ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

IMPROVING LANE CHANGE DECISIONS IN AUTONOMOUS DRIVING USING ADVERSARIAL LEARNING

M.Sc. THESIS

Aytuğ Onurhan EFİL

Department of Control and Automation Engineering

Control and Automation Engineering Programme

JUNE 2025

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

IMPROVING LANE CHANGE DECISIONS IN AUTONOMOUS DRIVING USING ADVERSARIAL LEARNING

M.Sc. THESIS

Aytuğ Onurhan EFİL
(504211102)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Prof. Dr. Mustafa DOĞAN

JUNE 2025

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

OTONOM SÜRÜŞTE ŞERİT DEĞİŞTİRME KARARLARININ KARŞIT ÖĞRENME YÖNTEMİYLE İYİLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Aytuğ Onurhan EFİL
(504211102)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Mustafa DOĞAN

HAZİRAN 2025

Aytuğ Onurhan EFİL, a M.Sc. student of ITU Graduate School student ID 504211102, successfully defended the thesis entitled "IMPROVING LANE CHANGE DECISIONS IN AUTONOMOUS DRIVING USING ADVERSARIAL LEARNING", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Mustafa DOĞAN** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Müjde GÜZELKAYA** Istanbul Technical University	.....
	Assoc. Prof. Dr. Yavuz EREN Yıldız Technical University	.....

Date of Submission : 30 May 2025
Date of Defense : 20 June 2025

To my spouse and my family,

FOREWORD

This thesis is submitted in partial fulfillment of the Master's Degree at Istanbul Technical University. The thesis of choice for this study is to develop an algorithm through which a car can change lanes on the highway on its own but in a manner consistent with the driver's preferences without the driver giving any specific instructions.

I have learned many things from this thesis and have gained much experience in this chapter. During the research inverse reinforcement learning, imitation learning, adversarial networks, reinforcement learning, and decision making were explored, which was quite enjoyable and informative. Of interest to me has been decision-making and autonomous lane changing, and while doing the research, the optimization component was fun and exhilarating.

In the first place, it felt like I was creating real world value by tweaking a virtual simulation of the actual world. The proposed research is expected to be utilized in real world industrial projects in the near future.

I want to take this chance to thank my supervisor, Prof. Dr. Mustafa Doğan, for his encouragement during my research. His feedback at important junctures made a great difference to me. Also, I want to thank my wife and my family for their love and support. They always inspired me to excel and for this, I have a deep appreciation for them.

June 2025

Aytuğ Onurhan EFİL
(Mechatronics Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

AV : Autonomous Vehicle
AI : Artificial Intelligence
ADAS : Advanced Driver Assistance Systems
BEV : Bird's Eye View
ML : Machine Learning
DL : Deep Learning
RL : Reinforcement Learning
IRL : Inverse Reinforcement Learning
IL : Imitation Learning
MDP : Markov Decision Process
DP : Dynamic Programming
GAIL : Generative Adversarial Imitation Learning
AIRL : Adversarial Inverse Reinforcement Learning
GAN : Generative Adversarial Network
DNN : Deep Neural Networks
CNN : Convolutional Neural Networks
RNN : Recurrent Neural Networks
PPO : Proximal Policy Optimization
MLP : Multilayer Perceptron
VAE : Variational Autoencoder
PCA : Principal Component Analysis
t-SNE : t-distributed Stochastic Neighbor Embedding
JSD : Jensen-Shannon Divergence

SYMBOLS

s	**:** State
s'	**:** Next State
Val	**:** Value, numeric demonstration of a state
r	**:** Reward
r(s)	**:** Reward function with respect to state
r(s,a)	**:** Reward function with respect to state and performing an action
r(s,a,s')	**:** Reward function with respect to state, taking an action and next state
A	**:** Action, e.g., left, right, accelerate, decelerate
γ	**:** Determines the weight of the rewards in the distant future compared to rewards in the near future
ε	**:** Epsilon
$\widehat{\boldsymbol{g}}$	**:** Estimator
$\widehat{\boldsymbol{E}}_{\boldsymbol{t}}$	**:** Empirical average over a finite batch of sample
$\boldsymbol{\pi}_{\boldsymbol{\theta}}$	**:** Policy
$\widehat{\boldsymbol{A}}_{\boldsymbol{t}}$	**:** Advantage function estimator
$\boldsymbol{L}^{\boldsymbol{PG}}$	**:** Loss of policy gradient
$\boldsymbol{L}^{\boldsymbol{CLIP}}$	**:** Loss of clipped surrogate objective
$\boldsymbol{r}_{\boldsymbol{t}}(\boldsymbol{\theta})$	**:** Probability ratio
t	**:** Timestep
$\boldsymbol{\theta}$	**:** Policy parameter
$\boldsymbol{\epsilon}_{\boldsymbol{clip}}$	**:** Clipping value
$\mathbf{L}^{\mathbf{VF}}$	**:** Value function loss
$\mathbf{V}_{\boldsymbol{\theta}}$	**:** Current value
$\widehat{\mathbf{R}}_{\mathbf{t}}$	**:** Target reward value
$\boldsymbol{\tau}$	**:** Trajectory
$\boldsymbol{c}_{\boldsymbol{\theta}}(\boldsymbol{\tau})$	**:** Unknown cost function
$\boldsymbol{p}(\boldsymbol{\tau})$	**:** Distribution of the demonstration
$\boldsymbol{q}(\boldsymbol{\tau})$	**:** Generator's density
$\boldsymbol{D}_{\boldsymbol{\theta}}(\boldsymbol{\tau})$	**:** The form of the AIRL's discriminator

$\boldsymbol{L_D(D)}$: AIRL's discriminator's loss function

$\boldsymbol{L_G(G)}$: AIRL's generator's loss function

$\boldsymbol{f_\theta(\tau)}$: Learnt function

$\boldsymbol{H(\pi)}$: The causal entropy of the policy

$\boldsymbol{\psi(c)}$: Convex reward function regularizer

$\boldsymbol{E}$: Entropy

$\boldsymbol{D_w(s,a)}$: GAIL's discriminator probability

$\boldsymbol{\lambda}$: Entropy coefficient

$\mathbf{Q(s,a)}$: State-action value function

$\mathbf{act}_{dim}$: The dimension of the action space

$\boldsymbol{a_{ev}}$: Acceleration of the ego vehicle

$\boldsymbol{P}$: Proportion gain coefficient of acceleration

$\boldsymbol{v_d}$: Desired speed

$\boldsymbol{v_c}$: Current speed

LIST OF TABLES

Page

LIST OF FIGURES

Page

IMPROVING LANE CHANGE DECISIONS IN AUTONOMOUS DRIVING USING ADVERSARIAL LEARNING

SUMMARY

Autonomous vehicles are the future of transportation. This is due to the technological improvement for future smart cars since they will have AI based driving, they will be driverless, accident-free, and efficient. The improvement in technology will minimize the problems such as human failure, delayed driver's, and the unsecured lane change, and hence traveling becomes safe. To achieve these goals, car makers have invested in research areas that are relevant to the current challenges in order to meet the ideal goals. With the current development in machine learning, autonomous vehicles are now being used within specific geographic regions, and it is expected that they will be used in a wide area in the market in the near future.

Autonomous vehicles have been in the spotlight for the past decade. It is no coincidence that this is simultaneous with the creation of deep learning models. Deep learning models were first introduced to autonomous vehicle systems through the use of convolutional neural networks (CNN) for image classification. The obtained outcomes motivated the researchers to create self-driving cars that utilize deep neural networks within the perception layer of advanced driver assistance systems (ADAS). The perception layer of ADAS defines the sensors that detect and classify objects within the ego vehicle's surroundings, including other cars, pedestrians, and cyclists. The sensor fusion layer receives the bird's eye view (BEV) map of the environment and recognizes and tracks other objects in the surroundings. Then, a decision making algorithm identifies high level actions (such as change lane, acceleration, or deceleration) that optimize a given cost function, to avoid collision and to arrive at a given location with the shortest possible time. After the optimal high level actions are determined, the low level controllers produce the required speed demand and steering angle to track the specified trajectories. This thesis investigates self-driving lane changing strategies using imitation learning, inverse reinforcement learning, adversarial neural network, and reinforcement learning policy.

One of the most significant parts of the decision making component of ADAS is autonomous lane changing during highway driving. Current autonomous vehicles available in urban areas are limitedly capable of performing safe and reliable lane changes on their own without the help of a human driver. Almost all organizations develop driving strategies to come up with reasonable decisions based on the context of the traffic. However, it is possible that such approaches may not be sufficiently comprehensive to capture a variety of scenarios that may occur on the road depending on the environment, the road and other traffic.

Over the last few years, reinforcement learning algorithms could produce discrete actions in simulation environments. Hence, we implemented these algorithms for the decision making and planning of highway controls in self-driving vehicles. For this, we also built Carla and Highway-env environments with four lanes and actions like

left turn, right turn, stay in lane, speed up, and slow down. Finally, we compare the result of two different adversarial learning approaches which are adversarial inverse reinforcement learning (AIRL) and generative adversarial imitation learning (GAIL) and present different work scenarios to figure out the applicability and efficiency of the proposed schemes.

OTONOM SÜRÜŞTE ŞERİT DEĞİŞTİRME KARARLARININ KARŞIT ÖĞRENME YÖNTEMİYLE İYİLEŞTİRİLMESİ

ÖZET

Otonom araç teknolojisi giderek daha fazla ilgi çekiyor ve gelecekte ulaşımın şeklini değiştirecek gibi görünüyor. Yapay zeka ve otomasyondaki gelişmelere bağlı olarak ilerleyen zamanlarda akıllı araçlar sürücü olmadan daha güvenli ve verimli bir şekilde yol alacaklar. İnsan sürücülerin yapabileceği hataları en aza indirerek seyahatleri daha güvenli hale getirecekler. Otomobil endüstrisi bu hedefe ulaşabilmek adına mevcut sorunları çözmek için yoğun bir şekilde çalışıyor. Makine öğrenme algoritmalarındaki ilerlemeler sayesinde otonom araç teknolojisinin kademeli olarak piyasaya sunulmaya başlanacak gibi gözüküyor. Son on yılda otonom araç teknolojisi büyük ilgi gördü ve gelişimine devam ediyor.

Makine öğrenmesi paradigmalarının ve algoritmalarının gelişimi, büyük ölçekli gerçek zamanlı sensör verilerinin toplanması ve etiketlenmesi ile LiDAR ve görüntü işleme sistemleri gibi bileşenlerin uygulanması, tam otonomiyi sağlama çabasında önemli rol oynamaktadır. Otonom sürüş alanında kaydedilen ilerlemelere rağmen, yüksek otomasyona sahip araçlar hala kontrollü ortamlar ve belirli coğrafi bölgelerle sınırlıdır. Bu karmaşıklığı sadeleştirmek için, Seviye 0'dan (otonom olmayan) Seviye 5'e (tam otomatik sürüş) kadar uzanan otonomi seviyeleri tanımlanmıştır. Tam sürüş otomasyonunu ifade eden Seviye 5 henüz gerçekleştirilememiş olsa da daha düşük otonomi seviyeleri belirli uygulamalarda kullanılmaktadır.

Derin öğrenme ve takviyeli öğrenme algoritmalarının geliştirildikleri zaman diliminde ADAS'ın da gelişim göstermesi tesadüfi değildir. İlk olarak görüntü sınıflandırması için CNN'nin kullanılmasıyla otonom araç uygulamalarına entegre olan derin öğrenme algoritmaları umut verici sonuçlar ortaya koymuş ve bu da araştırmacıları ADAS algılama katmanında derin sinir ağlarını uygulamaya teşvik etmiştir. ADAS'taki algılama katmanı ego aracın çevresindeki aktörleri (örneğin arabalar, yaya ve bisiklet sürücüleri gibi) tanımlar ve kategorize eder. Kuşbakışı haritasını oluşturduktan sonra sensör birleştirme katmanı diğer aktörleri belirler ve takip eder. Sonrasında bir karar verme algoritması genellikle çarpışmayı önlemek ve hedef konuma minimum sürede ulaşmak için belirli bir masraf fonksiyonunu optimize etmek amacıyla yüksek seviyeli aksiyonlar (örneğin şerit değiştirme, hızlanma veya durma) üretir. Basitçe optimal yüksek seviyeli aksiyonlar üretildikten sonra alt seviyede yer alan kontrolcüleri belirtilmiş komutları takip etmek için gerekli olan hız talebi ve direksiyona açısını üretir.

Bu çalışma kapsamında taklit öğrenme, ters takviyeli öğrenme, karşıt sinir ağı, takviyeli öğrenme politikası gibi farklı yaklaşımları kullanarak otonom şerit değiştirme yöntemleri araştırılmaktadır. Otonom şerit değiştirme sürecinde özellikle otoyollarda ADAS'ın karar verme sistemi için oldukça kritik bir rol oynamaktadır. Güncel olarak kentsel alanlarda kullanılan otonom arabalar, insan sürücü müdahalesi veya varlığı olmadan güvenli ve güvenilir şerit değişiklikleri yapmaları kısıtlanmıştır. Birçok kuruluş genellikle belirli teknikler kullanarak sezgisel kararlar üretmektedir.

Fakat bu teknikler çevresel ve trafik koşullarına göre değişen sürüş durumlarını yeterince kontrol etmede zorlanabilir.

Son yıllarda takviyeli öğrenme algoritmaları sürekli ve ayrık ortamları dikkate alarak umut vadeden sonuçlar göstermiştir. Bu sebeple otoyol sürüş görevlerinde kullanılmak üzere bu algoritmaları otonom araçlar için stratejik kararlarda kullanmaya karar verdik. Bu amaçla Carla ile Highway-env gibi ortamları dört şeritli otoyol oluşturarak modellendirdik; hızlanma, yavaşlama, sabit hızda seyretme; sol şeride geçme, sağ şeride geçme ve şeritte kalma gibi eylemleri simüle ettik. Son olarak iki ayrı çevrim dışı öğrenme tekniği sonuçlarını karşılaştırarak gelecekteki çalışma senaryolarında uygulanabilirliği gözlemledik.

Kullandığımız algoritmalar her adımda ego aracın bulunduğu şeritteki en yakın aracı belirler ve ego aracın şerit numarasıyla birlikte şerit değiştirmeye mi yoksa kendi şeridi üzerinde kalmaya mı karar vereceğini belirler. Her durum-eylem çifti beklenen gelecekteki ödülü temsil eder. Simülasyonda belirli bir sayıda adım için çalışma yapıldığında özellikle ödüller sabit değerlere yakınsar. Test aşamasında model en yüksek beklenen ödüle sahip manevrayı seçerek insan kontrolündeki sürüş davranışına uygun olarak hedefine ulaşır. Bu yöntemi uygulayarak, araçların çarpışmasını önlemek için insan sürücülerin politikalarına benzer bir politika öğrenen bir araç gözlemledik. Ancak eğitim verilerinde yeterince temsil edilmeyen sürüş manevraları nedenli zorluklar yaşandı. Etkenlerin öğrenmesi için yeterli çeşitlilikte gösteri verisi olmadığında, görülmeyen koşullara başarılı bir şekilde genelleme yapma yeteneklerinin azaldığını gördük. Ek manevra seçenekleri belirlendi ve gerçek dünya sürüşünde daha tutarlı otoyol otonom sürüş kararları üretmek için ek sürüş özellikleri incelendi.

Verilerden öğrenme ve daha önce karşılaşılmamış durumlara genelleme yapma yeteneği otonom araç uygulamaları için önemli bir faktördür. Sürüşte gerçek zamanlı olayların hem reaktif hem de proaktif olarak algılanması gerekmektedir; bu da dinamik senaryoların, görevlerin ve risklerin karmaşık doğasının bir parçasıdır ve bu durum karmaşık bir sorun alanıdır. Böylesine geniş bir sorun alanını ele almak için bilgi durumlarını ve arama süreçlerini kapsayan bir veri kümesi geliştirmek ve sürdürmek doğası gereği zordur.

Ayrıca, otonom araçlar bağlamında otoyol sürüş görevlerinin karmaşıklığı analiz edildi ve analizi doğrulamak için durum-eylem ödülleri ve çarpışma ölçümleri şeklinde kanıtlar sağlandı. Son olarak, bu çalışma taklit öğrenmenin ve ters takviye öğrenmesinin takviye öğrenmesinin belirli sürüş senaryoları için eğitim ve genelleme yeteneklerini geliştirdiğini gösterse de özetlenen sürüş senaryolarında karar alma yeteneklerini geliştirmek için birleşik bir yaklaşım olarak en son algoritmaları entegre etmek için daha fazla araştırmaya ihtiyaç vardır. Ek olarak, ego aracının çevresini matematiksel olarak temsil eden modülleri içeren gerçekçi bir simülasyon ortamına geçmek, bu araştırmanın pratik yararlarını daha da doğrulayacaktır.

Özetle, bu çalışma CARLA simülasyon ortamında otoyol sürüş senaryolarında GAIL ve AIRL algoritmalarının karar verme performanslarını değerlendiren bir çalışmadır ve çevrimdışı öğrenme yöntemlerinin otonom sürüşte yüksek seviyeli karar alma için güvenliği sağlamada konusundaki performanslarını değerlendirmektedir. Uzman sürücüden öğrenilen stratejilerle ajanların ödül çıkarımı yaparak etkili şekilde davranış sergilemesi sağlanmış, PPO ile entegre edilen hibrit yapı sayesinde öğrenme süreci daha dengeli ve verimli hale getirilmiştir. Bu algoritmalar gerçek dünyada uygulanabilir otonom sürüş sistemleri için önemli bir adım niteliğindedir; ancak, en

iyi kararları üretmek için ek bileşenler ile zenginleştirilmelidir. Taklit ve ters takviye öğrenme algoritmaları ile verimli bir karşıt ağ mimarisi kullanılarak eğitilen bir araçla çarpışma önleme ve şerit değiştirme politikasını öğrenerek bir aracın güvenli ve akıllı sürüş yapmasını sağlamayı hedeflenmektedir.

1. INTRODUCTION

Machine learning has recently applied to the domain of self-driving vehicles, especially autonomous cars, for the last 10 years. It is evident that there is no one way to solve the issue of safe and fully autonomous vehicles in real life. The enhancement of machine learning paradigms and algorithms, the collection and labelling of large-scale real-time sensor datasets, and the implementation of components such as LiDAR and computer vision cameras have all been important in the effort to achieve full autonomy. Despite the progress made in autonomous driving by automotive technology companies such as Zoox, Waymo, and Tesla's Autopilot, scalable and fully autonomous vehicles are still limited to controlled environments and geographic regions. To simplify this complexity, levels of autonomy have been introduced, starting from Level 0 (non-autonomous) and ending with Level 5 (fully automated driving). Although the Fifth Level of autonomy representing full driving automation has not yet been achieved, lower levels of automation are already in use in specific applications. Some examples include Level 1 (driver assistance) which controls steering or acceleration/deceleration such as cruise control, Level 2 (partial automation) such as ADAS that can control both steering and acceleration/deceleration, and Level 3 (conditional automation) systems that can detect the environment and make decisions for humans. For example, Waymo One is a Level 4 (high automation) that only operates in geofenced areas and is guided by GPS. Furthermore, inverse reinforcement learning (IRL) and imitation learning (IL) have come into use for the advancement of self-driving driving systems. IL and IRL address the problem of learning from observations, especially when the agent is designed to mimic expert demonstrations, providing a data based way to train agents to learn from human behavior. This can greatly speed up the learning process by cutting down on the need for exploration in high cost environments. The overall goal of this research is to compare and assess RL, IL, and IRL methods against a minimalistic highway autonomous driving environment. The simulations mimic the

specific highway driving conditions autonomous systems and both IL and IRL are used to deploy artificial intelligence agents in similar highway driving simulations.

1.1 Aims and Objectives

The primary purpose of this thesis is to analyze the effectiveness of adversarial inverse reinforcement learning and generative adversarial imitation learning in typical highway driving scenarios. The current consensus within the literature indicates that data-based techniques such as IL and IRL when combined with adversarial approaches can produce satisfactory results for autonomous driving on highways. Hence, the central hypothesis is: Can data driven reinforcement learning through adversarial inverse reinforcement learning or generative adversarial imitation learning produce reasonable behavior in a highway driving environment? In order to be consistent with the available literature and to be able to make meaningful baseline comparisons:

- Check the rewards obtained during learning and where applicable, check the success rates of the training completions.
- Evaluate collision avoidance metrics during training.
- Test the agents' capability to generalize to unseen scenarios.
- Discuss the highway driving tasks chosen, and the risk management capabilities of the autonomous agents in more complex environments.

In achieving these objectives, this thesis hopes to provide useful results to the scientific community and to the field of autonomous driving.

1.2 Literature Review

In this study, the literature review was conducted on several critical aspects of autonomous vehicle research:

- Examining current autonomous vehicle research in relation to existing industry standards.
- Examining the usage of reinforcement learning in realistic driving simulators as a safe and cost efficient way of researching self-driving vehicle technologies.

- Comparison of reinforcement learning results with less popular strategies like adversarial inverse reinforcement learning and generative adversarial imitation learning.
- A bottom-up research approach is adopted to analyze highway driving tasks in order to understand the basic performance of agents in the context of autonomous vehicles.
- A study of the attempts to integrate safety measures and find out about the collision avoidance failures in the autonomous vehicle systems.
- Reviewing standard experimentation methods in autonomous vehicle research, including evaluation processes, generalization capabilities, and metric collection techniques.

1.2.1 A brief review of autonomous vehicles

At the beginning, the domain of the autonomous vehicle employed model free reinforcement learning algorithms. At the same time, frameworks like Atari offered a relatively low cost computational environment with easily replicable and comparable evaluations, which led to the advancement of many algorithms and approaches to autonomous control. However, the progression could not be easily sustained to more sophisticated environments such as simulators or real world driving scenarios. A major issue was the enormous requirement for datasets which had to be processed by agents that run online or in real-time within vehicles. This gap between the possible and the practical resulted in the development of model-based reinforcement learning and data-centric approaches like imitation learning and inverse reinforcement learning [1]. Model based learning is currently employed for the development of production level AI agents in real world autonomous driving. However, the approach is expensive and not easy to propagate. Moreover, imitation learning and inverse reinforcement learning have started to show promise in improving the performance of reinforcement learning. These paradigms learn from either expert data from models or data collected through human interaction with the environment. However, each of them has its trade-offs. Imitation learning learns rewards from expert demonstrations, while inverse reinforcement learning requires state-action pairs from the dataset. Hence, both methods are data dependent.

1.2.2 Simulation environment

The research employs two specific driving simulators as foundational tools: CARLA and Highway-env. The CARLA as shown in Figure 1.1, open-source urban driving simulator is a crucial research instrument for reinforcement learning investigations and safety assessments, offering pre-training models for transferring learning applications with real vehicles and benchmarking data-driven techniques such as imitation and inverse reinforcement learning. According to the original CARLA paper [2], the authors explored three primary approaches to learning for self-driving vehicles: The authors researched three main learning approaches for autonomous vehicles which included a modular supervised learning pipeline alongside two end-to-end pipelines that combined with the learning methods such as reinforcement learning, inverse reinforcement learning and imitation learning [2], the learning methods were also studied in the other papers with using CARLA simulator [3][4][5]. Their researches presented important discoveries about how imitation learning compares with inverse reinforcement learning while also demonstrating generalization's essential role when systems encounter new situations. The research demonstrated strong performance outcomes through adversarial inverse reinforcement learning and generative adversarial imitation learning across both training and non-training scenarios when they implemented reinforcement learning policies. The research highlighted how supervised modular learning pipelines exhibited superior performance during training phases.

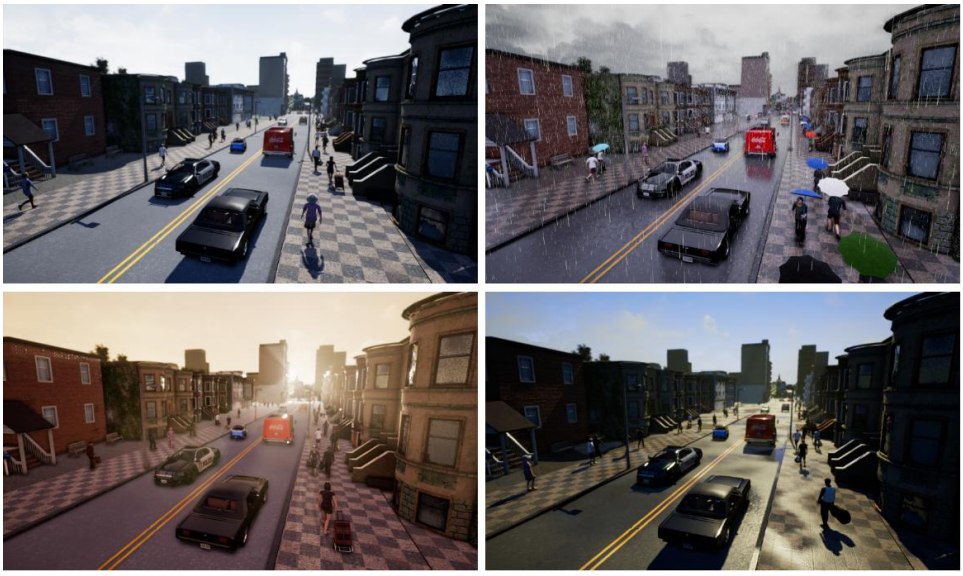

Figure 1.1 : View from CARLA simulator

Highway-env [6] as shown in Figure 1.2, serves as a basic driving simulator created explicitly for researching highway driving tasks. The testing environment offers structure for autonomous driving algorithm evaluation through basic kinematics and scenarios that enable research focus on fundamental driving actions including lane-keeping and collision avoidance in addition to overtaking [7][8][9]. Highway-env stands out as a lightweight platform that enables faster evaluation of reinforcement learning, imitation learning, and inverse reinforcement learning techniques when compared to the complex and resource-intensive CARLA simulator. The basic design of the platform makes it suitable for evaluating how well agents generalize in different highway scenarios and how they perform in changing traffic conditions. The environment works alongside CARLA to enable specialized exploration of driving tasks which function as essential benchmarks for the evaluation of autonomous highway driving learning methods.

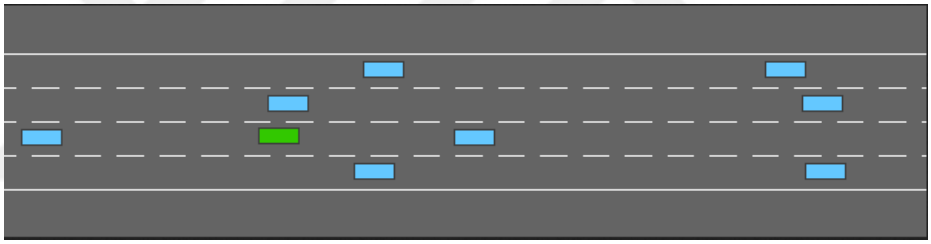

Figure 1.2 : A view from Highway-env simulator

1.2.3 Learning to drive in the simulation environment

Using reinforcement learning techniques on complex simulators like CARLA comes with certain challenges, as highlighted in the literature. These include the high demand for computational resources and, most importantly, the low level of transparency in the agent-environment interaction. Driving is a complicated series of actions in the real world with high failure costs, low error tolerance, and a large action space where many independent and interdependent variables act at the same time. This is rather challenging to pinpoint the failure cases or to know which of the driving subtasks, like overtaking, results in negative consequences such as collision. For instance, one study used generative adversarial imitation learning (GAIL) [10] for autonomous driving in urban environments. The approach took high dimensional inputs from multiple front cameras, vehicle speeds and trajectory points. The results implies that the GAIL based model was able to learn the expert driving policies, while an improved variant of GAIL

that incorporates behavior cloning was found to converge faster and was more stable during training [11]. Complementary, a second study employed an augmented adversarial inverse reinforcement learning (AIRL) [12] framework to decision making in autonomous driving. The improved AIRL model integrated semantic rewards to learn both the driving policy and the reward function jointly. When implemented on complicated scenarios in CARLA, the augmented AIRL outperformed baseline methods on various aspects and produced performance like that of expert drivers [13]. In another work on the limitations of imitation learning for autonomous driving [14], observed that while supervised learning with behavior cloning is better than traditional reinforcement learning; in executing complex lateral and longitudinal maneuvers, the former requires predefined executable actions. However, the study also notes some problems, including the fact that when dynamic behaviors are introduced, there are generalization issues, and the dataset is biased and tends to overfit. Nevertheless, this research combined pre-training with imitation learning and to some extent reinforcement learning to leverage the strengths of these approaches. Hence, using a bottom-up approach in virtual environments can help to understand the basic principles. Although the specific methods in that study were not directly applied here, they were used as reference and foundation for the research approach chosen in this study.

1.2.4 Potential of offline learning

Inverse reinforcement learning and imitation learning have demonstrated their effectiveness in using offline datasets to design effective policies. Offline reinforcement learning seeks to learn optimal policies from datasets gathered with expert demonstration, all without further interaction with the environment, which can be useful when data collection is high cost or impossible. Imitation learning can quickly design policies using expert representations of the input but fails to generalize correctly to distributions that are unlike the training data when the dataset contains non-optimal trajectories [15].

IRL is concerned with learning the reward function from expert representations of the reward, which can then be used to determine the optimal policy. Offline IRL approaches like those that employ GAN-based data augmentation have been successful in defining reward functions that are better than conventional methods

across a range of environments [16]. These techniques thus present a way of using offline datasets to design robust and adaptive policies, thereby mitigating the imitation reinforcement learning gap [17].

1.2.5 Combination of learning algorithms

Generative adversarial networks, paired with reinforcement learning and inverse reinforcement learning are some of the exciting tools used in autonomous driving, also paired with reinforcement learning and imitation learning present yet another promising approach. Suppose you want to teach a vehicle to run on a complicated highway without explicit human demonstrations but rather by adversarial feedback.

Imagine that you are pitted against two people: one will act as a generator to create driving behaviors that are indistinguishable from an expert driving, and the other will act as a discriminator that critiques and identifies the faults. This makes it a continuous loop, a driving coach in constant challenge of the student to achieve perfection. Algorithm basically observes expert demonstration and works to get at the rewards underlying their decisions.

1.3 Contribution

- Firstly, this thesis is the to evaluate and compare the performance of GAIL and AIRL specifically in a highway driving within the CARLA simulation environment, focusing on decision-making tasks.
- Unlike prior work that applies these adversarial learning frameworks to general robotics or classical control problems, this study makes a novel contribution by adapting adversarial imitation and inverse reinforcement learning methods directly to autonomous driving scenarios.
- The study introduces an interpretable state-action design for autonomous highway driving tasks, making it easier to classify and benchmark driving behaviors, which can provide as a reusable basis for future work in this domain.
- The thesis assumes that expert drivers follow an optimal policy based on latent reward functions and demonstrates how an agent can learn such expert strategies by inferring rewards and mimicking expert behavior in a structured and measurable environment.

- It proposes a hybrid learning architecture that integrates PPO with adversarial training to improve the stability and sample efficiency of the learning process while preserving adversarial dynamics that drive behavioral realism.
- The work shows how expert demonstrations, when used in an adversarial setting, can accelerate early-stage learning by providing strong initial guidance, while PPO ensures balanced exploration and exploitation in the online training phase.
- The study serves as a step forward toward robust, real-world applicable autonomous driving systems, offering insights into how adversarial learning, imitation learning, and reinforcement learning can be effectively combined for complex behavioral tasks.

1.4 Thesis Structure

The next chapters in this dissertation are organized as follows:

Chapter Two: Background and Related Work - This chapter highlights methods, algorithms, and the theories that give underpinning to the key areas of interest in artificial intelligence such as deep learning, reinforcement learning, inverse reinforcement learning, and imitation learning.

Chapter Three: Environment - This chapter introduces the frameworks and simulation tools utilized in this research and describes the various components of the self-driving vehicle simulations along with their integration into the selected framework.

Chapter Four: Results and Discussions - This chapter gives the experimental design aspects like hyperparameter selection and applicable experiments. Further, it states the results for each learning methods that were explored throughout this study.

Chapter Five: Conclusion - This chapter evaluates the performance of the learning methods within the autonomous vehicle scenarios and reflects on how they compare to the wider research incidental in the literature review. It also proposes potential implementations for future research and highlights the study's contribution to the field.

2. BACKGROUND AND RELATED WORK

Machine learning (ML) has become an important element in developing autonomous vehicle systems, with several approaches such as deep learning, reinforcement learning, inverse reinforcement learning, and imitation learning playing extremely critical roles. In essence, deep learning has been used for smart driving policy formulation, optimizing adaptively in complex and dynamic environments to enable decision-making from high dimensional data input [18][19]. Reinforcement learning (RL) helps the autonomous vehicles receive feedback about their autonomous performance from their environment and adjust their learning based on it in an attempt to reach its goals [20][21][22]. Inverse reinforcement learning (IRL) and imitation learning further allow vehicles to predict and imitate human driving behavior to make better predictions as reaction to other road users [23][24][25]. Collectively, these methodologies further amplify the performance of autonomous driving systems, making it more robust, effective, and complex to tackle when concerning real-world driving conditions, as also illustrated in the study of Figure 2.1.

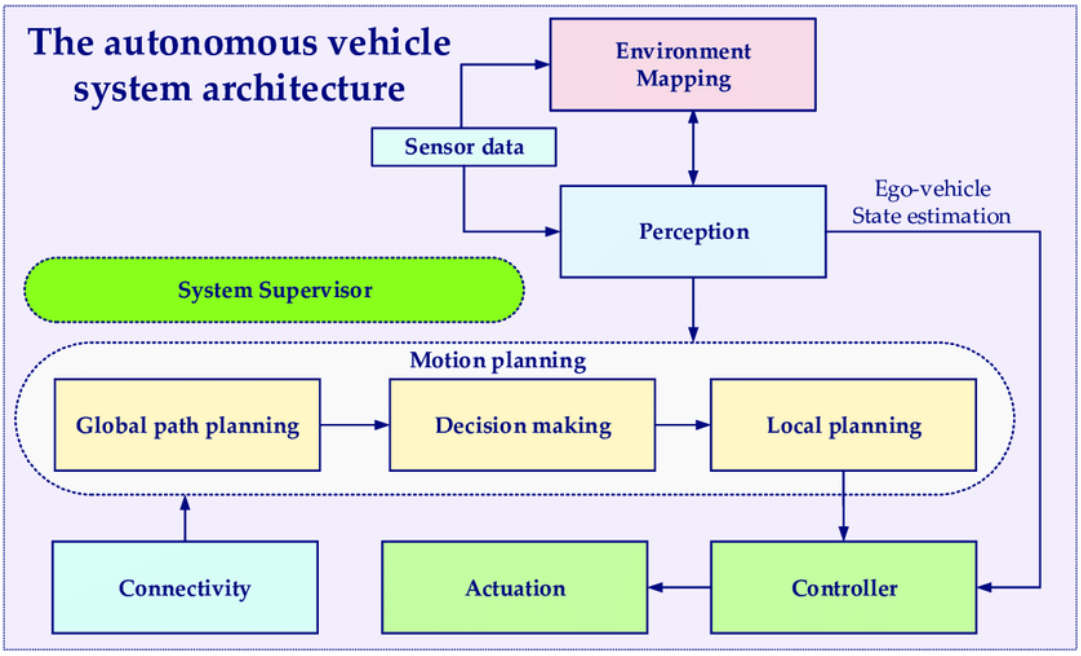

Figure 2.1 : Decision making process [26]

2.1 Deep Learning

2.1.1 Deep learning networks

Deep learning (DL) has become the core development process of autonomous driving systems, using various methods of neural network architectures, such as generative adversarial network (GANs), variational autoencoders (VAEs), deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) to enhance the perception and decision-making processes of vehicles. The data collection process is the most demanding of all steps in the operation and consists mostly of collecting large datasets from real-world analysis and simulations. For instance, simulated environments like the CARLA autonomous vehicle gaming environment generate high-density 3D point clouds used for training CNNs to segment and label urban fields [27]. In the case of autonomous driving, CNNs effectively deal with visual data to identify and track the lanes on which the vehicle will operate, following the example of robots for line-following [28], also the study illustrated in the Figure 2.2 shows the training process of a neural network in the autonomous driving.

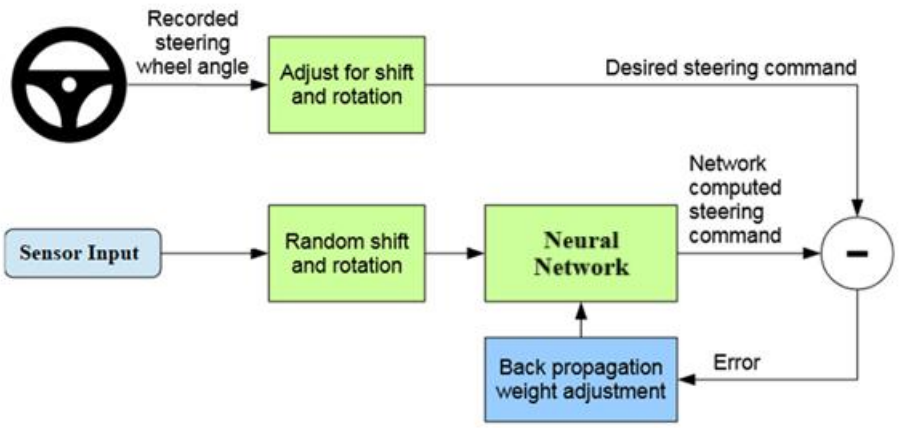

Figure 2.2 : Training a neural network architecture [29]

On the other hand, RNNs work excellently in the modeling of complex temporal sequences, thereby fitting Any prediction that classifies vehicles' states and actions over time [28][30]. So, through the combination of those networks, models capable of establishing end-to-end control systems with implementations that make sensory input to drive commands may be designed, using approaches like applying variational autoencoders for data bias adaptation and overall training improvement [31].

Autonomous vehicles thus achieve greater accuracies and high reliability through advanced deep learning technologies in dynamic driving environments.

2.1.2 Multilayer perceptron

The Multilayer perception (MLP), as shown in Figure 2.3, is a feedforward artificial neural network. MLP is employed in reinforcement learning problems with vector observations along with other algorithms in order to perform imitation and inverse reinforcement learning with the vector observations. The main training aim of the neural network would be to reduce the loss function. The learned policy and expert demonstration state-action pairs will be represented with some type of features distilled via gradient optimization by the MLP to separate between the learned state-action pairs by the policy and those of expert demonstrations. This discrimination outcome will allow us to update a policy so that it matches the behavior of the expert driver using the reinforcement learning algorithm. In research's experimental stage, the highway topology is implemented to analyze the performance of each neural network architecture or the significance of some critical elements to the algorithms' learning capability. In the study, MLP is employed to work on vectorized information from our environment observations which are kinematic observations of the vehicle dynamics in the environment.

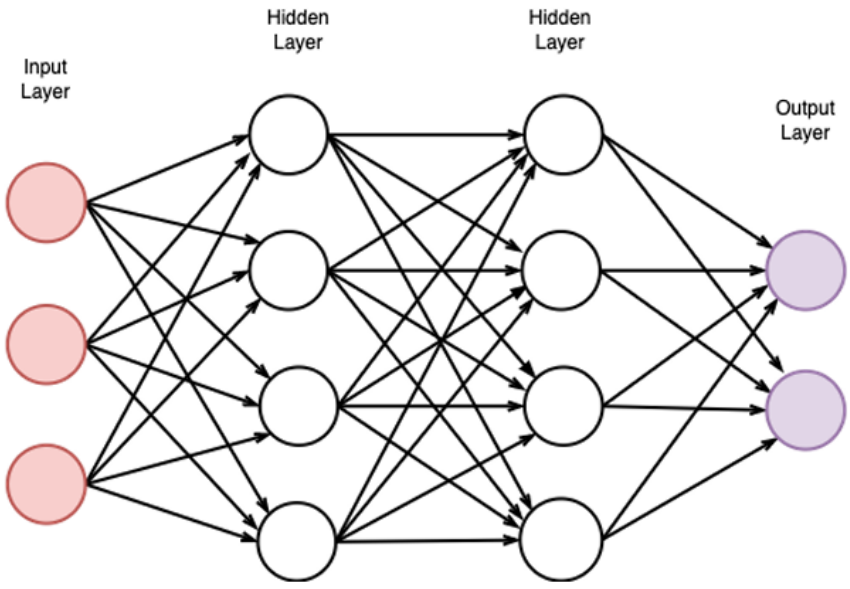

Figure 2.3 : A two-layered neural network [32]

2.1.3 Generative adversarial networks

GANs were introduced by [33], which include a generator and a discriminator competing with one another in a zero-sum structure. In this architecture, shown in Figure 2.4, the generator creates synthetic data or transitions from the environmental states used to confuse the discriminator with real data. The discriminator classifies the two inputs as either real or synthetic.

The generator aims to maximize its objective by feeding a noise vector or a few generated samples that can fool a discriminator. In so doing, the discriminator progressively increases its capability to discriminate real data from generated data. This leads to a system of competition: the generator develops its methods to outsmart the discriminator, while the discriminator hones its discrimination skills in response.

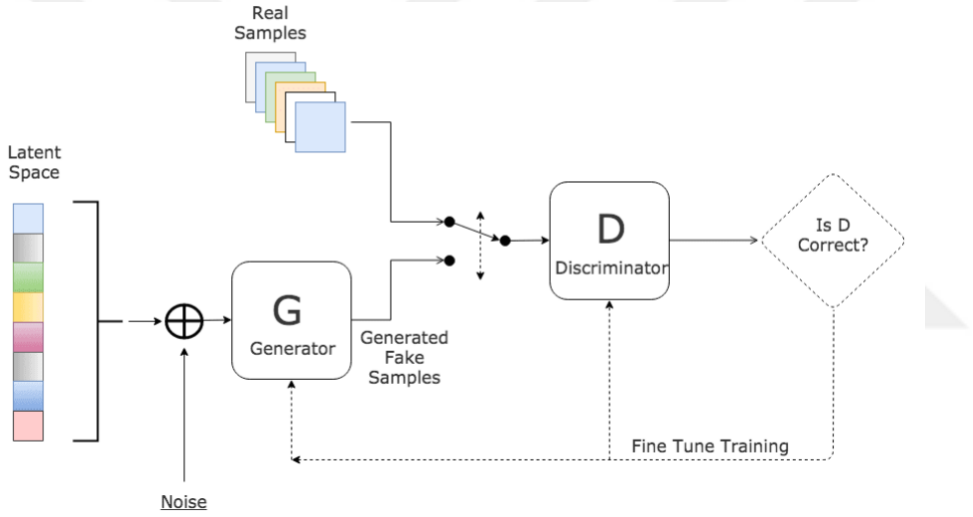

Figure 2.4 : Representation of generative adversarial network [34]

2.2 Reinforcement Learning

Reinforcement learning (RL) is a branch of artificial intelligence [35], which is one of the three approaches to machine learning (ML) with supervised and unsupervised learning. RL has turned out to be a very useful approach in the growth of the autonomous driving systems, especially in decision making and lane change decisions. This paper aims to investigate how reinforcement learning can be utilized to enhance the process of self-driving vehicles (AVs) with a focus on improving both safety and traffic flow in highway simulation environment. As the use of AVs becomes more widespread, they must operate in diverse and dynamically changing environments that involve other vehicles controlled by humans or other autonomous systems. RL offers

a way for AVs to learn how to drive optimally through a process of interaction from the environment, use the information to shape the rewards and feedback information as the reward. This approach helps AVs to learn to operate proactively to avoid hazards and enhance the quality of their journeys. For example, RL-based systems can model the aggressive driving patterns of the surrounding vehicles, which helps the AV to decide on the best action to avoid possible risks [36]. In addition, the RL methods have been used to increase the traffic flow and fuel economy by optimizing the lane changes and other traffic behaviors [37][38]. Thus, with the help of real-world data and sophisticated algorithms, RL helps to create AVs that are capable of performing many of the human-driven driving tasks, including lane changes, effectively and accurately [39][40]. Thus, the application of RL in the improvement of decision-making for AVs is likely to advance as research in this area progresses. The fundamental structure is shown in Figure 2.5. To understand the various methodologies, algorithms, strategies, and structures of reinforcement learning, it is first necessary to outline the fundamental concepts that underpin it. These concepts will be treated as essential knowledge leading up to all the topics to be discussed in this study, including offline, inverse reinforcement and imitation learning. Due to the fact that reinforcement learning is an extensive concept that touches upon many areas, the theoretical background will be presented in a simple manner.

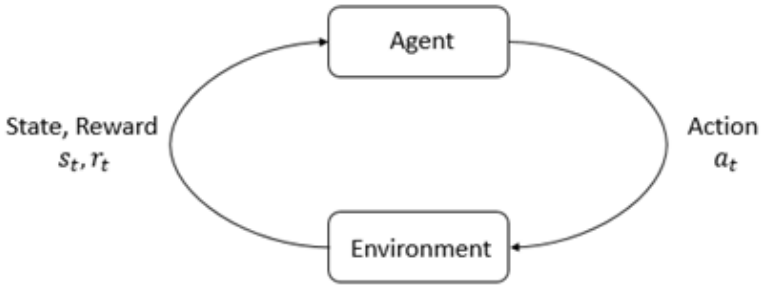

Figure 2.5 : Agent-environment interaction loop in reinforcement learning [41]

The Markov decision process, or MDP, is an important concept of reinforcement learning, and is split into two parts as the Markov chain and the Markov property. The Markov property states that the next state is completely determined by the current state, and not by the immediately past states. This property is utilized by the Markov chain, a stochastic model. The MDP is an extension of the Markov chain, which includes a

reward system and decision making. The reward mechanism in the reward function is to give a numerical value to the transitions in the chain, and the decision process is to choose actions in the context of the rewards and the state of the system according to the Markov property.

This sort of decision making based on states in the future is a form of learning that is wound to the idea of trial and error. While there are constraints, all of the strategies and methods of reinforcement learning are developed to increase the number of state-action pairs sampled for potential rewards in the future.

Dynamic programming, which was published by Bellman [42], is a technique for solving an optimization problem by solving it by breaking it down into a series of small sub-problems, solving the subproblems and then utilizing the solutions to solve the overall issue recursively. The sub issues and the large problem are connected through the optimal paths that meet the criteria of the Bellman's principle of optimality which is expressed through the Bellman optimality equation. DP reduces computational cost by not repeating calculations of subproblems that are used many times and instead using the Bellman equation to solve for them. However, there is a major problem with DP: We must know the dynamics of the environment to determine the optimal policy from them. This dependency makes DP infeasible to implement in scenarios where such knowledge is not easily attainable.

Two primary optimal policy identification DP strategies exist: These are value iteration and policy iteration as shown in Table 2.1.

Table 2.1 : Policy iteration versus value iteration [43]

Aspect	Value iteration	Policy iteration
Methodology	Keeps updating value functions until convergence	Responds between policy evaluation and improvement
Goal	Converges to optimal value function	Converges to the optimal policy
Execution	Directly computes value functions	Assess and enhance policies sequentially
Complexity	Easier to implement and discover	Involves more steps and computations
Convergence	May converge faster in many cases	May converge slower but yields better policies

Both value iteration and policy iteration are successful in solving reinforcement learning problems and finding the optimal policies. Value iteration is simple in implementation because it directly computes the optimal value function iteratively and, in special cases, may get quick convergence. Policy iteration shows smooth and slow convergence as it evaluates and improves the policy, but very often it discovers optimal policies.

The Bellman Equation is an important aspect of dynamic programming and reinforcement learning. MDPs or optimal policies are solved using a recursive manner by the Bellman Equation [44].

$$Val(s) = max_a.(r(s,a) + \gamma\, Val(s')) \tag{2.1}$$

The max represents the best action out of all the possible actions that the agent can select in a specific state that results in the maximum reward, and this is done for every consecutive step.

2.2.1 Policy based optimization method

Different optimization techniques in reinforcement learning attempt to improve learning algorithm performance, especially in complex decision-making tasks. Proximal policy optimization (PPO) is a method known for its effective performance and simplicity.

PPO, proposed by Schulman et al. [45] is an on-policy reinforcement learning algorithm that can be used for both discrete and continuous action spaces. Its purpose is to make important changes in policy performance without making critical changes that may lead to the instability or degradation of the policy. This is achieved through the use of simpler first-order optimization techniques which are computationally efficient but require other strategies to keep the updated policy near to the original.

The main implementations of the PPO algorithm are penalty PPO and clip PPO.

1. Penalty PPO uses a soft constraint on the divergence in the objective function to restrict the new policy from diverging too far from the old policy. The penalty coefficient varies dynamically during the training process to normalize the constraint.

2. Clip PPO, however, does not use constraints or explicit divergence penalties. Instead, it uses a clipping approach to regulate the policy changes. This clipping

technique is a regularizer and the parameter is a hyperparameter ϵ, which is the maximum distance of the new policy from the old policy.

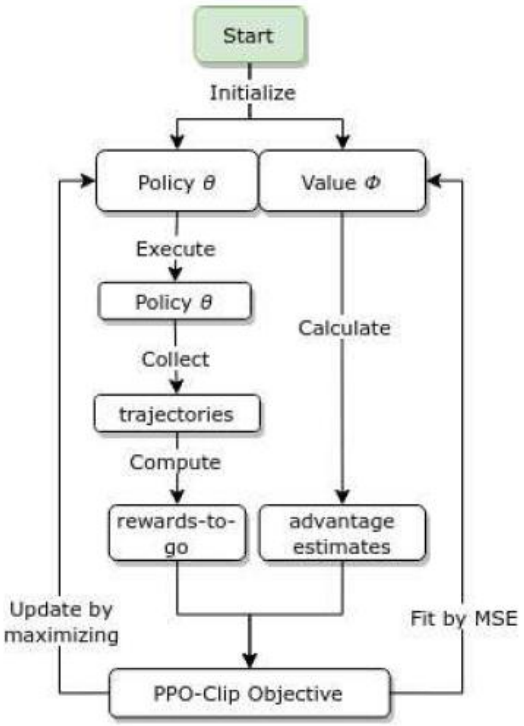

Figure 2.6 : PPO algorithm [46]

In this study, the focus is on the clip PPO variant, which does not require penalty terms but still maintains policy stability, as shown in Figure 2.6. Clip PPO employs minibatches of data and stochastic gradient descent to train the objective function. The clipping mechanism facilitates the robustness of updates to a policy that are restricted by some measure of advantage of a given state-action pair:

- If a state–action pair has a positive advantage, it indicates a preference for the action over alternative actions. As the probability of selecting this action increases, the objective function increases. However, if the increase is more than the clipping threshold, then the objective is minimized to avoid the policy drifting away from its previous version.
- On the other hand, if the advantage is negative, then the action is considered to be a worse action. The objective function is decreased to decrease the probability of choosing this action. If the decrease is more than the lower limit of the clip threshold, then the objective is maximized to avoid overfitting.

PPO's stochastic policy enables exploration by sampling actions according to the most recent policy distribution. However, as training progresses, the policy often becomes less stochastic, prioritizing exploitation of known rewards. This reduction in randomness can sometimes result in the policy converging to a local optimum, ceasing further learning.

Policy gradient approach methods work by estimating the policy gradient and using the estimate as an update into a stochastic gradient ascent [45]. The most common way the gradient is estimated is with the following structure:

$$\hat{g} = \hat{E}_t[\nabla_\theta \, log \, \pi_\theta(a_\theta \mid s_\theta) \, \hat{A}_t] \tag{2.2}$$

Here, the expectation $\hat{E}_t$ refers to the average observed across a limited set of sample returns gathered during a process that alternates between sampling and optimization. Implementations that leverage automatic differentiation tools operate by defining an objective function whose gradient provides the policy gradient estimator; in this scenario, the estimator $\hat{g}$ was determined by calculating the gradient of the objective function.

$$L^{PG}(\theta) = \hat{E}_t\big[log(a_t \mid s_t) \, \hat{A}_t\big] \tag{2.3}$$

PPO offers an easy implementation, a high sample efficiency, and applicability to a large variety of tasks. The following approach will be described for the clipped surrogate objective (2.5), as our focus is clip PPO in the study.

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}, r_t(\theta_{old}) = 1, \tag{2.4}$$

$$L^{CLIP}(\theta) = \hat{E}_t \left[min \left(r_t(\theta)\hat{A}_t, clip\big(r_t(\theta), 1 - \epsilon_{clip}, 1 + \epsilon_{clip}\big)\right) \hat{A}_t \right] \tag{2.5}$$

The value of ϵ_{clip} will be in hyperparameters section in 4.2.3. Additionally, $r_t(\theta)$ (2.4), is applied with the clipping function to prevent large deviations from the original θ_{old}. As it is noted, gradient descent is possible because the optimization problem has been defined via a differentiable loss function. This means we can now parameterize a critic with θ. PPO optimizes the critic through a value function loss. The loss for value function is in equation (2.6), and the final loss with adding entropy term can be found as in the equation (2.7):

$$L^{VF}(\theta) = (V_\theta(s_t) \; - \; \hat{R}_t)^2 \tag{2.6}$$

$$L^{CLIP+VF+S}(\theta) = \hat{E}_t[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \quad (2.7)$$

2.2.2 Inverse reinforcement learning

Inverse reinforcement learning (IRL) can particularly provide realistic driving behaviors of humans which is essential for self-driving vehicles to act safely in the traffic mixed with other vehicles. The focus of IRL techniques is to discover the reward functions that are inherent in the human decision making process so that autonomous systems can mimic the human like actions. This approach is useful for lane change policies since it allows the vehicle to come up with decisions that are expected from human drivers. Unlike the paradigm of IL, IRL [47] works on creating a reward function from expert demonstration. For example, from the study of real-world driving data, autonomous vehicles are able to learn lane change controls that are similar to those of a human driver, thus enhancing safety and acceptance [39]. The ability of the IRL to capture the subtleties of human driving makes it a suitable option, as it supports the aim of creating autonomous systems that can coexist seamlessly with human drivers. In that research, initial candidate trajectories are sampled, which will be selected based on the lowest cost as the anticipated trajectory. In a continuous setting, the reward functions are determined by using the speed profile sampler to calculate the partition function in [48]. In another study [49], state sequences are sampled from the policy and subsequently passed through an attention-based trajectory generator to produce valued future trajectories. To minimize computational expenses, Xin et al. proposed an energy-based generative model for the behavior of other vehicles and an inverse optimal control approach utilizing Langevin Sampling to learn the cost functions for these vehicles [50]. Furthermore, in [51], discrete latent driving intentions are employed with a polynomial trajectory sampler to derive reward functions grounded in the decision-making process. Basic form is shown in Figure 2.7.

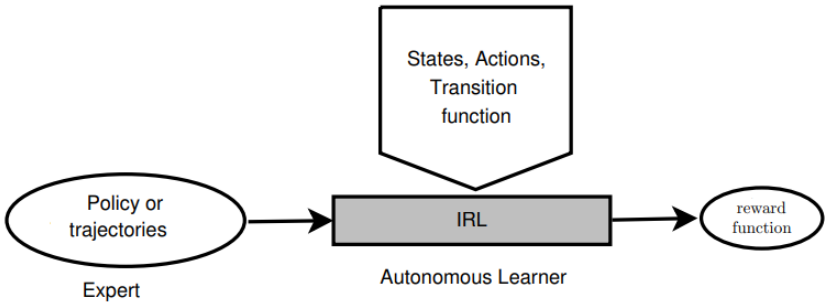

Figure 2.7 : Pipeline for classical inverse reinforcement learning [52]

2.2.3 Adversarial inverse reinforcement learning

Adversarial inverse reinforcement learning (AIRL) provides the decision making capability for self-driving vehicles in complicated environments, thereby contributing greatly to the development of self-driving cars. AIRL uses adversarial networks to improve the prediction of the behaviors of agents including other vehicles and pedestrians in shared environments. This approach is more accurate in modeling the interactions than the traditional single agent frameworks which cannot fully depict the dynamics of the real world environments [53]. In practical terms, AIRL can help autonomous vehicles to learn optimal control strategies from observing the behavior of other agents, including human drivers, in a traffic flow. This is done by using neural networks that can learn patterns and interactions, thus enabling the vehicles to make reasonable decisions even when they are operating under bounded rationality and partial information. Adversarial learning can be used to improve the adversarial training of autonomous vehicles to improve their prediction and response to other road users, thus improving safety and efficiency. The main architecture of AIRL is shown in Figure 2.8.

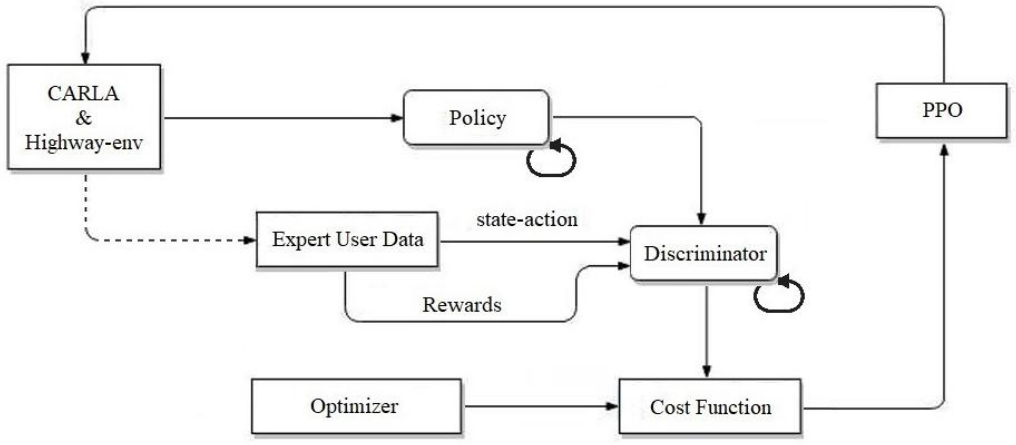

Figure 2.8 : Adversarial inverse reinforcement learning architecture

AIRL can be utilized to simulate the decision-making processes of skilled drivers in the context of autonomous driving [13]. By integrating AIRL into self-driving cars, it can lead to more robust and adaptable systems capable of maneuvering through the intricacies of real-world driving environments, ultimately improving the safety and dependability of autonomous transportation systems [36]. When implemented on

complicated scenarios in CARLA, the augmented AIRL outperformed baseline methods on various aspects and produced performance like that of expert drivers [13].

AIRL is directly related to guided cost learning (GCL) [54] and maximum entropy inverse reinforcement learning (MaxEnt IRL) [55]. It employs a unique form of the discriminator that differs from the one used in generative adversarial imitation learning (GAIL), allowing it to recover both a cost function and a policy simultaneously, similar to GCL, but through an adversarial approach. The key component of AIRL is the design of the discriminator.

Let's assume a behavior sequence represented as $\tau = (s_0,\ a_0, \ldots,\ s_T,\ a_T)$ derived from the demonstrated data, while $c_\theta(\tau) = \sum_t^T c_\theta(s_t,\ a_t)$ represents the unknown cost function parameterized by some variable. It is denoted $p(\tau)$ as the true distribution of the demonstrations and $q(\tau)$ as the density of the generator. According to the mathematical proof presented in [56], the discriminator can be formulated with the equation (2.8), wherein $p(\tau)$ is approximated using the distribution from maximum entropy IRL, and the probability of the state-action trajectories can be calculated approximately with utilizing the expert user dataset as a Boltzmann distribution (2.9).

$$D_\theta(\tau) = \frac{\frac{1}{z} exp\ (-c_\theta(\tau))}{\frac{1}{z} exp(-c_\theta(\tau)) + q(\tau)}, \tag{2.8}$$

$$p_\theta(\tau) = \frac{1}{z} exp\ (-c_\theta(\tau)) \tag{2.9}$$

The GCL presents an iterative, sample-driven approach for estimating Z within the MaxEnt IRL framework. It can effectively handle high-dimensional state-action datasets as well as nonlinear cost functions [54]. The algorithm derives an estimate for Z by developing a new sampling distribution $q(\tau)$ and employing importance sampling. In theory, the partition function Z represents the cumulative total of $exp(-c_\theta(\tau))$ across all potential trajectories, which poses a significant challenge for MaxEnt-IRL algorithms. The process of guided cost learning involves alternating between optimizing the cost function c_θ based on this estimate and optimizing $q(\tau)$ to reduce the variance of the importance of sampling estimate.

$$\begin{aligned} L_{cost}(\theta) &= E_{t\sim p}[-log p_\theta(\tau)] = E_{t\sim p}[c_\theta(\tau)] + log\ Z \\ &= E_{t\sim p}[c_\theta(\tau)] + log(E_{t\sim q}\ [\frac{exp(-c_\theta(\tau))}{q(\tau)}]) \end{aligned} \tag{2.10}$$

The optimal sampling distribution for estimating the partition function $\int \exp(-c_\theta(\tau))d\tau$ is $q(\tau) \propto |exp(-c_\theta(\tau))| = exp(-c_\theta(\tau))$. In the GCL, the sampling policy $q(\tau)$ is refined to conform to this distribution by minimizing the divergence between $q(\tau)$ and $\frac{1}{z} exp\,(-c_\theta(\tau))$. This process effectively involves minimizing the learned cost while maximizing entropy.

$$L_{samp}(q) = E_{t\sim q}[c_\theta(\tau)] + E_{t\sim q}\,[log\, q(\tau)] \tag{2.11}$$

To reduce the importance sampling variance for the moments if the sampling distribution q fails to cover some trajectories with high value of $exp(-c_\theta(\tau))$, a distribution $\mu = \frac{1}{2}p + \frac{1}{2}$q is utilized to be the mixture distribution over trajectory rollouts. Let $\tilde{p}(\tau)$ denote the rough approximation of the expert demonstration distribution. Consequently, GCL uses μ for the importance sampling with $\frac{1}{2}\tilde{p} + \frac{1}{2}$q as the importance weights and the loss function for the cost can be expressed as:

$$L_{cost}(\theta) = E_{t\sim p}[c_\theta(\tau)] + log(E_{t\sim \mu}\,[\frac{exp(-c_\theta(\tau))}{\frac{1}{2}\tilde{p}+\frac{1}{2}q}]) \tag{2.12}$$

For the GAN network, loss of discriminator and generator networks can be found in [33] as follows (2.13),(2.14) and from equation (2.8) and (2.9) $D_\theta(\tau)$ can be rewritten as (2.14):

$$L_D(D) = E_{t\sim p}[-log\, D_\theta(\tau)] + E_{t\sim q}\,[-log(1 - D_\theta(\tau))], \tag{2.13}$$

$$L_G(G) = E_{t\sim q}[-log\, D_\theta(\tau)] + E_{t\sim q}\,[log(1 - D_\theta(\tau))]\,, \tag{2.14}$$

$$D_\theta(\tau) = \frac{p_\theta(\tau)}{p_\theta(\tau) + q(\tau)} \tag{2.15}$$

The optimal solution for the discriminator does not rely on the generator, leading to enhanced training stability. Utilizing the entire trajectory for cost calculations can lead to significant variance [12]. Instead, we use a modified version of the discriminator that focuses on individual state-action pairs to address this problem. The discriminator can be reformulated as shown in (2.15), where $q(\tau)$ is replaced accordingly with the learnt stochastic policy $\pi(a|s)$ and for addressing individual state-action pairs $p_\theta(\tau)$ is replaced with $\exp(f_\theta(s,a))$.

$$D_\theta(s,a) = \frac{exp(f_\theta(s,a))}{exp(f_\theta(s,a)) + \pi(a|s)} \tag{2.16}$$

The approach can minimize the variance in the training practically. Also, the D_θ is trained to reduce the loss in the equation (2.13) with individual state-action pairs:

$$L_D(D) = E_{expert_\pi}[-log\, D_\theta(s,a)] + E_\pi\,[-log(1 - D_\theta(s,a))] \tag{2.17}$$

The reward function for policy training is defined in[12] as,

$$r(s,a) = log\, D_\theta(s,a) - log(1 - D_\theta(s,a)) \tag{2.18}$$

In [12], a method of learning disentangled reward is mentioned for AIRL algorithm. Where proposed of modification of the discriminator and reward function as,

$$D_{\theta,\emptyset}\,(s,a,s') = \frac{exp(f_{\theta,\emptyset}(s,a,s'))}{exp(f_{\theta,\emptyset}(s,a,s')) + \pi(a|s)}, \tag{2.19}$$

$$r(s,a,s') = log\, D_{\theta,\emptyset}(s,a,s') - log(1 - D_{\theta,\emptyset}\,(s,a,s')) \tag{2.20}$$

2.3 Imitation Learning

Imitation learning (IL) has emerged as an approach in machine learning where an agent replicates the action of an expert user (human or AI) by learning from observations, as well as actions performed during a specific task [57]. IL is not directly trained on a reward function or policy space, rather it uses expert demonstrations, which are data that maps observations to actions, to learn the desired behavior. The agent mimics these expert trajectories that are generated from the optimal policy to learn what state action pairs are good. IL is based on different forms of data analysis, from traditional supervised learning to adversarial networks. The fundamental structure is shown in Figure 2.9.

Imitation learning for supervised learning is a context since it aims to minimize a loss function. Different strategies are employed by the algorithms that fall under the IL cover to accomplish this. For example, DAgger (Dataset Aggregation) by Ross et al. [58] increases the size of the dataset by training the policy on all the expert data collected in each iteration to close the gap to the optimal policy. GAIL, introduced by Ho and Ermon [10], enables learning a policy directly from expert behavior without the need for inverse reinforcement learning (IRL) or reinforcement learning (RL). It achieves superior performance compared to existing methods, especially in complex, high-dimensional environments. deep Q-learning from demonstrations (DQfD) introduced by Hester et al. [59] includes expert user data into the replay buffer to serve

as a pre-training phase before engaging in learning from the environment. There are also other IL strategies that focus on specific topics in this area [60][61][62].

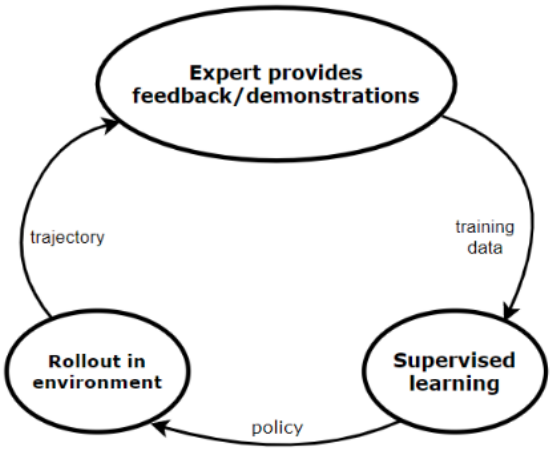

Figure 2.9 : Imitation learning process

2.3.1 Generative adversarial imitation learning

Generative adversarial imitation learning (GAIL) employs a generative adversarial network (GAN) to infer the cost function by differentiating between expert trajectories and those produced by the learned policy. A key concept in GAIL is the alignment of occupation measures, which illustrate the distribution of state-action pairs encountered by the agent while following a designated policy.

As discussed in Section 2.2.2, the generator network in a GAN seeks to create data that the discriminator network cannot distinguish between. In GAIL, the generator is trained to create trajectories that are similar to expert trajectories. The discriminator learns to differentiate between the expert trajectory distribution and the generated trajectory distribution by agent. If the discriminator is unable to tell the difference, the generator has successfully matched the dataset. The main architecture of GAIL is shown in Figure 2.10.

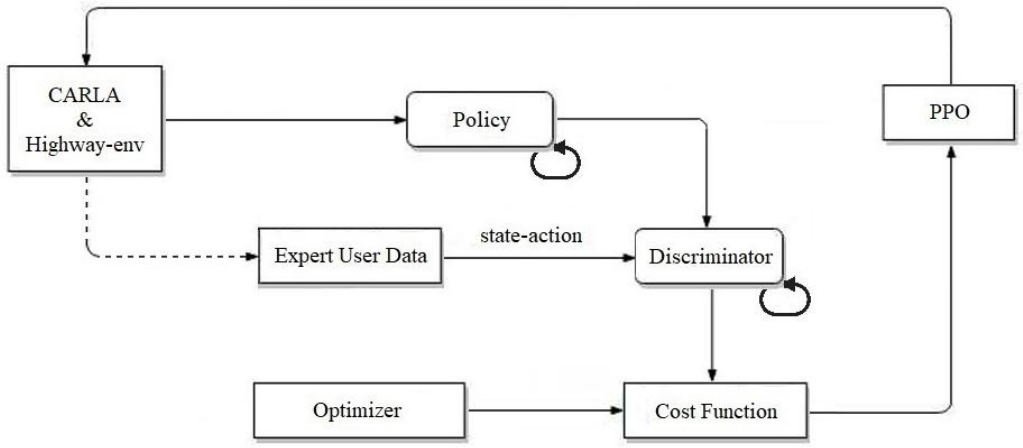

Figure 2.10 : Generative adversarial imitation learning architecture

In this scenario, the occupancy measure of the expert agent aligns with the actual data distribution, whereas the occupancy measure of the imitation learner aligns with the distribution of generated data. GAIL optimizes the generator to synchronize these distributions by minimizing their difference.

The optimization process in GAIL involves alternating between two steps:

• Optimization: The weights of the discriminator are updated through Adam stochastic gradient descent to enhance its capability to distinguish between expert trajectories and generated ones.

• Policy update: Stochastic gradient descent is utilized to adjust the policy, aligning it with the expert trajectories while ensuring stability.

Although GAIL is efficient in the amount of expert data it needs, it is resource-intensive regarding the interactions with the environment during training. The quantity of samples required to assess the gradient for the imitation objective is similar to the samples needed for training a reinforcement learning agent's policy directly.

In this thesis, one of the imitation learning algorithm method is utilized which is generative adversarial imitation learning. This algorithm was chosen for two main reasons:

- It is often employed as a reference implementation in imitation learning studies.
- It has been known to excel at replicating sophisticated behaviors in states spaces of high dimensionality.

Instead of deriving the reward function from expert demonstrations through IRL, GAIL derives policies directly from data. Similar to a GAN, the core concept of GAIL is straightforward; the generator produces a trajectory that closely resembles the expert trajectory as much as possible, while the discriminator attempts to identify whether a given trajectory is from an expert. When performing IL with RL and adversarial training, this method is known as closed-loop training [63][64]. Many articles apply GAIL for trajectory prediction in autonomous vehicles (AVs). In [65], Kuefler et al. enhance GAIL by optimizing RNNs to model human driving behavior, where the discriminator assesses the policies and actions. In [66], Li et al. attempt to use the information maximization theorem to uncover the underlying structure of expert demonstrations. In [67], the authors introduce a parameter sharing extension of GAIL to capture the interactions between multiple agents and provide them with domain-specific knowledge.

GAIL comes from a MaxEntIRL objective with cost regularization (2.21) [68]:

$$IRL_{\psi}(\pi_E) = arg\,max - \psi(c) + RL(c) - E_{\pi_E}[c(s,a)] \tag{2.21}$$

Where $RL(c) = \arg min - H(\pi) + E_{\pi}[c(s,a)]$, the causal entropy of the policy π is $H(\pi) = E_{\pi}[-\log \pi(a|s)]$ and convex reward function regularizer is $\psi(c)$ [10]. $H(\pi)$ represents the discounted causal entropy of a policy in relation to the distribution of state-action pairs generated by that policy, while ψ is a function that maps each value of the cost function c to extended real values. The regularization function ψ is critical for giving rise to GAIL and understanding its relationship with approaches from apprenticeship learning. In particular, [10] describes the results of RL run off of a MaxEntIRL cost output (2.22):

$$RL\ o\ IRL_{\psi}(\pi_E) = arg\,min_{\pi \epsilon \Pi} -\mathrm{H}(\pi) + \psi^{*}(\rho_{\pi} - \rho_{\pi_E}) \tag{2.22}$$

The interaction between RL and IRL is interesting to discuss. In RL, it is desired to discover the best approach π that maximizes reward, whereas in IRL, it is wanted to discover the best reward function, which maximizes the divergence among expert policy π_E and RL-derived policy π. Theoretically, RL can be considered a type of generator that generates samples from some reward function, while IRL represents a discriminator that distinguishes an expert policy from an RL policy. This interaction functions in a manner akin to the context of GANs, where a generator (G) produces

data, while a discriminator model (D) assesses whether a sample originates from the actual data or the generator. The generator and discriminator engage in a minimax game aiming for equivalent optimal results in which the samples from the generator are indistinguishable from the real data. The GAIL algorithm is presented with an appropriate choice of regularizer ψ(r) within the IRL framework in equation (2.21). A minimax game is established between the discriminator (D) and the policy (π) as illustrated in equation (2.23).

$$min_D \max_\pi \left(E_\pi[\log D(s,a)] + \left(E_{\pi_E}\left[\log\left(1 - D(s,a)\right)\right] - \lambda H(\pi)\right)\right) \quad (2.23)$$

To find a saddle point (π,D) that satisfies Equation (2.23), function approximations are utilized for both π and D, as they are unknown functions of time, making it very challenging, if not impossible, to define their precise forms. As we calculate the gradients of the objective function with respect to the associated parameters of π and D, both the generator and discriminator can be trained through backpropagation. In practice, gradient updates for π and D are typically conducted alternately until both networks reach convergence. The approach for updating the ω-parameterized discriminator to minimize the discriminator objective and for modifying the parameters of the policy generator to maximize the policy objective is presented [10]:

$$E_{(s,a)_\pi}[\nabla_\omega \, log(D_\omega(s,a))] + E_{(s,a)_{\pi_E}}[\nabla_\omega \, log(1 - D_\omega(s,a))] \quad (2.24)$$

$$E_{(s,a)_\pi}[\nabla_\theta \, log \, \pi_\theta(a|s) Q(s,a) - \lambda \nabla_\theta H(\pi_\theta)] \quad (2.25)$$

Where $Q(\bar{s},\bar{a}) = E_{(s,a)_\pi}\left[\log(D_{\omega_\pi}(s,a)) \mid s_0 = s, a_0 = a\right]$. The generator is designed to optimize the loss value of the discriminator. Given that the second term of equation (2.24) is independent of the generator parameters, the goal of the generator is to enhance the first term of equation (2.24). Therefore, the surrogate reward function can be expressed as follows:

$$r(s,a) = -log(D_\omega(s,a)) \quad (2.26)$$

3. ENVIRONMENT

3.1 Simulation Platform

Carla: The application of high-fidelity simulation environments, which provides a safe and controlled platform for testing and validation, has given a big boost to the enhancement of self-driving systems. CARLA is one of the most utilized simulation environments grounded in research and open-source software for evaluating autonomous driving. With its compatibility with Python, CARLA offers a rich feature set for developing, training, and testing of autonomous vehicles which provides testing of a large number of various scenarios and problems.

The simulation system of CARLA is built on Unreal Engine 4 that allows realistic rendering and physically based simulation. The virtual environment consists of multiple factors involving city, residential, and urban roads, traffic signs, traffic lights, roundabouts, and pedestrian crossings. These elements are complemented with other dynamic actors. Vehicles, pedestrians, and cyclists as dynamic actors can be implemented to face various behaviors in a more realistic manner. CARLA supports the customization in environmental conditions, including daytime, weather conditions, and the frictional properties of the road surfaces. It provides an extremely flexible test area for autonomous systems.

One notable aspect of CARLA is its support for predefined and customizable scenarios. Predefined scenarios can be simple lane-following tasks or active urban traffic scenarios with multiple actors and interactions. Researchers can implement their own scenarios using the ScenarioRunner tool, which allows them a detailed specification of the actor's behavior, triggers, and events. This flexibility makes CARLA useful to simulate rare or dangerous instances that are of critical concern for robust system development.

The sensor options of the simulation environment in CARLA are fully equipped to help develop and test perception algorithms. These are RGB cameras, depth cameras, semantic segmentation cameras, LiDAR, Radar, and GNSS sensors. The sensor data

is created with high realism and can be made to resemble real-life outputs, which can be used for object detection, semantic segmentation, and localization tasks. This ability to use sensors on and customize them for simulated vehicles makes CARLA a useful tool to address many research questions in a single environment.

In CARLA, data collection and training are directly streamlined through its Python API, which provides tools for controlling the simulation, managing actors, and retrieving sensor data output. This allows the researcher to create a dataset of sensor data together with its labels to train machine learning models. It also has the capability for reinforcement learning since it can provide a controlled environment for agents to learn and adapt their behavior through interaction with the environment. All this makes it possible to train autonomous systems for a variety of tasks, from perception to decision making and control.

In this thesis, CARLA version 0.9.13 with Python 3.7 is used for required a robust and flexible platform that allows development for scientific research in autonomous driving. Carla's realistic simulation environment, low cost usage, accurate physics model, customizable scenarios, and allowance of kinematic sensors to collect data and permit intensive testing and training of autonomous systems specifications were the key points of selection. By complementing real-world testing with simulation, CARLA accelerates the improvement of safe and robust self-driving vehicles.

Highway-env: In the enhancement of self-driving systems, it is often necessary to use simulation environments as a cost-effective way to train, test, and validate the systems. One such environment is Highway-env; an open source, Python based, simulation environment for self-driving research, with a particular focus on highway and multi-lane road cases. Highway-env is famous for its simplicity, adaptability and compatibility with reinforcement learning, imitation learning and inverse reinforcement learning frameworks, and is used by researchers and practitioners.

Highway-env has several predefined scenarios that are meant to capture typical highway conditions and scenarios like lane following, over taking and merging. These scenarios are used as reference points for assessing and comparing the effectiveness of the autonomous driving systems. In addition, the user can define his/her own scenarios by setting certain parameters like the number of lanes, traffic density and initial position of the vehicles. This flexibility helps researchers to develop particular

experiments in order to answer certain research questions or to investigate particular rare situations that are not easy to reproduce in real world tests.

Highway-env does not model the physical sensors themselves; however, it does represent the state of the environment in an abstract state, as seen by an autonomous vehicle. These representations comprise of information such as position, velocity and lane of the vehicles which can be employed for perception and control. It provides to control the autonomous system with kinematic sensors and approaches. It is also possible for researchers to define their own observation functions to model sensor noise or limited sensing, which can increase the realism of the experiment. The environment's state and action spaces are highly configurable, thus making it suitable for use with a number of machine learning approaches including reinforcement learning.

In this thesis, the Highway-env environment was initially trained and validated for simple tasks and to generate various driving scenarios. This environment mimics highway conditions and helps in the analysis of vehicle dynamics and interactions. The capability of the developed models to generalize and their performance was evaluated by testing under different traffic conditions and driving scenarios. Highway-env is a versatile and lightweight simulation environment for studying autonomous driving on highways. It is customizable, integrated with inverse reinforcement learning and imitation learning frameworks, and suitable for evaluating and learning decision-making strategies. Combined with CARLA testing, Highway-env quickly develops the field of autonomous driving systems and enhances safety and efficiency.

The CARLA and Highway-env environments can be integrated to form a comprehensive framework for collecting data, conducting experiments, and verifying solutions in the field of self-driving simulations. CARLA is a realistic simulation tool that can generate diverse and realistic traffic scenarios and evaluate safety in traffic conditions with self-driving vehicles and human controlled vehicles. It is capable of simulating critical safety scenarios that are necessary for assessing collision risks and for improving the performance of autonomous driving systems [69]. Highway-env, however, is a relatively simplified model that is useful for identifying and improving autonomous driving policies, especially in highway environments. CARLA's detailed scenario modelling, autonomous driving validation capabilities were combined with Highway-env's policy testing and driving maneuver generating capabilities to create a

through validation of autonomous driving algorithms. This integration assists in the creation of more realistic autonomous driving behaviors through extensive scenario testing and policy refinement [70].

3.2 Autonomous Vehicle Concept

Autonomous vehicles (AVs) are a new form of transportation that could fundamentally change urban life and travel behaviors. These vehicles function independently without direct physical control or oversight from a human driver and provide various degrees of automation and vehicle utilization [71]. Study [72] point out that the language used to describe these self-driving vehicles, facilitates the shaping of public perception and acceptance of such vehicles since different terms can generate different degrees of public concern and uncertainty.

Autonomous driving technology is evolving very rapidly and has significant growth potential in global markets. Key players in this field, such as Google and GM, take advantage of their locational advantages to develop industrial chains and knowledge sharing, with a focus on integrating communication systems and artificial intelligence for vehicle autonomy [73]. The vehicle-to-everything communication technology also enhances the capabilities of AVs by improving safety and traffic flow through the use of advanced perception, planning, and control [74].

As more AVs come into the market, it is crucial to get the relationship among drivers and semi-autonomous vehicles for the purpose of safety and consumer acceptance. Research has pointed out that it is crucial to develop systems that encourage the right kind of relationship between the driver and the vehicle [75]. The idea of connected autonomy merges the principles of connected and self-driving vehicles, presenting a potential for enhanced safety, environmental sustainability, and effectiveness in mobility solutions [76].

3.2.1 Vehicles

3.2.1.1 Human controlled vehicle

The human controlled vehicle, shown in Figure 3.1, is the vehicle managed by the human to collect expert data for further learning processes. It is denoted as a yellow vehicle in the graphical representations.

Figure 3.1 : Human controlled vehicle representation: a) CARLA simulator, b) Highway-env

3.2.1.2 Ego vehicle

The ego vehicle, illustrated in Figure 3.2, refers to the car that the AI agent manages through one of the learning methods discussed in the thesis. It is also represented by a green vehicle in the graphical representations.

Figure 3.2 : Ego vehicle representation: a) CARLA simulator, b) Highway-env

3.2.1.3 Non-ego vehicle

A non-ego vehicle, shown in Figure 3.3, is simulated along with the ego vehicle but operates independently, controlled by low level controllers. It is denoted as a blue vehicle in the graphical representations.

Figure 3.3 : Non-ego vehicle representation: a) CARLA simulator, b) Highway-env

Non ego vehicles have two main characteristics:

1. Longitudinal behavior: The Intelligent Driver Model (IDM) [77] determines the acceleration with respect to the leading vehicle's distance and speed. As for the longitudinal controller, the following aspects were identified:

- Current/target velocities and throttle adjustment
- Proportional gain for speed control

2. Lateral behavior: The Minimizing Overall Braking Induced by Lane Change (MOBIL) model [78] models lane changes according to traffic situations and penalizes dangerous maneuvers and encourages safe changes. The lateral controller is responsible for:

- The position control leverages the lateral distance from the lane's centerline, utilizes a control gain, and incorporates heading adjustments to regulate lateral velocity.
- Heading control takes into account elements such as yaw rate, front wheel steering angle, and the desired heading to ensure the vehicle aligns with the lane's direction.

In addition, an advanced non-ego vehicle model known as the Markov decision process (MDP) vehicle was employed. This vehicle operates within a discrete speed range of 20-30 m/s and features a prediction module that forecasts future trajectories by analyzing sequences of actions, the durations of those actions, the timing of vehicle state samples, and the simulation timestep.

3.2.2 Control

In order to control the ego vehicle, the agent takes a number of actions which can be both continuous and discrete. Continuous control of vehicle dynamics is by adjusting throttle and steering angle. However, discrete actions are used as controllers on top of the continuous action system. This enables the vehicle to move at a constant velocity on straight tracks and issue high level orders to change its lane or speed. Hence, the ego vehicle travels at a certain speed and takes certain actions, like steering, without needing to control throttle and speed at the same time. In a discrete action space, there are five primary meta-actions available:

- Steer left
- Steer right
- Idle
- Accelerate
- Decelerate

3.3 Sensors and Observations

There is a kinematic observation that captures the aspect of vehicle telemetry. It works as a virtual sensor module, which detects changes in the environment or events and reports that information to the controlling agent of the ego vehicle.

The kinematic observations establish a vectorized representation of the kinematics (speed and velocity) for nearby non-ego vehicles in relation to the ego vehicle. The features include the x-coordinate, the y-coordinate, the velocity in the x-direction v_x, and the velocity in the y-direction v_y for all vehicles. In standard process, the ego car observes nearby vehicles in front and behind on its lane, as well as on adjacent left and right lanes. However, some of these vehicles may not exist (e.g., no car on the leftmost and rightmost lane) or may be too far to consider. In such cases, the vehicle is marked as “not presented,” represented by a 0, and it can be ignored. Also, this presence indicator (0 or 1) is included as a state feature along with the other kinematics and provides local observation.

Kinematic observations come with various configurable options, such as the choice between using absolute or relative coordinates concerning the ego vehicle, sorting or

randomizing the order of the observed vehicles, whether to normalize the data, if vehicles trailing the ego vehicle should be included in the observations, if the method should take note of the destinations of non-ego vehicles, and the number of vehicles to monitor.

Vehicle	x	y	vx	vy
ego	5	5	25	0
non-ego-1	-5	5	22.5	0
non-ego-2	13	11	25	0
non-ego-3	17	8.5	27.5	5
...	...	...	...	...

Figure 3.4 : Absolute kinematic feature map for ego and nearby vehicles

In the example illustrated in Figure 3.4, a kinematic feature map for the vehicles is demonstrated. When assessing non-ego vehicles relative to the ego vehicle:

- The ego vehicle is moving straight at a speed of v_x, = 25 m/s and v_y = 0 with coordinates (5, 5).
- Non-ego vehicle 1 is moving straight at a speed of v_x, = 22.5 m/s and v_y, = 0 m/s with coordinates (-5, 5) and it's seen behind the ego vehicle.
- Non-ego vehicle 2 is moving straight at a speed of v_x, = 25 m/s and v_y, = 0 m/s with coordinates (13, 11) and it's seen in front of the ego vehicle.
- Non-ego vehicle 3 is crossing the lane at a speed of v_x = 27.5 m/s and v_y = 5 m/s with coordinates (17, 8.5).

Vehicle	x	y	vx	vy
ego	0.03	0.03	0.83	0.00
non-ego-1	-0.03	0.03	0.75	0.00
non-ego-2	0.09	0.07	0.83	0.00
non-ego-3	0.11	0.06	0.92	0.17
...	...	...	...	...

Figure 3.5 : Normalized kinematic feature map for ego and nearby vehicles

The observations are normalized as illustrated in Figure 3.5 and scaled to a fixed range of (150, 150, 30, 30) for (x, y, v_x, v_y). In the kinematic feature map that is relative-normalized, as shown in Figure 3.6, the values are calculated in relation to the ego vehicle. Non-ego vehicle 1 is in the same lane but positioned behind and is moving at a slower speed; non-ego vehicle 2 is ahead in an alternate lane, traveling at the same

speed; and non-ego vehicle 3 is also ahead but shifting lanes and moving at a higher speed.

Vehicle	x	y	vx	vy
ego	0.00	0.00	0.00	0.00
non-ego-1	-0.07	0.00	-0.08	0.00
non-ego-2	0.05	0.04	0.00	0.00
non-ego-3	0.08	0.02	0.08	0.17
...	...	...	...	...

Figure 3.6 : Relative-normalized kinematic feature map for ego and nearby vehicles

In this thesis, unless otherwise stated, the default parameters are used for this thesis. This means the kinematic sensor:

- Include vehicles situated behind the ego vehicle if they are near.
- Monitors the status of the non-ego vehicles in relation to the ego vehicle.
- Defines the observation window to 7 vehicles.
- The data is normalized.

3.4 Rewards

In the minimalist environment, the rewards are intentionally kept simpler than the complexity of real world autonomous driving scenarios. The reward function is for basic driving behaviors and focuses on collision avoidance and efficient navigation. These rewards are used to reward agents controlling autonomous vehicles to come up with good driving strategies. The main objective is to use the agent to drive as fast as possible on the road without crashing. Each simulation has particular rewards related to its particular navigation and risk assessment tasks.

3.5 Simulations

A simulation represents a customizable scenario in the environment, with changeable configurations of the world, defined observation, and action spaces, as well as adjustable action execution frequencies and the simulation itself. Each simulation includes a road network that outlines the geometric characteristics of the roads and lanes. This road network is filled with vehicles; it contains one controllable ego vehicle that can alter its lane and speed, along with multiple non-ego vehicles whose actions

are determined by the simulation. Since the action space is the same for all simulations, the reward functions and observation spaces are adaptable, which enables more flexible experimentation.

3.5.1 Highway

The highway simulation consists of a four-lane and one-way traffic road, as illustrated in Figure 3.7. The ego vehicle's goals are:

- Operate at maximum speed.
- Prevent collisions with other vehicles.
- Remain in the safest lane whenever possible.

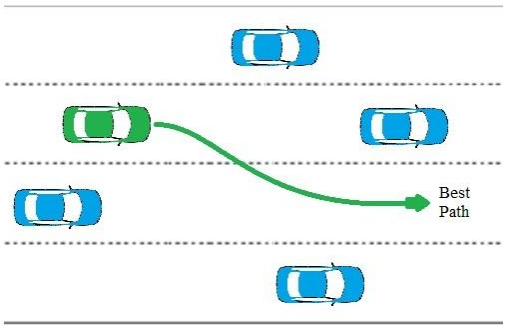

Figure 3.7 : Ego vehicle's path planning demonstration in the highway scenario

As a default, the simulation is conducted with a separate action space and kinematic observations. The basic configuration settings are:

- Traffic density: 20 vehicles generated
- Ego vehicle initial position: Random
- Episode duration: 30 seconds
- Speed range for rewards: 20–30 m/s
- Non-ego vehicle speed: Set uniformly to 20-30 m/s.

The following gains are achieved through specific actions for reward:

- Staying in the rightmost lane: +0.15
- Maintaining high speed: +0.5
- Lane changes: 0
- Collisions: -1 (penalty)

4. RESULTS AND DISCUSSIONS

This section will explore the establishment of an adversarial learning challenge to allow an autonomous vehicle to learn lane change decision making on highways. The primary objective here is to demonstrate how adversarial learning agents can navigate a realistic highway scenario by evaluating the design decisions made using vehicle and network models within the CARLA and Highway-env environments.

4.1 Simulation Setup

4.1.1 Setup

The implementation is executed using Python 3.7 alongside Highway-env and CARLA version 0.9.13 for the simulation; TensorFlow is utilized for the model training process. The simulations and training were conducted on a machine equipped with a single Nvidia RTX 3070 featuring 8 GB of video memory, a 4-core processor, and 32 GB of RAM.

4.1.2 Model architecture and algorithms

We selected proximal policy optimization (PPO) as the method for learning in our control problem experiments. PPO is a reinforcement learning technique that relies on policy gradients. In the section 2.2.1, we detailed the theory of the PPO, and it works by optimizing the current policy π_θ to be similar to the previous policy $\pi_{\theta_{old}}$. To fit this into the implementation, we will execute our optimization step in the highway environment. Also, to simplify the equation (2.5) in the section 2.2.1, the method [79] is utilized.

$$L(s, a, \theta_k, \theta) = min\left(r_t(\theta)\hat{A}_t, g\left(\epsilon_{clip}, \hat{A}_t\right)\right) \tag{4.1}$$

$$g\left(\epsilon_{clip}, \hat{A}_t\right) = \left(1 + \epsilon_{clip}\right)\hat{A}_t, while\ \hat{A}_t \geq 0 \tag{4.2}$$

In the equation (4.2), the advantage becomes positive for state-action pair. If $\pi_\theta(a|s)$ increases, the objective will rise since the advantage is positive. However, the

minimum term caps the extent to which this goal can rise. Once $\pi_\theta(a|s) > (1 + \epsilon_{clip})\pi_{\theta_k}(a|s)$ enters the min term, this term reaches a ceiling of $(1 + \epsilon_{clip})\,\hat{A}_t$. So, the new policy is quite similar to the previous one.

$$g(\epsilon_{clip}, \hat{A}_t) = (1 - \epsilon_{clip})\hat{A}_t, while\ \hat{A}_t < 0 \tag{4.3}$$

In the equation (4.3), the advantage becomes negative for state-action pair. Since the advantage is negative, the objective will rise if the action is less likely to occur, or if $\pi_\theta(a|s)$ decreases. However, the maximum in this term constrains the extent to which the advantage can rise. The maximum impacts at $\pi_\theta(a|s) < (1 - \epsilon_{clip})\pi_{\theta_k}(a|s)$, and this term is bounded by $(1 - \epsilon_{clip})\,\hat{A}_t$. So again, the new policy is quite similar to the previous one.

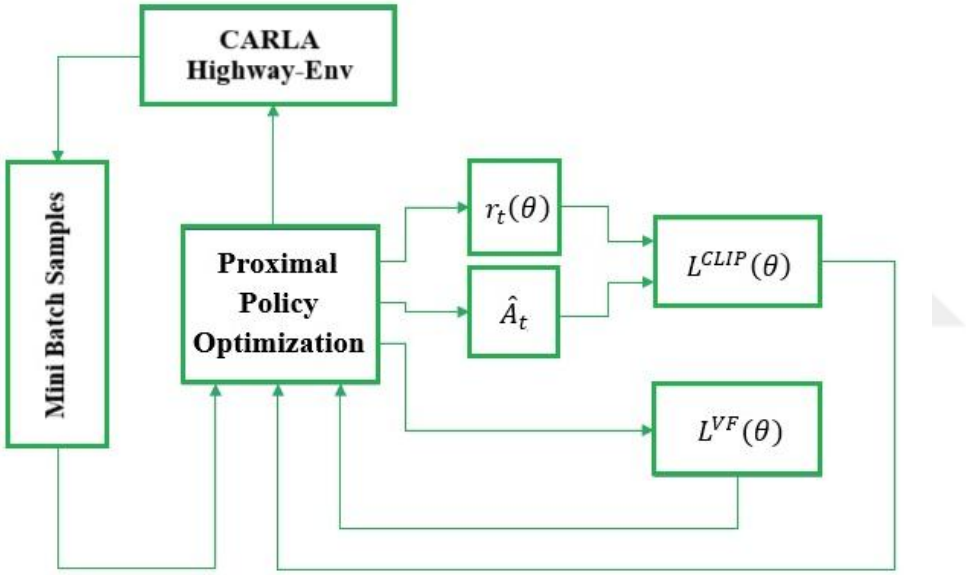

Figure 4.1 : PPO algorithm

The algorithm used within PPO clip, shown in Figure 4.1, utilizes the Adam optimizer, which allows us to efficiently update policy parameters θ while performing gradient ascent on the PPO objective function. Adam contributes to stable learning by dynamically modifying the step size for every parameter, utilizing first and second moment estimates of the gradient to enhance convergence speed and avoid instability. In the PPO framework, stochastic gradient descent plays a crucial role, as it guarantees that the newly suggested policy does not deviate significantly from the previous policy. By measuring Adam optimizer, we can either adaptively adjust the learning rate or implement a stopping criterion if the Adam optimizer exceeds a threshold. This further

ensures stable policy updates. The overview of the PPO clip algorithm steps is presented in Figure 4.2.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Figure 4.2 : PPO pseudocode

At the beginning of the study, two generative model architectures were considered: variational autoencoders (VAE) and generative adversarial network (GAN) generators. VAEs are known for their ability to learn structured latent spaces and provide a probabilistic interpretation of the data, which can be beneficial in tasks involving uncertainty or reconstruction. However, their reliance on reconstruction loss and divergence often leads to overly smooth outputs and increased training complexity. In contrast, GAN generators are designed to directly learn the data distribution through adversarial feedback, which generally leads to sharper and more realistic outputs, especially in behavior generation tasks. Considering the nature of the problem where the goal is to learn a reward function or policy that closely matches expert behavior, the adversarial generator architecture offered a better fit. It allowed the model to focus on capturing decision-level patterns without the overhead of reconstruction, making it a more efficient and task-aligned choice than VAE in this context.

For the policy network, an actor (policy) $\pi_\theta(a|s)$ and a critic (value) $V_\emptyset(s_t)$ networks are utilized. It is done by two separate MLP where a representation of actor-critic

architecture illustrated in Figure 4.3. We use MLP in this study as the input, s_t, is a vector with a size of 35.

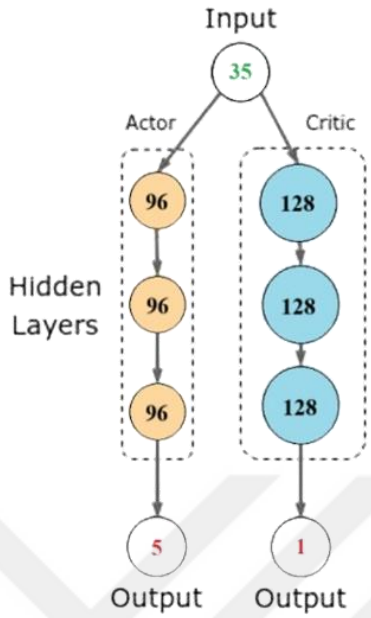

Figure 4.3 : Representation of the MLP for policy architecture

The policy MLP, shown in Figure 4.3, is formed with four layers with sizes 96, 96, 96, and act_{dim}, where act_{dim} represents the size of the action space, specifically 5. The ReLU activation function is used for all hidden layers, while the output layer makes use of Softmax. Consequently, it is adjusted that the output of the MLP fits the range of each action's corresponding range.

The value part, shown in Figure 4.3, is a MLP consisting of four layers with sizes 128, 128, 128, and 1, where the output gives $V_{\emptyset}(s_t) \approx r_t(\theta)$. We apply ReLU for the first three layers and no activation for the final layer. The absence of activation in the last layer allows the critic to represent any possible value of $r_t(\theta)$.

To optimize the network, we utilize the action means and standard deviations from the network, then determine the log probability $\log \pi\theta(a|s)$ for any action a under policy π given state s. It is mentioned that we need to calculate $r_t(\theta)$ to compute the clipped loss, L^{CLIP}. This can be done by combining equation (2.4) and the logarithm quotient rule:

$$\log \pi_\theta(a_t|s_t) - \log \pi_{\theta_{old}}(a_t|s_t) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} = r_t(\theta) \tag{4.4}$$

Therefore, we calculate the final loss, $L^{CLIP+VF+S}$, and utilize the Adam optimizer to optimize it.

For the discriminator network, demonstrations of the model architecture are presented in Figure 4.4. With four-layered MLP which has input channel with size 40 and hidden layers with size 64. The ReLU activation function is applied in every layer except for the output layer. The final layer consists of a single unit that utilizes a sigmoid activation function, representing the probability of the action.

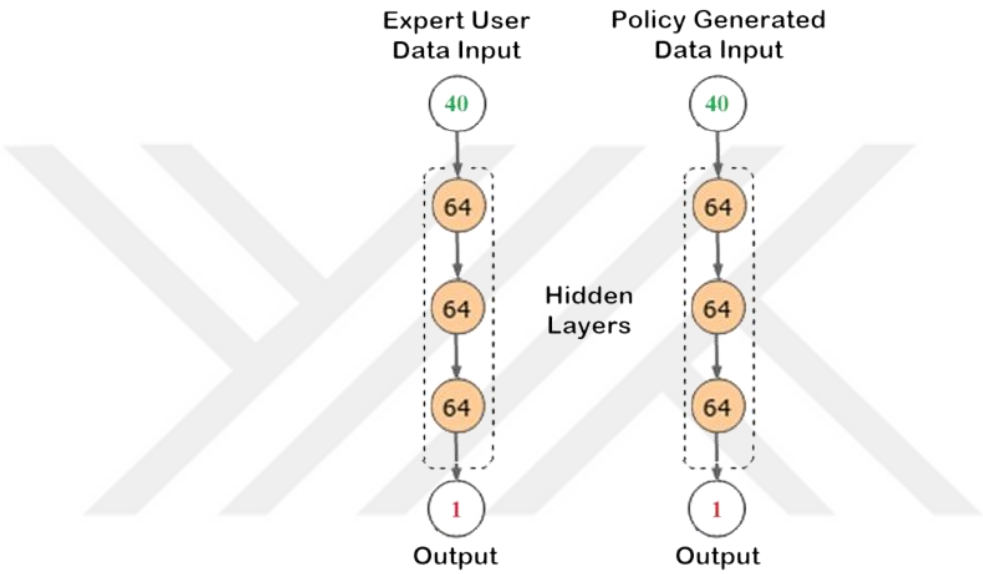

Figure 4.4 : Representation of the MLP discriminator architecture

The discriminator will be described with equations (2.19) for AIRL and $(D_\omega(s,a))$ for GAIL where probability can be calculated by using the MLP. AIRL minimizes the loss function with MLP by combining equation (2.17) with the disentangled modification, while GAIL uses MLP to minimize the loss based on equation (2.24). The MLP is anticipated to extract the essential characteristics of the data using stochastic gradient optimization and the Adam optimizer to distinguish between state-action pairs derived from the learned policy and those from expert demonstrations. The discriminator's output is fed back into the RL algorithm (with the reward represented by equations (2.20) for AIRL and (2.26) for GAIL) to assist in refining a policy that aligns more closely with human expertise. The architectures, input-output shapes, layer types and activation functions of the policy, value, and discriminator networks are summarized in Tables 4.1, 4.2, and 4.3.

Table 4.1 : Policy network architecture

Layer	Input shape	Layer type	Units/Filters	Activation	Output shape
input	35	-	-	-	35
dense1	35	fully connected	96	ReLU	96
dense2	96	fully connected	96	ReLU	96
dense3	96	fully connected	96	ReLU	96
output	96	fully connected	5	Softmax	5

Table 4.2 : Value network architecture

Layer	Input shape	Layer type	Units/Filters	Activation	Output shape
input	35	-	-	-	35
dense1	35	fully connected	128	ReLU	128
dense2	128	fully connected	128	ReLU	128
dense3	128	fully connected	128	ReLU	128
output	128	fully connected	1	Linear	1

Table 4.3 : Discriminator architecture

Layer	Input shape	Layer type	Units/Filters	Activation	Output shape
input	40	-	-	-	40
dense1	40	fully connected	64	ReLU	64
dense2	64	fully connected	64	ReLU	64
dense3	64	fully connected	64	ReLU	64
output	64	fully connected	1	Sigmoid	1

4.1.2.1 Adversarial inverse reinforcement learning

Adversarial inverse reinforcement learning (AIRL) is used in this study to extract interpretable and transferable reward functions from expert demonstrations, which helps make the policy learning process more transparent in autonomous driving. Unlike traditional imitation learning methods, AIRL not only mimics behavior but also identifies the underlying reward signals that drive the actions of the expert. This is particularly helpful in highway driving, where understanding the motivations behind actions like maintaining safety, improving efficiency, or making smooth lane changes. The reward functions learned by AIRL are better suited to adjusting the agent to new conditions or environments. The robustness provided by its adversarial training ensures that the learned reward function remains to be aligned with expert behavior, making AIRL a valuable method for safety-critical tasks in autonomous driving. The overview of the AIRL algorithm steps is presented in Figure 4.5.

Algorithm 1 Adversarial inverse reinforcement learning

1: Obtain expert trajectories τ_i^E
2: Initialize policy π and discriminator $D_{\theta,\phi}$.
3: **for** step t in $\{1, \ldots, \mathrm{N}\}$ **do**
4: Collect trajectories $\tau_i = (s_0, a_0, ..., s_T, a_T)$ by executing π.
5: Train $D_{\theta,\phi}$ via binary logistic regression to classify expert data τ_i^E from samples τ_i.
6: Update reward $r_{\theta,\phi}(s, a, s') \leftarrow \log D_{\theta,\phi}(s, a, s') - \log(1 - D_{\theta,\phi}(s, a, s'))$
7: Update π with respect to $r_{\theta,\phi}$ using PPO
8: **end for**

Figure 4.5 : AIRL pseudocode

4.1.2.2 Generative adversarial imitation learning

Generative adversarial imitation learning (GAIL) is employed in this research as it enables the acquisition of sophisticated sequential actions based on expert demonstrations without the need for a specific reward function to be designed or depended upon. This is particularly beneficial in the context of autonomous driving tasks such as the highway lane following and lane changing in which the design of a robust reward function is difficult, and in many cases may embed unintended biases weeping in the process. Using adversarial training with imitation learning, GAIL allows an agent to learn to imitate an expert's behavior by optimizing a policy such that the action-state distributions of the agent become indistinguishable from the expert. This allows to obtain high accuracy behavior cloning to be achievable in the environment with GAIL's ability to generalize and ensure the robustness of behaviors

across highway scenarios. Therefore, GAIL is a valuable approach in producing realistic policy-driven decision-making in control domains and as it aligns closely with the project goal. The overview of the GAIL algorithm steps is presented in Figure 4.6.

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the PPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta), \quad \text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \tag{18}$$

6: **end for**

Figure 4.6 : GAIL pseudocode

4.1.3 Hyperparameter selection

Table 4.4 presents the hyperparameters utilized during the training processes with the PPO policy and Table 4.5 describes the hyperparameters for GAIL discriminator and AIRL discriminator. The hyperparameters were kept the same within a model training. Different hyperparameter configurations were used in the training as shown in the tables to show the effect of the hyperparameters in the trainings.

Table 4.4 : PPO hyperparameter configurations used in trainings

Hyperparameters	Config-1	Config -2	Config -3	Config -4	Config -5
Learning rate	5e-5	5e-6	5e-6	1e-4	1e-3
# of iterations	150	150	150	150	150
Batch size	256	256	256	256	256
# of policy epoch	35	35	15	35	35
# of value epoch	20	20	20	20	20
Gamma	0.99	0.99	0.99	0.99	0.99
Lambda	0.95	0.95	0.95	0.97	0.97
Clipping	0.1	0.1	0.1	0.1	0.1

The values presented in Table 4.4 represent the hyperparameters utilized during training. The learning rate indicates the step size the optimizer employs when

modifying the model parameters throughout the training process. The number of iterations signifies how many iterations are sampled from each environment with each model update. The batch size specifies the count of samples in each minibatch applied during training. The number of policy epochs denotes how many times the policy model undergoes training for each update, while the number of value epochs relates to how many times the value model is trained within the same update cycle. The gamma parameter serves as a discount factor that helps establish the influence of rewards, particularly by prioritizing immediate rewards. The lambda parameter is essential for managing the balance between bias and variance in the framework. The clip value acts as the clipping parameter employed in PPO to avoid excessive alterations to the policy by constraining the difference between the old and new policy outputs. The configurations for the discriminator network are detailed in Table 4.5.

Table 4.5 : Discriminator hyperparameter configurations used in trainings

Hyperparameters	Config-1	Config -2	Config -3	Config -4	Config -5
Learning rate	7e-8	7e-7	7e-7	4e-8	7e-10
Batch size	512	512	512	512	512
# of epoch	5	10	10	5	10

4.2 Learning Setup

A formulation will be shown here for our path navigation problem which has explanations of the state space, action space, and reward function.

4.2.1 State space

Although this study is conducted in an offline simulation environment and does not involve raw sensor data such as images or LiDAR, it utilizes structured and measurable state-action vectors including position, velocity, throttle, steering, and heading information. These kinematic and control parameters are not only interpretable but also observable and recordable in real-world driving systems using common vehicle sensors such as GPS, IMU, Radar and CAN bus interfaces.

While the simulation environment simplifies certain aspects of the real world (e.g., perception noise, weather conditions, pedestrians/obstacles, or unexpected driver behavior), the core decision-making model-based on state-action mappings using imitation and inverse reinforcement learning remains transferable. Since the algorithm

focuses on learning driving behaviors such as lane keeping and lane changing from expert demonstrations, the methodology is generalizable and can serve as a foundation for real-world deployment, provided proper interfacing with sensor and control modules.

Therefore, although additional study would be required to integrate the model into a full autonomous driving task, the approach is considered realistic and grounded in fundamental real-world measurable dynamics.

As already mentioned in the Chapter 3, we used the current speed, steering value, throttle, and the vehicle's lane orientation vector as our state input, as illustrated in Figure 4.7. These properties can be specified as follows:

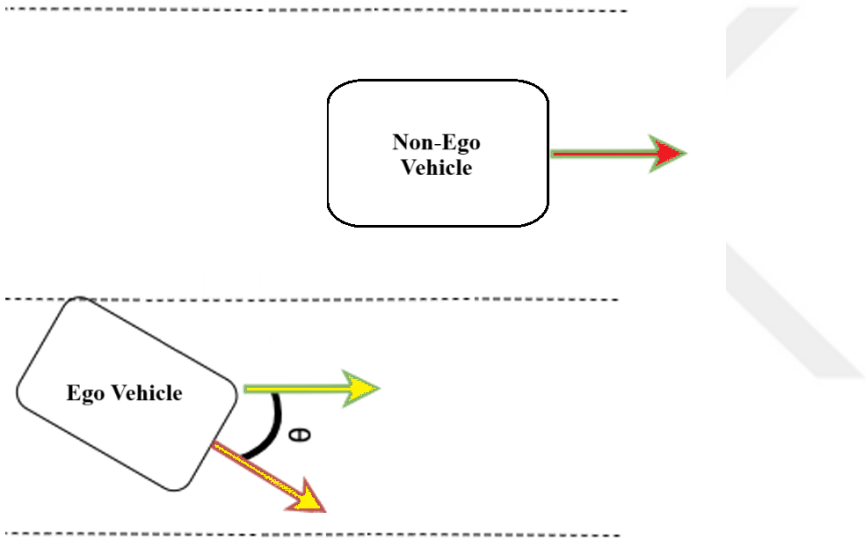

Figure 4.7 : External features

Throttle: This indicates the acceleration of the agent. The value varies while determining the target speed. The acceleration of the ego vehicle is calculated with $a_{ev} = P\ (v_d - v_c)$.

Current speed: The agent's current speed will be recorded in meters per second. However, for consistency and computational convenience, the speed values are normalized from m/s to a range of ([0, 1]), which is computed by dividing the current speed of the agent by the maximum reference speed of 30 m/s.

Steering: This measures the steering input of the agent in the time step. Steering values exist in the same range ([-1.0, 1.0]).

4.2.2 Action space

In the simulation, the vehicle is steered using the action values of the steering angle and the throttle input. Between these actions, steering spans to ([-1.0, 1.0]), while throttle is used to calculate the desired speed. We maintain these ranges in our implementation.

The input representation is given to our policy network, which is a multilayer perceptron (MLP), as shown in Figure 4.3. The outputs of the network is the predicted steering value or the predicted throttle value. We further simplify the problem, by operating strict speed interval conditions and without any other dynamic sources (except non-ego vehicles) that could desire further behavior necessity. This means we do not need to account for compliance with traffic rules and ultimately do not require an output for braking. In any case, our model ignores brake input, only uses the throttle input.

4.2.3 Reward function

Reward function r is defined in order to first obtain desirable driving behaviors, second improve agent performance, and third improve the training speed. To form a more natural and interpretable reward structure, the prominent factors that promote driving quality were first established.

- Desired velocity: Assume the current speed of the agent is represented as v_c m/s, while the target speed is set at v_d m/s. The agent earns the highest reward for maintaining a speed that closely to v_d. Due to inherent inaccuracies in throttle control, we establish a tolerance level characterized by a minimum speed v_{min} and a maximum speed v_{max}, within which the agent receives peak rewards.

- Preference for rightmost lane: In highway driving contexts, the rightmost lane is generally regarded as the safest lane, as it adheres to common traffic norms. To model these preferences in the reward process, we define a reward with +0.1 for the agent to stay or return to the rightmost lane whenever possible. If the agent is in the rightmost lane, it receives a positive reward. The increase in extra rewards stops when the ego vehicle leaves the rightmost lane unless a lane change is necessary to pass a slower vehicle in front. This constraint can

vary dynamically depending on the scenario complexity and any other actors present.

- Penalizations of failures: We must also penalize undesired behaviors. Any infraction or collision, as discussed in the Chapter 3, will be penalized -1 to the reward signal.

4.3 Results

This section presents the evaluation of the AIRL and GAIL algorithms in the autonomous driving task implemented in a simulation environment. The evaluation covers intra-algorithm comparisons across five different hyperparameter configurations for GAIL and AIRL individually, followed by a final cross-algorithm comparison between the best performing models from each method.

The expert policy was trained via PPO, which minimized reward function with the aim of maximizing efficient and safe highway driving, such as maintaining speed, staying within lane boundaries, and safely overtaking vehicles. This reward function was also utilized to evaluate the performances of the AIRL and GAIL agents, ensuring consistent benchmarking across expert and learning based policies. The expert dataset consisted of diverse state-action trajectories collected across a variety of traffic scenarios to support robust imitation learning.

For both GAIL and AIRL, five independent training runs (denoted as A1–A5 for AIRL and G1–G5 for GAIL) were conducted. Each configuration incorporated different hyperparameter settings, varying policy network parameters, policy learning rates, varying discriminator parameters and discriminator learning rates. This experimental design aimed to investigate both the sensitivity of each algorithm to hyperparameter variations and their robustness under diverse training conditions.

During training, the following key performance metrics were monitored at each iteration:

- Total reward: This is the cumulative sum of environment rewards earned by the agent, reflecting overall task performance. Higher total reward indicates the agent is more effectively achieving the driving objectives in CARLA.

- Collision rate: The fraction of time steps (or episodes) in which the agent collides with obstacles (e.g., non-ego vehicles). It is a direct safety compliance metric: lower collision rate means safer driving behavior.
- Jensen-Shannon divergence (JSD): A symmetrized f-divergence exists between two distributions. It is defined as the average of a pair of Kullback-Leibler divergences to the average distribution. We are here, therefore, computing the JSD based upon the state-action distributions of the expert and agent. A smaller JSD indicates the agent's behavior distribution closely matches the expert's, making it a natural metric for evaluating imitation sensitivity in learning methods (which can be viewed as minimizing distributional divergence).
- Wasserstein distance: This assesses the cost of converting one distribution into another through optimal transport. Intuitively, imagine the probability mass of one distribution as piles of earth; it is the minimal "work" required to move this mass to match the other distribution. Unlike KL divergence, the 1-Wasserstein distance is a true metric (satisfying symmetry and the triangle inequality) and remains meaningful even when distributions have non-overlapping support. In practice, lower Wasserstein values indicate closer match of the agent's distribution to the expert's, capturing differences in the full support of trajectories.
- Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) which are dimension reduction methods, were utilized to assess the alignment between expert and agent state-action distributions. PCA linearly transforms high-dimensional data into fewer dimensions, effectively retaining the largest variance and enabling straightforward interpretability. This makes PCA useful for examining broad patterns of how closely agent trajectories correspond to expert behaviors. t-SNE applies a nonlinear transformation, making it particularly suited to identifying and visualizing local structures and subtle clusters within complex data. Consequently, t-SNE visualizations offer detailed insights into the extent to which agent policies can replicate the finer-grained structures of expert trajectories.

- The Point-Adjusted F1 score serves as the harmonic average of precision and recall, as it strikes a balance between false positives and false negatives. We are using a “point-adjusted” version as is done in sequential tasks, where prior to scoring, the predicted segments for evaluation are temporally aligned with the ground-truth events. The point-adjusted F1 is primarily a metric for accounting for timing offsets involved by labeling the events. In short, the higher the point-adjusted F1 (closer to 1), the closer the discrete actions of the agent (i.e., steering, decelerating, etc.) were to those of the expert agent. F1 scores are commonly used to quantify fidelity of imitation.

  In the early training phases, GAIL and AIRL agents demonstrated little alignment to expert behavior with low F1 scores, high JSD and Wasserstein distances; such initial performance was not surprising as there had been limited updates in the policy and discriminator at the early training phase of the agents. As the iterations were progressed, improvements in behavior and metrics were observed. At iteration 150, we observed improvements in the configurations. To rigorously evaluate the learning quality of adversarial inverse reinforcement learning (AIRL) and generative adversarial imitation learning (GAIL) in autonomous highway driving tasks, five distinct configurations were executed for each algorithm, each with varying hyperparameter settings, including learning rates, policy and value network update frequencies, and discriminator training regimes. The models were assessed using a diverse set of metrics spanning task performance, safety, behavioral fidelity, and distribution alignment. The simulation results are shown in Table 4.6.

Table 4.6 : Hyperparameter comparison – PPO-AIRL and PPO-GAIL configurations

Configuration	AIRL-1	AIRL-2	AIRL-3	AIRL-4	AIRL-5	GAIL-1	GAIL-2	GAIL-3	GAIL-4	GAIL-5
Total reward	80.0739	12.9931	12.4818	79.9587	71.6619	81.3286	14.751	25.7431	82.4367	76.9414
Collision rate	0.00151	0.06467	0.06772	0.0015	0.0038	0.00116	0.05616	0.03085	0.00114	0.00251
PA F1 score	0.7518	0.8455	0.8655	0.7545	0.7582	0.5013	0.8332	0.826	0.4966	0.5208
JSD	0.4952	0.0998	0.2553	0.5061	0.4809	0.549	0.0978	0.1034	0.6078	0.6068
Wasserstein	1.0803	0.1306	0.3726	1.093	1.0612	1.6011	0.1099	0.1406	1.7011	1.6776

For AIRL, we evaluated five hyperparameter settings (varying learning rate, epochs, and lambda) and compared their performance across all metrics. The reward trajectories, shown in Figure 4.8, show that configurations AIRL-1 and AIRL-4 rapidly achieved high cumulative reward (~80) and stabilized, whereas AIRL-2 collapsed to a low reward (~13). Collision rates, demonstrated in Figure 4.9, mirrored this: the weaker runs (AIRL-2, AIRL-3) sustained high collision rates (~6.5%) throughout training, while the better runs (AIRL-1, AIRL-4, AIRL-5) drove collisions down nearly (~0.1%) after a few iterations. These patterns suggest that overly aggressive hyperparameters (e.g., too-high learning rate for discriminator) in AIRL-2 caused unstable policy learning (high collisions and low reward), while moderate settings (in AIRL-1 and AIRL-4) balanced exploration and imitation more effectively. Although AIRL-1, AIRL-4, and AIRL-5 achieved higher reward performance and lower collision rates, they did not align as closely with the expert behavior in terms of PA F1 scores and divergence measures as AIRL-2 and AIRL-3 did. Nevertheless, they still demonstrated a reasonably good level of imitation overall.

Jensen-Shannon divergence (JSD) and Wasserstein distance value showed varying patterns with varying AIRL configurations. AIRL-1, AIRL-4, and AIRL-5, for which rewards were high, and collision rate was low, depicted higher JSD and Wasserstein scores, 0.4952 and 1.0803 for AIRL-1, 0.5061 and 1.0930 for AIRL-4, and 0.4809 and 1.0612 for AIRL-5, respectively. Despite their higher reward and safety performance, these higher divergence values suggest a moderate difference between agent and expert behavior distributions. In contrast, AIRL-2 and AIRL-3 had substantially lower JSD (0.0998 and 0.2553) and Wasserstein (0.1306 and 0.3726) distances despite their much inferior reward performance and higher collision rate. This implies that, although these models closely approximated the expert regarding state-action distribution, they were not able to translate this closeness into effective task performance.

Figure 4.8 : Reward trends for AIRL configurations A1–A5 over training iterations

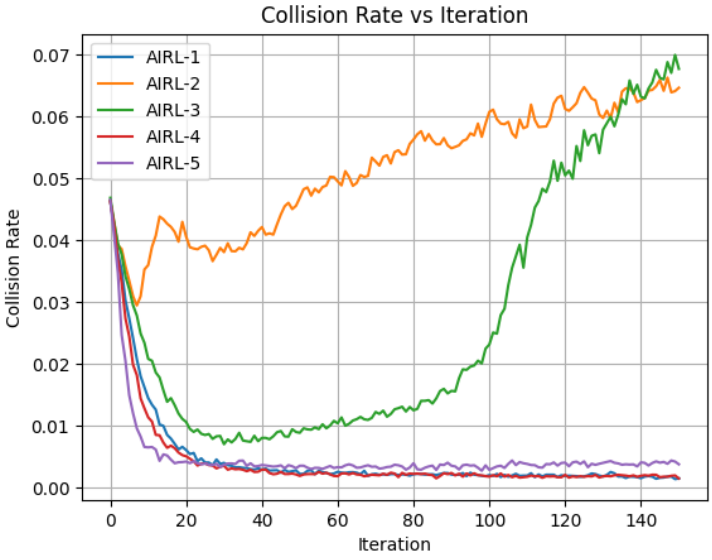

Figure 4.9 : Collision rates for AIRL configurations A1–A5 over training iterations

The GAIL experiments similarly showed stark differences across the five configurations. As in AIRL, two settings (GAIL-1 and GAIL-4) rapidly learned near-optimal reward (~80–82) and plateaued, whereas others (especially GAIL-2) collapsed to very low reward. In Figure 4.10 we see that GAIL-4 steadily attained the highest reward and kept it, while GAIL-2 spiked early then decayed to ~15. Collision rates, demonstrated in Figure 4.11, tracked this pattern: GAIL-1 and GAIL-4 rapidly reduced collisions to nearly 0.1%, whereas GAIL-2's collisions spiked above 5% before failing. These trends imply that GAIL-1, GAIL-4, and GAIL-5's hyperparameters

(e.g., adequate training epochs and moderated learning rate) enabled stable adversarial learning, while GAIL-2 and GAIL-3 likely used less training or an unstable configuration.

A similar pattern was observed for GAIL: methods GAIL-2 and GAIL-3, which had the lowest rewards and highest collision rates, produced higher PA F1 scores, whereas AIRL configurations 1, 4, and 5 yielded lower, moderate PA F1 scores.

The best configurations, GAIL-1, GAIL-4, and GAIL-5, with high reward scores and low collision rates, had much higher divergence scores, such as a JSD of 0.5490, 0.6078, and 0.6068 and Wasserstein distances of 1.6011, 1.7011, and 1.6776, respectively. On the other hand, GAIL-2 and GAIL-3 configurations also achieved very low JSD (0.0978 and 0.1034) and Wasserstein (0.1099 and 0.1406) scores, even though they performed suboptimal when reward and collision were concerned. This indicates in GAIL's interpretation of distribution-based metrics: low divergence does not guarantee successful imitation alone in actual task performance. The best-performing configurations, with the highest rewards and safety, also exhibited high divergence, indicating that optimizing GAIL strikes an effective balance in attaining high performance without being too far from a close, but not necessarily minimal, behavioral distribution to the expert.

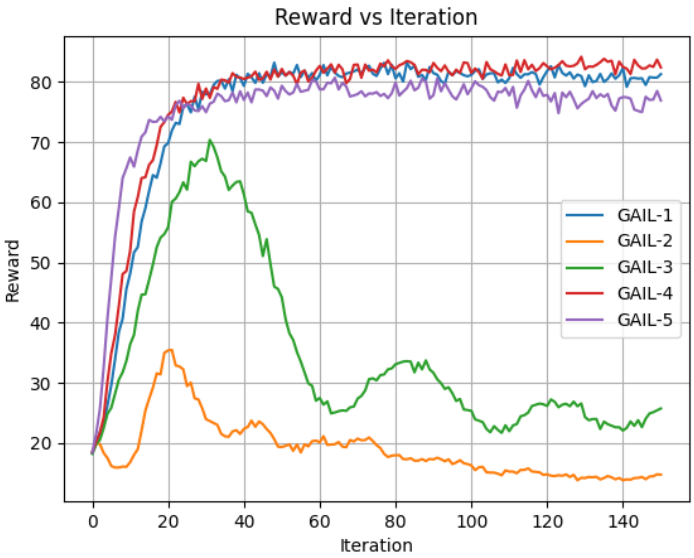

Figure 4.10 : Reward trends for GAIL configurations G1–G5 over training iterations

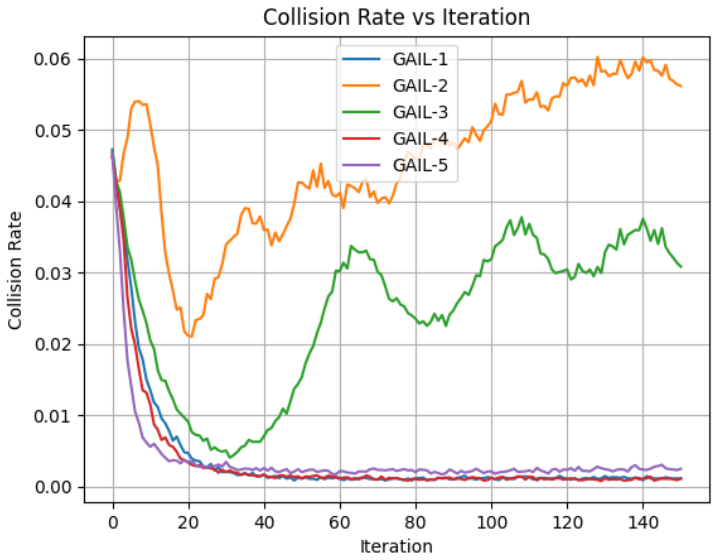

Figure 4.11 : Collision rates for GAIL configurations G1–G5 over training iterations

The varying behaviors across configurations reflect their hyperparameter choices. Configurations that achieved stable, high performance (AIRL-1, AIRL-4, AIRL-5; GAIL-1, GAIL-4, GAIL-5) likely used conservative learning rates and a balanced ratio of discriminator updates to policy updates. In adversarial IL, aggressive hyperparameters (e.g., too-high learning rates, excessive discriminator training per iteration, or insufficient policy epochs) can cause overfitting or oscillation . Indeed, GAIL-2 and GAIL-3 showed early reward decreases followed by collapse and high collisions, consistent with adversarial training becoming unstable. Similarly, AIRL-2 and AIRL-3 appear to have destabilized late in training. By contrast, the robust configurations likely had fewer discriminator training per iteration and learning rate, yielding smoother convergence.

Across both AIRL and GAIL, a general pattern emerges. Moderate learning rates and discriminator updates produce smooth improvement in reward and safe behavior, whereas extreme settings allow short-term gains but lead to divergence. For example, increasing discriminator capacity or update frequency tends to reduce distribution divergence metrics (lower JSD) but can harm policy stability, whereas higher entropy regularization or smaller learning rates improve stability (as in the more stable curves above). In our experiments, the best configured GAIL and AIRL models appear to have struck this balance: they achieved low JSD while still learning high reward policies.

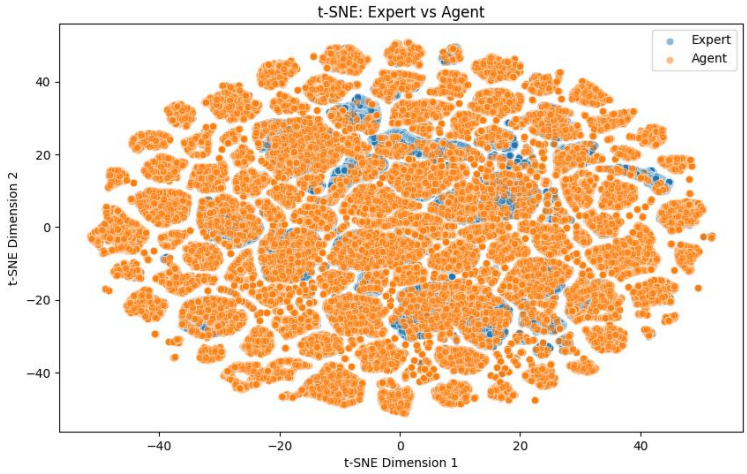

Figure 4.12 : t-SNE visualization of expert (blue) vs. AIRL-1 agent (orange) state-action samples

Each point is a sampled state-action pair embedded in 2D by t-SNE, as illustrated in Figure 4.12. The t-SNE map also shows the AIRL agent's trajectories (orange) largely covering the expert's clusters (blue). The orange clouds form a connected blob that encloses many blue points, indicating good coverage of the expert's variety of behaviors. Some blue points lie at edges or are slightly uncovered, suggesting minor discrepancies, but overall, the overlap is high. This visual overlap consists of AIRL's very low distribution distances.

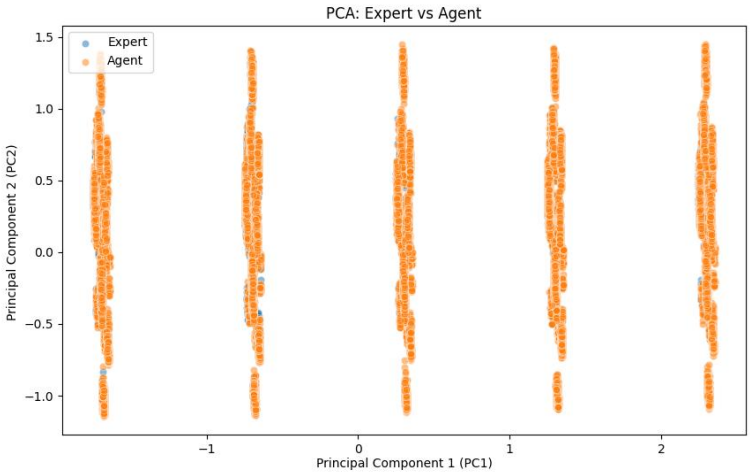

Figure 4.13 : PCA visualization of expert vs. AIRL-1 agent state-action pairs

In Figure 4.13, each column represents one principal component. A high degree of overlap in this low-dimensional embedding indicates that the agent's behaviors closely match those of the expert. The PCA plot for the AIRL-1 model, as demonstrated in Figure 4.13, shows that almost all agent points (orange) lie on the same 1D structures as the expert points (blue), implying very similar dominant state-action patterns. This geometric overlap corroborates low divergence metrics for AIRL: the agent's occupancy measure is closely aligned with the expert's in the principal subspace.

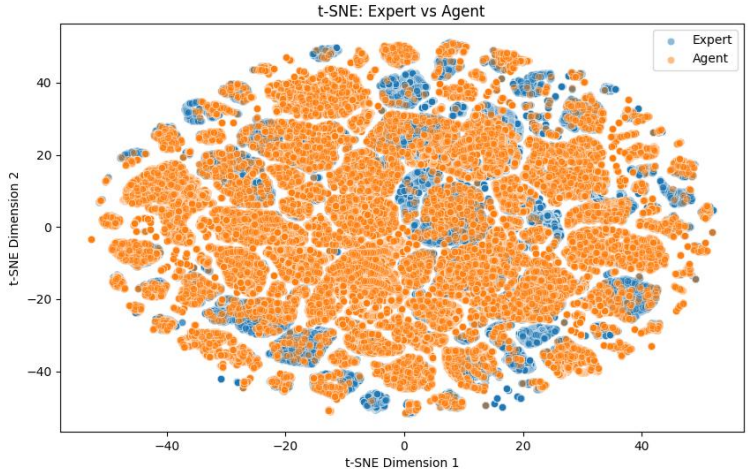

Figure 4.14 : t-SNE visualization of expert vs. GAIL-1 agent state-action samples

In Figure 4.14, the GAIL t-SNE plot also shows extensive overlap: the orange agent points fill the region around the blue expert points. Compared to AIRL's t-SNE, the GAIL agent's points (orange) are somewhat more spread out, and there are pockets of blue points not fully covered by orange. This indicates that while the GAIL policy captures the main modes of expert behavior, it has more dispersion and possibly extra areas. Nevertheless, the large overlap in both PCA and t-SNE spaces confirms that GAIL's best policy configurations also mimic the expert well, albeit slightly less tightly than AIRL's.

In Figure 4.15, the PCA visualization for the GAIL-1 models show five vertical columns of overlapping points. The expert (blue) and agent (orange) points largely coincide along these columns, but the spread of the agent's points is broader. Although the agent covers the expert support, there is less precise overlap – the GAIL agent appears to have more variance along the components. This matches the moderate slightly higher divergences: the GAIL policy's occupancy measure is close but not as perfectly aligned to the expert's as AIRL's.

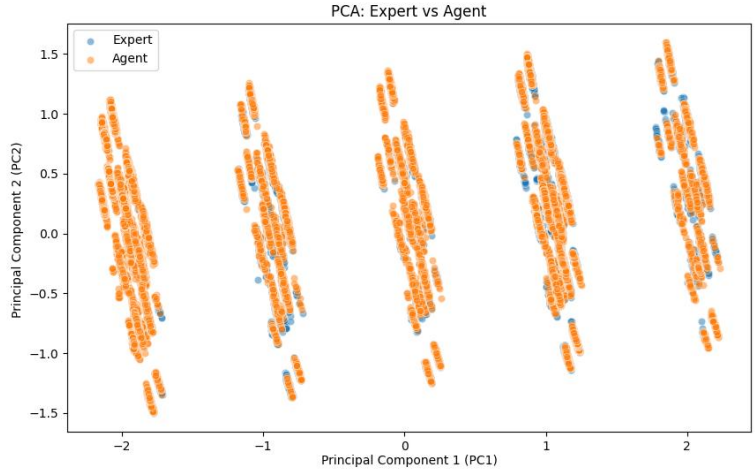

Figure 4.15 : PCA visualization of expert vs. GAIL-1 agent state-action pairs

The configurations from each algorithm (AIRL and GAIL) were selected for comparison based on high total reward, low collision count, and distribution alignment, as shown in Table 4.7.

Table 4.7 : AIRL-1 and GAIL-1 comparison

Metric	Total Reward	Collision Rate	PA F1 Score	JSD	Wasserstein
AIRL	80.0739	0.00151	0.7518	0.4952	1.0803
GAIL	81.3286	0.00116	0.5013	0.549	1.6011

Comparing the two methods, the selected AIRL and selected GAIL runs exhibited roughly equivalent final rewards (~80–81) and safety levels. However, GAIL-1 slightly outperformed AIRL-1 in several respects. GAIL-1's reward curve leveled out at the highest value, and it maintained near-zero collision, whereas AIRL-1's reward was marginally lower. On the other hand, the divergence metrics show that AIRL-1's agent distribution generally aligns more closely with the expert's. For example, the Jensen-Shannon and Wasserstein distances for AIRL-1 were lower than for GAIL-1, indicating a tighter match to the expert's behavior manifold. The PA F1 score was higher for AIRL-1, but GAIL's higher rewarded actions suggests it more faithfully reproduced expert decisions.

5. CONCLUSION

5.1 Conclusion

The main contribution of this thesis to the body of research is the method used to assess two distinct yet comparable reinforcement learning frameworks within an autonomous driving context. Instead of applying the methods used in this study to tasks unrelated to autonomous driving (such as robotic and classical control environments), this research contributes to the existing knowledge by integrating inverse reinforcement learning and imitation learning through adversarial learning. The study focuses on a specific driving task, enabling a meaningful and domain-specific application of reinforcement learning. Furthermore, we illustrate the practicality and advantages of evaluating driving tasks within a simplified environment as a foundational reference for future research.

Initially, a highway driving simulation environment was created using CARLA and highway-env to generate the datasets required to train the imitation learning and inverse reinforcement learning models, a human driver performed multiple driving maneuvers. The recorded data suitable for offline learning was stored in a dataset after filtering outliers. Then, the imitation learning and inverse reinforcement learning models were trained offline using these datasets. To further improve the vehicle's performance, a generative adversarial network (GAN) deep neural architecture was used. The trained models were then transferred to an ego vehicle in a highway driving environment using a policy-based approach. The results of the performance measurements and the rewards obtained during training, together with the crash assessments in the highway environment and the generalization assessments, support the hypothesis that data-driven learning methods (inverse reinforcement learning and imitation learning) can be used as effective learning approaches in advanced applications. However, the agent showed limited performance on driving maneuvers that were not explicitly demonstrated by the human driver.

We assume that when there is not enough variety of demonstration data for the agents to learn from, their ability to successfully generalize to unseen conditions is reduced. At each step, the agent obtains the distance to the closest vehicle in each lane, in addition to the lane number of its own vehicle. It then determines whether to change lanes or remain in its current lane. Each combination of state and action reflects the anticipated future reward. After running the simulation for number of iterations, the rewards converge to constant values. In the test phase, the model selects the maneuver with the highest expected reward to achieve its goal, which is closely aligned with human controlled driving behavior. Using this approach, it was observed that the vehicle learned a policy similar to a human driver's policy to avoid collisions with other vehicles. However, challenges arose due to driving maneuvers that were not sufficiently represented in the training dataset. Additional maneuver possibilities were identified, and complementary driving scenarios were examined to produce highway autonomous driving decisions more consistent with real-world driving.

Acquiring knowledge from data and applying it to unfamiliar situations is a vital goal for applications in autonomous vehicles. The act of driving presents a complex challenge due to its dynamic scenarios, tasks, and hazards, which demand both immediate and foresightful perception of real-time occurrences. Creating and sustaining a dataset that represents knowledge states and search procedures to tackle such an extensive challenge is inherently difficult.

Moreover, the intricacy of highway driving tasks in relation to autonomous vehicles has been examined, and the findings were reinforced with state-action rewards and collision data to support this analysis. Ultimately, while this research indicates that employing imitation learning and inverse reinforcement learning enhances the training and generalization abilities of reinforcement learning for specific driving contexts, additional studies are necessary to unify the state-of-the-art algorithms to enhance decision-making skills in the specified driving environments. Furthermore, transitioning to a realistic simulation setting that incorporates modules offering a mathematical depiction of the ego vehicle's surroundings will further substantiate the practical advantages of the research carried out in this study.

In summary, this study demonstrates that offline adversarial learning methods exhibit strong performance in ensuring safety for advanced decision-making in autonomous driving. However, they should be supported with other methods to produce high level

decisions. Using a generative adversarial network architecture within imitation and inverse reinforcement learning algorithms, a vehicle is trained to learn an optimal collision avoidance policy that enables safe and intelligent driving.

5.2 Future Work

This research has shown how adversarial learning technique, which is used with imitation learning and inverse reinforcement learning can enhance the decision making processes, for autonomous driving in highway environments. Nevertheless, there are possibilities for studies to build upon and broaden the conclusions reached in this research.

Using a combination of driving data and simulated information can enhance the adaptability of the models that are being trained. While the current study relies on human demonstrations in a simulated environment, integrating large-scale real-world datasets will provide a wider variety of driving scenarios and edge cases, leading to more robust policies. Moreover, implementing self-supervised learning methods can decrease the need for expert demonstrations provided manually and empower the vehicle to acquire knowledge from driving data. Furthermore, dealing with the difficulties of applying maneuvers in a variety of situations is still an area for study. While the trained models performed well in scenarios like those observed during training, their ability to handle novel and unseen situations was limited. Investigating methods for adaptation, innovative learning approaches and ways to enhance the dataset through augmentation could narrow this divide. Moreover, expanding the existing simulation setting to encompass challenging driving scenarios, like weather conditions, road construction, unpredictable variables encountered in time and evolving road shapes will enhance the practicality of the method suggested in real world situations. Integrating sensor data fusion, from LiDAR and camera sources can further enhance a sensing system. With adding such systems in the algorithm, deep learning methods like CNN or variational autoencoder can be used to boost the response rate of the vehicle behavior. Empower the vehicle to make well informed choices using more detailed environmental data.

Also, future work could expand on the results of this research by incorporating more advanced dimensionality reduction methods, alternative divergence measures, and algorithm comparisons. One possible direction involves using Independent

Component Analysis instead of or in conjunction with Principal Component Analysis to provide better separation of latent state-action features. Another promising approach would be to explore Cross Entropy as a replacement or complement to Jensen Shannon divergence to measure distributional similarity, which could provide finer insights into the fit between expert and agent behavioral patterns. Furthermore, while this work focuses on comparing generative adversarial imitation learning and adversarial inverse reinforcement learning, subsequent work could include evaluating generative adversarial imitation learning against Markov chain Monte Carlo-based imitation learning strategies, particularly in contexts that require probabilistic exploration and complex policy learning. These avenues could improve both the performance evaluation and interpretability of imitation and inverse reinforcement learning methodologies.

Putting the trained models into an autonomous driving system or a hardware in the loop (HIL) simulation will offer valuable insights into how useful they are in real life scenarios. This stage will include trying out the acquired strategies in driving situations and assessing how well they work within limitations, like delay times and safety standards. By addressing these challenges, future research can contribute to the development of more reliable, adaptive, and safe autonomous driving systems that integrate data-driven learning approaches with real-world driving complexities.

REFERENCES

[1] **Fadaie, J.** (2019). The state of modeling, simulation, and data utilization within industry: An autonomous vehicles perspective. *arXiv preprint arXiv:1910.06075*.

[2] **Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V.** (2017, October). CARLA: An open urban driving simulator. In *Conference on robot learning* (pp. 1-16). PMLR.

[3] **Codevilla, F., Müller, M., López, A., Koltun, V., & Dosovitskiy, A.** (2018, May). End-to-end driving via conditional imitation learning. In 2018 IEEE international conference on robotics and automation (ICRA) (pp. 4693-4700). IEEE.

[4] **Hu, A., Corrado, G., Griffiths, N., Murez, Z., Gurau, C., Yeo, H., ... & Shotton, J.** (2022). Model-based imitation learning for urban driving. Advances in Neural Information Processing Systems, 35, 20703-20716.

[5] **Pérez-Gil, Ó., Barea, R., López-Guillén, E., Bergasa, L. M., Gómez-Huélamo, C., Gutiérrez, R., & Díaz-Díaz, A.** (2022). Deep reinforcement learning based control for Autonomous Vehicles in CARLA. Multimedia Tools and Applications, 81(3), 3553-3576.

[6] **Leurent, E.** (2018). An environment for autonomous driving decision-making.

[7] **Wen, L., Fu, D., Li, X., Cai, X., Ma, T., Cai, P., ... & Qiao, Y.** (2023). Dilu: A knowledge-driven approach to autonomous driving with large language models. arXiv preprint arXiv:2309.16292.

[8] **Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., ... & Sapp, B.** (2023). Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. Advances in Neural Information Processing Systems, 36, 7730-7742.

[9] **Li, Y., Yuan, W., Zhang, S., Yan, W., Shen, Q., Wang, C., & Yang, M.** (2024). Choose your simulator wisely: A review on open-source simulators for autonomous driving. IEEE Transactions on Intelligent Vehicles.

[10] **Ho, J., & Ermon, S.** (2016). Generative adversarial imitation learning. *Advances in neural information processing systems*, *29*.

[11] **Couto, G. C. K., & Antonelo, E. A.** (2021, December). Generative adversarial imitation learning for end-to-end autonomous driving on urban environments. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7). IEEE.

[12] **Fu, J., Luo, K., & Levine, S.** (2017). Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.

[13] **Wang, P., Liu, D., Chen, J., Li, H., & Chan, C. Y.** (2021, May). Decision making for autonomous driving via augmented adversarial inverse reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1036-1042). IEEE.

[14] **Codevilla, F., Santana, E., López, A. M., & Gaidon, A.** (2019). Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9329-9338).

[15] **Xu, H., Jiang, L., Jianxiong, L., & Zhan, X.** (2022). A policy-guided imitation approach for offline reinforcement learning. *Advances in neural information processing systems*, *35*, 4085-4098.

[16] **Kang, Y., Liu, J., & Wang, D.** (2024). Off-Dynamics Inverse Reinforcement Learning. *IEEE Access*.

[17] **Jarboui, F., & Perchet, V.** (2021). Offline inverse reinforcement learning. *arXiv preprint arXiv:2106.05068*.

[18] **Udugama, B.** (2023). Review of Deep Reinforcement Learning for Autonomous Driving. *arXiv preprint arXiv:2302.06370*.

[19] **Kamil, Z., & Abdulazeez, A. M.** (2024). A Review on Deep Reinforcement Learning for Autonomous Driving. *The Indonesian Journal of Computer Science*, *13*(3).

[20] **Wang, S., Jia, D., & Weng, X.** (2018). Deep reinforcement learning for autonomous driving. *arXiv preprint arXiv:1811.11329*.

[21] **Sallab, A. E., Abdou, M., Perot, E., & Yogamani, S.** (2017). Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*.

[22] **Nageshrao, S., Tseng, H. E., & Filev, D.** (2019, October). Autonomous highway driving using deep reinforcement learning. In *2019 IEEE international conference on systems, man and cybernetics (SMC)* (pp. 2326-2331). IEEE.

[23] **Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P.** (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, *23*(6), 4909-4926.

[24] **Fernando, T., Denman, S., Sridharan, S., & Fookes, C.** (2020). Deep inverse reinforcement learning for behavior prediction in autonomous driving: Accurate forecasts of vehicle motion. *IEEE Signal Processing Magazine*, *38*(1), 87-96.

[25] **Zou, Q., Xiong, K., Fang, Q., & Jiang, B.** (2021). Deep imitation reinforcement learning for self-driving by vision. *CAAI Transactions on Intelligence Technology*, *6*(4), 493-503.

[26] **Url-1**, *<https://medium.com/@katalesanket90/machine-learning-in-self-driving-cars-8b5d1c685d3b>*, date retrieved 03.01.2025.

[27] **Spiegel, S., & Chen, J.** (2021). Using simulation data from gaming environments for training a deep learning algorithm on 3d point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *8*, 67-74.

[28] **Javanmard, R., Zabbah, A. H., Karimi, M., & Jeddisaravi, K.** (2020, December). Line following autonomous driving robot using deep learning. In *2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)* (pp. 1-5). IEEE.

[29] **Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zieba, K.** (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[30] **Wang, J., Zhang, W., Yang, H., Yeh, C. C. M., & Wang, L.** (2021). Visual analytics for rnn-based deep reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, *28*(12), 4141-4155.

[31] **Amini, A., Schwarting, W., Rosman, G., Araki, B., Karaman, S., & Rus, D.** (2018, October). Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 568-575). IEEE.

[32] **László, T.** (2022). Implementing a deep reinforcement learning model for autonomous driving.

[33] **Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y.** (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

[34] **Url-2**, *<https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html >*, date retrieved 07.01.2025.

[35] **Andrew, B., & Richard S, S.** (2018). Reinforcement learning: an introduction.

[36] **Kang, L., & Shen, H.** (2021, July). A reinforcement learning based decision-making system with aggressive driving behavior consideration for autonomous vehicles. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)* (pp. 1-9). IEEE.

[37] **Lichtlé, N., Jang, K., Shah, A., Vinitsky, E., Lee, J. W., & Bayen, A. M.** (2023, September). Traffic smoothing controllers for autonomous vehicles using deep reinforcement learning and real-world trajectory data. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 4346-4351). IEEE.

[38] **Lee, E., Han, Y., Lee, J. Y., & Son, B.** (2023). Modeling lane-changing behaviors for autonomous vehicles based on vehicle-to-vehicle communication. *IEEE Access*, *11*, 107997-108010.

[39] **Xu, D., Ding, Z., He, X., Zhao, H., Moze, M., Aioun, F., & Guillemard, F.** (2020). Learning from naturalistic driving data for human-like autonomous highway driving. *IEEE Transactions on Intelligent Transportation Systems*, *22*(12), 7341-7354.

[40] **Ye, M., Pu, L., Li, P., Lu, X., & Liu, Y.** (2022). Time-series-based personalized lane-changing decision-making model. *Sensors*, *22*(17), 6659.

[41] **Url-3,** *<https://spinningup.openai.com/en/latest/spinningup/rl_intro.html>*, date retrieved 10.01.2025.

[42] **Bellman, R.** (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, *60*(6), 503-515.

[43] **Url-4,** *<https://www.geeksforgeeks.org/what-is-the-difference-between-value-iteration-and-policy-iteration/>*, date retrieved 10.01.2025.

[44] **Url-5,** *<https://www.geeksforgeeks.org/bellman-equation/>*, date retrieved 12.01.2025.

[45] **Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O.** (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[46] **Hourigan, B.** Deep Reinforcement Learning and Imitation Learning for Autonomous Driving in a Minimalist Environment.

[47] **Adams, S., Cody, T., & Beling, P. A.** (2022). A survey of inverse reinforcement learning. *Artificial Intelligence Review*, *55*(6), 4307-4346.

[48] **Wu, Z., Sun, L., Zhan, W., Yang, C., & Tomizuka, M.** (2020). Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robotics and Automation Letters*, *5*(4), 5355-5362.

[49] **Deo, N., & Trivedi, M. M.** (2020). Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*.

[50] **Xu, Y., Xie, J., Zhao, T., Baker, C., Zhao, Y., & Wu, Y. N.** (2022). Energy-based continuous inverse optimal control. *IEEE transactions on neural networks and learning systems*, *34*(12), 10563-10577.

[51] **Huang, Z., Wu, J., & Lv, C.** (2021). Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning. *IEEE transactions on intelligent transportation systems*, *23*(8), 10239-10251.

[52] **Arora, S., & Doshi, P.** (2021). A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, *297*, 103500.

[53] **Alsaleh, R., & Sayed, T.** (2021). Markov-game modeling of cyclist-pedestrian interactions in shared spaces: A multi-agent adversarial inverse reinforcement learning approach. *Transportation research part C: emerging technologies*, *128*, 103191.

[54] **Finn, C., Levine, S., & Abbeel, P.** (2016, June). Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning* (pp. 49-58). PMLR.

[55] **Ziebart, B. D.** (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

[56] **Finn, C., Christiano, P., Abbeel, P., & Levine, S.** (2016). A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*.

[57] **Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C.** (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, *50*(2), 1-35.

[58] **Ross, S., Gordon, G., & Bagnell, D.** (2011, June). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 627-635). JMLR Workshop and Conference Proceedings.

[59] **Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., ... & Gruslys, A.** (2018, April). Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1).

[60] **Le Mero, L., Yi, D., Dianati, M., & Mouzakitis, A.** (2022). A survey on imitation learning techniques for end-to-end autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *23*(9), 14128-14147.

[61] **Zheng, B., Verma, S., Zhou, J., Tsang, I. W., & Chen, F.** (2022). Imitation learning: Progress, taxonomies and challenges. *IEEE Transactions on Neural Networks and Learning Systems*, *35*(5), 6322-6337.

[62] **Zhu, Z., & Zhao, H.** (2021). A survey of deep RL and IL for autonomous driving policy learning. *IEEE Transactions on Intelligent Transportation Systems*, *23*(9), 14043-14065.

[63] **Bronstein, E., Palatucci, M., Notz, D., White, B., Kuefler, A., Lu, Y., ... & Anguelov, D.** (2022, October). Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 8652-8659). IEEE.

[64] **Couto, G. C. K., & Antonelo, E. A.** (2024). Hierarchical generative adversarial imitation learning with mid-level input generation for autonomous driving on urban environments. *IEEE Transactions on Intelligent Vehicles*.

[65] **Kuefler, A., Morton, J., Wheeler, T., & Kochenderfer, M.** (2017, June). Imitating driver behavior with generative adversarial networks. In *2017 IEEE intelligent vehicles symposium (IV)* (pp. 204-211). IEEE.

[66] **Li, Y., Song, J., & Ermon, S.** (2017). Infogail: Interpretable imitation learning from visual demonstrations. *Advances in neural information processing systems*, *30*.

[67] **Bhattacharyya, R., Wulfe, B., Phillips, D. J., Kuefler, A., Morton, J., Senanayake, R., & Kochenderfer, M. J.** (2022). Modeling human driving behavior through generative adversarial imitation learning. *IEEE Transactions on Intelligent Transportation Systems*, *24*(3), 2874-2887.

[68] **Ziebart, B. D., Bagnell, J. A., & Dey, A. K.** (2010). Modeling interaction via the principle of maximum causal entropy**.**

[69] **Gu, X., Yin, X., Li, Y., Jin, X., & Li, Y.** (2024, February). Autonomous driving hazard scenario extraction and safety assessment based on crash reports and carla simulation. In *International Conference on Smart Transportation and City Engineering (STCE 2023)* (Vol. 13018, pp. 162-171). SPIE.

[70] **Zhao, J., Du, D., Yu, X., & Li, H.** (2024). Risk Scenario Generation for Autonomous Driving Systems based on Causal Bayesian Networks. *arXiv preprint arXiv:2405.16063*.

[71] **Rojas-Rueda, D., Nieuwenhuijsen, M., Khreis, H., & Frumkin, H.** (2020). Autonomous vehicles and public health. *Annual review of public health*, *41*, 1-17.

[72] **Kassens-Noor, E., Wilson, M., Cai, M., Durst, N., & Decaminada, T.** (2021). Autonomous vs. self-driving vehicles: the power of language to shape public perceptions. *Journal of Urban Technology*, *28*(3-4), 5-24.

[73] **Cho, R. L. T., Liu, J. S., & Ho, M. H. C.** (2021). The development of autonomous driving technology: Perspectives from patent citation analysis. *Transport Reviews*, *41*(5), 685-711.

[74] **Jung, C., Lee, D., Lee, S., & Shim, D. H.** (2020). V2X-communication-aided autonomous driving: System design and experimental validation. *Sensors*, *20*(10), 2903.

[75] **Revell, K. M., Richardson, J., Langdon, P., Bradley, M., Politis, I., Thompson, S., ... & Stanton, N. A.** (2021). Breaking the cycle of frustration: Applying Neisser's Perceptual Cycle Model to drivers of semi-autonomous vehicles. In *Designing Interaction and Interfaces for Automated Vehicles* (pp. 161-186). CRC Press.

[76] **Mehr, G., Ghorai, P., Zhang, C., Nayak, A., Patel, D., Sivashangaran, S., & Eskandarian, A.** (2022). X-CAR: An experimental vehicle platform for connected autonomy research. *IEEE Intelligent Transportation Systems Magazine*, *15*(2), 41-57.

[77] **Treiber, M., Hennecke, A., & Helbing, D.** (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, *62*(2), 1805.

[78] **Kesting, A., Treiber, M., & Helbing, D.** (2007). General lane-changing model MOBIL for car-following models. *Transportation Research Record*, *1999*(1), 86-94.

[79] **Url-6,** < >, date retrieved 28.01.2025.

[80] **Url-7,** <*https://carla.org*>, date retrieved 30.01.2025.

[81] **Url-8,** < *https://github.com/eleurent/highway-env*>, date retrieved 30.01.2025.

CURRICULUM VITAE

Name Surname : Aytuğ Onurhan Efil

EDUCATION :

- **B.Sc.** **:** 2019, Bahcesehir University, Faculty of Engineering and Natural Sciences, Mechatronics Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2020-2022 Eksenal Aluminum Extrusion Die Company, Design Engineer
- 2022 Bosch San.ve Tic. A.Ş, Application and Calibration Engineer

PUBLICATIONS AND PRESENTATIONS ON THE THESIS:

- **Efil A. O**., Doğan, M. 2025. Lane Change Decision Making For Autonomous Vehicles By Using Adversarial Learning Methods. *IGRS'25 – 4th International Graduate Research Symposium.*
- **Efil A. O**., Doğan, M. 2025. Lane Change Decision Making For Autonomous Vehicles By Using Adversarial Learning Methods. *IGRS'25 – 4th International Graduate Research Symposium, İstanbul, Turkey: 12-14 of May.*