Tunisian Republic
Ministry of Higher Education
and Scientific Research

University of Sfax

National School of Electronics
and Telecommunications of Sfax

Engineering Degree in:
Industrial Computer Engineering

Order No: PFA-GII 2025-39

End of The Year Project

Presented to

The National School of Electronics and Telecommunications of Sfax

For the completion of the

National Engineering Degree in:
Industrial Computer Engineering

Submitted by

Bali Rassem

Self-Driving Car Project Using a Simulation Environment

Presented on: 19/04/2025 before the examination committee:

Mr. Mohamed Neji	Supervisor
Mr. Ali Khalfallah	Examiner

DEDICATION

To my parents, **Mohammed** and **Chahrazed**, for their unconditional love, endless patience, and countless sacrifices. Your support has been the foundation of my journey, and everything I have achieved today is thanks to your constant presence and encouragement.

To my father and mother.

To my sister, **Sirine**, for always being there for me — with advice, motivation, and the little things that made a big difference. Your support has been invaluable throughout this experience.

To my beloved sister.

To my friends, for your encouragement, positivity, and loyalty that helped me stay strong in challenging moments.

To my dear friends.

To **all of you**,

I dedicate this work.

Bali Rassem

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to everyone who supported me throughout this final year project.

First and foremost, I would like to thank my supervisor, **Mr. Mohamed Neji**, for giving me the opportunity to work on this project and for trusting me with the freedom to explore, make decisions, and learn through experience. His guidance and confidence in my abilities played a crucial role in the successful completion of this work.

I would also like to extend my heartfelt thanks to my examiner, **Mr. Ali Khalfallah**, for his valuable time, insightful feedback, and constructive remarks. His expertise and evaluation contributed significantly to the quality and rigor of this work.

I wish to sincerely thank **Professor Imen Mabrouk** for her continuous encouragement, valuable advice, and belief in my potential. Her support during key phases of the project was deeply appreciated.

To my family — my parents and my sister — thank you for your unconditional support, encouragement, and patience. Your presence has always been my greatest source of strength.

I am also grateful to my friends, whose motivation and positivity helped me stay focused and determined throughout this journey.

Lastly, a special thanks to **OpenAI's ChatGPT**, whose assistance made the technical and writing aspects of this project clearer and more manageable. Your help truly made a difference during this intense academic experience.

Bali Rassem

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
BC	Behavioral Cloning
CARLA	Car Learning to Act
CNN	Convolutional Neural Network
DQN	Deep Q-Network
Fast-SCNN	Fast Semantic Segmentation Convolutional Neural Network
GTA V	Grand Theft Auto V
HD	Highway Driving
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
ML	Machine Learning
MobileNetV2	Mobile Vision Network Version 2
OpenCV	Open Source Computer Vision Library
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SB3	Stable-Baselines3
WHO	World Health Organization
YOLOv8	You Only Look Once version 8

GENERAL INTRODUCTION

In recent years, autonomous driving has moved from a futuristic concept to an active and rapidly growing area of research. Supported by advances in artificial intelligence, computer vision, and reinforcement learning, autonomous vehicles promise to transform the mobility landscape by improving road safety, reducing traffic congestion, and providing greater independence to various segments of the population.

Despite these advances, developing a reliable self-driving system remains a complex technical challenge. Training and evaluating such systems in real-world conditions is often expensive, risky, and constrained. As a result, simulation environments have become essential tools in the development of autonomous driving technologies, enabling safe and repeatable experimentation.

In this context, the video game Grand Theft Auto V (GTA5) was selected as the simulation platform for this project due to its high graphical realism, environmental diversity, and accessibility. Compared to standard research simulators like CARLA, GTA5 offers a more visually rich and variable environment, which is beneficial for creating robust and generalizable models.

The main goal of this project is to develop an AI-based self-driving system capable of navigating autonomously within the simulated environment. Several techniques were explored, including lane detection using OpenCV, semantic segmentation with YOLOv8, supervised learning through behavioral cloning, and reinforcement learning using DQN and PPO algorithms. After identifying the strengths and limitations of each approach, a hybrid solution was designed to combine the efficiency of supervised learning with the adaptability of reinforcement learning.

This report is structured into three main chapters. The first introduces the context, motivation, and challenges of the project. The second presents the technical development, including the implementation of various methods and the results obtained. The third chapter details the final

hybrid model, the custom reward system based on semantic segmentation, and a comparative performance analysis.

This work aims not only to document the development process, but also to explain the reasoning behind each decision, reflect on the challenges faced, and explore possible future improvements that could enhance the foundation laid by this project.

Chapter

1 Introduction & Problem Statement

Contents

1.1 Introduction

1.1.1 Overview of Self-Driving Cars and Their Challenges

Autonomous vehicles are a major innovation in the field of artificial intelligence and robotics. These systems are designed to navigate complex environments without human input, relying on a combination of sensors, vision algorithms, control systems, and decision-making models. However, achieving fully autonomous driving poses many technical challenges, including real-time object detection, lane keeping, traffic behavior prediction, and safe planning under uncertainty.

a) Brief Historical Context

The concept of self-driving cars dates back to the 1920s, when early demonstrations of radio-controlled vehicles captured public imagination. In the 1980s, significant academic progress was made through projects like Carnegie Mellon University's *Navlab* and the German *VaMoRs* vehicle developed by the University of Munich, which successfully drove at highway speeds.

A major milestone was reached in the 2000s with the DARPA Grand Challenges, where autonomous vehicles competed in desert and urban environments. These competitions accelerated research and laid the foundation for modern systems.

Since then, companies such as Google (Waymo), Tesla, and NVIDIA have driven commercial and research advancements, leading to increasingly capable autonomous systems that use deep learning, computer vision, LiDAR, radar, and real-time decision-making algorithms. Today, self-driving technology is being tested in real-world environments around the globe, though full Level 5 autonomy remains a complex and ongoing challenge.

b) Impact and Potential of Self-Driving Cars

Self-driving cars are more than just a technological curiosity — they have the potential to dramatically transform global transportation. According to the World Health Organization, over 1.3 million people die annually in road accidents, with more than 90% of these incidents

attributed to human error [25]. By removing the human factor, autonomous vehicles can help reduce fatalities, improve safety, and enhance mobility for people with disabilities or the elderly.

From an economic perspective, the autonomous vehicle market is expected to reach over $2.3 trillion by 2031, driven by industries such as logistics, ride-hailing, and smart city infrastructure [26]. In addition, optimizing traffic flow and fuel efficiency through AI could significantly reduce urban congestion and environmental pollution. When paired with electric vehicles, autonomous fleets can also contribute to long-term climate goals.

In short, self-driving technology represents a major leap toward safer, cleaner, and more intelligent transportation systems.

1.1.2 The Role of Simulation in Autonomous Driving

Simulation environments play a crucial role in the development of autonomous driving systems. They provide a safe, controlled, and cost-effective space to test and train models without the risks and limitations of real-world driving. In simulations, developers can easily reproduce scenarios, adjust conditions such as weather or time of day, and rapidly iterate on model performance. This flexibility allows for extensive experimentation, including the handling of rare or dangerous situations, which would be difficult or unsafe to replicate in reality.

Moreover, simulations are significantly less expensive than testing on real roads, which would otherwise require physical vehicles, test tracks, and safety protocols. By eliminating these logistical constraints, developers can conduct thousands of training hours and collect diverse datasets at virtually no cost.

Simulation environments also support seamless integration with AI development frameworks such as OpenCV, PyTorch, and reinforcement learning libraries — enabling a tight feedback loop between design, testing, and improvement. As a result, simulators are not just complementary tools; they are foundational platforms for scalable, reproducible, and robust self-driving AI development.

1.1.3 GTA V as a Practical and Flexible Simulation Platform

In the development of autonomous vehicle systems, the choice of simulation environment plays a crucial role in ensuring efficient experimentation and reliable model training. While CARLA is a widely used open-source simulator tailored for autonomous driving research, it presents several constraints, including a complex setup process, high-performance hardware demands, and limited environmental diversity.

Grand Theft Auto V (GTA V), on the other hand, offers a highly immersive and dynamic virtual world that is easier to configure and operate. With the help of tools such as Script Hook V and community-developed modifications, users can quickly adjust environmental variables—such as traffic density, time of day, or weather—without advanced technical effort.

GTA V stands out for its accessibility and development speed, making it particularly valuable in early-stage prototyping. It allows seamless automation of data collection (e.g., screen captures, driving inputs) via Python scripts, enabling efficient training of perception and control models. The rich and varied urban landscapes, combined with realistic visual fidelity, make it an excellent platform for simulating complex driving scenarios.

Due to these advantages, GTA V was selected as the simulation environment for this project. Its practicality, flexibility, and realism provide an ideal foundation for building and testing autonomous driving algorithms in a controlled yet challenging setting.

Figure 1.1: Example of a driving scenario in the simulation environment

1.2 Problem Definition & Initial Challenges

In the context of autonomous driving research, developing an AI system capable of making real-time driving decisions in complex environments remains a challenging problem. Specifically, this project aims to tackle the issue of controlling a virtual vehicle in a simulated world using only visual input. The agent must understand road structure, detect obstacles, and take safe and efficient actions — all without access to traditional navigation tools or sensor data like GPS, LiDAR, or radar. This constraint increases the reliance on deep learning models trained from simulation data to interpret the environment and make decisions reliably.

1.2.1 Main Goal

The primary objective of this project is to develop an AI system capable of driving a car autonomously in a simulated environment. The system must process visual input, identify road features and obstacles, and make real-time driving decisions with minimal human intervention.

1.2.2 Initial Technical Challenges

Several key challenges were encountered early in the development process:

- **Lane detection problems:** Traditional computer vision techniques such as edge detection and the Hough Transform were initially used to extract lane lines. However, they failed under variable lighting conditions, curved roads, and complex textures typical of open-world environments.
- **Object detection and segmentation difficulties:** Accurate distinction between roads, sidewalks, vehicles, and pedestrians required pixel-level understanding. This demanded the use of deep learning models and high-quality annotations. Fast-SCNN, despite being lightweight, produced low-quality segmentation results when tested using pre-trained weights, while YOLOv8 required substantial dataset cleaning and formatting to achieve consistent results.
- **Supervised learning limitations:** Behavioral cloning using CNNs was highly sensitive to dataset quality and balance. Early training results showed that overrepresented classes such as "going forward" dominated model behavior, requiring class balancing and extensive experimentation with different CNN architectures (CNN v1, CNN v2, MobileNetV2).
- **Reinforcement learning instability:** Methods such as Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) were tested, but both showed high sensitivity to reward tuning and exploration strategy. Without proper environment feedback and segmentation support, models were slow to converge and often learned suboptimal or unstable policies.

1.3 Methodology Overview

To address the project objectives, several approaches and architectures were explored across different learning paradigms:

- **Lane detection (OpenCV):** Basic computer vision techniques such as edge detection and Hough Transform were used to extract lane lines from the driving scene.
- **Object detection and semantic segmentation:** Two deep learning models were tested and compared. **YOLOv8**, a real-time object detection and segmentation framework, was trained to detect classes such as road, sidewalk, car, and person. Additionally, **Fast-SCNN**, a lightweight segmentation network designed for real-time performance, was implemented to evaluate its efficiency in segmenting urban scenes.
- **Supervised learning (behavioral cloning):** Driving data was collected from human gameplay, and three different convolutional neural network (CNN) architectures were trained to imitate driving behavior:
  - **CNN v1:** A simple model composed of 3 convolutional layers (with 32, 64, and 128 filters), followed by flattening and two dense layers (128 units and softmax output for 6 action classes).
  - **CNN v2:** A deeper model with two stacked Conv2D layers per block, more filters (up to 64), and a `Dropout(0.4)` layer before the dense output to reduce overfitting.
  - **MobileNetV2:** A pre-trained MobileNetV2 backbone (with frozen layers), followed by a custom classification head using Global Average Pooling and dense layers. This model was used to leverage transfer learning and improve generalization with limited data.
- **Reinforcement learning:** Two policy optimization strategies were tested:
  - **DQN (Deep Q-Network):** A value-based reinforcement learning algorithm that maps state-action pairs to Q-values.
  - **PPO (Proximal Policy Optimization):** A policy-gradient method that aims for stable and efficient learning by controlling policy updates.

- **Hybrid approach (Supervised + RL):** After training the supervised model via behavioral cloning, reinforcement learning was used to fine-tune the model's performance, correct poor driving behavior, and improve decision-making in unfamiliar scenarios.

All of these architectures will be tested and analyzed in the next chapter, where their performance will be compared based on accuracy, stability, training efficiency, and real-time responsiveness within the simulated environment.

1.4 Conclusion

This chapter introduced the general context of autonomous driving and highlighted the motivations behind using a simulation environment for developing AI-based self-driving systems. After evaluating different simulation tools, a high-fidelity virtual platform was selected to support both perception and decision-making tasks.

We then presented the main objectives of the project, discussed the early challenges encountered during the first development phases, and provided an overview of the techniques explored to address them — including lane detection, semantic segmentation, reinforcement learning, and behavioral cloning.

These initial insights laid the foundation for the technical implementation described in the following chapter, where each approach will be detailed and evaluated based on its results.

Chapter

2 Development & Experiments

Contents

2.1 Introduction

This chapter presents the technical evolution of the project, from early-stage approaches to the development of more robust solutions. It details the experiments conducted using traditional computer vision methods, supervised learning, and reinforcement learning, leading up to the final hybrid architecture. Each method was tested within the simulation environment, and its limitations were carefully analyzed to guide future decisions.

2.2 Early Approaches and Their Limitations

2.2.1 Lane Detection with OpenCV

The first approach focused on detecting road lanes using traditional image processing techniques, specifically Canny edge detection and the Hough Line Transform. While these methods performed well on clear, high-contrast roads, they struggled significantly under variable lighting conditions, complex textures, curved roads, and shadows.

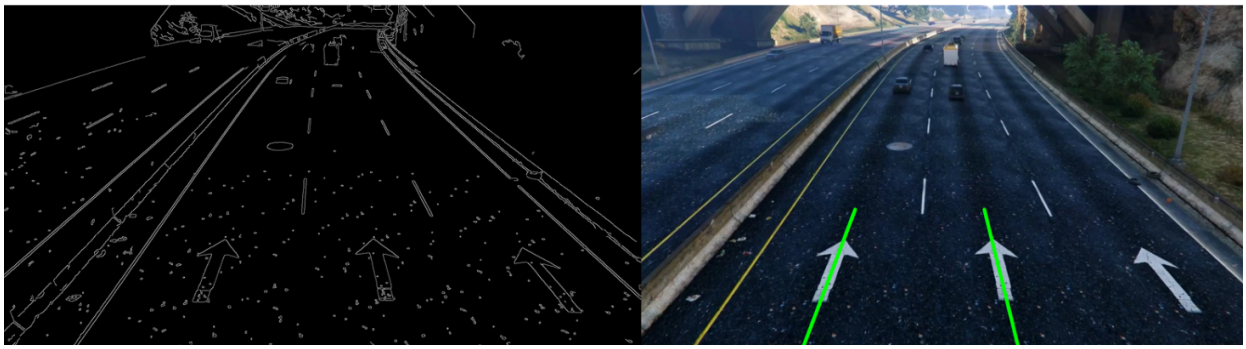


Figure 2.1: Result of lane detection using OpenCV techniques

Due to the lack of robustness and generalization, this approach was abandoned in favor of data-driven solutions based on deep learning.

2.2.2 Semantic Segmentation

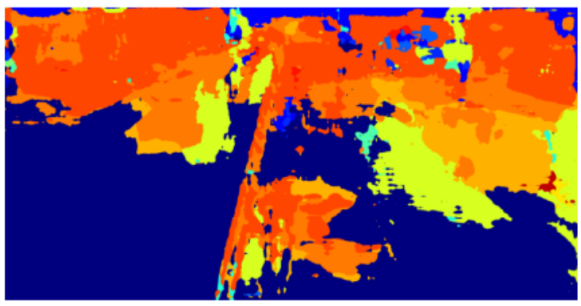
To enable the car to understand its environment and navigate effectively, semantic segmentation was used to classify each pixel into relevant categories. Two models were tested and compared for this task: **Fast-SCNN** and **YOLOv8 (nano)**.

a) Fast-SCNN vs YOLOv8 – Initial Comparison

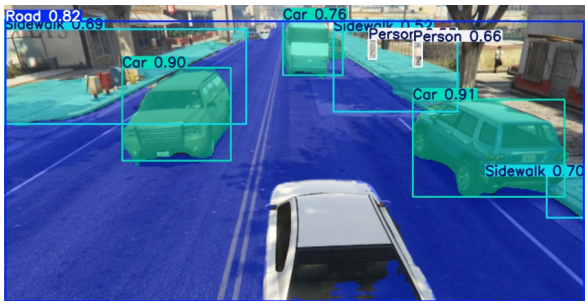
Fast-SCNN is a lightweight semantic segmentation model optimized for mobile deployment. A pre-trained version was tested without additional fine-tuning. However, the results were unsatisfactory — with blurry segmentation boundaries, missing classes, and poor handling of smaller objects like pedestrians.

YOLOv8 (nano) was also tested, using its built-in segmentation head and a short training phase on the filtered dataset. The results were significantly better: object contours were clearer, class separation was more accurate, and the segmentation output was more stable across different scenarios.

(a) Original Input Image



(b) Fast-SCNN Output



(c) YOLOv8 Output

Figure 2.2: Comparison between the original input and segmentation outputs from Fast-SCNN and YOLOv8

Model Selection Conclusion: After testing both models under identical conditions, **YOLOv8** was selected as the segmentation backbone for the project. Its ability to produce more accurate results with minimal training, along with its fast and efficient inference, made it the most suitable option for real-time deployment in the self-driving pipeline.

b) Overview of the Dataset

The dataset used in this project was sourced from the **Playing for Data** project [23], developed by the TU Darmstadt Visual Inference Lab. It consists of synthetic urban driving scenes rendered from the Grand Theft Auto V (GTA V) environment, and provides pixel-level semantic segmentation for each image.

The full dataset is divided into three parts, each containing high-resolution RGB images and their corresponding segmentation masks:

- **Part 1:** 5.71 GB of images, 69.2 MB of labels
- **Part 2:** 5.73 GB of images, 71.8 MB of labels
- **Part 3:** 5.72 GB of images, 67.4 MB of labels

In total, this includes tens of thousands of images, all annotated with semantic masks covering **19 classes**, such as road, sidewalk, building, car, pedestrian, vegetation, and more.

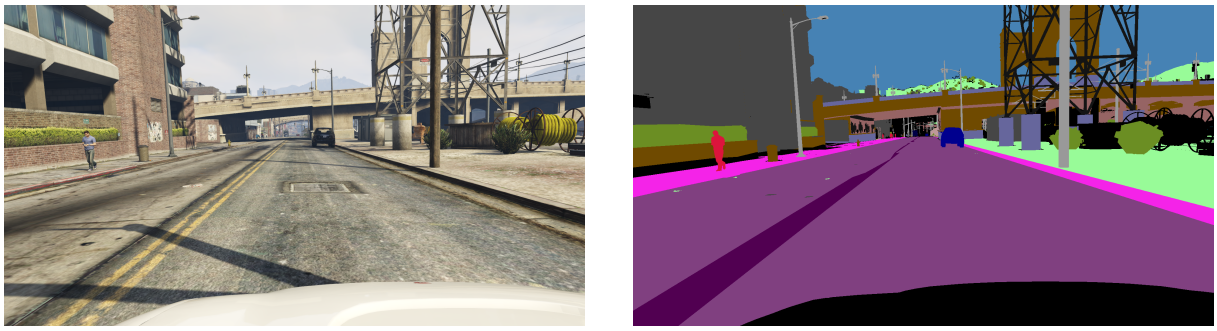
For the purpose of our training and evaluation, we filtered the dataset to focus on the four most relevant classes:

Dataset Characteristics: Selected Classes for Segmentation:

- Road
- Sidewalk
- Car
- Person

Table 2.1: Summary of key characteristics of the semantic segmentation dataset

Characteristic	Details
Total size used	17.1 GB of RGB images, 208.4 MB of label masks
Image resolution	1914 × 1052 pixels
Label format	PNG segmentation masks with RGBA color encoding
Classes in original dataset	19



(a) Original Image **(b) Segmentation Mask**

Figure 2.3: Example from the dataset showing a simulated driving scene and its corresponding segmentation mask

This figure presents a pair of images from the dataset. On the left, the original RGB image is shown as captured from the simulated driving environment. On the right, its corresponding segmentation mask is displayed, where each region is color-coded to represent classes such as road, sidewalk, vehicle, or pedestrian. These annotated masks are used to train the segmentation model to identify and distinguish between key elements in the driving scene.

c) Annotation Generation and Filtering Process

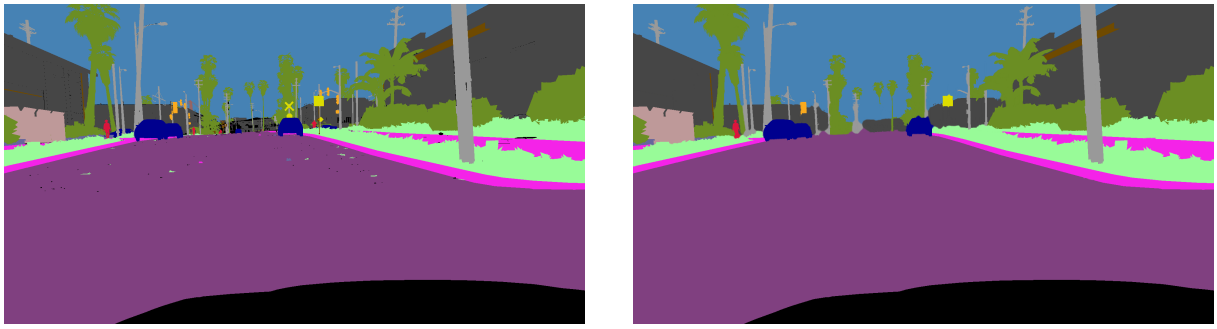
Although the dataset included segmentation masks, annotations were not provided in YOLO-compatible format. A custom script was used to convert RGBA mask regions into polygons and assign class IDs using color mappings.

A cleaning process was applied to:

- Remove corrupted or incomplete masks

- Eliminate irrelevant or rare classes
- Focus on 4 key classes: Road, Sidewalk, Person, and Car

d) Before and After Filtering Comparison



(a) Raw unfiltered mask **(b) Cleaned and filtered mask**

Figure 2.4: Comparison between unfiltered and filtered segmentation masks

This figure highlights the importance of preprocessing segmentation masks before training. The raw unfiltered mask (a) contains visual noise and small pixel artifacts resulting from labeling errors or weak contours in the dataset. These inconsistencies can negatively affect the learning process by introducing false patterns or mislabeled pixels. To address this, filtering techniques were applied to smooth boundaries, remove irrelevant isolated pixels, and correct inconsistencies in the masks. The result is a cleaner and more structured segmentation mask (b), which improves training stability and allows the model to better generalize on real-world or simulated inputs.

e) Final Dataset Classes and Distribution

The segmentation model was trained using 4 filtered classes:

Table 2.2: Mapping of segmentation class IDs to their corresponding labels

Class ID	Class Label
0	Road
1	Sidewalk
2	Person
3	Car

To prevent class imbalance, Road and Sidewalk annotations were downsampled to 25,000 examples each.

Table 2.3: Final class distribution after filtering and balancing

Class	Label ID	Final Annotations
Road	0	25,000
Sidewalk	1	25,000
Person	2	20,286
Car	3	30,341

f) Model Training and Evaluation Results

The YOLOv8 segmentation model was trained on Kaggle using GPU acceleration. Training metrics show that the model generalized well across simulation scenes. A visual result and confusion matrix are shown below:

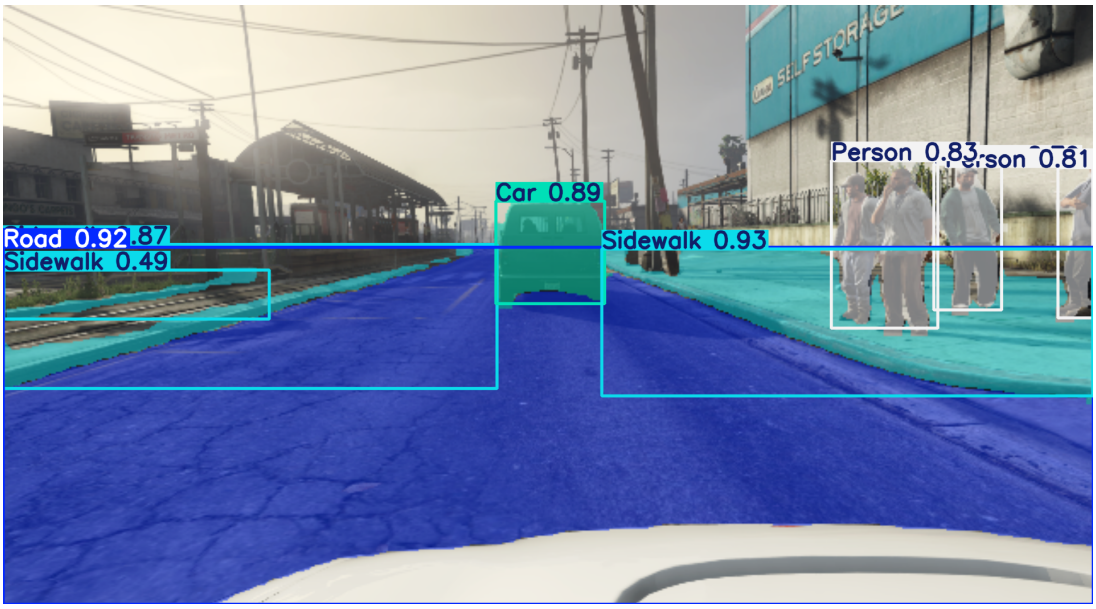


Figure 2.5: YOLOv8 predicted output for road, sidewalk, person, and car classes

This image illustrates a typical prediction output of the YOLOv8 segmentation model. It successfully detects and segments multiple key classes in the scene, including road, sidewalk, cars, and persons. The colored overlay highlights class-specific regions, while bounding boxes with confidence scores confirm the object detection aspect of the model. This visual output demonstrates the model's ability to differentiate between scene components in a real-time environment.

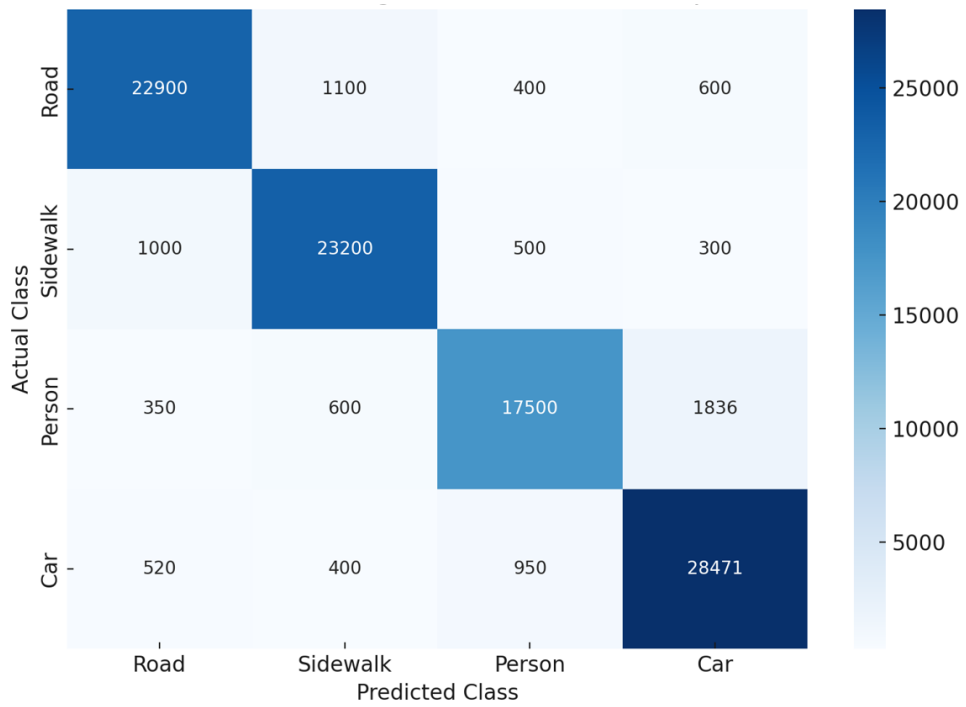


Figure 2.6: Confusion matrix for YOLOv8 on the test set

The confusion matrix provides a quantitative evaluation of the model's performance across the four targeted classes. Each row represents the actual class, while each column indicates the predicted class. The high diagonal values indicate that YOLOv8 was able to correctly classify most of the instances in each category, particularly for Road and Sidewalk. Minor confusion exists between similar classes like Car and Person, which is expected due to overlapping spatial or color features in certain scenarios.

g) F1 Score and Accuracy Metrics

Table 2.4: YOLOv8 performance metrics by class

Class	Precision	Recall	F1-Score	IoU
Road	0.925	0.916	0.920	0.852
Sidewalk	0.917	0.928	0.922	0.856
Person	0.904	0.863	0.883	0.791
Car	0.912	0.938	0.925	0.861
Macro Avg	0.914	0.911	0.913	0.840

The results presented in Table 2.4 demonstrate strong performance of the YOLOv8 segmentation model across all target classes. The F1-scores for each class are consistently high, with the **Car** class achieving the best result at 0.925, followed closely by **Sidewalk** and **Road**, both above 0.92. This indicates that the model is capable of accurately detecting and segmenting essential elements in the driving environment. The **Person** class exhibits a slightly lower F1-score (0.883), which can be attributed to the smaller size and higher variability of pedestrian instances in the dataset. Nevertheless, the macro average F1-score of **0.913** and IoU of **0.840** confirm the overall robustness and generalization ability of the model. These results validate the model's suitability for downstream tasks such as reward calculation and perception in autonomous driving scenarios.

h) Final Model Choice Justification

While both Fast-SCNN and YOLOv8 were tested, the decision to proceed with YOLOv8 was based on empirical results. Fast-SCNN, when used without additional training, failed to produce satisfactory segmentation for real-time control. In contrast, YOLOv8 offered superior results with minimal training and provided efficient inference, making it the most suitable choice for integration into the self-driving system.

2.2.3 Reinforcement Learning with DQN and PPO

Reinforcement learning was tested using Stable-Baselines3 implementations of DQN and PPO. These algorithms were used to train an agent based on trial-and-error behavior, with custom rewards designed from the segmentation model's outputs (e.g., driving on road = positive reward, hitting sidewalks = negative reward). The rewards were generated based on the agent's position and actions as classified by the segmentation model, which provided a way to evaluate its behavior in relation to the road, sidewalks, and other elements in the environment.

The training process was highly resource-intensive and time-consuming, requiring a very large number of episodes to learn even basic driving behaviors. Additionally, hardware limitations significantly affected training speed and convergence.

The agent was intended to learn in real-time during episodes. Instead of training in a conventional setup, the environment was allowed to interact with the agent and control actions, such as random steering commands, to facilitate learning. However, due to the high complexity and computational demands of this setup, the process was halted prematurely before it could yield meaningful results, and no final learning curve or performance could be generated.

During the development, both **Deep Q-Network (DQN)** and **Proximal Policy Optimization (PPO)** were explored. After conducting research and analysis, **PPO** was selected as the final reinforcement learning algorithm due to its superior stability in continuous driving tasks, which is essential for real-time driving environments.

The following table compares DQN and PPO, highlighting their key differences and why PPO was preferred for this project:

Table 2.5: Comparison of DQN and PPO for reinforcement learning in autonomous driving tasks

Feature	**DQN**	**PPO**
Algorithm Type	Value-based	Policy-based
Stability	Can struggle with stability in continuous tasks	More stable, better for continuous tasks
Sample Efficiency	Less efficient, requires more training data	More sample-efficient, works well with limited data
Exploration Strategy	epsilon-greedy exploration	Uses entropy bonus for exploration
Suitability for Continuous Action Space	Less effective for continuous spaces	Well-suited for continuous action spaces
Training Speed	Slower convergence in complex environments	Faster convergence in continuous environments
Use in Real-Time Systems	Requires fine-tuning to work in real-time applications	Can be adapted for real-time applications more easily

Based on the table above, **PPO** was chosen for this project because it offers more stability in continuous driving tasks and is better suited for real-time environments. In particular, PPO's ability to efficiently handle large action spaces and its stability during training made it the ideal choice for the dynamic and complex nature of autonomous driving.

Despite these advantages, reinforcement learning still proved to be highly resource-intensive and time-consuming. The agent required many episodes to learn even basic behaviors. As a result, reinforcement learning was set aside in favor of a more efficient supervised learning approach, which provided faster convergence and better resource utilization.

2.3 Supervised Learning for Autonomous Driving

2.3.1 Behavioral Cloning Approach

A dataset of human driving behavior was collected from the simulation environment. Each frame was recorded along with the corresponding key press (steering, throttle, brake), forming a dataset for behavioral cloning. A convolutional neural network (CNN) was trained to map visual input to driving commands.

Figure 2.7: Pipeline of behavioral cloning: image → steering command

This figure shows four examples of input images paired with their corresponding driving actions, such as turning left, going forward, stopping, and combining forward with right turns. These action labels were generated from key presses captured during data collection. By training on these annotated frames, the behavioral cloning model learns to predict steering commands directly from visual input, effectively mimicking human driving behavior in similar road scenarios.

2.3.2 Class Distribution Before and After Balancing

The initial dataset contained 158,047 total samples. The distribution across different actions was unbalanced, with the "Forward" action significantly outnumbering other actions. After removing idle frames, the class distribution was as follows:

Table 2.6: Class distribution before balancing (After removing idle frames — 158,047 total samples)

Class ID	Action	Samples	% of Dataset
0	Forward	118,484	74.96%
1	Left	4,147	2.62%
2	Right	5,906	3.74%
3	Stop	5,092	3.22%
4	Forward + Left	12,827	8.12%
5	Forward + Right	11,591	7.33%
Total		158,047	100%

After balancing the dataset to ensure a more even distribution, we reduced the total number of samples to 79,563. The class distribution after balancing is shown below:

Table 2.7: Class distribution after balancing (79,563 total samples)

Class ID	Action	Samples	% of Dataset
0	Forward	40,000	50.27%
1	Left	4,147	5.21%
2	Right	5,906	7.42%
3	Stop	5,092	6.40%
4	Forward + Left	12,827	16.12%
5	Forward + Right	11,591	14.57%
Total		79,563	100%

2.3.3 Model Evaluation Results

In this project, three different CNN-based architectures were trained to classify self-driving car actions from grayscale images:

- **CNN V1 (MiniDriveNet):** A simple and fast baseline model. While it achieved high training accuracy (98.3%), it began overfitting early and peaked at 76.3% validation accuracy. Validation loss increased after epoch 5.
- **CNN V2 (DeepDriveNet):** A deeper CNN architecture with dropout regularization. This model achieved 93.5% training accuracy and consistently reached 76.7% validation accuracy, showing the best generalization and stability across epochs. It demonstrated the lowest validation loss and was selected as the best-performing model.
- **CNN V3 (TransferDriveNet – MobileNetV2):** A transfer learning model using a frozen MobileNetV2 backbone. Despite stable training, it underperformed with 63.5% training accuracy and only 59.8% validation accuracy, indicating underfitting due to the frozen layers and domain difference.

All models were trained on a balanced dataset of 79,563 images, with a 15% validation split. The best model (CNN V2) will be used in downstream deployment and further experimentation.

The following plots compare the validation accuracy and validation loss for each of the three models:

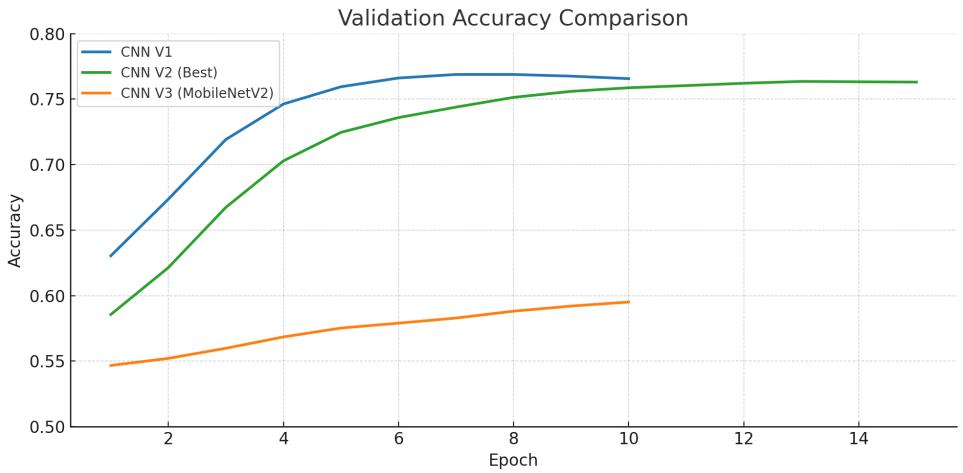


Figure 2.8: Validation Accuracy Comparison between CNN V1, CNN V2 (Best), and CNN V3 (MobileNetV2)

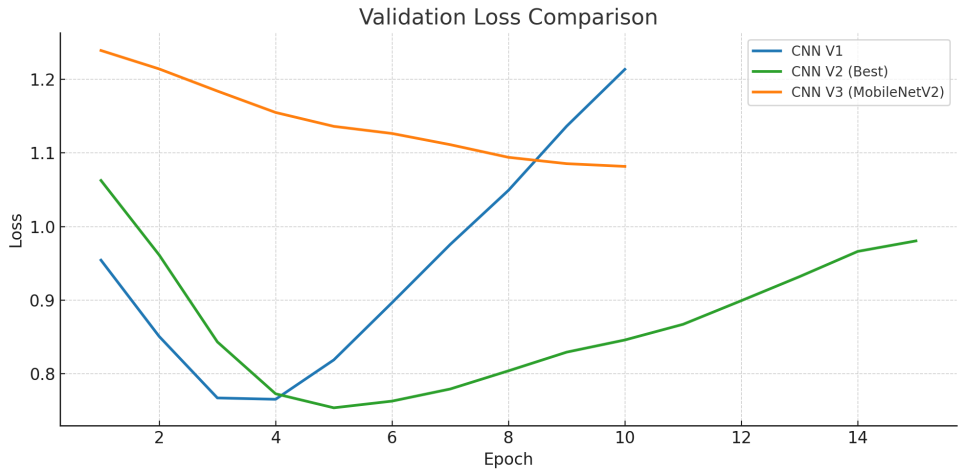


Figure 2.9: Validation Loss Comparison between CNN V1, CNN V2 (Best), and CNN V3 (MobileNetV2)

Figures 2.8 and 2.9 show the validation accuracy and loss curves for the three CNN architectures evaluated during training.

CNN V1 starts strong but quickly overfits, as evidenced by its decreasing accuracy and increasing loss after epoch 5. CNN V2 demonstrates the most stable behavior, with a consistently high accuracy and the lowest overall loss across epochs. This indicates a good balance between capacity and regularization, confirming it as the best-performing model.

CNN V3, which uses a pre-trained MobileNetV2 with frozen layers, shows lower and flatter curves for both accuracy and loss, suggesting underfitting. The domain difference between

ImageNet and the driving dataset, combined with frozen feature extraction, limits its performance in this task.

These results validate the decision to choose CNN V2 (DeepDriveNet) for further deployment and experimentation.

2.3.4 Developing the Hybrid Approach (Supervised + RL)

In this approach, the model was first trained using supervised learning. The supervised model was used to drive in the simulation environment, and the data collected during this phase—such as frames and logs of the car's actions—was recorded. Afterward, this data was uploaded to Kaggle for offline training.

To enhance the model's driving capabilities, reinforcement learning (PPO) was employed. The segmentation model was used to calculate reward functions based on the car's position—specifically, whether the car was driving on the road or off it. The PPO algorithm was then trained using this data, allowing the agent to improve over time.

The results of this hybrid approach were positive. The PPO-trained agent was able to drive better than the supervised model, particularly on highways, where it exhibited improved control and navigation. However, in more complex city sections, the performance was slightly lower compared to the highway results, as expected due to the variability in urban driving conditions.

2.3.5 Advantages of Combining Supervised Learning with Reinforcement Learning

The supervised model provided a strong baseline, capturing general driving behaviors but lacked adaptability to changing environments and edge cases. On the other hand, reinforcement learning alone was too slow and unstable for real-time driving tasks. Combining both approaches allowed the system to benefit from prior knowledge gained through supervised learning while still enabling the model to improve through exploration in more challenging situations.

2.3.6 Hybrid Implementation Strategy

The hybrid model was structured by using behavioral cloning as the base. The model was initially trained through supervised learning on the data collected during the simulation. Then, reinforcement learning (PPO) was used to fine-tune the model's decision-making abilities. This combination of methods enabled the model to handle edge cases more effectively, improving overall driving behavior and adaptability, particularly in complex scenarios where the supervised model alone struggled.

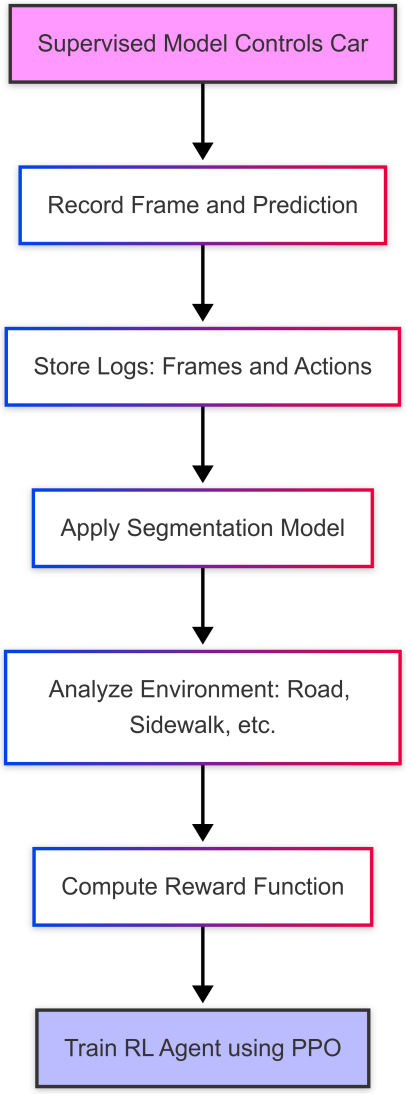


Figure 2.10: Proposed Hybrid Architecture: Supervised Learning + Reinforcement Learning for Autonomous Driving

This diagram illustrates the overall pipeline of the hybrid approach. The process begins with the supervised model controlling the vehicle, during which frames and actions are logged. A segmentation model is then applied to analyze the visual environment, identifying classes like road, sidewalk, and pedestrians. Based on this analysis, a reward function is computed to evaluate the model's performance. These rewards are used to train a reinforcement learning agent (PPO), which refines the driving policy by learning from both successful and suboptimal actions observed during the simulation.

2.4 Conclusion

This chapter detailed the experimental phases of the project, including the exploration of classical and deep learning methods. It justified the transition from simple image processing techniques to supervised learning and finally to a hybrid model. The combination of behavioral cloning and reinforcement learning was found to be the most promising approach to achieve a stable and responsive autonomous driving system in the simulation environment.

Chapter

3 Model Evaluation and Deployment

Contents

3.1 Introduction

This chapter presents the final evaluation and deployment of the two most promising approaches developed in this project: the **Supervised Learning model (CNN)** and the **Hybrid Method**, which combines Supervised Learning with Reinforcement Learning (PPO). After completing their respective training phases, both models were tested in real-time driving scenarios within a simulation environment. This chapter details their performance analysis, reward mechanism, PPO training insights, and how each model was integrated into the final deployment pipeline.

3.2 Testing and Results Analysis

In this section, we evaluate the behavior of the two selected models during simulation testing. These models were chosen due to their strong training results and feasibility for real-time deployment. The analysis focuses on their respective strengths and limitations under different driving conditions.

3.2.1 Supervised Learning Model

The supervised model was built using a Convolutional Neural Network (CNN V2 - DeepDriveNet), trained through behavioral cloning. It predicted one of six predefined driving actions based on input frames.

- Validation accuracy: **76.7%**
- Very fast inference speed (<50ms per frame)
- Robust performance on straight highways and regular roads

However, the model showed limitations in unfamiliar or complex environments, where it often failed to recover after drifting off-road or facing unseen obstacles.

Figure 3.1: Simulation of the car driving using the supervised CNN model

This figure shows example frames from the simulation environment where the supervised CNN model was deployed. Each frame includes both the model's predicted action and the ground truth label. Green labels indicate correct predictions, while red labels highlight mismatches between the predicted and actual driving actions. The model performs well in most cases, particularly in straightforward conditions, but occasionally misclassifies compound actions like "Forward + Right." These mispredictions often arise in complex curves or when multiple motion cues are present in the scene.

3.2.2 Hybrid Model (Supervised + RL)

The hybrid approach used the CNN to collect training data and a PPO agent to refine behavior through reinforcement learning. The agent received rewards based on a segmentation-based interpretation of the environment and learned to optimize decisions for long-term safety and efficiency.

- Better off-road recovery and adaptability
- Improved navigation on highways
- Slightly slower inference compared to the CNN due to PPO's policy computations
- Moderate performance in complex urban environments, with occasional over-corrections

Importantly, the PPO agent no longer used the segmentation model during deployment. It received only grayscale frames and selected actions based on its trained policy.

3.2.3 Reward Function Design

To guide the PPO agent during training, reward signals were calculated using YOLOv8 semantic segmentation outputs. Each pixel was labeled as road, sidewalk, car, or person, allowing the system to assign rewards based on the ego vehicle's position relative to these classes.

Reward logic:

- **+1**: if the car stays on the road (Class ID: 0)
- **-1**: if the car touches the sidewalk (Class ID: 1)
- **-1.5**: if the car overlaps with a pedestrian or another vehicle (Class ID: 2 or 3)

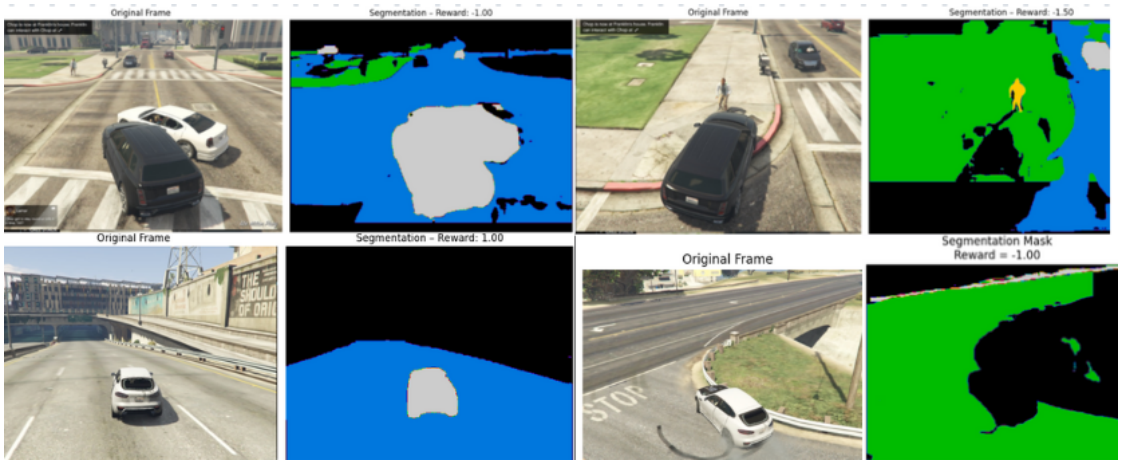


Figure 3.2: Reward signal examples based on segmentation masks and driving context

This figure illustrates how the reward function was applied during reinforcement learning. Each pair of images shows an original simulation frame and its corresponding segmentation mask. The reward value displayed at the top is calculated based on the car's overlap with the segmented classes. For example, the car receives a full reward when driving entirely on the road, a reduced or negative reward when partially or fully on the sidewalk, and a strong penalty when colliding with another vehicle or pedestrian. These dynamic visual cues allowed the PPO agent to learn safe driving behavior by maximizing positive outcomes and avoiding hazardous regions.

3.2.4 Visual Analysis of PPO Training

Key PPO training metrics were monitored to ensure stable and meaningful learning behavior. Below are selected plots:

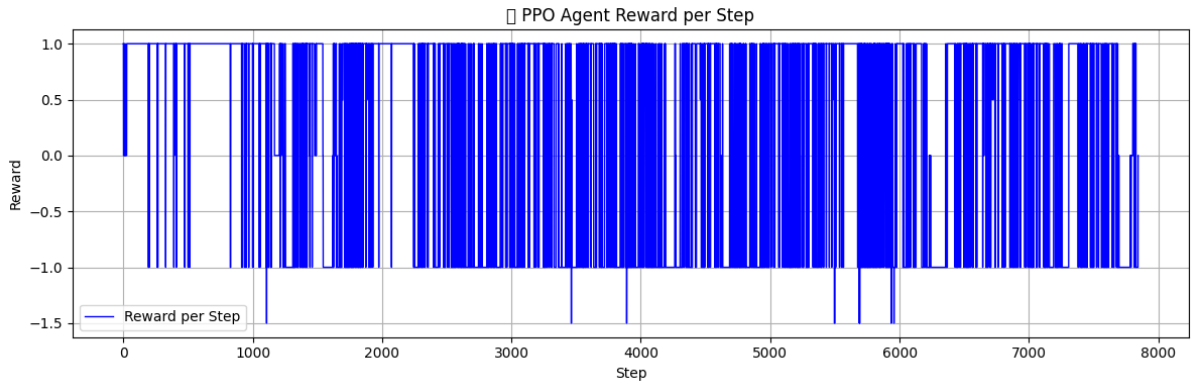


Figure 3.3: Reward per step during PPO training. High-density positive rewards indicate successful learning.

This figure shows the reward received by the PPO agent at each training step. The high concentration of values near +1 indicates that the agent consistently took correct actions that kept the car on the road. Sparse negative rewards reflect rare events such as driving on the sidewalk or colliding with other objects. Overall, the reward trend validates that the policy learned useful behaviors aligned with the task objectives.

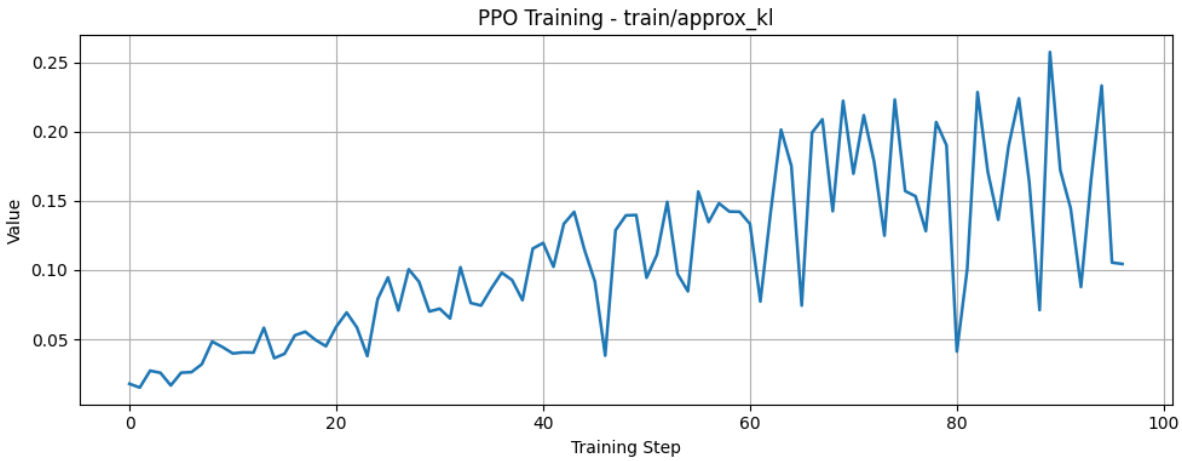


Figure 3.4: KL divergence between policy updates. Remains stable during training, which helps avoid sudden performance drops.

The KL divergence plot tracks how much the updated policy deviates from the previous one at each training step. A stable KL curve means that the PPO algorithm successfully constrained its policy changes, avoiding drastic updates that could destabilize performance. Slight increases are expected as the model explores better actions, but overall, the smoothness of this curve confirms safe policy progression.

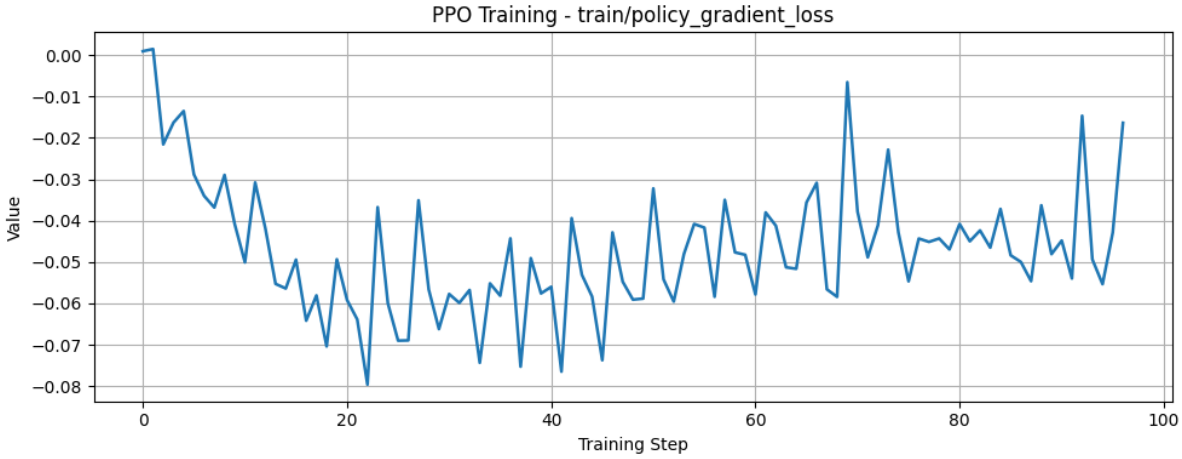


Figure 3.5: Policy gradient loss showing steady convergence. Reflects efficient learning from advantage values.

This figure displays the policy gradient loss during PPO training, which quantifies how much the policy is being adjusted based on calculated advantage values. The decreasing trend followed by convergence suggests that the model learned efficiently from its experiences. Occasional

spikes may represent adaptation to new scenarios, but the overall downward trajectory reflects successful optimization of the policy network.

3.2.5 Performance Comparison

Table 3.1: Comparison between Supervised and Hybrid Approaches

Metric	Supervised Model	Hybrid Model (PPO)
Training Speed	Fast	Slower
Generalization	Moderate	High
Recovery Ability	Weak	Strong
Inference Speed	Very Fast	Fast
Highway Driving	Good	Very Good
City Driving	Weak	Moderate
Adaptability	Low	High

This table summarizes the key differences between the supervised CNN model and the hybrid model that combines supervised learning with reinforcement learning using PPO. Overall, both models demonstrate functional driving capabilities in simulation; however, neither model is perfect.

The supervised model offers faster training and extremely low inference time, making it ideal for structured environments like highways. However, it struggles with recovery in complex or unfamiliar situations and tends to perform poorly in city-like environments where quick adaptations are required.

In contrast, the hybrid model shows stronger generalization and adaptability. It performs better in challenging scenarios, especially in terms of recovering from off-road situations or avoiding obstacles. Still, this comes at the cost of slower inference and training due to the added complexity of reinforcement learning.

In conclusion, both approaches are effective within their respective strengths, but each has its limitations. Combining their advantages or further improving perception and reward strategies could lead to a more robust autonomous driving system in future iterations.

3.3 Deployment Pipeline

Two distinct deployment pipelines were implemented to evaluate the models in real-time simulation: one for the supervised CNN model and one for the reinforcement learning PPO agent.

3.3.1 CNN-Based Deployment (Supervised Learning)

A Python script was used to capture live frames from the GTA V simulation. Each frame was then preprocessed by converting it to grayscale and resizing it to match the input dimensions expected by the CNN model. Once preprocessed, the frame was passed through the CNN, which predicted a corresponding driving action such as “Forward,” “Left,” or “Stop.” This predicted action was translated into actual keyboard key presses using an automation library like `pynput`. Finally, these key presses were used to control the vehicle directly within the simulation, enabling real-time autonomous driving.

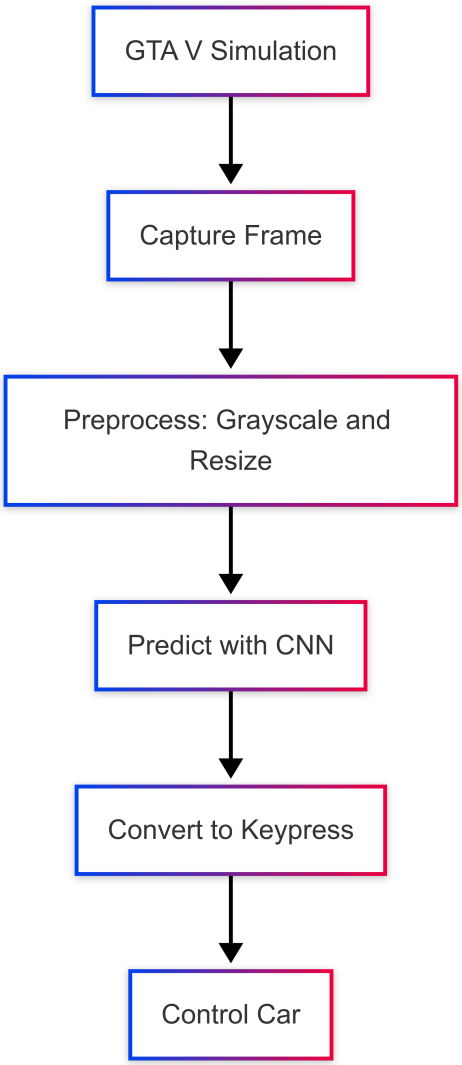


Figure 3.6: Deployment architecture of the CNN-based supervised learning model

This diagram illustrates the real-time deployment pipeline of the supervised CNN model within the GTA V simulation environment. The process begins with capturing each video frame during gameplay. These frames are then preprocessed by converting them to grayscale and resizing them to fit the CNN's expected input format. The CNN model predicts the corresponding driving action (e.g., forward, left, stop), which is subsequently translated into keypress commands using a keyboard emulation library. These keypresses are sent back to the simulator to control the vehicle in real time. The entire loop is executed rapidly, ensuring that the car can respond to new visual input within milliseconds.

3.3.2 PPO-Based Deployment (Hybrid Model)

For the PPO agent, the deployment procedure was slightly different due to its offline training. The PPO model, trained and exported from Kaggle, was loaded locally using a Python script. Similar to the CNN pipeline, live grayscale frames were continuously captured from the simulation environment. These frames were then fed into the PPO agent, which used its policy network to predict the next driving action based on the learned strategy. Finally, the selected action was executed via keyboard emulation, allowing the agent to control the car in GTA V in real time.

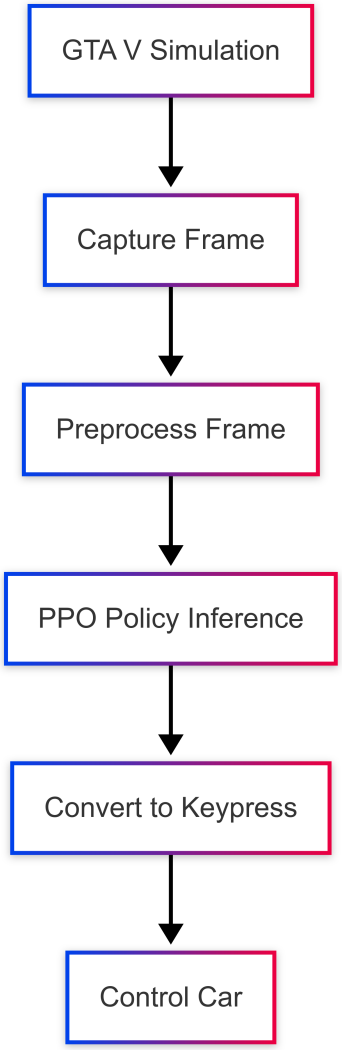


Figure 3.7: Deployment architecture of the PPO-based reinforcement learning model

This diagram represents the deployment pipeline of the PPO-based reinforcement learning model. Like the CNN approach, the system starts by capturing frames from the GTA V simulation.

These frames are preprocessed and passed into the PPO agent, which uses its trained policy network to infer the next optimal driving action. The predicted action is then converted into corresponding keyboard inputs through an emulation script. These inputs are used to control the vehicle in real time, allowing the PPO agent to drive autonomously based on its learned behavior.

Summary:

- The CNN model ensured fast and simple deployment, ideal for well-known driving scenarios.
- The PPO model enabled adaptive driving in more complex environments, leveraging experience-based learning at the cost of higher inference time.

Both systems used the same modular interface for frame capture and control execution, allowing for consistent testing across different learning strategies.

3.4 Future Work

While this project successfully demonstrated real-time deployment using both supervised and hybrid reinforcement learning approaches, several improvements can be made in future iterations to enhance driving performance, safety, and generalization.

3.4.1 Online Reinforcement Learning

Due to hardware limitations, the PPO agent was trained offline and deployed without ongoing learning. In future work, online reinforcement learning could be implemented. This would allow the agent to remain in the learning loop during deployment, adapting continuously based on its own actions and the resulting outcomes. Such an approach could drastically improve the model's ability to handle novel situations and fine-tune its behavior in real time.

3.4.2 Extended Semantic Segmentation

Currently, the segmentation model identifies only four main classes: road, sidewalk, car, and person. To improve decision-making and context awareness, more classes can be added, including:

- Road signs (e.g., stop signs, yield, speed limit)
- Traffic lights (with state recognition: red, green, yellow)
- Lane markings and directional arrows
- Bicycles, motorcycles, and animals

This richer perception would enable more complex behavior such as stopping at red lights, yielding to pedestrians, and respecting road signs.

3.4.3 Integration of Vehicle Dynamics

Another important enhancement involves extracting and integrating vehicle speed and acceleration into the model's decision-making process. This information could be used to:

- Adapt acceleration or braking intensity
- Improve smoothness of driving
- Prevent overshooting or abrupt turns

Combining visual input with real-time telemetry data would allow for more intelligent and human-like driving behavior.

3.4.4 Toward a More Realistic and Scalable Framework

Eventually, transitioning from simulated driving in GTA V to more advanced simulation platforms or even real-world datasets (e.g., KITTI, nuScenes) would bring the model closer to real-world

applicability. Using ROS (Robot Operating System) or real-time simulators like CARLA could allow sensor fusion (e.g., LiDAR + cameras), GPS-based positioning, and richer environmental modeling.

3.5 Conclusion

This chapter demonstrated that the hybrid approach provided improved adaptability and recovery in dynamic driving conditions, while the supervised model maintained faster inference and consistent performance on straightforward paths. The segmentation-based reward system proved effective for guiding the reinforcement learning process. Future work can explore improvements in city driving, richer perception signals, and deployment beyond simulation.

GENERAL CONCLUSION

This final year project focused on the design, implementation, and evaluation of an autonomous driving system within a simulated environment. The primary goal was to develop an artificial intelligence capable of controlling a vehicle in the game Grand Theft Auto V (GTA V), while ensuring realistic and safe behavior through both supervised and reinforcement learning techniques.

Initially, we developed a supervised learning model based on a Convolutional Neural Network (CNN), trained using the behavioral cloning approach. This model demonstrated solid performance on simple roads and familiar environments, delivering fast inference times and reasonably accurate driving action predictions.

However, in order to enhance the system's adaptability to more complex and unpredictable scenarios, a second model was proposed: a hybrid approach combining supervised learning with deep reinforcement learning (PPO). For this, we trained a PPO agent offline using a reward system based on semantic segmentation of the driving scene, leveraging YOLOv8 to classify critical elements such as roads, sidewalks, vehicles, and pedestrians. This allowed the agent to interpret its surroundings and make optimized decisions aimed at safety and efficiency.

Both models were then deployed and tested in real-time within the simulation. The CNN model stood out for its speed and simplicity, making it suitable for well-defined situations. The PPO agent, on the other hand, exhibited stronger recovery capabilities and greater adaptability, particularly in unfamiliar or complex environments. We implemented two distinct deployment pipelines tailored to each model, handling frame capture, preprocessing, inference, and game control.

Despite the encouraging results, several limitations were encountered — including hardware constraints, dataset size, and the absence of online training. These factors led to a number of

promising future directions: enabling real-time PPO training during deployment, expanding the segmentation model to include additional road elements (such as traffic signs, signals, and lane markings), and integrating vehicle dynamics (speed, acceleration) for more realistic control.

In conclusion, this project demonstrated how modern AI techniques can be effectively combined to build intelligent driving systems in a rich simulation environment. It lays a strong foundation for future work in autonomous driving, offering multiple paths toward more advanced and real-world-ready solutions.

WEBOGRAPHY

[1] **John Schulman et al.** Proximal Policy Optimization Algorithms **[online]**. July 2017. Available at: https://arxiv.org/abs/1707.06347

[2] **Eric Yu**. PPO-for-Beginners: A simple and well-styled implementation of Proximal Policy Optimization (PPO) using PyTorch **[online]**. GitHub repository. Available at: https://github.com/ericyangyu/PPO-for-Beginners

[3] **OpenAI**. Proximal Policy Optimization **[online]**. Spinning Up documentation. Available at: https://spinningup.openai.com/en/latest/algorithms/ppo.html

[4] **Vincent Moens**. Reinforcement Learning (PPO) with TorchRL Tutorial **[online]**. PyTorch Tutorials. March 2023. Available at: https://pytorch.org/tutorials/intermediate/reinforcement_ppo.html

[5] **DataCamp**. Proximal Policy Optimization with PyTorch and Gymnasium **[online]**. Available at: https://www.datacamp.com/tutorial/proximal-policy-optimization

[6] **ICLR Blog Track**. The 37 Implementation Details of Proximal Policy Optimization **[online]**. March 2022. Available at: https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/

[7] **Nikhil Barhate**. PPO-PyTorch: Minimal implementation of Proximal Policy Optimization **[online]**. GitHub repository. Available at: https://github.com/nikhilbarhate99/PPO-PyTorch

[8] **Hugging Face**. Proximal Policy Optimization (PPO) **[online]**. Available at: https://huggingface.co/blog/deep-rl-ppo

[9] **Papers-100-Lines**. Reinforcement Learning: PPO in Just 100 Lines of Code **[online]**. Available at: https://papers-100-lines.medium.com/reinforcement-learning-ppo-in-just-100-lines-of-code-1f002830cff4

[10] **Mariusz Bojarski et al.** End to End Learning for Self-Driving Cars **[online]**. NVIDIA. Available at: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

[11] **Madhu Dev**. Guiding Self-Driving Car Using Behavioral Cloning **[online]**. Medium. October 2021. Available at: https://medium.com/@madhusudhan.d/guiding-self-driving-car-using-behavioral-cloning-9c24541a425d

[12] **Alex Staravoitau**. Behavioral Cloning: End-to-End Learning for Self-Driving Cars **[online]**. GitHub repository. Available at: https://github.com/alexstaravoitau/behavioral-cloning

[13] **Sentdex**. Self-Driving Car AI — Deep Learning and Behavioral Cloning in GTA V **[online]**. YouTube Playlist. Available at: https://www.youtube.com/playlist?list=PLQVvvaa0QuDfSfqQuee6K8opKtZsh7sA9

[14] **Sentdex**. Python Plays GTA V — Self-Driving Car Deep Learning Project **[online]**. GitHub Repository. Available at: https://github.com/Sentdex/pygta5

[15] **KPIT**. Driver Behavior Cloning for Self-driving Vehicles **[online]**. Available at: https://www.kpit.com/workimpact/behavior-cloning-using-deep-learning/

[16] **ResearchGate**. Towards Autonomous Driving System Using Behavioral Cloning Approach **[online]**. Available at: https://www.researchgate.net/publication/386077871_Towards_Autonomous_Driving_System_Using_Behavioral_Cloning_Approach

[17] **Kathan Sheth**. Behavior-Cloning-Self-Driving-Car-using-CNN **[online]**. GitHub repository. Available at: https://github.com/KathanSheth/Behavior-Cloning-Self-Driving-Car-using-CNN

[18] **Ultralytics**. YOLOv8 Documentation **[online]**. Available at: https://docs.ultralytics.com/

[19] **Roboflow Blog**. What is YOLOv8? A Complete Guide **[online]**. Available at: https://blog.roboflow.com/what-is-yolov8/

[20] **Ultralytics**. ultralytics/ultralytics **[online]**. GitHub repository. Available at: https://github.com/ultralytics/ultralytics

[21] **DagsHub**. Welcome to Ultralytics YOLOv8 **[online]**. Available at: https://dagshub.com/Ultralytics/ultralytics/src/c985eaba0d928d96f2bcc7374dd830d6c885f16f/docs/index.md

[22] **Scientific Reports**. YOLOv8-seg-CP: a lightweight instance segmentation algorithm for chip pad detection **[online]**. Available at: https://www.nature.com/articles/s41598-024-78578-x

[23] **Richter, Stephan R., Vineet, Vibhav, Roth, Stefan, and Koltun, Vladlen.** *Playing for Data: Ground Truth from Computer Games*. TU Darmstadt Visual Inference Lab **[online]**. Available at: https://download.visinf.tu-darmstadt.de/data/from_games/

[24] **Car Throttle**. Researchers Are Using GTA V For Autonomous Car Development **[online]**. Available at: https://www.carthrottle.com/news/researchers-are-using-gta-v-autonomous-car-development-and-its-not-ridiculous-it-sounds

[25] **World Health Organization**. Global Status Report on Road Safety 2023 **[online]**. Available at: https://www.who.int/publications/i/item/9789240086517

[26] **Allied Market Research**. Autonomous Vehicle Market **[online]**. Available at: https://www.alliedmarketresearch.com/autonomous-vehicle-market

End of The Year Project

Self-Driving Car Project Using a Simulation Environment

Prepared by: **Bali Rassem**

Abstract:

This project presents the development of an autonomous driving system in a simulated environment (GTA V), combining supervised learning (behavioral cloning) and reinforcement learning (PPO). Semantic segmentation is used to generate reward signals based on image masks. The system's performance is analyzed across realistic driving scenarios.

Keywords: autonomous driving, supervised learning, reinforcement learning, segmentation, GTA V simulation.

Résumé :

Ce projet présente le développement d'un système de conduite autonome dans un environnement simulé (GTA V), combinant l'apprentissage supervisé (clonage comportemental) et l'apprentissage par renforcement (PPO). La segmentation sémantique est utilisée pour générer des récompenses à partir de masques d'images. Les performances sont analysées dans des scénarios de conduite réalistes.

Mots-clés : conduite autonome, apprentissage supervisé, apprentissage par renforcement, segmentation, simulation GTA V.