University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

AnyLight: A Generalizable Multi-Agent Reinforcement Learning Architecture for Heterogeneous Traffic Networks

Supervisor
Gian Antonio Susto
University of Padova

Co-supervisor
Hao Xue
University of New South Wales

Master Candidate
Marco Bustaffa

Academic Year
2025-2026

Abstract

Current traffic systems adapt poorly to dynamic demand. While Reinforcement Learning provides a solution, multi-agent approaches struggle on real-world networks with diverse intersection topologies. Independent policies scale linearly and isolate knowledge, whereas parameter-shared architectures demand rigid, fixed-dimensional spaces that inherently exclude heterogeneous networks.

This thesis presents *AnyLight*, a generalizable Multi-Agent Reinforcement Learning architecture for heterogeneous Traffic Signal Control, featuring three contributions. First, a movement-centric state representation ensures semantically aligned inputs across diverse junctions. Second, universal parameter-sharing enables a single Proximal Policy Optimization network to govern all intersections via dynamic padding and masking. Third, a cross-attention decoder and centralized critic exploit privileged neighbor-action information during Centralized Training, Decentralized Execution.

AnyLight is evaluated on six synthetic and real-world benchmark scenarios from the RESCO and MA2C suites. Compared to classical heuristics and learning-based baselines, *AnyLight* consistently demonstrates superior performance by minimizing intersection delays during high-density flows. Ablation studies confirm that combining the movement-centric representation, universal parameter sharing, and collaborative reward is crucial for scaling to complex urban environments.

Acknowledgments

I would like to express my sincere gratitude to Professor Flora Salim, Deputy Director of the UNSW AI Institute, for welcoming me into her group and providing me with this exceptional research opportunity.
I am also grateful to my supervisor, Prof. Gian Antonio Susto, and my co-supervisor, Dr. Hao Xue, for their continuous guidance, valuable feedback, and dedication throughout this research.
I also wish to thank my labmates at UNSW for their assistance, the stimulating discussions, and the supportive environment they provided during my stay.
I dedicate this work to the younger me, who dreamed of this moment so deeply yet always doubted himself along the way. To my family, whose continuous support has guided me from the very beginning, and especially to my mother, who taught me the true meaning of tenacity.

Contents

Abstract **iii**

Acknowledgments **v**

List of figures **ix**

List of tables **xi**

Listing of acronyms **xiii**

1 Introduction **1**
- 1.1 Thesis Outline . . . 3

2 The Traffic Signal Control Problem **5**
- 2.1 The Shift in Urban Management . . . 5
- 2.2 Limitations of Classical Control Theory . . . 6
- 2.3 TSC as a Stochastic Optimization Problem . . . 6

3 Theoretical Foundations of Reinforcement Learning **9**
- 3.1 The Reinforcement Learning Framework . . . 10
- 3.2 Markov Decision Processes (MDPs) . . . 11
- 3.3 Value Functions and Bellman Equations . . . 12
- 3.4 Deep Reinforcement Learning (DRL) . . . 12
- 3.5 Value-Based vs. Policy-Based Methods . . . 13
- 3.6 Actor-Critic Methods . . . 13
  - 3.6.1 The Advantage Function . . . 15
  - 3.6.2 Trust Region Policy Optimization (TRPO) . . . 16
  - 3.6.3 Proximal Policy Optimization (PPO) . . . 17
- 3.7 Online and Offline Reinforcement Learning . . . 19
  - 3.7.1 Online Reinforcement Learning . . . 19
  - 3.7.2 Offline Reinforcement Learning . . . 19
- 3.8 Multi-Agent Reinforcement Learning (MARL) . . . 20
  - 3.8.1 Execution and Training Paradigms . . . 21

4 Problem Formalization **23**
- 4.1 Fundamental Traffic Terminology . . . 23

4.2 The Simulation Framework: SUMO and TraCI . . . 25
4.3 State Space . . . 27
4.4 Action Space . . . 31
4.5 Reward Function . . . 32

5 AnyLight: A Generalizable Architecture and Training Strategy **35**
5.1 The Challenge of Network Heterogeneity . . . 36
5.2 Multi-Agent Topologies: Independent vs. Shared Policies . . . 37
5.3 Padding And Masking Strategy . . . 38
5.4 Proposed Neural Network Architecture . . . 41
5.5 Training . . . 44
5.5.1 Environment Runner . . . 45
5.5.2 Learner . . . 47

6 Experimental Framework and Simulation Scenarios **53**
6.1 Traffic Datasets . . . 53
6.2 Baselines and Comparison Models . . . 59
6.2.1 Traditional Traffic Signal Control Methods . . . 59
6.2.2 Reinforcement Learning Baselines . . . 61
6.3 Training and Evaluation Protocol . . . 62
6.3.1 Training Protocol . . . 62
6.3.2 Ablation Study Design . . . 63
6.4 Evaluation Metrics . . . 64
6.5 Hardware and Computational Setup . . . 65

7 Results and Comparative Evaluation **67**
7.1 Training Dynamics and Convergence . . . 67
7.2 Performance on Homogeneous Networks . . . 69
7.3 Performance on Heterogeneous Real-World Networks . . . 73
7.4 Ablation Studies . . . 77
7.5 Discussion of Results . . . 79

8 Conclusions and Future Work **81**
8.1 Summary of Contributions . . . 82
8.2 Limitations . . . 82
8.3 Future Work . . . 83

References **85**

Listing of figures

3.1 The Reinforcement Learning interaction loop between an agent and its environment. 10
3.2 The Actor-Critic architecture, with the Actor dictating agent behavior and the Critic evaluating action quality. 14

4.1 A standard four-way intersection with three movements per approach (left-turn, straight, right-turn), totaling 12 controlled movements. 27
4.2 The discrete action space of an agent, representing the set of green phases in a four-way intersection. 31

5.1 Architectural overview of the AnyLight model. 41
5.2 Architectural overview of the parallel experience collection pipeline. Three meta-agents manage independent simulation instances, aggregating heterogeneous experiences into a rollout buffer. The central learner retrieves these buffers, applies padding and normalization, and executes the PPO optimization cycle to update the shared model. 45

6.1 Topological structures of the six traffic networks under evaluation. The top row displays the synthetic homogeneous topologies, and the bottom row illustrates the real-world heterogeneous topologies. 54
6.2 Vehicle arrival rate distributions (vehicles per minute) over the simulation duration across the different traffic datasets. 58

7.1 Training curves for the AnyLight policy on the six benchmark networks. Each panel overlays the per-network learning signal: smoothed mean episode reward (upper-left), policy entropy in logarithmic scale (upper-right), critic value loss in logarithmic scale (lower-left), and action-change frequency at the agent level (lower-right). Curves are smoothed with a 5-point rolling mean for readability. 68
7.2 Behavioral and spatial analysis of the AnyLight policy on the grid4x4 network, illustrating temporal queue evolution and travel time distributions. 70
7.3 Behavioral and spatial analysis of the AnyLight policy on the arterial4x4 network, illustrating temporal queue evolution and travel time distributions. 71
7.4 Behavioral and spatial analysis of the AnyLight policy on the grid5x5 network, illustrating temporal queue evolution and travel time distributions. 72

7.5 Behavioral and spatial analysis of the AnyLight policy on the cologne_region network, illustrating temporal queue evolution and travel time distributions. 74
7.6 Behavioral and spatial analysis of the AnyLight policy on the ingolstadt_region network, illustrating temporal queue evolution and travel time distributions. 75
7.7 Behavioral and spatial analysis of the AnyLight policy on the monaco network, illustrating temporal queue evolution and travel time distributions. 76
7.8 Training curves for the ablation study on the Arterial 4×4 network, comparing the full AnyLight architecture against variants missing neighbor information and the collaborative reward. The plots illustrate episode reward, entropy loss, value loss, and action change frequency over the training duration. 78

Listing of tables

5.1 Full training configuration and architectural hyperparameters for the Any-Light model. 51

6.1 Overview of the traffic datasets, detailing network structure, volume, and vehicle arrival rates. 55
6.2 Intersection-specific structural details for the Cologne network. 56
6.3 Intersection-specific structural details for the Ingolstadt network. 56
6.4 Intersection-specific structural details for the Monaco network. 57

7.1 Performance comparison on the Grid 4×4 network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better. 70
7.2 Performance comparison on the Arterial 4×4 network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better. 71
7.3 Performance comparison on the Grid 5×5 network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better. 72
7.4 Performance comparison on the Cologne network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better. 74
7.5 Performance comparison on the Ingolstadt network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better. 75
7.6 Performance comparison on the Monaco network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better. 76

7.7 Ablation study on the Arterial 4×4 network, comparing the full AnyLight model against its two ablated variants. Reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. 77

Listing of acronyms

TSC . . . . . Traffic Signal Control

ITS . . . . . Intelligent Transportation Systems

MTT . . . . . Mean Travel Time

SCATS . . . . . Sydney Coordinated Adaptive Traffic System

SCOOT . . . . . Split Cycle Offset Optimisation Technique

MDP . . . . . Markov Decision Process

RL . . . . . Reinforcement Learning

MADRL . . . . . Multi-Agent Deep Reinforcement Learning

DQN . . . . . Deep Q-Network

PPO . . . . . Proximal Policy Optimization

MP . . . . . Markov Process

MRP . . . . . Markov Reward Process

TD . . . . . Temporal Difference

DRL . . . . . Deep Reinforcement Learning

MSE . . . . . Mean Squared Error

KL . . . . . Kullback-Leibler

TRPO . . . . . Trust Region Policy Optimization

CQL . . . . . Conservative Q-Learning

BCQ . . . . . Batch-Constrained Q-learning

MARL . . . . . Multi-Agent Reinforcement Learning

Dec-POMDP . . Decentralized Partially Observable Markov Decision Process

CTCE . . . . . Centralized Training Centralized Execution

DTDE . . . . . Decentralized Training Decentralized Execution

CTDE . . . . . Centralized Training Decentralized Execution

A2C . . . . . Advantage Actor-Critic

MA2C . . . . . Multi-Agent Advantage Actor-Critic

SARSA . . . . . State-Action-Reward-State-Action

Adam . . . . . Adaptive Moment Estimation

MATSC . . . . . Multi-Agent Traffic Signal Control

SUMO . . . . . Simulation of Urban MObility

TraCI . . . . . Traffic Control Interface

ATSC . . . . . Adaptive Traffic Signal Control

ATT . . . . . Average Travel Time

ILDs . . . . . Induction Loop Detectors

AI . . . . . Artificial Intelligence

DNN . . . . . Deep Neural Network

MAPPO . . . . . Multi-Agent Proximal Policy Optimization

MLP . . . . . Multi-Layer Perceptron

GRU . . . . . Gated Recurrent Unit

GAE . . . . . Generalized Advantage Estimation

REINFORCE . REward Increment = Non-negative Factor $\times$ Offset Reinforcement $\times$ Characteristic Eligibility

i.i.d. . . . . . Independently and Identically Distributed

1 Introduction

Global urbanization and the growing number of vehicles are exerting significant pressure on urban transportation networks. Traffic congestion has escalated from a localized inconvenience to a global crisis with severe economic, environmental, and public health consequences. Annually, urban areas incur billions of dollars in lost productivity due to transit delays. At the same time, emissions from idling vehicles, specifically CO_2 and NO_x, substantially degrade urban air quality and contribute directly to the broader climate crisis.

Historically, urban congestion was managed primarily through infrastructure expansion. Presently, spatial constraints and prohibitive costs render this approach unfeasible in dense urban environments. Consequently, a paradigm shift toward intelligent, data-driven infrastructure management is required.

This thesis investigates the application of Artificial Intelligence (AI) to optimize urban traffic flow via autonomous Traffic Signal Control (TSC). We hypothesize that Reinforcement Learning (RL) [21] agents can significantly outperform traditional, rule-based traffic signal systems. By leveraging real-time data and anticipating how congestion spreads to neighboring intersections, these intelligent agents can achieve a much higher degree of network-wide coordination.

To address these challenges, we introduce AnyLight, a novel Multi-Agent Reinforcement Learning (MARL) [31] framework designed for large-scale, heterogeneous traffic management. AnyLight is specifically developed to overcome the scalability and flexibility limitations of existing models. For instance, while Arguello et al. [1] addressed heterogeneity by training independent networks for each intersection, this strategy results in a linear increase in parameter count as the network scales and prevents knowledge sharing between agents. Furthermore, their image-based state representations are less flexible than movement-centric alternatives. In contrast, AnyLight utilizes a movement-centric paradigm and a "Universal Weights" architecture. This approach ensures a constant parameter count regardless of the network's scale, facilitates a common coordination protocol, and offers greater adaptability than the rigid topologies required by models such as CoLight [27] or PressLight [26]. The primary contributions of this work are as follows:

- **Movement-Centric Representation:**
  We propose a unified mapping of intersection dynamics that transitions from an intersection-centric to a movement-centric representation. This methodology eliminates the need for topology-specific architectural modifications, enabling a single model to process three-arm, four-arm, and asymmetrical intersections without structural changes.

- **Universal Parameter Sharing:**
  We implement a globally shared parameter space where a single neural network controls all traffic signal agents across the entire network. By enforcing a common coordination protocol, this strategy significantly mitigates multi-agent non-stationarity while maintaining a constant parameter count, regardless of the network scale.

- **Collaborative Reward Structure:**
  We integrate a collaborative reward system that explicitly encourages neighbor-aware traffic flow optimization. By combining this reward structure with Centralized Training and Decentralized Execution (CTDE), the agents learn a robust, implicit coordination protocol that substantially reduces network-wide intersection delay and manages high-density traffic more effectively than isolated control policies.

1.1 Thesis Outline

The remaining part of this thesis is structured as follows:

- **Chapter 2: The Traffic Signal Control Problem.**
  This chapter defines the fundamental challenges of urban traffic management and reviews the evolution of signal control, from fixed-time configurations to modern adaptive systems.
- **Chapter 3: Theoretical Foundations of Reinforcement Learning.**
  This chapter details the mathematical foundations of Reinforcement Learning. It focuses on Markov Decision Processes (MDPs) [4] and the unique challenges of training multiple agents in a shared network.
- **Chapter 4: Problem Formalization.**
  This chapter details the modeling of the traffic environment. It explains the integration with the SUMO [14] simulator and the specific design of the state, action, and reward spaces for the agents.
- **Chapter 5: AnyLight: A Generalizable Architecture and Training Strategy.**
  This chapter presents the core solution. It details the neural network architecture and the algorithmic strategies used to ensure the AI trains stably and effectively across diverse intersections.
- **Chapter 6: Experimental Framework and Simulation Scenarios.**
  This chapter outlines how we test the model, including the design of both synthetic and real-world traffic networks used for benchmarking.
- **Chapter 7: Results and Comparative Evaluation.**
  This chapter provides a quantitative analysis of the AI's performance, comparing its global efficiency against classical traffic control baselines.
- **Chapter 8: Conclusions and Future Work.**
  This chapter summarizes our findings, discusses the practical steps needed to deploy this system in the real world, and highlights promising directions for future research.

2

The Traffic Signal Control Problem

Traffic Signal Control (TSC) is a critical part of managing urban infrastructure. At its core, the TSC problem revolves around determining the best strategy for managing traffic lights to maximize the efficiency of the entire network. This involves dynamically adjusting the timing, phases, and duration of traffic signals to alleviate congestion, shorten vehicle waiting times, enhance road capacity, and improve overall road safety. As cities grow and vehicle ownership increases, existing transportation networks are reaching their capacity limits, resulting in chronic congestion. This inefficiency has serious economic and social consequences. Beyond the substantial loss in global productivity, the environmental impact is severe: vehicles idling in congested traffic release significant amounts of greenhouse gases, degrading public health and contributing to climate change.

2.1 The Shift in Urban Management

Historically, the primary strategy for reducing congestion was the construction of additional road infrastructure. Today, however, adding new infrastructure in dense cities is heavily restricted by a lack of physical space, high construction costs, and the issue of *induced demand*, where new roads quickly fill up with new traffic. Because of this, optimizing TSC is widely recognized as a pivotal, highly cost-effective measure. It improves transportation efficiency without the need to build expensive new roads or alter existing

physical infrastructure. The main goal of modern TSC is to intelligently schedule traffic light phases, aiming to minimize overall travel times while maximizing the number of vehicles that can smoothly pass through the network. The problem remains inherently challenging, however, because controllers must account for complex intersection layouts, highly dynamic traffic flow, and the need to balance multiple conflicting objectives.

2.2 Limitations of Classical Control Theory

The main difficulty with TSC is that traffic flow is highly complex and unpredictable. A city grid is not a collection of independent intersections; it is an interconnected network in which a phase change at one intersection propagates its effects to the surrounding ones. Over the years, engineers have addressed this problem through several conventional methods:

- **Fixed-Time Control:**
  Methods like the Webster algorithm [24] use strict, pre-calculated timing schedules based on historical traffic data. While easy to implement, these static configurations fundamentally struggle to adapt to the highly dynamic and unpredictable nature of modern urban traffic.
- **Actuated and Adaptive Systems:**
  Systems like SCATS [15] and SCOOT [10] use road sensors to adjust green times based on current traffic. However, they rely on centralized designs and manual, pre-defined rules that generalize poorly across the diverse layouts of a large city.
- **Pressure-based Heuristics:**
  Advanced algorithms, such as MaxPressure [22], stabilize traffic by balancing queue lengths between connected roads. While mathematically sound, these methods often prioritize the immediate clearance of local queues over the long-term flow of the entire network.

2.3 TSC as a Stochastic Optimization Problem

A further difficulty of traffic control is its unpredictability. A reliable system must handle conditions ranging from recurrent inflow surges to sudden accidents, emergency vehicles,

and adverse weather.

In dense areas, optimizing each intersection in isolation is insufficient. If one traffic light releases a large platoon of vehicles without coordinating with the next one, congestion is merely displaced downstream, a phenomenon known as *spillback*. Given this strong coupling between intersections, an effective system requires traffic lights that operate locally while cooperating with their neighbors.

To overcome these hurdles, researchers have increasingly focused on applying advanced Artificial Intelligence models. Because traffic control involves making a series of decisions over time and learning from delayed feedback, it naturally fits the mathematical model of a **Markov Decision Process (MDP)** [4]. By treating TSC as a Reinforcement Learning (RL) [21] problem, agents can learn the optimal signal control strategy through continuous trial-and-error and reward feedback. This removes the need for engineers to manually model the complex dynamics of the intersection network.

Furthermore, by modeling TSC as a multi-agent problem where each intersection acts as an individual agent, Multi-Agent Reinforcement Learning (MARL) [31] enables true network-wide coordination. This allows the traffic system to make decisions based on the real-time state of the intersections, achieving global optimization without manual intervention. This thesis investigates the transition from static, rule-based methods to this MARL approach as a foundation for more adaptive urban traffic management.

3

Theoretical Foundations of Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of machine learning inspired by the fundamental ways humans and animals learn. At its core, it is based on the concept of trial and error. By its very nature, RL is a highly interdisciplinary field, inheriting essential mathematical and conceptual frameworks from optimization, general mathematics, neuroscience, and psychology.

Unlike supervised learning, which relies heavily on a static, labeled dataset of input-output pairs to minimize an empirical risk, RL focuses on training an autonomous agent to make sequential decisions based on continuous feedback derived from its interactions with an active environment. In a supervised setting, a model is trained to imitate the data it is fed. However, RL aims at discovering optimal behaviors that can potentially surpass human knowledge, making mere imitation insufficient for solving complex, high-dimensional problems.

The central hypothesis underlying the RL paradigm is that all notions of goals and purposes can be mathematically formalized as the maximization of the expected value of a cumulative scalar signal, which is defined as the return. Consequently, the main focus of RL is the resolution of sequential decision-making problems. This framework allows an

agent to navigate the delicate balance between exploring new, potentially better strategies and exploiting the knowledge it has already acquired to maximize its long-term return.

3.1 The Reinforcement Learning Framework

The core interaction in this paradigm occurs between an agent and its environment. Specifically, an agent observes at time t the state S_t from the environment, from which it decides to take action A_t. Following this action, the agent transitions to state S_{t+1} and receives a scalar reward R_{t+1}.

The mechanism dictating the agent's decision-making process is called a policy. The policy can be either deterministic, formalized as $A_t = \pi(S_t)$, or stochastic, formalized as $A_t \sim \pi(S_t)$. In simple environments, this policy can be represented by a simple lookup table, whereas in highly complex scenarios, it must be approximated by function approximators such as neural networks.

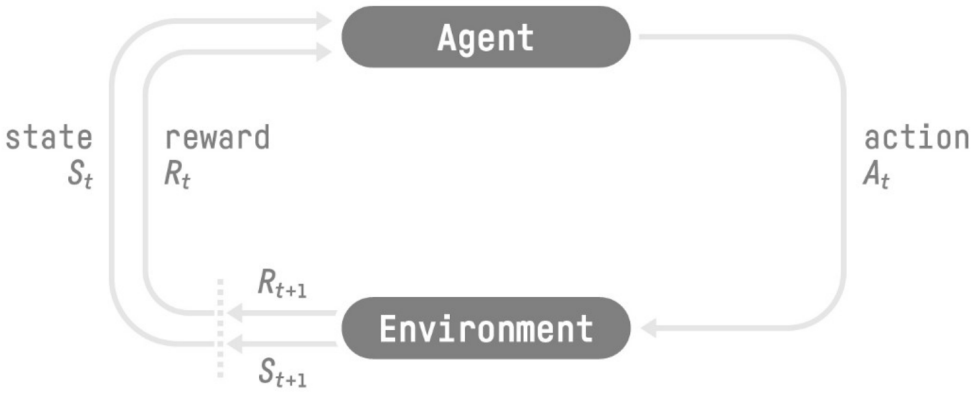


Figure 3.1: The Reinforcement Learning interaction loop between an agent and its environment.

Given that R_t is a scalar reward indicating how good the last action was, and given that past actions inherently influence future states, the agent cannot solely rely on maximizing the immediate reward greedily. The agent must consider how its current decision will

impact its future. Therefore, the problem is formulated as an optimization task where the agent is asked to maximize the cumulative reward, called the return. The basic return is defined as:

$$G_t = \sum_{i=t+1}^{\infty} R_i \tag{3.1}$$

However, in continuous or infinite-horizon tasks, this formulation most likely introduces infinite rewards. To resolve this and allow the infinite sum to converge while preserving a notion of future reward, a discount parameter γ is introduced:

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \tag{3.2}$$

The discount factor $\gamma \in [0, 1]$ dictates the agent's foresight. If $\gamma = 0$, the agent becomes entirely greedy; conversely, as $\gamma \to 1$, the agent cares more and more about its future.

3.2 Markov Decision Processes (MDPs)

The standard framework for modeling RL problems is the Markov Decision Process (MDP), which provides a rigorous mathematical formalization of sequential decision-making. An MDP extends basic stochastic models by introducing the notion of an action set $\mathcal{A}$. This fundamental addition requires transition probabilities and the reward function to be conditioned not only on the current state but also on the chosen action. Thus, an MDP is formally defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

- $\mathcal{S}$: A set of states representing the environment's configuration.
- $\mathcal{A}$: A set of actions available to the agent.
- $\mathcal{P}$: The transition probability $\mathcal{P}^a_{ss'} = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$, defining the likelihood of moving to state s' after taking action a in state s.
- $\mathcal{R}$: The reward function $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$, providing the expected immediate feedback for an action.
- γ: The discount factor $\gamma \in [0, 1]$, determining the importance of future rewards.

3.3 Value Functions and Bellman Equations

To evaluate the desirability of a state under a specific policy, RL introduces the State-Value function $V(s)$, defined as the expected return starting from state s. Using the recursive property of the return, this function can be rephrased via the Bellman expectation equation [4]:

$$V(s) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \tag{3.3}$$

While $V(s)$ evaluates a state, it does not explicitly evaluate the specific actions available in that state. Therefore, the Action-Value function $Q(s, a)$ is introduced, defined as the expected return of taking action a in state s and following the current policy thereafter:

$$Q(s, a) = \mathbb{E}[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \tag{3.4}$$

The ultimate objective of the RL process is to learn these functions accurately. Once an accurate estimation of the optimal Q function is achieved, the optimal policy can be derived by selecting the action that maximizes this value.

3.4 Deep Reinforcement Learning (DRL)

Before the advent of modern neural architectures, the estimation of $V(s)$ and $Q(s, a)$ relied heavily on tabular methods. However, tabular methods are fundamentally unsuited for real-world applications such as urban traffic control. The state space in a traffic network is astronomically large, if not entirely continuous. Maintaining a lookup matrix becomes computationally impossible, a limitation famously known as the curse of dimensionality [4].

To overcome this, the RL framework shifts to function approximation, giving rise to Deep Reinforcement Learning (DRL). By utilizing deep neural networks to parameterize policies and value functions ($V_w(s) \approx V(s)$, $Q_\theta(s, a) \approx Q(s, a)$), agents can automatically extract useful features from high-dimensional, unstructured observation spaces.

Depending on how the neural network is specifically utilized to solve the underlying MDP,

modern DRL algorithms are initially categorized into two foundational architectures before converging on hybrid solutions.

3.5 Value-Based vs. Policy-Based Methods

Value-based methods aim to find the optimal policy indirectly by estimating the expected return of state-action pairs. The foundational breakthrough in this category is the Deep Q-Network (DQN) [16]. While DQN represents a monumental leap in RL capabilities, it possesses several inherent drawbacks. Most notably, the algorithm is fundamentally designed for discrete action spaces, making it unwieldy for continuous control problems, and it often suffers from an overestimation bias of the true Q values.

Conversely, policy-based methods shift the optimization focus directly onto the policy itself. In this paradigm, the policy is explicitly parameterized by a neural network, outputting a probability distribution over the available action space. The most fundamental algorithm implementing this is REINFORCE [28]. While natively handling stochastic policies, pure policy gradient methods possess critical limitations. The reliance on full trajectory Monte Carlo returns introduces excessively high variance into the gradient estimates, leading to severe sample inefficiency and unstable training dynamics.

The theoretical necessity to balance the sample efficiency of value-based methods with the stability of policy-based optimization leads directly to the development of the Actor-Critic architecture.

3.6 Actor-Critic Methods

To bridge the theoretical and practical gaps between the previous two paradigms, the Actor-Critic architecture merges their respective strengths [3]. In this framework, the learning agent is bifurcated into two distinct but cooperative neural networks.

The first component, the ”Actor”, is responsible for maintaining and updating the policy distribution $\pi_\theta(a|s)$, essentially dictating how the agent behaves in a given state. The second component, the ”Critic”, evaluates the quality of the Actor’s decisions by estimating a value function.

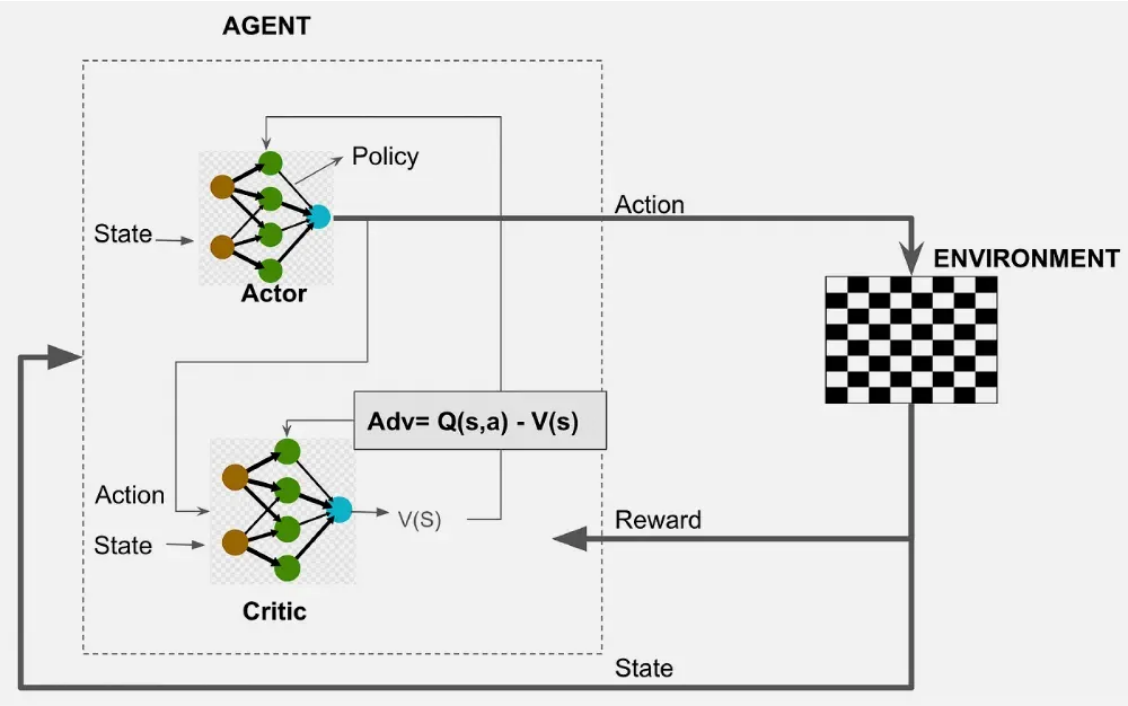


Figure 3.2: The Actor-Critic architecture, with the Actor dictating agent behavior and the Critic evaluating action quality.

In a basic Actor-Critic setup, the Critic might estimate the action-value function $Q_w(s, a)$, and the Actor updates its policy weights in the direction that maximizes this Q value. The fundamental policy gradient update rule then becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi_\theta(a|s) Q_w(s, a)] \tag{3.5}$$

While this approach successfully replaces the high-variance Monte Carlo returns used in pure policy algorithms with a learned, lower-variance approximation, the absolute Q values can still fluctuate significantly across different states. This fluctuation can introduce numerical instability during the optimization process.

3.6.1 The Advantage Function

To further stabilize training and rigorously reduce the variance of the gradient estimates, a fundamental mathematical trick is introduced: the baseline.

Instead of evaluating an action based on its absolute expected return, it is significantly more effective to evaluate how much better an action is compared to the average expected return of that exact state.

This concept is formalized as the Advantage function, denoted as $A(s, a)$. It is defined as the difference between the action-value function and the state-value function:

$$A(s, a) = Q(s, a) - V(s) \tag{3.6}$$

The intuition is straightforward. If $A(s, a) > 0$, the chosen action yielded a better return than the state's average, signaling the Actor to increase the probability of taking that action in the future. Conversely, if $A(s, a) < 0$, the action performed worse than expected, and its probability should be penalized.

Implementing this directly would require the Critic to learn and maintain two separate neural networks, one for $Q_w(s, a)$ and one for $V_v(s)$, which is computationally inefficient and introduces compounded approximation errors. Fortunately, by leveraging the Bellman expectation equations, the Q function can be approximated using the immediate reward and the discounted value of the subsequent state:

$$Q(s, a) \approx R_{t+1} + \gamma V(S_{t+1}) \tag{3.7}$$

Substituting this approximation back into the Advantage formulation yields the Temporal Difference (TD) error:

$$A_t \approx R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t) \tag{3.8}$$

This elegant reformulation, which forms the basis of the Advantage Actor-Critic (A2C)

architecture, allows the Critic to maintain only a single neural network that estimates the state-value function $V_w(s)$.

In practice, the Critic network parameters w are updated to minimize the mean squared TD error between its predictions and the observed returns, while the Actor network parameters θ are updated using the computed Advantage A_t. This structured feedback loop provides a highly stable baseline, successfully mitigating variance issues while naturally handling continuous or high-dimensional action spaces.

3.6.2 Trust Region Policy Optimization (TRPO)

While Advantage Actor-Critic methods provide a robust foundation, standard policy gradient updates remain highly sensitive to the chosen learning rate. A step size that is too small results in substantially slow convergence, whereas a step size that is too large can push the policy into a region of the parameter space where it performs poorly. Because the data collected in reinforcement learning depends directly on the current policy, a severe drop in policy performance leads to the collection of low-quality data, making recovery nearly impossible. This phenomenon is known as a catastrophic performance collapse.

TRPO [18] was introduced to solve this exact issue. TRPO ensures monotonic policy improvement by restricting the parameter update such that the new policy does not deviate too far from the old policy. This is achieved by maximizing a surrogate objective function subject to a strict Kullback-Leibler (KL) divergence constraint:

$$\max_{\theta} \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t \right] \text{ subject to } \mathbb{E}_t[D_{KL}(\pi_{\theta_{old}}(\cdot|s_t)||\pi_\theta(\cdot|s_t))] \leq \delta \tag{3.9}$$

Here, δ represents a hyperparameter that dictates the size of the trust region.

The primary drawback of TRPO is the computational cost of enforcing this constraint. The KL divergence between two parameterized policies cannot be computed in closed form and must be approximated from sampled states, while the constrained optimization itself relies on expensive second-order methods that scale poorly to large state and action spaces. Moreover, the quality of each update depends directly on the accuracy of this approximation: an underestimated divergence permits overly aggressive steps that can

destabilize the policy, whereas an overestimated divergence produces unnecessarily conservative updates that slow down convergence. These practical limitations motivated the development of a simpler, first-order alternative.

3.6.3 Proximal Policy Optimization (PPO)

To retain the stability guarantees of TRPO while eliminating its prohibitive computational complexity, PPO was developed. PPO stands as the current state-of-the-art Actor-Critic algorithm.

PPO [20] simplifies the optimization landscape by substituting the hard KL divergence constraint with a clipped surrogate objective function.

Let $r_t(\theta)$ denote the probability ratio between the updated policy and the old policy:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{3.10}$$

Instead of computing second-order derivatives, PPO directly bounds the policy update by maximizing the following objective function:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{3.11}$$

The clipping hyperparameter ϵ restricts the probability ratio within the interval $[1-\epsilon, 1+\epsilon]$. The $\min$ operator effectively creates a pessimistic bound on the policy update. If an action yields a positive advantage, its probability is increased, but the update is abruptly halted if the ratio exceeds $1+\epsilon$. Conversely, if an action yields a negative advantage, the update is halted once the ratio falls below $1-\epsilon$.

By preventing destructively large policy shifts, PPO achieves exceptional training stability and sample efficiency using standard first-order optimizers like Adam [11]. This elegant formulation grants PPO the robustness required to navigate the severe non-stationarity of a Multi-Agent Traffic Signal Control environment, allowing multiple intersections to learn cooperative phase optimization policies reliably and efficiently.

Algorithm 3.1: Proximal Policy Optimization (PPO) with Clipping

Input:

Initial policy parameters θ_0, initial value function parameters ϕ_0;

Initialize:

Step sizes α_{policy}, α_{value}, clipping threshold ϵ;

1. **for** iteration = 1, 2, …**do**

   (a) Collect a set of trajectories using the current policy π_θ;

   (b) Compute advantages A_t for each time step t;

   (c) **for** mini-batch = 1, 2, …, num_mini_batches **do**

      i. Compute probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ for each time step t;

      ii. Compute the clipped surrogate objective:

$$L^{PPO}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta) \cdot A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \cdot A_t)]$$

      iii. Update the policy parameters using gradient ascent:

$$\theta_{new} = \theta_{old} + \alpha_{policy} \cdot \nabla_\theta L^{PPO}(\theta)$$

      iv. Update the value function parameters using gradient descent:

$$\phi_{new} = \phi_{old} - \alpha_{value} \cdot \nabla_\phi \left(\frac{1}{N} \sum_{t=1}^{N} (V_\phi(s_t) - R_t)^2 \right)$$

   (d) Update the old policy parameters: $\theta_{old} \leftarrow \theta_{new}$;

3.7 Online and Offline Reinforcement Learning

Beyond the architectural categorization of DRL algorithms into Value-Based, Policy-Based, and Actor-Critic methods, a critical distinction lies in their operational paradigm regarding data collection and utilization. As these high-capacity models are trained, understanding how they acquire and process experience becomes paramount. This leads to the fundamental distinction between online and offline learning, a factor that profoundly impacts the practical applicability, sample efficiency, and stability of the training process, often interacting closely with the chosen DRL architecture.

3.7.1 Online Reinforcement Learning

In the online learning paradigm, the agent learns through active and continuous interaction with a live environment. The learning process is characterized by an iterative loop of observation, action, and immediate feedback.

The defining feature of online RL is the requirement to manage the exploration-exploitation trade-off in real-time. The agent must purposefully take suboptimal actions to discover better strategies while simultaneously attempting to maximize its current return. Algorithms such as State-Action-Reward-State-Action (SARSA) [21] and PPO are typically employed in this setting.

A critical limitation of online methods is their low sample efficiency. Since the agent updates its policy based on the data it actively collects, it often requires millions of interactions with the environment to converge to an optimal behavior. This high sample complexity necessitates either a safe physical environment or a high-fidelity simulator, as early learning stages are marked by high variance and unpredictable behavior.

3.7.2 Offline Reinforcement Learning

Offline reinforcement learning, also known as Batch Reinforcement Learning, decouples the learning process from environmental interaction. In this setting, the agent is trained using a static, fixed dataset of historical transitions previously collected by a different policy, such as a human operator or a heuristic controller.

Unlike online methods, offline RL offers significantly higher sample efficiency in terms of environmental interaction, as it leverages existing data without requiring any additional steps in the environment during the training phase. This makes it a powerful paradigm for scenarios where data collection is expensive, slow, or risky.

However, the primary challenge in offline RL is the distributional shift, where the agent might overestimate the value of actions that were never explored in the static dataset. Since the agent cannot interact with the environment to correct these errors, it must employ conservative optimization techniques to ensure the learned policy does not deviate dangerously from the behavior captured in the data. Algorithms like Conservative Q-Learning (CQL) [12] and Batch-Constrained Q-learning (BCQ) [8] are specifically designed to address these constraints.

3.8 Multi-Agent Reinforcement Learning (MARL)

The standard RL framework inherently assumes a single agent interacting with a static or purely stochastic environment. However, many complex real-world problems are fundamentally multi-agent in nature, consisting of multiple interacting entities that must learn and act simultaneously in a shared environment. MARL extends the classical RL paradigm to mathematically accommodate these distributed systems.

In a multi-agent environment, it is theoretically possible to employ a single, centralized global agent that observes the entire joint state and dictates a joint action for every entity. Yet, in large-scale systems with numerous actors, the joint state and action spaces grow exponentially with the number of agents. This phenomenon makes relying on a single central DRL controller computationally intractable and structurally fragile. To resolve this curse of dimensionality, practical MARL applications decentralize the decision-making logic, deploying multiple distributed agents where each manages a localized subsystem.

While decentralization drastically reduces the dimensionality of the local state and action spaces, it introduces a profound theoretical challenge known as environment non-stationarity. In a single-agent MDP, the transition probabilities $\mathbb{P}(s'|s,a)$ are fixed. In a distributed MARL setting, the transition dynamics from the perspective of any individual agent are constantly shifting because the adjacent agents are simultaneously updating their

own policies. Thus, a state-action pair that yielded a high reward in early training epochs might yield a drastically different outcome later, purely because a neighboring agent altered its behavior. Navigating this non-stationarity to achieve cooperative equilibrium is the defining challenge of distributed multi-agent systems.

3.8.1 Execution and Training Paradigms

To solve MDPs and address the compounded issues of non-stationarity, MARL algorithms are typically categorized by how information is shared during the training phase versus the execution phase. There are three primary paradigms:

- **Centralized Training, Centralized Execution (CTCE):**
  A single global policy maps the joint global state to a joint action for all agents. While this completely eliminates the non-stationarity problem by treating the system as a massive single-agent MDP, it suffers exponentially from the curse of dimensionality. Furthermore, it requires continuous, zero-latency communication across the entire physical network during execution, making it highly impractical and fragile for real-world traffic control.

- **Decentralized Training, Decentralized Execution (DTDE):**
  Each agent is treated as a completely independent learner, treating other agents merely as part of the stochastic environment. While highly scalable and easy to deploy on local intersection hardware, this paradigm suffers massively from non-stationarity. Since agents do not share information or intent during training, achieving cooperative behavior is notoriously difficult, and the training process often diverges or settles into highly suboptimal local minima.

- **Centralized Training, Decentralized Execution (CTDE):**
  This paradigm represents the state-of-the-art compromise for multi-agent systems and fits perfectly within the Actor-Critic architecture. During the simulated offline training phase, the algorithm is allowed access to the global state S_t and the joint actions of all agents. This privileged global information is fed to a centralized "Critic" network to accurately estimate the value function, effectively stabilizing the non-stationarity and providing a reliable baseline for the Advantage function. However, the "Actor" networks are strictly constrained to map only local observations o_t^i to local actions a_t^i.

4

Problem Formalization

In this chapter, we formulate the Multi-Agent Traffic Signal Control (MATSC) problem, mapping Reinforcement Learning (RL) theory to urban transportation environments. First, we introduce the traffic engineering terminology required to describe signalized intersections. This vocabulary defines the environment and the operational constraints of the learning agents.

We then introduce the Simulation of Urban MObility (SUMO) [14] framework, which serves as the simulation environment for our multi-agent formulation.

Subsequently, we map the MATSC problem onto the Multi-Agent Reinforcement Learning (MARL) framework. We define the three core components of the decision-making process: states, actions, and rewards. These components govern the learning dynamics and policy performance; thus, their design is critical to the system's effectiveness.

4.1 Fundamental Traffic Terminology

In this dissertation, we study generalizable network-wide Adaptive Traffic Signal Control (ATSC). The proposed method is robust across both heterogeneous networks, where intersections feature diverse topological structures, and homogeneous networks, where all intersections share the same layout.

To simulate the dynamics of these urban grids, we represent the traffic network using four core concepts:

- Incoming and outgoing lanes
- Traffic movements and their associated status
- Traffic signal phases
- Traffic agents

These components are detailed below.

Incoming and Outgoing Lanes
Incoming lanes direct traffic toward an intersection, while outgoing lanes channel vehicles away from it. A road connected to an intersection typically consists of multiple lanes to accommodate different traffic directions and volumes. We define $\mathcal{L}_{in}$ and $\mathcal{L}_{out}$ as the sets of incoming and outgoing lanes for a given intersection. These sets are used to construct the state space.

Traffic Movements and Movement Status
A traffic movement is a trajectory connecting an incoming lane $l_{in} \in \mathcal{L}_{in}$ to an outgoing lane $l_{out} \in \mathcal{L}_{out}$, denoted as $m(l_{in}, l_{out})$. For simplicity, we refer to a movement as m and omit the lane indices.

Each movement has a binary status variable A_m set by the traffic controller. If the movement is permitted, $A_m = 1$; if it is prohibited (e.g., by a red light), $A_m = 0$. The active movements at a given time define the state of the traffic signal.

Traffic Signal Phases
Traffic signal phases manage flow at an intersection by grouping non-conflicting movements, allowing them to proceed simultaneously without collision risks.

For an intersection i, the set of all movements $\mathcal{M}_i$ is the union of the movements active in each phase $p \in \mathcal{P}$:

$$\mathcal{M}_i = \bigcup_{p \in \mathcal{P}} M_{i,p} \tag{4.1}$$

where $\mathcal{P}$ is the set of operational phases for the intersection, and $M_{i,p}$ is the subset of active movements (where $A_m = 1$) during phase p. By selecting and sequencing these phases, the reinforcement learning agent determines which movements receive the right-of-way.

Traffic Agents
Traffic agents regulate traffic flow at intersections by dynamically selecting signal phases. To represent the network, we model the urban grid as a multi-agent system using a graph structure $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

Here, $\mathcal{V} = \{1, \ldots, N\}$ represents the set of N agents, each mapped to a specific intersection, and $\mathcal{E}$ represents the road segments connecting them.

4.2 The Simulation Framework: SUMO and TraCI

To simulate Multi-Agent Traffic Signal Control, we use the Simulation of Urban MObility (SUMO) [14]. SUMO is an open-source, microscopic, and continuous-space traffic simulation suite designed to handle large-scale networks and routing.

A microscopic simulator is used because, unlike macroscopic models that treat traffic as an aggregate fluid, it computes the dynamics of individual vehicles (e.g., car-following, lane-changing, and acceleration profiles). This detail produces emergent traffic phenomena such as congestion shockwaves and spillbacks, providing the granular data (e.g., speed and coordinates) required for reinforcement learning agents to perceive intersection states.

Interaction between the agents and SUMO is managed via the Traffic Control Interface (TraCI). Operating on a client-server architecture, TraCI pauses the simulation at discrete time steps, allowing Python-based learning algorithms to interact with the SUMO engine to:

- **Retrieve Observations:**

Extract real-time traffic data, such as queue lengths, halting vehicle counts, and occupancies, at the beginning of each decision step.

- **Execute Actions:**

  Send phase commands to dynamically override default traffic signal logic.

In SUMO, traffic light states are represented as character strings, where each character sets the right-of-way for a specific movement:

- **'G' (Priority Green):**

  Vehicles have the right-of-way and proceed without yielding.

- **'g' (Yielding Green):**

  Vehicles proceed but must yield to conflicting priority flows.

- **'r', 'y', 'u', 'o' (Non-Green states):**

  Represent red, yellow, or off states, where vehicles must halt or yield.

For example, consider the four-way intersection in Figure 4.1. If each of the four approaches (North, South, East, West) has three movements (right-turn, straight, left-turn), the intersection controls 12 distinct movements. A single signal phase is thus defined by a 12-character string.

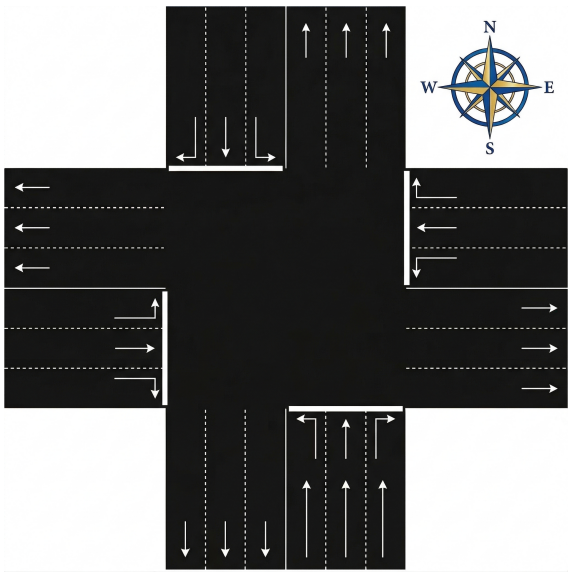


Figure 4.1: A standard four-way intersection with three movements per approach (left-turn, straight, right-turn), totaling 12 controlled movements.

The string `"GGrrrrGGrrrr"` indicates that the straight and right-turn movements for the North and South approaches have a priority green light (`'G'`), while East, West, and left-turn movements have a red light (`'r'`).

Similarly, a phase allowing left turns for the East and West approaches while yielding to other flows is encoded as `"rrrrgGrrrrgG"`. TraCI extracts these strings dynamically, and the environment wrapper translates them into numerical matrices for the neural networks.

4.3 State Space

The state space $\mathcal{S}$ (or observation space $\mathcal{O}$ in a decentralized setting) provides each agent with a detailed view of the traffic dynamics at its intersection. We adopt a movement-based observation approach rather than the intersection-centric methods used in earlier models. As noted in PressLight [26], high-dimensional or image-based states often cause computational inefficiency and slow convergence. By decomposing the intersection into standardized movements, we establish a unified representation that is consistent and in-

terpretable across varying configurations. This structure enables a **semantic alignment** of features, ensuring that equivalent movement types across different intersections share a common representation. This alignment helps mitigate the "conflicting learning" problem identified in CoLight [27] and CityLight [30], where parameter sharing is hindered by topological variations. By linking traffic conditions directly to signal phases, the model generalizes across heterogeneous networks.

Traffic State Vector

In this framework, environmental data is decomposed into traffic movements $m \in \mathcal{M}_i$. The traffic state vector $S_i(t)$ for intersection i at time t consists of the localized state vectors for all movements, represented as a concatenated matrix:

$$S_i(t) = [S_i^m(t) \mid m \in \mathcal{M}_i] \in \mathbb{R}^{|\mathcal{M}_i| \times 8} \tag{4.2}$$

For each movement m connecting an incoming lane l_{in} to an outgoing lane l_{out}, the localized observation vector $S_i^m(t)$ contains 8 features. These features are normalized to $[0, 1]$ to stabilize training and are extracted within a 50-meter detection zone upstream and downstream of the intersection:

$$S_i^m(t) = [A_m, Q_{l_{in}}, Q_{l_{out}}, N_{l_{in}}, N_{l_{out}}, O_{l_{in}}, O_{l_{out}}, C_{l_{out}}] \in \mathbb{R}^8 \tag{4.3}$$

The components of this feature vector are defined as follows:

1. A_m **(Signal Status):** The active status of the signal for movement m, represented as a discrete value: 1.0 for priority green, 0.5 for yielding green, and 0.0 for red or yellow states.
2. $Q_{l_{in}}$ **(Stopped Vehicles, In-Lane):** Queue length on the incoming lane l_{in} (vehicles with speed < 0.1 m/s within the 50m detection zone), normalized by a capacity constant of 20.
3. $Q_{l_{out}}$ **(Stopped Vehicles, Out-Lane):** Halted vehicles on the outgoing lane l_{out} (speed < 0.1 m/s within 50m after the junction), normalized by 20. This allows the agent to detect downstream spillback.
4. $N_{l_{in}}$ **(Moving Vehicles, In-Lane):** Active vehicles on incoming lane l_{in} (speed ≥ 0.1 m/s), normalized by 20.

5. $N_{l_{out}}$ **(Moving Vehicles, Out-Lane):** Active vehicles on outgoing lane l_{out} (speed ≥ 0.1 m/s), normalized by 20.

6. $O_{l_{in}}$ **(Occupancy, In-Lane):** Cumulative vehicle length on incoming lane l_{in} divided by the detection distance (50m), measuring traffic density.

7. $O_{l_{out}}$ **(Occupancy, Out-Lane):** Cumulative vehicle length on outgoing lane l_{out} normalized by 50m.

8. $C_{l_{out}}$ **(Downstream State):** The aggregate control status of the downstream lane, informing the agent whether the outgoing lane is controlled by a neighboring intersection.

By aggregating these vectors for all intersection movements, the agent constructs the state observation $S_i(t)$. This representation allows the policy to distinguish between competing flows and adapt to local and downstream constraints.

Traffic Phase Vector

Alongside the dynamic state vector $S_i(t)$, the agent uses a static Traffic Phase Vector to represent the intersection's structure. This vector maps the discrete actions (phase choices) to the physical traffic movements they control.

The traffic phase matrix G_i for intersection i consists of time-invariant phase vectors G_i^p, mapping available actions (rows) to movements (columns):

$$G_i = [G_i^p \mid p \in \mathcal{P}_i] \in \mathbb{R}^{|\mathcal{P}_i| \times |\mathcal{M}_i|} \tag{4.4}$$

where $\mathcal{P}_i$ is the set of valid phases (actions), and $\mathcal{M}_i$ is the set of traffic movements. For phase $p \in \mathcal{P}_i$, the vector $G_i^p \in \mathbb{R}^{|\mathcal{M}_i|}$ specifies the status of each movement.

The elements of this vector are populated as follows:

$$G_{i,m}^p = \begin{cases} 1.0 & \text{if movement } m \text{ has a priority green signal (G) in phase } p \\ 0.5 & \text{if movement } m \text{ has a yielding green signal (g) in phase } p \\ 0.0 & \text{if movement } m \text{ faces a red (r) or yellow (y) signal in phase } p \end{cases} \tag{4.5}$$

The phase matrix is concatenated with the state observation. This provides physical context to the discrete action space, which in standard RL is often opaque to the agent.

Neighbor Coordination and Downstream Information

To support decentralized coordination without explicit communication overhead, we augment the observation space with neighbor-specific features. This includes static topological links and dynamic coordination signals from adjacent intersections.

Neighbor information comprises three components:

1. **Neighbor Topology (Static Identification):**

   During environment initialization, the system identifies physical connections between RL-controlled intersections by comparing their incoming and outgoing lanes. A static flag indicates whether a movement connects to a neighboring agent, boundary, or non-agent junction.

2. **Neighbor Activation Status (Dynamic Signal):**

   The agent receives the right-of-way status of downstream intersections. For each outgoing lane connecting to a neighbor, the system extracts the neighbor's current phase activation value (1.0 for priority green, 0.5 for yielding green, and 0.0 for red/yellow). This informs the agent if the downstream road is open.

3. **State Integration and Normalization:**

   Coordination signals and the static neighbor flag are appended to the movement-based observation. Neighbor action vectors are zero-padded to a fixed maximum dimension to ensure a uniform network input shape.

Neighbor information serves three purposes: it prevents gridlock by alerting agents to saturated exit paths, enables implicit synchronization of green waves, and provides boundary detection to distinguish between internal roads and boundary sinks.

The structural alignment of these vectors for learning across diverse topologies is detailed in Chapter 5.

4.4 Action Space

Following established methods in RL-based ATSC, we define the action space of each agent as a discrete set of collision-free traffic signal phases.

These phases ensure that only non-conflicting traffic movements receive the right-of-way simultaneously, preserving safety at the intersection. At each decision step t, the agent selects an action $a_t \in \mathcal{A}$ to activate a specific phase $p \in \mathcal{P}$. This phase is applied for a fixed duration Δt. The agent operates dynamically and is not restricted to a fixed cyclical sequence, allowing the policy to respond to fluctuations in traffic demand.

To simplify the learning process, the action space contains only valid green phases. Transitional yellow phases are managed by the environment logic rather than the agent. If the agent switches to a new phase, the system inserts a yellow clearance phase of a predefined duration before starting the new green phase. Conversely, if the agent selects the currently active phase, the yellow transition is bypassed, and the green phase is extended by Δt. This abstraction separates safety constraints from control optimization, allowing the model to focus on green phase selection.

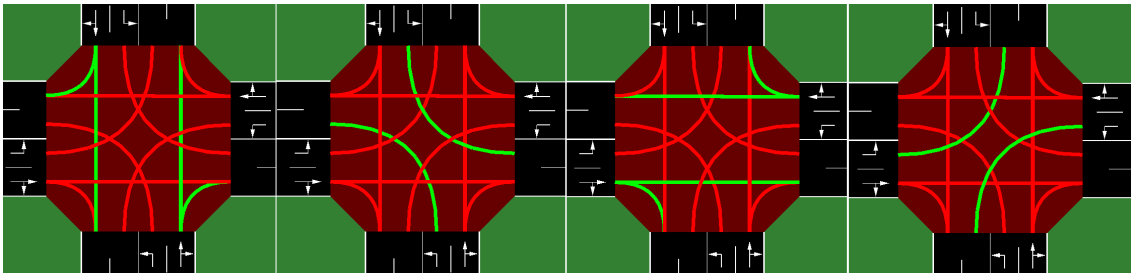


Figure 4.2: The discrete action space of an agent, representing the set of green phases in a four-way intersection.

In our experiments, the green phase duration Δt is set to 15 seconds and the yellow clearance interval to 5 seconds. The only exception is the Monaco scenario, where a 10-second green duration and a 3-second yellow interval are adopted to remain consistent with the original MA2C benchmark configuration [6], as detailed in Chapter 6.

4.5 Reward Function

The reward function provides the scalar feedback signal used to optimize the agent's policy. In traffic signal control, the primary global objective is to minimize network-wide congestion and reduce the Average Travel Time (ATT) of vehicles.

While ATT is the definitive metric of system performance, it is a delayed, macroscopic measurement. The travel time of any individual vehicle is influenced by downstream intersection policies, routing choices, and traffic injection rates. Consequently, using ATT directly as a step-by-step reward introduces temporal credit assignment issues and delayed feedback, hindering policy convergence.

To address this, we use a dense proxy reward that correlates with minimizing travel time. Under queuing theory, minimizing the number of vehicles in a system reduces the average wait time. Thus, the proxy reward R_t for each agent at step t is the negative sum of the average queue lengths on its incoming and outgoing lanes, expressed mathematically as:

$$R_t = -\left(\frac{1}{|\mathcal{L}_{in}|}\sum_{l_{in}\in\mathcal{L}_{in}} Q_{l_{in}} + \frac{1}{|\mathcal{L}_{out}|}\sum_{l_{out}\in\mathcal{L}_{out}} Q_{l_{out}}\right) \tag{4.6}$$

where $Q_{l_{in}}$ and $Q_{l_{out}}$ represent the queue lengths (number of halted vehicles) on incoming lane l_{in} and outgoing lane l_{out}, respectively. To maintain real-world applicability, these queue lengths are measured using simulated lane-area Induction Loop Detectors (ILDs) with a 50-meter range, installed on the road segments adjacent to the intersection. The resulting value is scaled by a constant factor of 0.1 to keep the reward magnitudes within a numerically stable range.

A key feature of this reward is its spatial coupling. By incorporating outgoing lane congestion, which represents the incoming lanes of downstream intersections, the reward links neighboring agent performance. This penalizes greedy policies that clear local queues by pushing traffic to downstream neighbors, encouraging cooperative behavior and optimizing overall travel times.

This formulation differs from common structures in the literature. Many MARL models, such as CoLight [27], GPLight [13], and HeteroLight [32], penalize only incoming

queue lengths. While simple, this approach can lead to sub-optimal states where agents push congestion downstream. Although the ”Pressure” metric in PressLight [26] considers both incoming and outgoing dynamics as a differential, our formulation prioritizes minimizing average local density. This incentivizes the agent to reduce the total number of vehicles within its influence, rather than just balancing incoming and outgoing flows.

5

AnyLight: A Generalizable Architecture and Training Strategy

The formalization detailed in Chapter 4 provides the mathematical foundation necessary to represent complex traffic dynamics as a Multi-Agent MDP. However, translating this theoretical formulation into a scalable, robust, and deployable DRL model introduces profound architectural challenges. This chapter presents the core technical contribution of this dissertation: the AnyLight architecture, a unified neural network framework explicitly engineered for generalizable TSC across highly heterogeneous urban networks.

We begin by rigorously defining the problem of network heterogeneity, illustrating why traditional, intersection-specific neural network deployments fail to scale. Following this, we evaluate the paradigms of multi-agent parameter sharing, which offer immense computational benefits but strictly demand uniform state and action spaces. To bridge this gap, we detail the dimensional alignment and padding techniques developed to standardize the input constraints of deep neural networks across varying intersection topologies. Finally, we present the internal forward-pass architecture of the proposed AnyLight model and the centralized training strategy employed to optimize it efficiently.

5.1 The Challenge of Network Heterogeneity

Real world urban environments are fundamentally heterogeneous. Unlike strictly controlled artificial grid networks, a standard metropolitan traffic system is composed of a diverse array of intersections that vary drastically in their physical topology, phase configurations, and traffic demand profiles. A single continuous urban corridor may feature complex five-way arterial junctions, standard four-way signalized intersections, and simple three-way T-junctions, all coexisting and interacting within the same geographical and operational area. Furthermore, these individual intersections often differ significantly in their inherent physical capacities, such as the varying number of incoming lanes, differing spatial lengths of the road segments, and distinct speed limits.

From a machine learning perspective, this physical asymmetry presents a severe structural obstacle. Standard Deep Neural Networks (DNNs) are strictly constrained by fixed-dimensional input and output layers. If an intersection's state space is defined by its specific number of traffic movements, a three-way junction will naturally generate a much smaller localized observation vector than a five-way junction. Similarly, the action space, defined by the number of valid traffic signal phases, will vary from intersection to intersection. Consequently, training a distinct neural network for every unique intersection topology in a city leads to an explosion of trainable parameters, rendering the system computationally intractable and highly susceptible to overfitting.

The primary objective of the AnyLight architecture is to achieve true *Generalizability*. In this context, generalizability is defined as the capability of a single, unified DRL model to effectively govern any intersection within a heterogeneous network, regardless of its specific geometric layout or phase configuration. By establishing a universally compatible mathematical representation, the proposed model aims to operate seamlessly across diverse topologies without requiring distinct architectural modifications, localized hyperparameter tuning, or costly retraining phases for every unique junction.

5.2 Multi-Agent Topologies: Independent vs. Shared Policies

To deploy autonomous agents across a distributed traffic network, researchers must determine how the underlying neural networks are instantiated and trained. While there is a vast spectrum of hybrid architectures that mix localized and globalized learning mechanisms, we focus on the two primary foundational paradigms for simplicity: Independent Policies and Parameter Sharing.

Independent Policies (Decentralized Learning)
The most intuitive approach to decentralized control is to instantiate a completely independent neural network for every individual intersection in the grid. While this architecture allows each agent to perfectly tailor its behavior to the specific geometry and localized traffic demand of its assigned junction, it introduces profound theoretical and practical limitations.

First, this approach scales exceedingly poorly. In an urban network consisting of N intersections, the system must simultaneously maintain, store, and update N distinct sets of neural weights. This strict isolation leads to severe sample inefficiency, as an agent managing one intersection cannot benefit from the successful experiences or penalized mistakes of an agent operating at a structurally identical intersection elsewhere in the city.

Most critically, deploying independent learners drastically exacerbates the issue of environment non-stationarity. In a multi-agent setting where every agent is continuously updating its own unique policy in parallel, the transition dynamics of the shared environment become highly unpredictable from the perspective of any single agent. An action that yielded a high reward during early training might result in heavy congestion later, simply because neighboring agents have independently altered their routing and phase behaviors. This moving-target problem makes achieving a stable, network-wide cooperative equilibrium notoriously difficult.

Parameter Sharing (Unified Policy)
Conversely, the Parameter Sharing paradigm addresses these limitations by requiring all

N agents to utilize a single, globally shared neural network policy. In this unified setup, the identical set of neural weights is employed to process localized observations and dictate actions for every intersection across the traffic grid. This approach is fundamental to the concept of a "Universal Model," which has been shown to effectively handle city-scale coordination by abstracting control logic from specific topological constraints [30]. While this paradigm necessitates a standardized input and output structure, it is highly desirable for large-scale deployments due to three primary advantages:

1. **Sample Efficiency:**
   The shared model learns from the collective, aggregated experiences of all agents simultaneously. This parallelized data acquisition exposes the network to a vast diversity of traffic states, significantly accelerating convergence compared to independent learners.

2. **Memory Footprint:**
   By maintaining a single set of weights, the system reduces the space complexity of the policy from $\mathcal{O}(N)$ to $\mathcal{O}(1)$ relative to the number of intersections [27]. This ensures that the computational resources required for deployment remain constant, even as the network scales to thousands of junctions.

3. **Scalability and Robustness:**
   A shared policy is inherently discouraged from memorizing localized geometric quirks or specific phase timings. Instead, it is forced to learn universal principles of traffic dynamics. This abstraction guarantees that the model scales effortlessly to massive networks without parameter bloat, while maintaining extreme robustness against chaotic, highly heterogeneous intersection topologies.

Despite its overwhelming advantages, deploying a shared policy across a heterogeneous network introduces a critical technical hurdle. Standard DNNs demand strictly fixed-size input tensors. An independent policy easily bypasses this; a 3-way intersection model can be initialized to expect 6 movements, while a 4-way model expects 12. A shared policy, however, must accept observations from *both* a 3-way and a 4-way intersection using the exact same input layer.

5.3 Padding And Masking Strategy

Within the AnyLight framework, we adopt a unified state-action representation approach, wherein each intersection's localized state, phase matrix, and privileged neighbor actions

are structured consistently based on their underlying traffic movements. However, due to the profound topological diversity inherent in urban networks, the total number of valid traffic movements and available signal phases varies drastically between junctions.

To bridge this gap and enable a globally shared policy, we implement a rigorous zero-padding and masking strategy. While zero-padding has been utilized as a heuristic in models such as PressLight [26] to handle irregular (e.g., 3-leg) intersections via vector concatenation, AnyLight extends this concept by integrating it with an information retrieval framework. Unlike naive concatenation, our Cross-Attention mechanism allows the model to dynamically query relevant traffic features while systematically ignoring padded elements through boolean masks. This approach standardizes varying localized representations into a uniform dimensionality, a prerequisite for efficient batch processing, without introducing the noise typically associated with artificial data. These masks ensure that padded entries are strictly excluded from both the internal attention computations and the final policy distribution. The specific padding transformations applied to the three core informational vectors are detailed below.

Traffic State Vectors

Because the number of navigable traffic movements $|\mathcal{M}_i|$ fluctuates based on topological differences, we enforce a uniform vector length by defining a global maximum number of movements, denoted mathematically as M_{max}. This specific scalar is determined by identifying the most complex intersection within the targeted traffic network, or it can be set empirically to a sufficiently large threshold. For intersections possessing fewer movements than M_{max}, we apply zero-padding to the tail end of the observation matrix. Consequently, the original localized state vector is expanded into a universally sized matrix $\tilde{S}_i(t) \in \mathbb{R}^{M_{max} \times 8}$. Within this padded matrix, each valid movement retains its standard 8-dimensional feature vector, and the remaining surplus slots are filled entirely with zeros.

Traffic Phase Vectors

Similarly, the number of operational traffic phases $|\mathcal{P}_i|$ varies widely across the network. To standardize this action-mapping representation, we establish a global maximum number of phases, denoted as P_{max}. The original phase matrix G_i must be padded along two distinct axes to successfully align with the shared architecture. First, each individ-

ual phase state vector is zero-padded along the movement dimension to reach the global M_{max}. Subsequently, the entire aggregated set of phases is zero-padded along the phase dimension to reach P_{max}. As a result, the varying traffic phase vector is expanded into a fixed-dimensional matrix $\tilde{G}_i \in \mathbb{R}^{P_{max} \times M_{max}}$, ensuring complete structural uniformity regardless of the intersection's actual phase complexity.

Neighbor Action Vectors (Critic Privileged Information)
In our specific multi-agent training paradigm, the centralized critic requires privileged information regarding the actions taken by adjacent neighbors to accurately estimate the value function. These neighbor action vectors, representing the activation status of connecting lanes at each discrete time step, are encoded using the exact same unified movement ordering as the primary traffic state and phase vectors. Accordingly, we apply the identical zero-padding strategy along the movement dimension. This transformation expands the neighbor's localized action data into a standardized vector of dimension $\mathbb{R}^{1 \times M_{max}}$. By maintaining this strict structural consistency across all input sources, the AnyLight architecture can seamlessly aggregate, batch, and process complex topological data without encountering tensor dimensionality conflicts.

Masking
While zero-padding successfully standardizes the dimensionality of the input tensors, it inherently introduces artificial data into the network. To ensure that the agent's learning dynamics and decision-making processes are strictly confined to valid, physically realizable phases, we apply a dedicated phase mask.

This mask is integrated directly into the core computational graph of the model. First, it is applied within the internal cross-attention mechanism, guaranteeing that the agent exclusively attends to valid phase entries during the feature extraction process, while systematically ignoring the padded zeros. Second, the mask is enforced on the output probability distribution of the actor network. By masking the logits prior to the softmax activation function, we mathematically guarantee that the probabilities of padded actions are forced to zero, ensuring the agent samples its decisions exclusively from the set of feasible phases. Finally, the identical mask is applied to the output value estimates of the critic to maintain strict mathematical consistency.

To illustrate this mechanism, consider a standard three-way intersection that natively possesses exactly three valid traffic phases, which must be aligned to a globally defined maximum of $P_{max} = 4$. The corresponding phase mask for this specific intersection is defined as the binary vector $[0, 0, 0, 1]$. In this explicit representation, a 0 indicates a valid operational phase requiring no alteration, and a 1 denotes a padded, artificial position that must be strictly masked and excluded from all downstream network computations.

5.4 Proposed Neural Network Architecture

The AnyLight architecture is built upon the Multi-Agent Proximal Policy Optimization (MAPPO) [29] framework, utilizing a parameter-shared Actor-Critic paradigm [21]. Operating under the CTDE framework [31], the model processes padded inputs to seamlessly control any intersection topology, regardless of its geometric complexity. Both the Actor and Critic networks share a foundational design philosophy: transforming variable-sized, heterogeneous inputs into a standardized latent space using Recurrent Memory and Cross-Attention mechanisms. By utilizing a single backbone to extract generalized traffic representations, the architecture solves the structural limitations of decentralized algorithms, enabling a unified, highly scalable control policy across an entire city network.

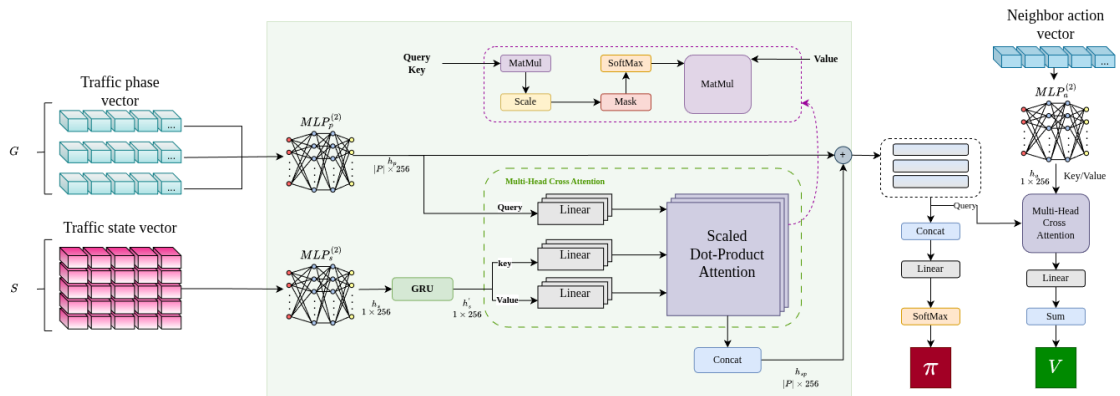


Figure 5.1: Architectural overview of the AnyLight model.

Temporal State Encoding

Because traffic is inherently dynamic, a single static snapshot of queue lengths is insuf-

ficient to determine whether congestion is building up or actively clearing. To grant the model a memory of recent traffic flow, the architecture first processes the padded traffic state vector $\tilde{S}_i(t)$.

Since the input $\tilde{S}_i(t)$ is a matrix of size $M_{max} \times 8$, a Multi-Layer Perceptron (MLP) [9] with two linear layers is applied *point-wise* to each individual movement's feature vector. This projects the 8-dimensional raw data into a higher-dimensional feature space, expressed as:

$$\mathbf{h}_s \in \mathbb{R}^{M_{max} \times d} = \text{MLP}_s^{(2)}(\tilde{S}_i(t)) \tag{5.1}$$

where $\mathbf{h}_s$ is the resulting state feature matrix and d denotes the target feature dimension. Next, this feature matrix is fed into a Gated Recurrent Unit (GRU) [7]. The GRU selectively integrates crucial historical traffic information while forgetting irrelevant data. The updated state feature matrix $\mathbf{h}'_s$ is computed recursively:

$$\mathbf{h}'_s \in \mathbb{R}^{M_{max} \times d} = \text{GRU}(\mathbf{h}_s, \mathbf{h}^{\text{GRU}}_{(s,t-1)}) \tag{5.2}$$

where $\mathbf{h}^{\text{GRU}}_{(s,t-1)}$ represents the hidden state matrix produced by the GRU at the previous decision step $t-1$. A residual connection is additionally employed here to ensure gradient stability during training.

The Cross-Attention Decoder

AnyLight resolves the challenge of mapping varying movement states to varying action phases by treating action selection as an information retrieval problem. Specifically, the padded traffic phase vector $\tilde{G}_i$ is also transformed into a high-dimensional feature matrix using an MLP:

$$\mathbf{h}_p \in \mathbb{R}^{P_{max} \times d} = \text{MLP}_p^{(2)}(\tilde{G}_i) \tag{5.3}$$

where P_{max} defines the maximum padded phase dimension.

To create phase-conditioned state features, we leverage a multi-head cross-attention mechanism [23]. Here, the available phases "query" the current traffic state to extract relevant congestion information. For each attention head h (with a total of 4 heads), the Query ma-

trix $\mathbf{Q}_h$ is derived from the phase features $\mathbf{h}_p$, while the Key $\mathbf{K}_h$ and Value $\mathbf{V}_h$ matrices are derived from the dynamically updated state features $\mathbf{h}'_s$:

$$\begin{aligned} \mathbf{Q}_h &= \mathbf{h}_p \mathbf{W}_h^Q \in \mathbb{R}^{P_{max} \times d_k} \\ \mathbf{K}_h &= \mathbf{h}'_s \mathbf{W}_h^K \in \mathbb{R}^{M_{max} \times d_k} \\ \mathbf{V}_h &= \mathbf{h}'_s \mathbf{W}_h^V \in \mathbb{R}^{M_{max} \times d_v} \end{aligned} \tag{5.4}$$

where $\mathbf{W}_h^Q$, $\mathbf{W}_h^K$, and $\mathbf{W}_h^V$ are learnable parameter matrices for head h.

The raw attention scores are computed via the scaled dot-product. Crucially, during this step, the phase mask is applied. The attention weights corresponding to padded, invalid phases are forcefully set to negative infinity ($-\infty$), mathematically guaranteeing that the Softmax function assigns them zero weight. The masked attention for each head is then calculated as:

$$\text{Att}_h(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{softmax}\left(\frac{\mathbf{Q}_h(\mathbf{K}_h)^T}{\sqrt{d_k}} + \text{Mask}\right)\mathbf{V}_h \tag{5.5}$$

Finally, the outputs from all H heads are concatenated and passed through a linear projection layer, yielding the aggregated, phase-specific feature matrix:

$$\mathbf{h}_s^p \in \mathbb{R}^{P_{max} \times d} = \text{Concat}(\text{Att}_1, \ldots, \text{Att}_H)\mathbf{W}_O \tag{5.6}$$

This matrix perfectly aligns the current traffic realities with the specific control requirements of each available intersection phase.

The Actor Network: Policy Formulation

The Actor network utilizes the aggregated feature matrix $\mathbf{h}_s^p$ to formulate the final control policy.

A final MLP policy head processes this information into raw logits for each of the P_{max} phases. Before action selection, the phase mask is applied a second time, forcefully setting the logits of invalid phases to $-\infty$. A Softmax activation is then applied, resulting in a valid probability distribution $\pi_\theta(a_t|s_t)$ where padded phases have exactly a 0% prob-

ability of being selected.

The Critic Network: Collaborative Value Estimation
While the Actor network operates strictly in a decentralized manner, the Critic network is tasked with estimating the expected long-term global return $V(s_t)$. To achieve this, it mirrors the Actor's structure but incorporates a specialized mechanism to address the inherent challenges of multi-agent coordination.

Learning efficient coordination among agents in complex, realistic traffic networks remains difficult, as agents face significant differences in their surroundings, state spaces, and action spaces. Unlike homogeneous systems, where agents operate under similar conditions and can easily establish an implicit consensus, heterogeneous environments lack this uniformity. For instance, the same phase index that signals straight traffic at one intersection might indicate a left turn at another, making it difficult for agents to align their understanding of neighboring states and actions.

To overcome this communication barrier, the AnyLight architecture leverages the CTDE paradigm. Because traffic networks are highly interconnected, the true state of one intersection depends heavily on the upstream traffic released by its neighbors. Consequently, the Critic is granted privileged access to the padded Neighbor Action Vectors introduced in Section 5.3. These neighbor actions are embedded via an MLP and fed into a secondary Cross-Attention Decoder within the Critic. By using the primary state features as the Query and the neighbor embeddings as the Key and Value, the network can dynamically translate heterogeneous neighbor actions into a standardized context, assessing how incoming traffic waves will impact local congestion. Finally, an MLP projects this enriched collaborative representation into a single scalar value estimation.

5.5 Training

The AnyLight framework utilizes a Distributed RL architecture, orchestrated via the Ray framework [17], to manage the computational complexity of training across large-scale traffic networks.

This architecture is partitioned into two primary functional components: the *Environment*

Runner and the *Learner*. Together, these components implement a robust CTDE pipeline.

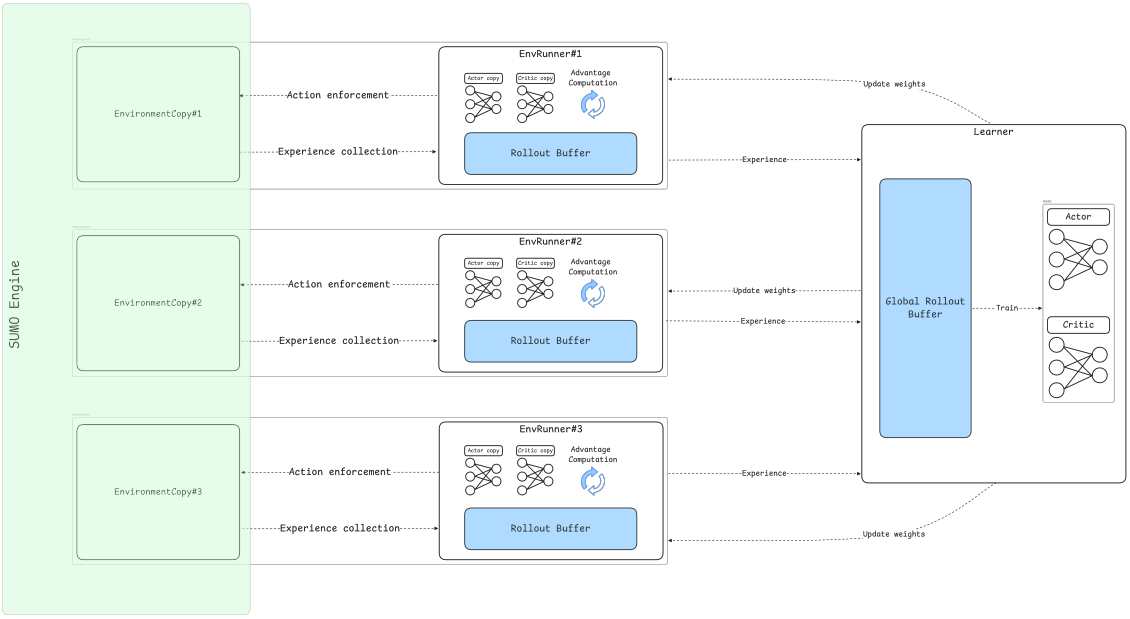


Figure 5.2: Architectural overview of the parallel experience collection pipeline. Three meta-agents manage independent simulation instances, aggregating heterogeneous experiences into a rollout buffer. The central learner retrieves these buffers, applies padding and normalization, and executes the PPO optimization cycle to update the shared model.

5.5.1 Environment Runner

The Environment Runner serves as the decentralized worker responsible for experience data acquisition. In this pipeline, the system instantiates multiple parallel simulation workers, referred to as meta-agents, each operating within its own isolated Ray process.

SUMO Interaction and Seeding

Each *EnvironmentRunner* maintains a dedicated representation of the target traffic network and manages its own independent simulation execution via the TraCI protocol [25]. Operating under a client-server architecture, each runner acts as a client that initiates a dedicated SUMO [14] subprocess (the server) on a unique communication port.

This design ensures absolute process isolation and prevents resource contention during parallel execution, as each worker operates with its own localized configuration files and traffic demand profiles.

Crucially, the architecture explicitly manages stochasticity to balance the competing requirements of robust policy training and fair model evaluation. In SUMO, randomness heavily dictates microscopic vehicle behavior by sampling driving parameters from predefined statistical distributions. For instance, a vehicle's desired speed is determined by a `speedFactor`, which by default is sampled from a normal distribution $\mathcal{N}(\mu = 1.0, \sigma = 0.1)$ relative to the road's speed limit, creating a realistic mix of cautious and aggressive drivers.

Furthermore, within the default Krauß car-following model, stochastic deceleration (often termed "dawdling", governed by the `sigma` parameter) introduces random, sub-optimal reductions in a driver's acceleration. This mathematically mimics human imperfection and inattention, naturally causing the spontaneous speed oscillations and stop-and-go waves seen in real-world traffic jams. Consequently, under different random states, an individual vehicle might systematically drive 10% slower than the limit or hesitate inconsistently before an intersection, directly altering the chaotic buildup of localized congestion.

To exploit these behavioral dynamics, the AnyLight framework employs a dual-seeding strategy. During the training phase, each simulation worker is initialized using the `--random` flag, which seeds the simulation using the system clock. This forces every worker to experience highly unpredictable, non-deterministic traffic conditions and driving behaviors. Exposing the global policy to this continuous stream of diverse and stochastic environments prevents the agents from overfitting to specific, deterministic traffic patterns, thereby significantly enhancing the model's robustness to real-world deployment conditions. Conversely, during the evaluation phase, the architecture strictly employs explicitly fixed integer seeds via the `--seed <INT>` parameter. This ensures that the generated microscopic behaviors, such as the exact moments of vehicle hesitation or speed reduction, are perfectly reproducible across all baseline models, guaranteeing a rigorous and fair benchmark comparison.

The interaction between the *EnvironmentRunner* and SUMO occurs in "green time" blocks.

Rather than making decisions at every sub-second simulation step, the agent selects a phase configuration that remains active for a fixed duration (e.g., 15 seconds). During this window, the environment executes multiple underlying SUMO simulation steps via the TraCI interface, ensuring that the RL agent operates at a temporal resolution suitable for high-level traffic management. A more detailed formalization of the discrete action space and the transition mechanics between signal phases is provided in Section 4.4.

Local Inference and Advantage Computation
Within its isolated thread, the *EnvironmentRunner* maintains a local copy of the current policy to perform inference. During the interaction phase, it executes the Actor network in a gradient-free mode to sample actions and interact with its assigned traffic grid. It stores the resulting trajectories, comprising observations, actions, rewards, log-probabilities, and valid phase masks, in a local rollout buffer.

Once a full episode is completed, the *EnvironmentRunner* performs local preprocessing to calculate the Generalized Advantage Estimation (GAE) [19]. As introduced in Chapter 3, the Critic network evaluates the given state to provide a baseline expectation of future returns. GAE blends one-step TD errors with multi-step returns to compute the advantage function, dictating how much better or worse an action was compared to the Critic's prediction. By offloading these computationally intensive advantage calculations to the distributed workers, the architecture prevents the central learner from becoming a processing bottleneck. Finally, at the start of each new training job, the *EnvironmentRunner* pulls the latest model weights from the shared repository, ensuring its local policy remains synchronized with the global brain.

5.5.2 Learner

The Learner acts as the central orchestrator of the AnyLight architecture, responsible for global model management and the execution of the optimization cycle.

Batch Aggregation and Normalization
At the conclusion of a training job, the learner retrieves the processed experience buffers from all active environment runners using a remote retrieval signal. As detailed in Section 5.3, the structural challenges posed by network heterogeneity are addressed via a rigorous

padding strategy. All localized experiences are standardized into fixed-dimensional tensors, which allows these heterogeneous trajectories to be seamlessly concatenated into a massive, globally shared training batch within the neural network's computational graph.

Before optimization begins, the learner applies batch-wide advantage normalization to the aggregated GAE estimates. Mathematically, the raw advantages A_i are transformed to have a zero mean and unit variance according to:

$$\hat{A}_i = \frac{A_i - \mu_A}{\sigma_A + 10^{-5}} \tag{5.7}$$

where μ_A and σ_A represent the mean and standard deviation of the advantages within the current training batch, respectively. This normalization serves three primary architectural purposes:

1. **Scale Invariance:**
   In microscopic traffic simulations, reward magnitudes, derived from negative queue lengths, fluctuate dramatically between peak-hour congestion and light midnight traffic. Without normalization, the massive gradients produced during high-congestion episodes would disproportionately dominate the optimization process, potentially destabilizing or collapsing the policy. Standardization ensures that the learning dynamics remain invariant to the absolute magnitude of the environment's rewards.

2. **Relative Action Evaluation:**
   PPO updates the policy based on whether an action performed better or worse than the Critic's baseline. By centering the batch mean at exactly zero, normalization mathematically guarantees a balanced learning signal. Approximately half of the actions in any given batch are assigned a positive advantage, encouraging the Actor to increase their probability, while the others are discouraged. This ensures a consistent, zero-centered pressure on the policy regardless of the overall traffic severity.

3. **Optimizer Stability:**
   The Adam optimizer [11] and the PPO [20] clipping mechanism are highly sensitive to the scale of the advantage signal. Standardizing the variance to unity ensures that the chosen learning rates remain effective throughout all stages of training, preventing numerical instabilities such as exploding gradients while ensuring the policy updates remain within the stable trust region.

In the highly stochastic environment of microscopic traffic simulation, where vehicle arrivals fluctuate and neighboring intersection actions cause complex cascading effects, relying solely on raw cumulative returns would cause training gradients to fluctuate wildly. GAE, augmented by this normalization strategy, is critical in this specific context because it significantly reduces the variance of the learning signal. It provides a clean, stable metric for the Actor to discern which actions genuinely cleared congestion versus which were merely lucky statistical anomalies.

The PPO Optimization Cycle

With the standardized batch prepared, the central learner executes the core optimization step. The training loop performs multiple epochs of gradient descent over the collected data, updating both networks simultaneously. The Actor network is updated using the clipped PPO objective. In a chaotic MARL setting, overly large policy updates can easily collapse the learning process, thus the clipping mechanism guarantees stable, monotonic, and incremental improvements to the traffic control strategy.

Simultaneously, the Critic network is updated by minimizing the Mean Squared Error (MSE) between its predicted value estimations and the actual observed returns from the rollout trajectory.

The most powerful architectural consequence of this pipeline is the realization of the parameter-shared "Global Brain," which serves as the implementation of the CTDE paradigm. Rather than maintaining N separate neural networks for N intersections, a single centralized Actor-Critic architecture governs the entire city. Consequently, a single gradient descent step updates the network weights using the collective, aggregated experience of every agent in the network. This dramatically accelerates convergence, as the model generalizes traffic management strategies from a vast and diverse dataset of junction topologies simultaneously.

Weight Synchronization and Monitoring

Following the optimization phase, the learner updates the master copy of the model and persists the weights to a shared repository. Throughout this cycle, the learner logs critical performance metrics, including cumulative rewards, policy entropy, and loss magnitudes,

to TensorBoard, providing real-time visibility into the convergence and health of the multi-agent system.

Training Configuration and Hyperparameters

To ensure that this complex MARL pipeline converges reliably and results remain reproducible, precise hyperparameter tuning is enforced. The full configuration, encompassing both the optimization settings and the internal architectural dimensions, is summarized in Table 5.1. The network relies on the Adam optimizer for efficient gradient descent, utilizing distinct learning rates to balance the network dynamics. The Actor updates carefully with a learning rate of 10^{-4}, while the Critic adapts slightly faster with a learning rate of 2×10^{-4} to ensure the baseline evaluation remains highly accurate.

Stability is further maintained via the PPO clipping parameter $\epsilon = 0.2$, which tightly bounds the policy updates. The temporal foresight of the agents is governed by a discount factor of $\gamma = 0.95$, ensuring they prioritize long-term traffic flow rather than myopic, immediate queue clearing. The GAE parameter is set to $\lambda = 0.98$ to optimally balance the bias-variance trade-off for the temporal dependencies of traffic signals. Finally, to prevent the policy from collapsing into a sub-optimal deterministic state early in training, an entropy coefficient of 2×10^{-3} is added to the Actor's loss. This exploration bonus actively encourages the network to maintain a degree of randomness, ensuring it continues to discover novel and potentially more efficient phase combinations throughout its training lifecycle.

Optimization Settings		Model Architecture	
Parameter	**Value**	**Parameter**	**Value**
Optimizer	Adam	Hidden Dim.	256
Actor Learning Rate (LR)	10^{-4}	Attention Heads	4
Critic LR	2×10^{-4}	Phase Layers	[128, 256]
PPO Clip (ϵ)	0.2	Policy/Value Layers	[512]
Discount (γ)	0.95	GRU Units	256
GAE (λ)	0.98		
Value Function (VF) Coeff.	0.5		
Entropy Coeff.	2×10^{-3}		

Table 5.1: Full training configuration and architectural hyperparameters for the AnyLight model.

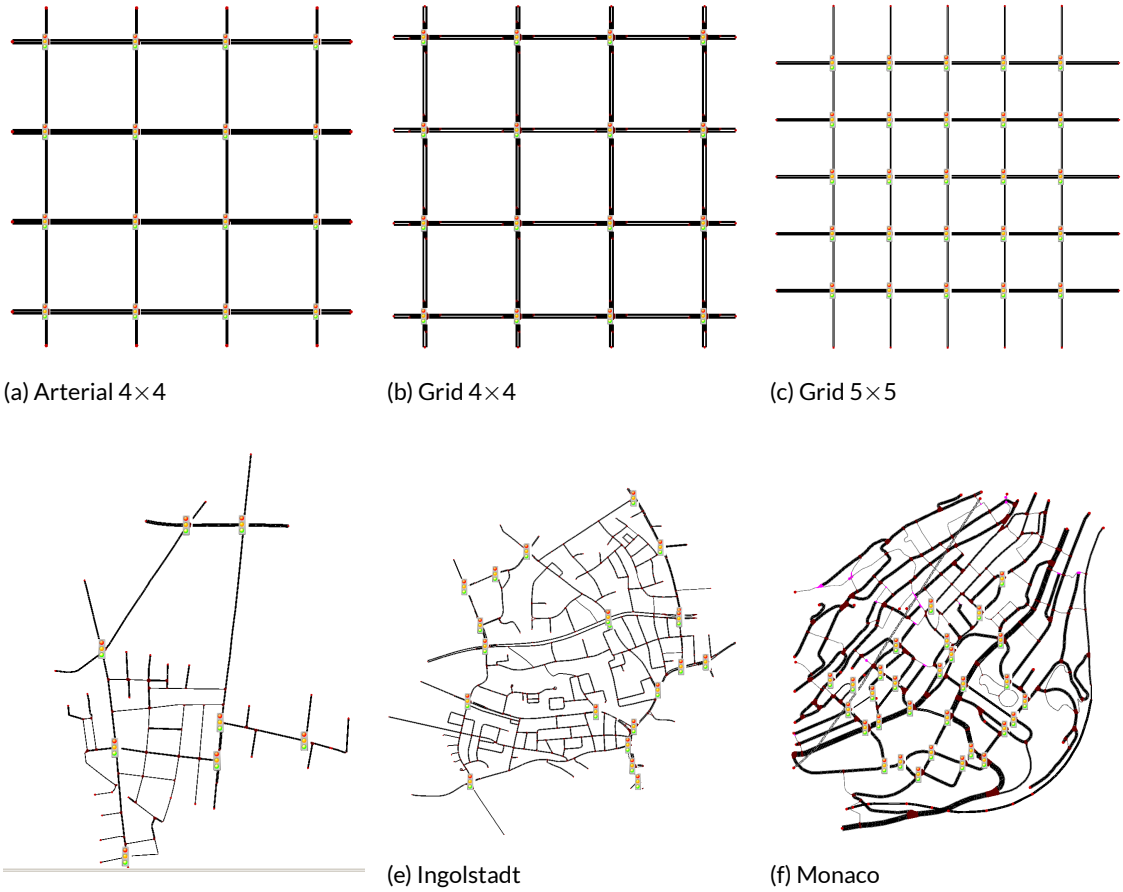
6

Experimental Framework and Simulation Scenarios

This chapter outlines the experimental methodology and the simulation scenarios designed to rigorously evaluate the performance and generalizability of the AnyLight architecture. Specifically, it details the simulation environment, the generation of dynamic traffic demands, the specific synthetic and real-world network topologies used for testing, the baseline models selected for comparison, and the core evaluation metrics.

6.1 Traffic Datasets

To evaluate the efficacy of the proposed method, we conduct comparative experiments against a range of baseline models across six distinct traffic datasets. All experiments are executed within the open-source microscopic traffic simulator SUMO [14]. The datasets utilized in these experiments include *Arterial 4×4*, *Grid 4×4*, *Cologne*, and *Ingolstadt*, which are adapted from the RESCO benchmark [2], as well as *Grid 5×5* and *Monaco*, obtained from the MA2C benchmark [6].



(a) Arterial 4×4 (b) Grid 4×4 (c) Grid 5×5

(d) Cologne (e) Ingolstadt (f) Monaco

Figure 6.1: Topological structures of the six traffic networks under evaluation. The top row displays the synthetic homogeneous topologies, and the bottom row illustrates the real-world heterogeneous topologies.

Each dataset is fundamentally composed of two elements: a physical traffic network and its corresponding traffic flows. The traffic network defines the structural and spatial organization of the environment, detailing intersection layouts, road segment configurations, and specific lane connectivity (as shown in Figure 6.1). Notably, the *Arterial 4×4*, *Grid 4×4*, and *Grid 5×5* networks are synthetically generated to be strictly homogeneous, meaning all intersections share identical topologies and phase settings. Conversely, the remaining datasets, namely *Cologne*, *Ingolstadt*, and *Monaco*, represent heterogeneous real-world networks.

The traffic flows dictate the movement patterns of the vehicles. These are characterized

by Origin-Destination (OD) pairs, flow rates, and specific vehicle generation times. To ensure fair comparisons, the OD pairs and flow generation settings remain identical to those established in their original benchmark works. All traffic flows are strictly time-varying. They are either synthetically designed to simulate specific temporal changes or calibrated directly from real-world data, ensuring they accurately reflect realistic and dynamic traffic conditions.

Table 6.1 provides detailed structural and temporal information for each of the six traffic datasets utilized in this evaluation. We exclusively provide an intersection-specific structural investigation for the heterogeneous networks, as their complex real-world topologies present the only operational dynamics worthy of such granular analysis. Consequently, detailed structural indicators for these specific networks, including the number of incoming and outgoing lanes, valid traffic movements, and signal phases per intersection, are provided in Tables 6.2, 6.3, and 6.4.

Dataset	**Network Structure**				**Volume (veh)**	**Arrival Rate (veh/min)**			
	#total-int	#2-arm	#3-arm	#4-arm		Mean	Std	Max	Min
Grid 4×4	16	0	0	16	1473	24.97	13.89	72.00	6.00
Arterial 4×4	16	0	0	16	2484	49.68	23.26	88.00	10.00
Grid 5×5	25	0	0	25	6219	121.57	106.69	752.00	0.00
Cologne Region	8	1	3	4	2046	4.26	12.13	77.00	0.00
Ingolstadt Region	21	0	17	4	4281	4.19	17.03	112.00	0.00
Monaco	30	7	9	14	2464	42.56	24.04	86.67	0.00

Table 6.1: Overview of the traffic datasets, detailing network structure, volume, and vehicle arrival rates.

Intersection	Incoming Lanes	Outgoing Lanes	Movements	Phases
Int 1	6	6	18	4
Int 2	4	4	16	2
Int 3	3	3	9	3
Int 4	6	6	18	4
Int 5	4	3	9	3
Int 6	2	4	8	2
Int 7	4	3	9	3
Int 8	4	4	16	4

Table 6.2: Intersection-specific structural details for the Cologne network.

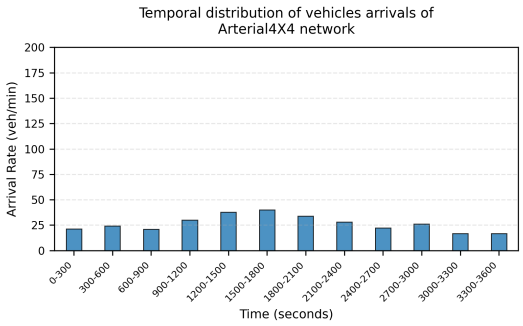
Intersection	Incoming	Outgoing	Movements	Phases	Intersection	Incoming	Outgoing	Movements	Phases
Int 1	7	4	7	3	Int 12	7	5	8	3
Int 2	6	3	6	3	Int 13	6	5	8	3
Int 3	4	3	6	3	Int 14	14	7	15	4
Int 4	6	5	10	3	Int 15	12	8	12	4
Int 5	8	4	12	4	Int 16	9	7	12	3
Int 6	8	7	14	3	Int 17	7	6	8	3
Int 7	8	7	15	3	Int 18	5	3	6	3
Int 8	7	5	9	2	Int 19	10	6	14	3
Int 9	7	6	11	3	Int 20	8	5	9	3
Int 10	8	4	12	4	Int 21	5	5	8	3
Int 11	6	4	12	3					

Table 6.3: Intersection-specific structural details for the Ingolstadt network.

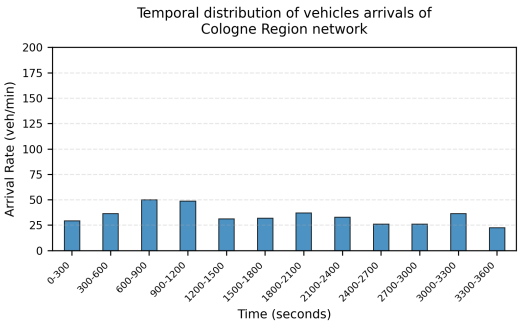
Intersection	Incoming	Outgoing	Movements	Phases	Intersection	Incoming	Outgoing	Movements	Phases
Int 1	6	6	14	6	Int 16	4	4	12	4
Int 2	4	4	12	4	Int 17	3	3	6	2
Int 3	3	3	6	2	Int 18	3	4	9	3
Int 4	2	2	3	2	Int 19	6	6	14	6
Int 5	3	3	6	2	Int 20	3	3	6	3
Int 6	4	4	12	4	Int 21	2	2	3	2
Int 7	2	3	5	2	Int 22	4	4	12	4
Int 8	4	4	12	4	Int 23	4	4	12	4
Int 9	3	3	6	2	Int 24	7	5	11	4
Int 10	6	8	18	5	Int 25	4	4	12	4
Int 11	5	4	6	2	Int 26	6	6	12	4
Int 12	2	1	2	2	Int 27	5	5	8	6
Int 13	4	4	12	4	Int 28	6	6	14	6
Int 14	3	3	6	2	Int 29	2	2	3	3
Int 15	3	3	4	2	Int 30	3	4	6	3

Table 6.4: Intersection-specific structural details for the Monaco network.

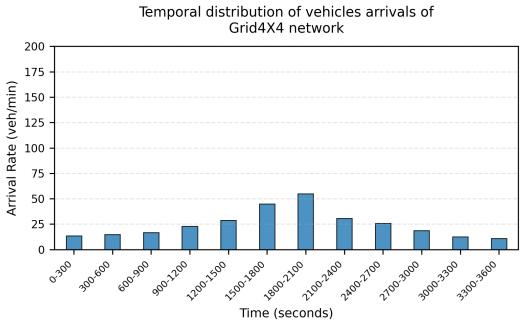
In addition, because each dataset adopts time-varying traffic flows to represent realistic dynamic conditions, we further analyze and visualize the vehicle arrival rates (measured in vehicles per minute) across the duration of the simulation. The results, as illustrated in Figure 6.2, demonstrate distinct temporal variation patterns across the different datasets. This variance highlights the extreme diversity and representativeness of the experimental traffic scenarios, ensuring the proposed architecture is evaluated under a robust spectrum of traffic demands.



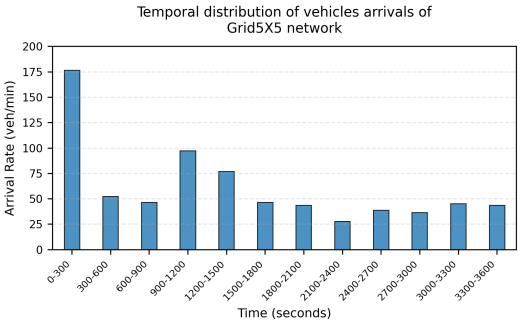
(a) Arterial 4×4



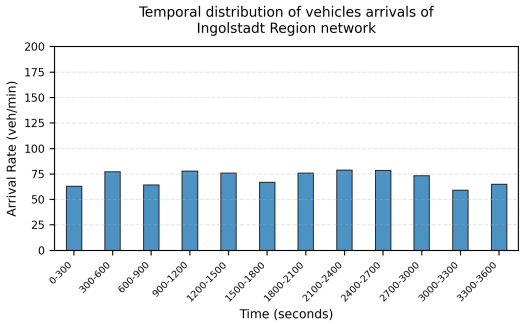
(b) Cologne



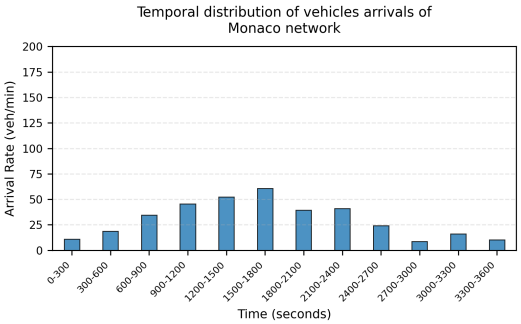
(c) Grid 4×4



(d) Grid 5×5



(e) Ingolstadt



(f) Monaco

Figure 6.2: Vehicle arrival rate distributions (vehicles per minute) over the simulation duration across the different traffic datasets.

6.2 Baselines and Comparison Models

To rigorously evaluate the performance of the AnyLight architecture, we compare it against a diverse set of baseline models. These models span from foundational, heuristic-based algorithms that represent standard legacy infrastructure, to state-of-the-art Deep Reinforcement Learning methodologies.

6.2.1 Traditional Traffic Signal Control Methods

The traditional traffic signal control baselines are rule-based systems that govern intersection dynamics without the use of learned neural representations. We evaluate three distinct classical strategies:

Fixed-Time Control (FIXED)
The Fixed-Time policy serves as the most fundamental baseline, representing legacy traffic infrastructure. It operates entirely statically and does not react to real-time traffic flow, congestion fluctuations, or vehicle queues. Its core logic relies on a cyclic, round-robin rotation through the intersection's predefined valid traffic signal phases. At each decision interval, the controller inspects the currently active phase index, increments it by 1, and applies a modulo operation based on the total number of valid phases. This ensures that upon completing the sequence, the controller loops back to the initial phase, providing a steady, strictly predictable rhythm of phase changes regardless of the underlying demand.

Greedy Control (GREEDY)
In contrast to the static nature of the Fixed-Time policy, the Greedy approach is a highly reactive, sensor-driven heuristic designed to maximize the immediate discharge of waiting traffic. Its primary target metric is the volume of halted vehicles, defined as vehicles whose current speed falls below a critical threshold, indicating they are actively queued. At every decision step, the controller iterates through all valid candidate phases. For each phase, it identifies the distinct traffic movements that would be granted a green light and aggregates the total number of halted vehicles currently waiting across those specific incoming lanes. The controller ultimately selects the phase that yields the highest cumulative halted vehicle count, aggressively prioritizing the immediate clearance of the largest queues. While practically deployed systems rely on localized lane-area loop detectors for

this information, our baseline evaluation queries the global simulation engine directly to obtain the exact halted vehicle count across the entirety of each incoming lane, representing an idealized, noise-free Greedy execution.

MaxPressure Control (MAXPRESSURE)

The MaxPressure policy [22] is a prominent traffic control algorithm designed to balance traffic loads across an entire network rather than myopically clearing localized incoming queues. It operates by measuring the dynamic imbalance between the incoming (upstream) demand and the outgoing (downstream) capacity. For any given traffic movement, the localized pressure is mathematically formalized as the difference between the incoming queue size and the outgoing queue size:

$$\text{Movement Pressure} = \text{Incoming Queue} - \text{Outgoing Queue} \tag{6.1}$$

During the phase selection process, the controller iterates through all candidate phases and calculates the pressure for every movement that would receive a green light. A high positive pressure indicates a heavily congested incoming lane coupled with an outgoing lane that possesses sufficient free capacity. Conversely, a negative or zero pressure indicates that the outgoing lane is already congested; releasing more vehicles into it would merely exacerbate the gridlock and block the intersection. By summing the pressures of all active movements for a given phase, the controller selects the configuration that maximizes this cumulative pressure difference.

In our simulation evaluations, to mirror the idealized sensor conditions of the Greedy baseline, the incoming queue size is quantified directly via the incoming lane's halted vehicle count, while the outgoing queue size is measured via the total vehicle count present on the outgoing lane. Ultimately, unlike the Greedy policy, which risks blindly feeding vehicles into already congested downstream roads, the MaxPressure policy naturally throttles green lights directed toward gridlocked areas. This mechanism induces a self-balancing network effect, implicitly coordinating traffic flow across adjacent intersections without requiring direct agent-to-agent communication.

6.2.2 Reinforcement Learning Baselines

The reinforcement learning baselines evaluate the performance of deeply parameterized, learning-based approaches. These baselines serve to benchmark the improvements achieved through multi-agent cooperation and shared representation learning. We evaluate the following methodologies:

Independent Deep Q-Network (IDQN)
The Independent Deep Q-Network (IDQN) policy [16] is a foundational value-based multi-agent baseline. In this decentralized formulation, each intersection is controlled by an isolated agent that estimates the state-action value function (Q-values) for all signal phases. Due to the absence of parameter sharing and explicit communication, IDQN operates in a highly non-stationary environment, frequently converging to sub-optimal Nash equilibria that fail to resolve network-wide congestion.

Independent Proximal Policy Optimization (IPPO)
The Independent Proximal Policy Optimization (IPPO) policy deploys the PPO algorithm [20] in a strictly decentralized configuration. By directly optimizing a parameterized policy and employing a baseline value network to compute advantage estimates, IPPO enforces a strict trust region during gradient updates. This clipping mechanism prevents catastrophic policy degradation, yielding superior stability compared to IDQN, though it remains fundamentally limited by its inability to proactively coordinate with adjacent nodes.

Independent Advantage Actor-Critic (IA2C)
The Independent Advantage Actor-Critic (IA2C) policy [6] is a decentralized baseline where each intersection independently optimizes localized actor and critic networks. Utilizing advantage estimates substantially reduces policy gradient variance, stabilizing the learning trajectory on isolated or strictly homogeneous grids. However, its independent architecture precludes spatial coordination, limiting its scalability and efficiency when deployed across highly heterogeneous urban topologies.

Multi-Agent Advantage Actor-Critic (MA2C)
The Multi-Agent Advantage Actor-Critic (MA2C) policy [6] extends IA2C by explicitly

integrating spatial coordination to mitigate environmental non-stationarity. MA2C augments the localized state and reward spaces of each agent with aggregated observations from immediate topological neighbors, enabling cooperative phase adjustments. Despite its strong performance, MA2C relies on independent neural weights and topology-specific feature engineering, severely restricting its scalability and applicability to highly heterogeneous road structures.

MPLight
The MPLight algorithm [5] is a decentralized deep reinforcement learning approach designed for large-scale traffic signal control. Serving as an extension of the Phase Competition (FRAP) method, it adopts network pressure as both the primary state representation and the optimization reward for a Deep Q-Network (DQN). By explicitly minimizing this pressure and enforcing strict parameter-sharing across all agents, MPLight achieves scalable, implicit coordination across neighboring intersections without complex communication overhead.

6.3 Training and Evaluation Protocol

To guarantee that the reported results are reproducible and directly comparable, this section formalizes the procedures governing model training, checkpoint selection, transfer evaluation, and the design of the ablation studies.

6.3.1 Training Protocol

A separate AnyLight instance is trained for each traffic scenario, with a single parameter-shared policy governing every intersection within the assigned network, as described in Chapter 5. Each training episode simulates one hour of traffic, corresponding to 3600 simulation seconds. The signal timing follows the conventions established by the original benchmarks: a 15-second green phase duration and a 5-second yellow clearance interval are applied to all networks, with the exception of Monaco, which adopts a 10-second green and 3-second yellow configuration in accordance with the MA2C benchmark [6]. The real-world scenarios additionally reproduce the time windows from which their traffic demands were calibrated. The Cologne and Ingolstadt simulations begin at 7:00 and 16:00 respectively, corresponding to the morning and evening peak hours captured by the

original datasets.

A further simulation setting inherited from the original benchmarks is the teleporting behavior of SUMO, which removes vehicles that remain stationary beyond a configurable time threshold and reinserts them further along their route. Because teleporting artificially dissolves severe gridlock, it directly affects every congestion-related metric, and its configuration must therefore match the original benchmark of each dataset to preserve comparability. Following the RESCO settings, teleporting is fully disabled for the *Grid 4×4*, *Arterial 4×4*, *Cologne*, and *Ingolstadt* scenarios, while thresholds of 600 and 300 seconds are applied to *Grid 5×5* and *Monaco*, in accordance with the GESA and MA2C configurations respectively.

Training proceeds for a maximum of 3000 episodes per scenario. Six parallel meta-agents collect complete episodes simultaneously, each within its own isolated simulation instance, as detailed in Chapter 5. At the end of each collection cycle, the central learner performs six optimization epochs over the aggregated experience batch, applying gradient norm clipping with a threshold of 10 to prevent destabilizing updates. To avoid unnecessary computation once the policy has converged, an early stopping criterion monitors the mean episode reward smoothed over a 50-episode window: if no improvement is observed for 1000 consecutive episodes, training is terminated. The checkpoint achieving the highest smoothed reward is retained as the final model for all subsequent evaluations.

6.3.2 Ablation Study Design

To isolate the contribution of the individual design choices, two ablated variants of the full AnyLight model are trained and evaluated under the identical protocol described above:

- **Without Neighbor Information:**
  The privileged neighbor action vectors are removed from the centralized critic, which must estimate the value function from local observations alone. This variant quantifies the benefit of the collaborative value estimation mechanism introduced in Chapter 5.

- **Without Collaborative Reward:**
  The reward function is restricted to the incoming queues only, mirroring the formulation commonly adopted in the literature. This variant validates the spatially coupled reward design motivated in Chapter 4.

The ablation studies are exclusively performed on the Arterial 4×4 grid network, as its homogeneous structure provides a controlled environment to clearly observe the impact of the architectural changes.

6.4 Evaluation Metrics

To ensure a robust and fair comparison across all selected baseline models, we execute the evaluation phase over 10 independent episodes. Each episode is initialized with a unique random seed, and we ensure rigorous consistency by utilizing identical seeds across all evaluated models for corresponding episodes. Specifically, the i-th evaluation episode is initialized with seed $1000+100\cdot i$, yielding the fixed seed set $\{1000, 1100, \ldots, 1900\}$. We adopt the testing settings, evaluation metrics, and computation methodologies established in the MA2C framework [6]. Specifically, we assess the performance of the traffic signal control algorithms using the following primary metrics:

- **Average Queue Length (veh):**
  This metric quantifies the spatial congestion of the network. It is calculated as the total number of stopped vehicles across the entire simulation network at each time step, averaged over the duration of the episode.
- **Average Vehicle Speed (m/s):**
  This metric reflects the overall fluidity of the traffic flow. It is defined as the mean speed of all active vehicles present in the network at a given time step, averaged across the episode.
- **Average Intersection Delay (s):**
  This metric measures the localized waiting time experienced by vehicles at intersections. It tracks the continuous time a vehicle spends halted in a queue. Crucially, this waiting time accumulator resets to zero the moment the vehicle begins moving again, thereby specifically isolating the delay caused by traffic signals rather than general travel time.
- **Trip Completion Rate (veh/s):**
  This metric indicates the throughput capacity of the network. It is calculated as the average number of vehicles that successfully reach their final destinations per simulation step.

- **Average Trip Delay (s):**
  This metric evaluates the temporal inefficiency of the network from the perspective of the completed journeys. It measures the total delay accumulated by vehicles that successfully reach their destinations, calculated as the difference between their actual travel time and their theoretical free-flow travel time.
- **Average Trip Time (s):**
  This metric represents the total temporal cost of traversing the network. It is defined as the total time taken by vehicles to successfully complete their trips, from generation at their origin to arrival at their destination.

These metrics collectively provide a comprehensive view of the network's performance, capturing both the localized efficiency at individual intersections and the global throughput of the entire traffic system. All metrics are reported as the mean and standard deviation computed across the 10 evaluation episodes.

6.5 Hardware and Computational Setup

To provide context for the computational efficiency and feasibility of the training pipeline, we detail the hardware and software configuration used for all experiments. The reinforcement learning training loop, which involves simultaneous microscopic traffic simulation and policy optimization, was executed on a single-node workstation. This machine was powered by an Intel Core i7-7700K CPU, providing the multi-threading capabilities required to manage concurrent simulation processes. This was paired with an NVIDIA GeForce RTX 2060 GPU equipped with 6 GB of dedicated GDDR6 VRAM, utilized to accelerate deep neural network training (backpropagation) and batch inference. Additionally, the system was equipped with 16 GB of DDR4 system RAM, which was carefully managed to host the workspace and active rollout buffers.

The underlying software stack was built primarily in Python, serving as the foundational programming language for the execution of the training scripts and data processing. Within this stack, PyTorch was heavily utilized to construct, train, and run inference for the actor and critic neural network models. To orchestrate the parallel execution of the traffic environments, we employed Ray (specifically Ray Core). To overcome the strictly single-threaded nature of the traffic simulator, Ray seamlessly manages a pool of decentralized

rollout workers, with each worker running an independent simulation instance. Given the specific hardware constraints of the workstation, the pipeline was configured to run six parallel workers, maximizing CPU core utilization while keeping the system memory footprint safely within the 16 GB limit. Finally, the microscopic traffic environments themselves were simulated using SUMO (Simulation of Urban Mobility) [14], controlled dynamically via the TraCI (Traffic Control Interface) API to extract real-time traffic observations (e.g., induction loop detector counts, queue states) and execute signal control decisions.

By decoupling simulation rollouts across parallel CPU workers and offloading policy optimization to the GPU, the training pipeline achieves high sample efficiency, demonstrating that complex multi-agent traffic control models can be trained within reasonable timeframes even on standard commodity hardware.

7

Results and Comparative Evaluation

This chapter presents a comprehensive analysis of the experimental results obtained by evaluating the AnyLight architecture across the simulation scenarios detailed in the previous chapter. The primary objective is to rigorously assess the performance, scalability, and adaptability of the proposed model in comparison to established traffic signal control baselines. First, the training dynamics and convergence properties of the algorithms are examined. Subsequently, the performance of the model is systematically evaluated on both structurally uniform synthetic grids and highly complex, heterogeneous real-world road networks. Finally, a series of ablation studies is conducted to isolate and validate the individual contributions of the core architectural components, concluding with a synthesized discussion of the overarching findings and their implications for large-scale traffic management.

7.1 Training Dynamics and Convergence

Before comparing final performance, we examine how the AnyLight policy evolves during training on each scenario. The analysis focuses on the mean episode reward, smoothed over a 50-episode window as described in Chapter 6, which directly reflects the network-wide queue reduction objective. Two aspects are of particular interest: the speed of con-

vergence, measured as the number of episodes required to approach the final performance plateau, and the stability of the learning process, assessed through the variance of the reward signal and the policy entropy logged throughout training. Alongside these primary indicators, we also report the value-function loss, which reflects how well the centralized critic tracks the true return, and the action-change frequency, which quantifies how often the shared policy switches the active phase at each decision point.

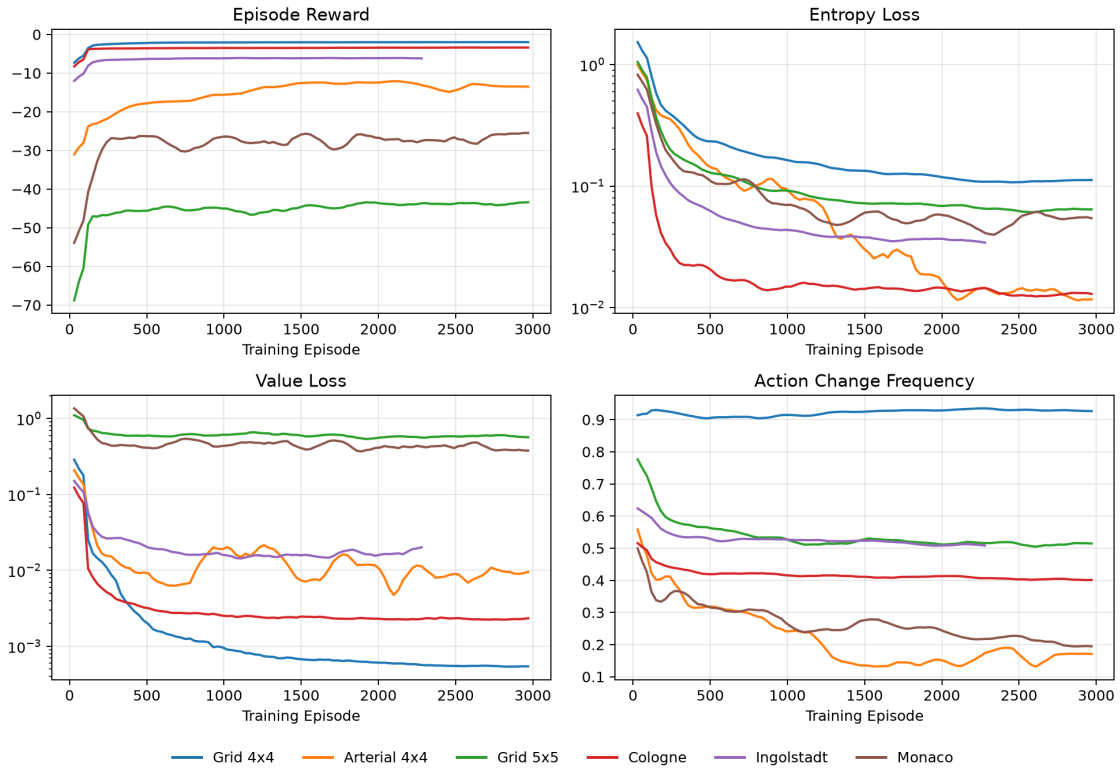


Figure 7.1: Training curves for the AnyLight policy on the six benchmark networks. Each panel overlays the per-network learning signal: smoothed mean episode reward (upper-left), policy entropy in logarithmic scale (upper-right), critic value loss in logarithmic scale (lower-left), and action-change frequency at the agent level (lower-right). Curves are smoothed with a 5-point rolling mean for readability.

Figure 7.1 shows that the reward curves rise rapidly during the first few hundred episodes on every scenario and then enter a long, slowly improving plateau, confirming that the parameter-shared MAPPO update is stable across both homogeneous and heterogeneous topologies. The homogeneous Grid 4×4 and the heterogeneous Cologne network are the easiest scenarios, converging to within a few percent of their final reward by roughly

episode 500. Arterial 4×4 and Monaco require substantially more interaction before plateauing, reflecting the additional difficulty introduced by directional demand imbalance and by Monaco's mixture of two-armed and six-phase junctions, respectively. Grid 5×5 is the hardest scenario in absolute terms: the reward improves monotonically but the policy is still climbing at the 3000-episode cap, consistent with the higher congestion level discussed in Chapter 6. Ingolstadt is the only scenario in which the early-stopping criterion triggers, terminating training around episode 2280 once the smoothed reward stops improving.

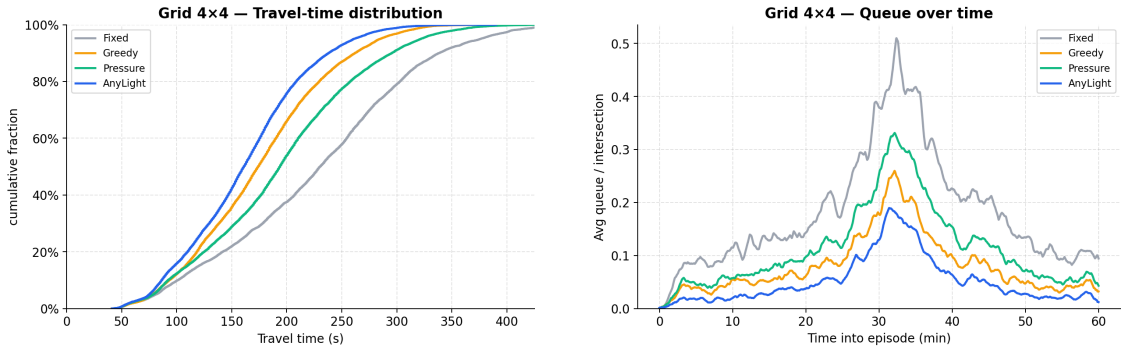
The entropy curves drop by roughly two orders of magnitude on every network, indicating a clean transition from broad exploration to a near-deterministic policy without entropy collapse. The value-loss curves mirror this behavior, falling by one to three orders of magnitude as the critic learns to predict the return accurately; the residual value loss is largest on Grid 5×5 and Monaco, consistent with the higher reward variance imposed by their longer queues and more complex demand patterns. Finally, the action-change frequency reveals a behavioral specialization that is not visible from the reward alone: on the small Grid 4×4 the policy maintains a high switching rate of around 0.9, exploiting the regular structure to cycle frequently between phases, whereas on Arterial 4×4 and Monaco the switching rate drops sharply from above 0.6 to below 0.2, showing that the policy learns to hold longer green windows on the dominant directions. The remaining networks settle at intermediate values, suggesting that the shared backbone is able to specialize the temporal cadence of its decisions to each scenario despite the unified parameterization.

7.2 Performance on Homogeneous Networks

We first evaluate AnyLight on the three synthetic homogeneous topologies: Grid 4×4, Arterial 4×4, and Grid 5×5. Because every intersection in these networks shares an identical layout and phase structure, this setting isolates the model's capacity for multi-agent coordination from the additional challenge of topological generalization. Tables 7.1, 7.2, and 7.3 report the six evaluation metrics defined in Chapter 6 for all models, computed over the shared fixed-seed episodes.

Grid 4×4 Network						
Model	**Queue (veh) ↓**	**Speed (m/s) ↑**	**Int. Delay (s) ↓**	**Compl. Rate (veh/s) ↑**	**Trip Delay (s) ↓**	**Trip Time (s) ↓**
Fixed-Time	0.18 ± 0.00	6.72 ± 0.02	84.50 ± 0.42	0.3994 ± 0.0004	115.08 ± 0.47	226.63 ± 0.50
Greedy	0.08 ± 0.00	8.73 ± 0.03	37.12 ± 0.55	0.4026 ± 0.0006	64.34 ± 0.67	175.85 ± 0.74
MaxPressure	0.11 ± 0.00	7.93 ± 0.03	52.47 ± 0.78	0.4018 ± 0.0005	82.96 ± 0.86	194.52 ± 0.99
IDQN	2.43 ± 0.03	7.02 ± 0.03	73.91 ± 1.02	**0.4092 ± 0.0000**	109.53 ± 1.11	220.69 ± 1.08
IPPO	3.12 ± 0.50	6.11 ± 0.05	102.83 ± 21.38	**0.4092 ± 0.0000**	134.31 ± 20.56	242.87 ± 18.64
MPLight	0.77 ± 0.02	8.74 ± 0.02	27.88 ± 0.50	**0.4092 ± 0.0000**	**45.78 ± 0.66**	162.62 ± 0.94
AnyLight	**0.05 ± 0.00**	**9.60 ± 0.03**	**22.90 ± 0.36**	0.4040 ± 0.0004	49.76 ± 0.45	**161.30 ± 0.63**

Table 7.1: Performance comparison on the Grid 4×4 network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better.



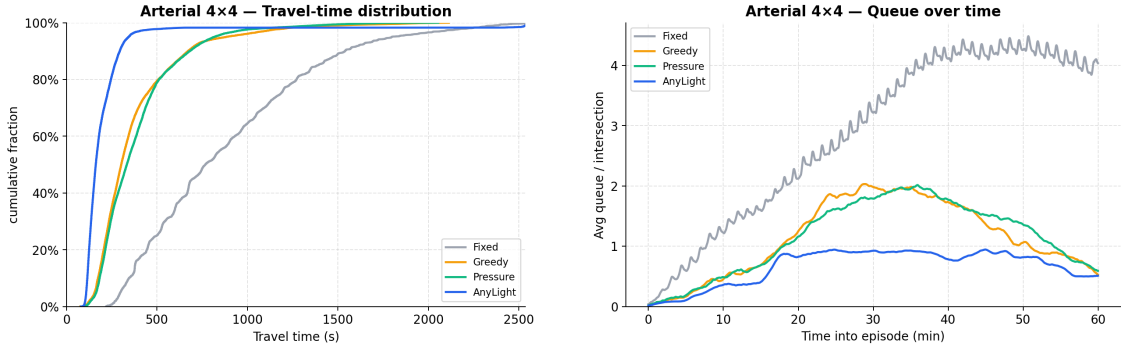
(a) Travel Time CDF

(b) Queue Dynamics

Figure 7.2: Behavioral and spatial analysis of the AnyLight policy on the grid4x4 network, illustrating temporal queue evolution and travel time distributions.

Arterial 4×4 Network						
Model	**Queue (veh) ↓**	**Speed (m/s) ↑**	**Int. Delay (s) ↓**	**Compl. Rate (veh/s) ↑**	**Trip Delay (s) ↓**	**Trip Time (s) ↓**
Fixed-Time	2.88 ± 0.03	1.32 ± 0.01	651.96 ± 7.83	0.2740 ± 0.0016	795.61 ± 9.22	883.03 ± 9.21
Greedy	1.14 ± 0.03	3.41 ± 0.06	196.80 ± 5.53	0.4566 ± 0.0036	294.13 ± 8.29	380.50 ± 8.28
MaxPressure	1.16 ± 0.03	3.29 ± 0.05	199.87 ± 4.20	0.5209 ± 0.0014	303.95 ± 5.70	390.61 ± 5.82
IDQN	2.34 ± 1.35	2.11 ± 2.04	125.30 ± 95.75	0.3086 ± 0.0914	493.84 ± 795.88	659.41 ± 791.90
IPPO	4.06 ± 2.00	1.07 ± 1.87	253.08 ± 166.95	0.11 ± 0.16	1488.89 ± 1112.30	1653.15 ± 1091.73
MPLight	1.61 ± 0.70	2.38 ± 0.64	115.34 ± 7.39	0.42 ± 0.0353	262.53 ± 197.70	465.77 ± 237.69
AnyLight	**0.75 ± 0.02**	**3.84 ± 0.09**	**86.29 ± 8.49**	**0.5330 ± 0.0036**	**148.09 ± 9.74**	**230.33 ± 9.64**

Table 7.2: Performance comparison on the Arterial 4×4 network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better.



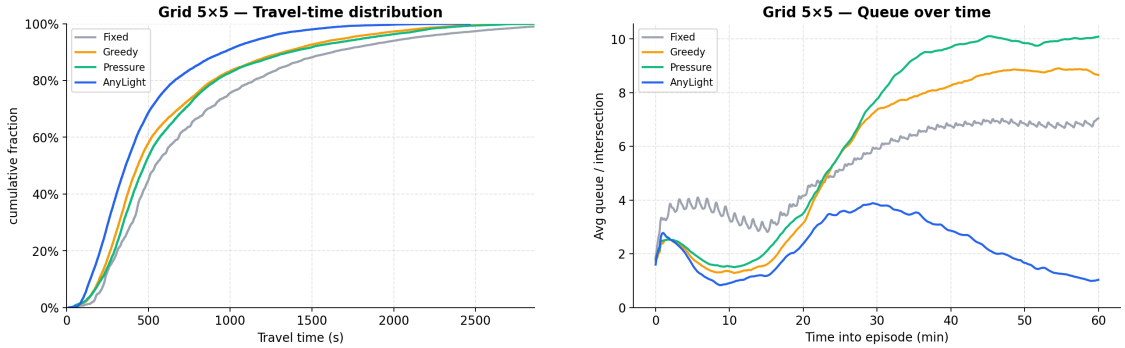
(a) Travel Time CDF

(b) Queue Dynamics

Figure 7.3: Behavioral and spatial analysis of the AnyLight policy on the arterial4x4 network, illustrating temporal queue evolution and travel time distributions.

Grid 5×5 Network						
Model	**Queue (veh) ↓**	**Speed (m/s) ↑**	**Int. Delay (s) ↓**	**Compl. Rate (veh/s) ↑**	**Trip Delay (s) ↓**	**Trip Time (s) ↓**
Fixed-Time	5.36 ± 0.31	1.09 ± 0.09	549.12 ± 34.24	0.6183 ± 0.0641	687.06 ± 36.29	770.64 ± 38.20
Greedy	5.76 ± 0.72	1.62 ± 0.19	374.29 ± 85.35	0.7417 ± 0.1247	509.23 ± 100.93	607.38 ± 103.05
MaxPressure	6.48 ± 0.40	1.37 ± 0.15	408.76 ± 87.55	0.6203 ± 0.0802	546.85 ± 97.28	644.87 ± 98.05
IA2C	3.94 ± 2.29	1.52 ± 1.09	81.49 ± 48.96	0.40 ± 0.26	546.28 ± 446.36	760.47 ± 510.71
MA2C	2.83 ± 1.49	2.14 ± 1.06	**38.44 ± 23.92**	0.65 ± 0.38	383.86 ± 346.76	595.30 ± 432.10
AnyLight	**2.22 ± 0.23**	**3.20 ± 0.23**	257.37 ± 32.27	**1.2370 ± 0.0431**	**364.01 ± 36.93**	**462.85 ± 38.11**

Table 7.3: Performance comparison on the Grid 5×5 network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better.



(a) Travel Time CDF

(b) Queue Dynamics

Figure 7.4: Behavioral and spatial analysis of the AnyLight policy on the grid5x5 network, illustrating temporal queue evolution and travel time distributions.

On the relatively simple Grid 4×4 network, overall congestion remains low, allowing both classical heuristics (such as Greedy) and advanced RL methods (like MPLight) to perform competently. Nevertheless, AnyLight minimizes network-wide queues to a near-zero average (0.05 vehicles) and maximizes the average vehicle speed (9.60 m/s), effectively matching or exceeding the best-performing baselines across all primary metrics.

In the Arterial 4×4 scenario, strong directional demand heavily penalizes independent learning agents. As shown in Table 7.2, IDQN and IPPO suffer from severe coordination failures due to multi-agent non-stationarity, resulting in massive trip delays and queue

accumulation. In contrast, AnyLight leverages its parameter-shared CTDE framework to enforce an implicit green-wave coordination along the main arterial, halving the trip delay (148.09 s) compared to the strongest classical baseline (Greedy, 294.13 s) and maintaining the highest completion rate in the group.

As the spatial scale and overall vehicle volume increase in the Grid 5×5 network, the most congested of the homogeneous environments, conventional multi-agent approaches like IA2C and MA2C struggle to prevent cascading spillback, capping their completion rates at roughly 0.65 veh/s. By explicitly accounting for downstream queues via its pressure-based collaborative reward, AnyLight sustains a remarkable throughput of 1.23 veh/s. This nearly doubles the operational capacity of the grid, ensuring that vehicles complete their trips significantly faster (462.85 s versus MA2C's 595.30 s) despite the heavy load.

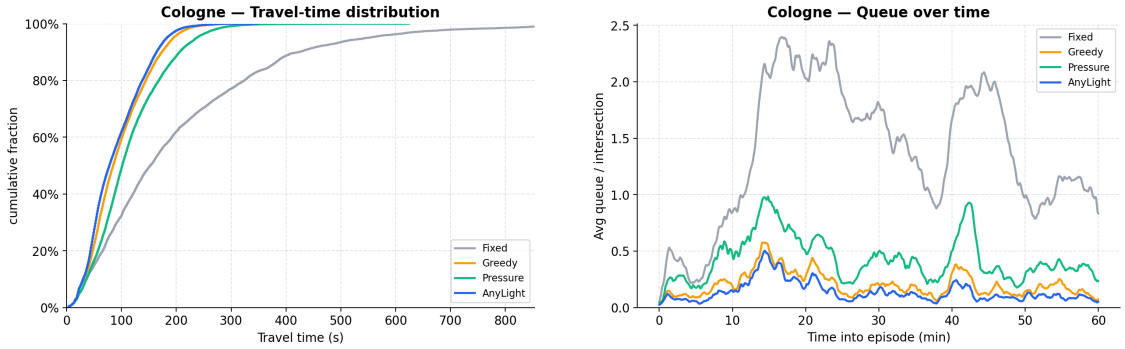
Overall, the quantitative results demonstrate that AnyLight consistently achieves superior traffic optimization across all three synthetic topologies, validating the efficacy of its movement-centric representation and collaborative reward structure. These numerical improvements are directly corroborated by the visual analysis in Figures 7.2, 7.3, and 7.4. The queue dynamics plots show stable, non-diverging queues throughout the simulation hour, effectively resisting gridlock even during peak inflow. Correspondingly, the steep travel time CDF curves indicate a high consistency in trip duration, while the spatial congestion heatmaps confirm that the policy prevents localized bottlenecks from propagating.

7.3 Performance on Heterogeneous Real-World Networks

The architecture is further evaluated on the heterogeneous real-world networks of Cologne, Ingolstadt, and Monaco. A single parameter-shared policy must simultaneously govern junctions with varying arm counts, lane configurations, and phase structures. Tables 7.4, 7.5, and 7.6 summarize the results.

Cologne Network						
Model	**Queue (veh) ↓**	**Speed (m/s) ↑**	**Int. Delay (s) ↓**	**Compl. Rate (veh/s) ↑**	**Trip Delay (s) ↓**	**Trip Time (s) ↓**
Fixed-Time	1.37 ± 0.03	4.10 ± 0.09	93.14 ± 3.38	0.5458 ± 0.0023	135.58 ± 4.46	203.83 ± 4.62
Greedy	0.20 ± 0.01	7.90 ± 0.04	11.80 ± 0.42	**0.5595 ± 0.0004**	29.76 ± 0.68	95.14 ± 0.72
MaxPressure	0.43 ± 0.02	6.78 ± 0.06	25.07 ± 0.95	0.5569 ± 0.0012	47.16 ± 1.16	113.33 ± 1.35
IDQN	11.48 ± 1.26	3.24 ± 0.06	137.43 ± 12.42	**0.5595 ± 0.0000**	230.38 ± 26.00	280.36 ± 21.17
IPPO	22.57 ± 0.44	2.16 ± 0.08	355.82 ± 25.91	**0.5595 ± 0.0000**	681.60 ± 81.77	420.60 ± 22.99
MPLight	12.00 ± 2.94	2.78 ± 0.09	241.39 ± 91.77	**0.5595 ± 0.0000**	320.32 ± 119.82	326.71 ± 84.11
AnyLight	**0.14 ± 0.00**	**8.41 ± 0.03**	**7.97 ± 0.31**	0.5594 ± 0.0004	**24.07 ± 0.52**	**89.40 ± 0.54**

Table 7.4: Performance comparison on the Cologne network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better.



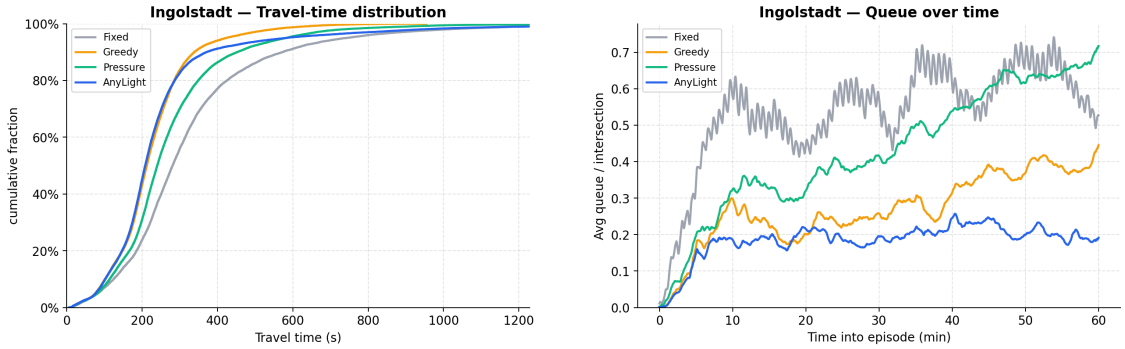
(a) Travel Time CDF

(b) Queue Dynamics

Figure 7.5: Behavioral and spatial analysis of the AnyLight policy on the cologne_region network, illustrating temporal queue evolution and travel time distributions.

Ingolstadt Network						
Model	**Queue (veh) ↓**	**Speed (m/s) ↑**	**Int. Delay (s) ↓**	**Compl. Rate (veh/s) ↑**	**Trip Delay (s) ↓**	**Trip Time (s) ↓**
Fixed-Time	0.54 ± 0.13	5.71 ± 0.45	121.10 ± 7.47	1.0206 ± 0.0619	182.35 ± 11.78	329.21 ± 14.59
Greedy	0.27 ± 0.16	6.06 ± 0.33	**40.37 ± 4.70**	0.9763 ± 0.1019	**81.28 ± 5.96**	**224.70 ± 5.82**
MaxPressure	0.43 ± 0.26	6.06 ± 1.37	76.67 ± 18.36	0.9385 ± 0.1355	126.27 ± 25.89	273.10 ± 28.83
IDQN	0.37 ± 0.09	5.45 ± 0.83	48.49 ± 19.00	**1.06 ± 0.49**	170.77 ± 520.62	380.30 ± 538.60
IPPO	1.48 ± 0.60	2.65 ± 1.95	108.82 ± 43.65	0.62 ± 0.33	732.29 ± 919.47	931.60 ± 937.90
MPLight	1.63 ± 0.50	2.22 ± 1.04	93.99 ± 26.55	0.57 ± 0.32	674.73 ± 838.57	882.31 ± 858.22
AnyLight	**0.22 ± 0.10**	**7.21 ± 1.10**	42.04 ± 5.84	1.0370 ± 0.0365	82.73 ± 6.32	230.32 ± 6.92

Table 7.5: Performance comparison on the Ingolstadt network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better.



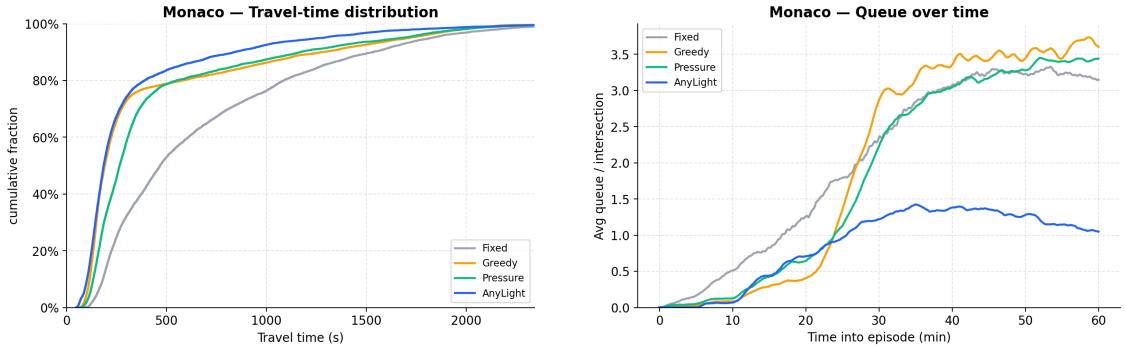
(a) Travel Time CDF

(b) Queue Dynamics

Figure 7.6: Behavioral and spatial analysis of the AnyLight policy on the ingolstadt_region network, illustrating temporal queue evolution and travel time distributions.

Monaco Network						
Model	**Queue (veh) ↓**	**Speed (m/s) ↑**	**Int. Delay (s) ↓**	**Compl. Rate (veh/s) ↑**	**Trip Delay (s) ↓**	**Trip Time (s) ↓**
Fixed-Time	2.02 ± 0.25	1.72 ± 0.08	475.05 ± 68.66	0.2819 ± 0.0530	592.50 ± 73.93	669.08 ± 73.09
Greedy	1.99 ± 0.11	2.96 ± 0.09	271.80 ± 54.40	0.2425 ± 0.0159	335.89 ± 57.50	414.57 ± 57.04
MaxPressure	1.85 ± 0.18	2.59 ± 0.06	292.08 ± 44.89	0.2502 ± 0.0143	370.20 ± 45.84	447.26 ± 46.36
IA2C	1.88 ± 1.12	2.14 ± 2.76	86.80 ± 48.27	0.22 ± 0.20	361.73 ± 527.91	482.28 ± 555.51
MA2C	1.54 ± 0.78	1.60 ± 1.45	**53.34 ± 28.88**	0.28 ± 0.20	456.80 ± 462.69	632.63 ± 505.03
AnyLight	**0.79 ± 0.19**	**3.79 ± 0.16**	169.24 ± 47.54	**0.3666 ± 0.0296**	**217.49 ± 51.16**	**295.34 ± 51.53**

Table 7.6: Performance comparison on the Monaco network, reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold. Arrows indicate whether lower (↓) or higher (↑) values are better.



(a) Travel Time CDF

(b) Queue Dynamics

Figure 7.7: Behavioral and spatial analysis of the AnyLight policy on the monaco network, illustrating temporal queue evolution and travel time distributions.

In the Cologne network, the traffic demand is characterized by sharp inflow surges. Here, classical actuated heuristics like Greedy perform surprisingly well, often outperforming the baseline RL models by a large margin. Nevertheless, AnyLight consistently surpasses or closely matches the best heuristics, achieving the absolute lowest network-wide queue (0.14 vehicles) and the highest average speed (8.41 m/s), reducing intersection delay by over 30% compared to the Greedy baseline.

Similarly, in the Ingolstadt network, the traffic flow exhibits high variability. AnyLight maintains superior throughput and speed while keeping trip delays tightly controlled. It

once again surpasses or closely matches the best heuristics, whereas independent RL models struggle to adapt to the fluctuations.

The Monaco network presents the most extreme architectural challenge, containing a mix of simple two-arm junctions and complex six-phase intersections. Fixed-dimensional RL models typically require extensive zero-padding or bespoke local models to handle such variance. In contrast, AnyLight effortlessly processes this heterogeneity through its movement-centric graph representation. The results highlight its superiority, AnyLight achieves a completion rate of 0.3666 veh/s, significantly outperforming both Greedy (0.2425 veh/s) and MA2C (0.28 veh/s), cutting trip delays by over a hundred seconds compared to the closest baseline.

Ultimately, the evaluation on real-world heterogeneous networks provides the definitive test for the architectural flexibility and robustness of the AnyLight policy. Traditional independent RL methods and standard multi-agent approaches completely collapse under these conditions, struggling with the severe non-stationarity and the inability to map diverse intersections to a unified observation space. The visual analysis confirms these achievements. The queue dynamics plots show that AnyLight prevents exponential queue growth during the heaviest inflow surges, and the spatial congestion heatmaps demonstrate that the policy effectively distributes the traffic load across irregular geometries, preventing critical bottlenecks from paralyzing the urban core.

7.4 Ablation Studies

To isolate the contribution of the individual architectural and reward design choices, the two ablated variants defined in Chapter 6 are compared against the full model: the variant without privileged neighbor information in the centralized critic, and the variant trained with the incoming-only reward formulation. Table 7.7 reports the results.

Variant	Queue (veh) ↓	Speed (m/s) ↑	Int. Delay (s) ↓	Compl. Rate (veh/s) ↑	Trip Delay (s) ↓	Trip Time (s) ↓
AnyLight (full)	**0.75 ± 0.02**	3.84 ± 0.09	**86.29 ± 8.49**	**0.5330 ± 0.0036**	**148.09 ± 9.74**	**230.33 ± 9.64**
w/o Neighbor Information	0.93 ± 0.01	**3.91 ± 0.06**	113.13 ± 4.20	0.52 ± 0.00	170.73 ± 4.09	254.91 ± 4.14
w/o Collaborative Reward	1.04 ± 0.01	3.64 ± 0.05	98.10 ± 3.92	0.51 ± 0.00	157.17 ± 3.93	240.75 ± 3.94

Table 7.7: Ablation study on the Arterial 4×4 network, comparing the full AnyLight model against its two ablated variants. Reported as the mean ± standard deviation over the evaluation episodes. The best value per metric is highlighted in bold.

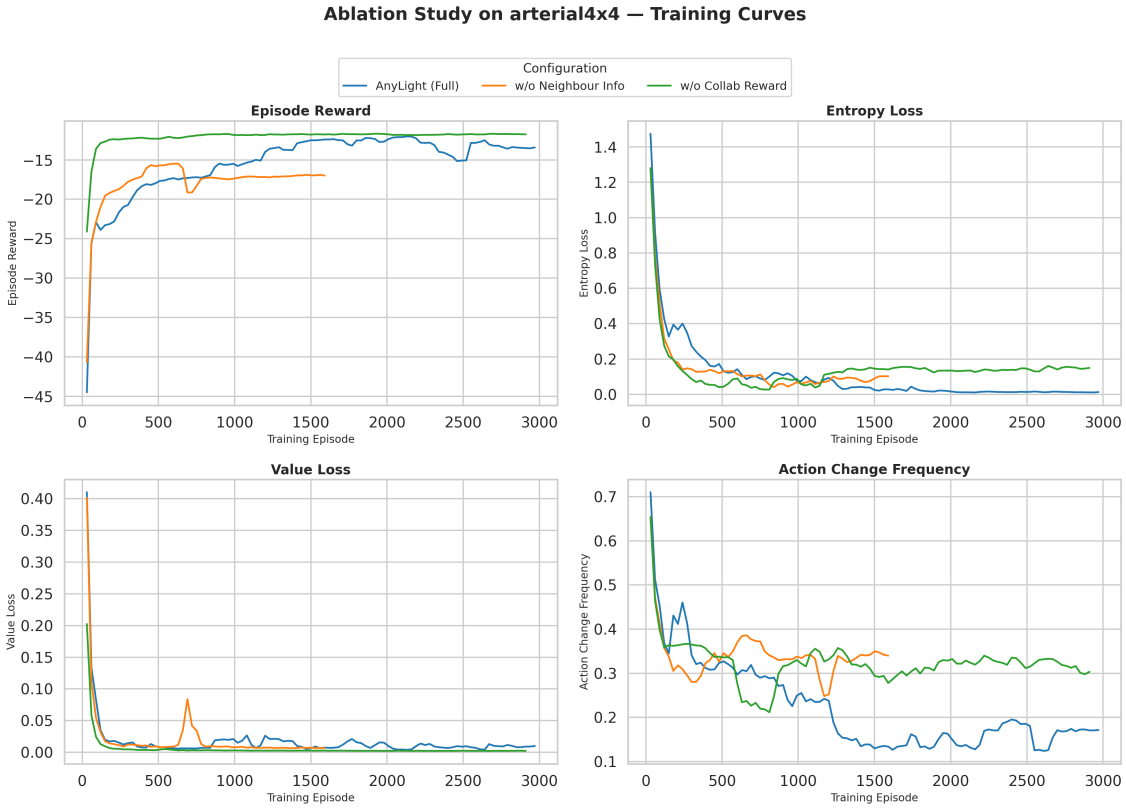


Figure 7.8: Training curves for the ablation study on the Arterial 4×4 network, comparing the full AnyLight architecture against variants missing neighbor information and the collaborative reward. The plots illustrate episode reward, entropy loss, value loss, and action change frequency over the training duration.

The results presented in Table 7.7 quantify the substantial performance degradation caused by removing either the centralized neighbor observations or the pressure-based collaborative reward. When the critic is deprived of privileged neighbor information, the policy loses its ability to anticipate incoming platoons. This omission leads to a sharp 31% increase in intersection delay (from 86.29 s to 113.13 s) and a noticeable drop in the overall completion rate. The model struggles to coordinate green waves along the arterial, defaulting to a reactive rather than proactive control strategy.

Similarly, eliminating the collaborative pressure term from the reward function causes a direct collapse in spatial coordination. Without an explicit penalty for pushing vehi-

cles into already congested downstream links, the network-wide queue length increases by nearly 40% (from 0.75 to 1.04 vehicles). This degradation is particularly severe in dense, highly interconnected topologies like the synthetic grids, where uncontrolled localized congestion rapidly cascades into widespread spillback. While sparse real-world networks might tolerate myopic routing to some extent, the collaborative reward proves essential for maintaining fluid traffic movement and maximizing throughput across complex urban environments.

To further contextualize these performance metrics, Figure 7.8 presents the training curves for the ablation study over the Arterial 4×4 network. Although the variant lacking the collaborative reward reaches a mathematically higher episode reward, this is merely an artifact of its simplified reward function, which omits the penalty terms for neighbor congestion. Its steep initial drop in entropy and persistently high action change frequency indicate a premature convergence into a suboptimal and oscillatory policy. In contrast, the full AnyLight architecture demonstrates a more robust and deliberate learning progression. The entropy decays gradually, ensuring comprehensive exploration of the state space, while the value loss stabilizes effectively despite the more complex learning target. Crucially, the full model achieves a significantly lower action change frequency, confirming that it learns a smooth and stable control strategy rather than the erratic phase-switching behavior exhibited by the ablated baselines.

7.5 Discussion of Results

The experimental evidence gathered across both homogeneous synthetic grids and heterogeneous real-world networks strongly supports the three primary contributions claimed in the Introduction. First, the movement-centric state representation successfully decouples the reinforcement learning architecture from the underlying intersection geometry. As demonstrated in the Cologne, Ingolstadt, and Monaco environments, a single unified architecture seamlessly handled junctions ranging from simple two-arm crossings to complex six-phase hubs without requiring any structural modifications, zero-padding, or bespoke local models.

Second, the universal parameter sharing paradigm proved highly effective. By centralizing the learning process, AnyLight maintained a constant parameter count regardless

of the network size. Despite this drastically reduced footprint, it matched or exceeded the performance of fully independent agents and topology-specific baselines, effectively combating the multi-agent non-stationarity that crippled standard approaches like IDQN and IPPO in scenarios with strong directional demand.

Third, the integration of a pressure-based collaborative reward structure yielded critical improvements in network-wide throughput. The ablation studies confirmed that explicitly coupling the reward signal to neighborhood queue dynamics prevents the myopic routing behaviors typical of isolated agents. This neighborhood-aware formulation proved essential in dense grids, where it successfully prevented localized congestion from cascading into widespread spillback and nearly doubled the operational capacity of the network compared to conventional multi-agent algorithms.

8

Conclusions and Future Work

This thesis investigated the design of a single, generalizable Multi-Agent Reinforcement Learning controller for urban Traffic Signal Control on heterogeneous road networks. The motivating observation was that the deployed control infrastructure in most cities, dominated by fixed-time schedules and rule-based actuated systems such as SCATS and SCOOT, cannot react to the non-stationary, spatially coupled demand patterns of modern traffic, while the multi-agent learning literature has so far converged on solutions that either replicate a private network per intersection or impose rigid, fixed-dimensional state and action spaces that exclude real-world topologies by construction.

To address this gap, we introduced AnyLight, a Multi-Agent Proximal Policy Optimization architecture that decomposes every intersection into its elementary traffic movements and processes them through a single, parameter-shared network. The model combines a movement-centric observation with a zero-padding and masking strategy that uniformly accommodates variable numbers of movements and phases, and a cross-attention decoder that treats phase selection as an information-retrieval problem over the current congestion state. Under the Centralized Training, Decentralized Execution paradigm, the centralized critic is further granted privileged access to neighbor-action vectors, encoded through the same movement-centric scheme, so that the value estimate can account for upstream traffic released by adjacent intersections without forcing the actor to depend on global information at execution time.

8.1 Summary of Contributions

The three contributions claimed in the Introduction are revisited here in light of the experimental evidence reported in Chapter 7:

- **Movement-Centric Representation.**
  The same network architecture controlled every benchmark scenario without any topology-specific modification, including three-arm, four-arm, and highly asymmetric intersections in Cologne, Ingolstadt, and Monaco. This validates the structural claim that elementary traffic movements form an adequate, topology-invariant unit of representation for signalized intersections.

- **Universal Parameter Sharing.**
  A single Actor-Critic model with a constant parameter count, independent of the number of intersections, was sufficient to match or exceed both classical and topology-specific learning baselines across the six evaluation scenarios. The ablation removing privileged neighbor information further isolated the contribution of the collaborative critic to this result.

- **Collaborative Reward Structure.**
  By coupling the centralized critic with a pressure-based collaborative reward system, the architecture successfully induced implicit network-wide coordination. The ablation studies confirmed that this neighbor-aware formulation was critical for effectively mitigating gridlock and clearing dense traffic far more efficiently than independent, localized reward strategies.

8.2 Limitations

Several limitations qualify the scope of these findings and should be transparently acknowledged.

First, all evidence is simulation-based. SUMO provides a microscopic, calibrated approximation of real traffic, but it does not reproduce sensor noise, communication latency, or actuator failure that any deployed system would have to tolerate. Second, the observation space assumes idealized lane-area detectors with a fixed 50-meter range and noise-free queue counts; in practice these counts must be reconstructed from imperfect inductive

loops, magnetometers, or camera-based estimators, and the policy has not been stress-tested against the resulting input perturbations. Third, the action space operates at a fixed decision interval and only over green-phase selection, with yellow transitions managed by the environment wrapper; deployments that require finer-grained timing, pedestrian phases, or priority signals would require an extension of the action interface. Finally, the largest network evaluated contains thirty intersections, which is representative of a dense city subregion but not yet of an entire metropolitan area, and the largest transfer was performed between networks of comparable scale.

8.3 Future Work

The architecture and the empirical results suggest several promising research directions to further enhance the capabilities of AnyLight.

Intersection-Aware Parameter Sharing.
A known limitation of fully parameter-shared architectures is their tendency to regress to the mean, occasionally settling into a sub-optimal policy that averages behaviors across all agents. To mitigate this issue, future work could explore injecting topological embeddings or unique structural identifiers into the state representation. Providing the network with explicit context about the specific junction it is operating would allow the single shared model to dynamically tailor its response to the local topology, overcoming the average-policy bottleneck while retaining the immense sample efficiency of centralized training.

Zero-Shot Generalization via Co-Training.
While AnyLight has demonstrated remarkable architectural flexibility when trained and tested within the same heterogeneous environment, its absolute generalization limits remain unexplored. A compelling next step is to co-train the architecture simultaneously over a diverse suite of distinct urban networks, allocating one meta-agent per network environment, and subsequently evaluate its zero-shot transfer capabilities on completely unseen topologies. This experimental setup would rigorously test whether the movement-centric representation successfully captures universal traffic-control heuristics.

Scaling to City-Scale Networks.

The constant parameter count of the shared model, combined with the distributed Ray-based experience-collection pipeline, makes scaling beyond the thirty-intersection Monaco scenario primarily an engineering problem rather than an architectural one. A clear extension is to train on a city-scale network and quantify how the marginal benefit of parameter sharing evolves as the agent population grows.

Continuous and Hybrid Action Spaces.
Replacing discrete green-phase selection with a continuous green-duration head, or with a hybrid head that jointly selects phase and duration, would let the policy modulate decision frequency in response to demand and would align the controller more closely with the actuation interface of real signal hardware.

Towards Real-World Deployment.
A practical deployment path would proceed through hardware-in-the-loop validation, shadow-mode operation against an existing actuated controller, and, finally, supervised activation on a limited corridor. Each of these stages introduces engineering, safety, and regulatory constraints that fall outside the scope of this dissertation, but the movement-centric, parameter-shared design of AnyLight is explicitly intended to make this transition tractable: the same policy, without retraining, can be moved from one signalized junction to the next as the controller is rolled out across the network.

References

[1] Jeancarlo Arguello Calvo and Ivana Dusparic. Heterogeneous multi-agent deep reinforcement learning for traffic lights control. In *Proceedings of the 1st International Workshop on Agent-Based Modelling of Urban Systems (ABMUS)*, 2018.

[2] James Ault and Guni Sharon. Reinforcement learning benchmarks for traffic signal control. In *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track*, December 2021.

[3] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[4] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

[5] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Mplight: Efficient city-wide traffic signal control via knowledge transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 2255–2266, 2020.

[6] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent actor-critic for integrated traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(1):251–262, 2019.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[8] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[10] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton. SCOOT - a traffic responsive method of coordinating signals. Technical Report 1014, Transport and Road Research Laboratory, Crowthorne, Berkshire, UK, 1981.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191, 2020.

[13] Yilin Liu, Guiyang Luo, Quan Yuan, Jinglin Li, Lei Jin, Bo Chen, and Rui Pan. Gplight: Grouped multi-agent reinforcement learning for large-scale traffic signal control. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 199–207, 2023.

[14] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.

[15] P. R. Lowrie. SCATS, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. Technical report, Roads and Traffic Authority NSW, Sydney, Australia, 1990.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[17] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications.

In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[18] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[19] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, second edition, 2018.

[22] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[24] F. V. Webster. Traffic signal settings. Technical Report 39, Road Research Laboratory, London, UK, 1958.

[25] Axel Wegener, Michal Piórkowski, Maximilien Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. TraCI: An interface for coupling road traffic and network simulators. In *Proceedings of the 11th Communications and Networking Simulation Symposium*, pages 155–163. ACM, 2008.

[26] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD International*

Conference on Knowledge Discovery & Data Mining (KDD), pages 1290–1298, 2019.

[27] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Cheng, Kan Yan, Hailin Li, Jie Ding, et al. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922, 2019.

[28] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[29] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

[30] Jinwei Zeng, Chao Yu, Xinyi Yang, Wenxuan Ao, Qianyue Hao, Jian Yuan, Yong Li, Yu Wang, and Huazhong Yang. Citylight: A neighborhood-inclusive universal model for coordinated city-scale traffic signal control. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 2025.

[31] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.

[32] Yifeng Zhang, Peizhuo Li, Mingfeng Fan, and Guillaume Sartoretti. Heterolight: A general and efficient learning approach for heterogeneous traffic signal control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1010–1017. IEEE, 2024.