Predicting Rushing Yards Using a Convolutional Autoencoder for Space Ownership

Abstract: Using the data provided for the 2020 Big Data Bowl, we utilize the methods in Fernandez and Bornn's 2018 paper [3] to create a grid of field control values for each play at handoff. We then build a convolutional autoencoder in using the keras package in R to reduce the dimension of the data. Finally, using the methods of Pospisil and Lee [4], we create a conditional density estimation model to predict the number of yards gained at the point of handoff.

1. Introduction

This year, the NFL's Big Data Bowl had participants set out with the goal of predicting the distribution of the number of yards gained on a rush using NFL tracking data. Any attempt to tackle this problem successfully would likely have to include some element of space control, as football is inherently a game centered around the ownership of space. When the ball is handed off as a rushing play begins, the offensive and defensive linemen battle it out in the trenches to create gaps for the running back to squeeze through, while wide receivers block defensive backs near the sidelines to potentially allow the rusher to swing out wide. Thus, creating a good measure of space control is imperative to successfully predict yards gained on the play.

A few methods have been utilized in football and other sports to quantify space ownership. Some methods, including Cervone et al [2], have utilized Voronoi tessellations, but these methods fail to incorporate any elements of speed and direction that are captured by player tracking data. A method that is much more successful in taking some of this movement information into account is the methodology defined in the paper by Fernandez and Bornn [3]. This methodology utilized a bivariate normal distribution whose mean and variance-covariance matrix is defined by the locations, speeds, and directions of all players on the field.

In this paper, we utilize the methods in [3] to create what amounts to an image of offense control at the exact moment the ball is handed off to the running back. We then create a convolutional autoencoder to shrink the dimension of the data while still capturing all the information contained in each image. We then illustrate how this data could be useful by utilizing the methods in [4] to create a conditional density estimator for the number of yards gained in a play.

2. Field Control using Fernandez and Bornn's Methodology

Fernandez and Bornn [3] lay out a parametric approach to modelling field control in soccer in their 2018 paper. The paper essentially models a player's control at a specific point on the soccer field with a bivariate normal distribution whose mean and shape is dictated by the player's direction and speed. This methodology inherently captures multiple football effects that could be happening simultaneously; for example, if the offensive line is winning the blocking battle, both the offensive and defensive linemen will be moving towards the defense's end zone, shifting the offense's control in a positive manner. This also mitigates some more surface-level statistics, such as the number of defenders in the box at the moment the ball is handed off - this effect is directly captured by measuring control, as it will swing more in the defense's direction if they have more players in the area.

After standardizing the tracking data so that all plays were moving from left to right, we decided to measure a team's control over a grid of points on the field that depended on the rusher's trajectory. Using the rusher's speed and direction, we projected their position (using a similar method to finding the mean of the player's bivariate normal distribution). We then took measurements every half yard, 8 yards up, 8 yards down, 2 yards backwards and 6 yards forwards. This resulted in a 16 by 32 grid of measurements for each play.

The problem with using field control as a way of forecasting a run's success is the fact that it is an incredibly granular measure. We cannot effectively measure how much of the football field the offense controls by looking at one or two points on the field; football is a game of space ownership, and space ownership close to the ball and away from the ball intuitively both play a factor in the success of a run. However, there is a curse of

dimensionality that arises with increasing the number of control points measured on the field - if we want to create a grid of 10 yards by 10 yards with control measurements calculated at every half yard, we end up with 441 variables. There has to be a compromise between the degree at which we measure space control on the football field and the dimension of the data that we use to model yards gained at handoff.

3. Building a Convolutional Autoencoder using Keras in R

As we begin to attack the problem of reducing the dimensionality of the control data, consider the following: a grid of control points (which lie between 0 and 1) can be considered to be a grid of pixels that make up an image. Certainly, when one looks at a field control plot without players, it looks like a picture.

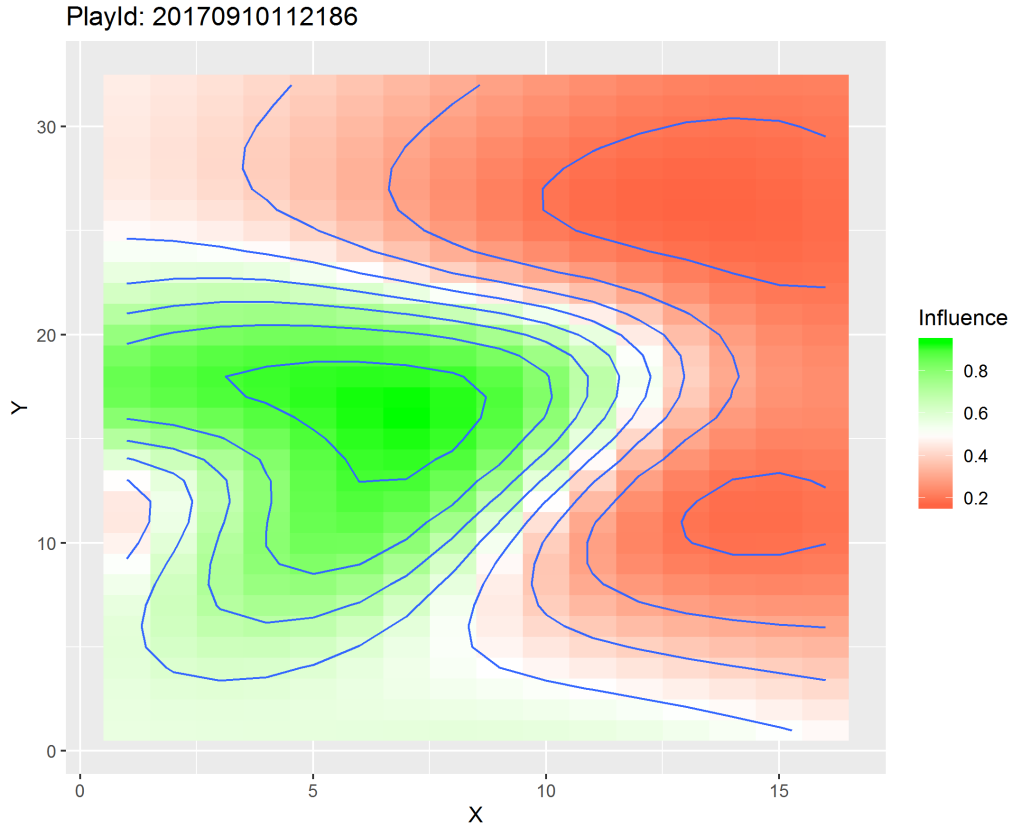

Figure 1: An example of a field control plot generated with the Big Data Bowl data.

3.1. Convolutional Autoencoder Methodology

Convolutional neural networks are a type of deep neural networks that are frequently used to analyze images; they are incredibly powerful tools for interpreting data that has high spatial correlation. These neural networks are excellent at identifying the subject of pictures. The problem laid out in the Big Data Bowl, however, is not a simple classification; we were tasked with predicting a distribution of expected yards gained.

In statistics, there are a couple of different widely used dimension reduction techniques. One of the most common ones, principal component analysis, cannot be used in our case; this method requires the data to be approximately normally distributed, and since we are dealing with values that are strictly scaled between 0 and 1 which are certainly not normally distributed, we cannot use this method. Another commonly used method is factor analysis; the assumptions needed to run factor analysis are also not necessarily met in this scenario. However, there is a machine learning method that is excellent at dimension reduction - the autoencoder. Autoencoders are deep neural networks whose inputs and outputs are the same. The idea is to shrink the number of neurons used in each hidden layer, then build them back up again to what we started with. We can continue to shrink the number of neurons in the smallest layer until we can no longer accurately predict our original data; we then cut the neural network at the middle layer and use these neurons as our output. Theoretically,

all important information should be stored in these neurons, since we can predict our original data using only this information.

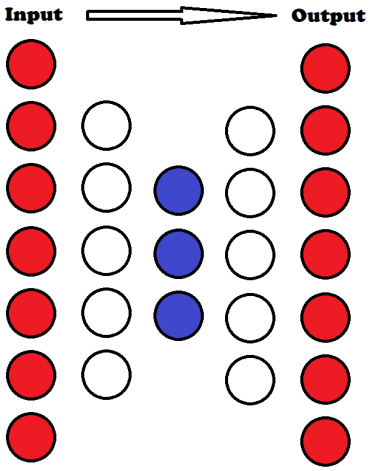

Figure 2: Convolutional Autoencoder structure; input and output layers are the same, while the target output layer is in the middle and has the fewest number of neurons.

Combining these methods, one can build a convolutional autoencoder; an autoencoder whose inut and output are an image, and whose hidden layers are all convolutional neural networks shrinking or building neurons back up. Before building our model, we attempted to standardize the images by flipping images where the runner is heading towards the right sideline so as to avoid flipped images giving different results. Using the keras [1] package in R, we took our 16 by 32 grids of points (512 points per image) and tried to shrink them down to $4 \times 2 \times 8$ (64), $4 \times 2 \times 6$ (48), $2 \times 1 \times 16$ (32), and $2 \times 1 \times 8$ (16) dimension arrays. We then compared output images to initial images, as well as loss rates, to decide which were the most successful.

Each autoencoder was trained for 40,000 epochs (done iteratively, 5,000 epochs at a time, over the course of a few days) with batch sizes of 400 and data shuffled between each batch. The autoencoders' structure alternated between convolutional layers and max pooling layers until the appropriate number of neurons was reached in the middle layer, followed by alternating convolutional layers and upsampling layers until the original data dimension was reached. The autoencoder trained on the Big Data Bowl stage one data using an 80/20 train/test split.

3.2. Results

It turned out that the autoencoder whose middle layer had only 16 neurons actually performed remarkably well; for comparison, the autoencoder with 64 neurons reported a test sample loss of 0.576, while the autoencoder with 16 neurons reported a test sample loss of 0.587. Figure 3 contains a few examples of field control images and their reconstructed versions using the autoencoder. Clearly there are imperfections in the reconstructed versions, but the autoencoder manages to reconstruct the general shape and locations of areas on the field that are mainly controlled by the offense or defense.

4. RFCDE Application

Reducing the dimensionality of a field control picture is certainly an interesting exercise, but being able to apply the output to other problems was what really drove creating the neural network. For the data available to us, the most immediate actionable tool we can create is a conditional density estimator for the number of yards gained at the point of handoff, i.e. the goal of the more general Big Data Bowl Kaggle competition. We certainly cannot compete with the machine learning methodology that top Kaggle finishers created in this competition, but we can demonstrate the usefullness of the field control data by applying the reduced dataset to the problem.

In order to create our conditional density estimator, we utilize the methods of Popisil and Lee [4]. Their methodology develops a random forest-based conditional density estimator (RFCDE), which helps capture nonlinearity and interactions between variables as a regular random forest estimator would as well, and is therefore an ideal way to handle complex data structures.

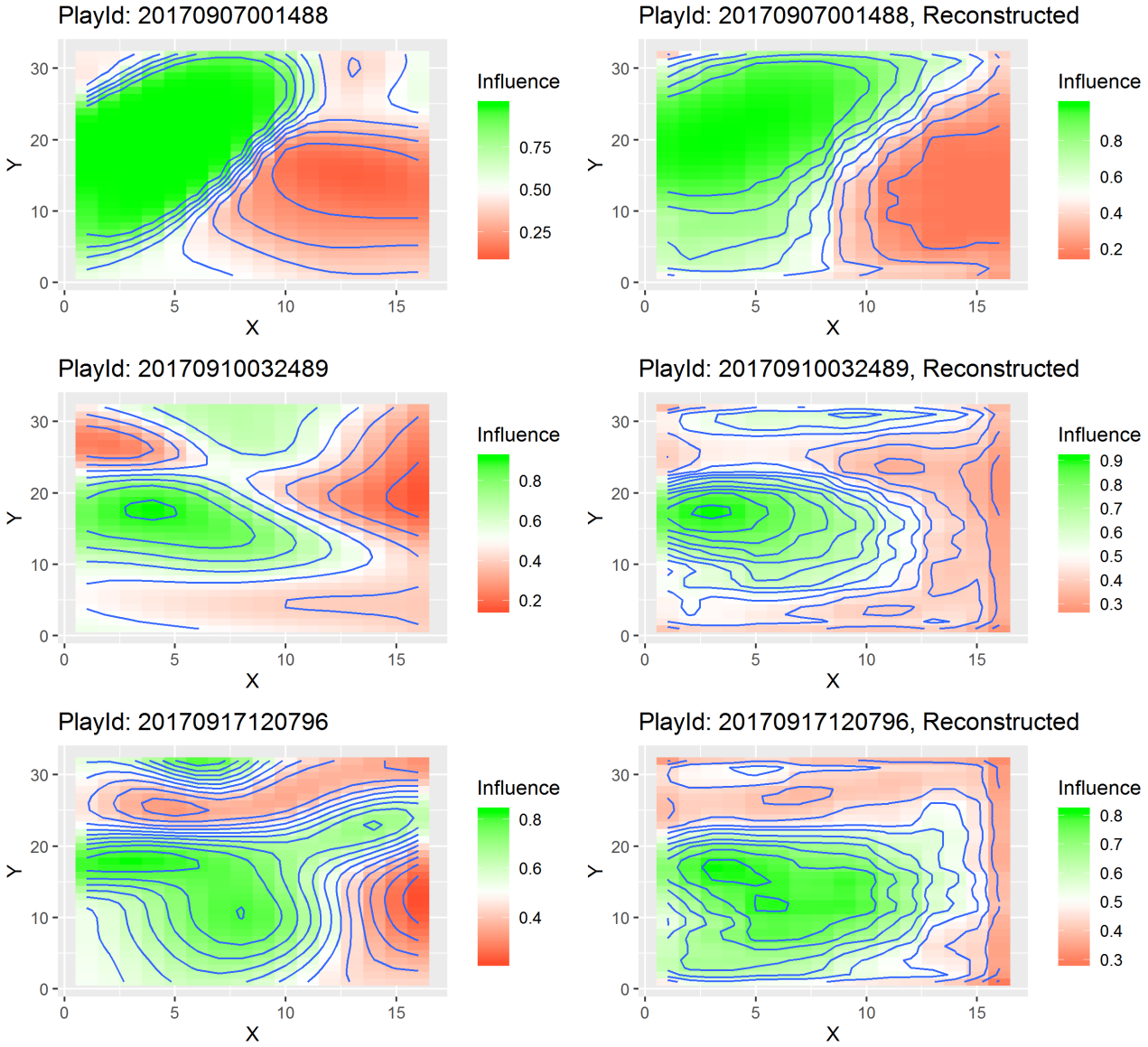

Figure 3: Reconstructed field control plots using the convolutional autoencoder with 16 neurons in the middle layer.

In our application, we utilized all convolutional autoencoder output as input variables into the RFCDE. We also included the distance the line of scrimmage was from the offense's end zone in order to allow the model to understand possible yard outputs. In order to truly test how well the autoencoder output captures the information of field control, only these variables were used as inputs.

The RFCDE output appears to perform well; unfortunately, due to the RFCDE package requiring an internet connection to install in R, we cannot compare our results to the general Kaggle leaderboard. However, results are certainly encouraging; the autoencoder output appears to contain much of the information about how plays might develop.

5. Discussion

In this paper, we have developed a way to significantly reduce the dimensionality of a grid of field control values for use in predictive modelling. Our RFCDE application demonstrates the utility of a convolutional autoencoder's output in predicting yards at handoff; if our results were combined with a stronger predictive modeller, they would likely strengthen the results.

It is worth noting that, while we utilized the methods in Fernandez and Bornn [3] to measure field control at

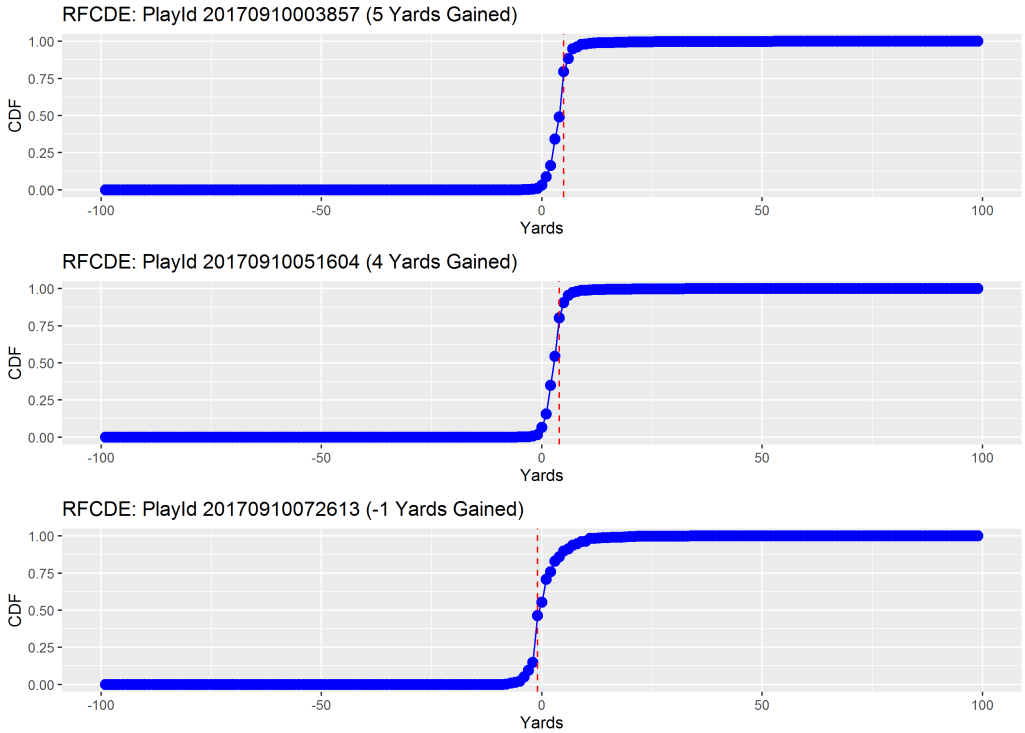

Figure 4: RFCDE CDF predictions using convolutional autoencoder output.

individual points, using other methods that measure control at individual points on the field could also provide similar results.

It is also important to recognize the scope of the convolutional autoencoder that we have created. The strength of this model is that we are analyzing identical moments in time for similar plays - for every rushing play, we analyze the exact moment when the ball was handed off. This model identifies similarities and differences between similar plays at the exact same point in time; this model would likely not be able to analyze trends at a different point in the play. For example, if we had player tracking data from the point when a rusher crosses the line of scrimmage, it is likely the play will have developed to the point where its field control picture is no longer reproducible with our autoencoder. However, we could develop a separate convolutional autoencoder using rushing plays at the exact point where the rusher reaches the line of scrimmage in order to analyze these moments in a play.

As we have just hinted at, our methods could easily extend to understanding how control varies at different points in different types of plays. For example, if we had a similar dataset to the one released during the Big Data Bowl, but for passing plays (i.e. a snapshot at the exact moment the pass is released), we could build multiple convolutional autoencoders to analyze passing success. We could analyze the space around the quarterback to determine the probability of a completion, or we could use information about where the pass is headed to determine whether the the pass is likely to be completed. We could even analyze kickoff and punt returns to understand if there is a rushing lane for the returner to squeeze through. The bottom line is that being able to quantify space ownership and reduce a large grid of field control values to a few important variables could truly help encapsulae some of the subtleties occurring during a play, leading to better predictions in all types of plays.

References

[1] Chollet, F. keras. https://github.com/keras-team/keras, 2019.
[2] D. Cervone, L. B., and Goldsberry, K. nba court realty. *MIT Sloan Sports Analytics Conference* (2016).
[3] Fernandez, J., and Bornn, L. Wide open spaces: A statistical technique for measuring space creation in professional soccer. *MIT Sloan Sports Analytics Conference* (2018).
[4] Pospisil, T., and Lee, A. B. Rfcde: Random forests for conditional density estimation. *arXiv:1804.05753v2* (2018).