Group Event Recognition in Ice Hockey

by

Sijia Tian

B.Eng., Tsinghua University, 2016

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

December 2018

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Group Event Recognition in Ice Hockey

submitted by **Sijia Tian** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Jim Little, Computer Science
Supervisor

Leonid Sigal, Computer Science
Additional Examiner

Abstract

With the success of deep learning in computer vision community, most approaches for group activity recognition in sports started relying on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). However, how to model the interactions among players and the interactions between players and the scene remains a challenging problem. In order to better model these interactions, we propose two models. Our first model combines features of all players in a scene through an attention mechanism. The aggregated feature is then concatenated with the feature of the frame and passed through an RNN to generate the final prediction. In our second model, we designed a spatial grid feature and a temporal grid feature calculated from appearance features and motion features of all players in a scene, as well as their locations. We then apply CNNs to the spatial grid feature, the temporal grid feature, target frame of the scene (the frame at which the event happens), and the stack of optical flow containing the target frame separately. Results from the four streams are fused through score fusion to make the final prediction. Inputs to our models are: the target frame image, a stack of optical flow images, bounding boxes of players and coordinates of players calculated from homography matrix of the frame. We evaluated the two models on an Ice Hockey dataset, and results show that both models produced promising results. We also provide a possible solution for event detection in a more general setting.

Lay Summary

Group activity recognition in sports can help teams better understand and analyze the games, as well as provide further insight for social scene understanding in a more general setting. The interactions among players and the interactions between players and the scene are crucial for analyzing group activities, but modeling these interactions remains a challenging problem. To address this issue, we propose two models in this work. The models predict the group activity using frame images from videos, sub-images of players and their locations. Both models are based on deep learning architectures, which have achieved great success on image and video applications in recent years. Our models are tested on an ice hockey dataset.

Preface

This thesis is submitted in partial fulfillment of the requirements for a Mater of Science Degree in Computer Science. The entire work presented here is original work done by the author, Sijia Tian, performed under the supervision of Professor James J. Little.

Table of Contents

Abstract . . . . . **iii**

Lay Summary . . . . . **iv**

Preface . . . . . **v**

Table of Contents . . . . . **vi**

List of Tables . . . . . **viii**

List of Figures . . . . . **ix**

Acknowledgments . . . . . **xiii**

1 Introduction **1**

2 Related Work **10**

2.1 Deep Neural Networks 10

2.1.1 Neural Networks 11

2.1.2 Convolutional Neural Networks 12

2.1.3 Recurrent Neural Networks 13

2.2 Traditional Methods for Action Recognition 15

2.3 Deep Methods for Single-person Action Recognition 16

2.3.1 Basic Models and their extensions 17

2.3.2 Development of Datasets, Computational Costs and Beyond 20

2.4 Deep Methods for Multi-person Event Recognition 22

3 Problem Formulation **25**
3.1 Dataset 25
3.2 Data Preprocessing 26
3.2.1 Event Labels and Target Frames 26
3.2.2 Bounding Boxes of Players 27
3.2.3 Mapping from image to rink 28
3.2.4 Team Identifications of Players 29
3.3 Problem Scope 31
3.4 Evaluation Metrics 31
3.5 Loss Function 33

4 Hierarchical LSTM Model with Attention **34**
4.1 Model Overview 34
4.1.1 Feature Extraction 35
4.1.2 Hierarchical Temporal Component Modeling 35
4.2 Implementation Details 38
4.3 Results 38

5 Grid Feature Model **40**
5.1 Model Overview 40
5.1.1 Two Stream Model for Frame Images 40
5.1.2 Two Stream Model For Grid Feature 42
5.2 Implementation Details 46
5.3 Results 46

6 Conclusions **51**

Bibliography **54**

List of Tables

Table 2.1 Accuracy of different networks 20
Table 2.2 Action Recognition Datasets 21

Table 3.1 Description of event labels 26

Table 4.1 Results for ablation studies 39

Table 5.1 Accuracy and average precision for each stream and different fusion methods 49
Table 5.2 Comparison with existing models. 49
Table 5.3 Ablative analysis 50

Table 6.1 Results of our models and comparison to previous work 53

List of Figures

Figure 1.1 Examples from the image classification dataset ImageNet [DDS+09]. Source: [KSH12] 1

Figure 1.2 Examples from the action recognition dataset UCF101 [SZS12]. Source: [SZS12] 2

Figure 1.3 Examples of the six group activities we aim to classify in our ice hockey dataset. In Figure *(a)*, the player in the red box is traveling in the rink controlling the puck. In Figure *(b)*, the player in the red box is dumping the puck in the direction of the arrow, so that the puck will enter the blue line indicating his team's offensive zone. In Figure *(c)*, the player in the red box is dumping the puck in the direction of the arrow, so that the puck will go out of the blue line indicating his team's defensive zone. In Figure *(d)*, the player in the red box is aiming the puck at the direction of the player in the blue box, which is a player on the same team with him. In Figure *(e)*, the player in the blue box is hiding the puck from the player in the blue box, which is a player on the opposite team. In Figure *(f)*, the player in the red box is aiming the puck at the goal. 5

Figure 1.4 Examples of the volleyball dataset introduced in [IMD+16]. This dataset contains labels for both group activities and individual actions. Labels for individual actions allow fine-tuning action recognition networks, which will make feature extraction from players more suitable for the task. 6

Figure 1.5 Overview of our first model. Each player is tracked by an LSTM modeling his motion, and an LSTM modeling his change in position. The two hidden states of the two LSTMs are then concatenated together and seen as hidden state for the player. Hidden states of the players at the same team are aggregated by attention pooling. During attention pooling, the weight for the player is decided by hidden state of the player, hidden state of the frame, and the hidden state of the event. Aggregated hidden state of players are then concatenated with hidden state of the frame, and fed into an LSTM used for modeling the progress of the event. 8
Figure 1.6 Overview of our second model. Final prediction is made through score fusion of four streams. Input to the four streams are: target frame of the event; an optical flow stack ranging from four frames before the target frame to five frames after the target frame; a spatial grid feature; a temporal grid feature. The grid features are designed to jointly represent the features of the players and their locations. 9

Figure 2.1 An example of a fully connected neural network. This network contains input and output layers, and one hidden layer. Connections between neurons are represented by arrows. Each neuron in the hidden layers and output layer is calculated as an activation of the weighted sum of all the neurons in the previous layer. Source: [Karb] 11
Figure 2.2 An example of a first convolution layer. Each neuron in a convolution layer is only connected to a small region in width and height, but to the full depth. Source: [Kara] 12
Figure 2.3 An example of a pooling operation. The receptive field of a neuron on the right is represented by squares in the same color on the left. Source: [Kara] 13
Figure 2.4 An example of a RNN and how it can be unrolled. Source: [Ola] 14
Figure 2.5 An illustration of LSTM. Source: [Ola] 14

Figure 3.1 Distribution of events 27
Figure 3.2 Detailed diagram of an ice hockey rink. Source: [rin] 28
Figure 3.3 An illustration of the rink coordinate system and offensive/defensive zones 30
Figure 3.4 Examples for bounding box annotation of players (left side), and their transformed coordinates in rink coordinate system calculated using the homography matrix (right side). 30
Figure 3.5 Examples of events after flipping. Upper left corner was a *dump in* by the blue player from right to left. Upper right corner is the event after flipping. Now the player dumps in (attacks) from left to right. Lower left corner was a *dump out* by the blue player from right to left. Lower right corner is the event after flipping. Now the player dumps out (defends) from right to left. 32

Figure 4.1 Confusion matrix for hierarchical LSTM model with attention 39

Figure 5.1 The structure of the two stream model. Spatial stream ConvNet is applied on one single RGB frame from videos, while temporal stream ConvNet operates on a stack of optical flow images. Source: [SZ14] 41
Figure 5.2 Grid layout designed for ice hockey rink. Based on the position of attacking zone, neutral zone on both sides, defending zone and the end zone faceoff spot and circle, we divide the rink into 8×12 grid cells. 43
Figure 5.3 An illustration of the spatial grid feature calculation process. Given a target frame, we first calculate each player's location using homography matrix, and extract the spatial feature from players as shown in the two upper left images. Then we iterate through the grid cells in the rink as shown in the lower left image. For each cell (m,n), we find the players in the cell and sum up the features of the players. This vector is the (m,n) element of the final spatial grid feature. 44

Figure 5.4 Block diagram of our grid feature model. 45
Figure 5.5 Confusion matrix for classifying on RGB image of the target frame of each event 47
Figure 5.6 Confusion matrix for classifying on a stack of optical flow images in each event ranging from 4 frames before the target frame to 5 frames after the target frame 47
Figure 5.7 Confusion matrix for classifying on G_{spatial} 48
Figure 5.8 Confusion matrix for classifying on G_{temporal} 48
Figure 5.9 Confusion matrix for score fusion among all streams 48

Acknowledgments

I would like to express my heartfelt gratitude to a number of people here for their generous support during my Masters.

First of all, I would like to thank my professor James J. Little, who is a great mentor. He was always supportive of my ideas and encouraging me to explore them. Along the way, he also helped me gain new insight of our project. I would also like to thank Prof. Leonid Sigal, for agreeing to be the second reader of my thesis and providing helpful feedback.

Next I would like to thank all the professors who I have taken courses with and helped me along the way. I would like to thank Prof Mark Greenstreet for being my advisor during my first year here. He gave me great advice about how to choose courses and research area when I first started. I would also like to thank Prof Mark Schmidt and Prof Hu Fu for giving great courses from which I learned so much.

I'm also grateful to my labmates, Lili, Jimmy, Julieta, Moumita, and Rayat, for generously giving me guidance and help along my research. I learned so much from you. I would also like to thank my boyfriend Daniel and my friends, Yanan, Itrat, Ariadna, Alex, Matthew, Chenxi, for not only the support, but also the fun we had.

Last but not least, I would like to thank my family, for their love, support, and belief in me, without which none of these wonderful things would have happened.

Chapter 1

Introduction

Classification and detection in images and videos have been hot topics in the computer vision research community for decades, due to their wide applications in areas such as navigation and surveillance. Results in image-related tasks had been improving steadily over the years with better designed hand-crafted feature representations such as HOG [DT05] and SIFT [Low04]. But this trend was changed by the revival of Convolutional Neural Networks (CNNs). Since the amazing performance of AlexNet [KSH12] on the image classification task ImageNet [DDS+09] in 2012, researchers have been improving performance in different tasks by designing different CNN architectures.

Since videos are consecutive temporal sequences of images, video-related tasks have been benefiting greatly from image-related tasks over the years. Early research extended state-of-the-art hand-crafted features to 3D to from video representations, such as HOG3D [KMS08] and HOF3D [LMSR08]. Similar to the de-

Figure 1.1: Examples from the image classification dataset ImageNet [DDS+09]. Source: [KSH12]

Figure 1.2: Examples from the action recognition dataset UCF101 [SZS12]. Source: [SZS12]

velopment of image-related tasks, since CNNs came back into fashion, the center of research has changed from designing hand-crafted features to designing network structures that can better learn feature representations by themselves. Great improvements have been achieved with the development of two stream architectures [SZ14, FPW16], 3D CNNs [TBF+15, TRS+17], and Long Short Term Memory (LSTM) networks [DHR+15].

In the above work on image and video related classification tasks, to simplify the problem, most of these architectures are designed for and tested on single-person action recognition datasets, such as UCF101 [SZS12] and HMDB51 [HHE+11]. However, in realistic settings, there usually would be many people involved in a social scene. So this prompts the need to further study social scene understanding in a more general and comlex setting.

Although there has been work studying general social scenes directly [BAF+17, AGR+16], more work in this area has been focusing on action recognition and detection in sports games. There are mainly two reasons for this. First of all, they have many applications in game analysis, which can help teams gain a better under-

standing of games. Second of all, compared with social scenes in a more general setting, sports games are highly constrained in terms of the number of people, size of the area, and the rules they need to follow according to different games. However, in spite of these differences between sports games and more general social scenes, sports games still carry important traits of social scenes, such as the multi-level analysis for scene understanding, and the spatial and temporal dependencies involved. So understanding activities in sports lays a solid foundation for understanding more general scenes.

As mentioned above, to analyze a social scene, or a group activity in a sports game, there are two crucial factors. First of all, a group activity contains activities happening at both scene level and individual player level. For example, for a *shot* in football games, there will be a player kicking the ball (individual player level) towards the direction of the goal (scene level). Second of all, a group activity requires interactions among players, i.e. the spatial and temporal dependencies among players. For example, for a *pass* in football or basketball games, a player needs to send the ball in the direction of another player at the same team; while for a *block*, a player tries to steal the ball from the player from the opposing team who is in possession of the ball.

In order to combine these factors more efficiently, there has been much work discussing what are the possible approaches for modeling the interaction between players and the scene, and the interaction among players. In [BAF+17], authors use fully-convolutional network (FCN) to generate multi-scale feature map of the scene, which is then passed through RNN to predict labels for group activities. [IMD+16] use LSTM to model temporal information of each individual player. This temporal information of players are then concatenated with spatial information to form the spatio-temporal representation of the players. These representations are aggregated through max-pooling, and passed through another LSTM modeling the progress of the event. However, this approach does not include the interactions between players and the scene, and there might be information missing through max-pooling. These problems are partially solved in [RHAEHG16]. In their model, information of players are aggregated through an attention-pooling mechanism instead of max-pooling. Before passing into the LSTM used to model the event, aggregation of the features of players is concatenated with feature repre-

sentation of the scene.

It is worth noting that, although sports games have restricted rules and yield an easier problem as a social scene, they have their own complexities as well. For example, fast movement of players and occlusion among players make player detection and player feature representation difficult; different games have different rules, which might require some effort to generalize the model applied in one game to another.

In this work, we focus on group activity recognition in ice hockey games. We conduct our experiments on a dataset containing National Hockey League (NHL) ice hockey game footage provided by SPORTLOGiQ. According to the annotations provided by SPORTLOGiQ, there are 15 possible group activities in a game: *face off*, *whistle*, *pass*, *loose puck recovery*, *reception*, *carry*, *block*, *shot*, *dump out*, *puck protection*, *dump in*, and *check*. Detailed descriptions of these activities and their distribution in games can be found in Appendix **??**. Of these 15 labels, we aim at classifying 6 in our work: *dump in*, *dump out*, *shot*, *pass*, *carry*, and *puck protection*. Examples and descriptions of these six group activities can be found in Figure 1.3 and Table 3.1.

Besides frame images that can be extracted from the footage, from annotations in the dataset, we can also extract bounding boxes of players and calculate their positions in the rink coordinate system. With both scene level information (frame images) and player level information (bounding boxes and coordinates of players), we can build networks to model their interactions resembling previous work on sports games we mentioned above.

However, our task can be more challenging than other group activity recognition tasks in several ways. First of all, different events may have different time spans, which is not only the case for events of different labels, but also for events of the same label. Since information regarding the ending frame of each event, the number of frames we choose to use as input may not cover the whole process of the event. Second of all, there might also be cases when two events overlap or one closely follows the other, which might cause confusion to the network. Third of all, although the position of the puck can be crucial in identifying an event, this information is not provided in the dataset. The high velocity of the puck makes it blurry in many frames, and background such as lines in the rink may overlap

Figure 1.3: Examples of the six group activities we aim to classify in our ice hockey dataset. In Figure *(a)*, the player in the red box is traveling in the rink controlling the puck. In Figure *(b)*, the player in the red box is dumping the puck in the direction of the arrow, so that the puck will enter the blue line indicating his team's offensive zone. In Figure *(c)*, the player in the red box is dumping the puck in the direction of the arrow, so that the puck will go out of the blue line indicating his team's defensive zone. In Figure *(d)*, the player in the red box is aiming the puck at the direction of the player in the blue box, which is a player on the same team with him. In Figure *(e)*, the player in the blue box is hiding the puck from the player in the blue box, which is a player on the opposite team. In Figure *(f)*, the player in the red box is aiming the puck at the goal.

with the puck. These factors make it difficult to annotate the puck position in many frames. Last but not least, unlike in volleyball games, we do not have action labels for individual players. Lacking this information, we cannot fine-tune single-person action recognition networks to make feature extraction from individual players more accurate. Due to this lack of some necessary annotations, we mainly aim at classifying each event exploiting the annotations we have, which are bounding boxes and coordinates of players in the scene.

Based on existing literature on action recognition and group activity recognition, we propose two models for group activity recognition in ice hockey, aiming at making full use of the information of individual players. Our first model is inspired by [RHAEHG16]. An overview of the model can be found in Figure 1.5. In this

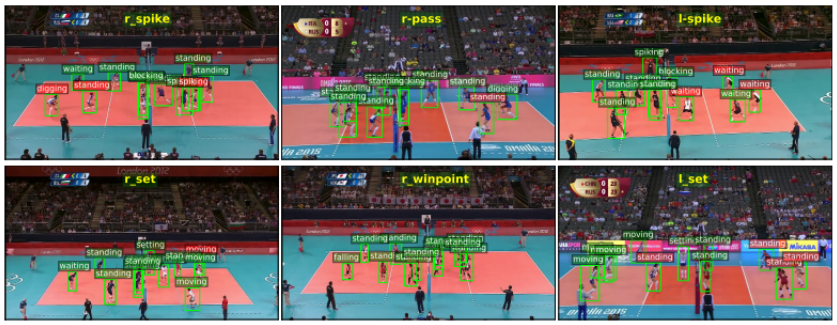

Figure 1.4: Examples of the volleyball dataset introduced in [IMD+16]. This dataset contains labels for both group activities and individual actions. Labels for individual actions allow fine-tuning action recognition networks, which will make feature extraction from players more suitable for the task.

model, we use four LSTMs to track appearance of each player, position of each player, the entire frame, and the event separately. An appearance feature of each player extracted by CNN is used as input to the LSTM modeling the motion of the player, which is represented by the hidden state of the LSTM. The position of each player is used as input to the LSTM modeling the player's change in position, which is represented by the hidden state of the LSTM. Hidden states of these two LSTMs are concatenated and viewed as feature for the player. To aggregate features of different players, instead of max pooling, we adopt the attention pooling technique described in [RHAEHG16]. The weight for each player is decided by his feature, hidden state of the frame, and hidden state of the event. The hidden state of the frame is then concatenated with the aggregated features of players, and fed into the LSTM used for modeling the progression of the event. The output of the event LSTM will be our final prediction for the event.

Our second model expands on the two stream model [SZ14] developed for single-person action recognition. An overview of the model can be found in Figure 1.6. In order to incorporate players' features with their positions, we develop one grid feature based on appearance feature of players, and another grid feature based on the motion feature of players. We then design two CNNs for classifying on these grid features. The output of these two networks are then combined with the

output of the original two stream model through score fusion to make the final prediction. Compared with the first model, this model enjoys the merit of losing less information of individual players; and taking interactions among players into consideration through convolution.

There are several contributions of this work. First, we learn a generalized model with videos in the same coordinate system, which can be further applied in event detection and team identification tasks. Second, we explore several possible approaches for incorporating scene level features with individual player features in group activity recognition tasks, based on current literature on single-person action recognition using LSTM and two stream models. Third, we propose several methods for fusing player level features that have the advantage of losing less crucial information over max-pooling.

The rest of the thesis is organized as follows: First, in Chapter 2, we review related work in single-person action recognition and group activity recognition. In this chapter, we briefly go over the main approaches used for action recognition both before and after deep learning comes along, and their relations to approaches used for image classification tasks. We also review the expansion of dataset and the improvement in results over the years. In addition, we provide a brief introduction of the types of deep networks that we have used in our systems, such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). In Chapter 3, we will introduce the annotations provided in the dataset we use, and the preprocessing techniques we apply. We will also discuss the scope of our problem and how it can fit into tasks in more realistic settings. In Chapter 4 and Chapter 5, we will explain our two models in detail and present the results. A comparison to previous work on the same dataset will be provided, along with complementary experiments to verify our network designing choices. We will summarize our work in Chapter 6 with possible extensions in the future.

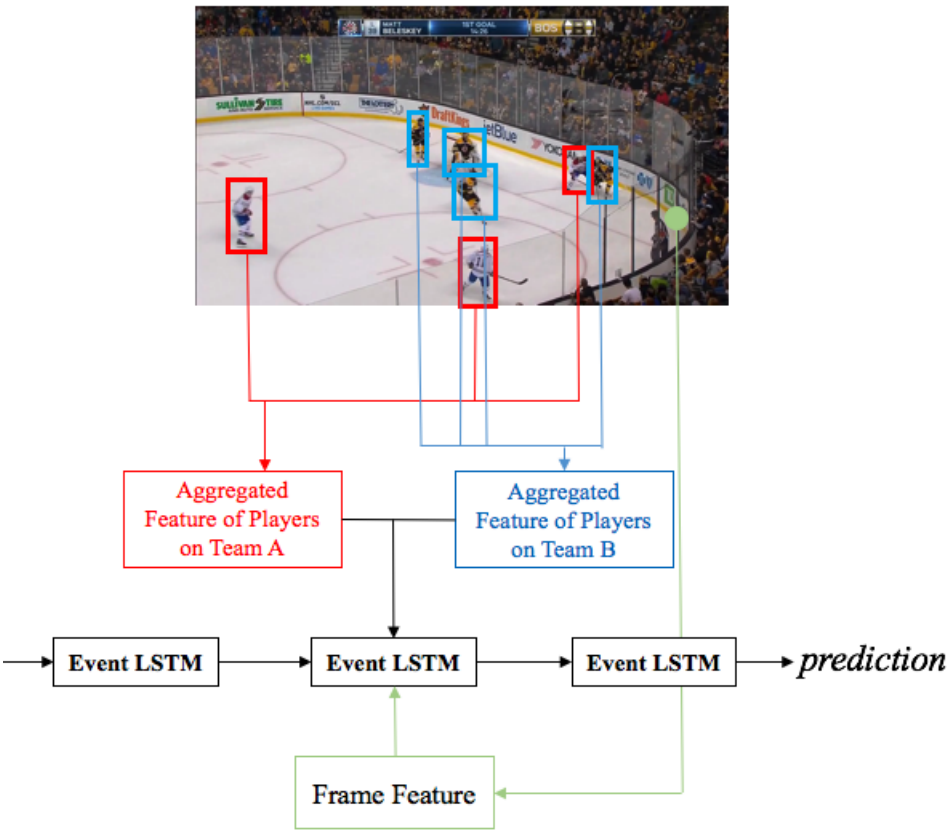

Figure 1.5: Overview of our first model. Each player is tracked by an LSTM modeling his motion, and an LSTM modeling his change in position. The two hidden states of the two LSTMs are then concatenated together and seen as hidden state for the player. Hidden states of the players at the same team are aggregated by attention pooling. During attention pooling, the weight for the player is decided by hidden state of the player, hidden state of the frame, and the hidden state of the event. Aggregated hidden state of players are then concatenated with hidden state of the frame, and fed into an LSTM used for modeling the progress of the event.

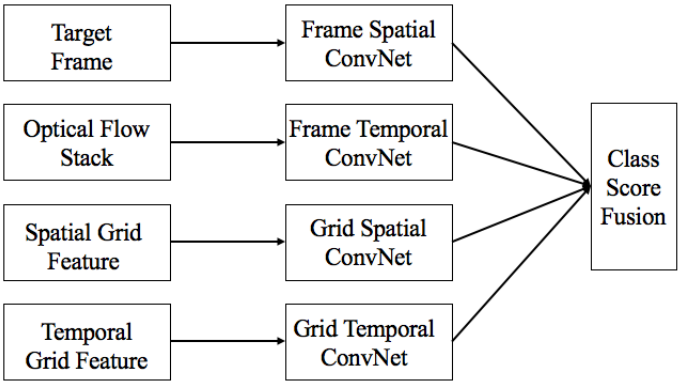

Figure 1.6: Overview of our second model. Final prediction is made through score fusion of four streams. Input to the four streams are: target frame of the event; an optical flow stack ranging from four frames before the target frame to five frames after the target frame; a spatial grid feature; a temporal grid feature. The grid features are designed to jointly represent the features of the players and their locations.

Chapter 2

Related Work

Image classification and action recognition in videos have been major tasks in computer vision for many years. Previously, researchers have focused on designing better hand-crafted features, such as SIFT [Low04] and HOG [DT05] for image classification. These features are then extended to 3D to from video representations [LL03, WTG08, KMS08]. But these all changed since people re-discovered the computational and representational power of neural networks [KSH12]. In this section, we will talk about action recognition using traditional methods, the change the research in the area has undergone ever since the success of convolutional neural networks, and how these networks work. We will also briefly talk about how our research are related to and influenced by previous work.

2.1 Deep Neural Networks

In recent years, deep neural networks, when combined with the fast development processing power of machines such as GPUs, have shown great potential in a number of computer vision applications. Both of our models are also based on deep networks. Convolutional neural networks (ConvNets or CNNs) trained on action recognition datasets are used to extract players' features in both models. In our first model, we apply ConvNets on aggregated players' features besides both appearance and motion streams. In our second model, LSTMs are used to model the dynamics of both individual players and the entire event. We will review both of

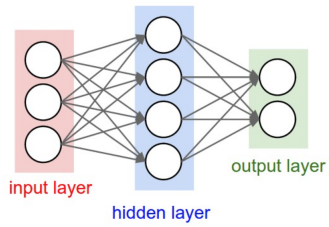

Figure 2.1: An example of a fully connected neural network. This network contains input and output layers, and one hidden layer. Connections between neurons are represented by arrows. Each neuron in the hidden layers and output layer is calculated as an activation of the weighted sum of all the neurons in the previous layer. Source: [Karb]

these networks in Sections 2.1.1, 2.1.2 and 2.1.3.

2.1.1 Neural Networks

The structure for neural networks was first inspired by human nervous system. In the nervous system, a neuron receives input signals (x_i) from a number of other neurons and this neuron would decide the influence each input signal will have on its output (i.e. a weight w_i). These signals are then summed up according to their given weights ($\sum w_i x_i$). If the sum exceeds a certain threshold, this neuron fires (i.e. sends a spike along its axon). Similar to this simplified structure, an example of a simple neural network can be found in Figure 2.1. A neural network contains an input layer, multiple hidden layers and an output layer. At each node in the hidden layers and the output layer, we first compute the weighted sum of the signals it receives (e.g. $\sum w_i x_i$). Then we need to decide if this neuron can "fire" or not. The thresholding function deciding if a neuron can fire is called an activation function (e.g. f). So the output of a neuron can be calculated as $f(\sum w_i x_i)$. In classification applications, the outputs of the output layer are regularized and generate confidence scores for each class.

Ever since the 1940s, there has been continuous work on using neural networks for classification [RHHD56, Ros58]. But it had not been the center of attention in research community until the success of convolutional neural networks in general computer vision applications.

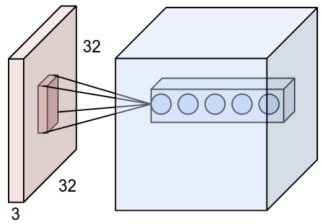

Figure 2.2: An example of a first convolution layer. Each neuron in a convolution layer is only connected to a small region in width and height, but to the full depth. Source: [Kara]

2.1.2 Convolutional Neural Networks

Although neural networks have proved to have strong representational power in [Cyb89], they have problems when being scaled to full images. For an RGB image that is 200 pixels in height and 200 pixels in width, one single neuron of a fully-connected layer operating on the whole image will have $200 \times 200 \times 3 = 120,000$ weights. The total number of parameters would explode quickly resulting in slow training and overfitting. To solve this problem, people developed 2D convolutions and 2D max-pooling so that the total number of parameters could be reduced while spatial information could be preserved.

2D convolutions can be seen as small filters operating on an input volume. Each filter is much smaller in width and height than the input volume, while the depth is the same as the input volume, thus the number of parameters we need to learn (i.e. number of parameters in the filters) is largely reduced. A convolution layer is composed of this kind of small filters. An example of a convolution layer can be found in Figure 2.2.

In order to control the number of parameters and reduce overfitting, besides using convolution layers instead of fully connected layers, it is also common to insert pooling layers between convolutional layers. Pooling layers can also be seen as filters operating on a input volume. In practice, researchers mostly use max-pooling, that is to say the value of a neuron in the next layer would be the maximum value in its receptive field. An example of a max-pooling operation can be found in Figure 2.3.

Convolution layers and pooling layers are the building blocks of a Convolu-

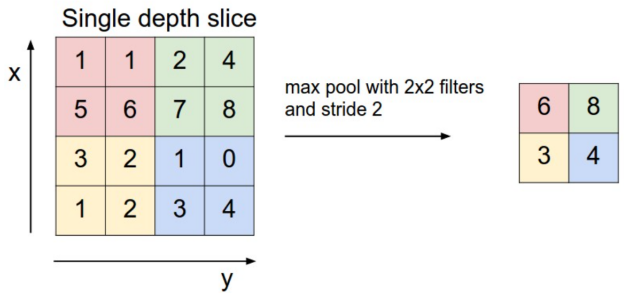

Figure 2.3: An example of a pooling operation. The receptive field of a neuron on the right is represented by squares in the same color on the left. Source: [Kara]

tional Neural Network (ConvNet) that is widely used in computer vision applications in recent years. After learning the spatial structure of an image through a certain number of convolution layers and pooling layers, the output volume will be passed through one or more fully connected layers to generate final class scores.

Although [LBBH98] designed a ConvNet called LeNet for handwritten digits recognition in the late 1980s, due to limited computational power and the huge amount of parameters in the network, it was not widely used at the time. The big breakthrough for ConvNets did not come until the implementation of large scale ConvNets on GPUs became possible. [KSH12] designed AlexNet which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 and beat previous hand-crafted features by a large margin. Follow up work on applying ConvNets on ImageNet challenge, such as VGG [SZ15], GoogLeNet [SLJ+15] and ResNet [HZRS16], has reduced the error rate on this dataset from 16% to 2%. This line of great success of ConvNets encouraged researchers to use ConvNets in computer vision applications other than image classification. A consistent boost in accuracy is witnessed in many areas, such as action recognition and image segmentation.

2.1.3 Recurrent Neural Networks

Because of the ability to store long term information and model temporal structure of input sequence, Recurrent Neural Networks (RNNs) have been widely used in

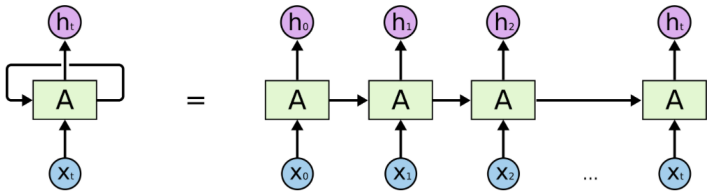

Figure 2.4: An example of a RNN and how it can be unrolled. Source: [Ola]

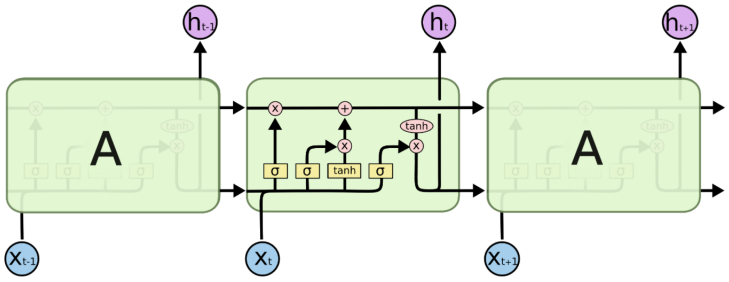

Figure 2.5: An illustration of LSTM. Source: [Ola]

research areas such as machine translation, image caption and action recognition in recent years and have achieved great success.

In order to store long term information, it has a basic looping structure of the same network, which allows long term information to persist, as shown in the left of the equation in Figure 2.4. The right side of the equation shows the unrolled version of this looping structure. It can been seen from this figure that each network sends information about itself to the next network in the chain.

The most-widely used kind of RNN is Long Short Term Memory (LSTM) networks. An example of an LSTM network can be found in Figure 2.5. Each big green block in the figure is called a cell.

Below is a detailed walkthrough about what happens inside a cell given an input x_t and the output of last cell h_t following the introduction in [Ola]. We use W and b to denote parameters for the linear transform, and σ for the activation function. $[a,b]$ denotes the concatenation of vector a and b. First, there is a forget gate deciding what content from the old material should be forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

Next, the cell state of the current cell C_t is decided by the cell state of the last cell C_{t-1}, and a candidate $\hat{C}_t$ generated by a tanh layer.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \tag{2.4}$$

Finally, based on the cell state, we decide the output h_t for the cell.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{2.6}$$

2.2 Traditional Methods for Action Recognition

Early research for video recognition before ConvNets came into fashion was largely driven by the improvement in hand-crafted feature descriptors and interest point detectors for image recognition. State-of-the-art hand-crafted features such as Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF) and Scale-invariant Feature Transform (SIFT), and successful interest point detectors such as Harris detector and Hessian detector were extended to 3D to form video representations [LL03, WTG08, KMS08]. For example, [LMSR08] first detect interest points using a space-time extension of the Harris operator. Then these interest points are described using HOG3D and HOF3D. The computed features at interest points are further encoded into bag-of-features (BoF) representations [SZ09]. A non-linear support vector machine (SVM) classifier [CV95] is applied on these representations to generate the final prediction.

In a later work, in order to compare different spatio-temporal detectors and description methods, [WUK+09] evaluate the performance of three space-time interest point detectors and six description methods, including the HOG3D, HOG/HOF descriptors and Harris3D detectors mentioned above, in three datasets, KTH ac-

tions dataset [SLC04], Hollywood2 dataset [MLS09] and UCF sports dataset [RAS08]. All detectors and descriptors are evaluated under the same bag-of-features encoding method and SVM recognition framework. Surprisingly, dense regular sampling of space-time features outperforms all interest point detectors across all datasets. As for descriptors, the combination of HOG3D/HOF3D yields the best result for the most challenging Hollywood2 dataset, which implies the important role optical flow might play in action recognition.

Another line of work describes video features using dense point trajectories instead of aggregating local video features over spatio-temporal grids. Inspired by the success of dense sampling in image classification and action recognition [WUK+09], and the importance of optical flow in time sequences, [WKSL11] propose dense trajectories where they use optical flow to track densely-sampled local features. This method is then further improved in [WS13] by estimating camera motion through the matching of frame features computed by SURF descriptors and dense optical flow.

2.3 Deep Methods for Single-person Action Recognition

Rapid improvement in the computing power of parallel machines in recent years facilitate the fast training of large CNNs on large datasets, which makes the training of models with higher representational power possible. This has been proved by work on architectures of deep neural networks, such as AlexNet [KSH12], VGG [SZ15], GoogLeNet [SLJ+15] and ResNet [HZRS16], all of which beat the result of traditional methods on ILSVRC [DDS+09] by a large margin. This vast success of the application of deep neural networks in image classification inspired several streams of work for action recognition in videos.

Now there are three major ways to model temporal information in action recognition:

- Add a recurrent layer, such as LSTM or RNN, to the image classification networks used as feature extractors. The input to the networks is a sequence of frames.

- Apply CNN on a stack of optical flow images to capture motion features. The input to the network are a stack of optical flows and a single RGB frame.

- Convert 2D convolutions in image classification networks into 3D convolutions, so that temporal structure can be learned through convolutions in the time domain. The input to the network is a stack of RGB frame images.

Later lines of work are mainly based on these three basic models [CZ17, XSH+17, FPZ16, FPW16, TRS+17, MCKA17].

Besides the attempt to better modeling temporal structure of videos, researchers also focus on other aspects of the model. [SLLG+17] replace the module for computing optical flow in previous methods with state-of-the-art networks for optical flow generation, so that optical flows are generated on-the-fly, and the whole network would be end-to-end. [CLS15, GR17] develop attention strategies so that the learned model will focus more on the parts which play more important roles in identifying the action.

Below we will discuss these streams of work on deep neural networks for action recognition mentioned above in detail.

2.3.1 Basic Models and their extensions

As mentioned above, the major difference between image classification and event recognition is that, besides spatial information, the temporal component of video provides an additional and crucial clue for the task.

Early attempts [KTS+14] use existing successful image classification networks as feature extractors for individual frames. Final predictions are pooled across the whole video. In this approach, deep neural networks are expected to learn motion related features in the first layers through an input of a stack of consecutive video frames. This proves to be difficult since information in the time domain tends to be lost after 2D convolutions and poolings in early stages of the network. Results [KTS+14] show that the results of all fusion methods are close to the result of classification on one single frame, and the learned network performs worse than state-of-the-art hand-crafted trajectory features, which means temporal structure is not captured fully in the network. This prompts the need for better modeling temporal change.

Recurrent neural networks show great potential in modeling long-term temporal dependencies by achieving state-of-the-art performances on language tasks,

such as machine translation. Inspired by the success, [DHR+15] proposed Long-term Recurrent Convolutional Networks (LRCNs), which connect recurrent sequence models to a image classification network extracting features from frames. Results imply that this architecture can be applied to numerous tasks, such as action recognition, image captioning, and natural language object retrieval, all of which achieve competitive results. This model enjoys the merit of being end-to-end and being able to accept input of various lengths.

Another possible approach for making the architecture learn modeling temporal information lies in modifications of convolution and pooling operations. In order to preserve and model information in the time domain, [TBF+15] use 3D convolutions and pooling instead of 2D in their network architecture design. Compared with LRCNs, this C3D model only accepts a fixed number of frames as inputs.

Before recurrent neural networks and 3D convolutions came along, optical flow has long been known as being capable of capturing motion between frames. So [SZ14] train a CNN on multi-frame optical flow. The output of this network, which is called motion stream or temporal stream, is then combined with the output of the ConvNet on single RGB frame through score fusion, which is referred to as appearance stream or spatial stream. Although this two-stream model achieve state-of-the-art results, optical flow needs to be extracted and stored prior to training and testing. One interesting discovery from the results is that accuracy achieved by the motion stream alone is higher than that of the appearance stream, implying that motion stream can play a more important role in action recognition than appearance stream when it is presented to the network properly.

Later work on the topic is mainly based on these three streams of basic models. [FPZ16] fuse output of the two stream model through 3D convolution and pooling at the last convolution layer instead of score fusion after softmax layer. Inspired by the success of residual networks [HZRS16], [FPW16] substitute the VGG network in two stream models with ResNet and inject skip connections from motion stream to appearance stream; [TRS+17] add residual units to C3D network. [CZ17] expand Inception-v1 [SLJ+15] to 3D, and apply this network on both appearance and motion streams. Based on this work, [XSH+17] replace 3D convolutions at the bottom of the network with 2D convolutions to achieve a better balance be-

tween accuracy and cost. As mentioned above, although two-stream models and 3D ConvNets achieve competitive results, they have the problem of only accepting input of fixed lengths, which makes them difficult to model temporal structure of longer ranges. So to solve this problem and enable the model to learn actions containing multiple stages and spanning over longer times, [WXW+16] propose dividing a video into segments, which are sparsely sampled and serve as inputs to a two stream network. Then consensus among the segments is formed through a temporal segment network.

In the previous two stream models, optical flow is computed using state-of-the-art traditional methods such as [BBPW04, CTH07]. So besides only accepting input of a fixed length, this approach would also require computing optical flow beforehand. After compressing optical flow with JPEG, the size for flow data for UCF101 is 27GB. So the size for flow data for Kinetics would be nearly 810GB. With the recent development of using ConvNets for optical flow computation, this storage space can be saved by computing optical flow using neural network flow methods such as FlowNets [DFI+15, IMS+17] and SpyNet [RB16]. According to [SZ14, IMS+17], the temporal stream trained by optical flow generated by state-of-the-art neural network flow methods and traditional methods have close classification accuracy (79.64% versus 81.2%). Besides enjoying the merit of saving storage space and having an end-to-end model, [SLLG+17] claim that through fine-tuning flow methods in order to minimize classification error instead of end-point-error (EPE), it is learning features that will be better suited for the task of action recognition. However, although results show that optical flow learned for the task of action recognition is different from traditional optical flow, there are no evident results suggesting that overall classification accuracy benefits from this end-to-end fine-tuning. This can be difficult to prove considering the doubling in time for flow generation compared with traditional methods [SZ14, IMS+17], and the two degrees more parameters FlowNet2 has than action recognition networks.

Besides the effort to optimize the structure of the networks to make them better model temporal information, researchers have also been aiming at developing attention strategies, so that the model can focus on the part of a frame which might play a more important role in identifying the action. Previous work on the generation of regions-of-interest [GDDM14] and pose estimates of the hu-

man body [YR11, NYD16, CSWS17] provides numerous possibilities for attention. [GGM15] argues that besides the person carrying out the action, contextual cues, such as scene surrounding the person or the actions of other people, can also play an important role in action recognition. So based on RCNN [GDDM14], they build an R*CNN network to encode both key person and auxiliary information from the proposed regions of interest in a image. Based on state-of-the-art pose estimators, instead of focusing on key person and auxiliary information separately, [CLS15] focus on different body parts of the person carrying out the action. They design a network that extracts temporal and spatial information from different patches of sub-images around human pose keypoints. Although these two attention mechanisms have proved to be able to boost performance, this kind of "hard attention" require labeling and detecting prior to training and testing. To solve this issue, [GR17] propose learning attention by low-rank second-order pooling, which takes place of the average pooling in ConvNets after forming the final spatial feature map. [STWH16] focus on attending the frames that play more important roles in the action recognition process.

2.3.2 Development of Datasets, Computational Costs and Beyond

Network	UCF101	HMDB51
LRCN [DHR+15]	82.9%	-
C3D [TBF+15]	85.2%	-
Two Stream [SZ14]	88.0%	59.4%
Convolutional Two Stream [FPZ16]	92.5%	65.4%
Temporal Segment Network [WXW+16]	94.2%	69.4%
I3D [CZ17]	98.0%	80.9%

Table 2.1: Accuracy of different networks

A comparison of the accuracy on HMDB51 and UCF101 using different models can be found in Table 2.1. From this table, we can see that as the model is designed so as to better model temporal structure of the video, the accuracy has drastically increased over time. However, similar to image classification, in addition to more and more complex models, the development of action recognition

has also benefited greatly from the expansion of datasets. Pre-training in action recognition on large datasets has been proved to be of great help in previous work. Experiments in [TBF+15] show that C3D pre-trained on Sports-1M improves the performance of the model on UCF101. [SZ14] discovered that using weights pre-trained on ILSVRC-2012 dataset gives the spatial stream a 30% boost in performance. Because large datasets for action recognition were unavailable at the time, to solve the issue that there were not enough examples to train temporal ConvNets, [SZ14] develop multi-task learning, so that results on HMDB51 and UCF101 could both benefit from the expansion of training examples. This drives the need to develop datasets which contain more training examples and more complex actions, such as [KTS+14] and [KCS+17]. The availability of these large datasets facilitates the training of even larger networks such as I3D [CZ17]. The accuracy on HMDB51 and UCF101 also benefits from the pre-training on these large datasets in return.

A comparison of the datasets for action recognition can be found in Table 2.2. From the table, we can see that early datasets used in the training and testing of traditional methods, such as KTH [SLC04], UCF sports [RAS08] and Hollywood2 [MLS09] are fairly small. Relatively larger datasets such as HMDB51 [HHE+11] and UCF101 [SZS12] facilitate the early development of action recognition using deep neural networks. The development of even larger datasets such as Kinetics further boost the representational power of the models.

Datasets	Number of Classes	Number of Examples
KTH actions [SLC04]	6	2,391
UCF sports [RAS08]	10	150
Hollywood2 actions [MLS09]	12	1,707
HMDB51 [HHE+11]	51	51,000
UCF101 [SZS12]	101	13,330
ActivityNet200 [HEGN15]	200	15,410
Sports1M [KTS+14]	487	1,133,158
Kinetics [KCS+17]	400	300,000

Table 2.2: Action Recognition Datasets

More complex models and larger datasets also make training more expensive

in terms of computational resources. Early models such as the original two stream model and temporal segment network usually use 4 GPUs to train, while [CZ17] use 64 GPUs for larger sizes of batches during training I3D on Kinetics.

Although results on previous work show that as the network is designed toward the goal of better modeling temporal structure of video sequences, and the expansion of dataset facilitate the training of more and more complex models, it is still unclear what these networks are capturing. [FPWZ14] study what different layers of image classification networks are learning through activation maximization, where gradient ascent is applied to the input to find the image that could increase the activation of some neurons. Inspired by this work, [FPWZ18] perform activation maximization on a four-dimensional input to find out what the network is learning. The results give some intuitions as to what the networks are learning, and can partly explain some failure cases in classification.

2.4 Deep Methods for Multi-person Event Recognition

Similar to image classification and single-person action recognition, multi-person event recognition, or group event recognition, has also enjoyed the benefit of the fast development of deep neural networks. However, unlike image classification and single-person action recognition, in a social scene that contains a number of people, such as sports games, actions of one person (or player) might have a large impact on others. Thus, besides extracting spatial features from the whole image or modeling the temporal structure of individual players, understanding and modeling the dynamics among players propose a new challenge for network structure design in group event recognition tasks.

In terms of how the actions one player take might influence others, there are mainly two aspects: changes in position and movements of body parts (or pose). Positions can easily be represented by coordinates, while there are different ways of representing the movements/changes in pose. Some possibilities include extracting features from the bounding box that contains the player using a action recognition network, such as two stream [SZ14] or [TBF+15], or representing the movement through the change in pose using state-of-the-art pose estimators such as [YR11, NYD16, CSWS17]. Individual action labels, if available, can make fea-

ture extraction of each player more accurate than directly using ConvNets trained on general action recognition datasets, which will help group activity recognition greatly. For example, in [BAF+17], there are labels of each player's action, such as moving, blocking or spiking, so researchers can aim at fine-tuning ConvNets used for feature extraction of players, or jointly predicting individual's action and group activity.

Another key issue besides what features of players to use (coordinate, output of fully connected layer of ConvNets, or pose) is how to merge/pool the features so that the dynamics among the players and the feature of the entire scene could be modeled properly. There has been much work exploring the different possibilities for modeling the dynamics among players in a sports game setting, and various pooling strategies have been developed. However, it is worth noting that since each sport has its very unique characteristics, a model designed and learned for a particular sport might be hard to generalized to other sports. So most of the models proposed so far are targeted at particular sports.

[IMD+16] develop a hierarchical LSTM model to classify events in volleyball. They represent person-level dynamics by concatenating output of fully connected layer of ConvNets and output of an LSTM tracking the player, so that both spatial and temporal features are concluded. The features of players on both teams are aggregated by max-pooling. Then dynamic of the entire scene is modeled by another LSTM. A potential problem of this model is that there might be useful information missing during the process of max-pooling.

In an effort to jointly detect players, infer their actions and predict group event in volleyball, [BAF+17] first pass each frame through a fully-convolutional network. The output is then separately passed into another fully-convolutional network for player detection, and a RNN for individual action prediction and group activity recognition.

One of our models resembles the one described in [RHAEHG16], which adopts an architecture similar to the hierarchical LSTM model for group activity recognition in basketball. But instead of max-pooling, a form of attention-pooling strategy is developed so as to identify the key player in the scene, which will contribute most to the final group activity recognition. Through this attention-pooling process, although there would still be information of the players missing, but information

from the key player in the scene can be completely preserved. The presumption is that this player's information is most crucial for identifying the group activity.

Besides bounding boxes of players, coordinates of players can also be calculated given the position of bounding boxes in the frame and the homography matrix of the frame. A detailed explanation of the homography matrix can be found in Section 3.2.3. [MZT+17] apply 1D convolution networks on concatenation of coordinates of players in a single frame, so that motion patterns for different activities can be learned. Although there has not been much literature studying how coordinates of players can influence group activity recognition, [AGR+16] proposed a social LSTM model to predict human trajectory in a social scene, which provides an interesting possibility for representing trajectories of players.

In our work, we aim at modeling the interactions among players with as little crucial information about the players missing as possible. In our first model, we use a combination of [RHAEHG16] and [AGR+16] in an attempt to identify the group activity with the key player who contribute most to the activity and at the most important locations in the rink. In our second model, position and motion information of all players are preserved in the grid feature we design. Then a CNN is used to model the dynamics in the information. Unlike max-pooling or attention-pooling, this has the benefit that not only information of all players can be preserved, but all of their interactions are modeled in the CNN as well.

Chapter 3

Problem Formulation

In this chapter, we will give a detailed introduction about the problem we will solve. We will describe the dataset along with the annotations included in the dataset, the events we are trying to classify, the scope of our research and how it can fit into more complicated tasks in a more realistic setting.

3.1 Dataset

Group activity recognition in sports has been studied by research community for many years. As a social scene, sports games are highly constrained in terms of the number of people involved, the area where they may take place, and the complicated rules they follow according to the type of sports. But they still carry many basic features of general social scenes. First of all, understanding social scenes involves analyzing all individuals in the scene as well as the environment they are in and positions they are at, which is a multi-level analysis. Second of all, social scenes involve spatial and temporal dependencies. For example, what one person does can have a great influence on what another person does; some actions can only happen when certain actions are carried out first. Fully understanding these dependencies is crucial to scene understanding. So studying group activity recognition in sports is a good first step towards social scene understanding. In our study, we focus on group activity recognition in ice hockey.

Our experiments are conducted on a dataset containing NHL ice hockey game

footage provided by SPORTLOGiQ. Each game consists of three twenty-minute periods. Information for certain periods are missing from some games, so we cannot use all three periods for all games. In our experiments, a total of 10 periods from 5 games are used.

The six group activities we are trying to classify are: *dump in, dump out, shot, pass, carry* and *puck protection*. Of the ten periods of games we use, these six events add up to 4257 examples in total. We randomly select 3,406 for training and 851 for testing. Descriptions of the six labels can be found in Table 3.1. From the distribution of the six events shown in Figure 3.1, we can see that the six classes are highly imbalanced, which is one of the challenges we are facing.

Event	Description
Dump in	A player strikes the puck into the offensive zone
Dump out	A defending player dumps the puck up the boards without targeting a teammate for a pass
Shot	A player strikes the puck in the direction of the net in order to score a goal
Pass	The possession of the puck change from one player to another player at the same team
Carry	A player travels in the rink with possession of the puck
Puck protection	A player in possession of the puck shields the puck from defender players

Table 3.1: Description of event labels

3.2 Data Preprocessing

3.2.1 Event Labels and Target Frames

In the dataset, annotation for each example contains an event label and a target frame marking the beginning of the event. Although duration of different events might be different, we make the assumption that all events are of equal length,

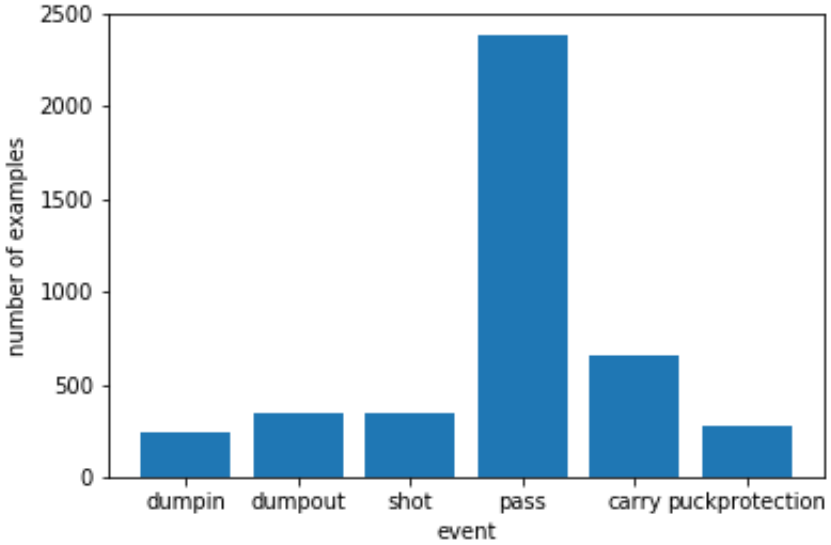

Figure 3.1: Distribution of events

since there is no annotation for the duration of the event. Based on the original two stream paper [SZ14], which uses 10 frames for feature extraction from optical flow stacks, we use ten frames for each event in our experiments as well. 5 frames before the target frame and 4 frames after are used for each event.

3.2.2 Bounding Boxes of Players

In each frame, bounding boxes of all people in the rink (players and referees) are provided in the dataset. Annotation for one player includes the position (x, y) of the player, and the width and height (w, h) of the bounding box of the player in the image coordinate system. Given this information, we can crop sub-images of players and use them as ConvNet inputs, so that we can extract features from these players. It is worth noting that these bounding box annotations can be noisy due to the fast movement of players and occlusions.

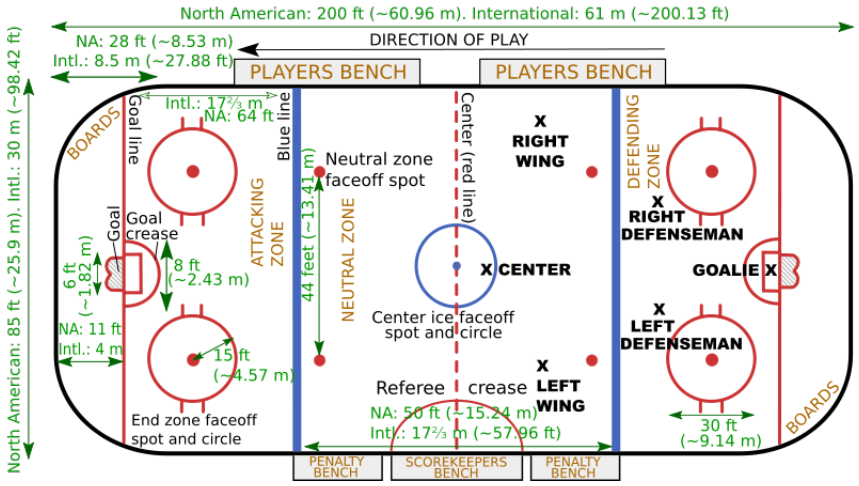

Figure 3.2: Detailed diagram of an ice hockey rink. Source: [rin]

3.2.3 Mapping from image to rink

In an ice hockey game, the rink is as shown in Figure 3.2. The rink is 85 ft in width and 200 ft in length. The two blue lines on both sides split the rink into three zones. From left to right, the three zones are attacking zone, neutral zone and defending zone. On the rink, there are 9 face-off spots in total. Eight of them are denoted in red, with two in attacking zone, defending zone, and each end of the neutral zone separately. One of them is denoted in blue and at the center of the rink. Face-off spots in attacking and defending zone are centers of red face-off circles, while the face-off spot at the center of the rink is the center of a blue face-off circle. All circles are 30 ft in diameter. At each end of the rink, there is a goal that is 11 ft away from the closer edge of the rink.

According to the description of events in Section 3.2.1, events can be highly position dependent. For example, *dump in* can only happen near the offensive zone of the attacking team, *shot* can only happen near a goal, and *dump out* can only happen near the defensive zone of the defending team. So positions of players are crucial for our task.

Because of the movement of cameras, annotations for players' location in the image coordinate are not good indications of the players' actual positions in the rink. But this information can be available if we combine bounding box annotations

with the homography matrices provided in the dataset.

A homography matrix is a 3 by 3 matrix relating two images viewing the same plane from a different angle. Because of the wide applications of homography matrix in areas such as camera calibration, image rectification and image registration, there has been much work exploring possible approaches for its estimation [GLW11, Tho07, Dub09].

Given the coordinates of a pixel (x_a, y_a) in image A, the coordinates of a pixel (x_b, y_b) in image B, and the homography matrix H that transforms pixels in image A to that in image B, the transformation between the two pixels can be denoted as:

$$\begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} = H \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} \tag{3.1}$$

In our case, (x_a, y_a) denotes the position of a player in the image coordinate system provided in the bounding boxes annotations. (x_b, y_b) denotes the position of a player in the template as shown in Figure 3.2. After projecting positions of players to the template, the *xy* coordinates in rink coordinate system can be obtained by shifting (x_b, y_b) by $(720, 32)$ and scaling by 6.95. The rink coordinate system is shown in Figure 3.3.

Example for visualization of bounding boxes of players and coordinates of players in world coordinate system calculated using homography matrix can be found in Figure 3.4

3.2.4 Team Identifications of Players

In the bounding boxes annotations we talked about above, referees in each frame are also annotated, which is unnecessary information in terms of group activity recognition. Besides the need to exclude bounding boxes of referees, team identifications can be useful in modeling the dynamics of interactions among players from different teams. For example, a *pass* event occurs between two players from the same team, while *puck protection* involves players from opposing teams. Unfortunately this information is not available in this dataset. So we label the team identifications of players in all target frames by hand.

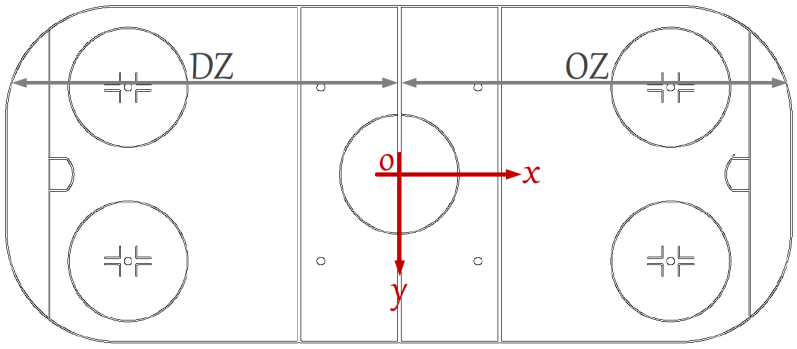

Figure 3.3: An illustration of the rink coordinate system and offensive/defensive zones

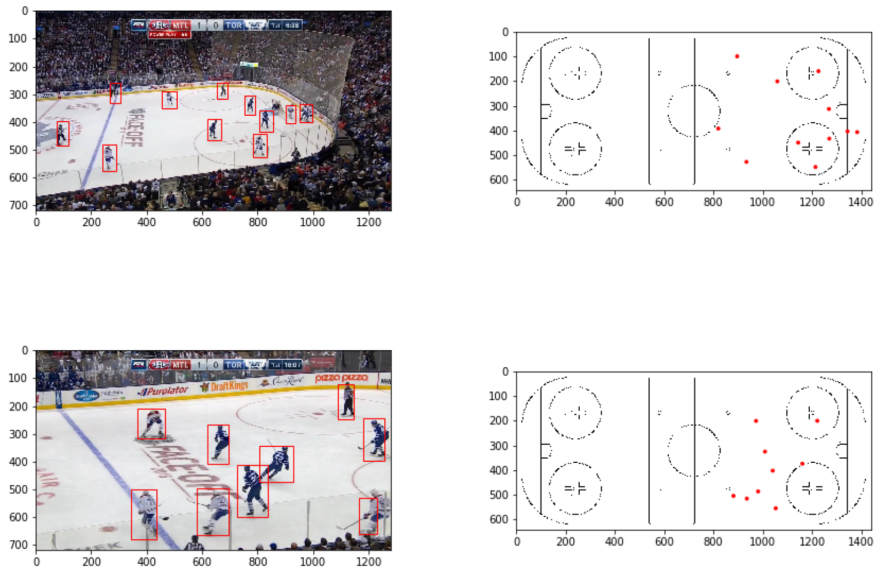

Figure 3.4: Examples for bounding box annotation of players (left side), and their transformed coordinates in rink coordinate system calculated using the homography matrix (right side).

3.3 Problem Scope

In an ice hockey game, when a new period starts, two teams will switch side. For example, in Figure 3.3, if in the first period team A attacks from left to right and defends from right to left, in the second period it attacks from right to left and defends from left to right.

In our experiments, we flip all events into the rink coordinate system shown in Figure 3.3, so that all players in all periods attack from left to right (offensive zone on the right hand side) and defend from right to left (defensive zone on the left hand side). Examples for flipping can be found in Figure 3.5.

This presents a slightly easier problem for the network, since some events can only happen in certain areas of the rink. For example, *dump in* and *shot* can only happen in the offensive zone on the right, while *dump out* can only happen in the defensive zone on the left.

Besides simplifying problem, building a model trained on flipped images also has another benefit. Since when given a raw video of a game, we will not have information regarding at which frame there is an event happening or which team is carrying out an event, event detection will be more useful in realistic settings. Based on existing literature discussing how to design an event detection network using event classification networks [EHNG16], we can easily modify our network so that it can perform event detection. So now when given a clip, suppose team A attacks from left to right, and team B attacks from right to left. We run both original clip and flipped clip through the event detection network. When an event x is detected in the original clip, we can say that team A carried out the event x; when an event y is detected in the flipped clip, we can say that team B carried out the event y.

3.4 Evaluation Metrics

Since the number of examples of different classes are highly imbalanced, we use average precision instead of accuracy as evaluation metric. In information retrieval tasks, we define precision and recall in terms of a set of retrieved documents and a set of relevant documents.

Precision evaluates how many of the retrieved documents are actually related

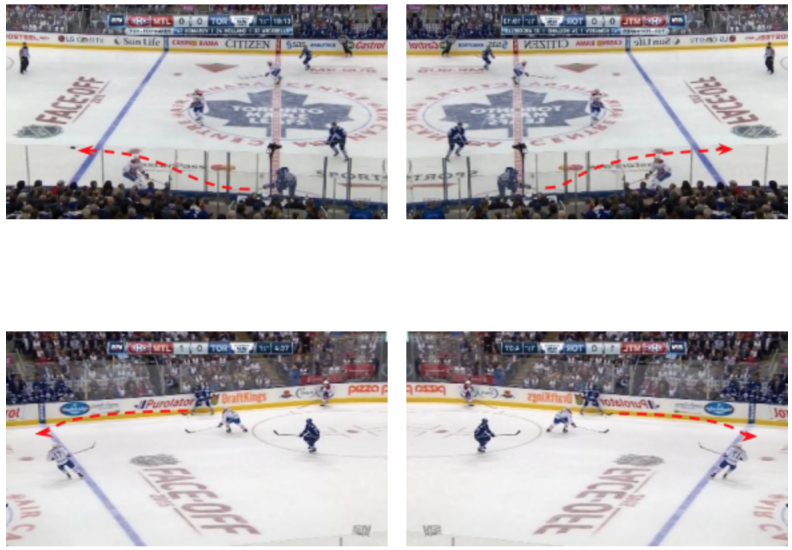

Figure 3.5: Examples of events after flipping. Upper left corner was a *dump in* by the blue player from right to left. Upper right corner is the event after flipping. Now the player dumps in (attacks) from left to right. Lower left corner was a *dump out* by the blue player from right to left. Lower right corner is the event after flipping. Now the player dumps out (defends) from right to left.

to the query:

$$P = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{3.2}$$

Recall evaluates how many of the relevant documents are retrieved eventually:

$$R = \frac{|\{\text{relevant documents}\}| \cap |\{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \tag{3.3}$$

For a list of retrieved documents in a specific order, we can have a precision-recall curve, which is precision P as a function of recall R. Average precision is calculated as:

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{3.4}$$

where P_n and R_n are the precision and recall at the nth document.

3.5 Loss Function

During training, since the number of examples across the six class labels are highly imbalanced, we use weighted softmax cross entropy loss as our loss function. Given an example, or observation o, its loss can be calculated as:

$$L = -\sum_{c=1}^{M} w_c y_{o,c} \log(p_{o,c}) \tag{3.5}$$

where c represents the predicted labels; M is the number of classes in total; y equals 1 if predicted label c is the correct classification for observation o, and equals 0 otherwise; p is the predicted probability observation o is of class c.

The weight for each class w_c is calculated as:

$$w_c = \frac{N}{N_c} \tag{3.6}$$

where N is the total number of training examples, and N_c is the number of training examples of class c.

Chapter 4

Hierarchical LSTM Model with Attention

As discussed in Section 2, there are three main approaches for modeling temporal information in videos: 3D convolution, two stream, and LSTM. In our first model, we explore using LSTM to model group activity in ice hockey. In this chapter, we will first walk through different parts of the model in Section 4.1, followed by implementation details in Section 4.2. We will present our experiment results in Section 4.3.

4.1 Model Overview

An overview of the model can be found in Figure 1.5. In our model, we use LSTM to model both scene level information and player level information, which are the actions and the trajectories of the players. When aggregating player level information, instead of max-pooling, we adopt the attention mechanism proposed in [RHAEHG16]. The major difference between our model and the one proposed in [RHAEHG16] is that, for each player, besides using the LSTM to model the player's action, we also use it to model the trajectory of the player, and the influence nearby players have on this player. In this section, we will introduce each part of our model in detail.

4.1.1 Feature Extraction

Since we do not have annotation regarding the length of each event, we make the assumption that all events have the same length in time. For each event, we use four frames before the target frame, the target frame, and five frames after the target frame, altogether ten frames, for its classification. For each frame in the event, we need feature representation for both the frame image and each of the individual players, which will be the inputs to our LSTMs modeling scene-level information and player-level information separately.

For feature extraction from frame images, we fine tune ResNet50 [HZRS16] on target frame images. Then for each frame image, we pass it through the fine-tuned network, and use the activation of the last fully connected layer as the feature representation. We denote the feature vector for a frame at time instant t as f_t.

For feature extraction from players, we fine tune ResNet50 on UCF101 [SZS12] and HMDB51 [HHE+11]. Then similar to above, we pass each bounding box through the fine-tuned network, and the activation of the last fully connected layer is perceived as the feature representation for the player. The feature vector for a player i at time instant t is denoted as $p_{t,i}$.

4.1.2 Hierarchical Temporal Component Modeling

In this section, we will denote LSTM as follows:

$$h_t = \text{LSTM}(h_{t-1}; A_1 \oplus A_2 \oplus ... \oplus A_n) \tag{4.1}$$

where h_t represents the hidden state at time t; h_{t-1} represents the hidden state at time $(t-1)$. Input to the LSTM is the concatenation of different features A_1 through A_n.

We use an LSTM to model the progression of the event:

$$h_t^a = \text{LSTM}_{event}(h_{t-1}^a; h_t^f \oplus p_t) \tag{4.2}$$

where h_t^f represents the latent representation of the frame at time t, and p_t represents the aggregation of the latent representations of the players in the frame, as described below.

Scene Level Temporal Component Modeling

At each time instant t, we derive a latent representation for the frame image using an LSTM:

$$h_t^f = \text{LSTM}_{frame}(h_{t-1}^f; f_t) \tag{4.3}$$

where h_t^f and h_{t-1}^f represents the hidden state of the frame at time t and $t-1$ separately; f_t is the feature we extracted from the frame as described above.

Player Level Temporal Component Modeling

For each player i at time t, we form a latent representation $h_{t,i}^a$ from his bounding box, and latent representation $h_{t,i}^c$ from his position in the rink coordinate system. The concatenation of these two vectors p_t is our feature representation for the player. Then the feature representations for all players at time t are aggregated into p_t through a weighted sum, which is our attention mechanism as described below.

$$p_t = \sum_{i=1}^{N} w_{t,i} h_{t,i}^p \tag{4.4}$$

$$h_{t,i}^p = h_{t,i}^a \oplus h_{t,i}^c \tag{4.5}$$

The latent representation for player i at time t is derived by the hidden state $h_{t,i}^a$ of the LSTM tracking the bounding box of the player:

$$h_{t,i}^a = \text{LSTM}_{appearance}(h_{t-1,i}^a; p_{t,i}) \tag{4.6}$$

where $h_{t,i}^a$ represents the hidden state for the bounding box of the player i at time t; $p_{t,i}$ is the feature extracted from player i at time t using the approach we mentioned above.

For modeling the trajectory of the player, we adopt the approach described in [AGR+16]. The latent representation for the trajectory of a player is described by both his own coordinate, and the trajectories of other players who are close to him:

$$h_{t,i}^c = \text{LSTM}_{trajectory}(h_{t-1,i}^c; c_{i,t} \oplus d_{i,t}) \tag{4.7}$$

where $h^c_{t,i}$ represents the hidden state for the position of the player i at time t, $c_{i,t}$ is derived from the position of player i at time t in the rink coordinate system; $d_{i,t}$ is derived from the hidden state of the players within a certain distance as described below.

The position embedding for a player i is derived as follows:

$$c_{t,i} = \phi(x_{t,i}, y_{t,i}) \tag{4.8}$$

where ϕ is a multi layer perceptron; $(x_{i,t}, y_{i,t})$ is the coordinate of the player i at time t in the rink coordinate system.

Then for each player, we form a "social hidden-state tensor" $H_{t,i}$ following the approach described in [AGR+16]. For this player i, a 2×2 grid is formed with his position $(x_{t,i}, y_{t,i})$ at the center. Then for a cell (m,n) in the grid, we find the players within. The sum of the latent representations of these players is the (m,n)th element for $H_{t,i}$. This process can be described as follows:

$$H_{t,i}(m,n,:) = \sum \mathbf{1}_{mn}[x_{t,j} - x_{t,i}, y_{t,j} - y_{t,i}] h^c_{t-1,j} \tag{4.9}$$

where $\mathbf{1}_{mn}[x,y]$ is an indicator function which returns 1 if (x, y) is in the (m,n)th cell of the grid, and 0 otherwise; $h^c_{t-1,j}$ is the latent representation for the trajectory of player j at time $t-1$.

The social hidden-state tensor $H_{t,i}$ for the player i is then embedded into $d_{t,i}$ through a multi layer perceptron:

$$d_{t,i} = \phi(H_{t,i}) \tag{4.10}$$

Attention Mechanism

When combining the features of all the players at time t, instead of max-pooling, we calculate the aggregation of features as a weighted sum of the player representations:

$$p_t = \sum_{i=1}^{N} w_{t,i} h^p_{t,i} \tag{4.11}$$

The weight for each player i is calculated through a linear layer:

$$w_{t,i} = \phi(h_t^a \oplus h_t^f \oplus h_{t,i}^p) \tag{4.12}$$

where ϕ represents the multi layer perceptron. The input is the concatenation of the latent representation of the activity h_t^a, the latent representation of the frame h_t^f, and the latent representation of the player $p_{t,i}$.

4.2 Implementation Details

The entire model is implemented using the PyTorch [PGC+17] framework. Part of the code for feature extraction from frame images and players is from the work [Hua17]. We train the model for 70 epochs with an initial learning rate of $1e-5$. We use the Adam optimizer [KB] for training and apply exponential decay. Hidden dimension for the model is 512. All hyper-parameters are chosen through grid search. The model is trained on a single NVIDIA Titan X GPU.

4.3 Results

Training our model following the protocols described above achieves an overall average precision of 50.02%. The final confusion matrix can be found in Figure 4.1.

From the confusion matrix, we can see that the most frequent class *pass* has the highest accuracy. *Dump out*, *shot* and *carry* have relatively lower accuracies, and tend to be mis-classified as *pass*. Almost all *dump in* and *puck protection* are mis-classified as the more frequent classes *pass* and *carry*.

There could be several possible reasons for the mis-classifications. First of all, although we use the weighted cross entropy loss during training, accuracies for less frequent classes are still relatively lower, indicating that more training examples are required for the model to learn the dynamics of these classes. Second of all, as the camera moves as the players move towards a certain direction, feeding a sequence of frames into LSTMs might interfere with the classifier learning the correlation between certain activities and the location in the rink they usually take place.

To verify our model design, we also conducted an ablation study. We ran the

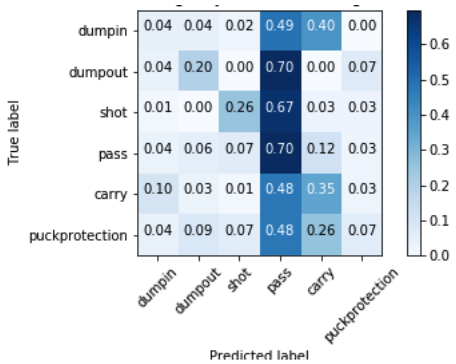

Figure 4.1: Confusion matrix for hierarchical LSTM model with attention

model with only LSTM$_{scene}$, without LSTM$_{scene}$, and using max-pooling instead of attention pooling. The results for the experiments can be found in Table 4.1. From the table, we can see that classifying on only frame images yields a low average precision, while classifying on only information of players has a competitive result compared with the full model. This indicates that scene information is useful for classification, but not to a great extent. The low average precision classifying on only scene information could be due to the fact that camera movement across time causes confusion for the classifier as to the location of each event. The results achieved by the full model is slightly higher than that achieved by max-pooling, indicating that the attention mechanism we adopted does better in combining information of players as intended.

	Average Precision
Only LSTM$_{scene}$	15.46%
Without LSTM$_{scene}$	49.23%
Max Pooling	49.62%
Full Model	50.02%

Table 4.1: Results for ablation studies

Chapter 5

Grid Feature Model

In this chapter, we will first walk through different parts of our grid feature model designed for group activity recognition in ice hockey in Section 5.1. Then we will explain our implementation details in Section 5.2. We will present our results achieved by this model in Section 5.3.

5.1 Model Overview

An overview of the model can be found in Figure 1.6. Inspired by the two stream model developed in [SZ14], we added two more streams, a spatial grid stream and a grid temporal stream. Input to the two streams are spatial grid feature and temporal grid feature. These four streams are then combined through score fusion. In the following subsections, we will discuss different parts of the model in greater detail.

5.1.1 Two Stream Model for Frame Images

The original two stream model was first proposed by [SZ14]. The authors decompose video into spatial and temporal components. They argue that the spatial component of the video carries appearance information of the video, such as what objects are in the frame, and the environment they are in. Meanwhile the temporal component captures motion information about the objects, such as the direction and velocity of the movement.

For figuring out the spatial component, using one single RGB frame image

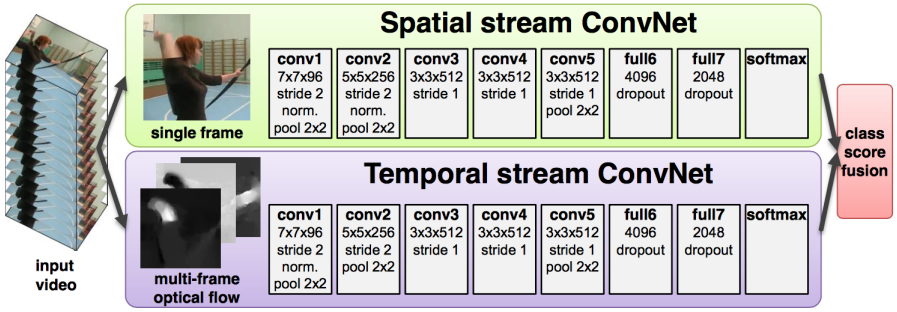

Figure 5.1: The structure of the two stream model. Spatial stream ConvNet is applied on one single RGB frame from videos, while temporal stream ConvNet operates on a stack of optical flow images. Source: [SZ14]

would be adequate. A ConvNet pre-trained on ImageNet can learn about certain features of certain objects. Fine-tuning this network on individual frame images of each event from the action recognition dataset can make the ConvNet learn the objects and scenes related to certain actions.

As for representing temporal components, there are several features extracted across several frames that might be good representations of the movements of objects. [SZ14] carried out experiments using different stacking methods of trajectories and optical flow, and came to the conclusion that a ConvNet trained using a stack of 10 optical flow images yielded the highest accuracy.

The softmax scores generated by the two ConvNets trained on spatial and temporal streams separately are then fused through score fusion. The entire structure of the model adopted in [SZ14] is as shown in Figure 5.1.

Following this structure, we train two ConvNets using our ice hockey dataset. For each event, RGB frame image of the target frame is used as input to the frame spatial stream; an optical flow stack containing four frames before the target frame up to five frames after the target frame is used as input to the frame temporal stream. We adopt ResNet50 [HZRS16] as our ConvNet structure considering its capability for reducing overfitting. We use the weights pre-trained on UCF101 and HMDB51 for frame spatial stream, and the averaged weights across channels at each layer for frame temporal stream.

5.1.2 Two Stream Model For Grid Feature

Two stream model described in Section 5.1.1 can be used to extract frame level appearance and motion features of each event, such as how many players are in the rink, and the approximate locations of the players. But it does not provide a detailed description of the appearance and motion of individual players, their exact location, and the interactions between the players. In order to solve this problem, we design a spatial grid feature and a temporal grid feature that can be used to model these information inspired by the two stream model described above. In this section, we will describe how we compute the grid features for events, and the network designed to predict labels for events given these grid features.

Feature Extraction of Players

As mentioned in 2.4, there are several possible approaches for representing the movement of players. In our work, we adopt the two stream model as described in Section 5.1.1 to extract spatial and temporal feature from individual players in each event. Since we do not have action labels for each individual player, we cannot fine-tune the network on players in the dataset. The network we used is pre-trained on general action recognition dataset UCF101 and HMDB51.

For each player in an event, the inputs for the two stream model are: the bounding box of the player at the target frame, and its optical flow images from four frames before the target frame to five frames after the target frame. The activation of the fully connected layer of spatial stream ConvNet is considered as the spatial feature of the player, while the activation of the fully connected layer of the temporal stream ConvNet is considered as the temporal feature of the player. So a player i is represented by a 2048 dimensional spatial feature vector s_i, and a 2048 dimensional temporal feature vector t_i.

Grid Feature Calculation

As discussed in Section 2.4 and Section 3.2.3, there are several key factors we should consider when doing group activity recognition. First, event label is highly position dependant. For example, in *dump in*, key player dumps the puck into his offensive zone; in *shot*, many players will be gathering near the goal. Second,

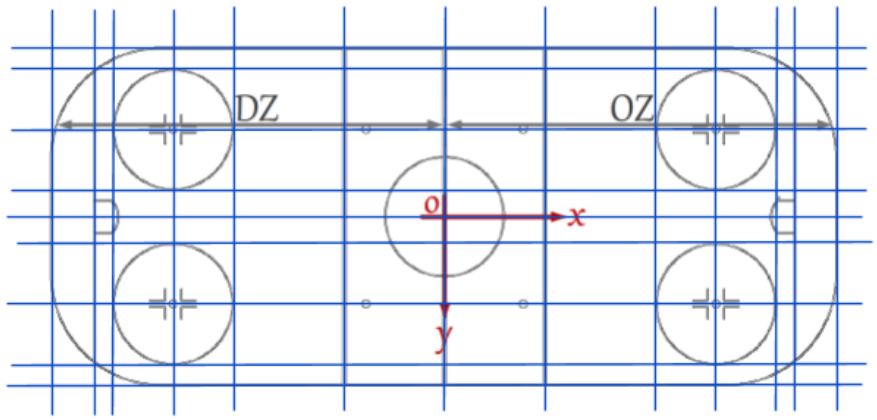

Figure 5.2: Grid layout designed for ice hockey rink. Based on the position of attacking zone, neutral zone on both sides, defending zone and the end zone faceoff spot and circle, we divide the rink into 8×12 grid cells.

the interactions between players and their team identifications are important for recognizing the group activity. For example, in *pass*, the player sends the puck in the direction of another player at the same team; in *puck protection*, two players from opposing teams will be very close to each other. In order to model these information, we develop a spatial grid feature and a temporal grid feature which preserve information about the appearance and motion feature of players, as well as the positions of these players in the rink.

We first divide the rink into 8×12 grid cells as shown in Figure 5.2. Then for each target frame of an event, we iterate through these grid cells and aggregate spatial and temporal features of the players in the cell to form a grid feature for the event. The m,n element of the grid feature is given by:

$$G_{\text{spatial}}(m,n) = \sum \mathbf{1}_{\text{mn}}[s_i] \tag{5.1}$$

$$G_{\text{temporal}}(m,n) = \sum \mathbf{1}_{\text{mn}}[t_i] \tag{5.2}$$

$\mathbf{1}_{\text{mn}}[s_i]$ or $\mathbf{1}_{\text{mn}}[t_i]$ are indicator functions checking if the player i is in the (m,n) cell of the grid. G_{spatial} and G_{temporal} are both of dimension $8 \times 12 \times 2048$.

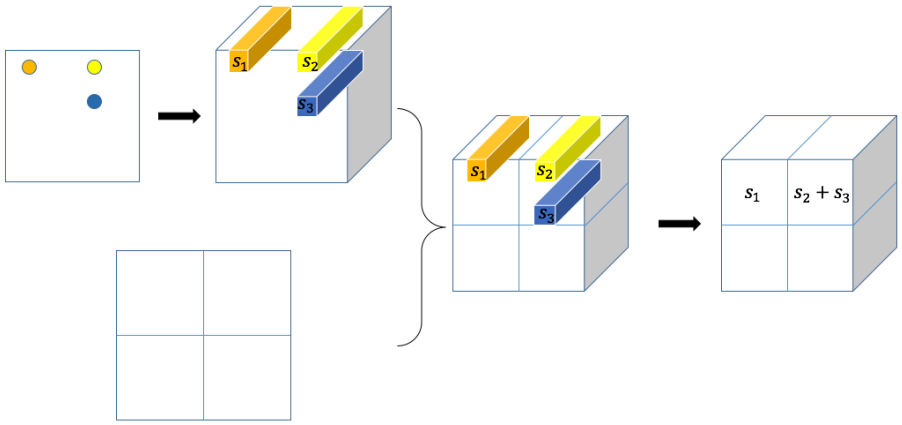

Figure 5.3: An illustration of the spatial grid feature calculation process. Given a target frame, we first calculate each player's location using homography matrix, and extract the spatial feature from players as shown in the two upper left images. Then we iterate through the grid cells in the rink as shown in the lower left image. For each cell (m,n), we find the players in the cell and sum up the features of the players. This vector is the (m,n) element of the final spatial grid feature.

As discussed before, team identifications of players are important for recognizing group activity. If we take team identification of players into consideration, at each cell, we sum up the features from both teams separately. These two sums are then combined through concatenation. So the dimensions of G_{spatial} and G_{temporal} would be $8 \times 12 \times 4096$

An illustration of the process for calculating G_{spatial} can be found in Figure 5.3.

ConvNets for Grid Features

To classify a group activity given the grid features, we design two ConvNets with the same structure as shown in Figure 5.4. The input to the two networks are G_{spatial} and G_{temporal} separately, the output are both the confidence scores of the six classes. The building block of each network are Residual Blocks [HZRS16], each composed of a convolution layer, batch normalization [IS15], and a Rectified Lin-

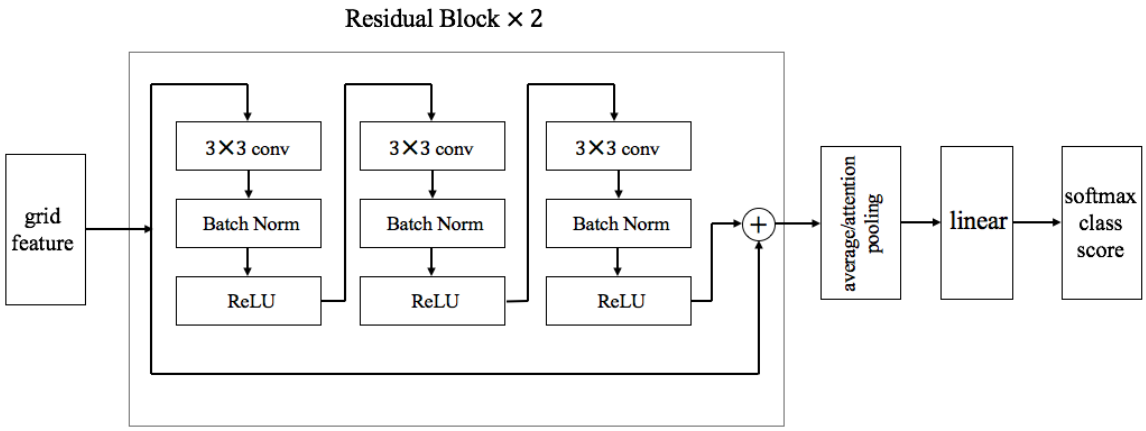

Figure 5.4: Block diagram of our grid feature model.

ear Units (ReLU) activition function [NH10]. The convolution layer is composed of 3×3 filters. The stride for the convolutions is 1, so that the width and height of the output of each convolution layer can be preserved. After each convolution, the channel of the feature volume is reduced. Output of the residual blocks is then passed through pooling and fully connected layer to generate the final class scores.

Attention Pooling

For different activities, since the location where they take place differ, we would want the model to learn different areas to focus on for different events. To learn this attention, we replace the average pooling layer in ConvNets with an attention pooling layer.

As describe above, the output of the residual block is a $8 \times 12 \times 128$ volume, with 8 and 12 representing the height and width of the grid layout of the rink. Each of the 128 channels of the volume is a 8×12 slice denoted by A. In attention pooling, output of the channel is $\frac{1}{n}\sum_i \sum_j A_{i,j}$. In attention pooling, we learn two sets of weights w_1 and w_2. The dimensions of the weights are 1×8 and 1×12. Output of the channel is calculated as $\frac{1}{n} w_2 (w_1 A)^T$.

Fusion Method

From the frame-level two stream model and the two stream grid feature model described above, we will have four sets of softmax scores. Through these four sets of scores, we make our final prediction through score fusion:

$$p = \text{argmax} \sum_{i=1}^{4} w_i \tag{5.3}$$

w_1, w_2, w_3 and w_4 denote the four sets of scores generated by the four streams separately.

5.2 Implementation Details

The entire model is implemented under the PyTorch [PGC+17] framework. Part of the code used for frame-level two stream network is inspired by [Hua17]. For optical flow estimation, we use the approach described in [CTH07]. The frame-level two stream model is pre-trained on ImageNet, and fine-tuned on UCF101 and HMDB51. This pre-trained model is also used for extracting spatial and temporal features from players. We further fine-tune the model on our ice hockey dataset with an initial learning rate of $5e-4$ and apply exponential decay. For the two stream grid feature model, we use an initial learning rate of $1e-4$, and apply exponential decay. A batch size of 16 is used in all parts of the model. All hyper-parameters are chosen through grid search. All parts of the model are trained using the Adam optimizer [KB], and trained on a single NVIDIA Titan X GPU.

5.3 Results

Fusion Among All Streams

Figures 5.5, 5.6, 5.7 and 5.8 show the confusion matrices for the classification results of the four streams separately. From the confusion matrices, we can see that frame temporal stream does better in more frequent classes such as *pass* and *carry*; while frame spatial stream, grid spatial stream and grid temporal stream do better in less frequent classes such as *dump in*, *dump out* and *shot*. This difference in

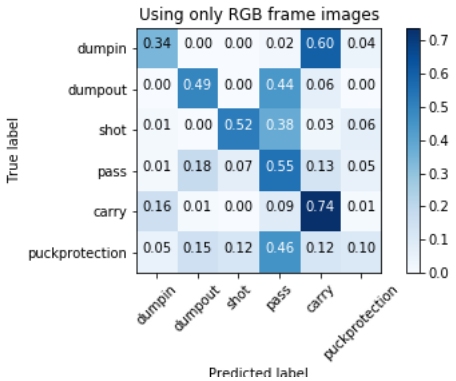

Figure 5.5: Confusion matrix for classifying on RGB image of the target frame of each event

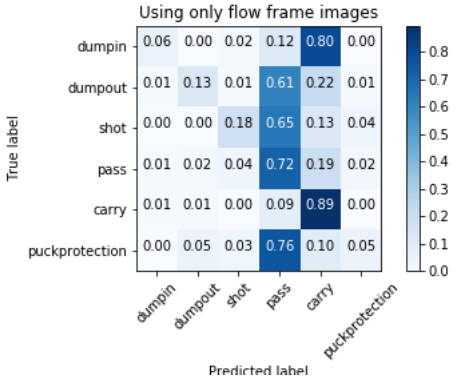

Figure 5.6: Confusion matrix for classifying on a stack of optical flow images in each event ranging from 4 frames before the target frame to 5 frames after the target frame

per-class accuracy implies that fusion among the four streams can further improve accuracy.

Table 5.1 shows the average precision of the four streams separately, fusion between the two frame streams, fusion between the two grid streams, and the fusion among all four streams. From the table, we can see that among the four streams, frame temporal stream has relative lower average precision. This indicates that the performance of the model is not well balanced across classes, which can also be seen from its confusion matrix in Figure 5.6.

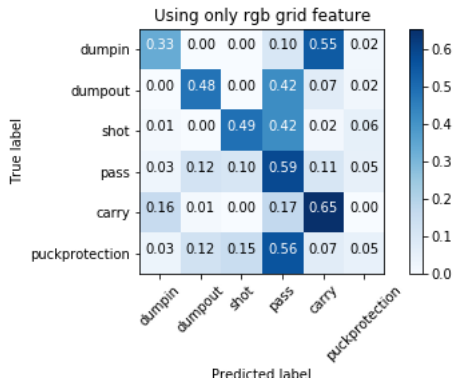

Figure 5.7: Confusion matrix for classifying on G_{spatial}

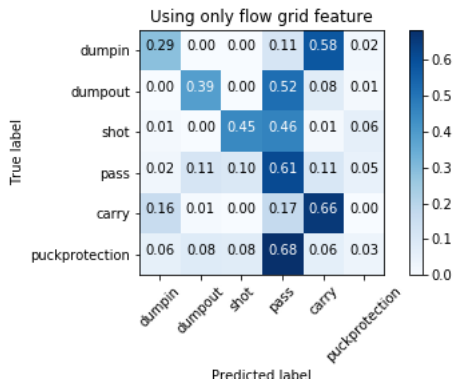

Figure 5.8: Confusion matrix for classifying on G_{temporal}

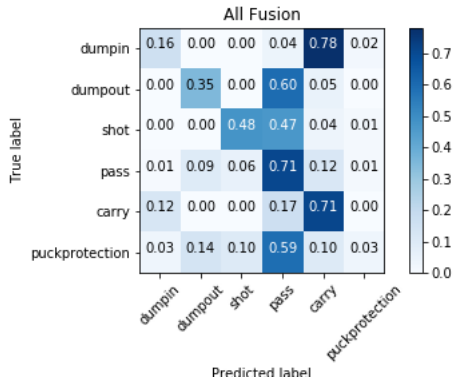

Figure 5.9: Confusion matrix for score fusion among all streams

	Average Precision
Frame Spatial	45.52%
Frame Temporal	41.07%
Grid Spatial	47.33%
Grid Temporal	46.18%
Frame Streams Fusion	49.27%
Grid Streams Fusion	48.18%
All Streams Fusion	50.71%

Table 5.1: Accuracy and average precision for each stream and different fusion methods

	Average Precision
Fine-tuned C3D [Meh15] (Unknown key player)	37.87%
Trajectory [Meh15] (Unknown key player)	39.54%
Fine-tuned C3D [Meh15] (Known key player)	48.21%
Trajectory [Meh15] (Known key player)	50.08%
Grid Streams Fusion (Ours)	48.18%
Frame Streams Fusion (Ours)	49.27%
Four Streams Fusion (Ours)	50.71%

Table 5.2: Comparison with existing models.

Although the fusion between the two frame streams and the fusion between the two grid streams both improve average precision compared to the four streams individually, fusion among the four streams gives an overall highest average precision. Fusion among all streams improves average precision by 3.4% compared with the highest of individual stream. Adding the two grid streams to the original two stream model improves average precision by 1.5%.

Table 5.2 compares our model with C3D [TBF+15] and trajectory representations for players developed in [MZT+17]. [MZT+17] provides both results under conditions where key players in the scene are known and unknown, while we do not use information about key players in our model. From the table we can see that when key players are unknown, our model outperforms C3D and trajectory representation by a large margin. Our model still outperforms both models by a small margin when these two models are provided with the information on key players.

	Grid Spatial	Grid Temporal	Fusion
w/o attention	46.59%	44.67%	50.41%
w/o team id	46.59%	45.18%	49.63%
Ours	47.33%	46.18%	50.71%

Table 5.3: Ablative analysis

Figure 5.9 shows the confusion matrix for fusion among all streams. From the confusion matrix, we can see that the less frequent classes *dump in*, *dump out*, *shot* and *puck protection* have a much lower accuracy than the more frequent classes *pass* and *carry*. This indicates that, although we use the weighted cross entropy loss during training, accuracies for the less frequent classes still suffer from the inadequacy of training examples. Among the less frequent classes, we can see that the most misclassified class is *puck protection*. Besides inadequate number of training examples, this also could be due to the fact that during *puck protection*, there would usually be occlusion between players, so the bounding box annotations can be highly inaccurate. The less frequent classes *dump out* and *shot* are mostly misclassified as *pass*. Since we are only looking at 10 frames around the target frames, which is approximately 0.3 seconds, the puck might not have reached its target. For example, the puck has not reached the blue line marking the defensive zone in *dump out*, or the puck has not reached the goal in *shot*.

Ablation Studies

We performed an ablative analysis to verify the need to form G_{spatial} and G_{temporal} based on team identifications and the need to perform attention pooling instead of average pooling. The results can be found in Table 5.3. From the table, we can see that both removing team identifications and removing attention pooling lead to a decrease in average precision in grid spatial stream, grid temporal stream, and fusion of all streams. Between these two factors, removing team identifications lead to a larger decrease in average precision.

Chapter 6

Conclusions

In this work, we focused on the group activity recognition task in an ice hockey dataset with bounding boxes and positions of players available in each event. Because of the lack of information in individual action labels and puck positions, the problems we aimed at providing possible solutions for are:

- How to combine scene-level information and player-level information more effectively
- How to incorporate appearance and motion features of players with their positions more effectively
- How to combine features of all players in a scene, aiming at losing minimum information, and modeling the interactions among players more effectively

To solve these problems, we proposed two models: a hierarchical LSTM model based on LSTM used for single-person action recognition; a four stream model based on two stream model used for single-person action recognition. Both of these models achieved competive results on the ice hockey dataset.

In the first model, scene-level information and player-level information are combined through concatenation of the hidden states of LSTMs; appearance feature and the trajectories of players are combined through concatenation of hidden states of LSTMs, with interactions among players modeled by a "social state tensor"; features of all players in a scene is combined through an attention mechanism,

so that more crucial information of the players can be kept and considered during classification.

In the second model, scene-level information and player-level information are combined through score fusion among different classification networks; appearance and motion features of players are incorporated with their positions through a grid feature we designed; features of different players are combined through a CNN, during which process the interactions among players can also be considered.

Results achieved with these two models and comparison with previous work can be found in Table 6.1. [Meh15] provides results under conditions when key player in the scene are known and unknown separately, while we do not use key player annotation in our model. From the table, we can see that both models outperform C3D and trajectory modeling by a large margin. Our model still have competitive results when C3D and trajectory model are trained with key player annotation. Of these two models, Model 2 achieved a slightly higher overall accuracy. When provided with only information of players, the two models achieve similar results. However, Model 1 performs much worse when trained only on scene information. One possible explanation is that for Model 2 we only use the target frame for classification, which contains information regarding the location of the event. But this information could be missing in Model 1, which is trained on a consecutive sequence of frames, due to camera motion. This might indicate the need to deal with camera motion when feeding the network with a consecutive sequence of frames.

Using state-of-the-art object detection and homography matrix estimation algorithms, we can easily get information about the bounding boxes of players, and their position in the world coordinate system. With these information available, we can apply our models to group activity recognition tasks in other sports, or general social scenes as well.

To test if our models can be applied with more flexibility to group activity recognition in other sports and other tasks in a more realistic setting, there are several possible future directions to this work:

- Experiment with other descriptors to represent the appearance and motion features of players, such as pose

	Average Precision
Fine-tuned C3D [Meh15] (Unknown key player)	37.87%
Trajectory [Meh15] (Unknown key player)	39.54%
Fine-tuned C3D [Meh15] (Known key player)	48.21%
Trajectory [Meh15] (Known key player)	50.08%
Model 1 with only player information (Ours)	49.23%
Model 1 with only scene information (Ours)	15.46%
Model 1 full model (Ours)	50.02%
Model 2 with only player information (Ours)	48.18%
Model 2 with only scene information (Ours)	49.27%
Model 2 full model (Ours)	50.71%

Table 6.1: Results of our models and comparison to previous work

- Test our model on group activity recognition tasks in other sports, such as volleyball and basketball
- Expand our model to perform group activity detection instead of recognition, using methods similar to the one described in [EHNG16]

Bibliography

[AGR+16] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: human trajectory prediction in crowded spaces. 2016. → pages 2, 24, 36, 37

[BAF+17] Timur Bagautdinov, Alexandre Alahi, Francois Fleuret, Pascal Fua, and Silvio Savarese. Social scene understanding: end-to-end multi-person action localization and collective activity recognition. 2017. → pages 2, 3, 23

[BBPW04] Thomas Brox, Andres Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. 2004. → page 19

[CLS15] Guilhem Cheron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. 2015. → pages 17, 20

[CSWS17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. 2017. → pages 20, 22

[CTH07] C.Zach, T.Pock, and H.Bischof. A duality based approach for realtime tv-l1 optical flow. 2007. → pages 19, 46

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector network. 1995. → page 15

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. pages 303–314, 1989. → page 12

[CZ17] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. 2017. → pages 17, 18, 20, 21, 22

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Lijia Li, Kai Li, and Feifei Li. Imagenet: a large-scale hierarchical image database. 2009. → pages ix, 1, 16

[DFI+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, and Vladimir Golkov. Flownet: learning optical flow with convolutional networks. 2015. → page 19

[DHR+15] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. 2015. → pages 2, 18, 20

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. 2005. → pages 1, 10

[Dub09] Elan Dubrofsky. Homography estimation. 2009. → page 29

[EHNG16] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: deep action proposals for action understanding. pages 768–784, 2016. → pages 31, 53

[FPW16] Christopher Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal residual networks for video action recognition. 2016. → pages 2, 17, 18

[FPWZ14] Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes, and Andrew Zisserman. Deep inside convolutional networks: visualizing image classification models and saliency maps. 2014. → page 22

[FPWZ18] Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes, and Andrew Zisserman. What have we learned from deep representations for action recognition? 2018. → page 22

[FPZ16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. 2016. → pages 17, 18, 20

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014. → pages 19, 20

[GGM15] Georgia Gkioxari, Ross Girshick, and Jitendra Malik. Contextual action recognition with r*cnn. 2015. → page 20

[GLW11] Ankur Gupta, James J. Little, and Robert J. Woodham. Using line and ellipse features for rectification of broadcast hockey video. 2011. → page 29

[GR17] Rohit Girdhar and Deva Ramanan. Attention pooling for action recognition. 2017. → pages 17, 20

[HEGN15] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: a large-scale video benchmark for human activity understanding. 2015. → page 21

[HHE+11] H.Kuehne, H.Jhuang, E.Garrote, T.poggio, and T.Serre. Hmdb: a large video database for human motion recognition. 2011. → pages 2, 21, 35

[Hua17] Jeffery Huang. Two stream action recognition. 2017. → pages 38, 46

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2016. → pages 13, 16, 18, 35, 41, 44

[IMD+16] Mostafa S. Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. Hierarchical deep temporal models for group activity recognition. 2016. → pages ix, 3, 6, 23

[IMS+17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet2.0: Evolution of optical flow estimation with deep networks. 2017. → page 19

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. 2015. → page 44

[Kara] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition - convnets. → pages x, 12, 13

[Karb] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition - neural networks. → pages x, 11

[KB] Diederik P. Kingma and Jimmy Ba. → pages 38, 46

[KCS+17] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. 2017. → page 21

[KMS08] Alexandar Klaser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. 2008. → pages 1, 10, 15

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012. → pages ix, 1, 10, 13, 16

[KTS+14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei-Fei Li. Large-scale video classification with convolutional neural networks. 2014. → pages 17, 21

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. pages 2278–2324, 1998. → page 13

[LL03] Ivan Laptev and Tony Lindeberg. Space-time interest points. 2003. → pages 10, 15

[LMSR08] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. 2008. → pages 1, 15

[Low04] David Lowe. Distinctive image features for scale-invariant keypoints. pages 91–110, 2004. → pages 1, 10

[MCKA17] Chih-Yao Ma, Min-Hung Chen, Zsolt Kira, and Ghassan AlRegib. Ts-lstm and temporal-inception: exploiting spatiotemporal dynamics for activity recognition. 2017. → page 17

[Meh15] Nazanin Mehrasa. Learning person trajectory representations for team activity analysis. 2015. → pages 49, 52, 53

[MLS09] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. 2009. → pages 16, 21

[MZT+17] Nazanin Mehrasa, Yatao Zhong, Frederick Tung, Luke Bornn, and Greg Mori. Deep learning of player trajectory representations for team activity analysis. 2017. → pages 24, 49

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. 2010. → page 45

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. 2016. → pages 20, 22

[Ola] Christopher Olah. Understanding lstm networks. → pages x, 14

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. → pages 38, 46

[RAS08] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. 2008. → pages 16, 21

[RB16] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. 2016. → page 19

[RHAEHG16] Vignesh Ramanathan, Jonathan Huang, Sami Abu-El-Haija, and Alexander Gorban. Detecting events and key actors in multi-person videos. 2016. → pages 3, 5, 6, 23, 24, 34

[RHHD56] N. Rochester, J. Holland, L. Haibt, and W. Duda. Tests on a cell assembly theory of the action of the brain, using a large digital computer. pages 80–93, 1956. → page 11

[rin] Ice hockey rink. → pages xi, 28

[Ros58] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. page 386, 1958. → page 11

[SLC04] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. 2004. → pages 16, 21

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2015. → pages 13, 16, 18

[SLLG+17] Laura Sevilla-Lara, Yiyi Liao, Fatma Guney, Varun Jampani, Andreas Geiger, and Michael J. Black. On the integration of optical flow and action recognition. 2017. → pages 17, 19

[STWH16] Yemin Shi, Yonghong Tian, Yaowei Wang, and Tiejun Huang. Joint network based attention for action recognition. 2016. → page 20

[SZ09] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. 2009. → page 15

[SZ14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. 2014. → pages xi, 2, 6, 18, 19, 20, 21, 22, 27, 40, 41

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015. → pages 13, 16

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: a dataset of 101 human action classes from videos in the wild. 2012. → pages ix, 2, 21, 35

[TBF+15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. 2015. → pages 2, 18, 20, 21, 22, 49

[Tho07] Graham Thomas. Real-time camera tracking using sports pitch markings. pages 117–132, 2007. → page 29

[TRS+17] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. 2017. → pages 2, 17, 18

[WKSL11] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. 2011. → page 16

[WS13] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. 2013. → page 16

[WTG08] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. 2008. → pages 10, 15

[WUK+09] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. 2009. → pages 15, 16

[WXW+16] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: towards good practices for deep action recognition. 2016. → pages 19, 20

[XSH+17] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: speed-accuracy trade-offs in video classification. 2017. → pages 17, 18

[YR11] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. 2011. → pages 20, 22