Utilizing Deep Learning for Autonomous Play in SuperTuxKart Ice-Hockey: A State-Based Approach

Samit Roy
samit.roy@utexas.edu

***Abstract*— This study explores the development of a state-based deep learning agent designed to autonomously play ice-hockey within the SuperTuxKart game environment. The objective was to create a model capable of competing in a 2 vs 2 tournament setting and scoring goals against predetermined opponents. The project integrates advanced neural network architecture to interpret game state data and output strategic movements. Performance was evaluated based on goals scored.**

I. Introduction

Deep learning has revolutionized the field of artificial intelligence, enabling significant advancements in computer vision, natural language processing, and autonomous control systems. In the realm of game AI, deep learning approaches have been instrumental in achieving superhuman performance in complex environments. This project aims to design and train a deep learning-based neural network to play ice-hockey within the SuperTuxKart game, focusing on a state-based approach.

The motivation for selecting a state-based agent over an image-based agent stemmed from the desire to emphasize the control aspects of the AI, leveraging direct state information to make strategic decisions and actions within the game. This approach aligns with recent research in reinforcement learning where agents utilize state information to navigate and interact with their environment efficiently.

II. Methods

Our approach involved the development of a deep neural network capable of processing the state information of the player's kart, the opponent's kart, and the puck's location. The network architecture was inspired by successful models in reinforcement learning and game AI, incorporating both feedforward and recurrent layers to capture spatial and temporal patterns within the game state.

A. Data Collection and Feature Extraction

Since jurgen agent was a most significant goal scorer, data was collected to gather around 20 games data with various combinations of players with jurgen agent namely the AI, geoffrey, yann and yoshua agents. This yielded data around the strongest player scoring the greatest number of goals. The intent was to learn from the features of the jurgen agent and be able to imitate it as much as possible. A state based approach seemed the plausible methodology to be able to learn from.

The following features were extracted during training from the data collected,

The algorithm followed was as such,

Initialize tensors for kart's front, center, and velocity using pstate
Calculate kart's direction vector and angle (orientation)
Calculate kart's speed using the velocity vector's norm

Initialize tensor for puck's center using soccer_state
Calculate direction and angle from kart to puck
Normalize angle difference between kart's orientation and direction to puck

Initialize tensor for goal line's center using soccer_state
Calculate direction and angle from puck to goal line
Normalize angle difference between kart's orientation and direction to goal line

Calculate angle from puck to goal line
Calculate distance from kart to puck

Calculate direction and angle from kart to goal
Normalize angle difference between kart's orientation and direction to goal

Combine all extracted features into a single tensor:
Kart's location (center)
Kart's direction vector
Kart's orientation angle
Kart's velocity vector
Kart's speed
Puck's location (center)
Angles and angle differences related to puck, goal line, and goal
Distance from kart to puck

Return the combined tensor containing all features

The features collected were,

1. **Kart's Location (Center)**: Indicates where the kart is on the field, which is essential for navigating towards the puck or goal.

2. **Kart's Direction Vector**: A normalized vector showing the direction the kart is facing, useful for planning movements and aiming.
3. **Kart's Orientation Angle**: The angle representing the kart's current facing direction, important for aligning the kart with the puck or goal.
4. **Kart's Velocity Vector**: Shows the current movement direction and speed, which affects how the model will reach the puck or position itself.
5. **Kart's Speed**: The speed at which the kart is moving, affecting the ability to chase the puck or retreat to defend.
6. **Puck's Location (Center)**: The position of the puck on the field, which is central to any interaction with it, whether dribbling, shooting, or passing.
7. **Kart to Puck Direction**: A normalized vector from the kart to the puck, guiding the model on how to approach the puck.
8. **Kart to Puck Angle**: The angle from the kart to the puck, which is used to determine how much the model needs to turn the kart towards the puck.
9. **Kart to Puck Angle Difference**: The angular difference between the kart's orientation and the direction to the puck, informing the model how to adjust its orientation to face the puck.
10. **Kart to Goal Line Angle Difference**: The angular difference between the kart's orientation and the direction to the goal line, which informs the model how to adjust the kart's orientation for shooting or goal-oriented actions.
11. **Kart to Puck Distance**: Measures the distance between the kart and the puck, which is vital for deciding when to accelerate or when to perform maneuvers like shooting or passing.
12. **Kart to Goal Direction**: A unit vector pointing from the kart towards the goal, which informs the model about the direction to move towards the opponent's goal to prepare for a shot.
13. **Kart to Goal Angle**: The angle from the kart to the goal, indicating how much the model needs to turn the kart to aim at the goal, which is crucial for shooting accuracy.

14. **Goal Line Center**: The central point of the team's goal line, useful for defending or understanding the goal area.

15. **Puck to Goal Line Direction**: A vector from the puck to the goal line, which can help plan where to shoot or pass the puck.
16. **Puck to Goal Line Angle**: The angle from the puck towards the goal line, guiding the model on aiming shots or passes to the goal area.
17. **Puck to Goal Angle**: The angle from the puck directly towards the goal, crucial for precise shooting or passing to maximize scoring chances.

B. *Architecture*

Here, an architectural diagram of the neural network used is provided,

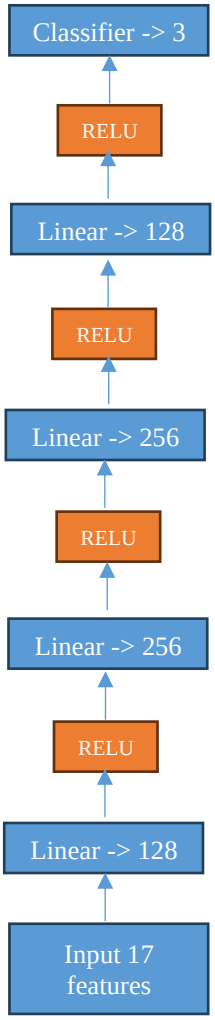


Fig: Action Net that intakes 17 features and yields 3 features namely acceleration, brake and steer

C. *Training Algorithm Description*

Initialize the model and set the computing device (GPU/CPU).
If continue_training is enabled, load the existing model.

Set up loggers for training and validation, if logging is enabled.
Initialize the optimizer (Adam) and learning rate scheduler (ReduceLROnPlateau).

Define the number of epochs, best validation loss, and early stopping criteria.
Define the loss function (MSE for regression).

For each epoch:
Load training and validation data in batches.

In training mode:
For each training batch:

Extract features and targets, move to the device.

Zero the optimizer's gradients.

Forward pass: predict acceleration, steer, and brake.

Calculate and backpropagate losses.

Clip gradients and perform an optimizer step.

Accumulate training losses and count samples.

Log average training losses per epoch.

In evaluation mode:

For each validation batch (no gradient calculations):

Extract features and targets, move to the device.

Forward pass: predict acceleration, steer, and brake.

Calculate losses.

Accumulate validation losses and count samples.

Log average validation losses per epoch.

Step the scheduler with the average validation loss.

Check for early stopping conditions and save the best model.

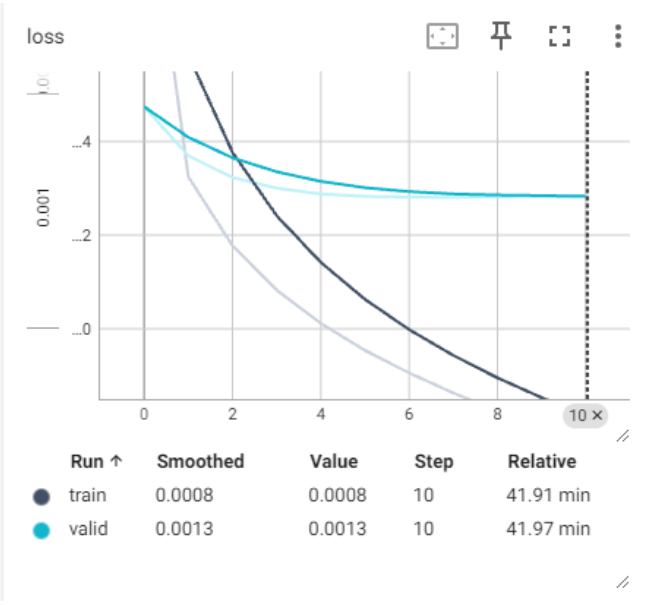
Save the final trained model

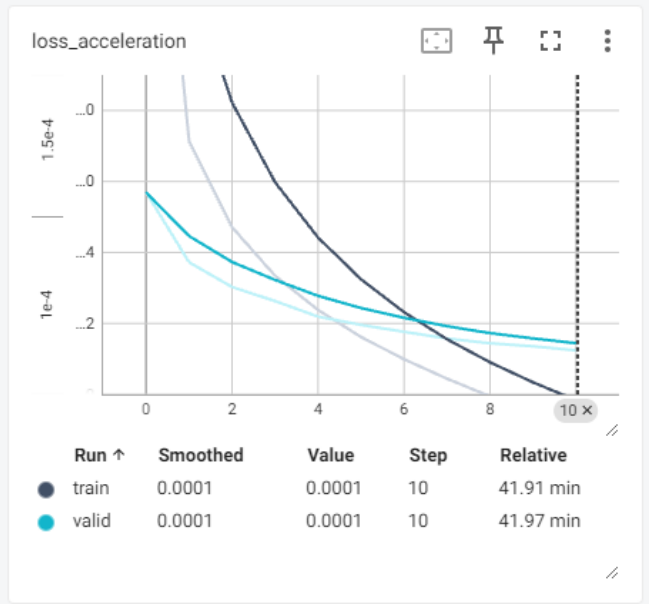
III. Results

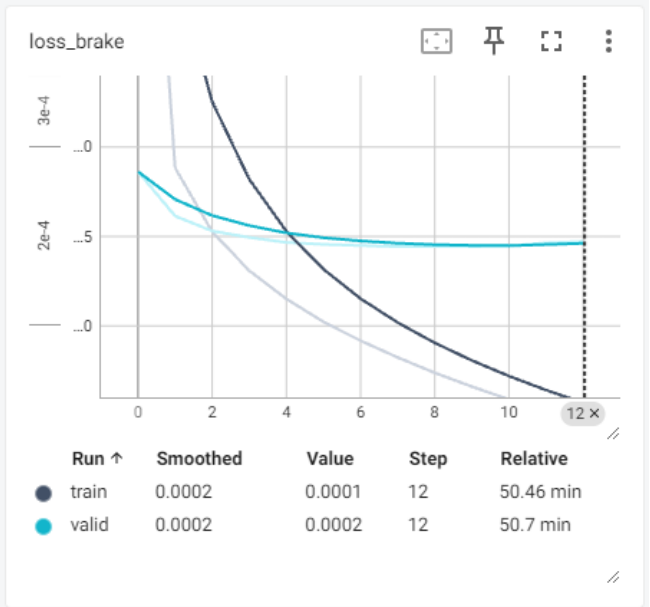
The agent was evaluated based on its ability to score goals and win matches against the baseline agents designed by the teaching assistants. The performance was measured in terms of goals scored per match, win rate, and computational efficiency.

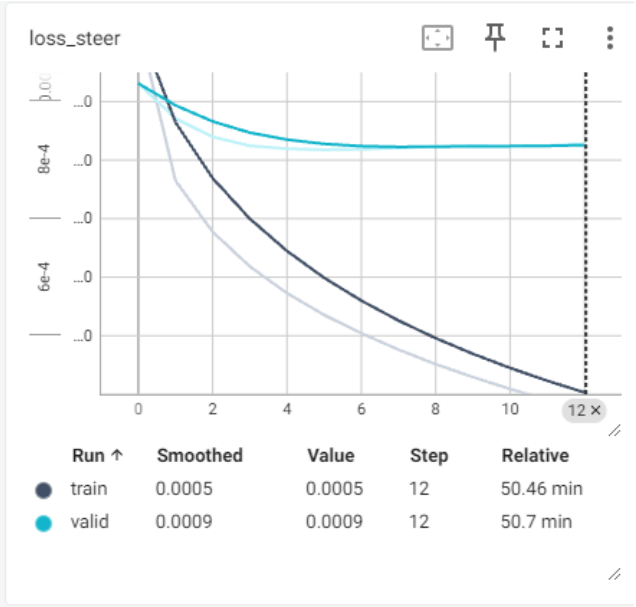
To assess the effectiveness of the proposed state-based deep learning agent, the following metrics were employed:

- Goals Scored: The average number of goals scored by the agent per game.
- Win Rate: The percentage of matches won against the opponents.
- Computational Efficiency: The average time taken for the agent to decide on an action, ensuring it stays within the 50ms time constraint per step.









These metrics not only gauge the agent's performance in the context of the game but also its practicality for real-time application. However, the model did not do well when graded and failed to score any goals as asked for.

```
Loading assignment

Loading grader

 * Match against Instructor/TA's agents

- geoffrey agent                                [ 0 goals scored in 8 games (0:2
0:0  0:3  0:3  0:3  0:3  0:3  0:3) ]

 - jurgen agent                                 [ 0 goals scored in 8 games (0:3
0:3  0:3  0:3  0:3  0:3  0:3  0:3) ]

 - yann agent                                   [ 0 goals scored in 8 games (0:3
0:3  0:3  0:3  0:3  0:3  0:3  0:3) ]

 - yoshua agent                                 [ 0 goals scored in 8 games (0:0
0:0  0:0  0:1  0:0  0:0  0:0  0:0) ]

 ---------------------------------------------------  [   0 / 100 ]

total score                                      0 / 100
```

CONCLUSION

The results indicate that the state-based deep learning agent was not successful in achieving its objectives within the SuperTuxKart ice-hockey environment. The agent failed to demonstrate a competitive win rate and was able to score nil goals while operating within the computational constraints.

Therefore, there are several areas for potential improvement and further research. For instance, incorporating a more complex reward structure could refine the agent's decision-making abilities. Additionally, exploring the integration of transfer learning could expedite the training process and improve performance.

Future work could also involve comparing the state-based agent with an image-based agent to determine the relative strengths and weaknesses of each approach in different game scenarios.

The following shortcomings were identified –

- Data Collection and Feature Extraction: More work needs to be done extensive covering more game scenarios. This data should include a variety of successful strategies and responses to different in-game situations. More relevant features need to be identified and extracted.

- Reinforcement Learning Setup: Since the task involves learning a policy that maximizes goal-scoring, a reinforcement learning approach would be appropriate.
    - Reward Function: Design a reward function that encourages goal-scoring and penalizes actions that lead to losing possession or conceding goals.
    - Policy Learning: Implement a policy gradient method like Proximal Policy Optimization (PPO) or Trust Region Policy Optimization (TRPO) to learn the optimal policy.
    - Value Function: Optionally, use an actor-critic architecture where the value function helps in estimating the expected return from each state, aiding the stability of the training process.

- Transfer Learning: Utilize pre-trained models on related tasks to improve learning efficiency and performance, especially in complex environments where training from scratch can be resource-intensive.

- Ensemble Techniques: Create an ensemble of different models, each with its strengths. For example, one model may be better at defensive play, while another excels in offensive strategies.

- Training: Extensive training of the model to data needs to be done

These techniques should improve the model considerably.

REFERENCES

[1] Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[2] Silver, D., et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.