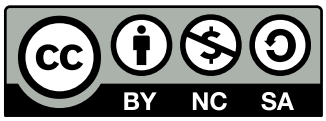
Aalto University
School of Electrical
Engineering

Master's programme in ICT Innovation

Highway Autopilot Using Deep Reinforcement Learning and Graph Neural Networks

Harikrishnan Devaraj

Master's Thesis
2023



Aalto University
School of Electrical
Engineering

Author Harikrishnan Devaraj

Title Highway Autopilot Using Deep Reinforcement Learning and Graph Neural Networks

Degree programme ICT Innovation

Major Autonomous systems

Supervisor Prof. Ville Kyrki

Advisors Börve Erik, Gökhan Alcan, Pathare Deepthi

Collaborative partner Volvo Group Trucks Technology

Date 20/11/2023 **Number of pages** 46+1 **Language** English

Abstract

Autonomous vehicles hold promise for enhancing road traffic safety by minimizing human errors through optimal decision-making approaches. The development of decision-making algorithms for complex traffic scenarios is crucial for the successful implementation and wider adoption of autonomous vehicles. In the context of a multi-laned highway traffic scenario, lane changing is a challenging coordination problem due to high speeds and the variation in the number of surrounding road users. Further, a robust input representation of the highway traffic is required for the autonomous vehicle to make optimal decisions for safe navigation. Learning-based or data-driven methods offer a promising solution for developing these robust decision-making algorithms.

In this thesis, the feasibility of developing a low-level controller model based on deep reinforcement learning and graph neural networks which can be used to safely change lanes in a highway with traffic in a simulated environment is investigated. The dependence on the input graph traffic configuration on the model's performance is shown for a given simulation environment configuration. Further, the usefulness of the graph attention mechanism for understanding the decision-making of the trained policy is explored.

Keywords Graph neural networks, Soft actor-critic, Graph attention network

Preface

I dedicate this work to my family and friends for their immense support during the whole time period of my Master's degree. Further, my heartfelt thanks to my Supervisor, advisors, and Volvo Trucks for their invaluable guidance and the opportunity to work on this project. Lastly, I would like to acknowledge the use of computational resources provided by the Aalto Science-IT project, without which this thesis would not have been possible.

Otaniemi, 20 November 2023

Harikrishnan Devaraj

Contents

Symbols and abbreviations

Symbols

A_{ij}	Attention coefficient for GAT
α_{ij}	Normalized attention coefficient for GAT
α	Entropy temperature term of SAC
b	Collision penalty weight
c	Right lane reward weight
E	Set of graph edges
γ	Discount factor
h	High speed reward weight
$\mathbf{h_i}$	Feature representation of node i for GAT
G	Graph network
K	Number of attention heads
N_i	Neighbours of node i
π	Policy
ϕ	Graph network update functions
$Q(s, a)$	State-action value function for state s and action a
$R(s, a)$	Reward function
ρ	Graph network aggregate functions
T	State transition model
τ	Target smoothing coefficient for SAC
$V(s)$	State value function for state s
V	Set of graph nodes
$\mathbf{W}$	Learnable weight matrix for GAT

Operators

$\|\|$	Concatenation operator
$\mathbb{E}$	Expectation operator
∇_θ	Gradient operator with respect to parameter θ
$LeakyReLU$	Leaky Rectified Linear Unit function
$ReLU$	Rectified Linear Unit function
$\sum_i$	Sum over index i
A^T	Transposition or Transpose of matrix A
$\bigcup_i E'_i$	Union of sets E'_i for all i

Abbreviations

AD	Autonomous Driving
AI	Artificial intelligence
AV	Autonomous vehicles
BERT	Bidirectional Encoder Representations from Transformers
CARLA	Car Learning to Act
DQN	Double Q-Network
D3QN	Double Deep Q-Network
DRL	Deep reinforcement learning
GAT	Graph Attention Neural Network
GCN	Graph Convolutional Neural Network
GN	Graph Nets
GPU	Graphics Processing Unit
LiDAR	Light Detection and Ranging
MDP	Markov decision processes
MPNN	Message passing neural network framework
NLNN	Non-local neural network framework
NN	Neural Network
POMDP	Partially observable Markov decision processes
RL	Reinforcement learning
RPF	Randomized Prior Functions
RTI	Road Traffic Injuries
SAC	Soft Actor-critic off-policy algorithm
SUMO	Simulation of Urban MObility
TORCS	The Open Racing Car Simulator

1 Introduction

According to the Global Status Report on Road Safety from the World Health Organization (WHO), around 1.35 million people die each year as a result of road traffic injuries (RTI) as shown in Figure 1. This surpasses the mortality rates of HIV/AIDS, tuberculosis and diarrhoeal diseases individually. In addition to that, RTI is the leading cause of death among children and young adults in the age range of 5-29 years, making it a significant concern for public health and societal well-being [1]. The impact of RTI extends beyond the loss of human life, as they result in profound emotional, social and economic consequences for families and communities around the globe [1]. The report further states human error due to speeding, driving under the influence of psychoactive substances and distracted driving are one of the main causes of this issue.

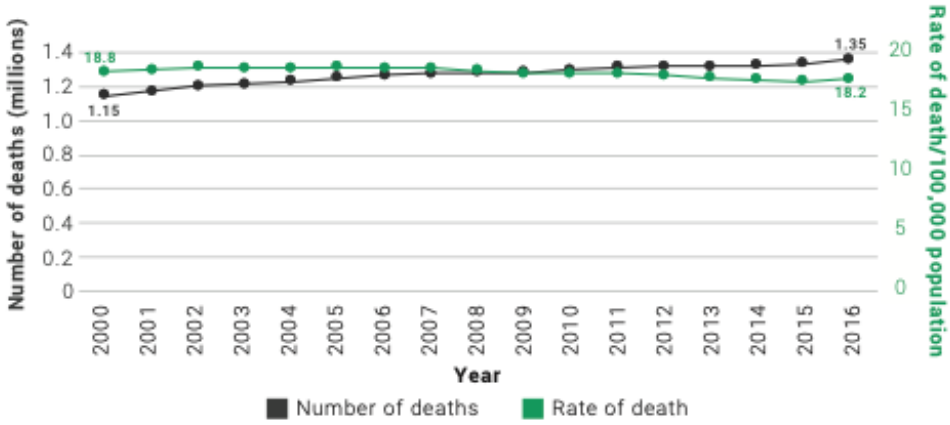


Figure 1: Number and rate of road traffic death per 100,000 population: 2000-2016 [1].

One possible way to improve human safety and reduce RTI is to develop Autonomous vehicles (AV) with better decision-making methods which can safely navigate complex driving scenarios encountered in the real world thereby mitigating the human error in decision-making [2]. Classical decision-making methods are not always effective in complex uncertain environments due to their limited adaptability and poor robustness. Learning-based decision-making methods mitigate the limitations of classical approaches by leveraging data-driven algorithms which helps the AV to make constantly improving decisions in real-time [3]. Deep reinforcement learning (DRL) is one commonly used learning-based method that shows promising results in complex driving scenarios like lane changing in highways using methods like Ensemble RPF and Double Q-learning (DQN) [4] [5]. With respect to the constraints of the simulation environment, using DRL the AV can learn from simulated experiences and improve its decision-making capabilities over time, leading to more efficient and informed actions.

Generally in DRL, deep neural networks help to scale decision-making problems with high-dimensional state and action spaces that were previously intractable [6]. But many of the DRL approaches which use neural networks for decision-making

in the domain of autonomous driving use vectorial representations as input data and this introduces a major constraint to predefine the order and the maximum number of vehicles in the simulation environments [7]. Since in the real world, the maximum number and the order of vehicles can change dramatically, this constraint could be an issue in the effectiveness of DRL-based controllers to reduce RTIs. However, graph neural networks (GNN), which are suitable for modelling non-euclidean systems like traffic networks [8] and capture interactions between agents in the graph network provide a promising solution in this context. They excel at processing graph-based inputs, allowing them to handle dynamic variations in the order and number of vehicles surrounding the autonomous vehicle [7]. By leveraging the power of GNNs, DRL-based controllers can benefit from the ability to capture complex relationships and dependencies among vehicles, leading to improved decision-making and enhanced safety in reducing RTI.

The main goal of this thesis is to develop a highway autopilot by employing GNN to model traffic dynamics and learning control policy through DRL which will set low-level control values like throttle acceleration and steering angle based on the state information of the autonomous vehicle and surrounding vehicles. In addition to that, one research goal of this thesis is to know how can GNN be used in a highway autopilot. In order to achieve both of the development and research goals, existing methods in the literature regarding DRL and GNN in the domain of autonomous driving should be analyzed. This will help in identifying which suitable combination of graph neural network, deep reinforcement learning algorithm and simulation environment can provide the optimal solution for developing the highway autopilot. Further, this thesis aims to investigate two research questions using the trained highway autopilot policy network. For training the autonomous vehicle, different configurations of the input graph can be constructed using the same output values like the position and velocity of the vehicles from the simulation. So, it needs to be investigated if the choice of the input graph structure affects the performance of the learned policy. Further, given that the machine learning systems often behave as black-boxes [9], another research problem is to see if there is a way to gain any insight into the decision-making processes by analyzing the GNN component of the model. These are the questions that will be investigated in this work.

This thesis document is divided into six sections. The next section 2 introduces the background related to the domains of reinforcement learning, graph neural networks and looks into previous methods done in this approach to autonomous driving. Then in section 3, the specific details about the algorithms, simulation environment and implementation details are explained in detail. Next in the experiments presented in section 4, the learned policy is used to investigate the research questions addressed in the above paragraph. The discussion in section 5 addresses the effectiveness of using a graph neural network, the limitations and future approaches to this method and lastly, the conclusions of this thesis are presented in the final section 6.

2 Background

In this section, a generic background regarding DRL and GNN is shared and then the current approaches using them in the domain of Autonomous driving (AD) are introduced. The general overview of this section includes, first a brief introduction to deep reinforcement learning and graph neural networks, then the current research in this domain is investigated and analysed to see which direction can be taken to tackle the research problems mentioned in the previous section.

2.1 Deep Reinforcement learning

Machine learning-based approaches have made significant progress in the field of artificial intelligence in the last two decades [10]. These approaches can generally be divided into three categories: (i) Supervised learning, (ii) Unsupervised learning and (iii) Reinforcement learning [11]. Supervised machine learning involves solving problems where the training dataset is fully labelled and the task is to predict a specific outcome based on the given labels [12]. In contrast, unsupervised learning involves solving problems where the training dataset does not have explicit labels or predefined outcomes and the main goal is to uncover patterns, structures or relationships within the data itself [12]. These two approaches have made significant advances in areas like computer vision [13] and natural language processing [14].

Reinforcement learning (RL) is a principled mathematical framework to solve problems in which an autonomous agent learns to make optimal decisions in a real or simulated environment to maximize a cumulative reward signal through trial and error[6]. However, due to complexity issues related to computation, memory and sampling, classical RL algorithms are difficult to scale and their use case is limited to fairly low-dimensional problems [6]. Deep reinforcement learning helps to mitigate this problem by leveraging the function approximation and representation learning properties of deep neural networks [6] thereby improving the scalability of RL to decision-making problems with high-dimensional state and action spaces. This new paradigm has made it possible to develop groundbreaking AI systems like AlphaZero [15] which has defeated human world champions in games like chess, shogi and Go.

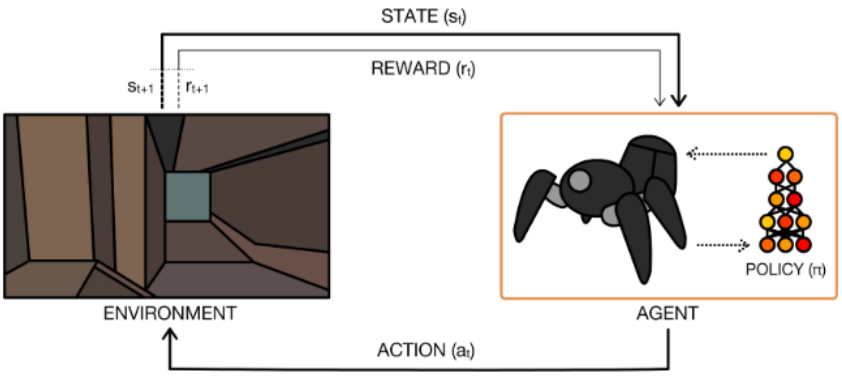


Figure 2: The perception-action learning loop in reinforcement learning from [6].

In general, a reinforcement learning problem can be formulated as an instance of a Markov decision process (MDP) [16]. An autonomous agent, as shown in Figure 2 interacts with the environment at timestep t by taking an action a_t in the state s_t of the environment. Depending on the current state and chosen action, the agent and the environment transition into a new state s_{t+1} and get a scalar reward r_{t+1}. The agent's main aim is to learn a control strategy or policy π such that its interactions with the environment maximizes the cumulative reward.

Generally, the policy π can be considered as a mapping from the states to a probability distribution over actions: π: $s \rightarrow p(A = a|s)$ [6]. For episodic tasks, in which the state gets reset after each episode of fixed length T, the sequence of states, actions and rewards in a given episode is known as the rollout or trajectory of the policy [6]. The rewards accumulated by the autonomous agent from its interactions with the environment can be defined as :

$$R = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \tag{1}$$

where $\gamma \in [0, 1]$, is the discount factor which determines the weight of the future rewards compared to immediate rewards. Using this formalism the optimal policy π^*, which the autonomous agent tries to learn in order to maximize the expected return from all the states [6] is given by :

$$\pi^* = \arg\max_{\pi} E[R|\pi] \tag{2}$$

One key concept in RL is the Markov property, which states that the future state s_{t+1} is conditionally independent of the past states $(s_0, s_1, s_2 \ldots s_{t-1})$ given the present or current state s_t [16]. This implies that by focusing on the current state which contains all the relevant information needed to make the decisions about future actions, the agent can maximize its expected return. Now for this assumption to hold, the environment states should be fully observable to the agent at each time step, which is unrealistic [6] because the decision-making process in the real world is affected by noisy or incomplete information about the system's state. Partially observable Markov decision

processes (POMDP) is a generalization of MDP which can be a useful formalism to solving problems that are characterized by uncertainty and partial observability [16]. The problem of finding the optimal policy in reinforcement learning can be formalized as the optimal control of an incompletely-known Markov decision processes [16].In the case of a POMDP, the agent does not have access to the true state of the environment, therefore it uses a belief state to represent its uncertainty about the true state, meaning the true state can only be estimated [6]. This Belief state is generally a probability distribution of the possible states of the environment and POMDP algorithms maintain a belief over the current state given the action taken, the previous belief state and the current observation [6]. This formalism has been particularly useful in various real-world applications in robotics, autonomous underwater vehicles [17], games and natural language processing [18].

Generally, reinforcement learning problems are approached using two fundamental methods: Model-based methods and Model-free methods [16]. Model-based methods involve constructing an internal model of the environment to simulate its dynamics and plan future actions without interacting with the environment directly [6]. However, in Model-free methods, learning is directly based on interactions with the environment and the agent does not use a learned model of the environment's dynamics [16]. Model-free methods are generally divided into three methods: Value function-based methods, Policy-based approaches and Actor-Critic methods [6].

Value-based methods aim to learn the optimal value function, which estimates the expected cumulative reward that an agent can obtain from a given state while following a specific policy [6]. The Bellman equation [16] expresses the relationship between the value of a state s and the value of its neighbouring states. It can be used to recursively compute the value function of the states in terms of the immediate reward and the value of the next state. The Bellman equation for taking the best action a in a state s and transitioning into a new state $s^{'}$ is defined as [16]:

$$V(s) = \max_{a} \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right) \tag{3}$$

where $R(s,a)$ is the immediate reward obtained, $P(s'|s,a)$ is the transition probability from s to s' after taking action a and $V(s')$ represents the new state's value function. Q-learning and Deep Q-Networks are some of the well-known examples of value-based reinforcement learning algorithms [6].

Policy-search methods explicitly focus on learning the optimal policy π^*, instead of maintaining a value function model like in the case of value-function method [16]. The main goal here is to find the policy that maximizes the expected cumulative reward over time [6]. Policy-based methods parameterize the policy using a set of parameters and use optimization techniques to update these parameters in a direction that increases the expected cumulative reward [16]. The policy gradient, which represents the gradient of the expected cumulative reward with respect to the policy's parameters can be

defined as [16]:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^{\pi_\theta}(s_t, a_t)\right] \tag{4}$$

where θ are the policy parameters, $\pi_\theta(a_t|s_t)$ and $Q^{\pi_\theta}(s_t, a_t)$ under policy π_θ are the probability of taking action a_t in state s_t and the state-action value function or the Q-function under policy while τ represents a trajectory of states and actions sampled from the policy. Reinforce and Proximal policy optimization (PPO) are some examples of policy-based approaches [6].

The Actor-critic method is a hybrid approach which uses both value functions and policy search to solve reinforcement learning problems [6]. The actor or the policy proposes actions based on the current state of the environment and the critic or the value function evaluates the values of these actions based on the expected future rewards. Depending on the critic's evaluation the actor updates its current policy. This feedback loop allows this method to introduce a trade-off between variance reduction and bias introduction which makes the learning process more stable and efficient [19]. Advantage actor-critic (A2C) algorithm and Soft actor-critic (SAC) are some of the well-known examples of this hybrid method [6].

2.2 Graph Neural Networks

A graph is a non-euclidean space data structure which represents a collection of entities known as nodes, the connections or interrelationships between them are known as edges and the global attributes provide information that pertains to the entire structure. This structure makes them a good framework for organizing, storing and processing information about real-world complex systems like molecules, traffic networks and biological structures like the brain, vascular system and nervous system [8]. Even data which has rich information like images or text can be converted into graph data to learn more about the unseen symmetry, structure and inter-dependencies between the various elements or components present within the data as shown in Figure 3.

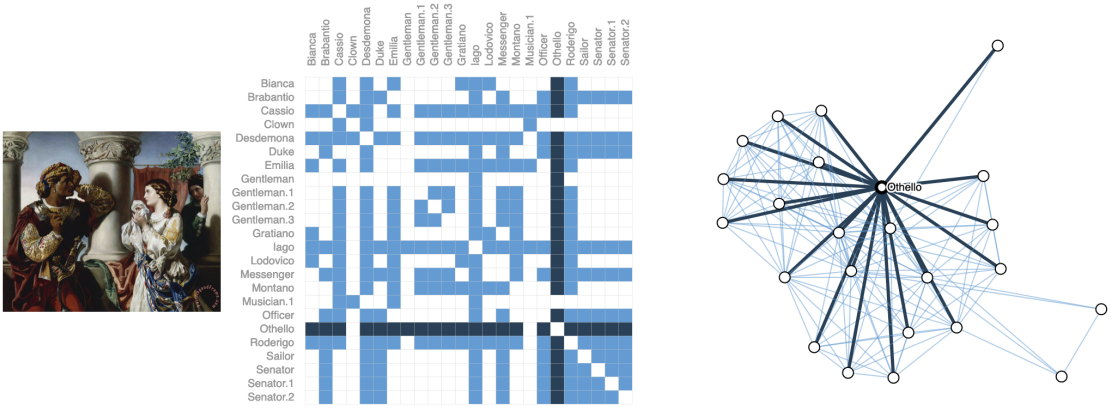


Figure 3: Graph representation of interactions from the play Othello with adjacency matrix from [20].

Graph neural networks are a subset of deep learning algorithms designed to capture and exploit structural information present in graphical data to make predictions, classify nodes and uncover hidden patterns which may not be available in grid-like data such as images and sequences [20]. They operate on the individual graph components like the nodes and edges in such a way that preserves the inherent graph symmetries like permutation invariance and rotation invariance [8]. This property allows GNNs to effectively capture the local and global structural patterns within the graph data, making them a versatile tool for a wide range of graph-related tasks [8]. One of the first major advancements in this field was the Message-passing neural network (MPNN) framework proposed by Gilmer et al. [21] as a general framework for predicting the quantum properties of organic molecules. Using this framework they popularized the idea of aggregating information using aggregate functions from the neighbours in the graph, which became a cornerstone of subsequent GNN models [8]. This aggregated information is then used to update the node features in subsequent iterations and the particular choice of aggregate functions and update functions allows the MPNNs to capture information from both local neighbourhoods and the overall graph structure [21]. In addition to that, Battaglia et al. [22] introduced the Graph nets (GN) framework which generalizes the concept of GNNs and emphasized the importance of incorporating relational inductive biases in deep learning while understanding how the nodes, edges and global attributes in the graph data can be harnessed for modelling complex relationships. Incorporating the relational inductive biases helps capture the inherent relationships and dependencies that are present in the graph data and helps the model in making meaningful predictions [8].

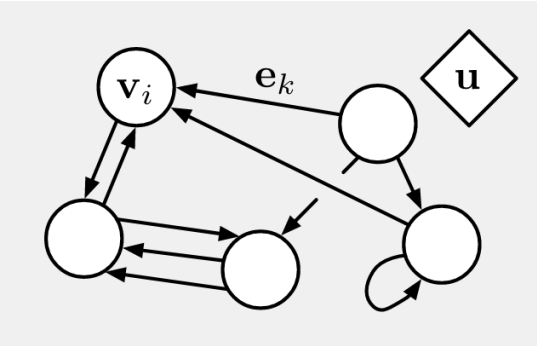


Figure 4: A directed, attributed multi-graph G in the Graph Network framework from [22].

Generally in the Graph network framework [22], a graph G with its global attribute u, set of nodes V and set of edges E is defined as :

$$\begin{aligned} G &= (u, V, E) \\ V &= \{v_i\}_{i=1:N^v} \\ E &= \{(e_k, r_k, s_k)\}_{k=1:N^e} \end{aligned} \tag{5}$$

in which v_i represents the node's attribute, e_k represents the edge's attribute, r_k and s_k represent the index of the receiver node v_{r_k} and sender node v_{s_k} respectively.

In addition to that the internal structure of a GN block includes three update functions, ϕ and the three aggregation functions, ρ for each of the three components of the graph data [22]:

$$\begin{aligned} e'_k &= \phi^e(e_k, v_{r_k}, v_{s_k}, u) & \bar{e}'_i &= \rho_{e\to v}(E'_i) \\ v'_i &= \phi_v(\bar{e}'_i, v_i, u) & \bar{e}' &= \rho_{e\to u}(E') \\ u' &= \phi_u(\bar{e}', \bar{v}', u) & \bar{v}' &= \rho_{v\to u}(V') \end{aligned} \tag{6}$$

where $E'_i = \{(e'_k, r_k, s_k)\}_{r_k=i,k=1:N^e}$ is the set of resulting per-edge outputs for each node i , $V' = \{v'_i\}_{i=1:N^v}$ is the set of resulting per-node outputs and $E' = \bigcup_i E'_i = \{(e'_k, r_k, s_k)\}_{k=1:N^e}$ is the set of the all per-edge outputs [22].

By iteratively applying the node, edge and global functions, the GNN model can capture patterns at various scales and levels of abstraction [22]. The GN framework provides a versatile architecture for learning and reasoning on graph-based data and introduced a broader framework that encompasses various GNNs as well as other graph-network frameworks like the Non-local neural network (NLNN) which unified various self-attention style methods as well as the MPNN framework [22]. Some notable examples of GNN models include Graph convolutional network (GCN) [23], GraphSAGE [24], Gated graph Neural networks (GGNN) [25] and Graph Attention networks (GAT) [26].

2.3 Literature Review

In this subsection, the previous research endeavours in the field of scenario-based policy learning for AD is outlined. The primary emphasis will be learning-based methods, specifically those employing DRL or a fusion of DRL with GNN. This approach allows for an in-depth analysis of existing methodologies, providing insights into the current state of the field and guiding the exploration of potential directions for the development of the highway autopilot.

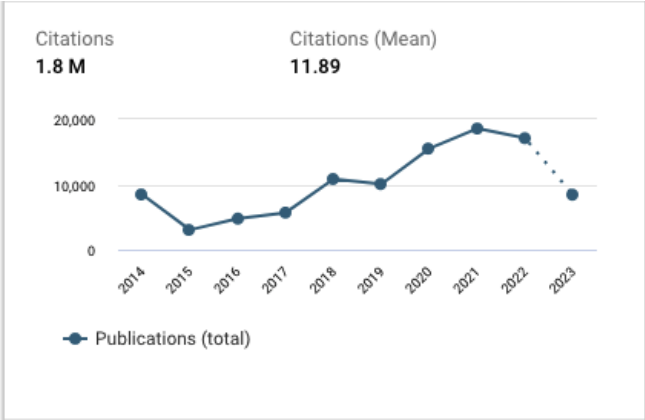


Figure 5: Recent publication trends for the following keywords: "deep reinforcement learning" AND ("autonomous vehicles" OR "autonomous cars") from [27].

Since the past decade as shown in Figure 5, deep learning based approaches have been used extensively to solve various issues related to perception, localization, path planning and developing learning-based motion controllers in the domain of autonomous driving [27] [28]. Deep reinforcement learning-based approaches have been used in problems related to decision-making and planning, for example in areas like path planning, driving policy and control optimization problems [4]. These methods have also been used for scenario-based policy learning in simulated environments like highways, intersections, merges and splits in which an autonomous vehicle or ego-vehicle learns the optimal vehicle driving policy depending on the presence of external traffic actors like pedestrians, other vehicles and road boundaries [4].

Generally in a scenario-based policy learning, an optimal vehicle policy is learned to control continuous-value actuators like steering angle, throttle, brake or discrete actuators like gear changes of the ego-vehicle [4]. One common method is using a reinforcement learning algorithm like Deep Q-Network (DQN) in a discrete action space, where values of the continuous actuators are discretized depending on the environment scenario [4] [29]. In an intersection driving scenario, Tram et al. [30] used deep Q-learning to develop a policy which infers the speed and distance between other vehicles to cross the intersection with a collision avoidance rate of 98 % while using a set of discrete action values. In addition to that Isele et al. [31] proposed another deep Q-learning-based method to safely navigate unsignaled intersections which surpassed in performance, task completion and goal success rate compared to a heuristic approach.

Now, finding an optimal vehicle policy for a scenario like a multi-laned highway is also a common area of deep reinforcement learning research in autonomous driving [4] [5] [32]. Hoel et al. [33] proposed a double deep Q-network with a convolutional neural network (CNN) based solution to handle speed and lane change decisions for a truck-trailer combination in which they showed the generality of the solution by training the policy on a real-world road overtaking scenario with oncoming traffic. The discrete action space here consists of values depending on which lane the ego-vehicle should accelerate, change to or keep the current speed. There are also methods which outperform the DQN approach to scenario-based policy learning. Hoel et al. [5] proposed a Bayesian RL technique based on an ensemble method of neural networks with randomized prior functions(RPF) in a highway scenario which outperforms a standard deep Q-network agent in a discrete action space. Using this approach the authors [5] also put forth a framework for calculating the uncertainty of the decision made by the agent which could enable the ego-vehicle to make safe driving decisions in a real-world environment.

The combination of using graph neural networks with deep reinforcement learning has produced promising results in the area of scenario-based driving policy learning [8] [7]. Generally in this approach each of the vehicles are considered as nodes of the input graph, the node attributes or features encode information about the position and velocity of the vehicles and the edges of the graph provide any relative information between the two vehicles. This type of representation provides rich information to the ego-vehicle to learn an optimal controller policy [7] [34]. For a three-lane highway

scenario with two exit ramps, Liu et al.[35] developed a graph convolutional network (GCN) combined with a reinforcement learning agent for generating cooperative lane change decisions. The authors used DQN, Double DQN, Dueling DQN and D3QN as the reinforcement learning algorithms in a discrete action space and proposed a modular framework which shows the generality of the policies learned. Similarly, Xiaoqiang et al. [36] propose a GCN combined DRL method using DQN, double DQN and D3QN for learning an optimal driving policy to overtake human-driven vehicles in a highway-simulated environment using a discrete action space. The proposed method shows an improvement in the overtaking process and at the same time reduces the accident rate [36]. Another method for a single ego-vehicle-based graph neural network with DQN known as SGRL was proposed by Yang et al. [37] for scenario-based driving tasks. The SGRL algorithm performs better than a traditional DQN and even a multi-agent training algorithm (MGRL) in the highway ramp driving scenario [37].

Some have also proposed methods which use other deep reinforcement learning algorithms instead of DQN and its variations for scenario-based policy learning. Huegle et al. [38] proposed the Deep Scene-Graphs architecture which combined Deep Scene Sets with a GCN in order to learn the policy in a SUMO simulation environment. The use of Deep Scene Sets in the architecture enabled it to process multiple variable-length sequences of different object types like vehicles, lanes or traffic signs in a highway driving scenario with discrete lane change action space [38]. There have also been approaches which combined actor-critic methods with GNNs to find the optimal driving policy in various simulated traffic environments. Using a combination of a Long-short term memory neural network (LSTM) with Proximal policy optimization (PPO) and updating node embeddings using GraphSAGE convolutions, Ma et al. [34] developed the STGSage model to infer the latent states of other traffic vehicles in a T-intersection scenario while using discrete action space values for the speed of the ego vehicle. Using this framework they improved the performance in the context of navigating T-intersections compared with other state-of-the-art baseline approaches [34]. Another approach put forth by Hart et al. [7] combined the actor-critic method PPO with graph neural networks to develop a highway scenario-based driving policy. In this work, the authors used a continuous action space for the ego-vehicle to show that GNNs are invariant to the number and order of vehicles in a scenario compared to using conventional deep neural networks [7]. These approaches show the promising efficiency of using a combination of graph neural networks and deep reinforcement learning algorithms in learning the optimal control policies for navigating different scenarios that can be encountered in the real world.

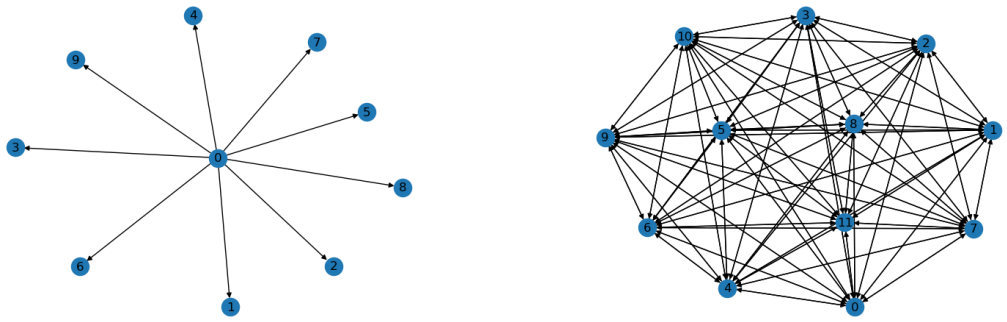
From the above paragraphs on current research done in scenario-based situations in AD, there appears to be a common trend in constraining the DRL agent to a discrete action space in the simulated environments. However, in their review paper [4] Kiran et al. argue that even though using discrete action has advantages like reduction in complexity and helps to accelerate the policy optimization [29], it can lead to jerky or unstable trajectories if the step values between the actions of the ego vehicle take are too large. Further, Sallab et al. [39] in the performance analysis of discrete and continuous action algorithms in the TORCS simulator shows that the latter provides

better performance in both straight and curved parts of the road track. Also, treating the action space as discrete might help in solving the problem easily but it may come at the cost of the feasibility of the solution when applied to real-world problems [40].

One other finding is that graph neural network-based approaches show promising results and they enable reinforcement learning agents to recover highly flexible, generalizable and scalable behaviour policies than those learned through other approaches [41] [42]. Due to connections between vehicles in the graph, GNNs introduce a relational bias to the scenario-based learning problem [7] which can help the ego-vehicle learn directly instead of inferring from collected experiences during training about vehicle-to-vehicle relationships. Also, according to Hart et al. [7], unlike conventional deep neural networks which would require learning all the possible combinations of the order of vehicles in the traffic, GNNs are invariant to both the number and order of vehicles present. Since in a real-world scenario, the number and order of vehicles can change rapidly, GNNs provide a much more robust approach to deal with the stochasticity of traffic situations [7]. Due to the above information from the literature review, a good approach to learning an optimal driving policy in a highway scenario would be to use a GNN-based DRL agent which chooses actions in a continuous space.

3 Methodology

From the literature review, it was seen that using a GNN with DRL model in a continuous action space is a promising approach. This section introduces the general methodology and details about the deep reinforcement learning algorithm and the graph neural network used in this thesis. Each of the vehicles in the highway scenario is treated as a node and their attributes like position and velocity will be encoded in the node attributes of the input graph. Further, relational bias will be explicitly added between the vehicles with the help of graph edges and edge attributes. The two input graph configurations that is used in this thesis are given in Figure 6. In the Ego-centric unidirectional graph 6a, the central node is the ego-vehicle and all the edges are connected between the ego-vehicle node and the surrounding vehicle nodes in a unidirectional manner. But, in the Combined vehicle input graph 6b, additional bidirectional edge connections between surrounding vehicles are also included in addition to the edge connections to the ego-vehicle.



(a) Ego-centric unidirectional. **(b)** Combined vehicle bidirectional.

Figure 6: Input graph configurations.

Using either of the two input graphs, the GNN will be used to compute new node embeddings for the ego-vehicle with respect to other vehicles in its surroundings. This new embeddings enables the DRL algorithm to learn an optimal policy for driving in the simulated highway environment. In the following subsection, the specific details about the MDP problem of this thesis is defined.

3.1 Problem formulation

Using DRL, the task of autonomous driving can be formulated as a Markov decision process (MDP) since the internal state of the driver models of the surrounding vehicles is observable within a certain perception distance of the ego-vehicle [5] [28].

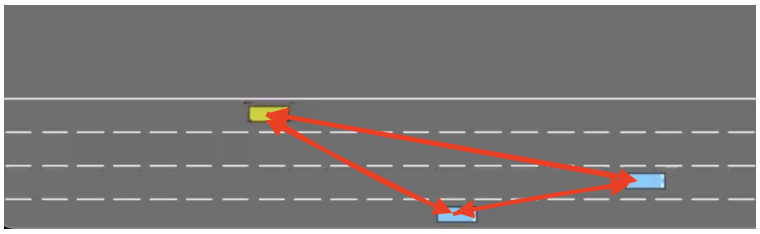


Figure 7: Bidirectional graph between the ego-vehicle and other vehicles in the simulation.

Following the notation from Grigorescue et al. [28], the problem of learning an optimal driving policy in the highway environment as shown in Figure 7, can be formulated as a MDP : $M := (O, S, A, T, R)$ where:

- O represents the set of bidirectional or unidirectional graph observations in which each graph at time t can be defined by $G = (V, E)$, following the Graph network framework mentioned in the background section 2. The choice of which of the two graph structures to use as input depends on the particular problem. The individual graph nodes represent the vehicles on the highway and the edges represent the relative information between the vehicles using edge attributes. Also, the node attributes contain information about the position and velocity of the respective vehicles. In this thesis, the global attribute is not used for constructing the input graph.

- S represents the finite set of states s at time t which defines the state of the ego-vehicle and surrounding vehicles in terms of their position, velocity and heading. For this scenario, these states are directly known by the ego-vehicle for all the vehicles within the perception distance of 200 metres in the simulation.

- A represents the finite set of actions a performed by the ego-vehicle at time t. This continuous action space includes the low-level control commands like throttle and the steering angle of the ego vehicle.

- T represents the state transition model which describes how the ego-vehicle's state evolves over time in response to its actions. This might also involve how the input graph structure evolves considering the movement of the surrounding vehicles in the highway.

- R represents the reward function which encourages the ego-vehicle to learn a safe and efficient driving policy in the highway environment. The main objective of the ego-vehicle is to travel faster, avoid collisions with other vehicles and try

to drive in the rightmost lane. In this thesis, the reward function is defined as:

$$
\begin{aligned}
R(s,a) &= h\left(\frac{v - v_{min}}{v_{max} - v_{min}}\right) + b(r_{collision}) + c(r_{rightlane}) + r_{onroad} \\
r_{collision} &= \begin{cases} 1 & \text{if there is a collision} \\ 0 & \text{if there is no collision} \end{cases} \\
r_{rightlane} &= \frac{\text{Current Lane Index}}{\max(\text{Total Number of Neighboring Lanes} - 1, 1)} \\
r_{\text{onroad}} &= \begin{cases} 1 & \text{if the vehicle is on the road} \\ 0 & \text{if the vehicle is off the road} \end{cases}
\end{aligned}
\tag{7}
$$

where the terms $r_{collision}$, r_{onroad} and $r_{rightlane}$ represent the reward for vehicle collision, travelling on the road and driving in the rightmost lane respectively. The first term in the reward equation represents the high speed reward part and the terms v, v_{min} and v_{max} represents the current, minimum, and maximum speed of the ego vehicle, respectively. The high speed reward is given to the ego-vehicle if it reaches the speed range [20,25] m/s^2. The max speed reached by the other vehicles is 20 m/s^2 and this speed range is chosen such that the ego-vehicle speeds up with other vehicles in the highway simulation. Also, the high speed reward values are normalized in the range [0,1]. h, b and c represent the reward weights for the high speed reward, collision penalty and right lane reward respectively. These weight values are chosen in such a way that the ego-vehicle satisfies the main objective mentioned before. The specific values used are shown in the below Table 1.

Reward weight	**value**
`high speed reward weight(h)`	0.5
`collision penalty weight(b)`	-1
`right lane reward weight(c)`	0.2

Table 1: Reward weight values.

In the next two subsections, the specific DRL algorithm and GNN model used in this thesis are introduced.

3.2 Soft Actor-critic algorithm

Soft actor-critic (SAC) is a deep reinforcement learning algorithm built upon the actor-critic architecture and it is used for training RL agents to perform well in continuous action spaces [43]. SAC extends the traditional Q-learning approach by introducing the concept of "soft" Q-functions [43] by which instead of relying on a single Q-function, the SAC algorithm uses two Q-functions to estimate the cumulative rewards [43]. The

equation for soft Q-function or state-action value function for starting from current state s, taking action a and following policy π can be defined as [44]:

$$Q(s,a) = \mathbb{E}_{s',r\sim p(\cdot|s,a)}\left[r + \gamma \cdot \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q(s',a') - \alpha \cdot \log \pi(a'|s')]\right] \tag{8}$$

where the γ is the discount factor, s' and a' are the next state and action terms, r is the reward, $\pi(a|s)$ denotes the policy and α denotes the entropy temperature which controls the trade-off between expected cumulative reward and encouraging exploration. The term $\alpha \cdot \log \pi(a'|s')$ denotes an important feature of SAC known as entropy regularization, which encourages exploration by discouraging overly deterministic policies [44]. If the entropy is increased the agent can explore more of the environment and it can speed up the agent's learning later during training [44]. In order to compute the target values for the soft Bellman backup, target Q-networks and target policy networks are used and these help in stabilizing the training [43]. The Soft state-value function, $V(s)$ for state s taking action a with Q value function $Q(s,a)$ can be defined as [44]:

$$V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q(s,a) - \alpha \cdot \log \pi(a|s)\right] \tag{9}$$

The combination of the actor-critic methods, entropy regularization and soft Q-learning allows SAC to address the key challenges in RL, such as sample efficiency, exploration and stability in learning [43] making it one of the reliable approaches when tackling problems with continuous action space. Since in this thesis, the action space is continuous, SAC is implemented as the main deep reinforcement learning algorithm to control the ego vehicle to learn the best driving policy in the simulated highway environment. The next subsection introduces the graph neural network model that will be combined with the SAC algorithm.

3.3 Graph Attention Neural Network

The self-attention mechanism, initially inspired by how human retinas allocate focus to relevant regions of the optic array [45] is a fundamental component in various machine learning models, especially in areas such as computer vision and natural language processing [8]. The attention mechanism in graph neural networks was first proposed by Veličković et al. [26] as the Graph attention network (GAT) which works on both inductive and transductive problems on graph-structured data [8]. It leverages the attention mechanism to enable nodes to focus on their neighbours with varying degrees of importance when performing computations [26].

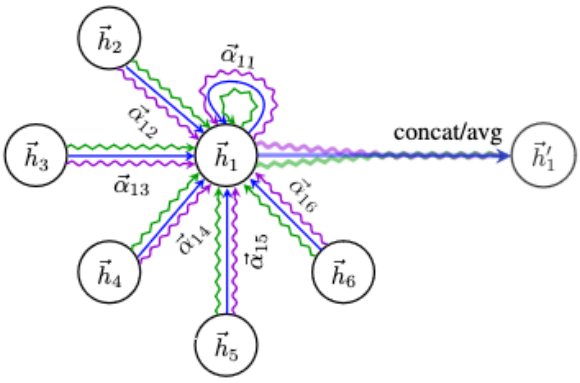


Figure 8: Multihead attention with K value of 3 heads from [26] for GAT.

In the GAT architecture, each node computes an attention score with respect to its neighbours during the training phase and they represent how important each neighbour is to the current node for the given task [26]. All of these attention scores are then used to compute the weighted sum of the neighbour node features which means that each of the nodes aggregates information from their neighbours, giving more weight to the relevant neighbours based on the learned attention scores [26]. In addition to that this aggregated information is then used to update the node's feature representation [26]. For each edge (i, j) in a given graph, the attention coefficient A_{ij}, the normalized attention coefficient for the aggregation α_{ij} and the updated node feature representation $\mathbf{h}'_i$ can be defined as [26]:

$$\begin{aligned} A_{ij} &= \text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \,||\, \mathbf{W}\mathbf{h}_j]) \\ \alpha_{ij} &= \frac{\exp(A_{ij})}{\sum_{k\in N_i}\exp(A_{ik})} \\ \mathbf{h}'_i &= \sum_{j\in N_i} \alpha_{ij}\mathbf{W}\mathbf{h}_j \end{aligned} \tag{10}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the feature representations of nodes i and j respectively, $\mathbf{W}$ is the learnable weight matrix, $\mathbf{a}$ is a learnable attention vector, N_i represents the neighbours of node i and LeakyReLU is a leaky rectified linear unit activation function [26]. The number of attention heads, K [26] is a hyperparameter that can be adjusted in the GAT model. Each of the attention heads in GAT independently computes the attention coefficients and aggregates information from the neighbouring nodes to produce the final node representations [26]. By using multiple attention heads ($K > 1$) as shown in Figure 8, the GAT model can capture different patterns and relationships from the graph data [26]. But one drawback with the GAT model is that the linear layers are applied right after each other which makes the ranking of the attended nodes unconditioned on the query node [46]. This form of static attention mechanism [46]severely reduces the expressiveness of the GAT and to rectify this Brody et al. [46] proposed the GATv2 network with a dynamic attention mechanism through which

every node in the graph can attend to any other node by applying the attention scoring layer after the activation function. This change in the order of computing the attention coefficient can be defined in the case of GATv2 as [46]:

$$A_{ij} = \mathbf{a}^T \text{LeakyReLU}(\mathbf{W} \cdot [h_i \,||\, h_j]) \tag{11}$$

With this modification, GATv2 outperforms GAT in 12 Open Graph Benchmark (OGB) and several other benchmark tests [46]. In this project, the GATv2 network is used to learn the node embeddings of the ego vehicle with respect to its neighbouring vehicles in the simulated highway driving scenario.

3.4 Implementation

For scenario-based simulations for autonomous driving, there are a variety of simulation environments that can be used like CARLA [47] and SUMO [48]. For this thesis, HighwayEnv was chosen due to the simplicity of the environment compared to others. HighwayEnv is a collection of simulation environments for autonomous driving which can be used to train reinforcement learning agents to learn optimal driving policies in scenarios like lane-changing, parking, intersections and other real-world traffic situations [49]. For this project, $highway-v0$ is used to simulate the environment of a highway where the ego-vehicle tries to learn an optimal driving policy while avoiding collisions with the boundary and the other vehicles in its surroundings. An instance of a yellow-coloured ego-vehicle in a four-lane highway environment is shown in the below Figure 9.

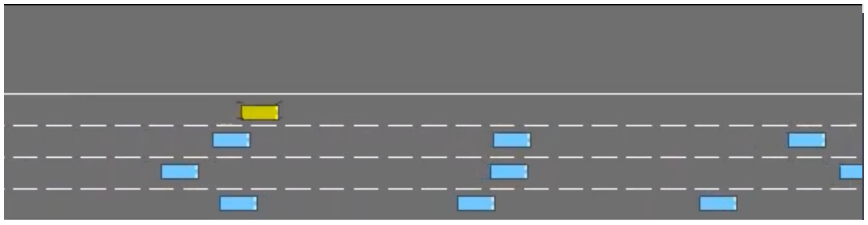


Figure 9: HighwayEnv

The main goal of the ego-vehicle is to travel quickly on the road while keeping to a certain lane at a time and avoiding accidents with other vehicles. The observation output from the simulation is defined to be a $KinematicObservation$ array and it contains information about features like the position and velocity of each of the vehicles up to a perception distance from the ego-vehicle's vicinity [50]. The action space is defined to be $ContinuousAction$ type which allows the agent to directly set low-level controls like throttle a and steering angle δ of the ego-vehicle. The low-level control values of the throttle a and the steering angle δ are in the continuous ranges of [-3.5,3.5] m/s^2 and [-10,10] radians respectively. The ego-vehicle chooses values from within these two ranges to learn how to drive optimally in the highway environment. The simulation is reset if the ego-vehicle crashes with road boundaries or with other vehicles or the total time specified for the simulation is over.

3.4.1 Input graph creation

For creating the input graph of traffic in the highway environment, the output from the $KinematicObservation$ array and the graph $Data$ object from the Pytorch geometric library [51] is used. Also, in order to make a distinction between the ego-vehicle and other vehicles, the first row of the output array and the node 0 in the input graph is fixed to contain information about the ego-vehicle. The creation process is similar for both of the input graph configurations and the process here is explained for the combined vehicle bidirectional graph shown in Figure 10.

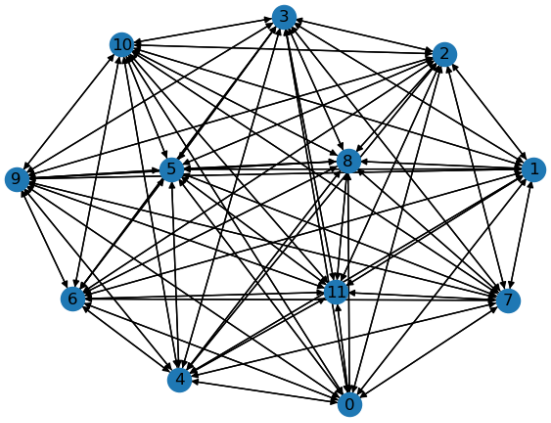


Figure 10: Bidirectional graph input for the GATv2 layer with node 0 representing the ego-vehicle.

Node Attribute	Definition
$presence$	Disambiguate agents at 0 offset from non-existent agents.
x	World offset of ego vehicle or offset to ego vehicle on the x-axis.
y	World offset of ego vehicle or offset to ego vehicle on the y-axis.
vx	The velocity component in the x-direction.
vy	The velocity component in the y-direction.
$heading$	Heading of the vehicle in radians.
cos_h	Trigonometric heading of vehicle.
sin_h	Trigonometric heading of vehicle
cos_d	Trigonometric direction to vehicle's destination.
sin_d	Trigonometric direction to vehicle's destination.
$long_{off}$	Longitudinal offset to closest lane.
lat_{off}	Lateral offset to closest lane.
ang_{off}	Angular offset to closest lane.

Table 2: Node Attributes from [50].

Edge Attribute	Definition
$delta\_x$	The change in the x-coordinate between two vehicles.
$delta\_y$	The change in the y-coordinate between two vehicles.
$delta\_vx$	The change in velocity component in the x-direction between two vehicles.
$delta\_vy$	The change in velocity component in the y-direction between two vehicles.
$delta\_theta$	The change in heading angle between two vehicles.

Table 3: Edge Attributes.

The $KinematicObservation$ array contains 11 features that can be used to create node attributes and edge attributes for each of the vehicles in the simulation within a perception distance of 200 metres from the ego-vehicle. By constructing these attributes and specifying the edge connections between each of the vehicles, a graph $Data$ object can be constructed for each instant of the simulation. Further using the $presence$ attribute, the graph creation process can handle the change in the number of nodes in the input graph structure during the entire length of the simulation. Figure 10 represents a visualized graph $Data$ object of the traffic in the simulation. Each of the nodes of the graph is a vehicle that is visible to the ego vehicle node 0 within the perception distance of the simulation. Each of the nodes has 11 node attributes and each of the edges has 5 edge attributes as shown in Table 2 and Table 3 respectively. This allows to encode information in the graph which can be used for getting new embeddings for the ego-vehicle after passing it through the GATv2 network. The specific details of the training process and the neural network architectures involved are discussed in the remaining parts of this section.

3.4.2 Neural Network architecture

In this thesis, the code used for implementing the Soft-actor critic algorithm is based on the CleanRL [52] implementation of SAC for continuous action space. For the GATv2 network the PyTorch geometric library (PyG) [51] is used and this specific GNN architecture was chosen after observing the model's performance in the test phase. Also, the designs of the Linear layer dimensions in the actor and critic architectures are kept similar to the CleanRL architecture which shows good benchmarked results in continuous action-space problems [52].

Layer name	Operation	Dimension	Attention Heads(K)
$conv1$	GATv2Conv	11x55	5
$conv2$	GATv2Conv	55x110	1
$dense_{fc1}$	Linear(Sequential)	110x1	-
$dense_{fc2}$	Linear(Sequential)	1x275	-

Table 4: GATv2 network architecture.

The general architecture for the GATv2 network or the SoftGATv2Network is given in the above Table 4. The network contains two GATv2Conv layers, $conv1$ and $conv2$ with attention head values of 5 and 1 respectively. The individual graphs are passed through each of the layers and then the resulting new embeddings for the ego node are passed through a sequential dense neural network layer to get the new ego embeddings. In order to introduce non-linearity to both of the linear layers $dense_{fc1}$ and $dense_{fc2}$ in the dense sequential layer, $ReLU$ activation function is applied. This GNN architecture is used as the initial neural network layer for both the actor and the two critic networks to extract relational information between the different nodes to the ego node for learning the optimal policy. Since the GNN layer is used to extract node embeddings related to information like positions and velocities of vehicles, quantities that can be positive or negative, a tanh activation function is applied to the final ego-vehicle embeddings.

Layer name	**Operation**	**Dimension**	**Attention Heads(K)**
$conv1$	GATv2Conv	11x55	5
$conv2$	GATv2Conv	55x110	1
$dense_{fc1}$	Linear(Sequential)	110x1	-
$dense_{fc2}$	Linear(Sequential)	1x275	-
$actor_{fc1}$	Linear	275x256	-
$actor_{fc2}$	Linear	256x256	-
fc_{mean}	Linear	256x2	-
fc_{logstd}	Linear	256x2	-

Table 5: SoftGATv2 Actor network architecture.

The policy network or the SoftGATv2 actor network is given in the above Figure 5. The first layer is the GNN network which takes the graph data from the environment as input. This is followed by a linear layer of dimensions 275x256, then another layer with 256x256 and finally a layer with 256x2 dimensions, since the ego-vehicle has to perform two actions. For both of the linear layers $actor_{fc1}$ and $actor_{fc2}$ in the actor architecture, $ReLU$ activation functions are applied to introduce non-linearity to the network. The architecture outputs the mean and logarithm of the standard deviation of the action distribution through fc_{mean} and fc_{logstd} respectively.

Layer name	Operation	Dimension	Attention Heads(K)
$conv1$	GATv2Conv	11x55	5
$conv2$	GATv2Conv	55x110	1
$dense_{fc1}$	Linear(Sequential)	110x1	-
$dense_{fc2}$	Linear(Sequential)	1x275	-
-	Concatenation	1x277	-
$critic_{fc1}$	Linear	277x256	-
$critic_{fc2}$	Linear	256x256	-
$critic_{fc3}$	Linear	256x1	-

Table 6: SoftGATv2 Critic network architecture.

The architecture of the critic network is given in the above Figure 6. Similar to the actor, the first layer is a GNN network, followed by a linear layer with dimensions 277x256, then 256x256 and finally a linear neural network layer with dimensions 256x1. The input dimension of $critic_{fc1}$ is 277 to account for the concatenation of the action distribution containing the two actions. For both the linear layers $critic_{fc1}$ and $critic_{fc2}$ in the critic architecture, $ReLU$ activation functions are applied to introduce non-linearity to the network. Since generally a SAC algorithm contains two critic networks, the same architecture is used for the other critic network. These three network architectures in Tables 4, 5 and 6 represent the main building blocks of the graph neural network combined deep reinforcement learning algorithm used in this thesis.

3.4.3 Training Process of SoftGATv2 actor-critic algorithm

The training process of the SoftGATv2 actor network is shown in Figure 11. Initially, each of the observation data the ego-vehicle receives from the environment is converted into an input graph and given as input to the actor and the two critic networks. The input graph is generated dynamically at each instant of the simulation and the networks are built to handle a varying size of graph data depending on the neighbouring vehicles. The actor contains a GNN layer which extracts the new ego node embeddings from the graph input data and then it is passed through dense neural network (NN) layers for getting the action distribution for the ego-vehicle to execute in the highway environment.

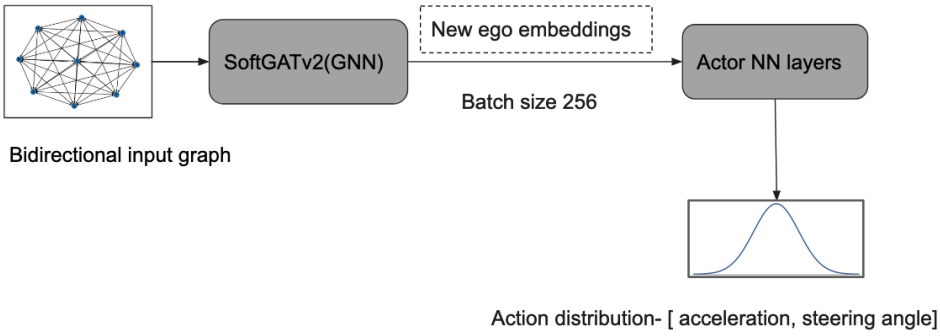


Figure 11: SoftGATv2 Actor network.

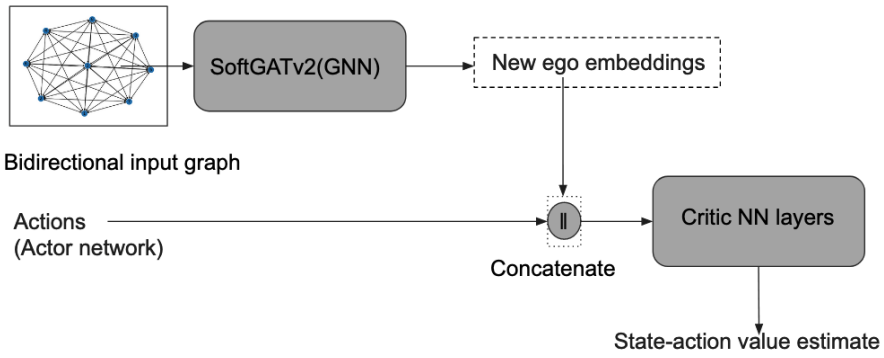


Figure 12: SoftGATv2 Critic network.

Similarly, the new ego-vehicle embeddings are extracted from the first GNN layer of the critic network and then concatenated with the action value estimates from the actor network as shown in Figure 12. This information is then passed through dense neural network (NN) layers to find the state-action value estimates for both of the two critic networks and the actor network is updated for the subsequent training iteration. These mentioned steps are involved in the training process of this algorithm.

4 Experiments

In this section using the trained SoftGATv2 actor policy network the following research questions are investigated:

(i) Is there a change in performance of the model depending on the input graph structure?

(ii) Can the decision-making process of the ego-vehicle be analyzed using the attention weights of the GNN component of the policy network?

In order to find if there is a change in performance depending on the input graph's structure, a comparison of two different input graph configurations is investigated. One of the configurations will be an ego-vehicle centric unidirectional graph and the other will be a combined vehicle bidirectional graph as shown in Figure 6. Further, to better understand the decision-making process of the proposed GNN-based algorithm, attention weights of the policy network are analyzed to look at how the graph neural network layers assign importance to various vehicles within the perception distance of the ego-vehicle.

4.1 Performance comparison of different graph input structures

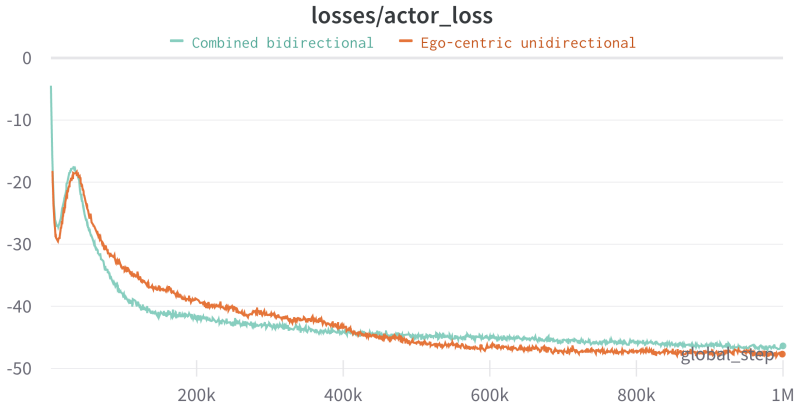
For investigating if the performance of the policy network changes with the configuration of the input graph data, the policy network is trained with two different structures of graph data - (i) Ego-centric unidirectional and (ii) Combined Vehicle bidirectional as introduced in Figure 6. The hyperparameter values used for training the Soft actor-critic are given in Table 7 and all of the values are kept constant for both of the input graphs. These hyperparameters were chosen by inferring the performance of the trained policy in a test simulation and adjusting the values through trial and error. For both input graph configurations, the model is trained for a total of 1 million time steps and each of the simulations has 25 vehicles driving in the forward direction on the highway. The performance is compared during both the training and testing phases between the two models trained on each of the input graphs. Also, a point to note is that in this experiment only a single starting point for the ego-vehicle is used with the random seed value of 42. So from this experiment, which input graph is better can not be determined due to the sparsity of the results.

Hyper-parameter	**value**
`Learning rates`	3e-4
`Discount factor(`γ`)`	0.99
`Buffer size`	1e6
`Target smoothing coefficient(`τ`)`	0.99
`Batch size`	256
`Simulation frequency`	15
`Total-timesteps`	1e6

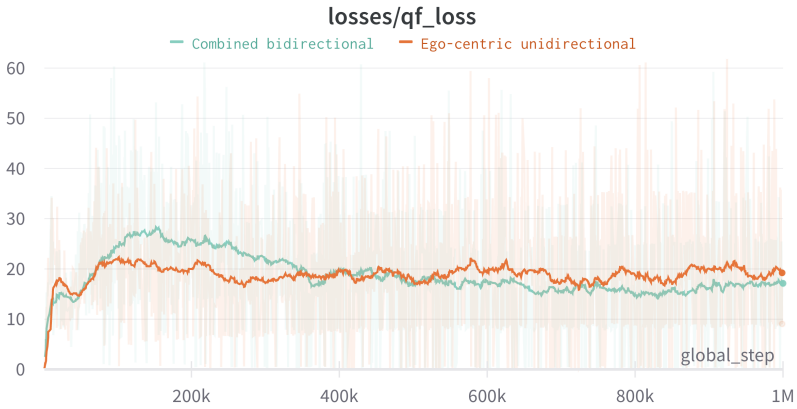
Table 7: Hyper-parameter values.

4.1.1 Training Results with both input graph configurations

The network loss plots for both the actor networks and the critic networks are given in Figure 13. In the actor network loss plot 13a, initially the loss for the bidirectional graph seems to decrease faster compared to the ego-centric input graph. But at the end of a million time-steps, the actor network losses for the bidirectional and ego-centric input graphs reach a similar convergence rate, specifically the loss values -46.37937 and -47.39774 respectively. Now in the case of the critic network loss plot 13b, initially the bidirectional input graph has a higher loss value compared to the ego-centric graph input. But after 400k global steps, the bidirectional graph loss starts to decrease to reach the loss value of 17.00853, while the ego-centric input graph loss reaches a higher loss value of 26.78267.

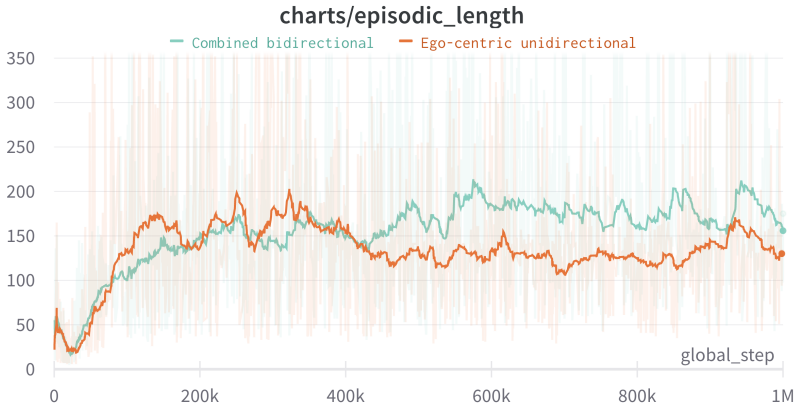


(a) SoftGATv2 Actor network loss vs Global step.

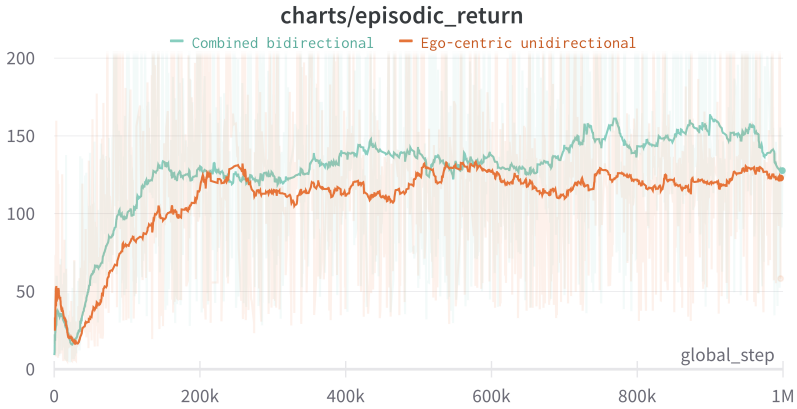


(b) SoftGATv2 Critic network loss vs Global step.

Figure 13: Network training loss plots.



(a) Episodic length vs Global step.



(b) Episodic reward return vs Global step.

Figure 14: Training plots for Episodic length and return.

In addition to that, the episodic return and episodic length for the entire training phase is shown in Figure 14. For the episodic length plot 14a, initially the ego-centric shows improvement compared to the bidirectional graph input, but after 400k global steps, the latter seems to improve till the end of the training phase. In the case of the episodic return plot 14b, after 600k global steps the bidirectional model accumulates more episodic rewards compared to the ego-centric model. So, both Figure 13 and Figure 14 indicate that using the combined vehicle bidirectional graph as input improves the model's performance compared to the ego-centric graph input for this particular highway environment configuration.

4.1.2 Test reward plots for both input graphs

To check if there is a change in performance between the two SoftGATv2 actor-critic models trained on different input graph configurations, both of the trained actor network policies are tested in the same highway environment with a random seed value of 42. The total reward along with the high speed reward, right lane reward and collision

Weighted test reward	Ego-centric Unidirectional	Combined Vehicle Bidirectional
Total reward	439.965	526.76386
High speed reward	230.2405	34.96523
Right lane reward	11.6	35.53333
Collision penalty	-1	0

Table 8: Test episode rewards.

penalty are tracked for both of the models as shown in Figure 15 and Figure 16. The final values of the rewards are given in Table 8.

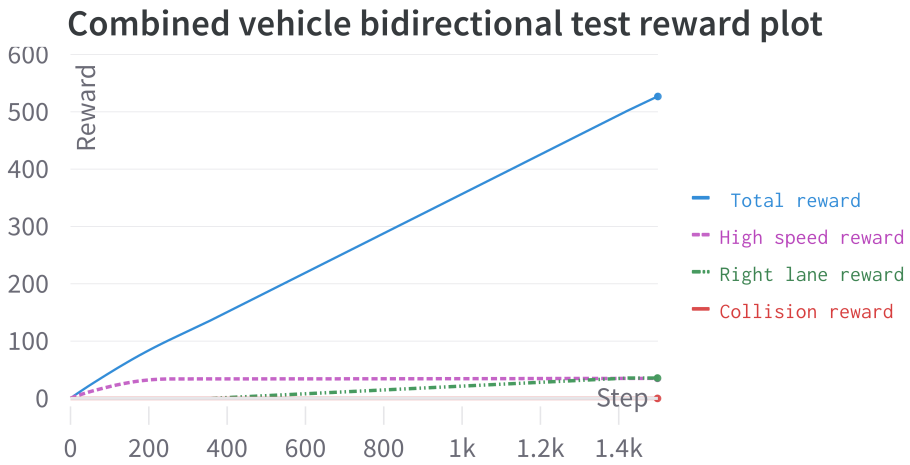


Figure 15: Episodic test reward plot for combined vehicle bidirectional input graph.

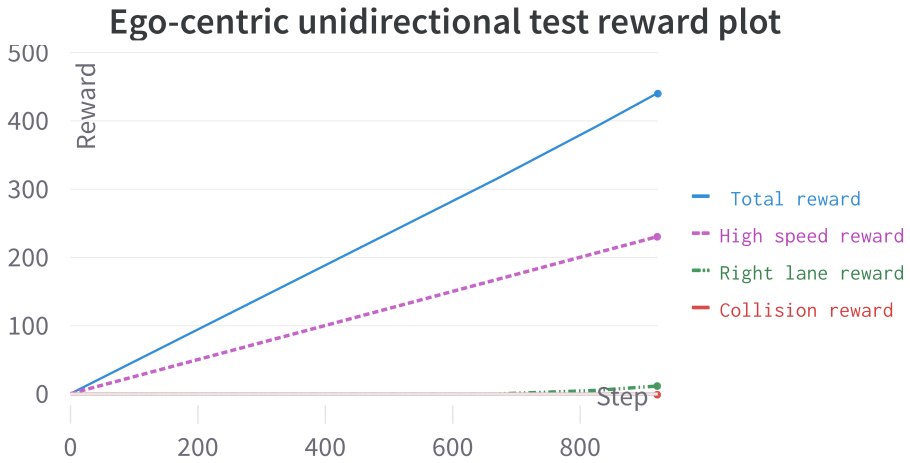
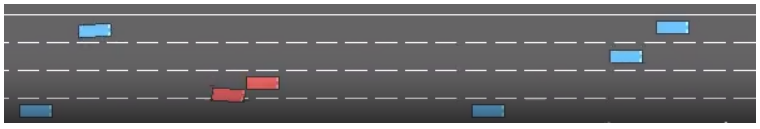


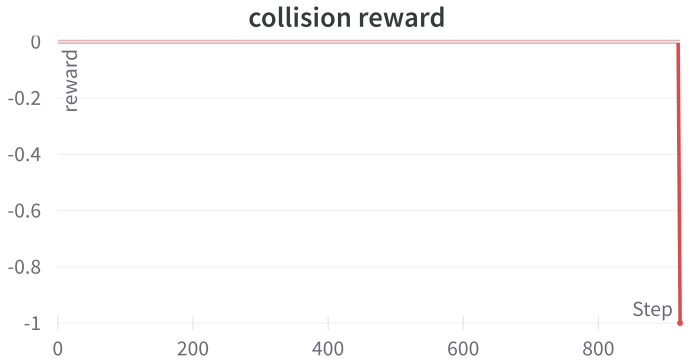
Figure 16: Episodic test reward plot for Ego-centric unidirectional graph.

In the test simulation for the Combined vehicle bidirectional model, the ego-vehicle

does not move much faster, but it keeps a safe distance from the other vehicles in its surroundings. By keeping up with its neighbours from a safe distance, the ego-vehicle avoids any collision and accumulates a total evaluation reward of 526.76386 with 1499 steps. Now in the test simulation for the Ego-centric unidirectional model, the ego-vehicle priorities moving faster through the highway and this can be seen by the high speed reward plot in Figure 16 and the total high speed reward value in Table 8. Further, the ego-vehicle tries to keep to a single lane while moving through the multi-laned highway most of the time, but after 800 steps it tries to move towards the right lane as shown in Figure 17a and a rear-end collision occurs.



(a) Rear-end collision in HighwayEnv.



(b) Collision penalty vs Episodic step.

Figure 17: Collision scenario for ego-centric unidirectional input graph.

The negative collision reward at that particular instant is given in Figure 17b. This collision terminates the test simulation and it affects the total reward accumulated by the ego-vehicle, reducing it to 439.965. Further, compared to the bidirectional model, this decreases the total steps in the test simulation for the Ego-centric unidirectional model to 922.

So from the results of the training plots, the test reward plots and the test simulations, it shows that the performance of the policy network depends on the input graph's structure for this particular highway environment setting. But since the results are for a single starting position of the ego-vehicle, which of the two models is better can not be ascertained.

4.2 Policy network attention weight visualization

The second research question this thesis aims to investigate is if there is a way to analyze the decision-making process of the actor policy network in controlling the ego-vehicle. Since one of the components of the actor network is a graph neural network, analyzing the connections between the ego-vehicle node and its surrounding nodes may provide some insight. One advantage of using a graph attention neural network is that the attention weights of the policy network for the ego-vehicle node can be visualized. In order to visualize it, in this experiment, the output data from the $KinematicObservation$ array is used from an initial instant of the test simulation as shown in Figure 18. Each of the vehicles in the output are visualized as nodes for both of the two layers of the GATv2 layer of the trained policy network. The nodes are scattered randomly in the visualization and the distances between the nodes do not correspond to the distance between the individual vehicles in the test simulation. The node number of the vehicles is given according to their row entry number in the $KinematicObservation$ array output from the simulation. The higher the node number, the farther the vehicle's position is from the ego-vehicle in the simulation. This means that node 2 is much closer to the ego-vehicle node 0 than the vehicle with node number 7. In the visualization, the attention weight values are multiplied to the thickness of each of the edge links to the ego vehicle node from the different nodes. This means that only vehicles with non-zero attention values with respect to the ego node will have edge connections in the visualization. The edge connection thickness to the other vehicle nodes from the ego-vehicle and the colour of individual nodes represent how much important the particular vehicle is to the ego-vehicle for decision-making. More the thickness and higher the colour of the node in the attention weight bar in the figure, the more prominent the particular node is involved in the decision-making of the ego-vehicle. The visualized attention weights with respect to the ego-vehicle is given below in Figure 19.

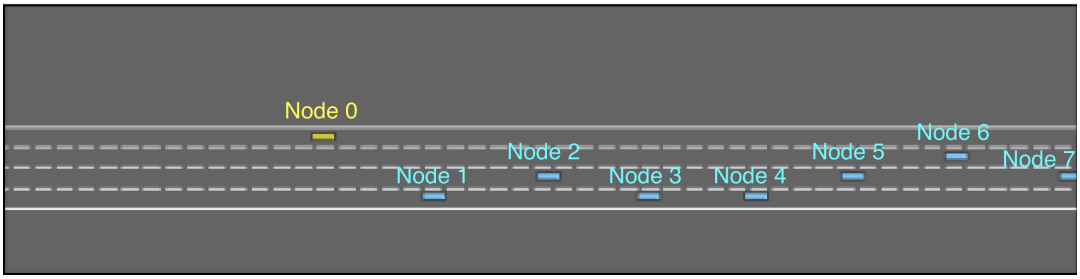
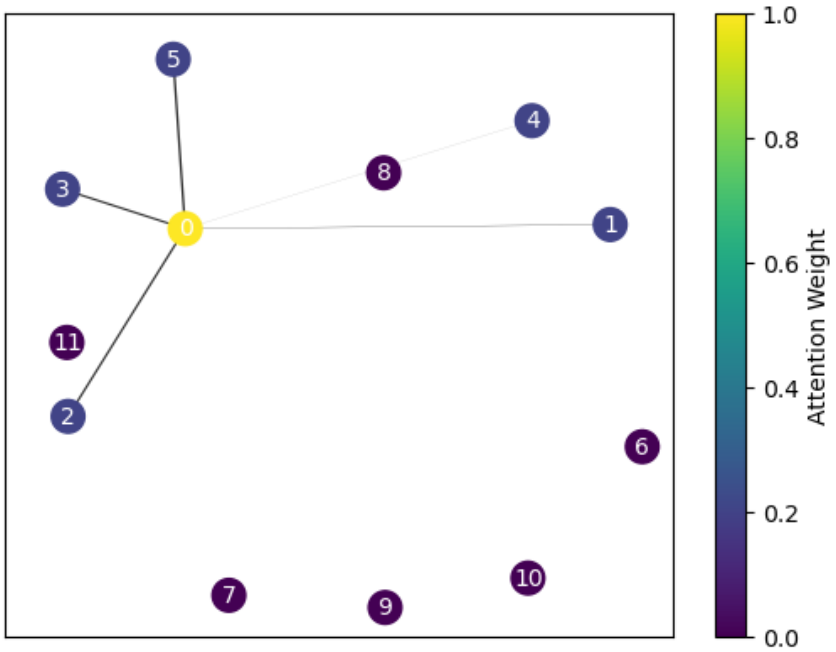
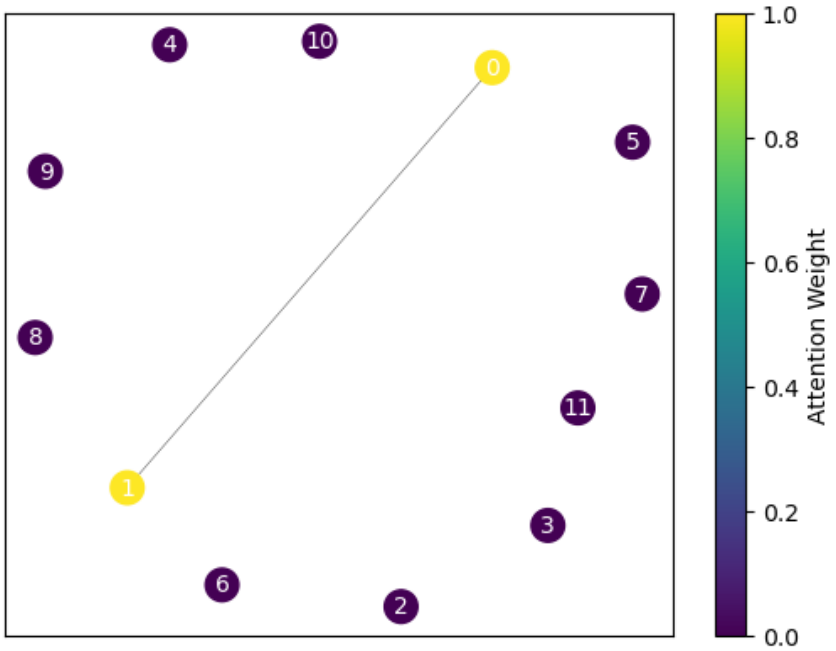


Figure 18: Initial instant of simulation



(a) Attention weight connections in SoftGATv2 layer 1 for ego-vehicle node 0.



(b) Attention weight connections in SoftGATv2 layer 2 for ego-vehicle node 0.

Figure 19: Attention weights for an initial instant of the test simulation .

In Figure 19, the attention weights for the ego-vehicle node to the other nodes within the perception distance are shown for an initial instant in the simulation. The

test simulation contains a total of 25 vehicles and in the initial instant, there are 10 vehicles within the 200 metre perception distance of the ego-vehicle. Each of the 10 vehicles with their node number is shown in Figure 19, with the yellow-coloured node 0 being the ego-vehicle. Also, Figure 19a represents the attention weights from the first GATv2 layer of the SoftGATv2 policy network and the ego vehicle appears to give more attention to vehicles near it, particularly vehicles 1, 2, 3, 4 and 5. These five nodes are represented with a medium dark shade blue colour in Figure 19a, while the other nodes, 6,7,8,9,10 and 11 are given colour value corresponding to zero on the attention weight colour scale. Also, according to Figure 19a, edge connection thickness to nodes 2, 3, 5 and 1 from the ego-vehicle node 0 is more compared to the other vehicle nodes. Further, in Figure 19b, the second layer weights are visualized and the ego-vehicle gives the most attention to vehicle 1 represented as a yellow node along with the ego-vehicle node. In the second layer, the ego-vehicle node seems to focus on just the closest vehicle to it, particularly vehicle node 1 and there is only one edge connection to the ego-vehicle node 0. Both of the images show that the ego-vehicle focuses on the vehicles near it during the start of the simulation. So, this form of visualisation does provide some transparency to the otherwise black-box-like decision-making process of the trained policy network.

5 Discussion

5.1 Effectiveness of using a GNN for highway autopilot

Based on the results presented in section 4, it can be seen that a GNN-based DRL low-level controller can be used in a simulated highway environment, even though it crashes in one of the experiments. The node embeddings from the SoftGATv2 layers provide rich information about the complex interactions and the spatial relationships between the ego-vehicle and its neighbours. These embeddings provide significant insights into the dynamics of its neighbouring vehicles to the ego-vehicle in the simulation. Further, it was shown that the performance of the model depends on the input graph's structure and configuration, even if individual data related to quantities like position and velocity are the same. However, since the current experiments only look at one particular configuration of the highway environment, which input graph configuration is better can not be ascertained. But, in theory, a bidirectional graph should provide more information to the agent compared to the ego-centric input graph. Having a bidirectional graph with all the other vehicles in the vicinity of the ego-vehicle will provide additional relationships in the input graph structure. In a real-world scenario, since the behaviour of one vehicle might influence the behaviour of others, having a bidirectional graph input may be useful for the ego-vehicle to understand the traffic flow behaviour. This will enable the ego-vehicle to learn a more nuanced and realistic representation of the traffic scenario which is crucial for avoiding accidents which cause RTIs.

Using the attention weights of the trained policy network it is possible to ascertain some parts of the decision-making of the ego-vehicle. This form of visualization may be used to understand how much importance the ego-vehicle gives to its neighbouring vehicles while driving in a scenario-based environment. In the real world, information collected from various sensors such as cameras, LiDAR and radar needs to be fused effectively for autonomous driving. Since these data can be encoded in the nodes and edges of an input graph, GNNs can provide a way to handle these multi-modal data and fuse information from different sensors to make more informed decisions. Also, since GNNs can integrate information from multiple sensor sources, it may provide a redundancy in perception of the highway autopilot which can help in improving the overall safety of the vehicle on the road. Further, the proposed method generates the input graph dynamically depending on neighbouring vehicles at each instant of the simulation and can handle graphs of varying sizes. This allows the trained model to adapt to different traffic densities effectively, which may be the situation that an autonomous vehicle encounters in real-world highway traffic. So using a GNN-based policy offers significant advantages in a highway autopilot system which can be a possible solution to avoid RTIs in a multilaned highway.

5.2 Limitations and Future work

One major limitation of the current work is that the results for the performance comparison of different input graph structures are only for a single starting point of the

ego-vehicle. Due to the sparsity of the results, the optimal graph configuration can not be decided with the results. In order to know which graph is better, the experiments should be done for different random seed values to check the robustness of the models in different starting positions of the ego-vehicle. Another limitation of the current work is the high computation time required to train the agent for both of the input graph configurations. In both cases, the model takes around five days to train for 1 million time steps with the mentioned hyperparameters. One reason could be related to the memory size and complexity of the input graphs used for training the model at each instant of the simulation. Compared to a model which takes tabular array data of position and velocities of the vehicle as input, the input graphs are more complex in terms of data structure and size. Further, the GNN model is computationally complex and it affects the time taken to find an optimal policy in a scenario-based problem. A possible solution to decrease the computation time could be to train the model with more than one GPU core and use effective parallelism for the transfer of information between each of the GPUs. Also, the reward function used in this thesis is only suitable for the highway simulation environment and it may fail to capture the complexity of the real-world traffic. Proper reward function modelling is needed for the efficient deployment of GNN-based DRL controllers in an on-road traffic scenario. These pitfalls may undermine the possible advantages of this method and its implementation in a real-world traffic scenario.

In this thesis, Soft actor-critic and GATv2 networks were used as the main components for learning the policy in the highway scenario. One future approach could be to try other reinforcement learning algorithms like Truncated Quantile Critics (TQC) [53] or a different GNN framework like GraphSAGE [24] in the same highway environment and compare the performance of the learned policy. Also, a possible next step would be to learn a driving policy with the current model architecture in traffic scenarios like road intersections where the use of GNN-based controllers can be useful to navigate the high density of surrounding vehicles. Further, in this thesis, the performance comparison of only two different input graph configurations was investigated. Experimenting with different vehicle starting points and highway environment configurations to know which of the two input graph configurations is better, can be a future direction for this thesis. In addition to that, comparing the model performance with other possible graph structures to find the most optimal configuration will be a relevant problem to solve in order to build a better highway autopilot system.

6 Conclusion

Autonomous driving-based approaches provide one possible solution to reduce RTIs caused due to human error in decision-making. The use of machine learning based approaches to develop better decision-making algorithms for autonomous driving has made significant strides since the last decade. In this thesis, the focus was to develop a highway autopilot using a hybrid approach with a graph neural network and deep reinforcement algorithm in a continuous action space. With the combination of graph attention neural network and soft actor-critic algorithm, the SoftGATv2 actor-critic algorithm was implemented to learn a driving policy in a multi-lane highway environment with traffic. Also, the research goal of this thesis was to know how can GNN be used in a highway autopilot. After reviewing the existing research and considering the findings of this thesis, a viable approach involves utilizing road-traffic information as input to the graph neural network. This network can then generate a new ego-vehicle node embedding relative to other vehicles, serving as training input for the policy network. Further, the dependence of the model's performance on the input graph's configuration is shown using two different graph input configurations. In addition to that, both the unidirectional and bidirectional input graph creation functions are dynamic with respect to the number of vehicles observed and can handle varying numbers of vehicles in the neighbourhood of ego-vehicle. The attention weights of the trained policy network can be used to visualize the importance the ego-vehicle gives to its neighbouring vehicles. This may help in understanding the driving decisions of the ego-vehicle in the highway environment which may not be possible to visualize in approaches without a GNN network. In conclusion, this thesis shows the usefulness of a GNN-based DRL low-level controller in a highway scenario-based learning problem.

References

[1] Heather E. Rosen et al. "Global road safety 2010–18: An analysis of Global Status Reports". In: *Injury* (July 20, 2022). ISSN: 0020-1383. DOI: 10.1016/j.injury.2022.07.030. URL: https://www.sciencedirect.com/science/article/pii/S0020138322005046 (visited on 06/12/2023).

[2] *Automated Vehicles for Safety | NHTSA*. URL: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety (visited on 10/15/2023).

[3] Qi Liu et al. "Decision-Making Technology for Autonomous Vehicles: Learning-Based Methods, Applications and Future Outlook". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). Sept. 2021, pp. 30–37. DOI: 10.1109/ITSC48978.2021.9564580. URL: https://ieeexplore.ieee.org/document/9564580 (visited on 10/30/2023).

[4] B Ravi Kiran et al. "Deep Reinforcement Learning for Autonomous Driving: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926. DOI: 10.1109/TITS.2021.3054625.

[5] Carl-Johan Hoel, Krister Wolff, and Leo Laine. "Tactical Decision-Making in Autonomous Driving by Reinforcement Learning with Uncertainty Estimation". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1563–1569. DOI: 10.1109/IV47402.2020.9304614.

[6] Kai Arulkumaran et al. "A Brief Survey of Deep Reinforcement Learning". In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), pp. 26–38. ISSN: 1053-5888. DOI: 10.1109/MSP.2017.2743240. arXiv: 1708.05866[cs,stat]. URL: http://arxiv.org/abs/1708.05866 (visited on 06/15/2023).

[7] Patrick Hart and Alois Knoll. "Graph Neural Networks and Reinforcement Learning for Behavior Generation in Semantic Environments". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1589–1594. DOI: 10.1109/IV47402.2020.9304738.

[8] Nurul A. Asif et al. "Graph Neural Network: A Comprehensive Review on Non-Euclidean Space". In: *IEEE Access* 9 (2021), pp. 60588–60606. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3071274. URL: https://ieeexplore.ieee.org/document/9395439/ (visited on 06/14/2023).

[9] Cynthia Rudin and Joanna Radin. "Why Are We Using Black Box Models in AI When We Don't Need To? A Lesson From an Explainable AI Competition". In: *Harvard Data Science Review* 1.2 (Nov. 1, 2019). ISSN: 2644-2353, 2688-8513. DOI: 10.1162/99608f92.5a8a3a3d. URL: https://hdsr.mitpress.mit.edu/pub/f9kuryi8/release/8 (visited on 09/23/2023).

[10] Nestor Maslej et al. "The AI index 2023 annual report". In: *AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Stanford, CA* (2023).

[11] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015). Publisher: American Association for the Advancement of Science, pp. 255–260. ISSN: 0036-8075.

[12] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[13] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[14] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[15] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (Dec. 7, 2018), pp. 1140–1144. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aar6404. URL: https://www.science.org/doi/10.1126/science.aar6404 (visited on 06/20/2023).

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, second edition: An Introduction*. Google-Books-ID: uWV0DwAAQBAJ. MIT Press, Nov. 13, 2018. 549 pp. ISBN: 978-0-262-35270-3.

[17] Hanna Kurniawati. "Partially Observable Markov Decision Processes and Robotics". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (May 3, 2022), pp. 253–277. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-042920-092451. URL: https://www.annualreviews.org/doi/10.1146/annurev-control-042920-092451 (visited on 09/01/2023).

[18] Xuanchen Xiang and Simon Foo. "Recent Advances in Deep Reinforcement Learning Applications for Solving Partially Observable Markov Decision Processes (POMDP) Problems: Part 1—Fundamentals and Applications in Games, Robotics and Natural Language Processing". In: *Machine Learning and Knowledge Extraction* 3.3 (July 15, 2021), pp. 554–581. ISSN: 2504-4990. DOI: 10.3390/make3030029. URL: https://www.mdpi.com/2504-4990/3/3/29 (visited on 09/01/2023).

[19] Vijay R. Konda and John N. Tsitsiklis. "OnActor-Critic Algorithms". In: *SIAM Journal on Control and Optimization* 42.4 (Jan. 2003), pp. 1143–1166. ISSN: 0363-0129, 1095-7138. DOI: 10.1137/S0363012901385691. URL: http://epubs.siam.org/doi/10.1137/S0363012901385691 (visited on 08/29/2023).

[20] Benjamin Sanchez-Lengeling et al. “A Gentle Introduction to Graph Neural Networks”. In: *Distill* 6.8 (Aug. 17, 2021), 10.23915/distill.00033. ISSN: 2476-0757. DOI: 10.23915/distill.00033. URL: https://distill.pub/2021/gnn-intro (visited on 06/19/2023).

[21] Justin Gilmer et al. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 6, 2017, pp. 1263–1272. URL: https://proceedings.mlr.press/v70/gilmer17a.html.

[22] Peter W. Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. In: *CoRR* abs/1806.01261 (2018). arXiv: 1806.01261. URL: http://arxiv.org/abs/1806.01261.

[23] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[25] Yujia Li et al. “Gated Graph Sequence Neural Networks”. In: *Proceedings of ICLR’16*. Edition: Proceedings of ICLR’16. Apr. 2016. URL: https://www.microsoft.com/en-us/research/publication/gated-graph-sequence-neural-networks/.

[26] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations*. 2018.

[27] *Publications - Dimensions*. URL: https://app.dimensions.ai/discover/publication (visited on 08/31/2023).

[28] Sorin Grigorescu et al. “A Survey of Deep Learning Techniques for Autonomous Driving”. In: *Journal of Field Robotics* 37.3 (Apr. 2020), pp. 362–386. ISSN: 1556-4959, 1556-4967. DOI: 10.1002/rob.21918. arXiv: 1910.07738[cs]. URL: http://arxiv.org/abs/1910.07738 (visited on 06/16/2023).

[29] Edouard Leurent. “A Survey of State-Action Representations for Autonomous Driving”. Oct. 2018. URL: https://hal.science/hal-01908175 (visited on 09/11/2023).

[30] Tommy Tram et al. “Learning negotiating behavior between cars in intersections using deep q-learning”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3169–3174.

[31] David Isele et al. “Navigating occluded intersections with autonomous vehicles using deep reinforcement learning”. In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2034–2039.

[32] Pin Wang, Ching-Yao Chan, and Arnaud de La Fortelle. “A reinforcement learning based approach for automated lane change maneuvers”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1379–1384.

[33] Carl-Johan Hoel, Krister Wolff, and Leo Laine. “Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2018, pp. 2148–2155. DOI: 10.1109/ITSC.2018.8569568. arXiv: 1803.10056[cs]. URL: http://arxiv.org/abs/1803.10056 (visited on 09/12/2023).

[34] Xiaobai Ma et al. “Reinforcement Learning for Autonomous Driving with Latent State Inference and Spatial-Temporal Relationships”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021 IEEE International Conference on Robotics and Automation (ICRA). Xi’an, China: IEEE, May 30, 2021, pp. 6064–6071. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9562006. URL: https://ieeexplore.ieee.org/document/9562006/ (visited on 06/14/2023).

[35] Qi Liu et al. “Graph Convolution-Based Deep Reinforcement Learning for Multi-Agent Decision-Making in Interactive Traffic Scenarios”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 4074–4081.

[36] Meng Xiaoqiang et al. “Graph Convolution Reinforcement Learning for Decision-Making in Highway Overtaking Scenario”. In: *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*. 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA). Chengdu, China: IEEE, Dec. 16, 2022, pp. 417–422. ISBN: 978-1-66540-984-1. DOI: 10.1109/ICIEA54703.2022.10006015. URL: https://ieeexplore.ieee.org/document/10006015/ (visited on 06/12/2023).

[37] Fan Yang et al. “Generalized Single-Vehicle-Based Graph Reinforcement Learning for Decision-Making in Autonomous Driving”. In: *Sensors* 22.13 (2022). ISSN: 1424-8220. DOI: 10.3390/s22134935. URL: https://www.mdpi.com/1424-8220/22/13/4935.

[38] Maria Huegle et al. “Dynamic Interaction-Aware Scene Understanding for Reinforcement Learning in Autonomous Driving”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris, France: IEEE, May 2020, pp. 4329–4335. ISBN: 978-1-72817-395-5. DOI: 10.1109/ICRA40945.2020.9197086. URL: https://ieeexplore.ieee.org/document/9197086/ (visited on 06/12/2023).

[39] Ahmad El Sallab et al. *End-to-End Deep Reinforcement Learning for Lane Keeping Assist*. Dec. 13, 2016. arXiv: 1612.04340[cs, stat]. URL: http://arxiv.org/abs/1612.04340 (visited on 06/14/2023).

[40] Pin Wang, Hanhan Li, and Ching-Yao Chan. "Continuous control for automated lane change behavior based on deep deterministic policy gradient algorithm". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1454–1460.

[41] Daniele Gammelli et al. "Graph Neural Network Reinforcement Learning for Autonomous Mobility-on-Demand Systems". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021 60th IEEE Conference on Decision and Control (CDC). Austin, TX, USA: IEEE, Dec. 14, 2021, pp. 2996–3003. ISBN: 978-1-66543-659-5. DOI: 10.1109/CDC45484.2021.9683135. URL: https://ieeexplore.ieee.org/document/9683135/ (visited on 06/12/2023).

[42] Paul Almasan et al. "Deep Reinforcement Learning meets Graph Neural Networks: exploring a routing optimization use case". In: *Computer Communications* 196 (Dec. 2022), pp. 184–194. ISSN: 01403664. DOI: 10.1016/j.comcom.2022.09.029. arXiv: 1910.07421[cs]. URL: http://arxiv.org/abs/1910.07421 (visited on 06/14/2023).

[43] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[44] OpenAI. *Soft Actor-Critic — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/algorithms/sac.html (visited on 09/05/2023).

[45] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. May 19, 2016. DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473[cs, stat]. URL: http://arxiv.org/abs/1409.0473 (visited on 09/08/2023).

[46] Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive are Graph Attention Networks?" In: *International Conference on Learning Representations*. 2021.

[47] CARLA Team. *CARLA*. CARLA Simulator. URL: http://carla.org// (visited on 10/31/2023).

[48] *SUMO Documentation*. URL: https://sumo.dlr.de/docs/index.html (visited on 10/31/2023).

[49] Edouard Leurent. *An Environment for Autonomous Driving Decision-Making*. Publication Title: GitHub repository. 2018. URL: https://github.com/eleurent/highway-env.

[50] *highway-env Documentation*. URL: https://highway-env.farama.org/user_guide.html (visited on 09/16/2023).

[51] Matthias Fey and Jan E. Lenssen. “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[52] Shengyi Huang et al. “CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms”. In: *Journal of Machine Learning Research* 23.274 (2022), pp. 1–18. URL: http://jmlr.org/papers/v23/21-1342.html.

[53] Arsenii Kuznetsov et al. *Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics*. May 8, 2020. DOI: 10.48550/arXiv.2005.04269. arXiv: 2005.04269[cs,stat]. URL: http://arxiv.org/abs/2005.04269 (visited on 11/19/2023).