim. representation passed to the cross-entropy loss.

3.4.5 Objective and its Optimization

During training, we combine MSE loss functions responsible for training hallucination streams with the class. loss:

$$\ell^*(\boldsymbol{\mathcal{X}}, \mathbf{y}; \bar{\boldsymbol{\Theta}}) = \frac{\alpha}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} \|\tilde{\boldsymbol{\psi}}_i - \boldsymbol{\psi}'_i\|_2^2 + \ell\Big(f(\boldsymbol{\psi}'_{(tot)}; \boldsymbol{\Theta}_{(pr)}), \mathbf{y}; \boldsymbol{\Theta}_{(\ell)}\Big),$$
$$\text{where: } \forall i \in \mathcal{H}, \tilde{\boldsymbol{\psi}}_i = g(\hbar(\boldsymbol{\mathcal{X}}, \boldsymbol{\Theta}_i), \eta), \boldsymbol{\psi}'_i = \boldsymbol{P}_i \boldsymbol{\psi}_i,$$
$$\boldsymbol{\psi}_{(haf)} = g\Big(\hbar(\boldsymbol{\mathcal{X}}, \boldsymbol{\Theta}_{(haf)}), \eta\Big),$$
$$\boldsymbol{\psi}'_{(tot)} = \boldsymbol{P}_{(tot)} \Big[\oplus_{i \in \mathcal{H}} \tilde{\boldsymbol{\psi}}_i; \boldsymbol{\psi}_{(haf)}\Big]. \tag{3.8}$$

The above equation is a trade-off between the MSE loss functions $\{\|\tilde{\boldsymbol{\psi}}_i - \boldsymbol{\psi}'_i\|_2^2, i \in \mathcal{H}\}$ and the classification loss $\ell(\cdot, \mathbf{y}; \boldsymbol{\Theta}_{(\ell)})$ with some label $\mathbf{y} \in \mathcal{Y}$ and parameters $\boldsymbol{\Theta}_{(\ell)} \equiv \{\boldsymbol{W}, \mathbf{b}\}$. The trade-off is controlled by a constant $\alpha \geq 0$ while MSE is computed over hallucination streams $i \in \mathcal{H}$, and $\mathcal{H} \equiv \{(fv1), (fv2), (bow), (off)\}$ is our set of hallucination streams which can be modified to multiple/few such streams depending on the task at hand. Moreover, $g(\cdot, \eta)$ is a Power Normalizing function chosen from the family described in Section 3.3.3, $f(\cdot; \boldsymbol{\Theta}_{(pr)})$ is the PredNet module with parameters $\boldsymbol{\Theta}_{(pr)}$ which we learn, $\{\hbar(\cdot, \boldsymbol{\Theta}_i), i \in \mathcal{H}\}$ are the hallucination streams while $\{\tilde{\boldsymbol{\psi}}_i, i \in \mathcal{H}\}$ are the corresponding hallucinated BoW/FV/OFF representations. Moreover, $\hbar(\cdot, \boldsymbol{\Theta}_{(haf)})$ is the HAF stream with the output denoted by $\boldsymbol{\psi}_{(haf)}$. For the hallucination streams, we learn parameters $\{\boldsymbol{\Theta}_i, i \in \mathcal{H}\}$ while for HAF, we learn $\boldsymbol{\Theta}_{(haf)}$. The full set of parameters we learn is defined as $\bar{\boldsymbol{\Theta}} \equiv (\{\boldsymbol{\Theta}_i, i \in \mathcal{H}\}, \boldsymbol{\Theta}_{(haf)}, \boldsymbol{\Theta}_{(pr)}, \boldsymbol{\Theta}_{(\ell)})$. Furthermore, $\{\boldsymbol{P}_i, i \in \mathcal{H}\}$ are the projection matrices for count sketching of the ground-truth BoW/FV/OFF feature vectors $\{\boldsymbol{\psi}_i, i \in \mathcal{H}\}$ while $\{\boldsymbol{\psi}'_i, i \in \mathcal{H}\}$ are the corresponding sketched/compressed representations. Finally, $\boldsymbol{P}_{(tot)}$ is the projection matrix for hallucinated BoW/FV/OFF representations concatenated with each other and HAF, that is, for $\boldsymbol{\psi}_{(tot)} = \Big[\oplus_{i \in \mathcal{H}} \tilde{\boldsymbol{\psi}}_i; \boldsymbol{\psi}_{(haf)}\Big]$ which results in the sketched counterpart $\boldsymbol{\psi}'_{(tot)}$ that goes into the PredNet module f. Section 3.3.4 details how to select matrices $\boldsymbol{P}$. If sketching is not needed, we simply set a given $\boldsymbol{P}$ to be the identity projection $\boldsymbol{P} = \mathbb{I}$. In our experiments, we simply set $\alpha = 1$.

Optimization. We minimize $\ell^*(\boldsymbol{\mathcal{X}}, \mathbf{y}; \bar{\boldsymbol{\Theta}})$ w.r.t. parameters of each stream, that is $\{\boldsymbol{\Theta}_i, i \in \mathcal{H}\}$ for hallucination streams, $\boldsymbol{\Theta}_{(haf)}$ for the HAF stream, $\boldsymbol{\Theta}_{(pr)}$ for PredNet and $\boldsymbol{\Theta}_{(\ell)}$ for the classification loss. In practice, we perform a simple alternation over two minimization steps shown in Figure 3.4. In each iteration, we perform one forward and backward pass regarding the MSE losses to update the parameters $\{\boldsymbol{\Theta}_i, i \in \mathcal{H}\}$ of the hallucination streams. Then, we perform one forward and backward pass regarding the classification loss ℓ. We update all network streams during this pass. Thus, one can think of our network as multi-task learning with BoW/FV/OFF and label learning tasks. Furthermore, we use the Adam minimizer with 10^{-4} initial

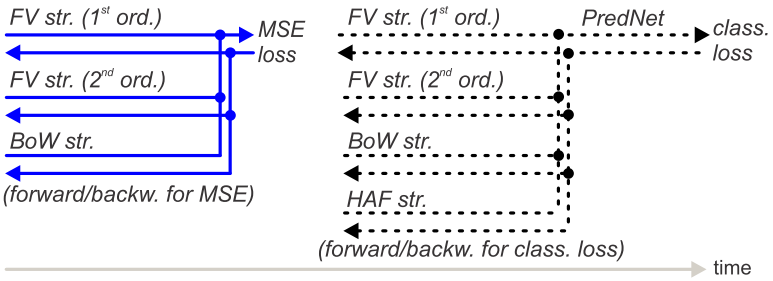


Figure 3.4: Optimization. In each step, we have (i) forward/backward passes via BoW/FV (optionally OFF) streams for the MSE loss followed by (ii) forward/backward passes via BoW/FV (opt. OFF), and HAF streams and PredNet for the classification loss.

learning rate which we halve every 10 epochs. We run our training for 50–100 epochs depending on the dataset.

Sketching the Power Normalized vectors.

Proposition 3. *Sketching PN vectors increases the sketching variance (ℓ_2-normalized by vec. norms) by $1 \leq \kappa \leq 2$.*

Proof 3. *Normalize variance $\mathbb{V}$ from Remark 4 by the norms $\|\boldsymbol{\psi}\|_2^2 \|\boldsymbol{\psi}'\|_2^2$. Consider $\mathbb{V}^{(\gamma)}$ which is the variance for d dimensional vectors $\{(\boldsymbol{\psi}^\gamma, \boldsymbol{\psi}'^\gamma) : \boldsymbol{\psi} \geq 0, \boldsymbol{\psi}' \geq 0\}$ power normalized by Gamma from Remark 1, and divide it accordingly by $\|\boldsymbol{\psi}^\gamma\|_2^2 \|\boldsymbol{\psi}'^\gamma\|_2^2$. For extreme PN ($\gamma \to 0$), we have:*

$$\lim_{\gamma \to 0} \mathbb{V}^{(\gamma)} = \frac{1}{d'} \lim_{\gamma \to 0} \left(\frac{\langle \boldsymbol{\psi}^\gamma, \boldsymbol{\psi}'^\gamma \rangle^2}{\|\boldsymbol{\psi}^\gamma\|_2^2 \|\boldsymbol{\psi}'^\gamma\|_2^2} + 1 \right) = \frac{2}{d'}. \tag{3.9}$$

Now, assume that d dimensional $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ are actually ℓ_2-norm normalized. Then, we have the following ratio of variances:

$$\kappa = \mathbb{V} / \mathbb{V}^{(\gamma)} = 2 / (\langle \boldsymbol{\psi}, \boldsymbol{\psi}' \rangle^2 + 1), \tag{3.10}$$

Thus, $1 \leq \kappa \leq 2$ depends on $(\boldsymbol{\psi}, \boldsymbol{\psi}')$, and κ varies smoothly between $[1; 2]$ for $1 \leq \gamma \leq 0$ of Gamma, a monotonically increasing function. For typical $\gamma = 0.5$, we measured for the actual data that $\kappa \approx 1.3$.

3.5 Experiments

3.5.1 Datasets and Evaluation Protocols

HMDB-51 [Kuehne et al., 2011] consists of 6766 internet videos over 51 classes; each video has ∼20–1000 frames. Following the protocol, we report the mean accuracy across three splits.

YUP++ [Feichtenhofer et al., 2017b] dataset contains so-called video textures. It has 20 scene classes, 60 videos per class, and its splits contain scenes captured with the static or moving camera. We follow the standard splits (1/9 dataset for training).

	sp1	*sp2*	*sp3*	mean acc.
HAF only	81.83%	80.78%	80.45%	81.02%
HAF+BoW/FV exact	83.00%	82.80%	81.70%	**82.50**%
HAF+BoW halluc.	82.29%	81.24%	80.98%	81.50%
HAF+FV halluc.	82.68%	81.05%	79.93%	81.22%
HAF+BoW/FV halluc.	82.88%	82.74%	81.50%	**82.37**%

Table 3.1: Evaluations of pipelines on the HMDB-51 dataset. We compare (*HAF only*) and (*HAF+BoW/FV exact*) which show the lower- and upper bound on the accuracy, and our (*HAF+BoW/FV halluc.*), (*HAF+BoW halluc.*) and (*HAF+FV halluc.*).

	static	*dynamic*	*mixed*	mean acc.
HAF only	92.03%	81.67%	89.07%	87.59%
HAF+BoW/FV exact	93.30%	89.82%	92.41%	91.84%
HAF+BoW halluc.	92.69%	85.93%	92.41%	90.34%
HAF+FV halluc.	92.69%	88.15%	91.48%	90.77%
HAF+BoW/FV halluc.	93.15%	89.63%	92.31%	**91.69**%

Table 3.2: Eval. of pipelines on YUP++. See Table 3.1 for the legend.

MPII Cooking Activities [Rohrbach et al., 2012] consist of high-resolution videos of people cooking various dishes. The 64 distinct activities from 3748 clips include coarse actions *e.g.*, *opening refrigerator*, and fine-grained actions *e.g.*, *peel*, *slice*, *cut apart*. We use the mean Average Precision (mAP) over 7-fold cross validation. For human-centric protocol [Cherian et al., 2017a, 2018], we use Faster-RCNN [Ren et al., 2015] to crop video around humans.

Charades [Sigurdsson et al., 2016] consist of of 9848 videos of daily indoors activities, 66500 temporal annotations and 157 classes.

3.5.2 Data Pre-processing

For HMDB-51 and YUP++, we use the data augmentation strategy described in the original authors' papers (*e.g.*, random crop of videos, left-right flips on RGB and optical flow frames. For testing, center crop, no flipping are used.

For the MPII dataset with human-centric pre-processing, human detector is used first. Then, we crop randomly around the bounding box of human subject (we include it). Finally, we allow scaling, zooming in, and left-right flips. For longer videos, we sample sequences to form a 64-frame sequence. For short videos (less than 64 frames), we loop the sequence many times to fit its length to the expected input length. Lastly, we scale the pixel values of RGB and optical flow frames to the range between -1 and 1.

3.5.3 Evaluations

We start our experiments by investigating various aspects of our pipeline and then we present our final results.

HAF, BoW and FV streams. Firstly, we ascertain the gains from our HAF and

	sp1	*sp2*	*sp3*	mean acc.
HAF-Conv+BoW/FV-FC halluc.	81.96%	80.39%	80.52%	80.95%
HAF-FC+BoW/FV-Conv halluc.	82.42%	81.30%	81.50%	81.74%
HAF-FC+BoW/FV-FC halluc.	82.88%	82.74%	81.50%	**82.37**%

Table 3.3: Evaluations of pipelines on the HMDB-51 dataset. We compare (*HAF+BoW/FV halluc.*) approach on different architectures used for HAF and BoW/FV streams such as (*FC*) and (*Conv*).

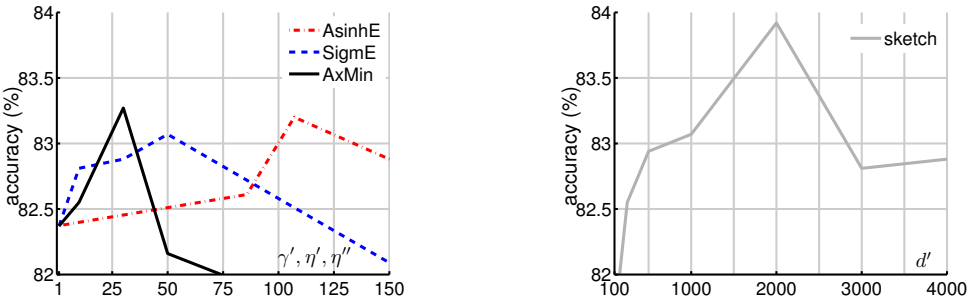
BoW/FV streams. We evaluate the performance of (i) the HAF-only baseline pipeline without IDT-based BoW/FV information (*HAF only*), (ii) the HAF-only baseline with exact ground-truth IDT-based BoW/FV added at both training and testing time (*HAF+BoW/FV exact*), and (iii) the combined HAF plus IDT-based BoW/FV streams (*HAF+BoW/FV halluc.*). We also perform evaluations on (iv) HAF plus IDT-based BoW stream (*HAF+BoW halluc.*) and HAF plus IDT-based FV stream (*HAF+FV halluc.*) to examine how much gain IDT-based BoW and FV bring, respectively. As Section 3.4.1 suggests that each stream can be based on either the Fully Connected (FC) or Convolutional (Conv.) pipeline, we firstly investigate the use of FC unit from Figure 3.3a, that is, we use HAF-FC, BoW-FC and HAF-FC streams. PredNet also uses FC. For ground-truth FV, we use 1000 dim. sketches.

Table 3.1 presents results on the HMDB-51 dataset. As expected, the (*HAF only*) is the poorest performer while (*HAF+BoW/FV exact*) is the best performer determining the upper limit on the accuracy. Hallucinating (*HAF+BoW halluc.*) outperforms (*HAF+FV halluc.*) marginally. We expect FV to perform close to BoW due to the significant compression with sketching by factor $\sim$52.5$\times$. Approaches (*HAF+FV/BoW halluc.*) and (*HAF+BoW/FV exact*) achieve the best results, and outperform (*HAF only*) by 1.35% and 1.48% accuracy. These result show that our hallucination strategy (*HAF+FV/BoW halluc.*) can mimic (*HAF+BoW/FV exact*) closely. Our 82.37% accuracy is the new state of the art. Below we show larger gains on YUP++.

Table 3.2 presents similar findings on the YUP++ dataset. Our (*HAF+FV halluc.*) brings the improvement of $\sim$2.2 and $\sim$6.5% over (*HAF+BoW halluc.*) and (*HAF only*) on scenes captured with the moving camera (*dynamic*). Our (*HAF+BoW/FV halluc.*) yields $\sim$8.0% over (*HAF only*) thus demonstrating again the benefit of hallucinating BoW/FV descriptors. The total gain for (*HAF+BoW/FV halluc.*) over (*HAF only*) equals 4.1%. Our (*HAF+FV/BoW halluc.*) matches results of (*HAF+BoW/FV exact*) without explicitly computing BoW/FV during testing. Below, we investigate different architectures of our streams.

Fully Connected/Convolutional streams. Figures 3.3a and 3.3b show two possible realizations of HAF, BoW and FV streams. While FC and Conv. architectures are not the only possibilities, they are the simplest ones. Table 3.3 shows that using FC layers (*FC*) for HAF and BoW/FV streams, denoted as (*HAF-FC+BoW/FV-FC halluc.*) outperforms Convolutional (*Conv*) variants by up to $\sim$1.5% accuracy. Thus, we use only the FC architecture in what follows.

Sketching and Power Normalization. As PredNet uses FC layers (see Figure 3.3c), we



(a) (b)

Figure 3.5: Evaluations of (fig. 3.5a) Power Normalization and (fig. 3.5b) sketching on the HMDB-51 dataset (split 1 only).

	sp1	*sp2*	*sp3*	mean acc.
HAF only	81.83%	80.78%	80.45%	81.02%
HAF+BoW/FV halluc.	83.46%	82.61%	81.37%	**82.48**%

ADL+ResNet+IDT 74.3% [Wang and Cherian, 2018] STM Network+IDT 72.2% [Feichtenhofer et al., 2017a]
ADL+I3D 81.5% [Wang and Cherian, 2018] Full-FT I3D 81.3% [Carreira and Zisserman, 2017]

Table 3.4: Evaluations of (*top*) our (*HAF+BoW/FV halluc.*) and (*bottom*) comparisons to the state of the art on HMDB-51.

expect that limiting the input size to this layer via count sketching from Section 3.3.4 should benefit the performance. Moreover, as visual and video representations suffer from so-called burstiness, we investigate AsinhE, SigmE and AxMin from Remarks 1, 2 and 3.

Figure 3.5a investigates the classification accuracy on the HMDB-51 dataset (split 1) when our HAF and BoW/FV feature vectors $\{\tilde{\boldsymbol{\psi}}_i, i \in \mathcal{H}\}$ and $\boldsymbol{\psi}_{(haf)}$ (described in Sections 3.4.4 and 3.4.5) are passed via Power Normalizing functions AsinhE, SigmE and AxMin prior to concatenation (see Figure 3.1). From our experiment it appears that all PN functions perform similarly and improve results from the baseline 82.29% to $\sim$83.20% accuracy. We observe a similar gain from 93.15% to 94.44% acc. on YUP++ (*static*). In what follows, we simply use AsinhE for PN.

Figure 3.5b illustrates on the HMDB-51 dataset (split 1) that applying count sketching on concatenated HAF and BoW/FV feature vectors $\boldsymbol{\psi}_{(tot)}$, which produces $\boldsymbol{\psi}'_{(tot)}$ (see Section 3.4.5 for reference to symbols), improves results from 82.88% to 83.92% accuracy for $d' = 2000$. This is expected as reduced size of $\boldsymbol{\psi}'_{(tot)}$ results in fewer parameters of the FC layer of PredNet and less overfitting. Similarly, for the YUP++ dataset and the split (*static*), we see the performance increase from 93.15% to 94.81% accuracy.

Comparisons with other methods. Below we present our final results and we contrast them against the state of the art. Table 3.4 shows results on the HMDB-51 dataset. For our method, we used sketching of $\boldsymbol{\psi}_{(tot)}$ with $d' = 2000$ and PN. Our (*HAF+BoW/FV halluc.*) model yields 82.48% acc. which beats results in the literature to

	static	*dynamic*	*mixed*	mean stat/dyn	mean all
HAF only	92.03%	81.67%	89.07%	86.8%	87.6%
HAF+BoW/FV halluc.	94.81%	**89.63%**	**93.33%**	**92.2%**	**92.6%**
T-ResNet [Feichtenhofer et al., 2017b]	92.41%	81.50%	89.00%	87.0%	87.6%
ADL I3D [Wang and Cherian, 2018]	**95.10%**	88.30%	-	91.7%	-

Table 3.5: Evaluations of (*top*) our (*HAF+BoW/FV halluc.*) and (*bottom*) comparisons to the state of the art on YUP++.

	sp1	*sp2*	*sp3*	*sp4*	*sp5*	*sp6*	*sp7*	mAP
HAF+BoW halluc.	73.9	71.6	76.2	70.7	76.3	71.9	63.4	71.9%
HAF+BoW halluc.+SK/PN	73.9	75.8	72.2	73.9	77.0	73.6	68.8	**73.6%**
HAF* only	74.6	73.2	77.0	75.1	76.1	75.6	71.9	74.8%
HAF*+BoW halluc.	78.8	75.0	84.1	76.0	77.0	78.3	75.2	**77.8%**
HAF*+BoW hal.+MSK/PN	80.1	79.2	84.8	83.9	80.9	78.5	75.5	**80.4%**
HAF•+BoW hal.+MSK/PN	80.8	80.9	85.0	83.9	82.0	79.8	79.6	**81.7%**
ditto+OFF hal.	81.2	81.2	84.9	83.4	84.2	78.9	79.1	**81.8%**
I3D+BoW MTL•	79.1	78.1	83.6	78.7	79.1	78.6	76.5	79.1%

KRP-FS 70.0% [Cherian et al., 2018] KRP-FS+IDT 76.1% [Cherian et al., 2018]
GRP 68.4% [Cherian et al., 2017a] GRP+IDT 75.5% [Cherian et al., 2017a]

Table 3.6: Evaluations of (*top*) our (*HAF+BoW halluc.*) pipeline without sketching/PN, with sketching/PN (*SK/PN*). The (*HAF* only*) is our baseline without the BoW stream, (*) denotes human-centric pre-processing while (*MSK/PN*)in pipeline (*HAF*+BoW hal.+MSK/PN*) denotes multiple sketches per BoW followed by Power Norm (*PN*). (*bottom*) Other methods on the MPII dataset.

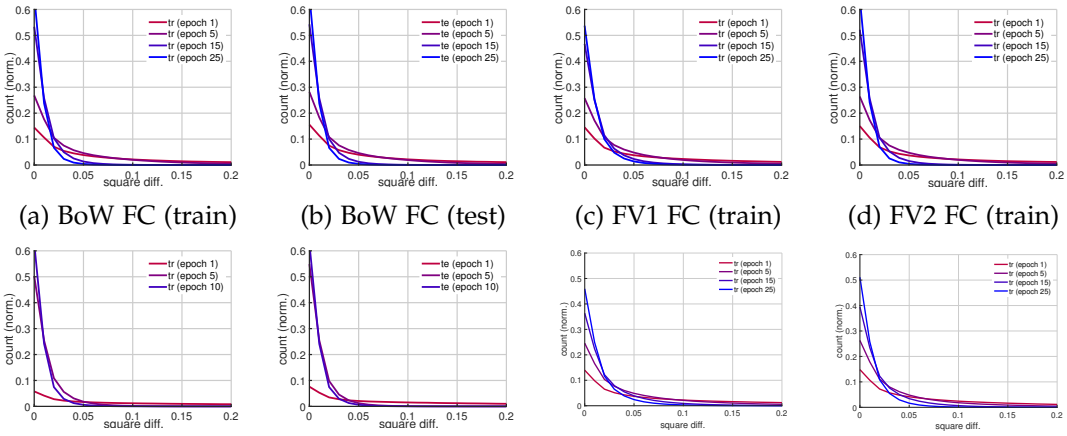
HAF only	HAF+BoW/ FV exact	HAF+BoW/FV/OFF halluc. +MSK×2/PN	HAF+BoW/FV/OFF halluc. +MSK×4/PN	HAF+BoW/FV/OFF halluc. +MSK×8/PN
37.2	41.9	42.0	42.2	**43.1**

Table 3.7: Evaluations of our methods on the Charades dataset.

the best of our knowledge. If we tune PN per split, our results reach 82.78% accuracy. However, we do not advise such tuning due to danger of overfitting. We note that we outperform more complex methods such as Adversarial Discriminative Learning (ADL) with I3D [Wang and Cherian, 2018] and Fully Fine-Tuned I3D [Carreira and Zisserman, 2017].

Table 3.5 shows results on the YUP++ dataset. Our (*HAF+BoW/FV halluc.*) model yields very competitive results on the static protocol and outperforms competitors on the dynamic and mixed protocols. With 92.2% mean accuracy over static and dynamic scores (*mean stat/dyn*), we outperform more complex ADL+I3D [Wang and Cherian, 2018] and T-ResNet [Feichtenhofer et al., 2017b].

Table 3.6 shows results for the MPII dataset for which we use HAF with/without the BoW (4000 dim.) hallucination stream (no FV stream). As MPII contains subsequences, we use integral pooling from Prop. 1. Our basic model (*HAF+BoW halluc.*) scores ∼71.9% mAP. Applying sketching and PN (*HAF+BoW halluc.+SK/PN*) yields 73.6% mAP. Unlike GRP+IDT [Cherian et al., 2017a] and KRP-FS+IDT [Cherian



(a) BoW FC (train) (b) BoW FC (test) (c) FV1 FC (train) (d) FV2 FC (train)

(e) BoW Conv (train) (f) BoW Conv (test) (g) FV1 FC (train), -SK/PN (h) FV2 FC (train), -SK/PN

Figure 3.6: Evaluation of the square difference between the hallucinated and ground truth representations on HMDB-51 (split 1). Experiments in the top row use (*FC*) streams with sketching and PN. Two leftmost plots in the bottom row use (*Conv*) streams. Two rightmost plots in the bottom row use (*FC*) streams without sketching/PN (*-SK/PN*).

et al., 2018], our first two experiments do not use any human- or motion-centric pre-processing. With human-centric crops, denoted with (*), our baseline without BoW (*HAF* only*) achieves 74.8% mAP. The model with BoW (*HAF+BoW halluc.*) scores 77.8% mAP. By utilizing 4 sketches for BoW and 4 BoW streams with Power Normalization (*HAF*+BoW hal.+MSK/PN*), we obtain 80.4% mAP.

Hallucinating Optical Flow. For (*HAF$^\bullet$+BoW hal.+MSK/PN*) in Table 3.6, we increased the resolution of RGB frames 2× to obtain larger human-centric crops and 2× larger optical flow res., which yielded 81.7% mAP. In the same setting, hallucinating optical flow feat. (*ditto+OFF hal.*) yielded 81.84% mAP, the new state of the art.

Charades. In Table 3.7, baselines (*HAF only*) and (*HAF+BoW/FV exact*) score 37.2% and 41.9% mAP. Moreover, our best pipeline (*HAF+BoW/FV/OFF halluc.+MSK×8/PN*) that hallucinates IDT BoW/FV and I3D optical flow features (OFF) with 8 sketches per BoW/FV/OFF and PN yielded 43.1% (a much more complex feature banks [Wu et al., 2019a] yield 43.4%). Finally, if 25% of this dataset was dedicated to testing, ∼55h of computations would be saved.

Hallucination quality. Below, we provide an analysis of the quality of hallucination of the BoW/FV streams compared to the ground-truth BoW/FV feature vectors. Figure 3.6 presents histograms of the square difference between the hallucinated features and ground-truth ones. Specifically, we plot histograms of $\{(\tilde{\psi}_{(bow),mn} - \psi_{(bow),mn})^2, m \in \mathcal{I}_{1000}, n \in \mathcal{N}\}$, where index m runs over features $m \in \mathcal{I}_{1000}$ and $n \in \mathcal{N}$ runs over each video. Counts for training and testing splits are normalized by 1000 (the number of features) and the number of training and testing videos, respectively. The histograms are computed over bins of size 0.01 thus allowing us to simply plot continuously looking lines instead of bins. Figure 3.6a shows that the BoW ground-truth descriptors

	sp1	*sp2*	*sp3*	*sp4*	*sp5*	*sp6*	*sp7*	mAP
HAF*+BoW halluc.	78.8	75.0	84.1	76.0	77.0	78.3	75.2	77.8%
HAF*+BoW hal.+MSK/PN	80.1	79.2	84.8	83.9	80.9	78.5	75.5	80.4%
HAF•+BoW halluc.	78.8	78.3	84.2	77.4	77.1	78.3	75.2	78.5%
HAF•+BoW hal.+MSK/PN	80.8	80.9	85.0	83.9	82.0	79.8	79.6	**81.7**%

Table 3.8: Evaluations on MPII. The (*HAF*+BoW halluc.*) is our pipeline with the BoW stream, (*) denotes human-centric pre-processing for 256 pixels (height) while (*HAF*+BoW hal.+MSK/PN*) denotes our pipeline with multiple sketches per BoW followed by Power Norm (*PN*). By analogy, (•) denotes human-centric pre-processing for 512 pixels (height).

for the training split are learnt closely by our BoW hallucinating unit based on FC layers (*FC*). We capture histograms for epochs $1, 5, 15, 25$ in colors interpolated from red to blue. As one can see, in early epochs, the peak around the first bin is small. As the epochs progress, the peak around the first bin becomes prominent while further bins decrease in size. This indicates that as the training epochs progress, the approximation error becomes smaller and smaller. Figure 3.6b shows that the BoW ground-truth descriptors for the testing split are also approximated closely by the hallucinated BoW descriptors.

We compared histograms for testing and training slits for BoW, first- and second-order FV and observed small differences only. Such a comparison can be conducted by computing the ratio of testing to training bins and it reveals variations between $0.8\times$ and $1.25\times$. Thus, without the loss of clarity, we skip showing plots for FV testing splits. Figures 3.6c and 3.6d show that the first- and second-order FV terms (*FV1*) and (*FV2*) can be also learnt closely by our hallucinating units. We show only the quality of approximation on the training split as behavior on testing splits matches closely the behavior on training splits. Figures 3.6e, 3.6f, 3.6g and 3.6h show the similar learning/approximation trend for BoW training and testing splits, and the first- and second-order FV terms (training only) given our hallucinating unit based on FC layers (*FC*) with no sketching or PN (*-SK/PN*).

Higher resolution frames on MPII. For human-centric pre-processing on MPII denoted by (*) in the main submission, we observed that the bounding boxes used for extraction of the human subject are of low resolution. Thus, we decided to firstly resize RGB frames to 512 pixels (height) rather than 256 pixels (as in our main submission) and then compute the corresponding optical flow, and perform extraction of human subjects for which the resolution thus increased $2\times$. The (*HAF*+BoW halluc.*), our pipeline with the BoW stream, and (*HAF*+BoW hal.+MSK/PN*) with multiple sketches and PN are computed for the standard 256 pixels (height) denoted by (*) are given in Table 3.8. Note that results for (*) are taken from our main submission. The (*HAF•+BoW halluc.*), our pipeline with the BoW stream, and (*HAF•+BoW hal.+MSK/PN*) pipeline are analogous pipelines but computed for the increased 512 pixel resolution (height) denoted by (•). According to the table, increasing the resolution $2\times$ prior to human detection, extracting subjects in higher resolution and scaling (to the 256 size for shorter side) yields 1.3% improvement in accuracy.

Discussion. There exist several reasons explaining why our pipeline works well *e.g.*, sophisticated IDT trajectory modeling is unlikely to be captured by off-the-shelf CNNs unless a CNN is enforced to learn IDT. We perform 'translation' of the I3D output into IDT-based BoW/FV descriptors thus enforcing I3D to implicitly learn IDT which co-regularizes I3D which resembles Domain Adaptation (DA) methods: a source network co-regularizes a target network [Koniusz et al., 2017; Tas and Koniusz, 2018; Koniusz et al., 2018a; Herath et al., 2019; Harandi et al., 2018; Kumar Roy et al., 2019] by the alignment of feature statistic of both streams. Related to DA is Multi-task Learning (MTL) known for boosting generalization/preventing overfitting of CNNs due to task specific losses [Caruana, 1997]. MTL training on related tasks is known to boost individual task accuracies beyond a vanilla feature fusion [Thrun, 1996]. Finally, our pipeline uses self-supervision *e.g.*, IDT BoW/FV and OFF descriptors represent easy to obtain self-information about videos. We train our halluc./last I3D layers via task-specific losses (similar to MTL). However, our halluc. layers distill the domain specific cues which are fed back into the network (PredNet) which boosts our results by further ∼2.7% compared to vanilla (*I3D+BoW MTL*$^{\bullet}$) in Table 3.6.

3.6 Conclusions

We have proposed a simple yet powerful strategy that learns IDT-based descriptors (and even optical flow features) and hallucinates them in a CNN pipeline for AR at the test time. With state-of-the-art results, we hope our method will spark a renewed interest in IDT-like descriptors.

Chapter 4

Statistical Moment and Subspace Descriptors

In this chapter, we build on a concept of self-supervision by taking RGB frames as input to learn to predict both action concepts and auxiliary descriptors *e.g.*, object descriptors. So-called hallucination streams are trained to predict auxiliary cues, simultaneously fed into classification layers, and then hallucinated at the testing stage to aid network. We design and hallucinate two descriptors, one leveraging four popular object detectors applied to training videos, and the other leveraging image- and video-level saliency detectors. The first descriptor encodes the detector- and ImageNet-wise class prediction scores, confidence scores, and spatial locations of bounding boxes and frame indexes to capture the spatio-temporal distribution of features per video. Another descriptor encodes spatio-angular gradient distributions of saliency maps and intensity patterns. Inspired by the characteristic function of the probability distribution, we capture four statistical moments on the above intermediate descriptors. As numbers of coefficients in the mean, covariance, coskewness and cokurtotsis grow linearly, quadratically, cubically and quartically w.r.t. the dimension of feature vectors, we describe the covariance matrix by its leading n' eigenvectors (so-called subspace) and we capture skewness/kurtosis rather than costly coskewness/cokurtosis. We obtain state of the art on five popular datasets such as Charades and EPIC-Kitchens.

4.1 Introduction

Action Recognition (AR) has progressed from hand-crafted video representations [Dalal et al., 2006; Scovanner et al., 2007; Kläser et al., 2008; Wang et al., 2011, 2013b; Wang and Schmid, 2013; Wang, 2017; Wang et al., 2019b,c] to Convolutional Neural Networks (CNN) [Simonyan and Zisserman, 2014; Tran et al., 2015; Feichtenhofer et al., 2016a; Carreira and Zisserman, 2017]. The two-stream networks [Simonyan and Zisserman, 2014], 3D spatio-temporal features [Tran et al., 2015], spatio-temporal ResNet model [Feichtenhofer et al., 2016a] and the new Inflated 3D (I3D) convolutions network pre-trained on Kinetics-400 [Carreira and Zisserman, 2017]. Often, AR combine the RGB and optical flow inputs, and benefit from a late fusion (next to the

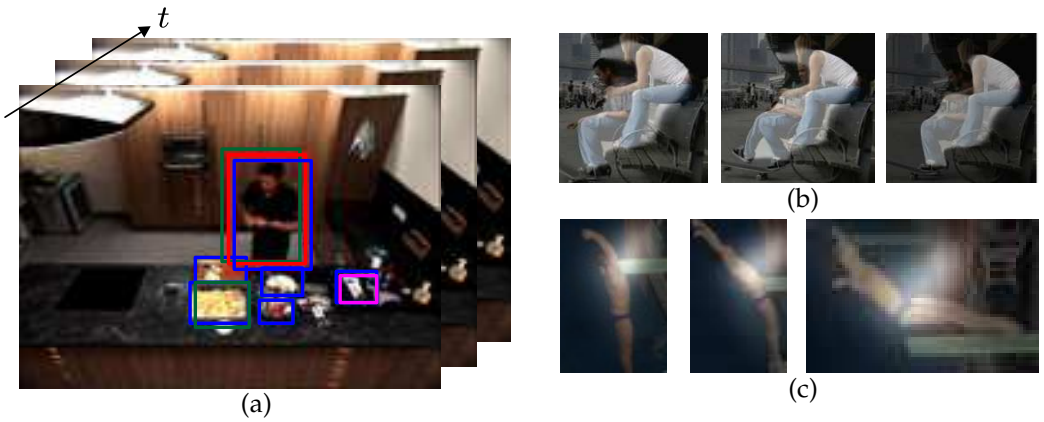


Figure 4.1: We use detectors and saliency in hallucination descriptors. Figure 4.1a shows bounding boxes from four detectors. The faster R-CNN detector with ResNet101 focuses on human-centric actions such as *stand*, *watch*, *talk*, *etc*. The other three detectors discover objects *e.g.*, *oven*, *sink*, *clock*, *etc*. Figure 4.1b shows that the MNL saliency detector focuses on spatial regions. Figure 4.1c shows ACLNet saliency detector discovers motion regions.

classifier) with low-level representations such as Improved Dense Trajectory (IDT) descriptors [Wang and Schmid, 2013] due to their highly complementary nature [Fernando and Gould, 2016; Cherian et al., 2017b, 2018; Wang and Cherian, 2018; Choutas et al., 2018]. Recently, AssembleNet and AssembleNet++ [Ryoo et al., 2020b], learnt with the Neural Architecture Search (NAS) have yielded superb results.

A recent AR pipeline [Wang et al., 2019d], called DEEP-HAL, used IDT descriptors encoded with Bag-of-Words (BoW) [Sivic and Zisserman, 2003; Csurka et al., 2004] and Fisher Vectors (FV) [Perronnin and Dance, 2007; Perronnin et al., 2010] to learn them by so-called hallucination streams and generate at the testing stage to boost results beyond a naive fusion of modalities. DEEP-HAL and approach [Tang et al., 2019] have shown that even optical flow frames encoded by a network can be learnt by another network trained on RGB frames only, thus pointing at redundancy in training both RGB and optical flow network streams. DEEP-HAL [Wang et al., 2019d] has attained the state of the art on several AR benchmarks by learning to hallucinate IDT-based BoW/FV and Optical Flow Features (OFF) from a single RGB-based I3D network stream.

DEEP-HAL opens up an exciting opportunity to investigate what other representations can co-regularize/self-supervise a backbone network for AR with the goal of learning to hallucinate costly representations at the training stage and simply leveraging outputs of halluciantion streams at the testing time. We build on DEEP-HAL which already includes IDT-based BoW/FV and OFF streams. However, we investigate the self-supervisory ability of object/saliency detectors in DEEP-HAL. Moreover, beyond I3D backbone, we investigate the use of AssembleNet and AssembleNet++ but we disable their (impractical to obtain) segmentation mask input.

In this chapter, we design and hallucinate two kinds of descriptors, namely Object Detection Features (ODF) and Saliency Detection Features (SDF). The ODF descriptor

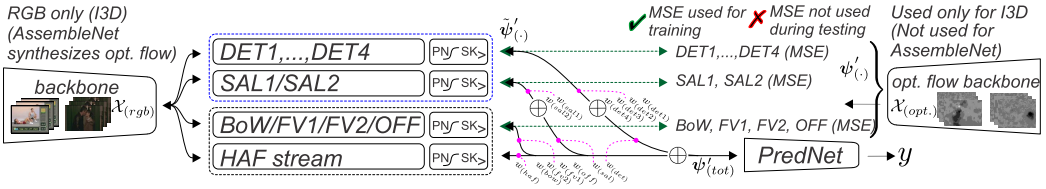


Figure 4.2: We build on DEEP-HAL [Wang et al., 2019d] which includes I3D RGB and Optical Flow networks (the latter net. is used only during training). For AssembleNet and AssembleNet++, the backbone encodes both RGB and the optical flow, which is synthesized on the fly from RGB frames. For the I3D variant, we remove the prediction and the last 1D conv. layers from I3D RGB and optical flow streams, we feed the 1024×7 feature representations $\mathcal{X}_{(rgb)}$ into *Bag-of-Words* (*BoW*), *Fisher Vector* (*FV*), *the Optical Flow Features* (*OFF*) and *the High Abstraction Features* (*HAF*) streams (dashed black) followed by the *Power Normalization* (*PN*) and *Sketching* (*SK*) blocks. The OFF stream is supervised by $\mathcal{X}_{(opt.)}$. For the AssembleNet variant, we obtain the 2048 feature representations $\mathcal{X}_{(rgb)}$ and do not use the OFF stream/optical flow backbone. Moreover, we introduce *DET1*, ...,*DET4*, *SAL1* and *SAL2* streams corresponding to our detector- and saliency-based descriptors (dashed blue). The resulting feature vectors $\tilde{\boldsymbol{\psi}}'_{(\cdot)}$, where $(\cdot)$ denotes the stream name *e.g.*, $(det1)$, ..., $(det2)$ *etc.*, are reweighted by corresponding weights $w_{(\cdot)}$ (magenta lines) and aggregated (sum) by $(\oplus)$. All $\tilde{\boldsymbol{\psi}}_{(\cdot)}$ are reweighted, aggregated (sum) and fed to *Prediction Network* (*PredNet*). By ✓, we indicate that the Mean Square Error (MSE) losses are used during training to supervise all streams outputting $\tilde{\boldsymbol{\psi}}'_{(\cdot)}$ by the ground-truth $\boldsymbol{\psi}'_{(\cdot)}$. By ✗, we indicate that the MSE losses are switched off for testing and $\tilde{\boldsymbol{\psi}}'_{(\cdot)}$ are hallucinated/fed into PredNet to obtain labels y.

leverages faster R-CNN detector [Ren et al., 2015] based on backbones such as (i) Inception V2 [Szegedy et al., 2016], (ii) Inception ResNet V2 [Szegedy et al., 2017], (iii) ResNet101 [He et al., 2016] and (iv) NASNet [Zoph et al., 2018]. The Inception V2, Inception ResNet V2 and NASNet are pre-trained on the COCO dataset [Lin et al., 2014] (91 object classes), whereas the ResNet101 is pre-trained on the AVA v2.1 dataset [Gu et al., 2018] (80 human AR classes). The above detectors are applied to training videos to identify humans and objects. Such detected objects together with their relevance and class labels summarized with our descriptor encourage the AR pipeline to focus on semantically important regions and actors relevant to the task of action recognition. Figure 4.1a shows a few of bounding boxes detected by these four detectors.

The SDF leverages image- and video-level saliency detectors such as MNL [Zhang et al., 2018c] and ACLNet [Zhang et al., 2019c] with the goal of identifying salient regions correlating with the human gaze in spatial and temporal sense. Saliency maps extracted from training videos and summarized by our descriptor help the AR pipeline learn spatial and temporal regions correlating with actions. Figures 4.1b and 4.1c show saliency maps from region-wise and temporal saliency detectors.

IDT descriptors are fused with the majority of modern AR pipelines [Fernando and Gould, 2016; Cherian et al., 2017b, 2018; Wang and Cherian, 2018; Choutas et al., 2018]

at the classifier level for the best performance while DEEP-HAL [Wang et al., 2019d] learns to hallucinate, and feeds them into the classification branch called PredNet. In this chapter, we go further and prepare two compact descriptors, ODF and SDF, and hallucinate them within DEEP-HAL. We equip each hallucination branch with a weighting mechanism adjusted per epoch to attain the best results. Figure 4.2 illustrates DEEP-HAL at the conceptual level.

For ODF descriptors, we concatenate together per bounding box per frame (i) the one-hot detection and (ii) ImageNet [Russakovsky et al., 2015] scores, (iii) embedded confidence scores, (iv) embedded bounding box coordinates, and (v) embedded normalized frame index. For all bounding boxes, we stack such features into a matrix. Inspired by the characteristic function of the probability density fun., we extract the mean, leading eigenvectors of covariance, skewness and kurtosis. For SDF descriptors, per frame, we encode saliency via (i) kernelized descriptor on spatio-angular gradient distributions of saliency maps and (ii) intensity patterns. We obtain an ODF per detector and an SDF per saliency detector. Our contributions are as follows:

i. We propose to utilize the object and human detectors to enhance the performance of AR pipelines.

ii. We design two types of statistically motivated high-order compact descriptors, Object Detection Features and Saliency Detection Features, for the use in AR pipelines.

iii. We build on the recent DEEP-HAL pipeline [Wang et al., 2019d] but we introduce AssembleNet and AssembleNet++ apart from I3D backbone. Moreover, we introduce a weight learning mechanism for hallucinated feature vectors, and ODF and SDF are hallucinated which leads to the state-of-the-art performance.

4.2 Related Work

Below, we describe handcrafted spatio-temporal video descriptors, their encoding strategies and the optical flow used by DEEP-HAL [Wang et al., 2019d]. We also describe deep learning pipelines for video classification. Finally, we discuss the object category and human detectors followed by the spatial and temporal saliency detectors used by us.

Early video descriptors. Early AR used on spatio-temporal interest point detectors [Laptev, 2005; Dollár et al., 2005; Chakraborty et al., 2012; Willems et al., 2008; Li et al., 2014; Wang et al., 2011] and spatio-temporal descriptors [Dalal et al., 2006; Scovanner et al., 2007; Uijlings et al., 2014; Wang et al., 2011, 2013b; Wang and Schmid, 2013] which capture various appearance and motion statistics. As spatio-temporal interest point detectors are unable to capture long-term motion patterns, a Dense Trajectory (DT) [Wang et al., 2011] approach densely samples feature points in each frame to track them in the video (via optical flow). Then, multiple descriptors are extracted along trajectories to capture shape, appearance and motion cues. As DT cannot compensate for the camera motion, the IDT [Wang and Schmid, 2013; Wang

et al., 2013b] estimates the camera motion to remove the global background motion. IDT also removes inconsistent matches via a human detector. For spatio-temporal descriptors, IDT employs HOG [Freeman and Roth, 1994], HOF [Dalal et al., 2006] and MBH [Wang et al., 2013b]. HOG [Freeman and Roth, 1994] contains statistics of the amplitude of image gradients w.r.t. the gradient orientation, thus it captures the static appearance cues. In contrast, HOF [Dalal et al., 2006] captures histograms of optical flow while MBH [Wang et al., 2013b] captures derivatives of the optical flow, thus it is highly resilient to the global camera motion whose cues cancel out due to derivatives. Thus, HOF and MBH contain the zero- and first-order optical flow statistics. Other spatio-temporal descriptors include HOG-3D [Kläser et al., 2008], SIFT3D [Scovanner et al., 2007], SURF3D [Willems et al., 2008] and LTP [Yeffet and Wolf, 2009].

We use the DEEP-HAL [Wang et al., 2019d] setup. We encode HOG, HOF, and MBH descriptors on the Improved Dense Trajectories [Wang et al., 2011; Cherian et al., 2017b; Choutas et al., 2018] via BoW [Sivic and Zisserman, 2003; Csurka et al., 2004] and FV [Perronnin and Dance, 2007; Perronnin et al., 2010].

BoW/FV encoding. BoW [Sivic and Zisserman, 2003; Csurka et al., 2004] uses a k-means vocabulary to which local descriptors are assigned. Variants include Soft Assignment (SA) [van Gemert et al., 2010; Koniusz and Mikolajczyk, 2011] and Localized Soft Assignment (LcSA) [Lingqiao et al., 2011; Koniusz et al., 2013b]. As we use DEEP-HAL [Wang et al., 2019d], we use BoW [Csurka et al., 2004] with Power Normalization [Koniusz et al., 2013b], and FV [Perronnin and Dance, 2007; Perronnin et al., 2010] which capture first- and second-order statistics of local descriptors assigned to GMM clusters. DEEP-HAL [Wang et al., 2019d] setup describes how to obtain the BoW/FV global descriptors.

Optical flow. Older optical flow methods cope with small displacements [Horn and Schunck, 1981; Papenberg et al., 2006] while newer methods cope with larger displacements *e.g.*, Large Displacement Optical Flow (LDOF) [Brox and Malik, 2011]. Recent methods use non-rigid descriptor or segment matching [Weinzaepfel et al., 2013; Braux-Zin et al., 2013], or edge-preserving interpolation [Revaud et al., 2015]. We use LDOF [Papenberg et al., 2006].

Object detectors. Modern deep learning methods include Region-based Convolutional Neural Networks (R-CNN) [Girshick et al., 2016], its faster variants [Girshick, 2015; Ren et al., 2015], its mask-based variants [He et al., 2017], and YOLO [Redmon et al., 2015], YOLO v2, YOLO v3 *etc.*, which use a single network for efficiency.

In this chapter, we use the faster R-CNN detector [Ren et al., 2015] with backbones such as (i) Inception V2 [Szegedy et al., 2016], (ii) Inception ResNet V2 [Szegedy et al., 2017], (iii) ResNet101 [He et al., 2016] and (iv) NASNet [Zoph et al., 2018]. As the Inception V2, Inception ResNet V2 and NASNet are pre-trained on the COCO dataset [Lin et al., 2014], they detect from 91 object classes good at summarizing *e.g.*, indoor environments and helping us associate the scene context with actions. The ResNet101 model is pre-trained on the AVA v2.1 dataset [Gu et al., 2018] with 80 different human actions, thus directly helping human-centric action recognition problems.

In addition to detection scores, we describe each bounding box with ImageNet [Russakovsky et al., 2015] scores from pre-trained Inception ResNet V2 [Szegedy et al., 2017].

Saliency detectors. Image regions correlating with human visual attention are detected by saliency detectors in the form of saliency maps. Deep saliency models [Wang et al., 2016a; Hou et al., 2017] outperform conventional saliency detectors [Zhu et al., 2014] but they require pixel-wise annotations. Recent models include MNL [Zhang et al., 2018c] (weakly-supervised model), RFCN [Wang et al., 2016a] (a fully-supervised model) and a cheap non-CNN Robust Background Detector (RBD) [Zhu et al., 2014] (see survey [Borji et al., 2015] for more details).

For the spatial saliency, we use MNL [Zhang et al., 2018c] trained on multiple noisy labels from weak/noisy unsupervised handcrafted saliency models. For temporal saliency, we use a CNN-LSTM ACLNet [Zhang et al., 2019c].

Deep learning AR. Early AR CNN models use frame-wise features and average pooling [Karpathy et al., 2014] discarding the temporal order. Thus, frame-wise CNN scores were fed to LSTMs [Donahue et al., 2015] while the two-stream networks [Simonyan and Zisserman, 2014] compute representations per RGB frame and per 10 stacked optical flow frames. Finally, spatio-temporal 3D CNN filters [Ji et al., 2013; Tran et al., 2015; Feichtenhofer et al., 2016a; Varol et al., 2018] model spatio-temporal patterns.

As two-stream networks [Simonyan and Zisserman, 2014] discard the temporal order, rank pooling [Fernando et al., 2015; Fernando and Gould, 2016; Cherian et al., 2018; Wang and Cherian, 2018] and higher-order pooling [Cherian et al., 2017b; Koniusz et al., 2016a,c; Fang et al., 2019; Koniusz et al., 2020] are popular. A recent I3D model [Carreira and Zisserman, 2017] 'inflates' 2D CNN filters pre-trained on ImageNet to spatio-temporal 3D filters, and implements temporal pooling. PAN [Zhang et al., 2019a] proposes a motion cue called Persistence of Appearance that enables the network to distill the motion information directly from adjacent RGB frames. Approach [Liu et al., 2019b] uses bootstrapping with long-range temporal context attention while approach [Kumar et al., 2020] proposes a graph attention model to explore the semantics. Slow-I-Fast-P (SIFP) [Li et al., 2020a] for compressed AR contains the slow and fast pathways I and P, resp., receiving a sparse sampling I-frame clip and a dense sampling pseudo optical flow clip.

AssembleNet [Ryoo et al., 2020b] automatically finds a neural architecture with a good connectivity to capture spatio-temporal interactions for AR through NAS. AssembleNet++ [Ryoo et al., 2020a] further learns the interactions between raw appearance and/or motion features and spatial object information through learning dynamic attention weights and search through the inter-block attention connectivity.

We use DEEP-HAL [Wang et al., 2019d] which employs a 1D convolution for temporal pooling (I3D net.). We also investigate the use of AssembleNet and AssembleNet++ as backbones to show that our proposed object and saliency descriptors are independent of the backbone. We focus on the design/ability of ODF/SDF to supervise DEEP-HAL.

Power Normalization. For BoW/FV and CNN-based streams, the so-called burstiness defined as *'the property that a given visual element appears more times in an image than a statistically independent model would predict'* [Jégou et al., 2009] has to be tackled. Thus, we employ Power Normalization [Koniusz et al., 2013b, 2021, 2016c, 2018b; Koniusz and Zhang, 2020] which suppresses the burstiness via the so-called MaxExp pooling [Koniusz et al., 2013b].

4.3 Approach

Notations. We use boldface uppercase letters to express matrices *e.g.*, $\boldsymbol{M}, \boldsymbol{P}$, regular uppercase letters with a subscript to express matrix elements *e.g.*, P_{ij} is the $(i,j)^{\text{th}}$ element of $\boldsymbol{P}$, boldface lowercase letters to express vectors, *e.g.* $\mathbf{x}, \boldsymbol{\phi}, \boldsymbol{\psi}$, and regular lowercase letters to denote scalars. Vectors can be numbered *e.g.*, $\mathbf{x}_n$ while regular lowercase letters with a subscript express an element of vector *e.g.*, $\mathbf{x}_i$ is the i^{th} element of $\mathbf{x}$. Operators ';' and ',' concatenate vectors along the first and second mode, respectively, $\circledcirc_{i\in\mathcal{I}_K}\mathbf{v}_i = [\mathbf{v}_1;...;\mathbf{v}_K]$ and $\circledcirc^2_{i\in\mathcal{I}_K}\mathbf{v}_i = [\mathbf{v}_1,...,\mathbf{v}_K]$ concatenate a group of vectors in the first and second mode, respectively, $\oplus$ denotes the aggregation (sum) while $\mathcal{I}_d$ denotes an index set of integers $\{1,...,d\}$.

Our pipeline is illustrated in Figure 4.2. It consists of (i) streams already present in DEEP-HAL [Wang et al., 2019d] such as the FV/BoW streams (black), the High Abstraction Features (HAF) stream and the Optical Flow Features (OFF) which are fed into (ii) the Prediction Network abbreviated as PredNet. In this chapter we focus on two non-trivial streams, that is the Object Detection Features and Saliency Detection Features (dashed blue) (ODF and SDF for short).

BoW/FV/OFF streams take the backbone intermediate representations generated from the RGB frames and learn to hallucinate BoW/FV and the optical flow (I3D only) representations via the MSE loss between the ground-truth BoW/FV/OFF and the outputs of BoW/FV/OFF streams. For AssembleNet / AssembleNet++, RGB and optical flow are combined by the backbone, thus we remove the OFF stream. The same MSE loss is applied to the ODF and SDF streams. However, the design of compact ground-truth ODF and SDF descriptors is one of our main contributions.

The HAF stream processes the backbone representations prior to combining them with the hallucinated streams. PredNet fuses the combined BoW/FV/OFF/HAF and our new ODF and SDF to learn actions on videos. Below, we start by describing how we obtain our ODF and SDF descriptors before we describe modules of DEEP-HAL [Wang et al., 2019d] and our modifications. One change is that we learn weights for the weighted mean pooling (*i.e.*, $\sum_i w'_i\boldsymbol{\psi} / \sum_i w'_i$) of each stream to avoid concatenation of streams (prevent overparametrization).

4.3.1 Statistical Motivation

We motivate the use of higher-order statistics. To compare videos, we want to capture a distribution of local features/descriptors *e.g.*, detection scores. The characteristic function $\varphi_Y(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{v}\sim Y}\left(\exp(i\boldsymbol{\omega}^T\boldsymbol{v})\right)$ describes the probability density $f_Y(\boldsymbol{v})$ of

some video features (local features $\boldsymbol{v} \sim Y$). We obtain the Taylor expansion of the characteristic function:

$$\mathbb{E}_{\boldsymbol{v}\sim Y}\left(\sum_{r=0}^{\infty}\frac{i^j}{r!}\langle \boldsymbol{v},\boldsymbol{\omega}\rangle^r\right) \approx \frac{1}{N}\sum_{n=0}^{N}\sum_{r=0}^{\infty}\frac{i^r}{r!}\langle \uparrow\otimes_r \boldsymbol{v}_n, \uparrow\otimes_r \boldsymbol{\omega}\rangle = \tag{4.1}$$

$$\sum_{r=0}^{\infty}\frac{i^r}{r!}\left\langle \frac{1}{N}\sum_{n=0}^{N}\uparrow\otimes_r \boldsymbol{v}_n, \uparrow\otimes_r \boldsymbol{\omega}\right\rangle = \sum_{r=0}^{\infty}\left\langle \boldsymbol{\mathcal{X}}^{(r)}, \frac{i^r}{r!}\uparrow\otimes_r \boldsymbol{\omega}\right\rangle,$$

where i is the imaginary number, and a tensor descriptor $\boldsymbol{\mathcal{X}}^{(r)} = \frac{1}{N}\sum_{n=0}^{N}\uparrow\otimes_r \boldsymbol{v}_n$. In principle, with infinite data and infinite moments, one can fully capture $f_Y(\boldsymbol{v})$. In practice, first-, second- and third-order moments are typically sufficient, however, second- and third-order tensors grow quadratically and cubically w.r.t. the size of $\boldsymbol{v}$. Thus, in what follows, we represent second-order moments not by a covariance matrix but by the subspace corresponding to the top n' leading eigenvectors. We also make use of the corresponding eigenvalues of the signal. Finally, it suffices to notice that $\boldsymbol{\kappa}^{(r)} = \mathrm{diag}\left(\boldsymbol{\mathcal{X}}^{(r)}\right)$ corresponds to the notion of order r cumulants used in calculations of skewness ($r=3$) and kurtosis ($r=4$) but it grows linearly w.r.t. the size of $\boldsymbol{v}$. Thus, in what follows, we use the ℓ_2 norm normalized mean, leading eigenvectors (and trace-normalized eigenvalues), skewness and kurtosis (rather than coskewness and cokurtosis) to obtain compact representation of ODF and SDF.

4.3.2 Positional Embedding

Let $G_\sigma(\mathbf{x}-\mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|_2^2/2\sigma^2)$ denote a standard Gaussian RBF kernel centered at $\mathbf{x}'$ and having a bandwidth σ. Kernel linearization refers to rewriting this G_σ as an inner-product of two infinite-dimensional feature maps. To obtain these maps, we use a fast approximation method based on probability product kernels [Jebara et al., 2004]. Specifically, we employ the inner product of d''-dimensional isotropic Gaussians given $\mathbf{x},\mathbf{x}'\in\mathbb{R}^{d''}$. Thus, we have:

$$G_\sigma(\mathbf{x}-\mathbf{x}') = \left(\frac{2}{\pi\sigma^2}\right)^{\frac{d''}{2}}\int_{\boldsymbol{\zeta}\in\mathbb{R}^{d''}} G_{\sigma/\sqrt{2}}(\mathbf{x}-\boldsymbol{\zeta})\, G_{\sigma/\sqrt{2}}(\mathbf{x}'-\boldsymbol{\zeta})\,\mathrm{d}\boldsymbol{\zeta}. \tag{4.2}$$

Eq. (4.2) is then approximated by replacing the integral with the sum over Z pivots $\boldsymbol{\zeta}_1,...,\boldsymbol{\zeta}_Z$, thus yielding a feature map $\boldsymbol{\phi}$ as:

$$\boldsymbol{\phi}(\mathbf{x};\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_Z}) = \left[G_{\sigma/\sqrt{2}}(\mathbf{x}-\boldsymbol{\zeta}_1),...,G_{\sigma/\sqrt{2}}(\mathbf{x}-\boldsymbol{\zeta}_Z)\right]^T, \tag{4.3}$$

$$\text{and } G_\sigma(\mathbf{x}-\mathbf{x}') \approx \left\langle \sqrt{c}\boldsymbol{\phi}(\mathbf{x}), \sqrt{c}\boldsymbol{\phi}(\mathbf{x}')\right\rangle, \tag{4.4}$$

where c is a const. Eq. (4.4) is the linearization of the RBF kernel. Eq. (4.3) is the feature map. $\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_Z}$ are pivots. As we use 1 dim. signals, we simply cover interval $[0;1]$ with Z equally spaced pivots. For clarity, we drop $\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_Z}$ and write $\boldsymbol{\phi}(\mathbf{x})$, *etc.*

4.3.3 Object Detection Features

Each object bounding box is described by the feature vector:

$$\boldsymbol{v} = \left[\boldsymbol{\delta}(y_{(det)}); \mathbf{y}_{(inet)}; \boldsymbol{\phi}(\varsigma); \circledcirc_{i \in \mathcal{I}_4} \boldsymbol{\phi}(v_i); \boldsymbol{\phi}\left(\frac{t-1}{\tau-1} \right) \right] \in \mathbb{R}^d, \tag{4.5}$$

where $\boldsymbol{\delta} = [0, ..., 1, ..., 0]^T$ is a vector with all zeros but a single 1 placed at the location y. As we have 91 object classes for detectors trained on the COCO dataset and 80 classes for a detector trained on the AVA v2.1 dataset, we simply assume $y_{(det)} \in \mathcal{I}_{91+80}$, that is, the labels $0, ..., 91$ describe classes from COCO while classes $92, ..., 80+91$ describe classes from AVA v2.1. Moreover, $\mathbf{y}_{(inet)} \in \mathbb{R}^{1001}$ is an ℓ_1 norm normalized ImageNet classification score, $0 \leq \varsigma \leq 1$ is the detector confidence score, $v_0, ..., v_4$ are the top-left and bottom-right Cartesian coordinates of a bounding box normalized in range $[0; 1]$, and $(t-1)/(\tau-1)$ is the frame index normalized w.r.t. the video sequence length τ. For feature maps $\boldsymbol{\phi}(\cdot)$ defined in Eq. (4.3), we simply use $Z = 7$ pivots and the σ of RBF is set to 0.5. Finally, for all detections per video from a given detector, we first compute the mean $\boldsymbol{\mu}([\boldsymbol{v}_1, ..., \boldsymbol{v}_N]) \in \mathbb{R}^d$ (we write $\boldsymbol{\mu}$) where N is the total number of detections. Then, we form a matrix $\boldsymbol{\Upsilon} \in \mathbb{R}^{d \times N}$:

$$\boldsymbol{\Upsilon} = \frac{1}{J} \left[\frac{1}{K_1} \left[\circledcirc^2_{i \in \mathcal{I}_{K_1}} (\boldsymbol{v}_{i1} - \boldsymbol{\mu}) \right], ..., \frac{1}{K_J} \left[\circledcirc^2_{i \in \mathcal{I}_{K_J}} (\boldsymbol{v}_{iJ} - \boldsymbol{\mu}) \right] \right], \tag{4.6}$$

where K_j denotes a number of detections per frame $j \in \mathcal{I}_J$, from which we extract higher-order statistical moments as described below. As N is large and its size varies from video to video, hallucinating $\boldsymbol{\Upsilon}$ directly is not feasible (nor it has invariance properties).

Firstly, we obtain $\mathbf{U}\boldsymbol{\lambda}\mathbf{V} = \text{svd}(\boldsymbol{\Upsilon})$ rather than $\mathbf{U}\boldsymbol{\lambda}^2\mathbf{U}^T = \text{eig}(\boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^T)$ as $N \ll d$, where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ...]$. Take $\boldsymbol{\mathcal{X}}^{(r)}(\{\mathbf{v} - \boldsymbol{\mu}\}_{n=0}^N)$ (which we abbreviate to $\boldsymbol{\mathcal{X}}^{(r)}$) and $\boldsymbol{\kappa}^{(r)} = \text{diag}\left(\boldsymbol{\mathcal{X}}^{(r)}\right)$ defined in Section 4.3.1. We form our multi-moment descriptor $\boldsymbol{\psi}_{(det)} \in \mathbb{R}^{d(4+n')}$, $n' \geq 1$:

$$\boldsymbol{\psi}_{(det)} = \left[\frac{\boldsymbol{\mu}}{||\boldsymbol{\mu}||_2}; \circledcirc^2_{i \in \mathcal{I}_{n'}} \mathbf{u}_i \left(\boldsymbol{\mathcal{X}}^{(2)} \right); \frac{\boldsymbol{\kappa}^{(3)}}{\left(\boldsymbol{\kappa}^{(2)}\right)^{3/2}}; \frac{\boldsymbol{\kappa}^{(4)}}{\left(\boldsymbol{\kappa}^{(2)}\right)^2}; \frac{\text{diag}(\boldsymbol{\lambda}^2)}{\sum_i \lambda_{ii}^2} \right], \tag{4.7}$$

The composition of Eq. (4.7) is described in Section 4.3.1. It is easy to verify that $\frac{\boldsymbol{\kappa}^{(3)}}{\left(\boldsymbol{\kappa}^{(2)}\right)^{3/2}}$ and $\frac{\boldsymbol{\kappa}^{(4)}}{\left(\boldsymbol{\kappa}^{(2)}\right)^2}$ are the empirical versions of skewness and kurtosis given by $\frac{\mathbb{E}_{\boldsymbol{v} \sim \Upsilon}\left((\boldsymbol{v}-\boldsymbol{\mu})^3\right)}{\mathbb{E}^{3/2}_{\boldsymbol{v} \sim \Upsilon}((\boldsymbol{v}-\boldsymbol{\mu})^2)}$ and $\frac{\mathbb{E}_{\boldsymbol{v} \sim \Upsilon}\left((\boldsymbol{v}-\boldsymbol{\mu})^4\right)}{\mathbb{E}^{2}_{\boldsymbol{v} \sim \Upsilon}((\boldsymbol{v}-\boldsymbol{\mu})^2)}$.

4.3.4 Saliency Detection Features

We extract directional gradients from saliency frames by discretised gradient operators $[-1, 0, 1]$ and $[-1, 0, 1]^T$ and obtain gradient amplitude and orientation maps $\boldsymbol{\Lambda}$ and $\boldsymbol{\theta}$

per frame encoded by:

$$\boldsymbol{v}'_{(sal)} = \sum_{i \in \mathcal{I}_W, j \in \mathcal{I}_H} \Lambda_{ij} \boldsymbol{\phi}(\theta_{ij}/(2\pi)) \otimes \boldsymbol{\phi}\left(\frac{i-1}{W-1}\right) \otimes \boldsymbol{\phi}\left(\frac{j-1}{H-1}\right), \tag{4.8}$$

where $\otimes$ is the Kronecker product and $\boldsymbol{\phi}(\theta)$ follows Eq. (4.3) with the exception that the assignment to Gaussians is realized in the modulo ring to respect the periodical nature of θ. We encode $\boldsymbol{\phi}(\theta)$ with 12 pivots which encode the orientation of gradients. The remaining maps $\boldsymbol{\phi}(\cdot)$ are encoded with 5 pivots each, which correspond to spatial binning. Note that $\boldsymbol{v}'_{(sal)}$ (we write $\boldsymbol{v}'$) is similar to a single CKN layer [Mairal et al., 2014] but is simpler: for one dimensional variables we sample pivots (*c.f.* learn) for maps $\boldsymbol{\phi}(\cdot)$. Each saliency frame is described as a feature vector $\boldsymbol{v}^\dagger = [\boldsymbol{v}'/||\boldsymbol{v}'||_2; \mathbf{I}_:/||\mathbf{I}_:||_1] \in \mathbb{R}^{d^\dagger}$, where $\mathbf{I}_:$ is a vectorized low-resolution saliency map. Thus, $\boldsymbol{v}^\dagger$ captures the directional gradient statistics and the intensity-based gist of saliency maps. Subsequently, we compute the mean $\boldsymbol{\mu}([\boldsymbol{v}_1^\dagger, ..., \boldsymbol{v}_J^\dagger]) \in \mathbb{R}^{d^\dagger}$ (we simply write $\boldsymbol{\mu}$) where J is the total number of frames per video. Then, we obtain $\boldsymbol{\Upsilon}^\dagger = \left[\boldsymbol{v}_1^\dagger, ..., \boldsymbol{v}_J^\dagger\right]/J \in \mathbb{R}^{d^\dagger \times J}$which is compactly described by the multi-moment Eq. (4.7) resulting in $\boldsymbol{\psi}_{(sal)} \in \mathbb{R}^{d(4+n^\dagger)}$.

4.3.5 Hallucinating Streams/High Abstr. Features

Each hallucinating stream takes as input the backbone intermediate representation $\mathcal{X}_{(rgb)}$ of size 1024×7 obtained by removing the classifier and the last 1D conv. layer of I3D pre-trained on Kinetics-400. For AssembleNet/AssembleNet++, instead of the classification layer (FC layer), we use a 2048 dimensional output from 3D AveragePooling layer. For the BoW/FV/OFF and HAL streams, we follow the steps described in the DEEP-HAL approach [Wang et al., 2019d]. For all streams, we use a Fully Connected (FC) unit shown in Figure 4.3a. Each stream uses Power Normalization (PN) realized via SigmE and Sketching (SK) from 1000 to 512 dim via $\tilde{\boldsymbol{\psi}}'_{(\cdot)} = \tilde{\boldsymbol{P}}_{(\cdot)} \tilde{\boldsymbol{\psi}}_{(\cdot)}$. Outputs $\tilde{\boldsymbol{\psi}}'$ can be now aligned with ground-truth $\boldsymbol{\psi}'_{(\cdot)}$ described below. The same steps are applied to High Abstraction Features (HAF), combined with other streams, and also fed into PredNet (see Fig. 4.2). While hallucinating streams co-supervise the backbone via external ground-truth tasks, HAF simply passes the backbone features into PredNet.

Ground-truth BoW/FV/OFF. We follow the DEEP-HAL setup [Wang et al., 2019d] and apply PCA to a concatenation of IDT trajectories (30 dim.), HOG (96 dim.), HOF (108 dim.), MBHx (96 dim.) and MBHy (96 dim.). The resulting 213 dim. local descriptors are encoded by FV and BoW with a 256 dim. and a 1000 dim. GMM and k-means dictionaries. For the OFF stream (not used with AssembleNet or AssembleNet++), we pre-computed I3D with LDOF $\mathcal{X}_{(opt.)}$ (Fig. 4.2). All ground-truth representations were Power Normalized by SigmE/sketched to 512 dim. each via $\boldsymbol{\psi}'_{(\cdot)} = \boldsymbol{P}_{(\cdot)} \boldsymbol{\psi}_{(\cdot)}$ and fed into the MSE loss. No ground-truth testing data is used in training/testing.

Ground-truth DET1, ...,DET4/SAL1/SAL2. The ODF ground-truth training representations are of size $1214 \times N$, where N is the total number of bounding boxes per video

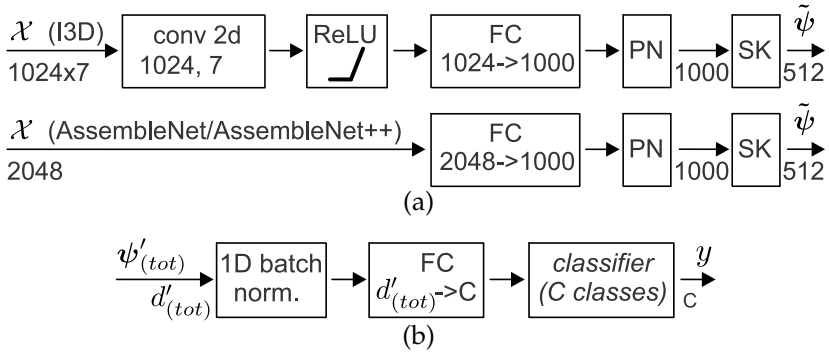


Figure 4.3: Stream details. Figure 4.3a a shows the stream architecture used by us for the FV, BoW, OFF, HAF, DET1, ...,DET4, SAL1 and SAL2 streams. Figure 4.3b shows our PredNet. Operation and their parameters are in each block *e.g.*, *conv2d* and its number of filters/size, *Power Normalization* (*PN*) and *Sketching* (*SK*). We indicate the size of input and/or output under arrows.

(50–10000). The feature dim. 1214 is composed of 80+91 dim. one-hot detection classes, 6×7 are the $\boldsymbol{\phi}(\cdot)$-embedded confidence, bounding box coordinates and the frame number, 1001 is the ImageNet score. We also consider a variant without the RBF embedding: $\boldsymbol{\phi}(\mathbf{x})=\mathbf{x}$ ($1178\times N$ size). The SDF ground-truth training repr. are of size $556\times J$, where J is the number of frames per video. 300 dim. ($12\times5\times5$) concern spatio-angular gradient distributions and 256 dim. (16×16) concern the luminance of saliency maps. Each ODF/SDF is encoded per video with the multi-moment descriptor in Eq. (4.7) yielding $1178\times(4+n')$ and $556\times(4+n^{\dagger})$ compact representations (we vary n' and $n^{\dagger}$ between 1 and 5). ODF and SDF are Power Normalized by SigmE/sketched to 512 dim. each via $\boldsymbol{\psi}'_{(\cdot)}=\boldsymbol{P}_{(\cdot)}\boldsymbol{\psi}_{(\cdot)}$ and fed into the MSE loss. No ground-truth testing representations were used for training/testing.

4.3.6 Objective Function

During training, we combine MSE loss functions which co-supervise hallucination streams with the classifier:

$$
\begin{aligned}
\ell^*(\boldsymbol{\mathcal{X}},\mathbf{y};\bar{\boldsymbol{\Theta}}) &= \frac{\alpha}{|\mathcal{H}|}\sum_{i\in\mathcal{H}}\left\|\tilde{\boldsymbol{\psi}}_i-\boldsymbol{\psi}'_i\right\|_2^2+\ell\Big(f(\boldsymbol{\psi}'_{(tot)};\boldsymbol{\Theta}_{(pr)}),\mathbf{y};\boldsymbol{\Theta}_{(\ell)}\Big)\,,\\
\text{where: } \forall i\in\mathcal{H},\tilde{\boldsymbol{\psi}}'_i &= \tilde{\boldsymbol{P}}_i g(\hbar(\boldsymbol{\mathcal{X}},\boldsymbol{\Theta}_i),\eta)\,,\boldsymbol{\psi}'_i=\boldsymbol{P}_i\boldsymbol{\psi}_i,\\
\boldsymbol{\psi}'_{(haf)} &= \boldsymbol{P}_{(haf)}g\Big(\hbar(\boldsymbol{\mathcal{X}},\boldsymbol{\Theta}_{(haf)}),\eta\Big)\,,\\
\boldsymbol{\psi}'_{(tot)} &= \frac{1}{|\mathcal{H}^*|+1}\Big(w_{(haf)}\boldsymbol{\psi}'_{(haf)}+\sum_{i\in\mathcal{H}^*}w_i\tilde{\boldsymbol{\psi}}'_i\Big),\\
\tilde{\boldsymbol{\psi}}'_{(det)} &= \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}}w_i\tilde{\boldsymbol{\psi}}'_i,\tilde{\boldsymbol{\psi}}'_{(sal)}=\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}w_i\tilde{\boldsymbol{\psi}}'_i.
\end{aligned}
\tag{4.9}
$$

The above equation is a trade-off between the MSE loss functions $\{\|\tilde{\boldsymbol{\psi}}'_i-\boldsymbol{\psi}'_i\|_2^2, i\in\mathcal{H}\}$ and the classification loss $\ell(\cdot,\mathbf{y};\boldsymbol{\Theta}_{(\ell)})$ with some label $\mathbf{y}\in\mathcal{Y}$ and parameters $\boldsymbol{\Theta}_{(\ell)}\equiv\{\boldsymbol{W},\mathbf{b}\}$. The trade-off is controlled by $\alpha\geq0$ while MSE is computed over hall. streams

$i \in \mathcal{H}$, and $\mathcal{H} \equiv \{(fv1), (fv2), (bow), (off), (det1), ..., (det4), (sal1), (sal2)\}$ is our set of hallucination streams. Moreover, $g(\cdot, \eta)$ is a Power Norm. in Eq. (3.6), $f(\cdot; \boldsymbol{\Theta}_{(pr)})$ is the PredNet module with parameters $\boldsymbol{\Theta}_{(pr)}$ which we learn, $\{\hbar(\cdot, \boldsymbol{\Theta}_i), i \in \mathcal{H}\}$ are the hallucination streams while $\{\tilde{\boldsymbol{\psi}}_i, i \in \mathcal{H}\}$ are resulting hallucinated BoW/FV/OFF/ODF/SDF representations. We set $\alpha = 1$. Moreover, $\hbar(\cdot, \boldsymbol{\Theta}_{(haf)})$ is the HAF stream with the sketched output $\boldsymbol{\psi}'_{(haf)} = \boldsymbol{P}_{(haf)} \boldsymbol{\psi}_{(haf)}$. For the hallucination streams, we learn parameters $\{\boldsymbol{\Theta}_i, i \in \mathcal{H}\}$ while for HAF, we learn $\boldsymbol{\Theta}_{(haf)}$. The full set of parameters we learn is defined as $\bar{\boldsymbol{\Theta}} \equiv (\{\boldsymbol{\Theta}_i, i \in \mathcal{H}\}, \boldsymbol{\Theta}_{(haf)}, \boldsymbol{\Theta}_{(pr)}, \boldsymbol{\Theta}_{(\ell)})$. Furthermore, $\{\tilde{\boldsymbol{P}}_i, i \in \mathcal{H}\}$ and $\{\boldsymbol{P}_i, i \in \mathcal{H}\}$ are the projection matrices for count sketching of streams $\{\tilde{\boldsymbol{\psi}}_i, i \in \mathcal{H}\}$ and the ground-truth feature vectors $\{\boldsymbol{\psi}_i, i \in \mathcal{H}\}$. Finally, for $\boldsymbol{\psi}'_{(tot)}$ is a weighted average of several streams fed into the PredNet module f. Moreover, $\mathcal{H}^* \equiv \{(fv1), (fv2), (bow), (off), (det), (sal)\}$, $\mathcal{D} \equiv \{(det1), ..., (det4)\}$ and $\mathcal{S} \equiv \{(sal1), (sal2)\}$. Section 3.3.4 details how to select matrices $\boldsymbol{P}$.

Let $\mathcal{T}$ be set to either $\mathcal{H}^*$, $\mathcal{D}$ or $\mathcal{S}$, then our weights are:

$$w_i = \frac{1}{|\mathcal{T}|} \frac{\max(w'^{\beta}_{i'} \rho)}{\sum_{j \in \mathcal{T}} \max(w'^{\beta}_{j'} \rho)}. \tag{4.10}$$

Prior to CNN training, we train an SVM on each ground-truth stream separately (using a manageable training subset), and we set weights w' proportionally to the accuracies obtained on the validation set. For the HAF stream, we simply set $w'_{(haf)} = \frac{1}{|\mathcal{H}^*|+1}$ and $\rho = 0.1$. For the first few epochs (*i.e.*, 10), we set $\beta = 0$ so that all streams receive equal weights. Subsequently, in each epoch, we run the Golden-section search to find the best $\beta \geq 0$. We start from initial boundary values $\beta \in \{0, 50\}$, we train an SVM on a manageable subset of training data and evaluate β on the validation set, and we update boundary values for the next epoch accordingly. Eq. (4.10) has a nice property: for $\beta = 0$, we have $w_i = 1/|\mathcal{T}|$. For $\beta \to \infty$, we have $w_i = 1$ if $w_i = \max(\{w_i\}_{i \in \mathcal{T}})$, otherwise $w_i = 0$. Thus, β interpolates between equalizing all weights and the winner-takes-all solution.

4.4 Experiments

4.4.1 Datasets and Evaluation Protocols

HMDB-51 [Kuehne et al., 2011] has 6766 internet videos/ 51 classes; each video has ∼20–1000 frames. We report the mean accuracy across three splits.

YUP++ [Feichtenhofer et al., 2017b] has 20 scene classes of video textures, 60 videos per class. Splits contain scenes captured by the static or moving camera. We use standard splits (1/9 dataset for training) for evaluation.

MPII Cooking Activities [Rohrbach et al., 2012] contains high-resolution videos of people cooking dishes. The 64 activities from 3748 clips include coarse actions *e.g.*, *opening refrigerator*, and fine-grained actions *e.g.*, *peel*, *slice*, *cut apart*. We use the mean Average Precision (mAP) over 7-fold cross validation. For human-centric protocol

	sp1	*sp2*	*sp3*	mean acc.
det1	42.00%	39.74%	40.39%	40.72%
det1	40.49%	40.13%	39.67%	40.09%
det3	43.78%	44.05%	41.97%	**43.26**%
det4	41.08%	39.22%	40.39%	40.23%
all+avg	42.50%	41.05%	41.01%	41.52%
all+max	43.25%	42.32%	42.09%	42.55%
all+wei	45.80%	44.52%	44.09%	**44.80**%
DEEP-HAL+all+avg	83.25%	82.24%	82.84%	82.77%
DEEP-HAL+all+max	83.18%	81.86%	82.84%	82.62%
DEEP-HAL+all+wei	84.01%	83.25%	83.10%	**83.45**%

Table 4.1: Evaluations of ODF on HMDB-51. (*top*) We evaluate backbones such as (*det1*) Inception V2, (*det2*) Inception ResNet V2, (*det3*) ResNet101 and (*det4*) NASNet. (*middle*) The average-pooled, max-pooled and the weighted mean combination of all detectors are given by (*all+avg*), (*all+max*) and (*all+wei*). (*bottom*) Pre-trained DEEP-HAL combined with all four detectors by the average-pooling, max-pooling and the weighted mean.

[Cherian et al., 2017a, 2018], we use the faster RCNN [Ren et al., 2015] to crop video around human subjects.

Charades [Sigurdsson et al., 2016] consist of of 9848 videos of daily indoors activities, 66500 clip annotations and 157 classes.

EPIC-Kitchens [Damen et al., 2018] is a multi-class egocentric dataset with 28K training videos associated with 331 noun and 125 verb classes. The dataset consists of 39,594 segments in 432 videos. We follow protocol [Baradel et al., 2018]. We evaluate our model on validation, standard seen (S1: 8047 videos), and unseen (S2: 2929 videos) test sets.

4.4.2 Evaluations

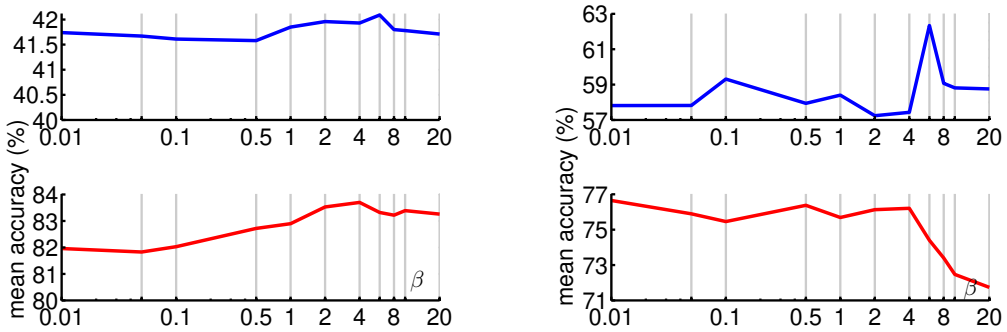
Below, we show the effectiveness of our method. For smaller datasets, we use the I3D backbone. For large Charades and EPIC-Kitchens, we additionally investigate AssembleNet and AssembleNet++ backbones. Firstly, we evaluate various design components.

Ground-truth ODF+SVM. Firstly, we evaluate our ODF on SVM given the HMDB-51 dataset. We set $n' = 3$ for Eq. (4.7) and compare various detector backbones and pooling strategies. Table 4.1 shows that all detectors perform similarly with (*det3*) being slightly better than other methods. Moreover, max-pooling on ODFs from all four detectors is marginally better than the average-pooling. However, only the weighted mean (*all+wei*) according to Eq. (4.10) outperforms (*det3*) which highlights the need for the robust aggregation of ODFs. Similarly, when we combine pre-trained DEEP-HAL with all detectors, the weighted mean (*DEEP-HAL+all+wei*) performs best. Table 4.2 shows the similar trend on YUP++. We trained SVM only on videos for which at least one detection occurred, thus a 75.74% accuracy is much lower than the main results reported on the full pipeline. Figure 4.4 shows that $\beta \neq 1$ has a positive impact on reweighting.

Ground-truth SDF. The SDF on HMDB-51 and YUP++ yielded 24.35% and 32.68%

	avg	*max*	*wei*
all	55.12%	42.34%	**60.52**%
DEEP-HAL+all	74.22%	71.85%	**75.74**%

Table 4.2: Pooling on YUP++. Results for the average-pooled (*avg*), max-pooled (*max*) and the weighted mean (*wei*) of all detectors (*all*) *vs*. pre-trained DEEP-HAL combined with all detectors by the average-pooling, max-pooling and the weighted mean.



(a) (b)

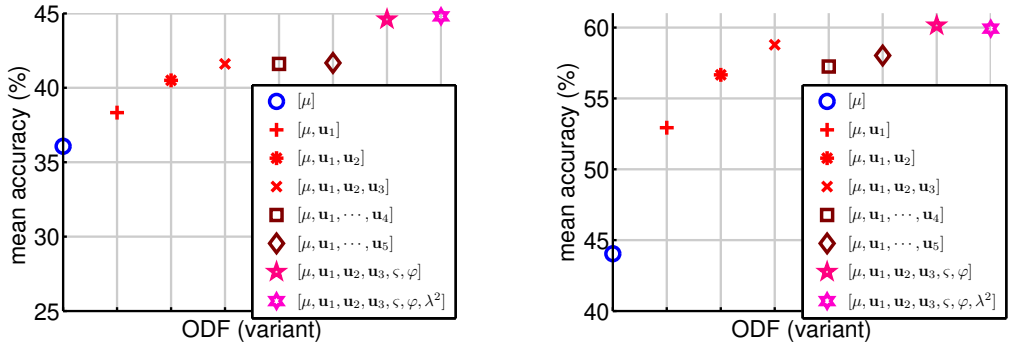
Figure 4.4: The impact of β in the weighted mean on the classification results. Figure 4.4a shows results for HMDB-51 on (*top*) four detectors combined+SVM and (*bottom*) DEEP-HAL with four detectors combined+SVM. Figure 4.4b shows results for YUP++.

accuracy. This is expected as SDFs do not capture a discriminative information per se but they locate salient spatial and temporal regions to focus the main network on them.

Multi-moment descr. Figure 4.5 shows that the concat. of the mean and three eigenvectors according to Eq. (4.7) yields good results but adding further vectors deteriorates the performance. Adding skewness and kurtosis (ς and $\boldsymbol{\varphi}$) further improves results, while adding eigenvalues has a limited impact.

HMDB-51. Table 4.3 shows several DEEP-HAL variants, which all hallucinate BoW/FV/OFF. DEEP-HAL with our reweighting mechanism. (*DEEP-HAL+W*) outperforms the original DEEP-HAL denoted as (*HAF/BoW/FV hal.*) [Wang et al., 2019d] by $\sim$0.8%. DEEP-HAL with our ODF and SDF descriptors (*DEEP-HAL+ODF*) and (*DEEP-HAL+SDF*) outperform (*HAF/BoW/FV hal.*) by $\sim$1.8% and $\sim$1.4%, resp. This shows that both ODF and SDF are effective. Combining DEEP-HAL, ODF and SDF outperform DEEP-HAL by $\sim$2.7% demonstrating the complementary nature of ODF and SDF. Utilizing our weighting mechanism with DEEP-HAL, ODF and SDF denoted as (*DEEP-HAL+W+ODF+SDF*) outperform (*HAF/BoW/FV hal.*) by $\sim$4.6%. Finally, DEEP-HAL with weighting, and ODF and SDF with RBF feature maps from Eq. (4.3) outperform (*HAF/BoW/FV hal.*) by $\sim$5.1%.

YUP++. Table 4.4 shows that ODF is better than SDF, that is (*DEEP-HAL+ODF*) and (*DEEP-HAL+SDF*) outperform (*HAF/BoW/FV hal.*) by $\sim$0.6% and $\sim$0.2%, resp. This is expected as YUP++ contains dynamic scenes without objects/specific saliency regions correlating with class concepts. However, a combination of detectors/saliency (*DEEP-HAL+SDF*) plus weighting (*DEEP-HAL+W+ODF+SDF*) plus the RBF maps



(a) (b)

Figure 4.5: ODF eval. on SVM on four detectors (the weighted mean). Fig. 4.5a and 4.5b show results on HMDB-51 and YUP++. $\boldsymbol{\mu}, \mathbf{u}_1, ..., \mathbf{u}_i, \boldsymbol{\varsigma}, \boldsymbol{\varphi}$, and $\boldsymbol{\lambda}^2$ correspond to the entries in Eq. (4.7).

	sp1	*sp2*	*sp3*	mean acc.
DEEP-HAL+W	83.94%	82.50%	83.34%	83.26%
DEEP-HAL+ODF	85.03%	83.59%	84.25%	84.29%
DEEP-HAL+SDF	84.64%	83.20%	83.82%	83.88%
DEEP-HAL+ODF+SDF	86.14%	83.66%	85.81%	85.20%
DEEP-HAL+W+ODF+SDF	87.78%	86.27%	87.06%	87.04%
DEEP-HAL+W+G+ODF+SDF	88.37%	86.80%	87.52%	**87.56**%

ADL+I3D 81.5% [Wang and Cherian, 2018] Full-FT I3D 81.3% [Carreira and Zisserman, 2017]
EvaNet (Ensemble) 82.3% [Piergiovanni et al., 2019] PA3D + I3D 82.1% [Yan et al., 2019]
HAF/BoW/FV exact 82.50% [Wang et al., 2019d] HAF/BoW/FV hal. 82.48% [Wang et al., 2019d]

Table 4.3: Evaluations of (*top*) our methods and (*bottom*) comparisons to the state of the art on HMDB-51.

(*DEEP-HAL+W+G+ODF+SDF*) outperform (*HAF/BoW/FV hal.*) by $\sim$0.7%, $\sim$1.6% and $\sim$1.8% accuracy, resp.

MPII. Table 4.5 shows a $\sim$3.0% mAP gain over (*HAF/BoW/FV hal.*) due to detectors capturing the human interaction with objects.

Charades. Table 4.6 (top) presents relative gains of our hallucination pipeline (*DEEP-HAL*) with weighted mean pooling (*W*) and the RBF maps (*G*) denoted as (*DEEP-HAL+W+G*). We evaluate Object Detection Features (*ODF*) and Saliency Detection Features (*SDF*) with 512 dim. sketching (*SK512*) and note that (*DEEP-HAL+W+G+ODF (SK512)*) outperforms (*DEEP-HAL+W+G+SDF (SK512)*), and both methods outperform the baseline (*HAF/BoW/FV hal.*) [Wang et al., 2019d].

Table 4.6 (bottom) shows that combining ODF and SDF into (*DEEP-HAL+W+G +SDF+ODF (SK512)*) yields 49.06% mAP which constitutes on a $\sim$**6**% gain over the baseline(*HAF/BoW/FV hal.*) [Wang et al., 2019d]. This demonstrates that ODF and SDF are highly complementary. Applying a larger sketch (*DEEP-HAL+W+G+ODF+SDF (SK1024)*) yields 50.14% mAP which matches the use (*DEEP-HAL+W+G+ODF+SDF (exact)*) that denotes a late fusion by concatenation of ODF and SDF with the stream resulting from DEEP-HAL fed into PredNet. Note that (*exact*) indicates that ODF and SDF are not hallucinated at the test time but they are computed. the results matching

	static	*dynamic*	*mixed*	mean stat/dyn	mean all
DEEP-HAL+ODF	95.00%	90.93%	93.52%	93.0%	93.2%
DEEP-HAL+SDF	94.96%	89.93%	93.58%	92.4%	92.8%
DEEP-HAL+SDF+ODF	95.10%	91.11%	93.61%	93.1%	93.3%
DEEP-HAL+W+SDF+ODF	96.30%	92.22%	94.17%	94.3%	94.2%
DEEP-HAL+W+G+SDF+ODF	96.30%	**92.40%**	**94.35%**	**94.4%**	**94.4%**
T-ResNet [Feichtenhofer et al., 2017b]	92.4%	81.5%	89.0%	87.0%	87.6%
ADL I3D [Wang and Cherian, 2018]	95.1%	88.3%	-	91.7%	-
HAF/BoW/FV hal. [Wang et al., 2019d]	94.8%	89.6%	93.3%	92.2%	92.6%
MSOE-two-stream [Hadji and Wildes, 2018]	**97.0%**	87.0%	91.8%	92.0%	91.9%

Table 4.4: Evaluations of (*top*) our methods and (*bottom*) comparisons to the state of the art on YUP++.

	sp1	*sp2*	*sp3*	*sp4*	*sp5*	*sp6*	*sp7*	mAP
DEEP-HAL+W+ODF+SDF	82.5	85.1	85.6	83.5	86.6	80.8	81.2	83.6%
DEEP-HAL+W+G+ODF+SDF	83.3	87.6	85.6	83.4	86.6	83.2	83.6	**84.8%**

KRP-FS+IDT 76.1% [Cherian et al., 2018] GRP+IDT 75.5% [Cherian et al., 2017a]
I3D+BoW/OFF MTL 79.1% [Wang et al., 2019d] HAF/BoW/OFF hal. 81.8% [Wang et al., 2019d]

Table 4.5: Evaluations of (*top*) our methods and (*bottom*) comparisons to the state of the art on MPII.

between (*DEEP-HAL+W+G+ODF+SDF (SK1024)*) and (*DEEP-HAL+W+G+ODF+SDF (exact)*) show that we can hallucinate ODF and SDF at the test time while regaining the full performance. We save computational time and hallucinate the detection and saliency features which boost results on Charades by $\sim$**6**% over the baseline.

Table 4.7 shows that our idea applied to AssembleNet and AssembleNet++ yields state of the art *e.g.*, we outperform these two networks by **4.5**% and **5.6**% mAP, respectively. We note that our detectors do not need to be computed at all at the test time.

In contrast, the best currently reported papers such as SlowFast networks [Feichtenhofer et al., 2019] and AssembleNet [Ryoo et al., 2020b] achieve 45.2% and 51.6% on Charades. As SlowFast networks and AssembleNet backbones can be used in place of I3D in our experimental setup, our approach is 'orthogonal' to these latest developments which focus on heavy mining for combinations of neural blocks/dataflow between them to obtain an 'optimal' pipeline. We achieve similar results with a simple approach based on self-supervised learning. Our pipeline is lightweight by comparison (no need for computations of the optical flow, or detections or segmentation masks at test time).

ImageNet (global score) *vs.* object detectors. Various scores from the object and saliency detectors which we use cannot be plugged directly into the DEEP-HAL due to the varying number of objects detected and the varying number of frames, thus we propose and use ODF and SDF descriptors. We also note that using a simplified variant of ODF which stacks up ImageNet scores per frame into a matrix (no detectors) to which we apply our multi-moment descriptor yielded $\sim$4% worse results on Charades than our DEEP-HAL+ODF (detectors-based approach) which yields 48.0% mAP. This is expected as ImageNet is trained in a multi-class setting (one object per image) while detectors let us model robustly distributions of object

HAF/BoW/FV hal. [Wang et al., 2019d]	*DEEP-HAL+ W+G+ODF (SK512)*	*DEEP-HAL+ W+G+SDF (SK512)*
43.1	47.22	45.30

DEEP-HAL+W+G+ ODF+SDF (SK512)	*DEEP-HAL+W+G+ ODF+SDF (SK1024)*	*DEEP-HAL+W+G+ ODF+SDF (exact)*
49.06	**50.14**	**50.16**

Table 4.6: Evaluations of our methods on Charades (I3D backbone).

AssembleNet++ 50 (Kinetics-400 pre-training)			
baseline	*ODF+SDF (SK512)*	*ODF+SDF (SK1024)*	*ODF+SDF (exact)*
53.8	55.81	**56.94**	**57.30**
AssembleNet++ 50 (without pre-training)			
baseline	*ODF+SDF (SK512)*	*ODF+SDF (SK1024)*	*ODF+SDF (exact)*
56.7	60.71	**61.98**	**62.29**

Table 4.7: Evaluations of our methods on the Charades dataset (AssembleNet and AssembleNet++ backbones). Note that we do not use segmentation masks for AssembleNet and AssembleNet++, thus baseline results reported by us are slightly lower compared to authors' results of 55.0% and 59.8% mAP, respectively.

	Verbs		Nouns		Actions	
	top-1	top-5	top-1	top-5	top-1	top-5
			Validation			
LFB Max [Wu et al., 2019a]	52.6	81.2	31.8	56.8	22.8	41.1
WeakLargeScale [Ghadiyaram et al., 2019]	58.4	84.1	36.9	60.3	26.1	42.7
DEEP-HAL+ODF+SDF(SK1024)	55.4	82.9	33.3	55.1	21.5	39.7
AssembleNet++ ODF+SDF(SK512)	57.2	84.6	34.8	56.4	23.2	41.3
AssembleNet++ ODF+SDF(SK1024)	58.7	85.6	36.0	57.3	**24.7**	**43.0**
AssembleNet++ ODF+SDF(exact)	60.0	86.7	37.1	59.2	**25.2**	**43.4**
			Test s1 (seen)			
TSN Fusion [Damen et al., 2018]	48.2	84.1	36.7	62.3	20.5	39.8
LFB Max [Wu et al., 2019a]	60.0	88.4	45.0	71.8	32.7	55.3
WeakLargeScale [Ghadiyaram et al., 2019]	65.2	87.4	45.1	67.8	34.5	53.8
DEEP-HAL+ODF+SDF(SK1024)	62.2	85.0	46.1	69.3	32.5	53.6
AssembleNet++ ODF+SDF(SK1024)	65.0	87.8	48.8	72.5	**35.0**	**56.1**
AssembleNet++ ODF+SDF(exact)	66.2	88.5	49.3	72.8	**35.8**	**56.8**
			Test s2 (unseen)			
TSN Fusion [Damen et al., 2018]	39.4	74.3	22.7	45.7	10.9	25.3
LFB Max [Wu et al., 2019a]	50.9	77.6	31.5	57.8	21.2	39.4
WeakLargeScale [Ghadiyaram et al., 2019]	57.3	81.1	35.7	58.7	25.6	42.7
DEEP-HAL+ODF+SDF(SK1024)	55.3	79.1	32.6	55.4	22.3	39.2
AssembleNet++ ODF+SDF(SK1024)	58.3	82.1	35.2	58.2	**25.9**	**42.9**
AssembleNet++ ODF+SDF(exact)	59.0	83.3	35.7	59.0	**27.3**	**44.0**

Table 4.8: Experimental results on the EPIC-Kitchens.

classes and locations per frame.

EPIC-Kitchens. Table 4.8 shows the experimental results. I3D and AssembleNet / AssembleNet++ learn human-like semantic features due to ODF/SDF, and there is no evidence a backbone can discover these without a guidance. By comparing MPII (3748 clips) with large EPIC-Kitchens (39594 clips) (both about cooking), SDF+ODF boost MPII from 81.8 to 84.8%, and SDF+ODF boost EPIC-Kitchens from 32.51% (DEEP-HAL) to 35.88% (on seen classes protocol), and from 22.33% (DEEP-HAL) to

	sp1	*sp2*	*sp3*	mean acc.
wei+flat	86.47%	85.56%	86.27%	86.10%
wei+3 levels	88.37%	86.80%	87.52%	**87.56**%

Table 4.9: Evaluations of the flat single level weighted mean (*wei+flat*) *vs*. three levels of weighted mean pooling (*wei+3 levels*) on HMDB-51.

	no. of frames	*av. frame count*	*no. of videos*	*no. of clips*	*no. of classes*
HMDB-51	628635	92.91	6766	6766	51
YUP++	166463	138.72	1200	1200	20
MPII	662394	176.73	44	3748	60
Charades	19978821	300.51	9848	66500	157
EPIC-Kitchens	$\sim$ 11.5M	290.43	432	39596	149

Table 4.10: Statistics of datasets used in our experiemnts.

27.32% (on unseen classes protocol). The boost is **3**% on both MPII and EPIC-Kitchens (nearly 10$\times$ more clips than MPII).

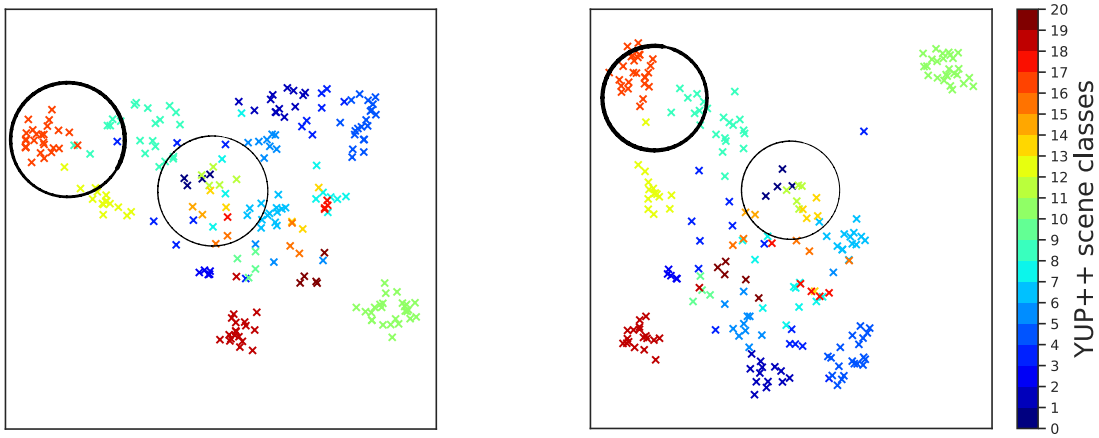
Reweighting mechanism. In this experiment, we employ pipeline (*DEEP-HAL+W+G+SDF+ODF (SK512)*) explained above. Typically, we use three levels of weighting mean pooling which are applied to (i) four object detectors constituting on ODF, (ii) two saliency detectors constituting on SDF, and (iii) the final combination of HAF/BOW/FV/OFF/ODF/SDF. Thus, below we investigate the performance of a single weighting mean pooling step applied simultaneously to four object detectors, two saliency detectors and the remaining streams.

Table 4.9 shows that using a flat single level weighted mean pooling yields 86.1% accuracy on the HMDB-51 which is a $\sim$1.4% less compared to utilizing three levels of weighted mean pooling. We expect that having one weighted mean pooling per modality is a reasonable strategy as for instance object category detectors may yield similar responses thus they should be first reweighted for the best 'combined detector' performance before being combined with highly complementary modalities.

Finally, Figure 4.8 (top) demonstrates how our Golden-search selects optimal β on the validation set of MPII (*split1*). Figure 4.8 (bottom) demonstrates the corresponding validation mAP (this is not the mAP score on the testing set). Note that for the first 10 epochs we use $\beta = 0$ and we start the Golden-search from epoch 11.

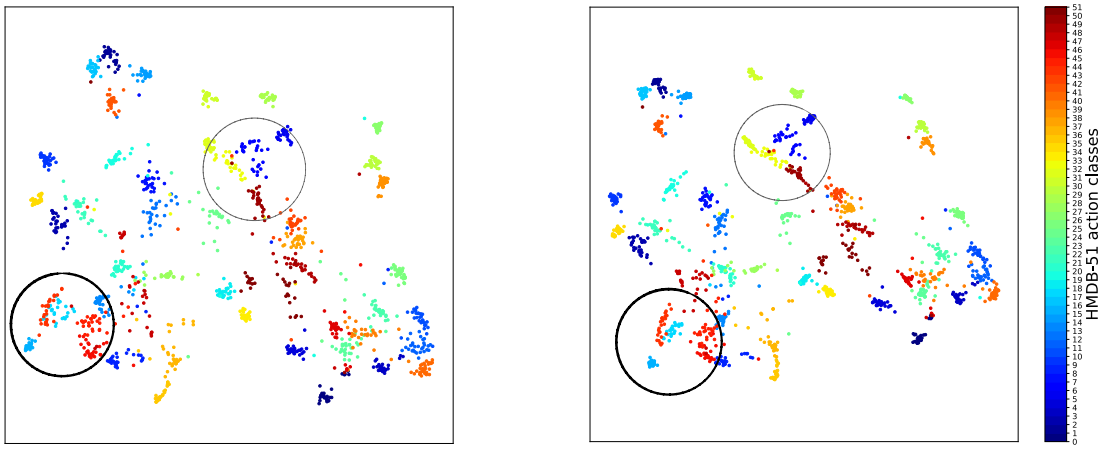
Dataset statistics and timing. Table 4.10 shows basic statistics re. datasets used in our experiments. We note that Charades with 66500 uniquely annotated clips, 157 action labels and an average frame count of 300 per clip is the largest among these datasets.

Table 4.11 introduces timing for object detectors used by our ODF descriptors during training. We note that detections with all four object detectors which we use take $\sim$1.47 second per frame. Thus, obtaining four ODF descriptors per clip (uniquely annotated sequence to train or classify) takes between 136 and 441 seconds. Table 4.12 introduces timing for saliency detectors used in our SDF descriptors during training. We note that detections with both saliency detectors which we use take $\sim$0.9 second per frame, and obtaining both SDF descriptors per clip takes between 84 and 271 seconds. We do note that the major computational cost is incurred due to detectors



(a) (b)

Figure 4.6: Visualization of the feature space (from PredNet) for DEEP-HAL in Fig. 4.6a and DEEP-HAL+ODF in Fig. 4.6b on the YUP++ dataset. For comparison, we circle regions with interesting changes.



(a) (b)

Figure 4.7: Visualization of the feature space (from PredNet) for DEEP-HAL in Fig. 4.7a and DEEP-HAL+ODF in Fig. 4.7b on the HMDB-51 dataset. For comparison, we circle regions with interesting changes.

rather than our ODF and SDF descriptors (their cost is minimal). We further note that the idea of learning these costly representations during training is very valuable. While the total computations per training clip vary between 220 and 712 seconds, during testing time we obtain these representations for free (milliseconds) thanks to DET1,...,DET4 and SAL1/SAL2 units from Figure 2 (the main submission). Assuming 25% of clips in charades for testing, that results in 137 days of computational savings on a single GPU (conversely, 1 day savings on 137 GPUs). Given the obtained 6% boost on Charades over the baseline without ODF and SDF, and the computational savings, we believe these statistics highlight the value of our approach.

Visualization using UMAP. Figure 4.6 is a visualization performed with UMAP

	DET1: Inception V2	*DET2: Inception ResNet V2*	*DET3: ResNet101 AVA*	*DET4: NASNet*	ODF total (+SVD)
sec. per frame	0.07	0.38	0.10	0.91	1.46 (+0.09)
s.p.c. HMDB-51	6.5	35.3	9.3	84.5	135.6 (+0.5)
s.p.c. YUP++	9.7	52.7	13.9	126.2	202.5 (+0.8)
s.p.c. MPII	12.4	67.1	17.7	160.8	258.0 (+1.3)
s.p.c. Charades	21.0	114.2	30.0	273.5	438.7 (+2.6)
s.p.c. EPIC-Kitchens	20.3	110.4	29.0	264.3	424.0 (+2.6)

Table 4.11: Statistics of object detectors we use. We provide timings such as seconds per frame (*sec. per frame*) and seconds per clip (*s.p.c.*) for detectors used by ODF. The total time incurred by a combined detector (*ODF total*) is also provided. We also compute the time taken by the full SVD and all remaining ODF operations, assuming $\sim$5 detections per frame.

	SAL1: MNL	*SAL2: ACLNet*	SDF *total* (+Eq. (10))	ODF+SDF total (+Eq. (10)+SVD)
sec. per frame	0.60	0.30	0.90 (+0.003)	2.36 (+0.1)
s.p.c. HMDB-51	55.7	27.9	83.6 (+0.3)	219.2 (+0.8)
s.p.c. YUP++	83.2	41.6	124.8 (+0.4)	327.3 (+1.2)
s.p.c. MPII	106.0	53.0	159.0 (+0.5)	417.0 (+1.8)
s.p.c. Charades	180.3	90.1	270.4 (+0.9)	709.1 (+3.5)
s.p.c. EPIC-Kitchens	174.3	87.1	261.4 (+0.9)	685.4 (+2.9)

Table 4.12: Statistics of saliency detectors we use. We provide timings such as seconds per frame (*sec. per frame*) and seconds per clip (*s.p.c.*) for detectors used by SDF. The total time incurred by a combined detector (*SDF total*) is also provided. We also compute the time taken by the descriptor in Eq. (10) and all remaining SDF operations. Finally, we also provide the combined ODF and SDF time (*SDF+ODF total*).

[McInnes et al., 2018] on the YUP++ dataset. In Fig. 4.6a, top left corner contains samples from classes in red, green, and blue colors which partially overlap. In Fig. 4.6b, top left corner contains the samples from the corresponding classes in red, green, and blue colors. This time, the samples of these three classes are well separated from each other.

Figure 4.7 is a visualization performed with UMAP [McInnes et al., 2018] on the HMDB-51 dataset. In Fig. 4.7a, bottom left corner contains samples from classes in red and blue colors which partially overlap. In Fig. 4.7b, bottom left corner contains the samples from the corresponding classes in red and blue colors. This time, the class-wise clusters seem to be more clearly delineated and samples of these classes are separated better from each other.

4.5 Conclusions

We have introduced two simple yet effective object and saliency descriptors, which perform self-supervision of an AR hallucination-based network. We have shown that modeling high-order statistical moments can result in small representations that can self-supervise our AR pipeline. The findings are in line with recent multi-task

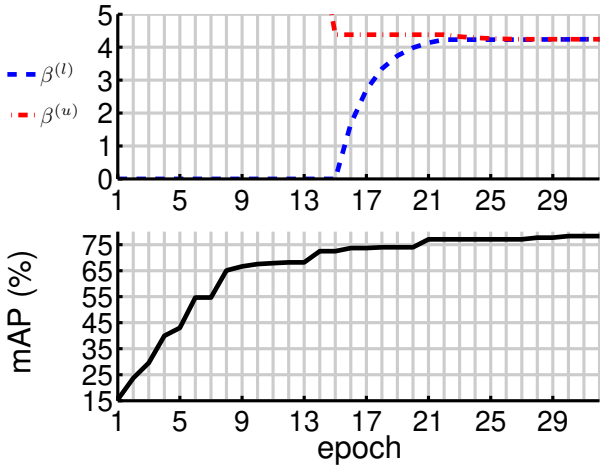


Figure 4.8: Visualization of the Golden-search for the weighting mechanism (final level weighting). (*top*) Illustration of how the lower and upper estimates $\beta^{(l)}$ and $\beta^{(u)}$ converge as epochs progress. (*bottom*) For every epoch, we set $\beta = 0.5(\beta^{(l)} + \beta^{(u)})$ and obtain the corresponding validation score (mAP) on MPII (*split1*). As the epoch number advances, mAP improves and remains stable as the Golden-search algorithm converges.

learning papers which argue that related tasks can co-supervise the main task. We are the first to hallucinate object and saliency detection descriptors with clear cut improvements in accuracy, and state-of-the-art results on the large-scale Charades and EPIC-Kitchens. More importantly, we demonstrate that hallucinating object and saliency detections is an attractive proposition even for the state-of-the-art AR backbones such as AssembleNet and AssembleNet++.

Chapter 5

Tensor Representations

Human actions in video sequences are characterized by the complex interplay between spatial features and their temporal dynamics. In this chapter, we propose novel tensor representations for compactly capturing such higher-order relationships between visual features for the task of action recognition. We propose two tensor-based feature representations, viz. (i) *sequence compatibility kernel* (SCK) and (ii) *dynamics compatibility kernel* (DCK). SCK builds on the spatio-temporal correlations between features, whereas DCK explicitly models the action dynamics of a sequence. We also explore generalization of SCK, coined SCK $\oplus$, that operates on subsequences to capture the local-global interplay of correlations, which can incorporate multi-modal inputs *e.g.*, skeleton 3D body-joints and per-frame classifier scores obtained from deep learning models trained on videos. We introduce linearization of these kernels that lead to compact and fast descriptors. We provide experiments on (i) 3D skeleton action sequences, (ii) fine-grained video sequences, and (iii) standard non-fine-grained videos. As our final representations are tensors that capture higher-order relationships of features, they relate to co-occurrences for robust fine-grained recognition [Lin and Maji, 2017; Koniusz et al., 2018b]. We use higher-order tensors and so-called Eigenvalue Power Normalization (EPN) which have been long speculated to perform spectral detection of higher-order occurrences [Koniusz et al., 2013a, 2016c], thus detecting fine-grained relationships of features rather than merely count features in action sequences. We prove that a tensor of order r, built from Z_* dimensional features, coupled with EPN indeed detects if at least one higher-order occurrence is 'projected' into one of its $\binom{Z_*}{r}$ subspaces of dim. r represented by the tensor, thus forming a Tensor Power Normalization metric endowed with $\binom{Z_*}{r}$ such 'detectors'.

5.1 Introduction

Human action recognition is a central problem in computer vision with potential impact in surveillance, human-robot interaction, elderly assistance systems, *etc.* While there have been significant advancements in this area over the past few years, action recognition in unconstrained settings still remains a challenge. Some papers simplify the problem from using RGB cameras to the use of Microsoft Kinect or the OpenPose library [Cao et al., 2017] to localize human body-parts, produce moving 3D skele-

tons [Shotton et al., 2013] and use them for recognition. However, skeletons can be noisy due to badly localized body-parts, self-occlusions, and sensor errors. Similarly, a popular strategy of classifying RGB frames into actions followed by average/max-pooling fails as only correlations of some features are informative [Mahmud et al., 2017; Cherian et al., 2017a; Cherian and Gould, 2018]. Such observations motivate the need for higher-order reasoning on 3D skeletons/frame-wise CNN classifier scores taking action recognition toward fine-grained modeling.

Recent approaches which work with skeletons can be mainly divided into two perspectives, namely (i) generative models that assume the skeleton points are produced by a latent dynamic model [Turaga and Chellappa, 2009] corrupted by noise and (ii) discriminative approaches that generate compact representations of sequences on which classifiers are trained [Presti and La Cascia, 2015]. Due to the huge configuration space of 3D actions and the unavailability of sufficient training data, discriminative approaches have been more successful. In this line of research, the main idea is to compactly represent the spatio-temporal evolution of 3D skeletons, and later train classifiers on these representations to recognize actions. Fortunately, there is a definitive structure to motions of 3D joints relative to each other due to the connectivity and length constraints of body-parts. Such constraints have been used with the Lie Algebra [Vemulapalli et al., 2014], positive definite matrices [Harandi et al., 2014; Hussein et al., 2013], torus manifold [Elgammal and Lee, 2009], Hanklet representations [Li et al., 2012], *etc*. While modeling actions with explicit manifold assumptions is useful, it is computationally costly.

However, action recognition from videos [Simonyan and Zisserman, 2014; Tran et al., 2015; Karpathy et al., 2014; Donahue et al., 2015] does not require elaborate skeletal models. A two-stream CNN framework [Simonyan and Zisserman, 2014] uses two streams to model RGB frames and optical flow. Tran *et al*. [Tran et al., 2015] use CNNs to learn spatio-temporal filters. Karpathy *et al*. [Karpathy et al., 2014] apply RGB and optical-flow fusion, whereas approach [Donahue et al., 2015] combines CNNs with LSTM to model temporal flow. Wang *et al*. [Wang et al., 2016b] apply a long-range temporal structure modeling. Tran *et al*. [Tran et al., 2018] study several forms of spatiotemporal convolutions. Recent works on fine-grained activity recognition use CNNs [Chéron et al., 2015; Ji et al., 2013] and the human pose estimation for high-level fine-grained reasoning [Rohrbach et al., 2012; Wang et al., 2013a; Zuffi and Black, 2013; Chéron et al., 2015]. Finally, the recent I3D model [Carreira and Zisserman, 2017] 'inflates' 2D CNN filters pretrained on ImageNet to spatio-temporal 3D filters yielding state-of-the-art results.

In contrast to these approaches, we present a novel representation of actions based on 3D skeleton sequences and the CNN classifier score sequences. We avoid assumptions about the data manifold by capturing higher-order statistics of the body-joints and the classifier score interactions per sequence. To this end, our scheme combines positive definite kernels and higher-order tensors, with the goal of obtaining rich and compact representations that benefit from the non-linearity of radial basis functions (RBF). Such a scheme captures higher-order data statistics [Koniusz et al., 2016c], complex action dynamics [Koniusz et al., 2016a; Cherian et al., 2017b] and

fine-grained relations [Lin and Maji, 2017; Koniusz et al., 2018b].

We present two representations for classification of 3D skeletons. Our first representation, *sequence compatibility kernel* (SCK), captures the spatio-temporal compatibility of body-joints between two sequences. To this end, we present an RBF kernel formulation that jointly captures the spatial and temporal similarity of each body-pose (normalized with respect to the hip position) in a sequence against those in another. We show that tensors generated from third-order outer-products of the linearizations of these kernels are a simple yet powerful representation capturing higher-order statistics of body-parts.

Our second representation, termed *dynamics compatibility kernel* (DCK), represents spatio-temporal dynamics of each sequence explicitly. We present a novel RBF kernel formulation that captures the similarity between a pair of body-poses in a given sequence explicitly, and then compare it against such body-pose pairs in other sequences. Such spatio-temporal modeling could be expensive due to the volumetric nature of space and time. However, we show that using an appropriate kernel model can shrink the time-related variable into a small representation of constant size after kernel linearization. With this approach, we can model both spatial and temporal variations in the form of co-occurrences which could otherwise be prohibitive. We show empirically that SCK and DCK are complementary.

As SCK/DCK work on entire sequences, we formulate an SCK-like kernel over multiple length subsequences as some of subsequences capture the gist of performed actions better than full sequences. To show the versatility of the extended SCK, we apply it to capture spatio-temporal compatibility of frame-wise CNN classifier scores from videos (regular and fine-grained actions).

We present experiments on seven standard datasets: (i) UTKinect-Action3D [Xia et al., 2012], (ii) Florence3D-Actions [Seidenari et al., 2013], (iii) MSR-Action3D [Li et al., 2010] and (iv) HMDB-51[Kuehne et al., 2011] datasets as well as two fine-grained datasets: (v) NTU RGB+D [Shahroudy et al., 2016a], (vi) MPII Cooking Activities [Rohrbach et al., 2012] and (vii) Kinetics [Kay et al., 2017]. We use the first three datasets as a source of 3D body joint sequences (as well as Kinetics), NTU for both 3D body joint sequences, and videos with RGB frames and optical flow frames, and HMDB-51 and MPII Cooking Activities for videos with RGB and optical flow frames. We show that our extensions can still achieve state-of-the-art accuracy two years after SCK/DCK were proposed [Koniusz et al., 2016a]. To summarize:

i. We design sequence and dynamics compatibility kernels that capture spatio-temporal evolution of 3D skeleton body-joints.

ii. We derive linearizations of these kernels by tensors.

iii. We extend these kernels to aggregation over multiple subsequences and CNN classifier scores.

iv. We conduct a novel theoretical analysis of Tensor Power Normalization which connects it to subspace methods. We are the first to conduct a theoretical analysis

of higher-order pooling with Tensor Power Normalization in Section 5.9, and use it for generic/fine-grained action recognition.

5.2 Related Work

In the first part of our chapter, we focus on action recognition from an articulated set of connected body-joints that evolve in time [Zatsiorsky, 1997]. A temporal evolution of the human skeleton is very informative for action recognition as shown by Johansson in his seminal experiment involving the moving lights display [Johansson, 1973]. At the simplest level, the human body can be represented as a set of 3D points corresponding to body-joints such as elbow, wrist, knee, ankle, *etc*. Action dynamics has been modeled using the motion of such 3D points in [Hussein et al., 2013; Lv and Nevatia, 2006], using joint orientations with respect to a reference axis [Parameswaran and Chellappa, 2006] and even relative body-joint positions [Wang et al., 2012; Yang and Tian, 2014a]. In contrast, we represent these 3D body-joints by kernels whose linearization results in higher-order tensors capturing complex statistics. We also note parts-based approaches that use connected body segments [Yacoob and Black, 1998; Ohn-Bar and Trivedi, 2013; Ofli et al., 2014; Vemulapalli et al., 2014]. For details, see a survey [Presti and La Cascia, 2015].

We also handle the temporal domain differently to other methods. 3D joint locations are modeled as temporal hierarchy of coefficients in [Hussein et al., 2013]. Pairwise relative positions of joints were modeled in [Wang et al., 2012] and combined with a hierarchy of Fourier coefficients to capture temporal evolution of actions. In [Yang and Tian, 2014a], the relative joint positions and their temporal displacements are modeled with respect to the initial frame. In [Vemulapalli et al., 2014], the displacements and angles between the body parts are represented as a collection of matrices belonging to SE(3), a special Euclidean group. The temporal domain is handled by the dynamic time warping and Fourier temporal pyramid matching. In contrast, we avoid expensive time warping by modeling the temporal domain with an RBF kernel invariant to local temporal shifts.

Our scheme also differs from works such as kernel descriptors [Bo et al., 2011] that sum gradient orientations over image patches, action recognition via kernelized covariances [Wang et al., 2015a; Cavazza et al., 2016; Zhang et al., 2020b], and a time series kernel [Gaidon et al., 2011] which extracts spatio-temporal autocorrelations. In contrast, our scheme sums over several multiplicative and additive RBF kernels. We capture higher-order statistics by linearizing a polynomial kernel and avoid evaluating costly kernels directly.

Third-order tensors have been used to form spatio-temporal tensors on videos in [Kim et al., 2007]. Non-negative tensor factorization is used for image denoising [Shashua and Hazan, 2005], tensors are used for texture rendering [Vasilescu and Terzopoulos, 2004] and for face recognition [Vasilescu and Terzopoulos, 2002]. A survey of multi-linear algebraic methods for tensor subspace learning is available in [Lu et al., 2011]. These methods use a single tensor, whereas we use tensors as

descriptors [Koniusz et al., 2013a, 2016c; Koniusz and Cherian, 2016; Zhao et al., 2012]. However, we use third-order tensors for action recognition, which poses a set of new challenges.

For fine-grained action recognition, high-level sophisticated action reasoning [Wang et al., 2013a; Rohrbach et al., 2012; Zuffi and Black, 2013; Chéron et al., 2015] is typically used together with pose estimation systems [Wei et al., 2016; Insafutdinov et al., 2016]. However, these approaches scale poorly to millions of video frames. Human-object interactions in the videos are analyzed in [Zhou et al., 2015]. Correlations between space-time features are proposed in [Shechtman and Irani, 2005].

Power Normalization approaches [Koniusz et al., 2013a, 2016c; Koniusz and Cherian, 2016; Koniusz et al., 2018b; Koniusz and Zhang, 2020] speculate that Eigenvalue Power Normalization prevents so-called burstiness, thus performing spectral detection of higher-order occurrences of features [Koniusz et al., 2013a, 2016c], which can be paraphrased as *'do a knife, a hand and a chopping board co-occur together?'* rather than *'how many knifes, hands and chopping boards appear in the scene?'*

Moreover, first-order pooling was successfully used for representing action recognition via hallucination [Wang et al., 2019d]. Papers [Koniusz et al., 2018b; Koniusz and Zhang, 2020] study second-order pooling, power normalizing functions and their taxonomy while fast pooling methods are proposed in [Lin and Maji, 2017; Koniusz and Zhang, 2020; Lin et al., 2018].

Finally, second-order pooling was successfully used for few-shot action recognition [Zhang et al., 2020a], few-shot classification [Zhang and Koniusz, 2019; Simon et al., 2020a], few-shot segmentation [Zhang et al., 2020d], modulating optimization [Simon et al., 2020b], style transfer [Shiri et al., 2017, 2018, 2019b,a] and action self-supervision [Wang and Koniusz, 2021]. Noteworthy are also graph convolutional networks [Yan et al., 2018; Sun et al., 2019; Zhu and Koniusz, 2021b] and embeddings [Zhu and Koniusz, 2021a] easily applicable to 3D skeleton action recognition.

5.3 Preliminaries

In this section, we review our notations and the necessary background on shift-invariant kernels and their linearizations.

5.3.1 Tensor Notations

Figure 5.1a illustrates the notion of tensors, their order and modes. Let $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ denote a third-order tensor. Using the Matlab notation, we refer to the k-th slice of this tensor as $\boldsymbol{\mathcal{V}}_{:,:,k}$, which is a $d_1 \times d_2$ matrix. For a matrix $\mathbf{V} \in \mathbb{R}^{d_1 \times d_2}$ and a vector $\mathbf{v} \in \mathbb{R}^{d_3}$, the notation $\boldsymbol{\mathcal{V}} = \mathbf{V} \uparrow \otimes \mathbf{v}$ produces a tensor $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ whose k-th slice is given by $\mathbf{V} \cdot v_k$, v_k being the k-th coefficient of $\mathbf{v}$. Figure 5.1b illustrates such an outer-product. Symmetric third-order tensors of rank one are formed by the outer-product of a vector $\mathbf{v} \in \mathbb{R}^d$ in three modes, that is, a rank-one $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d \times d \times d}$ is obtained from $\mathbf{v}$ as $\boldsymbol{\mathcal{V}} = (\uparrow \otimes_3 \mathbf{v} \triangleq (\mathbf{v}\mathbf{v}^T) \uparrow \otimes \mathbf{v})$ which yields $\mathcal{V}_{ijk} = v_i \cdot v_j \cdot v_k$, where $\mathcal{V}_{ijk}$ represents the ijk-th element of $\boldsymbol{\mathcal{V}}$. Matrices have two modes: the first and second

mode correspond to the row and column indexes i and j, respectively. Order r tensors have r modes addressed by $\mathcal{V}_{i_1 \dots i_r}$ where $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d_1 \times \dots \times d_k \times \dots \times d_r}$ and k indicates the mode k. Concatenation of n tensors in mode k is simply stacking them along mode k, denoted as $[\boldsymbol{\mathcal{V}}_i]_{i \in \mathcal{I}_n}^{\oplus_k} \equiv \text{numpy.concatenate}((\boldsymbol{\mathcal{V}}_1, \dots, \boldsymbol{\mathcal{V}}_n), \text{axis} = k-1))$. $\mathcal{I}_n$ is an index sequence $1, 2, \dots, n$. We define the Frobenius norm $\|\boldsymbol{\mathcal{V}}\|_F = \sqrt{\sum_{i,j,k} \mathcal{V}_{ijk}^2}$ and the inner-product between $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ as $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{ijk} \mathcal{X}_{ijk} \mathcal{Y}_{ijk}$. Also, $\boldsymbol{e}_z$ are spanning bases of $\mathbb{R}^Z$. Further basics on tensors and tensor algebra can be found in [Huckle, 2019].

5.3.2 Kernel Linearization

Let $G_\sigma(\mathbf{u} - \bar{\mathbf{u}}) = \exp(-\|\mathbf{u} - \bar{\mathbf{u}}\|_2^2 / 2\sigma^2)$ denote a standard Gaussian RBF kernel centered at $\bar{\mathbf{u}}$ and having a bandwidth σ. Kernel linearization refers to rewriting this G_σ as an inner-product of two infinite-dimensional feature maps. To obtain these maps, we use a fast approximation method based on probability product kernels [Jebara et al., 2004]. Specifically, we employ the inner product of d'-dimensional isotropic Gaussians given $u, u' \in \mathbb{R}^{d'}$. Thus, we have:

$$G_\sigma(\mathbf{u} - \bar{\mathbf{u}}) = \left(\frac{2}{\pi \sigma^2} \right)^{\frac{d'}{2}} \int_{\zeta \in \mathbb{R}^{d'}} G_{\sigma/\sqrt{2}}(\mathbf{u} - \zeta)\, G_{\sigma/\sqrt{2}}(\bar{\mathbf{u}} - \zeta)\, \mathrm{d}\zeta. \tag{5.1}$$

Eq. (5.1) is then approximated by replacing the integral with the sum over Z pivots $\zeta_1, \dots, \zeta_Z$. Thus, we obtain a feature map $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}(\mathbf{u}; \{\zeta_i\}_{i \in \mathcal{I}_Z}) = \left[G_{\sigma/\sqrt{2}}(\mathbf{u} - \zeta_1), \dots, G_{\sigma/\sqrt{2}}(\mathbf{u} - \zeta_Z) \right]^T, \tag{5.2}$$

$$\text{and } G_\sigma(\mathbf{u} - \bar{\mathbf{u}}) \approx \left\langle \sqrt{c}\boldsymbol{\phi}(\mathbf{u}), \sqrt{c}\boldsymbol{\phi}(\bar{\mathbf{u}}) \right\rangle, \tag{5.3}$$

where c is a const. Eq. (5.3) is the linearization of the RBF kernel. Eq. (5.2) is the feature map. $\{\zeta_i\}_{i \in \mathcal{I}_Z}$ are pivots. As we use 1 dim. signals, we simply cover interval $[-1; 1]$ (or $[0; 1]$) with Z equally spaced pivots. For clarity, we drop $\{\zeta_i\}_{i \in \mathcal{I}_Z}$ and write $\boldsymbol{\phi}(\mathbf{u})$, *etc.*

5.3.3 Equivalence between Polynomial Kernels and the Dot-product of Tensors

For any two Z' dim. feature vectors $\boldsymbol{\phi}, \bar{\boldsymbol{\phi}} \in \mathbb{R}^{Z'}$, we have:

$$\langle \boldsymbol{\phi}, \bar{\boldsymbol{\phi}} \rangle^r = \sum_{i_1=1}^{Z'} \dots \sum_{i_r=1}^{Z'} \phi_{i_1} \bar{\phi}_{i_1} \cdot \dots \cdot \phi_{i_r} \bar{\phi}_{i_r} = \langle \uparrow \otimes_r \boldsymbol{\phi}, \uparrow \otimes_r \bar{\boldsymbol{\phi}} \rangle, \tag{5.4}$$

where $\boldsymbol{\mathcal{X}} = (\uparrow \otimes_r \boldsymbol{\phi})$ is defined as $\mathcal{X}_{i_1 \dots i_r} = \phi_{i_1} \cdot \dots \cdot \phi_{i_r}$.

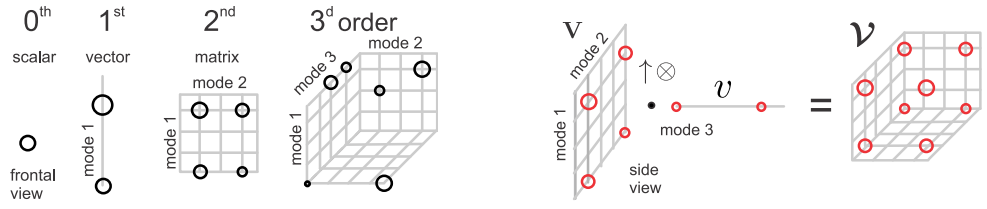


Figure 5.1: Figure 5.1a illustrates the notion of tensors, their order and modes. Figure 5.1b illustrates the matrix-vector order outer-product.

5.4 Proposed Approach

Below, we formulate the problem of action recognition from 3D skeleton sequences, which precedes an exposition of our two kernel formulations for describing actions, followed by their tensor reformulations through kernel linearization. We also introduce Eigenvalue Power Normalization and our improved kernels used for action recognition based on skeletons and/or classifier scores obtained from videos passed via CNNs.

5.4.1 Statistical Motivation

Before we outline our higher-order tensor representations, below we motivate the use of higher-order statistics. To compare skeleton sequences/videos, we want to capture distribution of local features/descriptors per sequence *e.g.*, body joints or receptive fields in CNN. The characteristic function $\varphi_\Phi(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\phi}\sim\Phi}\left(\exp(i\boldsymbol{\omega}^T\boldsymbol{\phi})\right)$ describes the probability density $f_\Phi(\boldsymbol{\phi})$ of a skeleton sequence/video (local features/descriptors $\boldsymbol{\phi}\sim\Phi$).

Taylor expansion of the characteristic function per sequence is:

$$\mathbb{E}_{\boldsymbol{\phi}\sim\Phi}\Big(\sum_{r=0}^{\infty}\frac{i^r}{r!}\left\langle\boldsymbol{\phi},\boldsymbol{\omega}\right\rangle^r\Big)\approx\frac{1}{N}\sum_{n=0}^{N}\sum_{r=0}^{\infty}\frac{i^r}{r!}\left\langle\uparrow\otimes_r\boldsymbol{\phi}_n,\uparrow\otimes_r\boldsymbol{\omega}\right\rangle \tag{5.5}$$
$$=\sum_{r=0}^{\infty}\frac{i^r}{r!}\Big\langle\frac{1}{N}\sum_{n=0}^{N}\uparrow\otimes_r\boldsymbol{\phi}_n,\uparrow\otimes_r\boldsymbol{\omega}\Big\rangle=\sum_{r=0}^{\infty}\Big\langle\boldsymbol{\mathcal{X}}^{(r)},\frac{i^r}{r!}\uparrow\otimes_r\boldsymbol{\omega}\Big\rangle.$$

Symbol $\boldsymbol{\mathcal{X}}^{(r)}=\frac{1}{N}\sum_{n=0}^{N}\uparrow\otimes_r\boldsymbol{\phi}_n$ defines a tensor descriptor while i is the imaginary number. In principle, with infinite data and infinite moments, one can fully capture $f_\Phi(\boldsymbol{\phi})$ which is intractable. In practice, third-order moments work well in what follows while second-order moments are somewhat insufficient.

5.4.2 Problem Formulation

Suppose we are given a set of 3D human pose skeleton sequences, each pose consisting of J body-keypoints. Further, to simplify our notations, we assume each sequence

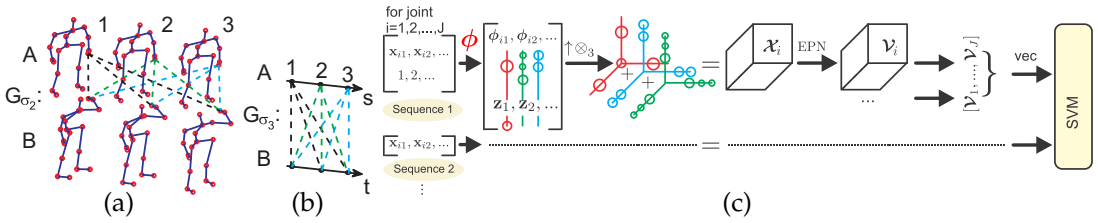


Figure 5.2: Figures 5.2a and 5.2b show how SCK works – kernel G_{σ_2} compares exhaustively *e.g.* hand-related joint i for every frame in sequence A with every frame in sequence B. Kernel G_{σ_3} compares exhaustively the frame indexes. Figure 5.2c shows this burden is avoided by linearization – third-order statistics on feature maps $\boldsymbol{\phi}(\mathbf{x}_{is})$ and $\mathbf{z}(s/N)$ for joint i are captured in tensor $\boldsymbol{\mathcal{X}}_i$ and whitened by EPN to obtain $\boldsymbol{\mathcal{Y}}_i$ which are concatenated over $i=1,...,J$ to represent a sequence. The final sequence tensors are vectorized per video by 'vec' and fed to an SVM.

consists of N skeletons, one per frame[1]. We define such a pose sequence Π as:

$$\Pi = \left\{\mathbf{x}_{is} \in \mathbb{R}^3, i \in \mathcal{I}_J, s \in \mathcal{I}_N\right\}. \tag{5.6}$$

Further, let each such a sequence Π be associated with one of K action class labels $\ell \in \mathcal{I}_K$. Our goal is to use the skeleton sequence Π and generate an action descriptor for this sequence that can be used in a classifier for recognizing the action class. In what follows, we will present two such action descriptors, namely (i) sequence compatibility kernel and (ii) dynamics compatibility kernel, which are formulated using kernel linearization and tensor algebra theories. We present both these kernel formulations next.

5.4.3 Sequence Compatibility Kernel

As alluded to earlier, the main idea of this kernel is to measure the compatibility between two action sequences in terms of the similarity between their skeletons and their temporal order. To this end, we assume each skeleton is centered with respect to one of the body-joints (say, hip). Suppose we are given two such sequences Π_A and Π_B, each with J joints, and N frames. Further, let $\mathbf{x}_{is} \in \mathbb{R}^3$ and $\mathbf{y}_{jt} \in \mathbb{R}^3$ correspond to the body-joint coordinates of Π_A and Π_B, respectively.

We define our *sequence compatibility kernel* (SCK) between Π_A and Π_B as[1]:

$$K_S(\Pi_A, \Pi_B) = \frac{1}{\Lambda} \sum_{(i,s)\in\mathcal{J}} \sum_{(j,t)\in\mathcal{J}} G_{\sigma_1}(i-j)\left(\beta_1 G_{\sigma_2}(\mathbf{x}_{is}-\mathbf{y}_{jt}) + \beta_2\, G_{\sigma_3}\left(\frac{s-t}{N}\right)\right)^r. \tag{5.7}$$

Symbol Λ is a normalization constant and $\mathcal{J} = \mathcal{I}_J \times \mathcal{I}_N$. As is clear, this kernel involves three different compatibility subkernels, namely (i) G_{σ_1}, capturing the com-

[1]We assume that all sequences have N frames for simplification of presentation. Our formulations are applicable to sequences of arbitrary lengths *e.g.*, M and N. Thus, we apply in practice $G_{\sigma_3}(\frac{s}{M} - \frac{t}{N})$ in Eq. (5.7).

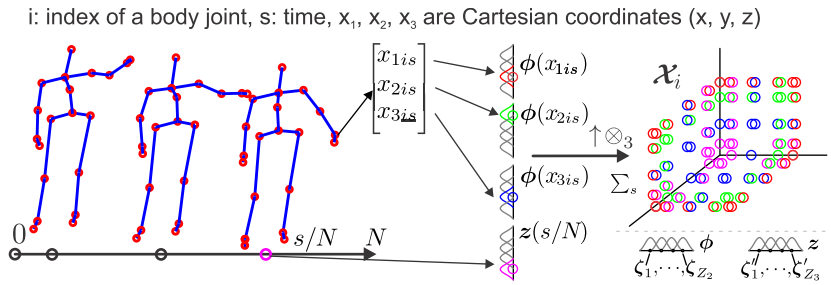


Figure 5.3: Order r statistics from Eq. (5.7) can be understood by studying the linearization in Eq. (5.10). For a given joint i at time s/N (normalized frame number), we embed a 3D joint coordinate $\mathbf{x}_{is}$ (all centered w.r.t. hip) via function $\boldsymbol{\phi}(\cdot)$ into a non-linear Hilbert space representing an RBF kernel according to Eq. (5.2). Similarly, we embed the time s/N via function $\mathbf{z}(\cdot)$ (also by Eq. (5.2)). Finally, $\otimes_r$ performs the third-order outer-product on concatenated embeddings aggregated next over frames s (note $\sum_s$). The interpretation: the Gaussians 'soft-divide' the the Cartesian coordinate system along x, y, z direction, resp., and time s/N. Thus, triplets (x,y,z), $(x,y,s/N)$, $(x,z,s/N)$ and $(y,z,s/N)$ assigned into such a 'soft-divided' space capture locally three-way occurrences. They factor out one spatial (or time) variable at a time (note invariance to such a variable).

patibility between joint-types i and j, (ii) G_{σ_2}, capturing the compatibility between joint locations $\mathbf{x}$ and $\mathbf{y}$, and (iii) G_{σ_3}, measuring the temporal alignment of two poses in two sequences. We also introduce weighting factors $\beta_1, \beta_2 \geq 0$ that adjust the importance of the body-joint compatibility against the temporal alignment, where $\beta_1 + \beta_2 = 1$. Figures 5.2a and 5.2b illustrate how this kernel works. It might come as a surprise that we use kernel G_{σ_1}. Note that our skeletons may be noisy and there is a possibility that some keypoints are detected incorrectly (for example, elbows and wrists). Thus, this kernel allows incorporating a degree of uncertainty into the alignment of such joints. To simplify our formulation, in this chapter, we will assume that such errors are absent from our skeletons, and thus $G_{\sigma_1}(i-j) = \delta(i-j)$. Furthermore, standard deviations σ_2 and σ_3 control the joint-coordinate selectivity and temporal shift-invariance, respectively. That is, for $\sigma_3 \to 0$, two sequences will have to match perfectly in the temporal sense. For $\sigma_3 \to \infty$, the algorithm is invariant to any permutations of the frames. As will be clear in the sequel, parameter r determines the order of statistics of our kernel (we use $r = 3$).

Next, we present linearization of our kernel using the method from Sections 5.3.2, 5.3.3, and Eq. (5.3), so that kernel $G_{\sigma_2}(\mathbf{x}-\mathbf{y}) \approx \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$ (see note[2]) while $G_{\sigma_3}(\frac{s-t}{N}) \approx \mathbf{z}(s/N)^T\mathbf{z}(t/N)$ (see note[3]). With these approximations and simplification to G_{σ_1}

[2]In practice, Cartesian coordinates of joints $\mathbf{x},\mathbf{y} \in \mathbb{R}^3$ are fed into a kernel. Thus, in place of kernel G_{σ_2}, we use the sum kernel $G'_{\sigma_2}(\mathbf{x}-\mathbf{y}) = G_{\sigma_2}(x_1-y_1)+G_{\sigma_2}(x_2-y_2)+G_{\sigma_2}(x_3-y_3)$ whose approximation is given as: $G'_{\sigma_2}(\mathbf{x}-\mathbf{y}) \approx [\boldsymbol{\phi}(x_1;\{\zeta_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(x_2;\{\zeta_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(x_3;\{\zeta_i\}_{i\in\mathcal{I}_{Z_2}})]^T[\boldsymbol{\phi}(y_1;\{\zeta_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(y_2;\{\zeta_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(y_3;\{\zeta_i\}_{i\in\mathcal{I}_{Z_2}})]$ but for simplicity we refer to it in our formulations by its generic form $G_{\sigma_2}(\mathbf{x}-\mathbf{y}) \approx \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$ because we can define $\phi(\mathbf{x}) = [\phi(x_1);\phi(x_2);\phi(x_3)]$.

[3]Feature maps $\mathbf{z}(\cdot) \equiv \boldsymbol{\phi}(\cdot)$ from Eq. (5.2). We simply write $\mathbf{z}$ rather than $\boldsymbol{\phi}$ to denote these feat. maps as they encode the time/frame number (*c.f.* the body joints). Note that $\mathbf{z}(\cdot;\{\zeta'_i\}_{i\in\mathcal{I}_{Z_3}})$ uses Z_3 pivots

described above, we rewrite our sequence compatibility kernel as:

$$K_S(\Pi_A,\Pi_B)\approx$$

$$\frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{s\in\mathcal{I}_N}\sum_{t\in\mathcal{I}_N}\left(\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is}),\ \text{(see note}^{2}\text{)}\\ \sqrt{\beta_2}\,\mathbf{z}(s/N),\ \text{(see note}^{3}\text{)}\end{bmatrix}^T\cdot\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right)^r \tag{5.8}$$

$$=\frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{s\in\mathcal{I}_N}\sum_{t\in\mathcal{I}_N}\left\langle\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\,\mathbf{z}(s/N)\end{bmatrix},\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right\rangle \tag{5.9}$$

$$=\sum_{i\in\mathcal{I}_J}\left\langle\frac{1}{\sqrt{\Lambda}}\sum_{s\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\mathbf{z}(s/N)\end{bmatrix},\frac{1}{\sqrt{\Lambda}}\sum_{t\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right\rangle. \tag{5.10}$$

Expansion of Eq. (5.8) into Eq. (5.9) simply follows the notion of equivalence between the polynomial kernels and tensor outer-products as detailed in Eq. (5.4). Similarly, the summations in Eq. (5.9) can be absorbed into the dot-product in Eq. (5.10) because the inner-product is a linear operation in each of its arguments *e.g.*, $\langle\mathbf{v}_1+\mathbf{v}_2,\bar{\mathbf{v}}\rangle=\langle\mathbf{v}_1,\bar{\mathbf{v}}\rangle+\langle\mathbf{v}_2,\bar{\mathbf{v}}\rangle$. The physical meaning of the above equation is detailed in Figure 5.3. While the first-, second- and third-order outer-products are connected to the sample mean, covariance and co-skewness of features, our tensors are not mere counts of features, as explained next. As is clear, (5.10) expresses $K_S(\Pi_A,\Pi_B)$ as a sum of inner-products on third-order tensors ($r=3$), as shown in Figure 5.2c. While, using the dot-product as the inner-product is an option, other alternatives for tensors of order $r\geq 2$ can act on their spectrum, leading to better representations. An example is the so-called *burstiness* [Jégou et al., 2009], which is a commonly encountered property that a given feature appears more/less often in a sequence than a statistically independent model predicts. Robust descriptors must be invariant w.r.t. the length of actions *e.g.*, a prolonged *hand waving* represents the same action as a short *hand wave*. Eigenvalue Power Normalization (EPN) [Koniusz et al., 2016c] suppresses burstiness by acting on higher-order statistics (see Fig. 5.2c). By incorporating EPN, we generalize (5.10) as:

$$K^*_S(\Pi_A,\Pi_B)=\sum_{i\in\mathcal{I}_J}\left\langle\boldsymbol{\mathcal{G}}\left(\frac{1}{\sqrt{\Lambda}}\sum_{s\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\mathbf{z}(s/N)\end{bmatrix}\right),\right.$$
$$\left.\boldsymbol{\mathcal{G}}\left(\frac{1}{\sqrt{\Lambda}}\sum_{t\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right)\right\rangle, \tag{5.11}$$

where the operator $\boldsymbol{\mathcal{G}}$ performs EPN by applying power normalization to the spectrum of the third-order tensor (by taking the higher-order SVD). Note that in general $K^*_S(\Pi_A,\Pi_B)\not\approx K_S(\Pi_A,\Pi_B)$ as $\boldsymbol{\mathcal{G}}$ is intended to manipulate the spectrum of $\boldsymbol{\mathcal{X}}$.

$\{\zeta'_i\}_{i\in\mathcal{I}_{Z_3}}$ (see Figure 5.3).

The final representation for linearized SCK becomes:

$$\boldsymbol{\mathcal{V}}_i = \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}_i), \text{ where } \boldsymbol{\mathcal{X}}_i = \frac{1}{\sqrt{\Lambda}} \sum_{s \in \mathcal{I}_N} \uparrow \otimes_r \begin{bmatrix} \sqrt{\beta_1}\, \boldsymbol{\phi}(\mathbf{x}_{is}) \\ \sqrt{\beta_2}\, \mathbf{z}(s/N) \end{bmatrix}. \tag{5.12}$$

We replace the sum over the body-joint indexes in (5.11) by concatenating $\boldsymbol{\mathcal{V}}_i$ in (5.12) along the fourth tensor mode, thus defining $\boldsymbol{\mathcal{V}} = \left[\boldsymbol{\mathcal{V}}_i\right]_{i \in \mathcal{I}_J}^{\oplus_4}$. Suppose $\boldsymbol{\mathcal{V}}_A$ and $\boldsymbol{\mathcal{V}}_B$ are the corresponding fourth order tensors for Π_A and Π_B respectively. Then, we obtain:

$$K_S^*(\Pi_A, \Pi_B) = \langle \boldsymbol{\mathcal{V}}_A, \boldsymbol{\mathcal{V}}_B \rangle . \tag{5.13}$$

Note that tensors $\boldsymbol{\mathcal{X}}$ have the following properties: (i) super-symmetry $\boldsymbol{\mathcal{X}}_{i,j,k} = \boldsymbol{\mathcal{X}}_{\pi(i,j,k)}$ for indexes i, j, k and their permutation given by π, $\forall \pi$, and (ii) positive semi-definiteness of every slice, that is, $\boldsymbol{\mathcal{X}}_{:,:,s} \in \mathcal{S}_+^d$, for $s \in \mathcal{I}_d$. Thus, we use only the upper-simplices of $\boldsymbol{\mathcal{V}}_i$ which consist of $\binom{d+r-1}{r}$ coefficients (which is the total size of our final representation times the number of body-joints) rather than d^r, where d is the side-dimension of $\boldsymbol{\mathcal{V}}_i$ *i.e.*, $d = 3Z_2 + Z_3$ (see notes[2,3]), and Z_2 and Z_3 are the numbers of pivots used in the approximation of G_{σ_2} and G_{σ_3} (see notes[2,3]).

Next, we pass tensors $\boldsymbol{\mathcal{X}}$ via (i) slice-wise EPN (sEPN) operator or (ii) HOSVD-based tensor whitening EPN (tEPN) [Koniusz et al., 2016c]. sEPN is faster but tEPN uses the entire tensor spectrum, thus being more accurate. The slice-wise EPN uses the Power-Euclidean dist. for rising matrices, slices of tensor tensor $\boldsymbol{\mathcal{X}}$, to the power of γ. Power norm. and re-stacking slices along the third mode yields:

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) = [\boldsymbol{\mathcal{X}}_{:,:,s}^{\gamma}]_{s \in \mathcal{I}_d}^{\oplus_3}, \text{ for } 0 < \gamma \leq 1. \tag{5.14}$$

We note that $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$ preserves listed earlier properties of tensors $\boldsymbol{\mathcal{X}}$ and it forms our final tensors $\boldsymbol{\mathcal{V}}$ for the action sequence.

The HOSVD-based tensor whitening EPN, proposed in [Koniusz et al., 2016c], is defined by the following operator $\boldsymbol{\mathcal{G}}$:

$$(\boldsymbol{\mathcal{E}}; A_1, ..., A_r) = \text{HOSVD}(\boldsymbol{\mathcal{X}}), \tag{5.15}$$

$$\hat{\boldsymbol{\mathcal{E}}} = \text{Sgn}(\boldsymbol{\mathcal{E}})\, |\boldsymbol{\mathcal{E}}|^{\gamma}, \qquad \left(\text{generally } \hat{\boldsymbol{\mathcal{E}}} = \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{E}})\right) \tag{5.16}$$

$$\hat{\boldsymbol{\mathcal{V}}} = ((\hat{\boldsymbol{\mathcal{E}}} \times_1 A_1) ...) \times_r A_r, \quad \left(\text{think } \hat{\boldsymbol{\mathcal{V}}} = \boldsymbol{\mathcal{X}}^{\frac{1}{2}}\right) \tag{5.17}$$

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) = \text{Sgn}(\hat{\boldsymbol{\mathcal{V}}})\, |\hat{\boldsymbol{\mathcal{V}}}|^{\gamma^*} \tag{5.18}$$

In the above equations, we distinguish the core tensor $\boldsymbol{\mathcal{E}}$, its power-normalized variant $\hat{\boldsymbol{\mathcal{E}}}$ with factor weights evened out by rising them to the power $0 < \gamma \leq 1$, singular vector matrices $A_1, ..., A_r$ and operation $\times_r$ which is the so-called tensor-product in mode r.

As our tensors $\boldsymbol{\mathcal{X}}$ are super-symmetric, we note that $A_1 = A_2 = ... = A_r$. However, the kernel which is proposed in Section 5.4.4 leads to a non-symmetric tensor representation. We refer the reader to paper [Koniusz et al., 2016c] for the detailed

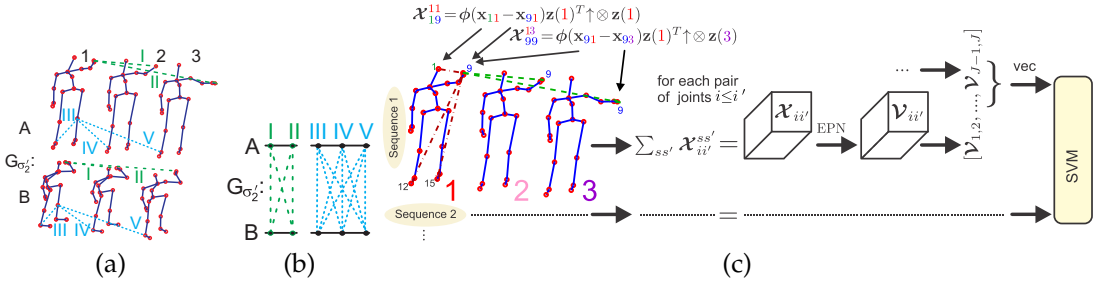


Figure 5.4: Figure 5.4a shows that kernel $G_{\sigma_2'}$ in DCK captures spatio-temporal dynamics by measuring displacement vectors from any given body-joint to remaining joints spatially- and temporally-wise (*i.e.* see dashed lines). Figure 5.4b shows that comparisons performed by $G_{\sigma_2'}$ for any selected two joints are performed all-against-all temporally-wise which is computationally expensive. Figure 5.4c shows the encoding steps in the proposed linearization which is fastn. We collect all $\boldsymbol{\mathcal{X}}_{ii'}$ for joints $i \leq i'$, whiten them by EPN to obtain $\boldsymbol{\mathcal{V}}_{ii'}$, concatenate, vectorize them per video with 'vec' and fed to an SVM. We introduced color-coded body joints/frame numbers to show how we assemble a single $\boldsymbol{\mathcal{X}}_{ii'}$.

description of the above steps.

Eq. (5.16) has a more general form $\hat{\boldsymbol{\mathcal{E}}} = \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{E}})$, where $\hat{\boldsymbol{\mathcal{G}}}$ can be any power normalizing function [Koniusz et al., 2018b]. In Sec. 5.9, we derive the exact interpretation of Eq. (5.15-5.18) for $\hat{\boldsymbol{\mathcal{G}}} = \text{Sgn}(\boldsymbol{\mathcal{E}})\,(1-(1-|\boldsymbol{\mathcal{E}}|)^{\bar{N}})$ for which $\text{Sgn}(\boldsymbol{\mathcal{E}})\,|\boldsymbol{\mathcal{E}}|^{\gamma}$ is an approximation [Koniusz et al., 2018b]. We prove in Sec. 5.9 that EPN performs in fact a spectral detection of higher-order occurrences of features, the base of fine-grained systems [Lin and Maji, 2017; Koniusz et al., 2018b]. Figure 5.9 illustrates details of such a spectral detection.

5.4.4 Dynamics Compatibility Kernel

The SCK kernel that we described above captures the inter-sequence alignment, whereas the intra-sequence spatio-temporal dynamics is lost. Thus, we propose a novel *dynamics compatibility kernel* (DCK). In what follows, we use the absolute coordinates of the joints in our kernel and follow notations from the prev. section.

DCK for two action sequences Π_A and Π_B is defined as:

$$K_D(\Pi_A,\Pi_B) = \frac{1}{\Lambda}\sum_{\substack{(i,s)\in\mathcal{J},\\(i',s')\in\mathcal{J},\\ i'\neq i, s'\neq s}}\sum_{\substack{(j,t)\in\mathcal{J},\\(j',t')\in\mathcal{J},\\ j'\neq j, t'\neq t}} G'_{\sigma_1'}(i-j,i'-j')\,G_{\sigma_2'}\big((\mathbf{x}_{is}-\mathbf{x}_{i's'})-(\mathbf{y}_{jt}-\mathbf{y}_{j't'})\big)\cdot G'_{\sigma_3'}\Big(\frac{s-t}{N},\frac{s'-t'}{N}\Big)\,G'_{\sigma_4'}(s-s',t-t'). \tag{5.19}$$

In contrast to SCK in (5.7), the DCK kernel uses the intra-sequence joint differences, thus capturing the dynamics, which is then compared against dynamics of other sequences.

Figures 5.4a-5.4c depict schematically how DCK captures co-occurrences. As in SCK, the first kernel, $G'_{\sigma'_1}$, captures the sensor uncertainty in body-keypoint detection, and is assumed to be a delta function in this chapter. The second kernel, $G_{\sigma'_2}$, models the spatio-temporal co-occurrences of the body-joints. Temporal alignment kernels, expressed as $G'_{\sigma'_3}(\boldsymbol{\alpha},\boldsymbol{\beta}) = G_{\sigma'_3}(\boldsymbol{\alpha})G_{\sigma'_3}(\boldsymbol{\beta})$, encode temporal start- and end-points from (s,s') and (t,t'). Finally, $G_{\sigma'_4}$ limits contributions of dynamics between temporal points if they are distant from each other, *i.e.* if $s' \gg s$ or $t' \gg t$ and σ'_4 is small. Similarly to SCK, the standard deviations σ'_2 and σ'_3 control the selectivity over spatio-temporal dynamics of body-joints and their temporal shift-invariance for the start and end points, resp. As discussed for SCK, the practical extensions from footnotes[1,2,3] also apply to DCK *e.g.*, the definition of $\mathbf{z}$, the pivot numbers Z_2 and Z_3 for $G_{\sigma'_2}$ and $G_{\sigma'_3}$ kernels.

Based on the above formulations, Section 5.6 shows that the linearization of DCK admits the form:

$$K_D(\Pi_A,\Pi_B) \approx \tag{5.20}$$
$$\sum_{\substack{i\in\mathcal{I}_J,\\ i'\in\mathcal{I}_J:\\ i'\neq i}} \left\langle \frac{1}{\sqrt{\Lambda_s}} \sum_{\substack{s\in\mathcal{I}_N,\\ s'\in\mathcal{I}_N:\\ s'\neq s}} G_{\sigma'_4}(s-s') \left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}\left(\frac{s}{N}\right)^T\right)\uparrow\otimes\mathbf{z}\left(\frac{s'}{N}\right), \right.$$
$$\left. \frac{1}{\sqrt{\Lambda_t}} \sum_{\substack{t\in\mathcal{I}_N,\\ t'\in\mathcal{I}_N:\\ t'\neq t}} G_{\sigma'_4}(t-t') \left(\boldsymbol{\phi}(\mathbf{y}_{it}-\mathbf{y}_{i't'})\cdot\mathbf{z}\left(\frac{t}{N}\right)^T\right)\uparrow\otimes\mathbf{z}\left(\frac{t'}{N}\right) \right\rangle.$$

Equation (5.20) expresses $K_D(\Pi_A,\Pi_B)$ as a sum over inner-products on third-order non-symmetric tensors (c.f. Section 5.4.3 where the proposed kernel results in an inner-product between super-symmetric tensors). However, we can decompose each of these tensors with a variant of EPN, which involves Higher Order Singular Value Decomposition (HOSVD), into factors stored in the so-called core tensor, and equalize the contributions of these factors to prevent bursts in the spatio-temporal co-occurrence dynamics of actions. For example, consider that a long *hand wave* versus a short *hand wave* yield different temporal statistics, that is, the prolonged action results in bursts. However, the final representation described below becomes invariant to bursts.

The final representation for linearized DCK with a non-linear operator $\mathcal{G}$ introduced into Eq. (5.20) to prevent burstiness becomes:

$$\boldsymbol{\mathcal{V}}_{ii'} = \mathcal{G}(\boldsymbol{\mathcal{X}}_{ii'}), \text{ where} \tag{5.21}$$
$$\boldsymbol{\mathcal{X}}_{ii'} = \frac{1}{\sqrt{\Lambda}} \sum_{s,s'\in\mathcal{I}_N:s'\neq s} G_{\sigma'_4}(s-s') \left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}\left(\frac{s}{N}\right)^T\right)\uparrow\otimes\mathbf{z}\left(\frac{s'}{N}\right).$$

The summation over pairs of body-joint indexes in (5.20) is equivalent to the concatenation of $\boldsymbol{\mathcal{V}}_{ii'}$ from (5.21) along the fourth mode. Thus, we obtain tensor representations $\left[\boldsymbol{\mathcal{V}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}$ for sequence Π_A and $\left[\tilde{\boldsymbol{\mathcal{V}}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}$ for sequence Π_B.

The physical meaning of Eq. (5.21) is detailed in Figure 5.5. The dot-product can

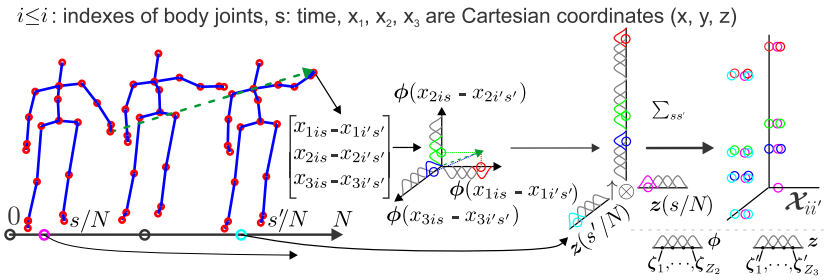


Figure 5.5: Third-order statistics from Eq. (5.19) can be understood by studying the linearization in Eq. (5.20). For a given pair of joints $i \leq i'$ at times s/N and s'/N (normalized frame numbers), we embed displacement vectors $\mathbf{x}_{is} - \mathbf{x}_{i's'}$ of 3D joint coordinates $\mathbf{x}_{is}$and $\mathbf{x}_{i's'}$ via function $\boldsymbol{\phi}(\cdot)$ into a non-linear Hilbert space representing an RBF kernel according to Eq. (5.2). Similarly, we embed the starting and ending times s/N and s'/N via function $\mathbf{z}(\cdot)$ (also by Eq. (5.2)). Finally, $\otimes$ performs the third-order outer-product on concatenated displacement and time embeddings aggregated next over frames s and s' (note $\sum_{ss'}$). The interpretation: the Gaussians 'soft-divide' the Cartesian coordinate system along x, y, z direction, resp., as well as time direction (s/N and s'/N). We project displacements along x, y, z directions of Cartesian coordinates and assign each projection to Gaussians. Thus, triplets $([x;y;z], s, s')$ assigned into such a 'soft-divided' space capture locally displacements of pairs of joints on the time grid (3-way soft-histogram). For DCK $\oplus$ in Section 5.4.6 we use velocity vectors $\frac{\mathbf{x}_{is} - \mathbf{x}_{i's'}}{\max(1, |s'-s|)}$ (*c.f.* displacement vectors) with short- and long-term estimates depending on $s'-s$ (3-way soft-histogram of short- and long-term speeds).

be now applied between these representations to compare them. Tensors $\boldsymbol{\mathcal{X}}$ in (5.21) are non-symmetric. Thus, for the operator $\boldsymbol{\mathcal{G}}$, we choose the HOSVD-based tensor whitening EPN, that is, tEPN defined in Eq. (5.15-5.18).

5.4.5 Sequence Compatibility Kernel 'Plus' (SCK $\oplus$)

Below, we extend the SCK formulation from Section 5.4.3 to aggregate over multiple subsequences extracted from the input sequence. Intuitively, this process is an equivalent of extracting local descriptors from images to attain so-called shift-invariance to the object location. As it is unlikely that relevant motion patterns stretch throughout a sequence, a specific pattern associated with some action classes may appear in one/few subsequences. Moreover, in what follows next, we will allow the aggregation to run over multiple modalities $q \in \mathcal{I}_Q$ *e.g.*, we use 3D body-joints and/or frame-wise CNN classification scores from RGB videos and/or optical flow. Thus, we can define our multimodal pose sequence Π as:

$$\Pi = \left\{ \mathbf{x}_{is}^{(q)} \in \mathbb{R}^{W_q}, i \in \mathcal{I}_J, s \in \mathcal{I}_M, q \in \mathcal{I}_Q \right\}, \tag{5.22}$$

where $W_1 = 3$, J is the total number of body-joints, W_q for $q > 1$ equals the size of modality q other than body-joints. Note that if modality $q > 1$ is global rather than per-joint specified, we can replicate it *e.g.*, $\mathbf{x}_{1s}^{(q)} = ... = \mathbf{x}_{Js}^{(q)}$.

SCK $\oplus$ on a pair of sequences Π_A and Π_B of length M and N is defined as:

$$K_{S^{\oplus}}(\Pi_A,\Pi_B) = \tag{5.23}$$
$$\frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{\substack{\tau\in\mathcal{P}_A\\ \tau'\in\mathcal{P}_B}}\sum_{\substack{u\in\mathcal{U}_\tau^A\\ u'\in\mathcal{U}_{\tau'}^B}}\sum_{\substack{s\in\mathcal{S}_\tau\\ t\in\mathcal{S}_{\tau'}}}\Big(\sum_{q\in\mathcal{I}_Q}\beta_1^{(q)}G_{\sigma_2^{(q)}}\left(\mathbf{x}_{i,u+s}^{(q)}-\mathbf{y}_{i,u'+t}^{(q)}\right)+$$
$$\beta_2\,G_{\sigma_3}(f(s,\mathcal{S}_\tau)-f(t,\mathcal{S}_{\tau'}))+$$
$$\beta_3\,G_{\sigma_4}(f(u,\mathcal{U}_\tau^A)-f(u',\mathcal{U}_{\tau'}^B))+$$
$$\beta_4\,G_{\sigma_5}(f(\tau,\mathcal{P}_A)-f(\tau',\mathcal{P}_B))\Big)^r.$$

Symbols $\mathcal{P}_A$ and $\mathcal{P}_B$ denote subsequence lengths, $\mathcal{P}_A=\mathcal{P}_B=\mathcal{P}$ is a possible assertion to make, so that *i.e.* $\mathcal{P}=\{8,10,12,...,20\}$. Moreover, $\mathcal{U}_\tau^A$ and $\mathcal{U}_{\tau'}^B$ are sets of all positions in sequences π_A and π_B for subsequences of lengths τ and τ', respectively, *i.e.*, if $N=100$ and $\tau=20$ then $\mathcal{U}_{20}^A=\{1,3,5,...,79\}$ is an example of a possible choice. Furthermore, $\mathcal{S}_\tau$ and $\mathcal{S}_{\tau'}$ are sets of all sampling positions in subsequences of lengths τ and τ', *i.e.*, if $\tau=20$ then $\mathcal{S}_{20}=\{0,1,2,...,19\}$ is an example of a possible choice. We define a function $f(s,\mathcal{S})=\frac{s-\mathcal{S}^{min}}{\mathcal{S}^{max}-\pi^{min}}$ which performs normalization on s w.r.t. set π, and $\mathcal{S}^{min}$ and $\mathcal{S}^{max}$ denote the smallest and largest element of set $\mathcal{S}$, respectively. Moreover, normalizations $f(u,\mathcal{U})$ and $f(\tau,\mathcal{P})$ are defined by analogy, $\Lambda=\Lambda_A\cdot\Lambda_B=(|\mathcal{I}_J|\cdot|\mathcal{P}_A|\cdot|\mathcal{U}_\tau^A|\cdot|\mathcal{S}_\tau|)\cdot(|\mathcal{I}_J|\cdot|\mathcal{P}_B|\cdot|\mathcal{U}_\tau^B|\cdot|\mathcal{S}_{\tau'}|)$. For simplicity, we do not model the within-sequence similarity between the body joints in contrast to Eq. (5.7), thus we skip G_{σ_1}. Kernels $G_{\sigma_2^{(i)}}$ capture the compatibility between body-joint locations $\mathbf{x}$ and $\mathbf{y}$ in a subsequence. Kernel G_{σ_3} measures the temporal alignment of two pose snippets in the given two subsequences. Kernel G_{σ_4} measures the temporal alignment of two subsequences in two sequences. Lastly, G_{σ_5} measures the match of two subsequence lengths. Weight factors $\beta_1^{(q)}\geq 0$ adjust the importance of each modality $q\in\mathcal{I}_Q$. Weight $\beta_2\geq 0$ is the importance of the temporal alignment of snippets within subsequences. Weight $\beta_3\geq 0$ is the importance of the temporal alignment of subsequences within sequences. Weight $\beta_4\geq 0$ is the importance of the match of two subsequence lengths. We let $\sum_q\beta_1^{(q)}+\beta_2+\beta_3+\beta_4=1$. Parameters $\sigma_2^{(q)}$ in $G_{\sigma_2^{(q)}}$ and $\beta_1^{(q)}$ are set per modality *e.g.*, for the 3D body-joints we chose $G_{\sigma_2^{(1)}}$ to be an RBF kernel, for frame-wise class predictions obtained from CNNs applied on (i) RGB and (ii) optical flow frames we choose $G_{\sigma_2^{(2)}}$ and $G_{\sigma_2^{(3)}}$ to be linear kernels (with no parameters). As previously, r denotes the order of captured statistics *i.e.*, $r=3$.

Below, we present the process of linearization of our kernel which follows the reasoning from Section 5.3.2 and Eq. (5.3). However, we feel it is interesting to show how various kernel components translate to various statistics encoded by the tensor:

i. $G_{\sigma_2^{(q)}}(\mathbf{x}-\mathbf{y})\approx\boldsymbol{\phi}^{(q)}(\mathbf{x})^T\boldsymbol{\phi}^{(q)}(\mathbf{y})$ (see note[2]) and, in order to reflect the choice of par. $\sigma_2^{(q)}$ for index q, we write $\boldsymbol{\phi}^{(q)}$,

ii. $G_{\sigma_3}(f(s,\mathcal{S}_\tau)-f(t,\mathcal{S}_{\tau'}))\approx\mathbf{z}'(f(s,\mathcal{S}_\tau)^T\mathbf{z}'(f(t,\mathcal{S}_{\tau'}))$,

iii. $G_{\sigma_4}(f(u,\mathcal{U}_\tau)-f(u',\mathcal{U}_{\tau'}))\approx\mathbf{z}''(f(u,\mathcal{U}_\tau)^T\mathbf{z}''(f(u',\mathcal{U}_{\tau'}))$,

iv. $G_{\sigma_5}(f(\tau,\mathcal{P}_A)-f(\tau',\mathcal{P}_B))\approx\mathbf{z}'''(f(\tau,\mathcal{P}_A)^T\mathbf{z}'''(f(\tau',\mathcal{P}_B))$.

With these approximations at hand, we rewrite our sequence compatibility kernel 'plus' as:

$$K_{S^{\oplus}}(\Pi_A,\Pi_B) \approx$$

$$\frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{\substack{\tau\in\mathcal{P}_A\\ \tau'\in\mathcal{P}_B}}\sum_{\substack{u\in\mathcal{U}_\tau\\ u'\in\mathcal{U}_{\tau'}}}\sum_{\substack{s\in\mathcal{S}_\tau\\ t\in\mathcal{S}_{\tau'}}}\left(\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(1)})\\ \cdots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(Q)})\\ \sqrt{\beta_2}\,\mathbf{z}'(f(s,\mathcal{S}_\tau))\\ \sqrt{\beta_3}\,\mathbf{z}''(f(u,\mathcal{U}_\tau))\\ \sqrt{\beta_4}\,\mathbf{z}'''(f(\tau,\mathcal{P}_A))\end{bmatrix}^T\cdot\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(1)})\\ \cdots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(Q)})\\ \sqrt{\beta_2}\mathbf{z}'(f(t,\mathcal{S}_{\tau'}))\\ \sqrt{\beta_3}\mathbf{z}''(f(u',\mathcal{U}_{\tau'}))\\ \sqrt{\beta_4}\mathbf{z}'''(f(\tau',\mathcal{P}_B))\end{bmatrix}\right)^r = \tag{5.24}$$

$$\sum_{i\in\mathcal{I}_J}\left\langle\boldsymbol{\mathcal{G}}\left(\frac{1}{\Lambda_A}\sum_{\tau\in\mathcal{P}_A}\sum_{u\in\mathcal{U}_\tau}\sum_{s\in\mathcal{S}_\tau}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(1)})\\ \cdots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(Q)})\\ \sqrt{\beta_2}\,\mathbf{z}'(f(s,\mathcal{S}_\tau))\\ \sqrt{\beta_3}\,\mathbf{z}''(f(u,\mathcal{U}_\tau))\\ \sqrt{\beta_4}\,\mathbf{z}'''(f(\tau,\mathcal{P}_A))\end{bmatrix}\right),\right. \tag{5.25}$$

$$\left.\boldsymbol{\mathcal{G}}\left(\frac{1}{\Lambda_B}\sum_{\tau'\in\mathcal{P}_B}\sum_{u'\in\mathcal{U}_{\tau'}}\sum_{t\in\mathcal{S}_{\tau'}}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1^{(1)}}\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(1)})\\ \cdots\\ \sqrt{\beta_1^{(Q)}}\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(Q)})\\ \sqrt{\beta_2}\mathbf{z}'(f(t,\mathcal{S}_{\tau'}))\\ \sqrt{\beta_3}\mathbf{z}''(f(u',\mathcal{U}_{\tau'}))\\ \sqrt{\beta_4}\mathbf{z}'''(f(\tau',\mathcal{P}_B))\end{bmatrix}\right)\right\rangle.$$

In the above equation, we set $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{X}}$ for Eq. (5.25) to be equivalent to Eq. (5.24). However, similarly to considerations in Section 5.4.3, a commonly encountered adversity in aggregated representations, the *burstiness*, requires some suppression. To this end, we let operator $\boldsymbol{\mathcal{G}}$ in Eq. (5.25) perform tEPN on the spectrum of the third-order tensor.

The final representation for linearized SCK $\oplus$ becomes:

$$\boldsymbol{\mathcal{V}}_i = \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}_i),\ \text{where}\ \boldsymbol{\mathcal{X}}_i = \frac{1}{\Lambda_A}\sum_{\tau\in\mathcal{P}_A}\sum_{u\in\mathcal{U}_\tau}\sum_{s\in\mathcal{S}_\tau}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(1)})\\ \cdots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(Q)})\\ \sqrt{\beta_2}\,\mathbf{z}'(f(s,\mathcal{S}_\tau))\\ \sqrt{\beta_3}\,\mathbf{z}''(f(u,\mathcal{U}_\tau))\\ \sqrt{\beta_4}\,\mathbf{z}'''(f(\tau,\mathcal{P}_A))\end{bmatrix}. \tag{5.26}$$

We can further replace the summation over the body-joint indexes in (5.25) by concatenating $\boldsymbol{\mathcal{V}}_i$ in (5.26) along the fourth tensor mode, thus defining $\boldsymbol{\mathcal{V}} = \left[\boldsymbol{\mathcal{V}}_i\right]_{i\in\mathcal{I}_J}^{\oplus_4}$. Suppose $\boldsymbol{\mathcal{V}}_A$ and $\boldsymbol{\mathcal{V}}_B$ are the corresponding fourth order tensors for Π_A and Π_B, then we have:

$$K^*_{S^{\oplus}}(\Pi_A,\Pi_B) = \langle\boldsymbol{\mathcal{V}}_A,\boldsymbol{\mathcal{V}}_B\rangle\,. \tag{5.27}$$

Note that in general $K^*_{S^{\oplus}}(\Pi_A,\Pi_B) \not\approx K_{S^{\oplus}}(\Pi_A,\Pi_B)$ as $\boldsymbol{\mathcal{G}}$ manipulates the spectrum of $\boldsymbol{\mathcal{X}}$. Finally, for our final representation, we use only the upper-simplices of $\boldsymbol{\mathcal{V}}_i$ which consist of $\binom{d+r-1}{r}$ coefficients each, rather than d^r, where d is the side-dimension of $\boldsymbol{\mathcal{V}}_i$ *i.e.*, $d = 3Z_2^{(1)} + \ldots + Z_2^{(Q)} + Z_3 + Z_4 + Z_5$ (see notes[2,3]), and $Z_2^{(1)},\ldots,Z_2^{(Q)}$ and Z_3, Z_4, Z_5 are the numbers of pivots used in the approximation of $G_{\sigma_2^{(1)}},\ldots,G_{\sigma_2^{(Q)}}$ and $G_{\sigma_3}, G_{\sigma_4}, G_{\sigma_5}$

(see notes[2,3]).

5.4.6 Dynamics Compatibility Kernel 'Plus' (DCK ⊕)

Below, we apply the aggregation over subsequences to our DCK kernel. We follow the same steps as for SCK ⊕ (Section 5.4.5) except that our subsequences for DCK ⊕ have a fixed length. For a pair of sequences Π_A and Π_B of length M and N, we have:

$$K_{D^{\oplus}}(\Pi_A, \Pi_B) = \frac{1}{\Lambda'} \sum_{u,u' \in \mathcal{U}_\tau} K_D(\Pi'_{A,\tau,u}, \Pi'_{B,\tau,u'}) G_{\sigma_4}(f(u, \mathcal{U}_\tau^A) - f(u', \mathcal{U}_\tau^B)), \tag{5.28}$$

where τ is a length of subsequences. $K_D(\Pi'_{A,\tau,u}, \Pi_{B',\tau,u'})$ is defined in Eq. (5.19). However, we use velocity vectors $\frac{\mathbf{x}_{is} - \mathbf{x}_{i's'}}{\max(1, |s'-s|)}$ (*c.f.* displacement vectors in DCK) with short- and long-term estimates depending on $s'-s$. Figure 5.5 provides an interpretation of this kernel. $K_D(\Pi'_{A,\tau,u}, \Pi_{B',\tau,u'})$ is evaluated over subsequences $\Pi'_{A,\tau,u}$ and $\Pi'_{B,\tau,u'}$ sampled from Π_A and Π_B according to sets of sampling coordinates $\mathcal{S}_{\tau,u} = \{\mathcal{S}_\tau\} + u$ and $\mathcal{S}_{\tau,u'} = \{\mathcal{S}_\tau\} + u'$ of length τ which are shifted by locations u and u' according to $\mathcal{U}_\tau$. Lastly, $\Lambda' = |\mathcal{U}_\tau^A| \cdot |\mathcal{U}_\tau^B|$. The remaining symbols follow definitions in Section 5.4.5. Kernel in Eq. (5.28) is then linearized in the similar manner to Eq. (5.19) which results in linearization similar to Eq. (5.21) but containing an additional mode corresponding to linearization of kernel G_{σ_4}. We skip this derivation for brevity.

$$\begin{aligned}
K_D(\Pi_A, \Pi_B) &= \frac{1}{\Lambda} \sum_{\substack{(i,s) \in \mathcal{J}, \\ (i',s') \in \mathcal{J}, \\ i' \neq i, s' \neq s}} \sum_{\substack{(j,t) \in \mathcal{J}, \\ (j',t') \in \mathcal{J}, \\ j' \neq j, t' \neq t}} G'_{\sigma'_1}(i-j, i'-j')\, G_{\sigma'_2}\left((\mathbf{x}_{is} - \mathbf{x}_{i's'}) - \left(\mathbf{y}_{jt} - \mathbf{y}_{j't'}\right)\right) G'_{\sigma'_3}\left(\frac{s-t}{N}, \frac{s'-t'}{N}\right) \cdot G'_{\sigma'_4}(s-s', t-t') \\
&= \frac{1}{\Lambda} \sum_{\substack{i,i' \in \mathcal{I}_f: \\ i' \neq i}} \sum_{\substack{s,s' \in \mathcal{I}_N: \\ s' \neq s}} \sum_{\substack{t,t': \\ t' \neq t}} G_{\sigma'_2}\left((\mathbf{x}_{is} - \mathbf{x}_{i's'}) - \left(\mathbf{y}_{jt} - \mathbf{y}_{j't'}\right)\right) G_{\sigma'_3}\left(\frac{s-t}{N}\right) G_{\sigma'_3}\left(\frac{s'-t'}{N}\right) \cdot G_{\sigma'_4}(s-s')\, G_{\sigma'_4}(t-t') \Big|_{\substack{j=i \\ j'=i'}} \\
&\approx \frac{1}{\Lambda} \sum_{\substack{i,i' \in \mathcal{I}_f: \\ i' \neq i}} \sum_{\substack{s,s' \in \mathcal{I}_N: \\ s' \neq s}} \sum_{\substack{t,t' \in \mathcal{I}_N: \\ t' \neq t}} \boldsymbol{\phi}\left(\mathbf{x}_{is} - \mathbf{x}_{i's'}\right)^T \boldsymbol{\phi}\left(\mathbf{y}_{it} - \mathbf{y}_{i't'}\right) \cdot \mathbf{z}\left(\frac{s}{N}\right)^T \mathbf{z}\left(\frac{t}{N}\right) \cdot \mathbf{z}\left(\frac{s'}{N}\right)^T \mathbf{z}\left(\frac{t'}{N}\right) \cdot G_{\sigma'_4}(s-s')\, G_{\sigma'_4}(t-t') \\
&= \frac{1}{\Lambda} \sum_{\substack{i,i' \in \mathcal{I}_f: \\ i' \neq i}} \sum_{\substack{s,s' \in \mathcal{I}_N: \\ s' \neq s}} \sum_{\substack{t,t' \in \mathcal{I}_N: \\ t' \neq t}} \left\langle G_{\sigma'_4}(s-s') \left(\boldsymbol{\phi}(\mathbf{x}_{is} - \mathbf{x}_{i's'}) \cdot \mathbf{z}\left(\frac{s}{N}\right)^T\right)\uparrow \otimes \mathbf{z}\left(\frac{s'}{N}\right),\, G_{\sigma'_4}(t-t') \left(\boldsymbol{\phi}(\mathbf{y}_{it} - \mathbf{y}_{i't'}) \cdot \mathbf{z}\left(\frac{t}{N}\right)^T\right)\uparrow \otimes \mathbf{z}\left(\frac{t'}{N}\right) \right\rangle \\
&= \sum_{\substack{i,i' \in \mathcal{I}_f: \\ i' \neq i}} \left\langle \frac{1}{\sqrt{\Lambda}} \sum_{\substack{s,s' \in \mathcal{I}_N: \\ s' \neq s}} G_{\sigma'_4}(s-s') \left(\boldsymbol{\phi}(\mathbf{x}_{is} - \mathbf{x}_{i's'}) \cdot \mathbf{z}\left(\frac{s}{N}\right)^T\right)\uparrow \otimes \mathbf{z}\left(\frac{s'}{N}\right),\, \frac{1}{\sqrt{\Lambda}} \sum_{\substack{t,t' \in \mathcal{I}_N: \\ t' \neq t}} G_{\sigma'_4}(t-t') \left(\boldsymbol{\phi}(\mathbf{y}_{it} - \mathbf{y}_{i't'}) \cdot \mathbf{z}\left(\frac{t}{N}\right)^T\right)\uparrow \otimes \mathbf{z}\left(\frac{t'}{N}\right) \right\rangle
\end{aligned} \tag{5.29}$$

$$K^*_D(\Pi_A, \Pi_B) = \sum_{\substack{i,i' \in \mathcal{I}_f: \\ i' \neq i}} \left\langle \boldsymbol{\mathcal{G}}\left(\frac{1}{\sqrt{\Lambda}} \sum_{\substack{s,s' \in \mathcal{I}_N: \\ s' \neq s}} G_{\sigma'_4}(s-s') \left(\boldsymbol{\phi}(\mathbf{x}_{is} - \mathbf{x}_{i's'}) \cdot \mathbf{z}\left(\frac{s}{N}\right)^T\right)\uparrow \otimes \mathbf{z}\left(\frac{s'}{N}\right)\right), \boldsymbol{\mathcal{G}}\left(\frac{1}{\sqrt{\Lambda}} \sum_{\substack{t,t' \in \mathcal{I}_N: \\ t' \neq t}} G_{\sigma'_4}(t-t') \left(\boldsymbol{\phi}(\mathbf{y}_{it} - \mathbf{y}_{i't'}) \cdot \mathbf{z}\left(\frac{t}{N}\right)^T\right)\uparrow \otimes \mathbf{z}\left(\frac{t'}{N}\right)\right) \right\rangle \tag{5.30}$$

5.5 Experiments

Below, we present experiments on our models on seven popular datasets. For datasets based on 3D skeletons, we use (i) the UTKinect-Action3D [Xia et al., 2012], (ii) Florence3D-Action [Seidenari et al., 2013], (iii) MSR-Action3D [Li et al., 2010], and (iv) Kinetics [Kay et al., 2017] (where stated). For datasets based on RGB frames, we use (v) the fine-grained MPII Cooking Activities [Rohrbach et al., 2012] and (vi) HMDB-51 [Kuehne et al., 2011] datasets. For experiments on the 3D skeletons fused with RGB frames, we use (vii) large scale NTU-RGBD [Shahroudy et al., 2016a] dataset. We also evaluate the influence of various hyper-parameters, such as the number of pivots Z used for linearizing the body-joint and temporal kernels, and the impact of Eigenvalue Power Normalization (we vary the factor equalization). We evaluate our older SCK and DCK kernels, and their newer counterparts SCK$\oplus$ and DCK$\oplus$. For skeletons, we feed them directly to our kernel representations while RGB-based datasets are firstly encoded by the two-stream CNN [Simonyan and Zisserman, 2014] or the I3D [Carreira and Zisserman, 2017].

5.5.1 Datasets

UTKinect-Action3D [Xia et al., 2012] consists of 10 actions performed twice by 10 different subjects, and has 199 action sequences. The dataset provides 3D coordinate annotations of 20 body-joints for every frame. The dataset was captured with a stationary Kinect sensor and contains significant viewpoint and intra-class variations.
Florence3D-Action [Seidenari et al., 2013] dataset consists of 9 actions performed 2–3$\times$ by 10 different subjects and it has 215 action sequences. 3D coordinate annotations of 15 body-joints captured with a Kinect sensor are provided. Significant intra-class variations are present *i.e.*, the same action articulated with the left/right hand, and actions like *drinking/performing a phone call* can be seen as fine-grained.
MSR-Action3D [Li et al., 2010] dataset is comprised of 20 actions performed 2–3$\times$ by 10 different subjects and it has 567 action sequences. 3D coordinates of 20 body-joints captured by a Kinect-like depth sensor are provided. MSR-Action3D has strong inter-class similarity.

In the above datasets, we use the cross-subject test setting (unless stated otherwise), in which half of the subjects are used for training and the remaining half for testing. Similarly, we divide the training set into two halves for the purpose of training/validation.
NTU-RGBD [Shahroudy et al., 2016a] is by far the largest 3D skeleton-based video action recognition dataset. It has 56880 video sequences across 60 classes, 40 subjects, and 80 views. The videos have on average 70 frames and consist of people performing various actions. Each frame is annotated with 25 human skeletal keypoints (some videos have multiple subjects). Two evaluation protocols are used for this dataset, namely, cross-subject and cross-view evaluation. This dataset can be considered as having many fine-grained classes *e.g.*, *make a phone call*, *playing with phone*, *punching other person*, *pushing other person*, *pat on back of other person*, *etc*.

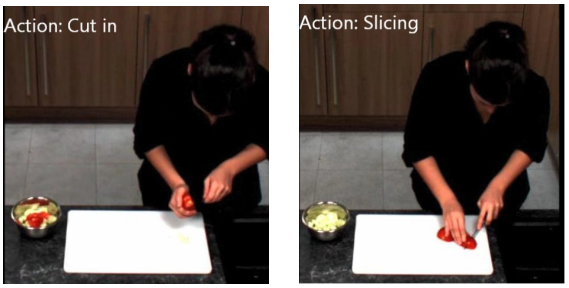


Figure 5.6: Fine-grained action instances (MPII Cooking Activities [Rohrbach et al., 2012]) from two different action categories: *cut-in* (left) and *slicing* (right).

MPII Cooking Activities [Rohrbach et al., 2012] dataset consists of high-resolution videos of cooking activities/people cooking various dishes. There are 64 distinct activities spread across 3748 video clips and one background activity (1861 clips). Activities include coarse actions *e.g.*, *opening refrigerator*, and fine-grained actions *e.g.*, *peel*, *slice*, *cut apart* (see Figure 5.6). This dataset is challenging due to (i) unbalanced action classes, (ii) significant intra-class differences (each subject cooks according to their own style). We use the mean Average Precision (mAP) over 7-fold cross-validation.
HMDB-51 [Kuehne et al., 2011] dataset is a popular video benchmark for human action recognition, consisting of 6766 Internet videos over 51 classes. Each video has about 20–1000 frames. We report the average classification accuracy on standard three-fold splits.
Kinetics [Kay et al., 2017] contains ∼300000 clips from YouTube which cover 400 human action classes, ranging from daily activities, sports scenes, to complex interactions. Each clip is ∼10 seconds long.

5.5.2 Experimental Setup

For our experiments, we distinguish four configurations: (i) for UTKinect-Action3D, Florence3D-Action and MSR-Action3D that provide 3D body-joints, we feed sequences of 3D body-joints to our kernel(s), (ii) for MPII Cooking Activities, HMDB-51 and NTU-RGBD that provide RGB frames, we train a two-stream ResNet-152 model (as in [Simonyan and Zisserman, 2014]) taking RGB frames (in the spatial stream) and a stack of optical flow frames (in the temporal stream) as input to obtain classification scores per frame per stream which are then passed to our kernel, (iii) for NTU-RGBD which contains both 3D body-joints and RGB frames, we investigate both such inputs separately as well as their combination, and (iv) for Kinetics, we use skeletons and combine ST-GCN with SCK.

For the sequence compatibility kernel on sequences of 3D body-joints, we first normalized all body-keypoints with respect to the hip joints across frames, as indicated in Section 5.4.3. We also normalized lengths of all body-parts w.r.t. to a reference skeleton. This setup follows pre-processing of [Vemulapalli et al., 2014]. For our dynamics compatibility kernel, we use unnormalized body-joints and assume that the displacements of body-joint coordinates across frames capture their temporal

evolution implicitly. For the sequence compatibility kernel on classifier scores, we take the scores before they are passed through the logistic function and we apply a rectifier.

CNN Training. To extract features with CNN, we train a two-stream ResNet-152 model [Simonyan and Zisserman, 2014] taking RGB frames (in the spatial stream) and a stack of optical flow frames (in the temporal stream) from a given training split as input. For optical flow, we use the Large Displacement Optical Flow (LDOF) [Brox and Malik, 2011]. We use the classifier predictions from each stream as inputs to our kernels. The two streams of the CNN are trained separately on the respective input modalities against a softmax cross-entropy loss. We simply follow the standard training protocols from [Simonyan and Zisserman, 2014]. For fine-tuning, we used a fixed learning rate of $1e-4$ and a momentum of 0.9. For the MPII Cooking Activities dataset, we used the sequences from the training set for training the CNNs (1992 sequences) and those from the validation set (615 sequences) to check for overfitting. For HMDB-51, we use three standard splits provided with the dataset. For NTU-RGBD dataset in the cross-subject evaluation, the training and testing sets have 40320 and 16560 samples, respectively. For NTU-RGBD dataset in the cross-view evaluation, the training and testing sets have 37920 and 18960 samples, respectively. We use 70% of the training set for training and 30% for validation. To train SVM, we simply vectorize our tensors and set $c=1e-2$.

To stay competitive w.r.t. the state of the art, we additionally use two newer backbones such as (i) Spatial Temporal Graph Convolutional Network (ST-GCN) [Yan et al., 2018] and (ii) Two-Stream Inflated 3D ConvNet (I3D) [Carreira and Zisserman, 2017]. For ST-GCN, we train it on skeletal sequences from NTU and Kinetics [Kay et al., 2017] datasets following the standard protocols. For Kinetics, we follow approach [Kay et al., 2017] and use skeletons extracted with OpenPose [Cao et al., 2017]. Finally, we combine our vectorized tensors from SCK or SCK$\oplus$ with the output of the last layer of ST-GCN preceding the classifier, and feed such a representation into the cross-entropy loss. As SCK is a shallow approach, we expect it to be highly complementary with ST-GCN. For I3D network, we train it on subsequences extracted from HMDB51 and MPII. We use RGB and optical flow (LDOF) streams. We extract subsequences of length 48, 64, 80, 96 given strides 1, 2 and 3. Then, subsequences shorter than 64 are lapped. We put together all training subsequences of all lengths and all strides, and we train RGB and LDOF I3D networks separately with a learning rate $1e-4$ halved every 10 epochs.

IDT Features. On HMDB-51 and MPII Cooking Activities, we also report accuracy when our kernel is combined with dense trajectories [Wang et al., 2011] encoded by Fisher Vectors [Perronnin et al., 2010].

5.5.3 Sequence compatibility kernel.

In this section, we first present experiments evaluating the influence of parameters σ_2 and σ_3 of kernels G_{σ_2} and G_{σ_3} which control the degree of selectivity for the 3D body-joints and the temporal shift invariance, respectively. See Section 5.4.3 for a full

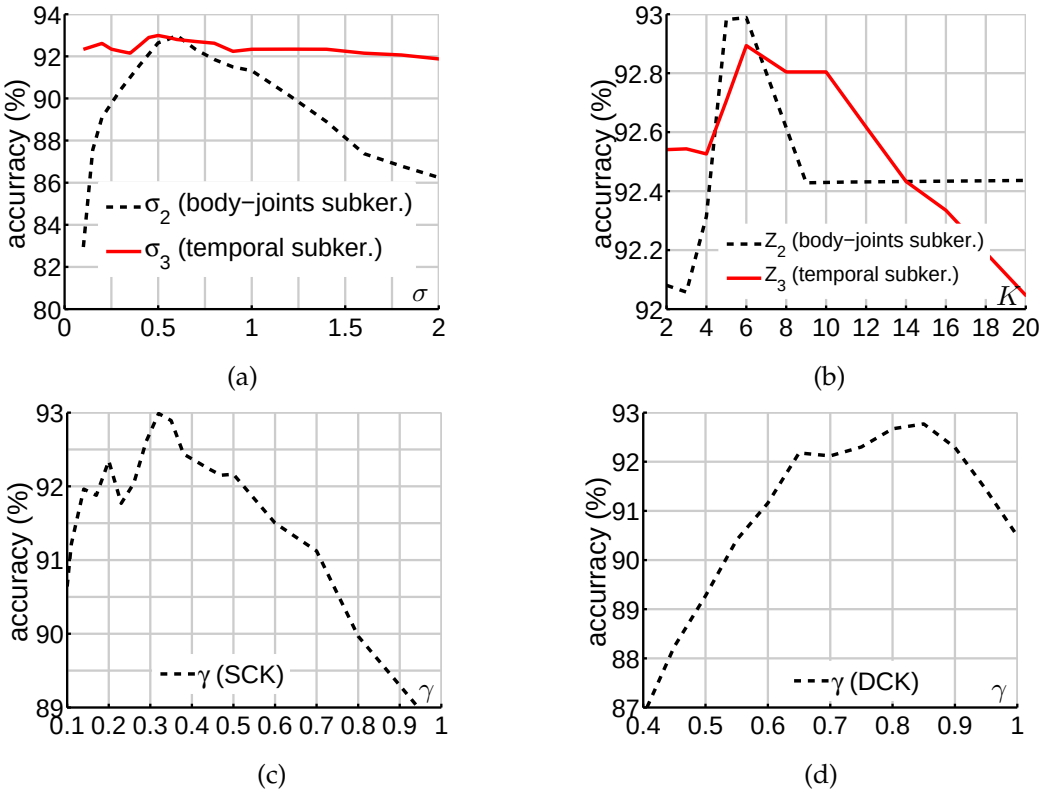
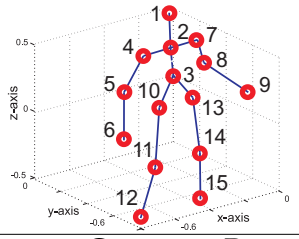


Figure 5.7: Figure 5.7a illustrates the classification accuracy on Florence3d-Action for the sequence compatibility kernel when varying radii σ_2 (body-joints subkernel) and σ_3 (temporal subkernel). Figure 5.7b evaluates behavior of SCK w.r.t. the number of pivots Z_2 and Z_3. Figure 5.7c demonstrates effectiveness of our slice-wise Eigenvalue Power Normalization in tackling burstiness by varying parameter γ. Figure 5.7d shows effectiveness of equalizing the factors in non-symmetric tensor representation by HOSVD Eigenvalue Power Normalization by varying γ.

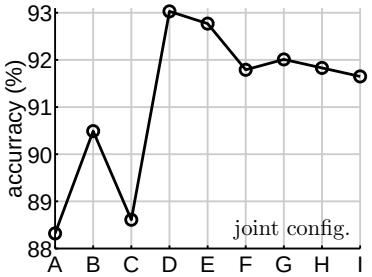
definition of these parameters. Recall that kernels G_{σ_2} and G_{σ_3} are approximated via linearizations according to Eq. (5.1) and (5.3). The quality of these approximations and the size of our final tensor representations depend on the numbers Z_2 and Z_3 of pivots chosen. See Section 5.3.2, Figure 5.3 and notes[2],[3] for details on pivots. In our experiments, the pivots ζ are spaced uniformly within interval $[-1;1]$ and $[0;1]$ for kernels G_{σ_2} and G_{σ_3} respectively.

Figures 5.7a and 5.7b present the results of this experiment on the Florence3D-Action dataset. Figure 5.7a shows that the body-joint compatibility subkernel G_{σ_2} requires a choice of σ_2, which is not too strict as specific body-joints (*e.g.*, elbow) are expected to repeat across sequences in similar locations due to zero-centering w.r.t. hip. On the one hand, very small σ_2 leads to poor generalization. On the other hand, very large σ_2 allows big displacements of the corresponding body-joints between sequences which results in a poor discriminative power of this kernel. Furthermore, Figure 5.7a demonstrates that the range of σ_3 for the temporal subkernel for which we obtain very good performance is large. However, as σ_3 becomes very small or very



A	B	C	D	E
6,9	1,6,9	6,9,12,15	4,6,7,9,11,14	4,6,7,9,
F	G	H	I	11,12,
4-15	1,4-15	1,2,4-15	1-15	14,15

(a)



(b)

Figure 5.8: Figure 5.8a enumerates the body-joints in the Florence3D-Action dataset. The table below lists subsets A-I of the body-joints used to build representations eval. in Figure 5.8b, which shows the accuracy of our dynamics compatibility kernel w.r.t. these subsets.

large, extreme temporal selectivity or full temporal invariance, respectively, result in a loss of performance. For instance, $\sigma_3 = 4$ results in 91% accuracy only.

In Figure 5.7b, we show the performance of our SCK kernel with respect to the number of pivots used for linearization. For the body-joint compatibility subkernel G_{σ_2}, we see that $Z_2 = 5$ pivots are sufficient to obtain good performance of 92.98% accuracy. We have observed that this is consistent with the results on the validation set. Using more pivots, say $Z_2 = 20$, deteriorates the results slightly, suggesting overfitting. We make similar observations for the temporal subkernel G_{σ_3} which demonstrates a good performance for as few as $Z_3 = 2$ pivots. Such a small number of pivots suggests that linearizing 1D variables and generating higher-order co-occurrences, as described in Section 5.4.3, constitute on a simple, robust, and effective linearization strategy.

Furthermore, Figure 5.7c demonstrates the effectiveness of our slice-wise Eigenvalue Power Normalization described in Eq. (5.14). When $\gamma = 1$, the EPN functionality is absent. This results in a drop of performance from 92.98% to 88.7% accuracy. This demonstrates that statistically unpredictable bursts of actions described by body-joints, such as long versus short *hand waving*, are indeed undesirable. It is clear that in such cases, EPN is very effective, as it deals with correlated bursts, *e.g.* co-occurring *hand wave* and associated with it elbow and neck motion. For more details regarding this concept, see [Koniusz et al., 2016c]. For our further experiments, we choose $\sigma_2 = 0.6$, $\sigma_3 = 0.5$, $Z_2 = 5$, $Z_3 = 6$, and $\gamma = 0.36$, as dictated by cross-validation.

5.5.4 Dynamics compatibility kernel.

Below, we evaluate the influence of parameters of the DCK kernel. Our experiments are based on the Florence3D-Action dataset. For ablations, we present results on the test set as results on the validation set match test results closely. As this kernel considers all spatio-temporal co-occurrences of body-joints, we firstly evaluate the

	SCK	DCK		SCK+DCK
accuracy	92.98%	93.03%	92.77%	**95.23**%
size	26,565	9,450	16,920	43,485

	SCK ⊕	DCK ⊕	SCK ⊕ + DCK ⊕
accuracy	96.50%	96.41%	**97.45**%
size	60,900	37,800	98,700

Bag-of-Poses 82.00% [Seidenari et al., 2013]	Kendal Traj. 93.04% [Tanfous et al., 2018]
$SE(3)$ 90.88% [Vemulapalli et al., 2014]	Kernel+ResNet 95.4% [Tas and Koniusz, 2018]

Table 5.1: Evaluations of (*top*) SCK/DCK, (*middle*) our improved SCK ⊕ / DCK ⊕, (*bottom*) the state of the art on Florence3D-Action.

	SCK	DCK	SCK+DCK
accuracy	96.08%	97.5%	**98.2**%
size	40,480	16,920	57,400

	SCK ⊕	DCK ⊕	SCK ⊕ + DCK ⊕
accuracy	98.50%	98.12%	**99.2**%
size	81,200	67,680	148,880

3D joints. hist. 90.92% [Xia et al., 2012]	Kendal Traj. 97.39% [Tanfous et al., 2018]
$SE(3)$ 97.08% [Vemulapalli et al., 2014]	Second-order DA 98.9% [Tas and Koniusz, 2018]

Table 5.2: Evaluations of (*top*) SCK/DCK, (*middle*) our improved SCK ⊕ / DCK ⊕ and (*bottom*) the state of the art on UTKinect-Action3D.

impact of the joint subsets we select for generating DCK, as not all body-joints need to be used for capturing actions.

Figure 5.8a enumerates all body-joints that describe every 3D human skeleton on the Florence3D-Action dataset, whereas the table underneath lists the proposed body-joint subsets A-I which we use for computations of DCK. In Figure 5.8b, we plot the performance of our DCK kernel for each subset. The plot shows that using two body-joints associated with the hands from Configuration-A in the DCK kernel construction, we attain 88.32% accuracy which highlights the informativeness of temporal dynamics.

Some body-joints may be noisy and thus detrimental to recognition, and should not be selected for experiments *e.g.*, Configuration-D, which includes six body-joints such as the knees, elbows and hands, yields 93.03%, which outperforms more complex configurations.

As Configuration-E includes eight body-joints such as the feet, knees, elbows and hands, we choose it for our further experiments as it represents a reasonable trade-off between performance and size of representations. This configuration scores 92.77%

	SCK	DCK	SCK+DCK
acc., prot. [Wang et al., 2012]	90.72%	86.30%	**91.45**%
acc., prot. [Li et al., 2010]	93.52%	91.71%	**93.96**%
size	40,480	16,920	57,400

	SCK $\oplus$	DCK $\oplus$	SCK $\oplus$ + DCK $\oplus$
acc., prot. [Wang et al., 2012]	97.50%	90.03%	**98.10**%
acc., prot. [Li et al., 2010]	98.12%	94.28%	**98.62**%
size	81,200	67,680	148,880

accuracy, protocol [Wang et al., 2012]	accuracy, protocol [Li et al., 2010]
Actionlets 88.20% [Wang et al., 2012]	$SE(3)$ 92.46% [Vemulapalli et al., 2014]
$SE(3)$ 89.48% [Vemulapalli et al., 2014]	Kendal Traj. 94.19% [Tanfous et al., 2018]
Kin. desc. 91.07% [Zanfir et al., 2013]	Ker-RP-RBF 96.9% [Wang et al., 2015a]

Table 5.3: Results of (*top*) SCK/DCK, (*middle*) our improved SCK $\oplus$ / DCK $\oplus$ and (*bottom*) the state of the art on MSR-Action3D.

accuracy. We see that if we utilize all the body-joints according to Configuration-I, performance of 91.65% accuracy is still somewhat lower compared to 93.03% accuracy for Configuration-D highlighting the issue of noisy body-joints.

In Figure 5.7d, we show the accuracy on our DCK kernel when HOSVD factors underlying our non-symmetric tensors are equalized by varying the EPN parameter γ. For $\gamma=1$, EPN is disabled, which leads to 90.49% accuracy only. For the optimal value of $\gamma=0.85$, the accuracy rises to 92.77% which indicates the presence of the burstiness effect in temporal representations.

5.5.5 SCK and DCK vs. the state of the art.

Below, we compare the performance of our representations against the state of the art. Along with comparing SCK and DCK, we also explore the complementarity of these representations by combining them via the so-called late fusion, that is, a simple weighted concatenation of vectorized SCK and DCK.

On the Florence3D-Action dataset, we present our best results in Table 5.1. Note that the model parameters for the evaluation was selected by cross-validation. Linearizing a sequence compatibility kernel using these parameters resulted in a tensor representation of size $26,565$ dimensions[4], and produced an accuracy of 92.98% accuracy. As for DCK, our model used Configuration-E (described in Figure 5.8a) resulting in a representation of dimensionality $16,920$, and achieved a performance of 92%. However, somewhat better results were attained by Configuration-D, that is, 92.27% accuracy for size of $9,450$. Combining SCK and DCK in Configuration-E yields 95.23% accuracy, a 4.5% improvement over the state of the art on this dataset, as listed in Table 5.1, which highlights the complementary nature of SCK and DCK.

[4]This is the length of a vector per sequence after unfolding our tensor represent./removing duplicate coefficients from the symmetries in the tensor.

		cross-subject	cross-view
SCK ($r=2$)	on 3D body-joints	64.08%	65.24%
SCK (no EPN)	on 3D body-joints	65.37%	67.18%
SCK	on 3D body-joints	69.20%	70.55%
SCK $\oplus$	on 3D body-joints	**72.82**%	**74.10**%
SCK	3D body-joints +ST-GCN	82.61%	89.52%
SCK $\oplus$	3D body-joints +ST-GCN	**83.58**%	**90.84**%

	cross-subject	cross-view
Two-stream+AP (ResNet-50)	74.4%	83.3%
Two-stream+MP (ResNet-50)	65.8%	58.7%
SCK $\oplus$ on RGB frames (ResNet-152)	**90.11**%	**93.62**%
SCK $\oplus$ 3D body-joints + RGB (ResNet-152)	**90.78**%	**94.15**%
SCK $\oplus$ 3D body-joints +RGB+optical flow (ResNet-152)	**91.56**%	**94.75**%

	cross-subject	cross-view
Second-order DA [Tas and Koniusz, 2018] (ResNet-50)	75.35%	79.30%
Frames + CNN [Ke et al., 2017b] (VGG-19)	75.73%	79.62%
Clips + CNN + MTLN [Ke et al., 2017b] (VGG-19)	79.57%	84.83%
VA-LSTM [Zhang et al., 2017b]	79.4%	87.6%
ST-GCN [Yan et al., 2018]	81.5%	88.3%
DSP [Wang and Cherian, 2018]	81.6%	88.7%
Multi-scale CNN [Li et al., 2017a] (ResNet-101)	84.6%	92.1%
Multi-scale CNN [Li et al., 2017a] (ResNet-152)	85.0%	92.3%
Deep Bilinear [Hu et al., 2018] (ResNet-101)	85.4%	90.7%

Table 5.4: Results on our SCK and the improved SCK $\oplus$ on (*top*) skeleton sequences and (*middle*) two-stream networks. We also indicate results on the baseline two-stream network with standard average pooling (*AP*) and maximum pooling (*MP*). We indicate backbones in parentheses. (*bottom*) The state of the art on NTU-RGBD.

Action recognition results on the UTKinect-Action3D dataset are presented in Table 5.2. For our experiments on this dataset, we kept all the parameters the same as those used on the Florence3D dataset. SCK and DCK representations yielded 96.08% and 97.5% accuracy, respectively. Combining SCK and DCK yielded 98.2% accuracy outperforming marginally a more complex approach [Vemulapalli et al., 2014] based on the Lie group algebra, dynamic time warping and Fourier temporal pyramids.

In Table 5.3, we present our results on the MSR-Action3D dataset. Conforming to the prior literature, we use two evaluation protocols, that is, (i) the protocol described in actionlets [Wang et al., 2012], for which the authors utilize the entire dataset with its 20 classes during the training and evaluation, and (ii) approach of [Li et al., 2010], for which the authors divide the data into three subsets and report the average in classification accuracy over these subsets. SCK yields the state-of-the-art accuracy of

	SCK+ST-GCN	SCK ⊕+ST-GCN	ST-GCN
top-1	31.2%	**31.8**%	30.7%
top-5	53.7%	**54.9**%	52.8%

Table 5.5: SCK and SCK ⊕ combined with ST-GCN *vs*. ST-GCN [Yan et al., 2018] alone on Kinetics [Kay et al., 2017] skeletons extracted by OpenPose [Cao et al., 2017].

90.72% and 93.52% for the two evaluation protocols, respectively. Combining SCK with DCK outperforms other approaches listed in the table and yields 91.45% and 93.96% accuracy for the two protocols, respectively.

5.5.6 SCK ⊕ and DCK ⊕ vs. the state of the art.

Our extended SCK ⊕ is trained with $3Z_2 = 15$, $Z_3 = Z_4 = 5$ and $Z_5 = 3$ while DCK ⊕ follows the same setting as DCK, except that we introduce quantity $Z_6 = 4$ which is the number of pivots encoding the subsequence position within the sequence, as dictated by Eq. (5.28). For the Florence3D-Action dataset, Table 5.1 shows that aggregating over subsequences across various scales results in SCK ⊕ outperforming SCK by ∼3.5%, DCK ⊕ outperforming DCK by ∼3.4% and the combined kernel SCK ⊕ + DCK ⊕ outperforming SCK+DCK by ∼2.2%. Table 5.2 shows the similar trend for the UTKinect-Action3D dataset, for which SCK ⊕ outperforms SCK by ∼2.4%, DCK ⊕ outperforms DCK by ∼0.6% and the combined kernel SCK ⊕ + DCK ⊕ outperforms SCK+DCK by ∼1.0%. Note that the results on UTKinect-Action3D should be considered as already saturated. Furthermore, Table 5.3 shows that on MSR-Action3D, SCK ⊕ outperforms SCK by ∼6.8%, DCK ⊕ outperforms DCK by ∼3.7% and the combined kernel SCK ⊕ + DCK ⊕ outperforms SCK+DCK by ∼7.5%.

Fine-grained Action Recognition. In what follows, we employ NTU-RGBD, a partially fine-grained dataset, and MPII Cooking Activities containing many fine-grained classes.

Our SCK ⊕ kernel is designed to capture specific subsequences of variable lengths. Kernels $G_{\sigma_2}, ..., G_{\sigma_5}$ from Section 5.4.5 capture higher-order statistics of joint locations in subsequences, the temporal alignment of pose snippets, the global alignment of subsequences, and the match of subsequence lengths. SCK ⊕ uses EPN in Eq. (5.15-5.18) which makes it act as a detector of spectral higher-order occurrences. Thus, SCK ⊕ addresses all hallmarks of modern fine-grained recognition systems: *it captures higher-order statistics* describing visual contents/objects and *discarding burstiness* [Koniusz et al., 2018b] (co-occurrence detection).

Moreover, our SCK ⊕ kernel captures higher-order occurrences of features representing spatio-temporal evolution of skeletons (for 3D body-joints) and/or frame-based classifier scores (semantic information) by feeding them into kernels $G_{\sigma_2^{(1)}}, ..., G_{\sigma_2^{(Q)}}$ from Eq. (5.23) for Q modalities.

Table 5.4 (top) shows that, our SCK ⊕ yields some ∼3.6% improvement over SCK and reaches 72.82% accuracy on the NTU-RGBD dataset in the cross- subject setting

	-	+IDT	+sec-ord	+sec-ord +IDT
Two-stream+AP (VGG-19)	38.1%	-	-	-
Two-stream+AP (ResNet-152)	45.3%	-	-	-
Subsequences+AP (I3D)	52.7%	-	-	-
HOK [Cherian et al., 2017b] (VGG-16)	60.1%	-	69.1%	73.1%
SCK ⊕ (VGG-19)	70.1%	-	74.0%	76.1%
SCK ⊕ (ResNet-152)	71.4%	-	75.5%	77.4%
SCK ⊕ (I3D)	77.8%	**80.4**%	-	-

KRP-FS 70.0% [Cherian et al., 2018] (VGG-19) KRP-FS+IDT 76.1% [Cherian et al., 2018] (VGG-19)
GRP 68.4% [Cherian et al., 2017a] (VGG-19) GRP+IDT 75.5% [Cherian et al., 2017a] (VGG-19)

Table 5.6: Results (mAP%) for (*top*) our HOK [Cherian et al., 2017b] and improved SCK ⊕. We also indicate results on the baseline two-stream network with standard average pooling (*AP*). We indicate backbones in parentheses. (*bottom*) The state of the art on MPII Cooking Activities.

	sp1	*sp2*	*sp3*	mean acc.
Two-stream+AP (ResNet-152)	65.30%	62.20%	62.55%	63.35%
Two-stream+MP (ResNet-152)	61.38%	60.58%	60.06%	60.66%
SCK ⊕ (ResNet-152)	72.55%	70.85%	71.63%	71.67%
SCK ⊕ (ResNet-152)+IDT	74.20%	73.73%	73.40%	73.77%
SCK ⊕ (r=2) (I3D)+IDT	85.61%	84.54%	85.25%	85.13%
SCK ⊕ (I3D)+IDT	86.31%	85.63%	86.41%	**86.11**%

DSP 72.4% [Wang and Cherian, 2018] (ResNet-152) ShuttleNet+MIF 71.08% [Shi et al., 2017]
DSP+IDT 74.3% [Wang and Cherian, 2018] (ResNet-152) I3D 80.2% [Carreira and Zisserman, 2017]

Table 5.7: Evaluations of (*top*) our improved SCK ⊕. We also indicate results on baseline two-stream network with standard average pooling (*AP*) and maximum pooling (*MP*). We indicate backbones in parentheses. (*bottom*) The state of the art on HMDB-51.

for the 3D body-joints as input. We expect that aggregating over subsequences can encode local fine-grained motion details essential for the good performance. Similar observations hold for the cross-view setting.

Table 5.4 (middle) shows that our SCK ⊕ attains 90.11% accuracy on the NTU-RGBD dataset in the cross- subject setting on the RGB frames (classifier scores) as input. With the 3D body-joints added, results increase to 90.78%. Lastly, adding optical flow as input to our SCK ⊕ yields 91.56%. This is $\sim$ 10.0% improvement over competing methods from Table 5.4 (bottom).

Table 5.6 shows that our SCK ⊕ yields some 1.4% mAP improvement over other state-of-the-art methods [Cherian et al., 2017a, 2018] on the MPII Cooking Activities dataset. Further improvements are attained by combining SCK ⊕ with the second-order descriptor (*sec-ord*) [Cherian et al., 2018] and the IDT representation, which yields 77.4% mAP. This compares favorably with other methods in the table. We also

note that SCK$\oplus$ outperforms the HOK descriptor [Cherian et al., 2017b] which is a variant of SCK with a suboptimal linearization of an *fc* layer. Finally, applying SCK$\oplus$ over I3D-based subsequences yields state-of-the-art 80.4% mAP (we comment on the reasons below).

Video Classification. Table 5.4 confirms that the classifier scores extracted from CNNs rather than mere 3D body-joints are a more informative input for SCK$\oplus$. Thus, we perform additional evaluations on the HMDB-51 dataset. Table 5.7 (top) shows that SCK$\oplus$ and SCK$\oplus$+IDT, trained with the two-stream ResNet-152 backbone, score 71.67 and 73.77% accuracy which is on a par with other best methods listed in Table 5.7 (bottom). Furthermore, applying the I3D backbone on SCK$\oplus$ yields state-of-the-art 86.11% accuracy. We believe that training I3D on subsequences of various lengths and strides, as detailed in Section 5.5.2 (bottom), is a more discriminative strategy than average-pooling of frame-wise features in standard two-stream networks. As SCK$\oplus$ is designed to combine subsequences of various lengths and strides rather than sequences, it captures informative higher-order occurrences of multiple complementary features, and also preserves a degree of individual statistics by factoring out one variable at a time *e.g.*, see the discussion in Figure 5.3.

Kinetics-400. Table 5.5 shows that our SCK and SCK$\oplus$ are complementary to powerful networks such as ST-GCN [Yan et al., 2018]. We work with Kinetics skeletons extracted with [Cao et al., 2017] and compare our method to the baseline ST-GCN [Yan et al., 2018]. We use the standard training/evaluation protocol (but we use skeletons rather than RGB or optical flow frames). As SCK and SCK$\oplus$ are shallow representations based on higher-order aggregation, it is unrealistic to expect them to outperform deep networks. However, SCK and SCK$\oplus$ capture very different statistics compared to deep networks, being highly complementary. Table 5.5 shows that we attain 1.1% and 2.1% gain over ST-GCN alone by concatenating both representations.

Signature Lengths. Section 5.5.6 indicates the number of pivots for SCK$\oplus$ on NTU (skeleton-based experiments) to amount to $d=3Z_2+Z_3+Z_4+Z_5=15+5+5+3=28$. The unique number of coefficients in the super-symmetric tensor of order r follows the formula $\binom{d+r-1}{r}$ discussed just below Eq. (5.11). As we obtain a tensor per joint, and we concatenate unique parts of tensors $j=1,...,J$, we have $\binom{d+r-1}{r}\cdot J$ coefficients in total in our representation. For SCK$\oplus$ on NTU with $J=25$ body joints, we obtain $4060\times 25 = 101500$ coefficients for SCK$\oplus$. For SCK and SCK ($r=2$) on NTU, we set $d=3Z_2+Z_3=24+5=29$ and obtain 112375 and 10875 coefficients, respectively. For Kinetics skeletons with $J=18$ body joints, OpenPose returns only two Cartesian coordinates, so we set $d=2Z_2+Z_3+Z_4+Z_5=20+5+5+3=33$ which yields $4545\times 18=117810$ coefficients.

For SCK$\oplus$ (NTU) on (i) RGB frames and (ii) RGB frames+optical flow, we obtain $d=Z_2+Z_3+Z_4+Z_5=60+5+5+3=73$ and $d=2Z_2+Z_3+Z_4+Z_5=73$ (for the latter case, we reduce the size of vector of classifier scores $2\times$ by the PCA). As we do not use any body joints here, we obtain 67525 coefficients. When we concatenate these representations with the skeleton-based one, we obtain $67525+101500=169025$ coefficients per video.

On SCK$\oplus$ given MPII and HMDB-51 datasets, we obtain 171700 and 125580 coefficients after reducing the size of vectors of RGB frame-wise and optical flow classifier scores from 2×64 to 100 and from 2×51 to 90, respectively.

Parameters in SCK$\oplus$. The main parameters shared between SCK and SCK$\oplus$ are evaluated in Figures 5.7 and 5.8. The parameters for SCK$\oplus$ that we start with are indicated in Section 5.4.5 (bottom). To select the best parameters, we cross-validate one parameter at a time while keeping the rest fixed. For NTU, we aggregated over subsequence lengths (using the Matlab notation) of 14:1:110, 14:2:110, 14:4:110 and 14:6:110, and we obtained 73.10%, 72.82%, 72.41% and 71.65% accuracy, respectively. For subsequence lengths $30:2:110$ and $30:2:80$, we obtained 72.54% and 72.12% accuracy. These evaluations show that SCK$\oplus$ is not overly sensitive to its parameters. For smaller skeleton-based datasets, we aggregate subsequences in range $6:2:24$, whereas on HMDB-51 we use 6:8:62, and for MPII we use 48:16:96.

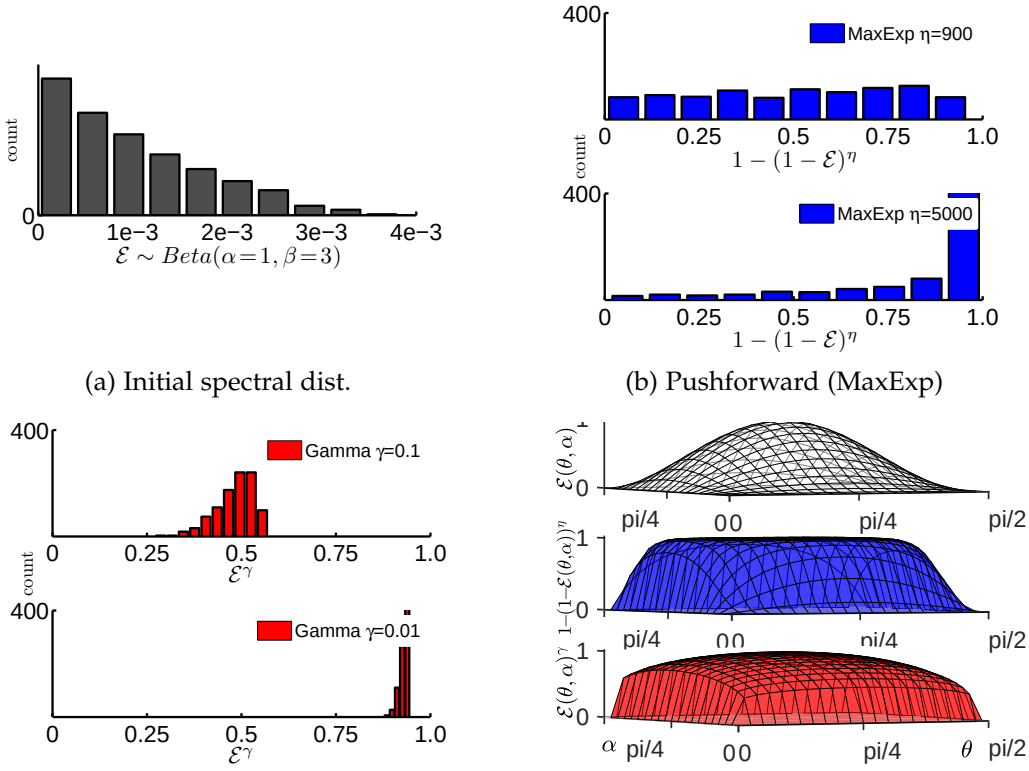
Processing Time. For SCK/DCK, processing a sequence with unoptimized MATLAB code on a single i5 core takes 0.2s and 1.2s, respectively. For SCK$\oplus$ / DCK$\oplus$, processing one sequence takes 0.5s and 3.0s. Training on full MSR-Action3D with the SCK+DCK takes about 13 min, whereas with the SCK$\oplus$ + DCK$\oplus$, it takes about 35 min. In comparison, extracting $SE(3)$ features [Vemulapalli et al., 2014] takes 5.3s per sequence, processing on the full MSR-Action3D dataset takes $\sim$50 min., whereas with post-processing (time warping and Fourier pyramids) it takes about 72 min. Thus, SCK+DCK is $\sim5.4\times$ faster while SCK$\oplus$ + DCK$\oplus$ is $\sim2\times$ faster. Section 5.8 contains the computational complexity analysis.

5.6 Linearizing Dynamics Compatibility Kernel

In what follows, we derive the linearization of DCK. Let us recall that $G_\sigma(\mathbf{u}-\bar{\mathbf{u}}) = \exp(-\|\mathbf{u}-\bar{\mathbf{u}}\|_2^2/2\sigma^2)$, $G'_\sigma(\boldsymbol{\alpha},\boldsymbol{\beta}) = G_\sigma(\boldsymbol{\alpha})G_\sigma(\boldsymbol{\beta})$ and $G_\sigma(\mathbf{i}-\boldsymbol{j}) = \delta(\mathbf{i}-\boldsymbol{j})$ if $\sigma\to0$, therefore $\delta(\mathbf{0}) = 1$ and $\delta(\mathbf{u}) = 0$ if $\mathbf{u}\neq\mathbf{0}$. Moreover, $\Lambda = J^2$ is a normalization constant and $\mathcal{J} = \mathcal{I}_J\times\mathcal{I}_N$. We recall that kernel $G_{\sigma'_2}(\mathbf{x}-\mathbf{y}) \approx \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$ while $G_{\sigma'_3}(\frac{s-t}{N}) \approx \mathbf{z}(s/N)^T\mathbf{z}(t/N)$. Thus, we obtain Eq. (5.29) which expresses $K_D(\Pi_A,\Pi_B)$ as a sum over dot-products on third-order non-symmetric tensors. We introduce operator $\boldsymbol{\mathcal{G}}$ into Eq. (5.29) to amend the dot-product with a distance which handles burstiness. We obtain a modified kernel in Eq. (5.30) based on which the following notation is introduced:

$$\begin{aligned}&\boldsymbol{\mathcal{V}}_{ii'} = \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}_{ii'}), \text{ where} \qquad (5.31)\\ &\boldsymbol{\mathcal{X}}_{ii'} = \frac{1}{\sqrt{\Lambda}}\sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}} G_{\sigma'_4}(s-s')\left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}\big(\frac{s}{N}\big)^T\right)\uparrow\otimes\mathbf{z}\big(\frac{s'}{N}\big),\end{aligned}$$

and the summation over the pairs of body-joints in Eq. (5.30) is replaced by the concatenation along the fourth mode to obtain representations $\big[\boldsymbol{\mathcal{V}}_{ii'}\big]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}$ and $\big[\tilde{\boldsymbol{\mathcal{V}}}_{ii'}\big]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}$ for Π_A and Π_B. Thus, K^*_D becomes:



(a) Initial spectral dist. (b) Pushforward (MaxExp)

(c) Pushforward (Gamma) (d) Spectral detectors

Figure 5.9: The intuitive principle of the EPN. Given a discrete spectrum following a Beta distribution in Fig. 5.9a, the pushforward measures by MaxExp and Gamma in Fig. 5.9b and 5.9c are very similar for large η (and small γ). Note that both EPN functions in bottom plots whiten the spectrum (map most values to be close to 1) thus removing burstiness. Fig. 5.9d illustrates the principle of detecting higher-order occurrence(s) in one of $\binom{Z_*}{r}$ subspaces represented by $\mathcal{E}_{\mathbf{u},\mathbf{v},w}$ (we write $\mathcal{E}$ for simplicity). Fig. 5.9d (*top*) No EPN: $\mathcal{E}(\theta,\alpha)$, (*middle*) MaxExp: $1-(1-\mathcal{E}(\theta,\alpha))^{\eta}$ and (*bottom*) Gamma: $\mathcal{E}(\theta,\alpha)^{\gamma}$. Note how MaxExp/Gamma reach high detection values close to borders. Refer Section 5.9 for def. of $\mathcal{E}(\theta,\alpha)$.

$$K_D^*(\Pi_A,\Pi_B) = \left\langle \sqrt{2}\left[\boldsymbol{\mathcal{V}}_{ii'}\right]_{i>i':\, i,i'\in\mathcal{I}_J}^{\oplus_4}, \sqrt{2}\left[\bar{\boldsymbol{\mathcal{V}}}_{ii'}\right]_{i>i':\, i,i'\in\mathcal{I}_J}^{\oplus_4} \right\rangle \tag{5.32}$$

As Eq. (5.32) suggests, we avoid repeating the same evaluations in our representations: we stack only unique pairs of body-joints $i>i'$. Moreover, we ensure we run computations temporally only for $s>s'$. In practice, we have to evaluate only $\binom{JN}{2}$ unique spatio-temporal pairs in Eq. (5.32) rather than naive J^2N^2 per sequence. The final representation is of $Z_2'\cdot\binom{JZ_3'}{2}$ size, where Z_2' and Z_3' are the numbers of pivots for approximation of $G_{\sigma_2'}$ and $G_{\sigma_3'}$.

We assume that all sequences have N frames for simplification of presentation. Our formulations are equally applicable to sequences of arbitrary lengths *e.g.*, M and N. Thus, we apply in practice $G'_{\sigma_3'}(\frac{s}{M}-\frac{t}{N}, \frac{s'}{M}-\frac{t'}{N})$ and $\Lambda=J^2MN$ in Eq. (5.29).

Moreover, a displacement between any pair of joints $\mathbf{x},\mathbf{y}\in\mathbb{R}^3$ lies within the

Cartesian coordinate system, thus $\mathbf{x}-\mathbf{y}\in\mathbb{R}^3$. In practice, in place of generic $G_{\sigma_2'}$, we use the sum kernel $G'_{\sigma_2'}(\mathbf{x}-\mathbf{y})=G_{\sigma_2'}(x_1-y_1)+G_{\sigma_2'}(x_2-y_2)+G_{\sigma_2'}(x_3-y_3)$ so the kernel $G'_{\sigma_2'}(\mathbf{x}-\mathbf{y})\approx[\phi(x_1);\phi(x_2);\phi(x_3)]^T[\phi(y_1);\phi(y_2);\phi(y_3)]$. However, for the simplicity of notation, we refer to it in our formulations by its generic form $G_{\sigma_2'}(\mathbf{x}-\mathbf{y})\approx\phi(\mathbf{x})^T\phi(\mathbf{y})$, as we can simply define $\phi(\mathbf{x})=[\phi(x_1);\phi(x_2);\phi(x_3)]$.

5.7 Positive Definiteness of SCK and DCK

SCK/DCK are sums over products of RBF subkernels. According to [Shawe-Taylor and Cristianini, 2004], sums, products and linear combinations (for non-negative weights) of positive definite kernels yield positive definite kernels. Moreover, subkernel $G_{\sigma_2'}\left((\mathbf{x}_{is}-\mathbf{x}_{i's'})-(\mathbf{y}_{jt}-\mathbf{y}_{j't'})\right)$ employed by DCK in Eq. (5.29) (top) can be rewritten as:

$$G_{\sigma_2'}\left(\mathbf{z}_{isi's'}-\mathbf{z}'_{jtj't'}\right), \quad \text{where } \mathbf{z}_{isi's'}=\mathbf{x}_{is}-\mathbf{x}_{i's'} \text{ and } \mathbf{z}'_{jtj't'}=\mathbf{y}_{jt}-\mathbf{y}_{j't'}. \tag{5.33}$$

The RBF kernel $G_{\sigma_2'}$ is positive definite (PD) by definition and the mappings from $\mathbf{x}_{is}$ and $\mathbf{x}_{i's'}$ to $\mathbf{z}_{isi's'}$ and from $\mathbf{y}_{jt}$ and $\mathbf{y}_{j't'}$ to $\mathbf{z}'_{jtj't'}$, respectively, are unique. Thus, the entire kernel is PD.

Whitening on SCK results in a positive (semi)definite (PSD) kernel as we employ the Power-Euclidean kernel *e.g.*, if $\mathbf{X}$ is PD then $\mathbf{X}^\gamma$ stays also PD for $0<\gamma\leq 1$ because $\mathbf{X}^\gamma=\mathbf{U}\boldsymbol{\lambda}^\gamma\mathbf{V}$ and element-wise rising of eigenvalues to the power of γ gives us $\text{daig}(\boldsymbol{\lambda})^\gamma\geq 0$. Thus, the sum over dot-products of positive (semi)definite matrices raised to the power of γ stays PSD/PD.

5.8 Computational Complexity

Non-linearized SCK with ker. SVM have complexity $\mathcal{O}(JN^2T^\rho)$ given J body joints, N frames per sequence, T sequences, and $2<\rho<3$ which concerns complexity of kernel SVM. Linearized SCK with linear SVM takes $\mathcal{O}(JNTZ_*^r)$ for total of Z_* pivots and tensor of order $r=3$. Note that $N^2T^\rho\gg NTZ_*^r$. For $N=50$ and $Z_*=20$, this is 3.5× (or 32×) faster than the exact kernel for $T=557$ (or $T=5000$) used in our experiments.

Non-linearized DCK+kernel SVM enjoys $\mathcal{O}(J^2N^4T^\rho)$ complexity. Linearized DCK+SVM enjoys $\mathcal{O}(J^2N^2TZ^3)$ for Z pivots per kernel, *e.g.* $Z=Z_2=Z_3$ given $G_{\sigma_2'}$ and $G_{\sigma_3'}$. As $N^4T^\rho\gg N^2TZ^3$, the linearization is 11000× faster than the exact kernel, for say $Z=5$. Slice-wise EPN applied to SCK has negligible cost $\mathcal{O}(JTZ_*^{\omega+1})$, where $2<\omega<2.376$ concerns complexity of eigenvalue decomposition applied per tensor slice.

Note that EPN incurs negligible cost (see [Koniusz et al., 2016b] for details). EPN applied to DCK utilizes HOSVD and results in complexity $\mathcal{O}(J^2TZ^4)$. As HOSVD is performed by truncated SVD on matrices obtained from unfolding $\boldsymbol{\mathcal{V}}_{ii'}\in\mathbb{R}^{Z\times Z\times Z}$

along a chosen mode, $\mathcal{O}(Z^4)$ represents the complexity of truncated SVD on matrices $\mathbf{V}_{ii'} \in \mathbb{R}^{Z \times Z^2}$ which have rank less than or equal Z.

Linearized SCK $\oplus$ with linear SVM also takes $\mathcal{O}(JNTZ_*^r)$ for a total of Z_*. However, $Z_* = 3Z_2 + Z_3 + Z_4 + Z_5$ thus $Z_* = 28$. The linearized DCK $\oplus$ takes $\mathcal{O}(J^2N^2TZ^3Z_6)$ where $Z_6 = 4$ in our experiments. EPN applied to SCK $\oplus$ and DCK $\oplus$ results in complexity $\mathcal{O}(JTZ_*^{2(r-1)})$ and $\mathcal{O}(J^2TZ^4Z_6)$.

5.9 What is (Tensor) Eigenvalue Power Normalization?

Below, we show that EPN in fact retrieves factors which quantify whether there is at least one datapoint $\boldsymbol{\phi}(\mathbf{x}_n)$ from $n \in \mathcal{I}_N$ projected into each subspace spanned by r-tuples of eigenvectors from matrices $\boldsymbol{A}_1 = \boldsymbol{A}_2 = ... = \boldsymbol{A}_r$. For brevity, assume order $r = 3$, a super-symmetric case, and a 3-tuple of eigenvectors $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$ from $\boldsymbol{A}$. Note that $\mathbf{u} \perp \mathbf{v}, \mathbf{v} \perp \boldsymbol{w}$ and $\mathbf{u} \perp \boldsymbol{w}$. Moreover, note that if we have Z_* unique eigenvectors, we can enumerate $\binom{Z_*}{r}$ r-tuples and thus $\binom{Z_*}{r}$ subspaces $\mathbb{R}^r \subset \mathbb{R}^{Z_*}$. For brevity, let $||\boldsymbol{\phi}(\mathbf{x})||_2 = 1$ and $\boldsymbol{\phi}(\mathbf{x}) \geq 0$. Also, we write $\boldsymbol{\phi}_n$ instead of $\boldsymbol{\phi}(\mathbf{x})$ for $n \in \mathcal{I}_N$. Next, let us write our super-symmetric tensor as:

$$\boldsymbol{\mathcal{X}} = \frac{1}{N} \sum_{n \in \mathcal{I}_N} \uparrow \otimes_r \boldsymbol{\phi}_n, \tag{5.34}$$

and the 'diagonalization' of $\boldsymbol{\mathcal{X}}$ w.r.t. by eigenvec. $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$ as:

$$\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}} = ((\boldsymbol{\mathcal{X}} \otimes_1 \mathbf{u}) \otimes_1 \mathbf{v}) \otimes_3 \boldsymbol{w}, \tag{5.35}$$

where $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}$ is a coefficient from the core tensor $\boldsymbol{\mathcal{E}}$ for eigenvectors $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$. Now, we combine Eq. 5.34 and 5.35 and obtain:

$$\begin{aligned} \mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}} &= \left(\left((\frac{1}{N} \sum_{n \in \mathcal{I}_N} \uparrow \otimes_3 \boldsymbol{\phi}_n) \otimes_1 \mathbf{u} \right) \otimes_2 \mathbf{v} \right) \otimes_3 \boldsymbol{w} \\ &= \frac{1}{N} \sum_{n \in \mathcal{I}_N} \langle \boldsymbol{\phi}_n, \mathbf{u} \rangle \langle \boldsymbol{\phi}_n, \mathbf{v} \rangle \langle \boldsymbol{\phi}_n, \boldsymbol{w} \rangle \end{aligned} \tag{5.36}$$

We assume $\boldsymbol{\phi}_n$ is projected into subspace spanned by $\mathbf{u}, \mathbf{v}$ and $\boldsymbol{w}$ when $\psi'_n = \langle \boldsymbol{\phi}_n, \mathbf{u} \rangle \langle \boldsymbol{\phi}_n, \mathbf{v} \rangle \langle \boldsymbol{\phi}_n, \boldsymbol{w} \rangle$ is maximized. As our $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$ are orthogonal w.r.t. each other and $||\boldsymbol{\phi}_n||_2 = 1$, simple Lagrange eq. $\mathcal{L} = \Pi_{i=1}^r \mathbf{e}_i^T \boldsymbol{\phi}_n + \lambda(||\boldsymbol{\phi}_n||_2^2 - 1)$ yields maximum of $\kappa = (1/\sqrt{r})^r$ at $\boldsymbol{\phi}_n = [(1/\sqrt{r}), ..., (1/\sqrt{r})]^T$. For each $n \in \mathcal{I}_N$, we store $\psi_n = \psi'_n / \kappa$ in vector $\boldsymbol{\psi}$.

Assume that $\boldsymbol{\psi} \in \{0,1\}^N$ stores N outcomes of drawing from Bernoulli distribution under the i.i.d. assumption for which the probability p of an event $(\psi_n = 1)$ and $1-p$ for $(\psi_n = 0)$ is estimated as an expected value, $p = \text{avg}_n\, \psi_n$ (even if $0 \leq \boldsymbol{\psi} \leq 1$ in reality). Then the probability of at least one projection event $(\psi_n = 1)$ into the subspace spanned by r-tuples in N trials becomes:

$$\hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},\boldsymbol{w}} = 1-(1-p)^N = 1-\left(1-\frac{\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}}{\kappa}\right)^N \approx \left(\frac{\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}}{\kappa}\right)^\gamma. \tag{5.37}$$

Thus, each of $\binom{Z_*}{r}$ subspaces spanned by r-tuples acts as a detector of projections into this subspace. The middle part of Eq. (5.37) (so-called MaxExp pooling) and its connection to the right-hand part of Eq. (5.37) (so-called Gamma) are detailed in [Koniusz et al., 2018b]. In fact, our $\boldsymbol{\psi}$ can be negative so extending Eq. (5.37) to $\text{Sgn}(\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}})\left(1-(1-\frac{|\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}|}{\kappa})^{N+\eta}\right)$ makes our model a detector of asymmetry between projections into 'positive' and 'negative' parts of each subspace, and η compensates for non-binary $\boldsymbol{\psi}$.

Figure 5.9 illustrates that MaxExp and Gamma are in fact very similar. Figure 5.9a shows an initial Beta distribution of spectrum. Figures 5.9b and 5.9c (bottom) show that for sufficiently large parameters η and γ, both MaxExp and Gamma shift most of the distribution values to be approximately equal 1. Figure 5.9c illustrates the effect of EPN on eigenvalue $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}$ (denoted as $\mathcal{E}$ for simplicity) representing a single subspace spanned by eigenvectors $\mathbf{u}, \mathbf{v}, \boldsymbol{w}$ such that $\mathbf{u} \perp \mathbf{v}, \mathbf{v} \perp \boldsymbol{w}$ and $\mathbf{u} \perp \boldsymbol{w}$. As a single projection into the subspace is defined as $\psi_n = \langle \boldsymbol{\phi}_n, \mathbf{u} \rangle \langle \boldsymbol{\phi}_n, \mathbf{v} \rangle \langle \boldsymbol{\phi}_n, \boldsymbol{w} \rangle / \kappa$, we note this is the product of three projections of $\boldsymbol{\phi}_n$ onto $\mathbf{u}, \mathbf{v}, \boldsymbol{w}$, respectively, measured by the cosine (dot-product). Thus, we parametrize such a projection by the spherical coordinates, that is:

$$\begin{aligned} &\pi_{\mathbf{u}}(\theta,\alpha) = \cos(\theta)\cdot\sin(\alpha),\ \pi_{\mathbf{v}}(\theta,\alpha) = \sin(\theta)\cdot\sin(\alpha), \\ &\pi_{\mathbf{u}}(\alpha) = \cos(\alpha), \end{aligned} \tag{5.38}$$

where the azimuthal coordinate θ runs from 0 to 2π and the polar coordinate α runs from 0 to π. We rewrite the projection as:

$$\pi_{\mathbf{u},\mathbf{v},\boldsymbol{w}}(\theta,\alpha) = \pi_{\mathbf{u}}(\theta,\alpha)\cdot\pi_{\mathbf{v}}(\theta,\alpha)\cdot\pi_{\boldsymbol{w}}(\alpha)/\kappa. \tag{5.39}$$

We note that $\pi_{\mathbf{u},\mathbf{v},\boldsymbol{w}}(\theta,\alpha)$ and ψ_n are isomorphic as $||\boldsymbol{\phi}_n||_2 = 1$, thus it suffices to note $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}} \sim \pi_{\mathbf{u},\mathbf{v},\boldsymbol{w}}(\theta,\alpha)$ and show the EPN pushforward output of $\mathcal{E}$ to understand how EPN behaves around the boundaries of the spanning vectors $\mathbf{u}, \mathbf{v}, \boldsymbol{w}$. Figure 5.9d (top) shows that $\mathcal{E}$ by itself has a weak response in the proximity of the spanning vectors $\mathbf{u}, \mathbf{v}, \boldsymbol{w}$. However, MaxExp and Gamma in Figures 5.9d middle and bottom manage to boost projections in the proximity of the spanning vectors in the similar manner to each other, both behaving like spectral detectors.

To conclude, let us consider the dot-product between Power Normalized tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ computed according to Eq. (5.15-5.17). Then:

$$\left\langle \hat{\mathcal{V}}(\mathcal{X}), \hat{\mathcal{V}}(\mathcal{Y}) \right\rangle = \left\langle \sum_{\substack{\mathbf{u}\in\mathbf{U}(\mathcal{X}) \\ \mathbf{v}\in\mathbf{V}(\mathcal{X}) \\ w\in W(\mathcal{X})}} \hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},w} \mathbf{u}\mathbf{v}^T \uparrow \otimes w, \sum_{\substack{\mathbf{u}'\in\mathbf{U}(\mathcal{Y}) \\ \mathbf{v}'\in\mathbf{V}(\mathcal{Y}) \\ w'\in W(\mathcal{Y})}} \hat{\mathcal{E}}'_{\mathbf{u}',\mathbf{v}',w'} \mathbf{u}'\mathbf{v}'^T \uparrow \otimes w' \right\rangle$$
$$= \sum_{\substack{\mathbf{u}\in\mathbf{U}(\mathcal{X}) \\ \mathbf{v}\in\mathbf{V}(\mathcal{X}) \\ w\in W(\mathcal{X})}} \sum_{\substack{\mathbf{u}'\in\mathbf{U}(\mathcal{Y}) \\ \mathbf{v}'\in\mathbf{V}(\mathcal{Y}) \\ w'\in W(\mathcal{Y})}} \hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},w} \hat{\mathcal{E}}'_{\mathbf{u}',\mathbf{v}',w'} \left\langle \mathbf{u}, \mathbf{u}' \right\rangle \left\langle \mathbf{v}, \mathbf{v}' \right\rangle \left\langle w, w' \right\rangle . \tag{5.40}$$

Eq.(5.40) shows that all subspaces of $\mathcal{X}$ and $\mathcal{Y}$ spanned by r-tuples (3-tuples in this example) are compared against each other for alignment by the cosine distance. When two subspaces $[\mathbf{u}\,\mathbf{v}\,w]^T$ and $[\mathbf{u}'\,\mathbf{v}'\,w']^T$ are aligned, for a strong similarity between these subspaces, a detection of at least one $\boldsymbol{\phi}_n$ and $\boldsymbol{\phi}'_n$ evidenced by $\hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},w}$ and $\hat{\mathcal{E}}'_{\mathbf{u}',\mathbf{v}',w'}$ is also needed. We term Eq. (5.40) together with Eq. (5.15-5.17) as Tensor Power Euclidean dot-product which has the associated Tensor Power Euclidean metric $||\mathcal{X}-\mathcal{Y}||_{\mathcal{T}} = ||\hat{\mathcal{V}}(\mathcal{X}) - \hat{\mathcal{V}}(\mathcal{Y})||_F$.

5.10 Conclusions

We have presented two kernel-based tensor representations, namely the sequence compatibility kernel (SCK) and dynamics compatibility kernel (DCK). SCK captures the higher-order correlations between 3D coordinates of the body-joints and their temporal variations. As SCK factors out the temporal variable, expensive Fourier temporal pyramid matching/dynamic time warping are not needed. Further, our DCK kernel captures the action dynamics by modeling the spatio-temporal co-occurrences of the body-joints.

Additionally, we have presented a highly effective extension of SCK, termed SCK $\oplus$, which aggregates over subsequences of multiple lengths, focusing on actions within subsequences. We have demonstrated that SCK $\oplus$ can aggregate over 3D body-joints and/or frame-wise classifier scores from CNNs to capture higher-order statistics between various features extracted from body-skeletons, classifier scores, and temporal positions.

Section 5.9 shows that (Tensor) Eigenvalue Power Normalization indeed acts as a spectrum-based metric with $\binom{Z_*}{r}$ subspace-based detectors of higher-order occurrence of datapoints of dim. Z_*, more specifically, detectors that capture asymmetry of projections into 'positive' and 'negative' parts of each subspace. As $\binom{Z_*}{3} \gg \binom{Z_*}{2}$, third-order tensors capture more dependencies than autocorrelation matrices, improving fine-grained systems.

Chapter 6

3Mformer: Multi-order Multi-mode Transformer

Many skeletal action recognition models use GCNs to represent the human body by 3D body joints connected body parts. GCNs aggregate one- or few-hop graph neighbourhoods, and ignore the dependency between not linked body joints. We propose to form hypergraph to model hyper-edges between graph nodes (*e.g.*, third- and fourth-order hyper-edges capture three and four nodes) which help capture higher-order motion patterns of groups of body joints. We split action sequences into temporal blocks, Higher-order Transformer (HoT) produces embeddings of each temporal block based on (i) the body joints, (ii) pairwise links of body joints and (iii) higher-order hyper-edges of skeleton body joints. We combine such HoT embeddings of hyper-edges of orders $1, ..., r$ by a novel Multi-order Multi-mode Transformer (3Mformer) with two modules whose order can be exchanged to achieve coupled-mode attention on coupled-mode tokens based on 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' and 'channel-only' pairs. The first module, called Multi-order Pooling (MP), additionally learns weighted aggregation along the hyper-edge mode, whereas the second module, Temporal block Pooling (TP), aggregates along the temporal block[1] mode. Our end-to-end trainable network yields state-of-the-art results compared to GCN-, transformer- and hypergraph-based counterparts.

6.1 Introduction

Action Recognition has applications in video surveillance, human-computer interaction, sports analysis, and virtual reality [Wang, 2017; Wang et al., 2019c,b,d; Koniusz et al., 2020, 2021; Wang and Koniusz, 2021; Wang et al., 2021b; Qin et al., 2022; Wang and Koniusz, 2022b,a]. Different from video-based methods which mainly focus on modeling the spatio-temporal representations from RGB frames and/or optical flow [Wang, 2017; Wang et al., 2019c,b,d; Wang and Koniusz, 2021; Koniusz et al., 2021], skeleton sequences, representing a spatio-temporal evolution of 3D body joints, have been proven robust against sensor noises and effective in action recognition

[1]For brevity, we write τ temporal blocks per sequence but τ varies.

while being computationally and storage efficient [Wang, 2017; Wang et al., 2019b; Koniusz et al., 2020; Wang et al., 2021b; Qin et al., 2022; Wang and Koniusz, 2022b,a]. The skeleton data is usually obtained by either localization of 2D/3D coordinates of human body joints with the depth sensors or pose estimation algorithms applied to videos [Cao et al., 2017]. Skeleton sequences enjoy (i) simple structural connectivity of skeletal graph and (ii) temporal continuity of 3D body joints evolving in time. While temporal evolution of each body joint is highly informative, embeddings of separate body joints are insensitive to relations between body parts. Moreover, while the links between adjacent 3D body joints (following the structural connectivity) are very informative as they model relations, these links represent highly correlated nodes in the sense of their temporal evolution. Thus, modeling larger groups of 3D body joints as hyper-edges can capture more complex spatio-temporal motion dynamics.

The existing graph-based models mainly differ by how they handle temporal information. Graph Neural Network (GNN) may encode spatial neighborhood of the node followed by aggregation by LSTM [Si et al., 2019; Zhang et al., 2019b]. Alternatively, Graph Convolutional Network (GCN) may perform spatio-temporal convolution in the neighborhood of each node [Yan et al., 2018]. Spatial GCNs perform convolution within one or two hop distance of each node, *e.g.*, spatio-temporal GCN model called ST-GCN [Yan et al., 2018] models spatio-temporal vicinity of each 3D body joint. As ST-GCN applies convolution along structural connections (links between body joints), structurally distant joints, which may cover key patterns of actions, are largely ignored. ST-GCN captures ever larger neighborhoods as layers are added but suffers from oversmoothing that can be mitigated by linear GCNs [Zhu and Koniusz, 2021c; Zhu et al., 2021a,a].

Human actions are associated with interaction groups of skeletal joints, *e.g.*, wrist alone, head-wrist, head-wrist-ankles, *etc*. The impact of these groups of joints on each action differs, and the degree of influence of each joint should be learned. Accordingly, designing a better model for skeleton data is vital given the topology of skeleton graph is suboptimal. While GCN can be applied to a fully-connected graph (*i.e.*, 3D body joints as densely connected graph nodes), Higher-order Transformer (HoT) [Kim et al., 2021] has been proven more efficient.

Thus, we propose to use hypergraphs with hyper-edges of order 1 to r to effectively represent skeleton data for action recognition. Compared to GCNs, our encoder contains an MLP followed by three HoT branches that encode first-, second- and higher-order hyper-edges, *i.e.*, set of body joints, edges between pairs of nodes, hyper-edges between triplets of nodes, *etc*. Each branch has its own learnable parameters, and processes temporal blocks[2] one-by-one.

We notice that (i) the number of hyper-edges of J joints grows rapidly with order r, *i.e.*, $\binom{J}{i}$ for $i = 1, ..., r$, embeddings of the highest order dominate lower orders in terms of volume if such embeddings are merely concatenated, and (ii) long-range temporal dependencies of feature maps are insufficiently explored, as sequences are split into τ temporal blocks for computational tractability.

[2]Each temporal block enjoys a locally factored out (removed) temporal mode, which makes each block representation compact.

Merely concatenating outputs of HoT branches of orders 1 to r, and across τ blocks, is sub-optimal. Thus, our Multi-order Multi-mode Transformer (3Mformer) with two modules whose order can be exchanged, realizes a variation of coupled-mode tokens based on 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' and 'channel-only' pairs. As HoT operates block-by-block, 'channel-temporal block' tokens and weighted hyper-edge aggregation in Multi-order Pooling (MP) help combine information flow block-wise. Various coupled-mode tokens help improve results further due to different focus of each attention mechanism. As the block-temporal mode needs to be aggregated (number of blocks varies across sequences), Temporal block Pooling (TP) can use rank pooling [Fernando et al., 2017], second-order [Lin et al., 2018; Zilin et al., 2019; Wang et al., 2018a; Girdhar and Ramanan, 2017; Zhang et al., 2020d; Koniusz and Zhang, 2020; Rahman et al., 2023] or higher-order pooling [Cherian et al., 2017b; Koniusz et al., 2021, 2020; Zhang et al., 2022d,c].

In summary, our main contributions are listed as follows:

i. We model the skeleton data as hypergraph of orders 1 to r (set, graph and/or hypergraph), where human body joints serve as nodes. Higher-order Transformer embeddings of such formed hyper-edges represent various groups of 3D body joints and capture various higher-order dynamics important for action recognition.

ii. As HoT embeddings represent individual hyper-edge order and block, we introduce a novel Multi-order Multi-mode Transformer (3Mformer) with two modules, Multi-order Pooling and Temporal block Pooling. Their goal is to form coupled-mode tokens such as 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' and 'channel-only', and perform weighted hyper-edge aggregation and temporal block aggregation.

Our 3Mformer outperforms other GCN- and hypergraph-based models on NTU-60, NTU-120, Kinetics-Skeleton and Northwestern-UCLA by a large margin.

6.2 Related Work

Below we describe popular action recognition models for skeletal data.

Graph-based models. Popular GCN-based models include the Attention enhanced Graph Convolutional LSTM network (AGC-LSTM) [Si et al., 2019], the Actional-Structural GCN (AS-GCN) [Li et al., 2019], Dynamic Directed GCN (DDGCN) [Korban and Li, 2020], Decoupling GCN with DropGraph module [Cheng et al., 2020a], Shift-GCN [Cheng et al., 2020b], Semantics-Guided Neural Networks (SGN) [Zhang et al., 2020c], AdaSGN [Shi et al., 2021a], Context Aware GCN (CA-GCN) [Zhang et al., 2020e], Channel-wise Topology Refinement Graph Convolution Network (CTR-GCN) [Chen et al., 2021b] and a family of Efficient GCN (EfficientGCN-Bx) [Song et al., 2022]. Although GCN-based models enjoy good performance, they have shortcomings, *e.g.*, convolution and/or pooling are applied over one- or few-hop neighborhoods, *e.g.*,

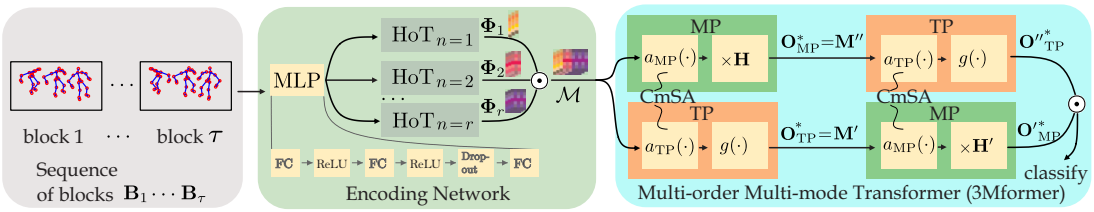
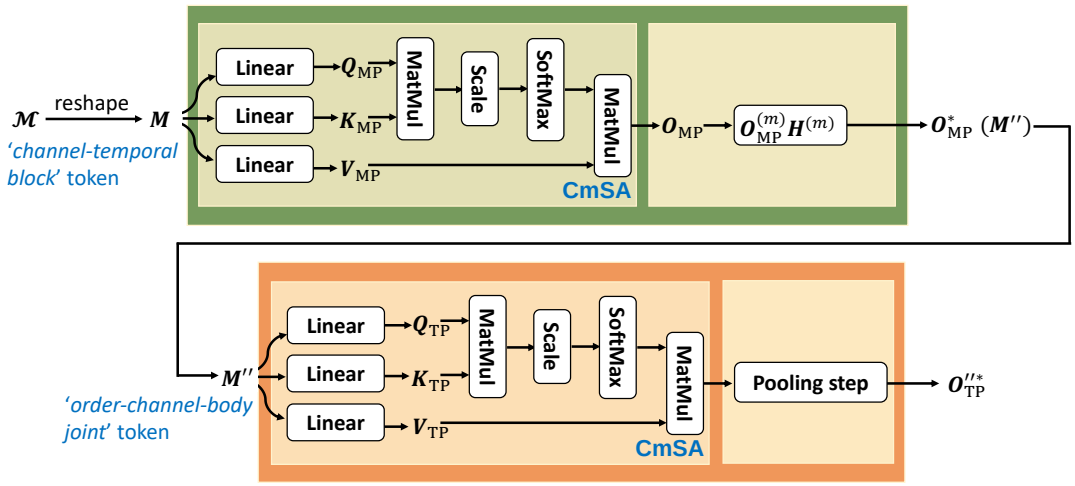


Figure 6.1: Pipeline overview. Each sequence is split into τ temporal blocks $\mathbf{B}_1, ..., \mathbf{B}_\tau$. Subsequently, each block is embedded by a simple MLP into $\mathbf{X}_1, ..., \mathbf{X}_\tau$, which are passed to Higher-order Transformers (HoT ($n = 1, ..., r$)) in order to obtain feature tensors $\mathbf{\Phi}_1, ..., \mathbf{\Phi}_\tau$. These tensors are subsequently concatenated by $\odot$ along the hyper-edge mode into a multi-order feature tensor $\mathcal{M}$. The final step is a Multi-order Multi-mode Transformer (3Mformer from Section 6.4), which contains two complementary branches, MP→TP and TP→MP, whose outputs are concatenated by $\odot$ and passed to the classifier. MP and TP perform the Coupled-mode Self-Attention (CmSA) with the so-called coupled-mode tokens, based on 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge' and 'channel-only' pairs. To this end, MP contains also weighted pooling along hyper-edge mode by learnable matrix $\mathbf{H}$ (and $\mathbf{H}'$ in another branch). TP contains also block-temporal pooling denoted by $g(\cdot)$ whose role is to capture block-temporal order with average, maximum, rank pooling, *etc*. In our experiments we show that such designed MP and TP are able to efficiently process hyper-edge feature representations from HoT branches. Section 6.4.4 shows full visualization of our 3Mformer.

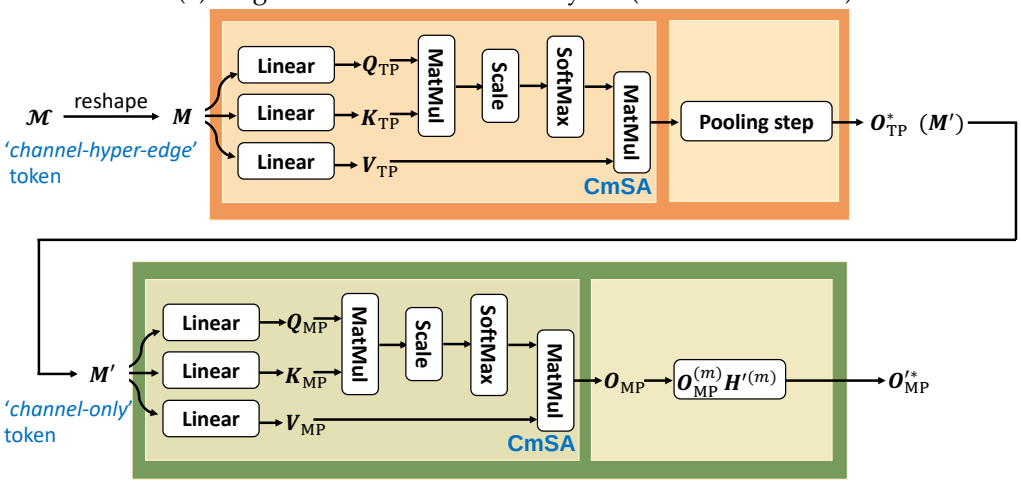
ST-GCN [Yan et al., 2018], according to the human skeleton graph (body joints linked up according to connectivity of human body parts). Thus, indirect links between various 3D body joints such as hands and legs are ignored. In contrast, our model is not restricted by the structure of typical human body skeletal graph. Instead, 3D body joints are nodes which form hyper-edges of orders 1 to r.

Hypergraph-based models. Pioneering work on capturing groups of nodes across time uses tensors [Koniusz et al., 2020] to represent the 3D human body joints to exploit the kinematic relations among the adjacent and non-adjacent joints. Representing the human body as a hypergraph is adopted in [Liu et al., 2020a] via a semi-dynamic hypergraph neural network that captures richer information than GCN. A hypergraph GNN [Hao et al., 2021] captures both spatio-temporal information and higher-order dependencies for skeleton-based action recognition. Our work is somewhat closely related to these works, but we jointly use hypergraphs of order 1 to r to obtain rich hyper-edge embeddings based on Higher-order Transformers.

Transformer-based models. Action recognition with transformers includes self-supervised video transformer [Ranasinghe et al., 2022] that matches the features from different views (a popular strategy in self-supervised GCNs [Zhang et al., 2022f, 2023]), the end-to-end trainable Video-Audio-Text-Transformer (VATT) [Akbari et al., 2021] for learning multi-model representations from unlabeled raw video, audio and text through the multimodal contrastive losses, and the Temporal Transformer Network



(a) Single-branch: MP followed by TP (denoted MP→TP).



(b) Single-branch: TP followed by MP (denoted TP→MP).

Figure 6.2: Visualization of 3Mformer which is a two- branch model: (a) MP→TP and (b) TP→MP. Green and orange blocks are Multi-order Pooling (MP) module and Temporal block Pooling (TP) module, respectively. (m) inside the MP module denotes the order $m \in \mathcal{I}_r$ of hyper-edges. These two modules (MP and TP) are the basic building blocks which are further stacked to form our 3Mformer. Each module (MP or TP) uses a specific coupled-mode token through matricization (we use reshape for simplicity), *e.g.*, 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' or 'channel-only', and the Coupled-mode Self-Attention (CmSA) is used to explore the coupled-mode relationships inside the coupled-mode tokens. We also form our multi-head CmSA as in standard Transformer (where the CmSA module repeats its computations multiple times in parallel and the attention module splits the query, key and value, each split is independently passed through a separate head and later combined together to produce the final coupled-mode attention score). We omit the multi-head visualization for simplicity and better visualization purposes.

with Self-supervision (TTSN) [Zhang et al., 2021d]. Motion-Transformer [Cheng et al., 2021] captures the temporal dependencies via a self-supervised pre-training on human actions, Masked Feature Prediction (MaskedFeat) [Wei et al., 2021a] pre-trained on unlabeled videos with MViT-L learns abundant visual representations, and video-masked autoencoder (VideoMAE) [Tong et al., 2022] with vanilla ViT uses the masking strategy. In contrast to these works, we use three HoT branches of model [Kim et al., 2021], and we model hyper-edges of orders 1 to r by forming several multi-mode token variations in 3Mformer.

Attention. In order to improve feature representations, attention captures relationship between tokens. Natural language processing and computer vision have driven recent developments in attention mechanisms based on transformers [Vaswani et al., 2017; Dosovitskiy et al., 2020]. Examples include the hierarchical Cross Attention Transformer (CAT) [Lin et al., 2021], Cross-attention by Temporal Shift with CNNs [Hashiguchi and Tamaki, 2022], Cross-Attention Multi-Scale Vision Transformer (CrossViT) for image classification [Chen et al., 2021a] and Multi-Modality Cross Attention (MMCA) Network for image and sentence matching [Wei et al., 2020]. In GNNs, attention can be defined over edges [Veličković et al., 2018; Zhang et al., 2018a] or over nodes [Lee et al., 2018]. In this work, we use the attention with hyper-edges of several orders from HoT branches serving as tokens, and coupled-mode attention with coupled-mode tokens based on 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' and 'channel-only' pairs formed in 3Mformer.

6.3 Background

Below we describe foundations necessary for our work.

Notations. $\mathcal{I}_K$ stands for the index set $\{1, 2, ..., K\}$. Regular fonts are scalars; vectors are denoted by lowercase boldface letters, *e.g.*, $\mathbf{x}$; matrices by the uppercase boldface, *e.g.*, $\mathbf{M}$; and tensors by calligraphic letters, *e.g.*, $\boldsymbol{\mathcal{M}}$. An rth-order tensor is denoted as $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_r}$, and the mode-$m$ matricization of $\boldsymbol{\mathcal{M}}$ is denoted as $\boldsymbol{\mathcal{M}}_{(m)} \in \mathbb{R}^{I_m \times (I_1 ... I_{m-1} I_{m+1} ... I_r)}$.

Skeletal Graph [Yan et al., 2018]. Let $G = (V, E)$ be a skeletal graph with the vertex set V of nodes (body joints) $\{v_1, ..., v_J\}$, and E be edges (bones) of the graph, and E consists of E_S and E_T. The subset $E_S = \{(v_{it}, v_{jt}) : i, j \in \mathcal{I}_J \text{ and } t \in \mathcal{I}_T\}$ represents that at time step t, each pair of joints (v_{it}, v_{jt}) corresponding to skeletal connectivity diagram is connected; whereas $E_T = \{(v_{it}, v_{i(t+1)}) : i \in \mathcal{I}_J \text{ and } t \in \mathcal{I}_T\}$ forms the connection of the same joint across time. The set of joints and edges together form the skeleton graph. If two body joints are connected by an edge, the corresponding element in the incidence matrix $\mathbf{H}$ is equal to 1, otherwise it is equal to 0, and the adjacency matrix $\mathbf{A} = \mathbf{H}^T\mathbf{H} - 2\mathbf{I}$ (where $\mathbf{I}$ is the identity matrix). The update rule of a common GCN model at time step t is defined as:

$$\mathbf{X}_t^{(l+1)} = \sigma\big(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}_t^{(l)}\boldsymbol{\Theta}^{(l)}\big), \tag{6.1}$$

where $\sigma(\cdot)$ is a non-linearity, $\widetilde{\mathbf{D}}$ is the graph degree matrix, $\mathbf{X}_t^{(l)}$ is the input data of the convolutional layer l at the time step t and $\boldsymbol{\Theta}^{(l)}$ is the learnable parameters of layer l. $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is a normalized graph adjacency matrix.

The tensor representation of graph data can be given by $\mathbf{X} \in \mathbb{R}^{J^2 \times d}$ where d is the feature channel dimension.

Skeletal Hypergraph [Liu et al., 2020a; Hao et al., 2021]. Hypergraph captures complex higher-order relationships by hyper-edges that connect more than two nodes (body joints). Each hyper-edge is a subset of all nodes. Let $G_h = (V_h, E_h, W_h)$ where V_h, E_h and W_h denote respectively the set of body joints, hyper-edges and the weights of hyper-edges. Given $v \in V_h$ and $e \in E_h$, the elements in the incidence matrix $\mathbf{H}_h$ of the skeleton hypergraph are defined as $H_{h,v,e} = 1$, or simply put $h(v,e) = 1$, if vertex v is part of edge e, 0 otherwise. The degree of node/body joint $v \in V_h$ is the number of hyper-edges passing through the node, which is defined as:

$$d(v) = \sum_{e \in E_h} w(e) h(v,e), \tag{6.2}$$

where $w(e)$ is the weight of hyper-edge e. The degree of hyper-edge $e \in E_h$ is the number of nodes (body joints) contained in the hyper-edge e that satisfies:

$$\delta(e) = \sum_{v \in V_h} h(v,e). \tag{6.3}$$

Moreover, let $\mathbf{D}_v$ and $\mathbf{D}_e$ be the diagonal matrices of node degrees $d(v)$ and the hyper-edge degrees $\delta(e)$ respectively. Let $\mathbf{W}$ denote the diagonal matrix of the hyper-edge weights (initially the weights of all hyper-edges are set to 1). Then the update rule of the Hypergraph Convolutional Network at the time step t is given by:

$$\mathbf{X}_t^{(l+1)} = \sigma\big(\mathbf{D}_v^{\frac{1}{2}} \mathbf{H}_h \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}_h^\top \mathbf{D}_v^{\frac{1}{2}} \mathbf{X}_t^{(l)} \boldsymbol{\Theta}^{(l)}\big), \tag{6.4}$$

where $\boldsymbol{\Theta}^{(l)}$ are learnable parameters for layer l.

Transformer layers [Vaswani et al., 2017; Dosovitskiy et al., 2020]. A transformer encoder layer $f : \mathbb{R}^{J \times d} \to \mathbb{R}^{J \times d}$ consists of two sub-layers: (i) a self-attention $a : \mathbb{R}^{J \times d} \to \mathbb{R}^{J \times d}$ and (ii) an element-wise feed-forward MLP: $\mathbb{R}^{J \times d} \to \mathbb{R}^{J \times d}$. For a set of J nodes with $\mathbf{X} \in \mathbb{R}^{J \times d}$, where $\mathbf{x}_i$ is a feature vector of node i, a transformer layer[3] computes:

$$a(\mathbf{x}_i) = \mathbf{x}_i + \sum_{h=1}^{H} \sum_{j=1}^{J} \alpha_{ij}^h \mathbf{x}_j \mathbf{W}_h^V \mathbf{W}_h^O, \tag{6.5}$$

$$f(\mathbf{x}_i) = a(\mathbf{x}_i) + \mathrm{MLP}(a(\mathbf{X}))_i, \tag{6.6}$$

where H and d_H denote respectively the number of heads and the head size, $\boldsymbol{\alpha}^h = \sigma\big(\mathbf{X}\mathbf{W}_h^Q (\mathbf{X}\mathbf{W}_h^K)^\top\big)$ is the attention coefficient, $\mathbf{W}_h^O \in \mathbb{R}^{d_H \times d}$, and $\mathbf{W}_h^V$, $\mathbf{W}_h^K$, $\mathbf{W}_h^Q \in \mathbb{R}^{d \times d_H}$.

[3] Normalizations after $a(\cdot)$ & $\mathrm{MLP}(\cdot)$ are omitted for simplicity.

Higher-order transformer layers [Kim et al., 2021]. Let the HoT layer be $f_{m\to n}: \mathbb{R}^{J^m\times d}\to\mathbb{R}^{J^n\times d}$ with two sub-layers: (i) a higher-order self-attention $a_{m\to n}: \mathbb{R}^{J^m\times d}\to\mathbb{R}^{J^n\times d}$ and (ii) a feed-forward $\mathrm{MLP}_{n\to n}: \mathbb{R}^{J^n\times d}\to\mathbb{R}^{J^n\times d}$. Moreover, let indexing vectors $\mathbf{i}\in\mathcal{I}_J^m\equiv\mathcal{I}_J\times\mathcal{I}_J\times\ldots\times\mathcal{I}_J$ (m modes) and $\mathbf{j}\in\mathcal{I}_J^n\equiv\mathcal{I}_J\times\mathcal{I}_J\times\ldots\times\mathcal{I}_J$ (n modes). For the input tensor $\mathbf{X}\in\mathbb{R}^{J^m\times d}$ with hyper-edges of order m, a HoT layer evaluates:

$$a_{m\to n}(\mathbf{X})_j=\sum_{h=1}^{H}\sum_{\mu}\sum_{\boldsymbol{i}}\boldsymbol{\alpha}_{\boldsymbol{i},\boldsymbol{j}}^{h,\mu}\mathbf{X}_{\boldsymbol{i}}\mathbf{W}_{h,\mu}^{V}\mathbf{W}_{h,\mu}^{O} \tag{6.7}$$

$$\mathrm{MLP}_{n\to n}(a_{m\to n}(\mathbf{X}))=\mathrm{L}_{n\to n}^{2}(\mathrm{ReLU}(\mathrm{L}_{n\to n}^{1}(a_{m\to n}(\mathbf{X})))), \tag{6.8}$$

$$f_{m\to n}(\mathbf{X})=a_{m\to n}(\mathbf{X})+\mathrm{MLP}_{n\to n}(a_{m\to n}(\mathbf{X})), \tag{6.9}$$

where $\boldsymbol{\alpha}^{h,\mu}\in\mathbb{R}^{J^{m+n}}$ is the so-called attention coefficient tensor with multiple heads, and $\boldsymbol{\alpha}_{\mathbf{i},\mathbf{j}}^{h,\mu}\in\mathbb{R}^{J}$ is a vector, $\mathbf{W}_{h,\mu}^{V}\in\mathbb{R}^{d\times d_H}$ and $\mathbf{W}_{h,\mu}^{O}\in\mathbb{R}^{d_H\times d}$ are learnable parameters. Moreover, μ indexes over the so-called equivalence classes of order-$(m+n)$ in the same partition of nodes, $\mathrm{L}_{n\to n}^{1}: \mathbb{R}^{J^n\times d}\to\mathbb{R}^{J^n\times d_F}$ and $\mathrm{L}_{n\to n}^{2}: \mathbb{R}^{J^n\times d_F}\to\mathbb{R}^{J^n\times d}$ are equivariant linear layers and d_F is the hidden dimension.

To compute each attention tensor $\boldsymbol{\alpha}^{h,\mu}\in\mathbb{R}^{J^{m+n}}$ from the input tensor $\mathbf{X}\in\mathbb{R}^{J^m\times d}$ of hyper-edges of order m, from the higher-order query and key, we obtain:

$$\boldsymbol{\alpha}_{\boldsymbol{i},\boldsymbol{j}}^{h,\mu}=\begin{cases}\frac{\sigma(\mathbf{Q}_{\boldsymbol{j}}^{h,\mu},\mathbf{K}_{\boldsymbol{i}}^{h,\mu})}{Z_{\boldsymbol{j}}} & (\boldsymbol{i},\boldsymbol{j})\in\mu\\ 0 & \text{otherwise,}\end{cases} \tag{6.10}$$

where $\mathbf{Q}^{\mu}=\mathrm{L}_{m\to n}^{\mu}(\mathbf{X})$, $\mathbf{K}^{\mu}=\mathrm{L}_{m\to m}^{\mu}(\mathbf{X})$, and normalization constant $Z_{\boldsymbol{j}}=\sum_{\boldsymbol{i}:(\boldsymbol{i},\boldsymbol{j})\in\mu}\sigma(\mathbf{Q}_{\boldsymbol{j}}^{\mu},\mathbf{K}_{\boldsymbol{i}}^{\mu})$. Finally, kernel attention in Eq. (6.10) can be approximated with RKHS feature maps $\psi\in\mathbb{R}_{+}^{d_K}$ for efficacy as $d_K\ll d_H$. Specifically, we have $\sigma(\mathbf{Q}_{\boldsymbol{j}}^{h,\mu},\mathbf{K}_{\boldsymbol{i}}^{h,\mu})\approx\boldsymbol{\psi}(\mathbf{Q}_{\boldsymbol{j}}^{h,\mu})^{\top}\boldsymbol{\psi}(\mathbf{K}_{\boldsymbol{i}}^{h,\mu})$ as in [Katharopoulos et al., 2020; Choromanski et al., 2021]. We choose the performer kernel [Choromanski et al., 2021] due to its good performance.

As query and key tensors are computed from the input tensor $\mathbf{X}$ using the equivariant linear layers, the transformer encoder layer $f_{m\to n}$ satisfies the permutation equivariance.

6.4 Approach

Skeletal Graph [Yan et al., 2018] and Skeletal Hypergraph [Liu et al., 2020a; Hao et al., 2021] are popular for modeling edges and hyper-edges. In this work, we use the Higher-order Transformer (HoT) [Kim et al., 2021] as a backbone encoder.

6.4.1 Model Overview

Fig. 6.1 shows that our framework contains a simple 3-layer MLP unit (FC, ReLU, FC, ReLU, Dropout, FC), three HoT blocks with each HoT for each type of input (*i.e.*, body joint feature set, graph and hypergraph of body joints), followed by Multi-order

Multi-mode Transformer (3Mformer) with two modules (i) Multi-order Pooling (MP) and (ii) Temporal block Pooling (TP). The goal of 3Mformer is to form coupled-mode tokens (explained later) such as 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' and 'channel-only', and perform weighted hyper-edge aggregation and temporal block aggregation. Their outputs are further concatenated and passed to an FC layer for classification.

MLP unit. The MLP unit takes T neighboring frames, each with J 2D/3D skeleton body joints, forming one temporal block. In total, depending on stride S, we obtain some τ temporal blocks (a block captures the short-term temporal evolution), In contrast, the long-term temporal evolution is modeled with HoT and 3Mformer. Each temporal block is encoded by the MLP into a $d \times J$ dimensional feature map.

HoT branches. We stack r branches of HoT, each taking embeddings $\mathbf{X}_t \in \mathbb{R}^{d \times J}$ where $t \in \mathcal{I}_\tau$ denotes a temporal block. HoT branches output hyper-edge feature representations of size $m \in \mathcal{I}_r$ as $\boldsymbol{\Phi}'_m \in \mathbb{R}^{J^m \times d'}$ for order $m \in \mathcal{I}_r$.

For the first-, second- and higher-order stream outputs $\boldsymbol{\Phi}'_1, ..., \boldsymbol{\Phi}'_r$, we (i) swap feature channel and hyper-edge modes, (ii) extract the upper triangular of tensors, and we concatenate along the block-temporal mode, so we have $\boldsymbol{\Phi}_m \in \mathbb{R}^{d' \times N_{E_m} \times \tau}$, where $N_{E_m} = \binom{J}{m}$. Subsequently, we concatenate $\boldsymbol{\Phi}_1, ..., \boldsymbol{\Phi}_r$ along the hyper-edge mode and obtain a multi-order feature tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{d' \times N \times \tau}$ where the total number of hyper-edges across all orders is $N = \sum_{m=1}^{r} \binom{J}{m}$.

3Mformer. Our Multi-order Multi-mode Transformer (3Mformer) with Coupled-mode Self-Attention (CmSA) is used for the fusion of information flow inside the multi-order feature tensor $\boldsymbol{\mathcal{M}}$, and finally, the output from 3Mformer is passed to a classifier for classification.

6.4.2 Coupled-mode Self-Attention

Coupled-mode tokens. We are inspired by the attentive regions of the one-class token in the standard Vision Transformer (ViT) [Vaswani et al., 2017] that can be leveraged to form a class-agnostic localization map. We investigate if the transformer model can also effectively capture the coupled-mode attention for more discriminative classification tasks, *e.g.*, tensorial skeleton-based action recognition by learning the coupled-mode tokens within the transformer. To this end, we propose a Multi-order Multi-mode Transformer (3Mformer), which uses coupled-mode tokens to jointly learn various higher-order motion dynamics among channel-, block-temporal-, body joint- and order-mode. Our 3Mformer can successfully produce coupled-mode relationships from CmSA mechanism corresponding to different tokens. Below we introduce our CmSA.

Given the order-r tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_r}$, to form the joint mode token, we perform the mode-m matricization of $\boldsymbol{\mathcal{M}}$ to obtain $\mathbf{M} \equiv \boldsymbol{\mathcal{M}}_{(m)}^\top \in \mathbb{R}^{(I_1 ... I_{m-1} I_{m+1} ... I_r) \times I_m}$, and the coupled-token for $\mathbf{M}$ is formed. For example, for a given 3rd-order tensor that has feature channel-, hyper-edge- and temporal block-mode, we can form 'channel-temporal block', 'channel-hyper-edge (any order)' and 'channel-only' pairs; and if

the given tensor is used as input and outputs a new tensor which produces new mode, *e.g.*, body joint-mode, we can form the 'order-channel-body joint' token. In the following sections, for simplicity, we use reshape for the matricization of tensor to form different types of coupled-mode tokens.

Our CmSA is given as:

$$a(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{SoftMax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_K}}\right)\mathbf{V}, \tag{6.11}$$

where $\sqrt{d_K}$ is the scaling factor, $\mathbf{Q}=\mathbf{W}^q\mathbf{M}$, $\mathbf{K}=\mathbf{W}^k\mathbf{M}$ and $\mathbf{V}=\mathbf{W}^v\mathbf{M}$ are the query, key and value, respectively, and $\mathbf{M}\equiv\boldsymbol{\mathcal{M}}_{(m)}^\top$. Moreover, $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}\in\mathbb{R}^{(I_1\ldots I_{m-1}I_{m+1}\ldots I_r)\times I_m}$ and $\mathbf{W}^q$, $\mathbf{W}^k$, $\mathbf{W}^v\in\mathbb{R}^{(I_1\ldots I_{m-1}I_{m+1}\ldots I_r)\times(I_1\ldots I_{m-1}I_{m+1}\ldots I_r)}$ are learnable weights. We notice that various coupled-mode tokens have different 'focus' of attention mechanisms, and we apply them in our 3Mformer for the fusion of multi-order feature representations.

6.4.3 Multi-order Multi-mode Transformer

Below we introduce Multi-order Multi-mode Transformer (3Mformer) with Multi-order Pooling (MP) block and Temporal block Pooling (TP) block, which are cascaded into two branches (i) MP→TP and (ii) TP→MP, to achieve different types of coupled-mode tokens.

6.4.3.1 Multi-order Pooling (MP) Module

CmSA in MP. We reshape the multi-order feature representation $\boldsymbol{\mathcal{M}}\in\mathbb{R}^{d'\times N\times\tau}$ into $\mathbf{M}\in\mathbb{R}^{d'\tau\times N}$ (or reshape the output from TP explained later into $\mathbf{M}'\in\mathbb{R}^{d'\times N}$) to let the model attend to different types of feature representations. Let us simply denote $d''=d'\tau$ (or $d''=d'$) depending on the source of input. We form an coupled-mode self-attention (if $d''=d'\tau$, we have, *i.e.*, 'channel-temporal block' token; if $d''=d'$, we have 'channel-only' token):

$$a_{\text{MP}}(\mathbf{Q}_{\text{MP}},\mathbf{K}_{\text{MP}},\mathbf{V}_{\text{MP}}) = \text{SoftMax}\left(\frac{\mathbf{Q}_{\text{MP}}\mathbf{K}_{\text{MP}}^\top}{\sqrt{d_{K_{\text{MP}}}}}\right)\mathbf{V}_{\text{MP}}, \tag{6.12}$$

where $\sqrt{d_{K_{\text{MP}}}}$ is the scaling factor, $\mathbf{Q}_{\text{MP}}=\mathbf{W}_{\text{MP}}^q\mathbf{M}$, $\mathbf{K}_{\text{MP}}=\mathbf{W}_{\text{MP}}^k\mathbf{M}$ and $\mathbf{V}_{\text{MP}}=\mathbf{W}_{\text{MP}}^v\mathbf{M}$ (we can use here $\mathbf{M}$ or $\mathbf{M}'$) are the query, key and value. Moreover, $\mathbf{Q}_{\text{MP}}$, $\mathbf{K}_{\text{MP}}$, $\mathbf{V}_{\text{MP}}\in\mathbb{R}^{d''\times N}$ and $\mathbf{W}_{\text{MP}}^q$, $\mathbf{W}_{\text{MP}}^k$, $\mathbf{W}_{\text{MP}}^v\in\mathbb{R}^{d''\times d''}$ are learnable weights. Eq. (6.12) is a self-attention layer which reweighs $\mathbf{V}_{\text{MP}}$ based on the correlation between $\mathbf{Q}_{\text{MP}}$ and $\mathbf{K}_{\text{MP}}$ token embeddings of so-called coupled-mode tokens.

Weighted pooling. Attention layer in Eq. (6.12) produces feature representation $\mathbf{O}_{\text{MP}}\in\mathbb{R}^{d''\times N}$ to enhance the relationship between for example feature channels and body joints. Subsequently, we handle the impact of hyper-edges of multiple orders by

weighted pooling along hyper-edges of order $m \in \mathcal{I}_r$:

$$\mathbf{O}_{\text{MP}}^{*(m)} = \mathbf{O}_{\text{MP}}^{(m)} \mathbf{H}^{(m)} \in \mathbb{R}^{d'' \times J}, \tag{6.13}$$

where $\mathbf{O}_{\text{MP}}^{(m)} \in \mathbb{R}^{d'' \times N_{E_m}}$ is simply extracted from $\mathbf{O}_{\text{MP}}$ for hyper-edges of order m, and matrices $\mathbf{H}^{(m)} \in \mathbb{R}^{N_{E_m} \times J}$ are learnable weights to perform weighted pooling along hyper-edges of order m. Finally, we obtain $\mathbf{O}_{\text{MP}}^{*} \in \mathbb{R}^{rd'' \times J}$ by simply concatenating $\mathbf{O}_{\text{MP}}^{*(1)}, ..., \mathbf{O}_{\text{MP}}^{*(r)}$. If we used the input to MP from TP, then we denote the output of MP as $\mathbf{O}'^{*}_{\text{MP}}$.

6.4.3.2 Temporal block Pooling (TP) Module

CmSA in TP. Firstly, we reshape the multi-order feature representation $\mathcal{M} \in \mathbb{R}^{d' \times N \times \tau}$ into $\mathbf{M} \in \mathbb{R}^{d'N \times \tau}$ (or reshape the output from MP into $\mathbf{M}'' \in \mathbb{R}^{rd'J \times \tau}$). For simplicity, we denote $d''' = d'N$ in the first case and $d''' = rd'J$ in the second case. As the first mode of reshaped input serves to form tokens, they are again coupled-mode tokens, *e.g.*, 'channel-hyper-edge' and 'order-channel-body joint' tokens, respectively. Moreover, TP also performs pooling along block-temporal mode (along τ). We form an coupled-mode self-attention:

$$a_{\text{TP}}(\mathbf{Q}_{\text{TP}}, \mathbf{K}_{\text{TP}}, \mathbf{V}_{\text{TP}}) = \text{SoftMax}\left(\frac{\mathbf{Q}_{\text{TP}} \mathbf{K}_{\text{TP}}^{\top}}{\sqrt{d_{K_{\text{TP}}}}}\right) \mathbf{V}_{\text{TP}}, \tag{6.14}$$

where $\sqrt{d_{K_{\text{TP}}}}$ is the scaling factor, $\mathbf{Q}_{\text{TP}} = \mathbf{W}_{\text{TP}}^{q} \mathbf{M}$, $\mathbf{K}_{\text{TP}} = \mathbf{W}_{\text{TP}}^{k} \mathbf{M}$ and $\mathbf{V}_{\text{TP}} = \mathbf{W}_{\text{TP}}^{v} \mathbf{M}$ (we can use here $\mathbf{M}$ or $\mathbf{M}''$) are the query, key and value. Moreover, $\mathbf{Q}_{\text{TP}}$, $\mathbf{K}_{\text{TP}}$, $\mathbf{V}_{\text{TP}} \in \mathbb{R}^{d''' \times \tau}$ and $\mathbf{W}_{\text{TP}}^{q}$, $\mathbf{W}_{\text{TP}}^{k}$, $\mathbf{W}_{\text{TP}}^{v} \in \mathbb{R}^{d''' \times d'''}$ are learnable weights. Eq. (6.14) reweighs $\mathbf{V}_{\text{TP}}$ based on the correlation between $\mathbf{Q}_{\text{TP}}$ and $\mathbf{K}_{\text{TP}}$ token embeddings of coupled-mode tokens ('channel-hyper-edge' or 'order-channel-body joint'). The output of attention is the temporal representation $\mathbf{O}_{\text{TP}} \in \mathbb{R}^{d''' \times \tau}$. If we used $\mathbf{M}''$ as input, we denote the output as $\mathbf{O}''_{\text{TP}}$.

Pooling step. Given the temporal representation $\mathbf{O}_{\text{TP}} \in \mathbb{R}^{d''' \times \tau}$ (or $\mathbf{O}''_{\text{TP}}$), we apply pooling along the block-temporal mode to obtain compact feature representations independent of length (block count τ) of skeleton sequence. There exist many pooling operations[4] including first-order, *e.g.*, average, maximum, sum pooling, second-order [Zilin et al., 2019; Wang et al., 2018a] such as attentional pooling [Girdhar and Ramanan, 2017], higher-order (tri-linear) [Cherian et al., 2017b; Koniusz et al., 2021] and rank pooling [Fernando et al., 2017]. The output after pooling is $\mathbf{O}_{\text{TP}}^{*} \in \mathbb{R}^{d'''}$ (or $\mathbf{O}''^{*}_{\text{TP}}$).

[4]We do not propose pooling operators but we select popular ones with the purpose of comparing their impact on TP.

6.4.3.3 Model Variants

We devise four model variants by different stacking of MP with TP, with the goal of exploiting attention with different kinds of coupled-mode tokens:

i. Single-branch: MP followed by TP, denoted MP→TP, (Fig. 6.1 top right branch).

ii. Single-branch: TP followed by MP, denoted TP→MP, (Fig. 6.1 bottom right branch).

iii. Two-branch (our 3Mformer, Fig. 6.1) which concatenates outputs of MP→TP and TP→MP.

iv. We also investigate only MP or TP module followed by average pooling or an FC layer.

The outputs from MP→TP and TP→MP have exactly the same feature dimension ($\mathbb{R}^{rd'J}$, after reshaping into vector). For two-branch (our 3Mformer), we simply concatenate these outputs ($\mathbb{R}^{2rd'J}$, after concatenation). These vectors are forwarded to the FC layer to learn a classifier.

6.4.4 Visualization of 3Mformer.

Fig. 6.2 shows the visualization of our 3Mformer. The green and orange blocks denote the Multi-order Pooling (MP) and the Temporal block Pooling (TP) respectively, which are two basic building blocks that can be stacked to form our 3Mformer. More precisely, our 3Mformer consists of two branches: (i) MP followed by TP (denoted MP → TP, Fig. 6.2a) and (ii) TP followed by MP (denoted TP → MP, Fig. 6.2b).

6.5 Experiments

6.5.1 Datasets and Protocols

(i) NTU RGB+D (NTU-60) [Shahroudy et al., 2016a] contains 56,880 video sequences.This dataset has variable sequence lengths and high intra-class variations. Each skeleton sequence has 25 joints and there are no more than two human subjects in each video. Two evaluation protocols are: (i) cross-subject (X-Sub) and (ii) cross-view (X-View).
(ii) NTU RGB+D 120 (NTU-120) [Liu et al., 2019a], an extension of NTU-60, contains 120 action classes (daily/health-related), and 114,480 RGB+D video samples captured with 106 distinct human subjects from 155 different camera viewpoints. There are also two evaluation protocols: (i) cross-subject (X-Sub) and (ii) cross-setup (X-Set).
(iii) Kinetics-Skeleton, based on Kinetics [Kay et al., 2017], is large-scale dataset with 300,000 video clips and up to 400 human actions collected from YouTube. This dataset involves human daily activities, sports scenes and complex human-computer interaction scenes. Since Kinetics only provides raw videos without the skeletons, ST-GCN [Yan et al., 2018] uses the publicly available OpenPose toolbox [Cao et al.,

2017] to estimate and extract the location of 18 human body joints on every frame in the clips. We use their released skeleton data to evaluate our model. Following the standard evaluation protocol, we report the Top-1 and Top-5 accuracies on the validation set.

(iv) Northwestern-UCLA [Wang et al., 2014] was captured by 3 Kinect cameras simultaneously from multiple viewpoints. It contains 1494 video clips covering 10 actions. Each action is performed by 10 different subjects. We follow the same evaluation protocol as [Wang et al., 2014]: training split is formed from the first two cameras, and testing split from the last camera.

6.5.2 Skeleton Data Preprocessing

Before passing the skeleton sequences into MLP, we first normalize each body joint w.r.t. to the torso joint $\mathbf{v}_{f,c}$:

$$\mathbf{v}'_{f,i} = \mathbf{v}_{f,i} - \mathbf{v}_{f,c}, \tag{6.15}$$

where f and i are the index of video frame and human body joint respectively. After that, we further normalize each joint coordinate into [-1, 1] range:

$$\hat{\mathbf{v}}_{f,i}[j] = \frac{\mathbf{v}'_{f,i}[j]}{\max([\text{abs}(\mathbf{v}'_{f,i}[j])]_{f \in \mathcal{I}_\tau, i \in \mathcal{I}_J})}, \tag{6.16}$$

where j is for selection of the x, y and z axes, τ is the number of frames and J is the number of 3D body joints per frame.

For the skeleton sequences that have more than one performing subject, (i) we normalize each skeleton separately, and each skeleton is passed to MLP for learning the temporal dynamics, and (ii) for the output features per skeleton from MLP, we pass them separately to the block-temporal HoT, *e.g.*, two skeletons from a given video sequence will have two outputs obtained from the the block-temporal HoT, and we aggregate the outputs through average pooling before passing our 3Mformer.

6.5.3 Experimental Setup

We use PyTorch and 1×Titan RTX 3090 for experiments. We use the Stochastic Gradient Descent (SGD) with momentum 0.9, cross-entropy as the loss, weight decay of 0.0001 and batch size of 32. The learning rate is set to 0.1 initially. On NTU-60 and NTU-120, the learning rate is divided by 10 at the 40th and 50th epoch, and the training process ends at the 60th epoch. On Kinetics-Skeleton, the learning rate is divided by 10 at the 50th and 60th epoch, and the training finishes at the 80th epoch. We took 20% of training set for validation to tune hyperparameters. All models have fixed hyperparameters with 2 and 4 layers for NTU-60/NTU-120 and Kinetics-Skeleton, respectively. The hidden dimensions is set to 16 for all 3 datasets. We use 4 attention heads for NTU-60 and NTU-120, and 8 attention heads for Kinetics-Skeleton. To form each video temporal block, we simply choose temporal block size to be 10 and stride to be 5 to allow a 50% overlap between consecutive temporal blocks.

Table 6.1: Search for the single best order n of hypergraph (except for $n=3\,\&\,4$ where we check if $n=3\,\&\,4$ are complementary).

Order-n	NTU-60		NTU-120		Kinetics-Skel.
	X-Sub	X-View	X-Sub	X-Set	Top-1 acc.
$n=1$	78.5	86.3	75.3	77.9	32.0
$n=2$	83.0	89.2	86.2	88.3	37.1
$n=3$	91.3	97.0	87.5	89.7	39.5
$n=4$	91.5	97.1	**87.8**	90.0	40.1
$n=5$	91.4	**97.3**	**87.8**	90.0	40.3
$n=3$ & 4	**91.6**	97.2	87.6	**90.3**	**40.5**

Table 6.2: Evaluations of our model variants with/without MP and/or TP. Baseline in the table denotes the backbone (MLP unit + HoTs) without the use of either MP or TP module.

Variants	NTU-60		NTU-120		Kinetics-Skel.
	X-Sub	X-View	X-Sub	X-Set	Top-1 acc.
Baseline	89.8	91.4	86.5	87.0	38.6
+ TP only	91.2	93.8	87.5	88.6	39.8
+ MP only	92.0	94.3	88.7	89.7	40.3
+ MP→TP	93.0	96.1	90.8	91.7	45.7
+ TP→MP	92.6	95.8	90.2	91.1	44.0
+ 2-branch(3Mformer	**94.8**	**98.7**	**92.0**	**93.8**	**48.3**

For Northwestern-UCLA, the batch size is 16. We adopted the data pre-processing in [Cheng et al., 2020b].

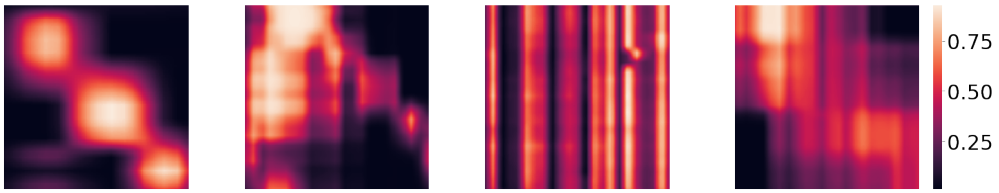
6.5.4 Ablation Study

Search for the single best order n. Table 6.1 shows our analysis regarding the best order n. In general, increasing the order n improves the performance (within $\sim 0.5\%$ on average), but causing higher computational cost, *e.g.*, the number of hyper-edges for the skeletal hypergraph of order $n=4$ is 3060 on Kinetics-Skeleton. We also notice that combining orders 3 and 4 yields very limited improvements. The main reasons are: (i) reasonable order n, *e.g.*, $n=3$ or 4 improves accuracy as higher-order motion patterns are captured which are useful for classification-related tasks (ii) further increasing order n, *e.g.*, $n=5$ introduces patterns in feature representations that rarely repeat even for the same action class. Considering the cost and performance, we choose the maximum order $r=3$ ($n=1,2,3$) in the following experiments unless specified otherwise.

Discussion on coupled-mode attention. Fig. 6.3 shows the visualization of some attention matrices in our 3Mformer, which show diagonal and/or vertical patterns that are consistent with the patterns of the attention matrices found in standard Transformer trained on sequences, *e.g.*, for natural language processing tasks [Vaswani et al., 2017; Kovaleva et al., 2019]. We also notice that the coupled-mode attention, *e.g.*, 'channel-temporal block' captures much richer information compared to single mode

Table 6.3: Experimental results on NTU-60, NTU-120 and Kinetics-Skeleton.

	Method	Venue	NTU-60		NTU-120		Kinetics-Skeleton	
			X-Sub	X-View	X-Sub	X-Set	Top-1	Top-5
Graph-based	TCN [Kim and Reiter, 2017]	CVPRW'17	-	-	-	-	20.3	40.0
	ST-GCN [Yan et al., 2018]	AAAI'18	81.5	88.3	70.7	73.2	30.7	52.8
	AS-GCN [Li et al., 2019]	CVPR'19	86.8	94.2	78.3	79.8	34.8	56.5
	2S-AGCN [Shi et al., 2019b]	CVPR'19	88.5	95.1	82.5	84.2	36.1	58.7
	NAS-GCN [Peng et al., 2020]	AAAI'20	89.4	95.7	-	-	37.1	60.1
	Sym-GNN [Li et al., 2022]	TPAMI'22	90.1	96.4	-	-	37.2	58.1
	Shift-GCN [Cheng et al., 2020b]	CVPR'20	90.7	96.5	85.9	87.6	-	-
	MS-G3D [Liu et al., 2020b]	CVPR'20	91.5	96.2	86.9	88.4	38.0	60.9
	CTR-GCN [Chen et al., 2021b]	ICCV'21	92.4	96.8	88.9	90.6	-	-
	InfoGCN [Chi et al., 2022]	CVPR'22	93.0	97.1	89.8	91.2	-	-
	PoseConv3D [Duan et al., 2022]	CVPR'22	94.1	97.1	86.9	90.3	**47.7**	-
Hypergraph-based	Hyper-GNN [Hao et al., 2021]	TIP'21	89.5	95.7	-	-	37.1	60.0
	DHGCN [Wei et al., 2021b]	CoRR'21	90.7	96.0	86.0	87.9	37.7	60.6
	Selective-HCN [Zhu et al., 2022]	ICMR'22	90.8	96.6	-	-	38.0	61.1
	SD-HGCN [He et al., 2021]	ICONIP'21	90.9	96.7	87.0	88.2	37.4	60.5
Transformer-based	ST-TR [Plizzari et al., 2021]	CVIU'21	90.3	96.3	85.1	87.1	38.0	60.5
	MTT [Kong et al., 2022]	LSP'21	90.8	96.7	86.1	87.6	37.9	61.3
	4s-GSTN [Jiang et al., 2022]	Symmetry'22	91.3	96.6	86.4	88.7	-	-
	STST [Zhang et al., 2021e]	ACM MM'21	91.9	96.8	-	-	38.3	61.2
	3Mformer (with avg-pool, *ours*)		92.0	97.3	88.0	90.1	43.1	65.2
	3Mformer (with max-pool, *ours*)		92.1	97.8	-	-	-	-
	3Mformer (with attn-pool, *ours*)		**94.2**	**98.5**	89.7	92.4	45.7	67.6
	3Mformer (with tri-pool, *ours*)		94.0	**98.5**	**91.2**	**92.7**	**47.7**	**71.9**
	3Mformer (with rank-pool, *ours*)		**94.8**	**98.7**	**92.0**	**93.8**	**48.3**	**72.3**



(a) (b) (c) (d)

Figure 6.3: Visualization of attention matrices. (a) single-mode attention matrix of 'channel-only' token, (b)–(d) coupled-mode attention matrices of 'channel-hyper-edge', 'order-channel-body joint' and 'channel-temporal block' tokens, respectively.

attention, *e.g.*, 'channel-only'. Our coupled-mode attention can be applied to different orders of tensor representations through simple matricization.

Discussion on model variants. To show the effectiveness of the proposed MP and TP module, firstly, we compare TP only and MP only with the baseline (No MP or TP module). We use the TP module followed by an FC layer instead of MP as in TP→MP, where the FC layer takes the output from TP ($\mathbb{R}^{d'N}$) and produces a vector in $\mathbb{R}^{3d'J}$, passed to the classifier. Similarly, for MP only, we use the MP module followed by an average pooling layer instead of TP as in MP→TP, where the average layer takes output from MP ($\mathbb{R}^{3d'J\times\tau}$) and generates a vector in $\mathbb{R}^{3d'J}$ (pool along τ blocks), passed to the classifier. Table 6.2 shows the results. With just the TP module, we outperform the baseline by 1.3% on average. With only the MP module, we outperform the baseline by 2.34% on average. These comparisons show that (i) CmSA in MP and TP are efficient for better performance (ii) MP performs better than TP which shows that

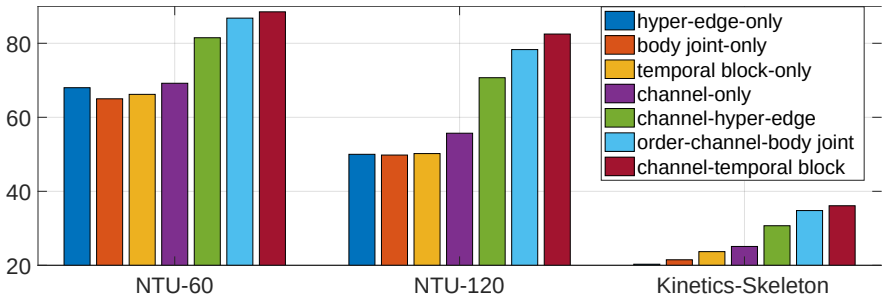


Figure 6.4: Evaluations of different single-mode (*baseline*) and joint-mode tokens. We use a 3rd-order HoT with a standard Transformer, but we replace the scaled dot-product attention with joint-mode tokens and joint-mode attention.

'channel-temporal block' token contains richer information than 'channel-hyper-edge' token. We also notice that MP→TP slightly outperforms TP→MP by $\sim$ 1%, and the main reason is that MP→TP has coupled-mode tokens 'channel-temporal block' and 'order-channel-joint' which attend 4 modes, whereas TP→MP has 'channel-hyper-edge' and 'channel-only' tokens which attend only 2 modes. Fig. 6.4 shows a comparison of different coupled-mode tokens on 3 benchmark datasets. This also suggests that one should firstly perform attention with coupled-mode 'channel-block' tokens, followed by weighted pooling along the hyper-edge mode, followed by attention with coupled-mode 'order-channel-body joint' and finalised by block-temporal pooling. Finally, with 2-branch (3Mformer), we further boost the performance by 2–4%, which shows that MP→TP and TP→MP are complementary branches. Below we use 2-branch (3Mformer) in the experiments (as in Fig. 6.1).

Comparison of pooling in TP. As shown in Table 6.3, average pooling (avg-pool) achieves similar performance (within $\sim$ 0.5% difference) as maximum pooling (max-pool), second-order pooling (attn-pool) outperforms average and maximum pooling by $\sim$ 1–2% and third-order pooling (tri-pool) outperforms second-order pooling by $\sim$ 1%. Interestingly, rank pooling (rank-pool) achieves the best performance. We think it is reasonable as rank pooling strives to enforce the temporal order in the feature space to be preserved, *e.g.*, it forces network to always preserve temporal progression of actions over time. With multiple attention modules, orderless statistics such as second- or third-order pooling may be too general.

6.5.5 Comparisons with the State of the Arts

We compare our model with recent state-of-the-art methods. On the NTU-60 (Tab. 6.3), we obtain the top-1 accuracies of the two evaluation protocols during test stage. The methods in comparisons include popular graph-based [Yan et al., 2018; Li et al., 2019; Shi et al., 2019b; Peng et al., 2020; Li et al., 2022] and hypergraph-based models [Hao et al., 2021; Wei et al., 2021b; Zhu et al., 2022; He et al., 2021]. Our 3rd-order model outperforms all graph-based methods, and also outperforms existing hypergraph-based models such as Selective-HCN and SD-HGCN by 0.45% and 0.35% on average

Table 6.4: Experimental results on Northwestern-UCLA.

	Shift-GCN (CVPR'20) [Cheng et al., 2020b]	CTR-GCN (ICCV'21) [Chen et al., 2021b]	InfoGCN (CVPR'22) [Chi et al., 2022]	2nd-order only (*ours*)	3rd-order only (*ours*)	3Mformer (*ours*)
acc.(%)	94.6	96.5	97.0	96.5	**97.2**	**97.8**

on X-Sub and X-View respectively. With 3Mformer for the fusion of multi-order features, our model further boosts the performance by $\sim$ 3% and 1.5% on the two protocols.

It can be seen from Tab. 6.3 on NTU-60 that although some learned graph-based methods such as AS-GCN and 2S-AGCN can also capture the dependencies between human body joints, they only consider the pairwise relationship between body joints, which is the second-order interaction, and ignore the higher-order interaction between multiple body joints in form of hyper-edges, which may lose sensitivity to important groups of body joints. Our proposed 3Mformer achieves better performance by constructing a hypergraph from 2D/3D body joints as nodes for action recognition, thus capturing higher-order interactions of body joints to further improve the performance. Note that even with the average pooling, our model still achieves competitive results compared to its counterparts.

For the NTU-120 dataset (Tab. 6.3), we obtain the top-1 performance on X-Sub and X-Set protocols. Our 2nd-order HoT alone outperforms graph-based models by 2–2.4% on average. For example, we outperform recent Shift-GCN by 0.3% and 0.7% on X-Sub and X-Set respectively. Moreover, our 3rd-order HoT alone outperforms SD-HGCN by 0.5% and 1.5% respectively on X-Sub and X-Set. With the 3Mformer for the fusion of multi-order feature maps, we obtain the new state-of-the-art results. Notice that our 3Mformer yields 92.0% / 93.8% on NTU-120 while [Peng et al., 2021] yields 80.5% / 81.7% as we explore the fusion of multiple orders of hyperedges and several coupled-token types capturing easy-to-complex dynamics of varying joint groups.

As videos from the Kinetics dataset are processed by the OpenPose, the skeletons in the Kinetics-Skeleton dataset have defects which adversely affect the performance of the model. We show both top-1 and top-5 performance in Table 6.3 to better reflect the performance of our 3Mformer. ST-GCN is the first method based on GCN, our 2nd-order HoT alone achieves very competitive results compared to the very recent NAS-GCN and Sym-GNN. The 3rd-order HoT alone outperforms Hyper-GNN, SD-HGCN and Selective-HCN by 3.4%, 3.1% and 2.9% respectively for top-1 accuracies. Moreover, fusing multi-order feature maps from multiple orders of hyper-edges via 3Mformer gives us the best performance on Kinetics-Skeleton with 48.3% for top-1, the new state-of-the-art result.

Table 6.4 shows results on the Northwestern-UCLA dataset. Our 3Mformer is also effective on this dataset–it outperforms the current state-of-the-art InfoGCN by 0.8%.

Table 6.5: Ablations of different pooling methods in MP.

Pooling	NTU-60		NTU-120		Kinetics-Skel.
	X-Sub	X-View	X-Sub	X-Set	Top-1 acc.
avg-pool	91.3	96.8	86.5	89.0	41.9
max-pool	92.7	98.0	88.5	91.0	43.8
wei-pool (*ours*)	94.8	98.7	92.0	93.8	48.3

6.6 Additional Results and Discussions

6.6.1 Ablations of MP

We choose average pooling (avg-pool) and max-pooling (max-pool) for hyper-edge features in comparison to our learned weighted pooling (wei-pool), and the comparisons are given in Table 6.5. As shown in the table, our learned weighted pooling (wei-pool) consistently achieves the best performance on all 3 datasets.

6.6.2 Learning the short-term temporal patterns

A block of T neighbor frames are passed to the MLP unit to capture the short-term temporal patterns. The whole sequence consists of τ such blocks, each passed separately through the MLP unit (and each joint $1, ..., J$). Thus, the MLP only mixes the information from $1, ..., T$ frames of a given block/body joint j and captures short-term relations (within-block) of a given 3D body joint (in contrast to between-block relations). The MLP unit input size is $3T$; 3 due to 3D coordinate). The MLP: $\mathbb{R}^{3T} \rightarrow \mathbb{R}^{d}$ contains: FC ($3T \rightarrow 6T$), ReLU, FC ($6T \rightarrow 9T$), ReLU, Dropout, FC ($9T \rightarrow d$). J body joints and τ blocks are treated as the batch dimension. Feature output size d: 100, 150, 420 on NTU-60, NTU-120, Kinetics-Skeleton.

6.6.3 Why 3Mformer works and when does it fail?

Our method works well as it (i) uses skeletal hypergraphs of various orders to learn the interaction of varying size groups of skeletal joints (as opposed to typical skeleton graph physical connectivity), (ii) fuses groups multiple orders by 3Mformer by several coupled-token types via two basic building blocks (MP & TP) that learn various aspects of higher-order motion dynamics. Multiple-order hyperedges are more resistant to noise (*e.g.*, Kinetics-Skeleton is noisy due to the pose estimation errors), if one body joint is noisy (but the rest is stable). We inject Gaussian noise into 3D *ankle* joints, vary noise amplitude, and we show the experimental results in Table 6.6. As shown in the table, our 3Mformer copes with noise better than ST-GCN.

Our method may underperform if (i) the backbone encoder cannot efficiently produce higher-order features (ii) the number of skeletal joints are very large (the number of hyper-edge features would be very large) (iii) when dataset is too small to learn high-order interactions (extra learnable parameters). For example, see the experimental results on MSRAction3D in Table 6.7.

Table 6.6: Comparisons of robustness w.r.t. Gaussian noise.

	original	× 1	× 1.5	× 2
ST-GCN	81.5	74.9 (↓6.6)	69.2 (↓12.3)	50.1 (↓31.4)
3Mformer	94.8	91.9 (↓2.9)	89.5 (↓5.3)	86.8 (↓8.0)

Table 6.7: Experimental results on MSRAction3D.

order	2	3	4
acc.(%)	73.82	63.64	55.27

Table 6.8: A comparison of the number of model parameters and FLOPs on NTU-60.

	ST-GCN	2S-AGCN	NAS-GCN	2rd-order only (*ours*)	3rd-order only (*ours*)	3Mformer (*ours*)
Params (M)	3.14	3.45	6.57	1.15	2.07	4.37
FLOPs (G)	16.36	37.22	108.82	6.54	35.53	58.45
Acc. (%)	81.5	88.5	89.4	83.0	91.3	**94.8**

We notice that small datasets may be not enough to train high-order models (Table 6.7). On key classic large datasets, NTU-60, NTU-120 and Kinetics-Skeleton, we do not observe any issue as human motions exhibit similar multi-joint dynamics for typical action classes. Perhaps some fine-grained unusual action classes could pose problems.

6.6.4 Model Complexity

Table 6.8 shows the number of model parameters/FLOPs and NTU-60 accuracy. Our cost is moderate. 2S-AGCN (37.22 GFLOPs & 3.45M param.) yields 89.4% accuracy. Our '3rd-order' uses 35.5 GFLOPs & 2.07M param. which is 2 GFLOPs & 1.37M param. less, yet we outperform 2S-AGC by **1.9%**. NAS-GCN uses 40.4 GFLOPs/2.2M param. more compared to our 3Mformer: we beat NAS-GCN by **4.4%**.

6.6.5 Limitation and Future Work

Despite the high accuracy of our model, there are still some limitations. Firstly, as we use r branches of HoT, the number of parameters and computational cost are higher than existing methods. However, our method with single branch, *e.g.*, 3rd-order HoT only, still achieves very competitive results compared to existing graph-, transformer- and hypergraph-based models for the same parameter scale on 3 benchmarks. Secondly, in this work, we only use HoT block to encode the temporal block feature representations. The more efficient way is to redesign HoT block so that it is able to encode both short-term and long-term spatio-temporal features to simplify

the backbone encoder, *i.e.*, without the need of MLP unit. Note that the design of our 3Mformer is independent of the backbone encoder. Our 3Mformer is especially suitable for tensorial data, *e.g.*, higher-order feature representations. Our future work will focus on applying our Multi-order Multi-mode Transformer (3Mformer) to other computer vision tasks with tensorial data.

6.7 Conclusions

In this chapter, we model the skeleton data as hypergraph to capture higher-order information formed between groups of human body joints of orders $1, ..., r$. We use Higher-order Transformer (HoT) to learn higher-order information on hypergraphs of r-order formed over 2D/3D human body joints. We also introduce a novel Multi-order Multi-mode Transformer (3Mformer) for the fusion of multi-order feature representations. Our end-to-end trainable 3Mformer outperforms state-of-the-art graph- and hypergraph-based models by a large margin on several benchmarks.

Chapter 7

Uncertainty-DTW

Dynamic Time Warping (DTW) is used for matching pairs of sequences and celebrated in applications such as forecasting the evolution of time series, clustering time series or even matching sequence pairs in few-shot action recognition. The transportation plan of DTW contains a set of paths; each path matches frames between two sequences under a varying degree of time warping, to account for varying temporal intra-class dynamics of actions. However, as DTW is the smallest distance among all paths, it may be affected by the feature uncertainty which varies across time steps/frames. Thus, in this chapter, we propose to model the so-called aleatoric uncertainty of a differentiable (soft) version of DTW. To this end, we model the heteroscedastic aleatoric uncertainty of each path by the product of likelihoods from Normal distributions, each capturing variance of pair of frames. (The path distance is the sum of base distances between features of pairs of frames of the path.) The Maximum Likelihood Estimation (MLE) applied to a path yields two terms: (i) a sum of Euclidean distances weighted by the variance inverse, and (ii) a sum of log-variance regularization terms. Thus, our uncertainty-DTW is the smallest weighted path distance among all paths, and the regularization term (penalty for the high uncertainty) is the aggregate of log-variances along the path. The distance and the regularization term can be used in various objectives. We showcase forecasting the evolution of time series, estimating the Fréchet mean of time series, and supervised/unsupervised few-shot action recognition of the articulated human 3D body joints.

7.1 Introduction

Dynamic Time Warping (DTW) [Cuturi, 2011] is a method popular in forecasting the evolution of time series, estimating the Fréchet mean of time series, or classifying generally understood actions. The key property of DTW is its sequence matching transportation plan that allows any two sequences that are being matched to progress at different 'speeds' not only in the global sense but locally in the temporal sense. As DTW is non-differentiable, a differentiable 'soft' variant of DTW, soft-DTW [Cuturi and Blondel, 2017], uses a soft-minimum function which enables backpropagation.

The role of soft-DTW is to evaluate the (relaxed) DTW distance between a pair of sequences $\boldsymbol{\Psi} \equiv [\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_\tau] \in \mathbb{R}^{d' \times \tau}$, $\boldsymbol{\Psi}' \equiv [\boldsymbol{\psi'}_1, ..., \boldsymbol{\psi'}_{\tau'}] \in \mathbb{R}^{d' \times \tau'}$ of lengths τ and τ',

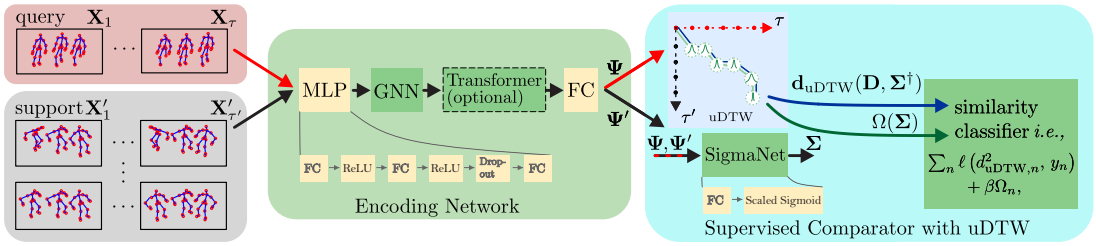


Figure 7.1: Supervised few-shot action recognition of the articulated human 3D body joints with the uncertainty-DTW (uDTW). Frames from a query and support sequences are split into short-term temporal blocks $\mathbf{X}_1, ..., \mathbf{X}_\tau$ and $\mathbf{X}'_1, ..., \mathbf{X}'_{\tau'}$ of length M given stride S. We pass all skeleton coordinates via Encoding Network to obtain feature tensors $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}'$, which are directed to the Supervised Comparator with uDTW. For each query-support pair $(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n)$, uDTW computes the base-distance matrix $\mathbf{D}_n$ reweighted by uncertainty $\boldsymbol{\Sigma}_n^\dagger$ to compare $\tau \times \tau'$ blocks, and SigmaNet generates underlying block-wise uncertainty parameters $\boldsymbol{\Sigma}_n$. uDTW finds the warping path with the smallest distance, and returns its Ω_n penalty (uncertainty aggregated along the path).

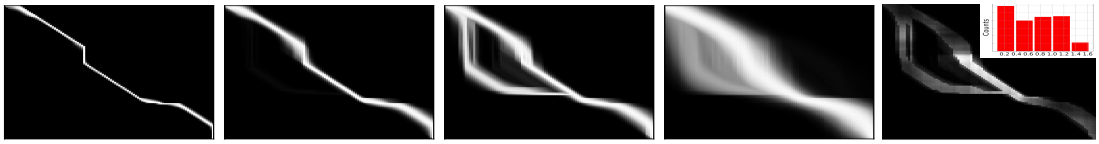
respectively. Under its transportation plan $\mathcal{A}_{\tau,\tau'}$, each path $\boldsymbol{\Pi} \in \mathcal{A}_{\tau,\tau'}$ is evaluated to ascertain the path distance, and the smallest distance is 'selected' by the soft minimum:

$$d^2_{\mathrm{DTW}}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') = \mathrm{SoftMin}_\gamma \left(\left[\langle \boldsymbol{\Pi}, \boldsymbol{D}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \rangle \right]_{\boldsymbol{\Pi} \in \mathcal{A}_{\tau,\tau'}} \right), \tag{7.1}$$

where $\mathrm{SoftMin}_\gamma(\boldsymbol{\alpha}) = -\gamma \log \sum_i \exp(-\alpha_i/\gamma)$ is the soft minimum, $\gamma \geq 0$ controls its relaxation (hard *vs*. soft path selection), and $\boldsymbol{D} \in \mathbb{R}_+^{\tau \times \tau'} \equiv [d^2_{\mathrm{base}}(\boldsymbol{\psi}_m, \boldsymbol{\psi}'_n)]_{(m,n) \in \mathcal{I}_\tau \times \mathcal{I}_{\tau'}}$ contains pair-wise distances between all possible pairings of frame-wise feature representations of sequences $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}'$, and $d^2_{\mathrm{base}}(\cdot, \cdot)$ may be the squared Euclidean distance.

However, the path distance $\langle \boldsymbol{\Pi}, \boldsymbol{D}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \rangle$ of path $\boldsymbol{\Pi}$ ignores the observation uncertainty of frame-wise feature representations by simply relying on the Euclidean distances stored in $\boldsymbol{D}$. Thus, we resort to the notion of the so-called aleatoric uncertainty known from a non-exhaustive list of works about uncertainty [Matthies, 2007; Kiureghian and Ditlevsen, 2009; Indrayan, 2008; Hüllermeier and Waegeman, 2021; Kendall and Gal, 2017].

Specifically, to capture the aleatoric uncertainty of the Euclidean distance (or regression, *etc*.), one should tune the observation noise parameter of sequences. Instead of the homoscedastic model (constant observation noise), we opt for the so-called heteroscedastic aleatoric uncertainty model (the observation noise may vary with each frame/sequence). To this end, we model each path distance by the product of likelihoods of Normal distributions (we also investigate other distributions in Sec. 7.8).



(a) $\text{sDTW}_{\gamma=0.01}$ (b) $\text{sDTW}_{\gamma=0.1}$ (c) $\text{uDTW}_{\gamma=0.01}$ (d) $\text{uDTW}_{\gamma=0.1}$ (e) uDTW uncert.

Figure 7.2: Plots (a)-(d) show paths of sDTW and uDTW (in white) for a pair of sequences. We power-normalized pixels of plots (by the power of 0.1) to see also darker paths better. With higher γ that controls softness, in (b) & (d) more paths become 'active' (fuzzy effect). In (c), uDTW has two possible routes *vs*. sDTW (a) due to uncertainty modeling. In (e), we visualise uncertainty $\boldsymbol{\Sigma}$. We binarize plot (c) and multiply it by the $\boldsymbol{\Sigma}$ to display uncertainty values on the path (white pixels = high uncertainty). The middle of the main path is deemed uncertain, which explains why an additional path merges in that region with the main path. See also the histogram of values of $\boldsymbol{\Sigma}$.

Our (soft) uncertainty-DTW takes the following generalized form:

$$\begin{cases} d^2_{\text{uDTW}}(\boldsymbol{D}, \boldsymbol{\Sigma}^{\dagger}) = \text{SoftMin}_{\gamma}\Big(\underbrace{\Big[\Big\langle \boldsymbol{\Pi}, \boldsymbol{D} \odot \boldsymbol{\Sigma}^{\dagger} \Big\rangle \Big]_{\boldsymbol{\Pi} \in \mathcal{A}_{\tau,\tau'}}}_{\boldsymbol{w}} \Big) & (7.2) \\ \Omega(\boldsymbol{\Sigma}) = \text{SoftMinSel}_{\gamma}\Big(\boldsymbol{w}, [\langle \boldsymbol{\Pi}, \log \boldsymbol{\Sigma} \rangle]_{\boldsymbol{\Pi} \in \mathcal{A}_{\tau,\tau'}} \Big), & (7.3) \\ \text{where } \boldsymbol{D} \equiv \boldsymbol{D}(\boldsymbol{\Psi}, \boldsymbol{\Psi}'),\ \boldsymbol{\Sigma} \equiv \boldsymbol{\Sigma}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \text{ and } \boldsymbol{\Sigma}^{\dagger} = \text{inv}(\boldsymbol{\Sigma}), \end{cases}$$

where $\odot$ is the Hadamard product, $\boldsymbol{\Sigma}^{\dagger}(\boldsymbol{\Psi}, \boldsymbol{\Psi}')$ is the element-wise inverse of matrix $\boldsymbol{\Sigma} \in \mathbb{R}_{+}^{\tau \times \tau'} \equiv [\sigma^2(\boldsymbol{\psi}_m, \boldsymbol{\psi}'_n)]_{(m,n) \in \mathcal{I}_{\tau} \times \mathcal{I}_{\tau'}}$ which contains pair-wise variances between all possible pairings of frame-wise feature representations from sequences $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}'$. $\text{SoftMin}_{\gamma}(\boldsymbol{\alpha}) = \sum_i \alpha_i \frac{\exp(-(\alpha_i - \mu_{\alpha})/\gamma)}{\sum_j \exp(-(\alpha_j - \mu_{\alpha})/\gamma)}$ with μ_{α} (the mean over coefficients of $\boldsymbol{\alpha}$) subtracted from each coefficient α_i to attain stability of the softmax (into which we feed $(\alpha_i - \mu_{\alpha})$). Moreover, $\text{SoftMinSel}_{\gamma}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_i \beta_i \frac{\exp(-(\alpha_i - \mu_{\alpha})/\gamma)}{\sum_j \exp(-(\alpha_j - \mu_{\alpha})/\gamma)}$ is a soft-selector returning $(\beta_{i^*}\colon i^* = \arg\min_i \alpha_i)$ if γ approaches zero.

Eq. (7.2) yields the uncertainty-weighted time warping distance $d^2_{\text{uDTW}}(\boldsymbol{D}, \boldsymbol{\Sigma}^{\dagger})$ between sequences $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}'$ because $\boldsymbol{D}$ and $\boldsymbol{\Sigma}^{\dagger}$ are both functions of $(\boldsymbol{\Psi}, \boldsymbol{\Psi}')$.

Eq. (7.3) provides the regularization penalty $\Omega(\boldsymbol{\Sigma})$ for sequences $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}'$ (as $\boldsymbol{\Sigma}$ is a function of $(\boldsymbol{\Psi}, \boldsymbol{\Psi}')$) which is the aggregation of log-variances along the path with the smallest distance, *i.e.*, path matrix $((\boldsymbol{\Pi}_{i^*} \in \{0,1\}^{\tau \times \tau'})\colon i^* = \arg\min_k w_k)$ if $\gamma = 0$, and vector $\boldsymbol{w}$ contains path-aggregated distances for all possible paths of the plan $\mathcal{A}_{\tau,\tau'}$.

Contributions. The celebrated DTW warps the matching path between a pair of sequences to recover the best matching distance under varying temporal within-class dynamics of each sequence. The recovered path, and the distance corresponding to that path, may be suboptimal if frame-wise (or block-wise) features contain noise (frames that are outliers, contain occlusions or large within-class object variations, *etc*.)

To this end, we propose several contributions:

i. We introduce the uncertainty-DTW, dubbed as uDTW, whose role is to take into account the uncertainty of in frame-wise (or block-wise) features by selecting the path which maximizes the Maximum Likelihood Estimation (MLE). The parameters (such as variance) of a distribution (*i.e.*, the Normal distribution) are thus used within MLE (and uDTW) to model the uncertainty.

ii. As pairs of sequences are often of different lengths, optimizing the free-form variable of variance is impossible. To that end, we equip each of our pipelines with SigmaNet, whose role is to take frames (or blocks) of sequences, and generate the variance end-to-end (the variance is parametrized by SigmaNet).

iii. We provide several pipelines that utilize uDTW for (1) forecasting the evolution of time series, (2) estimating the Fréchet mean of time series, (3) supervised few-shot action recognition, and (4) unsupervised few-shot action recognition.

Notations. $\mathcal{I}_\tau$ is the index set $\{1, 2, ..., \tau\}$. Concatenation of α_i into a vector $\boldsymbol{\alpha}$ is denoted by $[\alpha_i]_{i\in\mathcal{I}_I}$. Concatenation of α_{ij} into matrix $\mathbf{A}$ is denoted by $[\alpha_{ij}]_{(i,j)\in\mathcal{I}_I\times\mathcal{I}_J}$. Dot-product between two matrices equals the dot-product of vectorized $\boldsymbol{\Pi}$ and $\boldsymbol{D}$, that is $\langle\boldsymbol{\Pi}, \boldsymbol{D}\rangle \equiv \langle \mathrm{vec}(\boldsymbol{\Pi}), \mathrm{vec}(\boldsymbol{D})\rangle$. Mathcal symbols are sets, *e.g.*, $\mathcal{A}$ is a transportation plan, capitalized bold symbols are matrices, *e.g.*, $\boldsymbol{D}$ is the distance matrix, lowercase bold symbols are vectors, *e.g.*, $\boldsymbol{w}$ contains weighted distances. Regular fonts are scalars. Table 7.1 shows the notations with their short descriptions used in this chapter.

7.1.1 Similarity learning with uDTW

In further chapters, based on the distance in Eq. (7.2) and the regularization term in Eq. (7.3), we define specific loss functions for several problems such as forecasting the evolution of time series, clustering time series or even matching sequence pairs in few-shot action recognition. Below is an example of a generic similarity learning loss:

$$\arg\min_{\mathcal{P}} \sum_n \ell\left(d^2_{\mathrm{uDTW}}(\boldsymbol{D}(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n), \boldsymbol{\Sigma}^\dagger(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n)),\, \delta_n\right) + \beta\Omega(\boldsymbol{\Sigma}(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n)), \tag{7.4}$$

or

$$\arg\min_{\mathcal{P}, \boldsymbol{\Sigma}>0} \sum_n \ell\left(d^2_{\mathrm{uDTW}}(\boldsymbol{D}(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n), \boldsymbol{\Sigma}^\dagger),\, \delta_n\right) + \beta\Omega(\boldsymbol{\Sigma}), \tag{7.5}$$

where $\boldsymbol{\Psi}_n = f(\mathbf{X}_n; \mathcal{P})$ and $\boldsymbol{\Psi}'_n = f(\mathbf{X}'_n; \mathcal{P})$ are obtained from some backbone encoder $f(\cdot; \mathcal{P})$ with parameters $\mathcal{P}$ and $(\mathbf{X}_n, \mathbf{X}'_n) \in \mathcal{X}$ is a sequence pair to compare with the similarity label $\delta_n \in \{0, 1\}$ (where $\delta_n = 0$ if $y_n = y'_n$ and $\delta_n = 1$ otherwise), (y_n, y'_n) is a pair of class labels for $(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n)$, and $\beta \geq 0$ controls the penalty for high matching uncertainty. Figure 7.2 illustrates the impact of uncertainty on uDTW.

Table 7.1: Notations and their descriptions.

Notation	Description
$\boldsymbol{\Psi}$	Query feature maps
$\boldsymbol{\Psi}'$	Support feature maps
$\boldsymbol{\Pi}$	Path matrix
$\boldsymbol{D}(\cdot,\cdot)$	Pair-wise distances
$d_*^2(\cdot,\cdot)$	Distance functions and * can be base (squared Euclidean), DTW, sDTW or uDTW
γ	The relaxation parameter of sDTW/uDTW
τ	The number of temporal blocks for query
τ'	The number of temporal blocks for support
$\boldsymbol{\Sigma}$	Pair-wise variances between all possible pairs of two sequences
$\boldsymbol{\Sigma}^\dagger$	Element-wise inverse of $\boldsymbol{\Sigma}$
$f(\cdot;\cdot)$	Encoder function
$\mathcal{P}$	The set of parameters to learn
β	Regularization parameter
σ	Uncertainty parameter
$\mathbf{X}$	Query frames per block
$\mathbf{X}'$	Support frames per block
K	The size of dictionary
K'	The subset size for K' nearest anchors
$\mathbf{x}$	Time series for training
$\mathbf{x}'$	Time series for testing
$\boldsymbol{\mu}_c$	Class prototype for class c
$\Omega(\cdot)$	Regularization penalty
$\boldsymbol{\alpha}$	Coding vector
λ_{DL}	Learning rate for dictionary learning
λ_{EN}	Learning rate for encoder
$\boldsymbol{M}$	Dictionary anchors
B	The number of training episodes
N	The number of classes
Z	The number of samples from each class
J	The number of human body joints
d	Feature dimension after MLP
d'	Feature dimension (output of EN)
δ	The similarity label
N_c	The number of samples for class c

Note that minimizing Eq. (7.5) w.r.t. $(\mathcal{P}, \boldsymbol{\Sigma})$ assumes that $\boldsymbol{\Sigma} \in \mathbb{R}_{+}^{\tau \times \tau'}$ is a free variable to minimize over (derivation in Section 7.1.2). However, as sequence pairs vary in length, *i.e.*, $\tau \neq \tau'$, optimizing one global $\boldsymbol{\Sigma}$ is impossible (its size changes). Thus, for problems we tackle, we minimize loss functions with the distance/penalty in Eq. (7.4) and (7.5) where $\boldsymbol{\Sigma}$ is parametrized by $(\boldsymbol{\Psi}_n, \boldsymbol{\Psi}'_n)$:

$$d^2_{\text{uDTW}^\bullet}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \equiv d^2_{\text{uDTW}}(\boldsymbol{D}(\boldsymbol{\Psi}, \boldsymbol{\Psi}'), \boldsymbol{\Sigma}^{+}(\boldsymbol{\Psi}, \boldsymbol{\Psi}')), \tag{7.6}$$

$$\Omega_\bullet(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \equiv \Omega(\boldsymbol{\Sigma}(\boldsymbol{\Psi}, \boldsymbol{\Psi}')). \tag{7.7}$$

To that end, we devise a small MLP unit $\sigma(\cdot; \mathcal{P}_\sigma)$ or $\sigma(\cdot, \cdot; \mathcal{P}_\sigma)$ and obtain:

$$\boldsymbol{\Sigma} = 0.5 \cdot [(\sigma^2(\boldsymbol{\psi}_m; \mathcal{P}_\sigma) + \sigma^2(\boldsymbol{\psi}'_n; \mathcal{P}_\sigma))]_{(m,n) \in \mathcal{I}_\tau \times \mathcal{I}_{\tau'}} \tag{7.8}$$

or

$$\boldsymbol{\Sigma}' = [\sigma^2(\boldsymbol{\psi}_m, \boldsymbol{\psi}'_n; \mathcal{P}_\sigma)]_{(m,n) \in \mathcal{I}_\tau \times \mathcal{I}_{\tau'}}, \tag{7.9}$$

where Eq. (7.8) uses additive variance terms generated for individual frames $\boldsymbol{\psi}_m$ and $\boldsymbol{\psi}'_n$, whereas (7.9) is a jointly generated variance for $(\boldsymbol{\psi}_m, \boldsymbol{\psi}'_n)$.

7.1.2 Derivation of uDTW

We proceed by modeling an arbitrary path $\boldsymbol{\Pi}_i$ from the transportation plan of $\mathcal{A}_{\tau,\tau'}$ as the following Maximum Likelihood Estimation (MLE) problem:

$$\underset{\{\sigma_{mn}\}_{(m,n) \in \boldsymbol{\Pi}_i}}{\arg\max} \prod_{(m,n) \in \boldsymbol{\Pi}_i} p(\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|, \sigma^2_{mn}), \tag{7.10}$$

where p may be some arbitrary distribution, σ are distribution parameters, and $\|\cdot\|$ is an arbitrary norm. For the Normal distribution $\mathcal{N}$ which relies on the squared Euclidean distance $\|\cdot\|_2^2$, we have:

$$\underset{\{\sigma_{mn}\}_{(m,n) \in \boldsymbol{\Pi}_i}}{\arg\max} \prod_{(m,n) \in \boldsymbol{\Pi}_i} \mathcal{N}(\boldsymbol{\psi}_m; \boldsymbol{\psi}'_n, \sigma^2_{mn}) \tag{7.11}$$

$$= \underset{\{\sigma_{mn}\}_{(m,n) \in \boldsymbol{\Pi}_i}}{\arg\max} \log \prod_{(m,n) \in \boldsymbol{\Pi}_i} \frac{1}{(2\pi)^{\frac{d'}{2}} \sigma^{d'}} \exp\left(-\frac{\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|_2^2}{\sigma^2_{mn}}\right) \tag{7.12}$$

$$= \underset{\{\sigma_{mn}\}_{(m,n) \in \boldsymbol{\Pi}_i}}{\arg\max} \sum_{(m,n) \in \boldsymbol{\Pi}_i} -\frac{d'}{2} \log(2\pi) - d' \log(\sigma) - \frac{\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|_2^2}{\sigma^2_{mn}} \tag{7.13}$$

$$= \underset{\{\sigma_{mn}\}_{(m,n) \in \boldsymbol{\Pi}_i}}{\arg\min} \sum_{(m,n) \in \boldsymbol{\Pi}_i} d' \log(\sigma) + \frac{\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|_2^2}{\sigma^2_{mn}}, \tag{7.14}$$

where d' is the length of feature vectors $\boldsymbol{\psi}$. Having recovered uncertainty parameters $\{\sigma_{mn}\}_{(m,n) \in \boldsymbol{\Pi}_i}$, we obtain a combination of penalty terms and reweighted squared

Euclidean distances:

$$\beta\Omega_{\boldsymbol{\Pi}_i} + d^2_{\boldsymbol{\Pi}_i} = \sum_{(m,n)\in\boldsymbol{\Pi}_i} \beta\log(\sigma_{mn}) + \frac{\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|_2^2}{\sigma^2_{mn}}, \tag{7.15}$$

where $\beta \geq 0$ (generally $\beta \neq d'$) adjusts the penalty for large uncertainty. Separating the uncertainty penalty $\log(\sigma_{mn})$ from the uncertainty-weighted distance (both aggregated along path $\boldsymbol{\Pi}_i$) yields:

$$\begin{cases} d^2_{\boldsymbol{\Pi}_i} = \langle \boldsymbol{\Pi}_i, \boldsymbol{D}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \odot \boldsymbol{\Sigma}^\dagger \rangle \\ \Omega_{\boldsymbol{\Pi}_i} = \langle \boldsymbol{\Pi}_i, \log\boldsymbol{\Sigma} \rangle \,, \end{cases} \tag{7.16}$$

where $\boldsymbol{D} \in \mathbb{R}_+^{\tau\times\tau'} \equiv \left[\frac{d_2^2(\boldsymbol{\psi}_m, \boldsymbol{\psi}'_n)}{\sigma^2_{mn}}\right]_{(m,n)\in\mathcal{I}_\tau\times\mathcal{I}_{\tau'}}$ and $\boldsymbol{\Sigma} \in \mathbb{R}_+^{\tau\times\tau'} \equiv [\sigma^2_{mn}]_{(m,n)\in\mathcal{I}_\tau\times\mathcal{I}_{\tau'}}$. Derivations for other distributions, *i.e.*, Laplace or Cauchy, follow the same reasoning.

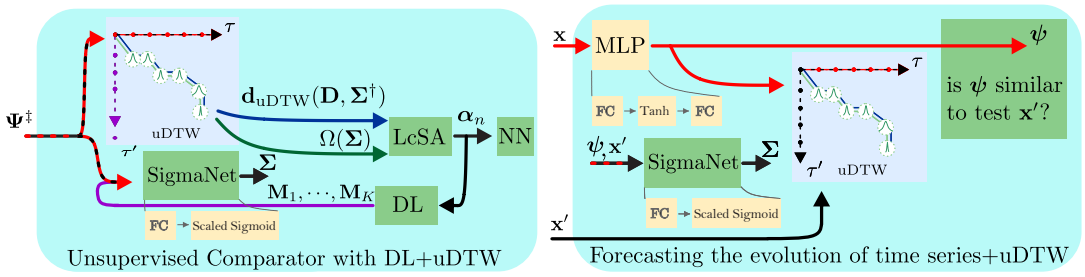
7.2 Related Work

Different flavors of Dynamic Time Warping. DTW [Cuturi, 2011], which seeks a minimum cost alignment between time series is computed by dynamic programming in quadratic time, is not differentiable and is known to get trapped in bad local minima. In contrast, soft-DTW (sDTW) [Cuturi and Blondel, 2017] addresses the above issues by replacing the minimum over alignments with a soft minimum, which has the effect of inducing a 'likelihood' field over all possible alignments. However, sDTW has been successfully applied in many computer vision tasks including audio/music score alignment [Mensch and Blondel, 2018], action recognition [Su and Wen, 2022; Cao et al., 2020], and end-to-end differentiable text-to-speech synthesis [Donahue et al., 2021]. Despite its successes, sDTW has some limitations: (i) it can be negative when used as a loss (ii) it may still get trapped in bad local minima. Thus, soft-DTW divergences (sDTW div.) [Blondel et al., 2021], inspired by sDTW, attempts to overcome such issues.

Other approaches inspired by DTW have been used to improve the inference or adapt to modified or additional constraints, *i.e.*, OPT [Su and Hua, 2019] and OWDA [Su et al., 2019] treat the alignment as the optimal transport problem with temporal regularization. TAP [Su and Wen, 2022] directly predicts the alignment through a lightweight CNN, thus is does not follow a principled transportation plan, and is not guaranteed to find a minimum cost path.

Our uDTW differs from these methods in that the transportation plan is executed under the uncertainty estimation, thus various feature-level noises and outliers are less likely to lead to the selection of a sub-optimal cost path.

Alignment-based time series problems. Distance between sequences plays an important role in time series retrieval [Su et al., 2019], forecasting [Cuturi and Blondel, 2017; Blondel et al., 2021], classification [Cuturi and Blondel, 2017; Blondel et al., 2021; Dempster et al., 2021; Yang et al., 2021], clustering [García-García et al., 2009; Sakoe and Chiba, 1978], *etc*. Various temporal nuisance noises such as initial states, different



(a) (b)

Figure 7.3: In (a) is the unsupervised comparator for unsupervised few-shot action recognition. The unsupervised head is wired with the Encoding Network from Figure 7.1, and trained from scratch without labels. In (b) is the pipeline for forecasting the evolution of time series (a.k.a. multistep-ahead prediction).

sampling rates, local distortions, and execution speeds make the measurement of distance between sequences difficult. To tackle these issues, typical feature-based methods use RNNs to encode sequences and measure the distance between corresponding features [Ramachandran et al., 2018]. Other existing methods [Wang, 2017; Wang et al., 2019c; Koniusz et al., 2020] either encode each sequence into features that are invariant to temporal variations [Abid and Zou, 2018; Lohit et al., 2019] or adopt alignment for temporal correspondence calibration [Su and Hua, 2019]. However, none of these methods is modeling the aleatoric uncertainty. As we model it along the time warping path, the observation noise may vary with each frame or block.

Few-shot action recognition. Most existing few-shot action recognition methods [Wang et al., 2019b,d; Wang and Koniusz, 2021] follow the metric learning paradigm. Signal Level Deep Metric Learning [Memmesheimer et al., 2020] and Skeleton-DML [Memmesheimer et al., 2021] one-shot FSL approaches encode signals into images, extract features using a deep residual CNN and apply multi-similarity miner losses. TAEN [Ben-Ari et al., 2021] and FAN [Tan and Yang, 2019] encode actions into representations and apply vector-wise metrics.

Most methods identify the importance of temporal alignment for handling the non-linear temporal variations, and various alignment-based models are proposed to compare the sequence pairs, *e.g.*, permutation-invariant spatial-temporal attention reweighted distance in ARN [Zhang et al., 2020a], a variant of DTW used in OTAM [Cao et al., 2020], temporal attentive relation network [Mina et al., 2019], a two-stage temporal alignment network (TA2N) [Li et al., 2021c], a temporal CrossTransformer [Perrett et al., 2021], a learnable sequence matching distance called TAP [Su and Wen, 2022].

In all cases, temporal alignment is a well-recognized tool, however lacking the uncertainty modeling, which impacts the quality of alignment. Such a gap in the literature inspires our work on uncertainty-DTW.

7.3 Pipeline Formulations

Below we provide our several pipeline formulations for which uDTW is used as an indispensable component embedded with the goal of measuring the distance for warped paths under uncertainty.

7.3.1 Few-shot Action Recognition

For both supervised and unsupervised few-shot pipelines, we employ the Encoder Network (EN) and the Supervised Comparator (similarity learning) as in Figure 7.1, or Unupervised Comparator (based on dictionary learning) as in Figure 7.3a.

Encoding Network (EN). Our EN contains a simple 3-layer MLP unit (FC, ReLU, FC, ReLU, Dropout, FC), GNN, with transformer [Dosovitskiy et al., 2020] and FC. The MLP unit takes M neighboring frames, each with J skeleton body joints given by Cartesian coordinates (x, y, z), forming one temporal block[1]. In total, depending on stride S, we obtain some τ temporal blocks (each block captures the short temporal dependency), whereas the long temporal dependency will be modeled by uDTW. Each temporal block is encoded by the MLP into a $d \times J$ dimensional feature map. Subsequently, query feature maps of size τ and support feature maps of size τ' are forwarded to a simple linear GNN model, and transformer, and an FC layer, which returns $\mathbf{\Psi} \in \mathbb{R}^{d' \times \tau}$ query feature maps and $\mathbf{\Psi}' \in \mathbb{R}^{d' \times \tau'}$ support feature maps. Such encoded feature maps are passed to the Supervised Comparator with uDTW.

Let support maps $\mathbf{\Psi}' \equiv [f(\boldsymbol{X}'_1; \mathcal{P}), ..., f(\boldsymbol{X}'_{\tau'}; \mathcal{P})] \in \mathbb{R}^{d' \times \tau'}$ and query maps $\mathbf{\Psi} \equiv [f(\boldsymbol{X}_1; \mathcal{P}), ..., f(\boldsymbol{X}_{\tau}; \mathcal{P})] \in \mathbb{R}^{d' \times \tau}$ for query and support frames per block $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{3 \times J \times M}$. Define $f(\mathbf{X}; \mathcal{P}) = \text{FC}(\text{Transf}(\text{S}^2\text{GC}(\text{MLP}(\mathbf{X}; \mathcal{P}_{MLP}); \mathcal{P}_{S^2GC}); \mathcal{P}_{Transf}); \mathcal{P}_{FC})$ where $\mathcal{P} \equiv [\mathcal{P}_{MLP}, \mathcal{P}_{S^2GC}, \mathcal{P}_{Transf}, \mathcal{P}_{FC}, \mathcal{P}_{SN}]$ is the set of parameters of EN, where $\mathcal{P}_{SN}$ are parameters of SigmaNet, and S^2GC is a Simple Spectral Graph Convolution (S^2GC) [Zhu and Koniusz, 2021b] whose details are in Sec. 7.7.3.

Supervised Few-shot Action Recognition. For the N-way Z-shot problem, we have one query feature map and $N \times Z$ support feature maps per episode. We form a mini-batch containing B episodes. We have query feature maps $\{\mathbf{\Psi}_b\}_{b \in \mathcal{I}_B}$ and support feature maps $\{\mathbf{\Psi}'_{b,n,z}\}_{b \in \mathcal{I}_B, n \in \mathcal{I}_N, z \in \mathcal{I}_Z}$. Moreover, $\mathbf{\Psi}_b$ and $\mathbf{\Psi}'_{b,1,:}$ share the same class (drawn from N classes per episode), forming the subset $C^\ddagger \equiv \{c_1, ..., c_N\} \subset \mathcal{I}_C \equiv \mathcal{C}$. To be precise, labels $y(\mathbf{\Psi}_b) = y(\mathbf{\Psi}'_{b,1,z}), \forall b \in \mathcal{I}_B, z \in \mathcal{I}_Z$ while $y(\mathbf{\Psi}_b) \neq y(\mathbf{\Psi}'_{b,n,z}), \forall b \in \mathcal{I}_B, n \in \mathcal{I}_N \setminus \{1\}, z \in \mathcal{I}_Z$. Thus the similarity label $\delta_1 = 0$, whereas $\delta_{n \neq 1} = 1$. Note that the selection of $C^\ddagger$ per episode is random. For the N-way Z-shot protocol, the Supervised Comparator is minimized w.r.t. $\mathcal{P}$ ($\mathbf{\Psi}_b$ and $\mathbf{\Psi}'$ depend on $\mathcal{P}$) as:

$$\arg\min_{\mathcal{P}} \sum_{b \in \mathcal{I}_B} \sum_{n \in \mathcal{I}_N} \sum_{z \in \mathcal{I}_Z} \left(d^2_{\text{uDTW}^\bullet}(\mathbf{\Psi}_b, \mathbf{\Psi}'_{b,n,z}) - \delta_n\right)^2 + \beta \Omega_\bullet(\mathbf{\Psi}_b, \mathbf{\Psi}'_{b,n,z}). \tag{7.17}$$

Unsupervised Few-shot Action Recognition. Below we propose a very simple unsupervised variant with so-called Unsupervised Comparator. The key idea is that

[1] We use temporal blocks as they were shown more robust than frame-wise FSAR [Zhang et al., 2020a] models.

with uDTW, invariant to local temporal speed changes can be used to learn a dictionary which, with some dictionary coding method should outperform at reconstructing the sequences. This means we can learn an unsupervised comparator by projecting sequences onto the dictionary space. To this end, let the protocol remain as for the supervised few-shot learning with the exception that class labels are not used during training, and only support images in testing are labeled for sake of evaluation the accuracy by deciding which support representation each query is the closest to in the nearest neighbor sense.

Firstly, in each training episode, we combine the query sequences $\mathbf{\Psi}_b$ with the support sequences $\mathbf{\Psi}'_{b,n,z}$ into episode sequences denoted as $\mathbf{\Psi}^{\ddagger}_{b,n}$ where $b \in \mathcal{I}_B$ enumerates over B episodes, and $n \in \mathcal{I}_{(N \cdot Z+1)}$. For the feature coding, we use Locality-constrained Soft Assignment (LCSA) [Liu et al., 2011b; Koniusz and Mikolajczyk, 2011; Koniusz et al., 2013b] and a simple dictionary update based on the least squares computation.

For each episode $b \in \mathcal{I}_B$, we iterate over the following three steps:

i. The LCSA coding step which expresses each $\mathbf{\Psi}^{\ddagger}_{b,n}$ as $\boldsymbol{\alpha}_{b,n} \in \mathbb{R}^K_+$ that assign $\mathbf{\Psi}^{\ddagger}_{b,n}$ into a dictionary with K sequences $\boldsymbol{M}_1, ..., \boldsymbol{M}_K \in \mathbb{R}^{d' \times \tau'}$ (dictionary anchors):

$$\forall_{k,n},\ \alpha_{k,b,n} = \begin{cases} \dfrac{\exp\left(-\frac{1}{\gamma'} d^2_{\text{uDTW}^\bullet}\left(\mathbf{\Psi}^{\ddagger}_{b,n}, \boldsymbol{M}_k\right)\right)}{\sum\limits_{l \in \mathcal{M}(\mathbf{\Psi}^{\ddagger}_{b,n}; K')} \exp\left(-\frac{1}{\gamma'} d^2_{\text{uDTW}^\bullet}\left(\mathbf{\Psi}^{\ddagger}_{b,n}, \boldsymbol{M}_l\right)\right)} & \text{if } \boldsymbol{M}_k \in \mathcal{M}\left(\mathbf{\Psi}^{\ddagger}_{b,n}; K'\right), \\ 0 & \text{otherwise,} \end{cases} \tag{7.18}$$

where $0 < K' \leq K$ is a subset size for K' nearest anchors of $\mathbf{\Psi}^{\ddagger}_{b,n}$ retrieved by operation $\mathcal{M}(\mathbf{\Psi}^{\ddagger}_{b,n}; K')$ (based on uDTW) from $\boldsymbol{M}_1, ..., \boldsymbol{M}_K$, τ' is set to the mean of τ (over training set), and $\gamma' = 0.7$ is a so-called smoothing factor;

ii. The dictionary update step updates $\boldsymbol{M}_1, ..., \boldsymbol{M}_K$ given $\boldsymbol{\alpha}_{b,n}$ from Eq. (7.18):

```
for i=1,...,dict_iter:
```

$$\forall_k,\ \boldsymbol{M}_k := \boldsymbol{M}_k - \lambda_{\text{DL}} \sum_{n=1}^{NZ+1} \nabla_{\boldsymbol{M}_k} d^2_{\text{uDTW}^\bullet}\left(\mathbf{\Psi}^{\ddagger}_{b,n}, \sum_{l=1}^{K} \alpha_{l,b,n} \boldsymbol{M}_l\right), \tag{7.19}$$

where `dict_iter` is set to 10 and $\lambda_{\text{DL}} = 0.001$;

iii. The main loss for the Feature Encoder update step is given as ($\lambda_{\text{EN}} = 0.001$):

$$\mathcal{P} := \mathcal{P} - \lambda_{\text{EN}} \sum_{n=1}^{NZ+1} \nabla_{\mathcal{P}} d^2_{\text{uDTW}^\bullet}\left(\mathbf{\Psi}^{\ddagger}_{b,n}, \boldsymbol{M}'\right) + \beta \Omega_\bullet\left(\mathbf{\Psi}^{\ddagger}_{b,n}, \boldsymbol{M}'\right), \tag{7.20}$$
$$\text{where } \boldsymbol{M}' = \sum_{l=1}^{K} \alpha_{l,b,n} \boldsymbol{M}_l.$$

During testing, we use the learnt dictionary, pass new support and query sequences via Eq. (7.18) and obtain $\boldsymbol{\alpha}$ codes. Subsequently, we compare the LCSA code of the query sequence with LCSA codes of support sequences via the histogram intersection

kernel. The closest match in the support set determines the test label of the query sequence.

7.3.2 Time Series Forecasting and Classification

One of key applications of DTW and sDTW is learning with time series, including forecasting the evolution of time series as in Figure 7.3b and time series classification.
Forecasting the Evolution of Time Series. Let $\mathbf{x} \in \mathbb{R}^t$ and $\mathbf{x}' \in \mathbb{R}^{\tau - t}$ be the training and testing parts of one time series corresponding to timesteps $1, ..., t$ and $t+1, ..., \tau$, respectively. The goal is to learn encoder $f(\mathbf{x}; \mathcal{P}) \in \mathbb{R}^{\tau - t}$ which will be able to take $\mathbf{x}$ as input, learn to translate it to $\mathbf{x}'$. Figure 7.3b show the full pipeline. We took the Encoding Network from the original soft-DTW pipeline [Cuturi and Blondel, 2017]. Our training objective is:

$$\arg\min_{\mathcal{P}} \sum_{n \in \mathcal{I}_N} d^2_{\text{uDTW}^\bullet}(\boldsymbol{\psi}_n, \mathbf{x}'_n) + \beta \Omega_\bullet(\boldsymbol{\psi}_n, \mathbf{x}'_n), \tag{7.21}$$

where $\boldsymbol{\psi} = f(\mathbf{x}; \mathcal{P})$ and N is the number of training time series, $\mathcal{P} \equiv [\mathcal{P}_{MLP}, \mathcal{P}_{SN}]$ is the set of parameters of EN and SigmaNet. In order to obtain $\boldsymbol{\Sigma}$, vectors $\boldsymbol{\psi}$ and $\mathbf{x}'$ are passed via SigmaNet. After training, at the test time, for a previously unseen testing sample $\mathbf{x}$, $f(\cdot)$ has to predict the remaining part of the time series given by $\mathbf{x}'$.
Time Series Classification. Below we follow the setting for this classical task according to the original soft-DTW paper [Cuturi and Blondel, 2017], and define the **nearest centroid** classifier. We estimate the Fréchet mean of training time series of each class separately. We do not use any Encoding Network but the raw features. Let $\mathbf{x} \in \mathbb{R}^\tau$ be training samples and $\boldsymbol{\mu} \in \mathbb{R}^{\tau'}$ be class prototypes (τ' is set to average of τ across all classes). We have:

$$\forall_c, \ \arg\min_{\mathcal{P}} \sum_{n \in \mathcal{I}_{N_c}} d^2_{\text{uDTW}^\bullet}(\mathbf{x}_n, \boldsymbol{\mu}_c) + \beta \Omega_\bullet(\mathbf{x}_n, \boldsymbol{\mu}_c), \tag{7.22}$$

where N_c is the number of samples for class $c \in \mathcal{I}_C$ and $\mathcal{P} \equiv [\mathcal{P}_{SN}, \boldsymbol{\mu}_c]$. During testing, we apply $\arg\min_{c \in \mathcal{I}_C} d^2_{\text{uDTW}^\bullet}(\mathbf{x}, \boldsymbol{\mu}_c) + \beta \Omega_\bullet(\mathbf{x}, \boldsymbol{\mu}_c)$ for $\mathbf{x}$ to find its nearest neighbor and label it. The variances of $\mathbf{x}$ are recovered through SigmaNet while variances of $\boldsymbol{\mu}_c$ were obtained during training (adding both yields $\boldsymbol{\Sigma}$ of testing sample). As in soft-DTW paper [Cuturi and Blondel, 2017], we use uDTW to directly find the **nearest neighbor** of $\mathbf{x}$ across training samples to label $\mathbf{x}$ (for uncertainty, we use SigmaNet from the nearest centroid task).

7.4 Experiments

Below we apply uDTW in several scenarios such as (i) forecasting the evolution of time series, (ii) clustering/classifying time series, (iii) supervised few-shot action recognition, and (iv) unsupervised few-shot action recognition.
Datasets. The following datasets are used in our experiments:

i. *UCR* archive [Dau et al., 2018] is a dataset for time series classification archive. This dataset contains a wide variety of fields (astronomy, geology, medical imaging) and lengths, and can be used for time series classification/clustering and forecasting tasks.

ii. *NTU RGB+D (NTU-60)* [Shahroudy et al., 2016a] contains 56,880 video sequences and over 4 million frames. NTU-60 has variable sequence lengths and high intra-class variations.

iii. *NTU RGB+D 120 (NTU-120)* [Liu et al., 2019a], an extension of NTU-60, contains 120 action classes (daily/health-related), and 114,480 RGB+D video samples captured with 106 distinct human subjects from 155 different camera viewpoints.

iv. *Kinetics* [Kay et al., 2017] is a large-scale collection of 650,000 video clips that cover 400/600/700 human action classes. It includes human-object interactions such as *playing instruments*, as well as human-human interactions such as *shaking hands* and *hugging*. We follow approach [Yan et al., 2018] and use the estimated joint locations in the pixel coordinate system as the input to our pipeline. As OpenPose produces the 2D body joint coordinates and Kinetics-400 does not offer multiview or depth data, we use a network of Martinez et al. [Martinez et al., 2017] pre-trained on Human3.6M [Catalin et al., 2014], combined with the 2D OpenPose output to estimate 3D coordinates from 2D coordinates. The 2D OpenPose and the latter network give us (x, y) and z coordinates, respectively.

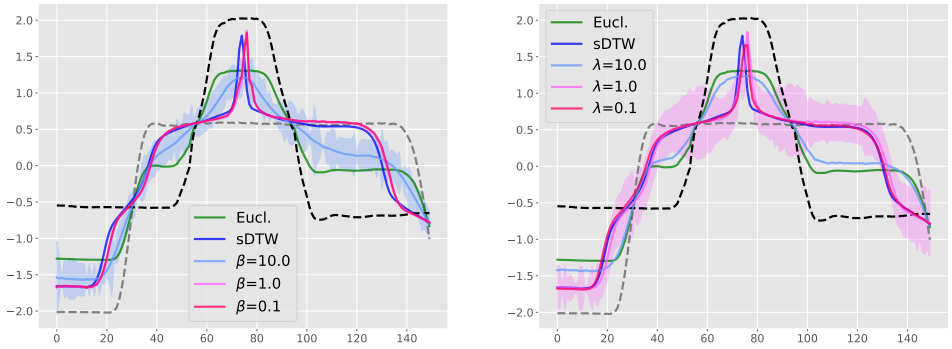
7.4.1 Fréchet Mean of Time Series

Below, we visually inspect the Fréchet mean for the Euclidean, sDTW and our uDTW distance, respectively.

Experimental setup. We follow the protocol of soft-DTW paper [Cuturi and Blondel, 2017]. For each dataset in UCR, we choose a class at random, pick 10 time series from the selected class to compute its barycenter. We use L-BFGS [Liu and Nocedal, 1989] to minimise the proposed uDTW barycenter objective. We set the maximum number of iterations to 100.

Qualitative results. We first perform averaging between two time series (Figure 7.4). We notice that averaging under the uDTW yields substantially different results than those obtained with the Euclidean and sDTW geometry.

Figure 7.5 shows the barycenters obtained using sDTW and our uDTW. We observe that our uDTW yields more reasonable barycenters than sDTW even when large γ are used, *e.g.*, for $\gamma = 10$ (right column of plots in Figure 7.5), the change points of red curve look sharper. We also notice that both uDTW and sDTW with low smoothing parameter $\gamma = 0.1$ can get stuck in some bad local minima, but our uDTW has fewer sharp peaks compared with sDTW (barycenters of uDTW are improved by the uncertainty measure). Moreover, higher γ values smooth the barycenter but introducing higher uncertainty (see uncertainty visualization around the barycenters by comparing, *e.g.*, $\gamma = 0.1$ *vs.* $\gamma = 10.0$). With $\gamma = 1$, the barycenters of sDTW and uDTW match well with the time series. More visualizations can be found in Sec. 7.5.



(a) β (where $\lambda = 1$) (b) λ (where $\beta = 10$)

Figure 7.4: Interpolation between two time series (grey and black dashed lines) on the Gun Point dataset. We compute the barycenter by solving $\arg\min_{\boldsymbol{\mu},\sigma_\mu} \sum_{n=1}^{2} d^2_{\text{uDTW}}(\boldsymbol{D}, \boldsymbol{\Sigma}^\dagger) + \beta\Omega(\boldsymbol{\Sigma}) + \lambda\Omega'(\boldsymbol{\Sigma})$ where $\boldsymbol{D} = (\mathbf{x}_n\mathbf{1}^\top - \mathbf{1}\boldsymbol{\mu}^\top)^2$ and $\boldsymbol{\Sigma} = \mathbf{1}\mathbf{1}^\top + \mathbf{1}\boldsymbol{\sigma}_\mu^\top$ where $\mathbf{x}_n$ is the given n-th time series. $\beta \geq 0$ controls the penalty for high matching uncertainty, Ω' is defined as in Eq. (7.3) but element-wise $\log\boldsymbol{\Sigma}$ is replaced by element-wise $(\boldsymbol{\Sigma}-1)^2$ so that $\lambda \geq 0$ favours uncertainty to remain close to one. β and λ control the uncertainty estimation and yield different barycenters than the Euclidean (green color) and sDTW (blue color) distances. As Ω and Ω' act similar, we only use Ω in our experiments.

7.4.2 Classification of Time Series

In this section, we devise the nearest neighbor and nearest centroid classifiers [Hastie et al., 2001] with uDTW, as detailed in Section 7.3. For the K-nearest neighbor classifier, we used softmax for the final decision. See Sec. 7.7.4 for details.

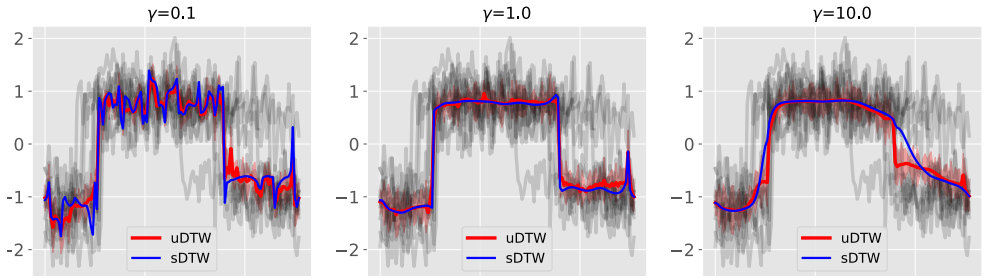
Experimental setup. We use 50% of the data for training, 25% for validation and 25% for testing. We report $K = 1$, 2 and 3 for the nearest neighbor classifier.

Quantitative results. Table 7.2 shows a comparison of our uDTW versus Euclidean, DTW, sDTW, and sDTW div. Unsurprisingly, the use of uDTW for barycenter computation improves the accuracy of the nearest centroid classifier, and it outperforms sDTW div. by $\sim 2\%$. Moreover, uDTW boosts results for the nearest neighbor classifier given K=1, 2 and 3 by 1.4%, 1.7% and 3.2%, respectively, compared to sDTW div.

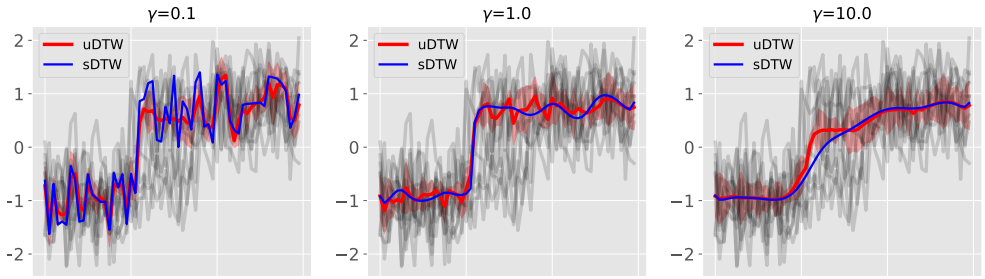
7.4.3 Forecasting the Evolution of Time Series

Experimental setup. We use the training and test sets pre-defined in the UCR archive. For both training and test, we use the first 60% of timesteps of series as input and the remaining 40% as output, ignoring the class information.

Qualitative results. The visualization of the predictions are given in Figure 7.6. Although the predictions under the sDTW and uDTW losses sometimes agree with each other, they can be visibly different. Predictions under uDTW can confidently predict the abrupt and sharp changes. More visualizations can be found in Sec. C.



(a) CBF



(b) Synthetic Control

Figure 7.5: Comparison of barycenter based on sDTW or uDTW on CBF and Synthetic Control. We visualize uncertainty around the barycenters in red color for uDTW. Our uDTW generates reasonable barycenters even when higher γ values are used, *e.g.*, $\gamma = 10.0$. Higher γ value leads to smooth barycenter but introducing higher uncertainty.

Quantitative results. We also provide quantitive results to validate the effectiveness of uDTW. We use ECG5000 dataset from the UCR archive which is composed of 5000 electrocardiograms (ECG) (500 for training and 4500 for testing) of length 140. To better evaluate the predictions, we use 2 different metrics (i) MSE for the predicted errors of each time step (ii) DTW, sDTW div. and uDTW for comparing the 'shape' of time series. We use such shape metrics for evaluation as the length of time series generally varies, and the MSE metric may lead to biased results which ignore the shape trend of time series. We then use the Student's t-test (with significance level 0.05) to highlight the best performance in each experiment (averaged over 100 runs). Table 7.3 shows that our uDTW achieves almost the best performance on both MSE and shape evaluation metrics (lower score is better).

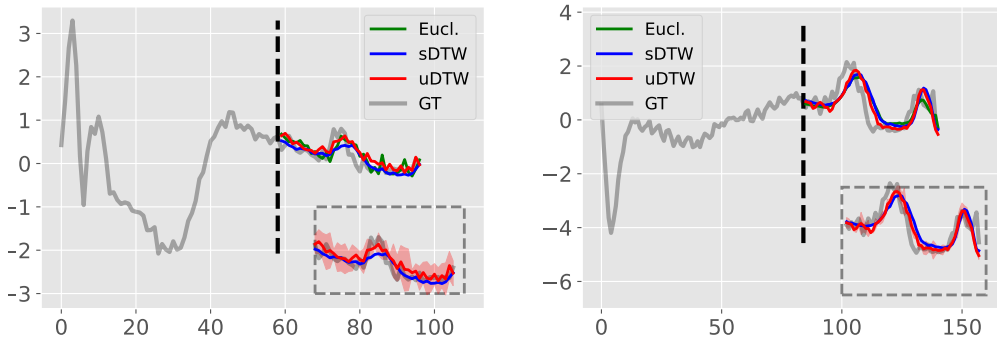
7.4.4 Few-shot Action Recognition

Below, we use uDTW as a distance in our objectives for few-shot action recognition (AR) tasks. We implement supervised and unsupervised pipelines (which is also novel).

Experimental setup. For NTU-120, we follow the standard one-shot protocols [Liu et al., 2019a]. Base on this protocol, we create a similar one-shot protocol for NTU-60, with 50/10 action classes used for training/testing respectively (see Sec. B for details).

Table 7.2: Classification accuracy (mean±std) on UCR archive by the nearest neighbor and the nearest centroid classifiers. In the column we indicate which distance was used for computing the class prototypes. K is the number of nearest neighbors in this context.

	Nearest neighbor			**Nearest centroid**
	$K=1$	$K=3$	$K=5$	
Euclidean	71.2±17.5	72.3±18.1	73.0±16.7	61.3±20.1
DTW [Cuturi, 2011]	74.2±16.6	75.0±17.0	75.4±15.8	65.9±18.8
sDTW [Cuturi and Blondel, 2017]	76.2±16.6	77.2±15.9	78.0±16.5	70.5±17.6
sDTW div. [Blondel et al., 2021]	78.6±16.2	79.5±16.7	80.1±16.5	70.9±17.8
uDTW	80.0±15.0	81.2±17.8	83.3±16.2	72.2±16.0



(a) ECG200 (b) ECG5000

Figure 7.6: Given the first part of a time series, we train 3 multi-layer perception (MLP) to predict the remaining part, we use the Euclidean, sDTW or uDTW distance per MLP. We use ECG200 and ECG5000 in UCR archive, and display the prediction obtained for the given test sample with either of these 3 distances and the ground truth (GT). Oftentimes, we observe that uDTW helps predict the sudden changes well.

We also evaluate the model on both 2D and 3D Kinetics-skeleton. We split the whole Kinetics-skeleton into 200 actions for training (the rest is used for testing). We choose Matching Nets (MatchNets) and Prototypical Net (ProtoNet) as baselines as these two models are very popular baselines, and we adapt these methods to skeleton-based action recognition. We reshape and resize each video block into 224×224 color image, and pass this image into MatchNets and ProtoNet to learn the feature representation per video block. We compare uDTW *vs.* Euclidean, sDTW, sDTW div. and recent TAP.

Quantitative results. Table 7.4, 7.5 and 7.6 show that our uDTW performs better than sDTW and sDTW div. on both supervised and unsupervised few-shot action recognition. On Kinetics-skeleton dataset, we gain 2.4% and 4.4% improvements on 3D skeletons for supervised and unsupervised settings. On supervised setting, we outperform TAP by $\sim$ 4% and 2% on NTU-60 and NTU-120 respectively. Moreover, we outperform sDTW by $\sim$ 2% and 3% on NTU-60 and NTU-120 for the unsupervised setting. More evaluations on few-shot action recognition are in Sec. 7.8.

Table 7.3: Time series forecasting results evaluated with MSE, DTW, sDTW div. and uDTW metrics on ECG5000, averaged over 100 runs (mean±std). Best method(s) are highlighted in bold using Student's *t*-test. Column-wise distances indicate the distance used during training. Row-wise distances indicate the distance used to compare prediction with the groundtruth at the test time (lower values are better).

	MSE	DTW	sDTW div.	uDTW
Euclidean	**32.1±1.62**	20.0±0.18	15.3±0.16	14.4±0.18
sDTW [Cuturi and Blondel, 2017]	38.6±6.30	**17.2±0.80**	22.6±3.59	32.1±2.25
sDTW div. [Blondel et al., 2021]	24.6±1.37	38.9±5.33	**20.0±2.44**	15.4±1.62
uDTW	23.0±1.22	**16.7±0.08**	**16.8±1.62**	**8.27±0.79**

Table 7.4: Evaluations on NTU-60.

#classes	10	20	30	40	50
Supervised					
MatchNets [Vinyals et al., 2016]	46.1	48.6	53.3	56.3	58.8
ProtoNet [Snell et al., 2017]	47.2	51.1	54.3	58.9	63.0
TAP [Su and Wen, 2022]	54.2	57.3	61.7	64.7	68.3
Euclidean	38.5	42.2	45.1	48.3	50.9
sDTW [Cuturi and Blondel, 2017]	53.7	56.2	60.0	63.9	67.8
sDTW div. [Blondel et al., 2021]	54.0	57.3	62.1	65.7	69.0
uDTW	56.9	61.2	64.8	68.3	72.4
Unsupervised					
Euclidean	20.9	23.7	26.3	30.0	33.1
sDTW [Cuturi and Blondel, 2017]	35.6	45.2	53.3	56.7	61.7
sDTW div. [Blondel et al., 2021]	36.0	46.1	54.0	57.2	62.0
uDTW	37.0	48.3	55.3	58.0	63.3

Table 7.5: Evaluations on NTU-120.

#classes	20	40	60	80	100
Supervised					
MatchNets [Vinyals et al., 2016]	20.5	23.4	25.1	28.7	30.0
ProtoNet [Snell et al., 2017]	21.7	24.0	25.9	29.2	32.1
TAP [Su and Wen, 2022]	31.2	37.7	40.9	44.5	47.3
Euclidean	18.7	21.3	24.9	27.5	30.0
sDTW [Cuturi and Blondel, 2017]	30.3	37.2	39.7	44.0	46.8
sDTW div. [Blondel et al., 2021]	30.8	38.1	40.0	44.7	47.3
uDTW	32.2	39.0	41.2	45.3	49.0
Unsupervised					
Euclidean	13.5	16.3	20.0	24.9	26.2
sDTW [Cuturi and Blondel, 2017]	20.1	25.3	32.0	36.9	40.9
sDTW div. [Blondel et al., 2021]	20.8	26.0	33.2	37.5	42.3
uDTW	22.7	28.3	35.9	39.4	44.0

7.5 Effectiveness of SigmaNet

In this section, we introduce several variants of how $\boldsymbol{\Sigma}$ is computed to verify the effectiveness of our proposed SigmaNet.

Firstly, we investigate whether SigmaNet is needed in its current form (as in taking features to produce the uncertainty variable), or if $\boldsymbol{\Sigma}$ could be learnt as the so-called free variable. To this end, we create a vector of parameters of size $\tau_{(0)} \cdot \tau_{(0)}$ which we register as one of parameters of the network (we backpropagate w.r.t. this parameter among others). We set $\tau_{(0)}$ to be the average integer of numbers of blocks over sequences. We then reshape this vector into $\tau_{(0)} \times \tau_{(0)}$ matrix and initialize with 0 ± 0.1 uniform noise. We then apply a 2D bilinear interpolation to the matrix to obtain $\boldsymbol{\Sigma}$ of desired size $\tau \times \tau'$, where τ and τ' are the number of temporal blocks for query and support samples, respectively. The $\tau \times \tau'$ matrix is then passed into the sigmoid

Table 7.6: Evaluations on 2D and 3D Kinetics-skeleton.

	Supervised		**Unsupervised**	
	2D	3D	2D	3D
Euclidean	21.2	23.1	12.7	13.3
TAP [Su and Wen, 2022]	32.9	36.0	-	-
sDTW [Cuturi and Blondel, 2017]	34.7	39.6	23.3	28.3
sDTW div. [Blondel et al., 2021]	35.0	40.1	24.0	28.9
uDTW	35.5	42.0	25.9	32.7

function to produce the $\boldsymbol{\Sigma}$ matrix.

For classification of time series, we create a vector of parameters of size $t_{(0)}$ which we register as one of parameters of the network (we backpropagate w.r.t. this parameter among others). We set $\tau_{(0)}$ to be the average integer of numbers of time steps of input time series. We initialize that vector with 0 ± 0.1 uniform noise, and we then use a 1D bilinear interpolation to interpolate the vector into desired length τ. The interpolated vector is passed into the sigmoid function to generate $\sigma_{\mathbf{x}}$ for the input sequence $\mathbf{x}$ of length τ. For sequence $\mathbf{x}'$ (exhaustive search via nearest neighbor) or $\boldsymbol{\mu}_c$ (via nearest centroid), we use exactly the same process to generate $\sigma_{\mathbf{x}'}$ or $\sigma_{\boldsymbol{\mu}_c}$ but of course they have their own vector of length $\tau_{(0)}$ that we minimize over. We obtain $\boldsymbol{\Sigma} = \sigma_{\mathbf{x}}^2 \mathbf{1}^\top + \mathbf{1} \sigma_{\mathbf{x}'}^{\top 2}$ (or $\boldsymbol{\Sigma} = \sigma_{\mathbf{x}}^2 \mathbf{1}^\top + \mathbf{1} \sigma_{\boldsymbol{\mu}_c}^{\top 2}$ if we use the nearest centroid), where squaring is performed in the element-wise manner.

Table 7.7: Comparisons of two different ways of generating $\boldsymbol{\Sigma}$ for few-shot action recognition. Evaluations on the NTU-60 dataset.

#classes	10	20	30	40	50
uDTW ($\boldsymbol{\Sigma}$ via the free variable)	54.1	56.5	61.0	64.1	68.0
uDTW ($\boldsymbol{\Sigma}$ via SigmaNet)	56.9	61.2	64.8	68.3	72.4

Table 7.8: Comparisons of two different ways of generating $\boldsymbol{\Sigma}$ for classification of time series. Evaluations on the UCR archive. K denotes the number of nearest neighbors used by the K nearest neighbors based classification.

	Nearest neighbor			**Nearest centroid**
	$K=1$	$K=3$	$K=5$	
uDTW ($\boldsymbol{\Sigma}$ via the free variable)	77.0	77.3	78.0	70.9
uDTW ($\boldsymbol{\Sigma}$ via SigmaNet)	80.0	81.2	83.3	72.2

In conclusion, the above steps facilitate the direct minimization w.r.t. the variable tied with $\boldsymbol{\Sigma}$ instead of learning $\boldsymbol{\Sigma}$ through our SigmaNet whose input are encoded features *etc*. Tables 7.7 and 7.8 show that using SigmaNet is a much better choice than trying to infer the uncertainty by directly minimizing the free variable. The result is expected as SigmaNet learns to associate feature patterns of sequences with their uncertainty patterns. Minimizing w.r.t. the free variables cannot learn per se.

7.6 Hyperparameters Evaluation

In this section, we evaluate the impact of key hyperparameters. Remaining hyperparameters are obtained through Hyperopt [Bergstra et al., 2015] for hyperparameter search on the validation set.

7.6.1 Evaluation of $\boldsymbol{\Sigma}$

We compare results given different formulations of $\boldsymbol{\Sigma}$ in Tables 7.9 and 7.10. We notice that on smaller datasets, it is hard to determine which variant of $\boldsymbol{\Sigma}$ is better (as these

Table 7.9: Evaluation of different variants of $\boldsymbol{\Sigma}$ computation on small-scale datasets (supervised few-shot action recognition). Operator $\odot$ is the Hadamart product.

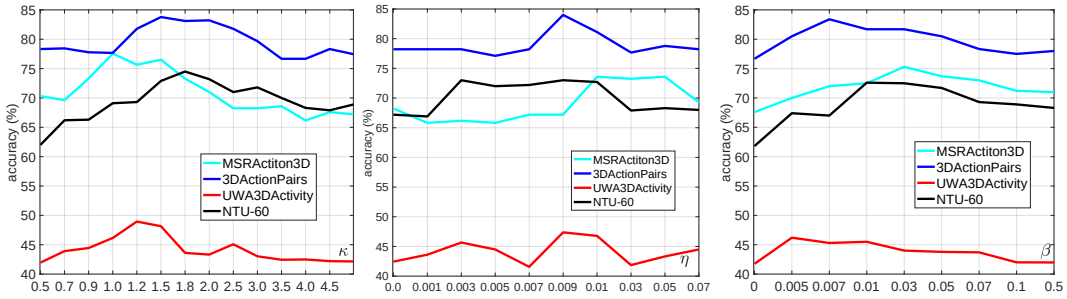
	$\sigma_{\psi}\mathbf{1}^{\top}\odot\mathbf{1}\sigma_{\psi'}^{\top}$	$\sigma_{\psi}^{2}\mathbf{1}^{\top}\odot\mathbf{1}\sigma_{\psi'}^{\top^2}$	$\sigma_{\psi}\mathbf{1}^{\top}+\mathbf{1}\sigma_{\psi'}^{\top}$	$\sigma_{\psi}^{2}\mathbf{1}^{\top}+\mathbf{1}\sigma_{\psi'}^{\top^2}$
MSR Action 3D	**72.32**	68.51	70.59	69.20
3D Action Pairs	82.78	80.56	82.22	**85.00**
UWA 3D Activity	43.86	**45.91**	**45.91**	45.03

Table 7.10: Evaluation of different variants of $\boldsymbol{\Sigma}$ computation on the large-scale NTU-60 dataset (supervised few-shot action recognition).

#classes	$\sigma_{\psi}\mathbf{1}^{\top}\odot\mathbf{1}\sigma_{\psi'}^{\top}$	$\sigma_{\psi}^{2}\mathbf{1}^{\top}\odot\mathbf{1}\sigma_{\psi'}^{\top^2}$	$\sigma_{\psi}\mathbf{1}^{\top}+\mathbf{1}\sigma_{\psi'}^{\top}$	$\sigma_{\psi}^{2}\mathbf{1}^{\top}+\mathbf{1}\sigma_{\psi'}^{\top^2}$
10	56.6	56.0	55.6	**56.9**
20	60.4	61.0	61.2	**61.2**
30	64.2	64.1	63.5	**64.8**
40	68.1	66.9	67.2	**68.3**
50	72.0	72.3	72.0	**72.4**

earlier datasets have fewer limited reliable skeletons compared to the new datasets). However, on bigger datasets, $\boldsymbol{\Sigma}=\sigma_{\psi}^{2}\mathbf{1}^{\top}+\mathbf{1}\sigma_{\psi'}^{\top^2}$ performs the best in all cases; thus we choose this formulation of $\boldsymbol{\Sigma}$ for large-scale datasets.

7.6.2 Evaluation of κ and η of SigmaNet



(a) (b) (c)

Figure 7.7: Evaluation of (a) κ which controls the maximum magnitude and (b) η offset from Eq. (7.25) in SigmaNet and (c) β from Eq. (7.17). Note that $\beta=0$ means no regularization term of uDTW in use. We notice that with the regularization term added to the uDTW, the overall performance is improved.

Figures 7.7a and 7.7b show the impact of κ and η of the scaled sigmoid function in SigmaNet on both small-scale datasets and the large-scle NTU-60 dataset. We notice that $\kappa=1.5$ performs the best on the three small-scale datasets and $\kappa=1.8$ works the best on NTU-60. We choose $\kappa=1.8$ in the experiments for the large-scale datasets. Moreover, $\eta\in[0.003, 0.01]$ works better on NTU-60, and on the small-scale datasets, $\eta=0.01$ achieves the best performance; thus we choose $\eta=0.01$ for the experiments.

Table 7.11: Experimental results on ECGFiveDays (from UCR) and NTU-60 (50-class, supervised / unsup. settings) for different warping window widths.

		$\gamma = 0.001$			$\gamma = 0.01$			$\gamma = 0.1$			$\gamma = 1$		
		r=1.0	r=3.0	r=5.0	r=1.0	r=3.0	r=5.0	r=1.0	r=3.0	r=5.0	r=1.0	r=3.0	r=5.0
ECG FiveDays	sDTW	83.4	82.8	82.0	79.7	76.8	77.8	75.4	69.0	65.3	62.5	61.7	60.2
	uDTW	85.6	91.2	81.0	93.5	82.8	80.6	79.7	73.9	67.3	69.0	65.3	62.5
	uDTW *w/o reg.*	75.4	74.0	69.0	79.7	77.9	76.8	65.3	62.5	61.5	61.2	62.0	60.2
NTU-60 (sup.)	sDTW	65.7	64.7	64.8	65.2	67.8	63.9	60.0	58.9	54.3	54.0	52.2	52.3
	uDTW	71.5	71.0	70.0	72.4	72.4	70.0	68.3	66.7	67.8	65.7	64.8	66.8
	uDTW *w/o reg.*	66.3	65.0	65.5	66.4	68.0	65.2	62.0	59.2	55.0	52.0	52.0	51.2
NTU-60 (unsup.)	sDTW	56.7	53.2	50.0	61.7	61.7	60.0	54.4	52.5	52.1	48.3	45.2	40.9
	uDTW	61.0	61.5	60.7	63.3	63.0	62.5	59.2	59.0	57.3	58.0	57.2	55.7
	uDTW *w/o reg.*	50.1	49.3	47.0	55.3	54.0	51.3	44.1	42.0	40.7	42.3	40.1	35.6

7.6.3 Evaluation of β

Figure 7.7c shows the evaluations of β for both small-scale datasets and NTU-60. Firstly, note that $\beta = 0$ means lack of the regularization term of uDTW, which immediately causes the performance deterioration. As shown in the figure, $\beta = 0.05$ performs the best on UWA 3D Activity, $\beta = 0.03$ achieves the best performance on MSR Action 3D and $\beta = 0.007$ works the best on 3D Action Pairs dataset. We use the corresponding best β values for the smaller datasets. On NTU-60, $\beta \in [0.01, 0.05]$ performs the best compared to other β values, thus we choose $\beta = 0.03$ for the experiments on all large-scale datasets.

7.6.4 Evaluation of warping window width

Table 7.11 on ECGFiveDays (from UCR) and NTU-60 (50-class, supervised / unsup. settings) shows that uDTW does not break quicker than sDTW (window size is parametrized by r). Very small r may preclude backpropagating through some paths (of large distance). For such paths 'beyond window', learning uncertainty is limited but this is normal. For similar reasons, choosing the right window size is required by other DTW variants too. Also, if r is very large, large uncertainty score may decrease the distance on multitude of paths by downweighting parts of paths (could lead to strange matching) but as the uncertainty is aggregated into the regularization penalty, this penalty prevents uDTW from unreasonable solutions. Lack of regularization penalty (*w/o reg.*) affects the most the unsupervised few-shot learning, while supervised loss can still drive SigmaNet to produce meaningful results.

7.7 Network Configuration and Training Details

Below we provide the details of network configuration and training process.

7.7.1 Skeleton Data Preprocessing

Before passing the skeleton sequences into MLP and a simple linear graph network (*e.g.*, S^2GC), we first normalize each body joint w.r.t. to the torso joint $\mathbf{v}_{f,c}$:

$$\mathbf{v}'_{f,i} = \mathbf{v}_{f,i} - \mathbf{v}_{f,c}, \tag{7.23}$$

where f and i are the index of video frame and human body joint respectively. After that, we further normalize each joint coordinate into [-1, 1] range:

$$\hat{\mathbf{v}}_{f,i}[j] = \frac{\mathbf{v}'_{f,i}[j]}{\max([\text{abs}(\mathbf{v}'_{f,i}[j])]_{f\in\mathcal{I}_\tau, i\in\mathcal{I}_J})}, \tag{7.24}$$

where j is for selection of the x, y and z axes, τ is the number of frames and J is the number of 3D body joints per frame.

For the skeleton sequences that have more than one performing subject, (i) we normalize each skeleton separately, and each skeleton is passed to MLP for learning the temporal dynamics, and (ii) for the output features per skeleton from MLP, we pass them separately to the graph neural network, *e.g.*, two skeletons from a given video sequence will have two outputs obtained from the graph neural network, and we aggregate the outputs through average pooling before passing to sDTW or uDTW.

7.7.2 Network Configuration

SigmaNet. It is composed of an FC layer and a scaled sigmoid function which translate the learned features of either actions or time series into desired $\boldsymbol{\Sigma}$. The input to FC is of the size of feature dimension (depends on the encoder) and the output is a scalar. SigmaNet with the scaled sigmoid function can be defined as:

$$\sigma(\boldsymbol{\psi}) = \frac{\kappa}{1+\exp(-\text{FC}(\boldsymbol{\psi}))} + \eta, \tag{7.25}$$

where $\eta > 0$ is the offset and $\kappa \geq 0$ is the maximum magnitude of sigmoid. For an entire sequence with τ blocks, the SigmaNet produces vector $\boldsymbol{\sigma}_{\mathbf{x}}$ for sequence $\mathbf{x}$ and $\boldsymbol{\sigma}_{\mathbf{x}'}$ for sequence $\mathbf{x}'$ (we concatenate per-block scalars to form these vectors), and we typically obtain $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_{\mathbf{x}}^2 \mathbf{1}^\top + \mathbf{1}\boldsymbol{\sigma}_{\mathbf{x}'}^{2\top}$.

Forecasting of the evolution of time series. The MLP for this task consists of two FC layers with a tanh layer in between. The input to the first FC layer is t and output size is t', and after the tanh layer, the input to the second FC layer is t' and output $(\tau - t)$ dimensional prediction. We set $t' = 30$ or 50 depending on the length of time series in each dataset.

Few-shot action recognition. Given the temporal block size M (the number of frames in a block) and desired output size d, the configuration of the 3-layer MLP unit is: FC ($3M \rightarrow 6M$), LayerNorm (LN) as in [Dosovitskiy et al., 2020], ReLU, FC ($6M \rightarrow 9M$), LN, ReLU, Dropout (for smaller datasets, the dropout rate is 0.5; for large-scale datasets, the dropout rate is 0.1), FC ($9M \rightarrow d$), LN. Note that M is the temporal

block size and d is the output feature dimension per body joint. We set $M=10$ for experiments.

For the encoding network, let us take the query input $\mathbf{X}\in\mathbb{R}^{3\times J\times M}$ for the temporal block of length M as an example, where 3 indicates that Cartesian coordinates (x,y,z) were used, and J is the number of body joints. As alluded to earlier, we obtain $\widehat{\mathbf{X}}^T=\text{MLP}(\mathbf{X};\mathcal{P}_{MLP})\in\mathbb{R}^{d\times J}$.

Subsequently, we employ a simple linear graph network, S^{2}GC from Section 7.7.3, and the transformer encoder [Dosovitskiy et al., 2020] which consists of alternating layers of Multi-Head Self-Attention (MHSA) and a feed-forward MLP (two FC layers with a GELU non-linearity between them). LayerNorm (LN) is applied before every block, and residual connections after every block. Each block feature matrix $\widehat{\mathbf{X}}\in\mathbb{R}^{J\times d}$ encoded by a simple linear graph network S^{2}GC (without learnable $\boldsymbol{\Theta}$) is then passed to the transformer. Similarly to the standard transformer, we prepend a learnable vector $\mathbf{y}_{\text{token}}\in\mathbb{R}^{1\times d}$ to the sequence of block features $\widehat{\mathbf{X}}$ obtained from S^{2}GC, and we also add the positional embeddings $\mathbf{E}_{\text{pos}}\in\mathbb{R}^{(1+J)\times d}$ based on the sine and cosine functions (standard in transformers) so that token $\mathbf{y}_{\text{token}}$ and each body joint enjoy their own unique positional encoding. We obtain $\mathbf{Z}_0\in\mathbb{R}^{(1+J)\times d}$ which is the input in the following backbone:

$$\mathbf{Z}_0 = [\mathbf{y}_{\text{token}};\text{S}^2\text{GC}(\widehat{\mathbf{X}})]+\mathbf{E}_{\text{pos}}, \tag{7.26}$$

$$\mathbf{Z}'_k = \text{MHSA}(\text{LN}(\mathbf{Z}_{k-1}))+\mathbf{Z}_{k-1},\ k=1,...,L_{\text{tr}} \tag{7.27}$$

$$\mathbf{Z}_k = \text{MLP}(\text{LN}(\mathbf{Z}'_k))+\mathbf{Z}'_k,\qquad k=1,...,L_{\text{tr}} \tag{7.28}$$

$$\mathbf{y}' = \text{LN}(\mathbf{Z}^{(0)}_{L_{\text{tr}}})\qquad \text{where}\qquad \mathbf{y}'\in\mathbb{R}^{1\times d} \tag{7.29}$$

$$f(\mathbf{X};\mathcal{P}) = \text{FC}(\mathbf{y}'^T;\mathcal{P}_{FC})\qquad\qquad \in\mathbb{R}^{d'}, \tag{7.30}$$

where $\mathbf{Z}^{(0)}_{L_{\text{tr}}}$ is the first d dimensional row vector extracted from the output matrix $\mathbf{Z}_{L_{\text{tr}}}$ of size $(J+1)\times d$ which corresponds to the last layer L_{tr} of the transformer. Moreover, parameter L_{tr} controls the depth of the transformer, whereas $\mathcal{P}\equiv[\mathcal{P}_{MLP},\mathcal{P}_{S^2GC},\mathcal{P}_{Transf},\mathcal{P}_{FC}]$ is the set of parameters of EN. In case of S^{2}GC, $|\mathcal{P}_{S^2GC}|=0$ because we do not use their learnable parameters $\boldsymbol{\Theta}$ (*i.e.*, think $\boldsymbol{\Theta}$ is set as the identity matrix in Eq. (7.31)).

We can define now a support feature map as $\boldsymbol{\Psi}'=[f(\boldsymbol{X}_1;\mathcal{P}),...,f(\boldsymbol{X}_{\tau'};\mathcal{P})]\in\mathbb{R}^{d'\times\tau'}$ for τ' temporal blocks, and the query map $\boldsymbol{\Psi}$ accordingly.

The hidden size of our transformer (the output size of the first FC layer of the MLP depends on the dataset. For smaller datasets, the depth of the transformer is $L_{\text{tr}}=6$ with 64 as the hidden size, and the MLP output size is $d=32$ (note that the MLP which provides $\widehat{\mathbf{X}}$ and the MLP in the transformer must both have the same output size). For NTU-60, the depth of the transformer is $L_{\text{tr}}=6$, the hidden size is 128 and the MLP output size is $d=64$. For NTU-120, the depth of the transformer is $L_{\text{tr}}=6$, the hidden size is 256 and the MLP size is $d=128$. For Kinetics-skeleton, the depth for the transformer is $L_{\text{tr}}=12$, hidden size is 512 and the MLP output size is $d=256$. The number of Heads for the transformer of smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton is set as 6, 12, 12 and 12, respectively.

The output sizes d' of the final FC layer are 50, 100, 200, and 500 for the smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton, respectively.

7.7.3 Linear Graph Network (S^{2}GC)

Based on a modified Markov Diffusion Kernel, Simple Spectral Graph Convolution (S^{2}GC) is the summation over l-hops, $l=1,...,L$. The output of S^{2}GC is given as:

$$\boldsymbol{\Phi}_{\text{S}^2\text{GC}} = \frac{1}{L}\sum_{l=1}^{L}((1-\alpha)\mathbf{S}^l\mathbf{X}+\alpha\mathbf{X})\boldsymbol{\Theta}, \tag{7.31}$$

where $L \geq 1$ is the number of linear layers and $\alpha \geq 0$ determines the importance of self-loop of each node (we use their default setting $\alpha = 0.05$ and $L = 6$). Choice of other graph embeddings are possible, including contrastive models COLES [Zhu et al., 2021a] or COSTA [Zhang et al., 2022f], adversarial Fisher-Bures GCN [Sun et al., 2019] or GCNs with rectifier attention [Zhang et al., 2022e]. One may also use kernels on 3D body joints as in [Tas and Koniusz, 2018] or even use CNN to encode 3D body joints as COLTRANE [Prabowo et al., 2019].

7.7.4 K-NN classifier with SoftMax

For the K-NN classifier, instead of using K best weights proportional to the inverse of the distance from the query sample $\mathbf{x}^*$ to the closest samples $\mathbf{x}_n$ (as is done in the soft-DTW paper [Cuturi and Blondel, 2017]) and expressed by

$$w(\mathbf{x}_n|\mathbf{x}^*) = \frac{1}{d^2(\mathbf{x}^*, \mathbf{x}_n)}, \tag{7.32}$$

we weigh the neighbors $\mathbf{x}_n$ of $\mathbf{x}^*$ using

$$w(\mathbf{x}_n|\mathbf{x}^*) = \frac{\exp\big(-\frac{1}{\gamma''}d^2(\mathbf{x}^*, \mathbf{x}_n)\big)}{\sum_{n'\in\mathcal{N}(\mathbf{x}^*;K)}\exp\big(-\frac{1}{\gamma''}d^2(\mathbf{x}^*, \mathbf{x}_{n'})\big)} \tag{7.33}$$

such that $\mathcal{N}(\mathbf{x}^*;K)$ produces K nearest samples $\mathbf{x}_{n'}$ of $\mathbf{x}^*$ according to distance $d(\cdot,\cdot)$, *e.g.*, the Euclidean distance, sDTW or uDTW. Parameter $\gamma'' > 0$ (in our case, we set $\gamma'' = 6$) further controls the impact of each sample $\mathbf{x}_n$ on the classifier based on the bell shape of Radial Basis Function in the above equation.

Table 7.12 shows the comparisons. We notice that the use of SoftMax in the K-NN classifier improves the performance for all the methods when $K=3$ and $K=5$.

7.7.5 Training Details

For both time series and few-shot action recognition pipelines, the weights are initialized with the normal distribution (zero mean and unit standard deviation). We use 1e-3 for the learning rate, and the weight decay is 1e-6. We use the SGD optimizer.

Table 7.12: Classification accuracy (mean±std) on UCR archive using nearest neighbor. K denotes the number of nearest neighbors in the K-NN classifier. Highlighted rows are the based on SoftMax from Eq. (7.33). Non-highlighted rows are based on Eq. (7.32).

	Nearest neighbor		
	$K=1$	$K=3$	$K=5$
Euclidean	71.2±17.5	69.5±18.0	67.5±17.6
Euclidean (SoftMax)	71.2±17.5	72.3±18.1	73.0±16.7
DTW [Cuturi, 2011]	74.2±16.6	72.8±16.9	71.4±16.8
DTW [Cuturi, 2011] (SoftMax)	74.2±16.6	75.0±17.0	75.4±15.8
sDTW [Cuturi and Blondel, 2017]	76.2±16.6	74.0±15.6	70.5±17.6
sDTW [Cuturi and Blondel, 2017] (SoftMax)	76.2±16.6	77.2±15.9	78.0±16.5
sDTW div. [Blondel et al., 2021]	78.6±16.2	76.5±16.4	74.8±15.8
sDTW div. [Blondel et al., 2021] (SoftMax)	78.6±16.2	79.5±16.7	80.1±16.5
uDTW	80.0±15.0	78.0±15.8	76.2±15.0
uDTW (SoftMax)	80.0±15.0	81.2±17.8	83.3±16.2

For time series, we set the training epochs to 30, 50 and 100 depending on the dataset in the UCR archive (due to many datasets, the epoch settings will be provided in the code directly).

For few-shot action recognition, we set the number of training episodes to 100K for NTU-60, 200K for NTU-120, 500K for 3D Kinetics-skeleton, 10K for small datasets such as UWA 3D Multiview Activity II.

7.8 Additional Evaluations for Few-shot Action Recognition

Table 7.13: uDTW derived under the Normal, Laplacian and Cauchy distributions. Evaluations of few-shot action recognition on small-scale datasets.

	Supervised			**Unsupervised**		
	MSR	3DActionPairs	UWA3D	MSR	3DActionPairs	UWA3D
TAP (HM) [Su and Wen, 2022]	67.40	77.22	37.13	-	-	-
TAP (Lifted) [Su and Wen, 2022]	65.20	78.33	34.80	-	-	-
TAP (Bino.) [Su and Wen, 2022]	66.67	78.33	36.55	-	-	-
sDTW [Cuturi and Blondel, 2017]	70.59	81.67	44.74	62.63	48.33	39.47
uDTW (Laplace)	72.24	82.89	45.64	**66.00**	**55.00**	41.22
uDTW (Cauchy)	70.88	**84.44**	45.03	65.12	50.32	40.50
uDTW (Normal)	**72.66**	83.33	**47.66**	65.00	52.22	**41.74**

We also evaluate our proposed uDTW versus sDTW on smaller datasets for both supervised and unsupervised settings. As uDTW was derived in Section 7.1.2 under modeling the MLE of the product of the Normal distributions, we investigate modeling each path $\boldsymbol{\Pi}_i$ by replacing the Normal distribution with the Laplace or Cauchy distributions. By applying MLE principles in analogy to Section 7.1.2, we arrive at $\beta\Omega_{\boldsymbol{\Pi}_i} + d^2_{\boldsymbol{\Pi}_i}$ for

i. Laplace: $\sum_{(m,n)\in\boldsymbol{\Pi}_i} \beta \log(\sigma_{mn}) + \frac{\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|_1}{\sigma_{mn}}$;

Table 7.14: uDTW derived under the Normal, Laplacian and Cauchy distributions. Evaluations of few-shot action recognition on the large-scale NTU-60 dataset.

#classes	10	20	30	40	50
			Supervised		
sDTW(baseline) [Cuturi and Blondel, 2017]	53.7	56.2	60.0	63.9	67.8
uDTW(Cauchy)	56.1	61.1	62.9	68.3	69.9
uDTW(Laplace)	55.3	59.2	63.3	67.7	70.3
uDTW(Normal)	56.9	61.2	64.8	68.3	72.4
			Unsupervised		
sDTW(baseline) [Cuturi and Blondel, 2017]	35.6	45.2	53.3	56.7	61.7
uDTW(Cauchy)	36.7	47.9	54.9	57.3	63.3
uDTW(Laplace)	36.2	48.2	54.3	57.8	63.1
uDTW(Normal)	37.0	48.3	55.3	58.0	63.3

ii. Cauchy: $\sum_{(m,n)\in\boldsymbol{\Pi}_i} \beta \log(\sigma_{mn}) + \log\left(1 + \frac{\|\boldsymbol{\psi}_m - \boldsymbol{\psi}'_n\|_2^2}{\sigma_{mn}^2}\right)$.

Table 7.13 shows that uDTW achieves better performance than sDTW, and the Laplace distribution is performing particularly well on the unsupervised few-shot action recognition. Table 7.14 shows that uDTW based on the Normal distribution is overall better than other distributions on large-scale datasets such as NTU-60. For this very reason we use uDTW based on the Normal distribution.

7.9 Conclusions

We have introduced the uncertainty-DTW which handles the uncertainty estimation of frame- and/or block-wise features to improve the path warping of the celebrated soft-DTW. Our uDTW produces the uncertainty-weighted distance along the path and returns the regularization penalty aggregated along the path, which follows sound principles of classifier regularization. We have provided several pipelines for time series forecasting, and supervised and unsupervised action recognition, which use uDTW as a distance. Our simple uDTW achieves better sequence alignment in several benchmarks.

Chapter 8

Temporal-Viewpoint Transportation Plan

We propose a Few-shot Learning pipeline for 3D skeleton-based action recognition by Joint tEmporal and cAmera viewpoiNt alIgnmEnt (JEANIE). To factor out misalignment between query and support sequences of 3D body joints, we propose an advanced variant of Dynamic Time Warping which jointly models each smooth path between the query and support frames to achieve simultaneously the best alignment in the temporal and simulated camera viewpoint spaces for end-to-end learning under the limited few-shot training data. Sequences are encoded with a temporal block encoder based on Simple Spectral Graph Convolution, a lightweight linear Graph Neural Network backbone. We also include a setting with a transformer. Finally, we propose a similarity-based loss which encourages the alignment of sequences of the same class while preventing the alignment of unrelated sequences. We show state-of-the-art results on NTU-60, NTU-120, Kinetics-skeleton and UWA3D Multiview Activity II.

8.1 Introduction

Action recognition is arguably among key topics in computer vision due to applications in video surveillance [Wang, 2017; Wang et al., 2019c], human-computer interaction, sports analysis, virtual reality and robotics. Many pipelines [Tran et al., 2015; Feichtenhofer et al., 2016b, 2017a; Carreira and Zisserman, 2017; Wang et al., 2019b; Koniusz et al., 2020] perform action classification given the large amount of labeled training data. However, manually collecting and labeling videos for 3D skeleton sequences is laborious, and such pipelines need to be retrained or fine-tuned for new class concepts. Popular action recognition networks include two-stream neural networks [Feichtenhofer et al., 2016b, 2017a; Wang et al., 2017] and 3D convolutional networks (3D CNNs) [Tran et al., 2015; Carreira and Zisserman, 2017], which aggregate frame-wise and temporal block representations, respectively. However, such networks indeed must be trained on large-scale datasets such as Kinetics [Carreira and Zisserman, 2017; Wang et al., 2019d; Wang and Koniusz, 2021; Koniusz et al., 2021] under a fixed set of training class concepts.

Thus, there exists a growing interest in devising effective Few-shot Learning (FSL) for action recognition, termed Few-shot Action Recognition (FSAR), that rapidly adapts to novel classes given a few training samples [Mishra et al., 2018; Xu et al., 2018; Guo et al., 2018; Dwivedi et al., 2019; Zhang et al., 2020a; Cao et al., 2020; Wang and Koniusz, 2022b]. However, FSAR for videos is scarce due to the volumetric nature of videos and large intra-class variations.

FSL for image recognition has been widely studied [Miller et al., 2000; Li et al., 2002; Fink, 2005; Bart and Ullman, 2005; Fei-Fei et al., 2006; Lake et al., 2011] including contemporary CNN-based FSL methods [Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Sung et al., 2018; Zhang and Koniusz, 2019], which use meta-learning, prototype-based learning and feature representation learning. Just in 2020–2022, many FSL methods [Guo et al., 2020; Dvornik et al., 2020; Wang et al., 2020; Lichtenstein et al., 2020; Luo et al., 2021; Fei et al., 2020; Guan et al., 2020; Li et al., 2020b; Elsken et al., 2020; Cao et al., 2020; Tang et al., 2020; Koniusz and Zhang, 2020; Simon et al., 2020b; Zhang et al., 2022a; Zhu and Koniusz, 2022; Lu and Koniusz, 2022] have been dedicated to image classification or detection [Yu et al., 2020; Zhang et al., 2021a, 2020d, 2022d,c]. Noteworthy mentioning is the incremental learning paradigm that can also tackle novel classes [Simon et al., 2021]. In this chapter, we aim at advancing few-shot recognition of articulated set of connected 3D body joints.

With an exception of very recent models [Liu et al., 2017; Liu et al., 2019a; Memmesheimer et al., 2020, 2021; Wang and Koniusz, 2022b; Qin et al., 2022], FSAR approaches that learn from skeleton-based 3D body joints are scarce. The above situation prevails despite action recognition from articulated sets of connected body joints, expressed as 3D coordinates, does offer a number of advantages over videos such as (i) the lack of the background clutter, (ii) the volume of data being several orders of magnitude smaller, and (iii) the 3D geometric manipulations of sequences being relatively friendly.

Thus, we propose a FSAR approach that learns on skeleton-based 3D body joints via Joint tEmporal and cAmera viewpoiNt alIgnmEnt (JEANIE). As FSL is based on learning similarity between support-query pairs, to achieve good matching of queries with support sequences representing the same action class, we propose to simultaneously model the optimal (i) temporal and (ii) viewpoint alignments. To this end, we build on soft-DTW [Cuturi and Blondel, 2017], a differentiable variant of Dynamic Time Warping (DTW) [Cuturi, 2011]. Unlike soft-DTW, we exploit the projective camera geometry. We assume that the best smooth path in DTW should simultaneously provide the best temporal and viewpoint alignment, as sequences that are being matched might have been captured under different camera viewpoints or subjects might have followed different trajectories.

To obtain skeletons under several viewpoints, we rotate skeletons (zero-centered by hip) by Euler angles [eul] w.r.t. x, y and z axes, or generate skeleton locations given simulated camera positions, according to the algebra of stereo projections [ste].

We note that view-adaptive models for action recognition do exist. View Adaptive Recurrent Neural Networks [Zhang et al., 2017b, 2019d] is a classification model equipped with a view-adaptive subnetwork that contains the rotation and translation

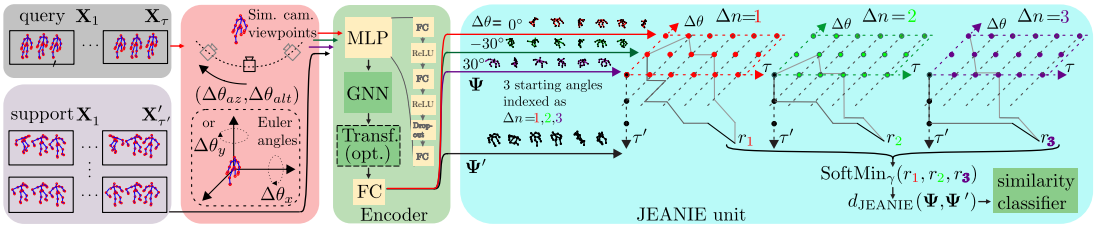


Figure 8.1: Our 3D skeleton-based FSAR with JEANIE. Frames from a query sequence and a support sequence are split into short-term temporal blocks $\mathbf{X}_1, ..., \mathbf{X}_\tau$ and $\mathbf{X}'_1, ..., \mathbf{X}'_{\tau'}$ of length M given stride S. Subsequently, we generate (i) multiple rotations by $(\Delta\theta_x, \Delta\theta_y)$ of each query skeleton by either Euler angles (baseline approach) or (ii) simulated camera views (gray cameras) by camera shifts $(\Delta\theta_{az}, \Delta\theta_{alt})$ w.r.t. the assumed average camera location (black camera). We pass all skeletons via Encoding Network (with an optional transformer) to obtain feature tensors $\mathbf{\Psi}$ and $\mathbf{\Psi}'$, which are directed to JEANIE. We note that the temporal-viewpoint alignment takes place in 4D space (we show a 3D case with three views: $-30°, 0°, 30°$). Temporally-wise, JEANIE starts from the same $t=(1,1)$ and finishes at $t=(\tau, \tau')$ (as in DTW). Viewpoint-wise, JEANIE starts from every possible camera shift $\Delta\theta \in \{-30°, 0°, 30°\}$ (we do not know the true correct pose) and finishes at one of possible camera shifts. At each step, the path may move by no more than $(\pm\Delta\theta_{az}, \pm\Delta\theta_{alt})$ to prevent erroneous alignments. Finally, SoftMin picks up the smallest distance.

switches within its RNN backbone, and the main LSTM-based network. Temporal Segment Network [Wang et al., 2019] models long-range temporal structures with a new segment-based sampling and aggregation module. However, such pipelines require a large number of training samples with varying viewpoints and temporal shifts to learn a robust model. Their limitations become evident when a network trained under a fixed set of action classes has to be adapted to samples of novel classes. Our JEANIE does not suffer from such a limitation.

Our pipeline consists of an MLP which takes neighboring frames to form a temporal block. Firstly, we sample desired Euler rotations or simulated camera viewpoints, generate multiple skeleton views, and pass them to the MLP to get block-wise feature maps, next forwarded to a Graph Neural Network (GNN), *e.g.*, GCN [Kipf and Welling, 2017], Fisher-Bures GCN [Sun et al., 2019], SGC [Wu et al., 2019b], APPNP [Klicpera et al., 2019] or S^{2}GC [Zhu and Koniusz, 2021b; Zhu et al., 2021a], followed by an optional transformer [Dosovitskiy et al., 2020], and an FC layer to obtain graph-based representations passed to JEANIE.

JEANIE builds on Reproducing Kernel Hilbert Spaces (RKHS) [Smola and Kondor, 2003] which scale gracefully to FSAR problems which, by their setting, learn to match pairs of sequences rather than predict class labels. JEANIE builds on Optimal Transport [Villani, 2009] by using a transportation plan for temporal and viewpoint alignment in skeletal action recognition.

Below are our contributions:

i. We propose a Few-shot Action Recognition approach for learning on skeleton-

based articulated 3D body joints via JEANIE, which performs the joint alignment of temporal blocks and simulated viewpoint indexes of skeletons between support-query sequences to select the smoothest path without abrupt jumps in matching temporal locations and view indexes. Warping jointly temporal locations and simulated viewpoint indexes helps meta-learning with limited samples of novel classes.

ii. To simulate different viewpoints of 3D skeleton sequences, we consider rotating them (1) by Euler angles within a specified range along x and y axes, or (2) towards the simulated camera locations based on the algebra of stereo projection.

iii. We investigate several different GNN backbones (including transformer), as well as the optimal temporal size and stride for temporal blocks encoded by a simple 3-layer MLP unit before forwarding them to GNN.

iv. We propose a simple similarity-based loss encouraging the alignment of within-class sequences and preventing the alignment of between-class sequences.

We achieve the state of the art on large-scale NTU-60 [Shahroudy et al., 2016a], NTU-120 [Liu et al., 2019a], Kinetics-skeleton [Yan et al., 2018] and UWA3D Multiview Activity II [Rahmani et al., 2016b]. As far as we can tell, the simultaneous alignment in the joint temporal-viewpoint space for FSAR is a novel proposition.

8.2 Related Works

Below, we describe 3D skeleton-based action recognition, FSAR approaches and GNNs.

Action recognition (3D skeletons). 3D skeleton-based action recognition pipelines often use GCNs [Kipf and Welling, 2017], *e.g.*, spatio-temporal GCN [Yan et al., 2018], an a-links inference model [Li et al., 2019], shift-graph model [Cheng et al., 2020b] and multi-scale aggregation node [Liu et al., 2020b]. However, such models rely on large-scale datasets, and cannot be easily adapted to novel class concepts.

FSAR (videos). Approaches [Mishra et al., 2018; Guo et al., 2018; Xu et al., 2018] use a generative model, graph matching on 3D coordinates and dilated networks, respectively. Approach [Zhu and Yang, 2018] uses a compound memory network. ProtoGAN [Dwivedi et al., 2019] generates action prototypes. Model [Zhang et al., 2020a] uses permutation-invariant attention and second-order aggregation of temporal video blocks, whereas approach [Cao et al., 2020] proposes a modified temporal alignment for query-support pairs via DTW.

FSAR (3D skeletons). Few FSAR models use 3D skeletons [Liu et al., 2017; Liu et al., 2019a; Memmesheimer et al., 2020, 2021]. Global Context-Aware Attention LSTM [Liu et al., 2017] selectively focuses on informative joints. Action-Part Semantic Relevance-aware (APSR) model [Liu et al., 2019a] uses the semantic relevance between each body part and action class at the distributed word embedding level. Signal Level Deep Metric Learning (DML) [Memmesheimer et al., 2020] and Skeleton-DML

[Memmesheimer et al., 2021] one-shot FSL approaches encode signals into images, extract features using CNN and apply multi-similarity miner losses. In contrast, we use temporal blocks of 3D body joints of skeletons encoded by GNNs under multiple viewpoints of skeletons to simultaneously perform temporal and viewpoint-wise alignment of query-support in the meta-learning regime.

Graph Neural Networks. GNNs are popular in the skeleton-based action recognition. We build on GNNs in this chapter due to their excellent ability to represent graph-structured data such as interconnected body joints. GCN [Kipf and Welling, 2017] applies graph convolution in the spectral domain, and enjoys the depth-efficiency when stacking multiple layers due to non-linearities. However, depth-efficiency costs speed due to backpropagation through consecutive layers. In contrast, a very recent family of so-called spectral filters do not require depth-efficiency but apply filters based on heat diffusion to the graph Laplacian. As a result, they are fast linear models as learnable weights act on filtered node representations. SGC [Wu et al., 2019b], APPNP [Klicpera et al., 2019] and S^2GC [Zhu and Koniusz, 2021b] are three methods from this family which we investigate for the backbone.

Multi-view action recognition. Multi-modal sensors enable multi-view action recognition [Wang et al., 2019b; Zhang et al., 2017b]. A Generative Multi-View Action Recognition framework [Wang et al., 2019d] integrates complementary information from RGB and depth sensors by View Correlation Discovery Network. Some works exploit multiple views of the subject [Shahroudy et al., 2016a; Liu et al., 2019a; Zhang et al., 2019d; Wang et al., 2019d] to overcome the viewpoint variations for action recognition on large training datasets. In contrast, our JEANIE learns to perform jointly the temporal and simulated viewpoint alignment in an end-to-end meta-learning setting. This is a novel paradigm based on similarity learning of support-query pairs rather than learning class concepts.

8.3 Background

Below we present a necessary background on Euler angles and the algebra of stereo projection, GNNs and the formulation of soft-DTW.

Euler angles [eul] are defined as successive planar rotation angles around x, y, and z axes. For 3D coordinates, we have the following rotation matrices $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}, \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.1}$$

As the resulting composite rotation matrix depends on the order of rotation axes, *i.e.*, $\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \neq \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x$, we also investigate the algebra of stereo projection.

Stereo projections [ste]. Suppose we have a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t} = [t_x, t_y, t_z]^T$ between left/right cameras (imagine some non-existent stereo camera). Let $\mathbf{M}_l$ and $\mathbf{M}_r$ be the intrinsic matrices of the left/right cameras. Let $\mathbf{p}_l$ and $\mathbf{p}_r$ be coordinates of the left/right camera. As the origin of the right camera

in the left camera coordinates is $\mathbf{t}$, we have: $\mathbf{p}_r = \mathbf{R}(\mathbf{p}_l - \mathbf{t})$ and $(\mathbf{p}_l - \mathbf{t})^T = (\mathbf{R}^T\mathbf{p}_r)^T$. The plane (polar surface) formed by all points passing through $\mathbf{t}$ can be expressed by $(\mathbf{p}_l - \mathbf{t})^T(\mathbf{p}_l \times \mathbf{t}) = 0$. Then, $\mathbf{p}_l \times \mathbf{t} = \mathbf{S}\mathbf{p}_l$ where $\mathbf{S} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$. Based on the above equations, we obtain ${\mathbf{p}_r}^T\mathbf{R}\mathbf{S}\mathbf{p}_l = 0$, and note that $\mathbf{R}\mathbf{S} = \mathbf{E}$ is the Essential Matrix, and $\mathbf{p}_r^T\mathbf{E}\mathbf{p}_l = 0$ describes the relationship for the same physical point under the left and right camera coordinate system. As $\mathbf{E}$ has no internal inf. about the camera, and $\mathbf{E}$ is based on the camera coordinates, we use a fundamental matrix $\mathbf{F}$ that describes the relationship for the same physical point under the camera pixel coordinate system. The relationship between the pixel and camera coordinates is: $\mathbf{p}^* = \mathbf{M}\mathbf{p}'$ and ${\mathbf{p}'_r}^T\mathbf{E}\mathbf{p}'_l = 0$.

Now, suppose the pixel coordinates of $\mathbf{p}'_l$ and $\mathbf{p}'_r$ in the pixel coordinate system are $\mathbf{p}^*_l$ and $\mathbf{p}^*_r$, then we can write ${\mathbf{p}^*_r}^T(\mathbf{M}_r^{-1})^T\mathbf{E}\mathbf{M}_l^{-1}\mathbf{p}^*_l = 0$, where $\mathbf{F} = (\mathbf{M}_r^{-1})^T\mathbf{E}\mathbf{M}_l^{-1}$ is the fundamental matrix. Thus, the relationship for the same point in the pixel coordinate system of the left/right camera is:

$$ {\mathbf{p}^*_r}^T\mathbf{F}\mathbf{p}^*_l = 0. \tag{8.2} $$

We treat 3D body joint coordinates as $\mathbf{p}^*_l$. Given $\mathbf{F}$, we obtain their coordinates $\mathbf{p}^*_r$ in the new view.

GNN notations. Firstly, let $G = (\mathbf{V}, \mathbf{E})$ be a graph with the vertex set $\mathbf{V}$ with nodes $\{v_1, ..., v_n\}$, and $\mathbf{E}$ are edges of the graph. Let $\mathbf{A}$ and $\mathbf{D}$ be the adjacency and diagonal degree matrix, respectively. Let $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ be the adjacency matrix with self-loops (identity matrix) with the corresponding diagonal degree matrix $\tilde{\mathbf{D}}$ such that $\tilde{D}_{ii} = \sum_j(\mathbf{A}^{ij} + \mathbf{I}^{ij})$. Let $\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ be the normalized adjacency matrix with added self-loops. For the l-th layer, we use $\boldsymbol{\Theta}^{(l)}$ to denote the learnt weight matrix, and $\boldsymbol{\Phi}$ to denote the outputs from the graph networks. Below, we list backbones used by us.

GCN [Kipf and Welling, 2017]. GCNs learn the feature representations for the features $\mathbf{x}_i$ of each node over multiple layers. For the l-th layer, we denote the input by $\mathbf{H}^{(l-1)}$ and the output by $\mathbf{H}^{(l)}$. Let the input (initial) node representations be $\mathbf{H}^{(0)} = \mathbf{X}$. For an L-layer GCN, the output representations are given by:

$$ \boldsymbol{\Phi}_{\text{GCN}} = \mathbf{S}\mathbf{H}^{(L-1)}\boldsymbol{\Theta}^{(L)} \text{ where } \mathbf{H}^{(l)} = \text{ReLU}(\mathbf{S}\mathbf{H}^{(l-1)}\boldsymbol{\Theta}^{(l)}). \tag{8.3} $$

APPNP [Klicpera et al., 2019]. The Personalized Propagation of Neural Predictions (PPNP) and its fast approximation, APPNP, are based on the personalized PageRank. Let $\mathbf{H}^{(0)} = f_{\boldsymbol{\Theta}}(X)$ be the input to APPNP, where $f_{\boldsymbol{\Theta}}$ can be an MLP with parameters $\boldsymbol{\Theta}$. Let the output of the l-th layer be $\mathbf{H}^{(l)} = (1-\alpha)\mathbf{S}\mathbf{H}^{(l-1)} + \alpha\mathbf{H}^{(0)}$, where α is the teleport (or restart) probability in range $(0, 1]$. For an L-layer APPNP, we have:

$$ \boldsymbol{\Phi}_{\text{APPNP}} = (1-\alpha)\mathbf{S}\mathbf{H}^{L} + \alpha\mathbf{H}^{(\mathbf{0})}. \tag{8.4} $$

SGC [Wu et al., 2019b] & **S^{2}GC** [Zhu and Koniusz, 2021b]. SGC captures the L-hops neighborhood in the graph by the L-th power of the transition matrix used as a

spectral filter. For an L-layer SGC, we obtain:

$$\boldsymbol{\Phi}_{\text{SGC}} = \mathbf{S}^L \mathbf{X} \boldsymbol{\Theta}. \tag{8.5}$$

Based on a modified Markov Diffusion Kernel, Simple Spectral Graph Convolution (S^2GC) is the summation over l-hops, $l = 1, ..., L$. The output of S^2GC is:

$$\boldsymbol{\Phi}_{\text{S}^2\text{GC}} = \frac{1}{L} \sum_{l=1}^{L} ((1-\alpha)\mathbf{S}^l \mathbf{X} + \alpha \mathbf{X})\boldsymbol{\Theta}. \tag{8.6}$$

Soft-DTW [Cuturi, 2011; Cuturi and Blondel, 2017]. Dynamic Time Warping can be seen as a specialized case of the Wasserstein metric, under specific transportation plan. Soft-DTW is defined as:

$$d_{\text{DTW}}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') = \underset{\mathbf{A} \in \mathcal{A}_{\tau,\tau'}}{\text{SoftMin}_\gamma} \left\langle \mathbf{A}, \boldsymbol{D}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \right\rangle, \tag{8.7}$$

$$\text{where } \text{SoftMin}_\gamma(\boldsymbol{\alpha}) = -\gamma \log \sum_i \exp(-\alpha_i / \gamma). \tag{8.8}$$

The binary $\mathbf{A} \in \mathcal{A}_{\tau,\tau'}$ denotes a path within the transportation plan $\mathcal{A}_{\tau,\tau'}$ which depends on lengths τ and τ' of sequences $\boldsymbol{\Psi} \equiv [\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_\tau] \in \mathbb{R}^{d' \times \tau}$, $\boldsymbol{\Psi}' \equiv [\boldsymbol{\psi}'_1, ..., \boldsymbol{\psi}'_{\tau'}] \in \mathbb{R}^{d' \times \tau'}$ and $\boldsymbol{D} \in \mathbb{R}_+^{\tau \times \tau'} \equiv [d_{\text{base}}(\boldsymbol{\psi}_m, \boldsymbol{\psi}'_n)]_{(m,n) \in \mathcal{I}_\tau \times \mathcal{I}_{\tau'}}$, the matrix of distances, is evaluated for $\tau \times \tau'$ frame representations according to some base distance $d_{\text{base}}(\cdot, \cdot)$, *i.e.*, the Euclidean or the RBF-induced distance. We make use of principles of soft-DTW. However, we design a joint alignment between temporal skeleton sequences and simulated skeleton viewpoints, an entirely novel proposal.

8.4 Approach

To learn similarity/dissimilarity between pairs of sequences of 3D body joints representing query and support samples from episodes, our goal is to find a smooth joint viewpoint-temporal alignment of query and support and minimize or maximize the matching distance d_{JEANIE} (end-to-end setting) for same or different support-query labels, respectively. Fig. 8.2 (top) shows that sometimes matching of query and support may be as easy as rotating one trajectory onto another, in order to achieve viewpoint invariance. A viewpoint invariant distance [Haasdonk and Burkhardt, 2007] can be defined as:

$$d_{\text{inv}}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') = \inf_{\gamma, \gamma' \in T} d\big(\gamma(\boldsymbol{\Psi}), \gamma'(\boldsymbol{\Psi}')\big), \tag{8.9}$$

where T is a set of transformations required to achieve a viewpoint invariance, $d(\cdot, \cdot)$ is some base distance, *e.g.*, the Euclidean distance, and $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}'$ are features describing query and support pair of sequences. Typically, T may include 3D rotations to rotate one trajectory onto the other. However, such a global viewpoint alignment of two sequences is suboptimal. Trajectories are unlikely to be straight 2D lines in the 3D space. Fig. 8.2 (bottom) shows that 3D body joints locally follow complicated

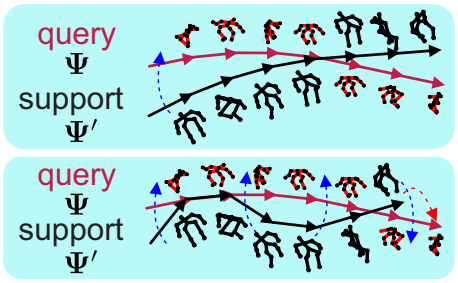


Figure 8.2: (*top*) In viewpoint-invariant learning, the distance between query features $\boldsymbol{\Psi}$ and support features $\boldsymbol{\Psi}'$ has to be computed. The blue arrow indicates that trajectories of both actions need alignment. (*bottom*) In real life, subject's 3D body joints deviate from one ideal trajectory, and so advanced viewpoint alignment strategy is needed.

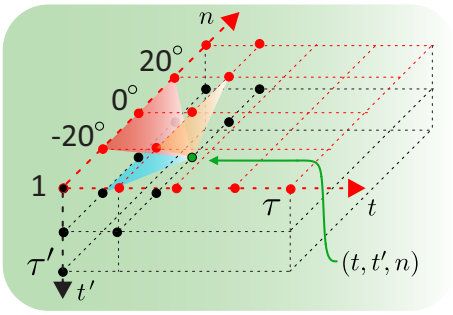


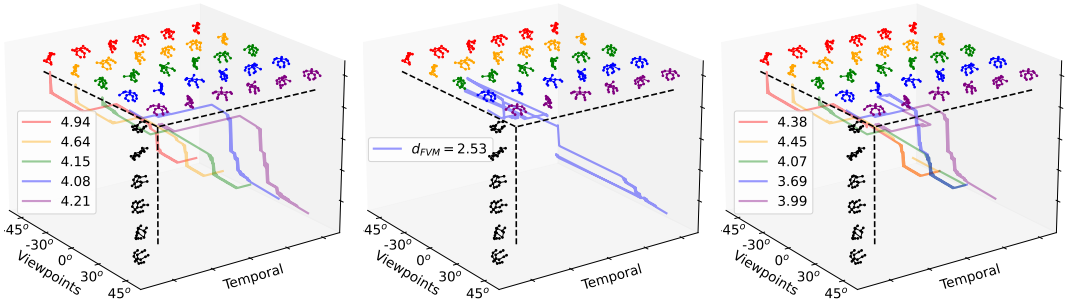
Figure 8.3: JEANIE (1-max shift). We loop over all points. At (t,t',n) (green point) we add its base distance to the minimum of accumulated distances at $(t,t'-1,n-1)$, $(t,t'-1,n)$, $(t,t'-1,n+1)$ (orange plane), $(t-1,t'-1,n-1)$, $(t-1,t'-1,n)$, $(t-1,t'-1,n+1)$ (red plane) and $(t-1,t',n-1)$, $(t-1,t',n)$, $(t-1,t',n+1)$ (blue plane).

non-linear paths.

Thus, we propose JEANIE that aligns and warps query/support sequences based on the feature similarity. One can think of JEANIE as performing Eq. (8.9) with T containing camera viewpoint rotations, and the base distance $d(\cdot,\cdot)$ being a joint temporal-viewpoint variant of soft-DTW to account for local temporal-viewpoint variations of 3D body joint trajectories. JEANIE unit in Fig. 8.1 realizes such a strategy (SoftMin operation is equivalent of Eq. (8.9)). While such an idea sounds simple, it is effective, it has not been done before. Fig. 8.3 (discussed later in the text) shows one step of the temporal-viewpoint computations of JEANIE. Below, we detail our pipeline shown in Figure 8.1, explain the proposed JEANIE and our loss function. Firstly, we present our notations.

Notations. $\mathcal{I}_K$ stands for the index set $\{1,2,...,K\}$. Concatenation of α_i is denoted by $[\alpha_i]_{i\in\mathcal{I}_I}$, whereas $\mathbf{X}_{:,i}$ means we extract/access column i of matrix $\boldsymbol{D}$. Calligraphic mathcal fonts denote tensors (*e.g.*, $\boldsymbol{\mathcal{D}}$), capitalized bold symbols are matrices (*e.g.*, $\boldsymbol{D}$), lowercase bold symbols are vectors (*e.g.*, $\boldsymbol{\psi}$), and regular fonts denote scalars.

Encoding Network (EN). We start by generating $K\times K'$ Euler rotations or $K\times K'$ simulated camera views (moved gradually from the estimated camera location) of query skeletons. Our EN contains a simple 3-layer MLP unit (FC, ReLU, FC, ReLU, Dropout, FC), GNN, optional Transformer [Dosovitskiy et al., 2020] and FC. The MLP unit takes M neighboring frames, each with J 3D skeleton body joints, forming one temporal block. In total, depending on stride S, we obtain some τ temporal blocks which capture the short temporal dependency, whereas the long temporal dependency is modeled with our JEANIE. Each temporal block is encoded by the MLP into a $d\times J$ dimensional feature map. Subsequently, query feature maps of size $K\times K'\times\tau$ and support feature maps of size τ' are forwarded to a GNN, optional Transformer (similar to ViT [Dosovitskiy et al., 2020], instead of using image patches, we feed each



(a) soft-DTW (view-wise) (b) FVM (c) JEANIE (1-max shift)

Figure 8.4: A comparison of paths in 3D for soft-DTW, Free Viewpoint Matching (FVM) and our JEANIE. For a given support skeleton sequence (green color), we choose viewing angles between $-45°$ and $45°$ for the camera viewpoint simulation. The support skeleton sequence is shown in black color. (a) soft-DTW finds each individual alignment per viewpoint fixed throughout alignment: $d_{\text{shortest}} = 4.08$. (b) FVM is a greedy matching algorithm that in each time step seeks the best alignment pose from all viewpoints which leads to unrealistic zigzag path (person cannot jump from front to back view suddenly): $d_{\text{FVM}} = 2.53$. (c) Our JEANIE (1-max shift) is able to find smooth joint viewpoint-temporal alignment between support and query sequences. We show each optimal path for each possible starting position: $d_{\text{JEANIE}} = 3.69$. While $d_{\text{FVM}} = 2.53$ for FVM is overoptimistic, $d_{\text{shortest}} = 4.08$ for fixed-view matching is too pessimistic, whereas JEANIE strikes the right matching balance with $d_{\text{JEANIE}} = 3.69$.

body joint encoded by GNN into the transformer), and an FC layer, which returns $\boldsymbol{\Psi} \in \mathbb{R}^{d' \times K \times K' \times \tau}$ query feature maps and $\boldsymbol{\Psi}' \in \mathbb{R}^{d' \times \tau'}$ support feature maps. Feature maps are passed to JEANIE and the similarity classifier.

Let support maps $\boldsymbol{\Psi}'$ be $[f(\boldsymbol{X}'_1; \mathcal{F}), ..., f(\boldsymbol{X}'_{\tau'}; \mathcal{F})] \in \mathbb{R}^{d' \times \tau'}$ and query maps $\boldsymbol{\Psi}$ be $[f(\boldsymbol{X}_1; \mathcal{F}), ..., f(\boldsymbol{X}_{\tau}; \mathcal{F})] \in \mathbb{R}^{d' \times K \times K' \times \tau}$, for query and support frames per block $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{3 \times J \times M}$. Moreover, we define $f(\mathbf{X}; \mathcal{F}) = \text{FC}(\text{Transf}(\text{GNN}(\text{MLP}(\mathbf{X}; \mathcal{F}_{MLP}); \mathcal{F}_{GNN}); \mathcal{F}_{Transf}); \mathcal{F}_{FC})$, $\mathcal{F} \equiv [\mathcal{F}_{MLP}, \mathcal{F}_{GNN}, \mathcal{F}_{Transf}, \mathcal{F}_{FC}]$ is the set of parameters of EN (note optional Transformer [Dosovitskiy et al., 2020]). As GNN, we try GCN [Kipf and Welling, 2017], SGC [Wu et al., 2019b], APPNP [Klicpera et al., 2019] or S^{2}GC [Zhu and Koniusz, 2021b].

JEANIE. Matching query-support pairs requires temporal alignment due to potential offset in locations of discriminative parts of actions, and due to potentially different dynamics/speed of actions taking place. The same concerns the direction of the dominant action trajectory w.r.t. the camera. Thus, JEANIE, our advanced soft-DTW, has the transportation plan $\mathcal{A}' \equiv \mathcal{A}_{\tau,\tau',K,K'}$, where apart from temporal block counts τ and τ', for query sequences, we have possible η_{az} left and η_{az} right steps from the initial camera azimuth, and η_{alt} up and η_{alt} down steps from the initial camera altitude. Thus, $K = 2\eta_{az}+1$, $K' = 2\eta_{alt}+1$. For the variant with Euler angles, we simply have $\mathcal{A}'' \equiv \mathcal{A}_{\tau,\tau',K,K'}$ where $K = 2\eta_x+1$, $K' = 2\eta_y+1$ instead.

Then, JEANIE is given as:

$$d_{\text{JEANIE}}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') = \underset{\mathbf{A}\in\mathcal{A}'}{\text{SoftMin}_\gamma} \left\langle \mathbf{A}, \boldsymbol{\mathcal{D}}(\boldsymbol{\Psi}, \boldsymbol{\Psi}') \right\rangle, \tag{8.10}$$

where $\boldsymbol{\mathcal{D}} \in \mathbb{R}_+^{K\times K'\times\tau\times\tau'} \equiv [d_{\text{base}}(\boldsymbol{\psi}_{m,k,k'}, \boldsymbol{\psi}'_n)]_{\substack{(m,n)\in\mathcal{I}_\tau\times\mathcal{I}_{\tau'} \\ (k,k')\in\mathcal{I}_K\times\mathcal{I}_{K'}}}$ and $\boldsymbol{\mathcal{D}}$ contains distances.

Figure 8.3 shows one step of JEANIE (1-max shift). Suppose the given viewing angle set is $\{-40°, -20°, 0°, 20°, 40°\}$. For 1-max shift, we loop over (t, t', n). At location (t, t', n), we extract the base distance and add it together with the minimum of aggregated distances at the shown 9 predecessor points. We store that total distance at (t, t', n), and we move to the next point. Note that for viewpoint index n, we look up $(n-1, n, n+1)$. Extension to the ι-max shift is straightforward.

Algorithm 1 illustrates JEANIE. For brevity, let us tackle the camera viewpoint alignment in a single space, *e.g.*, for some shifting steps $-\eta, ..., \eta$, each with size $\Delta\theta_{az}$. The maximum viewpoint change from block to block is ι-max shift (smoothness). As we have no way to know the initial optimal camera shift, we initialize all possible origins of shifts in accumulator $r_{n,1,1} = d_{\text{base}}(\boldsymbol{\psi}_{n,1}, \boldsymbol{\psi}'_1)$ for all $n \in \{-\eta, ..., \eta\}$. Subsequently, a phase related to soft-DTW (temporal-viewpoint alignment) takes place. Finally, we choose the path with the smallest distance over all possible viewpoint ends by selecting a soft-minimum over $[r_{n,\tau,\tau'}]_{n\in\{-\eta,...,\eta\}}$. Notice that accumulator $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{(2\iota+1)\times\tau\times\tau'}$. Moreover, whenever either index $n-i$, $t-j$ or $t'-k$ in $r_{n-i,t-j,t'-k}$ (see algorithm) is out of bounds, we define $r_{n-i,t-j,t'-k} = \infty$.

FVM. To ascertain whether JEANIE is better than performing separately the temporal and simulated viewpoint alignments, we introduce a baseline called the Free Viewpoint Matching (FVM). FVM, for every step of DTW, seeks the best local viewpoint alignment, thus realizing non-smooth temporal-viewpoint path in contrast to JEANIE. To this end, we apply DTW in Eq. (8.10) with the base distance replaced by:

$$d_{\text{FVM}(\boldsymbol{\psi}_t, \boldsymbol{\psi}'_{t'})} = \underset{m,n,m',n'\in\{-\eta,...,\eta\}}{\text{SoftMin}_{\bar{\gamma}}} d_{\text{base}}(\boldsymbol{\psi}_{m,n,t}, \boldsymbol{\psi}'_{m',n',t'}), \tag{8.11}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{d'\times K\times K'\times\tau}$ and $\boldsymbol{\Psi}' \in \mathbb{R}^{d'\times K\times K'\times\tau'}$ are query and support feature maps. We abuse the notation by writing $d_{\text{FVM}(\boldsymbol{\psi}_t, \boldsymbol{\psi}'_{t'})}$ as we minimize over viewpoint indexes in Eq. (8.11). We compute the distance matrix $\boldsymbol{D} \in \mathbb{R}_+^{\tau\times\tau'} \equiv [d_{\text{FVM}}(\boldsymbol{\psi}_t, \boldsymbol{\psi}'_{t'})]_{(t,t')\in\mathcal{I}_\tau\times\mathcal{I}_{\tau'}}$.

Fig. 8.4 shows the comparison between soft-DTW (view-wise), FVM and our JEANIE. FVM is a greedy matching method which leads to complex zigzag path in 3D space (assuming the camera viewpoint single space in $\boldsymbol{\psi}_{n,t}$ and no viewpoint in $\boldsymbol{\psi}'_{t'}$). Although FVM is able to find the smallest distance path compared to soft-DTW and JEANIE, it suffers from several issues (i) It is unreasonable for poses in a given sequence to match under sudden jumps in viewpoints. (ii) Suppose the two sequences are from two different classes, FVM still yields the smallest distance (decreased inter-class variance).

Loss Function. For the N-way Z-shot problem, we have one query feature map

Algorithm 1 Joint tEmporal and cAmera viewpoiNt alIgnmEnt (JEANIE).

Input (forward pass): $\boldsymbol{\Psi}, \boldsymbol{\Psi}'$, $\gamma > 0$, $d_{\text{base}}(\cdot,\cdot)$, ι-max shift.

$r_{:,:,:} = \infty$, $r_{n,1,1} = d_{\text{base}}(\boldsymbol{\psi}_{n,1}, \boldsymbol{\psi}'_1),\ \forall n \in \{-\eta, ..., \eta\}$
$\mathit{\Pi} \equiv \{-\iota, ..., 0, ..., \iota\} \times \{(0,1), (1,0), (1,1)\}$
for $t \in \mathcal{I}_\tau$:
  for $t' \in \mathcal{I}_{\tau'}$:
    if $t \neq 1$ or $t' \neq 1$:
      for $n \in \{-\eta, ..., \eta\}$:
        $r_{n,t,t'} = d_{\text{base}}(\boldsymbol{\psi}_{n,t}, \boldsymbol{\psi}'_{t'}) + \text{SoftMin}_\gamma\Big([r_{n-i,t-j,t'-k}]_{(i,j,k)\in\mathit{\Pi}}\Big)$

Output: $\text{SoftMin}_\gamma\Big([r_{n,\tau,\tau'}]_{n\in\{-\eta,...,\eta\}}\Big)$

and $N \times Z$ support feature maps per episode. We form a mini-batch containing B episodes. Thus, we have query feature maps $\{\boldsymbol{\Psi}_b\}_{b\in\mathcal{I}_B}$ and support feature maps $\{\boldsymbol{\Psi}'_{b,n,z}\}_{b\in\mathcal{I}_B, n\in\mathcal{I}_N, z\in\mathcal{I}_Z}$. Moreover, $\boldsymbol{\Psi}_b$ and $\boldsymbol{\Psi}'_{b,1,:}$ share the same class, one of N classes drawn per episode, forming the subset $C^\ddagger \equiv \{c_1, ..., c_N\} \subset \mathcal{I}_C \equiv \mathcal{C}$. To be precise, labels $y(\boldsymbol{\Psi}_b) = y(\boldsymbol{\Psi}'_{b,1,z}), \forall b \in \mathcal{I}_B, z \in \mathcal{I}_Z$ while $y(\boldsymbol{\Psi}_b) \neq y(\boldsymbol{\Psi}'_{b,n,z}), \forall b \in \mathcal{I}_B, n \in \mathcal{I}_N \setminus \{1\}, z \in \mathcal{I}_Z$. In most cases, $y(\boldsymbol{\Psi}_b) \neq y(\boldsymbol{\Psi}_{b'})$ if $b \neq b'$ and $b, b' \in \mathcal{I}_B$. Selection of $C^\ddagger$ per episode is random.

For the N-way Z-shot protocol, we minimize:

$$l(\boldsymbol{d}^+, \boldsymbol{d}^-) = \Big(\mu(\boldsymbol{d}^+) - \{\mu(\text{TopMin}_\beta(\boldsymbol{d}^+))\}\Big)^2 \tag{8.12}$$

$$+ \Big(\mu(\boldsymbol{d}^-) - \{\mu(\text{TopMax}_{NZ\beta}(\boldsymbol{d}^-))\}\Big)^2, \tag{8.13}$$

where $\boldsymbol{d}^+ = [d_{\text{JEANIE}}(\boldsymbol{\Psi}_b, \boldsymbol{\Psi}'_{b,1,z})]_{\substack{b\in\mathcal{I}_B \\ z\in\mathcal{I}_Z}}$ and $\boldsymbol{d}^- = [d_{\text{JEANIE}}(\boldsymbol{\Psi}_b, \boldsymbol{\Psi}'_{b,n,z})]_{\substack{b\in\mathcal{I}_B, \\ n\in\mathcal{I}_N\setminus\{1\}, z\in\mathcal{I}_Z}}$,

where $\boldsymbol{d}^+$ is a set of within-class distances for the mini-batch of size B given N-way Z-shot learning protocol. By analogy, $\boldsymbol{d}^-$ is a set of between-class distances. Function $\mu(\cdot)$ is simply the mean over coefficients of the input vector, $\{\cdot\}$ detaches the graph during the backpropagation step, whereas $\text{TopMin}_\beta(\cdot)$ and $\text{TopMax}_{NZ\beta}(\cdot)$ return β smallest and $NZ\beta$ largest coefficients from the input vectors, respectively. Thus, Eq. (8.12) promotes the within-class similarity while Eq. (8.13) reduces the between-class similarity. Integer $\beta \geq 0$ controls the focus on difficult examples, *e.g.*, $\beta = 1$ encourages all within-class distances in Eq. (8.12) to be close to the positive target $\mu(\text{TopMin}_\beta(\cdot))$, the smallest observed within-class distance in the mini-batch. If $\beta > 1$, this means we relax our positive target. By analogy, if $\beta = 1$, we encourage all between-class distances in Eq. (8.13) to approach the negative target $\mu(\text{TopMax}_{NZ\beta}(\cdot))$, the average over the largest NZ between-class distances. If $\beta > 1$, the negative target is relaxed.

8.5 Experiments

We provide network configurations and training details in Sec. 7.7. Below, we describe the datasets and evaluation protocols on which we validate our JEANIE.
Datasets. Table 8.9 contain details of datasets described below. Sec. B.1 of Appendix details the class concepts per split for small datasets.

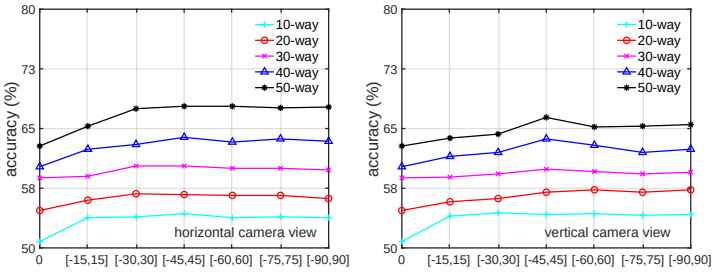
i. *UWA3D Multiview Activity II* [Rahmani et al., 2016b] contains 30 actions performed by 9 people in a cluttered environment. In this dataset, the Kinect camera was moved to different positions to capture the actions from 4 different views: front view (V_1), left view (V_2), right view (V_3), and top view (V_4).

ii. *NTU RGB+D (NTU-60)* [Shahroudy et al., 2016a] contains 56,880 video sequences and over 4 million frames. This dataset has variable sequence lengths and high intra-class variations.

iii. *NTU RGB+D 120 (NTU-120)* [Liu et al., 2019a], an extension of NTU-60, contains 120 action classes (daily/health-related), and 114,480 RGB+D video samples captured with 106 distinct human subjects from 155 different camera viewpoints.

iv. *Kinetics* [Kay et al., 2017] is a large-scale collection of 650,000 video clips that cover 400/600/700 human action classes. It includes human-object interactions such as *playing instruments*, as well as human-human interactions such as *shaking hands* and *hugging*. As the Kinetics-400 dataset provides only the raw videos, we follow approach [Yan et al., 2018] and use the estimated joint locations in the pixel coordinate system as the input to our pipeline. To obtain the joint locations, we first resize all videos to the resolution of 340 $\times$ 256, and convert the frame rate to 30 FPS. Then we use the publicly available *OpenPose* [Cao et al., 2017] toolbox to estimate the location of 18 joints on every frame of the clips. As OpenPose produces the 2D body joint coordinates and Kinetics-400 does not offer multiview or depth data, we use a network of Martinez *et al*. [Martinez et al., 2017] pre-trained on Human3.6M [Catalin et al., 2014], combined with the 2D OpenPose output to estimate 3D coordinates from 2D coordinates. The 2D OpenPose and the latter network give us (x, y) and z coordinates, respectively.

Evaluation protocols. For the UWA3D Multiview Activity II, we use standard multi-view classification protocol [Rahmani et al., 2016b; Wang, 2017; Wang et al., 2019b], but we apply it to one-shot learning as the view combinations for training and testing sets are disjoint. For NTU-120, we follow the standard one-shot protocol [Liu et al., 2019a]. Based on this protocol, we create a similar one-shot protocol for NTU-60, with 50/10 action classes used for training/testing respectively. To evaluate the effectiveness of the proposed method on viewpoint alignment, we also create two new protocols on NTU-120, for which we group the whole dataset based on (i) horizontal camera views into left, center and right views, (ii) vertical camera views into top, center and bottom views. We conduct two sets of experiments on such disjoint view-wise splits: (i) using 100 action classes for training, and testing on the same 100 action classes (ii) training



(a) horizontal camera view (b) vertical camera view

Figure 8.5: The impact of viewing angles on NTU-60.

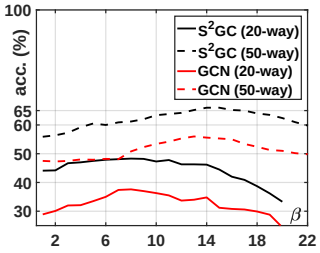


Figure 8.6: The impact of β in loss function on NTU-60 with S^{2}GC and GCN.

on 100 action classes but testing on the rest unseen 20 classes. Appendix Sec. B details new/additional eval. protocols on NTU-60/NTU-120.

Stereo projections. For simulating different camera viewpoints, we estimate the fundamental matrix F (Eq. (8.2)), which relies on camera parameters. Thus, we use the Camera Calibrator from MATLAB to estimate intrinsic, extrinsic and lens distortion parameters. For a given skeleton dataset, we compute the range of spatial coordinates x and y, respectively. We then split them into 3 equally-sized groups to form roughly left, center, right views and other 3 groups for bottom, center, top views. We choose $\sim$15 frame images from each corresponding group, upload them to the Camera Calibrator, and export camera parameters. We then compute the average distance/depth and height per group to estimate the camera position. On NTU-60 and NTU-120, we simply group the whole dataset into 3 cameras, which are left, center and right views, as provided in [Liu et al., 2019a], and then we compute the average distance per camera view based on the height and distance settings given in the table in [Liu et al., 2019a].

8.5.1 Ablation Study

We start our experiments by investigating the GNN backbones (Sec. 8.7.1), camera viewpoint simulation and their hyper-parameters (Sec. 8.7.3, 8.7.4, 8.7.5).

Camera viewpoint simulations. We choose 15 degrees as the step size for the viewpoints simulation. The ranges of camera azimuth/altitude are in $[-90°, 90°]$. Where stated, we perform a grid search on camera azimuth/altitude with Hyperopt. Below, we explore the choice of the angle ranges for both horizontal and vertical views. Fig. 8.5a and 8.5b (evaluations on the NTU-60 dataset) show that the angle range $[-45°, 45°]$ performs the best, and widening the range in both views does not increase the performance any further. Table 8.1 (top) shows results for the chosen range $[-45°, 45°]$ of camera viewpoint simulations. Euler simple $(K+K')$ denotes a simple concatenation of features from both horizontal and vertical views, whereas Euler/CamVPC$(K\times K')$ represents the grid search of all possible views. It shows that Euler angles for the viewpoint augmentation outperform Euler simple, and CamVPC (viewpoints of query sequences are generated by the stereo projection geometry)

Table 8.1: Experimental results on NTU-60 (left) and NTU-120 (right) for different camera viewpoint simulations. Below the dashed line are ablated few variants of JEANIE.

# Training Classes	NTU-60					NTU-120				
	10	20	30	40	50	20	40	60	80	100
Euler simple ($K+K'$)	54.3	56.2	60.4	64.0	68.1	30.7	36.8	39.5	44.3	46.9
Euler ($K\times K'$)	**60.8**	67.4	67.5	70.3	**75.0**	32.9	39.2	43.5	48.4	50.2
CamVPC ($K\times K'$)	59.7	**68.7**	**68.4**	**70.4**	73.2	**33.1**	**40.8**	**43.7**	**48.4**	**51.4**
V(Euler)	54.0	56.0	60.2	63.8	67.8	30.6	36.7	39.2	44.0	47.0
2V(Euler simple)	54.3	56.2	60.4	64.0	68.1	30.7	36.8	39.5	44.3	46.9
2V(Euler)	60.8	67.4	67.5	70.3	75.0	32.9	39.2	43.5	48.4	50.2
2V(CamVPC)	59.7	68.7	68.4	70.4	73.2	33.1	40.8	43.7	48.4	51.4
2V(CamVPC+crossval.)	63.4	72.4	73.5	73.2	78.1	37.2	43.0	49.2	50.0	55.2
2V(CamVPC+crossval.)+Transf.	**65.0**	**75.2**	**76.7**	**78.9**	**80.0**	**38.5**	**44.1**	**50.3**	**51.2**	**57.0**

Table 8.2: Experimental results on NTU-60 (left) and NTU-120 (right) for ι-max shift. ι-max shift is the max. viewpoint shift from block to block in JEANIE.

	NTU-60					NTU-120				
	10	20	30	40	50	20	40	60	80	100
$\iota=1$	60.8	70.7	72.5	72.9	75.2	36.3	42.5	48.7	**50.0**	54.8
$\iota=2$	**63.8**	**72.9**	**74.0**	**73.4**	**78.1**	**37.2**	**43.0**	**49.2**	**50.0**	**55.2**
$\iota=3$	55.2	58.9	65.7	67.1	72.5	36.7	**43.0**	48.5	49.0	54.9
$\iota=4$	54.5	57.8	63.5	65.2	70.4	36.5	42.9	48.3	48.9	54.3

outperforms Euler angles in almost all the experiments on NTU-60 and NTU-120. This proves the effectiveness of using the stereo projection geometry for the viewpoint augmentation. More baseline experiments with/without viewpoint alignment are in Sec. 8.7.2.

Evaluation of β. Figure 8.6 shows that if $\beta=8$ and 14, our loss function performs the best on 20- and 50-class protocol, respectively, on NTU-60 for the S^{2}GC and GCN backbone. Moreover, β is not affected by backbone.

The ι-max shift. Table 8.2 shows the evaluations of ι for the maximum shift. We notice that $\iota=2$ yields the best results for all the experimental settings on both NTU-60 and NTU-120. Increasing ι does not help improve the performance.

Block size and strides. Table 8.3 shows evaluations of block size M and stride S, and indicates that the best performance (both 50- and 20-class) is achieved for smaller block size (frame count in the block) and smaller stride. Longer temporal blocks decrease the performance due to the temporal information not reaching the temporal alignment step. Our block encoder encodes each temporal block for learning the local temporal motions, and aggregate these block features finally to form the global temporal motion cues. Smaller stride helps capture more local motion patterns.

Table 8.3: The impact of the number of frames M in temporal block under stride step S on results (NTU-60). $S = pM$, where $1-p$ describes the temporal block overlap percentage. Higher p means fewer overlap frames between temporal blocks.

	$S=M$		$S=0.8M$		$S=0.6M$		$S=0.4M$		$S=0.2M$	
M	50-class	20-class	50-class	20-class	50-class	20-class	50-class	20-class	50-class	20-class
5	69.0	55.7	71.8	57.2	69.2	59.6	73.0	60.8	71.2	61.2
6	69.4	54.0	65.4	54.1	67.8	58.0	72.0	57.8	**73.0**	**63.0**
8	67.0	52.7	67.0	52.5	**73.8**	**61.8**	67.8	60.3	68.4	59.4
10	62.2	44.5	63.6	50.9	65.2	48.4	62.4	57.0	70.4	56.7
15	62.0	43.5	62.6	48.9	64.7	47.9	62.4	57.2	68.3	56.7
30	55.6	42.8	57.2	44.8	59.2	43.9	58.8	55.3	60.2	53.8
45	50.0	39.8	50.5	40.6	52.3	39.9	53.0	42.1	54.0	45.2

Table 8.4: Results on NTU-60 (S^2GC backbone). Models use temporal alignment by soft-DTW or JEANIE (joint temporal-viewpoint alignment) except if indicated otherwise.

# Training Classes	10	20	30	40	50
Each frame to frontal view	52.9	53.3	54.6	54.2	58.3
Each block to frontal view	53.9	56.1	60.1	63.8	68.0
Traj. aligned baseline (video-level)	36.1	40.3	44.5	48.0	50.2
Traj. aligned baseline (block-level)	52.9	55.8	59.4	63.6	66.7
Matching Nets [Vinyals et al., 2016]	46.1	48.6	53.3	56.2	58.8
Matching Nets [Vinyals et al., 2016]+2V	47.2	50.7	55.4	57.7	60.2
Prototypical Net [Snell et al., 2017]	47.2	51.1	54.3	58.9	63.0
Prototypical Net [Snell et al., 2017]+2V	49.8	53.1	56.7	60.9	64.3
TAP [Su and Wen, 2022]	54.2	57.3	61.7	64.7	68.3
S^2GC (no soft-DTW)	50.8	54.7	58.8	60.2	62.8
soft-DTW	53.7	56.2	60.0	63.9	67.8
(no soft-DTW)+Transf.	56.0	64.2	67.3	70.2	72.9
soft-DTW+Transf.	57.3	66.1	68.8	72.3	74.0
JEANIE+Transf.	**65.0**	**75.2**	**76.7**	**78.9**	**80.0**

Considering the computational cost and the performance, we choose $M = 8$ and $S = 0.6M$.

Euler *vs.* CamVPC. Table 8.1 (bottom) shows that using the viewpoint alignment simultaneously in two dimensions, x and y for Euler angles, or azimuth and altitude the stereo projection geometry (*CamVPC*), improves the performance by 5-8% compared to (*Euler simple*), a variant where the best viewpoint alignment path was chosen from the best alignment path along x and the best alignment path along y. Euler simple is better than Euler with y rotations only ((V) includes rotations along y while $(2V)$ includes rotations along two axes). Using HyperOpt [Bergstra et al., 2015] to search for the best angle range in which we perform the viewpoint alignment (*CamVPC+crossval.*) improves results. Enabling the viewpoint alignment for support sequences yields extra improvement. With Transformer (*2V+Transf.*), JEANIE boosts results by $\sim 2\%$.

Table 8.5: Experimental results on NTU-120 (S^2GC backbone). Methods use temporal alignment by soft-DTW or JEANIE (joint temporal-viewpoint alignment) except VA [Zhang et al., 2017b, 2019d] and other cited works. For VA*, we used soft-DTW on temporal blocks while VA generated temporal blocks.

# Training Classes	20	40	60	80	100
APSR [Liu et al., 2019a]	29.1	34.8	39.2	42.8	45.3
SL-DML [Memmesheimer et al., 2020]	36.7	42.4	49.0	46.4	50.9
Skeleton-DML [Memmesheimer et al., 2021]	28.6	37.5	48.6	48.0	54.2
Prototypical Net+VA-RNN(aug.) [Zhang et al., 2017b]	25.3	28.6	32.5	35.2	38.0
Prototypical Net+VA-CNN(aug.) [Zhang et al., 2019d]	29.7	33.0	39.3	41.5	42.8
Prototypical Net+VA-fusion(aug.) [Zhang et al., 2019d]	29.8	33.2	39.5	41.7	43.0
Prototypical Net+VA*-fusion(aug.) [Zhang et al., 2019d]	33.3	38.7	45.2	46.3	49.8
TAP [Su and Wen, 2022]	31.2	37.7	40.9	44.5	47.3
S^2GC(no soft-DTW)	30.0	35.9	39.2	43.6	46.4
soft-DTW	30.3	37.2	39.7	44.0	46.8
(no soft-DTW)+Transf.	31.2	37.5	42.3	47.0	50.1
soft-DTW+Transf.	31.6	38.0	43.2	47.8	51.3
FVM+Transf.	34.5	41.9	44.2	48.7	52.0
JEANIE+Transf.	**38.5**	**44.1**	**50.3**	**51.2**	**57.0**

8.5.2 Comparisons With the State-of-the-Art Methods

One-shot action recognition (NTU-60). Table 8.4 shows that aligning query and support trajectories by the angle of torso 3D joint, denoted (*Traj. aligned baseline*) is not very powerful, as alluded to in Figure 8.2 (top). Aligning piece-wise parts (blocks) is better than aligning entire trajectories. In fact, aligning individual frames by torso to the frontal view (*Each frame to frontal view*) and aligning block average of torso direction to the frontal view (*Each block to frontal view*)) were marginally better. We note these baselines use soft-DTW. We show more comparisons in Sec. 8.8. Our JEANIE with Transformer (*JEANIE+Transf.*) outperforms soft-DTW with Transformer (*soft-DTW+Transf.*) by 7.46% on average.

One-shot action recognition (NTU-120). Table 8.5 shows that JEANIE outperforms recent SL-DML and Skeleton-DML by 6.1% and 2.8% respectively (100 training classes). For comparisons, we extended the view adaptive neural networks [Zhang et al., 2019d] by combining them with Prototypical Net [Snell et al., 2017]. VA-RNN+VA-CNN [Zhang et al., 2019d] uses 0.47M+24M parameters with random rotation augmentations while JEANIE uses 0.25–0.5M params. Their *rotation+translation* keys are not proven to perform smooth optimal alignment as JEANIE. In contrast, d_{JEANIE} performs jointly a smooth viewpoint-temporal alignment via a principled transportation plan ($\geq$3 dim. space) by design. Their use Euler angles which are a worse option than the camera projection of JEANIE. We notice that ProtoNet+VA backbones is 12% worse than our JEANIE. Even if we split skeletons into blocks to let soft-DTW perform temporal alignment of prototypes and query, JEANIE is still 4–6% better. JEANIE outperforms FVM by 2-4%. This shows that seeking jointly the best temporal-viewpoint alignment is more valuable than considering viewpoint alignment as a local task (free range alignment per each step of soft-DTW).

Table 8.6: Experiments on 2D and 3D Kinetics-skeleton. Note that we have no results on JEANIE or FVM for 2D coordinates (aligning viewpoints is an operation in 3D).

	S^2GC (no soft-DTW)	soft-DTW	*FVM*	JEANIE	JEANIE +Transf.
2D skel.	32.8	34.7	-	-	-
3D skel.	35.9	39.6	44.1	**50.3**	**52.5**

Table 8.7: Experiments on the UWA3D Multiview Activity II.

Training view	V_1 & V_2		V_1 & V_3		V_1 & V_4		V_2 & V_3		V_2 & V_4		V_3 & V_4		Mean
Testing view	V_3	V_4	V_2	V_4	V_2	V_3	V_1	V_4	V_1	V_3	V_1	V_2	
GCN	36.4	26.2	20.6	30.2	33.7	22.4	43.1	26.6	16.9	12.8	26.3	36.5	27.6
SGC	40.9	60.3	44.1	52.6	48.5	38.7	50.6	52.8	52.8	37.2	57.8	49.6	48.8
+soft-DTW	43.9	60.8	48.1	54.6	52.6	45.7	54.0	58.2	56.7	40.2	60.2	51.1	52.2
+JEANIE	47.0	62.8	50.4	57.8	53.6	47.0	57.9	62.3	57.0	44.8	61.7	52.3	54.6
APPNP	42.9	61.9	47.8	58.7	53.8	44.0	52.3	60.3	55.1	38.2	58.3	47.9	51.8
+soft-DTW	44.3	63.2	50.7	62.3	53.9	45.0	56.9	62.8	56.4	39.3	60.1	51.9	53.9
+JEANIE	46.8	64.6	51.3	65.1	54.7	46.4	58.2	65.1	58.8	43.9	60.3	52.5	55.6
S^2GC	45.5	64.4	46.8	61.6	49.5	43.2	57.3	61.2	51.0	42.9	57.0	49.2	52.5
+soft-DTW	48.2	67.2	51.2	67.0	53.2	46.8	62.4	66.2	57.8	45.0	62.2	53.0	56.7
+FVM	50.7	68.8	56.3	69.2	55.8	47.1	63.7	68.8	62.5	51.4	63.8	55.7	59.5
+JEANIE	**55.3**	**70.2**	**61.4**	**72.5**	**60.9**	**50.8**	**66.4**	**73.9**	**68.8**	**57.2**	**66.7**	**60.2**	**63.7**

JEANIE on the Kinetics-skeleton. We evaluate our proposed model on both 2D and 3D Kinetics-skeleton. We split the whole dataset into 200 actions for training, and the rest half for testing. As we are unable to estimate the camera location, we simply use Euler angles for the camera viewpoint simulation. Table 8.6 shows that using 3D skeletons outperforms the use of 2D skeletons by 3-4%, and JEANIE outperforms the baseline (temporal alignment only) and Free Viewpoint Matching (FVM, for every step of DTW, seeks the best local viewpoint alignment, thus realizing non-smooth temporal-viewpoint path in contrast to JEANIE) by around 5% and 6%, respectively. With the transformer, JEANIE further boosts results by 2%.

Few-shot multiview classification. Table 8.7 (UWA3D Multiview Activity II) shows that adding temporal alignment to SGC, APPNP and S^2GC improves the performance, and the big performance gain is obtained by further adding the viewpoint alignment. As this dataset is challenging in recognizing the actions from a novel view point, our proposed method performs consistently well on all different combinations of training/testing viewpoint variants. This is predictable as our method aligns both temporal and camera viewpoints which allows a robust classification. JEANIE outperforms FVM by 4.2%, and outperforms the baseline (with temporal alignment only) by 7% on average.

Table 8.8 (NTU-120) shows that adding more camera viewpoints to the training process helps the multiview classification. Using bottom and center views for training and top view for testing, or using left and center views for training and the right view for testing yields 4% gain (‘*same 100*’ means the same train/test classes but different views). Testing on 20 novel classes (‘*novel 20*’ never used in training) yields 62.7%

Table 8.8: Results on NTU-120 (multiview classification). Baseline is soft-DTW + S^{2}GC.

Training view	bott.	bott.	bott.& cent.	left	left	left & cent.
Testing view	cent.	top	top	cent.	right	right
100/same 100 (baseline)	74.2	73.8	75.0	58.3	57.2	68.9
100/same 100 (*FVM*)	79.9	78.2	80.0	65.9	63.9	75.0
100/same 100 (*JEANIE*)	**81.5**	**79.2**	**83.9**	**67.7**	**66.9**	**79.2**
100/novel 20 (baseline)	58.2	58.2	61.3	51.3	47.2	53.7
100/novel 20 (*FVM*)	66.0	65.3	68.2	58.8	53.9	60.1
100/novel 20 (*JEANIE*)	**67.8**	**65.8**	**70.8**	**59.5**	**55.0**	**62.7**

and 70.8% for multiview classification in horizontal and vertical camera viewpoints, respectively.

8.6 Network configuration and training details

Below we provide the details of network configuration and training process in the following sections.

8.6.1 Network configuration

Given the temporal block size M (the number of frames in a block) and desired output size d, the configuration of the 3-layer MLP unit is: FC ($3M \rightarrow 6M$), LayerNorm (LN) as in [Dosovitskiy et al., 2020], ReLU, FC ($6M \rightarrow 9M$), LN, ReLU, Dropout (for smaller datasets, the dropout rate is 0.5; for large-scale datasets, the dropout rate is 0.1), FC ($9M \rightarrow d$), LN. Note that M is the temporal block size and d is the output feature dimension per body joint. Note that ablations on the value of M are already conducted in Table 8.3.

Backbone with GNN and Transformer. Following EN described in Section 8.4, let us take the query input $\mathbf{X} \in \mathbb{R}^{3\times J\times M}$ for the temporal block of length M as an example, where 3 indicates 3D Cartesian coordinate and J is the number of body joints. As alluded to earlier, we obtain $\widehat{\mathbf{X}}^T = \text{MLP}(\mathbf{X}; \mathcal{F}_{MLP}) \in \mathbb{R}^{d\times J}$.

Subsequently, we employ a GNN and the transformer encoder [Dosovitskiy et al., 2020] which consists of alternating layers of Multi-Head Self-Attention (MHSA) and a feed-forward MLP (two FC layers with a GELU non-linearity between them). LayerNorm (LN) is applied before every block, and residual connections after every block. Each block feature matrix $\widehat{\mathbf{X}} \in \mathbb{R}^{J\times d}$ encoded by GNN (without learnable $\boldsymbol{\Theta}$) is then passed to the transformer. Similarly to the standard transformer, we prepend a learnable vector $\mathbf{y}_{\text{token}} \in \mathbb{R}^{1\times d}$ to the sequence of block features $\widehat{\mathbf{X}}$ obtained from GNN, and we also add the positional embeddings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(1+J)\times d}$ based on the sine and cosine functions (standard in transformers) so that token $\mathbf{y}_{\text{token}}$ and each body joint enjoy their own unique positional encoding. We obtain $\mathbf{Z}_0 \in \mathbb{R}^{(1+J)\times d}$ which is the

input in the following backbone:

$$\mathbf{Z}_0 = [\mathbf{y}_{\text{token}}; \text{GNN}(\widehat{\mathbf{X}})] + \mathbf{E}_{\text{pos}}, \tag{8.14}$$

$$\mathbf{Z}'_k = \text{MHSA}(\text{LN}(\mathbf{Z}_{k-1})) + \mathbf{Z}_{k-1}, \; k = 1, ..., L_{\text{tr}} \tag{8.15}$$

$$\mathbf{Z}_k = \text{MLP}(\text{LN}(\mathbf{Z}'_k)) + \mathbf{Z}'_k, \qquad k = 1, ..., L_{\text{tr}} \tag{8.16}$$

$$\mathbf{y}' = \text{LN}(\mathbf{Z}^{(0)}_{L_{\text{tr}}}) \qquad \text{where} \qquad \mathbf{y}' \in \mathbb{R}^{1\times d} \tag{8.17}$$

$$f(\mathbf{X}; \mathcal{F}) = \text{FC}(\mathbf{y}'^T; \mathcal{F}_{FC}) \qquad \in \mathbb{R}^{d'}, \tag{8.18}$$

where $\mathbf{Z}^{(0)}_{L_{\text{tr}}}$ is the first d dimensional row vector extracted from the output matrix $\mathbf{Z}_{L_{\text{tr}}}$ of size $(J+1) \times d$ which corresponds to the last layer L_{tr} of the transformer. Moreover, parameter L_{tr} controls the depth of the transformer, whereas $\mathcal{F} \equiv [\mathcal{F}_{MLP}, \mathcal{F}_{GNN}, \mathcal{F}_{Transf}, \mathcal{F}_{FC}]$ is the set of parameters of EN. In case of APPNP, SGC and S^{2}GC, $|\mathcal{F}_{GNN}| = 0$ because we do not use their learnable parameters $\boldsymbol{\Theta}$ (*i.e.*, think $\boldsymbol{\Theta}$ is set as the identity matrix).

As in Section 8.4, one can define now a support feature map as $\boldsymbol{\Psi}' = [f(\boldsymbol{X}_1; \mathcal{F}), ..., f(\boldsymbol{X}_{\tau'}; \mathcal{F})] \in \mathbb{R}^{d' \times \tau'}$ for τ' temporal blocks, and the query map $\boldsymbol{\Psi}$ accordingly.

The hidden size of our transformer (the output size of the first FC layer of the MLP in Eq. (8.16)) depends on the dataset. For smaller datasets, the depth of the transformer is $L_{\text{tr}} = 6$ with 64 as the hidden size, and the MLP output size is $d = 32$ (note that the MLP which provides $\widehat{\mathbf{X}}$ and the MLP in the transformer must both have the same output size). For NTU-60, the depth of the transformer is $L_{\text{tr}} = 6$, the hidden size is 128 and the MLP output size is $d = 64$. For NTU-120, the depth of the transformer is $L_{\text{tr}} = 6$, the hidden size is 256 and the MLP size is $d = 128$. For Kinetics-skeleton, the depth for the transformer is $L_{\text{tr}} = 12$, hidden size is 512 and the MLP output size is $d = 256$. The number of Heads for the transformer of smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton is set as 6, 12, 12 and 12, respectively.

The output sizes d' of the final FC layer in Eq. (8.18) are 50, 100, 200, and 500 for the smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton, respectively.

8.6.2 Training details

The weights for the pipeline are initialized with the normal distr. (zero mean and unit standard dev.). We use 1e-3 for the learning rate, and the weight decay is 1e-6. We use the SGD optimizer. We set the number of training episodes to 100K for NTU-60, 200K for NTU-120, 500K for 3D Kinetics-skeleton, 10K for small datasets such as UWA3D Multiview Activity II. We use Hyperopt for hyperparam. search on validation sets for all the datasets.

Table 8.9: Seven publicly available benchmark datasets which we use for FSAR.

Datasets		Year	Classes	Subjects	#views	#clips	Sensor	Modalities	#joints
MSRAction3D	[Li et al., 2010]	2010	20	10	1	567	Kinect v1	Depth+3DJoints	20
3D Action Pairs	[Oreifej and Liu, 2013]	2013	12	10	1	360	Kinect v1	RGB+Depth+3DJoints	20
UWA3D Activity	[Rahmani et al., 2014b]	2014	30	10	1	701	Kinect v1	RGB+Depth+3DJoints	15
UWA3D Multiview Activity II	[Rahmani et al., 2016b]	2015	30	9	4	1,070	Kinect v1	RGB+Depth+3DJoints	15
NTU RGB+D	[Shahroudy et al., 2016a]	2016	60	40	80	56,880	Kinect v2	RGB+Depth+3DJoints	25
NTU RGB+D 120	[Liu et al., 2019a]	2019	120	106	155	114,480	Kinect v2	RGB+Depth+3DJoints	25
Kinetics-skeleton	[Yan et al., 2018]	2018	400	-	-	$\sim$ 300,000	-	RGB+2DJoints	18

Table 8.10: Evaluations of backbones on 5 datasets.

	MSRAction3D		3DAct.Pairs	UWA3DActivity		NTU-60	NTU-120
	5-way	10-way	5-way	5-way	10-way	50-way	20-way
GCN	56.0 ± 1.3	37.6 ± 1.2	-	55.4 ± 0.8	42.4 ± 0.8	56.0	-
SGC	66.0 ± 1.1	48.3 ± 1.1	69.0 ± 1.8	56.4 ± 0.7	41.6 ± 0.6	68.1	30.7
APPNP	67.2 ± 0.8	58.1 ± 0.8	69.0 ± 2.0	60.6 ± 1.5	42.4 ± 1.3	68.5	30.8
S^2GC (Eucl.)	68.8 ± 1.2	63.1 ± 0.9	72.2 ± 1.8	69.8 ± 0.7	58.3 ± 0.6	75.6	34.5
S^2GC (RBF)	$\mathbf{73.2 \pm 0.9}$	$\mathbf{64.6 \pm 0.8}$	$\mathbf{75.6 \pm 2.1}$	$\mathbf{76.4 \pm 0.7}$	$\mathbf{58.9 \pm 0.7}$	**78.1**	**36.2**

8.6.3 Skeleton Data Preprocessing

Before passing the skeleton sequences into MLP and graph networks (*e.g.*, S^2GC), we first normalize each body joint w.r.t. to the torso joint $\mathbf{v}_{f,c}$:

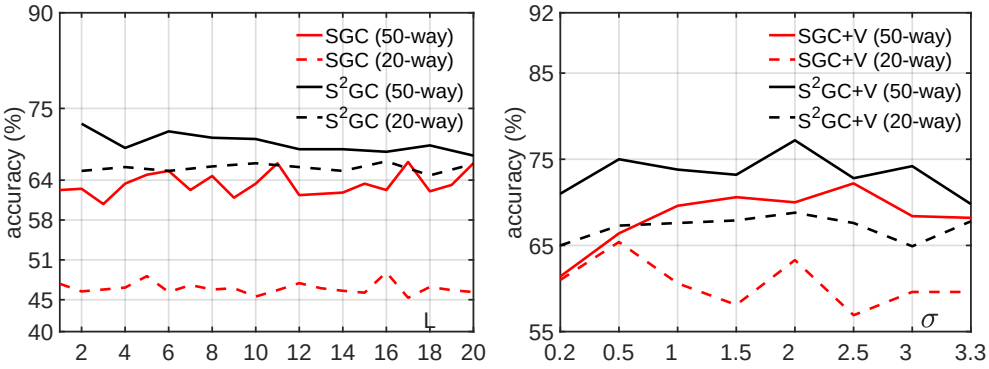
$$\mathbf{v}'_{f,i} = \mathbf{v}_{f,i} - \mathbf{v}_{f,c}, \tag{8.19}$$

where f and i are the index of video frame and human body joint respectively. After that, we further normalize each joint coordinate into [-1, 1] range:

$$\hat{\mathbf{v}}_{f,i}[j] = \frac{\mathbf{v}'_{f,i}[j]}{\max([\text{abs}(\mathbf{v}'_{f,i}[j])]_{f \in \mathcal{I}_\tau, i \in \mathcal{I}_J})}, \tag{8.20}$$

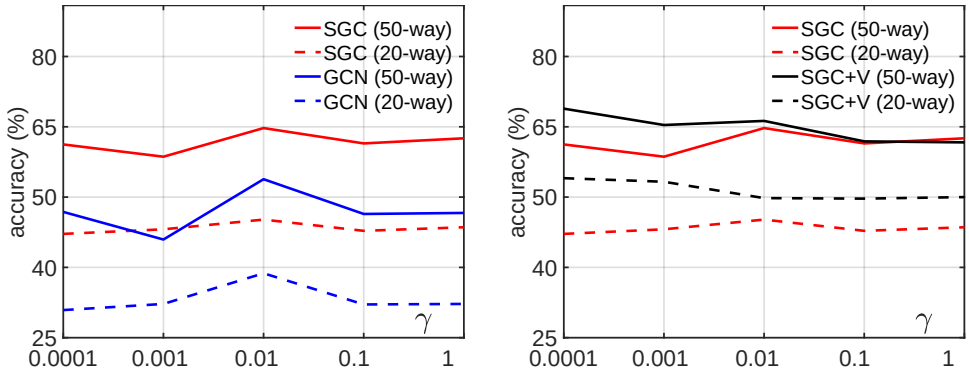
where j is for selection of the x, y and z axes, τ is the number of frames and J is the number of 3D body joints per frame.

For the skeleton sequences that have more than one performing subject, (i) we normalize each skeleton separately, and each skeleton is passed to MLP for learning the temporal dynamics, and (ii) for the output features per skeleton from MLP, we pass them separately to graph networks, *e.g.*, two skeletons from a given video sequence will have two outputs from the graph networks, and we aggregate the outputs through average pooling before passing to FVM or JEANIE.



(a) (b)

Figure 8.7: Evaluations of L and σ. (a): L for SGC and S^{2}GC. (b): σ of RBF distance for Eq. (8.10) (SGC and S^{2}GC, NTU-60).



(a) (b)

Figure 8.8: Evaluations w.r.t. γ. (a): γ in Eq. (8.10) with the temporal alignment alone. (b): comparisons of temporal alignment alone *vs*. temporal-viewpoint alignment (V) on NTU-60.

8.7 Backbone selection and hyperparameter evaluation

8.7.1 Backbone selection

We conduct experiments on 4 GNN backbones listed in Table 8.10. S^{2}GC performs the best on all datasets including large-scale NTU-60 and NTU-120, APPNP outperforms SGC, and SGC outperforms GCN. We note that using the RBF-induced distance for $d_{base}(\cdot,\cdot)$ of DTW outperforms the Euclidean distance. Fig. 8.7 shows a comparison of using SGC and S^{2}GC on NTU-60. As shown in the figure that using S^{2}GC performs better than SGC for both 50-way and 20-way settings. We also notice that results w.r.t. the number of layers L are more stable for S^{2}GC than SGC. We choose $L=6$ for our experiments. Fig. 8.7b shows evaluations of σ for the RBF-induced distance for both SGC and S^{2}GC. As $\sigma=2$ in S^{2}GC achieves the highest performance (both 50-way and 20-way), we choose S^{2}GC as the backbone and $\sigma=2$ for the experiments.

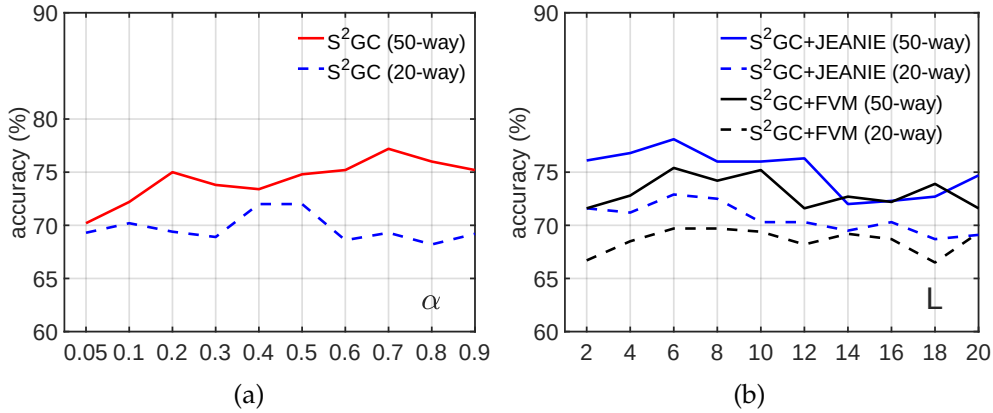


Figure 8.9: Evaluations of (a) α and (b) the number of layers L for S^{2}GC on NTU-60.

Table 8.11 is a comparison of CNN, RNN and GNN as backbones of JEANIE. The role of this backbone in JEANIE is to process the per-block per-viewpoint body joint features obtained from MLP and exploit interactions among body joints. The input and output feature dimension (per temporal block) of backbone (*e.g.*, CNN, RNN and GNN) are $J \times d$. For CNN, we simply use 2D convolution (square kernels and equal stride with auto padding to ensure the input and output feature maps have the same dimensions). For RNN (joint-wise), we use J RNN cells, each RNN processes d-dimensional vector for each body joint (J-input to J-output). Both input and output are d-dimensional feature vectors. The outputs from J RNN cells are concatenated to form $J \times d$ feature maps. For RNN (temporal-wise), we modify the first MLP in our pipeline, to produce $J \times d$ vector per each time-step $1, ..., T$ of temporal block. We use T RNN cells, each RNN processes $J \times d$-dimensional vector (T-input to 1-output). The output we use is from the last of T RNN cells is $J \times d$. As shown in the table, GNN outperforms RNN, and RNN outperforms CNN. Note that our GNN (S^{2}GC) is a simple linear projection on a spectral filter of graph and without learnable parameters. We believe RNN (temporal-wise) is better than RNN (joint-wise) as joints to not have well defined time-like order (thus RNN is bad) while GNN uses the graph connectivity (topological order).

Table 8.11: A comparison of different backbones in JEANIE on NTU-60 (#training classes = 10).

Backbones	CNN	RNN (joint-wise)	RNN (temporal-wise)	GNN (ours)
Results (acc.)	55.3	59.9	61.1	**65.0**

Table 8.12: Experimental results of stride for degrees on NTU-60.

	10	20	30	40	50
5°	63.8	73.7	74.2	75.0	76.5
10°	64.2	74.8	75.0	78.0	79.1
15°	**65.0**	**75.2**	**76.7**	**78.9**	**80.0**
30°	**65.0**	74.8	75.0	76.9	78.5
45°	60.0	68.5	71.0	71.5	72.0

8.7.2 Evaluations of viewpoint alignment

Fig. 8.8 shows comparisons with temporal-viewpoint alignment (V) *vs*. temporal alignment alone on NTU-60.

Fig. 8.8a shows evaluations of γ without the viewpoint alignment. Fig. 8.8b shows that temporal-viewpoint alignment (V) brings around 5% (20- and 50-way protocols, $\gamma = 0.0001$) improvement.

8.7.3 Evaluations w.r.t. α

Figure 8.9a shows the evaluations of α for the S^{2}GC backbone. As shown in the plot, for 50-way protocol, the best performance is achieved when $\alpha = 0.7$ ($\alpha = 0.5$ is the second best performer for the 50-way protocol)). For the 20-way protocol, the top performer is $\alpha = 0.4$ or $\alpha = 0.5$. Thus, we chose $\alpha = 0.5$ in our experiments. Please note we observed the same trend on the validation split.

8.7.4 Evaluations w.r.t. the number of layers L

Figure 8.9b shows the performance w.r.t. the number of layers L used by S^{2}GC and S^{2}GC+JEANIE. As shown in this plot, when $L = 6$, S^{2}GC with JEANIE performs the best for both 20- and 50-way experiments. For the Free Viewpoint Matching (FVM) using S^{2}GC (S^{2}GC+FVM), the performance is not as stable as in the case of S^{2}GC+JEANIE.

8.7.5 Evaluation of stride for viewing angles

The stride for viewing angles is a mere equivalent of stride parameter in CNNs, which is an equal interval location sampler. We show various sampling steps for viewing angles in Table 8.12. We notice that when stride is 15°, JEANIE performs the best on NTU-60.

8.8 More baselines on NTU-60

Table 8.13 shows more evaluations on NTU-60. Before GCNs have become mainstream backbones for the 3D Skeleton-based Action Recognition, encoding 3D body joints of

Table 8.13: Evaluations of additional baselines on NTU-60.

# Training Classes	10	20	30	40	50
Matching Nets [Vinyals et al., 2016] (skeleton to image tensor)	26.7	30.6	32.9	36.4	39.9
Proto. Net [Snell et al., 2017] (skeleton to image tensor)	30.6	33.9	36.8	40.2	43.0
Proto. Net (per block image tensor, temp. align.)	40.4	42.4	45.2	49.0	50.3
Proto. Net (per block image tensor, temp. & view. align.)	41.6	43.0	47.7	50.4	51.6

skeletons as texture-like images enjoyed some limited popularity, with approaches [Ke et al., 2017b, 2018; Tas and Koniusz, 2018] feeding such images into CNN backbones. This facilitates easy FSL with existing pipelines such as Matching Nets [Vinyals et al., 2016] and Prototypical Net [Snell et al., 2017]. Thus, we reshape the normalized 3D coordinates of each skeleton sequence or per block skeleton into image tensors, and pass them into Matching Nets [Vinyals et al., 2016] and Prototypical Net [Snell et al., 2017] for few-shot learning. Not surprisingly, using texture-like images for skeletons is suboptimal. Our JEANIE is more than 25% better.

8.9 Inference Time

Table 8.14 below compares training and inference times per query on Titan RTX 2090. For soft-DTW, each query is augmented by $K \times K' = 9$ viewpoints. In the test time, we average match distance over $K \times K' = 9$ viewpoints of each test query (this is a popular standard test augmentation strategy) w.r.t. support samples. This strategy is denoted as soft-DTW$_{\text{aug}}$. We also apply the above strategy to TAP (denoted as TAP$_{\text{aug}}$). JEANIE also uses $K \times K' = 9$ viewpoints per query. We exclude the time of

Table 8.14: A comparison of training/inference time (per query) on NTU-60 (#training classes = 10).

	Training time (s)	Inference time (s)	Total inference time (s)	Acc. (%)
soft-DTW$_{\text{aug}}$	0.098	0.019	178.5	56.8
TAP$_{\text{aug}}$	0.124	0.024	225.5	57.6
JEANIE	0.099	0.020	187.0	65.0

applying viewpoint generation as skeletons can be pre-processed at once (1.6h with non-optimized CPU code) and stored for the future use. Among methods which use multiple viewpoints, JEANIE outperforms soft-DTW$_{\text{aug}}$ and TAP$_{\text{aug}}$ by 8.2% and 7.4% respectively. JEANIE outperforms ordinary soft-DTW and TAP by 11.3% and 10.8%. For soft-DTW$_{\text{aug}}$ and TAP$_{\text{aug}}$, their total training and testing were about 5× and 9× slowed compared to counterpart soft-DTW and TAP. This is expected as they had to deal with $K \times K' = 9$ more samples. We tried also parallel JEANIE. Training JEANIE$_{\text{par}}$ with 4 Titan RTX 2090 took 44h, the total inference was 48s.

8.10 Drawbacks/Limitations

Some minor drawbacks of our work are: (i) extending our work to video-based action recognition requires more work as it is hard to simulate the views of pixels for RGB videos; (ii) alignment in 4D space is slower than in 2D space and slower than no alignment at all, but improvements in accuracy are significant, and in fact the alignment step can be written as the RKHS kernel, which can be linearized by the Nyström feature maps, casting 4D problem as two 2D problems.

8.11 Conclusions

We have proposed a Few-shot Action Recognition (FSAR) approach for learning on 3D skeletons via JEANIE. We have demonstrated that the joint alignment of temporal blocks and simulated viewpoints of skeletons between support-query sequences is efficient in the meta-learning setting where the alignment has to be performed on new action classes under the low number of samples. Our experiments have shown that using the stereo camera geometry is more efficient than simply generating multiple views by Euler angles in the meta-learning regime. Most importantly, we have introduced a novel FSAR approach that learns on articulated 3D body joints.

Chapter 9

Summary and Future Work

9.1 Summary and Contributions

This thesis tackles five significant challenges in action recognition: (i) Geometric Distortions: These include variations in camera viewpoints, rotations, *etc*. (ii) Photometric Distortions: Such as scene clutter, occlusions, self-occlusion, lighting conditions, camera noise, and blur. (iii) Speeds of Action Execution: Addressing the dynamics of actions, which can vary significantly. (iv) Temporal Dependencies: Both long-term and short-term temporal relationships between actions. (v) Cross-Subject Variations: Accounting for differences in human body sizes, which affect robust human action modeling. Moreover, we explore five related research directions that intersect with these challenges: (i) Video-Based *vs*. Skeleton-Based Features: Comparing the effectiveness of video-based and skeleton-based feature representations. (ii) Action Classification/Recognition *vs*. Few-Shot Learning: Investigating conventional action classification/recognition versus few-shot learning between pairs of action sequences. (iii) Self-Supervised *vs*. Supervised *vs*. Unsupervised Action Recognition: Assessing different learning paradigms for action recognition. (iv) Geometric and Photometric Distortion Issues in 3D Action Recognition: Focusing on challenges specific to 3D action recognition. (v) Uni-Modal *vs*. Multi-Modal Problem in Action Recognition: Examining the use of single modality versus multiple modalities in action recognition.

We evaluate the performance of ten state-of-the-art action recognition algorithms on six benchmark datasets, with a particular focus on comparing handcrafted features versus deep learning features and skeleton-based features versus depth-based features. We also explore the impact of cross-view versus cross-subject performance on human action recognition, considering whether features used are depth-based, skeleton-based, or depth+skeleton-based. Our findings reveal valuable insights: (i) Many methods perform better on cross-subject action recognition than cross-view action recognition. (ii) Skeleton-based features demonstrate robustness for both cross-subject and cross-view action recognition. (iii) Handcrafted *vs*. Deep Learning Features: Handcrafted features excel with smaller datasets, while deep learning features shine when trained on larger datasets. (iv) Computational Costs: Deep learning requires substantial computational resources primarily for training, while handcrafted features are more efficient during inference. (v) Memory Requirements: Deep learning models

often demand more memory for storage and feature extraction, whereas handcrafted features are memory-efficient. The choice between handcrafted and deep learning features depends on specific requirements, resources, and priorities. Handcrafted features offer resource efficiency; however, the pipeline typically necessitates multiple distinct computational steps, scripting, and disk storage for intermediate data. On the other hand, deep learning features provide higher accuracy, especially with larger datasets. Our comprehensive review also highlights the evolution of action recognition approaches, including the shift from handcrafted to deep learning features and the development of robust skeleton-based representations. Researchers have increasingly explored end-to-end learning with large-scale datasets and powerful GPUs.

In summary, this thesis contributes valuable solutions in the domains of video-based, skeleton-based, and few-shot skeletal action recognition.

9.1.1 Video-based Action Recognition

Our research draws inspiration from self-supervised learning (SSL) to acquire valuable representations that can benefit downstream tasks like action recognition. Our approach adopts a multi-task learning (MTL) framework, well-known for enhancing generalization and mitigating overfitting in modern action recognition models. This MTL strategy leverages self-supervisory signals, such as Improved Dense Trajectories (IDT) encoded using Bag of Words (BoW) and Fisher Vectors (FV) from videos. In this context, our work simplifies modern video-based action recognition by investigating feature representations that can co-regularize and self-supervise a backbone network. The goal is to learn how to synthesize or hallucinate expensive representations during training and then utilize the outputs of these hallucination streams at testing time.

Our approach involves two key aspects: (i) We propose that traditional IDT-based BoW/FV global video descriptors and I3D optical flow features (OFF) can be learned through dedicated CNN streams during training. These learned features can then be synthesized for classification using a CNN action recognition pipeline (*e.g.*, the I3D RGB stream) at testing time. Our observations indicate that even OFF, which is typically encoded by a separate network, can be learned by another network trained on RGB frames only. This suggests redundancy in training both RGB and optical flow network streams. We find that OFF and IDT-based BoW/FV descriptors are highly complementary, and their late fusion significantly improves performance and robustness. (ii) We introduce an alternative approach in which we learn a representation capable of approximating various descriptors through distinct projection heads. After training, we use this representation while discarding the hallucinated descriptor. This variant, while slightly decreasing performance by 1-2% on average, highlights the effectiveness of our design in leveraging hallucinated features to enhance performance.

Additionally, we propose the utilization of object and saliency detectors to enhance action recognition pipelines. We create two compact descriptors known as Object Detection Features (ODF) and Saliency Detection Features (SDF). These descriptors are statistically motivated high-order representations. Our findings indicate that object

and saliency detection features complement each other effectively and contribute to further improving action recognition performance. We also note that the effectiveness of our model depends on the learning capability of the backbone network, with AssembleNet and AssembleNet++ outperforming I3D in feature hallucination and action classification tasks. Furthermore, higher-order object/saliency detection features prove valuable for fine-grained action recognition tasks due to their rich visual information.

While our model simplifies architecture and achieves better action classification results, particularly in fine-grained action recognition, we acknowledge the computational costs during training. Preparing IDT-based BoW/FV, object/saliency detection features, and extracting optical flow features from I3D optical flow streams for self-supervisory signals can be resource-intensive. Moreover, training our hallucination streams atop backbone networks consumes additional time due to increased learning parameters. To address this, we suggest exploring lightweight action recognition models, such as SqueezeNet [Iandola et al., 2016], Xception [Chollet, 2017], ShuffleNet [Ma et al., 2018], EfficientNet [Tan and Le, 2019], MobileNet [Howard et al., 2019], the FASTER framework [Zhu et al., 2019] and Video Transformer Network (VTN) [Kozlov et al., 2020], which can be employed for both hallucination and classification tasks. These models can help mitigate computational demands while maintaining robust performance.

9.1.2 Skeleton-based Action Recognition

Unlike video-based methods, which primarily concentrate on modeling spatio-temporal representations, skeleton sequences have proven to be robust against sensor noises, computationally efficient, and storage-friendly. These sequences represent the spatio-temporal evolution of 3D body joints, and they are highly effective in action recognition. Human actions are characterized by interaction groups of skeletal joints, such as wrist-only, head-wrist, or head-wrist-ankles, and the influence of these joint groups on each action can vary significantly. Hence, it's crucial to design a better model for skeleton data, especially considering the suboptimal topology of the skeleton graph.

In our work, we introduce two essential components: the sequence compatibility kernel (SCK) and the dynamics compatibility kernel (DCK). These kernels capture the spatio-temporal evolution of 3D skeleton body joints. We extend these kernels to aggregate over multiple subsequences and CNN classifier scores for 3D action recognition. Our findings reveal that SCK captures higher-order correlations between 3D coordinates of body joints and their temporal variations, while DCK captures action dynamics by modeling the spatio-temporal co-occurrences of body joints. Furthermore, skeleton subsequences of varying lengths contain rich long-term and short-term temporal motion dependencies that significantly benefit classifying similar actions or actions with similar motion trajectories. Kernel-based tensor representations help capture higher-order relationships between spatial features and temporal dynamics, contributing to fine-grained action recognition tasks.

To further enhance our model, we adopt the concept of a hypergraph, representing 3D body joints as nodes of different orders. We introduce a novel Multi-order Multi-mode Transformer (3Mformer) consisting of Multi-order Pooling (MP) and Temporal block Pooling (TP) modules. This approach aims to create coupled-mode tokens, such as 'channel-temporal block' and 'order-channel-body joint', and perform weighted hyper-edge aggregation and temporal block aggregation. We observe that modeling larger groups of 3D body joints as hyper-edges can capture complex spatio-temporal motion dynamics effectively. The 3Mformer proves to be efficient in fusing higher-order feature representations, and coupled-mode tokens contribute to improved action recognition due to the diverse focus of each attention mechanism.

While our model demonstrates impressive performance, it has some limitations. The number of parameters and computational costs are higher than existing models, mainly due to the incorporation of r branches of the Higher-order Transformer (HoT). Furthermore, the HoT block focuses solely on encoding temporal block features of skeleton subsequences. A more efficient approach might involve redesigning the HoT block to encode both short-term and long-term spatio-temporal features, potentially simplifying the backbone encoder. Additionally, these models require substantial training on large-scale datasets like Kinetics-Skeleton, making them less suitable for scenarios where collecting and labeling videos for 3D skeleton sequences are labor-intensive and require retraining or fine-tuning for new class concepts.

Finally, when comparing our higher-order tensor representations to 3Mformer, we find that 3Mformer outperforms tensor representations on the NTU-60 dataset, with improvements of 3.24% and 3.95% for cross-subject and cross-view evaluation protocols, respectively. It's worth noting that 3Mformer is an end-to-end pipeline that incorporates various orders of feature representations, whereas SCK$\oplus$ is not an end-to-end pipeline and employs a higher-order pooling strategy for aggregation.

9.1.3 Few-shot Skeletal Action Recognition

There is a growing interest in developing effective Few-shot Learning (FSL) techniques for action recognition, often referred to as Few-shot Action Recognition (FSAR). FSAR aims to quickly adapt to new classes with minimal training samples, but it remains relatively scarce for video data due to the voluminous nature of videos and the significant intra-class variations they exhibit. Many methods recognize the importance of temporal alignment in handling nonlinear temporal variations, leading to the development of various alignment-based models for comparing sequence pairs.

Our work specifically focuses on advancing few-shot recognition within the context of articulated sets of connected 3D body joints. We have introduced a novel method called uncertainty-DTW, or uDTW, which considers the uncertainty associated with frame-wise (or block-wise) features when selecting the optimal path using Maximum Likelihood Estimation (MLE). We employ parameters, such as variance, from a distribution (typically the Normal distribution), within MLE (and uDTW) to model uncertainty. We present several applications of uDTW, including forecasting time series evolution, estimating the Fréchet mean of time series, supervised few-shot

action recognition, and unsupervised few-shot action recognition. Our observations indicate that uDTW, compared to soft-DTW, generates more potential routes for matching sequence pairs due to its uncertainty modeling capability, enhancing path warping in soft-DTW.

In addition, we propose a FSAR approach named Joint tEmporal and cAmera viewpoiNt alIgnmEnt (JEANIE), which aligns temporal blocks and simulates viewpoint indexes of skeletons between support and query sequences. This alignment ensures the selection of smooth paths without abrupt jumps in temporal locations and view indexes, aiding meta-learning with limited samples of novel classes. To simulate different viewpoints of 3D skeleton sequences, we explore two strategies: (1) rotating sequences using Euler angles within a specified range along the x and y axes and (2) simulating camera viewpoints based on stereo projection principles. We also introduce a similarity-based loss to encourage within-class sequence alignment and discourage between-class sequence alignment. Extensive experiments on one-shot action recognition and few-shot multiview classification tasks demonstrate our method's state-of-the-art performance on various benchmarks, including the large-scale Kinetics-skeleton dataset. Importantly, our uDTW and JEANIE methods can be viewed as metric-learning-inspired FSL approaches and can be extended to non-episodic problems.

Our findings highlight that using camera viewpoint simulation in JEANIE yields improved performance compared to using Euler angles, and aligning both temporal and camera viewpoints contributes to robust multiview classification. However, there are some minor limitations to our work: (1) Extending our approach to video-based action recognition is challenging due to the difficulty of simulating pixel views for RGB videos, and (2) alignment in 4D space is computationally slower than in 2D space and slower than no alignment at all, but the gains in accuracy are substantial. Additionally, the alignment step can be expressed as an RKHS kernel, which can be linearized using Nyström feature maps, effectively transforming the 4D problem into two 2D problems, addressing some computational challenges.

9.2 Future Work

While existing research has yielded promising results in action recognition, there is an ongoing demand for the development of new and more robust algorithms. These algorithms are needed to meet the pressing requirements for utilizing action recognition in various real-world and novel environments. In the following sections, we outline our short-term and long-term goals for future work in this domain.

Our short-term goal is to enhance existing methods by introducing more efficient action recognition models. For video-based action recognition, we intend to employ lightweight models to generate valuable descriptors efficiently. Additionally, we plan to explore feature descriptors derived from various sources such as audio, skeleton sequences, depth videos, point clouds, and more. In the context of skeleton-based action recognition, we aim to apply contrastive learning techniques using

a novel transformer or higher-order transformer for few-shot action recognition. Furthermore, we will investigate uncertainty-driven kernel tensor learning for action recognition with skeleton sequences. We also plan to extend the application of JEANIE to unsupervised few-shot skeletal action recognition and explore the use of our uncertainty-DTW method for anomaly detection in videos.

Furthermore, we intend to integrate these proposed methods into a unified framework with the following key components: (i) organizing data into a unified dataset that combines video and skeleton information for each action sequence, (ii) extracting video-based feature representations from RGB frames or optical flow sequences using deep learning models, extracting skeleton-based features from 3D joint coordinates or skeleton graphs using graph-based or hypergraph-based models or transformers, and combining these features into a unified representation through late fusion or attention-based fusion, and (iii) implementing a few-shot learning module that takes these unified feature representations and learns to recognize actions with limited examples. This unified framework incorporates modules for both video-based and skeleton-based action recognition and introduces a few-shot learning component that can adapt to new actions. Shared layers within the framework extract common features from video and skeleton data, allowing the model to leverage complementary information effectively.

In contrast to most existing approaches in human action recognition, which adopt model-centric solutions, our long-term goal is to explore data-centric solutions for action recognition in videos. In this context, we will address issues related to data quality, reliability, errors, and inconsistencies in videos and skeleton sequences. Our proposed solutions include: (i) revisiting the training set and modeling temporal uncertainty for video frames (or their feature representations), (ii) spatio-temporal uncertainty modeling for skeleton sequences (or frame-wise feature representations), and (iii) temporal-viewpoint uncertainty modeling for frame-wise (or video block-wise feature representations). For few-shot data-centric action recognition, we plan to explore this domain by considering high-quality and reliable human action videos/skeleton sequences (or the uncertainty-modeled versions) to model the alignment between pairs of video frames (or skeleton sequences) in both temporal and viewpoint spaces. This approach aims to achieve robust human action modeling, ultimately leading to faster and more reliable action recognition and anomaly detection models. The proposed solutions for few-shot data-centric action recognition include: (i) video data augmentation for domain gap alignment through few-shot learning, (ii) data-agnostic meta-learning, and (iii) finding a balance between data-centric and model-centric approaches in few-shot action recognition and anomaly detection tasks. For data-centric action recognition on edge devices, we will investigate data imbalance issues and explore auto-evaluation methods for action recognition and anomaly detection on mobile devices. The proposed solutions for this aspect include: (i) designing training data sampling strategies using statistical learning methods to form reliable training batch data and (ii) implementing auto-evaluation through pseudo-labeling.

Appendices

Appendix A

Datasets and their statistics

Table A.1: UCR archive (the latest version from 2018) which we use for time series analysis. The information is grouped based on the type of time series.

Dataset type	Avg. #train	Avg. #test	Total #classes	Avg. length
Device	1261	1135	44	895
ECG	708	1755	95	326
EOG	362	362	24	1250
EPG	40	249	6	601
Hemodynamics	104	208	156	2000
HRM	18	186	18	201
Image	595	1157	334	360
Motion	347	1057	99	517
Power	180	180	2	144
Sensor	420	1286	177	410
Simulated	203	1021	32	267
Spectro	179	147	24	553
Spectrum	305	388	17	1836
Traffic	607	1391	12	24
Trajectory	208	130	78	360

Table A.2: Popular benchmark datasets which we use for few-shot action recognition.

Datasets	Year	Classes	Subjects	#views	#clips	Sensor	#joints
MSR Action 3D	2010	20	10	1	567	Kinect v1	20
3D Action Pairs	2013	12	10	1	360	Kinect v1	20
UWA 3D Activity	2014	30	10	1	701	Kinect v1	15
NTU RGB+D	2016	60	40	80	56,880	Kinect v2	25
NTU RGB+D 120	2019	120	106	155	114,480	Kinect v2	25
Kinetics-skeleton	2018	400	-	-	$\sim$ 300,000	-	18

The UCR time series archive [Dau et al., 2018]. UCR, introduced in 2002, is an important resource in the time series analysis community with at least 1,000 published papers making use of at least 1 dataset from this archive. We use 128 datasets from the latest version of UCR from 2018, encompassing a wide variety of fields and lengths. Table A.1 details the statistics of this archive by grouping the whole dataset into different types.

Few-shot action recognition datasets. Table A.2 contains statistics of datasets used in our experiments. Smaller datasets listed below are used for more evaluations of supervised and unsupervised few-shot action recognition:

- *MSR Action 3D* [Li et al., 2010] is an older AR dataset captured with the Kinect depth camera. It contains 20 human sport-related activities such as *jogging*, *golf swing* and *side boxing*.
- *3D Action Pairs* [Oreifej and Liu, 2013] contains 6 selected pairs of actions that have very similar motion trajectories, *e.g.*, *put on a hat* and *take off a hat*, *pick up a box* and *put down a box*, *etc*.
- *UWA 3D Activity* [Rahmani et al., 2014b] has 30 actions performed by 10 people of various height at different speeds in cluttered scenes.

As MSR Action 3D, 3D Action Pairs, and UWA 3D Activity have not been used in FSAR, we created 10 training/testing splits, by choosing half of class concepts for training, and half for testing per split per dataset. Training splits were further subdivided for crossvalidation. Section B.1 details the class concepts per split for small datasets.

Appendix B

Evaluation Protocols

Below, we detail our new/additional evaluation protocols used in the experiments on Few-shot Action Recognition (FSAR).

B.1 Few-shot action recognition protocols (the small-scale datasets)

As we use several class-wise splits for small datasets, these splits will be simply released in our code. Below, we explain the selection process that we used.
FSAR (MSR Action 3D). As this dataset contains 20 action classes, we randomly choose 10 action classes for training and the rest 10 for testing. We repeat this sampling process 10 times to form in total 10 train/test splits. For each split, we have 5-way and 10-way experimental settings. The overall performance on this dataset is computed by averaging the performance over the 10 splits.
FSAR (3D Action Pairs). This dataset has in total 6 action pairs (12 action classes), each pair of action has very similar motion trajectories, *e.g.*, *pick up a box* and *put down a box*. We randomly choose 3 action pairs to form a training set (6 action classes) and the half action pairs for the test set, and in total there are $\binom{n}{k} = \binom{6}{3} = 20$ different combinations of train/test splits. As our train/test splits are based on action pairs, we are able to test whether the algorithm is able to classify unseen action pairs that share similar motion trajectories. We use 5-way protocol on this dataset to evaluate the performance of FSAR, averaged over all 20 splits.
FSAR (UWA 3D Activity). This dataset has 30 action classes. We randomly choose 15 action classes for training and the rest half action classes for testing. We form in total 10 train/test splits, and we use 5-way and 10-way protocols on this dataset, averaged over all 10 splits.

B.2 One-shot protocol on NTU-60

Following NTU-120 [Liu et al., 2019a], we introduce the one-shot AR setting on NTU-60. We split the whole dataset into two parts: auxiliary set (on NTU-120 the training set is called as auxiliary set, so we follow such a terminology) and one-shot evaluation set.

Auxiliary set contains 50 classes, and all samples of these classes can be used for learning and validation. Evaluation set consists of 10 novel classes, and one sample from each novel class is picked as the exemplar (terminology introduced by authors of NTU-120), while all the remaining samples of these classes are used to test the recognition performance.
Evaluation set contains 10 novel classes, namely A1, A7, A13, A19, A25, A31, A37, A43, A49, A55.

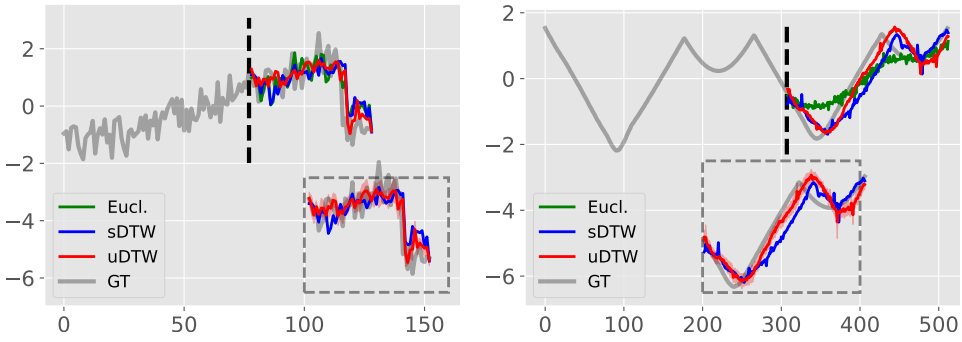
The following 10 samples are the exemplars:
(01)S001C003P008R001A001, (02)S001C003P008R001A007,
(03)S001C003P008R001A013, (04)S001C003P008R001A019,
(05)S001C003P008R001A025, (06)S001C003P008R001A031,
(07)S001C003P008R001A037, (08)S001C003P008R001A043,
(09)S001C003P008R001A049, (10)S001C003P008R001A055.
Auxiliary set contains 50 classes (the remaining 50 classes of NTU-60 excluding the 10 classes in evaluation set).

B.3 Few-shot multiview classification on NTU-120

Horizontal camera view. As NTU-120 is captured by 3 cameras (from 3 different horizontal angles: -45°, 0°, 45°), we split the whole dataset based on the camera ID to form our 3 horizontal camera viewpoints (left, center and right views). We then evaluate few-shot multiview classification using (i) the left view for training and the center view for testing (ii) the left view for training and the right view for testing (ii) the left and center views for training and the right view for testing.
Vertical camera view. Based on the table provided in [Liu et al., 2019a], we first group 32 camera setups into 3 groups by dividing the range of heights into 3 equally-sized ranges to form roughly the top, center and bottom views. We then group the whole dataset into 3 camera viewpoints based on the camera setup IDs. For few-shot multiview classification, we evaluate our proposed method using (i) bottom view for training and center view for testing (ii) bottom view for training and top view for testing (iii) bottom and center views for training and top view for testing.

Appendix C

Visualizations of Forecasting the Evolution of Time Series



(a) CBF (b) ShapesAll

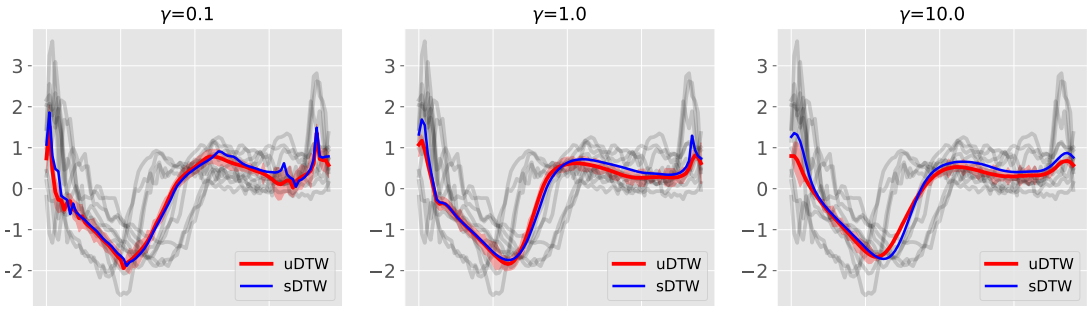
Figure C.1: Additional visualizations for forecasting the evolution of time series. Given the first part of a time series, we train the pipeline from Fig. 7.3b to predict the remaining part of the time series. We compare the use of the Euclidean, sDTW or uDTW distances within the pipeline. We use CBF and ShapesAll in UCR archive, and display the prediction obtained for the given test sample with either of these 3 distances, and the ground truth (GT). Oftentimes, we observe that uDTW helps predict the sudden changes well. (a) Our uDTW aligns well with the ground truth compared to sDTW. (b) Our uDTW generates better shape of prediction compared to sDTW (for example note the red curve following much closer the rising gray slope). Quantitative results of MSE and 'shape' metrics for the whole UCR archive are given in chapter 7.

We provide additional visualizations of forecasting the evolution of time series in Figure C.1. We notice that our uDTW produces predictions that are better aligned with the ground truth (see Fig. C.1a). Moreover, our uDTW generates better shape of the predictions compared to sDTW, and the predictions from sDTW have more perturbations/fluctuations (see Fig. C.1b). Quantitative results for the whole UCR archive can be found in chapter 7.

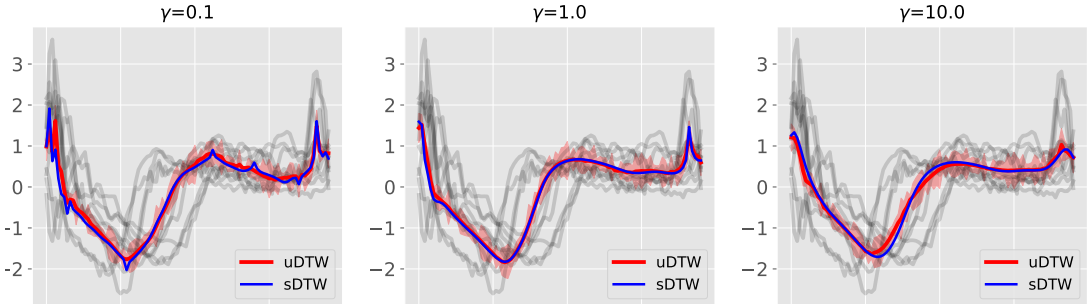
Appendix D

Visualizations on Barycenters

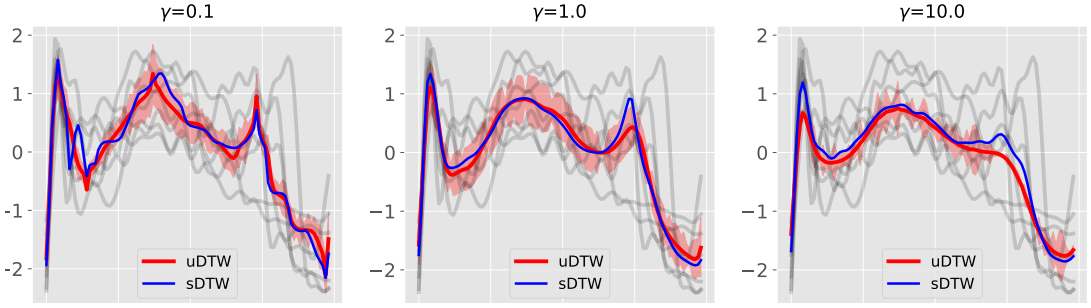
Figure D.1 shows more visualizations of barycenters of time series. With our SigmaNet, we obtain much better barycenters with our uDTW compared to sDTW.



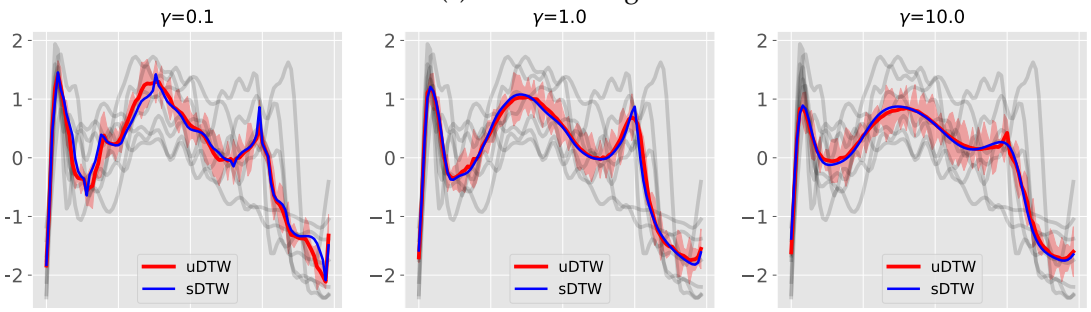
(a) ECG200



(b) ECG200 (our SigmaNet)



(c) Medical Images



(d) Medical Images (our SigmaNet)

Figure D.1: Comparison of barycenters based on our uDTW *vs*. sDTW. We visualize uncertainty around the barycenters in red color for uDTW. Our uDTW with SigmaNet generates reasonable barycenters even when higher γ values are used, *e.g.*, $\gamma = 10.0$. Higher γ value leads to smooth barycenters but introducing higher uncertainty.

Appendix E

Visualizations on JEANIE and FVM

To explain what makes JEANIE perform well on the task of comparing pairs of sequences, we perform additional visualisations.

To this end, we choose skeleton sequences from UWA3D Multiview Activity II for experiments and visualizations of FVM and JEANIE. UWA3D Multiview Activity II contains rich viewpoint configurations and so is perfect for our investigations.

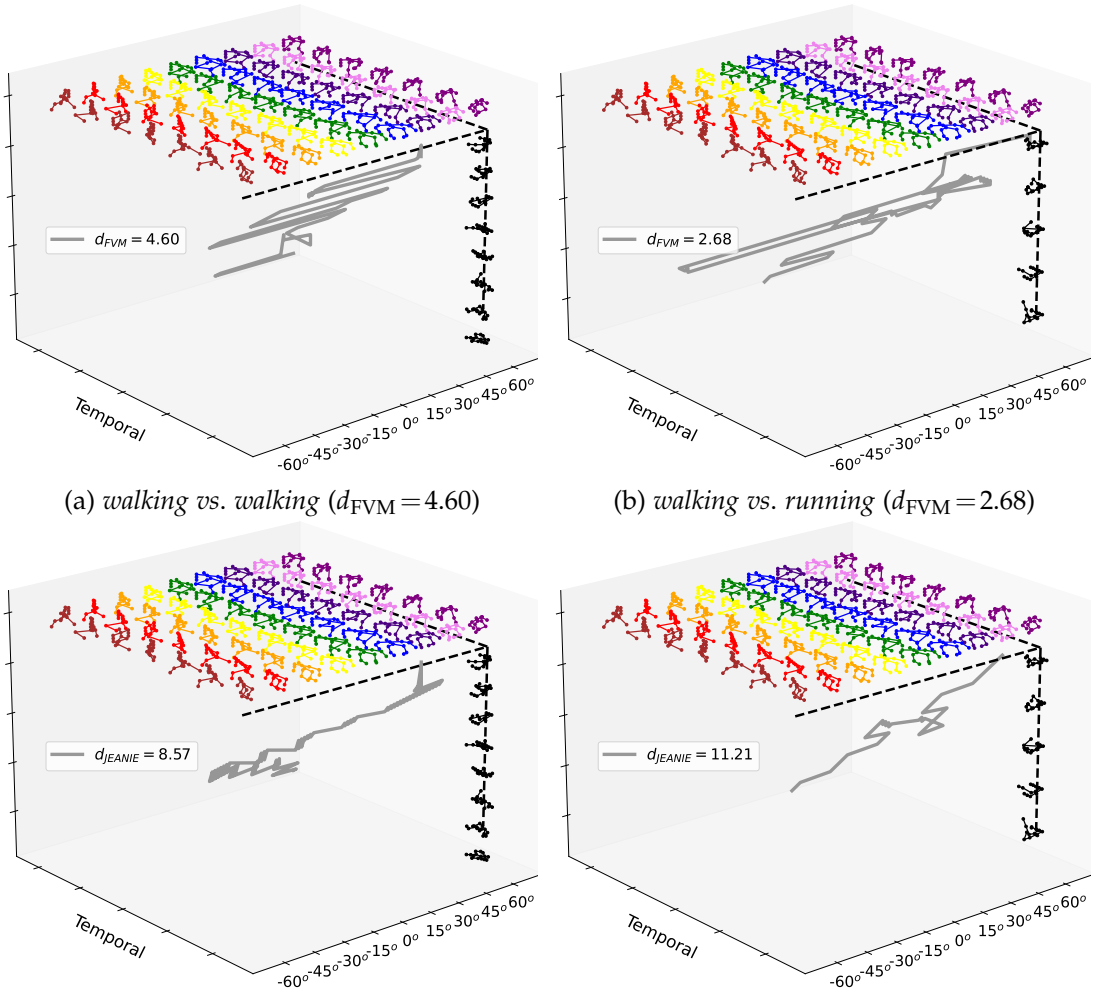
1. Matching similar actions. We choose a *walking* skeleton sequence (‘`a12_s01_e01_v01`’) as the query sample with more viewing angles for the camera viewpoint simulation, and we select another *walking* skeleton sequence of a different view (‘`a12_s01_e01_v03`’) and a *running* skeleton sequence (‘`a20_s01_e01_v02`’) as support samples respectively, to verify that our JEANIE is able to find the better matching distances compared to FVM.

2. Matching actions with similar motion trajectories. We choose a *two hand punching* skeleton sequence (‘`a04_s01_e01_v01`’) as the query sample with more viewing angles for the camera viewpoint simulation, and we select another *two hand punching* skeleton sequence of a different view (‘`a04_s05_e01_v02`’) and a *holding head* skeleton sequence (‘`a10_s05_e01_v02`’) as support samples respectively, to verify that our JEANIE is able to find the better matching distances compared to FVM.

Fig. E.1 shows the visualizations. Comparing Fig. E.1a and E.1b of FVM, we notice that for skeleton sequences from different action classes (*walking vs.running*), FVM finds the path with a very small distance $d_{\text{FVM}} = 2.68$. In contrast, for sequences from the same action class (*walking vs.walking*), FVM gives $d_{\text{FVM}} = 4.60$ which higher than in case of within-class sequences. This is an undesired effect which may result in wrong comparison decision.

In contrast, in Fig E.1c and E.1d, our JEANIE gives $d_{\text{JEANIE}} = 8.57$ for sequences of the same action class and $d_{\text{JEANIE}} = 11.21$ for sequences from different action classes, which means that the within-class distances are smaller than between-class distances. This is in fact a very important property when comparing pairs of sequences.

Fig E.2 shows additional visualizations. Again, our JEANIE produces more reasonable matching distances than FVM.



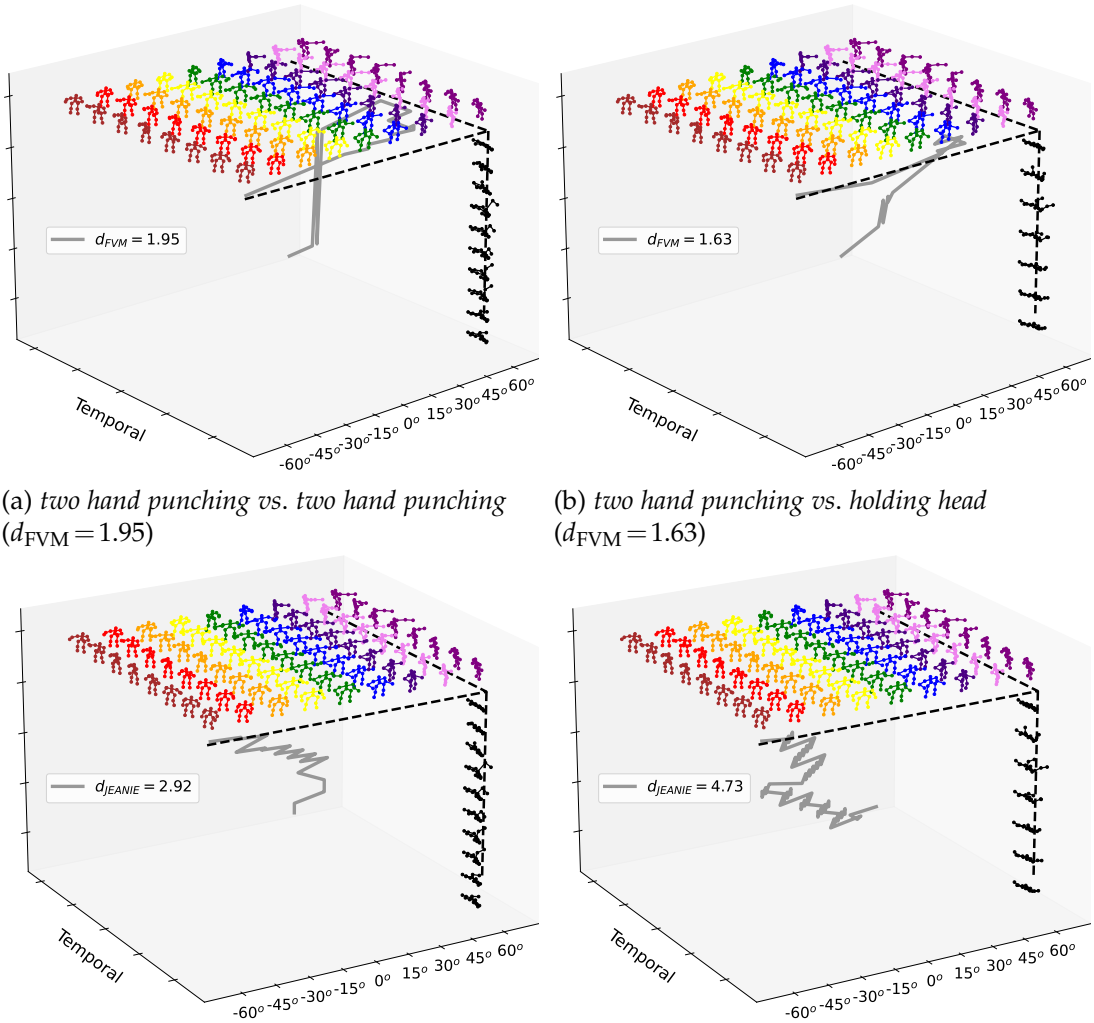
(a) *walking vs. walking* ($d_{\text{FVM}} = 4.60$)

(b) *walking vs. running* ($d_{\text{FVM}} = 2.68$)

(c) *walking vs. walking* ($d_{\text{JEANIE}} = 8.57$)

(d) *walking vs. running* ($d_{\text{JEANIE}} = 11.21$)

Figure E.1: Visualization of FVM and JEANIE for *walking vs. walking* (two different sequences) and *walking vs. running*. From UWA3D Multiview Activity II, we choose a *walking* sequence as the query sample ('`a12_s01_e01_v01`'). We choose another *walking* sequence from a different view ('`a12_s01_e01_v03`') and a *running* sequence ('`a20_s01_e01_v02`') as the support samples respectively. We notice that for two different action sequences in (b), the greedy FVM finds the path with a very small distance $d_{\text{FVM}} = 2.68$ but for sequences of the same action class, FVM gives $d_{\text{FVM}} = 4.60$. This is clearly suboptimal as the within-class distance is higher then the between-class distance (to counteract this issue, we have proposed JEANIE). Our JEANIE is able to produce a smaller distance for within-class sequences in (c) and a larger distance for between-class sequences in (d), which is a very important property when comparing pairs of sequences.



(a) *two hand punching vs. two hand punching* ($d_{\text{FVM}} = 1.95$)

(b) *two hand punching vs. holding head* ($d_{\text{FVM}} = 1.63$)

(c) *two hand punching vs. two hand punching* ($d_{\text{JEANIE}} = 2.92$)

(d) *two hand punching vs. holding head* ($d_{\text{JEANIE}} = 4.73$)

Figure E.2: Visualization of FVM and JEANIE for *two hand punching vs. two hand punching* (two different sequences) and *two hand punching vs. holding head*. From UWA3D Multiview Activity II, we choose a *two hand punching* sequence as the query sample ('`a04_s01_e01_v01`'), and another *two hand punching* sequence from a different view ('`a04_s05_e01_v02`') and a *holding head* sequence ('`a10_s05_e01_v02`') as the support samples respectively. We notice that for two different action sequences in (b), the greedy FVM finds the path which results in $d_{\text{FVM}} = 1.63$ for sequences of different action classes, yet FVM gives $d_{\text{FVM}} = 1.95$ for two sequences of the same class. The within-class distance should be smaller than the between-class distance but greedy approaches such as FVM cannot handle this requirement well. Our JEANIE gives smaller distance when comparing within-class sequences compared to between-class sequences. This is a very important property when comparing pairs of sequences.

Bibliography

Euler angles. Wikipedia, https://en.wikipedia.org/wiki/Euler_angles. Accessed: 24-03-2023. (cited on pages 164 and 167)

Lecture 12: Camera projection. On-line, http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf. Accessed: 24-03-2023. (cited on pages 164 and 167)

Abid, A. and Zou, J., 2018. Autowarp: Learning a warping distance from unlabeled time series using sequence autoencoders. NIPS'18. Curran Associates Inc., Red Hook, NY, USA. (cited on page 146)

Ahn, D.; Kim, S.; Hong, H.; and Ko, B. C., 2023. Star-transformer: A spatio-temporal cross attention transformer for human action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3330–3339. (cited on page 1)

Akbari, H.; Yuan, L.; Qian, R.; Chuang, W.-H.; Chang, S.-F.; Cui, Y.; and Gong, B., 2021. VATT: Transformers for multimodal self-supervised learning from raw video, audio and text. In *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=RzYrn625bu8. (cited on pages 10 and 122)

Alvarez, L.; Weickert, J.; and Sánchez, J., 2000. Reliable estimation of dense optical flow fields with large displacements. *IJCV*, 39, 1 (Aug. 2000), 41–56. doi: 10.1023/A:1008170101536. https://doi.org/10.1023/A:1008170101536. (cited on page 47)

Amor, B. B.; Su, J.; and Srivastava, A., 2016. Action Recognition Using Rate-Invariant Analysis of Skeletal Shape Trajectories. *TPAMI*, 38, 1 (2016), 1–13. (cited on pages 22 and 23)

Arnab, A.; Dehghani, M.; Heigold, G.; Sun, C.; Lučić, M.; and Schmid, C., 2021. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6836–6846. (cited on page 7)

Baradel, F.; Neverova, N.; Wolf, C.; Mille, J.; and Mori, G., 2018. Object level visual reasoning in videos. In *ECCV*, 1–16. Springer Science+Business Media, Munich, Germany. (cited on page 75)

Bart, E. and Ullman, S., 2005. Cross-generalization: Learning novel classes from a single example by feature replacement. *CVPR*, (2005), 672–679. (cited on pages 12 and 164)

Bashir, F. I.; Khokhar, A. A.; and Schonfeld, D., 2006. View-invariant motion trajectory-based activity classification and recognition. *Multimedia Systems*, 12 (2006), 45–54. (cited on pages 10 and 11)

Ben-Ari, R.; Shpigel Nacson, M.; Azulai, O.; Barzelay, U.; and Rotman, D., 2021. Taen: Temporal aware embedding network for few-shot action recognition. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2780–2788. (cited on pages 15 and 146)

Bergstra, J.; Komer, B.; Eliasmith, C.; Yamins, D.; and Cox, D. D., 2015. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8, 1 (2015), 014008. http://stacks.iop.org/1749-4699/8/i=1/a=014008. (cited on pages 155 and 177)

Bertasius, G.; Wang, H.; and Torresani, L., 2021. Is space-time attention all you need for video understanding? In *ICML*, vol. 2, 4. (cited on page 7)

Berti, S.; Rosasco, A.; Colledanchise, M.; and Natale, L., 2022. One-shot open-set skeleton-based action recognition. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 765–772. IEEE. (cited on page 15)

Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; and Gould, S., 2016. Dynamic Image Networks for Action Recognition. *CVPR*, (2016), 3034–3042. (cited on page 21)

Bishay, M.; Zoumpourlis, G.; and Patras, I., 2019. TARN: temporal attentive relation network for few-shot and zero-shot action recognition. *CoRR*, abs/1907.09021 (2019). http://arxiv.org/abs/1907.09021. (cited on page 13)

Blank, M.; Gorelick, L.; Shechtman, E.; Irani, M.; and Basri, R., 2005. Actions as Space-Time Shapes. *ICCV*, (2005), 1–8. (cited on pages 19 and 21)

Blondel, M.; Mensch, A.; and Vert, J.-P., 2021. Differentiable divergences between time series. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, vol. 130 of *Proceedings of Machine Learning Research*, 3853–3861. PMLR. (cited on pages 145, 153, 154, and 161)

Bo, L.; Lai, K.; Ren, X.; and Fox, D., 2011. Object recognition with hierarchical kernel descriptors. *CVPR*, (2011). (cited on page 88)

Bo, Y.; Lu, Y.; and He, W., 2020. Few-shot learning of video action recognition only based on video contents. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 584–593. doi:10.1109/WACV45572.2020.9093481. (cited on page 13)

Bobick, A. F. and Davis, J. W., 2001. The Recognition of Human Movement Using Temporal Templates. *TPAMI*, 23, 3 (3 2001), 257–267. (cited on pages 19 and 21)

Borji, A.; Cheng, M.-M.; Jiang, H.; and Li, J., 2015. Salient object detection: A benchmark. *TIP*, 24, 12 (2015), 5706–5722. doi:10.1109/TIP.2015.2487833. (cited on page 68)

Braux-Zin, J.; Dupont, R.; and Bartoli, A., 2013. A general dense image matching framework combining direct and feature-based costs. In *ICCV*, 185–192. (cited on pages 47 and 67)

Bregonzio, M.; Gong, S.; and Xiang, T., 2009. Recognising Actions as Clouds of Space-Time Interest Points. *CVPR*, (2009), 1948–1955. (cited on pages 19 and 21)

Broomé, S.; Pokropek, E.; Li, B.; and Kjellström, H., 2023. Recur, attend or convolve? on whether temporal modeling matters for cross-domain robustness in action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 4199–4209. (cited on page 1)

Brox, T. and Malik, J., 2011. Large displacement optical flow: Descriptor matching in variational motion estimation. *TPAMI*, 33, 3 (Mar. 2011), 500–513. doi:10.1109/TPAMI.2010.143. http://dx.doi.org/10.1109/TPAMI.2010.143. (cited on pages 47, 67, and 104)

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y., 2014. Spectral Networks and Deep Locally Connected Networks on Graphs. In *ICLR*, 1–14. (cited on page 26)

Buettner, M.; Prasad, R.; Philipose, M.; and Wetherall, D., 2009. Recognizing Daily Activities with RFID-Based Sensors. *UbiComp*, (2009). (cited on page 21)

Caba Heilbron, F.; Escorcia, V.; Ghanem, B.; and Carlos Niebles, J., 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*. (cited on page 10)

Cao, K.; Ji, J.; Cao, Z.; Chang, C.-Y.; and Niebles, J. C., 2020. Few-shot video classification via temporal alignment. In *CVPR*. (cited on pages 12, 13, 15, 145, 146, 164, and 166)

Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y., 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*. (cited on pages xxix, 4, 7, 85, 104, 110, 112, 120, 130, and 174)

Carreira, J. and Zisserman, A., 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*. (cited on pages 4, 5, 6, 9, 10, 12, 21, 43, 44, 47, 57, 58, 63, 68, 77, 86, 102, 104, 111, and 163)

Caruana, R., 1997. Multitask learning. *Machine Learning*, 28, 1 (Jul. 1997), 41–75. doi:10.1023/A:1007379606734. https://doi.org/10.1023/A:1007379606734. (cited on pages 9, 26, and 61)

Catalin; Ionescu; Dragos; Papava; Vlad; Olaru; Cristian; and Sminchisescu, 2014. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2014). (cited on pages 150 and 174)

Cavazza, J.; Zunino, A.; Biagio, M. S.; and Vittorio, M., 2016. Kernelized covariance for action recognition. *CoRR abs/1604.06582*, (2016). (cited on page 88)

Chakraborty, B.; Holte, M. B.; Moeslund, T. B.; and Gonzàlez, J., 2012. Selective spatio-temporal interest points. *CVIU*, 116, 3 (2012), 396–410. (cited on pages 45 and 66)

Chen, C.; Fan, Q.; and Panda, R., 2021a. Crossvit: Cross-attention multi-scale vision transformer for image classification. *CoRR*, abs/2103.14899 (2021). https://arxiv.org/abs/2103.14899. (cited on page 124)

Chen, T.; Zhou, D.; Wang, J.; Wang, S.; He, Q.; Hu, C.; Ding, E.; Guan, Y.; and He, X., 2022. Part-aware prototypical graph network for one-shot skeleton-based action recognition. *arXiv preprint arXiv:2208.09150*, (2022). (cited on page 15)

Chen, W.-Y.; Liu, Y.-C.; Kira, Z.; Wang, Y.-C. F.; and Huang, J.-B., 2019. A closer look at few-shot classification. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HkxLXnAcFQ. (cited on page 13)

Chen, Y.; Zhang, Z.; Yuan, C.; Li, B.; Deng, Y.; and Hu, W., 2021b. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13359–13368. (cited on pages 8, 121, 133, and 135)

Cheng, K.; Zhang, Y.; Cao, C.; Shi, L.; Cheng, J.; and Lu, H., 2020a. Decoupling gcn with dropgraph module for skeleton-based action recognition. In *Computer Vision – ECCV 2020*, 536–553. Springer International Publishing, Cham. (cited on pages 8 and 121)

Cheng, K.; Zhang, Y.; He, X.; Chen, W.; Cheng, J.; and Lu, H., 2020b. Skeleton-based action recognition with shift graph convolutional network. In *CVPR*. (cited on pages 8, 121, 132, 133, 135, and 166)

Cheng, Y.-B.; Chen, X.; Zhang, D.; and Lin, L., 2021. Motion-transformer: Self-supervised pre-training for skeleton-based action recognition. In *Proceedings of the 2nd ACM International Conference on Multimedia in Asia*, MMAsia '20 (Virtual Event, Singapore, 2021). Association for Computing Machinery, New York, NY, USA. doi:10.1145/3444685.3446289. https://doi.org/10.1145/3444685.3446289. (cited on page 124)

Cherian, A.; Fernando, B.; Harandi, M.; and Gould, S., 2017a. Generalized Rank Pooling for Activity Recognition. *CVPR*, (2017), 3222–3231. (cited on pages 21, 55, 58, 75, 78, 86, and 111)

Cherian, A. and Gould, S., 2018. Second-order temporal pooling for action recognition. *IJCV*, (2018). (cited on page 86)

Cherian, A.; Koniusz, P.; and Gould, S., 2017b. Higher-order pooling of cnn features via kernel linearization for action recognition. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 130–138. doi:10.1109/WACV.2017.22. (cited on pages xxix, 10, 21, 44, 46, 47, 64, 65, 67, 68, 86, 111, 112, 121, and 129)

Cherian, A.; Sra, S.; Gould, S.; and Hartley, R., 2018. Non-linear temporal subspace representations for activity recognition. In *CVPR*, 2197–2206. doi:10.1109/CVPR.2018.00234. (cited on pages 10, 44, 47, 55, 58, 64, 65, 68, 75, 78, and 111)

Chéron, G.; Laptev, I.; and Schmid, C., 2015. P-CNN: Pose-based CNN Features for Action Recognition. *ICCV*, (2015). (cited on pages 86 and 89)

Chi, H.-g.; Ha, M. H.; Chi, S.; Lee, S. W.; Huang, Q.; and Ramani, K., 2022. Infogcn: Representation learning for human skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 20186–20196. (cited on pages 8, 133, and 135)

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 191)

Choromanski, K. M.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J. Q.; Mohiuddin, A.; Kaiser, L.; Belanger, D. B.; Colwell, L. J.; and Weller, A., 2021. Rethinking attention with performers. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Ua6zuk0WRH. (cited on page 126)

Choutas, V.; Weinzaepfel, P.; Revaud, J.; and Schmid, C., 2018. PoTion: Pose motion representation for action recognition. In *CVPR*, 7024–7033. (cited on pages 10, 44, 46, 64, 65, and 67)

Cormode, G. and Hadjieleftheriou, M., 2008. Finding frequent items in data streams. *Proc. VLDB Endow.*, 1, 2 (Aug. 2008), 1530–1541. doi:10.14778/1454159.1454225. http://dx.doi.org/10.14778/1454159.1454225. (cited on page 50)

Csurka, G.; Dance, C. R.; Fan, L.; Willamowski, J.; and Bray, C., 2004. Visual categorization with bags of keypoints. *ECCV Workshop*, (2004), 1–22. (cited on pages 43, 46, 48, 64, and 67)

Cuturi, M., 2011. Fast global alignment kernels. In *International Conference on Machine Learning (ICML)*. (cited on pages 15, 139, 145, 153, 161, 164, and 169)

Cuturi, M. and Blondel, M., 2017. Soft-dtw: a differentiable loss function for time-series. In *International Con- ference on Machine Learning (ICML)*. (cited on pages 15, 139, 145, 149, 150, 153, 154, 160, 161, 162, 164, and 169)

Dalal, N.; Triggs, B.; and Schmid, C., 2006. Human Detection Using Oriented Histogram of Flow and Appearance. *ECCV*, (2006), 428–441. (cited on pages 43, 44, 45, 46, 63, 66, and 67)

Damen, D.; Doughty, H.; Farinella, G. M.; Fidler, S.; Furnari, A.; Kazakos, E.; Moltisanti, D.; Munro, J.; Perrett, T.; Price, W.; and Wray, M., 2018. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 1–17. Springer Science+Business Media, Munich, Germany. (cited on pages 2, 4, 75, and 79)

Dau, H. A.; Keogh, E.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; Yanping; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; and Batista, G., 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. (cited on pages 150 and 197)

Defferrard, M.; Bresson, X.; and Vandergheynst, P., 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, 1–9. (cited on page 26)

Dempster, A.; Schmidt, D. F.; and Webb, G. I., 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21 (Virtual Event, Singapore, 2021), 248–257. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3447548.3467231. (cited on page 145)

Dollár, P.; Rabaud, V.; Cottrell, G.; and Belongie, S., 2005. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the 14th International Conference on Computer Communications and Networks*, 65–72. http://dl.acm.org/citation.cfm?id=1259587.1259830. (cited on pages 19, 21, 45, and 66)

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*. (cited on pages 47, 68, and 86)

Donahue, J.; Dieleman, S.; Binkowski, M.; Elsen, E.; and Simonyan, K., 2021. End-to-end adversarial text-to-speech. In *International Conference on Learning Representations*. (cited on page 145)

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. (cited on pages 124, 125, 147, 158, 159, 165, 170, 171, and 180)

Du, D.; Shringi, A.; Hoogs, A.; and Funk, C., 2023. Reconstructing humpty dumpty: Multi-feature graph autoencoder for open set action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3371–3380. (cited on page 1)

Du, Y.; Wang, W.; and Wang, L., 2015. Hierarchical Recurrent Neural Network for Skeleton Based Action Recognition. In *CVPR*, 1110–1118. (cited on pages 11, 22, and 23)

Duan, H.; Zhao, Y.; Chen, K.; Lin, D.; and Dai, B., 2022. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2969–2978. (cited on page 133)

Dvornik, N.; Schmid, C.; and Mairal1, J., 2020. Selecting relevant features from a multi-domain representation for few-shot classification. In *ECCV*. (cited on pages 12 and 164)

Dwivedi, S. K.; Gupta, V.; Mitra, R.; Ahmed, S.; and Jain, A., 2019. Protogan: Towards few shot learning for action recognition. *arXiv*, (2019). (cited on pages 12, 164, and 166)

Elgammal, A. and Lee, C.-S., 2009. Tracking people on a torus. *TPAMI*, (2009). (cited on page 86)

Elmadany, N. E. D.; He, Y.; and Guan, L., 2018. Information Fusion for Human Action Recognition via Biset/Multiset Globality Locality Preserving Canonical Correlation Analysis. In *TIP*, 5275–5287. (cited on pages 9, 22, 23, and 24)

Elsken, T.; Staffler, B.; Metzen, J. H.; and Hutter, F., 2020. Meta-learning of neural architectures for few-shot learning. In *CVPR*. (cited on pages 12 and 164)

Fan, H.; Xiong, B.; Mangalam, K.; Li, Y.; Yan, Z.; Malik, J.; and Feichtenhofer, C., 2021. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6824–6835. (cited on page 7)

Fang, P.; Zhou, J.; Kumar Roy, S.; Petersson, L.; and Harandi, M., 2019. Bilinear attention networks for person retrieval. In *ICCV*. (cited on pages 47 and 68)

Fei, N.; Guan, J.; Lu, Z.; and Gao, Y., 2020. Few-shot zero-shot learning: Knowledge transfer with less supervision. In *ACCV*. (cited on pages 12 and 164)

Fei-Fei, L.; Fergus, R.; and Perona, P., 2006. One-shot learning of object categories. *IEEE TPAMI*, 28, 4 (2006), 594–611. (cited on pages 12 and 164)

Feichtenhofer, C.; Fan, H.; Malik, J.; and He, K., 2019. Slowfast networks for video recognition. In *ICCV*, 6202–6211. IEEE, Seoul, Korea. (cited on pages 6 and 78)

Feichtenhofer, C.; Pinz, A.; and Wildes, R. P., 2016a. Spatiotemporal residual networks for video action recognition. In *NIPS*, 3468–3476. (cited on pages 5, 43, 47, 63, and 68)

Feichtenhofer, C.; Pinz, A.; and Wildes, R. P., 2017a. Spatiotemporal multiplier networks for video action recognition. In *CVPR*. (cited on pages 6, 21, 57, and 163)

Feichtenhofer, C.; Pinz, A.; and Wildes, R. P., 2017b. Temporal residual networks for dynamic scene recognition. In *CVPR*. (cited on pages 54, 58, 74, and 78)

Feichtenhofer, C.; Pinz, A.; and Zisserman, A., 2016b. Convolutional two-stream network fusion for video action recognition. In *CVPR*. (cited on pages 6, 21, and 163)

Fernando, B.; Gavves, E.; M., J. O.; Ghodrati, A.; and Tuytelaars, T., 2015. Modeling video evolution for action recognition. In *CVPR*, 5378–5387. (cited on pages 47 and 68)

Fernando, B.; Gavves, E.; Oramas M., J. O.; Ghodrati, A.; and Tuytelaars, T., 2017. Rank pooling for action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39, 4 (apr 2017), 773–787. doi:10.1109/TPAMI.2016.2558148. https://doi.org/10.1109/TPAMI.2016.2558148. (cited on pages 121 and 129)

Fernando, B. and Gould, S., 2016. Learning end-to-end video classification with rank-pooling. In *ICML*, vol. 48, 1187–1196. (cited on pages 10, 44, 47, 64, 65, and 68)

Fink, M., 2005. Object classification from a single example utilizing class relevance metrics. *NeurIPS*, (2005), 449–456. (cited on pages 12 and 164)

Finn, C.; Abbeel, P.; and Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, vol. 70, 1126–1135. PMLR. (cited on page 164)

Fishkin, K. P.; Philipose, M.; and Rea, A., 2005. Hands-On RFID: Wireless Wearables for Detecting Use of Objects. *IEEE ISWC*, (2005). (cited on page 21)

Freeman, W. T. and Roth, M., 1994. Orientation histograms for hand gesture recognition. Technical Report TR94-03, MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139. http://www.merl.com/publications/TR94-03/. (cited on pages 44, 46, and 67)

Gaidon, A.; Harchoui, Z.; and Schmid, C., 2011. A time series kernel for action recognition. *BMVC*, (2011), 63.1–63.11. (cited on page 88)

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T., 2016. Compact Bilinear Pooling. *CVPR*, (2016), 1–10. (cited on page 24)

Gao, Z.; Wang, P.; Lv, P.; Jiang, X.; Liu, Q.; Wang, P.; Xu, M.; and Li, W., 2022. Focal and global spatial-temporal transformer for skeleton-based action recognition. In *Proceedings of the Asian Conference on Computer Vision*, 382–398. (cited on page 8)

García-García, D.; Parrado Hernández, E.; and Díaz-de María, F., 2009. A new distance measure for model-based sequence clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31, 7 (2009), 1325–1331. doi:10.1109/TPAMI.2008.268. (cited on page 145)

Gauglitz, S.; Höllerer, T.; and Turk, M., 2011. Evaluation of interest point detectors and feature descriptors for visual tracking. *IJCV*, 94, 3 (3 2011), 335. doi:10.1007/s11263-011-0431-5. https://doi.org/10.1007/s11263-011-0431-5. (cited on page 45)

Ghadiyaram, D.; Tran, D.; and Mahajan, D., 2019. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 12046–12055. IEEE, Long Beach,California,USA. (cited on page 79)

Girdhar, R. and Ramanan, D., 2017. Attentional pooling for action recognition. In *NIPS*. (cited on pages 121 and 129)

Girshick, R., 2015. Fast r-cnn. In *ICCV*, 1440–1448. IEEE, Santiago, Chile. (cited on page 67)

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J., 2016. Region-based convolutional networks for accurate object detection and segmentation. *TPAMI*, 38, 1 (Jan. 2016), 142–158. (cited on page 67)

Goyal, R.; Ebrahimi Kahou, S.; Michalski, V.; Materzynska, J.; Westphal, S.; Kim, H.; Haenel, V.; Fruend, I.; Yianilos, P.; Mueller-Freitag, M.; Hoppe, F.; Thurau, C.; Bax, I.; and Memisevic, R., 2017. The "something something" video database for learning and evaluating visual common sense. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 10)

Gu, C.; Sun, C.; Ross, D. A.; Vondrick, C.; Pantofaru, C.; Li, Y.; Vijayanarasimhan, S.; Toderici, G.; Ricco, S.; Sukthankar, R.; Schmid, C.; and Malik, J., 2018. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 65 and 67)

Guan, J.; Zhang, M.; and Lu, Z., 2020. Large-scale cross-domain few-shot learning. In *ACCV*. (cited on pages 12 and 164)

Guo, H.; Wang, H.; and Ji, Q., 2022. Uncertainty-guided probabilistic transformer for complex action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 20052–20061. (cited on page 7)

Guo, M.; Chou, E.; Huang, D.-A.; Song, S.; Yeung, S.; and Fei-Fei, L., 2018. Neural graph matching networks for fewshot 3d action recognition. In *ECCV*, 653–669. (cited on pages 12, 13, 15, 164, and 166)

Guo, Y.; Codella, N. C.; Karlinsky, L.; Codella, J. V.; Smith, J. R.; Saenko, K.; Rosing, T.; and Feris, R., 2020. A broader study of cross-domain few-shot learning. In *ECCV*. (cited on pages 12 and 164)

Gupta, S.; Girshick, R.; Arbelaez, P.; and Malik, J., 2014. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. *ECCV*, (2014), 1–16. (cited on pages 21 and 25)

Haasdonk, B. and Burkhardt, H., 2007. Invariant kernel functions for pattern analysis and machine learning. *Mach. Learn.*, 68, 1 (2007), 35–61. (cited on page 169)

Hadji, I. and Wildes, R. P., 2018. A new large scale dynamic texture dataset with application to ConvNet understanding. In *ECCV*. (cited on page 78)

Hao, X.; Li, J.; Guo, Y.; Jiang, T.; and Yu, M., 2021. Hypergraph neural network for skeleton-based action recognition. *IEEE Transactions on Image Processing*, 30 (2021), 2263–2275. doi:10.1109/TIP.2021.3051495. (cited on pages 8, 122, 125, 126, 133, and 134)

Harandi, M.; Salzmann, M.; and Hartley, R., 2018. Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods. *TPAMI*, (2018). (cited on page 61)

Harandi, M.; Salzmann, M.; and Porikli, F., 2014. Bregman divergences for infinite dimensional covariance matrices. *CVPR*, (2014). (cited on page 86)

Hashiguchi, R. and Tamaki, T., 2022. Vision transformer with cross-attention by temporal shift for efficient action recognition. doi:10.48550/ARXIV.2204.00452. https://arxiv.org/abs/2204.00452. (cited on pages 1 and 124)

Hastie, T.; Tibshirani, R.; and Friedman, J., 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA. (cited on page 151)

He, C.; Xiao, C.; Liu, S.; Qin, X.; Zhao, Y.; and Zhang, X., 2021. Single-skeleton and dual-skeleton hypergraph convolution neural networks for skeleton-based action recognition. In *Neural Information Processing*, 15–27. Springer International Publishing, Cham. (cited on pages 133 and 134)

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B., 2017. Mask r-cnn. In *ICCV*, 2980–2988. IEEE, Venice, Italy. (cited on page 67)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 65 and 67)

Herath, S.; Harandi, M.; Fernando, B.; and Nock, R., 2019. Min-max statistical alignment for transfer learning. *CVPR*, (2019). (cited on page 61)

Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Comput.*, (1997), 1735–1780. (cited on page 23)

Hodges, M. R. and Pollack, M. E., 2007. An 'Object-Use Fingerprint': The Use of Electronic Sensors for Human Identification. *UbiComp*, (2007). (cited on page 21)

Hong, J.; Fisher, M.; Gharbi, M.; and Fatahalian, K., 2021. Video pose distillation for few-shot, fine-grained sports action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 9254–9263. (cited on page 15)

Horn, B. K. P. and Schunck, B. G., 1981. Determining optical flow. *Artificial Intelligence*, 17 (1981), 185–203. (cited on pages 47 and 67)

Hou, Q.; Cheng, M.-M.; Hu, X.; Borji, A.; Tu, Z.; and Torr, P. H. S., 2017. Deeply supervised salient object detection with short connections. In *CVPR*, 3203–3212. IEEE, Honolulu, HI, USA. (cited on page 68)

Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; Adam, H.; and Le, Q., 2019. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1314–1324. IEEE Computer Society, Los Alamitos, CA, USA. doi:10.1109/ICCV.2019.00140. https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00140. (cited on page 191)

Hu, J.-F.; Zheng, W.-S.; Pan, J.; Lai, J.; and Zhang, J., 2018. Deep bilinear learning for rgb-d action recognition. *ECCV*, (2018). (cited on pages 21, 22, and 109)

Huang, Z.; Wan, C.; Probst, T.; and Gool, L. V., 2017. Deep Learning on Lie Groups for Skeleton-based Action Recognition. In *CVPR*, 6099–6108. (cited on pages 21 and 22)

Huckle, T., 2019. www5.in.tum.de/persons/huckle/tensor-kurs_1.pdf. (cited on page 90)

Hüllermeier, E. and Waegeman, W., 2021. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110, 3 (2021), 457–506. doi:10.1007/s10994-021-05946-3. (cited on page 140)

Hunter, I. M. L., 1980. Book review: Thinking in perspective: Critical essays in the study of thought processes. *Quarterly Journal of Experimental Psychology*, 32, 2 (1980), 358–359. doi:10.1080/14640748008401170i. https://doi.org/10.1080/14640748008401170i. (cited on page 47)

Hussein, M. E.; Torki, M.; Gowayyed, M.; and El-Saban, M., 2013. Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations. *IJCAI*, (2013). (cited on pages 86 and 88)

Iandola, F. N.; Moskewicz, M. W.; Ashraf, K.; Han, S.; Dally, W. J.; and Keutzer, K., 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360 (2016). http://arxiv.org/abs/1602.07360. (cited on page 191)

Indrayan, A., 2008. *Medical biostatistics*. Chapman & Hall/CRC,, Boca Raton :, 2nd ed. edn. http://www.loc.gov/catdir/toc/ecip0723/2007030353.html. (cited on page 140)

Insafutdinov, E.; Pishchulin, L.; Andres, B.; Andriluka, M.; and Schiele, B., 2016. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. *ECCV*, (2016). (cited on page 89)

Jebara, T.; Kondor, R.; and Howard, A., 2004. Probability product kernels. *JMLR*, 5 (2004), 819–844. (cited on pages 70 and 90)

Jégou, H.; Douze, M.; and Schmid, C., 2009. On the Burstiness of Visual Elements. *CVPR*, (2009), 1169–1176. (cited on pages 48, 69, and 94)

Ji, S.; Xu, W.; Yang, M.; and Yu, K., 2013. 3d convolutional neural networks for human action recognition. *TPAMI*, 35, 1 (1 2013), 221–231. doi:10.1109/TPAMI.2012.59. (cited on pages 5, 47, 68, and 86)

Jiang, Y.; Sun, Z.; Yu, S.; Wang, S.; and Song, Y., 2022. A graph skeleton transformer network for action recognition. *Symmetry*, 14, 8 (2022). doi:10.3390/sym14081547. https://www.mdpi.com/2073-8994/14/8/1547. (cited on pages 1, 8, and 133)

Johansson, G., 1973. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14, 2 (1973), 201–211. (cited on page 88)

Junejo, I. N.; Dexter, E.; Laptev, I.; and Pérez, P., 2008. Cross-view action recognition from temporal self-similarities. In *Computer Vision – ECCV 2008*, 293–306. Springer Berlin Heidelberg, Berlin, Heidelberg. (cited on pages 10 and 11)

Kang, M.-S.; Kang, D.; and Kim, H., 2023. Efficient skeleton-based action recognition via joint-mapping strategies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3403–3412. (cited on page 1)

Kar, A.; Rai, N.; Sikka, K.; and Sharma, G., 2017. AdaScan: Adaptive Scan Pooling in Deep Convolutional Neural Networks for Human Action Recognition in Videos. *CVPR*, (2017), 3376–3385. (cited on page 21)

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks. In *CVPR*, 1725–1732. doi:10.1109/CVPR.2014.223. https://doi.org/10.1109/CVPR.2014.223. (cited on pages 10, 47, 68, and 86)

Katharopoulos, A.; Vyas, A.; Pappas, N.; and Fleuret, F., 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, 5156–5165. PMLR. https://proceedings.mlr.press/v119/katharopoulos20a.html. (cited on page 126)

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; Suleyman, M.; and Zisserman, A., 2017. The kinetics human action video dataset. *CoRR abs/1705.06950*, (2017). (cited on pages xxix, 2, 4, 87, 102, 103, 104, 110, 130, 150, and 174)

Ke, Q.; An, S.; Bennamoun, M.; Sohel, F.; and Boussaid, F., 2017a. SkeletonNet: Mining Deep Part Features for 3D Action Recognition. *IEEE SPL*, 24, 6 (2017), 731–735. (cited on pages 3, 20, 21, 22, and 23)

Ke, Q.; Bennamoun, M.; An, S.; Sohel, F.; and Boussaid, F., 2017b. A New Representation of Skeleton Sequences for 3D Action Recognition. In *CVPR*. (cited on pages 3, 20, 21, 22, 23, 24, 26, 27, 30, 33, 35, 36, 38, 109, and 186)

Ke, Q.; Bennamoun, M.; An, S.; Sohel, F.; and Boussaid, F., 2018. Learning clip representations for skeleton-based 3d action recognition. *IEEE TIP*, 27, 6 (2018), 2842–2855. (cited on page 186)

Ke, Y.; Sukthankar, R.; and Hebert, M., 2007. Spatio-temporal Shape and Flow Correlation for Action Recognition. *Proc. 7th Int. Workshop on Visual Surveillance*, (2007). (cited on pages 19 and 21)

Kendall, A. and Gal, Y., 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. (cited on page 140)

Kim, J.; Oh, S.; and Hong, S., 2021. Transformers generalize deepsets and can be extended to graphs & hypergraphs. In *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=scn3RYn1DYx. (cited on pages 120, 124, and 126)

Kim, T.-K.; Wong, K.-Y. K.; and Cipolla, R., 2007. Tensor canonical correlation analysis for action classification. *CVPR*, (2007). (cited on page 88)

Kim, T. S. and Reiter, A., 2017. Interpretable 3d human action analysis with temporal convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1623–1631. doi:10.1109/CVPRW.2017.207. (cited on page 133)

Kipf, T. N. and Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*. (cited on pages 8, 26, 165, 166, 167, 168, and 171)

Kiureghian, A. D. and Ditlevsen, O., 2009. Aleatory or epistemic? does it matter? *Structural Safety*, 31, 2 (2009), 105–112. doi:https://doi.org/10.1016/j.strusafe.2008.06.020. Risk Acceptance and Risk Communication. (cited on page 140)

Kläser, A.; Marszalek, M.; and Schmid, C., 2008. A Spatio-Temporal Descriptor Based on 3D-Gradients. *BMCV*, (2008), 1–10. (cited on pages 43, 44, 46, 63, and 67)

Klicpera, J.; Bojchevski, A.; and Gunnemann, S., 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*. (cited on pages 165, 167, 168, and 171)

Koch, G.; Zemel, R.; and Salakhutdinov, R., 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, vol. 2. (cited on page 164)

Kong, J.; Bian, Y.; and Jiang, M., 2022. Mtt: Multi-scale temporal transformer for skeleton-based action recognition. *IEEE Signal Processing Letters*, 29 (2022), 528–532. doi:10.1109/LSP.2022.3142675. (cited on pages 1, 8, and 133)

Koniusz, P. and Cherian, A., 2016. Sparse coding for third-order super-symmetric tensor descriptors with application to texture recognition. *CVPR*, (2016). (cited on page 89)

Koniusz, P.; Cherian, A.; and Porikli, F., 2016a. Tensor representations via kernel linearization for action recognition from 3D skeletons. *ECCV*, (2016). (cited on pages 7, 9, 20, 22, 23, 24, 25, 27, 30, 33, 35, 36, 38, 47, 68, 86, and 87)

Koniusz, P.; Cherian, A.; and Porikli, F., 2016b. Tensor representations via kernel linearization for action recognition from 3D skeletons (extended version). *CoRR abs/1604.00239*, (2016). (cited on pages 9 and 115)

Koniusz, P. and Mikolajczyk, K., 2011. Soft assignment of visual words as linear coordinate coding and optimisation of its reconstruction error. In *2011 18th IEEE International Conference on Image Processing*, 2413–2416. doi:10.1109/ICIP.2011.6116129. (cited on pages 46, 67, and 148)

Koniusz, P.; Tas, Y.; and Porikli, F., 2017. Domain adaptation by mixture of alignments of second- or higher-order scatter tensors. *CVPR*, (2017). (cited on page 61)

Koniusz, P.; Tas, Y.; Zhang, H.; Harandi, M.; Porikli, F.; and Zhang, R., 2018a. Museum exhibit identification challenge for the supervised domain adaptation and beyond. In *ECCV*. (cited on page 61)

Koniusz, P.; Wang, L.; and Cherian, A., 2020. Tensor representations for action recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE. (cited on pages 9, 30, 68, 119, 120, 121, 122, 146, and 163)

Koniusz, P.; Wang, L.; and Sun, K., 2021. High-order tensor pooling with attention for action recognition. *arXiv*, (2021). (cited on pages 48, 69, 119, 121, 129, and 163)

Koniusz, P.; Yan, F.; Gosselin, P.; and Mikolajczyk, K., 2013a. Higher-order occurrence pooling on mid- and low-level features: Visual concept detection. *Technical Report*, (2013). (cited on pages 85 and 89)

Koniusz, P.; Yan, F.; Gosselin, P.; and Mikolajczyk, K., 2016c. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *TPAMI*, (2016). (cited on pages 45, 47, 48, 68, 69, 85, 86, 89, 94, 95, and 106)

Koniusz, P.; Yan, F.; and Mikolajczyk, K., 2013b. Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. *Computer Vision and Image Understanding*, 117, 5 (2013), 479–492. doi:https://doi.org/10.1016/j.cviu.2012.10.010. (cited on pages 45, 46, 48, 67, 69, and 148)

Koniusz, P. and Zhang, H., 2020. Power normalizations in fine-grained image, few-shot image and graph classification. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE. (cited on pages 12, 69, 89, 121, and 164)

Koniusz, P.; Zhang, H.; and Porikli, F., 2018b. A deeper look at power normalizations. *CVPR*, (2018), 5774–5783. (cited on pages 45, 48, 49, 69, 85, 87, 89, 96, 110, and 117)

Koppula, H. S.; Gupta, R.; and Saxena, A., 2013. Learning Human Activities and Object Affordances from RGB-D Videos. *IJRR*, (1 2013). (cited on page 21)

Korban, M. and Li, X., 2020. Ddgcn: A dynamic directed graph convolutional network for action recognition. In *Computer Vision – ECCV 2020*, 761–776. Springer International Publishing, Cham. (cited on pages 8 and 121)

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A., 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4365–4374. Association for Computational Linguistics, Hong Kong, China. doi:10.18653/v1/D19-1445. https://aclanthology.org/D19-1445. (cited on page 132)

Kozlov, A.; Andronov, V.; and Gritsenko, Y., 2020. *Lightweight Network Architecture for Real-Time Action Recognition*, 2074–2080. Association for Computing Machinery, New York, NY, USA. ISBN 9781450368667. https://doi.org/10.1145/3341105.3373906. (cited on page 191)

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E., 2012. ImageNet classification with deep convolutional neural networks. *NIPS*, (2012), 1106–1114. (cited on page 47)

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T., 2011. HMDB: A large video database for human motion recognition. In *ICCV*, 2556–2563. (cited on pages 2, 54, 74, 87, 102, and 103)

Kumar, D.; Kumar, C.; Seah, C.; Xia, S.; and Shao, M., 2020. Finding achilles' heel: Adversarial attack on multi-modal action recognition. In *MM*, 3829–3837. ACM, Seattle, United States. doi:10.1145/3394171.3413531. https://doi.org/10.1145/3394171.3413531. (cited on page 68)

Kumar Roy, S.; Harandi, M.; Nock, R.; and Hartley, R., 2019. Siamese networks: The tale of two manifolds. In *ICCV*. (cited on page 61)

Lake, B. M.; Salakhutdinov, R.; Gross, J.; and Tenenbaum, J. B., 2011. One shot learning of simple visual concepts. *CogSci*, (2011). (cited on pages 12 and 164)

Laptev, I., 2005. On space-time interest points. *IJCV*, 64, 2-3 (Sep. 2005), 107–123. doi:10.1007/s11263-005-1838-7. http://dx.doi.org/10.1007/s11263-005-1838-7. (cited on pages 45 and 66)

Laptev, I.; Marszalek, M.; Schmid, C.; and Rozenfeld, B., 2008. Learning Realistic Human Actions from Movies. *CVPR*, (2008), 1–8. (cited on pages 19 and 21)

Lee, I.; Kim, D.; Kang, S.; and Lee, S., 2017. Ensemble Deep Learning for Skeleton-based Action Recognition using Temporal Sliding LSTM networks. In *ICCV*, 1012–1020. (cited on pages 21, 22, and 23)

Lee, J. B.; Rossi, R.; and Kong, X., 2018. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18 (London, United Kingdom, 2018), 1666–1674. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3219819.3219980. https://doi.org/10.1145/3219819.3219980. (cited on page 124)

Lee, S.; Woo, S.; Park, Y.; Nugroho, M. A.; and Kim, C., 2023. Modality mixer for multi-modal action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3298–3307. (cited on pages 1 and 10)

Li, B.; Camps, O. I.; and Sznaier, M., 2012. Cross-view activity recognition using hankelets. *CVPR*, (2012). (cited on page 86)

Li, B.; He, M.; Cheng, X.; Chen, Y.; and Dai, Y., 2017a. Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn. *CoRR abs/1704.05645v2*, (2017). (cited on page 109)

Li, C.; Cui, Z.; Zheng, W.; Xu, C.; and Yang, J., 2018a. Spatio-Temporal Graph Convolution for Skeleton Based Action Recognition. In *AAAI*, 3482–3489. (cited on pages 21 and 22)

Li, C.; Su, B.; Wang, J.; and Zhang, Q., 2014. Human action recognition using multi-velocity STIPs and motion energy orientation histogram. *J. Inf. Sci. Eng.*, 30 (2014), 295–312. (cited on pages 45 and 66)

Li, F. F.; VanRullen, R.; Koch, C.; and Perona, P., 2002. Rapid natural scene categorization in the near absence of attention. *PNAS*, 99, 14 (2002), 9596–9601. (cited on pages 12 and 164)

Li, J.; Wei, P.; Zhang, Y.; and Zheng, N., 2020a. A slow-i-fast-p architecture for compressed video action recognition. In *MM*, 2039–2047. ACM, Seattle, United States. (cited on page 68)

Li, K.; Zhang, Y.; Li, K.; and Fu, Y., 2020b. Adversarial feature hallucination networks for few-shot learning. In *CVPR*. (cited on pages 12 and 164)

Li, L.; Wang, M.; Ni, B.; Wang, H.; Yang, J.; and Zhang, W., 2021a. 3d human action representation learning via cross-view consistency pursuit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4741–4750. (cited on page 11)

Li, M.; Chen, S.; Chen, X.; Zhang, Y.; Wang, Y.; and Tian, Q., 2019. Actional-structural graph convolutional networks for skeleton-based action recognition. In *CVPR*. (cited on pages 8, 22, 121, 133, 134, and 166)

Li, M.; Chen, S.; Chen, X.; Zhang, Y.; Wang, Y.; and Tian, Q., 2022. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44, 6 (2022), 3316–3333. doi:10.1109/TPAMI.2021.3053765. (cited on pages 1, 133, and 134)

Li, R. and Zickler, T., 2012. Discriminative virtual views for cross-view action recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2855–2862. doi:10.1109/CVPR.2012.6248011. (cited on pages 10 and 11)

Li, S.; Li, W.; Cook, C.; Zhu, C.; and Gao, Y., 2018b. Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN. *CVPR*, (2018), 5457–5466. (cited on pages 21, 23, 24, 26, 27, 33, 35, 36, and 38)

Li, S.; Liu, H.; Fei, M.; Yu, X.; and Lin, W., 2021b. Temporal alignment via event boundary for few-shot action recognition. In *BMVC*. (cited on page 13)

Li, S.; Liu, H.; Qian, R.; Li, Y.; See, J.; Fei, M.; Yu, X.; and Lin, W., 2021c. TTAN: two-stage temporal alignment network for few-shot action recognition. *CoRR*, (2021). (cited on pages 13, 16, and 146)

Li, W.; Wen, L.; Chang, M.-C.; Lim, S. N.; and Lyu, S., 2017b. Adaptive rnn tree for large-scale human action recognition. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 1453–1461. doi:10.1109/ICCV.2017.161. (cited on page 11)

Li, W.; Zhang, Z.; and Liu, Z., 2010. Action recognition based on a bag of 3D points. *CVPR Workshop*, (2010), 9–14. doi:10.1109/CVPRW.2010.5543273. (cited on pages 2, 3, 20, 21, 22, 27, 87, 102, 108, 109, 182, and 198)

Li, Z.; Gong, X.; Song, R.; Duan, P.; Liu, J.; and Zhang, W., 2023. Smam: Self and mutual adaptive matching for skeleton-based few-shot action recognition. *IEEE Transactions on Image Processing*, 32 (2023), 392–402. doi:10.1109/TIP.2022.3226410. (cited on page 15)

Lichtenstein, M.; Sattigeri, P.; Feris, R.; Giryes, R.; and Karlinsky, L., 2020. Tafssl: Task-adaptive feature sub-space learning for few-shot classification. In *ECCV*. (cited on pages 12 and 164)

Lin, H.; Cheng, X.; Wu, X.; Yang, F.; Shen, D.; Wang, Z.; Song, Q.; and Yuan, W., 2021. CAT: cross attention in vision transformer. *CoRR*, abs/2106.05786 (2021). https://arxiv.org/abs/2106.05786. (cited on page 124)

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L., 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, 740–755. Springer International Publishing, Cham. (cited on pages 65 and 67)

Lin, T.-Y. and Maji, S., 2017. Improved Bilinear Pooling with CNNs. *BMVC*, (2017). (cited on pages 85, 87, 89, and 96)

Lin, T.-Y.; Maji, S.; and Koniusz, P., 2018. Second-order democratic aggregation. *ECCV*, (2018). (cited on pages 89 and 121)

Lingqiao, L.; Wang, L.; and Liu, X., 2011. In Defence of Soft-assignment Coding. *ICCV*, (2011). (cited on pages 46 and 67)

Liu, D. C. and Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45 (1989), 503–528. (cited on page 150)

Liu, J.; Luo, J.; and Shah, M., 2009. Recognizing Realistic Actions from Videos in the Wild'. *CVPR*, (2009), 1–8. (cited on pages 19 and 21)

Liu, J. and Shah, M., 2008. Learning Human Actions via Information Maximization. *CVPR*, (2008), 1–9. (cited on pages 19 and 21)

Liu, J.; Shah, M.; Kuipers, B.; and Savarese, S., 2011a. Cross-view action recognition via view knowledge transfer. In *CVPR 2011*, 3209–3216. doi:10.1109/CVPR.2011.5995729. (cited on pages 10 and 11)

Liu, J.; Shahroudy, A.; Perez, M.; Wang, G.; Duan, L.-Y.; and Kot, A. C., 2019a. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2019). doi:10.1109/TPAMI.2019.2916873. (cited on pages 2, 3, 5, 10, 14, 15, 130, 150, 152, 164, 166, 167, 174, 175, 178, 182, 199, and 200)

Liu, J.; Shahroudy, A.; Xu, D.; Kot, A. C.; and Wang, G., 2018. Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates. In *TPAMI*, 1–14. (cited on pages 21, 22, and 23)

Liu, J.; Shahroudy, A.; Xu, D.; and Wang, G., 2016. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. *ECCV*, (2016), 816–833. (cited on pages 11, 22, and 23)

Liu, J.; Wang, G.; Hu, P.; Duan, L.; and Kot, A. C., 2017. Global context-aware attention lstm networks for 3d action recognition. In *CVPR*, 3671–3680. (cited on pages 11, 14, 15, 21, 22, 23, 164, and 166)

Liu, L.; Wang, L.; and Liu, X., 2011b. In defense of soft-assignment coding. In *2011 International Conference on Computer Vision*, 2486–2493. doi:10.1109/ICCV.2011.6126534. (cited on page 148)

Liu, S.; Lv, P.; Zhang, Y.; Fu, J.; Cheng, J.; Li, W.; Zhou, B.; and Xu, M., 2020a. Semi-dynamic hypergraph neural network for 3d pose estimation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 782–788. International Joint Conferences on Artificial Intelligence Organization. doi:10.24963/ijcai.2020/109. https://doi.org/10.24963/ijcai.2020/109. Main track. (cited on pages 8, 122, 125, and 126)

Liu, Z.; Gao, G.; Qin, A. K.; Wu, T.; and Liu, C. H., 2019b. Action recognition with bootstrapping based long-range temporal context attention. In *MM*, 583–591. ACM, Nice, France. doi:10.1145/3343031.3350916. https://doi.org/10.1145/3343031.3350916. (cited on page 68)

Liu, Z.; Ning, J.; Cao, Y.; Wei, Y.; Zhang, Z.; Lin, S.; and Hu, H., 2022. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3202–3211. (cited on page 7)

Liu, Z.; Zhang, H.; Chen, Z.; Wang, Z.; and Ouyang, W., 2020b. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *CVPR*. (cited on pages 133 and 166)

Lohit, S.; Wang, Q.; and Turaga, P., 2019. Temporal transformer networks: Joint learning of invariant and discriminative time warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 146)

Lu, C. and Koniusz, P., 2022. Few-shot keypoint detection with uncertainty learning for unseen species. *CVPR*, (2022). (cited on pages 12 and 164)

Lu, H.; Plataniotis, K. N.; and Venetsanopoulos, A. N., 2011. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44, 7 (2011), 1540–1551. (cited on page 88)

Luo, Q.; Wang, L.; Lv, J.; Xiang, S.; and Pan, C., 2021. Few-shot learning via feature hallucination with variational inference. In *WACV*. (cited on pages 12 and 164)

Lv, F. and Nevatia, R., 2006. Recognition and segmentation of 3-D human action using hmm and multi-class adaboost. *ECCV*, (2006), 359–372. doi:10.1007/11744085_28. (cited on page 88)

Ma, N.; Zhang, H.; Li, X.; Zhou, S.; Zhang, Z.; Wen, J.; Li, H.; Gu, J.; and Bu, J., 2022. Learning spatial-preserved skeleton representations for few-shot action recognition. In *Computer Vision – ECCV 2022*, 174–191. Springer Nature Switzerland, Cham. (cited on page 15)

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J., 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*. (cited on page 191)

Mahasseni, B. and Todorovic, S., 2013. Latent multitask learning for view-invariant action recognition. In *2013 IEEE International Conference on Computer Vision*, 3128–3135. doi:10.1109/ICCV.2013.388. (cited on pages 10 and 11)

Mahmud, T.; Hasan, M.; and Roy-Chowdhury, A. K., 2017. Joint prediction of activity labels and starting times in untrimmed videos. *ICCV*, (2017). (cited on page 86)

Mairal, J.; Koniusz, P.; Harchaoui, Z.; and Schmid, C., 2014. Convolutional kernel networks. *NIPS*, (2014). (cited on page 72)

Mallick, T.; Das, P. P.; and Majumdar, A. K., 2014. Characterizations of Noise in Kinect Depth Images: A Review. *IEEE SEN*, 14, 6 (2014), 1731–1740. doi:10.1109/JSEN.2014.2309987. (cited on page 20)

Martinez, J.; Hossain, R.; Romero, J.; and Little, J. J., 2017. A simple yet effective baseline for 3d human pose estimation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2659–2668. doi:10.1109/ICCV.2017.288. (cited on pages 150 and 174)

Matthies, H. G., 2007. Quantifying uncertainty: Modern computational representation of probability and applications. In *Extreme Man-Made and Natural Hazards in Dynamics of Structures*, 105–135. Springer Netherlands, Dordrecht. (cited on page 140)

McInnes, L.; Healy, J.; Saul, N.; and Grossberger, L., 2018. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3, 29 (2018), 861. (cited on page 82)

Memmesheimer, R.; Häring, S.; Theisen, N.; and Paulus, D., 2021. Skeleton-dml: Deep metric learning for skeleton-based one-shot action recognition. (cited on pages 14, 15, 146, 164, 166, 167, and 178)

Memmesheimer, R.; Theisen, N.; and Paulus, D., 2020. Signal level deep metric learning for multimodal one-shot action recognition. (cited on pages 14, 15, 146, 164, 166, and 178)

Mensch, A. and Blondel, M., 2018. Differentiable dynamic programming for structured prediction and attention. In *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, 3462–3471. PMLR. (cited on page 145)

Miller, E. G.; Matsakis, N. E.; and Viola, P. A., 2000. Learning from one example through shared densities on transforms. *CVPR*, 1 (2000), 464–471. (cited on pages 12 and 164)

Mina, B.; Zoumpourlis, G.; and Patras, I., 2019. Tarn: Temporal attentive relation network for few-shot and zero-shot action recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, 130.1–130.14. BMVA Press. doi:10.5244/C.33.130. (cited on pages 16 and 146)

Mishra, A.; Verma, V. K.; Reddy, M. S. K.; Arulkumar, S.; Rai, P.; and Mittal, A., 2018. A generative approach to zero-shot and few-shot action recognition. In *WACV*, 372–380. (cited on pages 12, 164, and 166)

Munro, J. and Damen, D., 2020. Multi-modal domain adaptation for fine-grained action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 122–132. (cited on page 10)

Ni, B.; Peng, H.; Chen, M.; Zhang, S.; Meng, G.; Fu, J.; Xiang, S.; and Ling, H., 2022. Expanding language-image pretrained models for general video recognition. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, 1–18. Springer. (cited on pages 7, 9, and 10)

Noguchi, C. and Tanizawa, T., 2023. Ego-vehicle action recognition based on semi-supervised contrastive learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 5988–5998. (cited on page 1)

Ofli, F.; Chaudhry, R.; Kurillo, G.; Vidal, R.; and Bajcsy, R., 2014. Sequence of the most informative joints (SMIJ). *J. Vis. Comun. Image Represent.*, 25, 1 (2014), 24–38. doi:10.1016/j.jvcir.2013.04.007. (cited on page 88)

Ohn-Bar, E. and Trivedi, M. M., 2013. Joint angles similarities and HOG^2 for action recognition. *CVPR Workshop*, (2013). (cited on page 88)

Oreifej, O. and Liu, Z., 2013. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In *CVPR*, 716–723. (cited on pages 2, 3, 4, 9, 20, 21, 22, 24, 26, 27, 29, 33, 35, 36, 38, 182, and 198)

Papenberg, N.; Bruhn, A.; Brox, T.; Didas, S.; and Weickert, J., 2006. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 67 (2006), 141–158. (cited on pages 47 and 67)

Parameswaran, V. and Chellappa, R., 2006. View invariance for human action recognition. *IJCV*, 66, 1 (2006), 83–101. doi:10.1007/s11263-005-3671-4. (cited on page 88)

Patravali, J.; Mittal, G.; Yu, Y.; Li, F.; and Chen, M., 2021. Unsupervised few-shot action recognition via action-appearance aligned meta-adaptation. *CoRR*, abs/2109.15317 (2021). https://arxiv.org/abs/2109.15317. (cited on page 14)

Peng, K.; Roitberg, A.; Yang, K.; Zhang, J.; and Stiefelhagen, R., 2023. Delving deep into one-shot skeleton-based action recognition with diverse occlusions. *IEEE Transactions on Multimedia*, (2023), 1–16. doi:10.1109/TMM.2023.3235300. (cited on page 15)

Peng, W.; Hong, X.; Chen, H.; and Zhao, G., 2020. Learning graph convolutional network for skeleton-based human action recognition by neural searching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 03 (Apr. 2020), 2669–2676. doi:10.1609/aaai.v34i03.5652. https://ojs.aaai.org/index.php/AAAI/article/view/5652. (cited on pages 133 and 134)

Peng, W.; Shi, J.; Varanka, T.; and Zhao, G., 2021. Rethinking the st-gcns for 3d skeleton-based human action recognition. *Neurocomputing*, 454 (2021), 45–53. (cited on page 135)

Perrett, T.; Masullo, A.; Burghardt, T.; Mirmehdi, M.; and Damen, D., 2021. Temporal-relational crosstransformers for few-shot action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 475–484. (cited on pages 13, 16, and 146)

Perronnin, F. and Dance, C., 2007. Fisher kernels on visual vocabularies for image categorization. *CVPR*, 0 (2007), 1–8. (cited on pages 43, 46, 48, 64, and 67)

Perronnin, F.; Sánchez, J.; and Mensink, T., 2010. Improving the Fisher Kernel for Large-Scale Image Classification. *ECCV*, (2010), 143–156. (cited on pages 43, 46, 48, 64, 67, and 104)

Pham, N. and Pagh, R., 2013. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 239–247. ACM. doi:10.1145/2487575.2487591. http://doi.acm.org/10.1145/2487575.2487591. (cited on pages 45 and 50)

Piergiovanni, A.; Angelova, A.; Toshev, A.; and Ryoo, M. S., 2019. Evolving space-time neural architectures for videos. In *ICCV*. (cited on page 77)

Plizzari, C.; Cannici, M.; and Matteucci, M., 2021. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208-209 (2021), 103219. doi:https://doi.org/10.1016/j.cviu.2021.103219. https://www.sciencedirect.com/science/article/pii/S1077314221000631. (cited on pages 8 and 133)

Prabowo, A.; Koniusz, P.; Shao, W.; and Salim, F. D., 2019. COLTRANE: convolutional trajectory network for deep map inference. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys 2019, New York, NY, USA, November 13-14, 2019*, 21–30. ACM. doi:10.1145/3360322.3360853. (cited on page 160)

Presti, L. L. and La Cascia, M., 2015. 3D skeleton-based human action classification: A survey. *Pattern Recognition*, (2015). (cited on pages 86 and 88)

Qin, Z.; Liu, Y.; Ji, P.; Kim, D.; Wang, L.; McKay, B.; Anwar, S.; and Gedeon, T., 2022. Fusing higher-order features in graph neural networks for skeleton-based action recognition. *IEEE TNNLS*, (2022). (cited on pages 1, 119, 120, and 164)

Qiu, Z.; Yao, T.; and Mei, T., 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 6)

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al., 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR. (cited on pages 7, 9, and 10)

Rahman, S.; Koniusz, P.; Wang, L.; Zhou, L.; Moghadam, P.; and Sun, C., 2023. Learning partial correlation based deep visual representation for image classification. In *CVPR*. (cited on page 121)

Rahmani, H. and Bennamoun, M., 2017. Learning Action Recognition Model From Depth and Skeleton Videos. In *ICCV*, 5832–5841. (cited on pages 3, 9, 20, 21, 22, 23, 24, and 25)

Rahmani, H.; Huynh, D. Q.; Mahmood, A.; and Mian, A., 2016a. Discriminative Human Action Classification using Locality-constrained Linear Coding. *Pattern Recognit. Lett.*, (2016), 62–71. (cited on pages 21 and 22)

Rahmani, H.; Mahmood, A.; Huynh, D. Q.; and Mian, A., 2014a. Action Classification with Locality-constrained Linear Coding. In *ICPR*, 3511–3516. (cited on pages 21 and 22)

Rahmani, H.; Mahmood, A.; Huynh, D. Q.; and Mian, A., 2014b. HOPC: Histogram of Oriented Principal Components of 3D Pointclouds for Action Recognition. In *ECCV*, 742–757. (cited on pages xvii, 2, 3, 21, 22, 27, 28, 182, and 198)

Rahmani, H.; Mahmood, A.; Huynh, D. Q.; and Mian, A., 2014c. Real Time Action Recognition Using Histograms of Depth Gradients and Random Decision Forests. In *WACV*, 626–633. (cited on pages 3, 4, 7, 9, 20, 21, 23, 24, 25, 27, 30, 31, and 32)

Rahmani, H.; Mahmood, A.; Huynh, D. Q.; and Mian, A., 2016b. Histogram of Oriented Principal Components for Cross-View Action Recognition. *TPAMI*, (2016), 2430–2443. (cited on pages 2, 3, 4, 20, 22, 24, 25, 26, 27, 28, 29, 33, 35, 36, 38, 166, 174, and 182)

Rahmani, H. and Mian, A., 2015. Learning a non-linear knowledge transfer model for cross-view action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2458–2466. doi:10.1109/CVPR.2015.7298860. (cited on page 11)

Rahmani, H. and Mian, A., 2016. 3D Action Recognition from Novel Viewpoints. *CVPR*, (2016), 1–12. (cited on pages 3, 20, 21, 22, 24, 25, 27, 30, 33, 36, and 38)

Rahmani, H.; Mian, A.; and Shah, M., 2015. Learning a Deep Model for Human Action Recognition from Novel Viewpoints. *TPAMI*, (2015), 1–14. (cited on pages 21 and 22)

Ramachandran, P.; Liu, P. J.; and Le, Q. V., 2018. Unsupervised pretraining for sequence to sequence learning. (cited on page 146)

Ranasinghe, K.; Naseer, M.; Khan, S.; Khan, F. S.; and Ryoo, M., 2022. Self-supervised video transformer. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition*. (cited on page 122)

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A., 2015. You only look once: Unified, real-time object detection. In *CVPR*, 779–788. IEEE, Boston, MA, USA. (cited on page 67)

Ren, S.; He, K.; Girshick, R.; and Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 91–99. MIT Press, Montreal, Canada. (cited on pages 55, 65, 67, and 75)

Revaud, J.; Weinzaepfel, P.; Harchaoui, Z.; and Schmid, C., 2015. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR*. (cited on pages 47 and 67)

Rohrbach, M.; Amin, S.; Andriluka, M.; and Schiele, B., 2012. A database for fine grained activity detection of cooking activities. In *CVPR*. (cited on pages xxi, 2, 4, 55, 74, 86, 87, 89, 102, and 103)

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L., 2015. ImageNet large scale visual recognition challenge. *IJCV*, 115, 3 (2015), 211–252. doi:10.1007/s11263-015-0816-y. (cited on pages 47, 66, and 68)

Ryoo, M. S.; Piergiovanni, A.; Kangaspunta, J.; and Angelova, A., 2020a. Assemblenet++: Assembling modality representations via attention connections. In *ECCV*, 1–19. Springer Science+Business Media, Glasgow, UK. (cited on pages 7, 9, 12, and 68)

Ryoo, M. S.; Piergiovanni, A.; Tan, M.; and Angelova, A., 2020b. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *ICLR*, 1–15. ICLR, Addis Ababa, Ethiopia. (cited on pages 7, 9, 12, 64, 68, and 78)

Sabater, A.; Santos, L.; Santos-Victor, J.; Bernardino, A.; Montesano, L.; and Murillo, A. C., 2021. One-shot action recognition in challenging therapy scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2771–2779. doi:10.1109/CVPRW53098.2021.00312. (cited on page 14)

Sakoe, H. and Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26, 1 (1978), 43–49. doi:10.1109/TASSP.1978.1163055. (cited on page 145)

Scovanner, P.; Ali, S.; and Shah, M., 2007. A 3-Dimentional SIFT Descriptor and its Application to Action Recognition. *CRCV*, (2007), 1–4. (cited on pages 43, 45, 46, 63, 66, and 67)

Seidenari, L.; Varano, V.; Berretti, S.; Bimbo, A. D.; and Pala, P., 2013. Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses. *CVPR Workshop*, (2013). (cited on pages 2, 87, 102, and 107)

Seon, J.; Hwang, J.; Mun, J.; and Han, B., 2023. Stop or forward: Dynamic layer skipping for efficient action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3361–3370. (cited on page 1)

Shah, K.; Shah, A.; Lau, C. P.; de Melo, C. M.; and Chellappa, R., 2023. Multi-view action recognition using contrastive learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3381–3391. (cited on pages 1, 10, and 11)

Shahroudy, A.; Liu, J.; Ng, T.-T.; and Wang, G., 2016a. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. *CVPR*, (2016), 1010–1019. (cited on pages 2, 3, 7, 10, 11, 20, 21, 22, 23, 24, 26, 27, 29, 30, 33, 35, 36, 38, 87, 102, 130, 150, 166, 167, 174, and 182)

Shahroudy, A.; Ng, T.-T.; Gong, Y.; and Wang, G., 2018. Deep Multimodal Feature Analysis for Action Recognition in RGB+D Videos. *TPAMI*, (2018), 1045–1058. (cited on page 21)

Shahroudy, A.; Ng, T.-T.; Yang, Q.; and Wang, G., 2016b. Multimodal Multipart Learning for Action Recognition in Depth Videos. *TPAMI*, 38, 10 (2016), 2123–2129. (cited on pages 9, 21, 22, 23, and 24)

Shashua, A. and Hazan, T., 2005. Non-negative tensor factorization with applications to statistics and computer vision. *ICML*, (2005). (cited on page 88)

Shawe-Taylor, J. and Cristianini, N., 2004. *Kernel methods for pattern analysis*. Cambridge University Press. ISBN 0521813972. (cited on page 115)

Shechtman, E. and Irani, M., 2005. Space-time behavior based correlation. *CVPR*, (2005). (cited on page 89)

Shi, L.; Zhang, Y.; Cheng, J.; and Lu, H., 2019a. Skeleton-Based Action Recognition with Directed Graph Neural Networks. In *CVPR*, 7912–7921. (cited on page 22)

Shi, L.; Zhang, Y.; Cheng, J.; and Lu, H., 2019b. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*. (cited on pages 22, 133, and 134)

Shi, L.; Zhang, Y.; Cheng, J.; and Lu, H., 2021a. Adasgn: Adapting joint number and model size for efficient skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 13413–13422. (cited on pages 8 and 121)

Shi, Y.; Tian, Y.; Wang, Y.; Zeng, W.; and Huang, T., 2017. Learning long-term dependencies for action recognition with a biologically-inspired deep network. In *ICCV*. (cited on page 111)

Shi, Z. and Kim, T.-K., 2017. Learning and Refining of Privileged Information-based RNNs for Action Recognition from Depth Sequences. In *CVPR*, 3461–3470. (cited on pages 22 and 23)

Shi, Z.; Liang, J.; Li, Q.; Zheng, H.; Gu, Z.; Dong, J.; and Zheng, B., 2021b. Multi-modal multi-action video recognition. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 13658–13667. doi:10.1109/ICCV48922.2021.01342. (cited on page 10)

Shiri, F.; Porikli, F.; Hartley, R.; and Koniusz, P., 2018. Identity-preserving face recovery from portraits. *WACV*, (2018). (cited on page 89)

Shiri, F.; Yu, X.; Porikli, F.; Hartley, R.; and Koniusz, P., 2019a. Identity-preserving face recovery from stylized portraits. *IJCV*, 127, 6-7 (2019), 863–883. doi:10.1007/s11263-019-01169-1. (cited on page 89)

Shiri, F.; Yu, X.; Porikli, F.; Hartley, R.; and Koniusz, P., 2019b. Recovering faces from portraits with auxiliary facial attributes. *WACV*, (2019). (cited on page 89)

Shiri, F.; Yu, X.; Porikli, F.; and Koniusz, P., 2017. Face destylization. *DICTA*, (2017). (cited on page 89)

Shotton, J.; Fitzgibbon, A.; Cook, M.; Sharp, T.; Finocchio, M.; Moore, R.; Kipman, A.; and Blake, A., 2011. Real-Time Human Pose Recognition in Parts from Single Depth Images. In *CVPR*, 1297–1304. (cited on pages 3, 4, 7, 20, and 22)

Shotton, J.; Sharp, T.; Kipman, A.; Fitzgibbon, A.; Finocchio, M.; Blake, A.; Cook, M.; and Moore, R., 2013. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, (2013). (cited on page 86)

Si, C.; Chen, W.; Wang, W.; Wang, L.; and Tan, T., 2019. An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition. In *CVPR*, 1227–1236. (cited on pages 8, 22, 23, 120, and 121)

Si, C.; Jing, Y.; Wang, W.; Wang, L.; and Tan, T., 2018. Skeleton-Based Action Recognition with Spatial Reasoning and Temporal Stack Learning. In *ECCV*, 1–16. (cited on pages 21 and 22)

Sigurdsson, G. A.; Varol, G.; Wang, X.; Farhadi, A.; Laptev, I.; and Gupta, A., 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*. (cited on pages 2, 4, 10, 55, and 75)

Simon, C.; Koniusz, P.; and Harandi, M., 2021. On learning the geodesic path for incremental learning. In *CVPR*, 1591–1600. (cited on page 164)

Simon, C.; Koniusz, P.; Nock, R.; and Harandi, M., 2020a. Adaptive subspaces for few-shot learning. *CVPR*, (2020). (cited on page 89)

Simon, C.; Koniusz, P.; Nock, R.; and Harandi, M., 2020b. On modulating the gradient for meta-learning. *ECCV*, (2020). (cited on pages 12, 89, and 164)

Simonyan, K. and Zisserman, A., 2014. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 568–576. http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf. (cited on pages 4, 5, 43, 47, 63, 68, 86, 102, 103, and 104)

Simonyan, K. and Zisserman, A., 2015. Very Deep Convolutional Networks for Large-scale Image Recognition. *ICLR*, (2015), 1–14. (cited on page 26)

Sivic, J. and Zisserman, A., 2003. Video Google: A text retrieval approach to object matching in videos. *ICCV*, 2 (2003), 1470–1477. (cited on pages 43, 46, 48, 64, and 67)

Smola, A. J. and Kondor, R., 2003. Kernels and regularization on graphs. *COLT*, (2003). (cited on page 165)

Snell, J.; Swersky, K.; and Zemel, R. S., 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 4077–4087. (cited on pages 12, 154, 164, 177, 178, and 186)

Song, S.; Lan, C.; Xing, J.; Zeng, W.; and Liu, J., 2017. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17 (San Francisco, California, USA, 2017), 4263–4270. AAAI Press. (cited on page 11)

Song, Y.-F.; Zhang, Z.; Shan, C.; and Wang, L., 2022. Constructing stronger and faster baselines for skeleton-based action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2022), 1–1. doi:10.1109/TPAMI.2022.3157033. (cited on pages 1, 8, and 121)

Su, B. and Hua, G., 2019. Order-preserving optimal transport for distances between sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41, 12 (2019), 2961–2974. doi:10.1109/TPAMI.2018.2870154. (cited on pages 145 and 146)

Su, B. and Wen, J.-R., 2022. Temporal alignment prediction for supervised representation learning and few-shot sequence classification. In *International Conference on Learning Representations*. (cited on pages 1, 16, 145, 146, 154, 161, 177, and 178)

Su, B.; Zhou, J.; and Wu, Y., 2019. Order-preserving wasserstein discriminant analysis. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9884–9893. doi:10.1109/ICCV.2019.00998. (cited on page 145)

Sun, K.; Koniusz, P.; and Wang, Z., 2019. Fisher-Bures adversary graph convolutional networks. *UAI*, 115 (2019), 465–475. (cited on pages 89, 160, and 165)

Sun, L.; Jia, K.; Yeung, D.-Y.; and Shi, B. E., 2015. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*. (cited on page 6)

Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H. S.; and Hospedales, T. M., 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*, 1199–1208. (cited on pages 12 and 164)

Sung, J.; Ponce, C.; Selman, B.; and Saxena, A., 2011. Human Activity Detection from RGBD Images. *PAIR*, (2011). (cited on pages 2, 3, 21, and 27)

Sung, J.; Ponce, C.; Selman, B.; and Saxena, A., 2012. Unstructured Human Activity Detection from RGBD Images. In *ICRA*. (cited on page 21)

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A., 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17 (San Francisco, California, USA, 2017), 4278–4284. AAAI Press. http://dl.acm.org/citation.cfm?id=3298023.3298188. (cited on pages 65, 67, and 68)

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 65 and 67)

Tan, M. and Le, Q., 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, 6105–6114. PMLR. https://proceedings.mlr.press/v97/tan19a.html. (cited on page 191)

Tan, S. and Yang, R., 2019. Learning similarity: Feature-aligning network for few-shot action recognition. In *International Joint Conference on Neural Networks (IJCNN)*, 1–7. (cited on page 146)

Tanfous, A. B.; Drira, H.; and Amor, B. B., 2018. Coding Kendall's Shape Trajectories for 3D Action Recognition. In *CVPR*, 2840–2849. (cited on pages 21, 22, 23, 107, and 108)

Tang, L.; Wertheimer, D.; and Hariharan, B., 2020. Revisiting pose-normalization for fine-grained few-shot recognition. In *CVPR*. (cited on pages 12 and 164)

Tang, Y.; Ma, L.; and Zhou, L., 2019. Hallucinating optical flow features for video classification. In *IJCAI*, 926–932. IJCAI, Macao, China. (cited on page 64)

Tang, Y.; Tian, Y.; Lu, J.; Li, P.; and Zhou, J., 2018. Deep Progressive Reinforcement Learning for Skeleton-based Action Recognition. In *CVPR*, 1–10. (cited on pages 21 and 22)

Tas, Y. and Koniusz, P., 2018. CNN-based Action Recognition and Supervised Domain Adaptation on 3D Body Skeletons via Kernel Feature Maps. In *BMVC*. (cited on pages 23, 61, 107, 109, 160, and 186)

Thrun, S., 1996. Is learning the n-th thing any easier than learning the first? *NIPS*, (1996), 640–646. (cited on page 61)

Tong, Z.; Song, Y.; Wang, J.; and Wang, L., 2022. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=AhccnBXSne. (cited on pages 7 and 124)

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M., 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. *ICCV*, (2015), 4489–4497. (cited on pages 4, 5, 6, 43, 47, 63, 68, 86, and 163)

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M., 2018. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*. (cited on pages 6 and 86)

Truong, T.-D.; Bui, Q.-H.; Duong, C. N.; Seo, H.-S.; Phung, S. L.; Li, X.; and Luu, K., 2022. Direcformer: A directed attention in transformer approach to robust action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 20030–20040. (cited on page 7)

Turaga, P. and Chellappa, R., 2009. Locally time-invariant models of human activities using trajectories on the grassmannian. *CVPR*, (2009). (cited on page 86)

Uijlings, J. R.; Duta, I. C.; Rostamzadeh, N.; and Sebe, N., 2014. Realtime Video Classification using Dense HOF/HOG. *ICMR*, (2014). (cited on pages 45 and 66)

van Gemert, J. C.; Veenman, C. J.; Smeulders, A. W. M.; and Geusebroek, J.-M., 2010. Visual word ambiguity. *TPAMI*, 32, 7 (Jul. 2010), 1271–1283. doi:10.1109/TPAMI.2009.132. http://dx.doi.org/10.1109/TPAMI.2009.132. (cited on pages 46 and 67)

Varol, G.; Laptev, I.; and Schmid, C., 2018. Long-term temporal convolutions for action recognition. *TPAMI*, 40, 6 (2018), 1510–1517. (cited on pages 47 and 68)

Vasilescu, M. A. and Terzopoulos, D., 2002. Multilinear analysis of image ensembles: Tensorfaces. *ECCV*, (2002). (cited on page 88)

Vasilescu, M. A. O. and Terzopoulos, D., 2004. TensorTextures: Multilinear image-based rendering. *ACM Transactions on Graphics*, 23, 3 (2004), 336–342. (cited on page 88)

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I., 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. https://proceedings.

neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. (cited on pages 124, 125, 127, and 132)

Veeriah, V.; Zhuang, N.; and Qi, G.-J., 2015. Differential Recurrent Neural Networks for Action Recognition. *CVPR*, (2015), 1–9. (cited on pages 22 and 23)

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y., 2018. Graph attention networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ. (cited on page 124)

Vemulapalli, R.; Arrate, F.; and Chellappa, R., 2014. Human action recognition by representing 3D skeletons as points in a Lie Group. *CVPR*, (2014), 588–595. doi:http://doi.ieeecomputersociety.org/10.1109/CVPR.2014.82. (cited on pages 3, 7, 11, 20, 22, 23, 25, 86, 88, 103, 107, 108, 109, and 113)

Vemulapalli, R. and Chellappa, R., 2016. Rolling Rotations for Recognizing Human Actions from 3D Skeletal Data. *CVPR*, (2016), 4471–4479. (cited on pages 3, 4, 7, 11, 20, 22, 23, 24, 25, 26, 29, 33, 35, 36, and 38)

Villani, C., 2009. *Optimal Transport, Old and New*. Springer. (cited on page 165)

Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; and Wierstra, D., 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 3630–3638. (cited on pages 12, 154, 164, 177, and 186)

Vyas, S.; Rawat, Y. S.; and Shah, M., 2020. Multi-view action recognition using cross-view video prediction. In *Computer Vision – ECCV 2020*, 427–444. Springer International Publishing, Cham. (cited on pages 10 and 11)

Wang, C.; Wang, Y.; and Yuille, A. L., 2013a. An approach to pose-based action recognition. *CVPR*, (2013). (cited on pages 86 and 89)

Wang, G.; Ye, H.; Wang, X.; Ye, W.; and Wang, H., 2021a. Temporal relation based attentive prototype network for few-shot action recognition. In *Proceedings of The 13th Asian Conference on Machine Learning*, vol. 157 of *Proceedings of Machine Learning Research*, 406–421. PMLR. https://proceedings.mlr.press/v157/wang21b.html. (cited on page 13)

Wang, H.; Kläser, A.; Schmid, C.; and Cheng-Lin, L., 2011. Action Recognition by Dense Trajectories. *CVPR*, (2011), 3169–3176. (cited on pages 5, 43, 45, 46, 63, 66, 67, and 104)

Wang, H.; Kläser, A.; Schmid, C.; and Liu, C.-L., 2013b. Dense Trajectories and Motion Boundary Descriptors for Action Recognition. *IJCV*, (2013). (cited on pages 5, 43, 44, 45, 46, 63, 66, and 67)

Wang, H. and Schmid, C., 2013. Action Recognition with Improved Trajectories. *ICCV*, (2013), 3551–3558. (cited on pages 10, 43, 44, 45, 51, 63, 64, and 66)

Wang, H. and Wang, L., 2017. Modeling Temporal Dynamics and Spatial Configurations of Actions Using Two-Stream Recurrent Neural Networks. In *CVPR*, 1–10. (cited on pages 21 and 22)

Wang, H. and Wang, L., 2018. Beyond Joints: Learning Representations From Primitive Geometries for Skeleton-Based Action Recognition and Detection. In *TIP*, 4382–4394. (cited on pages 21, 22, and 23)

Wang, J. and Cherian, A., 2018. Learning discriminative video representations using adversarial perturbations. In *ECCV*, 716–733. doi:10.1007/978-3-030-01225-0_42. https://doi.org/10.1007/978-3-030-01225-0_42. (cited on pages 10, 44, 47, 57, 58, 64, 65, 68, 77, 78, 109, and 111)

Wang, J.; Liu, Z.; Wu, Y.; and Yuan, J., 2012. Mining Actionlet Ensemble for Action Recognition with Depth Cameras. *CVPR*, (2012), 1290–1297. (cited on pages 11, 21, 22, 25, 88, 108, and 109)

Wang, J.; Nie, X.; Xia, Y.; Wu, Y.; and Zhu, S.-C., 2014. Cross-view action modeling, learning and recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2649–2656. (cited on page 131)

Wang, L., 2017. *Analysis and Evaluation of Kinect-based Action Recognition Algorithms*. Master's thesis, School of the Computer Science and Software Engineering, The University of Western Australia. (cited on pages 3, 4, 10, 14, 30, 33, 35, 36, 38, 45, 63, 119, 120, 146, 163, and 174)

Wang, L.; Ding, Z.; Tao, Z.; Liu, Y.; and Fu, Y., 2019a. Generative multi-view human action recognition. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on pages 10 and 11)

Wang, L.; Huynh, D. Q.; and Koniusz, P., 2019b. A comparative review of recent kinect-based action recognition algorithms. *TIP*, (2019). doi:10.1109/TIP.2019.2925285. (cited on pages 3, 4, 7, 9, 10, 14, 45, 63, 119, 120, 146, 163, 167, and 174)

Wang, L.; Huynh, D. Q.; and Mansour, M. R., 2019c. Loss switching fusion with similarity search for video classification. *ICIP*, (2019). (cited on pages 46, 63, 119, 146, and 163)

Wang, L. and Koniusz, P., 2021. *Self-Supervising Action Recognition by Statistical Moment and Subspace Descriptors*, 4324–4333. Association for Computing Machinery, New York, NY, USA. ISBN 9781450386517. https://doi.org/10.1145/3474085.3475572. (cited on pages 9, 10, 89, 119, 146, and 163)

Wang, L. and Koniusz, P., 2022a. Temporal-viewpoint transportation plan for skeletal few-shot action recognition. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 4176–4193. (cited on pages 1, 11, 119, and 120)

Wang, L. and Koniusz, P., 2022b. Uncertainty-dtw for time series and sequences. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI*, 176–195. Springer. (cited on pages 1, 12, 119, 120, and 164)

Wang, L. and Koniusz, P., 2023. 3mformer: Multi-order multi-mode transformer for skeletal action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5620–5631. (cited on page 1)

Wang, L.; Koniusz, P.; and Huynh, D. Q., 2019d. Hallucinating IDT descriptors and I3D optical flow features for action recognition with cnns. In *ICCV*. (cited on pages xix, 9, 10, 64, 65, 66, 67, 68, 69, 72, 76, 77, 78, 79, 89, 119, 146, 163, and 167)

Wang, L.; Liu, J.; and Koniusz, P., 2021b. 3d skeleton-based few-shot action recognition with jeanie is not so naïve. *arXiv preprint arXiv:2112.12668*, (2021). (cited on pages 119 and 120)

Wang, L.; Wang, L.; Lu, H.; Zhang, P.; and Ruan, X., 2016a. Saliency detection with recurrent fully convolutional networks. In *ECCV*, 825–841. Springer Science+Business Media, Amsterdam, The Netherlands. doi:10.1007/978-3-319-46493-0_50. (cited on page 68)

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Gool, L. V., 2016b. Temporal segment networks: Towards good practices for deep action recognition. *ECCV*, (2016). (cited on page 86)

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L., 2019. Temporal segment networks for action recognition in videos. *TPAMI*, 41, 11 (11 2019), 2740–2755. doi:10.1109/TPAMI.2018.2868668. (cited on pages 6 and 165)

Wang, L.; Zhang, J.; Zhou, L.; Tang, C.; and Li, W., 2015a. Beyond covariance: Feature representation with nonlinear kernel matrices. *ICCV*, (2015). (cited on pages 88 and 108)

Wang, P.; Li, W.; Gao, Z.; Zhang, J.; Tang, C.; and Ogunbona, P., 2015b. Deep Convolutional Neural Networks for Action Recognition Using Depth Map Sequences. In *CVPR*, 1–8. (cited on pages 21 and 22)

Wang, P.; Li, W.; Gao, Z.; Zhang, J.; Tang, C.; and Ogunbona, P. O., 2016c. Action Recognition From Depth Maps Using Deep Convolutional Neural Networks. *IEEE T HUM-MACH SYST*, (2016), 498–509. (cited on pages 21 and 22)

Wang, Q.; Gao, Z.; Xie, J.; Zuo, W.; and Li, P., 2018a. Global gated mixture of second-order pooling for improving deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/17c276c8e723eb46aef576537e9d56d0-Paper.pdf. (cited on pages 121 and 129)

Wang, S.; Yue, J.; Liu, J.; Tian, Q.; and Wang, M., 2020. Large-scale few-shot learning via multi-modal knowledge discovery. In *ECCV*. (cited on pages 12 and 164)

Wang, X.; Girshick, R.; Gupta, A.; and He, K., 2018b. Non-local neural networks. In *CVPR*. (cited on page 5)

Wang, Y.; Li, K.; Li, Y.; He, Y.; Huang, B.; Zhao, Z.; Zhang, H.; Xu, J.; Liu, Y.; Wang, Z.; Xing, S.; Chen, G.; Pan, J.; Yu, J.; Wang, Y.; Wang, L.; and Qiao, Y., 2022. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, (2022). (cited on page 7)

Wang, Y.; Long, M.; Wang, J.; and Yu, P. S., 2017. Spatiotemporal pyramid network for video action recognition. In *CVPR*. (cited on pages 6 and 163)

Wei, C.; Fan, H.; Xie, S.; Wu, C.; Yuille, A. L.; and Feichtenhofer, C., 2021a. Masked feature prediction for self-supervised visual pre-training. *CoRR*, abs/2112.09133 (2021). https://arxiv.org/abs/2112.09133. (cited on page 124)

Wei, J.; Wang, Y.; Guo, M.; Lv, P.; Yang, X.; and Xu, M., 2021b. Dynamic hypergraph convolutional networks for skeleton-based action recognition. *CoRR*, abs/2112.10570 (2021). https://arxiv.org/abs/2112.10570. (cited on pages 1, 133, and 134)

Wei, S.-E.; Ramakrishna, V.; Kanade, T.; and Sheikh, Y., 2016. Convolutional pose machines. *CVPR*, (2016). (cited on page 89)

Wei, X.; Zhang, T.; Li, Y.; Zhang, Y.; and Wu, F., 2020. Multi-modality cross attention network for image and sentence matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 124)

Weinberger, K.; Dasgupta, A.; Langford, J.; Smola, A.; and Attenberg, J., 2009. Feature hashing for large scale multitask learning. In *ICML*, 1113–1120. doi:10.1145/1553374.1553516. http://doi.acm.org/10.1145/1553374.1553516. (cited on pages 48 and 50)

Weinland, D.; Özuysal, M.; and Fua, P., 2010. Making action recognition robust to occlusions and viewpoint changes. In *Computer Vision – ECCV 2010*, 635–648. Springer Berlin Heidelberg, Berlin, Heidelberg. (cited on pages 10 and 11)

Weinzaepfel, P.; Revaud, J.; Harchaoui, Z.; and Schmid, C., 2013. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*. http://hal.inria.fr/hal-00873592. (cited on pages 47 and 67)

Willems, G.; Tuytelaars, T.; and Gool, L. V., 2008. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV* (Marseille, France, 2008), 650–663. doi:10.1007/978-3-540-88688-4_48. https://doi.org/10.1007/978-3-540-88688-4_48. (cited on pages 45, 46, 66, and 67)

Wu, C.-Y.; Feichtenhofer, C.; Fan, H.; He, K.; Krahenbuhl, P.; and Girshick, R., 2019a. Long-term feature banks for detailed video understanding. In *CVPR*. (cited on pages 59 and 79)

Wu, F.; Zhang, T.; de Souza Jr., A. H.; Fifty, C.; Yu, T.; and Weinberger, K. Q., 2019b. Simplifying graph convolutional networks. In *ICML*. (cited on pages 165, 167, 168, and 171)

Wu, W.; Sun, Z.; and Ouyang, W., 2023. Revisiting classifier: Transferring vision-language models for video recognition. (2023). (cited on pages 7, 9, and 10)

Wu, X. and Jia, Y., 2012. View-invariant action recognition using latent kernelized structural svm. In *Computer Vision – ECCV 2012*, 411–424. Springer Berlin Heidelberg, Berlin, Heidelberg. (cited on page 10)

Wu, X.; Wang, H.; Liu, C.; and Jia, Y., 2015. Cross-view action recognition over heterogeneous feature spaces. *IEEE Transactions on Image Processing*, 24, 11 (2015), 4096–4108. doi:10.1109/TIP.2015.2445293. (cited on page 11)

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y., 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32, 1 (2020), 4–24. (cited on page 8)

Xia, L. and Aggarwal, J. K., 2013. Spatio-Temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera. In *CVPR*, 2834–2841. (cited on pages 21 and 22)

Xia, L.; Chen, C.-C.; and Aggarwal, J. K., 2012. View invariant human action recognition using histograms of 3D joints. *CVPR Workshops*, (2012), 20–27. (cited on pages 2, 11, 22, 23, 87, 102, and 107)

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K., 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*. (cited on pages 5 and 6)

Xu, B.; Ye, H.; Zheng, Y.; Wang, H.; Luwang, T.; and Jiang, Y.-G., 2018. Dense dilated network for few shot action recognition. In *ACM ICMR*, 379–387. (cited on pages 12, 164, and 166)

Yacoob, Y. and Black, M. J., 1998. Parameterized modeling and recognition of activities. *ICCV*, (1998), 120–128. (cited on page 88)

Yan, A.; Wang, Y.; Li, Z.; and Qiao, Y., 2019. PA3D: Pose-action 3D machine for video recognition. In *CVPR*. (cited on page 77)

Yan, S.; Xiong, Y.; and Lin, D., 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*. (cited on pages xxix, 2, 4, 8, 21, 23, 24, 26, 27, 30, 31, 33, 35, 36, 38, 89, 104, 109, 110, 112, 120, 122, 124, 126, 130, 133, 134, 150, 166, 174, and 182)

Yang, C.-H. H.; Tsai, Y.-Y.; and Chen, P.-Y., 2021. Voice2series: Reprogramming acoustic models for time series classification. In *Proceedings of the 38th International Conference on Machine Learning*, vol. 139 of *Proceedings of Machine Learning Research*, 11808–11819. PMLR. (cited on page 145)

Yang, J.; Dong, X.; Liu, L.; Zhang, C.; Shen, J.; and Yu, D., 2022a. Recurring the transformer for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14063–14073. (cited on page 7)

Yang, L.; Huang, Y.; Sugano, Y.; and Sato, Y., 2022b. Interact before align: Leveraging cross-modal knowledge for domain adaptive action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14722–14732. (cited on page 10)

Yang, X. and Tian, Y., 2012. EigenJoints-based Action Recognition Using Naive-Bayes-Nearest-Neighbor. In *CVPR*, 14–19. (cited on pages 3, 20, 21, 22, and 23)

Yang, X. and Tian, Y., 2014a. Effective 3D action recognition using eigenjoints. *J. Vis. Comun. Image Represent.*, 25, 1 (2014), 2–11. doi:10.1016/j.jvcir.2013.03.001. (cited on page 88)

Yang, X. and Tian, Y., 2014b. Super Normal Vector for Activity Recognition using Depth Sequences. In *CVPR*, 804–811. (cited on pages 21 and 22)

Yeffet, L. and Wolf, L., 2009. Local trinary patterns for human action recognition. *ICCV*, (2009), 492–497. (cited on pages 46 and 67)

Yu, X.; Zhuang, Z.; Koniusz, P.; and Li, H., 2020. 6DoF object pose estimation via differentiable proxy voting regularizer. In *BMVC*. BMVA Press. (cited on pages 12 and 164)

Zanfir, M.; Leordeanu, M.; and Sminchisescu, C., 2013. The moving pose: An efficient 3D kinematics descriptor for low-latency action recognition and detection. *ICCV*, (2013). doi:10.1109/ICCV.2013.342. (cited on page 108)

Zatsiorsky, V. M., 1997. Kinematic of human motion. *Human Kinetics Publishers*, (1997). (cited on page 88)

Zhang, C.; Tian, Y.; Guo, X.; and Liu, J., 2017a. DAAL: Deep Activation-based Attribute Learning for Action Recognition in Depth Videos. *CVIU*, (2017), 37–49. (cited on page 22)

Zhang, C.; Zou, Y.; Chen, G.; and Gan, L., 2019a. PAN: persistent appearance network with an efficient motion cue for fast action recognition. In *MM*, 500–509. ACM, Nice, France. doi:10.1145/3343031.3350876. https://doi.org/10.1145/3343031.3350876. (cited on page 68)

Zhang, H. and Koniusz, P., 2019. Power normalizing second-order similarity network for few-shot learning. In *WACV*, 1185–1193. (cited on pages 89 and 164)

Zhang, H.; Koniusz, P.; Jian, S.; Li, H.; and Torr, P. H. S., 2021a. Rethinking class relations: Absolute-relative supervised and unsupervised few-shot learning. In *CVPR*, 9432–9441. (cited on pages 12 and 164)

Zhang, H.; Li, H.; and Koniusz, P., 2022a. Multi-level second-order few-shot learning. *IEEE Transactions on Multimedia*, (2022). (cited on pages 12 and 164)

Zhang, H.; Song, Y.; and Zhang, Y., 2019b. Graph convolutional lstm model for skeleton-based action recognition. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 412–417. doi:10.1109/ICME.2019.00078. (cited on page 120)

Zhang, H.; Zhang, J.; and Koniusz, P., 2019c. Few-shot learning via saliency-guided hallucination of samples. In *CVPR*, 2770–2779. IEEE, Long Beach California. (cited on pages 65 and 68)

Zhang, H.; Zhang, L.; Qi, X.; Li, H.; Torr, P.; and Koniusz, P., 2020a. Few-shot action recognition with permutation-invariant attention. In *European Conference on Computer Vision (ECCV)*. (cited on pages 12, 13, 15, 89, 146, 147, 164, and 166)

Zhang, J.; Shi, X.; Xie, J.; Ma, H.; King, I.; and Yeung, D.-Y., 2018a. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*, 339–349. AUAI Press. (cited on page 124)

Zhang, J.; Shum, H. P. H.; Han, J.; and Shao, L., 2018b. Action Recognition From Arbitrary Views Using Transferable Dictionary Learning. In *TIP*, 4709–4723. (cited on pages 20 and 22)

Zhang, J.; Wang, L.; and Zhou, L., 2020b. Beyond covariance: Sice and kernel based visual feature representation. *IJCV*, (2020). doi:10.1007/s11263-020-01376-1. (cited on page 88)

Zhang, J.; Xie, W.; Wang, C.; Tu, R.; and Tu, R., 2022b. Graph-aware transformer for skeleton-based action recognition. *The Visual Computer*, (2022). (cited on page 8)

Zhang, J.; Zhang, T.; Dai, Y.; Harandi, M.; and Hartley, R., 2018c. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *CVPR*, 1–10. IEEE, Salt Lake City,UT,USA. (cited on pages 65 and 68)

Zhang, P.; Lan, C.; Xing, J.; Zeng, W.; Xue, J.; and Zheng, N., 2017b. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *ICCV*. (cited on pages xxxi, 10, 109, 164, 167, and 178)

Zhang, P.; Lan, C.; Xing, J.; Zeng, W.; Xue, J.; and Zheng, N., 2019d. View adaptive neural networks for high performance skeleton-based human action recognition. *IEEE TPAMI*, 41, 8 (2019), 1963–1978. (cited on pages xxxi, 10, 11, 164, 167, and 178)

Zhang, P.; Lan, C.; Zeng, W.; Xing, J.; Xue, J.; and Zheng, N., 2020c. Semantics-guided neural networks for efficient skeleton-based human action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 8 and 121)

Zhang, Q.; Wang, T.; Zhang, M.; Liu, K.; Shi, P.; and Snoussi, H., 2021b. Spatial-temporal transformer for skeleton-based action recognition. In *2021 China Automation Congress (CAC)*, 7029–7034. doi:10.1109/CAC53003.2021.9728206. (cited on page 8)

Zhang, S.; Luo, D.; Wang, L.; and Koniusz, P., 2020d. Few-shot object detection by second-order pooling. In *ACCV*, vol. 12625 of *Lecture Notes in Computer Science*, 369–387. Springer. (cited on pages 89, 121, and 164)

Zhang, S.; Murray, N.; Wang, L.; and Koniusz, P., 2022c. Time-rEversed diffusioN tEnsor Transformer: A new TENET of Few-Shot Object Detection. In *ECCV*. (cited on pages 12, 121, and 164)

Zhang, S.; Wang, L.; Murray, N.; and Koniusz, P., 2022d. Kernelized few-shot object detection with efficient integral aggregation. In *CVPR*, 19207–19216. (cited on pages 12, 121, and 164)

Zhang, S.; Zhou, J.; and He, X., 2021c. Learning implicit temporal alignment for few-shot video classification. *CoRR*, abs/2105.04823 (2021). https://arxiv.org/abs/2105.04823. (cited on page 13)

Zhang, X.; Xu, C.; and Tao, D., 2020e. Context aware graph convolution for skeleton-based action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 8 and 121)

Zhang, Y.; Li, J.; Wu, G.; Zhang, H.; Shi, Z.; Liu, Z.; and Wu, Z., 2021d. Temporal transformer networks with self-supervision for action recognition. *CoRR*, abs/2112.07338 (2021). https://arxiv.org/abs/2112.07338. (cited on pages 1 and 124)

Zhang, Y.; Wu, B.; Li, W.; Duan, L.; and Gan, C., 2021e. Stst: Spatial-temporal specialized transformer for skeleton-based action recognition. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21 (Virtual Event, China, 2021), 3229–3237. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3474085.3475473. https://doi.org/10.1145/3474085.3475473. (cited on pages 8 and 133)

Zhang, Y.; Zhu, H.; Meng, Z.; Koniusz, P.; and King, I., 2022e. Graph-adaptive rectified linear unit for graph neural networks. In *Proceedings of the ACM Web Conference 2022*, WWW '22 (Virtual Event, Lyon, France, 2022), 1331–1339. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3485447.3512159. (cited on page 160)

Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; and King, I., 2022f. Costa: Covariance-preserving feature augmentation for graph contrastive learning. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, (2022). https://doi.org/10.1145/3534678.3539425. (cited on pages 122 and 160)

Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; and King, I., 2023. Spectral feature augmentation for graph contrastive learning and beyond. In *AAAI*. (cited on page 122)

Zhang, Z., 2012. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19, 2 (2012), 4–10. doi:10.1109/MMUL.2012.24. (cited on page 2)

Zhang, Z.; Wang, C.; Xiao, B.; Zhou, W.; Liu, S.; and Shi, C., 2013. Cross-view action recognition via a continuous virtual path. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2690–2697. doi:10.1109/CVPR.2013.347. (cited on pages 10 and 11)

Zhao, X.; Wang, S.; Li, S.; and Li, J., 2012. A comprehensive study on third order statistical features for image splicing detection. In *Digital Forensics and Watermarking*, 243–256. (cited on page 89)

Zheng, N.; Wen, J.; Liu, R.; Long, L.; Dai, J.; and Gong, Z., 2018. Unsupervised Representation Learning with Long-Term Dynamics for Skeleton Based Action Recognition. In *AAAI*, 2644–2651. (cited on pages 21, 22, and 23)

Zhou, Y.; Li, C.; Cheng, Z.-Q.; Geng, Y.; Xie, X.; and Keuper, M., 2022. Hypergraph transformer for skeleton-based action recognition. *arXiv preprint arXiv:2211.09590*, (2022). (cited on page 8)

Zhou, Y.; Ni, B.; Hong, R.; Wang, M.; and Tian, Q., 2015. Interaction part mining: A mid-level approach for fine-grained action recognition. *CVPR*, (2015). (cited on page 89)

Zhu, A.; Ke, Q.; Gong, M.; and Bailey, J., 2023. Adaptive local-component-aware graph convolutional network for one-shot skeleton-based action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 6038–6047. (cited on pages 1 and 15)

Zhu, H. and Koniusz, P., 2021a. REFINE: Random RangE FInder for network embedding. *CIKM*, (2021). (cited on page 89)

Zhu, H. and Koniusz, P., 2021b. Simple spectral graph convolution. In *International Conference on Learning Representations (ICLR)*. (cited on pages 89, 147, 165, 167, 168, and 171)

Zhu, H. and Koniusz, P., 2021c. Simple spectral graph convolution. In *International Conference on Learning Representations*. (cited on page 120)

Zhu, H. and Koniusz, P., 2022. EASE: Unsupervised discriminant subspace learning for transductive few-shot learning. *CVPR*, (2022). (cited on pages 12 and 164)

Zhu, H.; Sun, K.; and Koniusz, P., 2021a. Contrastive laplacian eigenmaps. *Advances in Neural Information Processing Systems*, 34 (2021). (cited on pages 120, 160, and 165)

Zhu, L.; Sevilla-Lara, L.; Tran, D.; Feiszli, M.; Yang, Y.; and Wang, H., 2019. FASTER recurrent networks for video classification. *CoRR*, abs/1906.04226 (2019). http://arxiv.org/abs/1906.04226. (cited on page 191)

Zhu, L. and Yang, Y., 2018. Compound memory networks for few-shot video classification. In *ECCV*. (cited on page 166)

Zhu, W.; Lan, C.; Xing, J.; Zeng, W.; Li, Y.; Shen, L.; and Xie, X., 2016. Co-occurrence Feature Learning for Skeleton based Action Recognition using Regularized Deep LSTM Networks. *AAAI*, (2016). (cited on pages 11, 22, and 23)

Zhu, W.; Liang, S.; Wei, Y.; and Sun, J., 2014. Saliency optimization from robust background detection. In *CVPR*, 2814–2821. IEEE, Columbus, OH, USA. doi: 10.1109/CVPR.2014.360. (cited on page 68)

Zhu, X.; Toisoul, A.; Pérez-Rúa, J.; Zhang, L.; Martínez, B.; and Xiang, T., 2021b. Few-shot action recognition with prototype-centered attentive learning. *CoRR*, abs/2101.08085 (2021). https://arxiv.org/abs/2101.08085. (cited on page 14)

Zhu, Y.; Huang, G.; Xu, X.; Ji, Y.; and Shen, F., 2022. Selective hypergraph convolutional networks for skeleton-based action recognition. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, ICMR '22 (Newark, NJ, USA, 2022), 518–526. Association for Computing Machinery, New York, NY, USA. doi: 10.1145/3512527.3531367. https://doi.org/10.1145/3512527.3531367. (cited on pages 1, 133, and 134)

Zilin, G.; Jiangtao, X.; Qilong, W.; and Peihua, L., 2019. Global second-order pooling convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 121 and 129)

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V., 2018. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 65 and 67)

Zuffi, S. and Black, M. J., 2013. Puppet flow. *IJCV*, 101, 3 (2013), 437–458. (cited on pages 86 and 89)