CS230

Deep Learning for In-Game NFL Predictions

Cameron Taylor*
Graduate School of Business
Stanford University
cntaylor@stanford.edu

Abstract

This project explores deep learning methods for predicting in-game NFL play outcomes. Systems that predict play outcomes in-game may help NFL team play callers improve their strategies at play-call time. I focus on two important outcomes, the yardage outcome of a play and the offensive play call (pass or run). My models combine pre-snap "situational data" with pre-snap images of the play. I run an array of learning models, including shallow CNNs and transfer learning, with and without the image data. The models show that while the features have no signal for yardage outcomes, offensive play calls can be predicted with much higher accuracy than benchmark models. However, the marginal value of the image data and sophisticated deep learning models is very low for both tasks.

1 Introduction

American football is a game with high demands for strategy. Because it is a play-by-play sport, both the offense and defense must make a "play-call" at every point in the game. Systems that improve these play-calling systems could have a major impact on in-game strategizing and how teams compete. In this paper I assess whether deep learning can improve NFL strategy by investigating whether in-game data combined with cutting-edge models can predict play outcomes. If these methods can predict outcomes better than other benchmarks, then they could be incorporated into the American football strategist's tool box as a way to improve in-game decision making.

One methodological novelty of this project is the exploration of combining in-game image data with traditional in-game data. Image data and tasks involving image recognition or classification have proved to be an important and exciting area of deep learning. A methodological focus of this project is in assessing the value of this image data over other data for the task at hand.

My approach combines pre-play image data with other pre-play data to predict two outcomes (separately): (1) the resulting yardage of the play; and (2) the offensive play call, which can be either "pass" or "run". To include the image data I use two models. First I build a shallow convolutional neural network. Second I use transfer learning with VGG19 [8]. My approach to combining the image and non-image data is to pass the image data through the convolutional neural network and then stack the resulting encodings with the other data and pass through another neural network layer.

My results for predicting yardage outcomes show that learning models do not outperform basic benchmarks that do not use any of the rich features I examine. My conclusion is that there is very little signal in the features I collect for the predicting yardage, and that predicting yardage pre-play is very challenging, though future work could explore this limitation more.

*cameronntaylor.github.io

CS230: Deep Learning, Winter 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

My results are much more encouraging for predicting offensive play calls. Sophisticated models significantly outperform benchmark models. However, the marginal value of the image data in this task is essentially 0, suggesting that either the image data provides little value in predicting the play call, or more/different data and models are required.

2 Related work

There is a relatively small literature in machine learning and deep learning on American football. Deep learning papers have attempted to develop player tracking technologies based on video data [1, 7], real-time value changes as the play evolves [11], play classification based on image data [3, 1, 10, 5], and optimal passing strategies based on positional data [4].

Machine learning papers have explored predicting overall NFL game outcomes [9, 2]. While these questions are important and interesting, they are not directly useful in improving strategy, the main motivation for this project.

To the best of my knowledge, the questions I ask in this paper are novel for the literature. The most similar papers to my own are those submitted as part of the 2020 NFL Big Data Bowl.[2] The prompt for the competition states "When an NFL ball carrier takes a handoff, how many yards should we expect the play to gain?". My project approaches a different version of this question using pre-play features to predict how many yards are gained instead of in-play features. I also address play call prediction. These papers use positional data provided by the NFL on where players are located on the field, whereas my approach utilizes more end-to-end deep learning, inputting only the raw image data instead of coding player locations.

The methodology I use in this paper uses convolutional neural networks and transfer learning from VGG19 [8]. The model architecture for the shallow CNN is inspired by LeNet-5 [6].

3 Dataset and Features

My dataset comes from two sources. The first is the Kaggle dataset "Detailed NFL Play-by-Play Data 2009-2018".[3] The Kaggle dataset contains my main outcomes of interest (yards gained on the play and play type) and contains rich features related to the situation of the game that may give some predictive power to what will happen in the play. For example, the features include the current score of the game and the yardline that the offensive team is on. A full set of features can be found in the Appendix.

The second source is the "NFL Rewind Coaches Film" videos from the NFL's website. I manually collected data from the NFL Rewind site from 9 games from the 2016 NFL season. My data collection consisted of taking 1 screenshot of my entire screen 0-3 seconds before the snap of each play. I collected images from 1,055 plays and end up having 1,049 valid plays to train, validate and test my models on. I use a 75/12.5/12.5 split for training, validation and testing. While collecting my data I intentionally sampled plays with a yardage outcome instead of a penalty or other outcome.

I then processed this data by cropping it to remove the parts of my screen that were not in the play and also to reduce the size of the images. I extracted the RGB pixel values using the Pillow package in Python and merged it with the Kaggle data at the play-by-play level to complete merging the data. The size of each image ends up being (230, 119, 3). See a sample raw image in Figure 1.

4 Methods

4.1 Predicting Yards

I focus on two metrics for models predicting yardage outcomes. The first is the mean absolute difference in yards and the second is the median absolute difference in yards. Both of these are preferred to metrics such as mean squared error (MSE) in this context as MSE will overweight rare long-yardage plays that are much less predictable in general and so less important to strategizing.

[2] https://operations.nfl.com/updates/the-game/2020-big-data-bowl-results/

[3] https://www.kaggle.com/maxhorowitz/nflplaybyplay2009to2016

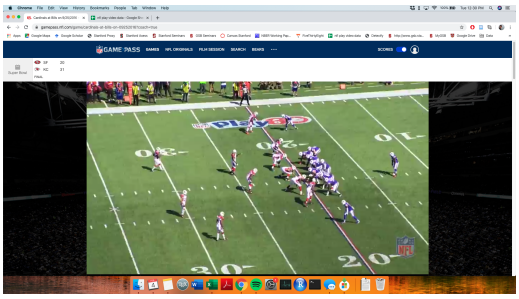

Figure 1: Sample Screenshot

With this metric in mind, all the deep learning models utilized (except the LASSO) in this project used a cost function of mean absolute yards:

$$J = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}^{(i)} - y^{(i)}| \tag{1}$$

The benchmark model is to predict the median outcome from the training data on the test data.

The benchmark machine learning models are a tuned LASSO model and a tuned Random Forest regression model. These models only use the Kaggle situational data and use no image data. They use a loss of MSE to be trained. They are meant to serve as benchmarks to measure the extra value of the deep learning and image data in the following sense:

$$\text{marginal value of image} \approx \text{accuracy deep learning with image} - \text{accuracy machine learning w/o image} \tag{2}$$

The first deep learning model employed is a shallow convolutional neural network inspired by LeNet-5 [6] that passes the image data through two convolutional layers (with pooling), and then stacks the output of these layers with the situational data, passes this through one more fully connected layer before passing to a linear output layer. A shallow CNN is chosen because of the small training set. L2 regularization was also added to the fully connected layer and output layers because of the small training set to avoid overfitting. See Figure 2 for more details.

The second deep learning model utilizes transfer learning from VGG19 trained on ImageNet [8].[4] This methodology essentially does the same thing as my shallow convolutional neural network approach described above except that I replace my own convolutional layers with the output from some layer of the pre-trained VGG19. I tune over which layer of VGG19 to keep, where I consider the outputs of each of the 5 maxpooling layers in VGG19.[5] The idea behind using transfer learning

[4]The model weights are loaded directly through Tensorflow Keras Applications.

[5]See this link for an overview of the VGG19 layers. I consider only choosing over layers with the prefix "pool".

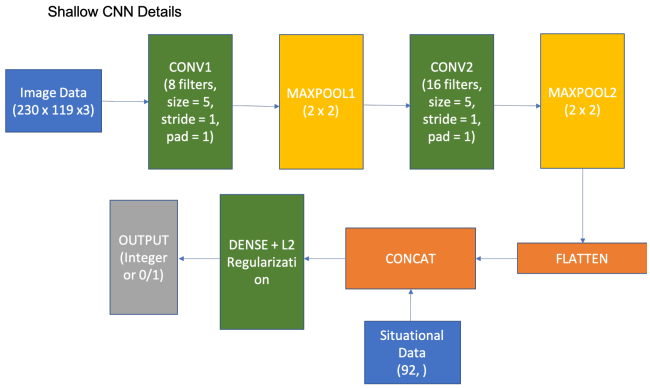

Figure 2: Shallow CNN Architecture

Model	Mean Absolute Yards	Median Absolute Yards
Benchmark: Guess Median of Training y	6.18	3.00
LASSO (HPs: α = 0.0176)	6.18	4.00
Random Forest (HPs: number trees = 3, max depth trees = 3)	6.16	3.66
Shallow CNN (HPs: learning rate = 2.5×10^{-5}, epochs = 15, mini-batch = 32, # hidden units in dense layer =5, L2 reg = 0.001)	6.18	3.00
VGG19 Transfer Learning (HPs: learning rate = 2.5×10^{-6}, epochs = 15, mini-batch = 32, # hidden units in dense layer = 2, conv layers from VGG = 5)	6.21	3.00

Notes: HP = Hyperparameters.

Table 1: Results for Predicting Yardage Outcomes

with VGG19 is that VGG19 trained on ImageNet should be able to capture low-level features in images that translate to capturing player locations in my image data, which then might be useful for predicting the yardage or play call. Since my dataset is small and the number of parameters in the VGG19 model so large, I do not consider training any of the VGG19 layers myself.

The deep learning models are trained using an Adam optimizer with default optimization algorithm parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

In all these models since yards must be integer valued in my data, I take the final predictions to be the integer-rounded values of the predictions provided by the algorithms.

4.2 Predicting Offensive Play Call

Since the offensive play call is either a pass or a run, I use binary cross-entropy as my loss. As well, since the classes are close to balanced, I use accuracy as my evaluation metric.

The benchmark model for play-call is to predict the most common play outcome ("pass").

The other models used are identical to those used for predicting yards but have a binary output (run or pass) instead of an integer valued output. Thus I replace the final linear output layer with a sigmoid output layer.

5 Experiments/Results/Discussion

5.1 Predicting Yards

The results for the main experiments for predicting yardage outcomes are given in Table 1. The LASSO hyperparameter was picked by tuning over 100 randomly selected points on a log-scale from 10^{-6} to 1. The Random Forest estimator is tuned over the number of trees and max depth of trees with a grid around the size of 50. For the Shallow CNN the focus of tuning was on the learning rate and the number of hidden units to put in the dense layer. I chose a mini-batch of 32 as this seemed standard. I examined about a dozen different models that varied the learning rate and number of hidden units and chose the model with the best mean absolute yards value on the dev set. The VGG19 Transfer Learning tuning procedure focused on the learning rate and the number of pre-trained convolutional VGG19 layers to use. I evaluated about 20 of these models on the dev set. 15 epochs was used for both deep models to avoid overfitting because of the small dataset size.

The results are quite striking in how negative they are: the shallow CNN model is able to reproduce the benchmark metrics, but at a substantially larger cost than just guessing a constant for all plays, which has essentially 0 computational cost. Many other CNN models did not perform as well. The Random Forest method improves on the benchmark by only 0.02 yards.

It seems likely that the image data I have gathered provides very little signal for predicting yardage outcomes since I utilized transfer learning which should overcome the fact that I have a small dataset.

Model	Accuracy
Benchmark: Guess all plays are pass	0.546
LASSO (HPs: α = 0.0176)	0.568
Random Forest (HPs: number trees = 2, max depth trees = 7)	0.614
Shallow CNN (HPs: learning rate = 1.0×10^{-3}, epochs = 15, mini-batch = 32, # hidden units in dense layer =4, L2 reg = 0.005)	0.606
VGG19 Transfer Learning (HPs: learning rate = 1.0×10^{-3}, epochs = 10, mini-batch = 32, # hidden units in dense layer = 4, conv layers from VGG = 5)	0.606

Note: HP = Hyperparameters

Table 2: Results for Predicting Play Call

Overall, the results suggest that (1) it is very difficult to predict yardage outcomes at play-time with the data I collected and (2) deep learning models trained on the limited dataset I have collected cannot improve on the simplest benchmark methods. Note also that the image data provided no marginal value over the non-image data.

5.2 Predicting Offensive Play Call

The results for the main experiments for predicting yardage outcomes are given in Table 2. The tuning procedures were essentially the same as for predicting yardage outcomes, except that I also tuned the L2 regularization parameters for the shallow CNN over a few grid points.

Here, we see that the deep learning methods provide a substantial increase of 6 percentage points in accuracy over benchmark methods, suggesting that the data used provides valuable signal and could potentially serve as an input to improving American football strategy as defensive coordinators often want to be able to predict whether the offensive team will call a run or pass play.

Comparing the results from the Random Forest and the deep learning models, it appears that there is no value to the image data and more complex deep learning methodologies using the heuristic (2). The Random Forest model on only situational data achieves a higher test accuracy than the other models.

Thus it appears all the valuable information for predicting play calls is already contained in the situational data, and can be extracted with less sophisticated learning techniques such as the Random Forest used here. Importantly though, the performance is well above the benchmark.

6 Conclusion/Future Work

This project explored using deep learning methods for predicting play outcomes in the NFL. The experiments showed that the pre-snap image data and situational data cannot do a good job predicting NFL yardage outcomes, even when combined with cutting-edge deep learning models. However, the accuracy of predicting offensive play calls is improved with machine learning models, suggesting a role for machine learning in changing NFL in-game strategy.

It was found that the image data I collected was not useful over the other pre-play features for both tasks addressed. While this suggests to me the possibility that seeing the players on the field may not provide much useful information for the tasks at hand, there are two important caveats. First the dataset was small. Second, my image data was severely limited since it was a single snapshot a few seconds before the play. It is sensible to think that multiple images or video data right before the play occurs could have more signal. However, the major challenge for a project exploring this route would be collecting enough examples to take advantage of these high dimensional features.

With more time and a larger team, I would have liked to explore transfer learning with other algorithms including but not limited to YOLO, collected more image data, and also labeled my data with bounding boxes on players so that I could break up my problem into two steps of first capturing player positions, and then using that data to predict the play outcomes.

7 Contributions and Acknowledgements

This was a solo project. However, I would like to thank Leo Mehr for providing many useful suggestions and helpful support throughout this project.

References

[1] Omar Ajmeri and Ali Shah. Using Computer Vision and Machine Learning to Automatically Classify NFL Game Film and Develop a Player Tracking System. page 9, 2018.

[2] Bryan L. Boulier and H.O. Stekler. Predicting the outcomes of National Football League games. *International Journal of Forecasting*, 19(2):257–270, April 2003.

[3] Sheng Chen, Zhongyuan Feng, Qingkai Lu, Behrooz Mahasseni, Trevor Fiez, Alan Fern, and Sinisa Todorovic. Play type recognition in real-world football video. In *IEEE Winter Conference on Applications of Computer Vision*, pages 652–659, Steamboat Springs, CO, USA, March 2014. IEEE.

[4] Dani Chu, Matthew Reyers, Lucas Wu, and James Thomson. NFL Big Data Bowl.

[5] Rishav Dutta, Ronald Yurko, and Samuel Ventura. Unsupervised Methods for Identifying Pass Coverage Among Defensive Backs with NFL Player Tracking Data. *arXiv:1906.11373 [stat]*, June 2019. arXiv: 1906.11373.

[6] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, November 1998.

[7] Timothy Lee. Automating NFL Film Study: Using Computer Vision to Analyze All-22 NFL Film. Class Project, Stanford University CS231A.

[8] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, April 2015. arXiv: 1409.1556.

[9] Shiladitya Sinha, Chris Dyer, Kevin Gimpel, and Noah A. Smith. Predicting the NFL using Twitter. *arXiv:1310.6998 [physics, stat]*, October 2013. arXiv: 1310.6998.

[10] Nathan Sterken. RouteNet: a convolutional neural network for classifying routes. page 12.

[11] Ronald Yurko, Francesca Matano, Lee F. Richardson, Nicholas Granered, Taylor Pospisil, Konstantinos Pelechrinis, and Samuel L. Ventura. Going deep: models for continuous-time within-play valuation of game outcomes in American football with tracking data. *Journal of Quantitative Analysis in Sports*, 0(0), January 2020.

Code

Citations/References: Python, Pillow, Tensorflow, Keras

Github Link: https://github.com/cameronntaylor/cs230project

Appendix

Features from Kaggle dataset

- Drive number (categorical - one-hot encode)
- Quarter of the game (categorical - one-hot encode)
- Down that the play occurs on (categorical - one-hot encode)
- Time passed in the game (categorical on the minute - one-hot encode)
- Yard line play occurs on (continuous)
- Yards to go to first down (continuous)

- Offensive team (categorical - one-hot encode)
- Defensive team (categorical - one-hot encode)
- Offensive team score (continuous)
- Defensive team score (continuous)