Visual Representation Learning with Progressive Data Scarcity

Hongtao Yang

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

August 2020

Preliminary version – 7 August 2020

Preliminary version – 7 August 2020

Except where otherwise indicated, this thesis is my own original work. This thesis has not been submitted by me in whole or part for a degree or diploma in any university or other tertiary education institution. The content of this thesis is mainly based on my publications listed below:

1. **(Chapter3, 4)** Yang, Hongtao & He, Xuming & Porikli, Fatih. Instance-Aware Detailed Action Labeling in Videos. IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, 2018, pp. 1577-1586.

2. **(Chapter 5)** Yang, Hongtao & He, Xuming & Porikli, Fatih. One-Shot Action localisation by Learning Sequence Matching Network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, Utah, 2018, pp. 1450-1459.

3. **(Chapter 6)** Yang, Hongtao & Tong, Zhang & Huang, Wenbing & He, Xuming & Porikli, Fatih. Towards Unsupervised Disentanglement of Appearance and Shape for Person Images Generation. https://arxiv.org/abs/2007.13098

Hongtao Yang
7 August 2020

To my wife, Tian Liang, my mom Hong Tao and my dad Jingyi Yang. I would like to take this opportunity to thanks my family. They were always there to support, encourage and listen whenever I felt depressed. Also, they were always there to cheer with me whenever I've achieved a goal. At the hardest time during my PhD, when I was both mentally and physically ill, they stayed with me and waited patiently for me to walk out and become stronger. This thesis and the life I have right now is not possible without you.

Acknowledgments

I would like to express my greatest appreciation to all my supervisors for their great efforts on my PhD projects. They are truly supportive at times when help is really needed. Without their devotion, insight suggestions and advises, it is not possible for me to complete my PhD projects.

I would like to thank my primary supervisor, Xuming He, for his devotion on my projects. In particular, his careful revision on my drafts not only helped to elevate the paper, but also taught me, on a philosophical level, how to structure my ideas, plan my stories and ultimately present my work. At times, we would even exchange results and thoughts late into the night. His insights on machine learning theories never cease to inspire me of new ideas, upon witch my project was built. I would also like to thank him for giving me great respect and freedom to select whatever topics that I am interested in, and his ability to provides insights on a broad range of topics.

I am also grateful to Fatih Porikli, my panel chair. He provided me with great supports and courage to carry on, even during my hardest times. His help was always timely and highly effective, whether its about hardware resources, high-level guide lines, thesis revision etc. A few words from him would always set me back to the correct path. I would like to thank my advisor Hongdong Li. As the only supervisor in my panel who are in ANU, He helped take care of my academic schedules and many other administrative issues, which takes up a lot of efforts. He always trust me with my research quality. In fact, my path of pursuing a PhD was inspired by him just because I took his computer vision course during my undergraduate study.

This research was supported by the scholarship provided by CSIRO. CSIRO provided the supplementary scholarship, comfortable office environment, and necessary hardware.

Abstract

One of the key advantages of supervised deep learning over conventional machine learning is that the learning process of a neural network does not rely directly on hand-crafted features. There is no denying that the availability of large-scale annotated data plays a vital role in the success of deep learning, among many other others.

On the other hand, given the capacity of deep neural networks, modern deep learning system is able to approximate complex functions and fit complex distributions of data of all kinds. As a result, the generalisability of a trained deep neural network is closely tied with the quality of its training data and annotation. Thus, the application and success of deep learning system are restrained by two main factors:

- High quality annotation of diverse data is expensive to obtain, both in terms of time and labour.
- Human annotations are inherently biased, leading to over-fitting and learning sub-optimal features.

It would be nice if we can 1. extract more useful information with existing valuable annotations; 2. achieve similar performance with weaker level of supervision, such as less amount of annotated data or indirect supervision from implicit annotations that is cheap to obtain.

To this end, our line of works widely explores various computer vision tasks, including action recognition, action localisation, disentangled representation learning and novel image generation, across different levels of supervision distinguished by data scarcity. Our journey begins with fully supervised action localisation with the aim to better utilise precious annotations. Then we investigate one-shot action recognition and localisation that aims to learn robust visual descriptors from very few annotated samples. Finally, we move from discriminative learning to generative learning with the purpose of exploring unsupervised feature disentanglement and image generation. In particular, our works can be divided into 4 parts:

We designed and constructed a large-scale RGB-D video dataset of gym activities to facilitate fully supervised video understanding. Contrary to existing public datasets, all actions in our dataset have clearly defined boundaries, which contributes to much less noise in annotation and thus is better fitted for video representation learning.

We then propose an instance-aware video labeling framework that explore the synergy of instance-level information and frame-level information. In this endeavour, we aim to take full advantage of precious annotations and combine multi-level information to mitigate the inherent noise in temporal action localisation. Our fu-

sion method is proven to capture complementary information and better localisation performance is reported.

After that, we propose a one/few shot action localisation method leveraging meta-learning concepts, aiming to make better use of very rare and limited data. We designed a similarity-based action localisation network under the meta-learning framework that can learn transferable and class-agnostic video descriptors. We also designed structured representation for time series that takes into account the natural evolution of actions. As a result, we achieve state-of-the-art performance in both one/few shot action recognition and localisation.

Further stripping down the strength of supervision, we propose an unsupervised disentangled representation learning framework that can generate novel images with desired attributes. Through this unsupervised exploration, our network can learn structured meaningful features from only implicit cycle and GAN-based constraints. The reported results are comparable or even better than its fully-supervised counterparts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Thanks to the rapid advancement in recent years of machine learning, especially deep learning, we now live in a much more intelligent world where machines can achieve comparable or even better performance across many tasks in many disciplines, such as image classification [Simonyan and Zisserman, 2014b], artistic style transfer [Gatys et al., 2015], image translation [Isola et al., 2017; Zhu et al., 2017a], game of GO [Silver et al., 2016] etc.

While artificial intelligence first came to birth around late 1950s [Turing, 2009], recent success of deep learning applications, both in consumer level or enterprise level, was only made possible by the achievements in big data and big hardware that kept in pace with state-of-the-art models and algorithms. There is no denying that data plays a vital role, almost as important as algorithms, in the huge recent success of real-world machine learning applications.

1.1 Motivation

Although data are readily available in many areas of our daily life (people uploading photos and videos, medical imaging generated every day etc.), it has to be extensively annotated before it can be effectively utilised by most modern algorithms. The annotations process can be both time-consuming and labour intensive. For example, it takes about 30 secs for a person to annotated one image for the purpose of object detection, even with a handy annotation tool such as LabelImg[1]. Imagine how much resources it require to annotate hundreds of thousands of images just to build a dataset for training reliable object detectors. Thankfully, people have been dedicate much resources into dataset building and there exist many public large-scale dataset to support research activities.

However, despite the fact that many large-scale annotated datasets are being published almost every year, existing datasets that are built for research or academic purposes are often regarded insufficient in real world applications. Possible reasons include:

1. Domain Discrepancies. E.g. an action detector trained on large-scale action localisation datasets such as Thumos [Jiang et al., 2014; Soomro et al., 2012a]

[1] Open-source project at https://github.com/tzutalin/labelImg

or Activity Net [Fabian Caba Heilbron and Niebles, 2015] is not good enough when applied on surveillance footage, possibly because of different lighting conditions, view points resolution etc.

2. Subtle difference in tasks. E.g. in facial recognition, we have to constantly deal with new identities (unknown people) and it is desirable to be able to accurately classify a give face as unknown or one of the known identities. However, state-of-the-art algorithms like Face-Net [Schroff et al., 2015] or ArcFace [Deng et al., 2019] as well as benchmark datasets like VGGFace2 [Cao et al., 2018a] are not tailored to handle this real-world requirements.

3. However large the size of existing datasets are, human annotation is inherently biased, making our learning system to overfit or simply not learn anything useful. Take action recognition for example, it is very hard to annotate mutually exclusive classes. We might have *running* as one class and *playing basketball* as another. Clearly running is part of playing basketball. This is also one of the motivation of the creation of the GADD dataset introduced in Chapter 3.

As a result, cooperates and institutions still resort to building task-specific datasets when trying to deploy real-world applications. This poses the desire of learning from data with limited or zero annotations. To this end, we investigate visual representation learning on various tasks under the setup of data scarcity.

1.2 Contributions

In this thesis, we conducted a line of works that explores different visual representation learning problems under different data scarcity setups, starting from fully supervised learning that require large amount of annotated data, to one/few shot learning that aims to quickly adapt to new data by learning class-agnostic features, then finally to unsupervised learning that attempts to capture disentangled representations by looking at data alone. In this journey, we also explored different learning paradigm to investigate the impact of data scarcity on learning performance, including meta-learning, unsupervised transfer learning.

Specifically, we present four different endeavours that covers various visual tasks of action recognition, action localisation, disentangled representation learning and novel image generation.

1.2.1 GADD - RGBD Dataset for Fine-grained Action localisation

In this work, we build a diverse RGBD human action dataset called **Gym Activity Depth Dataset** (GADD) that lays the foundation for research on action recognition and localisation. We choose to adopt gym exercises as the basic action instances in this dataset because each exercise has a well-defined start and end point, which minimises the ambiguity in action localisation annotations. On the other hand, trying to recognise and localise gym exercises are very challenging tasks because they are

complex actions that comprise of many basic movements involving all parts of the body and possibly some object interaction as well (barbell, mat etc.). Different exercises can have the same basic movements as part of the actions, and they can only be distinguished by the order and the length of the movements. This poses the challenge of learning long-term temporal dependencies in order to accurately classify different action instances.

We build this dataset to accommodate many learning tasks. Apart from action recognition and localisation, it can be used to investigate different contribution of RGB and depth data, also their collective effective. Although the dataset comes with frame-level annotation, it is suitable for unsupervised learning tasks such as action discovery, frame prediction etc.

Compared to GADD, existing datasets either have simple/naive actions classes [Sung et al., 2011], or ambiguous action boundaries [Fabian Caba Heilbron and Niebles, 2015], or no multimodal data formats. We use GADD in our next work to explore the synergy between instance level action information and frame level labels that achieve state-of-the-art performance in action localisation.

1.2.2 Synergy of Global and Fine-grained Action Information.

In this work we explore fully supervised learning for action localisation, which utilise all available annotations and multimodal data to extract as much supervision as possible. Specifically, we utilise the provided annotation at two different levels

1. Action instance level. At this level, annotations are presented as frame blocks, meaning that for each action instance, a start and end points are given with all frames in the middle sharing the same label. Using object detection terminologies, this corresponds to bounding boxes.

2. Frame level. At this level, annotations are presented as a sequence of labels for each frame in the video. Using image segmentation terminologies, this corresponds to pixel labels.

With action instance level information, we can capture global dynamics and roughly determine the location and duration of each action in a video. However, errors occur more frequently at the action boundaries or when two instances happen very close to each other in time. On the other hand, learning with frame level supervision provides more fine-grained local information that can benefit prediction around action transition period. However, it yields more noisy prediction in the middle of an action or background. It is obvious that the supervision at this two level can compensate each other.

We took insight of the above observation and explore the synergy between the instance-level and frame-level information. As a result, the accuracy for both frame labeling and action detection are boosted, and state-of-the-art performance is reported. In addition, we designed a robust depth frame representation that combines surface normal vectors and dynamic images [Bilen et al., 2016a] to capture short-term spatial-temporal features.

1.2.3 Action localisation with Limited Data

In this work, we limit the availability of annotated data and address the challenging problem of one/few-shot action recognition and localisation. Specifically, we propose a sequence matching network architecture based on the meta-learning framework. Our insight is that in order to quickly adapt to new classes using only one sample, our learning system has to learn class-independent features while capturing the temporal structures of an action. To this end, our sequence matching network learns similarity features through meta-level supervision. During the process, class-specific feature are gradually discarded by the network because the network is fed with randomly chosen new classes for each iteration.

To capture complex structures and actions, we also design a ranking based video descriptor that learns structured representation of sequences. The insight is that along the evolution of an action in time, the network should become more and more confident at its predictions. With our proposed structured video representation, and the meta-level similarity feature, we achieve superior performance in one/few-shot action recognition.

Adapting the above action recognition framework to action localisation in a non-trivial task. For the task of action localisation, we compose a correlation matrix through sliding window which captures the temporal evolution of similarities of an unconstrained video. Then a new network in applied on the correlation matrix that generates frame level predictions. We note that the correlation matrix consists of only similarity scores, thus it is class-independent and any network learned from the matrix is able to quickly adapt to new classes.

1.2.4 Unsupervised Frame Synthesis by Feature Disentanglement

In this work, we further strip down the strength of supervision and weakly-supervised and unsupervised disentanglement and novel image generation. Specifically, we study the problem of human and face image generation.

For weakly supervised case, we aim to learn disentangled appearance representation given pose features using unpaired images. Weak supervision is provided by the given pose information, usually come in the form of joints coordinates. Because there are no supervision as to what the generated image should look like, we resort to cycle consistency and patch consistency as regularisation terms to train the network. When a pose from image A is paired with appearance from image B to generate a novel image, the generated image should retain the same pose and appearance cue from A and B respectively. Therefore, we propose to further extract appearance information from the generated image and pair it with the pose of B in order to reconstruct image B. In addition, we also apply patch consistency around local patches because they tend to stay consistent no matter how the pose changes. This combination enables us to learn disentangled appearance representation that can be paired with arbitrary poses to generate realistic novel images.

For unsupervised case, we no longer have pose information and aim to learn purely data-driven disentangled representation of both pose and appearance. We

observe that the predefined attribute-specific cues such as joints coordinates may carry undesired bias into the system, which will hinder the quality of the generated images. For example, the use of skeleton or joint coordinates cannot comprehensively describe the shape of a person. Such pose information lack important details about e.g., thickness of limbs, hair styles etc, resulting in stiff and unnatural looking generated persons [Esser et al., 2018]. Instead, we propose an end-to-end learning system that generate pose masks from an input image as shape descriptor, which are then applied back onto the image itself to obtain disentangled appearance features. Finally, the appearance and shape features are combined to reconstruct the input image. This self-supervised process does not require any annotation or additional attribute-specific cues, and the obtained representations are purely data-driven without biases. To accelerate training convergence, we use the pre-trained OpenPose [Cao et al., 2018b] net as the shape mask extractor and finetune its weights. We show that our end-to-end unsupervised framework is free of engineering tricks, and can generate novel image of higher quality.

1.3 Thesis Outline

In remainder of this thesis, we first provide a brief review of the background theories and techniques that are relevant to our works in Chapter 2, then we present our GADD dataset, instance-aware video understanding, one/few shot action localisation through correlation matrix and unsupervised image generation in Chapter 3, 4, 5 and 6 respectively. Finally, our conclusions are given in Chapter 7

Chapter 2

Background

In this chapter, we give introductions to the background theories and models employed in this thesis. In our works, we mainly utilise neural networks to develop our frameworks. We have touched a wide range of computer vision sub-disciplines in our line of works, and therefore we have employed many variations of neural networks, including CNN, RNN, attention-networks, auto-encoder, GAN etc.

Below, we review the fundamentals of neural networks for sequence analysis, generative models, meta learning for few-shot recognition and unsupervised disentangled representation learning.

2.1 Sequence Analysis

For a long time after we have marched into the deep learning era, the better choice for sequence modelling task was RNNs because of their great ability to capture temporal dependencies in sequential data. Several variants of RNNs such as LSTM [Hochreiter and Schmidhuber, 1997], GRU [Cho et al., 2014b] have been proposed to capture long term dependencies in sequences. Further improved by the introduction of attention mechanism [Bahdanau et al., 2014] that enhanced the modelling capability of long term dynamics, they have achieved state of art performances in sequence-to-sequence (seq2seq) modelling tasks.

Recently, deep learning practitioners have been using Temporal Convolutional Networks (TCN) Lea et al. [2017] to better model sequences. The term is a simple descriptive term to refer to a family of architectures. It eliminate some of the key drawbacks of RNN like gradient vanishing [Bengio et al., 1994; Hochreiter et al., 2001] and exploding [Sutskever et al., 2011].

2.1.1 RNN & Attention Networks

In feed-forward neural networks, information only flow forward. Neural networks where information are fed back to it self through recurrent connections are called recurrent neural networks (RNNs). RNNs consist of high dimensional hidden states connected with non-linear activation. These hidden states serve as memory of the network and the states at time t are conditioned on previous states. Therefore in the-

ory, the states at time t are the summary of all the states in previous time steps. This structure enables the RNNs to store, remember, and process past complex signals for long periods of time, which enable the modelling capability of sequential data for sequence recognition and prediction [Bengio et al., 1994].

However, challenges such as vanishing and exploding gradient prevent a normal RNN from effectively modelling long-term dependencies. Some improvements of RNN have been proposed over the years. We will review major ones like LSTM [Hochreiter and Schmidhuber, 1997] and attention LSTM [Bahdanau et al., 2014]. In the remainder of this section, we will first explain the fundamentals of vanilla RNN as well as LSTM, then we explain the attention mechanism, which are widely adopted and proven to be effective in capturing long time dynamics.

2.1.1.1 RNN & LSTM

We note that RNNs are not limited to only time series. They have been applied successfully on non-temporal sequence data [Baldi and Pollastri, 2003; Oord et al., 2016; Salimans et al., 2017]. At times, the sequences have implicit or undesired temporal aspect in many important applications of RNNs, such as our proposed sequence matching networks in Section 5.2.1.2.

The input to an RNN is sequence and its target/output can also be a sequence. An input sequence can be denoted as $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots \mathbf{x}^{(T)})$ where each data point $\mathbf{x}^{(t)}$ is a real valued vector. The inputs are connected to the hidden state via a weight matrix $\mathbf{W}_{IH}$. The hidden state has M hidden units $\mathbf{h}_t = (h_1, h_2, \dots h_M)$. The hidden state $\mathbf{h}_t$ are passed into the next time step via another weight matrix $\mathbf{W}_{HH}$ along with the next input. Formally, the hidden layer defines the state or memory of the network as:

$$\mathbf{t}_t = \sigma_h(\mathbf{o}_t), \tag{2.1}$$

where $\sigma_h(*)$ is the hidden layer activation function, and

$$\mathbf{o}_t = \mathbf{W}_{IH}\mathbf{x}_t + \mathbf{W}_{HH}\mathbf{h}_{t-1} + \mathbf{b}_h \tag{2.2}$$

$\mathbf{b}_h$ is the bias of the hidden layer. Some common choices of σ include *ReLU*, *Tanh* and *Softmax* etc.

Similarly, the outputs are connect to the hidden state through a weight matrix and an activation function as well:

$$\mathbf{y}_t = \sigma_o(\mathbf{W}_{HO}\mathbf{h}_t + \mathbf{b}_o) \tag{2.3}$$

We can see from the above equations that hidden state summaries all the necessary information about the past states of the network over its entire history. This integrated information can define future behaviour of the network and make appropriate predictions at the output layer based on the past context.

Figure 2.1 gives an overview of a simple RNN layer and how it can be unfolded through time. Although the unfolding of RNN is only a visualisation technique, it

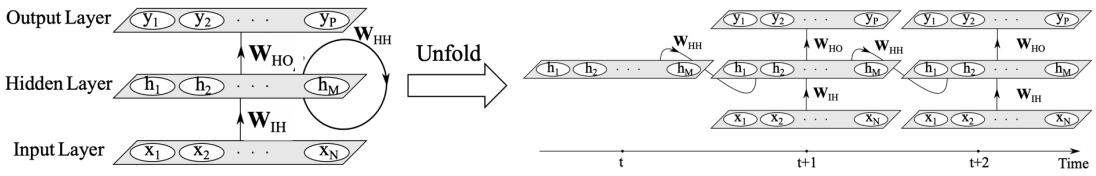


Figure 2.1: Overview of a simple RNN and its unfolded structure through time.

actually provides insights about training an RNN through Back Propagation (BP). As shown in the figure, a RNN network across multiple time steps can be interpreted not just as cyclic, but rather as a deep network with one layer per time step with shared hidden states across time steps. It is made though unfolding visualisation that the unfolded network can be trained across many time steps using BP. The algorithm hence got the name Back Propagation Through Time (BPTT), and it was first introduced by [Werbos et al., 1990]. BPTT has become the standard training algorithm for all variations of RNNs.

Despite the ability to model sequence dynamics, a simple RNN architecture fall short in learning to capture long-term dependencies. As the time span grows longer, RNNs become unable to learn to connect past information. This problem was investigated in depth by [Bengio et al., 1994] and [Hochreiter et al., 2001], and was found to be a fundamental limitation of the RNN architecture. The problem is termed vanishing and exploding gradients that will surface when back propagating errors across many time steps. The term refers to the exponential shrinking (or exploding) of gradients magnitudes as they are propagated through time. This phenomenon causes memory of the network to ignore long term dependencies and hardly learn the correlation between temporally distant events. It is easy to understand the problem from the following two aspects:

- Standard nonlinear functions such as sigmoid function have a gradient which is close to zero almost everywhere.
- The magnitude of gradient is multiplied over and over by the recurrent matrix as it is back-propagated through time. Depending on the eigenvalues of the recurrent matrix, the gradient value quickly converge to zero or explode to infinity. In some cases, it can take as short as $5 \sim 10$ steps of back-propagation for this to happen [Mikolov et al., 2014].

Among many successful attempts to overcome the limitation, LSTM architecture [Hochreiter and Schmidhuber, 1997] is one of the most famous and widely adopted, which utilise carefully designed gates and states to mitigate vanishing gradient problem. LSTM introduces memory cell whose input and outputs are controlled by different gates. These gates control flow of information to hidden neurons and preserve extracted features (hence memory) from past timesteps. Figure 2.2 depicts a typical LSTM cell which includes an input, forget, and output gate and a cell activation component. Each memory cell contains a node with a self-connected recurrent edge

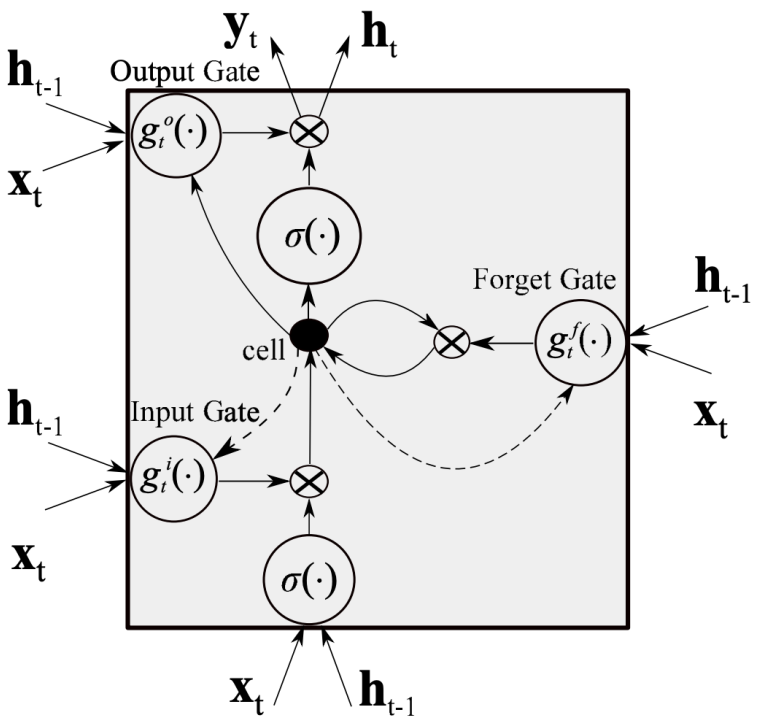


Figure 2.2: Architecture of a typical LSTM cell. The cell memory is controlled by both the input and forget gates, who will dictate how the cell integrate the current information relative to past information. Similarly, the output is controlled by an output gate.

with fixed weight of 1, encouraging that the gradient can pass though many time steps without vanishing or exploding. Because of the gated structure, an LSTM cell also have the ability to remove or add information to the cell state, therefore it can capture both long-term dependencies and shor-term ones.

Put formally, the weight update scheme of a LSTM unit can be expressed as below, where we use i, f, and o to refer to input, forget and output gates respectively.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2.4}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2.5}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}f_{t-1} + b_o) \tag{2.6}$$
$$g_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.7}$$
$$c_t = f_t c_{t-1} + i_t g_t \tag{2.8}$$
$$h_t = o_t tanh(c_t) \tag{2.9}$$

where σ and $tanh$ are activation function of sigmoid and hyperbolic tangent respectively. i_t, f_t control the degree to which the memory cell accumulate g_t, and o_t decides the hidden representation with respect to the memory cell. The value of the hidden

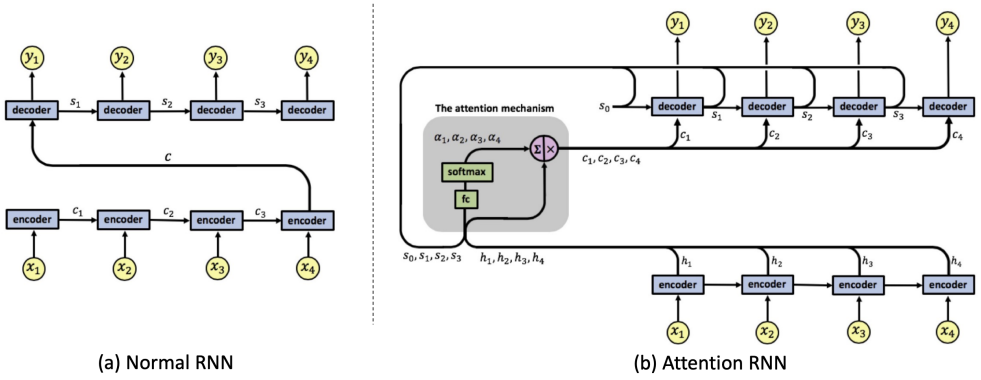


Figure 2.3: Difference between a normal RNN and a attention RNN.

layer of the LSTM at time t is the vector h_t, while h_{t-1} is the vector output by each memory cell in the hidden layer at the previous time.

2.1.1.2 Attention Network

Attention is a very influential idea in modern deep learning development, which is both powerful and versatile. It has been successfully applied on many vision tasks such as image captioning [You et al., 2016; Lu et al., 2017; Anderson et al., 2018], visual question answering [Lu et al., 2016; Xu and Saenko, 2016], time series prediction [Sharma et al., 2015; Liu et al., 2017a] etc.

Attention mechanism was originally proposed for Neural Machine Translation with seq2seq model [Bahdanau et al., 2014]. In seq2seq model, the input is encoded into a fixed length context vector, which is then decoded into desired outputs. The context vector should ideally preserve all relevant information in order to compute the output. However for long sequences, fixed-length context vector design is incapable to remember all the necessary information in the history, as shown by [Cho et al., 2014a]. Earlier parts of the sequence are often forgotten resulting in potential poor performance especially when there exist long-term dependencies. The attention mechanism was proposed to resolve this problem by allowing the model to automatically *attend* to relevant parts in history however long in the past they may be, without having to form these parts as a hard segment explicitly.

Attention mechanism achieves this by encoding the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively based on the current input, instead of just keeping the latest context vector. This frees the encoder from having to compress all relevant information in a potentially long sequence into a fixed length vector. Figure 2.3 show the difference between encoder-decoder models with a normal RNN and an attention one. The figure clearly illustrates how the attention mechanism aggregates and selects past information using attention weights.

Concretely, we define an input sequence of length T_x as $\mathbf{x} = (x_1, \ldots, x_{T_x})$, and an

output length of T_y as $\mathbf{y} = (y_1, \ldots, y_{T_y})$. The output probability is:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \tag{2.10}$$

where s_i is an decoder hidden state at time i, computed as:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{2.11}$$

f and g are some non-linear functions. For example, g can be stacked LSTM network whole f represents a single LSTM cell. Note that compare to a normal encoder-decoder approach, both the output probability and the hidden state are conditioned on an additional context vector c_i, which is a intelligent aggregation of past information, computed by

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.12}$$

where h_j is the latent representation obtained through he encoder. The weight α_{ij} for each h_j is calculated using softmax normalisation

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.13}$$

where $e_{ij} = a(s_{i-1}, h_j)$ is called an attention module with a represented by a feedforward network. e_{ij} indicates how strongly connected the input at time j is to the past outputs.

Intuitively, with attention as such, the decoder decides which parts of the source sentence to pay attention to through the attention weights α. By letting the decoder selectively look back at past inputs, the encoder is relieved from the burden of having to encode all information in the input sequence into a fixed length vector, thus the model as whole can effectively capture long-term dependencies.

2.1.2 3D & Temporal Convolution

TCN uses a 1D fully-convolutional network (FCN) architecture, where each hidden layer is the concatenation of 1D feature maps obtained from 1D kernel. Conventionally, each hidden layer has the same length as the input layer, and zero padding (with $kernelsize - 1$) is added to keep the length of subsequent layers consistent. TCN also uses causal convolutions, where an output at time t is convolved only with elements from features prior to time t in the previous layer.

When dealing with video data, TCN naturally extends to 3D convolution networks [Tran et al., 2015a]. 3D convolutions applies a 3 dimensional kernel to the input data and the filter moves in all 3 directions, x, y and t. This way, 3D CNNs extract the visual characteristics from a set of images and they take into account the order of the images in the video. In essence, it computes spatial-temporal feature representations of the input sequence. The feature space of 3D CNN can be think of

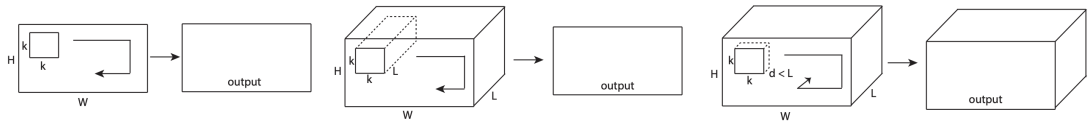


Figure 2.4: Difference between 2D CNN, stack-frame CNN and 3D CNN.

a 3 dimensional volume space such as cube or cuboid.

It should be noted that 3D CNN is fundamentally different than stack-frame CNN in modelling temporal dynamics [Simonyan and Zisserman, 2014a]. Figure 2.4 illustrate the difference between 3D CNN to 2D and stack-frame CNN. The distinguishing feature of 3D CNN is that the convolutional kernels swipe along the temporal dimension in a strided fashion. This offers flexibility of how long in time can the network model the sequence and also stronger modelling capabilities. For example, with a long input sequence, a stack-frame CNN would have to use huge kernels of size $x \times y \times t$ with t being as large as the sequence length to model global dynamics, resulting in a network that is inefficient and hard to tune. On the contrary, 3D CNN can use any t while extracting better spatial-temporal features because the kernels moves in the temporal dimension.

This basic architecture of 3D CNN using normal convolutions can only look back at sequence history with length that is linear to the depth of the network. As a result, capturing long term dependencies becomes difficult. Yu *et al.* proposed dilated convolution with wider kernels to mitigate this problem. We can formally define normal discrete convolution as:

Let $F : \mathbb{Z}^2 \to \mathbb{R}$ be a discrete function. Let $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$ and let $k : \Omega_r \to \mathbb{R}$ be a discrete filter of size $(2r + 1)^2$. The discrete convolution operator $*$ can be defined as:

$$(F * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}) \tag{2.14}$$

Similarly, a dilated convolution operator $*_l$ with dilation factor l can be defined as:

$$(F *_l k)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}) \tag{2.15}$$

With dilated convolution, we can achieve exponentially expanding receptive fields by applying the filters with exponentially increasing dilation:

$$F_{i+1} = F_i *_{2^i} k_i \; for \; i = 0, 1, \ldots n - 2 \tag{2.16}$$

where $F_0, F_1, \ldots F_{n-1} : \mathbb{Z}^2 \to \mathbb{R}$ are discrete functions and $k_0, k_1, \ldots k_{n-2} : \Omega_1 \to \mathbb{R}$ are discrete convolution filters. It is easy to see that the size of the receptive field of each element in F_{i+1} is $(2^{i+2} - 1)$x$(2^{i+2} - 1)$. When applying dilated convolution on the temporal dimension, we can look back at sequence history with length exponential to the depth of the network, which is illustrated in Figure 2.5

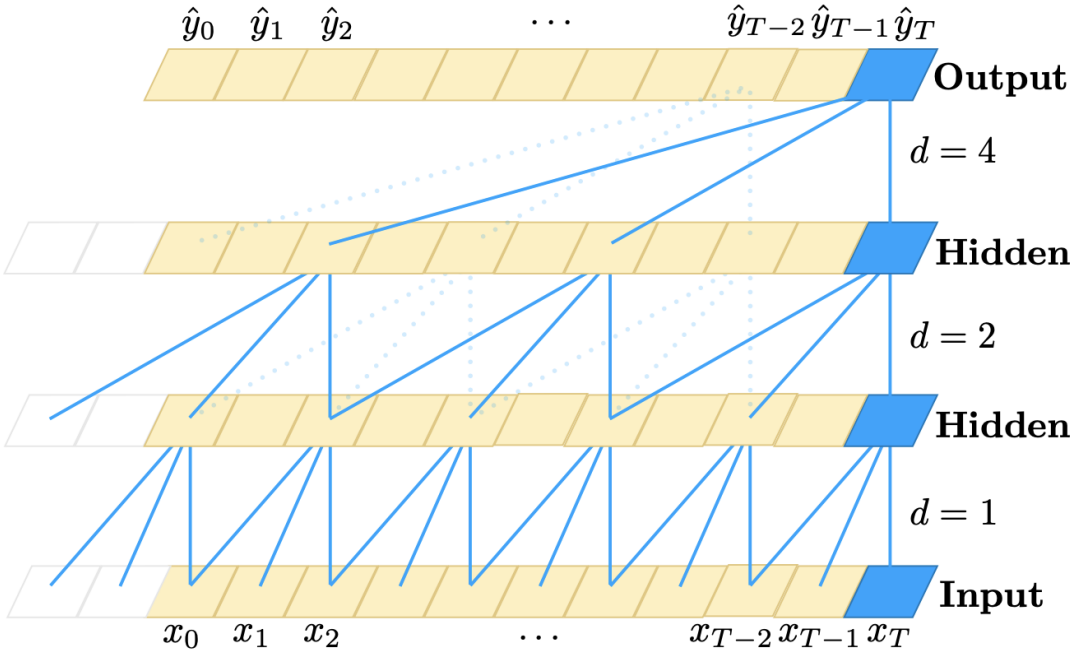


Figure 2.5: Temporal dilated convolution with exponential receptive fields where d is the dilation factor. We can look further back in history of the sequence to capture long-term dependencies.

2.2 Few-shot Learning

Modern deep learning succeeds in many data-intensive applications, but it cannot rapidly generalise from a few examples to perform the required tasks. On the contrary, human can easily learn from a few examples mainly because human can effectively utilise prior knowledge. An obvious example is that given only a few photos of a stranger, a human can identify the same person from a large number of photos with ease. Even harder, human can still identify a person given only artistically created cartoon-like images. This indicates that human are especially good at learning transferable features that can generalise to unseen data or even to other tasks.

Bridging this gap between machine learning systems and human-like intelligence to be able to learn from a few examples by learning transferable representations is one of the core challenge of machine learning research. Therefore, Few Shot Learning (FSL) is proposed [Fei-Fei et al., 2006; Fink, 2005]. When there is only one example available to learn from, FSL is also called one-shot learning. With few-shot learning, we can avoid collecting large scale dataset, which is labour intensive and cost inefficient, and also leverage past learned knowledge for better transfer learning.

Two family of methods, among others, have been proven effective for few-shot learning and are popularly adopted.

Embedding Learning Based Methods [Vinyals et al., 2016; Koch et al., 2015; Snell et al., 2017] It tries to learn a embedding space $z_i \in \mathcal{Z}$ for data samples x_i in a training dataset $\mathcal{X} = \{(x_i, y_i)\}$ such that similar and dissimilar pairs can be easily identified. The assumption behind embedding learning based methods is the similarity constraints which states that samples of the same class are more similar than sample from different classes based on some similarity measure in some embedding space. Therefore, by learning such discriminative embedding space, we can classify a target data sample based on its most similar samples in a Nearest Neighbour [Altman, 1992] fashion.

Specifically, let us define two functions $g(\dot{})$ and $f(\dot{})$ that embed a training sample x_i^{train} and testing sample x_i^{test} respectively. We also define a similarity measure $s(\dot{,}\dot{})$ which is used to calculate the similarities (or distances) between embeddings of a target sample $f(x_i^{test})$ and every training samples $\{gx_i^{train}, |i = 1, 2, \ldots, N^{train}\}$ where N^{train} is the total number of training samples. Note that function f and g are usually the same but in some cases using different embedding function for training and testing samples can offer better classification accuracy [Vinyals et al., 2016; Bertinetto et al., 2016]. After comparing the data samples, we obtain a set of similarity measures, and the target sample can be simply classified as the class as its nearest neighbour. Another popular family of methods MAML [Finn et al., 2017, 2018] also falls into this category, although they do not explicitly learn a similarity measure. MAML achieves embedding learning by combining a meta-learner with the learner, and directly computing and applying the gradient with respect to the meta-learning objective. A illustration of using the embedding learning method for few-shot learning is shown in Figure 2.6 Because the goal of embedding learning is to find a latent space that satisfy the similarity constraints, the learning process does not explicitly involve class information. In other words, the learned embedding function does not consider any class-specific knowledge and the learned feature are class agnostic. Hence when the system is presented with samples from previously unseen classes, the similarity constraint will most likely still hold in the embedding space, enabling the system to make prediction with as few as one labeled example.

External Memory Based Methods External memory for one-shot learning [Santoro et al., 2016a; Mishra et al., 2017a] is proposed to let the learning systems mimic human-like intelligence because we human rely heavily our memory and past experience to comprehend new things. Prestigious works such as Neural Turing Machine (NTM) [Graves et al., 2014] and memory networks [Weston et al., 2014; Sukhbaatar et al., 2015] allow short-term memorisation, and have offered state-of-the-art performance in various tasks including rule-based data manipulation, language modelling and question answering. To a normal learning system, when presented with a new task and new training data, the model has to be re-trained to incorporate useful information. Learning with external memory, however, the system can directly utilise the memorised knowledge from previous tasks through retrieving, adding, and modifying the memory entries for fast generalisation. Much like a human, the memory relieve the burden of learning from scratch. Figure 2.7 illustrate the idea behind

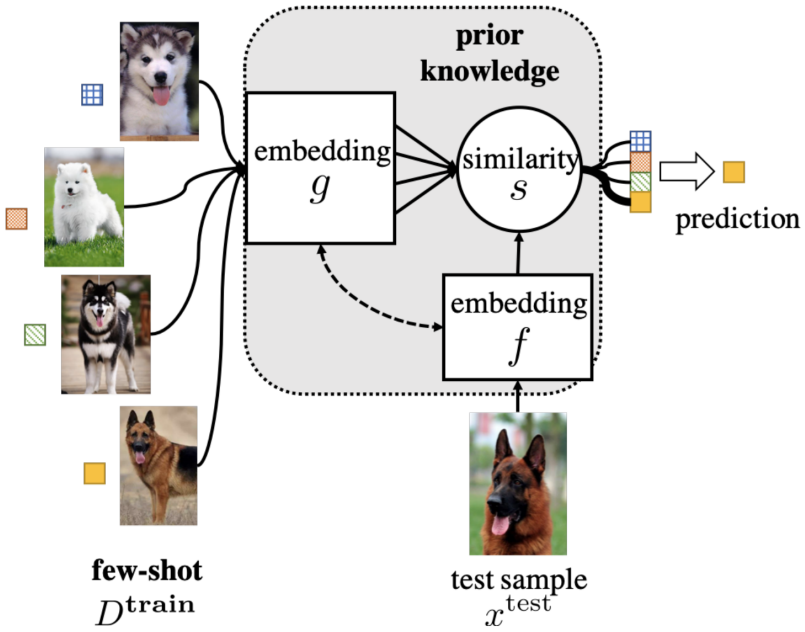


Figure 2.6: Illustration of employing embedding learning strategy for classification to address few-shot learning problem. Note that because the class prediction is based on similarity only, the embedding itself can be class agnostic, thus the learned feature can be transferred to other tasks.

external memory.

Formally, we denote the collection of external memory as a matrix $\mathbf{M} \in \mathbb{R}^{b \times m}$ where b is the memory capacity (number of memory slots) and m is the memory dimension. Given a sample x_i, a function f is first used to embed the sample to a query vector $\mathbf{q} = f(x) \in \mathbb{R}^m$. It is then attended to the memory through some similarity measure, and the system will decide how to make prediction and update memory accordingly. For memory retrieval, the memory is address by the query by a cosine similarity:

$$S(\mathbf{q}_t, \mathbf{M}_t(j)) = \frac{\mathbf{q}_t \cdot \mathbf{M}_t(j)}{\|\mathbf{q}_t\| \|\mathbf{M}_t(j)\|} \tag{2.17}$$

which is then used to create a read-weight vector w^r with elements calculated by softmax:

$$\mathbf{w}^r(i) = \frac{exp(S(\mathbf{q}_t, \mathbf{M}_t(i)))}{\Sigma_j exp(S(\mathbf{q}_t, \mathbf{M}_t(j)))} \tag{2.18}$$

Then, an associated memory $\mathbf{r}$ with $\mathbf{q}$ is retrieved by

$$\mathbf{r} = \sum_i \mathbf{w}^r(i) \mathbf{M}_t(i) \tag{2.19}$$

Finally, this retrieved memory is used by the system to make predictions.

For memory update, whether to modify existing slots or create new ones, a write

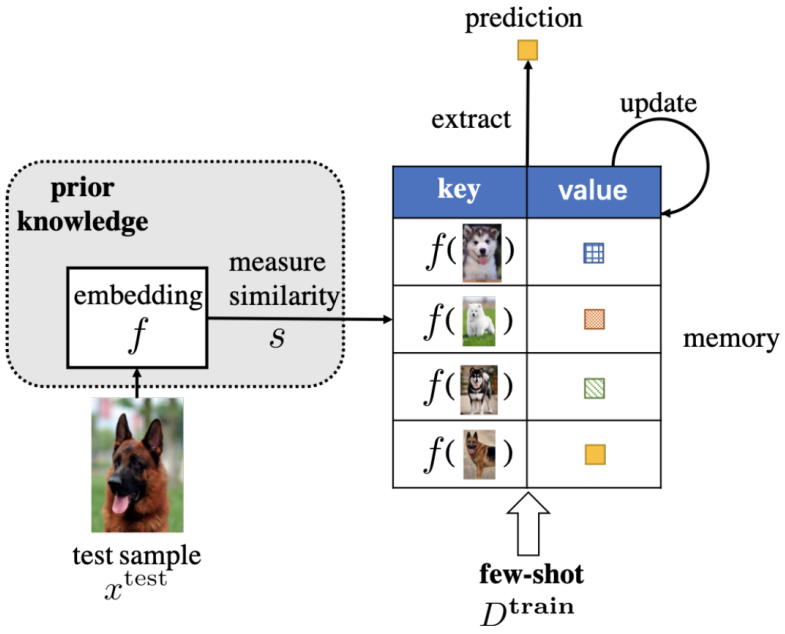


Figure 2.7: Illustration of networks with external memory. The memory is updated by measuring the similarity using training samples. The embedding for a test sample is obtained by 'attending' to these memories.

weight $\mathbf{w}^w$, a least-used weight $\mathbf{w}^{lu}$ and a update weight $\mathbf{w}^u$ are first computed:

$$\mathbf{w}_t^w = \sigma(\alpha)\mathbf{x}_{t-1}^r + (1 - \sigma(\alpha))\mathbf{w}_{t-1}^{lu} \tag{2.20}$$

$$\mathbf{w}_t^u = \gamma \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w \tag{2.21}$$

$$w_t^{lu}(i) = \begin{cases} 0 \text{ if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 \text{ if } w_t^u(i) \leq m(\mathbf{w}_t^u, n) \end{cases} \tag{2.22}$$

where $\sigma(\alpha)$ is a sigmoid function controlled by α: $\frac{1}{1+e^{-\alpha}}$; and we denote function $m(\mathbf{v}, n)$ as the nth smallest element of vector $\mathbf{v}$. According to the equations, the least used memory is found and set to zero, then memory updates occurs in accordance with the computed vector of write weights:

$$\mathbf{M}_t(i) = \mathbf{M}_{t-1}(i) + w_t^w(i)\mathbf{q}_t, \ \forall i \tag{2.23}$$

This new knowledge can be written into either a empty slot or the least used one.

2.3 Generative Models

Generative modelling can be defined as [Bishop, 2006]:

> Approaches that explicitly or implicitly model the distribution of inputs as well as outputs are known as generative models, because by sampling from them it is possible to generate synthetic data points in the input space.

A generative model describes how a dataset is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data. Generative modelling are the counter-part of widely studied *discriminative modelling*. Take image generation for example: suppose we have a dataset of flowers, we wish to build a model that can generate new images of flowers that has never existed in the dataset but still look genuine as if they were from the dataset. A generative model can achieve this by learning the rules that governs the appearance of a flower.

This is in fact a very difficult task considering the vast number of different values that each individual pixels can take in an image and the relatively tiny number of such arrangements that constitute an legit image of the entity we are trying to simulate. To put the problem to scale, for a grey-scale image of resolution 128x128 with 8-bit depth, there are in total $256^{128\times128}$ number of all possible arrangements, and only a tiny fraction among those will be interpreted as flowers by human.

Because of the complexity of the problem, the generative model has to be probabilistic in nature rather than deterministic. We should assume that a given dataset contain some unknown but intrinsic probabilistic distribution that dictates the characteristics of its images. Therefore, the generative model needs to mimic and converge towards this distribution for it to generate realistic looking images. Compared to discriminative modelling, generative model does not necessarily care about the category (or the label) of the data. Instead, it attempts to predict the probability of seeing an observation.

Since generative modelling is such a broad topic, we confine our discussing to two of the most popular deep generative models, namely Generative Adversarial Networks [Goodfellow et al., 2014] and Variational Autoencoders [Kingma and Welling, 2013].

2.3.1 Generative Adversarial Networks

Generative adversarial networks are one of the most interesting ideas in modern machine learning. It was originally proposed by [Goodfellow et al., 2014]. Two models are trained simultaneously in an adversarial process, where a generator tries to create images that look real, while a discriminator learns to distinguish fake images produced by the generator from real images. Both the generator and the discriminator are approximated by deep neural networks. The two networks are competing with each other with opposite objectives during training. As a result, the generator becomes progressively better at creating fake images that look real, while the discriminator becomes better at telling them apart. Note that the networks are getting better

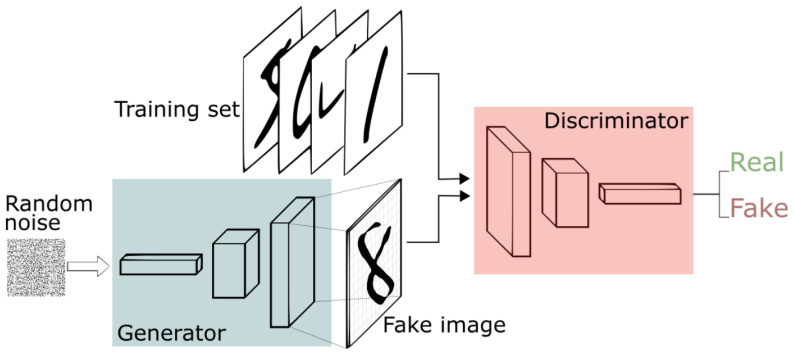


Figure 2.8: Overview of Generative Adversarial Networks. It generate realistic images from noise vectors that are usually sampled from a Gaussian distribution.

because of their opponents are getting better as well, which encourage the networks to work harder to achieve their objectives. This iterative process reaches equilibrium when the discriminator can no longer distinguish real images from fakes. Figure 2.8 gives an overview of the GAN in its simplest form.

To learn the generator's distribution p_g that approximate the real data distribution $p_{data}(\mathbf{x})$, we first sample an input noise variable $\mathbf{z}$ from a prior distribution $p_z(\mathbf{z})$. We define the mapping from this noise vector to data as $G(\mathbf{z};\theta_g)$ where G is the generator parameterised by θ_g. Similarly, we also define a discriminator as $D(\mathbf{x};\theta_d)$ that takes an image as input, and its output $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ came from the data distribution rather than p_g. D is trained to maximise the probability of assigning label 1 to real data from the dataset and assigning 0 to data generated from G. On the contrary, G is simultaneously trained to fool the discriminator to maximise the likelihood of fake data. We can represent this adversarial process as a two-player minimax game with value function $V(G,D)$:

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{z\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.24}$$

The generator and discriminator are learned jointly by the alternating updates of parameters through gradient descent. The generator is fixed while performing one (or a few) iteration(s) of SGD on the discriminator using the real and the generated images. Then the discriminator is fixed when updating the generator. The following diagram summarises the data flow and the training process of GAN.

Since the original GAN framework was introduced, many improvements were proposed for faster convergence or more stable training etc. One simple improvements is to change the loss function for the generator from $\mathbb{E}_{z\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$ to $-\mathbb{E}_{z\sim p_z(\mathbf{z})}[\log D(G(\mathbf{z}))]$. This modification was used in the original GAN paper [Goodfellow et al., 2014] to mitigate the issue with diminishing gradient. During the training process, it is usually easier to distinguish the generated images from real ones in early stage, which makes $V(G) = \mathbb{E}_{z\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$ to approach 0 quite quickly, resulting in very slow or even stalled optimisation.

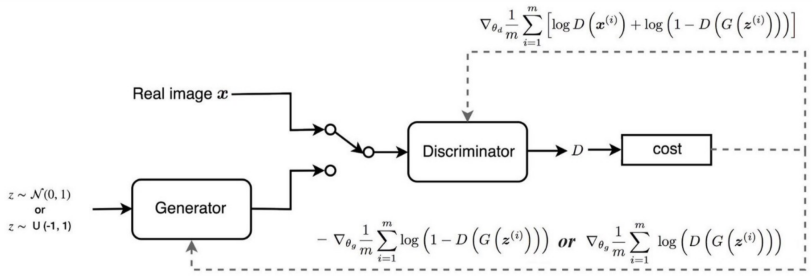


Figure 2.9: Training process of GAN. Generator and discriminator are updated alternatively using their respective objectives.

A more fundamental improvement for GAN is Wasserstein GAN [(WGAN) [Arjovsky et al., 2017]], which is based on the observation that the supports of p_{data} and p_g in many real world datasets' distribution often lies on low dimensional manifolds, which are almost certainly going to be disjoint in the low dimensional space. Therefore it is always capable to find a discriminator that can separates real and fake samples perfectly, which will lead to instability and mode collapse of GAN training [Arjovsky et al., 2017]. WGAN addresses this problem by using 1-Wasserstein Distance (Earth Moving Distance) instead of KL Divergence to measure how close two distributions are. This translate to the following modified objectives:

$$\min_G \max_D V_{WGAN}(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[D(\mathbf{x})] - \mathbb{E}_{z\sim p_z(\mathbf{z})}[D(G(\mathbf{z}))] \tag{2.25}$$

Other than the change in objective function, the author also proposed other changes compared to original GAN:

- Weight clipping. After the gradient update on the discriminator, clamp the weights to a small fixed range, $[-c, c]$.

- It is recommended empirically to use *RMSProp* optimiser on the discriminator, rather than a momentum based optimiser such as *Adam*.

Other works proposed WGAN with gradient penalty (WGAN-GP) [Gulrajani et al., 2017] to get around the engineering hack-like weight clipping. Instead of applying clipping, WGAN-GP penalises the model if the gradient norm moves away from its target norm, which is set to 1.

Similar to WAGN, LSGAN [Mao et al., 2017] also propose to use different objective functions. The key difference for LSGAN is the loss function is set up to use L2 loss instead the log loss in original GAN or the L1 loss in WAGN, and it does not need the problematic weight clipping. Formally, the objectives of LSGAN can be

expressed as:

$$\min_D V_{LSGAN}(D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[(D(\mathbf{x})-1)^2] + \mathbb{E}_{z\sim p_z(\mathbf{z})}[(D(G(\mathbf{z})))^2] \tag{2.26}$$

$$\min_G V_{LSGAN}(G) = \mathbb{E}_{z\sim p_z(\mathbf{z})}[(D(G(\mathbf{z}))-1)^2] \tag{2.27}$$

For its simplicity, we employ LSGAN training strategy in all of our experiments in Chapter 6.

2.3.2 Variational Autoencoders

Variational Autoencoders (VAEs) [Kingma and Welling, 2013] are powerful generative models just as GANs, with diverse applications ranging from generating fake human faces [Hou et al., 2017], to producing purely synthetic music [Roberts et al., 2017]. It neatly integrate unsupervised deep learning and variational Bayesian methods together.

From a application perspective, VAEs naturally offer the advantage of being able to control the generated image in a desired, specific direction. For example, we can tell VAE to generate image of digits with thicker lines the tilt the digit a little. This is because that VAE is a Latent Variable Model where each value in the learned latent representation corresponds to a specific feature of the generated results.

In standard autoencoders, the network learns to generate compact latent representations in order to reconstruct the input well, but the space it learns may not be continues, making it harder to interpolate the space in order to generate novel images. A VAE on the other hand, provides a probabilistic description for the observation in latent space, *i.e.* it learns a probability distribution for each latent variable rather than learning a single value, which is essentially enforcing a continuous, smooth latent space representation. Take face generation for example, an normal autoencoder would learn that the smile factor for the face in Figure 2.10 is 0.99, however, it is often desirable to represent a attribute as a range of possible values (a probability distribution). This way, when decoding the latent representation, we can randomly sample from the distribution to generate novel images, like the face with a sad expression but still look like the same person. Formally, let's suppose there exists some hidden variable $\mathbf{z}$ which generates an observation $\mathbf{x}$. A VAE would like to like to infer $\mathbf{z}$ based on the observation of $\mathbf{x}$. According to Bayes rule, $\mathbf{z}$ can be computed as:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \tag{2.28}$$

where

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{2.29}$$

However, the integration turns out to be intractable because it needs to be evaluated over all possible configurations of latent variables. We therefore need to approximate this posterior distribution by applying variational inference. This is how VAE got its name.

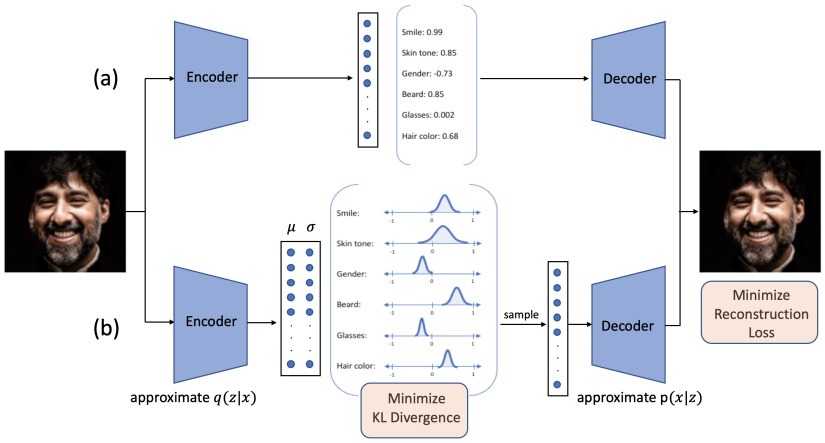


Figure 2.10: Overview of a Variational Autoencoder. Compared to a normal autoencoder, VAE learns a distribution (means and variances) instead of the feature vector itself.

Let's approximate $p(\mathbf{z}|\mathbf{x})$ by another distribution $q(\mathbf{z}|\mathbf{x})$ which we define to have a tractable distribution. All we need to do is to tune the parameter of q such that it is very similar to p. KL divergence is a natural choice for measuring the difference between two probability distributions. To ensure that $q(\mathbf{z}|\mathbf{x})$ is similar to $p(\mathbf{z}|\mathbf{x})$, we can minimise the KL divergence between the two, which is calculated as:

$$\mathbb{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}[\log(q(\mathbf{z}|\mathbf{x}))] - \mathbb{E}[\log(p(\mathbf{x},\mathbf{z}))] + \log p(\mathbf{x}) \tag{2.30}$$

By Jensen inequality, we can derive that minimising the above KL divergence is equivalent to maximising the following:

$$\max \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}) - \mathbb{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{2.31}$$

The first term can be interpreted as the reconstruction likelihood while the second term ensures the approximated distribution q is close to the true prior distribution p. Please observe that this maximisation aligns well with graphical representation in Figure 2.10. We can approximate the q by an encoder to generate the latent representation $\mathbf{z}$ of the input $\mathbf{x}$, then use a decoder who approximate p to map $\mathbf{z}$ back to a reconstructed image $\hat{\mathbf{x}}$. The loss function of VAE can be written as:

$$\mathcal{L}(\mathbf{x},\hat{\mathbf{x}}) + \sum_j \mathbb{KL}(q_j(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{2.32}$$

where j indicates each variable in the latent space.

As we are computing a probability distribution $q(\mathbf{z})$ for the latent variables rather than variables values directly, the encoder model of a VAE will output parameters describing a distribution for each dimension in the latent space. Normally, we assume that the prior follow a Gaussian distribution, so the outputs from the encoder is two

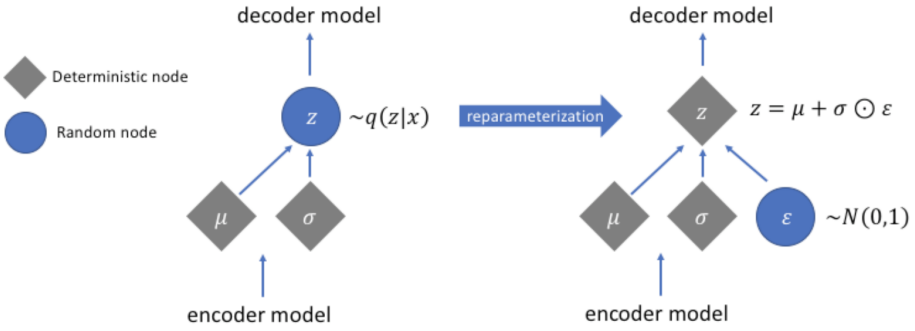


Figure 2.11: Reparameterization trick when training VAE. It avoids the random process of sampling from a distribution.

vectors describing the mean and the variance. The decoder model will then generate a latent vector by sampling from these learned distributions and proceed to develop a reconstruction model from the realizations. However, the training process using gradient decent with back propagation does not allow random sampling. Therefore, we leverage the *reparameterization trick* that introduces an additional random variable input ϵ sampled from a unit Gaussian distribution. Then ϵ is shifted by the latent distribution mean μ and scaled by the latent distribution variance σ. This process is illustrated in Figure 2.11. We can see now gradient descent can be applied as it would in any other networks.

So far, VAEs are trained to perform reconstruction. To use it as a generative model, we simply sample from the learned latent representation and decode. By sampling one variable at a time with others fixed, we can see which latent variables corresponds to which attributes in the image, thus we control the generated image to have certain desired features.

2.4 Summary

In this chapter, we introduced some theoretical foundations and background knowledge that are related to works in later chapters. They include:

- Sequence analysis. Temporal convolution and 3D CNN; RNN and its variations.
- One(few) shot learning. Embedding learning based learning methods and memory augmented networks.
- Generative modelling. GANs and VAEs as well as their variations and extensions.

Chapter 3

GADD - RGBD Dataset for Fine-grained Action localisation

We build a new RGBD video dataset called Gym Activities Depth Dataset (GADD) for complex activity analysis. While action recognition is a challenging and interesting task on its own, action detection takes the problem one step further and finds itself of great importance in many real world applications. Our GADD dataset is specifically designed to study the problem of sequence labeling and action detection, and can also be adapted to suit the investigation on action recognition. Unlike most existing RGBD datasets that only contain one action per video, the videos in our dataset consist of a sequence of consecutive actions. Such scenarios are commonly seen in the real world problems and localising actions becomes more challenging in this dataset. Figure 3.2 shows some sample frames from the GADD dataset. The dataset contain over 800 RGBD videos, each 1-2 minutes long at 30 frames per second, with at least 12 action instances inside each video. There are mainly 2 features of our dataset that makes it challenging for sequence labeling and action detection compared to existing ones:

- A wide range of action instance length, ranging from 1.5s for the shortest ones to over 15 seconds for longer ones. This poses a strict requirement on scalability for the detector or the labeling network.
- Some action instances contain other actions as a components. This may confuse the detector or labeling network if it is not capable of modelling longer-term dynamics.

These two features of the dataset will be explain in detail later in the Chapter.

3.1 Existing Datasets

Most commonly used large-scale action datasets are designed for understanding RGB videos [Soomro et al., 2012b; Kuehne et al., 2011; Karpathy et al., 2014]. The datasets that are available in depth data are often small in size and typically built for recognition or detection task only. The early RGB-D datasets from Microsoft [Yu et al., 2014;

Wang et al., 2012a,b; Li et al., 2010a; Yuan et al., 2009; Tian et al., 2012] have relatively low resolution and consist of simple actions. Recent RGB-D datasets include more complicated and challenging movements, such as CAD-60 [Sung et al., 2011], CAD-120[Koppula et al., 2013], Office Activity [Wang et al., 2014a] and RGBD-HuDaAct [Ni et al., 2013b]. Very recently, a new dataset aimed for data-driven algorithms called NTU RGB+D[Shahroudy et al., 2016] is introduced. It is by far the largest dataset with 56880 depth sequences containing over 4 million frames. However, they are mostly for the recognition task and include only one or a few simple actions per video. KSCGR dataset[Shimada et al., 2013] is a cooking RGBD dataset containing a sequence of actions. However, the dataset is relatively small and focuses on specific actions involving only two arms. Therefore, we build our own RGB-D dataset for action detection and labeling in this work, which consists of realistic and complex gym exercise and workout activities.

3.2 Dataset Description

We provide a thorough description of the dataset in this section.

3.2.1 Actions

Our dataset contains 22 different action classes (23 with background class) of gym workout exercises (push up, KB swing, lunge twist, etc.). A full list of all the exercises and their sample frames are shown in Figure 3.1. In addition, the mapping from class ID to class name is given in Table 3.1 Four of these classes require the subject to use a tool, therefore human-object interaction is included in the action sequences. The reason for choosing gym workout exercises is that all of the actions are well-defined and constrained. Compared to other dataset that contain actions such walking, running, playing basketball, our actions have clear and easily identified starting and ending points instead of a vague transition between actions. This is crucial for the proper evaluation of detection tasks where temporal localisation is the main challenge. Also, action such as *Jumping Burpee* contains *Push Up* as its components.

3.2.2 Subjects

There are in total 25 subjects participated in the video recording process. Different subjects are randomly assigned with different actions to perform. Once actions are determined, they perform all of them in a row (*e.g.* 3 push-ups followed by 4 roman-twist), thus creating an action sequence. Each subject are assigned at least 4 different action classes and they are asked to repeat each one at least 3 times, making the total count of action instances greater than 12 in every videos captured.

ID	Class Name
1	Push Up
2	Sit Up
3	V-Sit-Up
4	Roman Twist
5	Burpee
6	KB Swing
7	Dead Lift
8	BW Squat
9	Thruster
10	Mountain Climb
11	KB Lunges
12	Leg Raise
13	Single Leg Bridge on Hip
14	Bird Dog
15	Rotated Plank
16	Hip Extension
17	Lunge Twist
18	High Knees
19	Heel to Glute
20	Lower Side Step
21	World's Best Stretch
22	Side Squats

Table 3.1: ID-to-Name Mapping for GADD dataset.

3.2.3 View Points

We set up cameras from 4 different angles when recording every videos. They are separated in a 180°spread, at roughly about 0°, 60°, 120°and 180°respectively. Because the subject will be constantly moving, the angles relative to the subject will change during filming. The four different views are separated far enough, which creates large pose variation in the dataset. In the detection and sequence labeling task, we assume the location of the cameras are unknown and no calibrations are preformed on the depth maps from different angles. The purpose of creating 4 different view points is to increase data variability and encourage higher robustness of the action representations.

Figure 3.2 show the the RGB and depth frames from 4 different view points with 4 different performers.

3.2.4 RGBD Sensor and General Setup

Here we explain the whole setup for capturing actions, as well as some background movements that repeatedly occur.

Figure 3.3 shows the top view of the entire filming setup. The 4 Kinect sensors are separated evenly across a 180°span. Each camera simultaneously capture RGB and depth frames at 30 frames per second. All four cameras are not calibrated or synchronised, so the RGB and the depth video captured by each camera are independent. Because Kinect captures raw 24-bits RGB pixel and 16-bits depth pixel values without compression, the data cannot be written to disk in time thus the RAM, which serves as buffer, will be filled up. On the devices we use to capture the videos, the RAM can last about 2 minutes before being fed up. That is why each of our video only contains 12-15 action instances and lasts less than two minutes.

Every performer starts by standing roughly in the middle. When all 4 cameras are capturing, the person is then told to start doing gym exercises. The performer are given a list of actions to perform beforehand. Generally, it is expected that the person will repeat each action 3 time before moving on to the next, except for some really short actions which they are asked to repeat 5 times. However, we do not enforce this rule strictly, people are allowed improvise as they see fit. The goal is to contain at least action instances per video. Please note that some of the actions is actually an extension of other actions as explained in Section 3.2.1, which makes it especially challenging because the recognition algorithm may make false predictions if not observing enough of the action.

3.2.5 Data Formats

We collected the data using Microsoft Kinect v2, and recorded the RGB and depth maps for each frame. After filming all the actions, we manually label each actions by its start and end frame numbers. Each video sequence has three components: Depth frames, RGB frames and its action annotations.

- **Depth Frames**. We strive to preserve the depth details as much as possible so that 3D surface information are kept intact. Each depth frame is of resolution 512x424, with each pixel value representing the distance between a real world point and the Kinect camera plane. The depth value is the unit of mm, and is quantized using 16 bit unsigned integer. Therefore, each depth frame is a 512x424 1 channel uint16 png image. We applied some light pre-processing by applying median filter on the depth information provided by the Kinect sensor to filter out small noises (unwanted holes etc.) caused the imperfection of the sensors. No further pre-processing was done to keep the depth information as original as possible.

- **RGB Frames** The native resolution of the RGB frame coming from the Kinect sensor is 1920x1080. We figured that it is unnecessary to use HD RGB frames for action recognition and detection, while at the same time HD images takes up much disk space. Therefore each RGB frames is firstly compressed using JPEG, then we center crop each frame to 1080x1080 to retain its aspect ratio, and resized the cropped part to 270x270. At the end, each RGB frame is a 270x270 3 channel uint8 jpg image.

- **Video Annotations** For the annotation, we record the annotated class and frame indices of every action instances. Associated with every videos, a `label.txt` file is created containing the the following information: The first numbers of each row are the action label (a number between 1 to 22) that indicated the actions performed in the video. Within each row, from the second number to the end are the start and end frame indices of each occurrences of the action. Note that different actions in different video are repeated different times (normally 3 times), so not all lines in the `label.txt` file have the same length. We can see form the example annotation file below that that the video have action 1, 2, 8 and 12 in it, and action 1 is present from the 67th frame to the 103rd frame, and then again from 104th frame to 135th frame etc. This type of annotation essentially labels every frames to an action class.

```
1 67 103 104 135 136 169
2 402 464 465 528 529 558
8 761 797 798 840 841 878
12 1111 1189 1190 1270 1271 1336
```

3.2.6 Pre-processing of Frames

For each RGB frame, we capture at native Kinect resolution of 1920x1080. However, we do not need such high resolution frames for action recognition and localisation. Therefore, we center crop the original frame to 1080x1080 to keep the aspect ratio, and resize to 270x270 resolution. The resulting RGB frames are 270x270 `uint8` RGB images.

Similarly for depth frames, which are captured in native Kinect resolution of 512x424, we first center crop them to 424x424, and then resize to 227x227. After crop and resize, we apply median filter on all depth frames to fill up small holes and reduce the noise induced by the depth sensor. The resulting depth frames are 227x227 `uint16` single channel images.

3.3 Benchmark

Table 3.2 summarizes the performance of mainstream methods on GADD dataset as baselines for temporal action localization.

3.4 Conclusion

In this chapter, we present a RGB-D gym activity dataset called GADD to facilitate research on action recognition and action localisation. All the actions in the dataset have distinct start and end points which define a clear boundary between actions or action/background. The dataset is challenging for two main reasons:

	RGB		Depth	
Methods	**mAP@0.5**	**mAP@0.7**	**mAP@0.5**	**mAP@0.7**
multi-stage CNN [Shou et al.]	31.5	23.9	28.8	23.7
TCN [Dai et al., 2017]	40.7	36.4	37.1	35.5
Instance-aware [Yang et al., 2018]	44.3	40.6	45.8	40.9
PGCN [Zeng et al., 2019]	57.2	54.2	55.3	54.1

Table 3.2: Temporal action localization baseline performance on GADD dataset. This table covers major action localization methods from 2016 to 2019. Mean Average Precision score on IoU 0.5 and 0.7 are reported.

1. The duration of action differs significantly, ranging from 2 seconds to 30 seconds. This tests the ability of the localisation algorithms to handle multi-scale instances.

2. Some actions include other shorter actions as their components, meaning the pattern of movements are repeated in multiple different actions. This tests the ability of the localisation or recognition algorithms to effectively aggregate long-term temporal information.

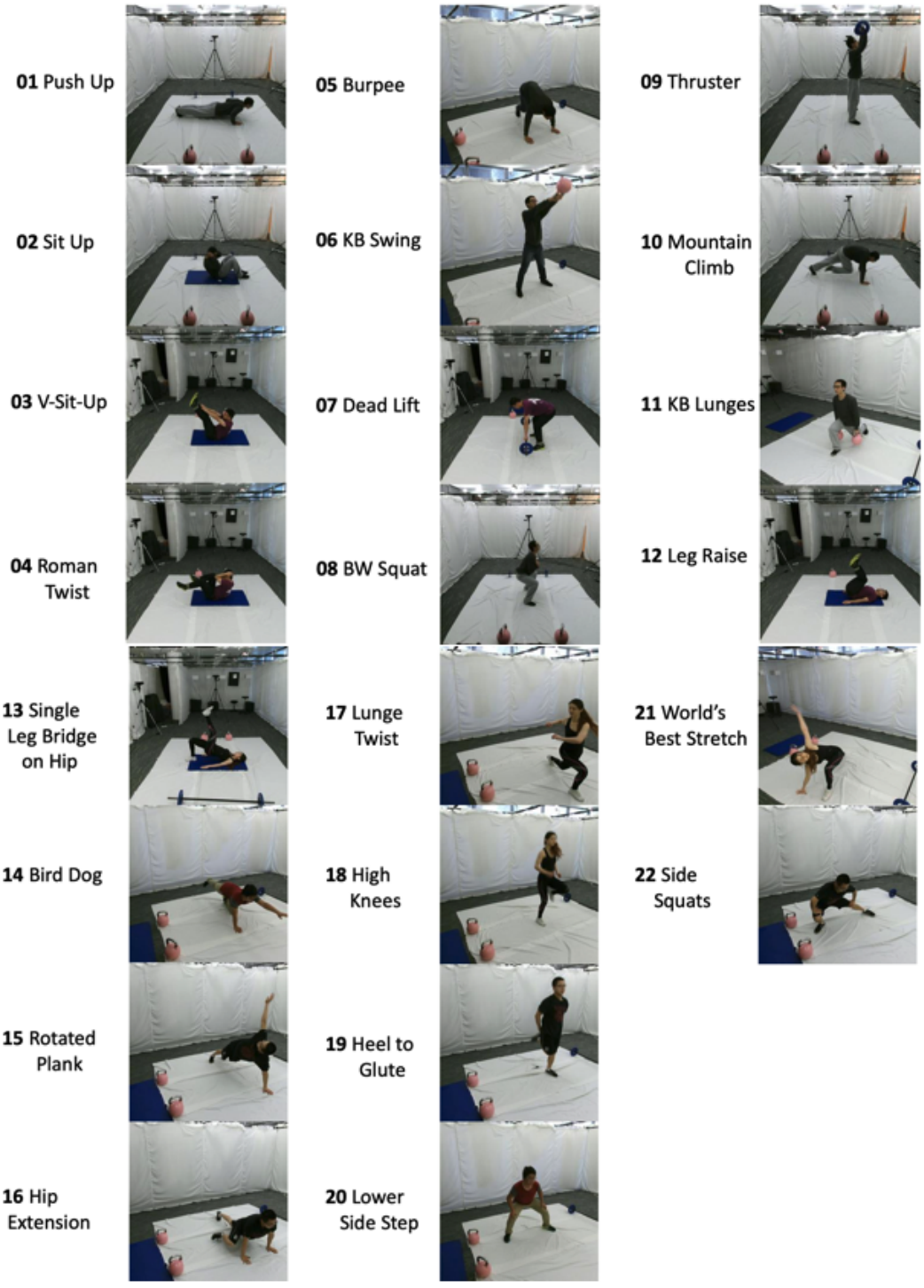


Figure 3.1: List of actions in the GADD dataset

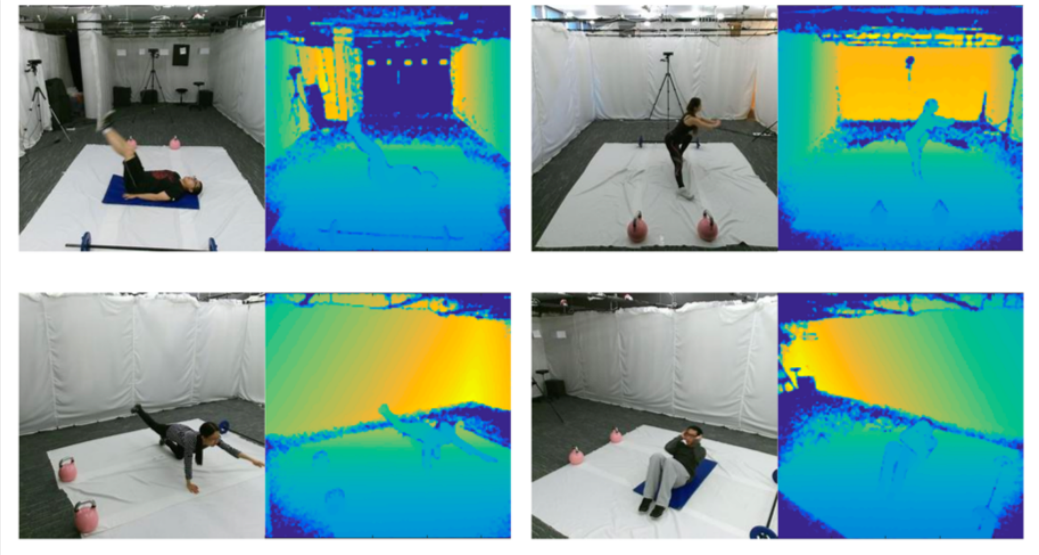


Figure 3.2: RGB and depth frame from 4 viewpoints with 4 different performers.

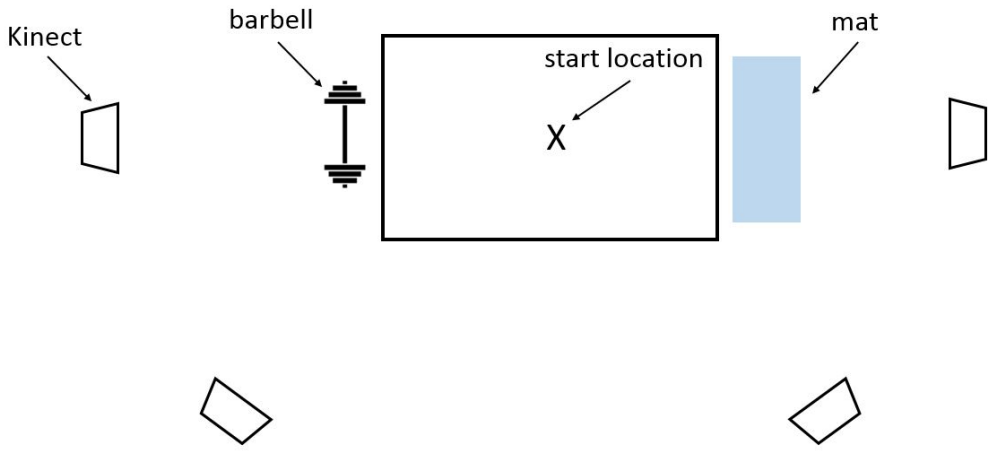


Figure 3.3: Filming setup.

Chapter 4

Synergy of Action Detection and Sequence Labeling

Understanding complex activities from videos is a fundamental problem in computer vision and has a wide range of applications in surveillance, human-computer interaction and personal robotics [Herath et al., 2016]. Much progress has been made in the key tasks involved in activity analysis, including action classification [Wang and Schmid, 2013; Karpathy et al., 2014; Bilen et al., 2016a; Simonyan and Zisserman, 2014a; Tran et al., 2015a; Donahue et al., 2015; Wang et al., 2017d], detection [Ni et al., 2016; Yeung et al., 2015b; Shou et al.; Yuan et al., 2017], sequence labeling [Yeung et al., 2015a; Singh et al., 2016; Ma et al., 2016; Shou et al., 2017; **?**] and activity prediction [Koppula et al., 2013; Koppula and Saxena, 2016], etc.

While most existing works focus on parsing activities in RGB videos, multi-modal analysis have attracted relatively less attention. However, recent advances in depth sensors (e.g., Kinect) enable us to efficiently obtain dense 3D measurements of dynamic environments, especially with indoor settings. Exploiting such multi-modal videos for activity understanding has promising prospect as they provide rich information about the object poses and shapes, and are less sensitive to object appearance and lighting condition [Wang et al., 2012a; Oreifej and Liu, 2013; Ni et al., 2013a; Shahroudy et al., 2016; Lillo et al., 2016].

Despite the progress in video-based action recognition, it remains a challenging task to fully parse complex activities that consists of a sequence of different actions in long video sequences. A key step towards understanding such activities is to produce a detailed frame-wise action labeling, which receives much less attention than other tasks in activity understanding (e.g., action classification or detection). Previous work typically focus on predicting action class labels for each frame without explicitly reasoning action instances in the sequence [Yeung et al., 2015a; Lillo et al., 2016; **?**; Yuan et al., 2017; Shou et al., 2017]. Nevertheless, such category-level frame-wise labeling strategy is limited in encoding more global constraints at the action instance level, which tends to produce inconsistent frame-level labeling for the entire sequence.

In this work, we propose to incorporate action instance information into detailed action labeling of complex activities in long videos by fusing the frame-level and

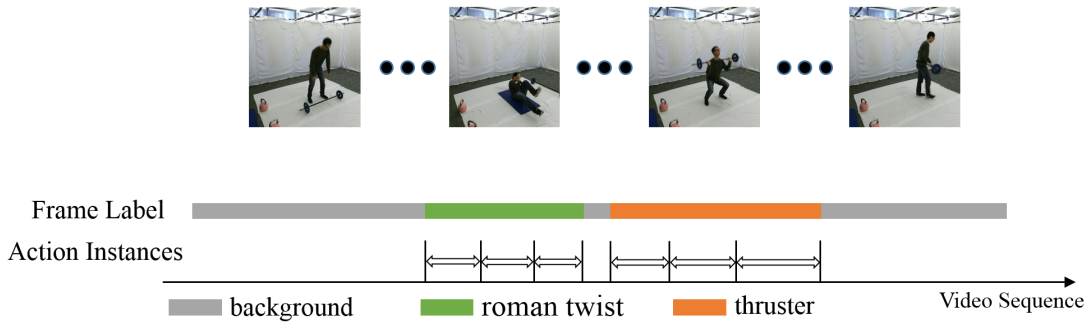


Figure 4.1: Dense sequence labeling & Detection. Each frame has an action class label, a set of consecutive frames forms an action instance. Note that several action instances can repeat right after each other.

instance-level action predictions. Here sequence labeling is performed in an online fashion, with prediction being made based on local observations and sequence history, while detection seeks action instances based on the spatio-temporal patterns of the full actions. As they capture complementary information at both instance and frame level, our approach exploits the synergy between these two subtasks, and can thus handle arbitrary long videos with consistent performance in parsing multiple action instances. We also note that action classification techniques are not suitable for labeling tasks because in labeling, the total video length and the action duration are unknown.

We design a deep neural network pipeline using both RNN and CNN to perform frame-level action labeling for complex videos with multi-class action instances. Our network consists of two interdependent modules for detection and labeling respectively. The action labeling module is a recurrent network based on the long short-term memory (LSTMs) [Donahue et al., 2015], while the detection is performed by a stack-frame CNN [Karpathy et al., 2014], which takes a set of sampled frames and predicts action scores for every generated action proposal. To achieve consistent labeling, we integrate the predictions from the action detection module with an LSTM network for a refined sequence labeling results. Both the networks are built on a short-term video representation that integrates RGB with depth cues, which provides the geometric information from the raw depth and its normal map. Finally, the representation is further enhanced by employing dynamic images [Bilen et al., 2016a; Fernando et al., 2015a].

We adopt a stage-wise training and inference approach to integrate action predictions from two sub-networks.

To evaluate our method, we utilise the GADD dataset described in the previous Chapter. It is specifically designed for sequence labeling and action detection with dense frame level labels. Unlike existing RGBD action recognition or detection datasets, videos in GADD include more complex activities, typically defined by a continuous sequence of actions. Also, the proposed method is tested on the Kitchen

Scene Context dataset (KSCGR) [Shimada et al., 2013] and compare with several baseline methods. Our network outperforms all the baseline methods and has proved to be beneficial to both the subtasks.

Our main contributions are two-fold:

1. We design a novel video representation that integrates short-term spatio-temporal patterns of both and RGB and depth information using dynamic images.
2. We develop a novel fusion method that explores the synergy between the two subtasks of action instance detection and frame-level action labeling, which results in better overall performance for both tasks.

4.1 Related Work

4.1.1 Action Recognition & Video Representation

Depth Based Some of the famous video descriptors are STIP[Laptev, 2005], HOG[Dalal and Triggs, 2005], HOG3D[Klaser et al., 2008], optical flow and trajectories[Wang et al., 2013][Wang and Schmid, 2013]. However, most of the interest points based methods cannot be easily applied to depth sequences due to lack of textures in depth images. Recent effort in action recognition in depth videos developed alternate methods.

For example, early work in depth-based action recognition typically use hand-crafted depth features, e.g., [Li et al., 2010b]. Recently, [Wang et al., 2012b] extract human skeleton from depth, then track the joints as the action features using the skeleton tracking algorithm [Shotton et al., 2013; Taha et al., 2015]. The main drawback of the estimated skeletons is that they can be noisy in a clutter scenes or under occlusion. Some holistic approach have also been proposed. In [Yang et al., 2012], the entire depth sequence is summarised into one Depth Motion Map (DMM) by accumulating the difference between frames, then HOG features are computed using the DMM. The main draw back of this method is that it fails to consider the temporal ordering, which we argue is vital in action recognition. Unlike it, the temporal dynamics is explicitly considered in [Oreifej and Liu, 2013], where the histograms of 4D normal vectors are computed evenly divided spatio-temporal cells. However, such a representation is sensitive to the noise in temporal gradients.

An alternative approach to depth-based action modelling relies on dense representations. In [Wang et al., 2012a], the authors use randomly sampling strategy to compute the occupancy patterns in the spatio-temporal volume of depth sequences.

Deep Network Based Recently, deep network based representations have been widely adopted in activity analysis of RGB videos[Herath et al., 2016] due to its robustness. Both [Karpathy et al., 2014] and [Simonyan and Zisserman, 2014a] use stacked frames as input and relies on the CNN network itself to extract temporal information. Many others resorts to the LSTM network to learn an action representation [Yue-Hei Ng et al., 2015; Donahue et al., 2015]. [Veeriah et al., 2015] modifies

the LSTM internal gates and propose Derivative of States (DoS) to get a video representation that emphasises the motion salience. In terms of representing temporal dynamics, it is natural to extend the powerful 2D CNN into 3D domain, by extending the input and every convolutional kernel to 3D. Following this idea, [Ji et al., 2013] and [Tran et al., 2015a] propose 3D convolutional neural network. Another series of work [Fernando et al., 2015a,b; Bilen et al., 2016a] aims to summarise the complex temporal dynamics into one compact image, which can be used by 2D CNNs for recognition. [Wang et al., 2017a] extends this idea to optical flow images, further improves recognition accuracy.

4.1.2 Action Detection & Sequence Labeling

Most action detection approaches aim to localise instances of a specific action class based on classifying generated action proposals. For example, [Ni et al., 2014] combines dense trajectories and frame level CNN features to detect actions from sliding window proposals. Such strategies have been improved by effective methods of generating high quality action proposals as in [Yu and Yuan, 2015]. To capture temporal dynamics, LSTM networks have been widely used in representing action instances. Singh and Shao [Singh et al., 2016] propose an LSTM on top of a multi-stream CNN to model the dynamics for each proposal. [Yeung et al., 2015b] use LSTM and reinforcement learning technique to progressively observe the video and refine its prediction on where to look next and when to make a decision. Other methods use structured representations to model the details of action instances. [Yuan et al., 2017] improves detection accuracy by explicitly finding the start, middle and end key frames of an action. Actions are commonly accompanied by objects, [Ni et al., 2016] build on this idea and propose detecting interactional objects and body parts using an object parsing network. They then extract motion features like HOF and trajectories on the parsed segments. RGB-D information has also been used in action detection literature [Ni et al., 2013a; Shahroudy et al., 2016]. [Ni et al., 2013a] extracts 3D spatial-temporal context of the actions using both grey scale and depth images. Shahroudy *et al.* [Shahroudy et al., 2016] extract human body part from depth data and propose a part-aware LSTM network for recognition and detection. However, it is challenging for those methods to produce a consistent parsing of long videos with consecutive actions from multiple classes.

A more detailed approach to understanding complex activities in videos is through sequence labeling. One key challenge here is to achieve temporal consistency in the labeling. To better model temporal dynamics, Yeung *et al.* [Yeung et al., 2015a] propose an attention based MultiLSTM model with multi-label loss that intelligently select input frames and produce multiple outputs for a range of frames. [Shou et al., 2017; **?**] utilise temporal convolution features and deconvolution operations to extract high level temporal dynamics for end-to-end sequence labeling. Lillo *et al.* [Lillo et al., 2016] decompose a complex action into atomic actions and propose a pose based method to detect atomic actions. Ma *et al.* [Ma et al., 2016] designed a novel ranking loss to enforce the monotonicity of prediction scores, which encodes the ac-

tivity progression constraint. By contrast, we integrate the detection with sequence labeling to achieve more consistent results. In addition, our work focus on learning action labeling from densely labeled videos, while some recent work [Huang et al., 2016; Papoutsakis et al., 2017] take unsupervised or weakly supervised approaches.

It is also appropriate to briefly review the methods of object detection since action detection can be seen as the temporal version (1d) of object detection. Recently, a line of works [Girshick et al., 2014; Girshick, 2015; Ren et al., 2015] have demonstrated near real-time high accuracy object detection in the wild. [Girshick et al., 2014] adopts a multi-stage detection pipeline. First extract region proposal on the input image, then compute CNN feature on each proposal and finally classify each region using linear SVMs. [Ren et al., 2015] improve the proposal generation process by applying a Region Proposal Network (FPN) on the convolution feature maps that enable enable end-to-end training. This way, full-image convolution features are shared to allow very fast region proposals. Such techniques may be adopted for action detection by extending the methods along time dimension using 3D convolution like [Shou et al.], but due hardware limitations, it is not feasible for longer actions. Single Stage Detectors [Liu et al., 2016a] (SSD) achieves even faster speed by removing the proposal generations stage. Instead, they use fixed prior boxes as anchors and they only require one forward pass during inference to predict boxes. Also, the speed is invariant to the number of object instances.

4.2 Methodology

We now address the problem of labeling in the complex videos that comprises a sequence of action instances from multiple action classes. We aim to integrate global information from action detection to achieve consistent labeling for parsing complex activities. To this end, we design an instance-aware action labeling system which consists of three components: 1) a frame-wise labeling LSTM, 2) an action detection system to acquire global information at instance level, and 3) a fusion network that integrate both local and global information for consistent sequence labeling.

A good frame representation is the basics in almost all video understanding tasks. In the proposed labeling framework, we employ a novel video representation that integrates short-term spatio-temporal patterns of both and RGB and depth information utilise the techniques of dynamic images [Bilen et al., 2016a; Fernando et al., 2015a]. Therefore, we first introduce the frame representation in this section, then we describe each model component in detail.

4.2.1 Frame Representation

Our labeling and detection task requires capturing spatio-temporal patterns of the sequence at multiple granularities. To this end, we adopt a local frame representation that encodes the short-term temporal dynamics for the frame-level labeling and detection tasks. Specifically, we first compute a dynamic image feature and a single-frame feature, and then fuse them by concatenating their CNN representations.

Dynamic Images We represent the short-term spatio-temporal patterns by exploring the concept of dynamic images [Bilen et al., 2016a] based on rank pooling [Fernando et al., 2015b]. The original dynamic image (DI) aims to summarise the temporal evolution of a video with a single image representation. Specifically, it learns a video representation by exploiting temporal ordering of the frames. Given a video sequence $\{x_1, x_2 ... x_T\}$, the method associates a score S_t with each frame, which is defined by a linear SVM: $S_t(v_t, u) = u^T \cdot v_t$ where v_t is the locally aggregated feature up to the tth frame: $v_t = \frac{1}{t}\sum_{\tau=1}^{t} \phi(x_t)$, and u is the parameter vector of the SVM. The linear SVM needs to satisfy the ranking constraint, which ensures that for any two frames x_p and x_q:

$$S_p(v_p, u) < S_q(v_q, u), \ \forall \ p < q \tag{4.1}$$

We refer the reader to [Fernando et al., 2015b] for the details of the SVM learning process. Once the SVM is trained based on the temporal ordering supervision, the weight vector u that characterize the SVM is used as the video descriptor.

In this work, we use an efficient approximation of dynamic images, which designs a fast rank pooling strategy. Concretely, the fast DI is computed as follows,

$$d = \sum_{t=1}^{T} \left[2(T - t + 1) - (T + 1)(H_T - H_{t-1})\right] x_t \tag{4.2}$$

where the coefficient $H_t = \sum_{\tau=1}^{t} \frac{1}{\tau}$. This fast approximation is essentially a weighted sum of the input frames. As in [Bilen et al., 2016a], we compute the dynamic images directly on the frame pixels, and the resulting d is a array of the same shape as the input frames features.

Figure 4.2 illustrates how we implement the dynamic images to get a frame representation. The DIs is applied to encode the local temporal windows centered at the video frames of interest. Our preliminary empirical study shows the DI is more effective to capture the short-term dynamics than being applied to long sequences. We therefore choose to use $n = 7$ frames centered at the frame f_t to construct the dynamic images at frame f_t. This way a dynamic image summarises the local motions over approximately a quarter of a second. As is shown later in the experiments section, we pre-compute such features for all frames for all experiments to use.

Single-frame Features We use slightly different single-frame features depending on the input is RGB or depth videos. For RGB videos, the single-frame feature is the raw RGB image. For depth videos, we also compute 3D surface normal map in addition to the raw depth frame. The normal vectors is able to better describe the surface shape [Oreifej and Liu, 2013]. Formally, the normal vector $\mathcal{N}$ at a point (x_0, y_0) on a surface $z = f(x, y)$ is given by

$$\mathcal{N} = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, -1]^T \tag{4.3}$$

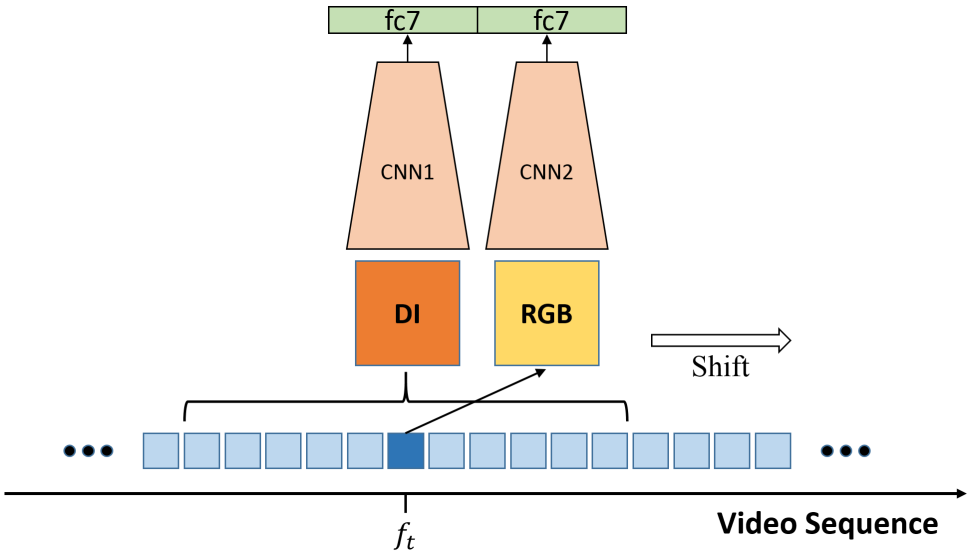


Figure 4.2: Representation for frame f_t. We use a window of n frames centered at the f_t to compute dynamic image. The dynamic image summarise the short term motion cue around f_t.

We first apply median filter to the depth images, removing most of the noisy edges and filling in holes in the raw depth. Then we calculate numerical gradient by calculating finite difference on every point both horizontally and vertically, obtaining a two-channel matrix corresponding to $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ respectively. We use the two channel matrix as the gradient normal map.

CNN-based Feature Fusion Given the dynamic image and single-frame features, we now develop a feature representation for the local temporal window of m consecutive video frames adopting a late fusion strategy that uses a two-stream CNN [Simonyan and Zisserman, 2014a]. As the DI and single frame features possess the same shape as the input images, we can easily concatenate them and feed into a CNN for late fusion. Instead of relying on optical flow which is often noisy and expensive to compute, we use a stream of CNN to extract $fc7$ feature from the DI, which is more effective for capturing the short-term dynamics. A second stream of CNN computes the $fc7$ features from the single-frame feature, and we concatenate these two features together to represent the frame. Figure 4.2 gives an overview of the frame representation. It should be noted that to represent a video segment, we simply stack such frame representations as the input to CNN, as depicted in Figure 4.4

4.2.2 Frame-wise Action Labeling

To predict frame-wise action labels, we build an LSTM-based recurrent neural network on top of our frame representation. The RNN allows us to encode the long-term

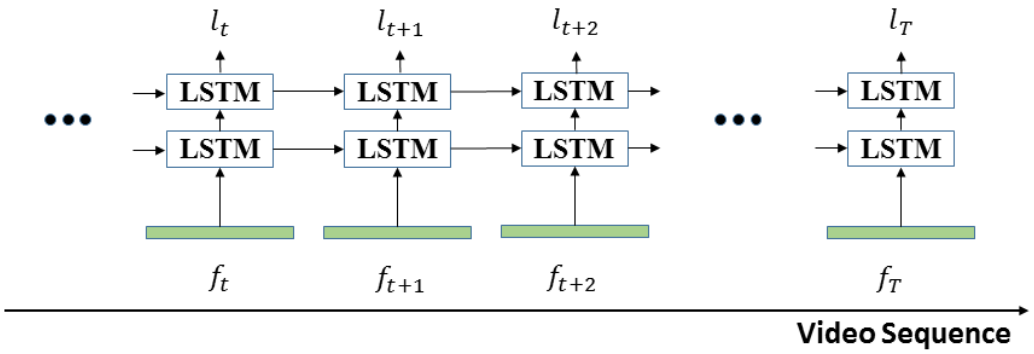


Figure 4.3: Overview of Sequence Labeling. We stack two layers of LSTM on top of the CNN feature to encode long-term temporal information.

dynamics of the activities so that our prediction exploits both current and history observations.

Specifically, at each given time step, we first compute the local frame representation described in Section 4.2.1 as the input to the LSTM units. Please note that the computation of dynamic images exploits current, future and past observations to capture more context information. Our RNN consists of two layers of stacked LSTM with 512 hidden states, and the hidden representation of the top layer units are used to generate the probability of action classes through a softmax layer. We refer the reader to [Graves et al., 2013; Donahue et al., 2015] for the detailed equations of the LSTM units.

Figure 4.3 shows an overview of the sequence labeling network.

To handle action labeling for general video sequences, we use the stateful LSTM to accommodate inputs that contain multiple instances and of variable lengths. The stateful LSTM accepts input of the current frames, but inherits the hidden state values from its previous iteration. This way, there is no need to specify the sequence length when initializing LSTM units. More importantly, in complex and long videos, a RNN with large temporal unroll step is not optimal because multiple actions with different patterns may be present within one temporal unroll and weights update. With stateful LSTM, the time unroll step can be set to be much smaller than the video length yet still capture long-term dynamics because of its state-inheritance nature. After all frames of a video are processed, the LSTM layer will reset its hidden state to make ready for the next video.

4.2.3 Action Detection with CNNs

We consider the task of action detection in the context of understanding complex activities, which typically involves a set of consecutive action instances from multiple classes. To efficiently localise these actions, we develop a two-stage detection method based on the Stack-frame CNN network [Shou et al.]. Our detection pipeline takes a

video as input, and first generates a pool of generic action proposals using an efficient CNN with binary output. We then learn a second CNN to classify the proposals into multiple classes. Both networks have the structure of stack-frame two-stream CNN that takes the stacked frame representation as input. The $fc7$ features of the two streams are concatenated and fed into a linear SVM for the final output. Figure 4.4 shows the overview of the detection module.

Action proposal generation . We reduce the search space for the action detection by first filtering out background and misaligned candidate windows. To achieve this, we build a CNN-based binary classifier, which is then applied to the input video sequence in a sliding window manner. The filter CNN predicts an actionness score for each window in its exhaustive search. The actionness score is the probability of the window being an action instance. Specifically, we evenly sample $k = 10$ frames and stack them together as the input to the CNN for each window. Our network uses the backbone of AlexNet [Krizhevsky et al., 2012] for efficiency, and shares the structure of the AlexNet except the input and output layers. Our network takes input that has 10 channels, and predict the probability of the window being an action instance.

We generate a pool of action proposals by removing the windows with probabilities lower than a threshold. We validate the threshold to strike a balance between speed and accuracy. In this work, it is empirically set to 0.8, which results in a 95% reduction in the number of windows and still maintains a recall rate of 1 with 0.5 IoU threshold.

Action instance classification . For each candidate window that passes the filter CNN, a multi-class CNN is used to predict the action label and its confidence scores. We further apply the non-maximum suppression (NMS) with a threshold of 0.3 to generate the final detection outcomes.

Concretely, we evenly select $k = 10$ frames from each sample window and stack the frame representations. Note that each dynamic image is computed using 5 consecutive frames centered at the frame of interest. By doing so, the actual frames used to obtain the video representation extends slightly beyond the windows itself, giving it extra context information [Girshick et al., 2014].

4.2.4 Instance-Aware Action Labeling Network

We now integrate the frame-wise action labeling module with the action detection module to achieve more consistent parsing of the input videos. The action labeling network captures rich context at the frame and category level while the detection module extracts instance-level information for the action units. In this work, we refer a instance as a complete sequence of any defined action classes in the dataset. The goal of the detection module is to extract such instances from unconstrained videos. By combining the frame labelling network and the detection module, a novel two-stage labeling process is used to exploit the synergy between those two modules.

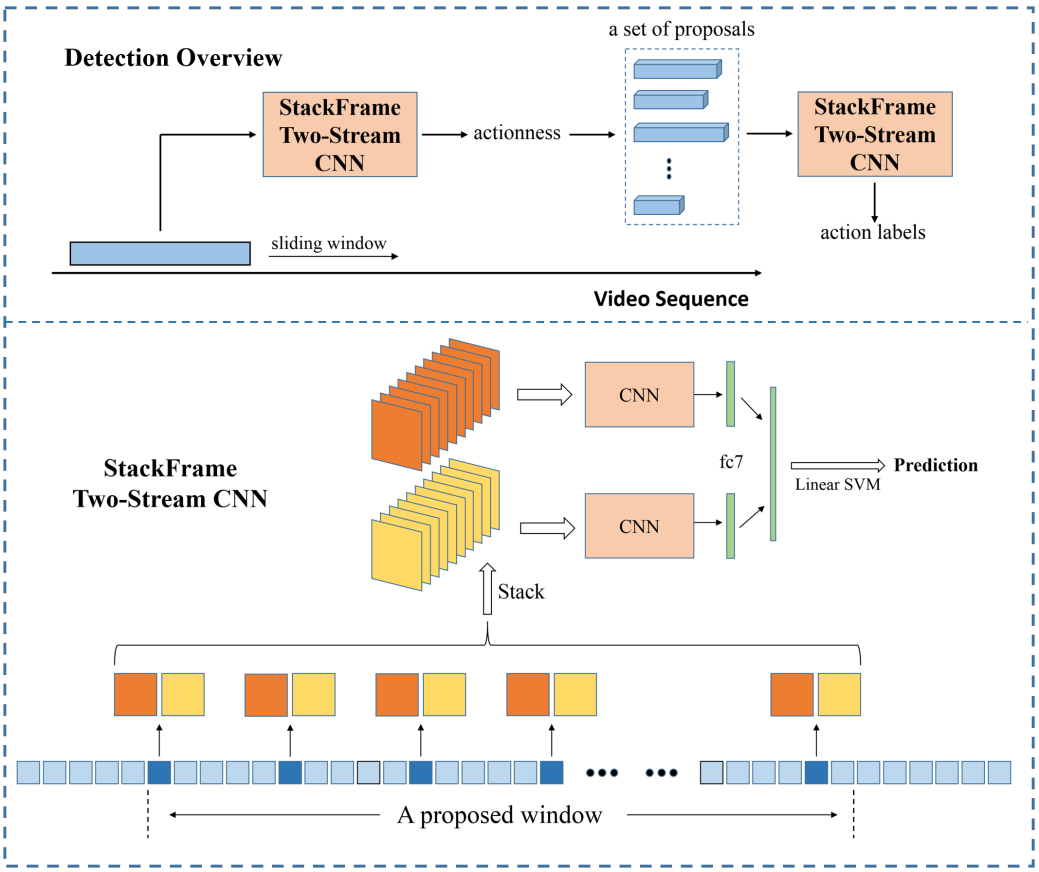


Figure 4.4: Overview of action detection module. We first compute the frame representation on 10 evenly sampled frames in the window. We then apply the two-stream stack-frame CNN. Note that in the process of obtaining dynamic images, frames outside the proposed window is used, which provides extra context information.

Our first stage trains the detection module to generate a set of temporal bounding boxes, each box is associated with an action class label and a confidence score. We then use the detection results to compute a Frame Label Prior matrix (FLP). Specifically, let the detected actions of class l be a set of N_l temporal windows, denoted by $\{(s_i^l, e_i^l, c_i^l)\}_{i=1}^{N_l}$, where s_i^l, e_i^l are the start and end frame index, and c_i^l is the confidence score. For each action class, we select top $M = 10$ windows with highest confidence scores and sort them according to the scores. Denote the total number of action classes as L. At each time step t, we define a label prior matrix D_t of size M by L as follows,

$$D_t(m, l) = c_m^l \mathbb{1}(t \in [s_m^l, e_m^l]) \tag{4.4}$$

In the second stage, we design a two-branch fusion network to integrate the detection outcomes with the frame-wise action labeling task. Using the FLP matrix as input, we construct a convolution branch to encode the matrix into a feature vector containing the action instance information. Our convolution branch has two con-

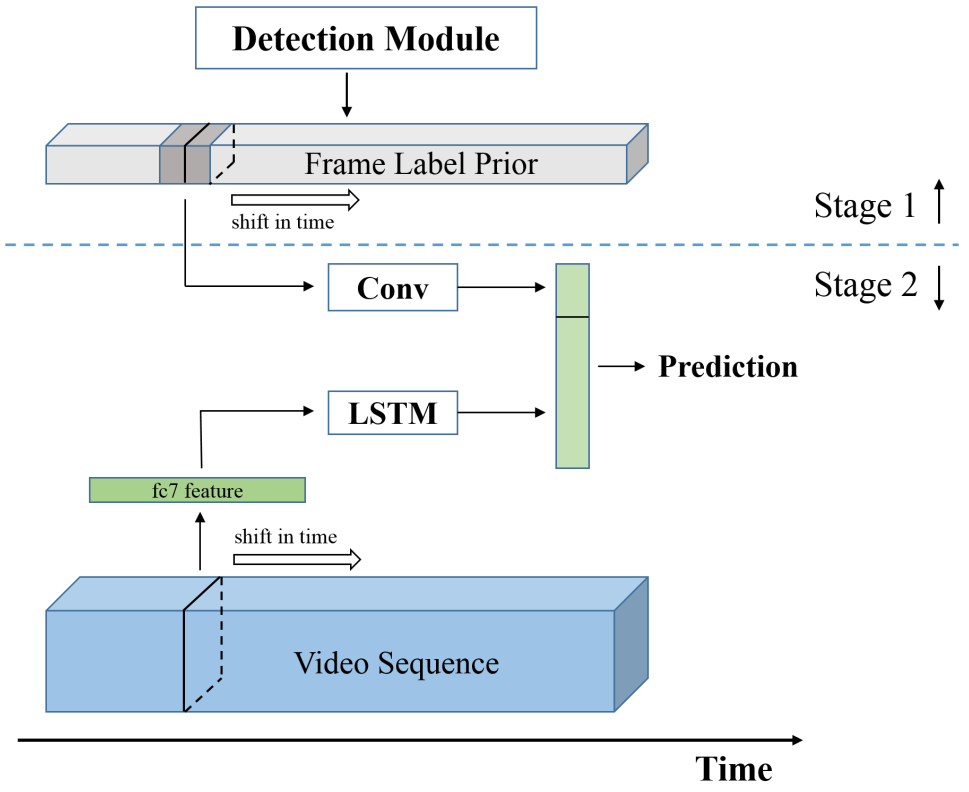


Figure 4.5: Two branch labeling network. The detection output around frame at time t (dark grey) is encoded using a small convolution network. The encoded instance cue is concatenated with the frame-level feature extracted by LSTM for prediction. Frame level features is computed according to Sec 4.2.1.

volution layers followed by a fully connected layer with 64 output neurones. The output is concatenated with the hidden vector of the top-layer LSTM for the frame label prediction.

Figure 4.5 illustrates the structure of fusion network. During the prediction of frame label at time t, a slice of the array FLP around time t serves as input to a convolution branch. We jointly train the instance-aware labeling network end-to-end to obtain the final labeling prediction, and to ensure the CNN branch learns how to correctly extract action instance knowledge from FLP.

4.3 Experiments

We evaluate our instance-aware sequence labeling method on the RGB and depth videos in the GADD 3 and the Kitchen Scene dataset KSCGR [Shimada et al., 2013]. We first describe the experiment setup and evaluation metrics in Sec. 4.3.1. Then, we report the results on sequence labeling with comparisons to several strong baseline

methods, which is the focus of this paper. Finally, we show through an ablation study that not only detection can help improve labeling, but labeling can boost detection performance as well, indicating that action detection and sequence labeling are mutually beneficial.

4.3.1 Data Preparation and Evaluation

Dataset Split & Training details For both datasets, we split the dataset into two training sets **TR1** and **TR2** as well as a test set **TST**. For the training of the fusion network, we use **TR1** training set to learn the detection system first and **TR2** training set to train the instance-aware labeling model.

For the stack CNN in action detection system, we train the network on TR1 and adopt the Adam optimiser with base learning rate is 0.0001 and batch size 256. For the frame-wise LSTM labeling network, we use the SGD optimiser with a base learning rate 0.01 and batch size 1. This is also trained on TR1. Then we apply the trained detection CNN and frame-wise LSTM on TR2 to obtain the detection and labeling results, which are used to train the instance-aware labeling net on TR2. For this network, we use the Adam optimiser with base learning rate 0.00001 and batch size 1.

Evaluation Metrics We employ two different evaluation metrics for the task of sequence labeling. One is *accuracy* which reflects the frame level performance (how many frames are correctly labeled). The other is *f-score* which is calculated with *precision* and *recall*. The *f-score* reflects the class specific instance level performance.

For the task of action detection, we compute the intersection-over-union (IoU) between the predicted temporal windows and the ground truth and consider the detection is correct if IoU$>$ 0.5. We then use the average precision to report the results on each class and the mean average precision (mAP) for overall performance.

4.3.2 Evaluation on Sequence Labeling

We show the results of sequence labeling in Table 4.1 and Table 4.2 for RGB and depth video respectively. We compare the performance of the basic LSTM model, three state-of-the-art methods [Yeung et al., 2015a; Singh et al., 2016; Ma et al., 2016] and our fusion LSTM model based on the same video representation. To put our instance-aware fusion method in context, we also implemented a naive smoothing method (majority voting) for comparison.

First, we note that by modelling the long-term dynamics with the basic LSTM, we witness significant improvements compared to the traditional CNN. Attention LSTM performs slightly better than basic LSTM, thanks to its attention mechanism that is capable of focusing on relevant frames and capturing longer-range dynamics. By introducing temporal consistency constraints [Singh et al., 2016; Ma et al., 2016], it also achieves slightly better performances. However, our instance-aware LSTM network fused with detection can effectively model global dynamics, outperforming

Figure 4.6: Comparing different sequence labeling results. *Black* is background, *Green* is an action, and *Red* is false prediction. **First row**: ground truth. **Second row**: labeling results with CNN only. **Third row**: labeling results with added LSTM. **Forth row**: attention LSTM labeling results. **Fifth row**: instance-aware labeling results. **Sixth row**: Smoothing applied on basic LSTM results. *Video 1* and *Video 2* illustrate how joint prediction can help improve accuracy by considering global information at instance level. *Video 3* shows a failure case due to poor-quality detections.

	Accuracy(%)	f score
CNN	76.32	0.72
LSTM	78.45	0.74
Attention LSTM[Yeung et al., 2015a]	80.27	0.74
Bi-LSTM[Singh et al., 2016]	79.88	0.75
RankingLoss LSTM[Ma et al., 2016]	80.75	0.76
LSTM (with smoothing)	80.33	0.73
Ours (instance-aware)	**83.78**	**0.81**

Table 4.1: Comparing sequence labeling accuracy and F-score of our proposed method against other state-of-the-art methods on RGB videos on GADD. Noticeable improvement can be seen in both evaluation metrics.

the other state-of-art methods consistently. The fusion of detection and sequence labeling generates a noticeable improvement in both accuracy and F-score, indicating the synergy between action detection and sequence labeling. Moreover, comparing the last two rows in the table, we can see moderate improvements on accuracy but large jumps in f-score. This clearly indicates that our instance-aware labeling method works more effectively for action instances, while naive smoothing methods that treat every frames equally are prone to errors. Although naive smoothing operation can improve accuracy to some extent, most improvements is happening within background frames. When applying such operation on complex sequences, it will blur the instance boundaries, leading to no improvements or even worse results in f-score.

Figure 4.6 visually shows the labeling results with and without the use of the proposed instance-aware technique. Figure 4.7 gives a detailed class-by-class comparison on the *f-score* induced by the fusion of action detection. We can see that our joint method outperforms the basic LSTM across all the action classes.

Concretely, we can make several observations from Figure 4.6. First, by compar-

	Accuracy(%)	f score
CNN	78.86	0.69
LSTM	81.25	0.78
Attention LSTM[Yeung et al., 2015a]	81.36	0.77
Bi-LSTM[Singh et al., 2016]	81.87	0.79
RankingLoss LSTM[Ma et al., 2016]	82.03	0.79
LSTM (with smoothing)	82.13	0.78
Ours (instance-aware)	**84.56**	**0.86**

Table 4.2: Comparing sequence labeling accuracy and F-score of our proposed method against other state-of-the-art methods on Depth videos on GADD. Noticeable improvement can be seen in both evaluation metrics.

ing the third row to the second, we can see that the basic LSTM mainly corrects the short chunks of false prediction inside an action, which produces a smooth labeling outcomes. Second, from the forth row, we observe that the behavior of attention LSTM has large variations: while it does capture longer-term dynamics, it also seems to attend to irrelevant frames. In contrast, as shown by the difference between the third and fifth row, by fusing the detection results, our method effectively corrects false predictions within the interior of action instances, as well as during the transition between non-action (background) and action, where the frame features can be ambiguous. Furthermore, in comparison with the sixth row, it is evident that the proposed instance-aware method can handle false predictions more adaptively. Naive smoothing methods are ineffective when a set of consecutive frames are predicted falsely.

The performance improvement of our method is likely due to the instance awareness introduced by our method. Within a complexity action sequence, the frame level prediction inferred by short term dynamics can easily make mistakes as the basic movements of human limbs that form a complex action are likely to appear in some other actions, which causes confusion with local information only. The problem can be alleviated by joint prediction with more global contextual information from sequence history and action instances. What's more, the frame labels during the transition between non-action and action are ambiguous in nature. If a correctly detected window has high IoU for an action instance, it can facilitate labeling of the frames during transition. However, it is noteworthy that combining detection results of poor quality can sometimes lead to deteriorated results, as shown in *Video 3* in Figure 4.6.

We also report our sequence labeling performance on the KSCGR dataset in Table 4.3, which summarizes the accuracy and F-score for action labeling with comparisons to baselines. For completeness, we also include several state-of-the-art methods using RGBD as input. It is shown that our method outperforms baseline methods and achieve better or comparable results against state-of-the art methods. This indicates that our improvement over existing methods is consistent across different

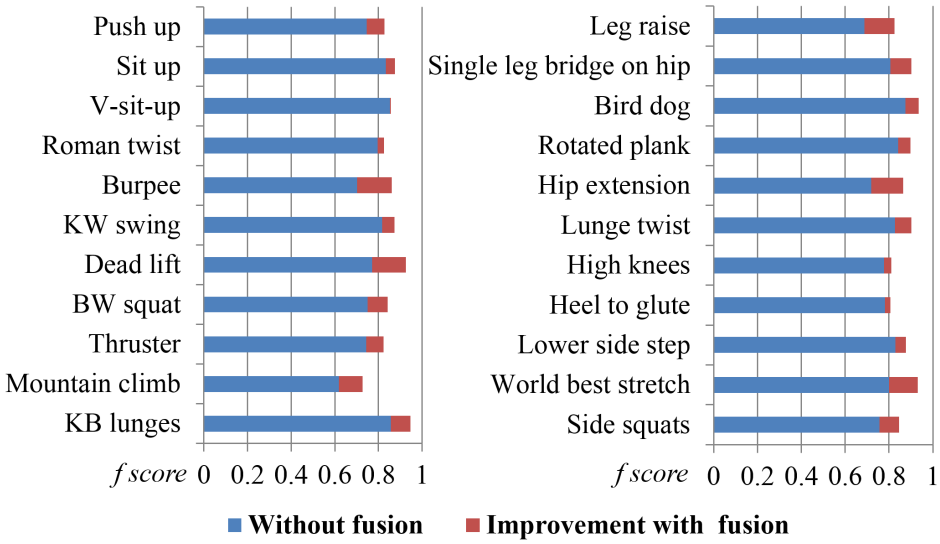


Figure 4.7: Per-class F-score of sequence labeling on GADD. After applying fusion with detected action instances, we witness performance gain on all the action classes.

datasets.

4.3.3 Ablation Study

4.3.3.1 Effect of Depth Video Representations

To validate our design of depth video representation introduced in Section 4.2.1, we compare frame-based sequence labeling accuracy against depth representations in Table 4.4. Here we predict the frame label based on a local window centered at each frame as shown in Figure 4.2. We can see that using depth alone leads to poor performance while dynamic images achieves reasonable results on its own. Combining depth with dynamic image (DI) boosts the labeling performance slightly (by 1.11%), and adding surface normal cues further improve the accuracy (by 1.71%). A pixel value of a depth image only indicate how far a point is from the camera plane, but surface normal describes the curvature around a small area. The results shows the temporal information from dynamic images and the spatial information from 3D surface normal vectors are indeed complementary to each other. Therefore, we use the *depth+DI+3D norm* as our frame representation for all of our detection and labeling experiments.

4.3.3.2 Effect of Instance-aware Fusion

While we have already demonstrated that our fusion method can significantly improve labeling performance in Figure 4.7, we now demonstrate that the labeling outcomes can help boost detection performance as well.

	accuracy(%)	f score
Doman and Kuai[Shimada et al., 2013]	-	0.74
IAT-Action[Ni et al., 2014]	-	0.79
Ni[Ni et al., 2016]	-	0.86
Depth+DI+Norm	61.3	0.58
LSTM	70.2	0.71
Instance aware (Depth)	74.0	0.74
Instance aware (RGBD)	78.5	0.84

Table 4.3: Sequence labeling accuracy and F-score on KSCGR. Our instance aware model, using only depth data as input, achieves comparable result with those using RGB or RGBD data input. When combine the results of both RGB and Depth inputs, our method output performs all baseline methods.

Representation	**Accuracy**(%)
depth alone	36.60
DI alone	76.04
depth+DI	77.15
depth+3D norm	66.90
depth+DI+3D norm (stack input)	**78.86**
depth+DI+3D norm (with LSTM)	**81.25**

Table 4.4: Sequence labeling accuracy with different depth representations. The combination of depth, dynamic image and 3D norm achieved the best result.

Table 4.5 and 4.6 shows the mAP scores and after re-weighting the detection score with labeling results on GADD and KSCGR datasets respectively. We can see that there is a significant improvement in the overall mAP. When the detection system localises the action instances, their class labels can be noisy due to lack of long-term temporal context. On the other hand, the action labeling can provide more stable results regarding the action class for each frame. By fusing them together, we are able to reduce the false positives from incorrect action classes and achieve better performance.

Video modality	**mAP** @0.5	**with SL** @0.5	**mAP** @0.7	**with SL** @0.7
RGB	39.98	44.38	33.55	40.62
Depth	35.59	45.81	31.70	40.89

Table 4.5: Mean Average Precision for action detection for RGB and depth videos on GADD. The IoU threshold is set to 0.5 and 0.7 during evaluation.

Figure 4.8 and 4.9 shows detailed class-by-class improvement on the *average precision* induced by the fusion of sequence labeling. Figure 4.10 shows the typical

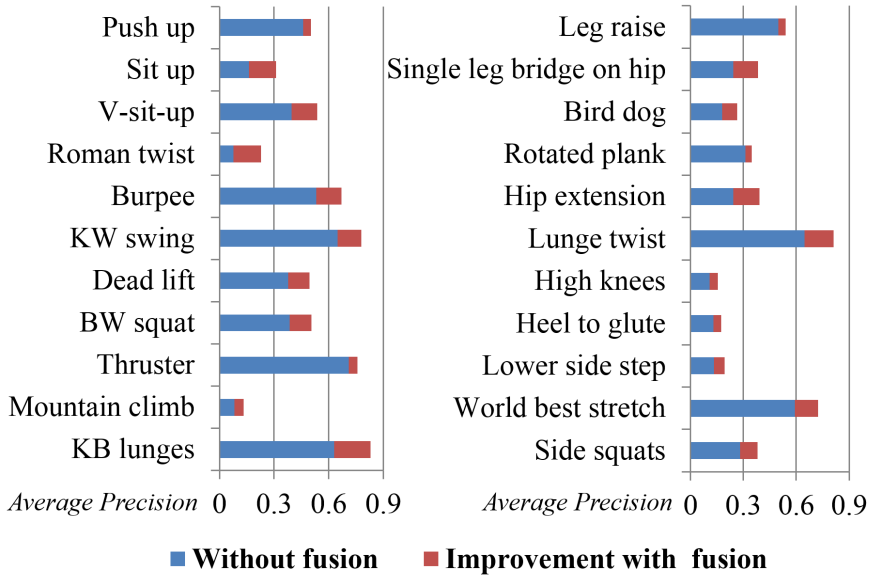


Figure 4.8: Per-class average precision of action detection on GADD. By the fusion with sequence labeling, we witness performance gain on all the action classes.

Video modality	**mAP** @0.5	**with SL** @0.5	**mAP** @0.7	**with SL** @0.7
RGB	36.1	43.8	25.5	31.2
Depth	32.3	38.2	21.9	26.3

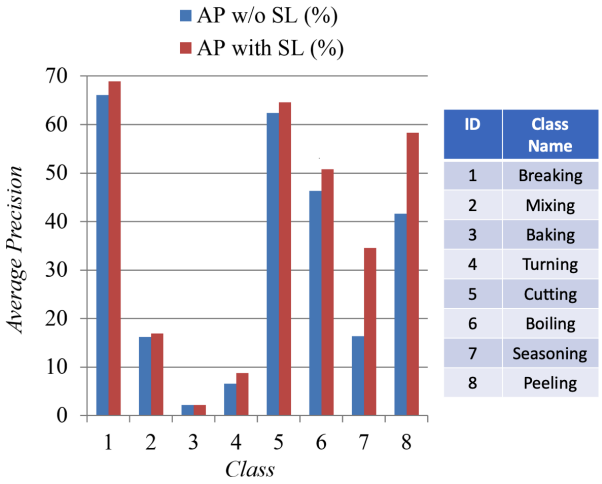
Table 4.6: Mean Average Precision for action detection for RGB and depth videos on KSCGR. The IoU threshold is set to 0.5 and 0.7 during evaluation.

precision recall curves of detection from several action classes. The left panel shows three action classes with highest PR curves and the three with lowest ones. The right panel demonstrates the improvement from fusion for two typical action classes, which is evident.

4.4 Conclusion

In this chapter we have presented an instance-aware fusion approach for dense activities labeling in complex videos, which combines the frame level information from LSTM with instance level information from action detection. By exploring the synergy between action detection and sequence labeling, we saw consistent improvements on both sequence labeling and action detection tasks over strong baselines methods on GADD and KSCGR datasets.

What's more, we propose a novel depth video representation which utilises dynamic images and 3D surface normal, and explore the synergy between action de-



ID	Class Name
1	Breaking
2	Mixing
3	Baking
4	Turning
5	Cutting
6	Boiling
7	Seasoning
8	Peeling

Figure 4.9: Per-class average precision of action detection on KSCGR. By the fusion with sequence labeling, we witness performance gain on all the action classes.

tection and sequence labeling. We design an LSTM + CNN fusion network to jointly solve two problems, leading to better performances in both tasks.

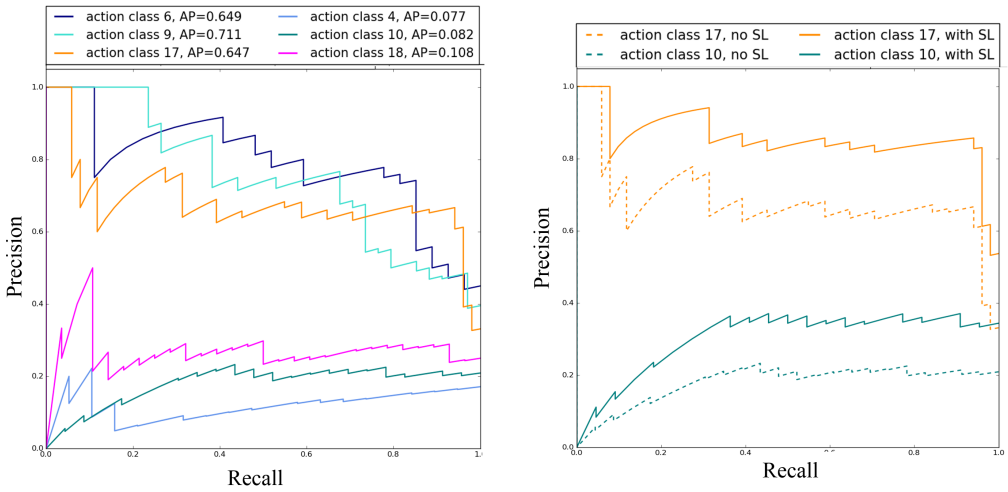


Figure 4.10: Precision Recall Curve on GADD. **Left**: The graph shows three action classes with high-quality detection and three classes with low-quality detection. **Right**: The graph shows the effect of sequence labeling fusion on detection for two action classes. Refer to Table 3.1 for the class names used in this figure.

Chapter 5

Few-shot Action localisation

As discussed in Chapter 4, temporal action localisation, which jointly classifies action instances and localises them in an untrimmed video, is a key task in video understanding. Due to a wide range of applications such as video surveillance and video analytics [Poppe, 2010; Herath et al., 2016], it has been an active research topic in recent years [Lillo et al., 2016; Ni et al., 2016; Singh et al., 2016; Yeung et al., 2016; Shou et al.; Xu et al., 2017; Girshick, 2015; Yuan et al., 2017; Zhao et al., 2017; Shou et al., 2017; **?**; Gao et al., 2017; Ma et al., 2016; Xiong et al., 2017; Dai et al., 2017]. While early attempts relied on handcrafted video features [Poppe, 2010; Karaman et al., 2014; Wang et al., 2014b; Lillo et al., 2016], deep neural network based approaches have made significant progress [Donahue et al., 2015; Karpathy et al., 2014; Shou et al.; Xu et al., 2017; Yeung et al., 2016; Shou et al., 2017] and reported the state-of-the-art performance due to data-driven spatiotemporal feature representations, effective end-to-end learning strategies, and the availability of large-scale action/activity datasets [Soomro et al., 2012a; Jiang et al., 2014; Fabian Caba Heilbron and Niebles, 2015].

Most existing deep network based action localisation methods, however, adopt a strongly supervised learning strategy that relies on a large amount of annotated video data, which are costly to collect [Karpathy et al., 2014; Kay et al., 2017]. While transfer learning or model pretraining may mitigate this problem to some extent [Shou et al., 2017; Carreira and Zisserman, 2017; Xu et al., 2017], it remains challenging to handle novel action classes and adapt learned network models to a new scenario with high data efficiency. By contrast, human beings require little supervision in the process of learning new concepts, yet achieve very good generalisation. We are capable of learning new action classes and localising their instances from only a few examples of each class [Frey and Gerry, 2006; D., 2007]. Part of the reason for the fast and flexible learning ability is that we can integrate past experience into interpreting the new information. Therefore, it is much desirable to incorporate such an efficient learning strategy into action localisation for more flexibility and better generalisation when annotations are scarce and costly to obtain.

To this end, in this chapter, we consider a one(few)-shot learning scenario of action localisation; given one (or a few) example of new action classes, typically one per class, our goal is to detect all occurrences of each class in an untrimmed video. This problem falls into the regime of one-shot learning and presents the challenge of

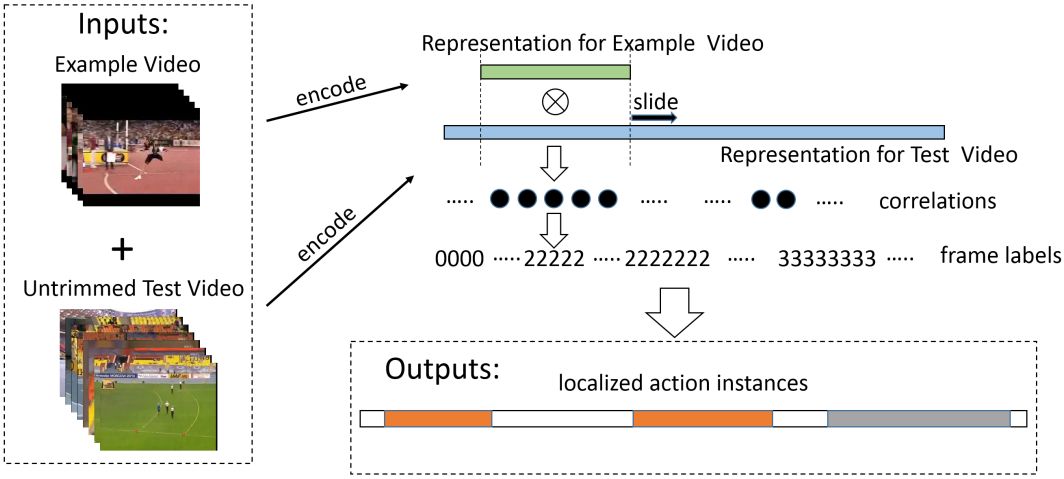


Figure 5.1: Example-based action localisation problem setup. Inputs are a few example videos and an untrimmed test video, after encoding and computing correlation, frame labeling are obtained by comparing correlations. Finally, frame labeling are combined into action instances.

how to rapidly adapt to new classes with minimum supervision.

While one-shot learning has been extensively studied in the context of visual recognition [Andrychowicz et al., 2016; Li and Malik, 2016; Ravi and Larochelle, 2017; Santoro et al., 2016b; Vinyals et al., 2016; Finn et al., 2017; Mishra et al., 2017b], few studies address the challenge of learning from a few instances to *detect* spatiotemporal patterns, such as actions, in videos. To tackle this problem, we develop a novel meta-learning strategy that integrates the task-level prior knowledge on matching video sequences into learning action localisation. The key ideas of our one-shot learning strategy are an intuitive, structured representation of action videos suitable for matching (partial) sequences and a similarity metric that allows us to transfer the labeling of action examples to action proposals in the untrimmed video. Figure 5.1 shows an example of our one-shot action localisation pipeline.

Specifically, we propose a new Matching Network architecture based on the meta learning framework. Our method takes as input a few examples per action class and predicts a dense temporal labeling of a test video in terms of action instances. In contrast to the classification network in [Vinyals et al., 2016], our localisation network first generates a sequence of action proposals in a sliding window manner and predicts their action labels through three network components, as follows. The first component, *a video encoder network*, computes a segment-based action representation for each action proposal and reference action, which maintains the temporal structure of actions and allows for accurate localisation. The action proposals are then compared with every reference action by the second network component, *a similarity network*, which generates a set of correlation scores at each time step. Based on the correlation scores within a time window, the third component, *a labeling network*,

predicts the action class label (as one of the foreground classes or the background) for the proposal in each time step.

Since all the network components are differentiable, our localisation network can be trained in an end-to-end fashion. We evaluate our approach on the Thumos14 and the ActivityNet dataset and report significantly better performance than the state-of-the-art methods while using only a small training set. Our ablative studies also demonstrate the efficacy of every model components in our architecture.

Our main contributions in this chapter are three-fold:

- We introduce an one(few)-shot action localisation problem that addresses the novel task of detecting action instances given only one (or a few) annotated video examples per class;
- We propose a meta-learning approach to the action localisation problem based on the Matching Network framework, which is capable of capturing task-level prior knowledge;
- We develop a structured representation of action videos for matching that encodes the temporal ordering information and produces more accurate localisation results.

In the remaining of this chapter, we first review the existing algorithms of one-shot(meta) learning and non-fully supervised video understanding techniques in Section 5.1. Then we present our one-shot action recognition network in Section 5.2, which lays the foundation for the one-shot action localisation task that we aim to address. After that, we show how to adapt the recognition network to localisation by introducing a meta-level localisation module. Finally, experimental results and conclusions are shown in Section 5.3 and 5.4 respectively.

5.1 Related Work

5.1.1 One-shot Learning and Meta Learning

Modern techniques for one-shot learning tend to utilise the meta-learning framework. Meta-learning techniques are applied on a set of datasets, with the goal of learning transferable knowledge among datasets.

Some approaches train a meta-learner that learns how to update the parameters of the learners' model, usually a deep neural network [Andrychowicz et al., 2016; Li and Malik, 2016; Ravi and Larochelle, 2017]. The meta-learner learns an update rule that will guide the learner to quickly adapt to each individual tasks. MAML [Finn et al., 2017] combines the meta-learner and the learner into one, by directly computing gradient with respect to the meta-learning objective. In [Santoro et al., 2016b], a memory-augmented neural network is trained to learn how to store and retrieve memories. Also tries to retrieve past memories, TCML [Mishra et al., 2017b] treats each individual dataset as a sequential input, and use temporal convolution to automatically discover learning strategies.

Metric learning methods are also employed by many one-shot learning algorithms that generate good results. Deep siamese networks [Koch et al., 2015] train a convolutional network to embed examples so that samples in the same class are close while samples in different classes are far away. [Vinyals et al., 2016; Shyam et al., 2017; Snell et al., 2017] refine this idea by introducing recurrence and attention mechanisms.

In this chapter, our method follow the framework of matching network [Vinyals et al., 2016], using correlation to classify given videos, and adapt it for the example based action detection problem.

5.1.2 Semi or Weakly Supervised Video Understanding

Fully supervised action localisation problem has been extensively studied [Lillo et al., 2016; Ni et al., 2016; Singh et al., 2016; Yeung et al., 2016; Shou et al.; Xu et al., 2017; Girshick, 2015; Yuan et al., 2017; Zhao et al., 2017; Shou et al., 2017; **?**; Gao et al., 2017; Ma et al., 2016; Xiong et al., 2017; Dai et al., 2017]. To capture temporal dynamics, LSTM networks have been widely used in representing action instances. [Singh et al., 2016; Yeung et al., 2016; Ma et al., 2016] use LSTM in addition to CNN to better model the temporal dynamics each proposal. In this work, we use LSTM to encode finer temporal details into the encoding vector of the video. 3D CNN and temporal convolution have proven to be effective in capturing spatial temporal features, and is reported to achieve state-of-the-art results for both action recognition and action localisation tasks. [Tran et al., 2015b; Shou et al.; Xu et al., 2017; Shou et al., 2017; Carreira and Zisserman, 2017; **?**]. However, 3D network is hard to train and require a large amount of data [Carreira and Zisserman, 2017]. Other methods use structured representations to model the details of action instances. Yuan *et al.* [Yuan et al., 2017; Zhao et al., 2017] improves localisation accuracy by explicitly finding the start, middle and end stage of an action. In this paper, we also employ a structured representation of the video, but we do not explicitly model different stages of an action.

Action localisation has been studied in other non-fully supervised setup by a few works. [Bojanowski et al., 2014; Huang et al., 2016] proposed some weakly supervised action labeling methods using only the action order information. UntrimmedNets [Wang et al., 2017b] employs a different type of weak supervision by using only untrimmed videos without annotations during training, and a attention based modelling method is proposed under such weak supervision. [Soomro and Shah, 2017] uses no supervision at all and formulate the unsupervised action detection problem as a Knapsack problem. They propose a supervoxel-based pipeline that first discover actions and then spatial-temporally localise them. Unlike these problem setups, we aim to address the example-based action detection problem, which can be formulated as one-shot action localisation. The one-shot aspect of the problem restrict us from using any pre-trained networks.

5.2 Methodology

An overview of our one-shot action localisation net is shown in Figure 5.2. The core component of it is the one-shot recognition network, which we will explain first. Then we explain how it can be easily adapted for action localisation. Finally, we introduce the loss functions and how to train the model.

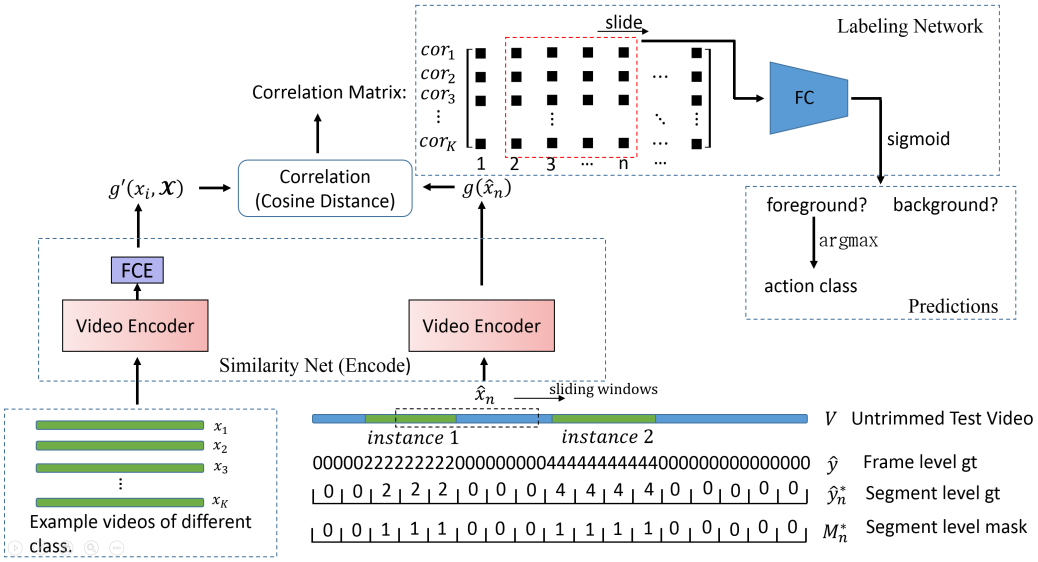


Figure 5.2: Overview of the one-shot localisation System. We use sliding window to swipe over the untrimmed test video with the stride equal to the segment length. For each window, we compute correlations with all reference examples. By concatenating correlations with every example at every time step, a correlation matrix is obtained. In the end, a FC network is applied on the correlation matrix over a certain time span to make segment level classification of fore/background. On the places of foreground, action classification is done through comparing correlations. We use different window sizes for multi-scale predictions.

5.2.1 Matching Network for Few-shot Action Recognition

We propose a meta-learning strategy to learn a structured representation of action instances and a matching similarity metric of actions that enable us to classify every action candidate in the untrimmed video based on the action examples and their class labels.

5.2.1.1 Video Encoder

Our matching-based action localisation relies on good alignment between the candidate and the reference videos. To achieve accurate alignment, we intend to maintain the temporal structure of action videos in our action representation. To this end, we

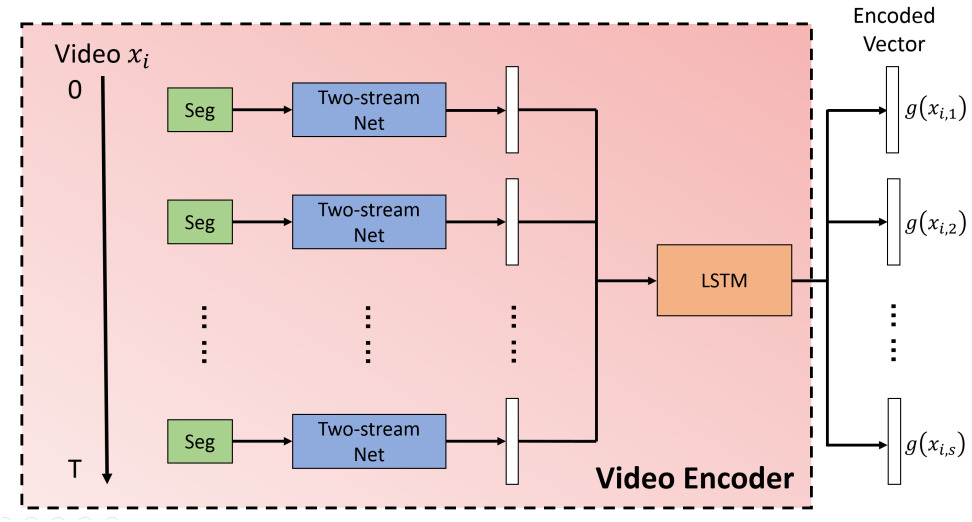


Figure 5.3: The video encoder. It uses temporal-segment two-stream network to encode the video into a structured representation. The final encoded vector for the video is the concatenation of the LSTM output for each segment.

develop a segment-based video representation with ranking LSTM to encode each action instance as a fixed-length sequence of video segment features.

More specifically, given a target video x_i, we evenly split the video into S segments. Each segment is first encoded by a two-stream temporal segment network [Feichtenhofer et al., 2016; Simonyan and Zisserman, 2014a; Wang et al., 2016], denoted as ϕ, that generates an initial encoding vector $\phi(x_{i,s})$ for the s-th segment where $s = 1, \cdots, S$. The temporal segment structure allows us to generate a finer-grained or partial alignment between two video sequences, e.g., a reference example and a query video. To capture the temporal context, we add an additional LSTM network layer that takes as input the sequence $\{\phi(x_{i,s})\}_{s=1}^{S}$, and produces the final encoding of the query video as

$$\{g(x_{i,s})\}_{s=1}^{S} = \text{LSTM}(\{\phi(x_{i,s})\}_{s=1}^{S}) \tag{5.1}$$

The structure of our video encoder is illustrated in Figure 5.3. The LSTM layer is trained with a ranking loss to enhance the temporal order structure, which we will explain in detail in Sec 5.2.3.3. The overall representation of the target video is formed by concatenating the encoding vectors of all the segments.

5.2.1.2 Similarity Network for Matching Actions

Given the video representation, we now turn to the task of classifying the query video clip into one of the action classes with example videos or the background. Inspired by the Matching Network architecture [Vinyals et al., 2016], we compute a correlation score between the action candidate and each action example using a

similarity network. The overview of the Similarity Network is shown in Figure 5.4.

Formally, we refer to the dataset of action examples and their labels as the support set and denote it as $\{\mathcal{X} = \{(x_i, y_i)\}_{i=1}^{c*k}$ where c is the number of classes and k is the number of examples per class. The similarity network first applies full context embedding [Vinyals et al., 2016] for each individual examples x_i with respect to the entire support set. Specifically[1], let $g(x_i)$ be the encoding vector of example x_i, we define the Full Context Embedding (FCE) $g'(x_i, \mathcal{X})$ as

$$g'(x_i, \mathcal{X}) = \overrightarrow{h}_i + \overleftarrow{h}_i + g(x_i) \tag{5.2}$$

$$\overrightarrow{h}_i, \overrightarrow{c}_i = \text{LSTM}(g(x_i), \overrightarrow{h}_{i-1}, \overrightarrow{c}_{i-1}) \tag{5.3}$$

$$\overleftarrow{h}_i, \overleftarrow{c}_i = \text{LSTM}(g(x_i), \overleftarrow{h}_{i+1}, \overleftarrow{c}_{i+1}) \tag{5.4}$$

where the FCE uses a bi-directional LSTM and treats the $g(x_i)$ as an input sequence. The FCE vector $g'(x_i, \mathcal{X})$ is the summation of outputs of ith step of the LSTM in both directions, and the original $g(x_i)$. The full context embedding encodes the dataset context and enriches the representations of action examples. Please note that the choice of bi-directional LSTM is necessary in FCE. Obviously, the order of different training examples x_i in a dataset should not convey any information because there are no temporal relations between each individual examples. Therefore a bi-directional LSTM is employed: by adding the outputs from both directions, the effect of the temporal order of x_i is mitigated.

Now given an target video $\hat{x}$ and its encoding vector $g(\hat{x})$, the similarity network computes the cosine distances between it and all the representations of the examples:

$$d(\hat{x}, x_i) = \frac{g(\hat{x})^T g'(x_i, \mathcal{X})}{|g(\hat{x})| \cdot |g'(x_i, \mathcal{X})|} \tag{5.5}$$

Based on the distances, the original matching network uses an attention mechanism and voting strategy to classify the test data into one of classes in the support set [Vinyals et al., 2016]:

$$\hat{y} = \sum_{i=1}^{c*k} a(\hat{x}, x_i) y_i \tag{5.6}$$

where $a(\hat{x}, x_i)$ is the softmax attention of test sample $\hat{x}$ to the examples x_i:

$$a(\hat{x}, x_i) = e^{d(g(\hat{x}), g'(x_i, \mathcal{X}))} / \sum_j e^{d(g(\hat{x}), g'(x_j, \mathcal{X}))} \tag{5.7}$$

Matching network provides a non-parametric nearest-neighbour like approach for one-shot learning. In this way, we can achieve one-shot action recognition.

However, we note that, as the support set is only composed of action classes (foreground), such classification method is not applicable to the localisation task, in which we also have to distinguish foreground from background. Therefore, in our

[1] For simplicity and readability, we drop the subscript s.

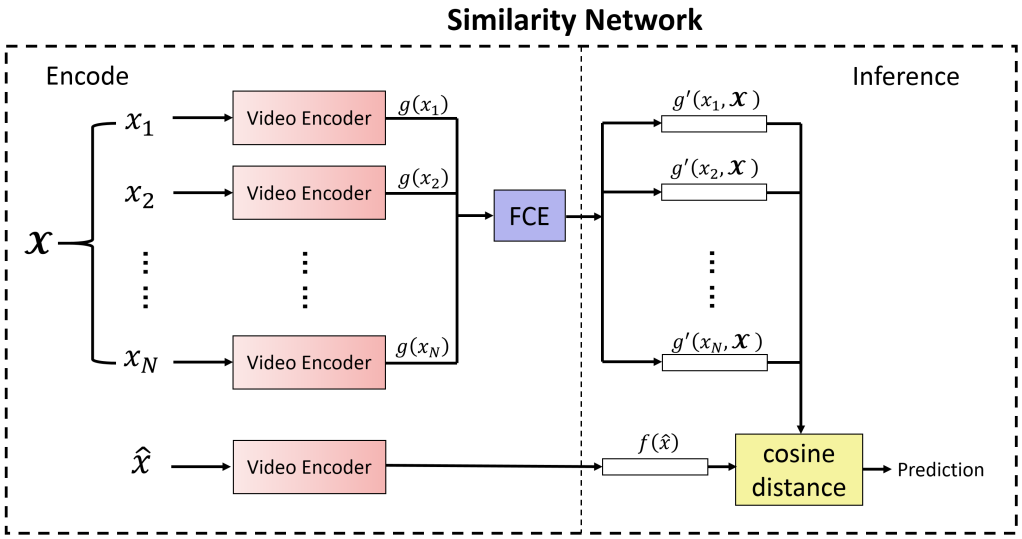


Figure 5.4: Similarity Network. All example videos along with a test video are the inputs for the similarity network. The similarity network applies FCE on each encoding vectors of example videos. The FCE is parameterised by a bi-directional LSTM.

one-shot action localisation architecture, we use the similarity network to compute the correlation scores, and design a separate labeling network to infer the class labels (including background) of each proposal. Details of the localisation structure and the labeling network are described in Sec 5.2.2.

5.2.2 Meta-network for Few-shot Sequence Labeling

Given a few typical examples from a set of new action classes, our goal is to locate all the instances of these classes in an untrimmed video. Specifically, we develop a labeling network on top of the recognition network describe above. When combined, the whole framework takes as input the examples of new action classes and predicts a dense labeling of the test video for localising action instances of these classes. The one-shot localisation framework adopts a dense sliding-window approach to generate a sequence of action proposals, and assign a class label to each of these proposals. Using our video encoder, a fixed length video representation consisting of multiple segments is obtained for each window. The same process is applied on the example video as well. At each time step, a correlation score between each pair of the proposal and the reference examples is computed by the Similarity Network. Then, the correlation scores at every time step are concatenated to form a correlation score matrix, which captures how the similarity between the example and the untrimmed test video changes through time. Finally, the labeling network predicts the classes of every action proposals based on the correlation scores in a local temporal window. The whole framework of one-shot action localisation that combines **Video Encoder**,

Similarity Network and **Labeling Network** is illustrated in Figure 5.2

Specifically, Our action localisation network uses the sliding-window strategy to generate action proposal with a fixed stride step. In this work, we set the stride size equal to the segment length used in the video encoder. For each window, we obtain an encoding vector and compute correlation scores with the example videos as discussed above. Formally, we denote the length of each window as l, the number of segments in each window as S, and the length of the test video (number of frames) as N. Then the segment length, also the stride size, is $\frac{l}{S}$ and the number of window is NS/l. Let $\hat{x}_n$ denote nth window proposal, $\hat{x}_n = \{f_{(n-1)l/S} \cdots f_{(n-1)l/S+l}\}$, where $n \in [1, \frac{NS}{l}]$. For each window we have:

$$cor_{i,\hat{x}_n} = g'(x_i, \mathcal{X})^T g(\hat{x}_n) \tag{5.8}$$

By concatenating $cor_{i,\hat{x}_n}$ for each i and n, we obtain a correlation matrix. Window length l indicate the scale under which we are searching. With different l, we can get correlation matrix for multiple scales, thus can obtain multi-scale predictions.

The labeling network is applied directly on the correlation matrix. Similar to Eq 5.6 where classification is done through comparing cosine distances, we compare different rows of the same column of the correlation matrix by applying a fully connected neural network on the correlation matrix over a short temporal span, and output a probability distribution over foreground and background via `sigmoid` activation. The fully connected network slides over the correlation matrix along the time dimension, determining fore/background for every proposal. On the proposals which the labeling network determines to be foreground, Eq 5.6 is applied to predict an action label, which is shown in the upper part of Figure 5.2.

Our labeling network is applied over a certain time span, in order to capture the contextual information and to reduce frequent label switch between fore/background. Note that the correlation matrix contains both information about specific action classes, and whether a proposal belongs to the background or foreground. For example, if the correlation with one example are much higher than that with the others, then the proposal most likely belongs to foreground and has the same action label as the example; If the correlations with all examples are relatively low, then it probably belong to the background. The labeling network learns these criteria through training.

Also note that the labeling network is, in a sense, independent to the action classes, as it is applied on the correlation matrix rather than the feature representation of each videos. This means that criteria it learned should be class-agnostic and applicable to input videos from any classes. This property makes it suitable for one-shot prediction. One can think of the correlation as an operation that filters out irrelevant video-specific information, and left behind only relevant similarity information. The task of the labeling network is only to determine whether there are any outstanding correlations at a specific location. Therefore, we can directly apply the trained network to other previously unseen classes and still expect a good performance. One can also understand this from another perspective: the encoder learned

through find the nearest sample will capture meta-level (class-agnostic) knowledge, so the output vectors (correlation matrix) are in meta-level as well. Therefore, any network applied only on top of the correlation matrix should also learn meta-level knowledge as well.

Post-processing In the post processing stage, we combine multi-scale proposal-level predictions to obtain a single frame-level predictions for the test video, and group adjacent frames of the same label to obtain action instances. Specifically, we employ three steps of post-processing.

1. We map the proposal prediction to its central segment and each frame in the segment share the same probability distribution as the segment;

2. We combine multi-scale prediction into one: Each frame has a probability distribution for each scale. We add the probabilities from every scales and choose the highest one and the corresponding class as the confidence score and the final prediction for the frame.

3. We group adjacent frames of the same label to get predicted action windows. The confidence score for the predicted window is the average score of every frames in that window. This post-processing step essentially achieves the Non-maximum Suppression (NMS) for the localisation.

5.2.3 One-shot Model Learning

Our localisation system is designed for one-shot action localisation, the corresponding meta-learning formulation should be employed in the training of the model. Every component of the system is differentiable and the system can be trained end-to-end. However, to get better initialisation and performance, we pre-train the video encoder and the similarity network with action recognition objective. In the remainder of this section, we first present the meta-learning formulation employed in this work, than the overall loss function for the localisation system. Finally, we present the the model pre-training using slightly different loss functions.

5.2.3.1 Meta Learning Formulation

In meta-learning, the model is trained in a meta-phase on a set of training tasks, and is evaluated on another set of testing tasks [Vinyals et al., 2016; Santoro et al., 2016b; Finn et al., 2017]. Formally, we define $\mathcal{T}_{meta-train}$ to be the collection of the meta-training tasks: $\mathcal{T}_{meta-train} = \{\mathcal{X}, \hat{\mathcal{X}}, \mathcal{L}(\mathcal{X}, \hat{\mathcal{X}}, \theta)\}$. Each task consists of its own training set $\mathcal{X} = \{x_i, y_i\}$, test set $\hat{\mathcal{X}} = \{\hat{x}_j, \hat{y}_j\}$ where y is the ground truth label for classification task, or a real value for regression task, and some loss function $\mathcal{L}$ to measure the performance of the meta-learner depending on the system parameters θ. Similarly, we can define the meta-testing tasks as $\mathcal{T}_{meta-test} = \{\mathcal{X}, \hat{\mathcal{X}}, \mathcal{L}(\mathcal{X}, \hat{\mathcal{X}}, \theta)\}$.

The goal of meta learning is to find the model that minimise the meta-test loss across a distribution over the meta-test tasks:

$$\theta^* = \text{argmin}_\theta \mathbb{E}_{\mathcal{T} \sim \mathcal{T}_{meta-test}}[\mathcal{L}(\mathcal{T}, \theta)] \tag{5.9}$$

During training, the test loss on $\hat{\mathcal{X}}$ of sampled tasks $\mathcal{T} \sim \mathcal{T}_{meta-train}$ is served as the training loss to the meta-learner. For each training set $\mathcal{X}$ in $\mathcal{T}_{meta_train}$, if only one or a few samples are presented, then the problem is known as one(few)-shot learning.

5.2.3.2 Optimisation for the Localisation System

The FC network in Figure 5.2 outputs a score $F_\theta(n)$ for each segment indicating whether it belongs to foreground or not. Using the segment level mask M_n^* as ground truth for fore/background, we can compute the localisation loss function as the binary cross entropy:

$$\begin{aligned} L_{loc} = -\frac{l}{NS} \sum_{n=1}^{NS/l} [M_n^* \log F_\theta(n) \\ + (1 - M_n^* \log(1 - F_\theta(n)))] \end{aligned} \tag{5.10}$$

We also compute classification loss at every segments whose ground truth is foreground:

$$L_{cls} = -\frac{l}{NS} \sum_{n=1}^{NS/l} [M_n^* \log P_\theta(\hat{y}_n | \hat{x}_n, \mathcal{X})] \tag{5.11}$$

where $\hat{y}$ is given by Eq 5.6. Note that during testing, we only further classify the segments who are predicted to be foreground into action classes. However during training, we compute classification loss on every ground truth foreground segments.

Using the meta-learning framework, we can have the final training loss for the one-shot localisation task as:

$$L = \mathbb{E}_{\mathcal{T} \sim \mathcal{T}_{meta-train}}[L_{loc} + L_{cls}] \tag{5.12}$$

5.2.3.3 Pretraining for Video Encoder & Similarity Net

The video encoder and the similarity network contain most of the trainable parameters in the system, including the two-stream temporal segment network, the LSTM layer and the FCE module. They serve as the backbone of the localisation system. Therefore, the overall performance can be improved with proper model pretraining.

We pretrain the video encoder and the similarity network under the action recognition task to get better initialisation. During the pretraining stage, only trimmed action instances are used, thus we are only concerned with the classification loss:

$$L_{cls,s} = \log P_\theta(\hat{y}_s | \hat{x}_s, \mathcal{X}) \tag{5.13}$$

where $\hat{y}$ is again given by Eq 5.6, $\hat{x}$ is the test action instance and $\mathcal{X}$ is the collection

of all example videos. With the model pretraining, we can also take advantage of the LSTM in the video encoder by employing ranking loss [Ma et al., 2016] to incorporate better temporal structures into the encoding vectors $g(x_i)$ given in Eq 5.1. Formally, the ranking loss is:

$$L_{rank,s} = \max(0,\ p_s^{\hat{y}} - p_s^{*\hat{y}}) \tag{5.14}$$

$$p_s^{\hat{y}} = (\hat{y}'_s)^T \cdot \hat{y} \tag{5.15}$$

$$p_s^{*\hat{y}} = \max_{s' \in [1, s-1]} p_{s'}^{\hat{y}} \tag{5.16}$$

where $\hat{y}'_s$ is the predicted probability distribution and $\hat{y}$ is the ground truth label. In other words, $p_s^{\hat{y}}$ and $p_s^{*\hat{y}}$ are the predicted probability with respect to the ground truth label $\hat{y}$ at segment s, and the past maximum predicted probability respectively.

The intuition of ranking loss is to encourage the classifier to make more confident prediction as it is presented with more and more of the video content. The loss incurs a penalty when the prediction confidence drops at the sth segment compared to the highest confidence among past segments. This encourages the network to make monotonic increasing predictions, which in turn, encourages the encoding vectors of different segments $\{g(x_{i,s})\}_{s=1}^{S}$ to be more discriminative to each other. The ranking loss can improve the temporal structure of the final encoding vector, i.e. the concatenation of $\{g(x_{i,s})\}_{s=1}^{S}$.

Under the meta-learning framework, we can write the final training loss for the pretraining of the video encoder and the similarity net as:

$$L = \mathbb{E}_{\mathcal{T} \sim \mathcal{T}_{meta-train}} \{ \sum_{s} [L_{cls,s} + L_{rank,s}] \} \tag{5.17}$$

5.3 Experiments

5.3.1 Datasets & Preparation

We use the Thumos14 [Jiang et al., 2014] and ActivityNet 1.2 [Fabian Caba Heilbron and Niebles, 2015] datasets to evaluate our one-shot action localisation algorithm. Thumos14 dataset has 1010 video of 20 classes in its validation set for action detection, and uses UCF-101 [Soomro et al., 2012a] as its trimmed training data. We do not use the Background Set during training. ActivityNet has 4819 untrimmed training videos containing 100 classes. There are 1.54 action instances in each video on average and no trimmed videos as additional data, so we cropped the action instances in the untrimmed video as the training data for the video encoder.

The one-shot problem setup requires that the classes during testing must not be present during training. Thumos14 contains 20 classes from UCF-101, so only the other 81 classes in UCF-101 are used during training the encoder. We denote the two splits of UCF-101 as UCF-101-81 and UCF-101-20. After training the encoder, we use a small part of Thumos14 validation set (contain 6 classes) to train the fully connected network with the features extracted by the video encoder, and use the

remaining 14 classes in the test set to test our one-shot localisation network. The two splits of Thumos14 validation set and test set are denoted as Thumos-val-6 and Thumos-test-14.

Similarly for activity net, we split the 100 classes into 80-20 splits. Our one-shot action localisation network is trained on videos containing only the 80 classes in the training set, denoted by ActivityNet-train-80, and is tested on the other 20 classes in the validation set, denoted by ActivityNet-val-20.

Model pre-training We split the UCF-101-81 dataset into 10000 small datasets each containing 5 example videos of 5 different classes and 1 test video belonging to one of the 5 classes. For each small dataset, the 5 action classes, 5 example videos and 1 test video are all randomly chosen. The 5 action classes are randomly assigned labels from 0-4. Same operation is applied for ActivityNet. We train the one-shot video encoder and the similarity net under the meta-learning setup on all such small datasets.

Training for localisation We split the Thumos-val-6 dataset and UCF-101-20 into small datasets in a similar manner. We randomly choose one video from Thumos-val-6, and pair it with 5 examples UCF-101-20 to form a small dataset. The examples are chosen from the mutual classes of UCF-101-20 and Thumos14-6. All such datasets are used for training the one-shot action localisation network. The 5 examples are randomly assigned labels from 1-5, with 0 represent background. Same operation is applied for ActivityNet. At this stage, the parameters of the video encoder are fixed, only the fully connected network are updated.

Testing for localisation During the meta-testing stage, the setup is identical to that of meta-training stage, only now we use Thumos-test-14. We pair a randomly chosen video in Thumos-test-14 with 5 examples from the mutual classes with UCF-101-20. As can be easily seen, there are a large number of different combinations of the 5 examples (random classes, and random sample within each class), and the localisation performance is dependent on the example videos chosen. To get a reliable test results, we randomly sample 1000 different datasets from each of the 14 test classes and calculate mAP across all these datasets. Same evaluation setup is use for ActivityNet.

Training details During training of the encoder as well as the FC network, the initial learning rate is set to be 0.001 and decrease by a factor of 5 once the training loss stop decreasing. To prevent including too many backgrounds during training, we only use the cropped clips of the untrimmed videos where the ratios of background and foreground frames are between 0.5:1 and 1.5:1 . Also, to ensure all such clips have the same length, we repeat multiple instances of shorter clips to make its length the same as longer ones. The segment ground truth and segment mask are repeated accordingly.

	mAP		mAP
Heilbron *et al.* [Caba Heilbron et al., 2016]	13.5	Ours@1	13.6
Yeung *et al.* [Yeung et al., 2016]	17.1	Ours@5	14.0
Yuan *et al.* [Yuan et al., 2017]	17.8	Ours@15	14.7
S-CNN [Shou et al.]	19.0	CDC@1	6.4
S-CNN + SST [Buch et al., 2017]	23.0	CDC@5	6.5
CDC [Shou et al., 2017]	23.3	CDC@15	6.8

Table 5.1: Comparison of our one-shot localisation method with state-of-the-art methods on Thumos14. The left half are the results under full supervision, the right half are results for one-shot setup. @n means n examples per class are used during training. The IoU threshold is set to 0.5 for all comparisons.

	mAP@0.5	Average mAP
TCN [Dai et al., 2017]	37.4	23.5
R-C3D [Xu et al., 2017]	-	26.8
Wang *et al.* [Lin et al., 2016]	42.2	14.8
Lin *et al.* [Lin et al., 2017]	48.9	32.2
Xiong *et al.* [Xiong et al., 2017]	41.1	24.8
CDC [Shou et al., 2017]	43.8	22.7
Ours@1	22.3	9.8
Ours@5	23.1	10.0
CDC@1	8.2	2.4
CDC@5	8.6	2.5

Table 5.2: Compare our one-shot localisation method with state-of-the-art methods on ActivityNet. The Upper half are the results under full supervision, the lower half are results for one-shot setup. @n means n examples per class are used during training.

5.3.2 Evaluation

To demonstrate the effectiveness of our approach, we make a comparison with one of the state-of-the-art action detection method under the one-shot setup, showing that a method which works well with a lot of training data will perform poorly with very little data. In addition, we include the results of other state-of-the-art methods under full supervision to put our method in context.

We see from Table 5.1 and 5.2 that, although there is still a performance gap between one-shot and fully supervised action detection, our method significantly outperforms the state-of-the-art method when tested in one-shot setup[2]. Fully supervised method like CDC requires a large amount of data to obtain meaningful

[2]In the comparison of one-shot performance, we train the CDC network from scratch, as opposed to using the pre-trained model on Sports-1M in the paper, because the one-shot setup forbids us of using any pre-training.

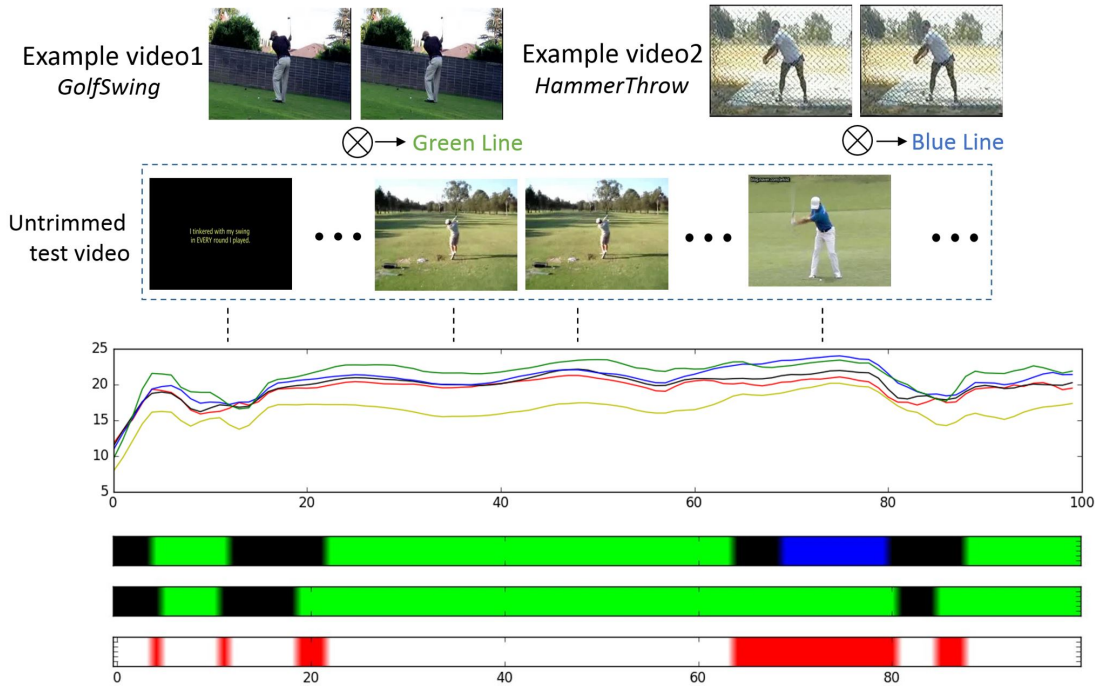


Figure 5.5: A portion of the correlation matrix with its corresponding video frames and predictions. Test video contains multiple instances of *GolfSwing*, one of which is similar to the *HammerThrow* example video. **Green** is the ground truth action, **black** is background and **blue** is another action class. On the last row, **red** is false predictions. A relationship between visual similarity and correlation scores can be observed.

representations where are ours can benefit from a few.

Under the meta-learning setup, the labels are randomly changed for each action class for every iteration during training, which forces the feature representation of our video encoder to decouple from the labels. This way the video encoder can treat new action classes just as any other classes, thus achieving fast adaptation from only one sample. The class-independent property of the parameters learned from meta-learning are particularly desirable since the matching network employs a non-parametric approach for classification. However, for existing methods, when a sample changes its ground truth label from one iteration to another, it can confuse the network, preventing the model from learning discriminative features effectively. Also, from only one or a few samples, the network suffers from heavy over-fitting.

Figure 5.5 visualise the correlation matrix and the classification criteria learned by the FC network. We can see that the FC network will output background when all correlations are relatively low, or when there is no outstanding correlation. Correct predictions can be made through comparing correlations most of the time, which indicates that the feature representation of our one-shot video encoder is discriminative. However, false predictions do appear around the action boundaries. Also,

different time steps of the same action may express resemblance with different examples (blue part in Fig 5.5), which will lead to false predictions.

5.3.3 Ablation Study

5.3.3.1 Effect of the Similarity Net

We conduct an ablative study and evaluate the one-shot performance where the similarity network is replaced by binary SVM classifiers. To show more context, we also show the performance achieved using Dynamic image [Bilen et al., 2016a]. The result is shown in Table 5.3, which demonstrates the vital role of the similarity network in learning effective representations for the one-shot localisation.

	mAP
Dynamic image [Bilen et al., 2016a] + Binary classifier	4.4
Our encoder + Binary classifier	6.0
Our encoder + Similarity net	13.6

Table 5.3: Comparison between the similarity network and a binary linear SVM classifier with the same video encoder.

The reason for the the significant drop with a simple SVM classifier is that it is not trained to capture similarity, but rather extracting class-dependent features, whereas the similarity network is class independent. The cosine distance layer in the similarity network (see Figure 5.4) masks all parameters in the network from the actual classes, thus giving the network the ability to adapt to new classes very quickly.

5.3.3.2 Effect of Temporal Alignment

In this work, a finer level of alignment is obtained through the segment-based video representation of the encoder, as well as the ranking loss LSTM. Here we show that such structured representation has the benefit of improving localisation performance. Below is the comparison one-shot recognition and localisation performance under different alignment schemes, shown in Table 5.4 and 5.5 respectively. This demonstrates the critical role of our alignment for the localisation task and its effectiveness in learning discriminative feature representation, respectively. Also, the recognition results show that our alignment does not sacrifice the feature representation capacity. The results for the first row in the table is obtained by doing classification separately on each segments, and then combine the prediction of every segments. Such late fusion technique are known to boost classification accuracy [Simonyan and Zisserman, 2014a; Wang et al., 2016]. However, on the second and third row, the results shown are the accuracy using the concatenated vector of all segments. In this comparison, the accuracy with and without ranking loss are practically identical, proving that feature alignment property hinders very little the expressive power of the representation.

Alignment Scheme	accuracy (%)
TS + prediction pooling	58.0
TS, w/o ranking loss	57.5
TS, with ranking loss	57.4

Table 5.4: One-shot recognition accuracy under different alignment scheme. *TS* refers to the temporal-segment two-stream network depicted in Figure 5.3.

Alignment Scheme	mAP	accuracy
No alignment	7.8	-
TS, w/o ranking loss	12.9	57.5
TS, with ranking loss	13.6	57.4

Table 5.5: One-shot detection mAP and one-shot recognition accuracy under different alignment scheme. In the table, *No alignment* means not employing segment-based representation, nor ranking LSTM. *TS* refers to the temporal-segment two-stream network depicted in Figure 5.3.

We can also see from the Table 5.5, a good alignment is critical in our algorithm. With segment-based representation, the finer temporal structure allows for more accurate localisation. Although most of the alignment property is gained through the temporal segment structure of the encoder, ranking loss also provides a noticeable performance boost, as it encourages the features from different segment to evolve over time, discriminating features from different segments, thus improved the structured representation. In the third column, the accuracy with and without ranking loss are practically identical, indicating that with feature alignment we can also learn an expressive representation.

5.3.3.3 Validity of the Video Encoder

As explained in Section 5.2.1.1, the video encoder employs the temporal-segment two-stream structure. Specifically, the architecture of the CNN is what we call mini-VGG, which follows the architecture of the VGG-16 network, but with only a quarter of the convolutional kernels, resulting in a significantly smaller network size. The main reason of employing such architecture is to reduce the memory requirements without causing too much performance degradation. Table 5.6 shows the encoder performance of the fully supervised action recognition task on the UCF-101 dataset as well as the number of parameters. Although our encoder is not the best performing one, it is much more efficient with only slight performance drop. More importantly, the segment-based encoder is critical in our localisation system, as shown before. Our encoder is well matched with dynamic image and two-stream network, with only about one tenth of the parameters. compared to two-stream fusion network, we achieve a reduction of parameters of 27 times with only 4% performance loss.

	Accuracy(%)	# parameters
two-stream network[Simonyan and Zisserman, 2014a]	82.9	∼27.5M
dynamic image [Bilen et al., 2016b]	76.9	∼36.1M
two-stream fusion [Feichtenhofer et al., 2016]	85.2	∼97.3M
temporal-segment network (VGG16)[Wang et al., 2016]	85.7	
temporal-segment LSTM (ResNet)[Ma et al., 2017a]	86.2	
my encoder (mini-VGG16)	81.3	∼3.6M

Table 5.6: Video encoder performance comparison for fully supervised action recognition. Our encoder performs reasonably well with only a fraction of the network size compared to other widely used encoders.

More importantly, the segment-based encoder is critical in our localisation system, as shown before.

5.3.4 Scalability of Method

We further test our method using slightly altered setup:

1. Using more samples per class.
2. Using larger number of classes.

	mAP@1	mAP@5	mAP@15
Ours	13.6	14.0	14.7
CDC	6.4	6.5	6.8

Table 5.7: Comparison of our few-shot localisation and CDC method on Thumos14 for different number of samples per class. @n means n examples per class are used during training. The IoU threshold is set to 0.5 for all comparisons.

mAP	5-way	20-way
Ours	13.6	13.5
CDC	6.4	5.5

Table 5.8: Additional results on 20-way one-shot action localisation performance.

Table 5.7 shows the results comparison of using 1, 5 and 15 samples per class between ours and CDC. Noticeable performance jumps can be seen for our methods as the number of sample increases while only small improvements are observed for CDC, which indicates a good scalability of our approach w.r.t. the number of samples. On the other hand, Table 5.8 shows localisation results with more number of classes in each datasets. We notice that the performance of our method is nearly

unaffected by the increase of number of classes while CDC witness a slight drop. We believe this is due to the fact that our network learns class-agnostic features through similarity learning and thus is robust against number of classes.

5.4 Conclusion

In this chapter, we present a one-shot action localisation system that employs meta learning concept and adopts a similarity based classification strategy. We develop the correlation matrix representation and train a labeling network on the meta-level to characterise how the correlations evolve over time. This enable us to distinguish between example classes and to detect background as well, which is crucial to the localisation task but lacking in previous one-shot learning methods. Our network is trained under the meta-learning framework so that it can quickly adapt to new classes with just one(few) training samples. We also develop a structured representation of action videos for matching that encodes the temporal ordering information and can benefit to more accurate localisations.

Chapter 6

Frame Synthesis by Feature Disentanglement

How to automatically, without any supervision, learn and decompose the characteristics of an object in the world is an ultimate goal in computer vision. Conversely, learning to synthesise components from different objects and generate novel images is also very challenging. This problem has attracted an extraordinary amount of interests recently due to rapid progress in deep generative models [Kingma and Welling, 2013; Oord et al., 2016; Goodfellow et al., 2014].

While a variety of generative frameworks have been proposed, the generative adversarial network (GAN) [Goodfellow et al., 2014] and its variants [Arjovsky et al., 2017; Radford et al., 2015; Gulrajani et al., 2017] have arguably become most prevalent due to high-quality images they can produce. Through different learning strategies on such models [Mirza and Osindero, 2014; Isola et al., 2017; Ma et al., 2017b; Zhang et al., 2017], the well-trained latent representation of images are able to synthesise novel images with desirable properties according to the external inputs on latent space. Unfortunately, these methods need laborious labeling, pairing or other supervision signals to enforce the network to memorise the mapping. More importantly, their performance are far from satisfactory in this task. At the same time, unsupervised disentangling methods [Chen et al., 2016; Higgins et al., 2017; Hu et al., 2018] currently are only able to be applied in very naive datasets. It is impractical to apply unsupervised learning to disentangle all the characteristics.

In this Chapter, in order to generate realistic and novel object images by leveraging as little supervision as possible, we narrow down components to the most prominent two properties of an object: *appearance* and *shape*. Learning a latent representation that properly disentangles those two factors from images, however, still remains a challenging task. Previous attempts [Ma et al., 2017b; Isola et al., 2017; Villegas et al., 2017b; Fan et al., 2018; Siarohin et al., 2018; Dong et al., 2018] to this task have always resorted to paired data (e.g., two images with the same appearance but different poses) in order to obtain ground-truth supervision on what generated images should look like. While in limited scenarios where the paired information is easy to acquire [Villegas et al., 2017b], obtaining paired data is often expensive, time-consuming and may require extra complex operations [Chan et al., 2018].

Weakly-supervised Disentanglement A more general learning strategy is to use unpaired data, which has been successfully applied to many image-to-image translation tasks [Zhu et al., 2017a; Choi et al., 2017; Liu et al., 2017b; Bousmalis et al., 2017; Liu and Tuzel, 2016; Taigman et al., 2016; Kim et al., 2017; Pumarola et al., 2018a; Yi et al., 2017; Lee et al., 2018]. However, these methods have limited capacity in learning disentangled representations in appearance and pose space. For instance, CycleGAN [Zhu et al., 2017a] and GANimation [Pumarola et al., 2018a] both assume rough alignment of the input and generated images and thus are unable to independently capture the spatial transforms required by controlling the pose factor. Recent works [Ma et al., 2018b; Esser et al., 2018] strive to learn such a disentangled representation by decomposing the task into pose and appearance representation learning. Nevertheless, they require both original image and the corresponding pose information as input to decouple the pose and appearance information. As such pose information is critical to defining their body ROI regions or appearance patches, their networks do not learn to extract appearance information independently, which are rather restrictive. We consider such unpaired disentangle frameworks requiring external attribute-specific cue such as joint coordinates as weakly supervised disentanglement.

Unsupervised Disentanglement Although a few attempts [Esser et al., 2018; Ma et al., 2018b; Raj et al., 2018; Han et al., 2018; Wang et al., 2018; Pumarola et al., 2018b] have managed to address this problem using only unpaired data by leveraging the recent success of unpaired image to image translation algorithms, they always require explicit attribute-specific cue as additional inputs, such as human skeleton [Esser et al., 2018; Pumarola et al., 2018b], joint heatmaps [Ma et al., 2018b], body segmentation [Raj et al., 2018; Mo et al., 2018], cloth patches [Han et al., 2018; Wang et al., 2018] or a combination of several cues [Dong et al., 2018], in order to individually control the two properties of the generated images and looking realistic at the same time. While these explicit attribute-specific cues can provide valuable supervision information, they are fixed and manually defined, thus exposing two limitations:

1. Complex pre-processing and heavy engineering tricks are often employed in order to utilise the attribute-specific cues. Examples include: convert joint coordinates to heatmaps using 2D Gaussian noise [Villegas et al., 2017b]; patching white circles of specific radius around each joints locations [Ma et al., 2017b, 2018b]; carefully drawing stickman [Esser et al., 2018; Pumarola et al., 2018b]; and manually crop or warp appearances patches around every body parts [Ma et al., 2018b; Esser et al., 2018]. Not only do those pre-processing severely limit the flexibility of their methods, they also impose strong prior feature representations rather than learning from data. In other words, using explicit attribute-specific cues limits the network's ability of leaning plausible disentangled representations in shape and appearance space.

2. The predefined attribute-specific cues inevitably carry undesired bias into the

system, which will hinder the quality of the generated images. For example, artificially defined human poses such as skeleton or joint coordinates cannot comprehensively describe the shape of a person. Such shape information lack important details about e.g., thickness of limbs, hair styles etc, resulting in stiff and unnatural looking generated persons [Esser et al., 2018]. Another example is that body segmentation obtained with human parsing [Gong et al., 2017, 2018] often focus on types of apparels rather than the underlying human structure, restricting the use case to only a few specific types of apparel [Mo et al., 2018]. Therefore, usage of the attribute-specific cues absent of further refinements may result in ill-suited representations for image translation task.

Therefore, we argue that the most general learning strategy is to extract more plausible data-driven disentangled representations for appearance and shape respectively through unsupervised learning. We consider such unpaired disentangle frameworks not requiring any external attribute-specific information as unsupervised disentanglement.

In the remaining of this chapter, we first review the literature related to image-to-image translation and disentanglement. Then we present our proposed weakly-supervised disentanglement framework in Section 6.2.1 and the purely unsupervised framework that do not require any external attribute-specific cues in Section 6.2.2. After that, we analyse the experimental results and demonstrate the capabilities of the two frameworks. Finally, we draw conclusions.

6.1 Related Work

6.1.1 Image to Image Translation

Promising results have been accomplished under supervised setting when input-output images are paired exactly. The most successful supervised method is the Pix2pix framework [Isola et al., 2017; Wang et al., 2017c], which applies a conditional discriminator [Mirza and Osindero, 2014] on image patches to generate high quality images. However, the problem becomes more challenging with no available paired data.

In the context of unpaied image to image translation, many works have also been proposed. CycleGAN [Zhu et al., 2017a] first introduced cycle consistent in this context, where the mapped image must be able to translate back to the same input image via a reverse mapping. Building on top of this idea, DiscoGAN [Kim et al., 2017] and UNIT [Liu et al., 2017b] propose to discover and utilise cross domain relations for better generation, assuming there exists a shared cross-domain latent space. Pumarola *et al.* [Pumarola et al., 2018a] and Perarnau *et al.* [Perarnau et al., 2016] also enforce cycle consistency on latent space for controlled image generation, such that the conditional vector can be extracted from the translated image [Chen et al., 2016]. Moving beyond just two domains, [Zhu et al., 2017b; Choi et al., 2017; Lee et al., 2018] learn to translate images across multiple domains. Using exemplars and

feature masks, Ma *et al.* [Ma et al., 2018a] is able to do many-to-many mapping in a controlled manner. This is similar in essence the patch consistency introduced in our method. In this work, we adapted cycle consistency with disentangled representation to achieve controlled human and face generation.

6.1.2 Conditional Transfer for Appearance and Shape

As a branch of Image-to-Image, conditional transfer for appearance and shape is a more specific but also more challenging problem, because the target object often possess large spatial variations.

A major limitation in previous image translation methods, whether they are paired or unpaired, [Zhu et al., 2017a; Choi et al., 2017; Liu et al., 2017b; Bousmalis et al., 2017; Liu and Tuzel, 2016; Taigman et al., 2016; Kim et al., 2017; Pumarola et al., 2018a; Yi et al., 2017; Lee et al., 2018] is that they can not manipulate attributes independently to generate novel images, severely restricting their use case to image with the same spatial structures. When spatial structures differs dramatically between inputs and outputs, as is the case in human image translation, they often produce unpleasant results cluttered with visual artefacts (See Figure 6.14 and 6.15). As a consequence, we can not directly apply similar methods on the task of conditional appearance and shape transfer. Instead, we resort to learning disentangled representations to manipulate image attributes and achieve realistic generation results.

A fair amount of works have been proposed to tackle unpaired appearance and shape transfer [Esser et al., 2018; Ma et al., 2018b; Raj et al., 2018; Han et al., 2018; Wang et al., 2018; Pumarola et al., 2018b], some of which also strive to extract disentangled representations. However, they all rely on some type of additional information such as joint coordinates [Esser et al., 2018; Pumarola et al., 2018b; Ma et al., 2018b] or stand-alone cloth patches [Han et al., 2018; Wang et al., 2018]. We argue they only managed to address the task in a weakly-supervised manner. Mo *et al.* [Mo et al., 2018] took a different approach by exploring domain-specific information, and propose instance-aware transfer using human parsing masks. However, their model can only be applied on specific types of clothes and needs to be retrained for other apparels, making it unpractical.

6.1.3 Disentangled Representation

Learning disentangled representation has been a pursuit of machine learning researchers. With the recent developments of generative models, many methods are able to discover factors of variation other than those relevant for classification. Mathieu *et al.* [Mathieu et al., 2016] utilises a set of labeled observations to discover other factors that are independent to the label through cross-domain image translation and reconstruction. [Yim et al., 2015; Peng et al., 2017; Liu et al., 2018] use images of different styles to learn a disentangled factor that control the style. Taking advantage of the temporal coherence and the rich pairwise information in videos, [Denton et al., 2017; Villegas et al., 2017a; Baddar et al., 2017; Tulyakov et al., 2017] attempt to factor

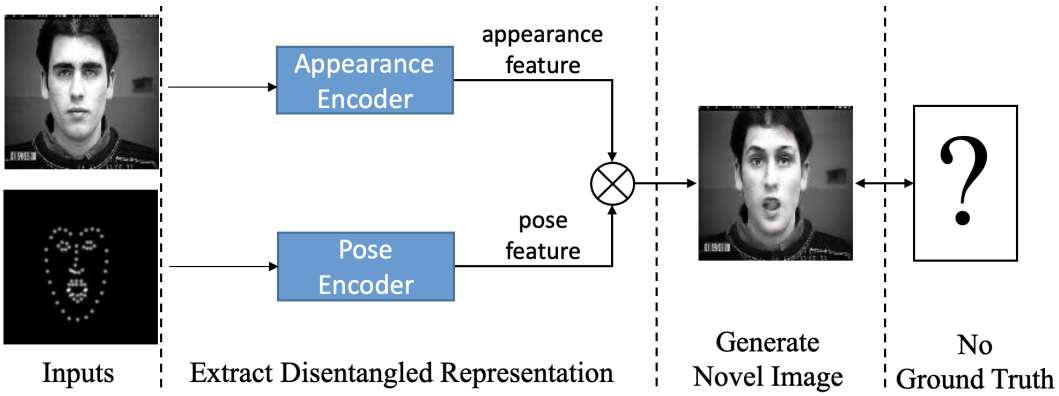


Figure 6.1: The problem setup of the weakly-supervised approach. We aim to learn disentangled appearance and pose representations to be able to control each aspect of the generated image. The inputs are a human image that provides appearance information; and a joint keypoints heatmap that represents pose. The outputs are novel yet realistic images with the desired attributes. There are no ground truth images to guide the generation process.

the video into temporal stationary and temporal varying components, or content and motion. The above works all try to acquire other factors of variation with the help of a given factor, such as the styles of the image or the poses of the frames. It is more challenging to acquire disentangled representations in an unsupervised manner. The method by [Hu et al., 2017] manages to automatically discover different factors by mixing and unfolding latent representations. We absorb inspirations from [Hu et al., 2017] but focus on mixing the appearance and pose features in our paper.

6.2 Methodology

6.2.1 UNet with Cycle and Patch Consistency

In this work, our goal is to generate realistic images based on the input of pose and appearance images in a weakly supervised fashion: we do not necessarily require ground-truth images to guide our training but we need an explicit pose cue input in order to learn an independent appearance representation. We then propose a conditional image generator for realistic image generation using the learned disentangled latent representations. The network is trained only from unpaired image data, which only requires an original image and a new pose as inputs in order to generate a target image with desirable appearance and pose. Figure 6.1 illustrates our problem setup and motivation.

To this end, we develop an encoder-decoder network consisting of three components: an appearance encoder module that extracts the appearance cues and a pose encoder represents the target pose from a pose input, while their outputs are fed into a decoder network to produce the desirable target image. To learn the net-

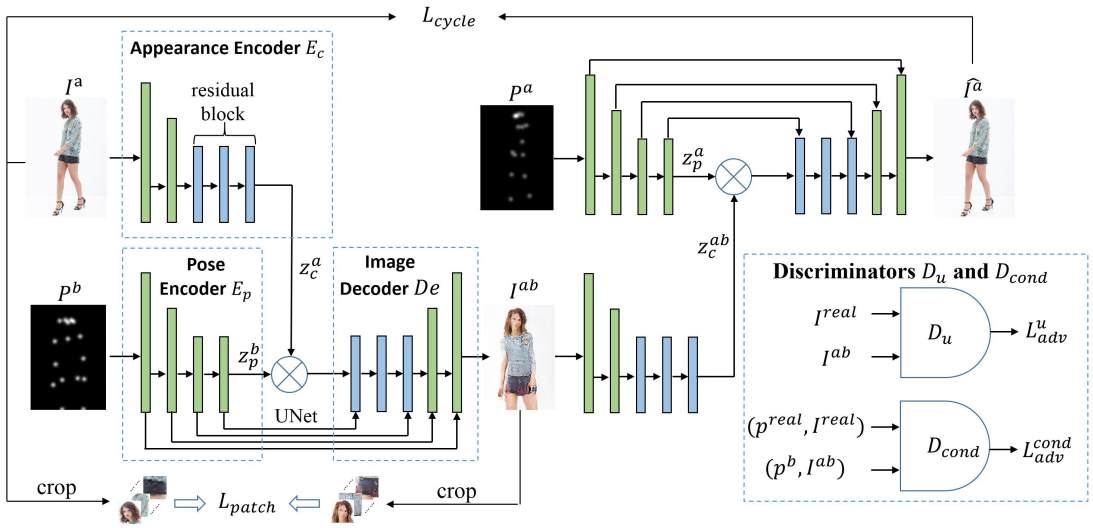


Figure 6.2: The overview of the network.

work from unpaired image data, we design a novel generating loss that integrates an image-level cycle consistency constraint based on pose annotations and an image-level adversarial loss, as well as a patch-level consistency loss. Our main insight is that such a dual-level loss enables the learning process to implicitly align the data and disentangles the appearance representation from the pose.

Later in Section 6.3.2, We demonstrate the capability of our proposed method on two distinct datasets, one of human bodies (DeepFashion) and the other of faces (CK+). We obtain interesting and convincing results on both datasets, with favourable comparisons to the state-of-the-art approaches.

Formally, let us denote by $I^a \in R^{H\times W\times 3}$ the RGB image with the height and width being H and W, respectively, and by $p^a \in R^{N_p\times 2}$ the corresponding pose vector, where N_p is the number of the 2D key-point coordinates (x, y). With the superscript a, the appearance I^a and pose p^a are supposed to describe the same object (e.g., , the same person or face). Our task is thus to learn a generator $G(I^a, p^b)$ where the appearance and pose are not necessary corresponded to the same object during model training, i.e., $a \neq b$. The core of our method is how to extract the disentangled representation z_c^a from the appearance image that is independent to the pose feature z_p^a. Once the disentanglement is achieved, we can combine the extracted feature z_c^a with any other pose to deliver novel images. As discussed before, we do not have ground truth information to guide the generation process, thus how to train our model requires specific network design, which is the topic of this section.

6.2.1.1 Architecture Design

We first present the details of our adversarial network including a **Generator** and a **Discriminator**, then we exploit two types of consistencies to further enhance the training of our model. The overview of the network is shown in Figure 6.2.

Generator Our generator network consists of three main modules: an Appearance Encoder E_c, a Pose Encoder E_p and an Image Decoder D_I. The Appearance Encoder E_c takes I^a as input, and obtains the feature z_c^a as output through stacked convolutional layers. Similarly, the Pose Encoder E_p derives the feature map z_p^b of p^b also by a set of convolutional layers. The concatenation of z_c^a and z_p^b will be fed to the Decoder D_I for image generation, namely, we attain the mixed image by $I^{ab} = D_I(E_c(I^a), E_p(p^b))$. Ideally, I^{ab} should present the same appearance characteristic as I^a but under the pose of p^b. Specially, the generated image I^{ab} should reconstruct I^a itself when $a = b$.

In particular, for the pose stream, we apply the UNet-like skip connections between the layers of the encoder and decoder following [Ronneberger et al., 2015]. We also utilise residual connections in the appearance encoder and decoder to preserve high frequency information.

Discriminator To teach the generator to output photo-realistic images, we follow the GAN [Goodfellow et al., 2014] framework and use a discriminator C_u[1] (called unconditional discriminator) on I^{ab} for adversarial training between the generated images and real ones. Besides, we define another discriminator C_{cond} (called conditional discriminator) on pairs of the appearance and pose images. The conditional discriminator is critical to get around the shortcut problem [Hu et al., 2017], namely preventing the case when I^{ab} contains little pose information from p^b. We will discuss this more in Eq. 6.2 later.

Cycle Consistency Cycle consistency has widely been proved successful in unpaired image-to-image translation [Zhu et al., 2017a; Liu et al., 2017b; Pumarola et al., 2018a; Choi et al., 2017]. Here, we take advantage of this in our framework for enhanced training. By re-encoding the generated image I^{ab} with the appearance encoder, we hope to extract the same appearance features as I^a. To do so, we further encourage our model to reconstruct the image I^a by mixing the appearance feature of the generated image I^{ab} with its associated pose feature from p^a.

Patch Consistency Although there is no ground-truth image to guide the generation for I^{ab}, we do have the fact that I^{ab} should resemble I^a locally around every keypoints because they have the same appearance characteristics. Therefore, we extract K small patches around each keypoint position on both I^{ab} and I^a, and then resize them to the same spatial dimensions. We denote by x_i^{ab} and x_i^a the ith resized spatial patches of the generated and input images, respectively. The difference between $\{x_i^{ab}, i = 1..K\}$ and $\{x_i^a, i = 1..K\}$ will be minimised by our patch-consistent loss which is explained in detail in Eq. 6.4.

6.2.1.2 Model Training

Motivated by the discussions above, we apply three losses: the *adversarial loss* that enables the photo-realistic generation; the *cycle consistency loss* that further enhances

[1] As the discriminator is basically a binary classifier, we use the C notation here to avoid namespace collision with the image decoder

the network training; and the *patch consistency loss* that allows local refinement of the generated images. We describe them separately below.

Image Adversarial Loss Various extensions such as WGAN [Arjovsky et al., 2017] and WGANGP [Gulrajani et al., 2017] have been proposed since the original GAN framework [Goodfellow et al., 2014]. Here, for the simplicity, we use the original GAN in this study. As discussed in Section 6.2.1.1, we make use of two discriminators for adversarial training. For the unconditional one, we define

$$\mathcal{L}_{adv}^{u} = \log(1 - C_u(I^{ab})) + \log(C_u(I^a)), \tag{6.1}$$

where the generated image $I^{ab} = D_I(E_c(I^a), E_p(p^b))$. Note that our generator should not only be able to hallucinates new images I^{ab} when $a \neq b$, but also to reconstruct I^a when $a = b$. To further enforce this property, we manually sample the appearance and pose pairs of the same object, and train our network by the unconditional adversarial loss every n iterations.

For the conditional discriminator, we define

$$\mathcal{L}_{adv}^{cnd} = \log(1 - C_{cond}(p^b, I^{ab})) + \log(C_{cond}(p^a, I^a)). \tag{6.2}$$

Unlike the unconditional loss, the discriminator in Eq. 6.2 additionally takes the pose as the input. As introduced by [Hu et al., 2017], disentanglement learning usually faces the short-cut issue where the pose feature of p^b is completely covered by the appearance feature of I^a and the network simply learns an identity mapping between I^a and I^{ab}. Using the conditional loss, we are able to alleviate this problem, as the comparisons between fake and real samples are conditional on the pose images. In other words, we encourages that $I^{ab} \neq I^a$ when $p^a \neq p^b$

Cycle Consistency Loss The generated image I^{ab} can also be fed to the appearance encoder again to reconstruct the original image I^a. We achieve this by feeding the pose p^a and image I^{ab} to the generator defined in Section 6.2.1.1, and enforce the output to be close to I^a by minimising the ℓ_1 loss as

$$\mathcal{L}_{cyc} = ||I^a - \widehat{I^a}||_1, \tag{6.3}$$

where $\widehat{I^a} = D_I(E_c(I^{ab}), E_p(p^a))$.

Patch Consistency Loss Given the resized keypoint patches $\{x_i^{ab}, i = 1..K\}$ and $\{x_i^a, i = 1..K\}$ from I^{ab} and I^a, respectively, we compute the following ℓ_1 loss by

$$\mathcal{L}_{patch} = \sum_i ||x_i^a - x_i^{ab}||_1, \tag{6.4}$$

Note that the L1 loss encourages pixel-wise similarities between the two patches, which is not always guaranteed due to the different spatial structures between them. However, we find that this loss works well in our experiments. The patch consistency encourages visual similarities between patches regardless of the locations they are

drawn from their respective images. Therefore, by minimising this loss the network are trained to obtain better disentangled representations.

The patches are extracted based on the keypoints coordinates given by the pose vector. Therefore by encouraging the two patches to be the same, the loss L_{patch} is guiding the network to extract appearances information from the relevant locations from I^a based on p^a, and apply them onto the correct locations on $\hat{I}^c$ based on p^b. As a result, the appearance features are independent to its original structure.

Full Loss. Putting above losses altogether, we attain the full loss as

$$\mathcal{L} = \lambda_1 \mathcal{L}_{adv}^{cnd} + \lambda_2 \mathcal{L}_{adv}^{u} + \lambda_3 \mathcal{L}_{cyc} + \lambda_4 \mathcal{L}_{patch}, \tag{6.5}$$

where $\lambda_1, \lambda_2, \lambda_3$ and λ_4 are hyper-parameters to control the weights of the four components, namely the conditional and unconditional adversarial loss, cycle and patch consistency loss respectively. The encoders E_c and E_p, the decoder D_I, and the discriminators D_u and C_{cond} are trained by

$$\arg \min_{E_c, E_p, D_I} \max_{D_{un}, C_{cond}} \mathcal{L}. \tag{6.6}$$

Similar to [Villegas et al., 2017b], the pose images are obtained by computing the grey-scale heat map from the 2-D keypoint coordinates. More specifically, we add Gaussian noises around each position to diffuse the pose spatially, which we find can further facilitate the network training.

6.2.2 Learning Disentangled Representations by Shape Mask Refinement

In this work, we aim to automatically infer better appearance and shape features from two randomly chosen images only, and generate novel image accordingly. To this end, we propose an end-to-end unsupervised network integrating prior knowledge of human pose detection, to automatically extract disentangled representations of both appearance and shape. Here we confine the target object to only humans. The human pose prior is adaptively refined with our training objectives, producing a more comprehensive shape descriptor. We argue that the learned shape and appearance descriptors, compared to explicit attribute cues, produce cleaner feature representations and are better suited for the task of human shape and appearance transfer. Also, not requiring any extra information input makes our network more flexible and widely applicable. Figure 6.3 illustrates our problem setup and motivation. Our goal is to generate novel realistic images based on two input images that respectively supply the desired appearance and shape conditions. We achieve this in an unsupervised manner where neither the ground truths of the output nor attribute-specific cues (*e.g.* human skeletons). The model we learn is able to automatically disentangle shape from appearance, and thus can generate images under arbitrary combination of them.

Specifically, our proposed method explores cycle consistency and adversarial training to simultaneously encourage independence and high expressive power of the learned features. The network consists of three components: A shape encoder

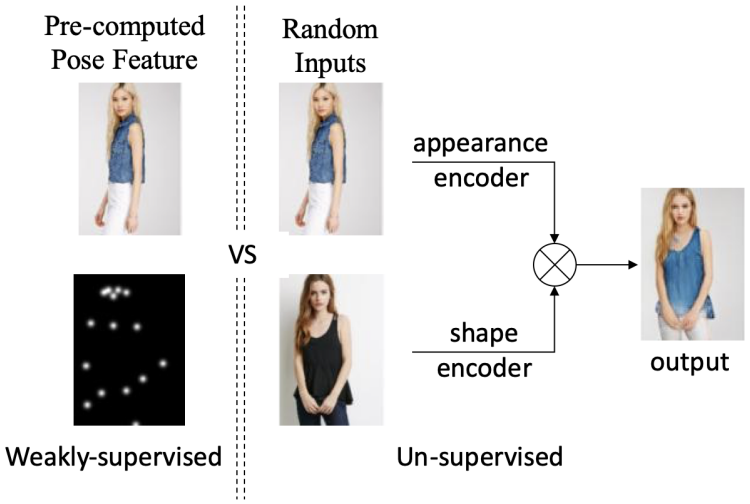


Figure 6.3: The problem setup of the unsupervised approach. We aim to learn disentangled appearance and shape representations from only randomly chosen images. There are no ground truth images nor any attribute-specific cues to guide the disentanglement and generation process. **Left**: Existing methods. The challenge is to learn disentangled appearance features given an image and its corresponding shape representation. We call this *weakly-supervised disentanglement*. **Right**: The problem setup of our approach. The challenge is to learn disentangled appearance and shape representations given only an image, without any task-specific cues. We call this *unsupervised disentanglement*. The output in this figure is a novel image generated by our framework.

that generates a set of masks corresponds to different body parts; An appearance encoder that capture appearance characteristics based on the shape masks; And finally a image decoder that takes the concatenated input of appearance features and shape masks to generate realistic images with desired attributes.

We demonstrate the capability of our proposed method on two different datasets: DeepFashion [Liu et al., 2016b] and Market1501 [Zheng et al., 2015]. We obtain convincing results on both datasets, which are shown in Section 6.3.3, with favourable comparisons to the state-of-the-art approaches.

Formally, let us denote by $I^a, I^b \in R^{H\times W\times 3}$ the two RGB images of the height H and width W. We suppose the shape and appearance representations for image I^a to be z_s^a and z_a^a, respectively, and define the features z_s^b and z_a^b of I^b similarly. Our task is thus to learn an image decoder $D_I(z_a^a, z_s^b)$ that combines the appearance from I^a with the shape of I^b to produce a novel image $\hat{I}^{mix}$ when $a \neq b$. The core of our method is how to encourage the disentanglement between the shape and appearance features, thus we can expect a realistic looking mixed image if we retain either of them but freely alter the other. As discussed before, we strive to address the problem with no external supervision, hence how to train our model requires specific network design, which is the topic of this section. We first present details of the network, then define

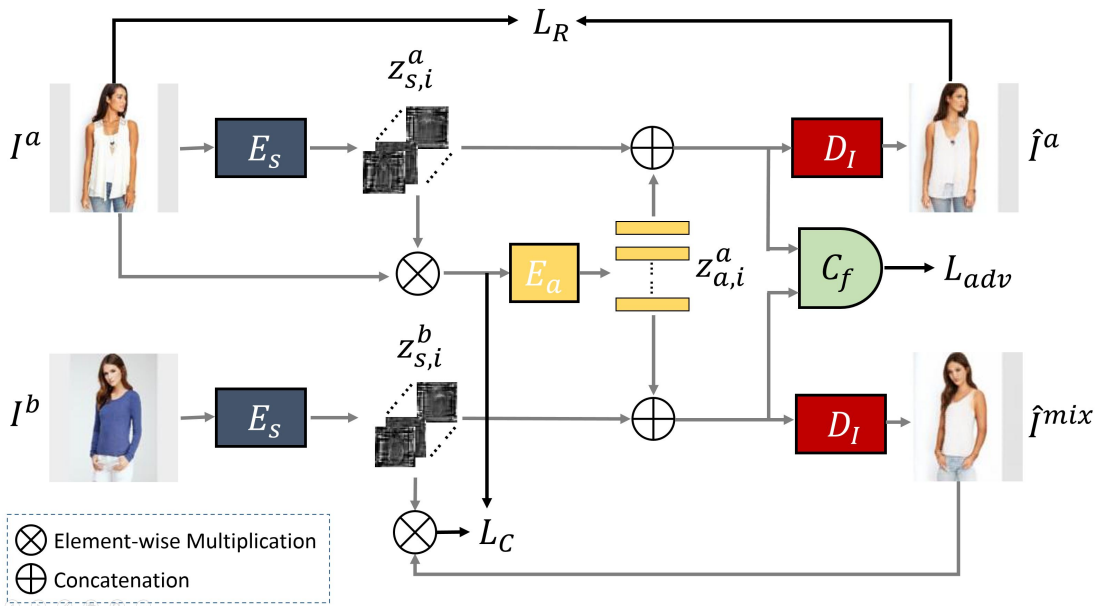


Figure 6.4: The overview of the network.

the objective functions and explain our training strategy.

6.2.2.1 Architecture Design

The network consists of four modules: 1. a shape encoder E_s that generates a set of shape masks; 2. an appearance encoder E_a that encodes appearance contents attended by the shape masks from E_s; 3. an image decoder D_I that generates photo-realistic images based on the concatenation of the shape and appearance features; 4. a feature classifier that learns to classify whether the given appearance and shape features are from the same image or not.

The overview of the network is shown in Figure 6.4. We generate 14 masks of size 16x16 for each input image, $\mathbf{z}_{s,i} \in R^{16\times16}$, $i = 1 \ldots 14$. The output of the appearance encoder E_a is a vector of length 32. All 14 output vectors are then concatenated to form a single vector of length 448. In order to concatenate this appearance vector with shape masks for decoding, we tile this vector to the same spatial dimension as the masks, and concatenate along the channel axis.

Shape and Appearance Encoders The shape encoder E_s is a convolutional network that down-samples the input image I^a and emit a set of m masks $\{z^a_{s,i},\ i = 1...m\}$, where the value of each pixel in $z^a_{s,i}$ is within $[0, 1]$. These masks serve dual purposes: one to filter local regions of the input images and the other one to provide conditional shape input to the image decoder D_I. Particularly for the first purpose, each of the m masks is first resized spatially and broadcasted along channels to match the size of the input image, and then is applied to the input image by element-wise multiplication, thus resulting in m filtered images. All these filtered images are passed through

the appearance encoder E_a (also a down-sampling convolution network) to deliver m appearance feature vectors, namely, $z_a^a = \{z_{a,1}^a, z_{a,2}^a ... z_{a,m}^a\}$. It should be noted that, the fusion of the shape and input images prior to the input of the appearance encoder makes our method clearly distinct from those conventional approaches [Pumarola et al., 2018b; Esser et al., 2018] where the encoding processes of both attributes are independent to each other. We contend that, the spatial-attended fusion is reasonable and efficient, as the shape mask should be naturally combinative to the raw image content and the filtered image could focus more on the meaningful regions of the person.

Image Decoder The shape and appearance representations z_s and z_a are concatenated as input to the image decoder D_I for image decoding. As is shown in Figure 6.4, we simultaneously decode reconstruction image $\hat{I}^a = D_I(z_s^a, z_a^a)$ and generate novel image $\hat{I}^{mix} = D_I(z_s^b, z_a^a)$ using their corresponding features. The reconstructed $\hat{I}^a$ is associated with an reconstruction loss to satisfy the cycle consistency, while the novel $\hat{I}^{mix}$ is used to compute the colour consistency loss. Details are explained in Section 6.2.2.2.

Feature Classifier Another core assumption of our method is that the distributions of the shape masks extracted from different persons should be consistent and identical. This assumption is crucial, as it enables the transferring ability of the pose from one person to another. To do so, we propose a feature classifier C_f which takes as input the concatenation of the shape and appearance representations, and determines if they are from the same person. For instance in Fig 6.4, we have the true pair $[z_s^a, z_a^a]$ and the false one $[z_s^b, z_a^a]$. The classifier will output 1 when $a = b$ and 0 otherwise. We then train the classifier and the encoder in an adversarial way to make close the distributions of the shape masks z_s^a and z_s^b.

6.2.2.2 Model Training

Motivated by the discussions above, we apply three losses: the *reconstruction loss* that trains the network to generate photo-realistic images; the *feature adversarial loss* that ensures universal representative ability of the shape mask; and the *colour consistency loss* that encourages consistent appearance transfer. Below we describe them in details.

Reconstruction Loss Conventional pixel-wise reconstruction loss such as L1-norm enforces exact alignments between two images. This yet is too rigorous for our framework because certain minor granularity is inevitably lost if we attempt to disentangle shape from appearance. Therefore, we adopt the image perceptual loss [Chen and Koltun, 2017; Johnson et al., 2016] instead to constrain the reconstructed image to be perceptually the same as the original. Formally, the reconstruction loss is:

$$\mathcal{L}_R = \sum_k \phi_k ||\Phi_k(I^a) - \Phi_k(D_I(z_s^a, z_a^a))||_1 \tag{6.7}$$

where Φ is the VGG [Simonyan and Zisserman, 2014b] network used to compute the perceptual similarities measured at the kth layer, and ϕ_k controls its weight.

Feature Adversarial Loss We impose this loss via an adversarial training between the feature classifier C_f and the shape/appearance encoders E_s and E_a. The feature classifier is trained to correctly classify whether its inputs are from the same image or not, while the encoders are trained to fool the classifier. We employ the LSGAN framework [Mao et al., 2017] to update the networks because of its simplicity and fast convergence. We leave the application of other GAN variants such as WGANGP [Gulrajani et al., 2017] in the future work. Specifically, we update the feature classifier with

$$\mathcal{L}_{adv}(C_f) = (C_f(z_s^a, z_a^a) - 1)^2 + (C_f(z_s^a, z_a^b))^2 \tag{6.8}$$

Conversely, we update encoders to output features that can 'fool' the classifier:

$$\mathcal{L}_{adv}(E_s, E_a) = (C_f(z_s^a, z_a^a) - 1)^2 + (C_f(z_s^a, z_a^a))^2 \tag{6.9}$$

Colour Consistency loss We add this regularisation term to our framework to ensure the appearance statistics remain unchanged when transferred to other shapes. In particular, we enforce colour consistency between the filtered patches[2] of the generated image and those of the original image. Let us define x_n to be the nth pixel in the ith image patch $Iz_{s,i}$, we can then calculate the mean μ and variance σ of that sample by $\mu_i = \sum_{\mathcal{N}} x_n / N$ and $\sigma_i = \sum_{\mathcal{N}} (x_n - \mu_i)^2 / N$ where $\mathcal{N}$ is the total number of pixels in the image.

We then impose colour consistency by minimising the squared difference of the colour statistics:

$$\mathcal{L}_C = \frac{1}{m} \sum_{i}^{m} ((\mu_i^{mix} - \mu_i^a)^2 + (\sigma_i^{mix} - \sigma_i^a)^2) \tag{6.10}$$

where m is the number of shape masks.

Full Loss Putting above losses altogether, we attain the full loss as

$$\mathcal{L} = \lambda_1 \mathcal{L}_R + \lambda_2 \mathcal{L}_{adv} + \lambda_3 \mathcal{L}_C \tag{6.11}$$

where λ_1, λ_2 and λ_3 are hyper-parameters to control the weights of the three components. We sample mini-batches of randomly chosen image pairs (I^a, I^b) to train our network.

[2] In our case, we refer $Iz_{s,i}$ as the i-th image patch, as opposed to the conventional definition of image patch', which often refer to a cropped part.

6.3 Experiments

In the experiments section, we present extensive experiments to prove the effectiveness of both the proposed weakly-supervised and unsupervised methods. We note that the two methods are fundamentally different with incomparable experimental setup, thus we choose to separate the experiments for each method into independent subsections of Section 6.3.2 and 6.3.3 respectively. Despite the difference in methods and implementation, the evaluation metrics are the same, which will be explained in Section 6.3.1

6.3.1 Evaluation Metrics

For the task of image translation or generation, the goal is to evaluate the realism and diversity of the generated images. Two main metrics are Structural Similarity Score (**SSIM**) [Wang et al., 2004] and Inception Score (**IS**) [Salimans et al., 2016].

SSIM score indicates the perceived quality of a target image against a reference image. It requires two images as a pair in order to the compute the similarities between them. Unlike traditional metrics such as PSNR or MSE, SSIM is a perception-based model that considers image degradation as perceived structural information with the observation that the pixels have a strong inter-dependencies especially when they are spatially close. SSIM score is measured on local image patches. Let x_i, y_i denote the ith patch of image X and Y, the SSIM score is calculated as:

$$SSIM(X,Y) = \mathbb{E}_i \frac{(2\mu_{x_i}\mu_{y_i} + c_1)(2\sigma_{x_i y_i} + c_2)}{(\mu_{x_i}^2 + \mu_{y_i}^2 + c_1)(\sigma_{x_i}^2 + \sigma_{y_i}^2 + c_2)} \tag{6.12}$$

where μ_x and μ_y are the averages of patches, σ_x and σ_y are the variances, σ_{xy} is the covariance of the two corresponding patches. c_1 and c_2 are two variables to stabilise the division with weak denominator. In our experiments, we use the built-in `skimage` function `skimage.measure.structural_similarity()` to get the measurements.

IS score is introduced in [Salimans et al., 2016] as a way to evaluate generative models, as there were no objective way to measure the quality and diversity of the generated images other way visual evaluation. The computation of IS involves the use of a pre-trained deep neural network model to classify the generated images. The probability distribution across the set of pre-trained classes are summarized in to the inception score. The score is a refection of two aspects of the image:

1. Image quality: can the image be classified into one of the classes?

2. Image diversity: is a wide range of objects being generated?

As a result, IS is not ideal in the task addressed in this chapter, because we are only aiming to generate person images given some pose and shape cues. Diversity is actually a undesired feature in such conditional image generation task. Also, depending on the pre-trained model used, "person" may not be a class in that model. We include this evaluation metric just to give more context to the reader.

6.3.2 Weakly Supervised Image Generation

In order to show the versatility of the proposed method, we test the network on two distinct datasets: DeepFashion [Liu et al., 2016b] for human images and CK+ [Lucey et al., 2010] for faces.

For **DeepFashion**, we use the In-shop Clothes Retrieval Benchmark of DeepFashion which contains 52712 of in-shop clothes images. The dataset offers various different clothes on many different persons. All back-view images are filtered out because it is hard to distinguish between back-view and front-view based on pose images. Including back-view ones will result in training instability. We use OpenPose [Cao et al., 2017] to extract poses for each image prior to training our network because the pose keypoints are not given for this dataset. We configure Openpose to use the *COCO* model, which outputs 18 keypoints. For **CK+**, the dataset has 10701 frames of different facial expressions performed by different persons. Every frame is FACS coded and comes with 68 landmarks annotation. This being a video dataset, image pairs of the the same person in different facial expressions are available. However, we do not exploit such paired information and treat each frame as an independent image.

6.3.2.1 Implementation Details

The network is optimized using Adam optimizer [Kingma and Ba, 2014] with initial learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is decreased by 10% every 2000 iterations. The batch size is set to 16 primarily due to hardware limitations. We update the generator and the discriminators alternately on every step. Also the hyper-parameters in Eq. (6.5) are simply set to 1, which we find to be good enough. This also indicates the simplicity of the model because little parameter tuning is required.

6.3.2.2 Appearance & Pose Transfer

We show that we can swap the appearance of an image to other appearances that we desire while leaving the pose untouched, or vice versa. The results also demonstrate the disentangle ability of our network.

The upper part of Figure 6.5 shows the same appearance applied on different poses while the lower part shows the same pose with different appearances on DeepFashion. Similar image manipulation results on CK+ dataset are shown in Figure 6.6. It can be observed that we are capable of arbitrarily pair appearances and poses and generate desired images that are hard to distinguish from real images.

DeepFashion It can be seen from the DeepFashion results, the network has learned to identify different body parts, so that it put different clothing on the correct place in the generated images, such as shirts, jeans and hats etc. In the lower part of Figure 6.5, it is more obvious that the appearance features of the person are successfully disentangled no matter what the pose is.

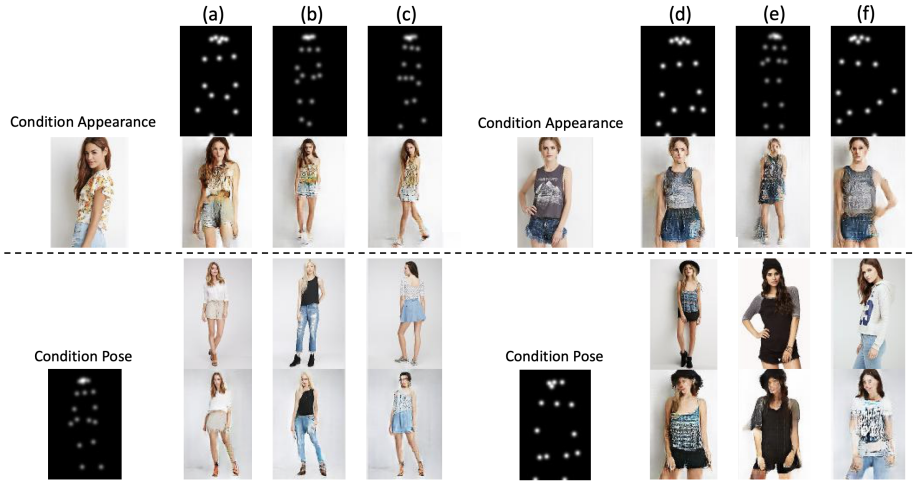


Figure 6.5: Image Manipulation Results on DeepFashion. The pose images and the appearance images are all randomly chosen. Images of upper body, whole body, side view, male and female are mixed together to show the robustness of our method. *Upper*: same appearance with different poses; *Lower*: same pose with different appearances.

CK+ The results are more interesting on CK+ dataset. The pose here is defined by the 68 facial landmarks. One of the attributes of the facial landmarks is that they not only define the facial expression, but also define the shape of eyes, nose, mouth, jawline etc and their relative positions. As a result, in addition to possessing the same expression defined by the landmarks, the generated face often look like their siblings from the original image rather than the exact same person. Such effect can be observed from upper part of Figure 6.6. This effect also indicates that the appearance features get extracted and transferred to the new faces because otherwise such resemblance would not exist. It can also be observed that the clothing around the collar area also get transferred to the generated images.

Because our network disentangle the appearance features from the spatial structures, there is no need of facial alignments, which is quite an important operation in many other face synthesis methods [Pumarola et al., 2018a]. Actually, we purposefully introduce spatial displacement during the testing of our model by randomly crop the appearance images as well as the landmark images. As is shown in the upper right part of Figure 6.6, the 3 landmark images are of different scales and in different locations. Nonetheless, all the resulting images not only have the correct expressions, but also the correct scales and locations correspondingly. This proves that our method is robust to spatial displacements.

Naturally, if the faces are all aligned and we only aim to manipulate the expressions, i.e. generating images using the landmarks and appearance of the same person, the results are even better. The lower part of Figure 6.6 shows the expression manipulation of the same person. As can be seen from the figure, the results looks

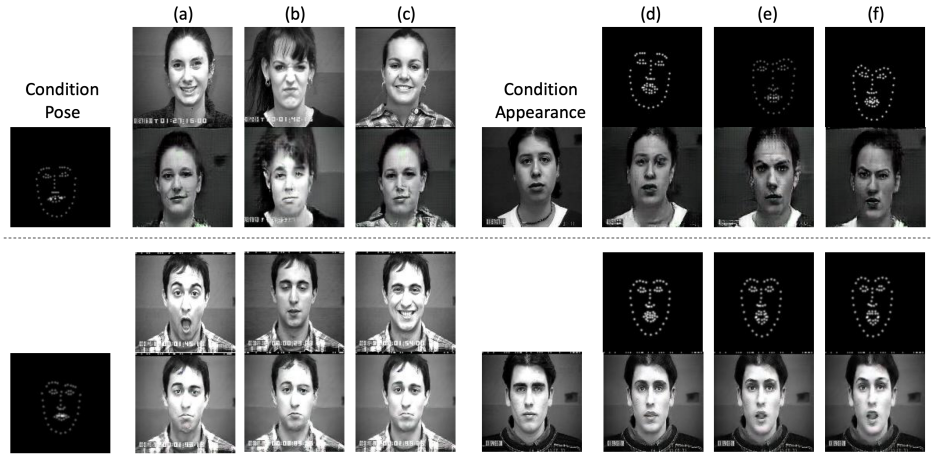


Figure 6.6: Image Manipulation Results on CK+. The landmark images and the appearance images are all randomly chosen. *Upper*: Unaligned face generation; *Lower*: Aligned face generation.

sharp and vivid. On the left side in the lower of Figure 6.6, we can generate nearly the same face based on the landmarks no matter how the conditional appearance changes; while the right side shows a natural progression of the expression, in line with the conditional landmarks.

6.3.2.3 Understanding The Disentangled Latent Features

With the disentangling capability of our network, we can visualise latent representations of appearance and pose simply by setting their complementary components to zero at the decoding stage. Figure 6.7 and Figure 6.8 shows the visualization of the appearance and pose on DeepFashion and CK+ respectively.

From the upper parts of both Figure 6.7 and Figure 6.8, the shape of bodies and faces can be barely identified. Instead, the visualisation shows prominent texture patterns such as hair, decor figures on the cloth and noses. On the other hand, The lower parts of the both figures clearly show human bodies and faces, but with blank textures. This is further proof that we achieve feature disentanglement.

It should be noted on Figure 6.8, it seems that providing only landmarks is sufficient for face generation. This is because landmarks provides rich information regarding the shapes of eyes, nose jawline etc. Therefore landmarks alone can dictate what the face should look like to some extent. Also the faces has less variation than human poses, thus the network simply generates an 'average' face when setting the appearance cues to all zeros.

6.3.2.4 Compare to other State-Of-The-Art

To put the performance of our method into context, we compare it with two state-of-the-art methods, one of human image generation [Ma et al., 2018b] and the other

Figure 6.7: Visualise appearance and pose latent features on DeepFashion.

of face synthesis [Pumarola et al., 2018a]. To the best of our knowledge, there are no existing methods that are proposed in the same problem setup as ours. Therefore, we chose to compare the methods that have the most similar problem setup to ours, so that the comparisons are most relevant. We also tweak the experimental setup of [Pumarola et al., 2018a] slightly for a more direct comparison.

Visual Comparison Figure 6.9 (a) compares our results with that in [Ma et al., 2018b] on DeepFashion. The problem setup in [Ma et al., 2018b] is similar to ours regarding the use of only unpaired data. However, pose information is required during testing in their method, whereas we do not need such information. It can be seen from the figure that although the generated images in [Ma et al., 2018b] looks sharper, especially around the face area, our method performs better in transferring the appearance to other poses. For example, details such as the the grey jacket with a white shirt inside on the woman and the pattern on the chest of the man can still be observed in the generated images. On the contrary, such details are lost in [Ma et al., 2018b] as the textures are very plain in their results.

Figure 6.9 (b) shows the results on CK+. We adapted the method in [Pumarola et al., 2018a] by changing the AU codes to the landmark images. Although GANimation achieves stunning results in facial expression manipulation using AU codes, it performs poorly in the case of using landmarks. This is because AU codes only provide high level semantic information while landmarks enforce more constrains and also contain much subtle variations. In the case shown in Figure 6.9 (b), where the face area are not cropped out and the landmarks are not aligned, our method performs much better: the generated faces are of the correct expression and the location. The disentangled representation of our network is vital in applying the appearance onto other spatial structures, whereas the attention mask, which plays a critical role

Figure 6.8: Visualise appearance and pose latent representation on CK+.

in [Pumarola et al., 2018a], limits their ability for large spatial transformations.

Quantitative Comparison For quantitative comparison, we choose to use the SSIM score proposed in [Wang et al., 2004] and Inception Score (IS)[Salimans et al., 2016] to compare the quality and the diversity of the generated images in regards of the ground truth. We compare our method with BodyROI7[Ma et al., 2018b] because their experimental setup is most similar to ours. We also compare with the adapted version of [Pumarola et al., 2018a] in the case of unaligned faces. An ablation study is conducted and the results are shown in the Table 6.1. It can be seen that the Patch

CK+	GANimation[Pumarola et al., 2018a]	Ours no patch	Ours no D_{un}	Ours full
SSIM	0.21	0.22	0.34	**0.51**
DeepFashion	BodyROI7[Ma et al., 2018b]	Ours no patch	Ours no D_{un}	Ours full
SSIM	0.614	0.462	0.505	**0.618**
IS	3.228	3.237	3.089	**3.364**

Table 6.1: Quantitative Comparison and Ablation Study.

Consistency is critical in our method. Actually, the patch loss trains the network to disentangle the appearance from pose, because the loss encourage two patches of the same body part (or landmark) to be the same even though they are cropped from different locations. Without patch consistency, the disentanglement capability are mostly lost and our method performs very similar to that of [Pumarola et al., 2018a].

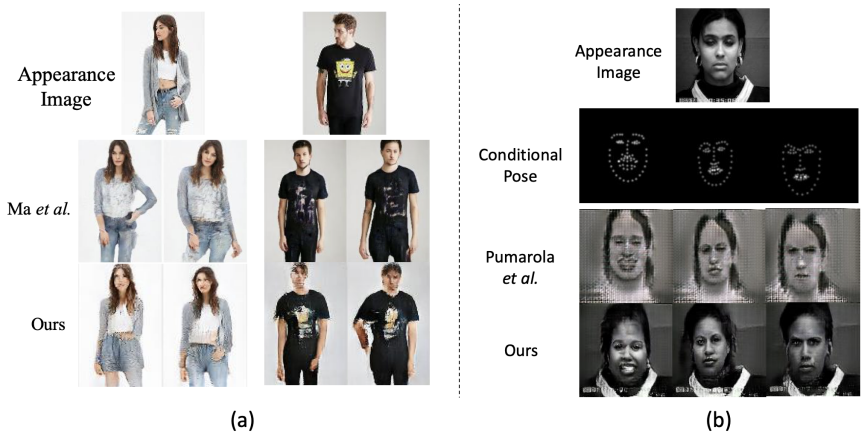


Figure 6.9: Results comparison on DeepFashion and CK+. (a) We can see that our method better preserves appearance cues from the original image. (b) It is obvious that our results are more realistic and the faces are correctly aligned with the conditional pose

6.3.3 Unsupervised Image Generation

In this section, we demonstrate the unsupervised disentanglement ability of the proposed method which can be applied more generally to randomly selected image. More importantly, we show that the learned data-driven representations offer superior performance than manually defined ones, such as the joint coordinates used in the previous section. We first evaluate the results on conditional person image generation in Section 6.3.3.2. Then we explain and visualise the disentangled representations learned by our network. After that, we present both qualitative and quantitative comparisons with current state-of-the-art algorithms, showing that we achieve favourable results against other weakly or even fully-supervised methods. Finally, an ablation study is conducted.

Regarding the proposed unsupervised disentanglement and image generation method, we confine the target object to humans, thus we selected two datasets of human images, **DeepFashion** [Liu et al., 2016b] and **Market1501** [Zheng et al., 2015]. Similar to the experiments on the weakly supervised method, we also use the In-shop Clothes Retrieval Benchmark of DeepFashion. Following the previous protocol [Esser et al., 2018], we filter out invalid images with no person present, and then randomly select 32988 images for training and 8245 images for testing. Note that back/side view images are all included for the experiments. For Market1501 dataset, it contains 322,668 images collected from 1501 persons. Each image is of spatial size 128x64. Joining the prior practice [Esser et al., 2018], we randomly utilise 9399 images for training and 3810 images for testing.

Figure 6.10: Appearance and shape transfer results on DeepFashion. The input images are all randomly chosen. Images of upper/whole body, side/front/back views, are shown here to demonstrate the robustness of our method. In each block, the appearance is inferred from the upper-left image, while shape is inferred from the upper-right one.

Figure 6.11: Appearance and shape transfer results on Market1501. In each block, the appearance is inferred from the upper-left image, while shape is inferred from the upper-right one.

6.3.3.1 Implementation Details

The network is optimised using Adam optimiser [Kingma and Ba, 2014] with initial learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is decreased by 5% every 2500 iterations, also, learning rate for the shape encoder E_s is further modified by a factor of 0.1. The batch size is set to 12 primarily due to hardware limitations. We update the generator(E_s, E_a and D_I) and the classifier (C_f) alternately on every step. The hyper-parameters in Eq 6.11 are set to be:

$$\lambda_1 = 0.01,\ \lambda_2 = 1,\ \lambda_3 = 1 \tag{6.13}$$

We initialize the shape encoder network with pretrained weights of OpenPose net [Cao et al., 2017] for stable training and fast convergence. The full details of the network structure is given in the appendix.

6.3.3.2 Appearance & Shape Transfer

We show that the proposed network can independently control the appearance and shape of an image based only on two conditional images, each providing cues for one of the attributes.

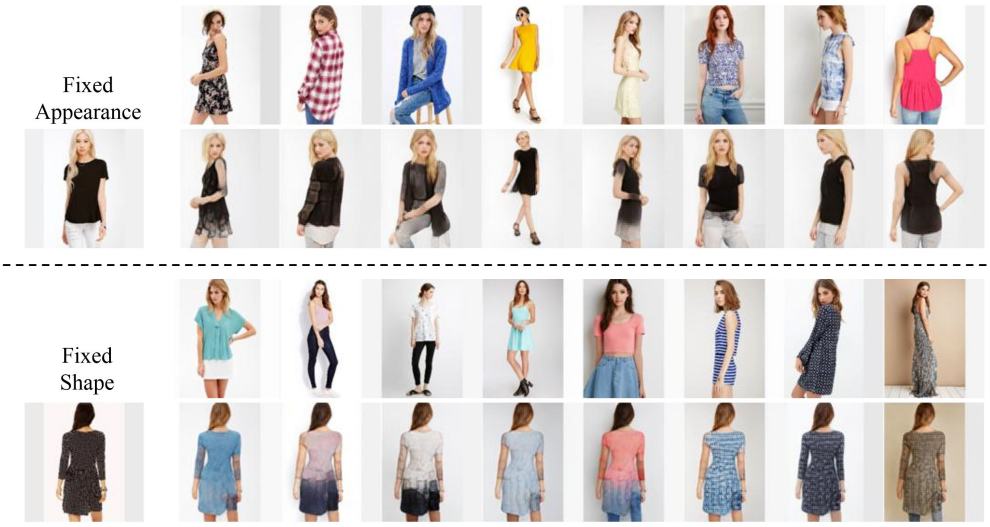


Figure 6.12: Demonstration of model stability. Upper: same appearance paired with different shape; Lower: same shape with different appearance. We can see the desired image attributes are well-preserved and consistent in both cases.

Figure 6.10 shows the image generation results on DeepFashion, where a mixture of upper/whole body images from side/front/back views are presented to show the robustness of our method. We can see the generated image preserves the conditional shape while being consistent with the appearance in colour. Even some detailed shape characteristics are successfully preserved during the transfer process. For example, in the second block of Figure 6.11, it can be observed that the hair style in the generated image is as natural as in the original one. Also the details around the corner of the cloth are well preserved. Similar image manipulation results on Market1501 dataset are shown in Figure 6.11 To demonstrate the stability of our model, transfer results using the same appearance image with multiple shape images (and the reverse) are shown in Figure 6.12. It is obvious that both the shape and appearance remain consistent and well-aligned with the conditional image across different generation results, proving good stability of the model.

6.3.3.3 Understanding The Disentangled Representation

As explained above, we use pre-trained OpenPose net as our shape prior, and adaptively refine the generated shape masks. The shape masks not only guide the appearance encoder to encode local patterns, but also serve as shape conditions for the image decoder. Figure 6.13 shows the details of shape mask refinement, as well as illustrate the positive effect it gives to image generation. We can see from Figure 6.13 (a) that shape masks changed dramatically after refinement. Although some background noise got encoded into the refined masks, they conform around the input shape much more tightly than the original ones, e.g., we can clearly see that the bottom left mask focuses on the lower legs and shoes while the original one only roughly

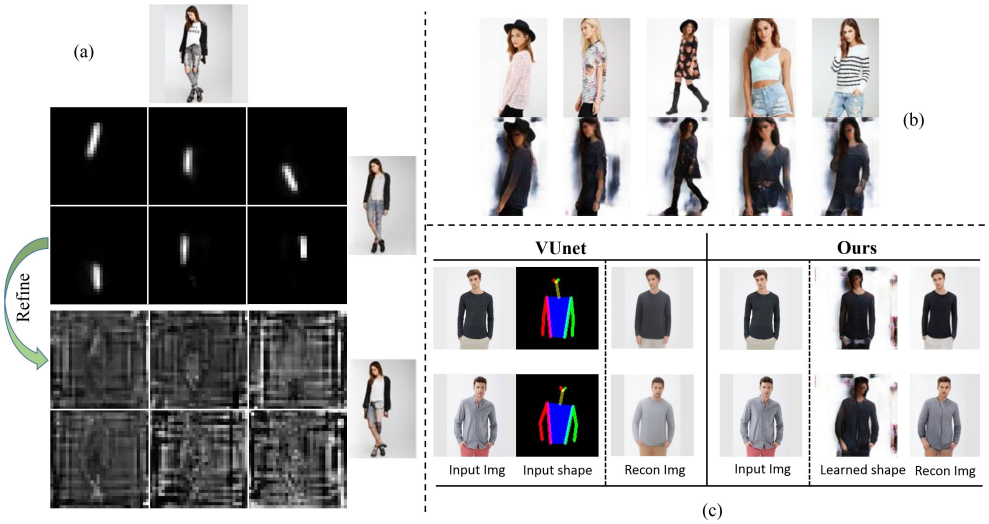


Figure 6.13: Visualisation of the learned shape masks. (a) shows the original masks obtained with pretrained OpenPose net, and the final refined masks using our learned model. What shown here are essentially $z_{s,i}$. (b) better visualise the learned shape masks by decoding $z_{s,i}$ with all-zero appearance features. (c) demonstrates the advantages of our learned masks over fixed stickman inputs employed in [Esser et al., 2018]. Comparison is carried on image reconstruction results.

indicates their location. Such mask refinement offer noticeable improvements in the quality of the generated images, as shown in the two reconstructed images on the right side. Figure 6.13 (b) shows better visualisation of the learned shape representation. In our framework, the shape encoder E_s generates a set of 14 masks, each focusing on a different body part. To better visualise the combined effect of all the masks, we decode images by setting the appearance features to all zeros, thus obtain clean shape visualisations of shape representation, which is shown in bottom row of (b). It is obvious that our network nicely preserve the input shape, including even small details such as hats (first column) and shoes (third column). Figure 6.13 (c) demonstrates the advantages offered by learning shape representations over using explicit shape cues. Comparing the reconstruction results between ours and [Esser et al., 2018], our images exhibits much more details and are hardly distinguishable to the original ones. Because our learned shape masks conforms to the input human contour, details like the gaps between arms and torso are well preserved.

6.3.3.4 Compare with Current State-Of-The-Art

To the best of our knowledge, our proposed method is the first unsupervised end-to-end framework that achieves automatic disentanglement and image generation without heavy pre/post-processing. We compare with three methods that are closest to ours: BodyRoI [Ma et al., 2018b], UPIS [Pumarola et al., 2018b] and V-Unet [Esser et al., 2018], where they also achieve unpaired image translation, but weak super-

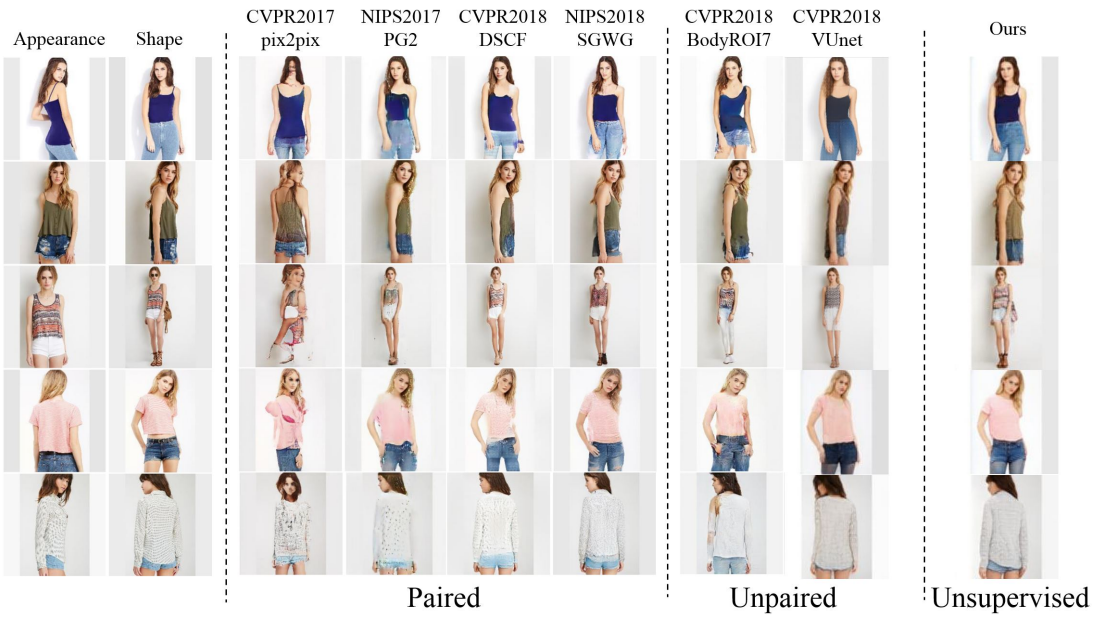


Figure 6.14: Visual comparison with current state-of-the-arts on DeepFashion. Zoom in for details.

vision (joint coordinates) are provided. To put the performance of our method into context, we also compare with supervised (paired) image translation methods, specifically PG2 [Ma et al., 2017b], DeformableGan [Siarohin et al., 2018] and SGWG [Dong et al., 2018]. As a general baseline, we also compare with pix2pix [Isola et al., 2017]. Experimental results shows that we compare favourably against all current state-of-the-arts, even supervised ones.

Visual comparisons are shown in Figure 6.14 and Figure 6.15 for DeepFashion and Market1501 respectively. We can see that we generate the most visual pleasing and realistic results with well-preserved details. We also conduct quantitative comparisons evaluated with SSIM and IS score. Results on both datasets are reported in Table 6.2, suggesting superior performance of our methods.

6.3.3.5 Ablation Study

We argue that the two most important contributing factors to our superior performance is *a learned more plausible shape representation* and *clean disentangled features* encouraged by feature adversarial and colour consistency loss. We conduct an ablation study to verify their claimed effects. Figure 6.16 shows results using three model variations. In the above figure, base model is the bare-bone network using fixed shape encoder trained with perceptual reconstruction loss only. The results indicates that, without feature disentanglement, the appearance contain much information about its original shape, yielding significant visual artefacts and incoherent human images. After adding the feature adversarial loss and colour consistency loss to encourage

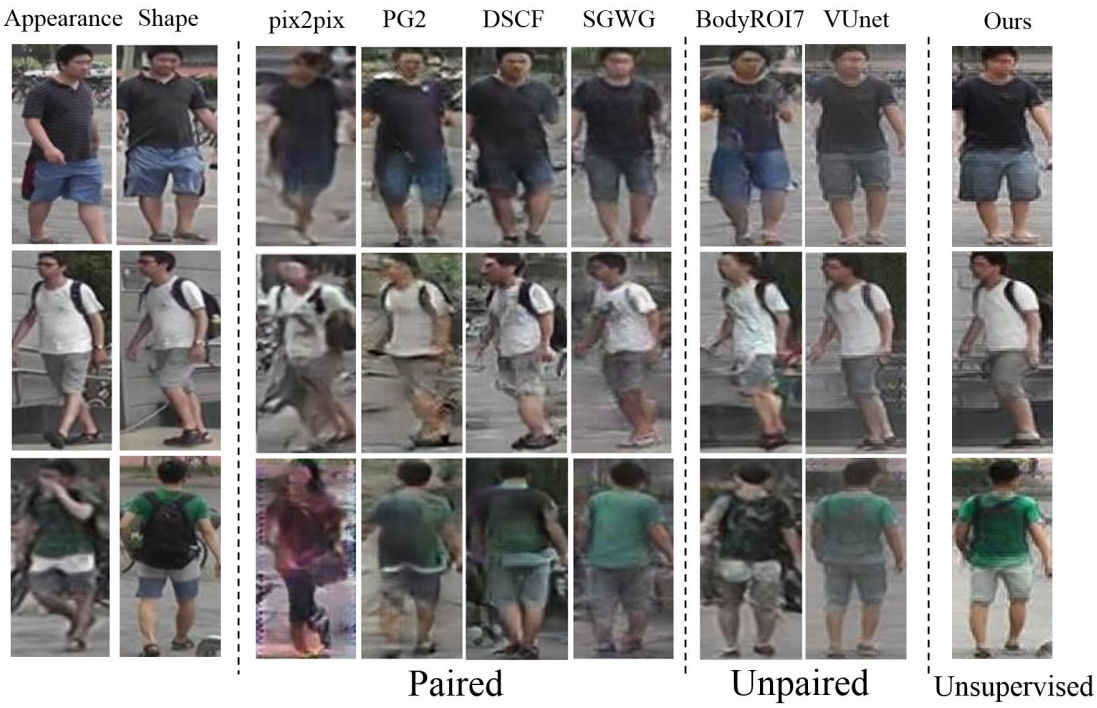


Figure 6.15: Visual comparison with current state-of-the-arts on Market1501.

disentanglement, results are noticeably better with different appearance patterns in the correct places. The performance are further improved by employing adaptive shape refinement in our full model. Refined shape offer much more details, such as body curvature, hair styles etc.

6.4 Conclusion

In this chapter, we have developed two realistic novel image generation methods that are suitable in weakly-supervised and unsupervised setups respectively. We demonstrate the effectiveness of both methods on different datasets of human bodies and faces. Experiments show that our methods can generate realistic results with the desired attributes. At the same time, our method are robust to spatial displacements like shifts, scale difference etc.

Surprisingly, our unsupervised framework that learns disentangled latent representations of appearance and shape through online mask refinement, out perform the weakly supervised method and even some fully supervised methods. Experiments results reports superior performance quantitatively and visually against many state-of-the-arts, including the supervised ones. Compared to existing methods using fixed shape cues, our learned shape representation encodes finer details, yielding more realistic images.

	DeepFashion		Market1501	
Methods	SSIM	IS	SSIM	IS
pix2pix[Isola et al., 2017]	0.692	3.249	0.183	2.678
PG2[Ma et al., 2017b]	0.762	3.090	0.253	3.460
DSCF[Siarohin et al., 2018]	0.761	**3.351**	0.290	3.185
SGWG[Dong et al., 2018]	0.793	3.314	0.356	3.409
UPIS[Pumarola et al., 2018b]	0.747	2.97	-	-
BodyROI7[Ma et al., 2018b]	0.614	3.228	0.099	**3.483**
VUnet[Esser et al., 2018]	0.786	3.087	0.353	3.214
Ours	**0.844**	3.096	**0.626**	2.48

Table 6.2: Quantitative Comparison using SSIM and IS score. Following [Esser et al., 2018] and [Dong et al., 2018], the SSIM score is calculated using reconstructed images in the test set. The first block are supervised methods using paired data; middle block are weakly-supervised methods requiring explicit shape inputs. Our unsupervised method achieve the highest score against all current state-of-the-art methods, including supervised ones.

Figure 6.16: Ablation study results comparing the performance of base model (B), base model with feature disentanglement (B+D), and our full model using both disentanglement and shape mask refinement (B+D+R).

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The overall goal of this thesis was to learn robust visual representations under progressive data scarcity. We argue that while new large-scale annotated datasets were constantly being published, they are not well suited for real-world applications. Therefore, there exists a growing desire for learning from limited supervision. To this end, we explore several computer vision tasks under different data scarcity setup:

In Chapter 3, we build a RGBD human action dataset called GADD that consists of gym exercises. The dataset posses desirable properties such as well-defined action boundaries, complex temporal structures of action instances and multimodal data format, it is therefore able to accommodate research on action recognition, localisation, unsupervised action discovery, RGB and depth fusion etc.

In Chapter 4, we fully explore the precious annotations available of a dataset to try to achieve better performance in action localisation. Specifically, we explore the annotations in both action instances level and local frames level and propose a fusion method that combines the supervision from the two levels. We utilise both our own GADD dataset and other public dataset to evaluate our fusion method, which is proven to boost action localisation accuracy.

In Chapter 5, we use limited data and address the challenging task of one/few-shot action localisation. In order to quickly capture the temporal dynamics of new action classes from just a few samples, we propose to learn similarity-based video descriptor which is independent on action classes. Classification is then carried out by comparing similarities between the test action instance and the few available training data. Our method achieve state-of-the-art performance in both one-shot action recognition and localisation.

In Chapter 6, we strip down as much supervision as possible and aim to learn disentangled visual representations and generate novel images in an unsupervised manner. First we utilise pre-extracted pose information and try to infer disentangled appearance features. We propose the combination of cycle consistency and patch consistency to learn such features through image generation. Then we moved one step closer to purely unsupervised learning by discarding the pre-extracted pose information. We propose an end-to-end framework that automatically extract disentangled representation for both pose and appearance. Surprisingly, our unsupervised

framework even generates more realistic images than some fully supervised (paired) methods.

Overall, we explored alternatives of learning with limited annotations and narrow the performance gap between supervised and weakly supervised learning for the tasks of action recognition, localisation, feature disentanglement and novel image generation.

7.2 Future Work

For the fully supervised action localisation framework, the current pipeline is divided distinctively into a detection part and a frame labeling part, which is inefficient and unscalable. It would be worth investigating how to combine faster-rcnn [Ren et al., 2015] or SSD [Liu et al., 2016a] like detection modules into our fusion network to form an end-to-end framework. Apart from that, it would also be interesting to replace our current labeling network with a more advanced seq2seq model, which has been proven to be effective for sequence labeling. Also, as a different way of incorporating instance information, graph convolution is a worthy candidate to explore long-term dependencies.

For the one-shot action localisation work, the main limitation is scalability, as shown in Figure 5.7 and 5.8. We think it has not been explored in depth how to augment a similarity based learning system with external memories and we believe it might offer better scalability. Also, in order to meet real-world one-shot task requirements, it is interesting to see the ont-shot performance when the meta-training is applied on more different dataset (higher domain discrepancy) than the evaluation dataset. For example, training on Thumos 14 dataset and then test on ActivityNet.

For the unsupervised disentangle project, we would like to continue with the journey of learning from limited annotation, we would like to go further to unsupervised disentanglement of shape and appearance. Specifically, instead of using pretrained OpenPose Net, which is trained on a closely related task (pose estimation), we would like to explore pretrained models on other general and unrelated task, such as image semantic segmentation [Long et al., 2015]. Moreover, we believe it would be beneficial to add more task-specific constraints to the shape masks (*e.g.* total variation regularisation) so that they conform to the contours of the body limbs and contain less noise.

Going forward, we would like to combine our research results to tackle more challenging problems of video synthesis. We want to borrow ideas like modelling instance information and the disentanglement framework to create realistic synthetic videos.

Generally speaking, we would dedicate our efforts toward unsupervised and transfer learning, with the hope to alleviate the burden of data annotation.

Bibliography

Altman, N. S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46, 3 (1992), 175–185. (cited on page 15)

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L., 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6077–6086. (cited on page 11)

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; and de Freitas, N., 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 3981–3989. (cited on pages 54 and 55)

Arjovsky, M.; Chintala, S.; and Bottou, L., 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, (2017). (cited on pages 20, 73, and 80)

Baddar, W. J.; Gu, G.; Lee, S.; and Ro, Y. M., 2017. Dynamics transfer gan: Generating video by transferring arbitrary temporal dynamics from a source video to a single target image. *arXiv preprint arXiv:1712.03534*, (2017). (cited on page 76)

Bahdanau, D.; Cho, K.; and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, (2014). (cited on pages 7, 8, and 11)

Baldi, P. and Pollastri, G., 2003. The principled design of large-scale recursive neural network architectures–dag-rnns and the protein structure prediction problem. *Journal of Machine Learning Research*, 4, Sep (2003), 575–602. (cited on page 8)

Bengio, Y.; Simard, P.; Frasconi, P.; et al., 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5, 2 (1994), 157–166. (cited on pages 7, 8, and 9)

Bertinetto, L.; Henriques, J. F.; Valmadre, J.; Torr, P.; and Vedaldi, A., 2016. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 523–531. (cited on page 15)

Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; and Gould, S., 2016a. Dynamic image networks for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3034–3042. (cited on pages 3, 33, 34, 36, 37, 38, and 68)

Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; and Gould, S., 2016b. Dynamic image networks for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 70)

Bishop, C. M., 2006. *Pattern recognition and machine learning*. springer. (cited on page 18)

Bojanowski, P.; Lajugie, R.; Bach, F.; Laptev, I.; Ponce, J.; Schmid, C.; and Sivic, J., 2014. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, 628–643. Springer. (cited on page 56)

Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; and Krishnan, D., 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 7. (cited on pages 74 and 76)

Buch, S.; Escorcia, V.; Shen, C.; Ghanem, B.; and Niebles, J. C., 2017. Sst: Single-stream temporal action proposals. (2017). (cited on page 66)

Caba Heilbron, F.; Carlos Niebles, J.; and Ghanem, B., 2016. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1914–1923. (cited on page 66)

Cao, Q.; Shen, L.; Xie, W.; Parkhi, O. M.; and Zisserman, A., 2018a. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, 67–74. IEEE. (cited on page 2)

Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.-E.; and Sheikh, Y., 2018b. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, (2018). (cited on page 5)

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 87 and 93)

Carreira, J. and Zisserman, A., 2017. Quo vadis, action recognition? a new model and the kinetics dataset. *arXiv preprint arXiv:1705.07750*, (2017). (cited on pages 53 and 56)

Chan, C.; Ginosar, S.; Zhou, T.; and Efros, A. A., 2018. Everybody dance now. *arXiv preprint arXiv:1808.07371*, (2018). (cited on page 73)

Chen, Q. and Koltun, V., 2017. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 1511–1520. (cited on page 84)

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P., 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, 2172–2180. (cited on pages 73 and 75)

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y., 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, (2014). (cited on page 11)

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y., 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, (2014). (cited on page 7)

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J., 2017. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint*, 1711 (2017). (cited on pages 74, 75, 76, and 79)

D., P., 2007. Human and animal cognition: Continuity and discontinuity. 104(35) (2007). (cited on page 53)

Dai, X.; Singh, B.; Zhang, G.; Davis, L. S.; and Qiu Chen, Y., 2017. Temporal context network for activity localization in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 5793–5802. (cited on pages 30, 53, 56, and 66)

Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. In *CVPR*, vol. 1, 886–893. IEEE. (cited on page 35)

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S., 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4690–4699. (cited on page 2)

Denton, E. L. et al., 2017. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems*, 4414–4423. (cited on page 76)

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 33, 34, 35, 40, and 53)

Dong, H.; Liang, X.; Gong, K.; Lai, H.; Zhu, J.; and Yin, J., 2018. Soft-gated warping-gan for pose-guided person image synthesis. In *Advances in Neural Information Processing Systems*, 472–482. (cited on pages xx, 73, 74, 96, and 98)

Esser, P.; Sutter, E.; and Ommer, B., 2018. A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8857–8866. (cited on pages xviii, xx, 5, 74, 75, 76, 84, 92, 95, and 98)

Fabian Caba Heilbron, B. G., Victor Escorcia and Niebles, J. C., 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 961–970. (cited on pages 2, 3, 53, and 64)

Fan, L.; Huang, W.; Gan, C.; Huang, J.; and Gong, B., 2018. Controllable image-to-video translation: A case study on facial expression generation. *arXiv preprint arXiv:1808.02992*, (2018). (cited on page 73)

Fei-Fei, L.; Fergus, R.; and Perona, P., 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28, 4 (2006), 594–611. (cited on page 14)

Feichtenhofer, C.; Pinz, A.; and Zisserman, A., 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1933–1941. (cited on pages 58 and 70)

Fernando, B.; Gavves, E.; Oramas, J. M.; Ghodrati, A.; and Tuytelaars, T., 2015a. Modeling video evolution for action recognition. In *CVPR*, 5378–5387. (cited on pages 34, 36, and 37)

Fernando, B.; Gavves, E.; Oramas, J. M.; Ghodrati, A.; and Tuytelaars, T., 2015b. Modeling video evolution for action recognition. In *CVPR*, 5378–5387. (cited on pages 36 and 38)

Fink, M., 2005. Object classification from a single example utilizing class relevance metrics. In *Advances in neural information processing systems*, 449–456. (cited on page 14)

Finn, C.; Abbeel, P.; and Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, (2017). (cited on pages 15, 54, 55, and 62)

Finn, C.; Xu, K.; and Levine, S., 2018. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 9516–9527. (cited on page 15)

Frey, S. H. and Gerry, V. E., 2006. Modulation of neural activity during observational learning of actions and their sequential orders. 26 (2006). (cited on page 53)

Gao, J.; Yang, Z.; Sun, C.; Chen, K.; and Nevatia, R., 2017. Turn tap: Temporal unit regression network for temporal action proposals. *arXiv preprint arXiv:1703.06189*, (2017). (cited on pages 53 and 56)

Gatys, L. A.; Ecker, A. S.; and Bethge, M., 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, (2015). (cited on page 1)

Girshick, R., 2015. Fast r-cnn. In *ICCV*, 1440–1448. (cited on pages 37, 53, and 56)

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 580–587. (cited on pages 37 and 41)

Gong, K.; Liang, X.; Li, Y.; Chen, Y.; Yang, M.; and Lin, L., 2018. Instance-level human parsing via part grouping network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 770–785. (cited on page 75)

Gong, K.; Liang, X.; Zhang, D.; Shen, X.; and Lin, L., 2017. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 932–940. (cited on page 75)

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y., 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680. (cited on pages 18, 19, 73, 79, and 80)

Graves, A.; Mohamed, A.-r.; and Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6645–6649. IEEE. (cited on page 40)

Graves, A.; Wayne, G.; and Danihelka, I., 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*, (2014). (cited on page 15)

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C., 2017. Improved training of wasserstein gans. In *Advances in neural information processing systems*, 5767–5777. (cited on pages 20, 73, 80, and 85)

Han, X.; Wu, Z.; Wu, Z.; Yu, R.; and Davis, L. S., 2018. Viton: An image-based virtual try-on network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7543–7552. (cited on pages 74 and 76)

Herath, S.; Harandi, M.; and Porikli, F., 2016. Going deeper into action recognition: A survey. *arXiv preprint arXiv:1605.04988*, (2016). (cited on pages 33, 35, and 53)

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A., 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*. (cited on page 73)

Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J.; et al., 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. (cited on pages 7 and 9)

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9, 8 (1997), 1735–1780. (cited on pages 7, 8, and 9)

Hou, X.; Shen, L.; Sun, K.; and Qiu, G., 2017. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1133–1141. IEEE. (cited on page 21)

Hu, Q.; Szabó, A.; Portenier, T.; Favaro, P.; and Zwicker, M., 2018. Disentangling factors of variation by mixing them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3399–3407. (cited on page 73)

Hu, Q.; Szabó, A.; Portenier, T.; Zwicker, M.; and Favaro, P., 2017. Disentangling factors of variation by mixing them. *arXiv preprint arXiv:1711.07410*, (2017). (cited on pages 77, 79, and 80)

Huang, D.-A.; Fei-Fei, L.; and Niebles, J. C., 2016. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, 137–153. Springer. (cited on pages 37 and 56)

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A., 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134. (cited on pages 1, 73, 75, 96, and 98)

Ji, S.; Xu, W.; Yang, M.; and Yu, K., 2013. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35, 1 (2013), 221–231. (cited on page 36)

Jiang, Y.-G.; Liu, J.; Roshan Zamir, A.; Toderici, G.; Laptev, I.; Shah, M.; and Sukthankar, R., 2014. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/. (cited on pages 1, 53, and 64)

Johnson, J.; Alahi, A.; and Fei-Fei, L., 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 694–711. Springer. (cited on page 84)

Karaman, S.; Seidenari, L.; and Del Bimbo, A., 2014. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, vol. 1, 5. (cited on page 53)

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. (cited on pages 25, 33, 34, 35, and 53)

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al., 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, (2017). (cited on page 53)

Kim, T.; Cha, M.; Kim, H.; Lee, J. K.; and Kim, J., 2017. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, (2017). (cited on pages 74, 75, and 76)

Kingma, D. P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014). (cited on pages 87 and 93)

Kingma, D. P. and Welling, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013). (cited on pages 18, 21, and 73)

Klaser, A.; Marszałek, M.; and Schmid, C., 2008. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 275–1. British Machine Vision Association. (cited on page 35)

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 pages 15 and 56)

Koppula, H. S.; Gupta, R.; and Saxena, A., 2013. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32, 8 (2013), 951–970. (cited on pages 26 and 33)

Koppula, H. S. and Saxena, A., 2016. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38, 1 (2016), 14–29. (cited on page 33)

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105. (cited on page 41)

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T., 2011. HMDB: a large video database for human motion recognition. In *ICCV*. (cited on page 25)

Laptev, I., 2005. On space-time interest points. *IJCV*, 64, 2-3 (2005), 107–123. (cited on page 35)

Lea, C.; Flynn, M. D.; Vidal, R.; Reiter, A.; and Hager, G. D., 2017. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 156–165. (cited on page 7)

Lee, H.-Y.; Tseng, H.-Y.; Huang, J.-B.; Singh, M.; and Yang, M.-H., 2018. Diverse image-to-image translation via disentangled representations. *arXiv preprint arXiv:1808.00948*, (2018). (cited on pages 74, 75, and 76)

Li, K. and Malik, J., 2016. Learning to optimize. *arXiv preprint arXiv:1606.01885*, (2016). (cited on pages 54 and 55)

Li, W.; Zhang, Z.; and Liu, Z., 2010a. Action recognition based on a bag of 3d points. In *CVPR Workshop*, 9–14. IEEE. (cited on page 26)

Li, W.; Zhang, Z.; and Liu, Z., 2010b. Action recognition based on a bag of 3d points. In *CVPR Workshops*, 9–14. IEEE. (cited on page 35)

Lillo, I.; Carlos Niebles, J.; and Soto, A., 2016. A hierarchical pose-based approach to complex action understanding using dictionaries of actionlets and motion poselets. In *CVPR*, 1981–1990. (cited on pages 33, 36, 53, and 56)

Lin, T.; Zhao, X.; and Shou, Z., 2016. Uts at activitynet 2016. *ActivityNet Large Scale Activity Recognition Challenge 2016*, (2016). (cited on page 66)

Lin, T.; Zhao, X.; and Shou, Z., 2017. Temporal convolution based action proposal: Submission to activitynet 2017. *arXiv preprint arXiv:1707.06750*, (2017). (cited on page 66)

Liu, J.; Wang, G.; Hu, P.; Duan, L.-Y.; and Kot, A. C., 2017a. Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1647–1656. (cited on page 11)

Liu, M.-Y.; Breuel, T.; and Kautz, J., 2017b. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, 700–708. (cited on pages 74, 75, 76, and 79)

Liu, M.-Y. and Tuzel, O., 2016. Coupled generative adversarial networks. In *Advances in neural information processing systems*, 469–477. (cited on pages 74 and 76)

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C., 2016a. Ssd: Single shot multibox detector. In *European conference on computer vision*, 21–37. Springer. (cited on pages 37 and 102)

Liu, Y.; Wang, Z.; Jin, H.; and Wassell, I., 2018. Multi-task adversarial network for disentangled feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3743–3751. (cited on page 76)

Liu, Z.; Luo, P.; Qiu, S.; Wang, X.; and Tang, X., 2016b. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 82, 87, and 92)

Long, J.; Shelhamer, E.; and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440. (cited on page 102)

Lu, J.; Xiong, C.; Parikh, D.; and Socher, R., 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 375–383. (cited on page 11)

Lu, J.; Yang, J.; Batra, D.; and Parikh, D., 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, 289–297. (cited on page 11)

Lucey, P.; Cohn, J. F.; Kanade, T.; Saragih, J.; Ambadar, Z.; and Matthews, I., 2010. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, 94–101. IEEE. (cited on page 87)

Ma, C.-Y.; Chen, M.-H.; Kira, Z.; and AlRegib, G., 2017a. Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *arXiv preprint arXiv:1703.10667*, (2017). (cited on page 70)

Ma, L.; Jia, X.; Georgoulis, S.; Tuytelaars, T.; and Van Gool, L., 2018a. Exemplar guided unsupervised image-to-image translation. *arXiv preprint arXiv:1805.11145*, (2018). (cited on page 76)

Ma, L.; Jia, X.; Sun, Q.; Schiele, B.; Tuytelaars, T.; and Van Gool, L., 2017b. Pose guided person image generation. In *Advances in Neural Information Processing Systems*, 406–416. (cited on pages 73, 74, 96, and 98)

Ma, L.; Sun, Q.; Georgoulis, S.; Van Gool, L.; Schiele, B.; and Fritz, M., 2018b. Disentangled person image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 99–108. (cited on pages 74, 76, 89, 90, 91, 95, and 98)

Ma, S.; Sigal, L.; and Sclaroff, S., 2016. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 1942–1950. (cited on pages 33, 36, 44, 45, 46, 53, 56, and 64)

Mao, X.; Li, Q.; Xie, H.; Lau, R. Y.; Wang, Z.; and Paul Smolley, S., 2017. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802. (cited on pages 20 and 85)

Mathieu, M. F.; Zhao, J. J.; Zhao, J.; Ramesh, A.; Sprechmann, P.; and LeCun, Y., 2016. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, 5040–5048. (cited on page 76)

Mikolov, T.; Joulin, A.; Chopra, S.; Mathieu, M.; and Ranzato, M., 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, (2014). (cited on page 9)

Mirza, M. and Osindero, S., 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, (2014). (cited on pages 73 and 75)

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P., 2017a. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2, 7 (2017). (cited on page 15)

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P., 2017b. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, (2017). (cited on pages 54 and 55)

Mo, S.; Cho, M.; and Shin, J., 2018. Instagan: Instance-aware image-to-image translation. (2018). (cited on pages 74, 75, and 76)

Ni, B.; Paramathayalan, V. R.; and Moulin, P., 2014. Multiple granularity analysis for fine-grained action detection. In *CVPR*, 756–763. (cited on pages 36 and 48)

Ni, B.; Pei, Y.; Moulin, P.; and Yan, S., 2013a. Multilevel depth and image fusion for human activity detection. *IEEE transactions on cybernetics*, 43, 5 (2013), 1383–1394. (cited on pages 33 and 36)

Ni, B.; Wang, G.; and Moulin, P., 2013b. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *Consumer Depth Cameras for Computer Vision*, 193–208. Springer. (cited on page 26)

Ni, B.; Yang, X.; and Gao, S., 2016. Progressively parsing interactional objects for fine grained action detection. In *CVPR*, 1020–1028. (cited on pages 33, 36, 48, 53, and 56)

Oord, A. v. d.; Kalchbrenner, N.; and Kavukcuoglu, K., 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, (2016). (cited on pages 8 and 73)

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 33, 35, and 38)

Papoutsakis, K.; Panagiotakis, C.; and Argyros, A. A., 2017. Temporal action co-segmentation in 3d motion capture data and videos. (2017). (cited on page 37)

Peng, X.; Yu, X.; Sohn, K.; Metaxas, D. N.; and Chandraker, M., 2017. Reconstruction-based disentanglement for pose-invariant face recognition. *intervals*, 20 (2017), 12. (cited on page 76)

Perarnau, G.; van de Weijer, J.; Raducanu, B.; and Álvarez, J. M., 2016. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, (2016). (cited on page 75)

Poppe, R., 2010. A survey on vision-based human action recognition. *Image and vision computing*, 28, 6 (2010), 976–990. (cited on page 53)

Pumarola, A.; Agudo, A.; Martinez, A. M.; Sanfeliu, A.; and Moreno-Noguer, F., 2018a. Ganimation: Anatomically-aware facial animation from a single image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 818–833. (cited on pages 74, 75, 76, 79, 88, 90, and 91)

Pumarola, A.; Agudo, A.; Sanfeliu, A.; and Moreno-Noguer, F., 2018b. Unsupervised person image synthesis in arbitrary poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8620–8628. (cited on pages 74, 76, 84, 95, and 98)

Radford, A.; Metz, L.; and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, (2015). (cited on page 73)

Raj, A.; Sangkloy, P.; Chang, H.; Lu, J.; Ceylan, D.; and Hays, J., 2018. Swapnet: Garment transfer in single view images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 666–682. (cited on pages 74 and 76)

Ravi, S. and Larochelle, H., 2017. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, (2017). (cited on pages 54 and 55)

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. (cited on pages 37 and 102)

Roberts, A.; Engel, J.; and Eck, D., 2017. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*. (cited on page 21)

Ronneberger, O.; Fischer, P.; and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer. (cited on page 79)

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X., 2016. Improved techniques for training gans. In *Advances in neural information processing systems*, 2234–2242. (cited on pages 86 and 91)

Salimans, T.; Karpathy, A.; Chen, X.; and Kingma, D. P., 2017. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, (2017). (cited on page 8)

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T., 2016a. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, (2016). (cited on page 15)

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T., 2016b. One-shot learning with memory-augmented neural networks. *International Conference on Machine Learning*, (2016). (cited on pages 54, 55, and 62)

Schroff, F.; Kalenichenko, D.; and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 815–823. (cited on page 2)

Shahroudy, A.; Liu, J.; Ng, T.-T.; and Wang, G., 2016. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. *arXiv preprint arXiv:1604.02808*, (2016). (cited on pages 26, 33, and 36)

Sharma, S.; Kiros, R.; and Salakhutdinov, R., 2015. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, (2015). (cited on page 11)

Shimada, A.; Kondo, K.; Deguchi, D.; Morin, G.; and Stern, H., 2013. Kitchen scene context based gesture recognition: A contest in icpr2012. In *Advances in depth image analysis and applications*, 168–185. Springer. (cited on pages 26, 35, 43, and 48)

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*, 56, 1 (2013), 116–124. (cited on page 35)

Shou, Z.; Chan, J.; Zareian, A.; Miyazawa, K.; and Chang, S.-F., 2017. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. (2017). (cited on pages 33, 36, 53, 56, and 66)

Shou, Z.; Wang, D.; and Chang, S.-F. Temporal action localization in untrimmed videos via multi-stage cnns. (cited on pages 30, 33, 37, 40, 53, 56, and 66)

Shyam, P.; Gupta, S.; and Dukkipati, A., 2017. Attentive recurrent comparators. *International Conference on Machine Learning*, (2017). (cited on page 56)

Siarohin, A.; Sangineto, E.; Lathuilière, S.; and Sebe, N., 2018. Deformable gans for pose-based human image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3408–3416. (cited on pages 73, 96, and 98)

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al., 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529, 7587 (2016), 484. (cited on page 1)

Simonyan, K. and Zisserman, A., 2014a. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 568–576. (cited on pages 13, 33, 35, 39, 58, 68, and 70)

Simonyan, K. and Zisserman, A., 2014b. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, (2014). (cited on pages 1 and 85)

Singh, B.; Marks, T. K.; Jones, M.; Tuzel, O.; and Shao, M., 2016. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1961–1970. (cited on pages 33, 36, 44, 45, 46, 53, and 56)

Snell, J.; Swersky, K.; and Zemel, R., 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 4077–4087. (cited on pages 15 and 56)

Soomro, K.; Roshan Zamir, A.; and Shah, M., 2012a. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*. (cited on pages 1, 53, and 64)

Soomro, K. and Shah, M., 2017. Unsupervised action discovery and localization in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 696–705. (cited on page 56)

Soomro, K.; Zamir, A. R.; and Shah, M., 2012b. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, (2012). (cited on page 25)

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al., 2015. End-to-end memory networks. In *Advances in neural information processing systems*, 2440–2448. (cited on page 15)

Sung, J.; Ponce, C.; Selman, B.; and Saxena, A., 2011. Human activity detection from rgbd images. In *Workshops at the twenty-fifth AAAI conference on artificial intelligence*. (cited on pages 3 and 26)

Sutskever, I.; Martens, J.; and Hinton, G. E., 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1017–1024. (cited on page 7)

Taha, A.; Zayed, H. H.; Khalifa, M.; and El-Horbaty, E.-S. M., 2015. Skeleton-based human activity recognition for video surveillance. *International Journal of Scientific & Engineering Research*, (2015). (cited on page 35)

Taigman, Y.; Polyak, A.; and Wolf, L., 2016. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, (2016). (cited on pages 74 and 76)

Tian, Y.; Cao, L.; Liu, Z.; and Zhang, Z., 2012. Hierarchical filtered motion for action recognition in crowded videos. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42, 3 (2012), 313–323. (cited on page 26)

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M., 2015a. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 4489–4497. (cited on pages 12, 33, and 36)

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M., 2015b. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 4489–4497. (cited on page 56)

Tulyakov, S.; Liu, M.-Y.; Yang, X.; and Kautz, J., 2017. Mocogan: Decomposing motion and content for video generation. *arXiv preprint arXiv:1707.04993*, (2017). (cited on page 76)

Turing, A. M., 2009. Computing machinery and intelligence. In *Parsing the Turing Test*, 23–65. Springer. (cited on page 1)

Veeriah, V.; Zhuang, N.; and Qi, G.-J., 2015. Differential recurrent neural networks for action recognition. In *ICCV*, 4041–4049. (cited on page 35)

Villegas, R.; Yang, J.; Hong, S.; Lin, X.; and Lee, H., 2017a. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, (2017). (cited on page 76)

Villegas, R.; Yang, J.; Zou, Y.; Sohn, S.; Lin, X.; and Lee, H., 2017b. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, (2017). (cited on pages 73, 74, and 81)

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al., 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, 3630–3638. (cited on pages 15, 54, 56, 58, 59, and 62)

Wang, B.; Zheng, H.; Liang, X.; Chen, Y.; Lin, L.; and Yang, M., 2018. Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 589–604. (cited on pages 74 and 76)

Wang, H.; Kläser, A.; Schmid, C.; and Liu, C.-L., 2013. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103, 1 (2013), 60–79. (cited on page 35)

Wang, H. and Schmid, C., 2013. Action recognition with improved trajectories. In *ICCV*, 3551–3558. (cited on pages 33 and 35)

Wang, J.; Cherian, A.; and Porikli, F., 2017a. Ordered pooling of optical flow sequences for action recognition. *arXiv preprint arXiv:1701.03246*, (2017). (cited on page 36)

Wang, J.; Liu, Z.; Chorowski, J.; Chen, Z.; and Wu, Y., 2012a. Robust 3d action recognition with random occupancy patterns. In *ECCV*, 872–885. Springer. (cited on pages 26, 33, and 35)

Wang, J.; Liu, Z.; Wu, Y.; and Yuan, J., 2012b. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 1290–1297. IEEE. (cited on pages 26 and 35)

Wang, K.; Wang, X.; Lin, L.; Wang, M.; and Zuo, W., 2014a. 3d human activity recognition with reconfigurable convolutional neural networks. In *Proceedings of the 22nd ACM international conference on Multimedia*, 97–106. ACM. (cited on page 26)

Wang, L.; Qiao, Y.; and Tang, X., 2014b. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 1, 2 (2014), 2. (cited on page 53)

Wang, L.; Xiong, Y.; Lin, D.; and Van Gool, L., 2017b. Untrimmednets for weakly supervised action recognition and detection. *arXiv preprint arXiv:1703.03329*, (2017). (cited on page 56)

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L., 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, 20–36. Springer. (cited on pages 58, 68, and 70)

Wang, T.-C.; Liu, M.-Y.; Zhu, J.-Y.; Tao, A.; Kautz, J.; and Catanzaro, B., 2017c. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv preprint arXiv:1711.11585*, (2017). (cited on page 75)

Wang, Y.; Long, M.; Wang, J.; and Yu, P. S., 2017d. Spatiotemporal pyramid network for video action recognition. (2017). (cited on page 33)

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P.; et al., 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13, 4 (2004), 600–612. (cited on pages 86 and 91)

Werbos, P. J. et al., 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78, 10 (1990), 1550–1560. (cited on page 9)

Weston, J.; Chopra, S.; and Bordes, A., 2014. Memory networks. *arXiv preprint arXiv:1410.3916*, (2014). (cited on page 15)

Xiong, Y.; Zhao, Y.; Wang, L.; Lin, D.; and Tang, X., 2017. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716*, (2017). (cited on pages 53, 56, and 66)

Xu, H.; Das, A.; and Saenko, K., 2017. R-c3d: Region convolutional 3d network for temporal activity detection. *arXiv preprint arXiv:1703.07814*, (2017). (cited on pages 53, 56, and 66)

Xu, H. and Saenko, K., 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, 451–466. Springer. (cited on page 11)

Yang, H.; He, X.; and Porikli, F., 2018. Instance-aware detailed action labeling in videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1577–1586. IEEE. (cited on page 30)

Yang, X.; Zhang, C.; and Tian, Y., 2012. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM international conference on Multimedia*, 1057–1060. ACM. (cited on page 35)

Yeung, S.; Russakovsky, O.; Jin, N.; Andriluka, M.; Mori, G.; and Fei-Fei, L., 2015a. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, (2015). (cited on pages 33, 36, 44, 45, and 46)

Yeung, S.; Russakovsky, O.; Mori, G.; and Fei-Fei, L., 2015b. End-to-end learning of action detection from frame glimpses in videos. *arXiv preprint arXiv:1511.06984*, (2015). (cited on pages 33 and 36)

Yeung, S.; Russakovsky, O.; Mori, G.; and Fei-Fei, L., 2016. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2678–2687. (cited on pages 53, 56, and 66)

Yi, Z.; Zhang, H. R.; Tan, P.; and Gong, M., 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2868–2876. (cited on pages 74 and 76)

Yim, J.; Jung, H.; Yoo, B.; Choi, C.; Park, D.; and Kim, J., 2015. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 676–684. (cited on page 76)

You, Q.; Jin, H.; Wang, Z.; Fang, C.; and Luo, J., 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4651–4659. (cited on page 11)

Yu, G.; Liu, Z.; and Yuan, J., 2014. Discriminative orderlet mining for real-time recognition of human-object interaction. In *ACCV*, 50–65. Springer. (cited on page 25)

Yu, G. and Yuan, J., 2015. Fast action proposals for human action detection and search. In *CVPR*, 1302–1311. (cited on page 36)

Yuan, J.; Liu, Z.; and Wu, Y., 2009. Discriminative subvolume search for efficient action detection. In *CVPR*, 2442–2449. IEEE. (cited on page 26)

Yuan, Z.; Stroud, J. C.; Lu, T.; and Deng, J., 2017. Temporal action localization by structured maximal sums. (2017). (cited on pages 33, 36, 53, 56, and 66)

Yue-Hei Ng, J.; Hausknecht, M.; Vijayanarasimhan, S.; Vinyals, O.; Monga, R.; and Toderici, G., 2015. Beyond short snippets: Deep networks for video classification. In *CVPR*, 4694–4702. (cited on page 35)

Zeng, R.; Huang, W.; Tan, M.; Rong, Y.; Zhao, P.; Huang, J.; and Gan, C., 2019. Graph convolutional networks for temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 7094–7103. (cited on page 30)

Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; and Metaxas, D., 2017. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1710.10916*, (2017). (cited on page 73)

Zhao, Y.; Xiong, Y.; Wang, L.; Wu, Z.; Lin, D.; and Tang, X., 2017. Temporal action detection with structured segment networks. *arXiv preprint arXiv:1704.06228*, (2017). (cited on pages 53 and 56)

Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Wang, J.; and Tian, Q., 2015. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*. (cited on pages 82 and 92)

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A., 2017a. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2223–2232. (cited on pages 1, 74, 75, 76, and 79)

Zhu, J.-Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A. A.; Wang, O.; and Shechtman, E., 2017b. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, 465–476. (cited on page 75)