Classifying Shots Taken in NBA Games with Deep Learning

Christien Williams (willch), Subby Olubeko (subbyo), Yaateh Richardson (yaatehr)

Abstract—In this project, we analyze methods for classifying videos of shots taken in NBA games. We attempt to do this using frame-by-frame classification with CNNs, video classification with 3D CNNs, and video classification through sequential RNN processing of frame-by-frame features extracted with CNNs.

Index Terms—Basketball, NBA, Videos, Shots, Classification, Deep Learning.

1 Introduction

Event and action recognition in videos are major areas of interest in computer vision. Great advances to single-person action/event detection have been made, but there has been markedly less progress in videos with multiple actors. One school of thought in considering events with multiple actors is that there exists a single, or small group of, actor(s) that determine overall event classification. This requires models with the ability to "attend" to certain players, but discerning and tracking people in a video implies expensive frame by frame annotations. Although "weakly supervised" strategies can get around this, they require complex architectures that take very long to train. The datasets for these methods are also complex, including specific video clips of events, segmented images to train drawing bounding boxes around individuals, and in many settings, data annotating artifacts (e.g. a basketball). We experimented with other alternatives to see if a fair trade off between complexity of architecture and classification success could be entertained.

We experiment with:

- Method 1: CNN frame based classification
- Method 2: 3D CNN video based classification
- Method 3: CNN + RNN video based classification

We hypothesized that deep networks would have the ability to learn how features in shot footage progress sequentially, and how these progressions map to specific actions. For frame by frame classification we aggregated the predictions of the CNN over 40 frames. With 3D CNNs and our combined CNN + RNN architecture, we hoped to track features through time. We decided to use the entire frame as global information can provide context about the action. In addition, creating accurate attention models is quite demanding as highlighted in the next section of this paper.

2 Related Work

A huge inspiration and proof of concept for our approach was *Detecting Key Actors in Multi Person Events* by Vignesh Ramanathan and other researchers [1]. The paper describes a model to detect and classify events in basketball videos. They used annotations during training to: 1) teach a region-based convolutional neural network (RCNN) to track players across time, 2) learn "time-varying attention weights" to incorporate the features of the RCNN tracks, and 3) process attended features and global features to classify events.

In the Ramanathan 2015 model, each frame went though an Inception7 network for global features, and localized features were made per player using another Inception7 and spatial

pyramid pooling. They used the global and player features to calculate three more vectors with two BLSTMs and one LSTM - for global, player, and event features respectively. They also used several auxiliary nets and a KLT algorithm [2]. Although this paper beat every technique it was compared to, it's saturated with models and the final model took 20 GPUs and an entire day (100k) iterations, to train. It's safe to infer their approach is not feasible for widespread use, and not desirable for real time classification.

This inspired us to construct significantly simpler networks that could be applied in real time with meaningful impact - so we reduced the scope of our approach to 5 categories: free throws, dunks, two pointers, threes, and layups. We also completely ignored determining shot success, as we didn't have enough data to differentiate misses.

3 Data

We created our own dataset for this project. We segmented and labeled 10 NBA games that took place between October and December 2018. The game videos were in HD 720p quality, and we only used clips from similar camera angles. The five types of events we aimed to categorize were: free throws, layups, threes, twos, and dunks. Once we had clips for each event, we used OpenCV to sequentially extract 40 frames over evenly spaced intervals from each shot clip. Our total dataset was 2006 video clips represented as sequences of 40 frames. The distribution of the shot classes in our dataset was as shown in Figure 1.

We organized each clip into a folder containing its constituent frames and created two data loaders: one that loads individual frames by shot class for frame-by-frame classifiers, and another that loads the entire 40 frame sequence associated with each clip for sequence based classifiers. Finally, we split our dataset into training, validation, and testing partitions by randomly assigning 75%, 10%, and the remaining 15% of shots in each class to the previously mentioned partitions respectively.

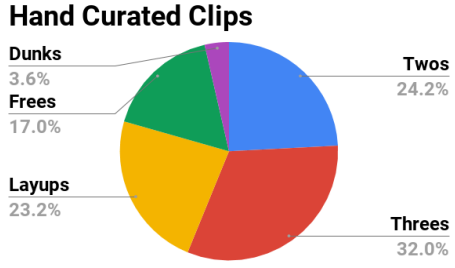

Figure 1. Overall Shot Frequencies

4 Methods

4.1 Frame-by-frame Classification with CNNs

Convolutional Neural Networks (CNNs) are extremely good at image classification. We believe that at some basic level, there may be discernible differences between the deterministic features in frames per type of basketball shot. For instance, when a freethrow is being taken, players line up in a very recognizable pattern (see Figure 2 for reference). Thus, we think training a convolutional neural network on individual frames from our dataset to detect such patterns is an appropriate solution for our problem. We experimented with the ResNet18 [3] model for this task.

ResNet is a state-of-the-art CNN architecture that uses residual blocks to permit the construction of deep networks without degradation. The inputs to the ResNet model were 128x128 images (scaled and padded appropriately) from our dataset, which resulted in 512 features extracted by the network's last average pool layer that are then fed into a fully connected layer with 5 outputs corresponding to each of our shot classes. Resnet18 is a popular variant of this network with 18 residual layers. We chose this variant because of its relative shallowness, which we deemed would be appropriate due to the increased risk of overfitting resulting from the scarcity of our curated dataset.

4.2 Convolution across 3 dimensions with 3D-CNNs

Both players (player features) and the spatial set up of the court (spatial context) contribute

Figure 2. Freethrow frame from our dataset

to shot classification. Deep nets like ResNet can learn this for a single frame at varying levels. In these architectures, earlier parts of the network learn edge recognition, and the deeper layers are increasingly abstract but create some comprehensive representation of players and their spatial interdependence. To truly understand the scene, temporal data must also be processed. One way to account for that is convolution in the temporal dimension.

3D CNNs have been shown to be an effective tool for action recognition [4]. Until recently, substantially deep CNN's for action recognition haven't been explored due to the large number of parameters and threats of accuracy degradation. However, deep models would be preferred due to their ability, in best case scenarios, to improve recognition accuracy. The deeper layers contribute to an improved approximation of mappings and reduce error.

4.2.1 3D ResNet

ResNet has been successful because its residual feature mappings are relatively easier to optimize, and it's architecture permits the use of deeper networks without the threat of degrading accuracy. Applying 3D structure to ResNet contributes further improvements to action recognition performance.

The difference between the 3D network and the original ResNet is the number of dimensions of the convolutional kernels and pooling layers.In our implementation, each 40 frame input clip to this model is divided into 16 frame temporal sections fed in by an average pool layer into a fully connected network.

4.2.2 3D Inception

Inception Networks are CNNs with that reduce structural error while increasing efficiency [5]. They are built around inception "blocks" that factorize larger filters into smaller ones (ex. 5 x 5 filters into two 3 x 3 filters or n x n into 1 x n and n x 1 filters). Each block also uses multiple kernel sizes to capture variations in spatial frequency of info, and use 1 x 1 convolutions before larger filters coupled with concatenation at the end of the block to preserve dimensions (each filter uses different padding).

The architecture we adapted for this paper was based on GoogleNet (Inveption V1) and modified by this paper [6]. Training it from scratch was successful, but fine tuning on top of a previously trained on Imagenet was exceptionally successful. We also replaced the final convolution layer with a fully connected layer and pooling (only loaded parts of pre-trained state). We also used dropout and trained the network with different schedulers, since it was prone to over-fitting. This net's architecture was built for 224x224 inputs, so the images were re-sized to preserve aspect ratio, and padded with zeros.

4.3 Frame-by-frame Feature Extraction using CNN + Sequential Processing with RNN

Another method for video classification through time we considered was passing sequences of features extracted from frames by a CNN into a Recurrent Neural Network (RNN). RNNs are effective at exploiting dependencies in sequences of inputs for output classification. We have obvious reason to believe that for our problem, the classification of a shot clip should depend strongly on the temporal relationship between the frames that make it up. Previous work in action recognition with RNNs suggests that it is often useful to feed in sequences of extracted features from individual frames to the RNN for classification [7]. Thus, we experimented with using the ResNet18 model described previously to extract features from each successive frame and constructing a sequence of 40 feature vectors for each video clip, which was then used as input for a

simple (unidirectional) Long Short-Term-Memory (LSTM) model. The feature extraction was accomplished by removing the last fully connected layer of the ResNet model and collecting the output from the previous layer. This gave us sequences of 40 512-dimensional feature vectors for each video clip. Our LSTM model took these sequences as input and passed them into a hidden layer with 64 output units that were then passed through a dropout layer and a fully connected layer with 5 outputs corresponding to each of our shot classes. The overall model architecture is depicted in figure 3.

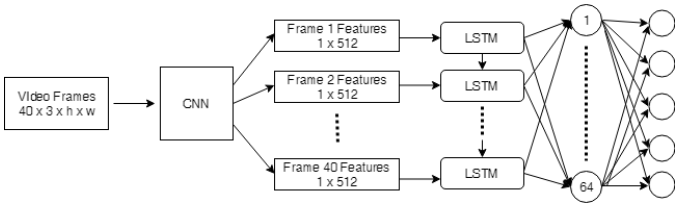

Figure 3. ResNet18 + LSTM Model architecture

5 Results

5.1 Model Accuracy

We collected training and validation accuracy for proper classification and top 2 accuracy in our models. We favored these metrics to loss because we had few classes and a very skewed distribution of data points. It was also easy to tell empirically. Our benchmarks were simple benchmarks when evaluating error: we compared to a uniform random guessing strategy, which would yield an expected accuracy of about 20%, and guessing the most common shot class which would result in 32% accuracy. In order for our optimal models to be of value, they would need to perform significantly better than these strategies.

5.1.1 Frame-by-frame ResNet18

Our ResNet18 model trained over 25 epochs with SGD using a learning rate of 1e-3 and dropout of 0.5 evolved as shown in figure 4. It achieved a classification accuracy of 60.14% and top-2 accuracy of 86.15% on the testing set with a confusion matrix as shown in figure 5.

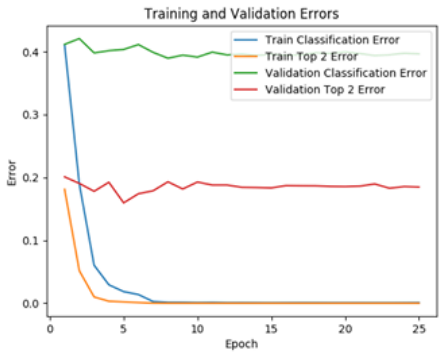

Figure 4. ResNet18 Training and Validation Errors

		Predicted				
		dunk	freethrow	layup	two	three
Truth	dunk	1	1	4	4	0
	freethrow	1	42	2	0	5
	layup	5	6	34	16	8
	two	1	3	8	35	25
	three	0	7	4	9	75

Figure 5. ResNet18 Confusion Matrix

5.1.2 3D Resnet

Our 3D ResNet model was trained with the ADAM optimizer. The optimal hyperparameter selections were a learning rate of 1e-5 and weight decay of 5e-3. We also made use of a reduce on plateau scheduler, which decreased the learning rate by a factor of 10 whenever the model's validation error stopped improving. The model's training errors over 25 epochs are shown in figure 6. We achieved test classification and top-2 accuracy's of 67.57% and 89.19% respectively using this model, and it's confusion matrix is shown in figure 7.

5.1.3 3D Inception1

The 3d Inception network was trained after preloading an imagenet model. It used a learning rate of 1e-5 with the Adam optimizer and 5e-3 weight decay, as well as 45% dropout rate. This model achieved a test classification accuracy of 70.27% and a top 2 accuracy of 91.75%.

5.1.4 Resnet18 + LSTM

For our model consisting of a Resnet18 for feature extraction and an LSTM for sequence

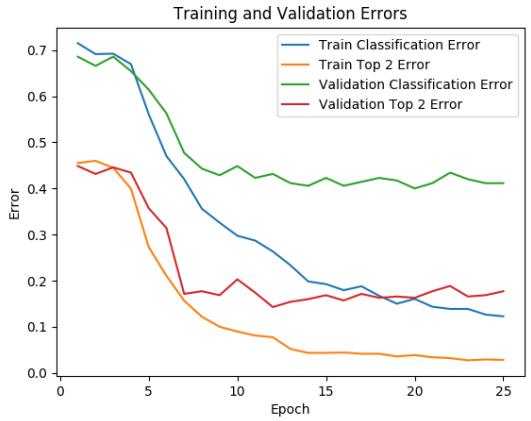

Figure 6. 3D ResNet Training and Validation Errors

		Predicted				
		dunk	freethrow	layup	two	three
Truth	dunk	0	1	6	0	3
	freethrow	0	42	3	1	4
	layup	0	0	44	19	6
	two	0	0	10	34	28
	three	0	1	4	10	80

Figure 7. 3D Resnet Confusion Matrix

classification, we carried out training over 25 epochs using SGD with a learning rate of 1e-3 and dropout of 0.75 between the hidden and output layers. This model achieved a best classification accuracy of 63.18% and top-2 accuracy of 75% on the test set. The confusion matrix results are shown in figure 10.

5.1.5 Ensemble Model

As a final task, we experimented with creating an ensemble model from all of our models. Specifically, we used an approach inspired by boosting, in which we assigned a distinct weight to the outputs of each of the four trained models we created for our classification task. The model then arrived at a final classification decision by considering the weighted sum of the model outputs. We chose the model weights by performing a grid search over weights on our validation set and selecting the configuration that minimized validation error. The derived weights were as shown in Table 1. This configuration resulted in a test classification accuracy of 79.05% and top-2 accuracy of

		Predicted				
		dunk	freethrow	layup	two	three
Truth	dunk	0	0	9	1	0
	freethrow	0	46	3	0	1
	layup	0	0	59	2	8
	two	0	2	17	19	34
	three	0	3	3	69	20

Figure 8. Inception 3d Confusion Matrix

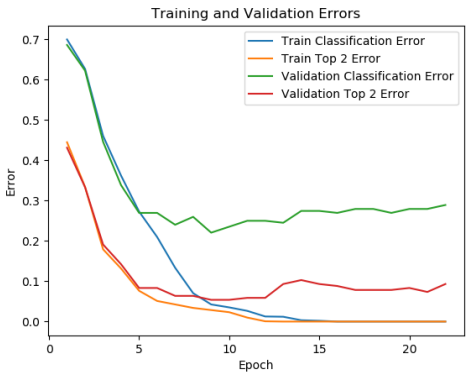

Figure 9. Inception 3d Training and Validation Error

94.96%. The confusion matrix for the ensemble model's classifications is shown in figure 12.

6 Conclusion

When it came to training and evaluation, training 2d convolutional nets over video frames was less efficient that using a 3 dimensional CNN. Out of the models we trained and tested, the most effective single net was the **Inception1 3D**. However, the Ensemble Model maximized accuracy. If we had a large enough dataset with evenly distributed shot classes, we could have further improved the Ensemble Model by using the confusion matrix to make averaged predictions with false positive rates. This would exploit some classifiers being better at identifying certain shot types.

Model	Weight
Resnet18	0
Resnet18 3D	0.15
Inception1 3D	0.74
Resnet18 + LSTM	0.11

Table 1
Ensemble model weights

Truth \ Predicted	dunk	freethrow	layup	two	three
dunk	0	0	5	3	2
freethrow	0	45	0	1	4
layup	4	0	32	19	14
two	0	0	18	25	34
three	1	0	8	14	72

Figure 10. Resnet 18 + LSTM Confusion Matrix

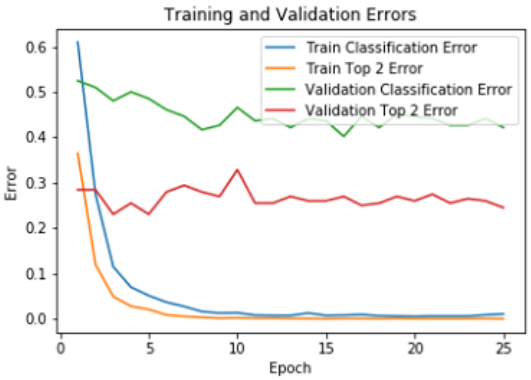

Figure 11. ResNet18 + LSTM Training and Validation Erros

All of our models performed reasonably well, especially when compared to our benchmark strategies. This makes us more confident in our hypothesis that there is some learnable structure underlying basketball shot videos that can be used to automatically classify them. One thing to note is that although our results are decent, almost all of our models were nearly perfectly fitting the observed samples set within 20 epochs of training. So our experiments can be strongly regarded as a thorough proof of concept that will hopefully inspire more approaches at this problem using larger datasets.

6.1 Reflection

For this project, I contributed by proposing the overall idea, collecting my share of the data, investigating methods for accomplishing our task, and implementing the frame-by-frame classification as well as combined CNN & LSTM methods. I also had a lot of responsibility in setting up the infrastructure of the project (dataloaders and train & test frameworks).

Truth \ Predicted	dunk	freethrow	layup	two	three
dunk	1	1	4	0	4
freethrow	1	42	2	0	5
layup	5	6	34	16	8
two	1	3	8	35	25
three	0	7	4	9	75

Figure 12. Ensemble Model Confusion Matrix

REFERENCES

[1] V. Ramanathan, J. Huang, S. Abu-El-Haija, A. N. Gorban, K. Murphy, and L. Fei-Fei, "Detecting events and key actors in multi-person videos," *CoRR*, vol. abs/1511.02917, 2015. [Online]. Available: http://arxiv.org/abs/1511.02917

[2] C. J. Veenman, M. J. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 1, pp. 54–72, 2001.

[3] S. R. K. He, X. Zhang and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, no. 1, p. 770–778, 2016.

[4] K. Hara, H. Kataoka, and Y. Satoh, "Learning spatio-temporal features with 3d residual networks for action recognition," *ICCVW*, vol. abs/10.1109, 2017. [Online]. Available: https://arxiv.org/pdf/1708.07632.pdf

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[6] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *CoRR*, vol. abs/1705.07750, 2017. [Online]. Available: http://arxiv.org/abs/1705.07750

[7] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.