Tennis Shot Recognition through Spatiotemporal Deep Neural Networks

Siddharth Buddhiraju
Department of Electrical Engineering
sbuddhi@stanford.edu

Ganapathy Sankararaman
Department of Mechanical Engineering
ganasank@stanford.edu

Code: https://github.com/ganasank/CS230

1 Problem description

Recognition of human action in videos has generated substantial attention from the deep learning community in recent years [1-4]. In this project, we aim apply the video classification problem to detect tennis shots. This work is also aimed at making progress towards a larger project that we are working on, where we envision a need to identify and correct player postures while playing various tennis shots. Consequently, there are two ML problems: to identify a particular shot, and to detect if the posture is correct. As a first step towards this goal, we aim to use deep neural networks which takes a video as input, and classifies it into one of 6 basic tennis shots.

2 Dataset

The THETIS dataset [5] consists of 12 Tennis shots performed by 31 amateurs and 24 experienced players. Each player performed the same shot 3 - 5 times. There are 1980 RBG videos in total in .avi format, with about 80 frames per video. A few frames are shown for reference in Fig. 1. We split the dataset into training, validation and test sets so that each set has an equal proportion of different shots. Training has 1584 videos, Validation has 204 videos and Test has 192 videos (roughly 0.8,0.1,0.1 split).

3 Baseline and Bayes Error

3.1 Baseline: LRCNN

As a baseline, we use the 'LRCNN' model by Chow and Dibua [6] from their CS 230 project in Winter 2018. In this project, Chow and Dibua use the THETIS RGB dataset and downsample the videos to 16 frames. Further, they condensed the 12 distinct shot classes into 6 by merging various types of backhands, forehands and serves, because using all 12 classes yielded poorer results. To make their shot prediction, they first extract features from video frames using the Inception V3 network pre-trained on ImageNet. Then, they feed the features into a many-to-many LSTM network, which is trained to output one of the 6 tennis shots as its prediction. Their results, which we were able to reproduce after modifications to the input layer of their Inception V3 network, are shown in the first row of Table 2.

3.2 Bayes error: Human performance

To estimate human-level performance for tennis shot identification, we asked five people with a good understanding of tennis to classify 24 videos into the 6 different shot categories. The set of 24 videos had a uniform distribution of shots from the testing set which was used for model evaluation. Among these five people, four of them had an accuracy of 79.17% and the fifth an accuracy of 83.33%.

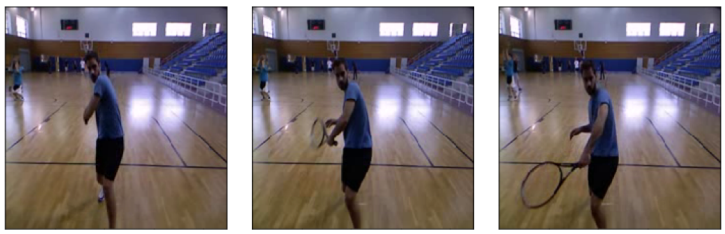

Figure 1: Three frames from a backhand volley (`bvolley`) being performed by a player

Further, we used a voting based approach with the 5 human votes to determine a class for a given video. Using this approach, the accuracy improved to **87.5**%. Hence, based on our human-level performance dataset, we conclude that the Bayes error is 12.5% with a team of experts (people with solid understanding of the game).

4 Methods

Since this project involves a classification task over six classes, we adopt categorical cross-entropy as the loss function for all models in this work.

$$J = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{5} y_j^{(i)} \log \hat{y}_j^{(i)} \tag{1}$$

4.1 Improved LRCNN model

As our first model, we attempted to improve on the baseline LRCNN model. To that end, we generated sequences for the THETIS RGB videos for 16 frames from Inception V3 network and used a recurrent neural network based temporal analysis to predict the output classes. Please see Sec. 5 for details and results.

4.2 EvaNet + Transfer Learning

In parallel, we explored the models in Ref. [7], where an evolutionary algorithm was employed to find optimal video classification neural architectures. In contrast to the traditional, manual design of deep neural networks, the authors in this work introduce a meta-architecture model for which the high level connectivity between modules is fixed, but the individual modules can evolve. In addition to this evolutionary architecture, the authors introduce an Inflated Temporal Gaussian Mixture(iTGM) layer as part of the search space. In this layer, the spatial 2D filter is expanded along a third, temporal dimension by following the weights according to a learned 1D temporal Gaussian mixture pattern. This evolutionary search algorithm was applied to Inception-like (shown in Fig. 2(b)) and ResNet-like meta-architectures and trained on video datasets such as HMDB, Kinetics-400 and Moments in Time. The authors report models that have higher than state-of-the-art performance on all these datasets and run significantly faster than prior models.

In their code on GitHub (https://github.com/piergiaj/evanet-iccv19), the weights for two such optimized models (Inception-based and ResNet-based) were made available. These two models (hereafter labeled Model 0 and Model 1) differ in their architecture and consequently have differing number of trainable parameters. Since their models are trained on an action recognition dataset (HMDB), we chose to apply transfer learning from their model to ours. To do so, we first passed the videos in the THETIS RGB dataset through their models to obtain the outputs sequences before the final two layers (Conv3D and Softmax, respectively). In Model 0, this output sequence for each video was of shape $(1, F, 7, 7, 2046)$, while for Model 1, it was $(1, F, 7, 7, 2048)$, where F is the number of frames varying between 11 and 21 depending on the video length. Subsequently, we average-pooled the sequences along `axis=1` in consistency with both the EvaNet models.

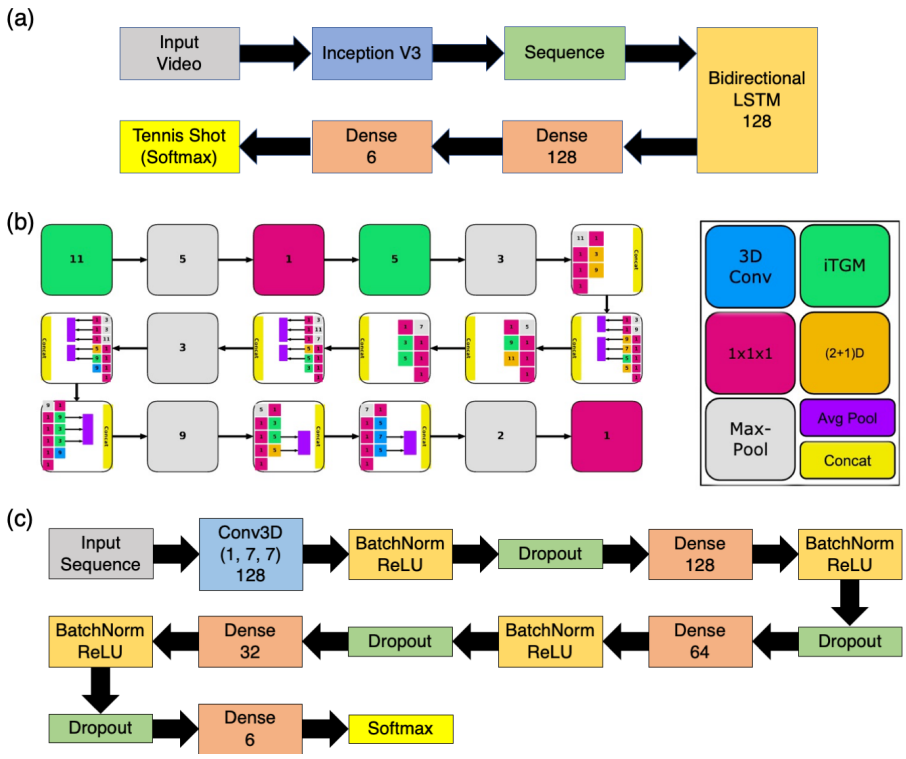

Figure 2: (a) Improved LRCNN architecture (b) An Inception-like architecture found by EvaNet [6] and (b) Deep neural network for training on output sequences from Model 0 in this work.

5 Results

5.1 Improved LRCNN model

Based on the observation of high variance in the baseline LRCNN model, we first trained the model with increased dropout of 50% (baseline had 30%). However, this did not yield any noticeable improvement. Subsequently, we changed the LSTM network in the baseline model to a Bidirectional LSTM network (128 hidden units) with a 50% dropout rate. Further, we added a fully connected layer with 128 units before the softmax layer. This network is shown in Fig. 2(a). The results are tabulated in Table 2 in the first two rows, showing that train and test accuracies and the F1 score are better with Bidirectional LSTM.

5.2 Evanet+Transfer Learning

Transfer learning: Shallow network. To perform transfer learning on the outputs from the EvaNet models, our first attempt involved training over a shallow network consisting of a `Conv3D` layer with 32 filters of shape $(1, 7, 7)$. We applied `BatchNormalization` on this `Conv3D` and passed it to a `Dense` Layer with 6 outputs before applying `Softmax` to get the output. The resulting network had $3,208,358$ trainable parameters for Model 0 and $3,211,494$ parameters for Model 1. For both models, the network was trained for 1000 epochs with a permutation of two `Dropout` rates (0.3,0.5) and four `BatchNormalization` momenta (0.01,0.1,0.5,0.9). The best results for both models are shown in the first row of Table 1. To understand the inferior performance of Model 0, we found that the classes in Model 1 were significantly better clustered, which may explain the superior performance of Model 1. Please see Section 6.1 for a detailed explanation.

Transfer learning: Deep network. To overcome the inferior performance of Model 0, we built a deeper network shown in Fig. 2(c) and evaluated the performance of both the models on the deep network with the same permutations of dropout and momentum. With this architecture, Model 0 had

EvaNet Transfer	Model 0	Model 1
Shallow network	71.87%	**80.21%**
Deep network	**77.60%**	78.12%

Table 1: Comparison of test accuracies of shallow and deep networks built to train on the sequences from EvaNet Model 0 and EvaNet Model 1, respectively.

Model	Train Accuracy	Test Accuracy	F1 Score
LRCNN-Baseline	98.7%	82.3%	0.82
LRCNN-Improved	100.0%	84.4%	0.84
EvaNet+Transfer	99.8%	84.4%	0.84

Table 2: Comparison of train, test accuracy and F1-score for the baseline LRCNN model from [6], the improved LRCNN model in this work, and the best results obtained from transfer learning on the EvaNet model [7].

$12,860,390$ trainable parameters and Model 1 had $12,872,934$ trainable parameters. The results for both models with this deep network are shown in the second row of Table 1.

Transfer learning: Ensembling. The performance of Model 1 deteriorated with a deep network while that of Model 0 improved. Therefore, we used a shallow network for Model 1 and and a deep network for Model 2 and ensembled their results. To that end, we obtained the outputs before the softmax from the shallow network of Model 1 and the deep network of Model 0, averaged them, and applied softmax to the result. After ensembling, the accuracy improved to **84.38%** and F1 Score to **0.84**. The best results from this ensembled EvaNet model are shown in the third row of Table 2.

6 Error analysis

6.1 EvaNet transfer learning on shallow networks

To understand why sequences obtained from Model 0 performed poorly on the shallow network of Sec. 4, we conjectured that in a high-dimensional space, these sequences were not sufficiently clustered together based on their true labels when compared to sequences obtained from Model 1. To verify this, we flattened the sequences from Model 0 into $7 \times 7 \times 2046 = 100,254$ dimensional vectors $\mathbf{u}_m^{(0)}$ and those from Model 1 into $7 \times 7 \times 2048 = 100,352$ dimensional vectors $\mathbf{u}_m^{(1)}$ for $m = 1, 2, ..., 1980$. Then, clustering these sequences based on their true labels, we computed the centroids:

$$\mathbf{c}_j^{(k)} = \frac{\sum_{\mathbf{y}_m=j} \mathbf{u}_m^{(k)}}{\sum_{\mathbf{y}_m=j} 1}, \quad j \in \{0, ..., 5\}; k \in \{0, 1\} \tag{2}$$

Here, $\mathbf{y}_m$ is the true label for video m and j is one of the 6 output classes. Then, we computed the normalized centroid distance between any two classes i and j as

$$d_{ij}^{(k)} = \frac{||\mathbf{c}_i^{(k)} - \mathbf{c}_j^{(k)}||_2}{\sqrt{||\mathbf{c}_i^{(k)}||_2 ||\mathbf{c}_j^{(k)}||_2}}, \quad j \in \{0, ..., 5\}; k \in \{0, 1\} \tag{3}$$

The normalized distance between a few classes for sequences from Model 0 and Model 1 is shown below:

Class$_i$-Class$_j$	Model 0	Model 1
backhand-forehand	2.83	9.59
forehand-bvolley	4.33	23.25
smash-serve	10.27	19.73

This clearly indicates that the sequences from Model 1 are clustered significantly better than those from Model 0, allowing better performance on classification tasks on even shallow networks. On

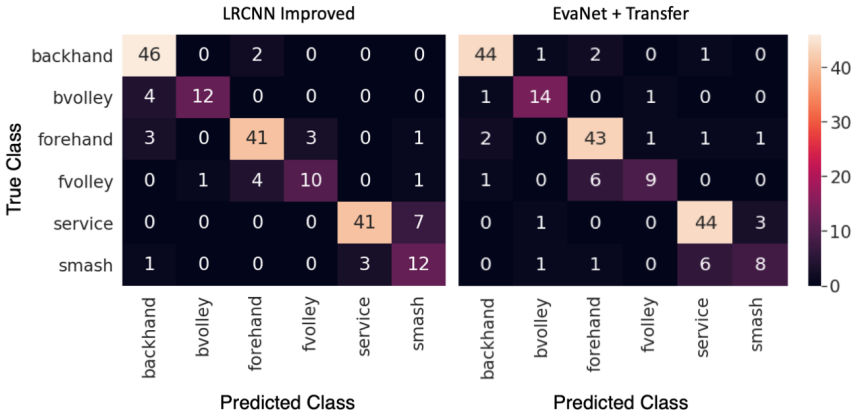

Figure 3: Confusion matrices for improved LRCNN model (left) and the best results from transfer learning on the EvaNet models (right).

the other hand, when deep networks are used, even small separations between clusters can be approximated well, resulting in improved performance on Model 0.

6.2 Confusion matrices

The confusion matrices for the improved LRCNN model and the EvaNet model with transfer learning are shown in Fig. 3. In both the LRCNN model and the ensembled EvaNet model, the classes `bvolley-backhand`, `fvolley-forehand`, and `service-smash` were misclassified often. Among the videos that were misclassified by humans, the errors occurred among the same sets of three classes. Finally, with the team of experts approach, one `backhand` was misclassified as `bvolley`, one `forehand` as `fvolley`, and one `fvolley` as `forehand`.

We conclude that the misclassifications are fairly similar in all methods used. This is likely due to the nature of similarity in these shots, since most of the players in this dataset are amateurs and the shots are filmed indoors without a tennis ball. With better video and shot quality and with more training data, we expect that the model would perform significantly better in predicting these classes accurately.

7 Conclusion and Future Work

With an improved version of LRCNN and doing transfer learning on Google's action recognition algorithm, we are able to identify 6 classes of shots with 84.4% accuracy. While analyzing the cause for the low accuracy, we realized that it is mainly because of the similarity of shots due to their quality. The other point to be noted is the huge variance in the model performance. This is due to lack of sufficient data to train the model. As a part of a bigger project of posture analysis, the current model's performance should be improved significantly if we want to make any analysis on posture. Generating mode quality data and training the model on it will help in this aspect.

S.B. and G.S. contributed equally to this project.

References

[1] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1725-1732).*

[2] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems (pp. 568-576).*

[3] Feichtenhofer, C., Pinz, A., & Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1933-1941).*

[4] Feichtenhofer, C., Pinz, A., & Wildes, R. P. (2017). Spatiotemporal multiplier networks for video action recognition. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 4768-4777).*

[5] Gourgari, S., Goudelis, G., Karpouzis, K., & Kollias, S. (2013). Thetis: Three dimensional tennis shots a human action dataset. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 676-681).*

[6] Chow, V., & Dibua, O. (2018). Action recognition in tennis using deep neural networks. Course project for CS 230. *http://cs230.stanford.edu/files_winter_2018/projects/6945761.pdf*.

[7] Piergiovanni A. J., Angelova A., Toshev A., & Ryoo M. S. (2019). Evolving space-time neural architectures for videos. *Proceedings of the IEEE International Conference on Computer Vision (pp. 1793-1802).*