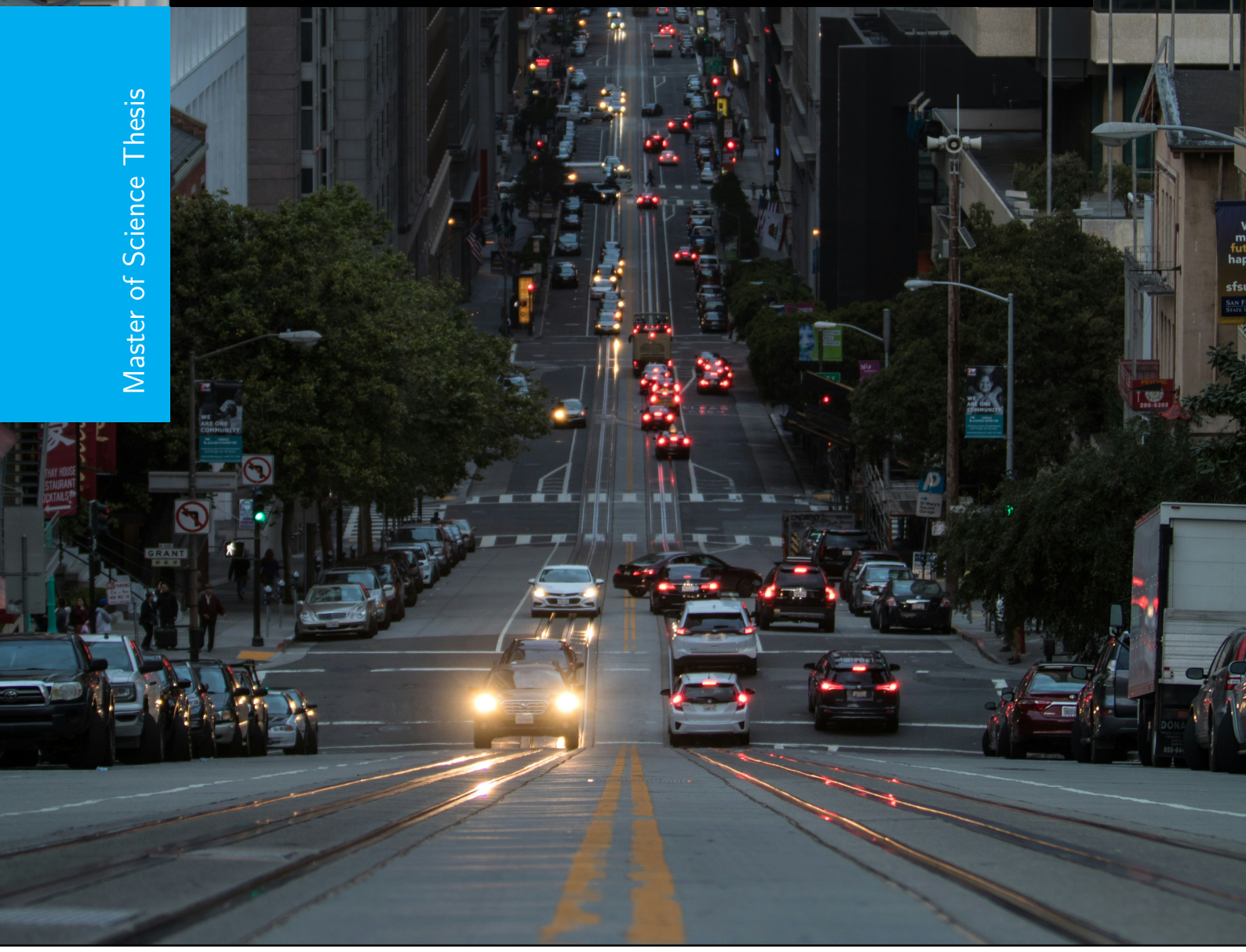
Advancing Deep Reinforcement Learning for Real-World Traffic Signal Control

Addressing Sampling Challenges and Multi-Modal Traffic Dynamics

K. F. Ceton

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Advancing Deep Reinforcement Learning for Real-World Traffic Signal Control

Addressing Sampling Challenges and Multi-Modal Traffic Dynamics

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

K. F. Ceton

November 25, 2024

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

This work was supported by Technolution B.V.

Abstract

Deep Reinforcement Learning (DRL) is a promising approach to Traffic Signal Control (TSC). However, significant challenges remain in translating this potential into real-world traffic management solutions. This thesis investigates obstacles hindering the application of DRL in real-world TSC, focusing on low sampling frequencies and the complexities of multi-modal traffic scenarios.

We developed a high-frequency sampling Proximal Policy Optimization (PPO) approach for TSC at a four-legged intersection, integrating both vehicle and pedestrian traffic in a multi-modal setting. By employing Invalid Action Masking (IAM), we effectively handle signal timing constraints across these modalities. The framework was evaluated through traffic volume sensitivity analyses, assessments of generalization capabilities, disturbance rejection tests, and comparisons of methods for handling invalid actions.

The results indicate that short sampling intervals, such as 1 second, do not improve performance in terms of time-loss, with 4 to 6 seconds identified as the optimal range for PPO in TSC of a four-legged intersection. The findings also demonstrate that IAM can effectively be incorporated without compromising performance. When evaluating the ability to handle sudden spikes in traffic volume, PPO demonstrated superior performance, outperforming baseline methods such as max-pressure and fixed-time strategies in terms of both overshoot and settling time. Also, the results show that PPO can effectively prioritize vehicle and pedestrian modalities, displaying a clear proportional decrease in time-loss for the prioritized modality.

Table of Contents

Acknowledgments

Coming from three-hour exams and quarterly projects, writing a year-long thesis really is something else. Making this thesis has been insightful, valuable, and for most of the time, fun. Luckily, I had help along the way and I would like to thank some people for that.

I would like to thank my first TU Delft supervisor Sergio Grammatico for the meetings and feedback, proving critical questions and useful perspectives.

Thanks to my second TU Delft supervisor, Giorgos Pantazis, for the insightful discussions, fresh ideas, and shared laughs during our monthly meetings.

I would like to thank my Technolution supervisor Tijs van Bakel the weekly meetings we had throughout the year. His positive can-do attitude, critical questions, curiosity and willingness to listen have have helped a lot and made my days at Technolution all the better.

Surely, I want to thank my friends and family who have had to listen to my moments of enlightenment and subsequent downfall. With a special thanks to my sister Lieke for the weekly dinners and games of squash.

Lastly, a big thanks to all those TU Delft employees who kept those vital coffee machines running.

Delft, University of Technology K. F. Ceton
November 25, 2024

Chapter 1

Introduction

Traffic congestion has been an societal problem for decades. There are various staggering statistics that *reinforce* these words; In big cities, such as London, the average commuter spends more than 150 hours stuck in traffic each year [16]. Or, traffic jams cost U.S. drivers more than 81 billion us dollars in 2022 [9]. Beside these economic and time related impacts, there are also health related consequences. Studies find that traffic-induced stress increases the risk of depression and anxiety [7], and in urban areas the extra pollution of traffic jams is harmful for the physical health [19]. In short, convenient and effective transportation benefits all.

Traffic Signal Control (TSC) has significantly evolved over the last century. In 1958, Webster introduced one of the first methods to minimize vehicle delay at intersections [41]. These early controllers optimized single intersections without coordinating with adjacent ones, causing network congestion elsewhere. To tackle this, fixed-time control strategies were developed to manage multiple intersections simultaneously, calculating control plans offline based on historical traffic data. MAXBAND [21] is one such method aiming to maximize arterial road bandwidth but becomes impractical with increased side-road traffic [28]. A key drawback of fixed-time controllers is their inability to adapt to real-time traffic fluctuations, such as accidents. This lead to what most traffic signal controllers use today, which are traffic-responsive controllers such as SCOOTS and SCATS [15].

Traffic adaptive control methods adapt to the traffic conditions in real-time. There is some input measure which indicates that state of traffic (e.g. loop detectors), a solving algorithm and a decision which is send to the traffic signalers. One of these algorithms is the max-pressure approach, where the pressure is defined as a measure of traffic density and the phase configuration with the highest pressure release is decided [37]. A more intricate approach, named Model Predictive Control (MPC), uses a predictive model to estimate the traffic evolution over time, optimizing an objective function to determine the optimal traffic light configuration for the next time step [46]. As urban traffic is a highly non-linear phenomenon, MPC can be challenging; Using a non-linear prediction model and optimization would require high computational demand whilst using a linear prediction model would sacrifice accuracy. [20] Is there a better approach?

In 2013-2015, Deep Reinforcement Learning (DRL) emerged as a breakthrough approach; Deep Neural Networks could effectively be used to serve as a decision maker in the Reinforcement Learning Framework [23]. Reinforcement Learning (RL) is a type of machine learning where a decision maker learns to make decisions by interacting with an environment and receiving feedback in the form of rewards to maximize cumulative rewards over time [33]. In contrast to other adaptive approaches, Deep Reinforcement Learning does not rely on a predictive model and can still effectively capture highly nonlinear relationships which makes it an notable competitor to handle the Traffic Signal Control problem [34]. In recent years, the amount of sensor, GPS and computer vision data has also increased significantly. DRL can handle this data since it can process various different data types. In short, Deep Reinforcement Learning has been researched to manage the Traffic Signal Control problem and shows great promise [42]. So, why is it not on the streets yet?

The current state-of-the art Deep Reinforcement Learning in Traffic Signal Control is mainly focused on performance in simplified simulations and is not ready to be implemented on the actual streets [42]. This gap needs to be bridged in order to start saving real time in real peoples lives. What are the main obstacles in between the current state-of-the-art and a real-life implementation of Deep Reinforcement Learning in Traffic Signal Control, and how can these be addressed? That is what this thesis report will focus on.

Two primary challenges were identified. On of which is the sampling frequency; The time in between observations of the Deep Reinforcement Learning agent is around 5-20 seconds in most research, which is low considering the speeds of vehicles [42]. Increasing this sampling frequency to 1 second could allow the agent to make faster decisions more frequency leading to better performance. However, this introduces complexity due to safety requirements for minimal light time, the minimum duration traffic lights must stay in a given state (e.g. 3-6 seconds for the yellow state) [1].

To counteract these complexities, we developed a novelty high-frequency framework which employs Invalid Action Masking (IAM) to counter the minimal light constraints. This framework is thoroughly detailed and evaluated through various tests, including a sensitivity analysis on traffic volumes, an assessment of its generalization capabilities, an assessment of its disturbance rejection capabilities, and a comparison of Invalid Action Masking with alternative methods, such as penalization, for minimizing invalid actions.

Secondly, in real-life traffic scenarios, traffic intersections have to deal with multiple traffic modes simultaneously. These modes are allowed priority in certain scenarios, such as school busses during rush hour and pedestrians when it is raining. To address this prioritization and extra complexity of multiple traffic modes, this thesis researches the inclusion of pedestrians and vehicles in a combined traffic scenario. In this research, we also apply the Invalid Action Masking technique to the multi-modal traffic scenario.

These improvements aim to bring Deep Reinforcement Learning for Traffic Signal Control closer to real-world application, moving the field a step closer to real-life impact and enabling technology to start saving real time.

1-1 Problem statement

In the context presented, the main research question of this thesis is:

What are challenges in applying Deep Reinforcement Learning to real-world traffic signal control, and how can these be addressed?

To effectively answer this question, two sub-questions were derived:

How does high-frequency sampling improve the performance of Deep Reinforcement Learning approaches in Traffic Signal Control, specifically in terms of reducing time-loss and rejecting input disturbances?

The first sub-question focuses on the impact of high-frequency sampling on the performance of DRL approaches in TSC. By exploring higher-frequency sampling intervals, this research aims to determine whether more frequent observations and actions can enable the DRL agent to make quicker and more effective decisions, thus reducing time-loss and better handling unexpected traffic disturbances. This research also considers the effectiveness of the methods to counteract minimal signal times and heavily varying traffic scenarios.

Can a Deep Reinforcement Learning Agent effectively balance prioritization between pedestrians and vehicles in terms of time-loss?

The second sub-question investigates whether a DRL agent can effectively balance the prioritization between pedestrians and vehicles at intersections. In real-world traffic scenarios, multiple modes of transportation must be managed simultaneously, each with different priorities. The research aims to assess the DRL agent's capability to minimize overall time-loss whilst effectively prioritizing multiple traffic modes.

1-2 Thesis outline

This thesis is structured as follows. In chapter 2, we introduce the foundations of DRL, covering the necessary preliminaries and providing an in-depth explanation of the DRL algorithm employed in this study, namely PPO. Thereafter, chapter 3 focuses on the TSC problem, detailing its terminology, classical approaches, simulation models, and the integration of DRL in TSC, along with the associated challenges. This chapter also outlines the approaches considered for managing high-frequency sampling and pedestrian inclusion.

In chapter 4, we present the results of the case studies and derive conclusions from the findings. Finally, chapter 5 provides a comprehensive conclusion to the research, offering a discussion to contextualize the results and recommendations for future work in this domain.

Chapter 2

Deep Reinforcement Learning

In this chapter, we present the foundational concepts of reinforcement learning and deep reinforcement learning such as the Markov Decision Process (MDP) framework without touching on value- and action-value functions. Thereafter, we go one step further to Deep Reinforcement Learning; Reinforcement Learning which leverages Deep Neural Networks as decision maker. We introduce various algorithms and go in depth on the Proximal Policy Optimization (PPO) algorithm. We explain the mechanics of the Proximal Policy Optimization algorithm and discuss concepts such as the Generalized Advantage Estimation and the clipped surrogate loss function. In the subsequent chapters, we will continue to apply Proximal Policy Optimization to our case, which is Traffic Signal Control.

2-1 Reinforcement Learning

In the following section, the concept of Reinforcement Learning will be presented. Firstly, the mathematical framework which is at the foundation of RL is presented, namely Markov Decision Process. From here, we will expand on methods to solve the MDP, such as Deep Reinforcement Learning approaches. We will shortly present the types of Deep Reinforcement Learning approaches, after which we will go into detail on the utilized method, namely PPO.

2-1-1 Markov Decision Processes (MDP)

Markov Chain

In 1906, the Russian mathematician A. A. Markov published a paper that laid the foundation for the concept of Markov Chains [11]. The Markov Chain is a mathematical system that undergoes transitions from one state to another according to certain probabilities [33]. A Markov Chain or Markov Process consists of two main elements: The *states* $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states. These are all the possible states the process can be in. The second element is the *The Transition Function* $\mathcal{T}$, which defines with what probabilities we

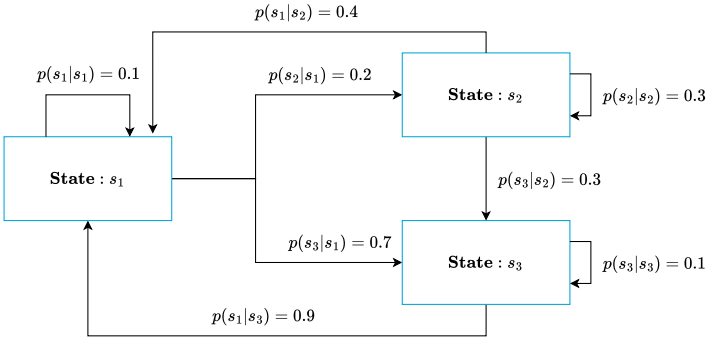


Figure 2-1: Schematic diagram of an example of a Markov Chain with three states $\{s_1, s_2, s_3\} \in \mathcal{S}$.

move through the states. This is defined as $p_{ij} = p(X_{t+1} = s_j \mid X_t = s_i)$ where p_{ij} represents the transition probability from state s_i to state s_j. In Figure 2-1, an example with three states $\{s_1, s_2, s_3\} \in \mathcal{S}$ can be observed. In Table 2-1, an overview of the components can be observed.

Component	Description	Notation	Mapping
The States	The possible states that the process can be in.	$s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states.	
The Transition Function $\mathcal{T}$	The probabilities of transitioning from one state to another.	$p_{ij} = p(X_{t+1} = s_j \mid X_t = s_i)$	$\mathcal{T} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$

Table 2-1: Components of the Markov Chain.

An Markov chain satisfies the *Markov Property* if the transition function depends only on the current state s as is depicted in Equation 2-1. Adhering to the Markov property reduces the complexity of the dynamics of the system, since the next state only depends on the present state [6].

$$p(X_{t+1} = s_{t+1} \mid X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = p(X_{t+1} = s_{t+1} \mid X_t = s_t) \tag{2-1}$$

Markov Decision Processes

A Markov Decision Process is an extension of the Markov Chain that incorporates decision making. In an MDP, a decision-maker interacts with the environment by taking actions that influence state transitions and receive rewards. The decision maker is named the *agent*. An MDP consists of the following main elements:

In Figure 2-2, a Markov Decision Process can be observed with three states and two actions for each state. A Markov Decision Process is defined by two functions, the transition function

Component	Description	Notation	Mapping
The States $\mathcal{S}$	The possible states that the process can be in.	$s \in \mathcal{S}$	
The Actions $\mathcal{A}$	The set of all possible actions the agent can take.	$a \in \mathcal{A}$	
The Transition Function $\mathcal{T}$	The probabilities of transitioning from one state to another given an action.	$p(s_{t+1} \mid s_t, a_t)$	$\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$
The Reward Function $\mathcal{R}$	The immediate reward received after transitioning from one state to another due to an action.	$r(s_t, a_t, s_{t+1})$	$\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$
The Policy π	The strategy that the agent employs to choose actions based on the current state.	$\pi(a \mid s)$	$\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$

Table 2-2: Components of the Markov Decision Process.

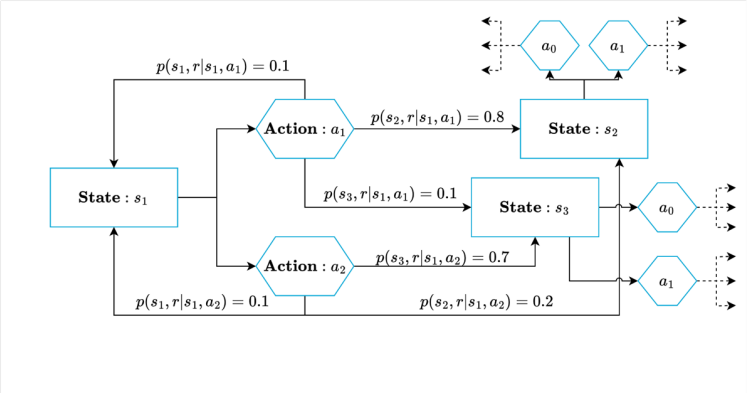


Figure 2-2: Schematic diagram of an example of a Markov Decision Process with three states $\{s_1, s_2, s_3\} \in \mathcal{S}$ and two actions $\{a_1, a_2\} \in \mathcal{A}$. To maintain overview, not all probability connections are drawn.

and the reward function. The transition function $\mathcal{T}$ defines in what state we end up after performing a certain action in a certain state. This defines the dynamics of the environment. The reward function $\mathcal{R}$ defines the reward given for certain transitions. This could influence the actions taken by the agent. When combining the two probability functions into one $p(\cdot)$, we can define a Markov Decision Process with the following equation [33]:

$$p\left(s', r \mid s, a\right) \doteq \Pr\left\{s_t = s', r_t = r \mid s_{t-1} = s, a_{t-1} = a\right\} \tag{2-2}$$

for all $s', s \in \mathcal{S}, r \in \mathcal{R}$, and $a \in \mathcal{A}(s)$. Pr defines a probability function that specifies the distribution over next states and rewards, given a current state and action. The notation $s_t = s$ is used to emphasize that s_t is a random variable representing the state at time t, while s denotes a specific realization (value) of that random variable.

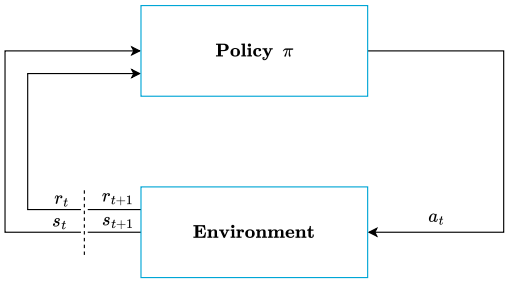


Figure 2-3: Schematic diagram of the Reinforcement Learning Framework

Reinforcement Learning Framework

In Figure 2-3, the Reinforcement Learning Framework is presented. The agent interacts with the environment in discrete time steps. At each time step t, the agent observes the current state s_t, takes an action a_t, receives a reward r_t, and transitions to the next state s_{t+1}. The goal of Reinforcement Learning to maximize the cumulative reward, or return, which is typically defined as the sum of discounted rewards over time, named the *Return*:

$$R_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2-3}$$

where γ is the discount factor, $0 \leq \gamma \leq 1$, that determines the importance of future rewards.

As an optimization problem, the objective is to find an optimal policy π, denoted as π^*, which maximizes the expected return:

$$\pi^* := \arg\max_{\pi} \mathbb{E}_{\pi} \left[R_t \mid s_t = s \right]$$

The expected value, denoted as $\mathbb{E}[\cdot]$, can be viewed as the long-run average of a random variable. We maximize the expected return instead of the true return due to the stochasticity in the transition function $\mathcal{T}$. The optimization problem is completely defined as:

$$\pi^* := \arg\max_{\pi} \left(\mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right] \right) \quad \text{for all } s \in \mathcal{S} \tag{2-4}$$

A policy is deemed optimal when it adheres to the Bellman Optimality [33]. This optimization ensures that the agent learns to make decisions that maximizes long-term rewards, balancing immediate and future rewards. This is the problem to solve in Reinforcement Learning, in the next sections, we will explore possible approaches to find the optimal policies.

Solving the MDP

Solving a Markov Decision Process (MDP) involves finding an optimal policy that maximizes the expected cumulative reward. Apart from Deep Reinforcement Learning, there are various other methods to do this which are shortly mentioned: Dynamic Programming (DP) methods,

such as value iteration and policy iteration, provide a systematic approach to solving MDPs by breaking them down into simpler subproblems. These methods rely on the Bellman equations to iteratively compute the value function and improve the policy. Monte-Carlo (MC) methods estimate the value of states and actions by averaging the returns from multiple sample episodes. Unlike DP, MC methods do not require knowledge of the environment's dynamics and can be applied to problems with unknown transition probabilities. Temporal Difference (TD) learning combines aspects of both DP and MC methods. TD learning updates the value estimates based on a single sample of the next state and reward, making it suitable for online learning and situations where a model of the environment is not available. Notable TD methods include Q-learning and SARSA. [33]. The next section will present Deep Reinforcement Learning.

2-2 Deep Reinforcement Learning

In the following section, *Deep Reinforcement Learning (DRL)* is presented. Deep Reinforcement Learning leverages a neural network (NN) as agent to do the decision making. As a result, DRL is capable of employing complex policies in complex environments. Below, a general impression of the field of algorithms in DRL is presented. Thereafter, we provide an detailed explanation of the mechanics of the leveraged algorithm, namely Proximal Policy Optimization.

2-2-1 Types of Deep Reinforcement Learning Algorithms

One possible distinction in the wide field of Deep Reinforcement Learning algorithms is the value- or policy-based property. Value-based methods provide estimations of the Q-values, which will be introduced in subsection 2-2-2. These algorithms include Deep Q-networks, also known as DQN [23]. The principle of Deep Q-Networks got improved by techniques such as Double, Duelling DQN [36, 40]. This lead eventually to the Rainbow DQN, which is the most advanced DQN model to date [12].

The REINFORCE family of algorithms, introduced by Williams in 1992, was the first policy optimization approach in reinforcement learning [44]. However, REINFORCE tends to have high variance in its gradient estimates, which can result in slow and unstable learning [33]. To address these issues, Schulman introduced Trust Region Policy Optimization (TRPO) in 2017, another policy gradient method that focuses on improving sample efficiency and maintaining stability by constraining policy updates [29]. TRPO introduces a trust region constraint that limits the maximum change in the policy during each update, ensuring that updates are conservative and do not lead to large, destabilizing policy changes. Building on TRPO, OpenAI developed Proximal Policy Optimization (PPO) in the same year. In this thesis work, PPO was used as the primary algorithm for the experiments conducted.

2-2-2 Proximal Policy Optimization

In the following section, Proximal Policy Optimization will be presented and discussed. Firstly, the full architecture of the PPO algorithm is presented to give an overview of the

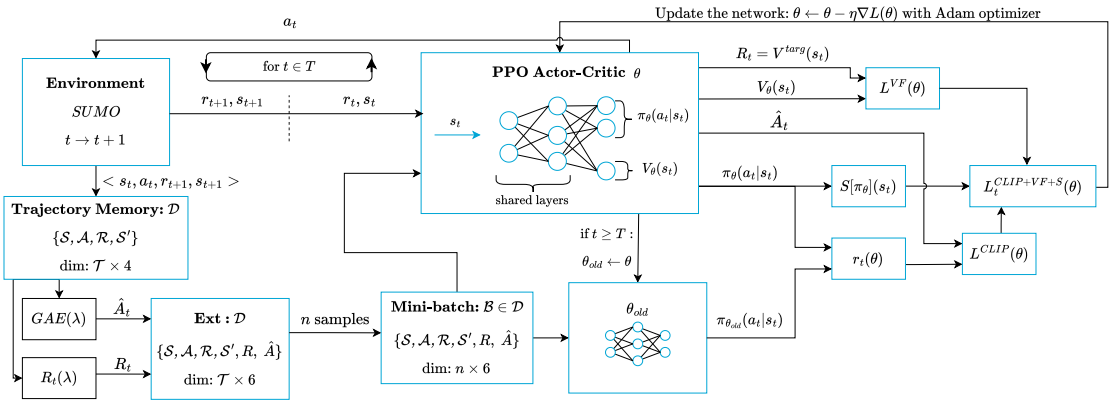


Figure 2-4: Architecture of the Proximal Policy Optimization Algorithm. Arrow indicate a general data flow.

algorithm. Thereafter, we zoom into individual parts and concepts such as value-functions, action-value functions, neural networks, n-step returns, general advantage estimation and the clipped surrogate loss function.

Proximal Policy Optimization Architecture

In this section, we will provide an overview of the Proximal Policy Optimization algorithm. In Figure 2-4, the complete architecture of the PPO algorithm can be seen. The figure displays the flow of data and the individual steps taken in the training process. Individual blocks indicate a certain operation on with the data inserted, indicated with an arrow. A full update cycle in PPO generally consists of two parts; experience collection and a neural network update.

The first step in a batch is to collect experiences for the trajectory memory $\mathcal{D}$. The agent interacts with the environment using the current policy π to collect rewards for n_{steps} simulation steps. The collection of experiences are also known as a *batch* where n_{steps} is the *batch size*. Note: the size of a batch can also be referred to as T. Once these experiences are collected, two values are calculated for each experience tuple: The return R_t and the estimated advantage $\hat{A}_t$. The function of these values will be explained in later sections.

In the second phase, the agent is trained with the collected experiences. Firstly, n experience samples are taken randomly from the trajectory memory, named a *mini-batch* $\mathcal{B}$. The amount of samples in the mini-batch can be referred to as $n_{mini-batch}$. For each sample in the mini-batch loss is calculated through a series of operations, as can be seen in Figure 2-4 and will be explained in further sections. This loss over this mini-batch is then averaged and back-propagated over the weights and biases θ in the Actor-Critic Network. One of these cycles is called an *epoch* and gets repeated n_{epoch} times.

Finally, the updated policy will be used for the collection of the next batch of experiences while the policy before updating is saved as θ_{old}. Since the policy that is learned is also used to collect new experiences in the next update cycle, PPO is an *on-policy* algorithm. The total number of time-steps over the complete training process is referred to as T_{total}

Value- and Action-Value function

Reinforcement learning algorithms leverage *value functions* and *action-value functions*. The value function represents the expected cumulative reward that an agent can obtain, starting from state s_t and following policy π thereafter. Formally, it is defined as the expected sum of discounted future rewards, taking into account the probability distribution of future states and actions dictated by the policy π [34]. More informally, it is a measure of how "good" a certain state is under a certain policy. Mathematically, it is defined as:

$$V^{\pi}(s) := \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right]$$

The Q-value function, or action-value function, represents the expected cumulative reward that an agent can obtain, starting from state s_t, taking action a_t, and subsequently following policy π. It provides a measure of the long-term benefit of taking a particular action in a particular state under a given policy, guiding the agent to choose actions that maximize cumulative rewards. Mathematically, it is defined as:

$$Q^{\pi}(s,a) := \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right]$$

The value- and action-value function can be used to calculate the *advantage function*. The advantage function, $A(s,a)$, measures the relative value of a state-action pair compared to the value of the state alone. It provides a way to assess how much better or worse it is to take a specific action a_t in state s_t, compared to the average value of being in state s_t and following policy π [34]. It is defined as:

$$A^{\pi}(s,a) := Q^{\pi}(s,a) - V^{\pi}(s)$$

Both the value- and action-value function are functions of π and expected values since the process is assumed stochastic and the transition functions are unknown. To estimate these functions, PPO leverages a neural network as function estimator. In the next paragraphs, it is explained how to estimate these functions and how to balance bias and variance in the estimation.

N-step returns

N-step returns is an important concept in value- and action-value function estimation. As mentioned, the value-function is approximated by a neural network. A neural network is a graph of weights and biases θ which can estimate highly non-linear behavior. More in depth mechanics will be discussed in section 2-2-2. For now, $V_{\theta}(s_t)$ is defined as the value-function estimator produced by the NN.

When estimating the cumulative discounted reward R_t, we can use the Monte-Carlo return, which is defined as:

$$R_t^{(\infty)} := \sum_{l=0}^{T-t} \gamma^l r_{t+l}$$

$R_t^{(\infty)}$ provides an unbiased sample of the expected return at a given state, but due to stochasticity from the dynamics of the environment and policy, each reward r_t can be a random variable, the sum of which can result in a high variance estimator of the expected return [26]. On the other hand, there is the 1-step return which is defined as:

$$R_t^{(1)} := r_t + \gamma V_\theta(s_{t+1})$$

In the 1-step return, the expected return is the sum of the reward after the first time-step and the value-function estimator at the next state. The value-function estimator is a value assigned to a state which has been updated over many samples. Therefore, it has lower variance but does introduce bias. Logically, the 1-step return has relatively high bias and low variance. To balance the bias and variance trade-off, n-step returns was introduced and is defined as follows:

$$R_t^{(n)} := \sum_{l=0}^{n-1} \gamma^l r_{t+l} + \gamma^n V(s_{t+n}) \tag{2-5}$$

Here, we can balance the variance-bias trade-off by setting n. This concept is leveraged in Generalized Advantage Estimation and is presented in the following paragraph.

λ-returns

Another method to trade off between the bias and variance of the estimator is to use a λ-return [33], calculated as an exponentially-weighted average of n-step returns with decay parameter λ:

$$R_t(\lambda) := (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$$

Note that this is a summation over the $R_t^{(n)}$ n-step returns as defined in Equation 2-5. Assuming all rewards after step T are 0, such that $R_t^{(n)} = R_t^{T-t}$ for all $n \geq T-t$, the infinite sum can be calculated according to:

$$R_t(\lambda) := (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t^{T-t} \tag{2-6}$$

$\lambda = 0$ reduces to the single-step return $R_t^{(1)}$, and $\lambda = 1$ recovers the Monte-Carlo return $R_t^{(\infty)}$. Intermediate values of $\lambda \in (0,1)$ produces interpolants that can be used to balance the bias and variance of the value estimator. In short, the $R_t(\lambda)$ is the expected cumulative reward in a certain state s which balances bias and variance by balancing the weighing in of the existing value function estimator and the sample.

Generalized Advantage Estimator $GAE(\lambda)$

In section 2-2-2, the advantage function was defined as the action-value function minus the value function. As a true-value function approximator, $R_t(\lambda)$ is used. The Generalized Advantage Estimation (GAE) is used to estimate the advantage function $\hat{A}$.

$$\text{GAE}(\gamma, \lambda) : \hat{A}_t := R_t(\lambda) - V_\theta(s_t)$$

$GAE(\gamma, \lambda)$ is a function of γ, which indicated how heavily we weight future rewards as seen in Equation 2-5. It is also a function of λ, which controls how heavily we weight either the sample or the estimator. In Figure 2-5, a clarifying overview is shown.

$$\text{GAE}(\gamma, 0) : \quad \hat{A}_t = \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$\text{GAE}(\gamma, 0)$ results in the TD-error, defined as δ_t.

$$\text{GAE}(\gamma, 1) : \quad \hat{A}_t = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t)$$

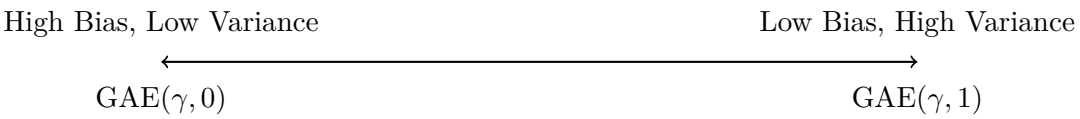


Figure 2-5: Bias-Variance Spectrum with $\text{GAE}(\gamma, 0)$ representing high bias and low variance, and $\text{GAE}(\gamma, 1)$ representing low bias and high variance.

Clipped surrogate loss function

The loss-function is a function which magnifies errors in the neural network estimation. It is a *surrogate* loss function since it is an estimation of the true policy gradient objective. In PPO, it is defined as:

$$L_t^{\text{CLIP+VF+S}}(\theta) = \mathbb{E}_t \left[L_t^{\text{CLIP}}(\theta) - c_{vf} L_t^{\text{VF}}(\theta) + c_{ent} S[\pi_\theta](s_t) \right] \tag{2-7}$$

It consists of three separate elements, the clipped actor loss, the weighted value function loss, and the weighted entropy bonus [30]. Firstly, the clipped loss function is defined as:

$$L_t^{\text{CLIP}}(\theta) = \min \left(r_t(\theta)\hat{A}_t,\ \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \tag{2-8}$$

It is a minimal function of two elements. $r_t(\theta)$ is the ratio between the old policy and the new policy for a given state s_t and action a_t. It is defined as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t}{\pi_{\theta_{old}}(a_t|s_t)}$. If $r_t(\theta) > 1$, action a_t at state s_t is more likely in current policy. If $r_t(\theta) \in [0, 1]$, action a_t at state s_t is more likely in old policy. The first element, $r_t(\theta)\hat{A}_t$, therefore represents a measure for the relevance and positive impact of an action. The second element is a clipped version of this

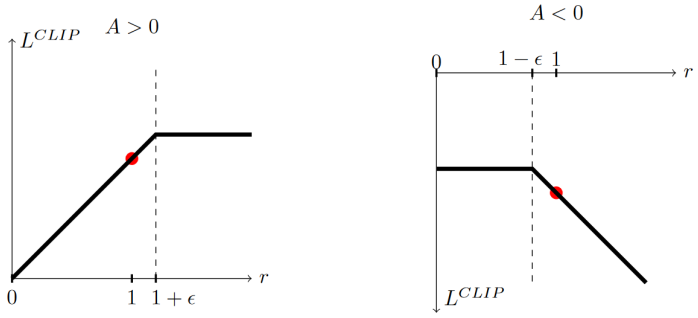


Figure 2-6: Loss behavior based on ratio and advantage. [30]

term. The clipping constrains the magnitude of the L_{CLIP} for actions that are more likely in the new policy and have a positive impact. On the other hand, negative actions that are more likely in the old policy will receive a constant penalty. Also, actions that are more likely in the new policy and have a negative impact are not constraint on the magnitude of the penalty. In Figure 2-6 and Table 2-3, provide intuition and overview of the concept.

	$A > 0$	$A < 0$
$r_t(\theta) \in [0, 1]$	Gradual increase of loss	Negative upper bound on loss
$r_t(\theta) > 1$	Positive upper bound on loss	Gradual decrease of loss

Table 2-3: Loss behavior based on ratio and advantage.

The value-function loss term is defined as:

$$L_t^{\mathrm{VF}} = \left(V_\theta(s_t) - V_t^{\mathrm{targ}}\right)^2 \tag{2-9}$$

Where V^{target} is an estimation of the sample with $R_t(\lambda)$ and $V_\lambda(s_t)$ is the value-function estimator at s_t. This represents the value-function estimation error and grows in magnitude as the accuracy of the estimation is poor. This term is used as expected value over all the n samples in a mini-batch. Therefore, the term in Equation 2-7 is equal to the following:

$$\mathbb{E}_t[L_t^{\mathrm{VF}}] = \frac{1}{n}\sum_{i=0}^{n}\left(V_\theta(s_i) - V_i^{\mathrm{targ}}\right)^2 \tag{2-10}$$

The final term, the entropy bonus, is defined as:

$$S[\pi_\theta](s_t) = -\sum_{a\in\mathcal{A}} \pi_\theta(a|s_t)\log(\pi_\theta(a|s_t)) \tag{2-11}$$

In reinforcement learning, the entropy represents the unpredictability of the actions. The higher the entropy, the less certain the taken action is. Ideally, the entropy should decrease over time during training. The complete loss function is used to update the Actor-Critic neural network through stochastic gradient descent which is elaborated on the following paragraph.

Neural Networks

neural network consist of nodes interconnected layers of nodes which are able to map highly non-linear function between in- and output layers. Each connection between nodes has an associated weight θ_w, and each node has a bias θ_b, both are referred to as θ [8]. The linear operation for a single node can be expressed as:

$$z = \theta_w \cdot x + \theta_b,$$

where x is the input to the node. This linear combination is then passed through an activation function $\phi(z)$, which introduces non-linearity into the model. One common activation function is the Rectified Linear Unit (ReLU), defined as:

$$\text{ReLU}(z) = \max(0, z).$$

In Proximal Policy Optimization, the actor-critic framework is used where shared layers provide a common representation, and separate output layers generate the policy (actor) and value function (critic). Stochastic Gradient Descent (SGD) is an optimization algorithm used to minimize the loss function by iteratively updating the model parameters in the direction of the negative gradient [8]. The update rule is given by:

$$\theta := \theta - \alpha \nabla_\theta L(\theta),$$

where θ represents the weights and biases in the network, α is the learning rate, and $L(\theta)$ is the loss function. $\nabla_\theta L(\theta)$ is the partial derivative of the loss with respect to all parameters θ defined as $\nabla_\theta L(\theta) = \left(\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \ldots, \frac{\partial L}{\partial \theta_{max}}\right)$. The Adam optimizer is an extension of SGD that adapts the learning rate for each parameter, combining the advantages of both AdaGrad and RMSProp [17].

2-3 Conclusions

In conclusion, this chapter established the foundational principles of reinforcement learning and deep reinforcement learning by introducing the Markov Decision Process (MDP) framework and techniques to solve it.

We then transitioned to Deep Reinforcement Learning, focusing on the Proximal Policy Optimization algorithm. An in-depth examination of PPO covered its underlying mechanics, such as Generalized Advantage Estimation and the clipped surrogate loss function.

In the upcoming chapters, we will apply the PPO algorithm to our specific case study of Traffic Signal Control, building upon the concepts discussed here.

Chapter 3

Deep Reinforcement Learning in Traffic Signal Control

Traffic Signal Control is a complex challenge due to the unpredictable and dynamic nature of traffic flow in urban environments. Traditional methods like fixed-time scheduling and adaptive strategies such as max-pressure control and Model Predictive Control (MPC) have been utilized to manage traffic effectively. However, these approaches often require detailed traffic models and extensive domain knowledge, making them less adaptable to rapidly changing traffic conditions.

With the increasing availability of real-time data from sensors and vision systems, there is an opportunity to develop methods that can leverage this data more effectively. Deep Reinforcement Learning offers a promising solution by learning control policies directly from data without relying on explicit traffic models.

The application of Deep Reinforcement Learning in Traffic Signal Control has been studied and developed over time. Several works utilize Deep Q-Networks (DQN) [35, 39, 43, 22], while others apply Proximal Policy Optimization combined with a traffic signal control unit to manage minimal light times [24].

In this chapter, we focus on using Proximal Policy Optimization in single-agent setting to manage traffic in a single 4-legged intersection. In section 3-1, we introduce the TSC environment and terminology. We will cover classical control methods such as fixed-time and max-pressure strategies, as well as the traffic simulation models. In section 3-2, we explain the integration of PPO into TSC in terms of design choices in observation space, action space and reward function. Thereafter, we present the novelty high sampling frequency approach in combination with strategies like Invalid Action Masking and penalizing invalid actions. Finally, we present a framework capable of handling pedestrian traffic flows at the intersection.

3-1 Traffic Signal Control

The following section introduces the Traffic Signal Control problem. The case specific configuration is presented, the simulation models and the traffic generation models. Finally, classical Traffic Signal Control policies are introduced.

3-1-1 Intersection configuration

In this research scope, a single 4-legged intersection is considered. In Figure 3-1, a schematic overview of the crossing is shown. The intersection has traffic coming and flowing from four directions; North, West, South and East. From each direction, vehicles can flow into the other three remaining directions; turn left, go straight or turn right. This results in twelve directions in total. This defines the configuration of the intersection and remains the same for varying amounts of lanes l. In Figure 3-1, a 12 lane layout is shown, which is also used for the case studies. The lanes are numbered clockwise starting from the link North to East. The incoming lanes are therefore notated as:

$$l_i \in \mathcal{L} \quad \text{where } \mathcal{L} = \{l_0, l_1, \cdots, l_{in-max}\}$$

Similarly, the outgoing lanes are defined as:

$$l_{out,i} \in \mathcal{L}_{out} \quad \text{where } \mathcal{L}_{out} = \{l_{out,0}, l_{out,1}, \cdots, l_{out,out-max}\}$$

A *link* is defined as a path from an incoming to an out coming lane. Logically, a lane l can have multiple links. Each link is controlled with a single traffic-signaler. The controlled links are denoted as $c_{i,o}$, where i is the incoming lane index and o is the outgoing lane index. For Figure 3-1, the set of traffic signalers is defined as:

$$c_{i,o} \in \mathcal{C} \quad \text{where } \mathcal{C} = \{c_{0,11}, c_{1,7}, \ldots, c_{i,o}\}$$

In Figure 3-1, every lane has a single link, therefore, the $c_{i,o}$ notation can be simplified to indexing the links clockwise, similarly to the l indexing. Also, the controlled links c can have three possible values: green, red and yellow. Intuitively, green indicates the cars can leave the lane, red indicates the cars have to wait and yellow indicates a transition between green and red. Therefore, the following holds:

$$c_{i,o} = \begin{cases} \text{'g'} & \text{if cars can leave the lane} \\ \text{'r'} & \text{if cars have to wait} \\ \text{'y'} & \text{if in transition between green and red} \end{cases}$$

A *phase* is defined as a specific combination of controlled links in which none of the controlled links conflict. In this thesis, both an 4 phase and an 8 phase configuration was used in the control of the intersection. These are based on the *ring-and-barrier* scheme, a broadly used phase configuration scheme [31]. in Figure 3-2, all the possible phases are shown. Phases

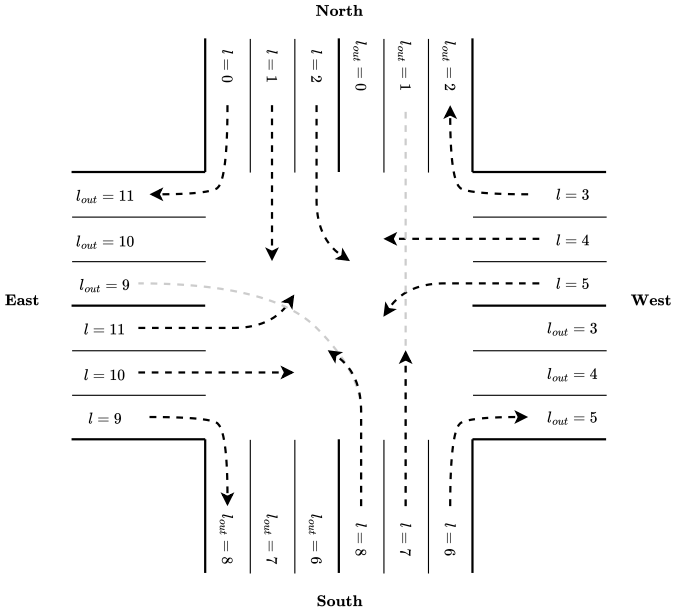


Figure 3-1: Schematic overview of an intersection with four directions, twelve incoming lanes l, twelve outgoing lanes l_{out} and twelve controlled links c.

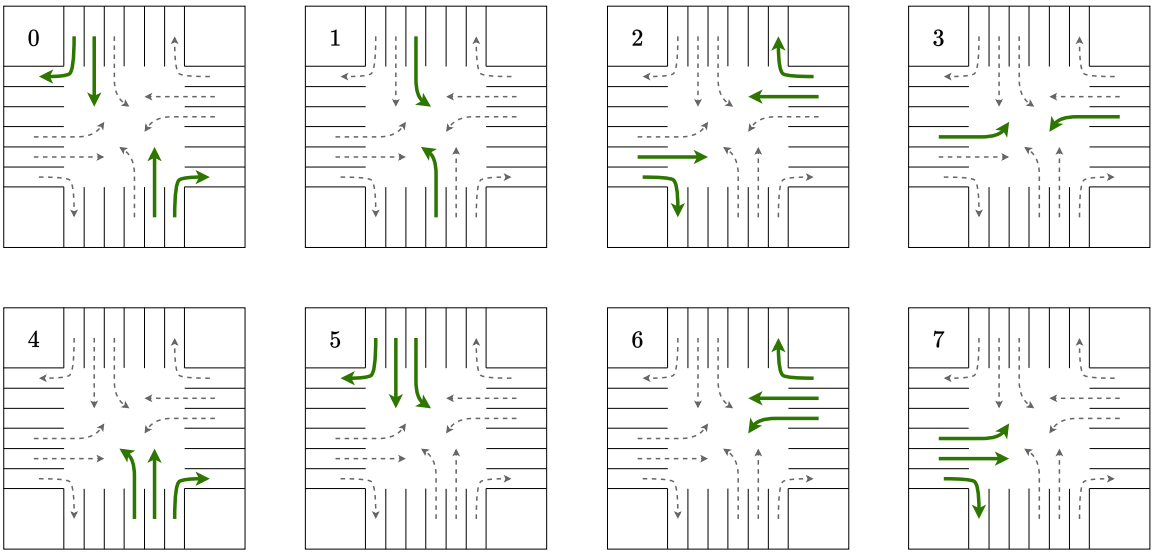


Figure 3-2: Overview of the possible phases. In the 4-phase approach, phase 0 to 3 are included. In the 8-phase approach, all the phases above are included.

can also be referred to as actions. When using the 4-phase configuration, only the phases $\{0, 1, 2, 3\}$ are possible to transition to. In the 8-phase configuration, the applied policy or algorithm can transition to all of the phases 0 to 7.

In Traffic Signal Control, there are multiple traffic signaler constrains such as the *yellow time* and the *minimal green time (MGT)*. The yellow time is the minimal time that a part of the phase is to be in transition before completely switching to a new phase. In most of the US, the yellow time must be a minimum of 3 seconds and a maximum of 6 seconds [1]. Minimal green time assures that once a lane switches to green, there is a minimal time it needs to maintain the green state. This is also to assure safety and reliability. These constrains will play a important role in the development of a high-frequency sampling approach. In the following section, we will present the traffic flow models and simulation tools.

3-1-2 Simulation models

Real-life traffic scenarios are inherently highly non-linear and chaotic due to the unpredictability of driving behavior and the constant cause-and-effect of small movements. Traffic can be modeled on a macroscopic and microscopic level. On a macroscopic level, the vehicles are considered as unified flows and not as individual vehicles. A relatively simple approach is to model these flows as a linear, first order set of differential equations which represent the queue length per lane over time [2]. To improve the model and better fit the non-linear behavior, models in increasing complexity exist such as the S-model [20].

The idea of microscopic modeling of traffic flow is to describe the dynamics of each individual vehicle as a function of the positions and velocities of the neighboring vehicles. In general, the two dynamical processes have to be considered; a car-following model and a lane-switching model.

Car-following model

The fundamental model underneath microscopic traffic flow simulators is a car-following model of which there are many variations in literature. In this thesis, it was decided to use the Krauss Car-following model [18]. It is a safe-distance model that models how a vehicle adjusts its speed based on the distance to the vehicle ahead. This model ensures that each vehicle maintains a safe distance to avoid collisions, accounting for current speed, desired speed, and the deceleration capabilities of the vehicle. These dynamics are governed by multiple equations and multiple tuning parameters to adjust the driving behavior. The Krauss model is computationally efficient, making it suitable for large-scale simulations and widely used in traffic simulators [18]. An open source traffic simulator that uses this model is Simulator of Urban MObility (SUMO).

Lane-switching model

Besides the car-following model, ensuring realistic one-dimensional driving behavior, a lane-switching model is used to replicate driving behavior across multiple lanes. For this thesis, the 'LC2013' lane-changing model was used [5]. This model derives lane-changing rules based on multiple parameters which reflect the decision process of changing lanes from a drivers

perspective. The 'LC2013' model formulates safety and incentive criteria, preventing critical lane changes and considering the advantages and disadvantages of other drivers using the 'politeness factor'. This allows varying motivations from egoistic to cooperative driving behavior, which creates realistic lane-changing dynamics.

3-1-3 Classical methods

In the following section, two classical Traffic Signal Control methods are presented; a fixed-time and a max-pressure approach. Fixed-time is a previously defined approach that does not leverage information of the traffic-state, whilst max-pressure is an adaptive method which does use real-time information on the state of traffic [4]. Both methods will serve as a baseline for the Deep Reinforcement Learning approach.

Fixed-time

Fixed-time does not leverage any real-time information of the traffic state. Therefore, it is a relatively naive approach. the current phase cycles through either the 4-phase or the 8-phase configuration, according to predefined time-intervals.

Max-pressure

The max-pressure approach is an adaptive approach as it adapts to the traffic conditions in real-time. For every controlled link, the 'pressure' difference is calculated between the incoming lane and the outgoing lane. Pressure in Traffic Signal Control (TSC) is defined as the difference in density of lanes between the incoming and outgoing lanes. Mathematically, pressure P for a link (i, o) can be represented as:

$$P_{i,o} = \frac{N_i}{N_{i,max}} - \frac{N_o}{N_{o,max}}$$

where N_i is the number of vehicles in the incoming lane, N_o is the number of vehicles in the outgoing lane and $N_{i,max}$ and $N_{o,max}$ are the maximal number of vehicles that fit on the lane in- and outgoing lane, respectively. The phase with the highest pressure is chosen at time step t:

$$a(t) = \arg\max_{a \in \mathcal{A}} \sum_{(i,o) \in a} P_{i,o}$$

where $\mathcal{A}$ is the set of all possible actions or phases, and $(i, o) \in a$ represents the links controlled by phase a. This selection aims to maximize the pressure relief on the network. Since outgoing lanes have no pressure (i.e., $N_o = 0$) the max-pressure approach operates as a greedy method [37].

3-2 Deep Reinforcement Learning in TSC

This section presents the application of Proximal Policy Optimization to the Traffic Signal Control problem. First, the architecture and design choices for the Proximal Policy Optimization agent's parameters are discussed. Next, the concept of high-frequency sampling is introduced, opposed to low-frequency sampling. Following this, four strategies are proposed to address the challenges posed by minimal green time complexities. Finally, a multi-model DRL framework incorporating both pedestrian and vehicle traffic is presented.

3-2-1 Architecture

In the following section, the design choices and architecture of a Deep Reinforcement Learning Proximal Policy Optimization approach to a 4-legged single traffic intersection are presented.

Observation space

When using Deep Reinforcement Learning in Traffic Signal Control, the observation space can be defined in various ways. To include information on the state of the vehicles, the queue length per lane and the density per lane are commonly considered [42]. The observation space at time-step t should contain enough information to map the states and actions consistently to reward. A larger observation space requires more computational demand, while a too small observation space may lack the information needed for effective decision-making. To find an effective observation space representation, there are various methods such as self-supervised and unsupervised pre-training [45]. In this thesis, given the range of computational strength and time, we have mainly considered the following observation entries for the vehicle 4-legged intersection. At each time-step t, the observation space can be represented as a vector $\vec{s}_t$. This vector equals the s_t variable as defined in chapter 2.

$$s_t : \quad \vec{s}_t = [\vec{s}_{\mathrm{CF}}, s_{\mathrm{MGT}}, \vec{s}_{\mathrm{QL}}, \vec{s}_{\mathrm{LD}}]$$

Where:

- $\vec{s}_{\mathrm{CF}}$: **Current phase (CF)** — The current phase active in the environment at time step t, represented as a one-hot encoded vector. The length is equal to the size of the action space. Example: $[0, 0, 1, 0]$ indicates that the third phase is active in a 4-phase configuration.
- s_{MGT}: **Minimal green timer (MGT)** — A scalar value that indicates whether any minimal green time constraint is active at time step t. Example: [1] indicates there is a minimal-green time running.
- $\vec{s}_{\mathrm{QL}_i}$: **Queue length (QL)** — The normalized length of the queue of cars per lane i, where a car is considered part of the queue if it is traveling at a speed lower than 0.1 m/s. The dimension of this part of the vector corresponds to the maximum number of input lanes l_{in-max}.

- $\vec{s}_{\text{LD}_i}$: **Lane density (LD)** — The normalized density of the cars present in lane i at time step t. If a lane is completely full, the density is equal to 1. The dimension of this entry also corresponds to the maximum number of input lanes l_{in-max}.

Action space

In literature, many possible action spaces are considered. Whether or not to switch to the next phase in a cycle based sequence, the phase duration in a cycle based-sequence or what phase to choose next in a non-cycle based sequence [42]. In this work, we proposed the latter. As described in subsection 3-1-1, the action space used was either the 4-phase or the 8-phase configuration, as visualized in Figure 3-2. The agent was free to choose any action from any previous action. The action space at each time-step t can be represented as a vector of probabilities conditioned on the current state s_t, denoted as:

$$\vec{a}_t = [p(a_1|s_t), p(a_2|s_t), \ldots, p(a_n|s_t)]$$

Where $p(a_i|s_t)$ represents the probability of selecting action a_i given the state s_t, and n is the number of possible actions. As described in subsection 2-2-2, during training the performed action will be chosen according to the probability distribution. During the testing, the action with the highest probability will be performed.

- **Dynamic transition states:** In case the agent wants to switch phase, we calculate a transition state between the current and the next phase. We check a conflict matrix, a matrix containing all conflicting lanes, to see if there are any conflicts between the two phases. In some cases, lanes can maintain the green state over a transition.
- **Prolonged green states:** If a link $c_{i,o}$ is green in the current phase and in the next, the green light is maintained. The same goes for a red state.

Reward Function

The reward function guides the learning process of the agent, aiming to minimize the total time-loss for all traffic. Two reward function options were implemented. The first is the *Negative Summed Queue Length*, which penalizes the agent based on the total queue length across all lanes and vehicles. It is defined as:

$$r_t = -\sum_{i=1}^{L_{in,max}} \sum_{j=1}^{v_{last}} q_{i,j}(t) \tag{3-1}$$

where r_t is the reward at time step t, $q_{i,j}(t)$ represents the queue contribution of vehicle j in lane i at time t, $L_{in,max}$ is the total number of lanes, and v_{last} is the index of the last vehicle in each lane. A smaller queue length results in a less negative reward, encouraging the agent to minimize queues.

The second option is the *Negative Summed Difference in Delay*, which focuses on minimizing the difference in delays experienced by all vehicles. It measures the change in waiting time between two time steps and penalizes the agent for increasing delays. It is defined as:

$$r_t = - \sum_{i=1}^{L_{in,max}} \sum_{j=1}^{v_{last}} \Big[\text{delay}_{i,j}(t) - \text{delay}_{i,j}(t-1) \Big] \tag{3-2}$$

where $\text{delay}_{i,j}(t)$ represents the waiting time of vehicle j in lane i at time t, and $\text{delay}_{i,j}(t-1)$ is the waiting time for the same vehicle in the previous time step. The summation covers all lanes till $L_{in,max}$ and vehicles v_{last}. By penalizing increases in waiting time, this reward function encourages smoother traffic flow and reduces variations in delays.

3-2-2 A High-Frequency Sampling Approach

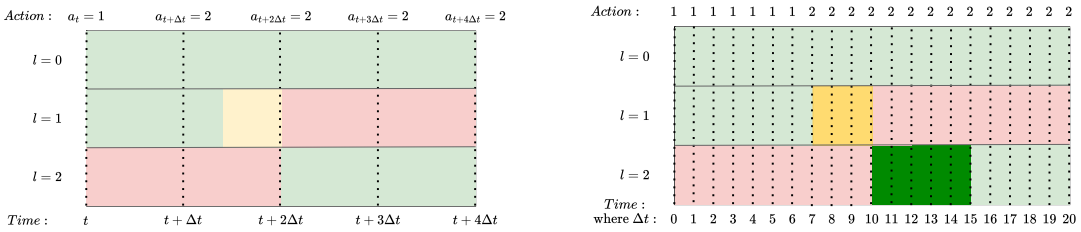
To the best of our knowledge, the effect of the sampling frequency for Deep Reinforcement Learning approaches in Traffic Signal Control has not been thoroughly considered. The time in between actions is referred to as the delta time Δt, which is inversely proportional to the sampling frequency. Some works use variable Δt with a minimal of 3 seconds [35]. Others use constant high update delta times between the 5-25 seconds [39, 43]. Other works do not explicitly mention the Δt used during training and testing [22]. We propose a high-frequency sampling approach, where the agent will update its observation space and provide an action every second. As mentioned in chapter 1, this could lead to a more agile and flexible policy, which can lead to better performance in terms of time-loss and disturbance rejection.

In Figure 3-3a and Figure 3-3b, a visualization of the low-frequency sampling (LFS) and high-frequency sampling (HFS) approach is presented. In both figures, the states of the traffic signalers of three separate lanes, $l = 0$, $l = 1$ and $l = 2$ are depicted. $l = 0$ maintains the green state over all of time. $l = 1$ transitions from green to red and $l = 2$ transitions from red to green. The dotted lines indicate the moments in time where the agent observes the state-space and produces an action. In Figure 3-3a, on $t + \Delta t$, the agent decides to switch from action $a = 1$ to action $a = 2$.

In Figure 3-3b, the sampling occurs every second. Therefore, the agent considers the state-space and produces an action probability every second. Without any constraints or modifications, this approach could violate the yellow time and the minimal green time constrains in $l = 1$ and $l = 2$, respectively. In the following paragraphs, four strategies are proposed to counter-act on this problem; Action Overrule, penalization, Naive Invalid Action Masking (NIAM) and Non-naive Invalid Action Masking (NNIAM).

3-2-3 Invalid Action Masking

The core idea of Invalid Action Masking is to "mask out" actions that are invalid in the current state-space. Invalid Action Masking has been extensively applied in solving big open-map computer games with significantly large discrete action spaces in combination with Deep Reinforcement Learning [25, 38]. In policy optimization methods, Invalid Action Masking refers to an extra operation on the output probability distribution of the action space. Invalid



(a) Low frequency sampling.

(b) High frequency sampling.

Figure 3-3: Comparison of low-frequency sampling (LFS) and high-frequency sampling in equal traffic signaler configurations. The picture displays when the decision moments occur (dotted vertical lines) and when the actual decision occurs (colored background). In Figure 3-3b, the constraint states are colored more strongly.

Action Masking can be subdivided into two types; *Naive Invalid Action Masking* and *Non-naive Invalid Action Masking* [13] which will be presented in subsection 3-2-4. In this section, the case-specific action masking function and the industry standard Action Overrule will be presented.

Masking function

Invalid Action Masking (IAM) starts with a masking function, $m(a|s)$. This function takes in a state s and an action a and applied case-specific logic to decide whether the action is valid. It outputs a 1 if the action is valid in the state and a 0 if it is invalid. The masking vector $\mathbf{m}$ is defined as the action validity for the full action space for a certain state, defined as:

$$\begin{aligned} \mathbf{m} &= \left[m_a\left(a_1 \mid s\right), m_a\left(a_2 \mid s\right), \cdots, m_a\left(a_n \mid s\right)\right], \\ \text{where } m_a\left(a_i \mid s\right) &= \begin{cases} 1, & \text{if } a_i \text{ is valid in } s, \\ 0, & \text{if } a_i \text{ is invalid in } s. \end{cases} \end{aligned} \tag{3-3}$$

The masking function build into the single 4-legged intersection produces either a 0 or a 1 based on the yellow times and minimal green times. During simulation, we keep track of the yellow time and minimal green time per link per time step. If a new action is inserted, every individual link will be checked on whether it will switch to another state and whether that would violate any minimal time constraints. Since it is done on link level (per individual link c_i), flexible action transitions are possible, as visualized in Figure 3-4.

Action Overrule

Action Overrule is a relatively straightforward approach. In the case of Proximal Policy Optimization with deterministic action-sampling, the agent outputs an action for $t + 1$. This action is fed to the masking function $m(a|s)$, which will either declare it *valid* or *invalid*. If valid, the environment is fed the preferred action. If invalid, the environment is fed the previous valid action. In Figure 3-5, a schematic overview of the masking function and Action Overrule can be observed.

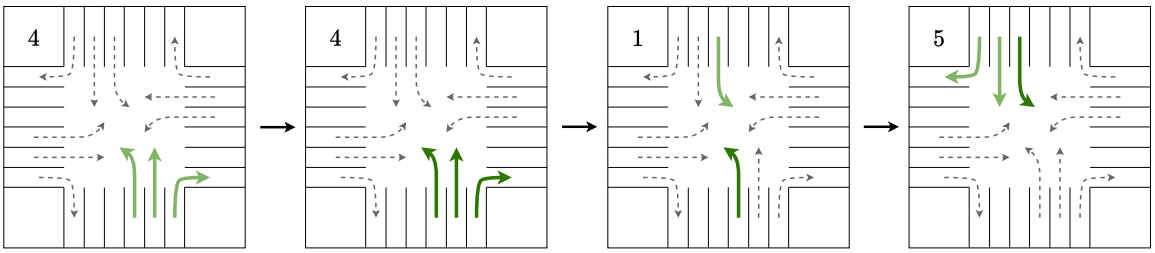


Figure 3-4: Light green arrows represent green state links which have not fulfilled its minimal green time yet and dark green arrow means they have fulfilled the minimal green time and can be changed. Due to link depended masking mechanics, this phase transitions and comparable transitions are possible, opposed to more strict minimal green time mechanics.

This approach can be considered strongly naive because when an action is invalid and 'overruled' it is not performed in the actual environment whilst it is fed back to the agent as if it is performed. This causes the agent to link certain actions in certain states to certain rewards whilst these actions did not actually occur. Therefore, it can perform random actions during these occurrences whilst not receiving any constructive feedback. Since there is a MGT timer in the observation space, the agent can link this "ineffectiveness" to the MGT being 1. When the MGT is 0, the agent does have effect, since its actions are actually implemented.

Penalizing Invalid Actions

A possible method to minimize invalid actions is by penalizing invalid actions. Every time the agent takes an invalid action, it is penalized by a certain negative scalar. This method could learn the agent to not take invalid actions and possible improve or worsen its performance. However, this can be considered a *soft constraint* as we still need another mechanism to overrule the invalid action chosen.

3-2-4 Naive- and Non-Naive Invalid Action Masking

Naive- and Non-Naive Invalid Action Masking in policy optimization methods both employ the same forward pass: The action probabilities of invalid actions in state s are set to zero, thereafter the probability distribution is re-normalized and the action is sampled. The difference between Naive and Non-Naive is in the propagation [14].

Naive Invalid Action Masking

The masked policy is defined as the policy in which the action probabilities of the invalid actions are set to zero and is notated as $\pi_\theta^m(a|s)$. In NIAM, the masked policy is used to interact with the environment. However, the unmasked policy is used to update the L^{CLIP} loss function and perform stochastic gradient descent over the parameter θ, as visualized in Figure 2-4. Therefore, the agent is not updated over the perturbed action probabilities and is *'naive'* to the masking concept.

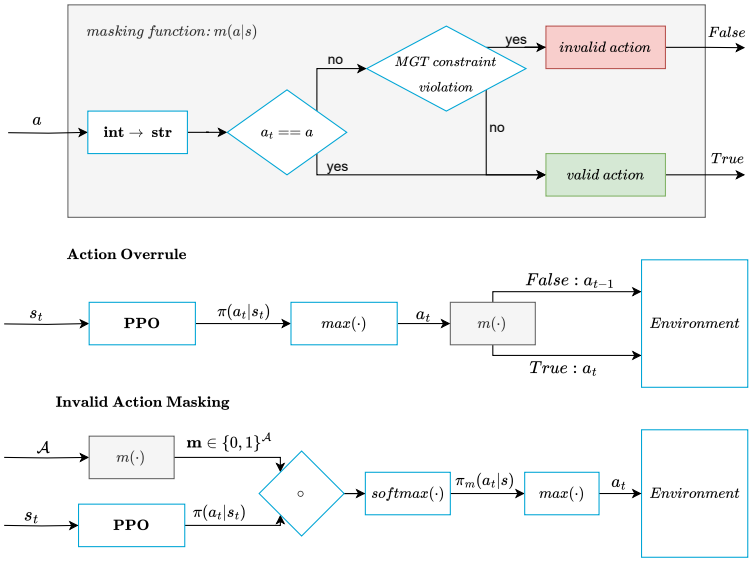


Figure 3-5: The forward pass mechanics of action adjustment strategies. In gray, the masking function is visualized. Below, the *Action Overrule* and Invalid Action Masking strategy are depicted.

Non-naive Invalid Action Masking

Non-naive Invalid Action Masking, as the name suggests, does use the masked probability distribution to update the Proximal Policy Optimization parameters θ. Therefore, the term $r_t(\theta)$ in Equation 2-8 will be constructed with masked distributions:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \quad \rightarrow \quad r_t^m(\theta) = \frac{\pi_\theta^m(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}^m(a_t \mid s_t)} \tag{3-4}$$

Also, the policy used to calculate the entropy bonus in Equation 2-11 is replaced by the masked policy. Huang et. al. empirically showed in an experiment that Non-Naive converges slightly faster during training [14]. Hou et. al. continues on this concept, provides mathematical proofs and presents a novel Invalid Action Masking approach where Naive and Non-Naive strategies are combined [13]. In Figure 3-5, the architecture with the (Non)-Naive Invalid Action Masking functionality is depicted.

In Algorithm 1, a pseudo-code of the full Proximal Policy Optimization algorithm with the Non-naive Invalid Action Masking implementation is shown. The difference between naive and non-naive Invalid Action Masking is also visible here; If we use the non-masked policy π_θ in line 22 and 25 to calculate the ratio and the entropy loss, respectively, we would have Naive Invalid Action Masking. If we do use the masked policy π_θ^m, we have Non-naive Invalid Action Masking.

Algorithm 1 Proximal Policy Optimization with Non-Naive Invalid Action Masking

$\theta \leftarrow$ initialize random weights
while not terminated or truncated **do**
  $s_0 \leftarrow$ initiate state
  for step $= 1, \ldots, T$ **do**
    $s \leftarrow s'$ previous end state is new state
    $a \sim \pi_\theta^m(a|s)$ use masked policy to sample action
    Apply a and simulate forward one step
    $s' \leftarrow$ end state
    $r \leftarrow$ reward
    record sample (s, a, r, s') into memory D
  end for
  for each (s_i, a_i, r_i, s_i') in D **do**
    $R_i \leftarrow$ compute return using $R(\lambda)$ and add to memory D
    $\hat{A}_i \leftarrow$ compute advantage using $GAE(\lambda)$ and add to memory D
  end for
  $\theta_{\text{old}} \leftarrow \theta$
  for each epoch **do**
    Randomly shuffle samples in memory D
    for each minibatch **do**
      Sample minibatch of n samples $\{(s_i, a_i, r_i, s_i', R_i, \hat{A}_i)\}$ from D
      for each sample in minibatch **do**
        Compute policy ratio $r_i(\theta) = \frac{\pi_\theta^m(a_i|s_i)}{\pi_{\theta_{\text{old}}}^m(a_i|s_i)}$
        Compute policy loss $L_i^{\text{CLIP}} = \min\left(r_i(\theta)\hat{A}_i,\ \text{clip}\left(r_i(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right)$
        Compute value function loss $L_i^{\text{VF}} = \left(V_\theta(s_i) - V_i^{\text{targ}}\right)^2 \quad \text{note}: V_i^{\text{targ}} = R_i$
        Compute entropy loss $L_i^{\text{S}} = -\sum_{a\in\mathcal{A}} \pi_\theta^m(a|s_i) \log \pi_\theta^m(a|s_i)$
      end for
      Compute $L^{\text{CLIP+VF+S}} = \frac{1}{n}\sum_{i=0}^{n}(L_i^{\text{CLIP}} + L_i^{\text{VF}} + L_i^{\text{S}})$
      $\theta \leftarrow \theta + \alpha \nabla_\theta L^{\text{CLIP+VF+S}}$
    end for
  end for
end while

3-2-5 Pedestrian Traffic Flows

In many real-life scenarios, pedestrians form a part of the traffic flow mechanics. Therefore, pedestrians were included to the framework. In several recent works, pedestrian traffic flows are included [10]. These do not consider minimal light times and higher sampling frequencies. In Figure 3-6a, the new intersection layout can be seen in the SUMO simulation environment.

Observation Space

To make the agent aware of the new pedestrian traffic flows, three elements were added to the observation space, as summarized below. In Figure 3-6b, the new observation space elements

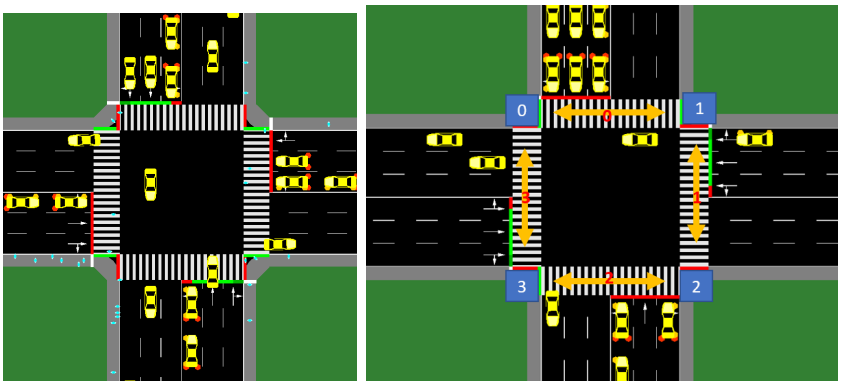
can be observed. At each time-step t, the extended observation space can be represented as a vector $\vec{s}_t$:

$$s_t: \quad \vec{s}_t = [\vec{s}_{\text{CF}},\ s_{\text{MGT}},\ \vec{s}_{\text{QL}},\ \vec{s}_{\text{LD}},\ \vec{s}_{\text{WA}},\ \vec{s}_{\text{CD}},\ s_{\text{MGT-PED}}]$$

Where:

- $\vec{s}_{\text{WA}}$: **Walking Area Densities** — The normalized densities of pedestrians in each walking area. The walking area is defined as the pedestrian corner on the intersection. For a four-legged intersection, there are four walking areas in total, labeled "0" to "3" as can be seen in Figure 3-6b. The normalized density is defined as $WA_{\text{count}}/WA_{\text{max count}}$.
- $\vec{s}_{\text{CD}}$: **Crossing Densities** — The normalized densities of pedestrians waiting to cross each crossing. There are four crossings labeled "0" to "3". Pedestrians in walking areas contribute to the crossing densities, providing more precise information on the direction of the pedestrians.
- $s_{\text{MGT-PED}}$: **Minimal Green-time pedestrians** — A scalar value that indicates whether any minimal green time constraint for pedestrians is active at time step t. Pedestrians need more green time to fully cross; therefore, this timer strictly follows the pedestrian green times.

As these three elements are added to the observation space, the total number of elements in the extended observation space is now 44.



(a) 4-legged intersection with pedestrian traffic flows.

(b) Observation space extension visualization for pedestrian traffic flows.

Figure 3-6: A visualization of the expansion to a pedestrian traffic flow included environment. In (b) we can observe the "walking areas" in blue, labeled 0 to 3. Also, we can observe the crossings, indicated with a orange arrow, labeled 0 to 3 in red.

Action Space

To have pedestrians cross, there are pedestrian crossing phases needed. The most basic action space expansion to include pedestrian crossings is by introducing two new phases which are

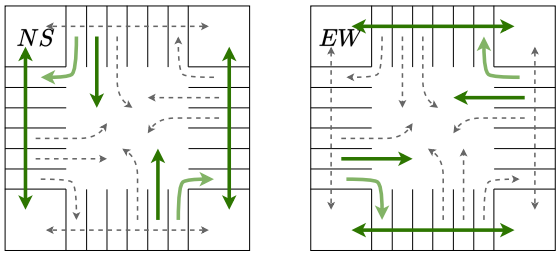


Figure 3-7: Pedestrian actions North-South (Left) and East-West (Right). Long bidirectional arrows indicate pedestrian crossings. Light-green arrows indicate movement is allowed, but drivers must yield to pedestrians crossing.

visualized in Figure 3-7. Therefore, the 4- and 8-phase action space configurations as defined in Figure 3-2 can be expanded to include pedestrians. Adding the new pedestrian phases this would give two new configurations named: 6-phase and 10-phase action space.

A further expansion can also be done to more possible actions, where it is also allowed to have partial pedestrian crossings. This included having on only one pedestrian crossing for instance. The expansion would end up in a 16-phase action space. This would increase the training time as the agent is to explore more possible state-action configurations.

Reward Function

The reward function is also to include pedestrian traffic flows. This will be a new element in the reward function. The reward function for the pedestrian traffic flow framework is defined as:

$$r_t = w_1 \times \Delta D_{t,\text{veh}} + w_2 \times \Delta D_{t,\text{ped}} \tag{3-5}$$

The reward function consists of the following elements. r_t is the total reward at time t. The term w_1 represents the weight factor for vehicle traffic optimization, and $\Delta D_{t,\text{veh}}$ denotes the change in vehicle delay at time t as defined for vehicles in Equation 3-2. Similarly, w_2 is the weight factor for pedestrian traffic optimization, and $\Delta D_{t,\text{ped}}$ signifies the change in pedestrian delay at time t.

Pedestrians or vehicles can be prioritized by adjusting the weight factors w_1 and w_2 in the reward function. Increasing w_1 places more emphasis on reducing vehicle delays, thus prioritizing vehicle traffic flow. Conversely, increasing w_2 emphasizes reducing pedestrian delays, thereby prioritizing pedestrian movement.

3-3 Conclusions

In this chapter, we introduced Traffic Signal Control for 4-legged single intersections. We presented the configuration, the simulation tools and the classical control methods such as a max-pressure and fixed-time approach. Thereafter, we expanded to using Deep Reinforcement Learning in Traffic Signal Control. We presented recent works in this field which relate

to our research and presented the flaws in these works: Low-frequency sampling and little consideration of pedestrian traffic flows.

Next, we presented the main idea of a high-frequency sampling Deep Reinforcement Learning approach to the problem and we identified newly introduced complexities. For these complexities, we presented various possible solutions such as NIAM, NNIAM and penalizing invalid actions. We also introduce a framework which includes pedestrian traffic flow flows and can potentially prioritize either pedestrians or vehicles on over the other.

In the following chapter, we will present the case studies which were conducted to test the research questions and hypothesis we presented.

Chapter 4

Case Study

This chapter presents three case studies. We begin with section 4-1, which outlines the general setup, including how the `Python` frameworks, libraries, and simulation tools are interconnected. This section also describes the traffic demands, the evaluation procedures for traffic control policies, and the performance metrics used.

We then present three case studies: A-I, A-II, and B.

In section 4-2 contains Case Study A: Here, we evaluate the performance of the PPO model over various sampling frequencies. We approach it in a relatively straightforward manner as we apply the low-frequency Action Overrule approach to all models. In section 4-3 we present Case Study A-II: We take a deeper dive into the high-frequency sampling PPO model, applying NNIAM and parallel training in an effort to improve performance. In this case study, we also conduct a detailed comparison between the PPO models and baseline models such as max-pressure and fixed-time. We perform several sensitivity analyses and conclude with an assessment of how effectively invalid actions are handled. Finally, in section 4-4, we present Case Study B: It shifts focus from high-frequency sampling to address the second sub-research question. We apply NNIAM and train models to prioritize between traffic modes in this multi-modal traffic scenario.

4-1 Set-up

This section outlines the set-up for the case studies. We present the intersection layouts used, the configuration of the `Python` classes, the traffic demands and their variations from the baseline flows, the random seed generators for reproducibility, the testing settings for evaluating policies, and the performance metrics used to judge and evaluate the models.

4-1-1 Software Design

To facilitate the case studies, we constructed multiple interconnected `Python` classes that share information at each time step. Figure 4-1 illustrates the inter connectivity of these classes.

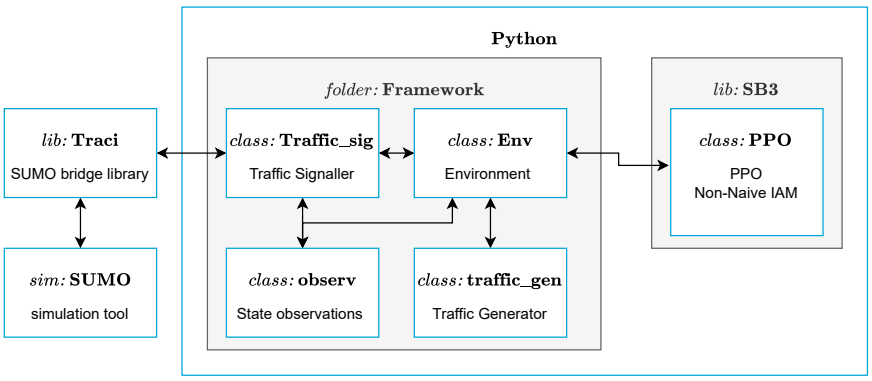


Figure 4-1: Flow of information in the Python setup. This overview provides a general idea of the various Python classes built to run the experiments. Arrows indicate data flow.

The `Env` class functions as the environment. We leaverage the open-source `Gymnasium` library, which allows us to model custom systems as a Reinforcement Learning framework. This class communicates with `Traffic_sig`, which controls the traffic signals. It can overwrite actions and start a protocol to send actions to the simulation tool SUMO. To communicate with SUMO, we use the open-source Python library `Traci` as a data pipeline to communicate with SUMO. The `Stable Baseline 3 (SB3)` library allows us to integrate PPO mechanisms into the environment. The `observ` class observes the traffic conditions and encodes them to an observation space. Finally, the `Traffic_gen` class generates various traffic volumes, modes, and variations for the environment. This setup provides a general framework for the configuration and data flows in our experiments.

4-1-2 Simulation Configuration

Intersection Layout

We consider two intersection layouts in the case studies. The first is a vehicle-only, four-legged single intersection, described in subsection 3-1-1, used in Case Studies A-I and A-II. The second is an intersection with both vehicle and pedestrian traffic, described in subsection 3-2-5, used in Case Study B.

Traffic Demand Modeling

We describe the traffic demand settings used in the simulations. Traffic demands are characterized by flow rates, with vehicle inter-arrival times sampled from an exponential distribution to introduce randomness and mimic real-world traffic conditions. Vehicle dynamics follow the Krauss car-following model, as described in subsection 3-1-2. In the baseline scenario, straight-going lanes have a flow rate of 300 vehicles per hour, and turning lanes have a flow rate of 150 vehicles per hour, establishing a 2:1 ratio between straight and turning movements. Table 4-15 With a total of four straight lanes and eight turning lanes, this gives a total throughput of 1800 veh/hour for the baseline scenario. In the course of the case studies, more varying traffic volume scenarios will be introduced and applied.

4-1-3 Reproducability and Testing

Seeding

In the experiment setup, there are multiple initial values and seeds which influence the final results. In order to maintain full reproducability and reliability, the existing seeds are stated below:

1. **PPO Seed:** This seed is used to initialize the neural network parameters θ in the Proximal Policy Optimization algorithm. Also, during the training phase, PPO samples from the action probability distribution, which is seeded here.
2. **SUMO Seed:** This seed determines the initial positions of vehicles and the sampling from the exponential distribution for incoming traffic in the SUMO environment. This also influences traffic behavior over time.
3. **Traffic Generator Seed:** When generating highly variable traffic patterns, this seed controls how intervals and volume ranges are sampled from random or normal distributions.

Testing Procedure

To ensure consistent and reliable performance evaluation, all trained policies are tested under identical traffic flow scenarios. Each policy is tested over twenty episodes, each lasting 3100 seconds, equivalent to one hour of traffic simulation. During the final 100 seconds of each episode, no additional traffic inflow is introduced, allowing the policies time to clear the intersection if possible. The episodes use different SUMO and traffic generator seeds to introduce variability while maintaining controlled conditions.

One of the visualization methods we use to analyze the testing data is the *box plot*. A box plot summarizes key aspects of the data distribution, including the median and the interquartile range (IQR). The median represents the middle value when the data are ordered from lowest to highest. The IQR measures the spread of the middle 50% of the data, calculated as the difference between the third quartile (75th percentile) and the first quartile (25th percentile). The "whiskers" of the box plot extend to 1.5 times the IQR above the third quartile and below the first quartile, encompassing most of the data points. Data points outside this range are considered outliers. Using box plots allows us to compare the performance metrics of different policies clearly and get an idea of the spreading of the data.

Another common plot is a time-series with a variance indication. Many plots will show the one standard deviation from the mean value for every time step. This will be display as a band around the mean. Another displays the median value and three percentile ranges per time step. These are the 25-75 percentile, 10-90 percentile and finally the 5-95 percentile. This also gives a proper idea of the distribution of data.

To reduce noise and plot trends more clearly, we apply moving averages to the time-series. For the performance metrics over an episode, we mostly apply a moving average window of twenty simulation time steps. As for the learning curves, we apply the *exponential moving average*. The exponential moving average at time t, denoted EMA_t, is recursively defined by:

$$\text{EMA}_t = \alpha \cdot x_t + (1 - \alpha) \cdot \text{EMA}_{t-1}$$

where x_t is the data point at time t, α (the smoothing factor) is a constant between 0 and 1, and EMA_{t-1} is the exponential moving average at the previous time step. This allows us to see trend more clearly and reduce noise.

4-1-4 Performance Measures

In this subsection, we present the key performance measures used to evaluate the traffic signal control strategies during the testing and training phases. Below follow the testing phase metrics.

Name	Entity	Description
Time Loss	Seconds	Average extra time spent in simulation compared to free-flow conditions per traffic mode.
Average Speed	Meters/second	Average speed of all vehicles.
Number of Vehicles	Count	The amount of vehicles in simulation.
Total Travel Time	Seconds	Total travel time of all vehicles over an episode.
Average Duration	Seconds	Average trip duration for traffic mode over an episode.
Vehicles Halting	Count	Number of vehicles moving at a speed below 0.1 m/s.

Table 4-1: Performance metrics used to evaluate the testing phase.

Name	Entity	Description
Reward Over Time	Reward Value	average reward received per episode over training.
Computational Time	Seconds	Training time and CPU usage during the training phase.
Invalid Actions Over Time	Count	Frequency of invalid actions taken by the agent during training.
Policy Loss	Loss Value	Measures the loss associated with the policy effectiveness.
Value Loss	Loss Value	Loss related to the value function, associated with the agent's estimation accuracy of future rewards.
Entropy Loss	Loss Value	Entropy loss, an indicator of how certain the actions are. Low entropy loss indicates concentrated action probabilities.

Table 4-2: Performance metrics used to evaluate the training phase.

These performance measures will be plotted in the coming case studies. Thereafter, in the conclusion and discussion, they will be thoroughly evaluated.

4-2 Case Study A-I: Performance Across Sampling Frequencies

Case Study A-I examines the variations in performance, specifically in terms of time-loss, across different sampling frequencies. It is presented to observe the effect of varying the sampling frequency without any improvement efforts. Therefore, the *Action Overrule* method, which demonstrates strong performance at a baseline sampling frequency $\Delta t = 5$, is applied to neighboring frequencies for equal comparison. As mentioned, the case considered is the 4-legged vehicle only single intersection described in subsection 3-1-1. In Table 4-3, the Reinforcement Learning parameters and environment configuration parameters are stated. In Table 4-4, the used hyper-parameters for the Proximal Policy Optimization algorithm are presented.

Parameter	**Value**
Episode Length	3000 seconds
Minimum Green Time	7 seconds
Yellow Time	3 seconds
Delta Time	variable
Observation Space	CF, MGT, QL, LD*
Action Space Configuration	4-phase**
Reward Function	Equation 3-2
Traffic volume straight	300 vehicles/hour
Traffic volume turns	150 vehicles/hour

Table 4-3: Environment and DRL Parameters. *The Observation space abbreviations can be found in section 3-2-1. **The exact action space can be found in subsection 3-1-1.

In Table 4-3, the observation space is a combined tensor of the mentioned abbreviations, which can be found in section 3-2-1. For the action space, the 4-phase configuration was used, which can be found in subsection 3-1-1. The reward function is defined as the negative summed difference in delay, as defined in Equation 3-2.

Parameter	**Symbol**	**Value**
Policy	-	MlpPolicy
Learning Rate	α	0.001
Number of Steps	n_{steps}	2048
Mini-Batch Size	$n_{\text{mini-batch}}$	64
Number of Epochs	n_{epochs}	10
Gamma	γ	variable
GAE Lambda	λ	0.95
Clip Range	ϵ	0.2
Value Function Coefficient	c_{vf}	0.5
Entropy Bonus Coefficient	c_{ent}	0
Max Gradient Norm	-	0.5
Network Architecture	-	[128, 256, 128]

Table 4-4: Proximal Policy Optimization Configuration Parameters Case Study A-I.

The value of the "policy" in Table 4-4 refers to *Multi-Layer Perceptron Policy*, which is a type of fully connected neural network as mentioned in subsection 2-2-2. In Table 4-5, the parameter variation over the different models can be observed.

Model	Δt	Training steps [s]
A-I-1	1	500000
A-I-2	2	250000
A-I-3	3	166667
A-I-4	4	125000
A-I-5	5	100000
A-I-6	6	83333
A-I-7	7	71429
A-I-8	8	62500
A-I-9	9	55555

Table 4-5: Overview of the various models trained for Case Study A-I.

As we aim to have a *clean* comparison of varying frequencies, all models should receive the same amount of information, therefore the training time steps were scaled appropriately. This way, all models have seen the same amount of input data.

4-2-1 Results

In the following section, the results of the experiment are presented and discussed. In the first section, the performance of the trained agents against the baseline traffic flow inputs is shown. It is tested according to the testing procedure as described in section 4-1-3. In the next section, the results during the training phase are presented and partly discussed. In the final section, conclusions are drawn and an intro to the Case Study A-2 is given.

Performance

In Figure 4-2, we see the time loss during our testing sessions for all different models. We can observe an increase in time-loss as we sample more frequently. Also, we see a steep increase in time-loss as we sample every 9 seconds. In Figure A-5, there are plots which show the vehicles still running in simulation, the average speed, the average duration of a trip and the total time spend for all vehicles combined. These median metrics and additional mean metrics are summarized in Table A-3, and Table 4-6. In Table A-2, the training time and CPU usage are presented for the training sessions of the various models. In Figure A-10, training data such as the learning curve can be observed.

Δt	Vehicles [#]	Speed [m/s]	Duration [s]	Time Loss [s]	TTT [s]
1	2	6.53	50.58	28.75	96583
2	1	6.60	49.61	27.77	94775
3	1	6.71	48.46	26.70	92626
4	1	6.80	47.74	25.89	91262
5	1	6.77	48.02	26.17	91798
6	1	**6.85**	**47.20**	**25.36**	**90250**
7	**0**	6.72	48.39	26.56	92482
8	**0**	6.62	49.76	27.93	95099
9	1	6.37	52.39	30.55	100147

Table 4-6: Mean Metrics for Different Δt Values.

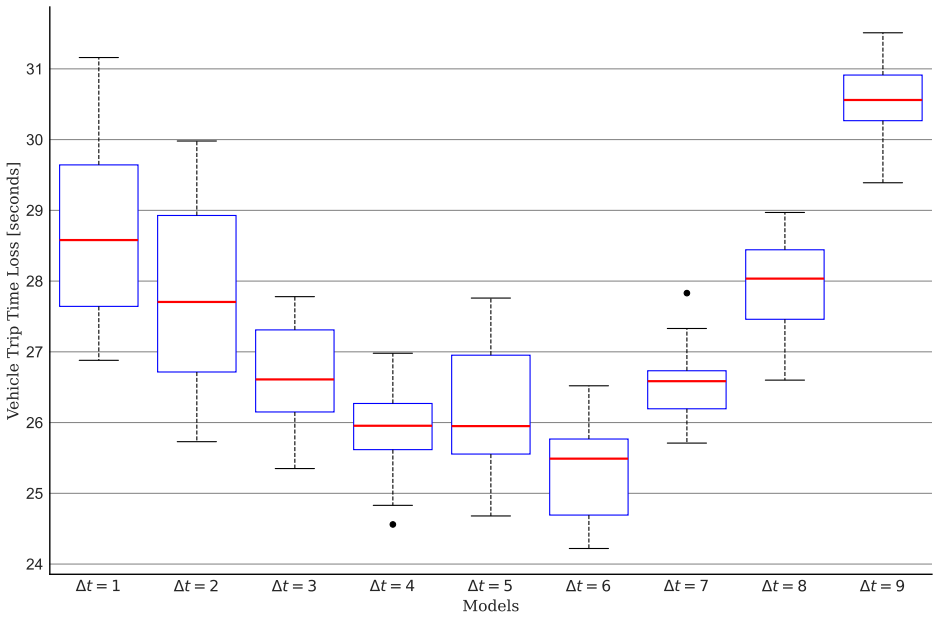


Figure 4-2: Time loss observed during testing procedure for all the A-I models. The chart illustrates a trend of increasing time loss with higher sampling rates, as well as a sharp rise when crossing the $\Delta t = 7$.

The observation that higher-frequency sampling does not improve performance in terms of time-loss for PPO raises an important question: Why does this approach fail to yield improvements, despite the expectation that finer-grained resolution should enable more precise decision-making?

A possible explanation concerns the accuracy and variance of the return R_t and the advantage estimation $\hat{A}$. When sampling every second, we collect calculate a reward (difference in delay) over every second. Most of the time, vehicles are yet to accelerate from 0 km/hour, which is only measurable after multiple seconds. This delay is then discounted with γ and summed to form the return R_t, whereafter it used to calculate the estimated advantage $\hat{A}$, as explained in subsection 2-2-2. This frequent but minimally informative reward calculation can introduce noise, increasing the variance of the estimations, ultimately hindering the learning process. In contrast, the default update interval is five seconds, which has a longer horizon for the return R_t and less measurements, possibly leading to more stable learning. It is important to note that this is a hypothesis and not a confirmed explanation for the performance gap. Further analysis of these findings are discussed in chapter 5.

We can also observe a strong upward trend towards $\Delta t = 9$. A possible explanation is reduced time it takes to react to a change in traffic, the reason we suggested to increase the sampling frequency in the first place. Another reason could be insufficient training time. Figure A-6, suggests that the $\Delta t = 8$ and $\Delta t = 9$ models have not fully converged to an optimum yet. This cutoff was done to provide equal information to all models.

4-2-2 Conclusions

Case Study A-1 is an effort to translate the mechanics of the PPO at low sampling frequencies to high sampling frequencies. From Figure 4-2 and Table A-3, we can conclude that in this environment and deep reinforcement learning setting a higher sampling frequency does not improve performance in terms of time loss. We can observe a clear upward trend. When comparing the best performing frequency $\Delta t = 6$ to the high-sampling frequency model with $\Delta t = 1$ in terms of time-loss in seconds, we can roughly observe a 12 % increase in average time-loss and a 90 % increase in standard deviation.

To improve the performance of high-frequency sampling models, we have made an effort in Case Study A-II. This case study is dedicated to finding possible solutions and improving the performance of the high frequency sampling models.

4-3 Case Study A-II: High-Frequency Approach Improvements

From Case Study A-I, we learned that simply creating a new high-frequency framework does not inherently improve performance. Therefore, in this case study, we aim to improve performance by implementing and evaluating the NNIAM method, as presented in subsection 3-2-4. Additionally, we employ parallel training techniques. In the first section, we test and evaluate these two methods based on their training and testing data, focusing on improving the high-frequency model.

In the subsequent section, we examine six models: the best-performing low-frequency sampling PPO model, the high-frequency sampling PPO model with new implementations, three max-pressure models, and a fixed-time model. First, we perform a sensitivity analysis with varying traffic volumes to determine if the models can maintain performance under consistently higher traffic volumes. We also train models for generalization across various traffic domains to enhance their performance under different traffic conditions. Following this, we conduct a disturbance rejection experiment to evaluate how the models respond to sudden increases in traffic flow.

To further understand the handling of invalid actions, we introduce a new set of models that penalize invalid actions instead of masking them out, as done with NNIAM. Penalizing invalid actions is a different approach, as it allows the agent to choose any action but imposes a penalty when an invalid action is selected. In the final section of this case study, we present and analyze results comparing the effectiveness of NNIAM and the penalty models.

4-3-1 Considered Approaches

In this section, we shortly describe three methods. The first being NNIAM, which as been extensively described in subsection 3-2-4. The second improvement is parallel training. This comes down to training one agent with multiple environments at once. The third approach is penalizing invalid actions, which would minimize the agents deciding on invalid actions through the reward instead of through the action probability distribution.

Non-Naive Invalid Action Masking

The mechanics of Non-Naive Invalid Action Masking have been in depth explained in subsection 3-2-4. It uses a masking function $m(\cdot)$ which sets the probabilities of invalid actions to 0. Therefore, they will not be selected during training and testing. After the masking, the probability distribution is redistributed again to sum to 1. In an 8-phase action space, there can be multiple options after the masking, as explained in Figure 3-4.

Multiple parallel environments.

Parallel environments in Proximal Policy Optimization (PPO) involve running multiple instances of the environment simultaneously during training, allowing the agent to collect experiences from several sources at once. This improves the variation of the data which improves general performance [3]. While using parallel environments is common in PPO implementations, it requires a more complex setup in a `Python` environment.

Penalizing invalid actions.

Invalid Action Masking can be considered a hard constraint; it never allows wrong actions. Therefore, the learning process can be more challenging since there can be little exploration at times. Penalizing invalid actions would be a soft constraint. However, Traffic Signal Control does not allow any errors, therefore just penalizing invalid actions would be dangerous. In this effort, the Action Overrule method is combined with a penalizing method, such that we ensure safety.

To see the effect of a penalty, it was implemented for both the $\Delta t = 1$ and $\Delta t = 5$ model. For both models, the penalty was varied over a applicable range. In Table 4-8, the various trained models with their respective penalties are presented. The new reward function for the penalized models is as follows:

$$r_t = - \sum_{i=1}^{L_{in,max}} \sum_{j=1}^{v_{last}} \Big[\text{delay}_{i,j}(t) - \text{delay}_{i,j}(t-1) \Big] - p$$

Overview models

To maintain an overview of the various models, in the following section, the models are listed. There are 18 new models in total. The first four are $\Delta t = 1$ models with the various improvements. The next four are the same, but operating with the $\Delta t = 5$ mechanism. The latter models are penalty models with varying penalty and update times. In Table 4-7, the hyper parameters of the models in A-II are presented. In Table 4-8, the overview of all models trained and tested in Case Study A-II are shown.

Parameter	Symbol	Value
Policy	-	MlpPolicy
Learning Rate	α	0.001
Number of Steps	n_{steps}	2048
Mini-Batch Size	$n_{\text{mini-batch}}$	64
Number of Epochs	n_{epochs}	10
Gamma	γ	0.99
GAE Lambda	λ	0.95
Clip Range	ϵ	0.2
Value Function Coefficient	c_{vf}	0.5
Entropy Bonus Coefficient	c_{ent}	0
Max Gradient Norm	-	0.5
Network Architecture	-	[128, 256, 128]
Training time-steps	T_{total}	500000

Table 4-7: Proximal Policy Optimization hyper parameters for Case Study A-II.

4-3-2 Results

In the following section, we will present the results of Case Study A-II. We will present data that provides insights into the performance of the models, as well as point out interesting results and discuss possible causes for these results. A more detailed analysis of several

Model ID	Env. Type	NNIAM	Δt	Penalty	Traffic Scenario
A-II-1	SE	No	1	-	Baseline Scenario
A-II-2	SE	Yes	1	-	Baseline Scenario
A-II-3	PE	No	1	-	Baseline Scenario
A-II-4	PE	Yes	1	-	Baseline Scenario
A-II-5	SE	No	5	-	Baseline Scenario
A-II-6	SE	Yes	5	-	Baseline Scenario
A-II-7	PE	No	5	-	Baseline Scenario
A-II-8	PE	Yes	5	-	Baseline Scenario
A-II-9	SE	No	5	1	Baseline Scenario
A-II-10	SE	No	5	2	Baseline Scenario
A-II-11	SE	No	5	3	Baseline Scenario
A-II-12	SE	No	5	4	Baseline Scenario
A-II-13	SE	No	5	5	Baseline Scenario
A-II-14	SE	No	1	0.2	Baseline Scenario
A-II-15	SE	No	1	0.4	Baseline Scenario
A-II-16	SE	No	1	0.6	Baseline Scenario
A-II-17	SE	No	1	0.8	Baseline Scenario
A-II-18	SE	No	1	1	Baseline Scenario
A-II-19	SE	No	5	-	Scenario 2
A-II-20	SE	No	5	-	Scenario 3
A-II-21	SE	No	5	-	Scenario 4
A-II-22	SE	No	5	-	Scenario Gaussian
A-II-23	SE	No	5	-	Scenario Random

Table 4-8: Overview of Trained PPO Models for Case Study A-II. The baseline traffic scenario is defined in Table 4-15, and the other traffic scenarios are elaborated in section 4-3-2.

findings and their possible causes is provided in chapter 5. The results of Case Study A-II are structured into five parts: performance of NNIAM and parallel training, Sensitivity analysis on input traffic volumes, analysis of generalization capabilities, disturbance rejection capabilities and an analysis of the effectiveness of handling invalid actions.

In section 4-3-2, we present the performance of the models with the notable improvement efforts. We will evaluate the effect of NNIAM and parallel training on the performance through the testing metrics and at the training phase through the training data. Thereafter, in section 4-3-2, a sensitivity analysis of the five models is presented. In total, four traffic scenarios are evaluated. Also, in section 4-3-2, we display results concerning generalization over various traffic input domains. Next, in section 4-3-2, we will evaluate the disturbance rejection capacities of the best performing models against baseline performers such as max-pressure and fixed time methods. We will introduce a disturbance and evaluate the performance. Finally, in section 4-3-2, we present results to gain more insights into the effectiveness of the approaches to minimize invalid actions. We will plot the results of the trained penalty models and compare them to the performance of the NNIAM models in terms of time-loss and percentage of actions chosen which were invalid.

Performance NNIAM and Parallel Training

In the following section, the performance plots of the models with and without Non-naive Invalid Action Masking and with and without parallel training are presented. In Figure 4-3, the performance of four different high frequency models is presented. These are the models

A-II-(1-4) in Table 4-8. Again, the testing is done according to the testing protocol described in section 4-1-3. In Table 4-9, the mean values of the performance metrics of the tests are shown. In Figure A-15, the performance parameters are shown more extensively.

Considering the training phase, we can look at the training metrics, one of which is the average return of reward over time, as displayed in Figure 4-4. To evaluate reproducability and robustness of these models, two extra models for both A-II-1 and A-II-2 were trained with the exact same hyper parameters but different initial weights. In Figure 4-5, the mean and standard deviation over the learning curves of these models is shown.

As the performance improvements are not significant they could be merely coincidental. As we can observe, the distributions are very similar in Figure 4-3, with slight improvements in median, mean and variance as we introduce more improvement efforts. The slightly improved performance for parallel trained models opposed to single trained models is expected. Parallel training has proven to improve performance for DRL approaches in various scenarios. For the slight improvement of the NNIAM approach there is no strong scientific explanation yet. Possible causes for the similarity in performance are discussed in chapter 5.

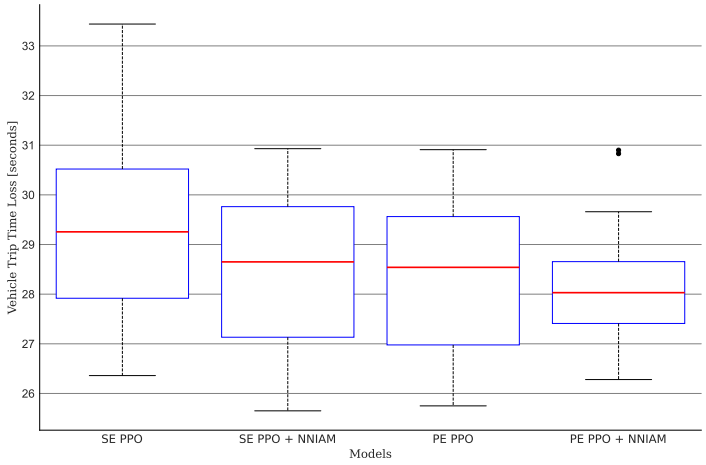


Figure 4-3: Time loss observed during testing procedure for the A-II-(1-4) models: Single environment (SE) and parallel environment (PE) PPO with and without Non-naive Invalid Action Masking. From left to right, we see can observe a slight downward trend in both median time loss and variance.

Metric	SE PPO	SE PPO+NNIAM	PE PPO	PE PPO+NNIAM
Veh. Running [#]	2.27	**2.0**	**2.0**	2.5
Speed [m/s]	6.4561	6.5272	6.5478	**6.5522**
Duration [s]	51.1839	50.3633	50.1483	**50.0406**
Time Loss [s]	29.3444	28.5283	28.3089	**28.1994**
TTT [s]	97775	96270	95827	**95609**

Table 4-9: Mean Metrics for models A-II-(1-4). The parallel training PPO with NNIAM shows superior performance in most metrics compared to other high-frequency models.

To compare the improved high-frequency sampling model to the low-frequency sampling model in terms of performance, we present the next plot. Here we compare the performance of the low frequency $\Delta t = 5$ model, the high-frequency NNIAM parallel training $\Delta t = 1$

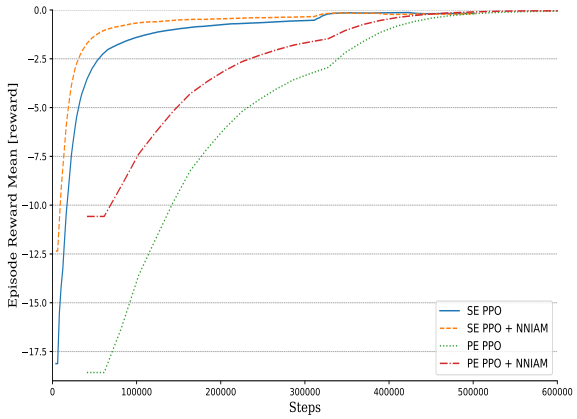


Figure 4-4: The learning curves for the A-II-(1-4) models. For smoothing, an exponential moving average was used with a smoothing factor of $\alpha = 0.3$.

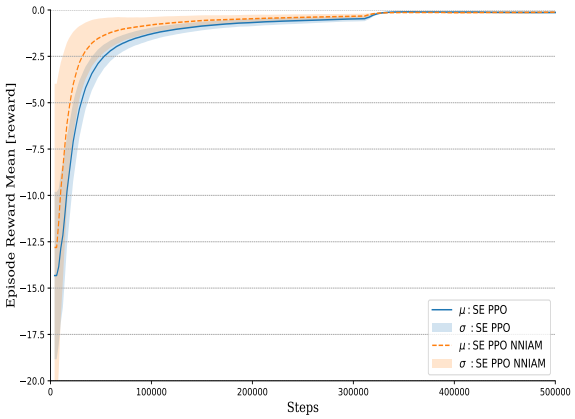


Figure 4-5: Learning curves of SE PPO (A-II-1) and SE PPO + NNIAM (A-II-2) models over training iterations. Each curve represents the mean performance across three training runs, with standard deviation and an exponential moving average with smoothing factor $\alpha = 0.3$.

model, a max-pressure and a fixed-time approach. The performance in terms of time-loss can be observed in Figure 4-6 and all the performance indicators can be seen in Table 4-10. This is the final indicator of the performance on average.

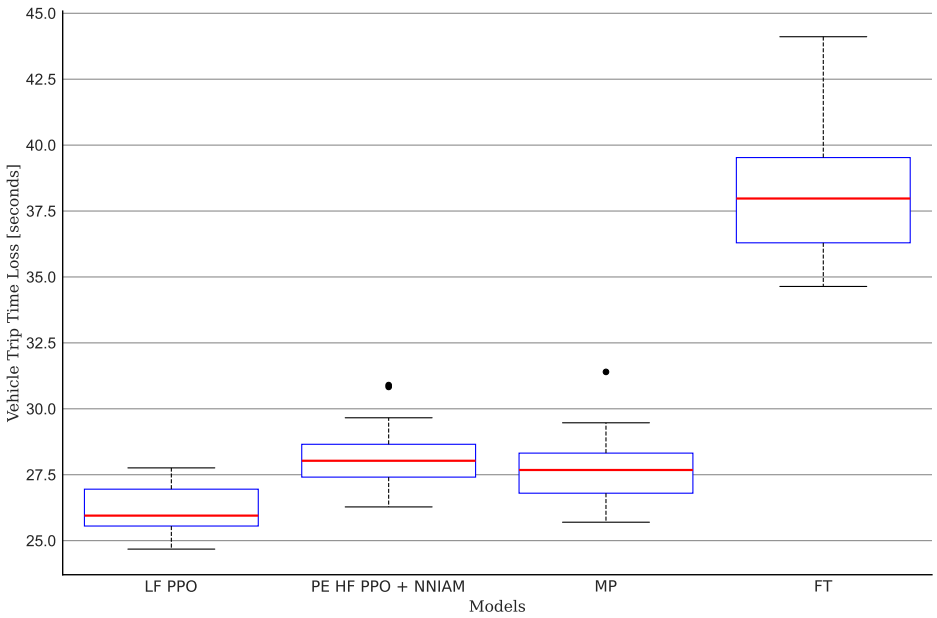


Figure 4-6: The performance in terms of time-loss for the Low Frequency Sampling PPO ($\Delta t = 5$), the Parallel Environment High Frequency Sampling ($\Delta t = 1$) PPO with Non-Naive Invalid Action Masking, Max-pressure method and fixed-time method.

Metric	LF PPO	PE HF PPO + NNIAM	MP	FT
Vehicles Running [#]	1.0	2.5	**0**	3.5
Speed [m/s]	**6.77**	6.56	6.61	5.78
Duration [s]	**47.8**	49.93	49.51	59.94
Time Loss [s]	**25.95**	28.03	27.68	37.97
Total Travel Time [s]	**91597.5**	94908.5	96181.5	112899.5

Table 4-10: Median Metrics for Different Models for Final Comparison. We can observe that the $\Delta t = 5$ PPO outperforms the other methods.

During the training phase, we can observe in Figure 4-4 that the models that we parallel trained converge slower. This is merely due to the fact that it runs 10 environments in parallel. Therefore, it takes 10 times as many steps but collects the same amount of experiences. In Figure 4-5, we can observe that a masked PPO approach converges faster than a non-masked PPO approach. However, it does also experience more variance.

As we can observe in Figure 4-6, the LF PPO performs best in terms of time loss. However, the performance of max-pressure and the HF PE PPO with NNIAM follow it closely. A fixed-time approach shows significant worse performance. This suggests that the adaptive models strongly profit from their real-time traffic data. In the next section, we will continue to see how these four methods uphold against disturbances and varying traffic volumes.

Sensitivity Analysis: Traffic Volume

In this next section, we will further evaluate and test the four final models of previous sections: We take the best performing PPO model in general (A-II-5), the best performing high-frequency PPO model (A-II-4), a max-pressure and a fixed-time approach. In this section, we will evaluate the performance in varying traffic scenarios. The flow rate in increased gradually three times, which brings us to a total of four traffic scenarios which can be observed in Table 4-12. The baseline scenario is the traffic volume on which the PPO agents are trained.

During testing, it was noted that the max-pressure approach suffered from too frequent changes of phase. Therefore, we introduce a second max-pressure approach, named Max-pressure extended. This approach maintains minimal green times (13 seconds), which improves the performance in higher traffic flow scenarios. In Table 4-11, a table is shown to give an overview of what models are being tested. These five models were tested for the four traffic scenarios.

In Figure 4-7, the performance of the four various models over the various traffic scenarios is depicted. We can observe an upward trend for all models, which is deemed logical since more traffic will induce more waiting time. For the HF PPO, we see a strong increase in time-loss, both in median and in variance. It still outperforms the HF PPO and the max-pressure approach. The max-pressure approach shows a significant increase in performance and performs worst in scenario three. This can be related to the frequent changing of phase since the max-pressure extended follows the same logic but maintains longer green times. This approach performs significantly better in scenario 2, 3 and 4. The fixed time approach has a relatively constant performance over the higher traffic volume scenarios. In Table 4-13,

Model ID	Name	Description
A-II-5	LF PPO	The best performing PPO model in general performing at $\Delta t = 5$.
A-II-4	HF PE PPO + NNIAM	A PPO model trained with parallel environments with Non-Naive Invalid Action Masking performing at $\Delta t = 1$
MP	max-pressure	A max-pressure model that adapts based on current traffic pressure.
MP Ext	max-pressure extended	A modified max-pressure model with minimal green times to reduce phase change frequency.
MP adpt.	max-pressure adaptive	A modified max-pressure which has minimal green times proportional to the traffic volumes.
FT	Fixed-Time	A fixed-time approach which cycles through the possible phases without any real-time adaptability

Table 4-11: Overview of Models used for the traffic volume sensitivity analysis of Case Study A-II.

the extended data on the performance of all model over the various performance metrics is shown.

Traffic Busyness	Flow Rate Straight [veh/h]	Flow Rate Turn [veh/h]
Baseline	300	150
Scenario 2	400	200
Scenario 3	500	250
Scenario 4	600	300

Table 4-12: Traffic flow rate scenarios for the traffic volume sensitivity analysis of Case Study A-II.

To gain more insight into the actual performance over the episode, we have plotted the traffic behavior over the 20 episodes. In Figure 4-8, we can observe the amount of vehicles in the simulation environment over every time-step for each scenario. Subfigure (a) shows the baseline scenario, while subfigure (b) displays scenario 2. Similarly, subfigures (c) and (d) illustrate the evolution of the amount of vehicles for scenarios 3 and 4, respectively. These figures show that in the baseline scenario, all methods are able to maintain a constant amount of vehicles in simulation and empty the simulation as soon as the traffic flows stop. The fixed-time approach has most trouble and seems to increase slightly over time, in mean and in variance. As we increase the volume to scenerio 2, we can observe that the amount of vehicles in simulation continues to rise over the episode. The max-pressure extended methods strongly outperforms the other methods. In scenario 3 and 4 all models suffer from the high traffic volumes.

In Figure A-23, Figure A-28 and Figure A-33, the amount of vehicles halting in the simulation, the average driving speed and the average travel time over the testing episodes is displayed. These metrics mainly display the same trends that the time-loss plots over the varying scenarios show us. High mean speed correlates with low time-loss which correlates with little vehicles halting which correlates with a low mean travel time.

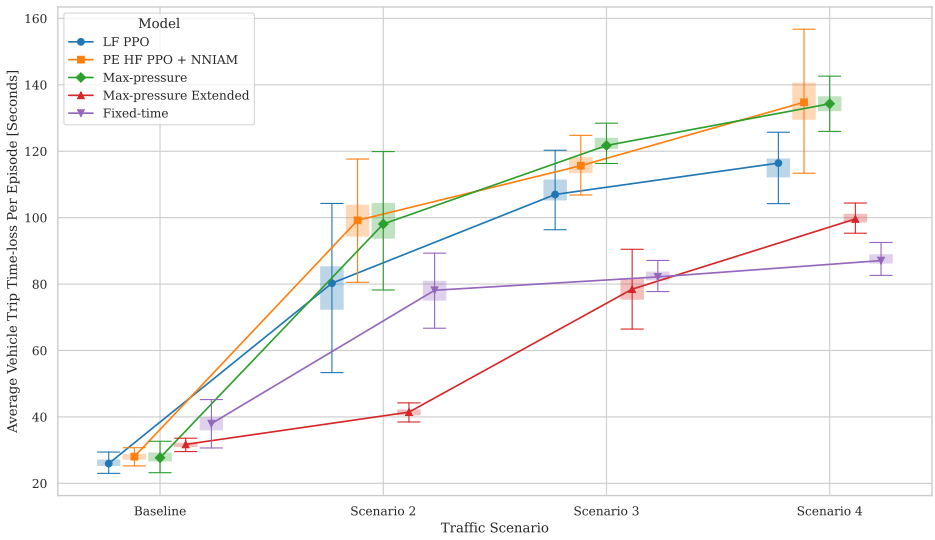
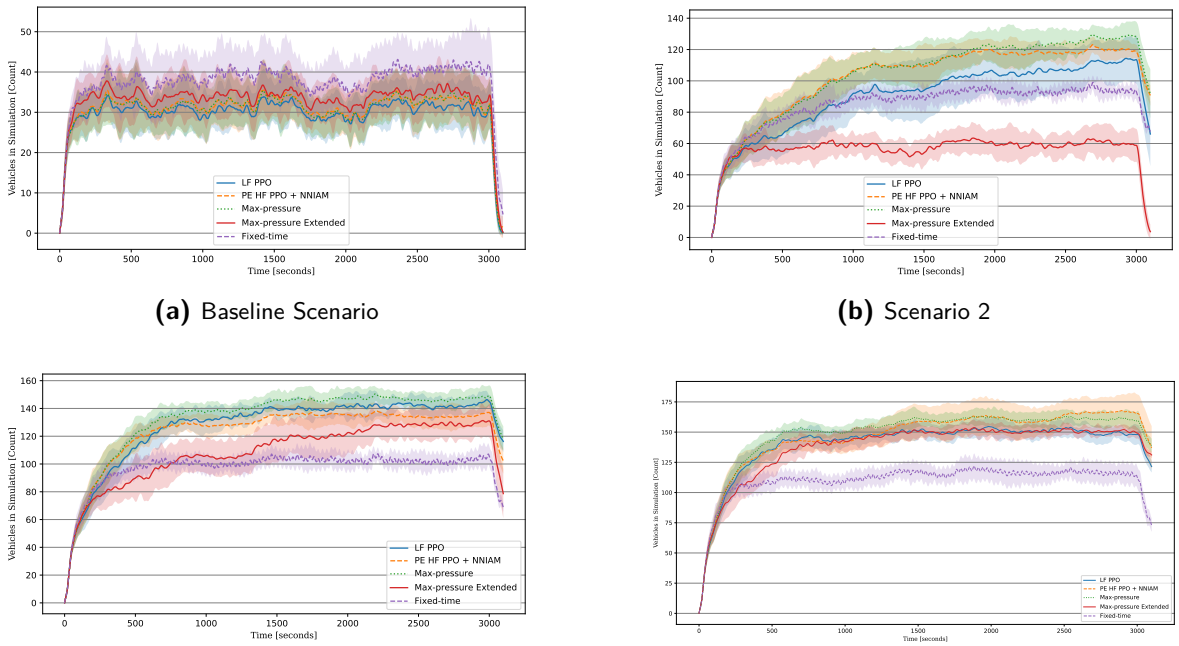


Figure 4-7: Performance of five models across the traffic scenarios defined in Table 4-12. All models exhibit an increasing trend. The LF PPO model's performance decreases significantly in busier scenarios but still surpasses the HF PPO. The max-pressure approach performs poorly, while its extended version improves notably in scenario 2. The fixed-time approach maintains relatively constant performance in higher-volume scenarios.



(a) Baseline Scenario

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Figure 4-8: The amount of vehicles in simulation over time for the baseline scenario (a), Scenario 2 (b), Scenario 3 (c), and Scenario 4 (d).

The sensitivity analysis gives great insight into the performance of the models over constant higher traffic volumes. Figure 4-8a shows us that all five approaches can maintain a steady-

Scn.	Metric	LF PPO	HF PPO	MP	MP ext.	FT
Baseline	**Veh. Running** [#]	1.0	2.5	**0**	**0**	3.5
	Speed [m/s]	**6.775**	6.565	6.615	6.415	5.78
	Duration [s]	**47.8**	49.93	49.515	53.58	59.945
	Time Loss [s]	**25.95**	28.03	27.68	31.725	37.975
	TTT [s]	**91597.5**	94908.5	96181.5	101932.0	112899.5
Scenario 2	**Veh. Running** [#]	63.5	90.0	94.5	**2.5**	70.0
	Speed [m/s]	4.14	3.75	3.67	**5.535**	4.35
	Duration [s]	102.085	121.055	120.0	**63.305**	99.93
	Time Loss [s]	80.26	99.185	98.08	**41.43**	78.135
	TTT [s]	274788.5	310087.0	312843.0	**173057.5**	257608.0
Scenario 3	**Veh. Running** [#]	115.5	102.0	117.5	76.0	**70.0**
	Speed [m/s]	3.54	3.365	3.215	3.985	**4.12**
	Duration [s]	128.815	137.515	143.505	**100.295**	104.0
	Time Loss [s]	106.98	115.645	121.725	**78.47**	82.15
	TTT [s]	379528.5	373167.0	397876.5	327074.5	**295065.0**
Scenario 4	**Veh. Running** [#]	122.0	130.5	138.0	130.0	**72.0**
	Speed [m/s]	3.3	2.825	2.855	3.51	**3.78**
	Duration [s]	138.24	156.54	156.16	121.45	**108.875**
	Time Loss [s]	116.425	134.745	134.295	99.61	**87.07**
	TTT [s]	420727.5	437062.5	444485.5	418333.5	**334756.5**

Table 4-13: Median Metrics for the five models described in Table 4-11 on the varying traffic volume scenarios. Considering the five performance metrics, we can see a significant shift towards max-pressure extended and finally a fixed-time approach.

state in the baseline traffic scenario. However, once we move to scenario 2, most models experience a constant increase in amount of vehicles in simulation. In Figure 4-8b, we can observe that only the extended max-pressure can effectively keep the amount of vehicles in simulation low and in steady-state. Fixed-time also maintains steady-state but at a much higher average. So, this suggests that the PPO models trained on baseline traffic conditions do not generalize well to busier traffic scenarios. Therefore, we aimed to develop a Proximal Policy Optimization approach capable of performing effectively across diverse traffic scenarios in the next section.

Training for Better Generalization

The sensitivity analysis on traffic volumes revealed that PPO models trained on specific traffic conditions struggle to generalize when exposed to different traffic volumes. This limitation poses a significant challenge for real-world traffic management systems, where traffic patterns can fluctuate widely. To address this issue, we explored whether training models on varying traffic volumes would improve their generalization capabilities.

To address the challenge of generalization in our traffic management models, we trained a total of five new Proximal Policy Optimization (PPO) models. Three of these models—designated as A-II-19, A-II-20, and A-II-21—were trained on progressively higher fixed traffic volumes to assess performance under increased load conditions.

In addition, we developed two models, A-II-22 and A-II-23, trained on dynamically varying traffic volumes over the training time steps. Model A-II-22 was trained using traffic volumes

sampled from a Gaussian distribution, as detailed in Table 4-14. Model A-II-23, on the other hand, was trained on traffic volumes randomly sampled from a predefined range specified in the same table.

For both A-II-22 and A-II-23, the intervals at which traffic volumes switch were also randomized. The timing of these switches was sampled from a uniform distribution within a range of 200 to 400 seconds. Importantly, these volume changes occurred independently for each lane, introducing asynchronous variability across the network. For example, lane 1 might experience a traffic volume change from 160 vehicles per hour to 290 vehicles per hour at time step 230, while lane 2 might switch from 550 to 420 vehicles per hour at time step 390. This methodology ensures significant variability in traffic patterns across all directions, improving the models' ability to generalize to fluctuating traffic conditions.

Model ID	**Straight [veh/h]**	**Turn [veh/h]**	**Scenario**	**Avg. throughput [veh/h]**
A-II-5	300	150	Baseline	1800
A-II-19	400	200	Scenario 2	2400
A-II-20	500	250	Scenario 3	3000
A-II-21	600	300	Scenario 4	3600
A-II-22	$\mu = 450, \sigma = 75$	$\mu = 225, \sigma = 37$	Scenario Gaussian	2700
A-II-23	[300, 600]	[150, 300]	Scenario random	2700

Table 4-14: Overview of PPO models trained on different traffic volumes. Each model is identified by its ID and the traffic scenario it was trained on.

To evaluate the generalization performance of the newly trained models, we assessed each one across all four predefined traffic volume scenarios. As the main performance indicator, we evaluate average time loss per episode over the testing episodes.

In Figure 4-9, we present the performance of the models trained on constant traffic volumes; A-II-5, 19, 20, 21. The testing procedure for each model was also done on each traffic volume. We can observe the following; Training on the baseline traffic scenario does not generalize well to busier traffic scenarios. Also, training on either scenario 2 to 4 does perform substantially better on scenarios 2 to 4. However, the model trained on the baseline scenario still outperforms the other models on the baseline tests. This suggests that the traffic flow dynamics of scenario 2 to 4 behave slightly similar as the models perform equally well over these scenarios.

In Figure 4-10, we include the performance of models A-II-22 and A-II-23. Note that in this plot, we have tested four models: A-II-19, A-II-22 and A-II-23 and the adaptive max-pressure approach. The adaptive max-pressure approach adjusts its minimal green times dynamically, maintaining longer durations as traffic volume increases to optimize flow. The primary objective of incorporating this adaptive functionality is to assess how effectively a control policy can perform under varying traffic conditions. In Figure 4-10, we tested the models on all scenarios as defined in Table 4-14.

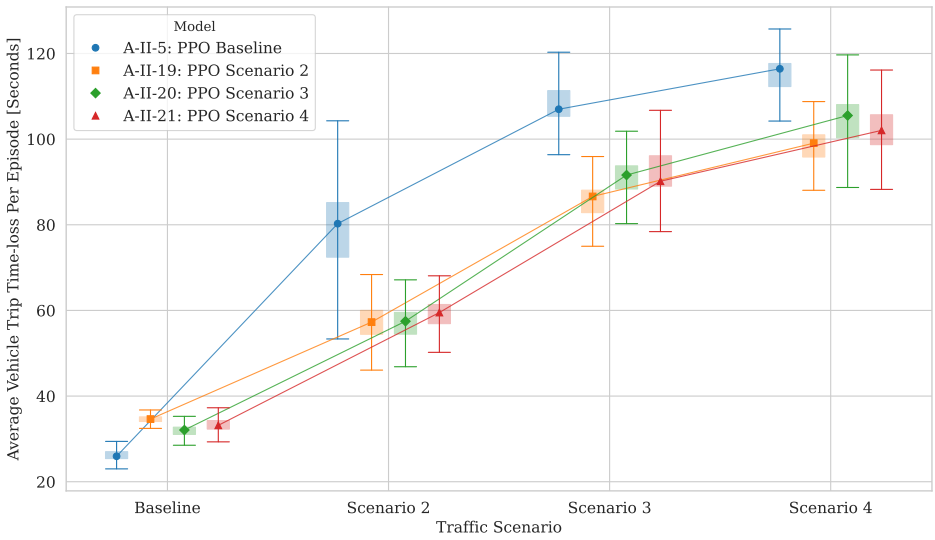


Figure 4-9: Performance comparison of the PPO models (A-II-5, 19, 20, 21) each trained on the traffic scenarios specified in Table 4-14. Each model's average time-loss per vehicle is plotted for all scenarios to evaluate generalization.

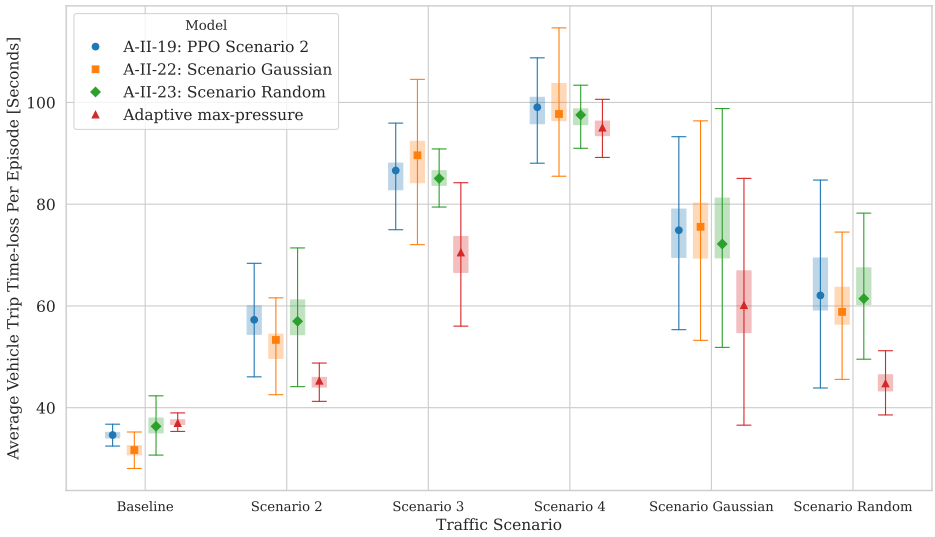


Figure 4-10: Performance comparison of PPO models A-II-19, A-II-22 and A-II-23, along with the adaptive max-pressure approach. We can observe a strong performance of the adpt. max-pressure in all scenarios as well as high variance in performance over all scenarios.

In Figure 4-10, we can observe two more models trained on Gaussian and randomly sampled traffic as explained previously. We can observe high variance in the testing procedure of the Gaussian scenario. The large data spread can be explained by the Gaussian distribution, which has no upper limit and can generate very high traffic volumes. These high volumes can cause severe congestion, which is hard to resolve once it starts. This could lead to episodes with much higher time-loss, increasing the variation in the results.

Also, it is notable that the adaptive max-pressure approach outperforms all other PPO ap-

proaches except for the baseline scenario. Also, training on randomly sampled traffic (A-II-23) seems to perform better that models trained on Gaussian sampled traffic volumes (A-II-22). The strong performance of the adaptive max-pressure approach suggests that maintaining a phase longer and switching less frequency leads to better performance in terms of time loss.

Sensitivity Analysis: Disturbance Rejection

In this section, we will test four models with disturbance rejection: The LF PPO, HF PE PPO + NNIAM, the max-pressure and the fixed time approach as described in Table 4-11. To introduce disturbances and assess the robustness of the control strategies, we increase the flow rates threefold for a duration of 300 seconds during the simulation. The flow rates are shown in Table 4-15. This increased traffic flow will come from four directions: North to South, North to West, South to North and South to East. This temporary surge in traffic demand is designed to test the system's ability to handle sudden changes in traffic volume. The flow rates and the corresponding vehicle insertion times into the simulation are illustrated in Figure 4-11.

Lane Type	Flow Rate Baseline [veh/h]	Flow Rate Disturbance [veh/h]
Straight-going lanes	300	900
Turning lanes	150	450

Table 4-15: Traffic flow rates in vehicles per hour. The baseline traffic flow and a disturbance flow rate.

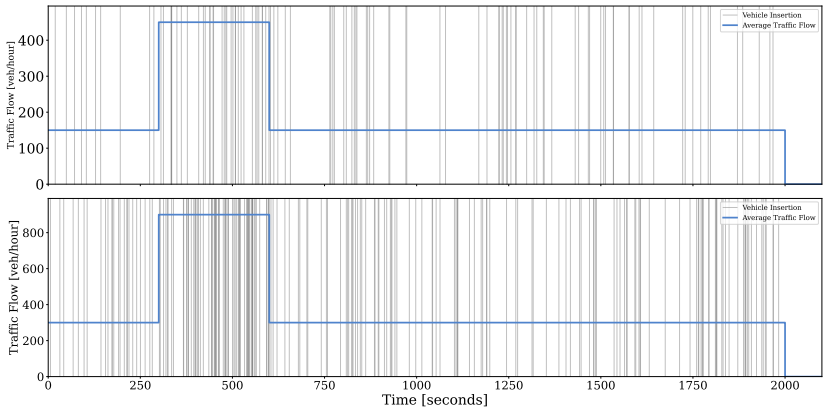


Figure 4-11: Traffic input flow rates (blue) and vehicle insertion times (gray lines) into the simulation, where inter-arrival times are sampled from an exponential distribution. The upper plot represents a turning lane, while the lower plot represents a straight lane.

In Figure 4-12, the amount of vehicles in simulation over the test episodes is plotted. It displays a mean with a moving average over 20 time steps and a standard deviation. In Figure A-38, other performance metrics are plotted. We can observe the sudden surge of vehicles in simulation for all models around 300 seconds. At 600 seconds, the increased volume returns to the baseline volume and we see a decrease of vehicles in simulation for all four models. The LF PPO returns to a steady state value for amount of vehicles in simulation fastest. Thereafter, the HF PPO and max-pressure, which display very similar performance.

A fixed-time approach does not seem to handle these sudden spikes in traffic properly and cannot return to steady state value. To show the percentiles of the performance, in Figure 4-13 the 25-75, 10-90 and 5-95 percentiles of the time-loss are shown for all four models. In these plots, we observe the lowest spread in performance for the fixed-time approach. However, this comes with the worst performance in terms of returning to steady state. The HF PPO and max-pressure have the biggest spread in data.

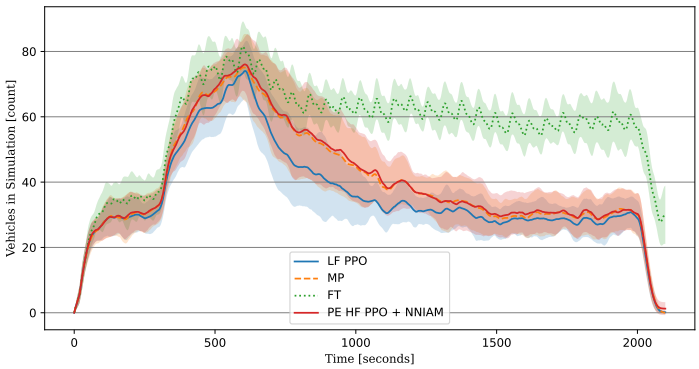
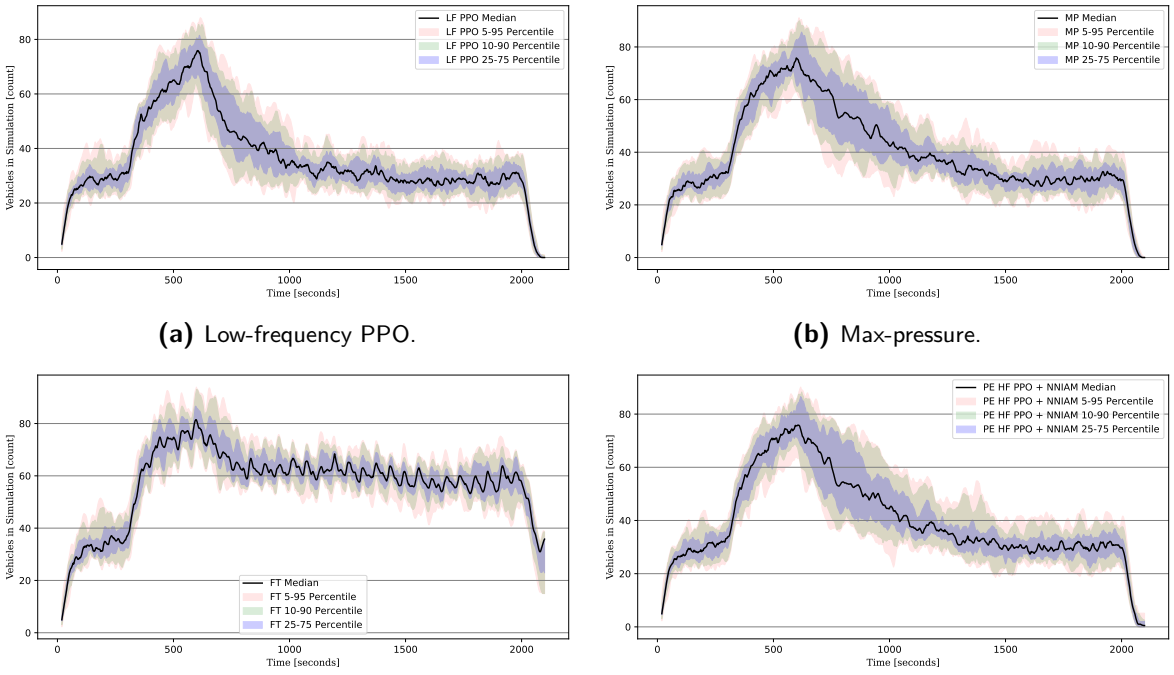


Figure 4-12: Disturbance rejection over time for LF PPO, HF PE PPO + NNIAM, max-pressure, and fixed-time approaches, showcasing the number of vehicles in simulation across test episodes. The figure shows the mean with a 20-step moving average and standard deviation. We observe a surge in vehicle volume around 300 seconds, with LF PPO returning to steady-state fastest, while the fixed-time approach struggles to manage it.



(a) Low-frequency PPO.

(b) Max-pressure.

(c) Fixed-time.

(d) High-Frequency PE PPO + NNIAM.

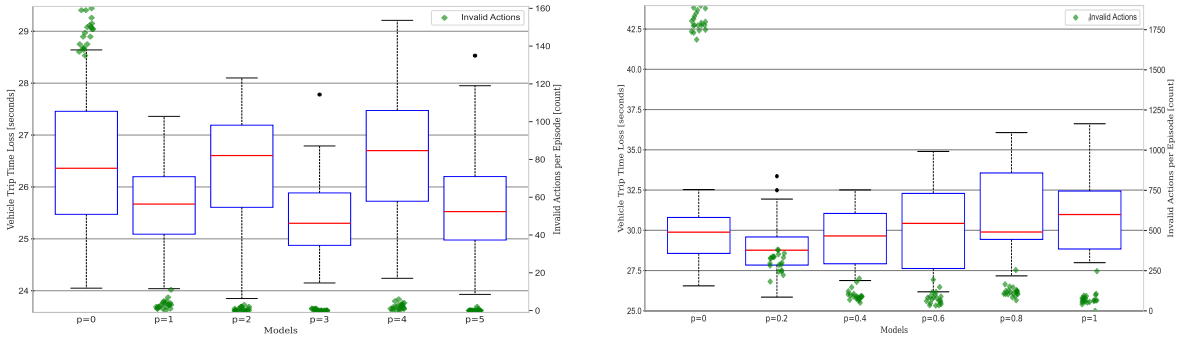
Figure 4-13: Percentile analysis of time-loss performance for four traffic control models: LF PPO, max-pressure, fixed-time, and HF PE PPO + NNIAM. Each sub-figure displays the 25-75, 10-90, and 5-95 percentile ranges.

An interesting result is that similarity for the HF PPO and the max-pressure approach. As we can observe in Figure 4-12, the amount of vehicles running in simulation are oddly similar. We currently lack data to support a specific explanation for this behavior. The strong performance of an LF PPO against the fixed-time approach is also promising; This could indicate that the PPO approach has learned behavior to from the data which can solve big traffic congestion. This is one of the key elements of the first research sub-question.

Effectiveness Invalid Action Approaches

To evaluate the effectiveness of Non-naive Invalid Action Masking in handling invalid actions, we present the following results. We consider three approaches: a PPO model with action overrule without any penalty on invalid actions, PPO models with various penalties on invalid actions (models A-II-9 to A-II-18 in Table 4-8), and the Non-naive Invalid Action Masking approach. We evaluate these models at both high and low frequencies, operating at $\Delta t = 1$ and $\Delta t = 5$.

In Figure 4-14, we present the performance of the 10 penalized models during training, along with the number of invalid actions chosen per episode plotted on a secondary y-axis. This analysis is conducted for both high-frequency and low-frequency PPO models. For the low-frequency PPO, there is no observable trend in performance across varying penalty values, although a high variance in performance is relatively high. Regardless of the penalty magnitude, there is a significant decrease in the number of invalid actions chosen. The high-frequency PPO at $\Delta t = 1$ shows a slight increase in time-loss as penalties increase. The number of invalid actions chosen significantly decreases with higher penalties as well. This suggests that it is possible to substantially reduce invalid actions per episode without adversely affecting performance.



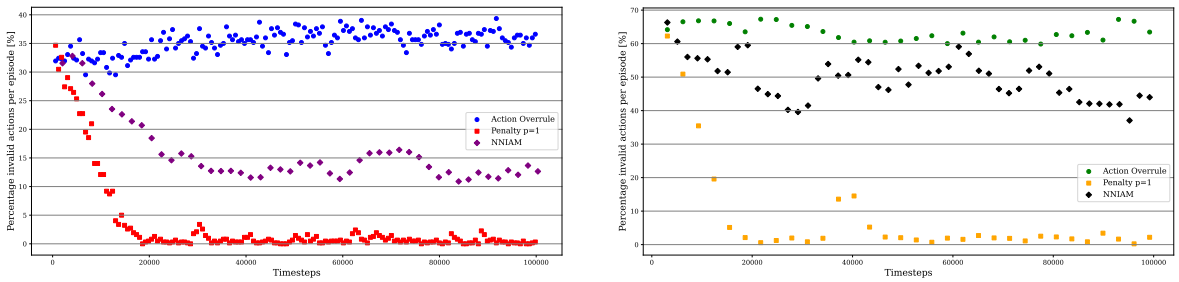
(a) Time-loss and amount of invalid actions at $\Delta t = 5$.

(b) Time-loss and amount of invalid actions at $\Delta t = 1$.

Figure 4-14: Performance comparison of 10 penalized models during training at different Δt (5 and 1). Both low- and high-frequency models do not show a clear trend in performance and keep within performance bounds. However, both models are effective at minimizing invalid actions chosen per episode, with the low-frequency PPO ($\Delta t = 5$) reducing it to close to 0 and the high-frequency PPO ($\Delta t = 1$) pushing it down to 100 to 200 invalid actions per episode. In Figure A-42, the figures are shown magnified.

To evaluate the effectiveness of NNIAM compared to penalizing methods in handling invalid actions, we present Figure 4-15. These plots show the percentage of invalid actions chosen per

episode over the training time steps for both $\Delta t = 5$ and $\Delta t = 1$ models. Initially, all models exhibit similar percentages of invalid actions. In the non-penalized model, this percentage does not decrease over time and even slightly increases for $\Delta t = 5$. In contrast, the penalized model quickly reduces the percentage of invalid actions to near zero for both operating frequencies. For the Non-naive Invalid Action Masking approach, the agent continues to select invalid actions despite continuous masking, with a more pronounced decrease observed in the $\Delta t = 5$ model compared to the $\Delta t = 1$ model. Further discussion on this behavior is provided in chapter 5.



(a) Invalid actions per episode over training time at $\Delta t = 5$.

(b) Invalid actions per episode over training time at $\Delta t = 1$.

Figure 4-15: Percentage of invalid actions chosen per episode over training time for different models at different Δt. The penalized models rapidly reduce invalid actions to near zero, while the NNIAM models show a gradual decrease, more pronounced at $\Delta t = 5$. Non-penalized models do not decrease invalid actions over time and may even exhibit a slight increase. In Figure A-44, a magnified version of the plots are shown.

4-3-3 Conclusions

In this case study, we investigated the performance of high-frequency sampling Proximal Policy Optimization models. We presented improvement efforts such as Non-naive Invalid Action Masking and Parallel Training and evaluated these against the low-frequency sampling Proximal Policy Optimization model, a max-pressure and a fixed-time approach. Thereafter, we conducted sensitivity analysis for both varying traffic volumes and sudden traffic disturbances. Finally, we conducted experiments to gain insights into methods to manage invalid actions during training and testing. In the following sub-sections, we will draw conclusions on each of the five sub-sections of this case study.

Conclusions Non-naive Invalid Action Masking and Parallel Training

From Case Study A-I, we concluded that the high-frequency sampling approach did not produce the performance as expected in the hypothesis. Therefore, we made several improvement efforts. One of which is Non-naive Invalid Action Masking and the other is training on multiple environments in parallel. From Figure 4-3, Table 4-9, we can observe gradual improvements in performance with the proposed improvements. In terms of time-loss seconds during the testing phase, we observe a 4 % decrease in mean time and a 36 % decrease in standard deviation. This improvement is small can can be attributed to randomness. Figure 4-5 also shows that methods with Non-naive Invalid Action Masking converge faster. We can observe a higher average reward per episode for all episodes in the training phase.

Thereafter, this model was compared to the low-frequency sampling model, a max-pressure and fixed-time approach. In Figure 4-6, we can observe that a low-frequency sampling model still outperforms the high-frequency model in terms of time-loss tested against the baseline traffic scenario.

Conclusions Sensitivity Analysis: Varying Volume

In conclusion, the sensitivity analysis demonstrated that all models experienced increased time-loss as traffic volumes rose, which is expected due to higher demand. The Low-Frequency sampling PPO model has a significant drop in performance: As the traffic volumes are increased by 33 %, the time-loss mean increases by roughly 300 % and the standard deviation by 850 %. This suggests ineffective generalization over other traffic volumes. The performance continues to decrease as the traffic volumes get increased, but more gradual than the first jump. The high-frequency sampling PPO with NNIAM performed similar, however, the Low-frequency sampling PPO model did consistently outperform the High-Frequency PPO model.

The standard Max-Pressure approach performed poorly in higher traffic scenarios. This could be due excessive phase changes as the Max-Pressure Extended model, which introduced minimal green times to reduce phase change frequency, showed substantial performance improvements in scenarios with increased traffic flow. The Fixed-Time approach maintained relatively constant performance across higher-volume scenarios and even outperformed other models in certain metrics under the highest traffic conditions. The fixed-time approach maintained a relatively constant mean and variance over the higher traffic volumes.

These results suggest that when training on PPO models on certain traffic scenarios with these configurations, they do not perform well in higher traffic volumes scenarios. The max-pressure and max-pressure extended results also suggest that maintaining longer green light times improves performance at higher traffic volume scenarios.

Conclusions generalization

The experiments reveal that PPO models trained on specific traffic volumes have limited generalization capabilities when faced with different traffic conditions, which is a significant challenge for real-world traffic management systems with fluctuating patterns. Training models on higher traffic volumes improves performance in similarly high-volume scenarios but fails to generalize back to the baseline scenario—where the baseline-trained model (A-II-5) still

outperforms others. Aside from generalization, PPO models are outperformed by approaches such as extended max-pressure and fixed time in heavier traffic volume scenarios.

Models trained on varying traffic volumes over time, particularly the one trained on randomly sampled volumes (A-II-23), demonstrate slightly generalization than the model trained on Gaussian-sampled volumes (A-II-22). However, the adaptive max-pressure approach consistently outperforms the PPO models across most scenarios except the baseline. These findings suggest PPO models still struggle to match the adaptability of traditional control methods like adaptive max-pressure in dynamic traffic environments.

Conclusions Sensitivity Analysis: Disturbance Rejection

In the disturbance rejection sensitivity analysis the models were subjected to a threefold increase in vehicle arrival rates for 300 seconds. All models experienced a significant rise in the number of vehicles within the simulation. The low-frequency sampling PPO best managed to bring down the spike in vehicles in simulation to a steady state in roughly 550 seconds, outperforming the high-frequency sampling PPO, max-pressure and fixed-time approach. The high-frequency PPO and max-pressure model exhibit significantly similar behavior, both returning to steady state after roughly 800 seconds. The fixed-time approach does not aquire a low steady-state value in terms of vehicles in simulation after the disturbance. It maintains a high steady-state of roughly 60 vehicles in simulation. The steady-state of vehicles in simulation of the low-frequency PPO is around 30 vehicles, whilst the high-frequency sampling PPO and max-pressure are slightly above.

This suggests that while the fixed-time approach performed best at constant high volumes, it struggles with disturbance rejection and maintaining steady state. Something the high- and low-frequency sampling ppo and max-pressure are able to do. The percentile analysis highlighted that while the fixed-time approach had the lowest variability in performance metrics, it also had the poorest ability to manage the increased traffic demand. These findings suggest that the low-frequency sampling PPO model is more effective in rejecting disturbances and maintaining traffic flow stability, making it possibly a more robust choice for scenarios with sudden changes in traffic volume.

Conclusions methods handling invalid actions

The evaluation demonstrates that penalizing invalid actions in PPO models significantly reduces their occurrence without adversely affecting overall performance. At both low frequency ($\Delta t = 5$) and high frequency ($\Delta t = 1$), introducing penalties leads to a rapid decrease in invalid actions chosen by the agents. For low-frequency sampling models, the amount of invalid actions per episode can be brought to close to 0, whilst for high-frequency sampling models, it remains around 100 invalid actions every 3000 decisions. For low- and high-frequency sampling PPO models, the results suggest that penalizing invalid actions does not effect performance and reduces invalid actions strongly. In contrast, the Non-naive Invalid Action Masking (NNIAM) approach is less effective in decreasing invalid actions over time. As we have learned from Figure 4-15, the amount of invalid actions chosen by the agent at $\Delta t = 5$ and $\Delta t = 1$ when no invalid action strategy is employed is roughly 30% to roughly 70 %. After training with Non-naive Invalid Action Masking, this reduces to 10% and 40%, respectively. This indicates that penalty-based methods are more effective than action masking in guiding agents away from invalid choices during training.

4-4 Case Study B: Prioritization of Traffic Modes

In this final Case Study B, pedestrian traffic flows are introduced to the Reinforcement Learning framework. This introduces an extra layer of complexity. We aim to answer the second sub research question: Can a Deep Reinforcement Learning Agent effectively balance prioritization between pedestrians and vehicles in terms of time-loss? To do this, we constructed the pedestrian included framework as described in subsection 3-2-5. Compared to the previous frameworks, the observation space, action space and reward function get extended to include the pedestrian traffic flows. Also, we have applied the Non-naive Invalid Action Masking mechanic to this framework.

In this section, we will elaborate shortly on various design choices and present the hyper parameters and various models trained. In subsection 4-4-1, we will present the results from the experiment. Finally, in subsection 4-4-2, we will draw conclusions from the conducted experiments and gathered results.

For the pedestrian framework, the environment changes in comparison to the vehicle only environment. In simulation, pedestrians behave as small vehicles. The differences are the moving speed and acceleration speed, which are significantly lower. Also obviously, pedestrian traffic flows move over pedestrian roads, which are located on the outside of the vehicle roads.

In Figure 3-6a, we can observe the difference in framework. As for the observation space, it gets extended with the three observations as mentioned in subsection 3-2-5: The minimal green pedestrian timer, the walking area densities and the crossing densities. The action space also gets extended to the 6-phase configuration. In Table 4-17, an overview of the environment parameters can be seen. In Table 4-16, an overview of the parameters for the PPO can be seen.

Parameter	**Symbol**	**Value**
Policy	-	MlpPolicy
Learning Rate	α	0.001
Number of Steps	n_{steps}	2048
Mini-Batch Size	$n_{\text{mini-batch}}$	64
Number of Epochs	n_{epochs}	10
Gamma	γ	0.99
GAE Lambda	λ	0.95
Clip Range	ϵ	0.2
Value Function Coefficient	c_{vf}	0.5
Entropy Bonus Coefficient	c_{ent}	0
Max Gradient Norm	-	0.5
Network Architecture	-	[128, 256, 128]
Training time-steps	T_{total}	500000

Table 4-16: Proximal Policy Optimization hyper parameters for Case Study B.

Parameter	Value
Episode Length	3000 seconds
Minimum Green Time	7 seconds
Minimum Green Time Pedestrians	17 seconds
Yellow Time	3 seconds
Delta Time	5 seconds
Observation Space	CF, MGT, QL, LD, MGT, MGTP, WA, CD
Action Space Configuration	6-phase**
Reward Function	Equation 3-5
Vehicle Traffic Flow Straight	300 veh/hour
Vehicle Traffic Flow Turns	150 veh/hour
Pedestrian Traffic Flow Straight	300 ped/hour
Pedestrian Traffic Flow Turns	150 ped/hour

Table 4-17: Selected Environment and DRL Parameters. *The Observation space abbreviations can be found in section 3-2-1. **The exact action space can be found in subsection 3-1-1.

We take a step back from the the high-frequency approach and use the best performing PPO model with the pedestrian parameters as described above in Table 4-17. The framework is described in subsection 3-2-5. The used reward function to balance vehicle and pedestrian prioritization is defined as:

$$r_t = w_v \times \Delta D_{t,\text{veh}} + w_p \times \Delta D_{t,\text{ped}} \tag{4-1}$$

Again, here w_v and w_p are the weights for vehicles and pedestrians, respectively. In Table 4-18, an overview of the trained models for Case Study B can be observed, including their respective weights.

Model ID	Weight w_v	Weight w_p	NNIAM
B-1	10	1	no
B-2	5	1	no
B-3	3	1	no
B-4	2	1	no
B-5	1	1	no
B-6	1	2	no
B-7	1	3	no
B-8	1	4	no
B-9	1	5	no
B-10	1	10	no
B-11	1	15	no
B-12	1	20	no
B-13	10	1	yes
B-14	1	1	yes
B-15	1	10	yes

Table 4-18: Overview of Trained Models for Case Study B.

4-4-1 Results

In this section, we present the results of the experiment. In Figure 4-16, we observe the total time loss over all vehicles and pedestrians during the testing procedure. It is evident that to minimize the total time loss for both parties, a balanced weighting ratio between w_v and w_p is most effective. To determine whether pedestrians and vehicles can be prioritized, we examine Figure 4-17. In this plot, the waiting times for both traffic groups are displayed per model. As the balance between the weights shifts towards either w_v or w_p, there is a clear trend in time loss. The vehicle time loss shows more variability in the data. Additionally, as the weights become larger for one mode, the lesser-weighted traffic mode experiences increased neglect.

As shown in Figure A-48, the number of vehicles that do not complete their trip rises significantly in models B-10, B-11, and B-12, which correspond to vehicle-pedestrian weighting ratios of (1, 10), (1, 15), and (1, 20). Conversely, pedestrian time loss experiences a sharp increase in models B-1 and B-2. In Table 4-19, the mean and standard deviation per model for each traffic mode are presented. This data also indicates a clear trend, with model B-1 prioritizing vehicles the most—having the lowest average vehicle time loss and the lowest standard deviation—and model B-12 prioritizing pedestrians the most, having the lowest average pedestrian time loss.

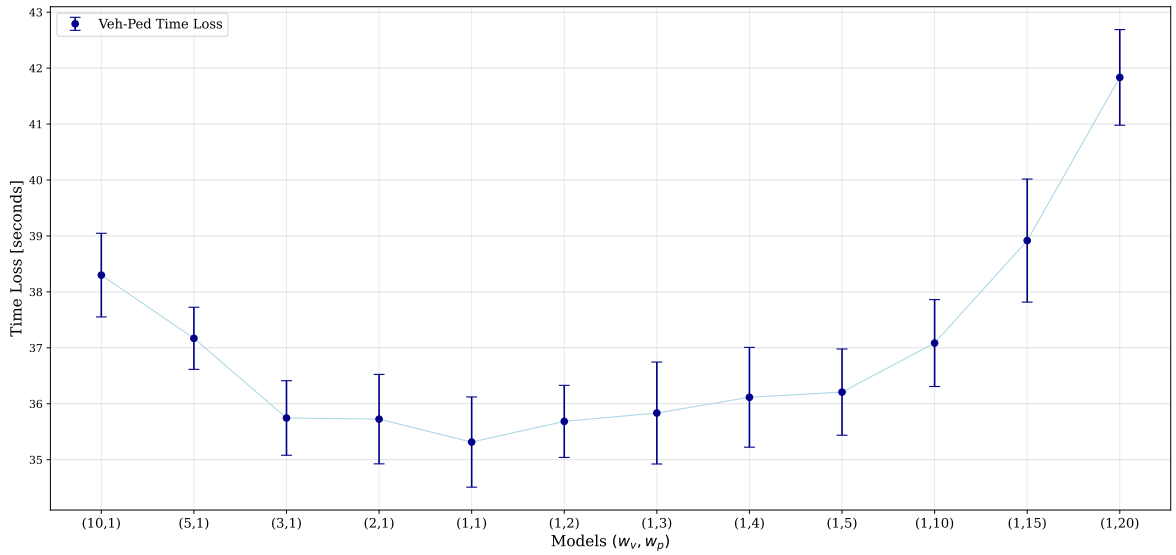


Figure 4-16: Total time loss for vehicles and pedestrians across different weighting ratios with mean and variance. The minimal total time loss occurs when the weighting ratio between w_v and w_p is balanced.

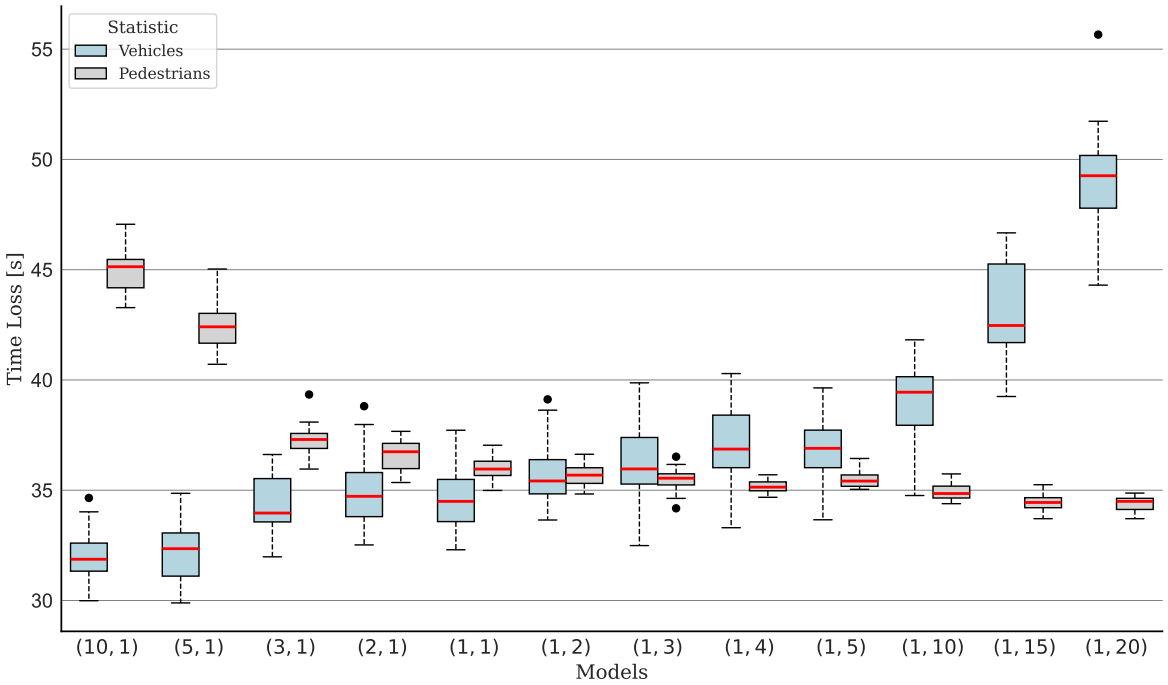


Figure 4-17: Time loss for vehicles and pedestrians per model with varying weighting ratios. As the weighting shifts towards one traffic mode, the time loss for the other mode increases, indicating prioritization.

Model ID	(w_v, w_p)	μ Time-Loss veh.	σ veh.	μ Time-Loss ped.	σ ped.
B-1	(10, 1)	**31.93**	1.03	44.95	0.79
B-2	(5, 1)	32.25	1.19	42.36	0.86
B-3	(3, 1)	34.32	1.11	37.25	0.48
B-4	(2, 1)	34.89	1.34	36.59	0.58
B-5	(1, 1)	34.67	1.46	35.94	0.46
B-6	(1, 2)	35.68	1.26	35.65	0.38
B-7	(1, 3)	36.20	1.60	35.48	0.39
B-8	(1, 4)	37.03	1.59	35.17	0.26
B-9	(1, 5)	36.94	1.35	35.46	0.27
B-10	(1, 10)	39.14	1.38	34.94	0.41
B-11	(1, 15)	43.18	1.98	34.47	0.27
B-12	(1, 20)	48.99	1.60	**34.36**	0.30

Table 4-19: Summary of mean and standard deviation of time loss for vehicles and pedestrians for each model. The lowest mean time-loss values are highlighted in bold.

In the following, we will show the results of using Non-naive Invalid Action Masking in the pedestrian framework against an Action Overrule method. In this scenario, the masking function has more functionality since there are more scenarios in which there are multiple actions possible after the masking. In Figure 4-18, we can observe the performance in terms of time-loss, split in the vehicle- and pedestrian time-loss. For the models with weights $(10, 1)$, $(1, 1)$ and $(1, 10)$, a masked version was trained. These correspond to models B-13, B-14 and B-15 in Table 4-18.

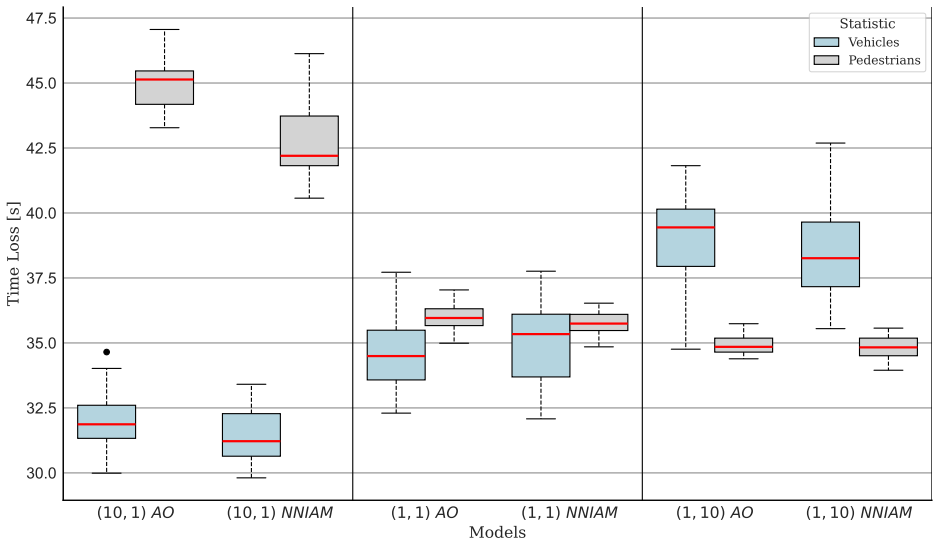


Figure 4-18: The performance in terms of time-loss of Action Overrule (AO) against NNIAM. The methods are compared at three different weight scenarios: (10, 1), (1, 1) and (1, 10). We can observe similar performance for masked and unmasked models.

In terms of performance, we can observe roughly similar performance for the AO and NNIAM methods. This suggests that NNIAM can be incorporated without effecting performance.

4-4-2 Conclusion

This case study is dedicated to managing The data demonstrates that prioritization between vehicles and pedestrians can be effectively managed by adjusting the weighting ratios w_v and w_p in the deep reinforcement learning model. As evidenced by the plots in Figure 4-17, shifting the weights towards one traffic mode increases the time loss for the other, confirming successful prioritization. However, an optimal balance is achieved at the (1, 1) weighting ratio, where both vehicles and pedestrians experience moderate time loss. Also, NNIAM can effectively be applied in a pedestrian framework and shows similar performance compared to a non-masked alternative in terms of time-loss.

This case study suggests that deep reinforcement learning can be utilized to prioritize certain traffic modes during specific times, enhancing traffic management strategies through flexible and adaptive control.

Chapter 5

Conclusions and Discussion

In this thesis, we investigated the application of Deep Reinforcement Learning, specifically Proximal Policy Optimization with Invalid Action Masking, in multi-modal Traffic Signal Control environments. By integrating TSC, we addressed the complexities associated with minimal signal timing constraints, ensuring safety and compliance with traffic regulations at intersections.

We analyzed the impact of high-frequency sampling intervals, particularly a one-second interval ($\Delta t = 1$), on the performance of PPO in various traffic scenarios. Our approach included sensitivity analyses on traffic volumes, assessments of generalization capabilities, disturbance rejection tests, and comparisons of methods for handling invalid actions. Through reward shaping, we demonstrated effective prioritization of different traffic modalities, providing a means for decision-makers to balance time-loss between vehicles and pedestrians.

In this concluding chapter, we consolidate the findings of our research by addressing the main and sub-research questions, discussing the implications of our results, acknowledging the limitations of our study, and proposing future directions for advancing DRL applications in real-world TSC systems.

5-1 Conclusions

The main research question of this thesis was:

What are challenges in applying Deep Reinforcement Learning to real-world traffic signal control, and how can these be addressed?

To properly answer this extensive research question, we presented two research sub-questions. Below, we will draw our final conclusions on the sub-questions:

- *How does sampling frequency affect the performance of Deep Reinforcement Learning approaches to Traffic Signal Control, specifically in terms of reducing time-loss and rejecting input disturbances?*

  To address this question, we implemented a PPO model for TSC of a four-legged single intersection, capable of operating at various sampling intervals. In Case Study A-I, we found that an optimal sampling interval (Δt) of 4 to 6 seconds minimizes time-loss. A PPO approach using short sampling intervals, such as 1 second, resulted in increased time-loss. Methods proposed to improve the performance of the high-frequency sampling ($\Delta t = 1$ second) model, such as IAM or parallel training, did not improve performance. In terms of rejecting input traffic volume disturbances, a PPO model with $\Delta t = 5$ seconds demonstrated superior performance, achieving steady-state roughly 250 seconds faster than the high-frequency sampling PPO with IAM. It also outperformed traditional approaches such as max-pressure and fixed-time strategies in terms of overshoot and settling time.

- *Can a Deep Reinforcement Learning Agent effectively prioritize between pedestrians and vehicles in terms of time-loss?*

  In Case Study B, we addressed this sub-question by developing a dual-modal DRL framework which included both vehicle and pedestrian traffic flows. It utilized methods such as IAM to manage minimal signal timing constraints of the different traffic modalities. The results indicated a clear proportional decrease in time-loss for the prioritized group. Again, IAM did not improve performance in terms of time-loss compared to the more straightforward Action Overrule method. The results of this thesis show that PPO can be effectively used to prioritize traffic modes in multi-modal TSC scenarios.

With the research sub-questions addressed, we now turn to the main research question. This study has identified four challenges in applying DRL to TSC in real-world scenarios, and made an effort to address these challenges. The following section provides an informed response to the main research question:

Sampling Frequency. Selecting an appropriate sampling frequency is critical, as it influences the agent's ability to make timely and effective decisions. A frequency that is too high may introduce excessive noise, leading to suboptimal performance, while a frequency that is too low may cause the agent to react too slowly to changing traffic conditions.

Our results empirically show that increasing the sampling frequency for a Proximal Policy Optimization (PPO) model in a four-legged single-intersection traffic control problem does not improve performance in terms of time-loss. We found that an optimal sampling interval (Δt) of 4 to 6 seconds minimizes time-loss. The hypothesis that a higher sampling frequency would result in better performance was not supported in our experimental setup. However, this does not necessarily mean that PPO cannot perform better at higher frequencies in other contexts; further research is needed to explore this possibility. The challenge of improving responsiveness for DRL agents in traffic signal control without sacrificing performance remains unresolved.

Prioritization in Multi-Modal Traffic Environments. Real-world intersections often involve diverse road users, such as vehicles, pedestrians, bicyclists, etc. A Traffic Signal Control system should effectively manage various traffic modes to minimize overall time-loss while allowing for dynamic prioritization.

In Case Study B, we built a multi-modal Traffic Signal Control environment and applied PPO with IAM to manage the traffic signaler. The environment consisted of a 4-legged intersection with vehicle- and pedestrian traffic flows. We have shown that PPO can effectively prioritize different modalities in a two-modal traffic scenario, displaying a clear proportional decrease in time-loss for the prioritized modality. This suggests that DRL can be used to address the multi-modality in real-world Traffic Signal Control scenarios.

Robustness Against Varying Traffic. Urban traffic flows naturally fluctuate throughout the day, presenting a challenge for designing DRL agents that can effectively handle varying scenarios in real-world implementations.

In Case Study A-II, we trained PPO models on various traffic scenarios, ranging from constant light traffic (1800 vehicles per hour) to constant heavy traffic (3600 vehicles per hour). Results showed that PPO operating at $\Delta t = 5$ seconds was ineffective at higher traffic volumes, being outperformed by an adaptive max-pressure approach in all but the light traffic scenario.

We also trained PPO models on fluctuating traffic scenarios but did not observe performance improvements across different conditions. This suggests that training a PPO model on highly varying traffic does not improve its ability to handle diverse traffic scenarios.

However, a PPO model trained on light traffic effectively mitigated a threefold increase in traffic volume over a 300-second period. It outperformed methods such as max-pressure and fixed-time by producing a smaller rise in vehicle numbers and achieving the fastest settling time. This indicates a degree of resilience against traffic volume disturbances when overall throughput is relatively low.

These findings suggest that while DRL shows promise in managing certain traffic conditions, the highly varying patterns observed in real-world settings remain too complex for consistent handling. Therefore, DRL is not yet fully prepared to manage all such disturbances effectively.

Managing Minimal Signal Timing Constraints. In real-world TSC, different traffic modalities require varying minimal signal timings, with pedestrians often needing longer minimal green times. Ensuring safety necessitates a reliable method to enforce these constraints, which poses significant challenges for DRL agents in complex TSC environments.

We experimented with various methods to manage minimal signal timing constraints, including a straightforward Action Overrule method, Invalid Action Masking (IAM), and penalization of invalid actions. In Case Studies A-II and B, we showed that applying IAM to vehicle-only and multi-modal traffic environments did not affect performance in terms of time-loss compared to the Action Overrule method. Both methods were effective at nullifying invalid actions and ensuring safety. A safety mechanism is essential in applying DRL to TSC due to safety requirements. Therefore, IAM is a viable method for DRL approaches in TSC, ensuring safety without compromising performance.

5-2 Scientific Contributions

This thesis has identified and investigated numerous aspects of the application of DRL in TSC, bringing the field a step closer towards real-world implementation. Below, we name the key contributions to the field of TSC and DRL:

- An application of PPO with IAM to a set of control problems: Multi-Modal Traffic Signal Control of 4-legged single intersection.
- A parametric analysis of the sampling frequency for PPO methods in a 4-legged single intersection.
- An extensive comparison of PPO methods against baseline methods in terms of sensitivity analysis on traffic volume, generalization and disturbance rejection.
- Provide insight into the effectiveness of IAM against other invalid action approaches such as penalization.
- A demonstration of the capacity of PPO to prioritize traffic modes in an intersection with both vehicle and pedestrian traffic flows.

5-3 Limitations and Future work

Although this thesis showed promising results, in the following sections we mention limitations of the research and suggestions for future work.

High Speed Traffic Scenario. To assess the effect of high-frequency sampling in TSC cases, we propose exploring scenarios where high-frequency sampling is more advantageous. One such scenario is *ramp metering*, where a traffic signal controller manages vehicles merging onto a highway.

Incorporating Prior Knowledge into Action Selection. Given a certain state, only a limited number of action combinations are possible, as defined by the traffic signal configuration known a priori. However, the PPO agent spends considerable time learning these restrictions. An interesting approach would be to incorporate this prior knowledge into the action selection process. By predefining available lanes and actions, we could use a policy decision maker, such as PPO, more efficiently. This might involve deploying multiple PPO agents, each with its own action and state space.

Invalid Action Masking Scenario. An observation of the mechanics of a PPO with Invalid Action Masking (IAM) in a 4-phase action space is that only one decision is possible when actions are masked. In other words, when unmasked, all actions are valid choices; when masked, only one action remains available, as shown below:

$$p(a|s) = [0.05, 0.8, 0.1, 0.05] \xrightarrow{m(\cdot)} p^m(a|s) = [0, 1, 0, 0]$$

Therefore, any advantage $\hat{A}$ linked to the action is irrelevant since the action is predetermined in this state s. Consequently, we update the policy loss L^{CLIP} based on samples that cannot be improved. In bigger action spaces, where multiple actions are possible after the masking procedure, IAM might be more interesting. In these scenarios, it is more useful for DRL agents to be directed to valid subset of actions. Masking is primarily used in scenarios with very large and complex action spaces [25, 38]. For more complex traffic control scenarios, action masking could become a more relevant topic for future research.

Improved Observation and Action Space. Expanding the observation space to include multiple measurements of lanes, or even encoded video material, would allow the agent to see incoming cars rather than just queue length and density. This could drastically improve performance. In complex traffic scenarios such as big, a larger action space could also be beneficial.

Model Predictive Control with Deep Reinforcement Learning. Combining Model Predictive Control (MPC) for a stable, robust baseline with Deep Reinforcement Learning (DRL) for flexible, high-frequency adjustments is a promising approach, especially in traffic signal control scenarios. While not explored in this thesis, it represents an interesting direction for future research. Sun [32] provides an excellent review of MPC and DRL integration in urban traffic control. Remmerswaal et al. [27] use a DQN agent to adaptively adjust the controller input $u_c = u_{\text{MPC}} + u_{\text{RL}}$, where u_{RL} is a continuous variable representing the percentage of green time. In short, merging the robustness of MPC with the adaptability of DRL remains a compelling area for future work.

Ensemble Methods. Ensemble methods can improve robustness in Traffic Signal Control (TSC) by combining the decisions of multiple DRL agents trained with different initializations or data subsets. This approach reduces individual model biases and improves generalization to unseen traffic patterns. By aggregating decisions through techniques like voting or averaging, ensemble methods enhance stability and adaptability, making them promising for real-world, dynamic TSC scenarios.

Self-supervised pre-training Finding the most effective observation space before using deep reinforcement learning approaches could be done by leveraging self-supervised learning. An approach could be to train a neural network to predict returns for predefined state-action pairs. If the neural network achieves constant predictions, one could argue that the observation space contains enough information to effectively predict returns. This way, the dimensions of the DRL agent can be defined more effectively.

Appendix A

Supportive Figures

A-1 Case Study A-I

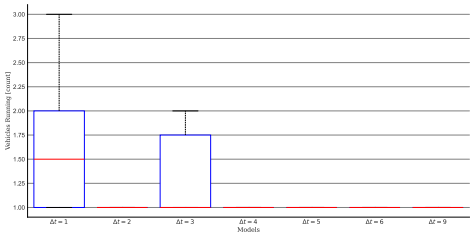


Figure A-1: Vehicles Running: Number of vehicles running over time (seconds).

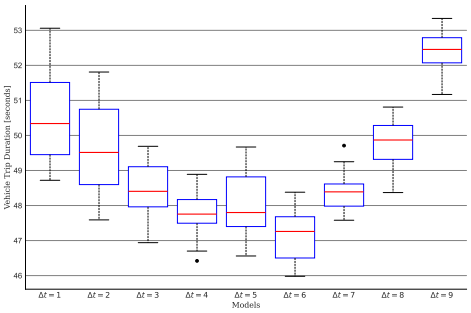


Figure A-2: Duration: Total duration (seconds) for different simulation runs.

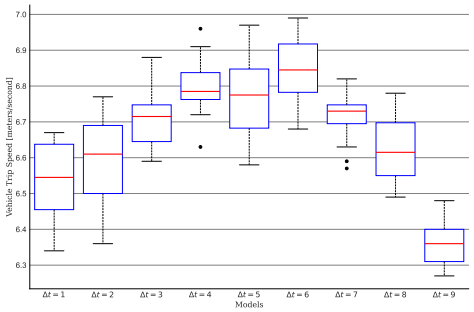


Figure A-3: Average Speed: Speed (m/s) over time for varying models during the testing phase.

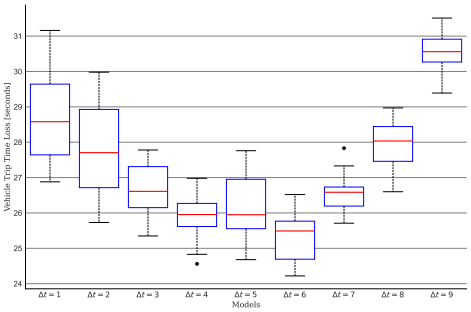


Figure A-4: Total Travel Time (TTT): Total time (seconds) spent by all vehicles in the simulation.

Figure A-5: Overview of key metrics for Case Study A-1, illustrating vehicle dynamics and performance over time.

Δt	Speed [m/s]	Duration [s]	Time Loss [s]	Total Travel Time [s]
1	0.0119	1.93	1.88	14875035
2	0.0151	1.71	1.75	16365516
3	0.0068	0.69	0.55	7676896
4	0.0056	0.41	0.39	6663728
5	0.0128	0.85	0.81	8641622
6	0.0079	0.55	0.52	7174361
7	0.0046	0.29	0.27	5443347
8	0.0078	0.45	0.42	6461675
9	0.0040	0.35	0.33	6289215

Table A-1: Variance of Metrics for Different Δt Values.

Metric	A-I-1	A-I-2	A-I-3	A-I-4	A-I-5	A-I-6	A-I-7	A-I-8	A-I-9
Training Time [s]	4963.91	2587.09	1824.13	1486.52	1284.75	1119.27	1017.38	954.32	881.32
CPU Usage [%]	54.04	43.67	37.71	37.75	37.87	28.98	25.85	24.76	22.23

Table A-2: Training Time and CPU Usage for Different Models.

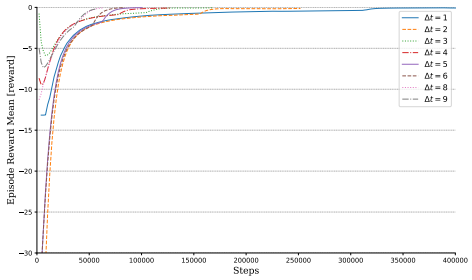


Figure A-6: Mean episode reward over time with exp. moving average with $\alpha = 0.3$

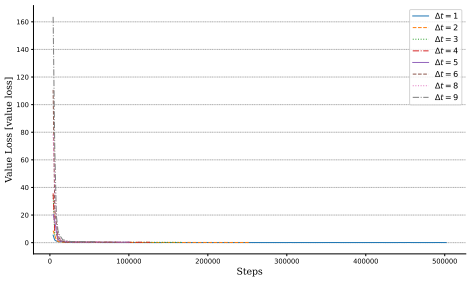


Figure A-7: Entropy Loss: The entropy loss over training steps.

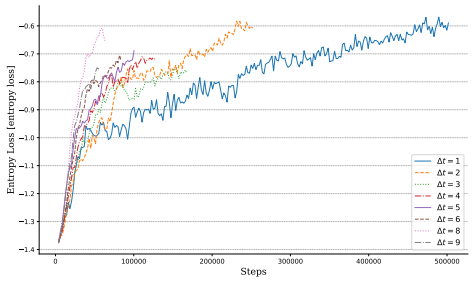


Figure A-8: Entropy Loss: The entropy loss over training steps.

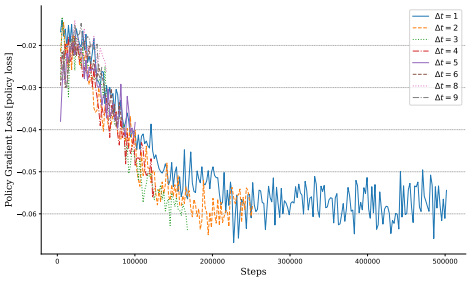


Figure A-9: Gradient Loss: The Gradient loss over training steps.

Figure A-10: Overview of key metrics for the training phase, illustrating loss functions and reward collection over time for Case Study A-1.

Δt	Vehicles [#]	Speed [m/s]	Duration [s]	Time Loss [s]	TTT [s]
1	2	6.55	50.34	28.58	96237
2	1	6.61	49.52	27.71	94886
3	1	6.72	48.41	26.61	92683
4	1	6.79	47.76	25.96	91685
5	1	6.78	47.80	25.95	91598
6	1	**6.85**	**47.26**	**25.49**	**90216**
7	1	6.73	48.39	26.59	92352
8	**0**	6.62	49.87	28.04	95526
9	1	6.36	52.46	30.56	100859

Table A-3: Median Metrics for Different Δt Values.

A-2 Case Study A-II

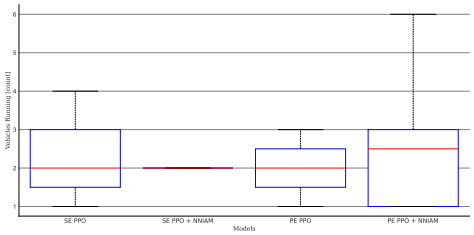


Figure A-11: Vehicles Running: Number of vehicles running over time (seconds).

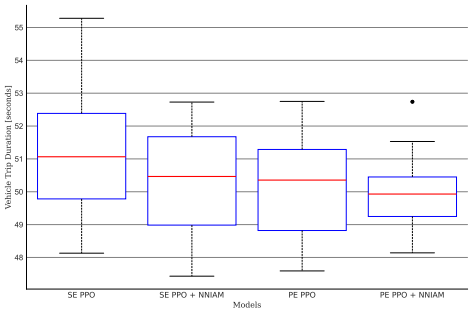


Figure A-12: Duration: Total duration (seconds) for different simulation runs.

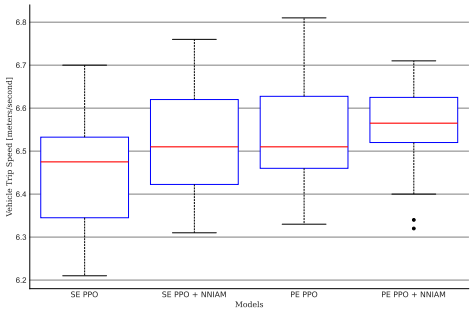


Figure A-13: Average Speed: Speed (m/s) over time for varying models during the testing phase.

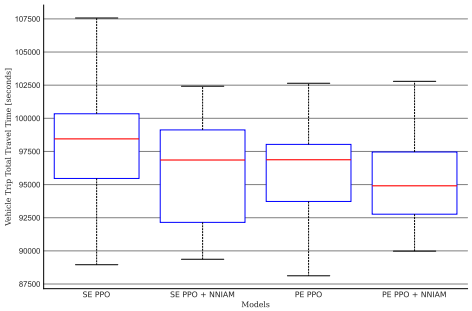
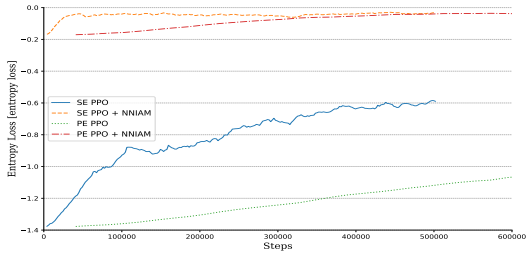
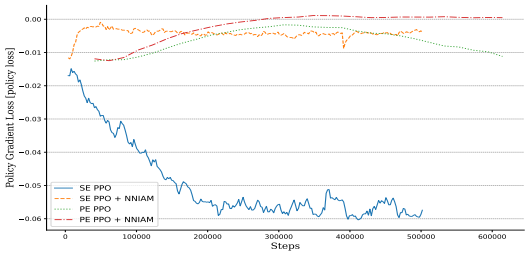


Figure A-14: Total Travel Time (TTT): Total time (seconds) spent by all vehicles in the simulation.

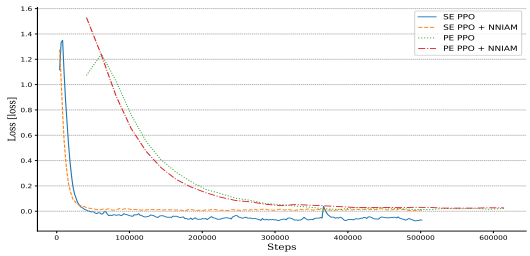
Figure A-15: Overview of key metrics for Case Study A-II, illustrating vehicle dynamics and performance over time.



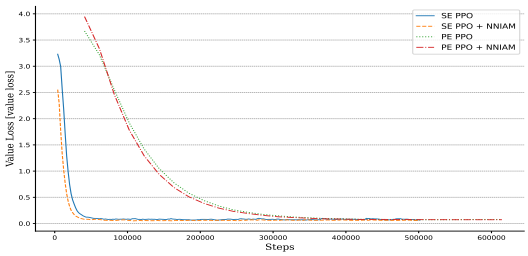
(a) Entropy Loss: Entropy loss over training epochs.



(b) Gradient Loss: Gradient loss during training.

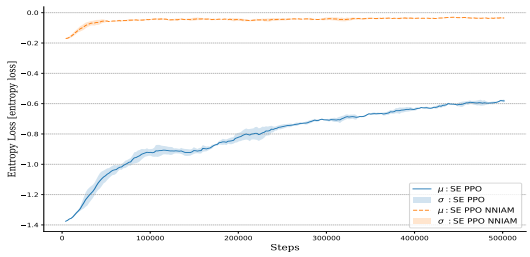


(c) Total Loss: Overall loss during training.

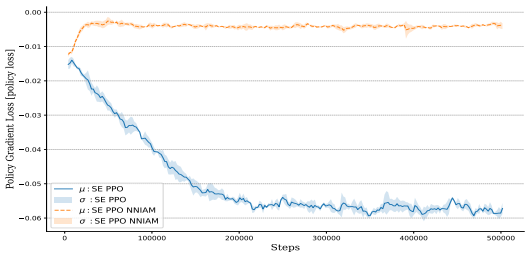


(d) Value Loss: Loss related to the value function.

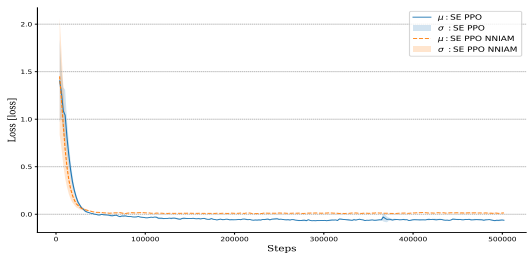
Figure A-16: Overview of key training metrics for Case Study A-II, illustrating different losses over the training process.



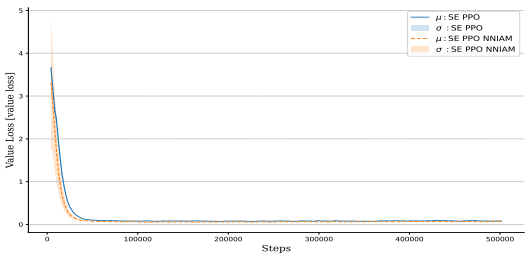
(a) Entropy Loss: Entropy loss over training epochs.



(b) Gradient Loss: Gradient loss during training.

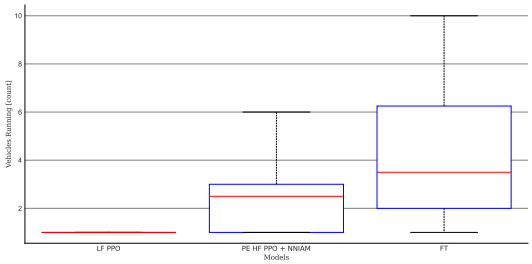


(c) Total Loss: Overall loss during training.

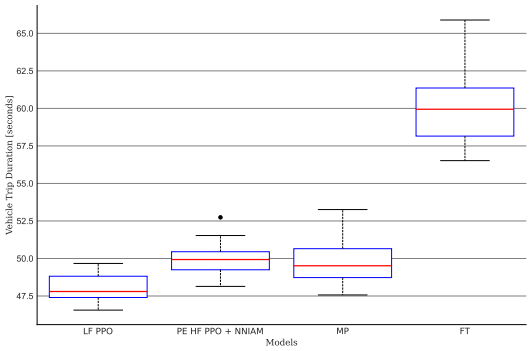


(d) Value Loss: Loss related to the value function.

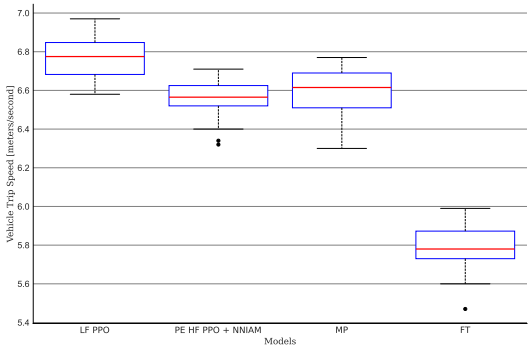
Figure A-17: Overview of key training metrics with variance for Case Study A-II, illustrating different losses over the training process.



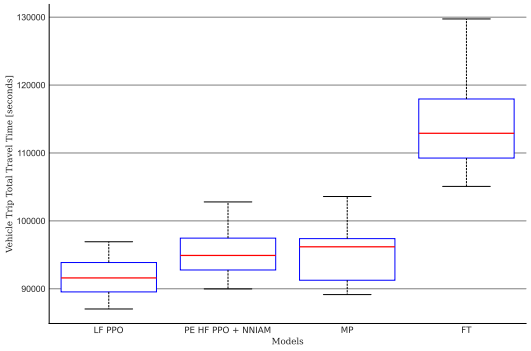
(a) Comparison of best-performing models based on vehicles running.



(b) Comparison of best-performing models based on trip duration.



(c) Comparison of best-performing models based on average speed.



(d) Comparison of best-performing models based on total travel time.

Figure A-18: Overview of performance metrics for the best models across different scenarios, highlighting trip duration, vehicles running, speed, and total travel time for Case Study A-II.

Metric	SE PPO	SE PPO+NNIAM	PE PPO	PE PPO+NNIAM
Veh. Running [#]	**2.0**	**2.0**	**2.0**	2.5
Speed [m/s]	6.475	6.51	6.51	**6.565**
Duration [s]	51.065	50.465	50.355	**49.93**
Time Loss [s]	29.255	28.65	28.54	**28.03**
TTT [s]	98448.0	96854.0	96873.5	**94908.5**

Table A-4: Median Metrics for models A-II-(1-4). In most performance metrics, the parallel training PPO with NNIAM outperforms the other high-frequency models.

Metric	SE PPO	SE PPO+IAM	PE PPO	PE PPO+IAM
Veh. Running [#]	1.10	N/A	1.00	1.69
Speed [m/s]	0.1473	0.1430	0.1479	**0.1078**
Duration [s]	1.9247	1.6652	1.6838	**1.2773**
Time Loss [s]	1.9105	1.6575	1.6661	**1.2689**
TTT [s]	4876.69	4288.27	4453.04	**3706.38**

Table A-5: Standard Deviation of Metrics for models A-II-(1-4). The parallel training PPO with NNIAM shows lower variability in most metrics, indicating more stable performance.

Scn.	Metric	LF PPO	HF PPO	MP	MP ext.	FT
Baseline	**Veh. Running [#]**	1	2	**0**	**0**	5
	Speed [m/s]	**6.7733**	6.5522	6.5839	6.4117	5.7867
	Duration [s]	**48.0156**	50.0406	49.8156	53.4139	59.9667
	Time Loss [s]	**26.1717**	28.1994	27.7729	31.5617	38.1156
	TTT [s]	**91798**	95609	95252	102045	114548
Scenario 1	**Veh. Running [#]**	62	90	89	**4**	70
	Speed [m/s]	4.2283	3.7533	3.6829	**5.5417**	4.3567
	Duration [s]	101.1694	120.6683	119.7406	**63.2389**	99.7839
	Time Loss [s]	79.3267	98.8339	97.9	**41.3811**	77.9372
	TTT [s]	272116	308950	314525	**174216**	256646
Scenario 2	**Veh. Running [#]**	114	100	117	74	**70**
	Speed [m/s]	3.5212	3.3250	3.1867	3.9711	**4.1267**
	Duration [s]	129.5633	138.0994	144.1161	**101.0261**	104.1994
	Time Loss [s]	107.7439	116.2828	122.2939	**79.1750**	82.3550
	TTT [s]	379098	372567	398888	329629	**295326**
Scenario 3	**Veh. Running [#]**	121	134	137	130	**74**
	Speed [m/s]	3.2844	2.7900	2.8594	3.5028	**3.7783**
	Duration [s]	136.5939	157.1317	156.08	121.1383	**109.2606**
	Time Loss [s]	114.7950	135.3144	134.2733	99.3117	**87.4300**
	TTT [s]	418031	438019	445962	415650	**334212**

Table A-6: Mean Metrics for models on varying traffic volume scenarios. Performance metrics are compared across LF PPO, HF PPO, Max-pressure, Max-pressure Extended, and Fixed-time models.

Scn.	Metric	LF PPO	HF PPO	MP	MP ext.	FT
Baseline	**Veh. Running [#]**	N/A	1.69	N/A	N/A	2.84
	Speed [m/s]	0.11	0.11	0.14	0.06	0.14
	Duration [s]	0.92	1.28	1.64	0.59	2.55
	Time Loss [s]	0.90	1.27	1.45	0.59	2.56
	TTT [s]	2939.66	3706.38	4377.24	2363.12	6536.44
Scenario 1	**Veh. Running [#]**	15.77	5.68	15.50	3.52	1.82
	Speed [m/s]	0.20	0.11	0.15	0.07	0.08
	Duration [s]	7.57	5.13	6.55	1.10	2.86
	Time Loss [s]	7.56	5.14	6.54	1.10	2.87
	TTT [s]	21770.98	13797.10	18725.00	5216.13	8119.63
Scenario 2	**Veh. Running [#]**	6.71	6.32	8.14	16.22	1.75
	Speed [m/s]	0.13	0.13	0.10	0.17	0.07
	Duration [s]	3.48	3.89	2.97	5.44	1.56
	Time Loss [s]	3.49	3.88	2.94	5.44	1.58
	TTT [s]	10340.09	9332.50	8360.20	18982.42	5240.54
Scenario 3	**Veh. Running [#]**	4.24	15.73	4.49	2.47	3.69
	Speed [m/s]	0.15	0.16	0.08	0.06	0.08
	Duration [s]	3.86	6.19	2.17	2.53	1.53
	Time Loss [s]	3.84	6.17	2.18	2.52	1.53
	TTT [s]	9024.30	14728.57	6232.27	9393.99	6443.00

Table A-7: Standard deviation metrics for models on varying traffic volume scenarios. The table displays standard deviations for key performance metrics across LF PPO, HF PPO, Max-pressure, Max-pressure Extended, and Fixed-time models.

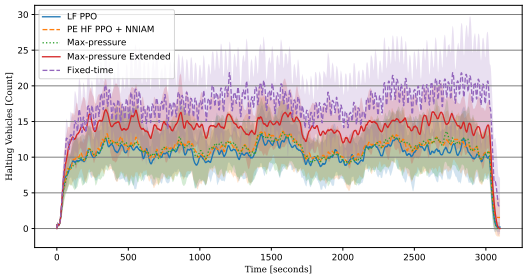


Figure A-19: Vehicles Halting - Scenario Baseline

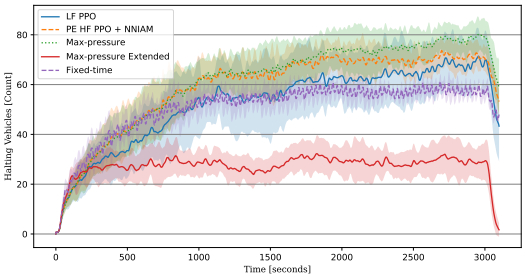


Figure A-20: Vehicles Halting - Scenario 2

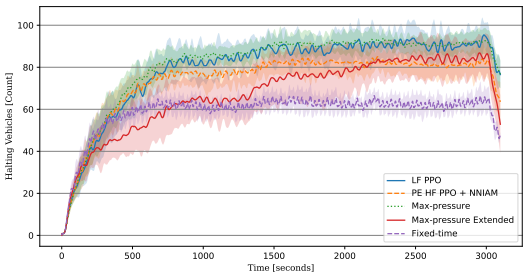


Figure A-21: Vehicles Halting - Scenario 3

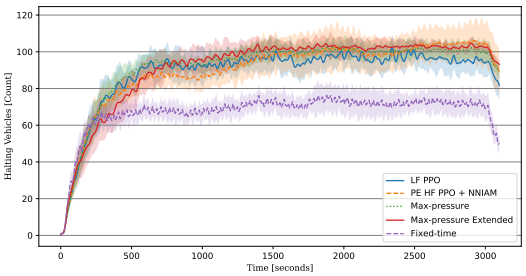


Figure A-22: Vehicles Halting - Scenario 4

Figure A-23: Vehicles Halting in Different Traffic Scenarios

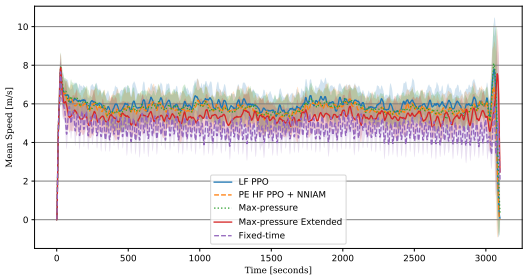


Figure A-24: Mean Speed - Scenario baseline

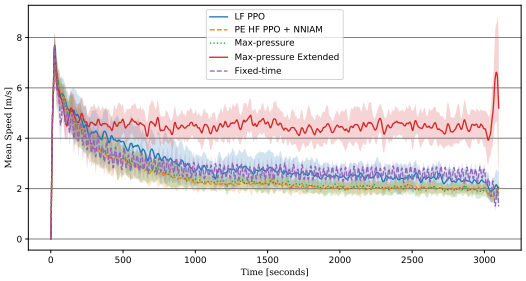


Figure A-25: Mean Speed - Scenario 2

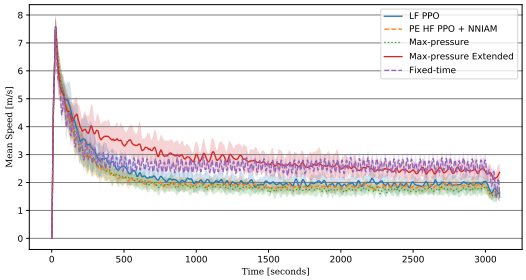


Figure A-26: Mean Speed - Scenario 3

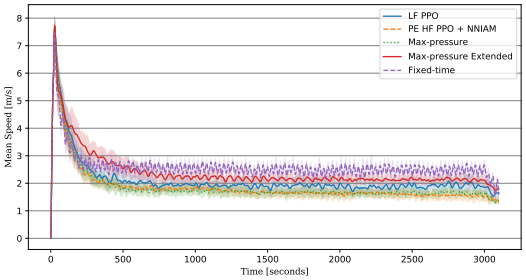


Figure A-27: Mean Speed - Scenario 4

Figure A-28: Mean Speed in Different Traffic Scenarios

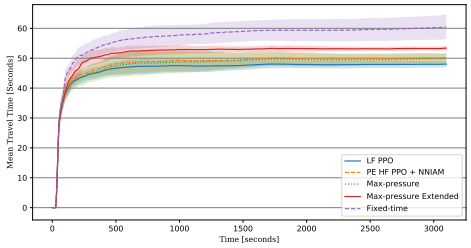


Figure A-29: Mean Travel Time - Scenario 300

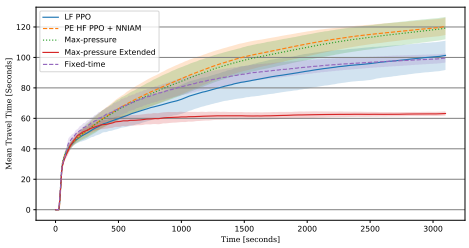


Figure A-30: Mean Travel Time - Scenario 400

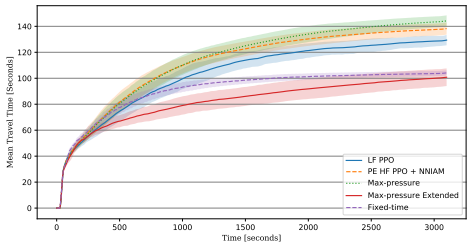


Figure A-31: Mean Travel Time - Scenario 500

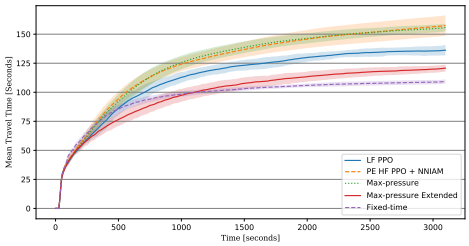


Figure A-32: Mean Travel Time - Scenario 600

Figure A-33: Mean Travel Time in Different Traffic Scenarios

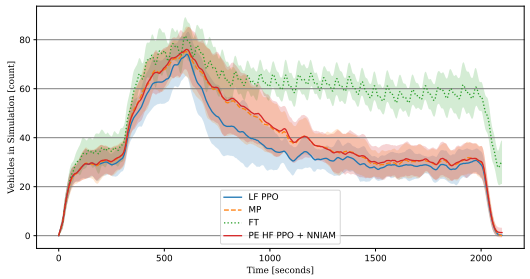


Figure A-34: Mean Travel Time - Scenario Baseline

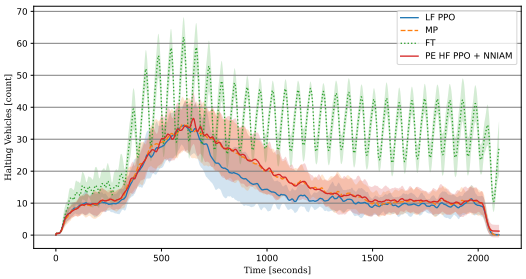


Figure A-35: Mean Travel Time - Scenario 2

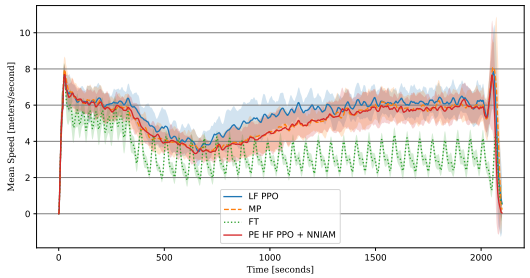


Figure A-36: Mean Travel Time - Scenario 3

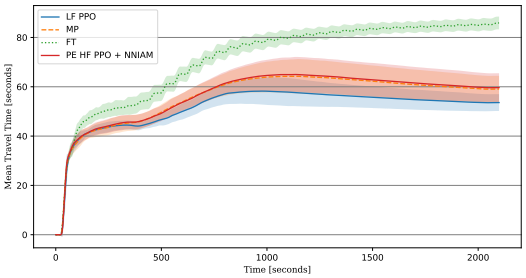


Figure A-37: Mean Travel Time - Scenario 4

Figure A-38: Mean Travel Time in Different Traffic Scenarios

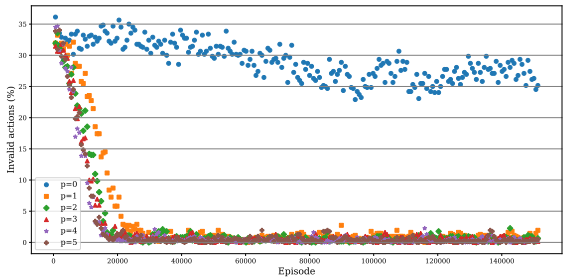


Figure A-39: Training with Penalization (Invalid Actions, d5).

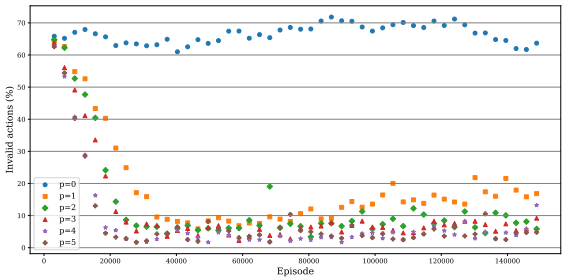


Figure A-40: Training with Penalization (Invalid Actions, d1).

Figure A-41: Comparison of training results with penalization for invalid actions under different discount factors (d5 and d1).

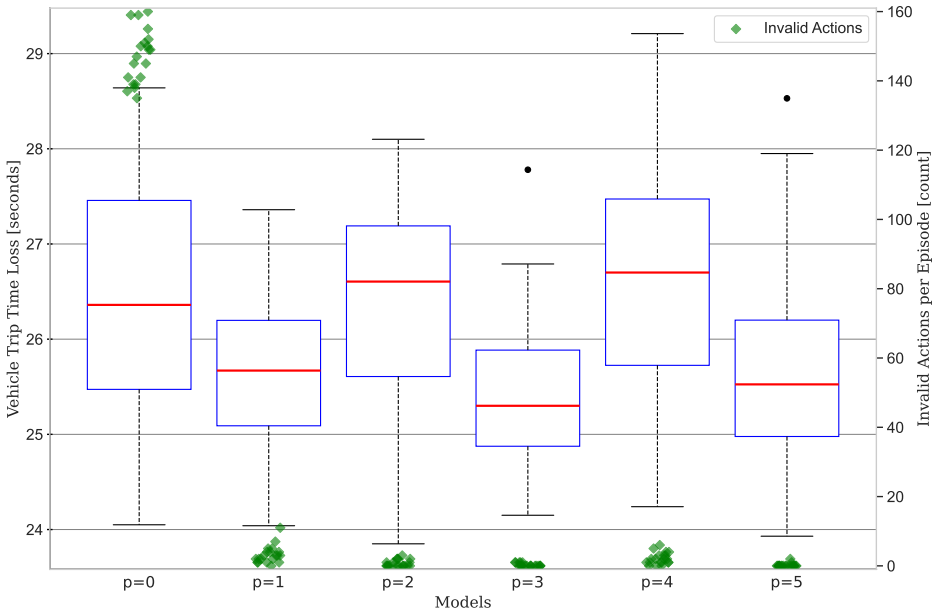


Figure A-42: Time-loss and amount of invalid actions at $\Delta t = 5$. These results highlight the performance of penalized models at a lower sampling frequency.

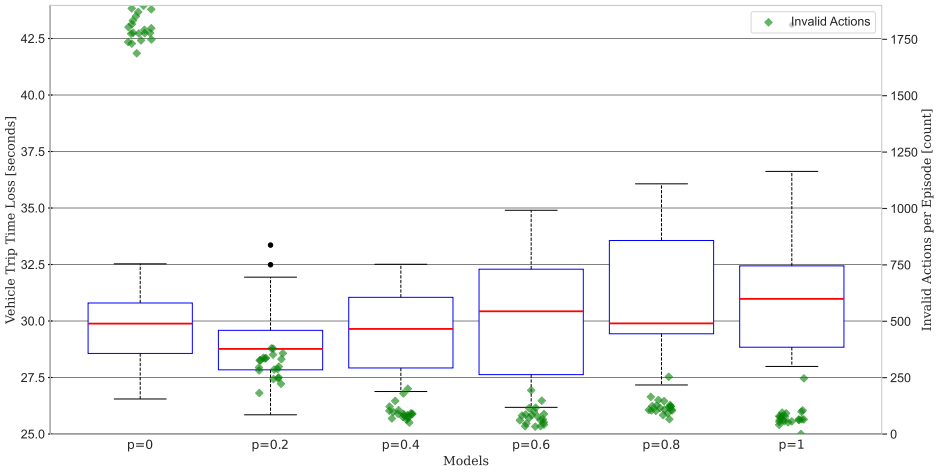


Figure A-43: Time-loss and amount of invalid actions at $\Delta t = 1$. These results emphasize the performance of penalized models at a higher sampling frequency.

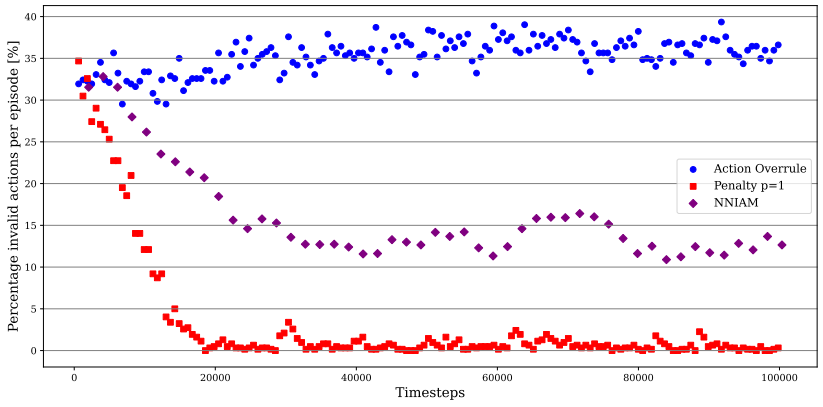


Figure A-44: Magnified view of invalid actions per episode over training time at $\Delta t = 5$. The penalized models rapidly reduce invalid actions to near zero, while NNIAM models show a gradual decrease.

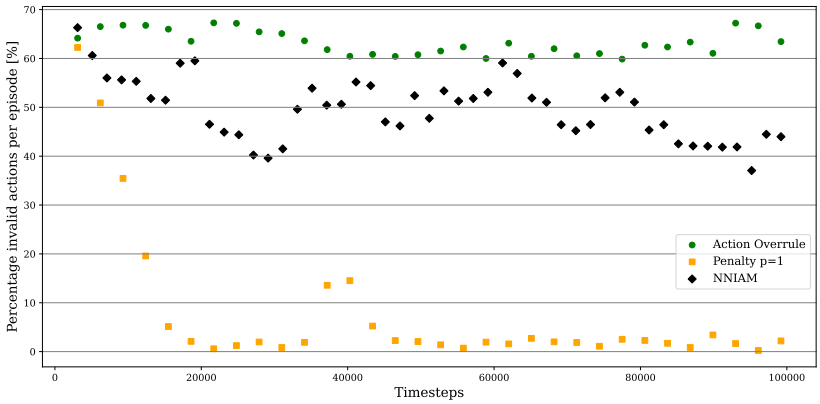


Figure A-45: Magnified view of invalid actions per episode over training time at $\Delta t = 1$. The penalized models rapidly reduce invalid actions to near zero, while NNIAM models show a gradual decrease.

A-3 Case Study B

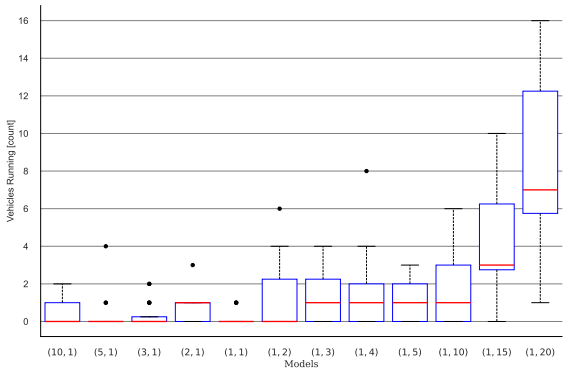


Figure A-46: Amount of vehicles in simulation as simulation ends.

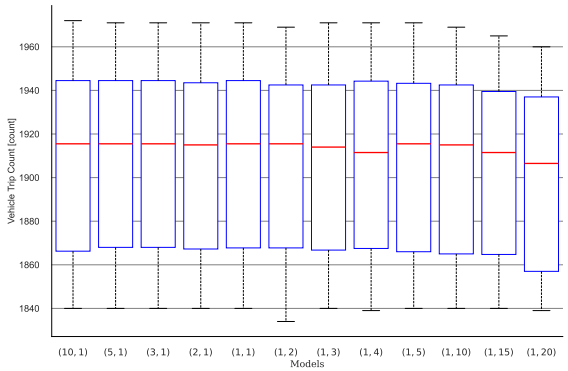
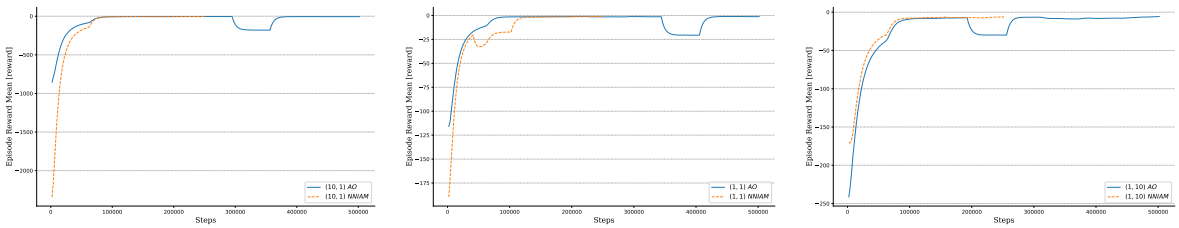


Figure A-47: Amount of finished vehicle trips.

Figure A-48: These plots display the decrease in priority for vehicles as the pedestrian weighing factor grows. We see a rise in unfinished vehicle trips.



(a) Weight scenario (10, 1).

(b) Weight scenario (1, 1).

(c) Weight scenario (1, 10).

Figure A-49: Comparison of Action Overrule against NNIAM across different weight scenarios.

Bibliography

[1] MUTCD 11th edition - part 4. page 116.

[2] Francesco Abbracciavento, Francesco Zinnari, Simone Formentin, Andrea G. Bianchessi, and Sergio M. Savaresi. Multi intersection traffic signal control: A decentralized MPC-based approach. 23:100214.

[3] Alfredo V. Clemente, Humberto N. Castejón, and Arjun Chandra. Efficient parallel methods for deep reinforcement learning.

[4] Myungeun Eom and Byung-In Kim. The traffic signal control problem for intersections: a review. 12(1):50.

[5] Jakob Erdmann. SUMO's lane-changing model. In Michael Behrisch and Melanie Weber, editors, *Modeling Mobility with Open Data*, pages 105–123. Springer International Publishing. Series Title: Lecture Notes in Mobility.

[6] Paul A Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation.* Wiley, 1 edition.

[7] Gilbert C Gee and David T Takeuchi. Traffic stress, vehicular burden and well-being: A multilevel analysis. 59(2):405–414.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* Adaptive computation and machine learning. The MIT Press.

[9] Jim Gorzelany. These are the cities where motorists lose the most time and money sitting in traffic. Section: Cars & Bikes.

[10] Guangjie Han, Qi Zheng, Lyuchao Liao, Penghao Tang, Zhengrong Li, and Yintian Zhu. Deep reinforcement learning for intersection signal control considering pedestrian behavior. 11(21):3519.

[11] Brian Hayes. First links in the markov chain. 101(2):92.

[12] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning.

[13] Yueqi Hou, Xiaolong Liang, Jiaqiang Zhang, Qisong Yang, Aiwu Yang, and Ning Wang. Exploring the use of invalid action masking in reinforcement learning: A comparative study of on-policy and off-policy algorithms in real-time strategy games. 13(14):8283.

[14] Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. 35.

[15] P. Hunt, D. Robertson, R. Bretherton, and M. Royle. THE SCOOT ON-LINE TRAFFIC SIGNAL OPTIMISATION TECHNIQUE.

[16] INRIX. INRIX 2023 global traffic scorecard: London most congested city in europe; congestion costing the UK £7.5 billion.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[18] Stefan Krauss. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. page 129.

[19] Jonathan I. Levy, Jonathan J. Buonocore, and Katherine von Stackelberg. Evaluation of the public health impacts of traffic congestion: a health risk assessment. 9(1):65.

[20] Shu Lin, Bart De Schutter, Yugeng Xi, and Hans Hellendoorn. Efficient network-wide model-based predictive control for urban traffic networks. 24:122–140.

[21] John D C Little, Mark D Kelson, and Nathan H Gartner. MAXBAND: A program for setting signals on arteries and triangular networks.

[22] Meng Long, Xiexin Zou, Yue Zhou, and Edward Chung. Deep reinforcement learning for transit signal priority in a connected environment. 142:103814.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning.

[24] Arthur Müller, Vishal Rangras, Georg Schnittker, Michael Waldmann, Maxim Friesen, Tobias Ferfers, Lukas Schreckenberg, Florian Hufen, Jürgen Jasperneite, and Marco Wiering. Towards real-world deployment of reinforcement learning for traffic signal control.

[25] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning.

[26] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. 37(4):1–14.

[27] Willemijn Remmerswaal, Dingshan Sun, Anahita Jamshidnejad, and Bart De Schutter. Combined MPC and reinforcement learning for traffic signal control in urban traffic networks. In *2022 26th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 432–439. IEEE.

[28] D. Robertson. RESEARCH ON THE TRANSYT AND SCOOT METHODS OF SIGNAL COORDINATION.

[29] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms.

[31] A. Stevanovic. SCOOT and SCATS: A closer look into their operations. University of Utah.

[32] D. Sun. Multi-level and learning-based model predictive control for traffic management.

[33] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. MIT Press.

[34] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction (2nd)*. MIT Press, Cambridge, MA.

[35] Kai Liang Tan, Subhadipto Poddar, Anuj Sharma, and Soumik Sarkar. Deep reinforcement learning for adaptive traffic signal control.

[36] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. version: 3.

[37] Pravin Varaiya. Max pressure control of a network of signalized intersections. 36:177–195.

[38] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II: A new challenge for reinforcement learning.

[39] Yanan Wang, Tong Xu, Xin Niu, Chang Tan, Enhong Chen, and Hui Xiong. STMARL: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control.

[40] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning.

[41] F. V. Webster. Traffic signals. road research technical paper.

[42] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods.

[43] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. IntelliLight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2496–2505. ACM.

[44] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. 8(3):229–256.

[45] Zhihui Xie, Zichuan Lin, Junyou Li, Shuai Li, and Deheng Ye. Pretraining in deep reinforcement learning: A survey.

[46] Bao-Lin Ye, Weimin Wu, Keyu Ruan, Lingxi Li, Tehuan Chen, Huimin Gao, and Yaobin Chen. A survey of model predictive control methods for traffic signal control. 6(3):623–640.

Glossary

List of Acronyms

RL Reinforcement Learning

DRL Deep Reinforcement Learning

MDP Markov Decision Process

DP Dynamic Programming

MC Monte-Carlo

TD Temporal Difference

TSC Traffic Signal Control

SUMO Simulator of Urban MObility

PPO Proximal Policy Optimization

TRPO Trust Region Policy Optimization

NN neural network

GAE Generalized Advantage Estimation

IAM Invalid Action Masking

NIAM Naive Invalid Action Masking

NNIAM Non-naive Invalid Action Masking

List of Symbols

Number Sets

$\mathbb{R}$	Set of Real numbers.
$\mathbb{R}_{\geq 0}$	Set of Non-negative real numbers.
$\mathbb{N}$	Natural numbers (positive integers).

Reinforcement Learning Symbols

s_t	State in the environment at time t.
s	Realization of state in environment.
$\mathcal{S}$	Set of all possible states (state space).
a_t	Action at time t.
a	Action taken by the agent.
$\mathcal{A}$	Set of all possible actions (action space).
$p(s_{t+1} \mid s_t, a_t)$	Transition function, probability of state s_{t+1} given state s_t and action a_t.
$\mathcal{T}$	Transition function.
$r(s_t, a_t, s_{t+1})$	Reward function, immediate reward from a transition.
$\mathcal{R}$	Reward function.
$\pi(a \mid s)$	Policy, probability of taking action a in state s.
π^*	Optimal policy that maximizes expected return.
γ	Discount factor.
R_t	Return, cumulative discounted reward from time t.
$\mathbb{E}_\pi[R_t \mid s_t = s]$	Expected return under policy π starting from state s.
$V^\pi(s)$	Value function, expected return starting from state s under policy π.
$Q^\pi(s, a)$	Action-value function, expected return starting from state s, taking action a, and following policy π.
$A^\pi(s, a)$	Advantage function, the relative benefit of taking action a in state s under policy π.

Proximal Policy Optimization Symbols

θ	Parameters of the neural network (weights and biases).
θ_w	Weights in the neural network.
θ_b	Biases in the neural network.
$\phi(z)$	Activation function applied to z.

$V_\theta(s_t)$	Value function estimator at state s_t.
V_t^{targ}	Target value function at time t.
$R_t^{(1)}$	1-step return at time t.
$R_t^{(n)}$	n-step return at time t.
$R_t^{(\infty)}$	Monte-Carlo return from time t.
$R_t(\lambda)$	λ-return at time t.
λ	Decay parameter for GAE ($0 \leq \lambda \leq 1$).
$\hat{A}_t$	Estimated advantage (GAE) at time t.
$L_t^{\text{CLIP+VF+S}}(\theta)$	Full loss function in PPO.
$r_t(\theta)$	Probability ratio between new and old policy at time t.
ϵ	Clipping parameter in PPO (0.2).
c_{vf}	Value function coefficient.
c_{ent}	Entropy bonus coefficient.
$\pi_\theta(a_t \mid s_t)$	Policy parameterized by θ.
$\pi_{\theta_{\text{old}}}(a_t \mid s_t)$	Old policy.
α	Learning rate.
n_{steps}	Number of steps per update.
n_{batch}	Batch size.
n_{epochs}	Number of epochs per update.
T	Number of steps per update ($T = n_{steps}$)
$\mathcal{D}$	Trajectory memory.
$\mathcal{B}$	Mini-batch sampled from trajectory memory.