UiA University of Agder

Master

Imitation Accelerated Q-learning on a Simulated Formula Student Driverless Racecar

NIKOLAI KJÆREM ELLINGSEN

SUPERVISOR
Morten Goodwin
Martin Holen

University of Agder, 2020
Faculty of Engineering and science
Department of ICT

Abstract

In the international Formula Student competition, only a handful compete in the driverless category. Most of them using expensive hardware such as LIDAR's. By leveraging reinforcement learning, a cheaper camera based system can be created. In order to train this system a simulator based on a fork of Microsoft's AirSim by Formula Technion was used. A virtual replica of a Formula Student car designed for 2020 by Align Racing UiA, functioned as the test vehicle.

In order to decrease the required training time, a pre-trained imitation learning network was used. This was implemented into a Deep Q-Learning network in four different methods. The most successful method was able to accelerate the learning process by 36%.

Table of Contents

Glossary

AI Artificial Intelligence. 4

ANN Artificial Neural Network. 6

AR20 Formula Student race car under construction by Align Racing UiA 2020. i, 12, 14, 15, 20, 23, 27, 28, 31

FSG Formula Student Germany. i, 27, 28

FSUK Formula Student United Kingdom. 10

IL Imitation Learning. 24–26, 28, 29, 31

LIDAR LIght Detection And Ranging. 2, 3, 10, 12

ReLU Rectified Linear Unit. 7, 17

RL Reinforcement Learning. 29

RMSprop Root Mean Square propagation. 8

List of Figures

Chapter 1

Introduction

Driverless vehicles offer many potential benefits such as higher efficiency, reduced cost, universal access and increased convenience compared to human controlled vehicles. In order to be classified as a Society of Automotive Engineers level 4 vehicle, no human input is permitted. This is a particular issue when attempting to operate the vehicle close to its limits of handling for example during collision avoidance or sudden loss of traction. [1]

High-speed autonomous racing can be considered an extreme version of the self-driving car problem, as precise actions in physically complex environments must be performed at a high frequency. Autonomous car racing provides a platform to develop and validate new technologies under these challenging conditions. The formula student driverless competition arranged by the SAE provides an unique opportunity to test state of the art methods against each other under safe conditions. Each team is responsible for both building the car and the autonomous vehicle suite. The sensors found in this suite can range from simple accelerometers to highly complex and expensive LIDAR systems.

By leveraging reinforcement learning, the sensor suite can be simplified to a comparatively cheaper camera and driving dynamics sensors. In the case of image based reinforcement learning, a high correlation between simulator and real-life can only be achieved with sufficient visual fidelity. Reinforcement learning is however highly dependant on lengthy training time, as it learns from essentially zero experience. In general, reinforcement learning has two issues, feature learning and policy learning. In image based reinforcement learning, the agent is expected to interpret the images as well, thus further increasing the training time. It therefore suffers from poor initial performance as it needs to construct the high-level features from raw images.

1.1 Motivation

The competitions arranged in Formula Student provide a great opportunity to test state-of-the art methods in a safe environment. Similarly to the DARPA competition, several teams compete against each other on the same "playing field". As a result, comparisons can be directly drawn between the different methods chosen by the teams. In order to begin development of a self-driving car system, the cost must be kept low. At the moment, few Formula Student teams compete in the driverless competitions compared to the combustion engine and electric engine categories. Most of these rely on LIDAR systems to provide accurate feedback of the track. Some exceptions are Sapienza Corse and Formula Technion. The first of which use cameras to map the track in a similar fashion to a LIDAR system and the latter uses imitation learning from raw images. [2, 3]

Traditionally, complex models are used to provide an approximation of how fast the car can drive through different parts of the course. This does however not account for sudden loss of traction or other disturbances. By using reinforcement learning, that has instead been trained to control the car and correct itself, a higher possible speed can be achieved. This does however require that the simulator is as accurate as possible to optimize potential transfer learning. By using a traditional Deep Q-learning method, training time would be a major issue. Even when used on a simulator, as the policy needs many examples and attempts until convergence begins. By leveraging supervised imitation learning, pre-training of the network and an initial policy can be instilled on a reinforcement network. Thus potentially accelerating the training convergence.

1.1.1 Project Goals

Goal 1: Create a simulation for AI training

Goal 2: Import and configure a Formula Student racecar

Goal 3: Apply imitation learning method from Formula Technion

Goal 4: Implement an image-based Deep Q-Learning method for self-driving

Goal 5: Accelerate the DQN learning process with imitation learning

1.2 Report Structure

This report will outline the background for the methods and technologies used in the project. Current state-of-the will be mentioned briefly followed by a proposed solution and the details of its implementation. Finally, the conclusion and potential future work

Chapter 2

Background

This section will provide some background to the technologies used to complete this project.

2.1 Formula Student Driverless

Formula student is an international engineering competition with contenders from over 600 universities. These are divided into three categories, combustion, electric and driverless vehicles. [4] The concept of the competition is to design and create a prototype formula style racecar which competes in a series of static and dynamic events. The static events consist of a business plan, cost/manufacturing and engineering design. These are as the name implies, events where the vehicle is static. The dynamic events are acceleration, skidpad, trackdrive and finally efficiency.

The trackdrive is the major obstacle during the dynamic events as its layout is the most complex. It consists of ten laps around an unknown track designated by blue and yellow cones. The track is up to 500m with a track width of 3m and distance between cones of maximum 5m. [4] In 2017, the first formula student driverless competition was arranged in Germany with other countries following suit in 2018. Formula Student Germany has previously stated that they wish to remove the combustion engine category and make driverless participation mandatory. [5]

2.1.1 Formula Student Racecar

Each car built for the Formula Student competitions are built in accordance to a strict formula defined by its rule book. Their propulsion system are categorized as either electric or combustion, whereas the driverless vehicles can be either. The vehicles can differ in weight from 100kg to over 300kg, but are generally less than 3m long and 2.5m wide. [4]

2.2 Artificial Neural Networks

An artificial neural network or ANN is, as the name implies, is similar in function to brain neurons. The ANN can be made in both software and hardware, but in the context of this project, software is the main focus. The network itself consists of several interconnected processing elements called neurons, that work together to solve specific problems. In order for this to happen, an input layer is used to send data to the neurons, the neurons then fire off in a specific pattern based on the response and output an answer. In much the same way organic neurons are able to change or modify their function; artificial neurons are able to change what they output in order to learn.

A convolutional neural network is a more specialized neural network that is able to find patterns in large inputs. This is particularly useful for images as they can have both high resolution and several color channels. By multiplying a filter matrix and the input matrix, a feature map can be created. The purpose of the filter can be to for example, perform edge detection, sharpen or blur the image input. An RGB image, for example, can also be converted from 3 channels to 32 by changing the filter size. This allows for greater detection of specific large objects compared to smaller filter sizes. Typically, the stride of the filter matrix is 1, which means that it moves across the input matrix 1 pixel at a time. The stride can be increased to allow the network to search for an overlapping pattern, but finer details are however lost. If an image is too large to process efficiently, a pooling layer can be applied. This pooling layer downsamples the image input into either the max, average or sum of a specified area such as 2x2. By doing this, dimensionality is reduced, but important information is retained.

By connecting every the neurons between every layer, a fully connected or dense layer can be created. This is especially useful in a convolutional neural network as the outputs need to be interpreted. To do this, the final layer of the convolutional

network is flattened and input into a dense layer. Afterwards, the output can be sent to an output layer with an activation function to provide a response.

2.2.1 Activation Functions

The neurons in an artificial neural network need to have a function that defines how they output information. Below, the most commonly used activation functions are shown:

- Linear:$f(x) = x$
- Binary step: $f(x)$
- ReLU: $f(x) = max(0, x)$
- Sigmoid: $f(x) = 1/(1 + e^{-x})$
- Tanh: $f(x) = 2/(1 + e^{-2x}) - 1$

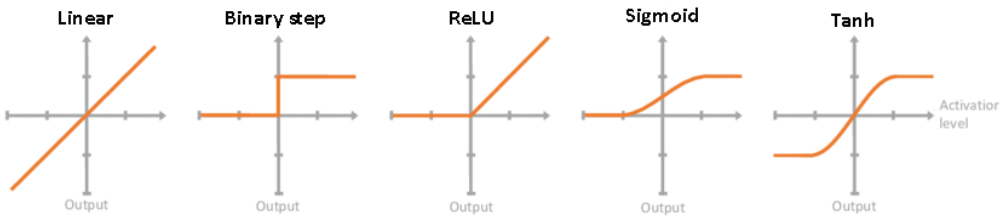

Figure 2.1: Graphed activation functions

2.2.2 Training

In order to train a network, changes must occur. To train as efficiently as possible, an optimizer is used. This is optimization is done by deciding the amount of change in relation to the input. Gradient decent is perhaps the most well known optimizer, it works by calculating the resulting loss when changing a weight. As the name implies, it uses the gradient and momentum to calculate the change in weight. Another way of implementing momentum is used in Adam. By adding fractions of previous gradient to the current one, a more accurate momentum can

be achieved. Both Adagrad and RMSprop are similar to gradient decent, but adagrad uses individual learning rates for different weights and RMSprop only lets gradient accumulate momentum in a fixed window.

2.2.3 Imitation learning

Imitation learning techniques seek to imitate a given behaviour by recognizing and reproducing similar actions. This behaviour can either be given from a demonstrator or demonstrations. This allows it to learn the ideal action on the specific states provided in the demonstrations. As a result, performance scales with the demonstrations given. If a state in which no exact demonstration exists, the actor will attempt to perform a similar action.

2.2.4 Reinforcement Learning

By interacting with an environment, the reinforcement learning agent is able to learn the results of its actions. The actions are chosen either on the basis of past experiences, or by attempting new actions. This is called exploitation and exploration respectively. After each action, a numerical reward is given based on the environment feedback. The agents goal is to maximize the accumulated reward over time based on trial and error.

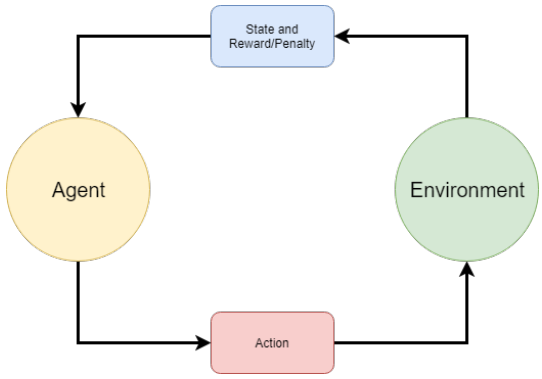

Figure 2.2: Reinforcement learning

2.2.5 Deep Q-Learning

Deep Q-learning uses a neural network to approximate the Q-value function. Unlike regular Q-learning, only the state is input and the neural network outputs a Q-value for each of the actions that are possible. Essentially, if given a state, it will return the Quality of each action instead of just one Q-value. The action with the highest Q-value can then be chosen. Similarly to regular Q-learning, the network is able to learn without any pre-existing knowledge or policy. It is however reliant on a reliable scoring system from the environment.

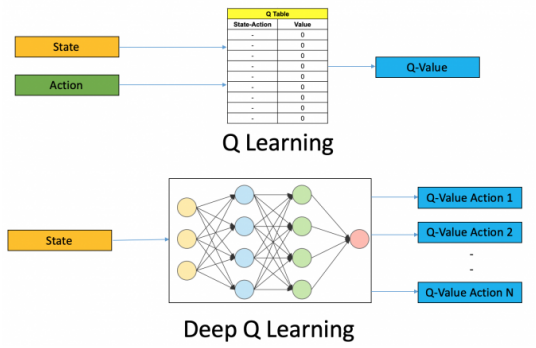

Figure 2.3: Q-learning and Deep Q-learning

2.3 Simulator

2.3.1 Unreal Engine

Unreal is a game engine that is programmed using C++ . It is able to support both 2D and 3D games and has a plethora of tutorials and existing code available. Unreal engine was chosen as it provides high fidelity environments with little work. It is also highly customizable to suit most needs for the project. [6]

Chapter 3

State-of-the-art

Due to the time investment needed as well as large startup cost, few teams are competing in the driverless categories. In 2019 FSUK out of 81 teams competing 3, were driverless and 0 raced their own car. The competition is higher in Germany, where 20 driverless teams competing out of 99 total. There were however only 5 teams that were able to actually complete all events. Based on recent advances in AI such as OpenAI, DeepMind and the ever growing presence of automotive companies in the field. It is of course not expected that student teams are able to compete at the cutting edge, however, many of the methods that are available have not been utilized yet. [7, 8, 9]

The majority of competing teams currently use either a LIDAR and/or Image based SLAM to locate the track cones. Afterwards, the locations are superimposed on a virtual environment and methods such as Model Predictive Contouring Controller are used to control the car. In general, most of the top teams in the driverless competition use methods similar to MPCC with a state-of-the art car dynamics model. In the field of deep learning however, there seems to be untapped potential. Formula Technion has already shown that an imitation based deep learning method is viable. It should therefore be reasonable to assume that a pure reinforcement learning method can be viable as well. [3]

One of the major issues is however the required training time. By using a simulator, this can be made more viable. However, since reinforcement learning is often performed in near real-time, the training time is still an obstacle. Using human examples to improve training is a fairly old concept, imitation learning however

has gained more interest in recent years.

A landmark paper in this field is DeepMind's Deep Q-Learning from Demonstrations (DQfD) [saus]. Their method pre-trains the DQN with human demonstrations to closely imitate the demonstrator. Alpha Go similarly trains a supervised learning actor on human demonstrations. The supervised learner is then used to initialize the reinforcement learning network. How this can be applied to an existing Deep Q-learning network, especially one used for autonomous racing, will be explored in this paper. [10, 11]

Chapter 4

Proposed Solution

This project will attempt to create an estimated model and simulation of a planned formula student car, AR20, from Align Racing at the University of Agder. Three different methods will be used to provide the driverless system of the car.

4.1 Simulator

For this project, Microsoft's Airsim will be used to provide the simulation aspect. The specific fork of Airsim is created by Formula Technion and includes their 2018 Driverless contender and setup. Modifications have however been made to make it more compatible with reinforcement learning. The most important of which is the map rotator and checkpoint system. The simulator can be used to train drivers and show off the car during recruitment as well. Therefore, additional improvements have been made to the user interface and plug-and-play compatibility for various input methods. [12, 13]

The simulator was chosen as it provides high-fidelity environment with great support for modifications. AirSim includes a weather system, that affects both the visuals and physics. In addition to this, AirSim also includes, color cameras, depth cameras, distance sensors and a LIDAR implementation. It should therefore be more than adequate for future driverless vehicle development as well. Traditionally, its used to simulate urban environments, but due to it being based on Unreal, major modification can be made.[12]

Figure 4.1: Race Track environment from PolyPixel

4.1.1 Maps

The maps used during testing are the same as used by Formula Technion which were provided by PolyPixel. It consists of a high fidelity racetrack environment, with an additional flat asphalt plane. The main bulk of the training was performed using an asphalt plane as it closely resembles the competition venue in Formula Student. Several cones are placed to outline the track and invisible walls are placed between them to ensure that the car does not leave the track. The car is not able to detect the walls, as they are only used to trigger overlap events. A penalty is then given to the car, its position reset and an end of episode flag is set.

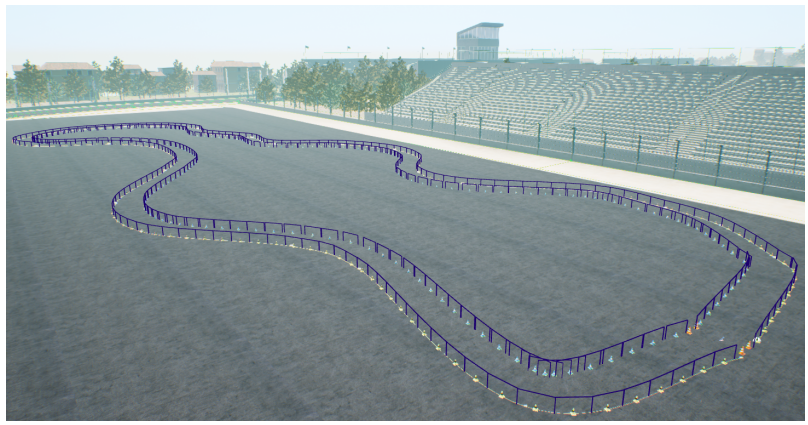

Figure 4.2: One of the maps used for training

4.1.2 Car

The car used in the simulation is based on and adjusted to imitate AR20, a formula student car designed by Align Racing for the 2020 season. By comparing prior vehicles, design spec sheets, as well as competitors with similar designs, an approximation of its performance has been created. The basis of the vehicle is the same method that was used by Formula Technion, an Nvidia Physx vehicle. This allows for the model to use the same gear ratios, torque/horsepower curves, suspension profiles etc. as AR20 is currently designed. Some simplifications have been made regarding the aerodynamic downforce and general temperature effects to allow real-time simulation. Fine tuning of the car was performed using video and driver feedback based on prior cars from Align Racing UiA as well as competitors. The final model performs and acts as is expected from its design. Ideally, this should be validated against the real car, but due to the ongoing COVID-19 pandemic, the manufacturing of the car has been delayed. [13]

In order to get accurate training data for a potential deployment on the real car, the model should be visually identical to real-life. Since Align Racing UiA uses computer-aided design software to design the car, the same model has been converted to function in the simulator. The CAD model is however massive in both size and complexity since it contains every single component including screw threads and resistors on circuit boards. It was therefore simplified, converted to a mesh and skinned to be visually similar to the finished and painted AR20. A skeletal system is then applied to the mesh to allow the wheels to rotate as well as the steering wheel. The skeleton contains six nodes, a center mass node and one node for each wheel and steering wheel. This is important since the tyres can be deformed and compressed, but they should not be affected by the suspension of the car. Additionally, the sprung weight of the car would be 20 kg higher if the tyres are added, thus greatly affecting handling.

Two models of the car exist, one for driverless and one with a human driver. The driverless model weighs around 230 kg, which is the best case weight with additional hardware. Due to the addition of a human, the weight of the other model is 70 kg higher. This is again the best case weight based on expected drivers.

Figure 4.3: Model of AR20 in simulator

In order to function with the DQN, a discretized action space is applied. It consists of the following potential actions: Set throttle 0.4, brake 0 and steering 0

- Set throttle 1 (Not available for IM)
- Set brake 1 (Not available for IM)
- Steering of 0.25 steps from -1 ($-27°$) to 1 ($27°$)
- Do nothing (Steering 0)

4.1.3 Sensors

A single camera with a 60° field of view is mounted to the main roll hoop of the car with a 20° angle downwards. This is done since most of the valuable data is below the horizon. Due to the height of the camera location, this also provides better parallax which should aid the car in determining depth. The camera image is a color image of size 144x255, that is cut into 61x184 to show the most important details. In addition to this, the velocity and lateral acceleration in the Y-axis is input on a lower layer of the network.

Considering the vantage point, the cameras were placed on the main roll hoop, above the driver's seat in the car, see Figure 4. This offers the advantage that the occlusion among cones is reduced to a minimum and even the cones placed one behind the other (in line of sight) can be perceived sufficiently well.

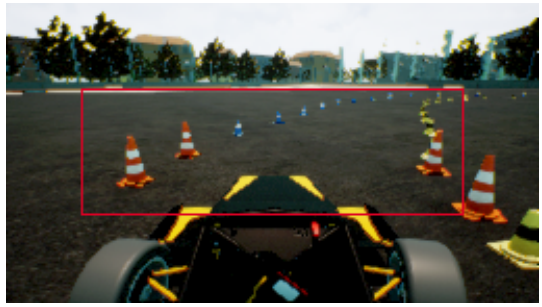

Figure 4.4: View from camera with cropped area in red

4.1.4 Simulator

The simulator is setup as a Markov Decision Process issue, where a state st and stlm are observed every time step t. An action at is then performed which leads to state s_t+1 and s_tlm+1 and a reward r_t = R((s_t,s_tlm),a_t). The transition between each action is defined as T(s_t,s_tlm,a_t,s_t+1,s_tlm+1)

Each time step t corresponds to an image st from the environment as well as the lateral acceleration in the Y-axis and the current speed in any direction as s_tlm. After the state has been processed by the network, a response a is returned. This does however mean that it is possible for large state changes if the network takes too long to respond. To mitigate this, a semi-discrete method is used. This works by pausing the physics of the simulator when a state is received, the simulator is then unpaused after the action has been sent. This does however cause some issues with the physics engine which can cause the car to jump upwards when hitting a cone. This is mitigated by making the cones static and resetting the car if its Z-acceleration is too high, no negative penalty is given in this case.

4.2 Driverless System

In order for the car to drive autonomously, three methods have been used. They are all based on neural networks, the first of which is a recreation of an imitation learning method used by Formula Technion in 2018. The latter two are based on deep Q-learning, a subset of reinforcement learning.

4.2.1 Network

Two slightly different networks will be used to allow efficient training of both the imitation and reinforcement learning algorithms. Nvidia's PilotNet has already been proven to work by Formula Technion with their imitation learning solution. As a result, the same general setup will be used for the reinforcement learning as well. [13]

It consists of 5 convolutional layers, with a dropout layer with probability 0.5 between convolution layer 3 and 4. After convolution layer 5, the output is flattened to 1x11776 and input into 3 dense layers. This differs a bit from PilotNet as the additional layer of 1164 is missing. During testing, this proved to be unnecessary to achieve good results and caused additional response delay. The networks used differ a bit after the flattening as the reinforcement learning network receives an additional input. This input is the telemetry data from the car, namely speed and Y-acceleration.

Regarding activation functions, all convolutional layers use ReLU as negative values wont improve the performance during feature recognition in the images. It is also comparatively cheap regarding computation compared to other activators such as Sigmoid.

In order to provide an additional temporal axis, a stack of 4 grey image were tested in place of the RGB image. This allows the network to infer the movement of the car directly from the images. The accuracy of the cone detection drops dramatically as it now essentially gets one real time image and 3 images from the past. Unlike the RGB method which has three identical images to consult. Additional networks were tested to improve this, but in general, the method of injecting telemetry data was chosen instead due to performance.

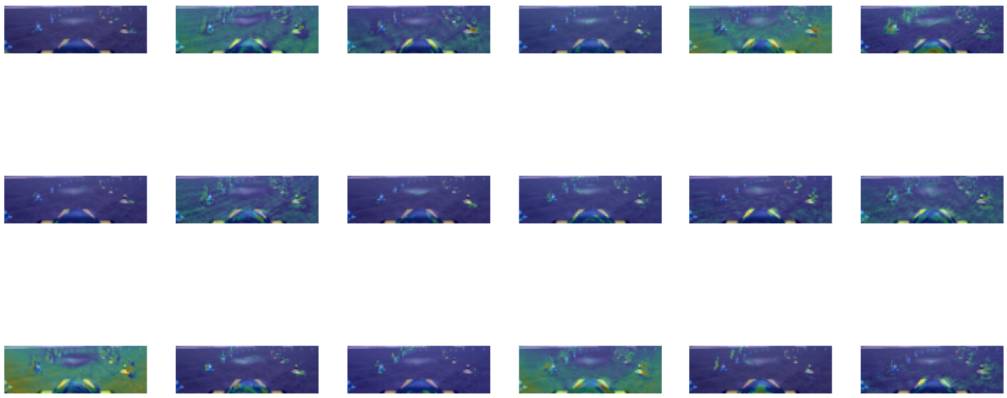

Figure 4.5: Some heatmaps of activations from last convolutional layer

The network used by Formula Technion is a modified version of PilotNet from NVIDIA. It consists of 3 convolutional layers, before a dropout layer with keep probability of 0.5. This is followed by two additional convolutional layers and 3 fully connected layers. [13]

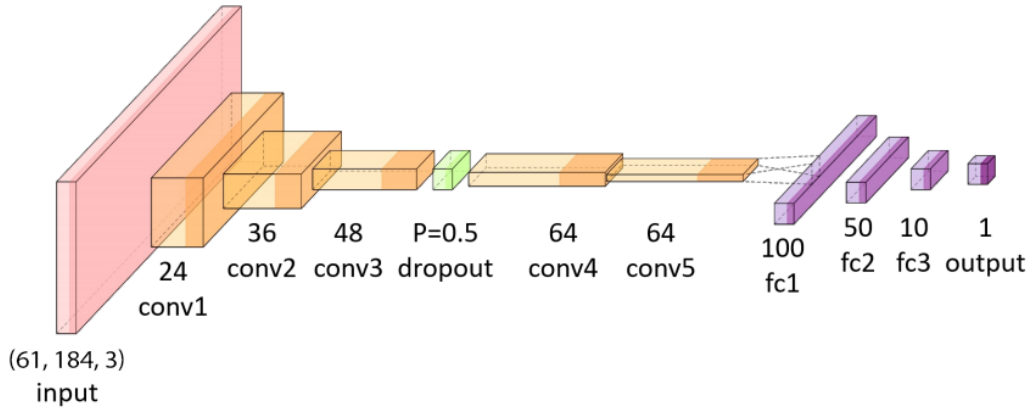

Figure 4.6: Modified PilotNet from Formula Technion

When applying the same network to DQN training, one of the fully connected layers have been dropped. The two remaining fully connected layers are instead increased in size.

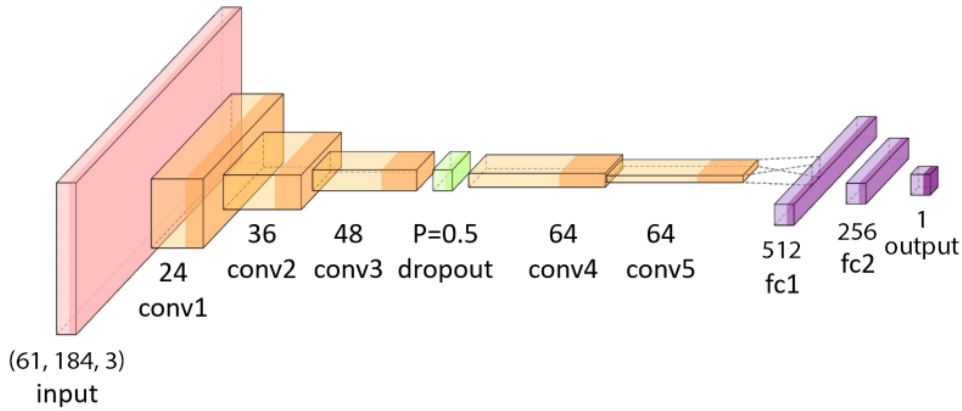

Figure 4.7: Modified PilotNet used in DQN

4.2.2 Scoring

The reward is based on the current forward velocity of the car. It is measured in meters per second and should therefore provide an approximation of the distance travelled as well. In order to promote careful driving and avoid crashes, the cumulative reward in each episode is given as the reward. In the case of the DQN, this allows it to measure the change in reward to maximize our Q value. Since the car is reset after every crash, it also resets the episode reward.

The maps are too narrow in relation to its 27° maximum steering angle and it can therefore not turn 180° around and drive backwards. If the car hits a cone or the invisible boxes between the cones. The car and score is reset to 0. This is however only the case during training. When testing, collisions are penalized, but the car is not reset. Several checkpoints are placed around the track which start and stop a timer function in the simulator. If a lap is completed in the correct checkpoint order, the lap time is displayed and saved as an additional performance metric.

4.2.3 Technion Imitation Learning

The method used by Formula Technion is based on an older version of both Airsim and Unreal engine and their trained model is therefore no longer performing ideally. The same method used to train their system has therefore been repeated. Data was recorded from five different maps over the course of 250 minutes using the AR20 model. The data consists of several images and corresponding inputs given to the car, the recorded data was then cooked and trained using the same method as Formula Technion.

4.2.4 DQN

The DQN setup starts by filling the replay buffer without updating the weights during an observation phase. Random actions are performed throughout the observation, which consists of 50000 frames. One frame corresponds to an input and an output from the network. This does however mean that the training time may influence the time between frames. A semi-discrete setup was therefore used for the simulation, which pauses the simulator between responses. When transitioning to the exploration phase, DQN actions are chosen if a random number is smaller than epsilon. Epsilon is also annealed from 0.5 to 0.001 over the course of 200000 exploration frames.

During training, the network samples mini-batches of size 32 from the replay memory to train on. The reason for this was to allow the network to respond quicker as training is performed on every frame. With higher batch sizes, for example 64, the estimated training time would be several days as the simulator would run at less than 1 action per second. With a batch size of 32, this is increased to 2.5 actions per second. The reason for this is that for each time step, the network is used to discover the max Q values of the current mini-batch. Then a forward pass to calculate the error value, before updating the weights. The performance impact on the network is minor with a low mini-batch, but the training efficiency in relation to real time is much higher. This is however not an issue when not training.

4.2.5 Imitation Accelerated DQN

In order to accelerate the DQN learning, the same DQN setup was used and imitation actions introduced in three different ways. The first method works by allowing training during the observation phase and letting the imitation learning system control the car instead of random actions. This also means that the replay buffer contains imitation examples that are then sampled by the experience replay. Since the imitation learning setup outputs a steering angle, it has been discreetized at 0.25 steps from -1 to 1. The second method is to seed the actions with responses from the imitation learning system. When the a random number is lower than epsilon, which is annealed from 1 to 0.01, it has a 50 percent chance to choose an imitation action rather than a random action.

Due to the examples given to the imitation learning system, it essentially tries to keep itself in the middle of the track. This can be leveraged as part of the reward function. When an action is performed that is similar to what the imitation system would have chosen, a minor reward of 0.01 is given. These methods can also be combined to further aid the training of the DQN.

Chapter 5

Results

This chapter will outline the process used to test the networks and their performance.

5.1 Introduction

Each training step corresponds to the amount of telemetry packets received from the unity socket. The decay rate was set to 0.99 since the system is highly reliant on consecutively performed actions. The epsilon was set to 0.5 and decreased to 0.001 after a specified amount of exploration frames. A mini-buffer of size 50 and replay memory of 500 was used as well as an initial 500 observation frames for every run to populate the replay memory. To test the car, four maps of increasing difficulty were used. Modifications to this setup was performed based on the maps used for training.

All testing was performed locally, but during training, the UiA servers were used for the imitation learning method. It is possible to get actions directly from a remote server to the simulator running locally, but latency impedes the training process. Generally, the delay between an image being sent and a response received has been kept at less than 500ms. During testing AMZ found that their greatest bottleneck was up to 500ms in their vision pipeline. The simulator and code used for this project can however run at less than 5ms per action if needed. In real-life this performance is expected to be closer to the 500ms experienced by AMZ. [14]

5.2 Result of Imitation Learning

A major limitation of imitation learning is that the actor is only able to perform as well as the behaviour it is imitating. Since the data is gathered from the car at a fixed speed and fed into the imitation learning system, a major increase in speed is not viable. Formula Technion used a simple function to decrease the throttle in relation to the steering angle, thus increasing the speed when going straight. This is however an issue on the model of AR20 as it changes gears based on the current RPM. When the throttle is decreased in a corner and the RPM subsequently drops, a gear change is initiated. This causes the car to "jump" and become unsettled, as the new gear is initiated and the gear ratio changes. This is the same behaviour that is expected of a car in real life. Due to this, the gear system is locked into gear 1, which is the same as when recording.

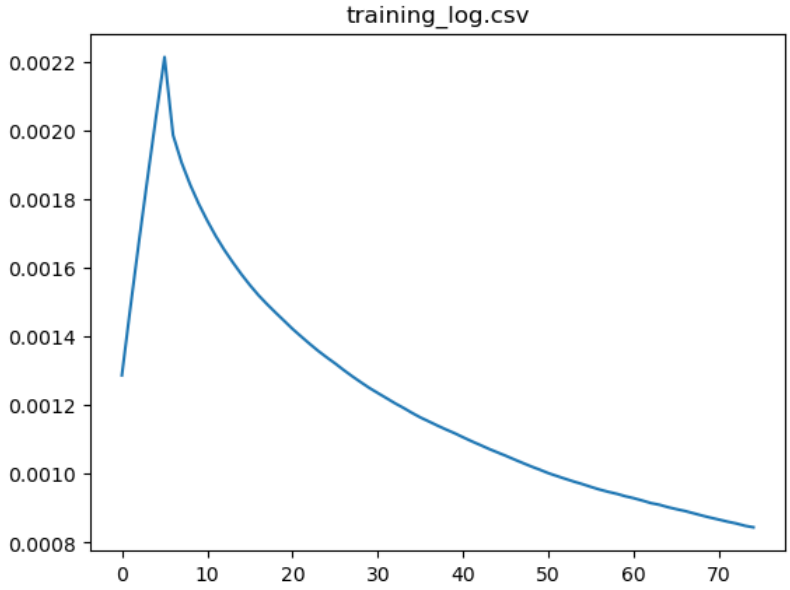

Figure 5.1: Loss per epoch during imitation training

The performance of the imitation learning method is as expected. It was trained on a server provided by the University of Agder over the course of one hour. After 73 epochs, the network is fully trained and able to be deployed. When applied to the simulator, it is instantly able to navigate the environment with a fairly reasonable score. When trained locally, the results were similar, but the training time was greatly increased.

The performance of the IL is a relatively straight line. The network is sometimes fooled by simulated lens flares which causes it to deviate. This deviation can cause the network to fail if an example of the specific state was not present in the training data.

5.3 Result of DQN

When training the DQN, unlike the imitation learning method, the simulator must be directly interacted with. As a result, the training time required is greatly increased. The network was first given an observation phase of 25000 timesteps, followed by 100000 frames of observation/training. This allowed it to navigate the track with partial success, as it still collided with a few cones. Since the reward given is the cumulative reward for each episode, and the reward is set to 0 when a collision occurs. This means that the potential loss is massive. As a result, the braking action was chosen too frequently and many time steps were wasted. The highest quality action does however change after a few time steps of zero reward increase, which causes the car to move again. The rewards start at around 1400 after the observation phase ends. Afterwards a minor increase in rewards occur consistently until time step 100000.

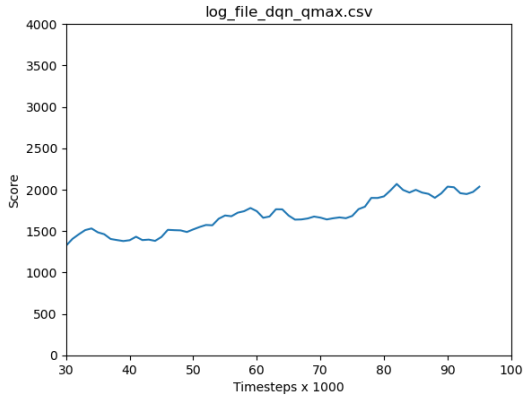

Figure 5.2: Reward from DQN on time step 30000 to 100000

5.4 Result of Accelerated DQN

The exact same network as used in the DQN method was used for the imitation accelerated DQN as well. A total of four methods were tested.

Pre training: The network is pretrained with examples from the imitation learning network.

Seeding: When selecting a random action, there is a 50 percent chance to select an action from the imitation learning network instead.

Q-Imitation: The DQN is scored positively if it performs the same actions as would have been chosen by the IL network.

Combined: All prior methods combined.

In order to test the different methods, a smaller run of 100000 time steps was performed. For each run, only one of the methods were applied, with the exception of the combined method.

Pre training offered an initial improvement burst that did not persist for more than 25000 time steps. It was however useful in stabilizing the scores at a higher level compared to the regular DQN. Seeding however, had the biggest improvement as it essentially gives the DQN a close to ideal action to perform. This in as sense solves the reliability issue when training a DQN. Since the action in a regular

Method	Improvement
DQN	0%
Pre training	5%
Seeding	31%
Q-Imitation	22%
Combined	36%

Figure 5.3: Improvement of methods in percent after 100000 timesteps

DQN is random, there is a possibility that an ideal action is never performed. This subsequently causes the network to train a non optimal policy.

The Q-Imitation method is similar in its final result, although the training method is far more difficult. Since the reward function only returns a reward if the action was similar to the IL and not the speed, the seeding method has an advantage. However, when all these methods are combined, their combined improvement is better than the individual methods.

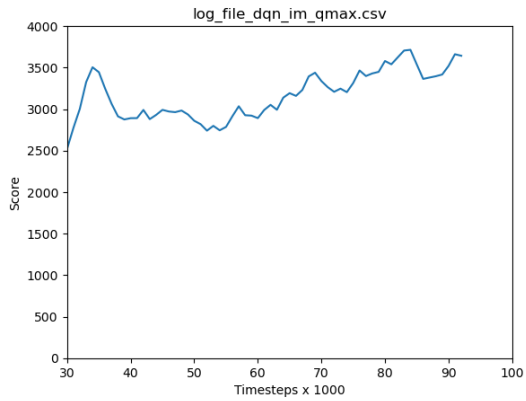

Figure 5.4: Reward from Combined method DQN on time step 30000 to 100000

When comparing the graphs of figure 5.2 and figure 5.4. A clear difference can be seen. In graph X, the starting point is substantially higher, although the growth rate is fairly similar. Since the imitation accelerated DQN already has a fairly good policy, a large increase in the reward is therefore harder to achieve. And the growth rate is therefore more or less similar across the 100000 time steps.

5.5 Lap Time Metric

In order to provide an accurate performance baseline, the same map as used in Formula Student Germany 2018 has been recreated. It will however only be an approximation and the car will also not be identical to the vehicles used during the competition.

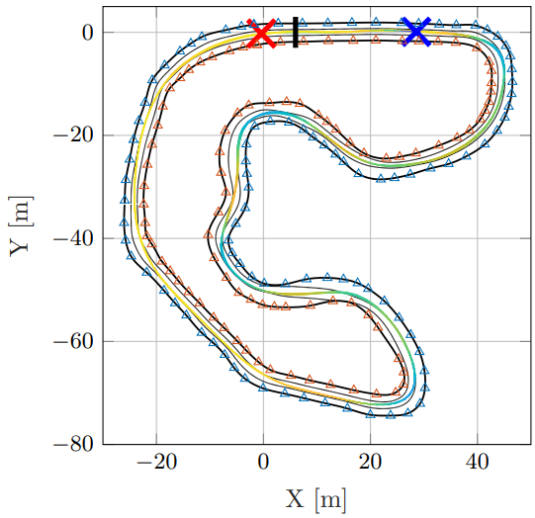

Figure 5.5: Map from FSG 2018 trackdrive discipline [14]

The best time during FSG Driverless was achieved by Akademischer Motorsportverein Zürich (AMZ). The map used in figure X is from their paper "AMZ Driverless: The Full Autonomous Racing System" which includes the lap times they achieved on their simulator. When deployed on the actual track in 2018, they were able to achieve an error of only 1% between the simulator lap time and real life.

Since the car used by AMZ is substantially different than AR20, notably since AMZ uses an electric car. This allows for much greater acceleration, in their case, a 0-100 km/h time of under two seconds. AR20 is expected to be closer to 4 seconds. Despite this, it is still interesting to see how the solution in this paper compares to the current top team in Formula Student Driverless. [14]

Controller	Time (Seconds)
AMZ	28.5928
IL	55.1013
DQN	47.4523
Accelerated DQN	44.8932
Human	31.7732

Figure 5.6: Average time from 10 laps on FSG map from 2018

The imitation learning setup is severely limited by the fact that it has been trained to stay in the center of the track. This does however mean that it is able to avoid the cones more successfully, at the cost of speed. It would however be able to get a better lap time if the control method is optimized further. Currently, it only accounts for steering angle and sets the throttle cautiously to avoid drifting.

The DQN was able to perform a bit better than the IL method since it is able to increase its throttle. It is however extremely prone to crashing, but was given less than ideal training time.

When using the IL to accelerate the DQN, the results were very similar, although the probability of crashing was lowered. Generally, it seems like the setup has gained more stability due to the more efficient training.

The best time achieved by AR20 was from a human driver, which is somewhat to be expected. It is however possible that minor improvements can be made, but given the performance of the human drivers, large improvements are unlikely. This therefore makes it an appropriate lap time to compare against the others.

Chapter 6

Conclusion

By applying a pre-trained IL network to an existing RL network via the four different approaches, a clear improvement can be made. In the context of autonomous racing this is especially the case as the general policy can successfully be inferred on the RL network. By training the IL to stay in the middle of the track, it also successfully functioned as part of the reward function. Since the IL is trained to function by itself, its also possible to train it further to provide a solid baseline performance for other reinforcement learning methods.

The DQN setup is still not fully trained and has room for more improvements. Event though the training time was greatly reduced, it is still substantial. It could therefore be an improvement to move the simulator to a more powerful server or send actions from the server to a the local simulator via RPC.

The straightforward solution of performing supervised learning on demonstrated data and applying it to the DQN was not ideal on its own. The IL network was therefore used during the DQN learning phase as well. The result of this allows the DQN to outperform both the IL network and demonstrations given if enough training time is provided.

The convolutional network chosen for this task proved to function well, but it is perhaps more complex than necessary. Since PilotNet is a fairly general network, which is most often used for urban driving. In the context of autonomous racing, the environment is much more sparse, nevertheless, it was able to successfully detect the cones.

The network is greatly affected by latency and delay when training, a more efficient layout would help to mitigate this. Additionally, the code used for this project is based on an old version of tensorflow which should be updated. It is still sufficient, for the purpose of this thesis.

In order to control the car, the throttle, braking and steering is used as input. This is however not ideal as it causes a major increase in training time compared to using a static throttle. When using a simplified driving model which controls the throttle based on current steering angle, similar to Formula Technion. The required training time does not increase substantially, it can therefore be assumed that something similar to the bicycle type model used by teams such as AMZ can be applied. This does however mean that the driving model must be fine tuned, unlike the DQN which learns by itself. [14]

Chapter 7

Future Work

Initially, the system was going to be tested on an Nvidia JetBot as a proof of concept. But this was sadly not possible due to the covid-19 pandemic. Regarding the real-life AR20, which was scheduled for release in April 2020. Its construction has been delayed and some minor changes may occur as it will be modified to compete in 2021 instead. This additional development does however also means that there is a greater chance for more driverless specific system to be implemented. There is still a massive amount of work to compete in Formula Student Driverless, but the methods used in this paper should facilitate future development.

Since the IL has the ability to learn directly from gathered data in the simulator, it should therefore also be able to learn from real-life data. Which will further enhance the accuracy of the simulator via comparison with direct demonstrations.

Even though a partial parallax effect is achieved by placing the camera high on the car, additional cameras should also be considered. This can be especially useful if cameras with different field of views are used. Similarly, a depth camera can be used instead or in conjunction with regular cameras. This was however not explored further, in order to reduce cost in the planned real world proof of concept.

Even though the temporal dimension was perceived by the DQN via additional telemetry data injected into the fully connected layers, more research into this should be performed. In order to increase the performance further in edge cases under high speed.

Since the current simulator is non-continuous, a DQN performed adequately. AirSim is however designed to be continuous, and it would therefore be interesting to see how a gradient based method such as DDPG or possibly PPO functions with the same imitation acceleration method. A more advanced version of a DQN such as Rainbow would also be interesting to test, as it can provide better results at the cost of additional required training time.

References

[1] Society of Automotive Engineers *"Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles"*, Available: https://www.sae.org/standards/content/j3016_201609/ Accessed: 20 January 2020

[2] Sapienza Corse *"Sapienza Corse"*, Available: http://www.sapienzacorse.it/ Accessed: 12 December 2019

[3] Formula Technion *"Formula Technion"*, Available: https://www.formulatechnion.com/ Accessed: 12 December 2019

[4] IMechE *"Formula Student Rules"*, Available: https://www.imeche.org/events/formula-student/team-information/rules Accessed: 20 January 2020

[5] Formula Student Germany *"Formula Student Germany"*, Available: https://www.formulastudent.de/ Accessed: 20 January 2020

[6] Unreal *"Unreal Engine"*, Available: https://www.unrealengine.com/ Accessed: 14 December 2019

[7] Imeche *"FSUK Driverless results"*, Available: https://www.imeche.org/docs/default-source/1-oscar/formula-student/2019/results/fs_uk_2019-results-ai-ads-overall.pdf?sfvrsn=2 Accessed: 14 December 2019

[8] Imeche *"FSUK Driverless results Self built vehicle"*, Available: https://www.imeche.org/docs/default-source/1-oscar/formula-student/2019/results/fs_uk_2019-results-ai-ddt-overall.pdf?sfvrsn=2 Accessed: 14 December 2019

[9] Formula Student Germany *"FSG Driverless participants"*, Available: https://www.formulastudent.de/fsg/results/2019/ Accessed: 14 December 2019

[10] Todd Hester and Matej Vecerik and Olivier Pietquin and Marc Lanctot and Tom Schaul and Bilal Piot and Dan Horgan and John Quan and Andrew Sendonaris and Gabriel Dulac-Arnold and Ian Osband and John Agapiou and Joel Z. Leibo and Audrunas Gruslys *"Deep Q-learning from Demonstrations"*, Available: `https://arxiv.org/abs/1704.03732`, 2017 Accessed: 14 December 2019

[11] David Silver and Aja Huang and Christopher J. Maddison and Arthur Guez and Laurent Sifre and George van den Driessche and Julian Schrittwieser and Ioannis Antonoglou and Veda Panneershelvam and Marc Lanctot and Sander Dieleman and Dominik Grewe and John Nham and Nal Kalchbrenner and Ilya Sutskever and Timothy Lillicrap and Madeleine Leach and Koray Kavukcuoglu and Thore Graepel and Demis Hassabis *"Mastering the game of Go with deep neural networks and tree search"*, Available: `https://research.google/pubs/pub44806/`, 2016 Accessed: 14 December 2019

[12] Shital Shah and Debadeepta Dey and Chris Lovett and Ashish Kapoor *"AirSim"*, Available: `https://arxiv.org/abs/1705.05065`, 2017 Accessed: 14 December 2019

[13] Zadok, Dean and Hirshberg, Tom and Biran, Amir and Radinsky, Kira and Kapoor, Ashish *"Explorations and Lessons Learned in Building an Autonomous Formula SAE Car from Simulations"*, Available: `https://arxiv.org/pdf/1905.05940.pdf`, 2019, Accessed: 14 December 2019

[14] Juraj Kabzan and Miguel de la Iglesia Valls and Victor Reijgwart and Hubertus Franciscus Cornelis Hendrikx and Claas Ehmke and Manish Prajapat and Andreas Bühler and Nikhil Gosala and Mehak Gupta and Ramya Sivanesan and Ankit Dhall and Eugenio Chisari and Napat Karnchanachari and Sonja Brits and Manuel Dangel and Inkyu Sa and Renaud Dubé and Abel Gawel and Mark Pfeiffer and Alexander Liniger and John Lygeros and Roland Siegwart *"FSG Driverless participants"*, Available: `https://arxiv.org/abs/1905.05150`, 2019, Accessed: 23 April 2020

Appendices

The code and simulator is available on:
`https://github.com/Align-NikolaiEllingsen/IKT590`