Speaker Identification with CNNs

ARI3210: Speech Technology Assignment

Karl Attard

karl.attard.19@um.edu.mt

B.Sc. (Hons) Artificial Intelligence
University of Malta

Contents

The implementation for this project can be found ***here***.
This Google Drive link contains a Jupyter Notebook implementing this assignment as well as 2 sub-folders; original corpus and generated corpus containing all chunked Mel-Spectrograms for each speaker.

1 Introduction

Speaker Identification (SID) is the problem of identifying the identity of the speaker given a speech audio, chosen from various known speakers. Subsequently, to build a SID system, a corpus of speech data for different speaker need to be collected, and models are then built to solve such a problem. This problem can be solved in using different approaches; generative, discriminative and neural.

In this assignment, a corpus of speech data is given, where each folder name represent a unique speaker. Inside every folder, three distinct audio wav files of long duration is found, where the speaker is reading a passage. Furthermore, Convolutional Neural Networks (CNNs) based classifier are built after pre-processing the corpus, which will be able to learn to determine speaker identity based on a voice sample from a speaker.

This research paper is organised as follows: Section 2 below highlight the implementation methodology, Section 3 evaluated every model built by using plots whilst Section 4 summarises this research paper.

2 Methodology

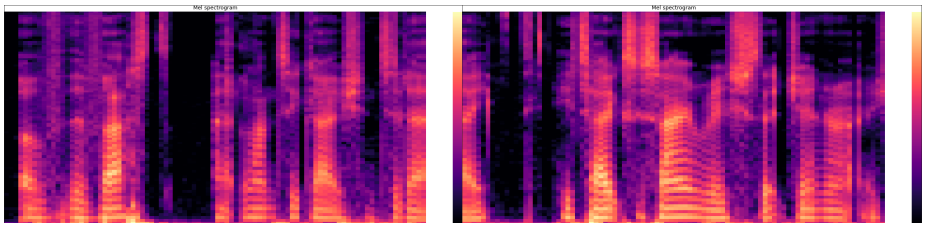
This section describes the methodology used to solve the SID problem, starting from the pre-processing of the corpus up to building the models.

2.1 Pre-Processing the Speech Corpus

Pre-processing the speech data corpus included to iterate over the entire given corpus and generate the needed CNN training data. Hence, this is achieved by going through all the speaker folders and all their respective audio wav files. Then, the output generated will be stored inside another folder mirroring the current corpus structure (i.e., every folder corresponds to a unique speaker), and this generated corpus will contain the data images files for each speaker (the training data).

To achieve this, a function called 'dataset_generation()' is created. This implemented function generates the aforementioned corpus containing the CNN's training data for every speaker. This function iterates over every speaker folder, and generates a mel-spectrogram for every wav file. Since the generated mel-spectrograms is of size $M\ X\ T$ but the input supplied to the CNN classifiers must be $M\ X\ M$ (sqaure image), then the generated mel-spectrogram need to be scanned and chunked appropriately into $M\ X\ M$ images, where each chunk will be saved to disk as a PNG file. Figure 1 below is an example of a Mel spectrum chunks generated.

After having successfully generated and saved all chunked mel-spectrograms to disk, a list of every path of each chunked image are stored. This list was not saved to a text file but was just displayed to the user since Jupyter Notebooks was used for this implementation, and thus, was not needed.



(a) First Chunk Image (b) Second Chunk Image

Figure 1: Chunk Mel-Spectrum Images Generated Examples

It is important to note that to convert an audio file to a Mel-spectrum, the time-domain signal was normalised to a range between -1.0 and 1.0 to take care of any differences in scale of amplitude, using the 'normalize' function found in the Librosa documentation [1]. Furthermore, after extracting the mel-spectrum, decibel (dB) conversion is applied to all before saving. Apart from these, important parameters values choices had to be made. The following are the chosen parameters values:

- mel_spec_frame_size = 1024
- n_mels = 90

The first parameter refers to the size, in samples of each frame to be processed. This is dependent on the sampling rate of the audio files. The second parameter refers the the resolution of the image. The majority of the audio files (not all where checked) had a sampling rate of 22050, and it was decided that every frame will be approximately 21/22ms. Therefore, the former parameter was set to 1024 whilst the latter parameter was set to 90, which is a good balance between 64 and 128 that yields good image resolution for our images.

2.2 Data Splitting

Prior to training and evaluating the built CNN classifiers, it was imperative to split the corpus in the best way possible. Hence, the corpus was split 20% for test set and 80% for training and validation set. The latter 80% is the subsequently split into 80% for training set and 20% for validation set. These sets are all used when building and running the models.

2.3 CNN Design, Training and Evaluation

This section describes the both CNN classifiers implemented in terms of design, training and evaluation. These are the ShallowNet and Novel Network CNN models which are used to solve the SID problem.

2.3.1 ShallowNet

The design for this ShallowNet model included only one convolutional layer (without pooling), using the *RELU* activation function and one flattened dense layer. Finally, a softmax classifier is used to predict the output.

The ShallowNet was built using the following important hyperparameters values:

- Epochs = 500
- Learning Rate = 0.005
- Batch Size = 8
- Kernel Size = (3x3)

2.3.2 Novel Network

The design of the Novel Network includes the following: 5 convolutional layers, each having a max pooling layer, RELU activation function as well as a dropout layer (avoids over-fitting). Moreover, this designed Novel Network has 3 fully connected dense layers, with the first 2 being flattened and also have a dropout layer. Finally, a softmax classifier is used to predict the output. The built model is a relaxed version of the AlexNet CNN classifier and has the following hyperparameters:

- Epochs = 750
- Learning Rate = 0.01
- Batch Size = 8
- Convolution Filters = 32
- Kernel Size = (3x3)
- Stride = (2x2)
- Pool Size = (2x2)

For the models to classify well on unseen data, a number of convolutional layers were added, which included the aformentioned addition of layers (like pooling, dropout etc) together with more dense layers. This was done in order for the models to learn more abstract features from the chunked mel-spectrogram, and thus, would be able to perform with higher accuracy on the test data. The number of epochs was increased to allow the model sufficient time to learn these abstract features, and learning rate was slightly increased in order to learn not only faster, but also to not get stuck in local optima. On the other hand, the maximum pooling layer is chosen to select the maximum value within that pool size to reduce the dimensions of the image but at the same time not eliminating important features. The number of convolutional features was set equal to the N size of the chunked image, whilst the kernel size was set to (3x3) in order to allow for efficient computations.

2.4 k-Fold Cross Validation

The results of the ShallowNet and Novel Networks CNN classifiers built will undergo k-Fold Cross Validation, which essentially splits training/validation/test for 'k' times[1], and the results of each run are collated to average global results. Furthermore, a box-and-whisker plots are outputted over the 'k' folds, one for each CNN model, outlining the accuracy/precision/recall and f1-score in each box plot.

3 Evaluation

In this section, all built models will be compared with each other by using the generated reports and plots.

3.1 ShallowNet vs Novel Network

As previously mentioned, the ShallowNet and Novel Network models built differ in their architecture. The latter CNN classifier is a more sophisticated model than the former having higher depth of convolutional layers and dense layers as well as additional max pooling and dropout layers.

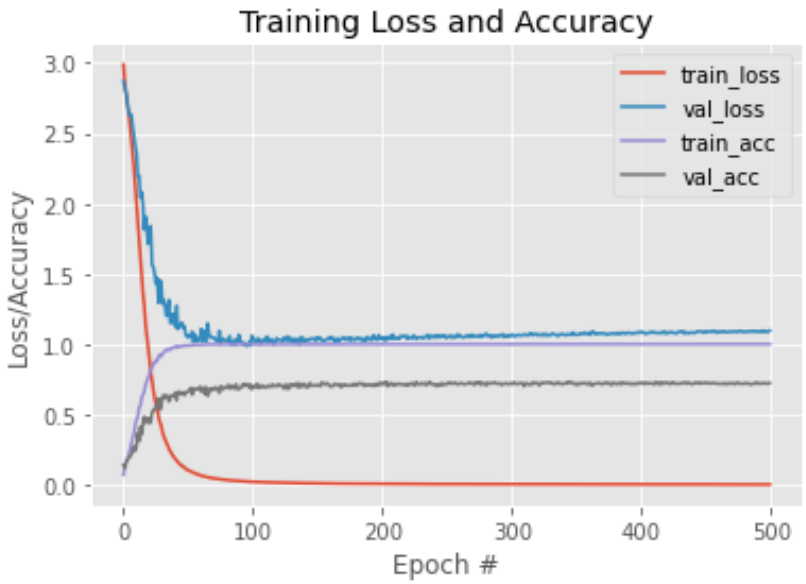


Figure 2: Performance Plot for ShallowNet

As illustrated in figure 2 above, it is the plot describing the Training/Validation loss and accuracy when training ShallowNet on the split training set.

[1] ***NB: For this project, the k value has been set to 5***

As can be observed, the training and validation loss begins at a relatively high number and then both decrease and converge to a number. For the training loss, it is converging to 0 whilst for validation loss, it is converging to approximately 1 after a number of epochs. On the other hand, the training and validation accuracy are beginning at a very low number but are then converging to a solution after a number of epochs, with the former converging to 100% whilst the latter converges to approximately 70%.

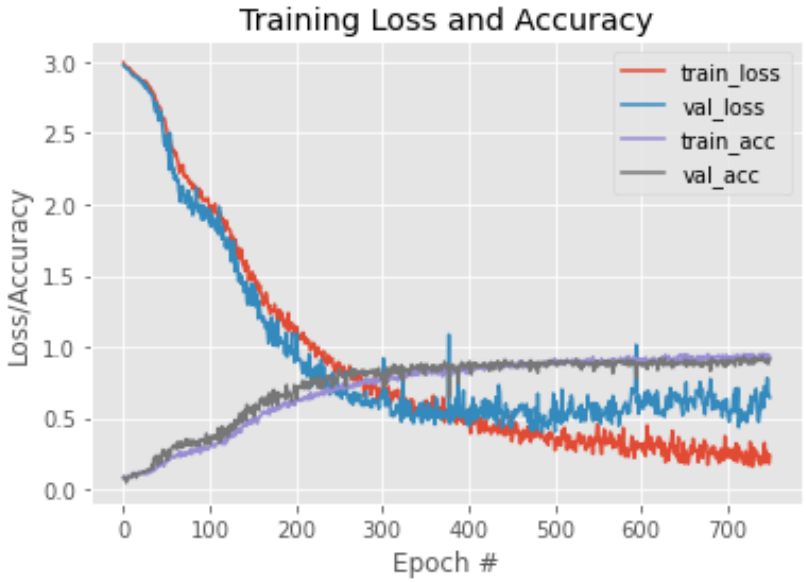


Figure 3: Performance Plot for Novel Network

As illustrated in figure 3 above, similar observations can be deduced. The training and validation loss starts at a high value but then decreases throughout the epochs, with the former loss converging to almost 0 whilst the latter loss converges to around 0.5. On the other hand, the training and validation accuracy begin very low but increase throughout every epoch until they converge after 750 epochs where training accuracy is reaches almost 100% and validation accuracy reaches around 90%.

Furthermore, the following table outlines the classification report of each model on unseen data (without k-fold cross validation).

As can be observed in table 1 below, the Novel Network model is performing much better than the ShallowNet, having an increase of 19% in accuracy only. However, in order to guarantee a fair evaluation, both these built models underwent 5-fold cross validation, and the following outputs were obtained.

Although obtained global average results in table 2 differ from those of 1, the same conclusion can be inferred. This is because the Novel Network model is performing much better on unseen data (21% increase in accuracy) and thus, is

	Accuracy	Precision	Recall	F1-Score
ShallowNet	72%	72%	68%	68%
Novel Network	91%	86%	87%	86%

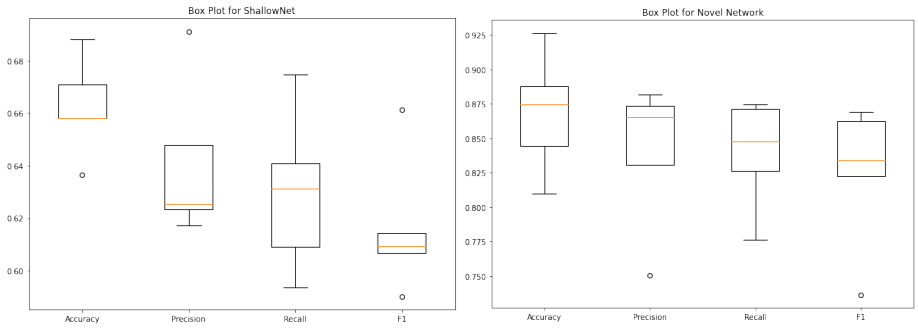
Table 1: Classification Report for each CNN model

	Accuracy	Precision	Recall	F1-Score
ShallowNet	66%	64%	63%	62%
Novel Network	87%	84%	84%	82%

Table 2: Classification Report for each CNN model after Performing 5-Fold Cross Validation

able to determining the speaker's identity with higher accuracy than the other ShallowNet model.

Furthermore, box-and-whisker plots were used to further analyse these CNN classifiers. Each box-and-whisker plot highlights the range of values, the upper and lower quartile values, the interquartile range as well as the median value. In figure 4 below, the obtained results over the 5 folds are presented, where each plot contains 4 distinct box-and-whisker plots denoting the accuracy, precision, recall and f1-score boxes respectively. In figure 4a below is the box plot for ShallowNet, and as can be observed all of its box-and-whisker plots vary a lot between one another, as opposed to figure 4b which highlights the box plot for Novel Network. In figure 4b, all box-and-whisker plots are close to one another and have higher values than figure 4a, and thus, the argument that Novel Network performs better than ShallowNet is better substantiated.



(a) ShallowNet Box-and-Whisker Plot (b) Novel Network Box-and-Whisker Plot

Figure 4: Box-and-Whisker Plots for both CNN Classifiers

4 Conclusion

This research paper focused on solving a popular problem in speech technology domain, i.e., Speaker Identification (SID). This problem has been solved using 2 different built CNN classifiers, one more sophisticated than the other. All of the models were trained on a given speech data corpus which has been pre-processed accordingly in such a way to be appropriate for the CNN classifiers. Both models implemented were then compared, and the Novel Network model proved to perform better than the ShallowNet model on unseen data, and thus, is better for the SID problem. The evaluation was carried out by outputting a classification report, by performing k-fold cross validation as well as by various plots for a better comparative study.

References

[1] B. McFee, A. Metsai, M. McVicar, *et al.*, *Librosa/librosa: 0.9.0*, version 0.9.0, Feb. 2022. DOI: 10.5281/zenodo.5996429. [Online]. Available: https://doi.org/10.5281/zenodo.5996429.