Speaker Recognition using Mel Spectrograms and Convolutional Neural Networks

Merwinraj Raja
Department of Computer Science & Engineering
University at Buffalo
Buffalo, NY 14260
merwinra@buffalo.edu
50336724

Kishore Mummala Narayana
Department of Computer Science & Engineering
University at Buffalo
Buffalo, NY 14260
kmummala@buffalo.edu
50379582

Abstract

With the advent of new and effective Deep Learning techniques, new methods and perspectives have risen in the field of audio recognition. Gaussian Mixture Models (GMM) were typically used for speech and speaker recognizing models in the past. It is observed that using Mel Spectrograms and Convolutional Neural Networks (CNN) were very effective in recognizing audio from different set of sources. Therefore, we propose a model for generating images from audio files which will be used in training and validating a particular individual's voice. We use a similar combination of CNNs and Spectrographic techniques to create a framework, which attempts to identify various speakers this time around.

1 Introduction

Speech is human vocal communication using language. Almost every language uses phonetic combinations of a limited set of perfectly articulated sounds that form the words. Therefore, it is undeniable that sound is one of the main senses that we use to understand the world around us. Even though it appears simple, the acts of speaking and listening have a lot of nuances and comprises of various hidden processes that are at play, to convey something to someone meaningfully. Yet our human brain is capable of doing all of that behind the scenes with virtually no difficulty, and at a very fast pace at that as well.

1.1 Recognizing and verifying speakers.

Technologies relating to speech have become ubiquitous today. This can be understood from the presence and the evolution of various intelligent personal assistants (IPA) devices/software in the market such as Apple's Siri, Microsoft's Cortana, Amazon's Alexa and the Google Assistant. Speaker identification is the process of determining the speaker's identity in an utterance from a given set of speakers. A test utterance has to be tested against all the speakers in the set. As for the task of verifying a speaker, the system needs to identify whether a given speech utterance belongs to the claimed speaker or not.

2 Objectives

The goal is to build a model that generates spectrogram images from the audio files of respective speakers, and by using Convolutional Neural Networks (CNNs), we attempt to recognize these speakers. Traditionally GMM have been used to gauge and rectify audio files. Here, we are using the prowess of CNN, which is typically used for analyzing visual imagery, to indirectly find the speakers. By creating spectrograms for each speaker, and feeding them to the CNN model, it will be able classify the images into a set of classes. These sets of classes represent each speaker in our case.

3 Project

We coded our model in python. We made use of libraries such as LibROSA which will be useful for analyzing audio files. Then, by generating the respective spectrograms for every speaker, we will train our model. Later, we test the performance of our model to gauge how efficient it is. This is how the project's model is going to be evaluated to see if this model is a feasible solution for the problem of speaker recognition.

3.1 Creation of Spectrograms

We have around 130 audio files for each speaker that we were able to find from the VoxCeleb dataset. Using the LibROSA library, we were able to create spectrograms for each of these audio files. Figure 1 shows this process:

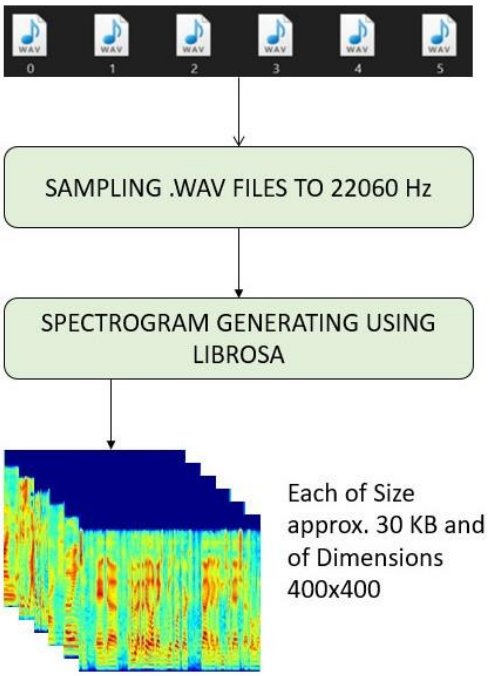

Figure 1: Generating spectrograms for all the audio files

One of the advantages is that the audio files have a fixed sampling rate which makes it easier to generate spectrogram through the LibROSA function. Converting to a standard sampling rate of 22,050Hz helps to improve consistency and reduce complexity in the code for training the model while also reducing the computation resources used by the project.

3.2 Design of our model

There is a total of 3 CNN layers followed by ReLU and max pooling layers. The CNN layers consist of 3 max pooling layers and 3 core layers (Flatten, Dense, and Dropout) as well.

- First Layer consists of: Convolution (3x3) + Pool (2x2) with 32 filters. The input size is 400x400 and output shape is 198, 198, 32.
- Second Layer consists of: Convolution (3x3) + Pool (2x2) with 32 filters. The output shape is 97, 97, 32
- Third Layer consists of: Convolution (3x3) + Pool (2x2) + Core Layers with 64 filters. The output shape is 46, 46, 64

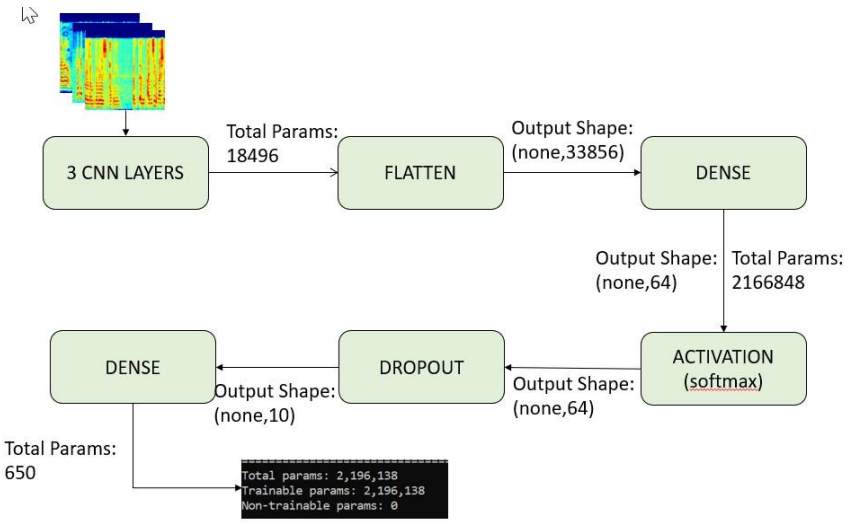

Figure 2: The design of our system

```
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 198, 198, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 198, 198, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 99, 99, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 97, 97, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 97, 97, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 46, 46, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                2166848
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0
=================================================================
Total params: 2,196,138
Trainable params: 2,196,138
Non-trainable params: 0
_________________________________________________________________
Found 1583 images belonging to 20 classes.
Found 538 images belonging to 20 classes.
Epoch 1/10
```

Figure 3: The configuration of the underlying CNN Layers. There are 3 layers of CNN and after each Convolutional Neural Network layer there is a ReLU function which is the activation function of the model.

3.3 Loading and Training data

Now that our spectrograms are generated, we load these images for training the model. The images are loaded as per our required parameters, such as batch size and image size, just as how we discussed previously in the design of the model. Each epoch will not have the same set of images as they are chosen randomly from the dataset during each epoch iterations. For every epoch the data is then sent to the neural network module for training. If the parameters such as precision, recall and accuracy of the model have improved at the end of the epoch then the model is saved at current point.

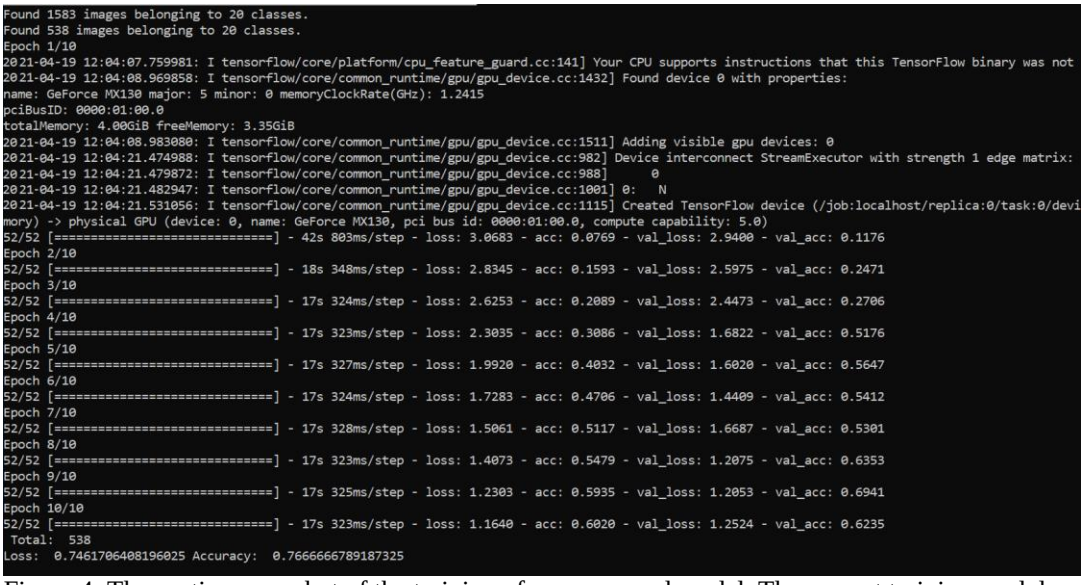

Figure 4: The runtime snapshot of the training of our proposed model. The current training module has 10 epochs.

3.4 Validation

Our project uses 90% of the spectrogram images for training and the remaining 10% of the images for validation. By using parameters such as, Confusion Matrix, F1score, Precision and Recall, the performance of the trained model is evaluated. Each batch is tested to see if the spectrogram features fall into the trained speaker module. The parameters are printed at the end of validation along with the accuracy and loss.

```
(env) C:\pro\speaker_recogntion_CNN>python performanceanalysis.py
[[ 3  0  0  0  0  3  0  0  5  0  3  0  0  2  0  0  0  0  0  0]
 [ 0 14  0  0  0  0  0  0  0  0 10  0  3  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  3  0  0  4  0  9  0 18  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  1  0  0  4  0  5  0  1 11  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  1  0  6  0  1  3  0  0  0  0  0  0]
 [ 0  0  0  0  0  2  0  0  0  0  1  0  0  1  0 21  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4 12  1  0  0]
 [ 0  0  0  0  0  2  0  0  0  0  0  0  5  0  0  1  0 14  0  0]
 [ 0  0  0  0  0  6  0  0  2  0  0  0  0  4  0  0  0  0  0  0]
 [ 0  0  0  0  0  3  0  0  1  1  4  0  0  2  0  0  0  0  0  0]
 [ 0  0 26  0  0  0  0  0  0  0  2  0  1  1  0  1  0  1  0  0]
 [ 0  1  0  2  0  1  0  0  1  0  6  0  3  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  2  0  1  0  0  2  0  1  0  0  0  0  4  0  0]
 [ 0  0  1  0  0 28  0  0  7  1  5  0  1  5  0  0  0  0  0  0]
 [ 0  0  0  0  0  3  1  0  1  1  3  0  0  2  0  0  0  0  0  0]
 [ 0  2  2  0  0  2  0  3  2  0  1  0  1  1  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 32  0  2  0  2  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  5  0  0  6  0  0  0  0  3  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0 36  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  3  0  0  5  0  2  0  0  2  0  0  0  1  0  0]]
C:\pro\env\lib\site-packages\sklearn\metrics\classification.py:11
es.
  'precision', 'predicted', average, warn_for)
Precision 0.1213327375022953
Recall 0.0686925343175343
C:\pro\env\lib\site-packages\sklearn\metrics\classification.py:11
.
  'precision', 'predicted', average, warn_for)
F1score 0.07106438215963155
```

Figure 5: The runtime snapshot of the "performanceanalysis.py", where the precision, recall, F1score and confusion matrix are calculated.

3.5 Inference

We now label the spectrogram images to corresponding speakers. The model we proposed produces results having accuracy of 82% with 3 CNN Layers. The neural network module that trains on the images can be tuned by changing the batch size and epoch values. This observation of 82% accuracy happened when batch size is 32 and epoch is 20. We have observed that typically increasing the batch size and the number of epochs, does increase the accuracy.

4 Observations and Results

As mentioned in the Inference previously, we did achieve an accuracy of 82%. We changed the batch size and the number of epochs to see how it affects the accuracy and the observations were recorded in a table. Please refer to Table 1.

20 SPEAKERS:CUSTOM	Batch Size	Epoch	Loss	Accuracy	Percision	Recall	F1			
1st iteration	16	5	1.640949912	0.521739141	0.082760033	0.048627646	0.043811275			No .wav: 2121
2nd iteration	16	10	1.091904809	0.671428586	0.080601343	0.068247354	0.06859052	Each Epoch takes 78secs on an average		Train no: 1688
3rd iteration	16	20	1.789673019	0.721428583	0.080538716	0.081134259	0.077976672			Test no: 433
										RUNNING ON VIRTUAL
1st iteration	32	5	1.717259792	0.470588244	0.038922588	0.060079365	0.043223546			ENVIRONMENT
2nd iteration	32	10	1.160873682	0.642857156	0.110161367	0.061779592	0.063653085			
3rd iteration	32	20	0.737835331	0.828571437	0.09952381	0.095902778	0.089858586			
VRAM ERROR	100	3								
	80	3						OVERHEAD		
	50	3								
1st iteration	5	5	1.435737557	0.591224032	0.121332738	0.068692534	0.071064382			
2nd iteration	5	10						Each Epoch takes 110secs on an average		
3rd iteration	5	20	1.836458789	0.739030034	0.036553125	0.054955484	0.042451278			
RESULT			0.737835331	0.828571437			0.089858586			

Table 1: These are the observations we made on various iterations that involves various epochs.

5 Future Work

For improving the model, we suggest:

- Enabling training with single audio file of a standard length.
- The accuracy can be improved by adding more Neural Network layers and the concepts of RNN and LSTM.

This particular work can be used as a voice biometric system to identify and catch criminals. Speech Recognition + Speaker Recognition would serve perfect for practical implementation in society.

6 References

[1] Boyang Zhang, Jared Leitner, Sam Thorton (2019) 'Audio Recognition using Mel Spectrograms and Convolution Neural Networks', Ph.D. dissertation, University of California, San Diego.

[2] Arjun. P. H (2005) 'Speaker Recognition in Indian Languages: A Feature Based Approach', Ph.D. dissertation, Indian Institute of Technology Kharagpur, India..

[3] Lukic. Y, C. Vogt, O. Dürr and T. Stadelmann (2016) 'Speaker identification and clustering using convolutional neural networks', IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1-6.

[4] Badshah. A.M, J. Ahmad, N. Rahim and S. W. Baik (2017) 'Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network, International Conference on Platform Technology and Service, pp. 1-5.