République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
École nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département électronique

End-of-study project dissertation
for obtaining the State Engineer's degree in (Electronics)

UAV speech control for human drone interaction

Amina BOULKOUT

Under the supervision of Dr. Cherif LARBES Prof. ENP, Algiers

Presented and defended on July 4th, 2023 before the members of jury :

President	Rachid	ZERGUI	MAA.	ENP, Alger
Supervisor	Cherif	LARBES	Prof.	ENP, Alger
Examiner	Mohammed O.	TAGHI	MAA.	ENP, Alger

ENP 2023
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
École nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département électronique

End-of-study project dissertation
for obtaining the State Engineer's degree in (Electronics)

UAV speech control for human drone interaction

Amina BOULKOUT

Under the supervision of Dr. Cherif LARBES Prof. ENP, Algiers

Presented and defended on July 4th, 2023 before the members of jury :

President	Rachid	ZERGUI	MAA.	ENP, Alger
Supervisor	Cherif	LARBES	Prof.	ENP, Alger
Examiner	Mohammed O.	TAGHI	MAA.	ENP, Alger

ENP 2023
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
École nationale Polytechnique

Département électronique

Mémoire du Projet de Fin d'Études

Pour l'obtention du diplôme : Ingénieur d'Etat en Electronique

Contrôle vocal des UAV pour l'interaction homme-drone

Amina BOULKOUT

Sous la supervision de Dr. Cherif LARBES Prof. ENP, Alger

Présenté et défendu le 04 juillet 2023 devant les membres du jury :

Président	Rachid	ZERGUI	MAA.	ENP, Alger
Encadreur	Cherif	LARBES	Prof.	ENP, Alger
Examinateur	Mohammed O.	TAGHI	MAA.	ENP, Alger

ENP 2023

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

Dedication

“

TO my family,

TO my special friends,

Thank you

”

- Amina BOULKOUT

Acknowledgment

First i would like to thank Allah for granting me the determination to accomplish this work. I would like to express my deepest gratitude to my loving parents, and my brothers Amir and Aymen.

In addition, I would like to seize this moment to extend my heartfelt appreciation and profound gratitude to my project supervisor, Professor Cherif LARBES, an esteemed educator at the National Polytechnic School. His continuous guidance and steadfast support have been truly priceless. I'm sincerely thankful for his kindness, motivation, and unwavering availability.

I would also like to express my gratitude to the members of the jury, particularly to the President of the jury, Pr. Rachid ZERGUI, a respected Professor at the National Polytechnic School, and the Examiner, Mr. Mohamed Oussaid Taghi, a dedicated Teacher Researcher at the National Polytechnic School. Theire willingness to read and give constructive analysis of my work. Their commitment to my academic journey, as well as their valuable time, imparted knowledge, and unwavering support, deserve my utmost appreciation. I am truly indebted to them for their contributions throughout the three-year duration of our studies.

Lastly, i would like to express my deep appreciation for the invaluable contributions made by my teachers, professors, and fellow graduate electronic students, my best friend Manel and Mr. Issam SEDDIKI. I recognize and acknowledge each and every one of you with utmost gratitude, as you have played a crucial role in shaping my journey and propelling me to the point where i am today. Your guidance, knowledge, and support have been instrumental in my personal and academic growth.

ملخص

في السنوات الأخيرة ، زاد أهمية تفاعل الإنسان مع الطائرات بدون طيار في البحث العلمي بشكل كبير. عند التفاعل مع الطائرات بدون طيار ، يتحمل البشر مسؤوليات متنوعة ، تعتمد على تطبيق الطائرة بدون طيار ومستوى الذكاء الصناعي. تهدف هذه الدراسة إلى تنظيم حركة الطائرات الجوية بدون طيار باستخدام نظام الكشف عن الكلمات المفتاحية. لتحقيق هذا الهدف ، يتم تدريب شبكة عصبية عميقة (DNN) على فهم كلام المستخدم في بيئات صاخبة وخالية من الضجيج وتوليد أوامر التحكم المطلوبة وفقًا لذلك. يُظهر التنفيذ الأجهزة للنظام المطور دقة عالية في التعرف على الكلام وسهولة التحكم.

كلمات مفتاحية : التعرف على الكلام، SEGAN، DNN، طائرة بدون طيار، التحكم، تفاعل الإنسان مع الطائرة بدون طيار.

Résumé

Ces dernières années, l'importance de l'interaction entre l'homme et le drone dans la recherche scientifique a considérablement augmenté. Lorsqu'ils interagissent avec des drones, les humains assument diverses responsabilités qui dépendent de l'application et du niveau d'autonomie du drone. Cette étude vise à réguler les mouvements des véhicules aériens sans pilote (UAV) en utilisant un système de détection de mots-clés. Pour ce faire, un réseau neuronal profond (DNN) est entraîné à comprendre la parole de l'utilisateur dans des environnements bruyants et sans bruit, et à générer les commandes de contrôle souhaitées en conséquence. La mise en œuvre matérielle du système développé démontre à la fois une grande précision dans la reconnaissance de la parole et une facilité de contrôle.

Mots clés : Reconnaissance vocale, SEGAN, DNN , UAV , contrôle, interaction homme-drone.

Abstract

In recent years, the significance of human-drone interaction in scientific research has grown substantially. When engaging with drones, humans undertake various responsibilities, which are contingent upon the drone's application and level of autonomy. This study aims to regulate the movements of unmanned aerial vehicles (UAVs) by utilizing a keyword spotting system. To achieve this, a deep neural network (DNN) is trained to comprehend user speech in both noisy and noiseless environments and generate the desired control commands accordingly. The hardware implementation of the developed system demonstrates both high accuracy in speech recognition and ease of control.

Keywords : Speech recognition, SEGAN, DNN, UAV, Control, Human-Drone Interaction.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

General introduction **13**

1 Human drone interaction : state of art **15**
1.1 Introduction . . . 16
1.2 Definition of the human drone interaction . . . 16
1.3 User interfaces . . . 16
1.3.1 Graphical user interface . . . 16
1.3.2 Natural user interface . . . 17
1.4 Novel use cases of the human drone interaction . . . 19
1.4.1 Emergency . . . 19
1.4.2 Entertainment and art . . . 19
1.4.3 Sports . . . 20
1.5 Conclusion . . . 20

2 Keyword spotting in noisy and noiseless environment **22**
2.1 Introduction . . . 23
2.2 Signal Processing and Feature Extraction . . . 23
2.2.1 Spectrogram . . . 23
2.3 Deep learning models for speech recognition . . . 26
2.3.1 Convolutional Neural Networks . . . 29
2.3.2 Autoencoder . . . 33
2.3.3 Speech Enhancement Generative Adversarial Network (SEGAN) . . . 35
2.4 Conclusion . . . 36

3 Fundamentals of Quadrotors **37**
3.1 Introduction . . . 38
3.2 Classification of drones . . . 38
3.2.1 Classification by size . . . 38
3.2.2 Classification by propulsion mode . . . 39
3.3 The quadrotor . . . 41
3.4 Movements of the quadrotor . . . 42
3.5 Brushless DC Motors . . . 42

3.5.1 CONSTRUCTION AND OPERATING PRINCIPLE 43
3.5.2 Stator 43
3.5.3 Rotor 44
3.5.4 Hall Sensors 44
3.5.5 TORQUE/SPEED CHARACTERISTICS 45
3.5.6 Theory of Operation 46
3.5.7 Control Methods of the brushless dc motor 46
3.6 Conclusion 49

4 Controlling the quadrotor with keyword spotting model : implementation and results 50
4.1 Introduction 51
4.2 Keyword spotting 51
4.2.1 Software and Libraries 51
4.2.2 Hardware 52
4.2.3 Metrics Used 54
4.2.4 Problem statement 56
4.2.5 Proposed solutions 56
4.2.6 Solutions for the noiseless : the CNN model 60
4.2.7 Solutions for the noisy environment 65
4.2.8 The first solution for the noisy environment : Autoencoder with CNN model 02 73
4.2.9 The second solution for the noisy environment : SEGAN with CNN model 74
4.3 Quadrotor 75
4.3.1 Equipments 76
4.3.2 Software 78
4.3.3 Assembly 79
4.3.4 components configuration 80
4.3.5 Algorithm Description 81
4.4 controlling quadrotor with keywordspotting 82
4.5 Conclusion 85

Conclusion et perspectives 86

Bibliography 88

Liste des tableaux

4.1 Percentage of data in the CNN model for the clear data 59
4.2 Percentage of data in the CNN model for noisy environment model 02 :. . 60
4.3 Percentage of data in the binary classifier 60
4.4 Percentage of data in the Autoencoder model 60
4.5 The architecture of the CNN model 62
4.6 The parameters of the CNN model 63
4.7 The reults of Precision, Recall, F1-score, AUC-ROC score of the CNN model 01 65
4.8 The architecture of the binary classifier. 67
4.9 The parameters used in the binary classifier. 67
4.10 The architecture of the CNN model 02. 69
4.11 The architecture of the autoencoder 71
4.12 The architecture of the discriminator 73

Table des figures

1.1 High-Level Description of Hand Gesture NUI[5] 18
1.2 Flying drones with brains 19
1.3 Drones to the rescue WCape emergency services. 19
1.4 UAV took a picture of a landscape. 20
1.5 Drone Chi[9]. 20

2.1 Short time Fourier transform 24
2.2 example of an input speech signal. 24
2.3 example of Frame of the input speech signal. 25
2.4 Hamming window 25
2.5 Spectogram of the signal. 26
2.6 Architecture of a perception[11]. 28
2.7 Neural network with one hidden layer[11]. 28
2.8 Standard architecture of a CNN. 29
2.9 Convolutional layer[11] 30
2.10 Operation of pooling[11] 30
2.11 Operation of flattening[11] 31
2.12 neural network fully connected [11] 31
2.13 ReLu activation function[11] 32
2.14 Sigmoid activation function[11] 32
2.15 Autoencoder architecture[18]. 34
2.16 Illustration of the architecture of speech enhancement GAN (SEGAN)[19]. 36

3.1 Micro-Drone. 38
3.2 Mini drone. 39
3.3 MALE drone. 39
3.4 HALE drone. 39
3.5 Fixed-wing drone. 40
3.6 Flapping-wing drone. 40
3.7 VTOL rotating wings drone. 40
3.8 Multirotor. 41
3.9 the structure model of a quadrotor[20]. 41
3.10 :The movements of a quadrotor. 42
3.11 STATOR OF A BLDC MOTOR[21]. 43
3.13 ROTOR MAGNET CROSS SECTIONS[21]. 44
3.14 BLDC MOTOR TRANSVERSE SECTION[21]. 45
3.15 TORQUE/SPEED CHARACTERISTICS[21]. 45
3.16 PWM signal with a varying duty cycle. 47

3.17 Electronically commutated BLDC motor drive [22]. 47
3.18 Hall sensor signal, back-EMF, output torque and phase current [22]. 48

4.1 Diagramm of the methodology 57
4.2 Number of wav files in each folder of the dataset 58
4.3 The architecture of the CNN model. 61
4.5 Confusion matrix 64
4.6 The AUC ROC score of the CNN model 01. 65
4.7 The architecture of thebinary classifier model. 66
4.9 The confusion matrix of the binary classifier. 68
4.10 The architecture of the CNN model 02. 69
4.12 The effect of noise on CNN 71
4.13 The effect of noise on CNN when using the autoencoder. 74
4.14 The effect of noise on CNN when using SEGAN 75
4.15 The brushless dc motor. 76
4.16 The electronic speed controller. 77
4.17 The propellers. 78

Liste of abreviations

IA *Intelligence artificielle.*

DL *Deep learning.*

CNN *Convolutional neural network.*

DNN *Deep neural network.*

RVB *Rouge Vert Bleu.*

FCN *Fully connected network.*

MLP *Multi Layer Perceptron.*

ReLu *Fonction Unité Linéaire Rectifiée.*

SGD *Stochastic gradient descent.*

Adam *Adaptive Moment Estimation.*

GPU *Graphics Processing Unit.*

TPR *True positive rate.*

FPR *False positive rate.*

TNR *True negative rate.*

FNR *False negative rate.*

BLDC *Brushless dc.*

PWM *Pulse width modulation.*

GAN *Generative Adversarial Network.*

SEGAN *Speech enhancement Generative Adversarial Network.*

DC *Direct current.*

EMF *Electromotive force.*

HDI *Human-Drone Interaction.*

NUI *Natural User Interface.*

GUI *graphical user interface.*

BCI *Brain-computer interface.*

ASR *Automatic speech recognition.*

FFT *Fast Fourier Transform.*

STFT *Short-Time Fourier Transform.*

MFCC *Mel-frequency cepstral coefficients.*

FOC *Field-Oriented Control.*

DTC *Direct Torque Control.*

MPC *Model Predictive Control.*

General introduction

Drones, also known as UAVs (Unmanned Aerial Vehicles), have revolutionized the aviation industry by enabling autonomous flight missions without human pilots onboard. With the rise of personal drones, there is a growing need to design effective ways to interact with these flying robots.
Thus, in order to comprehend this new developing field, we aimed to discuss not only what social drone companions can accomplish, but also their design and prespective domain areas in which they may be employed[1]..

As the popularity of drones continues to increase, even among individuals with limited knowledge of the subject, there is a need to develop user-friendly interfaces that facilitate intuitive control and operation. Natural User Interfaces (NUIs), such as body gestures and voice commands, have been tested and found to enhance human drone interaction.

The main focus of this work is to respond to the following question : How can we control a UAV with our voice instead of joysticks ? Or can a drone understand English commands.

For this we were interested in designing a voice-controlled quadrotor. This work is divided into four chapters, each addressing a specific aspect.

The first chapter introduces the topic of Human-Drone Interaction, highlighting the significance of drones in our lives and the importance of effective engagement with them.

The second chapter delves into fundamental concepts of voice recognition and the deep neural network, which plays a crucial role in speech feature extraction and recognition. This provides the necessary background for understanding the technical aspects of voice-controlled drones.

The third chapter delves into the theoretical aspects of quadrotors and provides an in-depth exploration of their components and functioning. By understanding the underlying principles of quadrotor systems, we can gain valuable insights into their capabilities and limitations.

In the fourth chapter, the implementation aspect is explored, discussing the application of the deep learning speech recognition models, the quadrotor and the quadrotor controlled with the voice recognition.

The work concludes with a summary of the findings and suggestions for future work. It emphasizes the need to continue exploring and improving the proposed subject to ensure its continuity and performance in the field of human-drone interaction. By addressing the challenges and opportunities in the design and implementation of voice-controlled drones, this work contributes to the advancement of human-drone interaction and opens up new possibilities for intuitive and user-friendly drone operation.

Chapitre 1

Human drone interaction : state of art

1.1 Introduction

The area of human-robot interaction (HRI) has emerged as a result of robot research focusing on understanding and creating interactions with human users during the last decade[1].
With recent technological advances, UAVs have appeared as a new type of robot that has captivated the interest of HRI research, leading to a whole new field of human-drone interaction (HDI) research[2].
The term "drone" simply refers to an unmanned aerial vehicle. Drone technology emerged following World War I. Despite the fact that they were designed for military uses. Drones are currently being employed in a wide range of applications, shifting from the military to the civilian sphere[3].
Drone use has risen at an exponential rate in recent years. This rapid growth is both exciting and frightening. On the one hand, drones open up new opportunities, with applications ranging from entertainment to delivery, assistance to people with special needs, sports, agriculture, and even rescue.[1].
On the other hand, there are several risks to the use of drones in our environment[4]. Therefore, it is important to study the field of HDI (Human-Drone-Interaction) to understand how the interaction between humans and drones can be extended to more areas of use[1].
In this chapter we delve into the field of Human-Drone Interaction (HDI) and explore various aspects related to the interaction between humans and drones, then we will focuse on the user interfaces especially the NUI, after that we will explore the novel use cases of the human drone interaction.

1.2 Definition of the human drone interaction

Human-drone interaction is a diverse field of research. It can be defined as a field of study focused on the understanding and evaluation of the interaction distance and the development of new use cases[1].

1.3 User interfaces

The user interface (UI) is the point at which a computer, website, or application interacts with humans. The purpose of a good UI is to make the user's experience simple and straightforward, requiring the least amount of work from the user to get the maximum desired results. The most important formats are GUI and NUI[5].

1.3.1 Graphical user interface

The graphical interface, which remains in use today, facilitates interaction through a familiar and predictable approach known as WIMP (Windows, Icons, Menu, and Pointer). The graphical interface provides interaction with the vehicle, as well as observation of

states and dynamics, and also the presentation of graphical views and pictures to assist the user in understanding the vehicle's internal behaviour[5].

1.3.2 Natural user interface

Natural User Interfaces (NUIs) are the next level in user interface evolution, compared to graphical user interfaces (GUIs)[6].
These novel approaches allow users to interact with drones via gesture, speech, touch, and even brain-computer interfaces (BCIs) like electroencephalography (EEG)[4].
Many applications may now be found in our daily lives. Voice assistants, Alexa, and Siri, for example, respond to a voice-activated NUI[6]. Natural user interfaces major goal is to provide an easy method of control, which is described as an interface that operates as expected by the user. Non-expert users can interact with natural user interfaces[4].

Gesture

The gestures used in the system of [7] can be classified into two categories : posture and dynamic gesture. Postures are predetermined finger configurations and involve hand movement, while dynamic gestures require hand motion. To interpret these gestures, the NUI must measure the configurations of the hand and arm, either statically or dynamically, using some kind of device. Early attempts at hand/arm based NUIs used glove-based devices with sensors that directly measured spatial position and joint angles.However, these devices were cumbersome and unwieldy. Nowadays, most research in this field employs depth cameras, such as the Kinect, to capture depth-based hand gestures. For instance, in one study, researchers developed a gesture recognition system based on depth imagery and created a depth-based hand gesture database for controlling drones. Another group of researchers successfully tested a tour-guide robot that recognized and responded to user hand gesture commands or augmented reality virtual button selection.However, these devices have limitations when it comes to accurately recognizing depth-based hand gestures. They suffer from reduced resolution, high noise, and missing data, which make it challenging to extract reliable information about hand/finger poses.
The Leap Motion controller :
It is a new design tool that is a sensor designed to track hand and finger motions in a small working space, it interprets hand movement and poses in a three dimensional space .The sensor works by projecting infrared light upward and detecting reflections using monochromatic infrared cameras. It's field-of-view (FOV) extends from 25mm to 600mm with a 150°spread from the device.The Leap Motion calculates the orientation and position of the user's hand in relation to its own axes. So, as it is shown in Figure 1.1, pitch, roll and yaw represent rotations around the x, z and y axes respectively. In contrast, the drones axes follow a North-East-Down (NED) configuration, which means pitch, roll and yaw are rotations around the y, x and z axes respectively. One way of using this information to interact with the drone is to send pitch, roll, yaw rate and thrust commands to the drone by directly mapping the orientations and position of the hand to the drones coordinate system.Another possibility is to send higher level commands such as velocity or position [5].

FIG. 1.1 : High-Level Description of Hand Gesture NUI[5]

Speech

Speech is used as a method of interaction by 38% of American users and 58% of Chinese users, according to studies on natural user interfaces. Speech control is considered to be simpler than other approaches, resulting in shorter training times. However, voice recognition, like gesture control, can add delays to the system, limiting its uses. Furthermore, operating drones with speech has some unique problems, since the propellers produce a loud noise that can decrease voice recognition accuracy. Another problem is that if voice recognition is conducted on board, users are confined to collocated interaction because the drone must be close to the user to receive spoken instructions. This issue does not affect systems where a ground-control station is used to decode voice commands and control the drone[1].

Brain-Computer Interaction (BCI)

Brain-computer interface devices have broad potential as assistive technologies and novelty control methods. Researchers have been investigating the use of brain-computer interfaces (BCIs) to control unmanned aircraft (fixed wing) since 2010, and the first brain-controlled multirotor project was reported in 2013. In 2016, the University of Florida hosted the first brain- controlled drone race, which was followed by a race at the University of Alabama. The pilot needs to use a BCI headset. The most popular of these are Electroencephalography (EEG) headsets, which operate drones via brain signals. These devices use machine learning algorithms to analyse the brain's electrical activity on the human scalp, which is then used to operate physical systems using brain- waves. As a result, interactions with BCIs are currently restricted compared to conventional control interfaces, and additional research is needed to improve the fidelity and reliability of these systems before they are deployed in users homes. Hands-free interaction and accessibility for disabled people will be possible if BCI reliability and accuracy reach levels similar to traditional control modalities[1].

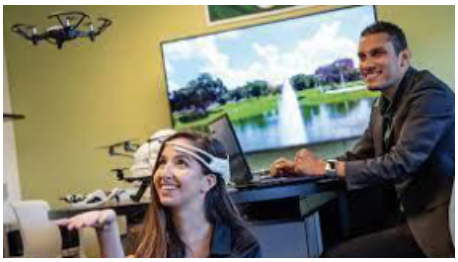


FIG. 1.2 : Flying drones with brains

Multimodal

it combines two or more interaction modalities , for example if we want to control a drone using it we can control the take off and landing of the drone with our voice and the movements of this drone with gesture.

1.4 Novel use cases of the human drone interaction

1.4.1 Emergency

Drones can be utilized as a single unit or in groups in various scenarios, including natural disasters such as floods, earthquakes, and fires like it is shown in figure 1.3. These emergency situations often involve complex environments, such as cities or mountains, where both drones and humans operate. HDI plays a crucial role in emergency operations, benefiting both first responders and bystanders and potential victims. First responders, who are trained professionals, use drones to gain situational awareness and assist in their mission. On the other hand, bystanders, who may or may not require assistance, may not be familiar with drone technology. The emergency domain presents a multitude of challenges for HDI due to the constantly changing and complex nature of the environment, involvement of various stakeholders, and life-threatening situations[8].

FIG. 1.3 : Drones to the rescue WCape emergency services.

1.4.2 Entertainment and art

Drones are utilized in various roles for entertainment purposes, such as playing games with individuals or in drone racing. They can move along with players or they can even become active players, like in hide and seek. In addition, drones are also used in the arts, particularly by graffiti artists who co-create with drones to gain access to hard-to-reach

places. Drones are also utilized for painting and landscape art as it is depicted in figure 1.4. Furthermore, drone swarms are utilized for light shows to create large public displays for entertainment purposes.Real-time or pre-programmed drone displays enable complex information visualization for entertainment, giving artists the ability to create interactive drone displays[8].

Fig. 1.4 : UAV took a picture of a landscape.

1.4.3 Sports

Drones have been integrated into various sports activities, such as jogging, Tai Chi as shown in figure 1.5, boxing, cycling and climbing. They serve as sports coaches or assistants and have an advantage over ground-based robots in that they are not constrained by the landscape. Therefore, they can be utilized in both mountain and water sports, such as skiing and rowing.Additionally, when jogging in remote areas, drones can be used to scan for potentially dangerous animals and alert the jogger if necessary[8].

Fig. 1.5 : Drone Chi[9].

1.5 Conclusion

In summary, this chapter delved into the realm of human-drone interaction (HDI) and explored the different user interfaces employed in this field. We examined graphical interfaces and natural user interfaces (NUIs) such as gesture-based, speech-based, and multimodal interfaces. These interfaces enable intuitive and seamless communication between humans and drones.

Furthermore, we discussed novel use cases of HDI, including emergency scenarios, entertainment, and art. Drones have proven to be valuable assets in emergency situations, facilitating quick communication and response. In the realm of entertainment and art, drones have pushed boundaries and offered captivating aerial performances and immersive artistic experiences.

Overall, the chapter emphasized the significance of user interfaces in enabling effective HDI. By understanding the capabilities and limitations of graphical and NUI interfaces, we can design user-friendly systems that enhance the interaction between humans and drones. The emerging use cases demonstrated the versatility of drones and their potential for innovation in various domains.

Chapitre 2

Keyword spotting in noisy and noiseless environment

2.1 Introduction

Speech recognition is a method that analyses sounds captured by a microphone and transcribes them into a series of words that machines can understand. Automatic speech recognition has progressed rapidly since its start in the 1950s, especially with the help of phoneticians, linguists, mathematicians, and engineers, who have defined the acoustic and linguistic knowledge necessary to fully understand a human's speech. However, the result isn't great and is dependent on a number of factors. Favourable conditions for speech recognition involve native speech, belonging to a single speaker with proper diction (not presenting a voice pathology), recorded in a quiet and noiseless environment, and based on a common vocabulary (known words by the system)[1]. In this chapter, we extensively examine the key elements and methodologies of signal processing and feature Extraction. Moreover, we delve into the remarkable advancements in deep learning models. We highlight the evolution of these models over time. The chapter also encompasses a detailed exploration of various deep learning models, such as Convolutional Neural Networks (CNN), Perceptron, Multilayer Perceptron (MLP), Autoencoder, and the Speech Enhancement Generative Adversarial Network (SEGAN), and finally we'll talk about the models that we've used in our project.

2.2 Signal Processing and Feature Extraction

Signal processing encompasses a range of techniques applied to audio signals .It involves tasks such as noise and channel distortion removal, as well as the transformation of the signal from the time-domain to the frequency-domain. By applying various filters and algorithms, signal processing aims to improve the intelligibility and clarity of the speech within the audio signal.
Feature extraction plays a crucial role in the speech recognition system as it involves identifying and extracting relevant characteristics from the processed audio signal. These extracted features serve as inputs to subsequent stages of the system, enabling accurate transcription of speech. One widely used technique for feature extraction in speech recognition is the spectrogram, Apart from the spectrogram, various other feature extraction techniques can be employed in speech recognition. Examples include Mel-frequency cepstral coefficients (MFCCs) and different filter banks. Each technique focuses on extracting specific aspects of the audio signal, allowing the model to capture different acoustic properties and variations.

2.2.1 Spectrogram

The spectrogram which is derived from the Short-Time Fourier Transform (STFT) , it's actually the magnitude of the STFT, provides valuable insights into the temporal and frequency characteristics of the audio signal.
It allows us to identify important acoustic features, by visualizing the distribution of frequency components over time. Typically, the spectrogram is presented as a two-dimensional plot with time on the x-axis, frequency on the y-axis, and color intensity or shading indi-

cating the magnitude or power of the corresponding frequency components.
the Short-Time Fourier Transform (STFT) :
The STFT breaks down the audio signal into short overlapping segments and applies a Fourier transform to each segment, revealing its frequency content, as shown in figure 2.1

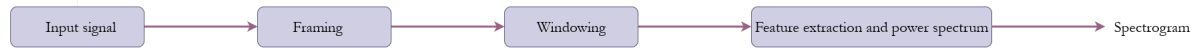


Fig. 2.1 : Short time Fourier transform .

- **Input Signal** : The input voice signal is an analog signal so the instantaneous voltage of the signal changes continuously with the pressure of the sound waves, and it is typically saved in (.wav) format for further processing and feature extraction, here's an example shown in figure 2.2

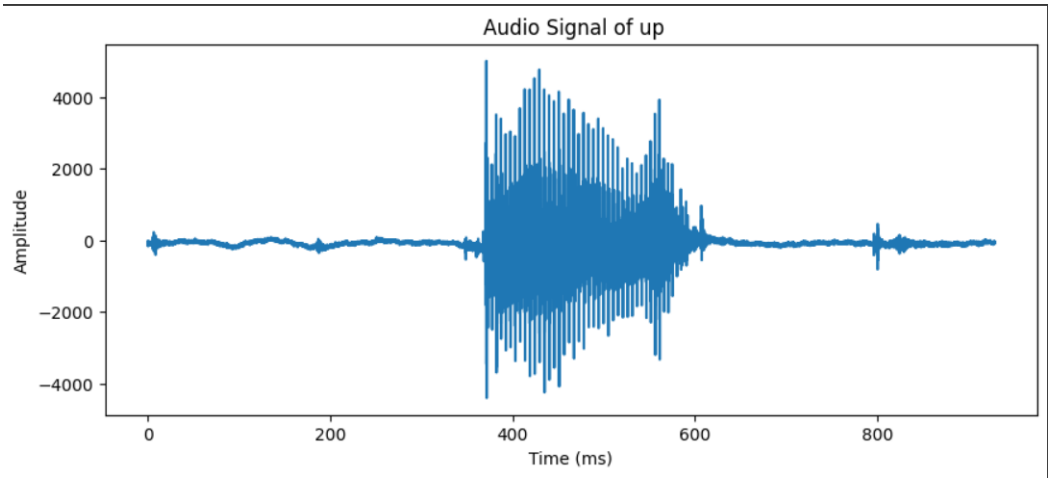


Fig. 2.2 : example of an input speech signal.

- **Framing** : To simplify the analysis of the audio signal, we assume that it doesn't vary significantly over short time scales. Therefore, we divide the signal into frames of 20-40 milliseconds. Each frame captures a snapshot of the signal's characteristics within a specific time interval. It is important to choose an appropriate frame length since very short frames may not provide enough samples for reliable spectral estimation, while longer frames may encompass significant changes in the signal

throughout their duration. In practice, the number of frames is often set to 256, a power of 2, which facilitates Fast Fourier Transform (FFT) calculations.The figure 2.3 shows example of frames of a signal.

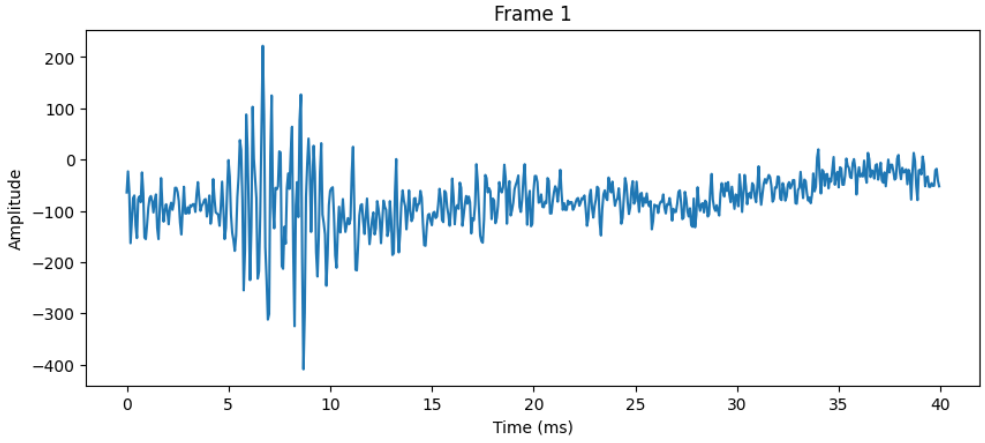


FIG. 2.3 : example of Frame of the input speech signal.

- **Windowing** : Following the framing step, each individual frame undergoes windowing. The purpose of windowing is to reduce signal discontinuities at the beginning and end of all the frames.
  A window function, denoted as W(n), is applied to each sample within the frame. The length of the frame is represented by the letter N.
  The windowed signal, Y(n), is obtained by multiplying the original signal, x(n), with the window function in the equation 2.1. In this context, the Hamming window is commonly used due to its favorable side-lobe characteristics, as depicted in Figure 2.4.

$$\mathbf{Y(n) = x(n)w(n), \quad 0 < n < N - 1} \tag{2.1}$$

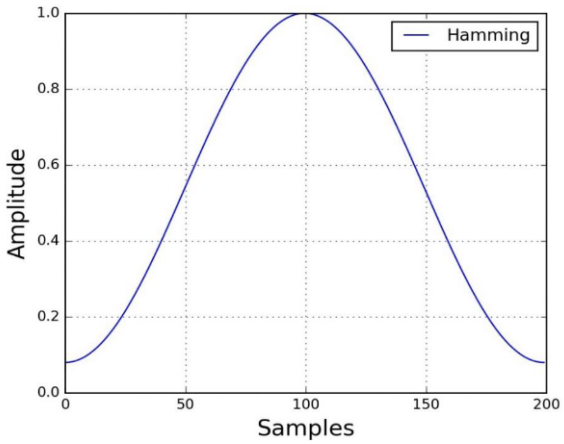


FIG. 2.4 : Hamming window

- **Fourier Transform and Power Spectrum** : Once the signal is windowed, each frame is subjected to Fast Fourier Transform (FFT). This process transforms the

time-domain representation of each frame into the frequency domain. The power spectrum, denoted as P, is computed using equation 2.2, where x_i represents, the ith frame.

$$\boldsymbol{P} = \frac{|FFT(xi)|^2}{N} \tag{2.2}$$

- **Spectrogram** : the spectrogram is obtained when applying the Mel scale filter bank to the power spectrum of the signal. the spectrogram captures the dynamic changes in the signal's frequency content over time. It serves as a valuable input to our Deep Neural Network for training and testing the speech recognition model.
  The figure 2.5 shows an example of a spectrogram.

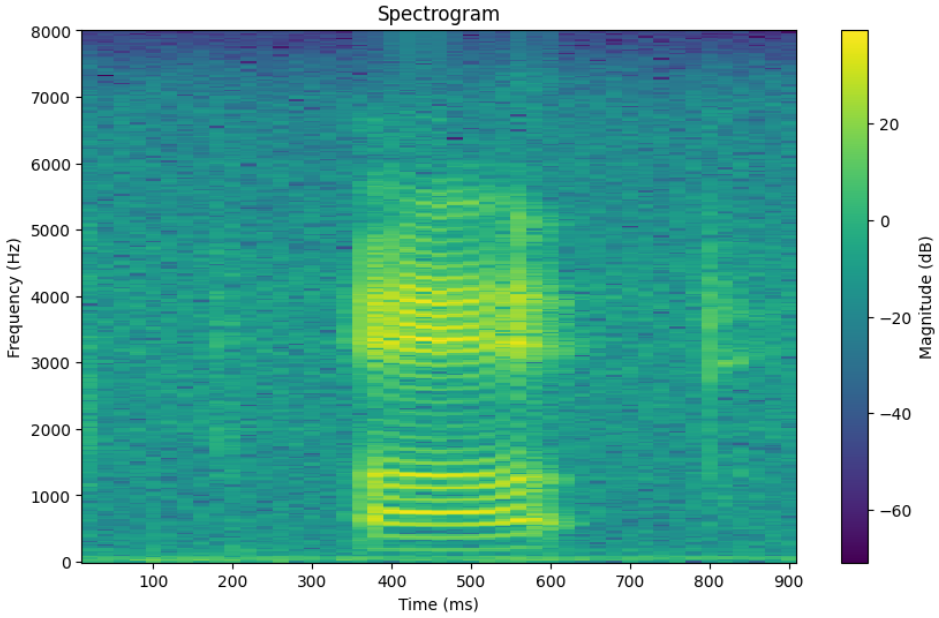


FIG. 2.5 : Spectogram of the signal.

2.3 Deep learning models for speech recognition

Convolutional neural network through the years

Over the past years, significant advancements have been made in artificial intelligence (AI), which lies at the intersection of computer science, mathematics, and neurobiology. The goal of AI is to enable machines to imitate or simulate human brain behavior, tackling complex problems like classification, recognition, and detection. The origins of artificial neural networks can be traced back to 1943 when Warren McCulloch, a neurophysiologist, and Walter Pitts, a mathematician, published a paper [10] titled "A logical calculus of the ideas immanent in nervous activity." In this paper, they introduced the first artificial neuron model using electrical circuits inspired by biological neurons[11].

In 1949, Donald Hebb expanded on the concept of neurons in his book [12] "The Organization of Behavior," explaining how neural pathways strengthen with use and providing valuable insights into human learning. In the 1950s, as computers became more advanced, Nathaniel Rochester from IBM research laboratories led the first attempt to simulate an artificial neural network, documented in multiple publications [11].
In 1957, building upon previous work, Frank Rosenblatt developed the perceptron model [13], which represents the earliest neural network based on a single layer of neurons. But, the perceptron had limitations, which is that, it is a binary classifier ,it could only separate two classes, and the data had to be linearly separable.During that time, computers lacked sufficient processing power to effectively handle large neural networks, resulting in stagnant research. However, in the mid-1980s, as computers gained greater processing power, it was realized that neural networks with two or more layers outperformed single-layer perceptrons. This led to the emergence of the multilayer perceptron (MLP).In the late 1980s, the advancement of neural networks was hindered as the lack of conceptual tools made it challenging to analyze and predict their behavior. However, the introduction of the backpropagation learning algorithm [14] and the contributions by Hopfield [15] marked a significant breakthrough. These developments revitalized neural networks, leading to the creation of various models for prediction, classification, detection, and other applications. However, these models often require a preprocessing step to effectively extract features and represent the data.
In recent years, the growing computational capabilities of infrastructure have enabled computers to process large datasets using deep learning techniques.Deep learning involves the integration of additional layers in artificial neural networks (ANNs) that are specifically designed for feature generation, in addition to the traditional layers. The first model , called LeNet, was introduced by LeCun et al. in 1998 [16] for handwritten digit recognition.Following that, numerous models designed for specific applications have been developed. Additionally, several models more suitable for diverse applications have been proposed. In the upcoming sections, we will introduce key definitions necessary for comprehending the CNN (Convolutional Neural Network) utilized in our work.

Perceptron

The perceptron is a fundamental neural network architecture, it consists of a single formal neuron. It has been widely recognized as the earliest learning algorithm used for classifying two classes that can be separated by a straight line. A perceptron comprises one or more inputs, a processing unit and an output.In Figure 2.6, we can observe that given an input vector $\mathrm{A} = (\mathrm{a}_0, \mathrm{a}_1, \ldots, \mathrm{a}_K) \in \mathbb{R}^{K+1}$ this input vector is weighted by the corresponding elements of the weight vector Ω, represented as $\Omega = (w_0, w_1, \ldots, w_K) \in \mathbb{R}^{K+1}$ The resultant sum is then passed through an activation function denoted as Z(.), which yields the output O[11].

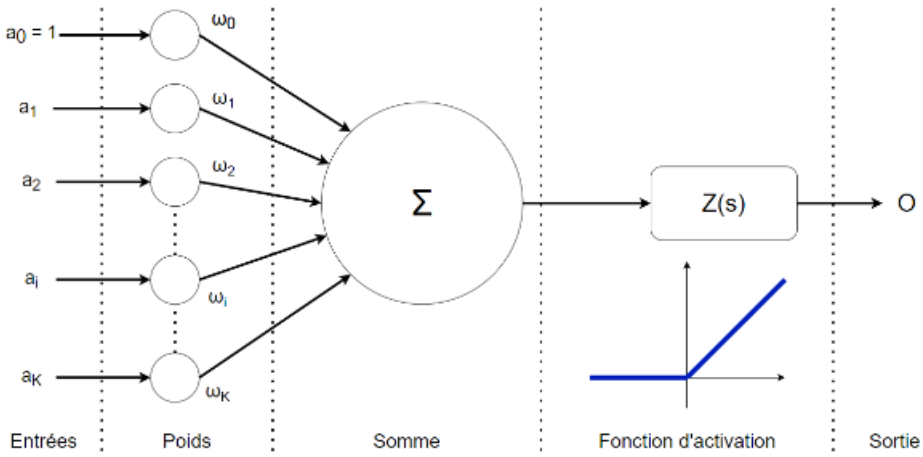


FIG. 2.6 : Architecture of a perception[11].

Multilayer Perceptron (MLP)

The multilayer perceptron (MLP) is an extended version of a perceptron, which includes an input layer, multiple hidden layers, and an output layer. In this architecture, each neuron in a layer is connected to every neuron in the adjacent layer, forming a fully connected network with a minimum of three layers (Figure 2.7). The size of the input layer determines the dimensionality of the input feature vector, while the number of neurons in the output layer corresponds to the number of classes in the classification task.

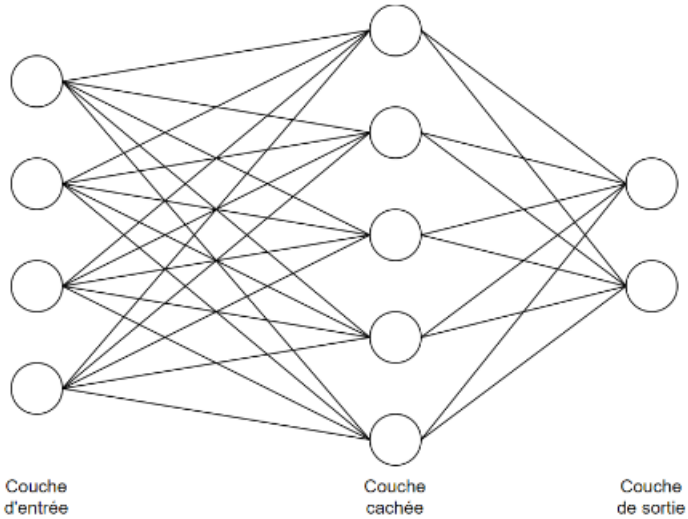


FIG. 2.7 : Neural network with one hidden layer[11].

The MLP employs backpropagation, a technique that utilizes the backpropagation of the error gradient during the learning process, to determine the synaptic weights w_i. The learning rate and momentum are two parameters employed to ensure the stability of the convergence process. They help reduce weight variations between consecutive itera-

tions, facilitating smooth and efficient learning of the MLP model[11].

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs or ConvNets) are an extension of MLPs that effectively overcome some of their limitations. They are specifically designed to autonomously extract features from input data. They accomplish an automatic extraction of features without the need for manual intervention this by assigning adjustable weights and biases that enable differentiation between various aspects of the input image.Technically, each input image undergoes two blocks of layers. The first block consists of convolutional layers, which is made for generating features, and the second block is the classification block, comprising fully connected layers that share a similar architecture to MLPs. Figure 2.8 provides a visual representation of the layer arrangement within a CNN[11].

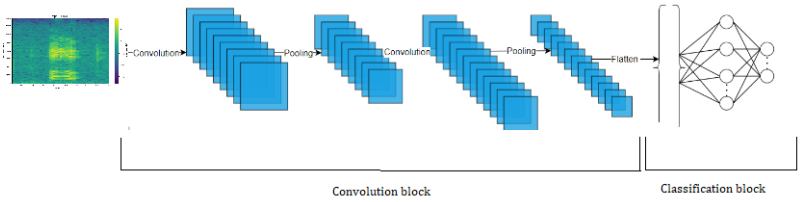


Fig. 2.8 : Standard architecture of a CNN.

Feature Extraction Block

The initial block in a CNN serves a distinct purpose as a feature extractor. Its first layer applies convolutional operations to the input image using multiple kernels, producing normalized or resized features through an activation function.This process can be iterated several times. The resulting feature maps undergo further filtering using new kernels, generating additional features that are by next normalized and filtered again, and so forth. Finally, the values of the last set of features are combined and flattened into a column vector [17].This vector represents the output of the first block, which serves as the input to the second block. Specifically, this part of the network comprises two primary types of layers : convolutional layers and pooling layers. Convolutional layers employ filters to extract features from the input data, while pooling layers downsample the feature maps, reducing their spatial dimensions.

- **Convolutional Layers** :
  A 2-dimensional convolutional layer are responsible for extracting features from an input image while preserving the relationships between pixels. They achieve this by learning image characteristics using square-shaped filters.
  These layers take two inputs : an input image matrix with dimensions h × w × d (where h represents the height, w is the width, and d is the number of channels),

and a filter or kernel with dimensions fh × fw × d (where fh is the filter height, fw is the filter width, and d is the number of channels).Typically, RGB color images are represented with three channels as the input to convolutional layers. However, for binary or grayscale images, a single channel is used.
The output matrix is computed by performing a sequential dot product operation between a portion of the input matrix and the filter. The filter is applied by sliding it over the image from left to right and top to bottom, with strides which is the number of steps. This process is repeated to perform the same calculation across the entire image, as illustrated in Figure 2.9

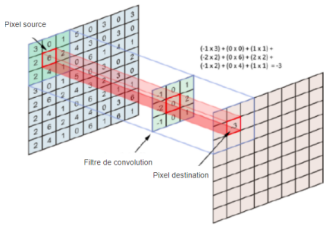


FIG. 2.9 : Convolutional layer[11]

- **Pooling layers** :
  Pooling layers are responsible for downsampling the image by focusing solely on its spatial dimensions, namely width and height. The input image is divided into non-overlapping squares, and this layer is often inserted between successive convolutional layers in a CNN architecture. Its purpose is to reduce the number of parameters and computational power required while preserving the most important features.
  During pooling,in practise the image is divided into adjacent square cells, with a specified stride to avoid excessive information loss.The number of output images retains the same as the input, but each image has a reduced number of pixels [17].Two main types of pooling are commonly used : max pooling and average pooling.Max pooling selects the maximum value within each pooling region, while average pooling calculates the average of all values in the region. Max pooling is typically preferred due to its superior performance compared to average pooling. Figure 2.10 provides visual examples of max pooling and average pooling operations.

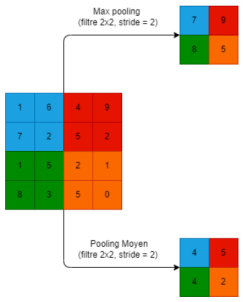


FIG. 2.10 : Operation of pooling[11]

- **Flatten layer** :
  also referred to as the flattening layer, serves as the final step within the feature

extraction block.Its primary function is to transform the structured matrices into a one-dimensional vector by concatenating their rows. This resulting vector can then be seamlessly connected to the subsequent prediction block (Figure 2.11).

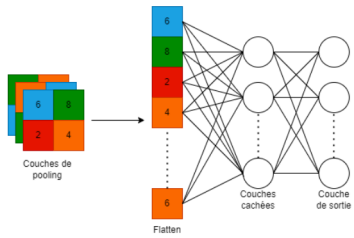


FIG. 2.11 : Operation of flattening[11]

The Classification layers block

It consists of fully connected layers (FC), which are similar to the multilayer perceptron (MLP) and depicted in Figure 2.12. The primary purpose of fully connected (FC) layers is to use the flattened input vector to learn the optimal parameters for image classification.To enhance feature learning and decision-making, multiple FC layers are typically stacked consecutively. The last FC layer produces the CNN's output, which is a vector with a size equal to the number of classes. The neurons in this layer employ the softmax activation function to generate probabilities that indicate the likelihood of an input image belonging to various classes.

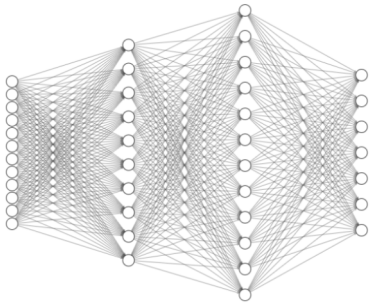


FIG. 2.12 : neural network fully connected [11]

Parameters of CNN

Besides the learning rate, momentum, number of epochs, number of hidden layers, and neurons, which are patameters used in the MLP , they are also applied in the FC layer block, the convolutional block of a CNN also utilizes several parameters. This section provides a brief overview of these parameters.

- **Filters** :
  Convolution involves sliding a series of filters or convolution kernels over the input image. The choice of filters, their size, and their number (also known as depth) is an experimental task. The depth of the convolutional layer refers to the number of

kernels used, which is the reason behind the term Deep Learning If we use a number q of filters, the image will be convolved q times. This process creates q filtered matrices called feature maps.

- **Activation Functions** :
  Activation functions play a crucial role in determining the output of a neural network. Based on their relevance to the model's prediction, activation functions determine whether a neuron should be activated or not. Here are some commonly used activation functions :

  ■ **Rectified Linear Unit (ReLU)** :
  ReLU sets negative values to 0 and leaves positive values unchanged. It enhances the network's non-linear behavior, promoting better feature representation. The ReLU function is defined as :

$$G(E) = \max(0, E) = \begin{cases} E & \text{if} \quad E \geq 0 \\ 0 & \text{else} \end{cases} \tag{2.3}$$

  and has the following graph :

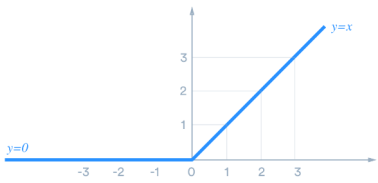


FIG. 2.13 : ReLu activation function[11]

  ■ **Sigmoid function** :
  The sigmoid function represents the cumulative distribution function of the logistic distribution. It is differentiable and commonly used in neural networks.the derivative of the inverse function of sigmoid has a simple form, aiding in algorithm performance. It is defined as :

$$\alpha(x) = \frac{1}{1 + e^{-x}} \quad \forall x \in \mathbb{R} \tag{2.4}$$

  and has the following graph :

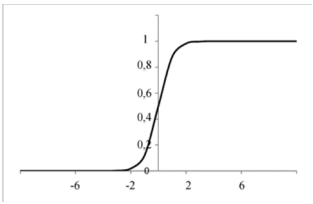


FIG. 2.14 : Sigmoid activation function[11]

■ **Softmax function** :
In most of the time ,it is utilized in the final layer of the network, it is a normalization function, providing an approximation of the probability that a class is correct.[11]The probability value for each class is calculated using the softmax equation :

$$G\left(e_j\right) = \frac{e^{e_j}}{\sum_i e^{e_i}} \tag{2.5}$$

where e_j represents the j-th element of the input vector. The class with the highest probability is selected as the output of the neural network.

- **Stride** :
The stride determines the number of pixels by which the filters are shifted over the input matrix. A stride of 1, moves the filters one pixel at a time, while a stride of 2 moves them two pixels at a time, and so on.

- **Zero-padding** :
Following each convolutional layer, the resulting image size is smaller than the input due to the fact that the filter cannot process the border pixels. When multiple convolutional layers are used, the output image becomes significantly smaller compared to the input. To preserve the original image size, zero-padding is employed by adding zeros along the image borders,considering the filter size. This ensures that, regardless of the number of convolutional layers, the output image maintains the same dimensions as the input.

- **Dropout** :
Dropout is a technique applied during the training phase to mitigate overfitting issues and reduce testing error associated with fully connected layers. It involves randomly deactivating neurons to enhance the network's responsiveness and expedite learning.

2.3.2 Autoencoder

Autoencoder Definition

An autoencoder is a neural network architecture that is designed to reconstruct input data by encoding it into a lower-dimensional representation and then decoding it back to its original form. One specific type of autoencoder is the autoencoder based on convolutional layers, the encoder utilizes convolutional layers, as described in the previous section ,to extract features from the input image and encode it into a lower-dimensional representation. On the other hand, the decoder employs transposed convolutional layers (also known as deconvolutional layers, which reverses the operation of a convolutional layer by effectively increasing the spatial dimensions of the input data. This is achieved by sliding a filter over the input and inserting zeros or padding values between the original values, to decode the representation and reconstruct the original image.

Autoencoder Architecture

The architecture of an autoencoder consists of two main components : an encoder and a decoder.
Here is the general architecture of an autoencoder as depicted in figure 2.15 :

- **Encoder** :
  - Input layer : Represents the input data.
  - Hidden layers : Comprise several layers, typically including convolutional , that gradually reduce the dimensionality of the input.
  - Bottleneck layer : Represents the encoded representation, which has the lowest dimensionality. It captures the most important features of the input data.

- **Decoder** :
  - Hidden layers : Comprise several layers, deconvolutional layers, usually mirroring the structure of the encoder's hidden layers in reverse order. These layers gradually increase the dimensionality of the encoded representation.
  - Output layer : Represents the reconstructed data, which aims to match the original input.

The encoder and decoder are typically symmetric, meaning that the number of layers and their configurations are mirrored. However, the number of nodes in the bottleneck layer is usually smaller than the number of nodes in the hidden layers of the encoder and decoder. The architecture of an autoencoder can be customized based on the specific requirements of the problem at hand. It can include additional layers, regularization techniques, or other modifications to enhance performance or address specific data characteristics.

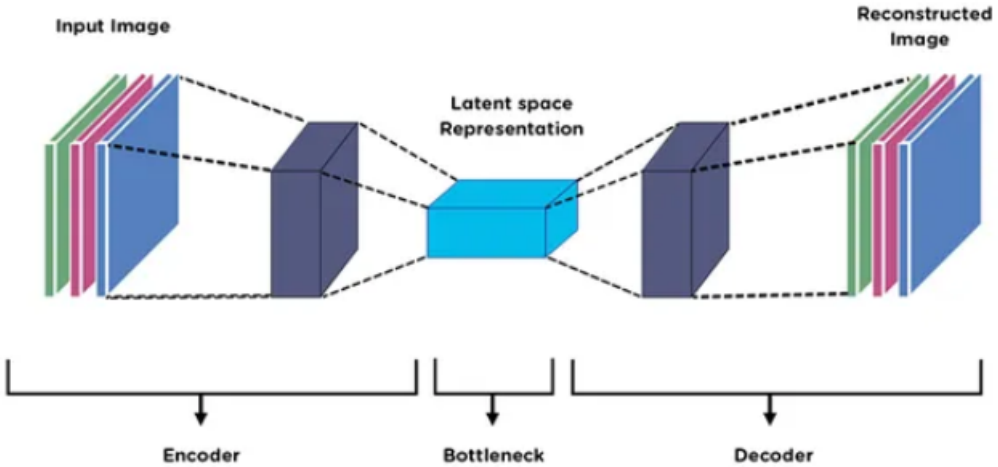


Fig. 2.15 : Autoencoder architecture[18].

2.3.3 Speech Enhancement Generative Adversarial Network (SEGAN)

The Speech Enhancement Generative Adversarial Network, is a specialized type of Generative Adversarial Network (GAN) used for speech enhancement. Its primary objective is to enhance the quality of speech signals that have been degraded by noise or other distortions.
The architecture of SEGAN as shown in figure 2.16 is based on a modified GAN design. It consists of two key components : the generator network and the discriminator network. The generator network takes a noisy speech signal as input and aims to generate a clean speech signal as output. On the other hand, the discriminator network, classifies the input speech as either clean or distorted.
The usual architecture of SEGAN (Speech Enhancement Generative Adversarial Network) consists of several layers, including :
Generator Network :

- **Input Layer** :
  Takes the noisy speech signal as input.
- **Convolutional Layers** :
  These layers use filters to extract relevant features from the input signal. They capture the temporal dependencies and spatial patterns in the speech.
- **Batch Normalization** :
  Normalizes the activations of the previous layers, stabilizing the training process and accelerating convergence.
- **LeakyReLU Activation** :
  Introduces non-linearity and helps prevent the generator from collapsing the output.
- **Deconvolutional Layers** :
  These layers upsample the feature maps, gradually increasing the resolution and generating the clean speech signal.
- **Output Layer** :
  Produces the enhanced/clean speech signal.

Discriminator Network :

- **Input Layer** :
  Takes the input speech signal, either clean or distorted, as input.
- **Convolutional Layers** :
  Similar to the generator, these layers capture features from the input speech signal.
- **LeakyReLU Activation** :
  Introduces non-linearity and helps prevent the discriminator from saturating.

- **Fully Connected Layers** :
  These layers process the extracted features and make a binary classification between clean and distorted speech.

- **Output Layer** :
  Outputs a probability indicating the classification result.

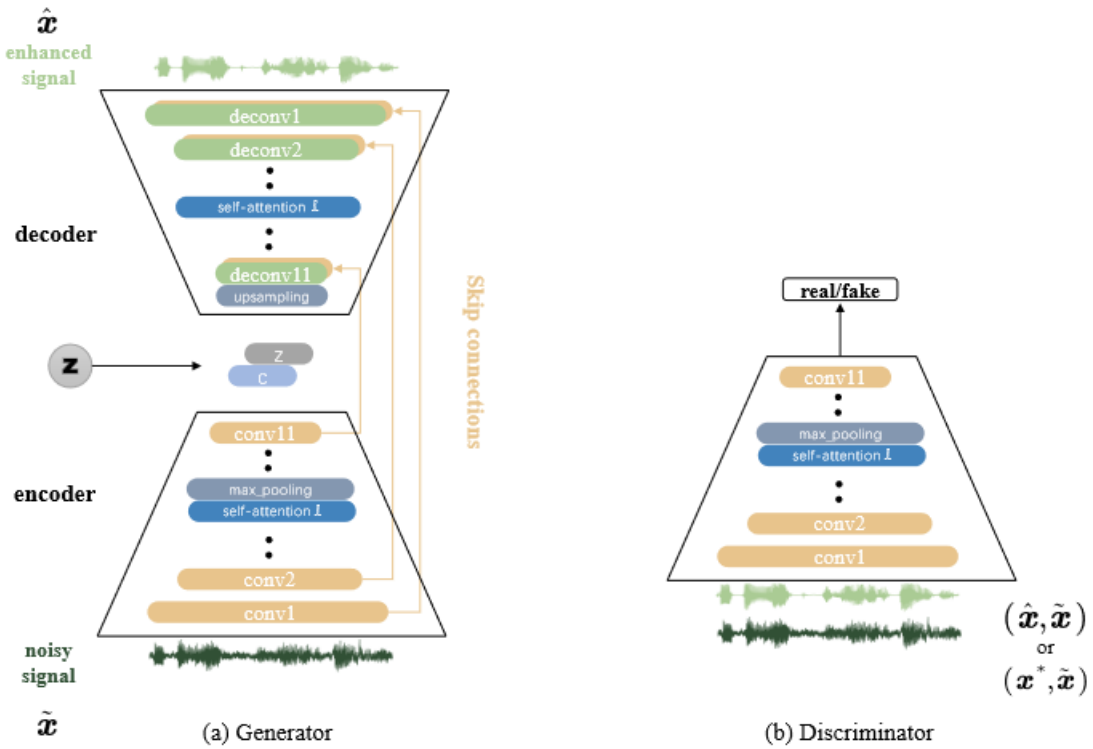


FIG. 2.16 : Illustration of the architecture of speech enhancement GAN (SEGAN)[19].

2.4 Conclusion

In conclusion, this chapter has provided a comprehensive overview of the speech recognition system, exploring its key components and methodologies. We have discussed the importance of signal processing and feature extraction, Furthermore, we have delved into the advancements in deep learning models, including CNN, Autoencoder, and SEGAN. These models have revolutionized the field by improving the accuracy and robustness of speech-to-text conversion. By gaining a deeper understanding of the underlying principles and architectures of these models, we can further enhance the performance and efficiency of speech recognition systems. The research and development in this area continue to drive progress and open new possibilities for real-world applications.

Chapitre 3

Fundamentals of Quadrotors

3.1 Introduction

In recent years, drones have emerged as versatile and innovative technologies with a wide range of applications across various industries. From aerial photography and surveillance to package delivery and research, drones have revolutionized many fields. One of the most common types of drones is the quadrotor, which utilizes the unique capabilities of brushless DC (BLDC) motors to achieve controlled flight and maneuverability.
This chapter focuses on the classification of drones and provides a detailed exploration of quadrotors, their movements, and the underlying theory of operation of BLDC motors. Understanding these fundamental concepts is essential for anyone interested in the design, operation, and application of drones.

3.2 Classification of drones

There is not a single way to classify drones, as they can be categorized according to several criteria such as autonomy, range, altitude, mission, control systems, dimensions, mode of propulsion.

3.2.1 Classification by size

Micro-drones

Their size range is from centimeters to a few tens of centimeters and their traits are defined by a low payload and reduced flight performance in windy conditions.figure 3.1 shows the micro drone

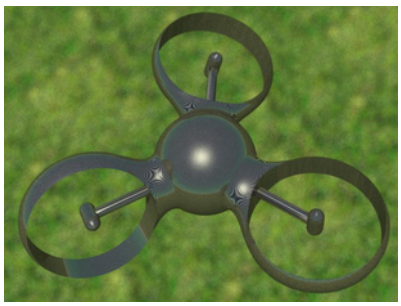


Fig. 3.1 : Micro-Drone.

Mini-drones

They weight a few kilograms and their wingspan is up to 1,2 meters.figure 3.2 shows the micro drone

FIG. 3.2 : Mini drone.

Male (Medium Altitude Long Endurance)

These flying drones have a medium-altitude, long-endurance capability and they can carry weapons, which makes them comparable in size to conventional airplanes.figure 3.3 shows the MALE drone

FIG. 3.3 : MALE drone.

Hale (High Altitude Long Endurance)

These are high-altitude, long-endurance flying drones with a wide wingspan.figure 3.4 shows the HALE drone

FIG. 3.4 : HALE drone.

3.2.2 Classification by propulsion mode

Fixed-wing

These drones use fixed wings for their mode of movement.figure 3.4 shows the Fixed-wing drone.

Fig. 3.5 : Fixed-wing drone.

Flapping wings

They are an alternative propulsion system for mini and micro aircraft. Wing flapping increasingly reproduces bird or insect flight .figure 3.6 shows the Flapping-wing drone.

Fig. 3.6 : Flapping-wing drone.

VTOL rotating wings

These drones perform vertical takeoff and landing. They are capable of performing stationary flight at low speed and low altitude.figure 3.7 shows the VTOL rotating wings drone.

Fig. 3.7 : VTOL rotating wings drone.

-Monorotors :
They use a single rotor as the main actuator.

-Birotors :
There are several types of two-rotor configurations such as the classic helicopter .There are also devices with two rotors on the same axis rotating in opposite directions such as the Hover-Eye , and the drone with a coaxial counter-rotating variable pitch birotor.

-Trirotors :
There are three categories of three-rotor drones, namely the trirotor , the vectron , and the self-stabilizing helicopter

-Multirotors :
It has multiple rotors with opposing rotation directions in pairs to compensate for the reaction torque.figure 3.8 shows the Multirotor.

FIG. 3.8 : Multirotor.

3.3 The quadrotor

A quadrotor is a rotary-wing drone consists of four rotors, It consists of four fixed-pitch blades coupled to direct current motors arranged in a cross formation, with the control electronics typically placed at the center. To prevent the quadrotor from spinning on itself or around its z-axis, two rotors spin in one direction while the other two spin in the opposite direction. For control purposes, each pair of rotors spins in the same direction, positioned at opposite ends of a cross arm.The operation of a quadrotor is quite unique. By cleverly adjusting the motor power, it can ascend, descend, tilt left or right (roll), tilt forward or backward (pitch), and rotate on its own axis (yaw). The quadrotor has six degrees of freedom : three rotational and three translational movements. Remarkably, these six degrees of freedom can be controlled using just four triggers.The quadrotor's structure as it is shown in figure 3.9 is depicted with black lines, the fixed body (B-frame) is shown in green, and the blue indicates the angular velocities of the rotors and their corresponding variable names. For each rotor, two arrows are displayed : curves indicate the rotation direction, while the others represent the speeds. Importantly, the latter vectors always point upward, disregarding the clockwise direction[20].

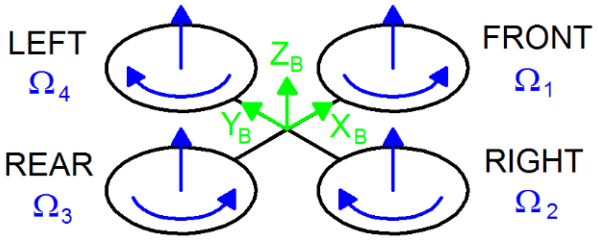


FIG. 3.9 : the structure model of a quadrotor[20].

.

3.4 Movements of the quadrotor

The quadrotor has four (04) rotors defined in space by six (06) Degrees of Freedom (DOF), three rotational movements, and three translational movements. By cleverly varying the rotation speeds of the motors, it is possible to make it : rise/fall, tilt left/right (roll), or forward/backward (pitch), or even to rotate on itself (yaw) ,as it is shown in the figure 3.9 The movements of a quadrotor while the arrow width is proportional to rotor speeds.

- **Rotate on itself (yaw) Movement** :
  It is obtained by increasing the speed of rotors (1 and 3) and proportionally decreasing the speed of rotors (2 and 4).
- **Forward/backward (pitch) Movement** :
  Similarly, by applying a couple around the y-axis, i.e., a difference in thrust between rotor (01) and rotor (03),increasing (or decreasing) the speed of the rear propeller and decreasing (or increasing) the speed of the front propeller. a pitch movement is obtained. This movement is coupled with a translation movement along the x-axis.
- **Tilt left/right (roll) Movement** :
  To achieve a roll movement, a couple is applied around the x-axis ,i.e., a difference in thrust between rotor (02) and rotor (04).increasing (or decreasing) the speed of the left propeller and decreasing (or increasing) the speed of the right propeller. This movement is coupled with a translational movement along the y-axis.
- **Vertical (Upward/downward) Movement** :
  It simply corresponds to the ascent/descent of the quadrotor.Ascending is achieved by increasing the speed of all four motors equally.While Descending is achieved by reducing the speed of the motors.

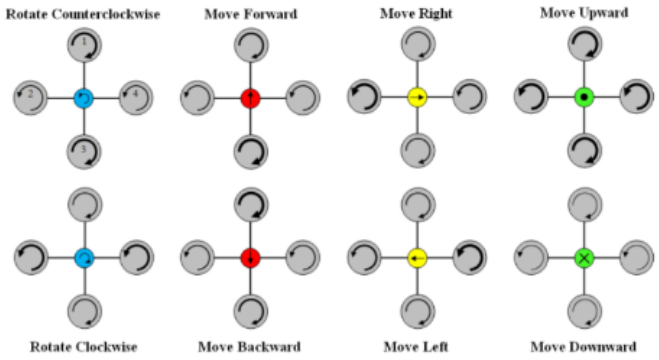


Fig. 3.10 : :The movements of a quadrotor.

3.5 Brushless DC Motors

A Brushless Direct Current (BLDC) motor is an actuator that converts electrical energy into mechanical energy without the use of brushes. Instead, it relies on electronic commutation for operation.

3.5.1 CONSTRUCTION AND OPERATING PRINCIPLE

BLDC motors belong to the category of synchronous motors, where the magnetic field generated by the stator and the magnetic field generated by the rotor rotate at the same frequency. BLDC motors can be found in single-phase, 2-phase, and 3-phase configurations, with the stator having a corresponding number of windings. Among these options, 3-phase motors are the most widely used.[21]

.

3.5.2 Stator

The stator of a BLDC motor is a fixed component consisting of stacked steel laminations with windings placed in axially cut slots along the inner periphery (as depicted in Figure 3.11). In most BLDC motors, three stator windings are connected in a star configuration.[21]

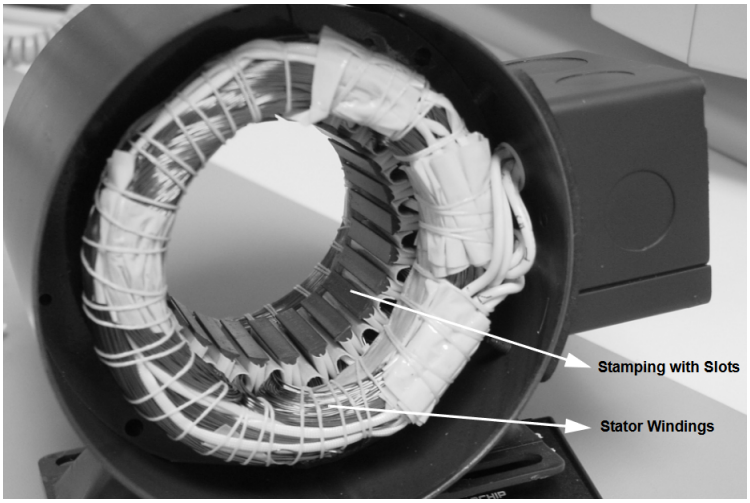
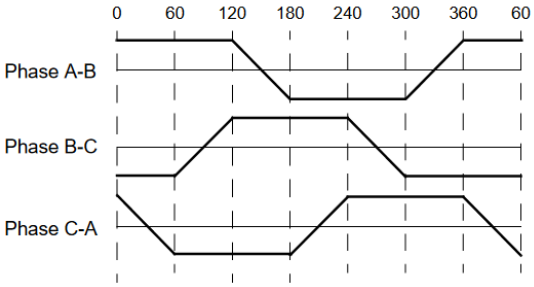
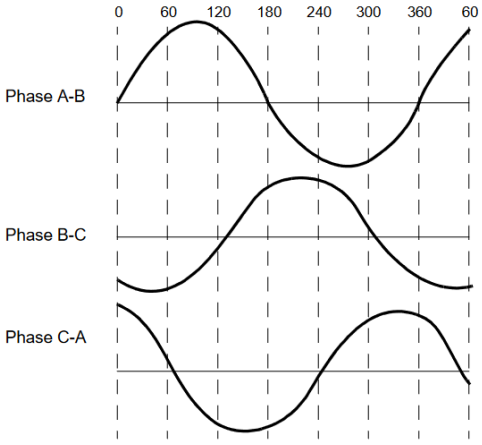


Fig. 3.11 : STATOR OF A BLDC MOTOR[21].

There are two types of stator winding variants : trapezoidal and sinusoidal motors. These variants are distinguished by the interconnection of coils in the stator windings, which determines the type of back Electromotive Force (EMF) they produce.The trapezoidal motor generates a back EMF in a trapezoidal fashion, while the sinusoidal motor produces a sinusoidal back EMF (as shown in Figure 3.12a and Figure 3.12b). Additionally, the phase current in each motor type exhibits either trapezoidal or sinusoidal variations. This characteristic makes the torque output of a sinusoidal motor smoother compared to that of a trapezoidal motor.[21]



(a) TRAPEZOIDAL BACK EMF[21].



(b) SINUSOIDAL BACK EMF[21].

3.5.3 Rotor

The rotor of a BLDC motor is comprised of permanent magnets, as shown in figure 3.13 which can range from two to eight pole pairs, with alternating North (N) and South (S) poles. The selection of magnetic material for the rotor depends on the desired magnetic field density. Typically, ferrite magnets are utilized to create permanent magnets. Figure 4 illustrates various cross-sectional arrangements of magnets in a rotor.[21]

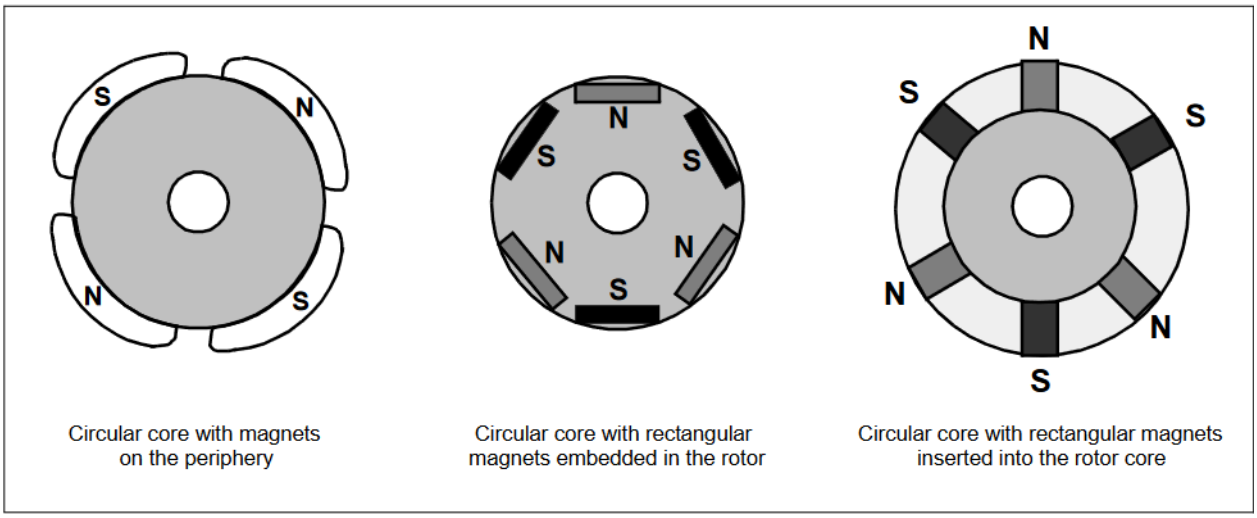


Fig. 3.13 : ROTOR MAGNET CROSS SECTIONS[21].

3.5.4 Hall Sensors

In order to rotate a BLDC motor, the stator windings need to be energized in a specific sequence. To determine which winding should be energized next in the sequence, it is crucial to know the position of the rotor. This is accomplished using Hall effect sensors that are integrated into the stator.In general, BLDC motors have three Hall sensors embedded into the stator on the non-driving end of the motor.When the rotor's magnetic poles come

close to the Hall sensors, they generate either a high or low signal, indicating the presence of a North (N) or South (S) pole near the sensors. By analyzing the combination of signals from these three Hall sensors, the precise commutation sequence can be determined.[21]

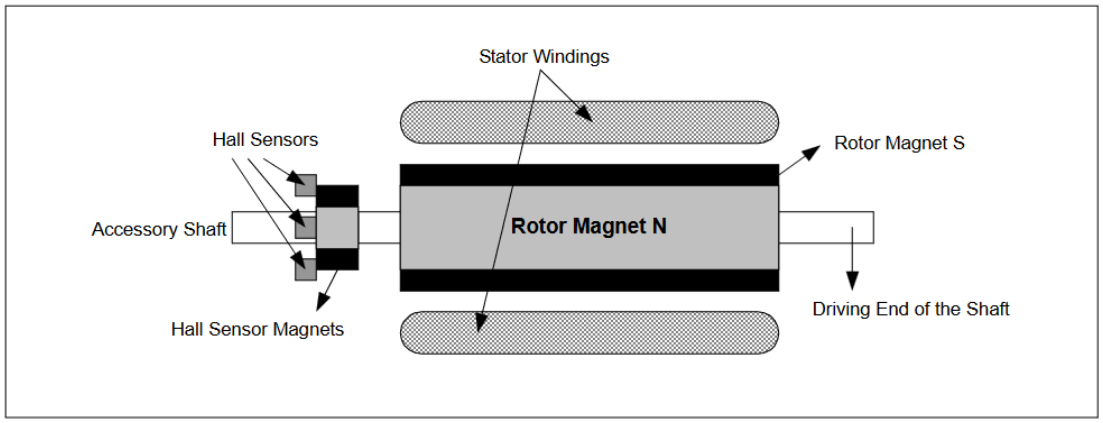


Fig. 3.14 : BLDC MOTOR TRANSVERSE SECTION[21].

Figure 3.14 depicts a transverse section of a BLDC motor with a rotor containing alternating N and S permanent magnets. Depending on the physical placement of the Hall sensors, there are two versions of output : a 60° or 120° phase shift between the sensors. The motor manufacturer defines the commutation sequence based on this configuration, which must be followed when controlling the motor.[21]

3.5.5 TORQUE/SPEED CHARACTERISTICS

The motor has two torque parameters : peak torque (T P) and rated torque (TR). Under continuous operations, the motor can handle loads up to the rated torque. Within a specific speed range which can go up to the rated speed, the torque remains constant .The motor can be run up to the maximum speed but the torque starts decreasing. Figure 3.15 shows an example of torque/speed characteristics. [21]

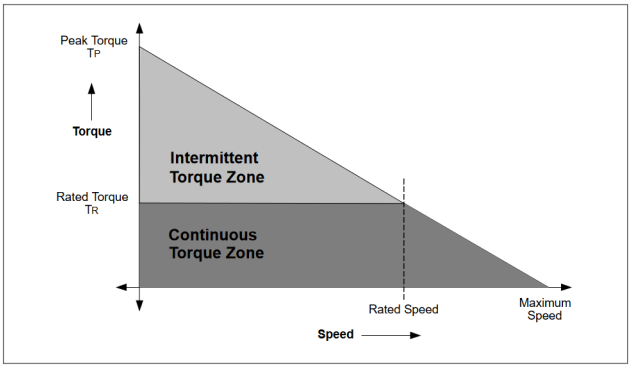


Fig. 3.15 : TORQUE/SPEED CHARACTERISTICS[21].

3.5.6 Theory of Operation

In a BLDC motor, the commutation sequence determines which winding is energized positively, which is energized negatively, and which remains non-energized. The interaction between the magnetic field generated by the stator coils and the permanent magnets produces torque. The peak torque occurs when these fields are at a 90° angle to each other, and the torque decreases as the fields move together. To maintain the motor's operation, the magnetic field produced by the windings must shift position as the rotor catches up with the stator field. This shifting is achieved through "Six-Step Commutation," which defines the sequence of energizing the windings.[21].

3.5.7 Control Methods of the brushless dc motor

There are numerous control methods available for brushless DC motors, each with its own principles and advantages.Such as :

- **Sensorless Control** : This method eliminates the need for additional sensors by estimating the rotor position and speed using advanced algorithms. It relies on the back electromotive force (EMF) and other motor parameters for estimation.
- **Field-Oriented Control (FOC)** : Also known as vector control, FOC separates the control of the motor's torque and flux components. It enables precise control of the motor's performance by decoupling these components and controlling them independently.
- **Direct Torque Control (DTC)** : DTC focuses on controlling the motor's torque directly, without explicitly controlling the flux. It offers fast torque response and precise control by continuously estimating the flux and torque using a hysteresis-based approach.
- **Model Predictive Control (MPC)** : MPC utilizes a mathematical model of the motor and predicts its future behavior to generate control actions. It allows for advanced control strategies, optimal performance, and the ability to handle constraints.
- **Adaptive Control** : This method adjusts the controller parameters in real-time based on the motor's operating conditions. It enables the motor to adapt to variations in load, temperature, and other factors for improved performance and efficiency.

However, one conventional control method commonly employed is sensor-based control, the control of the BLDC motor is achieved through the utilization of Hall-effect sensors to determine the rotor's position and enable proper commutation. However, the speed control of the motor is accomplished by manipulating the voltage across the motor, which is facilitated by varying the duty cycle of the PWM signal used to control the six switches of the three-phase bridge. The PWM signal is responsible for controlling the energization of the motor's windings by switching the current flow through the active phases of the three-phase winding system. By adjusting the duty cycle of the PWM signal, the average

voltage applied to the motor can be modulated. This variation in voltage, in turn, allows for precise control of the motor's speed.[22] Figure 3.16 shows an example of a PWM signal.

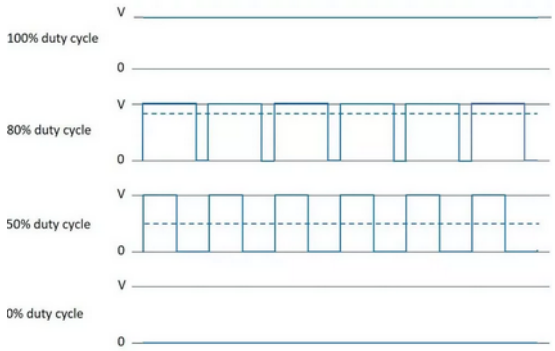


Fig. 3.16 : PWM signal with a varying duty cycle.

In this control method, only two inverter switches, one in the upper inverter bank and one in the lower inverter bank, are conducting at any given instant. These discrete switching events ensure the maintenance of a specific sequence of conducting pairs of stator terminals, which is crucial for the generation of a constant output torque. The Hall-effect sensors, mounted at appropriate points around the stator with a 60° relative offset from each other, deliver digital pulses based on the passing rotor magnets' magnetic field. These pulses are decoded into the desired three-phase switching sequence for the motor's inverter. The Hall sensors change their state every 60 electrical degrees of rotation, resulting in a six-step sequence to complete an electrical cycle.The process of switching the current to flow through only two phases for every 60 electrical degree rotation of the rotor is called electronic commutation. However, it's important to note that one electrical cycle may not correspond to a complete mechanical revolution of the rotor. The number of electrical cycles required to complete a mechanical rotation is determined by the rotor pole pairs, where each rotor pole pair corresponds to one electrical cycle[22].
A BLDC motor drive with a six-step inverter and Hall position sensors is shown in Figure 3.17

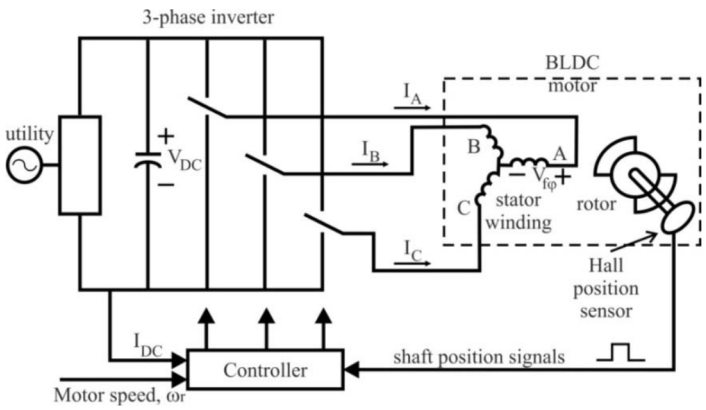


Fig. 3.17 : Electronically commutated BLDC motor drive [22].

Figure 3.18 shows an example of Hall sensor signals with respect to back-EMF and the phase current. One of the Hall sensors changes the state every 60 electrical degrees of rotation.

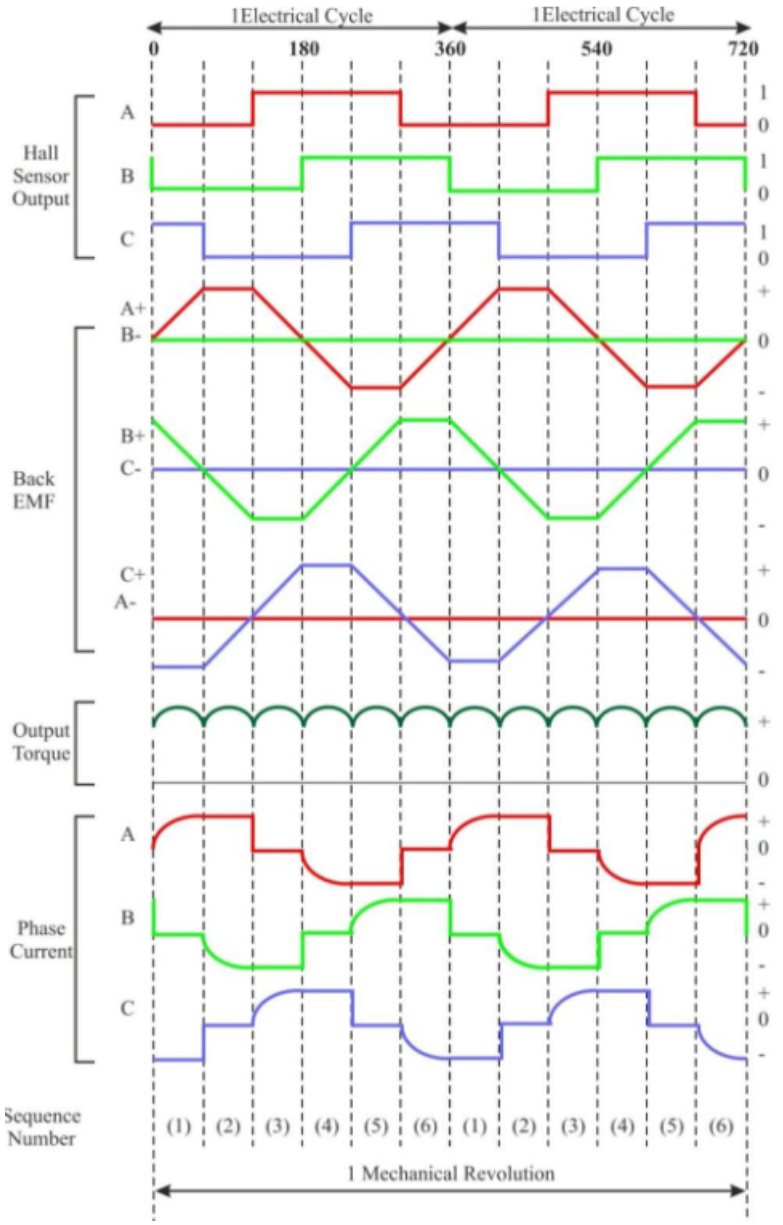


Fig. 3.18 : Hall sensor signal, back-EMF, output torque and phase current [22].

Overall, in the Conventional Control Method Using Sensors, the PWM signal's duty cycle is adjusted to regulate the motor's speed by controlling the voltage across the motor. This, coupled with the precise commutation determined by the Hall-effect sensors, enables the production of a constant output torque for the BLDC motor.

3.6 Conclusion

In conclusion, this chapter provided an overview of drones, focusing on the classification of drones, quadrotor movements, and the theory of operation of BLDC motors. The classification of drones allows for a better understanding of their diverse applications and capabilities. The exploration of quadrotor movements revealed their agility and maneuverability in performing complex aerial tasks. The discussion on BLDC motors emphasized their crucial role in powering quadrotors and enabling controlled flight.

Chapitre 4

Controlling the quadrotor with keyword spotting model : implementation and results

4.1 Introduction

This chapter focuses on the development and implementation of a keyword spotting model for controlling a quadrotor in both noisy and noiseless environments. The goal is to enable precise and responsive control of the quadrotor through spoken commands. The chapter starts by building two keyword spotting models, one for the noiseless environment and one for the noisy environement while executing two solutions. The models are trained using a diverse dataset to ensure adaptability. The next step , we built and operated a quadrotor that executed commands from an input. finally, integrating the model into the quadrotor system and conducting experiments. The chapter highlights the model's effectiveness in controlling the quadrotor and achieving accurate movements.

4.2 Keyword spotting

We experimented with a keyword spotting model trained using Google Colab. We tested the model using the usb microphone. This approach allowed us to quickly prototype and evaluate the performance of the model used. in this section we will talk about all the approaches used to use the best model and implement it.

4.2.1 Software and Libraries

Python

Python is an interpreted, multi-paradigm, and cross-platform programming language. It supports structured imperative, functional, and object-oriented programming. It features strong dynamic typing, automatic memory management through garbage collection, and exception handling.Python programming language was created in 1989 by Guido van Rossum in the Netherlands. The name "Python" is a tribute to the television series "Monty Python's Flying Circus," which G. van Rossum is a fan of. The first public version of the language was released in 1991[11].

Scikit-learn

Scikit-learn is a free Python library for machine learning. It is developed by numerous contributors, including academics. It provides a framework with many ready-to-use algorithm libraries. These libraries are made available to data scientists. It includes functions for estimating random forests, logistic regressions, classification algorithms, and support vector machines (SVM). It is designed to work well with other Python libraries, such as NumPy and Pandas[11].

Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks.
Since version 2.4, it only supports TensorFlow. It is designed to enable rapid experimentation with deep neural networks and aims to be user-friendly, modular, and extensible. It was developed as part of the research effort for the ONEIROS project (Open-ended Neuro-Electronic Intelligent Robot Operating System)[23].

TensorFlow

TensorFlow is a comprehensive open-source framework for machine learning applications. It is a symbolic mathematics toolkit that performs a variety of tasks, including training and inferring deep neural networks. It allows programmers to build machine learning applications using a variety of tools, frameworks, and community resources. It was created by Google and is currently the most widely used deep learning package in the world. Google utilizes machine learning in all its products to enhance search, translation, image captions, and recommendations. TensorFlow was initially released in late 2015, with the first stable version following in 2017. It is free and open-source under the Apache Open Source License. Without paying anything to Google, you can use, modify, and redistribute the updated version for a fee. For our project, through its Keras API, TensorFlow played a central role in designing various deep neural network models for classifications. Additionally, we used TensorFlow for data preparation and model evaluation.[11].

4.2.2 Hardware

In terms of hardware, our experimental setup for this section involved the use of a Raspberry Pi card, a USB microphone, and Google Colab.

Raspberry pi card

The Raspberry Pi 4 Model B with 4GB of RAM is a powerful single-board computer. It offers various features and capabilities for different projects and applications. Key features of the Raspberry Pi 4 Model B (4GB RAM) include :

- **Processor** : The Raspberry Pi 4 is equipped with a quad-core ARM Cortex-A72 processor, providing increased performance compared to previous models.
- **Memory** : It has 4GB of LPDDR4 RAM, allowing for smooth multitasking and running resource-intensive applications.

- **Connectivity** : The board offers dual-band 2.4GHz and 5GHz Wi-Fi, as well as Gigabit Ethernet for fast internet connectivity. It also has Bluetooth 5.0 for wireless communication.
- **GPIO Pins** : The Raspberry Pi 4 has a 40-pin GPIO header, providing a range of digital and analog input/output options for connecting external devices and sensors.
- **USB Ports** : It features two USB 2.0 ports and two USB 3.0 ports, allowing for the connection of various USB devices such as keyboards, mice, storage drives, and peripherals.
- **Video and Display** : The board supports dual-monitor setups with two micro HDMI ports, capable of outputting 4K resolution. It also has a MIPI CSI camera interface for connecting a Raspberry Pi camera module.
- **Storage** : The Raspberry Pi 4 uses a microSD card for primary storage, offering flexibility in terms of storage capacity and operating system installations.
- **Operating System** : It supports various operating systems, including the official Raspberry Pi OS (formerly known as Raspbian), as well as other Linux distributions and specialized software.
- **Power Supply** : The board requires a 5V USB-C power supply to operate, providing sufficient power for the CPU, RAM, and connected peripherals.

Usb microphone

The microphone is a USB microphone,it has the following specifications :

- Microphone Type : USB Microphone.
- Sensitivity : -67 dBV/Pa (-47 dBV/Pascal) +/- 4 dB.
- Frequency Response : 100 Hz to 16 kHz.

As a USB microphone, it provides the convenience of plug-and-play functionality, allowing easy connectivity to devices such as computers, laptops, and Raspberry Pi. The sensitivity of -67 dBV/Pa (-47 dBV/Pascal) +/- 4 dB indicates its ability to capture sound accurately, even at low sound pressure levels. The microphone's frequency response of 100 Hz to 16 kHz indicates the range of frequencies it can effectively capture, covering a wide spectrum of audible sounds.

Google colaboratery

Google Colaboratory (Colab) is a product of Google Research. Colab provides an environment where anyone can write and execute Python code through a web browser. It is particularly well-suited for machine learning, data analysis, and education. Technically speaking, Colab is a hosted service for Jupyter notebooks that requires no setup and provides free access to computing resources, including GPUs.

4.2.3 Metrics Used

Our problem is a classification problem. Therefore, the choice of metrics is important to accurately measure the model's quality. The most commonly used metric in machine learning is accuracy but we will use also other metrics .
. we have chosen to use the following metrics to assess the quality of our models , (not all the models used uses all of these metrics) :
- AUC ROC Score.
- F1-score.
- Accuracy.
- Confusion matrix.
- Sensitivity.
- Precision.

We will describe some of these metrics in more detail in the following sections.

Accuracy

It measures the overall correctness of the model's predictions by calculating the following formula :
Accuracy = (Number of correct predictions) / (Total number of predictions) As mentioned earlier, accuracy alone may not fully represent the quality of the model. However, when combined with the other mentioned metrics, it can provide meaningful insights for verifying the quality of the classification.

Confusion matrix

The confusion matrix is a graphical representation of our model's performance.

The confusion matrix provides a summary of the prediction results in a classification problem. It highlights both correct and incorrect predictions, distributing them by class. The results are then compared with the actual values. This matrix helps us understand how the classification model gets confused when making predictions.

AUC ROC Score

Rank metrics, such as the ROC curve, focus on evaluating classifiers based on their effectiveness in separating classes through the probabilities of each sample belonging to one class or the other.

A ROC curve is a diagnostic graph that summarizes the behavior of a model by calculating the false positive rate and the true positive rate for a set of predictions by the model under different thresholds. This results in a curve graph that allows us to assess the quality of our model. The closer the area under the curve (AUC) tends to 1, the better the model is. Therefore, the AUC ROC Score is simply the integral of the generated function, representing the area under the curve.

This metric was primarily designed for binary classification, comparing the quality of

a model in distinguishing between two distinct classes. However, it can be adapted to a multiclass problem like ours using two strategies :

1. One vs One (OvO) : This strategy considers all possible combinations between each pair of classes, with each pair having a defined score indicating the model's quality in differentiating between the two classes.

2. One vs Rest (OvR) : With this strategy, we compare the model's quality in recognizing a specific class, in other words, differentiating between one particular class and the rest of the classes. with each combination calculated separately, and then the average is taken. In our case, we have a list the combinations .

In this work, we have chosen the second strategy, OvR, as it seems more meaningful for our project.

F1 score

The F1-score is the most commonly used metric in imbalanced problems, and the use of this metric combines two important metrics into one : precision and recall. Therefore, it is a perfect metric for assessing the quality of classification.

Precision and recall (or recall rate) are two complementary metrics that can be described as follows : Precision represents the fraction of examples assigned to the positive class that truly belong to the positive class, while recall represents how well the positive class was predicted. They are calculated using the following equations :

Precision = true positives / (true positives + false positives)

Recall = true positives / (true positives + false negatives)

Sensitivity and Specificity

They are threshold-based metrics, quantify the prediction errors in classification.They are designed to summarize the fraction, ratio, or rate of cases where a predicted class does not match the expected class in a given dataset.

These two metrics indicate the ratios between false positives, true positives, false negatives, and true negatives, as described by the following formulas :

Sensitivity : Also known as true positive rate, it summarizes how well the positive class was predicted :

Sensitivity = true positives / (true positives + false negatives)

Specificity : It is the complement of sensitivity, or the true negative rate, summarizing how well the negative class was predicted. It is calculated in the same way as recall :

Specificity = true negatives / (true negatives + false positives)

4.2.4 Problem statement

In noisy environments, the presence of background noise poses a significant challenge for accurate speech recognition. The noise interferes with the clarity of the speech signal, making it difficult for the CNN model to extract relevant features and classify the input correctly. This results in a decrease in the overall classification accuracy.

4.2.5 Proposed solutions

The problem statement revolves around finding a solution to enhance the performance of the CNN model in noisy environments. This involves developing an approach or techniques that can effectively remove or mitigate the impact of noise on the speech signals. By reducing the noise interference, the CNN model can better capture the essential features of the speech and make accurate classifications, in the next section we propose solutions for this problem

Methodology

The deep learning models used in this project are 5 : CNN classifier model using clear data (model 01) , Binary classifier (model 02) and an Autoencoder model, a SEGAN model and a CNN classifier model for noisy environment (CNN model 02) .
With a deep understanding of the problem domain and the intricacies of speech recognition, we designed and implemented all of these models .

CNN classifier model (model 01) : specifically designed for clear data. Binary classification model : to differentiate between data and noise, for noisy environment . Autoencoder model and SEGAN model are built to remove the noise from data before inputing it into the CNN classifier model for noisy environment (model 02), to be respectively solution one and solution two. The CNN classifier for noisy environment (model 02) : is built to classify clear data coming from the autoencoder or the SEGAN.

For the methodology described in figure 4.1, we first had a base CNN model which is a multiclassifier, model (01), with 11 classes : up, down, right, left, on, off, stop, go, forward, backward, noise. This model was initially trained without any specific noise removal techniques using Google Colab. We then tested this model using the microphone on our laptop. Afterward, We proceeded to implement the model on a Raspberry Pi along with a USB microphone.

Later on, We explored methods for reducing noise in the audio data. We experimented with different approaches, first we ve used a band-pass filter, this is the code used for this filter :

```
def apply_band_pass_filter(audio_signal, lowcut, highcut, fs,order=5):
    nyquist_freq = 0.5 * fs
    low = lowcut / nyquist_freq
    high = highcut / nyquist_freq
    b, a = butter(order, [low, high], btype='band')
```

```
    filtered_signal = lfilter(b, a, audio_signal)
    return filtered_signal
```

The apply_band_pass_filter function applies a band-pass filter with the specified low_freq = 20 Hz and high_freq = 4000 Hz to the audio signal to reduce noise from it. However, it was found that this approach did not yield satisfactory results in reducing noise, as noise can have frequencies similar to the data so this filter can remove a part from data which will lead to a misclassification.
To address this issue, we employed a binary classifier to distinguish between noisy data and noise, the noisy data was gotten by mixing clear data with different levels of noise. If the input audio was classified as data, it was passed through a noise removal model and then further classified with a cnn multiclassifier different then cnn classifier used in the base model.
The first solution : We used as shown in figure 4.1 an autoencoder after that a CNN classifier The second solution : We used as shown in figure 4.1 a SEGAN model after that a CNN classifier

These two solutions will be further discussed in the next sections.

Then, based on the environment , if it is noisy or noiseless, the correspondant model is used for implementation. In our case, the environment was noiseless so we used model 01 as a keywordspotting model of our project .
The performance of each model will be evaluated and analyzed. For the initial model trained without noise removal techniques, its accuracy, precision, recall, F1-score, and confusion matrix will be examined to assess its classification quality.
Next, for the model trained using noise reduction model, the metrics which will be calculated are : Accuracy,loss and confusion matrix and compared to the initial model. This evaluation will provide insights into the effectiveness of the noise reduction techniques in improving the model's performance.

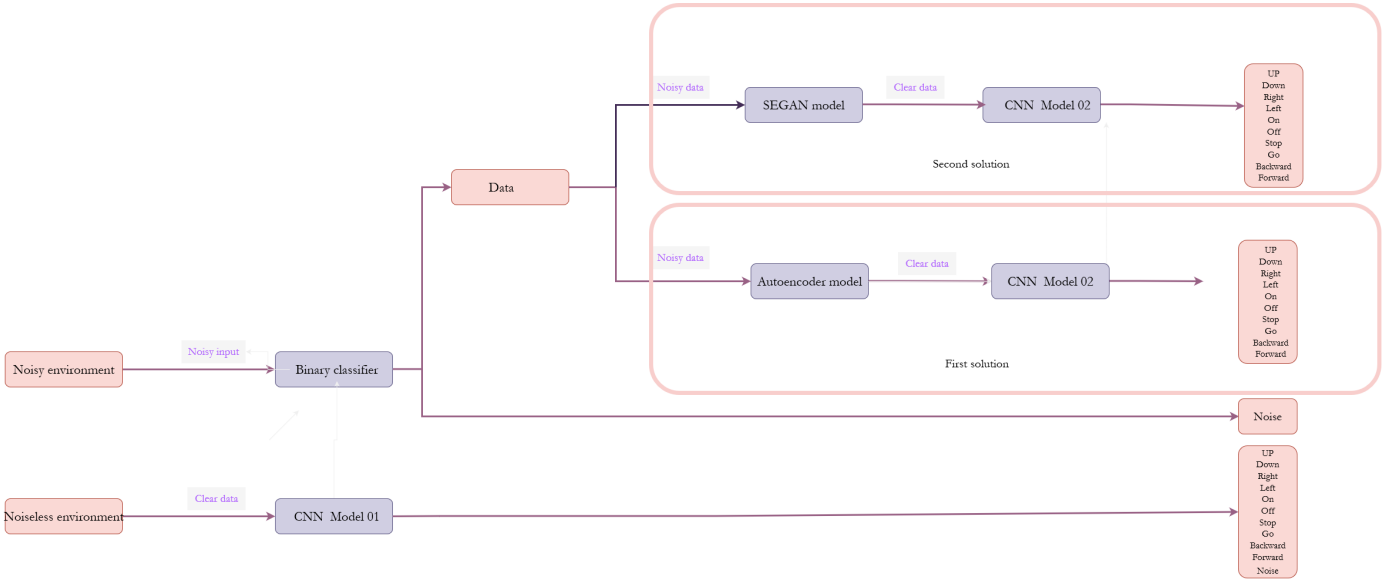


FIG. 4.1 : Diagramm of the methodology

Dataset Used

The dataset we worked with is available on Tensorflow[24].It is An audio dataset of spoken words designed to help train and evaluate keyword spotting systems. It consists of a total of 100,503 wav file constituting our dataset. This set is of one-second .wav audio files, each containing a single spoken English word. These words are spoken by a variety of different speakers. The audio files are organized into 31 folders based on the word they contain.
The words are "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", and "Nine". To help distinguish unrecognized words, there are also ten auxiliary words, These include "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow", plus Background Noise ,that containes 6 wav files : doing_the_dishes.wav of 1min35sec longer,dude_miaowing.wav of 1min01sec longer, exercise_bike.wav of 1min1sec longer, pink_noise.wav of 1min longer ,running_tap.wav of 1min1sec longer,white_noise.wav of 1min longer, to help train networks to cope with noisy environments. No details were kept of any of the participants age, gender, or location, and random ids were assigned to each individual.
we provide an histogram shows the number of wav files in each folder shown in figure 4.2

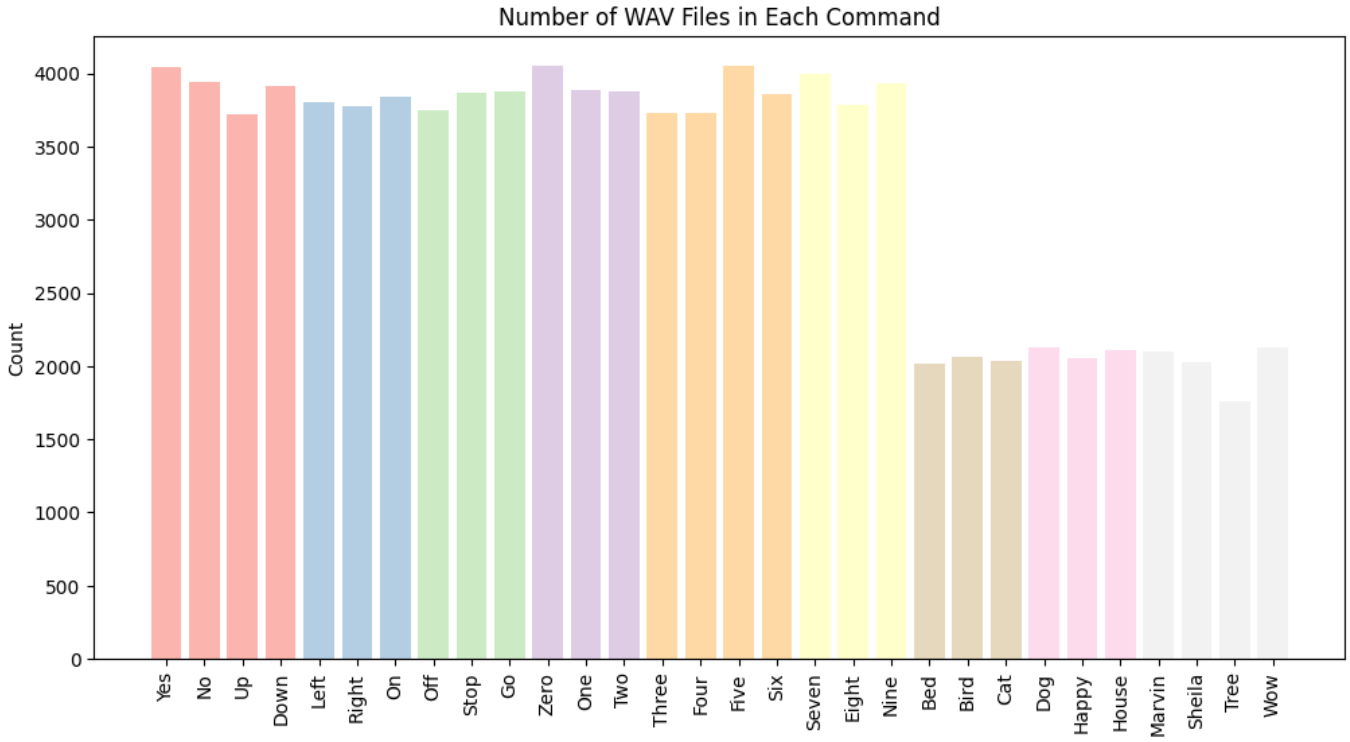


Fig. 4.2 : Number of wav files in each folder of the dataset

Data preparation and preprocessing

:

In this section, we outline the steps taken to prepare and preprocess the dataset for our

keywordspotting task. We will be working with a total of 11 classes of commands : ”up” ,”down”, ”right”, ”left” ,”on” ,”off” ,”stop” ,”go” ,”forward” ,”backward” and a special class called ”noise” for the CNN model 01 using the clear data , we will work with two classes of commands : ”data”,”noise”, with data contains 1000 wav files of each of these 10 commands :”up” ,”down”, ”right”, ”left” ,”on” ,”off” ,”stop” ,”go” ,”forward” ,”backward” for the binary classifier , and for the CNN model 02 we will work with the 10 commands without noise
for the autoencoder and segan , they work also with 10 noisy commands.

The ”noise” class consists of audio samples obtained by dividing the WAV files from the background_noise folder into 1-second segments.
Our data preprocessing involves removing any non-1-second WAV files from the dataset. This ensures consistency in the length of audio samples across all classes.
After preprocessing the data, we observed that the class ”noise” contained 398 WAV files, and the class ”forward” had 1452 WAV files. Since the number of WAV files in the ”noise” and ”forward” class fell short we decided to apply data augmentation techniques to these two classes.the data augmentation for audio technique is explained next.

Data augmentation for audio
Data augmentation is a technique used to increase the quantity of data by adding slightly modified copies of existing data or newly created synthetic data based on existing data. It acts as a regularizer and helps reduce overfitting during the training of a machine learning model.
Data augmentation in the context of audio involves applying various techniques to the existing audio data to create additional samples.These are the techniques we used in audio data augmentation : Time Stretching, Speed Perturbation and Pitch Shifting.
The data augmentation process allowed us to generate additional WAV files for the ”noise” class and ”forward” class, surpassing the initial count of 398 and 1452 respectively.

To train and evaluate our models effectively, we split the dataset into two sets : an 80% training set and a 20% validation set. This division allows us to train the model on a majority of the data while reserving a portion for evaluating its performance. we provide an information about the percentage of data in the sets training and validation and the correspondant number of wav files for each model.
In table 4.1 is for the CNN model for clear data model 01

sets	percentage	number of WAV files
training	80	13200
validation	20	3300
total	100	16500

Tab. 4.1 : Percentage of data in the CNN model for the clear data

In table 4.2 is for the CNN model for noisy environment model 02 :

sets	percentage	number of WAV files
training	80	12000
validation	20	3000
total	100	15000

TAB. 4.2 : Percentage of data in the CNN model for noisy environment model 02 :.

In table 4.3 is for the binary classifier

sets	percentage	number of WAV files
training	80	14074
validation	20	3518
total	100	17592

TAB. 4.3 : Percentage of data in the binary classifier

In table 4.4 is for the Autoencoder model

sets	percentage	number of WAV files
training	80	3200
validation	20	800
total	100	4000

TAB. 4.4 : Percentage of data in the Autoencoder model

after splitting the dataset, another preprocessing step is performed which is converting the wav files into spectrogram using STFT, after this the spectrograms will be loaded and will be ready to be an input to the models .
for the models : autoencoder and SEGAN noise was add to data for : 1000 hz to 5000hz. and then other steps were done, will be discussed in the next sections.

4.2.6 Solutions for the noiseless : the CNN model

CNN model for clear data

It takes raw WAV files as input and processes them using the Short-Time Fourier Transform (STFT) magnitude to compute the spectrogram, the model is trained and tested using WAV files without additional noise, the model's output consists of predicted commands : "up" ,"down", "right", "left" ,"on" ,"off" ,"stop" ,"go" ,"forward" ,"backward","noise".

Architecture

in this work, we propose to develop our CNN architecture as shown in figure 4.10

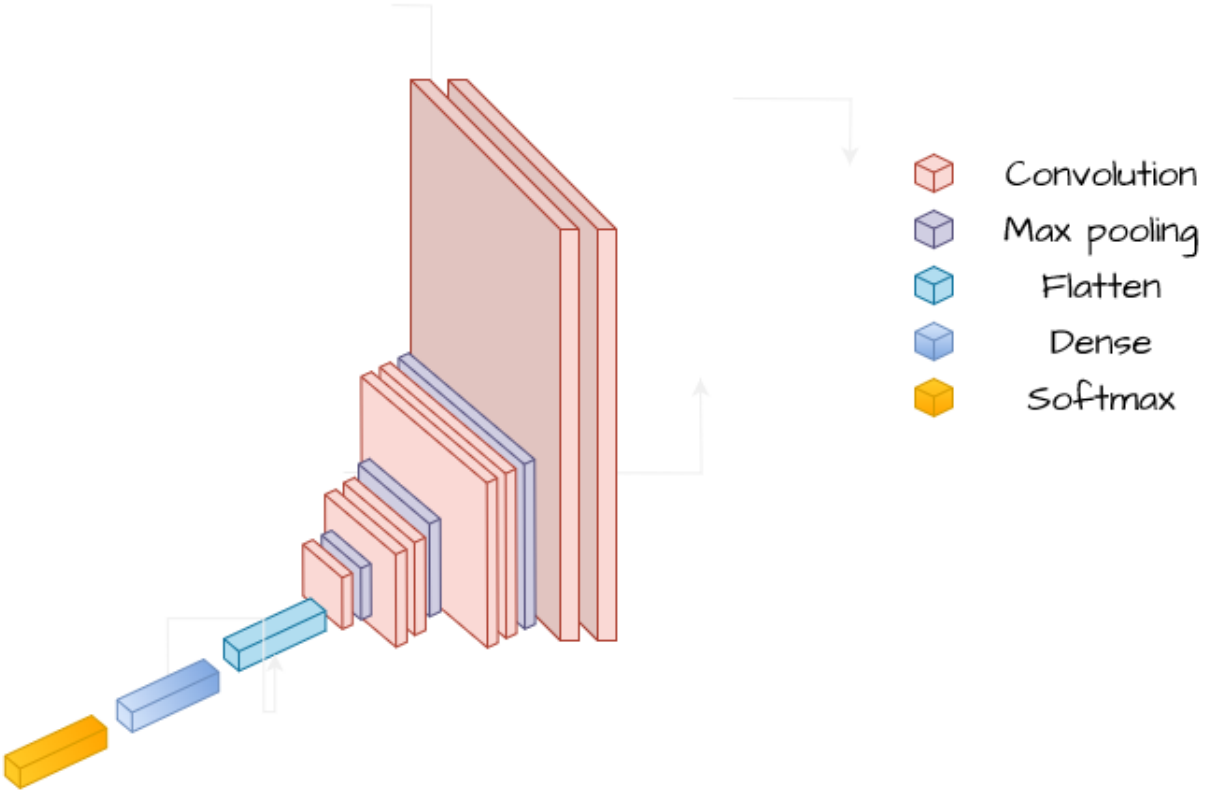


FIG. 4.3 : The architecture of the CNN model.

The architecture consists of over 136563 weights to be learned during the training process.It contains several layers. The input shape is (513, 33, 1), indicating an input image with a height of 513 pixels, width of 33 pixels, and 1 channel(grayscal).

The first Conv2D layer has 64 filters of size 3x3, which perform convolutions on the input image to extract relevant features. This is followed by a second Conv2D layer with 64 filters of the same size.

The MaxPooling2D layer reduces the spatial dimensions of the feature maps generated by the previous convolutional layers. In this architecture, it reduces the size by half, resulting in feature maps of size 256x16.

Next, there are two Conv2D layers with 32 filters of size 3x3 each. These layers further extract features from the reduced-size feature maps.

Another MaxPooling2D layer is applied, reducing the size to 128x8.

The subsequent Conv2D layers consist of two layers with 16 filters of size 3x3 each. These layers continue to extract features from the previous feature maps.

Another MaxPooling2D layer reduces the size to 64x4.

The Dropout layer is used for regularization, randomly setting a fraction of input units to 0 during training to prevent overfitting.

Following that, a Conv2D layer with 8 filters of size 3x3 is applied.

The Flatten layer converts the output from the previous layer into a 1D vector, ready to be passed to the fully connected layers.

The Dense layer is the hidden layer with 2048 neurons. It takes the flattened input and performs computations to generate a representation of the features extracted from the image.

Finally, there is an output Dense layer with 11 neurons, representing the number of classes in the classification task. This layer uses the softmax activation function to provide probabilities for each class.

This architecture is summarized in Table 4.5.

Type	Forme	Parametrs
input	(None, 513, 33, 1)	0
Conv2D	(None, 513, 33, 64)	960
Conv2D	(None, 513, 33, 64)	57408
MaxPooling2D	(None, 256, 16, 64)	0
Conv2D	(None, 256, 16, 32)	28704
Conv2D	(None, 256, 16, 32)	14368
MaxPooling2D	(None, 128, 8, 32)	0
Conv2D	(None, 128, 8, 16)	7184
Conv2D	(None, 128, 8, 16)	3600
MaxPooling2D	(None, 64, 4, 16)	0
Dropout	(None, 64, 4, 16)	0
Conv2D	(None, 64, 4, 8)	1800
Flatten	(None, 2048)	0
Dense	(None, 11)	22539
Total		136563

TAB. 4.5 : The architecture of the CNN model

Training

In our project, we conducted various training sessions, For these training sessions, we conducted training in the Python environment using the Tensorflow framework described above.
The input image size for this model is 513x33 with a single channel (1).As it is mentioned

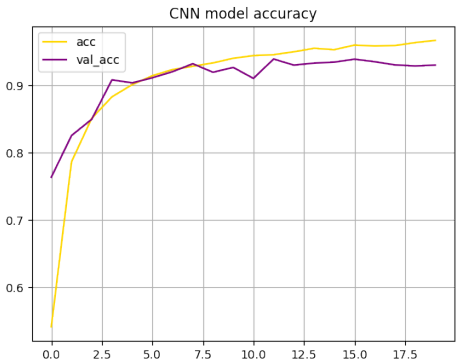
in the section the previous section.
- the Training was for 20 epochs.
To track the progress of training and monitor the model's performance, various metrics such as loss and accuracy were recorded and the confusion matrix.
Adam (Stochastic Gradient Descent) optimizer was employed. A batch size of 64 was used for training .
the table 4.11 Summarize all the parameters used during training.

	Architecture
Training parameters	136563
Input image size	513x33
Number of convolutional layers	7
Number of max-pooling layers	3
Dropout value	0.3
Activation function	RELU
Learning rate	0.001
Epochs	20
Batch size	64
Optimizer	Adam
Number of parameters	136 563

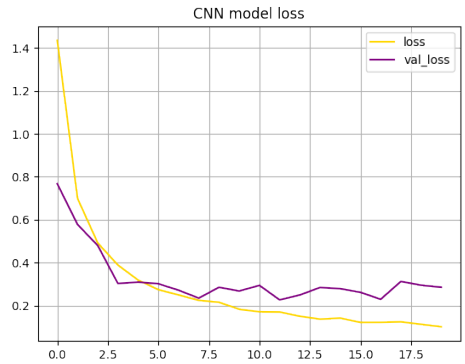
TAB. 4.6 : The parameters of the CNN model

Performances achieved and discussion

The model exhibited excellent performance in terms of accuracy and loss. It has shown high accuracy in recognizing the distinct spoken words and noise , namely "Up", "down", "right","left", "stop","go", "forward","backward" , as depicts in 4.11a and a a low loss as depicts in 4.11b The model also was evaluated on a test dataset and achieved a prediction accuracy of 94% to 97%.



(a) CNN Accuracy.



(b) CNN loss.

When analyzing the confusion matrix. It provided a comprehensive overview of the

model's performance across all 11 classes. As it is shown the model only missclassified : 12 in the class up and 29 in the class down and 25 in the classe go , 30 in the class stop, 5 in the class right,12 in the class left, 21 in the class on , 50 in the classe off , 2 in the class backward, 33 in the class forward and finally 6 in the class noise.
so we can say that there is a remarquable confusion between the classes : up and off, up and stop ,backward and on, down and go .

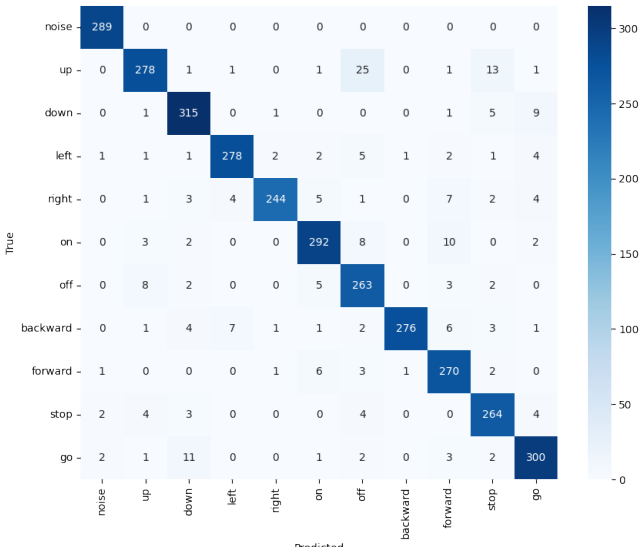


Fig. 4.5 : Confusion matrix

This analysis can be confirmed by the accuracy and recall values provided in Table 4.7. A higher precision value indicates that the model minimizes the number of false positives, while a higher recall value indicates that the model maximizes the number of true positives. To obtain a comprehensive view of the model's performance, we consider the F1 score. A higher F1 score indicates a more effective and performing model. so the model is very performing when it comes to the classes : noise, backward,down, forward,left,stop,go and it is less performing when it comes to the classes : up,off.
And based on the AUC-ROC score depicts in figure 4.6 we can say that this model is a performent classifier .

Classes	Precision (%)	Recall (%)	F1-score (%)	AUC-ROC score (%)
noise	97.97	100	98.97	100
up	93.29	86.60	89.82	99.32
down	92.10	94.88	93.47	99.80
left	95.86	93.29	94.56	99.72
right	97.99	90.04	93.85	99.77
on	93.29	92.11	92.7	99.71
off	84.02	92.93	88.25	99.45
backward	99.28	91.39	95.17	99.90
forward	89.11	95.07	91.99	99.66
stop	89.79	93.95	91.83	99.67
go	92.31	93.17	92.73	99.44
Average	93.18	93.04	93.03	99.68

TAB. 4.7 : The reults of Precision, Recall, F1-score, AUC-ROC score of the CNN model 01

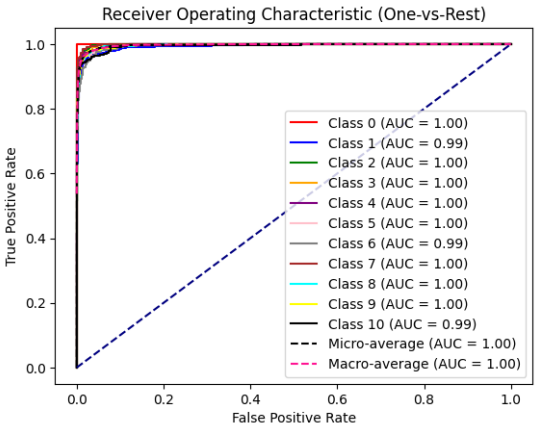


FIG. 4.6 : The AUC ROC score of the CNN model 01.

4.2.7 Solutions for the noisy environment

we will first describe the architecture of each model we used, after that we'll introduce the solutions which are : autoencoder with CNN model, SEGAN with CNN and we'll see the effect of noise on the CNN in each case

binary classifier model for noisy data data

It takes raw WAV files as input and processes them using the Short-Time Fourier Transform (STFT) magnitude to compute the spectrogram, the model is trained and tested using WAV files without additional noise, the model's output consists of predicted commands : "data","noise".

Architecture

in this work, we propose to develop our CNN architecture as shown in figure 4.7

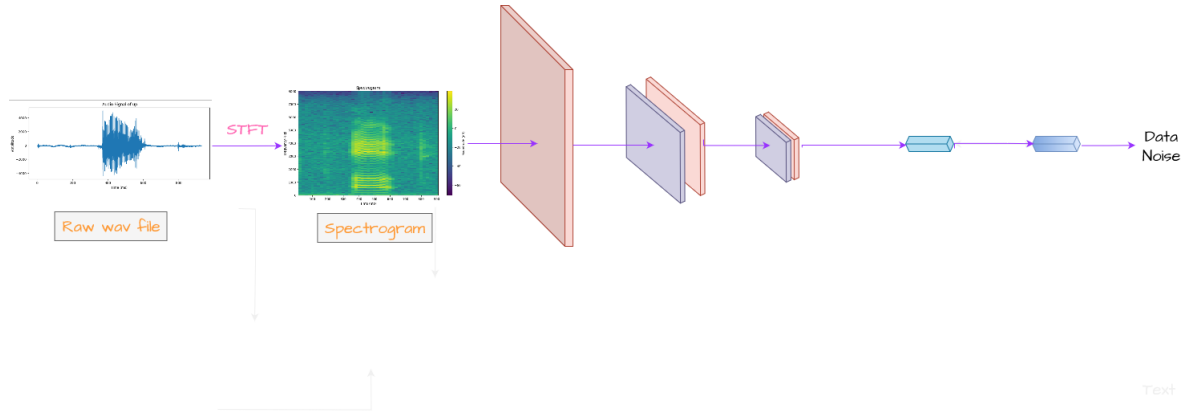


FIG. 4.7 : The architecture of thebinary classifier model.

The architecture consists of over 19,330 weights to be learned during the training process.It contains several layers. The input shape is (513, 33), indicating an input image with a height of 513 pixels, width of 33 pixels
Conv2D : Applies a 2D convolution operation with 16 filters, a kernel size of 3x3, and a stride of 1. The output shape is (None, 513, 33, 16).

MaxPooling2D : Performs max pooling with a pool size of 2x2, reducing the spatial dimensions by half. The output shape is (None, 256, 16, 16).

Conv2D : Another convolutional layer with 8 filters, a kernel size of 3x3, and a stride of 1.

MaxPooling2D : Another max pooling layer with a pool size of 2x2. The output shape is (None, 128, 8, 8).

Conv2D : A third convolutional layer with 8 filters, a kernel size of 3x3, and a stride of 1.

Flatten : Flattens the input, converting it into a 1D array. The output shape is (None, 8192).

Dense : Fully connected layer with 2 units, representing the output classes. It has 16,386 trainable parameters.
This architecture is summarized in Table 4.8.

Type	Forme	Parametrs
input	(None, 513, 33, 1)	0
Conv2D	(None, 513, 33, 16)	240
MaxPooling2D	(None, 256, 16, 16)	0
Conv2D	(None, 256, 16, 8)	1800
MaxPooling2D	(None, 128, 8, 8)	0
Conv2D	(None, 128, 8, 8)	904
Flatten	(None, 8192)	0
Dense	(None, 2)	16386
Total		19330

TAB. 4.8 : The architecture of the binary classifier.

Training

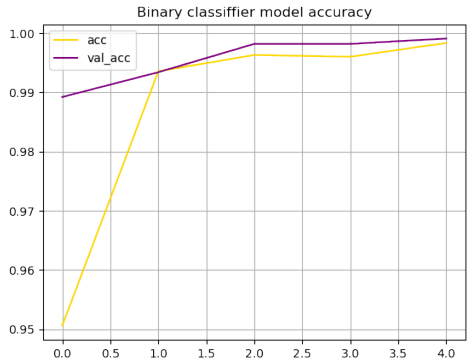
The input image size for this model is 513x33 with a single channel (1).As it is mentioned in the previous section .
- the Training was for 5 epochs.
The metrics : accuracy and loss and confusion matrix were recorded.
Adam (Stochastic Gradient Descent) optimizer was employed. A batch size of 128 was used for training .
the table 4.9 Summarize all the parameters used during training.

	Architecture
Training parameters	19330
Input image size	513x33
Number of convolutional layers	3
Number of max-pooling layers	2
Activation function	RELU
Learning rate	0.001
Epochs	5
Batch size	128
Optimizer	Adam
Number of parameters	19330

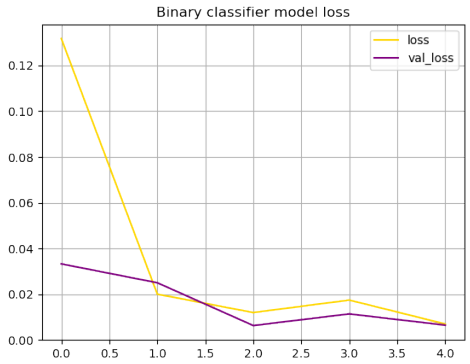
TAB. 4.9 : The parameters used in the binary classifier.

Performances achieved and discussion

The model shows excellent performance in terms of accuracy and loss. It has shown high accuracy in recognizing the distinct classes noise and data as depicts in 4.8a and a a low loss as depicts in 4.8b The model also was evaluated on a test dataset and achieved a prediction accuracy of 95% to 99%.



(a) Binary classifier Accuracy.



(b) Binary classifier loss.

When analyzing the confusion matrix shown in 4.9 It provided a comprehensive overview of the model's performance across the two classes. As it is shown the model only missclassified : only three times when distinguishing between the class data and the class noise.
which makes this model a very performent model that can easily distinguish between the two classes noise and data

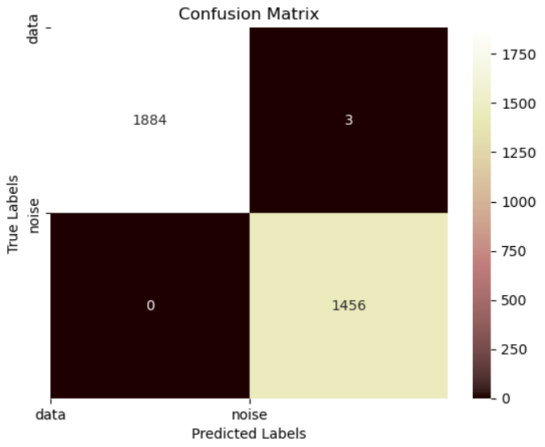


Fig. 4.9 : The confusion matrix of the binary classifier.

CNN model 02

It takes raw WAV files came from autoencoder or the SEGAN model, as input and generates the spectrogram, the model is trained and tested using WAV files, the model's output consists of predicted commands : "up","down", "right","left","on","off","stop","go","forward","back

Architecture

the CNN architecture as shown in figure 4.10

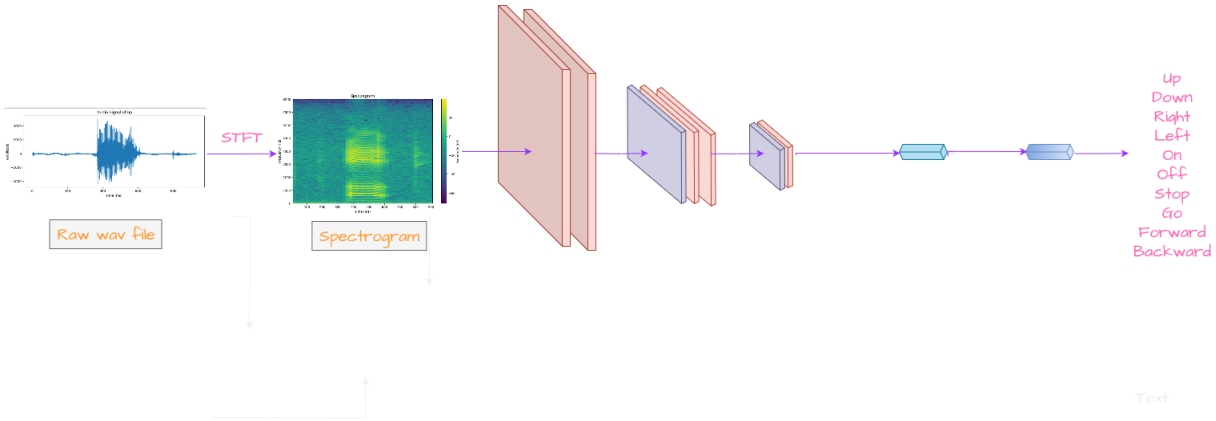


FIG. 4.10 : The architecture of the CNN model 02.

The architecture consists of over 803,818 weights to be learned during the training process.It contains several layers. The input shape is (516, 36), indicating an input image with a height of 516 pixels, width of 36 pixels and the output are the predicted 10 classes. This architecture is summarized in Table 4.10.

Type	Forme	Parametrs
input	(None, 516, 36, 1)	0
Conv2D	(None, 516, 36, 128)	1920
Conv2D	(None, 516, 36, 128)	229504
MaxPooling2D	(None, 258, 18, 128)	0
Conv2D	(None, 258, 18, 128)	114752
Conv2D	(None, 258, 18, 128)	57408
MaxPooling2D	(None, 129, 9, 64)	0
Conv2D	(None, 129, 9, 64)	28704
Flatten	(None, 37152)	0
Dense	(None, 10)	371530
Total		803,818

TAB. 4.10 : The architecture of the CNN model 02.

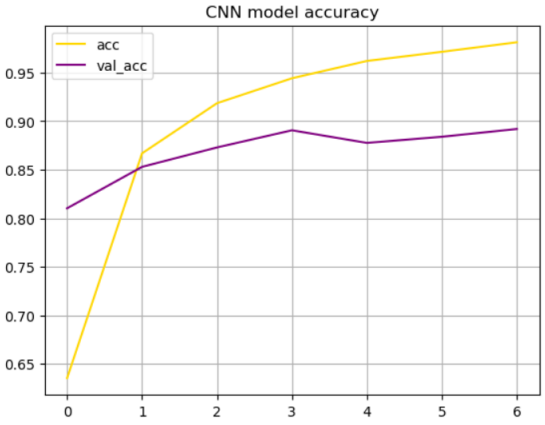
Training

The input image size for this model is 516x36 with a single channel (1).As it is mentioned in the previous section . - the Training was for 7 epochs.
The loss and accuracy were recorded.
Adam (Stochastic Gradient Descent) optimizer was employed.
A batch size of 64 was used for training .

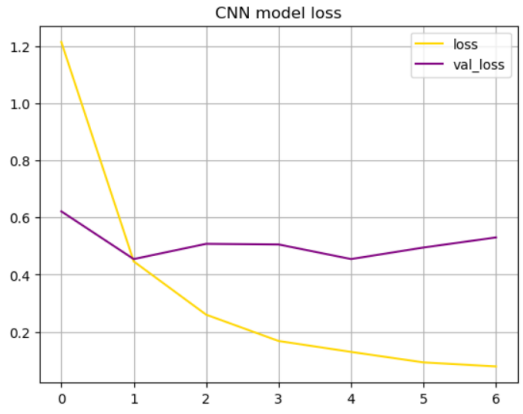
Performances achieved and discussion

As shown in figure 4.11a , the model has a good accuracy ranging from 60% to 98% and a very low value of loss 0.05 % as depicts in 4.11b The high accuracy implies that

the model is correctly classifying a large proportion of instances, indicating its ability to make accurate predictions. It shows that the model is learning the underlying patterns in the data and performing well in differentiating between classes. and the low loss value suggests that the model's predictions are close to the actual values. It signifies that the model has successfully minimized the discrepancy between predicted and actual values during the training process.



(a) CNN Accuracy.



(b) CNN loss.

This demonstrates that the model's effectiveness in accurately classifying instances and capturing the underlying patterns in the data.

To evaluate the robustness of the CNN model for the speech recognition, we did some experiments shown in figure 4.12 where we added different levels of noise to the clean audio data.This was accomplished by adding noise at frequencies ranging from 1000Hz to 5000Hz, and subsequently evaluating the CNN model's accuracy in recognizing ten spoken words.The findings revealed that the model's performance degraded as the level of noise increased, with the highest accuracy achieved at the lowest level of noise. The degradation in performance was particularly significant for high-frequency noise, with the model's accuracy dropping to as low as 20% for noise at 5000Hz. These outcomes underscore the limitations of the CNN model in effectively recognizing speech in noisy environments, thereby emphasizing the necessity for more advanced techniques such as the proposed SEGAN method presented in this study.

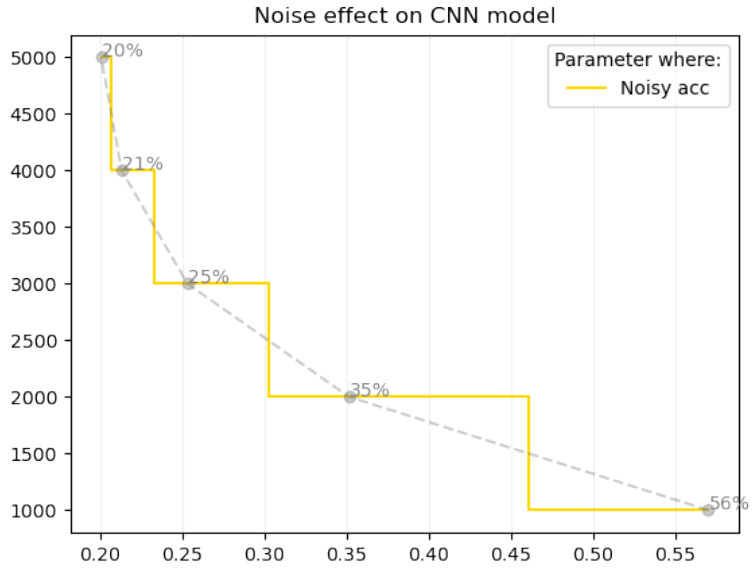


FIG. 4.12 : The effect of noise on CNN

Autoencoder model

The autoencoder used is described in section 2.3.2, has an architecture is shown in figure 2.15.
The architecture consists of over 4193 weights to be learned during the training process. This architecture is summarized in Table 4.5.

Type	Forme	Parametrs
input	(None, 516, 36, 1)	0
Conv2D	(None, 516, 36, 16)	160
MaxPooling2D	(None, 258, 18, 16)	0
Conv2D	(None, 258, 18, 8)	1160
MaxPooling2D	(None, 86, 6, 8)	0
Conv2D	(None, 86, 6, 8)	264
UpSampling2D	(None, 258, 18, 8)	0
Concatenate	(None, 258, 18, 16)	0
Conv2D	(None, 258, 18, 16)	2320
UpSampling2D	(None, 516, 36, 16)	0
Concatenate	(None, 516, 36, 32)	0
Conv2D	(None, 516, 36, 1)	289
Total		4193

TAB. 4.11 : The architecture of the autoencoder

Conv2D : This layer applies 16 filters to the input, resulting in an output shape of (None, 516, 36, 16). It has 160 parameters to learn.

MaxPooling2D : This layer performs max pooling operation with a pool size of 2x2, reducing the spatial dimensions by half. The output shape becomes (None, 258, 18, 16).

Conv2D : Another convolutional layer with 8 filters is applied to the previous layer's output, resulting in an output shape of (None, 258, 18, 8). It has 1160 parameters.

MaxPooling2D : Similar to the previous max pooling layer, this layer reduces the spatial dimensions further to (None, 86, 6, 8).

Conv2D : Another convolutional layer with 8 filters is applied, resulting in an output shape of (None, 86, 6, 8). It has 264 parameters.

UpSampling2D : This layer performs upsampling to increase the spatial dimensions back to (None, 258, 18, 8).

Concatenate : The output of the previous layer is concatenated with the output from the second convolutional layer, resulting in an output shape of (None, 258, 18, 16).

Conv2D : Another convolutional layer with 16 filters is applied, resulting in an output shape of (None, 258, 18, 16). It has 2320 parameters.

UpSampling2D : Upsampling is performed again to increase the spatial dimensions to (None, 516, 36, 16).

Concatenate : The output of the previous layer is concatenated with the output from the first convolutional layer, resulting in an output shape of (None, 516, 36, 32).

Conv2D : The final convolutional layer with 1 filter is applied, resulting in an output shape of (None, 516, 36, 1). It has 289 parameters.

SEGAN model

The SEGAN model architecture consists of two sub-models : an autoencoder described in the section 2.3.2 that serves as a generator to produce noise-free audio signals, and a discriminator model that we will describre its architecture in table 4.12 it uses clean data from our database. The autoencoder is responsible for generating audio signals without noise by learning to map the noisy signals to their corresponding clean signals. On the other hand, the discriminator is responsible for distinguishing between the clean audio signals and the noise-corrupted ones generated by the autoencoder. By combining the outputs of these two sub-models we get our SEGAN model .

Architecture

the SEGAN architecture as shown in figure of section 2.3.3 the architecture of the discriminator model :4.5.

Type	Forme	Parametrs
Conv2D	(None, 258, 18, 128)	1408
LeakyReLU	(None, 258, 18, 128)	0
Dropout	(None, 258, 18, 128)	0
Conv2D	(None, 129, 9, 64)	81984
LeakyReLU	(None, 129, 9, 64)	0
Dropout	(None, 129, 9, 64)	0
Conv2D	(None, 129, 9, 64)	20512
LeakyReLU	(None, 65, 5, 32)	0
Dropout	(None, 65, 5, 32)	0
Flatten	(None, 65, 5, 32)	0
Dense	(None, 10400)	10401
Total		114,305

Tab. 4.12 : The architecture of the discriminator

4.2.8 The first solution for the noisy environment : Autoencoder with CNN model 02

In an effort to ameliorate the CNN model's robustness in speech recognition, we explored the utilization of a basic autoencoder to eliminate noise from the audio data prior to its input into the CNN model. As shown in figure 4.13 Unexpectedly, we observed a significant decline in classification performance, with an approximate accuracy of 10% for 5000 hz of noise.We observed a decrease in classification score from 2% to 25% compared to using the CNN model alone on noisy data. This outcome suggests that the simple autoencoder alone may not be adequate in eliminating noise from speech data while preserving the essential information required for accurate speech recognition. It is possible that the encoding and decoding processes of the autoencoder are not optimized for speech data, or that more advanced denoising techniques are necessary for effective noise removal. Nonetheless, these findings underscore the importance of careful consideration when combining different techniques for speech recognition and the significance of evaluating the performance of each component individually and in combination.

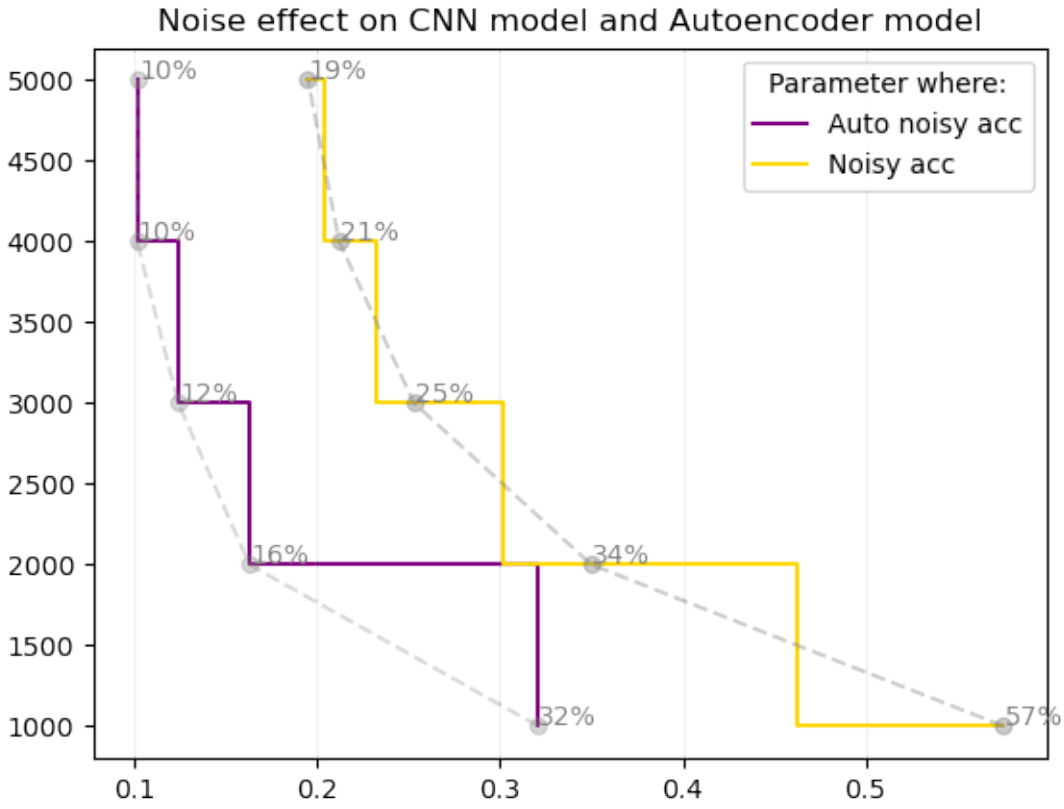


FIG. 4.13 : The effect of noise on CNN when using the autoencoder.

4.2.9 The second solution for the noisy environment : SEGAN with CNN model

In contrast to the simple autoencoder, our experiments revealed that the Spectrogram Generative Adversarial Network (SEGAN) exhibited remarkable effectiveness in enhancing the robusteness of the CNN model for speech recognition. By leveraging the SEGAN generator to generate clean speech signals from noisy inputs before feeding them into the CNN model, as shown in figure 4.14 we observed a substantial increase in the classification score. This improvement ranged from 2% to 23% when compared to using the CNN model alone on noisy data. The significant performance boost indicates that SEGAN's ability to generate high-quality, clean speech signals from noisy inputs can greatly enhance the CNN model's capability to recognize speech in adverse conditions. These findings highlight the potential of SEGAN and other generative adversarial network (GAN) techniques in improving the robustness of speech recognition systems. They also underscore the importance of exploring advanced approaches to address the limitations of traditional methods.

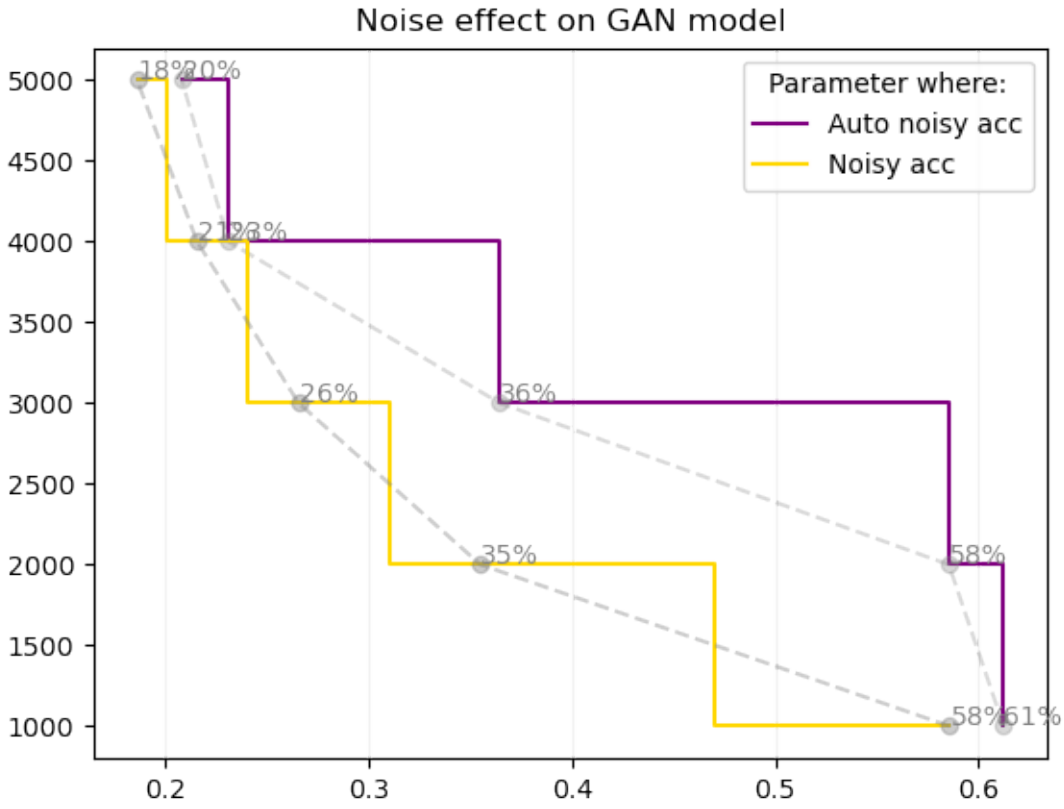


FIG. 4.14 : The effect of noise on CNN when using SEGAN

Summerizing discussion

Our experiments showed that in noiseless environment while using a CNN model, It gives us very good performances and high effectiveness.in noisy environment, the performance of the CNN model degraded as the level of noise increased. The autoencoder, which was very promising in theory,it actually was unable to effectively remove noise from speech data without sacrificing critical information needed for voice recognition. However, the SEGAN method showed great promise in enhancing the CNN model's robustness in recognizing speech in noisy environments, with an improvement in accuracy up to 23% compared to using the CNN model alone on noisy data. These results demonstrate the potential of advanced techniques such as SEGANs in addressing the limitations of traditional methods and improving the performance of speech recognition systems in real-world scenarios.

4.3 Quadrotor

In this section, we delve into the comprehensive exploration of the various aspects related to building and operating a quadrotor. This section provides an overview of the essential equipment and software employed in the quadrotor's construction, as well as the

configuration of its components. Additionally, we present a detailed description of the algorithm utilized for controlling the quadrotor's movements. To conclude, we evaluate the quadrotor's performance based on its responsiveness and motor speed balance.

4.3.1 Equipments

Raspberry pi card

Raspberry pi card was used, section 4.2.2 covers an in-depth explanation of the Raspberry Pi card.

Brushless dc motor

In our case, we used the A2212/13T 2200KV BLDC motor as shown in figure 4.15. The motor specifications are as follows :
- KV : 2200
- Max Efficiency : 80/- Maximum current : 12A / 60s
- Power : 239 W
- Number of Cells : 2-3 Li-Poly
- Motor Dimensions : 27.5 x 30mm
- Shaft Diameter : 3.17mm
- Weight : 49g

The velocity KV, it corresponds to the number of revolutions per minute per volt. The KV is obtained by dividing the number of revolutions per minute by the voltage KV = RPM / U. For example our motorwith a KV of 2200 rpm/V will operate at 26400 rpm if powered by 12 V.

Fig. 4.15 : The brushless dc motor.

Electronic Speed Controller

An electronic speed controller (ESC) is a component used in electric models operated by remote control most of the time, to manage and control the speed and direction of the motor.
For this experiment, 30A ESCs were employed to control each brushless motor, as depicted in figure 4.16 These ESCs are capable of delivering a constant and sufficient current to effectively drive the brushless motors. The ESCs possess the following specifications :

- Constant Current : 30A.
- Battery : 2-4 cells Lithium Polymer.
- Motor Type : Brushless.
- Size : 55mm x 26mm x 13mm.
- Weight : 32gms.

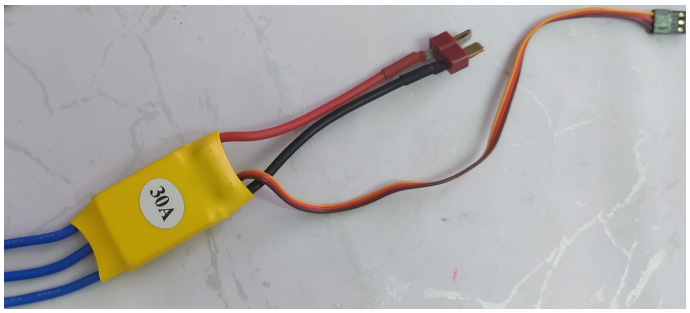


Fig. 4.16 : The electronic speed controller.

Propellers

The propellers used have 3 blades and they are made of plastic, as shown in figure 4.17 and has the following characteristics :

- dimension of a blade : 6cm x 1.5cm
- Weight : 9 grams

FIG. 4.17 : The propellers.

Lipo Battery

A single LiPo cell (1S) provides a voltage of 3.7 V. In our case , 3S battery was used. The capacity of a LiPo battery is expressed in mA/h, indicating the current it can deliver over one hour.In our experiment battery is a 3000 mA/h LiPo battery, so it can sustain a current of 3000 mA for one hour.

4.3.2 Software

Python on raspberry pi was used, Section 4.2.1 covers an in-depth explanation of Python.

Bitvise SSH Client

Bitvise SSH Client is a robust software tool that facilitates secure remote access and administration of servers. It establishes a secure communication between a client device, like a laptop, and a server, such as a Raspberry Pi, utilizing the SSH (Secure Shell) protocol.
The SSH (Secure Shell) protocol is a popular network protocol that creates a secure and encrypted connection between a client and a server.It provides a secure means for remote login, file transfer, and system administration.By employing robust encryption algorithms, SSH ensures the confidentiality and integrity of data.

pigpio

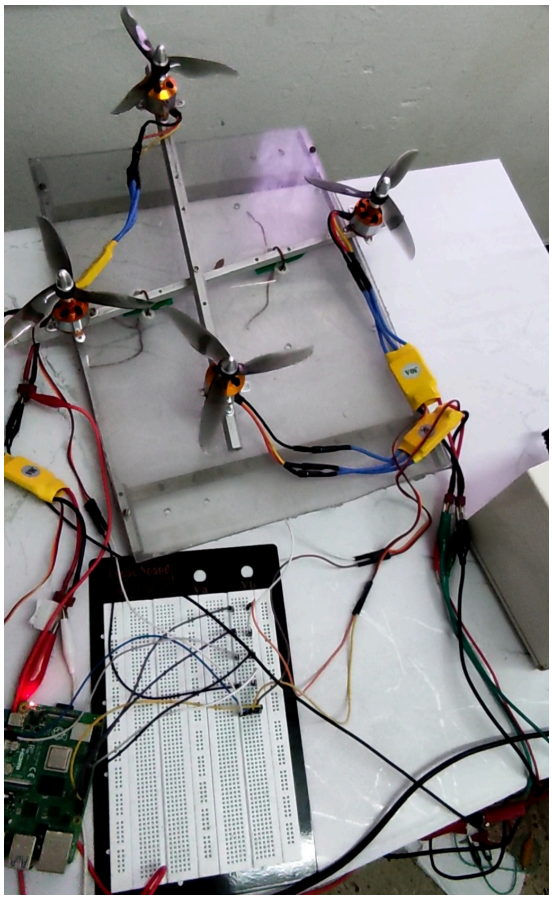
Pigpio is a Python library designed specifically for the Raspberry Pi, providing convenient access to its GPIO (General Purpose Input/Output) pins.This library enables users to effectively control the GPIO pins for a range of tasks,

threading

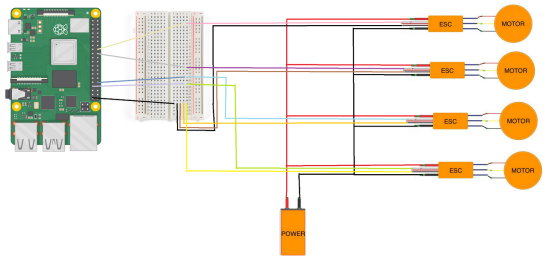
Threading is a Python library that enables the concurrent execution of multiple threads within a single process. Threads are lightweight and independent units of execution that enable programs to perform multiple tasks simultaneously. With the threading library, developers can create, manage, and synchronize threads, allowing for efficient and parallel execution of tasks in Python programs.

4.3.3 Assembly

The assembly of our project can be seen in Figure 4.18a, which depicts the actual physical arrangement. It showcases the integration of four motors, four ESCs, a Raspberry Pi card, and a LiPo battery. Additionally, Figure 4.19b presents the circuit diagram, illustrating the electrical connections and components used in our project.



(a) Assembly of the quadrotor.



(b) Circuit Diagram.

4.3.4 components configuration

After completing the assembly, the calibration of the ESCs, is necessary for the flight controller to read incoming signals.

ESC Calibration :

Most ESCs are configured from the manufacture to operate from 1000µs to 2000µs, calibration is required to ensure that all 4 motors start simultaneously. The following procedure is used :

1. Connect the ESCs to the Raspberry Pi GPIO pins, ensuring the proper wiring connections.
2. Upload the code for ESC calibration.
3. Start by sending a signal with a pulse width of 1000 microseconds to the ESCs to initialize them.
4. Set the pulse width to 2000 microseconds, allowing the ESCs to detect the maximum throttle point.
5. After reaching the maximum value, decrease the pulse width back to the minimum value which is 1000 microseconds, to complete the calibration process.

The ESCs at each step emits a series of beeps as an indication to signal successful calibration.

4.3.5 Algorithm Description

After importing the necessary libraries, and assigning motor pins to the pins of the raspberry pi, Here's an explanation of the code used : The variables max_value, min_value define the maximum and minimum pulse widths respectively for controlling the motor speed. They are used in the calibration process.

A function set_motor_direction sets the direction of a motor by configuring the GPIO pin as an output and setting the corresponding pin value.

A calibrate_motor function performs the calibration of a motor. It sets the GPIO pin as an output, sets the motor direction to forward, and changes the pulse width to calibrate the motor. The pi.set_servo_pulsewidth function is used to set the pulse width, and time.sleep is used to introduce delays for calibration.
The control_motors function controls the motors based on the given command.

A set_motor_speed function sets the speed and direction of a motor based on the given parameters. It calculates the appropriate pulse width based on the desired speed, sets the GPIO pin value for the direction, and uses pi.set_servo_pulsewidth to set the pulse width.

The control_thread function implements a thread for user input and motor control. It continuously prompts the user for a command and calls the control_motors function to control the motors accordingly.

```
calibrate_motor(motor1_pin)
time.sleep(2)
# Calibrate motor 2
calibrate_motor(motor2_pin)
time.sleep(2)
# Calibrate motor 3
calibrate_motor(motor3_pin)
time.sleep(2)
# Calibrate motor 4
calibrate_motor(motor4_pin)
time.sleep(2)
```

These lines of code perform the calibration process for each motor. The calibrate_motor function is called for each motor, followed by a pause with time.sleep to allow time for calibration.

```
thread = threading.Thread(target=control_thread)
thread.start()

try:
    thread.join()
```

```
except KeyboardInterrupt:
    pass

control_motors("stop")
pi.stop()
os.system("sudo killall pigpiod")
```

These lines of code start the control thread, wait for the thread to finish, stop the motors, and clean up the GPIO on program exit. The threading.Thread class is used to create a new thread with the control_thread function as the target. The thread is then started with thread.start(). The program waits for the user to press Ctrl+C to exit, at which point it stops the motors, cleans up the GPIO resources, and terminates the pigpiod daemon.

Performances

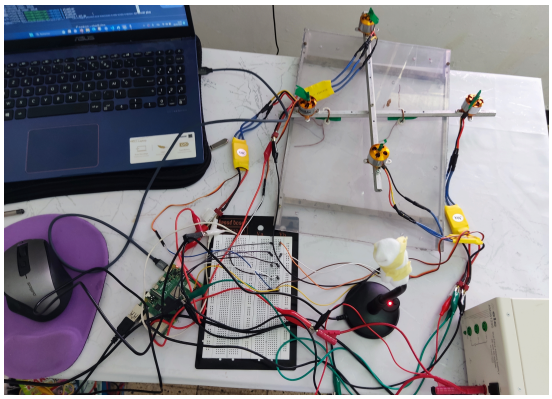
After executing the code, we can say we achieved responsive controls and balanced motor speeds by observing the following indicators :

1. Responsive Controls :
We've tested the quadrotor's response to control inputs, by applying changes in throttle, pitch, roll, and yaw commands (executing the commands up,down,right,left,on,off,stop,go) and assess if the quadrotor reacts promptly and accurately. and the quadrotor was reacting accurately without any noticeable delays or lag.

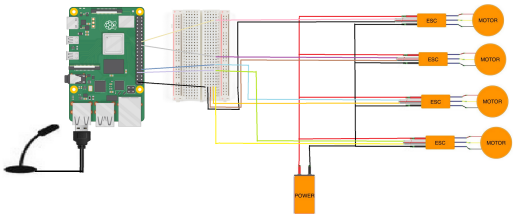
2. Balanced Motor Speeds :
We checked if the speeds of all four motors are similar and stable, by visually observing the quadrotor's behavior we can definetly say that the speeds of the motors were stable.

4.4 controlling quadrotor with keywordspotting

In Section 4.3, we presented the initial implementation of only the quadrotor code, which laid the foundation for its operation. In this section, we expand the previous code by relating the keywordspotting model with the quadrotor by introducing additional functions. the assembly of the quadrotor and the circuit diagram is shown in figure 4.19a and figure 4.19b



(a) Assembly of the quadrotor.



(b) Circuit Diagram.

After importing the necessary libraries, and assigning motor pins to the pins of the raspberry pi, Here's an explanation of the code used :

The TensorFlow Lite model for keyword spotting is then loaded . The labels list contains the class labels corresponding to the model's output.

The sample_rate is set to 16000 samples per second, and the duration is set to 1 second.
Next the variables max_value, min_value are defined and the function set_motor_direction is used as the previous code.
we used after that these functions : calibrate_motor, control_motors, set_motor_speed same as the previous code

```
def audio_callback(indata, frames, time, status):
    # Preprocessing steps
    # ...

    # Perform inference
    interpreter.set_tensor(input_details[0]['index'], spectrogram)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])

    # Process the output (e.g., determine the predicted class)
    predicted_class = np.argmax(output_data)
    predicted_command = labels[predicted_class]

    # Control the motors based on the predicted command using threads
    motor_thread = threading.Thread(target=control_motors,
    args=(predicted_command,))
    motor_thread.start()
    motor_thread.join(timeout=3)  # Wait for 3 seconds before
    #predicting the next command
```

The audio_callback function is called when new audio data is available. It performs preprocessing on the audio data, such as computing the spectrogram. Then, it passes the preprocessed data to the TensorFlow Lite interpreter for inference. The output of the model is processed to determine the predicted class (command). Based on the predicted command, a separate thread is created to control the motors using the control_motors function. The timeout argument in motor_thread.join ensures that the program waits for 3 seconds before predicting the next command.
Next we calibrate the 4 motors using the function calibrate_motor

```
stream = sd.InputStream(callback=audio_callback, channels=1,
samplerate=sample_rate, blocksize=sample_rate)
stream.start()

try:
    while True:
        continue
except

 KeyboardInterrupt:
    pass

stream.stop()
stream.close()
```

Here, an audio stream is created using the sd.InputStream class. The callback parameter is set to the audio_callback function, which will be called for each new audio input. The stream is then started, and a loop is used to continuously process the audio data. The loop runs until a keyboard interrupt (Ctrl+C) occurs, at which point the stream is stopped and closed.
After the audio stream is stopped, the control_motors function is called with the stop command to ensure the motors are stopped.

Performances

After executing the code, We achieved impressive results when applying the keyword spotting model to control a quadrotor, resulting in a highly efficient and responsive system. By leveraging the power of the keyword spotting model, we were able to accurately predict and execute commands on the quadrotor based on the recognized keywords.

The combination of the keyword spotting model and the quadrotor platform allowed for seamless interaction and control. The model effectively recognized specific keywords or commands from the input audio, enabling precise and reliable control of the quadrotor's movements and actions.

The performance of the quadrotor in executing commands based on the keyword spotting model was exceptional. The system demonstrated a remarkable level of responsiveness and accuracy, promptly responding to the recognized keywords and performing the corresponding actions with precision.

4.5 Conclusion

In conclusion, this chapter presented the development and implementation of a keyword spotting model for both noisy and noiseless environments. The primary objective was to leverage this model to control a quadrotor effectively and assess its performance in real-world scenarios.

First, a robust keyword spotting model was built, capable of accurately recognizing and classifying spoken commands in various acoustic conditions. The model was trained using a diverse dataset that encompassed both noisy and noiseless speech samples, ensuring its adaptability to different environments.

Next, the noiseless version of the keyword spotting model was integrated into the quadrotor system. The model's predictions were used to control the quadrotor's movements and actions based on the recognized keywords. Through thorough testing and evaluation, the performance of the quadrotor using the noiseless keyword spotting model was found to be highly satisfactory.

The results showcased the model's ability to accurately identify spoken commands. The quadrotor demonstrated remarkable responsiveness and precision in executing the recognized keywords, indicating the successful integration of the keyword spotting model into the real-world quadrotor platform.

These achievements highlight the potential of keyword spotting models in enhancing the control and automation capabilities of robotic systems. By effectively recognizing and interpreting spoken commands, such models can enable intuitive and seamless human-machine interaction in various applications.

Conclusion and perspectives

Our work was a response to the question asked in the general introduction, so in this work we were interested in establishing in english a keyword spotting system in order to use it as an application to control UAV movements.

To achieve this, we first described the interaction between humans and drones to really understand this type of interaction, because to master something you have to understand the details and the basics. Then we moved on to the building of this recognition system in noisy and noiseless environments. Good simulation results were obtained.after that we built and operated a quadrotor using input command, and then we implemented the keyword spotting model to control the quadrotor, and we had very performant system.

In the context of more research, it would be interesting to build a multilingual keyword spotting model because it offers several advantages over one-lingual keyword spotting, it allows for greater flexibility and adaptability in diverse language environments and it facilitates internationalization and globalization efforts. we can also Optimize the keyword spotting system to achieve real-time performance, ensuring quick and responsive control of the quadrotor. This may involve optimizing the computational efficiency of the algorithms used or exploring hardware acceleration techniques to speed up the processing time, and another way to ameliorate the system is to combine keyword spotting with gesture recognition or natural language processing, to enable more intuitive and expressive control of the quadrotor.

Bibliography

1. MEBARKIA, Nihad ; KACEL, Yasmine. *Multilingual voice recognition using deep learning for human-drone interaction*. 2021. Mém. de mast. Blida 01 University.
2. KARJALAINEN, K. ; ROMELL, A. *Human-Drone Interaction : Drone as a companion ? An explorative study between Sweden and Japan*. 2017. Mém. de mast.
4. TEZZA, D. ; ANDUJAR, M. The State-of-the-Art of Human–Drone Interaction : A Survey. *IEEE Access*. 2019, t. 7, p. 167438-167454.
5. FERNANDEZ, R.A.S. ; SANCHEZ, J.L. ; SAMPEDRO, C. et al. Natural user interfaces for human-drone multi-modal interaction. In : *2016 International Conference on Unmanned Aircraft Systems*. IEEE, 2016, p. 1013-1022.
6. YAM-VIRAMONTES, B. A. ; MERCADO-RAVELL, D. Implementation of a Natural User Interface to Command a Drone. In : *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2020.
7. SUÁREZ FERNÁNDEZ, Ramón A. ; SANCHEZ-LOPEZ, Jose Luis ; SAMPEDRO, Carlos ; BAVLE, Hriday ; MOLINA, Martin ; CAMPOY, Pascual. Natural User Interfaces for Human-Drone Multi-Modal Interaction. *IEEE Xplore*. 2016.
8. HERDEL, Viviane ; YAMIN, Lee J ; CAUCHARD, Jessica R. Above and beyond : A scoping review of domains and applications for human-drone interaction. In : *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, p. 1-22.
9. LA DELFA, Joseph ; BAYTAS, Mehmet Aydin ; PATIBANDA, Rakesh ; NGARI, Hazel ; KHOT, Rohit Ashok ; MUELLER, Florian'Floyd'. Drone chi : Somaesthetic human-drone interaction. In : *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, p. 1-13.
10. MCCULLOCH, Warren S. ; PITTS, Walter. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*. 1943, t. 5, n° 4, p. 115-133.
11. NENNOUCHE, Mohamed ; ATCHI, Abdel Malek. *Fusion de caractéristiques pour la classification des différents niveaux de démence de la maladie d'Alzheimer*. 2022. Mém. de mast. Ecole nationale polytechnique.
12. HEBB, Donald Olding. *The organization of behavior : A neuropsychological theory*. Psychology Press, 2005.

13. ROSENBLATT, Frank. The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*. 1958, t. 65, nº 6, p. 386.

14. RUMELHART, David E. ; DURBIN, Richard ; GOLDEN, Richard ; CHAUVIN, Yves. Backpropagation : The Basic Theory. In : *Backpropagation : Theory, Architectures, and Applications*. 1995, p. 1-34.

15. HOPFIELD, John J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*. 1982, t. 79, nº 8, p. 2554-2558.

16. LECUN, Yann ; BOTTOU, Léon ; BENGIO, Yoshua ; HAFFNER, Patrick. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*. 1998, t. 86, nº 11, p. 2278-2324.

17. SAHA, Sumit. *A Comprehensive Guide to Convolutional Neural Networks — The ELI5 Way*. 2022. Accessed on 04/22/2022.

19. LI, Lujun ; LU, Zhenxing ; WATZEL, Tobias ; KÜRZINGER, Ludwig ; RIGOLL, Gerhard. Light-Weight Self-Attention Augmented Generative Adversarial Networks for Speech Enhancement. *Electronics*. 2021, t. 10. Disp. à l'adr. DOI : https://doi.org/DOI.

20. MOHAMEDI, F. ; SACI, N. *Simulation d'un drone sous MATLAB : Cas d'étude - Quad-copter*. 2016. Mém. de mast. UNIVERSITE Abderrahmane Mira de Béjaïa.

21. YEDAMALE, Padmaraja. *Microchip Technology Inc Brushless DC (BLDC) Motor Fundamentals*. Rapp. tech., AN885. Microchip Technology Inc.

22. GAMAZO-REAL, José Carlos ; VÁZQUEZ-SÁNCHEZ, Ernesto ; GÓMEZ-GIL, Jaime. Position and Speed Control of Brushless DC Motors Using Sensorless Techniques and Application Trends. *Sensors*. 2010, t. 10, nº 7, p. 6901-6947. ISSN 1424-8220. Disp. à l'adr. DOI : 10.3390/s100706901.

24. WARDEN, P. Speech Commands : A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints*. 2018. Disp. à l'adr. arXiv : 1804.03209 [cs.CL].

Webography

3. DIGITALIST MAGAZINE. *Are Drones Changing the Way We Live ?* [Website]. 2019. URL : https ://www.digitalistmag.com/digital-economy/2019/11/05/are-drones-changing-way-we-live-06201367/.

18. DEEPAK BIRLA. *Autoencoders* [Medium]. Year. URL : https ://medium.com/@birla.deepak26/au 76bb49ae6a8f.

23. WIKIPEDIA. *Keras* [Wikipedia]. Year. Aussi disponible à l'adresse : https://en.wikipedia.org/wiki/Keras.