From Line Drawings to Human Actions: Deep Neural Networks for Visual Data Representation

Fang Wang

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

November 2017

Declaration

Except where otherwise indicated, this thesis is my own original work.

Fang Wang
7 November 2017

This research is supported by an
Australian Research Training Program (RTP)
Scholarship.

Acknowledgments

I would like to express my deepest gratitude to my supervisors Dr. Yi Li and Prof. Fatih Porikli. They provided me great supports and encouraged me to continue my research when things became hard. Their passions about research and vision of the future directions in computer vision inspired me deeply. It has been my great pleasure to work with them in the past few years. I am grateful for their patient supervision on research, careful guidance on shaping immature ideas, and late night editing on papers. Furthermore, I would like to thank Dr. Justin Domke for providing valuable suggestions along the journey towards my Ph.D. degree.

Further, I would like to express my sincere appreciation to Prof. Cornelia Fermüller and Prof. Yiannis Aloimonos. They gave me great support and invaluable advice during my visiting in the Computer Vision Laboratory at the University of Maryland. They have always been supportive and kind to me in the past few years. I have been fascinated by their pioneering ideas of the research in computer vision and robotics. I was very fortunate to have the opportunity to work with them. Their insightful discussions about the research area kept me motivated during my study.

I would like to thank Le Kang, Aiwen Jiang, Yezhou Yang, Konstantinos Zampogiannis, Yi Zhang, Francisco Barranco, Michael Pfeiffer, and Chinmaya Devaraj, who have collaborated with me and contributed amazing efforts in our studies. They showed me how to effectively shape research ideas and execute pioneer research. I thank my colleagues and friends at the Australian National University, who gave me valuable support and help during my study. My thanks go to Hanxi Li, Lin Gu, Tianxing Li, Dylan Campbell, Buyu Liu, Wei Zhuo, Haoyang Zhang, and Gao Zhu. I would also like to thank my friends at the University of Maryland, Ren Mao, Tao Wen, Fangyi Zhang, Kanishka Ganguly, Gregory Kramida, and Aleksandrs Ecins. Thank you for giving me so much wonderful time during my visiting and I have learned a lot from you all.

Finally, I would like to show my deepest thanks to my family. Thank you for your love and trust. You have encouraged me all the time and always been my strongest backing. Without you, I could never be where I am today.

Abstract

In recent years, deep neural networks have been very successful in computer vision, speech recognition, and artificial intelligent systems. The rapid growth of data and fast increasing computational tools provide solid foundations for the applications which rely on the learning of large scale deep neural networks with millions of parameters. The deep learning approaches have been proved to be able to learn powerful representations of the inputs in various tasks, such as image classification, object recognition, and scene understanding. This thesis demonstrates the generality and capacity of deep learning approaches through a series of case studies including image matching and human activity understanding. In these studies, I explore the combinations of the neural network models with existing machine learning techniques and extend the deep learning approach for each task. Four related tasks are investigated: 1) image matching through similarity learning; 2) human action prediction; 3) finger force estimation in manipulation actions; and 4) bimodal learning for human action understanding.

Deep neural networks have been shown to be very efficient in supervised learning. Further, in some tasks, one would like to group the features of the samples in the same category close to each other, in additional to the discriminative representation. Such kind of properties is desired in a number of applications, such as semantic retrieval, image quality measurement, and social network analysis, etc. My first study is to develop a similarity learning method based on deep neural networks for image matching between sketch images and 3D models. In this task, I propose to use Siamese network to learn similarities of sketches and develop a novel method for sketch based 3D shape retrieval. The proposed method can successfully learn the representations of sketch images as well as the similarities, then the 3D shape retrieval problem can be solved with off-the-shelf nearest neighbor methods.

After studying the representation learning methods for static inputs, my focus turns to learning the representations of sequential data. To be specific, I focus on manipulation actions, because they are widely used in the daily life and play important parts in the human-robot collaboration system. Deep neural networks have been shown to be powerful to represent short video clips [Donahue et al., 2015]. However, most existing methods consider the action recognition problem as a classification

task. These methods assume the inputs are pre-segmented videos and the outputs are category labels. In the scenarios such as the human-robot collaboration system, the ability to predict the ongoing human actions at an early stage is highly important. I first attempt to address this issue with a fast manipulation action prediction method. Then I build the action prediction model based on Long Short-Term Memory (LSTM) architecture. The proposed approach processes the sequential inputs as continuous signals and keeps updating the prediction of the intended action based on the learned action representations.

Further, I study the relationships between visual inputs and the physical information, such as finger forces, that involved in the manipulation actions. This is motivated by recent studies in cognitive science which show that the subject's intention is strongly related to the hand movements during an action execution. Human observers can interpret other's actions in terms of movements and forces, which can be used to repeat the observed actions. If a robot system has the ability to estimate the force feedbacks, it can learn how to manipulate an object by watching human demonstrations. In this work, the finger forces are estimated by only watching the movement of hands. A modified LSTM model is used to regress the finger forces from video frames. To facilitate this study, a specially designed sensor glove has been used to collect data of finger forces, and a new dataset has been collected to provide synchronized streams of videos and finger forces.

Last, I investigate the usefulness of physical information in human action recognition, which is an application of bimodal learning, where both the vision inputs and the additional information are used to learn the action representation. My study demonstrates that, by combining additional information with the vision inputs, the accuracy of human action recognition can be improved steadily. I extend the LSTM architecture to accept both video frames and sensor data as bimodal inputs to predict the action. A hallucination network is jointly trained to approximate the representations of the additional inputs. During the testing stage, the hallucination network generates approximated representations that used for classification. In this way, the proposed method does not rely on the additional inputs for testing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last five years, we witness the rapid development and significant success of deep learning methods in both academia and industry. The idea of automatically learning of features has been shown to be more efficient than the traditional feature engineering methods with domain knowledge. Some notable applications include image classification and natural language processing. It may be interesting to raise the question about what applications, in additional to these mainstream ones, can benefit from deep learning. This thesis is an attempt to answer this question with deep analysis and case studies of applications. The studies in this thesis focus on the deep learning methods and their variations to solve computer vision tasks that traditionally have been considered difficult. The research work in this thesis is two folded. One is the learning of static inputs such as images. The other one is the learning of dynamic inputs with temporal information such as videos.

The performance of existing computer vision algorithms depends heavily on the representations or features of the given inputs. Traditional learning algorithms rely on manually designed features for specific data in different tasks, such as human speech, language, handwritten texts, and natural images. This procedure is called feature engineering, which requires expert knowledge of the given inputs as well as a certain understanding of the learning algorithms to design a good feature. Such procedures are very inefficient because they usually do not generalize well across domains. Based on these handcrafted features, researchers have proposed many learning algorithms that are proved to be successful for computer vision, for example, naive Bayes [Russell and Norvig, 2003], logistic regression [Cox, 1958] and support vector machines [Cortes and Vapnik, 1995], etc. However, most of these methods treat the features as fixed inputs or perform simple preprocessings to the input features. One step towards the feature learning is the sparse coding method, which aims at finding the representation of input data by learning the linear combination of a

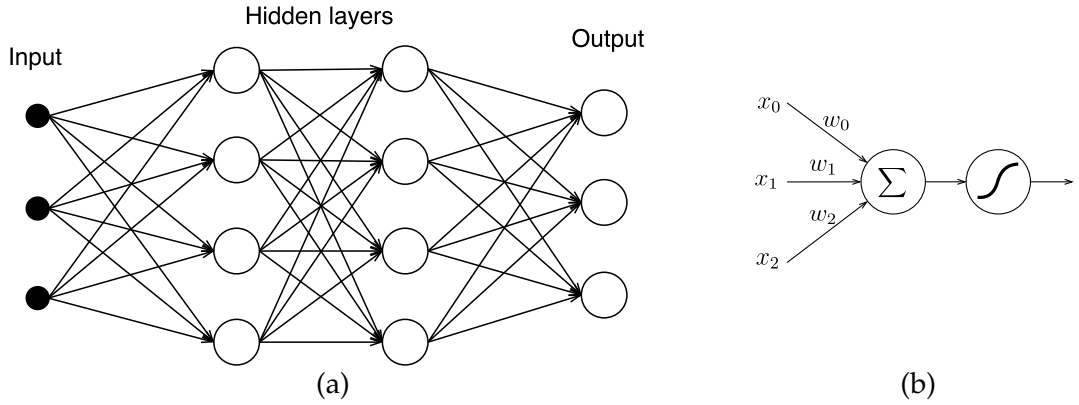


Figure 1.1: An illustration of the artificial neural network structure and the activation function. (a) In an artificial neural network, a number of hidden layers with neural units are used to connect the input and the output layers. (b) The activation function takes the weighted summation of the inputs and generates the output response with a non-linear function.

set of basic feature elements [Olshausen and Field, 1996; Donoho and Elad, 2003]. In this way, the model can learn a sparse representation for each input. However, the basic elements are still manually defined and the model can only learn simple combinations of the basis with a shallow structure.

Motivated by this, the idea of learning more powerful feature representations with deep structures is proposed to alleviate the problems for handcrafted features. The artificial neural network [Holyoak, 1987] is one of the earliest attempt to automatically learn features with multiple layers. As an analog of the neural system in the human brain, an artificial neural network typically consists of a large number of neural units that are connected and joined in one network (See Figure 1.1a). Each of these neural units simulates the axons in biological brains, which takes multiple input signals and generate an output response. For each unit, an activation function is used to decide how the output response is calculated from the inputs. Typically, the activation function is a non-linear function over the weighted summation of the inputs (See Figure 1.1b). With all the neural units connected into a neural network, one can train the network with backward propagations [Rumelhart et al., 1988]. The weights of all neural units are tuned at the same time and learned to activate to the specific patterns in the inputs. In such neural networks, the intermediate units, also called hidden units, are trained to learn the representations of the given inputs.

To represent highly complex and redundant inputs such as natural images, a large scale artificial neural network with millions of units is necessary. However, training of such large scale networks requires plenty of computational resources and labeled training samples, which limits their usage.

In recent years, the rapid accumulating of data makes it possible to collect large scale datasets with millions of training samples. Further, the emergence of fast parallel computing devices enables the development of learning algorithms that have high computational cost. Both these new developments make the training of large scale neural network possible. These large scale neural networks are called deep neural networks since they usually consist of many connected neural layers and form a deep structure. The deep neural networks are derived from the traditional artificial neural networks. Different from the other popular machine learning methods that have shallow structures, deep neural networks typically have more layers and parameters, thus they have the potential to represent more complex inputs. These complex networks allow people to build intelligent agents that are capable of complicated perception tasks and even reach the performance of human in some cases.

In the following sections of this chapter, Section 1.1 introduces the basic concepts and commonly used tools in deep learning, and discusses the typical neural network structures for different tasks. Section 1.2 discusses the deep learning approaches for static input learning. It briefly introduces the popular network structures used for discriminative learning as well as similarity learning tasks. Section 1.3 discusses about deep learning methods for sequential data. This section introduces the network structures that can handle sequential inputs. These networks have temporal recursive loops and are suitable for modeling the dynamics in sequential data. In this thesis, I use these network architectures to learn representations of human actions and briefly introduce three related tasks about manipulation action recognition. At last, the outline of the whole thesis is given in Section 1.4 and a list of publications is presented in Section 1.5.

1.1 Deep neural networks

A deep neural network is an artificial neural network that consists of multiple layers of hidden units to extract features of given inputs. A non-linear activation function is applied to each hidden unit to generate the output. The activation functions add non-linearity to the model so that the deep neural networks is able to model complex non-

linear relationships. The hidden units are connected with all or partial of the units in the previous layer and feed the outputs forward to the next layer. With the stacked structure of many hidden layers, the deep neural network is able to learn multiple levels of feature representations that correspond to different levels of abstraction. Analysis of the weights in each layer shows that the earlier layers can extract the lower level patterns from the inputs, and the latter layers tend to learn high-level features by combining the lower level patterns [Zeiler and Fergus, 2014]. With such structures, the deep neural networks are able to extract complex representations.

A deep neural network can be trained using standard backpropagation method. Rumelhart et al. [1988] showed that it is possible to train the hidden units in a neural network to represent the important features of the task domain using backpropagation method. For each training input, the backpropagation algorithm first calculates the response of each unit in the network using the forward procedure. Next, the outputs are compared with the ground-truth labels to calculate the error or loss. Then this error can be propagated back to all units and get the adjust value of the unit parameters. The standard backpropagation method for training deep neural networks is the stochastic gradient descent (SGD) method. It converges much faster than the batch gradient descent method. In practice, an alternation of the stochastic gradient descent method is often used which is call "mini-batch" gradient descent. Other than calculating the gradient updates for each training sample, the "mini-batch" method uses multiple training samples at each step. This is a good compromise between the standard batch gradient descent and the global gradient descent method. One benefit is that with multiple training samples at each step, one can utilize the parallel computing tools to accelerate the training process. Another benefit is that using "mini-batch" makes the training process more stable than using individual inputs because it uses multiple training samples for gradient updates in each step.

1.2 Learning of static inputs

Deep learning methods have shown to be very effective in supervised learning. For vision based recognition tasks, the convolution neural network (CNN) [LeCun and Bengio, 1998] is one of the most frequently used network structures. It stacks multiple convolutional layers, each of which conducts convolution operations on its input and generates a set of response maps. These response maps are then used as inputs of the next layer. The usage of convolution operation makes it very suitable for

processing grid-like data such as speech signals (1D grid) and images (2D grids). To automatically learn features from image inputs, an important ability is to be robust to deformations and image noise. CNN models use max-pooling layers to learn robust features and handle deformations. With these techniques, the CNNs have shown to be effective in both vision and speech tasks. For example, in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [Russakovsky et al., 2015], over one million images are used to train a 1000-way classifier. In this scenario, convolution neural networks achieved the best performance on both classification and detection tasks, and the best method reported less than 3% classification errors over 1000 categories [ILSVRC, 2016], which is competitive with the performance of human recognition.

Although the CNN model is able to learn powerful representations for discriminative learning, in some tasks we would like to learn the representation that more similar objects have smaller distances. Such kind of properties is desired in a number of applications, for example, semantic retrieval, image quality measurement, and social network analysis. In computer vision, the Siamese network has been adopted for face verification [Chopra et al., 2005] and digit recognition [Yih et al., 2011]. It uses a shared deep neural network to learn features from pairs of samples instead of process individual samples. Then the extracted features are compared with a contrastive loss function, which allows the Siamese network to learn the feature representations as well as the similarities for pairs of data.

In this work, the Siamese network is studied and applied to solve the sketch based 3D shape retrieval problem. Existing methods for 3D shape retrieval depend on handcrafted features and extensive searching for the best viewpoint to match the 3D models and the 2D sketches. The proposed method using learned shape representations for shape matching, which is shown to be superior to traditional shape features. We show that the learned features using deep neural networks are good for cross-domain similarity learning as well as simple sketch matching.

1.3 Learning of sequential data

All these neural network structures mentioned before are designed for static inputs. In vision tasks, there is a large family of problems requiring feature learning for sequential data. There are a lot of practical applications in recognition in videos, such as human activity recognition, scene understanding, and object in the wild,

etc. To model sequential data, the Recurrent Neural Network (RNN) is the most frequently used network architecture. RNN contains a recursive loop that unfolds over time, which makes it suitable for modeling non-linear time series with long time correlations. RNN has been widely used in natural languages processing as well as video sequences modeling. However, the traditional RNN is hard to train using gradient descent method because of the vanishing gradient problem. To tackle this problem, the Long Short-Term Memory (LSTM) model is then proposed. It introduces memory cells with gates control, which makes the LSTM architecture more robust when learning temporal dynamics with long time constants.

This thesis focuses on the human action recognition problem to explore the representation learning method of sequential data. Human action recognition is one of the fundamental problems in computer vision. Due to its high complexity, many research works try to simplify the visual inputs into a set of key points and only focus on the trajectories of these key points to model the dynamics [Laptev, 2005; Rohrbach et al., 2012; Wang et al., 2011]. Other researchers aim at finding spatial-temporal representations that are suitable for action recognition [Dollár et al., 2005; Laptev and Lindeberg, 2006; Klaser et al., 2008; Scovanner et al., 2007]. With all these efforts, it is still challenging to learn a good representation of human actions and model the dynamics. In this thesis, I study the effectiveness of the recurrent neural network architecture in human action recognition. The goal is to understand how to effectively learn a good representation of human actions using neural networks. To be specific, three aspects of the action recognition problem has been investigated. 1) I propose to predict the intended actions as a continuous task instead of classifying the actions for segmented video clips. 2) I propose to estimate the physical motoric information such as finger forces by watching the hand movements. 3) I explore the bimodal learning for action recognition to utilize the physical information to improve the recognition performance.

1.3.1 Predicting actions

Different from traditional approaches that treat action recognition as a classification problem, we study the action prediction problem which aims at predicting human actions by watching the early stage of hand and body movements. This is useful for cognitive robots in the human-robot collaboration scenario, where the robots need to understand their partner's intended actions before the action is completely finished to prepare the correct response for real time collaborations.

In this thesis, I focus on hand manipulation actions that are often involved in human-robot collaboration tasks. For a robotic system, it is essential to have the ability to update the beliefs about the observed actions and predict the intended actions of their human counterpart. An efficient action prediction method needs well-learned action representations that are able to discriminate the subtle differences of the early hand movements. Therefore, I proposed to build an action prediction model based on the LSTM architecture. The proposed approach processes the sequential inputs as continuous signals and generates confidence map over all candidate action categories for each frame. With the recurrent loops in the LSTM architecture, the history information is implicitly encoded in the model, so that the prediction for each frame is based on both the latest input as well as the previous visual inputs. In this way, the prediction model is able to update the confidence of action categories along with time changes.

This is a challenging task since we are aiming at distinguishing the intended action by watching the dexterous movements of hands. The convolutional neural networks are known to represent image frames, and the recurrent neural networks are designed to learn the dynamic changes of sequential inputs. So we combined these two models in our proposed method to make use the representative power of both these architectures. With this combination, our method is able to capture the delicate differences of the dynamics of the manipulation actions.

1.3.2 Predicting hand forces

Recent studies in cognitive science show that the subject's intention is strongly related to his/her movements during an action execution. Also, the physical contacts between hands and the objects can be altered with different attentions. For example, when a subject intends to use a hammer to pound a nail, it is highly likely that he/she will grasp with more forces than just to passing the tool. Human observers can easily interpret other's actions in terms of movements and forces, and predict the consequences driven by these patterns. If a robot system can estimate the force patterns from videos, it will be able to learn how to manipulate an object by watching video demonstrations.

We focus on the estimation of the forces on the fingertips by regression. Since the lower level layers that are close to the input layer are similar to the action prediction model, we also build the regression model based on the LSTM architecture. The LSTM model is formulated for regression to estimate the force values from video

frames. To train this model, videos of manipulation actions need to be collected with synchronized streams of force measurements on the hand. This was done with a specially designed sensor glove that attached force sensors to the fingertips.

1.3.3 Learning with bimodal information

With the action prediction model and the hand forces regressor, it is natural to ask how to combine the additional information with vision inputs to improve action prediction performance. In many real applications, the additional information, such as accelerometer data, hand forces, gazing points, needs to be gathered with special devices. This limits the use of bimodal learning approaches. To alleviate this problem, I explore a bimodal learning method that can learn the relationship between the two input streams and use the learned representation to improve manipulation action recognition for samples that do not have the additional information.

Motivated by the success of the hand force regression model, a natural idea is to make use of the hand forces to help with the action recognition. One simple extension is to directly use the regression model and generate estimations of hand forces and train the recognition model on both vision input and the forces. However, such framework fails to model the relationship between the hand forces and the vision inputs, which is crucially important for a bimodal learning model. To model the relationship between videos and forces, a possible solution is to learn a hallucination network to bridge the two inputs. During the training stage, the hallucination network dedicates to approximate the learned representation of the forces using vision inputs, so that it can learn the relationship between video frames and hand forces. During the testing stage, the hallucination network can generate a simulated representation of the forces. The whole network is trained end-to-end to jointly learn the action recognition model as well as the hallucination network.

1.4 Thesis outline

The remain of this thesis is organized as follows:

Chapter 2: Background and Related Works This chapter introduces the basic concepts of deep neural networks and the tools used in deep learning approaches. It reviews the literature of deep learning methods and the key issues in applying neural networks for computer vision problems.

Chapter 3: Sketch Based 3D Shape Retrieval This chapter describes a case study of representation learning for sketches to solve the problem of sketch based 3D shape retrieval. A similarity learning method for matching sketch images using Siamese network is proposed. This work was published in CVPR 2015.

Chapter 4: Manipulation Action Prediction Using LSTM To study the representation learning methods of sequential data, we focus on human manipulation action recognition tasks. In this chapter, I show how to use the RNN and LSTM models to learn representations of manipulation actions and proposed a novel approach to predict intended actions while watching the manipulation movements. This work was published in IJCV, Special Issue on Chalearn: Looking at people, 2017.

Chapter 5: Hand Force Prediction Using LSTM This chapter demonstrates how to estimate hand forces from vision input. The LSTM model is adopted for regression of the hand force. I show that the estimated forces can be used to improve the action recognition performance. A new dataset is collected with synchronized sequences of hand forces and action videos to evaluate the proposed method.

Chapter 6: Manipulation Action Recognition Using Bimodal Inputs This chapter expands the manipulation action recognition method to incorporate with additional information such as hand forces or motoric data of the tools or objects that involved in the actions. I show how to train a hallucination network for sequential inputs, and generate hallucinated representations of the additional information for action recognition tasks. This work was submitted to ICCV, 2017.

Chapter 7: Conclusion Finally, we summarize the main contributions in this thesis and discuss the future directions.

1.5 Publications

During my study in NICTA/ANU, I have published the following papers:

Journal publications:

1. Aiwen Jiang, Fang Wang, Fatih Porikli, and Yi Li. "Compositional memory for visual question answering." (Submitted to T-CSVT 2017)

2. Cornelia Fermüller, Fang Wang, Yezhou Yang, Konstantinos Zampogiannis, Yi Zhang, Francisco Barranco, and Michael Pfeiffer. "Prediction of Manipulation Actions." IJCV, Special Issue on Chalearn: Looking at people, 2017.

Conference publications:

1. Chinmaya Devaraj, Fang Wang, Yi Li, Cornelia Fermüller, and Yiannis Aloimonos. "Leveraging Motoric Information for Recognizing Manipulation Actions" (Submitted to ICCV 2017)

2. Fang Wang, Le Kang, and Yi Li. "Sketch-based 3d shape retrieval using convolutional neural networks." CVPR 2015.

3. Fang Wang and Yi Li. "Spatial matching of sketches without point correspondence." ICIP 2015.

4. Fang Wang and Yi Li. "Beyond physical connections: Tree models in human pose estimation." CVPR 2013.

5. Fang Wang and Yi Li. "Learning visual symbols for parsing human poses in images." IJCAI 2013.

Chapter 2

Background and Related Work

In recent years, deep learning has achieved great success in speech processing and computer vision tasks. Unlike the traditional methods that try to manually design features for specific inputs, deep learning provides a powerful tool to learn representations directly from inputs. The idea of learning features automatically from inputs is attractive because it relieves the need of expert knowledge to extract useful features for specific tasks. Recent studies show that the data driven feature engineering methods often have better performance than handcrafted features. This is possibly because the automatically learned features can capture more discriminative features.

In this chapter, we introduce the basic concepts of deep learning approaches and briefly discuss the key techniques that are used for training of deep neural networks. Section 2.1 revisits the basic concepts of deep neural networks and introduces a frequently used network structure in representation learning approaches. In Section 2.2, several special network structures are introduced as the backgrounds of our case studies. Section 2.2.1 introduces the Siamese network for matching problem. Section 2.2.2 introduces the recent models for sequential data learning.

2.1 Deep neural network structures

Deep neural networks are a series of neural network structures that consist of many neural layers. With a deep structure, a neural network can effectively learn complicated mappings from the inputs to the target using end-to-end training, which does not rely on domain knowledge. Several techniques have been devised to overcome the difficulties in the training of complex neural networks.

2.1.1 Convolution neural network

A convolutional neural network (CNN) is a neural network that is suitable for processing images. As indicated by its name, the core component in CNN is the convolutional layer, which applies the "convolution" operation on the layer inputs to filter out useful information. The second important component is the pooling layer, which downsamples the inputs using a given selection method. The last is the fully connected layer, which combines all the outputs of the previous layer and generates the feature representations. With a deep structure, a CNN can effectively learn complicated mappings from raw images to the target, which requires less domain knowledge compared to handcrafted features and shallow learning frameworks.

A typical CNN structure consists of several convolutional layers that have a small field of perception, and a few fully connected layers that each combines all the outputs from the last layer and yields a feature vector with several thousands of dimensions. To efficiently train the CNN model, the standard method is using Stochastic Gradient descent (SGD) with backpropagation algorithm. Figure 2.1 shows the structure of AlexNet, which is one of the early CNN models.

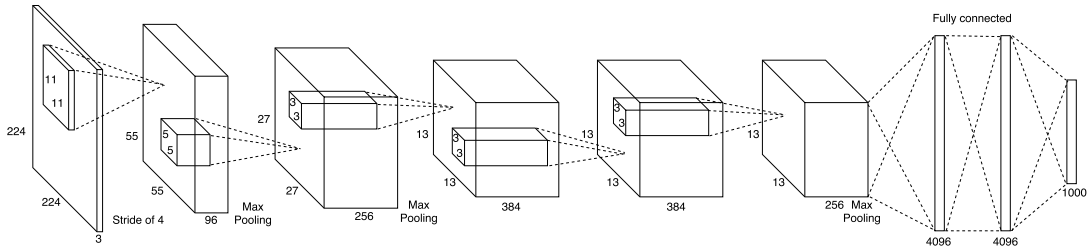


Figure 2.1: A typical convolutional neural network structure (Redraw of the AlexNet structure [Krizhevsky et al., 2012]).

The CNN structure is originally designed for classification tasks, thus the last layer of the network works as a classifier based on the learned features from the immediate precedent layer. One advantage of these networks is that one can easily adapt a pre-trained neural network to new tasks where only small number of training samples are available. This technique is called finetuning. By removing the classifier layer of a neural network and attaching a new layer with randomly initialized parameters, one can train these new parameters efficiently and achieve good performance on the new task. Because the pre-trained model has already learned

comprehensive representations through millions of training samples, the finetuning procedure can start from exploring useful representations for the new task without going all the way from scratch.

2.1.2 Recurrent Neural Network

The Recurrent Neural Network (RNN) is a popular architecture that has been widely used to processing sequential inputs, such as natural language and video frames. The RNN model has a temporal recurrent loop that can capture compositional representations in the time domain and is suitable for modeling the dynamic of sequential inputs. The temporal recurrent loop creates a deep structure in the RNN model when unfolding in the time series. Figure 2.2 shows the structure of RNN and the unfolded network.

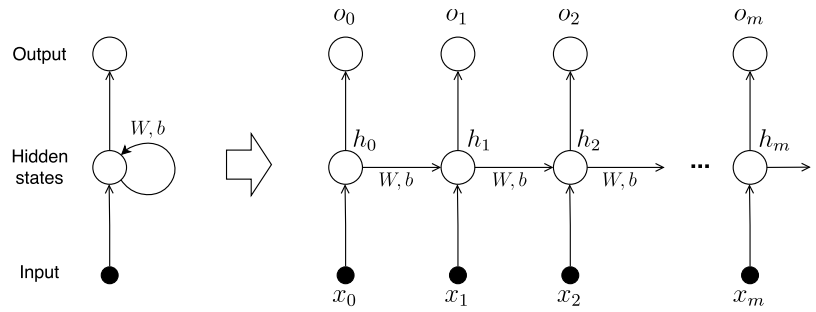


Figure 2.2: An Recurrent Neural Network and the unfolded structure.

Given a sequence $x = \{x_1, x_2, \ldots, x_T\}$, an RNN computes a sequence of hidden states $h = \{h_1, h_2, \ldots, h_T\}$ and outputs $y = \{y_1, y_2, \ldots, y_T\}$ as follows:

$$h_t = \mathcal{H}(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \tag{2.1}$$

$$y_t = \mathcal{O}(W_{ho}h_t + b_o), \tag{2.2}$$

where W_{ih}, W_{hh}, W_{ho} denote weight matrices, b_h, b_o denote the biases, and $\mathcal{H}(\cdot)$ and $\mathcal{O}(\cdot)$ are the activation functions of the hidden layer and the output layer, respectively. Typically, the activation functions are defined as logistic sigmoid functions.

The traditional RNN is hard to train due to the so-called vanishing gradient problem. If the input sequence is too long, the gradient update through backpropagation becomes inefficient. The weight updates decrease exponentially with the number

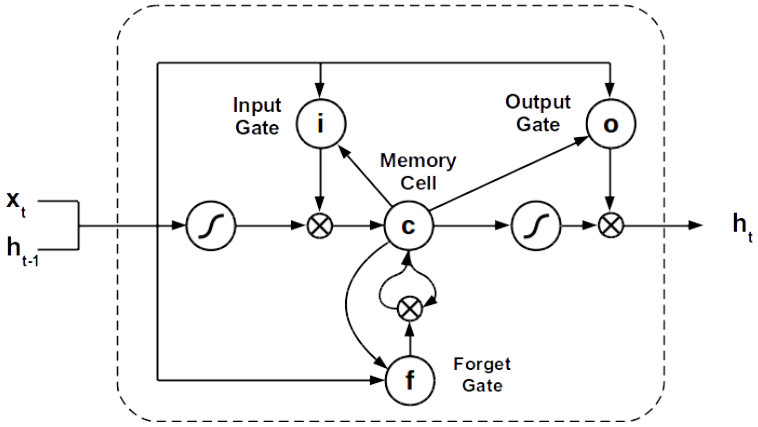


Figure 2.3: A diagram of a LSTM memory cell (adapted from Graves et al. [2013]).

of backpropagation steps, which makes the training extremely slow. This problem limits the maximum length of sequences that an RNN can accept.

To alleviate the vanishing gradient problem, the Long Short-Term Memory (LSTM) model is then proposed by Hochreiter and Schmidhuber [1997]. Specifically, in addition to the hidden state h_t, the LSTM also includes an input gate i_t, a forget gate f_t, an output gate o_t, and the memory cell c_t (shown in Figure 2.3). These gates are regularized with sigmoid functions and control the portion of information passed through the update functions. To be specific, in this architecture i_t and f_t are sigmoidal gating functions, and these two terms learn to control the portions of the current input and the previous memory that the LSTM takes into consideration for overwriting the previous state. Meanwhile, the output gate o_t controls how much of the memory should be transferred to the hidden state. These mechanisms allow LSTM networks to learn temporal dynamics with long time constants.

The hidden layer and the additional gates and cells are updated as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.4}$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.6}$$

$$h_t = o_t \tanh(c_t) \tag{2.7}$$

The LSTM model has been used for image description generation [Donahue et al., 2015], translating videos to language [Venugopalan et al., 2014], object recognition

[Visin et al., 2014], and the estimation of object motion [Fragkiadaki et al., 2015]. Ng et al. [2015] used the LSTM model to learn dynamic changes for action recognition.

Although the LSTM is the most commonly used structure, other variations have been proposed to improve the performance. Cho et al. [2014] proposed a simplified version of LSTM with fewer gate controls, named Gated Recurrent Unit (GRU). It coupled the input and the forget gates into an update gate and replaced the output gate with a reset gate that only controls the recurrent connections to the block input. To improve the capability of long-term memory learning, Weston et al. [2014] proposed the memory networks. It introduced a long-term memory component which can learn flexible combinations of history information. This architecture provides a mechanism for selective learning of long sequences.

2.1.3 Training neural networks

Deeper structures bring many challenges for the training such as overfitting and slow convergence problems. To effectively learn the representations of inputs, nonlinearities play an important part in building a deep neural network. Inspired by the biology, rectifying functions are used to control the outputs followed by normalizations. Nair and Hinton [2010] proposed the Rectified Linear Units (ReLUs) function as the activation function, while Krizhevsky et al. [2012] showed that it also improved the convergence speed of the training procedure. For normalization, a number of different methods have been studied. Jarrett et al. [2009] used local contrast normalization after each layer inspired by computational neuroscience model [DiCarlo et al., 2008; Lyu and Simoncelli, 2008]. The local contrast normalization enforces local competition between adjacent features in the feature map. Krizhevsky et al. [2012] used brightness normalization, which is similar to the local contrast normalization but without the subtraction of the mean activity. Another difficulty in training a deep neural network is overfitting. Hinton et al. [2012] proposed to use dropout regularization during the training stage. This is an efficient way to perform model average and has been proved to be very effective that it has already become a standard method in deep neural network training. Recent works on very deep neural networks reveal that the network depth is crucial important [Simonyan and Zisserman, 2014b; Szegedy et al., 2015; He et al., 2015, 2016]. However, learning a network with dozens or hundreds of layers is extremely challenging. This problem is largely addressed with the batch normalization [Ioffe and Szegedy, 2015] and introducing residual units into the network [He et al., 2016].

Many machine-learning algorithms use variants of the standard gradient descent method for optimization. However, for the training of deep neural networks, the global gradient descent method is inefficient since calculating the sum-gradient of a large training set is extremely expensive. Many research works have been dedicated to creating efficient training methods for deep neural networks. One of the most frequently used optimization methods to train deep neural networks is the Stochastic gradient descent (SGD) method [Murata, 1998; Bousquet and Bottou, 2008]. This is a random optimization method that approximate the global gradient descent optimization by accumulating gradient updates over individual samples. In practice, the gradient updates are often calculated over a small number of samples, which is also called a "mini-batch", to utilize the parallel computational mechanisms. It also makes the convergence more smooth by using more training samples at each step. Research works show that the stochastic gradient descent converges faster than global gradient descent method. When the training set has a large number of samples, using global gradient descent is practically impossible. In such cases, the stochastic gradient descent is the best choice for the optimization.

Although stochastic gradient descent is widely used, the training may not be efficient enough in some cases. A number of adaptive updating algorithms have been proposed to improve the stability or convergence speed of the optimization. To achieve more robust update of gradients, the RMSProp method keeps a moving average of the squared gradients and use this value to normalize the gradient update steps [Hinton, 2012]. Duchi et al. [2011] proposed the adaptive gradient algorithm to improve the training performance over sparse inputs. The basic idea is to use parameter-wise learning rate instead of one global learning rate for all parameters. The adaptive gradient method gives larger update step length for more sparse parameters and reduces the update step length for less sparse parameters so that the training speed can be adjusted according to the geometry of the observed data. ADADELTA is one of the most frequently used updating algorithms for its simplicity and low overhead cost [Zeiler, 2012], which requires no manual tuning of learning rates and is robust to noisy gradients and different network structures. Another effective method is Adaptive Moment Estimation (Adam) proposed by Kingma and Ba [2014]. It considered both the first-order gradients and the lower-order moments, which are very robust to highly noisy and sparse gradients. Generally, these adaptive optimization methods have better performance compared to the standard stochastic gradient descent, with the cost of more or less additional computation overheads.

2.2 Applications: case studies of deep learning

In this thesis, I use deep learning approaches for two categories of computer vision tasks: 1) similarity learning of static inputs; 2) the representation learning of sequential data. This section briefly discusses the backgrounds and closely related works of the deep learning applications in the case studies.

2.2.1 Learning similarity

In this thesis, I select Siamese network as the similarity learning tool and use it to solve the sketch based 3D shape retrieval problem. A typical Siamese network structure is shown in Figure 2.4. The Siamese network has been initially proposed for signature verification [Bromley et al., 1993]. It is designed to learn the relationship of an input pair. The Siamese network consists of a shared sub-network that learn the representations of sample pairs. In different variations of the Siamese network, the shared sub-networks may be used with various network layers at different levels, then the extracted pair of features is joined at the last layers with a contrastive loss function. Different from the other single modal network structures, it learns the representation of inputs with constraints of the pairwise relationship of samples. The goal of the network is to minimize the feature distance if the input pair is labeled as similar, and push the features far away from each other for dissimilar pairs. Such kind of network structure can be used as supervised metric learning model.

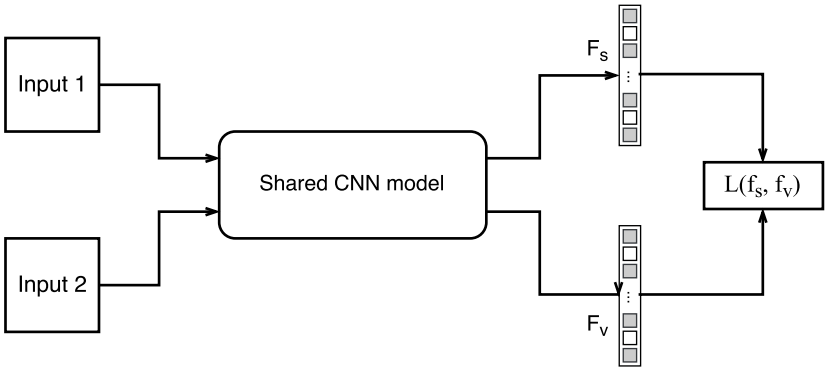


Figure 2.4: A typical Siamese network structure.

Chopra et al. [2005] use the Siamese network for face verification. They propose to use the convolutional network to map the raw input to the target space to

achieve robustness to geometric distortions. Salakhutdinov and Hinton [2007] use multilayer neural networks with Siamese network architecture to learn non-linear embedding of high dimensional inputs. Recently, the Siamese network has been successfully applied to semantic text similarity measurements [Yih et al., 2011]. The authors show that Siamese network achieves high accuracy on cross-lingual document retrieval as well as ad relevance measure tasks, which demonstrate the ability of Siamese network to learn cross domain similarities. Chen and Salman [2011] applied Siamese network for speech feature classification task and successfully learn the speaker-specific information. In recent year, the Siamese network has been adopted in more vision tasks for image matching. Zagoruyko and Komodakis [2015] extended and generalized the Siamese network to learning similarity between image patches. Bertinetto et al. [2016] further extended the Siamese network with fully convolutional networks for object tracking. Follow the line of multi-stream structures, Wang et al. [2014] proposed the triplet network for image rank learning to learn fine-grained image similarity.

2.2.2 Learning representations of sequential data

This section focuses on the deep learning approaches for sequential data modeling. Recently the RNN and the LSTM architectures have been widely used for sequential data learning. They are popularized in language processing and have been used for generating sentences that describe images [Karpathy and Fei-Fei, 2015; Vinyals et al., 2015]. In these works, the recurrent networks are adopted to process the language data to either learn the language model or generate sentences using visual features. Instead of using recurrent networks for feature encoding, Vinyals et al. [2015] proposed to use the convolutional neural network to encode the image feature and decode it with an LSTM model. Karpathy and Fei-Fei [2015] further extended the decoding module with a bidirectional recurrent network and an attention model to generate sentences.

Following these studies, a number of works proposed to extend the recurrent networks to the image domain to encode video frames and generate video descriptions [Venugopalan et al., 2014; Donahue et al., 2015]. Donahue et al. [2015] introduced a generalized structure called Long-term Recurrent Convolutional Networks (LRCN), which combines recurrent networks and convolutional neural networks for sequential data modeling. Under this architecture, the recurrent networks can be used for action recognition by encoding image features and generate action categories. [Ng

et al., 2015] extended the LSTM model with temporal pooling over frame features and ordered sequence of frames to learn action representations.

In my study, specifically, I explore the representation learning approaches for human action recognition problem. Most traditional methods consider the action recognition problem as a classification task. Given a pre-segmented input video, the action recognition methods extract features of the video clips and generate a set of category labels as the output. In a very recent work, Ma et al. [2016] trained an LSTM using novel ranking losses for early activity detection.

2.3 Summary

In this chapter, we revisited the basic concepts and recent developments of deep learning methods and summarized the advantages and drawbacks of these new approaches. We described the convolutional neural network model for static representation learning as well as the Siamese network for similarity learning. For sequential data modeling, we described the Recurrent Neural Network and the Long Short-Term Memory model for feature learning. These research findings are the basis of the works proposed in this thesis. Many of them provide deep insights of the research area and motivations of new ideas.

Chapter 3

Sketch Based 3D Shape Retrieval

In this chapter, my study is to learn sketch representations and similarities for shape matching. The 3D shape retrieval is an important research topic in content based object retrieval. It has important applications in computer graphics, information retrieval, and computer vision [Eitz et al., 2012b; Furuya and Ohbuchi, 2013; Li et al., 2014a]. Early studies of 3D shape retrieval directly use 3D shapes as queries and focus on creating proper feature descriptions that are suitable for 3D shape matching [Shilane et al., 2004]. Such methods suffer from the difficulty of 3D shape orientation alignment and high computational cost. Using 3D shapes as queries also limited the application of these methods. In contrast, a more intuitive approach is retrieving 3D shapes using hand drawn sketches. The idea of sketch based 3D shape retrieval is very attractive. It provides a user friendly way to create query inputs and visually depictive to specify shape.

Directly matching 2D sketches to 3D models suffers from significant differences between the 2D and 3D representations. Many existing methods project the 3D models to multiple 2D views and do the sketch matching in the 2D space. Figure 3.1 shows a few examples of 2D sketches and their corresponding 3D models. One can immediately see the variations in both the sketch styles and 3D models. In almost all state of the art approaches, sketch based 3D shape retrieval amounts to finding the "best views" for 3D models and handcrafting the right features for matching sketches and views. First, an automatic procedure is used to select the most representative views of a 3D model. Ideally, one of the viewpoints is similar to that of the query sketches. Then, 3D models are projected to 2D planes using a variety of line rendering algorithms. Subsequently, many 2D matching methods can be used for computing the similarity scores, where features are always manually defined (*e.g.*, dense SIFT [Furuya and Ohbuchi, 2009] and Gabor local line based feature (GALIF)

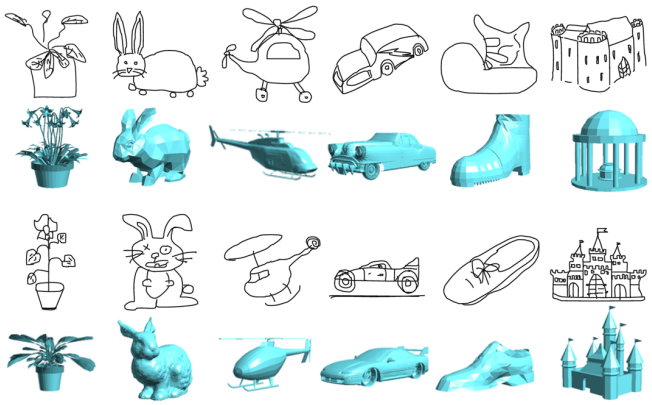


Figure 3.1: Examples of sketch based 3D shape retrieval.

[Eitz et al., 2012b]).

This stage-wise methodology appears pragmatic, but it also brings a number of puzzling issues. To begin with, there is no guarantee that the best views have similar viewpoints with the sketches. The inherent issue is that identifying best view is an unsolved problem by itself, partially because the general definition of best views is elusive. In fact, many best view methods require manually selected viewpoints for training, which makes the view selection by finding "best views" a chicken-and-egg problem. Further, this viewpoint uncertainty makes it dubious to match samples from two different domains without learning their metrics. Take Figure 3.1 for example, even when the viewpoints are similar the variations in sketches as well as the different characteristics between sketches and views are beyond the assumptions of many 2D matching methods.

Considering all the above issues arise when we struggle to seek the viewpoints for matching, can we bypass the stage of view selection? In this work, we demonstrate that by learning cross domain similarities, we no longer require the seemingly indispensable view similarity assumption. Instead of relying on the elusive concept of "best views" and handcrafted features, we propose to define our views and learn features for views and sketches. Assuming that the majority of the models are upright, we drastically reduce the number of views to two for the whole dataset. We also make no selection of these two directions as long as they are significantly different. Therefore, this can be considered as the minimalist approach as opposed to multiple best views. This upright assumption appears to be strong, but it turns out

to be sensible for 3D datasets. Many 3D models are naturally generated upright (*e.g.*, the Princeton Shape Benchmark [Shilane et al., 2004]). We choose two viewpoints because it is very unlikely to get degenerated views for two significantly different viewpoints. An immediate advantage is that our matching is more efficient without the need of comparing to more views than necessary.

This seeming radical approach triumphs only when the features are learned properly. In principle, this can be regarded as learning representations between sketches and views by specifying similarities, which gives us a semantic level matching. To achieve this goal, we need comprehensive shape representations rather than the combination of a bunch of shallow features that only capture the low level visual information. We learn the shape representations using Convolutional Neural Network (CNN). Our model is based on the Siamese network [Chopra et al., 2005]. Since the two input sources have distinctive intrinsic properties, two different CNN models are used, one for handling the sketches and the other for the views. This two model strategy can give us more power to capture different properties in different domains. Most importantly, a loss function is defined to "align" the results of the two CNN models. This loss function couples the two input sources into the same target space, which allows us to compare the features directly using a simple distance function.

The remain of this chapter is organized as follows. In Section 3.1, I first review the closely related works for sketch based shape retrieval and introduce the benchmark datasets. In Section 3.2, I show how to build the cross domain matching model using Siamese network and present the network architecture. Section 3.3 shows the experiment results of our method. I evaluated our method on three datasets. Comparison results show that our method significantly outperforms existing approaches in a number of metrics, including precision-recall and the nearest neighbor. I further demonstrate the retrieval performances within each domain.

The main contributions in this work include

1. I proposed to learn feature representations for sketch based shape retrieval, which bypasses the dilemma of best view selection;

2. I adopted two Siamese Convolutional Neural Networks to successfully learn both the within domain and the cross domain similarities;

3. The proposed method significantly outperformed previous methods on three large datasets.

3.1 Related work

Sketch based shape retrieval has received many interests for years [Funkhouser et al., 2003; Li et al., 2014a,a]. In this section, we review three key components in sketch based shape retrieval: public available datasets, features for shape matching, and similarity learning.

Datasets The effort of building 3D datasets can be traced back to decades ago. The Princeton Shape Benchmark (PSB) is probably one of the best known sources for 3D models [Shilane et al., 2004]. There are some recent advancements for general and special objects, such as the SHREC'13 Benchmark [Li et al., 2014a], the SHREC'14 Benchmark [Li et al., 2014b] and the Bonn Architecture Benchmark [Wessel et al., 2009]. 2D sketches have been adopted as input in many systems [Daras and Axenopoulos, 2010]. However, the large-scale collections are available only recently. Eitz et al. [2012b] collected sketches based on the PSB dataset. Li et al. [2014a] organized the sketches collected by Eitz et al. [2012a] in their SBSR challenge.

Features Global shape descriptors, such as statistics of shapes [Osada et al., 2002] and distance functions [Kazhdan et al., 2002], have been used for 3D shape retrieval [Tangelder and Veltkamp, 2008]. Recently, local features is proposed for partial matching [Funkhouser and Shilane, 2006] or used in the bag-of-words model for 3D shape retrieval [Bronstein et al., 2011].

Boundary information together with internal structures are used for matching sketches against 2D projections. Therefore, a good representation of line drawing images is a key component for sketch based shape retrieval. Sketch representation such as shape context [Belongie et al., 2002] was proposed for image based shape retrieval. Furuya and Ohbuchi [2009] proposed BF-DSIFT feature, which is an extended SIFT feature with Bag-of-word method, to represent sketch images. One recent method is the Gabor local line based feature (GALIF) by Eitz et al. [2012b], which builds on a bank of Gabor filters followed by a Bag-of-word method.

In addition to 2D shape features, some methods also explored geometry features as well as graph-based features to facilitate the 3D shape retrieval [Li et al., 2015]. Semantic labeling is also used to bridge the gaps between different domains [Gong et al., 2013].

3.2 The approach

We propose to learn the representations of 2D sketches using CNN model and jointly learn the similarities using Siamese network architecture. As introduced in Chapter 2, the CNN model is suitable for learning of hierarchical feature representations. We extend the Siamese network architecture with two CNN models for the two different input domains, one for the hand drawn sketches and the other for the projected views of 3D models. These two CNN models have identical network structure but are updated separately. This two model strategy can give us more power to capture different properties in different domains.

Siamese Network combined with convolutional networks has been successfully used for dimension reduction in weakly supervised metric learning [Salakhutdinov and Hinton, 2007]. The most important part of Siamese network is the loss function defined on the pairs of extracted features, which is designed to "align" the learned features from the two domains. A typical loss function of a pair has the following form:

$$L(s_1, s_2, y) = (1-y)\frac{1}{C_p}D_w^2 + yC_n e^{\frac{-2.77}{C_n}D_w}, \tag{3.1}$$

$$D_w = \|f(s_1; w_1) - f(s_2; w_2)\|_1 \tag{3.2}$$

where s_1 and s_2 are two input samples, y is the binary similarity label, $y = 0$ for matched pairs and $y = 1$ for mismatched pairs. D_w is the distance, and C_p and C_n are two constants. In the experiments, we empirically use $C_p = 0.2$ and $C_n = 10.0$, respectively. The loss function consists of two terms. For matched pairs, the loss function penalizes large distances of the projected features; for mismatched pairs, the loss function minimizes the exponential of negative feature distance so that closer feature pairs are penalized. This can be regarded as a metric learning approach. Unlike methods that assign binary similarity labels to pairs, the network aims at bringing the output feature vectors closer for input pairs that are labeled as similar, or push the feature vectors away if the input pairs are labeled as dissimilar. The proposed model is trained with Stochastic Gradient Descent (SGD) method.

The Siamese network is frequently illustrated as two identical networks for two different samples. In each SGD iteration, pairs of samples are processed using two identical networks, and the error computed by Equation (3.1) is then back-propagated and the gradients are computed individually base on the two sample

sets. The Siamese network is updated by the average of these two gradients.

3.2.1 Cross-domain matching using Siamese network

In this section, we propose a method to match samples from two domains without the heavy assumption of view similarity. We first provide our motivation using an illustrated sample. Then, we propose our extension of the basic Siamese network. Specifically, we use two different networks to handle sources from different domains.

An illustrated example The matching problem can be cast as a metric learning paradigm. In each domain, Siamese network effectively maps the line drawings to a feature space, respectively. The cross domain matching can be successful if these two domains are "aligned" correctly.

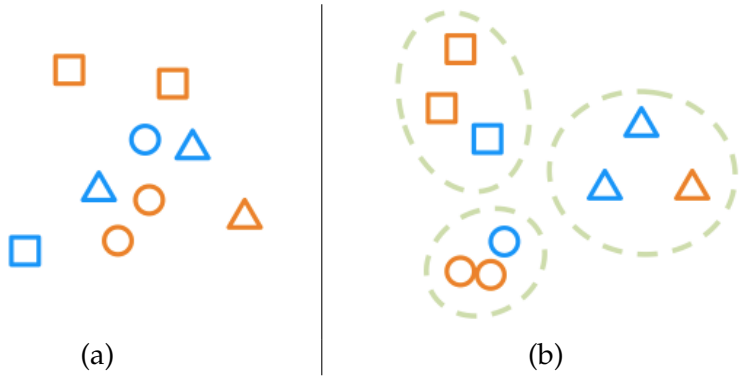


Figure 3.2: An illustrated example, a) the shapes in the original domain may be mixed, and b) after cross-domain metric learning, similar shapes in both domains are grouped together.

This idea is illustrated in Figure 3.2. Blue denotes samples in the sketch domain, and the orange denotes the ones in the view space. Different shapes denote different classes. Before learning, the feature points from two different sources are initially mixed together (Figure 3.2a). If we learn the correct mapping using pair similarities in each domain as well as their cross-domain relations jointly, the two point sets may be correctly aligned in the feature space (Figure 3.2b). After this cross-domain metric learning, matching can be performed in both the same domain (sketch-sketch and view-view) and cross-domain (sketch-view).

Note that, there are no explicit requirements about viewpoint similarity in this

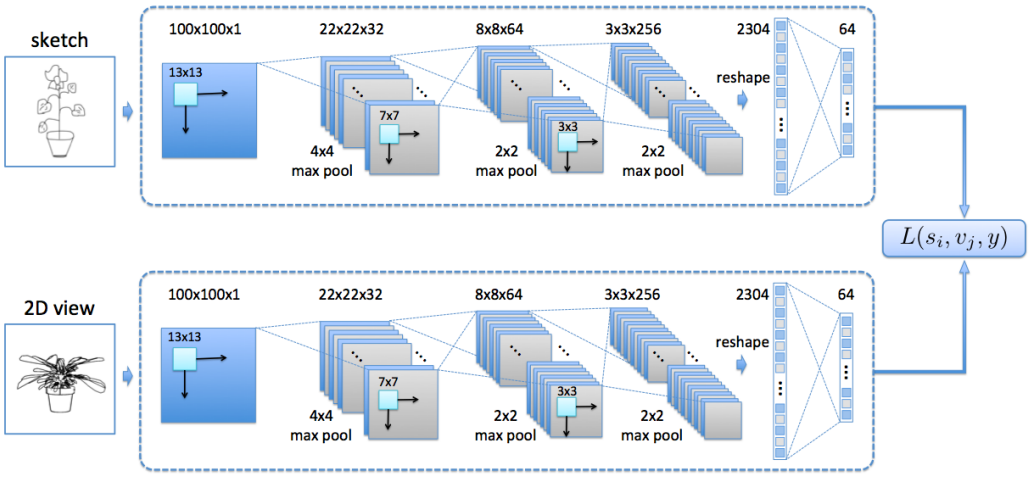


Figure 3.3: Dimension reduction using Siamese network.

perspective. That is, whether the matched pairs are from the same viewpoints is less important. Instead, the focus is the metric between the two domains.

Two networks, one loss The basic Siamese network is commonly used for samples from the same domain. In the cross domain setting, we propose to extend the basic version to two Siamese networks, one for the view domain and the other for the sketch domain. Then, we define the within-domain loss and the cross domain loss. This hypothesis is supported by the experiments in the Section 3.3.

Assuming we have two inputs from each domain, *i.e.*, s_1 and s_2 are two sketches and v_1 and v_2 are two views. For simplicity, we assume s_1 and v_2 are from the same class and s_2 and v_1 are from the same class as well. This can be achieved by restricting the class of sketch/view samples during pair sampling. Therefore, the three pairs are guaranteed to have the same matching type and one label y is enough to specify their relationships.

As a result, our loss function is composed of three terms: the similarity of sketches, the similarity of views, and the cross-domain similarity.

$$\mathcal{L}(s_1, s_2, v_1, v_2, y) = L(s_1, s_2, y) + L(v_1, v_2, y) + L(s_1, v_1, y), \tag{3.3}$$

where $L(\cdot, \cdot, \cdot)$ is defined by Equation (3.1).

3.2.2 Network architecture

Figure 3.3 shows the architecture of our network for the inputs are views and sketches, respectively. We use the same network design for both networks, but they are learned simultaneously. Our input patch size is 100×100 for both sources. The structure of the single CNN has three convolutional layers, each with a max pooling, one fully connected layer to generate the features, and one output layer to compute the cross domain loss.

The first convolutional layer followed by a 4×4 pooling generates 32 response maps, each of size 22×22. The second layer and pooling outputs 64 maps of size 8×8. The third layer has 256 response maps, each is pooled to size 3×3. The 2304 features generated by the final pooling operation are linear transformed to 64×1 features in the last layer. Rectified linear units are used in all layers.

3.2.3 View definitions and line drawing rendering

This section presents our procedure of generating viewpoints and rendering 3D models. As opposed to multiple best views, we find it sufficient to use two random views to characterize a 3D model because the chance that both views are degenerated is very little. Following this observation, we impose the minimal assumptions on selecting views for the whole dataset:

1. Most of the 3D models in the dataset are upright;

2. Two viewpoints are randomly generated for the whole dataset, provided that their angle larger than 45 degree.

Figure 3.4 shows some of our views in the PSB dataset. The first row shows that the upright assumption does not require strict alignments of 3D models, because some models may not have well defined orientation. Further, while the models are upright, they can still have different rotations (second row).

It is worth to stress that this approach does not eliminate the possibility of selecting more (best) views as input, but the comparisons among view selection methods are beyond the scope of this chapter.

Once the viewpoints are chosen, we use two methods to render line drawings. Rendering line drawings that include strong abstraction and stylization effects is a very useful topic in computer graphics, computer vision, and psychology. Outer edges as well as internal edges both play an important role in this rendering process.

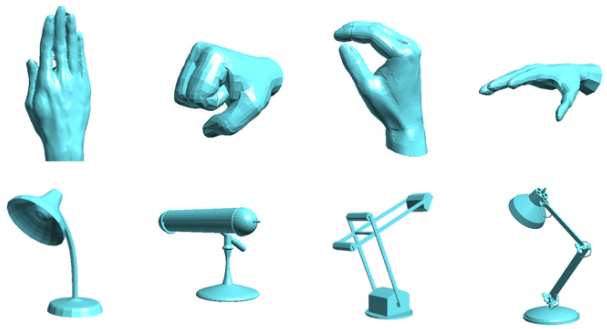


Figure 3.4: 3D models viewed from predefined viewpoints.

Therefore, we use the following descriptors: 1) closed boundaries and 2) Suggestive Contours [DeCarlo et al., 2003] (Figure 3.5).

(a) Shaded (b) SC (c) Final

Figure 3.5: Line rendering of a 3D model. (a) shaded rendering of the 3D model; (b) the suggestive contour; (c) the combination of shaded contour with suggestive contour.

3.3 Experiments

We present our experiments on three recent large datasets in this section. In all experiments, our method outperformed the state of the arts in a number of well recognized metrics. In addition to the cross domain retrieval, we also present our within-domain retrieval results, which have not been reported in any other comparison methods. These experiments demonstrate our siamese network successfully learns the feature representations for both domains.

3.3.1 Datasets

PSB / SBSR dataset The Princeton Shape Benchmark (PSB) [Shilane et al., 2004] is widely used for 3D shape retrieval system evaluation, which contains 1814 3D models and is equally divided into training set and testing set. The 3D models are collected from 161 object classes. Notice that only part of the classes is shared by the training and testing set. To be specific, 21 classes appear in both training and testing set, 69 classes appears only in the training set, and 71 classes are used only for testing. In section 3.3.3, we will evaluate the retrieval performance of the classes exists only in the testing set. As these 3D models are unseen during the training stage, they are more challenging for the similarity learning task.

In [Eitz et al., 2012b], the Shape Based Shape Retrieval (SBSR) dataset is collected for pairing with the PSB 3D models. The 1814 hand drawn sketches were collected using Amazon Mechanical Turk. In the collection process, participants were asked to draw sketches given only the name of the categories, thus the sketches are drawn without any visual clue from the 3D models.

SHREC'13 & '14 dataset Although the PSB dataset is widely used in shape retrieval evaluation, there is a concern that the number of sketches for each class in the SBSR dataset is not enough. Some classes have only very few instances (27 of 90 training classes have no more than 5 instances), while some classes have dominating number of instances, *e.g.*, the "fighter jet" class and the "human" class have as many as 50 instances.

To remove the possible bias when evaluation the retrieval algorithms, Li et al. [2014a] reorganized the PSB/SBSR dataset and proposed the SHREC'13 dataset where a subset of PSB with 1258 models was used and the sketches in each class had 80 instances. These sketch instances were split into two sets: 50 for training and 30 for testing. Please note, the number of models in each class still varies. For example, the largest class have 184 instances but there are 23 classes having no more than 5 models

Recently, SHREC'14 was proposed to address some above concerns [Li et al., 2014b], which greatly enlarged the number of 3D models to 8987, and the number of classes was doubled. The large variation of this dataset made it much more challenging, and the overall performance of all reported methods are very low (*e.g.*, the accuracy of the best algorithm is only 0.16 for the top 1 candidate). This is probably due to the fact that the models are from various sources and arbitrarily oriented.

While our performance is still superior (see Figure 3.11 and Table 3.4), we choose to present our results using the SHREC'13 dataset.

Evaluation criteria In our experiment, we use all datasets and measure the performance using the following criteria: 1) *Precision-recall curve* is calculated for each query and linear interpolated, then the final curve is reported by averaging all precision values for fixed recall rates; 2) *Average precision (mAP)* is the area under the precision-recall curve; 3) *Nearest neighbor (NN)* is used to measure the top 1 retrieval accuracy; 4) *E-Measure (E)* is the harmonic mean of the precision and recall for the top 32 retrieval results; 5) *First/second tier (FT/ST)* and *Discounted cumulated gain (DCG)* as defined in the PSB statistics.

3.3.2 Experimental settings

Stopping criteria All three of the datasets had been split into training and testing sets, but no validation set was specified. Therefore, we terminated our algorithm after 50 epochs for PSB/SBSR and 20 for SHREC'13 dataset (or until convergence). Multiple runs were performed and the mean values were reported.

Generating pairs for Siamese network To make sure we generate the reasonable proportion of similar and dissimilar pairs, we used the following approach to generate pair sets. For each training sketch, we random selected k_p view peers in the same category (matched pairs) and k_n view samples from other categories (unmatched pairs). Usually, our dissimilar pairs are ten times more than the similar pairs for successful training. In our experiment, we use $k_p = 2$, $k_n = 20$. We perform this random pairing for each training epoch.

Computational cost The implementation of the proposed Siamese CNN is based on the Theano library [Theano Development Team, 2016]. We measured the processing time on a PC with 2.8GHz CPU and GTX 780 GPU. With preprocessed view features, the retrieval time for each query is approximately 0.002 sec on average on SHREC'13 dataset. The code is available from http://users.cecs.anu.edu.au/~fwang/sbsr_prj/sbsr_demo.html.

The training time is proportional to the total number of pairs and the number of epochs. Overall training takes approximately 2.5 hours for PSB/SBSR, 6 hours for SHREC'13, respectively.

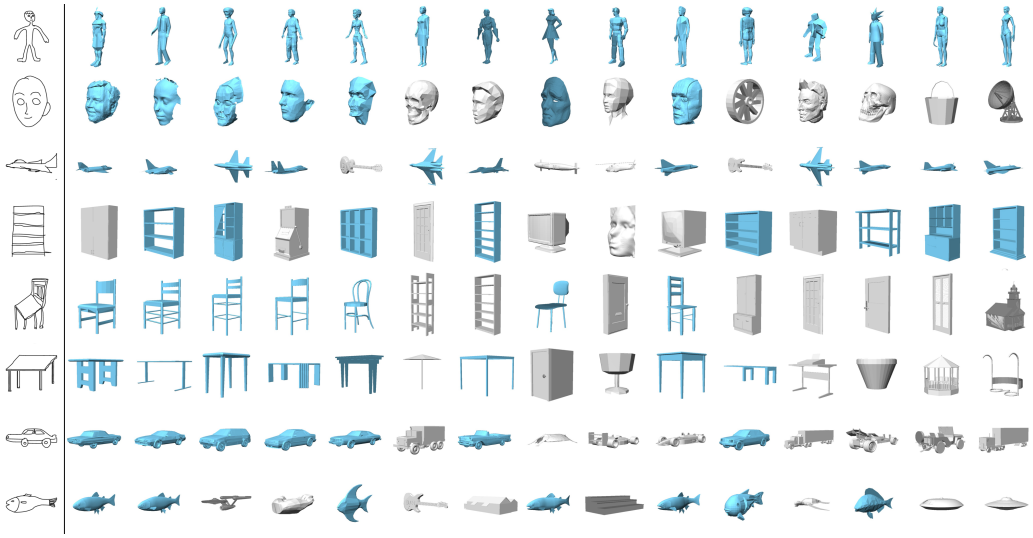


Figure 3.6: Retrieval examples of PSB/SBSR dataset. The cyan denotes the correct retrievals.

3.3.3 Shape retrieval on PSB/SBSR dataset

Examples In this section, we test our method using the PSB/SBSR dataset. First, we show some retrieval examples in Figure 3.6. The first column shows 8 queries from different classes, and each row shows the top 15 retrieval results. The cyan denotes the correct retrievals, and gray denotes incorrect ones.

The method performs exceptionally well in popular classes such as human, face, and plane. We also found that some fine-grained categorizations are difficult to distinguish. For instance, the shelf and the box differ only in a small part of the model. However, we also want to note that some of the classes only differ in semantics (*e.g.*, barn and house only differ in function). Certainly, this semantic ambiguity is beyond the scope of this chapter.

Finally, we want to stress that the importance of viewpoint is significantly decreased in our metric learning approach. Some classes may exhibit a high degree of freedom such as the plane, but the retrieval results are also excellent (as shown in Figure 3.6).

Analysis We further show some statistics on this dataset. First, we provide the precision-recall values at fixed points in Table 3.1. Compared to Figure 9 in [Eitz et al., 2012b], our results are approximately 10% higher. We then show six standard evaluation metrics in Table 3.2.

Since other methods did not report the results on this dataset, we leave the comprehensive comparison to the next section. Instead, in this analysis, we focus on the effectiveness of metric learning for shape retrieval.

Table 3.1: Precision-recall on fixed points.

5%	20%	40%	60%	80%	100%
0.616	0.286	0.221	0.180	0.138	0.072

Table 3.2: Standard metrics on the PSB/SBSR dataset.

NN	FT	ST	E	DCG	mAP
0.223	0.177	0.271	0.173	0.451	0.218

Retrieval for "pure test" classes As we have mentioned before, PSB/SBSR is a very imbalanced dataset, where the training set and the testing set are partially overlapped. This makes it an excellent dataset for investigating similarity learning because the "unseen" classes verify the learning is not biased to the training classes.

In this experiment, we use the same model trained on the original training and testing split, and evaluate the performance on three different test sets. The "pure test" contains test samples from the classes that never appears in training. The "shared test" contains test samples from the classes that also used in training. The "whole test" consists of all test samples.

Some retrieval examples of the unseen classes are shown in Figure 3.7. It is interesting to see that our proposed method works well even on failure cases, such as the flower, the retrieval returns similar shapes ("potting plant"). This demonstrates that our method learns the similarity effectively. Figure 3.8 shows our comparison on the split testing set according to their class label.

3.3.4 Shape retrieval on SHREC'13 and SHREC'14 dataset

In this section, we use the SHREC'13 and SHREC'14 benchmark to evaluate our method. We compare to all other methods evaluated in these two benchmarks and also show the retrieval results within the same domain.

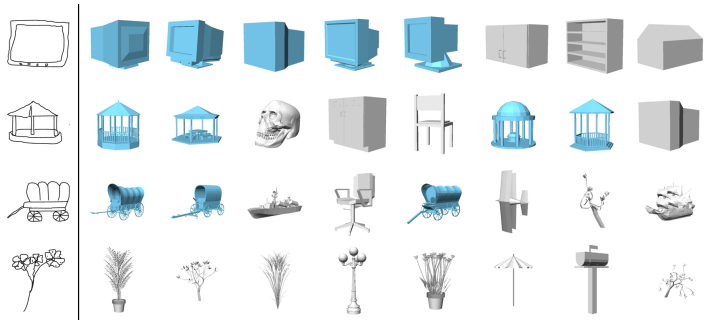


Figure 3.7: Retrieval examples of unseen samples in PSB/SBSR dataset. The cyan denotes the correct retrievals.

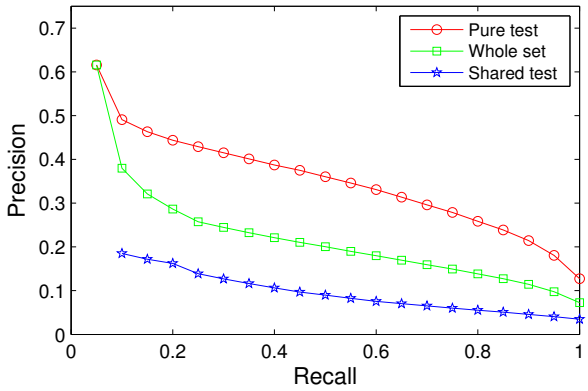


Figure 3.8: Split test set performance on SBSR.

A visualization of the learned features First, a visualization of our learned feature space is presented in Figure 3.9. We perform a principle component analysis on the features learned by our network and reduce the dimension to two for visualization purpose. The green dots denote the sketches, and the yellow ones denote views. For simplicity, we only overlay the sampled views over the point cloud.

While this is a coarse visualization, we can already see some interesting properties of our method. First, we can see that classes with similar shapes are grouped together automatically. On the top right, different animals are mapped to nearby spaces. On the left, various types of vehicles are grouped autonomously. Other examples include house and church, which are very similar.

Note that this is a weakly supervised method, where only the similarity is labeled. This localization suggests that the learned features are effective in capturing the similarities of samples.

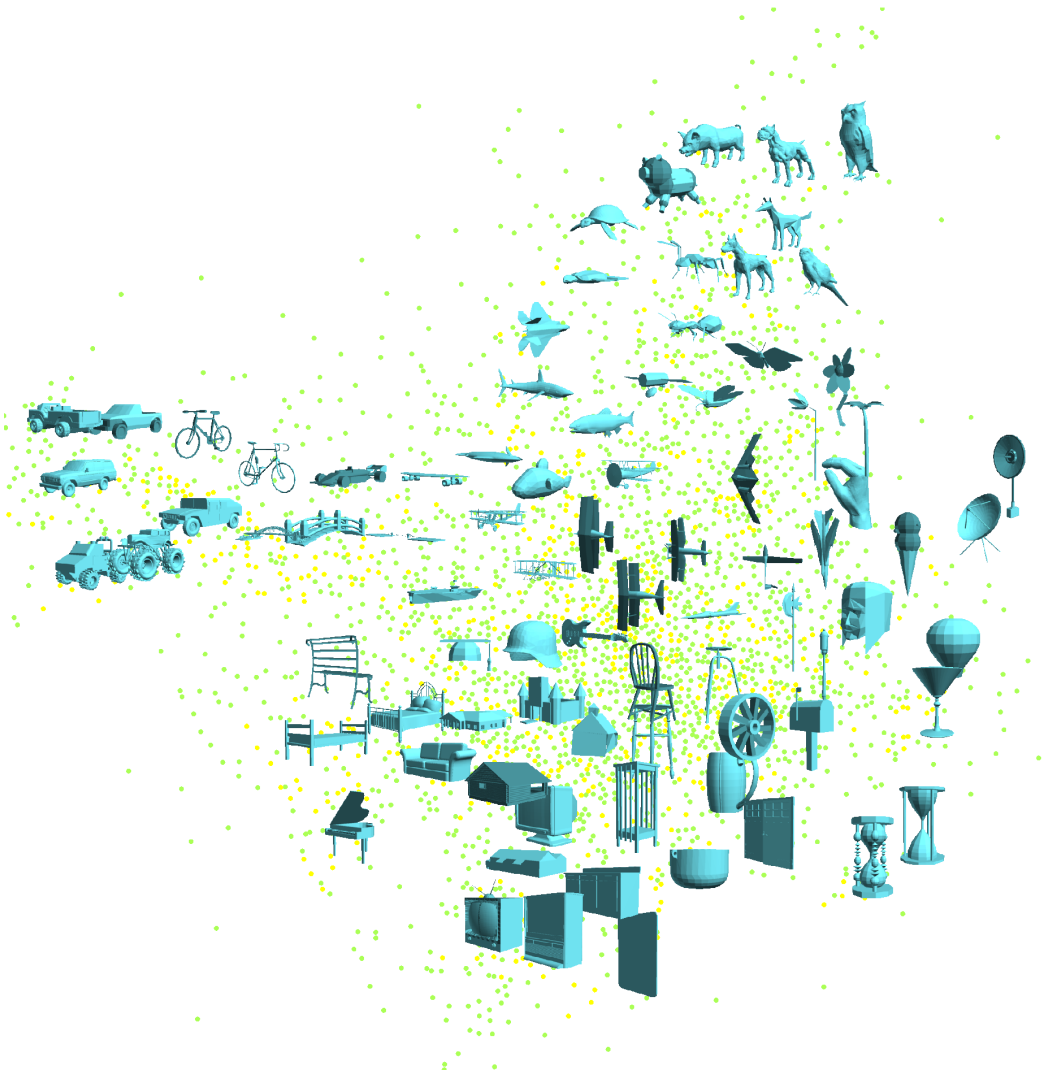


Figure 3.9: Visualization of feature space on SHREC'13. Sketch and view feature points are shown by green & yellow, respectively.

Statistical results This section presents the statistical results on SHREC'13 and SHREC'14. First, we compare the precision-recall curve against the state of the art methods reported in [Li et al., 2014a] and [Li et al., 2014b].

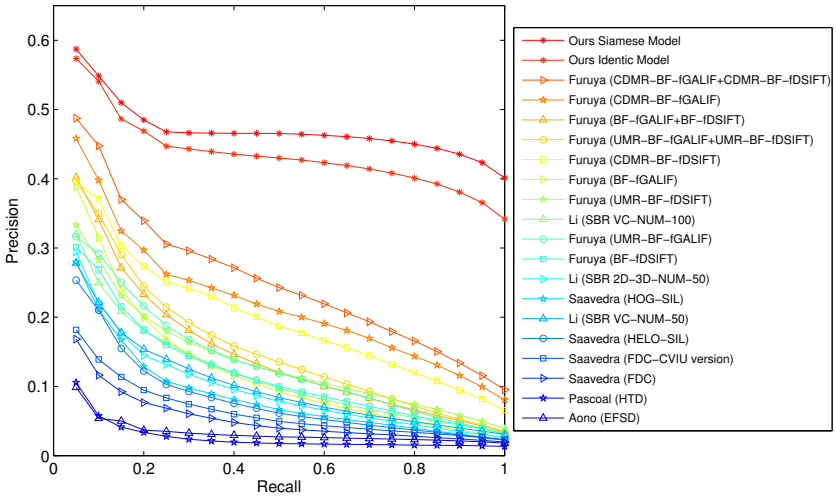


Figure 3.10: Performance comparison on SHREC'13. Please refer to [Li et al., 2014a] for the descriptions of the compared methods.

From the Figure 3.10 we can see that our method significantly outperformed other comparison methods. On SHREC'13 benchmark, the performance gain of our method is already 10% when the recall is small. More importantly, the whole curve decreases much slower than other methods when the recall increases, which is desirable because it shows the method is more stable. The proposed method has a higher performance gain (30%) when recall reaches 1. Figure 3.11 shows that, on SHREC'14 benchmark, our method consistently shows increased precision as the recall increase (approx. 10%). This demonstrates our method exceeds other methods that use handcrafted features. The comparison clearly shows that our methods benefited from the learned features a lot, and is much more stable at higher recall level. One reason is that the CNN features are more representative than the handcrafted features, thus it is more capable to capture similarities of different classes without loose the discriminative power.

We further show the standard metrics for comparison. These metrics examine the retrieval results from different perspectives. For simplicity, only the best method is selected from each research group reported in [Li et al., 2014a] and [Li et al., 2014b]. As shown in Table 3.3, the proposed method has a better number in every metric on

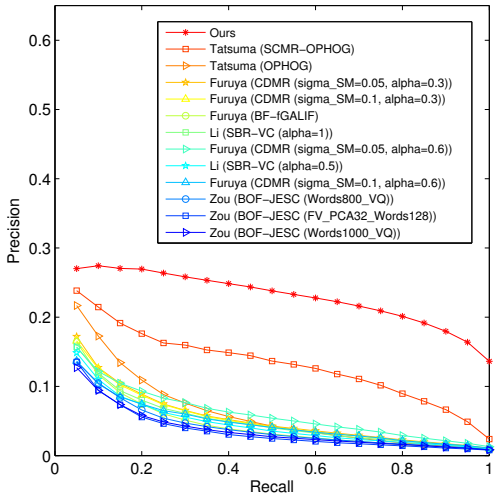


Figure 3.11: Performance comparison on SHREC'14. Detailed descriptions of the compared methods can be found in [Li et al., 2014b].

both benchmarks. This further demonstrates our method is superior.

We also compared to the case where both networks are identical, *i.e.*, both views and sketches use the same Siamese network. Figure 3.10 suggests that this configuration is inferior to our proposed version, but still, it is better than all other methods. This supports our hypothesis that the variations in two domains are different. This also sends a message that using the same features (handcrafted or learned) for both domains may not be ideal.

Table 3.3: Comparison on SHREC'13 dataset. The best results are shown in red, and the second best results are shown in blue.

SHREC'13						
	NN	FT	ST	E	DCG	mAP
Ours	0.405	0.403	0.548	0.287	0.607	0.469
Identical Model	0.389	0.364	0.516	0.272	0.588	0.434
[Furuya and Ohbuchi, 2013]	0.279	0.203	0.296	0.166	0.458	0.250
[Li et al., 2014a]	0.164	0.097	0.149	0.085	0.348	0.116
[Sousa and Fonseca, 2010]	0.017	0.016	0.031	0.018	0.240	0.026
[Saavedra et al., 2012]	0.110	0.069	0.107	0.061	0.307	0.086

Table 3.4: Comparison on SHREC'14 dataset. The best results are shown in red, and the second best results are shown in blue.

	SHREC'14					
	NN	FT	ST	E	DCG	mAP
Ours	0.239	0.212	0.316	0.140	0.496	0.228
[Tatsuma et al., 2012]	0.160	0.115	0.170	0.079	0.376	0.131
[Furuya and Ohbuchi, 2013]	0.109	0.057	0.089	0.041	0.328	0.054
[Li et al., 2014a]	0.095	0.050	0.081	0.037	0.319	0.050

Within-domain retrieval Finally, we show the retrievals in the same domain. This interesting experiment shall be straightforward to report because the data is readily available, but was not shown before in any literature. Since this is a "by-product" of our method, we do not tune up any parameter or re-train the system.

Figure 3.12 and 3.13 visualize some retrieval results in each domain, respectively. Table 3.5 further reports the statistics. The retrieval results demonstrate our method is powerful in learning the features for both within domain and cross-domain. From these figures, one can see that the view domain is much more consistent than the sketch domain. Comparing Table 3.5 to Table 3.3, we conclude that the inconsistency in the sketches is the most challenging issue in the sketch based 3D shape retrieval.

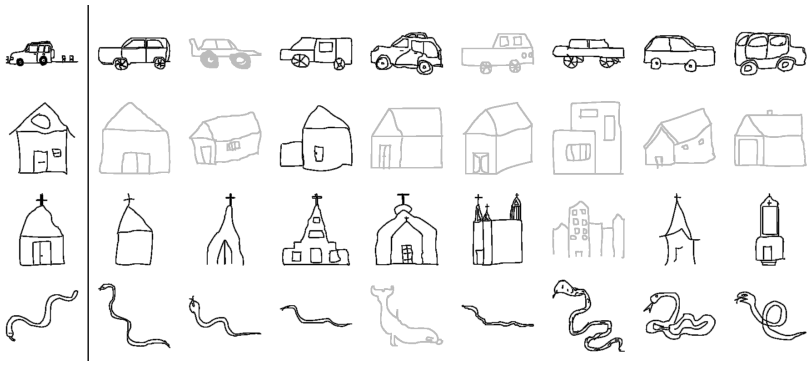


Figure 3.12: Sketch-sketch retrieval on SHREC'13. The incorrect retrievals are marked as light gray.

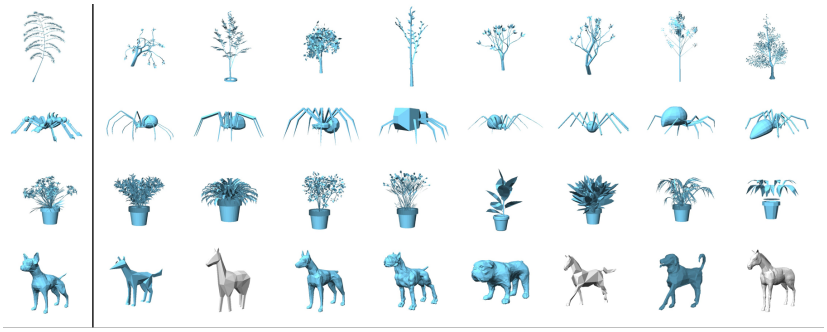


Figure 3.13: View-view retrieval on SHREC'13. The cyan denotes the correct retrievals.

Table 3.5: Standard metrics for the within-domain retrieval on SHREC'13.

	NN	FT	ST	E	DCG	mAP
view	0.965	0.877	0.982	0.536	0.971	0.909
sketch	0.431	0.352	0.514	0.298	0.679	0.373

3.4 Summary

This chapter proposes to learn feature representations for sketch based 3D shape retrieval. Instead of computing "best views" and match them against queries, we use predefined viewpoints for the whole dataset and adopt two Siamese CNNs, one for views and one for sketches. The experiment results on three large datasets demonstrated that our method is significantly superior. At the end, I show an application of the proposed method on hand drawn sketch retrieval and use fully convolutional networks to learn the sketch representations. The experiment results show that the proposed similarity learning method is effective for general tasks.

Chapter 4

Manipulation Action Prediction Using LSTM

In the last chapter, I studied the neural network approaches for representation learning and similarity learning of static inputs. From this chapter, I turn to the investigation of representation learning approaches for sequential data, because it has more complex feature representations due to the dynamic changes and how to effectively learn representations for such kind of inputs is an interesting question. Sequential data has wide applications in many vision tasks, such as human activity recognition, scene understanding, and object tracking. I choose to focus on the human actions recognition problem and study three related tasks around this topic. In this chapter, I first study the action prediction problem and propose an approach for human action prediction based on the Long Short-Term Memory model.

Human action and activity understanding has been a topic of great interest in Computer Vision and Robotics in recent years. Many techniques have been developed for recognizing actions and large benchmark datasets have been proposed, most of them focus on full-body actions [Mandary et al., 2015; Takano et al., 2015; Schuldt et al., 2004; Moeslund et al., 2006; Turaga et al., 2008]. Typically, computationally approaches treat action recognition as a classification problem, where the input is a previously segmented video, and the output a set of candidate action labels.

However, there is more to action understanding, as demonstrated by biological vision. As we humans observe, we constantly perceive and update our beliefs about the observed action and about future events. We constantly recognize the ongoing action. Similarly, cognitive robots that will assist human partners will need to understand their intended actions at an early stage. If a robot needs to act, it cannot have a long delay in visual processing. It needs to recognize in real-time to plan its actions. A fully functional perception-action loop requires the robot to predict, so it

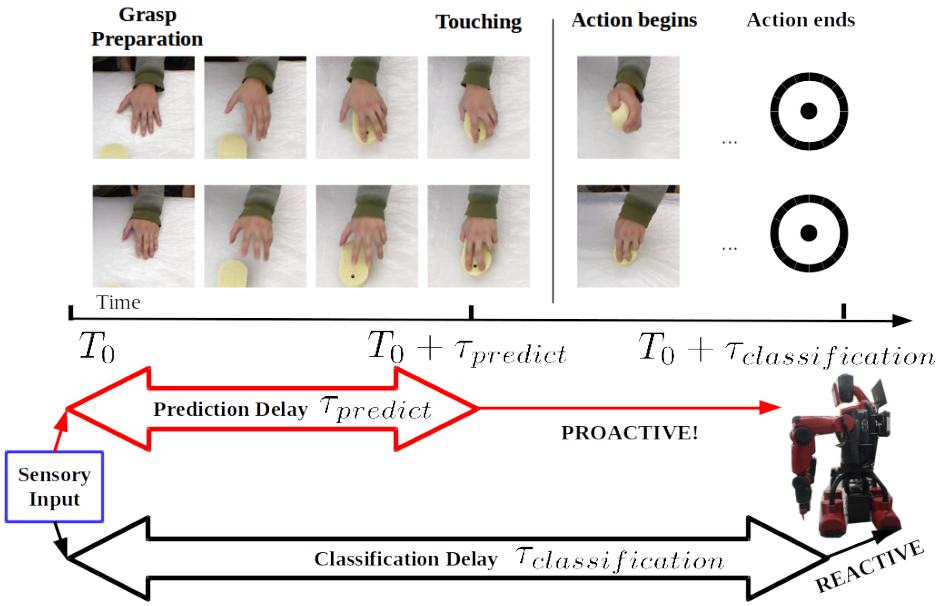


Figure 4.1: Two examples demonstrate that early movements are strong indicators of the intended manipulation actions. Inspired by this, our system performs action predictions from early visual cues. Compared to the classification delay, earlier prediction of action significantly reduces the delay in real-time interaction, which is fundamentally important for the proactive system. (Top row: squeezing a sponge; bottom row: wiping a table with a sponge.)

can efficiently allocate future processes.

Finally, even vision processes for multimedia tasks may benefit from being predictive. Interpreting human activities is a very complex task and requires both, low-level vision processes and high-level cognitive processes with knowledge about actions. [Gupta and Davis, 2008; Kulkarni et al., 2013]. Considering the challenges in state of the art visual action recognition, Aloimonos and Fermüller [2015] argue that by integrating closely the high-level with the low-level vision processes, with the high-level modifying the visual processes, a better recognition may be achieved. Prediction plays an essential component in this interaction. In this work, I focus on manipulation actions and how visual information of hand movements can be exploited for predicting future action so that the crucial delay in the control loop can be shortened (for an illustration see Figure 4.1).

Hand movements and actions have long been studied in Computer Vision to create systems for applications such as recognition of sign language [Erol et al., 2007]. More recent applications include gesture recognition [Molchanov et al., 2015], visual interfaces [Melax et al., 2013], and driver analysis [Ohn-Bar and Trivedi, 2014].

Different methods model the temporal evolution of actions using formalisms such as Hidden Markov models [Starner et al., 1998], Conditional Random Fields [Wang et al., 2006] and 3D Convolutional Neural Networks [Molchanov et al., 2015]. While in principle, some of these approaches, could be used for online prediction, they are always treated as recognition modules.

In recent years a number of works have developed tools for general hand pose estimation and hand tracking, which can be building blocks for applications involving hand movement recognition. For example, building on work on full-body recognition [Shotton et al., 2013], Keskin et al. [2013] develops a learning-based approach using depth contrast features and Random Forest classifiers. Oikonomidis et al. [2011] in a model-based approach use a 27-degree of freedom model of the hand built from geometric primitives and GPU accelerated Particle Swarm Optimization. So far, these trackers and pose estimators work well on isolated hands, but methods still struggle with hands in interaction with objects [Supancic et al., 2015], although there are efforts underway to deal with such situations [Panteleris et al., 2015].

The inspiration for this work comes from studies in Cognitive Sciences on hand motion. The grasp and the movement kinematics are strongly related to the manipulation action [Jeannerod, 1984]. It has been shown that an actor's intention shapes his/her movement kinematics during movement execution, and, furthermore, observers are sensitive to this information [Ansuini et al., 2015]. They can see early differences in visual kinematics and use them to discriminate between movements performed with different intentions. Kinematic studies have looked at such physical differences in movement. For example, Ansuini et al. [2008] found that when subjects grasped a bottle for pouring, the middle and the ring fingers were more extended than when they grasped the bottle with the intent of displacing, throwing, or passing it. Similarly, Crajé et al. [2011] found that subjects placed their thumb and index fingers in higher positions when the bottle was grasped to pour than to lift.

It appears that the visual information in the early phases of the action is often sufficient for observers to understand the intention of an action. Starting from this intuition, we a.) conducted a study to evaluate humans' performance in recognizing manipulation actions; b.) implemented a computational system using state-of-the-art learning algorithms.

The psychophysical experiment was designed to evaluate human's performance in recognizing manipulation actions in their early phases. These include: 1) the grasp preparation, which is the phase when the hand moves towards the object and the

fingers shape to touch the object; 2) the grasp, when the hand comes in contact with the object to hold it in a stable position; and 3) the early actual action movement of the hand together with the object. Throughout these three phases, observers' judgment of the action becomes more reliable and confident. The study gives us an insight into the difficulty of the task and provides data for evaluating our computational method.

Our computational approach processes the sensory input as a continuous signal and formulates action interpretation as a continuous updating of the prediction of intended action. From the stream of video input, we continuously predict the identity of the ongoing action. We humans are able to update our beliefs about the observed action, and predict it before it is completed. This capability is essential to be proactive and react to the actions of others. Robots that interact with humans also need such capability. Predicting future actions of their counterpart allows them to allocate computational resources for their own reaction appropriately. For example, if a person is passing a cup to the robot, the robot has to understand what is happening well before the action is completed, so it can prepare the appropriate action to receive it. Furthermore, vision processes have to be initiated and possibly tuned with predicted information, so the cup can be detected at the correct location, its pose estimated, and possibly other task-specific processes performed (for example, the content of the cup may need to be recognized).

In order to solve the action prediction task, we We use an RNN to recognize the ongoing action from video input. A camera records videos of humans performing a number of manipulation actions on different objects. For example, they "drink" from a cup, "pour" from it, "pound", "shake", and "move" it; or they "squeeze" a sponge, "flip" it, "wash", "wipe", and "scratch" with it. Our system extracts patches around the hands and feeds these patches to an RNN, which was trained offline to predict in real-time the ongoing action.

The main contributions of this work include

1. We present the first computational study on the prediction of observed dexterous actions

2. We demonstrate an implementation for predicting intended dexterous actions from videos;

3. We provide new datasets that serve as test-beds for the action prediction task.

4.1 Related work

We will focus our review on studies along the following concepts: the idea of prediction, including prediction of intention and future events, prediction beyond appearance, the representations of human actions, works on hand actions, manipulation datasets, and action classification as a continuous process using various kinds of techniques and different kinds of inputs.

Prediction of action intention and future events A small number of works in Computer Vision have aimed to predict intended action from visual input. For example, Joo et al. [2014] use a ranking SVM to predict the persuasive motivation (or the intention) of the photographer who captured an image. Pirsiavash et al. [2014] seek to infer the motivation of the person in the image by mining knowledge stored in a large corpus using natural language processing techniques. Yang et al. [2015] propose that the grasp type, which is recognized in single images using CNNs, reveals the general category of a person's intended action. In [Koppula and Saxena, 2016], a temporal Conditional Random Field model is used to infer anticipated human activities by taking into consideration object affordances. Other works attempt to predict events in the future. For example, Kitani et al. [2012] use concept detectors to predict future trajectories in surveillance videos. [Fouhey and Zitnick, 2014] learn from sequences of abstract images the relative motion of objects observed in single images. Walker et al. [2014] employ visual mid-level elements to learn from videos how to predict possible object trajectories in single images. More recently, [Vondrick et al., 2016] learn using CNN feature representations how to predict from one frame in the video the actions and objects in a future frame. Our study also is about the prediction of future events using neural networks. But while the above studies attempt to learn abstract concepts for reasoning in a passive setting, our goal is to perform online prediction of specific actions from video of the recent past.

Physics beyond appearance Many recent approaches in Robotics and Computer Vision aim to infer physical properties beyond appearance models from visual inputs. Xie et al. [2013] propose that implicit information, such as functional objects, can be inferred from videos. [Zhu et al., 2015] takes a task-oriented viewpoint and models objects using a simulation engine. The general idea of associating images with forces has previously been used for object manipulation. The technique is called vision-based force measurement and refers to the estimation of forces according to

the observed deformations of an object [Greminger and Nelson, 2004]. Along with this idea, recently Aviles et al. [2015] proposed a method using an RNN for the classification of forces due to tissue deformation in robotic-assisted surgery.

Representations of actions Spatial-temporal features have been proposed to represent video clips [Dollár et al., 2005; Laptev and Lindeberg, 2006; Klaser et al., 2008]. A number of research works using spatial-temporal representations based on optical flow [Laptev and Lindeberg, 2006] or 3D gradients [Scovanner et al., 2007; Klaser et al., 2008]. However, such methods suffer from high computational costs. To avoid the high computational costs for video processing, [Laptev, 2005] and [Rohrbach et al., 2012] used trajectories of key points instead of the raw video frames. The representation is then extracted from these trajectories to model the dynamics of human action. But, these methods are often sensitive to noise and the tracking error of the key points. Wang et al. [2011] proposed dense trajectory feature, which achieved better robustness by drastically increasing the number of tracked trajectories, then the video features around these tracks are extracted as the action representation.

Dexterous actions The robotics community has been studying perception and control problems of dexterous actions for decades [Shimoga, 1996]. Some works have studied grasping taxonomies [Cutkosky, 1989; Feix et al., 2009], how to recognize grasp types [Rogez et al., 2015] and how to encode and represent human hand motion [Romero et al., 2013]. Pieropan et al. [2013] proposed a representation of objects in terms of their interaction with human hands. Real-time visual trackers [Oikonomidis et al., 2011] were developed, facilitating computational research with hands. Recently, several learning based systems were reported that infer contact points or how to grasp an object from its appearance [Saxena et al., 2008; Lenz et al., 2015].

Manipulation datasets A number of object manipulation datasets have been created, many of them recorded with wearable cameras providing egocentric views. For example, the Yale grasping dataset [Bullock et al., 2015] contains wide-angle head-mounted camera videos recorded from four people during regular activities with images tagged with the hand grasp (of 33 classes). Similarly, the UT Grasp dataset [Cai et al., 2015] contains head-mounted camera video of people grasping objects on a table and was tagged with grasps (of 17 classes). The GTEA set [Fathi et al., 2011] has egocentric videos of household activities with the objects annotated. Other datasets have egocentric RGB-D videos. The UCI-EGO [Rogez et al., 2014]

features object manipulation scenes with annotation of the 3D hand poses, and the GUN-71 [Rogez et al., 2015] features subjects grasping objects, where care was taken to have the same amount of data for each of the 71 grasp types. Our datasets, in contrast, are taken from the third-person viewpoint. While having less variation in the visual setting than most of the above datasets, it focuses on the dynamic aspects of different actions, which manipulate the same objects. The dataset can be accessed from http://users.cecs.anu.edu.au/~fwang/action_prediction/index.html.

Action recognition as an online process Action recognition has been extensively studied. However, few of the proposed methods treat action recognition as a continuous (in the online sense) process; typically, action classification is performed on *whole* action sequences [Schuldt et al., 2004; Ijina and Mohan, 2014]. Recent works include building robust action models based on MoCap data [Wang et al., 2014] or using CNNs for large-scale video classification [Karpathy et al., 2014; Simonyan and Zisserman, 2014c]. Most methods that take into account action dynamics usually operate under a stochastic process formulation, e.g., by using Hidden Markov Models [Lv and Nevatia, 2006] or semi-Markov models [Shi et al., 2011]. HMMs can model relations between consecutive image frames, but they cannot be applied to high-dimensional feature vectors. In [Fanello et al., 2013] the authors propose an online action recognition method by means of SVM classification of sparsely coded features on a sliding temporal window. Most of the above methods assume only short-time dependencies between frames, make restrictive assumptions about the Markovian order of the underlying processes and/or rely on global optimization over the whole sequence.

In recent work, a few studies proposed approaches to recognition of partially observed actions under the headings of *early event detection* or *early action recognition*. Ryoo [2011] creates a representation that encodes how histograms of spatio-temporal features change over time. In a probabilistic model, the histograms are modeled with Gaussian distributions, and MAP estimation over all subsequences is used to recognize the ongoing activity. A second approach in the paper models the sequential structure in the changing histogram representation, and matches subsequences of the video using dynamic programming. Both approaches were evaluated on full body action sequences. In [Ryoo and Matthies, 2013] images are represented by spatio-temporal features and histograms of optical flow, and a hierarchical structure of video-subsegments is used to detect partial action sequences in first-person

videos. Ryoo et al. [2015] performs early recognition of activities in first person-videos by capturing special sub-sequences characteristic for the onset of the main activity. Hoai and De la Torre [2014] propose a maximum-margin framework (a variant of SVM) to train visual detectors to recognize partial events. The classifier is trained with all the video sub-sequences of different length. To enforce the sequential nature of the events, additional constraints on the score function of the classifier are enforced, for example, it has to increase as more frames are matched. The technique was demonstrated in multiple applications, including detection of facial expressions, hand gestures, and activities.

Our contribution regarding action recognition is that we demonstrate an online prediction system based on the deep learning architectures. Furthermore, the subject of our study is novel. The previous approaches consider the classical full body action problem. Here our emphasis is specifically on the hand motion, not considering other information such as the objects involved.

4.2 The approach

This section describes our proposed model for action prediction. We focus on manipulation actions where a person manipulates an object using a single hand. Given a video sequence of a manipulation action, the goal is to generate a sequence of predicted actions while watching the video. Instead of assigning an action label to the whole sequence, we continuously update the prediction as frames of the video are processed.

Visual representation The visual information most essential for manipulation actions comes from the pose and movement of the hands, while the body movements are less important. Therefore, we first track the hand using a mean-shift based tracker [Bradski, 1998], and use cropped image patches centered on the hand. In order to create abstract representations of image patches, we project each patch through a pre-trained CNN model (shown in Figure 4.2). This provides the feature vectors used as input to the RNN.

Action prediction In our model, the LSTM is trained using as input a sequence of feature vectors $x = \{x_1, x_2, \cdots, x_T\}$ and the action labels $y \in [1, N]$. The hidden states and the memory cell values are updated according to Equations (2.3)-(2.7).

Then logistic regression is used to map the hidden states to the label space as follows:

$$P(Y = i|h_t, W_u, b_u) = softmax_i(W_u h_t + b_u). \tag{4.1}$$

Then the predicted action label is obtained as:

$$\hat{y}_t = argmax_i P(Y = i|h_t, W_u, b_u). \tag{4.2}$$

Model learning We follow the common approach of training the model by minimizing the negative log-likelihood over the dataset $\mathcal{D}$. The loss function is defined as

$$l(\mathcal{D}, W, b) = -\sum_{i=0}^{|\mathcal{D}|} \log(P(Y = y^{(i)}|x^{(i)}, W, b)), \tag{4.3}$$

where W and b denote the weight matrix and the bias term. These parameters can be learned using the stochastic gradient descent algorithm.

Since we aim for the ongoing prediction rather than a classification of the whole sequence, we do not perform a pooling over the sequences to generate the outputs. Each prediction is based only on the current frame and the current hidden state, which implicitly encodes information about the history. In practice, we achieve learning by performing backpropagation at each frame.

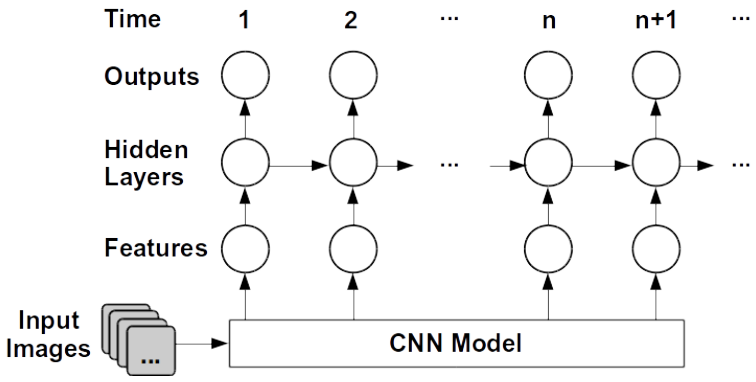


Figure 4.2: The flowchart of the action prediction model, where the LSTM model is unfolded over time.

4.3 Data collection

A Manipulation Action Dataset (MAD) is collected. It contains videos of people performing dexterous actions on various objects. The focus was to have different actions (with significant variation) on the same object. We asked five subjects to perform a number of actions with five objects, namely *cup*, *stone*, *sponge*, *spoon*, and *knife*. Each object was manipulated in five different actions with five repetitions, resulting in a total of 625 action samples. Table 4.1 lists all the object and action pairs considered in MAD.

Table 4.1: Object and Action pairs of MAD

Object	Actions
cup	drink, pound,shake,move,pour
stone	pound,move,play,grind,carve
sponge	squeeze,flip,wash,wipe,scratch
spoon	scoop,stir,hit,eat,sprinkle
knife	cut,chop,poke a hole,peel,spread

Since our aim was to build a system that can predict the action as early as possible, we wanted to study the prediction performance during different phases in the action. To facilitate such studies, we labeled the time in the videos when the hand establishes contact with the objects, which we call the "touching point."

4.4 An experimental study with humans

We were interested in how humans perform in prediction at different phases during the action. Intuitively, we would expect that the hand configuration and motion just before the grasping of the object, when establishing contact, and shortly after the contact point can be very informative of the intended action. Therefore, in order to evaluate how early we can accurately predict, we investigated the prediction performance at certain time offsets with respect to the touching point.

We picked three objects from the MAD dataset for the study, namely *cup*, *sponge* and *spoon*. The prediction accuracy at four different time points was then evaluated: 10 frames before the contact point, exactly at contact, 10, and 25 frames after the contact point. Figure 4.3 shows the interface subjects used in this study.

In a first experiment, we asked 18 human subjects to perform the prediction task.

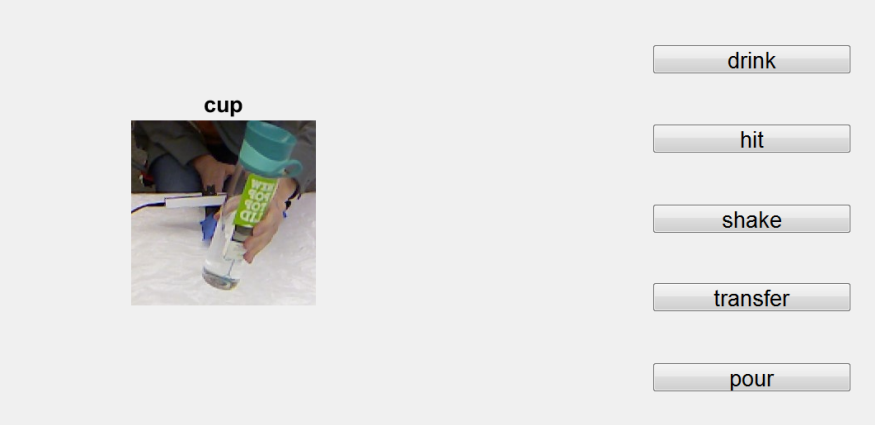
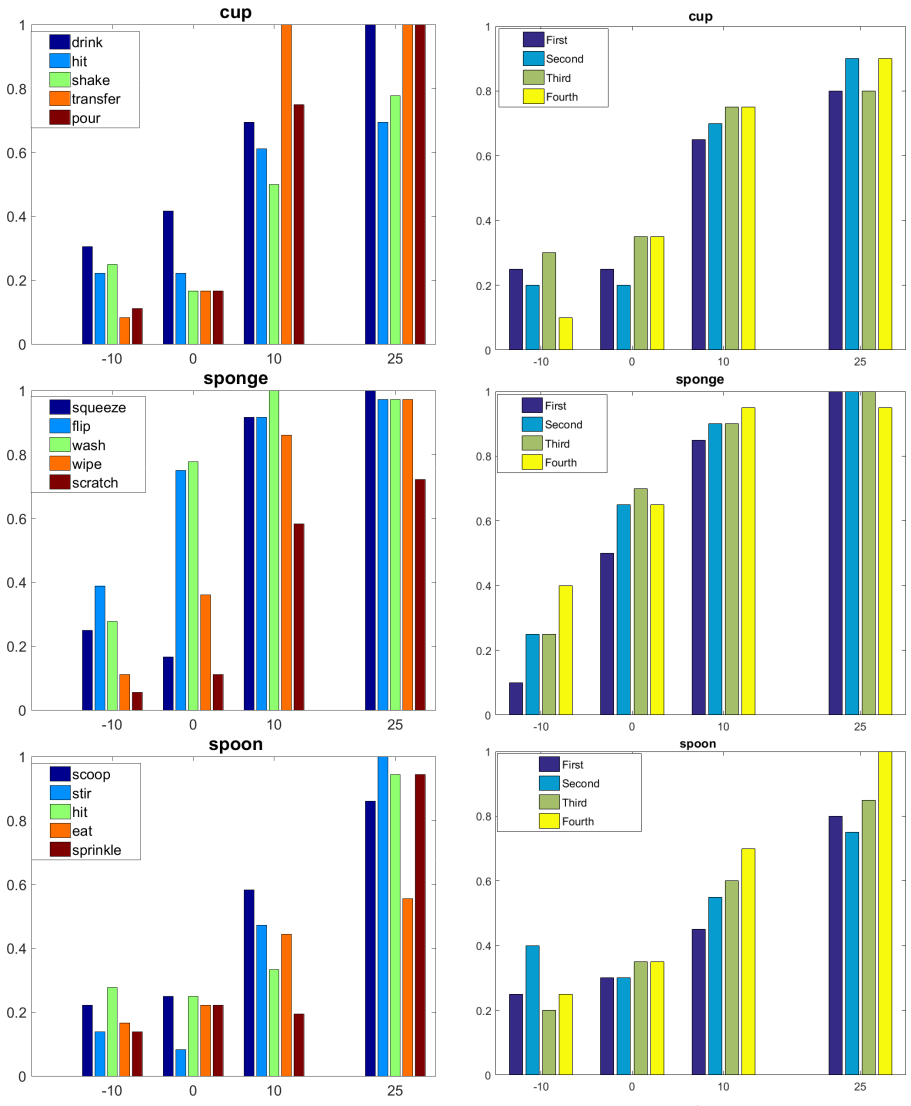


Figure 4.3: Interface used in the human study.

For each of the three objects, after a short "training" phase in which all actions were demonstrated at full length, each subject was shown a set of 40 video segments and was asked to identify the currently perceived action. Each segment ended at one of the four time points relative to the contact point described above and was constructed from the same hand patches used in the computational experiments. All actions and all time offsets were equally represented. Figure 4.4(a) plots subjects' average prediction performance for the different objects, actions and time offsets. With five actions per object, 20% accuracy corresponds to chance level. As we can see, the task of judging before and even at contact point, was very difficult and classification was at chance for two of the objects, the spoon and the cup, and above chance at contact only for the sponge. At 10 frames after contact human classification becomes better and reaches in average about 75% for the sponge, 60% for the cup, but only 40% for the spoon. At 25 frames subjects' judgment becomes quite good with the sponge going above 95% for four of the five actions, and the other two actions in average at about 85%. We can also see which actions are easily confused. For the cup, 'shake' and 'hit' were even after 25 frames still difficult to recognize, and for the spoon, the early phases of movement for most actions appeared similar, and 'eat' was most difficult to identify.

To see whether there is additional distinctive information in the actors' movement, and subjects can take advantage of it with further learning, we performed a second study. Five participating subjects were shown 4 sets of 40 videos for each object, and this time they were given feedback on which was the correct action. Figure 4.4(b) shows the overall success rate for each object and time offset over the four



(a) without feedback (b) with feedback

Figure 4.4: Human prediction performance. (a) First study (without feedback). Success rate for three objects (cup, sponge, and spoon) for five different actions at four time offsets. (b) Second study (with feedback). Success rate for three objects averaged over five actions over four sets of videos at four offsets.

sets. If learning occurs, subjects' should improve from the first to the fourth set. The graphs show that there is a bit of learning. The effect is largest for the spoon, where subjects can learn to better distinguish at 10 frames after contact. The focus was to have different actions (with significant variation) on the same object.

4.5 Experimental results

The two algorithms have been implemented in a system that runs in real-time on a GPU. This sections reports three experimental evaluations. The first experiment evaluates the prediction performance as an on-going task, the second compares our action recognition algorithm against human performance, and the third evaluates our force estimation.

4.5.1 Hand action prediction on MAD dataset

Our approach uses visual features obtained with deep learning, which serves as input to a sequence learning technique.

First, we apply the mean-shift based tracker of Comaniciu et al. [2000] to obtain the locations of the hand. We crop image patches of size 224×224 pixels, centered on the hand. Then our feature vectors are computed by projecting these patches through a convolutional neural network. To be specific, we employ the VGG network [Simonyan and Zisserman, 2014b] with 16 layers, which has been pre-trained on the ImageNet. We take the output of layer *fc7* as feature vector (4096 dimensions), which we then use to train a one layer LSTM model for action prediction.

Our LSTM model has hidden states of 64 dimensions, with all the weight matrices randomly initialized using the normal distribution. We first learn a linear projection to map the 4096 input features to the 64 dimensions of the LSTM. We use mini-batches of 10 samples and the adaptive learning rate method to update the parameters. The training stops after 100 epochs in all the experiments.

To evaluate the action prediction performance, we performed leave-one-subject-out cross-validation over the five subjects. Each time we used the data from one subject for testing and trained the model on the other four subjects. Then all the results were averaged over the five rounds of testing.

On-going prediction Our goal is to understand how the recognition of action improves over time. Thus, we plot the prediction accuracy as a function of time, from

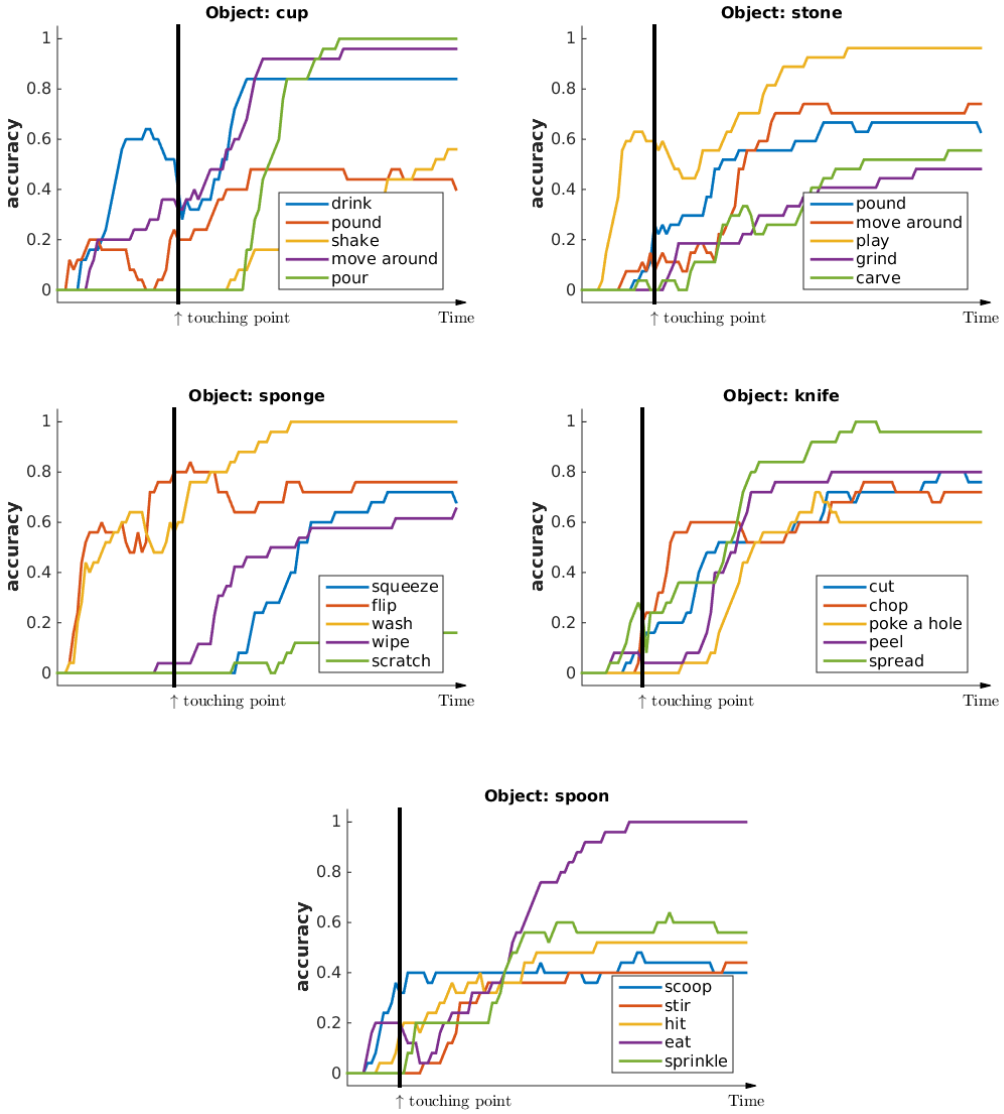


Figure 4.5: Prediction accuracies over time for the five different objects. The black vertical bars show the touching point. For each object, we warped and aligned all the sample sequences so that they align at the same touching point. Best viewed in color.

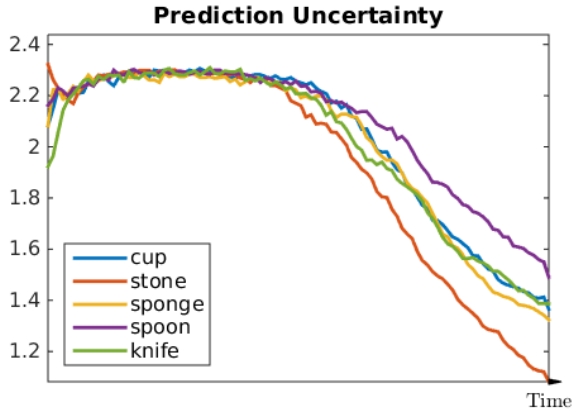


Figure 4.6: Prediction uncertainty computed from the entropy. All sample sequences are warped and aligned to the same length. Best viewed in color.

the action preparation to the end of the action. Our system performs predictions based on every new incoming frame as the action unfolds.

Figure 4.5 shows the change in prediction accuracy over time. For a given action video, our system generates for each frame a potential score vector (with one value for each action) to form a score sequence of the same length as the input video. Since the actions have different length, we aligned them at the touching points. To be specific, we resampled the sequences before and after the touching points to the same length. For each object, we show the prediction accuracy curves of the five actions. The vertical bar in each figure indicates the time of the *touching point*. The touching point splits the sequence into two phases: the "preparation" and the "execution". It is interesting to see that for some object-action pairs our system yields high prediction accuracy even before the touching point, e.g. the "cup - drink" and "sponge - wash".

Figure 4.6 shows the change of prediction uncertainty over time for each of the five objects. This measure was derived from the entropy over the different actions. As can be seen, in all cases, the uncertainty drops rapidly as the prediction accuracy rises along time.

Classification results At the end of the action, the on-going prediction task becomes a traditional classification. To allow evaluating our method on classical action recognition, we also computed the classification results for the whole video. The estimate over the sequence was derived as a weighted average over all frames using a linear weighting with the largest value at the last frame. To be consistent with the above, the classification was performed for each object over the five actions consid-

Table 4.2: Comparison of classification accuracies on different objects

Object/Action	SVM	HMM	LSTM HOG	LSTM VGG16
cup/drink	79.1%	96.0%	82.9%	92.5%
cup/pound	20.0%	81.7%	40.0%	73.3%
cup/shake	64.3%	56.8%	32.6%	83.3%
cup/move	62.7%	53.2%	51.9%	82.1%
cup/pour	60.0%	100.0%	80.3%	80.8%
stone/pound	26.7%	73.3%	60.0%	73.3%
stone/move	87.8%	68.0%	90.0%	61.4%
stone/play	64.6%	97.1%	60.5%	86.7%
stone/grind	28.3%	45.0%	60.0%	46.7%
stone/carve	43.3%	28.5%	66.0%	39.1%
sponge/squeeze	41.1%	81.7%	64.3%	83.4%
sponge/flip	53.3%	91.0%	96.0%	71.0%
sponge/wash	85.9%	84.6%	91.1%	92.5%
sponge/wipe	46.9%	47.5%	58.1%	46.3%
sponge/scratch	30.0%	0.0%	43.3%	15.0%
spoon/scoop	39.0%	27.1%	53.6%	32.0%
spoon/stir	45.3%	30.0%	20.0%	74.3%
spoon/hit	28.9%	20.0%	22.4%	56.7%
spoon/eat	65.0%	79.2%	78.1%	81.1%
spoon/sprinkle	60.0%	25.0%	40.5%	69.1%
knife/cut	33.5%	33.7%	49.6%	75.3%
knife/chop	0.0%	45.0%	43.3%	72.7%
knife/poke a hole	33.3%	20.0%	51.0%	72.0%
knife/peel	66.3%	28.9%	90.0%	72.5%
knife/spread	38.2%	28.3%	54.3%	74.2%
Avg.	48.1%	53.7%	59.2%	68.3%

ered. Figure 4.7 shows the confusion matrix of the action classification results. One can see that our model achieved high accuracy on various object-action combinations, such as "cup/drink" and "sponge/wash", where the precision exceeds 90%.

We used two traditional classification methods as our baseline: Support Vector Machine (SVM) and Hidden Markov Model (HMM). For the HMM model, we used the mixture of Gaussian assumption and we chose the number of hidden states as five. Since the SVM model doesn't accept input samples of different length, we used a sliding window ($size = 36$) mechanism. We performed the classification over each window and then combined the results using majority voting. For both these baseline methods, we conducted a dimension reduction step to map the input feature

vectors to 128 dimensions using PCA. To further explore the efficiency of the LSTM method in predicting actions on our dataset, we also applied the LSTM model using HoG features as input. The average accuracy was found 59.2%, which is 10% higher than the HMM and 23% higher than the SVM, but still significantly lower than our proposed method.

Discussion It should be noted that this is a novel, challenging dataset with no equivalent publicly available counterparts. Subjects performed the action in unconstrained conditions, and thus there was a lot of variation in their movement, and they performed some of the actions in very similar ways, making them difficult to distinguish, as also our human study confirms.

The results demonstrate that deep learning based continuous recognition of manipulation actions is feasible, providing a promising alternative to traditional methods such as HMM, SVM and other methods based on handcrafted features.

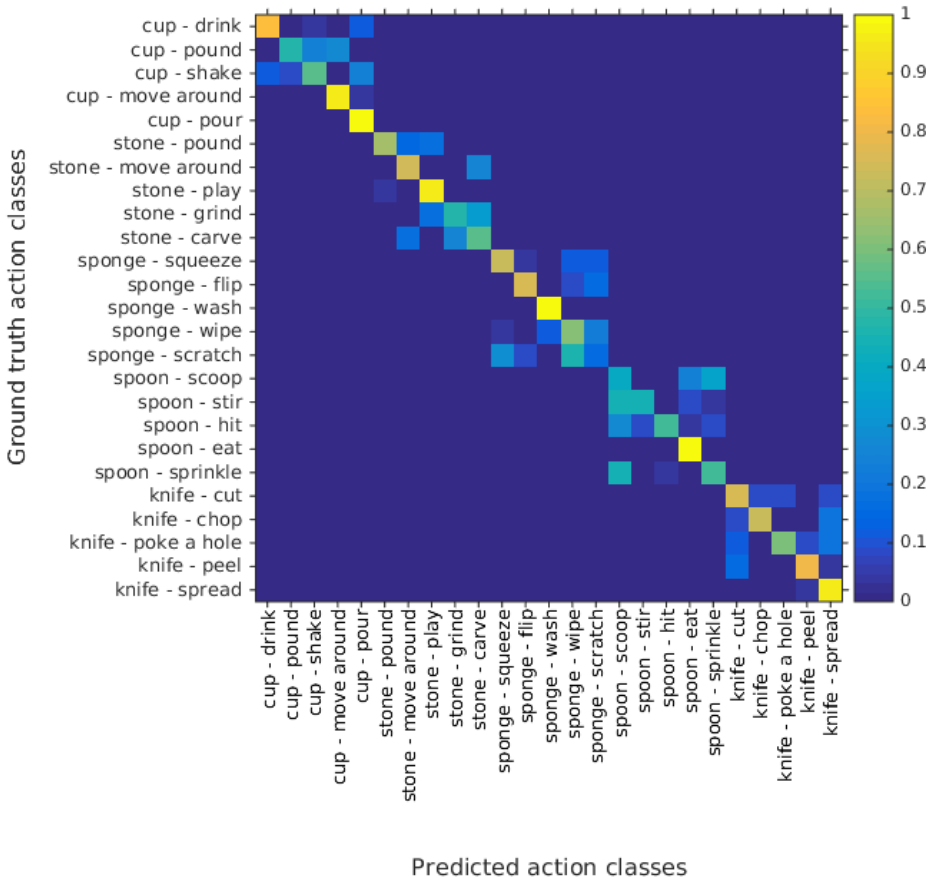


Figure 4.7: Confusion matrix of action classification.

4.5.2 Action prediction on continuous sequences

To further validate the effectiveness of our approach, we applied our model on continuous sequences to predict actions.

Dataset The **50 SALADS** dataset [Stein and McKenna, 2013a] consists of over 4.5 hours of RGBD video and accelerometer data of 25 subjects preparing two salads. It includes 50 videos at a resolution of 640×480, captured at 30 Hz, along with accelerometer data from sensors mounted on tools and objects. The videos are synchronized with accelerometer data.

The activities are described in four granularities. At the semantically highest level, the activities are categorized into "cut and mix ingredients", "prepare dressing", and "serve salad". The lowest level consists of more basic actions, such as "cut cucumber" or "pour olive oil". We are using one of the intermediate levels following [Stein and McKenna, 2013a], which consists of 17 actions. From these 17 actions, we group similar actions. For instance "adding salt" and "adding pepper" are grouped into "adding salt-like condiments". This results in ten action categories that are used in our evaluations.

Evaluation metrics We use two evaluation metrics: the frame-wise classification accuracy, and the edit score of the generated segmentations. The edit score is defined as $(1 - D(y, \hat{y})) * 100.0$, where y denotes the ground truth segmentation, $\hat{y}$ denotes the predicted segmentation, and $D(y, \hat{y})$ is the normalized Levenshtein distance of the two sequences. Higher values indicate more consistency between the predicted sequences and the ground truth.

Results In this dataset, each video contains consecutive action sequences, which makes the action prediction task much harder without segmentation. We compare to the method proposed in [Lea et al., 2016c], which combined the spatio-temporal CNN model with video segmentation methods. In contrast, we didn't use any kind of video segmentation in our prediction framework. Benefited from the LSTM model, our method can automatically learn the transitions between different actions and produce accurate predictions over all frames. We achieved higher performance by raising the frame-wise accuracy from 72.00% to 88.50%. Our edit score is lower than compared method, because, without any temporal constraints, it is inevitable to have some fluctuated predictions around the action boundaries. This effect can

also be seen in Figure 4.8, which shows two prediction examples of consecutive actions. The first and third bars show the ground truth action classes, while the second and fourth bars show the predicted class labels using our method (different colors indicate different action classes).

Table 4.3: Comparison of classification accuracies on the 50 Salad dataset.

Method	Accuracy	Edit Score
ST-CNN + Seg	72.00%	62.06
Our approach	88.50%	50.25

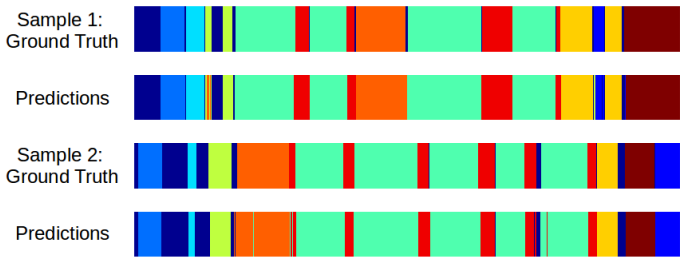


Figure 4.8: Samples of action prediction on the 50 Salad dataset.

4.5.3 Action prediction at the point of contact, before and after

We next compare the performance of our online algorithm (as evaluated in Section 4.5.1) against those of human subjects. Figure 4.9 summarizes the prediction performance per object and time offset. As we can see our algorithm's performance is not significantly behind those of humans. At ten frames after contact, computer lags behind human performance. However, at 25 frames after the contact point, the gaps between our proposed model and human subjects are fairly small. Our model performs worse on the spoon, but this is likely due to the large variation in the way different people move this object. Our human study already revealed the difficulty in judging spoon actions, but the videos shown to subjects featured less actors than were tested with the algorithm. Considering this, we can conclude that our algorithm is already close to human performance in fast action prediction.

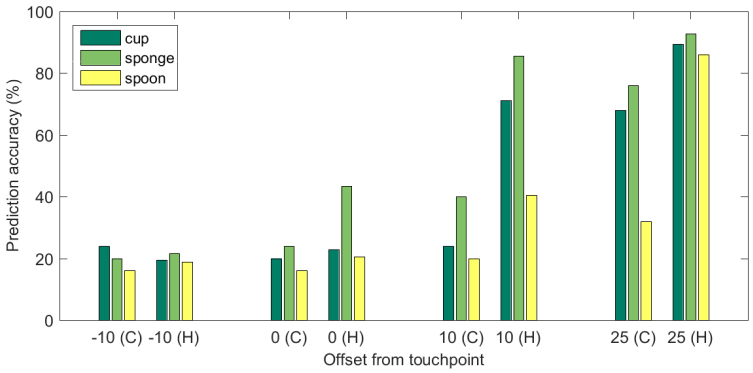


Figure 4.9: Comparison of prediction accuracies between our computational method (**C**) and data from human observers (**H**). Actions are classified at four different time points before, at, and after the touching point (at -10, 0, +10, +25 frames from the touching point). C denotes the learned model, H denotes the psychophysical data).

4.6 Summary

In this chapter, I proposed an approach to action interpretation, which treats the problem as a continuous updating of beliefs and predictions. The ideas were implemented for two tasks: the prediction of perceived action from visual input, and the prediction of force values on the hand. The methods were shown to run in real-time and demonstrated high accuracy performance. The action prediction was evaluated also against human performance, and shown to be nearly on par. Additionally, new datasets of videos of dexterous actions and force measurements were created.

The methods presented here are only a first implementation of a concept that can be further developed along a number of directions. Here, I applied learning on 2D images only, and clearly, this way we also learn properties of the images that are not relevant to the task, such as the background textures. In order to become robust to these "nuisances", 3D information, such as contours and depth features, could be considered in future work. While the current implementation only considers action labels, the same framework can be applied for other aspects of action understanding. For example, one can describe the different phases of actions and predict these sub-actions since different actions share similar small movements. One can also describe the movements of other body parts, e.g., the arms and shoulders. Finally, the predicted forces may be used for learning how to perform actions on the robot. Future work will attempt to map the forces from the human hands onto other actuators, for example, three-fingered hands or grippers.

Chapter 5

Hand Force Prediction Using LSTM

In the previous chapter, I studied an efficient approach for manipulation action prediction. One observation of the manipulation action is that they are driven directly by the physical interactions between human hands and the objects. If one can learn the interaction of the hand and object, it would be helpful for understanding the manipulation action and improving the recognition performance. In this chapter, I study the hand object relationship by estimating the hand forces for manipulation actions. In the meantime, I investigate how to generalize the representation learning methods from vision features to representations of physical or kinematic information.

In recent years, the fast development of the field of cognitive robotics pushed the focus of action recognition research from full-body movements to the more complex manipulation actions. In this kind of actions, multiple entities interact with each other including human hands, tools, and the target objects. Different manipulation targets significantly increased the variations of the hand movements, which make the action recognition task extremely challenging. One possible solution is to explore the additional dynamic information that closely related with the actions, such as the motoric data of the target objects, the forces applied on the objects, or the action consequences. Besides distinguishing the hand movements, we are also interested in the interactions of hand and the manipulated targets, and explore the force patterns that applied on the objects.

One motivation for predicting forces is that the additional data may help increase recognition accuracy. There is evidence that human understand others' actions in terms of their own motor primitives [Gallesse and Goldman, 1998; Rizzolatti et al., 2001]. For a human observer, it is easy to interpret the body movements of others and estimate the forces applied on the target object when watching other's actions. One can also predict the consequences driven by these patterns. These findings have not been modeled in computational terms so far.

Findings of neuroscience on the mirror neuron system [Gallesse and Goldman, 1998; Rizzolatti et al., 2001] provide evidence for a close relationship between mechanisms of action and perception in primates. Humans develop haptic perception through interaction with objects and learn to relate haptic with visual perception. They further develop the capability of hallucinating the haptic stimulus when seeing hands in certain configurations interacting with objects [Tiest and Kappers, 2014]. For example, human observers can easily estimating the deformation of objects by seeing others perform the manipulation action. If someone is holding a paper cup with strong forces, a human observer can tell that the cup may be crushed by watching the hand movements. This capability of hallucinating force patterns from visual inputs is essential for a more detailed analysis of the interaction with the physical world. It can be used to reason about the current interaction between the hand and the object, and to predict the action consequences driven by the estimated force pattern.

Furthermore, the force patterns may be used in robot learning. A popular paradigm in Robotics is imitation learning or learning from demonstration [Argall et al., 2009], where the robot learns from examples provided by a demonstrator. If the forces can be predicted from images, then the force profiles together with the positional information can be used to teach the robot with video only. Many researchers are trying to teach robots actions and skills that involve forces, for example, wiping a kitchen table [Gams et al., 2010], pull and flip tasks [Kober et al., 2000], ironing or opening a door [Kormushev et al., 2011]. These approaches rely on haptic devices or force and torque sensors on the robot to obtain the force profiles for the robot to learn the task. If we can predict the forces exerted by the human demonstrator, the demonstration could become vision only. This would allow us to teach robots force interaction tasks much more efficiently.

By associating vision clues with hand forces, it can be expected to obtain better computational action recognition model. Intuitively, the force vectors, whose dimensions are much lower than the visual descriptors, should provide useful compact information for classification, especially when small number of training data are available.

The aim of this hand force estimation approach is to predict the tactile signal by watching the hand movements of manipulation actions. Figure 5.1 shows one example of the estimated forces over time for the “cup-drink” action. While the action is being conducted, we generate estimations of finger forces for each video frame. The middle plot of curves in Figure 5.1 shows an example of estimated forces

of four fingers, while the bottom plot shows the changes of the actual force values collected with sensors.

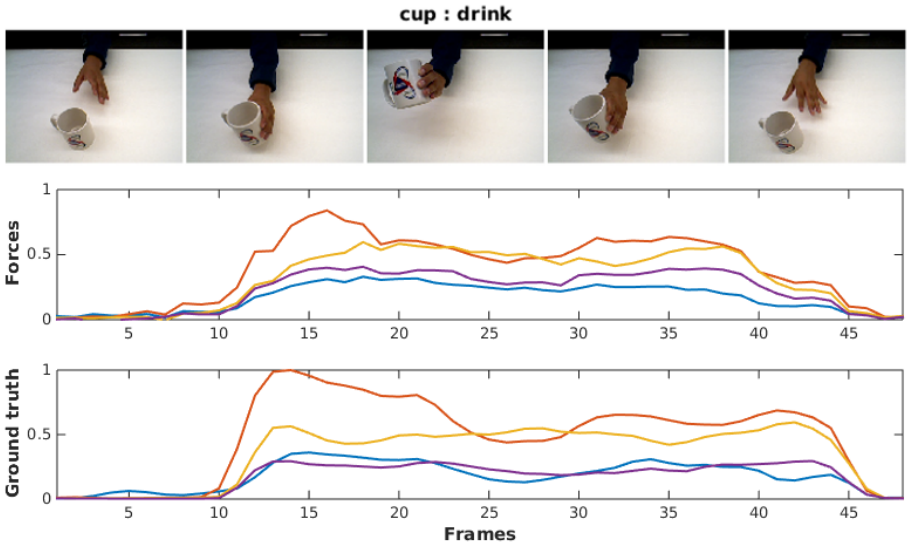


Figure 5.1: Illustration of the hand force estimation. The video frames in the top row show samples of the action "cup-drink". The middle plot shows the estimated forces of four fingers and the bottom plot illustrates the corresponding ground truth force values collected with sensors.

In this work, we propose to adapt the LSTM model to perform regression over finger force values. This can be done by replacing the classification layer in the LSTM architecture with a regression layer that generates force values in a continuous space. To train this force regression model, a dataset with synchronized video sequences of actions and the finger force streams is necessary. Since there is no such kind of dataset available yet, a new dataset is collected with a specially designed sensor glove that can record the finger forces. We collect synchronized streams of videos of actions and force data on the hand, and we used this dataset to evaluate the effectiveness of our approach. The dataset can be accessed from http://users.cecs.anu.edu.au/~fwang/action_prediction/index.html.

5.1 Related work

This section reviews several closely related studies of hand-object interaction and hand force analysis. We focus on the following concepts: the analysis of hand-object interaction, the simulation methods of hand forces, and the inference approaches of manipulation forces.

Analysis of hand-object interaction Most of the works on hand-object interaction formulate the task as an articulated tracking problem, while only a few of these works focus on object manipulation. Romero et al. [2010] proposed a method to track the hand with the object in it by matching the observed hand pose with samples in a large scale dataset with known poses. In contrast, other works focus on pose estimation of the manipulated objects to analysis the hand-object interaction [Kyriazis and Argyros, 2013].

Simulation of hand forces The first work in the computer vision literature to simulate contact forces during hand-object interactions is [Rogez et al., 2015]. The authors segment the hand from RGBD data in single egocentric views and classify the pose into 71 functional grasp categories as proposed in [Liu et al., 2014]. Classified poses are matched to a library of graphically created hand poses, and theses poses are associated with force vectors normal to the meshes at contact points. Thus the forces on the observed hand are obtained by finding the closest matched synthetic model. The method only considered static RGBD images, so it lacks analysis of the dynamic forces for a given action. Besides, the force values are simulated using geometric methods on 3D meshes. Analysis of hand forces for the real actions is necessary.

Inference of manipulation forces Another approach on contact force analysis is proposed in [Pham et al., 2015]. Using as input RGB data, a model-based tracker estimates the poses of the hand and a known object, from which then the contact points and the motion trajectory are derived. Next, the minimal contact forces (nominal forces) explaining the kinematic observations are computed from the Newton-Euler dynamics solving a conic optimization. Humans typically apply more than the minimal forces. These additional forces are learned using a neural network on data collected from subjects, where the force sensors are attached to the object. However, since the framework in this paper is based on a model-based 3D tracking method, it will be quite sensitive in real world applications. The force sensors are put on the object, and thus the data collection is delayed until the hand touches the object, and needs to be adapted to the specific object.

The prior approaches derive the forces using model based-approaches. The forces are computed from the contact points, the shape of the hand, and dynamic observations. In contrast, we propose to collect force feedback from sensors attached to the hand, which is more general and can collect the dynamic changes of the force values.

Furthermore, both the prior methods use RGBD data, while ours is an end-to-end learning approach using as input only RGB images.

5.2 Force estimation framework

This section describes the proposed hand force estimation model and the details of the hand force dataset collection.

5.2.1 Predict hand forces with LSTM

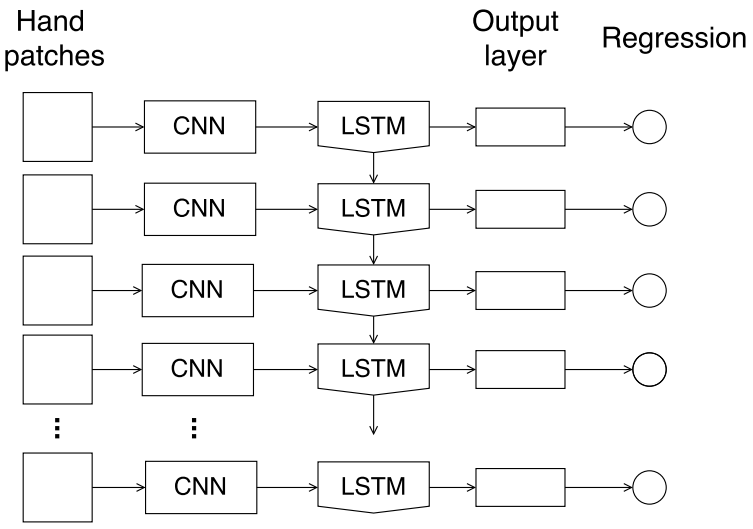


Figure 5.2: The flowchart of the force estimation model. The cropped patches of hands are processed with CNN model to extract features, then the features are passed through the LSTM model to generate estimated forces. The regression layer minimizes the distance between the regressed hand forces and the ground truth.

We use an LSTM based regression model to predict the forces on the fingers from vision input. Given video sequences of actions, as well as simultaneously recorded sequences of force values, we reformulate the LSTM model, such that it predicts force estimates as close as possible to the ground truth values.

The network structure is shown in Figure 5.2. Given a sequence of feature vectors $x = \{x_1, x_2, \cdots, x_T\}$ as the input, the force measurements $v = \{v_1, v_2, \cdots, v_T\}$, $v_t \in R^M$, are used as target values, where M is the number of force sensors attached to the hand. The hidden states and the memory cell values are updated according to

Equations (2.3)-(2.7). Then the forces are estimated as:

$$\hat{v}_t = W_v h_t + b_v. \tag{5.1}$$

To train the force estimation model, we define the loss function as the least squares distance between the estimated value and the ground truth and minimize it over the training set using stochastic gradient descent as:

$$l(\mathcal{D}, W, b) = \sum_{i=0}^{|\mathcal{D}|} \sum_{t=0}^{T} \|\hat{v}_t{}^{(i)} - v_t^{(i)}\|_2^2 \tag{5.2}$$

As for the visual representation, we use pre-trained CNN model to extract features from image patches. Since we focus on manipulation actions, we are more interested in the hand movements rather than the objects, so we first extract the image patches centered on human hands and feed them into the CNN model to get the features of hand movements.

5.2.2 Training

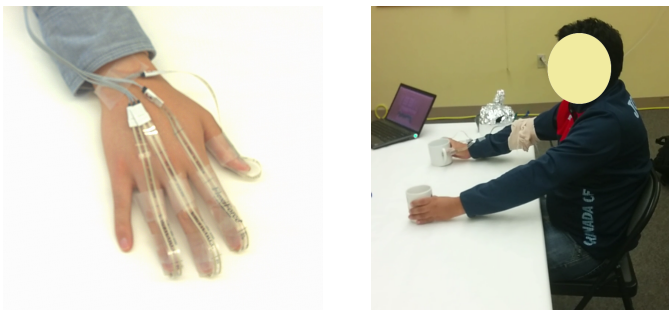
The LSTM regression model is used to estimate the hand forces for each frame. Since people have different preferences in performing actions, the absolute force values can vary significantly for the same action. Therefore, a normalization is applied on the force data, before the training started. All the force values are normalized to the range $[0, 1]$. To generate the visual features, we first run a mean-shift hand tracker to get the locations of hands and then crop patches that centered on the hand with fixed size. A pre-trained CNN model is used to project the hand patches into visual features. The regression model has one LSTM layer with 128 hidden states. Adaptive learning rate method is adopted in training of the network. The batch size is set to be 10 and a fixed training length of 100 epochs is applied to all experiments.

5.3 Data collection of hand forces

5.3.1 A device for capturing finger forces

In this section, we describe the design of a data glove, which is cheap and easy to replicate. Other studies (e.g., Pham et al. [2015]) have measured the contact forces by placing the pressure sensors on the object. However, this may restrict subjects in their manipulation of objects. Our design overcomes this limitation by putting the

sensors on the subject's hand so that the interaction with the object becomes more natural. One immediate advantage is that it is easier to study the manipulation forces for different objects. For example, once the subject is wearing the force sensing glove, there is no extra setup necessary for the objects. As a result, we can easily extend the recordings to multiple objects or objects with irregular geometries.



(a) (b)

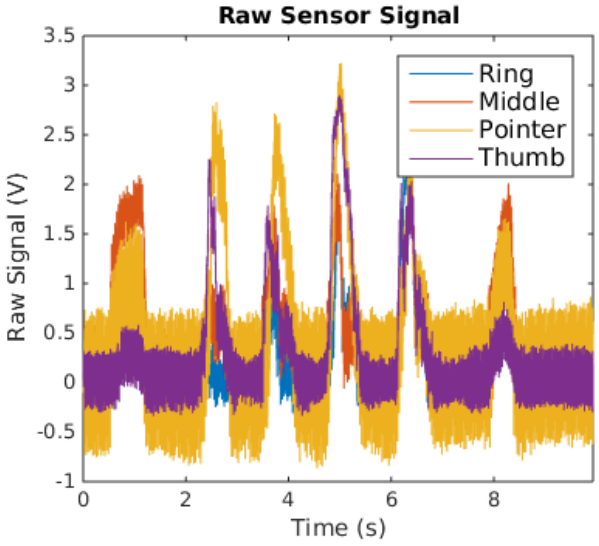
Figure 5.3: Illustration of the force-sensing device. (a) The sensors on four fingers; (b) The set up of data collection.

We use a force sensing device with four force sensors attached directly to four fingers: the thumb, the pointer, the middle and the ring finger (See Figure 5.3(a)). The pinky finger is omitted, as the forces on this finger are usually quite small and not consistent across subjects (as found also by [Pham et al., 2015]). We used the Piezoresistive force sensors by Tekscan, with a documented accuracy (by the manufacturer) of $\pm 3\%$. The sensors at the finger tips have a measurement range of 0 to 8.896 N (2 lb), with a round sensing area of 9.53 mm in diameter. The entire sensing area is treated as one single contact point.

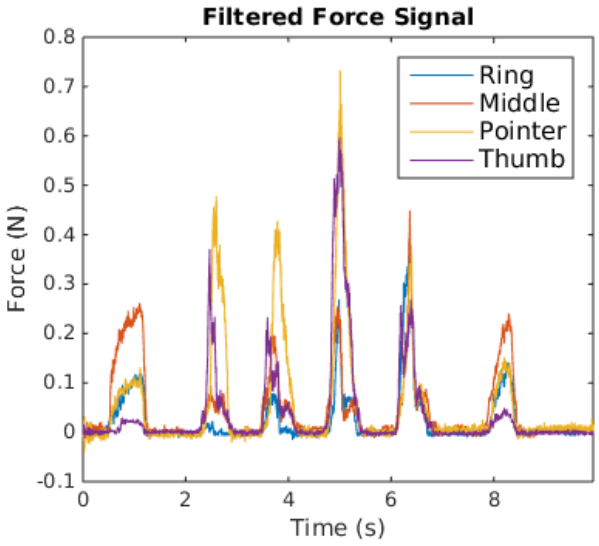
The raw sensor outputs are voltages, which should be translated to the force measurements first. More details of this translation can be found on Tekscan's website: https://www.tekscan.com. In our experiment, we derived the forces perpendicular to the sensor surfaces from voltages as follows,

$$F = 4.448 * \left(C_1 * \frac{V_{out}}{V_{in} - V_{out}} - C_2 \right), \tag{5.3}$$

where V_{out} is the sensor measurement. V_{in}, C_1, and C_2 are fixed constants of the



(a)



(b)

Figure 5.4: Force data collection and preprocessing. (a) The raw, unfiltered voltage signal from the fingertip force sensors. (b) The filtered force signal from the fingertip sensors.

system. To remove environmental noise, we applied notch filtering to the raw data, which gave us clear and smooth force outputs (See Figure 5.4).

5.3.2 Hand actions with force dataset (HAF)

We collected a new dataset to evaluate our force estimation approach, namely the Hand Actions with Force (HAF) dataset. To solve the problem of synchronization, we asked subjects to wear on their right hand the force sensing device, leave their left hand bare, and then perform with both hands the same action, with one hand mirroring the other (see Figure 5.3(b) for the setting). We use this so called "mirrored actions" to capture synchronized data on both hands simultaneously. In this way, the hand force data and the visual data of the bare hand movements can be collected at the same time. This is not a perfect synchronization, but it is much better than doing the same action twice with the same hand. The force sensing glove captured the forces exerted by four fingers except for the little finger. We recorded from five subjects performing different manipulation actions on four objects, namely "sponge", "cup", "fork", and "knife" (See Table 5.1). Each action was performed with five repetitions, resulting in a total of 500 sample sequences.

Table 5.1: Object and Action pairs of HAF

Object	Actions
cup	drink, move, pound, pour, shake
fork	eat, poke a hole, pick, scratch, whisk
knife	chop, cut, poke a hole, scratch, spread
sponge	flip, scratch, squeeze, wash, wipe

5.4 Experiment results

This section demonstrates the ability of the proposed model to predict forces on the fingers directly from images. The preliminary model has been developed and applied on testing videos.

We first show examples of our force estimation and then report the average errors. Figure 5.5 shows six sample results. For each of the samples, the first column shows the ground truth curves, while the second column shows the estimated forces using our approach. To provide a baseline results for this experiment, we generated force curves using nearest neighbor search for each testing frame. To be specific, we used

the fc7 layer of the VGG features and reduced the dimensions to 128 using Principal Component Analysis. The results are shown in the third column. It can be seen that our system estimates well the overall force patterns for different actions. For example, for the "sponge/squeeze" action, the estimated forces correctly reproduce the three peaks of the real action, or for the "cup/move" action, the output forces predict the much smoother changes. For most of the cases, the LSTM regression method can recover more accurate dynamic patterns of the force curves, while the results of baseline method are more fluctuating.

Table 5.2 provides the average error of estimated force for each finger, and Table 5.3 gives the average estimation error for all the actions. The errors are in the range of 0.075 to 0.155, which demonstrates that the method also has good quantitative prediction and potential for visual force prediction. We also demonstrated the average errors of the nearest neighbor method. The results are shown in Table 5.2 and 5.3. It can be seen that our method yield less average errors in almost all the comparisons.

Table 5.2: Average errors of estimated force for each finger (unit in N).

Methods	Ring	Middle	Pointer	Thumb
NN	0.117	0.116	0.157	0.148
Ours	0.103	0.098	0.130	0.119

Table 5.3: Average errors of estimated force for each action (unit in N).

Cup		Drink	Move	Pound	Pour	Shake
	NN	0.121	0.145	0.176	0.121	0.152
	Ours	0.096	0.122	0.108	0.107	0.110
Fork		Eat	Hole	Pick	Scratch	Whisk
	NN	0.119	0.115	0.078	0.113	0.127
	Ours	0.106	0.090	0.075	0.094	0.100
Knife		Chop	Cut	Poke	Scratch	Spread
	NN	0.181	0.167	0.132	0.154	0.132
	Ours	0.157	0.155	0.109	0.123	0.110
Sponge		Flip	Scratch	Squeeze	Wash	Wipe
	NN	0.107	0.134	0.126	0.149	0.137
	Ours	0.101	0.130	0.112	0.127	0.121

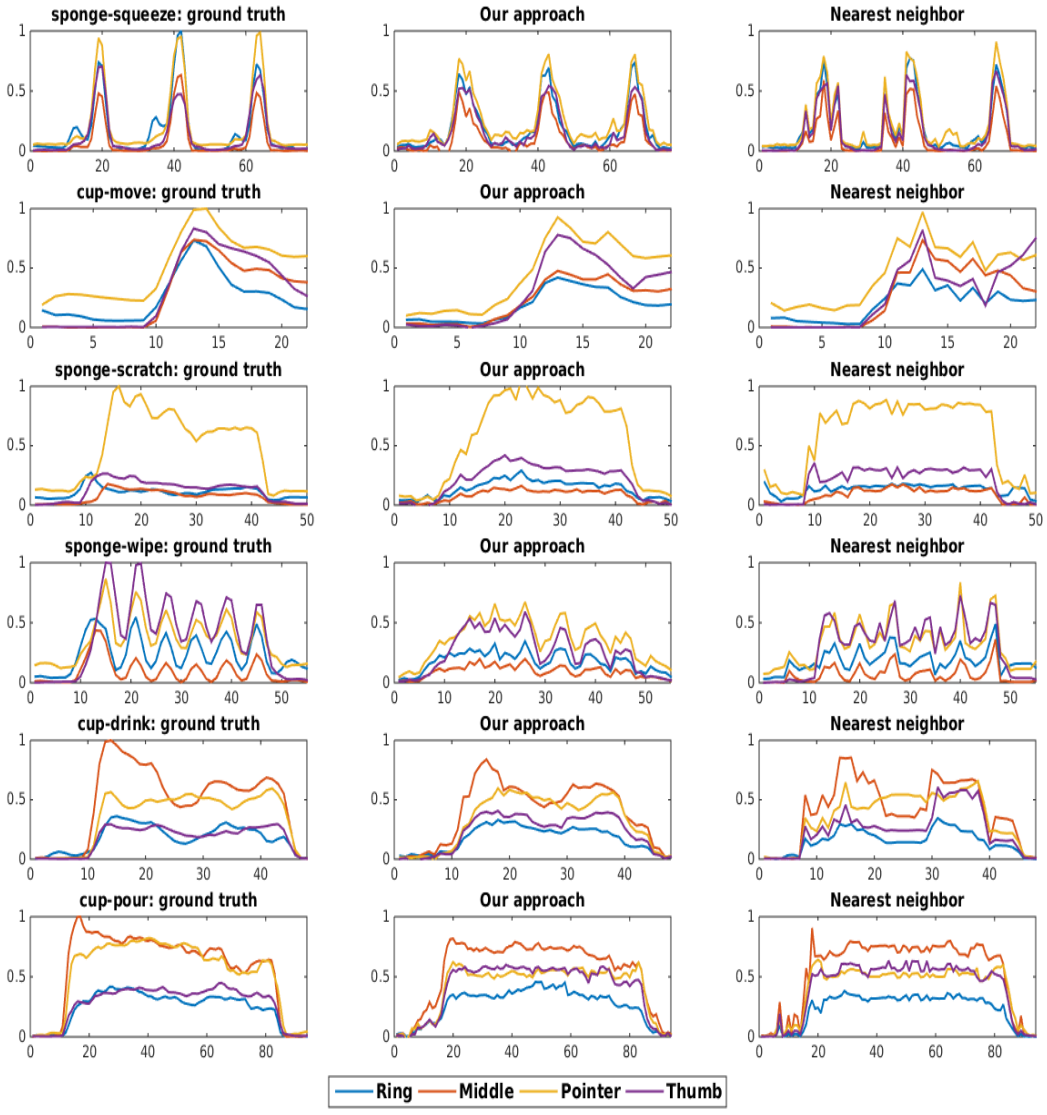


Figure 5.5: Samples of force estimation results. The first column show the ground truth force curves of six actions, the second column shows our estimation results using LSTM regressor, and the third column illustrates the results of the nearest neighbor method for comparison.

Table 5.4: Action prediction accuracy. Comparison of prediction using vision data only ("Vision") against using vision and force data ("V+F").

Object	cup	stone	sponge	spoon	knife	Avg.
Vision	82.4%	61.4%	61.6%	62.6%	73.3%	68.3%
V + F	88.2%	75.1%	59.1%	57.5%	72.7%	70.5%

5.4.1 Use forces for action prediction

To evaluate the usefulness of the predicted forces, we applied our force estimation algorithm on the MAF dataset to compute the force values. Then we used the vision data together with the regressed force values as bimodal information to train a network for action prediction. Table 5.4 shows the results of the prediction accuracy using the bimodal information on different objects. Referring to the table, the overall average accuracy for the combined vision force data (V+F) was 2.2% higher than only using vision. This first attempt on predicting with bimodal data demonstrates the potential of utilizing visually estimated forces for recognition. Future work will further elaborate on the idea and explore networks [Hoffman et al., 2016a], which can be trained to learn "hallucination" of the forces and predict actions.

As discussed in the introduction, the other advantage is that we will be able to teach robots through video demonstrations. If we can predict forces exerted by the human demonstrator and provide the force profile of the task using vision only, this would have a huge impact on the way robots learn force interaction tasks. In the future work, we plan to develop and employ sensors that can also measure the tangential forces, i.e. the frictions, on the fingers. We also will expand the sensor coverage to the whole hand. With these two improvements, our method could be applied to a range of complicated task such as screwing or assembling.

5.5 Summary

In this chapter, I proposed the prediction of force values on the hand. The methods were shown to run in real-time and demonstrated high accuracy performance. Additionally, new datasets of force measurements of dexterous actions were created. Finally, the predicted forces may be used for learning how to perform actions on the robot. Future work will attempt to map the forces from the human hands onto other actuators, for example, three-fingered hands or grippers.

Chapter 6

Manipulation Action Recognition Using Bimodal Inputs

In this chapter, I focus on utilizing additional information about the dynamics of the hand and the object along with image data to perform fine-grained action recognition. I have shown in Chapter 5 that combining the force data with visual inputs can help improving the action recognition accuracy. Now I explore the bimodal learning methods that can make use of additional information for action recognition tasks. I present a model to hallucinate motoric data from vision inputs. The proposed model is trained for recognizing actions, while at the same time generating frame-wise classification results for the whole video.

Humans can understand manipulation actions not just in terms of the movements involved, but also in terms of the objects, tools, body parts and their geometric relations. A number of works at the intersection of Computer Vision, AI, and Robotics have proposed recognition of everyday actions using representations that encode a combination of these quantities [Aksoy et al., 2011; Gupta et al., 2009; Summers-Stay et al., 2012; Zampogiannis et al., 2015]. Such representations provide excellent high-level cognitive models for everyday actions.

In addition, humans also understand such actions from the interaction of the body with the physical world in terms of kinematics and dynamics. In other words, humans represent actions in both vision and motor space. There is evidence from the field of neuroscience [Gallese and Goldman, 1998; Rizzolatti et al., 2001] of a common representation for actions and perception. The so-called mirror neurons in primates exhibit activity when they observe someone doing an action as well as when they

[1]This work is a joint work with Chinmaya Devaraj at the University of Maryland, under the supervision of Prof. Cornelia Fermüller. The experiment results reported in Section 6.3.2 and 6.3.3 were generated by Chinmaya Devaraj.

themselves perform the action. It appears that, through this mechanism, a hallucination of motoric information occurs. Inspired by these findings, we propose that when representing manipulation actions, it is beneficial to also encode the kinematics and dynamics of the person's hand and/or the object being moved.

Why would motoric information be useful for helping recognize actions? Intuitively, there are two main reasons. First, the large variations in appearance in actions in general, and manipulation actions specifically pose a great challenge for visual recognition. Motoric data does not depend on the visual nuisances, for example, the appearance changes due to lighting, texture, and choice of viewpoint. Thus, motoric data should be valuable complimentary information to vision. Second, motoric data is of much lower dimension, which should help in extracting the relevant subspaces in visual information, even if only a small amount of training data is available.

Some previous works have used motoric data, but the bi-modal data was analyzed separately [Lea et al., 2016a,e]. Lea et al. [2012] reported the accuracy of both vision-based and sensor-based input but used different tools for analyzing the two sources of data. Rupprecht et al. [2016] predicted sensor values from videos, which were then used for predicting the action labels. However, these works require the motoric data to be regressed very well in order for it be useful for recognizing the actions. Otherwise, the error would propagate during the prediction of actions.

I consider the motoric data available only during training, as "privileged information," using the terminology in [Vapnik and Vashist, 2009]. In the Support Vector Machine framework, this privileged information has been exploited to improve the performance and reduce the amount of training data necessary [Vapnik and Vashist, 2009]. Srivastava and Salakhutdinov [2014] proposed deep Boltzmann machines that exploit multi-modal data even when some modalities are absent. This idea also relates to the hallucination network presented in [Hoffman et al., 2016b], where depth input is hallucinated in single images to improve object detection.

I present a model based on the Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) model (shown in Figure 6.1) for fine-grained manipulation action recognition and segmentation that implements this idea. The network is trained on two modalities, videos and kinematic/dynamic data, to predict action labels. During testing, we use only video data. Our model learns a mapping from the image inputs to the representation of kinematic/dynamic data, and this mapping is used jointly with videos to recognize actions.

Specifically, in order to learn the mapping from images to the kinematic/dynamic

data, we first construct two network structures to learn the representations of vision input and motoric input respectively. Then, we additionally construct a hallucination network that replicates the vision network structure and use a loss function to connect it to the motoric network, which aims at minimizing the difference of the hallucinated representations and the motoric representations. With this method, we are successful in capturing the additional information available from the motoric data.

One limitation of the approach used in [Mahasseni and Todorovic, 2016] is that regularization can only be helpful for short video sequences which are already segmented into actions. Our approach is generalizable even for longer videos. Our method also handles asynchronous data naturally.

We validate our idea on two multi-modal fine-grained action datasets, namely, the 50 SALAD [Stein and McKenna, 2013b] dataset and Hand Action with Force (HAF) dataset collected in Chapter 5. Each of these datasets captures subtle differences between the manipulation actions. We conduct experiments on them to show that our network outperforms vision-only approaches.

6.1 Related work

Fine-grained manipulation actions Fine-grained action recognition and detection is an important vision task, especially as a component of imitation learning in robotics. Unlike in whole-body action recognition, where the categories are quite different visually, fine-grained action recognition involves discerning the actions when there are very subtle changes in the surroundings. In [Jain et al., 2015], they show that identifying objects plays an important role in action recognition. Previous works on fine-grained action recognition [Rohrbach et al., 2015; Ni et al., 2014] have used precise hand trajectories to aid action recognition. Hierarchal temporal convolutional networks have been used in this task for detection and segmentation [Lea et al., 2016a,e]. There is also work using motoric data [Lea et al., 2016d]. The novelty of our approach lies in the integration of visual and motoric data and the demonstration of the induced recognition performance gains.

Learning in perception with additional information The fusion of multi-modal data in networks has been successfully demonstrated in various applications, including object recognition [Eitel et al., 2015], object detection and segmentation [Gupta et al., 2014], and activity recognition [Song et al., 2016]. However, it is often challeng-

ing to collect more than two modalities of data for the same activity. Thus, the idea has arisen to collect two modalities of data in controlled situations and use both for training, while only one modality is available for testing. Vapnik and Vashist [2009] call the second modality of data "privileged information" since it is present only during the training stage. In related work, Hoffman et al. [2016b] have shown that it is possible to hallucinate the depth information from RGB data and thereby improve detection in RGB. Their work processes single images using the hallucination with a CNN. Mahasseni and Todorovic [2016] used motion capture data to improve action recognition from RGB data, but they do not use a hallucination mechanism. Moreover, their approach is not applicable to the action segmentation problem. We use a hallucination network in the temporal domain: we transfer information from motoric data to improve action recognition from RGB data, and we also achieve temporal segmentation.

Action recognition using deep learning Deep learning methods for action recognition usually use either temporal CNN or Recurrent Neural Networks (RNN). The former learns spatial-temporal filters over raw image sequences [Ji et al., 2013; Karpathy et al., 2014] or optical flows [Simonyan and Zisserman, 2014a]. The RNN models, which have temporal recurrent loops, can capture compositional representations in the time domain. The LSTM model was proposed as a variation of the RNN model to overcome the so-called "vanishing gradient" problem [Hochreiter and Schmidhuber, 1997]. Recently, it has been widely used in image description generation [Vinyals et al., 2015], video captioning [Venugopalan et al., 2014], and action recognition tasks [Donahue et al., 2015; Ng et al., 2015]. In our work, we adopt the LSTM model for learning on vision data as well as kinematic/dynamic data.

6.2 Our approach

In this section, we describe our deep network structure that uses CNN model and the LSTM architecture to learn motoric data representations from video data.

6.2.1 Motivation

The goal of our approach is to create a hallucination network, which takes the same visual input as the vision network but is different in the sense that it is trained using the information from the motoric data.

To achieve this goal, we replicate the vision network as the hallucination network and re-train it by enforcing its output to be similar to the motoric data network. By accomplishing this, we should be successful in capturing the additional knowledge from motoric data. In the following subsections, we explain the architecture and optimization details.

6.2.2 Network architecture

The architecture we used for training is shown in Figure 6.1. Our network has three components: the vision network, the motoric network, and the hallucination network. For both, the vision network and the hallucination network, we use a CNN to extract features from video frames. For the motoric inputs, we use a linear projection to generate features. Then both the CNN features and motoric features are fed into LSTM models to get the dynamic representations after temporal convolution and pooling.

In this architecture, the hallucination network learns the relationship between the two modalities. It simulates the motoric network during the training stage. To achieve this, we use an L_2 loss function to minimize the difference between the motoric representation and the hallucinated outputs. This way, the hallucination network generates representations in the higher layers similar to the motoric network. It can thus be used to replace the motoric network when only video input is available. The three networks are jointly trained. During the testing phase, only the vision and hallucination networks are used.

6.2.2.1 Vision network

The vision network is used to represent both the temporal and spatial aspects of the action. It consists of a convolutional network similar to the VGG architecture Simonyan and Zisserman [2014b] followed by an LSTM cell. The convolutional network is the same for all layers prior to the FC7 layer in VGG Simonyan and Zisserman [2014b]. Instead of the 4096 dimensional fully connected layer in fc7, we use a 512-dimensional vector, and this choice is based on empirical studies.

Long term dependencies in the input video are captured well by the LSTM. However, for manipulation actions, there is a lot of inter-subject variances, which also has to be eliminated, before feeding the input to the LSTM. We use the idea of temporal convolution followed by pooling to remove these inter-subject variations.

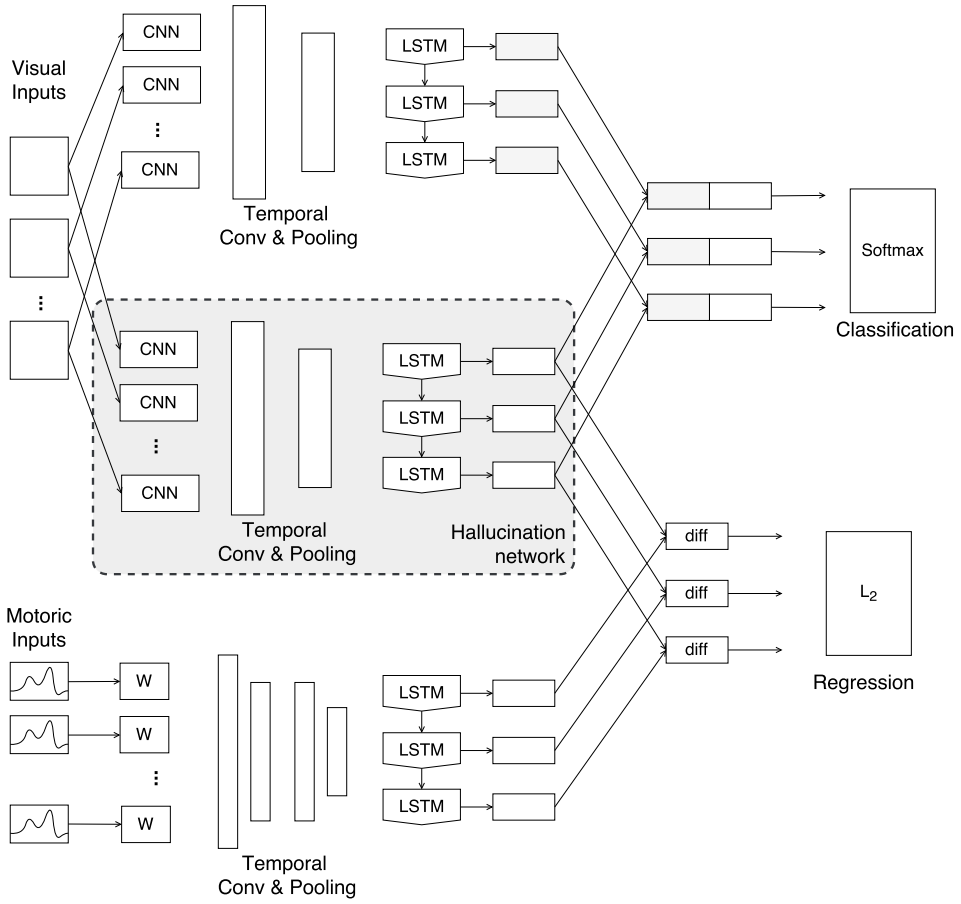


Figure 6.1: The network structure of the hallucination network. The vision network and the hallucination network share the same network structures and take the same inputs. A motoric network is constructed to learn the representations of motoric inputs. A L_2 loss function is applied to the outputs of the hallucination network and the motoric network to achieve "hallucination" of the motoric information. The learned representations of visual network and the hallucination network are combined and used for action classification.

In our implementation, we use a filter of size (4×1) and convolve it across the frames. We then use max pooling with a stride of (4×1). We repeat these temporal convolution and max pooling steps to achieve downsampling by a factor of 16. This makes our model resilient to small variations in the action so that mostly long term dependencies can be captured by the LSTM. The temporal convolution and pooling steps are followed by an embedding layer of dimension 128. We use 128 hidden nodes in the LSTM. A fully connected layer of 128 dimensions precedes the LSTM cell.

6.2.2.2 Hallucination network

The hallucination network is similar in structure to the vision network, except that the final fully connected layer has the same dimension as the motoric representation layer.

This network is designed to bridge the two modalities by accepting video inputs and approximating the motoric representations. One benefit of using the same network structure is that we can use the same CNN model for both the vision and the hallucination network. In our experiment, we find that using the same fixed CNN model does not harm the performance and significantly reduces the computational cost. That being said, the proposed architecture allows more flexible network structures. The hallucination network is then jointly trained with the video and motoric inputs.

6.2.2.3 Motoric network

The motoric network consists of a fully-connected layer followed by temporal convolution and max-pooling layers as shown in Figure 6.1. This is followed by another set of temporal convolution and max-pooling layers. These layers are similar to the vision model in terms of detail. This, in turn, downsamples the motoric data by a factor of 16. An embedding layer of 128 dimensions and an LSTM cell with 128 hidden nodes precedes it. Finally, a fully-connected layer of dimensionality 128 is present after the LSTM cell.

All three networks are followed by a softmax layer. Since the kinematic/accelerometer data is already in a lower dimensional space, we did not use a deeper network and a single fully connected layer is present before the LSTM cell to address the above issue.

6.2.3 Training

Action representation To perform action recognition, we concatenate the vision and hallucination network representations to get the action representation. During the training stage, the hallucination network approximates the representation of the motoric input, while at the testing stage, the output of the hallucination network represents the estimated motoric information that is encoded in the model.

Mirror constraints To force the hallucination network to yield similar representations as the motoric network, we apply a L_2 loss on the two outputs as follows:

$$l_{dis} = \|u_{mo} - u_{mi}\|_2^2, \tag{6.1}$$

where u_{mo} and u_{mi} are the outputs of the motoric and the hallucination networks, respectively.

Loss function We apply the cross-entropy loss to the action representation for the purpose of training the recognition model. To further regularize the training, we also add the cross-entropy loss to each of the sub-networks. Combined with the hallucination constraints defined in Equation (6.1), we present the general form of our loss function as follows:

$$Loss = \alpha * l_v + \beta * l_{mi} + \gamma * l_{mo} + \zeta * l_{vm} + \eta * l_{dis} \tag{6.2}$$

where l_v, l_{mi}, and l_{mo} in Equation (6.2) refer to the cross entropy softmax loss of the vision network, the hallucination network, and the motoric network, respectively. l_{vm} in Equation (6.2) refers to the cross entropy softmax loss of the concatenated output of the vision and motoric networks. l_{dis} in Equation (6.2) represents the L_2 loss between the FC layer output of the motoric network and that of the hallucination network. Constants α, β, γ, ζ, and η are fixed empirically after running the program for iterations. Since the L_2 loss is of high magnitude, we modify η to an appropriate value such that the contribution of l_{dis} towards the overall loss function is around 50%.

Model pre-training Training all the three networks jointly with all the weights randomly initialized would be unstable. Thus, we first pre-train the vision network and motoric networks separately to predict frame-wise action class labels. To train the

vision network, we initially fine-tuned the modified VGG network described in Section 6.2.2 using the Caffe framework Jia et al. [2014]. We used a base learning rate of 0.001, a momentum value of 0.9 and a weight decay value of 0.0005.

Joint training The pre-trained parameters are then used to train the complete architecture using the Theano framework Theano Development Team [2016]. To reduce over-fitting, we add dropout layers after each temporal convolution layer and the final fully connected layers. We find that the dimensionality of the embedding layers is critical for effective training. In practice, we use 128 for all three networks since this yields the best results.

6.3 Experiments

We used two datasets to evaluate the proposed method: *1)* the 50 SALADS dataset which has been described in Section 4.5.2; *2)* the Hand Action with Force (HAF) dataset introduced in Section 5.3.2. Subsequent subsections describe the experiment results.

6.3.1 Datasets

Both datasets capture fine-motor activity and the differences between the featured manipulation actions are subtle. Figure 6.2 and 6.3 shows a few samples from each of the datasets. We evaluate action recognition on the 50 SALADS dataset using the metric of per-frame accuracy. The HAF dataset has only one action per video, thus we can only use it for evaluation of action classification.

Different modalities in most of the publicly available datasets are usually synchronized, and most state-of-the-art methods cannot handle asynchronous data. However, going beyond just recognition, we evaluate our method's performance when the video data and force data are *asynchronous*. Our method can accommodate asynchronous data by using dynamic time warping to resynchronize the streams.

6.3.2 Results on 50 SALADS

We present our results on 50 SALADS dataset in this sections. Table 6.1 shows the per-frame accuracy on this dataset. We compared our approach to "TCN" and "ST-CNN+Seg", which are two state-of-the-art methods applied to this dataset.

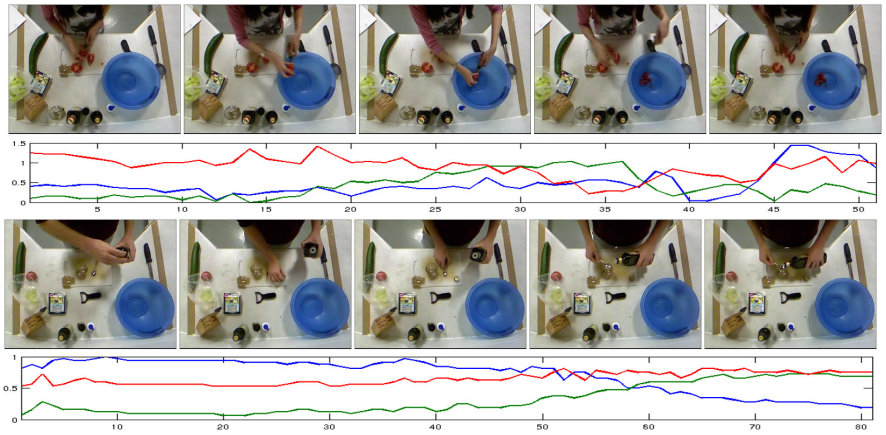


Figure 6.2: Sample actions of the 50 salads dataset. The first sequence shows an example of a "cut" action. The curves below show the 3-axis accelerometer data recorded from a sensor attached to a knife (blue, green, and red are the signal along the x, y, and z axis, respectively). The second sequence shows an example of the "add oil" action and the curves show the accelerometer data from the sensor attached to the bottle.

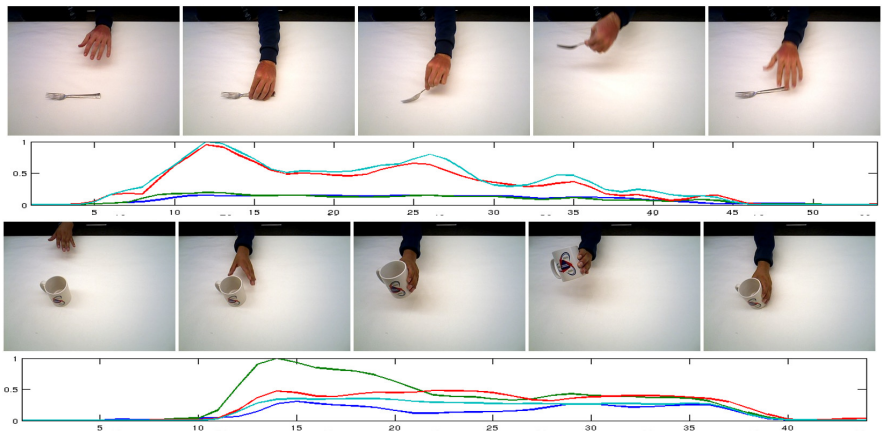


Figure 6.3: Sample actions of the HAF dataset. The first sequence shows an example of an "eat" action using "fork" and the second sequence shows a "drink" action using "cup". The curves below show the force values of four finger tips (blue, green, red, and cyan indicate thumb, index, middle, and ring finger, respectively).

First, we compare the performance using different modalities in our framework. In this dataset, the accelerometer data is available. If we use only the direct measurement of movement, our method achieved an accuracy of 71.57%. Alternatively, using only visual input presented an accuracy of 72.09%, which is slightly higher than using motoric input. However, if we use both modalities in training, and test on the visual data using both the vision network and hallucination network, we achieved a higher accuracy of 74.02%.

Compared to "TCN" and "ST-CNN+Seg", our network structure for testing is simpler and more intuitive, while still achieving the best performance measures. Specifically, our vision-only network has an inferior performance compared to "TCN" while achieving a performance comparable to that achieved by "ST-CNN+Seg". With the hallucination mechanism employed using the accelerometer data, our method outperformed both methods. It must be noted that in this case, our method using accelerometers only had an inferior performance compared to both methods. Therefore, we conclude that our hallucination network structure is effective in borrowing useful information from one modality and improving the performance of the other, which suggests that knowledge of motoric information about actions can provide rich and useful information for perception tasks.

Algorithm	Accuracy	Edit Score
ST-CNN + Seg[Lea et al., 2016b]	72%	62.06
TCN [Lea et al., 2016a]	73.4%	72.2
Vision Only (Ours)	72.09%	77.77
Accelerometer Only (Ours)	71.57%	75.42
Vision+Hallucination (Ours)	74.02%	72.659

Table 6.1: Results for the 50 SALADS dataset.

6.3.3 Results on HAF dataset

This subsection presents results on the HAF dataset. In this section, our focus is on demonstrating that our method can handle asynchronous data.

First, we studied the results for synchronized data. Possibly due to differences in input sensors, the results for this dataset is qualitatively different from that achieved by the 50 SALADS dataset in the sense that the performance of the vision modality is much higher than the sensor modality (80%, as compared to 62.3%). Nevertheless, similar to the 50 SALADS dataset, Table 6.2 shows that the statistical result for the hallucination network shows a noticeable improvement (81.5%) over the vision-only

modality.

The performance increase is notable because the "Force only" performance by itself is very poor (Table 6.2). Nevertheless, by "combining" the two inputs directly, we conclude that our hallucination network structure is able to successfully facilitate this information fusion during training, and is able to use the complementary information correctly.

In this dataset, we further study the effect of asynchronous data. Although the HAND dataset contains synchronous bimodal data, asynchronous data can be generated by interchanging the motoric data of one subject with that of another subject performing the same set of actions. In order to apply our method to asynchronous video and motoric data, we need to synchronize the signals. To account for timing differences, we time-normalize all sequences of a given action class to the same duration. To achieve this, we leverage the fact that video and motoric signals are synchronized *per execution instance* and use Dynamic Time Warping to temporally align the motoric signals to a reference execution profile. This is done in a preprocessing stage; the aligned sequences are then used for training. This process gives significant improvements in the "Force only" and "Vision + Hallucination' scenarios (Table 6.2).

Object	Sponge	Cup	Fork	Knife	Average
Vision only	81	78	80	81	80
Force only	84	59	60	46	62.3
Asynchronous	83.5	64.3	59.5	52.2	64.9
Vision + Hallucination	87	78	81	80	81.5
Hallucination (asynchronous)	91	76	78	89	83.8

Table 6.2: Results for the HAF dataset (shown in percentage).

6.4 Summary

In this chapter, we studied the problem of incorporating knowledge about motoric information to improve visual perception of manipulation actions. We provided an approach for visual recognition and segmentation of fine-grained manipulation actions based on a recurrent neural network architecture. During training, a hallucination structure is learned from visual and motoric data, and this mirroring structure helps recognition during the testing phase when only visual data is present. We validated our method on two multi-modal fine-grained action datasets and showed that the network outperforms vision-only approaches.

Chapter 7

Conclusion

In this thesis, I studied the representation learning approaches using deep neural networks. Two types of input data were studied: static inputs and sequential inputs. To explore the different perspectives of the representation learning approaches, four case studies were investigated. This chapter summarizes the proposed approaches and our contributions and discusses several future directions to study.

7.1 Summary and contributions

My first study was the neural network approaches for the representation learning and the similarity learning of static inputs. In Chapter 3, I chose to study the sketch based 3D shape retrieval problem by learning the similarity of sketch images using Siamese networks. My method directly learns the feature representations of hand drawn sketches, which bypasses the dilemma of best view selection of 3D shapes and drastically reduced the number of views to two from several dozens. I adopted the Siamese network and extend the architecture to use two identical convolutional neural networks for similarity learning of cross domains. The proposed model are trained directly on sampled pairs, which is more efficient than training on individual samples. The experiment results on three large-scale datasets show that our approach outperformed previous methods significantly.

After studying the representation learning methods of static inputs, I turn to focus on learning of sequential data. From Chapter 4 to Chapter 6, I studied three closely related tasks around the human action recognition problem and explored the neural network approaches for representation learning of dynamic information.

In Chapter 4, I adopt the recurrent neural network and the long short-term memory model to learn representations of human actions. In contrast to most existing methods that treat the action recognition as a classification task on pre-segmented

video inputs, we predict the intended actions and dynamically change the predicted category while watching the body and hand movements. This schema allows fast response of a robotic system in human-robot collaboration scenarios. This is the first computational study on the prediction of manipulation actions. I demonstrated that predicting the intended actions by watching the movement at an early stage is possible. A new dataset of manipulation actions (MAD) for evaluation of our action prediction model was also collected.

In Chapter 5, I turned the focus from action prediction to the estimation of other dynamic data, for example, the hand forces, that are closely related to the actions. We adopted the long short-term memory model for regression and learn to estimate the forces of fingertips from video inputs. The experiment results demonstrated that our model can effectively learn the force patterns from visual features and the generated force curves are well approximated the ground truth values. Further experiments of combining forces with visual inputs showed that the estimated forces can help improve the action recognition accuracy. The contribution is twofold. First, I proposed an efficient force estimation approach that generates forces from 2D video frames. Second, we collected a new dataset with synchronized videos and the hand forces as a test bed for force estimations of manipulation actions (HAF).

At last, in Chapter 6, I investigated the methods to use additional data to improve action recognition performance. I demonstrated that, by combining the visual features with additional data such as hand forces and accelerometer data, the manipulation action recognition can be more accurate. To eliminate the dependency of the additional information during testing, we built a hallucination model to learn the relationships between the visual input and the additional data. The hallucination network learns to approximate the representations of the additional data using visual inputs, which can be used in the testing stage to generate the final action representations. With the hallucination model, our approach achieves better performance on manipulation action recognition tasks.

7.2 Future works

The research presented in this thesis demonstrate the representation learning approaches using deep neural networks with several case studies. A number of opportunities for extending the scope of this thesis should be pursued. This section discusses some of these future directions.

General cross-domain similarity learning methods I presented a cross-domain similarity learning architecture in Chapter 3, which can successfully learn similarities of hand drawn sketches and 2D line renderings of 3D models. In this case, the two domains share several characteristics. For example, they are both 2D images and mainly contain black line strokes in a white background. In more general scenarios, one need to study similarity learning methods for inputs from significantly different domains. A possible research direction is to learn the similarities of hand drawn sketches and 3D models directly or to learn the relationships of sketches and natural images. For these domains, the low level features differ significantly from each other, so the learning techniques need to take these disparities into consideration to achieve good performance. To push this line further, the cross-domain feature matching has many uses in vision and language tasks, such as image captioning and question/answering tasks.

Representation learning for human actions in complex scenarios I explored the representation learning methods of manipulation actions in Chapter 4 and 6. However, it is remaining a challenging task to learn representations of human actions with cluttered backgrounds. In complex scenarios, tracking the hand is much more difficult in the first place. When multiple tools and objects are existing in the context, one also need to consider the interactions between different tools or objects to understand the action. Deep neural networks have shown to be potentially capable of learning good representations in complex scenarios. Using spatial-temporal feature extraction with learned features using deep neural networks is a promising direction for future research.

Completing the robot control loop In Chapter 4, I have stressed the importance of the prediction ability for a proactive system. A natural direction of future work is to apply the prediction approaches in such systems and make the robots take responding actions accordingly. With the action prediction module, one can complete the robot control loop and build robotic systems that can accomplish human-robot collaboration tasks. There are a number of open questions raised by this task, for example, how to handle relative changing positions between the robot and the human partner, recognition in cluttered environments, and how to manipulate a real object with the robot grippers.

Fast multi-modal learning in robotic system I studied the problem of bimodal learning of manipulation actions in Chapter 6, which suggests the idea of using multiple dynamic data in robotic systems. It is common that a robot is integrated with many different kinds of sensors, which can collect multi-modal information simultaneously. How to effectively use these data to improve the performance of perception tasks is an important question.

7.3 Summary

In this thesis, I presented several case studies about the representation learning approaches using deep neural networks. For representation learning static inputs, I designed a Siamese network to learn shape representations as well as cross-domain similarities for the application of sketch based 3D shape retrieval. I showed that the learned representations with simple nearest neighbor retrieval method significantly outperforms existing methods that using hand crafted features. For representation learning of sequential data, I developed an LSTM model to learn the action representations by adapting the long short-term memory model. Two similar architectures are used for manipulation action prediction as well as regression of hand force patterns. My results showed that the combination of convolutional neural networks and the long short-term memory model provide an effective way to learn action representations. Finally, a hallucination network was adopted for bimodal learning of human actions, which is shown to be beneficial for the action recognition accuracy.

Bibliography

Aksoy, E.; Abramov, A.; Dörr, J.; Ning, K.; Dellen, B.; and Wörgötter, F., 2011. Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research*, 30, 10 (2011), 1229–1249. (cited on page 73)

Aloimonos, Y. and Fermüller, C., 2015. The cognitive dialogue: A new model for vision implementing common sense reasoning. *Image and Vision Computing*, 35, 12 (2015), 2891–2903. (cited on page 42)

Ansuini, C.; Cavallo, A.; Bertone, C.; and Becchio, C., 2015. Intentions in the Brain: The Unveiling of Mister Hyde. *The Neuroscientist*, 21, 2 (2015), 126–135. (cited on page 43)

Ansuini, C.; Giosa, L.; Turella, L.; Altoé, G.; and Castiello, U., 2008. An object for an action, the same object for other actions: effects on hand shaping. *Exp. Brain Research*, 185, 1 (2008), 111–119. (cited on page 43)

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B., 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57, 5 (2009), 469–483. (cited on page 62)

Aviles, A. I.; Marban1a, A.; Sobrevilla, P.; Fernandez, J.; and Casals, A., 2015. A recurrent neural network approach for 3d vision-based force estimation. In *aCoRR, abs/1505.00393*. (cited on page 46)

Belongie, S.; Malik, J.; and Puzicha, J., 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24, 4 (Apr. 2002), 509–522. (cited on page 24)

Bertinetto, L.; Valmadre, J.; Henriques, J. F.; Vedaldi, A.; and Torr, P. H., 2016. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, 850–865. Springer. (cited on page 18)

Bousquet, O. and Bottou, L., 2008. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, 161–168. (cited on page 16)

Bradski, G. R., 1998. Computer vision face tracking for use in a perceptual user interface. (1998). (cited on page 48)

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R., 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7, 04 (1993), 669–688. (cited on page 17)

Bronstein, A. M.; Bronstein, M. M.; Guibas, L. J.; and Ovsjanikov, M., 2011. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph.*, 30, 1 (Feb. 2011), 1:1–1:20. doi:10.1145/1899404.1899405. http://doi.acm.org/10.1145/1899404.1899405. (cited on page 24)

Bullock, I. M.; Feix, T.; and Dollar, A. M., 2015. The Yale Human Grasping Data Set: Grasp, Object, and Task Data in Household and Machine Shop Environments. *International Journal of Robotics Research*, (2015). (cited on page 46)

Cai, M.; Kitani, K. M.; and Sato, Y., 2015. A scalable approach for understanding the visual structures of hand grasps. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 1360–1366. IEEE. (cited on page 46)

Chen, K. and Salman, A., 2011. Extracting speaker-specific information with a regularized siamese deep network. In *NIPS 2011* (Eds. J. Shawe-Taylor; R. Zemel; P. Bartlett; F. Pereira; and K. Weinberger), 298–306. http://papers.nips.cc/paper/4314-extracting-speaker-specific-information-with-a-regularized-siamese-deep-network.pdf. (cited on page 18)

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, (2014). (cited on page 15)

Chopra, S.; Hadsell, R.; and LeCun, Y., 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR 2005*, vol. 1, 539–546. IEEE. (cited on pages 5, 17, and 23)

Comaniciu, D.; Ramesh, V.; and Meer, P., 2000. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 142–149. IEEE. (cited on page 53)

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine learning*, 20, 3 (1995), 273–297. (cited on page 1)

Cox, D. R., 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, (1958), 215–242. (cited on page 1)

Crajé, C.; Lukos, J.; Ansuini, C.; Gordon, A.; and Santello, M., 2011. The effects of task and content on digit placement on a bottle. *Exp. Brain Research*, 212, 1 (2011), 119–124. (cited on page 43)

Cutkosky, M. R., 1989. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5, 3 (1989), 269–279. (cited on page 46)

Daras, P. and Axenopoulos, A., 2010. A 3d shape retrieval framework supporting multimodal queries. *International Journal of Computer Vision*, 89, 2-3 (2010), 229–247. (cited on page 24)

DeCarlo, D.; Finkelstein, A.; Rusinkiewicz, S.; and Santella, A., 2003. Suggestive contours for conveying shape. *ACM Trans on Graphics*, 22, 3 (Jul. 2003), 848–855. (cited on page 29)

DiCarlo, J. J.; Pinto, N.; and Cox, D. D., 2008. Why is real-world visual object recognition hard? (2008). (cited on page 15)

Dollár, P.; Rabaud, V.; Cottrell, G.; and Belongie, S., 2005. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 65–72. IEEE, San Diego, CA. (cited on pages 6 and 46)

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2625–2634. (cited on pages vii, 14, 18, and 76)

Donoho, D. L. and Elad, M., 2003. Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell 1$ minimization. *Proceedings of the National Academy of Sciences*, 100, 5 (2003), 2197–2202. (cited on page 2)

Duchi, J.; Hazan, E.; and Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, Jul (2011), 2121–2159. (cited on page 16)

Eitel, A.; Springenberg, J. T.; Spinello, L.; Riedmiller, M. A.; and Burgard, W., 2015. Multimodal deep learning for robust RGB-D object recognition. *CoRR*, abs/1507.06821 (2015). http://arxiv.org/abs/1507.06821. (cited on page 75)

Eitz, M.; Hays, J.; and Alexa, M., 2012a. How do humans sketch objects? *ACM Trans. on Graphics*, 31, 4 (2012), 44:1–44:10. (cited on page 24)

Eitz, M.; Richter, R.; Boubekeur, T.; Hildebrand, K.; and Alexa, M., 2012b. Sketch-based shape retrieval. *ACM Trans. Graphics*, 31, 4 (2012), 31:1–31:10. (cited on pages 21, 22, 24, 30, and 32)

Erol, A.; Bebis, G.; Nicolescu, M.; Boyle, R. D.; and Twombly, X., 2007. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108, 1 (2007), 52–73. (cited on page 42)

Fanello, S. R.; Gori, I.; Metta, G.; and Odone, F., 2013. Keep it simple and sparse: Real-time action recognition. *The Journal of Machine Learning Research*, 14, 1 (2013), 2617–2640. (cited on page 47)

Fathi, A.; Ren, X.; and Rehg, J. M., 2011. Learning to recognize objects in egocentric activities. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference On*, 3281–3288. (cited on page 46)

Feix, T.; Pawlik, R.; Schmiedmayer, H.; Romero, J.; and Kragic, D., 2009. A comprehensive grasp taxonomy. In *Robotics, Science and Systems Conference: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*. (cited on page 46)

Fouhey, D. F. and Zitnick, C., 2014. Predicting object dynamics in scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on page 45)

Fragkiadaki, K.; Levine, S.; Felsen, P.; and Malik, J., 2015. Recurrent network models for kinematic tracking. In *arXiv:1508.00271*. (cited on page 15)

Funkhouser, T.; Min, P.; Kazhdan, M.; Chen, J.; Halderman, A.; Dobkin, D.; and Jacobs, D., 2003. A search engine for 3D models. *ACM Transactions on Graphics*, 22, 1 (Jan. 2003), 83–105. (cited on page 24)

Funkhouser, T. and Shilane, P., 2006. Partial matching of 3d shapes with priority-driven search. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy, 2006), 131–142. Eurographics Association. http://dl.acm.org/citation.cfm?id=1281957.1281974. (cited on page 24)

Furuya, T. and Ohbuchi, R., 2009. Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In *Proceedings of the ACM international conference on image and video retrieval*, 26. ACM. (cited on pages 21 and 24)

Furuya, T. and Ohbuchi, R., 2013. Ranking on cross-domain manifold for sketch-based 3d model retrieval. In *International Conference on Cyberworlds 2013*, 274–281. IEEE. (cited on pages 21, 37, and 38)

Gallese, V. and Goldman, A., 1998. Mirror neurons and the simulation theory of mind-reading. *Trends in cognitive sciences*, 2, 12 (1998), 493–501. (cited on page 73)

Gallesse, V. and Goldman, A., 1998. Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, 2, 12 (1998), 493–501. (cited on pages 61 and 62)

Gams, A.; Do, M.; Ude, A.; Asfour, T.; and Dillmann, R., 2010. On-line periodic movement and force-profile learning for adaptation to new surfaces. In *IEEE International Conference on Robotics Research (ICRA)*, pp.3192–3199. IEEE. (cited on page 62)

Gong, B.; Liu, J.; Wang, X.; and Tang, X., 2013. Learning semantic signatures for 3d object retrieval. *Multimedia, IEEE Transactions on*, 15, 2 (2013), 369–377. (cited on page 24)

Graves, A.; Mohamed, A.; and Hinton, G. E., 2013. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778 (2013). http://arxiv.org/abs/1303.5778. (cited on page 14)

Greminger, M. and Nelson, B., 2004. Vision-based force measurement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 3 (2004), 290–298. (cited on page 46)

Gupta, A. and Davis, L. S., 2008. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *European Conference of Computer Vision (ECCV)*, 26–29. (cited on page 42)

Gupta, A.; Kembhavi, A.; and Davis, L. S., 2009. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31, 10 (2009), 1775–1789. (cited on page 73)

Gupta, S.; Girshick, R. B.; Arbelaez, P.; and Malik, J., 2014. Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736 (2014). http://arxiv.org/abs/1407.5736. (cited on page 75)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034. (cited on page 15)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 15)

Hinton, G., 2012. rmsprop: Divide the gradient by a running average of its recent magnitude. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. (cited on page 16)

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, (2012). (cited on page 15)

Hoai, M. and De la Torre, F., 2014. Max-margin early event detectors. *International Journal of Computer Vision*, 107, 2 (2014), 191–202. (cited on page 48)

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9, 8 (1997), 1735–1780. (cited on pages 14 and 76)

Hoffman, J.; Gupta, S.; and Darrell, T., 2016a. Learning with side information through modality hallucination. In *In Proc. Computer Vision and Pattern Recognition (CVPR)*. (cited on page 72)

Hoffman, J.; Gupta, S.; and Darrell, T., 2016b. Learning with side information through modality hallucination. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 826–834. doi: 10.1109/CVPR.2016.96. http://dx.doi.org/10.1109/CVPR.2016.96. (cited on pages 74 and 76)

Holyoak, K. J., 1987. Parallel distributed processing: explorations in the microstructure of cognition. *Science*, 236 (1987), 992–997. (cited on page 2)

Ijina, E. and Mohan, S., 2014. Human action recognition based on recognition of linear patterns in action bank features using convolutional neural networks. In *International Conference on Machine Learning and Applications*. (cited on page 47)

ILSVRC, 2016. Large scale visual recognition challenge 2016. http://image-net.org/challenges/LSVRC/2016/results. Accessed: 2016-12-19. (cited on page 5)

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 448–456. (cited on page 15)

Jain, M.; van Gemert, J. C.; and Snoek, C. G. M., 2015. What do 15,000 object categories tell us about classifying and localizing actions? In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 46–55. doi:10.1109/CVPR.2015.7298599. (cited on page 75)

Jarrett, K.; Kavukcuoglu, K.; LeCun, Y.; et al., 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, 2146–2153. IEEE. (cited on page 15)

Jeannerod, M., 1984. The timing of natural prehension movements. *Journal of motor behavior*, 16, 3 (1984), 235–254. (cited on page 43)

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 76)

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, (2014). (cited on page 81)

Joo, J.; Li, W.; Steen, F. F.; and Zhu, S.-C., 2014. Visual persuasion: Inferring communicative intents of images. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 216–223. IEEE. (cited on page 45)

Karpathy, A. and Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*. (cited on page 18)

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1725–1732. (cited on pages 47 and 76)

Kazhdan, M. M.; Chazelle, B.; Dobkin, D. P.; Finkelstein, A.; and Funkhouser, T. A., 2002. A reflective symmetry descriptor. ECCV '02, 642–656. Springer-Verlag, London, UK, UK. http://dl.acm.org/citation.cfm?id=645316.649205. (cited on page 24)

Keskin, C.; Kiraç, F.; Kara, Y. E.; and Akarun, L., 2013. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, 119–137. Springer. (cited on page 43)

Kingma, D. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014). (cited on page 16)

Kitani, K. M.; Ziebart, B. D.; Bagnell, J. A.; and Hebert, M., 2012. Activity forecasting. In *European Conference on Computer Vision (ECCV)*. (cited on page 45)

Klaser, A.; Marszałek, M.; and Schmid, C., 2008. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, 275–1. British Machine Vision Association. (cited on pages 6 and 46)

Kober, J.; Gienger, M.; and Steil, J., 2000. Learning movement primitives for force interaction tasks. In *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 142–149. (cited on page 62)

Koppula, H. 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). (cited on page 45)

Kormushev, P.; Calinon, S.; and Caldwell., D. G., 2011. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25, 5 (2011), 581–603. (cited on page 62)

Krizhevsky, A.; Sutskever, I.; and Hinton, G., 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. (cited on pages 12 and 15)

Kulkarni, G.; Premraj, V.; Ordonez, V.; Dhar, S.; Li, S.; Choi, Y.; Berg, A. C.; and Berg, T., 2013. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34 (2013), 42–44. (cited on page 42)

Kyriazis, N. and Argyros, A., 2013. Physically plausible 3d scene tracking: The single actor hypothesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9–16. (cited on page 64)

Laptev, I., 2005. On space-time interest points. *International Journal of Computer Vision*, 64, 2 (2005), 107–123. (cited on pages 6 and 46)

Laptev, I. and Lindeberg, T., 2006. Local descriptors for spatio-temporal recognition. In *Spatial Coherence for Visual Motion Analysis*, 91–103. Springer. (cited on pages 6 and 46)

Lea, C.; Flynn, M. D.; Vidal, R.; Reiter, A.; and Hager, G. D., 2016a. Temporal convolutional networks for action segmentation and detection. *CoRR*, abs/1611.05267 (2016). http://arxiv.org/abs/1611.05267. (cited on pages 74, 75, and 83)

Lea, C.; Reiter, A.; Vidal, R.; and Hager, G. D., 2016b. Efficient segmental inference for spatiotemporal modeling of fine-grained actions. *CoRR*, abs/1602.02995 (2016). http://arxiv.org/abs/1602.02995. (cited on page 83)

Lea, C.; Reiter, A.; Vidal, R.; and Hager, G. D., 2016c. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference of Computer Vision (ECCV)*, 36–52. (cited on page 58)

Lea, C.; Vidal, R.; and Hager, G. D., 2016d. Learning convolutional action primitives for fine-grained action recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 1642–1649. IEEE. (cited on page 75)

Lea, C.; Vidal, R.; Reiter, A.; and Hager, G. D., 2016e. Temporal convolutional networks: A unified approach to action segmentation. *CoRR*, abs/1608.08242 (2016). http://arxiv.org/abs/1608.08242. (cited on pages 74 and 75)

Lea, C. S.; Fackler, J. C.; Hager, G. D.; and Taylor, R. H., 2012. Towards automated activity recognition in an intensive care unit. In *MICCAI*, 10. (cited on page 74)

LeCun, Y. and Bengio, Y., 1998. The handbook of brain theory and neural networks. chap. Convolutional networks for images, speech, and time series, 255–258. MIT

Press, Cambridge, MA, USA. ISBN 0-262-51102-9. http://dl.acm.org/citation.cfm?id=303568.303704. (cited on page 4)

Lenz, I.; Lee, H.; and Saxena, A., 2015. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, (2015). (cited on page 46)

Li, B.; Lu, Y.; Godil, A.; Schreck, T.; Bustos, B.; Ferreira, A.; Furuya, T.; Fonseca, M. J.; Johan, H.; Matsuda, T.; et al., 2014a. A comparison of methods for sketch-based 3d shape retrieval. *Computer Vision and Image Understanding*, 119 (2014), 57–80. (cited on pages 21, 24, 30, 36, 37, and 38)

Li, B.; Lu, Y.; Li, C.; Godil, A.; Schreck, T.; Aono, M.; Burtscher, M.; Chen, Q.; Chowdhury, N. K.; Fang, B.; et al., 2015. A comparison of 3d shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding*, 131 (2015), 1–27. (cited on page 24)

Li, B.; Lu, Y.; Li, C.; Godil, A.; Schreck, T.; Aono, M.; Chen, Q.; Chowdhury, N.; Fang, B.; Furuya, T.; Johan, H.; Kosaka, R.; Koyanagi, H.; Ohbuchi, R.; and Tatsuma, A., 2014b. ShrecâĂŹ14 track: Comprehensive 3d shape retrieval. In *Proc. EG Workshop on 3D Object Retrieval*. (cited on pages 24, 30, 36, and 37)

Liu, J.; Feng, F.; Nakamura, Y. C.; and Pollard, N. S., 2014. A taxonomy of everyday grasps in action. In *14th IEEE-RAS International Conf. on Humanoid Robots, Humanoids*. (cited on page 64)

Lv, F. and Nevatia, R., 2006. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *Computer Vision–ECCV 2006*, 359–372. Springer. (cited on page 47)

Lyu, S. and Simoncelli, E. P., 2008. Nonlinear image representation using divisive normalization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8. IEEE. (cited on page 15)

Ma, S.; Sigal, L.; and Sclaroff, S., 2016. Learning activity progression in lstms for activity detection and early detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 19)

Mahasseni, B. and Todorovic, S., 2016. Regularizing long short term memory with 3d human-skeleton sequences for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 75 and 76)

Mandary, C.; Terlemez, O.; Do, M.; Vahrenkamp, N.; and Asfour, T., 2015. The kit whole-body human motion database. In *International Conferences on Advanced Robotics*, 329–336. (cited on page 41)

Melax, S.; Keselman, L.; and Orsten, S., 2013. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013*, 63–70. Canadian Information Processing Society. (cited on page 42)

Moeslund, T.; Hilton, A.; and Krüger, V., 2006. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104, 2 (2006), 90–126. (cited on page 41)

Molchanov, P.; Gupta, S.; Kim, K.; and Kautz, J., 2015. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 1–7. (cited on pages 42 and 43)

Murata, N., 1998. A statistical study of on-line learning. *Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK*, (1998), 63–92. (cited on page 16)

Nair, V. and Hinton, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814. (cited on page 15)

Ng, J. Y.-H.; Hausknecht, M.; Vijayanarasimhan, S.; Vinyals, O.; Monga, R.; and Toderici, G., 2015. Beyond short snippets: Deep networks for video classification. *CVPR 2015*, (2015). (cited on pages 15, 18, and 76)

Ni, B.; Paramathayalan, V. R.; and Moulin, P., 2014. Multiple granularity analysis for fine-grained action detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 75)

Ohn-Bar, E. and Trivedi, M. M., 2014. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15, 6 (2014), 2368–2377. (cited on page 42)

Oikonomidis, I.; Kyriazis, N.; and Argyros, A., 2011. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proceedings of the 2011 British Machine Vision Conference*, 1–11. BMVA, Dundee, UK. (cited on pages 43 and 46)

Olshausen, B. A. and Field, D. J., 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381, 6583 (1996), 607. (cited on page 2)

Osada, R.; Funkhouser, T.; Chazelle, B.; and Dobkin, D., 2002. Shape distributions. *ACM Transactions on Graphics*, 21, 4 (Oct. 2002), 807–832. (cited on page 24)

Panteleris, P.; Kyriazis, N.; and Argyros, A., 2015. 3d tracking of human hands in interaction with unknown objects. In *British Machine Vision Conference (BMVC 2015)*, 123–1. BMVA Press. (cited on page 43)

Pham, T.-H.; Kheddar, A.; Qammaz, A.; and Argyros, A. A., 2015. Towards force sensing from vision: Observing hand-object interactions to infer manipulation forces. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*, (2015). (cited on pages 64, 66, and 67)

Pieropan, A.; Ek, C. H.; and Kjellstrom, H., 2013. Functional object descriptors for human activity modeling. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 1282–1289. IEEE. (cited on page 46)

Pirsiavash, H.; Vondrick, C.; and Torralba, A., 2014. Inferring the why in images. *arXiv preprint arXiv:1406.5472*, (2014). (cited on page 45)

Rizzolatti, G.; Fogassi, L.; and Gallese, V., 2001. Neurophysiological mechanisms underlying the understanding and imitation of action. *Nature Reviews Neuroscience*, 2, 9 (2001), 661–670. (cited on pages 61, 62, and 73)

Rogez, G.; Khademi, M.; Supančič III, J.; Montiel, J. M. M.; and Ramanan, D., 2014. 3d hand pose detection in egocentric rgb-d images. In *Workshop at the European Conference on Computer Vision*, 356–371. Springer International Publishing. (cited on page 46)

Rogez, G.; Supancic, J. S., III; and Ramanan, D., 2015. Understanding everyday hands in action from rgb-d images. In *IEEE International Conference on Computer Vision (ICCV)*. (cited on pages 46, 47, and 64)

Rohrbach, M.; Amin, S.; Andriluka, M.; and Schiele, B., 2012. A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 1194–1201. IEEE. (cited on pages 6 and 46)

Rohrbach, M.; Rohrbach, A.; Regneri, M.; Amin, S.; Andriluka, M.; Pinkal, M.; and Schiele, B., 2015. Recognizing fine-grained and composite activities using hand-centric features and script data. *CoRR*, abs/1502.06648 (2015). http://arxiv.org/abs/1502.06648. (cited on page 75)

Romero, J.; Feix, T.; Ek, C. H.; Kjellstrom, H.; and Kragic, D., 2013. A metric for comparing the anthropomorphic motion capability of artificial hands. *IEEE Transactions on Robotics*, 29, 6 (2013), 1342–1352. (cited on page 46)

Romero, J.; Kjellström, H.; and Kragic, D., 2010. Hands in action: real-time 3d reconstruction of hands in interaction with objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 458–463. IEEE. (cited on page 64)

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J., 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 3 (1988), 1. (cited on pages 2 and 4)

Rupprecht, C.; Lea, C.; Tombari, F.; Navab, N.; and Hager, G. D., 2016. Sensor substitution for video-based action recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. (cited on page 74)

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al., 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 3 (2015), 211–252. (cited on page 5)

Russell, S. J. and Norvig, P., 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edn. ISBN 0137903952. (cited on page 1)

Ryoo, M.; Fuchs, T. J.; Xia, L.; Aggarwal, J.; and Matthies, L., 2015. Robot-centric activity prediction from first-person videos: What will they do to me? In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 295–302. ACM. (cited on page 48)

Ryoo, M. S., 2011. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. (cited on page 47)

Ryoo, M. S. and Matthies, L., 2013. First-person activity recognition: What are they doing to me? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2730–2737. (cited on page 47)

Saavedra, J. M.; Bustos, B.; Schreck, T.; Yoon, S.; and Scherer, M., 2012. Sketch-based 3d model retrieval using keyshapes for global and local representation. In *EG 3DOR'12*, 47–50. Eurographics Association. doi:10.2312/3DOR/3DOR12/047-050. http://dx.doi.org/10.2312/3DOR/3DOR12/047-050. (cited on page 37)

Salakhutdinov, R. and Hinton, G. E., 2007. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, vol. 11. (cited on pages 18 and 25)

Saxena, A.; Driemeyer, J.; and Ng, A. Y., 2008. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27, 2 (2008), 157–173. (cited on page 46)

Schuldt, C.; Laptev, I.; and Caputo, B., 2004. Recognizing human actions: A local SVM approach. In *Proc. International Conference on Pattern Recognition*. (cited on pages 41 and 47)

Scovanner, P.; Ali, S.; and Shah, M., 2007. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, 357–360. ACM. (cited on pages 6 and 46)

Shi, Q.; Cheng, L.; Wang, L.; and Smola, A., 2011. Human action segmentation and recognition using discriminative semi-markov models. *International journal of computer vision*, 93, 1 (2011), 22–32. (cited on page 47)

Shilane, P.; Min, P.; Kazhdan, M.; and Funkhouser, T., 2004. The princeton shape benchmark. In *Shape modeling applications, 2004. Proceedings*, 167–178. IEEE. (cited on pages 21, 23, 24, and 30)

Shimoga, K. B., 1996. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15, 3 (1996), 230–266. (cited on page 46)

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 43)

Simonyan, K. and Zisserman, A., 2014a. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, 568–576. (cited on page 76)

Simonyan, K. and Zisserman, A., 2014b. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556 (2014). (cited on pages 15, 53, and 77)

Simonyan, K. and Zisserman, A., 2014c. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, (2014). (cited on page 47)

Song, S.; Chandrasekhar, V.; Mandal, B.; Li, L.; Lim, J.-H.; Sateesh Babu, G.; Phyo San, P.; and Cheung, N.-M., 2016. Multimodal multi-stream deep learning for egocentric activity recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. (cited on page 75)

Sousa, P. M. A. and Fonseca, M. J., 2010. Sketch-based retrieval of drawings using spatial proximity. *J. Vis. Lang. Comput.*, 21, 2 (2010), 69–80. (cited on page 37)

Srivastava, N. and Salakhutdinov, R., 2014. Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15 (2014), 2949–2980. http://jmlr.org/papers/v15/srivastava14b.html. (cited on page 74)

Starner, T.; Weaver, J.; and Pentland, A., 1998. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 12 (1998), 1371–1375. (cited on page 43)

Stein, S. and McKenna, S. J., 2013a. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and U Computing*, 729–738. ACM. (cited on page 58)

Stein, S. and McKenna, S. J., 2013b. User-adaptive models for recognizing food preparation activities. In *Proceedings of the 21st ACM International Conference on Multimedia, 5th International Workshop on Multimedia for Cooking and Eating Activities (CEA'13), Barcelona, Spain*. ACM. (cited on page 75)

Summers-Stay, D.; Teo, C. L.; Yang, Y.; Fermüller, C.; and Aloimonos, Y., 2012. Using a minimal action grammar for activity understanding in the real world. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 4104–4111. IEEE. (cited on page 73)

Supancic, J. S.; Rogez, G.; Yang, Y.; Shotton, J.; and Ramanan, D., 2015. Depth-based hand pose estimation: data, methods, and challenges. In *Proceedings of the IEEE International Conference on Computer Vision*, 1868–1876. (cited on page 43)

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A., 2015. Going deeper with convolutions. In *CVPR*. http://arxiv.org/abs/1409.4842. (cited on page 15)

Takano, W.; Ishikawa, J.; and Nakamura, Y., 2015. Using a human action database to recognize actions in monocular image sequences: Recovering human whole body configurations. *Advanced Robotics*, 29, 12 (2015), 771–784. (cited on page 41)

Tangelder, J. W. and Veltkamp, R. C., 2008. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl.*, 39, 3 (Sep. 2008), 441–471. doi:10.1007/s11042-007-0181-0. http://dx.doi.org/10.1007/s11042-007-0181-0. (cited on page 24)

Tatsuma, A.; Koyanagi, H.; and Aono, M., 2012. A large-scale shape benchmark for 3d object retrieval: Toyohashi shape benchmark. *Signal and Information Processing Association*, (2012), 3–6. (cited on page 38)

Theano Development Team, 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688. (cited on pages 31 and 81)

Tiest, W. M. B. and Kappers, A. M., 2014. *Multisensory Softness*, chap. Physical Aspects of Softness Perception, 3–15. Springer, London. (cited on page 62)

Turaga, P.; Chellappa, R.; Subrahmanian, V.; and Udrea, O., 2008. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18, 11 (2008), 1473–1488. (cited on page 41)

Vapnik, V. and Vashist, A., 2009. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22, 5-6 (2009), 544–557. doi:10.1016/j.neunet.2009.06.042. http://dx.doi.org/10.1016/j.neunet.2009.06.042. (cited on pages 74 and 76)

Venugopalan, S.; Xu, H.; Donahue, J.; Rohrbach, M.; Mooney, R.; and Saenko, K., 2014. Translating videos to natural language using deep recurrent neural networks. In *arXiv:1412.4729*. (cited on pages 14, 18, and 76)

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D., 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164. (cited on pages 18 and 76)

Visin, F.; Kastner, K.; Cho, K.; Matteucci, M.; Courville, A.; and Bengio., Y., 2014. A recurrent neural network based alternative to convolutional networks. In *International Conference on Image Processing Theory, Tools and Applications (IPTA)*. (cited on page 15)

Vondrick, C.; Pirsiavash, H.; and Torralba, A., 2016. Anticipating visual representations from unlabeled video. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on page 45)

Walker, J.; Gupta, A.; and Hebert, M., 2014. Patch to the future: Unsupervised visual prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3302–3309. (cited on page 45)

Wang, H.; Kläser, A.; Schmid, C.; and Liu, C.-L., 2011. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 3169–3176. IEEE. (cited on pages 6 and 46)

Wang, J.; Liu, Z.; Wu, Y.; and Yuan, J., 2014. Learning actionlet ensemble for 3d human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 5 (2014), 914–927. (cited on pages 18 and 47)

Wang, S. B.; Quattoni, A.; Morency, L.-P.; Demirdjian, D.; and Darrell, T., 2006. Hidden conditional random fields for gesture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 1521–1527. (cited on page 43)

Wessel, R.; Blümel, I.; and Klein, R., 2009. A 3d shape benchmark for retrieval and automatic classification of architectural data. In *3DOR '09* (Munich, Germany, 2009), 53–56. Eurographics Association. doi:10.2312/3DOR/3DOR09/053-056. http://dx.doi.org/10.2312/3DOR/3DOR09/053-056. (cited on page 24)

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

Xie, D.; Todorovic, S.; and Zhu, S.-C., 2013. Inferring "dark matter" and "dark energy" from videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2224–2231. (cited on page 45)

Yang, Y.; Fermüller, C.; Li, Y.; and Aloimonos, Y., 2015. Grasp type revisited: A modern perspective on a classical feature for vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 45)

Yih, W.; Toutanova, K.; Platt, J. C.; and Meek, C., 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11 (Portland, Oregon, 2011), 247–256. http://dl.acm.org/citation.cfm?id=2018936.2018965. (cited on pages 5 and 18)

Zagoruyko, S. and Komodakis, N., 2015. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4353–4361. (cited on page 18)

Zampogiannis, K.; Yang, Y.; Fermüller, C.; and Aloimonos, Y., 2015. Learning the spatial semantics of manipulation actions through preposition grounding. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 1389–1396. IEEE. (cited on page 73)

Zeiler, M. D., 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, (2012). (cited on page 16)

Zeiler, M. D. and Fergus, R., 2014. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, 818–833. Springer. (cited on page 4)

Zhu, Y.; Zhao, Y.; and Zhu, S.-C., 2015. Understanding tools: Task-oriented object modeling, learning and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2855–2864. (cited on page 45)