A Mixed-Data Predictive Model for the Success of Rush Attempts in the National Football League

December 22, 2019

1 Introduction

In the National Football League (NFL), roughly a third of a teams offensive yardage comes from run plays. Ball carriers are assigned the most credit for these plays, but their teammates (by way of blocking), coach (by way of play call), and the opposing defense also play a critical role. These facets of the game are understood in great detail by coaches, but using data to provide evidence for this "eye-test" is tricky. Traditional metrics such as 'yards per carry' or 'total rushing yards' can be flawed since they can be inflated by long runs and are not informative of the underlying distribution of production. Instead, using detailed data at the time of the hand-off, we will construct a cumulative distribution for the predicted yards gained on the play. We will then utilize this model to investigate the performance of rushers relative to a play's expected outcome to evaluate the value added by various rushers.

2 Data

The data set [5] chosen contains Next Gen Stats tracking data for National Football League (NFL) league games between the 2017 and 2018 NFL seasons. This data is collected by sensors stationed around the stadium that track tags in player's shoulder pads during the game. The (x, y) location of all twenty-two players on the field is accounted for as well as their velocity vector and magnitude of acceleration. Other game conditions such as weather, play information, and game score information are also contained in the data set for a total of 49 unique features. These unique features are listed in full in Table 3 in Appendix A. Figure 1 shows how position and direction data are recorded. Note that the data is specific to the time-of-handoff, i.e. there is no time-series data for plays, only a snapshot of the positional data at the instant that the ball is given to the ball carrier. Moreover, only plays where the ball is handed off are recorded, so there is no data for passing plays or quarterback rushes.

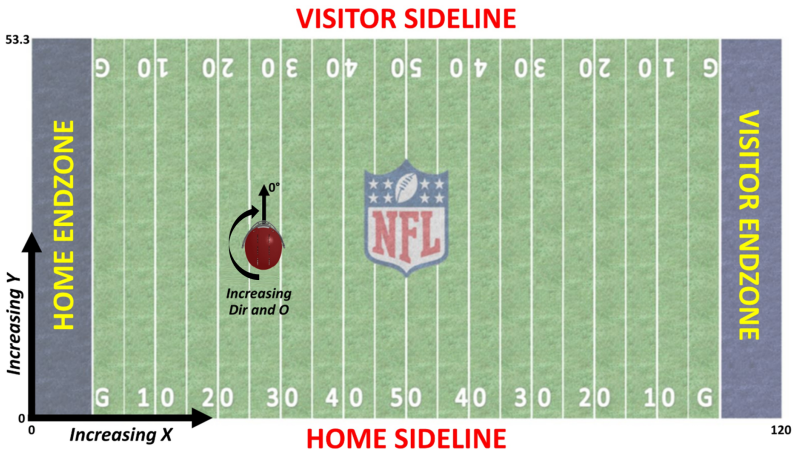

Figure 1: Next Gen Stats Data Description [5].

2.1 Data Cleaning and Feature Engineering

Despite the data coming from the official NFL source, Next Gen Stats, there was a considerable amount of cleaning required to standardize the observations. For example, when observations are fed to a model, it is important to standardize the direction that the team on offense is facing, so that positional information may be understand in context. Moreover, inconsistencies in the data, such as different city abbreviations for different categorical features must be accounted for. Positions and velocities are standardized such that the offense is always facing the right, speed is defined by a counter-clockwise angle, and 0° corresponds to the x-axis. Two examples of players and their respective velocities are presented in Figures 2 and 3.

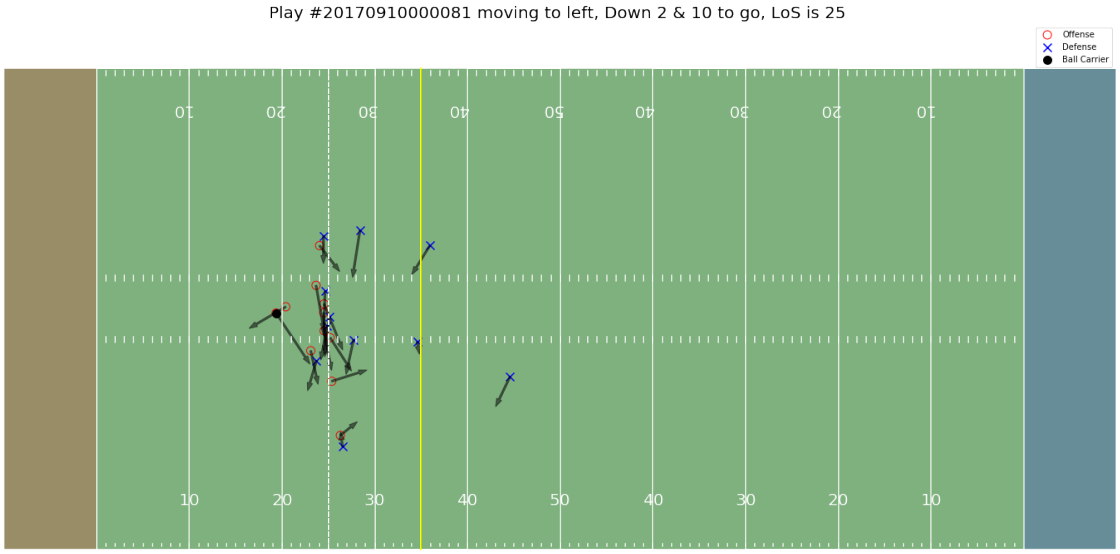

Figure 2: Player positions and velocities for a sample play. Corresponding Video.

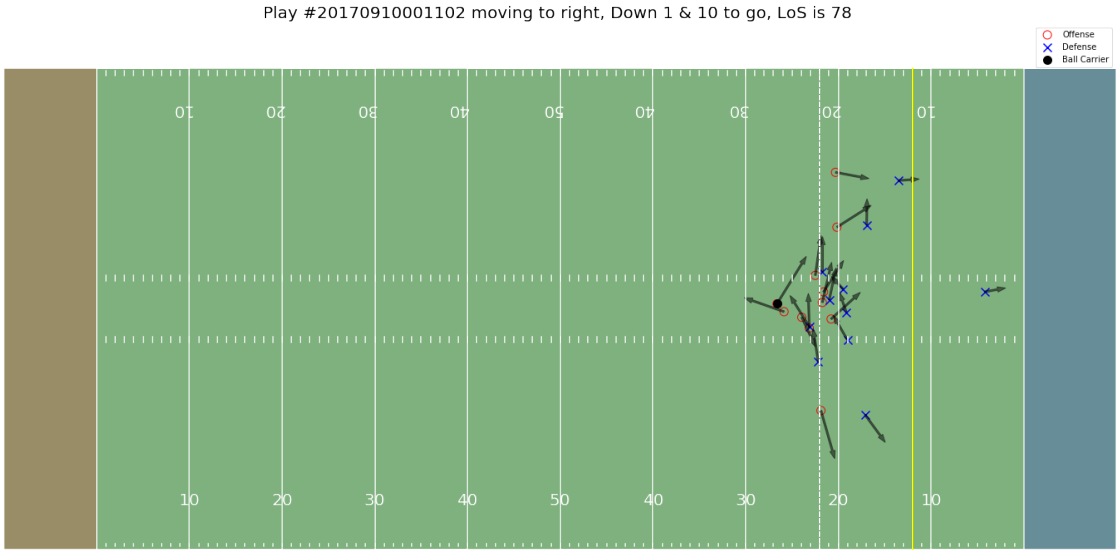

Figure 3: Player positions and velocities for a sample play. Corresponding Video.

In addition to basic data scrubbing and normalization, there are a number of features unique to the game of football that are particularly informative, such as the positions that players on either side of the ball are classified as. In particular, personnel groupings are extracted from the raw data, i.e. the number of defensive lineman (DL), linebackers (LB), defensive backs (DB) for the defensive unit and quarterback (QB), running back (RB), wide receiver (WR), tight end (TE), offensive lineman (OL) for the offensive unit. Additionally, categorical features such as the offensive formation, e.g. Shotgun, Singleback, I-Form, etc., are cast to a one-hot encoding while other insignificant features such as player number or humidity are dropped. The final data set contains categorical features for quarter, down, position groups, if the team on offense is at home and normalized numerical features for time remaining and length of field remaining. Non-normalized features include the distance to the first-down marker, score differential, and the player coordinates on the field.

2.2 Data Augmentation

Note that since the yards gained are recorded as integers, there are 199 distinct values that can be obtained, ranging from -99 yards to +99 yards. Moreover, since the official judgement on yardage gained is an eyeball measurement by the on-field referees, there is undoubtedly noise in this measurement. We have assumed that the official recorded distance is accurate to ± 1 yard and due to the integer nature of the outcome, we augment the data by creating a triplicate of each observation with different yardage outcomes corresponding to the Yards Gained - 1, Yards Gained, and Yards Gained + 1. This has a regularizing effect on the model as it effectively forces the model to learn continuity of the CDF, which is a reasonable goal. This regularizing effect is particularly important for the models we will construct as there are multiple factors after the hand-off that we are unable to account for, e.g. broken or missed tackles, which have great impact on the final yardage gained and influence the underlying distribution but that we are unable to observe. It should be noted that to prevent possible data leakage, data should be augmented only after it is segmented into appropriate train/validation/test splits.

3 Methods

3.1 Measuring Field Control

Field control, especially at the line of scrimmage, is essential to a successful ground game in the NFL. To measure this, we construct spatial control fields based upon the idea of Player Influence Areas from [3]. For each player, we wish to quantify the degree of control they have over any point on the field. We can then sum these values for every player on the same team or subtract players from the other team to determine what team controls which parts of the field. Our method for computing these fields for a given play is as follows:

1. For each player, create a covariance matrix that is skewed in the direction of their velocity, and create a mean vector that is their position shifted slightly by their velocity vector
2. Using this mean and covariance, evaluate a multivariate Gaussian PDF on a grid of points representing the entire football field
3. After these densities are obtained for each player, we are interested in the following fields: (a) sum all of the defensive players, (b) sum all of the offensive players, (c) just the ball carrier, and (d) all offense minus all defense, i.e. $\mathrm{SC} = \sigma\left(\alpha\left(\sum_{i\in\mathcal{I}_o} I(p_i) - \sum_{i\in\mathcal{I}_o} I(p_j)\right)\right)$ where $\mathcal{I}_o$ is the set of offensive players, $\mathcal{I}_d$ is the set of defensive players, $\sigma(\cdot)$ is the sigmoid function, α is a scaling factor, and $I(p_\ell)$ is the "influence" of player ℓ
4. Normalize each field and convert to uint8 to save memory and save as a 100x200x4 tensor

To demonstrate this method, we consider the two plays from earlier, Play ID 20170910000081 and Play ID 20170910001102. For the fields presented in Figures 4 and 5, the image resolution is 200x100 pixels and the scaling parameter $\alpha = 1$. On the first 3 fields, the values range from 0 (no control) to 127 (maximum control). On the aggregate plot, the values range from 0 (maximum defensive control) to 127 (maximum

offensive control). In most places on the field, neither team has significant control, so their value is 63 and they are colored white.

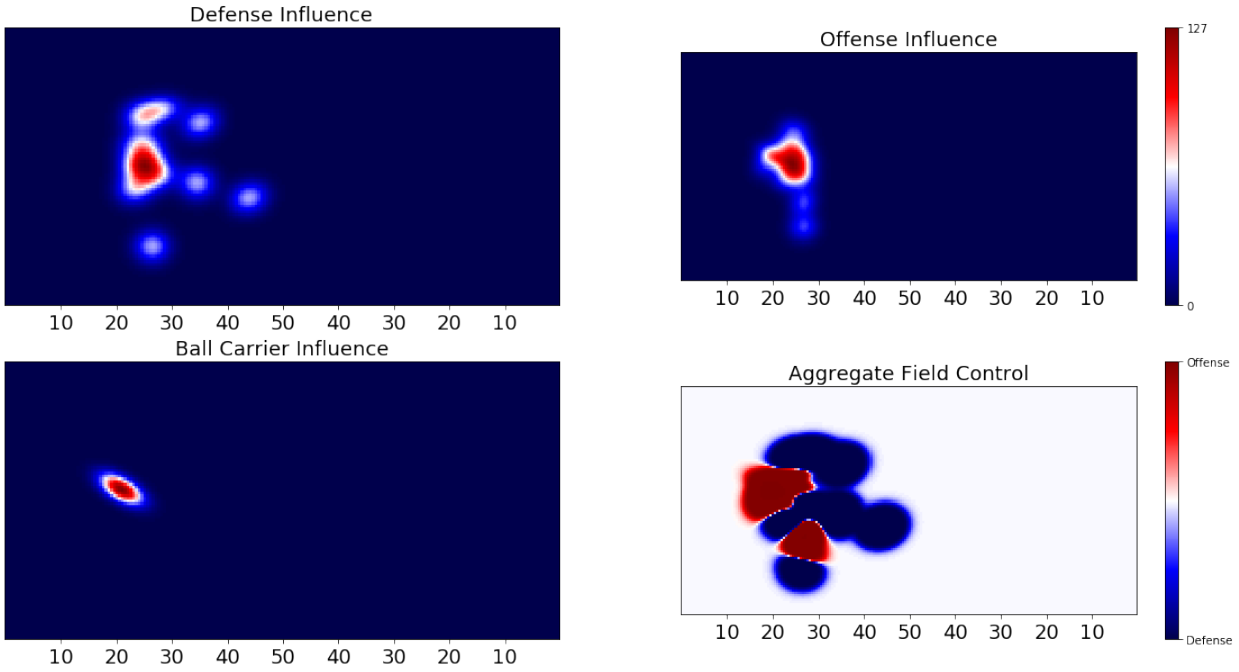

Figure 4: Field control for Play ID: 20170910000081

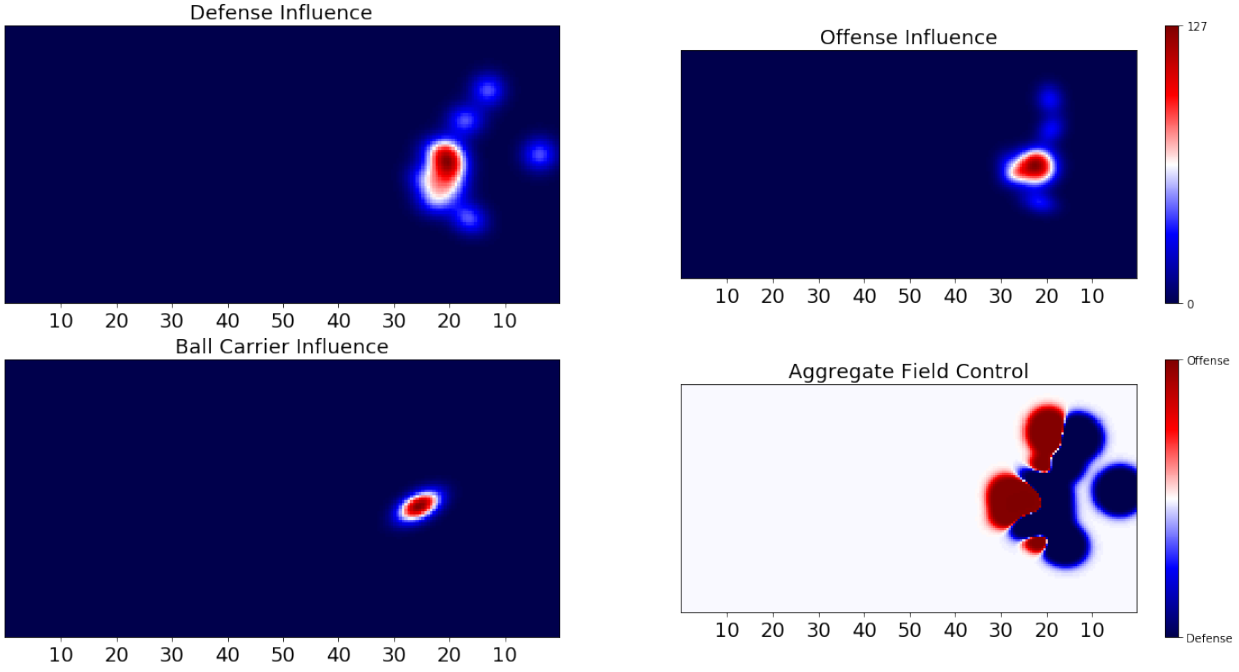

Figure 5: Field control for Play ID: 20170910001102

From these examples, we can see that the positions of the players' areas of control match up with their true positions. This is particularly noticeable for the defensive backs, who are isolated from other players and

are not moving very quickly. We can also see that the densities as skewing in the correct directions. This is readily apparent by looking at the ball carriers, whose areas of influence are skewed in the direction shown in the raw data. The aggregate fields of control are also reasonable, as in both these plays, the defense has players behind the line of scrimmage, which correspond to the juts of blue pointing to the left in both cases.

3.2 Evaluation Metric & Model Architectures

Instead of generating a point estimate for the predicted yards gained, we wish to model the distribution of predicted yards to be gained on a play. To measure the convergence of the CDF to the true outcomes, we implement the Continuous Ranked Probability Score:

$$\text{CRPS} = \frac{1}{199N} \sum_{m=1}^{N} \sum_{n=-99}^{99} \left(P(y \leq n) - H(n - Y_m)\right)^2$$

where $H(\cdot)$ is the Heaviside step function and Y_m is the actual yardage gained on the play. In our implementation, the final layer is a dense layer with 199 outputs and a softmax activation which is then cumulatively summed and normalized in the loss function to obtain the CRPS. This allows us to interpret the output of our model as the discrete probability mass function of yards gained and train on convergence of the CDFs.

3.2.1 Multilayer Perceptron

As an initial model, we train a multilayer perceptron (MLP) on our data. In total, there are 81 features in the data set and we use a fully connected model architecture with layers of 128 neurons, 256 neurons, and an output layer of 199 neurons. The activation function for the two dense layers is a ReLU, while a softmax activation function is used for the final output layer. Batch normalization and 50% dropout are incorporated between each of the dense layers. The data is split into train/validation/testing splits of 70%/15%/15%. We expect for this model to perform poorly as it will not be able to adequately parse the spatial information provided by the player positions.

3.2.2 Convolutional Neural Network

Since spatial information is of the utmost importance in the context of determining spatial control, we implement a convolutional neural network (CNN) to operate on the four 100x200 spatial control images that we generate. The architecture is presented in Figure 6 and is a 5x5 convolutional layers with 16 filters, followed by a 2x2 max-pooling layer, a 3x3 convolutional layer with 32 filters, a 2x2 max-pool layer, a 3x3 convolutional layer with 64 filters, a 2x2 max-pool layer, flattened to a vector and passed through a fully connected layer with 1024 units, a fully connected layer with 128 units, a fully connected layer with 256 units, and a final fully connected output layer with 199 units. All layers use a ReLU activation function except the final fully connected layer, which uses a softmax activation function to be interpreted as our discrete PDF. Moreover, every convolutional layer utilizes "same" padding. Note that batch normalization is implemented between every layer and 50% dropout is added between every fully connected layer. For this model, the data is split into train/validation/testing sets of 70%/15%/15% of the total number of observations.

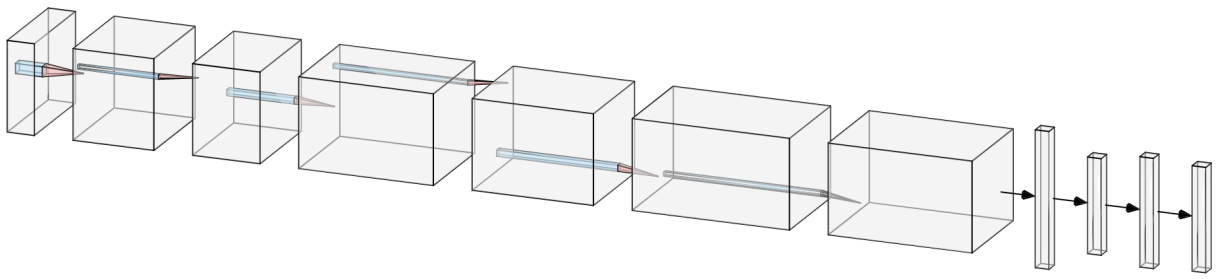

Figure 6: Convolutional Neural Network Model Architecture (image generated using [4]).

3.2.3 Mixed-Data Model

Ideally, a model would have access to both game state information, such as the game time remaining and score differential that will influence the playcall as well as the spatial control of the field at the time of hand-off. We hypothesize that the multilayer perceptron model is adept at parsing numerical information pertaining to play design, while the CNN model excels at extracting spatial information regarding the execution of the play. To do so, we propose the architecture diagrammed in Figure 7 to combine modified versions of the two models. In this model, the game state vector is only 37 dimensional, since it does not include spatial information. The CNN is also modified to utilize three fully connected layers after flattening the convolutional filters, with 1024, 128, and 64 units, respectively.

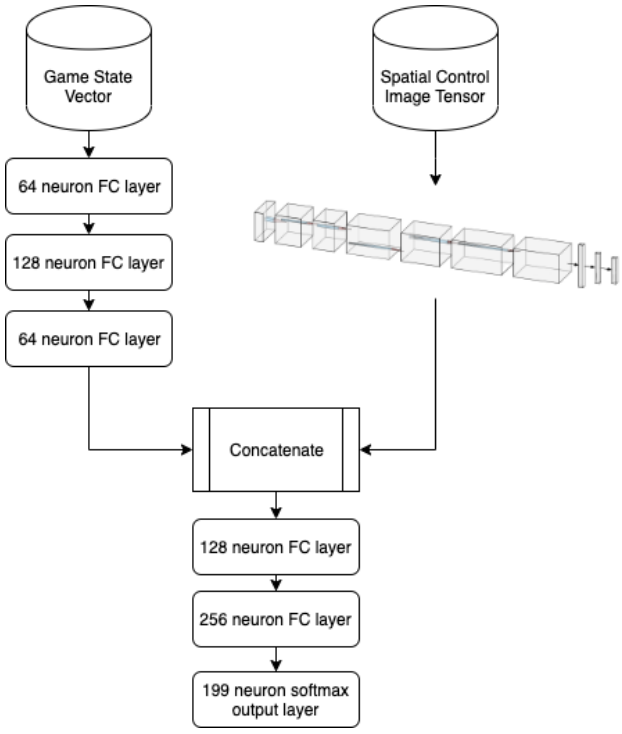

Figure 7: Mixed-Data Model Architecture.

Once again batch normalization and 50% dropout are included between all of the fully-connected layers and ReLU activation functions are used for all but the final softmax output layer. The data is split into train/validation/testing sets of 70%/15%/15% of the total number of observations.

3.3 Training Methods

3.3.1 Data Generators

Since we are generating 100x200x4 images for each of our data points, which are tripled for training, it is inefficient to load all of the data into memory for the CNN and mixed-data models. To fix this, we wrote our own data generator, which allowed the models to train, evaluate, and predict on data loaded into memory in batches. Each game state, field control image, and yards gained value was saved into its own file and was loaded into randomly generated batches during training. The batches were shuffled at the start of each epoch to avoid overfitting.

3.3.2 Learning Rate Reduction

To ensure that the model trains as efficiently as possible, we added a callback that detects plateaus in validation loss during training and reduces the learning rate of the optimizer after some number of epochs

with no improvement. This allows the optimizer to hopefully avoid passing over minima during the training steps. In our training, we set the learning rate to halve after going 10 epochs with no improvement in validation loss. Another possible method that could have been paired with this would be to change the batch size as the training progressed, as larger batches incur larger gradient updates, which are correlated with the learning rate.

3.3.3 Early Stopping

To avoid wasting walltime during training, we also incorporated early stopping into our training process. This lets us end the training before the prescribed training epochs are completed if there is no improvement in validation loss after some number of epochs. In our case, we force the training to stop after 21 epochs with no improvement. This number is chosen carefully, in combination with the epoch requirement for learning rate reduction. With this scheme, the training is allowed to reduce step sizes twice before resorting to an early stop. This avoids a situation where a reduced step size could improve the validation loss, but the early stopping prevents the training from proceeding.

3.3.4 Train/Validation/Test Splits

As aforementioned, we utilize a 70/15/15 split for training/validation/testing data. We visualize the distributions of our various sets in Figure 8. We observe that the three distributions appear to be similar. Furthermore, the augmented training set has indeed had a smoothing effect on the data, which we believe will have a positive regularizing effect on training. We note that this may smooth some inherent inconsistencies in the data, such as the dip in probability at exactly 10 yards due to referees penchant for spotting either a yard shy or a yard long when the distance to gain is exactly 10 yards.

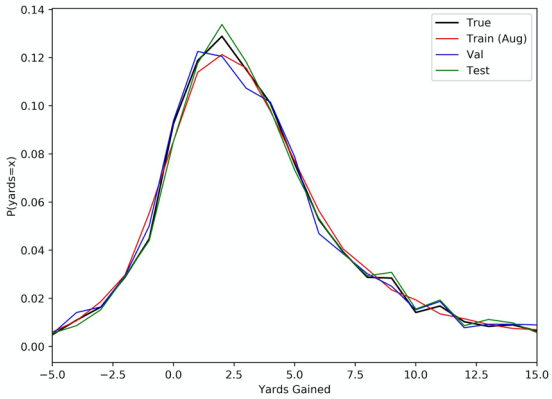

Figure 8: PDF of various data splits.

4 Results

All three models were trained for 100 epochs and in each case, and the model with the best validation CRPS value was saved. Since the CNN and mixed-data models are relatively large, their training was done on an HPC with 16 processor cores. We found that 8 workers along with a batch size of 64 allowed to reasonable training times for each epoch. Results on both validation and tests sets are reported in Table 1.

Table 1: CRPS of various models for validation and test sets.

Model	Validation CRPS	Test CRPS	Epoch Train Time
MLP	0.0138	0.0136	< 1 second
CNN	**0.0094**	0.0132	≈ 15 minutes
Mixed-Data	0.0098	**0.0100**	≈ 15 minutes

Surprisingly, we find the CNN has the best out-of-sample CRPS value during training. This may be due to the fact that the game-state vector adds too many degrees of freedom to the model and that these degrees of freedom are not particularly informative. Therefore, the mixed-data model has a much larger search space for relatively little gain, which results in less efficient training than the CNN alone. Notice also that the MLP model performs very poorly compared to the others. This is likely because many of the features in the game state are not informative relative to spatial features. The combination of generating spatial control fields and using convolutional layers allows for highly complex and correlated structures to be gleaned from the relatively sparse spatial information provided in the data. However, the mixed-data model's performance on unseen test data in comparison to the CNN seems to suggest that perhaps this has prevented the model from over-fitting and that game-state information may help the model relate new plays to previously seen plays. The MLP model does not provide the spatial features enough structure to extract meaningful relationships between the data on the same level as the other models considered. The formation of relationships are also hindered by the influence of the other features, whereas in the mixed-data model, the spatial features only interact with each other for a significant portion of the model.

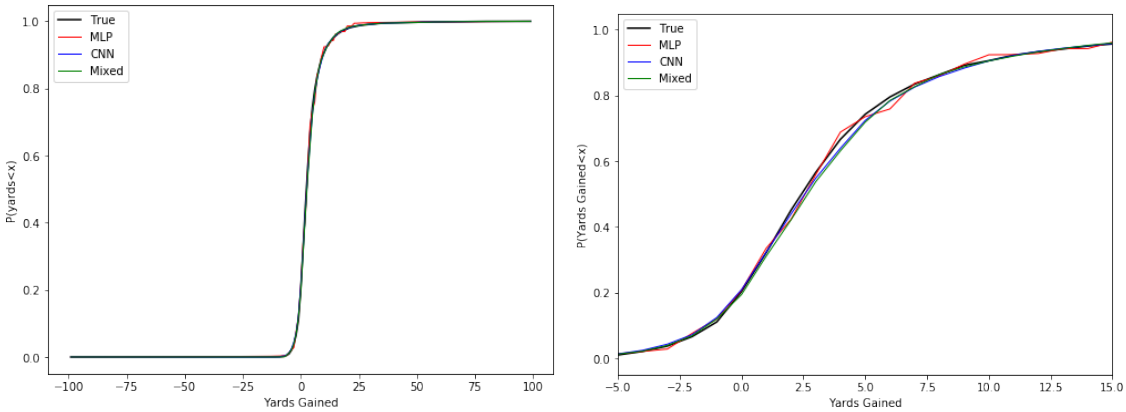

Figure 9: Comparison of True and predicted CDFs.

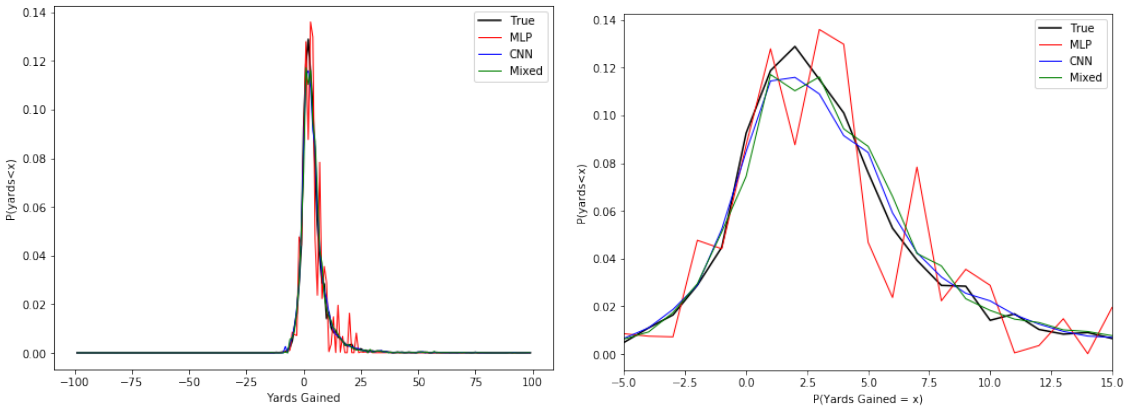

Figure 10: Comparison of True and predicted PDFs.

4.1 Rusher Performance Above Expectation

A salient use-case for a predictive model lies in the realm of player evaluation. We apply our model to evaluate the performance of every rusher in the 2017 and 2018 seasons with a minimum of 20 attempts, a cutoff that corresponds to roughly one full game as a starter. A scatter plot of all rusher's actual mean performance relative to the mean of our CNN model prediction for yards gained on those plays is presented in Figure 11.

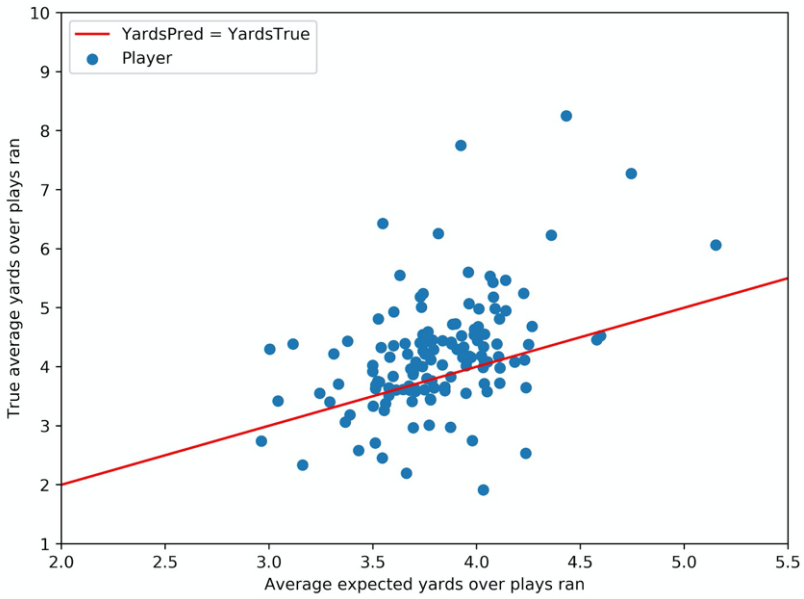

Figure 11: Scatter plot of eligible rushers actual and expected performance.

From this plot, we find that most rushers performed close to their expectation and that the bulk of rushers are expected to gain between 3 and 4 yards. Moreover, there are significantly more positive performing outliers than there are rushers who perform poorly. The best and worst 5 rushers are presented in Table 2.

Rank	Player Name	Yards Above Expectation
1	Brandon Bolden	3.83
2	Robert Woods	3.82
3	Cordarrelle Patterson	2.88
4	Raheem Mostert	2.53
5	Brian Hill	2.44
131	Branden Oliver	-1.09
132	Andre Ellington	-1.23
133	Paul Perkins	-1.47
134	Chris Johnson	-1.70
135	Ronald Jones	-2.12

Table 2: Top and Bottom 5 Rushers Above Expectation

We note that the players who tremendously out-performed are not running backs, rather they are wide receivers whose yardage on hand-off plays mainly comes from jet sweep style plays. This intuits as these plays have a screen-like effect that sucks the defense to the line of scrimmage. Our model assigns a large

amount of control to the defense, which depresses the predicted yardage value. However, if the ball carrier is quick enough or able to evade a tackle or two, they will quickly make it to the second-level of the defense and gain many yards on a single play. As for the rushers at the bottom of the list, we mainly find running backs that are now out of the league, such as a Chris Johnson playing shortly before his retirement. This gives us some assurance that our model is appropriate for player evaluation as it appears to align with NFL coaches judgement.

5 Discussion

5.1 Strengths

One of the biggest strengths in our model is its reliance almost solely on spatial data. From our results, the CNN preformed the best on out-of-sample data, and this model only operates on spatial data. This may allow data collectors for the NFL to allocate more resources to improving the quantity and quality or kinematic and positional data gathered from players. The reliance on spatial control fields can also allow for additional insight into the factors that influence play success. For example, guided backpropagation or deconvolution may allow one to determine spatial patterns that are correlated with play success, which can then be incorporated into NFL playcalling.

Another important strength of our model is interpretability. One of the main reasons for our design of a softmax output combined with the CRPS loss function is that it allows the model to easily output both PDFs and CDFs of yards gained. The 199 softmax outputs can be interpreted as a discrete PDF of yards gained, which simply cumulatively summing these outputs yields that CDF of the same quantity. With this interpretation in mind, one can compute meaningful quantities of interest such as expected yard gained, median yards gains, variance, skewness, or even individual players' performance relative to expected yards gained for equivalent plays.

Moreover, we demonstrated the aptitude of this predictive model for the purpose of player talent evaluation by ranking running backs that consistently perform above expectation. Results from this study can indicate that a back is able to break tackles at the line of scrimmage and get to the second-level of the defense, resulting in a more efficient and effective running game or that there are specific play times that out-perform given the offense's position.

5.2 Weaknesses

It is worth noting that our evaluation metric measures convergence in the CDF, not the PDF. It is possible for the model to generate CDFs with low CRPS that have PDFs with very undesirable behavior, such as oscillatory tails and peaks. Thus, we may not be able to accurately model the tails of the distribution, such as unexpected long runs that result from missed tackles. This inability to accurately model runs that can reach the second layer of the defense may mean that our model is too heavily biased towards the bulk of runs in the -2 to 3 yard ranges. It is possible that more heavily weighting data with longer runs via more data augmentation could alleviate this problem.

Another weakness of our model is its reliance on the control fields, which can be computationally expensive to generate for many data points. In our case, we were only able to generate around 1000 fields a minute. This could be sped up by generating the images for different plays in parallel, but, to the authors' knowledge, the underlying operations in calculating the Gaussian densities cannot be significantly sped up. The reliance on spatial control fields also makes the model quite large, which would affect real-world applicability.

5.3 Further Work

There are many potential avenues to extend this work. One such area lies in the domain of feature engineering, where more football-specific features that may better inform offensive and defensive strategies, such as

defining "garbage time", when team strategies greatly change from "normal" game-flow [1].

An interesting extension of this project would be to create a tool that would allow coaches to move player positions and compute the effect on predicted yards gained. Moreover, the predictive capabilities of this model could be leveraged to improve other in-game decisions, such as audible calling through the construction of certifiable optimal rule lists for altering game state variables, such as offensive formation [2].

References

[1] M. Clay, *Defining 'garbage time'*, 2012. [Online]. Available: https://www.pff.com/news/defining-garbage-time.

[2] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin, "Learning certifiably optimal rule lists," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2017, pp. 35–44.

[3] J. Fernandez, "Wide open spaces: A statistical technique for measuring space creation in professional soccer," in *Sloan Sports Analytics Conference*, 2018.

[4] A. LeNail, "Nn-svg: Publication-ready neural network architecture schematics," *Journal of Open Source Software*, 2019.

[5] NextGenStats, *Next gen stats nfl big data bowl dataset*, Note: data retrieved from Kaggle https://www.kaggle.com/c/nfl-big-data-bowl-2020/data, 2019.

Appendix A: Detailed Data Description

Table 3: Raw Features and Descriptions

Feature	Description
GameId	a unique game identifier
PlayId	a unique play identifier
Team	home or away
X	player position along the long axis of the field.
Y	player position along the short axis of the field.
S	speed in yards/second
A	acceleration in yards/second^2
Dis	distance traveled from prior time point in yards
Orientation	orientation of player (deg)
Dir	angle of player motion (deg)
NflId	a unique identifier of the player
DisplayName	player's name
JerseyNumber	jersey number
Season	year of the season
YardLine	the yard line of the line of scrimmage
Quarter	game quarter (1-4, OT=5)
GameClock	time on the game clock
PossessionTeam	team with possession
Down	the down (1-4)
Distance	yards needed for a first down
FieldPosition	which side of the field the play is happening on
HomeScoreBeforePlay	home team score before play started
VisitorScoreBeforePlay	visitor team score before play started
NflIdRusher	the NflId of the rushing player
OffenseFormation	offense formation
OffensePersonnel	offensive team positional grouping
DefendersInTheBox	number of defenders lined up near the line of scrimmage
DefensePersonnel	defensive team positional grouping
PlayDirection	direction the play is headed
TimeHandoff	UTC time of the handoff
TimeSnap	UTC time of the snap
Yards	the yardage gained on the play (you are predicting this)
PlayerHeight	player height (ft-in)
PlayerWeight	player weight (lbs)
PlayerBirthDate	birth date (mm/dd/yyyy)
PlayerCollegeName	where the player attended college
Position	the player's position (the specific role on the field that they typically play)
HomeTeamAbbr	home team abbreviation
VisitorTeamAbbr	visitor team abbreviation
Week	week into the season
Stadium	stadium where the game is being played
Location	city where the game is being played
StadiumType	description of the stadium environment
Turf	description of the field surface
GameWeather	description of the game weather
Temperature	temperature (deg F)
Humidity	humidity
WindSpeed	wind speed in miles/hour
WindDirection	wind direction