CS 395T – Deep Learning: Final Project Manuscript

Mohammad Aljubran
(mja3224)

Shubham Bhardwaj
(sb58294)

Nivethi Krithika
(nj6294)

Md Mesbahur Rahman
(mr58937)

Abstract

The authors briefly describe the process of designing an agent to play SuperTuxKart, and particularly compete with the AI oracle (and other classmate AI agents) in a 2v2 hockey game. Our strategy was to maximize puck possession and minimize puck distance to the opponent's goal. Imitation Learning and DAgger could not perform sufficiently well when trained using the AI oracle of the game. Instead, an internal state controller was built and found superior to the AI, where it wins 70% of the time and scores an average of 3.1 goals per game when competing in 2v2 against the AI oracle. Based on supervised learning, a planner was trained to detect puck presence and location. Playing 10 2v2 games, this agent wins 30% of the games and scores an average of 1.2 goals per game. Future work can involve training a DAgger learner on the internal state controller.

1. Introduction

Recently, there has been significant research focus on developing AI algorithms (agents) that learn how to efficiently solve complex tasks through the medium of games. The authors briefly describe the process of designing an agent to play SuperTuxKart, and particularly compete with the game AI oracle (and other classmate AI agents) in a 2v2 hockey game. The agent is constrained to only access the camera image and kart information during the game and allowed a maximum of 100ms/frame on CPU to act, which governs our design decisions and choice of experiments. Our strategy was to maximize puck possession and minimize puck distance to the opponent's goal. This strategy increases the chances of scoring goals and reduces the chances of accepting goals.

In this manuscript, we briefly outline the algorithms we have used to develop our agent. Section 1 highlights a visualization tool we prepared to expedite experimental design during the agent development process. Section 2 sheds light on design experiments using reinforcement learning techniques, i.e. imitation learning and DAgger. Section 3 describes the development of a controller that runs fully on the internal state of the game. Section 4 shows the results of coupling the controller with a planner to yield an agent that does not use the internal state of the game.

1. Tool Development

While hockey games are judged by wins and losses, it is strategy and execution that govern the final outcome. In order to efficiently evaluate our agent's performance at performing different tasks in various scenarios, we developed a visualization tool to display the arena in 2D with minimal computations. Seen in Fig. 1 for instance, one scenario is when our agent is nearby the opponent's goal and has possession of the puck where it is expected to score from a tight angle. To efficiently test the agent at performing this task, it is infeasible to run games all over and await such a scenario to occur. Rather, the proposed 2D visualization tool allows for readily placing the puck and players and visualizing the arena state without actually playing the game. We further enabled this tool to create a full video of the game in 2D visualization without having to store images, which significantly enhanced the efficiency of our experiments.

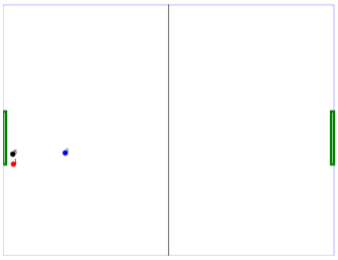

Fig. 1: Game state in 2D generated using the proposed visualization tool. It shows our agent kart (red), opponent kart (blue), and puck (black). This scenario represents a case where we would expect our agent to score.

2. Reinforcement Learning

In this project, we decided that playing the game ourselves would not be viable due to time constraints. Given the SuperTuxKart hockey AI oracle, it was intriguing to simply imitate it using convolutional networks. The most recent five images were concatenated as input and the AI oracle's action (i.e. acceleration, steer, brake, etc.) were used as the training output. While continuous properties (acceleration and steer) were regressed, a threshold of 0.5 was used on network output to classify binary properties (e.g. brake, drift, etc.). Unfortunately, we frequently observed erratic behaviors where the imitation learning agent would get stuck. This can be explained by noting that imitation learning can easily fail when it drifts from what it learned in training. In addition, we discovered that the AI oracle does not perform as well when playing 2v2 games, especially at reversing and actually hitting the puck. This introduced more noise to data and increased the chances of drifting which eventually led to insufficient results in imitation learning.

Alternatively, we chose to experiment with DAgger to alleviate the problem of drift. In DAgger, the agent plays the game while the labels come from the AI oracle. This on-policy approach alleviates the difficulty arising from behavior drifting and data distribution mismatch. While DAgger showed improvement over imitation learning, it did not yield a sufficiently good agent because the AI oracle did not behave as expected in 2v2 games. Hence, we decided to develop our own controller based on the internal state of the game.

3. Internal State Controller

Realizing that reinforcement learning was not viable due to imperfections in the performance of the AI oracle, we decided to

develop a controller that is fully based on the internal state as a benchmark. Using the 2D visualization tool, we set up multiple scenarios and built a microcontroller for each. These microcontrollers were 1) spline, 2) nudge, 3) reverse, and 4) release. Ultimately, these microcontrollers were integrated based on a variety of conditions into a single master controller to compete with the AI oracle. Meanwhile, conditions (e.g. puck is ahead of kart, puck is behind kart, opponent goal is ahead of kart, own goal is ahead of kart, distance from kart to puck, etc.) were determined using the 3D vectors of objects in world view as accessed using the internal state of the game.

The spline microcontroller is a track planner that draws a cubic spline on the 2D kart camera view across three (x,y) coordinates: kart front location, puck location, and opponent goal line location (average of goal post locations is used here). A new spline track is planned for each frame during the game to determine the steering angle of the kart. This microcontroller allows for chasing, overtaking, and scoring whenever the puck is within the front vicinity of the kart. It also controls the kart speed to enable sharp turns. It also allows for turning the puck around to the opponent's goal direction when the kart is found to be facing the puck and own goal.

The nudge microcontroller is used with the puck is very close to the kart (less than 0.1 in 2D camera view). In those cases, steering the puck with spline becomes difficult because the puck does not necessarily follow the steering direction of the kart due to the slippery nature of the hockey field. In those cases, the kart should steer in the opposite direction of the opponent goal and at a sharp angle in order to nudge the puck to the direction of the opponent goal. The kart then resumes spline behavior as the puck gets away. This cycle of spline/nudge controller then repeats till a goal is scored or the corresponding conditions are violated which would invoke another microcontroller.

The reverse microcontroller aims is activated when the puck is not ahead of the kart. It reverses at full speed, and in the direction where the puck was last scene where it keeps a record of puck location in 2D over time. It stops reversing when the puck is within 20 degrees from the kart front axis.

The release microcontroller is used when the puck is determined to be stuck for 15 consecutive frames. It keeps track of the most recent actions taken (spline, chase, or reverse) and moves in the opposite direction for a total of 40 frames. If no progress, then it activates the "rescue" functionality of the game.

It is important to note that there are multiple coefficients and thresholds used in these microcontrollers. Each of these parameters was tuned using grid search, such that the corresponding microcontroller performs its task smoothly. After integrating the microcontrollers, the thresholds of the different conditions which activate the different microcontrollers were also tuned. Playing 1v1 and 2v2 games against the AI oracle based on the default game setup with 1200 frames, this internal state controller wins 3-0 and draws 1-1, respectively. To add stochasticity and further evaluate the performance of this internal state controller, we modified the tournament to randomly change the location of the puck sideways across the centerline at the beginning of every game and right after a goal is scored. We then performed 10 games of 1v1 and 2v2 each to evaluate uncertainty in the controller's performance. The controller won in 80% and 70% of the 1v1 and 2v2 games, respectively, and scored an average of 3.1 and 2.2 goals per game, respectively.

4. Planner

Given the superior results of the internal state controller, we decided to build a planner to predict puck location in the camera view, hence the internal state additional information is no longer needed. To collect training data, we played 2v2 games of internal state controller versus AI oracle. The controller kart was always fixed to be "Konqi" as only these images are used for training. We randomly picked characters to be the AI karts for different games in order to generalize well. We alternated the controller to be team 0 and team 1 in different games, it gets to see both kart colors in training (red and blue). We repeated this method for 20 games capturing 500 frames per game. The puck location is changed randomly at the start of every game and when a goal is scored to allow for more variation in images. Labeling is done based on whether the puck shows in the kart camera view or not, along with the puck (x,y) coordinates. Note that we always eliminated the first 40 frames after a goal is scored because those turned out to be mislabeled. In addition, we had to be careful with the mislabeled cases where 3D vector analysis suggests the puck is not in the camera view when the puck is falling just at the edge of the frame and showing partially.

To be computationally efficient and ensure we meet the 100ms/frame target, we first attempted to train an end-to-end CNN model to perform two tasks: classification (whether or not the puck is in the kart camera view) and regression (x and y locations of the puck). We conditionally allow the regression output of the CNN network to be computed using peak detection if the classification output showed that a puck exists. Otherwise, no regression loss is incurred. The loss function was simple a weighted sum of the classification and regression losses. While this model performed well in classification with 94% F-1 score on test data, it did not do as well in regression with a residual of 0.22. This might be due to the absence of regression signal in many examples where the puck was not present.

To make the CNN model more robust and accurate for detecting puck presence in an image and predicting its location we decided to train two separate CNNs and combine them into a pipeline model. The First CNN classifies the input image for detecting puck's presence in the image. If the first CNN detects the presence of the puck in the image, only then we use the second CNN to regress the location of the puck in the image. The classification model scored 97% accuracy and 96% F1-score on test set while the regression model scored 0.0106 residual on the test set.

We then coupled this planner with the controller. In cases where the planner classification CNN shows low confidence, then we explore the arena using cycles of spline to reverse microcontroller activations, depending on a set of conditions, to limit issues arising from false negative detections. In the rare occasions of having a persistent false positive signal and the kart gets stuck, then the release microcontroller is activated. This agent resulted in good results when challenging the AI oracle. Playing 10 2v2 games, this agent won 30% of the games and scored an average of 1.2 goals per game while requiring only 50±10ms/frame of computations.