Speaker Recognition

Pranta Deb Emon
ID 1731202642
deb.emon@northsouth.edu

Noshin Nirvana Prachi
ID 1610394042
noshin.nirvana@northsouth.edu

Mohammad Tashfiq H. Choudhury
ID 1611508042
mohammad.choudhury@northsouth.edu

Faisal Mahmud Nahiyan
ID 1712699042
Faisal.nahiyan@northsouth.edu

***Abstract*—Speaker Recognition has become a vastly popular and useful research subject which has spawned countless essential applications in security, replication, authentication, automation to name a few. The primary goal had been to create improved robust techniques to identify audio and to improve accuracy to human levels of comprehension. Many techniques have been implemented using deep learning and neural network concepts and various datasets. Our work presented in this paper makes use of Timit and LibriSpeech datasets. We have used a closed set and an open set implementation procedure respectively on both the aforementioned datasets. The closed set implementation uses a common machine learning convention of utilizing the same sets of data for training and testing leading to a higher accuracy. On the other hand, the open set implementation makes use of one dataset to train and another to test on each occasion. The accuracy in this case turned out to be relatively lower. On each dataset, CNN and LSTM have been used to identify sound and this led to the observation that implementing CNN resulted in a larger accuracy. Our work focusses on using MFCC to transform audio to spectrograms without losing the features that is essential to the audio file in question. This has evidently worked well on both datasets using an original model and this has translated to a higher accuracy.**

***Index Terms*—deep learning, speaker recognition, speaker verification, speaker identification**

I. Introduction

Sound has been a key indication of identification from the very start of the human kind. Humans learnt to differentiate between different sounds to distinguish danger, weather, enemy and foe. And thus humans have grown an inbuilt capability to identify the source of the sound based on the characteristics of the sounds. We can identify the being behind the voices by this inbuilt ability given by evolution. Classification of sounds to recognise speakers has been a major field of research in recent times, and people have been trying to use more and more techniques to recognise speakers from the sounds which has been proven quite accurate. Speaker recognition can be used in the field of bioinformatics,scam-identification,copyright verification and search and rescue. To classify the speaker based on the voice, the audio signal has to be pre processed, the feature has to be extracted and the final classification of speaker. The audio samples are pre- processed into several segments in order to extract essential features,Mel-frequency Cepstral Coefficient (MFCC) , spectral flux,chroma vector, poly features etc are among the most sorted out features for audio classification. We have used MFCC in our project to classify speaker. We worked on the comparison between two neural network models, Convolutional Neural network (CNN) and Long Short-term Memory(LSTM) based on the MFCC feature to find which works better in speaker recognition. The idea of MFCC is to convert time-domain signals into frequency domain signals and use Mel filters to mimic cochlea that has more filters at low frequency and fewer filters at high frequency. Thus it is safe to conclude that the feature MFCC and its characteristics are focused on the audibility of the human hearing system, that can accommodate the dynamic nature of true-life sounds with the way that they are treated with feature vectors for classification The datatset we are using for our purpose of speaker recognition are TIMIT an LibriSpeech datasets, and comparing between the classification done by CNN and LSTM using the same features and the same dataset, to see which model gives a better accuracy result in recognising the speaker from the given dataset. LibreSpeech dataset was highly optimised and requires less pre processing than TIMIT dataset, the methodology and techniques used are described in the later part of the paper.

II. Related Works

Several works have been presented on how to recognize speakers irrespective of the spoken phrase by using LSTM and also there have been several works recently explored the use of low-level speech representations to process audio and speech with CNNs. The main steps in recognition systems are feature extraction and classification. There are many techniques for extracting distinctive features such as MFCCs, Linear Prediction Coefficients and Linear Prediction Cepstral Coefficients. There are some necessary techniques of robust feature extraction such as feature normalization, model-domain compensation and score normalization [14]. Ricardo Sant'Ana et al. [15] proposed a text-independent speaker recognition system based on new features called Hurst features and a new classifier that depends on the concept of the multi-dimensional fractional Brownian motion. The performance of this system was compared to those of the Gaussian mixture models, autoregressive vector and Bhattacharyya distance classifiers [15]. H. Gish and M. Schmidt [16] introduced some work in the field of speaker identification. And though spectrograms retain more information than standard hand-crafted features, their design still requires careful tuning of

some crucial hyper-parameters, such as the duration, overlap, and typology of the frame window, as well as the number of frequency bins. For this reason, a more recent trend is to directly learn from raw waveforms, thus completely avoiding any feature extraction step. This approach has shown good works in emotion tasks[17], speaker recognition[18], spoofing detection[19].Another idea related to the proposed method has been recently explored in [20], where a set of parameterized Gaussian filters are employed. This approach operates on the spectrogram domain, while SincNet directly considers raw time domain waveform. To the best of our knowledge, this study is the first to show the effectiveness of the proposed sinc filters for time-domain audio processing from raw waveforms using convolutional neural networks.

III. Methodology

A. Dataset

The dataset or corpus used for training the models plays a vital role since performance measures cannot be compared properly if the testing circumstances vary too much. There are a number of

datasets when it comes to speaker recognition task such as VoxCeleb, CSS10, Ted-Lium, CHiME, VCTK, WSJ etc. Limited number of speakers could provide vast speech data in some cases while small and relevant speech data could be obtained from a large number of speakers. Considering that fact, this study worked with two popular SR-related datasets Timit and LibriSpeech.

1) Timit: The Timit corpus consists of 630 different speakers/classes, with 10 samples collected from each class. The speakers belong to 8 major dialect regions of the USA. The Train set has 462 speakers while the Test set contains the samples from the other 168 speakers. When we first took a closed set-like approach to the problem, we only considered the Train set, making an 80-20 split for training and testing on both datasets that we used. However, the speakers appearing in the Test set are entirely different from the Train set, making it an open set problem. Our open set approach maintained the train-test arrangement found in the original setup, with 4620 audio files in the Train folder and 1680 audio files in the Test folder.

2) LibriSpeech: The LibriSpeech ASR corpus (train-clean-100) contains 28539 audio samples from 251 speakers, which we have used for the closed set implementation. The open set implementation, however, worked with a different setup of the dataset which followed the same distribution as [5]. The training and test materials were randomly sampled; 12–15 s of material was used for training, and 2–6 s in testing. The total number of samples was 21,933, where 14,481 were used as training data and the rest for testing, with the starting and end silence removed.

B. Audio Exploratoty

The audio samples are in .wav format where the continuous waves of sounds are digitized. Each file is sampled at discrete interim and transformed into a one-dimensional numpy array of digital values (Fig.). The waves shown are one-directional and, at any given time instance, can represent specific amplitude or frequency.

Using the sampling rate, the sample values can be assembled to reconstruct the audio if required [6]. The Nyquist-Shannon theorem [7] suggests a sampling rate where the minimum sampling frequency of a signal should be double the frequency of its highest frequency component in order to not distort its underlying information. In our case, we chose the libROSA python package which normalizes the data such that the array contains values between -1 and 1. This package uses a default sample rate of 22050 Hz, reducing the array size and decreasing the training time. When the audio is loaded using librosa.load() function and the mono parameter is set to true, it creates a mono audio signal by combining the two channels of the stereo audio signal to create a one dimensional numpy array. The default frame length and hop lengths are 2048 and 512 samples, respectively [8]. Following the array representation of the audio, various types of features are extracted from it.

C. Feature Extracted (MFCC)

For the classification problem, extracting spectral features from the audio requires the audio to be converted into frequency domain from time domain, using Fourier transformation. A number of spectral features can be used in this regard such as Mel Frequency Cepstral Coefficients (MFCC), Mel Spectogram, Spectral Centroid, Spectral Rolloff. We chose MFCC because it has produced better results than the likes of Chroma Chroma CENS, STFT, Chroma CQT, Spectral Contrast, Tonnetz etc in providing distinguishable information and

representations which are effective and ensures better accuracy in audio-based classification tasks. Moreover, it is used as a primary features in many research works which deal with audio signals [9]. MFCCs closely mimic how humans perceive audio signals. The libROSA library came in handy in the feature extraction task as well since it has built-in functions [9] to generate the required spectogram. The librosa.feature.mfcc() function requires passing a few parameters such as the loaded audio and the sample rate of the audio (default sample rate of 22050 Hz) in order to generate MFCCs. The function returns 40 MFCCs over 173 frames by default unless the number of MFCCs to return has also been passed through the function. We took the mean of the 173 frames returned, which converts the two-dimensional array into one-dimensional array with the mean values of the frames.

IV. Model Architecture

A. CNN Model

A CNN is typically composed of various kind of layers, the combination of which formulates the overall architecture. Training a CNN involves making different kind of decisions in terms of the architectural patterns as well as the hyperparameters. Deciding the input data format, the number of convolutional and pooling layers, and the filter dimension brings

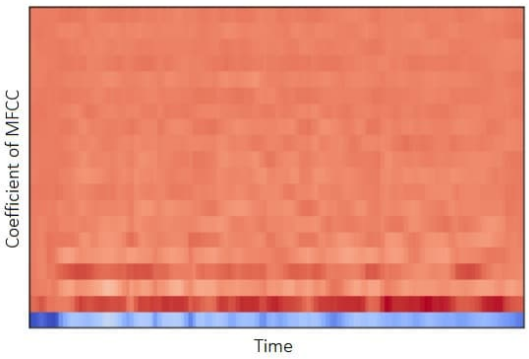

Fig. 1. MFCC

forth deciding the learning rate, the dropout [9] probability, the number of epochs, and the batch size etc [10]. The model that we created is a sequential model comprising two Conv2d layers followed by a batch normalization layer and then three dense layers, where the final layer is the output layer.

1) Convolutional Layers: The first Conv2D layer consists of 64 filters, receiving input shapes of 20x5x1. The kernel size set to 5 results in a 5x5 filter matrix and the stride set to 1 makes the filter converge around the input volume by moving one unit at a time. With the padding set to 'same' for the operation, the output produces an output with same height and width as the input. The activation function used in this layer is ReLU which is computationally more efficient than sigmoid units [11]. The second Conv2D layer also uses the same activation function and has the same configuration for the kernel size, strides and padding as the first Conv2D layer. However, it consists of 128 filters and the input shape is 20x5x64. It also has a dropout of 30A MaxPooling2D layer is used after each of the two Conv2D layers. The MaxPooling2D layer reduces the dimension of the input shape.

2) Batch Normalization: Since the parameters of the previous layers change during the training process of deep neural network, lower learning rates and careful parameter initialization saturates the model with nonlinearities, increasing the training duration. Batch normalization accelerates the training process by reducing this issue also known as the internal covariate shift [12]. It allows us to be less cautious about initialization and allows higher learning rates.

3) Flatten Layer: The flatten layer uses global average pooling to convert the output of the convolutional layers into a one-dimensional array which will then be inputted into the next hidden layers.

4) Dense Layers: Further processing of the model has been carried out with two dense layers and an output layer. The two dense layers have 256 and 512 nodes, respectively. The nonlinear function used in the first two dense layers is ReLU as it reduces the time required for gradient descent by switching all the negative activations to zero.

5) Output Layer: In our closed set implementation, the output layer had softmax activation function consisting as many nodes as the number of classes in the training split

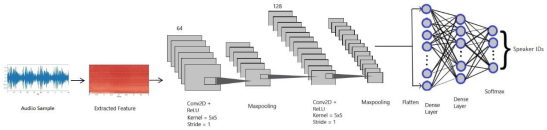

Fig. 2. CNN

in both the datasets. Softmax turns logits into probabilities and predicts the choice with the highest probability. We built an intermediate model for the open set implementation by popping out the last softmax layer after the initial model has learned the features and optimized weights after several epochs. Then, we extract the MFCC spectral features from the corresponding Test set of the Train set that was used to train the model.

Finally, the intermediate model is used to ouput an embedding vector for a test sample from each of the speaker and it is used to match the other test samples of the same speaker using cosine similarity, euclidian distance, and manhattan distance. Then the labels are matched to find out the accuracy of the model.

B. LSTM Model

LSTM's recent popularity in audio-based classification and an impressive accuracy rate in classification process inspired us to try it in the speaker recognition task as well. This too is a sequential model that used two LSTM layers, two Time Distributed layers, followed by a flatten layer and another dense layer to finish with.

1) LSTM Layers: Both of the LSTM layers consist of 128 hidden units. The first layer takes input of shape 20x5 where 20 represents the time steps which acts as a indicator to the LSTM layer as to how many times it should repeat itself after being applied to the input. A dropout of 0.3 was used in the second input to reduce overfitting.

2) Batch Normalization: This model also uses batch normalization after the first two layers. The training time was effectively reduced by using batch normalization opposed to the training time without using it.

3) Time Distributed Layers: Among the two time distributed dense layers in the model the first layer takes an input size of 128 and produces 256 nodes as output which then act as the input shape for the second Time Distributed layer. The output for the second Time Distributed layer contains 512 nodes. In both layers, ReLU has was used as the activation function.

4) Flatten Layer: The flatten layer flattens the 3D output from the second Time Distributed layer and passes the long vector of input data onto the dense layer which is the last layer.

5) Dense Layer: Like the CNN model, the output dense layer used a softmax activation function for the closed set implementation. It converted the inputs into a discrete probability distribution. The open set implementation again used an intermediate model by popping out the last softmax layer after a few training epochs. Following the feature extraction from

the Test sets, the intermediate model produced an embedding vector for each test class which is then matched with other instances of that particular test classusing cosine similarity, Manhattan distance, and euclidean distance.

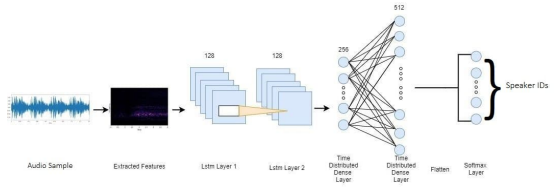

Fig. 3. LSTM

C. Model Summary of CNN and LSTM

For training our CNN and LSTM model on closed sets, we started with 500 epochs on the Timit dataset. In the open set implementation we used 300 epochs for the CNN model and 100 epochs for the LSTM model on the same dataset. On the LibriSpeech dataset, we chose to use epochs for the closed set implementation. For the open set implementation we used epochs. The Fig.1 and Fig.2 displays the training and testing

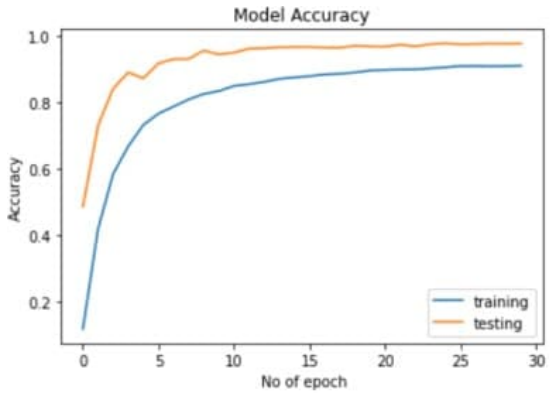

Fig. 4. LibriSpeeh Accuracy

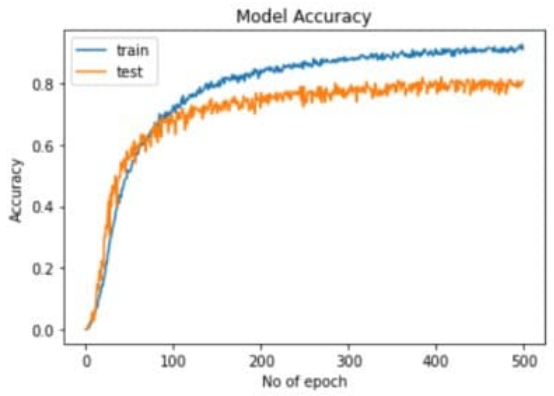

Fig. 5. Timit accuracy

accuracy for CNN and LSTM on the Timit and LibriSpeech dataset, respectively, in the Y-axis and number of epochs in the X-axis. It is observable that with the increase in the number of epochs the accuracy of the model increases for both the training and testing data. We split the dataset into a batch size of 50 for both models and both datasets.

D. Model Compilation of CNN and LSTM

1) Loss Function: The loss function we have used in this case is categorical cross-entropy loss. This loss function is for multi-class classification tasks and it performs better than MSE and RMSE since it penalizes incorrect more heavily than those. The equation of cross entropy loss is:

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j)$$

Fig. 6. Loss Function

2) Metric: The metric that we have used to measure the performance of this metric is accuracy. It is the ratio between the number of correct predictions and the number of total predictions made by our model. This value, when multiplied by 100, gives us the percentage of model's success in recognizing which speaker a given audio sample belongs to.

$$Accuracy = No.ofCorrectPredictions/No.ofTotalPredications \quad (1)$$

3) Optimizer: Optimizers are used to reduce the losses by fine-tuning the attributes of a neural network like weights and learning rates. We used the Adam optimizer which uses the adaptive learning rate for each parameter. It combines Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) which allows it to handle sparse gradient on noisy problems.

V. Results

A. Comparison between CNN & LSTM

CNN is a very powerful technique for image-based classifications. Its design allows it to exploit "spatial correlation" in data images and speeches. On the other hand, LSTM performs better in cases involving time-series data or sequential data. When it comes to audio-classification, LSTM outperforms CNN model in most cases. Testing accuracy can increase by up to 4since we did not opt for data augmentation. As shown in Table.2, the CNN model achieved 77.51closed set implementations, respectively. The training accuracy was 100and closed set implementations, respectively, using the same model. The LSTM model, however, performed poorly, producing 62.13set implementations, respectively. We achieved 99.26the closed set implementation on the LibriSpeech dataset. The same model on the customized LibriSpeech dataset produced 64.7accuracy for training and an 84.96

VI. Conclusion

Although we have not been able to reach a state-of-the-art solution in this regard, we are proposing a completely different model for speaker recognition task which requires relatively less number of parameters, making the training process fairly shorter. Using the same architecture for two diverse datasets resulted in a better result

References

[1] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[2] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

[3] Disc, N. S., Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., & Dahlgren, N. L. Acoustic-Phonetic Continuous Speech Corpus.

[4] Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015, April). Librispeech: an asr corpus based on public domain audio books. In 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 5206-5210). IEEE.

[5] Ravanelli, M., & Bengio, Y. (2018, December). Speaker recognition from raw waveform with sincnet. In 2018 IEEE Spoken Language Technology Workshop (SLT) (pp. 1021-1028). IEEE.

[6] Ridley, M., & MacQueen, D. (2004). Sampling plan optimization: A data review and sampling frequency evaluation process. Bioremediation journal, 8(3-4), 167-175.

[7] Song, Z., Liu, B., Pang, Y., Hou, C., & Li, X. (2012). An improved Nyquist–Shannon irregular sampling theorem from local averages. IEEE transactions on information theory, 58(9), 6093-6100.

[8] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference (Vol. 8, pp. 18-25).

[9] Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE transactions on acoustics, speech, and signal processing, 28(4), 357-366.

[10] Piczak, K. J. (2015, September). Environmental sound classification with convolutional neural networks. In 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). IEEE.

[11] Nair, V., & Hinton, G. E. (2010, January). Rectified linear units improve restricted boltzmann machines. In Icml.

[12] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR

[13] Baldi, P., & Sadowski, P. J. (2013). Understanding dropout. Advances in neural information processing systems, 26, 2814-2822.

[14] Kinnunen T, Li H (2010) An overview of text-independent speaker recognition: from features to supervectors. Speech Comm 52:12–40

[15] Sant'Ana R et al (2006) Text-independent speaker recognition based on the Hurst parameter and the multidimensional fractional Brownian motion model. IEEE Trans Audio Speech Lang Process 14(3): 931–940

[16] Gish H, Schmidt M (1994) Text-independent speaker identification. IEEE Signal Process Mag 11:18–32

[17] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in Proc. of ICASSP, 2016, pp. 5200–5204.

[18] H. Muckenhirn, M. Magimai-Doss, and S. Marcel, "Towards directly modeling raw speech signal for speaker verification using CNNs," in Proc. of ICASSP, 2018.

[19] H. Dinkel, N. Chen, Y. Qian, and K. Yu, "End-toend spoofing detection with raw waveform CLDNNS," Proc. of ICASSP, pp. 4860–4864, 2017.

[20] H. Seki, K. Yamamoto, and S. Nakagawa, "A deep neural network integrated with filterbank learning for speech recognition," in Proc. of ICASSP, 2017, pp. 5480–5484.