NFLNet: A hard hitting evaluation of deep learning approaches to tackle prediction

Kiernan McGuigan, Lily de Loë, Joshua Kurien, Qasim Ali
Vision and Image Processing Research Group, Systems Design Engineering, University of Waterloo
{kmcguiga, lcbdeloe, j2kurien, m45ali}@uwaterloo.ca

Abstract

This paper presents four architectures for predicting tackle probability in an NFL football game. Accuracy, precision, recall, loss, and F1 scores are compared to identify the best classification model for the 2024 NFL Big Data Bowl. The models leverage NFL tacking data, including player position, speed, direction, and location relative to key field markers. Tracking information was processed to extract meaningful plays, determine which features should be used in the solution, and identify plays with successful and unsuccessful tackle outcomes. A feed-forward network is presented as a baseline, and the performance of a convolutional neural network, transformer, and graph transformer are compared. The feed-forward network yielded an accuracy of 75%, which establishes the minimum accuracy of a simple architecture that uses minimal features. The convolutional network outperformed the baseline with an accuracy of 85%, but performed worse than the transformer and graph transformer, which achieved accuracy results of 90% and 92%, respectively. Ultimately, the graph transformer is found to be most effective at predicting the tackle probability for a league-average player.

1 Introduction

It is often said that offense wins games and defense wins championships, and while not as glamorous as its high scoring counterpart, a strong defense is critical to a solid football team. Despite the importance of defenders, offensive players have historically received more attention when it comes to developing novel metrics to analyze their performance [1, 2]. In response to this imbalance in analytical focus, this paper presents a solution to the 2024 NFL Big Data Bowl: a Kaggle competition where entrants propose novel metrics for tackling, which National Football League (NFL) teams can use to inform strategy [3].

Using data from the 2022 NFL season, we compare four deep learning models that predict the probability of a tackle occurring at different points within a play. Models accept the past one second of play data and must classify if a tackle will occur within the next half-second. The best-performing model must, to a high degree of accuracy, be able to determine the combination of player position, speed, and orientation that should result in a successful tackle. By comparing the expected and observed outcomes, this metric will allow teams to compare if a player's tackling performance was above, below, or equivalent to the league average. This strategy can be extended by looking at the difference between outcomes and expectations over a larger sample size of plays to identify scenarios in which a player's performance deviates from the league average. This task is best addressed by what we refer to as an "expectation model", which predicts the "correct" outcome based on the league average. Consequently, the proposed model cannot be extremely accurate (i.e., exceeding 99% accuracy) because it may indicate one of two things: either every player always plays exactly like the league average, which is not true, or the model is over-fitting and accounting for player specific details. As a result, model success is verified by confirming that average accuracy and error are consistent over the training, validation, and test sets.

Sports analytics–which increasingly involves data-driven approaches–is crucial for NFL teams to maintain a competitive advantage. Analysis of historical data is necessary to develop strategies, assess performance, and improve player selection [4]. In turn, a classifier that predicts tackle probability based on the league average is useful for teams to identify both strong players and loose cannons–those who either make home run tackles or miss completely. A league average prediction determines the most probable outcome; a player's performance above or below this metric has the potential to shape game strategy and player selection.

2 Background

The four proposed deep learning models build on literature and submissions to previous Big Data Bowls. This study presents a novel metric, and uses a private dataset. Thus, in the absence of comparable models, we present a simple feed-forward network (FFN), which acts as a performance baseline. FFNs have been applied in previous Big Data Bowl submissions [5], achieving success with tasks such as tackle decomposition. Additionally, FFNs have also been applied to football for predicting game outcomes and developing player-specific movement attributes, which inform play success [6], [7].

Convolutional neural networks (CNNs) perform well on image and spatial data [8]. For tackle prediction, CNNs are an appropriate model choice because they can utilize the inherent spatial inductive biases. Specifically, the data can be modeled spatially – each player's actions are largely influenced by the players nearest to them. Previous works have solved similar problems with CNNs; these include the winning solution for the 2020 Big Data Bowl, which predicted yards gained by the rusher during a running play [9]. Further, other works (including [10] and [11]) utilized convolutional models to predict the type of offensive and defensive coverage a team employed. These works demonstrate the effectiveness of CNNs for analyzing spatio-temporal NFL data.

Transformers are the current state-of-the-art architecture for long-term sequence-to-sequence modeling [12]. NFL tracking data can be used as a temporal sequence, where the model predicts tackles in future frames. Thus, transformers are a viable architecture for this problem. Transformers have also been used to solve related problems, such as soccer matches [13]. We employ a similar approach to [13], where a transformer predicts the next event and event location in a soccer game. However, applying transformers directly by treating each frame as a token ignores the spatial relationships between the players of a given frame. Previous works propose spatio-temporal transformers to address this limitation [14].

Alternatively, each frame in a sequence can be treated as a graph neural network (GNN) rather than a single large feature embedding, with the GNN being used as a token [14]. GNNs are a class of neural networks designed to work on graph-structured data. Nodes are used to encode observations, typically represented with a vector. Edges encode relationships between nodes, and may or may not contain feature information of their own [15]. Graphs are an efficient and meaningful representation format for non-euclidean data that is difficult to model with a regular grid. Existing works have applied graph neural networks to diverse fields such as citation mining or protein-protein interactions [16]. The structure of NFL play data can be considered irregular due to the intricate interactions between individual players. To address the tackle prediction problem, extracting and disseminating features from nearby players would be beneficial. Using a traditional GNN, however, would ignore the temporal relationship between the frames. Other works have proposed layers to leverage such spatio-temporal data [17], [18].

3 Methodology

The development of the tackle prediction model started with dataset preprocessing 3.1 to ensure compatibility with model requirements. Four distinct architectures were then hyperparameter tuned, trained, and evaluated on their ability to predict tackles.

3.1 Dataset

Data from the 2022 season is provided by the NFL Next Gen Stats team and Pro Football Focus [3]. The dataset includes plays where a rush (the offense runs with the ball), scramble (the quarterback makes an impromptu evasive maneuver), or completion (a forward pass succeeds) occurs. Tracking data is comprised of 66 features, which track defensive contact events and player statistics by game, play, and player ID [3].

Only a subset of features relate to our metric, so exploratory data analysis (EDA) guided feature selection. Game footage was compared to the data, and event flags were found in the frame preceding the tackle. We analyzed the 28 event types, noting event flags before tackles, and cropped plays to start at the 'pass_outcome_caught' flag, ensuring the ball carrier always has possession. Plays with multiple carriers or rare events, like scrambles, were excluded. Eighteen meaningful events were selected, and plays were mirrored so the offensive team always moves right, avoiding directional bias.

Out of 66 features, ten core features were chosen: game ID, play ID, NFL ID, ball carrier ID, play frame, X and Y position, speed, direction, and event flag. Twelve additional metrics were calculated per player: distance to the ball carrier, distance to the line of scrimmage, distance to the first down marker, distance from a team's end zone, distance from the opposing team's end zone, X and Y components of direction, X and Y components of speed, change in X and Y position, and location relative to the ball carrier. A combination of these features is used in the CNN (3.3), transformer (3.4), and graph-transformer (3.5) architectures. Distance, position, and orientation features are required for models that frame

tracking information relative to the ball carrier. Relative distance features are necessary because strategy can differ based on proximity to the line of scrimmage, first down marker, and end zone.

For classification, play sequences were identified as positive or negative based on the tackle outcome, with models using 10-frame inputs starting 15 frames from the ground truth. Positive events are plays that end in a successful tackle, while negative events end in first contact. First-contact events were chosen as unsuccessful tackles because the data would exhibit similar behaviour to a successful tackle. Here, the defense still converges on the ball carrier, making it more challenging to predict a tackle compared to a sequence where the defense is far behind. Models use a training:testing:validation split of 70:20:10, with positive and negative events being evenly distributed across the sets.

3.2 Feed-Forward Network

The feed-forward model uses the data format described in Section 3.1. A custom PyTorch data loader uses four features: x-position, y-position, speed, and direction. The feed-forward model acts as a performance baseline; thus, it uses the subset of features from the three other models, which are included in the competition dataset. X and Y position are scaled based on the maximum values for the dataset. Speed and distance use their mean and standard deviations for normalization. These processes are used to improve model performance and training stability. The final architecture is summarized in Table 1, and illustrated in Figure 1.

Table 1: Optimized architecture for the feed-forward network.

Parameter	Value
Batch size	1
Dropout probability	0.6
Number of layers	2
Batch normalization	True
Activation function	ReLU
Learning rate	e-6 to e-8
Scheduler	Lambda
Optimizer	ADAM
Loss	Binary Cross Entropy
Number of epochs	20

3.3 Convolutional Neural Network

The CNN architecture was inspired by the winner of the 2020 Big Data Bowl [9]. Additionally, the data processing was informed by [19], which modeled time series data as 2D images for convolutions. Here, play data was restructured to resemble a 2D image for the CNN, with an input shape of 12x22x10, where 12 represents player metrics, 22 the number of players, and 10 the frames in a sequence. The restructuring process begins with handling 2D/tabular features, such as position, which are 10x22 matrices representing player positions across frames. A custom PyTorch Dataset class transformed these matrices into 3D volumes, like multi-channel images, with each feature normalized. Position metrics used known maximum values (based on field dimensions), while others were normalized by their mean and variance. The 2D matrices were then concatenated into a 3D volume. Additionally, the 1D ballCarrier array, indicating which player had the ball, was converted into a one-hot encoded 10x22 matrix and concatenated with other features. The model's architecture is composed of two major sections: a series of convolutional blocks, followed by fully connected layers. The results of tuning are summarized in Table 2, and is illustrated in Figure 2.

Table 2: Optimized architecture for the convolutional neural network.

Parameter	Value
Batch size	2048
Dropout probability	0.2579
Number of conv layers	5
Kernel size	2
Batch normalization	True
Activation function	Tanh
Learning rate	0.0618
Scheduler	Plateau
Optimizer	ADAM
Loss	Binary Cross Entropy
Number of epochs	80

3.4 Transformers

The model (Figure 3) is based on the original transformer presented in [20], and is similar to the architecture in [13]. Each frame is treated as a unique token by concatenating the features of all 22 players in that frame. Our model begins by projecting high-dimensional tokens to a lower-dimensional subspace to boost computational efficiency and compression. A zero-initialized x_{class} token, used to predict tackles, is added to the 10 frames, followed by layer normalization. Positional information is added with sinusoidal non-learnable encodings. The model then uses a multi-head transformer encoder for attention across the 11 tokens. After encoding, the x_{class} token is fed into a 64-unit hidden layer, activated, and passed to a single neuron output layer. A Sigmoid activation function provides the probability of a tackle in

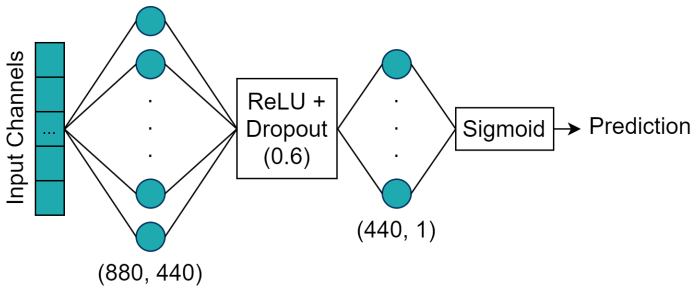

Figure 1: Diagram of the feed-forward network. A sigmoid activation function provides the probability of a tackle in the next frame.

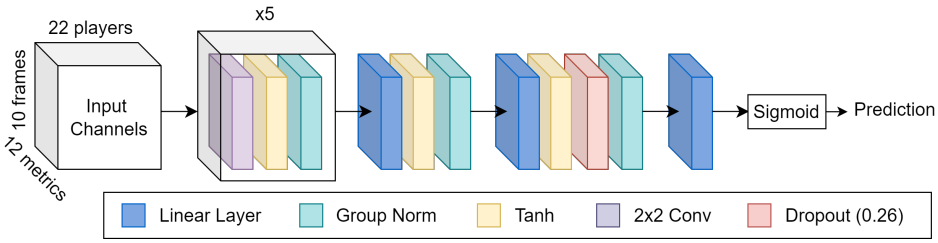

Figure 2: Diagram of the convolutional neural network. A sigmoid activation function provides the probability of a tackle in the next frame.

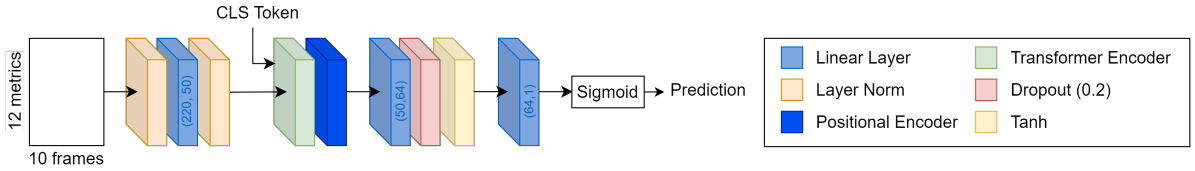

Figure 3: Diagram of the transformer model. A sigmoid activation function provides the probability of a tackle in the next frame.

the next frame. The optimized architecture is summarized in Table 3

Table 3: Optimized architecture for the transformer.

Parameter	Value
Dropout probability	0.29
Number of layers	2
Dimension of feed forward layer	256
Number of heads in attention layer	5
Dimension of tokens	50
Activation function	Tanh
Learning rate	2e-5
Weight decay	9e-3
Optimizer	ADAMW
Loss	Binary Cross Entropy
Number of epochs	20

3.5 Graph Transformers

Building on the concept of graph neural networks (Section 2), the final proposed model is a graph transformer. This architecture (Figure 4) required modifications to the play data, focusing on the field state half a second before tackle prediction. All player features and inter-player features (e.g., distances, relative positions) are included if players are within 12 yards, limiting edge growth. Graphs are batched with eight disconnected components and passed through spatial blocks implementing modified graph attention. Self-loops are removed, replaced by a linear transformation on each node, and skip connections allow for selective node updates. Nodes are aggregated by feature channel, concatenated, and fed to a projection head for tackle/no-tackle prediction. Hyperparameters and architecture specifications such as node and edge dimensions are summarized in Table 4.

Table 4: Optimized architecture for the graph transformer.

Parameter	Value
Number of graph attention layers	4
Node dimensions	16
Edge dimensions	5
Learning rate	0.01
Optimizer	ADAM
Loss	Binary Cross Entropy
Number of epochs	Early Stopping

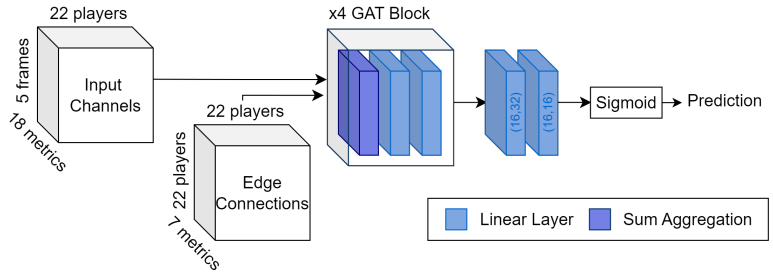

Figure 4: Diagram of the graph transformer architecture. Spatial blocks employ a graph attention layer, which uses edge embeddings to determine attention scores between nodes and an additional self-attention component created from a linear layer. The embedded nodes are aggregated as both a mean and max over the entire graph, and the concatenated results are fed into a projection layer to get a single predicted value. A sigmoid activation function provides the probability of a tackle in the next frame.

4 Results

A summary of model performance is provided in Table 5. Tackle projection is framed as a binary classification problem; thus, Binary Cross-Entropy Loss, Accuracy, Precision, Recall, and F1-Score were selected as performance metrics. These metrics are computed and reported for the test dataset.

Table 5: Performance of each model across various metrics on the Test dataset.

Metric	FFN	CNN	Transf	GTransf
Cross-Entropy Loss	0.51	0.36	0.40	0.20
Accuracy	0.75	0.85	0.90	0.92
Precision	0.80	0.86	0.91	0.95
Recall	0.83	0.92	0.93	0.92
F1 Score	0.82	0.89	0.92	0.93

4.1 Feed Forward Network

The feed-forward model was not expected to yield accuracy as high as the other models and was meant to provide a baseline for which to compare them. Notably, tuning was required to prevent the model from overfitting as a result of network depth, batch size, and learning rate. The model converged at 73%, 74%, and 75% for the respective training, validation, and test datasets. The loss was similarly comparable (0.51,0.52, and 0.51). The test set reported a precision of 80.23%, recall of 83.00%, and f1 score of 81.59%. These results are reiterated in Table 5.

4.2 Convolutional Neural Network

Table 5 summarizes the performance metrics for the tuned network compared to the baseline FFN. During tuning, the convolutional neural network with the best validation loss after tuning was selected. A combination of the hyperbolic tangent activation function, low dropout probability, normalization, a learning rate scheduler, and increased network depth improved performance.

When comparing the classification metrics, the CNN performed better than the feed-forward network, but underperformed relative to the transformer and graphs networks. Notably, the metrics across splits for the network were relatively similar. The train, validation, and test loss were all approximately 0.36 (0.368, 0.368, and 0.365 respectively). Additionally, accuracies were similar across train, validation, and test splits, all close to 85% (85%, 83%, and 85% respectively). This is in line with the goal of creating a model that has predictive capability but does not overfit.

4.3 Transformer

The results of the Optuna study (Table 3), revealed that a combination of the tanh activation function, lower learning rates, higher weight decay, and low dropout rates helped the model train. Furthermore, smaller token dimensions, fewer encoder layers, and fewer attention heads help the model achieve better results. The Optuna run with the lowest validation loss was selected as the final transformer model. It far outperformed the feed-forward baseline, achieving an accuracy of 90%. Further, the test set produced nearly equal precision, recall, and f1 score (all over 90%), indicating that the model is effective at both tackle and non-tackle event prediction. Accuracies were similar across train, validation, and test splits, all close to 90% (95%, 90%, and 90% respectively). This is in line with the goal of creating a model that has predictive capability but does not overfit.

4.4 Graph Transformer

The hyperparameter tuning results (Table 4) showed that smaller node dimensions, no edge updating, and the use of five-dimensional edge dimensions achieved

the best performance. The graph transformer was the most successful architecture. This was expected due to its ability to effectively process spatial information. The graph transformer achieved a test accuracy of 92% and performed similarly for training and validation (91% and 92%). Further, it achieved recall, precision, and F1 scores of 92%, 95%, and 93%, respectively. Since these results were found on the test set and were comparatively similar to those in the validation and training set, it is fair to conclude that the graph transformer was successful at generating play expectations given the first contact and the tackle frames that were provided.

5 Discussion

Overall, all three advanced models (CNNs, Transformers, and Graph Transformers) outperformed the FFN baseline (Table 5). These models achieved high performance on the test dataset. Further, accuracy, precision, and recall indicate that the models can predict what player and team configurations lead to successful tackles. All models performed equivalently over the training, test, and validation sets, verifying that these accuracies do not result from overfitting.

5.1 Limitations of the model

As anticipated, the feed-forward network performed poorly, failing to achieve "reasonable accuracy" (Section 1). These models lack the structure to capture spatial sequences, making them sensitive to noise and prone to overfitting. Additionally, the model used a limited feature set without the ball-carrier references, which proved essential for understanding play dynamics. Enhancing the feature space to include relative positioning could improve results; however, the model will likely continue to under-perform relative to the other three architectures.

The convolutional neural network could benefit from structuring 2D features with spatial and temporal locality, as well as incorporating ball-carrier-relative metrics. Use of advanced architectures like ResNets [21], which feature residual connections, may also enhance performance. The Transformer exploited temporal but not spatial data, treating each frame as a single token without considering individual players. Future work could explore spatio-temporal transformers (e.g., [14] and [22]) to capture within-frame spatial interactions. Finally, Graph Transformers modeled spatial information well, with players as nodes and edges between close players. Their success suggests spatial data may be more critical than temporal data for this problem. However, they only used field state a half second before tackle prediction, missing temporal player movements. Adding temporal data and expanding the edge cutoff could improve performance, capturing relationships like that between a quarterback and receiver.

Overall, none of the models fully leveraged both spatial and temporal information, limiting their ability to capture complex player interactions, such as defenders positioning to block escape routes. Modeling these interactions over time would likely yield stronger insights from player tracking data.

5.2 Limitations in Data

Data quality in the 2024 Big Data Bowl presented a major limitation due to missing and mislabeled plays. While we removed these plays from the dataset, individual plays must be compared to game footage to verify that the correct flag was applied. Thus, it is likely that a subset of the plays are incorrectly flagged. Additionally, the classification of tackle events is dependent on human interpretation. Inconsistencies in the flagging of when a tackle event begins and differences in tackle length both contribute to ambiguities in data labels.

5.3 Conclusions

In this paper, we assess the performance of four deep learning models, which predict the probability of a tackle event given a second of position, direction, and speed data for both offensive and defensive NFL players. Three potential models (a convolutional neural network, transformer, and graph transformer) were compared against a baseline feed-forward network. Ultimately, the graph transformer was found to be most effective, indicating that the spatial connectivity of other players in relation to the ball carrier was key to determining the likelihood that a tackle would happen. The transformer achieved the next best performance, demonstrating the importance of temporal information within the plays. These networks are capable of successfully predicting league-average performance for post-game analysis of tackles. As a result, NFL teams can assess the performance of a defensive player by comparing the actual play to the model's prediction.

Acknowledgments

Thank you to the National Football League for providing this dataset, and to Dr. Bryan Tripp for supervising our team during this project.

References

[1] R. Yurko, S. Ventura, and M. Horowitz, "nflwar: A reproducible method for offensive player evaluation in football," 2018. [Online]. Available: https://arxiv.org/abs/1802.00998

[2] F.-X. Aubet and E. Ehrlich, "The science behind nfl next gen stats' new passing metric," Jun 2023. [Online]. Available: https://www.amazon.science/blog/the-science-behind-nfl-next-gen-stats-new-passing-metric

[3] T. N. F. League, "Nfl big data bowl 2024," https://www.kaggle.com/competitions/nfl-big-data-bowl-2024/overview, accessed: 2023-12-01.

[4] Y. F. Roumani, "Sports analytics in the nfl: classifying the winner of the superbowl," *Annals of operations research*, vol. 325, no. 1, pp. 715–730, 2023. [Online]. Available: http://dx.doi.org/10.1016/j.metrad.2023.100017

[5] H. Liu, X. Lu, and M. Qitai, "Decompose tackling, a neural network approach," https://www.kaggle.com/code/hongzeliu7/decompose-tackling-a-neural-network-approach/notebook.

[6] L. Boll, "Gridiron genius: Using neural networks to predict college football," https://deepblue.lib.umich.edu/bitstream/handle/2027.42/176935/Luke_Boll_Honors_Capstone_Report_-_Luke_Boll.pdf?sequence=1.

[7] H. Horn, E. Laigaie, A. Lopez, and S. Reddy, "Using geographic information to explore player-specific movement and its effects on play success in the nfl," *SMU Scholar*, vol. 7, no. 2, p. 22, 2023. [Online]. Available: https://scholar.smu.edu/cgi/viewcontent.cgi?article=1249&context=datasciencereview

[8] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.

[9] P. S. Dmitry Gordeev, "Nfl big data bowl 2020: The zoo," https://www.kaggle.com/competitions/nfl-big-data-bowl-2020/discussion/119400, accessed: 2023-12-01.

[10] H. Song, M. A. Jazaery, H. Ding, L. L. Cheong, J. Jung, M. Band, M. Chi, and T. Bliss, "Explainable defense coverage classification in nfl games using deep neural networks," in *MIT Sloan Sports Analytics Conference 2023*, 2023. [Online]. Available: https://www.amazon.science/publications/explainable-defense-coverage-classification-in-\nfl-games-using-deep-neural-networks

[11] J. Newman, A. Sumsion, S. Torrie, and D.-J. Lee, "Automated pre-play analysis of american football formations using deep learning," *Electronics*, vol. 12, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/3/726

[12] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, Z. Wu, L. Zhao, D. Zhu, X. Li, N. Qiang, D. Shen, T. Liu, and B. Ge, "Summary of chatgpt-related research and perspective towards the future of large language models," *Meta-Radiology*, vol. 1, no. 2, p. 100017, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.metrad.2023.100017

[13] I. Simpson, R. J. Beal, D. Locke, and T. J. Norman, "Seq2event: Learning the language of soccer using transformer-based match event prediction," p. 3898–3908, 2022, accessed: 2023-12-01. [Online]. Available: https://doi.org/10.1145/3534678.3539138

[14] J. Grigsby, Z. Wang, N. Nguyen, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," 2023.

[15] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2021.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[17] P. Chu, J. Wang, Q. You, H. Ling, and Z. Liu, "Transmot: Spatial-temporal graph transformer for multiple object tracking," *CoRR*, vol. abs/2104.00194, 2021. [Online]. Available: https://arxiv.org/abs/2104.00194

[18] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting," *CoRR*, vol. abs/1709.04875, 2017. [Online]. Available: http://arxiv.org/abs/1709.04875

[19] A. N. Sayed, Y. Himeur, and F. Bensaali, "From time-series to 2d images for building

occupancy prediction using deep transfer learning," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105786, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095219762200776X

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[22] H. Yan, X. Ma, and Z. Pu, "Learning dynamic and hierarchical traffic spatiotemporal features with transformer," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22 386–22 399, 2022.