Vrije Universiteit Amsterdam

Bachelor Thesis

Comparative Analysis of Machine Learning Approaches in Developing AI for Supertuxkart Soccer Mode Bots

Author: Marcel Niedzielski (2726454)

1st supervisor:	Michael Cochez
daily supervisor:	Michael Cochez
2nd reader:	Mauricio Verano Merino

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

July 5, 2024

Abstract

Over the past years, the integration of Artificial Intelligence (AI) has significantly shaped various industries. Integrated more and more into people's daily lives, the research on AI has become more relevant than ever. This study investigates the effectiveness of three distinct machine learning approaches for AI in the soccer mode of the open-source game Supertuxkart. It compares a scoring model, a reinforcement learning (RL) model, and a scoring model enhanced with higher-level in-game functions. Each model's performance was evaluated based on training and validation metrics as well as real time gameplay effectiveness. The scoring model, despite promising training results, did not work effectively in gameplay. The RL model improved significantly over time, showcasing its potential for dynamic and adaptive decision-making. The enhanced scoring model performed well against the baseline AI, winning 72 out of 100 games. The limitations of this study include resource limits that affect the duration of model training and the diversity of the dataset. Future research should concentrate on optimizing hyperparameters, executing more extensive reinforcement learning training, and investigating other advanced machine learning approaches. This study seeks to provide significant insights into the utilization of machine learning for AI in gaming, hence contributing to the larger field of AI development.

Keywords: Artificial Intelligence, Machine Learning, Reinforcement Learning, Supertuxkart, Video Games, AI in Gaming, Model Evaluation

1 Introduction

Over the past years, the growth of artificial intelligence (AI) and its integration into gaming have progressed significantly, enhancing both the user experience and game dynamics. AI in gaming is not just about creating intelligent opponents but also about crafting immersive environments where it can interact and adapt in real time, as said by Yannakakis and Togelius on page 24 of their book [10]. What started off with the early rule-based systems progressed to advanced algorithms of machine learning, which made the games much more exciting yet complex [3].

One such game that benefits from AI is Supertuxkart, which is an open-source racing game with a soccer mode. In this mode, players steer a kart across a soccer pitch in an effort to score goals while at the same time preventing their rivals from doing so. In racing and sports games such as Supertuxkart, artificial intelligence is very important to simulate realistic competition. Hence, effective AI improves gameplay by making strategic decisions, adapting to different scenarios, and providing dynamic challenges for players. The soccer mode therefore requires the AI to make real time decisions about correct positioning and scoring opportunities. Effective AI must subsequently navigate the field, control the ball, and interact with opponents seamlessly.

Developing effective AI for Supertuxkart soccer mode bots presents several challenges. Those include the need for real time decision-making, strategic maneuvering, and adaptability to dynamic game environments. The primary challenge, however, is to create an AI that can consistently perform well in competitive matches, making intelligent decisions that lead to goal-scoring opportunities while defending against the opponent. Given the complexity of the Supertuxkart soccer mode, comparing different machine learning approaches can provide insights into their strengths and weaknesses in this specific context. By evaluating various models, the aim is to identify which methods offer the best performance in terms of accuracy and gameplay effectiveness. This comparison is done to enhance AI performance in real time strategy games and contribute to the broader field of AI in gaming.

This research aims to contribute to the field by providing a comprehensive analysis of three different machine learning approaches for AI development in Supertuxkart soccer mode. By studying the performance of different models, insights into their effectiveness and practical applications can be gained, further advancing the understanding and development of AI in gaming. This paper focuses on the following research question: How do the scoring model, reinforcement learning model, and scoring model enhanced with higher-level in-game functions compare in their effectiveness for AI performance in Supertuxkart's soccer mode? This study strives to determine the most effective methods for creating competitive and intelligent AI, thereby improving the gaming experience of Supertuxkart and similar games.

2 Method

Supertuxkart is an open-source racing game that includes various game modes. As said before, one of them is the soccer mode, in which players control karts to drive around a soccer field and attempt to score goals with a soccer ball. What differs from regular soccer is that there is no such thing as out-of-bounds, fouls or off-sides. The game also includes power-ups, which are collectibles that spawn around the field, and once driven over, give an offensive item to the player. These include items like cake grenades or bowling balls, which you can aim at opponents to slow them down or launch them into the air. The combination of racing and strategic gameplay elements in this mode made it suitable for testing the AI's decision-making abilities. During the game, the AI must make real time decisions about driving maneuvers and tactical positioning in order to win the match.

In the exploration of implementing a machine learning model for Supertuxkart soccer mode AI bots, three different approaches were taken: a scoring model, a reinforcement learning (RL) model, and a scoring model enhanced with higher-level in-game functions. Each approach was selected for its potential to address different aspects of the AI challenge in this game mode. The source code for the experiments can be found on github [1].

2.1 Data Collection

All approaches were trained with the same dataset collected during gameplay, which was 101.1 MB in size. The features were collected every 250 milliseconds from the current game state. The data collection involved recording various game metrics such as the positions of the karts and the ball, velocities, accelerations, and other relevant in-game parameters (see chapter 2.2.1 Feature Engineering).

The dataset was gathered from multiple 1-versus-1 matches to ensure diversity and robustness. The data was collected based on the games of the already existing AI provided by Supertuxkart, and the games were run without the interference of any real players. In order to differentiate between desired and undesired actions, the ones that were taken by the bot within 10 seconds before scoring a goal were labeled as desirable, and the rest of the actions were labeled as undesirable.

[1]https://github.com/Potat0-Salad/SuperTuxKartAI

All of the games were run on the same track, named Icy Soccer Field, to ensure that the evaluation environment remained consistent, thereby providing a reliable basis for comparing the performance of the different machine learning models.

2.2 Data Preprocessing

Several steps were taken to make certain that the data collected was appropriate for training the machine learning models, such as feature engineering, normalization, and handling data imbalance.

2.2.1 Feature Engineering

Choosing and transforming game metrics was part of feature engineering to be able to select a comprehensive dataset that would best train the models. To provide a clear illustration of the features and their relevance, a top-view drawing of the arena is visible in Figure 1. The features listed below were created based on this process:

- Positions of the kart (x, y, z coordinates)
- Positions of the ball (x, y, z coordinates)
- Ball aiming position (x, y, z coordinates)
- Previous positions of the kart (x, y, z coordinates, 0.25 seconds before current state)
- Ball heading direction (Numerical value)
- Ball approaching goal (binary indicator: yes or no)
- Distance of the kart to the ball (meters)
- Kart velocity (x, y, z components)
- Kart speed (meters per second)
- Kart break (binary indicator: yes or no)
- Steering angle (-1 until 1: -1 = left, 0 = straight, 1 = right)
- Acceleration (0 until 1: 1 = 100% throttle, 0 = 0% throttle)
- Sector of the map (Numerical index)
- Target (Ball, Opponent, Power-up)
- Target's position (x, y, z coordinates)
- Sector numbers of the other karts in-game (Numerical indexes)
- Coordinates of the other karts in-game (x, y, z coordinates)
- Presence of power-ups (binary indicator: yes or no)

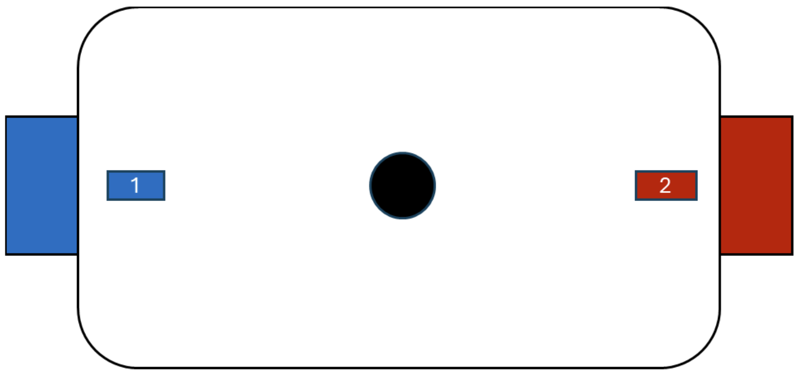

Figure 1: Top-view of the arena, showing the positions of the blue and red goals, karts 1 and 2, and the ball at the tip-off position.

2.2.2 Dealing with Data Imbalance

Data imbalance was the major difficulty in this study, especially when categorizing actions into desirable and undesirable. Two strategies were tested to deal with this problem. Firstly, labeling the actions taken 10 seconds before the goal as desirable and the rest as undesirable. This method led to a great imbalance in data because most of the actions were labeled as undesirable, far outnumbering those that were desirable.

To deal with the imbalance, a second approach was taken to achieve a higher level of equilibrium in the data collection. As previously, the actions taken 10 seconds before the goal were marked as desirable. However, the method for labeling negative actions was different. A random sequence of 10 seconds from the negative actions was chosen and labeled as undesirable, while the remaining data was discarded. This approach guaranteed that the dataset had an identical count of both positive and negative actions, hence ensuring a balanced training process.

2.2.3 Normalization

Normalization was a crucial step in pre-processing the data to ensure that all features contributed equally to the model's learning process. Each continuous feature was scaled to have a mean of zero and a standard deviation of one. This was accomplished using the StandardScaler from the Sklearn library. For binary indicators, scaling was not necessary as they take values of either 0 or 1, so they already fit the model. Categorical features were one-hot encoded to create binary vectors before being fed into the model.

The same mean and scale parameters obtained during the normalization process were stored and later used to normalize the continuous inputs passed from the game before being fed into the model. This consistent normalization ensured that the model received data in the same scale as it was trained on, which is essential for maintaining the model's performance during gameplay.

2.3 Scoring Model

The scoring model is a machine learning approach in which the AI evaluates potential actions based on a scoring system. During training, the model's task was to perform binary classification of actions that led to a goal versus those that did not. The model processed these inputs to evaluate the desirability of each potential action. During gameplay, the model was provided with the environmental features and kart controls at that specific moment in the game. Six of the following possible actions to take were provided as well: forward, forward left, forward right, reverse, reverse left, reverse right. Based on the potential movements provided, the benefit of performing each one of them in the given situation on the field was evaluated. Then, the movement with the best score was executed. While this approach allows AI to make informed decisions, its effectiveness depends on the accuracy and relevance of the scoring system, which may change depending on the game scenario and dynamics.

2.3.1 Configuration and Parameters Set for the Scoring Model

After multiple runs, based on trial and error, the optimal parameters for the model were identified and applied. Different values and configurations of these parameters were tried out, to determine the setup that provided the best performance in terms of the model's accuracy and efficiency in the game environment. The network's layers consisted of input layers, several hidden layers, and an output layer. Each hidden layer used activation functions such as ReLU or Leaky ReLU to introduce non-linearity and enhance the model's capacity to recognize complex patterns and relationships in the data. The L2 regularization was employed to discourage excessively large weights and prevent overfitting, ensuring the model remained generalizable. Furthermore, the features Target and Target position were dropped as they were not relevant for this model. The range of parameter values tested included the following:

- Step size (learning rate): 0.0001 to 0.001
- Number of epochs: 20 to 1000
- Number of layers: 3 to 7
- Batch size: 32 to 128
- Dropout rates: 0.2 to 0.5
- Activation functions: ReLU, Leaky ReLU
- Hidden layer sizes: 32 to 512 neurons per layer
- Optimization algorithms: Adam, SGD
- Loss function: Binary Cross-Entropy (BCELoss)
- Regularization techniques: L2 regularization

The Binary Cross-Entropy Loss (BCELoss) function, used to measure the performance of the model, calculates the loss by comparing the predicted probabilities and the actual binary outcomes. It is formulated as follows:

$$\text{BCELoss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i))]$$

where y_i is the actual label, $p(y_i)$ is the predicted probability, and N is the number of samples.

The optimal parameters were selected based on their capacity to minimize the loss function and optimize the AI's performance in accomplishing its objectives within the game. The primary advantage of this model is its ability to quickly evaluate and select optimal actions based on the learned scoring system, potentially leading to improved gameplay performance and strategic decision-making.

2.4 Reinforcement Learning Model

The reinforcement learning (RL) model is a machine learning approach in which the AI learns to make decisions by interacting with the game environment while receiving feedback in the form of rewards or penalties [8]. The RL model learns through a process of trial and error, continually improving its strategy based on the outcomes of its actions.

The reinforcement learning model implemented in Supertuxkart was first trained similarly to the scoring model, but less extensively, in order to give it a head start in making the right decisions. This initial training phase involved using the pre-collected dataset to help the model learn basic gameplay mechanics and decision-making strategies. After this initial phase, the RL model was matched 1 versus 1 against the existing AI, initiating its actual training process. During these matches, the model continuously interacted with the game environment, making decisions and receiving real time feedback. During the gameplay, the model assigned scores to potential actions, which were heavily influenced by the reward system, and subsequently selected the action with the highest score to be performed. Over time, the model was able to improve its performance by fine-tuning its strategies through feedback in the form of rewards for desired actions and penalties for undesirable ones. The iterative process of this training made it possible for the RL model to change and improve, which eventually made it better able to compete in the Supertuxkart soccer mode.

2.4.1 Configuration and Parameters Set for RL Model

The RL model was set up using a neural network trained through an RL algorithm. To identify the optimal configuration, various parameters were adjusted, and different values were tested to evaluate their performance in terms of the model's learning efficiency and gameplay performance:

- Step size (learning rate): 0.0001 to 0.001
- Number of episodes: 500 to 5000
- Discount factor (gamma): 0.9 to 0.99
- Exploration rate (epsilon): 0.1 to 1.0, with a decay strategy to balance exploration and exploitation
- Batch size: 32 to 128
- Replay buffer size: 10000 to 50000
- Update frequency: Every 1 to 10 episodes
- Activation functions: ReLU, Leaky ReLU
- Regularization techniques: L2 regularization

The RL model was trained by allowing it to play numerous games, where it learned to optimize its actions based on the rewards received. The reward structure was designed to encourage desirable behaviors such as scoring goals and defending, while penalizing undesirable actions. The rewards and penalties were given as follows:

- Scoring a goal: A high reward of ten points was given for the actions that led to scoring a goal.
- Conceding a goal: A high penalty of ten points was given for the actions that led to conceding a goal.
- Approaching the ball: A reward of one point was given for shortening the distance between the kart and the ball.
- Moving the ball towards the opponent's goal: A reward of three points was given for actions that moved the ball towards the opponent's goal.
- Moving the ball towards own goal: A penalty of three points was applied for actions that moved the ball towards the AI's own goal.

In this investigation, the reinforcement learning model demonstrated the ability to adapt and improve its strategy over time, showing potential for dynamic and complex decision-making in the Supertuxkart soccer mode. This model's primary advantage lies in its capacity to learn directly from the environment, making it highly adaptable to different gameplay scenarios and strategies.

2.5 Scoring Model Enhanced with Higher-Level In-Game Functions

The scoring model enhanced with higher-level in-game functions builds on the basic scoring model by incorporating more complex decision-making processes. Instead of picking the controls, like acceleration and steering, this approach involves dynamically selecting the kart's targets with the use of the scoring model. The model is provided with the environmental features and kart controls at a specific moment in the game, and based on that information, it evaluates the potential for scoring a goal for each possible target. There are four different targets available: the ball, closest opponent of the kart, the ball-chaser from the opposing team, or a power-up. Once the selection is complete, a corresponding higher-level action is executed to reach the target, which had been assigned the highest score.

2.5.1 Configuration and Parameters Set for the Scoring Model with Higher-Level Functions

Similar to the basic scoring model, the enhanced model underwent multiple iterations to identify the optimal parameters through trial and error. The parameters were adjusted systematically to determine the best configuration for maximizing the model's performance in terms of accuracy, minimizing the loss function, and efficiency in the game environment. The network's layers consisted of input layers, several hidden layers, and an output layer. Each hidden layer used activation functions like ReLU or Leaky ReLU to introduce non-linearity, improving the model's ability to learn complex patterns. The range of parameter values tested included:

- Step size (learning rate): 0.0001 to 0.001
- Number of epochs: 100 to 1000
- Number of layers: 3 to 7
- Batch size: 32 to 128
- Dropout rates: 0.2 to 0.5
- Activation functions: ReLU, Leaky ReLU
- Gradient clipping: 0.1 to 0.5
- Regularization techniques: L1 and L2 regularization

By incorporating higher-level in-game functions, the model can evaluate multiple potential targets and make more informed decisions about which target to pursue. For instance, the model might decide to chase the ball if it is closer to scoring a goal, attack an opponent to disrupt their strategy, or collect a power-up to gain a temporary advantage. This dynamic target selection allows the AI to adapt to changing game situations and optimize its actions based on the current state of the game.

The enhanced scoring model underwent experimentation with both L1 and L2 regularization techniques in order to fine-tune its performance. L1 regularization was applied in an attempt to encourage sparsity within the model. This was achieved by penalizing the sum of the absolute values of weights, resulting in a more simplistic and interpretable model structure. Conversely, L2 regularization was also implemented, with its primary role being to avoid overfitting by penalizing large weights present in the model, thus enabling better generalization to new gaming scenarios. Moreover, gradient clipping was used to manage large gradients that could occur due to the increased complexity of dynamically selecting targets and executing higher-level actions, ensuring more stable training and improved performance.

3 Results

The subsequent section will present a comprehensive evaluation of the performance of the machine learning models developed for the Supertuxkart soccer mode AI. An analysis of three main areas forms the evaluation: training and validation loss, accuracy, and in-game performance. The scoring model, reinforcement learning (RL) model, and scoring model enhanced with higher-level in-game functions were considered in terms of their decision-making and gameplay effectiveness. From the analysis of training and validation metrics, it was possible to identify the models' learning processes as well as their ability to generalize to new data. Moreover, gameplay testing was conducted to evaluate how effectively these models can translate their learned behaviors into real time strategic actions during gameplay. Those tests involved matches played against the baseline AI, with each game continuing until one team scores 100 points. The baseline AI is the default AI implemented by the Supertuxkart developers, that works based on hard-coded behaviours. One example of its behavior is that if a kart is not the closest one from its team to the ball and it does not have an active power-up, it will drive towards the closest power-up. This comparison against the existing AI allows for a thorough evaluation of the new AI model's performance and effectiveness in dynamic game scenarios.

3.1 Scoring Model

3.1.1 Training and Validation Performance

The parameters that proved most effective for this model were: a step size of 0.0005, 100 epochs, a batch size of 64, a dropout rate of 0.5, Leaky ReLU activation functions, and L2 regularization with a weight decay of 1e-4. The model consisted of a total of six layers, including five hidden layers and one output layer. The scoring model was trained up to 100 epochs, and the step size was adjusted using the ReduceLROnPlateau scheduler. The plots visible in Figure 2 present the training and validation losses, as well as the accuracy of the model.

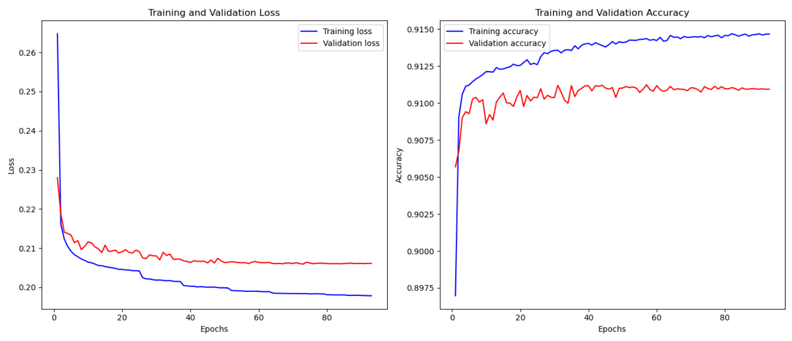

Figure 2: Training and Validation Losses and Accuracy for the Scoring Model

These plots demonstrate several key trends. Training loss decreases sharply in the first few epochs and then gradually afterward, suggesting consistent learning and error rate minimization. The final training loss stabilized around 0.20. The training accuracy shows a rapid increase in the early epochs and then stabilizes around the 20th epoch. This indicates that the model quickly learns to predict correctly on the training data. The final training accuracy was around 0.912.

The validation loss initially declines and then steadies with small oscillations after around 10 epochs, indicating that the model successfully generalizes to new data without overfitting. The final validation loss stabilized at around 0.21. The validation accuracy shows similar trends, increasing rapidly in the early epochs before stabilizing around 0.910. Small oscillations in this trend suggest that the model performs consistently on validation data. The average values observed were: training loss of 0.202, training accuracy of 0.9125, validation loss of 0.2125, and validation accuracy of 0.9100.

Early stopping at the 84th epoch was triggered to prevent overfitting, as indicated by the stabilization of validation loss and accuracy. It is a technique used during training to halt the process when the model's performance on a validation set stops improving. With this method, the model is likely to maintain its generalization abilities without unnecessary overtraining.

Overall, this scoring model demonstrates robust training and validation performance, highlighting the benefits of incorporating advanced features such as dropout, batch normalization, and L2 regularization for improved strategic gameplay.

3.1.2 In-Game Performance

Despite the promising results in the training and validation phases, the model did not perform well during the test match. The first game ended with the scoring model losing to the base AI 0:100. To ensure fairness, the sides on the field were switched, and an additional game was played. However, the match still resulted in a 0:100 loss for the scoring model. This contrast between the in-game performance and the statistical results from the training and validation phases highlights a significant limitation of the current model. Despite the fact that the training and validation metrics showed strong performance, with low training and validation loss and high accuracy, the gameplay did not benefit from these metrics. The model's general poor performance against the base AI and its inability to score goals indicate that the features and strategies learned during training did not align well with the dynamics of the actual game.

3.2 Reinforcement Learning (RL) Model

3.2.1 Training and Validation Performance

The RL model was initially pre-trained with supervised learning in order to provide a head start for the RL training phase. This pre-training was less extensive than that of the scoring model, as its main purpose was to ensure the model began with a basic understanding of gameplay mechanics, such as moving towards the ball rather than performing ineffective maneuvers.

The pre-training used the following parameters: a step size of 0.0005, a batch size of 64, a dropout rate of 0.3, Leaky ReLU activation functions, and L2 regularization with a weight decay of 1e-4. It was conducted for a duration of 50 epochs. The plots visible in Figure 3 depict the training and validation losses, as well as the accuracy of the model.

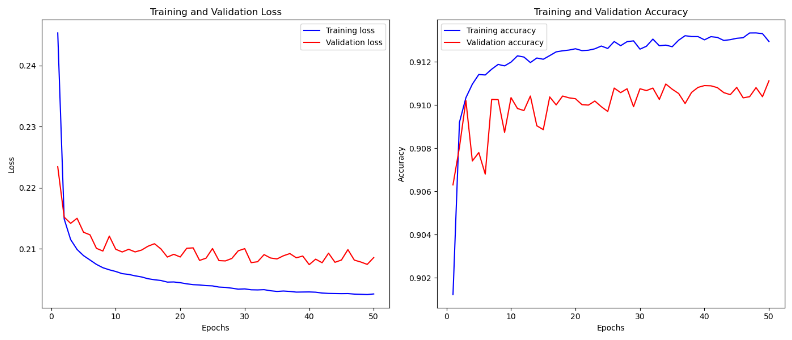

Figure 3: Pre-Training Loss and Accuracy for the RL Model

The pre-training phase demonstrated several key trends. The training loss decreased sharply in the first few epochs and then gradually afterward. It stabilized around 0.2025 by the 50th epoch. The training accuracy showed a rapid increase in the early epochs, stabilizing around the 20th epoch, with a final training accuracy of approximately 0.9133. The validation loss initially declined. However, after 10 epochs, it steadied with small oscillations and then proceeded to stabilize around 0.2086.

Once the RL training started, the model's loss initially spiked as it began making moves that did not yield optimal rewards. The loss during the early stages of RL training was around 0.4. However, as training progressed, the model visibly improved, and after approximately 72 hours of training, the loss fell below 0.25. That indicated improvement, with room for further progress.

3.2.2 In-Game Performance

The in-game performance of the RL model showed clear improvements over time. At first, the model's cumulative reward was low, reflecting its inexperience and suboptimal decision-making. However, as training progressed, the model's cumulative reward steadily increased, signifying that it was learning to make more effective strategic moves. Furthermore, as the process continued, it was also evident that the kart's movements and decision-making improved.

In the early stages of the RL training, the model lost games with a score of 0:100 against the baseline AI. However, after approximately 72 hours of continuous training, the RL model managed to improve its performance, eventually scoring 3:100 against the baseline AI. This progression indicates that the model may be capable of learning and adapting its strategies. Further testing and validation will be necessary to determine the consistency and reliability of these improvements in competitive gameplay scenarios.

3.3 Scoring Model Enhanced with Higher-Level In-Game Functions

3.3.1 Training and Validation Performance

The parameters that proved most effective for this scoring model enhanced with higher-level in-game functions were: a step size of 0.0005, 100 epochs, a batch size of 64, a dropout rate of 0.5, Leaky ReLU activation functions, and L2 regularization with a weight decay of 1e-5. There were 6 layers in total, including 5 hidden layers and an output layer. Even though the model was trained up to 100 epochs, early stopping was triggered at the 57th epoch. The plots visible in Figure 4 show the training and validation losses, as well as the accuracy of the model.

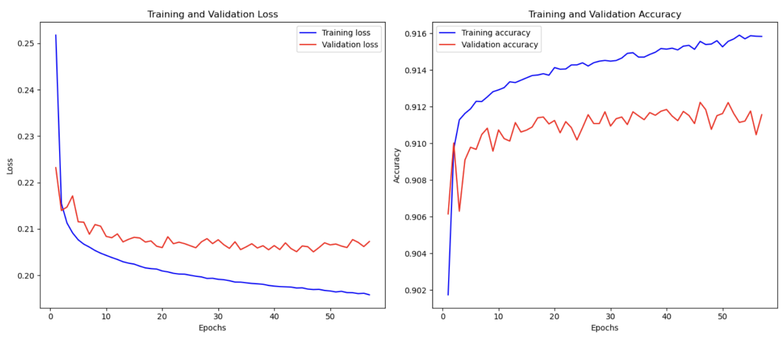

Figure 4: Training and Validation Losses and Accuracy for the Enhanced Scoring Model

The model's training and validation loss, as well as the accuracy plots, demonstrate several key trends. Training loss decreases sharply in the first few epochs and then gradually after that, suggesting constant learning and error rate minimization. Training accuracy shows that the model quickly learns to predict correctly on the training data, improving sharply in the early epochs and stabilizing around the 20th epoch.

The validation loss first declines and then steadies with small oscillations after around 10 epochs, indicating that the model successfully generalizes to new data without overfitting. Similar trends can be seen in validation accuracy, which increases rapidly in the early epochs before stabilizing at 90%. Small oscillations in this trend suggest that the model performs consistently on validation data. The average values observed were: training loss of 0.2013, training accuracy of 0.9140, validation loss of 0.2078, and validation accuracy of 0.9109. Early stopping at the 57th epoch was triggered to prevent overfitting, as indicated by the stabilization of validation loss and accuracy.

Overall, the enhanced scoring model with higher-level in-game functions demonstrates solid training and validation performance, suggesting that incorporating advanced features can contribute to improved strategic gameplay.

3.3.2 In-Game Performance

As with the other models, the in-game performance was evaluated through a duel against the baseline AI, with each match up to 100 points. The game ended with the model losing 75:100 to the base AI. Typically, only one match was played per model. However, due to the close score of the first game, a series of additional ones were played to further evaluate the performance. A total of 100 games up to a 1000 points were played. To ensure fairness, the sides and team colors were switched after game 50. The outcome of the games was 72 wins for the enhanced scoring model and 28 wins for the baseline AI. In order to assess whether there was enough evidence to conclude that the scoring model is performing significantly better, a one-sided paired binomial test was conducted.

Hypotheses:

$H_0: \quad \pi_A = \pi_B = 0.5$
$H_1: \quad \pi_A > \pi_B$

where π_A is the probability that the scoring model wins and π_B is the probability that the baseline AI wins.

Input data:

- Number of games, $N = 100$
- Number of wins by the scoring model, $k = 72$
- $N - k$: Number of wins by the other model

P-value: Probability that A wins k times or more, given the null hypothesis is correct.

The probability of observing k wins of A:

$$P(N,k) = \binom{N}{k} \pi_A^k \pi_B^{N-k}$$

P-value Formula:

$$P(N,k|\pi_A = \pi_B = 0.5) = \sum_{i=k}^{N} \binom{N}{i} \pi_A^i \pi_B^{N-i} = \sum_{i=k}^{N} \binom{N}{i} 0.5^N$$

If A wins 72 games out of 100 from B, then:

$$P(N = 100, k = 72|\pi_A = \pi_B = 0.5) = \sum_{i=72}^{100} \binom{100}{i} 0.5^{100}$$

Using statistical software to find the p-value:

$$P(X \geq 72|n = 100, p = 0.5) \approx 1.611 \times 10^{-6}$$

The p-value is approximately 0.000001611, which is much smaller than the common significance level of 0.05. This indicates that there is very strong evidence against the null hypothesis. Therefore, it can be concluded that the scoring model performs significantly better than the baseline AI.

3.4 In-Game Comparison of the Scoring Model Against the Reinforcement Learning Model

When compared to the baseline AI, the scoring and reinforcement learning models performed very similarly. Based on that score, it's difficult to say which one is more effective in terms of gameplay. As a result, an additional series of games were played between them. A total of ten matches have been played, all of them with the goal of reaching 100 points. To ensure fairness, the scoring model played on the red team and the reinforcement learning model on the blue team for the first five games. The teams were then switched, with the scoring model playing on the blue team and the reinforcement learning model on the red team for the next five games. The result of those matches came out to be very one-sided, as all ten games were won by the RL model.

This result suggests that the reinforcement learning model has better adaptable and strategic decision-making capabilities. Despite having similar training and validation performance metrics, its seems that the capacity to learn and improve through interaction with the game environment enabled it to outperform the scoring model during gameplay. This aligns with a prior study by Sutton and Barto [8], which, in its first chapter, emphasizes the effectiveness of reinforcement learning in dynamic and interactive settings.

4 Discussion

This research explored the performance of three machine learning approaches for developing AI in Supertuxkart's soccer mode: scoring model, reinforcement learning (RL) model, and a scoring model enhanced with higher-level in-game functions. The investigation focused on training and validation performance, as well as in-game effectiveness. The scoring model performed well in the training and validation stages, with a validation loss stabilizing at roughly 0.208 and a training loss at 0.202. However, in spite of such promising results, the model failed to transfer these advantages into effective gameplay. The evaluation game against the baseline AI has ended with a 0:100 loss for the scoring model. The RL model improved over time, eventually scoring 3:100 against the baseline AI. Moreover, it dominated the scoring model in a series of games. The enhanced scoring model showed robust metrics and significantly outperformed the basic scoring model in gameplay by winning 72 out of 100 games.

The discrepancy in the scoring model's performance between the training and validation phases in relation to the effectiveness of the game implies flaws in its ability to manage dynamic gameplay. While the RL model's early high loss values signal initial inefficiencies, its improvement trend shows that reinforcement learning can eventually increase strategic decision-making capabilities. Mnih et al. [4] demonstrated that techniques like experience replay and target networks could enhance learning stability, suggesting potential improvements for the RL model. The enhanced scoring model's success in gameplay highlights the effectiveness of incorporating higher-level in-game functions.

The results highlight how important it is for AI models in games to perform well not only in training and validation metrics but also in converting these metrics into dynamic gameplay strategies. The enhanced scoring model's success suggests that higher-level in-game functions are optimal for improved performance, suggesting a potential path for AI advancement in gaming in the future.

The study faced several limitations, primarily due to resource constraints. The amount and duration of model training were limited by the computational resources at hand. This was especially true for the RL model, which gains a great deal from longer training times. Moreover, the restricted dataset size and diversity might have affected the models' ability to generalize to a wider variety of in-game scenarios. These restrictions align with issues raised by Goodfellow, Bengio, and Courville [2], who, in chapter 11 of their work, emphasized the importance of data diversity and size in creating reliable AI models. Additionally, the data labeling process could have been improved to ensure more accurate and comprehensive annotations. Furthermore, the lack of computational resources hindered in-depth hyperparameter tuning, which could have further enhanced model performance.

5 Related Work

Recent research has significantly contributed to the understanding and development of AI models in gaming. In chapter 1 of their book, Yannakakis and Togelius [10] provide a comprehensive overview of the historical development and current advancements in AI for games. At the same time, on page 24, they highlight how these models enhance game dynamics and user experience. Furthermore, their work emphasizes the application of AI in games and how creating intelligent opponents capable of strategic decision-making is crucial for realistic and challenging gameplay.

Perez-Liebana et al. [5] provide an in-depth examination of the General Video Game AI framework and competition. They discuss the challenges of creating AI that can adapt to a wide range of games. The authors emphasize the necessity for versatility and robustness in AI agents to handle various game dynamics without relying on game-specific heuristics.

Recent advancements in AI for gaming have explored dynamic game difficulty scaling, adaptive behavior-based AI, and reinforcement learning approaches to enhance player experience and game design [9].

Reinforcement learning (RL) has been widely adopted for developing AI in dynamic and interactive environments. In chapter 1, Sutton and Barto [8] give a thorough description of reinforcement learning (RL) theory and practice, showing how RL models learn through interacting with their surroundings. This method works especially well for AI in games, where optimal performance requires constant learning and adaptability. Implementing this approach was attempted in this study. It was done to explore its potential in creating AI that can adapt and improve through continuous interaction with the dynamic environment of Supertuxkart's soccer mode.

Furthermore, Silver et al. [7] demonstrated the application of a general reinforcement learning algorithm to master chess and shogi through self-play, highlighting the versatility and effectiveness of RL techniques. Berner et al. [1] demonstrated the application of large-scale deep reinforcement learning to complex real time strategy games like Dota 2. The findings show the potential of RL in handling intricate game dynamics, and the approaches described by the authors could be used to further enhance the Supertuxkart AI. Silver et al. [6] discuss the integration of advanced features such as deep learning and neural networks, which enable AI to make more informed and complex decisions. These improvements are necessary to create competitive AI in complex game modes. Implementing these advanced techniques could further enhance the AI's performance in Supertuxkart, making it more competitive and realistic.

6 Conclusion

This research aimed to compare the effectiveness of three different machine learning approaches in developing AI for Supertuxkart's soccer mode: the scoring model, the reinforcement learning (RL) model, and the scoring model enhanced with higher-level in-game functions. This paper's investigation into the training and validation performance, along with in-game effectiveness, turned out to be useful in understanding the strengths and limitations of each of these approaches. While the scoring model seemed promising, based on training and validation metrics, it completely failed to beat the base AI in the game. The results pointed toward possible issues of overfitting and dynamic gameplay adaptation. In contrast, the reinforcement learning model improved with time, suggesting its potential for improving strategic decision-making in complex in-game situations. The enhanced scoring model's success in winning 72 out of 100 games highlighted the benefits of incorporating higher-level in-game functions. The outcome suggested that such enhancements can significantly boost AI performance in real time scenarios.

7 Future Work

Future research should aim to optimize the hyperparameters and training processes for both the scoring and RL models to improve their robustness, generalizability, and adaptability. This could involve longer training durations and the use of more diverse datasets to better generalize the models' performance across various scenarios. Furthermore, there should be a focus on training the RL model more extensively with more resources. Moreover, other deep machine learning techniques, such as deep reinforcement learning and recurrent neural networks, can be explored to further improve the AI in making strategic decisions in dynamic environments. Testing these models through the different game modes available within Supertuxkart and on other games would give a better insight into their generalizability and practical applications. Alternative methods of model training, beyond binary classification, could also be a potential area to explore. Furthermore, labelling of the actions as desirable and undesirable could be taken to a new, more complex approach. An example of that would be adding a label "very undesirable" to actions that lead to conceding a goal.

Further investigation should also consider what real-world implications these AI models can have and how they can be applied beyond gaming to those problem areas requiring intelligent, adaptive decision-making. Continued work in this field will evolve sophisticated and more capable AI systems that will set a base moving into the future for further innovation in AI research and applications.

References

[1] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., De Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., ... & Zhang, S. (2019). Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* (Cornell University). `https://doi.org/10.48550/arxiv.1912.06680`

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

[3] Lu, Y., & Li, W. (2022). Techniques and paradigms in modern game AI systems. *Algorithms, 15*(8), 282. `https://doi.org/10.3390/a15080282`

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529-533. `https://doi.org/10.1038/nature14236`

[5] Perez-Liebana, D., Samothrakis, S., Togelius, J., Schaul, T., & Lucas, S. M. (2016). General Video Game AI: Competition, Challenges, and Opportunities. *Proceedings of the AAAI Conference on Artificial Intelligence, 30*(1). `https://doi.org/10.1609/aaai.v30i1.9869`

[6] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484-489. `https://doi.org/10.1038/nature16961`

[7] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv* (Cornell University). `https://doi.org/10.48550/arxiv.1712.01815`

[8] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.

[9] Wu, Y., Yi, A., Ma, C., & Chen, L. (2023). Artificial intelligence for video game visualization, advancements, benefits and challenges. *Mathematical Biosciences and Engineering, 20*(8), 15345–15373. `https://doi.org/10.3934/mbe.2023686`

[10] Yannakakis, G. N., & Togelius, J. (2018). *Artificial Intelligence and Games.* Springer. `https://doi.org/10.1007/978-3-319-63519-4`