SPEECH COMMANDS RECOGNITION FOR DRONE INSTRUCTIONS

Andrea Fumagalli – 46930A

Università degli studi di Milano - Audio pattern recognition project - AA 2024/2025

ABSTRACT

In this project I realised a system to recognise different speech commands acquired at real time in the mean of a convolutional neural network trained using log scale Mel spectrogram of the samples from the *Speech Commands* [1] dataset by Google as features.

1. INTRODUCTION

The project has the fictional purpose of realising a system able to recognise different commands given by voice to some robotic dispositive in order to control its movements in a 3D space. In particular, just to give the project some real-life connotations and requirements, I choose a drone. Nonetheless the structure and the methods described in this report could be applied, choosing the right set of commands, to a lot of different field. For example, to control a robot moving on the ground, or a car, or a character in a videogame or even some multi-purpose software.

This task is a simplification of the more general task of speech recognition in which we choose to recognise only a subset of words (called commands). The speech recognition task is a well explored and documented task for which a variety of solid and performing solution are already available. But I thought that for the purposes of an academic project it could be interesting to try to develop a new one from scratch.

In order to be able to do this I started from the *Speech Commands* dataset by Google, a well-known and documented dataset which present a complete enough set of commands for our case. As learning structure, I choose a convolutional neural network (CNN) which is a simple enough structure for me to be understood and set up but it is also reliable and able to learn quickly and with good performances the patterns we are interested in. CNN are in fact frequently use in this type of tasks as seen during the lessons. As of the features to feed to the network, after testing the performance of some other candidates (see section 2.2 for more information) I opted to use a log scale Mel spectrogram which seemed to be a good choice for our task.

In the end to test the results and to have some sort of real life feeling to them I implemented a quick and simple script that listen for real time audio from an input source and recognize, using the trained CNN, if any of the commands learned are been spoken.

2. METHODS

2.1 Dataset

The *Speech Commands* dataset is composed of 68.717 samples at 16 kHz divided between train, validation and test. It contains samples of 30 different commands and some background noise samples.

From the available commands, considering the goal of the project, I choose to use only 6 of them: “go”, “stop”, “left”, “right”, “up” and “down”. I believe that this subtest allows us to define properly enough the movement of a drone in a 3D space while also being contained and simple.

2.2 Features

When starting to work on the project the first features I tested were the first 13 Mel-frequency cepstral coefficients (MFCC), a standard largely used in speech processing, but their performances were poor (~20% accuracy rate). I then tried some expansions of those: the first 25 MFCC [2] and 13 MFCC + 13 deltaMFCC + 13 deltaDelta MFCC but neither of them, while obtaining an accuracy rate of around 90% on the training dataset, was obtaining more than 50% accuracy rate on the validation dataset. I then decided to try to use the log scaled Mel spectrogram that was discussed during the lesson regarding deep learning for audio analysis and since it immediately gave some nice results I picked this as the feature of choice for the task.

The Mel spectrogram is a decompressed version of MFCC (in fact MFCC are obtained from the spectrogram). A spectrogram is obtained by applying a Fast Fourier Transform on overlapping windowed fragment of the signal, the result is a 2D array with frequency on one dimension and time on the other and it represents how the amplitude of a signal varies over time at different frequencies. In particular, the spectrogram I used is in mel scale: a unit of pitch such that equal distances in pitch sounded equally distant to the listener.

In this project the features are extracted by resizing all the samples to 1 second each, the mel spectrogram of the sample is then calculated and in the end we take the logarithm of the spectrogram. This way we obtain a log-scale spectrogram which is very similar to the human perception.

Figure 1: Feature extraction method

2.3 Convolutional Neural Network

The architecture of the network employed for the task is very simple and standard and it has not been questioned much during the development, in part because it wasn't the focus of the work in part because it would have been a very complicated and log work to evaluate and test accurately all the possible solutions.

The network is composed of:

- 1 input layer.
- 3 convolutional layer (composed of a convolution, a batch normalization and a ReLu layer) each followed by a max pooling module.
- 2 other convolutional layers (structured as before).
- A final max polling layer that pools globally over time.
- A dropout module
- A fully connected layer
- A softmax layer

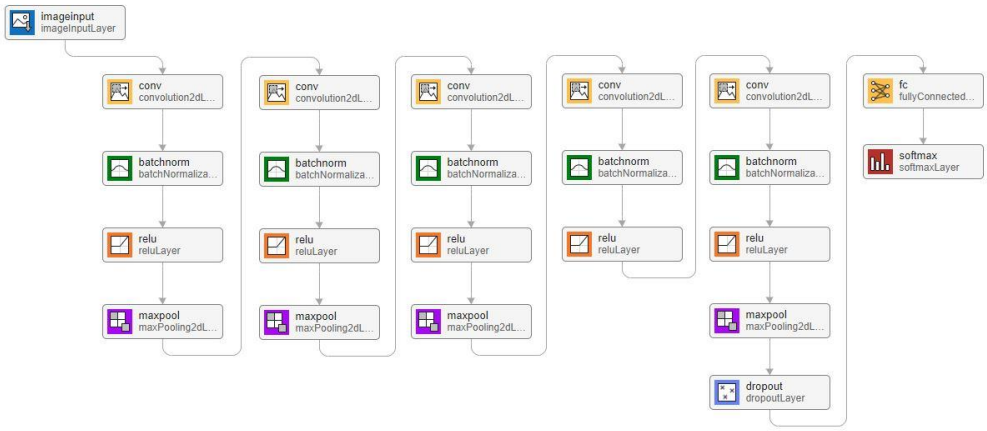

Figure 2: CNN architecture

3. IMPLEMENTATION

All of the project has been implemented in MATLAB 2024b using some add-on toolboxes, mainly the *Audio Toolbox* and the *Deep Learning Toolbox.*

After loading the dataset using the *buildDataset* script, the *trainNet* script defines two *audioDatastore* structures one for the training and one for the validation. Each one contains the samples of the selected commands, some background samples and some samples of the excluded commands labelled as "unknown". The two datasets obtained are composed by 22530 samples for the training set and by 3167 samples for the validation set.

The features of both the datastores are then extracted using an *audioFeatureExtractor* configured to extract the mel spectrogram using windows of 0.025 seconds overlapping for 0.015 seconds. The samples are processed according to the methods defined in 2.2 and the results are the spectrograms of each sample organized in 32 bins.

A network structured as defined in section 2.3 is then built and trained using the function *trainnet* for 10 epochs with a learning rate of 0.0003 using cross entropy.

The output of the function is a trained net which after being saved is used in the *realTimeTest* script. This script listens to the audio coming from an input device using an *audioDeviceReader* and an *AsyncBuffer*, then each sample collected is processed to extract its spectrogram which is given as input to a predictor that uses the trained net to output one of the possible classes. If the prediction is not output the script displays it.

4. RESULTS

The trained network has an accuracy of 94.5% on the training dataset and an accuracy of 92% on the validation dataset.

Testing with real time audio also shows pretty good results with the net being able to almost always identify and distinguish the commands it has been trained for specially if they are pronounced clearly. The accuracy falls off a bit if we use similar words to the commands (for example "no" is classified as "go" or "laugh" is classified as "left") or if the pronounce is unclear, but in a real life application this cases shouldn't be very frequent since we expect the device to only receive appropriate commands.

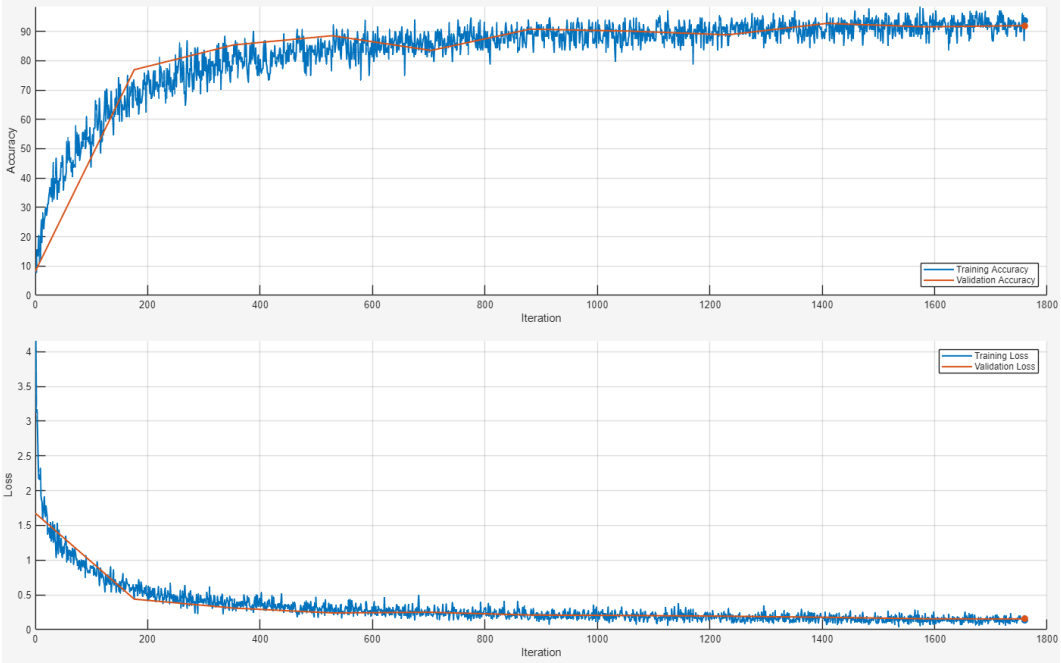

Figure 3: Accuracy and loss during training

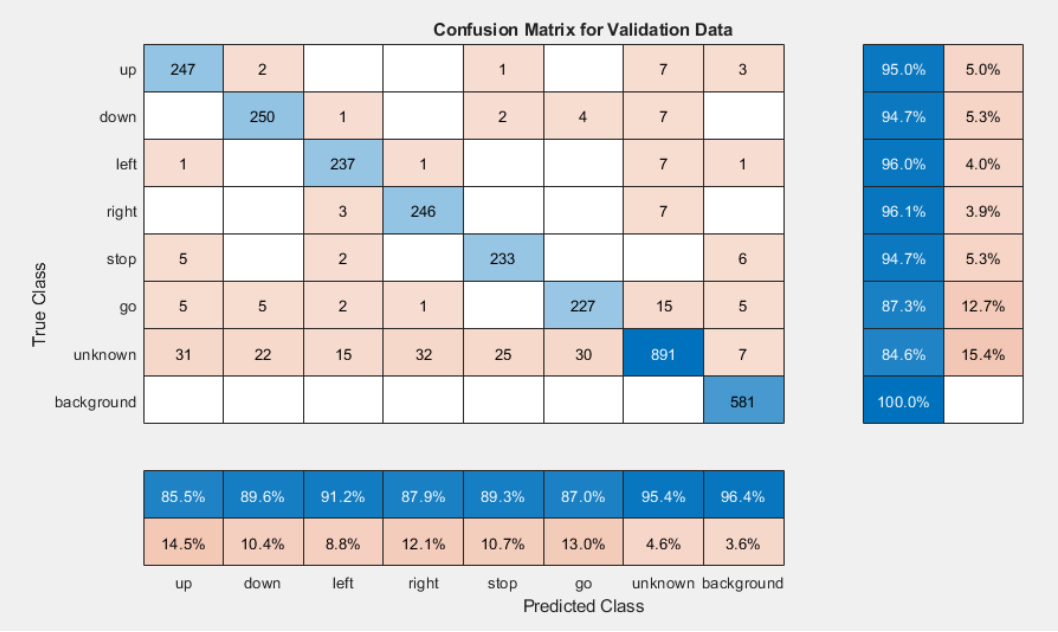

Figure 4: Confusion matrix for validation data

5.CONCLUSIONS

I have obtained a system able to recognise different commands at real time with a high rate of success. The process and techniques employed could be used to expand the set of commands or even used for some entirely new and different keywords. For example, a possible expansion of the set would be to include numbers in order to discretize the 360° space of rotation. This would allow us to specify more refined and accurate movements.

Working on the project I have increased my knowledge of the features useful for speech recognition, how they are obtained and what information do they carry. I also have had the possibility to build and train a neural network having hand-on experience in this field.

REFERENCES

[1] Pete Warden. Speech commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz, 2017.*

[2] Hasan, Md Rakibul & Hasan, Md & Hossain, Md. (2021). How many Mel-frequency cepstral coefficients to be utilized in speech recognition? A study with the Bengali language. The Journal of Engineering. 2021. 817-827. 10.1049/tje2.12082.