Speech commands

Cristian Andreoli

May 2023

Contents

Abstract

The task of speech recognition involves inferring words from a voice recording. There are applications where only a specific word or a short sentence needs to be recognized, such as trigger word detection. In this lab exercise, the focus is on recognizing a single word from a list of 35 words, which is a 35-class classification problem. The Speech Commands Data Set is used, which includes 105,829 recordings of the 35 words. Feature extraction has already been performed, and the features are spectrograms. The lab activity involves visualizing the data, normalizing the features, training a neural network with different architectures, and analyzing the results with a confusion matrix and classification errors. Finally, the scripts are refined and experiments are repeated with different network architectures. Additionally, the effectiveness of different feature normalization techniques is compared, and the set of weights of the MLP without hidden layers is visualized.

1 Lab activity

1.1 Preliminaries

The Speech Commands Data Set includes 105,829 recordings of 35 words, which are divided into training, validation, and test sets. Feature extraction has already been performed, and the features are spectrograms that encode the power distribution of the signal over a set of frequencies and a given time period. Spectrograms have been made uniform in size, each represented as a vector of 1600 components. The original clips can be downloaded from a web address, and the features are stored in different files. To classify the data, multilayer perceptrons will be used.

1.2 Visualize the data

Figure 1: Spectrogram of the data.

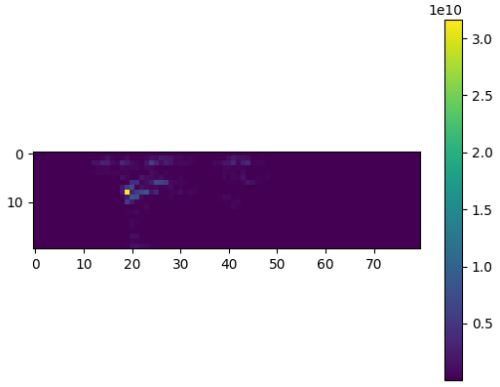


Figure 2: Mean of the data.

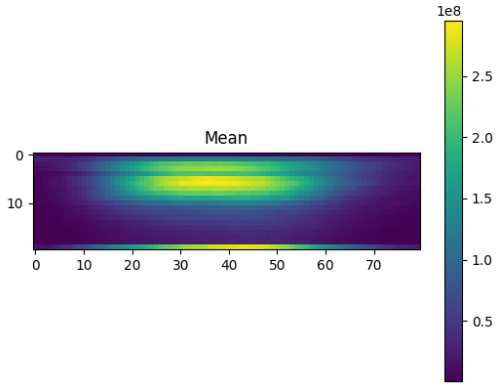


Figure 3: Variance of the data.

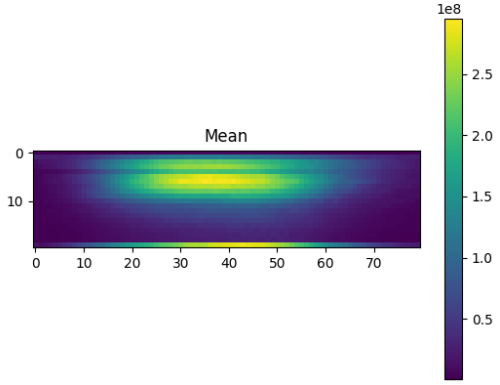
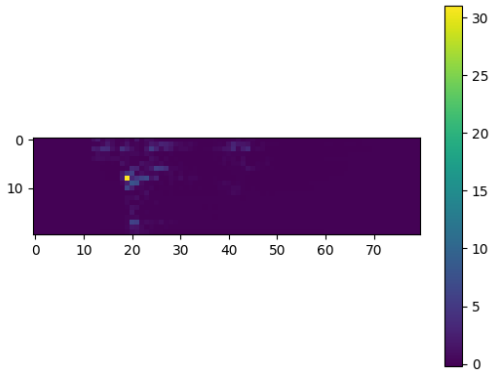


Figure 4: Spectrogram of the normalized data.



1.3 Feature normalization

There is no one-size-fits-all answer to which normalization technique is the best for speech command recognition with a small number of words. The effectiveness of different normalization techniques may vary depending on the specific dataset and classification problem. Therefore, it is recommended to experiment with different normalization techniques and evaluate their performance to determine the most suitable normalization method for the specific task at hand.
Let's start with mean-variance normalization. Mean normalization and variance normalization can help improve the performance of neural network MLP in several ways:

- Reducing the impact of outliers: Mean normalization can help reduce the impact of outliers in the dataset by centering the data around the mean value. Outliers can have a significant effect on the training process and can cause the model to overfit the training data, resulting in poor generalization performance;
- Reducing the impact of different scales: Variance normalization can help address input scaling issues by scaling all input features to a similar range. This can prevent features with larger scales from dominating the training process and making it harder for the model to learn from other features;

- Improving convergence speed: Variance normalization can also help improve the convergence speed of the training process. By scaling the data to have unit variance, the gradients of the cost function become more consistent, which can help the optimizer converge to the minimum more quickly;
- Making the model more robust: Variance normalization can help make the model more robust to changes in the input data. By scaling the data to have zero mean and unit variance, the model becomes more invariant to changes in the scale and distribution of the input data;

Figure 5: Results with different Learning rate and normalization techniques.

	MLP	epoch	lr	steps	batch	train acc	test acc
NO NORMALIZATION	[1600,35]	100	1e-4	1	all the samples	19.21%	14.41%
NO NORMALIZATION	[1600,35]	100	1e-3	1	all the samples	18.94%	14.37%
NO NORMALIZATION	[1600,35]	100	1e-5	1	all the samples	18.51%	14.55%
NO NORMALIZATION	[1600,35]	100	1e-2	1	all the samples	19.18%	14.69%
MEAN VARIANCE	[1600,35]	100	1e-2	1	all the samples	17.75%	15.30%
MEAN VARIANCE	[1600,35]	100	1e-3	1	all the samples	09.76%	10.04%

Unfortunately, it doesn't seem to have given significantly better results.

1.4 Train a neural network

Let's now try with different batch size and look at the results.

Figure 6: Results with different batch size.

	MLP	epoch	lr	steps	batch	train acc	test acc
NO NORMALIZATION	[1600,35]	100	1e-3	m//16	16	21.25%	13.71%
NO NORMALIZATION	[1600,35]	100	1e-3	m//100	100	28.49%	17.57%
NO NORMALIZATION	[1600,35]	100	1e-3	m//400	400	29.25%	18.33%

We can observe an improvement because this this makes the algorithm faster and more efficient.

1.5 Network architecture

Let's try different network architecture adding more hidden layer. Remember that adding more hidden layers to a multilayer perceptron (MLP) can increase the model's capacity and potentially improve its performance. The additional layers allow the model to learn more complex and abstract representations of the input data. However, adding more hidden layers also increases the risk of overfitting, where the model becomes too complex and fits the training data too closely, resulting in poor generalization to new data.

Figure 7: Results with different network architectures.

	MLP	epoch	lr	steps	batch	train acc	test acc
NO NORMALIZATION	[1600,100,100,35]	65	1e-3	m//100	100	0.6839	0.4148
NO NORMALIZATION	[1600, 1000, 200, 200, 35]	30	1e-3	m//100	100	0.76933	0.4588
NO NORMALIZATION	[1600, 1200, 35]	45	1e-3	m//100	100	0.8167	0.4748
NO NORMALIZATION	[1600, 140, 70, 35]	48	1e-3	m//100	100	0.6928	0.4147

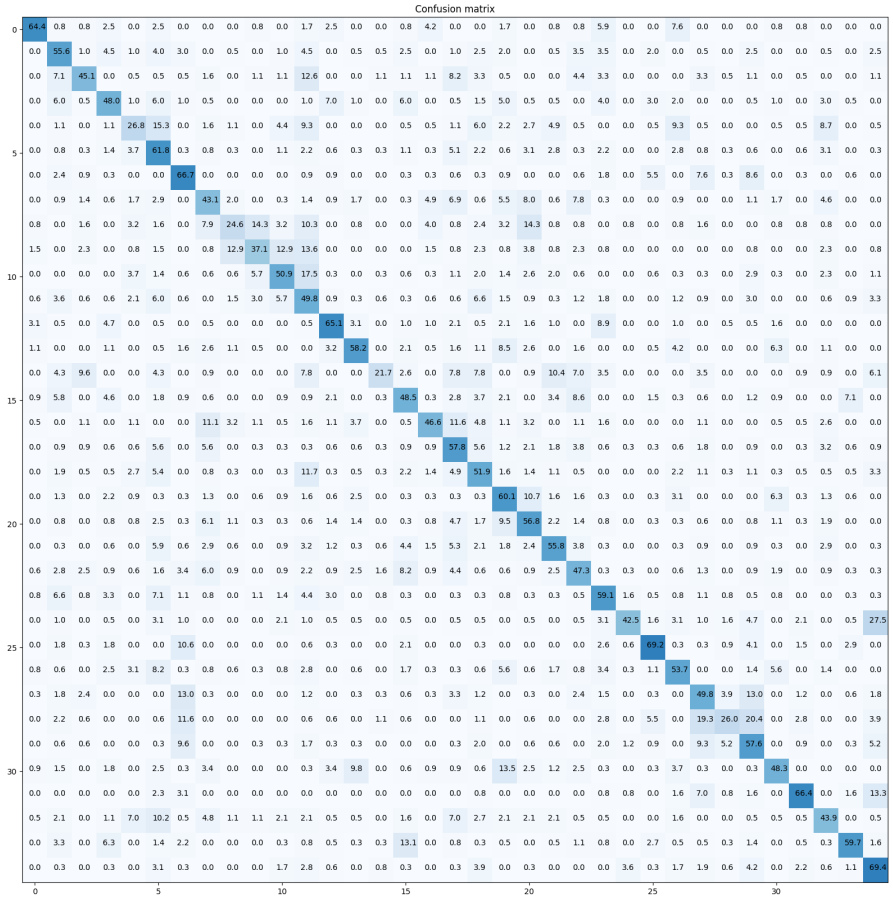
Some experiments were stopped earlier due to the huge amount of time needed for the training, or simply because there was no improvement.
We decided to continue the model with: L2 normalization, [1600, 140, 70, 35] architecture, batch size 100, learning rate 0.001.

1.6 Analysis

The confusion matrix is a useful tool in machine learning for evaluating the performance of a classification model. It provides a summary of the actual and predicted classifications of a set of data, allowing for a quick and easy comparison of the model's performance.
Some of the key utilities of a confusion matrix are:

- Evaluation of model performance: The confusion matrix provides information on how well the model is performing in terms of correctly identifying the different classes of data.
- Identification of errors: The confusion matrix can help to identify the types of errors that the model is making, such as false positives and false negatives. This information can be used to improve the model and reduce errors in future predictions.

Figure 8: Confusion matrix.



We can observe that there are some class that are not good classified, such as class 8, 14, 28.

2 Assignment

2.1 Feature normalization

With the chosen model we tried more normalization techniques, lets summarize all the experiments done with the following figure.

Figure 9: All the experiments done.

	MLP	epoch	lr	steps	batch	train acc	test acc
NO NORMALIZATION	[1600,35]	100	1e-4	1	all the samples	0.1921	0.1441
NO NORMALIZATION	[1600,35]	100	1e-3	1	all the samples	0.1894	0.1437
NO NORMALIZATION	[1600,35]	100	1e-5	1	all the samples	0.1851	0.1455
NO NORMALIZATION	[1600,35]	100	1e-2	1	all the samples	0.1918	0.1469
MEAN VARIANCE	[1600,35]	100	1e-2	1	all the samples	0.1775	0.1530
MEAN VARIANCE	[1600,35]	100	1e-3	1	all the samples	0.0976	0.1004
MEAN VARIANCE	[1600,35]	100	1e-3	m//16	16	0.2125	0.1371
MEAN VARIANCE	[1600,35]	100	1e-3	m//100	100	0.2849	0.1757
MEAN VARIANCE	[1600,35]	100	1e-3	m//400	400	0.2925	0.1833
MEAN VARIANCE	[1600,100,100,35]	65	1e-3	m//100	100	0.6839	0.4148
MEAN VARIANCE	[1600, 1000, 200, 200, 35]	30	1e-3	m//100	100	0.76933	0.4588
MEAN VARIANCE	[1600, 1200, 35]	45	1e-3	m//100	100	0.8167	0.4748
MEAN VARIANCE	[1600, 140, 70, 35]	48	1e-3	m//100	100	0.6928	0.4147
L1	[1600, 140, 70, 35]	31	1e-3	m//100	100	0.1983	0.1880
L2	[1600, 140, 70, 35]	48	1e-3	m//100	100	0.8101	0.5258
maxabs	[1600, 140, 70, 35]	100	1e-3	m//100	100	34.1815	29.1062
whitening	[1600, 140, 70, 35]	46	1e-3	m//100	100	65.9334	17.1821
maxabs	[1600, 140, 70, 35]	100	1e-3	m//100	100	34.1815	29.1062

Let's have a look at some detailed graph of the training:

Figure 10: maxabs normalization, [1600, 140, 70, 35] architecture, 100 iterations, 0,001 learning rate, batch size 100. I initially decide to trust this normalization due both of train and test accuracy were growing.

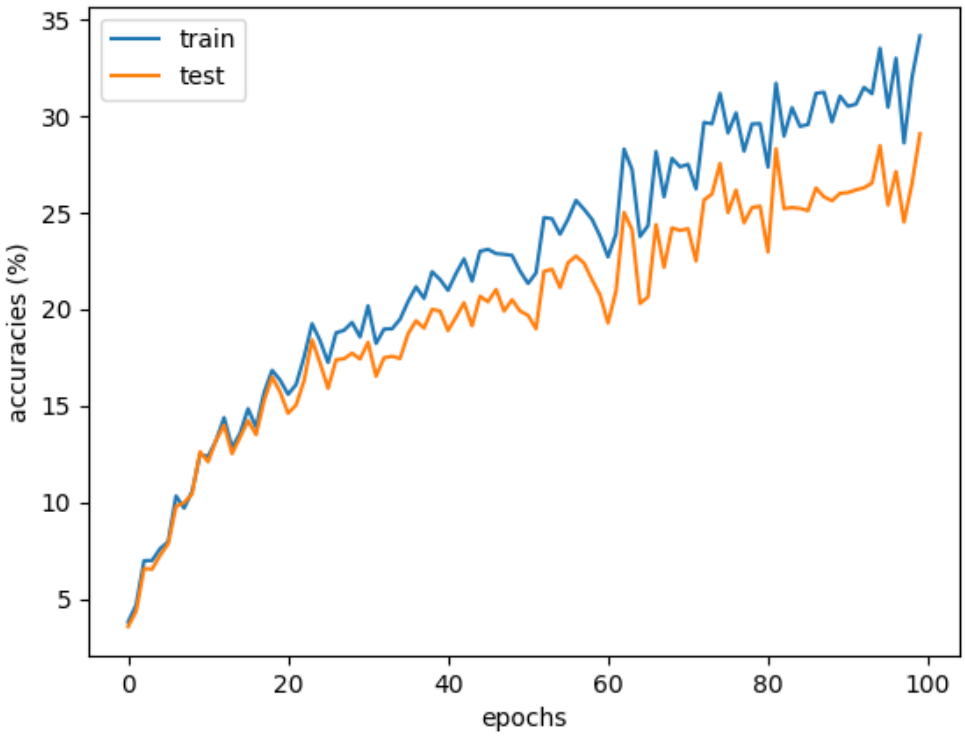


Figure 11: maxabs normalization, [1600, 140, 70, 35] architecture, 100 iterations, 0,001 learning rate, batch size 350. Unfortunately the model proved to be no better than the previous one.

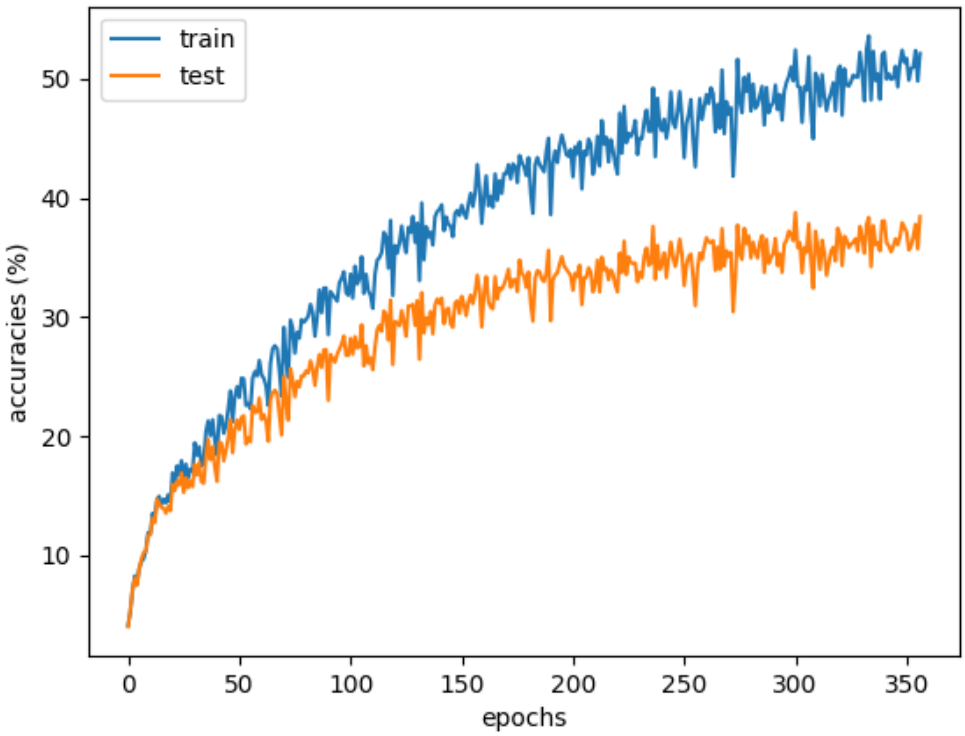
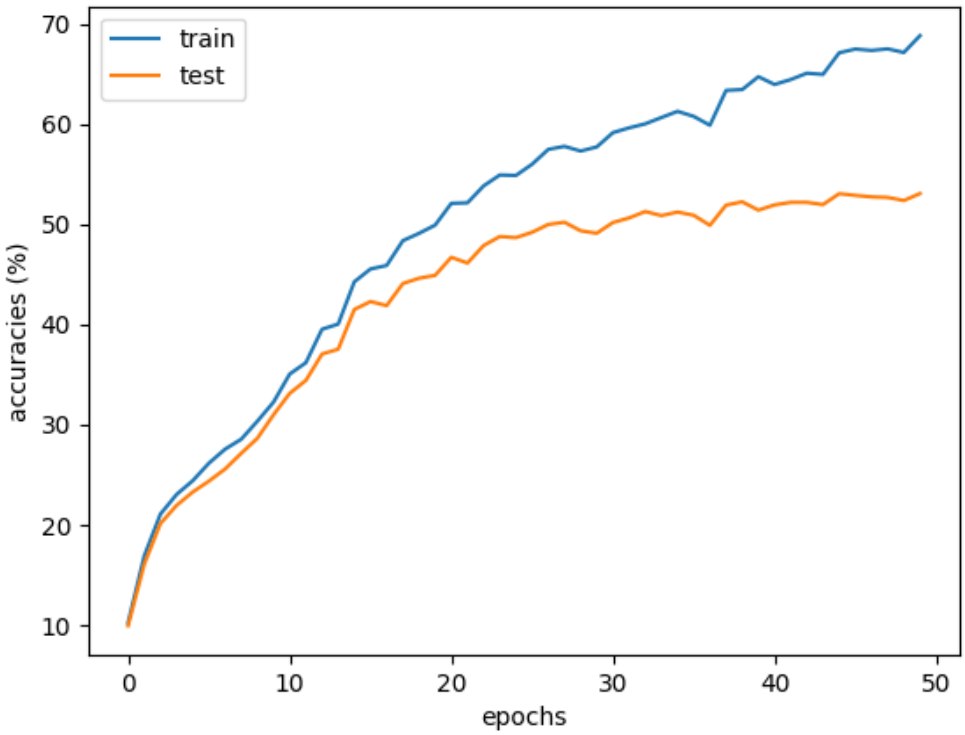


Figure 12: L2 normalization, [1600, 140, 70, 35] architecture, 50 iterations, 0,001 learning rate, batch size 100.



2.2 Visualization

Let's have a look at the weights.

Figure 13: Weights for the chosen model.

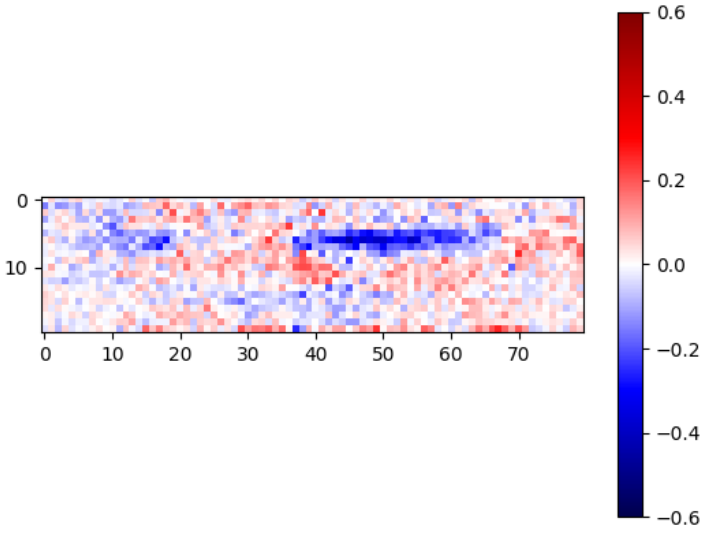
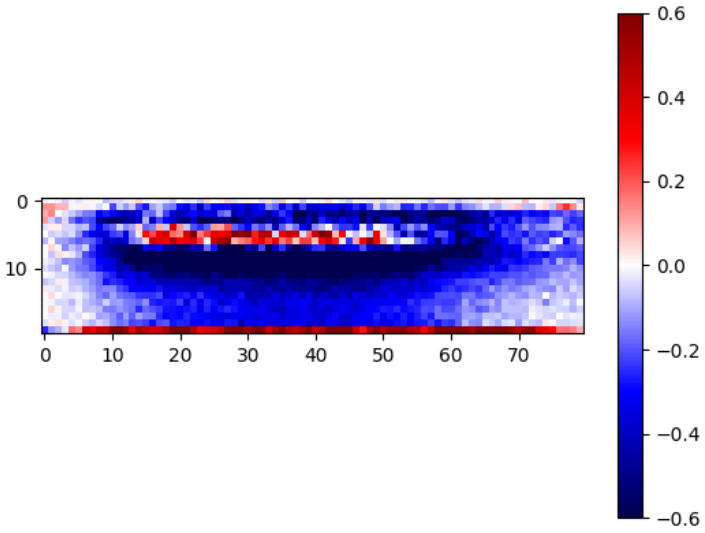


Figure 14: Weights for L2 normalization, [1600 35] architecture, 100 iterations, 0,001 learning rate, batch size 100 model. We can see the most relevant part used by the model to identify the class in the higher part, the strokes between blue and red.



2.3 Report

"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."