National Technical University of Athens
School of Mechanical Engineering

Development of a voice command recognition model based on artificial neural networks.

Diploma Thesis

Anna Maria Iatridi

Supervisor Professor Panorios Benardos

Athens 2024

Abstract

In this thesis was studied the development of a voice recognition model based on artificial neural networks for industrial applications. More specifically, the case study is the robotic arm Staubli RX 90L located at the Manufacturing Technology laboratory, in National Technical University of Athens. The neural network is designed to recognize single-word commands and translate them into written text, for manipulation of the robotic arm. The development of the robot interface is outside of the thesis-scope, but the aim is the model to be able to collaborate with Staubli RX 90L in the future and therefore the commands used are relevant to the robot's action.

In the first step of the process, the vocabulary for recognition is decided based on the V^+ language commands the robot understands. To achieve that, the foundation was to study V^+ language and its most important and basic keywords and commands. For the audio dataset, a part of Google's Speech Commands Dataset was used. This dataset contains single-word commands from a representative sample of human. The relevant commands are the digits from 0 to 9 and short words, like "*on*", "*off*", "*stop*" among others.

The pre-processing of the signals is done to be able to extract the characteristic features to identify the spoken word. The pre-processing, is used to remove background noise form the data and balance the frequency spectrum. The correct pre-processing is the one resembling human's hearing ability. Pre-emphasis, windowing and Fast Fourier transform, are few of the key parameters for speech recognition.

The feature extraction phase is the most vital for successful recognition. The Mel-Frequency Cepstral Coefficients (MFCCs) method is used to "normalize" the frequencies to the scale that the human ear perceives them. The 12 MFCCs are characteristic of the input signal and include the most important information. The coefficients are used to classify the unknown recording to one of the known classes.

For the classification, a pattern recognition, artificial neural network (ANN) is used. To select the most suitable model, the ANN parameters have been investigated, architecture, training function etc. The final network structure is 600x450 neurons in hidden layer one and two respectively. The final model recognizes in total 18 spoken commands with accuracy 82%. There are still improvements to be done, but the main goals of the thesis have been achieved and the results show that with the proper optimization ANNs are a competitive and relatively simple method for voice commands recognition.

Περίληψη

Στην παρούσα εργασία μελετάται η ανάπτυξη μοντέλου αναγνώρισης φωνητικών εντολών με χρήση τεχνητών νευρωνικών δικτύων, για βιομηχανικές εφαρμογές. Η βιομηχανική εφαρμογή είναι ο ρομποτικός βραχίονας Staubli RX 90L που βρίσκεται στο εργαστήριο του Τομέα Κατεργασιών στο Εθνικό Μετσόβιο Πολυτεχνείο. Το νευρωνικό δίκτυο σχεδιάστηκε για να αναγνωρίζει μονολεκτικές φωνητικές εντολές και να τις μετατρέπει σε γραπτό κείμενο, με σκοπό τον προγραμματισμό του ρομπότ. Η ανάπτυξη του μοντέλου διεπαφής υπολογιστή-ρομπότ είναι εκτός του φάσματος της εργασίας, όμως η προοπτική συνεργασίας του μοντέλου με το ρομπότ καθορίζει σε μεγάλο βαθμό τις προδιαγραφές του ίδιου του μοντέλου.

Το πρώτο στάδιο της εργασίας είναι ο καθορισμός του λεξιλογίου προς αναγνώριση, βασισμένο στις εντολές της προγραμματιατικής γλώσσας V+, την οποία καταλαβαίνει το ρομπότ. Η γλώσσα V+ αναπτύχθηκε από την εταιρεία Adept Technologies και σχεδιαστικές κατά κύριο λόγο για βιομηχανικές εφαρμογές ρομποτικών συστημάτων. Είναι κατάλληλη για τον έλεγχο ρομπότ κίνησης, για συστήματα βιομηχανικής όρασης και για διεργασίες εισόδου-εξόδου. Όσον αφορά την εκμάθηση, είναι μία πολύ εύκολη γλώσσα, με γρήγορο self-compilation και αποδοτική χρήση μνήμης. Κύριο χαρακτηριστικό της είναι η ευστάθεια, που την καθιστά ιδανική για την διαχείριση απαιτητικών ρομποτικών ενεργειών. Η V+ είναι απαραίτητη σε κατασκευαστικές βιομηχανίες και αυτοκινητοβιομηχανίες και προσφέρει δυνατότητες μοντέρνων γλωσσών προγραμματισμού.

Το λεξιλόγιο της εφαρμογής καθορίζεται από τις εντολές της V+ και από την διαθεσιμότητα εντολών από το Google’s Speech Commands Dataset. Η βιβλιοθήκη εντολών, περιέχει διάφορες εντολές στην αγγλική γλώσσα, όπως τα αριθμητικά ψηφία από το μηδέν μέχρι το εννιά και άλλες μικρές και απλές λέξεις, όπως “on”, “off”, “stop” κ.α. Το λεξιλόγιο που χρησιμοποιήθηκε στην εργασία συμπεριλαμβάνεται στον παρακάτω πίνακα.

Table 1 – Λεξιλόγια εργασίας

Reference Number	Command	Function
0	*“Zero”*	Αριθμητική τιμή.
1	*“One”*	Αριθμητική τιμή.
2	*“Two”*	Αριθμητική τιμή.
3	*“Three”*	Αριθμητική τιμή.
4	*“Four”*	Αριθμητική τιμή.
5	*“Five”*	Αριθμητική τιμή.
6	*“Six”*	Αριθμητική τιμή.
7	*“Seven”*	Αριθμητική τιμή.
8	*“Eight”*	Αριθμητική τιμή.
9	*“Nine”*	Αριθμητική τιμή.
10	*“Up”*	Άμεσο άνοιγμα αρπάγης.
11	*“Down”*	Άμεσο κλείσιμο αρπάγης.
12	*“Left”*	Αρνητικό πρόσημο.
13	*“Right”*	Θετικό πρόσημο.
14	*“Stop”*	Ακύρωση τρέχουσας διεργασίας.
15	*“On”*	Αρχή προγράμματος.
16	*“Off”*	Τέλος προγράμματος.
17	*“Go”*	Παύση τρέχουσας διεργασίας.

Για να γίνει η αναγνώριση των εντολών είναι απαραίτητη η σωστή προεπεξεργασία των ηχητικών σημάτων και η εξαγωγή των χαρακτηριστικών μαθηματικών παραμέτρων, των οποίων ο συνδυασμός οδηγεί στην αναγνώριση της εντολής. Η προετοιμασία του σήματος, πριν την αναγνώριση, προσομοιώνει το τρόπο πρόσληψης και ανάλυσης των ηχητικών σημάτων του ανθρώπινου εγκεφάλου. Η προεπεξεργασία περιέχει πρώτον και κύριον το στάδιο της αποθορυβοποίησης, όπου χρησιμοποιούνται φίλτρα pre-emphasis για να καθαρίσουν το σήματα από περιττή και άχρηστη πληροφορία. Το φίλτρο αυτό αποτελεί μία μαθηματική συνάρτηση υπολογισμού της διαφοράς διαδοχικών σημείων του σήματος με έναν συντελεστή. Η συνάρτηση αυτή μειώνει την συνολική ένταση των σημάτων, λειτουργώντας σαν ένα είδος κανονικοποίησης. Με αυτόν τον τρόπο τα σήματα έχουν περισσότερη ομοιογένεια.

Την αποθορυβοποίηση διαδέχεται ο διαχωρισμός του σήματος σε επιμέρους τμήματα, πριν το στάδιο υπολογισμού του φάσματος. Ο λόγος για τον κατακερματισμό του σήματος είναι ότι ο υπολογισμός του φάσματος συχνοτήτων στο σύνολο του σήματος χάνει πληροφορία για την χρονική εξάρτηση της συχνότητας. Αντίθετα ο υπολογισμός του φάσματος σε μικρότερα τμήματα του σήματος, διασφαλίζει την χρονική εξάρτηση της συχνότητας, ως πληροφορία που θα συμβάλλει στην αναγνώριση των εντολών. Η διάσπαση του σήματος σε μικρότερα δημιουργεί ασυνέχειες, οι οποίες οδηγούν σε διαρροές φάσματος. Η διαρροή φάσματος είναι όταν εμφανίζονται συχνότητες, καθ όλο το εύρος, οι οποίες δεν αντιστοιχούν σε πραγματική πληροφορία, αλλά σε ασυνέχειες. Η απαλοιφή των ασυνεχειών έρχεται σε σύγκρουση με την διακριτότητα του σήματος. Η συνάρτηση Hamming window, εξασφαλίζει απουσία διαρροών και ταυτόχρονα καλή διακριτότητα. Τώρα το σήμα είναι έτοιμο για την εφαρμογή του διακριτού μετασχηματισμού Fourier (DFT).

Η εξαγωγή των φασματικών συντελεστών της κλίματας Mel (MFCCs), αποτελεί το πιο καίριο βήμα για την αναγνώριση εντολών. Αρχικά, το σήμα μετασχηματίζεται από την κλίμακα συχνοτήτων στην κλίμακα των Mel. Η κλίμακα Mel είναι μια αντιληπτική κλίμακα συχνοτήτων με ισαπέχοντα διαστήματα συχνοτήτων που αντιλαμβάνονται ως ισαπέχουσες απ' το ανθρώπινο αυτί. Ο άνθρωπος δεν έχει την ίδια ευαισθησία σε όλες τις συχνότητες¨στις χαμηλές μπορεί και αναγνωρίζει πολύ εύκολα ακόμα και πολύ μικρές μεταβολές, ενώ στις υψηλότερες η αντιληπτικοτητα του μειώνεται καιτ διαφορετικές συχνότητες τις αντιλαμβάνεται ως ίδιες ή παρεμφερείς. Για κάθε ένα από τα τμήματα, υπολογίζονται οι 12 φασματικοί συντελεστές. Οι φασματικοί συντελεστές λειτουργούν ως ταυτότητα των διαφορετικών φωνημάτων και καθιστούν δυνατή την διαφοροποίηση των ηχητικών λέξεων. Αυτοί αποτελούν την είσοδο του νευρωνικού δικτύου, για την κατηγοριοποίηση άγνωστων εντολές, σε γνωστές κλάσεις.

Για την αναγνώριση των φωνητικών εντολών, γίνεται χρήση τεχνητών νευρωνικών δικτύων αναγνώρισης μοτίβων. Η επιλογή κατάλληλου μοντέλου μηχανικής μάθησης είναι καίρια για την επιτυχημένη αναγνώριση των εντολών. Κατά την εκπόνηση της διπλωματικής δόθηκε μεγάλη έμφαση στην εύρεσης της βέλτιστης αρχιτεκτονικής νευρωνικού δικτύου, προς την επίτευξη της μέγιστης απόδοσης. Το τελικό νευρωνικό δίκτυο επιλέχθηκε με 600 κρυμμένους νευρώνες στο πρώτο επίπεδο και 450 στο δεύτερο. Η σύγκριση πολυπλοκοτερο αρχιτεκτονικών δεν κρίθηκε απαραίτητη, λλά θα αποτελούσε ενδιαφέρουσα διερεύνηση. Το τελικό μοντέλο αναγνωρίζει 18 φωνητικές εντολές με ακρίβεια 80%, υπό προϋποθέσεις. Η μέγιστη ακρίβεια εμφανίζεται όταν οι άγνωστες, προ αναγνώριση, λέξεις ανήκουν στο σύνολο Google's Speech Commands Dataset. Οι εντολές που δίνονται από ανεξάρτητους ομιλητές αναγνωρίζεται με ακρίβεια κοντα στο 60%. Αυτό δείχνει σημάδια υπερ-εκπαίδευσης και αδυναμία γενίκευσης προβλέψεων.

Σε κάθε περίπτωση, η χρήση τεχνητών νευρωνικών δικτύων θεωρείται ανταγωνιστική μέθοδος στο κομμάτι της αναγνώρισης εντολών και με μικρές διορθώσεις μπορεί να φτάσει καλύτερες επιδόσεις.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor assistant professor Panorios Benardos, for the great collaboration we have and for trusting me with this interesting topic, which gave me the opportunity to deepen my knowledge in machine learning and improve my engineering mindset. His guidance and patience were vital throughout the whole project.

I would also like to thank my friends and colleagues in university, Christos, Stathis and Anastasia for their continuous support and love since day one. Of course, a big thanks to “my partners in crime”, my teammates Stefania and Kostis, who made the university projects a good learning and a fun process. I am grateful to have shared this journey with you!

Last but not least, I want to thank my parents and sister, for their unconditional love and for always pushing me beyond my limits and supporting my dreams.

Anna Maria Iatridi
Athens, July 2024

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα επίκουρο καθηγητή Πανώριο Μπενάρδο για την εξαιρετική συνεργασία μας και που μου εμπιστεύτηκε ένα τόσο ενδιαφέρον θέμα, που υπήρξε η αφορμή να εμβαθύνω τις γνώσεις μου στο πεδίο της μηχανικής μάθησης και να βελτιώσω τον τρόπο σκέψης μου ως μηχανικός. Η καθοδήγηση και η υπομονή του υπήρξαν καθοριστικές, καθ' όλη την διάρκεια της εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τους φίλους και συμφοιτητές μου Χρήστο, Στάθη και Αναστασία για την συνεχόμενη στήριξη και αγάπη τους απ' την πρώτη μέρα. Φυσικά, δεν θα μπορούσα να παραλείψω τους «συνεργούς μου στο έγκλημα» Κωστή και Στεφανία που έκαναν τις εργασίες της σχολής παραγωγικές και ευχάριστες. Είμαι ευγνώμων που μοιραστήκαμε μαζί αυτό το ταξίδι.

Τέλος, ένα τεράστιο ευχαριστώ στους γονείς και την αδερφή μου για την ανιδιοτελή αγάπη τους και που πάντα με σπρώχνουν πέρα απ' τα όριά μου και στηρίζουν τα όνειρά μου.

Άννα Μαρία Ιατρίδη
Αθήνα, Ιούλιος 2024

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Thesis Scope

In this thesis was studied the development of a voice recognition model based on artificial neural networks for industrial applications. The study focuses on the implementation of machine learning techniques for single-word speech recognition. The commands were chosen for an industrial robotic arm application, as the aim is that the model can be used to manipulate the robot to perform basic tasks, like approach an object, grab it in with the end-effector etc. The case study robot is Staubli RX 90L, an articulated robotic arm with six degrees of freedom (DoF), located at the Manufacturing Technology laboratory in National Technical University of Athens (NTUA). These robots are usually used for welding, surfacing and pick-and-place tasks in many industrial applications. The thesis workflow is described in the picture below.

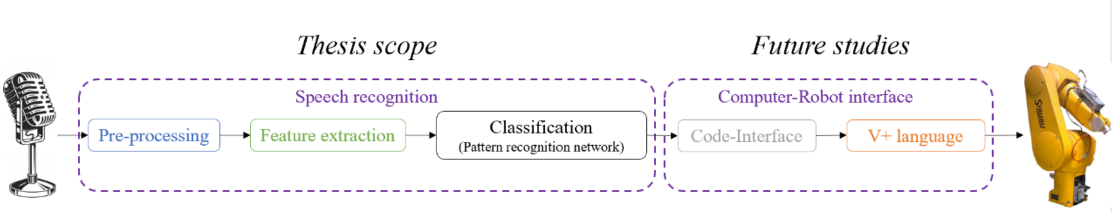


Figure 1 – Thesis flow chart

The role of the voice recognition object is to translate spoken words into written text. The system's input is the audio signal, recorded in real time by the user, and the system's output the corresponding command. The voice commands are recorded with ordinary microphone (phone, computer), so no special equipment is required, and then processed through filters and functions to extract the Mel Frequency Cepstral Coefficients (MFCCs). MFCCs contain the most important information of the audio signal, and are used as input for an Artificial Neural Network (ANN), which will correlate the recordings with the corresponding commands.

Briefly the methodology is:

- Signal pre-processing:
  - Noise removal and frequency balancing
  - Signal segmentation into frames
  - Power spectrum calculation
- Feature extraction:
  - MFCCs calculation
- Classification:
  - Optimization of a pattern recognition ANN

The acceptance criteria the development of a model with accuracy higher than 80%. Nowadays, the state-of-the-art speech recognition models can reach errors of 5%-10%. A custom model of 80% performance is a good starting point, which has still some areas of improvement and finetuning.

1.2. Main Challenges

During the development of the model there are many things to take into account that require a lot of attention. The key, for the system to work successfully, is the correct pre-processing of audio signals. It is important to identify and isolate the information of the voice command that is distinctive for this specific command, so the recognition is possible and effective. If the signal contains unnecessary sounds and noise or is missing specific characteristics, the accuracy of the machine learning model is limited. It is crucial to find which parameters are distinguishing the words from one another and find the mathematical portions to describe them. Only with right processing the investigation and development of the machine learning model is valid. It should be considered, also, that an audio signal contains a lot of information, for example regarding the speaker's identity, the intensity of the voice, potential sentiment condition and the sound of the phoneme. In this application, the object of interest is the word that is pronounced and not any other details. Therefore, the result must be insensitive to other characteristics and disregard this extra information. It's good to mention, that humans have the ability to understand and analyze many different characteristics when listening to others, when AI can generally perform one task at a time.

A challenge of speech recognition, is the variety and non-uniformity of human a speech and thus of the recordings. People speak in different speeds, with different accents and voice characteristics. These variations require a large representative dataset, so the model will be able to identify the word, regardless of ither factors. In single-word voice recognition, where the word is pronounced within a pre-defined time frame, the audio signals can vary a lot. The word can be placed differently in that time framed and occupy smaller or bigger part of the total duration, depending on the speech speed and the moment the user starts speaking. When the signal is divided into frames the complexity intensifies, since the corresponding frames deviate a lot from one another in different examples. The network should be trained to recognize patterns between and within the frames to reach to the right conclusion.

For a prediction model to work effectively, except the suitable method, it is crucial to have a good dataset. Good dataset consists mainly of two things: representative observations and sufficient amount of data. The requirement for representative observations, is obvious considering all the above. A poor dataset would make the final predictions very sensitive to details and would lose the ability for generalization. The amount of data needed, depends on the problem's nature and the number of inputs. Speech recognition is a very complex problem and the distinction between different words relies on the speech details. Additionally, the network's inputs are many, considering that, for each observation, 12 MFCCs are extracted per frame. It is important to secure a good dataset for the smooth and efficient operations of the system.

2. Literature Review

2.1. Machine Learning

2.1.1. Introduction

Artificial Intelligence (AI) is the ability of computers and machines to simulate human capabilities and intelligence to solve problems and perform task [1]. Machine Learning (ML) is the implementation of AI-driven techniques into applications, to imitate how humans learn from data and produce more data. ML is only a part of the big spectrum that is called AI. A subset of ML is Deep Learning (DL). DL is the application of ML techniques with higher complexity used to extract progressively higher-level features.

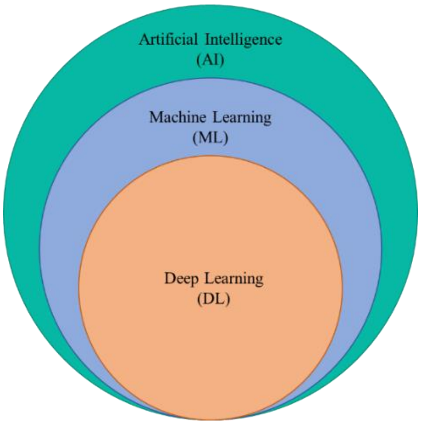


Figure 2– Artificial intelligence, machine learning and deep learning

It's a very common mistake to confuse the concept of these three terms, so it's very important to distinguish them properly. AI is a wider term that refers to projects of developing systems reinforced with human intellectual abilities. Machine learning and deep learning are subcategories of artificial intelligence and are distinguished based on the learning method. Both algorithms use neural networks as a learning technique from the data set [2]. The main difference relies on the type of network and the training process. Neural networks consist of interconnected nodes that transfer information similarly to biological neurons. The artificial neurons are split in different layers, the input layer, the hidden layers and the output layer. Each neuron is connected to others and has its own threshold and weight. The threshold controls whether a neuron is activated or not, if it's activated it passes the data to the next level, if not no data are transferring forward. In all cases the networks are trained to predict what the data represents.

Typical machine learning techniques are limited to supervised learning, meaning that human expertise is necessary to categorize the data and assign labels. Deep learning procedure is based on unsupervised training, so the objects extract features from large scale unlabeled data [2]. Additionally, the deep learning models consist of higher complexity of layers and connections. It's very important, in order to be able to evaluate an AI technique, to be able to evaluate the results and understand the explanation of the output. Rule of thumb, to increase the accuracy, complexity increases as well and human understanding decreases. The selection of an AI technique is often a trade-off between accuracy and control over the results

2.1.2. History

The history of artificial intelligence and machine learning starts after the 1940s decade and continues until today, where it is more relevant than ever. The term "machine learning" was introduced by Arthur Samuel in 1959 in his paper "Some Studies in Machine Learning Using the Game of Checkers" published in IBM Journal of Research and Development [3]. He used the game checkers to support that computer can be programmed to improve its own performance by analyzing previous games. Samuel defines machine learning as the field of study where computers have ability to learn from past experience. His work in the field of computer gaming and artificial intelligence started earlier in 1952, when he created Checkers-Playing Program", the first self-learning program to play games [4]. He developed a program to calculate the winning possibility in checkers for both players and explained that a computer can outplay the programmer, with the right programming.

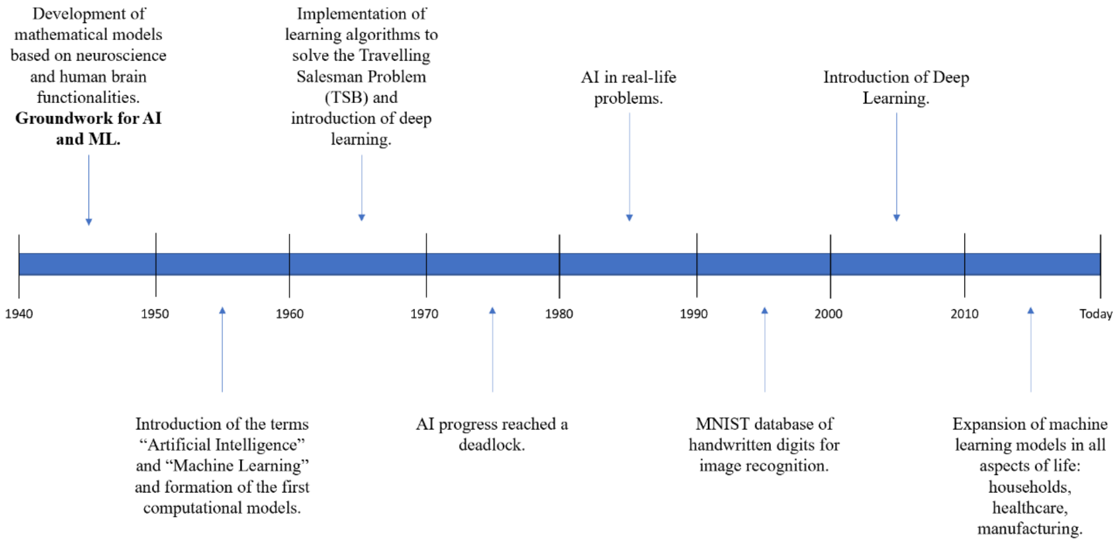

Development of mathematical models based on neuroscience and human brain functionalities. **Groundwork for AI and ML.**
Implementation of learning algorithms to solve the Travelling Salesman Problem (TSB) and introduction of deep learning.
AI in real-life problems.
Introduction of Deep Learning.
1940
1950
1960
1970
1980
1990
2000
2010
Today
Introduction of the terms "Artificial Intelligence" and "Machine Learning" and formation of the first computational models.
AI progress reached a deadlock.
MNIST database of handwritten digits for image recognition.
Expansion of machine learning models in all aspects of life: households, healthcare, manufacturing.

Figure 3– The history of machine learning

The first approach to machine learning was in early 40s, with the article "A logical calculus of the ideas immanent in nervous activity" by Walter Pitts and Warren McCulloch in 1943 [5]. They developed a mathematical model of neural network based on the human thought process. A similar study was published by Donald Hebb in 1949, "The Organization of Behavior: A neuropsychological Theory", introducing the Hebbian theory, focusing on machine learning based on human brain activity and behavior [6]. During this decade started the groundwork of AI and the introduction of mathematical models mimicking the brain functionalities. Later, in 1950, Alan Turing sets the foundation of AI with his work "Computing Machinery and Intelligence", with the controversial question: “Can a machine think?” [7]. One year later, in 1951, Marvin Minskey and Dean Edmonds trained the first ANN, using 3000 vacuum tubes to simulate a network of 40 neurons. The biggest milestone of the decade was the Dartmouth Summer Research Project on Artificial Intelligence, John McCarthy, Marvin Minsky, Claude Shannon and Nathaniel Rochester gathered some of the leading personalities in AI and computer science to investigate about the future steps. It was there, when the term artificial intelligence was firstly defined. In 1959 Arthur Samuel introduced the definition of machine learning [3]. In the next decade, 1960s, the usage of machine learning algorithms escalated in many scientific fields and problems, like solving of the Travelling Salesman Problem (TSP) with the nearest neighbor algorithm and the foundation of Deep Learning (DL) were set. In 1966, the first ever chatbot, Eliza, was created, a computer program with human characteristics capable of engaging in conversations. Eliza kicked-off the first generation of chatbots with simple recognition capabilities and although they had big restrictions regarding their input data, they showed that AI can soon be part of real life.

In the 70s, the progress reached a deadlock, due to limited amount of data and computational power. In parallel, many governments terminated the fundings on AI projects. Despite the difficulties, pattern recognition models continued to evolve slowly. In the coming years, the study and usage of machine learning attracted the interest and engagement of an increasing number scientists, but yet, machine learning was not a part of the reality. This was soon to be changed, when LeCun with Bengio and Haffner in 1989 demonstrated how neural networks can be applied to real-world problems, by sharing a convolutional neural network that could recognize handwritten characters [8]. Ten years later, LeCun continued his work and released the Modified National Institute of Standards and Technology (MNIST) database, a huge dataset of handwritten digits, which was widely adopted as a handwriting recognition evaluation benchmark and as a base for image recognition.

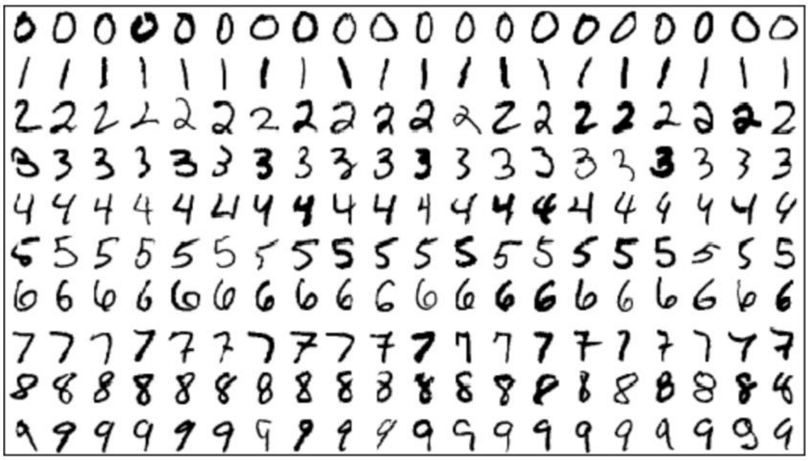


Figure 4– Modified National Institute of Standards and Technology (MNIST) database [9]

During the early 2000s several key milestones took place. The rise of the big data enabled the processing of large amount of data in real-time. The key attributes of big data are volume, the amount of data generated and collected in petabytes and more, velocity, the real-time data generation and processing speed and variety, the different type of data available, like texts, images, etc. This change helped overcoming important obstacles and leading to evolution of various methods. Speech recognition and natural language processing (NLP) became more adept and laid the foundations for virtual assistants as Siri. The development of convolutional networks enabled the improvements in classification and image recognition tasks.

Nowadays, machine learning is more relevant than ever and continues to evolve, offering potential across industries. Deep learning and neural networks applied for speech recognition, computer vision and autonomous systems are extensively used in healthcare, entertainment and other industries. In healthcare, deep learning improves diagnostic accuracy and provides personalized treatment in crucial conditions as cancer. In autonomous systems, ML thrives with autonomous drones and vehicles which rely on computer vision for vehicle navigation and pedestrian protection. Machine learning exceeds the scientific and industrial application and becomes one of the main means in everyday life. In household machinery, smart home devices, as voice assistants and thermostats, ramp up the comfort and home quality, robotic vacuum cleaners make the everyday tasks easier and smart ovens offer extra safety and assist in cooking process. Computers, smartphones and TVs have implemented AI techniques such as voice recognition for user identification, image recognition for face identification and object recognition, and all these for the overall improvement of user experience and extension of the device's capabilities. At the same, chatbots, such as ChaGPT, are capable of having full human-like conversations in real-time and have access to huge database of information and are gradually used more and more for professional and personal tasks.

2.1.3. Classification models

The machine learning prediction problems are categorized in regression and classification problems. A regression model predicts a quantity from dataset of continuous real values, by using independent inputs. The prediction must have minimum error from the target value. Regression analysis is a statistical method to analyze data and make predictions by understanding the relationships between variables and outputs. The regression can be linear and non-linear. The main metrics to evaluate a regression model are the accuracy, the mean squared error (MSE), mean absolute error (MAE) and R-squared (R^2) scores. Examples of regression problems, across different domains, are the prediction of a disease progress in medical applications, the estimation of housing marketing value, weather predictions and others. In manufacturing and production regression models can predict material and surface quality after surface processing and treatment like milling, turning.

Table 2 – Classification vs. regression problems

	Classification Model	Regression Model
Target variables	Discrete	Continuous
Desired output	Decision boundary to separate categories	Best fit trend to the dataset
Evaluation metrics	Accuracy Precision Recall F1 scores	Accuracy Mean squared error Mean absolute error R-squared scores
Problem type	Binary/Multi-class	Linear/Non-linear

A classification model predicts the category/label of the data, from a dataset of discrete values. The model should identify a decision boundary in order to separate the data to the categories. In classification problems, the model should identify trends and dominant characteristics in each category to classify data to the corresponding label. The main metrics to evaluate a classification model are the accuracy, the precision, the recall and F1-scores. The key components for classification are:

- Features: the input variables that the model categorizes,
- Labels: the output variables that the model predicts,
- Train set: the dataset used to train the model,
- Test set: the dataset used to check the accuracy of the model.

Image classification, speech recognition, face recognition and sentiment analysis are the main ML applications of classification problems. In everyday life, user identification in smart devices, via voice and face recognition, in healthcare, medical diagnosis based on patient's history and test results and in autonomous driving, vehicle navigation.

2.1.3.1. History

The history of classification models follows closely the progress of ML and statistical analysis. In the 1950s, the first classification algorithms, Linear Discriminant Analysis (LDA) and Logistic Regression, were used for binary problems. Later on, in the 1960s, the first decision trees were developed. These models split the feature space into subsets and use a tree-like model for decisions and their consequences, with conditional control statements. Neural network development dominants in the 1980s, with Multi-layer Perceptions (MLPs) and Backpropagation algorithms. MLP is a feedforward network of fully connected neurons and non-linear activation function. Backpropagation is a model using optimization algorithm, to train neural network. The first phase, called forward pass, the information is transferred through the network layers until the output layers gives the prediction. Then, the prediction is compared with the target value and error values and the gradients of loss are generated. The second phase, named backward pass, consists of transferring the loss gradients backwards through the layers and the weights are adjusted in the direction to reduce the loss, with the gradient descent optimization. In the 1990s, the decision tree algorithm is developed to improve accuracy and robustness, into random forest algorithm. In modern era, from 2000s till present, machine learning has been evolved to deep learning and the simple artificial neural networks to convolutional and recurrent neural networks. These networks have a significant performance improvement in image, speech and text classification. The main challenge in this new era is the lack of interpretability, due to the increasing complexity.

2.2. Speech recognition

2.2.1. Introduction

Speech recognition is a field of computer science that develops algorithms and models for recognition and translation of spoken language into written text by computer machines. It combines knowledge and research from other scientific fields, as computer science, computer engineering, linguistics, acoustics and neuroscience. The automatic speech recognition (ASR) resembles the human understanding of natural language. The models mimic the human ear functionality, with filters and transform functions, and the brain activities, with neural networks. The inverse process, the production of human speech, is called speech synthesis. The Speech recognition can be used in many applications, such as virtual assistants (Apple Siri, Amazon Alexa and Google Assistant), voice search and speech to text services, language translation, gaming and education.

Natural Language Processing (NLP) is a subfield of speech recognition, which enables computers to interact with humans through the natural language. With NLP computer are able to both understand and produce physical language and contribute in actual conversation. NLP simplifies everyday life with hands-free communication on smart devices and with computer assistance in customer services. NLP combines computational linguistics machine learning algorithms. Computational linguistics is a data science discipline for speech analysis. NLP is a lot more complicated than single word classification, since it has the additional difficulty of separating the words and identifying the most important words for analysis, by ignoring the ones like “the”, “and” etc. which don’t add meaning.

2.2.2. History

Historically, the speech recognition evolution follows the learning process of human beings, from simple single word understanding, like babies, until skillful handling of complex sentences and ability to answer challenging questions [10]. The first ever speech recognition model was AUDREY, the Automatic Digit Recognizer, in 1952. AUDREY could recognize digits from zero to nine with 90% accuracy, if they were given by his inventor [11]. Through the 1950s and 1960s many similar machines have been developed in laboratories around the world. One of the most important pioneers in continuous speech recognition is Raj Reddy. He was the first who researched the problem of continuous speech, where the users didn’t need to pause between different words. During this period there were two different approaches for speech recognition. The first one was using pre-recorded template waveforms, after morphing them to match the talking speeds, for comparison with the unknown signal. The second school was based on complex rules from linguistic knowledge, to guess the unknown signal. In 1962, at the Seattle World’s Fair (aka. Century 21 Exposition), IBM (International Business Machines Corporation) introduced to the world “Shoebox” machine, an improved version of AUDREY, which, additionally to the ten digits, could recognize sixteen English words [10]. In 1970s, speech recognition considered a lot of interest around the globe, mainly because of the fundings from the U.S. Department of Defense. The Speech Understanding Research (SUR) program, DARPA, was funding researches for five years with the goal of developing a model to recognize one thousand words by 1967. As part of this program, the first model was Hearsey-I, a system with spoken language as input and written text as output and was applied on chess tasks, due to the syntax structure. The winner was Harpy, a system which could understand 1011 words and introduced a more efficient search method called Beam Search. Both models were developed by Raj Reddy’s PhD students.

In the 1980-decade, various new methods marked a milestone in speech recognition, like the Hidden Markov Models (HMMs). The breakthrough of the method was the consideration of unknown sounds as potential words, rather than just the match of sound patterns between existing data and new inputs. By the end of this decade, speech recognition could be used by not only scientists, with the Worlds of Wonder's Julie doll, a doll which children could train to understand their voices. Despite the big progress, until then, all models were able to understand only single words, but not continuous speech, so the user should stop after each word. A second limitation of these models was that they understood mainly their inventor or speakers they have been already trained on. It was another of Reddy's students, Kai-Fu Lee, who combined the beam search with HMM to create SPHINX-I, the first system with speaker independence. In the 1990s, computers with faster processors made speech recognition available for consumers. Dragon, in 1998, launches the first product "Dragon Naturally-Speaking", which could understand natural speech with hundred words per minute, but needed training for an hour.

Until the early 2000s, speech recognition has reached accuracy over 80%, but models were still struggling with statistical methods to guess between similar-sounding words. Speech recognition gets a big boost with Google Voice Action in Androids and later on Google Voice Search on iPhones. Google was able to use the computational power of Cloud and the large data volume saved there, to identify user's speech. In 2010, Google implemented "Personalized Recognition", which means the user voice search was saved to produce better results. That way Google could constantly enrich its database.

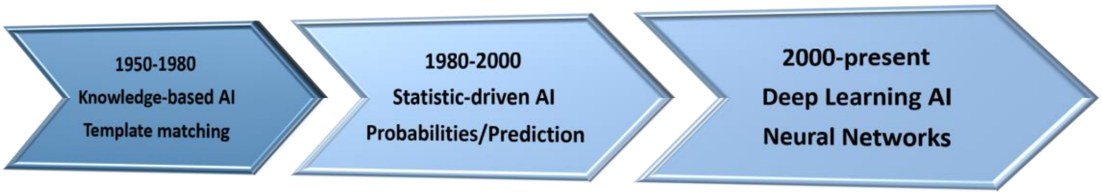


Figure 5– Speech recognition journey

In a nutshell, speech recognition passed through three main phases. From 1950 till 1980 is the era of knowledge-based AI and speech recognition relies on the matching with existing templates. From 1980 until 2000 is time for statistic driven AI, where statistical, probabilistic methods are implemented in the models to predict unknown signals as words. From 2000 and onwards, Deep Learning AI dominates and speech recognition is combined with deep and convolutional neural networks.

2.3. Acoustics

Psychoacoustics is the branch of acoustics and psychophysics involving the scientific study of sound perception and audiology—how human auditory system perceives various sounds, like speech and music. Based on the compression methods are used for reducing signal's size without decreasing sound quality and are called psychoacoustic methods.

The human auditory system can perceive frequencies between 2 Hz and 2 kHz and produce sounds of frequencies between 85 Hz and 155 Hz, for male adults, and between 165 Hz and 255 Hz, for female adults. For the frequencies in the hearing spectrum the "Absolute Threshold of Hearing—ATH" (or "Threshold of quiet") is defined and it refers to the minimum level of amplitude of a tone, that can be detected by normal hearing, assuming no other interfering sounds are present. [12] That way frequencies with lower amplitude cannot be distinguished and so can be removed from the original signal. The mathematic equation [12], which describes ATH is the following:

$$T_q(f) = 3.64\cdot\left(\frac{f}{1000}\right)^{-0.8} - 6.5\cdot e^{-0.6\cdot\left(f/_{1000}-3.3\right)^{2}} + 10^{-3}\cdot\left(\frac{f}{1000}\right)^{4}$$

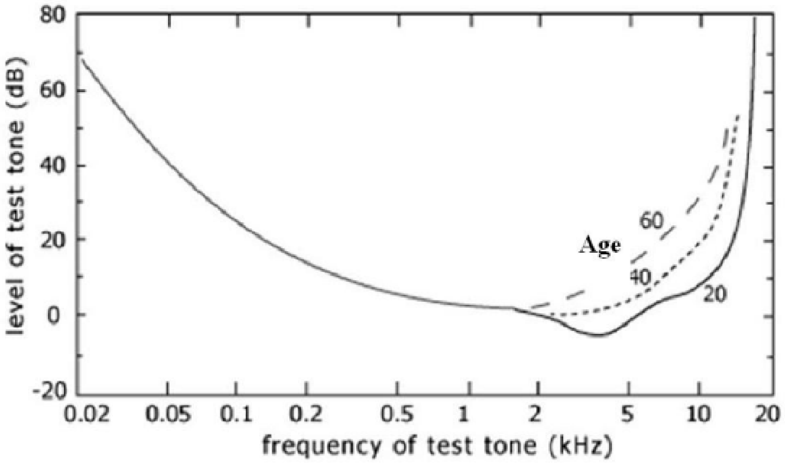


Figure 6– Absolute threshold of hearing (ATH)

As shown in Figure 6, the threshold of quiet is affected by the age, especially in frequencies higher than 2 kHz.

Humans, for biologic and evolution reasons, are more sensible in the middle frequencies, the spectrum where the human speech lies. The ear can hear and recognize these frequencies better and distinguish them from nearby ones, even when they exist in lower amplitudes. In very high or very low frequencies this ability fades and, therefore, neighbor frequencies can be perceived as the same. That is also connected with the absolute threshold of hearing, as seen in Figure 6Figure 8, high ($> 10\ kHz$) and low frequencies ($< 50\ Hz$) must be in higher levels. Another factor that affects the perception of sounds is the "Auditory masking", which in the frequency domain is called simultaneous masking, frequency masking or spectral masking and in the time domain is called temporal or non-simultaneous masking. This phenomenon occurs with the presence of multiple sound sources that affect and compromise the ear perception, and affects the ATH. Now, in order a frequency to be distinguished by humans the level of tone should be even higher. Similar to ATH, a new threshold is defined: "Masked threshold" is the quietest level of the signal perceived when combined with a specific masking sound. In Figure 7Figure 8 there are two examples of auditory masking at 410 Hz and 100 Hz respectively.

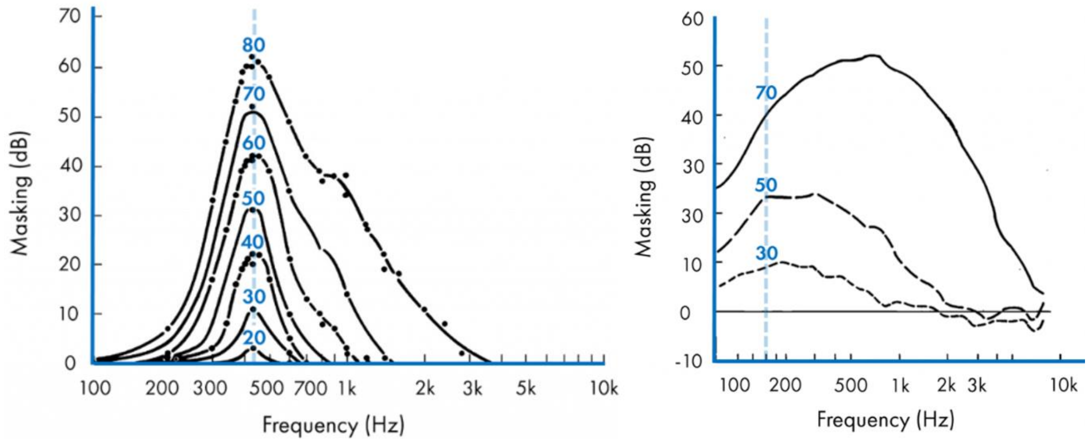


Figure 7– Auditory masking

a. 410 Hz

b. 100 Hz

The figures above show how the masking threshold changes over frequency for different amplitudes of the masking frequency. The maximum level of masking occurs when the two sounds have same frequency and reduces when moving away. This is called on-frequency masking and happens because the two signals belong to the same auditory filter and are perceived as equals from the ear. When the level of the masking frequency increases the range of masked frequencies becomes wider and the masked threshold increases as well. In higher frequencies the curve is steeper and the maximum masking occurs at the masking frequency. In lower frequencies the curve is smoother and wider and the maximum value is located in frequencies higher than the masking one. With the decrease of the masking frequency, the curve becomes asymmetric and covers larger area towards higher than lower frequencies.

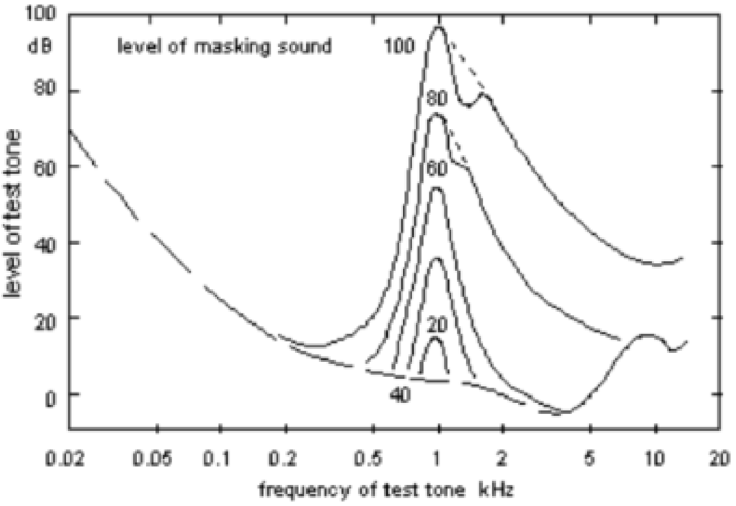


Figure 8– Threshold in quiet and masked threshold

3. Industrial robotic arms

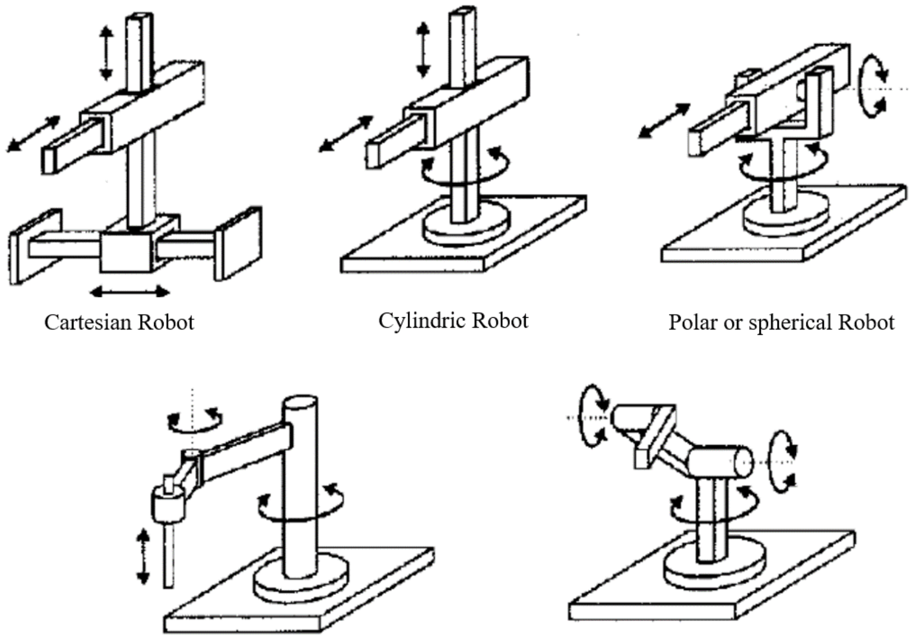
An industrial robot is a robot system used for manufacturing processes, it is automated, programmable and capable of movement on three or more axes. A commonly used type of industrial robots is the robotic arm, a mechanical arm with similar functions to a human arm. Robotic arms can be individual mechanisms or part of a more complex one. They consist of links connected by joints (usually 2-6) of rotational motion or linear displacement. Industrial robotic arms are used for several manufacturing applications such as:

- Assembly and dispensing (assembly and adhesive dispensing robots)
- Handing and picking (material handling, liquid handling, pick and place, and order picking robots)
- Machining and cutting (machine tending and loading, milling, drilling, cutting etc.)
- Welding and soldering
- Inspection and quality control etc. [13]

Robotic arms, are categorized based on their design, use and functions:

- Articulated robot arm
- Cartesian robot arm
- Cylindrical robot arm
- Delta robot arm
- Polar or spherical robot arm
- Selective Compliance Assembly Robot Arm or Selective Compliance the Articulated Robot Arm (SCARA) [14]

In Figure 9 are the schematic representations of joint movement for each robot type.



Cartesian Robot

Cylindric Robot

Polar or spherical Robot

SCARA

Angular or anthropomorphic Robot [1]

Figure 9 – Industrial robotic arm types [15]

Articulated robot arms

[1] *Angular or anthropomorphic robot is same as articulated robot.*

An articulated robotic arm resembles the human arm, is the most common type of robot arms and it consists of a single mechanical arm attached to a base with twisting joint. They are considered to be one of the most versatile and flexible tools and normally they have four to six axes, with six to be the most commonly used. They are suitable for automating many robotic applications, including assembly, material handling, arc and spot welding, painting and many more. They are known for their extensive range of motion, high precision, linear reach and because of their numerous axis points they can reach virtually everywhere within their workspace. Their most important drawback is the limitation in performing at higher speeds.

Cartesian robot arms

Cartesian robot arms, linear or gantry robots work in three linear axes, using the Cartesian coordinate system (x, y and z), so they move in straight lines: up and down, in and out and side to side. The three joints are manipulated to spatial movements, giving extra flexibility to cover most of the space. Additionally, cartesian robots give to user the ability to adjust the speed, precision, stroke length, and size of the robot arm. One of the disadvantages is that they require the most space compare with all other robotic arms. Their variety of tasks, mainly in small applications, includes pick-and-place work, operating machine equipment, arc welding and assembly tasks and are often used for CNC machines and 3D printing.

Cylindrical robot arms

Cylindrical robots are designed around a single-arm base, capable to move up and down vertically. This type consists of a rotary shaft and an extendable arm that support sliding and vertical displacement. In their base there is a rotary joint (1 rotational DoF) and between the links a prismatic one (1 translational DoF). The combination of mechanism complexity with lack of significant advantages, makes them the less preferable choice. Typical applications of cylindrical robots are assembly, machine tending, or coating.

Delta robot arms

Delta robot arm or parallel robot arm is a type of parallel robot with a triangular base and interconnected arms, attached to a central end effector. Delta robots can move in all three dimensions, with precise movements at high speeds, and are commonly used for automation in manufacturing, packaging and assembly. Their unique shape allows the three arms to control every joint of the end effector, making them a great fit for food, pharmaceutical and electronic industries.

Polar robot arms

Polar robots are from the first industrial robots created. Their mechanism is a combination of one linear joint (1 translational DoF), at the base, with two rotary joints (1 rotary DoF each), leading to a spherical work envelop. Key applications are die casting, injection moulding and material handling.

SCARA

SCARA robots are a special type of articulated robots, that have rotational joints. They are mechanically compliant in x- and y-axis and rigid in z-axis. Compared to articulated robots, SCARAs are less flexible, since they have fewer axes and so their motion is more limited. They perform better than cartesian robots in lateral motions with higher speeds, which they maintain even with high loads. Their strongest advantage is their position repeatability.

3.1. Staubli RX 90L

The industrial robot Staubli RX 90L is an articulated robotic arm with six axes, manufactured by Stäubli. Each of the six joints works as an axis around which two members rotate. The movements of them are generated by brushless motors coupled to resolvers and are equipped with parking brakes. The robot consists of motors, brakes, motion transmission mechanisms, cables, pneumatic and electric circuits both for the user and the counterbalance system. The balance is maintained by an integrated spring system, a build-in spring counterbalance. Data for the absolute position are provided by a counting system at any time. The assembly id reliable and robust, flexible and able to perform various tasks. The robot is used mainly for surfacing in many industrial applications, such as plastic and metal engine parts, bikes, agricultural equipment etc. The key components, inspired by human arms, are the base (A), the shoulder (B), the arm (C), the elbow (D), the forearm (E) and the wrist (F), as shown in Figure 10.

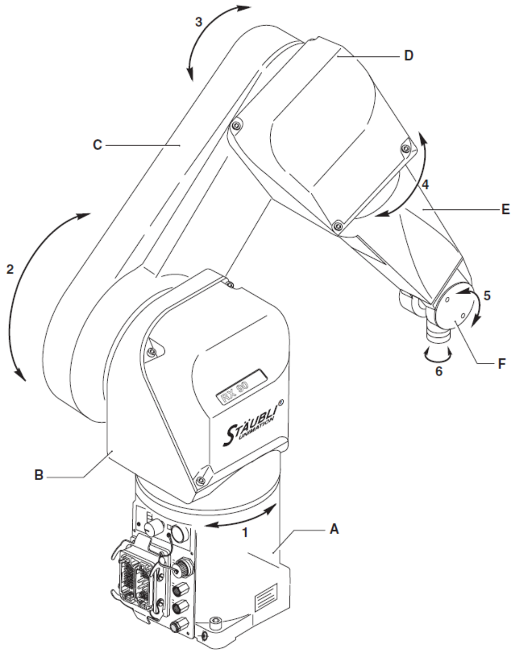


Figure 10 – Staubli RX 90L

The specifications of Staubli are summarized in Table 3. All the technical characteristics and numerical values presented below are given by Staubli in the manual [16].

Table 3 – Staubli RX 90L specifications

Designation: RX 90 B L	Robot family	RX (changed to B)
	Maximum reach between 2nd and 5th axis[2]	9 dm
	Number of active axis (≡ DoF)	0 ≡ 6 (variation with 5 axis)
	Forearm version	Extended forearm (L)
General characteristics	Working temperature	+5℃ to +40℃
	Humidity	30% to 90%
	Altitude	2000 m
	Weight	113 kg
Performance	Maximum speed at load center of gravity	12.6 $^{m}/_{s}$
	Repeatability	$\pm 0.025\ mm$
Load capacity	At nominal speed	3.5 kg
	At reduced speed	6 kg

[2] *That is the reach for the original - not extended – version. With the longer forearm it becomes* $11\ dm$ *instead of* $9\ dm$.

Below (Figure 11) it's the drawing of the robotic arm, with all important dimensions.

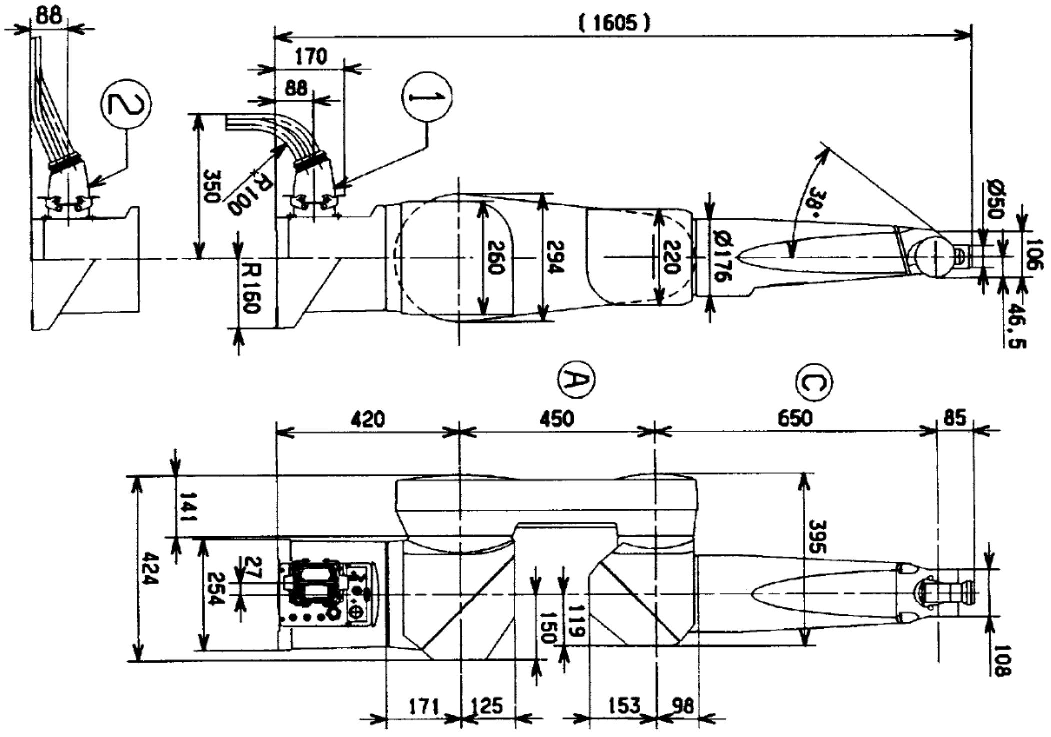


Figure 11 – Staubli RX 90L drawings

Work envelop is defined as the workspace, where the end effector, of the robotic arm, can reach with any orientation, and it depends on the dimensions of the components. The parameters that define the workspace are: the maximum reach between the 2nd and the 5th joints, which controls the maximum reach the end effector in x-z plane, the minimum reach between the 2nd and the 5th axis,

Table 4 contains the three parameters of the workspace and Table 5 the amplitude, the speed and the resolution for each axis.

Table 4 – Staubli RX 90L work envelop

Parameter	Symbol	Value
Maximum reach between 2nd and 5th axis	$R.M.$	$1100\ mm$
Minimum reach between 2nd and 5th axis	$R.m.$	$401\ mm$
Reach between 3rd and 5th axis	$R.b.$	$650\ mm$

Table 5 – Staubli RX 90L amplitude, speed and resolution

Axis	1	2	3	4	5	6
Amplitude (°)	320	275	285	540	225	540
Working range (°)	A ± 160	B ± 137.5	C ± 142.5	D ± 270	E $+120$ -105	F ± 270
Nominal speed $\left({}^{\circ}/_{s}\right)$	236	200	286	401	800	1125
Maximum speed $\left({}^{\circ}/_{s}\right)$	356	356	296	409	800	1125
Resolution $(^{\circ} \cdot 10^{-3})$	0.87	0.87	0.72	1	1.95	2.75

Figure 12 below is a schematic representation of the work envelop, with marks for the parameters R.M., R.m. and R.b. and axis ranges (A-F).

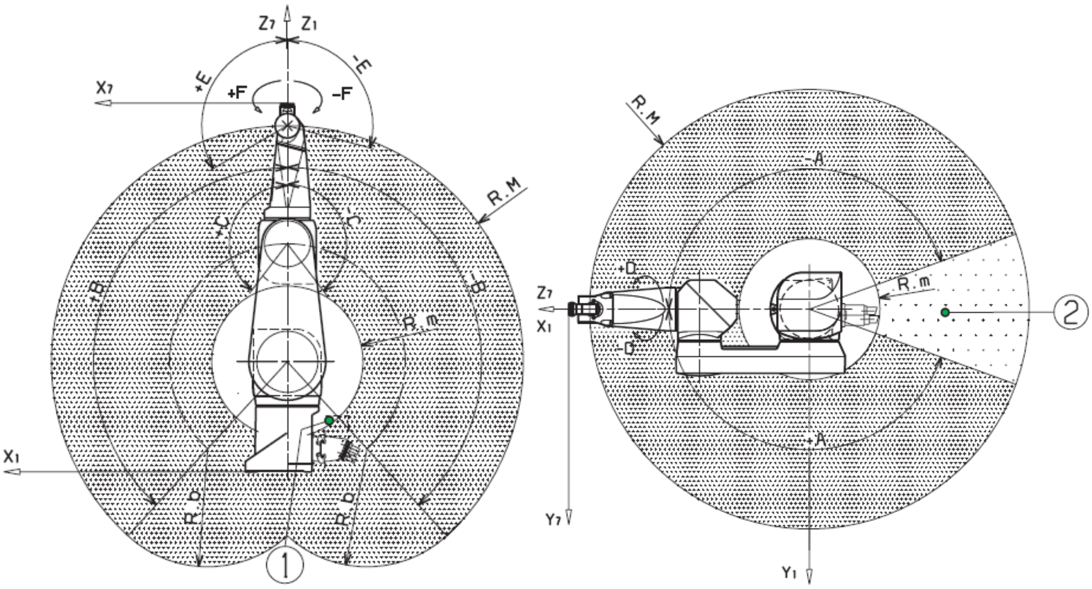


Figure 12 – Staubli RX 90L work envelop

3.2. V$^+$ Language for industrial robot applications

V$^+$ is a programming language, developed by Adept Technologies, designed specifically for industrial robot applications. It is suitable for controlling robot motion, vision systems and input-output operations. V$^+$ is known for its simplicity to learn and use, the fast (self-) compilation and the efficient and flexible memory management. Key characteristic is its robustness when managing demanding robotic tasks and integrating robotic components. As a real-time system, the constant calculation of orbit allows complex moves to be executed immediately, with efficient usage of system's memory and with the minimum system complexity [17]. The V$^+$ system produces control commands for the robot while it interacts with the user, allowing that way the creation and modification of programs. The precise control and coordination of the robots makes V$^+$ very important in automotive and manufacturing industries. V$^+$ language offers the same functionalities as the modern, high-level programming languages, as subroutines, control structures, multitasking environment and recursively program execution with re-entry. The V$^+$ Reference Guide [18] and V$^+$ User's Guide [19] contain all information to understand and learn V$^+$ from scratch.

To manipulate the robot, V$^+$ has a numerous motion key-words that correspond to different action. The most relevant for this study are summarized in Table 6.

Table 6 – Motion Control Operations [19]

Keyword	Function
APPRO	Start joint-interpolated motion towards a location defined relatively to a specified location.
APPROS	Start straight-line robot motion towards a location defined relatively to a specified location.
BRAKE	Abort current robot motion.
BREAK	Suspend program execution until the current motion completes.
CLOSE	Close robot gripper.
CLOSEI	Close robot gripper immediately.
DELAY	Cause robot motion to stop for the specified period of time.
DEPART	Start joint-interpolated motion away from the current location.
DEPARTS	Start straight-line robot motion away from the current location.
DRIVE	Move an individual joint of the robot.
MOVE	Initiate a joint-interpolated robot motion to the position and orientation described by the given location.
MOVES	Initiate a straight-line robot motion to the position and orientation described by the given location.
OPEN	Close robot gripper.
OPENI	Close robot gripper immediately.
#PPOINT	Return a precision-point value composed from the given components.
RELAX	Limp the gripper
RELAXI	Limp the gripper immediately.
ROBOT	Enable or disable one or all robots.
SPEED	Set the nominal speed for subsequent robot motions.

3.2.1. Robot Speed

The robot motion, from one point to another, has three phases: acceleration, constant speed and deceleration. The acceleration phase is from the start until the maximum speed and deceleration is from the constant speed until the end position. The constant/maximum speed is specified as a percentage of the default/nominal speed of the robot. For example, "*SPEED 25*" sets the motion speed to the 25% of the default speed.

3.2.2. Basic Motion Operations

For the robotic arm to move from one place to another, there are two possible ways/paths: joint-interpolated motion and straight-line motion [19]. Joint-interpolated motion moves each joint simultaneously, at a constant speed and so the end-effector moves in a smooth and predictable path. This type of motion is ideal for precise tasks that require high accuracy, like assembly, welding etc. Straight-line motion moves the tool tip in a straight line, from the start till the end position, and so the control system calculates the corresponding motion of each joint to achieve this motion. This type is used in cutting processes. To distinguish between the two types of motion in V^+ an "s" is added in case of straight-line paths, like "*DEPARTS*" instead of "*DEPART*".

Table 7 – Keywords for Basic Motion Operations

Joint-interpolated motion	**Straight-line motion**
APPRO	*APPROS*
DEPART	*DEPARTS*
MOVE	*MOVES*

3.2.3. End-effector operations

The tool tip or end effector is the part attached to the end of the robotic arm, providing functionalities similar to human hand [20]. End effector are of different types for the different industrial processes, as the same robot can be used in various number of applications. The right selection of the end effector is crucial for the robot to be able to carry out its tasks. The categories of a tool tip are gripper, processing tools and sensors. End effectors can be of one of the mentioned categories or even combination of them, depending on the desired output. The object of this study is a robotic gripper, the most common end effector type. The gripper's functionality is very similar to a human hand functionality, it can be used for tasks like picking and placing, shorting items, assembly etc.

The gripper can be in one of the following stages: open, closed or relax, which are defined by the commands *OPEN/OPENI*, *CLOSE/CLOSEI* and *RELAX/RELAXI*. The "*I*" at the end of the command specifies that the action will happen immediately, before the next action; otherwise, it is executed at the same time with the next command.

Table 8 – Keywords for End-effector Operations

Execution in parallel with next command	**Execution before the next command starts**
OPEN	*OPENI*
CLOSE	*CLOSEI*
RELAX	*RELAXI*

4. Methodology Speech Recognition

This thesis is about the development of a speech recognition model, to understand spoken commands and translate them to written text. The input commands are single word recordings and not continuous natural speech. Since the application is intended for an industrial environment, is important that the features extracted are not sensitive of the environment, the background noise or the microphone mismatches.

The speech recognition process includes all the steps from recording the voice signal, until classifying the commands based on given dataset/vocabulary. Pre-processing is the first step, where the signal is isolated from noise, or unnecessary information, and its size is reduced. Feature extraction is the stage where from the pre-processed signal are extracted specific parameters, indicatives for the content of the signal. These parameters are the significant characteristics of the audio signals and are used from the machine learning model, at the classification stage, to assign the recordings to the corresponding commands. The sequence of the steps is shown in Figure 13.

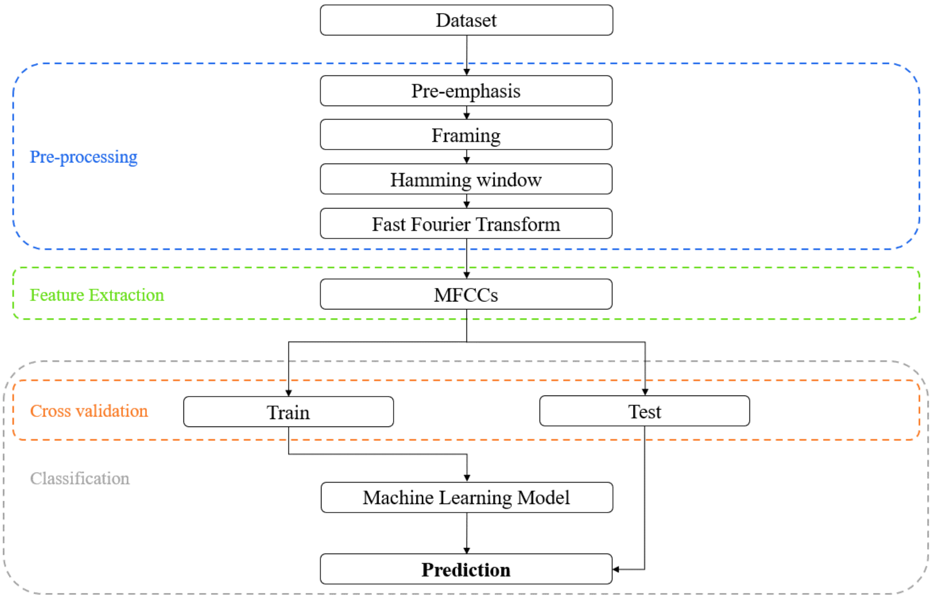


Figure 13– Speech recognition methodology [21]

Figure 13 is the workflow of the training process, it describes the process from the creation of the dataset until the neural network training. In a similar way, Figure 14 is the workflow of the final application, with input a command verbally given and output the command in written text.

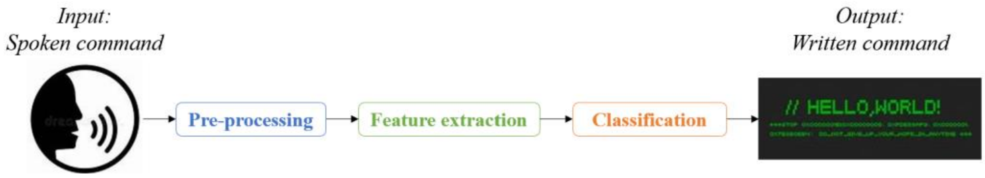


Figure 14– Application workflow

This chapter is separated in training process methodology and testing process methodology, The training process includes of the steps for the development and optimization of the speech recognition object and the testing process refers to the final evaluation and usage of the developed model.

4.1. Training Process

The training process includes the sequence of steps for the development of the speech recognition model, the definition of the vocabulary, the implementation of signal processing and feature extraction methods, and the optimization of the machine learning technique.

4.1.1. Vocabulary definition and recording

The number one step, before starting developing the model, is to establish the application and, based on that, define the vocabulary that the model must recognize. As mentioned before, the system should be joined with a robotic arm and be able to manipulate it for different actions. The vocabulary is designed on the robot in a way that all the words correspond to specific commands in V^+.

After deciding on the vocabulary, next step is to establish the format of the commands given to the system. The inputs are given one by one, as the model is not capable to recognize continuous, natural speech, and with certain order, since the model can't understand the meaning of the words. In previous chapter was given the list with the relevant commands for the application. They can be categorized in three main categories: the ones that specify the action of the end effector (open/openi and close/closei), the ones that specify the motion type (drive, move/moves, appro/appros and depart/departs) and the ones that specify a short stop or total termination of the executed action (delay, break and brake).

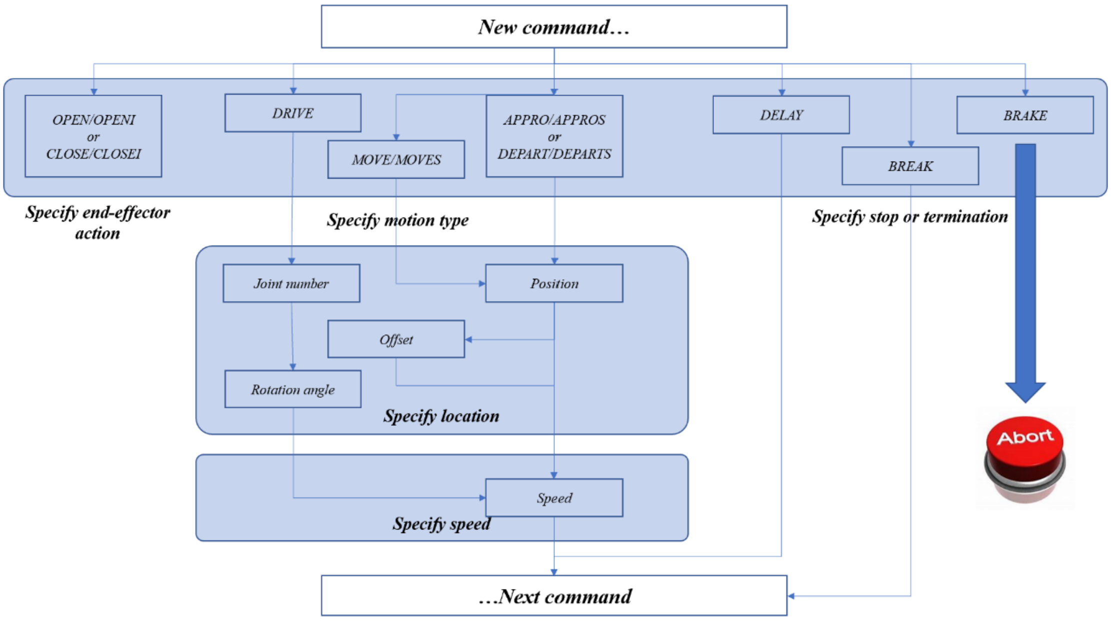


Figure 15– Robot command workflow

One of the difficulties in speech recognition, is the inability to distinguish between homophones, words with the same pronunciation and different meaning (and/or different spelling). Sometimes, one words with same sound and spelling can have different meaning, depending on the context. The humans understand this type of words based on the context of the sentence or of the topic. An example of homonym words is: "*BREAK*" and "*BRAKE*", that even people have difficulties to distinguish. These words have the exact same pronunciation and the only way to identify which one is used, is by understanding the sentence around it.

Another difficulty, is the distinction between near-homophones, words with different but similar sounds, like "*MOVE*" and "*MOVES*". This specific example is not of a problem in real life: when referring to the noun "move", "moves" is the plural of the same meaning and when referring to the verb “move”, “moves” is the third-person singular present tense. But in the case of V$^+$, “*MOVE*” and “moves” correspond to different movement type.

In case of natural speech recognition, where the input is whole sentences, the model is reinforced with this extra knowledge, to be able to understand the meaning based on the context or even guess word relatively to the words spoken before and after. For that, it is necessary to teach the neural network linguistics and grammar rules, similar to how people learn. The extra amount of data for training and the addition of rules, increase the complexity of the model.

In case of single-word recognition there are several ways to solve the problem of homonyms or near-homophones, but all of them are artificial ones for the model and don’t based on human understanding. One solution is to add a follow-up command to specify which of the possible words is the correct. This second commands can be a number, for example if the word is either “*BREAK*” or “*BRAKE*” then “*one*” can be to “*BREAK*” and “*two*” can be “*BRAKE*”. A variation of this solution is to use, as the follow-up command, a word connected to the action, for example if the word is either “*MOVE*” or “*MOVES*”, the command “*joint*” corresponds to “*MOVE*” and “*line*” to “*MOVES*”. Another approach is to use a different word of a similar meaning to avoid the speech confusion, for example, instead of “*BRAKE*” the keyword could be “stop” or “abort”. In general, is not necessary that the spoken word is exactly the same as the corresponding V$^+$ command and it is the computer-robot interface that will connect the ANN vocabulary with the V$^+$ vocabulary. The described vocabulary transform is shown in Figure 16.

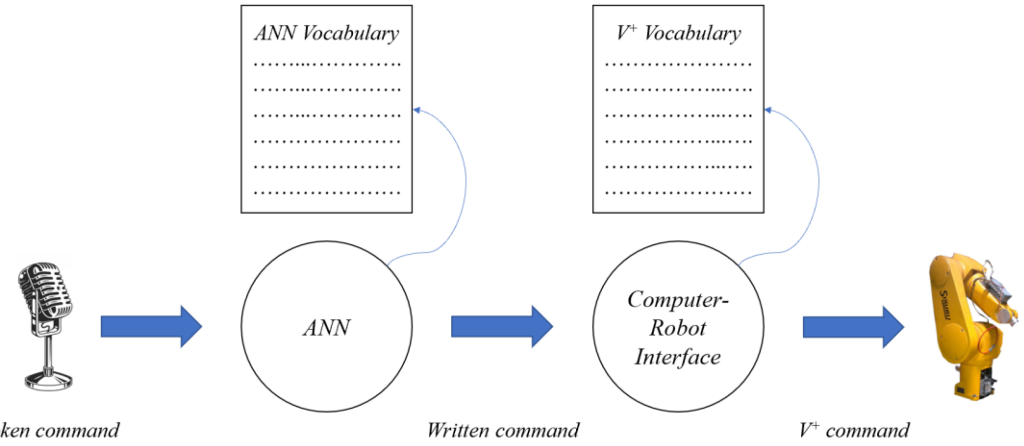


Figure 16– Vocabulary transform

For this study, it is decided the third approach, since it is simpler for the user and doesn’t add extra commands and steps for the model.

4.1.2. Pre-processing

Pre-processing is a sequence of steps to prepare the input signal for analysis and recognition. During this process, the signal compressed and cleared from background noise, normalized in a standard level and broken into smaller, overlapping pieces, the frames. The detailed steps are Pre-emphasis, Framing, Windowing and Fast Fourier Transform (FFT).

4.1.2.1. Pre-emphasis

Pre-emphasis is a Finite Impulse Filter (FIR) applied on the signal, in the time domain. It is used to remove the background noise, by improving the signal-to-noise ratio, and enhance the clarity of the audio signal. The term noise, in speech recognition, refers to any unwanted sound that occurs and interferes with the main, useful signal. The noise, in high volume, can cover part of the word and make it impossible to understand, even for the human ear, or adds extra frequencies and leads to wrong interpretation of the command. The noise can be either a continuous, low frequency disturbance or an instant, high frequency sound. In order to remove the noise from the signal, the noise characteristics should be identified and expressed in mathematic terms. Useful sounds, like music or human speech, have certain frequency range, follow predictable amplitude patterns and normally have harmonic structures. On the other hand, noise occurs in a larger range of frequencies and generally has irregular amplitude changes.

The pre-emphasis filter has another effect; besides the noise removal, it helps in balancing the frequency spectrum. In physical speech, the signals experience spectral roll-off of $\sim 6\ dB$ per octave, which means that for each doubling of the frequency, the amplitude is reduced by half. Therefore, there is more energy concentrated to low frequencies, as the amplitude is higher, and significantly less energy is allocated in high frequencies. In that case, the neural network will handle the low frequencies as more significant than the higher ones and miss important information [22]. By applying the pre-emphasis filter, it boosts the higher frequencies and the overall frequency spectrum becomes more balanced.

The mathematical expression of the filter is:

$$y(n) = x(n) - a \cdot x(n-1), \qquad n \subseteq [1, N]$$

Where,

- α is the pre-emphasis constant, $0.9 < \alpha < 1$,
- y is the new signal, after pre-emphasis $[dB]$,
- x is the original signal, before pre-emphasis $[dB]$,
- $n-1$ and n are two consecutive moments $[sec]$ and
- N is the total duration/length of the signal $[sec]$

The pre-emphasis constant can take different values between 0.9 and 1, but the output is not sensitive in this change, therefore is not need to do investigation for the alternatives, according to literature review. The most commonly used values are 0.95 and 0.97; for this application it's assumed $\alpha = 0.97$.

4.1.2.2. Framing

After pre-emphasis, the next action is to break the signal into smaller parts, the frames, before applying the Fast Fourier Transform (FFT). Speech signals are not stationary by nature, but can be considered as stationary in shorter segments because of the vocal tract inertia. By segmenting, the non-stationary signal can be represented as short stationary time frames [23].

With the FFT, the signal is transformed from the time domain to the frequency domain. If the transform is applied on the whole signal at once, the output spectrum is time independent, when the signal itself is highly dependent on time. To keep the information that varies with time the signal is split into frames. There are two alternative methods for framing, overlapping frames and non-overlapping frames. With overlapping frames is more likely that all the useful information from the signal is used, when with non-overlapping frames the risk is to miss the information in the transition points between different frames. With overlapping frames, this problem is solved, since the end points of each frame are contained in the neighbor frames. That way, the discontinuities are avoided and all the signal is used in the process.

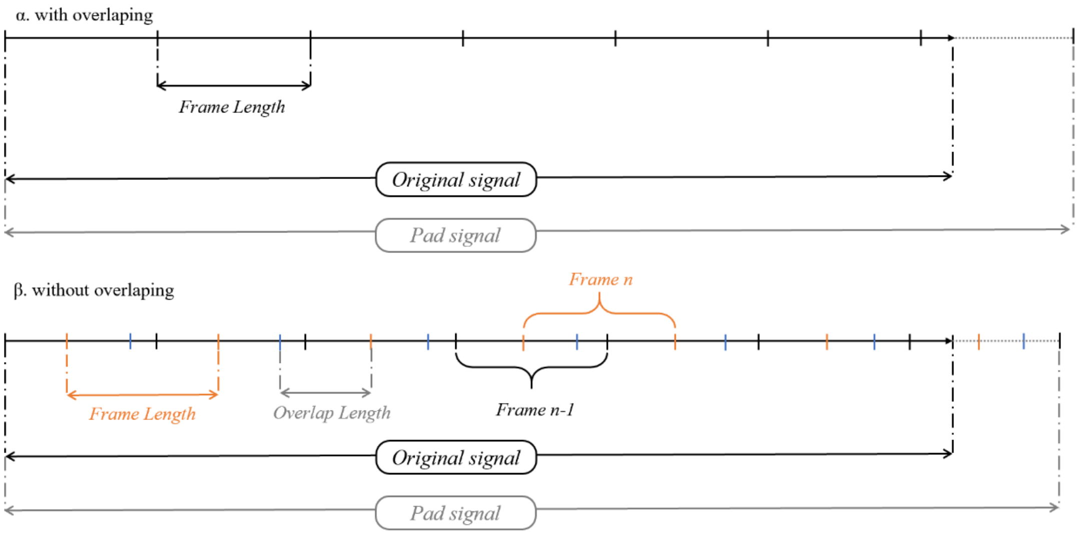


Figure 17 – Speech recognition methodology

For the framing stage, two things are needed: the frame length and the overlap length. The frame length typically is chosen between $15 - 25\ ms$ and overlapping length between $10 - 15\ ms$. Depending on the frame length and overlap, it is possible the same part of the signal to be present in three different segments. Before choosing the parameters, the sample rate should be taken into account, so each frame contains a sufficient amount of data.

4.1.2.3. Window Function

The signal segmentation into frames is essential for audio signal analysis, but can create problems when is not used correctly and carefully. When applying FFT in each frame, the frame is considered to repeat periodically. If the signal is not smooth from end-to-end, discontinuities occur, which with FFT are leading to spectrum leakages. Spectrum leakage is when the energy of the audio is spread in several frequencies, making it extra difficult for the speech recognition. The window functions, by leading the end-points to zero values, reduce the discontinuities and so, limit the spectrum leakages. The output spectrum is more representative of the frequency content of the audio. The window function, also, contributes to preserve the signal characteristic for further analysis, such as frequency formant. Lastly, window filters control the resolution of the spectrum. Frequency resolution is the ability to distinguish between two close frequencies in a signal. The window function works as a band-pass filter of different form for the different functions. The main lobe width and the side lobes affect the frequency resolution and it’s always a trade-off between frequency resolution and spectrum leakages. With narrow main lobe, the frequency resolution is better, but the side lobes are higher, leading to spectrum leakages. With wider main lobe, the results are the opposite: lower frequency resolution but less spectrum leakages.

The general mathematic form of a window function use is:

$$y(n) = x(n) \cdot w(n), \qquad n \subseteq [1, N]$$

Where,

- w is the window function,
- y is the new signal, after windowing $[dB]$,
- x is the original signal, before windowing $[dB]$,
- n is a random moment $[sec]$, and
- N is the total duration/length of the frame $[samples]$

Rectangular window function:

The rectangular window or Dirichlet window is the simplest form of a window function, impulse response, with unit response for all in-between moments and zero response at the end points. It's not usually chosen because of its low stopband attenuation. [24] Stopband attenuation, expressed in Decibel, is the difference between the maximum gain in passband region and the minimum gain in stopband region, as shown in Figure 18. [25]

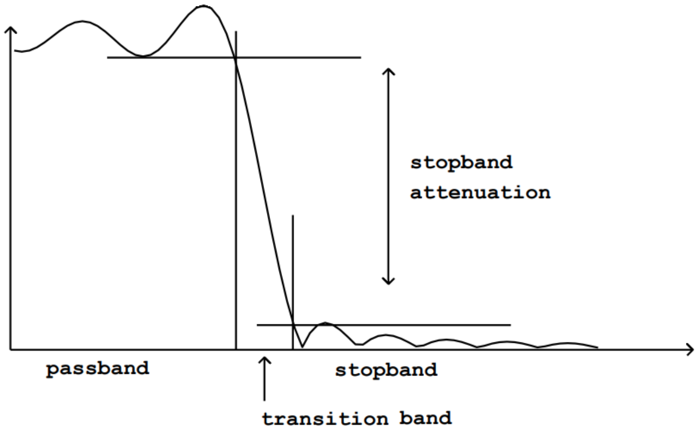


Figure 18 – Stopband attenuation

The rectangular window function is the following:

$$w(n) = \begin{cases} 0, & n = 1 \\ 1, & n \subseteq [2, N-1] \\ 0, & n = N \end{cases}$$

Hanning window function:

Compared with the rectangular window, Hanning window has a wider transition band between stopband and passband regions. The Hanning window function is the following:

$$w(n) = \frac{1}{2} \cdot \left[1 - \cos\left(\frac{2 \cdot \pi \cdot n}{N-1}\right)\right], \qquad n \subseteq [1, N]$$

Hamming window function:

Hamming window is the most commonly evolution of Hanning window, with minimum stopband attenuation. [24] The response at the edges of the frame, unlike the previous filters, is not zero and so small-scale discontinuity is expected between frames. The Hamming window function is the following:

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2 \cdot \pi \cdot n}{N-1}\right), 0 \leq n \leq N-1$$

Other methods

There are more window functions like the Blackman and Kaiser, with similar logic behind them. The Blackman window has high stopband attenuation, which makes it suitable for many different applications. The Kaiser window, known as the optimal window, adds an extra parameter to control the width of transition region.

Conclusions

The representation of the above functions in the time domain are shown in Figure 19. Hamming window is the only function with non-zero values throughout the whole curve, included the end-points.

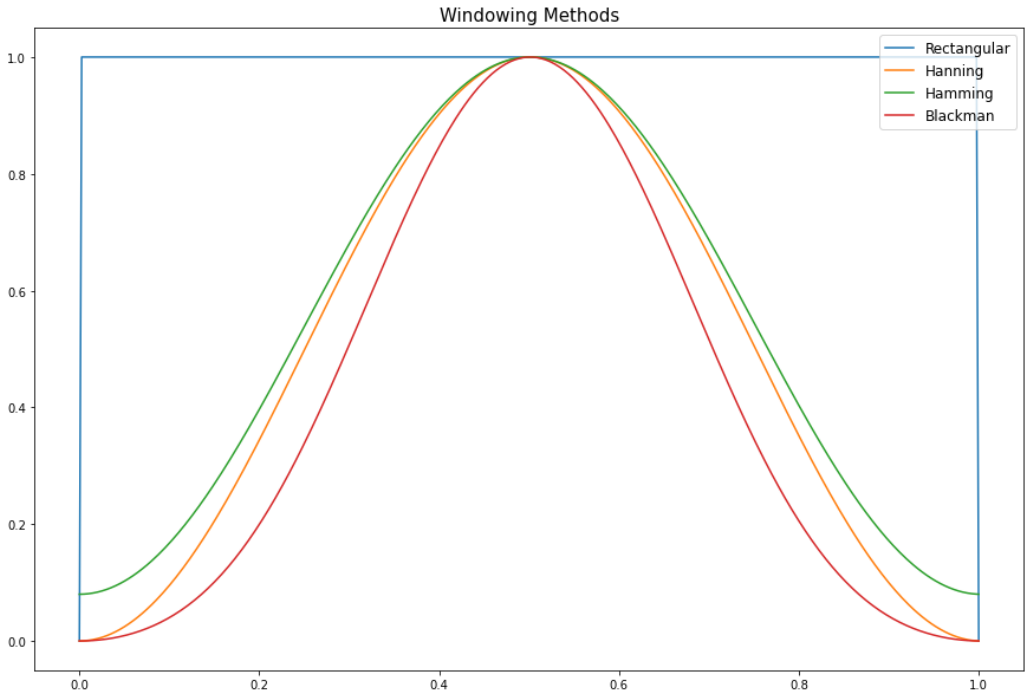


Figure 19 – Different window functions in time domain

The representation of the filters in frequency domain are in Figure 20.

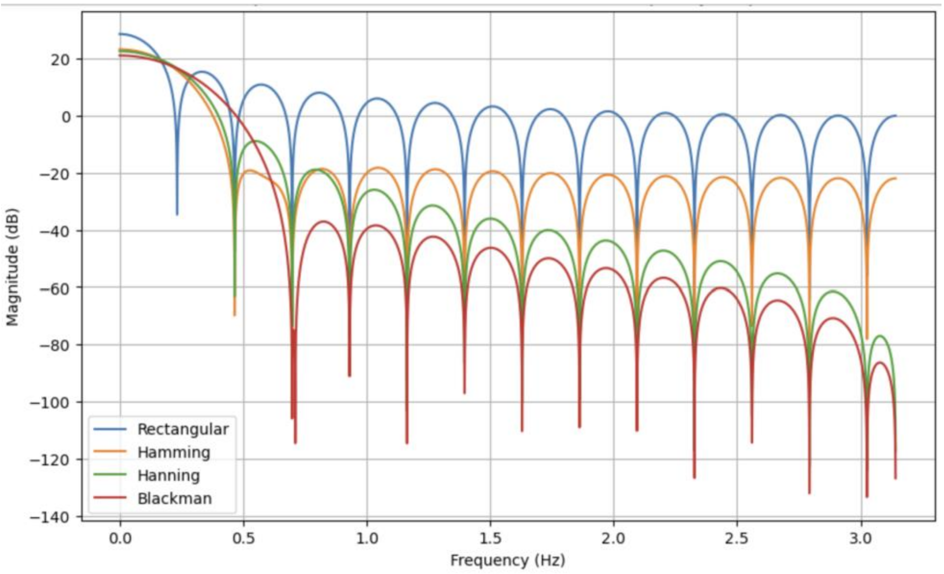


Figure 20 – Different window functions in frequency domain

The rectangular function, which acts almost as no window at all, has the narrowest main lobe, but the highest side lobes, so it maintains the best frequency resolution, but the most spectrum leakages. The exactly opposite is achieved with the Blackman window, that has the widest main lobe and lowest side lobes. Hamming and Hanning window functions have a good balance of high frequency resolution and low spectrum leakages and are preferred in speech recognition applications. Hamming window, with lower side lobes, is the favorite method.

4.1.2.4. Fast Fourier Transform (FFT)

The final step of the pre-processing stage is the Fast Fourier Transform, which transforms the recording from the time domain to the frequency domain. Practically, FFT is an optimized algorithm to calculate the Discrete Fourier Transform (DFT). Fourier transform converts the time domain signal to its frequency representation, also known as frequency spectrum, and identifies all the different frequencies that are present in the signal. The audio signal is not a continuous signal, but sampled at intervals, so the transform applied is DFT.

$$X[n] = \sum_{i=0}^{N-1} x[n] \cdot e^{-j \cdot \frac{2\pi}{N} \cdot k \cdot n}$$

Fast Fourier Transform is an optimized algorithm of the Discrete Fourier Transform and reduces the complexity from $O(N^2)$ to $O(N \ln N)$. FFT is a faster and more efficient solution than the original DFT, which makes it suitable for real time processing application.

4.1.3. Feature extraction

When the pre-processing is done, the feature extraction takes place. Feature extraction is the calculation of the information from the signal that leads to the correct classification of the words. The feature extraction process mimics the way human brain perceives the different words. One of the most common methods for speech recognition, among the Linear Predictive Coefficient (LPC) and the Hidden Markov Models (HMM), is the Mel-Frequency Cepstral Coefficients algorithm (MFCCs). To understand the MFCCs approach and method, the terms Mel scale and Cepstrum analysis should be defined.

Mel scale, from the word melody, is a perceptual scale of pitches judged by humans to be equidistant from one another. These pitches are not actually equidistant in the normal frequency scale, but are perceived as equidistant from humans. Thus, for each tone with an actual frequency, f, measured in Hz, a subjective tone is measured on the Mel scale. As mentioned in section 2.3, the human ear doesn't understand all the frequencies, of the hearing spectrum, in a same way. Humans can distinguish very good small frequency differences in low frequencies and a lot harder in higher frequencies. This effect starts to be noticeable after the 500 Hz. The mathematic relation between the Mel- and the Hertz-scale is:

$$m = 2595 \cdot \log\left(1 + \frac{f}{700}\right)$$

The Mel frequency scale is a linear frequency spacing below 1000Hz and a logarithmic spacing above 1000Hz, as shown in Figure 21.

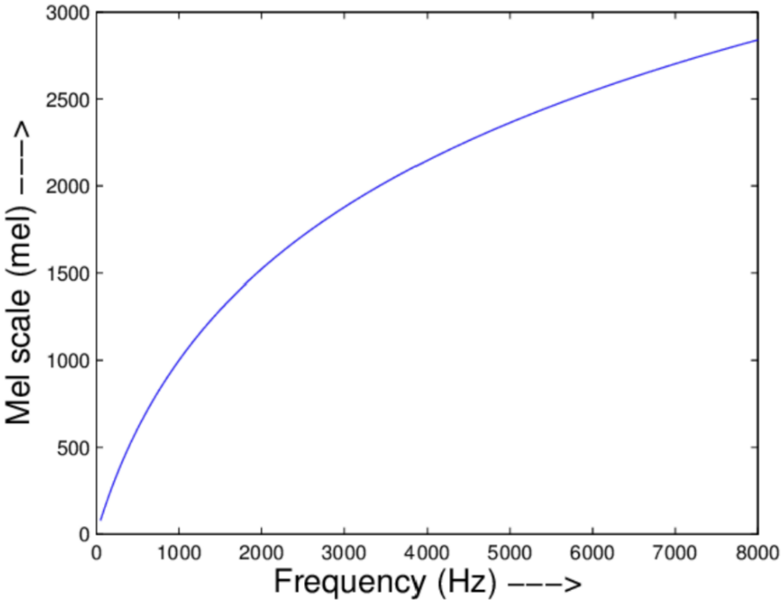


Figure 21 –Mel- and Hertz-scale relation [17]

The term Cepstrum come from reversing the first four letters of the word Spectrum and was used for the first time in 1960s at MIT during the study of echoes in seismic signals. Cepstrum is connected also with the terms quefrency or liftering analysis, which come from reversing the words frequency and filtering. The Cepstrum is a representation of a signal's spectrum in a way that allows the identification of different audio components, like the vocal tract and the excitation source, and contains information regarding the fundamental frequency of human speech. Cepstrum in the quefrency domain is what exactly is the spectrum in the frequency domain and the connection between them is shown in the following equation:

$$C\big(x(t)\big) = F^{-1}\left(\log Mel\left(\left(F\big(x(t)\big)\right)\right)\right)$$

Where,

- $x(t)$ is the audio signal,
- $C\big(x(t)\big)$ is the cepstrum of the signal,
- $F\big(x(t)\big)$ is the spectrum of the signal (Fourier Transform), and
- F^{-1} is the inverse Fourier transform.

As the above equation shows, the steps to calculate the MFCCs, from the last step which was the FFT, are:

- Apply the Fast Fourier Transform of the segmented signal, to calculate the frequency spectrum.
- Convert the frequencies from Hertz scale to Mel scale.
- Compute the logarithm of each frequency.
- Apply the Discrete Cosine Transform (DCT), to convert cepstrum back in the time domain.

To calculate the DCT it is used:

$$c_n = \sum_{n=1}^{k} \log(X[n]) \cdot \cos\left(n \cdot \left(k - \frac{1}{2}\right) \cdot \frac{\pi}{k}\right)$$

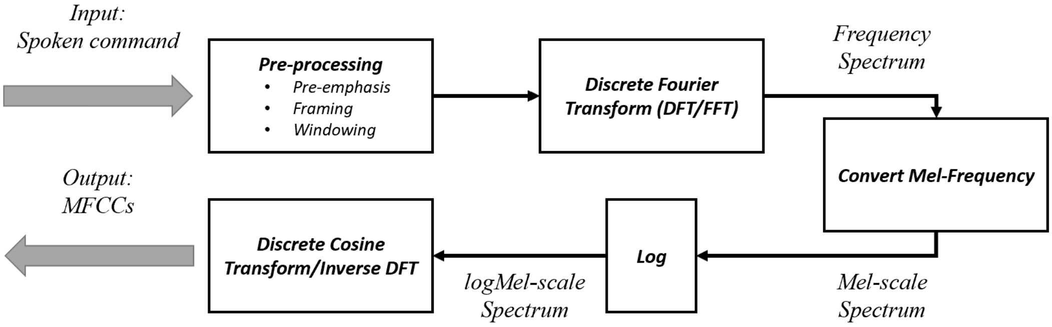


Figure 22 –MFCCs pipeline [26]

From research it is proven that the most important coefficients are the first 12 and that's the recommended value from literature.

4.1.4. Classification

After extracting the MFCCs, the signal processing is ready and next step is the classification. Classification is the process of assigning an unknown quantity in one of the known categories. The classification model, in this case, is an artificial neural network for pattern recognition. Artificial Neural Networks (ANNs) have structure and functionalities inspired by the biological network of neurons [27]. The human Central Nervous System consists of cells, the neurons; each neuron consists of the main body – soma – the apophysis – dendrite – and the main axon. The axon of one neuron is connected, via synaptic gaps, with the dendrites of the neighbour neurons and is transferring information through them. In a similar way, an artificial neural network consists of artificial neurons connected with one another, transferring data along the model structure. Each neuron has the input signals, each one multiplied with a custom weight, the activation threshold, which controls whether the specific neuron will be activated or not, based on the summary of the input signals and their weights, and the output signal. In every neuron the input signals are multiplied by the weights and added all together. The sum is the function output of the neuron.

The neurons are organized in layers, which are levels of information. The input layer consists neurons as many as the design variables. For this problem, the inputs are the MFCCs for all the signal frames. The output layer is the solution and in this study the label of the command. All the in-between layers are called hidden layers. All the neurons of a layer are receiving the output signals of all the neurons in the previous layer and are sending their output signal to all the neurons of the next layer, as shown in Figure 23.

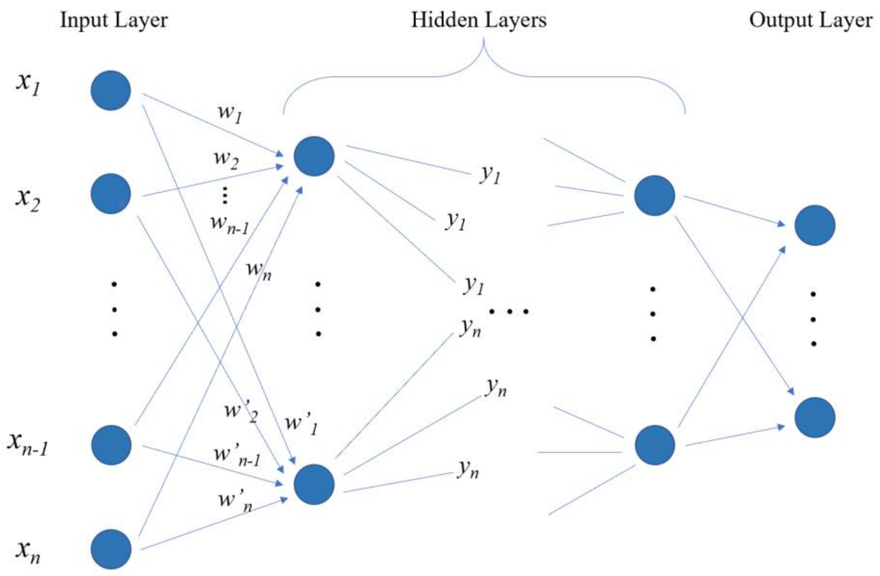


Figure 23 – Artificial Neural Network Architecture

The aim is that the final output of the neural network corresponds to the actual labels of the input recordings. This is achieved after the training process, where the weights are iterated until the obtained values result to the correct total output. The iterations are similar to optimization steps with the goal to minimize the error between actual value and predicted value. In regression problems the output is a real number, in classification problems the output must be the category label. Actually, the network gives as output *N* real numbers (between 0 and 1), when the potential labels are *N*, and each one of them represents the possibility of the command to belong in the corresponding category. The recording is classified in the category with the highest predicted possibility.

To develop a machine learning model the whole dataset is divided into train and tests subsets. The train set, as the name implies, is used to train the network, so the output is given to the network and the model tries to predict them. The test set is used to check the accuracy of the model and its generalization ability, so the output is not given together with the input, but is fed to the network in later stages.

4.1.4.1. k-Fold Cross Validation

To train a machine learning object the data are split into train and test subsets and the optimization and evaluation of the model is based on the accuracy in the testing predictions. The risk with using only one test set is that the accuracy can vary significantly depending on the observation in the test set. That means that, potentially, if the model is used for different testing sets the prediction error will have big deviations. To ensure that the accuracy is representative of the model at any kind of testing set, the k-fold cross validation method is used.

The cross validation divides randomly the whole dataset into k groups, the “folds”, of the same roughly size. One of the folds is used as testing set and the k-1 remaining folds are used as the training set, as shown in Figure 24. With this setup the optimization process runs until the end and the accuracy of the model is calculated. This routine is repeated k times, until all the folds become once the test set. After this is done the average error is calculated:

$$MSE_{tot} = \frac{1}{k} \cdot \sum_{i=1}^{k} MSE_i$$

The ANN must have a constant behavior regardless of the test set, since the real testing data are totally unknown. The acceptance criteria is when the average accuracy is above 80%, but also the individual errors are not deviating a lot.

The iteration can be done with two different ways: the weights of the networks are initialized before every iteration or each iteration uses the last wight update from the previous iteration. With the second way, the training process converges earlier and can reach a better result. The first approach has the advantage of separating the testing set completely from the training. When the initialization is using previous values, these values contain information from the previous train set, part of which is now the test set. In this study, the first approach is selected.

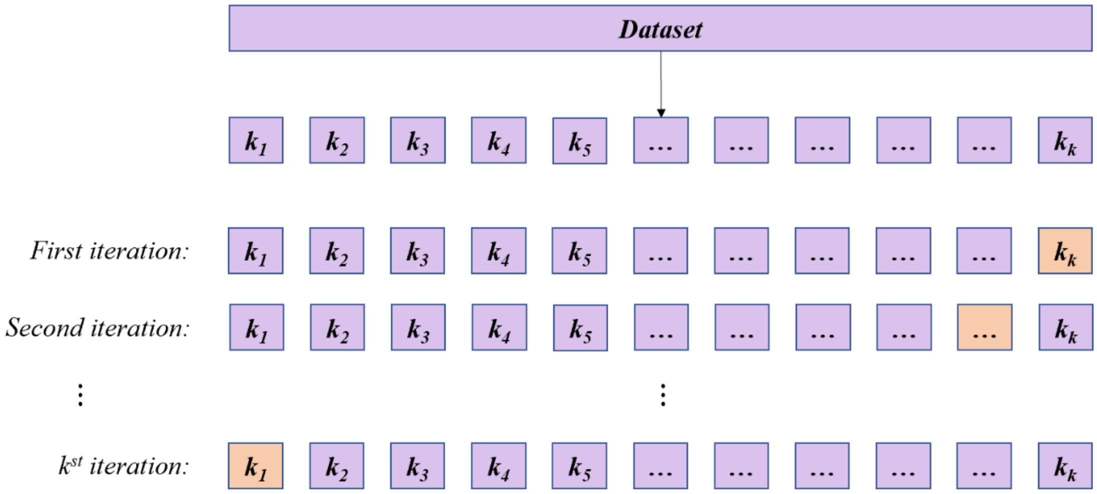


Figure 24 – k-fold Cross Validation Visualization

The selection of the number of folds (k) needs careful consideration. The more folds used in cross validation the more representative the results are and the lower the bias is, but at the same time the amount of testing data are less and of training data more. The fewer folds used the higher the bias but the lower the variance. It is therefore a trade-off between bias and variance when it comes to k selection.

4.1.4.2. Artificial Neural Network Parameters

The artificial neural networks have many parameters, that should be chosen carefully. There is not always a rule of thumb for the selection, since the best combination differs from case to case. The key is to keep a good balance between memorization and generalization. The model should be fitting well on the training data, but if it overfits on them, it will lose the ability of generalization, the ability to predict data that are not exactly same as the ones in the train set. The main parameters of a neural network are the architecture, the training function and the stop criteria.

The most important parameter is the network’s architecture, the selection of the number of hidden layers and the number of neurons in each layer. Very small networks, few layers with few neurons each, have a limited ability to learn when the problem has many inputs and high complexity. The benefit of small models is that the require less memory and need less computational time to be trained and used. Very big networks, many layers or many neurons in each layer or both, can handle better complicated datasets, but need more space and time. If the network is too small for a certain problem, it will have low accuracy because of poor learning capability. If the network is too big, it will also have reduced accuracy, due to overfitting on the known data and losing the skill to generalize predictions for unknown data

Another parameter, that can differentiate the networks, is the training function, the algorithm that controls the learning process and the optimization of the set of weights. The steps of the learning process are the forward propagation, the loss calculation and the backpropagation. The forward propagation is the computation of the network's output based on the inputs. In the first iteration the weights are initialized either randomly or by the user based on his existing knowledge and in any other iteration they are corrected based on the previous value. The loss calculation step is the calculation of the delta between the actual and the predicted value. The most common loss function is the Mean Squared Error (MSE) for regression problems and the Cross Entropy Loss for classification. Lastly, the backpropagation is the process of updating the weights by transferring the error value through the network, at the opposite direction of the forward propagation. The training algorithm defines the method for updating the weights. The simplest method is the Gradient Descent. The loss function is also a parameter for investigation, but in this study the Cross Entropy Loss function was chosen, based on literature review.

The last of the parameters is the stop criteria, the condition which needs to be fulfilled for ending the learning process. There are several stop criteria, like the maximum number of epochs, convergence of loss function and training time. The idea is that all different criteria have a target value and the learning process will be terminated, when any of them is fulfilled. The best scenario is of course to fulfill the convergence of the loss function, which indicates a good prediction accuracy. The reason for having more that one stop criteria is because maybe one is never fulfilled, regardless of the iteration number, and in that case the training would never stop.

4.1.4.3. Evaluation Metrics

The evaluation metrics are indicative of the network's quality. Based on the evaluation metrics the quality and efficiency of the network are established and different networks can be compared and ranked. For classification neural networks these metrics are accuracy, precision, recall and F1 scores.

Accuracy, the most common and widely understandable parameter, is the network's ability to predict correctly, the true class, and is expressed as the percentage of the correct predictions divided by the total predictions. The higher the accuracy is, the more are the correct predictions. Accuracy is the most important performance indicator in machine learning models.

$$Accuracy = \frac{correct\ predictions}{total\ predictions}\ [\%]$$

The metrics precision and recall are easier to explained per class, or in a binary classification problem, and then raised in a multi-class level. In binary classification problems, the unknown portion belongs either to class A or class B, or in different words, either belongs to class A or not. Instead of asking the question: "In which class does the object belong?", the question is rephrased to: "*Does the object belong to class A*?". If the object does belong to class A the answer is positive and if it doesn't the answer is negative. This type of problems exist in the majority of medical test, e.g. "*Is the woman pregnant*?", "*Is the patient ailing from this disease*?". The predictions that classify the unknown quantity in class A are considered positive predictions and the others negative. The predictions correctly classified as A are true positives and the ones wrongly classified as A are false positives. Similarly, the prediction correctly categorized as non-A are true negatives and the rest, wrongly categorized as non-A, as false negatives. Now that these terms are explained, is easy to define the metrics precision and recall.

Precision is the percentage of the correct positive predictions divided by the total amount of positive prediction, or simpler "*Out of all the positive predicted examples, how many are actually positive?*". The total positive predictions are the summary of the true positives and the false positives. High precision means that if a case is predicted positive, it is very likely that it actually is positive. There are some applications, where the precision is more important that the accuracy. An example is face identification problems, where is very important that only the authorized users are recognized.

$$Precision = \frac{true\ positive\ predictions}{total\ positive\ predictions}\ [\%]\ or \frac{TP}{TP + FP}\ [\%]$$

Recall is the percentage of the correct positive predictions on the actual positive cases, or "*Out of all positive case, how many were actually predicted as positive?*". Medical diagnosis, especially in high-risk disease detections, is a field where recall is very important, as the patients that are positive to the disease must be identified and not considered as healthy, to start the treatment immediately.

$$Recall = \frac{true\ positive\ predictions}{total\ actual\ positives}\ [\%]\ or \frac{TP}{TP + FN}\ [\%]$$

Lastly, F1-score is a combination of precision and recall.

$$F1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

A good visualization of the evaluation metrics is the confusion matrix.

Table 9 – Confusion Matrix for Binary Classification

		Predicted Class	
		Positive	*Negative*
True Class	*Positive*	*True Positive (TP)*	*False Negative (FN)*
	Negative	*False Positive (FP)*	*True Negative (TN)*

In multi-class classification problems, precision and recall are calculated for every class individually, in the same way as they would be calculated in a binary classification case and are called precision per class and recall per class. The metrics for the whole model are occur from averaging the precision and recall per command.

$$precision = \sum_{i=class\ A}^{class\ N} precision|_i \Big/ N$$

$$recall = \sum_{i=class\ A}^{class\ N} recall|_i \Big/ N$$

4.2. Testing process

The training part is done, when the investigation converges to one final architecture of the ANN. For that network, the accuracy, the precision, the recall and the F1-scores are calculated. In this application, accuracy is the most significant of the evaluation metrics and precision and recall are of similar importance. When checking the overall error, is important to check the individual errors per command. It is interesting to see if the model has the same almost accuracy throughout the classes, if in case of wrong prediction, it confuses the words that sound similar to human ears and identified which commands are usually miss-predicted and which are the most likely predictions.

Still this evaluation is not enough. The model should be tested in real-time recognition with different users, to check the performance in that case as well. Until now the evaluation was based on the testing subset, which belongs to the same dataset as the training set, which means the samples are produced from the same individuals in both sets. In order to confirm the calculated accuracy, the model is tested by other users.

4.2.1. Speech Processing

The real-time recording is directed to the same speech processing algorithm, to be prepared the same way as the dataset. The steps are again:

- Signal pre-processing:
  - Noise removal and frequency balancing
  - Signal segmentation into frames
  - Power spectrum calculation
- Feature extraction:
  - MFCCs calculation

This time the processing of the audio is real-time, so it's important the algorithm's needs in time are not too big. When created the dataset, it was not that evident, how much time the script needs to compute the coefficients, but here it's a good opportunity to check.

4.2.2. Command Prediction

The MCFFs table is given to the pre-trained ANN. All the 18 commands are tested several times to ensure that the model is behaving as expected. It's good to store the real-time recordings and use them later on to enrich the dataset and continue the model training. It is possible, when obtaining new data, to combine them with the original dataset and improve the networks accuracy.

5. Results and analysis

In the previous chapter, the methodology was detailly explained and analyzed. This section presents the practical components of this study, including method implementation and results analysis. Here are discussed findings from the ANN investigation and the evaluation of the final network. Additionally, results from supplementary studies are included, to examine the effect of different pre-processing techniques. For example, is evaluated the recognition without pre-emphasis and with different window functions.

5.1. Training process

5.1.1. Vocabulary definition and recording

The most common formats for audio files are WAV or MP3. In the majority of the publications, researchers prefer the WAV files, because they include more information for the signal and span the full spectrum of frequencies audible to humans. On the other hand, MP3 files are compressed and part of the information is lost, but they occupy less memory. There are several public datasets online, with voice commands in WAV format, but only limited ones have single-word recording for robotic applications. The most suitable one, for this study, is the Speech Commands Dataset [28], that contains a great variety of commands from many different speakers. In speech recognition application, the dataset should be representative, in terms of speakers' masse and commands' number, and sufficient to give the information needed. This is a word classification application, so the important information to be extracted is the spoken word. The audio signals contain a lot more information than just the word, as the speaker's identity, the speaker's sentiment etc. The model should be trained to ignore the unnecessary data and focus on the significant ones, e.g., the command identification has to be totally user independent. For that reason, is very important that the dataset is created by a variety of speakers that repeat each word several times. In Table 10 are summarized the most crucial parameters for the dataset to be representative.

Table 10 – Recording parameters

Parameter	Notes
Number of speakers	
Sex	Male or female speaker
Accent	Pronunciation of the word
Intensity	How quiet or loud the command is pronounced
Speed	How fast or slow the word is given
Starting time	How long after the recording starts, the speaker gives the commands
Background noise	If there are additional, unwanted sounds included in the recording

Speaker variety is a very vague requirement and must be determined better. For start, the number of speakers should be at least five to ten to have a good representation of different voice characteristics, and of course the more the merrier. Only by controlling the number of users is not a guarantee that the dataset is good enough, the speakers must have different characteristics. It is known that men and women don't produce sounds with the same frequency spectrum; men normally produce lower frequencies than women. The recommendation is, therefore, to have the ratio between male and female speakers around 50%-50%.

The accent of the users affects a lot the accuracy of speech recognition. Even in real life is hard to understand people with very strong or intense accents, even more for a machine learning model. To avoid extra confusion, the assumption, that only clear accents are used, is made. The dataset consists of well-pronounced English words and is recommended for the future users of the model to speak clear and in normal speed. The stalking speeds affects the clarity of the word and the length that occupies in the total signal. The network must recognize the commands in different speeds, of course within certain limits, so the dataset has a variety of talking speeds. The classification must be also independent of the timing the user starts and stops speaking and the background noise. Of course, in case of heavy noise the recording must be canceled by the user, but the presence of background sounds is essential for the learning process, as the robot is located in an industrial environment where noise will occur at any time.

5.1.1.1. Keywords selection

After defining the format for recording, the next step is to finalize the vocabulary. As mentioned in chapter 4.1.1, it is preferrable that the keywords don't sound similar, because that would add extra difficulty and complexity to the model. If the model was designed to distinguish differences between very similar words, the vocabulary should have many more recordings per command and all of them very clearly pronounced. For simplicity, all the robot motions are considered joint-interpolated and not straight line and there is no need to specify that further. The keywords that specify straight-line motion (*APPROS*, *DEPARTS* and *MOVES*) are not relevant for now, but in future studies they should be included to add more functionality to the system. In a similar way, all the gripper's operations are executed immediately and not in parallel with the next action (*OPENI*, *CLOSEI*, *RELAXI*).

Table 11 – Final Vocabulary

Command Type	Command	V$^+$ Keyword	Function
Robot Motion Commands	*"Approach"*	*APPRO*	Move towards a relative location.
	"Depart'	*DEPART*	Move away from the current location.
	"Move"	*MOVE*	Move towards a specified location.
	"Joint"	*DRIVE*	Move a joint of the robot.
	"Speed"	*SPEED*	Set speed (% of the nominal).
End Effector Operations	*"Up"*	*OPENI*	Close the robot gripper immediately.
	"Down"	*CLOSEI*	Close the robot gripper immediately.
	"Relax"	*RELAXI*	Limp the robot gripper immediately.
Numerical Commands	*"Zero"*	0	Number.
	"One"	1	Number.
	"Two"	2	Number.
	"Three"	3	Number.
	"Four"	4	Number.
	"Five"	5	Number.
	"Six"	6	Number.
	"Seven"	7	Number.
	"Eight"	8	Number.
	"Nine"	9	Number.
	"Left"	-	Sign | negative direction.
	"Right"	+	Sign | positive direction.
Other Commands	*"Stop"*	*BRAKE*	Abort current robot motion.
	"Break"	*BREAK*	Stop action until the current motion completes.
	"Go"	*DELAY*	Stop action for a period of time.
Interface Commands	*"Robot"*	*ROBOT*	Enable or disable one or all robots.
	"On"		Start of program.
	"Off"		End of program.

Except the V$^+$ commands, there are some additional "interface" commands needed for the system to work properly. The keywords "*On*" and "*Off*" for example are indicating the start and the end of the program. All the commands should be given between those two, in order for the robot to follow them.

Unfortunately, the available dataset doesn't contain all the needed commands. The aim is to build an initial model based on the available recordings, which later can be improved. The useful commands from the Speech Commands Dataset are summarized in **Error! Reference source not found.**. The dataset contains around 2350 recordings per command, with some exceptions, like the commands "*one*" and "*on*" that have 2347 and 2330 audio files respectively.

Table 12 –Vocabulary from Speech Commands Dataset

Command Type	Reference Number	Command	Number of Recordings	Function
Robot Motion Commands	*10*	*"Up"*	2350	Close the robot gripper immediately.
	11	*"Down"*	2350	Close the robot gripper immediately.
Numerical Commands	*0*	*"Zero"*	2350	Number.
	1	*"One"*	2330	Number.
	2	*"Two"*	2350	Number.
	3	*"Three"*	2350	Number.
	4	*"Four"*	2350	Number.
	5	*"Five"*	2350	Number.
	6	*"Six"*	2350	Number.
	7	*"Seven"*	2350	Number.
	8	*"Eight"*	2350	Number.
	9	*"Nine"*	2350	Number.
	12	*"Left"*	2350	Sign | negative direction.
	13	*"Right"*	2350	Sign | positive direction.
Other Commands	*14*	*"Stop"*	2350	Abort current robot motion.
	17	*"Go"*	2350	Stop action for a period of time.
Interface Commands	*15*	*"On"*	2347	Start of program.
	16	*"Off"*	2350	End of program.
		TOTAL	**42277**	

From the recordings, the data table is created with rows to represent the different recordings and columns to represent the signal amplitude over time. The matrix has dimensions 42277x16000.

In the following figure (Figure 25) there are some examples to show the differentiation of the audio recordings. Unfortunately, the dataset includes also some outliers, like the figure in bottom right position. It's not an easy case to ignore those outliers, but they can be avoided by analyzing the dataset and removing outlier points and with the proper noise cancellation method. Some of them might end up in the final dataset, but in that case the classifier won't take them into account as they are not representative recordings.

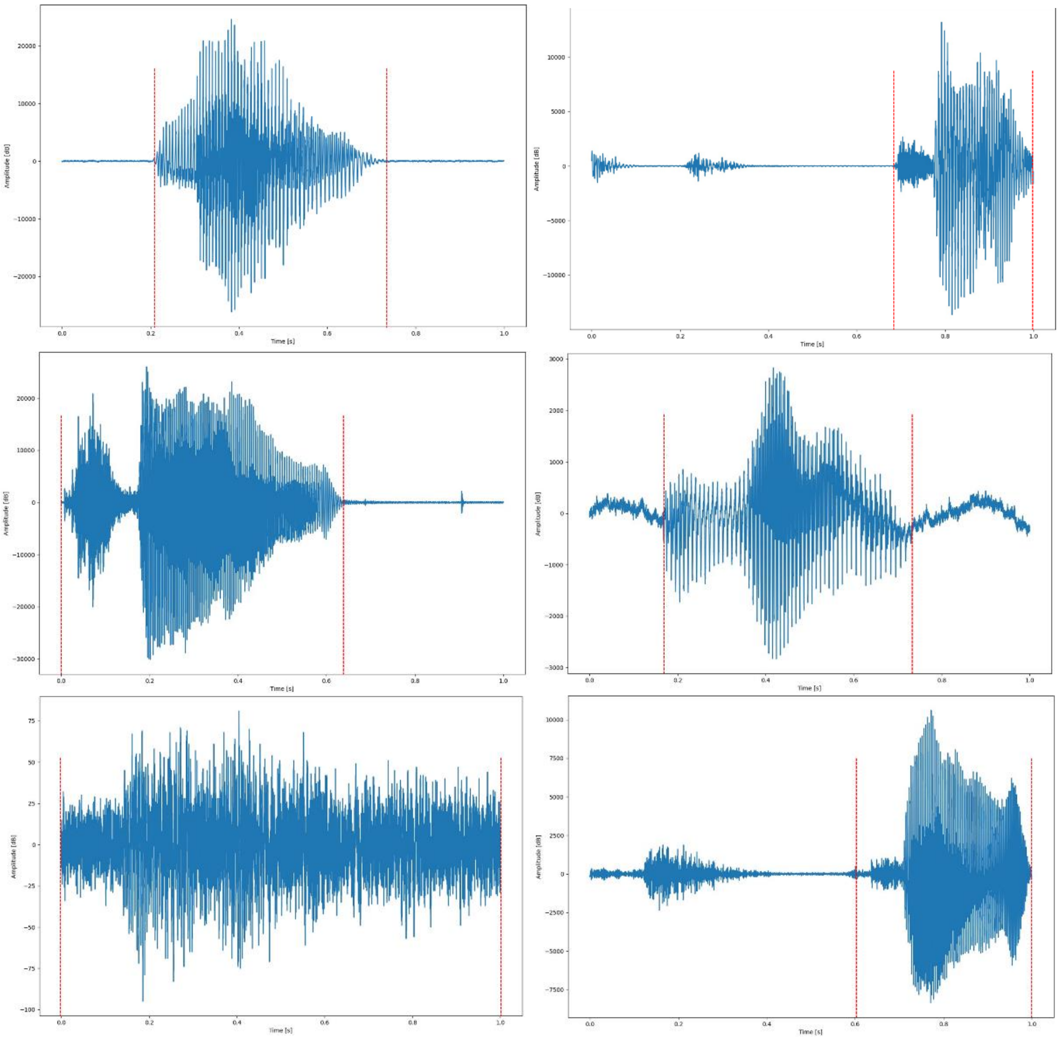


Figure 25 – Variety of recording in the dataset

All recordings in Speech Commands Dataset have duration of one second. So, the user needs to give each command, also within one second. That is important to keep consistent the number and the size of the frames, during the processing. For duration $1\ sec$ and sample rate $f_s = 16000\ Hz$, each recording consists of 16000 samples. A second is not long, so the user should start right away to include the whole word clearly, but for these single-word commands it is enough.

5.1.2. Pre-processing

The dataset is stored in a matrix 42277x16000, where the number of rows is the total number of available recordings and the number of columns the resolution of the recording. An additional column is added in the beginning of the table, that contains the corresponding word to each recording. The pre-processing is done in Python.

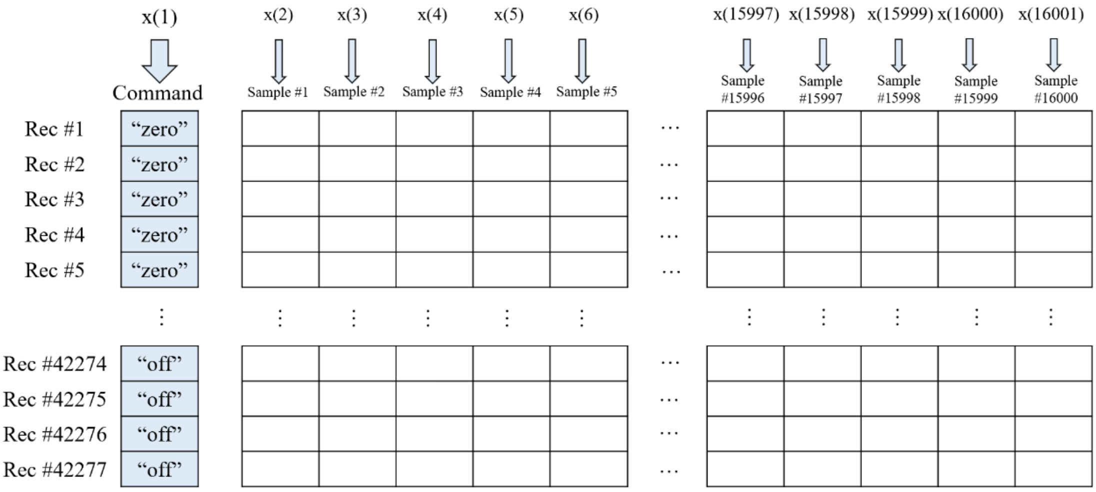


Figure 26 – Dataset table with original signals [42277x16001]

5.1.2.1. *Pre-emphasis*

Firstly, the pre-emphasis filter is applied on the dataset to improve the signal-to-noise ratio and balance the frequency spectrum.

$$y(n) = x(n) - a \cdot x(n-1), \qquad n \subseteq [1, N]$$

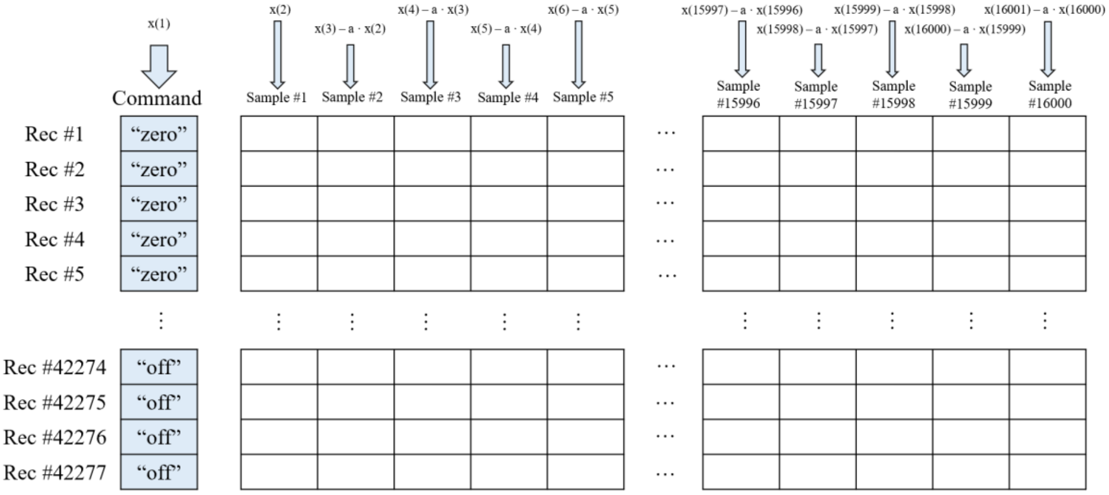


Figure 27 – Dataset table after pre-emphasis [42277x16001]

Figure 28 shows some examples before and after applying the pre-emphasis filter. The representations of the original and emphasized signals are very similar, but the emphasized one is like a scaled down version of the original. Looking closer to the graphs, the emphasized signal's amplitude is approximately half of the original one.

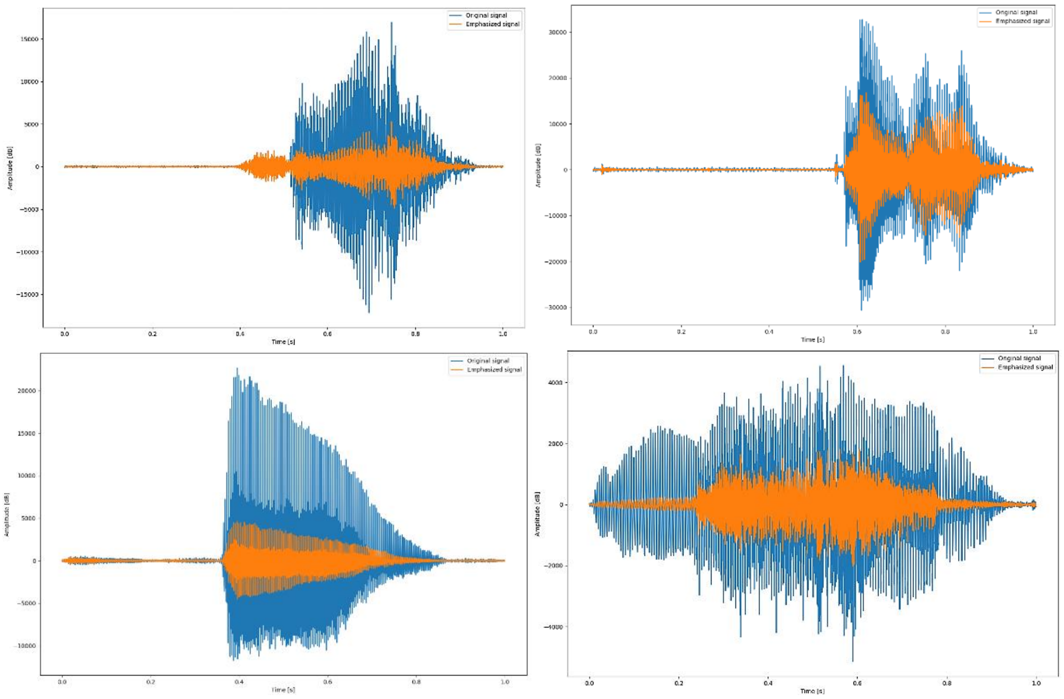


Figure 28 – Pre-emphasis filter

The pre-emphasis filter balances the frequency spectrum, by boosting more the higher frequencies and weakening the lower ones. At the same time, it removes effectively the background noise, especially when it’s generated by a constant source. As mentioned before, the amplitude of the signal after pre-emphasis is reduced approximately by half. Actually, the filter has effect similar to normalization, bringing the signals within the same amplitude range. That way the model becomes less sensitive to speech volume. If a word is pronounced louder in the majority of the recordings the model will connect the loudness with this word and when receiving high volume audios will tend to classify them as that word. Pre-emphasis filter reduces that risk, helping the model to focus in the right speech characteristics to classify the words. In Figure 29 is a zoomed in graph to understand better the effect of pre-emphasis filters.

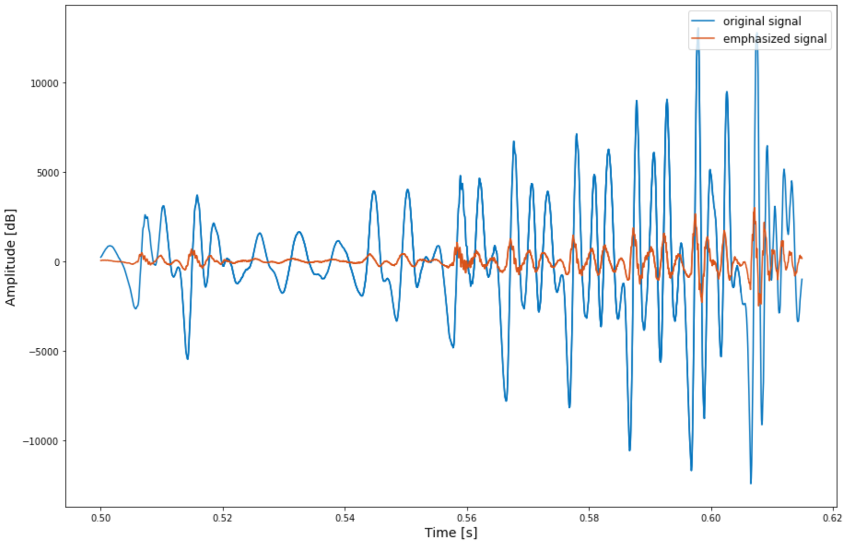


Figure 29 – Pre-emphasis filter effect

The pre-emphasized dataset has the same table dimensions as the original one: 42277x16000.

5.1.2.2. Framing

The framing parameters are the frame length and the overlapping length. The frame must be short enough to be considered as time stationary, but also, long enough to capture all the important speech characteristics. The choice of a few big frames minimizes the benefits of segmentation but requires less space and computational power. On the other hand, selecting very small frames increases the sensitivity in small sound fluctuations that might be even artificial and requires a lot more space. A bottle neck for the frame selection is the sampling ratio. The signal resolution, the amount of data points, is depended on the sampling frequency. It should be clear that the segmentation in shorter frames doesn't increase the resolution of the audio recording. The overlapping frames reduce the information loss for splitting the signa and cover for the characteristics skewed or lost on the frame's end-points. If the overlapping between frames is too long, then most of the sample points will be taken too many times into account. This probably won't compromise the overall accuracy, but will feed the network with a lot unnecessary data. From literature review, the recommended values for frame length and overlap are between $15-25\ ms$ and $10-15\ ms$, respectively. The frame length is chosen to be $25\ ms$, to occupy less memory, and overlap $15\ ms$.

For sampling frequency $f_s = 16000\ Hz$ the frame length and overlap are converted from time to samples:

$$frame\_length = frame\_size \cdot sample\_rate \Rightarrow frame\_length = 400\ samples$$

$$frame\_step = frame\_stride \cdot sample\_rate \Rightarrow frame\_step = 160\ samples$$

The number of frames per signal is calculated:

$$num\_frames = round\left(\frac{signal\_length - frame\_length}{frame\_step}\right) = 98\ frames$$

In case the sound signal cannot be divided in integer number of frames, the last frame will have shorter length or zero values should be added to the original signal. In order to have perfect integer division, the signal should have length equal to:

$$\begin{gathered} pad\_signal = num\_frames \cdot frame\_step + frame\_length \Rightarrow \\ \Rightarrow pad\_signal = 16080\ samples \end{gathered}$$

80 columns with zero values are added at the end of the original signal.

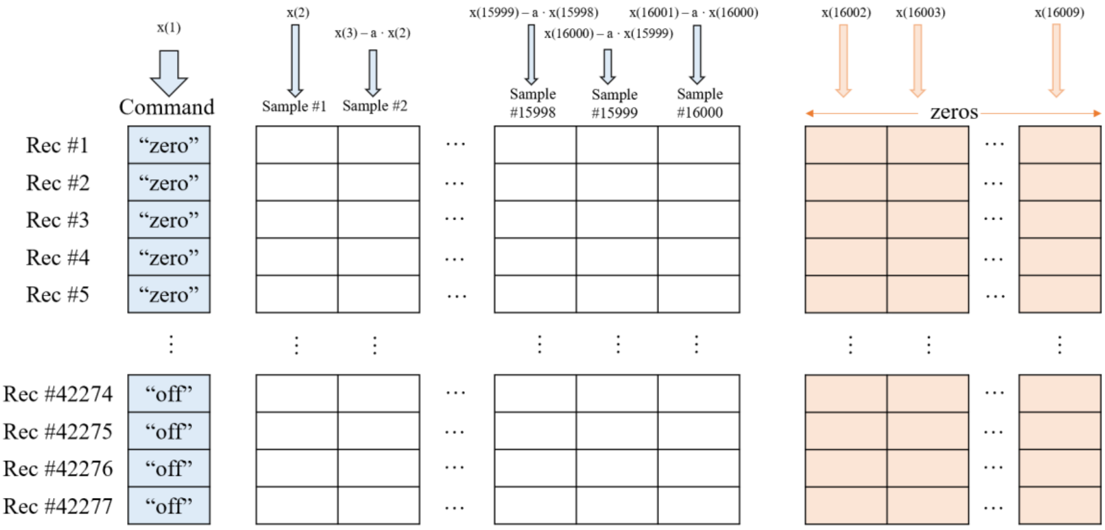


Figure 30 – Pad signal dataset

The signal is split in 98 frames with 400 samples per frame. The dataset after framing is reshaped into a table 42277x39200.

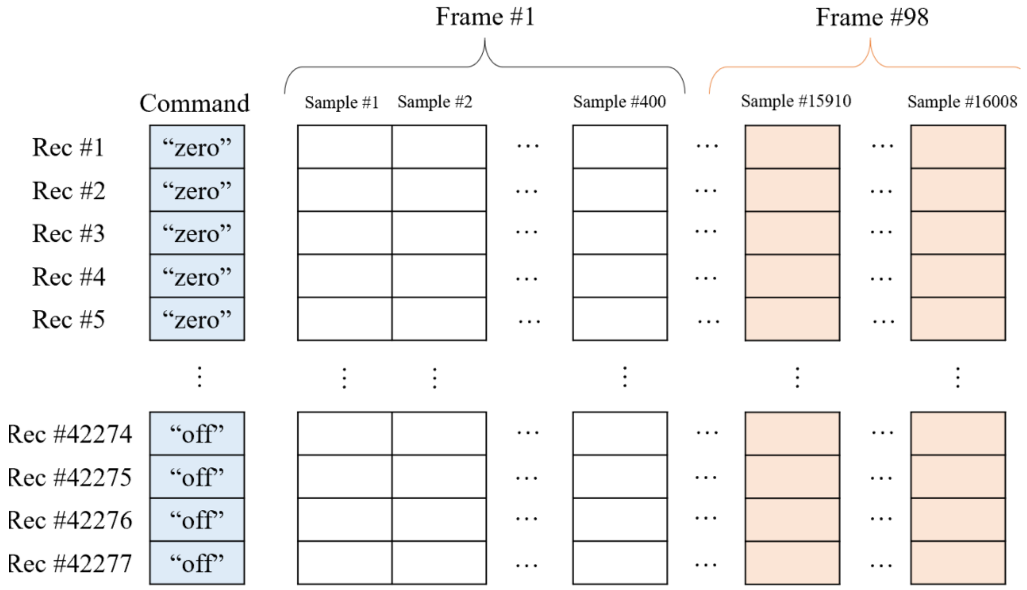


Figure 31 – Framed dataset

The signal divided into frames looks like the ones in Figure 32, where the different colours represent the different frames.

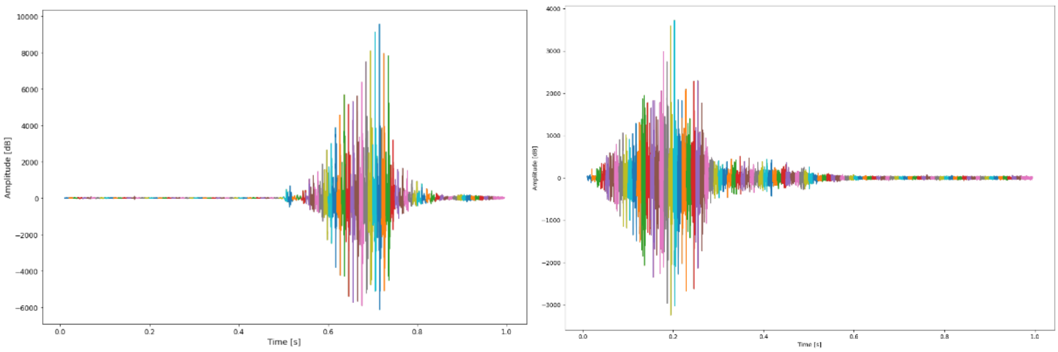


Figure 32 – Framed signal examples

It is obvious, that for the different recordings the frames are representing different part of the command. As an example, frame number 30 in the left picture doesn't include any part of the phoneme, when in the right picture it is located in the middle of it.

5.1.2.3. Hamming Window

Hamming window function is applied to each frame to improve the transition between the frames, reduce the spectrum leakages and improve the frequency resolution. The Hamming window function is the following:

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2 \cdot \pi \cdot n}{N - 1}\right), 0 \leq n \leq N - 1$$

After applying the function, the main aim is to have the same value at the edges of each frame. That is important because is frames is handled by FFT as periodical and if the ends are not matching then discontinuity rises, which leads to spectrum leakages. By leading the end-point values to zero, the information contained in this part of the signal is lost, but thanks to the overlapping segments it is included in the next or previous frame. In the following image (Figure 33) is shown the effect of window applied on a frame a. of the original and b. of the emphasized signal.

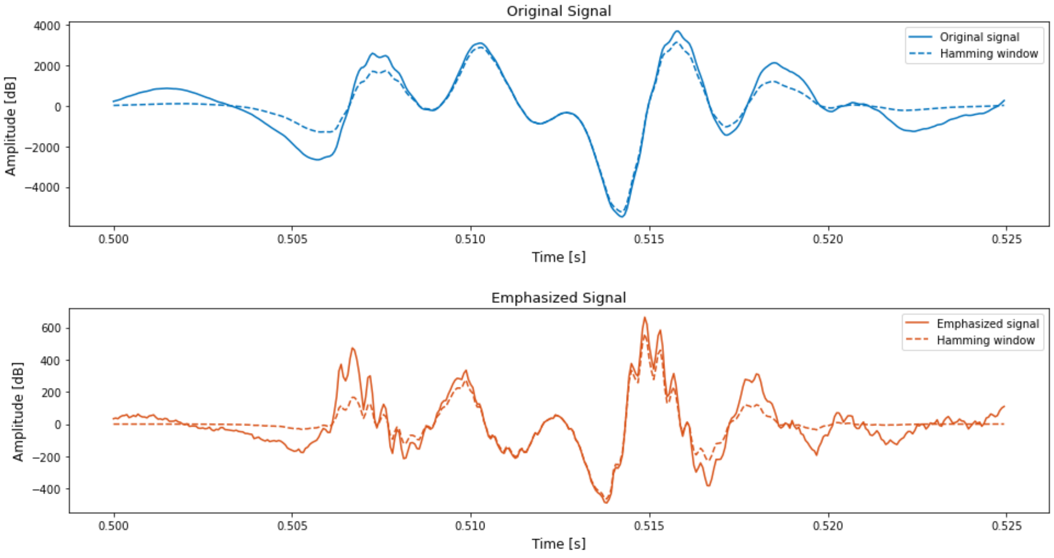


Figure 33 – Window function on original and emphasized signal

It is interesting to see how other window functions are modifying the audio signal. Figure 34 shows the representation of the different functions and the final form of the signal. All of them, except Hamming Window, have zero amplitude at the edges of the frame and maximum in the middle.

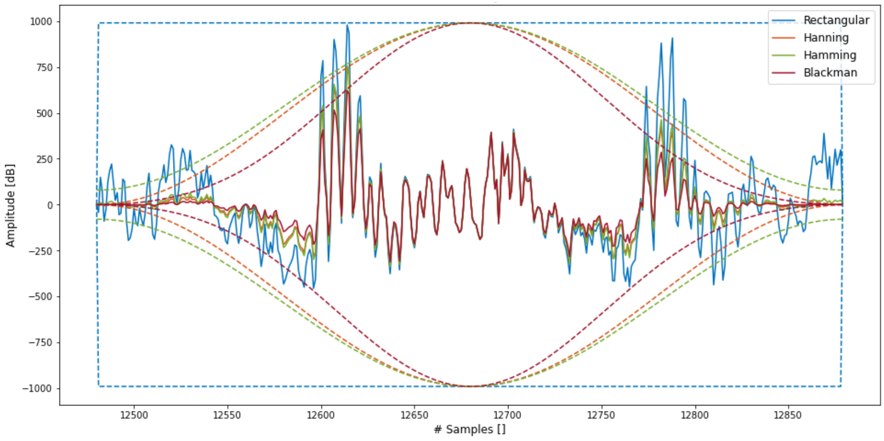


Figure 34 – Window functions

Figure 35 shows the window filter applied in three consecutive frames. It is visible that the parts of the signal, which the amplitude is minimized in one frame, is maximized in the previous or next one, preserving all the necessary information for the classification.

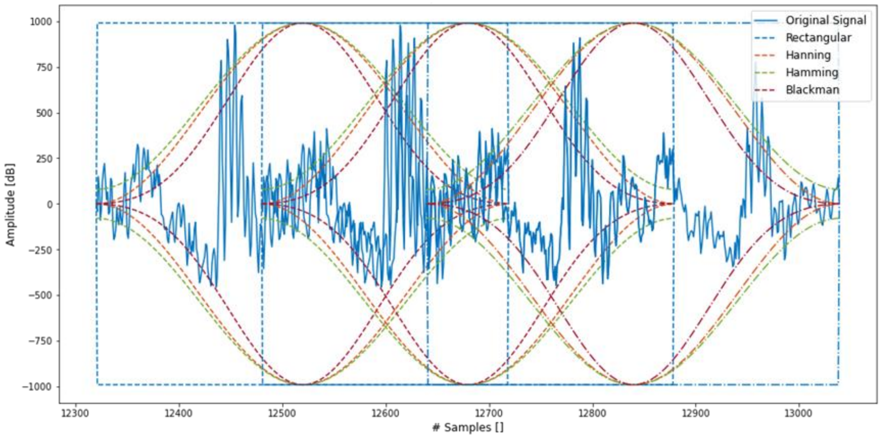


Figure 35 – Window functions

The dataset after the window filter application has the same table dimensions as before: 42277x39200.

5.1.2.4. Fast Fourier Transform (FFT)

Finally, to complete the signal pre-processing the Fast Fourier Transform is applied on each frame to calculate the frequency representation of the time domain signal. The recommended Fourier lengths are 256 or 512; here is chosen 512.

$$X[n] = \sum_{i=0}^{N-1} x[n] \cdot e^{-j\cdot\frac{2\pi}{N}\cdot k\cdot n}$$

In the figures below are some examples of spectrum for different recordings. On the vertical axis is the Frequency [kHertz], on the horizontal is time [sec], or sample number and the color represent different magnitudes. The position relatively with the horizontal axis shows the moment in time that the word was pronounced. The pattern of the color plot shows the energy distribution among different frequencies.

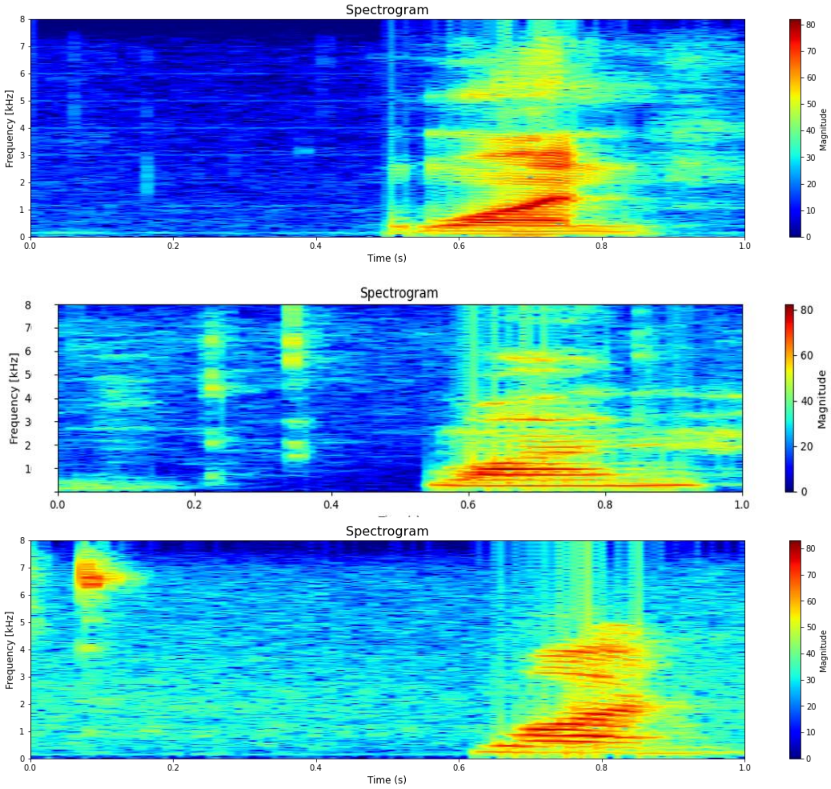


Figure 36 – Frequency Spectrum "one"

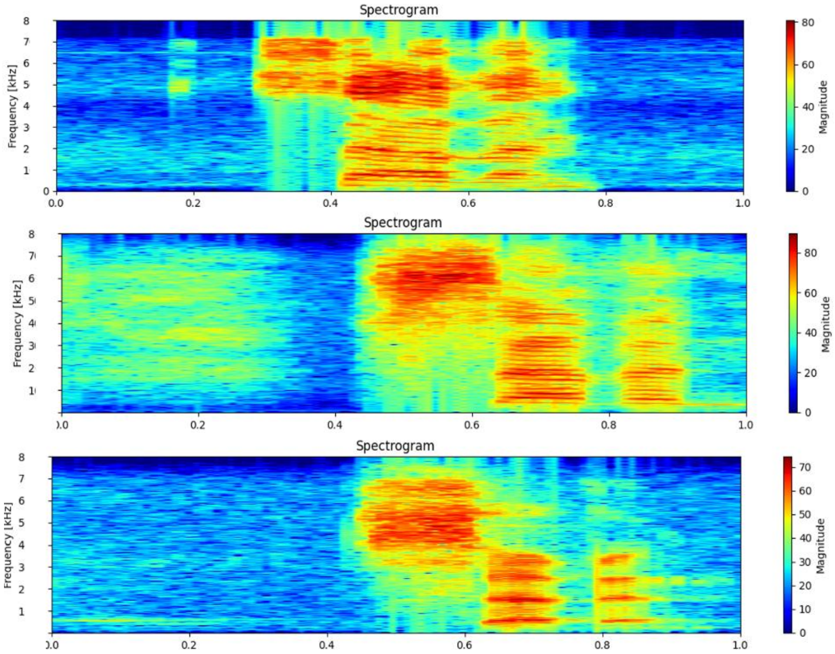


Figure 37 – Frequency Spectrum "seven"

In Figure 36, there are the spectrums for two different recordings of the command "*one*" and in Figure 37the spectrums of the command "*seven*". It's obvious that the spectrums of the same commands follow similar patterns.

5.1.3. Feature extraction

The process from the spectrum to the cepstrum and MFCCs generation is described below.

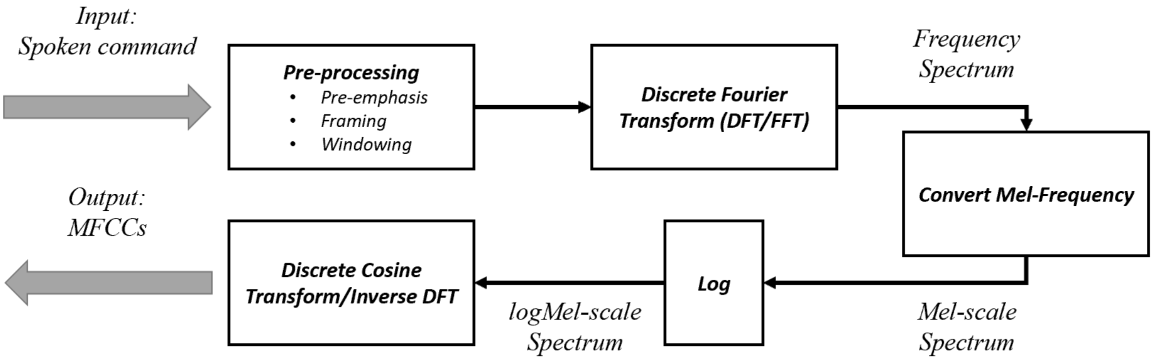


Figure 38 –MFCCs pipeline [26]

First, the frequency is converted from Hertz scale to Mel scale, then the logarithmic function is applied on the Mel spectrum. The logarithmic Mel Spectrum is converted to MFCCs cepstrum with the Discrete Cosine Transform.

From the power spectrum and applying FFT, we have the Mel-spectrum. In the graphs below, is the visualization of Mel-scale spectrum. The same characteristic patterns of each command, are still visible. It is very interesting to see the spectrum representation to vary significantly for different commands and to match for the same commands.

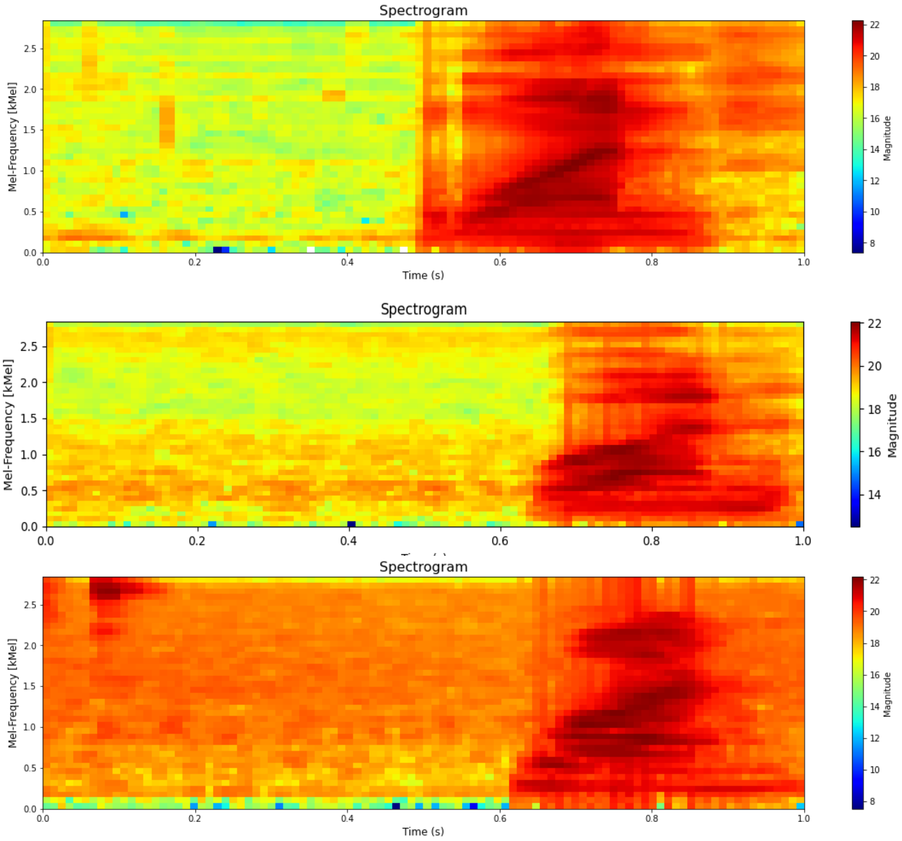


Figure 39 – Mel-frequency Spectrum "one"

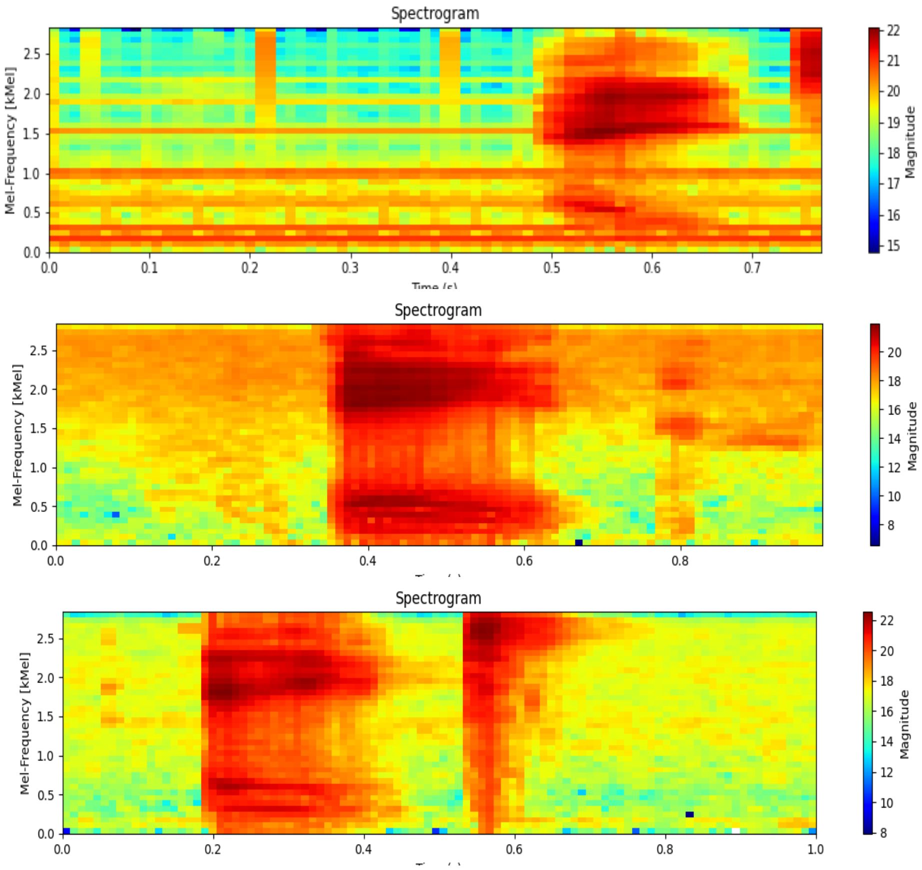


Figure 40 – Mel-frequency Spectrum "eight"

In Figure 41 are shown the frequency spectrum and Mel spectrum for the same recording of the commands "*six*", to look closer to the transformation from one scale to the other.

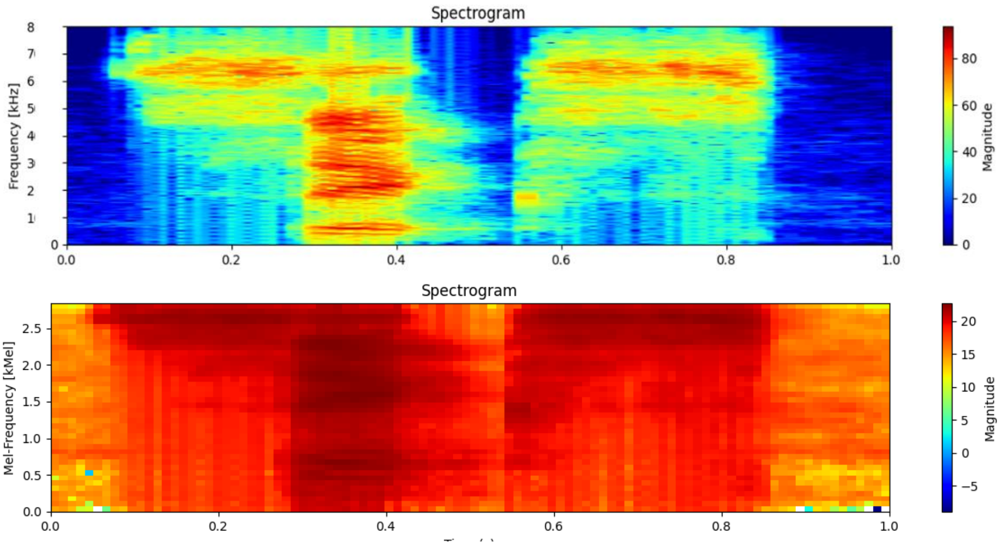


Figure 41 – From Frequency to Mel Spectrum "six"

The spectrum pattern is the same in both graphs, but in Mel spectrum plot the colors the magnitude is more uniform, especial at the high frequency range. As mentioned in previous chapters, the human ear cannot distinguish small variations in high frequencies, the way it can in lowers ones. So, it perceives different, but similar, high frequencies as the same frequency. Let's focus on the top red, high-energy areas, between 6kHz-7kHz and the green, mid energy areas on top, between 7kHz-8kHz. The same are in the Mel spectrum is located above the 2.5 Mel, but instead of having two different colors, the whole are is represented by dark red, which corresponds to high energy. So, the obvious difference in frequency spectrum is not noticeable for humans and thus is not included in the Mel spectrum.

From the Mel-spectrum and by applying logarithmic function and DCT on top, we get the MFCC spectrogram, aka. the cepstrum. The previous spectrums were quite easy to identify the commands by the pattern and it could be a next step to try image recognition on them. When it comes to cepstrum representation, is not that obvious. Still there are some similarities between the MFCCs of the same command, but the identification is not so straightforward.

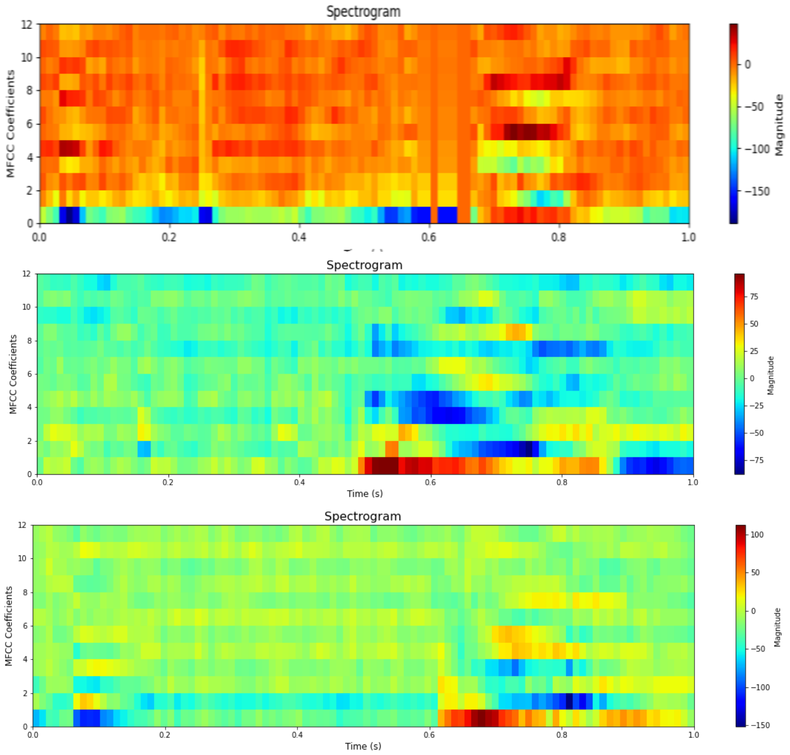


Figure 42 –MFCC – Cepstrum “one”

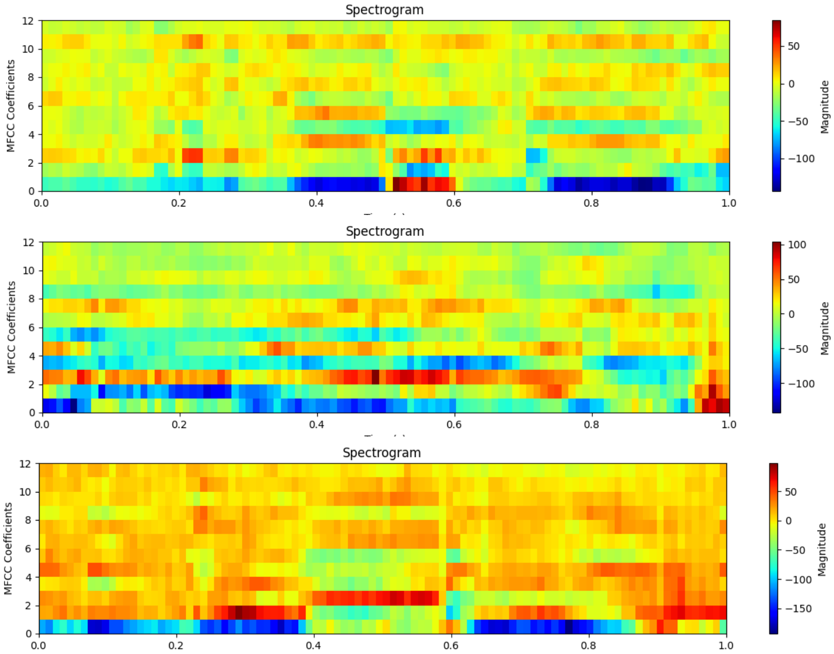


Figure 43 –MFCC – Cepstrum “six”

5.1.4. Classification

After the feature extraction the dataset is ready for the classification stage. Now is the moment of truth, as the classification results will show how good is the data preparation. With the right, optimal network it should be possible to achieve 80%-85% accuracy in the test subset. The classification part is done in MATLAB R2023b and the model is a pattern recognition network (patternnet).

5.1.4.1. k-Fold Cross Validation

As mentioned before, the k-fold cross validation method is used to obtain more representative accuracy of the ANN and have less bias. When a network is biased, it means that certain factors have high influence on it and the results are skewed towards a specific direction. From bibliography the recommended values are between 5 and 10 folds. There is also the approach of n-fold cross validation or Leave-One-Out Cross Validation (LOOCV), where the number of folds is equal to the number of observations. In this approach the testing set each time consists of only one observation and the process is repeated n times. This approach has the least biased results but is very demanding in computational power and very time consuming. It is chosen k=5, so the training set consists of 33822 observations and the testing set of 8455. The inputs are the 12 coefficients for each frame, for 98 frames and the classes are the 18 different commands.

It is important to check that the distribution of commands in both train and test subsets are in similar levels. If the training set includes uneven number of each word, the network will be obviously biased towards some class. The training subsets contains 80% of the total recordings. All different commands must be present in approximately same percentage, so the network is equally trained to recognize all commands and doesn't tent to classify towards certain words. Similarly, for the test subset, that contains 20% of the total recordings, the commands should be equally distributed. It is expected that in training set there are 1600 recordings of each word and 400 in test set.

5.1.4.2. Artificial Neural Network Parameters

The main investigation is regarding the network's architecture, with constant training function and stop criteria. The network has two hidden layers, as they are judged to be enough for this problem, and the number of hidden layers in each one is the object of the optimization. The training function used is the Scaled Conjugated Gradient algorithm (SCG). The SCG method is an evolution of the conjugated methods, but instead of using line search to define the optimal step, it calculates the interval based on a size scaling mechanism, improving the method's efficiency. Additionally, it is using the Hessian matrix with second order information, unlike first order simple gradient method, accelerating the convergence. The maximum number of epochs is set to 500 to have a good balance of high accuracy and low computational time. The output from the feed forward network is the possibility, the observation to belong in each one of the 18 classes, but the desired output is the written command, so the class label. The transfer function "SoftMax", does this job, by selecting the class with the highest probability.

Table 13 – Initial ANN characteristics

Type	Pattern network
k-fold cross validation	k = 5
Hidden layers	2
Training function	Scaled Conjugate Gradient ('trainscg')
Train ratio	80%
Test ratio	20%
Transfer function	SoftMax function ('softmax')
Maximum number of epochs	500 epochs

5.1.4.3. Artificial Neural Network Investigation

The investigation of the network architecture is done with the brute force method; manually "all" of the possible architectures are examined and the one with the best accuracy is selected in the final model. The main criteria for the selection is the accuracy to be above 80%, but for the final network also the evaluation metrics are calculated, to evaluate deeper its performance. The concept is to start from simple, small networks and continue with larger, more complex ones until the desired accuracy is achieved. This way, the selected architecture will be almost the least complex possible.

As first step, the different networks are trained in only ten words, the digits from 0 to 9, starting from small architectures with 30 neurons in each layer.

Table 14 – Speech recognition models – Only Digits

k = 5	*30x30*	*40x40*	*50x50*	*60x60*	*80x80*	*100x100*
Fold No 1	69.5%	73.5%	75.8%	78.3%	82.0%	83.7%
Fold No 2	61.5%	73.9%	75.4%	79.5%	80.2%	81.6%
Fold No 3	69.2%	73.3%	76.7%	78.0%	80.8%	75.3%
Fold No 4	69.6%	75.6%	76.9%	79.0%	80.6%	82.2%
Fold No 5	70.6%	73.5%	76.0%	78.7%	80.1%	82.0%
Average	***68.1%***	***74.0%***	***76.1%***	***78.7%***	***80.7%***	***80.9%***

Even the 30x30 network has quite good accuracy (~70%) and by increasing the neurons to 100x100 the accuracy exceeds the target (80%). By investigating more complicated architectures, 300x300 networks, the accuracy increases to almost 90%. In all cases the train accuracy is between 95% and 100%.

Table 15 – Speech recognition models – Only Digits

k = 5	*120x120*	*140x140*	*180x180*	*250x250*	*300x300*
Fold No 1	84.3%	85.2%	86.3%	86.4%	88.7%
Fold No 2	84.5%	84.7%	86.4%	87.9%	88.0%
Fold No 3	83.8%	84.9%	85.2%	86.6%	86.9%
Fold No 4	84.3%	85.8%	86.3%	87.0%	86.8%
Fold No 5	83.8%	84.8%	85.4%	86.0%	87.2%
Average	***84.1%***	***85.1%***	***85.9%***	***86.8%***	***87.5%***

The next step is to add more commands for classification, expect the digits the words "*Left*", "*Right*", "*Stop*", "*On*" and "*Off*" are added to the dataset, increasing the classes from 10 to 17. The expansions of the vocabulary – almost double the initial classes – leads to accuracy drop approximately 10% for the same network architectures.

Table 16 – Speech recognition models – 17 Commands

k = 5	*80x80*	*100x100*	*120x120*	*140x140*	*180x180*	*250x250*
Fold No 1	*67.5%*	*70.7%*	*74.2%*	*73.4%*	*77.5%*	*78.6%*
Fold No 2	*69.3%*	*71.4%*	*73.3%*	*74.3%*	*76.4%*	*78.9%*
Fold No 3	*68.0%*	*72.6%*	*75.2%*	*75.9%*	*76.8%*	*81.1%*
Fold No 4	*67.8%*	*71.7%*	*74.4%*	*76.1%*	*76.6%*	*74.3%*
Fold No 5	*67.2%*	*69.8%*	*73.4%*	*74.6%*	*77.8%*	*79.4%*
Average	***68.0%***	***71.2%***	***74.1%***	***74.9%***	***77.0%***	***78.5%***

The command “Go” is added in the dataset, to form the final version of the vocabulary. With the new dataset, the networks are trained again. For the same architecture, with the addition of one extra command, the accuracy drops by1%-2%, but is still able to catch the target (80%), with the 300x300 model.

Table 17 – Speech recognition models – Final Dataset

k = 5	*120x120*	*140x140*	*180x180*	*250x250*	*300x300*
Fold No 1	*73.3%*	*74.1%*	*74.6%*	*77.8%*	*80.3%*
Fold No 2	*70.5%*	*74.3%*	*74.5%*	*76.3%*	*81.1%*
Fold No 3	*73.5%*	*74.5%*	*73.6%*	*74.9%*	*80.4%*
Fold No 4	*71.0%*	*74.0%*	*76.1%*	*80.2%*	*81.0%*
Fold No 5	*73.4%*	*74.8%*	*76.5%*	*77.0%*	*81.2%*
Average	***72.4%***	***74.3%***	***75.1%***	***77.2%***	***80.8%***

The results from this initial study, set the foundations and limits for the next steps of the investigation. By increasing the complexity of the network structure, the improvement in the testing accuracy is still noticeable, which shows that the direction is correct and the absolute best performance hasn’t been achieved yet. Of course, in smaller architectures the benefit in accuracy is bigger with the same number of neurons increase, compared with the benefit in larger networks. It is also clear, that there is not point of using networks with less than 100 neurons per layer, as their accuracy is less than 70%. The networks, so far, are of squared setup, meaning that both layers have the same number of hidden neurons. For the next step, the brute force method is applied for architectures between 100x100 and 850x850, with interval of 50 neurons.

Table 18:ANN accuracy investigation – part 1

		Hidden Layer #2									
		100	***150***	***200***	***250***	***300***	***350***	***400***	***450***	***500***	***Mean***
Hidden Layer #1	***100***	70.70%	71.03%	72.59%	71.78%	72.99%	71.73%	73.98%	73.40%	74.96%	***72.57%***
	150	72.32%	74.49%	75.03%	74.96%	75.86%	75.27%	75.76%	75.34%	76.84%	***75.10%***
	200	75.63%	75.81%	76.73%	77.73%	76.17%	76.00%	78.16%	76.78%	77.02%	***76.67%***
	250	77.29%	77.31%	77.56%	77.98%	77.93%	78.92%	78.23%	79.13%	77.12%	***77.94%***
	300	78.02%	77.85%	77.16%	77.53%	78.13%	77.92%	77.41%	79.39%	79.34%	***78.08%***
	350	78.17%	78.58%	77.76%	78.80%	78.88%	75.57%	79.13%	78.25%	79.70%	***78.32%***
	400	78.89%	78.83%	79.28%	80.23%	80.16%	79.60%	79.57%	79.87%	78.61%	***79.45%***
	450	79.17%	80.15%	78.70%	79.35%	79.26%	79.43%	80.16%	80.07%	80.60%	***79.65%***
	500	79.63%	79.81%	80.57%	79.37%	80.00%	79.96%	79.77%	78.06%	79.77%	***79.66%***
	550	79.9%	80.2%	80.2%	80.4%	78.8%	79.6%	80.5%	79.9%	80.1%	***79.96%***
	600	79.9%	79.7%	80.2%	79.1%	79.1%	79.1%	80.9%	79.5%	80.6%	***79.79%***
	650	80.2%	80.6%	80.7%	79.4%	80.2%	80.2%	80.7%	81.4%	78.0%	***80.14%***
	Mean	***77.49%***	***77.86%***	***78.04%***	***78.05%***	***78.12%***	***77.77%***	***78.68%***	***78.43%***	***78.56%***	

In Table 18 are summarized the results for different architecture combinations. The number of neurons in layers 1 is set as n_1 and n_2 is the number of neurons in second layer. The columns are of constant n_2 and different n_1, and the rows are of constant n_1 and different n_2. In the first table are the results for $n_1 \in [100,650]$ and $n_2 \in [100,400]$. For each column and row, the average accuracy is calculated. The conclusions from these results are:

- The minimum accuracy, 70%, occurs for architecture 100x100.
- The maximum accuracy is around 80%-81% and exists for different combinations.
- The increase in layer 1 size is more effective than the same increase in layer 2. By observing the average per row, the accuracy improves, with incremental increase of n_1, more in the first steps (between 100 and 300 almost 2% per increment), less in the middle (0.5% until 500 neurons) and in the end the delta is almost zero. The average value per column, doesn't follow the same trend, and is almost same throughout for all different sizes of layer 2 and equal to 78%.
- The same change (e.g., increase layer 1 by 50 neurons) is more beneficial in smaller architectures, than in more complex ones. This makes sense, when reaching closer to the maximum performance the potentials are less.
- The best accuracy can be achieved with several combinations.
    - n_1=400, it should be $n_2 \geq 300$
    - n_1=500 or n_1=600, it should be $n_2 \geq 200$

Table 19:ANN accuracy investigation – part 2

		Hidden Layer #2										
		400	***450***	***500***	***550***	***600***	***650***	***700***	***750***	***800***	***800***	***Mean***
Hidden Layer #1	***550***	80.5%	79.9%	80.1%	79.6%	81.3%	81.0%	81.2%	80.8%	80.6%	80.6%	***80.65%***
	600	80.9%	79.5%	80.6%	81.0%	80.5%	81.6%	80.9%	81.7%	81.3%	81.3%	***80.95%***
	650	80.7%	81.4%	78.0%	80.4%	80.1%	78.0%	79.9%	79.1%	78.5%	78.5%	***79.65%***
	700	80.7%	81.4%	78.0%	80.4%	80.1%	81.7%	79.5%	81.6%	80.0%	80.0%	***80.43%***
	750	80.2%	81.5%	81.2%	81.0%	79.2%	81.8%	81.6%	81.0%	81.1%	81.1%	***80.79%***
	800	80.3%	79.7%	81.7%	81.1%	80.4%	81.4%	81.7%	81.9%	81.7%	82.1%	***81.01%***
	Mean	***80.30%***	***80.32%***	***79.77%***	***80.59%***	***80.26%***	***80.91%***	***80.81%***	***81.01%***	***80.32%***	***80.76%***	

Table 18Table 19 includes combination the trends and findings are similar. For these, more complex architectures, the accuracy can reach 82%, but the improvement is not that big; for double size of the network (from 400x300 to 800x500) the benefit is only 2%.

It seems that the accuracy follows a trend similar to logarithmic, from n_1=100 until n_1=400 the accuracy delta is 10%, and from n_1=400 until n_1=800 only 2%. The model's sensitivity to the size the second hidden layer, is of similar trend as with the first layer size, but scaled down, to smaller deltas. From the results is clear that the performance of the artificial neural networks is affected more by the first layer size, than the rest. That is reasonable, considering that the information flow is from the input layer to the first hidden layer and then to second hidden layer. If the first layer doesn't have the correct structure, the performance will be limited by default. The following graphs show the same trend.

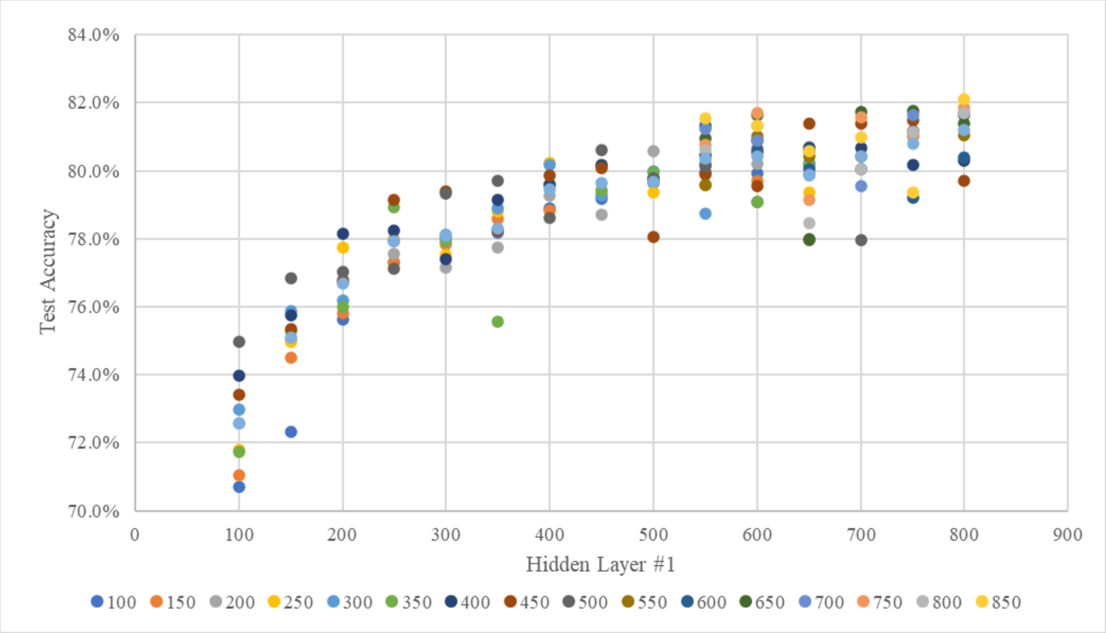


Figure 44 –Test Accuracy for different sizes of hidden layer #1

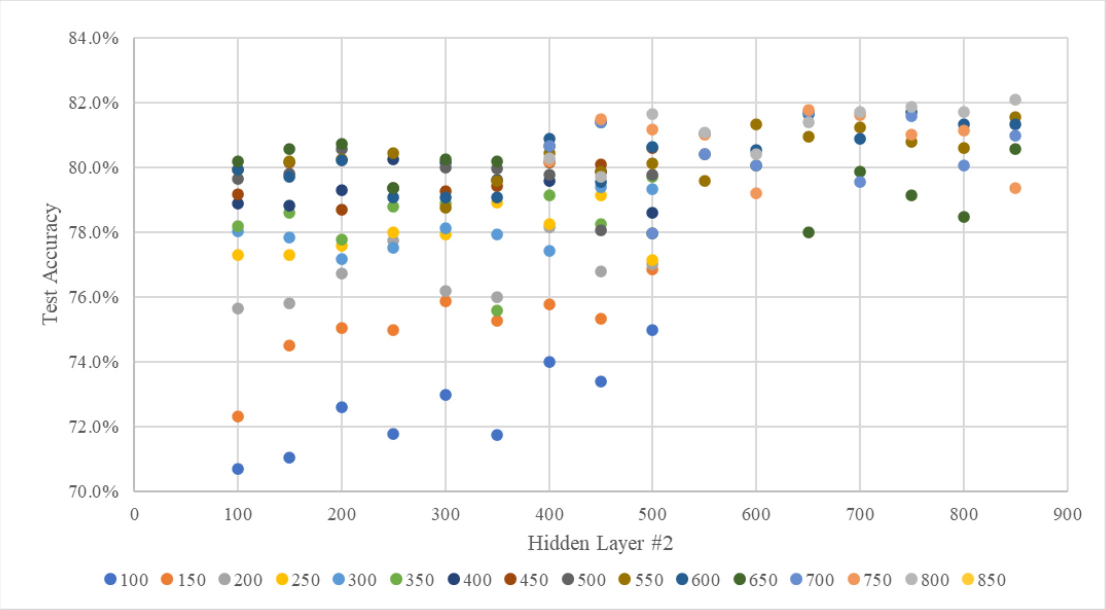


Figure 45 – Test Accuracy for different sizes of hidden layer #2

Figure 44 demonstrates the relation between accuracy (vertical axis) and first layer's size (horizontal axis). The different colours represent different numbers of neurons in layer #2. In Figure 45 is plotted the network's accuracy for variant size of layer #2. Here the different colours stand for different size of layer #1. Comparing the two figures it's obvious that the relation with the overall accuracy is very different: the second hidden layer doesn't have the same influence as the first. In general, it's reasonable to assume that the first layer's structure sets the accuracy level and the increase in the number of neurons in second level does the finetuning.

From the investigation, is not clear which architecture is the best, since the requirement for 80% accuracy, can be achieved with different combinations. It is worth to examine mire that one options further and use the evaluation matrix to take the final decision.

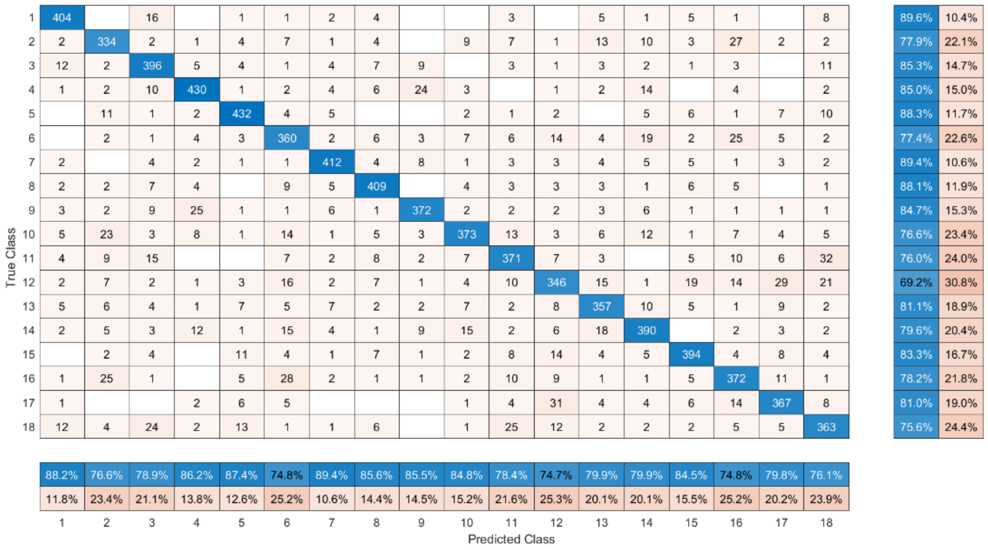
5.1.4.4. Evaluation Metrics – ANN 600x450

Here are the classification results for the *600x450* network. The overall accuracy is 80.7%, which remains at the same level in all five folds. The performance is very balanced, as all the evaluation metrics are very close to each other. In Table 20 are all the metrics for this network.

Table 20 – Test Accuracy 600x450

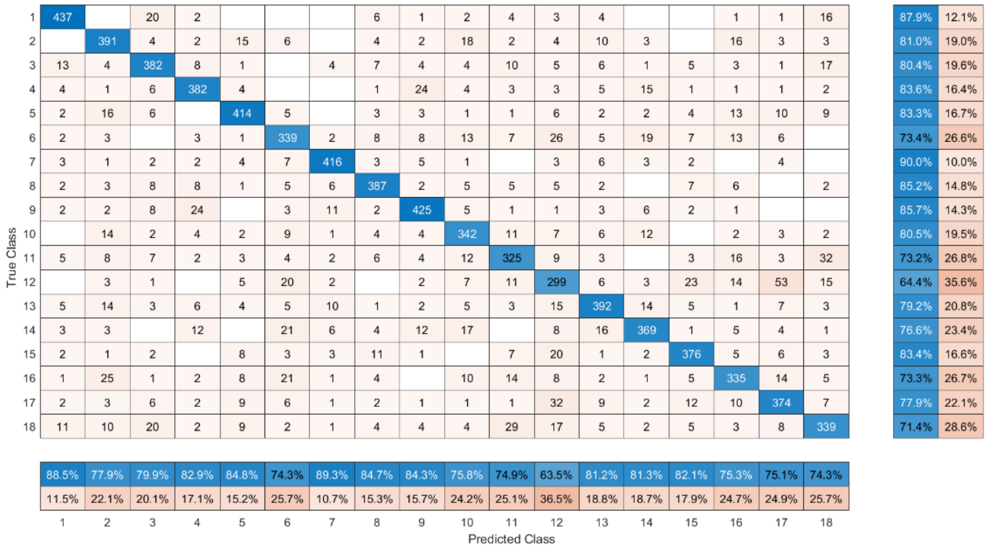
	Fold No 1	Fold No 2	Fold No 3	Fold No 4	Fold No 5	Average
Accuracy	81.4%	79.5%	80.5%	80.9%	81.1%	**80.7%**
Precision	80.1%	78.5%	80.3%	81.0%	81.1%	**80.7%**
Recall	79.7%	80.3%	78.5%	79.9%	79.4%	**80.7%**
F1-score	79.8%	79.3%	79.4%	80.4%	81.2%	**80.7%**

The confusion matrices for three, out of five folds, are shown in the graphs below and give an indication for the recognition balancing for the different commands. Their behaviour is very similar.



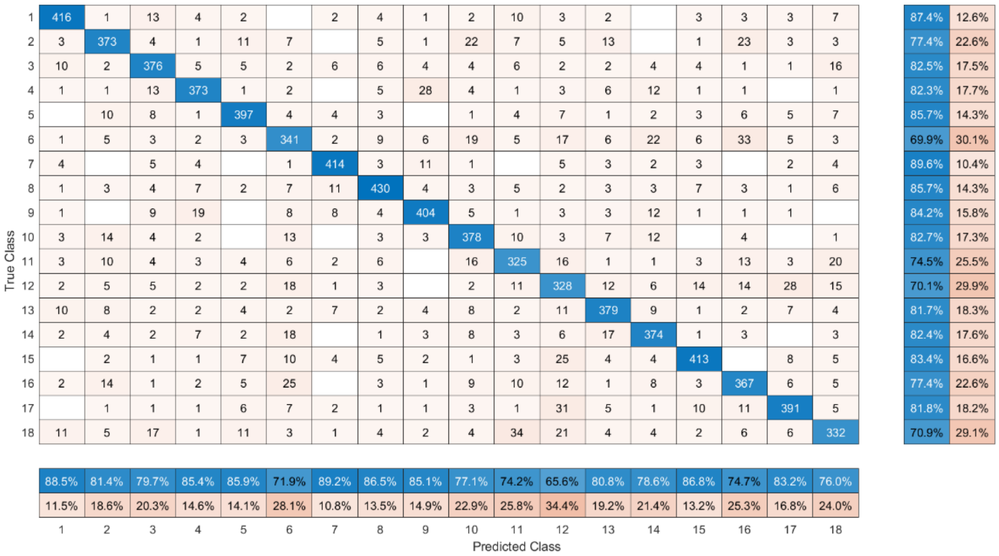
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		
1	404		16		1	1	2	4			3		5	1	5	1		8	89.6%	10.4%
2	2	334	2	1	4	7	1	4		9	7	1	13	10	3	27	2	2	77.9%	22.1%
3	12	2	396	5	4	1	4	7	9		3	1	3	2	1	3		11	85.3%	14.7%
4	1	2	10	430	1	2	4	6	24	3		1	2	14		4		2	85.0%	15.0%
5		11	1	2	432	4	5			2	1	2		5	6	1	7	10	88.3%	11.7%
6		2	1	4	3	360	2	6	3	7	6	14	4	19	2	25	5	2	77.4%	22.6%
7	2		4	2	1	1	412	4	8	1	3	3	4	5	5	1	3	2	89.4%	10.6%
8	2	2	7	4		9	5	409		4	3	3	3	1	6	5		1	88.1%	11.9%
9	3	2	9	25	1	1	6	1	372	2	2	2	3	6	1	1	1	1	84.7%	15.3%
10	5	23	3	8	1	14	1	5	3	373	13	3	6	12	1	7	4	5	76.6%	23.4%
11	4	9	15			7	2	8	2	7	371	7	3		5	10	6	32	76.0%	24.0%
12	2	7	2	1	3	16	2	7	1	4	10	346	15	1	19	14	29	21	69.2%	30.8%
13	5	6	4	1	7	5	7	2	2	7	2	8	357	10	5	1	9	2	81.1%	18.9%
14	2	5	3	12	1	15	4	1	9	15	2	6	18	390		2	3	2	79.6%	20.4%
15		2	4		11	4	1	7	1	2	8	14	4	5	394	4	8	4	83.3%	16.7%
16	1	25	1		5	28	2	1	1	2	10	9	1	1	5	372	11	1	78.2%	21.8%
17	1			2	6	5				1	4	31	4	4	6	14	367	8	81.0%	19.0%
18	12	4	24	2	13	1	1	6		1	25	12	2	2	2	5	5	363	75.6%	24.4%
	88.2%	76.6%	78.9%	86.2%	87.4%	74.8%	89.4%	85.6%	85.5%	84.8%	78.4%	74.7%	79.9%	79.9%	84.5%	74.8%	79.8%	76.1%		
	11.8%	23.4%	21.1%	13.8%	12.6%	25.2%	10.6%	14.4%	14.5%	15.2%	21.6%	25.3%	20.1%	20.1%	15.5%	25.2%	20.2%	23.9%		

Figure 46 – Confusion matrix k-fold1 – 600x450



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		
1	437		20	2				6	1	2	4	3	4			1	1	16	87.9%	12.1%
2		391	4	2	15	6		4	2	18	2	4	10	3		16	3	3	81.0%	19.0%
3	13	4	382	8	1		4	7	4	4	10	5	6	1	5	3	1	17	80.4%	19.6%
4	4	1	6	382	4			1	24	4	3	3	5	15	1	1	1	2	83.6%	16.4%
5	2	16	6		414	5		3	3	1	1	6	2	2	4	13	10	9	83.3%	16.7%
6	2	3		3	1	339	2	8	8	13	7	26	5	19	7	13	6		73.4%	26.6%
7	3	1	2	2	4	7	416	3	5	1		3	6	3	2		4		90.0%	10.0%
8	2	3	8	8	1	5	6	387	2	5	5	5	2		7	6		2	85.2%	14.8%
9	2	2	8	24		3	11	2	425	5	1	1	3	6	2	1			85.7%	14.3%
10		14	2	4	2	9	1	4	4	342	11	7	6	12		2	3	2	80.5%	19.5%
11	5	8	7	2	3	4	2	6	4	12	325	9	3		3	16	3	32	73.2%	26.8%
12		3	1		5	20	2		2	7	11	299	6	3	23	14	53	15	64.4%	35.6%
13	5	14	3	6	4	5	10	1	2	5	3	15	392	14	5	1	7	3	79.2%	20.8%
14	3	3		12		21	6	4	12	17		8	16	369	1	5	4	1	76.6%	23.4%
15	2	1	2		8	3	3	11	1		7	20	1	2	376	5	6	3	83.4%	16.6%
16	1	25	1	2	8	21	1	4		10	14	8	2	1	5	335	14	5	73.3%	26.7%
17	2	3	6	2	9	6	1	2	1	1	1	32	9	2	12	10	374	7	77.9%	22.1%
18	11	10	20	2	9	2	1	4	4	4	29	17	5	2	5	3	8	339	71.4%	28.6%
	88.5%	77.9%	79.9%	82.9%	84.8%	74.3%	89.3%	84.7%	84.3%	75.8%	74.9%	63.5%	81.2%	81.3%	82.1%	75.3%	75.1%	74.3%		
	11.5%	22.1%	20.1%	17.1%	15.2%	25.7%	10.7%	15.3%	15.7%	24.2%	25.1%	36.5%	18.8%	18.7%	17.9%	24.7%	24.9%	25.7%		

Figure 47 – Confusion matrix k-fold2 – 600x450



1	416	1	13	4	2		2	4	1	2	10	3	2		3	3	3	7	87.4%	12.6%
2	3	373	4	1	11	7		5	1	22	7	5	13		1	23	3	3	77.4%	22.6%
3	10	2	376	5	5	2	6	6	4	4	6	2	2	4	4	1	1	16	82.5%	17.5%
4	1	1	13	373	1	2		5	28	4	1	3	6	12	1	1		1	82.3%	17.7%
5		10	8	1	397	4	4	3		1	4	7	1	2	3	6	5	7	85.7%	14.3%
6	1	5	3	2	3	341	2	9	6	19	5	17	6	22	6	33	5	3	69.9%	30.1%
7	4		5	4		1	414	3	11	1		5	3	2	3		2	4	89.6%	10.4%
8	1	3	4	7	2	7	11	430	4	3	5	2	3	3	7	3	1	6	85.7%	14.3%
9	1		9	19		8	8	4	404	5	1	3	3	12	1	1	1		84.2%	15.8%
10	3	14	4	2		13		3	3	378	10	3	7	12		4		1	82.7%	17.3%
11	3	10	4	3	4	6	2	6		16	325	16	1	1	3	13	3	20	74.5%	25.5%
12	2	5	5	2	2	18	1	3		2	11	328	12	6	14	14	28	15	70.1%	29.9%
13	10	8	2	2	4	2	7	2	4	8	2	11	379	9	1	2	7	4	81.7%	18.3%
14	2	4	2	7	2	18		1	3	8	3	6	17	374	1	3		3	82.4%	17.6%
15		2	1	1	7	10	4	5	2	1	3	25	4	4	413		8	5	83.4%	16.6%
16	2	14	1	2	5	25		3	1	9	10	12	1	8	3	367	6	5	77.4%	22.6%
17		1	1	1	6	7	2	1	1	3	1	31	5	1	10	11	391	5	81.8%	18.2%
18	11	5	17	1	11	3	1	4	2	4	34	21	4	4	2	6	6	332	70.9%	29.1%
	88.5%	81.4%	79.7%	85.4%	85.9%	71.9%	89.2%	86.5%	85.1%	77.1%	74.2%	65.6%	80.8%	78.6%	86.8%	74.7%	83.2%	76.0%		
	11.5%	18.6%	20.3%	14.6%	14.1%	28.1%	10.8%	13.5%	14.9%	22.9%	25.8%	34.4%	19.2%	21.4%	13.2%	25.3%	16.8%	24.0%		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		

Figure 48 – Confusion matrix k-fold3 – 600x450

The individual classes have a deviation of 10% in their accuracy. The accuracies are in general above 75%, but in all different folds, there are three, approximately, classes that their accuracy is around 70%. The miss-predictions seems to be random and not follow any pattern. For example, it would be expected to confuse the commands "*one*" (No2) and "*on*" (No16), or "*on*" (No16) and "*off*" (No17), as the sound can be similar, but the model doesn't show this kind of sensitivity. In all cases, the word "*zero*" (No1) is one of the best predicted classes, where "*down*" (No12) is one of the worst.

The commands "*two*" and "*three*" have very good percentages, much higher than "*five*" or "*down*". When the user is using the model, this difference in accuracy would rise, and in some words the classification performance would be lower than in others.

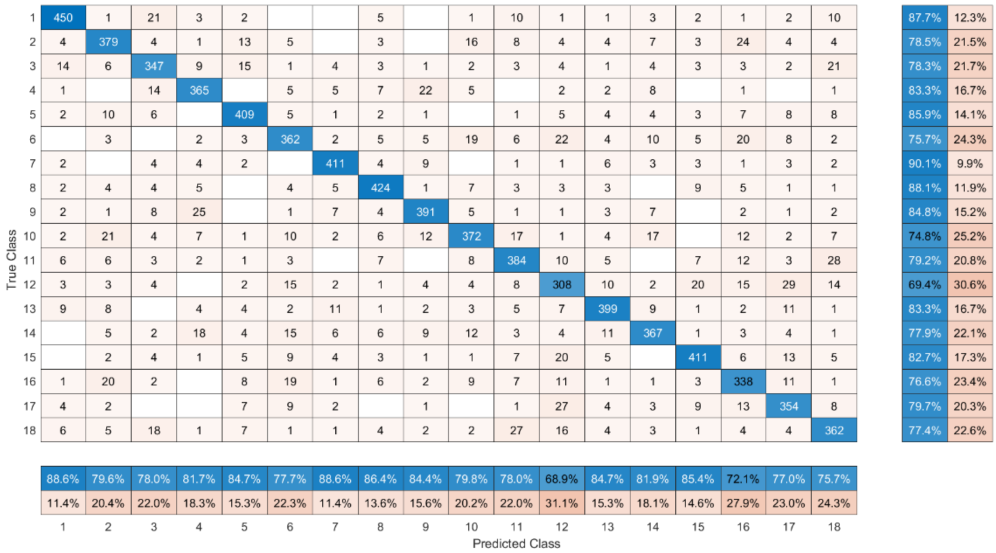
5.1.4.5. Evaluation Metrics – ANN 500x200

Here are the classification results for the *500x200* network. The overall accuracy is 79.7%. All the metrics, for each fold, all on the exact same level, which is a bit strange to be that close (within 0.2%).

Table 21 – Test Accuracy 500x200

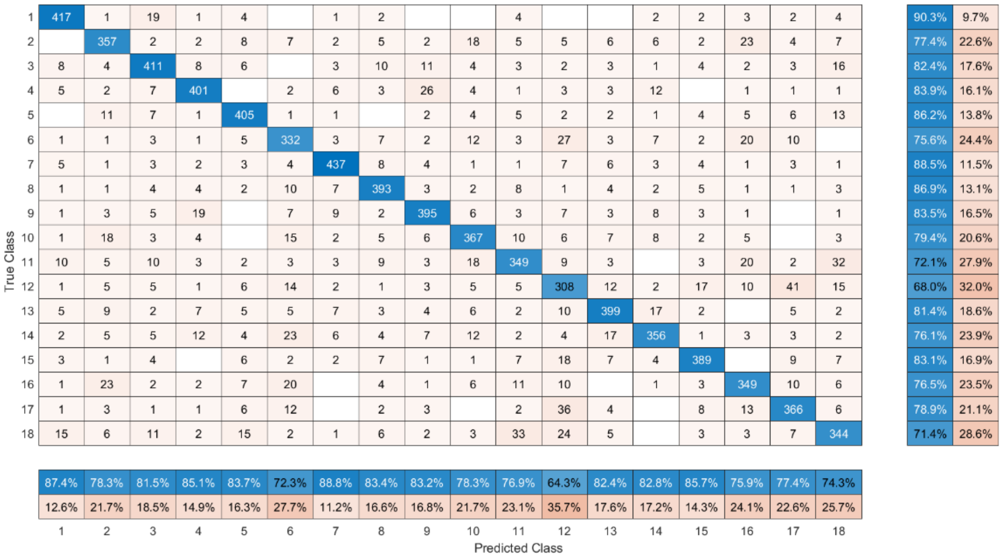
	Fold No 1	*Fold No 2*	*Fold No 3*	*Fold No 4*	*Fold No 5*	***Average***
Accuracy	*80.2%*	*78.4%*	*80.1%*	*80.1%*	*79.9%*	***79.7%***
Precision	*80.7%*	*78.2%*	*80.2%*	*80.1%*	*79.9%*	*79.8%*
Recall	*80.8%*	*78.3%*	*80.2%*	*80.1%*	*79.9%*	*79.8%*
F1-score	80.7%	78.2%	80.2%	80.1%	79.9%	*79.8%*

The confusion matrices for three, out of five folds, are shown in the graphs below and give an indication for the recognition balancing for the different commands.



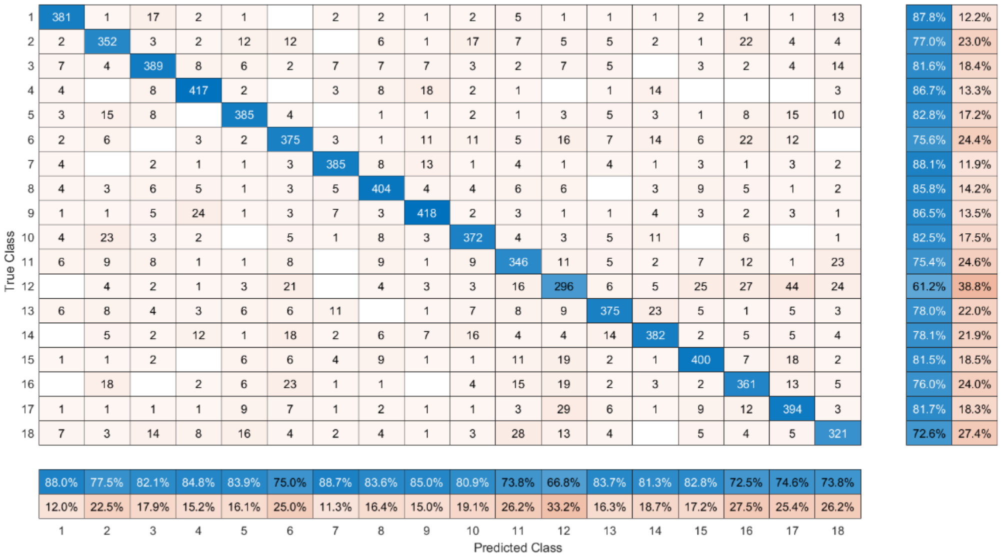
True Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		
1	450	1	21	3	2			5		1	10	1	1	3	2	1	2	10	87.7%	12.3%
2	4	379	4	1	13	5		3		16	8	4	4	7	3	24	4	4	78.5%	21.5%
3	14	6	347	9	15	1	4	3	1	2	3	4	1	4	3	3	2	21	78.3%	21.7%
4	1		14	365		5	5	7	22	5		2	2	8		1		1	83.3%	16.7%
5	2	10	6		409	5	1	2	1		1	5	4	4	3	7	8	8	85.9%	14.1%
6		3		2	3	362	2	5	5	19	6	22	4	10	5	20	8	2	75.7%	24.3%
7	2		4	4	2		411	4	9		1	1	6	3	3	1	3	2	90.1%	9.9%
8	2	4	4	5		4	5	424	1	7	3	3	3		9	5	1	1	88.1%	11.9%
9	2	1	8	25		1	7	4	391	5	1	1	3	7		2	1	2	84.8%	15.2%
10	2	21	4	7	1	10	2	6	12	372	17	1	4	17		12	2	7	74.8%	25.2%
11	6	6	3	2	1	3		7		8	384	10	5		7	12	3	28	79.2%	20.8%
12	3	3	4		2	15	2	1	4	4	8	308	10	2	20	15	29	14	69.4%	30.6%
13	9	8		4	4	2	11	1	2	3	5	7	399	9	1	2	11	1	83.3%	16.7%
14		5	2	18	4	15	6	6	9	12	3	4	11	367	1	3	4	1	77.9%	22.1%
15		2	4	1	5	9	4	3	1	1	7	20	5		411	6	13	5	82.7%	17.3%
16	1	20	2		8	19	1	6	2	9	7	11	1	1	3	338	11	1	76.6%	23.4%
17	4	2			7	9	2		1		1	27	4	3	9	13	354	8	79.7%	20.3%
18	6	5	18	1	7	1	1	4	2	2	27	16	4	3	1	4	4	362	77.4%	22.6%
	88.6%	79.6%	78.0%	81.7%	84.7%	77.7%	88.6%	86.4%	84.4%	79.8%	78.0%	68.9%	84.7%	81.9%	85.4%	72.1%	77.0%	75.7%		
	11.4%	20.4%	22.0%	18.3%	15.3%	22.3%	11.4%	13.6%	15.6%	20.2%	22.0%	31.1%	15.3%	18.1%	14.6%	27.9%	23.0%	24.3%		

Figure 49 – Confusion matrix k-fold1 – 500x200



True Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		
1	417	1	19	1	4		1	2			4			2	2	3	2	4	90.3%	9.7%
2		357	2	2	8	7	2	5	2	18	5	5	6	6	2	23	4	7	77.4%	22.6%
3	8	4	411	8	6		3	10	11	4	3	2	3	1	4	2	3	16	82.4%	17.6%
4	5	2	7	401		2	6	3	26	4	1	3	3	12		1	1	1	83.9%	16.1%
5		11	7	1	405	1	1		2	4	5	2	2	1	4	5	6	13	86.2%	13.8%
6	1	1	3	1	5	332	3	7	2	12	3	27	3	7	2	20	10		75.6%	24.4%
7	5	1	3	2	3	4	437	8	4	1	1	7	6	3	4	1	3	1	88.5%	11.5%
8	1	1	4	4	2	10	7	393	3	2	8	1	4	2	5	1	1	3	86.9%	13.1%
9	1	3	5	19		7	9	2	395	6	3	7	3	8	3	1		1	83.5%	16.5%
10	1	18	3	4		15	2	5	6	367	10	6	7	8	2	5		3	79.4%	20.6%
11	10	5	10	3	2	3	3	9	3	18	349	9	3		3	20	2	32	72.1%	27.9%
12	1	5	5	1	6	14	2	1	3	5	5	308	12	2	17	10	41	15	68.0%	32.0%
13	5	9	2	7	5	5	7	3	4	6	2	10	399	17	2		5	2	81.4%	18.6%
14	2	5	5	12	4	23	6	4	7	12	2	4	17	356	1	3	3	2	76.1%	23.9%
15	3	1	4		6	2	2	7	1	1	7	18	7	4	389		9	7	83.1%	16.9%
16	1	23	2	2	7	20		4	1	6	11	10		1	3	349	10	6	76.5%	23.5%
17	1	3	1	1	6	12		2	3		2	36	4		8	13	366	6	78.9%	21.1%
18	15	6	11	2	15	2	1	6	2	3	33	24	5		3	3	7	344	71.4%	28.6%
	87.4%	78.3%	81.5%	85.1%	83.7%	72.3%	88.8%	83.4%	83.2%	78.3%	76.9%	64.3%	82.4%	82.8%	85.7%	75.9%	77.4%	74.3%		
	12.6%	21.7%	18.5%	14.9%	16.3%	27.7%	11.2%	16.6%	16.8%	21.7%	23.1%	35.7%	17.6%	17.2%	14.3%	24.1%	22.6%	25.7%		

Figure 50 – Confusion matrix k-fold2 – 500x200



1	381	1	17	2	1		2	2	1	2	5	1	1	1	2	1	1	13	87.8%	12.2%
2	2	352	3	2	12	12		6	1	17	7	5	5	2	1	22	4	4	77.0%	23.0%
3	7	4	389	8	6	2	7	7	7	3	2	7	5		3	2	4	14	81.6%	18.4%
4	4		8	417	2		3	8	18	2	1		1	14				3	86.7%	13.3%
5	3	15	8		385	4		1	1	2	1	3	5	3	1	8	15	10	82.8%	17.2%
6	2	6		3	2	375	3	1	11	11	5	16	7	14	6	22	12		75.6%	24.4%
7	4		2	1	1	3	385	8	13	1	4	1	4	1	3	1	3	2	88.1%	11.9%
8	4	3	6	5	1	3	5	404	4	4	6	6		3	9	5	1	2	85.8%	14.2%
9	1	1	5	24	1	3	7	3	418	2	3	1	1	4	3	2	3	1	86.5%	13.5%
10	4	23	3	2		5	1	8	3	372	4	3	5	11		6		1	82.5%	17.5%
11	6	9	8	1	1	8		9	1	9	346	11	5	2	7	12	1	23	75.4%	24.6%
12		4	2	1	3	21		4	3	3	16	296	6	5	25	27	44	24	61.2%	38.8%
13	6	8	4	3	6	6	11		1	7	8	9	375	23	5	1	5	3	78.0%	22.0%
14		5	2	12	1	18	2	6	7	16	4	4	14	382	2	5	5	4	78.1%	21.9%
15	1	1	2		6	6	4	9	1	1	11	19	2	1	400	7	18	2	81.5%	18.5%
16		18		2	6	23	1	1		4	15	19	2	3	2	361	13	5	76.0%	24.0%
17	1	1	1	1	9	7	1	2	1	1	3	29	6	1	9	12	394	3	81.7%	18.3%
18	7	3	14	8	16	4	2	4	1	3	28	13	4		5	4	5	321	72.6%	27.4%
	88.0%	77.5%	82.1%	84.8%	83.9%	75.0%	88.7%	83.6%	85.0%	80.9%	73.8%	66.8%	83.7%	81.3%	82.8%	72.5%	74.6%	73.8%		
	12.0%	22.5%	17.9%	15.2%	16.1%	25.0%	11.3%	16.4%	15.0%	19.1%	26.2%	33.2%	16.3%	18.7%	17.2%	27.5%	25.4%	26.2%		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		

Figure 51 – Confusion matrix k-fold3 – 500x200

The *500x200* network, follows the same trends and patterns as the *600x450*, with slightly lower accuracy, but the difference is almost insignificant.

6. Conclusions

This master thesis focuses on the development of a speech command recognition model based on artificial neural networks and MFCCs coefficients. The main part of the thesis was focused on finding the right method for signal pre-processing and vocal feature extraction. The second part was the training of an ANN model for 18 command classification. The commands were chosen for an industrial robot application, as a future goal would be to implement this model in the laboratory to communicate with the robot. The signal processing approach is the filtering of the signal to remove the main noise quantity and balance the frequency spectrum, the segmentation of the signal in smaller frames, which can be considered stationary with time, and the extraction of the Mel-Frequency Cepstral Coefficients, by applying the Discrete Fourier Transform, converting to Mel-scale and back in time domain with the Inverse Discrete Fourier. For the machine learning process, the object is an artificial neural network for pattern recognition. The model's architecture was the main part of investigation, to be able to learn from the dataset and predict with accuracy 80% the unknown commands.

This project is a very good starting point to involve with AI development and speech processing. The methodology described and used covers all the fundamentals of speech recognition. Is both challenging and interesting to understand and analyse the hearing process and perception of sounds, find the equivalent mathematic parameters and function and create a model to mimic that from scratch. The key findings are summarized in the following list

- Human speech contains a lot of information regarding the identity of the speaker, their sentiment situation and of course the words pronounced. Humans are able to understand and process all this information at once, when models are focusing on one task at a time. The right pre-processing of the signal and the calculation of the mathematic portions that contain the important information are key steps for speech recognition.
- The splitting of the signal into smaller frames, allows to handle each part as time stationary. The segmentation gives the necessary focus to smaller details of the signal (especially time dependent ones) that they would be lost is the processing was on the whole signal. When separating the signal, is very beneficial to use overlapping frames to minimize the information lost, especially on the frame's edges. Windowing functions are vital for the frames to behave properly.
- Regarding classification methods, there are many possible options, but whichever used must be optimized for the specific application. When using a feed-forward neural network, its architecture is the number one key factor for investigation. When using multiple layers. The first hidden layer is the most important and defines the performance boundaries of the system, where the other layers are finetuning the results.
- Using simple and small size structures leads to light models, but under-educated to the dataset, especially in this type of complex problems. Using big, complex structures, gives more flexibility to the model and broadens its accuracy. When pushing to the limit and using a model too big for the case study, the model tends to overfit on training data and loses the ability of generalization.
- The quality of the dataset, or how representative and rich is, can be a real bottle neck for achieving high performance. From literature review the rule-of-thumb is that there should be at least 1000 recording for complex problems and from thousands of speakers. The existing dataset contains 2350 recordings per commands from thousand speakers, with different voices, pronunciations and accents. So, it is considered representative.
- The accuracy on the test subset of Google's Speech Commands Dataset is around 80% and accepted according to the requirements, but it's expected to be less for commands

from random speakers. It is recommended to try the dataset with many different users, starting point can be 10 and then expand till 100, and check if the accuracy remains in higher levels, or drops significantly.

7. Future Steps

7.1. Improvements

This study is the first approach to develop a machine leaning model for speech recognition. The findings and results from this study are good and promising, but there are many areas to improve and finetune. Regarding the dataset, the vocabulary that the network can recognize is not enough for manipulation of a robotic arm. It was not possible to find an existing dataset with all the necessary commands and on the other hand, the creation of a custom dataset would be very time consuming and required a big amount and variety of speakers. One recommended improvement would be the generation of a larger dataset with all the needed commands, most of them were mentioned above. For expanding the dataset, the same trained model can be used.

As mentioned before, it is recommended to test the ANN, not only on Google's Speech Commands Dataset, but with different users, to ensure that the accuracy remains stable. For the results to be valid, there should be tried many users, with variety in voice, accent, pronunciation etc. The users should be more than 10 and if possible, reach 100 or more.

Regarding the classification network, only artificial neural networks with two layers were investigated. It would be interesting to see more complicated networks, of three or more hidden layers, especially if the classes increase to recognize more words. Also, it would be insightful to compare one-to-one the optimum artificial network with other classification models, like Naïve Bayes (NB), Random Forest (RF) and Nearest Neighbour (k-NN) models or convolutional neural networks (CNN). A different approach would be to use convolutional neural networks (CNN) for image recognition and instead of computing the MFCCs the model would classify the commands based on the visualization of the MFCCs or the frequency spectrum.

7.2. Additional Studies

The objective of the projects is to build a machine learning model for voice command recognition, to be coupled with an industrial robot arm. In the current study, the only consideration of the robot, was for defining the vocabulary, since the development of the computer-robot interface was not included in the thesis scope. It would be very interesting to check the recognition object "in action" and couple it with the robot interface. All the network's classes should be connected with different robot's actions, so the recognition for each command leads to a robot action. For this to be done there are two things needed. First one is the correlation between the written word (predicted class) and the V^+ command. For example, the prediction "*approach*" should direct to the command "*MOVE*", or even better the sequence of the predictions: "*MOVE*", "*five*", "*zero*", "*slash*", "*two*", "*zero*", "*slash*", "*two*", "*zero*", should indicate the V^+ command: "*MOVE (50, 20, 30)".* V^+ is the language to program Staubli RX 90L and is used in many industries, including system programming, web development, and game development. The main advantage is its compilation speed, which makes it as fast as C and suitable for real time applications. The interface code will transform the abstract single-word commands into meaningful V^+ code commands. So, the one recommended addition is the development of the interface and V^+ codes.

8. Bibliography

[1] B. Copeland, "Artificial Intelligence definition," Encyclopaedia Britannica, 2024. [Online].

[2] J. Holdsworth and M. Scapicchio, "Deep learning vs. machine learning," IBM American multinational technology corporation, 2017. [Online]. Available: https://www.ibm.com/topics/deep-learning.

[3] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development,* p. 21, 1959.

[4] R. Karjian, "History and evolution of maching learning: A timeline," TechTarget, 13 June 2024. [Online]. Available: https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History.

[5] W. Pitts and W. McCulloch, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology,* p. 17, 1943.

[6] D. Hebb, The Organization of Behavior: A neuropsychological Theory, 1949.

[7] A. Turing, Computing Machinery and Intelligence, Mind, 1950.

[8] Y. LeCun , Y. Bengio and P. Haffner, Backpropagation Applied to Handwritten Zip Code Recognition, MIT Press, 1989.

[9] S. Albahli, F. Alhassan, W. Albattah and R. U. Khan, Handwritten Digit Recognition: Hyperparameters-Based Analysis, MDPI Applied Science, 2020.

[10] M. Pinola, Speech Recognition Through the Decades: How we ended up with Siri, 2011.

[11] D. Spicer, "AUDREY, Alexa and more: A history of automatic speech recognition," 2021. [Online]. Available: https://computerhistory.org/blog/audrey-alexa-hal-and-more/.

[12] H. Kumari, J. Biji and K. A. Navas, "A Novel Objective Audio Quality Measure," *10th National Conference on Technological Trends,* 2009.

[13] UNIVERSAL ROBOTS, "Best Applications of Robotic Arms," 2022.

[14] UNIVERSAL ROBOTS, "Types of Robotic Arms," 2022.

[15] E. M. Rosales and Q. Gan, "Forward and Inverses Kinematics Models for a 5-dof Pioneer 2 Robot Arm," University of Essex - Department of Computer Science, 2002.

[16] Staubli, Arm - RX series 90B family, 2008.

[17] P. Makrylakis, "Industrial robot programming through voice commands," National Technical University of Athens, Athens, 2023.

[18] A. Techhnology, "V+ Language Reference Guide," 1997.

[19] A. Technology, "V+ Language User's Guide, Ver. 12.1," 1997.

[20] B. Automation, "15 Robot End Effector Types and Selection Criteria," 2022. [Online]. Available: https://www.b2eautomation.com/insights/15-robot-end-effector-types-and-selection-criteria.

[21] A. T. Ashraf, A. S. Hasanen and F. N. Mohammad, "Voice recognition system using machine learning techniques," *Elsevier,* April 2021.

[22] A. N. S. S. M.M. Hasan, "An approach to voice conversion using feature statistical mapping," *Elsevier,* p. 21, May 2005.

[23] D. Eringis and G. Tamulevičius, "Improving Speech Recognition Rate through Analysis Parameters," *De Gruyter,* 2014.

[24] S. K. Kumar, B. Yazdanpanah and D. G. S. N. Raju, "Performance Comparison of Windowing Techniques for ECG Signal Enhancement," *International Journal of Engineering Research,* p. 4, December 2014.

[25] M. Puckette, "Taxonomy of filters," in *Theory and Techniques of Electronic Music*, University of California, San Diego, World Scientific, 2003.

[26] V. Tiwari, "MFCC and its applications in speaker recognition," *International Journal on Emerging Technologies,* p. 4, February 2010.

[27] G.-C. Vosniakos and P. Benardos, "Artificial Neural Networks in Manufacturing Systems," National Technical University of Athens.

[28] P. Warden, "Speech Commands: A public dataset for single-word speech recognition - Copyright Google 2017," [Online]. Available: http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz. [Accessed 2017].

[29] V+ Language Users Giude Version 12.1, USA, 1997.

[30] R. M. V. V. L. Svitlana Maksymova, "Software for Voice Control Robot: Example of Implementation," *Open Access Library Journal,* p. 12, 2017.

[31] L. Muda, M. Begam and I. Elamvazuthi, "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques," p. 6, March 2010.

[32] S. Khawatreh, B. Ayyoub, A. Abu-Ein and Z. Alqadi, "A Novel Methodology to Extract Voice Signal Features," *International Journal of Computer Applications,* vol. 179, p. 4, 2018.

[33] H. Hofling, T. Berglund and A. Vaara, "Audio Compression," Uppsala University, Uppsala, 2002.

[34] J. P. Egan and H. W. Hake, "On the masking pattern of a simple auditory stimulus," *The Journal of the Acoustical Society of America,* pp. 622-630, 1950.

[35] J. V. Tobias, "Low-frequency masking patterns," *The Journal of the Acoustical Society of America,* pp. 571-575, 1977.

[36] B. Y. D. G. S. N. R. K.Sravan Kumar, "Performance Comparison of Windowing Techniques for ECG Signal Enhancement," *International Journal of Engineering Research,* p. 4, December 2014.

[37] W. L. Hosch, "Machine Learning definition," Encyclopaedia Britannica, 2024. [Online].

[38] A. Bryson and Y.-C. Ho, Applied optimal control, Hemisphere Pub. Corp., 1975.

9. Appendix - Scripts

9.1. Python Script – Signal Processing

Read input signal:

```
sample_rate, signal = scipy.io.wavfile.read(filepath)
time = len(signal)/sample_rate
dt=np.arange(0,time,1/sample_rate)
```

Pre-processing:

Pre-emphasis

```
pre_emphasis = 0.97
emphasized_signal = np.append(signal[0], signal[1:] - pre_emphasis * signal[:-1])
```

Framing

```
frame_size = 0.025
frame_stride = 0.01

frame_length, frame_step = frame_size * sample_rate, frame_stride * sample_rate  # Convert from seconds to samples
signal_length = len(signal)
frame_length = int(round(frame_length))
frame_step = int(round(frame_step))
num_frames = int(np.ceil(float(np.abs(signal_length - frame_length)) / frame_step))  # Make sure that we have at least 1 frame
pad_signal_length = num_frames * frame_step + frame_length
z = np.zeros((pad_signal_length - signal_length))
pad_signal = np.append(signal, z) # Pad Signal to make sure that all frames have equal number of samples without truncating any samples from the original signal
indices = np.tile(np.arange(0, frame_length), (num_frames,1)) + np.tile(np.arange(0, num_frames * frame_step, frame_step), (frame_length, 1)).T
time_indices = indices/sample_rate
frames = pad_signal[indices.astype(np.int32, copy=False)]
```

Hamming Window

```
frames_window = frames.copy()
frames_window *= np.hamming(frame_length) #Hamming window
```

Fast Fourier Transform

```
NFFT = 512
mag_frames = np.absolute(np.fft.rfft(frames_window, NFFT))  # Magnitude of the FFT
rows, cols = mag_frames.shape
#print(f"Magnitude Frames \nNumber of frames: {rows} \nNumber of FFt points: {cols}")

pow_frames = ((1.0 / NFFT) * ((mag_frames) ** 2)) # Power Spectrum
rows, cols = pow_frames.shape
#print(f"Power Frames \nNumber of frames: {rows} \nNumber of FFt points: {cols}")
frequency = np.linspace(0, sample_rate/2, len(pow_frames.T))
```

Frequency Spectrum

```
col = 5
row = math.ceil(num_frames/col)
Columns = np.arange(0,col)
Rows = np.arange(0,row)
Size = np.arange(0,num_frames)
```

```
# Create grid
Column = np.tile(Columns, row)
Column = Column[:num_frames]
Row = np.array([])
for r in Rows:
    Row = np.concatenate((Row, np.tile(int(r), col)[:num_frames]))
Row = Row.astype(int)
Row = Row[:num_frames]
Grid = pd.DataFrame({"Row":Row, "Column":Column})
```

Feature Extraction:

Mel-scale

```
nfilt = 40
low_freq_mel = 0
high_freq_mel = (2595 * np.log10(1 + (sample_rate / 2) / 700))  # Convert Hz to Mel
mel_points = np.linspace(low_freq_mel, high_freq_mel, nfilt + 2)  # Equally spaced in Mel scale
hz_points = (700 * (10**(mel_points / 2595) - 1))  # Convert Mel to Hz
bin = np.floor((NFFT + 1) * hz_points / sample_rate)

fbank = np.zeros((nfilt, int(np.floor(NFFT / 2 + 1))))
for m in range(1, nfilt + 1):
    f_m_minus = int(bin[m - 1])   # left
    f_m = int(bin[m])             # center
    f_m_plus = int(bin[m + 1])    # right

    for k in range(f_m_minus, f_m):
        fbank[m - 1, k] = (k - bin[m - 1]) / (bin[m] - bin[m - 1])
    for k in range(f_m, f_m_plus):
        fbank[m - 1, k] = (bin[m + 1] - k) / (bin[m + 1] - bin[m])
filter_banks = np.dot(pow_frames, fbank.T)
filter_banks = np.where(filter_banks == 0, np.finfo(float).eps, filter_banks)  # Numerical Stability
filter_banks = 20 * np.log10(filter_banks)  # dB
```

MFCCs

```
num_ceps = 12
mfcc = dct(filter_banks, type=2, axis=1, norm='ortho')[:, 1 : (num_ceps + 1)] # Keep 2-13
(nframes, ncoeff) = mfcc.shape
n = np.arange(ncoeff)
```

Create Excel Files with MFCCa values:

```
excel_file_path = os.path.join(excel_path, f"{filename}.xlsx")
# Create a new Excel workbook and add a worksheet
workbook = openpyxl.Workbook()
worksheet = workbook.active
# Write the NumPy array to the worksheet
for row in mfcc:
    worksheet.append(list(row))
workbook.save(excel_file_path) # Save the workbook to an Excel file
```

9.2. MATLAB Script – ANN Training

```
%% This scrip uses brute force to optimize the architecture of an Artificial Neural Network
clear all
clc

%% Load dataset
load Commands.mat\dataset.mat
uniqueValues = unique(dataset(:,1));
numRows = size(dataset,1);
numCols = length(uniqueValues);
randomOrder = randperm(numRows);
datasetNew = dataset(randomOrder, :);

dataIn = datasetNew(:,2:end)';
dataOut_ = datasetNew(:,1)';                          % 1 x #Recordings: array that contains the
reference number of each recording
dataOut = zeros(numCols,numRows);                         % #Commands x #Recordings: table
that contains 0-1
for i = 1:numRows
    dataOut(dataOut_(i)+1,i) = 1;
end
%% Cross-Validation Sheme - Train Validation and Test
k = 5;
c = cvpartition(numRows,"KFold",k);
for i = 1:k
    trainSet(:,i) = training(c,i);
    testSet(:,i) = test(c,i);
end
%% Labels
classLabels = ["zero" "one" "two" "three" "four" "five" "six" "seven" "eight" "nine" "down" "up" "left"
"right" "stop" "on" "off" "go"];
categories = categorical(classLabels);
categories = reordercats(categories, classLabels);
%%
hiddenSizes = [500 200];
trainFcn = 'trainscg';
net = patternnet(hiddenSizes,trainFcn);                    % Create a ML model
net.divideParam.trainRatio = 1;                          % Set data for training subset
net.divideParam.valRatio = 0.00;                          % Set data for validation subset
net.divideParam.testRatio = 0.00;                         % Set data for testing subset
net.layers{3}.transferFcn = 'softmax';

for i = 1:k
    trainSet_in{i} = dataIn(:,trainSet(:,i));
    trainSet_out{i} = dataOut(:,trainSet(:,i));                 % To use to train the network
    trainSetout{i} = dataOut_(:,trainSet(:,i));               % To use for confusion matrix
    testSet_in{i} = dataIn(:,testSet(:,i));
    testSet_out{i} = dataOut_(:,testSet(:,i));

    trained_net = train(net,trainSet_in{i},trainSet_out{i});
    temp_train = sim(trained_net,trainSet_in{i});
    temp_test = sim(trained_net,testSet_in{i});
    [~, est_train{i}] = max(temp_train);
    [~, est_test{i}] = max(temp_test);

    est_train{i} = est_train{i} - ones(size(est_train{i},1),size(est_train{i},2));
    est_test{i} = est_test{i} - ones(size(est_test{i},1),size(est_test{i},2));

    a_train(i) = sum(trainSetout{i} == est_train{i})/size(trainSetout{i},2);
    a_test(i) = sum(testSet_out{i} == est_test{i})/size(testSet_out{i},2);

    for j = 1:length(uniqueValues)
        s_train(i,j) = sum(trainSetout{i} == uniqueValues(j));
        s_test(i,j) = sum(testSet_out{i} == uniqueValues(j));
    end
    %
    % C_train{i} = confusionmat(trainSetout{i},est_train{i});
    % C_test{i} = confusionmat(testSet_out{i},est_test{i});
    % figure('Name',['TEST Confusion Matrix k = ', int2str(i)])
    % A = confusionchart(C_test{i});
    % A.RowSummary = 'row-normalized';
    % A.ColumnSummary = 'column-normalized';
    % figure('Name',['TRAIN Confusion Matrix k = ', int2str(i)])
    % A = confusionchart(C_train{i});
    % A.RowSummary = 'row-normalized';
    % A.ColumnSummary = 'column-normalized';

end

%% Labels
classLabels = ["zero" "one" "two" "three" "four" "five" "six" "seven" "eight" "nine" "down" "up" "left"
"right" "stop" "on" "off" "go"];
categories = categorical(classLabels);
categories = reordercats(categories, classLabels);

%% Confusion Matrix: TRAIN vs. TEST
for i = 1:k
    a_train(i) = sum(trainSetout{i} == est_train{i})/size(trainSetout{i},2);
    a_test(i) = sum(testSet_out{i} == est_test{i})/size(testSet_out{i},2);
    C_train{i} = confusionmat(trainSetout{i},est_train{i});
    C_test{i} = confusionmat(testSet_out{i},est_test{i});

    % Plot and save test confusion matrix
    fig_test = figure('Name',['TEST Confusion Matrix k = ', int2str(i)]);
    set(fig_test, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]); % Make figure full screen
    A = confusionchart(C_test{i});
    A.RowSummary = 'row-normalized';
    A.ColumnSummary = 'column-normalized';
    saveas(fig_test, ['Test_Confusion_Matrix_k_', int2str(i), '.png']);

    % Plot and save train confusion matrix
    fig_train = figure('Name',['TRAIN Confusion Matrix k = ', int2str(i)]);
    set(fig_train, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]); % Make figure full screen
    A = confusionchart(C_train{i});
    A.RowSummary = 'row-normalized';
    A.ColumnSummary = 'column-normalized';
    saveas(fig_train, ['Train_Confusion_Matrix_k_', int2str(i), '.png']);
end
```

9.3. MATLAB Scripts – Real time classification

```
%% Record audio file
% file_name = 'unknown';
% Specify the audio recording parameters
sample_rate = 16000; % Sample rate in Hz
duration = 1; % Recording duration in seconds

% Create an audiorecorder object
recorder = audiorecorder(sample_rate, 16, 1); % 16-bit, 1 channel (mono)

disp('Start speaking.');
recordblocking(recorder, duration);
disp('End of recording.');

randomInt = randi([1, 8000]);
% Get the recorded audio data
signal = getaudiodata(recorder);
% Save the recorded audio to a file
audiowrite([folderPath2save, '\unknown (', num2str(randomInt), '.wav'], signal, sample_rate);
signal = signal * (2^15);

time = length(signal) / sample_rate;
dt = 0:1/sample_rate:(time - 1/sample_rate);

%% Pre-emphasis
pre_emphasis = 0.97;
emphasized_signal = [signal(1); signal(2:end) - pre_emphasis * signal(1:end-1)];

%% Frame parameters
frame_size = 0.025;
frame_stride = 0.01;
frame_length = round(frame_size * sample_rate);
frame_step = round(frame_stride * sample_rate);
signal_length = length(emphasized_signal);
num_frames = ceil(abs(signal_length - frame_length) / frame_step);
pad_signal_length = num_frames * frame_step + frame_length;
z = zeros(pad_signal_length - signal_length, 1);
pad_signal = [emphasized_signal; z];

%% Generate frames
indices = 1 + repmat(0:frame_length-1, num_frames, 1) +
repmat(0:frame_step:num_frames*frame_step-1, frame_length, 1)';
frames = pad_signal(indices);

%% Apply Hamming window
frames = frames .* hamming(frame_length)';

%% FFT and Power Spectrum
NFFT = 512; % Set your desired NFFT value
mag_frames = abs(fft(frames, NFFT, 2));
mag_frames = mag_frames(:,1:NFFT/2+1);
pow_frames = (1.0 / NFFT) * (mag_frames.^2);
pow_frames_new = 10 * log10(pow_frames');
frequency = linspace(0, sample_rate/2, size(pow_frames, 2));


%% Mel filter bank
nfilt = 40;
low_freq_mel = 0;
```

```
high_freq_mel = 2595 * log10(1 + (sample_rate / 2) / 700); % Convert Hz to Mel
mel_points = linspace(low_freq_mel, high_freq_mel, nfilt + 2); % Equally spaced in Mel scale
hz_points = 700 * (10.^(mel_points / 2595) - 1); % Convert Mel to Hz
bin = floor((NFFT + 1) * hz_points / sample_rate);

fbank = zeros(nfilt, floor(NFFT / 2) + 1);
for m = 2:nfilt+1
   f_m_minus = bin(m-1);   % left
   f_m = bin(m);           % center
   f_m_plus = bin(m+1);    % right

   for k = f_m_minus:f_m-1
      fbank(m-1, k+1) = (k - bin(m-1)) / (bin(m) - bin(m-1));
   end
   for k = f_m:f_m_plus-1
      fbank(m-1, k+1) = (bin(m+1) - k) / (bin(m+1) - bin(m));
   end
end

% Compute filter banks
filter_banks = pow_frames * fbank.';
filter_banks = max(filter_banks, eps); % Numerical Stability
filter_banks = 20 * log10(filter_banks); % dB

%% MFCC computation
num_ceps = 12; % Set your desired number of MFCC coefficients
mfcc = dct(filter_banks')';
mfcc = mfcc(:, 2:num_ceps+1);

%% Save to excel
excelFilename = [folderPath2save, '\unknown (', num2str(randomInt), '.xlsx'];
writematrix(mfcc, excelFilename)

%% Convert matrix to input format for ANN
data = reshape(mfcc.', 1, []);
data = data';

%% Classification
% load 'C:\Users\AnnaMaria\Documents\Industrial robot programming through voice
commands\Classification\ANN_1'
load 500x200.mat % Loads the pre-trained ANN

classLabels = ["zero" "one" "two" "three" "four" "five" "six" "seven" "eight" "nine" "down" "up" "left"
"right" "stop" "on" "off"];

guess = sim(trained_net,data); % Classification
[~, command] = max(guess);

Command = classLabels(command);

disp(Command)
```