Speaker Verification Using Recurrent Neural Network

Mohit Agarwal
San Diego State University
San Diego, United States
magarwal8188@sdsu.edu

Vivek Kumar
San Diego State University
San Diego, United States
vkumar6934@sdsu.edu

Yage Jin
San Diego State University
San Diego, United States
yjin7099@sdsu.edu

***Abstract*—Humans are able to recognize a speaker by listening to their voice over any number of sources of audio. This ability to identify sound, recognize and differentiate between them has been introduced. This verification system recognizes and differentiates between speakers by extracting features and characteristics from the speaker's voice and analyzing them. Speaker Verification is emerging as a biometric security tool which adds convenience of non-physical interaction. Specifically, this project provided a 4-way speaker identification and speaker verification functionality, which identified the person speaking as Jacob, Mohit, Vivek or others. The idea being first three speakers will be granted access to the system whereas others will not be granted access.**

I. Introduction

In a fast-growing digital world, the need for systems to understand and work better with human speech is ever growing. The ability of working with speech for identification and verification can be used in an array of fields like opening or closing security doors, using voice commands to access labs and homes, biometric phone security based on sound. In order to be able to use voice verification systems for authentication and authorization purposes, it is imperative that they be able to distinguish an authorized user from an unauthorized user which can be done by feature recognition, speaker identification and speaker verification.

The major objective of this study is to classify a speaker into 4 categories: Jacob, Mohit, Vivek or others, so that specific speakers may be granted access to corresponding systems and others will not be able to access these systems. We used our own dataset created by samples collected for Jacob, Mohit, Vivek and 5 friends. A Text-Dependent approach was used where the samples recorded were based on a fixed transcript. To reduce hardware dependencies, samples were collected on multiple devices. This dataset was used for feature extraction to create a speaker model for the speakers and others. These models helped us achieve our objective of speaker verification. The second objective was to see how the variation in the parameters of the models affected the performance of the model. Several parameters were varied such as l2 regularization values, width and depth of the model.

Previous work done in the field of speaker verification gave us ideas on how to proceed with this experiment. We studied that Deep neural network were shown to be successful in implementing non-linear complex models to learn unique features of data [1]. Also, there are four major sound verification steps. Firstly, we need to acquire speech. Second step consists of feature extraction and feature selection. The third step is to cluster the feature vectors then store it in a database. Fourth step is to decide through pattern matching [2].

II. Methods

In this study, we recorded our own data set from 8 people, who are Jacob, Mohit, Vivek and 5 friends. The first three participants recorded 10 audio files each reading 10 transcripts. The remaining 5 participants recorded 2 audio files each, using the same transcripts, which are considered as "other" people. Each audio file is recorded using a single channel and 8000 Hz sample rate. The recording was done on 3 different machines. Each recording is between 30 seconds to 2 minutes, depending on the length of the transcript and the talking speed of the speaker.

We started the experiment with parsing the audio files into frames with 20 milliseconds frame size and 10 milliseconds advance, which was 160 samples per frame. A spectrogram was constructed by adding a hamming window on the frames and applying a Short-Time Fourier Transform (STFT) after, for each of the audio files. Once the spectrogram was computed, a mel scale filter with 20 mels was applied on the squared spectrogram in order to reduce the complexity and provide a better resolution for lower frequency samples. The result was then converted to decibels. In order to prevent calculating log of 0, a very small number, 10^{-9}, was added before the decibel conversion.

We then worked on removing all the noise frames from the mel spectrogram. These noise frames included pauses during speech and silences. A root mean square signal for each audio file was calculated and passed to a Gaussian mixture model, where the RMS signals were categorized into 2 classes, noise and speech. Boolean values were used to label these signals, where true represented speech and false represented noise. Using these labels, the noise frames were removed from the mel spectrogram leaving only the speech frames.

Recurrent neural networks were used to train and test on these mel spectrograms using 5-fold cross validation. The training data consisted of 32 mel spectrograms, 8 from each person (the recordings from "other" category was counted as one person), and the testing data consisted of 8 mel spectrograms, 2 from each person. Each speaker was labeled with a speaker index, where 0 represented Jacob, 1 represented Mohit, 2 represented Vivek, and 3 represented others. For both training and testing data, a batch generator with batch size of 4 was used to group the data. For each batch, the 4 mel spectrograms were padded with 0s to the same length in the time domain, and each frame of the spectrograms was labeled with their speaker index using one-hot encoding.

We used both error rate from speaker predictions per frame as well as speaker predictions per audio file to evaluate the performance of the models. The prediction per frame was considered using the prediction of the speaker with the highest probability. The prediction per audio file was calculated by finding the most predicted speaker for all frames from the audio file. Error rate average and standard deviation were calculated for the 5 models used during 5-fold cross validation to better eliminate randomness.

III. Experiments

In our experiment, we started with a base model with a masking layer followed by 2 hidden LSTM layers each with 100 nodes, batch normalization layers after each hidden LSTM layer, and a dropout layer after the first batch normalization layer. Each LSTM layer used l2 regularizer and the dropout layer used a 20% drop rate. The output layer used softmax as its activation function, and it was wrapped with a time-distributed layer for the model to handle the time dimension of our data. For each of our experiments, we used the best performing model with the lowest average error rate on file prediction, from the previous experiment as the new base model, and then varied the parameters.

The first experiment we did was altering the l2 regularizer penalty, testing how the penalties affect the performance of our recurrent neural network. The different l2 penalties tested were 0.001, 0.005, 0.01 and 0.02. All the other parameters of the model remained unchanged.

We then attempted to change the number of nodes in each hidden LSTM layer using the parameters from the previous best performing model, which was used as the new base model. We wanted to see how well our model learned when we increased and decreased the learning capacity for the neural network. We tested models with 50 and 200 nodes in the hidden LSTM layer and compared the result with the new base model.

Number of hidden LSTM layers were also changed for us to find out the parameters of the best performing model. Modifying this parameter also changed the learning capacity of the model. Models with 1 and 3 hidden LSTM layer were tested against the previous best performing model.

Lastly, we would like to test our best performing model with a different speaker prediction algorithm. We kept all parameters the same and changed how we train and test the model. In this experiment, we only trained the model with the audio files from Jacob, Mohit, and Vivek, using 5-fold cross validation and batch generator, and tested the model with the test fold plus all 10 recordings from others. To determine the performance of this model, we only used error rate on speaker prediction per audio file. The speaker prediction per file was calculated using the formula below, where p represented all predicted probabilities for all frames of the given speaker, and N represented the number of frames in the audio files:

$$\exp(\mathrm{sum}(\log(p)) / N)$$

This gave us three probabilities of the audio file spoken by the three of us. We then took the highest probability as the speaker prediction. If the probability was higher than 60%, we would take the predicted speaker as the final prediction. If the probability fell below 60%, then we would reject the prediction and mark the prediction as other.

IV. Results

We will talk about the findings based on different hyper-parameters in this part.

To begin, an experiment was carried out with the following parameters: two hidden layers with 100 nodes each, batch normalization after each hidden layer, dropout layer with 0.2 drop rate after the first normalization layer, L2 regularization with a setting of 0.01, and softmax activation function in the output layer. The trial yielded an accuracy of roughly 80%, or a 20% error rate averaged for a 5-fold.

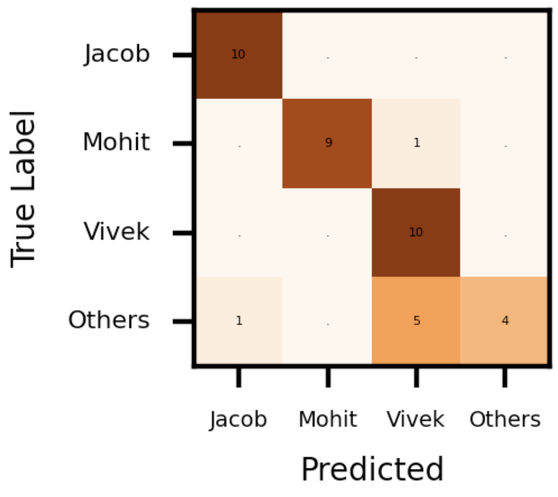

Fig. 1 Confusion matrix for file prediction with best performing model

IV.I. Variation in l2 penalty values

Increasing or decreasing the l2 penalty didn’t result in the increase of the performance of the model as compared to that of the base case. In both the variations the error rate was increased as compared to that of the base case model.

Variation in l2 penalty				
Diff. l2 Penalty values	Avg. Frame Error Rate (%)	Std. of Frame Error Rate (%)	Avg. File Error Rate (%)	Std. in File Error Rate (%)
0.001	36.21	16.28	27.50	27.39
0.005	32.28	16.61	30.00	16.96
0.01	26.75	14.67	20.00	10.00
0.02	32.75	16.84	25.00	23.72

Table 1 Test average and standard deviation error rate for models with different l2 penalty values

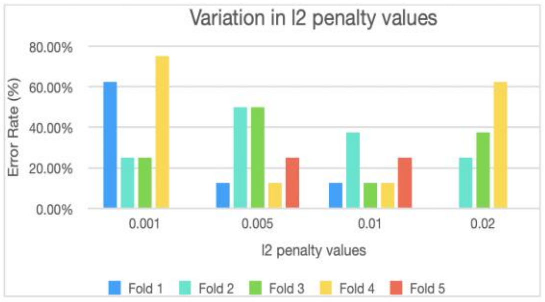

Fig. 2 Test error rate on file prediction for models with different l2 penalty values for all five-fold

IV.II. Variation in width of model

The width of the model was the second parameter to be altered. The neural network performed better with the model with fewer nodes. The model with twice as many nodes in the hidden layer as the base case had a lower model performance. When the network's width was doubled, the error rate increased by 10% as that of the base case.

Variation in width				
Num. of Nodes	Avg. Frame Error Rate (%)	Std. of Frame Error Rate (%)	Avg. File Error Rate (%)	Std. in File Error Rate (%)
50	26.43	10.93	17.50	12.75
100	26.75	14.67	20.00	10.00
200	38.81	22.89	32.50	26.93

Table 2 Test average and standard deviation error rate for models with different width

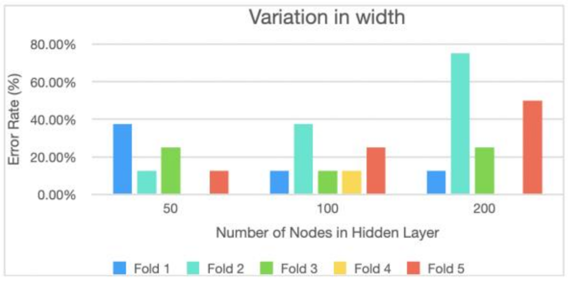

Fig. 3 Test error rate on file prediction for models with different width for all five-fold

IV.III. Variation in depth of model

The performance of the neural network was likewise reduced as the depth of the model was reduced. Additionally, boosting the hidden layer did not improve the model's performance. As the depth of the model was increased or decreased, the error rate was nearly doubled as compared to that of the base case.

Variation in depth				
Num. of Hidden layer	Avg. Frame Error Rate (%)	Std. of Frame Error Rate (%)	Avg. File Error Rate (%)	Std. in File Error Rate (%)
1	36.69	15.53	30.00	23.18
2	26.43	10.93	17.50	12.75
3	45.63	21.28	40.00	25.50

Table 3 Test average and standard deviation error rate for models with different depth

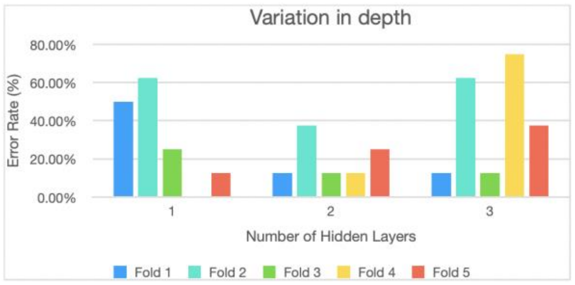

Fig. 4 Test error rate on file prediction for models with different depth for all five-fold

IV.IV. Compare to threshold model

The performance of the threshold model was far worse than the best performing model from our previous experiment, with an error rate of 46.25% compared to the previous error rate of 17.5%. This is expected as the model was not confident in some of the predictions. We also noticed that our model with threshold falsely recognized many recordings from others as Vivek, which would cause a huge problem for speaker verification systems.

Comparison with Threshold Model		
Model	**Avg. File Error Rate (%)**	**Std. in File Error Rate (%)**
2 Layer 50 nodes 0.01 l2	17.50	12.75
Threshold	46.25	26.10

Table 4 Test average and standard deviation error rate for best performing models and threshold model

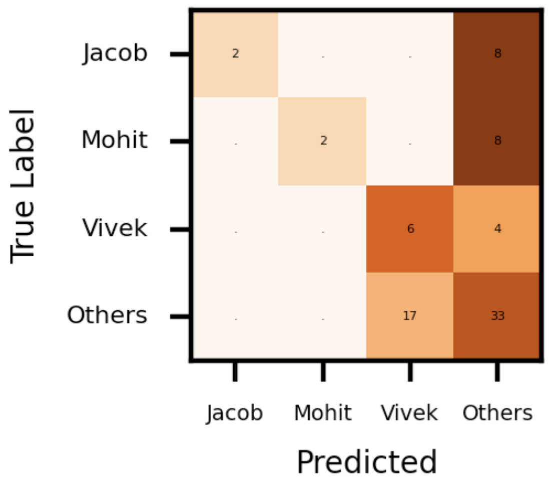

Fig. 6 Confusion matrix for file prediction with threshold model

V. Discussion

There were some limitations in our experiment. The first one being the recordings were done with different machines during our data collection. This made the recordings slightly different based on the machine it was recorded on, and it was possible for our model to pick up these little differences and learn the features of the machines instead of the voice for every speaker [3]. Another limitation was that we did not have a large dataset. We only recorded 40 audio files with 10 recordings for each category. This may very possibly cause our model to overfit during training, and it may not be able to learn the voice features for each speaker we were verifying.

To best eliminate the randomness from training and testing our model, we used 5-fold cross validation to train and test the models with different parameters. The average error rate was calculated to determine the overall performance for each model. Even with this effort, it is still possible that we may have different results if we conduct this experiment again using the same parameters.

In our experiment, we found that the parameters that affected the learning capacity of the model had the biggest impact on testing error rate in file prediction. We believed that this was a side effect of having a small dataset. With a smaller capacity to learn, the model is less likely to overfit, which would help the model to perform better during testing. However, when the model had little capacity to learn, in our case, 1 hidden layer with 50 nodes, the model performed very poorly due to not being able to learn well.

We also found that the learning rate, l2 penalty, had some effects on the accuracy of the model, but it was not as significant as the width or depth of the model. We noticed that when the width and depth of the model was fixed, the model with a penalty of 0.01 had the best performance at 20% error rate. And when the penalty increased or decreased from this value, the error rate always increased.

As for our threshold model, it performed far worse than we expected, with an error rate of 46.25%. It correctly verified the speaker 10 out of 30 times during testing, and falsely verified other people as Vivek 17 out of 50 times.

Meanwhile, our best performing model was able to correctly verify the speaker 29 out of 30 times, and falsely verify other people as one of us 6 out of 10 times. We believed that with a larger dataset for each speaker, and some fine tuning on the model and the prediction threshold, we could improve the accuracy of this threshold model significantly.

VI. Conclusion

In this study, we trained recurrent neural networks to verify the identity of the speaker. We were able to reduce the average testing error rate per file prediction across 5-fold to 17.5% through modifying the configuration of the model. We also used this configuration to only train with the recordings from 3 of us and test it with the remaining recordings with 60% confidence threshold. We found that this model with threshold did not perform well, and it has an average error rate per file prediction at 46.25%. In the future, we would like to work on fine tuning the model using threshold to achieve better accuracy in speaker verification.

VII. References:

[1] A. Irum and A. Salman, ‘‘Speaker verification using deep neural networks: A review, ’’Int. J. Mach. Learn. Comput., vol. 9, no. 1, pp. 1–6, 2019.

[2] P. Premakanthan and W. Mikhael, ‘‘Speaker verification/recognition and the importance of selective feature extraction, ’’in Proc. 44th IEEE Midwest Symp. Circuits Syst. (MWSCAS), vol. 1, Jun. 2001, pp. 57–61.

[3] Whitman, B., Flake, G., and Lawrence, S, "Artist detection in music with Minnowmatch," in Proc. IEEE Neural Networks Signal Processing XI, 2001, pp. 559-568. doi:10.1109/nnsp.2001.943160

[4]http://librosa.org/doc/main/generated/librosa.stft.html