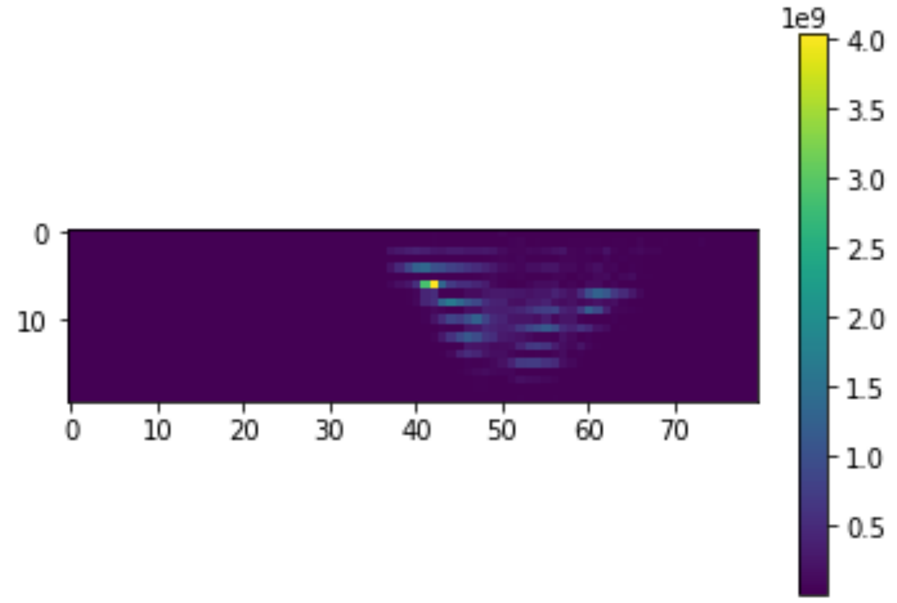
Speech commands
Machine Learning assignment

Marija Maneva

Abstract
The following assignment is about speech recognition, which involves determining the words spoken by a person based on an audio recording.
In general, speech recognition systems are designed to transcribe any meaningful sequence of words accurately. In this case, we are specifically focusing on a speech recognition scenario where the audio represents the pronunciation of a single word chosen from a predefined list of 35 words. The aim is to classify the spoken words in one of the 35 predefined classes. The input is the audio signal and the output is the corresponding word class from the list.

1.1 Visualize the data



In the spectrogram the range of values of the features represents the intensity or magnitude of the frequency components in the audio signal at different time intervals. The spectrogram is a visual representation of how the spectral content of a signal changes over time.
In this spectrogram we can see an higher intensity values in more or less 40 on x-axis. This indicates that there are stronger frequency components in that interval.

1.2 Feature normalization
I normalized my data by using three different techniques :
- mean/variance normalization
- min/max scaling normalization
- L2 normalization

I calculated the spectrograms in everycase and I trained the data by applying these techniques separately. I obtained the best results with the mean/var normalization , it can be seen from the spectrogram in which I have higher intensity values in all the range and from the accuracies. In fact, with mean/var normalization the model reached higher accuracies, up to 20%.

1.3 Train a neural network

To train a neural network with batch gradient descent and with stoъchastic gradient descent I used the data normalized with mean-var normalization.

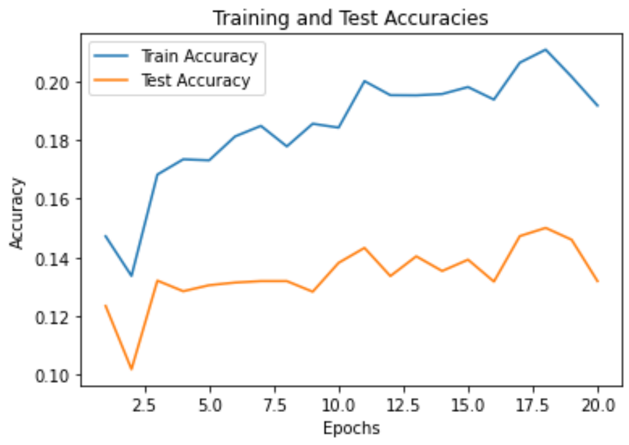
- In order to train the data with gradient descent I put the number of batches equal to the size of the data set (m in my case)
- In order to train the data with stochastic gradient descent I changed the number of batches to 1 and then I tried to evaluate the model with minibatches of different sizes (2,10,100)

The main difference between gradient descent and stochastic gradient descent is that SGD is an extension of the GD method that updates the model parameters based on the gradients coŭmputed on individual training examples or small subsets of data called minibatches.

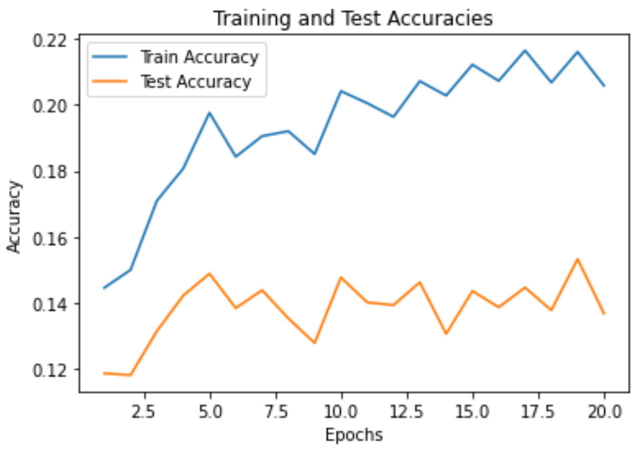
Without hidden layers

Stochastic gradient descent

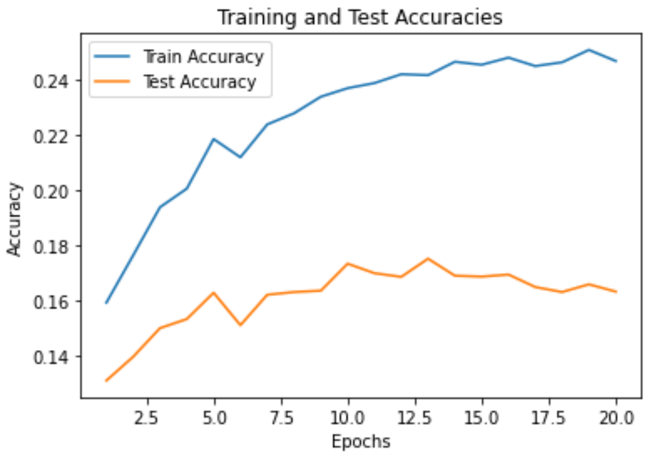
n.batches = 1



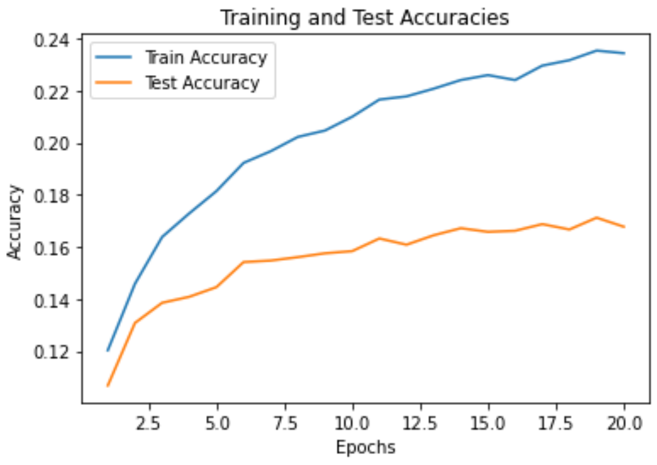
n.batches = 2



n.batches = 10



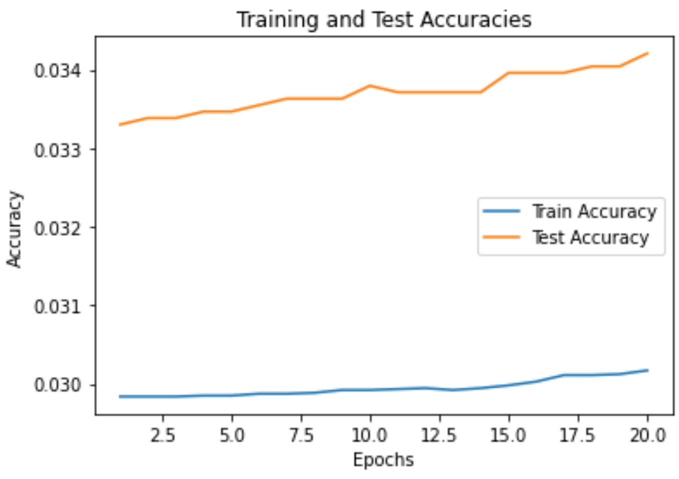
n.batches = 100



Using smaller minibatches (e.g 1 or 2) increases the noise in the parameter updates, bu it makes the updates also more frequent. This leads to faster convergence and better generalization.

Using larger minibatches (e.g 10 or 100) reduces the noise in the updates, but it can also make the updates less frequent. This slows down the convergence process. They can also lead to more stable updates.

Gradient descent



conclusions :

- SGD is faster than GD because it updates the model parameters more frequently
- GD converges to the optimal solution in fewer steps compared to SGD
- GD provides more stable updates since it uses the complete dataset
- SGD provides a better generalization

1.4 Network architecture

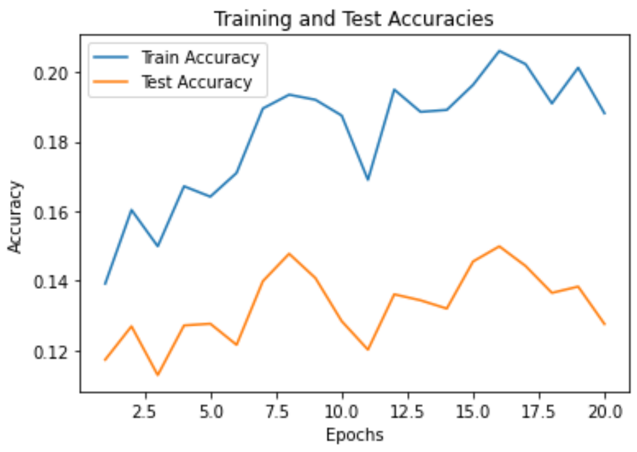
With hidden layers

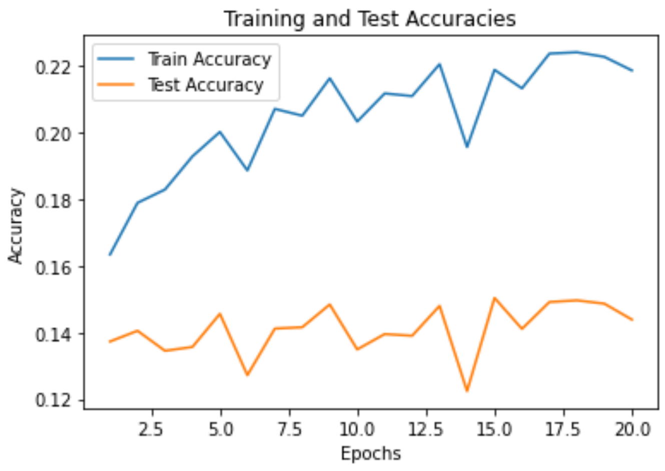
I add hidden layers to the model by adding in the pvml.MLP([1600, 35]) the value of the width I want my layer/layers to have.

1 hidden layer : 100

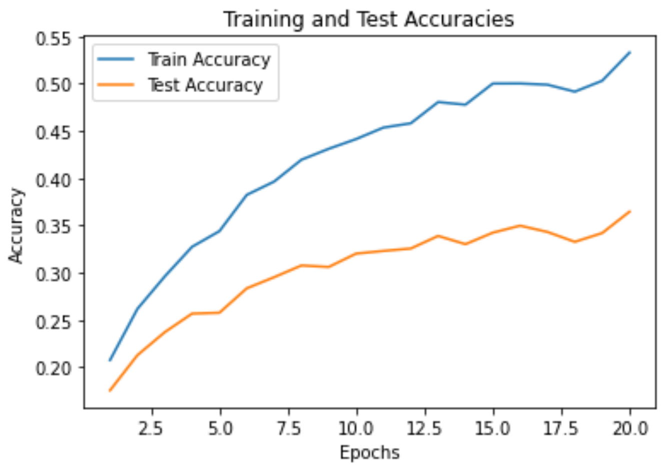
Stochastic gradient descent

n.batches = 1 n.batches = 2

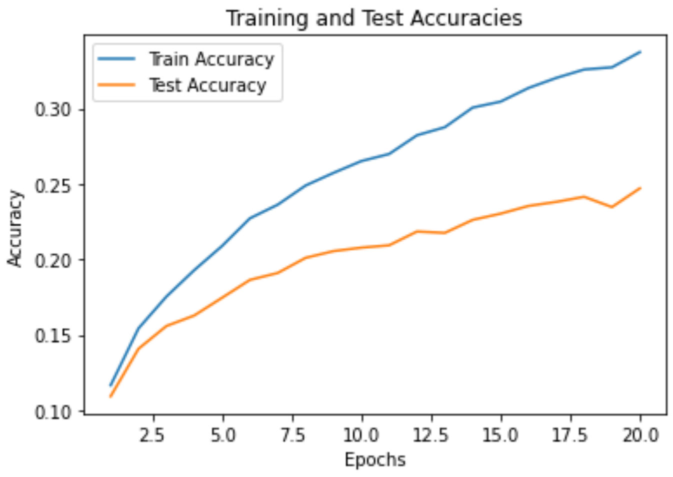




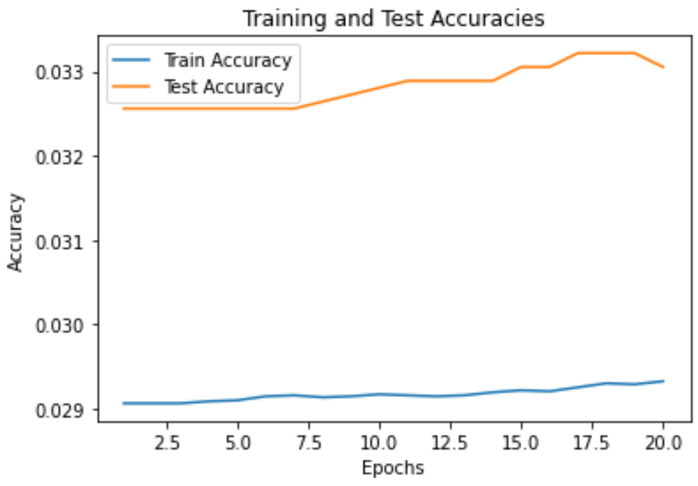
n.batches = 10



n.batches = 100



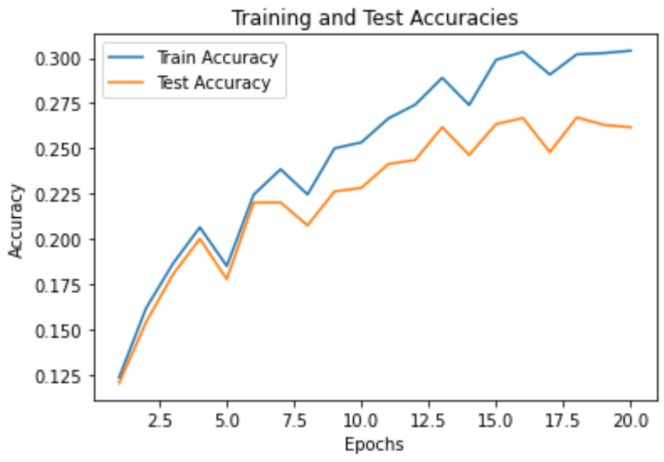
<u>Gradient descent</u>



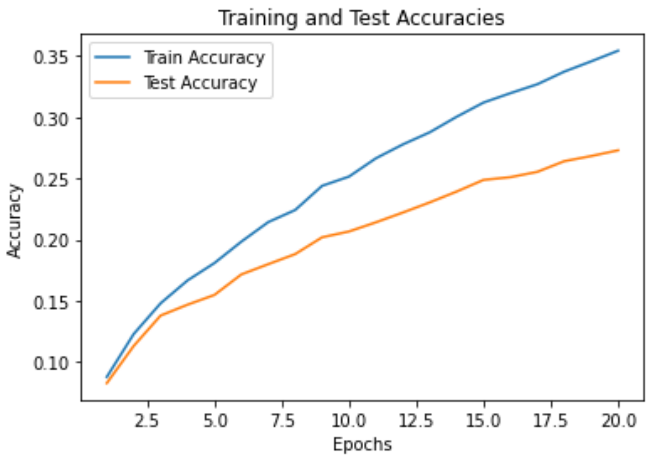
2 hidden layers : 100 & 50

<u>Stochastic gradient descent</u>

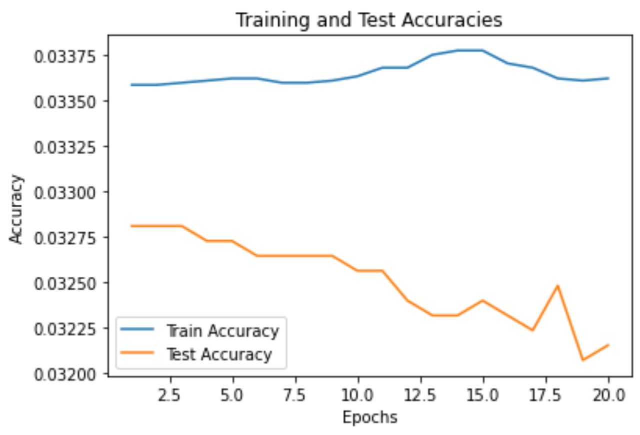
n.batches = 1



n.batches = 100



Gradient descent

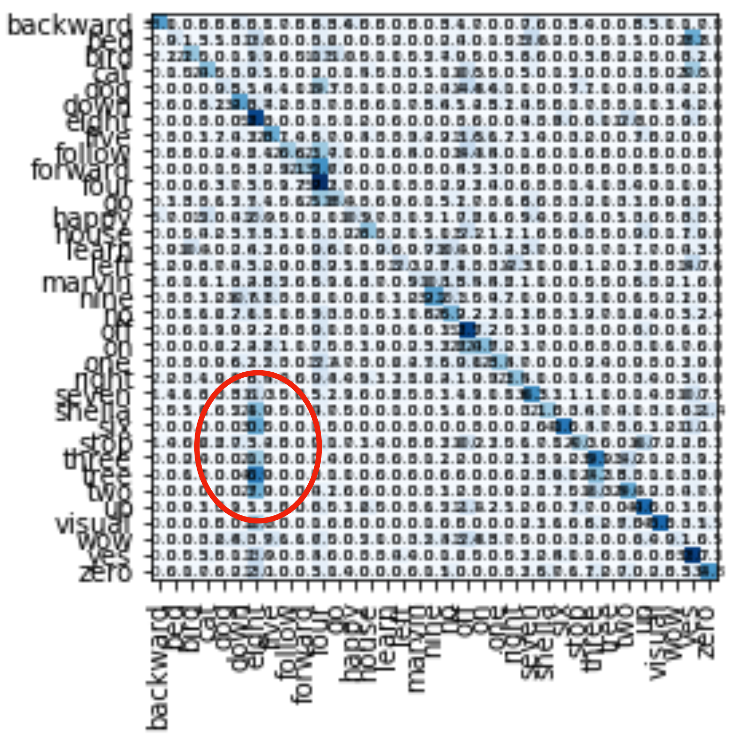


Effect of more layers:
Adding Hidden layers increases the capacity of the neural network to learn more complex representations and capture more intricate patterns from the data. This improves the model's ability to recognize spoken digits.
Each hidden layer learns different levels of abstraction. With more Hidden layers, the network learns more meaningful and informative features from the data.
Adding more layers leads also to a slower convergence.

Effect of changing width:
A wider layer captures more complex relationships in the data, improving the performance.
Increasing the width (also number of layers) can lead to overfitting.

1.4 Analysis

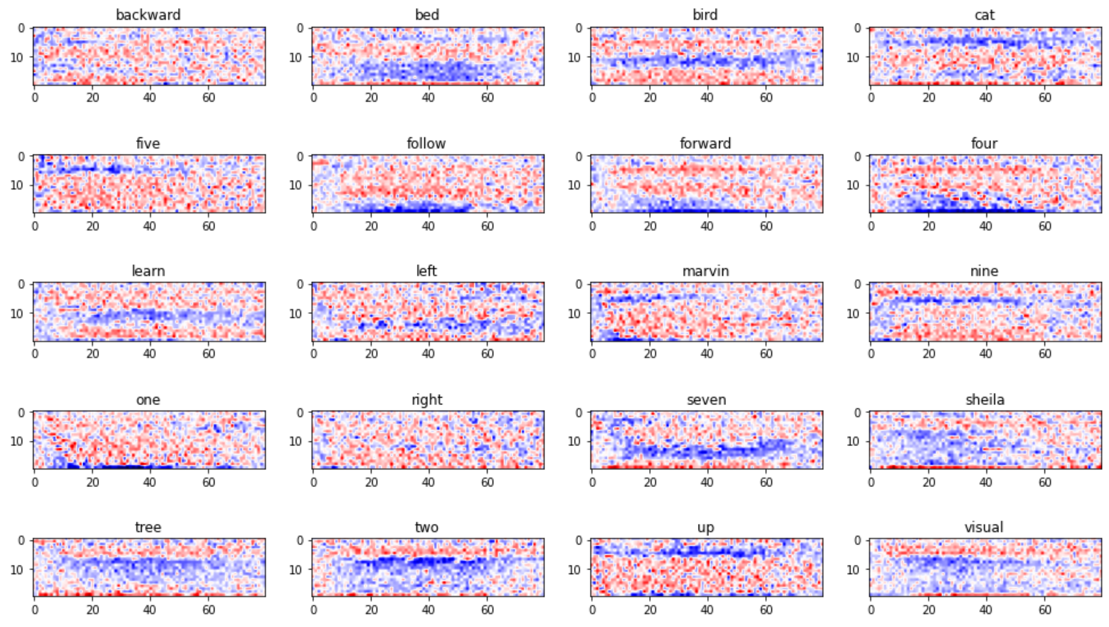
In a confusion matrix, the classes that are more likely to be confused can be identified by examining the off-diagonal elements. These elements represent the misclassifications where the predicted class is different that the true class. Higher values in these off-diagonal elements indicate a higher confusion between specific classes.
To identify which classes are more likely to be confused I looked at the highest values in the off-diagonal elements.
I computed the confusion matrix by applying each of the three normalization methods I have considered in this experiment. I obtained a more accurate confusion matrix with the mean/var normalization and by adding more layers the accuracy increases and it can bee seen a better classification.
In the case of confusion matrix with mean var normalization and with one hidden layer.

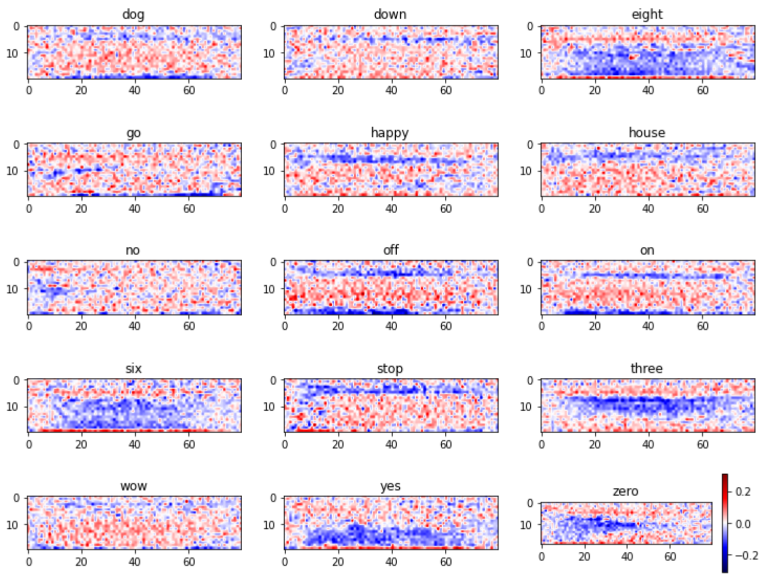


It can be seen that there are higher values in the red area which indicates confusion between the classes “eight” and “two”,”three”,”tree”,”six” and “shella”.

The kind of samples that are easily misclassified are those that:
- sound similar when spoken. In this case for example, as I have seen from the matrix confusion, the classes “three” and “eight” might have similar acoustic characteristics making them prone to confusion
- sharing certain acoustic features : some of the digits can be difficult to differentiate because of overlapping or ambiguous features. For example they can have similar patterns in the frequency domain especially when the pronunciation is unclear

2.2 Visualization





In this case, the spectrogram is referring to the data normalized with mean-var normalization and without hidden layers.
Each horizontal strip represents a frequency used by the model to predict the class. However, since it's a linear model, in many cases, the strips are similar and don't separate well from each other.
In order to choose the output class the spectrogram uses the weights of the model. The weights represent the Learned parameters that the model uses to make predictions; each Weight corresponds to a connection between an input feature and output class. In the spectrogram, the red part indicated the positive weights and the blue part refers to the negative ones.

"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."