Action Recognition in Tennis Using Deep Neural Networks

Vincent Chow
Department of Mechanical Engineering
Stanford University
chowv@stanford.edu

Ohi Dibua
Department of Mechanical Engineering
Stanford University
odibua@stanford.edu

Abstract

The long-term motivation of this work is to create a computer vision system that tracks tennis players, and identifies their actions on real-time video data. A first step is to properly identify tennis strokes. In order to achieve this, we employ deep learning architectures. In particular, we compare different methods of feature generation, and different RNN architectures for processing time-series data. We generate features using two different techniques. One method employs a pre-trained CNN and the second method captures optical flow. We feed these features into RNN networks with LSTM units in order to compare how well each approach classifies videos of players performing tennis strokes. We achieve the best results by feeding features generated from a pre-trained CNN into a many-to-many LSTM network, and averaging the softmax outputs to classify videos.

1 Introduction

The goal of this project is to apply deep learning to action recognition in tennis, with the ultimate goal of exploring effective methods for automatic video annotation in sports. We explore two methods of generating features from video data: pre-trained CNNs and optical flow. We utilize these features in conjunction with RNN networks in order to perform action recognition in tennis. We use two types of RNN architectures. The first is a many-to-many LSTM network, in which predictions are made by averaging the softmax probabilities produced by the LSTM at each timestep, and the second is a many-to-one LSTM. The ability to automatically annotate tennis matches has great potential for providing invaluable tools to tennis players for collecting data about their hitting form, and to allow sports broadcasters to give fans insight into trends from major tennis matches. For our particular project, the inputs are a series of images from a video of a tennis player hitting a stroke, and the output is the class of tennis stroke that the player performs in the video.

2 Dataset and Features

The THree Dimensionsal Tennis (THETIS) dataset used in this project comprises of approximately 8734 video clips in AVI format containing RGB, depth, and 2D/3D skeleton data.[1]. For the purposes of this project, we utilize all 1980 videos containing RGB data. Each video contains approximately 80 frames, sized 640 x 480 pixels. In each video clip, a player performs one of up to 12 classes of tennis strokes. Example strokes include: forehand, backhand, service, and smash.

In this project, we down-sample each video to 16 frames. Before feature extraction, we preprocess each video frame by normalizing each RGB pixel value by the mean and standard deviation across the

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

entire image. In order to deal with limited data, we consolidate 12 possible classes to 6, by grouping similar tennis strokes together (e.g. two-handed backhand and one-handed backhand). Finally, we split our data into training/validation/test sets according to a 80/10/10 distribution, and ensure that all classes are evenly sampled.

3 Methods

We describe in detail the two methods we use for tennis stroke classification. The first combines a Convolutional Neural Network (CNN) and LSTMs based on the work by Donahue et al [2]. In this architecture, known as Long-term Recurrent Convolutional Neural Network (LRCNN), a pre-trained CNN network, such as Inception V3, extracts features from each video frame, and passes them to an LSTM network [3]. The LSTM consists of a softmax layer that outputs probabilities corresponding to each class. The input video is classified by averaging the result of the softmax outputs across all frames in a video. The Inception V3 CNN architecture is shown in Figure 1, and the overall LRCNN approach is shown in Figure 3.

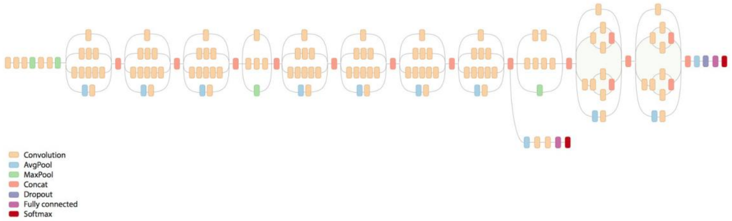


Figure 1: Inception V3 CNN Architecture

Our second approach uses a similar LSTM architecture to classify videos. However, instead of using a pre-trained CNN to extract features from video frames, we use standard computer vision techniques. This approach offers a compromise between deep learning and traditional techniques, as seen in the work by Baccouche et al [4]. In particular, we calculate the optical flow of video frames, split the images into cells, and use information about the magnitude and direction of optical flow in each cell to construct dynamic word representations of each frame [5]. Figure 2 shows an example of an image extracted from a video sequence, and the resulting optical flow visualized. We note that the direction and magnitude are quantized as follows:

$$\omega = [A, B, C, D, E, F, G], \beta = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] \tag{1}$$

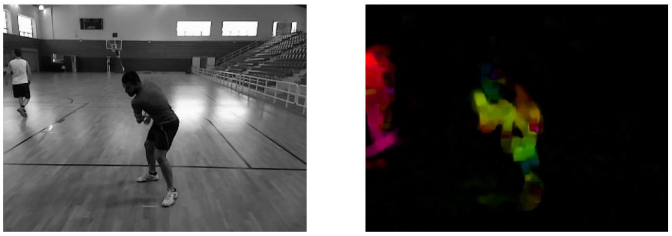
where ω quantizes the direction between 0 and 2π, and β quantizes the sum magnitude of the cells based on the maximum observed in a frame. Given q total cells, the dynamic word representation is

$$\omega_0\beta_0....\omega_q\beta_q \tag{2}$$

In the case of the images in Figure 2, we split the image into 4x4 cells and obtain a dynamic word representation of:

$$F2D0E0E0G9B6B2E0G5B3C5D0E0C0C1C0 \tag{3}$$

The features found through optical flow are passed into a many-to-one LSTM. The output of the final LSTM is fed to a softmax layer, which is used to classify the input video. Figure 3 illustrates this second method on an image.



(a) Gray scale backhand

(b) Optical flow of grayscale backhand

Figure 2: (a) shows an example of a gray-scale image, and (b) the corresponding optical flow

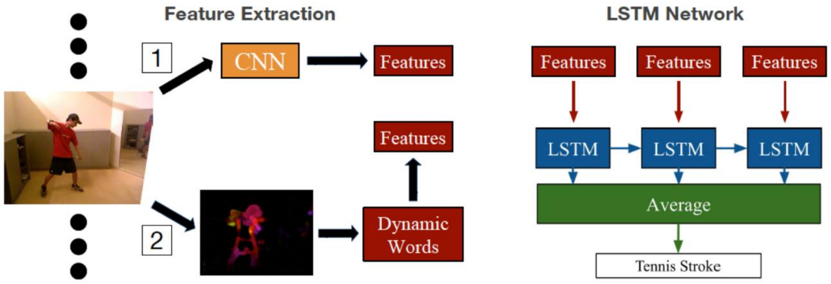


Figure 3: Project Architectures

For both methods, We define our loss function to be the categorical cross entropy:

$$J = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} y_j^{(i)} log(p_j^{(i)}) \tag{4}$$

where m is the number of training examples, n is the number of classes, y is the ground truth one-hot label, and p is the softmax probabilities output by the LSTM network.

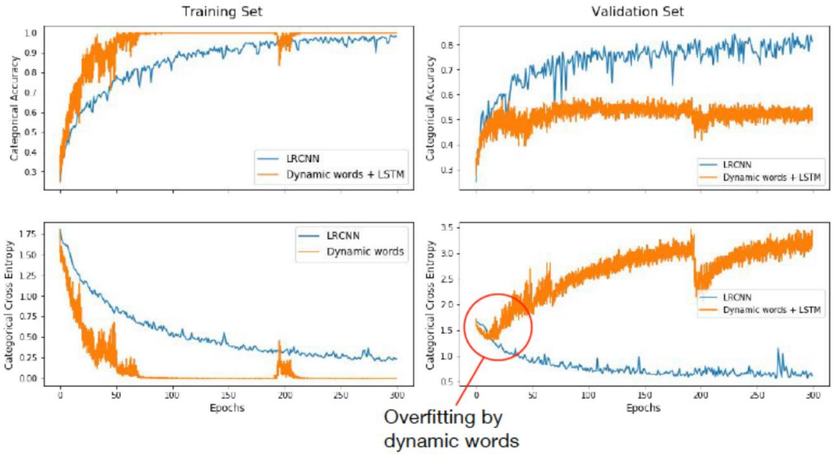
4 Experiments/Results/Discussion

In this section, we discuss our experiments and results. To improve model performance, we tuned our hyperparameters on the LRCNN model by performing a uniform grid search on a logarithmic scale for each hyperparameter. These include: learning rate, batch size, number of LSTM hidden units, and the dropout rate. Since the RNN networks for each feature generation method are similar, we use the same optimal hyperparameters used for the LRCNN architecture with the optical flow-based architecture as well. We train both architectures using the Adam optimization algorithm, and perform dropout regularization. Optimal hyperparameters are listed in Table 1.

Table 1: Hyperparameters

Learning Rate	Batch Size	LSTM Hidden Units	Dropout Rate
1e-3	128	128	0.3

Our first round of experiments were trained on data split into all 12 of the original classes. These experiments yielded poor results. For the LRCNN architecture, the categorical accuracy on the test set reached approximately 62%, while for the optical flow-based architecture, the categorical accuracy on the test set reached only 25%. Intuiting this as a problem due to lack of data, we then proceeded to consolidate the 12 classes of tennis strokes to 6, achieving much better results due to the increased number of samples within classes. Figure 4 shows the accuracy and losses for the training and validation data. We see that both models fit the training data well. The optical flow-based model converges in loss and accuracy more quickly than with LRCNN. However, it is clear that LRCNN generalizes much better. The loss on validation data for the optical flow-based model quickly shows signs of overfitting, and the accuracy converges quickly to a maximum of approximately 50%. However, the LRCNN model generalizes very well, reaching a max of approximately 78% on validation data. As shown in Table 2, LRCNN achieves a final test set accuracy of 82.3%, and the optical flow-based model achieves a final test set accuracy of 53.2%.



Overfitting by dynamic words

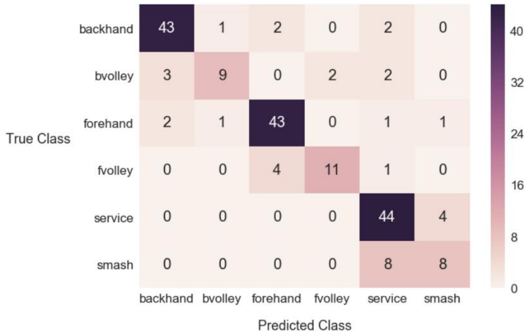
Figure 4: Learning curves for validation and training sets

Based on these results, we suspect that, given the similar LSTM architecture employed by both feature generation methods, the optical flow method of extracting features loses useful information that is otherwise captured by Inception V3 CNN network. We suspect that incorporating additional features, such as keypoints to capture pose information, could potentially improve the performance of the optical flow-based model.

Table 2: Categorical Accuracies

Architecture	Training	Validation	Test
LRCNN	98.7%	77.9%	82.3%
Optical flow + LSTM	100%	53.2%	53.2%

Figure 5 shows the confusion matrix generated by predicting test data with the LRCNN model. Note that "bvolley" stands for "backhand volley," and "fvolley" stands for "forehand volley." The trends are intuitive. The model tends to perform better on classes that contain more data due to class consolidation, including backhand, forehand, and service strokes. Forehand volleys are often confused with forehand ground strokes, since they have similar motions. A further analysis of misclassified forehand volley videos also shows several players performing the stroke incorrectly, making it more likely for the model to mistake the stroke as a forehand. This result also holds for backhand volleys and backhand ground strokes. Finally, the algorithm has difficulty with the smash stroke, because the action requires a similar motion as a service stroke, and furthermore, as with volleys, we did not have as many samples of this class in the training set.



True Class \ Predicted Class	backhand	bvolley	forehand	fvolley	service	smash
backhand	43	1	2	0	2	0
bvolley	3	9	0	2	2	0
forehand	2	1	43	0	1	1
fvolley	0	0	4	11	1	0
service	0	0	0	0	44	4
smash	0	0	0	0	8	8

Figure 5: Confusion matrix for CNN + LSTM (LRCNN) architecture

5 Conclusion/Future Work

In this project, we demonstrate that using deep neural networks, we are able to classify videos of players performing tennis strokes with up to 82.3% accuracy, using a LRCNN model trained on RGB video data from the THETIS dataset. A major theme throughout our project has been encountering the importance of the quantity and quality of data used for training. We discovered that the dataset consists of primarily amateur players, several of which performed strokes incorrectly, affecting the performance of the models employed. Regarding quantity, we discovered that 1980 videos are not enough to train a model to sufficiently distinguish among 12 different classes of tennis strokes. However, by consolidating the number of classes to 6, we are able to achieve much higher accuracies, suggesting that the quantity of the dataset limits the performance of the models. Secondly, we conclude that the deep learning approach outperforms the optical flow approach in extracting useful features for this video classification task.

In the future, we hope to accomplish several tasks. For one, we would like to collect a large quantity of data from players with higher skill levels, and secondly from multiple perspectives, so we can generalize classification to more than one camera angle on the tennis court. Secondly, since the ultimate goal is real-time video classification, computation time is an important consideration. In order to speed up computation, we believe it is still worthwhile to continue exploring standard computer vision techniques. Incorporating pose keypoints with optical flow is a logical next step. Finally, in order to improve the performance of the LRCNN architecture, we wish to extend backpropagation through to the last few layers of the Inception V3 CNN network to optimize feature extraction.

References

[1] Sofia Gourgari, Georgios Goudelis, Konstantinos Karpouzis, and Stefanos Kollias. Thetis: Three dimensional tennis shots a human action dataset. *International workshop on Behavior Analysis in Games and modern Sensing devices*, 2013.

[2] 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. *arXiv preprint arXiv:1411.4389*, 2016.

[3] TensorFlow. How to retrain inception's final layer for new categories | tensorflow, Jan 2018.

[4] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Action classification in soccer videos with long short-term memory recurrent neural networks. *International Conference on Artificial Neural Networks*, 2015.

[5] Jonathan Vainstein, Jose F. Manera, Pablo Negri, Claudio Delrieux, and Ana Maguitman. Modeling video activity with dynamic phrases and its application to action recognition in tennis videos. *CIARP 2014: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 2014.

6 Contributions

Vincent (1) implemented the Inception V3 CNN method to extract features, (2) trained the many-to-many LSTM architecture used in LRCNN, and (3) performed hyperparameter tuning experiments.

Ohi (1) implemented the optical flow and dynamic words method to extract features, and (2) trained the many-to-one LSTM architecture.