

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Université 8 Mai 45 -Guelma
Faculty of Science and Technology
Department of Electronics and Telecommunications



End of study dissertation
For obtaining the Academic master's degree
Domain : Science and Technology
Sector : Electronics
Specialty : Instrumentation

**Autonomous Vehicle with Jetson Nano:
An Approach for Automated Navigation**

Presented by :

SELAIMIA YASSINE

Under the Supervision Of :
Dr. TABA Mohamed Tahar

2023

THANKING

First, I thank ALLAH, our one, and merciful GOD, for giving me the health and

courage to complete this modest work.

*My sincerest thanks to Dr. TABA Mohamed Tahar for his availability,
contributions, valuable guidance and understanding throughout the
development of this dissertation.*

*I would also like to extend my gratitude and appreciation to the **Laboratory of
Automatic and Computer Science of Guelma (LAIG)** for their invaluable
support throughout my research journey.*

*I address my sincere thanks to the members of the jury for accepting,
examining, and evaluating this modest work. also, thanks to all my teachers
from the Electronics department.*

Selaimia Yassine

DEDICATION

I humbly dedicate this modest work to my beloved family and my two little nieces Jouri and Jana, whose unwavering support and boundless love have been a source of inspiration and strength throughout my journey. In particular, I express heartfelt gratitude to my dear parents, who have selflessly provided me with everything I needed, without expecting anything in return. Their unwavering belief in me has been instrumental in shaping my life. May ALLAH, the Almighty, One and Merciful, bless you with good health and long life.

To all my friends and colleagues who have always been there for me. Their unconditional support and encouragement have been a great help.

Thank you

الملخص

تقدم هذه الأطروحة تطوير مركبة ذاتية القيادة باستخدام Nvidia Jetson Nano™ (SBC) كوحدة تحكم مركزية من خلال الاستفادة من الرؤية الحاسوبية وتقنيات التعلم العميق. الهدف هو تحقيق مركبة صغيرة الحجم ذات قدرة على التنقل في طريق بشكل مستقل وكشف العقبات في الوقت الفعلي. جهاز Jetson Nano™ مسؤول عن التحكم في أنظمة السيارة، بما في ذلك محركات DC المستخدمة في القيادة. يتم الحصول على البيانات من خلال تطبيق وحدة الكاميرا IMX219، والتي تلتقط الصور لتدريب نماذج الشبكة العصبية. من خلال الاستفادة من قوة التعلم النقلي، يتم استخدام نموذج ResNet-18 لتدريب طبقته المتصلة بالكامل المخصصة لمجموعة البيانات الخاصة بنا. يتم نشر النموذج المدرب على لوحة Jetson Nano للتشغيل في الوقت الفعلي، مع النتائج التفصيلية وعرض أداء السيارة المستقلة المطورة.

الكلمات الرئيسية: الذكاء الاصطناعي، المركبة المستقلة، الرؤية الحاسوبية، التعلم العميق، Nvidia Jetson Nano™، مجموعة البيانات، نقل التعلم.

Abstract

This dissertation presents the development of an autonomous vehicle using a Nvidia Jetson Nano™ (SBC) as the central control unit by leveraging on computer vision and deep learning techniques. The objective is to realize a small-scale vehicle with the capability of navigating a road autonomously and detecting obstacles in real-time. The Jetson Nano™ device is responsible for controlling the vehicle's systems, including the DC motors used for the drive. Data acquisition is achieved through the application of the IMX219 camera module, which captures images for training neural network models. By leveraging the power of transfer learning, the ResNet-18 (ANN) model is utilized for training its custom fully connected layer for our specific dataset. The trained model is deployed on the Jetson Nano™ board for real-time operation, with the detailed results and showcasing the performance of the developed autonomous vehicle.

Keywords: Artificial Intelligence, Autonomous vehicle, Computer vision, Deep learning, artificial neural network, Nvidia Jetson Nano™, Transfer learning, Dataset.

RÉSUMÉ

Cette mémoire présente le développement d'un véhicule autonome utilisant un Nvidia Jetson Nano™ (OM) comme unité centrale de contrôle en tirant parti de la vision informatique et des techniques d'apprentissage profond. L'objectif est de réaliser un véhicule à petite échelle avec la capacité de naviguer une route de manière autonome et de détecter les obstacles en temps réel. L'appareil Jetson Nano™ est chargé de contrôler les systèmes du véhicule, y compris les moteurs à courant continu utilisés pour l'entraînement. L'acquisition de données est réalisée grâce à l'application du module caméra IMX219, qui capture des images pour la formation de modèles de réseaux neuronaux. En tirant parti de la puissance de l'apprentissage par transfert, le modèle ResNet-18 (RNA) est utilisé pour former sa couche entièrement connectée sur mesure pour notre ensemble de données spécifique. Le modèle formé est déployé sur la carte Jetson Nano™ pour un fonctionnement en temps réel, avec les résultats détaillés et la présentation des performances du véhicule autonome développé.

Mots-clés: Intelligence artificielle, Véhicule autonome, Vision par ordinateur, L'apprentissage en profondeur, réseau neuronal artificiel, Nvidia Jetson Nano™, Apprentissage par transfert, Base de données

.

TABLE OF CONTENTS

List of Figures	IX
List of Tables	XI
GLOSSARY	XII
GENERAL INTRODUCTION.....	1
CHAPTER 1 STATE OF THE ART	3
1.1 Introduction	4
1.2 Autonomous Systems.....	4
1.2.1 Autonomous Systems Definition.....	4
1.2.2 Advantages and Disadvantages of Automation.....	5
1.2.3 Intelligent Automation.....	5
1.3 Autonomous Vehicles	5
1.3.1 Autonomous Vehicles Definition	5
1.3.2 Historical Development of Autonomous Vehicles.....	5
1.3.3 Types of Automated Vehicles	7
1.3.3.1 Self-Driving Cars	8
1.3.3.2 Autonomous Mobile Robots (AMRs)	8
1.3.4 Levels of Automation SAE J3016 Standard.....	8
1.3.5 Current State	10
1.3.6 Anticipated Impacts of Autonomous Vehicles	10
1.3.6.1 Safety.....	10
1.3.6.2 Traffic congestion and Fuel consumption	11
1.3.6.3 Travel Behavior Impacts	11
1.3.7 Autonomous vehicles Technologies	11
1.3.7.1 Sensing and Perception	12
1.3.7.2 Decision Making	13
1.4 Conclusion.....	13
CHAPTER 2 COMPUTER VISION AND DEEP LEARNING.....	14
2.1 Artificial Intelligence	15
2.1.1 Definition of Artificial Intelligence	15
2.1.2 Historical Development of Artificial Intelligence	15
2.1.3 Types of Artificial Intelligence.....	16
2.1.4 Applications of Artificial Intelligence	17
2.1.5 ways AI can be archived.....	18
2.1.6 Advantages & Disadvantages of Artificial Intelligence	19
2.2 Computer Vision	19
2.2.1 Computer Vision Definition	19
2.2.2 Computer Vision tasks.....	20
2.2.3 Digital Image Definition.....	20
2.2.4 Computer Vision Libraries	22
2.2.4.1 OpenCV — Evolution in Computer Vision	22

2.3	Machine Learning and Deep Learning Components.....	22
2.3.1	Machine Learning.....	23
2.3.1.1	Fields of Machine Learning	23
2.3.1.2	Types of Machine Learning	23
2.3.1.3	Major Challenges Faced by Machine Learning	26
2.3.2	Deep Learning	26
2.4	Basics of Artificial Neural Networks (ANN's)	26
2.4.1	Artificial Neural Network Definition	26
2.4.2	Artificial Neural Network Architecture.....	27
2.4.3	Activation Functions.....	27
2.4.4	Types of Perceptron models	28
2.4.5	Artificial Neural Network Structure	28
2.4.6	Mathematical Model of ANN.....	29
2.5	Convolutional Neural Network	29
2.5.1	Convolutional Neural Network Definition	29
2.5.2	Layers in a CNN	30
2.5.3	Basic Architecture of LeNet-5 CNNs.....	32
2.6	Tools and Frameworks	34
2.6.1	PyTorch	34
2.6.2	ResNet-18	35
2.7	Conclusion.....	36
CHAPTER 3	REALIZATION OF THE PROJECT	37
3.1	Introduction	38
3.2	System Architecture	38
3.2.1	Hardware Architecture	38
3.2.1.1	NVIDIA Jetson Nano.....	40
3.2.1.2	CSI Camera.....	41
3.2.1.3	PCA9685 Driver	42
3.2.1.4	L298N Motor Driver.....	43
3.2.1.5	Power	44
3.2.1.6	The Vehicle.....	44
3.2.2	Hardware Assembly	45
3.3	Technologies and software installations	46
3.3.1	Setup the Jetson Nano.....	46
3.3.2	Configure System	47
3.3.2.1	Jupyterlab.....	47
3.3.2.2	Adjusting PCA9585 address Frame.....	48
3.4	Conception Methodology of the Autonomous Vehicle.....	50
3.4.1	The Experimental Environment.....	50
3.4.2	Road Navigation	51
3.4.2.1	Data Collection	51
3.4.2.2	Training.....	52
3.4.3	Collision Avoiding	55
3.4.3.1	Data collection	55
3.4.3.2	Training.....	55
3.4.4	Models Deployment	57

3.4.4.1 PD Controller feedback System.....	60
3.5 Experimental Results.....	61
3.6 Problems Faced in the Implementation and their solutions	65
3.7 Improving Performance of the Autonomous Vehicle prototype.....	66
3.8 Limitation of the vehicle	67
3.9 Conclusion.....	67
General Conclusion.....	68
REFERENCES.....	70

LIST OF FIGURES

Figure	Page
1.1 Francis.houdina remote controlled car.....	6
1.2 James Adams' "Stanford Cart"	6
1.3 Tsukuba Engineering automated car.....	6
1.4 Carnegie Mellon's NavLab 5.....	7
1.5 Pal-Robotics "Aran AMR".	8
1.6 J3016TM "Levels of Driving Automation".....	9
1.7 State of Avs in 2023.....	10
1.8 Autonomous Vehicles cyclic process	11
1.9 Autonomous Vehicles technologies.....	12
 2.1 Types of Artificial Intelligence	 16
2.2 Kismet and Sophia	17
2.3 Applications of Artificial Intelligence	17
2.4 ways AI can be archived	18
2.5 Human vision system vs computer vision system	19
2.6 Computer vision common tasks.....	20
2.7 Pixel Map for a handwritten digit	21
2.8 RGB image channels.....	21
2.9 3 RGB (4D) tensor representation	21
2.10 AI, ML and DL	22
2.11 The four different Machine Learning algorithms	23
2.12 Overview of supervised learning	24
2.13 Overview of Unsupervised learning	24
2.14 Machine Learning Full Map	25
2.15 Biological neuron.....	26
2.16 Artificial neural network Structure	27
2.17 Common ANNs activation functions.....	28
2.18 Architecture of a single and a multilayered Perceptron's	28
2.19 Classification of feed-forward and recurrent feedback NNs architectures	29
2.20 Outline of CNN.....	30
2.21 Convolution Layer	31

2.22	CNN Pooling layer.....	31
2.23	The Architecture of LeNet-5 ANNs	32
2.24	LeNet-5 Fully Connected Network.....	34
2.25	ResNet-18 five blocks Architecture	36
3.1	System Hardware Architecture.....	38
3.2	NVIDIA® Jetson Nano™ 2GB Developer Kit	40
3.3	GPIO of Jetson Nano™	41
3.4	IMX219-77 Camera	41
3.5	PCA9685 I2C Driver inputs.....	42
3.6	PCA9685 I2C address.....	42
3.7	Dual H Bridge Motor DriverL298N.	43
3.8	Lithium-polymer power bank and 2S18650 Li-ion cells.....	44
3.9	Vehicle Chassis Kits	44
3.10	Autonomous Vehicle Wiring diagram.....	45
3.11	Autonomous Vehicle Prototype fully assembled.....	45
3.12	JupyterLab view.....	48
3.13	i2c bus check command	48
3.14	Autonomous Vehicle Differential Drive.....	49
3.15	Obstacle object and the test track.....	50
3.16	Camera view for collecting datasets with actual Vehicle position.	51
3.17	Samples of Dataset.....	52
3.18	Road Navigation Training Flowchart.	53
3.19	The training procedure for road Navigation ANN's Model.	55
3.20	Data collection with Actual vehicle position on track.	55
3.21	Collision Avoiding Training Flowchart.....	56
3.22	Flowchart of autonomous vehicle control.....	58
3.23	Proportional-derivative (PD) steering controller feedback system.....	61
3.24	Experiment Result.....	62
3.25	Models result at Different Scenario.	64
3.26	Look ahead point distance effect.	67

LIST OF TABLES

Table	Page
1.1 Advantages and Disadvantages of Automation.	5
1.2 Categorization of Autonomous Vehicles by Operating Environment.	7
1.3 Automobile crashes in ALGERIA in 2022.	10
2.1 Usage of Machine Learning.....	23
3.1 Jetson Nano™ and Raspberry Pi 4 technical specifications.	39
3.2 Jetson Nano™ Technical specifications	40
3.3 IMX219 Camera parameters.....	42

GLOSSARY

AI: Artificial Intelligence.

ML: Deep Learning.

DL: Machine Learning.

AV's: Autonomous Vehicle.

AMRs: Autonomous Mobile Robots.

UVs: Unmanned Vehicles.

CNN: Convolutional Neural Network.

NNs: Neural Networks.

SAE: Society of Automotive Engineers.

ADAS: Advanced Driver Assistance Systems.

GPS: Global Positioning System.

ANI: Artificial Narrow Intelligence.

AGI: Artificial General Intelligence.

ASI: Artificial Super Intelligence.

NLP: Natural language processing.

MRI: Magnetic Resonance Imaging

ANNs: Artificial Neural Networks

SBC's: Single-Boards Computers.

GPIO: General Purpose Input/Output.

I2C: Inter-Integrated Circuit.

CSI: Camera Serial Interface.

SD-CARD: Secure Digital Card.

HDMI: High-Definition Multimedia Interface.

OpenCV: Open-source Computer Vision.

CPU: Central Processing Unit.

GPU: Graphics Processing Unit.

PWM: Pulse Width Modulation.

LPDDR: Low-Power Double Data Rate.

GENERAL INTRODUCTION

بسم الله الرحمن الرحيم

وَالْخَيْلَ وَالْبِغَالَ وَالْحَمِيرَ لَتَرْكُبُوهَا وَزِينَةٌ وَيَخْلُقُ مَا لَا تَعْلَمُونَ -

صدق الله العظيم

In recent years, autonomous vehicles have emerged as one of the most rapidly advancing areas of technology. The integration of artificial intelligence, computer vision, and deep learning algorithms has paved the way for groundbreaking advancements in automation, enabling vehicles to navigate and make decisions without human intervention. While the main function of a car has traditionally been mobility and transporting people and goods, the development of autonomous vehicles has expanded the scope of interests paving the way for a new era of possibilities.

Road safety remains a critical concern worldwide, with automobile crashes resulting in significant loss of life and property damage. In Algeria, as highlighted in [Table 1.3](#), fatal crashes from 2014 to 2017 attributed over 93.29% to driver errors such as lack of respect for traffic laws, careless driving, and excessive speed. These alarming statistics emphasize the pressing need for innovative solutions to mitigate human error and enhance road safety.

One such promising solution is the adoption of autonomous vehicles, which offer advanced safety features that are designed to assist and support drivers, promoting safer and more comfortable driving experiences.

This dissertation focuses on the development of an autonomous vehicle using a Jetson Nano board as it's the ideal SBC for AI-based computer vision applications. The objective is to design and implement a system that can navigate a road autonomously, while accurately detecting and responding to obstacles in its path. By harnessing the power of computer vision and deep learning techniques and jetson nano computing power, the aim is to achieve robust and reliable performance, paving the way for the integration of autonomous vehicles into real-world environments for the ultimate goal of creating a safer, comfortable driving environment.

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

Chapter 1 STATE OF THE ART: it provides a general overview of autonomous systems and the concept of intelligent automation. It delves into the realm of autonomous vehicles, tracing their historical development and exploring its various types. The chapter also discusses the

levels of automation defined by the SAE J3016 Standard and examines the current state of autonomous vehicles, along with their potential impacts and the technologies involved.

In Chapter 2 COMPUTER VISION AND DEEP LEARNING: the focus shifts towards computer vision and deep learning. It starts with an exploration of artificial intelligence then delves into computer vision, discussing its definition, tasks, and the importance of digital image understanding. It also provides an overview of machine learning and deep learning components, specifically artificial neural networks (ANNs), and convolutional neural networks (CNNs), which form the foundation of the proposed autonomous vehicle system. Tools and frameworks, such as PyTorch and ResNet-18 model, are also discussed.

Chapter 3 REALIZATION OF THE PROJECT: forms the core of the dissertation, as it details the realization of the autonomous vehicle project. It begins with a comprehensive overview of the system architecture, including the hardware components and their assembly. The chapter then delves into the installation of relevant software on the Jetson Nano board. It further presents the conception methodology of the autonomous vehicle, including the experimental environment, road Navigation and collision avoidance techniques and models deployment. Experimental results, encountered problems, and potential solutions are discussed, along with ways for improving the performance of the vehicle and its limitations.

We end our study with a general conclusion and prospectives for future work.

CHAPTER 1 STATE OF THE ART

1.1 Introduction

Throughout history, humans have always looked at different ways to develop automated systems that offer great productivity, comfort, safety, and enhanced quality of life over traditional manual methods. From the earliest ways of transportation, such as animal-drawn carts and horse carriages, to the revolutionary inventions of steam engines and internal combustion engines, it has paved the way for automated means of travel.

Today, advancements in the field of artificial intelligence (AI) and Machine learning algorithms with high-performance computing, have brought in a new era of automation that surpasses any previous capabilities. This phase, known as intelligent automation, harnesses the power of AI algorithms to revolutionize various industries, including the transportation field, particularly in the realm of autonomous vehicles.

1.2 Autonomous Systems

1.2.1 Autonomous Systems Definition

Autonomous systems are technological systems or devices that are capable of operating and making decisions without direct human intervention. It is a system that can achieve a certain set of goals in a changing environment. It gathers information about the unfamiliar environment and acts by itself. Autonomous cars and autonomous mobile robots (AMRs) are two common examples of it. Most current systems are only semi-autonomous rather than fully autonomous. Such as cars with driver support systems, robotic surgery systems, robot vacuums and most unmanned vehicles (UVs).

For a system to achieve autonomy, it necessitates the capability to:

- Operating without direct human intervention.
- Sense the environment by perceiving and understanding from diverse data sources.
- Determine what action to take.

1.2.2 Advantages and Disadvantages of Automation

Automation has become a driving force in a variety of industries, revolutionizing how tasks are carried out. The use of automation technologies is becoming more and more widespread. And, like any major change, automation brings benefits and disadvantages that must be considered.

Advantages	Disadvantages
Increased Efficiency and Productivity	Worker Displacement
Higher Level of Safety	Can Become Unnecessary
Can Replace Human in Risky Tasks	High Initial Investment
Consistency and time saving	Still Needs Human Assistance
Increased Production	Dependency on Technology

Table 1.1: Advantages and Disadvantages of Automation [1].

1.2.3 Intelligent Automation

Intelligent automation is a way to automate tasks, processes, and systems through the combination of AI And Automation. By leveraging on (AI) capabilities such as computer vision, Intelligent Automation enhances traditional automated systems, enabling them to learn, adapt, and make intelligent decisions. One sector where Intelligent Automation can make a difference is in the automobile industry, particularly the development of **autonomous vehicles** [2].

1.3 Autonomous Vehicles

1.3.1 Autonomous Vehicles Definition

Autonomous vehicles (AVs) are a type of vehicles that are capable of sensing their environment and operating with or without human involvement depending on the level of automation. They can be categorized based on SAE's automation levels, from conditional driving automation to fully autonomous ones that could go anywhere without humans' interaction [3].

1.3.2 Historical Development of Autonomous Vehicles

The history and development of autonomous vehicles dates back several decades ago, it reflects the human desire for automated transportation. As early as the 1920s, inventors began envisioning a future where vehicles could operate without human control. In this part we

explore the journey from the early ideas to the present-day advancements in the field of autonomous vehicles:

- In 1925, Francis.Houdina illustrated a radio-controlled car in Manhattan that can drive without anyone controlling the steering wheel.

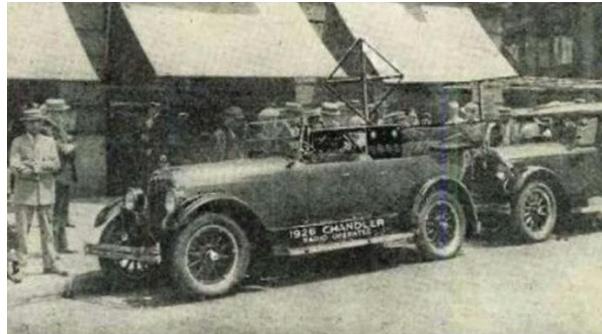


Figure 1.1: Francis.houdina remote controlled car [4].

- In 1961, James Adams created the Stanford Cart, it has a camera with lane following capabilities. It was the first use of cameras in an autonomous vehicle.



Figure 1.2: James Adams' "Stanford Cart" [5].

- In 1977, the Japanese Tsukuba Engineering, with the use of camera system that sends data to a computer to process frames of the road. This led to the world's first self-driving vehicle which was capable of following lane markings and could reach speeds of up to 30 km/h [6].

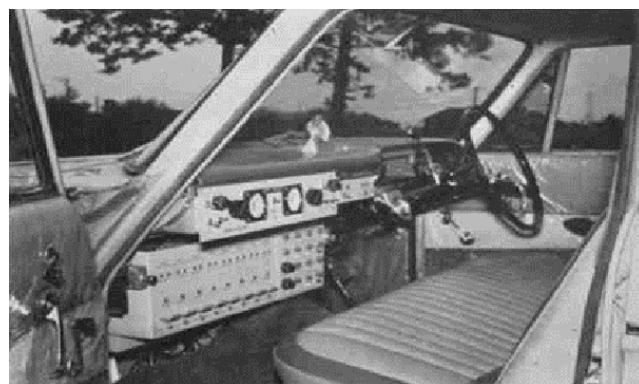


Figure 1.3: Tsukuba Engineering automated car [7].

- In 1995, using live camera feed with **ALVINN** NN's architecture. Carnegie Mellon University researchers took their self-driving car, called NavLab 5, in “No Hands Across America” tour a 4501 km trip. The accelerator and brakes were controlled manually, but the car was otherwise autonomous in terms of steering.



Figure 1.4: Carnegie Mellon’s NavLab 5 [8].

As of 2023, the market leading company in autonomous vehicles is Tesla with their Full Self-Driving package, enabling autonomous driving for highway and low speed city. Also, Waymo (Google self-driving car project) with its autonomous ride-hailing services in Phoenix and San Francisco [6].

1.3.3 Types of Automated Vehicles

By classifying AVs according to their intended operating environment, we can better comprehend their functionalities and their capabilities to specific use cases. The environment in which AVs operate can strongly affect their performance. We can categorize (AVs) as below:

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 1.2: Categorization of Autonomous Vehicles by Operating Environment.

By examining specific examples of autonomous vehicles, such as **self-driving cars** and **Autonomous mobile robots**, we can gain a deeper understanding of their applications, functionalities, and their underlying technologies.

1.3.3.1 Self-Driving Cars

Self-driving cars are type of vehicles that include autonomous capabilities, they use sensors and artificial intelligence to transport people and goods from point A to point B, with different capabilities, from simple automated systems to a fully autonomous cars, as identified by the six levels standards defined by SAE J3016 [9].

1.3.3.2 Autonomous Mobile Robots (AMRs)

Autonomous mobile robots (AMRs) are robots capable of moving around in their environment without an operator, they vary from simple robot floor cleaners to complex ones. They use different sensors to understand their environment and perform tasks in the most efficient way. They are fast and safe and can offer excellent assistance with many manual tasks.



Figure 1.5: Pal-Robotics “Aran AMR” [10].

There are four types of AMRs, and each type is better suited for a different task:

- AMRs for Transportation
- AMRs for Picking
- AMRs for Sorting
- AMRs for Inventory Visibility [11].

1.3.4 Levels of Automation SAE J3016 Standard

SAE International is a global organization of over 128,000 engineers, aerospace, automotive and commercial vehicle industries. **J3016TM** is an SAE International “Levels of Driving Automation” standard that defines the six levels of driving automation for Both passenger and commercial vehicles, from no automation to fully autonomous [12].



SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety		You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat"	When the feature requests, you must drive	These automated driving features will not require you to take over driving
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/acceleration support to the driver	These features provide steering AND brake/acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		These features can drive the vehicle under all conditions
Example Features	<ul style="list-style-type: none"> • automatic emergency braking • blind spot warning • lane departure warning 	<ul style="list-style-type: none"> • lane centering OR adaptive cruise control 	<ul style="list-style-type: none"> • lane centering AND adaptive cruise control at the same time 	<ul style="list-style-type: none"> • traffic jam chauffeur 	<ul style="list-style-type: none"> • local driverless taxi • pedals/steering wheel may or may not be installed 	<ul style="list-style-type: none"> • same as level 4, but feature can drive everywhere in all conditions

Figure 1.6: J3016TM “Levels of Driving Automation” [13].

Level 0 (No Driving Automation): human controlled.

Level 1 (Driver Assistance) Hands on control: The vehicle includes a single advanced driver assistance system like: adaptive cruise control which allows it to be kept at a safe distance of another car with auto brake and accelerate.

Level 2 (Partial Automation) referred to as "**hands-off**": This means (two or more) **advanced driver assistance systems [ADAS]**, The vehicle can control the steering with the ability to accelerate/decelerate.

Level 3 (Conditional Automation) referred to as "**eyes-off**": These vehicles have “environmental detection” capabilities and can make informed decisions for themselves, still with driver attention, an example here is the Audi A8 with Traffic Jam Pilot [14].

Level 4 (High Automation) referred to as "**mind-off**": These vehicles can intervene if there is a system malfunction. The driver isn't required to intervene at all in most circumstances still with the option to manually override. Such as the Google Waymo Level 4 self-driving taxi [15] service in USA Arizona, Phoenix, San Francisco...

Level 5 (Full Automation) These vehicles do not require human attention, they are fully autonomous. Level 5 cars won't even have steering wheels or pedals, such as Audi's Aicon [16] future concept car [17].

1.3.5 Current State

According to (SAE) in 2023 this is where we are in the field of self-driving cars (Figure 1.7).

How close are we to a true self-driving car?

According to the Society of American Engineers (SAE) classification system, a vehicle is capable of autonomous driving at SAE level 3 and higher.

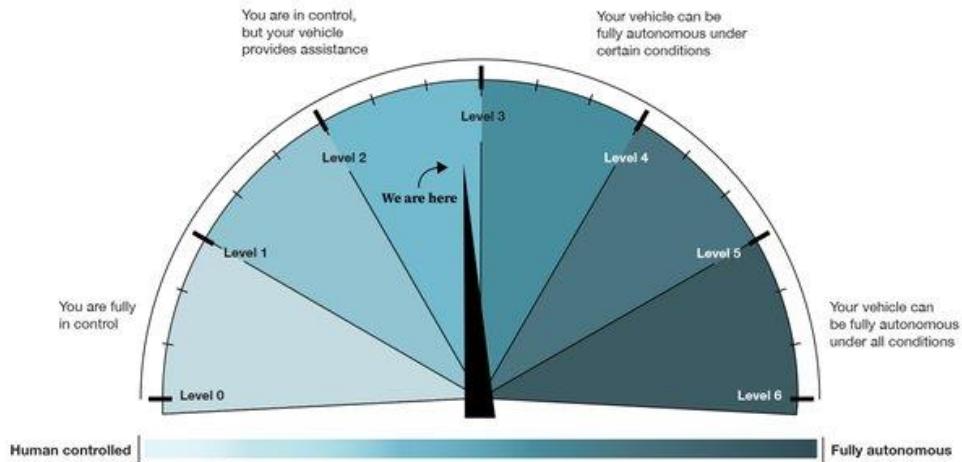


Figure 1.7: State of Avs in 2023 [18].

1.3.6 Anticipated Impacts of Autonomous Vehicles

As the Automated Vehicles continue to evolve and become more prevalent, it is crucial to examine the potential impacts they may have on society, some of the largest potential benefits are:

1.3.6.1 Safety

Autonomous vehicles have the potential to significantly decrease crashes. (Table 1.3) highlights the magnitude of automobile crashes in ALGERIA and indicates the sources of driver error, From 2014 to 2017 Over 93.29 % of these fatal crashes are due to a lack of respect from the Drivers of traffic law and careless driving and excessive speed. Even in worldwide Over 90 percent of all crashes are attributed to driver error as the primary cause [19].

Total Crashes in 2022	17.186
human cause as primary factor	The human element and lack of respect for traffic law in general
Total Fatal	709
Total Injurious	20.575
Total Financial Losses	Estimated at 100 billion dinars annually

Table 1.3: Automobile crashes in ALGERIA in 2022 [20].

Autonomous vehicles can have many advanced safety features systems [ADAS], that work together to help and assist the drivers to stay safe, along a comfortable and stress-free driving experience.

According to USA Department of Transportation study [21], it's estimated that collectively these ADAS technology could impact 3.59 million total crashes per year. Forward collision system could prevent 1.7 million crashes, while lane keeping assist impacts another 1.12 million crashes, the ADAS systems can prevent 20,841 deaths/year, or about 62% of total traffic deaths in USA, while preventing damage to 4.60 million vehicles.

1.3.6.2 Traffic congestion and Fuel consumption

In addition to enhancing automotive safety, researchers are actively working on leveraging AV technology to address challenges related to congestion and fuel consumption. AVs can sense vehicles braking and acceleration decisions. This allows for smoother braking and speed adjustments leading to fuel savings, and less brake wear [22].

1.3.6.3 Travel Behavior Impacts

AVs have the capacity to create substantial changes in travel behavior. For example, AVs can provide mobility for those too young to drive, the elderly and the disabled, as a result, generating new roadway capacity demands [22].

1.3.7 Autonomous vehicles Technologies

In order to achieve vehicle autonomy, a cyclic process depicted in (figure 1.8) must be executed. First, the sensors capture the environment, then in perception the vehicle will find its position with respect to its obstacles so that motion planning can take place to calculate the intended trajectory. Controller will generate an output that goes to the actuators so the vehicle can autonomously navigate [23].

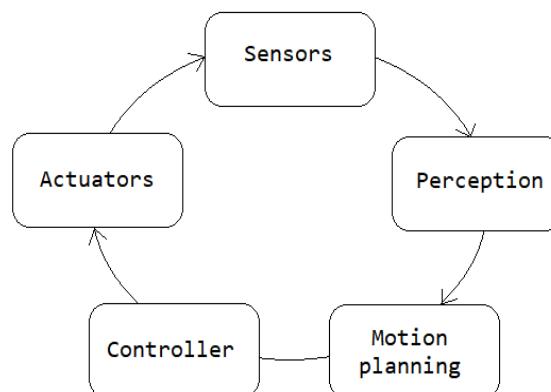


Figure 1.8: Autonomous Vehicles cyclic process [24].

1.3.7.1 Sensing and Perception

AVs employ a combination of technologies and sensors to perceive the road, surrounding vehicles, and objects located on and alongside the road. This data from sensors is then processed using (AI) algorithms that identify the trajectory of the vehicle with objects recognition, such as pedestrians, other vehicles, traffic lights, and road signs [25].

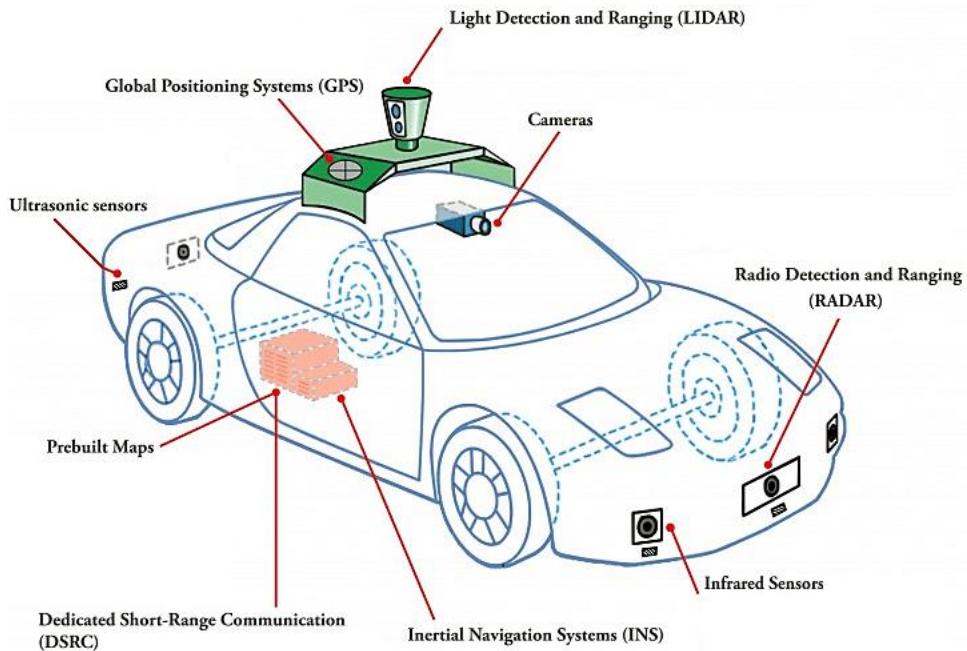


Figure 1.9: Autonomous Vehicles technologies [26].

The next section sets out, in particular, how the application of multiple sensors can be used to overcome various weaknesses, with an overview of each sensor and its respective functions.

Camera sensors: In computer vision application it's the primary sensor used, when multiple cameras are used together, they can offer a complete 360° perspective this make it possible to identify objects and individuals.

Ultrasonic sensors: It uses High-frequency sound waves to measure the distances in close-range applications, like aiding in parking maneuvers.

Infrared sensors: They capture images under lowlighting situations, like night vision surveillance systems.

LiDAR sensors: They utilize laser technology to calculate distances and generate detailed 3D images of the environment.

Radar sensors: By employing short range (24 GHz) mixed with long range (77 GHz) radio waves, it can determine the distance, angle, and speed of objects.

Global positioning system (GPS) receiver: It's a sensor that utilizes satellite triangulation to determine the precise location of the system within its immediate surroundings.

There is also a Gyro, Accelerometer sensors, and Wireless Vehicle Communications [27].

1.3.7.2 Decision Making

Self-driving cars use artificial intelligence to make real-time decisions based on the data they gather from their sensors. Deep neural networks serve as a key element in the technology of autonomous driving. Neural networks analyze the camera feeds for roads, signs, cars, obstacles, and people, a further study about AI and computer vision is in the next chapter.

1.4 Conclusion

In this chapter, we explored the field of autonomous systems with a specific focus on autonomous vehicles and intelligent automation, which combines artificial intelligence and automation. Autonomous vehicles have the potential to revolutionize transportation by enhancing safety, efficiency, and accessibility. They offer benefits such as reduced accidents and improved mobility with an improved quality of life.

In the next chapter we delve into computer vision and deep learning, key components necessary for developing intelligent autonomous systems such as Autonomous vehicles.

CHAPTER 2

COMPUTER VISION AND DEEP LEARNING

2.1 Artificial Intelligence

2.1.1 Definition of Artificial Intelligence

According to **Oxford Dictionary** AI is “The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.”. In other words, AI is using a computer for doing a task that normally would require human intelligence [28].

2.1.2 Historical Development of Artificial Intelligence

The field of artificial intelligence (AI) has a long history that extends over several decades. Here's an overview of the historical development of AI:

- Origins (1940s-1950s):

In 1950, Turing published the famous "Turing Test," a theoretical test to determine a machine's ability to exhibit intelligent behavior indistinguishable from that of a human.

- The Dartmouth Conference (1956):

In 1956, a group of scientists gathered by John McCarthy for “the Dartmouth Conference” about a proposal on Artificial Intelligence, which was the birth of Artificial Intelligence.

- Deep Learning Revolution (2010s-Present):

Deep learning made a significant improvement in 2010s. Deep neural networks with multiple layers showed exceptional performance in image and speech recognition tasks.

In 2012, The AlexNet system achieved remarkable success in the ImageNet Large Scale Visual Recognition Challenge. “AlexNet” achieved a level of accuracy that was twice as good as its competitors, effectively reducing the error rate in the image-recognition contest by half.

In 2020, the releases of (COVID-19) lead to “LinearFold” AI algorithm, the algorithm can predict the virus in only 27s, which is 120 times faster than other methods at that time [29].

2.1.3 Types of Artificial Intelligence

When it comes to the classification of artificial intelligence types, we can classify AI into two main types based on their functionality and Capabilities.

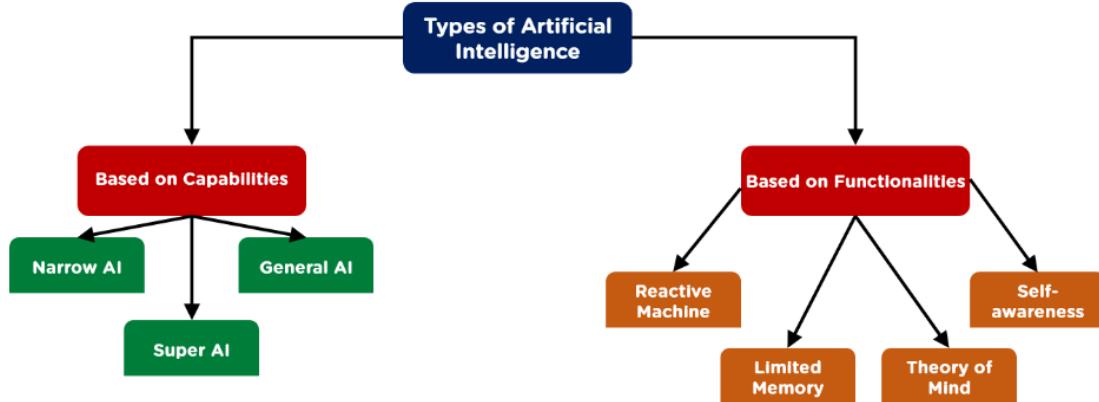


Figure 2.1: Types of Artificial Intelligence [30].

Type 1: Based on Capabilities

Artificial Narrow Intelligence (ANI) (weak AI): This type represents all the existing AI in this era, it's an AI system that can only perform a specific task autonomously using human-like capabilities.

Artificial General Intelligence (AGI) (Strong AI): This type has the ability to learn, perceive, understand, and function completely like a human being. This makes (AGI) systems just as capable as humans, this type is non-existing right now.

Artificial Superintelligence (ASI): If Artificial Superintelligence ever be archived it will be the pinnacle of AI research. (ASI) will be exceedingly human beings at everything they do these machines may also threaten our existence or the way of life [31].

Type 2: based on Functionalities

Reactive Machines: Their capabilities are confined to the specific task they were programmed for, they lack memory or any understanding of their surroundings, making them suitable only for reactive functions.

Limited Memory: They possessed the same characteristics as purely reactive machines, along with the ability to learn from short historical data in order to make decisions. This technology finds application in areas such as **self-driving vehicles**.

Theory of mind: Only exists as a concept. It can comprehend people's emotions, and thoughts. One example of the theory of mind AI is Kismet that can mimic human emotions and recognize them [32], and Sophia that can sustain eye contact, recognize individuals, and follow faces [33].



Figure 2.2: Kismet and Sophia [34] [35].

Self-awareness: Self-awareness AI only exists hypothetically, these advanced systems possess an understanding of their internal traits, states, and conditions, these machines will surpass the intellectual capabilities of the human mind. If this type of AI ever was achieved, it will be one of the greatest discoveries ever made in the humankind [31].

2.1.4 Applications of Artificial Intelligence

Artificial intelligence (AI) applications have rapidly transformed industries, automating tasks, analyzing data, and driving innovation. With its ability to analyze data, recognize patterns, and make intelligent decisions, AI is revolutionizing our lives From self-driving cars to Alexa/google search. Here are just some of its applications:

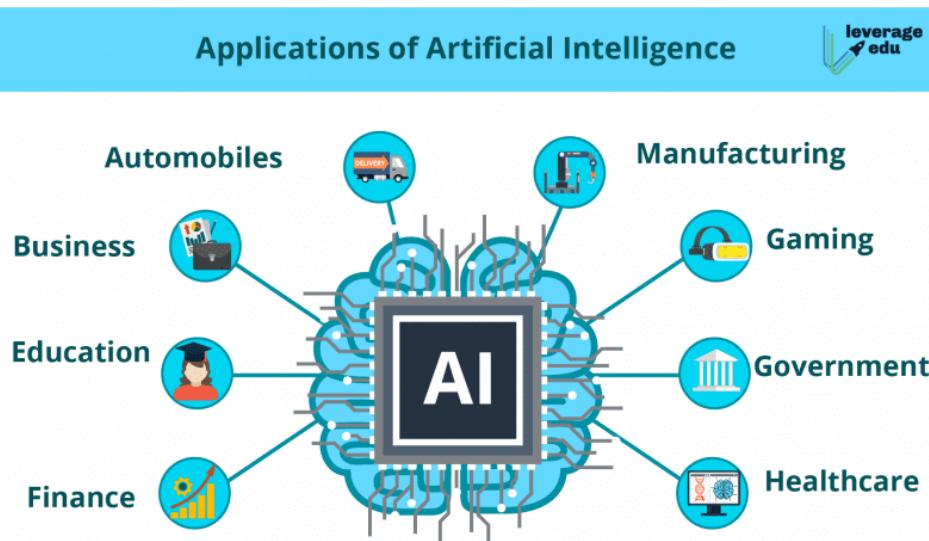


Figure 2.3: Applications of Artificial Intelligence [36].

- Healthcare and Medical Imaging Analysis
- Education (grading assignments, online study materials, e-conferencing)
- Searching (Google machine learning algorithms, ChatGPT AI chatbot)
- Precision agriculture (Autonomous Farm Tractors, Analyze weather patterns)
- Social media (Track user behavior).
- Industry (predictive maintenance, optimized production processes, analyze sensors data, and reduce waste)
- Transportation Automation (Self-driving cars and trucks, metros, and London Automatic Train Control) [37].

2.1.5 ways AI can be archived

There are many ways AI can be achieved some of them are as follows (figure 2.4).

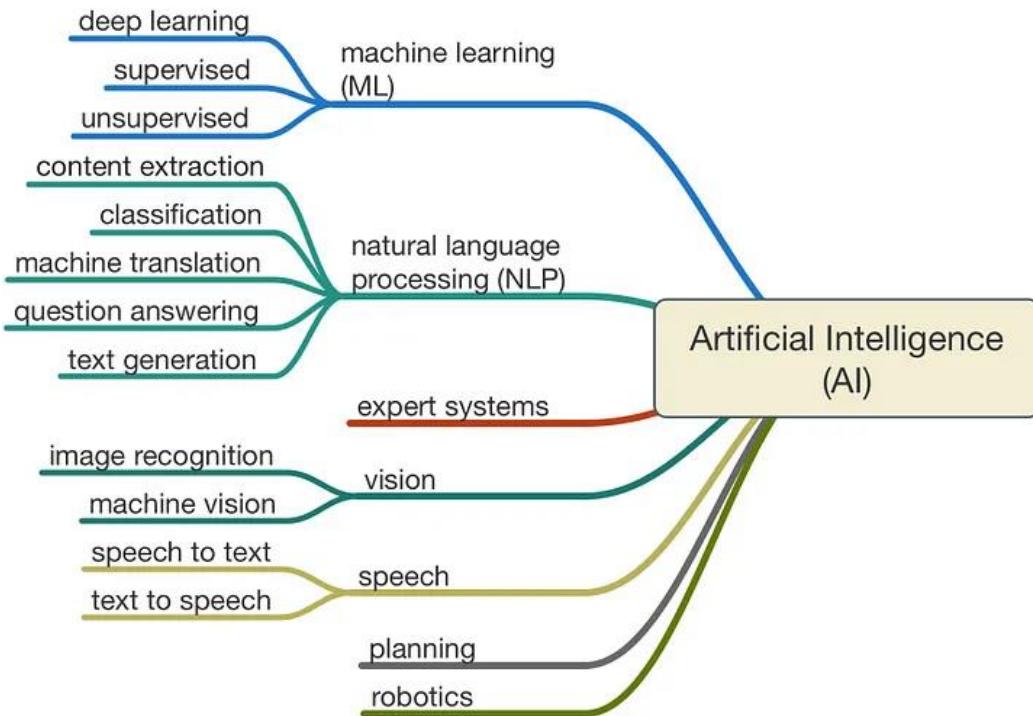


Figure 2.4: ways AI can be archived [38].

Main Branches are:

- Machine Learning (ML)
- Deep learning
- Computer Vision
- Natural Language Processing (NLP)
- Transfer Learning

2.1.6 Advantages & Disadvantages of Artificial Intelligence

Advantages

- Emotionless decisions
- Needs no sleep
- Easier spreading of knowledge

Disadvantages

- Lack of creativity
- Lack of common sense in reasoning
- Ethical Challenges (substituting humans in every field causing a high rate of unemployment, Lack of Originality and Authenticity)
- Cost [39].

2.2 Computer Vision

2.2.1 Computer Vision Definition

Computer vision is an extent of artificial intelligence (AI) that allows computers to extract meaningful information from digital images, videos and other visual inputs. These insights enable them to make informed decisions and take appropriate actions. In other words, computer vision allows computers to see, observe and understand like human being [40].

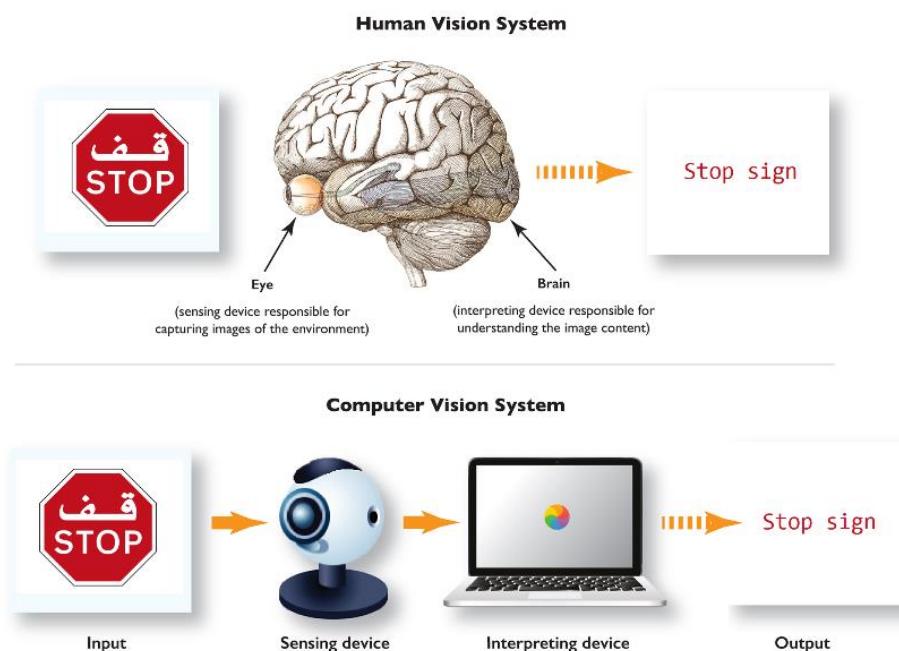


Figure 2.5: Human vision system vs computer vision system [41].

2.2.2 Computer Vision tasks

The following are some common tasks used by computer vision systems:

- Object detection: Object and its placement in the image.
- Object segmentation: The pixel items that make up the object.
- Object classification: the wide class that the item lies in.
- Object localization: Object position.

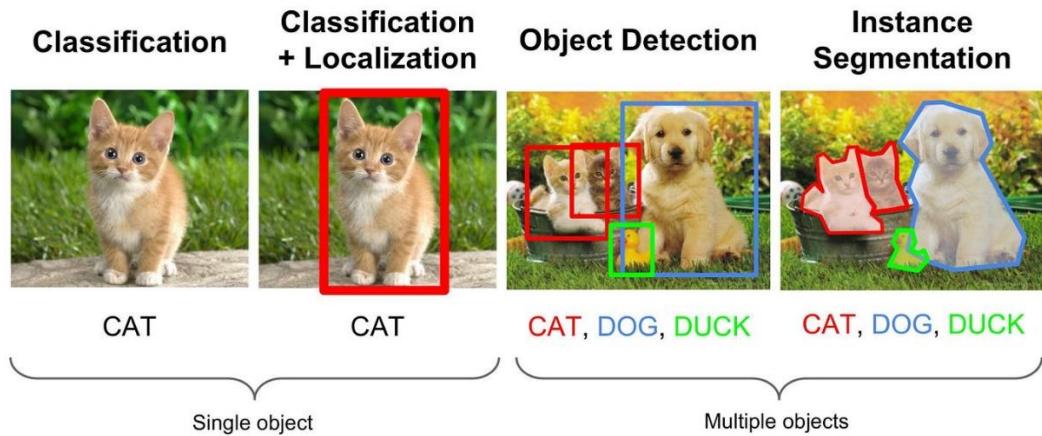


Figure 2.6: Computer vision common tasks [42].

Applications of computer vision

By enabling machines to understand and interpret visual information like humans do Computer vision technology has the potential to enhance efficiency, improve safety, and unlock new possibilities across various domains. Such as:

- Autonomous vehicles
- Medical imaging (X-rays, MRI, Cancer Detection)
- Facial recognition
- Agriculture

2.2.3 Digital Image Definition

A digital image is a visual representation of an object, or a scene that is stored and displayed in a digital format, composed of a grid of **pixels**. where each pixel represents a specific color. that can be captured using digital cameras, created through computer graphics, or scanned from physical photographs or documents [43].

Every image has three main properties:

- Size (pixels resolution)
- Color space (RGB, HSV)
- Channel

There are various types of images:

Binary images: Have two-pixel elements, 0 and 1 (black or white).

Grayscale image (1 channel): The pixel is an 8-bit integer value between 0–255 (black to white) (figure 2.7).

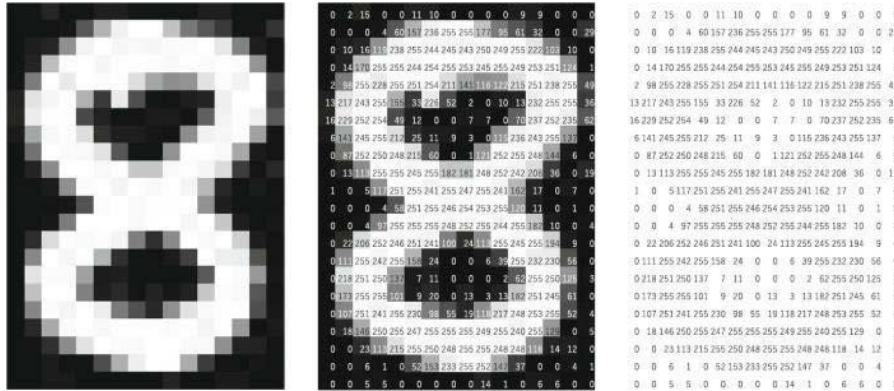


Figure 2.7: Pixel Map for a handwritten digit [44].

RGB image (3 channels): It contains three layers of 2D image (red, green, blue) channels, they are represented in a 3-dimensional array each layer represents a color (figure 2.8).

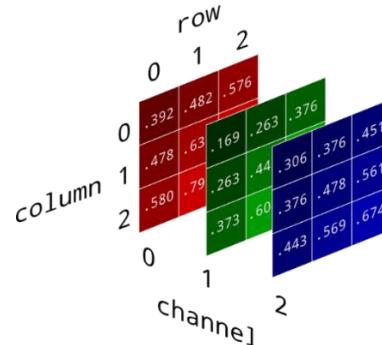


Figure 2.8: RGB image channels [45].

A Tensor is a multidimensional array, where Rank 0 tensor is a scalar (0D tensor), rank 1 is a vector (1D tensor), rank 2 is a matrix (2D tensor), rank 3 is a (3D tensor). A batch of 3 RGB images can be represented using a four-dimensional (4D) tensor (figure 2.9).

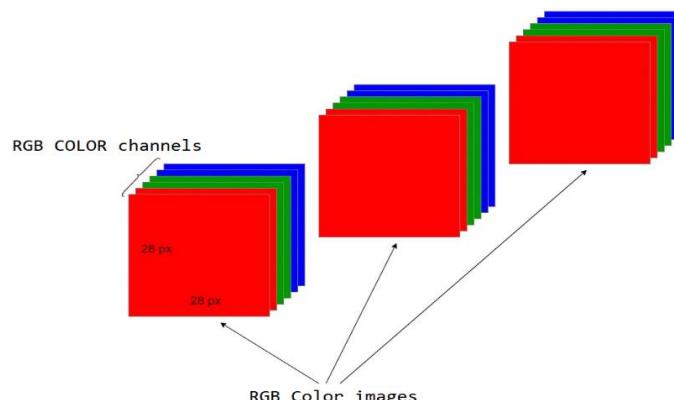


Figure 2.9: 3 RGB (4D) tensor representation [46].

These 3 RGB images can be represented in two different ways Channels-last or Channels-first (4D) tensor, In case of a batch of 3 RGB images:
 Channels-last (samples, height, width, color channel), **(3, 28, 28, 3)** [47].

2.2.4 Computer Vision Libraries

2.2.4.1 OpenCV — Evolution in Computer Vision

OpenCV (Open-Source Computer Vision Library) is a computer vision and machine learning library, that was built to provide a common infrastructure for computer vision applications, it has more than 2500 optimized algorithms, and more than 47000 users community and an estimated 18 million downloads. It's used by individuals, companies, research groups and by governmental companies. It has C++, Python, Java, and MATLAB interfaces, and supports Windows, Linux, Android and Mac OS.

It can be used to:

- Converting images from one color space to another
- Applying custom filters to images
- Performing morphological operations on images (Erosion, Dilation...).
- Edge Detection and Face Detection

There is other Image processing that also used in computer vision tasks like: Scikit-image, Pillow, NumPy...etc. [48].

2.3 Machine Learning and Deep Learning Components

Machine Learning is a branch of Artificial Intelligence, which includes Deep Learning.

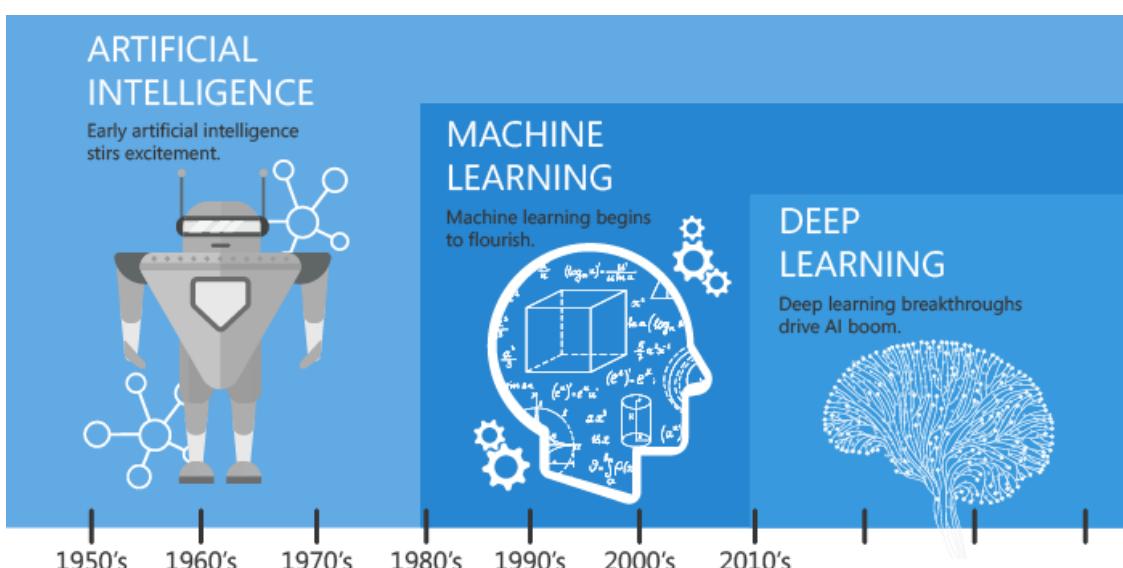


Figure 2.10: AI, ML and DL [49].

2.3.1 Machine Learning

Machine learning is a field of artificial intelligence (AI) that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Over the last decades, the technological advances in **storage** and **computing power** have enabled some innovative products based on machine learning, such as self-driving cars. Machine learning is an important component of the field of data science. Through the use of statistical methods, algorithms are trained to make classifications, predictions, and decisions. Machine learning algorithms are typically created using frameworks that accelerate solution development, such as PyTorch and TensorFlow [50].

2.3.1.1 Fields of Machine Learning

Nowadays, machine learning is gaining significant traction across numerous industries, such as: medical field, finance, transportation, e-commerce, and many others.

Field	Application
Computer Vision	Self-Driving Cars, Object Recognition, Robotics, Face Recognition, etc.
Signal	Language NLP, Speech Processing, etc.
Medical	Drug Development, Biomedical Data Analysis, Neurosciences, etc.
Finance & E-commerce	Analytics, Insurance, Fraud detection, Financial Forecasting, Spam Email, etc.
Geography	Weather – Earthquake prediction, etc..

Table 2.1: Usage of Machine Learning [51].

2.3.1.2 Types of Machine Learning

Machine learning can be divided according to the nature of the data labeling into four types:

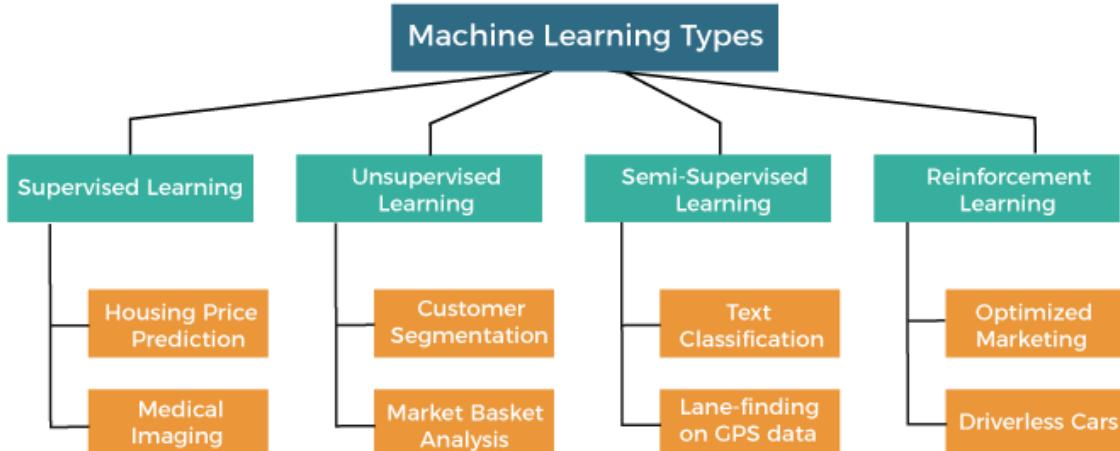


Figure 2.11: The four different Machine Learning algorithms [52].

a) Supervised Learning

The machine is trained using "labelled" dataset, and based on the training, the machine predicts the output. Labeled data refers to input data that is paired with corresponding target labels. The primary objective of supervised learning is for the model to grasp the mapping between input features and their respective labels. This understanding enables the model to make precise predictions when confronted with new and unseen data.

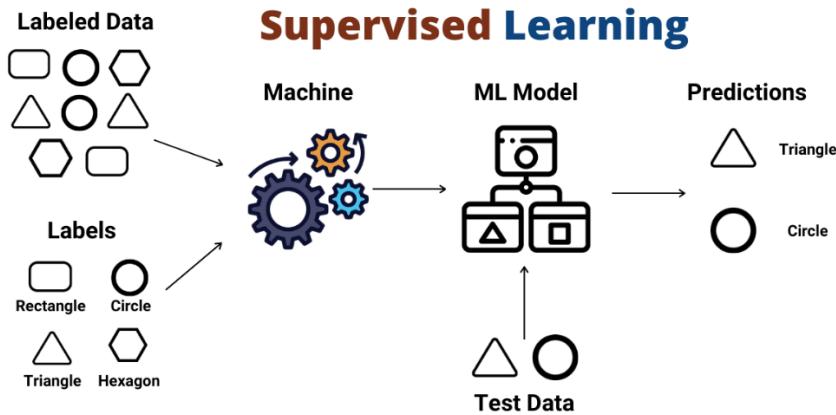


Figure 2.12: Overview of supervised learning [53].

List of common algorithms:

- Linear Regression Algorithm
 - Multivariate Regression
 - Decision Tree
 - Lasso Regression
- Classification Algorithms
 - Random Forest
 - Decision Tree
 - Logistic Regression
 - Support Vector Machine

b) Unsupervised Learning

In the Unsupervised learning technique, there is no need for supervision. The machine is trained using the unlabeled dataset, It can predict the output without any supervision. By using machine learning algorithms to analyze and cluster the unlabeled datasets [52].

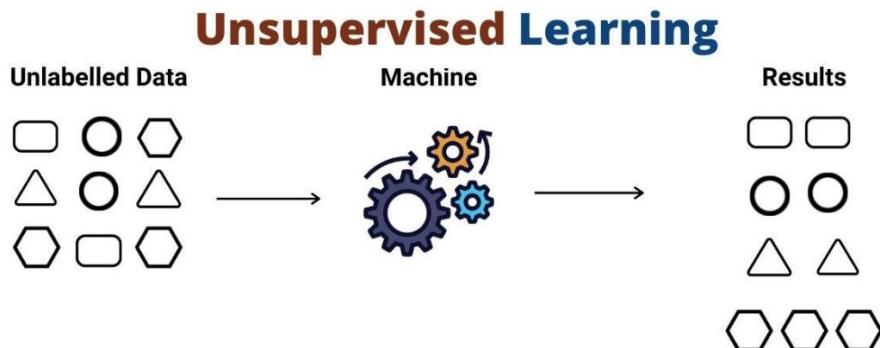


Figure 2.13: Overview of Unsupervised learning [53].

List of common algorithms:

- Clustering
 - Hierarchical Cluster Analysis (HCA).
 - Expectation Maximization.
 - K-Means.
- Visualization and dimensionality reduction
 - Kernel PCA.
 - Locally Linear Embedding (LLE).
 - T-distributed Stochastic Neighbor Embedding (t-SNE).

c) Semi-Supervised Learning

It combines both supervised and unsupervised learning. It's useful in those areas where the unlabeled data is present and getting the labeled data is a slow process [52].

d) Reinforcement Learning

It's a type of learning that makes decisions based on which actions to take. Unlike supervised learning, reinforcement learning does not rely on labeled data. Instead, the machine learns through trial and error by interacting with the environment and receiving feedback in the form of rewards or penalties [54].

In the (figure 2.14) below is the full map of Machine Learning algorithms:

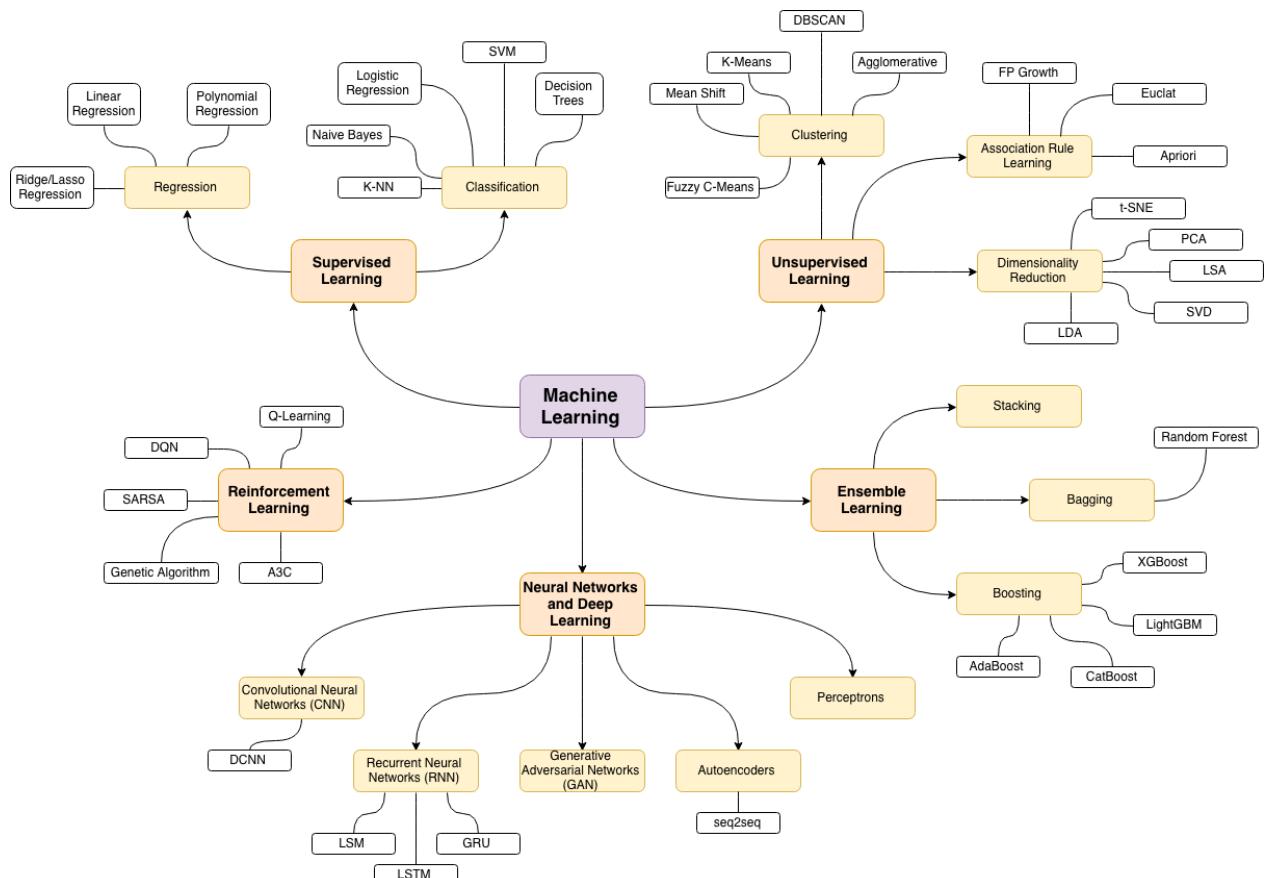


Figure 2.14: Machine Learning Full Map [55].

2.3.1.3 Major Challenges Faced by Machine Learning

- Poor Quality of Training Data
- Insufficient Quantity of Training Data
- Lack of Training Data
- Underfitting of Training Data
- Overfitting of Training Data [56].

2.3.2 Deep Learning

Deep learning is a subset of machine learning. It involves training a neural network to recognize patterns and extract meaningful information from large amounts of data, that are combined to form the deep neural networks. These neural networks try to mimic the behavior of the human brain allowing it to “learn” from large amounts of datasets. More layers can help to optimize and refine the accuracy of the model.

There are many different types of neural networks:

- Perception and multilayer perceptron's, they are the oldest and the simplest.
- Convolutional neural networks (CNNs), used primarily in computer vision and image classification applications.
- Recurrent neural networks (RNNs), used primarily in natural language and speech recognition applications [57].

2.4 Basics of Artificial Neural Networks (ANN's)

2.4.1 Artificial Neural Network Definition

Artificial neural network (ANN) is derived from the biological concept of neurons. A biological neuron has four parts dendrites, nucleus, soma, and axon.

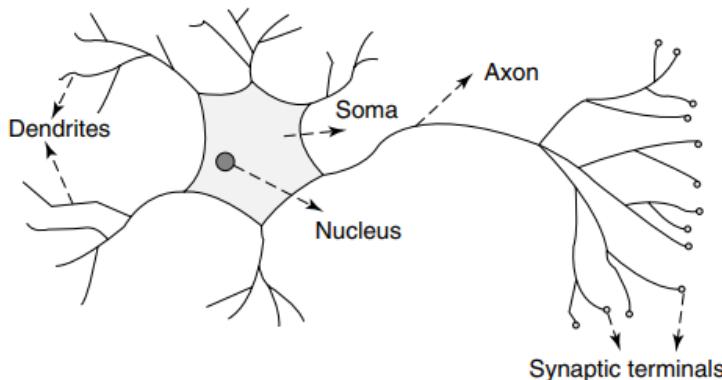


Figure 2.15: Biological neuron [58].

The dendrites receive electrical signals. Soma processes the electrical signal. The output of the process is carried by the axon to the dendrite terminals where the nucleus is the heart of the neuron.

An artificial neural network behaves the same way. It works in three layers, the input layer takes input from the other elements or other artificial neurons (like dendrites). The hidden layer processes the input with weight and bias (like soma and axon). Finally, the result is then transformed by a transfer function into the output (like dendrite terminals) transfer function may be anything like Sigmoid, hyperbolic tangent functions or a step [54].

2.4.2 Artificial Neural Network Architecture

The ANNs have two kinds of modes operation: the training mode and the applying mode. The artificial neural network is often divided in three main parts constitute:

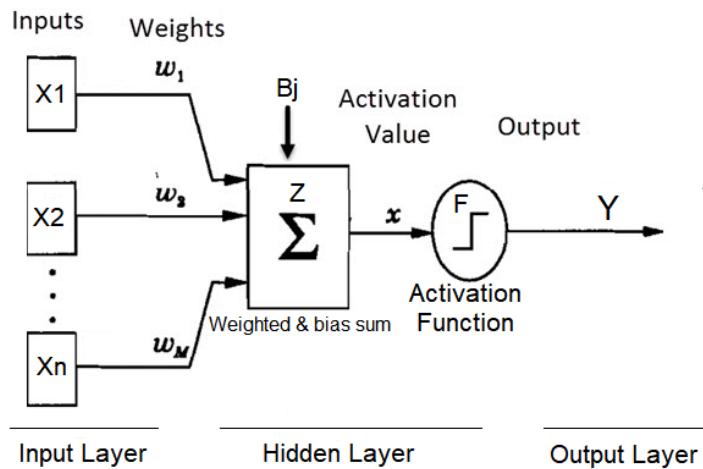


Figure 2.16: Artificial neural network Structure [59].

Input Layer: It's a set of nodes that takes the input data as one per node.

Hidden Layers: An ANN can have one or more hidden layers. Each layer can have any number of Neurons.

Output Layer: Each ANN has one Output Layer which provides the output of the model.

Weights, Bias, and activation functions: Each connection mentioned in the (figure 2.16) is assigned with weight and bias. The weight value assigned randomly at first and updated by the optimizers in each epoch (iteration) [60].

2.4.3 Activation Functions

It's a mathematical function applied to the output of each neuron in a network. It determines whether the neuron should be activated based on the weighted sum of its inputs.

Some of the forms of Activation Functions are: Binary Threshold, Linear Threshold, Sigmoid and Gaussian.

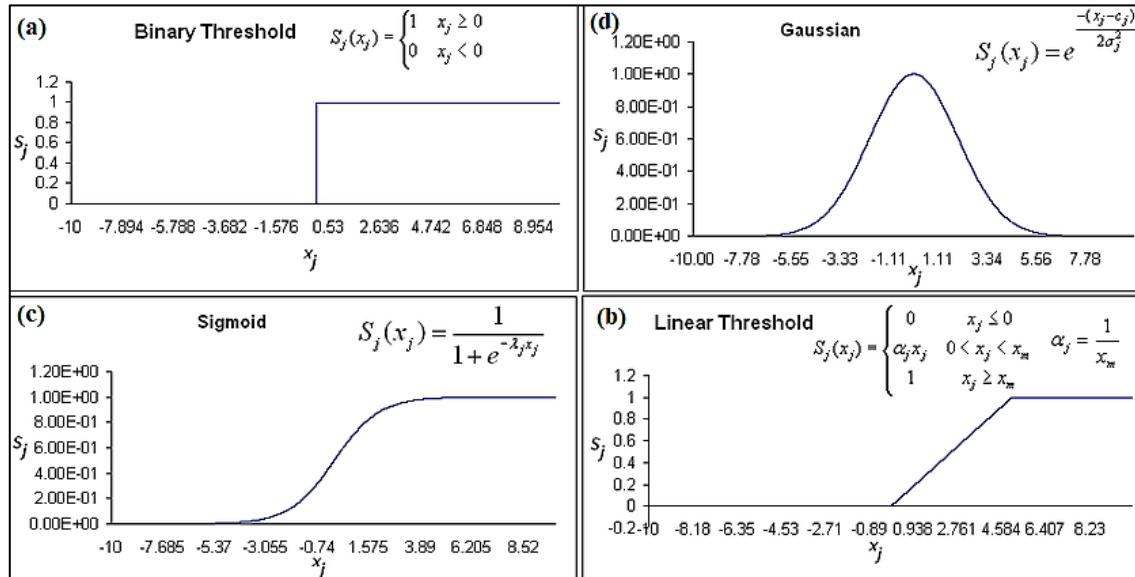


Figure 2.17: Common ANNs activation functions [61].

2.4.4 Types of Perceptron models

There are two types of Perceptron models:

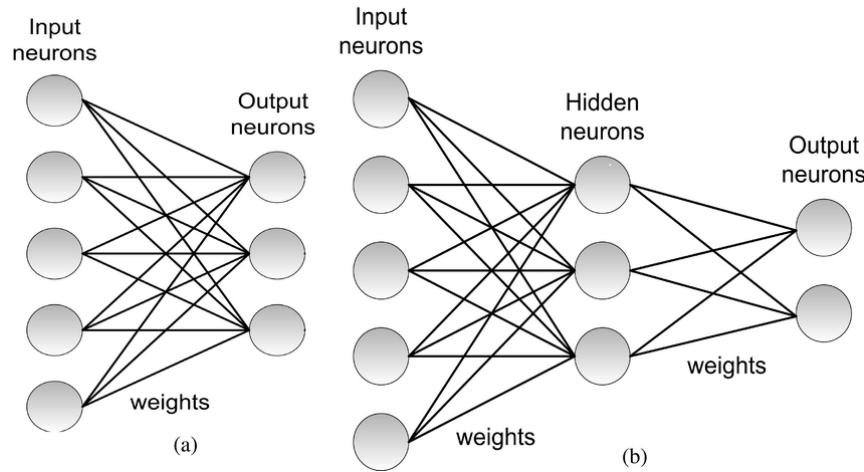


Figure 2.18: Architecture of a single and a multilayered Perceptron's [62].

- Dimensional layer (Single Layer Perceptron) **(a)**: it Contains layer on input fully connected to a single layer of output neurons.
- Multi-Layer Perceptron (MLP) **(b)**: It has 3 or more layers (input, hidden and output layer).

2.4.5 Artificial Neural Network Structure

There are basically two types of structures, feedforward neural network (non-recurrent) and feedback neural network (Recurrent). In Feed forward Network, the signal travels in one way

only but in Feedback Network, the signal travels in both the directions by introducing loops in the network. (figure2.19) show Feed forward and feedback most famous structures [63].

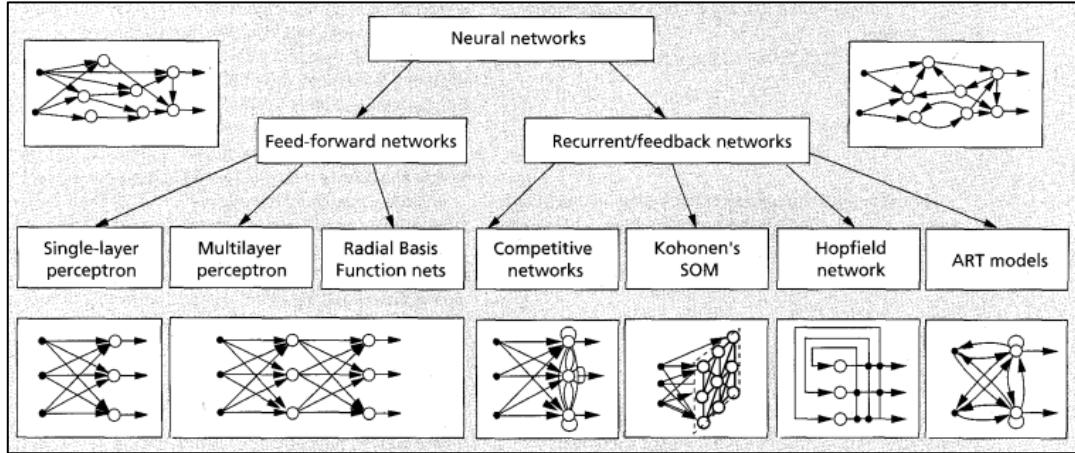


Figure 2.19: Classification of feed-forward and recurrent feedback NNs architectures [63].

2.4.6 Mathematical Model of ANN

We take the case of a Single Perceptron as it's the simplest, the output can be expressed as $f_j(x)$ of the input:

$$\mathbf{x} = (x_1, \dots, x_d)$$

weighted by a vector of connection weights:

$$\mathbf{w}_j = (w_{j1}, \dots, w_{jd})$$

We have: $y_i = f_i(\sum_{i=1}^n W_{ij} \cdot X_i + b_{ij})$

x_j : inputs.

b_{ij} : bias.

$f_j(x)$: activation function.

w_{ij} : weights.

y_j : output.

n : Number of inputs.

2.5 Convolutional Neural Network

2.5.1 Convolutional Neural Network Definition

CNN is a type of deep learning model for processing data that has a grid pattern, such as images. It's got a mathematical construct that consists of three types of layers: convolutional, pooling and fully connected. In order to extract features, the first two convolution and pooling layers are used whereas the third layer is a fully integrated layer that maps extracted features into final output (figure 2.20).

In a CNN architecture, digital images serve as the input, where the pixel values are organized in a two-dimensional (2D) grid, in the Convolution a small grid of parameters called kernel (an optimizable feature extractor “Filter”) is applied at each image position, that separates and identifies the various features of the image it's a process called **Feature Extraction**. The network of feature extraction consists of many pairs of convolutional and

pooling layers. After that a fully connected layer comes that utilizes the output from feature extraction process to predict the class of the image.

The process of optimizing parameters (kernels) is called training, which is performed so as to minimize the difference between outputs and ground truth labels (intended values) through different optimization algorithms [64].

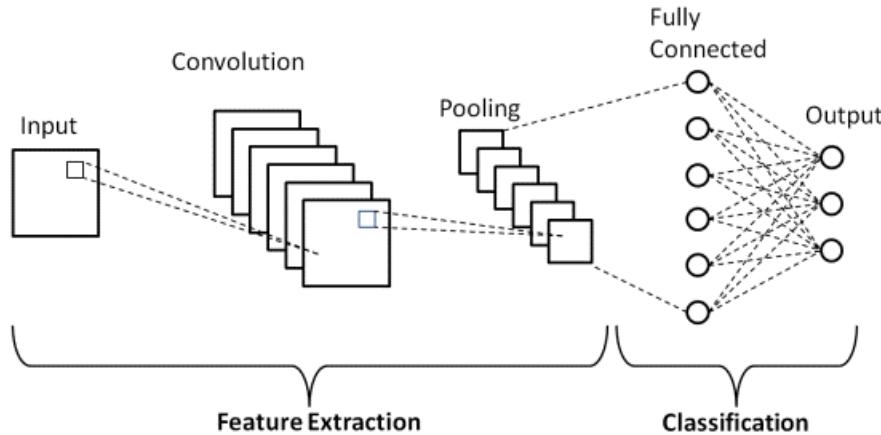


Figure 2.20: Outline of CNN [65].

2.5.2 Layers in a CNN

Input Layer: In CNN, Generally, the input will be an image or a sequence of images (batches). This layer holds the raw input of the image batches (samples, width, height, and depth).

Convolution layer: For 2-dimensional grids such as images, K is a convolution kernel (filter) applied to a 2D image (I). The principle of 2D convolution is to drag a convolution kernel on the image. At each position, we get the convolution between the kernel and the part of the image that is currently treated. The kernel moves by a number s of pixels called the **stride**. If we apply C0 kernels size of $(k \times k)$ on an image. If the size of the input image is $(w \times h \times c)$, the volume of the **output** is $w_0 \times h_0 \times c_0$, where c_0 corresponds to the number of kernels that we consider.

$$W_0 = \frac{W_i - k + 2p}{s} + 1$$

$$H_0 = \frac{H_i - k + 2p}{s} + 1$$

If the image (I) has 3 channels ($5 \times 5 \times 3$) with $3 \times 3 \times 1$ filter, the convolution with the image (I) with the kernel K1 corresponds to get a $3 \times 3 \times 1$ convolved feature [66].

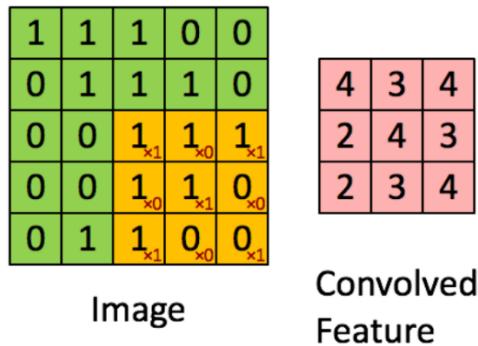


Figure 2.21: Convolution Layer [67].

The Kernel shifts 9 times because of Stride = 1, every time performing an elementwise multiplication operation (**Hadamard Product**) between K and the portion P of the image [68].

Pooling layer A pooling layer is a component commonly used in convolutional neural networks (CNNs) to down sample the feature maps generated by the convolutional layers. It helps reduce the spatial dimensions of the input, reducing the computational complexity and extracting key features.

The pooling layer operates on each feature map independently. The most common type of pooling is max pooling, where the maximum value within the pooling window is selected as the representative value. However, there are other types, such as average pooling and minimum pooling [66]. If we consider 3×3 patches, with $s = 2$, on a $(5 \times 5 \times 1)$ feature map:

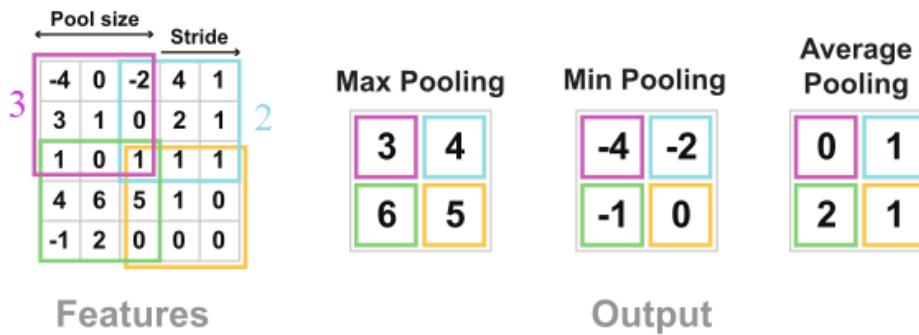


Figure 2.22: CNN Pooling layer [69].

To calculate the output dimensions:

$$\text{Output Height} = (\text{Input Height} - \text{Patch Height}) / \text{Stride} + 1 \text{ (same for width)}$$

For the given input:

$$\text{Output Height} = (5 - 3) / 2 + 1 = 2 \quad \text{Output Width} = (5 - 3) / 2 + 1 = 2$$

The output feature map will have a **(2x2)** dimensions (figure 2.22).

Fully connected layer

After several convolution and pooling layers, the CNN generally ends with a fully connected layer. The tensor that we have at the output of these layers is flattened into a vector then fed to a feed-forward neural network for the training to be applied over a series of epochs to get the correct output [66].

2.5.3 Basic Architecture of LeNet-5 CNNs

As we have seen previously different layers of CNN, to show how these layers are combined to form an architecture of CNN network. For a better understanding we take an example of The LeNet-5 network, proposed by the inventor Yann LeCun shown in (Figure 2.23). This network is a digit recognition. It's a simple architecture composed of only a few layers and filters [66].

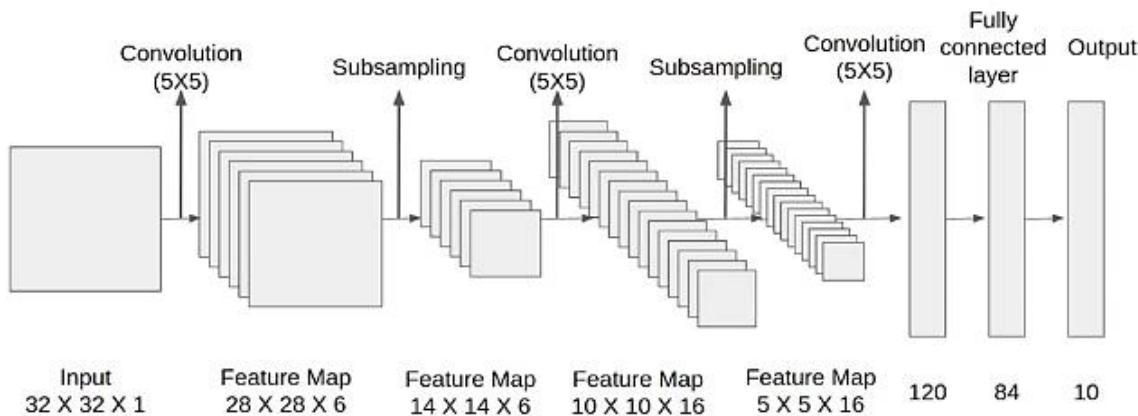


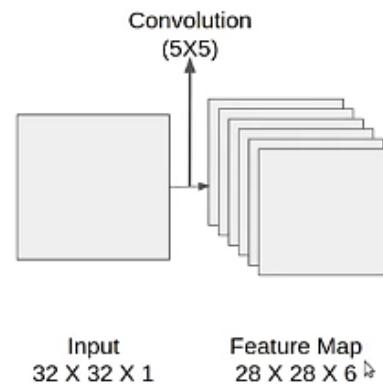
Figure 2.23: The Architecture of LeNet-5 ANNs [70].

Input layer: Input is a $(32 \times 32 \times 1)$ one channel image:

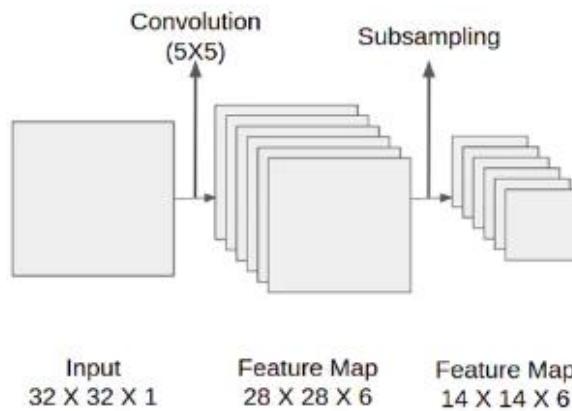


Input
 $32 \times 32 \times 1$

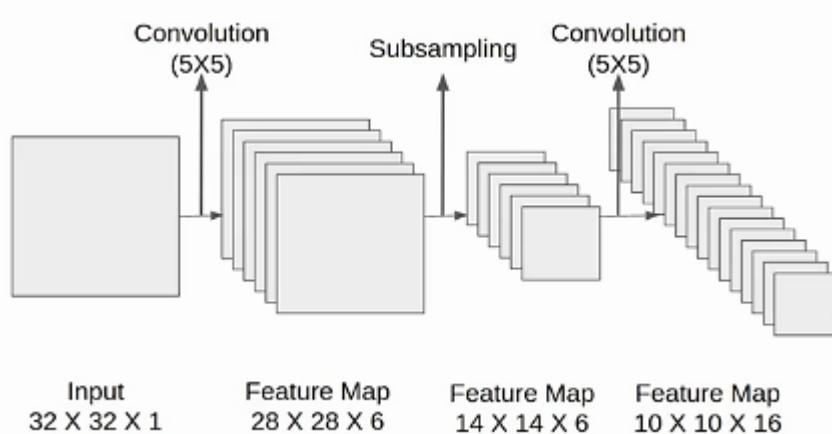
Convolution 1 (Layer 1): convoluting with a 5×5 , $s=1$, 6 filters, therefore the depth of conv1 is 6. Therefore, output shape= $(32-5)+1 = 28 \times 28 \times 6$.



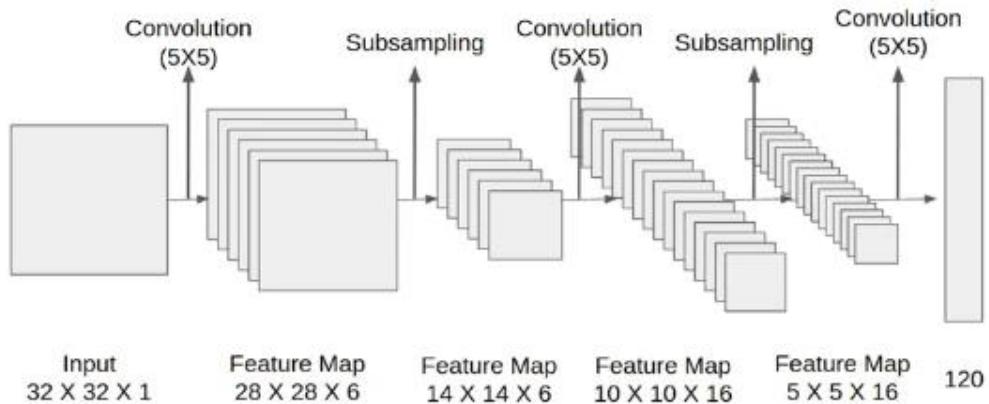
Pooling 1 (Layer 2): taking the $(28 \times 28 \times 6)$ Feature map as input and applying average pooling of a 2×2 patch and a stride of 2. With the average pool the output is: $14 \times 14 \times 6$.



Convolution 2 (Layer 3): taking the $(14 \times 14 \times 6)$ feature map and convoluting it with a filter of size $(5 \times 5 \times 16)$, with a stride=1 and with zero paddings we get an output of $(10 \times 10 \times 16)$ due to 16 filter.



Pooling 2 (Layer 4): we take the output of the previous layer and perform average pooling with a filter of (2×2) and stride=2. here imposing the filter on the $10 \times 10 \times 16$ layers for each 10×10 we obtain 5×5 outputs with a depth of 16, $(5 \times 5 \times 16)$.



Fully Connected Layer: a final convolution layer of size 5×5 with 120 filters with the $(5 \times 5 \times 16)$ feature map. As shown in the above (figure 2.24) Leaving the feature map of the size $(1 \times 1 \times 120)$. After flattening it the result will be 120 values in a vector format. The 120 values are then inputted to a feed-forward neural network with a hidden layer of 84 neurons connected and 10 neurons as the output layer [71].

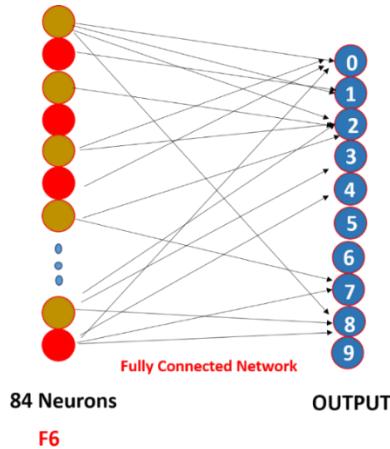


Figure 2.24: LeNet-5 Fully Connected Network [72].

2.6 Tools and Frameworks

There are many open-source libraries that offer simplified representations and tools for building deep neural networks like (resnet-18), without having to explicitly write the code from scratch. one of the most widely used libraries is pytorch.

2.6.1 PyTorch

PyTorch is an open-source deep learning framework, it was developed primarily by Facebook's artificial intelligence research group (FAIR) it's used for building and training neural networks. PyTorch provides excellent support for GPUs, and multi-dimensional arrays called tensors, which are similar to NumPy arrays. Tensors can be created, manipulated, and

operated upon using various mathematical operations. This makes it a popular choice for fast experimentation and prototyping.

Key Benefits of PyTorch:

- Large and vibrant community at PyTorch.org.
- Supports CPU, GPU, and parallel processing.
- Written in Python and integrated with popular Python libraries.
- Diverse collection of tools and libraries [73].

2.6.2 ResNet-18

Resnet-18 was introduced by Microsoft Research in 2015 for image classification. It is a variant of the original ResNet architecture, ResNet-18 consists of 18 layers (5 blocks), including convolutional layer, max pooling layer (block A), 4 ConvNets (each have two residual blocks (2xconv)), and additional Average pooling layer, and fully connected layer (figure 2.25).

Deep CNN models can take days to train on very large datasets. A way to accelerate the process is by **Transfer learning**, it refers to a process where a model trained on one problem is used in a second related problem. it has the benefit of decreasing the training time and a lower generalization error. A residual block in the ResNet network takes an input, applies a series of operations to it (typically involving convolutional layers), and then adds the original input back to the output. This addition is referred to as the "residual connection". This helps in preserving the important information throughout the network.

To train a custom dataset with ResNet-18, the last fully connected layer needs to be modified to match the number of classes in the intended dataset. This is done by replacing the fully connected layer with a new one that has the desired number of output nodes. the weights of the pre-trained ResNet-18 layers are frozen with PyTorch, so that only the weights of the new fully connected layer are updated during training.

Next step is defining a loss function and the optimizer, so the model can be trained with a custom dataset. During training, the dataset is used to update the weights of the last fully connected layer, while keeping the weights of the pre-trained layers fixed (frozen). Overall, ResNet-18 is a powerful and popular architecture for image classification that can be adapted to a wide range of custom datasets using transfer learning [74] [75].

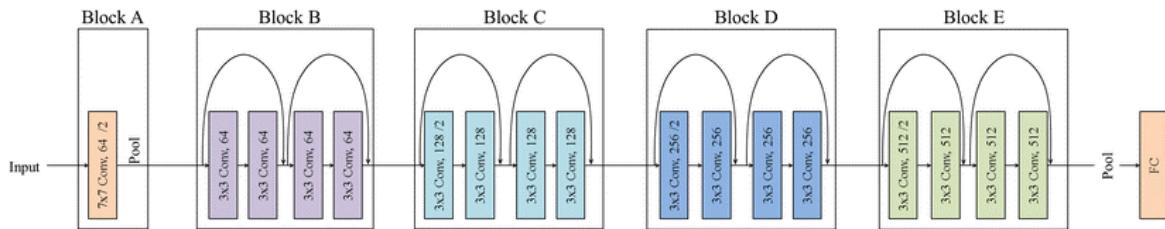


Figure 2.25: ResNet-18 five blocks Architecture [76].

2.7 Conclusion

In Chapter 2, we explored the foundations of computer vision, and deep learning, essential components in the development of an autonomous vehicle. We then delved into machine learning and deep learning. As we covered the basics of artificial neural networks (ANNs), including their architecture, activation functions, and perceptron models. We also discussed convolutional neural networks (CNNs) giving an example with LeNet-5 CNN architecture. Furthermore, we introduced key tools and frameworks such as PyTorch and ResNet-18, which we explored the concept of transfer learning, By utilizing transfer learning, we can take advantage of the learned features from a large dataset and adapt them to our specific autonomous vehicle project. Next chapter will be about the realization of the project.

CHAPTER 3

REALIZATION OF THE PROJECT

3.1 Introduction

The main goal of Chapter 3 is to provide a detailed description of how the hardware and software components were set up and configured for the project, including an explanation for data collection and preprocessing techniques used for models training, with presenting the results obtained, testing limitations, and outlining future work of the project implementation.

The project involves the development of an autonomous car using a Jetson Nano 2GB board, a CSI camera, a 4 DC motor car chassis paired with N298L motor driver, and a PCA9685 module for converting signals from I2C to PWM. The vehicle will be capable of navigating in its environment autonomously with the ability to detect and avoid collisions using computer vision and deep learning techniques.

3.2 System Architecture

This section portrays the system architecture, describing how every tool and component are associated with one another, resulting in a final autonomous vehicle (figure 3.1).

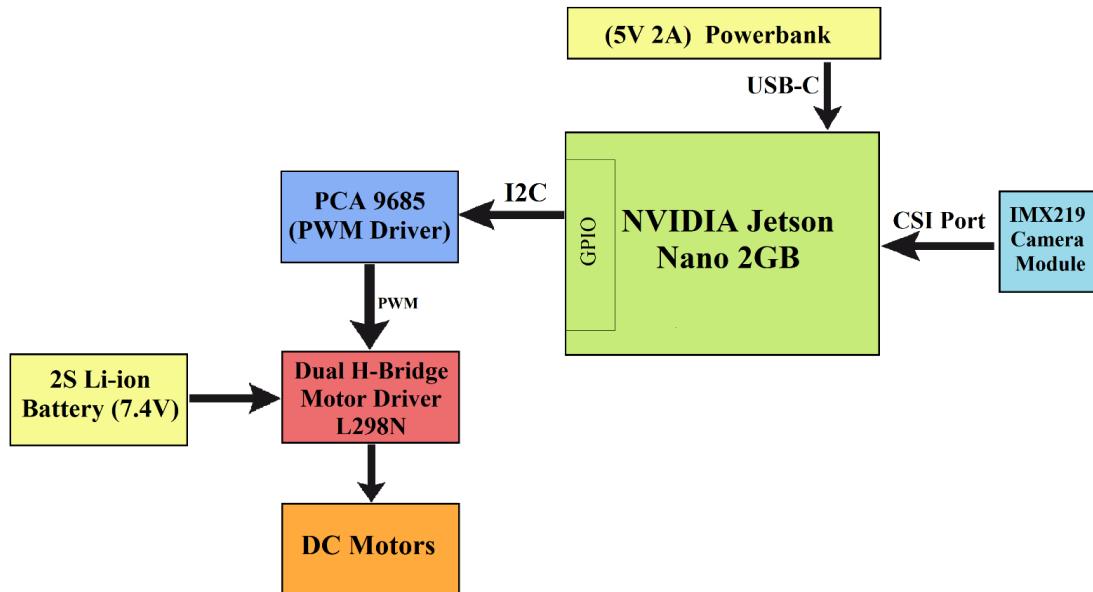


Figure 3.1: System Hardware Architecture.

3.2.1 Hardware Architecture

Processing data received from vision modules (IMX219) in real-time and sending feedback control signals needs high computing power. Due to the development of embedded devices, it is possible now to use them for tasks that require real-time calculations. An example

of such devices is SBC's (**Single Board Computers**). They have good computing power, with a very small shape, as a result, they can be embedded in devices with limited space.

There are several types of SBCs on the market, each with its own set of pros and cons depending on the application field, some of the popular examples are the Raspberry Pi, and NVIDIA® Jetson family.

	Jetson Nano 2GB October 2020	Raspberry Pi 4B 2GB June 2019
CPU	Quad-Core ARM Cortex-A57 64 bit @ 1.43 GHz.	Quad-Core ARM Cortex-A72 64-bit @ 1.5GHz
GPU	NVIDIA Maxwell 128 CUDA cores @ 921 MHz	Broadcom VideoCore VI @500 MHz
Memory (RAM)	2-GB LPDDR4	2-GB LPDDR4

Table 3.1: Jetson Nano™ and Raspberry Pi 4 technical specifications.

In the (table3.1) above we see the two SBC's specifications, both equipped with Quad-Core ARM processors that produce very similar performance paired with the same 2-GB LPDDR4 memory and number of GPIO pins.

The biggest difference between both is their Graphical Processing Unit (GPU), the **Broadcom VideoCore VI** GPU used by the Raspberry Pi 4B is a classic dual Shader architecture, with a theoretical peak performance of only **32 GFlop/s** (Floating Point Operations Per Second).

While the (**NVIDIA Maxwell 128 CUDA cores**) GPU performance of the NVIDIA® Jetson Nano™ can reach an exceptional **472 GFlop/s**, it also supports CUDA parallel computing platform that allow the use of the GPU processing power for accelerated computations, this makes the Jetson Nano ideal for AI-based computer vision, as we can run multiple neural networks in parallel and processes several high-resolution sensors simultaneously, it's a better choice over the Raspberry Pi and why the NVIDIA® Jetson Nano™ was chosen for this project.

3.2.1.1 NVIDIA Jetson Nano

We have used the **Jetson Nano™ Developer Kit** (figure 3.2)

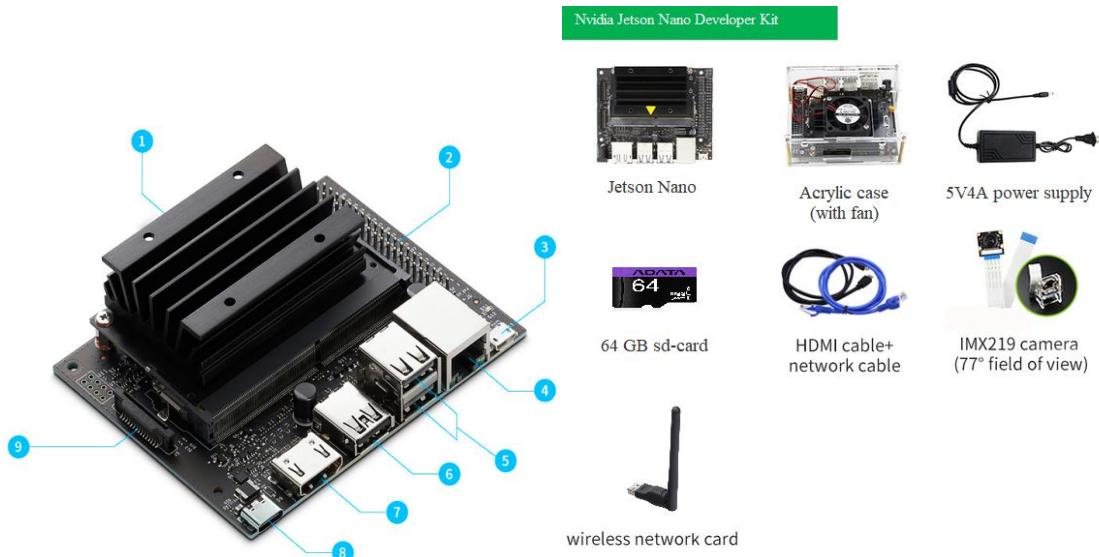


Figure 3.2: NVIDIA® Jetson Nano™ 2GB Developer Kit [77].

Technical Specifications

- | | |
|-------------------------|---------------------------|
| 1-Micro SD card slot | 2-40-pin expansion header |
| 3-USB 2.0 Micro-B | 4-Gigabit Ethernet port |
| 5-2x USB 2.0 type A | 6-USB 3.0 type A |
| 7-HDMI output port | 8-USB-C 5V 4A |
| 9-MIPI camera connector | |

CPU	Quad-core ARM A57 @ 1.43 GHz
GPU	128-core NVIDIA® Maxwell @ 921MHz
MEMORY	2 GB 64-bit LPDDR4 25.6 GB/s
STORAGE	micro-SD card (UHS-1 32GB minimum)
CAMERA	1x MIPI CSI-2 connector
EXTENSION	1x USB 3.0, 2x USB 2.0 Type A, 1x USB 2.0 Micro-B
INTERFACES	40-pin header (GPIO, I2C, I2S, SPI, UART) 12-pin
I/O	header (Power and related signals) 4-pin Fan header Gigabit Ethernet HDMI Port

Table 3.2: Jetson Nano™ Technical specifications

To interact with its environment the jetson nano have a **GPIO** (general-purpose input/output), it's the way in which the Nano can control and monitor the outside world by being connected to electronic circuits, it's a 40-pin expansion header carrying several pins (2xPWM physical pins, pins 32 and 33, 2xi2c Buses...).

Jetson Nano J41 Header										
Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO	gpio16	SPI_1_MOSI	19	20	GND
	3.3 VDC Power	1	2	5.0 VDC Power		gpio17	SPI_1_MISO	21	22	SPI_2_MISO
	I2C_2_SDA I2C Bus 1	3	4	5.0 VDC Power		gpio18	SPI_1_SCK	23	24	SPI_1_CS0
	I2C_2_SCL I2C Bus 1	5	6	GND			GND	25	26	SPI_1_CS1
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1			I2C_1_SDA I2C Bus 0	27	28	I2C_1_SCL I2C Bus 0
	GND	9	10	UART_2_RX /dev/ttyTHS1		gpio149	CAM_AF_EN	29	30	GND
gpio50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79	gpio200	GPIO_PZ0	31	32	LCD_BL_PWM
gpio14	SPI_2_SCK	13	14	GND		gpio38	GPIO_P6	33	34	GND
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232	gpio76	I2S_4_LRCK	35	36	UART_2_CTS
	3.3 VDC Power	17	18	SPI_2_CS0	gpio15	gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN
							GND	39	40	I2S_4_SDOUT

Figure 3.3: GPIO of Jetson Nano™ [78].

3.2.1.2 CSI Camera

Image recording is possible by equipping the vehicle with an IMX219-77 Camera.

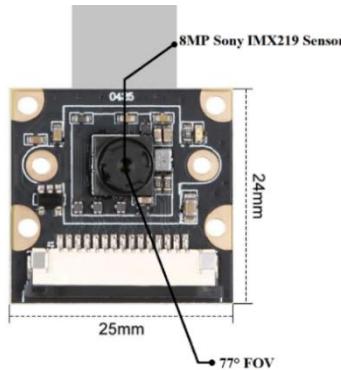


Figure 3.4: IMX219-77 Camera [79].

One of the significant advantages of using a monocular setup is its simplicity. It involves only one camera and one data stream, resulting in:

- less computing, expenses, and Lower power consumption
- Flexible placement

Resolution	8MP 3280x2464
Photosensitive chip	MX219
Aperture	F 2.0
Focal Length	2.96mm
Diagonal angle	77 degrees
CMOS size	6.35 mm
Frame Rate	30fps@8MP, 60fps@1080p, 120fps@720p

Table 3.3: IMX219 Camera parameters.

3.2.1.3 PCA9685 Driver

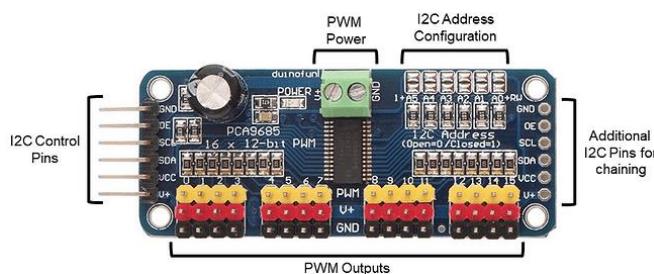


Figure 3.5: PCA9685 I2C Driver inputs [80].

An easy and simple method to control DC Motors on the NVIDIA Jetson Nano is to associate the Jetson Nano with a PWM driver through I2C communication protocol, it simplifies the motor control process and provides more precision over the speed of the motors.

The PCA9685 is an I2C-controlled PWM driver with a built-in clock, it accepts 3.3V logic pull-ups, and can be wired up to a 62 driver on a single i2c bus, for a total of 992 outputs, with a 16-channel, 12-bit resolution (2^{12} 4096 steps) Adjustable PWM frequency for up to about 1.6 KHz.

Addressing the Boards

In I2C protocol bus the slave address is a **7-bit** unique address to each slave that identifies the slave called Address Frame. The I2C base address for each board is **0x40**, to program the address offset, bridge the corresponding address jumper for each binary '1' in the address.in Board 1: Address = 0x41 Offset = binary 00001.

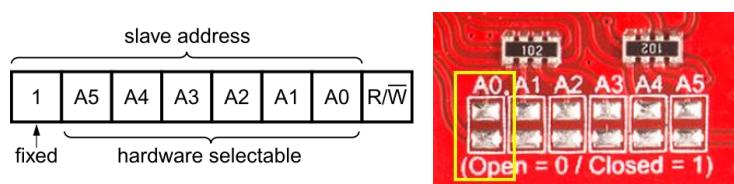


Figure 3.6: PCA9685 I2C address [81].

The PCA9685 was chosen in the project for these reasons:

- The PWM on jetson Nano GPIO has very low current (max 16mA) to drive the L298n.
- simple 2 wires communication (With only 2 pins, it output 16 signal)
- It can be controlled from a 3.3V logic pull-up.
- easy to use with Python and Adafruit MotorHat library to control DC motors.

So, The PCA9685 can generate the required PWM signals for controlling the speed of DC motors. is not designed to drive high current loads such as motors directly as its max output current is up to 25 mA, to drive the DC motors, it is typically used in conjunction with a motor driver like the L298N, which can handle higher current loads.

3.2.1.4 L298N Motor Driver

The L298N motor drive module contains a dual channel full H-bridge driver. This driver allows up two DC motors to be independently controlled in both forward and reverse directions Capable of powering 5-35V motors, in our project it was used to drive “4x 3-6V DC” motors in a 2x2 configuration in parallel means, two motors in parallel with the first L298N channel, and the other two motors in parallel to the second channel (figure 3.10).

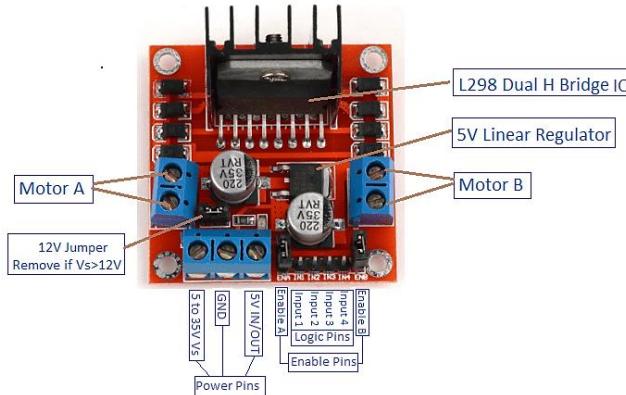


Figure 3.7: Dual H Bridge Motor DriverL298N.

General Specifications

- Dual H Bridge Motor Driver.
- Integrated 5V power regulator.
- 5V – 35V motors drive voltage.
- 2A max drive current.
- Logical Current: 0-36mA.

3.2.1.5 Power

Supplying enough power to the NVIDIA® Jetson Nano™ was important as the neural networks are process intensive. A (5v 2.1A) 5500 mAh lithium-polymer power bank was used to supply enough power to the Jetson Nano™, with a 2S 7.4V 18650 Li-ion cells Configuration that outputs sufficient current for the L298n to drive the 4 motors.



Figure 3.8: Lithium-Polymer power bank and 2S 18650 Li-ion cells.

3.2.1.6 The Vehicle

The vehicle used is 4-Wheel Robot Chassis Kit, which include:

- 4x 67mm wheels.
- 4x Geared brushed 3 – 6V DC motors (Gears for reducing motor speed and increase the torque output).
- Acrylic Chassis.



Figure 3.9: Vehicle Chassis Kits [82].

3.2.2 Hardware Assembly

In the above sections we stated the Hardware part involved in this Project, all of them were combined to complete the entire assembly. The hardware was selected according to the build configuration. We modified the camera and component placement for optimal Field of view and balanced 50/50 weight distribution for a better tires Grip, With the full wiring between component (figure 3.10), and final Autonomous Vehicle prototype fully assembled (figure3.11).

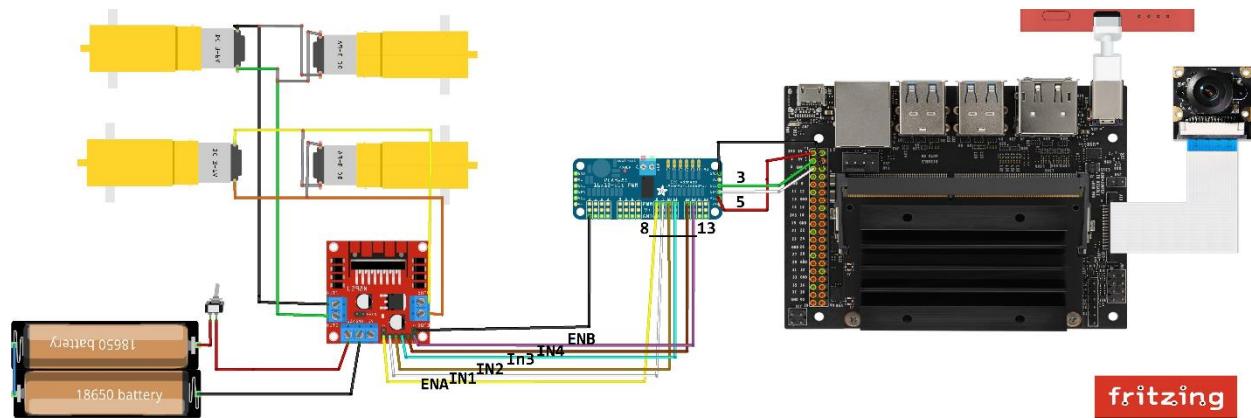


Figure 3.10: Autonomous Vehicle wiring diagram.

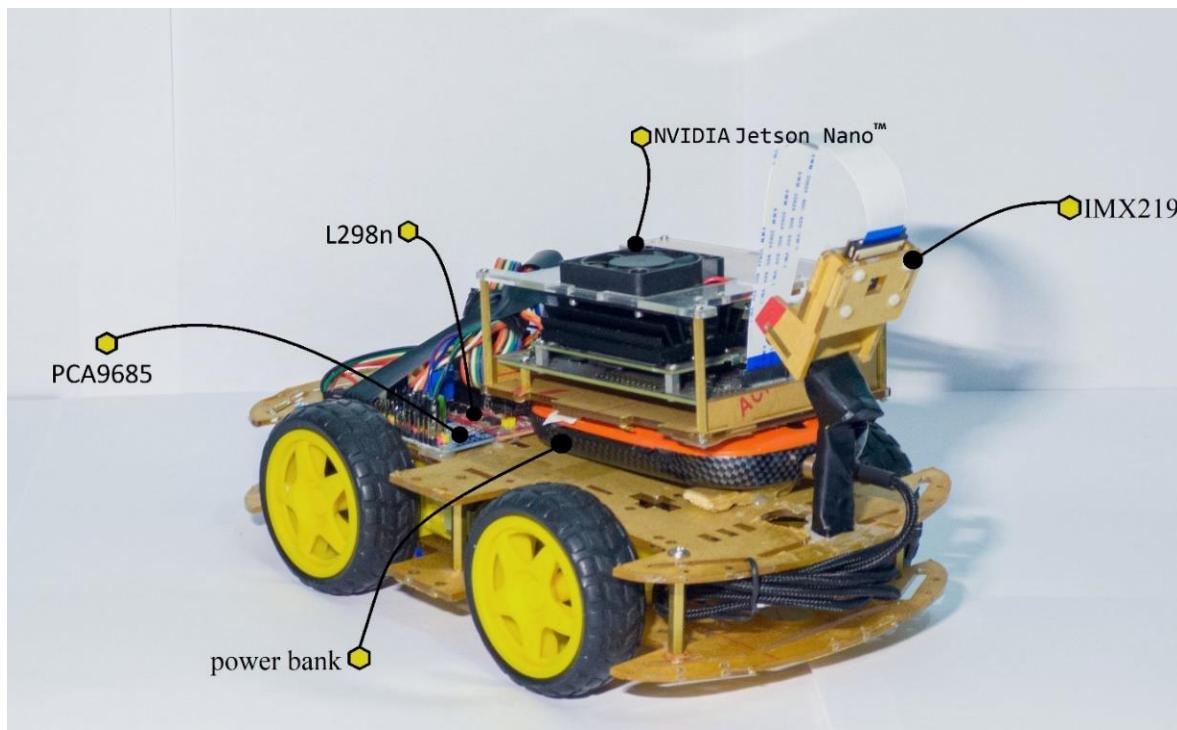


Figure 3.11: Autonomous Vehicle Prototype fully assembled.

3.3 Technologies and software installations

3.3.1 Setup the Jetson Nano

Required Components

- NVIDIA Jetson module
- 802.11ac wireless adapter
- microSD Card (UHS-1 32 GB minimum)
- USB keyboard and mouse and Display with HDMI port
- USB-C power supply (5V=4A)

Initial setup (Write Image to the microSD Card)

To start with we need to write an Operating system to the microSD Card, the jetson nano uses a microSD card as a boot device and for primary storage. It's essential to have a fast and large card for our project, we used a **64 GB UHS-I Class10 one**. In this step we need a computer with internet connection and an SD-card reader:

Step1: Download the JetPack 4.5 image from NVIDIA'S website [83].

Then follow these steps to correctly write the image to the microSD card using our windows computer:

- Format the microSD card using SD Memory Card Formatter from the SD Association.
- Download, install, and launch Etcher to write the image to SD card.

After the microSD card is ready, we proceed to set up and first boot phase. There are two ways to interact with the developer kit:

- 1) with display, keyboard and mouse attached.
- 2) in "headless mode" from another computer.

The first one was used.

Step 2: Initial Setup with Display Attached

- 1) Insert the microSD card (with **JetPack 4.5 image** already written on it).
- 2) Set the developer kit on a non-conductive surface with display turned on and USB keyboard and mouse attached.
- 3) Then Connect the USB-C power supply (5V=4A) and the nano will power on and boot automatically.

Step 3 First Boot: When booting for the first time, the system will take you through some initial setup, including:

- Select system language, keyboard layout, and time zone.
- Create username, password, and computer name.
- Select APP partition size. It is recommended to use the max size suggested.

Now we have a ready Ubuntu based Linux system flashed on the SD-card we proceed with the following setup.

3.3.2 Configure System

First, we connect to Wi-Fi network, then we open the terminal and do this command:

sudo systemctl set-default multi-user	Disable GUI to free up more RAM
sudo nvpmodel -m 0	Default to Max-N (10W power mode)

configure the Container (a container is an executable unit of software where an application and its run time dependencies can all be packaged together into one entity.)

We are going to use the l4t-ml docker image. It contains TensorFlow, PyTorch, JupyterLab, and other popular ML and data science frameworks such as scikit-learn, scipy, NVIDIA CUDA Deep Neural Network library, and Pandas pre-installed in a Python 3 environment.

Start by: **sudo docker pull nvcr.io/nvidia/l4t-ml:r32.5.0-py3**

“Docker is an open-source platform for creating, deploying, and running containers.”

Run it by: **sudo docker run -it --rm --net=host --runtime nvidia -e DISPLAY=\$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix --restart unless-stopped nvcr.io/nvidia/ l4t-ml:r32.5.0-py3**

The container will be automatically started after a system reboot along with **JupyterLab** server instance.

3.3.2.1 Jupyterlab

Now once the Jetson is connected to the WI-FI and the System is configurated we no longer need a monitor, we can connect to the Jetson nano from a computer or a phone web browser by performing the following steps:

- Power the jetson and wait a bit for it to boot.
- Check its IP address from the DHCP (Dynamic Host Configuration Protocol) list of the router config page, OR in our project the Wi-Fi Access point was created with an Android phone we can look up in the List of connected Devices to know the IP address of the Jetson nano.

- Once we have the jetson IP address now we can Navigate to **http://<Jetson_IP_address>:8888** from any desktop's web browser on the local network and sign in to entre a web-based interactive development environment called JupyterLab (Figure 3.12).

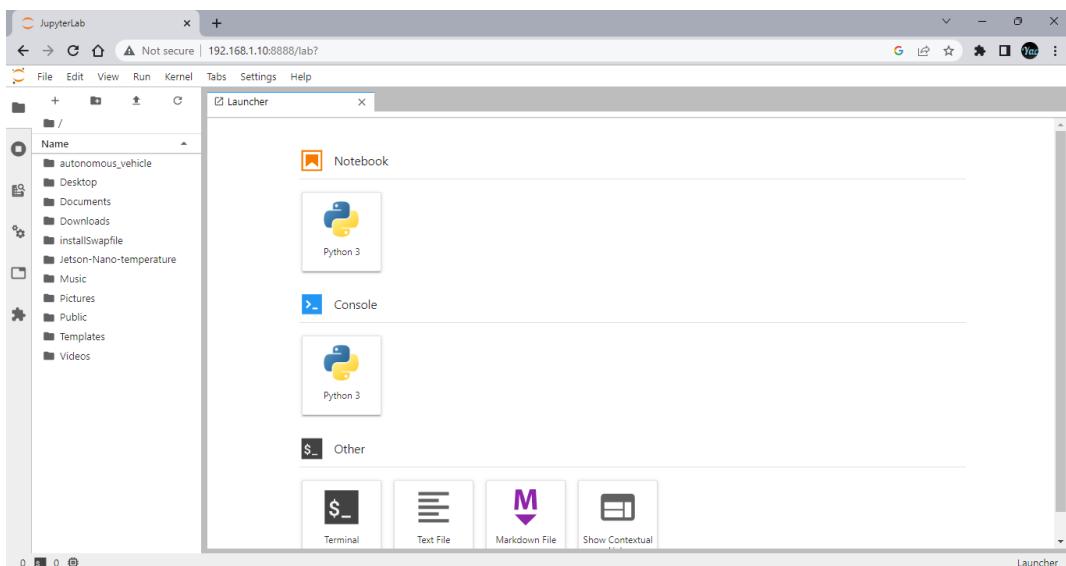


Figure 3.12: JupyterLab view.

So now we have connected all the components with each other, flashed the Jetpack 4.4 image to the SD-card and configurated the jetson nano, we can access the **JupyterLab remote programming environment (Version 2.2.6)** where all the coding will be from now on.

3.3.2.2 Adjusting PCA9585 address Frame

Is the communication between the jetson nano i2c bus and the **PCA9585**, we are using the **Adafruit-Motor-HAT-Python-Library** for that, It's an open-source Python library for interfacing with the Adafruit Motor HAT to control DC motors with speed control.

First, we need to check that the wiring is correct and the jetson nano is communicating with the PCA9685 correctly over the **i2c Bus**, we can do that by typing these commands in the Terminal (Using jupyterLab terminal or PuTTY SSH client) figure 3.13.

```
root@jetson:~# i2cdetect -r -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: --
10: --
20: --
30: --
40: 40
50: --
60: --
70: 70
root@jetson :~#
```

Figure 3.13: i2c bus check command

After scanning the I2C bus there is a connection in address 0x40, so now the wiring is correct we go to the next step.

As the Adafruit-Motor-HAT-Python-Library is Designed specifically to work with the Adafruit Motor Hat that has i2c base address of 0x60, and we work with PCA9585 that uses i2c base address of 0x40, we need to change the Library code that it works with our driver. We open the JupyterLab via browser and open the Terminal, now we navigate to Adafruit Motor HAT Library.py by this command:

```
root@jetson:/workspace# cd /usr/local/lib/python3.6/dist-packages/Adafruit_MotorHAT-1.4.0-py3.6.egg/Adafruit_MotorHAT/
root@jetson:/usr/local/lib/python3.6/dist-packages/Adafruit_MotorHAT-1.4.0-py3.6.egg/Adafruit_MotorHAT# ls
Adafruit_MotorHAT_Motors.py Adafruit_PWM_Servo_Driver.py __init__.py __pycache__
root@jetson:/usr/local/lib/python3.6/dist-packages/Adafruit_MotorHAT-1.4.0-py3.6.egg/Adafruit_MotorHAT# vim Adafruit_MotorHAT_Motors.py
```

In the Vim editor we scroll down and press **i** to edit then change the 0x60 to 0x40, now press **esc** and **:wq** and press **entre**, now we edited and saved the python file.

```
def __init__(self, addr = 0x60, freq = 1600, i2c=None, i2c_bus=None):
    def __init__(self, addr = 0x40, freq = 1600, i2c=None, i2c_bus=None):
```

Now the code is compatible with our hardware we are ready for the next step.

DC MOTORS rotation direction

A differential drive is the most basic drive and the simplest to implement (figure 3.14).

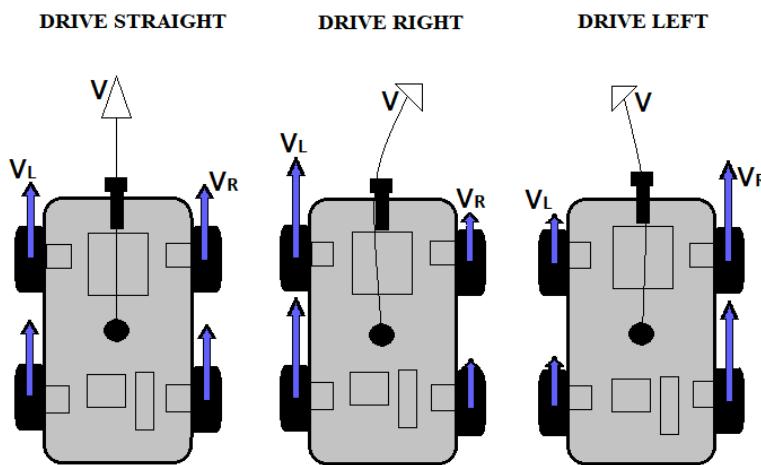


Figure 3.14: Autonomous Vehicle Differential Drive.

We used a **robot.py** code with Motor HAT Library to control the motor movement in 4 directions.

The method takes a speed value ranging from 0 to 255, the input speed is multiplied by 255.

```
def left(self, speed=1.0):
    speed = int(speed*255)
    self.motor_driver.set_drive(0, 1, speed)
    self.motor_driver.set_drive(1, 0, speed)
    self.motor_driver.enable()
```

Multiplication by 255 scales the input **speed** value from the range of 0 to 1 to the range of 0 to 255. Where: self.motor_driver.set_drive (channel A/B, Rotation R/L, speed)

We can check that the wiring of motors is correct with the L298n driver by simple code after we import the **robot.py** into a code:

```
from autonomous_vehicle import Robot
robot = Robot()
robot.left(speed=0.3)|
```

If the vehicle turns left the wiring is correct otherwise swapping the terminals of the incorrect motor direction until all 4 directions are correct.

3.4 Conception Methodology of the Autonomous Vehicle

The methodology was divided into three sections, two sections for Data collection and Training of the Road Navigation and Collision avoidance, and third one about the Deployment of models for testing phase.

3.4.1 The Experimental Environment

It consists of 2 lanes 15 centimeters wide apart, and the size of the overall track is 1.5m x 1.3m (length x width), the experimental track has many obstacles (pedestrian crossing, car, stop sign and traffic light), that was created to simulate a road going vehicle. All obstacles and experimental track shown in Figure 3.15.



Figure 3.15: Obstacle object and the test track

3.4.2 Road Navigation

The task for the autonomous vehicle is to follow the track (figure 3.15), using a neural network model the application processes images from a camera and predicts the necessary steering angle to keep the vehicle on track. This step consists of two main stages: Data Collection, and Training.

3.4.2.1 Data Collection

With the camera mounted on the vehicle a widget is initialized to show a preview on one screen and on the second one to precisely define the coordinates of the direction (figure 3.16), the key information is the coordinates of the vector and the direction in which the vehicle should move to autonomously, the coordinates are saved in the image file name (X, Y values of this orange dot) along with a unique identifier for the image in a resolution of 224x224, which is the input size required by the neural network.

Setting the vector position is a task that must be performed manually using `jupyter_clickable_image_widget`. The more varied data (different orientations, different lighting, different textured floors...) we have of scenarios the robot will encounter in the real world, the greater the chance of more precise autonomous driving. A set of (336 frames) was collected in this step. (figure 3.17)



Figure 3.16: Camera view for collecting datasets with actual Vehicle position.

Every picture taken with filenames in the format "xy_<x value><y value><uuid>.jpg", the x and y values that represent target, with a **uuid** value to ensure that each annotated image has a unique name (to avoid naming conflicts).



Figure 3.17: Samples of Dataset.

3.4.2.2 Training

We will be using PyTorch deep learning framework to make a model for a road Navigation application. The algorithm for autonomous driving was based on the ResNet18 neural network. This architecture was chosen for its effectiveness and the small size of the model. In a process called transfer learning, the code for this training is utilized to train the final fully connected (fc) layer of the (ResNet18) model for a new task that's to take an input image frame, and output a set of x, y values.

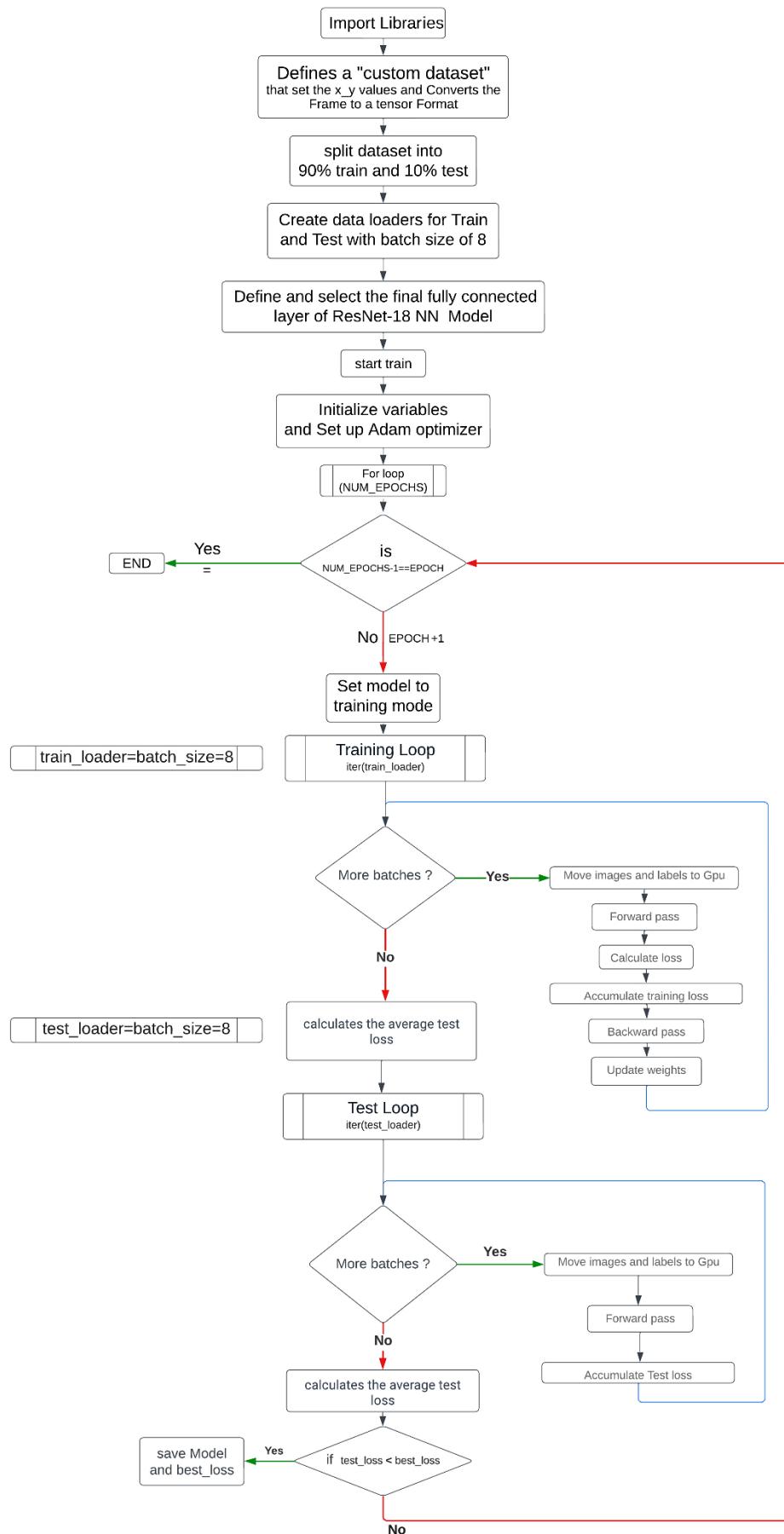


Figure 3.18: Road Navigation Training Flowchart.

As we see in (figure 3.18) Flowchart we start by Importing the needed Libraries, and:

Create Dataset Instance: that defines the x and y values and applies color jitter transformation for diversity (Data augmentation) to the images then converts the image to a rank3 **tensor** format.

Split Dataset: The (336 frames) was divided into a training set and a test set using a 90%-10% ratio, the test set will be used to evaluate the model's accuracy.

Create data loaders: a data loader is an object that helps manage and load data in batches during the training process, the data loaders are set to have a **batch size of 8**, this means that during training, the model will process 8 images and their corresponding labels together in each iteration, allowing for Parallel computing and utilization of the available computational resources.

Defining the Neural Network Model: (ResNet-18) model from the TorchVision library is used by selecting only the final fully connected layer (fc) and modifying it to have 2 outputs.

Training: During each training epoch, the model makes predictions on a batch of training samples in our case we have: (336 frame), with total samples in Training set 302 images, and 34 in testing, with batch size of 8 during each of the 70 epochs, the training loop will iterate 38 times, and the test loop will iterate 5 times.

The model uses Adam optimizer algorithm to update the parameters based on the gradients computed during the backpropagating pass, The training and testing losses are calculated using the mean squared error (MSE) for both the train and test sets. The model with the lowest test loss is saved.

During each training epoch, the model makes predictions on a batch of training samples in our case we have: (336 frame), with total samples in Training set 302 images, and 34 in testing, with **batch size of 8** during each of the 70 epochs, the training loop will iterate 38 times, and the test loop will iterate 5 times.

(Figure 3.19) simplify training procedure steps with every tool(library/algorithim) used.

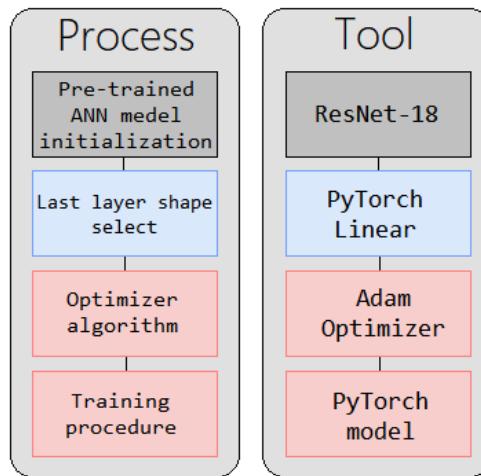


Figure 3.19: The training procedure for road Navigation ANN's Model.

3.4.3 Collision Avoiding

In this section, we see the procedure of the collision avoided algorithm. Same as the Road Navigation there are two main steps (Data collection, Training).

3.4.3.1 Data collection

The neural network used takes a 224x224 image as input. We'll create a folder dataset that will contain two sub-folders free and blocked, where we'll place the images for each scenario, a widget is created so we can capture the images in different scenarios where the road is blocked or free as (figure 3.20) and saved in two sub-folders, the more data we have of scenarios that robot will encounter, the better the collision avoidance will perform.

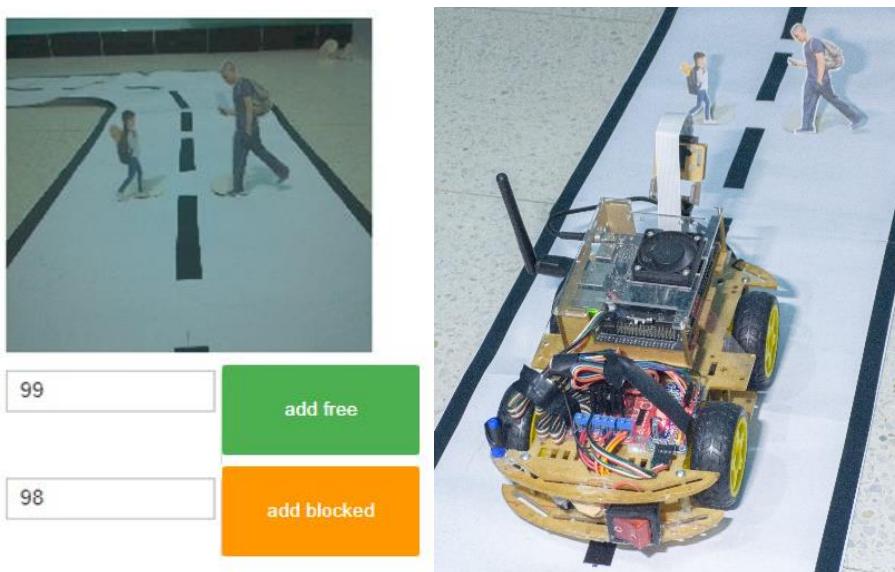


Figure 3.20: Data collection with Actual vehicle position on track.

3.4.3.2 Training

We'll train our model to detect two classes free and blocked, which we'll use for avoiding collisions. It's very similar to the Road Navigation training process.

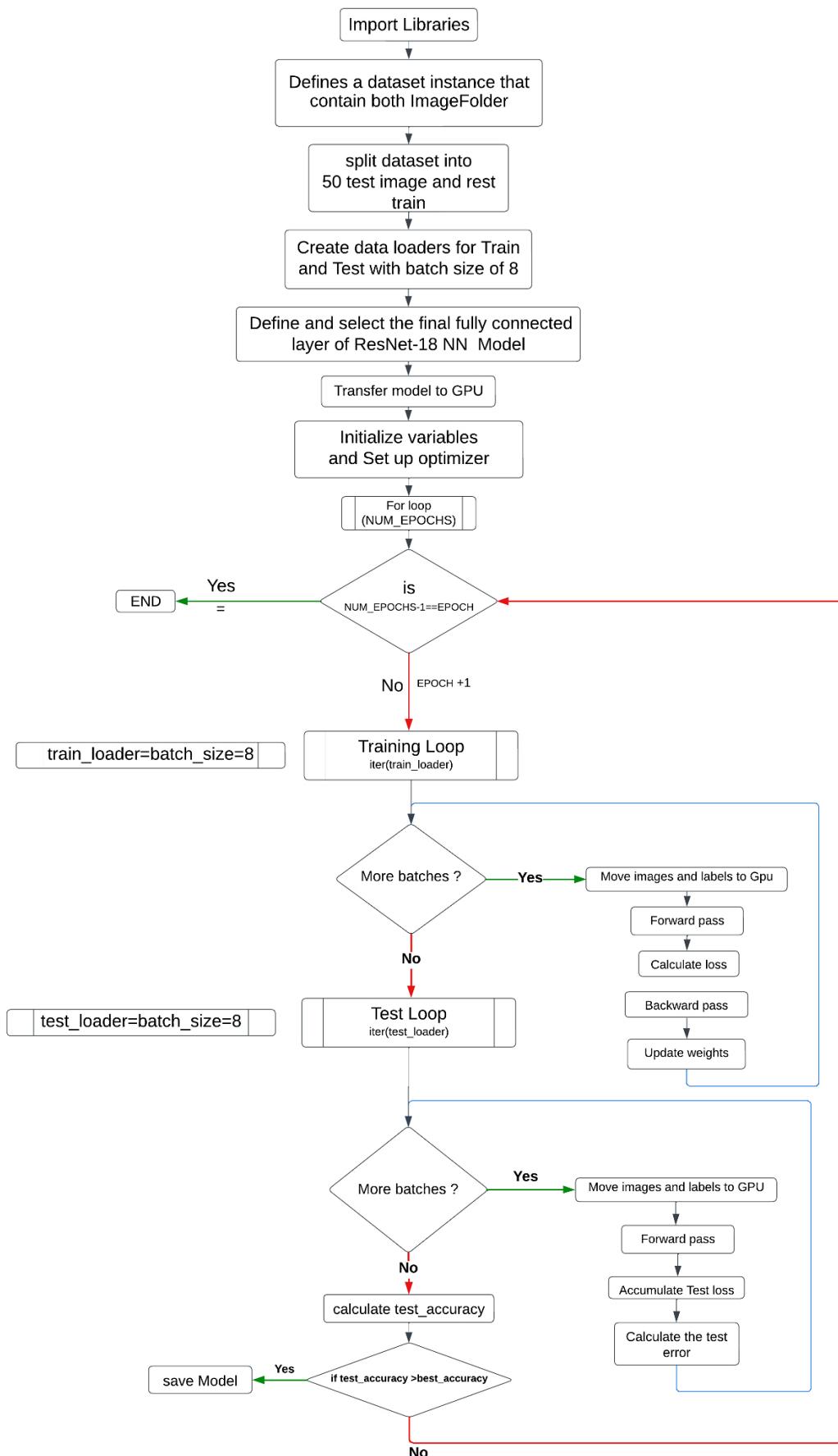


Figure 3.21: Collision Avoiding Training Flowchart.

As we see in (figure 3.21), we start by importing the necessary libraries. Then creates a dataset instance that applies ColorJitter for (Data augmentation) and converts the image to a rank 3 **tensor** format.

To evaluate the model's accuracy the dataset is split into training and test sets, the test set consisting of 50 samples. Same as previous method (from road navigation training) Two **DataLoader** instances are created for the train and test set for parallel loading of samples.

The ResNet18 model is used as the neural network architecture, the final fully connected (fc) layer is replaced with a new layer having two output features corresponding to the two classes. The model is transferred to the GPU for efficient and faster parallel computation.

Finally, we start training the neural network for a specified number of epochs, during each epoch we iterate over the training data in batches, compute the loss, perform backpropagation, and update the model's parameters using the optimizer. Then, we evaluate the model's accuracy on the test set.

If the current accuracy is higher than the previous best accuracy, we save the model's state. Once that is finished, a file best_model_resnet18.pth is saved.

3.4.4 Models Deployment

The Full code of the deployment of the two Models that we got after the Training can be explained with this Flowchart:

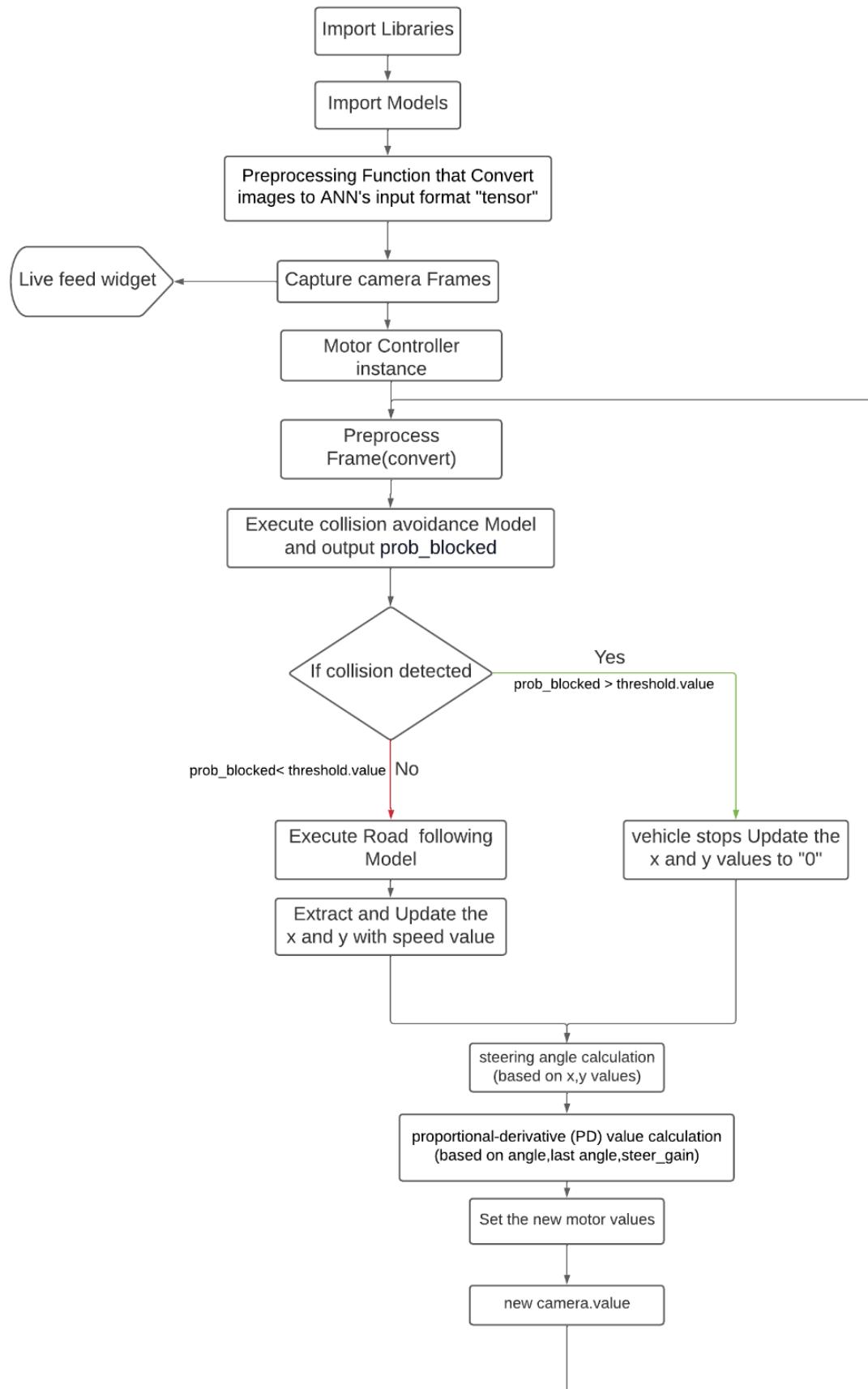


Figure 3.22: Flowchart of autonomous vehicle control.

As we see in the (figure 3.22), the Deployment phase starts by transferring the code execution from CPU memory to the GPU device to leverage its computational power. This step ensures faster processing of the neural network models that was done by:

```
import torch  
device = torch.device('cuda')
```

First step is importing the necessary libraries then follows by:

Loading Models: the two previously trained models for road Navigation and collision avoidance are loaded using PyTorch deep learning framework,

Defining the Preprocess Function: that converts the image format to the neural network input (rank3 tensor).

Initializing Camera: A camera instance is initialized with a specific width, height, and frame rate, with a widget for displaying live feed preview.

Creating motor control Instance: instance is called to allow the control over the motors.

The speed, steering gain, steering bias, and collision detection threshold are parameters that need to be set and adjusted manually for a better performance.

In the last step an execution function is defined that gets called whenever the camera's value changes. This function performs the following steps:

- Pre-processes the camera frames.
- Executes the collision avoidance model to determine the probability of an obstacle in front of the vehicle then compare it with the threshold value.
- Check if the road is free continue road Navigation or blocked to stop the vehicle.
- Calculates the steering value (angle) based on (x, y) new values and old angle value with proportional and derivative control (PD) to ensure smooth navigation.

3.4.4.1 PD Controller feedback System

In this project, the motors of the autonomous vehicle were controlled using a proportional-derivative (PD) control algorithm. The PD control is commonly used to achieve accurate and responsive control. It's a feedback control strategy used in systems to regulate a variable process based on the deviation between the desired setpoint and the measured value. It consists of two main components: the proportional (P) control and the derivative (D) control.

Proportional Control (P): produces an output that is directly proportional to the error. It is determined by multiplying the error by the proportional gain (Kp).

Derivative Control (D): measures the rate of change of the error over time. It provides a damping effect by anticipating the future trend of the error. Determined by calculating the difference between the current angle and the previous angle, representing the change in angle, and multiplying this difference by derivative gain (Kd).

The combined action of the proportional and derivative components in PD control helps in achieving a faster response, and improved stability.

The formula used for the PD control:

$$\text{PD} = (\text{angle} \times \text{steer_gain}) + ((\text{angle} - \text{angle_last}) \times \text{steer_dgain})$$

'angle' in radians, represents the current orientation of the vehicle with respect to desired trajectory that is calculated based on the values of x_y given by the model.

'angle_last' stored from a previous orientation value, 'steer_gain' and 'steer_dgain' are the proportional and derivative gains Kp, Kd respectively values.

The control signal, **PD**, is obtained by adding the proportional and derivative terms.

To ensure smooth and safe operation, the control signal **PD** is then added to a bias term, 'steer_bias' to get last 'steer_val', this last value with the speed value will directly set the motors speed If 'angle' is positive vehicle will turn left, and if its negative it will turn right. (figure 3.23).

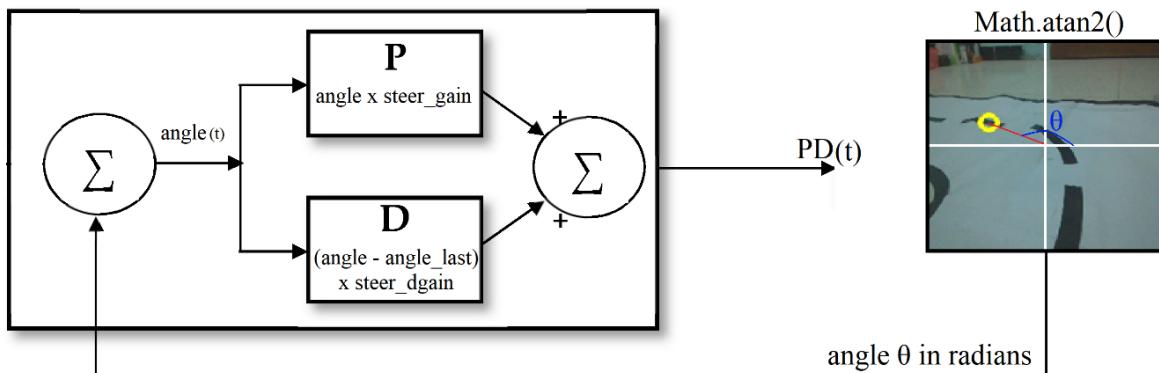
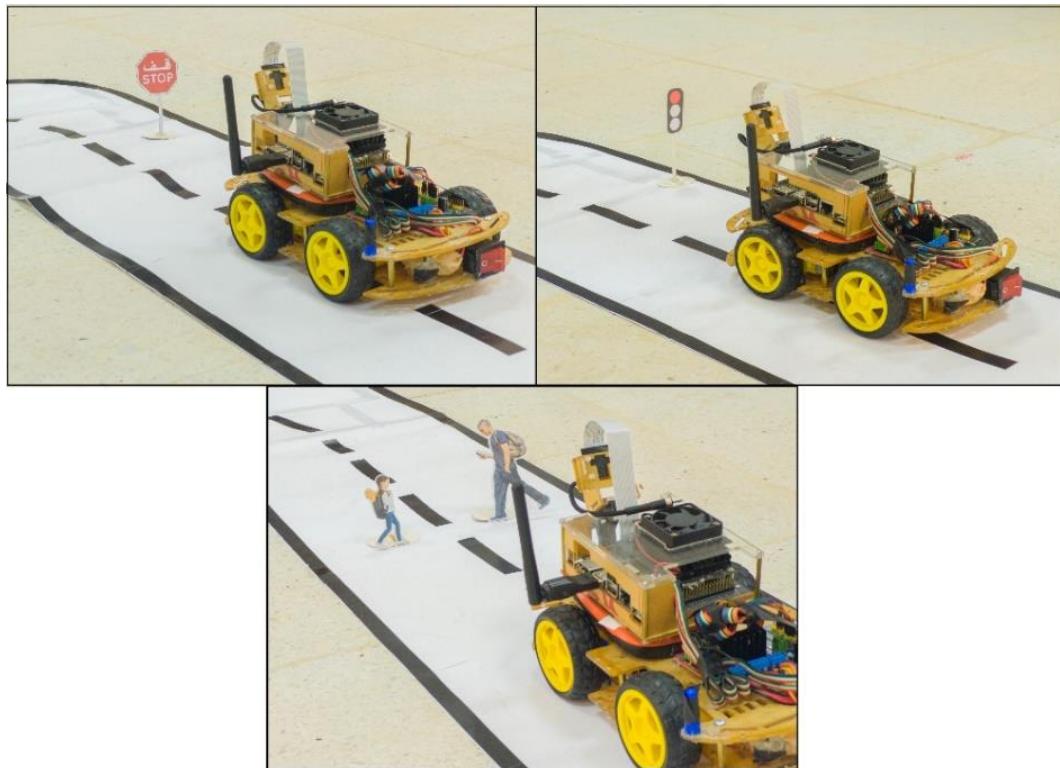


Figure 3.23: Proportional-Derivative (PD) Steering controller Feedback system.

3.5 Experimental Results

The Testing phase was done on the Experimental Environment, the vehicle was able to recognize the obstacles (pedestrian, stop sign, car, and traffic light) and react by avoiding them, and follow planned trajectory smoothly with a great accuracy rate, it was an autonomous driving experience.



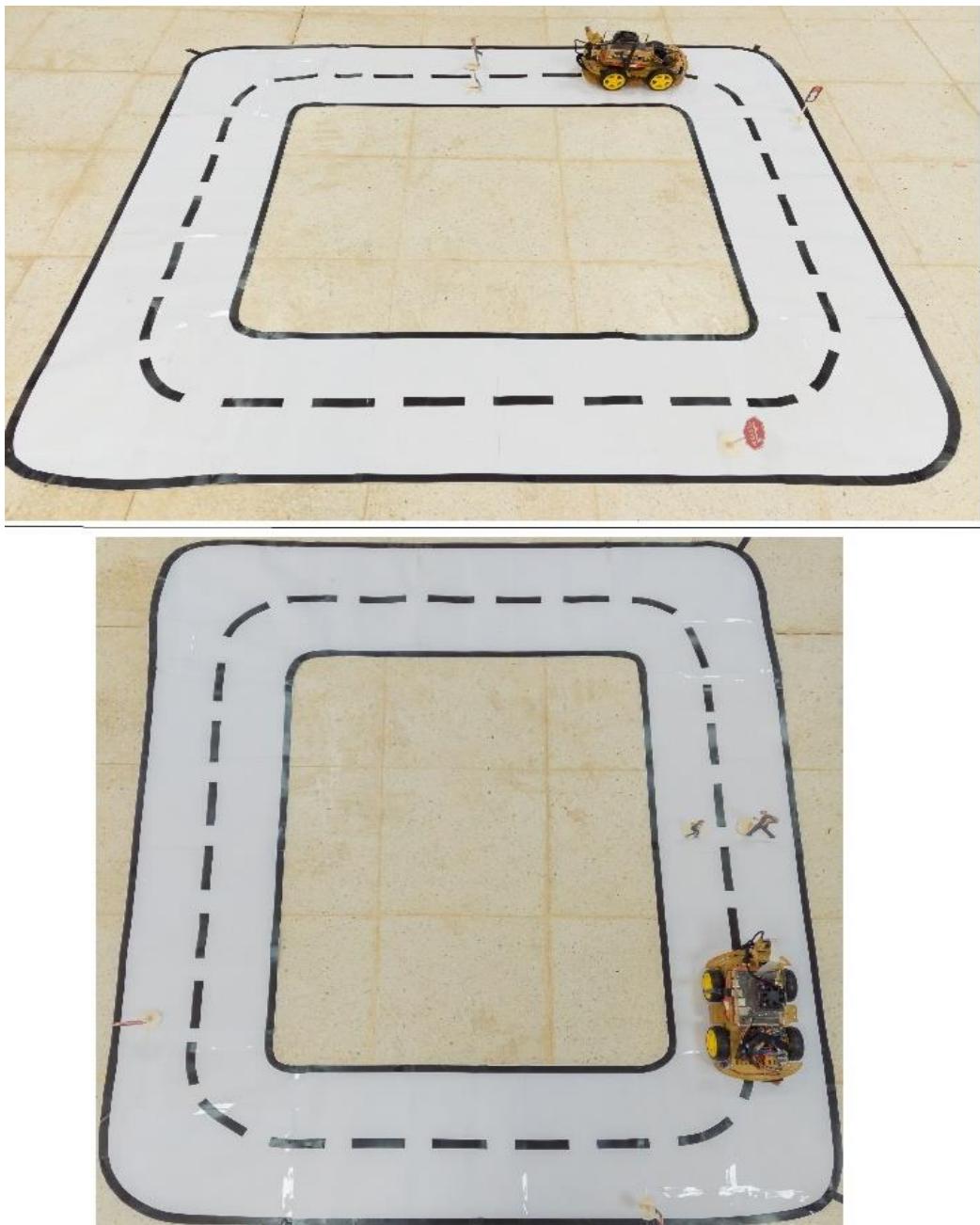
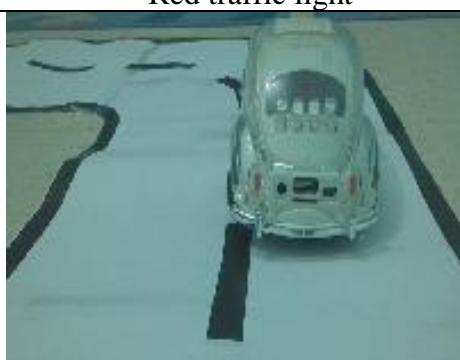


Figure 3.24: Experiment Result.

In this test, we will see collision probability and All steering parameters at different scenarios in the test environment, from different obstacles and turns to different lighting and speed states, with results shown below:

All steering parameters	Collision probability	Scenario
X= -0.23 Y= 0.20 Angle: -0.8550 PD: -0.2394 Steer Value: - 0.1794 Left Speed Motor: 0.0505 Right Speed Motor: 0.4094	02 %	 Green traffic light
X=- 0.13 Y= 0.20 Angle: -0.5763 PD: -0.0844 Steer Value: - 0.0244 Left Speed Motor: 0.2055 Right Speed Motor: 0.2544	98 %	 Red traffic light
X=- 0.10 Y= 0.06 Angle: -1.0303 PD: -0.2366 Steer Value: - 0.1766 Left Speed Motor: 0.0533 Right Speed Motor: 0.4066	99 %	 Car
X=- 0.07 Y= 0.19 Angle: -0.3529 PD: -0.0061 Steer Value: 0.0538 Left Speed Motor: 0.2838 Right Speed Motor: 0.1761	70-87 %	 Pedestrian

X=-0.05 Y=0.20 Angle: -0.2449 PD: -0.0368 Steer Value: 0.0231 Left Speed Motor: 0.2531 Right Speed Motor: 0.2068	81-95 %	 Stop sign
X= -0.29 Y=0.13 Angle: -1.1493 PD: -0.2997 Steer Value: -0.2397 Left Speed Motor: 0 Right Speed Motor: 0.4697	02 %	 Left Turn
X=-0.4 Y=0.15 Angle: 1.2120 PD: 0.4428 Steer Value: 0.5028 Left Speed Motor: 0.7328 Right Speed Motor: 0	01 %	 Right turn

Figure 3.25: Models result at Different Scenario.

In the next test **10 laps** was performed in each different lighting and speeds scenarios, the result is in (table 3.4) below, the vehicle performed well in the optimal condition (lighting, speed, DC motors current), but the result was poor in scenarios that differ from the collected dataset as this is one of its biggest limitations.

Speed (m/s)	Percentage of Correct Laps (%)	Lighting Condition	details
≤ 0.20	100%	Daylight (same as datasets)	Too slow, but Completed the 10 laps without deviation
0.40	90%		Average speed Completed 9 laps without deviation
> 0.40	10%		Deviated from the track at a speed of 0.4m/s and more
≤ 0.20	60%	Dim Light (low lighting)	Completed 6 laps without deviation.
0.40	40%		Completed 4 laps without deviation
> 0.40	0%		Vehicle can't steer at all
≤ 0.20	0%	Night (super low lights)	Vehicle can't steer in all speeds
0.40			
0.40			

Table 3.4: Experimental Results.

Note: The dataset for training contains only daylight scenario.

The results were fluctuating in the optimal condition vehicle archived its primary goal with an efficient road Navigation capability staying on the correct trajectory while reacting to the obstacles implemented in its path, but otherwise in low lighting and higher speeds the vehicle was deviating from track or not reacting to the obstacles limiting its capabilities.

3.6 Problems Faced in the Implementation and their solutions

Throughout the project, several problems accrued, from the selection of suitable hardware, to freezing issues in deployment due to low memory on the Jetson Nano, and inadequate torque from the DC motors.

The initial challenge encountered was selecting the appropriate hardware components for the autonomous vehicle. This decision required careful consideration of factors such as processing power, and memory capacity.

The chosen hardware configuration should ensure the optimal performance of the autonomous vehicle.

Freezing Issues due to Low Memory

During the code deployment phase, the Jetson Nano would freeze immediately after running the code, it was due to insufficient memory.

To remove the freezing issue caused by low memory, the solution was to limit the FPS (frames per second) to **10** and add an 8GB swap memory to augment the existing memory resources, with that we prevent memory overconsumption and reduced computational load on the Jetson Nano for further stability and reliability.

Dc motors Low Torque

The DC experienced low torque, which affected the vehicle's maneuverability and overall performance. In order to address the issue, the solution was an adjustment to the PWM frequency. By trying different frequencies, the **24 Hz** was the best, the motors torque output was significantly improved, and robot moved without a problem.

3.7 Improving Performance of the Autonomous Vehicle prototype

Various parameters can be tuned to optimize and improve the performance of the vehicle resulting in more efficient and reliable autonomous driving, like:

Expanding Field of View with a Wide Lens Camera: The field of view (FOV) plays a crucial role in enabling the autonomous vehicle to perceive its surroundings accurately. By incorporating a wide lens camera, the FOV can be expanded, allowing the vehicle to capture a broader perspective of the road and surrounding objects.

Adding a Light source: Adding a light (LED as it's the best for low power usage) to the vehicle would enhance its performance in the low light/dark scenarios, minimizing its limitation furthermore.

Optimizing Look-Ahead Point Selection: The selection of the look-ahead point is another critical parameter that can significantly impact the performance of the autonomous vehicle. By adjusting the distance at which the vehicle anticipates its trajectory, the vehicle can react more effectively to road conditions and potential obstacles.

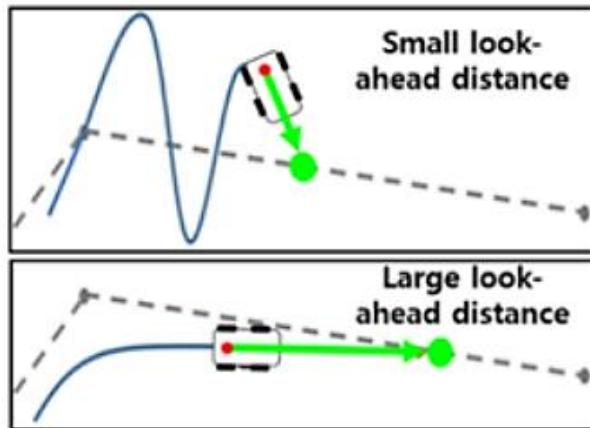


Figure 3.26: Look ahead point distance effect [84].

More data: we can include different environment, and lighting conditions and more overall data (more scenarios), that will make the vehicle performance better and more reliable.

3.8 Limitation of the vehicle

One of the primary limitations of the developed autonomous vehicle is its dependency on the trained environment. While the vehicle demonstrates efficient performance within the specific environment it was trained in, it may encounter challenges when faced with different or unseen environments and lighting conditions. Making it hard for the adaptability in a real-world deployment of the autonomous vehicles. To achieve robust and reliable performance, the system needs to be trained and validated in diverse environments that encompass a wide range of road types, weather conditions, and traffic scenarios, different lighting..., also the speed of the vehicle is limited due to hardware capabilities an increasing computing power will lead to a higher Frame rates eventually higher speeds of the vehicle.

3.9 Conclusion

In Chapter 3, we delved into the conception methodology of our autonomous vehicle. We began by introducing the system architecture, including the hardware components and assembly process, then we established the experimental environment and outlined the approach for road Navigation and collision avoidance including system setup and coding flowchart at different stages, with showcasing the experimental results, and the performance of our autonomous vehicle.

GENERAL CONCLUSION

The goal of this master's dissertation was to develop an autonomous vehicle that's has the capability of road navigation in pair with collision avoidance (stop sign, pedestrian, traffic light, car). Through the utilization of deep learning algorithms, a transfer learning on the pretrained ResNet-18 architecture was employed on our specific dataset using a Jetson Nano SBC as the primary computing device paired with CSI camera. By leveraging the strengths of ResNet-18, we aimed to enhance the accuracy and efficiency of the road navigation and collision avoidance tasks in our autonomous vehicle system.

For the testing procedure, an experimental track was constructed that has different scenarios aiming for a road going vehicle simulation, the experiment tests shows that the vehicle archived its primary goal in certain situations (ideal ones) with an efficient road Navigation capability staying on the correct trajectory with a smooth driving while reacting to the obstacles implemented in its path, ultimately leading to successful navigation.

But the vehicle performed poorly in the different lighting or higher speeds conditions as we have seen in our tests, Its performance is currently dependent on the trained environment, making it less adaptable to different or unseen environments. To address this, the system should be trained and validated in diverse environments that encompass a wide range of road types, weather conditions, and traffic scenarios. This would contribute to achieving robust and reliable performance in real-world deployments of autonomous vehicles.

Perspective and Future work for this project

As for future work, several potential enhancements can be considered in area such as:

- Integrating **LiDAR** sensor would provide valuable additions to the perception system. It can offer accurate distance measurements, improving mapping capabilities.
- Implementing advanced **object recognition** algorithms would enhance the vehicle's ability to interpret and understand its environment, enabling it to make informed decisions and respond appropriately in various scenarios.
- Increasing the quantity and diversity of training data has the potential to improved performance and reliability of the vehicle in different scenarios.

- Upgrading to a more powerful computing platform can significantly contribute to improving the performance of the autonomous vehicle system. With increased computing power, the system can handle more intensive tasks, such as real-time sensor fusion, and object detection at higher frame rate (FPS) enabling the system to react faster to dynamic changes in the environment.

In conclusion, this dissertation covered the basic principles of developing an autonomous vehicle that could effectively navigate through a road and stop at obstacles in limited conditions. The suggested improvements and future work outlined above will pave the way for further advancements in the prototype, making the limitation smaller and enhancing its performance. So, we can keep improving in little steps paving the way towards safer and more efficient transportation systems in the near future.

REFERENCES

- [1] “Advantages and Disadvantages of Automation - Javatpoint,” [www.javatpoint.com](http://www.javatpoint.com/advantages-and-disadvantages-of-automation).
<https://www.javatpoint.com/advantages-and-disadvantages-of-automation> (accessed Jun. 09, 2023).
- [2] “What is Intelligent Automation? | IBM.” <https://www.ibm.com/topics/intelligent-automation>
- [3] “What is an Autonomous Car? – How Self-Driving Cars Work | Synopsys.”
<https://www.synopsys.com/automotive/what-is-autonomous-car.html>
- [4] C. Engelking, “The ‘Driverless’ Car Era Began More Than 90 Years Ago,” Discover Magazine, May 2019, [Online]. Available:
<https://www.discovermagazine.com/technology/the-driverless-car-era-began-more-than-90-years-ago>
- [5] Stanford University, “Stanford’s robotics legacy | Stanford News,” Stanford News, May 23, 2019. <https://news.stanford.edu/2019/01/16/stanfords-robotics-legacy/> (accessed Jun. 09, 2023).
- [6] A. Reiser, “History of Autonomous Cars,” TOMORROW’S WORLD TODAY®, Apr. 2023, [Online]. Available: <https://www.tomorrowsworldtoday.com/2021/08/09/history-of-autonomous-cars/>
- [7] “Figure 2 1977. Tsukuba Mechanical Engineering Lab, Japan, 1977....,” ResearchGate. https://www.researchgate.net/figure/Tsukuba-Mechanical-Engineering-Lab-Japan-1977-computerized-driverless-car-achieved-spe_fig2_365874855 (accessed Jun. 09, 2023).
- [8] “Fig.6 ‘Navlab 5’ vehicle by Carnegie Mellon.,” ResearchGate.
https://www.researchgate.net/figure/Navlab-5-vehicle-by-Carnegie-Mellon_fig4_338412717 (accessed Jun. 09, 2023).
- [9] M. Silverio, “What Are Self-Driving Cars?,” Built In, Aug. 2021, [Online]. Available: <https://builtin.com/transportation-tech/self-driving-cars>
- [10] Palanalytics, “ARan - The Smartest AMR Designed For You,” PAL Robotics, Apr. 13, 2023. <https://pal-robotics.com/robots/aran/> (accessed Jun. 09, 2023).
- [11] A. , “Types of Autonomous Mobile Robots (AMRs) and Their Use in Warehousing!,” Supply Chain Game Changer™, Apr. 2023, [Online]. Available: <https://supplychaingamechanger.com/types-of-autonomous-mobile-robots-amrs-and-their-use-in-warehousing/>
- [12] “The Mission of SAE International is to advance mobility knowledge and solutions,” Jun. 08, 2023. <https://www.sae.org/>

-
- [13] “SAE J3016 automated-driving graphic,” May 15, 2020.
<https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic#:~:text=The%20J3016%20standard%20defines%20six,graphic%20first%20deployed%20in%202016.> (accessed Jun. 09, 2023).
- [14] “Footage Audi A8: Audi AI traffic jam pilot - Audi MediaTV.” <https://www.audi-mediacenter.com/en/audimediatv/video/footage-audi-a8-audi-ai-traffic-jam-pilot-3785>
- [15] Waymo, “Sense, Solve, and Go: The Magic of the Waymo Driver,” YouTube. Aug. 01, 2022. [Online]. Available: https://www.youtube.com/watch?v=hA_-MkU0Nfw
- [16] “Audi Aicon (2017),” Audi MediaCenter. <https://www.audi-mediacenter.com/en/audi-aicon-2017-9299>
- [17] “The 6 Levels of Vehicle Autonomy Explained | Synopsys Automotive.” <https://www.synopsys.com/automotive/autonomous-driving-levels.html>
- [18] “truly self-driving cars,” Twitter,
<https://twitter.com/BillGates/status/1641496541625221121>, Mar. 30, 2023.
- [19] جامعة البويرة (الجزائر), ”د. ف. مساني, ”قراءة إحصائية لحوادث المرور في الجزائر 1AD.
- [20] “2022 حوادث المرور خلال 2022,” <https://www.aps.dz/>, Jan. 2023, Accessed: Jun. 09, 2023,
<https://www.aps.dz/ar/societe/138205-2022-700-20575#:~:text=%D9%88%D9%83%D8%B4%D9%81%20%D8%A7%D9%84%D9%85%D8%B1%D8%A7%D9%82%D8%A8%20%D8%A7%D9%84%D8%B9%D8%A7%D9%85%20%D9%84%D9%84%D8%B4%D8%B1%D8%B7%D8%A9%20%D9%81%D9%8A,74%20%D8%B1%200%20%D8%A8%D8%A7%D9%84%D9%85%D8%A6%D8%A9%20%D9%88%D9%81%D9%8A>
- [21] Wang, J.-S, “Target Crash Population For Crash Avoidance Technologies in Passenger Vehicles,” Washington, DC: National Highway Traffic Safety Administration, Art. no. (Report No. DOT HS 812 653), Mar. 2019.
- [22] K. Kockelman, “The future of fully automated vehicles : opportunities for vehicle-and ride-sharing, with cost and emissions savings,” Aug. 01, 2014.
<https://repositories.lib.utexas.edu/handle/2152/25932>
- [23] J. Díaz-Dorronsoro, “Self-Driving Car Autonomous System Overview - Industrial Electronics Engineering - Bachelors’ Thesis -,” Oct. 01, 2020.
<https://dadun.unav.edu/handle/10171/59563>
- [24] “Self-Driving Car Autonomous System Overview,” - Industrial Electronics Engineering - Bachelors’ Thesis -, 2020.
- [25] K. Gülen, “Artificial Intelligence And Self-driving Cars Explained,” Dataconomy, Dec. 29, 2022. [Online]. Available: <https://dataconomy.com/2022/12/28/artificial-intelligence-and-self-driving/>

-
- [26] “Autonomous vehicles sensor ecosystem,” ResearchGate.
https://www.researchgate.net/figure/Autonomous-vehicles-sensor-ecosystem-Image-source-The-Economist_fig4_354203119 (accessed Jun. 09, 2023).
- [27] “Autonomous Systems | Ultimate Guides | BlackBerry QNX.”
<https://blackberry.qnx.com/en/ultimate-guides/autonomous-systems>
- [28] “OVERVIEW artificial intelligence,” Oxfordreference.
<https://www.oxfordreference.com/display/10.1093/oi/authority.20110803095426960;jsessionid=0E1A994FE9302CBA65046394E3824514> (accessed Jun. 09, 2023).
- [29] K. Kelley, “What is Artificial Intelligence: Types, History, and Future,” Simplilearn.com, Mar. 2023, [Online]. Available:
<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-artificial-intelligence>
- [30] A. Biswal, “7 Types of Artificial Intelligence That You Should Know in 2023,” Simplilearn.com, May 2023, [Online]. Available:
<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/types-of-artificial-intelligence>
- [31] M. Spremić, “The implementation of artificial intelligence and its future potential,” REPEFZG, Sep. 03, 2019. <https://repozitorij.efzg.unizg.hr/en/islandora/object/efzg%3A2697>
- [32] Tony J Prescott, “How Kismet Works,” YouTube. Nov. 07, 2016. [Online]. Available: <https://www.youtube.com/watch?v=Kw-gOmJwzuc>
- [33] CNBC, “Interview With The Lifelike Hot Robot Named Sophia (Full) | CNBC,” YouTube. Oct. 25, 2017. [Online]. Available:
<https://www.youtube.com/watch?v=S5t6K9iwcdw>
- [34] Kismet-IMG 6007. [Online]. Available:
https://commons.wikimedia.org/wiki/File:Kismet-IMG_6007-white.jpg
- [35] I. Pictures, ITU Featured Photos. [Online]. Available:
<https://www.flickr.com/photos/itupictures/27254369347/>
- [36] L. Edu, “Applications of Artificial Intelligence | Leverage Edu,” Leverage Edu, Mar. 06, 2022. <https://leverageedu.com/blog/applications-of-artificial-intelligence/>
- [37] Q. A.-P. a P. W. Movement, “Applications of Artificial Intelligence Across Various Industries,” Forbes, Jan. 06, 2023. [Online]. Available:
<https://www.forbes.com/sites/qai/2023/01/06/applications-of-artificial-intelligence/?sh=4ef3e2303be4>
- [38] “Fig. 1. Representation of number of ways AI can be achieved,” ResearchGate.
https://www.researchgate.net/figure/Representation-of-number-of-ways-AI-can-be-achieved-6_fig1_343638158

-
- [39] J. Borana, “Applications of Artificial Intelligence & Associated Technologies,” 2016. <https://www.semanticscholar.org/paper/Applications-of-Artificial-Intelligence-%26-Borana/d5b061e6565ce421b4b0b7d56296e882085dc308>
- [40] “What is Computer Vision? | IBM.” <https://www.ibm.com/topics/computer-vision#:~:text=Resources-,What%20is%20computer%20vision%3F,recommendations%20based%20on%20that%20information.>
- [41] Manning Publications, “How Does Computer Vision Work? - Manning,” Manning, Aug. 18, 2019. <https://freecontent.manning.com/mental-model-graphic-grokking-deep-learning-for-computer-vision/>
- [42] “Medium,” Medium. <https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e,%20Mike%20Tamir>
- [43] Educative, “What are digital images?,” Educative: Interactive Courses for Software Developers. <https://www.educative.io/answers/what-are-digital-images>
- [44] “A Review of Convolutional Neural Networks,” Scientific Figure on ResearchGate. https://www.researchgate.net/figure/Pixel-Map-for-a-handwritten-digit-CSuperDataScience_fig1_340968733 (accessed Jun. 09, 2023).
- [45] “Signal Processing Techniques,” e2eml.school, Nov. 14, 2019. https://e2eml.school/convert_rgb_to_grayscale.html (accessed Jun. 09, 2023).
- [46] R. Pramoditha, “How RGB and Grayscale Images Are Represented in NumPy Arrays,” Medium, Jan. 04, 2022. [Online]. Available: <https://towardsdatascience.com/exploring-the-mnist-digits-dataset-7ff62631766a>
- [47] R. Pramoditha, “How RGB and Grayscale Images Are Represented in NumPy Arrays,” Medium, Jan. 04, 2022. [Online]. Available: <https://towardsdatascience.com/exploring-the-mnist-digits-dataset-7ff62631766a>
- [48] OpenCV, “About - OpenCV,” OpenCV, Nov. 04, 2020. [https://opencv.org/about/#:~:text=OpenCV%20\(Open%20Source%20Computer%20Vision,pexception%20in%20the%20commercial%20products.](https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,pexception%20in%20the%20commercial%20products.)
- [49] “Evolution of Artificial Intelligence,” ResearchGate. https://www.researchgate.net/figure/Evolution-of-Artificial-Intelligence-23_fig1_348174488 (accessed Jun. 09, 2023).
- [50] “What is Machine Learning? | IBM.” <https://www.ibm.com/topics/machine-learning>
- [51] G. Authors and G. Authors, “Machine Learning Overview: Everything You Need to Know,” My TechDecisions, Aug. 27, 2019. [Online]. Available: <https://mytechdecisions.com/it-infrastructure/machine-learning/>

-
- [52] “Types of Machine Learning - Javatpoint,” [www.javatpoint.com](http://www.javatpoint.com/types-of-machine-learning).
<https://www.javatpoint.com/types-of-machine-learning>
- [53] R. Raj, “Supervised, Unsupervised and Semi-supervised Learning with Real-life Usecase.” <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>
- [54] S. Laurent, “Machine Learning Algorithms: A Review,” ResearchGate, Aug. 2022, [Online]. Available:
https://www.researchgate.net/publication/362711297_Machine_Learning_Algorithms_A_Review
- [55] Trekhleb, “GitHub - trekhleb/homemade-machine-learning: Python examples of popular machine learning algorithms with interactive Jupyter demos and math being explained,” GitHub. <https://github.com/trekhleb/homemade-machine-learning>
- [56] GeeksforGeeks, “7 Major Challenges Faced By Machine Learning Professionals,” GeeksforGeeks, Oct. 2021, [Online]. Available: <https://www.geeksforgeeks.org/7-major-challenges-faced-by-machine-learning-professionals/>
- [57] “What is Deep Learning? | IBM.” <https://www.ibm.com/topics/deep-learning>
- [58] A. Abraham, “Artificial Neural Networks,” Handbook of Measuring System Design, Jul. 2005, doi: 10.1002/0471497398.mm421.
- [59] Administrator, “Artificial Neural Networks (ANN) | Basics, Characteristics, Elements, Types,” ElectronicsHub, May 2019, [Online]. Available:
<https://www.electronicshub.org/artificial-neural-networks-ann/>
- [60] A. Ponraj, “Artificial Neural Network Explained with a Regression Example,” DevSkrol, Jan. 2022, [Online]. Available: <https://devskrol.com/2020/11/22/388/>
- [61] “Neural networks: evolution, topologies, learning algorithms and application,” Scientific Figure on ResearchGate. https://www.researchgate.net/figure/Commonly-used-neural-network-activation-functions-a-Binary-threshold-b-Bipolar_fig1_236268473 (accessed Jun. 09, 2023).
- [62] “Neuromorphic Spiking Neural Networks and Their Memristor,” Scientific Figure on ResearchGate. https://www.researchgate.net/figure/a-Architecture-of-a-single-layer-perceptron-The-architecture-consists-of-a-layer-on_fig2_335438509 (accessed Jun. 09, 2023).
- [63] “Artificial neural networks: a tutorial,” IEEE Journals & Magazine | IEEE Xplore, Mar. 01, 1996. <https://ieeexplore.ieee.org/document/485891>
- [64] R. Yamashita, M. Nishio, R. K. G DO, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” Insights Into Imaging, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/s13244-018-0639-9.

-
- [65] S. Balaji, “Binary Image classifier CNN using TensorFlow - Techiepedia - Medium,” Medium, May 30, 2023. [Online]. Available: <https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697>
- [66] “Neural Networks and Introduction to Deep Learning,” www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-hdstat-rnn-deep-learning.pdf. Accessed: Jun. 10, 2023. [Online]. Available: [https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-hdstat-rnn-deep-learning.pdf](http://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-hdstat-rnn-deep-learning.pdf)
- [67] “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog,” May 16, 2023. <https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/>
- [68] “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog,” May 16, 2023. <https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/>
- [69] “Pooling (CNN) — EpyNN 1.0 documentation.” <https://epynn.net/Pooling.html>
- [70] S. Saxena, “The Architecture of Lenet-5,” Analytics Vidhya, Mar. 2021, [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/>
- [71] S. Saxena, “The Architecture of Lenet-5,” Analytics Vidhya, Mar. 2021, [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/#:~:text=What%20is%20Lenet5%3F,handwritten%20and%20machine%2Dprinted%20characters>.
- [72] P. Ratan, “What is the Convolutional Neural Network Architecture?” Analytics Vidhya, Jan. 2021, [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/>
- [73] “What is PyTorch?,” NVIDIA Data Science Glossary. <https://www.nvidia.com/en-us/glossary/data-science/pytorch/>
- [74] J. Brownlee, “Transfer Learning in Keras with Computer Vision Models,” MachineLearningMastery.com, Aug. 2020, [Online]. Available: <https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>
- [75] M. Gao, P. Song, F. Wang, J. Liu, A. Mandelis, and D. Qi, “A Novel Deep Convolutional Neural Network Based on ResNet-18 and Transfer Learning for Detection of Wood Knot Defects,” Journal of Sensors, vol. 2021, pp. 1–16, Aug. 2021, doi: 10.1155/2021/4428964.
- [76] Convolutional Neural Networks for On-Board Cloud Screening - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/ResNet-with-18-layers-depicted-in-five-blocks_fig4_333787753 [accessed 12 Jun, 2023]

-
- [77] “NVIDIA Jetson Nano 2GB Developer Kit, Get Hands-on with AI, and Robotics. Small Size. Small Price. Big AI Discoveries.” <https://www.waveshare.com/jetson-nano-2gb-developer-kit.htm?sku=18605>
- [78] Ericyu, “How to use PWM on Jetson Nano - Latest Open Tech From Seeed,” Latest Open Tech From Seeed, Jun. 02, 2020.
<https://www.seeedstudio.com/blog/2020/05/27/configure-pwm-output-on-jetson-nano-m/>
- [79] “8MP IMX219-77 Camera for NVIDIA Jetson Nano Developer Kit,” MakerFocus.
<https://www.makerfocus.com/products/8mp-imx219-77-camera-for-nvidia-jetson-nano-developer-kit>
- [80] Probots Techno Solution, “PCA9685 16 Channel Servo Motor Driver Controller Module,” Probots Techno Solution. <https://probots.co.in/16-channel-servo-motor-driver-module-pca9685-for-arduino.html#additional>
- [81] “PCA9685 16-Channel 12 Bit I2C Bus PWM Driver - Wiki.”
http://wiki.sunfounder.cc/index.php?title=PCA9685_16-Channel_12_Bit_I2C_Bus_PWM_Driver
- [82] “Amazon.com: EMOZNY 4 Wheel 2 Layer Robot Smart Car Chassis Kits with Speed Encoder for Arduino DIY (Yellow): Toys & Games.” <https://www.amazon.com/wheel-layer-Chassis-Encoder-Arduino/dp/B06VTP8XBQ>
- [83] “NVIDIA Jetson Nano 2GB Developer Kit - Get Started,” NVIDIA Developer, Mar. 22, 2023. <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit#write>
- [84] J.-W. Ahn, S. Shin, M. S. Kim, and J. Park, “Accurate Path Tracking by Adjusting Look-Ahead Point in Pure Pursuit Method,” International Journal of Automotive Technology, vol. 22, no. 1, pp. 119–129, Feb. 2021, doi: 10.1007/s12239-021-0013-7.