UNIVERSITE HASSIBA BENBOUALI DE CHLEF

Faculté de Technologie

Département d'Electronique

MEMOIRE DE MASTER

Domaine : SCIENCES ET TECHNOLOGIES

Filière : : ELECTRONIQUE

Spécialité : ELECTRONIQUE DES SYSTEMES EMBARQUES

Approches de Deep Learning pour la segmentation d'image en temps réel dans la conduite autonome.

Par

Djihane SLIMANE

Hind BOUFADES

Encadreurs :

M. BAHI AZZOUOUM Ahmed

MCB à l'UHBC

M. BENYAHIA Ahmed

MRB à CRTI Alger

Chlef, Juin 2025

UNIVERSITY OF HASSIBA BENBOUALI - CHLEF

Faculty of technology

Department of Electronic

MASTER'S Dissertation

Domain : SCIENCE AND TECHNOLOGY

Field : : ELECTRONIC

Specialization : ELECTRONIC EMBEDDED SYSTEMS

Deep Learning Approaches for real time Image Segmentation in Autonomous Driving.

By

Djihane SLIMANE

Hind BOUFADES

Supervisors:

M. BAHI AZZOUOUM Ahmed

MCB at UHBC

M. BENYAHIA Ahmed

MRB at CRTI Algiers

Chlef, June 2025

Acknowledgments

First and foremost, we express our deepest gratitude to Allah, the One and Most Merciful, for granting us the strength, health, and patience to complete this modest work.

We would like to extend our sincere thanks to our supervisor**,** Dr. BAHI AZZOUOUM Ahmed, and our co-supervisor Dr BENYAHIA Ahmed for their constant availability, insightful guidance, and valuable contributions throughout every stage of this dissertation. Their support and understanding were truly appreciated.

Our gratitude also goes to the Research Centre of Industrial Technologies (CRTI) for welcoming us during our internship and providing a supportive environment for learning and growth.

We warmly thank the members of the jury for accepting to evaluate this work, and for their time, attention, and valuable remarks.

Finally, we express our heartfelt appreciation to all the professors of the Electronics Department for their dedication and the knowledge they have shared with us over the years.

Thank you all.

Dedication

I humbly dedicate this modest work to:

To the memory of my beloved father, whose absence has never dimmed his presence in my heart. May Allah grant him eternal peace and mercy.

To my great mother, the most extraordinary woman I know, your strength, your prayers, your endless love and sacrifices are the pillars of everything I am today. You are not just my mother, you are my guiding light, my quiet force and eternal blessing.

To my dear sisters, Haifaa and Raghda, and my one and only brother Khalil, your encouragement, presence and support have carried me through more than you know.

To my little nephew Souhail, whose innocence and smile brought warmth to even the hardest days.

To all my teachers and mentors, especially those who believed in me when I doubted myself, and encouraged me when I needed it most. Your words, your time, and your faith in me have left a mark I will carry forever.

My precious friends: Bekhti Nor El Houda, Boufades Hind, Mekrelouf Affaf, Messafa Ikram, Bouaaza Boutheina thank you for always being there.

And to the friends I made during Erasmus: Bernardo Lança, Nursena Akgul, Aleksei Khozhaev, Pitchaya Opassathian, Gabriel Santos and others thank you for making this journey even more special and memorable.

May Allah, the Almighty and the Most Merciful, bless you all with good health, joy, and long, fulfilling lives.

With all my heart - Thank you.

SLIMANE Djihane

Dedication

To my dearest parents,

Whose endless love, sacrifices, and prayers have been the light guiding me through every challenge I owe this journey to you.

To my wonderful sisters Fatima Elzahraa, Fatiha, Djemia, Asmaa, Sara, and Nourehane.

Each of you has been a source of strength, laughter, and inspiration. Your constant presence in my life has made even the hardest days easier.

To my brother, Mohammed Elaamine,

Your belief in me has always pushed me forward. Thank you for being my quiet supporter and my greatest motivator.

To all my nieces and nephews,

Your presence and pure love have been a source of joy and comfort throughout this journey.

To my precious friends Bouriche Amira, Messafa Ikram, Bouaaza Boutheina, Slimane Djihane and Mekrelouf Affaf.

Thank you for your encouragement, your faith in me, and your ability to lift me up when I needed it most. You've been more than friends you've been family.

To all my classmates,

With whom I shared unforgettable moments of learning, growth, and friendship.

And to all the other people who supported me, even if I may have forgotten to mention you by name know that your kindness is not forgotten, and your presence truly made a difference.

This work is dedicated to all of you with love, gratitude, and all my heart.

BOUFADES Hind

Abstract

Real-time semantic segmentation is a key task in computer vision, especially for autonomous driving systems where quick and accurate perception is essential. This thesis explores deep learning techniques for this task using the U-Net architecture, applied to the Cityscapes dataset of urban driving scenes.

To enhance performance, we replaced the original U-Net encoder with EfficientNet-B0, aiming to achieve both high segmentation accuracy and real-time inference speed. The proposed model reached over 61.8% mean IoU and 128 FPS, making it highly suitable for real-time applications.

The work was developed using Google Colab and benefited from an internship at CRTI, which provided technical support and valuable resources. Despite certain challenges such as limited annotation in the test set, the final results demonstrate the model's efficiency and robustness.

Keywords: Deep learning, semantic segmentation, U-Net, EfficientNet-B0, Cityscapes, autonomous driving, real-time, mIoU, FPS.

Résumé

La segmentation sémantique en temps réel est une tâche essentielle en vision par ordinateur, notamment pour les systèmes de conduite autonome où la perception rapide et précise est cruciale. Ce mémoire explore les techniques d'apprentissage profond pour cette tâche en utilisant l'architecture U-Net, appliquée au jeu de données urbain Cityscapes.

Afin d'améliorer les performances, nous avons remplacé l'encodeur d'origine de U-Net par EfficientNet-B0, dans le but d'obtenir à la fois une segmentation précise et une vitesse d'inférence en temps réel. Le modèle proposé a atteint plus de 61.8 % de mIoU et 128 FPS, ce qui le rend parfaitement adapté aux applications temps réel.

Le travail a été réalisé sur Google Colab et a bénéficié d'un stage au CRTI, qui a fourni un soutien technique et des ressources précieuses. Malgré certaines limitations, comme l'absence d'annotations dans l'ensemble de test, les résultats finaux démontrent l'efficacité et la robustesse du modèle.

Mots-clés : Apprentissage profond, segmentation sémantique, U-Net, EfficientNet-B0, Cityscapes, conduite autonome, temps réel, mIoU, FPS.

ملخص

تُعدّ التجزئة الدلالية في الوقت الحقيقي من المهام الأساسية في رؤية الحاسوب، خاصةً في أنظمة القيادة الذاتية حيث تُعدّ سرعة ودقة الإدراك أمرًا بالغ الأهمية. يتناول هذا البحث تقنيات التعلم العميق لهذه المهمة باستخدام نموذج U-Net، وذلك على مجموعة بيانات Cityscapes الخاصة بمشاهد القيادة الحضرية.

ولتحسين الأداء، تم استبدال وحدة التشفير الأصلية في U-Net بالنموذج المحسن EfficientNet-B0، بهدف تحقيق دقة عالية في التجزئة وسرعة استدلال مناسبة للتطبيقات في الوقت الحقيقي. وقد حقق النموذج المقترح أكثر من 61.8٪متوسط تداخل الاتحاد mIoU و128 إطار في الثانية (FPS)، مما يجعله مناسبًا للغاية لمشاريع القيادة الذاتية في الزمن الحقيقي.

تم تنفيذ العمل باستخدام Google Colab، واستُفيد من فترة تدريب قصيرة بمركز البحث في التكنولوجيات الصناعية CRTI، مما أتاح دعمًا تقنيًا وموارد مهمة. وعلى الرغم من بعض التحديات، مثل نقص التعليقات التوضيحية في مجموعة الاختبار، تُظهر النتائج النهائية كفاءة النموذج وموثوقيته.

الكلمات المفتاحية: التعلم العميق، التجزئة الدلالية، U-Net، EfficientNet-B0، Cityscapes، القيادة الذاتية، الزمن الحقيقي، mIoU، FPS.

Table of Contents

CHAPTER II: Deep Learning

CHPTER III: Implementation And Experimental Results

List of figures

Chapter I: Image Segmentation In The Context Of Autonomous Vehicles

Chapter II: Deep Learning

List of tables

List of abbreviations

AVS : Autonomous Vehicles

AI : Artificial Intelligence

DL : Deep Learning

ML : Machine Learning

AMRs : Autonomous mobile robots

DARPA: Defense Advanced Research Projects Agency

SAE : Society of Automotive Engineers

LiDAR : Light Detection and Ranging

IMU : Inertial Measurement Unit

V2V : Vehicle-to-vehicle

ROI : Region of Interest

ANNs : Artificial neural networks

CNNs : Convolutional neural networks

FNN : Feedforward Neural Network

RNN : Recurrent Neural Network

LSTM : Long Short-Term Memory

GAN : Generative Adversarial Network

SNN : Spiking Neural Network

FCNs : Fully Convolutional Networks

NIST : National Institute of Standards and Technology

CPU : Central Processing Unit

GPU : Graphics Processing Unit

ReLU : Rectified Linear Unit

MLPs : Multilayer perceptron’s

MSE : Mean Squared Error

MAE : Mean Absolute Error

CCE : Categorical Cross-Entropy

SGD : Stochastic gradient descent

CRF : Conditional Random Field

ASPP : Atrous Spatial Pyramid Pooling

FPN : Feature Pyramid Networks

GD : Gradient descent

LR : Learning rate

Adam : Adaptive Moment Estimation

SGD : Stochastic Gradient Descent

PA : Pixel accuracy

BN : Batch Normalization

TL : Transfer learning

FPS : Frames Per Second

mIoU : mean Intersection over Union

Opencv : Open-source Computer Vision

GENERAL INTRODUCTION

The development of autonomous vehicles (AVs) represents one of the most transformative and technically demanding challenges in modern engineering and computer science. As intelligent transportation systems evolve, AVs are expected to navigate complex urban environments safely and efficiently, relying on minimal human intervention. This paradigm shift is being driven by rapid advancements in artificial intelligence (AI), embedded systems, and multi-modal sensor technologies, all of which converge to power the autonomous driving pipeline.

AVs operate through a continuous, closed-loop system composed of four tightly integrated modules: sensing, perception, path planning, and control and actuation. This pipeline must function under stringent real-time constraints, where any delay can compromise safety and performance. At the core of this system lies the perception module, responsible for understanding the driving environment by interpreting data from various sensors, including LiDAR, radar, GPS, and high-resolution cameras. Among these, cameras provide critical visual information required for detecting road boundaries, lane markings, vehicles, pedestrians, and other environmental features.

A cornerstone of the perception module is semantic segmentation, a dense prediction task that involves assigning a semantic class label to every pixel in an image. This pixel-level understanding enables AVs to distinguish between drivable and non-drivable areas, identify dynamic obstacles, and support safe decision-making in real-time traffic scenarios. As such, semantic segmentation is indispensable for urban scene understanding in autonomous navigation.

Modern segmentation models rely heavily on deep learning, particularly Convolutional Neural Networks (CNNs), which have demonstrated outstanding performance in extracting spatial hierarchies of features. Among these, U-Net has emerged as a widely adopted architecture due to its symmetric encoder-decoder structure, which allows it to capture both fine-grained spatial details and high-level semantic context a crucial requirement for semantic segmentation in complex road scenes.

In this thesis, we develop a semantic segmentation pipeline based on the U-Net architecture, trained and evaluated on the Cityscapes dataset, a benchmark dataset composed of high-resolution, finely annotated images of urban driving environments. The Cityscapes dataset is particularly suitable for this task as it reflects the challenges and variability of real-world cityscapes.

To improve feature extraction capabilities while maintaining computational efficiency, the U-Net model is enhanced with a high-performance encoder. The architecture is designed with a clear focus on achieving an optimal balance between segmentation accuracy and computational cost, aiming for seamless integration into embedded automotive systems.

Real-time deployment in AVs requires models not only to be accurate but also computationally efficient. To this end, the segmentation model is optimized for GPU acceleration to ensure high-speed inference. Moreover, it is designed to function effectively within embedded systems, which are standard in modern vehicles. These systems must handle high-frequency data streams and provide rapid responses, often communicated through Controller Area Network (CAN) buses, which serve as the real-time communication backbone in automotive control architectures. The output from the segmentation module feeds directly into the path planning system, influencing steering, acceleration, and braking decisions.

The structure of the proposed work is outlined in the following chapters:

Chapter 1 introduces the key concepts in autonomous driving, with a particular focus on the functional modules of AVs and the significance of semantic segmentation within the perception stack.

Chapter 2 presents the theoretical background of deep learning and CNN-based segmentation architectures, including encoder-decoder models, training methodologies, and evaluation metrics such as mean Intersection over Union (mIoU) and Dice coefficient.

Chapter 3 details the practical implementation of the proposed model. It includes data preparation using the Cityscapes dataset, model architecture design, training strategy, and performance evaluation in terms of accuracy, inference speed, and real-time viability on GPU-enabled embedded systems.

We end our study with a general conclusion.

In summary, this work presents an efficient and deployable semantic segmentation pipeline tailored for real-world autonomous driving applications. By leveraging the capabilities of deep learning, the U-Net architecture, and the Cityscapes dataset, the system demonstrates the potential to meet the stringent performance demands of embedded automotive platforms operating in dynamic and time-sensitive environments.

Chapter I

Image Segmentation In The Context Of Autonomous Vehicles

I.1 Introduction

The world is currently experiencing rapid technological advancements that are reshaping various industries. Among the most significant of these innovations is the emergence of autonomous driving, a field that relies heavily on recent breakthroughs in Artificial Intelligence (AI) and particularly Deep Learning.

These technologies have enabled machines to perform complex tasks such as perception, decision-making, and control tasks that are essential for self-driving vehicles to operate safely and efficiently. Autonomous vehicles (AVs) once considered a futuristic dream are now a tangible reality being developed and tested by major companies such as Tesla, Waymo, and Baidu. These vehicles aim to reduce human intervention, increase road safety and revolutionize personal and public transportation.

I.2 Autonomous Vehicles (AVs)

I.2.1 Definition

Autonomous vehicles, commonly known as self-driving cars, are designed to operate independently without human control. They rely on a combination of advanced sensors, intelligent software, and real-time data processing to perceive their surroundings and make driving decisions. Enabled by recent progress in artificial intelligence and sensor technology, these vehicles aim to enhance road safety, reduce traffic congestion, and offer more efficient mobility solutions, figure (I.1), [1].

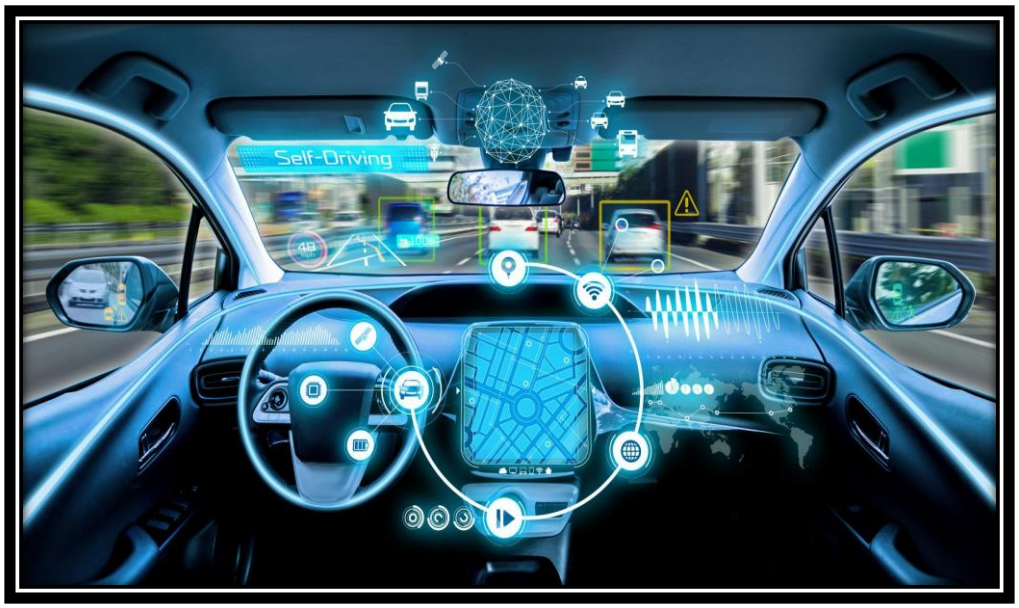


Figure (I.1): A self-driving car.

I.2.2 Types of Autonomous Vehicles

Autonomous vehicles can be classified based on the environments in which they are designed to operate. This categorization helps clarify their specific functions and limitations, as performance is often closely tied to the operational context. Whether navigating on land, in the air, or underwater, each type of autonomous system requires tailored technologies and strategies. In this thesis, our focus is limited to autonomous ground vehicles**,** with particular attention to self-driving carsused in road transportation. The following table presents an overview of the main categories of autonomous vehicles [2].

Category	**Operating Environment**	**Examples**
Ground Vehicles	Ground	Self-driving cars Autonomous trucks Agricultural vehicles Autonomous mobile robots (AMRs)
Aerial Vehicles	Sky	Drones Unmanned aerial vehicles Aerial surveillance systems
Surface Vehicles	Sea Surface	Autonomous boats Unmanned surface vessels Surface research vehicles
Underwater Vehicles	Underwater	Autonomous underwater Vehicles Underwater drones

Table (I.1): Categorization of Autonomous Vehicles by Operating Environment [2].

I.2.3 Historical Background of Autonomous Driving

I.2.3.1 Early Developments (1960s–1970s)

The journey towards autonomous vehicles began with foundational projects like the Stanford Cart in the 1960s [3] and Japan's Tsukuba Mechanical Engineering Lab's line-following vehicle in the 1970s [4].These early experiments laid the groundwork for future advancements in vehicle automation, figure (I.2).

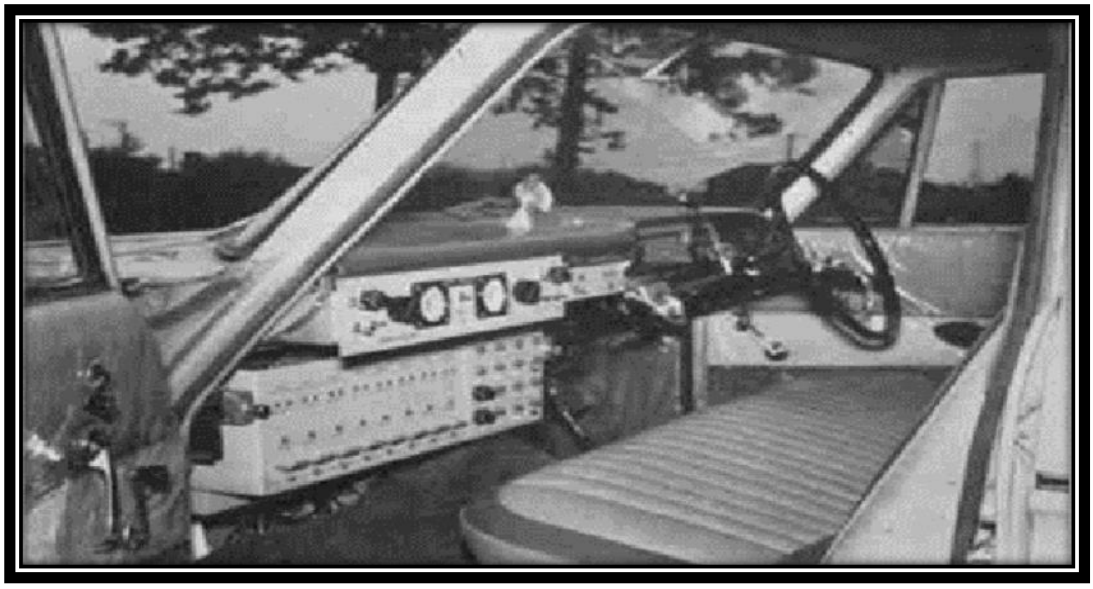


Figure (I.2): Tsukuba Mechanical Engineering driverless car [4].

I.2.3.2 VaMoRs Project (1980s)

In the 1980s, figure (I.3), Ernst Dickmanns and his team at Bundeswehr University Munich in collaboration with Mercedes-Benz, developed the VaMoRs project. This initiative led to a modified Mercedes van capable of semi-autonomous driving on German autobahns using cameras and sensors for navigation [5].

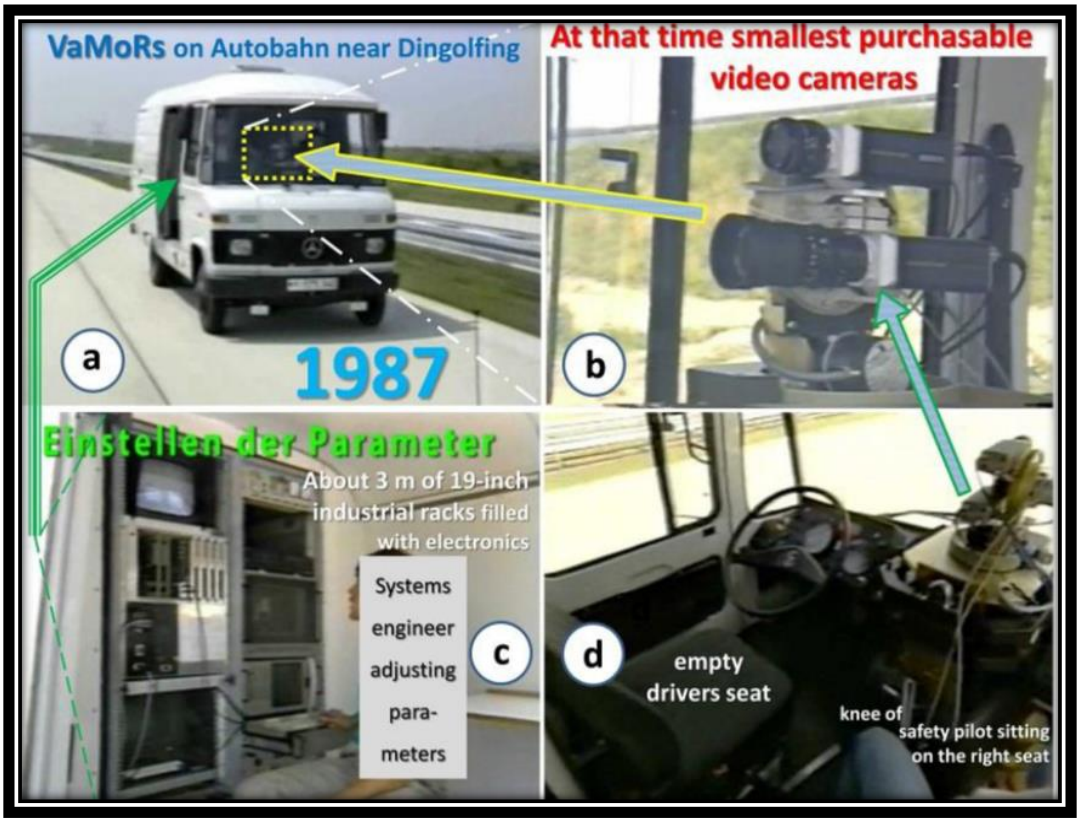


Figure (I.3): Ernst Dickmanns' VaMoRs Mercedes Van (1987) [5].

I.2.3.3 DARPA Grand Challenge (2000s)

The early 2000s saw significant progress with the DARPA Grand Challenge, figure (I.4), a series of competitions organized by the U.S Defense Advanced Research Projects Agency. In 2005, Stanford University's autonomous vehicle, "Stanley," won the challenge by successfully navigating a 132-mile desert course without human intervention [6].

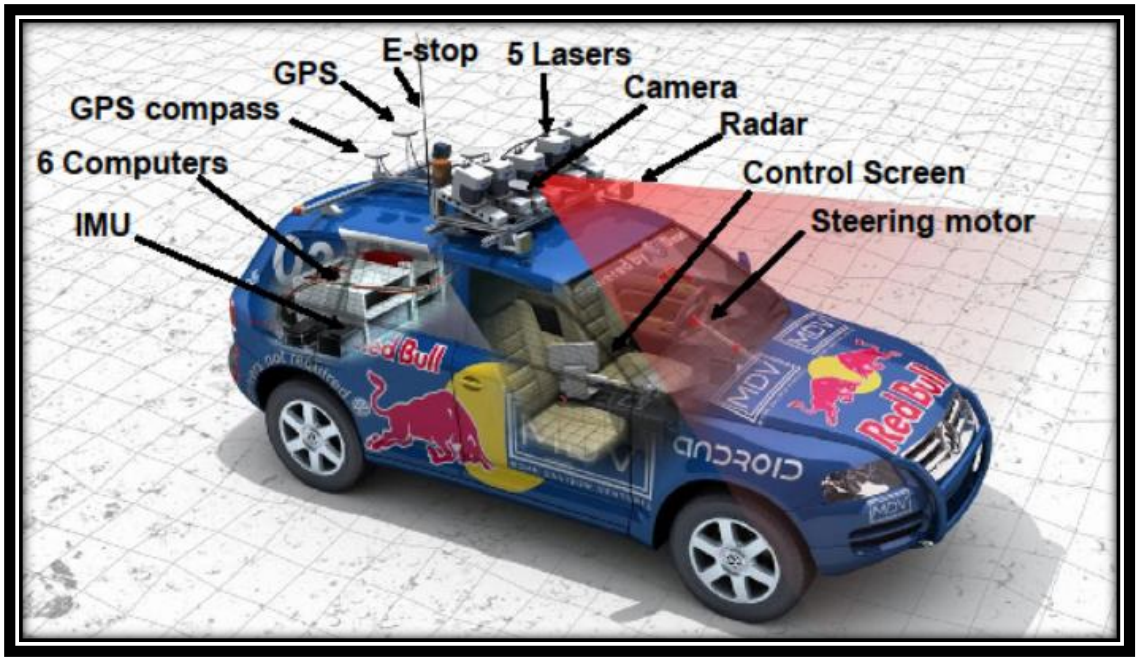


Figure (I.4): Stanford's "Stanley" Vehicle - DARPA Grand Challenge Winner [7].

I.2.3.4 Google's Self-Driving Car Project - Waymo (2010s–Present)

In 2009, Google initiated its self-driving car project, figure (I.5), which later evolved into Waymo. It has since become a leader in autonomous vehicle technology, operating fully autonomous ride-hailing services in cities like Phoenix, Arizona [8].

Figure (I.5): Waymo's Autonomous Vehicle Operating in Phoenix, Arizona.

I.3 Levels of Autonomy

Autonomous vehicles are classified into six levels of automation, ranging from Level 0 (no automation) to Level 5 (full automation), figure (I.6). These levels, defined by the Society of Automotive Engineers (SAE), serve as a standard for understanding the capabilities of autonomous systems in vehicles [9]:

- **Level 0**: No automation. The driver is fully responsible for controlling the vehicle at all times.
- **Level 1**: Driver assistance. The vehicle can assist with specific tasks, such as cruise control or steering, but the driver remains responsible.
- **Level 2**: Partial automation. The vehicle can control both steering and acceleration, but the driver must remain engaged and monitor the driving environment.
- **Level 3**: Conditional automation. The vehicle can handle most driving tasks, but a human driver must be ready to take over if needed.
- **Level 4**: High automation. The vehicle can operate autonomously within certain conditions or environments (e.g: within a city or on a highway) without human intervention.
- **Level 5**: Full automation. The vehicle can operate autonomously in any environment or situation without any need for human input.

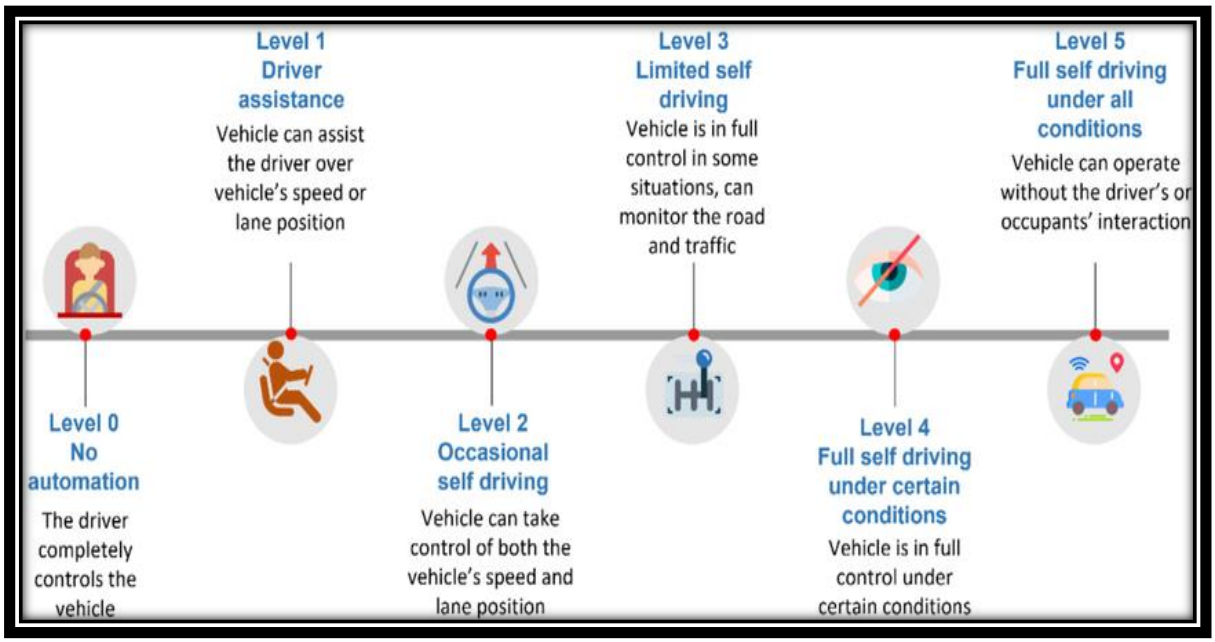


Figure (I.6): Levels of autonomy [10].

I.4 Functional Components of AVs

The operation of an autonomous vehicle depends on a complex integration of several key technologies, figure (I.7), [2]:

- **Sensors**: AVs rely on a wide array of sensors to collect data about their surroundings. The next table summarizes the main sensors used in autonomous vehicles and highlights their respective roles in the perception system [11]:

Sensor	Role
Cameras	Provide visual data that helps detect road signs, pedestrians, traffic signals, and other vehicles
LiDAR (Light Detection and Ranging)	A laser-based sensor that creates detailed 3D maps of the environment, allowing the vehicle to detect objects and obstacles with high precision.
Radar	Used to detect objects at longer distances, even in low-visibility conditions (e.g: fog or rain).
GPS and IMU (Inertial Measurement Unit)	Provide critical information on the vehicle's location, speed, and orientation to help navigate and localize the car within a map.

Table (I.2): Sensors in Autonomous Driving and Their Purposes.

- **Software and Algorithms**: Once data is gathered from these sensors, sophisticated algorithms process and interpret this information. These algorithms enable the vehicle to make decisions, such as determining the best path, identifying potential hazards, and adjusting speed accordingly [12].

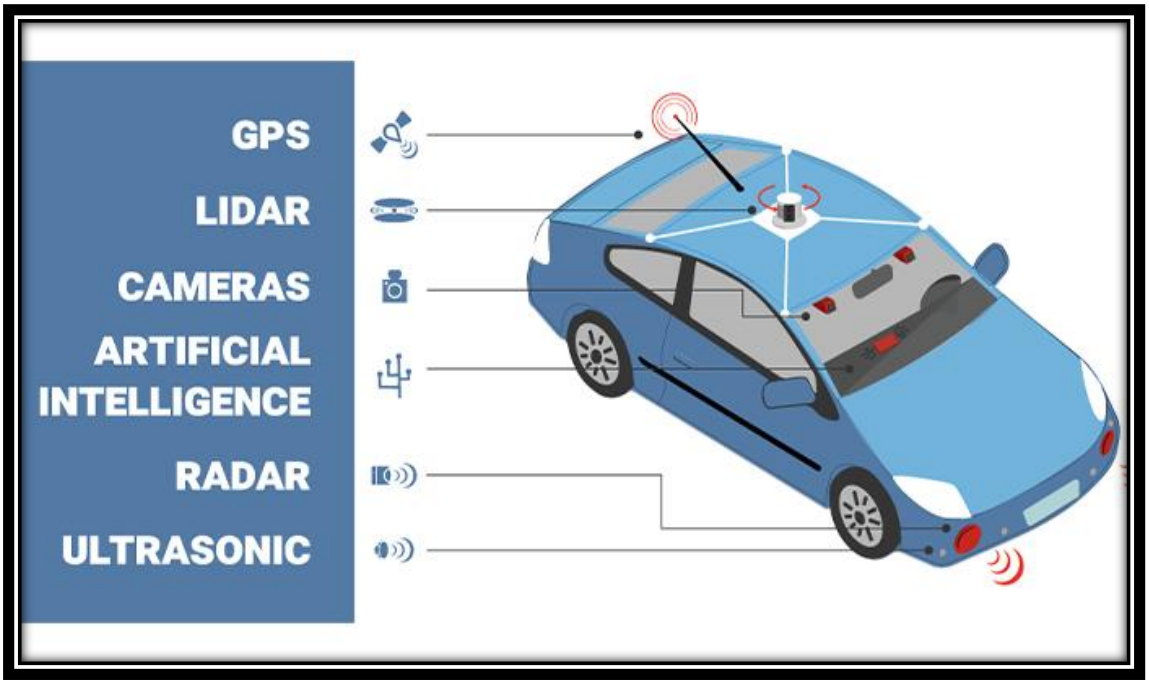


Figure (I.7): The Core Components of Autonomous Vehicles.

I.5 Core Architecture of an Autonomous Driving System

Autonomous driving systems rely on a tightly integrated architecture that bridges hardware and software components to perceive the environment, make decisions, and execute actions. The process begins with data collection from various sensors and vehicle-to-vehicle (V2V) communication [12], which provide continuous input about the driving environment. This raw data is processed in the perception module, where the vehicle builds an understanding of its surroundings and estimates its precise location, figure (I.8).

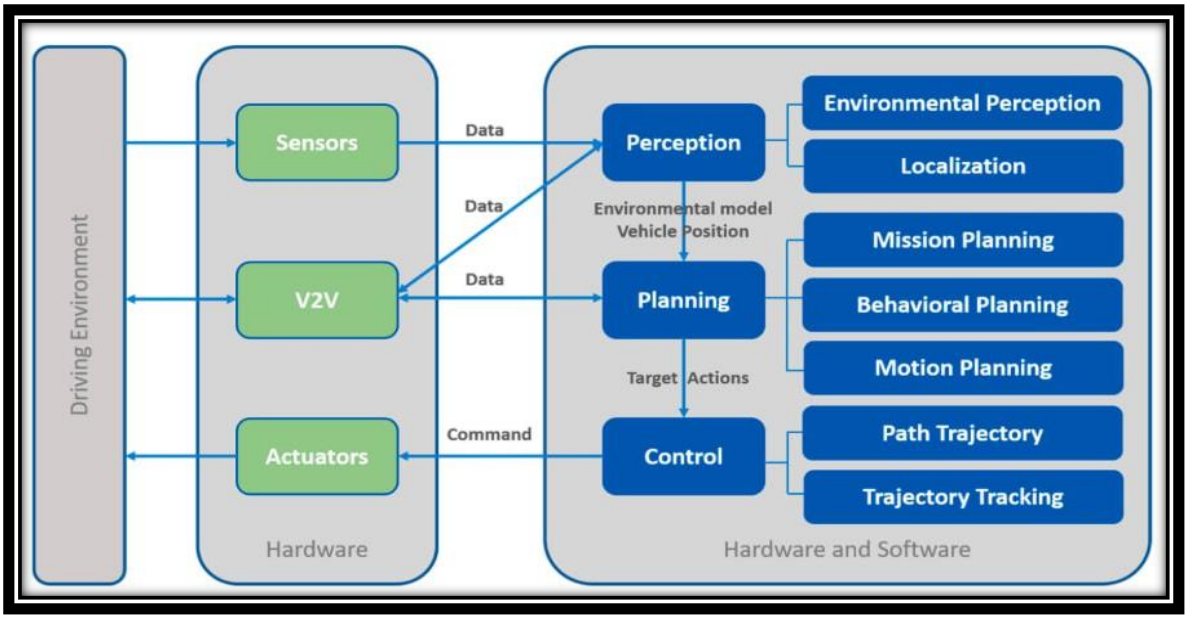


Figure (I.8): The Core Architecture of an Autonomous Vehicles [13].

The resulting environmental model and vehicle position are passed to the planning module, which handles several layers of decision-making, including mission planning, behavioral responses, and motion strategies. Once a path is determined, the control module translates these decisions into specific commands sent to the vehicle's actuators, ensuring accurate trajectory tracking and real-time responsiveness. This architecture ensures a seamless flow of information and actions, allowing the vehicle to operate autonomously and safely in dynamic environments [13].

I.6 Simulation and Scenario Generation

Simulation plays a critical role in the development and validation of autonomous driving systems, figure (I.9). It enables the creation of realistic, controlled virtual environments where vehicles can be tested under a wide variety of conditions and scenarios without physical risks. This process typically begins with designing detailed maps, roads, and buildings, followed by the configuration of elements like weather, lighting, and traffic conditions. Virtual sensors are initialized to mimic real-world perception systems, and driving scenarios are generated by introducing pedestrians, vehicles, and specific behavioral rules. Algorithms are then integrated into the simulation loop to test their decision-making capabilities. Through benchmark and diversity testing, the system's performance, safety, and adaptability can be thoroughly evaluated before real-world deployment. This environment accelerates development and ensures that the vehicle’s software is robust, reliable, and ready to handle complex situations.

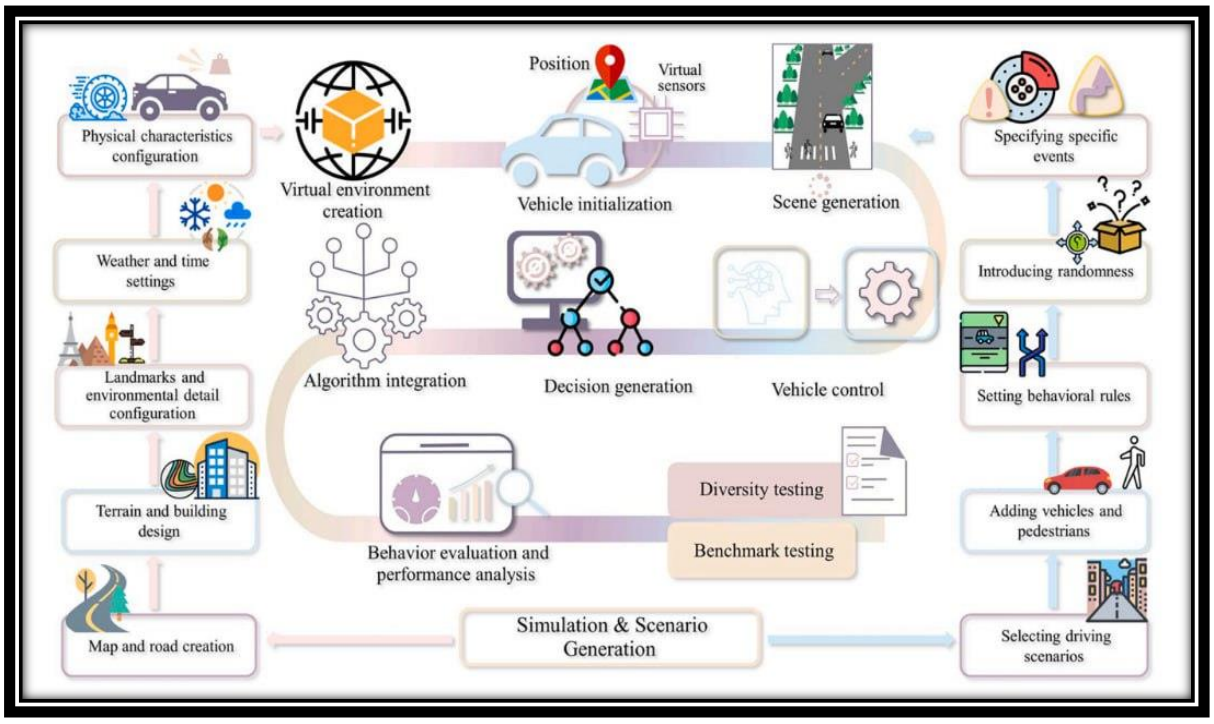


Figure (I.9): Simulation and Scenario Generation [13].

I.7 Stages of Autonomous Driving Powered by Deep Learning

The intelligence of an autonomous vehicle lies in its ability to perceive the environment, plan its path, and control its actions all in real time. These capabilities are typically structured into three main stages: **Perception**, **Planning**, and **Control**, each of which can be enhanced using deep learning techniques, figure (I.10), [11].

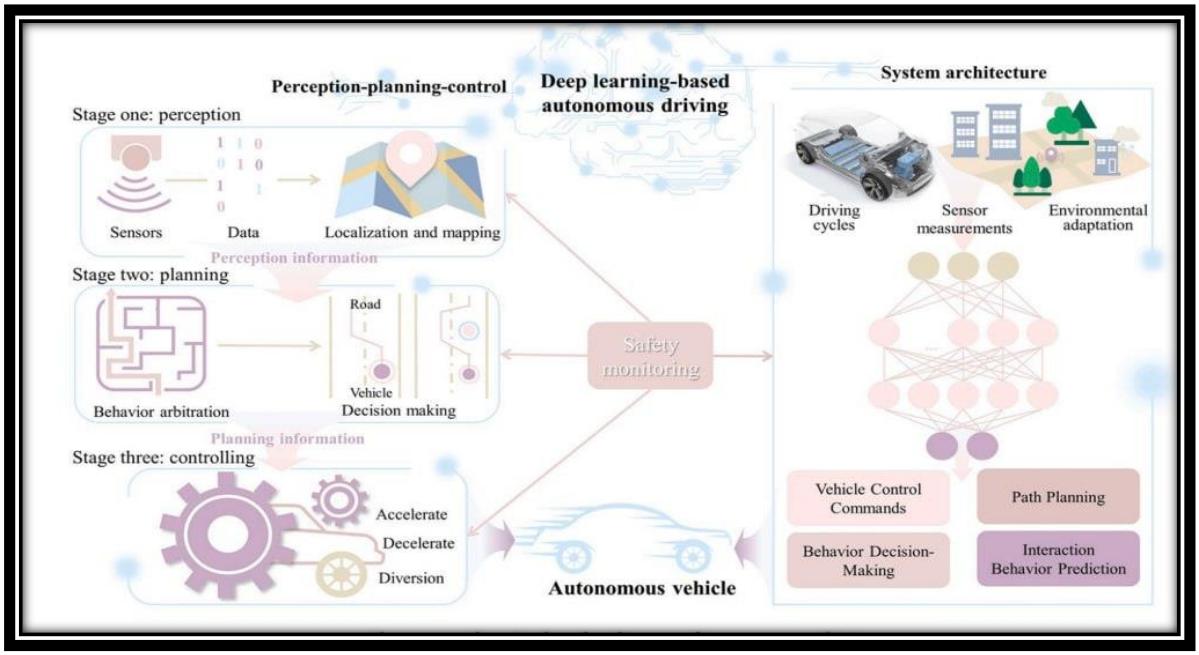


Figure (I.10): the stages of AVs powered by DL [13].

In the **Perception** stage, sensor data such as images, point clouds, or radar signals are processed to detect and classify objects, recognize lane markings, and understand the structure of the environment. Deep learning models, especially convolutional neural networks (CNNs), are widely used in this phase to perform tasks like object detection, semantic segmentation, and depth estimation.[13, 14].

The **Planning** stage involves making decisions based on the output of the perception module. Here, the vehicle determines its trajectory, taking into account dynamic elements like other vehicles, pedestrians, and road rules. Deep reinforcement learning and neural motion planning approaches are increasingly being explored to improve the flexibility and safety of these decisions [13]. Finally, the **Control** stage translates the planned path into precise steering, acceleration, and braking commands. While traditional control methods remain dominant in this stage, deep learning is also being investigated to handle complex maneuvers or adapt to uncertain environments [11].This deep learning pipeline offers significant advantages in improving perception accuracy, decision-making under uncertainty, and robustness in diverse real-world scenarios.

I.8 Role of Perception in AVs

The perception system of an autonomous vehicle is essential for accurately interpreting its environment, enabling tasks like object detection, road layout understanding, pedestrian recognition, and localization. Beyond mere detection, this system must semantically distinguish between elements (e.g., a pedestrian vs. a lamppost or a road vs. a sidewalk), leveraging computer vision techniques such as classification, tracking, and most critically image segmentation. By assigning pixel-level labels to each object or surface, segmentation provides a granular understanding of the scene, clarifying not just what the vehicle sees but also where elements are positioned. This precision is indispensable for lane detection, free space estimation, and dynamic object tracking, forming the foundation for safe autonomous driving [11], [14].

I.9 Autonomous Driving Tasks

Autonomous driving involves the coordination of multiple tasks that allow the vehicle to perceive its environment, make decisions, and execute actions safely. These tasks include **perception**, **self-localization**, **motion planning**, **control**, and **automated parking**. Supporting tasks such as **pedestrian detection**, **traffic sign recognition**, and **road-marking detection** are essential for understanding road context.

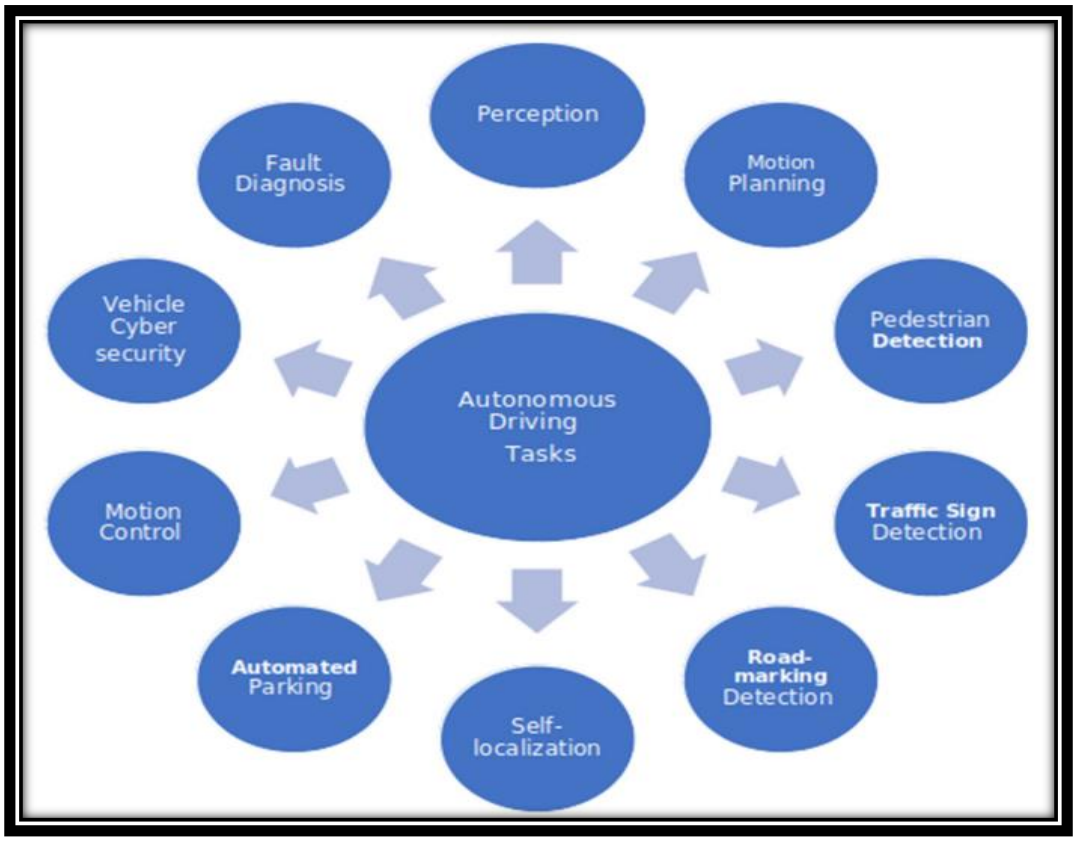


Figure (I.11): The Main tasks involved in Autonomous Driving Systems [15].

In addition, systems like **fault diagnosis** and cybersecurity ensure vehicle reliability and protection against threats [15]. The figure (I.11) summarizes the main tasks required for autonomous vehicle operation:

I-10 Challenges and limitations in Autonomous Driving Systems

Autonomous driving systems face a wide array of challenges that go beyond the technical scope of perception and control. As shown in the following figure, these include the need for higher safety standards, robust performance under adverse weather, and the ability to navigate complex and dynamic environments. Other key challenges involve interpreting human behavior, ensuring cybersecurity, and adapting to legal and ethical frameworks. Achieving public trust and developing reliable high-definition mapping solutions are also crucial to the widespread adoption of autonomous vehicles [13].

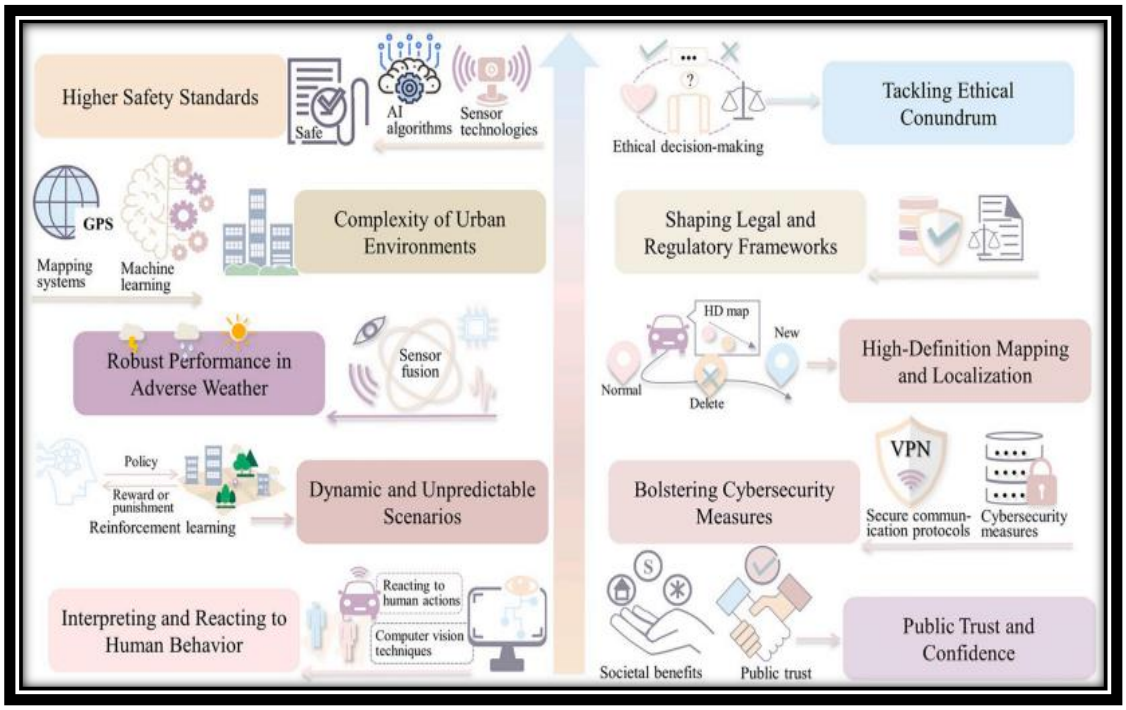


Figure (I.12): Current challenges and limitations in Autonomous Driving [13].

I.11 Image Segmentation in the Context of Autonomous Driving

While image segmentation can be performed using classical methods based on color, edge detection, or clustering, these approaches often struggle in real-world, dynamic environments such as urban roads. In our thesis, we focus on deep learning-based segmentation, which leverages convolutional neural networks (CNNs) to learn complex spatial and semantic features directly from labeled data. Specifically, we apply semantic segmentation to road scenes, where each pixel is assigned a class label to aid perception in autonomous vehicles.

I.11.1 Definition

Image segmentation is a fundamental technique in computer vision that involves dividing an image into meaningful regions or segments. In the context of autonomous driving, segmentation allows the vehicle to understand its surroundings by identifying and localizing different objects and surfaces such as roads, pedestrians, vehicles, traffic signs, sidewalks, and obstacles at the pixel level [16].

Figure (I.13): An example of an image segmented.

I.11.2 Key Image Properties Relevant for Road Scene Segmentation

In the context of autonomous driving, not all image characteristics are equally important. Since segmentation is performed on road scenes captured in real-time, specific image properties directly affect both the accuracy and speed of the system. Understanding these properties helps optimize segmentation performance in real-world scenarios.

I.11.2.1 Pixel

A pixel is the smallest unit of an image. In image segmentation, each pixel is individually classified into a specific category such as road, vehicle, pedestrian, or background. The accuracy of pixel-level labelling is essential in autonomous driving, where incorrect classification can lead to unsafe decisions [17, 18].

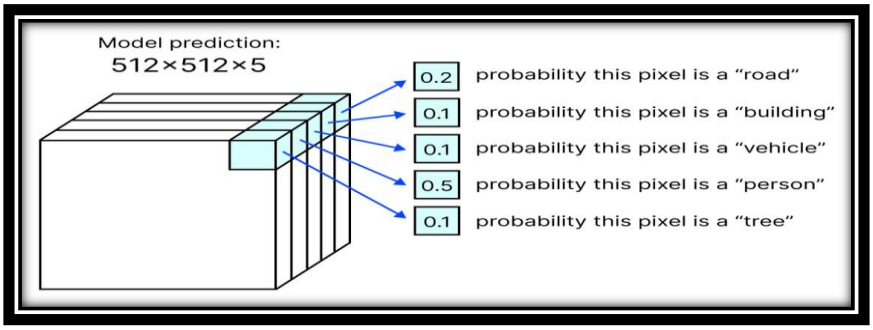


Figure (I.14): Pixel-level prediction example [19].

I.11.2.2 Region of Interest (ROI)

The region of interest refers to the specific area of an image that contains the most relevant information for a task. In autonomous driving, this is usually the area directly ahead of the vehicle where lanes, vehicles, and pedestrians are located. Focusing segmentation on this region reduces unnecessary computation and improves real-time performance [20].

Figure (I.15): ROI prediction example [20]

I.11.2.3 Image Resolution

Resolution defines the number of pixels in an image. Higher resolution can capture finer details, improving segmentation quality. However, it also increases computational cost. In real-time systems, an optimal resolution must balance performance with processing speed [18].

Figure (I.16): Image Resolution example

I.11.2.4 Contrast and Lighting Conditions

Changes in contrast, shadows, or lighting (e.g., at night or in fog) can make segmentation harder. Low visibility may reduce the accuracy of object boundaries. Robust segmentation models must handle these variations to ensure reliability in all driving conditions , figure (I.17), [21].

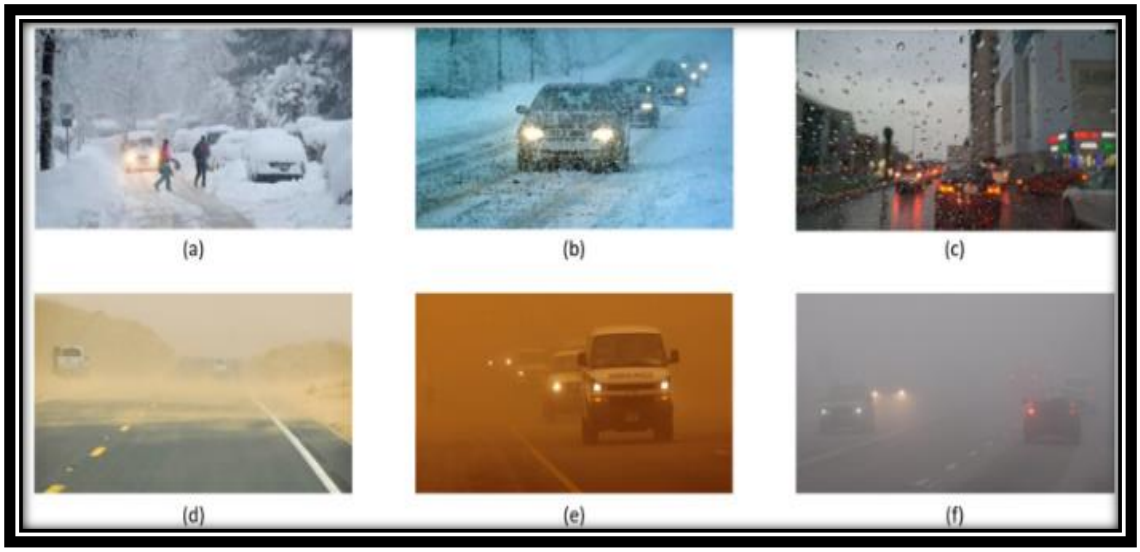


Figure (I.17): Examples of road scene images under different conditions [21].

I.11.3 Types of Image Segmentation

There are several types of segmentation used in autonomous vehicle systems , figure (I.18) , [18]:

- **Semantic Segmentation**: Assigns a class label (e.g., road, vehicle, pedestrian) to every pixel in the image. It does not distinguish between individual instances of the same class.
- **Instance Segmentation**: Extends semantic segmentation by differentiating between separate objects of the same class (e.g., two pedestrians will be labeled separately).
- **Panoptic Segmentation**: Combines both semantic and instance segmentation, providing a complete understanding of the scene.

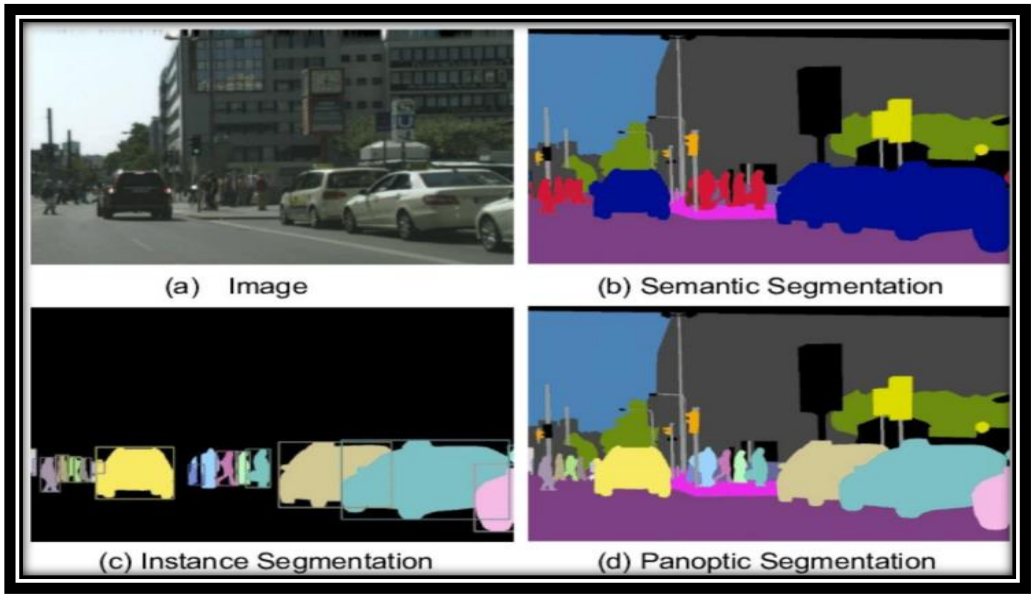


Figure (I.18): The Different Types of Image Segmentation [22].

These segmentation tasks are often powered by **deep learning models**, particularly Convolutional Neural Networks (CNNs) and architectures like U-Net**,** DeepLab**,** Mask R-CNN, which have shown high performance in complex urban environments.

I.11.4 Importance of Image Segmentation in AVs

In autonomous driving, segmentation is crucial for several reasons [23]:

- **Lane and Road Detection**: Segmenting the road surface helps determine the drivable area.
- **Obstacle Recognition**: Allows the system to detect static and dynamic obstacles.
- **Traffic Element Detection**: Identifies lane markings, traffic signs, and signals to guide decision-making.
- **Scene Understanding**: Helps the system adapt its behavior in urban, highway, or rural settings by understanding the context.

I.12 Conclusion

This chapter has presented the fundamental concepts and background necessary to understand the challenges and objectives of this thesis. We began with defining autonomous vehicles, their history, and the key tasks involved in their operation, such as perception, localization, planning, and control. We then presented the various challenges associated with deploying these systems in real-world environments.

A central focus of this chapter was image segmentation, a critical component of the perception system in autonomous driving. We defined the concept, discussed its relevance to road scenes, and introduced key image properties that influence segmentation performance, such as pixel-level labeling, region of interest, resolution, and lighting conditions. Various types of segmentation were also reviewed, along with their specific roles in understanding complex driving environments.

This theoretical foundation sets the stage for the next chapter, where we will explore deep learning approaches that enable high-performance, real-time image segmentation. These methods are at the heart of modern perception systems in autonomous vehicles and represent the core of this research.

References

[1] H. T. Ngoc et al., "Lane Road Segmentation Based on Improved U-Net Architecture for Autonomous Driving," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 7, 2023

[2] SELAIMIA Yassine, "Détection des objets pour véhicules autonomes," Université 8 Mai 1945 - Guelma, Mémoire de fin d'études, 2022

[3] Stanford University, "Stanford's robotics legacy," Stanford News, May 23, 2019. [Online]. Available: https://news.stanford.edu/2019/01/16/stanfords-robotics-legacy/

[4] Tsukuba Mechanical Engineering Lab, "Japan 1977: Computerized Driverless Car," ResearchGate. https://www.researchgate.net/figure/Tsukuba-Mechanical-Engineering-Lab-Japan-1977-computerized-driverless-car-achieved-spe_fig2_365874855

[5] "The first high-speed autonomous vehicle VaMoRs (1986–2004)," Research Gate. https://www.researchgate.net/figure/The-first-high-speed-autonomous-vehicle-VaMoRs-from-1986-to-2004-with-three-generations_fig3_341211118

[6] S. Thrun et al., "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robot.*, vol. 23, no. 9, 2006

[7] Universitat Oberta de Catalunya (UOC), "Cityscapes Segmentation - CodaLab Challenge," CodaLab Competitions. https://codalab.sunai.uoc.edu/competitions/111

[8] Waymo, "Waymo One in Phoenix," https://waymo.com/waymo-one-phoenix

[9] M. Rawat, "Artificial Intelligence in Self-Driving Cars: Applications, Implications and Challenges," Research Gate, 2022.

[10] "Levels of automation in autonomous vehicles," Research Gate https://www.researchgate.net/figure/Levels-of-automation-in-autonomous-vehicles-based-on-16_fig1_389736284

[11] S. Liu, L. Li, J. Tang, S. Wu, and J.-L. Gaudiot, Creating Autonomous Vehicle Systems, Morgan & Claypool Publishers

[12] Ü. Özgüner, T. Acarman, and K. Redmill, *Autonomous Ground Vehicles*, Artech House, 2011.

[13] A. Badawy *et al.*, "Autonomous Driving System: A Comprehensive Survey," *IEEE Trans. Intell. Transp. Syst*, 2022.

[14] H.-H. Jebamikyous and R. Kashef, "Autonomous Vehicles Perception (AVP) Using Deep Learning: Modeling, Assessment, and Challenges," *IEEE Access*, vol. 10, pp. 11278–11291, 2022.

[15] R. Qayyum, J. Qadir, A. Al-Fuqaha, and M. Imran, "Secure and Robust Machine Learning for Healthcare: A Survey," *Mach. Learn. Appl.*, vol. 6, p. 100164, 2021

[16] D. B. L. Deepak, N. Manasa, B. Shabarish, M. V. S. Sai Kumar, and G. Maheshwari, "Real-Time Image Segmentation for Self-Driving Cars Using Deep Learning," *Int. J. Res. Anal. Rev. (IJRAR)*, vol. 10, no. 2, May 2023.

[17] Labellerr, "Optical Flow in Autonomous Vehicles," *Labellerr Blog*, 2023

[18] R. Baroudi and Z. Banchiha, "Approche d'apprentissage profond pour la segmentation sémantique d'images," Université d'Ain Temouchent - Belhadj Bouchaib, Mémoire de fin d'études, 2024.

[19] R. Pulapakura, "Image Segmentation: A Beginner's Guide," *Medium*, 2022. [Online]. Available: https://medium.com/@raj.pulapakura/image-segmentation-a-beginners-guide-0ede91052db7

[20] "Initializing the Region of Interest (ROI)," *ResearchGate*. [Online]. Available: https://www.researchgate.net/figure/Figure4-Initializing-the-Region-of-interest-ROI_fig3_325401096

[21] K. Abbas *et al.*, "Autonomous Vehicles Perception (AVP) Using Deep Learning: Modeling, Assessment, and Challenges," *Electronics*, vol. 13, no. 15, p. 3063, 2024.

[22] Labellerr, "Semantic vs Instance vs Panoptic: Which Image Segmentation Technique to Choose" *Labellerr Blog*, 2023

[23] T. Tirumalapudi, N. Kumar, and S. Sirisha, "Onward and Autonomously: Expanding the Horizon of Image Segmentation for Self-Driving Cars Through Machine Learning," *Scalable Comput.: Pract. Exp.*, vol. 25, no. 4, pp. 3163–3171, 2024.

Chapter II
Deep Learning

II.1 Introduction

Deep learning has become a cornerstone of modern artificial intelligence, enabling machines to learn complex patterns from raw data. Unlike traditional methods, it eliminates the need for manual feature engineering by leveraging multi-layer neural networks. In this chapter, we explore key components of deep learning, including artificial neural networks, perceptrons, activation functions, loss functions, gradient descent, and backpropagation. A special focus is given to convolutional neural networks (CNNs), which are highly effective for image-based tasks. These models have shown remarkable success in computer vision, especially in object detection and segmentation. Such capabilities are essential for perception systems in autonomous driving, where real-time scene understanding is critical. This foundation sets the stage for the advanced segmentation models applied later in this thesis.

II.2 Artificial Intelligence

Artificial Intelligence (AI) involves the development of computer systems Focus on human-level intelligence. According to the National Institute of Standards and Technology (NIST), AI is defined as "A machine-based system that can, for a given set of human-defined objectives, make predictions, recommendations, or decisions influencing real or virtual environments"[1].

AI integrate specially for computer science, mathematique, engineering to create autonomous system. These systems use algorithms to learn from data and perform tasks like real-time object detection in self-driving cars and image segmentation...

II.3 Deep learning

Deep learning is a form of machine learning uses multilayered of artificial neural networks that enables computers to learn from experience and understand the world in terms of a hierarchy of concepts (learn hierarchical representations of data). Inspired from the knowledge of the human brain. Success of Deep learning based in three key advancements in computational (GPU …), power large datasets (KITTI, cityscapes…), and algorithmic innovations (CNN…) [2].

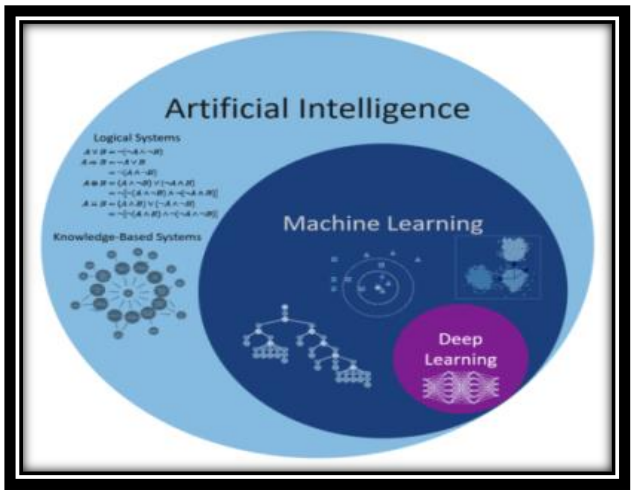


Figure (II.1) : The Levels of Artificial Intelligence.

II.4 DL process

The deep learning begins with a problem definition, followed by collecting and preprocessing DATA, then designed the artificial neural network architecture. Training the model, finally evaluation and validation.

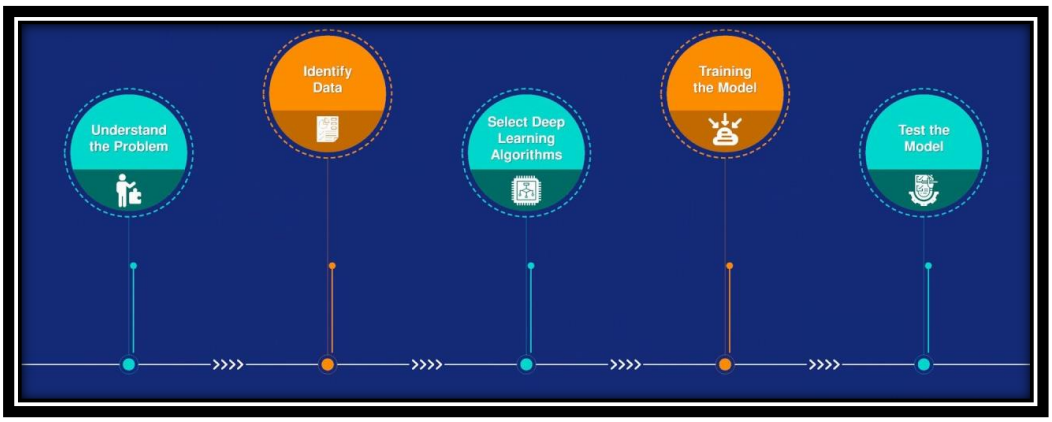


Figure (II.2) : Deep learning process steps [3].

II.5 Deep learning applications

- Image Recognition.
- Artistic Style Transfer.
- Drug Discovery.
- Autonomous Vehicles.
- Medical image analysis.
- Financial Forecasting: Neural networks are utilized in predicting stock market trends, credit risk assessment.

- Customer Service Chabot's: Neural networks power Chabot's that provide customer support and respond to inquiries [4].

II.6 Machine learning VS deep learning

Deep Learning (DL), a subset of Machine Learning (ML), uses complex deep neural networks to learn from large, unstructured datasets, requiring specialized GPUs for training. It automates feature extraction, achieves high accuracy, but demands significant time and resources. Machine Learning, a branch of Artificial Intelligence (AI), employs simpler algorithms for structured data, runs efficiently on CPUs, and trains faster with moderate data. While ML often relies on manual feature engineering and yields lower precision, DL excels in tasks like image recognition at the cost of computational intensity [5].

II.7 Artificial Neural Networks

Artificial neural networks (ANNs) are software implementations of the neuronal structure of our brains. The brain contains neurons, which are kind of like organic switches. These can change their output state depending on the strength of their electrical or chemical input. The neural network in a person's brain is a hugely interconnected network of neurons, where the output of any given neuron may be the input to thousands of other neurons. Learning occurs by repeatedly activating certain neural connections over others, and this reinforces those connections. This makes them more likely to produce a desired outcome given a specified input. This learning involves feedback – when the desired outcome occurs, the neural connections causing that outcome becomes strengthened [6].

Brain	Artificial Neural Network
Neuron. Connection of neurons.	Perceptron (node). Connection weight.

Table (II.1): Difference between brain & Artificial neural network [7].

II.8 Architecture of Artificial Neural Networks

Neural network alters or learns, in a sense based on the input and output, the information that goes through the network modifies the structure of the ANN

The most popular sort of artificial neural network is made up of three sets of layers: a layer of "input" units connected to a layer of "hidden" units connected to a layer of "output" units [8].

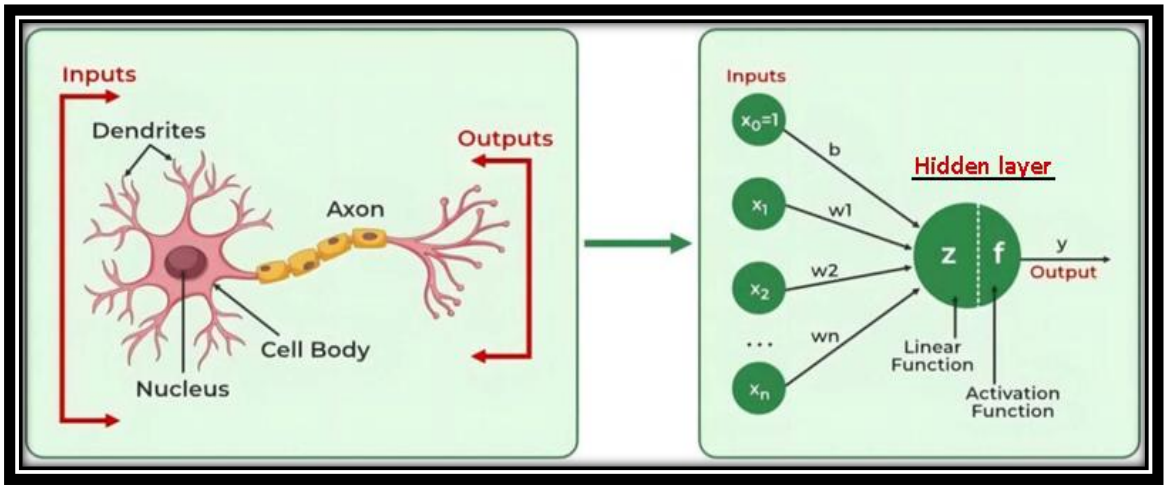


Figure (II.3): Biological neural vs artificial neural.

- **Input Layer:** Receives raw data (e.g., pixels, text) and passes it to the network. Each neuron represents a single feature, like one pixel in an image.
- **Hidden Layer(s):** Process data through weighted connections and activation functions to extract patterns. Multiple layers can detect edges, shapes, or complex relationships.
- **Output Layer:** Produces the final result, like a classification label or regression value. Uses activation functions tailored to the task.

II.9 Perceptron

The history of deep learning can be traced back to the 1940s and 1950s, when researchers first began exploring the idea of building artificial intelligence based on the structure and function of the brain. The earliest form of neural network was the perceptron (Rosenblatt, 1958), developed by Frank Rosenblatt in the late 1950s. The perceptron was a simple model that consisted of a single layer of artificial neurons [9].

There are two types of perceptron models:

Single-layer neural network		Input layer – output layer
Multi-layer neural network	Shallow neural network	Input layer – hidden layer – Output layer
	Deep Neural Network	

Table (II.2): Difference between single and multilayer neural network.

- **Signal layer perceptron:** is the simplest form of a neural network, consisting of just one layer of input nodes connected directly to output nodes [9].

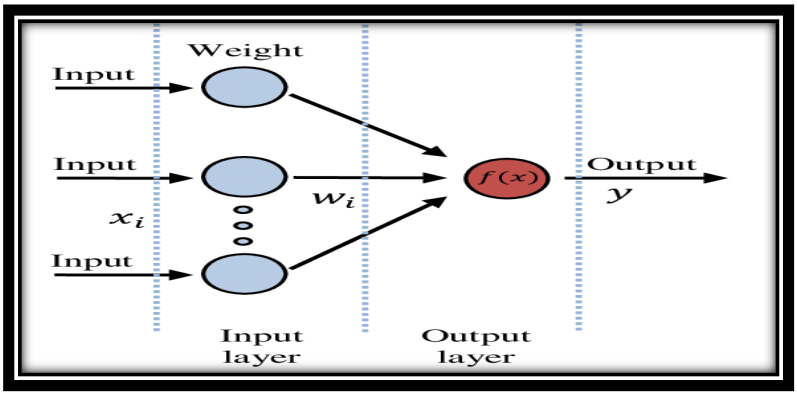


Figure (II.4): Architecture of single-layer perceptron.

❖ **Multi-layers perceptron:** In the 1960s and 1970s, neural network research faced challenges due to limited computational power and simple models, though multilayer networks emerged. Breakthroughs came in the late 1980s and 1990s with improved hardware and algorithms like backpropagation, enabling deep neural networks. These deep networks, with multiple layers, could learn complex patterns, revolutionizing fields like image and speech recognition. This marked the beginning of modern AI advancements. Today, deep learning drives innovations across industries. Often used as a basis for more advanced models such as convolutional neural networks (CNNs) [9].

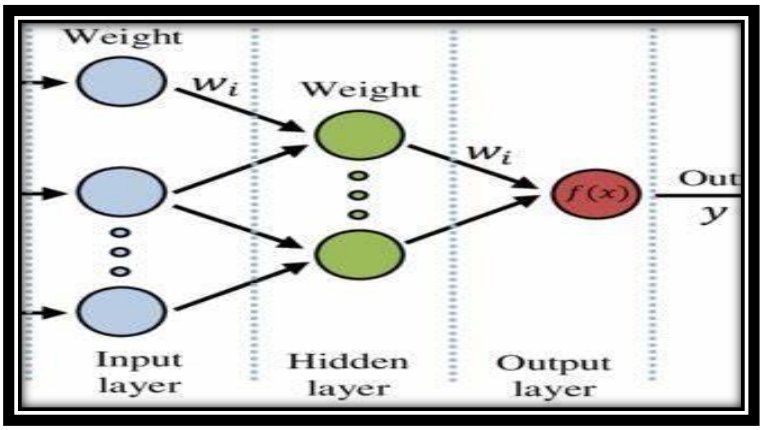


Figure (II.5): Architecture of multi-layers perceptron.

II.10 Perceptron components

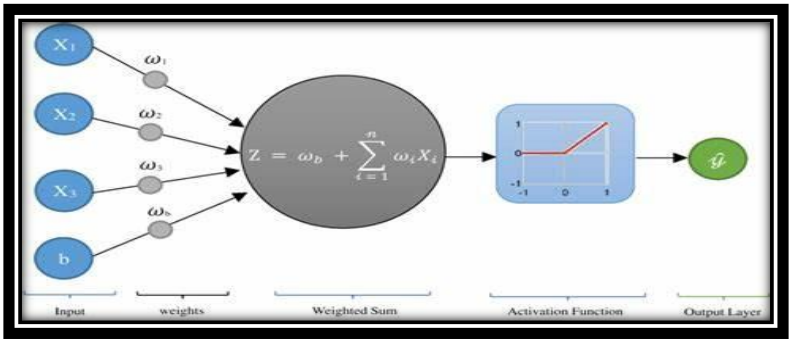


Figure (II.6): Basic components of a perceptron.

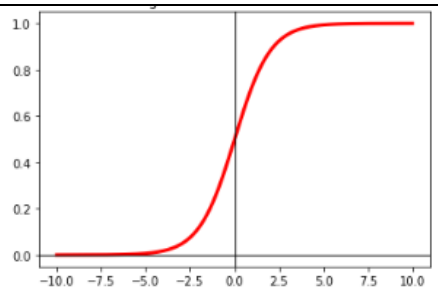
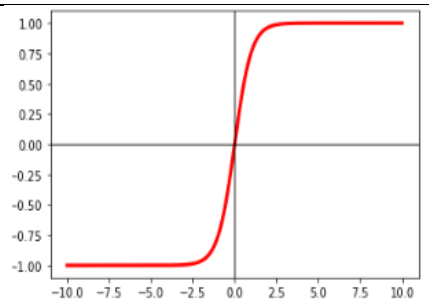
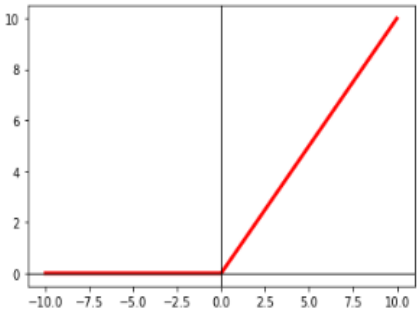
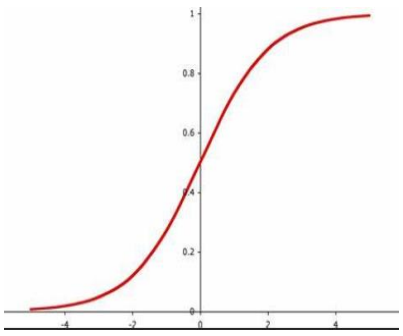
Activation function	Equation	Marge	Graph
Sigmoid: Binary classification problems where the output represents probabilities.	$\sigma(x) = \frac{1}{1+e^{-x}}$	[0,1]	
Tanh: Similar to the sigmoid function but with an output range that is symmetric around zero.	$\tanh(x) = \frac{(e^{x}-e^{-x})}{(e^{-x}+e^{-x})}$	[-1,1]	
ReLU (Rectified Linear Unit): Widely used due to its simplicity and ability to alleviate the vanishing gradient problem. It sets negative values to zero and keeps positive values unchanged.	$\mathrm{ReLU}(x) = \max(0, x)$	[0,x]	
Softmax: Commonly used in the output layer for multi-class classification problems to produce a probability distribution across different classes.	$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$	[0,1]	

Table (II.3): Different activation functions.

- **Input layer:** This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input perceptron contains a real numerical value.

- **Weight:** In a perceptron, weights (constrained to [-1,1]) parameterize the influence of inputs on the output: negative weights invert input contributions, positive weights reinforce them, and zero weights nullify connections. Geometrically, the weight vector w defines the normal to the decision hyperplane in $\mathbb{R}^n$, partitioning the input space via the inequality $(w * x) + b \geq 0$ where classification hinges on the alignment of inputs with w [10].
- **Bais:** A constant used to shift the decision boundary to correctly fit the dataset.
- **Sum:** Each input is multiplied by a corresponding weight assigned to the connection.
- **Activation function:** Activation functions are used at the end of a hidden unit after the sum weights to introduce non-linear complexities to the model
- **Output:** Based on the current parameters, the network generates an output.

II.11 Learning in Neural Networks

II.11.1 Feedforward Neural Networks

Deep feedforward networks, also called feedforward neural networks, or multilayer perceptron's (MLPs), are the quintessential deep learning models [11]. This means there are no loops in the network– information is always fed forward, never fed back. Through the hidden layers. The activations of each neuron are calculated using weights, biases and activations functions. If we did have loops, we'd end up with situations where the input to the function depended on the output. That'd be hard to make sense of, and so we don't allow such loops [9].

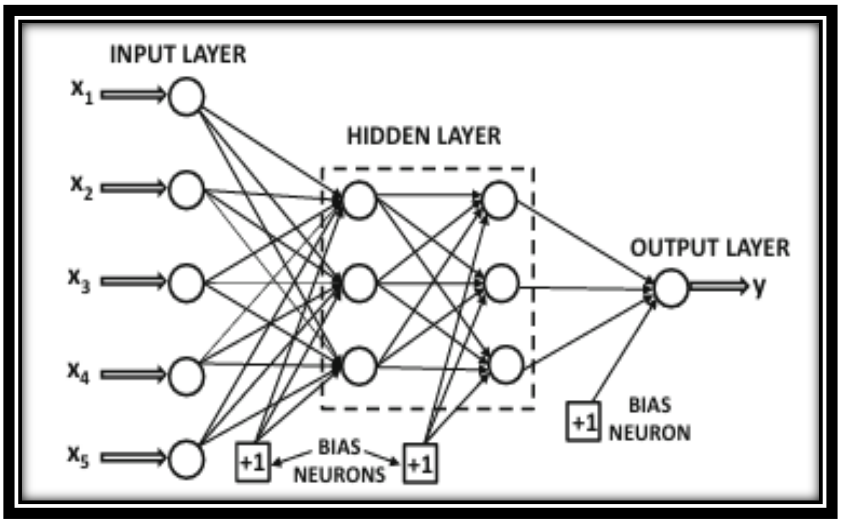


Figure (II.7): Feedforward Neural Networks.

II.11.1.1 Supervised Learning of a Neural Network

Supervised learning in the neural network proceeds in the following steps:

- Start by setting the network's weights to small, random values to create a baseline for learning.
- For each training example (input + correct output pair), run the input through the network. Compare its prediction to the correct output to calculate the error.
- Update the weights slightly using methods like backpropagation and gradient descent to reduce the error.
- Cycle through all training data repeatedly until predictions align closely with the correct outputs.
- This process mirrors general supervised learning but focuses on tuning neural network weights instead of simpler parameters. The goal is always the same: minimize the gap between predictions and ground-truth labels [12].

II.11.1.2 Loss Function

The loss function is essential for guiding the learning process in deep learning. It provides a numerical measure of the discrepancy between the model's predicted outputs and the actual target labels in the dataset. This measure, called the **loss**, is expressed as a single scalar value that reflects the model's performance for a given set of parameters [13].

During training, the model aims to minimize this loss by adjusting its internal parameters primarily the weights and biases b. This adjustment is typically achieved through optimization algorithms such as gradient descent, which iteratively update the parameters in the direction that reduces the loss. Mathematically, the objective is to find the set of weights ω and biases b that minimize the total loss function $J(\omega, b)$ over the entire training dataset [14]:

$$J(\omega, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \qquad \text{(II. 1)}$$

Where m is the number of training examples, $\hat{y}^{(i)}$ is the predicted output, $y^{(i)}$ and is the true label.

This process of minimizing the loss is analogous to minimizing residuals in classical statistics, where the goal is to reduce the deviation between observed values and model predictions. Similarly, a lower loss indicates that the model's predictions are closely aligned with the true values, implying better performance [14]. Loss function depends largely on the type of problem being solved.

For regression problems, I often rely on loss functions like Mean Squared Error (MSE) or Mean Absolute Error (MAE). On the other hand, for classification tasks**,** Cross-Entropy Loss is a common choice because of how effectively it penalizes confident but incorrect predictions.

a) **Categorical Cross-Entropy Loss Function**

In semantic segmentation tasks, the categorical cross-entropy loss is widely used to measure the discrepancy between the predicted class probabilities and the true labels at the pixel level. It is particularly effective for multi-class classification problems with mutually exclusive categories.

$$\mathcal{L}_{CCE} = -\sum_{c=1}^{C} y_c * \log(\hat{y}_c) \quad \text{(II.2)}$$

Where C is the number of classes, y_C is the ground-truth label (one-hot encoded), and $\hat{y}_C$ is the predicted softmax probability for class c.

In practice, deep learning frameworks such as PyTorch and TensorFlow refer to this loss simply as cross-entropy, and handle label encoding and softmax internally. This function penalizes incorrect predictions more strongly when the model is overly confident, guiding the training process toward more accurate and calibrated outputs [2].

b) **Focal Loss:** Designed to address class imbalance by down-weighting easy examples [15].

$$\mathcal{L}_{Focal} = -\alpha_t(1-\hat{y}_t)^{\gamma}\log(\hat{y}_t) \quad \text{(II.3)}$$

II.11.1.3 Gradient descent

Gradient descent is an optimization technique categorized as a first-order iterative algorithm, primarily employed to minimize objective functions. It is particularly effective for functions that are continuous and convex. In the context of neural networks, this algorithm systematically updates the model's weights in an attempt to minimize the associated cost function. During each iteration, the weight vector ω is adjusted in proportion to the negative gradient of the cost function J(ω), effectively moving the solution closer to a local minimum. This update rule is mathematically represented as:

$$\omega' = \omega + \Delta\omega \quad \text{(II.4)}$$

$$\omega' = \omega - \eta * \Delta J(\omega) \quad \text{(II.5)}$$

In this expression, ω' denotes the updated weight vector, η is the learning rate that governs the step size, and $\Delta J(\omega)$ is the gradient of the cost function with respect to ω.

The learning rate plays a critical role in ensuring convergence, with ω_j inappropriate values potentially leading to divergence or excessively slow learning. When considering an individual neuron with weight, the update rule becomes:

$$\omega'_j = \omega_j - \eta \frac{\partial J(\omega)}{\partial \omega_j} \tag{II.6}$$

This formulation uses the partial derivative to update each weight independently. In simple cases such as linear transformations, these gradients can be derived analytically. However, in more complex models, particularly deep neural networks, gradient computations are typically performed using the backpropagation algorithm [16].

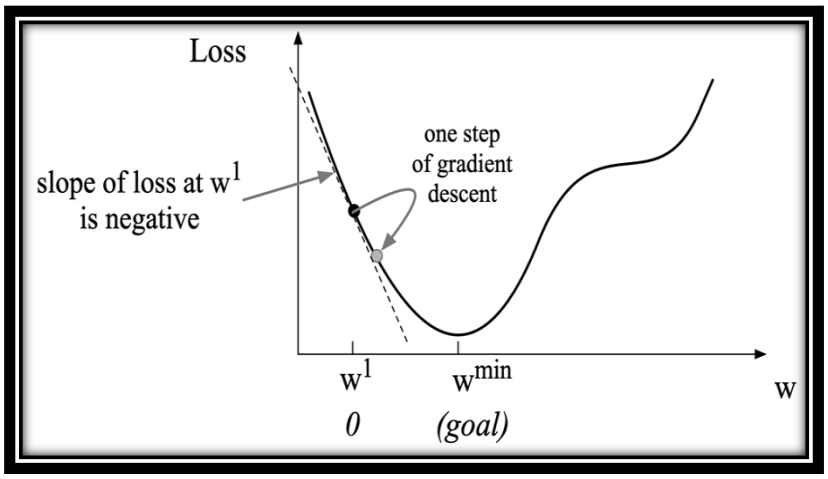


Figure (II.8) : 1d-gradient-descent.

II.11.1.4 Back propagation

Backpropagation is a foundational algorithm in the training of artificial neural networks. It facilitates the efficient computation of the gradient of the loss function with respect to the network's trainable parameters, such as weights and biases. The primary goal in supervised learning is to minimize the discrepancy between the predicted outputs and the actual ground truth labels. To achieve this, the network's parameters are updated iteratively in a manner that reduces the loss function. In linear models, such as linear regression, this is typically accomplished using gradient descent. In deep neural networks, the backpropagation algorithm extends this principle by applying the chain rule of calculus to compute gradients through multiple layers.

Backpropagation defines the overall procedure that includes both the computation of gradients and their application within gradient-based optimization methods, such as stochastic gradient descent (SGD).

It computes the partial derivatives of the loss function with respect to each parameter in the network by recursively applying the chain rule from the output layer back to the input layer. This iterative and recursive mechanism allows the model to refine its internal parameters in a structured and computationally efficient manner.

The backpropagation process is composed of two main phases: the forward pass and the backward pass [17].

a) Forward Pass

In the forward pass, the input data is propagated through the network layer by layer to compute the predicted output. At each layer l, the pre-activation value$[Z^l]$ and activation $[a^l]$ are calculated as follows [18]:

$$Z^{[l]} = \omega^{[l]} a^{[l-1]} + b^{[l]},\ a^{[l]} = \sigma\ (Z^{[l]}) \tag{II.7}$$

Where:

- $\omega^{[l]}$ and $b^{[l]}$ are the weights and biases at layer l
- σ is the activation function (e.g., ReLU, softmax, sigmoid).
- $a^{[l-1]}$ is the activation from the previous layer (or the input if l=1).

b) Backward Pass

The backward pass involves calculating how the error from the loss function propagates backward through the network. Using the chain rule of calculus, the algorithm computes the gradient of the loss with respect to each weight and bias [18].

Starting from the output layer:

$$\delta^{[l]} = \frac{\partial \mathcal{L}}{\partial a^l} \circ \sigma'(Z^{[l]}) \tag{II.8}$$

The error term $\delta^{[L]}$ for each hidden layer is then computed recursively:

$$\delta^{[l]} = \left(\omega^{[l+1]}\right)^T \delta^{[l+1]} \circ \sigma'(Z^{[l]}) \tag{II.9}$$

Gradients with respect to weights and biases are then:

$$\frac{\partial \mathcal{L}}{\partial \omega^{[l]}} = \delta^{[l]}\left(a^{[l-1]}\right)^T,\ \frac{\partial \mathcal{L}}{\partial b^{[l]}} = \delta^{[l]} \tag{II.10}$$

Finally, these gradients are used to update the parameters using gradient descent:

$$\omega^{[l]} := \omega^{[l]} - \eta \frac{\partial \mathcal{L}}{\partial \omega^{[l]}} \ , b^{[l]} := \omega^{[l]} - \eta \frac{\partial \mathcal{L}}{\partial b^{[l]}} \tag{II.11}$$

Where η is the learning rate.

By alternating between the forward and backward passes, backpropagation enables the network to iteratively refine its parameters, minimizing the loss function and improving performance on the training data.

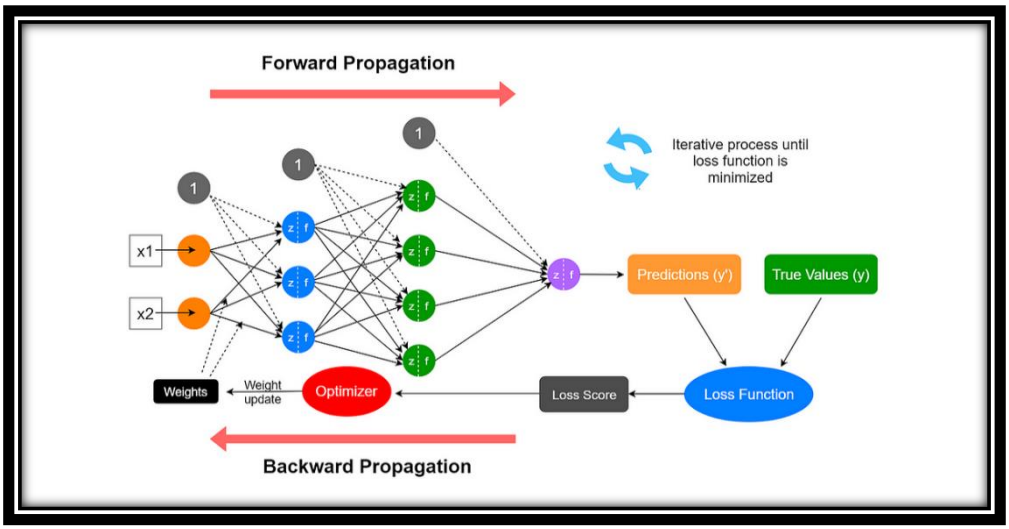


Figure (II.9): Forward and Backward Propagation.

II.12 Types of Neural Networks

- Feedforward Neural Network (FNN).
- Convolutional Neural Network (CNN).
- Recurrent Neural Network (RNN).
- Long Short-Term Memory (LSTM).
- Generative Adversarial Network (GAN).
- Spiking Neural Network (SNN).

II.13 Convolutional Neural Network (CNN)

A convolutional neural network (CNN) is a network architecture for deep learning that learns directly from data, eliminating manual feature extraction. CNNs are especially useful for finding patterns in pictures to recognize objects, faces, and scenes.

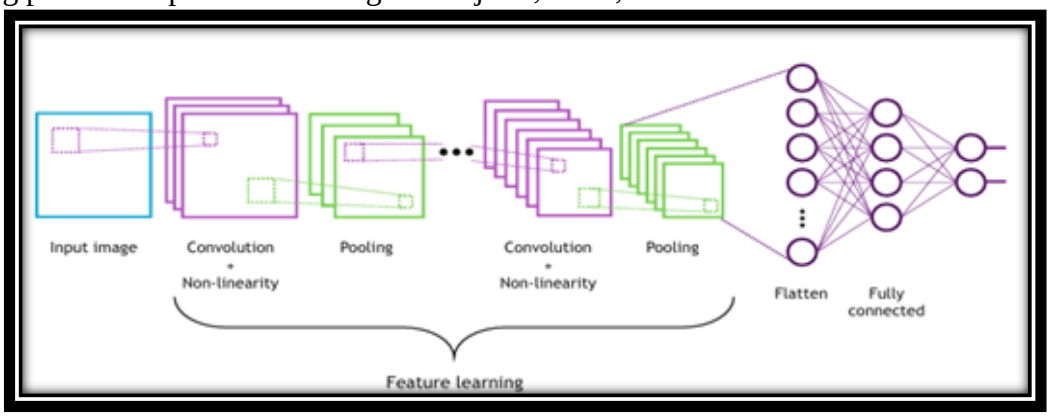


Figure (II.10): Convolutional Neural Network.

II.13.1 CNN component (architecture)

A Convolutional Neural Network is composed by several kinds of layers, that are described in this section : convolutional layers, pooling layers and fully connected layers.

The CNN is a combination of two basic building blocks:

II.13.1.1 Convolution Block

Typically consists of one or more convolutional layers followed by an activation function (such as ReLU) and a pooling layer. This block plays a fundamental role in feature extraction by capturing local spatial patterns and hierarchical representations from the input data [19].

- **Filter:** Filters or 'kernels' are also an image that depict a particular feature.
- **Convolutional layers:** In convolutional layers, a feature map is generated by applying a filter (kernel) across the input volume. The filter moves pixel by pixel (controlled by stride), and overlapping occurs unless stride > 1. If the filter reaches beyond the input's edge, zero padding is used to fill in missing values. For RGB images, both the input and filters have depth (e.g., 5×5×3 input and 3×3×3 filter), and the convolution operates across all channels. Filters can be customized to detect specific features, such as vertical or horizontal edges, by setting particular patterns (e.g., zeros in certain positions). The result is one or more feature maps, whose size is calculated as [20]:

$$O = [\mathrm{N} - \mathrm{F} + 1] * [\mathrm{N} - \mathrm{F} + 1] \qquad \text{(II.12)}$$

Where n is the input size and f is the filter size. The stride defines how many pixels the filter moves at each step as it slides over the input.

- **Stride:** of 1 means the filter moves one pixel at a time, producing a larger feature map. A larger stride reduces the output size, effectively performing down sampling [21].
- **Padding:** is used when the filter does not fit perfectly over the input dimensions, especially at the borders. Zero padding is the most common technique, where the input is surrounded with zeros to allow the filter to fully cover the edge regions. Padding can help preserve the spatial dimensions of the input after convolution [21].

$$\left[\left[O = \frac{N-F+2P}{S} + 1\right], \left[O = \frac{N-F+2P}{S} + 1\right]\right] \quad \text{(II.13)}$$

Where:

- ✓ O: size of the Output.
- ✓ N: size of the Input.
- ✓ S: when we moved filter with a stride.
- ✓ P: when padding is used, denoted by P.

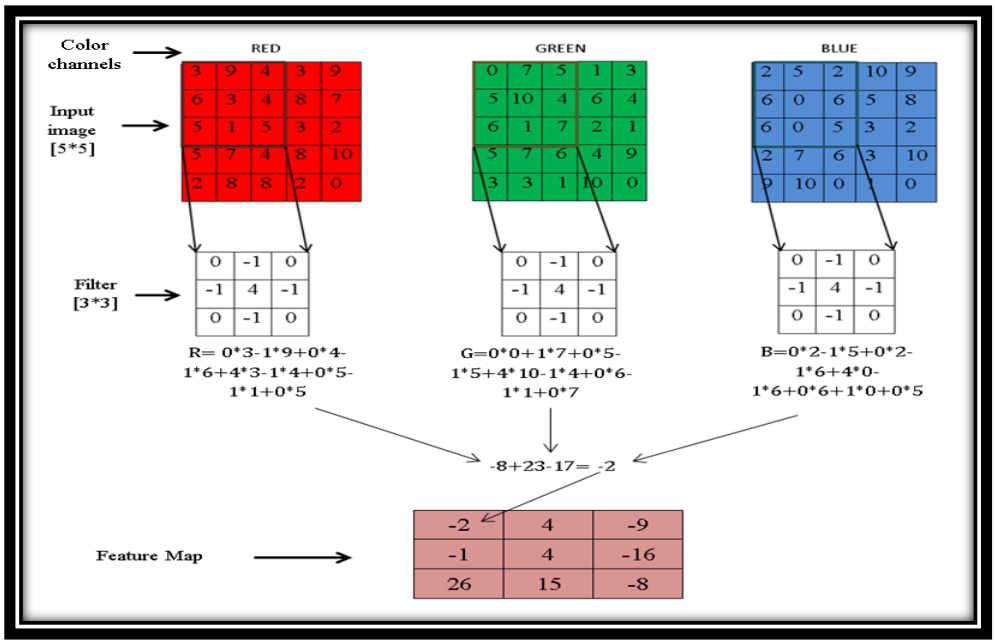


Figure (II.11): Convolution Process in Convolutional Neural Networks with no padding & stride=1.

- **ReLU activation:** The Rectified Linear Unit (ReLU) is an element-wise operation that replaces all negative values in its input with zero. It is mathematically defined as:

$$\text{ReLU}(x) = \max(0, x) \quad \text{(II.14)}$$

Compared to activation functions like sigmoid or tanh, ReLU significantly accelerates the training process, which improves computational efficiency.

In typical Convolutional Neural Network (CNN) architectures, ReLU layers follow convolutional layers to transform the output feature maps. These outputs are then called rectified feature maps.

The application of ReLU after convolution offers several key advantages [22]:

- ✓ While convolution operations are linear (involving only addition and multiplication), applying ReLU introduces non-linearity by discarding negative pixel values. This enables the network to learn more complex patterns.
- ✓ The Faster computation: By eliminating negative activations, ReLU reduces the amount of data processed, leading to faster training and less computational load.
- ✓ ReLU enhances the extracted features by increasing the contrast between them, especially by suppressing non-informative negative values.

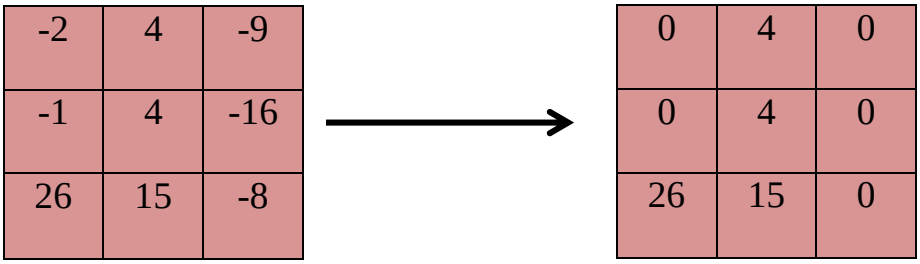


Figure (II.12): application of the ReLU activation in the convolutional layer.

- ❖ **Polling layers:** These layers are periodically inserted after convolutional layer in a typical ConvNet architecture. Their function is to progressively reduce the spatial size of the input representation providing multiple advantages [23]:

- ✓ Obtaining an abstract representation of the input data where only the most important features are being considered; hence, this helps to avoid overfitting the model to the input data.
- ✓ Reducing the computational cost as the number of learnable parameters becomes less than those for corresponding networks not using pooling.
- ✓ Providing basic translation invariance to the input representation. The pooling layer operates independently on each depth slice of the input representation.
- ✓ The most common type is max pooling, where a 2×2 window moves across the input (usually with a stride of 2), and the maximum value within each window is retained. This operation reduces the input size by 75%, keeping only 25% of the original information while preserving the most prominent features.

Alternatively, average pooling can be used. In this case, the average value within each window is calculated instead of the maximum. While max pooling tends to preserve sharp and strong features (like edges), average pooling provides a smoother and more generalized representation. The choice between the two depends on the specific task and desired sensitivity to detail.

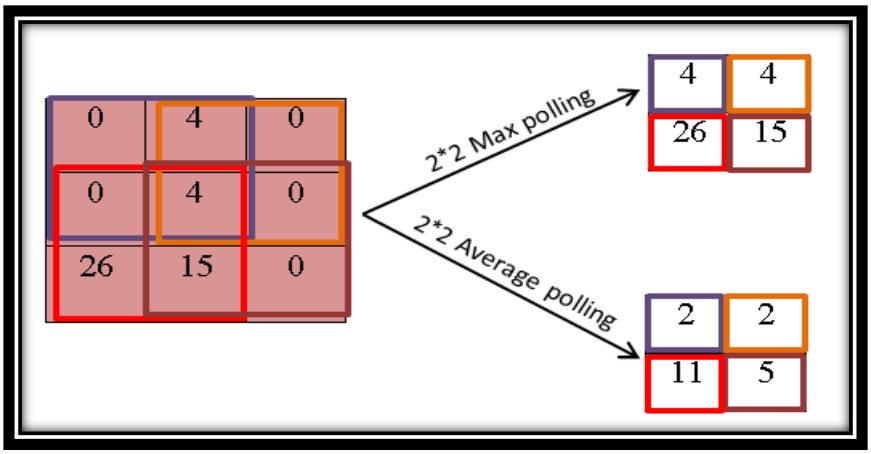


Figure (II.13): CNN Polling layer with (Max Pooling and Average Pooling) stride=1 & no padding.

The convolutional layer and the pooling layer, we form the convolutional block of the CNN architecture. Generally, a simple CNN architecture constitutes of a minimum of three of these convolutional block, that performs feature extraction at various levels.

II.13.1.2 Fully Connected Block

Typically refers to a dense layer in which each neuron is connected to all activations in the previous layer. While traditionally used for classification tasks, in image segmentation architectures, such as U-Net, fully connected layers are generally avoided to preserve spatial information. However, in certain hybrid designs, they may appear in the bottleneck to capture global context before reconstruction [20].

- **Fully Connected layer:** also known as a dense layer, refers to a layer in neural network architectures where each neuron is connected to all the neurons in the preceding layer. Traditionally, fully connected layers are used in the final stages of classification models to aggregate high-level features extracted by convolutional layers and map them to specific class probabilities. However, in image segmentation tasks, where the objective is to generate a dense, pixel-wise prediction map, the use of fully connected layers is generally avoided. This is because fully connected layers flatten the spatial dimensions of feature maps, leading to the loss of crucial spatial information necessary for accurate segmentation.

Instead, segmentation networks such as U-Net and Fully Convolutional Networks (FCNs) rely on convolutional and upsampling operations that maintain the spatial structure of the input throughout the network. Nevertheless, in certain architectures or hybrid designs, fully connected layers might still be utilized within the bottleneck or latent space to capture global contextual relationships between features before the decoding process reconstructs the segmentation output.

Thus, the Fully Connected Block in the context of segmentation is either omitted or carefully integrated to ensure that spatial resolution and localization accuracy are preserved.

Figure (II.14) illustrates a typical CNN processing pipeline. It begins with an input image consisting of multiple channels (e.g., RGB); followed by a sequence of convolutional layers that extract low- to high-level features using convolutional kernels. These features are then passed through activation functions and downsampled via pooling operations to form compressed but informative response maps. The final output of this process serves as the input for subsequent layers such as fully connected blocks or decoder components in segmentation models like U-Net.

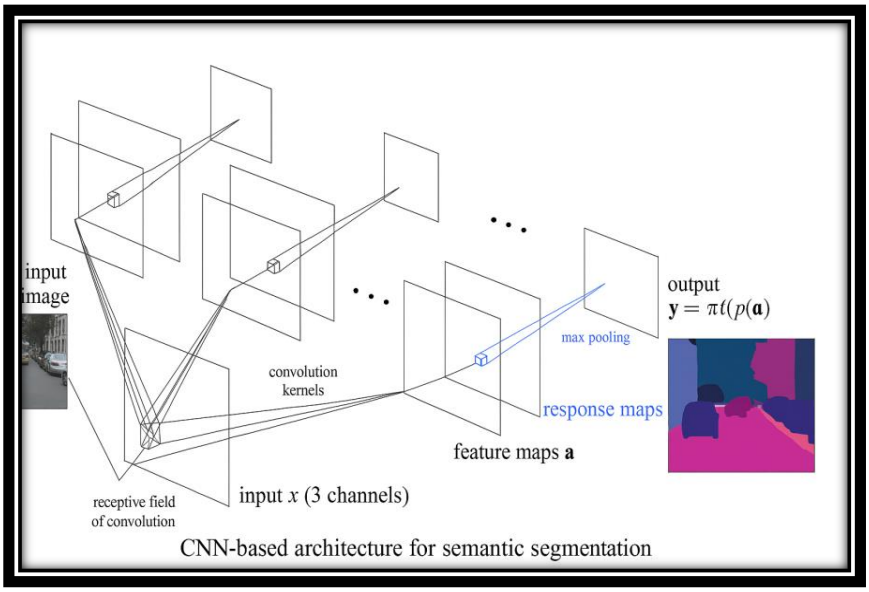


Figure (II.14): Illustration of feature extraction in a CNN.

II.14 Advanced CNN Architectures

Convolutional Neural Networks (CNNs) are specifically designed to handle data with a grid-like structure, such as images, by leveraging layers of convolutions and pooling operations. They have demonstrated remarkable success across various computer vision applications, including image classification, object detection, segmentation, and image synthesis.

Despite their widespread effectiveness, the performance improvements of conventional CNN models have begun to plateau in recent years. To overcome this limitation, researchers have proposed enhanced architectures that introduce novel strategies and structural innovations to boost accuracy and efficiency [24].

II.14.1 CNN-based Segmentation

CNNs have also demonstrated strong performance in segmentation tasks. After the record-breaking performance of AlexNet in 2012, we have got many state-of-the-art models of semantic segmentation and instance segmentation. Some of them are highlighted below [25].

II.14.1.1 Deep learning based semantic Segmentation methods

a) **DeepLab:** Deep convolutional neural networks (CNNs) have shown great potential in semantic segmentation, but they come with two main issues: downsampling, which reduces image detail, and spatial invariance, which makes it harder to capture fine structures. To solve the first problem, the authors used a technique called atrous convolution, which adds spaces (or holes) between filter elements to preserve resolution. To deal with the second issue, they added a Conditional Random Field (CRF) to help the model better capture detailed edges and object boundaries. The first version of DeepLab reached a solid 71.6% IoU on the PASCAL VOC 2012 test set. Later, DeepLabv2 improved on this by introducing Atrous Spatial Pyramid Pooling (ASPP), which became its main advancement over the original version [25].

b) **SegNet:** deep learning model specifically developed for semantic segmentation, aiming to assign a category label to every pixel within an image.

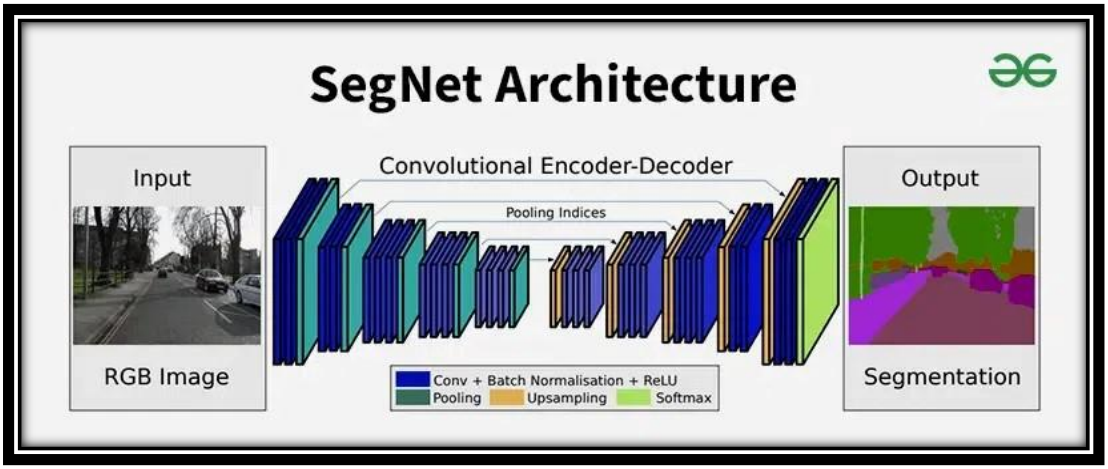


Figure (II.15): SegNet architecture.

It follows an encoder-decoder architecture that enables accurate pixel-level segmentation, making it well-suited for tasks requiring fine-grained detail. By learning to classify each pixel according to its semantic category, SegNet offers a detailed interpretation of image content. This architecture is particularly valuable in domains such as autonomous driving, medical imaging, and urban scene analysis, where high segmentation precision is essential [26].

c) **U-Net:** U-Net is a powerful convolutional neural network architecture specifically designed for image segmentation, particularly in medical imaging. Introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net gained recognition by winning the Cell Tracking Challenge at ISBI 2015. The architecture is shaped like a "U", consisting of two main parts: a contracting path (encoder) and an expanding path (decoder). The encoder performs downsampling using convolution and max-pooling operations, reducing the spatial dimensions of the feature maps.

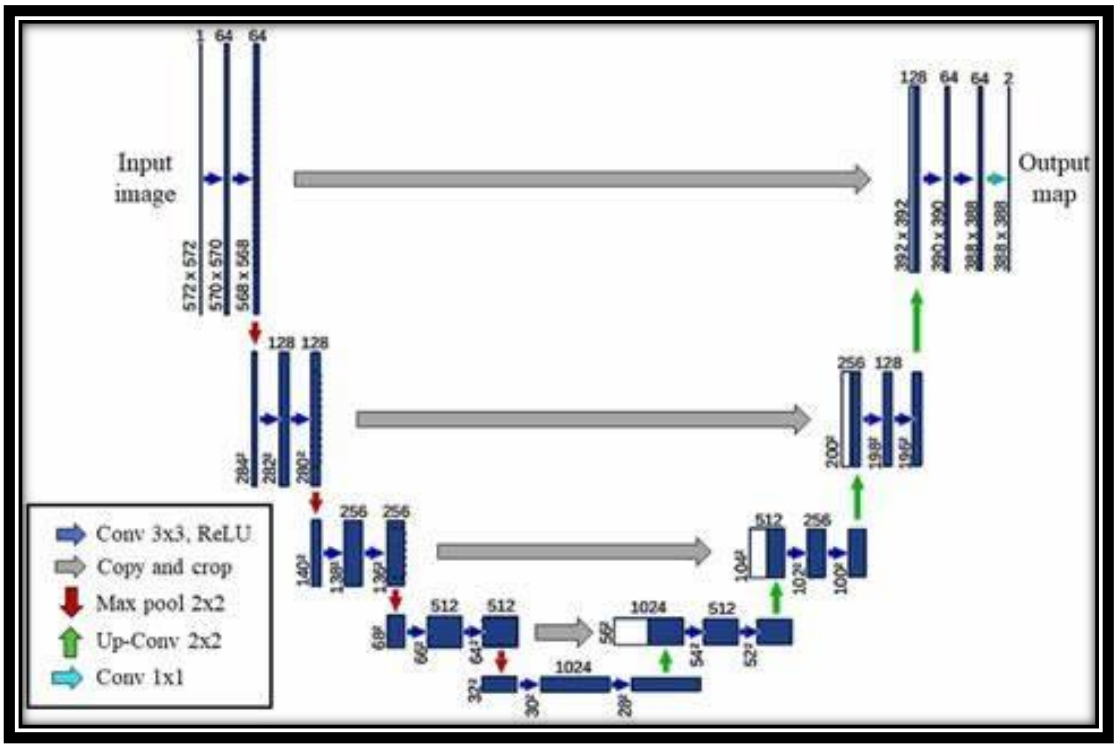


Figure (II.16): U-Net architecture.

Each convolution uses a 3×3 kernel without padding and is followed by a ReLU activation, which causes a slight reduction in spatial size at each layer. In the decoder, the network upsamples the compressed feature maps to reconstruct spatial information and produce segmentation masks. The final output represents class predictions for each pixel. The loss function typically used is the pixel-wise categorical cross-entropy across the whole image.

One detail of U-Net is that the input and output dimensions differ. For instance, an input of 572×572 results in an output of 388×388. During training, the network compares this output to the corresponding 388×388 center patch of the ground truth image. This approach enables effective data augmentation—by cropping multiple 572×572 patches from larger images (e.g., 1024×1024), the model can be trained on more samples, even when the dataset is limited [21].

d) **Fully Convolutional Network (FCN):** Convolutional neural networks (CNNs), known for their success in classification and object detection, have been extended to more complex tasks like semantic segmentation and scene parsing, which involve per-pixel classification. Traditional methods used region-based features classified by SVMs. To overcome their limitations, Long et al. proposed Fully Convolutional Networks (FCNs), which remove fully connected layers to allow inputs of variable sizes. FCNs introduced deconvolution (upsampling) layers to restore spatial resolution, enabling accurate pixel-wise predictions by combining intermediate feature maps [23].

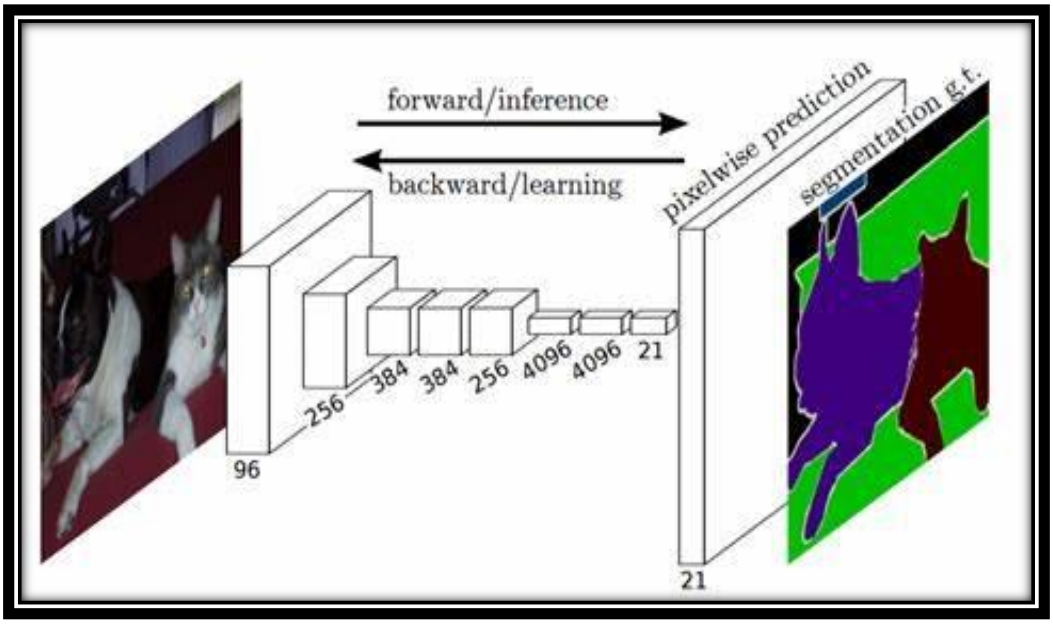


Figure (II.17): FCN architecture.

II.14.1.2 Deep learning based Instance segmentation methods

a) Mask R-CNN: mask R-CNN is an instance segmentation technique which locates each pixel of every object in the image instead of the bounding boxes. It has two stages: region proposals and then classifying the proposals and generating bounding boxes and masks. It does so by using an additional fully convolutional network on top of a CNN based feature map with input as feature map and gives matrix with 1 on all locations where the pixel belongs to the object and 0 elsewhere as the output.

b) **CenterMask:** This framework adopts an anchor-free, single-stage approach to instance segmentation, built upon a convolutional neural network (CNN) backbone. It integrates Feature Pyramid Networks (FPN) to facilitate accurate predictions across multiple spatial scales.[27]

II.15 Transfer learning (TL)

Transfer learning refers to the reuse of a pre-trained model on a new but related task. It is a widely adopted technique in deep learning, especially when labeled data is scarce or when training a deep network from scratch is computationally infeasible. Rather than initializing model parameters randomly, transfer learning allows the model to begin with weights learned from large-scale datasets such as ImageNet, thus accelerating convergence and improving generalization.

There are two main approaches in transfer learning: feature extraction, where the pre-trained model acts as a fixed feature extractor, and fine-tuning, which involves continuing the training of some or all of the model's layers to adapt them to the new task. Fine-tuning typically focuses on higher-level layers, as lower layers tend to capture generic features like edges or textures that are often reusable.

However, effective application of transfer learning requires careful consideration. Factors such as the learning rate, compatibility of architectures, and which layers to fine-tune can significantly influence performance. It is common to reuse standard architectures such as ResNet or VGG to ensure architectural consistency and ease integration.

In the context of semantic segmentation for autonomous driving, transfer learning plays a crucial role. Pixel-level annotation is costly and time-consuming, making it difficult to build large segmentation datasets. As a result, many state-of-the-art segmentation models—like U-Net and DeepLab—use pre-trained encoders from image classification tasks, enabling robust performance even with limited data [28].

II.15.1 Image net

The ImageNet dataset has played a pivotal role in advancing deep learning research, particularly in the field of computer vision. Introduced by Krizhevsky, Sutskever, and Hinton [29], ImageNet consists of over a million labeled images spanning 1,000 object categories. Their groundbreaking work demonstrated how deep convolutional neural networks (CNNs) could be effectively trained on this large-scale dataset, significantly outperforming traditional methods in image classification tasks.

This success marked a turning point in the adoption of deep learning architectures, leading to the widespread use of ImageNet-pretrained models as a foundation for transfer learning in various vision applications. By leveraging pretrained weights from ImageNet, models can achieve faster convergence and improved accuracy, especially when the target task suffers from limited labeled data, such as in semantic segmentation.

II.16 Hyper parameters

II.16.1 Learning rate (LR)

It is a crucial hyperparameter in gradient descent (GD) that determines the step size taken during the training of neural networks. Choosing an appropriate learning rate significantly impacts both the training efficiency and the accuracy of the model. While many practitioners select the highest learning rate that still allows convergence to speed up training, using a value that is too large can degrade generalization and reduce final accuracy. On the other hand, very small learning rates may improve stability but result in unnecessarily long training times without meaningful gains in accuracy. Exhaustively searching through a wide range of learning rates to identify the optimal one is often computationally expensive, particularly in large-scale problems involving complex networks and big datasets. As a result, selecting an effective constant learning rate remains a challenging yet essential task to balance training time and model accuracy [30, 31].

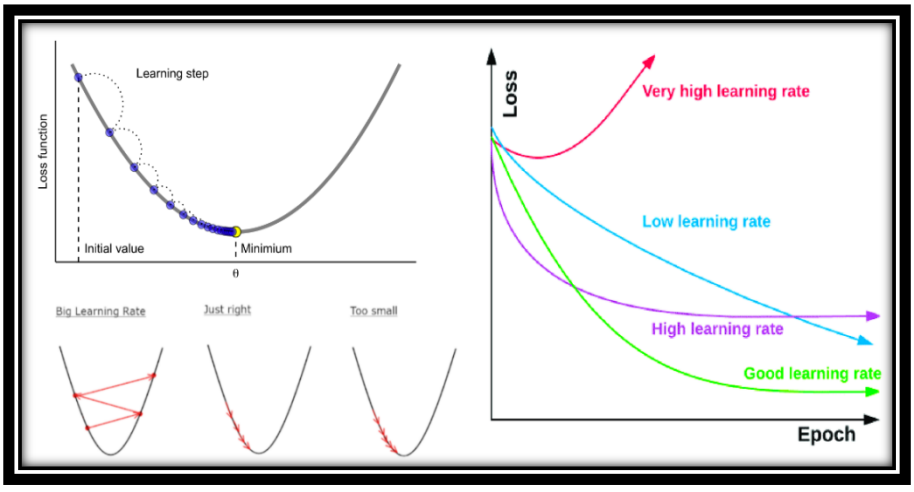


Figure (II.18): Illustrative effects of learning rate on training dynamics.

II.16.2 Epochs

The number of epochs is a key hyperparameter in neural network training that determines how many times the learning algorithm will iterate over the entire training dataset.

Each epoch allows the model to update its internal parameters based on all training samples, typically processed in smaller subsets called batches. Training involves nested loops: an outer loop over epochs and an inner loop over batches. A high number of epochs—often in the range of tens to thousands—is commonly used to ensure sufficient learning and error minimization. Monitoring model performance across epochs using learning curves (plots of error versus epoch count) helps identify underfitting, overfitting, or appropriate convergence[32].

II.16.3 Batch size

Batch size is a key hyperparameter in neural network training that defines the number of samples processed before the model parameters are updated. The training process consists of multiple epochs, where each epoch represents one full pass over the training data. Within each epoch, the data is divided into batches, and for each batch, the model performs a forward pass, calculates error, and updates the weights using gradient descent. Each batch update corresponds to one iteration. Therefore, the number of iterations per epoch is equal to the total number of training samples divided by the batch size.
Different optimization strategies are defined by the batch size [32]:

- ✓ **Batch gradient descent**: one iteration per epoch (batch size = dataset size).
- ✓ **Stochastic gradient descent**: one iteration per sample (batch size = 1).
- ✓ **Mini-batch gradient descent**: multiple iterations per epoch (1 < batch size < dataset size), and is the most commonly used in practice.

II.16.4 Dropout

Dropout is a regularization technique used in neural networks to improve generalization and mitigate overfitting. It operates by randomly deactivating a subset of neurons or connections during each training iteration based on a predefined probability. This stochastic omission of units prevents the network from becoming overly reliant on specific result, dropout effectively simulates the training of multiple sparse sub-networks. During inference, the full network is restored, typically with scaled weights, yielding a model with improved robustness and reduced overfitting [33].

II.16.5 Optimizer

The optimizer is a fundamental component in the training process of deep neural networks, as it directly influences the efficiency and accuracy of learning. Several optimization algorithms have been proposed to improve convergence speed, stability, and performance, including Stochastic Gradient Descent (SGD), Momentum-based SGD, RMSProp, and Adam. While SGD serves as the foundation for many learning algorithms, it often requires careful tuning and may converge slowly. Momentum and RMSProp enhance this process by introducing stability and adaptive learning rates, respectively. Among these, Adam has emerged as the most widely adopted optimizer in practice. It combines the strengths of Momentum and RMSProp by incorporating both first- and second-moment estimates of gradients. Empirical studies have consistently shown that Adam outperforms most other optimizers in training deep learning models, due to its adaptive learning rate and bias correction mechanisms, which enable fast and robust convergence across diverse tasks [34].

Optimizer	Update Rule	Characteristics
SGD	$\omega_{t+1} = \omega_t - \eta \nabla J(\omega_t)$	Basic gradient update using mini-batches.
SGD + Momentum	$v(t) = \gamma * v(t-1) + \eta * \nabla J(\omega(t))$ $\omega_{t+1} = \omega(t) - v(t)$	Adds momentum to accelerate updates and reduce oscillations.
RMSProp	$E[g^2](t) = \rho * E[g^2](t-1) + (1-\rho) * \left(\nabla J(\omega(t))\right)^2$ $\omega(t+1) = \omega(t) - \frac{\eta}{sqrt(E[g^2](t) + \varepsilon)} ** \nabla J(\omega(t))$	Adaptive learning rate per parameter.
Adam	$m(t) = \beta_1 * m(t-1) + (1+\beta_1) * \nabla J(\omega(t))$ $v(t) = \beta_2 * m(t-1) + (1+\beta_2) * (\nabla J(\omega(t)))^2$ $\hat{m}(t) = m(t)/(1-{\beta_1}^t)$ $\hat{v}(t) = v(t)/(1-{\beta_2}^t)$ $\omega(t+1) = \omega(t) - \eta * \hat{m}(t)/(\text{sqrt}(\hat{v}(t))+\varepsilon)$	Adaptive moment estimation with bias correction.

Table (II.4): Optimizers in Deep Learning.

II.17 Performance metrics

II.17.1 Pixel accuracy (PA)

In the context of semantic segmentation, accuracy typically refers to pixel-wise accuracy It quantifies the overall proportion of correctly predicted pixels over the total number of pixels in the dataset or image.

$$\text{Pixel Accuracy} = \frac{\text{Number of Correctly Classified Pixels}}{\text{Total Number of Pixels}} \quad \text{(II.15)}$$

II.17.2 Intersection-Over-Union and mean IoU

It is a segmentation performance parameter that measures the overlap between two objects by calculating the ratio of intersection and union with ground truth masks. This metric is also known as Jaccard Index and mean IoU is the average over all classes. The class wise IoU values were calculated using Equation:[34]

$$\text{IoU} = \frac{\text{TP}}{\text{TP+FP+FN}} \quad \text{and} \quad \text{mIoU} = \frac{1}{c}\sum_{c=1}^{c} \text{IoU} \quad \text{(II.16)}$$

Where TP denotes true positive, FP denotes false positive, FN denotes false negative and IOU denotes Intersection over union value [35].

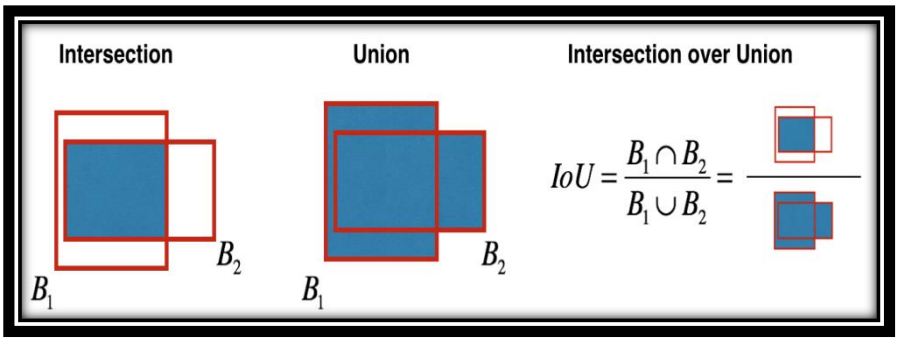


Figure II.19: Intersection-Over-Union.

II.17.3 Dice score

Also referred to as the Dice Coefficient, is a statistical metric used to evaluate the similarity between two sets, most commonly the predicted segmentation output and the corresponding ground truth. It measures the degree of overlap between the two regions and assigns greater weight to correctly predicted pixels compared to the Intersection over Union (IoU) metric.

The Dice Score is particularly valuable in applications requiring precise spatial correspondence, such as in autonomous driving systems, where accurate segmentation of road scenes including elements like lanes, vehicles, and pedestrians is essential for reliable perception and decision-making.

$$\text{Dice} = \frac{2*Prediction \cap ground\ truth}{|Prediction|+|Ground\ truth|} \quad \text{(II.17)}$$

II.17.4 Recall & Precision

Are key evaluation metrics in semantic segmentation, particularly useful when dealing with class imbalance. Precision quantifies the proportion of correctly predicted pixels of a given class among all pixels that the model assigned to that class:

II.17.5 Frames per seconds

Widely used performance metric that assesses the inference speed of a model by measuring the number of image frames it can process per second. In the context of real-time applications such as autonomous driving, FPS plays a critical role, as it directly impacts the system's ability to respond swiftly to dynamic environments. A higher FPS value indicates faster segmentation and shorter decision-making latency, which are essential for ensuring timely and accurate perception in safety-critical scenarios. Consequently, models designed for real-time image segmentation must balance accuracy with computational efficiency to maintain an adequate FPS for practical deployment.

II.17.6 Over fitting & under fitting

Overfitting and underfitting are two common issues that affect the generalization ability of deep learning models. Overfitting occurs when a model learns the training data too well, including noise and irrelevant patterns, resulting in high performance on training data but poor accuracy on unseen data. In contrast, underfitting happens when the model is too simple to capture the underlying structure of the data, leading to poor performance on both training and validation sets. To mitigate overfitting, various regularization techniques can be employed, one of which is early stopping. Early stopping monitors the model's performance on a validation set during training and halts the training process once the validation loss stops improving. This prevents the model from continuing to fit the training data at the cost of generalization. As such, early stopping serves as a practical and effective strategy for improving the robustness of deep learning models, especially in real-time or resource-constrained applications.

II.18 Regularization and Training Stabilization Techniques

II.18.1 Data augmentation

Data augmentation is a widely used technique to reduce overfitting by artificially increasing the size and diversity of the training dataset. It involves applying a variety of transformations to input images such as flipping, cropping, rotation, scaling, and color jittering while preserving their labels. This encourages the model to generalize better by exposing it to multiple variations of the same object or scene. In image segmentation, data augmentation is especially valuable for training deep models with limited annotated data, such as in autonomous driving or medical imaging applications [36].

II.18.2 Batch Normalization

Batch Normalization (BN) is a training technique that normalizes the activations of intermediate layers within mini-batches during training. It stabilizes learning by reducing internal covariate shift and helps accelerate convergence. Mathematically, for a batch xxx, BN transforms the input as:

$$\hat{x} = \frac{x - \mu_{batch}}{\sqrt{\sigma^2_{batch} + \varepsilon}} \quad \text{(II.18)}$$

Where μ_{batch} and σ^2_{batch} are the batch mean and variance, and ε is a small constant for numerical stability. BN improves training dynamics, allows higher learning rates, and introduces a mild regularization effect [37].

II.18.3 L2 Regularization

L2 regularization, also known as weight decay, is a penalty term added to the loss function to discourage large weights. It reduces overfitting by constraining the model complexity. The modified objective function becomes:

$$J(\theta) = J_{original}(\theta) + \lambda \sum_i \theta_i^2 \quad \text{(II.19)}$$

Where λ is a hyperparameter $\lambda \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ that controls the strength of the regularization. Smaller weights encourage simpler and more generalizable models. L2 regularization is typically used in combination with dropout and batch normalization to improve robustness.

II.19 Conclusion

This chapter has provided a comprehensive overview of the foundational principles of deep learning as a significant subfield within artificial intelligence. It explored the architecture and learning paradigms of artificial neural networks, including key components such as perceptron, loss functions, gradient descent, and the backpropagation algorithm. Furthermore, it examined the distinctions between traditional machine learning and deep learning approaches, emphasizing the enhanced representational capacity and performance of deep models. Special attention was given to convolutional neural networks (CNNs), which serve as a cornerstone in visual data processing and segmentation tasks. The theoretical foundations laid out in this chapter form the conceptual and technical groundwork for the deep learning methods employed in the subsequent chapters of this thesis.

References

[1] H. Booth, M. Souppaya, A. Vassilev, M. Ogata, M. Stanley, and K. Scarfone, Secure Software Development Practices for Generative AI and Dual-Use Foundation Models (An SSDF Community Profile), NIST Special Publication 800-218A, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2024. [Online].

[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016

[3] "Deep Learning Process Data Ppt Powerpoint Presentation Professional Summary," PowerPoint Slides Diagrams, [Online].

[4] K. I. M. Al-Malah, Machine and Deep Learning Using MATLAB: Algorithms and Tools for Scientists and Engineers, 1st ed, 2024

[5] L. Bahamid, Système de Détection des Défauts et des Ralentisseurs sur les Chaussées par Vision Artificielle, Master thesis, 2024.

[6] A. Thomas, An introduction to neural networks for beginners (Adventures in Machine Learning).

[7] P. Kim, MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence, 2017.

[8] R. Qamar and B. A. Zardari, "Artificial Neural Networks: An Overview," 2023.

[9] M. Nielsen, Neural Networks and Deep Learning.

[10] M. Rahman, Perceptrons, June 2023

[11] M. H. SALZI, A brief review of feed-forward neural network, 2006.

[12] C. Aggarwal, Neural Networks and Deep Learning overview.

[13] J. R. Terven, D. M. Córdova-Esparza, A. Ramírez-Pedraza, E. A. Chávez-Urbiola, and J. A. Romero-González, "A comprehensive survey of loss functions and metrics in deep learning," Artificial Intelligence Review, Apr. 15, 2025. [Online]. Available: https://doi.org/10.1007/s10462-025-11198-7

[14] Loss Functions and Their Use In Neural Networks," Towards Data Science

[15] D. L. Mestre, "Gradient Descent: An Algorithm for Deep Learning Optimisation," Empathy.co | Medium, https://medium.com

[16] T.-Y. Lin et al., "Focal loss for dense object detection," in Proc. ICCV, 2017, pp. 2980–2988.

[17] NEURAL NETWORKS Unit-5-AIML, PDF document Unit-5-AIML.pdf.

[18] "Neural networks and deep learning," [Online]. https://www.deeplearningbook.org.

[19] H. S. Chatterjee, "A Basic Introduction to Convolutional Neural Network," Jul. 16, 2019

[20] J. Brownlee, Deep Learning With Python: Develop Deep Learning Models On Theano And TensorFlow Using Keras, 2016.

[21] S. Pattanayak, Pro Deep Learning with TensorFlow: A Mathematical Approach to Advanced ArtificialIntelligence in Python. Apress, 2017

[22] K. Dahmane, Analyse d'images par méthode de Deep Learning appliquée au contexte routier en conditions météorologiques dégradées, Ph.D. thesis, 2020

[23] M. Hassaballah and A. I. Awad, Deep Learning in Computer Vision: Principles and Applications. Springer,2020.

[24] "Advanced CNN Architectures for Computer Vision Tasks," Learn Computer Vision, [Online]. https://learnopencv.com

[25] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, "Fundamental Concepts of Convolutional Neural Network," Dept. of Computer Science, University of Gour Banga; A.K. Choudhury School of IT, University of Calcutta; Dept. of CSE, M.A.K.A.U.T., India. January 2020.

[26] "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," GeeksforGeeks, https://www.geeksforgeeks.org

[27] A. Mittal, "Instance segmentation using Mask R-CNN," Medium, [Online]. Available: https://medium.com

[28] A. Garcia-Garcia, S. Orts-Escolano, S. O. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," arXiv preprint arXiv:1704.06857v1, Apr. 22, 2017.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012.

[30] A. Ezzeldin, M. Saeed, S. Elfeky, and A. Khater, "Neural Network with Adaptive Learning Rate," in Proc. 2nd Novel Intelligent and Leading Emerging Sciences (NILES 2020), Cairo, Egypt, Oct. 2020. doi: 10.1109/NILES50944.2020.9257880

[31] D. R. Wilson and T. R. Martinez, "The need for small learning rates on large problems," in Proc. Int. Joint Conf. Neural Networks (IJCNN), vol. 1, Washington, DC, USA, Feb. 2001, pp. 115–119. doi: 10.1109/IJCNN.2001.939002

[32] J. Brownlee, "What is the Difference Between a Batch and an Epoch in a Neural Network?", Machine Learning Mastery, Jul. 20, 2018.

[33] S. H. Khan and R. Iqbal, “A comprehensive survey on architectural advances in deep CNNs: Challenges, applications, and emerging research directions,” Artificial Intelligence Lab, Dept. of Computer Systems Engineering, University of Engineering and Applied Sciences (UEAS), Swat, Pakistan.

[34] I. Pointer, Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications, 1st ed. Sebastopol, CA, USA: O’Reilly Media, 2019.

[35] A. Sagar and R. Soundrapandiyan, “Semantic segmentation with multi-scale spatial attention for self-driving cars,” in Proc. IEEE/CVF Int. Conf. on Computer Vision Workshops (ICCVW), Montreal, QC, Canada, Oct. 2021, pp. 2650–2656.

[36] L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning”, 2017.

[37] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Machine Learning (ICML), 2015, pp. 448–456.

[38] A. Krogh and J. A. Hertz, “A simple weight decay can improve generalization,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 4, 1992.

Chapter III

Implementation And Experimental Results

III.1 Introduction

In this chapter, we present the practical implementation of our semantic segmentation system, which was developed as part of an internship carried out at the Research Centre in Industrial Technologies (CRTI) in Cheraga, Algiers. This internship allowed us to gain hands-on experience in a professional research environment and benefit from expert guidance provided by our supervisor Dr.Benyahia Ahmed. We also had access to valuable computational resources and technical feedback, which greatly supported the development, training, and evaluation phases of our deep learning models.

The primary goal of this work is to design a robust semantic segmentation pipeline that achieves a high level of accuracy while maintaining excellent real-time performance. Specifically, we focused on optimizing two key metrics: mean Intersection over Union (mIoU), which evaluates segmentation quality, and Frames Per Second (FPS), which reflects the inference speed crucial for real-time applications such as autonomous driving.

The sections that follow detail each step of our methodology, including dataset preprocessing, model architecture selection, training configurations, evaluation procedures, and result analysis. We put particular emphasis on the performance of U-Net architectures with different encoder backbones and demonstrate how our final model using EfficientNet-B0 achieves a strong balance between accuracy and speed, developed and trained on the Cityscapes dataset using Google Colab.

III.2 Development Environment

III.2.1 Tools and Platforms

All experiments were carried out using Google Colab, a cloud-based development platform provided by Google that allows users to write and execute Python code through Jupyter Notebooks. One of the key advantages of Colab is its free access to GPUs (e.g., Tesla T4 or P100), which significantly accelerates deep learning model training and testing.

Google Colab supports direct integration with Google Drive, making it easy to manage large datasets and save outputs. It also comes pre-installed with many popular libraries and frameworks used in AI and machine learning. Its collaborative features, such as notebook sharing and commenting, made it a practical and accessible environment for the development of this project, especially in the absence of powerful local hardware.

Figure (III.1): Logo of Google Colab [1].

III.2.2 Libraries and Frameworks

Our project was implemented using the Python 3.10 programming language, and a suite of essential libraries and frameworks for deep learning and image processing. These tools were used to define the model, handle data, perform training, and visualize results. The table below summarizes the key libraries used, along with their specific versions and roles:

Library	Version	Purpose
PyTorch	2.0.1+cu117	Main deep learning framework used to implement and train the U-Net architecture.
Torchvision	0.15.2+cu117	Provided dataset utilities and image transformations.
NumPy	2.0.2	Numerical operations, matrix manipulation, and array handling.
Matplotlib	3.10.0	Visualizing training curves and segmentation results.
PIL (Pillow)	11.2.1	Image loading, resizing, and processing.
OpenCV	4.11.0	Additional image processing tasks (e.g., color conversion, resizing).
Albumentations	1.3.0	Applied powerful and flexible image augmentations during training

Table (III.1): Main Libraries used in the Implementation Environment.

These libraries were selected due to their stability, flexibility, and compatibility with the PyTorch. Together, they provided the necessary foundation to build, train, and evaluate the U-Net model effectively for the semantic segmentation task on the Cityscapes dataset.

III.3 Dataset Description

In our thesis work, we used the Cityscapes dataset [2], which provides high-resolution street-view images (original size: 2048×1024 pixels) from 50 cities in Germany, captured in various lighting, weather, and traffic conditions. These images are paired with fine-grained pixel-level annotations, making the dataset ideal for training and evaluating semantic segmentation models in the context of autonomous driving.

III.3.1 Dataset Structure and Pre-processing

The dataset of Cityscapes provides:

- **LeftImg8bit** the RGB input images.
- **GtFine** the high-quality ground truth segmentation masks.

We worked specifically with the leftImg8bit folder for the input images and the gtFine folder for the corresponding labels. Due to hardware limitations on Google Colab, we were unable to work with the original resolution of 2048×1024 pixels. Instead, we resized the images to a more manageable resolution of 512×256 pixels, allowing faster training while still preserving sufficient detail for meaningful segmentation results.

III.3.1.1 Custom Dataset Split

While the official dataset structure includes 2975 training**,** 500 validation, and 1525 test images [3], we chose not to use the official test set. This is because Cityscapes does not provide ground truth annotations for test images-only the Cityscapes server evaluates them.

To overcome this limitation, we manually split the 500 official validation images into:

- A new validation set (for tuning hyperparameters)
- A custom test set (for final evaluation)

The following table summarizes our dataset split:

Subset	Source	Number of Images	Purpose
Training	leftImg8bit/train	2975	Model training
Validation	leftImg8bit/val	250	Hyperparameter tuning
Test (custom)	leftImg8bit/val	250	Final model evaluation

Table (III.2): Manually Defined Split of Cityscapes Dataset

III.3.1.2 Annotation Classes

Cityscapes provides fine-grained annotations for 30 object classes [2], but only 19 classes are used for semantic segmentation evaluation in most published research [4]. We followed this standard 19-class setup to align with state-of-the-art benchmarks and to ensure that our results would be comparable with other academic works in the field, the following table represents the standard 19 classes used by most segmentation models trained on Cityscapes:

Group	Classes
Flat	Road, sidewalk
Construction	Building, wall, fence
Object	Pole, traffic light, traffic sign
Nature	Vegetation, terrain
Sky	Sky
Human	Person, rider
Vehicle	Car, truck, bus, train, motorcycle, bicycle

Table (III.3): The standard 19 classes of Cityscapes.

III.4 Model Architecture

In our work on this thesis, we adopted the U-Net architecture as the base for our semantic segmentation model. U-Net is a well-established deep learning model it has proven highly effective in different domains, including autonomous driving. We choosed this model for its simple yet powerful encoder–decoder structure, which allows the model to capture fine-grained details while maintaining computational efficiency also because it offers a favorable trade-off between accuracy and speed, making it a suitable foundation for our real-time segmentation objectives.

It follows an encoder-decoder structure, where the encoder progressively reduces the spatial dimensions of the input image to extract semantic features, while the decoder reconstructs the segmentation mask at the original resolution.

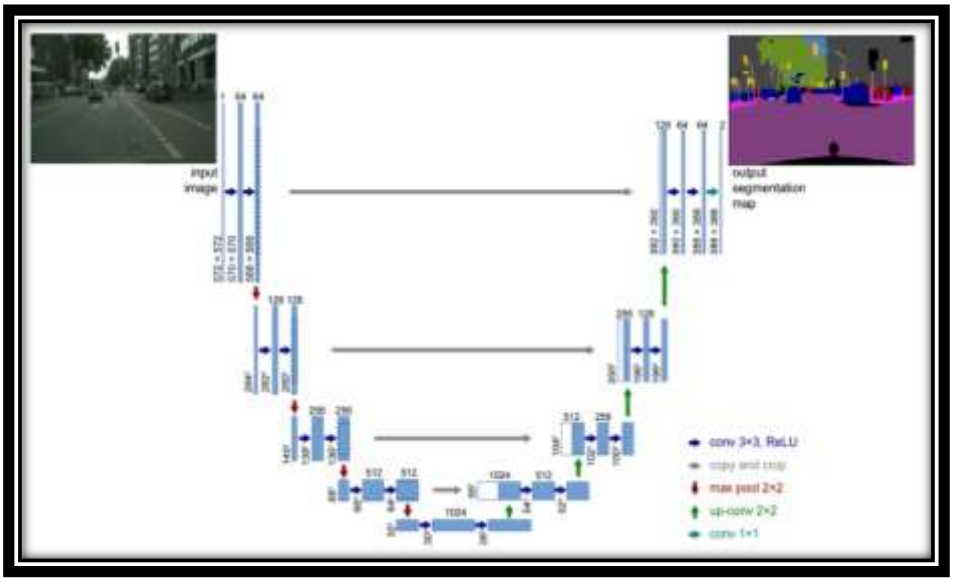


Figure (III.2): The diagram of the U-Net architecture.

To improve the model's efficiency while maintaining strong segmentation accuracy, we used EfficientNet-B0 as the encoder backbone. EfficientNet is a family of convolutional neural networks designed to balance performance and speed through a compound scaling method. The B0 variant offers a lightweight yet expressive feature extractor, making it especially suitable for real-time applications and resource-constrained environments. This choice was particularly aligned with our goal of achieving high inference speed without sacrificing segmentation quality. We used a version pre-trained on ImageNet to benefit from transfer learning and accelerate convergence during training. In addition, the compact architecture of EfficientNet-B0 makes it a strong candidate for deployment on embedded systems, where real-time, accuracy and computational efficiency are critical.

The decoder part follows the original U-Net structure, with skip connections linking the encoder and decoder stages at corresponding resolutions. These connections help preserve spatial information and improve the sharpness of the segmentation masks. The final layer of the network outputs a pixel-wise classification over the 19 semantic classes defined in the Cityscapes dataset.

III.5 Implementation Details

III.5.1 Data Loading and Preprocessing

Due to the dataset's size (more than 11 GBs) and to facilitate access during training on Google Colab, we stored all our data in Google Drive. This allowed us to mount the dataset directly into the Colab environment and interact with it seamlessly.

The following code snippet shows how we connected Google Drive to our working environment:

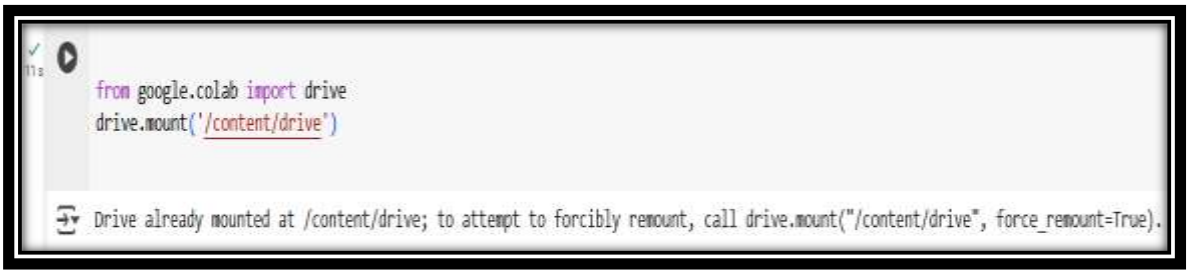


Figure (III.3): Mounting Google Drive into Google Colab.

III.5.2 Training Configuration

In this experiment, we trained our semantic segmentation model over the course of 100 epochs. Although the training script included an early stopping mechanism with a patience of 20 epochs which represent the number of consecutive epochs without improvement in validation loss after which training would be stopped. This mechanism was not triggered, as the validation loss continued to improve throughout training, and this indicates that the model maintained a stable learning curve with no signs of overfitting, which allowed the full training process to complete as scheduled. The consistent improvement in validation metrics confirmed that our data augmentation strategy such as horizontal flipping and random cropping, model architecture, and encoder choice were well-balanced, enabling the model to generalize effectively without early termination.

The training was performed using a batch size of 8, which was chosen as a compromise between available GPU memory and gradient stability. The optimizer used was Adam, known for its effectiveness in computer vision tasks, and the initial learning rate was set to 1e-3. We also applied a learning rate scheduler (ReduceLROnPlateau) to dynamically adjust the learning rate when the validation loss plateaued.

The loss function used was CrossEntropyLoss, which is appropriate for multi-class segmentation problems where each pixel is assigned to a single class. In addition, L2 regularization was applied via the weight decay parameter to encourage model generalization. The table below summarizes all the key hyperparameters used in our training configuration:

Hyperparameter	Value
Epochs	100
Batch size	8
Optimizer	Adam
Learning Rate	0.001
Loss Function	CrossEntropyLoss
Scheduler	ReduceLROnPlateau
Weight Decay	0.00001
Augmentation	Yes (Albumentations)
EarlyStopping	Yes

Table (III.4): Training Configuration and Hyperparameters.

III.6 Evaluation Metrics

In order to assess the performance of the semantic segmentation models developed in this thesis, we rely on standard evaluation metrics widely used in the field of deep learning projects, namely Pixel Accuracy, Mean Intersection over Union (mIoU), Dice Score, and Frames Per Second (FPS), as discussed in Chapter 02. These metrics allow us to quantify how accurately the model predicts the class of each pixel in the input image.

III.7 Results and Discussion

III.7.1 Comparative Analysis of U-Net Backbone Architecture

During the development of this thesis, we explored several encoder backbones for our U-Net-based model including ResNet-50, ResNet-101, EfficientNet-B5 and EfficientNet-B0 in order to select the most suitable model architecture for our real-time image segmentation task. Each of these models was trained under the same conditions using the Cityscapes dataset. We evaluated their performance based on the previous critical metrics. We evaluated their performance based on the previous critical metrics.

Among the various U-Net-based models evaluated on the Cityscapes dataset, the U-Net architecture enhanced with an EfficientNet-B0 encoder clearly stands out as the most robust, accurate, and reliable. The EfficientNet-B0 variant outperformed the alternative backbones we tested across all key evaluation metrics. It achieved a mean Intersection over Union (mIoU) of 61.8% and a test accuracy of 92.3%, confirming its ability to produce highly precise and consistent segmentation results. In addition, the model delivered an outstanding 128 FPS, making it not only accurate but also extremely fast.

This combination of high accuracy and real-time speed demonstrates that the EfficientNet-B0-based U-Net is exceptionally well-suited for demanding applications like autonomous driving, where both performance and efficiency are critical. Therefore, it can be considered the most reliable and efficient solution among the tested models for real-time semantic segmentation tasks.

III.7.1.1 Training Behavior and Convergence

Before presenting the final evaluation metrics, we first illustrate the model's learning dynamics throughout the training process. The following graphs show the training and validation accuracy and loss curves across 100 epochs:

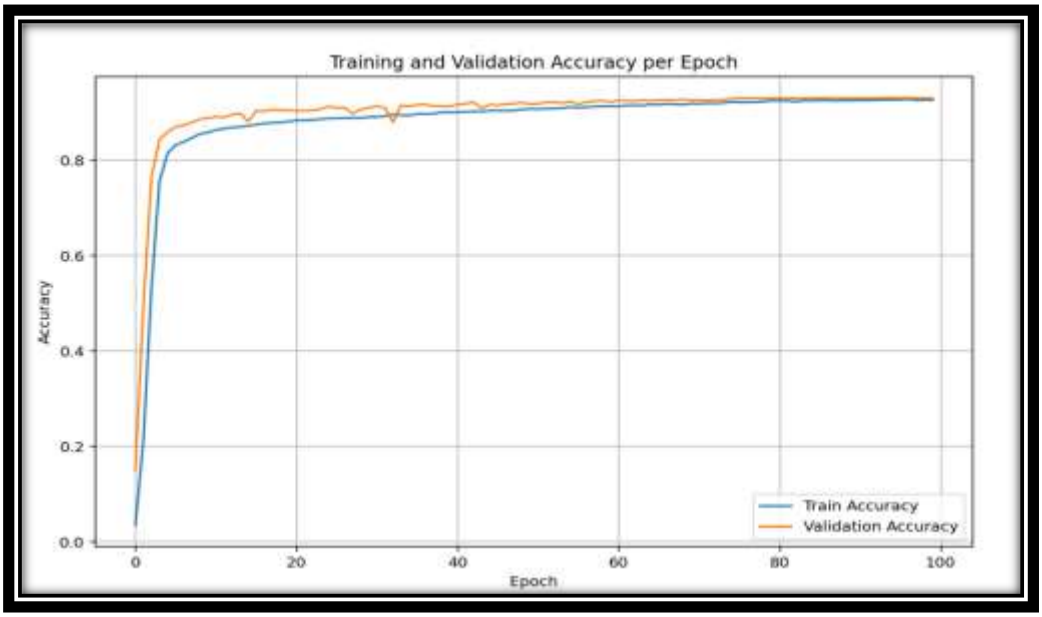


Figure (III.4): Training and Validation Accuracy over 100 Epochs.

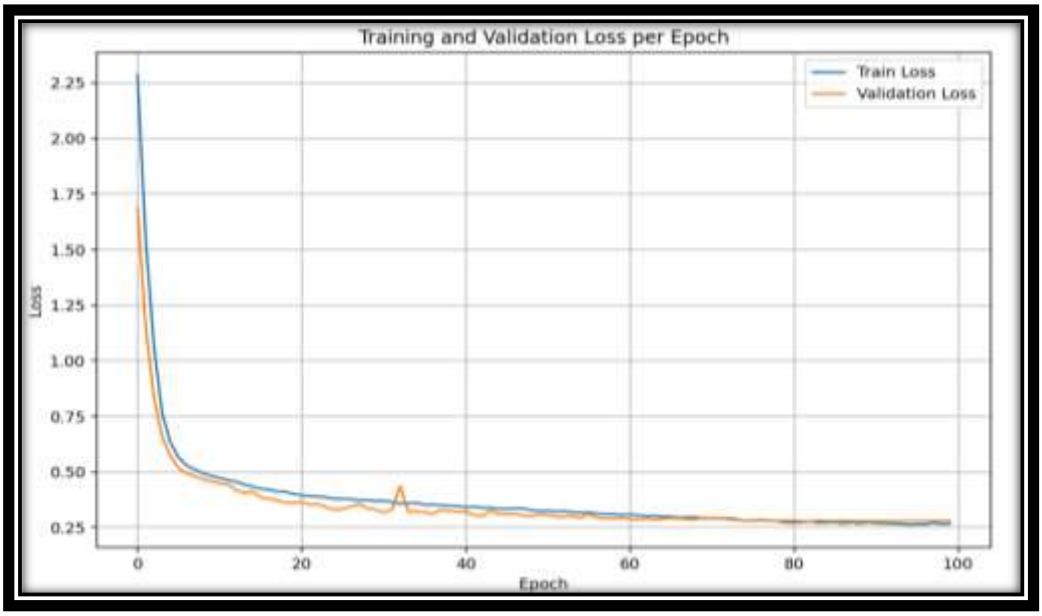


Figure (III.5): Training and Validation Loss over 100 Epochs.

As observed, both accuracy and loss remained well-aligned between the training and validation sets, which strongly indicate that the model did not overfit.

Although the training script included an early stopping mechanism with a patience of 20 epochs, it was never triggered. The model continued to improve steadily until the final epoch, confirming a smooth and stable convergence. This consistent performance during training laid the foundation for the strong test results we present in the next section.

III.7.2 Quantitative Results

In this section, we present the quantitative evaluation of our image segmentation model, which is based on the U-Net architecture with an EfficientNet-B0 encoder. The results reflect the model's performance after training and testing on the Cityscapes dataset. To provide a comprehensive understanding, we include multiple evaluation metrics and visual summaries such as the confusion matrix, per-class Dice scores, Intersection over Union (IoU), and the overall segmentation report.

III.7.2.1 Confusion Matrix

The confusion matrix offers a visual representation of how well the model distinguished between different semantic classes in the Cityscapes dataset. Each row represents the actual class, while each column indicates the predicted class [5].

Figure (III.6): Confusion Matrix of our Model.

This matrix illustrates how accurately the model classified each semantic class, with a strong diagonal trend indicating high agreement between predicted and ground truth labels. The presence of minimal off-diagonal values highlights the model's ability to correctly distinguish between similar classes such as "road", "sidewalk", and "car", which is essential in real-world urban scene understanding for autonomous driving.

III.7.2.2 Dice Coefficient per Class

As illustrated in the figure below, the Dice coefficient per class confirms the strong performance of our model, it achieved outstanding scores in key semantic classes such as road, building, and car, where the Dice scores exceeded 92.1%

These classes are among the most critical for autonomous driving systems, and achieving such high overlap indicates that the model has effectively learned to segment large and well-defined structures.

Even in more challenging and fine-grained categories such as pole or traffic sign, the model maintained consistent and respectable scores, reinforcing its robustness. Overall, these results demonstrate that our model performs with high reliability across diverse urban scene elements, which is essential for real-time perception in autonomous vehicles.

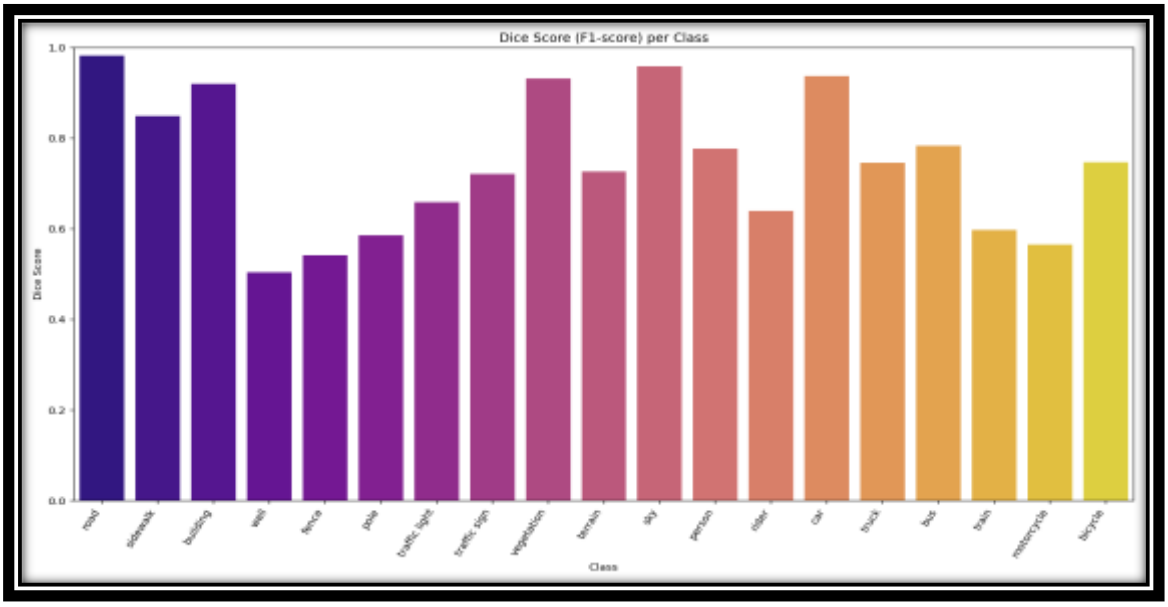


Figure (III.7): The Dice Coefficient per Class of our Model.

III.7.2.3 Intersection over Union (IoU) per Class

The figure displaying the IoU per class offers further insight into the model's segmentation quality. It achieved IoU values above **0.90** for several essential categories. These results are particularly significant given the complexity and variability of urban scenes in the Cityscapes dataset.

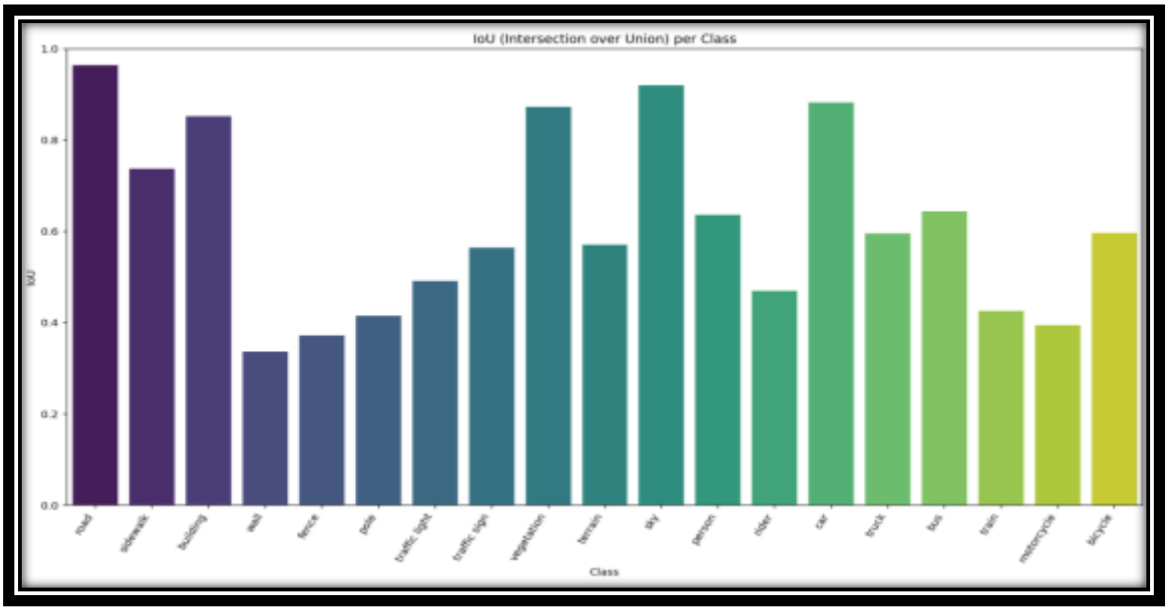


Figure (III.8): Bar Chart of IoU Performance for Each Semantic Class.

However, the overall mean IoU is 61.8 %, In real-time semantic segmentation for autonomous driving, achieving a mean Intersection over Union (mIoU) above 50% is generally considered a functional and acceptable threshold [6-10]. And this is a strong indicator of balanced performance across all classes which confirm the model's reliability for real-world applications. As illustrated by the mean IoU graph :

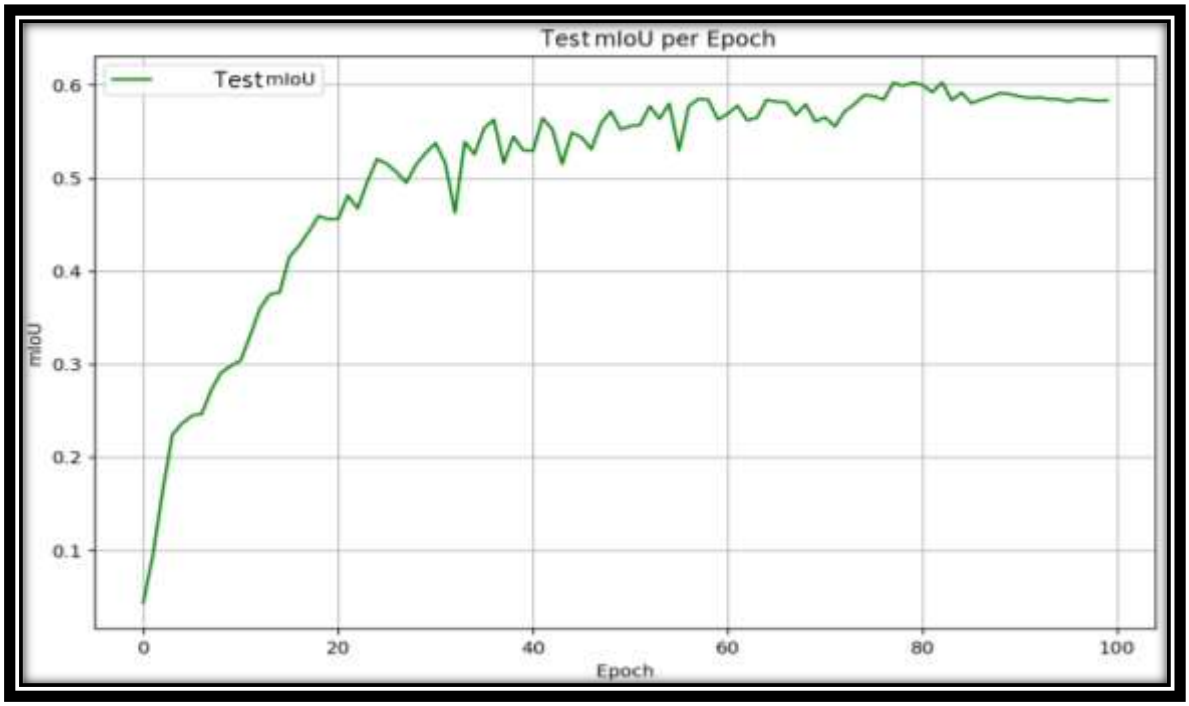


Figure (III.9): mean Intersection over Union over 100 Epochs.

III.7.2.4 Inference Speed (FPS) Comparative Study

In real-time systems such as autonomous driving, inference speed is a critical requirement. A model must process frames fast enough to allow immediate reaction to the surrounding environment. Generally, achieving more than 30 FPS is considered acceptable for real-time applications [11-15] however, surpassing 128 FPS is a significant milestone, demonstrating not only real-time capability but also efficiency and scalability. One of the core objectives of this thesis was to design a model that balances high segmentation accuracy with fast inference speed. Our final U-Net model, powered by an EfficientNet-B0 encoder, reached an impressive inference speed of 128.22 FPS, even while maintaining high accuracy and mIoU scores.

This result confirms that the proposed model is not only precise in its pixel-level predictions but also optimized for deployment in real-time scenarios such as autonomous vehicles, where both speed and reliability are non-negotiable.

III.7.2.5 Comparative Study

To further validate the efficiency of our proposed model, we conducted a comparative study against several well-known semantic segmentation networks evaluated on the Cityscapes dataset. We focused on two key performance metrics**:** mean Intersection over Union (mIoU) and inference speed (FPS)**.**

The table below summarizes the results obtained from the literature and our own model. As shown, our approach based on U-Net with an EfficientNet-B0 encoder achieves superior balance between segmentation accuracy and real-time inference capability. It outperforms popular lightweight models like ENet**,** ESPNet**,** and SegNet**,** offering higher mIoU and significantly faster processing speed, making it more suitable for real-time embedded systems such as autonomous vehicles

Model	Dataset	mIOU (%)	FPS	Source
Our Model (U-net+EfficientNet-B0)	Cityscapes	**61.8**	**128.2**	Our thesis 2025
ENet	Cityscapes	58.3	76.9	Paszke et al. [16,17]
SegNet	Cityscapes	56.1	53	Badrinarayanan et al. [18,19]
ESPNet	Cityscapes	60.3	112	Mehta et al. [20]

Table (III.5): mIoU and FPS comparison on Cityscapes dataset.

III.7.2.6 Segmentation Classification Report

The segmentation classification report provides a detailed summary of key evaluation metrics such as Precision, Recall, and F1-score (Dice) for each class. As shown in the report, the model consistently delivers Precision and Recall values above 90% in dominant classes such as road, building, and car, demonstrating both its accuracy and sensitivity in detecting these features.

The F1-score (or Dice) values follow a similar trend, reflecting strong agreement between the predicted and true segmentation masks. These metrics highlight the model's ability to generalize well across different scenes and minimize both false positives and false negatives. Importantly, the report also confirms a balanced performance across classes, without a significant drop in underrepresented categories a clear strength of using the EfficientNet-B0 backbone, which extracts richer multi-scale features and handles class imbalance more effectively.

	precision	recall	f1-score	support
road	0.9835	0.9804	0.982	10798044.0
sidewalk	0.8391	0.8592	0.849	1629635.0
building	0.9043	0.9372	0.9205	5994127.0
wall	0.6712	0.4031	0.5037	198376.0
fence	0.6198	0.4813	0.5418	301028.0
pole	0.5942	0.5783	0.5862	471999.0
traffic light	0.6566	0.6605	0.6585	62306.0
traffic sign	0.7838	0.6681	0.7214	207851.0
vegetation	0.9396	0.9249	0.9322	5005130.0
terrain	0.7215	0.7314	0.7264	282683.0
sky	0.9535	0.9634	0.9584	1005083.0
person	0.753	0.8035	0.7774	391556.0
rider	0.6385	0.6401	0.6393	69535.0
car	0.9346	0.941	0.9377	1716495.0
truck	0.7418	0.7512	0.7465	75983.0
bus	0.8497	0.7273	0.7838	128322.0
train	0.7631	0.4905	0.5972	38982.0
motorcycle	0.6537	0.4988	0.5658	28036.0
bicycle	0.7027	0.7985	0.7476	217506.0
accuracy	0.923	0.923	0.923	0.923
macro avg	0.7739	0.7283	0.7461	28622677.0
weighted avg	0.9219	0.923	0.9219	28622677.0

Figure (III.10): The Segmentation Report of our Model.

III.7.3 Qualitative Results and Visualization

In this section, we present qualitative results to visually assess the performance of our proposed model. The following figures illustrate segmentation outputs produced by the U-Net architecture with an EfficientNet-B0 encoder on sample images from the Cityscapes dataset. Each example compares the input image, the ground truth mask, and the predicted segmentation map. These visualizations allow us to verify how accurately the model captures

fine details such as road boundaries, cars, and pedestrians. The results demonstrate that our model is capable of generating high-quality, precise segmentation masks that are consistent with the annotated ground truths. The following figures represent some of the obtained results that demonstrate the efficiency of our model:

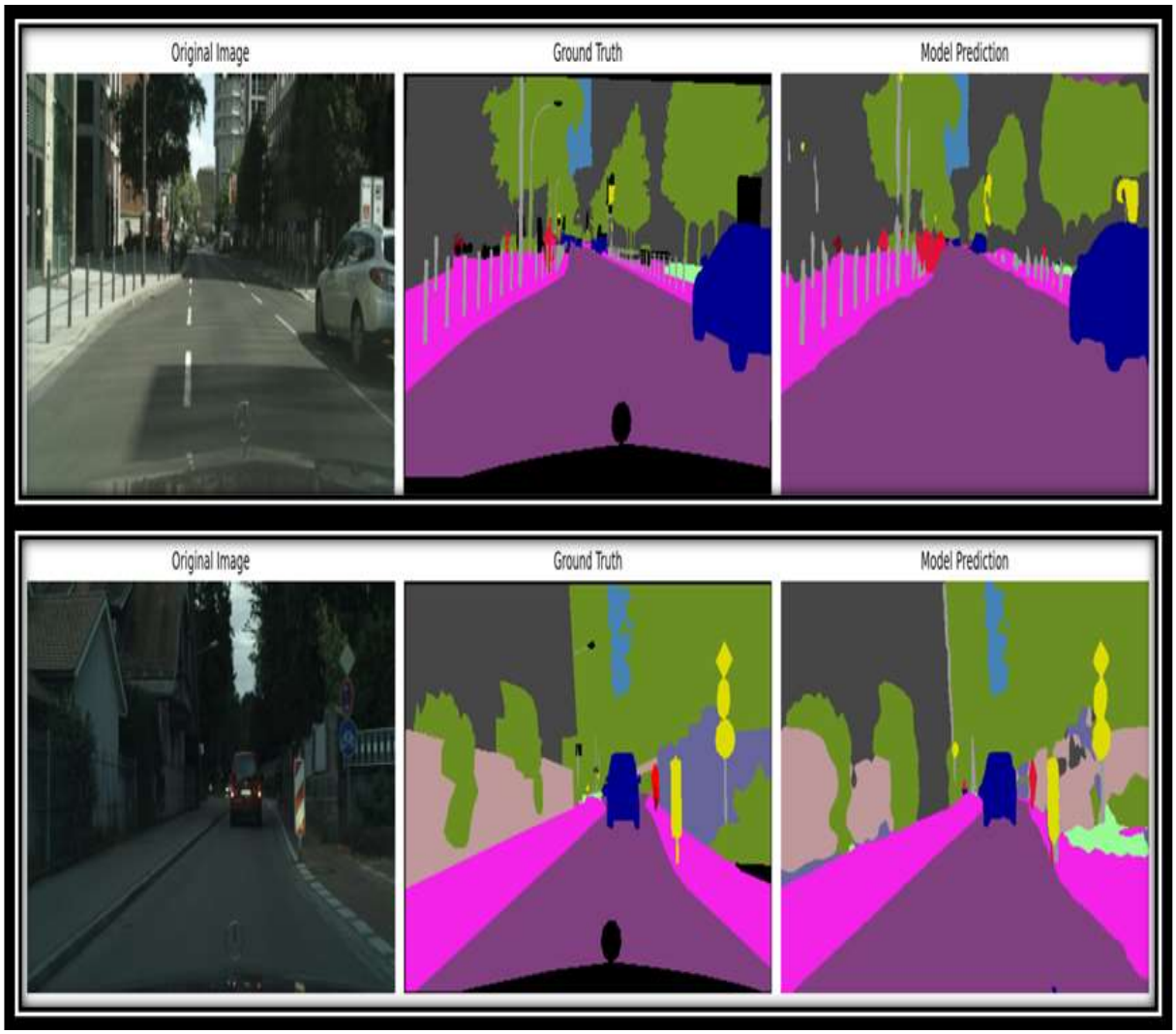


Figure (III.11): The Visual Comparison.

It's important to note that semantic segmentation on high-resolution, real-world images like those in the Cityscapes dataset is a complex task. Factors such as occlusion, small objects, and varying lighting conditions make perfect segmentation challenging, even for advanced models. Some small segmentation errors are visible, particularly in fine structures, but the global scene understanding remains accurate.

III.8 Challenges and Limitations

Throughout the development of this thesis, several challenges emerged. This section outlines the key difficulties faced during the project:

III.8.1 Hardware Constraints and Limited Resources

Due to limited computational resources, we had to resize high-resolution Cityscapes images, which led to the loss of fine spatial details. Although we conducted part of the training at CRTI, the short internship period was not sufficient to fully leverage the available infrastructure.

III.8.2 Class Imbalance in the Dataset

The dataset contains highly unbalanced class distributions, which affected the model's ability to accurately segment underrepresented categories such as poles, traffic signs, and pedestrians.

III.8.3 Real-Time Constraints vs. Accuracy Trade off

Achieving both high segmentation accuracy and real-time performance is a difficult balance. While our model achieves fast inference speeds (FPS > 100), optimizing for speed sometimes meant compromising on finer-level accuracy. This tradeoff is inherent in real-time systems, particularly in the autonomous driving domain.

III.8.4 Lack of Annotations in the Official Test Set

The official Cityscapes test set lacks annotations, requiring us to create a custom test set from the validation data. This limited our ability to benchmark against standardized test results.

III.8.5 U-Net Architectural Limitations

The U-Net architecture, although effective, has difficulty capturing long-range dependencies and complex context in crowded urban scenes, which can lead to misclassification of small or visually similar classes.

III.9 Conclusion

In this chapter, we presented the full implementation pipeline of our deep learning approach for real-time semantic segmentation using the Cityscapes dataset. We explored multiple U-Net-based architectures with different encoder backbones and conducted a comparative study to evaluate their performance across various metrics such as accuracy, mean IoU, Dice score, and inference speed (FPS).

Among the tested models, the U-Net with EfficientNet-B0 encoder demonstrated the most reliable performance, achieving a strong balance between segmentation quality and real-time inference capabilities. The model consistently delivered high accuracy, robust class-wise

metrics, and an impressive frame rate exceeding 128 FPS, making it well-suited for real-time applications such as autonomous driving.

Despite the limitations encountered including hardware constraints, dataset challenges, and architectural trade-offs the proposed system achieved promising results. The quantitative and visual evaluations confirmed that our model is not only efficient but also scalable and adaptable to real-world scenarios.

This practical investigation provides a solid foundation for further improvements and integration of advanced techniques such as attention mechanisms, multi-scale fusion, and post-processing strategies in future work.

Refrences

[1] H. Libre, “Colaboratoire Google : Qu'est-ce que c'est et à quoi ça sert,” HWLibre, https://www.hwlibre.com/fr/colaboratoire-google/ (accessed Jun. 12, 2025).

[2] M. Cordts et al., “The Cityscapes Dataset: Semantic Urban Scene Understanding,” [Online]. Available: https://www.cityscapes-dataset.com/

[3] M. Cordts et al., “The Cityscapes Dataset for Semantic Urban Scene Understanding,” Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016.

[4] A. Sagar and R. Soundrapandiyan, “Semantic segmentation with multi-scale spatial attention for self-driving cars,” in Proc. IEEE/CVF Int. Conf. on Computer Vision Workshops (ICCVW), Montreal, QC, Canada, Oct. 2021, pp. 2650–2656.

[5] Q. Geng, X. Huang, Z. Zhou, and R. Yang, “A network structure to explicitly reduce confusion errors in semantic segmentation”, 2018

[6] A. Rosebrock, “Intersection over Union (IoU) for object detection,” PyImageSearch, Nov. 7, 2016. [Online].Available: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection

[7] D. Shah, “Intersection over Union (IoU): Definition, Calculation, Code,” V7 Labs, May 30, 2023. [Online]. Available: https://www.v7labs.com/blog/intersection-over-union-guide/

[8**]** Viso.ai, “Intersection over Union (IoU) Explained: Everything You Need to Know,” Viso Suite Blog, 2023. [Online]. Available: https://viso.ai/computer-vision/intersection-over-union-iou/

[9]FinanceBand, “How much IoU is good?,” FinanceBand AI Blog, 2023. [Online]. Available: https://financeband.com/how-much-iou-is-good

[10]Picsellia, “COCO Evaluation Metrics Explained,” Picsellia Blog, 2023. [Online]. Available: https://www.picsellia.com/post/coco-evaluation-metrics-explained

[11] Aiphile, “Detecting Face at 30 FPS on CPU on MediaPipe Python,” *Medium*, May 2023. [Online]. Available: https://medium.com/@aiphile/detecting-face-at-30-fps-on-cpu-on-mediapipe-python-dda264e26f20

[12] Stack Overflow, “In image processing, what is real time?,” *Stack Overflow*, Sep. 2010. [Online]. Available: https://stackoverflow.com/questions/3957376/in-image-processing-what-is-real-time

[13] Ultralytics, “Understanding the Role of FPS in Computer Vision,” *Ultralytics Blog*, 2023. [Online]. Available: https://www.ultralytics.com/blog/understanding-the-role-of-fps-in-computer-vision

[14] Cloudinary, “Video Frame Rates Explained: Along with Tips for Picking the Right FPS,” *Cloudinary Guides*, 2023. [Online]. Available: https://cloudinary.com/guides/video-formats/video-frame-rates-explained-along-with-tips-for-picking-the-right-fps

[15] Roboflow, “What is FPS in Computer Vision?,” *Roboflow Blog*, 2023. [Online]. Available: https://blog.roboflow.com/what-is-fps

[16]A. I. Khennane and M. Ghanbari, “Multi-scale attention-based encoder-decoder network for semantic image segmentation,” IET Computer Vision, vol. 18, no. 4, pp. 191–204, Apr. 2024. [Online]. Available: https://doi.org/10.1049/cvi2.12178

[17]A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation,” *arXiv preprint arXiv:1606.02147,* 2016. [Online]. Available: https://arxiv.org/abs/1606.02147

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder–decoder architecture for image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2481–2495, Dec. 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2644615

[19] H. Chen, Z. Xiao, B. Ge, and X. Li, “LMANet: A Lightweight Asymmetric Semantic Segmentation Network Based on Multi-Scale Feature Extraction,” Electronics, vol. 13, no. 17, p. 3361, 2024. [Online]. Available: https://doi.org/10.3390/electronics13173361

[20]S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, “ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation,” in *Proc. ECCV,* 2018. [Online]. Available: https://arxiv.org/abs/1803.06815

GENERAL CONCLUSION

General Conclusion

This thesis has explored deep learning-based approaches for real-time image segmentation in the field of autonomous driving, a domain where both accuracy and speed are essential for safety and operational efficiency. The study focused primarily on the U-Net architecture due to its proven success in semantic segmentation tasks, particularly in capturing fine-grained object boundaries.

To enhance the model's adaptability to complex road scenes, we modified the encoder of the original U-Net and replaced it with EfficientNet-B0, known for its balance between computational efficiency and representational power. This adaptation significantly boosted the model's performance on the Cityscapes dataset, which is specifically tailored for urban driving environments.

Throughout this work, we made several key adjustments to overcome dataset-related limitations, notably the lack of ground truth annotations in the official test set, which led us to create a custom test split. Despite these challenges, our main objective remained clear: to design a segmentation model that maintains strict real-time performance while ensuring reliable segmentation quality.

This objective was successfully met. The final model achieved an inference speed of 128 FPS, which exceeds the threshold typically required for real-time applications, and a mean Intersection over Union (mIoU) above 61.8%**,** a level that is generally accepted for operational deployment in real-world scenarios.

The implementation was carried out on Google Colab, and the internship conducted at the Research Centre in Industrial Technologies (CRTI) in Chéraga - Algiers, played an instrumental role in shaping the results. The hands-on experience, guidance from experts, and access to high-performance hardware positively influenced the model's development and evaluation.

Furthermore, the application of data augmentation techniques contributed to improved generalization and segmentation quality. However, we believe that even greater performance could be achieved with further optimization and more advanced augmentation strategies a direction we recommend for future research.

In conclusion, this work demonstrates that with careful model selection, architectural refinement, and training strategy, it is possible to develop a deep learning solution that meets the dual requirements of accuracy and real-time efficiency, making it a promising candidate for deployment in autonomous vehicle perception systems**.**