CS753 : Automatic Speech Recognition

A Course Project Report on

Text-Independent Speaker Verification Using 3D Convolutional Neural Networks

Submitted by

Royston Rodrigues (16307R004)
Kanhaiya Kumar (13D070046)

Guided by

Prof. Preethi Jyothi

Indian Institute of Technology Bombay

Contents

1 Task Definition

The Speaker verification problem refers to verifying the claimed identity of a speaker by using their voice characteristics. In this report we will discuss a method to solve the speaker verification problem using a 3D-CNN in a text independent setting. The method implement here is inspired from [2]. Any speaker verification system consists of three phases:

- **Development Phase:** In the development phase a background model is generated by classifying a large number of speakers at the utterance level. The aim here is to create a speaker representation that is distinctive enough. Here, we trained a 3-D CNN model to perform the classification task.
- **Enrollment Phase:** In the enrollment phase speaker specific model is developed for new speakers, this is generated with the help of the background model. The output of (N-1)th fully connected layer of the trained 3D-CNN is used as a fixed feature extractor to generate speaker specific model during the enrollment phase.
- **Evaluation Phase:** In the evaluation phase, the identity of an unknown person is verified by previously generated speaker models. During verification phase a test speaker's utterance is passed through the trained 3D-CNN model and its feature is extracted. Distance between the extracted feature with all the enrolled speaker's features is computed and the speaker with the least distance in the dictionary is identified as the test speaker.

2 Model Description and Implementation Details

2.1 Data Representation

In any ASR system, MFCC features are normally used as the data representation of the spoken utterances at the frame level. But MFCC's lose their local characteristics due to the last DCT operation for generating MFCCs. Since we are using Convolutional Neural Network, local characteristics of the signal are of importance. Hence we use log-energies, which is referred to as MFECs. The extraction of MFECs is exactly same as that of the MFCCs, just the MFCC's last DCT operation is discarded. The temporal features are overlapping 20ms

windows with the stride of 10ms, which are used for the generation of spectrum features. From a 0.8-second sound sample, 80 temporal feature sets are extracted (each forms a 40 dimensional MFEC feature vector).

In-order to capture different within-speaker utterances simultaneously we stack MFEC feature maps for several different utterances spoken by the same speaker this is used as the input to the CNN. This also helps us to get a text independent system. We use 20 sound samples to get a 20 x 80 x 40 3-D data representation that we use as a single example for training.

Inorder to extract the MFEC features we use the package python_speech_features which is available at `https://github.com/jameslyons/python_speech_features`. The function python_speech_features.logfbank() was used to compute the same.

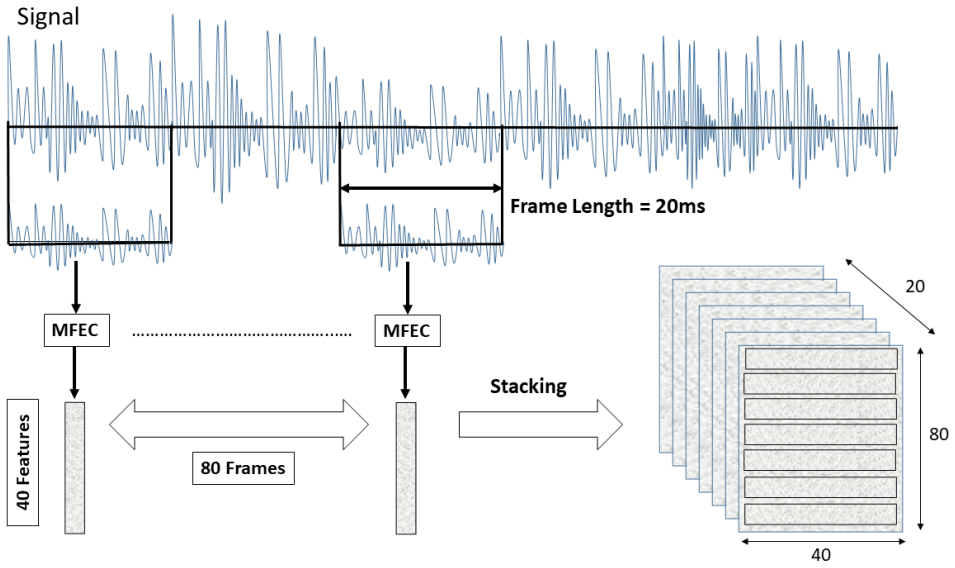

Figure 1: Data Representation

The reason to stack multiple MFEC feature vectors was motivated by the working of d-vector systems. In a d-vector system a single MFCC frame is used as input to a Deep neural network for classification of speakers during development stage. While generating a speaker specific model in the enrollment phase multiple MFCC frames correspond to multiple feature vectors these are averaged to get the final speaker specific feature. The d-vector system is a late fusion technique. Here, stacking of multiple MFEC features can be thought as an early fusion technique. This is advantageous because now the network can decide how to use multiple MFEC features, rather than using a standard

averaging kind of late fusion that is done in a d-vector system.

2.2 Architecture Details of Development Phase

In Table 1 we have mentioned the model architecture that we have used.

Layer Name	Input Size	Output Size	Kernel	Stride
Conv1-1	20 x 80 x 40 x 1	18 x 80 x 36 x 16	3 x 1 x 5	1 x 1 x 1
Conv1-2	18 x 80 x 36 x 16	16 x 36 x 36 x 16	3 x 9 x 1	1 x 2 x 1
Pool1	16 x 36 x 36 x 16	16 x 36 x 18 x 16	1 x 1 x 2	1 x 1 x 2
Conv2-1	16 x 36 x 18 x 16	14 x 36 x 15 x 32	3 x 1 x 4	1 x 1 x 1
Conv2-2	14 x 36 x 15 x 32	12 x 15 x 15 x 32	3 x 8 x 1	1 x 2 x 1
Pool2	12 x 15 x 15 x 32	12 x 15 x 7 x 32	1 x 1 x 2	1 x 1 x 2
Conv3-1	12 x 15 x 7 x 32	10 x 15 x 5 x 64	3 x 1 x 3	1 x 1 x 1
Conv3-2	10 x 15 x 5 x 64	8 x 9 x 5 x 64	3 x 7 x 1	1 x 1 x 1
FC4	8 x 9 x 5 x 64	64	-	-
FC5	64	200	-	-

Table 1: Architecture Details

- **Activation Function:** Here, we used a PRelu activation function after every layer except the last layer. The last layer was passed through a softmax fuction, in-order to interpret scores for each class as a probability distribution function. We used a PRelu instead of a standard Relu because we wanted to retain the negative part of the intermediate features captured at different layers of the network. PRelu has a learn-able parameter α that decides how much of amount of the negative portion of a particular layer output must be retained.

- **Loss Function:** The network was trained for classification on 200 speakers data. Standard cross entropy loss was used where the ground truth label was one hot encoded and compared with the softmax layer output of the above 3-D CNN architecture. Adam optimizer was used for this task.

- **Pooling:** Pooling was preformed only in the frequency domain. Temporal domain pooling was done very carefully, this is because speech being a temporal signal we were interested in capturing local temporal patterns by the convolutional structure of our model. We have used Max-pooling in the above architecture.

- **Dropout:** We did try training our model with a droupout rate of 0.1, 0.3 and 0.5 at the Conv-1, Conv-2 and Conv-3 layers respectively. This was done to remove over fitting. However our model normally takes 6 hours to reach a train accuracy of 80 percent. It took us almost 8 hours to just touch a train accuracy of 40 percent. The training is extremely slow in the presence of dropout. We decided to settle for L_2 weight decay with hyper parameter decay rate as 0.0001.

- **Batch Norm:** Every layer shown in Table 1 was passed through a batch norm layer. Batch norm helps models to converge faster. Without using batchnorm we had noticed that our model was stuck and there was no sign of learning. Batchnorm is really important and makes a very significant difference, as beginners in training large networks we realized many papers tend to ignore mentioning this detail and its assumed that everyone in the community knows how to use it and consider it redundant.

- **Batch Size:** We had access to Nvidia 1080 Ti GPU which has 11 GB of memory. We wanted to increase our batch size to as much as possible to fit it in the GPU memory. We used a batch size of 128.

2.3 Enrollment Phase

For the enrollment of a new speaker, first speech data is converted to the required 3-D data cube format of dimensions 20 x 80 x 40 this is explained in section 2.1. This example is forward passed through the above 3-D CNN network and the output of FC4 layer is treated as the current speaker's specific model. With this FC4 feature a dictionary is created where all enrolled speaker's features are stored. One implementation detail we want to mention is that batchnorm and dropout layers behave differently during the forward pass of training time and during the forward pass of test time, we had to ensure that the network was working in test mode and the train flag was set to false, else the fixed feature extractor from FC4 would not have been the correct one.

2.4 Evaluation Phase

During the evaluation phase we take the data of the test speaker in the 3-D cube format and generate the FC4 feature. This feature is compared with each and every speaker that is enrolled in a euclidean distance fashion. The speaker with least distance is the identified speaker.

There are two challenges we faced using the method during the evaluation phase :

- **Multiple enrolled speakers having almost equal distances verification phase:** This was an indication of the model not being distinctive enough and our system being confused. For this, we considered to retrain the CNN with more speaker's data in order to create a more distinctive model. Also, a held-out development set was kept aside to keep checking for over fitting during training. Our model did over fit and the solution to that was reducing the network parameters FC4 layer was initially 128 but was brought down to 64, also initially, we started off with four depths of convolutional layers, this was reduced to three.

- **Case for unenrolled speaker:** A threshold for the system was developed. This threshold was decided by monitoring empirically distances enrolled speakers exhibit during verification phase. All distances above this threshold were ignored and not be considered as a valid speaker from the distance. This enabled us to calculate the EER (Equal Error Rate Metric).

3 Experimental Setup

3.1 Dataset

The VoxCeleb dataset [1] contains over 100,000 utterances for 1,251 celebrities, extracted from videos uploaded to YouTube. The size of this dataset is 160 GB it took us 3 days to download the entire data set, and it was left to download in parallel on four machines. Once this was done, for each speaker based on duration of utterances available we created multiple samples of the 3-D cube as shown in section 2.1.

Now we wanted to split the dataset into train, test and validation. We did not use the standard splitting given by the oxford group as their test speakers where just 40, also there was no guarantee that we would have enough uttrances to enroll the speaker into the system.

In our initial split setting, we used 15,000 samples of 3-D cubes to classify 200 distinct speakers. This was done to enshure that each sample had atleast greater than 50 cubes for training. Later we realized that this was not enough and our model was overfitting, So we then moved on to have 24,000 samples of 3-D cubes to classify 200 distinct speakers, where it was guaranteed that each sample had at-least 100 cubes for training. Based on this criterion we created

a subset of the dataset of 200 speakers and split it into train and validation set.

For testing we used 32 speakers data such that there is enough samples to enroll them, also another set of 160 speakers data was used for evaluation phase to consider case of an unenrolled speaker, trying to get himself verified.

3.2 GPU

We had access to Electrical Engineering deparment's Vision and Image processing lab, this had enabled us to use the Nvidia 1080 Ti GPU, which has 11GB of GPU memory. Our Model during training consumed 8 GB of GPU memory.

4 Results

Case 1: EER (Equal Error Rate) and SER (Speaker Error Rate) for enrolled speakers from training set: Here, we plot the ROC curve in Figure 2 when all the enrolled speakers into the system are just from the training set. In Table 2, EER and SER are reported for this case.

Error	Percentage
EER	16.875 %
SER	6.09375 %

Table 2: Error rate for users from training dataset

Case 2: EER (Equal Error Rate) and SER (Speaker Error Rate) for enrolled speakers from unseen test set: Here, we plot the ROC curve in Figure 3 when all the enrolled speakers into the system are just from the test set. In Table 3, EER and SER are reported for this case.

Error	Percentage
EER	27.8125 %
SER	22.5 %

Table 3: Error rate for users only from test dataset

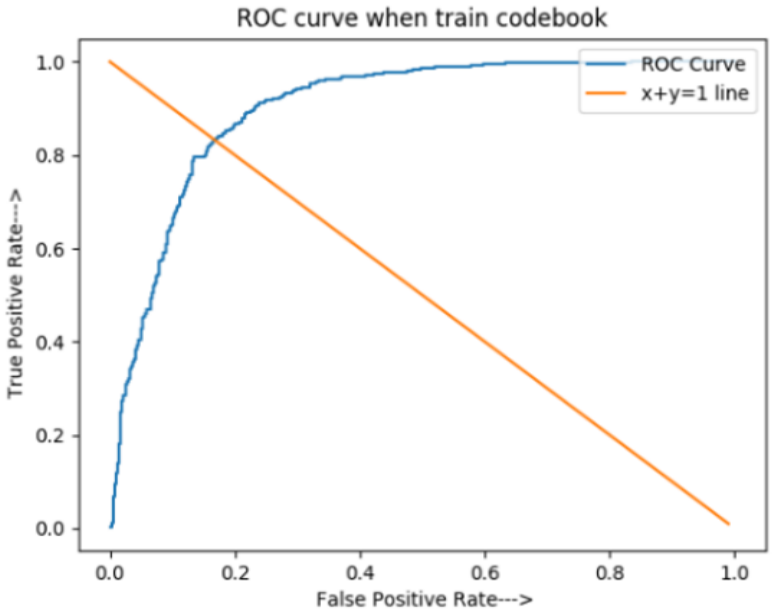

Figure 2: ROC curve for users from training dataset

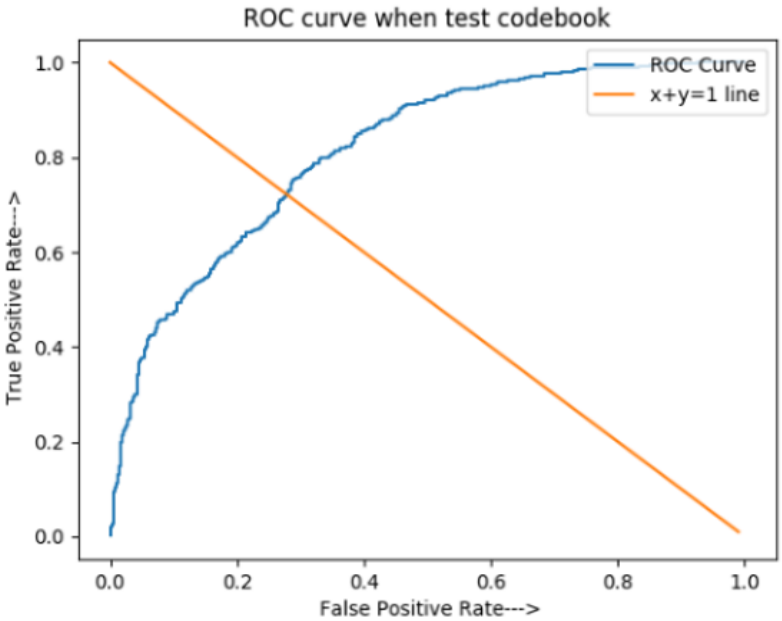

Figure 3: ROC curve for users only from test dataset

Case 3: Analysis of amount of Data Required for enrollment: The development phase model over suffers from over fitting the accuracies of train and validation for classification of 200 speakers is reported in Table 4, to still effectively enroll speakers and create a meaningful codebook we considered keeping many FC4 features generated from each enrolled speaker's utterances. So during enrollment many voice samples are taken and 3-D cubes are generated, all the 3-D cubes are used to get different FC4 features, these FC4 features are recorded in the codebook. The plot in Figure 4 shows SER rate vs Number of 3-D cubes each of 20 x 80 x 40 (16 seconds of speech) used for enrollment. We fixed 80 3-D cubes for all the results shown above.

Data	Accuracy
Train	92.491 %
Validation	68.213 %

Table 4: Development phase training results

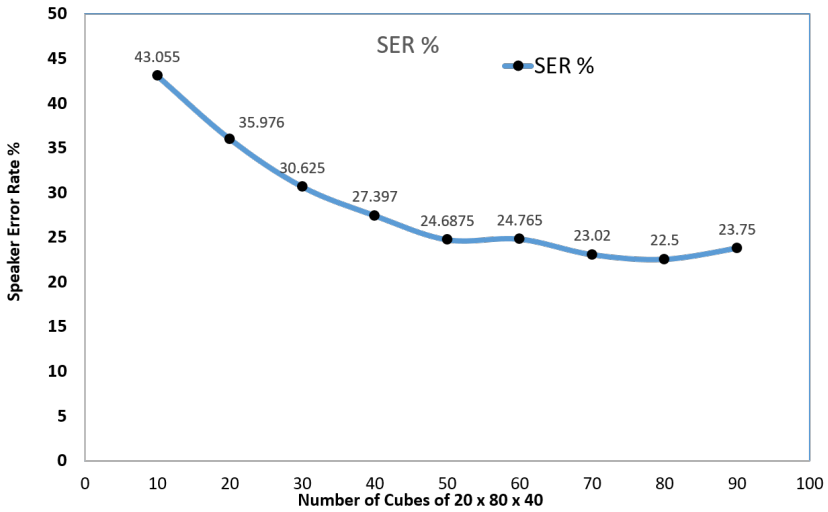

Figure 4: Speaker Error Rate as a function of Number of utterances

5 Summary

We have created a text independent speaker verification using 3D CNN. Various challenges were faced and their proposed solution are indicated in the report. Some of the key decisions we had to make were regarding the following:

- **Finalizing the Network architecture:** This involved including batch norm, considering time required for training the network with dropout component and reducing network size based on availability of data. Considering weight decay as regularisation for the network.
- **Splitting of VoxCeleb dataset into train, test and validate sets:** This was done based on how many utterances we had per speaker and if we could get significant number of 3-D cubes for training.
- **Continuing with enrollment phase, despite overfitting in development phase:** Here it was decided to take multiple 3-D cubes of utterances even during enrollment inorder to gererate speaker specific model.
- **Selection of threshold for unenrolled speaker rejection :** If we had not include this component in our project then an un-enrolled speaker would have been mapped to some person closest in the enrollment code book.

References

[1] A. Nagrani, J. S. Chung, A. Zisserman. "VoxCeleb: a large-scale speaker identification dataset " In *INTERSPEECH*, 2017.

[2] Amirsina Torfi, Nasser M. Nasrabadi, Jeremy Dawson. *Text-Independent Speaker Verification Using 3D Convolutional Neural Networks*
`https://arxiv.org/abs/1705.09422`