Learning Person Trajectory Features for Sports Video Analysis

by

Yatao Zhong

B.Sc., University of Science and Technology of China, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Approval

Name:	**Yatao Zhong**
Degree:	**Master of Science (Computing Science)**
Title:	***Learning Person Trajectory Features for Sports Video Analysis***
Examining Committee:	**Chair:** Mark Drew Professor

Greg Mori
Senior Supervisor
Professor

Brian Funt
Supervisor
Professor

Luke Bornn
Internal Examiner
Assitant Professor
Department of Statistics and Actuarial Science

Date Defended: 9 January 2017

Abstract

We propose a generic deep model to learn features describing person trajectories. This network uses layers of 1D temporal convolutions over person location inputs. The network can model the patterns of motion exhibited by people when performing different activities. These trajectory features are used in a two-stream deep model that takes as input both visual data and person trajectories for sports video analysis. Our model utilizes one stream to learn the visual temporal dynamics from video clips and the other stream to learn the space-time dependencies from trajectories. We evaluate our trajectory feature learning model on data from NBA basketball games. We also utilize a dataset from NHL hockey games, which contains broadcast videos and uses state of the art automatic camera calibration, human detection, and tracking algorithms to estimate player positions in world coordinates. Experiments show that person trajectories can provide strong spatio-temporal cues, which improve performance over baselines that do not incorporate trajectory data.

Keywords: Trajectory Feature; 1D Convolution; Two-Stream Neural Network; Sports Video Analysis

Acknowledgements

I would first like to thank my supervisor Dr. Greg Mori for his unreserved support, patience and encouragement during the course of this project. There have been times when I nearly gave up but Greg kept encouraging me and guiding me through the obstacles. I am really grateful for his great support on my research. In addition, I would also like to thank Greg for always encouraging me (as well as other students) to enjoy life and draw a clear line between life and work.

I also deeply grateful to Nazanin Mehrasa who gave me a great help throughout this project. I want to thank her for joining my project with her own thesis project set aside, helping me labeling the data without hesitation and sharing her ideas when I got stuck. Without her generous help I will not be able to finish my thesis.

I would also like to express my gratitude to Dr. Luke Bornn who provided me with deep insights on various aspects of trajectory analytics that have a great effect on my research.

I have sincere thanks to Dr. Brian Funt and Dr. Mark Drew, the members of my supervisory committee, for their helpful suggestions and insightful comments.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Human activity analysis is a fundamental problem in computer vision. The trajectory a person takes while moving can provide vital information to conduct this analysis. For example, the path followed by a walking person would look very different from that of a basketball player dribbling around an opponent. In this paper we develop a deep network for learning representations of person trajectories for activity analysis.

Consider the example in Fig. 1.1. Determining what event is taking place in this scene can be done based on the visual appearance of the people in the scene, augmented by descriptions of their trajectories over time. A large volume of work has focused on visual features for recognizing individual actions. These are typically built from challenging unconstrained internet video datasets such as UCF Sports [21], UCF-101 [25], HMDB-51 [14], and Sports-1M [12]. These datasets have been used to learn powerful feature descriptors (e.g. C3D [26]), which we will leverage in our models.

A body of literature focuses on group activity and human interaction [16, 1, 15, 20, 2, 13, 5, 4, 10], some of which incorporate spatial information of individuals. However, these representations tend to be hand-crafted and do not sufficiently encode the rich information of individual person movements and their interactions over time.

In this paper, we address the problem of analyzing the behaviors of players in sports video. Sports video analysis presents numerous challenges. First, players move extremely fast and often frames are blurred due to this rapid movement. Thus, the input video clips do not always carry the rich visual information expected. Second, sports video, especially team sports, contains lots of player interactions. Interpreting those interactions can help understand their activities as a group, but the representations used to decode such interactions remains an open challenge.

We propose a method to analyze player activities in team sports by using their locations in world coordinates over time. For example, consider a snapshot of time in a hockey game. It is hard to determine what a crowd of players is doing by simply inspecting the pixels inside the bounding boxes of players in a video clip. It is helpful to analyze the relative

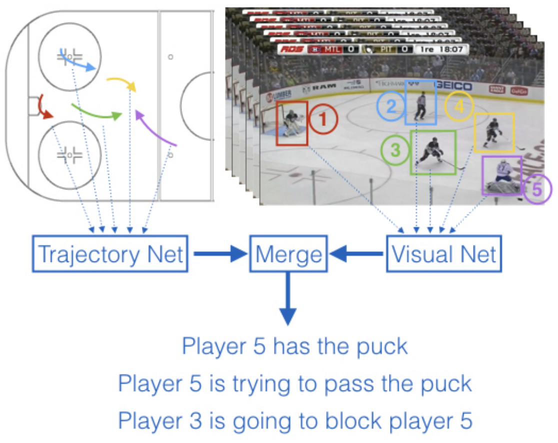

Trajectory Net
Merge
Visual Net
Player 5 has the puck
Player 5 is trying to pass the puck
Player 3 is going to block player 5

Figure 1.1: Combining visual features with trajectory features for human behaviour analysis in videos. We develop a deep network for learning features from person trajectories. These can be combined with visual features for analyzing human activities.

positions of players over time as well, so as to understand the spatio-temporal context and then understand the behavior of each player.

Analysis of trajectory data is not a new area of research. Many works have proposed computational methods to analyze player behaviors using trajectories. We refer readers to [8] for a recent survey on this field of research. However, to the best of our knowledge, we are the first to learn features describing trajectories with convolutional neural networks and combine both video data and trajectories in a single framework for activity analysis.

We implement this network in a two-stream framework that handles multiple persons. For each person, the data in two domains (video and trajectory) go through a network shared by all people and then the outputs of each person are merged into a single vector as the eventual feature representation. To demonstrate the generality of our model, we analyze team formations, puck carrier detection, and event recognition across NBA trajectory and NHL video datasets.

Although in this paper we only addressed the problem of analyzing player behaviours in sports videos, our model is general and can be applied to various situations. In terms of the two-stream framework, one can augment any existing action recognition model by considering trajectory data as an extra cue. In terms of the trajectory stream, applying 1D convolutions to learn trajectory features also has broad applications. One case could be detecting the abnormalities in surveillance videos. Usually the trajectory pattern of people and vehicles perceived by a surveillance camera form a distribution. If at some time a traffic accident happens, this could disrupt the usual pattern, leading to abnormal and deviated trajectories. We can use proposed 1D temporal convolutions to handle this task. In fact, the proposed trajectory model is capable of learning the patterns of temporal dynamics of any feature points defined by users. For example, a feature point can be a mass point when an object is treated as a whole, as in the case of player trajectory analysis. A feature point can also be a part of an object like a body joint.

The contribution of this work is two-fold. First, we propose a novel deep neural network for behavior analysis for a group of people, which is capable of capturing spatio-temporal dependencies from visual appearance and person trajectories. Second, experiments show that our model outperforms baseline models that do not use trajectory data, which supports our conjecture that person position over time plays an important role when analyzing human activities.

Chapter 2

Related Work

The existing literature on analyzing human activities is extensive. Thorough surveys of earlier work include Gavrila [7] and Weinland et al. [28]. Below, we review closely related work in activity recognition, including individual actions, group multi-person activities, and trajectory analysis.

2.1 Deep Learning for Action Recognition

Recently, deep learning has been brought to bear on the problem of action recognition. Approaches for video-based action recognition include the two-stream network of Simonyan and Zisserman [24], which fuse motion and appearance feature branches into a single network. Karpathy et al. [12] did extensive experiments on when and how to fuse information extracted from video frames. Donahue et al. [6] extract features from each frame and encode temporal information using a recurrent neural net (LSTM [9]) for action recognition. Tran et al. [26] extended traditional 2D convolution to the 3D case, where filters are applied to the spatial dimensions and temporal dimension simultaneously.

2.2 Group Activity Recognition

Group activity recognition examines classifying the behaviour of multiple, interacting people. Effective models typically consider both individual actions and person-level interactions within the group. Previous works use hand-crafted features and model interactions with graphical models. Choi et al. [3] build hand-crafted descriptors of relative human poses. Lan et al. [16] and Amer et al. [1] utilize hierarchical models to understand collective activity among a group of people at different levels, ranging from atomic individual action to group activity in the scene. The concept of social roles performed by people during interactions has also been studied [15, 20]. All of these methods use hand-crafted representations of inter-person relationships.

Another line of work introduces structures into deep learning frameworks by integrating neural networks and graphical models in a unified framework [22, 29, 23]. For example, Deng et al. [4, 5] apply deep structured models to collective activity recognition, learning dependencies between the actions of people in a scene. However, these works do not consider spatio-temporal relationships between participants, which we believe would provide strong indication about how a group activity is formulated. Thus, we propose a model to incorporate spatial information by learning the dynamics of trajectories of each participant as well as their relative movements.

The incorporation of track-level features as extra cues for interaction modeling was done by Choi et al. [2] and Khamis et al. [13]. Recent work has developed more sophisticated deep temporal models for activity analysis. Ramanathan et al. [19] utilize attention models to focus on key players in sports activties. Ibrahim et al. [10] build hieararchical LSTMs to model multiple interacting people over time. In contrast with this, our work learns trajectory features directly from human position inputs.

2.3 Trajectory Data Analytics

There exists significant literature on trajectory analysis focusing on team sports, such as basketball, soccer, and hockey. Applications within sports analytics include analyzing player and team performance, and mining underlying patterns that lead to certain results. Work in this field has included various statistical models to capture the spatio-temporal dynamics in player trajectories. We refer readers to a recent survey [8] on detailed team sports analysis with trajectory data.

Classic examples in the vision literature include Intille and Bobick [11] who analyzed American football plays based on trajectory inputs. Médioni et al. [18] utilized relative positioning between key elements in a scene, such as vehicles and checkpoints, to recognize activities.

In the sports context, Lucey et al. [17] use a basis representation of person trajectories that utilizes roles. The work that is most similar to ours is Wang and Zemel [27], which uses a traditional CNN plus RNN to classify NBA offensive patterns. Person trajectories are converted to an image representation for input to the CNN. However, an image representation of trajectories is not an effective coding mechanism in that time is ignored, and further most pixels are zeros, making this representation redundant. We instead propose to build a neural network directly on top of raw trajectories.

Chapter 3

Trajectory-Visual Model

Our model is built in a two-stream framework, one analyzing trajectory information, the other direct visual appearance. Each stream takes incoming data in different domains as input. In our model, video clips and person trajectories are fed into the network. For ease of elaborating our model, we call these two streams the visual stream and trajectory stream respectively. To integrate the two streams into a unified framework, we build a combined feature vector by concatenating the output of each stream. This is followed by a fully connected layer(s) for classification.

Since we aim to analyze human behaviour as a group, there are multiple people to handle per sample, each requiring a separate neural network for feature extraction. To this end, we let all people share the same network for feature extraction. Afterwards we merge the features of all individuals and treat the concatenated feature as the resulting representation. In this section, we will delve into the details of the model formulation, starting with the structure of each stream, followed by the construction of the model for a single person and finally the architecture of our model for a group of people in a scene.

3.1 Trajectory Stream

We start by describing the analysis of the trajectory of one person in a scene. The input to the trajectory stream is a sequence of person world coordinates in the form of (x_t, y_t), where t is frame number. These inputs are obtained via state of the art tracking and camera calibration systems, which provide reasonably accurate, though sometimes noisy, data. To learn the space-time variations in person trajectories, we propose to use 1D convolutions.

Recall that a person trajectory is essentially a continuous signal. We propose a direct way of interpreting a trajectory. A 2D trajectory in world coordinates (player position in court / rink coordinates) has two separate continuous signals, one for the x series and one for y series. We can split the input $[(x_1, y_1), (x_2, y_2), \cdots, (x_T, y_T)]$ into two sequences $[x_1, x_2, \cdots, x_T]$ and $[y_1, y_2, \cdots, y_T]$, each being a 1D continuous signal. In our approach

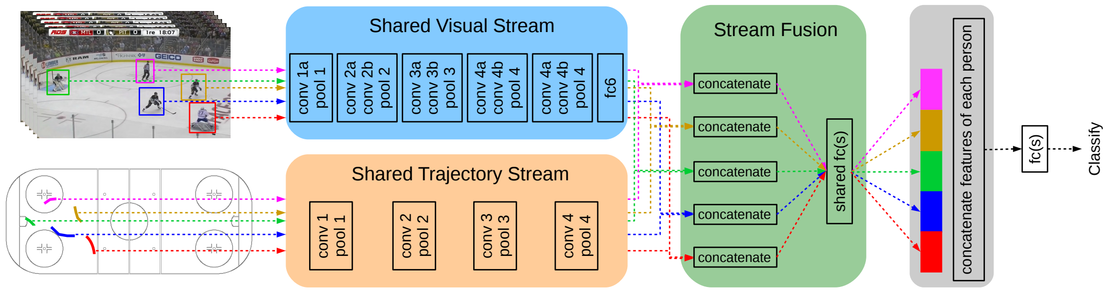


Figure 3.1: Proposed two-stream network for video analysis.

we treat these two sequences as two channels. We build a convolutional neural network on top of these inputs, with 1D convolution operating on each input. By stacking layers of 1D convolution, we can learn combinations of x and y movements that are indicative of particular action classes.

In detail, let $X \in \mathbb{R}^{N \times T}$ denote the input, $F \in \mathbb{R}^{N \times W \times M}$ denote the filters in a convolutional layer and $O \in \mathbb{R}^{M \times T}$ denote the output, where N is the number of input channels, T is the length of input sequence, W is the filter size and M is the number of filters. To model the behaviour of a convolutional layer[1], we do the basic operation as follows:

$$O_{k,t} = \sigma(\sum_{i=1}^{N} \sum_{j=1}^{W} X_{i,t+j-1} F_{i,j,k}). \tag{3.1}$$

In the above formula, $\sigma(\cdot)$ can be any activation function. In our case, we choose ReLU for all activations. Each convolutional layer is followed by a max pooling layer to make the model shift-invariant and help reduce the dimension of the output.

Let $Z \in \mathbb{R}^{M \times \lceil \frac{T}{S} \rceil}$ be the output of max pooling, where S is the step size in the pooling operation, then we have

$$Z_{k,t} = \max_{1 \le j \le S} O_{k,\ (t-1)\cdot S+j}. \tag{3.2}$$

To build a network with stacked convolutional and max pooling layers, we use the output Z^{l-1} at layer $l-1$ as the input X^l at layer l:

$$X^l = Z^{l-1}. \tag{3.3}$$

We repeat the process described in Eq. 3.1 and Eq. 3.2 for a number of layers. To obtain the output of the trajectory stream, we flatten the output of the last layer.

[1]We choose a step size of 1 when doing convolution and pad zeros to the input if the domain of a filter goes outside of the input.

3.2 Visual Stream

We choose the C3D network [26] to handle incoming visual data. In our experiments, we use the C3D structure shown in Fig. 3.1 and use the feature from fc6 where the number of output neurons is set to 512. The architecture of visual stream is not our key contribution in this work. In fact, this subnetwork takes video sequences as input, so another of the many powerful networks used for video classification could be used as our visual stream.

3.3 Stream Fusion

Since each stream is constrained to learning a certain pattern within its own data domain, it is helpful to take advantage of the two-stream architecture, forcing the two separate streams to share information with each other. To merge information, we concatenate the output of each stream and pass the fused feature to a fully connected layer(s) to establish inter-stream/domain connectivity. The resulting feature vector is a representation of individual activity in a short sequence.

Let row vectors $X_t \in \mathbb{R}^{F_t}$ and $X_v \in \mathbb{R}^{F_v}$ be the features extracted from the trajectory stream and visual stream respectively, where F_* denotes the corresponding feature length (or the number of neurons in a fully connected layer). The fused output $Z_f \in \mathbb{R}^{F_f}$ is mathematically expressed as

$$Z_f = \sigma([X_v, X_t]W_f), \tag{3.4}$$

where $W_f \in \mathbb{R}^{(F_v+F_t)\times F_f}$ are the weights in a fully connected layer. More fully connected layers could be built on top of Z_f to accommodate complicated cases.

3.4 Shared Network For Multiple Persons

To design a system for analyzing behaviours of a group of people in a scene, we need a separate network for each person. However, this is prohibitive in the sense that the large number of resulting parameters would lead to large consumption of memory and long training time. Thus we propose to let all individuals share the same network and concatenate the output feature of each person. Then we establish inter-person connectivity using a fully connected layer.

Note that when we do concatenation, we implicitly enforce an order among this group of people. Arbitrarily enforcing such order is problematic. To resolve this issue, we have to renumber the persons in the input list. We propose two approaches to achieve this. First, we could augment the training dataset by random permutation. Every time a sample (containing videos and trajectories of multiple persons) is fed into the network we shuffle the list of people beforehand. As such, our network automatically learns to handle the permutation issue. Second, we could automatically mark a person as the "center" person according to a predefined rule and put this center person always in the first position of the list. Then we could number other people according to their distances to the center. In our experiments, we apply the first approach to the task of puck carrier detection and the second approach to the task of event recognition and team classification.

Now suppose we have the fused feature $Z_f^{(i)} \in \mathbb{R}^{F_f}$ for person i $(1 \leq i \leq N_p)$. Let $\{Z_r^{(i)} | 1 \leq i \leq N_p\}$ be a new set of features after renumbering and $h(\cdot)$ be an operator that returns the new rank of an input. For example, $h(3)$ might return 1, meaning the person originally at index 3 will be placed at index 1 after renumbering. Therefore, we have

$$Z_r^{(h(i))} = Z_f^{(i)}. \tag{3.5}$$

To obtain the eventual feature representation of a group of people, we concatenate all $Z_r^{(i)}$ and apply a fully connected layer afterwards to learn the inter-person relationships, shown below.

$$Z_e = \sigma([Z_r^{(1)}, Z_r^{(1)}, \ldots, Z_r^{(N_p)}]W_e) \tag{3.6}$$

In the above formula, $\sigma(\cdot)$ denotes softmax normalization, $W_e \in \mathbb{R}^{(F_f \cdot N_p) \times F_e}$ are the weights of the fully connected layer, and $Z_e \in \mathbb{R}^{F_e}$ is the eventual feature representation which can be fed into a loss function for end-to-end training.

In summary, we have a combined network that represents the trajectories and visual appearance of all the people in a scene. This will be used for a variety of activity analysis tasks, described next.

Chapter 4

Datasets

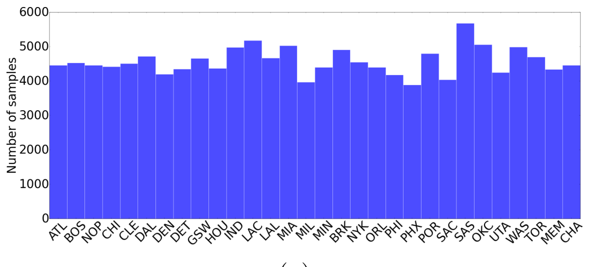
We conduct experiments on two datasets. The first includes trajectory features only: player tracks extracted from an external tracking system recording player positions in NBA basketball games. The second incorporates visual and trajectory features: player positions and appearances obtained from broadcast video footage of NHL hockey games.

4.1 The STATS SportVU NBA Dataset

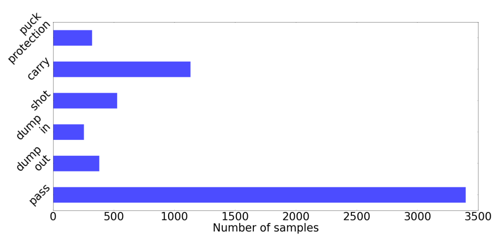
The STATS SportVU data consist of real-time positions of players and the ball in 2D world coordinates captured by a six-camera system at a frame rate of $25Hz$. Each frame has complete annotations of the events happening in this frame, such as dribble, possession, shot, pass and rebound. The dataset we use has 1076 games during the 2013–2014 NBA season with around 10^6 frames in each game. We will use this dataset for team classification – determine the identity of a team from the trajectories of its players during a game.

Data Preprocessing We extract 137176 possessions from the 1076 games for experiments. Each possession starts with an offensive team having possession of the ball and ends with a shot. We fix possession length to 200 frames. If a possession is longer than 200 frames, we crop it starting from the last frame and count the number of frames backward until it reaches 200. If a possession is shorter than 200 frames, we pad zeros to it. Originally there are 25 frames per second, but we sample only half of the frames in a second, so the sampled 200 frames actually represent a 16[1] second long sequence. There are in total 30 NBA teams. Fig. 4.1a shows the number of possessions we extracted from each team in the dataset. We can see that this is a relatively balanced dataset, each team having a similar number of samples for experiments.

[1] $16s = \frac{200 frames \times 2}{25 fps}$



(a)



(b)

Figure 4.1: (a): Number of possessions of each team in the dataset. (b): Number of samples per event in the dataset.

4.2 The SPORTLOGiQ NHL Dataset

The NHL dataset has both video and trajectory data. Unlike the NBA dataset where person trajectories are obtained from a multi-camera system, the player positions in the NHL dataset are estimated using a homography, which maps a pixel in image coordinates to a point in world coordinates. SPORTLOGiQ Inc. utilizes state of the art algorithms to automatically detect and track players in raw broadcast videos. If we have the bottom-mid point of a player bounding box, we can map this point to world coordinates with a homography matrix, hence acquiring the player position. Similarly, the NHL dataset also has detailed event annotation for each frame, each event being categorized into a super class and a fine-grained class. In our experiment, we use 8 games with 6 super classes: pass, dump out, dump in, shot, carry and puck protection. Fig. 4.1b shows the fraction of each event in the 8-game dataset. A reader might notice that this is a highly unbalanced dataset in terms of events. We will describe how we handle such imbalance in Sec. 5.3.

Data Preprocessing In a hockey game, typically there are 4 on-ice officials and 12 players (6 on each team). Thus, there can be at most 16 persons on the rink at the same time. In the following we do not make any distinction between officials and players and we use “player" to refer to all people on the rink. Because the dataset is created from NHL broadcast videos where not all players are visible in each frame, we need to set a threshold N_p so that our model can handle a fixed number of players. If the number of players available in a frame is less than N_p, we pad with zeros the part where players are unavailable. Each training sample consists of data from N_p players. The data of each player includes a T-frame video clip (cropped from raw video using bounding boxes) and the corresponding T-frame trajectory estimated from this video clip. Note that our model supports variable-length input. If in some frames a player is not available, we set the data in these frames to zeros. In our experiments, N_p is set to 5 and video frame size is set to 96×96. We set T to 16 by first locating the center frame where an event happens and then cropping 7 frames before the center frame plus 8 frames after it. If the center frame of a certain event happens to be close to that of another event within 15 frames, we drop this sample.

Chapter 5

Experiments

We conduct our experiments on the NBA and NHL datasets. To demonstrate that 1D convolution is capable of learning temporal dynamics of person trajectories, we do team classification using the NBA dataset. Then we do puck carrier detection and event recognition on the NHL dataset to show that adding trajectory information can help boost the performance on both tasks.

5.1 Team Classification on the NBA Dataset

5.1.1 Experiment Settings

Since the NBA dataset only has trajectory data, we do not use the whole structure described in Fig. 3.1. Instead, we only use the 1D convolutional network. To handle the permutation issue mentioned in Sec. 3.4, we renumber players according to their distances to the ball. The closest is No.1 and the farthest is No.5. Then the x and y coordinates of the ball and 5 players are stacked together, resulting in a 200×12 matrix as input, where 200 is the length of the input sequence and 12 is the number of channels. We use 60% of the 1076 games for training, 20% for validation and 20% for test.

5.1.2 Measurement

We measure the performance of our model according to the following metrics: accuracy and hit-at-k accuracy[1], both of which are calculated over possessions. However, a single trajectory series can hardly display the full underlying pattern a team might possess. To resolve this issue, we propose to use all possessions in a game and classify the game as a whole using majority voting. For example, if most possessions in a game are predicted as Golden State Warriors, then the model predicts this game to be with the Golden State Warriors. Our experiments show that the per-possession accuracy can be largely improved

[1]Hit-at-k accuracy means if any one of the top-k predictions equals the ground truth label, we claim it as being correctly classified.

layers	acc	hit@2	hit@3	game acc
2conv	10.68%	18.09%	24.31%	50.00%
3conv	18.86%	28.89%	36.47%	87.05%
4conv	22.34%	33.03%	40.47%	93.41%
5conv	24.78%	35.61%	42.95%	95.91%
5conv+2fc	25.08%	35.83%	42.85%	94.32%

Table 5.1: Metrics on models with different number of layers. All convolutional layers use a filter size of 3 except the first layer, where the filter size is 5. The number of filters in next layer is double the number in previous layer except the fifth layer (if any), where the number of filters is the same as that in the fourth layer. The number of neurons in fully connected layer is set to 1024.

when aggregated to game level (see results of "acc" and "game acc" in Table 5.1, 5.2 and 5.3). These numbers are significantly higher than chance performance of $\frac{1}{30} = 3.3\%$.

5.1.3 Analysis

We explore the architecture of our model by varying the number of convolutional layers, the filter size and the number of filters in each layer. Tables 5.1, 5.2 and 5.3 show the results respectively. From Tables 5.1 and 5.3, we have that by increasing the number of layers and filters, generally one could obtain a more complex model to achieve better performance. However, as we increase the number of parameters in the model, there could be a potential limit that could prohibit us from acquiring further improvement by increasing the model complexity. For example, by adding two fully connected layers after the 5conv model in Table 5.1, we only have a slight elevation in possession-based accuracy and a drop in game-based accuracy. Also note that in Table 5.2, using small filter sizes generally leads to good results (see the first three models in Table 5.2). If we slightly increase the filter size, we have a large decrease in model performance (see the last model in Table 5.2). This is possibly because by stacking multiple convolutional layers, the top layer is capable of perceiving everything in the input even with small filters while when we have larger filters, the filters are convolved with many zeros padded to the bottom layer, leading to unacceptable inaccuracies. That is why we have a considerable drop in performance with large filter sizes.

Fig. 5.1b shows the confusion matrix created from the 5conv model in Table 5.1. For most teams, our model (when aggregated to game level) can correctly predict the label. The worst case is Phoenix Suns (PHX in Fig. 5.1b), the model has only a probability around 65% to classify it correctly, but this is still much better than chance performance.

To see what kind of patterns the model learns over the time dimension, we visualize a small fraction of the filters in the first convolutional layer. In Fig. 5.1a, we show 64 filters learned from the input sequence of x coordinates of the ball. Some of them appear to be

filter sizes	acc	hit@2	hit@3	game acc
3 3 3 2 2	24.24%	35.36%	43.25%	94.10%
5 3 3 3 3	24.78%	35.61%	42.95%	95.91%
7 5 5 3 3	23.12%	33.48%	41.04%	95.45%
9 7 7 5 5	14.13%	23.15%	30.01%	62.05%

Table 5.2: Metrics on models with different filter sizes. All models in the table use five convolutional layers with no fully connected layer. The filter sizes listed is in a bottom-up order and the number of filters used are 64, 128, 256, 512, 512 (bottom-up order).

base # filters	acc	hit@2	hit@3	game acc
16	20.37%	30.71%	38.21%	81.14%
32	23.73%	34.55%	41.85%	92.95%
64	24.78%	35.61%	42.95%	95.91%
128	21.81%	32.10%	39.24%	94.45%

Table 5.3: Metrics on models with different number of filters. All models in the table use five convolutional layers with no fully connected layer. The base number of filters listed in the table is the number of filters in the first layer. The number of filters in next layer is double the number in previous layer except that the fourth and the fifth layers have the same number of filters.

“Z" or “S" shaped and some appear to be “M" or “W" shaped. Some of them are similar, so there could be redundancy in these filters. These temporal patterns are the building blocks that form discriminative representations to distinguish teams.

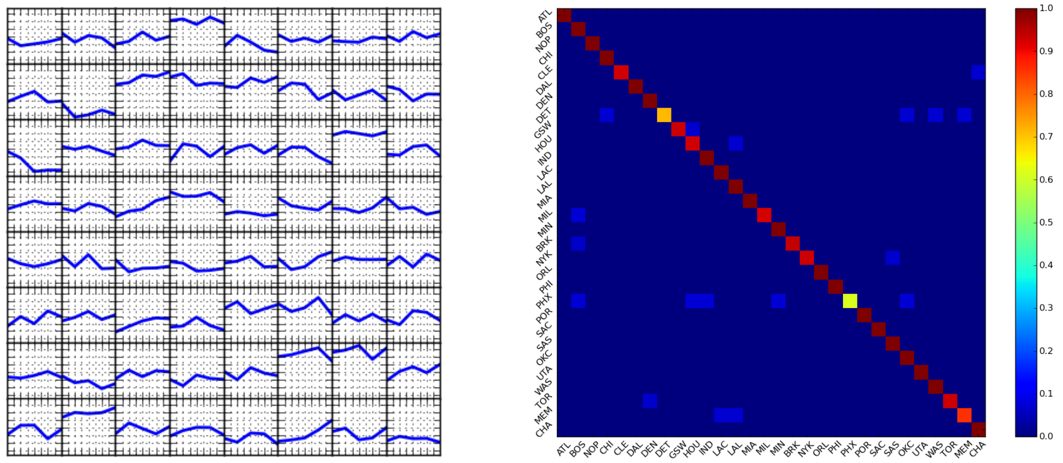
5.2 Puck Carrier Detection on the NHL Dataset

5.2.1 Task Description

Given an input scene, our goal is to detect the player who is currently in possession of the puck. We annotate the dataset by which player has the puck at the moment an event takes place. For example, if a player is passing the puck to a teammate, within a small time window (16-frame window in our case) the player must have the puck, so he is the puck carrier. The events we use are pass, dump in, dump out, shot, carry and puck protection as shown in Fig. 4.1b. We use a one-hot vector to represent the ground truth label (who has the puck) and model the task as a classification problem.

5.2.2 Experiment Settings

We use accuracy to evaluate on the proposed two-stream model as well as two baselines. The two baselines use only either the visual stream or the trajectory stream. For the two-stream model, we use the exact model shown in Fig. 3.1 except that two shared fully connected layers (with 2048 and 512 output neurons respectively) are used to merge the



(a) (b)

Figure 5.1: (a) is the visualization of the filters in the first convolutional layer. (b) is the confusion matrix based on game-wise classification. Both figures are created using the 5conv model in Table 5.1.

	C3D	1D conv	C3D+1D conv
pass	74.23%	45.93%	74.23%
dump out	73.63%	38.46%	74.73%
dump in	69.57%	40.58%	72.46%
shot	82.09%	52.24%	82.84%
carry	72.57%	56.25%	75.00%
puck protection	73.33%	29.33%	70.67%
all events	74.31%	46.88%	74.88%

Table 5.4: Puck carrier detection accuracy for each event.

trajectory stream and visual stream. For trajectory stream, the filter sizes are 3, 3, 3, 2 and the numbers of filters in each layer are 64, 128, 256, 512 (all in bottom-up order). All max pooling uses a step size of 2. To handle the permutation problem as described in Sec. 3.4, we randomly shuffle the list of player candidates for each sample during training. In our experiments, we use 4 games for training, 2 games for validation and 2 games for test.

5.2.3 Experiment Results

Table 5.4 shows the results. We found that by combining visual data with trajectory data, we achieve better accuracy. Compared to the 1D conv model, considering visual features as extra cues in the two-stream model leads to large improvement in performance. Compared to C3D, the two-stream model has a small performance gain.

	C3D	1D conv	C3D+1D conv
pass	77.30%	77.73%	79.15%
dump out	10.17%	22.30%	23.27%
dump in	10.25%	39.39%	37.29%
shot	34.17%	42.42%	50.86%
carry	86.37%	77.21%	86.21%
puck protection	11.83%	9.87%	8.43%
mAP	38.35%	44.89%	47.54%

Table 5.5: Average precision for each event.

5.3 Event Recognition On The NHL Dataset

5.3.1 Task Description

The events used are pass, dump out, dump in, shot, carry and puck protection. The goal is to predict the event label given the short video clips and trajectories of 5 players on the rink. The number of samples of each event in the dataset are shown in Fig. 4.1b. It is obvious that this dataset is highly unbalanced with the pass event taking up half of the dataset. To resolve this problem, we minimize a weighted cross-entropy loss function during training. The weighting for each class is in inverse proportion to its frequency in the dataset.

5.3.2 Experiment Settings

We use average precision as the metric and compare the performance of the proposed two-stream model with that of the C3D model and the 1D convolutional network. For the two-stream model, we use the exact model shown in Fig. 3.1, where one shared fully connected layer with 2048 neurons is used to merge the two streams. The weights in the loss function for pass, dump out, dump in, shot, carry and puck protection are 0.07, 0.6, 1, 0.4, 0.2 and 0.7 respectively. To resolve the permutation issue mentioned in Sec. 3.4, we enforce an order among the N_p players by renumbering the players according to the following rule. We define the player directly related to an event as the "center" player. Then we calculate the distances of other players to the center and rank them by increasing distances. The closest has the highest rank and the farthest has the lowest rank. In our experiments, we use 4 games for training, 2 games for validation and 2 games for test.

5.3.3 Experiment Results

The results are shown in Table 5.5. The mean average precision with the two-stream model is nearly 10 percentage points higher than that of C3D. Further, in Fig. 5.3, it is clear to see that the precision-recall curve of the two-stream model is better than that of C3D for most events. The two-stream model outperforms C3D by a large margin, demonstrating the effectiveness of adding trajectory data.

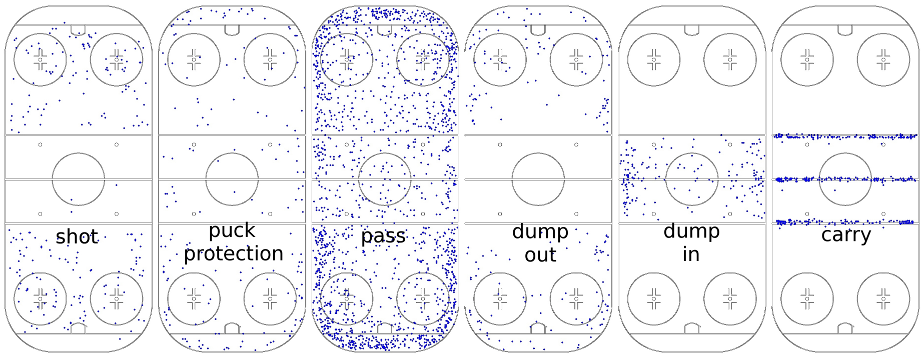


Figure 5.2: Visualization of locations where events happen. Samples are drawn from the test set.

Even considering 1D convolution on trajectory data alone can beat the C3D model. We believe this is due to the strong relationship between events and the spatial locations. As is shown in Fig. 5.2, different events tend to have different spatial distributions over the rink. For example, carry happens near the three lines in the middle; dump in happens within the neutral zone; dump out mostly happens around the corner and boundary. This strong spatial correlation explains the importance of trajectory data for analyzing player behaviours.

We visualize the top 5 candidates retrieved as dump in and dump out in Fig. 5.4. For other events, the top 5 candidates are either all true positive (for pass, carry and shot) or false positive (for puck protection). As we can see from Fig. 5.4, the retrieved events look similar. Even from a human perspective, it is hard to predict the label of a given sample, showing the difficulty of this task.

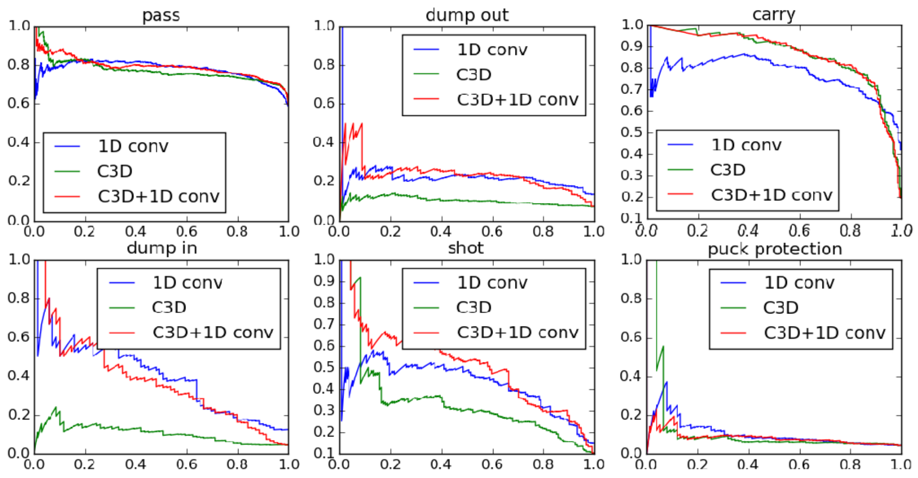
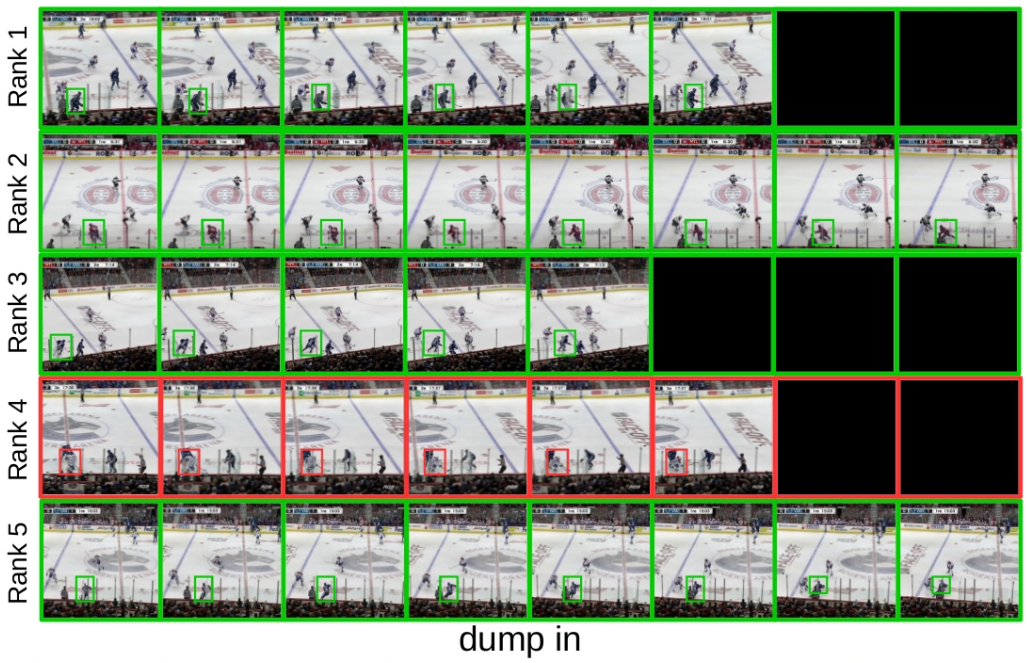


Figure 5.3: Precision recall curves for each event in the hockey dataset.



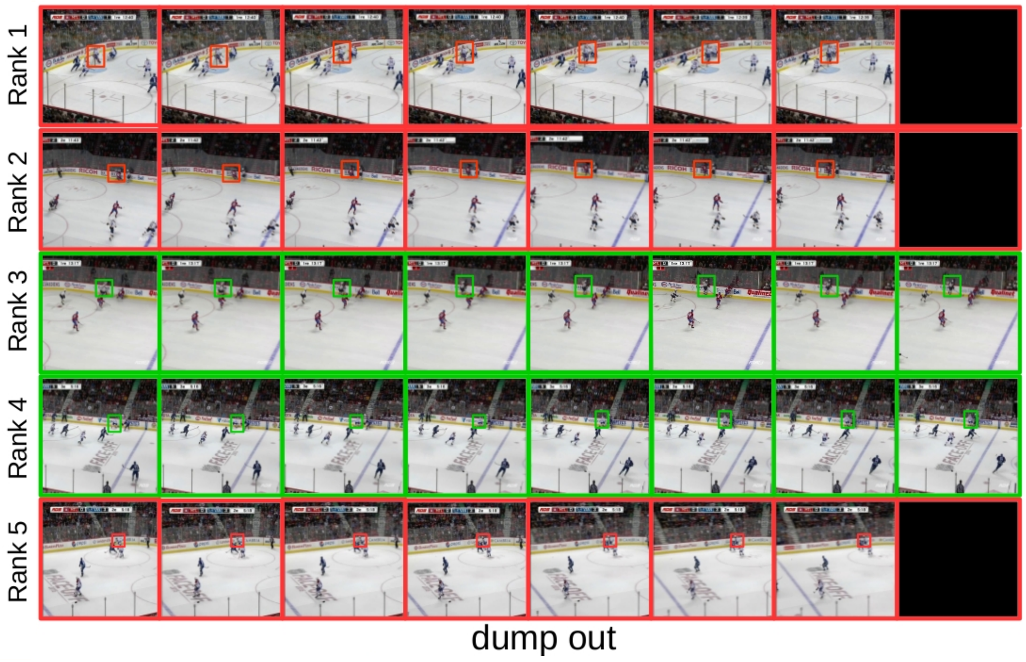


Figure 5.4: Top 5 candidates retrieved as dump in and dump out respectively. Green ones are the true positives while red ones are the false positives. The person with a bounding box is the "center" player who is performing the action. We only show 8 frames of the 16-frame video clip by sub-sampling. If a frame is black, it means the "center" player is missing because of failure to detect and track the player. Zoom in to see the details.

Chapter 6

Conclusion and Future Directions

We presented a method for deep learning on human trajectories for activity analysis. Input coordinates are passed through a network that contains layers of 1D convolutions, applied to a spatial position representation of a person in a scene. This type of network is capable of learning complex representations of human movement. We demonstrate its efficacy on several tasks on real-world basketball and hockey trajectories and experiments show that adding trajectory data can actually help improve performance compared to baselines that do not consider trajectory. This method can be applied to person trajectories acquired from automatic tracking systems and combined with visual features to conduct analyses of sports videos.

Our current model does not distinguish between the two teams in a game. Even if the model detects correctly who is carrying the puck or what action a player is performing, it has no idea which team this player belongs to. We conjecture that the player behaviour is strongly associated with the role he is playing. For example, is he in an offensive team or defensive team? Thus, one future direction could be designing a model that can discover the role/team of a player and relate that information to the player behaviour.

Another limitation of our model is that we use a fixed number of players as input. If some player is missing, we zero out the input of this player. Because the number of visible players in broadcast videos varies a lot, in order not to have many player inputs zeroed out, this fixed number we use is small (i.e., 5 in our case). Therefore, one possibility is to extend the current model to a more flexible one, which can accommodate situations where the number of visible players range from 1 to 16 (for hockey games).

Bibliography

[1] Mohamed Rabie Amer, Peng Lei, and Sinisa Todorovic. Hirf: Hierarchical random field for collective activity recognition in videos. In *European Conference on Computer Vision (ECCV)*, pages 572–585. Springer, 2014.

[2] Wongun Choi and Silvio Savarese. A unified framework for multi-target tracking and collective activity recognition. In *European Conference on Computer Vision (ECCV)*, pages 215–230. Springer, 2012.

[3] Wongun Choi, Khuram Shahid, and Silvio Savarese. What are they doing?: Collective activity classification using spatio-temporal relationship among people. In *Computer Vision Workshops (ICCV Workshops)*, pages 1282–1289. IEEE, 2009.

[4] Z. Deng, M. Zhai, L. Chen, Y. Liu, S. Muralidharan, M. Roshtkhari, , and G. Mori. Deep structured models for group activity recognition. In *British Machine Vision Conference (BMVC)*, 2015.

[5] Zhiwei Deng, Arash Vahdat, Hexiang Hu, and Greg Mori. Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[7] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding (CVIU)*, 73(1):82–98, 1999.

[8] Joachim Gudmundsson and Michael Horton. Spatio-temporal analysis of team sports–a survey. *arXiv preprint arXiv:1602.06994*, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Moustafa Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. A hierarchical deep temporal model for group activity recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] Stephen S. Intille and Aaron Bobick. Recognizing planned, multiperson action. *Computer Vision and Image Understanding (CVIU)*, 81:414–445, 2001.

[12] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[13] Sameh Khamis, Vlad I Morariu, and Larry S Davis. Combining per-frame and per-track cues for multi-person action recognition. In *European Conference on Computer Vision (ECCV)*, pages 116–129. Springer, 2012.

[14] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *The International Conference on Computer Vision (ICCV)*, 2011.

[15] Tian Lan, Leonid Sigal, and Greg Mori. Social roles in hierarchical models for human activity recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] Tian Lan, Yang Wang, Weilong Yang, Stephen Robinovitch, and Greg Mori. Discriminative latent models for recognizing contextual group activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(8):1549–1562, 2012.

[17] Patrick Lucey, Alina Bialkowski, G. Peter K. Carr, Stuart Morgan, Iain Matthews, and Yaser Sheikh. Representing and discovering adversarial team behaviors using player roles. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[18] G. Médioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 23(8):873–889, 2001.

[19] Vignesh Ramanathan, Jonathan Huang, Sami Abu-El-Haija, Alexander Gorban, Kevin Murphy, and Li Fei-Fei. Detecting events and key actors in multi-person videos. In *Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, June 2016.

[20] Vignesh Ramanathan, Bangpeng Yao, and Li Fei-Fei. Social role discovery in human events. In *Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[21] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[22] Stephane Ross, Daniel Munoz, Martial Hebert, and J Andrew Bagnell. Learning message-passing inference machines for structured prediction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2737–2744. IEEE, 2011.

[23] Alexander G Schwing and Raquel Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[24] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[25] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[26] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015.

[27] Kuan-Chieh Wang and Richard Zemel. Classifying nba offensive plays using neural networks. In *Sloan Sports Analytics Conference*, 2017.

[28] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding (CVIU)*, 115(2):224–241, 2011.

[29] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Computer Vision and Pattern Recogntion (CVPR)*, pages 1529–1537, 2015.