Machine Learning

Speech Commands Recognition

Yasamin Hosseinzadeh Sani

University of Pavia, Italy
Email: yasamin.hosseinzadehsa01@universitadipavia.it

May 23, 2024

Abstract

This initiative sought to create a highly precise speech recognition model that can identify 35 distinct spoken words, using a dataset of 105,829 audio clips from a variety of speakers. The model's success in detecting speech patterns across diverse conditions underscores the strength of the deep learning algorithms utilized and underscores the possibilities for further enhancements and practical applications.

1 Introduction

This project is about the implementation of a Neural Network for the classification of audio files. This problem is known as *Speech Command Recognition* and its importance has been rising in the last years due to the development of smart devices and the need for more natural interaction with them. The main library exploited for the implementation is *pvml*.

1.1 Available Data

The dataset comprises 105,829 recordings of 35 words spoken by a diverse group of speakers. For analytical purposes, it is segmented into three subsets: *train*, *validation*, and *test*.

1.2 Goal

The goal is to fit the data to a Multi Layer Perceptron (MLP) so that it can be exploited to classify new unseen data on the 35 different classes.

1.3 Multi Layer Perceptron

A Multi Layer Perceptron (MLP) is a form of Feed Forward Neural Network that includes one or more hidden layers of linked neurons situated between an input and an output layer. In an MLP, each neuron receives inputs from all neurons in the preceding layer, computes a weighted sum of these inputs, applies an activation function, and then sends the output to the next layer. The MLP fine-tunes its parameters (weights and biases) by minimizing a specified Loss Function (Cross Entropy) via a technique known as Backpropagation.

2 Model Building

Before beginning model construction, it's crucial to preprocess the data to ensure it's appropriately formatted for the training algorithm. This crucial step, known as *Feature Extraction*, lays the groundwork for successful model development.

2.1 Features Extraction

Beginning with the raw audio data, feature extraction involves generating a spectrogram for each word. To standardize these spectrograms, the original signals are either padded or truncated to uniform dimensions.

2.2 Data Visualization

Visualizing the spectrograms can enhance understanding of the data. Figure 1 showcases spectrograms for various words, highlighting areas with higher frequency presence in red. It's crucial to note that spectrograms for the same word can vary based on the speaker.

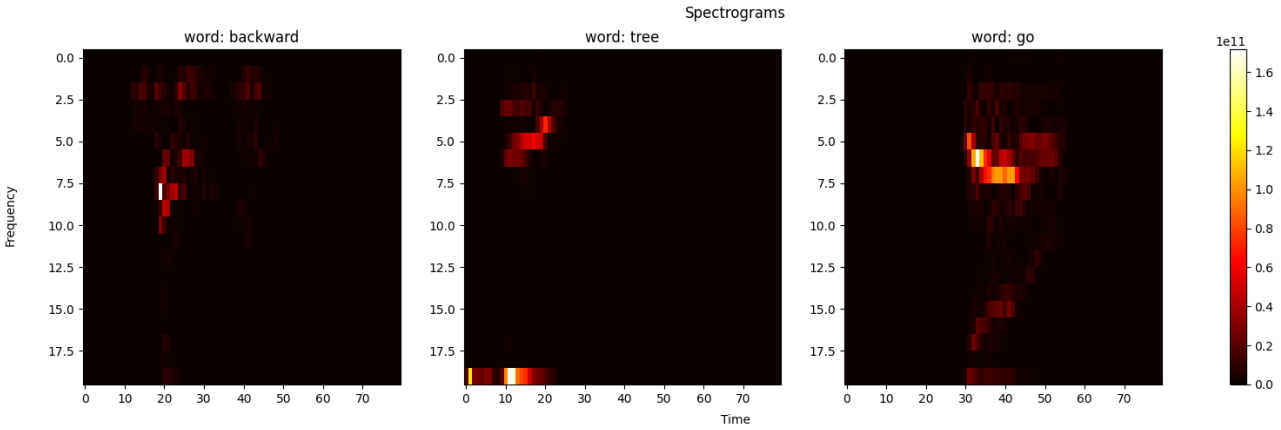

Figure 1: Spectrograms example

2.3 Feature Normalization

Normalizing the features is a critical step in the preprocessing phase. This project employs four normalization techniques: *Mean-Var*, *Min-Max*, *Whitening*, and *Max-Abs*. These methods are evaluated and compared in Figure 2. Among them, Mean-Variance normalization proves to be the most effective.

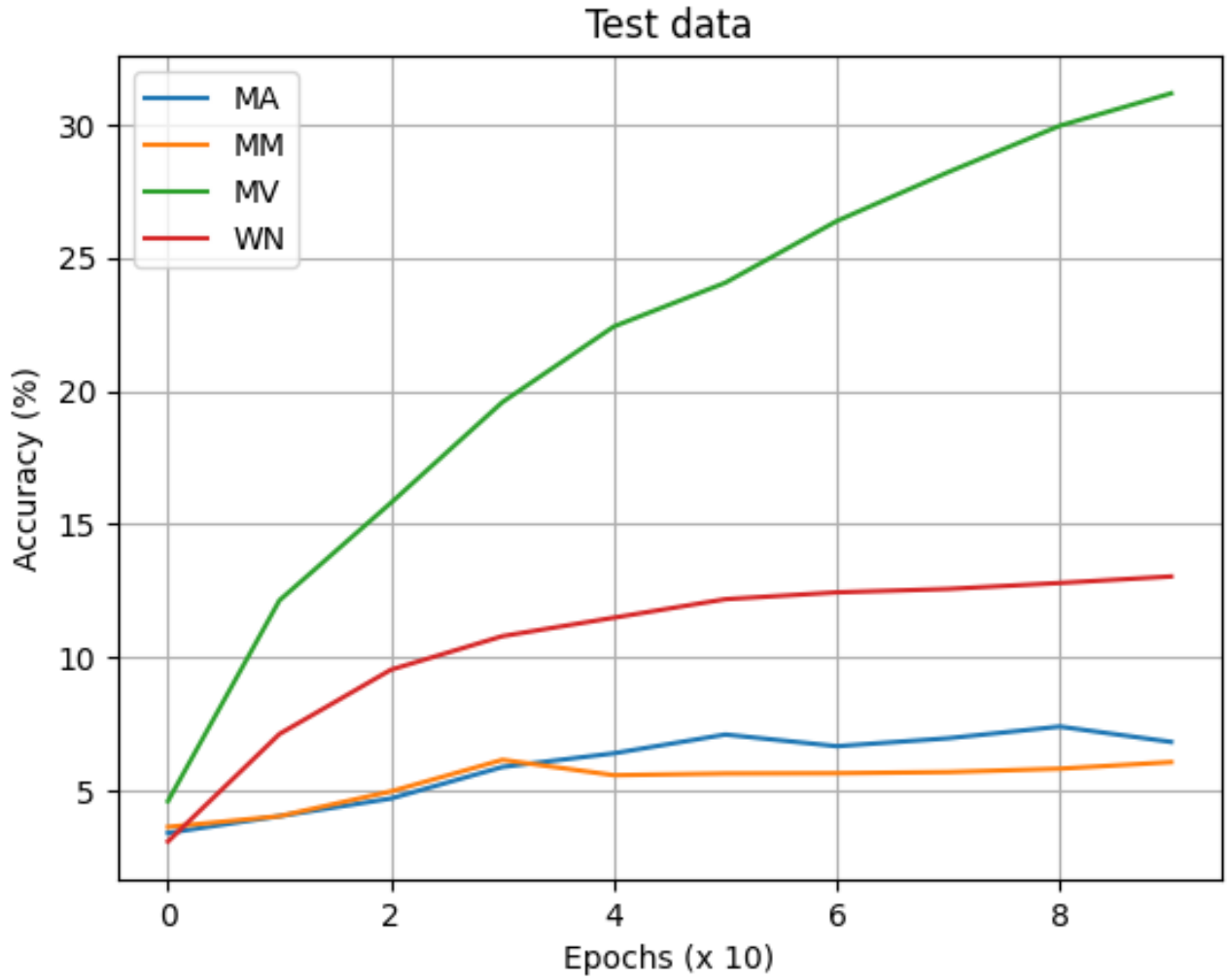

Figure 2: Normalization comparison

2.4 Train Classifier

Batch size

To enhance training efficiency, the data are segmented into mini-batches. Batch size significantly affects model performance, prompting tests with various sizes, the results of which are displayed in Figure 3. The model tested lacked hidden layers, leading to uniformly low performance across batch sizes, each achieving around 18 percent accuracy on the test set but differing in the number of epochs required. The batch size of 50 proved to be the most expedient and was selected as optimal.

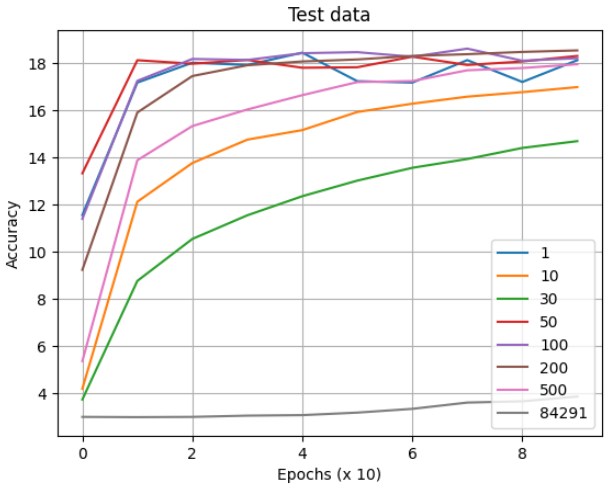

Figure 3: Batch comparison

Architecture

The network's architecture has a profound impact on its performance, leading to the comparison of various configurations as shown in Figure 4. Due to time constraints, the number of training epochs was capped at 100, even though some architectures with more parameters might require more time to fully realize their capabilities.

Using a more aggressive strategy, the architecture featuring multiple hidden layers was identified as the most effective. This configuration was subjected to an extended training period of 400 epochs, and the outcomes are depicted in Figure 5. The blue and orange models achieved 80 percent training accuracy, yet their performance on an independent test set was suboptimal, indicating potential overfitting to the training data.

To mitigate this issue, various regularization strategies were explored, and a regularization parameter of $\lambda = 0.001$ proved to be the most beneficial. Although this measure slightly alleviated overfitting, it did not eliminate it completely. Ultimately, the red model, which exhibited the best overall performance, was chosen for detailed analysis.

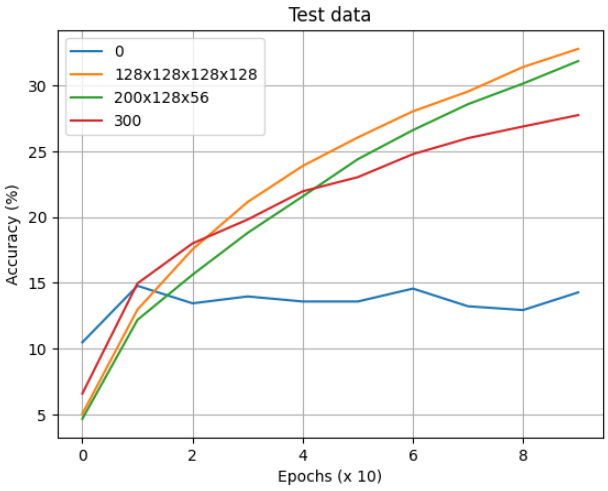

Figure 4: Architecture comparison

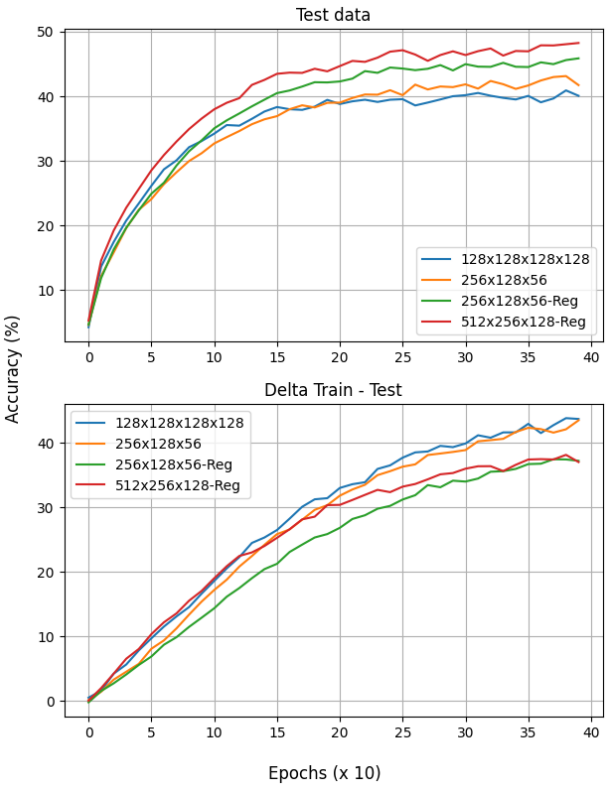

Figure 5: Deep Networks comparison

3 Model Analysis

To evaluate the performance of the selected model, a confusion matrix was generated, as shown in Figure 12. This matrix illustrates the counts of correct and incorrect predictions for each class, with darker colors indicating a higher frequency of samples from class i (row index) being classified as class j (column index). As anticipated, the matrix's diagonal is the most densely populated area, indicating frequent correct classifications by the model.

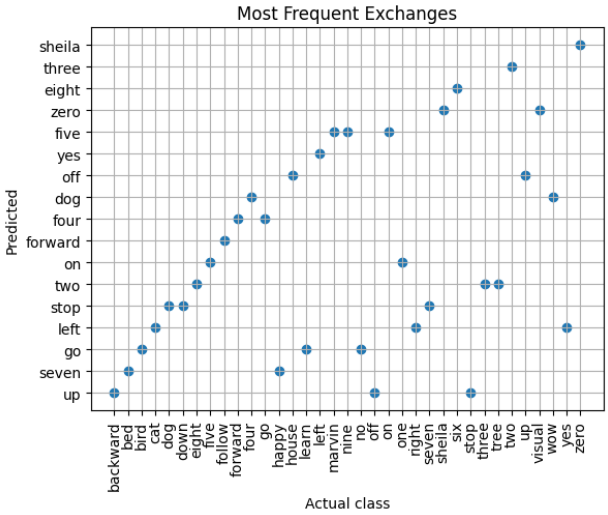

Figure 6: Most exchanged

Most Exchanged Classes
The most exchanged classes are those in which the model has more difficulties to distinguish between. In figure 6 are reported for each word the class to which the model classifies them (excluded correct classifications) most of times. Lots of exchanges are reasonable like *Forward* classified to *Four* and *Three* to *Two* whose spectrograms are very similar (figure 7).

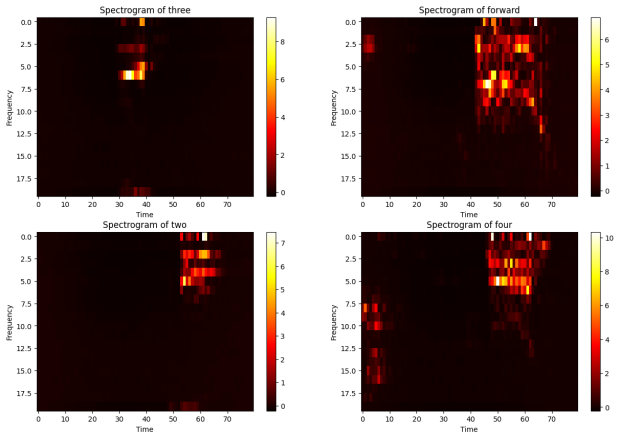

Figure 7: Three VS Two, Forward VS Four

Most Misclassified Samples

Certain words prove more challenging to classify than others, as depicted in Figure 8, which highlights the samples most frequently misclassified. Notably, *Tree* and *Learn* are among the most misclassified words; for instance, the audio files for *Learn* often contain mispronunciations that could even confuse human listeners. The underlying cause of these frequent misclassifications is illustrated in Figure 9. It becomes clear that words with fewer samples tend to be misclassified more often because the model has less data from which to learn, significantly impacting its overall effectiveness.

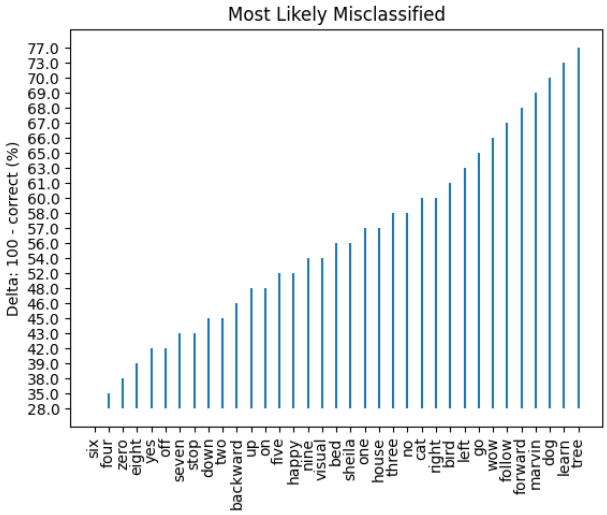

Figure 8: Most misclassified

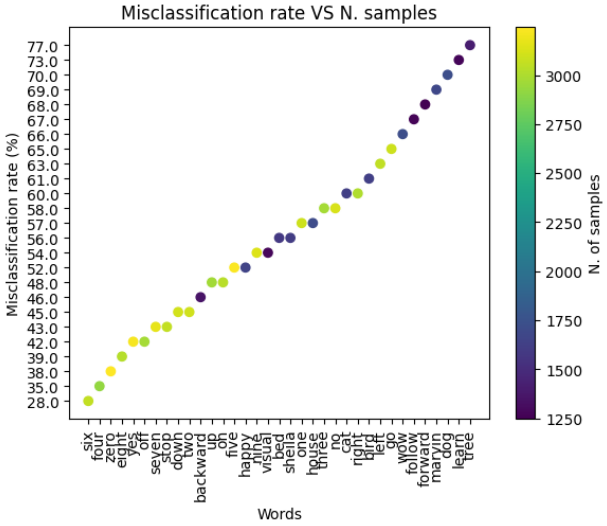

Figure 9: Misclassified VS N. of samples

Weights Visualization

A revealing method to comprehend the workings of the model is by examining the weights of the first layer, as displayed in Figure 11. In this visualization, regions shaded in red indicate a preference for a specific class, whereas blue areas suggest opposition. The distribution of these weights varies from word to word and aligns with their respective spectrograms. Given that the spectrograms represent frequency on the y-axis and time on the x-axis, one can discern the frequencies at which the model predominantly focuses.

For instance, higher frequencies tend to disadvantage words like *Follow* while benefiting words such as *up* and *wow*. In cases where words like *Forward*, *Four*, and *Follow* have similar spectrograms, the model tends to assign similar weights to the corresponding frequencies. This alignment demonstrates the model's capability to identify crucial acoustic features, as evidenced by the correlation between the weights assigned and the spectrograms of specific words, shown in Figure 10.

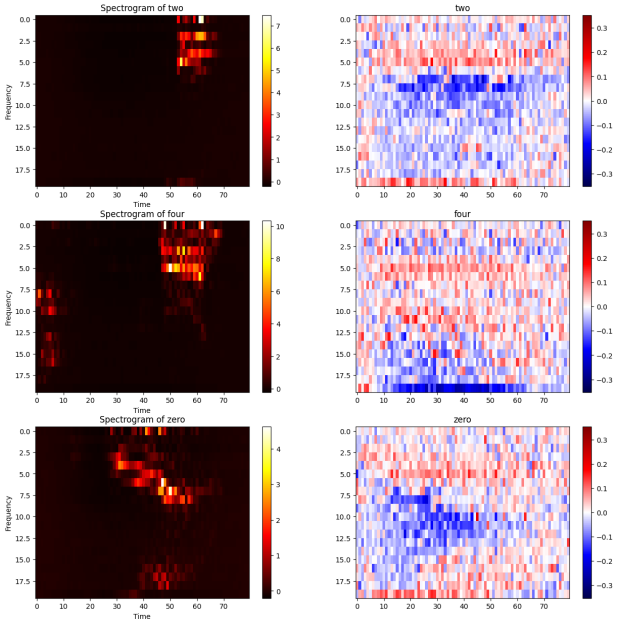

Figure 10: Spectrogram VS Weights

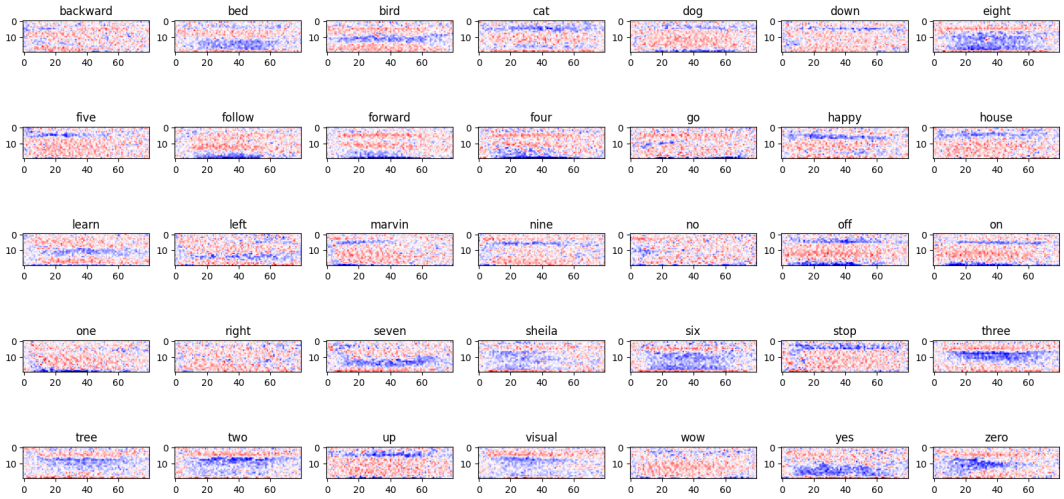

Figure 11: Weights visualization

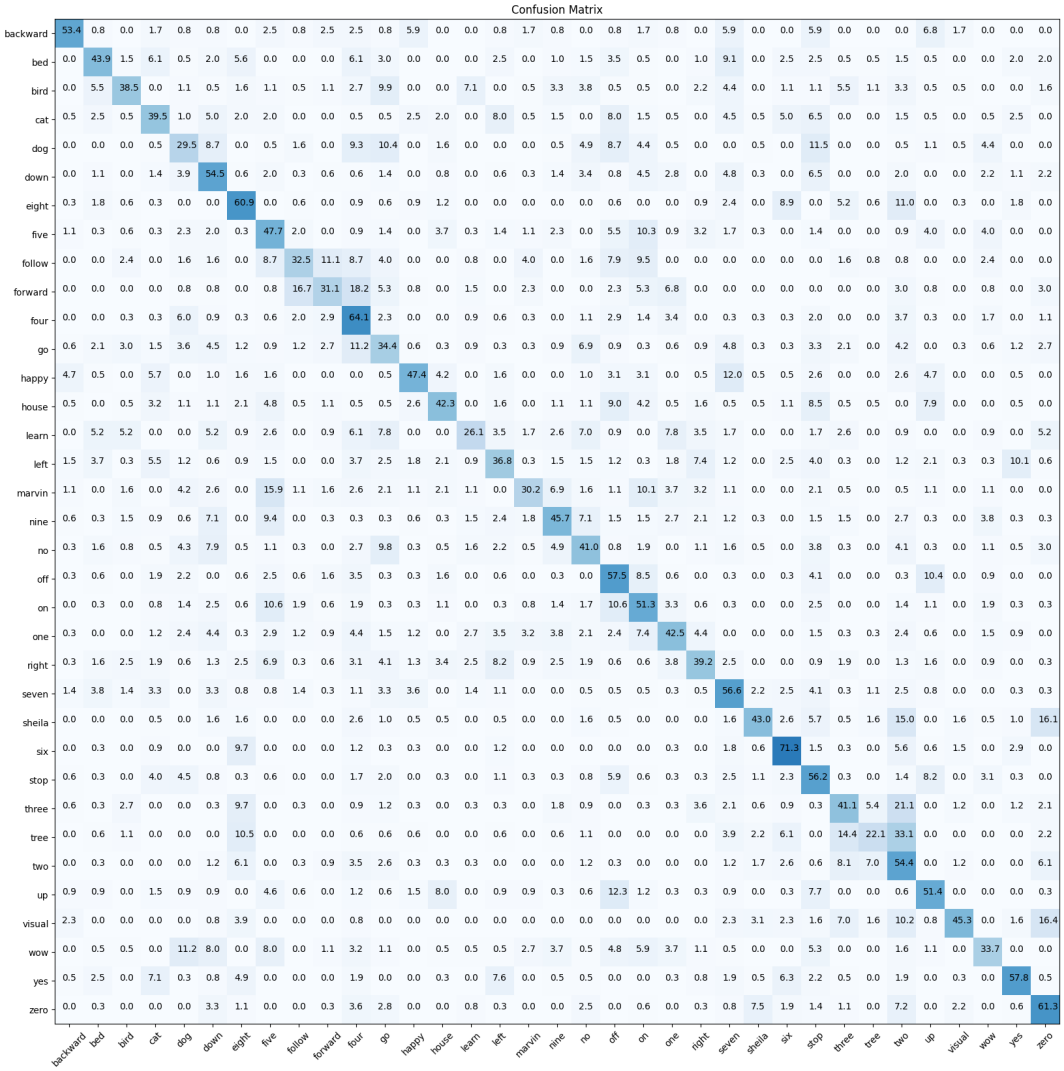

Figure 12: Test accuracies comparison

I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.