Speech processing and recognition Project Report

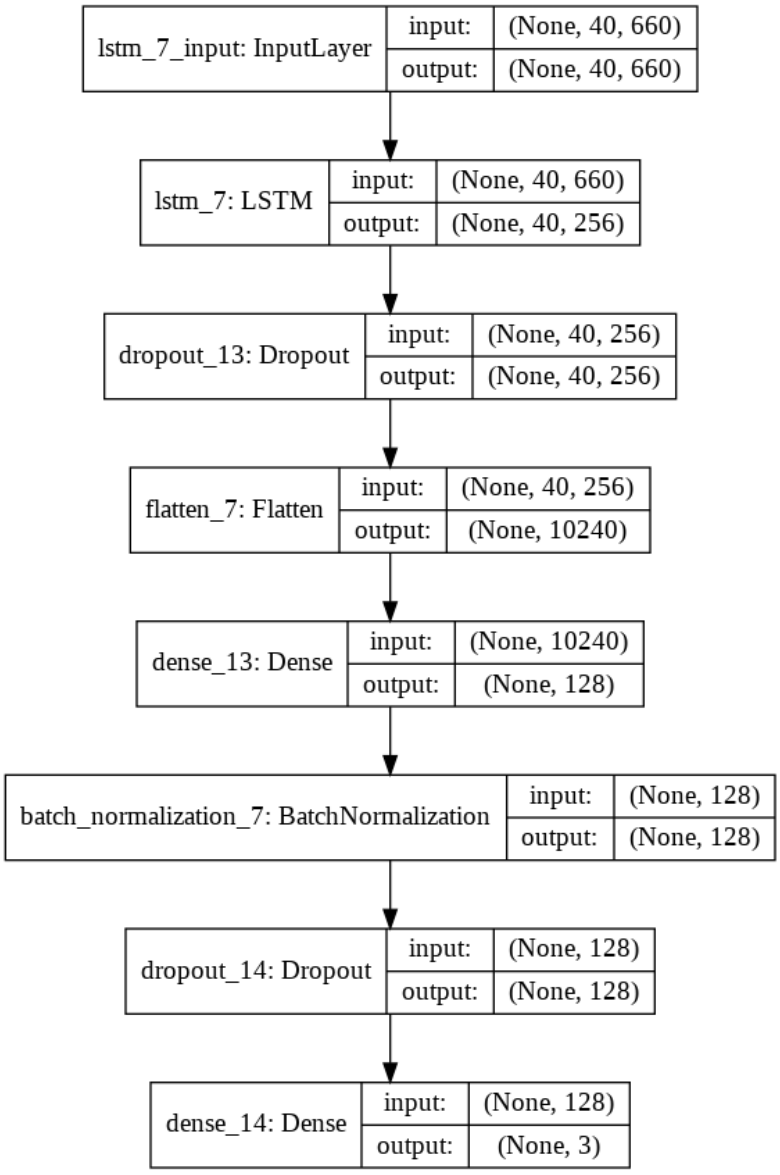
Arezou Ranjbarpour Maralani and Lorenzo Tibaldi

Introduction

We decided to develop a Speaker Verification Neural Network, we didn't download a dataset from the internet but opted instead to create a small one of around 100 wav files ourselves, for that purpose we used a free recording software: Audacity.

The project has been completely developed in python on Google collaboratory.

The main challenges of this project have been finding a satisfactory balance between a small dataset and a good neural network and finding empirically the best parameters.



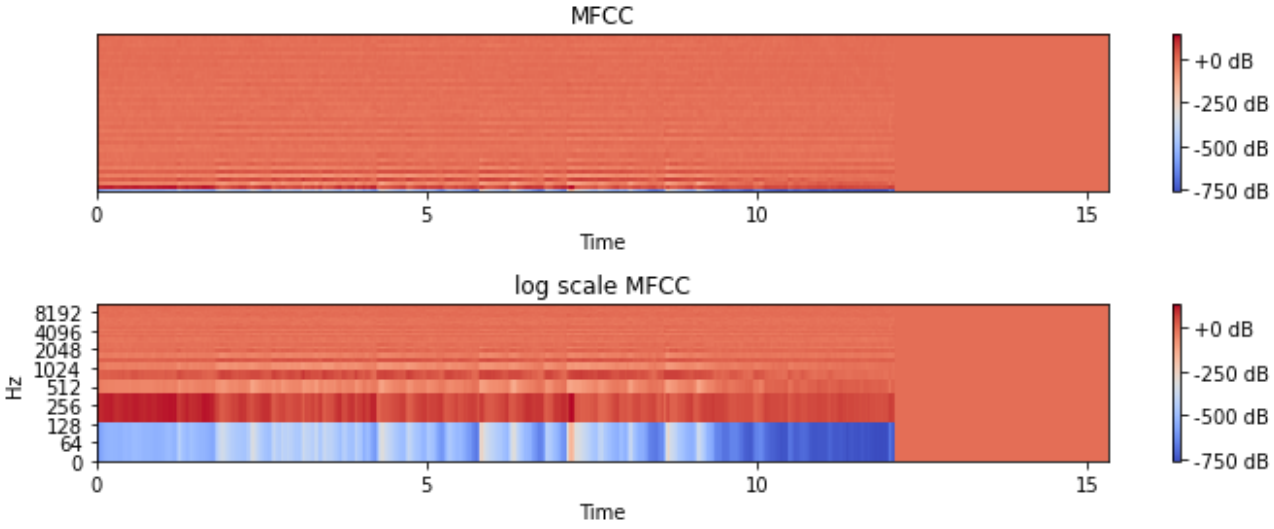
The Structure of the model used.

Implementation

We decided to use Mel-frequency cepstral coefficients as features of a Sequential model.

Our dataset contains audio files of lengths between 5 and 15 seconds, as such we had to pad the shorter ones so that all the inputs would be the same size when fed to the model.

The Input size of the model is (40, 660), 40 is the number of mfcc that we settled on after trying some and comparing the model's accuracies, 660 is the length of all the audio samples (around 15 seconds because of the padding)



A single data point (all the mfcc of one voice line) where the padding on the right can be seen.

The main layers of the model are:

- A LSTM layer with 256 neurons.
- A Dense layer with 128 neurons with ReLU as activation function.
- Lastly a final Dense layer with 3 neurons and softmax activation function.

Considerations

Since we didn't want to record hours of voice lines we decided to use a recurrent layer (the LSTM layer) because recurrent layers help with few shots learning.

Both layers before the output layer are followed by a Dropout layer with a parameter of dropped connections equal to 50%, this number might look high but models that take advantage of recurrent layers are very prone to overfitting, that's the reasoning for the 0.5 value.

However doing a dropout after the LSTM is not enough, we also had to add a dropout of 33% inside the recurrent layer itself, another value that we found empirically.

A flatten layer is used to go from the LSTM section to the fully connected one.

A BatchNormalization layer is also used, we noticed empirically that right after the first Dense layer is the best place to have one.

Lastly the output layer contains 3 neurons instead of 1 (like a Speaker Verification model usually does) because we changed the scope a bit, we decided to create a model that would answer to the question: Is this audio from Lorenzo, Arezou or neither?

That is why we are using a softmax activation for the last layer, instead of learning from a database of 1 person and recognizing if that person is the speaker on a new sample we trained the model with 3 different types of audio files, 1 for Lorenzo, 1 for Arezou and 1 that contains a collection of noises.

At the very start of the recording step we were obtaining suspisciously good results, we theorized that the model might have learned to rely on the padding because one of the speakers would always record shorter voicelines, this is obviously a form of meta training that we weren't interested in, as such in the next recording sessions we alternated shorter and longer recordings between the 2 speakers, just to be sure.

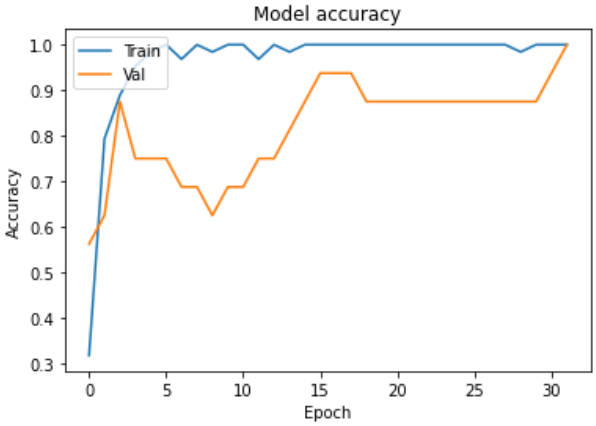
The final result got "worse" but the model got better.

Results

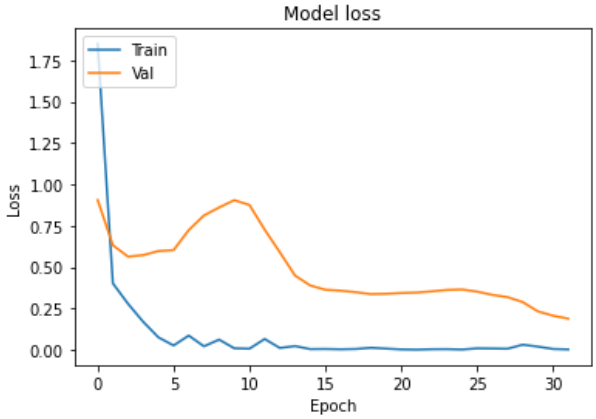
After a couple of tries we settled on a batch size of 16 and 32 epochs.

The data has been split into 64% training set, 16% validation set and 20% test set.

The training takes less than a minute with results that we deemed satisfactory.



One of the best iterations of the model. The accuracy obviously can't be mantained at 100% like the last epoch, but it's reasonable to believe that it would plateau at around 88% until overfitting.



This loss is an interesting graph aswell, it can be seen how the model reached a local minimum around 3 epochs but given enough time it was able to improve bit more.

The model stops its training during a monotonically decreasing sequence of losses, like we would expect from a good model.

In the end the test set is classified with an accuracy of 80%

Of course there is variation in the different executions but the test accuracy almost never dips under 65%, which we consider still a good result.

Reference: https://research.ijcaonline.org/volume117/number1/pxc3902361.pdf