Reinforcement Learning on NHL Expected Goal Models in SuperTuxKart

Hubert Luo [1]

Abstract

Applying key factors and lessons learned from National Hockey League (NHL) expected goal models to SuperTuxKart using reinforcement, imitation, and deep learning approaches via a state-based agent.

1. Introduction

The state of quantitative analytics in professional hockey has evolved significantly over the last few years, specifically in the world's top hockey league, the National Hockey League (NHL). Early attempts involved looking at puck possession metrics such as Corsi that focused on whether a player has possession of the puck. Later models looked at scoring chances and expected goals which evaluated what players did with the puck once they had possession of it. (Goldman, 2021)

Although there are clear differences between hockey games in real life and a SuperTuxKart hockey tournament, fundamental principles about puck possession and expected goals translate well between the two environments and formed the inspiration for this work.

2. Previous Work

Expected goal models such as Evolving-Hockey and NaturalStatTrick based in the NHL found that the distance from the puck to the net when taking a shot was consistently the most important feature. Other important factors in these NHL expected goal models included not only shot location, but also player locations. (Goldman, 2021)

These factors were important considerations in data generation and feature engineering as outlined in sections 3.1 and 3.2

*Equal contribution [1]Department of Computer Science, University of Texas at Austin.

3. Method

A multi-step pipeline was created to first generate game state data, then generate labels based on their expected goals, engineer features based on a team's state, and finally model expected goal outcomes based on those features.

3.1. Data Generation

In order to generate data, AI models with added random noise were played against each other. Game state data from these matches were then collected and used for training. 1250 matches were played with an 80-20 training-validation split, i.e., 1000 matches in training set and 250 in validation set.

States from each match were then labelled based on an expected goals metric. Motivated by NHL expected goal metrics as explained earlier in sections 1 and 2, this metric captured what conditions on the rink lead to goals for a specific team.

The expected goals of a team right before they score a goal was assigned a value of 1. On the other hand, a value of -1 corresponds to the situation immediately prior to their opponent scoring a goal.

An exponential decay was used to account for the fact that events occurring in the timesteps immediately prior to the goal are much more important than those that occurred at the beginning of the game. Specifically, the formula used was:

$$t = \text{time until next goal} \in \mathbb{Z}_{\geq 0} \tag{1}$$

$$a = \mathbb{1}_{\text{next goal scorer}} \tag{2}$$

$$y = \text{expected goals} \tag{3}$$

$$= (2a - 1)e^{-\frac{1}{16}t} \tag{4}$$

Note that in the above formula, if the team was the next goal scorer, $a = 1$ and thus $y = e^{-\frac{1}{16}t}$ However, if the opponent is the next goal scorer, $a = 0$ and thus $y = -e^{-\frac{1}{16}t}$. The decay rate $\frac{1}{16}$ was empirically derived to ensure the expected goals measurement accurately captured sufficient pre-goal data while also ignoring irrelevant states that occurred long before the goal was scored.

3.2. Feature Engineering

Important features in NHL expected goal models involve the distance from the puck to the net and player positions on the ice relative to not only the puck but also to each other (Goldman, 2021). Therefore, similar features were engineered based on available data from SuperTuxKart states.

First, the x/y coordinates of the goal, puck, player kart, and closest opponent were calculated for each player. In addition, the angle of the puck, kart, and closest opponent were determined relative to both the player and to the puck. Furthermore, we accounted for the distance from the kart to the closest opponent and the distance between the puck and the opposing goal. These engineered factors account for the important spatial and positional factors found in NHL expected goal models.

In conjunction, puck possession factors were created - for example, if a player was close to the puck and no members of the opposing team were nearby, that player was judged to have possession of the puck. A similar feature was used to designate opposing possession of the puck and a contested puck, i.e., when one or more member of each team is in close proximity to the puck.

In the NHL, one of the primary ways to create offense is through rush chances when the attacking team quickly moves the puck up the ice and catches the defence flat-footed. This can be in the form of a breakaway, i.e., no defender between the attacking player and the goalie, and an odd-man rush. What this means is there are more attackers than defenders between the attacking players and the goalie.

Figure 1. Example of a 3-on-1 rush chance. From Dallas Stars vs Toronto Maple Leafs NHL game on October 20, 2022

For example, Figure 1 has three attacking players (Toronto Maple Leafs in blue jerseys) while only one defender (Dallas Stars in white jerseys) between them and their goalie, constituting a 3-on-1 rush. These rush chances are particularly challenging to defend given attacking players can employ various passing mechanisms to pull the defender(s) and goalie out of position and thus are more likely to score a goal. Therefore, we account for these situations in our model.

Table 1. Summary of Selected Features.

VARIABLE	MEAN
PLAYER HAS POSSESSION	0.114
CONTESTED PUCK	0.268
# DEFENDERS BETWEEN ATTACKERS AND GOAL	1.043

A subset of engineered features and their distributions in the training data collected in section 3.1 are in table 1, which demonstrates that in 11.4% of all situations, a player has possession of the puck while in 26.8% of situations, a puck is contested. Odd-man rushes and breakaways were relatively rare as in most cases there was at least one defender between the attacker and the goal.

3.3. Expected Goals Model

A model that takes in a game state as input and predicts the expected goals was necessary given we do not know *a priori* what the true expected goals of a situation is. Therefore, we designed a fully-connected neural network for this prediction task.

The model's architecture consists of three blocks which each had a linear layer of dimension 64 paired with a ReLU non-linearity. The network's design was the result of extensive hyperparameter tuning on the number of blocks, the dimension of the linear layers, and the learning rate. The best-performing model consisted of the above design along with a learning rate of 0.01 with an ADAM optimizer and a learning rate scheduler that decreased the learning rate when the loss plateaued.

We experimented with dropout layers in various configurations such as between blocks and prior to the final output, however results did not seem to show considerable difference in performance and ability to generalize. Therefore, we decided not to use dropout layers in the final output.

Training was performed on 8000 game states derived from the training data generated in section 3.1. Likewise, for validation, 1855 game states were used from the validation data generated in section 3.1. Model results are outlined in section 4.1.

3.4. Action Model and Match Expected Goals Metric

A kart's action was determined by a neural network where the input was the features described in section 3.2 and the output was the kart's acceleration, steering, and breaking. We then ensured the action model's output were applicable to their given ranges using the below formulae:

$$z = \text{action model output} \in \mathbb{R}^3 \quad (5)$$
$$\text{acceleration} = \text{sigmoid}(z_0) \in [0, 1] \quad (6)$$
$$\text{steer} = \tanh(z_1) \in [-1, 1] \quad (7)$$
$$\text{brake} = \mathbb{1}_{z_2>0} \in \{0, 1\} \quad (8)$$

This action model was consistent throughout a match - therefore, we were motivated to create a match expected goals metric to evaluate the effectiveness of an action model throughout an entire match.

The intuition behind the match expected goals metric was that we took the sum of the expected goals of a team's player who was most likely to score. This was because the position of the other player was already considered as a feature in the expected goals model. As mentioned earlier, we focused on the most relevant game states closest to when the next goal was scored in coming up with this metric to avoid misleading predicted values of expected goals that did not result in an actual goal. The match expected goals formula for one team was:

$$N = \text{number of time steps in a match} \quad (9)$$
$$\hat{y_{t,i}} = \text{predicted expected goals at time } t \text{ for player i} \quad (10)$$
$$y = \text{match expected goals for one team} \quad (11)$$
$$= \sum_{t=N-14}^{N} \max(\hat{y_{t,0}}, \hat{y_{t,1}}) \quad (12)$$

3.5. Gradient-Free Optimization

Our first approach was to use gradient-free optimization. A different action model was rolled out for each match and a Match Expected Goals metric was created to evaluate the effectiveness of that model. First, the expected goals of each game state in a match were predicted as we did not know *a priori* what the true expected goals of that state were.

Each match was played until one team scored so we focused on the most relevant game states that occurred at the end of a match right before that goal was scored. We then rolled out a different action model for each match and calculated the match expected goals to evaluate that model's effectiveness using the methodology in section 3.4. This was repeated across 100 epochs to find the action net with the best initialization.

After finding a good initial action model, it was fine-tuned using a random iterative searching procedure. Random noise was added to the initial model weights and this altered model was then rolled out in different matches to find the model with the best match expected goals metric. This was repeated across 200 iterations. See section 4.2 for detailed results.

3.6. Imitation Learning

Our next approach was motivated by the idea of finding a better initial action model - to do this, we leveraged the existing AI model which was treated as an expert that we imitated using a neural network.

The model's architecture consisted of two blocks, one with a linear layer of dimension 64 and another with a linear layer of dimension 32, both paired with a ReLU non-linearity, again derived after hyperparameter tuning. Tuning also found that our best-performing model had a learning rate of 0.001 with an ADAM optimizer and a learning rate scheduler that decreased the learning rate when the loss plateaued.

After getting a good initial model from imitation learning, we again fine-tuned it using a random iterative searching procedure, adding random noise to the initial model weights and rolling out the altered model across 200 iterations to find the model with the best match expected goal metric. See section 4.3 for detailed results.

4. Results

4.1. Expected Goal Model Results

A fully-connected neural network was used to predict the expected goals given any game state. Motivation, methodology, and architecture were outlined previously in section 3.3.

Figure 2 shows the training and validation mean squared error (MSE) on training and validation datasets. The x-axis is the number of epochs and the y axis is the MSE value. Training loss decreases steadily as the number of epochs increase. On the other hand, the validation loss fluctuates. As expected, the training loss is usually lower than the validation loss.

4.2. Gradient-Free Optimization Results

Leveraging the methodology described in section 3.5, an action net with a good initialization was first identified over 100 trials. This initial action net achieved a match expected goals metric of 0.185, meaning it generally underperformed the AI agent during matches.

However, fine-tuning through the random iterative search outlined in section 3.5 across 200 iterations demonstrated significant improvement in the best-performing action net, which achieved a match expected goals metric of 1.747.

See table 2 for a summary of match expected goal metrics from a subset of trials, demonstrating the considerable

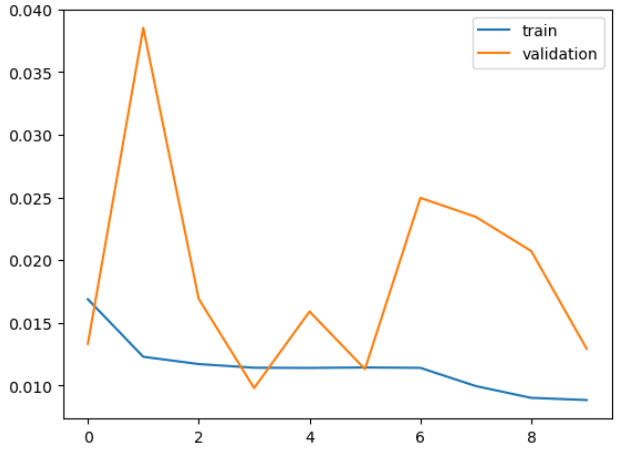

Figure 2. xG Model MSE on Training and Validation Datasets.

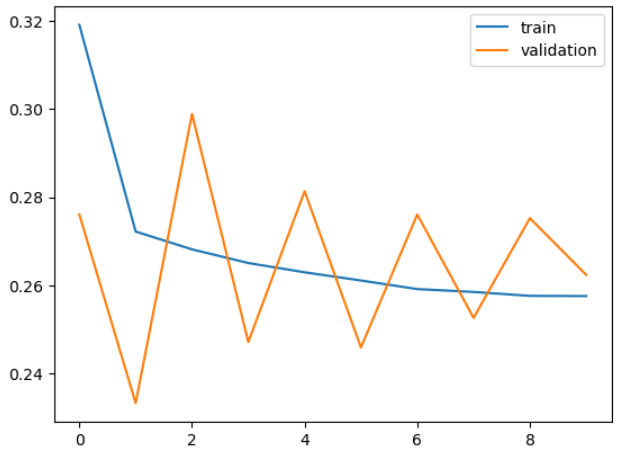

Figure 3. Imitation Learning MSE on Training and Validation Datasets.

Table 2. Summary of Selected Gradient-Free Optimization Trials and Imitation Learning.

MODEL	MATCH XG
FINE-TUNED BEST MODEL	1.747
FINE-TUNED IMITATION LEARNING MODEL	1.116
IMITATION LEARNING MODEL	0.205
GOOD INITIALIZATION (TRIAL 61)	0.185
BAD INITIALIZATION (TRIAL 17)	-1.180

improvement moving from a bad initialization to a good initialization to the fine-tuned best model.

Potential reasons for why this model struggled were due to the extensive search space of all possible actions, which limited the usefulness of the match expected goals function as a reward and also restricted our ability to search across the whole space.

4.3. Imitation Learning Results

As outlined in section 3.6, a network was first trained to imitate the existing AI agent as a good model initialization. See figure 3 for the training and validation mean squared error (MSE) on training and validation datasets. The x-axis is the number of epochs and the y axis is the MSE value. Training loss decreases steadily as the number of epochs increase, however the validation loss fluctuates.

After a good initial model based on the AI agent was found, a random iterative search as outlined in section 3.6 was conducted. See table 2 for a summary of the match expected goal metrics for both the imitation learning model as well as the imitation learning model after fine-tuning.

While the imitation learning model outperformed the initial good initialization outlined in section 4.2, the fine-tuned imitation learning model underperformed the best model by the match expected goals metric. This demonstrates there may have been a local maxima around the imitation learning model that prevented it from outperforming the best model found in section 4.2.

As previously mentioned, this model may have likewise struggled due to the extensive search space of all possible states, limiting the usability of the match expected goals metric as a reward function.

5. Conclusion

Recent NHL approaches looking at expected goals were easily applied to SuperTuxKart using reinforcement, deep, and imitation learning approaches via a state-based agent. Although promising, there were limitations in the applicability given clear differences between the two environments.

References

Goldman, S. Comparing public expected goal models: How they work and what we should take away from them. *The Athletic*, 2021.