15TH ANNUAL MIT SLOAN SPORTS ANALYTICS CONFERENCE APRIL 8-9, 2021 | VIRTUAL presented by ESPN 42 ANALYTICS

Prediction of Defensive Player Trajectories in NFL Games with Defender CNN-LSTM Model

Lin Lee Cheong[1], Xiangyu Zeng[1] and Ankit Tyagi[2]
[1] Amazon ML Solutions Lab, [2] NFL Next Gen Stats (NGS)

1. Introduction

Defensive players in football have complex trajectory patterns that are hard to accurately model and evaluate. This is due to the defenders' inherently reactive role, which requires them to dynamically respond to offensive strategies and on-the-spot decisions by the quarterback. Amongst defensive players, *defensive backs* are the most challenging to model. They often travel long distances during passing plays to fulfill their coverage responsibilities and change trajectories depending on their perception of where the ball will go. This is in contrast to other defensive players, whose total distance travelled and directions of movement are generally much more limited.

There are 11 players lined up on each of the offensive and defensive side for every play in a National Football League (NFL) game, which translates to 22 players interacting with each other at any given time. Individual player trajectories are affected by their personal assignments, the current overall strategy (e.g., man versus zone coverage), and the movement and decisions of surrounding players and the ball. A football human expert is able to evaluate and predict a defensive back's trajectory, as there exists an innate ability in humans to predict near-future events and take sequential actions while accounting for complex arrays of factors and potential outcomes via joint attention. However, evaluating large numbers of interacting inputs to generate sequential predictions remains difficult.

Developing the ability to predict and evaluate these trajectories is of paramount importance to better assessments of defensive coverage, offensive strategy, quarterback decision-making quality and even the probability of winning plays and games. It is even more challenging to predict "what-if" scenarios – for example, how should the defensive backs' trajectories change if the receiver targeted in a passing play is changed? Answering these questions in a quantitative manner can provide talking points to football enthusiasts, and also help analytics-driven teams to better understand their offensive and defensive decisions.

An example of a "what-if" scenario in a passing play is shown in Figure 1, where a defender covering two receivers may move toward different directions depending on who the player perceives as the *targeted ball receiver*. A targeted receiver refers to the player that is thrown the ball to in the passing play, while a non-targeted receiver is an eligible offensive player who could but was not thrown the ball to during the play. As shown in Figure 1, the task is to predict the trajectory from pass forward to pass arrival (in red dashed lines) of Defender #25 (shaded in yellow) when the targeted receiver is Player #13, and Defender #25 moves to the left to stop the player. The other possible targeted receiver is Player #81, and a what-if scenario would be: what would be Defender #25's trajectory if the targeted receiver was Player #81 instead of Defender #25? We expect the model to adjust predictions based on this change in the targeted receiver, and to shift the defender's predicted trajectory towards the right to stop Player #81.

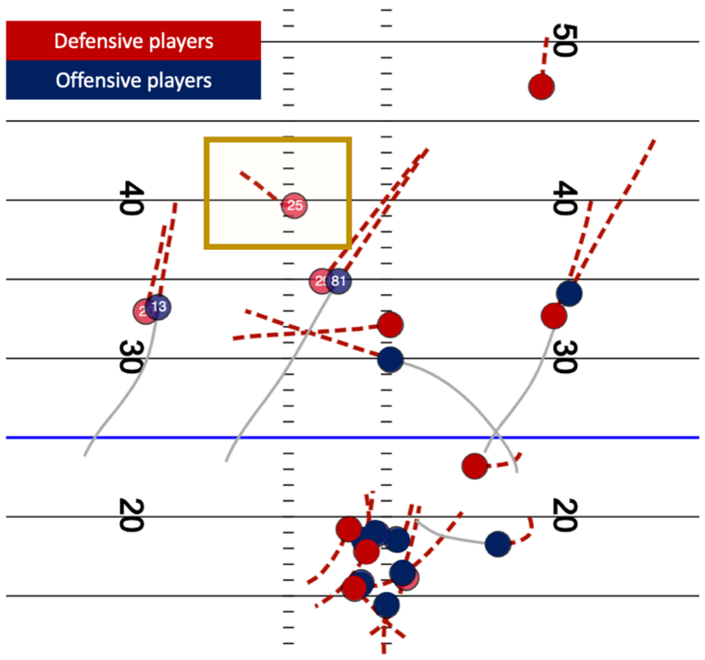


Figure 1. Schematic of a passing play, with defensive players in red and offensive players in blue. Our task is to predict the trajectory (in red dashed lines) of Defender #25 (shaded in yellow). In this example, the *targeted* receiver is Player #13 so Defender #25 moves to the left to stop the player. The other potential targeted receiver is Player #81, and a what-if scenario would be: what would be the Defender #25's trajectory if the targeted receiver was Player #81?

In this paper, we utilize player and ball sensor data collected through radio-frequency identification (RFID) tags on player's shoulder pads and in the game ball, with the tags transmitting location data every 10th of a second. The data includes coordinates, orientation and direction information for each player in the field, as well as the location and speed of the football. We also focus on passing plays, and not rushing plays, as passing plays generally require the defensive backs to cover long distances and change trajectories based on the targeted receiver.

The prediction of defensive backs' trajectory is essentially equivalent to a sequence prediction problem where the input sequence is the observed positions of a player and the output is a sequence identifying the player's future positions. At each time step t, sensor data is processed to include the spatial x-y coordinates (x_t^i, y_t^i) of player i at previous time steps [1, 2, 3]. Our task is to predict the defensive backs' x-y coordinates for a fixed number of future time steps T_{obs+1} to $T_{obs+pred\_window}$ while observing their ground-truth positions from time step *1* to T_{obs}. An example of using 5 previous time steps to predict defender trajectories up to 10-time steps later is schematically depicted in Figure 2.

Following the success of recurrent neural network (RNN) models for sequence prediction tasks, we utilize long short-term memory (LSTM) as part of our models to predict defender trajectories in football passing games. The models are capable of addressing what-if situations as depicted in Figure 1, and generate realistic trajectories when switching the targeted receiver for a play. We also develop trajectory metrics, as traditional metrics are unable to fully evaluate performance for this use case. Regression metrics such as root mean squared error (RMSE) require us to calculate errors relative to known results; this does not exist for hypothetical, what-if situations. For a defensive back guarding two eligible receivers, we calculate the RMSE with the actual receiver's recorded trajectory versus the predicted trajectory. For the other potential receiver, we cannot calculate RMSE due to a lack of actual trajectories. The newly-developed metrics score each trajectory independently, and incorporate spatial information and the maximum possible physical effort that a defender could realistically exert. This enables efficient evaluation and modeling by being able to differentiate between real-world blown coverage and modeling errors.

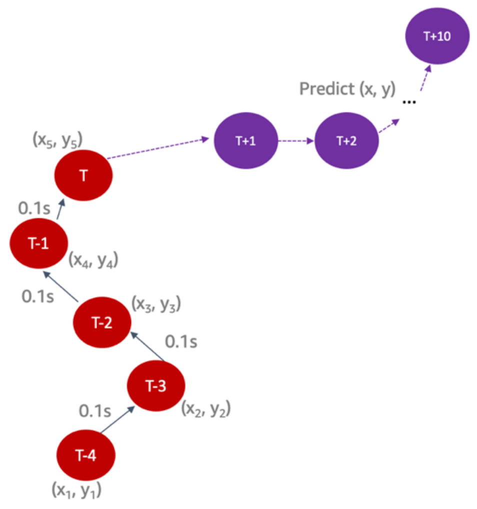


Figure 2. Task definition of sequence prediction, with a prediction window of 10-time steps illustrated.

The paper is structured as follows: first, we describe previous work in this space and some key concepts that are used as building blocks for this work in Section 2. We detail steps that were taken to mitigate data quality issues that affect model training and performance, and present sequence models capable of predicting defender trajectories and providing what-if predictions in Section 3 and Section 4. Finally, we describe trajectory metrics capable of evaluating the performance of a defender's trajectory, aggregating individual trajectory metrics to quantify performance of a play or games in Section 5.

2. Related Work and Key Concepts

Different aspects of this problem have been studied by respective scientific communities, including computer vision, robotics, and self-driving vehicles. Broadly, there are two main groups of human behavior forecasting as classified by interaction type: human-space interactions, and human-human interactions [4]. The former learns scene-specific motion patterns [5,6,7,8,9,10,11]; the latter focuses on how people interact with each other, and where the model learns about the dynamic content of the scenes. In this section, we focus and review only the most closely related literature.

2.1. Human-Human Interaction

A pioneering model of human motion, called social forces model was proposed by Helbing and Molnar [12]. The social forces model proposed two types of forces: 1) attractive forces that guide people towards their goal and 2) repulsive forces that encourage people to avoid collisions. Although the model was first realized in 1995, it continued to achieve competitive results on modern pedestrian datasets. Social forces model was later extended to robotics [13], and activity understanding [14,15,16,17,18]. Separately, Gaussian processes were used for modeling human motion by Tay and Laugier [19] and Wang *et. al.* [20].

The original method and its extensions were limited by two assumptions. First, these methods used hand-crafted functions to model "interactions" for specific settings rather than inferring them in a data-driven fashion. Thus, these models captured simple interactions but had difficulty generalizing in more complex settings, such as football games where settings change considerably depending on play calls, teams and player skillsets. Secondly, these methods do not take interactions that occur in the most distant future into account. In contrast, generic data-driven approaches that were proposed recently have outperformed these traditional approaches.

2.2. Recurrent Neural Networks (RNNs) for Human Motion Trajectory Prediction

There exist variants of RNNs, most notably Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU). LSTM networks, which are used in this paper, are well-suited for processing and making predictions based on time series data, as there can be lags of unknown duration between important events in a time series. Compared to traditional vanilla RNNs, GRUs and LSTMs are less negatively affected by the vanishing gradient problem during model training and have less long-range dependency issues. Both LSTM and GRU networks have found success with applications in sequence prediction in a variety of domains including machine translation [21], speech recognition [22, 4, 23], and image captioning [24, 25].

Following this, Alahi *et. al.* [1] approached the human trajectory prediction problem as a sequence prediction task and applied RNNs to solve it. There were challenges in capturing interactions between people with one independent RNN per person, and Alahi demonstrated that the addition of a social pooling layer that connected all neighboring LSTMs could enable interaction modeling. We apply and describe an adaptation of Alahi's social pooling concept to football games in Section 3.2, and utilize tabular sensor-based data as opposed to Alahi's image-based data.

2.3. 1-Dimensional (1D) Convolutional layers to extract time-dependent features

A convolutional neural network (CNN) is a standard model building block in the machine learning community, with 2-dimensional (2D) convolutions successfully employed in images and 3-dimensional (3D) convolutions in videos [30,31,32,33]. Since player tracking data arrives from sensors in the form of one-dimensional (1D) signals, we employ 1D convolutions with different kernel sizes to extract information from multiple players and to predict defender trajectory.

A pooling layer is commonly applied after CNN layers, and the resultant pooled output is a summarized version of the detected features. In additional to reducing model over-fitting, pooling provides translational invariance: input translations in space would not result in different trajectories (e.g., same trajectory shifted in space), and mirrored inputs would generate mirrored predicted trajectories.

3. CNN-LSTM based models for defender trajectory predictions

Two related models are described; both are capable of predicting a defender's trajectory while accounting for the targeted receiver and ball movement. A CNN-LSTM based model is presented in Section 3.1, and CNN-LSTM model with social pooling adaptation is described Section 3.2.

3.1. Defender CNN-LSTM model

A schematic of the deep learning model architecture is shown in Figure 5. Potential features from tabular sensor data and game-related information were initially evaluated via feature importance using a traditional XGBoost model, and this initial screening step is further described in Section 4.2. Important features comprise of information from the defender, ball and targeted receiver. The defender and targeted receiver data are treated as separate inputs to easily accommodate what-ifs situations (described in Section 1), where we would preferentially change the targeted receiver.

All inputs are first passed independently through 1D-CNN layers followed by pooling. The CNN layers extract relevant patterns through time (usually 3 to 5-time steps prior to current time step) that can be easily interpreted by the LSTM network. These CNN-extracted features are then concatenated and repeated for K time steps, and the LSTM sequence model predicts for K future time steps (usually set to 10 or 15-time steps). A dense layer is finally applied to each output step of the LSTM as we seek to minimize the RMSE between the predicted position and the real position.

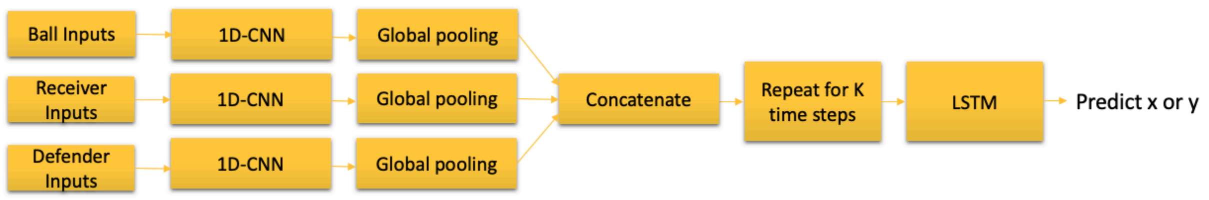


Figure 5. Defender CNN-LSTM model architecture. Features are passed through 1D-CNN, pooled and then concatenated together. The model is trained to predict spatial positions.

3.2. Defender CNN-LSTM model with social pooling

The inclusion of a social pooling strategy into the Defender CNN-LSTM model improves defender trajectory predictions, as it enables the model to account for interactions with other players who are spatially close to the defender. This is similar to how a defender would account for the behaviors of surrounding players when deciding next course of action. The social-pooled model architecture is shown in Figure 6. Inputs used in Defender CNN-LSTM model such as ball location and targeted receiver data are retained, and social-pooling simply augments the model's ability to extract essential, high-level features from ball and receiver data.

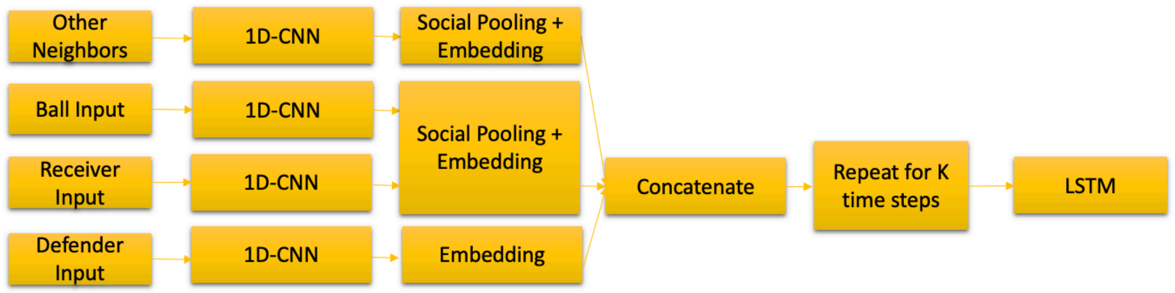


Figure 6. Defender CNN-LSTM Model with Social Pooling, for predicting defender trajectory with other neighbors' information.

In detail: $$S_t^i = [\, \phi(c_t^i); \phi(c_t^{R,B}); \phi(\textstyle\sum_{j \in N_i} c_t^j) \,]$$

where c is the vector of features from convolutional layer, specifically c_t^j is the output from convolutional layer of the j^{th} defensive neighbor to i^{th} defender at timestamp t. $c_t^{R,B}$ is the feature from receiver (R) and ball (B) inputs, and c_t^i stands for the i^{th} defender (whom we are predicting). This is followed by an embedding function $\phi(.)$ with rectified linear units (ReLU) non-linearity. And the resultant embeddings are concatenated to form the S_t^i which is the social pooled convolution feature for i^{th} defender at time step t.

In Alahi's approach, all players within a rectangle grid with four vertices $(x_t^j + \frac{m}{2}, y_t^j + \frac{n}{2})$, $(x_t^j + \frac{m}{2}, y_t^j - \frac{n}{2})$, $(x_t^j - \frac{m}{2}, y_t^j + \frac{n}{2})$, $(x_t^j - \frac{m}{2}, y_t^j - \frac{n}{2})$ to form a neighbor set, where m, n represent the length and width of the rectangle and we look to predict the jth defender's trajectory. Instead of a spatial cutoff, we apply social pooling to all other defensive backs within the play, and separately to ball and receiver inputs. Note the social pooling layer does not introduce any additional parameters.

4. Experiments

Results presented in this section utilized an NFL-collected dataset in 2018 and 2019. Data prior to 2018 was unavailable due to changes in sensor hardware and software, which limited the total number of observations available. The data was split into 1,068,910 unique trajectories of defenders for training, and 264,155 unique trajectories for testing. The train and test set were also split by play to prevent information leakage.

4.1. Data cleaning, outlier detection and replacement

As with most real-world noisy data, data cleaning was necessary prior to model training to ensure realistic trajectories and accurate performance. We first filtered and removed all plays that contained 'superhuman' movements, i.e., movements by a player that exceeds limitations of human beings. Specifically, we upper bounded the recorded speed at 12.47 yards per second. Extending from this speed limit and with a time step of 0.1s between recorded data points, we also removed plays where the distance between two consecutive observed points in a player's trajectory is larger than 1.247 yards.

Secondly, we removed plays that contained player trajectories that were outside of the football field during a play. Retaining these problematic trajectories in the dataset would result in the model learning inappropriate behaviors such as running off the field during a play. Finally, we only retained data from pass forward to pass arrived. This is because we are primarily concerned about trajectories of the defensive back and how they reacted to different potential ball receivers (which is hardest to predict) while the ball in still in the air.

4.2. Input features

We applied an XGBoost model and simple LSTM models to quickly explore, sub-select and validate a variety of raw and engineered features. Potential features included player speed, acceleration, types and number of personnel on the field for each play, x-y coordinates of players a few time steps prior, direction and orientation of the players in motion, and ball trajectory and velocity. Useful feature engineering steps included differencing (which renders the time series stationary) and directional decomposition (which decomposed a player's rotational direction in x and y directions). Game-related features such as player role, yards to go to first down, yard from endzone were evaluated but deemed as not important by these initial-screening models. This feature reduction is essential in avoiding model overfitting, due to the limited number of observations and the redundancy of information in the many input features.

Experimentation indicated that using defenders' inputs between 3 to 5 previous time steps as inputs were sufficient, and longer time steps did not result in better performance. Inputs from the ball and the targeted receiver were also provided to the model on a real-time basis beyond current time step.

4.3. Additional implementation details

The number of filters used in the convolution layer was 32, and a fixed hidden state dimension of 32 was used for all the LSTM models. We used an embedding layer with ReLU (Rectified Linear Units) non-linearity on top of the pooled convolution features. An adaptive learning rate optimization algorithm (Adam) optimizer was used, and MSE selected as the loss function. The model was trained on on a single GPU with a Keras implementation.

The model was trained to predict up to ten timesteps out ($t + 1$ to $t + 10$) during initial development phase and to predict up to 15 timesteps ($t + 15$) when implemented into production. Majority of passing plays were up to 10 timesteps long, and plays longer than 10- or 15-time steps were "tiled" in a sequential manner to generate long sequences. Fixing the sequences between 10-15-time steps enabled us to sufficiently augment the dataset by creating multiple training sequences out of a single long sequence, as there was insufficient data to adequately train a direct sequence-to-sequence model. We trained and predicted the spatial positions x and y separately. It is possible to train a single model to predict both (x, y), with the single-model performing slightly worse than two independent models.

For the social-pooled Defender CNN-LSTM model, there were at most five neighbors at any time. We padded neighbor inputs into fixed shape ($batch\ size$, $time\ steps$, $feature\ dimensions$, 5) and applied a convolutional 2D layer with kernel size ($1, feature\_dim$) for computational speed. This is mathematically equivalent to applying a 1D-CNN for each neighbor independently followed with social pooling.

4.4. Results

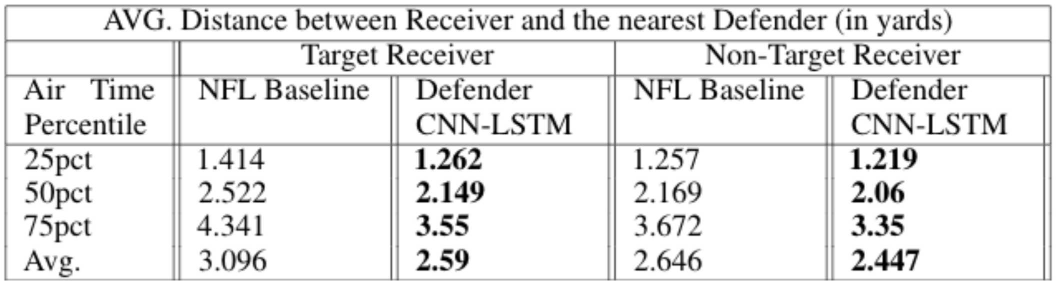
Table 1 summarizes the prediction errors using root mean squared error (RMSE). All the errors were calculated using the test dataset, which was previously unseen by the model during training. The Defender CNN-LSTM model outperformed other models on most of the time steps except for the first-time step. We believe the Defender CNN-LSTM model outperformed other models in the later time steps because it was able to account for its own previous actions and the actions of others (in the form of the hidden state). The XGBoost model, which is not a sequence model, was trained directly against the target for that specific time step. It is not surprising that XGBoost initially performs better but is vastly outperformed by neural networks for subsequent time steps, as the loss function in the sequence models were applied against all time steps (single overall loss).

For additional validation, we deployed the Defender CNN-LSTM model into production and tested on a separate internal test dataset with a different distribution compared to our training data. In this case, we predicted the trajectories of all defenders for 15-time steps on two categories: (1) the defender closest to the targeted receiver and (2) defenders whose closest offensive player was a non-targeted receiver. As we do not have recorded trajectories of defenders when the ball is passed to the non-targeted receiver (i.e., a hypothetical play), we instead calculated the distances between receiver location and predicted location nearest defender in each time step.

The results are in yards, and tabulated in Table 2. The percentile in the table represents the air time: number of seconds the ball travels in the air. All plays are sorted by air time. For instance, 25pct means first 25% plays with shortest air time. For all air time percentiles, the predicted trajectories of the nearest defender from the Defender CNN-LSTM model are closer to the receivers than trajectories from NFL baseline model (a simple LSTM model). This approach of quantifying model performance and trajectories in hypothetical situations is at best comparative, and is not possible to evaluate the quality of predicted trajectories. Independent metrics for evaluating predicted trajectories are proposed and elaborated in Section 5.2.

AVG. RMSE Results (in yards)				
Time Stamp	XGBoost	LSTM	Defender CNN-LSTM	Defender CNN-LSTM with Social Pooling
1	**0.0278**	0.0736	0.0768	0.0606
2	0.0670	0.0793	0.0763	**0.0667**
3	0.1188	0.0982	0.0861	**0.0712**
4	0.1873	0.0983	0.1016	**0.0922**
5	0.2735	0.1350	0.1322	**0.1233**
6	0.3765	0.1852	0.1758	**0.1674**
7	0.4991	0.2515	0.2316	**0.2235**
8	0.6317	0.3331	0.2986	**0.2906**
9	0.7787	0.430	0.3762	**0.3687**
10	0.9484	0.5615	0.4650	**0.4583**

Table 1: Quantitative results of all the methods on the NFL dataset to predict 10 time steps. The XGBoost model was trained with the Amazon SageMaker built-in algorithm. The LSTM model is a simple sequence model without CNN layer.



AVG. Distance between Receiver and the nearest Defender (in yards)				
	Target Receiver		Non-Target Receiver	
Air Time Percentile	NFL Baseline	Defender CNN-LSTM	NFL Baseline	Defender CNN-LSTM
25pct	1.414	**1.262**	1.257	**1.219**
50pct	2.522	**2.149**	2.169	**2.06**
75pct	4.341	**3.55**	3.672	**3.35**
Avg.	3.096	**2.59**	2.646	**2.447**

Table 2: Quantitative performances on NFL internal test data. Evaluated by average distances between the ball receiver and the nearest defender over 15-time steps.

5. Analysis of predicted trajectories

We analyzed the actual and predicted trajectories to gain further intuition and insights into model behavior. The optimal behavior of a defensive back is defined here as moving as quickly as possible towards the targeted receiver, who is expecting to catch the ball. Desired but less common behaviors such as interceptions and "jumping the route" are not considered in this paper as these incidents account for relatively few trajectories within the collected dataset, and require additional labeling and information not currently available to us. By this definition, a correctly trained model is expected to predict markedly different defender trajectories when the targeted receiver is changed, with the defender moving towards the targeted receiver to stop him.

Sample trajectories for visual evaluation are provided in Section 5.1, and are from the Defender CNN-LSTM model without social pooling (which was developed initially and currently in deployment). We then present newly-developed metrics capable of quantifying these trajectories in Section 5.2. These metrics enable efficient evaluation of trajectories, plays and overall model performance.

5.1. Qualitative evaluation of defender trajectories

A pair of sample scenes is shown in Figure 7, with different targeted receivers in the same play and a focus on Defender #3's predicted trajectory. Offensive players are colored in blue circles, while defensive players are in red. The predicted trajectories are denoted in dashed lines, while previous trajectories (just prior to the quarterback making a forward pass to the targeted receiver) are in solid lines.

In the left play, the targeted receiver is #6, and defender #3 moves to the left to stop the player. When the targeted receiver is changed to #7, the model correspondingly adjusts the trajectory to the right. The turn by defender #3 is necessary as the player's orientation and current movements are also taken into consideration by the model.

Additional examples on two other plays are shown in Figure 8 (defender #0) and Figure 9 (defender #7).

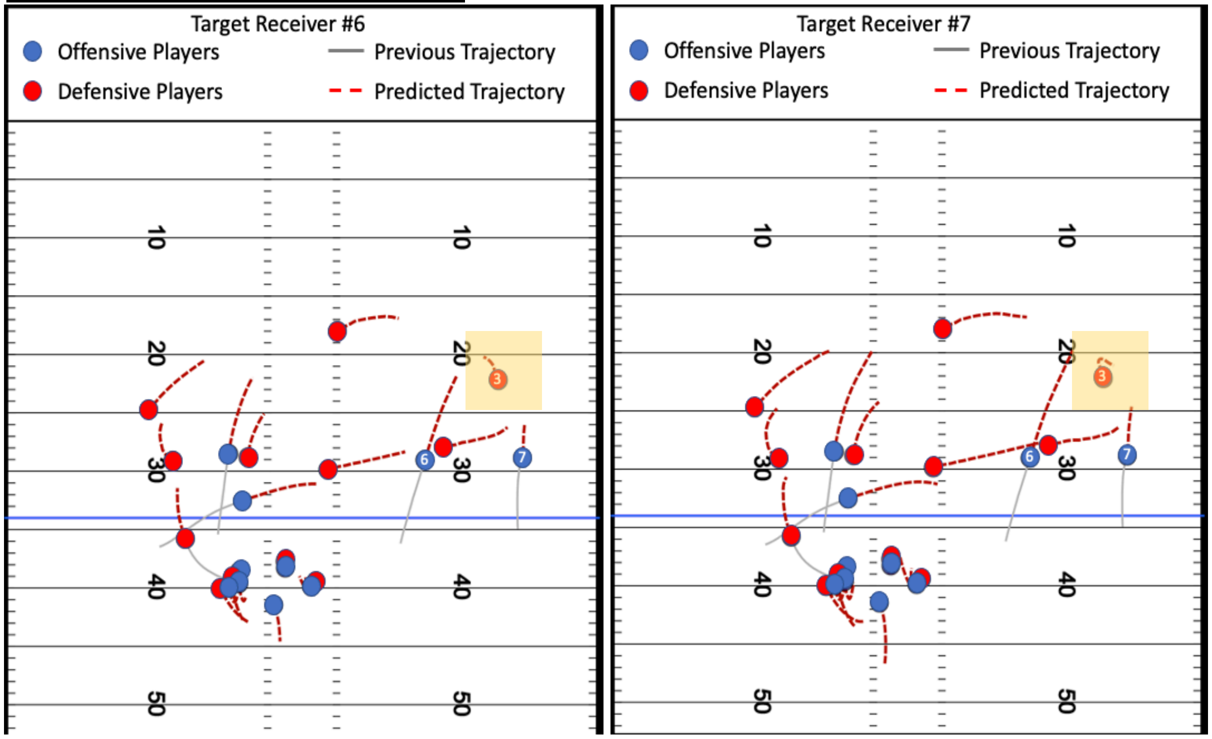


Figure 7. Example of changing target ball receiver. Defender #3 moves towards different directions based on who is the perceived targeted receiver, as predicted by the Defender CNN-LSTM model.

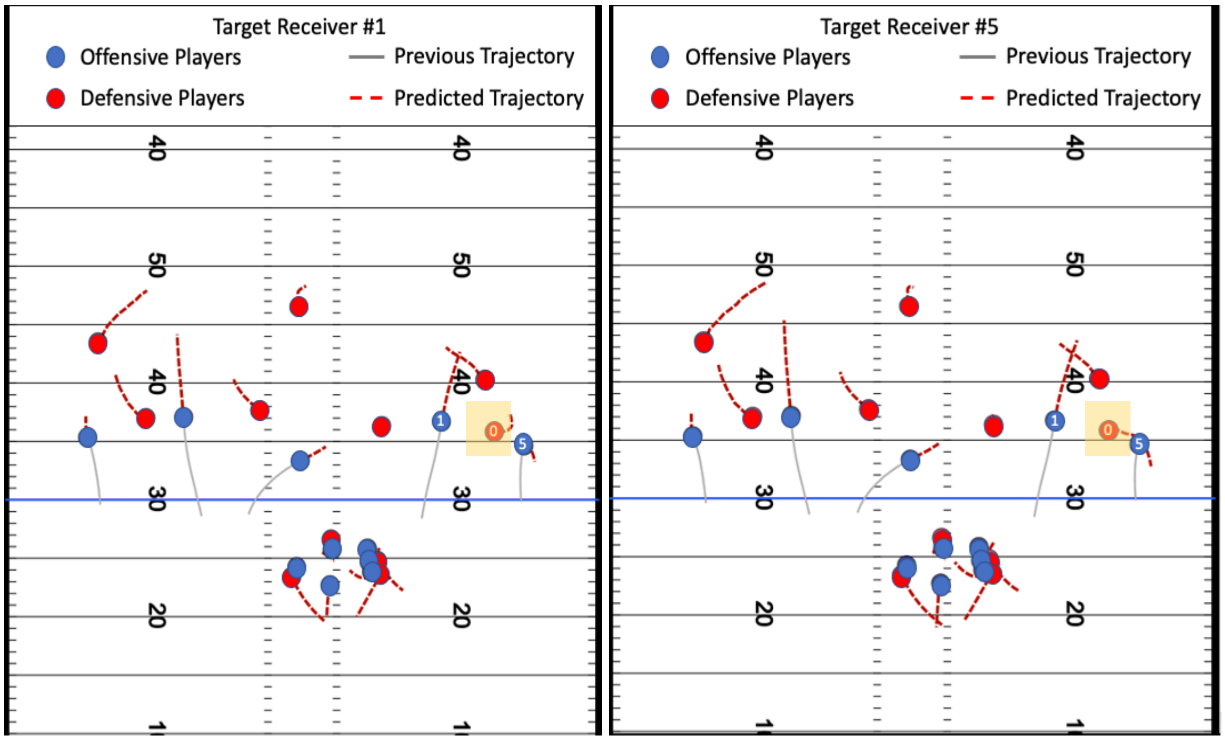


Figure 8. Defender #0 moves towards different directions based on who is the targeted receiver.

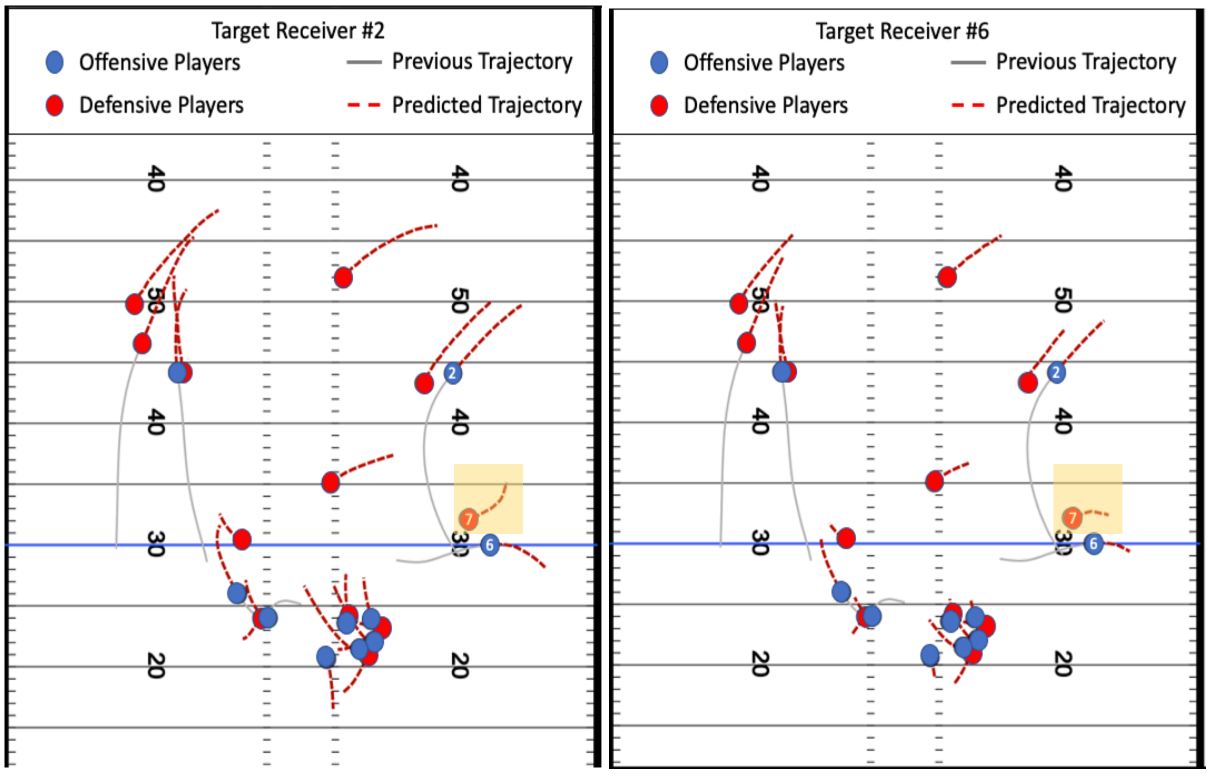


Figure 9. Defender #7 moves towards different directions based on who is the targeted receiver.

5.2. Quantitative evaluation of predicted trajectories

We presented RMSE results in Table 2 on the NFL private test dataset using the Defender CNN-LSTM by comparing distances between the nearest defender and the receiver, which is an incomplete picture. The evaluation of performance via RMSE metric in Table 1 was made possible by comparing against actual defender trajectories, but actual trajectories are not available in what-if situations shown in the right of Figures 7, 8, and 9. Metrics capable of independently evaluating trajectories are needed, and three new metrics are introduced to enable us to assess overall model performance:

1. a trajectory score (TS) to evaluate a single trajectory,
2. a play score that aggregates multiple defenders' TS scores, and
3. an overall model performance score based on a set of play scores.

5.2.1 Scoring a single trajectory

The trajectory score is an aggregate score composed of three different evaluations: positional convergence (PS), distance ratio (DR) and superhuman (S). With optimal defensive back behavior as defined at the start of Section 5, we now introduce the concept of positional convergence (PS) metric defined as the weighted average of the rate of change of distance between the two players over multiple time steps. As shown in Figure 10, using the rate of change enables fair evaluation between the defender and receiver regardless of actual distance. When equally weighted across all time steps, the sign of the PS metric indicates whether the two players are: 1/ spatially converging when negative, 2/ running in parallel when zero, or 3/ spatially diverging (moving away from each other) when positive. There are multiple options for weighting the rate of change across multiple time steps, and the choice is application-dependent. If the entire trajectory is needed for a use case, then equal weights should be applied to every time step. If parts of the trajectory are more important, i.e., using final location as inputs into other systems, then the final time step should be more heavily weighted relative to other time steps.

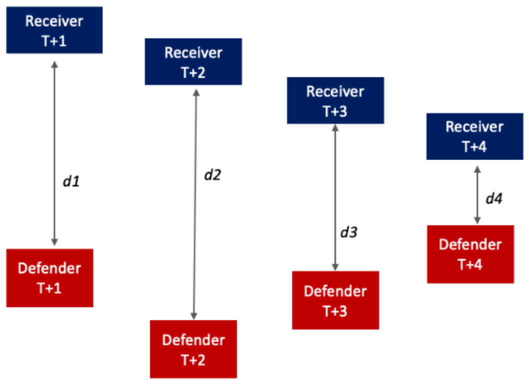


Figure 10: Schematic depicting concept of positional convergence: weighted average of rate of change of the distance between the defender and the targeted receiver. We show the positions of a targeted receiver and predicted defender trajectory at four-time steps. The distance at each time step is denoted in arrows d_t, and we use the average rate of change of this distance to compute the PS metric.

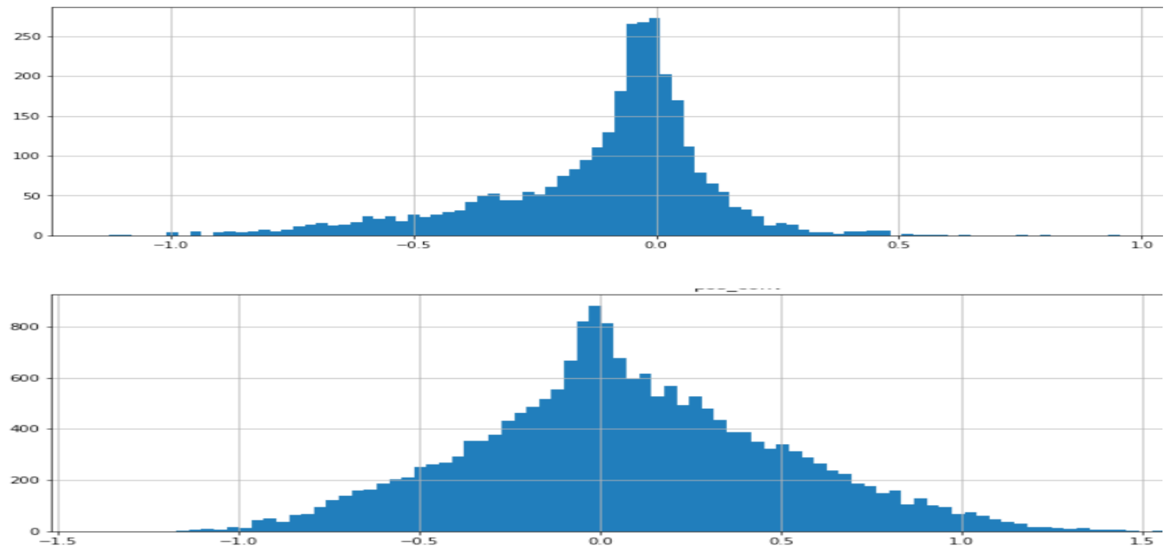


Figure 11: (top) Histogram of weighted PS for the nearest defender to targeted receiver. (bottom) Histogram of weighted PS for all defenders. Note the increase in non-convergence by defenders who are far away, which is represented by the right, positive half of the figures.

Histograms for the calculated PS metric are shown in Figure 11, both for the nearest defender to the targeted receiver (top), and for all defenders (bottom). As expected, there are far less non-convergence cases when subset to nearest defender, and the majority of the scores are between -1 and 1.

The PS metric alone is insufficient to evaluate the quality of a play as a defender could be running too slowly towards the targeted receiver, i.e., insufficient effort. We thus compare the distance covered by the defender, against the "best-effort" distance that *could* be covered by the defender. This optimal distance is approximated by calculating the distance between the defender's location pass-forward and the position of the receiver at pass-arrived, and is shown in Figure 12. The maximum possible distance is upper bounded by the maximum distance a defender can humanly cover per time step. We note that it is possible to improve the accuracy of optimal trajectory by considering player orientation and current speed, at the cost of computational speed and complexity. This effort estimate, termed distance ratio (DR), is used to modulate the PS metric. The defender's actual distance covered is compared against the approximate optimal distance, and a score that rewards trajectories that indicate that the defender has covered close to optimal or humanly possible distances is assigned. The DR equation is as follows:

$$DR = \frac{|\,optimal\ distance - actual\ distance|}{optimal\ distance + actual\ distance}$$

The DR metric returns a score between zero and 1, with scores closer to one being closest to optimal distances. Examples of two trajectories scored differently are shown in Figure 13.

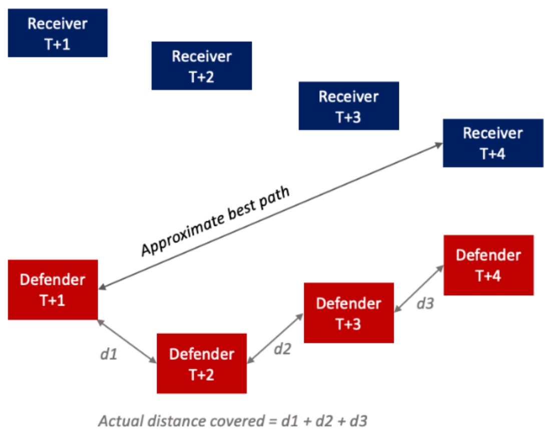


Figure 12: Approximate "best-effort" approximate distance covered by a defender, and the actual distance covered by the defender. The maximum possible distance is upper bounded by the maximum distance a defender can humanly cover per time step. We note that it is possible to improve the accuracy of optimal trajectory by considering player orientation and current speed, at the cost of computational speed and complexity.

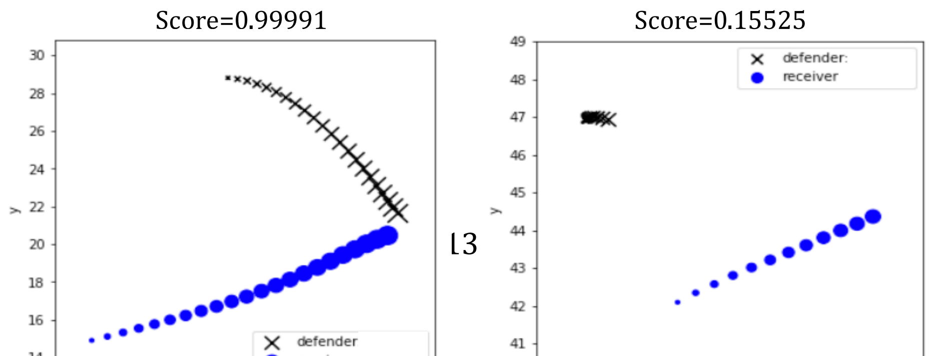


Figure 13: (left) example of a trajectory that was scored highly by the DR metric, and (right) example of a trajectory that was scored poorly as the defender could've covered more distance towards the targeted receiver.

We also heavily penalized predicted trajectories that contain movements that are humanly impossible, which we termed "superhuman" (SP) behavior. This is achieved by counting the number of instances whereby the total distance covered in a single time step is beyond a cutoff value. Taking into account signs changes, each trajectory score can be computed as follows:

$$Trajectory\ Score\ = \begin{cases} PS * (1 + DR) + SP, & PS < 0 \\ PS * (2 - DR) + SP, & PS \geq 0 \end{cases}$$

The resultant trajectory score (TS), maintains similar intuitive meaning to PS. A more negative score indicates a better overall trajectory and typically falls in the range between -2 and 2.

5.2.2 Scoring multiple defender trajectories within a play

To aggregate multiple trajectory scores into a single play score that reflects the effective quality of defensive play, players are exponentially weighted based on their distance to the targeted receiver. Players in a net effective radius contribute the most to the aggregated score. A highly-scored play is shown below in Figure 14, with defenders (in black) actively converging towards the targeted receiver (in blue).

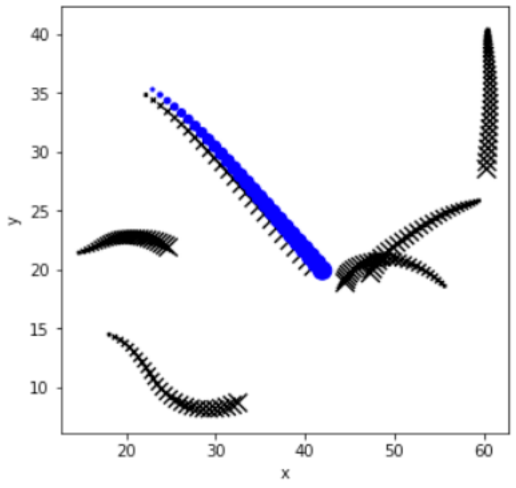


Figure 14: Example of a highly scored play with an aggregated score of -1.175.

5.2.3 Scoring model performance

The quality of all the predicted trajectories can be compared either via averaging the distance-weighted play scores or via averaging all defender trajectories. Comparisons can be made between actual trajectories and model-predicted trajectories, or when actual trajectories are not present in what-if situations. We also sanity checked the scores by replacing a fraction of all trajectories with bad trajectories (randomized initial direction and speed so one in four defenders moves non-optimally), and the scores deteriorated as expected by becoming more positive.

Trajectories	Average Trajectory Score (TS)
Actual, real trajectories	-0.1036
Traditional LSTM-CNN model (Predicted trajectories)	-0.0825
Bad model (25% bad trajectories)	0.0452

5. Discussion and Conclusion

In this paper, we designed and trained CNN-LSTM based models to predict defender trajectories from pass forward to pass arrived. This resulted in a 20% improvement compared to the baseline LSTM model. The 1D-CNN with multiple kernel sizes extracted relevant features from multiple inputs over several time steps, while pooling layers enabled us to limit model size and reduce overfitting. We additionally presented a new social pooling CNN-LSTM model that accounted for ball movement, targeted receiver, *and* nearby players' movements. The approach of combining of player movements and known important features via modified social pooling is one that we have not seen reported in existing literature, and outperforms state-of-the-art methods applied to time-series sensor data (ball and players' data) collected from NFL games. Both performant models are simple and shallow compared to modern vision-based or natural-language-based models, and were specifically designed as an effective tradeoff between complexity and data availability (2-season long real sensor data).

As shown in the sample plays in Figures 7-9, the model is not simply intuiting future trajectories based on current movements of the defender, but is able to adjust and produce realistic trajectories when the targeted receiver is changed. In short, the model successfully understands and captures the defensive responsibility of chasing and stopping the targeted receiver. All plays in the dataset were also reviewed by a football expert within NFL, and the expert determined that the majority of the predicted trajectories were sound. As expected, performance suffers slightly on very long plays compared to plays of normal duration. A few predicted trajectories were flagged as potentially problematic, and presented as less than ideal convergence towards the targeted receiver. Further investigation indicated that the non-optimal orientation and speed of the defender limited the defender's ability to respond.

We also presented metrics capable of evaluating defender trajectories, independent of a reference 'golden' trajectory. This enabled us to quantitatively evaluate large numbers of trajectories and plays, as well as evaluate what-if trajectories' performance. The metrics are also useful in automatically flagging and identifying plays and trajectories that are noisy and problematic.

References

[1] V. Ramanathan A. Robicquet L. Fei-Fei A. Alahi, K. Goel and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 961–971, 2016.
[2] L. Fei-Fei S. Savarese A. Gupta, J. Johnson and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2255–2264, 2018.
[3] J. Wu-J. B. Tenenbaum C. Sun, P. Karlsson and K. Murphy. Stochastic prediction of multi-agent interactions from partial observations. In arXiv preprint, 2019.
[4] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. pages 602–610.
[5] A. Alahi F. Palmieri L. Ballan, F. Castaldo and S. Savarese. Knowledge transfer for scenespecific motion prediction. In Proceedings of the European Conference on Computer Vision, 2016.
[6] F. A. Palmieri L. Ballan A. Alahi P. Coscia, F. Castaldo and S. Savarese. Point-based path prediction from polar histograms. In Proceedings of the International Conference on Information Fusion (FUSION), 2016.
[7] L. Li X. Zeng W. Hu, N. Xie and S. Maybank. A survey on visual content-based video indexing and retrieval. In IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2011.
[8] D. Lee K. Kim and I. Essa. Gaussian process regression flow for analysis of motion trajectories. In Proceedings of the International Conference on Computer Vision, 2011.
[9] J. A. Bagnell K. M. Kitani, B. D. Ziebart and M. Hebert. Activity forecasting. In Proceedings of the European Conference on Computer Vision, 2012.
[10] B. T. Morris and M. M. Trivedi. Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. In IEEE transactions on pattern analysis and machine intelligence, pages 2287–2301.
[11] X. Wang B. Zhou and X. Tang. Random field topic model for semantic region analysis in crowded scenes from tracklets. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.
[12] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. In Physical review E, volume 51, page 4282, 1995.
[13] G. D. Tipaldi M. Luber, J. A. Stork and K. O. Arras. People tracking with human motion predictions from social forces. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 464–469, 2010.
[14] A. Kuznetsova B. Rosenhahn L. Leal-Taixe, M. Fenzi and S. Savarese. Learning an imagebased motion context for multiple people tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3542–3549, 2014.
[15] G. Pons-Moll L. Leal-Taixe and B. Rosenhahn. Everybody needs somebody: modeling social and grouping behavior on a linear programming multiple people tracker. In Proceedings of the IEEE international conference on computer vision workshops (ICCV workshops), pages 120–127, 2011.
[16] A. Oyama R. Mehran and M. Shah. Abnormal crowd behavior detection using social force model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 935–4942, 2009.
[17] A. Ess S. Pellegrini and L. Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In Proceedings of the European conference on computer vision, pages 9452–465, 2010.

[18] L. E. Ortiz K. Yamaguchi, A. C. Berg and T. L. Berg. Who are you with and where are you going? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1345–1352, 2011.
[19] M. K. C. Tay and C. Laugier. Modelling smooth paths using gaussian processes. In Field and Service Robotics, pages 381–390, 2008.
[20] D. J. Fleet J. M. Wang and A. Hertzmann. Gaussian process dynamical models for human motion. In IEEE transactions on pattern analysis and machine intelligence, pages 283–298, 2007.
[21] L. Dinh K. Goel A. C. Courville J. Chung, K. Kastner and Y. Bengio. A recurrent latent variable model for sequential data. In Proceedings of the Advances in neural information processing systems, pages 2980–2988, 2015.
[22] K. Cho J. Chorowski, D. Bahdanau and Y. Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. In arXiv preprint, 2014.
[23] A. Joulin A. Karpathy and L. F. Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. In Proceedings of the Advances in neural information processing systems, pages 1889–1897, 2014.
[24] S. Bengio O. Vinyals, A. Toshev and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3156–3164, year =2015.
[25] R. Kiros K. Cho A. Courville R. Salakhudinov R. Zemel K. Xu, J. Ba and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International conference on machine learning, pages 2048–2057, 2015.