A Deep Learning Approach to Predicting Pass Completion in Soccer

Ashvin Ramabadran

Abstract

The motivation behind this project is to develop a tool that soccer teams can use for tactical analysis and scouting purposes. Passing is at the heart of the game of soccer, and accurately modeling a team's likelihood of completing a pass can grant insight into a team's play style, as well as the natural ability of their players. Such information can then be used by the team and their opponents to formulate tactical strategies that maximize or counter the observed trends during a match. The team itself can use this encapsulation of their quality to help build developmental programs and inform recruitment decisions.

Dataset

The increase usage of analytics to gain a competitive edge in sports has spurred the need for a more granular level of data points. Sports data provider StatsBomb has open sourced soccer data at the individual event level. This means we can see a singular pass made during a match. The specific data used for this the project was passing event data for Chelsea Football Club during the 2015/16 Premier League season. Essentially, the cleaned dataset consists of almost every pass made by a Chelsea player across their 38 Premier League matches that year. Some samples were dropped if the player attempting the pass had a low amount of total attempted passes over the course of the season(I set minimum pass attempts required to 200). This prevents the model from having to learn about a player that there is not much data about. The features of the model state the player making the pass, and they describe the characteristics of the attempted pass.

The features of the model are: player identification number, angle of the pass, length of the pass, the x and y location of where the pass was attempted, the x and y location of where the pass ended, if the player was under pressure(yes or no), the type of the pass(cross, cut back, through ball, switch, inswinger outswinger, normal), the pattern of play the pass was attempted during(corner kick, counter attack, free kick, kick off, throw in, goal kick, directly from

goalkeeper, regular), the foot the pass was attempted with(left or right), the height level of the pass(ground, low, high).

The label of the model is whether the pass was completed or not(yes or no).

Model Architecture

The neural network architecture designed was a fully connected network. The player identification numbers are passed through an embedding layer to bake in hidden context that the model learns about each player. The outputted embeddings are then concatenated with the standard features to form the input into the first linear layer. This layer has a width of input_size and maps to a space of input_size * 1.25. Outputs are then passed through a ReLU activation function. Dropout is then applied as data flows into the next linear layer of width input_size * 1.25 and maps to a space of input_size * 1.5. Outputs are again passed through a ReLU activation function. Dropout is again applied as data continues to flow into the next linear layer of width input_size * 1.5 and maps to a space of input_size * 1.75. Once again, outputs are passed through a ReLU activation function. Dropout is once again applied as data flows into the final linear layer of width input_size * 1.75 and maps to a space of 2 for the two classes, pass completed or pass incomplete. The outputted logits from the final layer are then passed into a cross entropy loss function, and the whole model is optimized using the Adam optimizer.

Training

Due to how soccer is naturally played, there are many more completed passes than incomplete passes over the course of a match and thus on the season overall. In the cleaned dataset, there are 15,869 samples of completed passes, and only 2,764 samples of incomplete passes. This creates an issue where the model can learn to excessively predict the majority class and still achieve a low loss/high accuracy. To account for this class imbalance during the training phase, the model was trained using stratified k-fold cross validation. A stratified sampling of the data points for each fold allows for the relative proportion of each class in the dataset to be reflected in each fold of the partition. This creates a better training environment for the model as it can learn from data that looks similar to how it naturally appears, opposed to training and subsequently testing on folds that relatively overrepresent or underrepresent the classes. To ensure

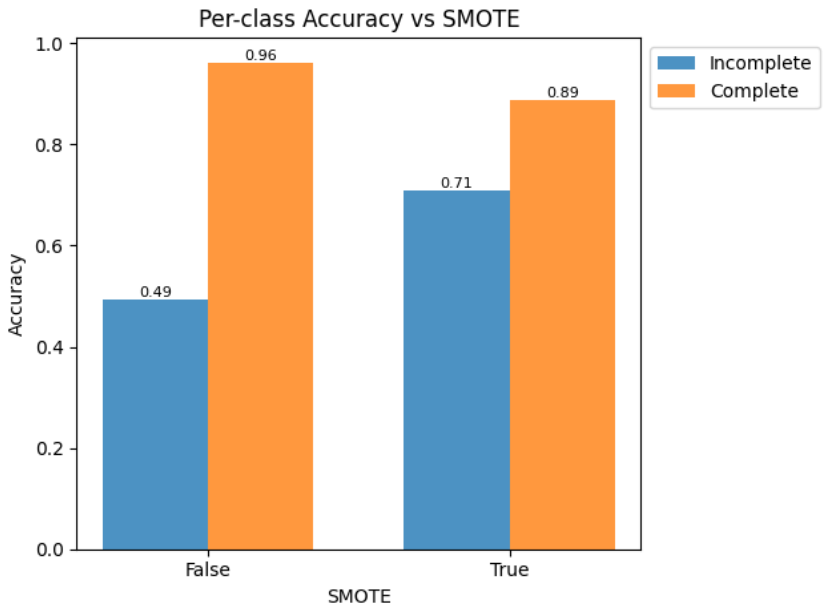
balanced performance across both classes, testing accuracy was averaged across all folds and reported on a per class basis.

Experiments

Applying SMOTE

To further address the class imbalance, I experimented with applying synthetic minority oversampling technique to generate artificial samples of incomplete passes. SMOTE goes through each minority class sample C, finds C's k nearest neighbors(I let it default to k=5), chooses one neighbor sample N at random, computes the difference between the features of C and N, multiples that result by a random float between 0 and 1, and adds this result to C's features to generate the artificial sample for C. This process is repeated until there is a balanced distribution of samples across both classes. The specific challenge with my data was that it makes sense to generate artificial data of the standard features that describe the characteristics of a pass, but not artificial players. As a result, SMOTE was applied on a a per-player basis by separating the player id from the standard features. I then looped over each player id and ran SMOTE on samples for the specific player, concatenating the results across all players in the end.

As shown in the figure, without SMOTE, the model can achieve near perfect accuracy on the majority class, completed passes. However, it performs basically at a random guessing level on incomplete pass samples. When SMOTE is applied, there is a decrease in performance in completed pass samples, but this a worthy tradeoff for the dramatic improvement in accuracy in identifying incomplete passes.

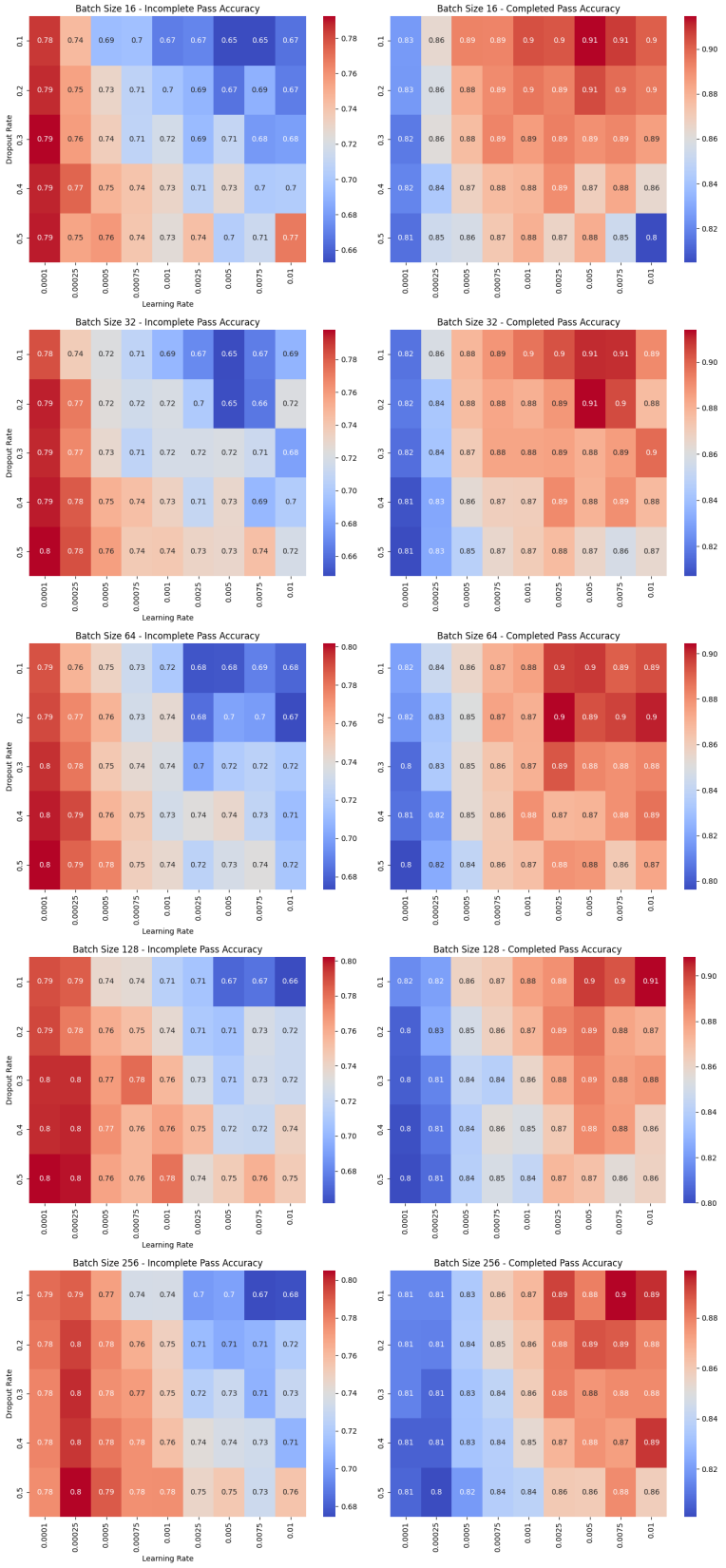


These results show that in order to have good performance on both classes, SMOTE is needed to give the model enough minority sample data to learn from.

Hyperparameter Search

Hyperparameters such as batch size, learning rate, and dropout rate can be interconnected. Resultantly, tuning them one by one in a linear manor will often not produce the optimal combination. Thus, I defined a range of common values for each hyperparameter and conducted a grid search over all possible combinations of these values.

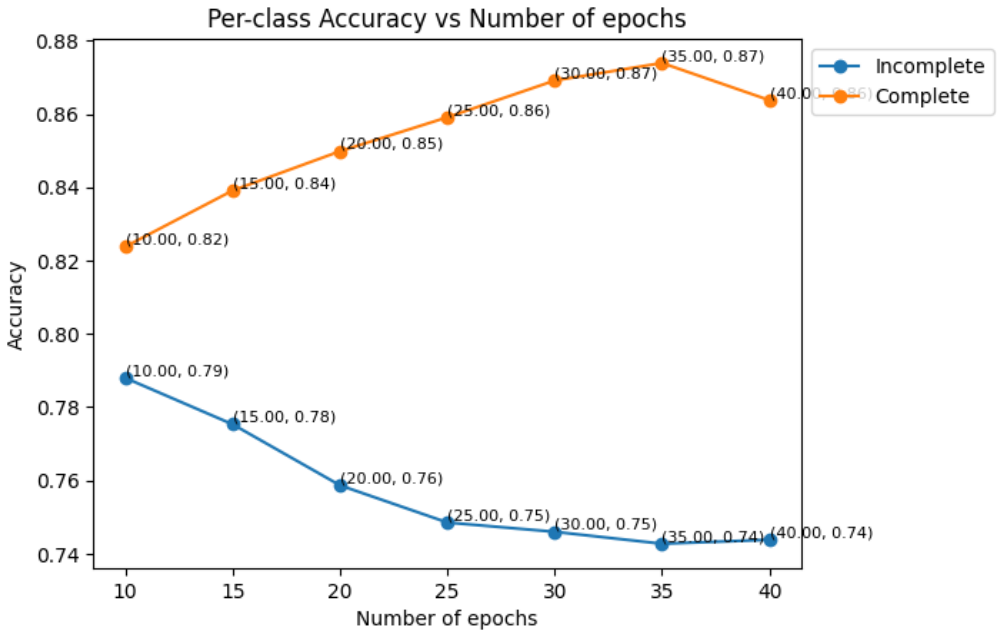
Each cell shows the accuracy on a particular class for a certain batch size-learning rate-dropout rate combination. When holding dropout rate constant, lower learning rates improved performance on incomplete pass samples, but higher learning rates improved performance on completed pass samples. When holding learning rate constant, higher dropout rates slightly improved performance on incomplete pass samples, but lower dropout rates slightly improved performance on completed pass samples. More regularization favoring incomplete pass samples likely implies dropout helps prevent the model from overfitting to the majority class. When holding learning rate and dropout rate constant, bigger batch sizes slightly improved performance on incomplete pass samples, but smaller batch sizes slightly improved performance on completed pass samples.



To maintain balanced performance on both classes, the final model was trained with a batch size of 256, a learning rate of 0.00075 and a dropout rate of 0.4.

Varying Number of Epochs

Gauging when the model has trained enough is an essential component of the model to tune. The danger of training for too many epochs is the model can begin to overfit to the training data, while too few epochs can prevent the model from learning complex patterns. I defined a range of epochs I was interested in testing and cross validated the model with those values.



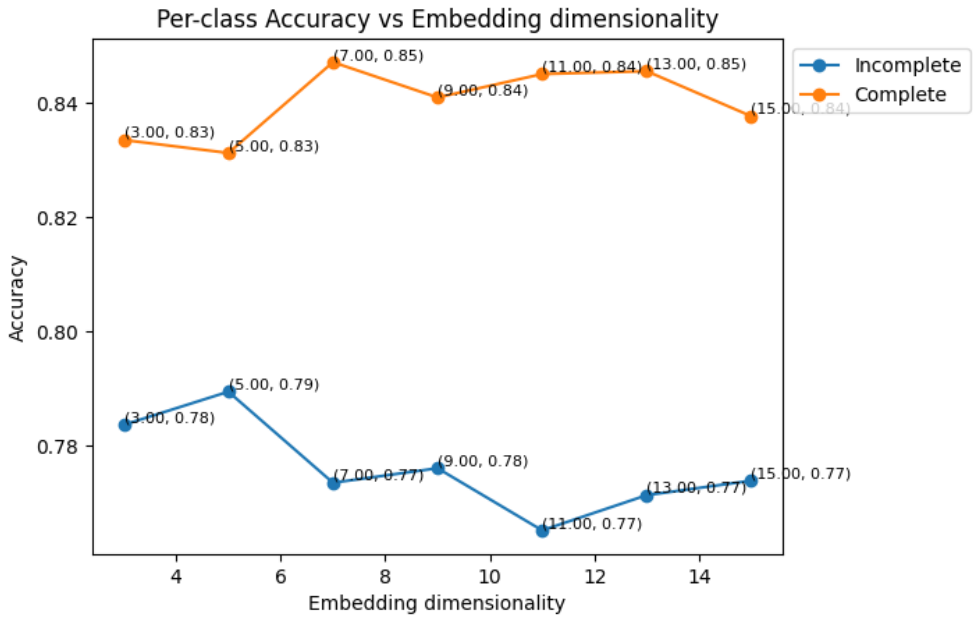
The figure shows that the more epochs the model is allowed to train for, accuracy on the majority class, completed passes, increases at the expense of accuracy on the minority class, incomplete passes. The reason training for too many epochs starts to degrade the performance on incomplete pass samples is mostly likely due to the fact that artificial samples of incomplete passes are being trained on after applying SMOTE. Excessive training on these artificial samples that do not perfectly represent natural incomplete passes can cause the model to overfit to the artificial samples and struggle with identifying the naturally occurring incomplete pass samples that are held out for testing.

To maintain balanced performance on both classes, 15 epochs was chosen to train the final model with.

Varying Embedding Layer Dimensionality

Embedding layers work by taking discrete categorical inputs and training a continuous vector encoding for each input. In my model, player id's are mapped

to their vector representations to encapsulate the hidden characteristics the model learns about each player. This in a way could represent a player's natural skill level. Hence, concatenating the embedding output with the standard features describing the context of the pass allows the model to consider the inherent talent of a player when determining if the pass is likely to be completed or not. Too little dimensions in the output may not properly capture the ability of players, while too many dimensions might overweigh the importance of players and fail to give the context of the pass enough consideration. The cleaned dataset has 19 players, so I defined a range of embedding layer dimensions that were about 15% to about 80% of this value and cross validated the model with each value.



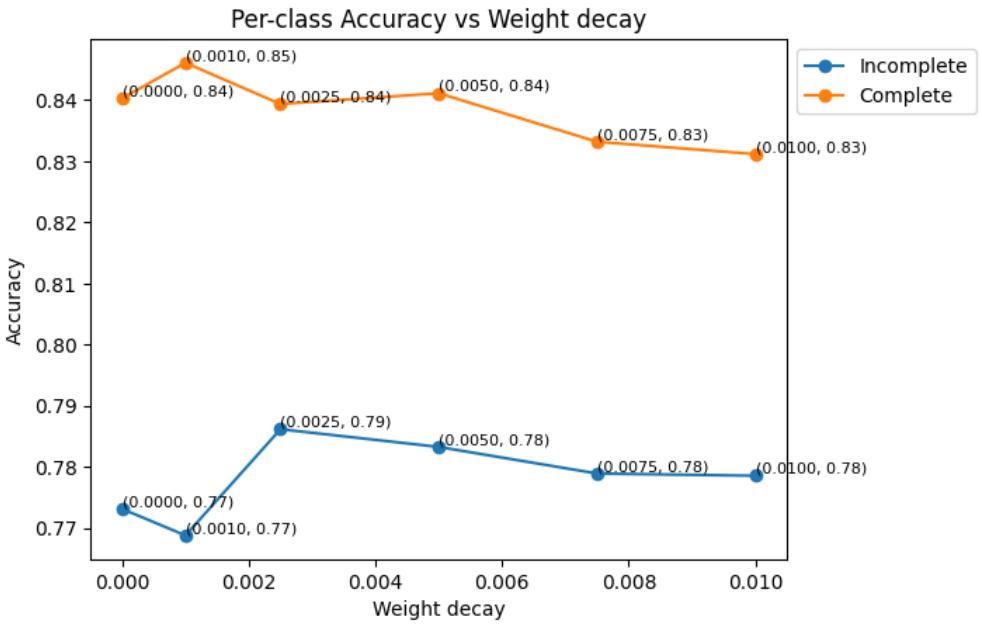
As seen in the figure, performance is quite similar across different number of dimensions in the embedding output. This could be a product of many reasons. The context of the pass could simply be a more relevant predictor of pass completion than the player attempting the pass. It is possible the lower dimensionality representation already captures enough of the player context and adding more dimensions has diminished returns. There might also just be a lack of data with respect to certain players to properly learn the influence a player can have on successfully completing a pass.

The final model was built with 13 dimensions in the embedding layer output.

Varying Amount of Weight Decay

Weight decay is a critical regularization technique to prevent the weights of the network from growing too large. This helps prevent overfitting by enforcing numerical stability in the weight values, and has the added benefit of avoiding the

exploding and vanishing gradient issues during training. There is a balancing act in how much is applied, as too much weight decay can make the model overly simple and thus underfit the data, while not enough weight decay can allow the model to become too complex and sensitive to noise in the training data. I defined a range of common weight decay values I was interested in testing and cross validated the model with those values.



As seen I the figure, performance is quite similar across varying levels of weight decay. This could be a product of many reasons. The use of dropout could already regulate the model complexity enough on its own to allow for sufficient generalizability, making weight decay insignificant. The Adam optimizer managing parameter updates during training can reduce the dependency on weight decay for convergence. The dataset itself could also be simple enough to the point where there is a lack of noise for weight decay to help filter out.

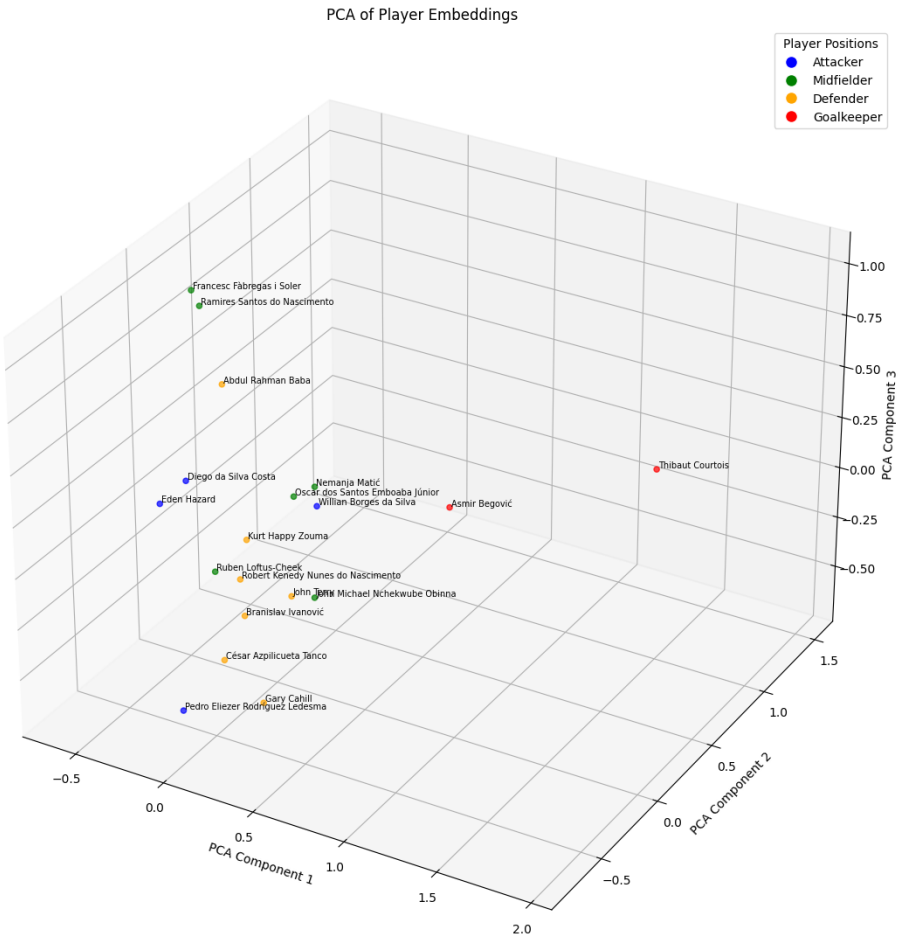
The final model was trained with a weight decay of 0.0025.

Evaluation

Embedding Analysis

The training of embedding vectors happens in a way that similar categorical data inputs are brought close to each other in geometric space. This means the representations the model learns about each player should be similar for players with comparable impact on pass completion. Principal Component Analysis is a way of reducing the dimensionality of data to the directions that capture the majority of the data's variance. PCA can be leveraged to reduce the embedding

output down to the three dimensions that reflect the most variance in the model's understanding of the players, and then the points can be plotted out.



The magnitude of the data points along each axis represent how much the players aligns with the pattern being captured by that principal component. What is most interesting in the plot is the observed clusters, which represent similarity in passing influence. The model has learned that Fabregas and Ramires have analogous control over pass completion, which makes sense because they both play the midfielder position, meaning they should share certain play style characteristics. Hazard and Costa are also near each other in the space, and they both fulfill attacking roles for the team. An evident pattern is that along the third principal component axis, almost all defenders fall very low, further emphasizing their strong positional group relationship.

Feature Visualization

Another way to interpret what the model has learned is to question what type of inputs drive certain units of the model. Each unit of the model contributes to capturing some aspect of the underlying pattern in judging whether a pass will be completed or not. What a particular unit has learned can be assessed by running the dataset through the trained model and inspecting the activations produced by that unit.

The tables shows the corresponding players of the samples with the top ten and bottom ten activations for the tenth unit in the first linear layer of the model architecture. High activation means the feature values of this sample strongly align with what this unit is detecting, while low activation means the feature values of this sample lack what this unit is detecting. What should be noticed is that despite the position of a player not being a feature the model considers, of the top ten activating samples, eight of them are of goalkeepers(Courtois and Begovic). This displays a pattern in the data of goalkeepers that this unit has been trained to emphasize in order to correctly predict pass completion. Conversely, Fabregas samples contain trends that go against this unit's emphasis on goalkeeper feature values. From domain knowledge, this makes perfect sense as goalkeepers during the 2015/16 season were not normally tasked with much passing responsibility. However, Fabregas was a midfielder, and midfielders often shoulder much of the passing responsibility for their team. This juxtaposition in roles is reflected in the top and bottom activations of this unit.

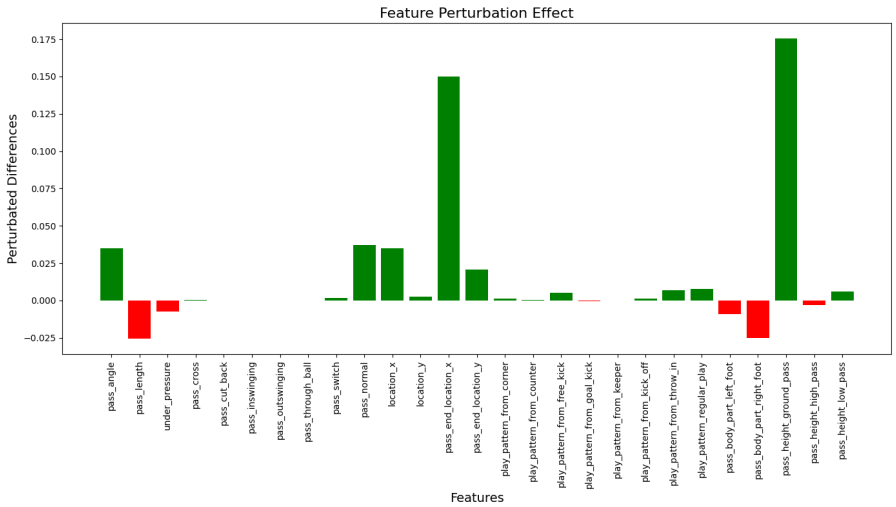
Player Name	Activation
Thibaut Courtois	1.160160
Asmir Begovic	1.150030
Thibaut Courtois	1.134221
Asmir Begovic	1.054928
Asmir Begovic	1.052382
Thibaut Courtois	1.048491
John Michael Nchekwube Obinna	1.044934
Cesar Azpilicueta Tanco	1.036970
Asmir Begovic	1.017302
Thibaut Courtois	1.002612

Player Name	Activation
Francesc Fabregas i Soler	-0.415007
Francesc Fabregas i Soler	-0.417828
Francesc Fabregas i Soler	-0.420969
Francesc Fabregas i Soler	-0.425386
Francesc Fabregas i Soler	-0.425658
Francesc Fabregas i Soler	-0.427616
Francesc Fabregas i Soler	-0.427954
Francesc Fabregas i Soler	-0.433965
Francesc Fabregas i Soler	-0.451398
Francesc Fabregas i Soler	-0.455231

Feature Perturbation

It is also important to see the impact features have in guiding the model towards the correct prediction of whether a pass will be completed or not. For a given sample, the model will output class scores, which can be converted into the probability that the model thinks this sample is a completed pass and the probability it thinks this sample is an incomplete pass. The prediction is then made by selecting the class with the higher probability. What can be done is that a sample can be passed through the model to get its probability on the true class. Subsequently, a particular feature can be removed(set value to zero), the sample can be ran through the model once again, and the probability the model gives to the sample being of the correct class now without this feature can be measured. This change in probability with and without the feature communicates the features importance to making a correct classification of the pass.

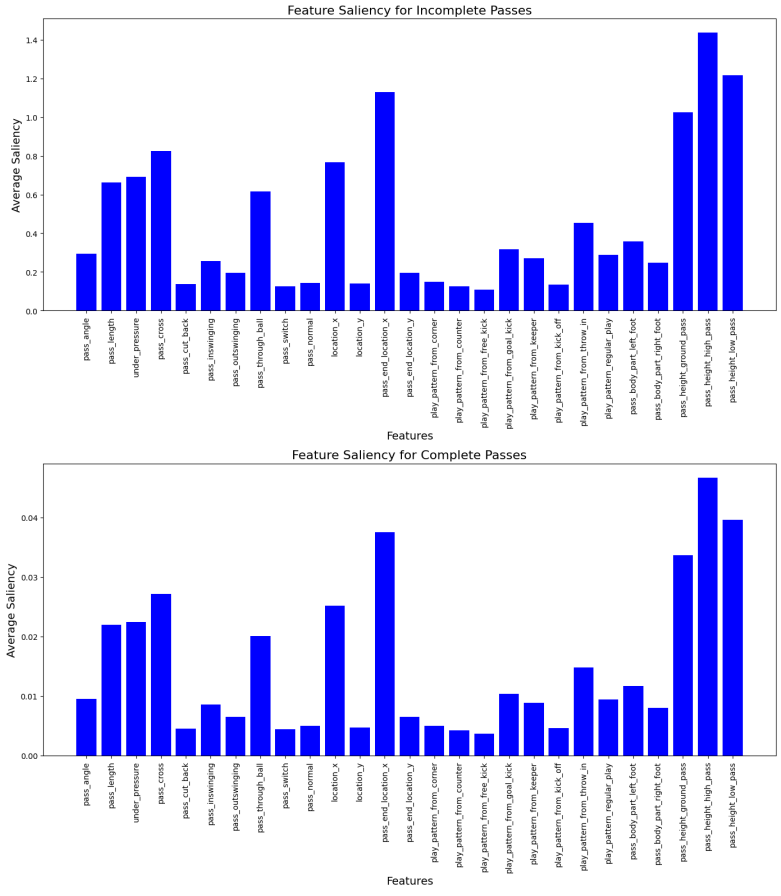
Green bars in the plot below indicate a positive difference between the true class probability with the feature and the true class probability without the feature. This means when this feature is removed from consideration, the model has a tougher time predicting the correct class. Red bars indicate a negative difference between the true class probability with the feature and the true class probability without the feature. This means when we remove this feature from consideration, the model actually gets more confident in predicting the correct class. The below plot shows that features like the x location of where a pass ends and the fact that a pass was kept on the ground have high perturbated differences, meaning they are critical to correctly predicting if a pass was completed or not. However, the negative perturbated differences for length of a pass and the fact that a pass was attempted with a player's right foot signal that the model performs better when ignoring those features.



Gradient Attribution

To train a neural network, the gradient of the output with respect to the input is calculated to guide the update of the model weights in a way that gets the output closer to a target output. To understand the impact a feature is having on the output, we can calculate the magnitude of the gradient of the output with respect to the feature, commonly referred to as the saliency of the feature. We can sum up these saliencies across all samples and average them to get a more concrete view of the extent to which changes in a feature affect the output of the model. High saliency for a feature indicates that small changes in that feature can dramatically alter the output of the model.

As can be seen in the plot below, the relative scale of saliences across both classes is the exact same. This conveys that the model has uniform sensitivity to features across both completed pass samples as well as incomplete pass samples, essentially displaying a lack of class-specific feature dependence. High saliencies for features such as the height level of a pass, the x location of where a pass ended, and the fact that a pass was of the cross type highlight the large influence they have over the model's prediction. Low saliences for features such as the fact that the play pattern during the pass was from a free kick and the fact that a pass was of the cut back type imply they are insignificant predictors.



Final Thoughts

Data analytics has been a growing component of the sports player development and scouting process for a while, but deep learning is still not heavily applied. The interpretability issues with neural networks is the main challenge to overcome before there is wide scale adoption of such models in sports research. Traditional machine learning techniques provide more explainable insights, which can then be used to formulate clear strategies.

A goal of this project was to show the power of deep learning when applied to sports data. My dataset and architecture were simple, and my experimental phase showed deep learning was able to produce solid results. More importantly, my evaluation phase showed ways insights into what the model has learned can be attained. Those insights are what can be relayed to decision makers within a sports organization so that they can go out and improve the team accordingly. I know with more advanced architectures and more detailed datasets, deep learning can have a transformational impact in uncovering trends in sports.

All it takes is opening up the back box.