Learning Feature Representations from Football Tracking

Michael Horton[1]
Sportlogiq
Michael.horton@sportlogiq.com

1 Introduction

In this paper, we present a flexible neural network framework that accepts as input the raw trajectory data produced by the player tracking systems deployed in many professional sports and learns an internal representation of the individual and coordinated movement of all involved players. The framework eliminates the necessity to manually engineer features, deals with trajectories of variable time duration, and does not require an ordering of players in the input.

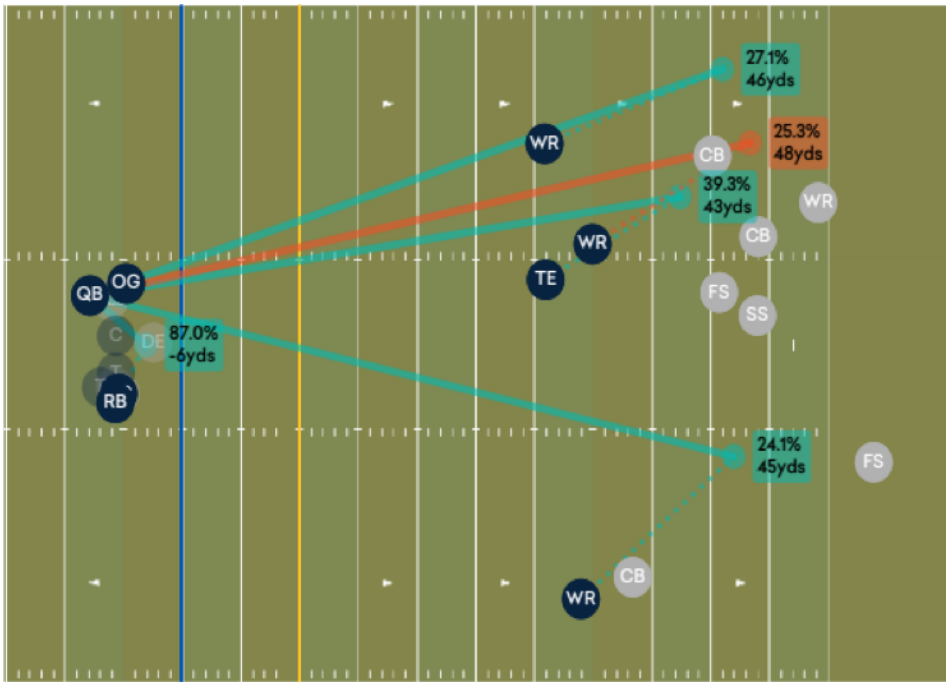
Player tracking systems, introduced in many elite sports competitions in recent years, provide rich datasets for analyzing player and team performance. A de-facto format for the data generated by such systems has emerged, where player movements are captured as *trajectories*—a sequence of time-stamped (x, y)-coordinates—and the game state is recorded in *event logs*—a sequence of time-stamped player actions and match events.

Concurrently, machine learning methods have been applied to address many sports analytic questions. These methods, and in particular deep neural networks, have been shown to be effective over a broad range of problems, and often require less effort to develop than bespoke algorithms for the same task. The methods place some constraints on the structure of the input they accept, typically requiring that each input is a predefined shape, and that the values within the inputs are consistently ordered.

However, there is a fundamental mismatch between the data generated by player tracking systems and the implied structure of inputs to standard neural network models. Tracking data is fundamentally temporal, and the interval between events is also variable, resulting in variability in the temporal dimension of inputs, henceforth referred to as the *variable duration problem*. Furthermore, teams make personnel decisions resulting in differing distributions of player roles between, or within, games. In some cases, even the number of players may vary, such as in power-plays in hockey or red-card situations in soccer. There is thus no intrinsic ordering of players in a given interval of play that persists from interval to interval, or game to game. This is the *player-ordering problem.*

The variable duration and player ordering problems are intrinsic to tracking data, and they are inconsistent with the input structure requirements of many standard machine learning models, and to date, there is no canonical neural network architecture that accepts raw tracking data as input. This mismatch is typically circumvented by introducing a preprocessing step where the raw tracking data is

[1] *The author would like to thank Luke Bornn for his advice and the suggestions he made during the preparation of this paper.*



transformed into structured feature representations designed for the task at hand. For example, the variable duration problem has previously been dealt with by considering player locations at only a fixed number of times [1] [2] or through neural network architectures that support variable dimensionality of inputs, such as convolutions and adaptive pooling [3] or recurrent neural network models [4] [5] [6].

Figure 1. Illustration of predictions made when a pass is thrown. The solid lines show the predicted trajectories of the ball to the expected reception locations for each of the eligible receivers. The annotation for each pass shows the probability the pass will be successful, and the expected air-yards gained. The trajectory for the targeted receiver is shown in orange. A visualization of the prediction output is available here https://youtu.be/A9khHSAOXI8.

The player ordering problem has been addressed by a variety of techniques , such as treating all players as a *bag-of-words* [7], learning a role-based ordering [4], ordering by proximity to an anchor point like the ball [1] [8], or by randomly permuting the order [9].

However, the drawbacks of having to construct feature representations are obvious: It is time-consuming to design, implement and process the structured feature representation; and the feature representation will necessarily omit some of the information available in the raw tracking. The framework presented in this paper obviates the need for such preprocessing by directly addressing the variable duration and player ordering issues, and by learning an internal feature representation directly from the raw tracking data.

In particular, we deal with the variable duration problem by using convolutions and adaptive pooling [3], and the player ordering problem is addressed by using the Deep Sets network architecture [10] that treats

the input players as an unordered set, and learns the same internal representation of the play **regardless of the order of the player trajectories in each example.**

We evaluate the framework by implementing neural network models for several *prediction problems*, where the task is to determine outcome of a future event, based only on the information available up to the current moment. For example: At the time that the pass is thrown, we may wish to make a prediction whether a pass will be successfully caught; or predict at the time the player first receives the ball which defender is most likely to tackle the ball-carrier, see Figure 1 for an example.

Accurate prediction of future events such as this have several use-cases. For example, in player and team performance analysis the ability to accurately predict the probability of success of a pass allows the skill of the quarterback and catcher in executing the pass to be isolated from the situational context in which the pass was made, such as the play run and the defensive formation. A further use-case is in in-game sports betting, where a key problem for the sports book is to be able to decide when to close the line on an in-game proposition.

The prediction models are experimentally evaluated using the football tracking dataset provided by the NFL for the Big Data Bowl competition in 2019. The dataset contained full tracking data for all the games played in the first six weeks of the 2017 NFL season. The competition had three themes, including the identification of the best receiver-route combinations, and yielded several contributions that are relevant to this work: Chu *et al.* identified routes using a center-based clustering method where candidate centers were Bezier curves [11], Yurko *et al.* used a recurrent neural network to continuously predict the expected yards gained on the play during the play [6], and Deshpande and Evans constructed a feature-based Bayesian model to make predictions about the value of passes to all eligible receivers [8].

In the remainder of this paper, we detail the structure of the input dataset in Section 2. The theoretical basis and architecture of the framework is outlined in Section 3. The experimental model used to evaluate the framework is outlined in Section 4, along with the results.

2 Data

The set-learning framework is designed to accept as input trajectory and event data, without the need for an intermediate step of calculating hand-designed features. The input requirements a sufficiently general to support the standard tracking data sources in a variety of sports, such as NFL's Next Gen Stats, ChyronHego's soccer tracking data, or Sportlogiq's multi-camera hockey tracking data.

Each input example contains three component tensors: player tracking data, player metadata, and game-state metadata; and models a single play, see Figure 2.

Trajectory Tensor The trajectory of a single player is represented as a sequence of time-stamped *frames* where each frame is a vector containing the x- and y-coordinate of the player at that point, and possibly additional information such as the orientation, speed and direction of the player. Frames are captured at uniform intervals, e.g. every tenth of a second, for every player participating in the play.

The player trajectories for a single play i $\in 1 \ldots N$ can thus be represented as a $M_{\text{track}} \times S \times T^{(i)}$ dimensional tensor, where M_{track} is the length of the player trajectory vector, S is the number of players tracked in the play, and $T^{(i)}$ is the number of frames within the play i.

Player Tensor For each player involved in the play, a vector of player attributes is captured, such as their position, weight, height, age, and whether they are on the offensive or defensive team. The information for all players participating in the play is represented as a $M_{\text{player}} \times S$ dimensional tensor, where M_{player} is the length of the player attribute vector.

Play Tensor The context of each play is captured as a vector of play attributes, such as the down, yards to first down, yard-line, game clock and score. This is represented as an M_{play} dimensional tensor, where M_{play} is the length of the play attribute vector.

3 Framework

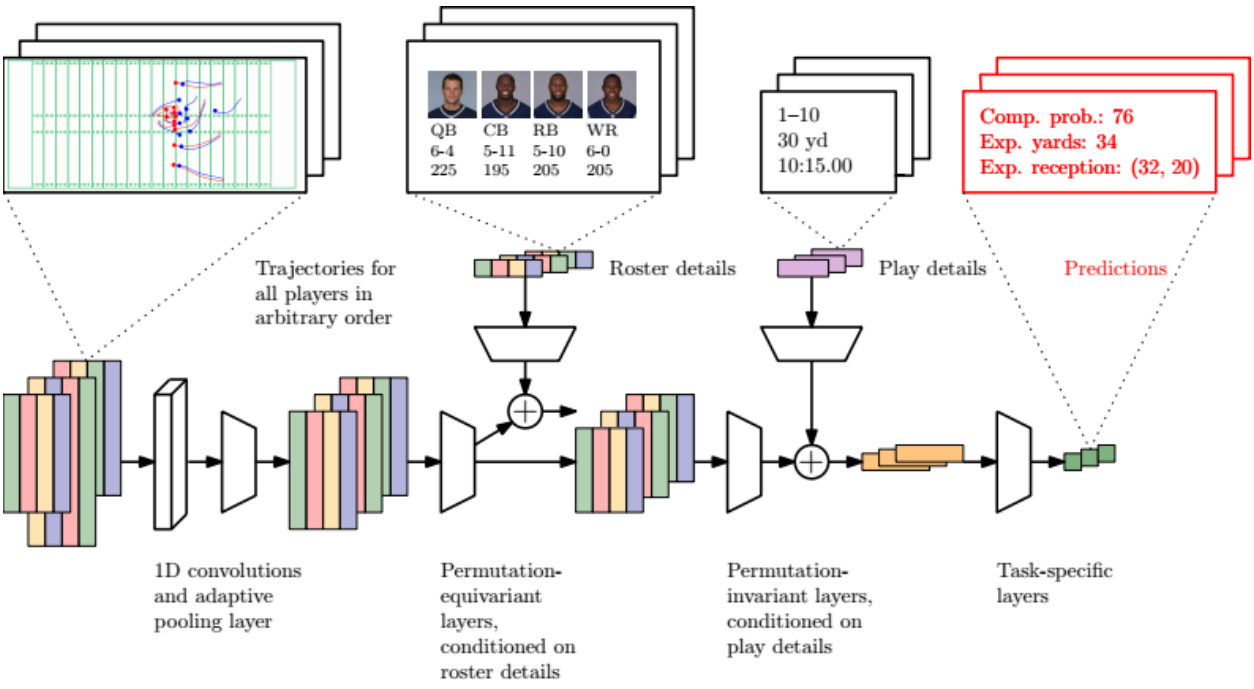


Figure 2. The trajectory learning framework architecture. The model accepts input examples containing: the trajectories of the players—up to the point the prediction is made; the roster details; and the play details.

3.1 Framework Design

The fundamental design objective of the model was that the framework accepts raw trajectory data as input, without the need for an intermediate step of feature engineering. In this section, the design challenges that this requirement presented are described.

3.1.1 Variable Play Lengths

From an analytic perspective, football has a convenient property in that the game is segmented into downs, i.e. short sequences of play that all have a discrete start point—the snap—and where the players are consistently spatially arranged—the line of scrimmage. From this point, most plays unfold in a linear manner with minimal branching of the main events, e.g. snap—pass/handoff—reception—tackle/out-of-bounds/touchdown.

However, the duration of each play is variable, and even for similar plays, the timing of the critical events will vary. This issue manifests itself in the prediction problems we describe in this paper by the variable interval between the time of the snap and the time that the pass is thrown for the pass outcome prediction problems, and between the time of the snap and the tackle/out-of-bounds for the tackle prediction

problems. Thus, for each play example i, described in Section 2, the value of $T^{(i)}$ varies from play to play. There are several design options to deal with this, such as recurrent models such as LTSM, or using convolutional and pooling layers, which is the approach taken here.

3.1.2 Set-based Learning

The crucial design challenge for learning directly from player trajectories is that there is no natural ordering of the players from play to play, such that the same player will always occupy the same rank between plays, as detailed in Section 1. To overcome this issue, we relax the requirement of imposing an ordering along the player dimension of the trajectory and player tensors, and use a neural network architecture designed to accept set-based inputs [10].

The framework is thus robust to personnel choices made in each play: The model can learn from plays where the offense runs two wide-receivers and then make predictions on a play with three wide-receivers, without having to deal with the issue in a rigid ordering, where one of the wide-receivers would need to be aligned with a player of some other position in the input examples.

The set-based learning framework is implemented as neural network layers that have the property of being either *invariant* or *equivariant* to the order along the player dimension. Roughly speaking, a network layer is *invariant* if it instantiates a function that accepts a set of features for each player as input and returns a scalar response that is the same regardless of the order of the players in the input. For example, the pass prediction model yields an estimate of the probability the pass will be successful and will produce the same estimate regardless the position of the receiver in the input.

On the other hand, a network layer is *equivariant* if it accepts a set of feature vectors for each player, and outputs a set of response values, one for each input, that are the same regardless of the order of the input. For example, in the tackle model, we assign a probability of making the tackle to each defensive player, and the assigned probabilities should be the same, regardless of the order of the players in the input.

Formally, let $X \in \mathrm{R}^{M \times S}$ be an input matrix where S is the set dimension, and let $P \in \mathrm{R}^{S \times S}$ be any permutation matrix over S. Consider the following functions: Let $\mathrm{f}_{\mathrm{inv}}: \mathrm{R}^{M \times S} \rightarrow R$ be a function that maps X to a scalar; and $f_{\mathrm{eq}}: \mathrm{R}^{M \times S} \rightarrow \mathrm{R}^{S}$ be a function that maps X to a S-dimensional vector. Then we say that f_{inv} is permutation invariant if

$$f_{\mathrm{inv}}(X \cdot P) \equiv f_{\mathrm{inv}}(X)$$

and f_{eq} is permutation equivariant if

$$f_{\mathrm{eq}}(X \cdot P) \equiv P \cdot f_{\mathrm{eq}}(X)$$

3.1.3 Conditioning

The player trajectory information is the main source of information for the model, but it seems obvious that simple meta-information about the players and the play will be relevant. When trying to predict the probability of a pass being completed, the (relative) heights of the receiver and the defender in coverage

would be significant. Similarly, the down and the yards to first down will influence the decisions made by players.

We thus wish to *condition* the learned representations of player movements with details of the players, and further we want to condition the aggregate behavior of all players during the play with details of the play state. Fortunately, the set-based learning model we use [10] allows for conditioning information to be incorporated into the model in a straightforward fashion, for example, by using vector concatenation or bilinear transformations.

3.2 Framework Implementation

The framework used here is a multi-layer neural network consisting of three layers, see Figure 2.

3.2.1 Trajectory Layers

The trajectory layers operate on the player trajectory tensor and deals with the variable duration of each play. This consists of several interleaved convolutional, pooling and nonlinear layers, followed by an adaptive pooling layer. The layers accept a player trajectory tensors of variable duration dimension and output a tensor with fixed duration dimension T. In aggregate, the trajectory layers encode a function:

$$\mathrm{f}_{\mathrm{traj}}\colon \mathrm{R}^{M_{\mathrm{traj}}\times S\times T^{(i)}} \rightarrow \mathrm{R}^{M'_{\mathrm{traj}}\times S\times T}$$

In this design, the x- and y-coordinates, along with the other features, are treated as separate *channel*s in the initial convolutional layer, and a $1 \times k$ dimension kernel "slides" along each of the 22 player trajectories in the input and detects various features such as spatial location, changes in direction, changes in speed or acceleration. The adaptive pooling layer then standardizes the duration dimension by pooling the output of the convolution to a sequence of the most germane features. The output thus can be considered a fixed-length summary of each player trajectory.

3.2.2 Player Interaction Layers

The transformed fixed-length trajectory representation tensor that is output by the trajectory layers is then conditioned by performing a bilinear transformation with the player metadata tensor, and the result is passed through a series of permutation equivariant layers. These layers are intended to detect high-level actions such as route run, and interactions between subsets of players, such as collisions, coverage and evasive actions.

The output from the player interaction layers is a vector representing the player movements and interactions, one for each player. The player interaction layers thus encode the following permutation equivariant function:

$$f_{\mathrm{player}}\colon \mathrm{R}^{M'_{\mathrm{traj}}\times S\times T} \times \mathrm{R}^{M_{\mathrm{player}}\times S} \rightarrow \mathrm{R}^{M'_{\mathrm{player}}\times S}$$

3.2.3 Scene Interaction Layers

The scene interaction layers are intended to aggregate the set of representation vectors for each player into a single representation vector that describes the play in aggregate. The output tensor from the player interaction layers is aggregated along the player dimension using a permutation-invariant linear transform followed by a non-linear transform. The output from this transform is then conditioned with the play metadata by applying a bilinear transformation, followed by a series of interleaved linear and non-linear transformations. The layers thus implement a permutation invariant function:

$$f_{\text{scene}}: \mathrm{R}^{M'_{\text{player}} \times S} \times \mathrm{R}^{M_{\text{scene}}} \rightarrow \mathrm{R}^{M'_{\text{scene}}}$$

3.2.4 Task-Specific Layers

Finally, layers designed for the particular tasks can be stacked directly on the player interaction or scene interaction layers as required. These will typically be dense layers followed by the required activation function.

4 Experiments

4.3 Experimental Networks

The framework described in Section 3.2 was used to create two neural network models to make predictions about pass receptions and about tackles. Both networks are multi-task, in that they make multiple predictions simultaneously.

The **pass** model makes predictions about the pass reception on passing plays, evaluated at the time that the pass was thrown. The input trajectory tensor contains player trajectory details from 2 seconds prior to the snap until the moment the pass is thrown. The model then makes predictions on the following properties of the reception:

- *Pass completion:* the probability that the pass will be successfully caught by the targeted receiver.
- *Air-yards:* the expected number of yards gained when the catch is made.
- *Reception location:* the (x, y)-coordinates of the expected location where the pass is received.

See Figure 1 for an illustration of the predictions made when the pass is thrown.

The **tackle** model makes predictions about the tackle and is evaluated at the *established play direction* (EPD) time, which is moment that the receiver catches the pass on passing plays, or the moment that the ball-carrier cuts up-field on rushing plays. The model accepts all player trajectories from 2 seconds prior to the snap until the EPD time, and makes the following predictions about the tackle:

- *Likely tackler:* determine the probability distribution over the eleven defensive players that each player will be the first to attempt a tackle on the ball-carrier.

- *Yards until tackle:* the expected yards gained implied by the location of the first tackle attempt or where the ball-carrier runs out of bounds.

- *Tackle location:* the (x, y)-coordinates of the expected location where the first tackle is attempted, or the ball-carrier runs out of bounds.

The model makes predictions about the first opportunity for a defensive player to make a tackle, which is distinct from the point where the tackle is actually made for the play. The identity of the player, and the location of this tackle opportunity are determined using a simple criterion of the first point that a defender was within 2 yards of the ball-carrier and was not blocked.

4.4 Evaluation Dataset

The models were evaluated on the NFL dataset provided for the 2019 Big Data Bowl[2]. This dataset contains the tracking and event information from the games in the first six weeks of the 2017 NFL regular season. The dataset contained 6,963 eligible passing plays and 11,558 eligible passing and running plays. The examples were randomly partitioned into training and evaluation sets with an 4:1 ratio. The number of examples is relatively small for the task, and the experience of training these models suggested that additional data would improve model performance, however the models showed encouraging results in these tasks.

4.5 Evaluation Method

The pass and tackle neural networks were implemented using the PyTorch deep learning framework. The loss function for the classification tasks (pass completion and likely tackler) was binary cross-entropy, and for the remaining regression tasks mean-squared error was used. Hyper parameter selection was carried out by evaluating the models on the evaluation dataset, using accuracy as the criteria for the classification tasks and mean-squared error for the regression tasks.

The networks were bench-marked against existing model implementations that accept hand-designed features are input.

4.6 Results

Experimental results for the model presented in this paper—henceforth called the *trajectory learning model*—showed that the trajectory learning models generally outperformed existing models that rely hand-crafted features, demonstrating the ability of the framework to learn directly from tracking data.

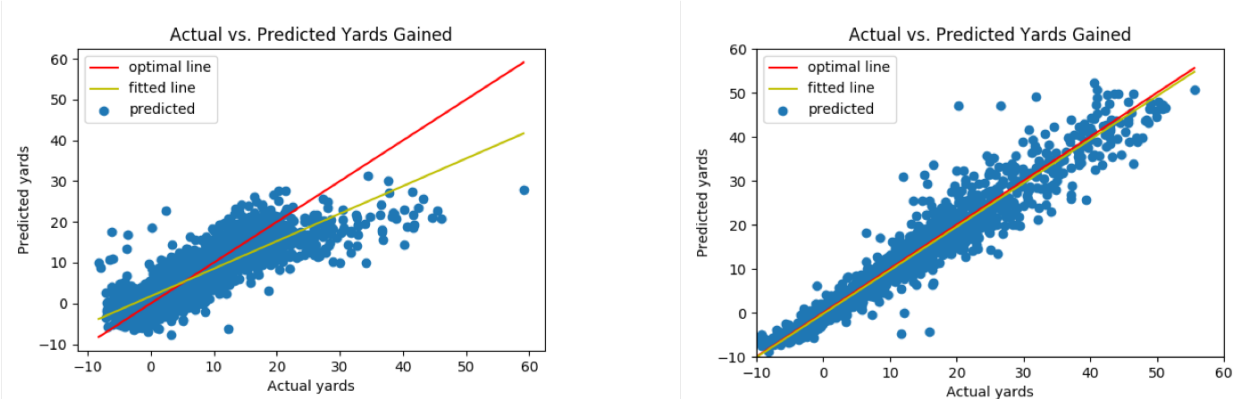
4.6.1 Pass Model

The pass completion classification task obtained an accuracy score of 73.1% on the evaluation dataset and outperformed two models that used the hand-crafted features as input: a logistic regression model and a dense multi-layer perceptron (MLP) model.

[2] https://operations.nfl.com/the-game/big-data-bowl/2019-big-data-bowl/

	Accuracy	Precision	Recall
Logistic Regression	0.694	**0.803**	0.694
MLP	0.722	0.761	**0.826**
Trajectory Learning	**0.731**	0.732	0.672

For the task of predicting the air-yards, the trajectory learning model was able to predict yards gained with a relatively small error. Figure 3(b) is a scatter plot of actual vs. predicted air-yards. The predicted air-yards are highly correlated with the actual air-yards, with a Pearson correlation of 0.977. The model also outperformed the feature based MLP regressor, shown in Figure 3(a).



(a) MLP Model *(b) Trajectory Learning Model*

Figure 3. Scatterplot showing correlation between actual and predicted air-yards.

The trajectory learning model predicts the reception location with high fidelity: The Pearson correlation coefficient between actual and predicted x and y coordinates are 0.94 and 0.89 respectively. Figure 4 shows the predicted locations for a random sample of passes, and pass prediction policy inferred by the model is apparent: the model tends to successfully predict passes in the center of the field. The model is less successful for passes near the end-zone and the edges of the field. Furthermore, the confidence of the model in predicting passes—shown by the area of the dots in the figure—also decreases towards the end-zone and sidelines.

4.6.2 Tackle Model

The tasks that the tackle model is designed to solve are inherently more difficult than those for the passing model, as there is more inherent uncertainty in the tackler and location of the tackler. Regardless, the model shows promising results, and outperformed the benchmark models.

The likely-tackler classification task evaluated the probability of all 11 defensive players being the first to tackle the ball-carrier. On the evaluation dataset, the trajectory learning model correctly chose the first tackler 48% of the time, see the table, below. This model outperformed the benchmark models: a simple rule-based model, and a permutation equivariant neural network that used hand-crafted features.

	Accuracy	Precision	Recall
Logic Model	0.417	0.423	0.385
Set Play	0.461	0.461	0.461
Trajectory Learning	**0.481**	**0.482**	**0.480**

The trajectory learning model was also able to make reasonable predictions on the yards until the tackle, see Figure 5(b). The Pearson correlation between the actual and predicted locations is 0.780, and while the majority of the predictions are close to the actual result, there are two clear groups of outliers: where the actual gain is approximately zero yards, and the predicted gain is much larger; and the converse case where the predicted gain is near zero, and the actual gain is larger. These would suggest cases where the model misclassified the likely tackler, and thus predicted a tackle location that was different from the actual location. Again, the trajectory learning model shows improved performance over the benchmark MLP regressor using hand-crafted features, shown in Figure 5(a). [12]

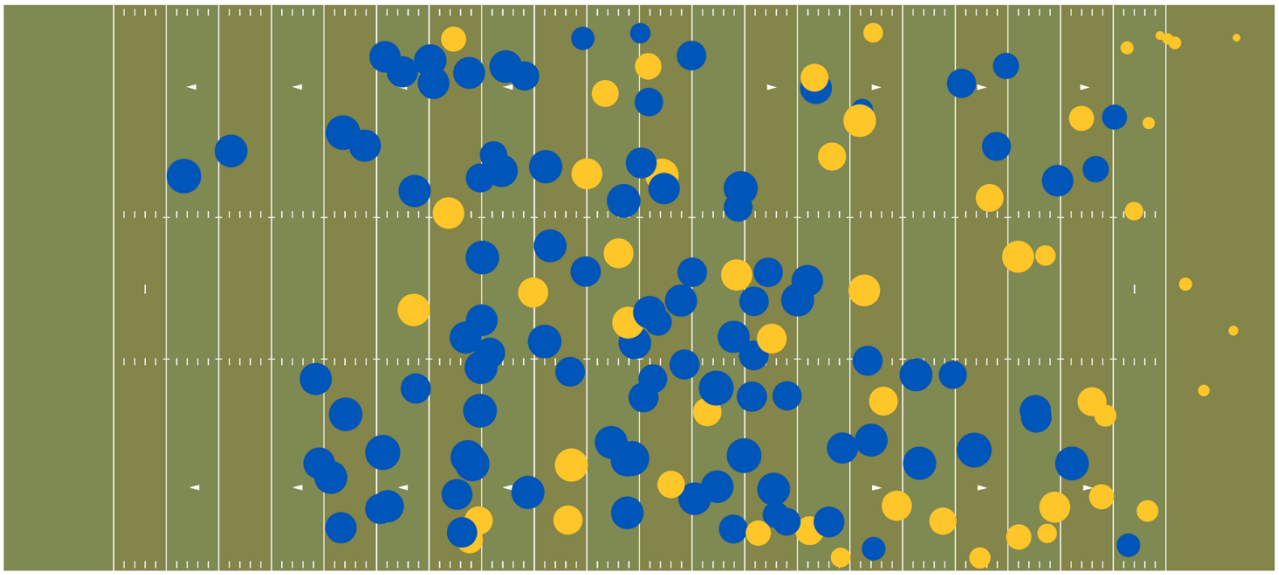
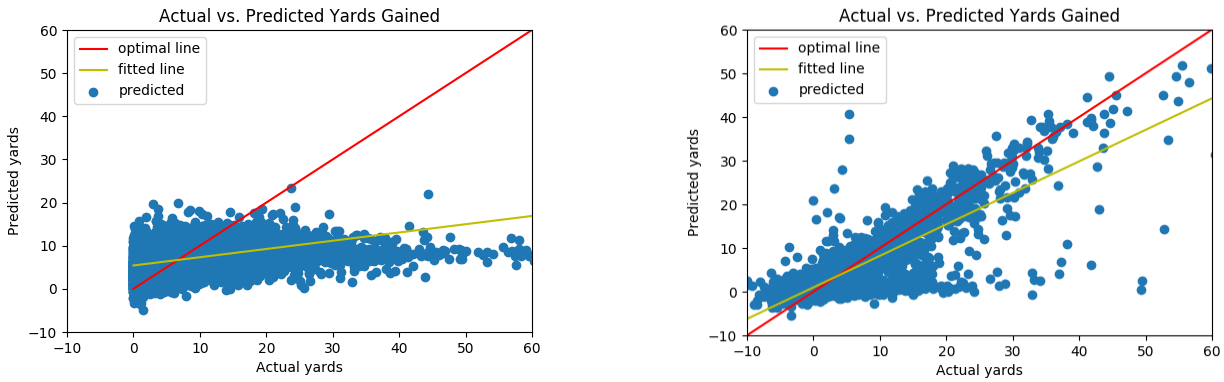


Figure 4. Predicted locations pass receptions for 150 randomly selected plays from evaluation dataset. The offense is playing from left to right in all plays, and the area of the marker is proportional to the evaluated pass completion probability. Blue denotes successful passes and yellow are unsuccessful.



(a) Feature-based Model *(b) Trajectory Learning Model*

Figure 5. Scatterplot of correlation between actual vs. predicted yards to first tackle attempt.

Figure 6 shows examples of the actual and predicted tackle locations for a random sample of plays. In this sample, some of the error modes of the predictions made by the model are apparent, for example, the model will often fail to predict situations where the ball-carrier has a route to the side-line but predicts an earlier tackle further in-field. Moreover, the model performs better at correctly predicting the likely tackler towards the sidelines, where there are fewer candidate tacklers. Predicting the identity and location of the tackler at EPD time is clearly a difficult task, however even in its failed predictions, the model shows some understanding of the dynamics of the game.

5 Conclusion

In this paper we present a flexible neural network framework that accepts as input the raw trajectory data that is the standard output of the player tracking systems used professional sports. Furthermore, we demonstrate the effectiveness of the framework by implementing models for six football prediction tasks.

The framework eliminates the need for typical preprocessing tasks such as feature engineering and trajectory ordering and is able to accept player trajectories of variable duration. The framework consists of set of neural network layers that learn an internal feature representation of the coordinated movement of all players (and the ball). This simplifies the design of models to specifying the higher-level task-specific layers.

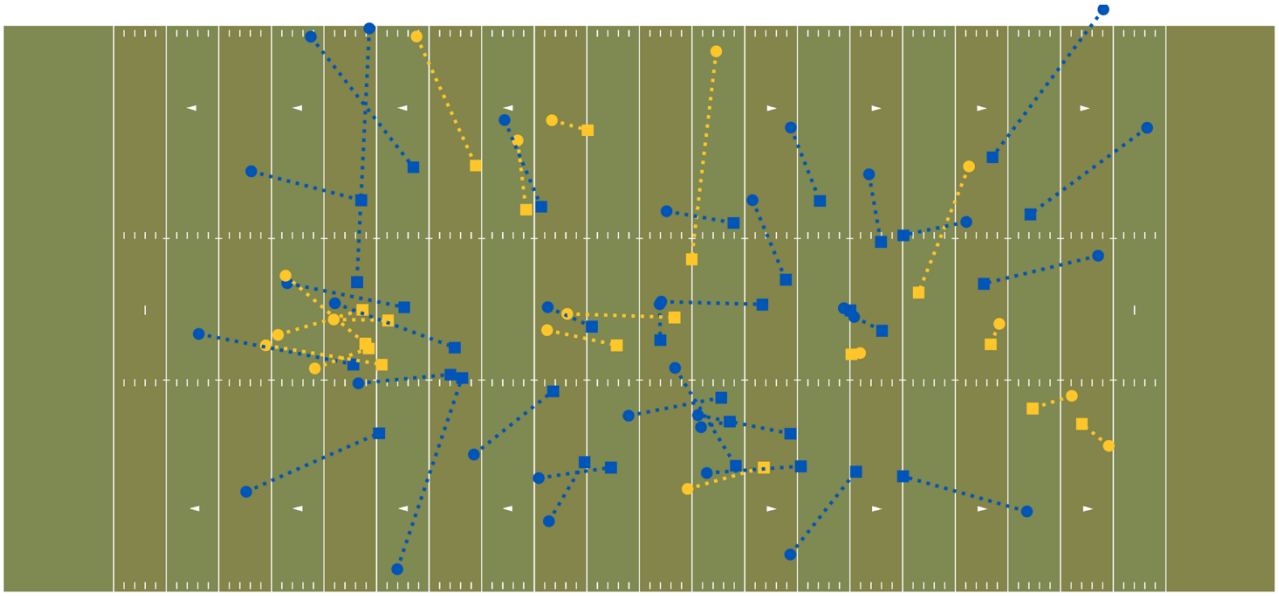


Figure 6. The actual and predicted tackle locations for 50 random plays from the evaluation dataset. The marker for the actual locations is a circle and the predicted locations are marked by a square. Plays where the identity of the tackler was correctly predicted are in blue, and incorrectly identified plays in yellow. In all plays the offense is targeting the right endzone.

References

[1] B. Burke, "DeepQB: Deep Learning with Player Tracking to Quantify Quarterback Decision-Making & Performance," in *13th MIT Sloan Sports Analytics Conference*, 2019.

[2] S. Chawla, J. Estephan, J. Gudmundsson and M. Horton, "Classification of Passes in Football Matches Using Spatiotemporal Data," *ACM Transactions on Spatial Algorithms and Systems,* vol. 3, pp. 1-30, 8 2017.

[3] N. Mehrasa, Y. Zhong, F. Tung, L. Bornn and G. Mori, "Deep Learning of Player Trajectory Representations for Team Activity Analysis," in *11th MIT Sloan Sports Analytics Conference*, 2017.

[4] H. M. Le, P. Carr, Y. Yue and P. Lucey, "Data-driven ghosting using deep imitation learning," in *11th MIT Sloan Sports Analytics Conference*, 2017.

[5] H. M. Le, Y. Yue, P. Carr and P. Lucey, "Coordinated Multi-Agent Imitation Learning," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017.

[6] R. Yurko, F. Matano, L. F. Richardson, N. Granered, T. Pospisil, K. Pelechrinis and S. L. Ventura, "Going Deep: Models for Continuous-Time Within-Play Valuation of Game Outcomes in American Football with Tracking Data," 5 6 2019.

[7] A. C. Miller and L. Bornn, "Possession sketches: Mapping NBA strategies," in *11th MIT Sloan Sports Analytics Conference*, 2017.

[8] S. K. Deshpande and K. Evans, "Expected hypothetical completion probability," *Journal of Quantitative Analysis in Sports,* vol. 0, 11 2019.

[9] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan and I. Matthews, "Large-scale analysis of soccer matches using spatiotemporal tracking data," in *2014 IEEE International Conference on Data Mining*, 2014.

[10] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov and A. J. Smola, "Deep Sets," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017.

[11] D. Chu, M. Reyers, J. Thomson and L. Y. Wu, "Route identification in the National Football League," *Journal of Quantitative Analysis in Sports,* vol. 0, 12 2019.

[12] S. Zheng, Y. Yue and J. Hobbs, "Generating Long-term Trajectories Using Deep Hierarchical Networks," in *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., 2016, pp. 1543-1551.