Preprocessing Techniques for Speaker Recognition

Mentor

Dr. Ramesh K Bhukya,

Assistant Professor,
Room No: 5158 (CC-III),
Department of Electronics and Communication Engineering,
Indian Institute of Information Technology Allahabad,
Deoghat, Jhalwa, Prayagraj-211015, Uttar Pradesh, India.
Email: rkbhukya@iiita.ac.in

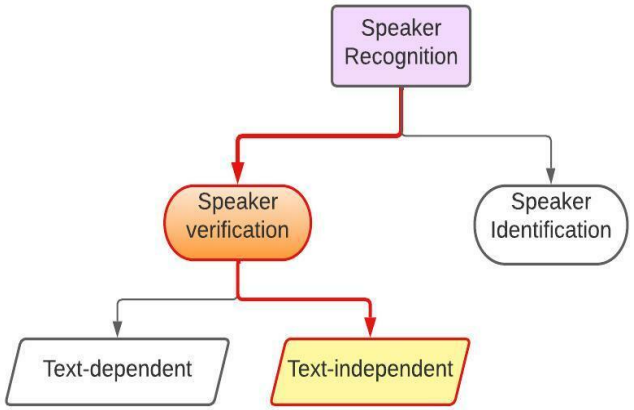
Presenter:
Ankit Singh
IEC2018076
Email: iec2018076@iiita.ac.in

Preprocessing Techniques for Speaker Recognition

Objective

- Our focus is on Exploring Preprocessing Techniques for Speaker Recognition and Implementing a Speaker recognition model that is able to recognize the speaker based on the speaker-specific information included in the speech waves

Introduction



- Speaker Recognition is the process of automatically recognizing who is speaking by using the speaker-specific information included in speech waves
- It is divided into two : Speaker Verification and Speaker Identification
- This thesis consists of exploring a text-independent speaker verification system and Implementing it
- It is used to verify identities being claimed by people accessing systems
- Preprocessing steps are Explored before feeding the data to the neural network model
- VAD is implemented to detect the absence/presence of speech in an audio signal.If a non-speech part is detected then it will be rejected.

- MFCCs Features and all the information like mapping and label are stored
- Saving the pre-processed data so that processing time will be saved while training
- Processed data is splitted for training and evaluation
- Then Implemented the Model
- Multiple layers are added to model to reduce computation time

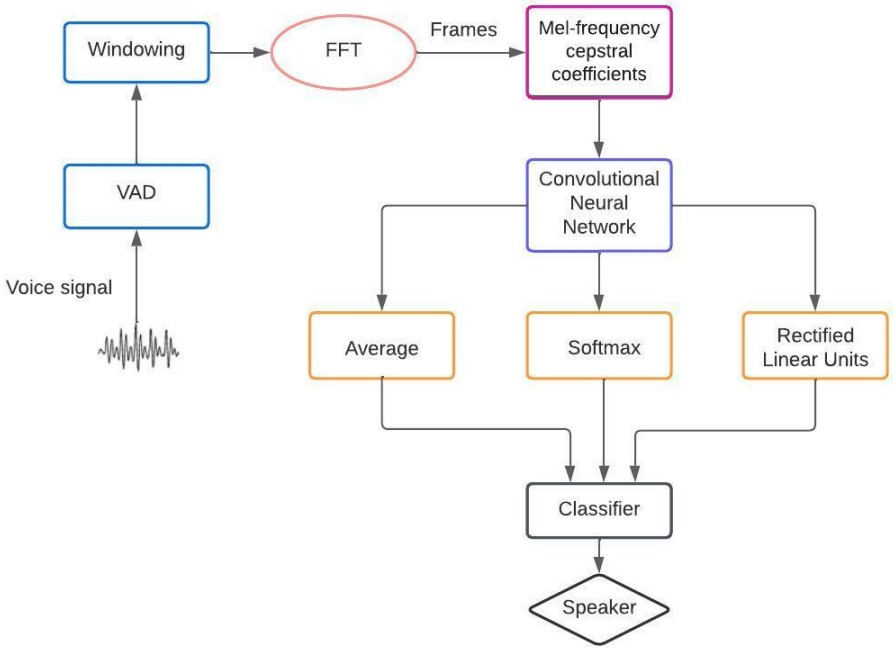


Figure : Flowchart of Speaker Recognition

Motivation and Needs:

- Success of deep learning for speech recognition
- Improve voice based authentication
- Vad can be used as part of systems for Speaker Recognition

Block Diagram of Speaker Recognition System

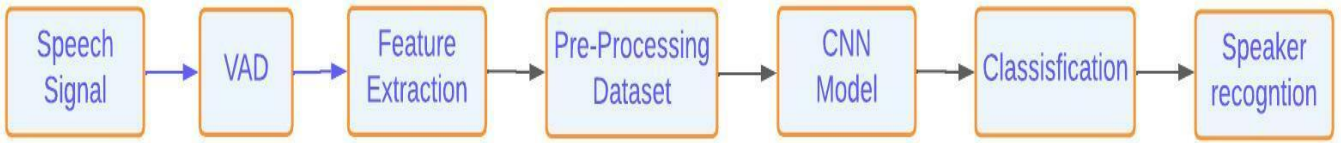


Figure: Block Diagram of Speaker Recognition System

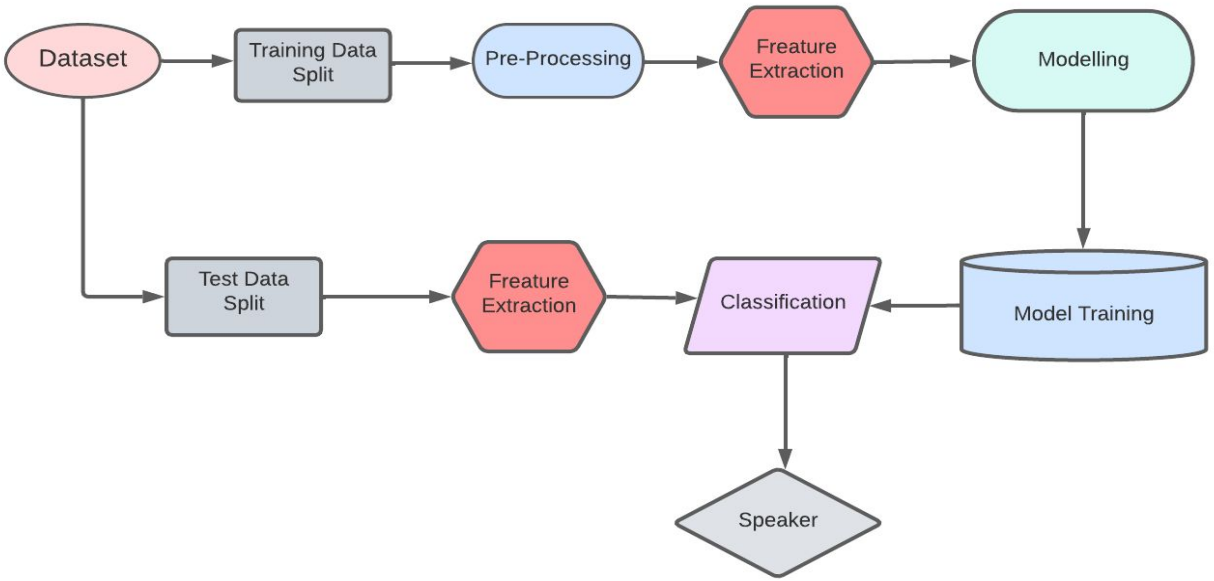


Figure : Approach Diagram

Preprocessing : VAD

- The method consists of two passes of denoising followed by a voice activity detection (VAD) stage.
- In the first pass, high-energy segments in a speech signal are detected by using a posteriori signal-to-noise ratio (SNR) weighted energy difference
- If no pitch is detected within a segment,the segment is considered as a high-energy noise segment and set to zero.
- In the second pass, the speech signal is denoised by a speech enhancement method
- Next, neighbouring frames with pitch are grouped together to form pitch segments,
- In the end,extended pitch segments of the denoised speech signal are used for detecting

Why:

- Reduces CPU
- memory, and
- battery consumption

Issue:

There are some issues that remain unsolved:

- Loud noise is classified as speech
- Soft speech is classified as noise voice activity

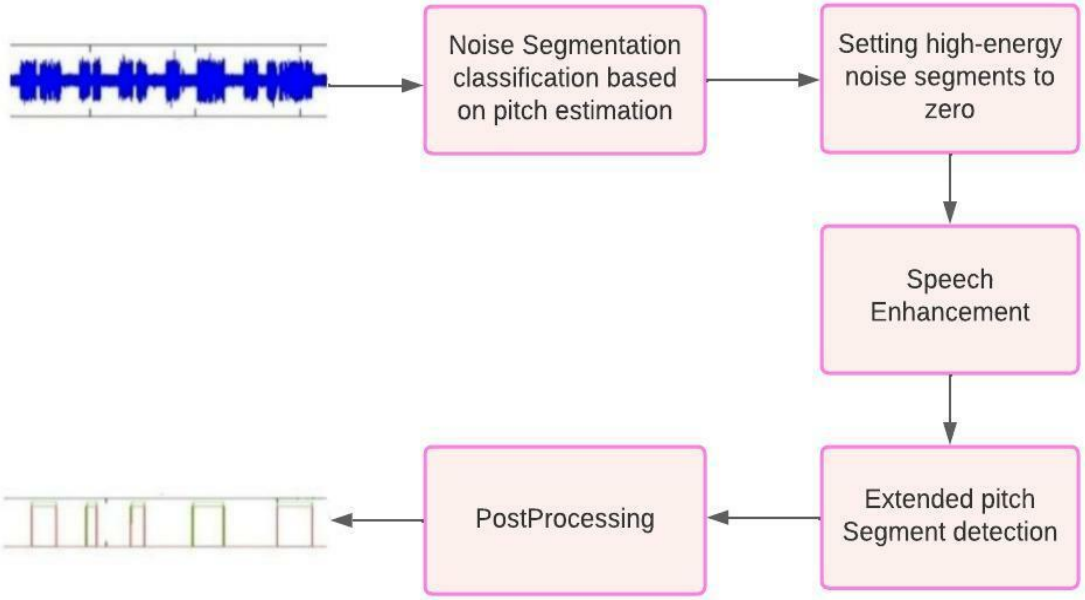


Figure : Block Diagram VAD

Feature Extraction: MFCCs

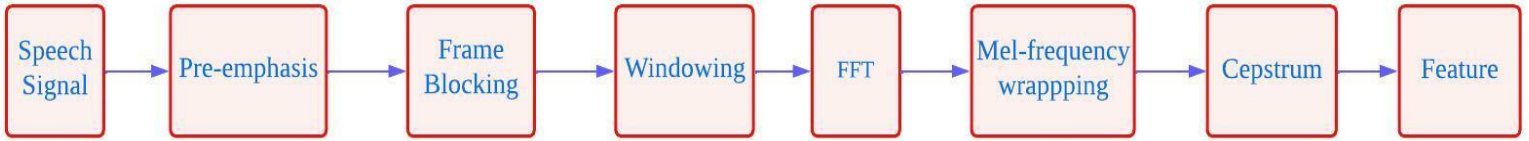


Figure 5: Computation of MFCCs

- The main Purpose of MFCC processor is to mimic the behavior of the human ears.
- MFCCs less susceptible to variations.
- MFCCs are commonly derived as follows:

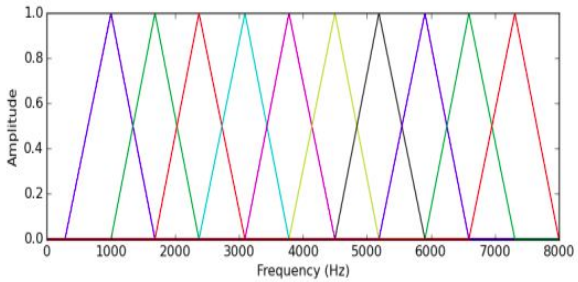
Step 1: Compute the Fourier transform of a signal (a windowed snippet).

Step 2: Using triangular overlapping windows, map the power of the spectrum produced on top of onto the mel scale.

Step 3: Record the power logs for each of the Mel frequencies.

Step 4:Use Mel log powers as signal, compute the discrete cosine transform..

Step 5: The MFCCs are the resulting spectrum's amplitudes. There could be variations on this method, such as changes to the form or spacing of the windows used to map the size, or the addition of dyna capabilities such as "delta" and "delta-delta" (first- and second-order frame-to-body distinction) factors.



Feature Extraction: MFCCs

- First 13 MFCCs coefficient are used
- These contains most info like formants, Identification of sound, phonemes and timbres
- Used Delta & double delta MFCCs
- These are 1st & 2nd order derivatives of MFCCs Coeff.
- Adding up, Total coeff. == 39

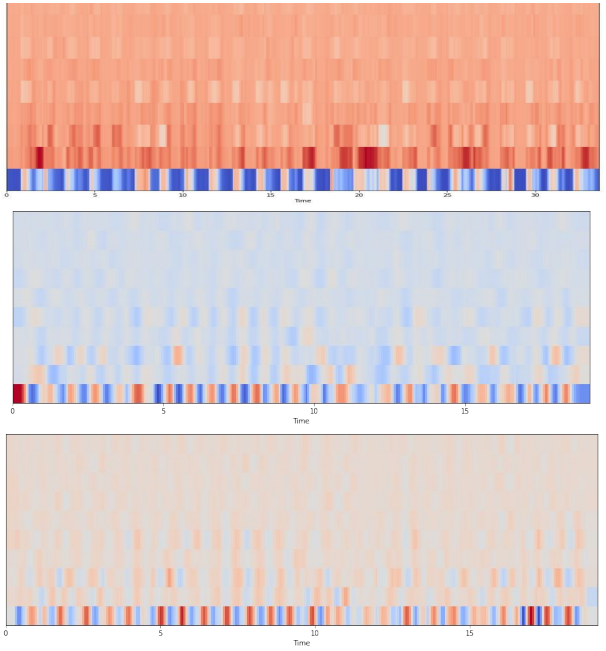


Fig : MFCCs, Delta , Double Delta

Delta and Double Delta Coefficient

- These are the derivatives of MFCCs
- These content dynamic features like the emotion of the speaker
- Delta coefficients can be calculated by subtracting MFCCs coefficients of that frame from the previous frame. For the nth frame
- where dt is delta coefficient for frame t,computed in terms of the static coefficients ct. A typical value

$$d_t = \frac{\Sigma_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\Sigma_{n=1}^{N} n^2}$$

- Delta-Delta (Acceleration) coefficients are derived similarly to static coefficients but from deltas rather than static coefficients.
- Double Delta coefficients can be calculated by subtracting the Delta coefficients of that frame from the previous frame. For nth frame

$$dd_t = \frac{\Sigma_{n=1}^{N} n(d_{t+n} - d_{t-n})}{2\Sigma_{n=1}^{N} n^2}$$

Processing Data

- It turns out that it takes quite a lot of time to pre-process all this while training.
- It's better to do that offline then save all of that information in a JSON file and then retrieve it at training time .
- Mapping is very important as without mapping the data it will be difficult to pass it to the Model
- Data is stored in the dictionary

  (i) mappings ,

  (ii) labels,

  (iii) and features.
- Mappings basically map to different users, for example, user1 is mapped with zero and user2 can be mapped with one and so on.
- All the subdirectories are explored. So that all the speakers must be included.
- Ensured that an audio file which is a signal has enough samples to consider which is equal to one second.
- Further, used the above Feature Extraction steps to extract MFCCs.

Model

- A sequential 3 Layer CNN model is Implemented using Tensorflow
- Input shape is (16,39 ,1)

  16 → is number of segments,

  39 → is number of MFCCs coefficients,

  1 → is indicating for grayscale images
- Batch Normalization is used to speed up the training and decrease the number of training epochs required to train
- Max-pooling is applied to downsample the output of the convolutional layer by the factor of two
- The activation function :

  (i) rectified linear unit (ReLu) : convolutional layers

  (ii) Softmax : Final Layer
- L2 Regularization : tackle the issue of overfitting,
- Model : Train , Evaluate on the test splits

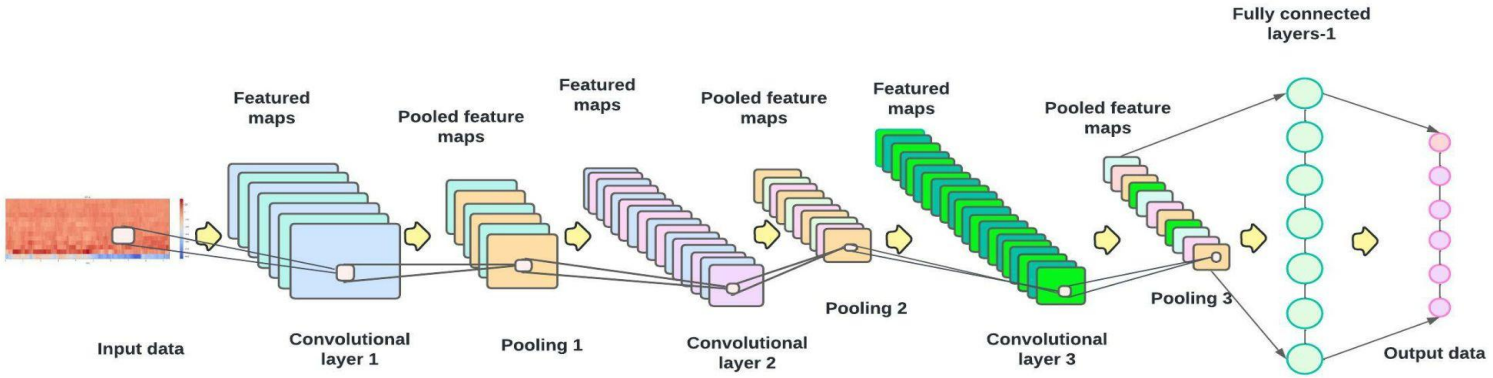


Figure : Structure of CNN Model

Model Overview

- The first layer has 64 filters and a kernel of 3 by 3. A (3,3) filter and (2,2) strides is used in the max-pooling layer.
- The second convolutional layer has 32 filters with a kernel size of 3 by 3. ReLu activation function is used. Max pooling layer has strides 2 by 2.
- The third layer has 32 filters with kernel sizes of 2 by 2. The activation function is Relu and the max-pooling layer has strides of 2 by 2.
- After implementing the 3rd convolution layer, the Output of this layer is flattened and fed into a dense layer. Then used a softmax output layer.It uses cross-entropy loss.
- The softmax function is used to compress raw class scores into normalized positive values that sum to one when added together, allowing the cross-entropy loss to be applied.

Table : Model Overview

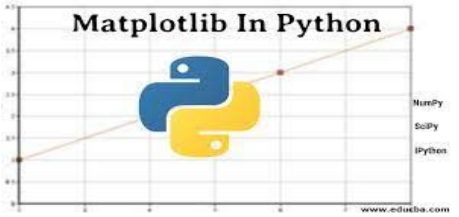
Layer(type)	**Output Shape**	**Parameter**
Conv2D_1	(14,37,64)	640
batch_normalization_1	(14,37,64)	256
max_pooling2d_1	(7,19,64)	0
Conv2d_2	(5,17,32)	18464
batch_normalization_2	(5,17,32)	128
max_pooling2d_2	(3,9,32)	0
Conv2d_3	(2,8,32)	4128
batch_normalization_3	(2,8,32)	128
max_pooling2d_3	(1,4,32)	0
flatten	(128)	0
Dense	(60)	8256

Experimental Setup

1. A simpler setup involves uploading dataset to google drive and running the code in colab platform with python 3.6+ version
2. It which provides with the appropriate GPU's necessary for performing calculations.
3. Dataset have 30k audio file, It is divided into 18k for training, 6k for validating and 6k for testing.
4. The operating system used is Windows 10 with 8GB RAM and Intel i5 10th gen processor.
5. Necessaries Libraries -

- TensorFlow,
- NumPy,
- matplotlib
- sklearn
- json

Dataset Overview



data (audioMNIST)

- The dataset consists of 30000 audio samples of 60 different speakers.
- There is one directory per speaker holding the audio recordings.
- .wav format
- Additionally "audioMNIST_meta.txt" provides meta information such as gender or age of each speaker.

Result

For 70-30 split

- Test Accuracy : 88.79 %
- Test Loss : 0.57 %

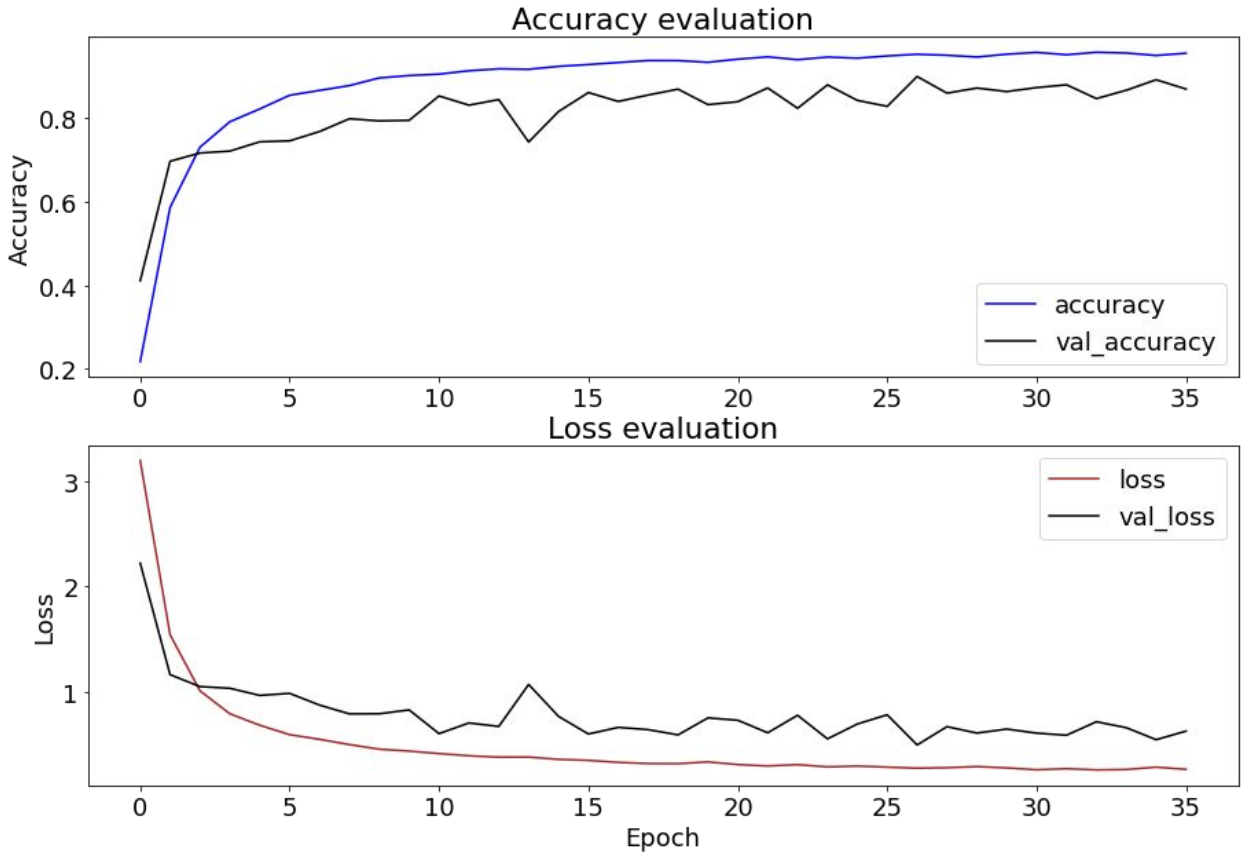


Figure: Accuray and loss curve for train 70% and test 30%

Result

For 80-20 split

- Test Accuracy : 90.03 %
- Test Loss : 0.49 %

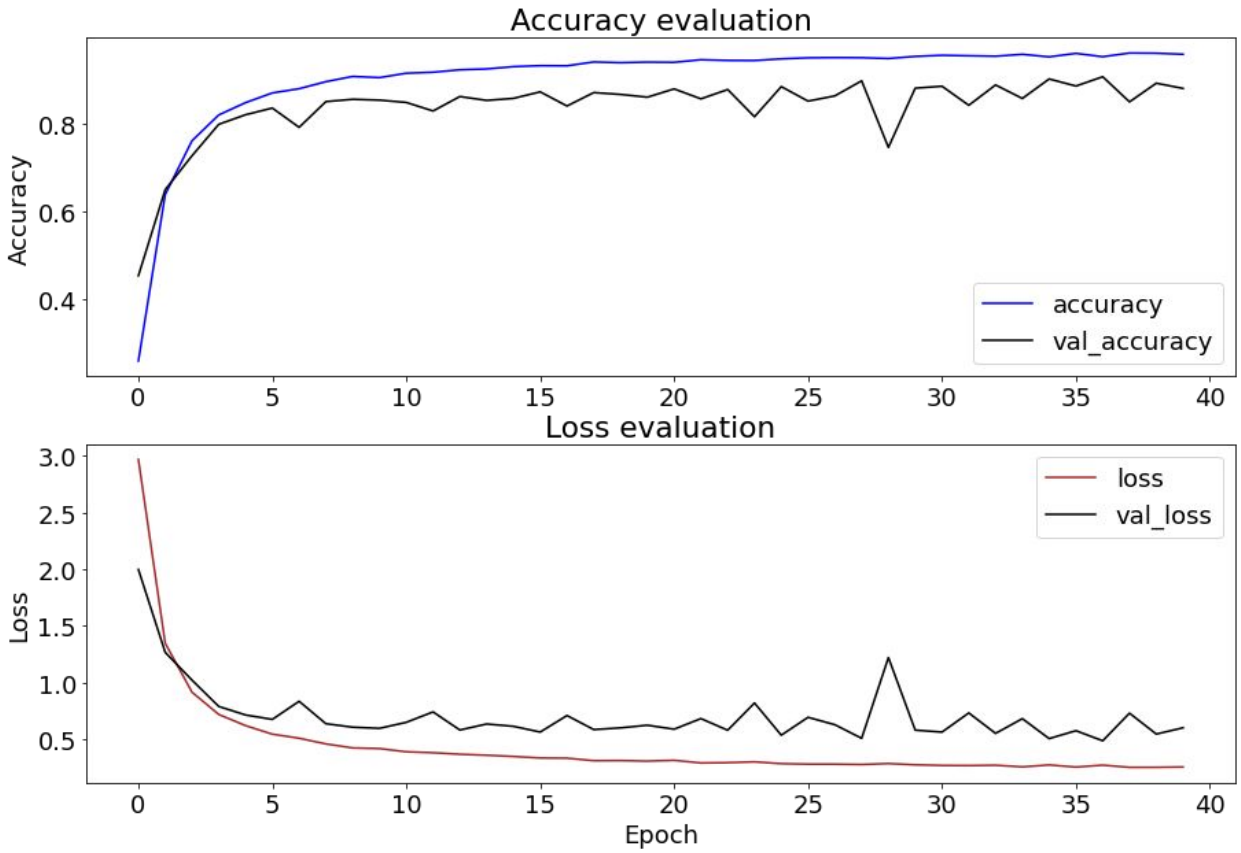


Figure: Accuray and loss curve for train 80% and test 20%

Result

For 90-10 split

- Test Accuracy: 89.49 %
- Test Loss: 0.52 %

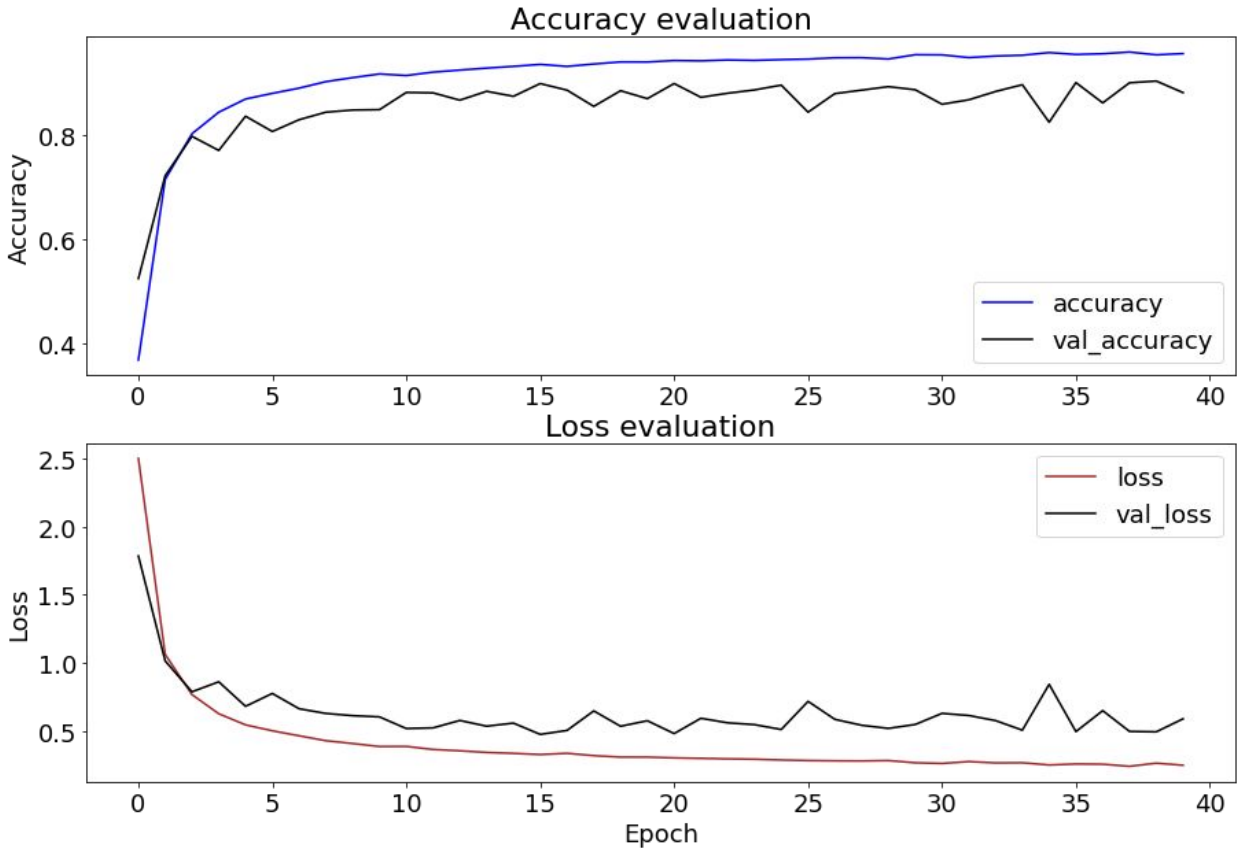


Figure: Accuray and loss curve for train 90% and test 10%

Result Summary

Method	Model	Data Split	No. of Speaker	Accuracy
[1] CNN based speaker recognition in language and text-independent small scale system," 2019	CNN	Train:80% Test:20%	50	71%
	DNN	Train:80% Test:20%	50	61%
Proposed Method	CNN	Train: 70% Test: 30%	60	88.79%
	CNN	Train: 80% Test : 20%	60	90.03%
	CNN	Train: 90% Test : 10%	60	89.49%

Conclusion

- In this paper the aim is to Explore Preprocessing Techniques for speaker recognition and implement a deep learning based speaker recognition system.
- Speaker recognition system is successfully implemented and got better results in terms of accuracy on the AudioMNIST Dataset of 60 different users.
- Result are compared with three data splits,

  For 70% train and 30% test split, accuracy is 88.79% and loss is 0.57% .

  For 80% train and 20% test split, accuracy is 90.03% and loss is 0.49% .

  For 90% train and 10% test split, accuracy is 89.49% and loss is 0.52%
- The presence of irrelevant noise can degrade the accuracy of the system but our pre-processing steps helps us overcome this problem.

Future Works :

- Work on the accuracy
- Building Real time System

Application :

- Authentication
- Surveillance
- Speech Data Management

Challenges:

- Accuracy
- Inconsistencies in the different types of audio and their quality
- Ability to efficiently isolate speakers

References

[1] El-Moneim, Samia & El-Rabaie, El-Sayed & Nassar, Ma'En & Dessouky, M.I. & Ismail, Nabil & El-Fishawy, Adel & Abd El-Samie, Fathi. (2020). Speaker recognition based on pre-processing approaches.

[2] R. Jagiasi, S. Ghosalkar, P. Kulal and A. Bharambe, "CNN based speaker recognition in language and text-independent small scale system," 2019

[3] Bai, Zhongxin, and Xiao-Lei Zhang. "Speaker recognition based on deep learning: An overview." *Neural Networks* 140 (2021)

[4] D Anggraeni et al "The Implementation of Speech Recognition using Mel-Frequency Cepstrum Coefficients (MFCC) and Support Vector Machine (SVM) method based on Python to Control Robot Arm", 2018 IOP Conf. Ser.

[5] Kabir, Muhammad & Ph. D., M. & Shin, Jungpil & Jahan, Israt & Ohi, Abu. (2021)."A Survey of Speaker Recognition: Fundamental Theories, Recognition Methods and Opportunities". IEEE Access. PP. 1-1. 10.1109/ACCESS,2021

[6] M. Wang, T. Sirlapu, A. Kwasniewska, M. Szankin, M. Bartscherer, and R. Nicolas, "Speaker Recognition Using Convolutional Neural Network with Minimal Training Data," 2018 11th International Conference on Human System Interaction (HSI), 2018

[7] K. R. Farrell, R. J. Mammone and K. T. Assaleh, "Speaker recognition using neural networks and conventional classifiers," in IEEE Transactions on Speech and Audio Processing, vol. 2, no. 1, pp. 194-205, Jan. 1994

[8] Lei, Yun et al. "A novel scheme for speaker recognition using a phonetically-aware deep neural network." *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014)

[9] Zheng-Hua Tan, Achintya kr. Sarkar, Najim Dehak,rVAD: An unsupervised segment-based robust voice activity detection method, Computer Speech & Language,Volume 59,2020,

[10] Bennani, Y., & Gallinari, P . (1994) . Connectionist approaches for automatic speaker recognition. In: Automatic Speaker Recognition, Identification and Verification

[11] X.-L. Zhang, J. Wu, Deep belief networks based voice activity detection, IEEE Transactions on Audio, Speech, and Language Processing 21 (4) (2013) 697–710.

Thank You all.