Reading the Game: Predicting Soccer Defender Movement Using Neural Networks

Ethan Creagar
ethancreagar@gmail.com
Colorado State University Honors Program
Fort Collins, Colorado, USA

ABSTRACT

Perhaps the most important development in 21st century sports analytics has been the popularization of a form of data called tracking or GPS data. With high-tech camera arrays placed strategically around professional sports stadiums, analysts have access to every movement a player makes during a game or match at a rate of 25 frames per second. These data are commonly used for analyzing patterns in play of a player or team, calculating average positions or formations, or identifying specific in-game situations that might be of interest to a coaching staff. However, when it comes to making predictions about teams or players, tracking data's potential is just starting to be tapped. This thesis will discuss experimentation with sizes and structures of neural networks to create an accurate movement prediction model and discuss the feasibility, most desirable features and parameters, and potential of such models.

1 INTRODUCTION

Predicting movement is something that players on a soccer pitch do every moment. For attackers, prediction of a defender's movement or overall tendencies can be crucial when making a split second decision of where to play the ball, whether to make an attacking run, or which direction to dribble to relieve pressure. The skill of a top-class attacker is manipulating defensive players and structures through their understanding of what a defensive player or players would be expected to do in any given situation and reacting accordingly. Using the power of soccer tracking data and neural networks, we can attempt to create a prediction algorithm that acts like a world class attacker, predicting the movement of defensive players based on the on field factors at any given movement.

1.1 Motivation

There are many situations when knowing how defenders in general or a specific defense are likely to react to a situation would be valuable. For example, set pieces and corner kicks are often an area of interest for analysts and coaching staffs, and predicting movement in these dead-ball situations based on player location and ball position might assist in understanding the defensive patterns that teams are most likely to use. Movement prediction might also be useful in quantifying the unpredictability of a player or team, or classifying a general team defensive style. If a player pressures the ball more often than our model expects them to, we might classify them as being more aggressive than expected. This play-style classification might also extend to use in more specific scenarios. We could find instances of a certain passage of play, like a switch from one side of the field to another, and use the same movement prediction analysis to help us understand how a defender or team responds to that situation compared to the expected response that we obtain from our predictions. By analyzing a variety of situations this way, we could start to understand the best ways to create space against a certain team, or why a team is more vulnerable in some situations than others. Play-style classification might also be useful as an extension of Similarity Scores, a popular method of quantifying how similar two players are across teams or leagues used by scouts and analysts at professional clubs. Similarity Scores were first popularized by Bill James in 1994 when he attempted to develop a model which explained how similar two baseball players were by examining both physical traits and play style. This model has been shown to be very good at forecasting a player's future through an extension called PECOTA, developed by Nate Silver in 2003 for Baseball Prospectus [1]. By adding a measure of in-game movement tendencies, we may be able to provide even more accuracy to these scores.

2 REVIEW

2.1 Previous Discussion

Due to the recent popularization of tracking data combined with the recency of the current boom in the use of Neural Networks [2], prediction of player movement in this manner would not have been considered a possibility as recently as 10 years ago. Modern Neural Net location prediction systems in general saw some usage in the early 2000's [3], and by the mid 2010's, Neural Net location prediction was being widely used [4] [5].

Soccer analytics was also an emerging field in the mid 2010's. One of the seminal analytics measures called expected goals, or "xG", was introduced in 2012 by Sam Green while working for Opta Sports [6] (expected goals were studied as early as 2004 [7], but Green's model popularized the method and nomenclature). This metric examined the likelihood of a shot being scored based on location, defender position, and other relevant information. Aside from "expected" statistics like xG, the analytics side of the game has made slow advancements until the late 2010's, when more advanced measures of the game like pitch control [8] and off-ball scoring opportunities [9] rose in popularity.

In the same time frame, professional sports analytics departments began using advanced prediction methods on tracking data, when the Toronto Raptors developed a method for analyzing player decisions that they called "ghosting" [10]. Though the Raptors' method was effective and revolutionary, it would still be four years until Neural Networks were used on tracking data, when a research group from California Institute of Technology and Disney led by Hoang M. Le published their paper on ghosting using Deep Imitation Learning [11]. This paper explored the usefulness of modeling defensive situations using Deep Learning with Long Short Term Memory Netwoks. The researchers attempted to model the average decisions that a defender would make using Imitation Learning

methods, as well as developing specific ghosting models for each team in the premier league. This research would be translated over to the NBA, where Thomas Seidl extended ghosting to predict how a defense would be likely to react to a play drawn in real time by a coach on the sidelines in a product he called "Bhostgusters" [12]. Seidl also pointed out some shortcomings of Le's method, stating that "Although [Le's method] could model the tendencies of specific teams, it didn't take into account the relevant context, such as the score, the fatigue of the players, etc". Seidl used what he called Deep Multi-Agent Imitation Learning, a system comprised of five distinct two-layer LSTM models, one for each "role" or position on the basketball court.

3 EXPERIMENTS

3.1 Formulation

I am interested in exploring three main points through the experiments in this paper:

- Is it possible to predict player movement using Neural Networks? What are the challenges, drawbacks, and benefits of different approaches?
- What data structures, types of models, and model parameters provide the best prediction accuracy? What other types of models might we try with more resources?
- How could we use these results in an analytics environment? What future extensions might help maximize their utility?

In an analytics environment, any questions we choose to explore must be evidently useful and explainable, as any results will need to be explained to non-technical stakeholders such as members of the coaching staff or even players. To this end, we will focus on questions of accuracy, usefulness, and feasibility in the discussion of our models.

To answer these questions, we will first run somewhat simple experiments that evaluate simple performance indicators that our models must satisfy to be considered useful. For example, we must ensure that our predictions are getting the direction of movement correct at least over half of the time, or there will be no point in examining the predictions any further. We will evaluate this by comparing our prediction error to the error were we to predict no movement at all, and claiming that the proportion of models that out perform a no-movement prediction is a proxy for the utility of predicting a player's movement at the given step size. By examining many models, we will also be able to test different structures of networks to create some general guidelines of what tuning parameters might help our networks achieve their maximum accuracy. Finally, we will examine the methods in which our results might be useful in an analytics environment, and describe the attributes that our model would need to display to make the use of these methods realistic.

3.2 Methods

In the first phase of modeling, I will evaluate the overall usefulness and feasibility of predicting the way in which a defender moves on a soccer field by predicting one small movement from the defender's current position. To have success in future experiments, we will count on the fact that movement is predictable at all, which is not trivial. At any given moment, a player may move in a 360° radius at a range of different magnitudes. Given that a player can move at 10 m/s at maximum speed, their possible movement after 5 frames (0.2 seconds) can land anywhere in a 2m circle, an area of 4πm, or about 12.5m. Of these 12.5m worth of possibilities for prediction, only about $r^2 * (\frac{2\pi}{3} - \frac{\sqrt{3}}{2})$ or $\approx$ 4.9m worth of the prediction gives us an estimate that is closer to the point than no prediction at all[1] This area, about 40% of the circle, is illustrated below.

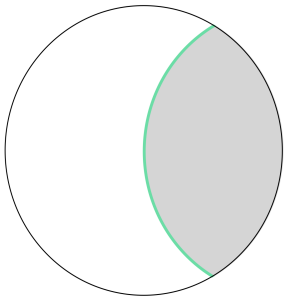

Figure 1: The area in which a prediction made by a model is better than predicting no movement at all..

To perform these initial experiments, I will test various neural network structures for frame step sizes of 1, 10, and 25. To examine which neural network structures work best, I will experiment with different numbers of hidden layers, numbers of nodes in each hidden layer, activation functions in each hidden layer, and batch sizes for the training of the networks. For this phase, all of the neural networks are created in Keras and Pytorch, as they allows for customization of each layer. Using these packages, we can specify all of the details of the layer like the number of inputs, size of the layer, and activation function. Our networks will all be Feed-Forward networks which use back propagation to learn and correct their patterns. With this method, we calculate the error in every layer in terms of the error in the next layer via the formula

$$\epsilon^l = ((w^{l+1})^T \epsilon^{l+1}) \odot \sigma'(z^l)^2 \quad (1)$$

We can also add as many layers as we want to, leading to a highly flexible style of model creation. When creating these Neural Networks and testing which network structures work best for the application of player movement prediction, I will follow some Neural Network rules of thumb [13].

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.

[1] We can think of this as the intersection of two circles at their midpoints. If we label the points of intersection A and B and the centers of the circles C and C', we can see that CC', AC, AC', BC, etc. are al the same length, r. One half of this area is then the area of a circular segment with radius r and angle $\frac{2\pi}{3}$ (the angle ACB). The total area is then $r^2(\theta - sin(\theta))$, or $r^2(\frac{2\pi}{3} - \frac{\sqrt{3}}{2})$

[2] where ϵ denotes the error vector, l denotes denotes the current layer, w denotes the weight vector, z^l is the weighted input to the neurons in layer l, and $\odot$ denotes the Hadamard product (elementwise product) between two vectors.

- The number of hidden neurons should be less than twice the size of the input layer.

Following these patterns, I will construct many neural networks, using the cross-validation techniques naturally supplied in the Keras and Pytorch packages. I will also experiment with layers of hidden neurons that expand the input space by containing more neurons than the input layer. Furthermore, I will test several of neural network depths to examine the effectiveness of using shallow and deep networks. Following the trend in recent years of expanding neural network depth, I will train networks from 1 hidden layer to 4 hidden layers in depth and compare them. By using a variable number of neurons and weights, we may allow our networks to better approximate the functions and linear relationships between the input variables and target variables. Our networks will use two output neurons, one for the predicted x position and one for the predicted y position.

We will look to minimize the difference between the prediction and the actual position of the player after the selected number of frames. We will evaluate the effectiveness of the predictions via the Mean Squared Error of the Euclidean Distance between the prediction and actual values:

$$1/N \sum_{i=1}^{N} \frac{(x_{i\_pred} - x_i)^2 + (y_{i\_pred} - y_i)^2}{2} \tag{2}$$

Our first predictions will be made for one step size only. We will fit many networks using the rules mentioned above, and evaluate the error of these predictions using equation 2, analyzing the player's actual position at the frame against our prediction of his position at the frame. For the smaller step sizes, we will also evaluate the error that would result in our network predicting no movement at all on the test set; this will give us an indication of whether the model is predicting the correct direction of movement for the frame.

After this initial testing phase, I will begin implementing the networks that make sequential predictions based on a starting position and the attacking player movement. These networks will give us the ability to move beyond prediction of one movement and into more realistic player movement forecasts. For this purpose, we will use a form of sequential networks called LSTMs [14].

These networks are more complex than the simple feed-forward networks used above, but also provide major advantages, like the ability to predict full sequences as well as the ability to learn multiple sequences at once through batch learning [15]. LSTMs differ from vanilla Recurrent Neural Networks (RNNs) through the addition of "gates" that help regulate the flow of information through the network. Through this added complexity, LSTMs retain the ability of RNNs to hold a "state" and keep track of long-term dependencies in data, but do not have the same vanishing gradient issues that RNNs tend to have [16]. The addition of a "forget" gate also allows the network to releases information that it doesn't deem important while holding the rest of the information constant from state to state. All of the LSTMs trained will use an Adam optimizer [17].

For our purposes, this means that the LSTMs trained for defenders will not only look at the positions of the attacking team and ball at the time of prediction like in the previous models, but will also learn movement based on their previous positions as well as the defender's previous positions. By forming these dependencies between all of the given predictors and responses, the networks will then predict smooth movement for defenders rather than singular predictions.

Following the thought process from Seidl's ghosting models, we will train separate LSTM's for different "roles" on the soccer field. However, this is not as straight forward as defining roles in basketball, where lineups of 5 are created with 5 distinct positional archetypes. Soccer has many different possible formations and is extremely fluid in terms of team structure and organization, making positional tagging much tougher. To get around this, we will train LSTMs to learn the reactions of the defenders by proximity to the ball, using separate models for each of the 5 nearest defenders to the ball. These models will make the assumption that team and opponent do not make a difference in the ways in which defenders will defend, but further extensions of the models would have the ability to learn different team defending styles through the addition of more training examples.

These sequential methods will provide insight about how to implement a ghosting algorithm like the one implemented in the NBA [12] for prediction of a defense's movement based on a drawn offensive play. To evaluate the accuracy of this prediction method, we will still use equation 2, but this time we will be predicting the difference between the player's true position after multiple steps and the position that we predicted at the same moment.

3.3 Data

Tracking data is a difficult thing to obtain. The vast majority of tracking data is owned by clubs and data providers, and any free data has great limitations. Therefore, all of the data used for these models is privately owned and will not be shared, but has been authorized for use on this paper. The data used for training and testing the model will be tracking data from the Friends of Tracking research group [18] as well as from Nashville Soccer Club. The full data from Nashville SC contains every player's position at 25 frames per second for a ninety minute match, resulting in a total of $22 * 25 * 60 * 90 \approx 3mil.$ data points for each match. Before cleaning, the data are in a JSON format with each frame and player listed separately.

{'period': 1,
'frameIdx': 0,
'gameClock': 0.0,
'wallClock': 1619287681553,
'homePlayers': ['playerId': '95309', 'number': 1, 'xyz': [-49.19, 0.24, 0.0], 'speed': 0.0,
...],
'awayPlayers': ['playerId': '119575', 'number': 30, 'xyz': [-0.58, -14.8, -0.0], 'speed': 0.0,
...],
'ball': 'xyz': [-0.02, 0.07, 0.34], 'speed': 10.15,
'live': False,
'lastTouch': 'home'}

After cleaning the data, we will have a more usable format which contains information abut the game clock, ball location, player location, and player unit (defense, midfield, attack), as well as the

response variables containing those players' positions 1, 10, and 25 frames into the future.

I will be using two matches for the training from this Nashville data, which will be the matches between Nashville SC and Montreal CF on the 24th of April, 2021 and between Nashville SC and New England Revolution on the 8th of May, 2021. In total, we will have about 2 million data points from these matches after filtering out defensive players and goalkeepers, as we won't be interested in the prediction of these players' movement. For the sequential prediction, we will be training on threatening attacking moves that either end in a shot or a turnover after a sustained period of possession. The Friends of Tracking Data is more limited; it only contains about 14 players in each frame and frames are only captured at a rate of 10 per second. For this reason, this data will be used for testing and analysis of the model only. This could cause biases in the results, as the leagues in the two matches are different and the analysis data will have missing players. This paper will therefore should serve as an academic example of how Neural Networks *could* be used in player movement prediction rather than providing a detailed analysis of player specific results. Accurate, detailed player analysis is a possible extension of the model if and when the data is available.

For the sequential models, we need to find longer sequences of data, each coming from an attacking situation such that target players are only being tracked while they are defending. We start with sequences leading up to shots, of which there are 51 between the two matches, and supplement this data with more long attacking sequences found in the data. In all, this gives us 211 attacking sequences to train and test our networks on, which is not as many as we would like for training of LSTMs in practice, but will be sufficient for our purposes.

4 RESULTS

4.1 Single-step predictions

Training the Feed-Forward Neural Network models in the one-step prediction approach yield the results found in the appendix of the paper in tables 4, 5, and 6. The three best models for each prediction interval (1 frame, 10 frames, and 25 frames) are below (the models are also highlighted in yellow in the results tables in the appendix). In these results, we transpose the error units from the Mean Squared Test Error that we used to validate the models into yards to obtain units that we are familiar with, but the true MSE can also be viewed in the results in the appendix. This transposition is necessary due to the min-max scaling and standardizing of the variables before fitting our networks. [3]

Layer Sizes	Batch Size	Activation	Error (yards)
0	0	Last Position	0.085
50, 25, 12, 6	100	ReLu, Sigmoid, R, S	0.085
24	100	R	0.086

Table 1: Predicting movement 1 frame ahead.

[3]ReLu (R) and Sigmoid (S) will be abbreviated after the first use of the terms. For further explanation on these tables and models, please check Methods (section 3.2) and the Appendix.

In our predictions of one frame ahead, no models performed much better than simply predicting the same position that the player was in in the last frame. This is not necessarily surprising - a player can only move so much in 1/25th of a second. Of the models that get close to predicting the same position, one is our deepest network with four layers of size 50, 25, 12, and 6. The Activation functions in this network are alternating, and it uses the mini batch algorithm described in section 3.2. Figure 2 in the appendix shows an example of a good prediction made for a step size of 1 frame.

Layer Sizes	Batch Size	Activation	Test Error
0	0	Last Position	0.75
50, 25	100	R, S	0.617
50, 25, 12, 6	100	R, S, R, S	0.623
24, 12, 6	100	R, R, R	0.635

Table 2: Predicting movement 10 frames ahead.

In our predictions of the 10th frame in the future, most of the models that we fit are able to beat a prediction of the position previous, which means the models are at least picking the right direction of movement on average. Our best model is one which uses two layers of sizes 50 and 25, with ReLu and Sigmoid activation functions respectively. All of the best models are deeper than one layer, and two of the three use at least one Sigmoid activation function. The mini-batch method consistently performs the best once again. Figure 3 in the Appendix shows an example of a bad prediction made for a step size of 10 frames ahead.

Layer Sizes	Batch Size	Activation	Error (yards)
0	0	Last Position	1.658
50, 25	100	R, S	1.516
24, 12, 6	100	R, R, S	1.524
50, 25, 12, 6	100	R, S, R, S	1.538

Table 3: Predicting movement 25 frames ahead

In the predictions of 25 frames into the future, almost all of the models fit outperform the prediction of the player's last position, which again means that the models are at least tending to pick the right direction of player movement. However, the predictions are not very close to the actual position, with the best one being about 1.51 yards away from the player's actual position on average whereas guessing the same exact position the player was just in would yield an average error of 1.66 yards. We see many of the same models performing well; the best model uses two layers of size 50 and 25, with a ReLu and Sigmoid activation function respectively. Making the list of the top three models again is the network 4 layers deep, with alternating ReLu and Sigmoid activation functions. Again, the mini-batch method performs the best on the data. Figure 4 shows an example of a good prediction that our models made for a step size of 25 frames.

The two consistently best performing networks had some similarities: their first hidden layer contains 50 nodes and their second

hidden layer contains 25 nodes. The two networks both use our mini-batch training method with a batch size of 1/100th of the total data. They also shared the similarity that their first hidden layer used ReLu activation functions, while the second used a Sigmoid activation function. The shallower network tended to perform better on the larger prediction steps (10 and 25), while the deep network made the best predictions on the prediction of 1 frame. The shallower network contains 90 nodes and 1950 edges, while the deeper network contains 108 nodes and 2284 edges. All of the layers in both networks are dense layers, which means that each of their neurons receive input from all neurons of the previous layer.

4.2 Sequential predictions

Moving away from single-step predictions to more realistic prediction scenarios, we will analyze the results from our sequential networks. These results will differ from the single-step predictions in several ways, the most obvious being the error reported. Rather than reporting the error at the end of the series of frames, we report the average error over the number of predictions made in the sequence. This means that the errors between the two types of prediction cannot be directly compared, and should instead only be compared to other networks of the same structure.

As mentioned before, the sequential networks were created with LSTM models ranging in size and structure. The architecture of these networks ranged from 1 LSTM network with 1 layer of 5 hidden units to 3 stacked LSTMs with variable hidden layer sizes, giving the networks many chances to approximate the function of player movement as accurately as possible. Many more network architectures were tested, but only a few are shown below for brevity.

The results of fitting several LSTM structures using 5-fold cross-validation are displayed in the table below.

Hidden units by layer	Epochs	Test Error (yards)
1	100	22.62
5, 5	100	8.14
25, 25, 25	100	13.19
1	1000	22.79
5, 5	1000	6.25
25, 25, 25	1000	10.11
1	2500	16.93
5, 5	2500	8.82
25, 25, 25	2500	13.21

Table 4: Sequential Predictions

We use a linear network, 1 layer and 1 hidden unit, as a baseline for the rest of our models. We notice that adding more layers and hidden units does improve the network over the linear network, though the networks are prone to over-fitting. The networks with 2 hidden layers and 5 hidden units per layer performed the best across the number of epochs, and performed best itself after training with 1000 epochs, where the network performed at an error average of 6.1 yards.

These networks tended to make predictions that were quite good, even in their worse cases. Examples of predictions made with our best model, ranging from very good to relatively poor, are shown in figures 5 through 8.

5 DISCUSSION

We will first aim to answer the first two questions in the Formulation section regarding the feasibility of offensive player movement prediction using Neural Networks and the tuning parameters that result in the best predictions.

The predictions made in our networks did generally beat the prediction no movement, but not always by as much as we might expect or want them to if we were putting a model into production at a club. Still, it is positive that we did beat a prediction of no movement, because this tells us that our predictions were generally in the correct direction of movement. We could describe the level of utility in our model's prediction of a new point as $\frac{point_{no\ movement} - point_{predicted}}{point_{no\ movement}}$, where a higher percentage number describes a better prediction model. With this metric, evaluating our best prediction for each model type would yield the results in table 6.

Step Size of Predictions	Utility
1	0%
10	17.7%
25	8.5%
Sequential	72.5%

Table 5: The percentage utility, defined as the improvement over over predicting no movement, of making a prediction with our best model for each step size.

Our sequential predictions provide more utility than our one step predictions. The defending players moved more in the sequential prediction phase since more frames were shown, but the sequential models also almost never predicted incorrect directions of movement on the test cases.

5.1 Applications

It is imperative to any analytics method that we consider in a professional environment that it provide feasible and actionable applications. There are two applications that immediately stand out: one being an extension of player similarity scores and the other being a ghosting model such as the ones created by Le and Seidl.

To provide an insight into player similarity or play style classification, we can use our model to predict the direction that a player is expected move in a given situation. We can then observe the player's actual movement, and classify certain characteristics about their play such as their level of unpredictability in movement and their level of aggressiveness in defensive movement compared to what is expected. To explore these questions, we will use the best of our trained sequential models, analyzing sequences of play in which a defender makes a movement in response to attackers.

This metric will be most useful when comparing players making movements in similar situations in a match (for instance, when they are the on-ball defender). We can find all situations like this, make predictions of how a given defender will move in the situation, then

compare this to how the defender actually moves. Due to a lack of data, we will use our testing sequences as an example, but in practice we could apply our model to many situations over several matches to get a more robust similarity metric.

Table 6 shows the results of running our prediction model in these situations and comparing the actual movement of a player to the expected movement. We use an angle measurement to do so, where 180° points directly toward the opponent's goal, and 0° points toward the player's own goal. The numbers below denote a *difference* in the true movement vs. expected movement: a negative value denotes a defender moving more toward their own goal than expected, while a positive value denotes a defender moving more toward the opponent's goal than expected. These can be translated as being more or less aggressive as an on-ball defender.

The graphics for the two attacking scenarios analyzed are found in figures 9 - 12 in the appendix. In the first play, we see that the predicted attacker movement is small, and that the true movement of CF Montreal's Center Back Aljaz Struna follows this predicted movement quite closely. We then find a similar attacking scenario, and notice that Nashville SC's Right Back Alistair Johnston is slightly more toward the opponent's side of the pitch than the predicted movement. We can quantify these differences as average movement vs. expected:

Player	Avg. Movement vs. Expected
Johnston	24.4°
Struna	9.2°

Table 6: Defensive player movement in attacking scenarios found in figures 9-12.

In our results, see that though both players move slightly more aggressively than expected, Johnston moves an average of about 15 degrees more aggressively toward the opponent's end of the field than we would expect him to.

This analysis gives us an insight into player movement and allows us to compare the movement across situations and players. This comparison is just over one attacking sequence, and more data would lead to more interpretable and stable play-style metrics.

The other application for our model is for ghosting, or creating predictions of how players will move for longer periods of time. To create a ghosting model, we can train sequential networks to learn movement in different defensive roles, defined here by proximity to the ball such that the on-ball defender is one role, the second closest defender is another, and so on. We can then make predictions about the movement of each player in each role on real or created data to see the predicted ways in which defensive players will react to an an attacking movement. An example of a 'ghosted' prediction is shown in figure 13 in the appendix. More data would improve this ghosting model as well and would allow for interesting extensions, such as the creation of a separate ghosting model for different players and/or teams.

5.2 Conclusion

We began by asking whether it was possible to predict player movement with neural networks, the types of data structures and models that might allow us to do so, and how we could use the results in an analytics environment. We can now say that predicting defender movement is possible to a fairly accurate level using sequential models like LSTMs, and that similarity scores and ghosting models are possible use cases for these prediction models in an analytics environment.

ACKNOWLEDGMENTS

Thank you to:

- David Sumpter, SkillCorner, and the Friends of Tracking research group for allowing me to use the tracking data highlighted in this paper.
- Caleb Shreve and Nashville Soccer Club for allowing me to use their tracking data for this research.
- Aaron Nielsen, thesis advisor.
- Ben Prytherch, thesis committee member.

REFERENCES

[1] Nate Silver. Introducing PECOTA - p. 507-514. *Gary Huckabay, Chris Kahrl, Dave Pease et al., Eds., Baseball Prospectus 2003*, 2003.
[2] Quentin Hardy. Reasons to believe the AI boom is real. 2018.
[3] Lucian Vintan, Arpad Gellert, Jan Petzold, and Theo Ungerer. Person movement prediction using neural networks. *Institut Für Informatik*, 2004.
[4] Abdulrahman Al-Molegi, Mohammed Jabreel, and Baraq Ghaleb. Stf-rnn: Space time features-based recurrent neural network for predicting people next location. pages 1–7, 2016.
[5] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. *Center for Research on Intelligent Perception and Computing*, 2016.
[6] Sam Green. Assessing the performance of Premier League goalscorers. *Stats Perform*, 2012.
[7] Jake Ensum, Richard Pollard, and Samuel Taylor. Applications of logistic regression to shots at goal in association football: calculation of shot probabilities, quantification of factors and player/team. *Journal of Sports Sciences*, 2004.
[8] Javier Fernandez and Luke Bornn. Wide open spaces: A statistical technique for measuring space creation in professional soccer. *MIT Sloan Sports Conference*, 2018.
[9] William Spearman and Luke Bornn. Beyond expected goals. *MIT Sloan Sports Conference*, 2018.
[10] Zach Lowe. Lights, cameras, revolution. *Grantland*, 2013.
[11] Hoang M. Le, Peter Carr, Yisong Yue, and Patrick Lucey. Data-driven ghosting using deep imitation learning. *MIT Sloan Sports Conference*, 2017.
[12] Thomas Seidl. Bhostgusters: Realtime interactive play sketching with synthesized nba defenses. *MIT Sloan Sports Conference*, 2018.
[13] Jeff Heaton. Introduction to neural networks with java - chapter 5: Feed forward neural networks, 2005.
[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
[15] César Laurant, Gabriel Pereyra, Philémon Brakel, Ying Zhang, and Yoshua Bengi. Batch normalized recurrent neural networks. 2015.
[16] Sepp Hochreiter. The vanishing gradient problem during learning: Recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 1998.
[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.
[18] SkillCorner via Friends of Tracking Research Group and David Sumpter. Premier League freeze frames 2020-2021. *Data Available by Request*, 2021.

A EXTENDED RESULTS AND FIGURES

The following describe the models and errors (test MSE) for all of the models that were fit, described in the experiments section.

Layer Sizes	Batch Size	Activation	Test Error
0	0	Last Position	6.27E-05
24	1	ReLu (R)	0.0001
24	5	R	6.72E-05
24	100	R	5.68E-05
2	100	R	6.15E-05
20	100	R	5.68E-05
6	100	R	6.13E-05
12, 6	100	R, R	6.69E-05
24, 12	100	R, R	5.47E-05
50, 25	100	R, Sigmoid (S)	5.08E-05
24, 12, 6	100	R, R, R	5.36E-05
24, 12, 6	100	R, R, S	5.37E-05
50, 25, 12, 6	100	R, S, R, S	5.24E-05

Table 7: Predicting movement 10 frames ahead.

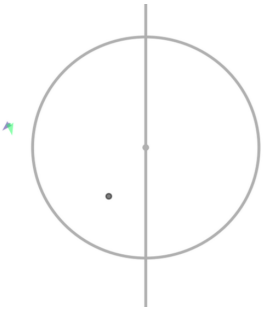

Figure 2: An example of a prediction made 1 frame ahead with our best single-step model (Error = 0.1 yards). Ball: black, true player movement: purple, movement prediction: green

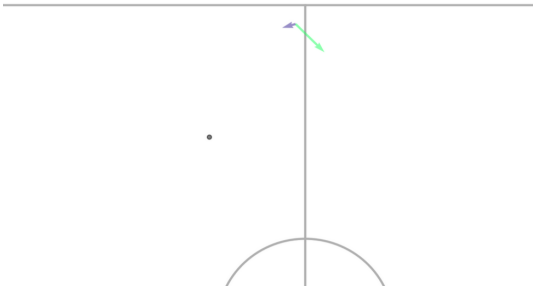

Figure 3: An example of a prediction made 10 frames ahead with out best single-step model (Error = 0.8 yards).

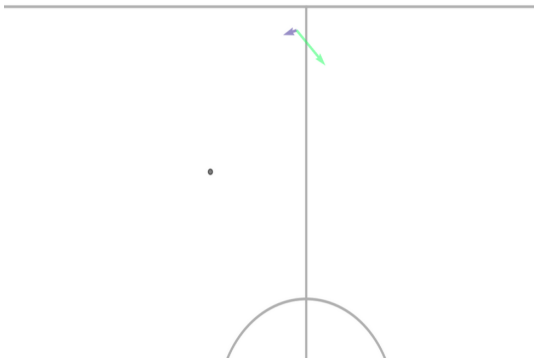

Figure 4: An example of a prediction made 25 frames ahead with our best single-step model (Error = 0.6 yards).

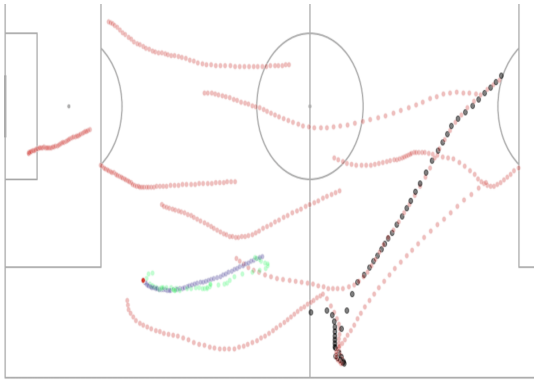

Figure 5: An example of a good prediction made 5 seconds worth of frames ahead with our best sequential model (Error = 1.2 yards). Note: In the sequential images, the red lines denote the attackers, the black line denotes the ball's movement, the purple line denotes the defender's true movement, and the green shows the network's prediction.

Figure 6: An example of a good prediction made 5 seconds worth of frames ahead with our best sequential model (Error = 2.4 yards).

Figure 7: An example of a prediction made 5 seconds worth of frames ahead with our best sequential model (Error = 4.7 yards).

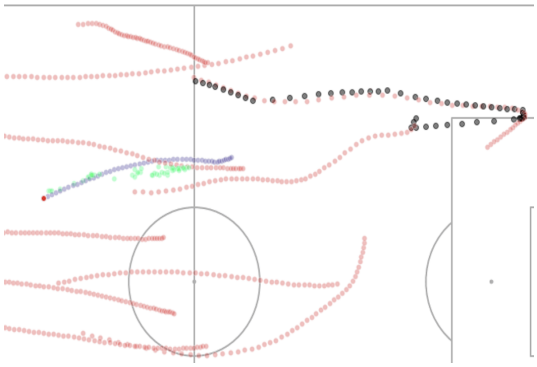

Figure 8: An example of a poor prediction made 5 seconds worth of frames ahead with our best sequential model (Error = 9.5 yards).

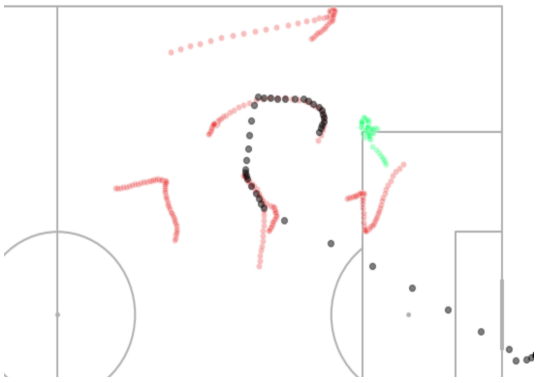

Figure 9: The predicted defender movement (green) in an attacking scenario.

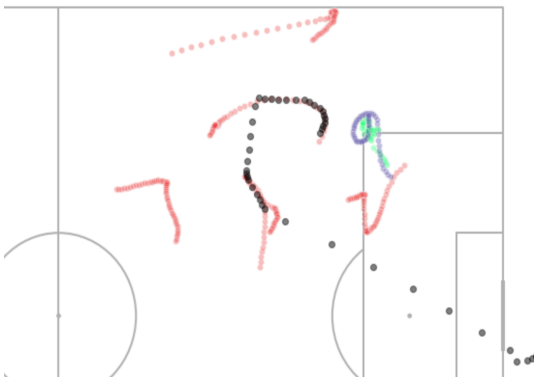

Figure 10: The true movement (purple) of CF Montreal Center Back Struna.

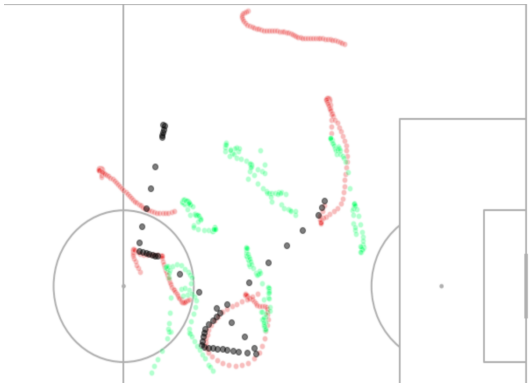

Figure 13: A ghosted prediction to an attacking sequence.

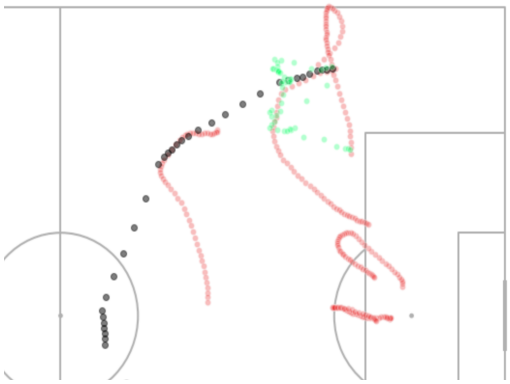

Figure 11: The predicted defender movement in a similar attacking scenario.

Figure 12: The true movement of Nashville Right back Johnston.