AUDIO FILE CLASSIFICATION USING SNR AND RATE OF SPEECH

[1] Akshiya Sree J, [2]Jayanth J, [3]Jagadeeshwaran T, [4] Devi Bala,
[5]Dr. Jayabalan Ramesh, [6]Periyasamy P, [7]Sandip Bapat
[1,2,3]UG Scholars, Department of IT, [4]UG Scholar, Department of ECE, [5]Professor, Department of ECE,
PSG College of Technology,
[7]Architect, Samsung, [8]Senior Principal
Engineer, Samsung, India

***Abstract*— A good Audio Speech Recognition (ASR) system requires good quality 'balanced' data- in terms of rate of speech and ambient noise. To build a balanced corpus, it is essential to identify the utterances that have high rate of speech from a large unlabeled corpus. This paper introduces a speech processing system that can detect the location and the type of audio signal in variable noisy environments. This system detects the location of the audio source using real time audio datasets; the system examines the audio first, determines if it is speech/nonspeech, then estimates the value of the signal to noise ratio (SNR) and rate of speech using machine learning algorithms and deep learning techniques. Using this value, instead of trying to adapt the speech signal to the speech processing system, this adapts the speech processing system to the surrounding environment of the captured speech signal. Here an algorithm is incorporated which uses deep neural network (DNN) as the neural architecture, that determines SNR with accuracy greater than 90% and rate of speech with accuracy greater than 90%. This paper proposes a novel deep learning approach to build a robust ASR to classify the audio signals based on SNR and Rate of Speech.**

Keywords— deep learning, SNR, Rate of Speech.

I. INTRODUCTION

In recent times, the technological advancement has made progress from text-based intelligence to audio-based intelligence. For any of those reliable intelligence non-destructive audio classification is cardinal. The basic process of obtaining well defined audio signal involves characterizing the features of audio such as speech and noise with clear differentiation between them. The existing audio classification systems are not robust to noise and disturbances which persuade the segregated use of speech signals in an audio file.

In order to avoid this, a categorization on the measurable quantity of noise present in an audio signal has to be derived so that the further enhancements on the speech or elimination of noise can be made of the audio signal. Further on imparting out the noises, speech signals have various intensities of highs and lows which gets varied based on the source speaker of the speech signal. Human propagation of speech various are not static and hence varies dynamically in its rate based on the speaker. Thus, well-defined classification of amount of noise and rate of speech in an audio signal will be a rudimentary part in audio signal-based intelligence.

II. LITERATURE WORK

Ahmad R. Abu-El-Quran, Adrian D. C. Chan, And Rafik A. Goubran et al. [1] explains an approach that uses a multiengine speech processing system that can detect the location and the type of audio signal in variable noisy environments. This system detects the location of the audio source using a microphone array; the system examines the audio first, determines if it is speech/non-speech, and then estimates the value of the signal to noise ratio using a Discrete-Valued SNR Estimator. Using this SNR value, instead of trying to adapt the speech signal to the speech processing system, the speech processing system is adapted to the surrounding environment of the captured speech signal. In this paper, the concepts of Discrete-Valued SNR Estimator and a multiengine classifier were introduced, using Multiengine Selection or Multiengine Weighted Fusion. Also, the inference used the Speaker Identification (SI) as example of the speech processing. The Discrete-Valued SNR Estimator achieves an accuracy of 98.4% in characterizing the environment. Compared to a conventional single engine Speaker Identification (SI) system,

the improvement in accuracy was as high as 9.0% and 10.0% for the Multiengine Selection and Multiengine Weighted Fusion, respectively.

Based on audio analysis by Honglak Lee, Yan Largman, Peter Pham, Andrew Y. Ng et al. [2] the most preferable digital format should portray variations between different audio samples. A mono-channel audio is usually preferred for a clean classification over the multi-channel audio formats. A 16-bit PCM (Pulse Code Modulation) is used to convert analog signals to digital formats. The inference suggests the use of audio signal after stipulating the signals with Pulse-code modulation (PCM) which is a method used to digitally represent sampled analog signals. The inference also suggests the use of a certain definite audio format across the whole system, which in turn reduces the factors of variability in the input audio formats used in the system. The paper endorses the use of 16-bit PCM as the preferable audio source for the applications based on Audio Classification which is accredited to have higher reliability while manipulating the input audio signal features for further processing. Based on the study, the source audio components which are used in the audio file classification system are converted to 16-bit PCM which biases the whole system under a uniform shelter of input audio format.

R. Gomez And T. Kawahara et al. [3] puts forward that digitized audio can vary depending on the application aspects. This variation has effect on various audio features. The study on different features proves that most of the audio variation is categorized by Mel Frequency Cepstral Coefficients (MFCC) feature which is represented widely using 40 coefficients of variation. In audio signal processing, the Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum").

The difference between the cepstrum and the Mel-frequency cepstrum as explained by Azhar S. Abdulaziz1, Veton Z. Kepuska et al. [4] is that in the MFCC, the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

The basic variation between model building based on SNR and rate of speech is referenced by Abdellah Yousfi, Abdelouafi Meziane et al. [5]. The variation can be best suited for audio signal processing in a natural environment. The speech classification needs other qualities of an audio to assist the difference. Based on references on speech recognition, the spectral contrast features for speech analysis. Spectral contrast considers the spectral peak, the spectral valley, and their difference in each frequency sub-band.

III. SPECIFICATIONS OF NEURAL NETWORK MODEL

1. Deep Neural Network (DNN) Model

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers.

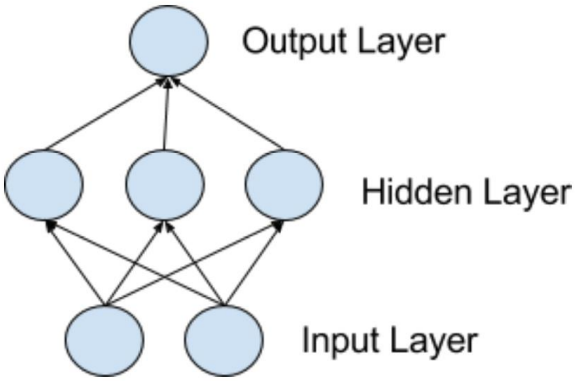

Figure 2.1 Layers of DNN

Input or Visible Layers: The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset.

Hidden Layers: These layers are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value. These hidden layers consist of neurons which trains based on data previously collected.

Output Layer: The multiclass classification has multiple output layers where each of the layer is based on the corresponding output class of the classification system.

2. Neurons

The building block for neural networks are artificial neurons. These are simple computational units that have input signals with their weights and produce an output signal using an activation function.

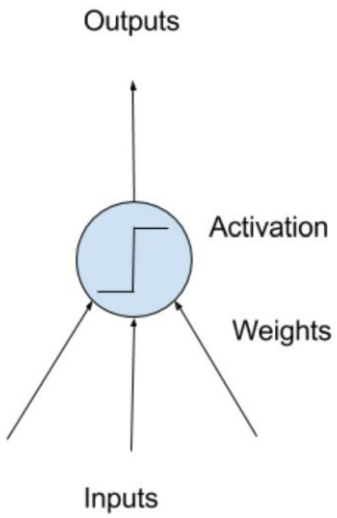

Figure 2.2 Output Neuron

3. Activation Function

An activation function is a simple mapping of summed weighted input to the output of the neuron. The activation function governs the threshold at which the neuron is activated and the strength of the output signal. The Audio Classification system uses activation functions “ReLU” and “Softmax” in the neural architecture.

4. ReLU Activation Function

The Rectified Linear Activation (ReLU) function, is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. In order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned. The function must also provide more sensitivity to the activation sum input and avoid easy saturation.
The solution is to use the rectified linear activation function.

5. Softmax Activation Function

Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. The most common use of the Softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output a specific number of values, one for each class in the classification task, and the Softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the Softmax function is interpreted as the probability of membership for each class.

6. Simplified process flow of DNN:

The Audio Classification system uses custom functions to extract various features from the input source audio files. These features are passed as the input parameters to the input layer of the deep neural network (DNN).The number of input layers in the neural architecture corresponds to the total number of features extracted from the source audio files. Then the input layer passes it to the hidden layer which consists on number of neurons which works based on the activation functions used and produces the convolved output to the output layer. The output layer corresponds to the total number of different output classes involved in the output prediction of the Audio Classification system.

IV. DATASET

1. Limitations on noise estimation

For estimation of noise present in an audio signals SNR (signal to noise ratio) could be an exact representation for further computation. But the estimation of noise in an unlabeled audio file is a tedious process and there are no predefined systems in existence which could accurately measure the amount of noise present in an unlabeled audio file. For the calculation of SNR, the accurate levels of normal audio and the noise present in the audio file is a necessity. This accuracy in measurement of SNR directly varies the output prediction or accuracy of the DNN model and so the trusted measurable format of noise should be a clean asset in the basic structure of the DNN model.

2. Overcome limitation of noise estimation

To overcome the limitations of obtaining an exact measurable quantity of SNR a noise mixer technique is incorporated. This noise mixture is a python-based snippet which takes in a clean audio dataset and separate noise dataset as inputs. The noise mixer snippet iterates for each clean audio and adds particular decibel (dB) of noise to each clean audio file in the dataset. For example, a clean audio file will be added with various amounts of noise (10dB, 20dB, 30dB). Thus, the output of the noise mixer would produce four different classes of audio files which has 0dB (highly noisy), 10dB, 20dB, and 30dB (very less noisy) classes.

3. Dataset statistics for SNR model

Dataset	Number of audio files	Number of train files	Number of test files
DEMAND +Libri Speech	18,000	14,400	3,600

Table 3.1: Dataset statistics for SNR classification

4. Creating a dataset for Rate of Speech classification

The creation of dataset for distinct classes of Rate of Speech involves a manual process using "*Audacity*" tool. The rudimentary part of processing the dataset is that only the playback rate of the audio file is to be changed without changing the pitch of the audio file. The clean audio dataset is imported into "*Audacity*" tool and the playback rate is changed from 1.0x to 0.6x and 1.4x. Thus, after processing the final dataset consists of three classes namely fast (1.4x), normal (1.0x) and slow (0.6x).

5. Dataset statistics for Rate of Speech model

Dataset	Number of audio files	Number of train files	Number of test files
Libri Speech	3000	2400	600

Table 3.2 Dataset statistics for Rate of Speech classification

V. DESIGN AND IMPLEMENTATION

1. Audio Preprocessing for SNR classification

To create a labelled corpus as the noise speech files, the speech dataset of both male and female in clean environment was downloaded from "Libri Speech" and the noise files were downloaded from the "DEMAND" dataset which consisted of noise like park, hallway, meeting hall, cafe, etc. Both these audio files should be mixed together to get the noisy speech file. To mix these audio files python libraries were used. Finally, Audio files were created at specific SNR values like 0db, 10db, 20db, 30db. These audio files are the dataset for the project. A CSV file was created with two columns i.e., id (file name in wav format) and snr value. For the deep learning model to be trained 18000 files were created. Each of these files were used to extract the features. Using the inbuilt functions in python, the 40 co-efficient of MFCC was extracted. These extracted features along with the snr values were converted into a data frame for further processing.

The data frame was then again split into input and output values i.e., the features were input and SNR was the output. These were made into a "numpy" array and the output was encoded to categorical since the model uses a categorical cross entropy as its loss function. Using "sklearn" package the dataset was split into 20% for testing and 80% for training.

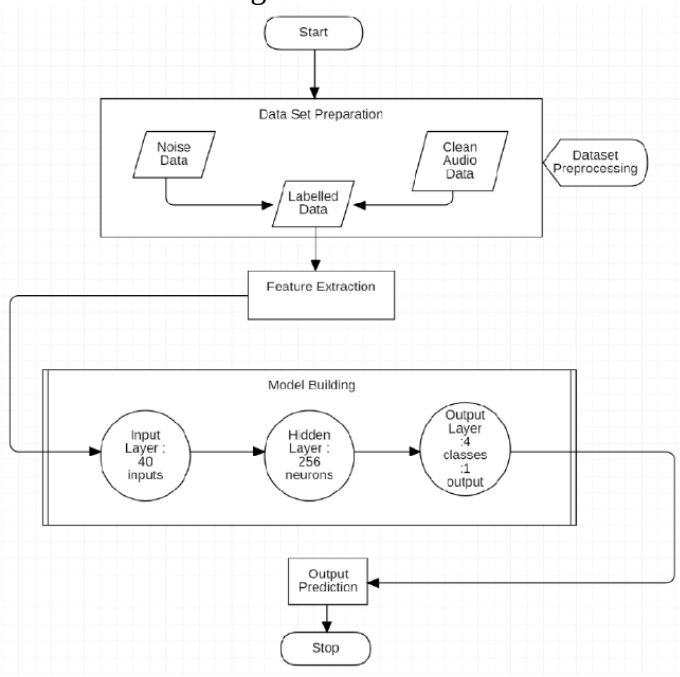

Figure 3.1 Flow Chart of SNR Model

2. *Audio Preprocessing for Rate of Speech classification*

To create a labelled corpus, the speech dataset of both male and female in clean environment was taken from "LibriSpeech". The audio files with about equal words per minute are taken for the manipulation. Used "audacity" application to manipulate the playback rate without changing the pitch of the audio files. By this, a labelled corpus of audio was created with playback rate x1.4, x1.0 and 0.6x with their classes as high, normal and low respectively. With the audio file name as id and their respective classes as output class name a csv file is created for the labelled corpus. For few audio files, the text files were used to count the number of words in each audio file. The words per minute was calculated by using the word count and duration of the audio file. These were classified into three classes as *LOW, NORMAL, HIGH* with each category consisting of nearly 650 audio samples. These audio files are the dataset for the project. A CSV file was created with nearly 3000 rows with two columns i.e., id (file name in wav format) and category. For the deep learning model to be trained 3000 files were created. Each of these files were used to extract the features. Using the inbuilt functions in python, the 40 co-efficient of MFCC and their SPECTRAL CONTRAST features was extracted. These extracted features along with the category were converted into a data frame for further processing. The data frame was then again split into input and output values i.e., the features were input and category was the output. These were made into a "numpy" array and the output was encoded to categorical since the model uses a categorical cross entropy as its loss function. Using "sklearn" package the dataset was split into 20% for testing and 80% for training.

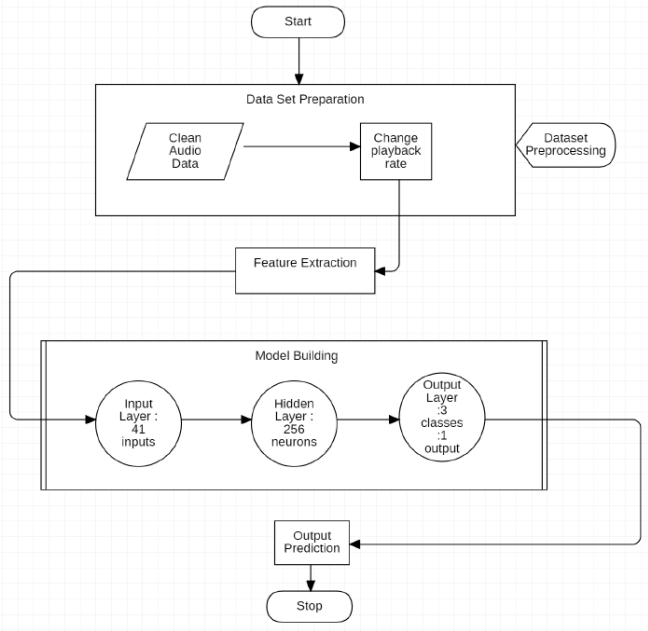

Figure 3.2 Flow Chart of Rate of Speech Model

3. *Deep Neural Networks Model*

A sequential model was built with one input layer and one hidden layer and an output layer. The input layer is a dense layer with 256 nodes and n as the input shape, where n is the number of feature points for the particular model. The number of nodes was chosen based on trial-and-error method corresponding to efficient output. The input layer has an Activation function as "relu" and dropout to avoid overfitting. The hidden layer is also a dense layer with 256 nodes and "relu" as its Activation function and dropout was used to avoid overfitting. The output layer is

a dense layer with N nodes as output, where N is the number of output classes based on the module. It uses Activation function as “Softmax” since the output is multiple class.

4. Implementation

Training and Testing: To train the model, the loss function used was “categorical cross entropy” and “accuracy” was used as a metrics to evaluate the model. The optimizer used was “adam”. The evaluation before building the model was tested. The pre-training accuracy was 32%. Further, 100 epochs were used to train the model with a batch size of 32.

Results: After training the model the training accuracy was 95% and testing accuracy was 93%. New files were used for testing the model which led to correct predictions. The SNR model reduces efficiency at about 30db or higher than that. The Rate of Speech model reduces efficiency when rate of speech is high.

VI. SYSTEM REQUIREMENTS

1. Software Used

Operating system: Windows 10. The following sections review important python packages used in realization of the Audio classification system.

2. Spectrogram Generation

*Wave:*It is used to read the wave file and then the attributes required for spectrograms are found.

*PyLab:*It is used to plot the spectrogram for the wave file and also used for saving the figure.

*OS:*Used to access the wave files in the directory and to split the path.

3. Preprocessing the image

*Cv2:*It is used for Image Processing. It is used in conversion of the spectrogram image into gray scale image, performing operations like opening and closing, cropping the image and writing back the image.

*OS:*Used to access the spectrogram images in the directory.

4. Neural network - Keras

Keras is used as a high-level neural network API, written in python and is capable of running on top of TensorFlow.

“from keras.applications.vgg16 import VGG16” is used since VGG16 is a pre-trained and the convolution base is used.

“from keras import models” is used to get the convolution base of VGG16.

“from keras import layers” is used to add layers on the top of the convolution base.

“from keras import layers” is used to train the model to use the difference between the results obtained and the values which it already knows it to be true, to adjust the weights on the nodes so that the network steps towards a solution.

“from keras.callbacks import Model Checkpoint” is used to save the weights of the model whenever the accuracy of the model increases.

5. Numpy:

“Numpy.zero” is used for keeping the features and labels in an array as it returns a new array of given shape and type, filled with zeros.

“Numpy.reshape” is used to reshape the train features and validation features array to a desired dimension as it gives a new shape to an array without changing its data.

“Numpy.asarray” converts the predictions of the validation data to an array.

“Numpy.round” rounds the predictions of the validation data array.

6. sklearn.metrics:

The sklearn.metrics.classification_report is used for checking the quality of prediction. It includes the classification summary involving metrics such as precision, recall, F1 score for each class and also the overall accuracy of the model. The reported averages include macro average (averaging the unweighted mean per label),

weighted average (averaging the support-weighted mean per label).

7. Matplotlib:

The matplotlib.pyplot is used to plot the accuracy and loss graph for training and testing datasets. The X axis represents the epoch numbers in case of accuracy and loss graph. The Y axis represents the accuracy in case of accuracy graph and loss in case of loss graph. Thus, the results are easily analyzed through the visual representation.

VII. RESULTS AND DISCUSSION

1. Metrics

Metrics are used for evaluating the performance of a machine learning model.
A Classification report is used to measure the quality of predictions from a classification algorithm. It computes the number of predictions which are true and number of predictions which are false with respect to ground truth label. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

2. Precision

Precision is a measure of how precise/accurate your model is out of those predicted positive, how many of them are actual positive.
Precision = True Positive/ Total predicted Positive

3. Recall

Recall calculates the number of the Actual Positives the model captures through labeling (True Positive).
Recall = True Positive/Total Actual Positive

4. F1 Score

F1 Score is a better measure to use which provides a balance between Precision and Recall.
F1 score = (2 * Precision * Recall) / (Precision + Recall)

5. Support

The support is the number of occurrences of the given class in the dataset.

6. Results

Using the proposed methodology 97% accuracy is achieved for devices from which samples were collected to train the model. For devices from which samples weren't used for training the model, the accuracy achieved is 88%. Fig 6.1 shows the quality of predictions for the dataset which were used for training the model.

classification report	precision	recall	f1-score	support
0.0	1.00	0.94	0.97	31
1.0	0.94	1.00	0.97	29
accuracy			0.97	60
macro avg	0.97	0.97	0.97	60
weighted avg	0.97	0.97	0.97	60

Fig 6.1 Classification Report for data used from trained dataset

Fig 6.2 shows the quality of predictions for the dataset which weren't used for training the model.

classification report	precision	recall	f1-score	support
0.0	0.90	0.87	0.88	30
1.0	0.87	0.90	0.89	30
accuracy			0.88	60
macro avg	0.88	0.88	0.88	60
weighted avg	0.88	0.88	0.88	60

Fig 6.2 Classification Report for data used from untrained dataset

Devices	Same	Different
Accuracy	97%	88%

Table 6.1 Comparison between same and different test dataset accuracy

VIII. CONCLUSION

In summary, an efficient model for the classification of audio files based on SNR and Rate of Speech is developed using a sequential deep learning model consisting of one input layer, one hidden layer and one output layer. This algorithm can be deployed in other applications, such as biomedical applications. An example of a biomedical application area that can benefit from the proposed instrument is hearing aid devices.

Hearing aids could utilize the proposed instrument to customize their parameters to achieve an optimal hearing performance in variable environments. The deployment of the proposed algorithm in hearing aids instrument can automate the selection of the predefined settings according to the surrounding environment. The increasing use of speech processing techniques and voice assistants can use this model as an aid to improve its services.

IX. FUTURE WORK

The above audio file classification has provided evident results of accuracy and further both the models could be combined to form a multi classifier system. This multi classifier system can take in an audio file and predict the SNR ratio of an audio file and further the audio samples can be processed to remove the noise in the dataset. The processed input could be used for classification based on Rate of Speech. The concoction of these models could be used to build a multi classifier system. The nature and variation of the dataset could be increased to enhance the extensiveness of the application.

REFERENCES

[1] Ahmad R. Abu-El-Quran, Adrian D. C. Chan, and Rafik A. Goubran, "Multiengine Speech Processing Using SNR Estimator in Variable Noisy Environments", Advances in Acoustics and Vibration Conference, February 2012.

[2] Honglak Lee,Peter T. Pham, Yan argman, Andrew Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks", Advances in Neural Information Processing Systems Conference, 2009.

[3] R. Gomez and T. Kawahara, "Optimizing spectral subtraction and Wiener filtering for robust speech recognition in reverberant and noisy conditions", *IEEE* International Conference on Acoustics, Speech, and Signal Processing March 2010.

[4] Azhar S. Abdulaziz1, Veton Z. Këpuska, "The Short-Time Silence of Speech Signal as Signal-To-Noise Ratio Estimator", International Journal of Engineering Research and Application, August 2016.

[5] Abdellah Yousfi and Abdelouafi Meziane, "Introduction of the speaking rate in the model of speech recognition", International Journal of Mathematics and Mathematical Sciences", 2002.

[6] N. Kitaoka, S. Hamaguchi, and S. Nakagawa, "Noisy speech recognition based on integration/selection of multiple noise suppression methods using noise GMMs," IEICE Transactions on Information and Systems, 2008.

[7] Ian V Mcloughlin, Haomin Zhang, Zhipeng Xie, Yan Song, Wei Xiao, "Robust Sound Event Classification Using Deep Neural Networks", IEEE Transactions on Audio, Speech and Language Processing, March 2015.

[8] Aditya Khamparia, Deepak Gupta, Nhu Gia Nguyen, Ashish Khanna, Babita Pandey, And Prayag Tiwari, "Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network", IEEE Access, January 2019.