– Independent Work Report Spring 2020 –

Reinforcement Learning Applied to Autonomous Driving and Multi-Agent Traffic Environments

Bobby Morck
Adviser: Karthik Narasimhan

Abstract

With the growing popularity of autonomous driving, many problems encompassed by this area of study have arisen that lend themselves well to reinforcement learning. This paper explores partially observable environments where multiple vehicle agents learn to autonomously navigate a given environment or control traffic in fixed track environments. For this task, I study the impact that Proximal Policy Optimization (PPO) versus a Deep Q-Network (DQN) has on the ability of a single agent or multiple agents to navigate a given environment or decreasing traffic (measured by increasing velocity). Previous models have made similar attempts to solve this problem and achieved impressive benchmarks with respect to decreasing the density of traffic and increasing velocity; however, this study builds on past ideas by assessing the ability of vehicle agents to generalize learned behavior from one traffic environment to another. This paper finds similar results to previous benchmarks are achieved using PPO and DQN within the individual traffic environments, and generalization performs near baseline levels (without learning) in most scenarios.

1. Introduction

The presence of autonomous vehicles (AVs) in our society is growing rapidly. Current estimates predict that the number of AVs used will grow to 51,000 by 2021 and then catapult to 1 million by 2025. [8] Consequently, many have looked into the potential benefits that AVs have to offer our society as a whole. Some of these benefits include a reduction by 45% in the total road transportation energy demand [11]; however, potentially the most notable is the expected reduction in traffic. With the increase of AVs within our roads, it is expected that "traffic jams and lost time are reduced," likely due to the decreased headways needed between vehicles and increased traffic densities enabled

by AVs allowing for greater overall traffic flow. [4] Specifically, current models predict that capacity limits can be improved within city traffic by 40% and by 80% in highway traffic. [4]. Due to the expected benefits of the current literature and models and as autonomous driving is a problem that can be modeled using Reinforcement Learning (RL), the primary goal of this project is to train multiple vehicle agents using RL on various different traffic environments so that they can alleviate or eliminate the traffic within them as a result of learned behavior. As vehicles within the real world see many different traffic environments, to make this project have realistic applications, it is another goal to have vehicle agents learn behavior from one traffic environment and generalize their learned behavior to another.

To achieve this goal, the results of prior work should be understood. Many past works have built on the idea of the multi-agent RL environment including recent projects by OpenAI [2] [6], which have displayed impressive emerging behavior from collaborating agents. Most notably, the FLOW project has made similar attempt towards developing a framework for multi-agent RL traffic environments. The FLOW project utilizes the Simulation of Urban Mobility (SUMO) interface for creating the traffic environments and networks, examples of which can be seen in Figure 1. FLOW's suite of environments includes single-lane and multi-lane environments, some of which contain intersections and traffic control signals. As a result of their experiments, substantial increases from baseline metrics with zero RL-trained agents were archived using Trust-Region Policy Optimization (TRPO) and various amounts of RL-trained agents. [12]

This project will introduce a novel multi-agent environment traffic environment to attempt to achieve similar benchmarks as seen in the FLOW project. Within the FLOW project, agents were only trained on a single traffic environment, without understanding how learned behavior can generalize across environments. This project will address this gap by applying the model of agents in one traffic environment to another. Specifically, within the project, the single-lane ring and figure eight traffic environments will be developed and explored. Initially the approach was to develop environments where agents not only control acceleration, but also direction of travel, allowing agents to have more control than within the FLOW project. Due to the inability of agents to

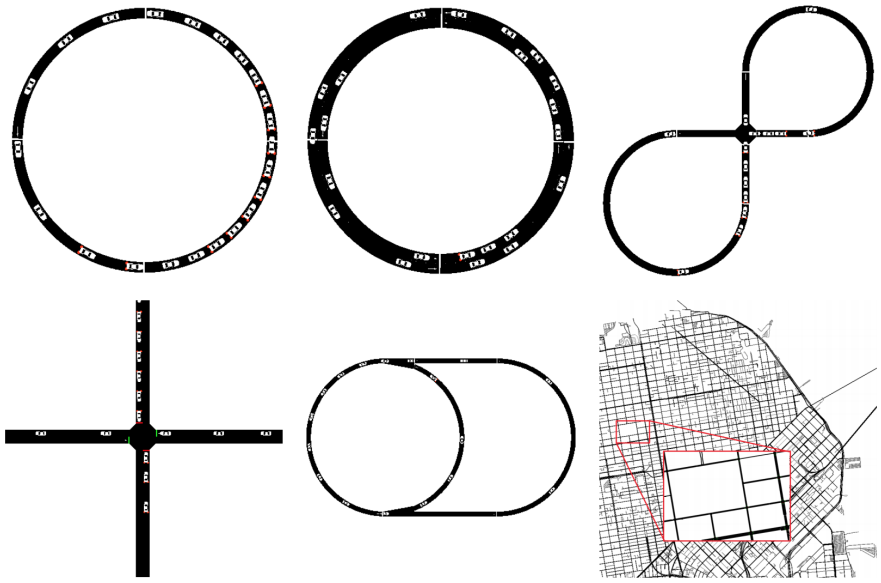

Figure 1: Example of traffic networks utilized within the FLOW project containing single lane and multi-lane environments. [12]

learn to navigate the entire environment, the approach transitioned to using fixed-track environment, similar to those built using SUMO. Within these environments agents can only control magnitude of acceleration and direction of travel is constrained by the environment. In terms of implementation, a python based game development library, PyGame, was used to create the traffic environments that interface with RLLib, which is a RL library that provides various training algorithms. For both the ring track and the figure eight track, a suite of single and multi-agent environments was created having 1, 2, or all vehicles being trained by RL. This project aims to apply newer algorithms, Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN) to this problem and compare results to those achieved using TRPO. For generalizing, the DQN model of the ring tracks were applied to the figure eight track environments.

Experimentation involved applying both the DQN and PPO algorithms to each of the traffic environments. Following initial experimentation, the DQN model of the ring track was applied to the figure track to test the ability to generalize. Within the ring track environment, DQN performed better than PPO in almost all cases and in the case with all RL-trained agents, DQN achieved optimal performance. Within the figure-eight track, DQN again performed better than PPO in almost all cases, though still performed near baseline levels (without any RL-trained agents). In

generalizing, all cases performed near baseline levels. Though the single-agent case using the model trained on the ring track performed better on the figure-eight track than agents using a model trained on the figure eight track.

2. Background and Related Work

2.1. Markov Decision Processes (MDP)

Reinforcement learning is largely grounded in the idea of the Markov Decision Process (MDP). Within this process there is a set of states (S), a set of actions (A), a reward function that maps states, S, and actions, A to real numbers, a transition probability distribution between states, a discount factor of future rewards $\gamma \in (0,1]$ [12], and a policy that maps the states, S, to actions, A. The decision process is as follow: Agent(s) have an initial state and take an action based on the policy, π, which will update the state and return a reward to the agent based on this updated state. Then the agent will take another action based on the polcy evaluated at this new state. It is important to note that all states in this paradigm are Markov States that are assumed to adhere to the Markov assumption that each state is only dependent on the state directly prior. [3] In determining this policy, the value function $V(s)$ is often used to help shape the policy. This value function essentially signals how "good" a state is and is equal to the expected total reward from a given state. From this value function, some policies use the action-value function, $Q(s,a)$ where $s \in S, a \in A$, for shaping the policy. This action-value function is an indicator for how good it is to be within a certain state $s \in S$ while choosing an action $a \in A$ and is equal to the expected total reward from state s by doing action a. An example of an MDP can be seen in Figure 2.

This project will utilize the previously described MDP, but will include two different parameters as this project utilizes a partially observable markov decision process (POMDP) as agents do not have full observation of the underlying state after making each action. These two parameters are the set of observations Ω and the observation probability distribution O. For the purposes of this project it is not necessary to understand the difference in the formulation of the decision process; however, it is useful to understand that when the state is updated, the agent

receives an observation $o \in \Omega$ that depends on s, a and its probability is given by $O(o|s,a)$. [13]

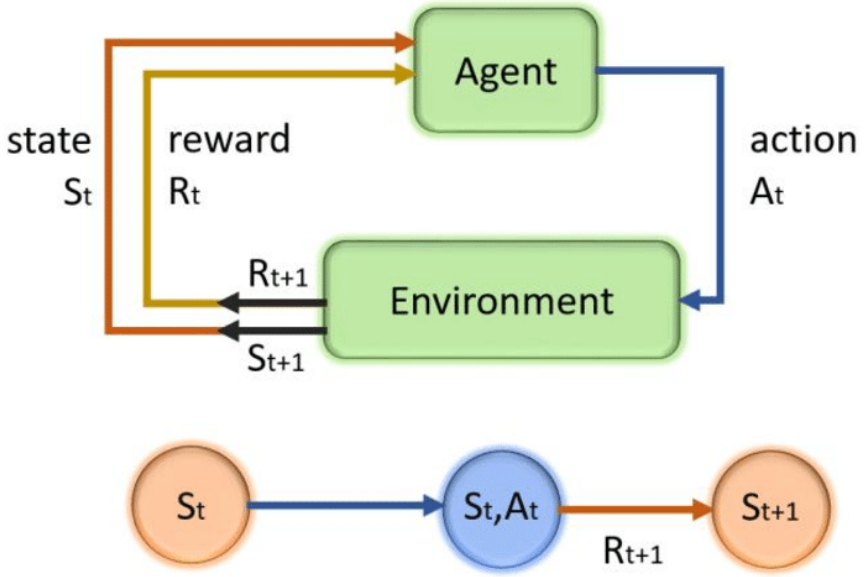

Figure 2: Visualization of the MDP displaying state and actions made by an agent.

2.2. Related Work

This project uses multi-agent environments in which the state updates after each agent uses an action chosen from a shared policy and then each agent receives an observation of the state. Within the field of multi-agent RL, OpenAI has made recent advances. Lowe et al. introduces multi-agent deep deterministic policy gradient (MADDPG). Although the specific implementation details of MADDPG are beyond the scope of this paper, Lowe et al.'s results display the some of the challenges of multi-agent RL and a potential solution to these challenges. Due to multiple agents making actions at each step of the environment, the environment is non-stationary from the perspective of each agent, meaning that the state is not dependent solely on the actions of that agent. This violates the Markov assumption described previously. Lowe et al., however, propose a centralized action-value function that takes in the actions of each agent and the observations of all agents as input. [6] This essentially re-validates the Markov assumption as the all agents have access to a policy which considers the observations of all agents and as "the environment is stationary even as the policies change." [6]. Using MADDPG, agents performed better than when

using other decentralized policies in the defined cooperative and competitive games.[6]. Building on this work, Baker et al. of OpenAI utilize this same idea, except in this project Proximal Policy Optimization (PPO) is used to estimate future rewards. All agents used a shared policy and during training, the policy had full observability of the state. Baker et al. found greater sample efficiency using this shared, fully observable policy in the hiding agents learning the optimal behavior within a hide-and-seek environment than when using separate, decentralized policies for each agent. [2] The project explored in this paper will focus on the shared policy but will not provide the policy with full observation, unlike Baker et al.

Many multi-agent RL environments have been built and studied for regulating traffic signal control. Arel et al. developed a framework for scheduling traffic lights to minimize delay at an intersection. In this environment the intersections act as agents whose actions are combinations of traffic light signals that would allow certain lanes to safely cross the intersection without causing collisions. [1] A Deep Q-Network (DQN) was applied and achieved substantial improvements over a standard Longest Queue First (LQF) algorithm, especially as the rate of traffic arrival at each intersection increased. [1] This project does improve the flow of traffic; however it does so through controlling traffic signals rather than through the vehicles, posing itself as a separate problem than by minimizing traffic through AVs.

The FLOW project is the first to study vehicular moderated traffic flow and is the most applicable study to the project studied in this paper. Within this project, Wu et al. create a modular framework for training vehicle agents to improve the traffic flow and increase the average velocity of all agents in various traffic environments (examples of which can be seen in Figure 1). The Simulation of Urban Mobility (SUMO) interface is used for implementing and visualizing the various traffic networks and RLLib is interfaced with for training the vehicle agents. The FLOW project focuses on both mixed autonomy and fully autonomous traffic environments, mixed autonomy meaning that only some of the vehicle agents are being controlled by RL. All vehicles, those controlled by RL and those not controlled by RL, only control the magnitude of acceleration and not direction of travel. Traditional car following models (CFM) exist that attempt to mimic

human driving behavior for a given vehicle using only information about itself and the vehicle in front. Specifically, the Intelligent Driver Model (IDM) is the CFM used for controlling the acceleration of non-RL controlled vehicles. The formula for controlling the acceleration of vehicle i is as follows

$$a_i = (1 - (\frac{v_i}{v_o})^{\delta} - (\frac{H(v_i, \Delta v_i)}{h})^2), \text{ where } H(v_i, \Delta v_i) = s_o + v_i T + \frac{v_i \Delta v_i}{2\sqrt{ab}} \text{ [12]}$$

Within this model, v_i is the velocity of the current vehicle, v_o is the velocity of free flow traffic, Δv_i is the difference in velocity of the current vehicle and the vehicle in front, T is the minimum time to reach the vehicle in front, h is the current headway between the current vehicle and the vehicle in front, s_o is the desired headway between vehicles, a is maximum vehicle acceleration, and b is a constant for determining breaking deceleration. [12] It is not necessary to completely understand the differential equations or the solution, but it is important to understand the general behavior resulting from IDM. This essentially results in deceleration when too close to the vehicle in front and acceleration when the vehicle in front is at a safe distance. The project explored in this paper will follow a heuristic similar to IDM to achieve this behavior in non-RL controlled vehicles.

To understand the results of the FLOW project it is first necessary to examine the state space, action space, and reward function. For the ring and figure eight tracks, the "state consists of a vector of velocities and positions of each vehicle in the network", where the positions are all relative to a defined point in the track. [10] However, the observations of each vehicle is limited to only "local vehicles and aggregate statistics, such as average speed." [12]. The action space is a list of accelerations for each agent, where the acceleration is a real number chosen from the continuous interval $[a_{min}, a_{max}]$. [10]. Lastly, the reward function is $r := max(||v_{des} \cdot 1^k|| - ||v_{des} - v||, 0)/||v_{des} \cdot 1^k||$, which essentially aims to achieve an objective of high velocities of all vehicles, where v_{des}, v are vectors of length k that contain arbitrarily large velocity elements and the velocities of all vehicles respectively. [10] Using this setup and the Trust Region Policy Optimization (TRPO) algorithm (described in more detail in section 2.3), the FLOW project was able to achieve the results shown

in Figure 3, 4. Within the single-lane ring track (shown in the top left of Figure 1) and the figure eight track (shown in the top right of Figure 1) there were 22 vehicles total and 14 vehicles total respectively. From these results, we see substantial increases from baseline metrics (using 0 RL-trained agents) with these metrics improving the more agents that are trained in RL. Within the ring track, a 27.07% improvement was achieved using 11 RL-trained vehicles. Within the figure eight track, a 150.49% improvement was achieved using all 14 RL-trained vehicles. [12]. Within the ring track, experiments displayed agent's ability to learn to both stabilize and "platoon." The stabilization occurs when agents learn to slow to a stop during cascading traffic and then accelerate such that all vehicles reach a constant velocity. The "platooning" refers to RL-trained vehicles learning to require less headway than those controlled by IDM, resulting in greater average speed of all vehicles. Within the figure-eight track, vehicles traverse the intersection using a right of way policy implemented by SUMO. Using a single RL-trained vehicle, the traffic caused by the right of way policy is stabilized in a similar manner as in the ring track. With all 14 RL-trained vehicles, the agents learn to cross the intersection without as much deceleration while avoiding collisions. [12]

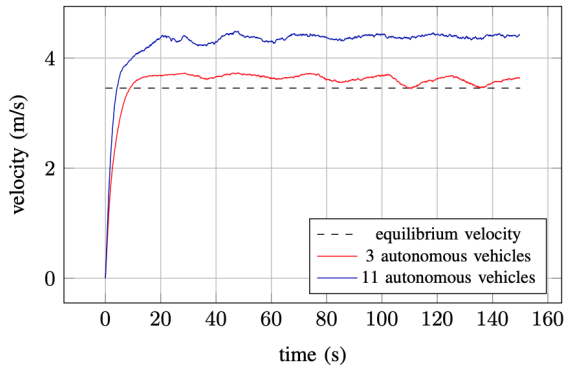

Figure 3: Average velocity of all vehicles from FLOW ring track using 3 and 12 RL-trained agents. Equilibrium is achieved using 0 RL-trained agents. [12]

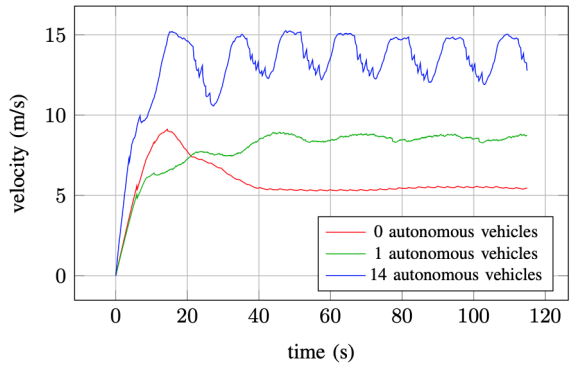

Figure 4: Average velocity of all vehicles from FLOW figure eight track using 0, 3, and 14 RL-trained agents. [12]

The FLOW project presents impressive benchmarks within vehicular moderated traffic flow; however, FLOW does not apply the models of the vehicle agents from one traffic network to another.

This project aims to fill this gap in order to understand how learned traffic control laws generalize from one traffic environment to another while applying newer and different algorithms.

2.3. Algorithm Overview

This project will use Proximal Policy Optimization (PPO) and a Deep Q-Network (DQN) for all experiments. Before understanding PPO, it is useful to understand TRPO and the default policy gradient algorithm to highlight the advantages of the PPO algorithm. We can first examine the objective function of the default policy gradient algorithm. [9]

$$L^{PG}(\theta) = E[log_{\pi_\theta}(a_t|s_t)A_t]$$

The policy π_θ is a neural network that outputs a stochastic policy, meaning that given the state as input, it outputs the weights for the set of actions with the weights corresponding to the probabilities the agent will take each action with. A_t is the advantage function and is equivalent to expected total reward from the current state subtracted from the discounted sum of rewards from the current state given the current action ($A_t = Q(s,a) - V(s)$). The policy gradient performs gradient ascent on this objective to converge to the optimal policy; however, it is possible that the policy update can be too large and the policy will not converge to ideal behavior. To combat this, TRPO implements a similar objective function subject to a constraint. [9]

$$\text{argmax}_\theta \;\; E[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t], \quad \text{subject to} \quad E[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]] \leq \delta$$

TRPO attempts to solve the problem of a large policy update by constraining the magnitude of policy update using the KL constraint seen above. The problem with this method is that this introduces additional computational overhead in the optimization. PPO solves this problem by introducing the constraint of the policy update directly into the objective function. [9] If we let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$,

the objective of PPO becomes [9]

$$L^{CLIP} = E[min(r_t(\theta)A_t, clip(r_t(\theta), 1+\varepsilon, 1-\varepsilon)A_t)]$$

With this objective function, ε is a hyperparameter and this essentially clips $r_t(\theta)A_t$, when too large.

The next algorithm that will be explored is DQN. At a high level, DQN scales traditional Q-learning to larger action and state spaces using a neural network to approximate the action-value function. Unlike policy gradient methods, DQN does not use a stochastic policy but instead has the agents use the action corresponding to the largest Q-value in the output of the neural network. The neural network approximates action-value function as defined by the Bellman equation.

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a')$$

$r(s,a)$ is the empirical reward from taking action a at state s and γ is the discount factor for future rewards. If $Q(s,a)$ is the optimal Q-function then we can determine which a' will maximize $Q(s',a')$ and, in turn, maximize $Q(s,a)$. The neural network approximates $Q(s,a) \approx Q(s,a,\theta)$, where θ are the weights of the network, and optimizes its weights by minimizing the loss function [7]

$$L = (y_i - Q(s,a,\theta))^2$$

y_i is the target, defined by the Bellman equation above, and $Q(s,a,\theta)$ is the estimate provided by the neural network. Lastly when training a DQN, experience replay is utilized, where the agent's experiences from taking actions are stored and updates to the policy is made by taking batches from this memory. [7] This can cause DQN to be more sample efficient than PPO, as PPO only relies on the most recent step, not storing experience. In determining an action, DQN uses an ε-greedy approach, where $\varepsilon \in [0,1]$. In this approach, the agent selects a random action with probability ε and an action decided by the policy with probability $1-\varepsilon$. This ε value decays over of the course

of many steps, reducing random actions. [7] These random actions allow the agent to explore the environment, collecting experience in the process.

3. Approach

3.1. Autonomous Driving

Initially, the approach was to build two fully autonomous environments where agents would control not only acceleration, but also their direction of travel. Contrary to the previously described FLOW project, this would more realistically model autonomous driving as AVs would learn to navigate their environment while also controlling traffic. To understand whether agents would feasibly be able to learn a policy that controlled navigating the environment correctly and improving traffic flow with many vehicles present in the environment, it is necessary to break apart the problem and understand each part separately. The first step is to determine if agents can correctly navigate the environment. To achieve autonomous navigation of the environment, the approach is to build an environment similar to those in Figure 1 and train a single agent using DQN and PPO. The vehicle agent would have observation of its environment given by sensors protruding from the body of the vehicle that would measure the distance of the closest object (either a barrier of the track or other vehicles). The reward function would follow a checkpoint system similar to Kurach's proposed method where checkpoints were used to aid the shaping of the reward function for training agents in a soccer game. [5] The agent would receive reward for traversing checkpoints sequentially. As further discussed in section 5.1, this approach proved infeasible, so a new approach was explored.

3.2. Fixed Tracks

This new approach involved building environments similar to those within the SUMO interface, where vehicles only determine the magnitude of acceleration and not direction of travel. Specifically, these environments would follow the structure of the the single-lane ring and figure eight track seen in Figure 1. The environment would have full control over the direction the vehicle is traveling, ensuring it follows the track. Similar to the previous approach, all vehicles would be given partial

observation of the environment through sensors protruding from the vehicle, measuring the closest distance to other objects. These sensors would enable the implementation of a Car Following Model (CFM) in non-RL vehicles as they can measure distance to other vehicles. These sensors also do not provide observations to the agents that are specific to the environment, so these observation spaces should provide the ability to generalize the learned model from one fixed track environment to another. In addition, the two environments chosen would have different control laws as the figure eight track contains an intersection, while the ring track does not. The ring track, without an intersection, can be seen as representing highway traffic and the figure eight track, with an intersection, can be seen as representing city traffic. As the goal of this project is to understand how vehicles trained in one environment will perform in another, testing the ability to generalize using tracks that simulate two common, yet quite different, traffic scenarios will provide a good understanding of how training in one environment applies to other traffic environments.

In terms of training, as with the first approach, DQN and PPO will be used. As shown in section 2.3, PPO provides significant advantages to traditional policy gradient algorithms as it clips the policy update. TRPO provides this advantage as well; however, PPO includes this within its objective function, so the training is much more computationally efficient than TRPO. Therefore, evaluating the performance of the vehicle agents using PPO could display the advantages of PPO over TRPO, which was used in the FLOW project. [12] DQN has also been shown in section 2.3to be more sample efficient than policy gradient algorithms, such as PPO, as it uses experience replay in the training process. Due to this, comparing the results of PPO to DQN could display this advantage as well. Lastly, the agents will use a shared policy, similar to the policy developed by Lowe et al. [6], where the policy will take as input the actions of each agent, as this was shown to produce better results than decentralized policies in multi-agent environments.

4. Implementation

Prior to creating the environments, the tools in building the environments and interfacing with RL training algorithms needed to be decided. A python-based game development library,

Pygame, was chosen to build the various tracks. Other alternatives such as Unity were explored, but the vast number RL libraries and machine learning resources available in python along with the easy to understand documentation provided by PyGame, made PyGame the ideal tool to build the traffic environments. The PyGame traffic environments interfaced with RLLib to train the vehicle agents. RLLib provides a similar interface to the more common OpenAI Gym, but is more suitable for multi-agent training. Built-in training algorithms are provided that allow for agents to use a shared policy. Within these training algorithms, RLLib provides the ability to tune the hyperparameters to achieve the desired results. Using these tools, it is possible to create multiple traffic environments and train agents to moderate the traffic within them, potentially learning general behavior that can be applied to multiple traffic environments.

4.1. Autonomous Ring Track Implementation

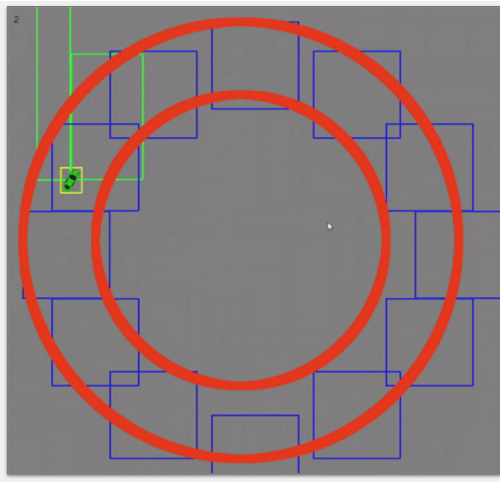

Figure 5: Autonomous ring track environment using one agent. The green boxes displayed are the front 3 sensors. The blue boxes scattered about the track and the checkpoints the agent needs to reach to receive a reward.

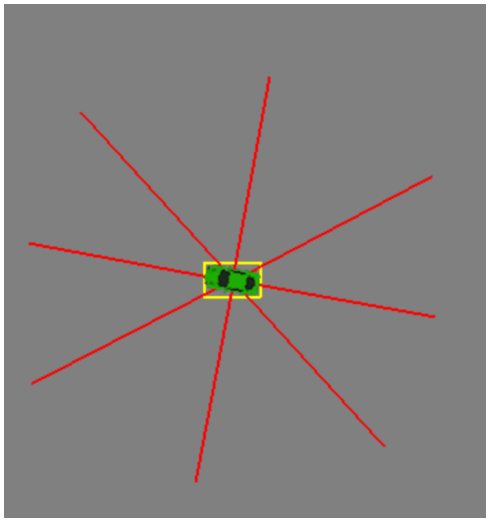

Figure 6: Vehicle implementation displaying car sensors. There are 8 sensors protruding from the vehicle at 45 degree angles. Sensors displayed in red for visualization.

The first environment built was the autonomous ring track. The track consists of two circles of different radii that are treated as barriers of the track. There are 13 square checkpoints scattered between the barriers of the track. The vehicle is treated as a smaller rectangle and is initially placed within a checkpoint. As mentioned in 3.1, the vehicle has 8 sensors protruding from

it that give it observation of the environment. Functionally, the sensors are treated as the bounding rectangle, orthogonal to the game screen, of the sensor lines displayed within Figure 6. These bounding rectangles can be seen within Figure 5 along with the implementation of the autonomous ring track.

Next, we can define the MDP of this environment. The action space of the vehicle agent is a set of 5 discrete actions, corresponding to accelerating, decelerating, turning right, turning left, and doing nothing. As the action space is discrete, the amount the vehicle can accelerate by or turn by is a preset value. The agent has the ability to do nothing as for optimal navigation of the environment, the agent does not need to accelerate or turn at each step. It is important to note that the agent can decelerate but the minimum velocity of the vehicle is 0, ensuring it cannot travel in reverse. The observation space is defined as a vector of 10 elements, corresponding to the distance to the closest objects detected by each sensor from the vehicle, the maximum checkpoint the vehicle has reached, and the time elapsed within the episode. Furthermore, each checkpoint is assigned a value from 1 to 13 starting from the checkpoint where the vehicles starts. The reward function assigns a reward of 1 for each new checkpoint reached by the vehicle sequentially by their assigned values. After reaching a new checkpoint and being assigned a reward for reaching it, the agent will not receive additional reward until it reaches the next checkpoint sequentially. As discussed in section 3.1, the reward function followed the ideas presented by Kurach et al., where checkpoints were used to help shape a reward function that encouraged exploration of the environment. [5]. Moreover, other reward functions were explored which included a weighted sum of the maximum checkpoint and the distance the closest barrier of the track to incentivize driving further from the barriers. However, the evaluation discussed in section 5.1 refers to the previous reward function. The done condition, the condition in which an episode terminates, is set to true whenever the vehicle collides with a barrier of the track or reaches a checkpoint with a value lower than the value of the maximum checkpoint reached thus far.

In training the vehicle, the hyperparameters for each algorithm were set appropriately. In training the DQN model, the ε parameter is set to 1 initially and decays to 0.15 after 100,000 steps.

With this slow decay and a high final ε value, the agent will choose to make random actions at a relatively high proportion, encouraging exploration of the environment. The experience replay buffer was set to the default value of 50,000 experiences and the training batch size was set to the default value of 32, where the policy used 32 elements of the experience replay buffer to train after each iteration. The default hyperparameters were used in the PPO training, where the value function clip parameter default value is 10. In both algorithms, the discount factor, γ, was set to 0.85.

4.2. Fixed Ring Track Implementation

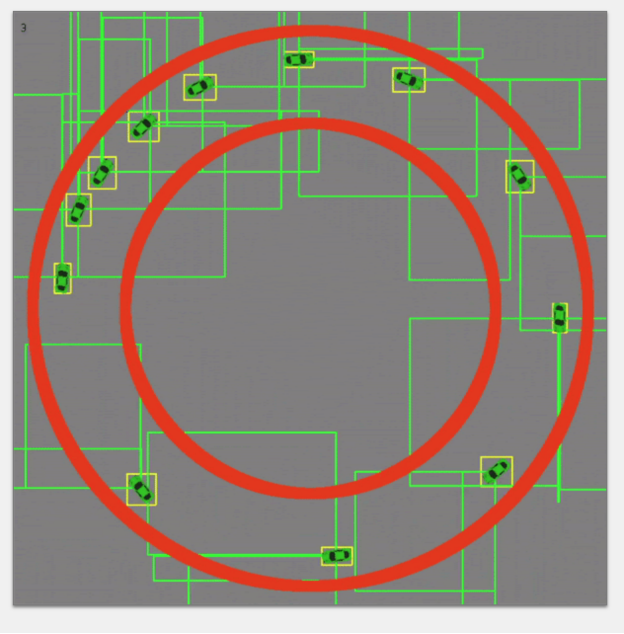

Figure 7: Fixed ring track environment with 12 vehicles. The green boxes displayed are the front three sensors of each vehicle.

The implementation of the fixed ring track environment is similar to the previous environment. The environment consists of two circles of different radii representing the barriers of the track. The vehicles are constructed the same as in the previous section with the same sensors displayed in Figure 6. Within this environment however, both RL and non-RL controlled vehicles do not control the direction of travel. The direction is maintained by having the vehicles travel in the direction tangent to a circle with the center point being the center of the track. The vehicles that are not learning follow a CFM similar to the previously described Intelligent Driver Model

(IDM). Within this model, the vehicles monitor the velocity of the vehicle in front of them. If a vehicle in front is within a distance of 75 units (a unit being the length of a pixel within the PyGame game screen) and is traveling slower than the current vehicle, then the current vehicle will half its speed at each step until its speed is less than or equal to the vehicle in front. When a vehicle in front is greater than a distance of 75 units away from the current vehicle or has a velocity greater than the current vehicle, then the current vehicle will accelerate. In addition, traffic is simulated by the following: all vehicles (RL and non-RL) begin at the maximum velocity (10) and after 20 steps from the beginning of an episode, a predetermined vehicle will slow down to a stop over the next 20 steps. Following this, it will follow the CFM described above or follow actions decided by the policy if it is controlled by RL. This will cause cascading traffic as seen in Figure 7.

Next, we can describe the MDP for this environment. The action space is defined as a set of 3 discrete actions: accelerate, decelerate, or do nothing. As with the previous environment, these actions are discrete and the amount the agent accelerates or decelerates is a preset value. The done condition is set to true after 500 time steps or when an RL-vehicles collides with another vehicle. It is important to note that although the vehicles controlled by RL do not follow the CFM described above, they will slow down a vehicle within 75 units in front is traveling slower than the current vehicle regardless of the action decided by the policy. This is done to avoid collisions as slowing down will minizinze collisions, allowing for more steps per episode. The observation space is similar to the previous observation space as it is a vector of 10 elements. The first 8 elements correspond to the distance to the closest vehicles detected by each sensor (and not to the barriers of the track unlike the previous environment). The next two elements are the velocity of the current vehicle and the average velocity of all vehicles. The observation space does not consider the distances to the barriers of the track as these measurements would be specific to the environment, potentially decreasing the ability to generalize from one track to another. Lastly, the reward function assigns the average velocity of all vehicles as reward to each agent at each step. Although this is not entirely the same as the reward function used in the FLOW project, described in section 2.2, this aims to achieve a similar goal of increasing average velocity. This is desirable as increased

average velocity of all vehicles within the environment directly corresponds to a decrease in traffic or increase in traffic flow.

With this environment setup and MDP, 3 environments were created each with 12 vehicles total, but with differing amounts of RL trained vehicles. The environments had 1, 2, and 12 RL controlled vehicles respectively. Within the PPO training, the default hyperparameters of the RLLib PPO implementation were used, except for the clip parameter of the value function. The policy is updated based on the cumulative reward over the course of the episode (adding together the rewards from each agent as well), so the values are of a high magnitude. Consequently, the clip parameter was set to 3000, 6000, and 36000 for each environment respectively. Within the DQN training, final ε value was set to 0.05, and the decay rate of ε was set depending on the environment. As the environments with more agents involve more total steps, the ε decay was set to occur over 20,000, 40,000, and 240,000 steps for each environment respectively. The discount factor γ was set to 0.99 in all environments.

4.3. Fixed Figure Eight Track Implementation

The implementation of the figure eight track environment can be thought of primarily as two separate ring tracks, connected by a two-way intersection. The vehicles are constructed in the same manner described in section 3.1. The direction of the vehicles is also controlled by the environment within this implementation as well. As shown in Figure 8, the vehicles travel counter-clockwise in the lower ring and clockwise in the upper ring. The environment controls the direction of each vehicles by separating the track into 4 distinct regions: the upper ring, the lower ring, the path connecting the upper ring to the lower ring, and the path connecting the lower ring to the upper ring. Depending on which section of the track the vehicle is in, its direction is controlled by a separate rule. The non-RL vehicles follow the same CFM as in the ring track, except within this environment, there is an additional control law implemented at the intersection. This control law is similar to the one provided by SUMO [12] and provides a right of way queuing system. Essentially, if the front 3 sensors of the car detect another vehicle while the current vehicle is within the intersection

(displayed in Figure 8 as the blue square at the intersection), then the vehicle is placed on a queue. The vehicle at the front of the queue is the allowed to proceed and is removed from the queue once it leaves the intersection.

Next the MDP of the figure eight track can be understood. The action space of the RL controlled vehicles is the same as in the ring track: 3 discrete actions representing acceleration, deceleration, and doing nothing. The observation space is also the same for all agents: a vector of 10 items corresponding to the distance to the closest vehicles detected by each sensor, the velocity of the vehicle, and the average velocity of all vehicles. The action and observation spaces must be the same so that the model from the ring track can be applied to the figure eight track. It is important to note that the RL controlled agents slow down when within 75 units away from a vehicle in front that is driving slower than it, and follows the queuing system at the intersection. The RL controlled vehicles are only added to the queue if they detect another vehicle within 30 units of distance, however, to allow for interweaving traffic in scenarios of many RL controlled vehicles. The done condition is set to true when 1000 steps in the current episode have passed. Lastly, as with the ring track environment, the reward function assigns as reward the average velocity of all vehicles at each step.

Using this MDP, 3 environments were created each with 10 vehicles total, but with differing amounts of RL trained vehicles. The environments had 1, 2, and 10 RL controlled vehicles respectively. Within the DQN training, final ε value was set to 0.05, as with the ring track, and the decay rate of ε was set depending on the environment. The ε decay was set to occur over 40,000, 80,000, and 400,000 steps for each environment respectively. These values are twice those of the ring track as there are 1000 steps per episode versus 500 in the ring track environments. Within the PPO training, the default hyperparameters of the RLLib PPO implementation were used, except for the clip parameter of the value function. The clip parameter was set to 6000, 12000, and 60000 for each environment respectively to account for the large cumulative rewards returned by the environment. The discount factor γ was set to 0.99 in all environments.

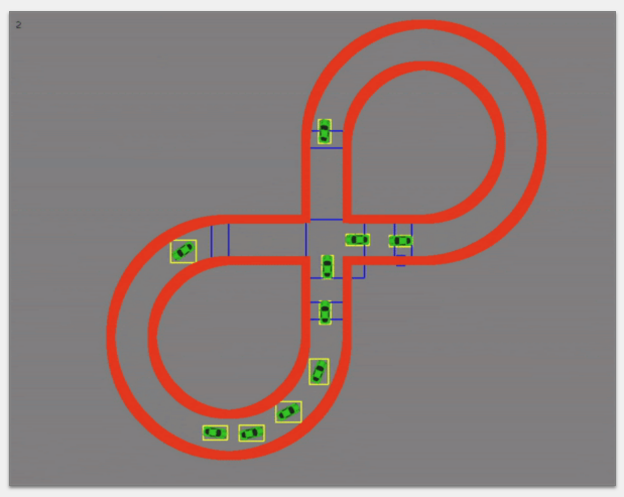

Figure 8: Implementation of the figure eight track

5. Training and Evaluation

The following sections display the results from the autonomous ring track, the fixed ring track, the fixed figure eight track, and the generalization of the model from the ring track to the figure eight track. Within the fixed tracks, the performance was measured by tracking the average velocity of all vehicles within the environments over the course of the training episodes. These same metrics are used for evaluating the performance of generalizing the DQN model of the ring track to the figure eight track.

5.1. Autonomous Ring Track

The results from training a single vehicle agent on the ring track environment, where the agent has control over both its acceleration and direction of travel, can be seen in Figure 9. From these results we can see that the vehicle agent was not able to learn to successfully traverse the entire track. There were 13 checkpoints that needed to be reached to successfully traverse the track and the agent was able to maximally reach the third checkpoint when trained using PPO and reached the fourth when trained using DQN. Within the PPO case, we see the policy converge after approximately 1000 episodes, reaching only the first checkpoint on average before the episode terminates. Within the DQN case, the results are more noisy as the agent learns to reach the second or third checkpoint after 10000 episodes.

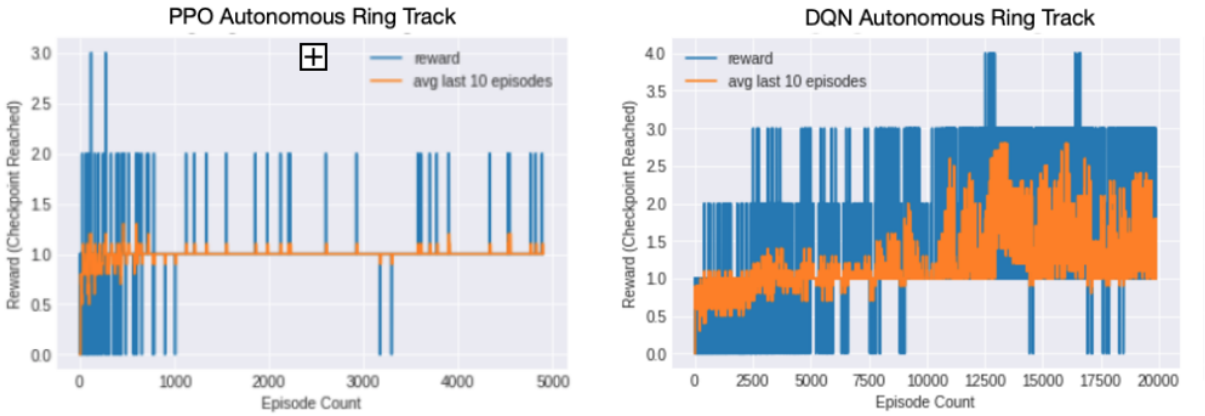

Figure 9: Results from the autonomous ring track displaying average velocity versus episode count

From these results, it is seen that the vehicle agents would not be able to successfully traverse the ring track, unless trained on a much higher magnitude of episodes. The DQN training displays potential is gradual improvement, but this would only occur after several magnitudes higher of training episodes. If more agents were to be added to this environment, the training process would require more computation, making traffic control infeasible with the available resources. Moreover, the agent likely was not able to fully traverse the environment due to very sparse sets of actions and states. This is a very large magnitude of possible states to explore due to the vehicle having control over both direction and acceleration, many of which would not be part of the optimal policy. Consequently, both PPO and DQN converge to a non-optimal policy from the set of states the vehicle agent was able to explore.

5.2. Ring Track Traffic Control

The results of the fixed ring track implementation, shown in Figure 10, display promising results. The baseline average velocity achieved using 0 RL trained vehicles is 6.47 and in all cases the RL trained vehicles were able to improve the average velocity above this baseline. In the single agent case (1 RL trained vehicle, 11 Non-RL), PPO and DQN perform similarly achieving an average velocity of approximately 7. In the 2 agent case (2 RL trained vehicles, 10 Non-RL), PPO and DQN improve further upon the baseline, achieving average velocities of approximately 8.

Lastly, in the 12 agent case (12 RL trained vehicles, 0 Non-RL), PPO performs significantly worse than DQN, as PPO converges to approximately the baseline while DQN trained agents completely eliminate the traffic and reach optimal performance. PPO likely converges to a less than optimum performance due to collisions between agents which guides the policy to the resulting performance. DQN, on the other hand, benefits from experience replay and may not be guided to a non-optimum policy from vehicle collisions, so long as it has experience of better performance to learn from in its replay buffer. Moreover, comparing these results to the FLOW project, seen in Figure 3, we see similar improvements from the baseline overall. The results from training 2 vehicles using PPO and DQN, however, show greater improvement per RL trained vehicle than the benchmarks of FLOW. The FLOW project sees a 4.07% improvement from the baseline using 3 RL trained agents. [12] Using 2 RL trained agents with PPO and DQN, there is a 23.07% improvement from the baseline.

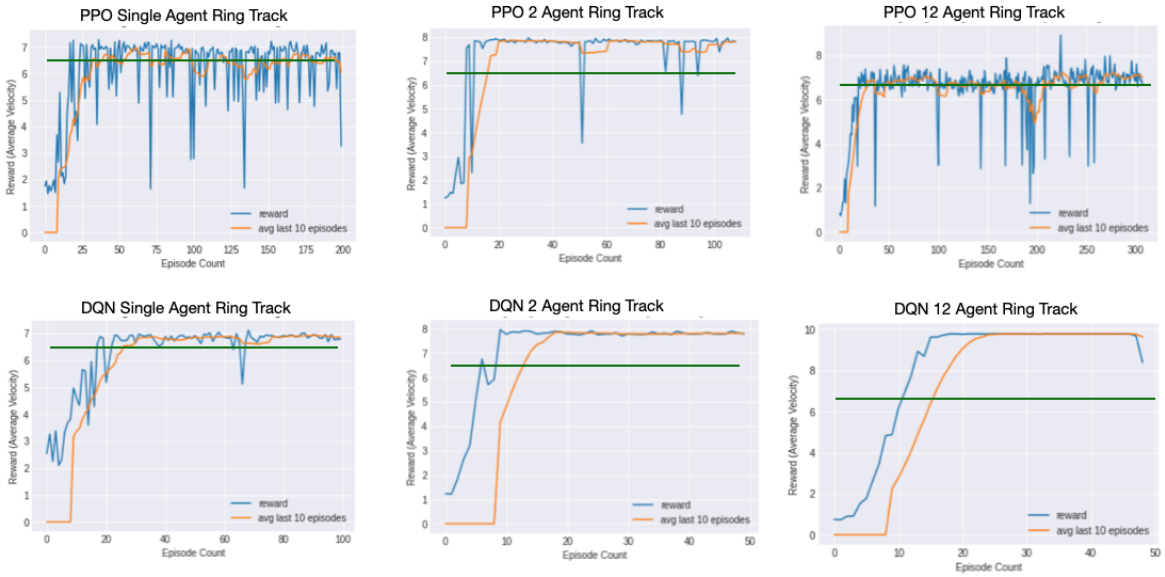

Figure 10: Matrix of results from the fixed ring track implementation. Green line in each plot represents the average velocity using 0 RL trained vehicles

Within Figure 11 we can better understand how the vehicles reach optimum performance when all 12 vehicles are trained using DQN. Within the first plot we see the average velocity decrease from the first step reaching a minimum by approximately step 80. This is caused by the vehicle slowing down as described in section 4.2 Following this, the agents accelerate together, but

do not do so uniformly. As the episodes progress, the agents learn a policy, or control law, that involves decelerating more rapidly and accelerating uniformly, as the valley in the plot decreases in width. This is similar to the control law learned by agents within the FLOW project [12], as the vehicles learn to decelerate and then accelerate together to stabilize traffic.

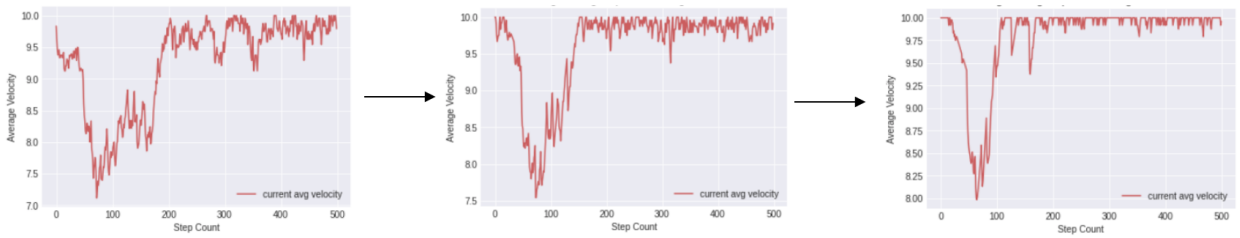

Figure 11: In-episode progression of average velocity of all vehicles. All 12 vehicles are trained using DQN. Arrow signifies moving forward 5 training episodes.

5.3. Figure Eight Traffic Control

The results of training on the implementation of the figure eight track, shown in Figure 12, display less promising results that in the ring track. Using both PPO and DQN, all results perform around the baseline of 3.49. DQN outperforms PPO in all cases, especially when 2 vehicles are trained using PPO. In this case, PPO likely converges to a non-optimum policy due to collisions between vehicles. As described in the previous section, DQN benefits from its experience replay so if the policy experiences a set of episodes with collisions, it may not be guided to a non-optimum policy due to using its experience from episodes with better performance. Further, when training all 10 vehicles with DQN, the vehicles perform better than the baseline, achieving an average velocity of approximately 4. The vehicles perform around the baseline likely due to the added difficulty of learning a policy that considers the intersection. The intersection adds additional control laws to the vehicles, creating additional complexity in the learned policy. Comparing these results to those of FLOW, seen in Figure 4, FLOW's benchmarks outperform the results of this environment using PPO and DQN, as FLOW achieved a 57.45% improvement from the baseline using a single RL trained agent (using TRPO) and achieved a 150.49% improvement from the baseline using all 14 RL trained vehicles. From the results displayed in Figure 12, we see no improvements from the

baseline except for the DQN 10 agent case where we see approximately an 11.7% improvement from the baseline.

Figure 12: Matrix of results from the fixed figure eight track implementation. Baseline denotes using 0 RL trained vehicles

5.4. Generalizing Ring Track to Figure Eight Track

In testing the generalization of the model of the ring track to the figure 8 track, only the DQN model was used. This is because the DQN model can be trained on the ring track environment until the reward from the policy for each episode converges and the ε parameter decays to 0.05. As the ε parameter is at 0.05 when applying the model to the figure eight track, the RL controlled vehicles follow the action chosen by the policy a proportion of 0.95 of the time. Since the policy has converged and agents are only acting randomly at a proportion of 0.05 of the time, the results of applying the DQN model to the figure eight track can be treated as evaluating the model, while not training it. The same is not the case when applying the PPO model of the ring track to the figure eight track. The PPO model outputs a stochastic policy, where the agent's choice of action is chosen according to the probability distribution of the stochastic policy output. Accordingly, the vehicle agents would choose actions that are not the highest probability action at a certain proportion. This

means that the policy would still learn after being applied to the figure eight track. Therefore, this cannot be treated as an evaluation of the PPO model from the ring track on the figure eight track.

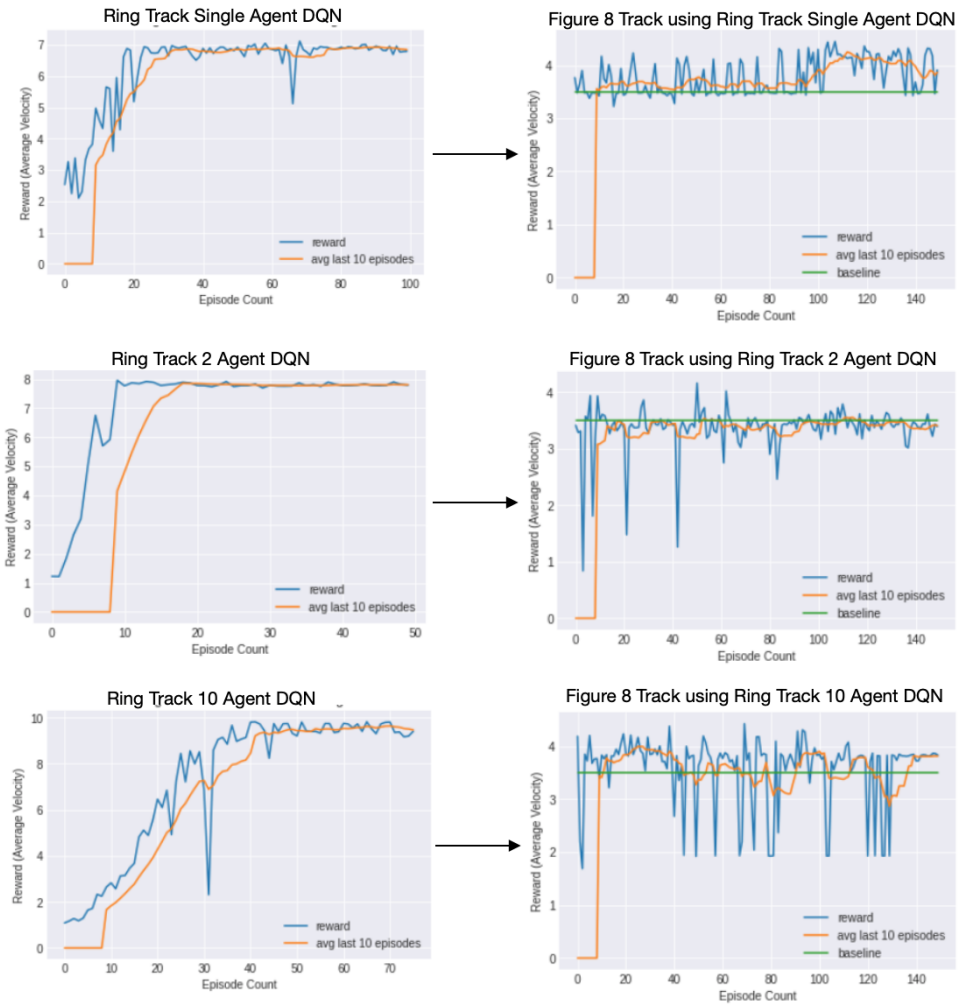

Figure 13: Results from applying DQN model of ring track environment to figure eight track environment.

From the results of applying the DQN model of the ring track environments to the figure eight track environments, displayed in Figure 13, the average velocity in most cases does not improve much, performing around the baseline where there are 0 RL trained vehicles. Within the single agent case, however, the DQN model from the ring track performs better than the baseline on the figure eight track. The average velocity reaches approximately 3.8 while the baseline is 3.49, an 8.89% improvement. Important to note is that this is better than the performance of the DQN model trained on the figure eight track in the single agent case. Within the 2 agent case (2 RL

controlled vehicles, 8 Non-RL), the DQN model of the ring track performs worse than the baseline on the figure eight track. Within the 10 agent case, (10 RL controlled vehicles, 0 Non-RL), the DQN model from the ring track applied to the figure eight track performs better than the baseline after approximately 130 episodes, performing similarly to the DQN model trained on the figure eight track using 10 agents. In this case, the average velocity reached approximately 3.9, an 11.7% improvement from the baseline. When using the DQN model of the ring track on the figure eight track, the average velocity reaches approximately 3.85, a 10.3% improvement from the baseline. The results when using the DQN model of the ring track on the figure eight track are more noisy than when using the DQN model trained on the figure eight track, however. This is likely due to the different traffic control laws from each environment, as the figure eight track contains an intersection, causing more collisions between vehicles. Overall, the model from the ring track generalized well when comparing to the performance of the DQN model trained on the figure eight track.

6. Discussion

6.1. Conclusion

From the evaluation of the ring and figure eight tracks differing results were produced. Within the ring track environment, similar improvements from baseline benchmarks as the FLOW project were seen. The improvement from the baseline using PPO and DQN in the environment built within this paper displayed a greater increase in the average velocity per additional vehicle trained using RL. Although the FLOW project did not display results using all 22 RL trained vehicles within the ring track implementation, the results presented here displayed that vehicles learned the optimal policy when all 12 vehicles were trained using DQN. Notably, the results from the ring track implementation displayed the relationship that using more vehicles trained with RL results in higher average velocities, reflecting the results of the FLOW project. [12] Within the figure eight track environment, there was not much improvement from the baseline. As noted in section 5.3, all experiments achieved minimal improvement from the baseline or a decline from the baseline, with

the exception of the case when all 10 vehicles were trained using DQN. The vehicle agents were not able to learn the optimal policy likely due to the limitations of queuing system at the intersection.

In generalizing the model of the ring track to the figure eight track, it was expected that the model would not generalize well due to the differences in the traffic control laws of each environment. Although the vehicles follow the same CFM as in the ring track, the intersection introduces complexity not seen by vehicles trained within the ring track environment. However, the DQN model of the ring track applied to the figure eight track generalized well, comparatively. The generalized model performed better than the DQN model trained on the figure eight track in the single agent case and achieved similar improvements within 10 agent case as well. This displays that even with different traffic control laws, the vehicle agents are able to apply their behavior from one traffic environment and see improvements from the baseline in another traffic environment, albeit not substantial improvements, so long as their observation spaces do not contain information specific to a certain environment.

6.2. Limitations

There are several limitations in the implementation of this project. The CFM implemented within both the ring track and figure eight tracks was a heuristic of IDM and is not equivalent to IDM. This means that it does not properly simulate human driving behavior. In addition, the vehicles agents' action space was a set of discrete actions that corresponded to fixed values for acceleration and deceleration. This means the vehicle agents did not have full control over their acceleration, as a continuous range of values to accelerate or decelerate would be needed. Moreover, the vehicle agents did not have autonomous control over their acceleration as they adhered to a rule that required them to decelerate when approaching vehicles moving slower than them to avoid collisions. Moreover, the observation spaces of the vehicles lack information about the positions of other vehicles relative to a staring position, as in the FLOW project. [10]. Lastly, due to computational limitations, the vehicle agents were trained over a small number of episodes, usually under 200, so a more complete exploration of the set of possible states was not achieved.

6.3. Future Work

The results from the generalization explored within the paper display improvements from the baseline when using two environments with different traffic control laws. This suggests that generalization using two tracks with the same traffic control laws but different structures should generalize better and should therefore be explored in future work. In addition, the FLOW project explores multiple lane environments [12] and future work should evaluate the ability of multi-lane traffic environments to generalize to each other. Lastly, as actor critic algorithms have seen impressive performance in multi-agent settings [6], future work should explore the performance of actor critic methods such as Advantage Actor Critic (A2C) on multi-agent traffic control environments.

7. Acknowledgements

I would like to thank my adviser, Prof. Karthik Narasimhan, for his guidance throughout this project, especially in helping me determine my project idea. I would also like to thank my fellow classmates in the seminar, *COS IW 03: Hands-On Reinforcement Learning*, for their continued feedback and advice throughout the semester.

All code for this project will soon be available in a public repository at: https://github.com/bmorck/MultiCar-RL

This paper represents my own work in accordance with University regulations

Bobby Morck

References

[1] I. Arel *et al.*, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 06 2010, copyright - Copyright The Institution of Engineering Technology Jun 2010; Document feature - Illustrations; Equations; Graphs; ; Last updated - 2014-12-11. Available: https://search-proquest-com.ezproxy.princeton.edu/docview/1635064307?accountid=13314

[2] B. Baker *et al.*, "Emergent tool use from multi-agent autocurricula," 2019.

[3] R. BELLMAN, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957. Available: http://www.jstor.org/stable/24900506

[4] B. Friedrich, *The Effect of Autonomous Vehicles on Traffic*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 317–334. Available: https://doi.org/10.1007/978-3-662-48847-8_16

[5] K. Kurach *et al.*, "Google research football: A novel reinforcement learning environment," *CoRR*, vol. abs/1907.11180, 2019. Available: http://arxiv.org/abs/1907.11180

[6] R. Lowe *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments," *CoRR*, vol. abs/1706.02275, 2017. Available: http://arxiv.org/abs/1706.02275

[7] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[8] E. Ritchie, "Self-driving automobiles: Two visions of the future." Available: https://www.forbes.com/sites/uhenergy/2019/04/17/self-driving-automobiles-two-visions-of-the-future/#6b0ad77226f6

[9] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] E. Vinitsky *et al.*, "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard *et al.*, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 399–409. Available: http://proceedings.mlr.press/v87/vinitsky18a.html

[11] Z. Wadud, D. MacKenzie, and P. Leiby, "Help or hindrance? the travel, energy and carbon impacts of highly automated vehicles," *Transportation Research Part A: Policy and Practice*, vol. 86, pp. 1 – 18, 2016. Available: http://www.sciencedirect.com/science/article/pii/S0965856415002694

[12] C. Wu *et al.*, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *CoRR*, vol. abs/1710.05465, 2017. Available: http://arxiv.org/abs/1710.05465

[13] K. J. Åström, "Optimal control of markov processes with incomplete state information i," vol. 10, pp. 174–205, 1965. Available: https://lup.lub.lu.se/search/ws/files/5323668/8867085.pdf