COSC 5P30 Final Project - SportGNN

Raymond Romaniuk[1]
[1]Brock University
1812 Sir Isaac Brock Way
St. Catharines, Ontario L2S 3A1
rr16di@brocku.ca

Introduction

From an early age there have been two things that have always been a constant in my life: hockey and statistics. I started out as a hockey goaltender at seven years old (20 years ago now) and early on became the only goalie I've ever heard of to count the number of shots they faced while playing. Shot counting is always done by someone not directly involved in the on-ice action, but I found it helped me focus on the task at hand and would provide more accurate personal statistics, since I obviously had a better understanding of which shots actually hit me and which ones didn't. I still have all my hand tracked statistics, going all the way back to when I was ten years old. From number of saves, to number of goals allowed, to number of breakaways stopped and even number of practices attended.

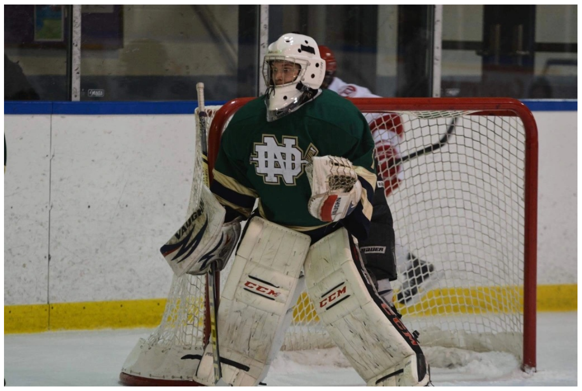
Figure 1: Me in goal for my high school team that went to the provincial championships two years in a row.

These days, since there isn't a market for undersized goaltenders, I've decided to focus my attention on how I can continue to have an impact on the best position in all of sports, but in a data driven way. It may not be the crisp footwork and perfect rebound control that I've grown up learning, and now teaching, but with the explosion of sports tracking data (and desire for insights from this data) I feel there's an opportunity to refine how the position is played by leveraging cutting edge methods from the machine learning and artificial intelligence fields. That brings me to the topic of this paper.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Topic

Our lack of access to the proprietary Amazon Web Services (AWS) National Hockey League (NHL) tracking data limits some potential avenues of exploration, but we can still develop an expected goals model using a novel graph neural network (GNN) approach, which we'll call SportGNN.

What Are Expected Goals (xGoals)?

Before we get to what an expected goal (xGoal or xG) is let's first introduce how they fit into the game of hockey to demonstrate why they are important.

Similar to many sports the objective in hockey is to score more goals than your opponent. To do this you need to shoot pucks past the opponents goaltender and into the opposing teams net. There are many ways to score a goal, but different strategies have different likelihoods of being successful. Taking a shot from 50 feet away with no players blocking the goaltenders vision is unlikely to lead to a goal. However, making a pass ”backdoor” (imagine the puck is on one side of the ice and a pass is made towards the goalpost that is furthest away) to a teammate forces the goaltender to move from one side to the other and cover the part of the net that was previously open. Even the example of the shot from 50 feet away can be made dangerous by adding players in front of the net to block the goaltenders vision of the puck, and then ”tipping” (deflecting the puck in mid air to change its trajectory) the shot before it reaches the goaltender, so it is no longer going where the goaltender is expecting leaving him unable to react in time.

Expected goals, as the name suggests, are the number of goals expected to be scored on a shot on goal. They can also be understood as the probably that a shot on goal results in a goal. So we would expect more dangerous shots to have a greater likelihood of being a goal and thus have a higher expected goals value. Likewise if a good shooter is shooting on a bad goalie there would be a greater likelihood of him scoring, so also a greater number of expected goals.

In practice xGoals are commonly used to quantify player and goaltender performance. If a player scores more goals than he is expected to, then we would say that this player is over performing relative to the chances he is getting to score. If a goaltender is allowing fewer goals than expected, then he is exceeding expectations and his team will have to score fewer goals to win games. Naturally, outpacing the expected goals, or goals against, that are expected of you can help lead your team to more victories.

This season Leon Draisaitl has scored 12.8 more goals than he's expected to and Connor Hellebuyck has allowed 18.3 fewer goals than expected to (he's saving almost one more extra goal a game!). With Draisaitl being one of the premier offensive players in the sport and Hellebuyck being the favourite to win the Vezina Trophy (best goaltender) these xGoal leaders make sense (xGoals from *Moneypuck* (Tanner 2024)). Conversely, Drew O'Connor's scored 6.2 goals fewer than he is expected to and Connor Ingram's allowed 10.9 goals more than he is expected to.

Problem

Being able to accurately evaluate players is crucial for NHL front offices that need to put the teams together. Teams are involved in a constant battle for the best information available and gaining even the slightest advantage over other teams could result in more wins and a better chance at lifting the Stanley Cup in June.

Developing a cutting edge xGoals model will help executives better value players across the league, allowing them to make better data driven decisions when making trades and signing free agents. Having superior insights would be game-changing and allow a team to get better value out of trades and contract signings than they otherwise would have.

In this project we will attempt to build this cutting edge xGoals model by employing the graph neural network (GNN) architecture and explore the impact of neighbouring players to the likelihood of a goal being scored on a given shot.

This work appears to be the first of its kind in the publicly available hockey analytics field and will hopefully lead to future advances in hockey analytics when more data becomes available.

Related Work

Unlike most research areas the premier research in sports analytics is mostly proprietary and used by individual teams to try to exploit the lesser knowledge of their opponents for their own gain. Making this cutting edge research public would remove a teams competitive advantage and it is thus understandable why it is kept behind closed doors.

Despite the lack of high level research available publicly, there are some exceptions in other sports, most notably soccer. This includes *TacticAI: an AI assistant for football tactics* (Wang et al. 2024) by Google DeepMind in collaboration with Liverpool FC, and *Detection of Tactical Patterns Using Semi-Supervised Graph Neural Networks* (Anzer et al. 2022) which won the Research Paper Competition at the 2022 MIT Sloan Sports Analytics Conference.

TacticAI

TacticAI: an AI assistant for football tactics provides a great example of the amazing research going on behind closed doors. This paper comes from Google DeepMind and is the product of their collaboration with English Premier League (EPL) powerhouse Liverpool FC (who seems to be a leader in the sports analytics world based upon the book I read called *Data Game* (Williams 2024)).

In this paper the authors focus on corner kicks and create three models to solve three different problems:

- *Receiver prediction*: the first model accounts for the attributes of the players on the field for a corner kick and predicts which three of the 22 players on the field are the likeliest to get the first touch of the ball.
- *Shot prediction*: the second model takes the same information as the receiver prediction model, but this time predicts whether a shot on goal will occur following the corner kick.
- *Tactic refinement*: the third model is a generative model tasked with optimizing offensive/defensive corner kick alignments to try to maximize the likelihood of an offensive/defensive outcome (shot or no shot).

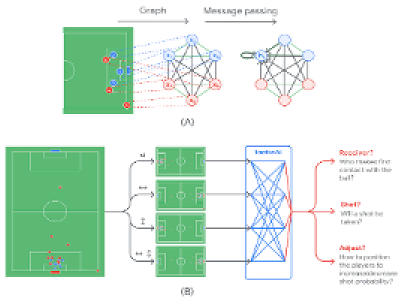
Graph
Message passing
(A)
TacticAI
Receiver?
Who makes first contact with the ball?
Shot?
Will a shot be taken?
Adjust?
How to position the players to increase/decrease shot probability?
(B)

Figure 2: Diagram of the three problems the TacticAI authors try to solve in the paper.

The shot prediction aspect of this paper most closely follows the approach we take with our xGoals model, as they're essentially developing an expected shots (xShots) model.

Similar to us the authors leverage the GNN architecture to build their model, however, more specifically, they build a graph attention network with a two-layer multilayer perceptron (MLP) as the attention mechanism. Additionally, in the same vein as our work, the nodes in the TacticAI graphs represent the players (position on the field, velocity, height, weight, etc.), while the edge weights indicate whether the pair of nodes are teammates, with the graphs being fully connected.

The receiver prediction and shot prediction models are used to help develop the generative tactic refinement model, which helps paint a clear picture of how these two models can be used in practice.

The authors do a great job developing insights that can immediately be actioned upon by the decision makers at Liverpool FC, and their generative tactic refinement model was even preferred by subject matter experts when compared to real corner kick examples. Unfortunately the work is not reproducible as the data and code is proprietary to Liverpool FC, so the authors results cannot be validated by other researchers.

Detection of Tactical Patterns

The goal of the *Detection of Tactical Patterns Using Semi-Supervised Graph Neural Networks* paper is to use German National team data to create a tactical pattern detection model, named SequentialM2, that:

- reduces the amount of manually labeled data needed from human experts.
- generalizes across different multi-agent scenarios (permutation invariant).
- can detect tactical patterns of unknown size that vary from pattern to pattern.

The authors focused on detecting overlapping runs, which are limited to two attackers versus one defender with one attacker having the ball and the other running past the player with the ball at a high speed, attempting to create a new passing option, and forcing the defender to choose which of the attackers to bias his coverage towards. To obtain this information from the dataset, subject matter experts were employed to label examples of overlapping runs.

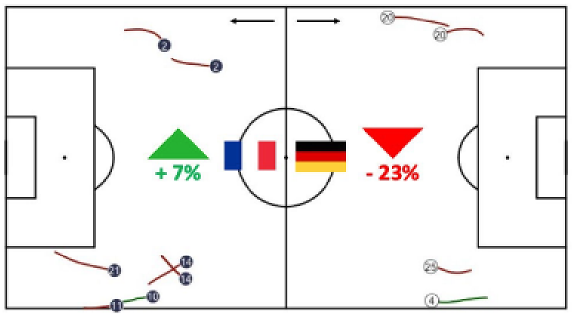

Figure 3: Example of SequentialM2's overlapping run detection during a game between the German and French National teams.

As the player tracking data used is temporally structured the authors used the ideas outlined in Chung et al. (Chung et al. 2016) combining a variational autoencoder (VAE) with the hidden state of a recurrent neural network (RNN). This sequential model is then combined with the semi-supervised approach from Kingma et al. (Kingma et al. 2014), where the objective function is divided into supervised and unsupervised parts, where the supervised part can be thought of as a regularized classification objective. The authors propose to combine these two approaches under the name SequentialM2.

Like us the authors implement a GNN, more specifically a GAT. The output for each player from the attention is represented as a node in a graph that can be used by a GNN for information propagation and modeling interaction patterns between players. The authors also used a k-nearest neighbours approach to ensure distant nodes had less impact on the trajectory of a player than the players immediate neighbours. The GNN then updates the player states in the RNN and the process is repeated again.

In the papers Appendix the authors demonstrate that the algorithm can also successfully model player movement in basketball, in addition to providing a major improvement over the current state-of-the-art in both AUC and F_1-Score. Unfortunately, the data for this work is very limited, only a handful of German National team games, so it is unclear whether these results are as strong for other teams. The results of this paper are also not reproducible as the data used does not seem to be publicly available.

Approach

Dataset

The dataset for this project comes from MoneyPuck (Tanner 2024), a leading contributor to the publicly available NHL analytics community. This dataset contains information about each shot (on goal, missed, blocked) taken during the past five NHL seasons. Each shot captured in the dataset contains information about shot location, shot angle, shooter, goaltender, shot type, and the number of players on the ice for each team, to name a few variables.

Unfortunately, player tracking data from AWS is currently proprietary, so a problem with the dataset that will be addressed later is that it does not contain any information about where players are on the ice, aside from the the shooter and the goaltender. This means that it is likely suboptimal for graph based machine learning.

Data Engineering

As with any data driven project the most important aspect is the quality of the data. Garbage data will create garbage results regardless of the effort exerted or complexity of the model. That is why the first thing we'll have to do before we train any machine learning or artificial intelligence models is complete some preliminary data engineering.

The first thing we did was check that all five years worth of data was structured identically. This was not the case, so we removed the columns that did not appear in all five years, which, after concatenating the five datasets left us with a single dataset having 548752 rows and 124 columns.

With all of our data consolidated we proceeded to standardize team abbreviations, account for players with identical names, remove shots that missed the net, remove shots with missing data and remove shots where the number of players on the ice was inconsistent with the maximum/minimum number of players that can be on the ice.

Once this preliminary data engineering was completed we're left with 387932 shots in the five years of data.

Categorical Encoding

A non-trivial aspect of our data engineering work focused on categorical encoding, namely how to handle categorical columns with a large number of classes, like player name or goalie name. The *shooterName* column contains 2017 unique player names and it wouldn't be feasible to one-hot encode this column and create over 2000 new columns that are predominantly filled with zeros.

To ensure that we didn't need to drastically increase the number of columns in our dataset we decided to use target encoding. Target encoding entails replacing each category in a categorical column with a number derived from the target, in our case goals. The major added benefit of using target encoding on a column with many categories is that it does not increase the size of the dataset, target encoding a column does not lead to many more columns like with one-hot encoding, which we will use for columns with fewer classes. Another feature we leveraged in the target encoding is smoothing.

Imagine a player begins his career with 15 goals in his first five games. Is it reasonable to believe that this production will sustain over time and 200 games into his career he will continue to score at this pace? Obviously not. Using smoothing on the encoded column accounts for this by ensuring categories with a small sample size have their encoded value adjusted toward the overall mean. This helps minimize potential outliers occurring in data points of players that have played few games.

Table 1 lists the categorical columns, the number of classes in that column and which type of encoding was used.

Table 1: Type of Encoding Used for Categorical Columns

Column	Classes	Encoding
homeTeamCode	32	Target
awayTeamCode	32	Target
team	2	One-Hot
location	3	One-Hot
shotType	7	One-Hot
lastEventCategory	17	Target
lastEventTeam	30	Target
goalieNameForShot	**237**	Target
shooterName	**2017**	Target
shooterLeftRight	2	One-Hot
playerPositionThatDidEvent	5	One-Hot
teamCode	32	Target

Graph Data

The raw shot data for this project is in tabular form, but to create SportGNN we'll need to convert each row in our dataset into its own graph.

Similar to TacticAI (Wang et al. 2024), the nodes in our graphs will represent the players on the ice. Since, we only have information about the shooter and the goaltender, our initial graphs will only have two nodes, one for the shooter and one for the goaltender. Despite our dataset having over 100 columns there is a lack of useful information available, especially for the goaltender (we only know who it is). Thus, each node will only contain three features, while the edge between the nodes will represent the distance the shooter is away from the net.

Since, our data is limited we tried to choose the information that would be most likely to contribute to the likelihood of a goal being scored. For the shooter these features are:

- *shooterName (target)*: target encoded value for the shooter, a higher value represents a more talented player with a higher likelihood of scoring a goal.
- *shotType (WRIST)*: one-hot encoded column for the WRIST class in the shotType column. Wrist shots are the most common type of shot, any other type, for example a deflection or wrap around, may be more likely to result in a goal.
- *shotAngleReboundRoyalRoad*: indicates whether this shot is a rebound and the puck crossed the royal road from the previous shot. The royal road is an imaginary line from the center of one net to the center of the other net (see Figure 4). A puck crossing from one side of this line to the other forces the goaltender to move a greater distance and creates a shooting opportunity at uncovered net if he does not arrive in the proper position on-time. From experience I can attest that a shot having this characteristic will be more likely to result in a goal.

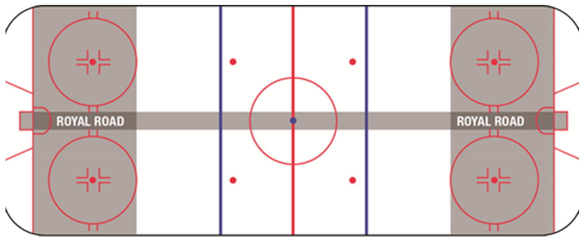

Figure 4: Diagram of the royal road down the center of the ice. Pucks crossing from one side of this line to the other force the goaltender to move a greater distance and can result in a greater likelihood of scoring a goal.

The only information we have explicitly pertaining to the goaltender is the name of the goaltender, as the majority of the columns within the dataset are the amount of time players have been on the ice. Only having this information we try to make the most effective use of it possible and use the following three features for our goaltender node:

- *goalieNameForShot (target)*: target encoded value for the goaltender, a lower value represents a more talented goaltender with a higher likelihood of making a save.
- *defendingTeamAverageTimeOnIce*: average time on ice (TOI) of the defending team is the average amount of time (in seconds) the team's players have been on the ice. If this number is high then the defenders will be tired and the shooter will be more likely to have a higher quality scoring chance.

- *defendingTeamMaxTimeOnIceOfDefencemen*: max TOI of defencemen of the defending team is the maximum amount of time (in seconds) one of the defencemen has been on the ice. Defencemen that have been on the ice for a long period of time are likely stuck in their own defensive zone and prone to making mistakes that could lead to dangerous scoring chances for the shooter.

With our node feature choice explained Figure 5 depicts an example of a goal that was scored.

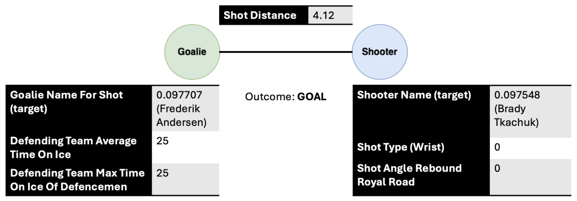
Shot Distance
4.12
Goalie
Shooter
Goalie Name For Shot (target)
0.097707 (Frederik Andersen)
Defending Team Average Time On Ice
25
Defending Team Max Time On Ice Of Defencemen
25
Outcome: GOAL
Shooter Name (target)
0.097548 (Brady Tkachuk)
Shot Type (Wrist)
0
Shot Angle Rebound Royal Road
0

Figure 5: Example of the two node graph of a goal scored by Brady Tkachuk on Frederik Andersen from 4.12 feet away. Andersen's defencemen were fairly fresh, only having been on the ice for 25 seconds. While Tkachuk's shot was not a wrist shot and was not a rebound that crossed the royal road.

Looking at our two node graph we have a very obvious issue. The goaltender and shooter are not the only two players on the ice! There could be as many as ten more skaters on the ice impacting the likelihood of a shot being a goal. Unfortunately we don't have access to the information about all of these adjacent players, but that won't stop us from trying to simulate some.

The first thing we need to do is decide what our adjacent players features will be. We have information about the number of forwards and defencemen on the ice for both the shooting and defending team, so we can leverage this to create more accurate adjacent players. Having this information allowed us to split the adjacent nodes into four categories: forwards on the shooting team, forwards on the defending team, defencemen on the shooting team and defencemen on the defending team. Additionally, our TOI data is broken down by forwards and defencemen for each team. Thus, for each of the four categories we have the following node features:

- *Shooting/Defending Team Forward/Defencemen Average TOI*: the average TOI of players in the specified category. If this TOI is high for the defending teams defencemen and low for the shooting teams forwards there's a greater chance of a high danger scoring opportunity.
- *Shooting/Defending Team Forward/Defencemen Average TOI Since Faceoff*: same as the previous feature, however it is the amount of time since the last whistle when there would have been a small break for the players.
- *Forward*: one if the player is a forward, zero if they are a defencemen. Forwards are more likely to score goals, but are also usually weaker defensively.

Our graphs will have twelve nodes, the maximum number of players allowed on the ice, and hockey being a dynamic game where every player on the ice can impact all other players, the graph will be fully connected. We don't have information about the distance that each player is away from one another, so we simulated uniform random numbers between 0 and 75 and assigned these simulated distances to be the edge features from our simulated nodes.

Note that there may not always be twelve players on the ice, so in the cases where this occurs we create "dummy" nodes to ensure the total number of nodes in the graph is always twelve. These dummy nodes contain three zeros for the node features and a zero for the edge feature.

Models

It's great to build a fancy cutting edge model, but in practice if a simpler and less computationally expensive model can outperform the more advanced one then you're better off saving your resources and going with the simpler model. This is why before we create our GNN models we'll first develop a couple baseline models to test them against. These baseline will be a simple logistic regression model and an extreme gradient boosting (xGBoost) model with hyperparamters chosen through k-fold cross-validation.

Our models are trained on the 2019-20, 2020-21, 2021-22 and 2022-23 seasons, and are tested on the 2023-24 season. The logistic regression model is rather simple to implement without any hyperparameters to tune, but our xGBoost model requires a greater effort. Note that even though our baseline models are able to handle all of the columns in our dataset we have limited the columns to only those used in SportGNN.

To train our xGBoost model we split our training data into four folds (the four seasons) and perform 4-fold cross-validation to select the optimal hyperparameters for our final model. The hyperparameters we explored the performance of were max depth and learning rate, where max depth is the maximum depth of its decision trees and the learning rate controls the rate at which the model updates after each epoch.

We checked the performance of learning rates in the range 0.05 to 0.4 and max depth values between 1 and 4 and using cross-validation found the optimal hyperparameter combination to be a max depth of three and learning rate of 0.15.

Having trained our baseline models we moved on to our GNN's. The model we settled on contained three GCNConv layers with three input channels and 64 or 128 output channels. Batch Normalization was used on the output of th final GCNConv layer and finally passed to a linear layer. The output of the first two GCNConv layers each passed through the ReLu activation function and global mean pooling was used on the output of the final GCNConv layer. A dropout rate of 50% was used and the final output of the model was passed through a sigmoid activation function. Note that the data loader used a batch size of 1024 for both the training and testing sets, while the learning rate for our Adam optimizer was set to 0.0001. Additionally, we used binary cross entropy loss as our loss function.

We used this architecture for both the two node graph and twelve node graph models.

Data Imbalance

Goals in hockey are difficult to score and occur on less than 10% of shots, this means that there is a significant imbalance between the number of goals and saves in our dataset. Thus, a model could achieve an accuracy over 90% by simply predicting that every shot was not a goal. Unfortunately, this results in a model that tells us nothing about what we care most about, goals.

How can we try to handle this issue? Over/undersampling. Simply put, to oversample a dataset we take samples from the minority class (in our case goals), with replacement, and add these samples to our dataset. We continue to sample data points until we have a dataset that is half goals and half saves. To undersample we essentially do the opposite and sample from the majority class, deleting these samples from the dataset until we have the same number of goals and saves.

Obviously strictly over/undersampling creates problems, as when we oversample we add duplicate data to our dataset with a greater class imbalance resulting in more duplicates and when we undersample we are throwing away valuable data that could be used to enhance our model.

Since, we would prefer to add more data to our dataset, rather than limit ourselves, we will use an oversampling algorithm, specifically the Synthetic Minority Oversampling Technique (SMOTE). SMOTE randomly samples the minority class and then chooses one of the sampled points neighbours at random using k-nearest neighbours. These data points are then used to create an artificial data point as follows,

$$x_{new} = x_i + (\hat{x_i} - x_i) \cdot \delta$$

.

In this formula x_i is the features of the *i*th minority class observation (the data point we randomly sampled), $\hat{x_i}$ is the randomly selected neighbour of x_i and δ is a random number between 0 and 1. This formula allows us to create artificial minority class data with similar characteristics to real minority data points and ensures we don't need to add many duplicates to our dataset, allowing us to balance the data with unique minority class data points.

To test the effectiveness of SMOTE in our project we used it on the data from the 2022-23 season and created a training dataset out of just that season with 155356 observations (before SMOTE it had 86240). The 81649 2023-24 observations remained our test set as over/undersampling is not performed on test sets.

Once our SMOTE data was prepared we fit our six models (logistic regression, xGBoost, SportGNN-2N (64 and 128 hidden channels), SportGNN-12N (64 and 128 hidden channels)) as we had before and compared to our previous results. The optimal hyperparameters we found for our xGBoost model are a max depth of seven and a learning rate of 0.4.

Findings

Given we were missing most of the data that would've helped us predict xGoals with graphs, I think that our models performed pretty well. By no means were they completely ground breaking, but considering we didn't even have data about the positioning of most of the players on the ice I think we did as well as we could have hoped. Table 2 shows the accuracies and mean squared errors of each of our original models, along with the accuracy a model that predicts that no shot results in a goal would attain.

Table 2: xGoal Model Test Performances

Model	Accuracy	MSE
Logistic Regression	89.9264	0.086552
xGBoost	**90.4910**	**0.081220**
SportGNN-2N-64	89.9264	0.100668
SportGNN-2N-128	89.9264	0.100703
SportGNN-12N-64	89.9264	0.100718
SportGNN-12N-128	89.9264	0.100726
No Goals	89.9264	-

From the table it is evident that all of our models, except for xGBoost, converged to a prediction of "No Goal" on every shot. This is not ideal, since we have a model that tells us nothing about the event we are interested in.

One interesting thing about our SportGNN-12N models is that they take significantly longer to converge to the prediction of every shot not being a goal than the SportGNN-2N models. The SportGNN-2N models take three or four epochs while the SportGNN-12N models take in excess of 20 epochs. This is likely due to the noise we've added with the ten extra nodes, but it is interesting to see that despite the introduction of some randomness the models ultimately perform identically upon the completion of training (even if that's just predicting every shot isn't a goal).

Since, our initial attempts seem to be impacted by the significant data imbalance, let's have a look at how augmenting our data with SMOTE impacts the performance of our models. Table 3 shows our model performances after using SMOTE on our training data.

Table 3: xGoal Model Test Performances Using SMOTE

Model	Accuracy	MSE
Logistic Regression	66.5250	0.207975
xGBoost	80.4688	0.138379
SportGNN-2N-64	89.3348	0.098578
SportGNN-2N-128	89.5504	0.096904
SportGNN-12N-64	84.1051	0.125365
SportGNN-12N-128	85.9790	0.116599
No Goals	89.9264	-

Looking at the table, even though predicting no goal everytime provides the highest accuracy, it is nice to see that our models are doing something differently and not converging directly there. Due to the lack of time I ended up having to work on the SMOTE part of the project I was only able to train my SportGNN models for 30 epochs, so it is possible that they may converge to all ”No Goal” prediction after that based on their behaviours in the first 30 epochs.

The SportGNN-2N models both reached 88% accuracy by the 11th and 13th epoch while subsequently bouncing between accuracies of 87% and 89%, with the highest accuracy being the accuracy obtained by predicting no goals. At only 30 epochs it is unclear if this behaviour would've continued indefinitely, or if there would have been some sort of convergence.

The SportGNN-12N models also followed a similar pattern to the SportGNN-2N models. They both reached 85% accuracy by the 11th and 13th epoch while subsequently bouncing around in the 80%'s, but never quite reaching the ”No Goals” benchmark. Again, due to having to stop a bit early I am unable to say whether the models current behaviours would have continued or if they would have converged.

Despite the lack of convergence, it is fascinating to see that the SportGNN models all outperformed the baseline logistic regression and xGBoost models. Again due to the time crunch I couldn't have a deep look at why this happened, but it seems to be a good start at the very least.

Conclusion

Ultimately, despite lacking the AWS NHL tracking data we successfully created a novel xGoals model called SportGNN.

As with all raw data we performed the necessary data engineering tasks to optimally position it for inference. We solved the column explosion problem that one-hot encoding would cause by leveraging target encoding on categorical variables having more than ten classes.

Since our raw data was in a tabular format we created our own classes to transform each row in our dataset into its own graph. Two types of graphs were developed, two node and twelve node. The two node graphs contained only the shooter and goaltender with no neighbouring players impacting the likelihood of a goal. For the twelve node graphs we used the data available to develop meaningful node features for the neighbouring players and created fully-connected graphs of all the players with uniform randomly sampled distances as the edge features of the adjacent players.

Using our tabular data we developed logistic regression and xGBoost models as baselines and then trained our four SportGNN models. All of our models, except xGBoost, were impacted by the severe data imbalance, so we used SMOTE on the training dataset and trained our models again.

Using SMOTE seemed to alleviate the data imbalance problem, however we did not have the time to train the models for more than 30 epochs to confirm. Most notably our SportGNN models were able to significantly outperform our baseline models.

Even in spite of lack of NHL tracking data it seems we've done an admirable job of creating a novel initial prototype that could be useful should tracking data become available (fingers crossed).

Challenges

Initially one of the big challenges was creating the graphs to train SportGNN on. It took a non-trivial amount of time to get that piece of the project working, but thankfully we were able to eventually create the graphs we needed and create our SportGNN models.

Additionally, the significant data imbalance seems to be playing a role in the training of our models, since predicting every shot isn't a goal results in a model with over 90% accuracy. This is something that likely still needs to be tackled, but we've taken a step in that direction by testing the effectiveness of using SMOTE on the training data.

Possible Extensions

The ideal possible extension is that tracking data becomes available and we are able to use it to make more accurate graphs to train SportGNN on. In the interim our dataset still suffers from a lack of information about the goaltender facing the shot. Enriching our dataset with more information about the goaltender, like height, catching hand, games played, etc., is a logical first step to enhancing our models performance. It also might be advantageous to add some additional data for the shooters, but more information about the goaltender should be gathered first, since we only know who he is and nothing else.

As noted when analyzing the results of using SMOTE on the training data, we were unable to find some time to train the SportGNN models for more than 30 epochs. This means we are unable to provide a firm conclusion on the performance of the SportGNN models when SMOTE is used. Training the models for more epochs would give us a better understanding whether model performance will continue to bounce around the previously mentioned ranges or if the models will eventually converge to some stable level of performance. Once this information is in hand we'd be able to give a more comprehensive evaluation of how good our final SportGNN models are.

Something I wasn't able to fit in was using average accuracy to analyze model performance instead of accuracy. My Masters research looked at combatting imbalanced data and this was one of the main things I looked at. This could be useful in understanding model performance in a case where, for example, we have 100 observations 99 are in class A and 1 is in class B. If we predict class A everytime we have a model with 99% accuracy, but the models average accuracy across classes is only 50%, since it gets 100% on class A, but 0% on class B. Using average accuracy will help us focus more attention on the minority class, which in class imbalanced situations is usually the class that's more interesting (goals, fraudulent transactions, car accidents).

The architecture used for our SportGNN model didn't change much from the beginning of the project to the end, so I think it would be good to check how architecture changes impact performance. We'd also like to test how well graph attention network's (GAT) perform on this data, since both papers we referenced in the Related Work section made use of them while developing cutting edge work. With the Google DeepMind team creating a generative models whose tactical adjustments were preferred to real corner kick tactics, and the tactical pattern detection paper winning the best paper award at the MIT Sloan Sports Analytics Conference. If SportGNN could garner even a fraction of that type of performance we'd be headed in a great direction.

Lastly, it'd likely be ideal if I transitioned this work to a GPU and used a greater proportion of the data available. To ensure I had time to get through everything I had planned I used only five years worth of data instead of all ten to save some compute time. It'd be good to check if there is any change in model performance if we use all the data. In sports things are always changing, so there is always the question of if shots from ten years ago are still valuable when predicting whether the shots taken today will be goals. I think there could be some value in going back the full ten years, even if we decide to give the older data less weight in the model than the newer data.

References

Anzer, G., Bauer, P., Brefeld, U., Fassmeyer, D. 2022. Detection of Tactical Patterns Using Semi-Supervised Graph Neural Networks. *MIT Sloan Sports Analytics Conference*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., Bengio, Y. 2016. A Recurrent Latent Variable Model for Sequential Data. *CoRR*.

Kingma, D., Mohamed, S., Rezende, D., Welling, M. 2014. Semi-supervised Learning with Deep Generative Models. *In Advances in Neural Information Processing Systems*.

Tanner, P. 2024. Moneypuck.com - NHL Analytics, Playoff Odds, Power Rankings, Player Stats. *https://moneypuck.com/*.

Wang, Z., Veličković, P., Hennes, D. et al. 2024. TacticAI: an AI assistant for football tactics. *Nature Communications*.

Williams, J. 2024. Data Game: The Story of Liverpool FC's Analytics Revolution. *Pitch Publishing (Brighton) Limited*.