Speech Command Recognition

Report by

Rehapriadarsini Manikandasamy

The George Washington University

Author Note

This report was prepared for DATS 6203, taught by Professor Amir Jafari

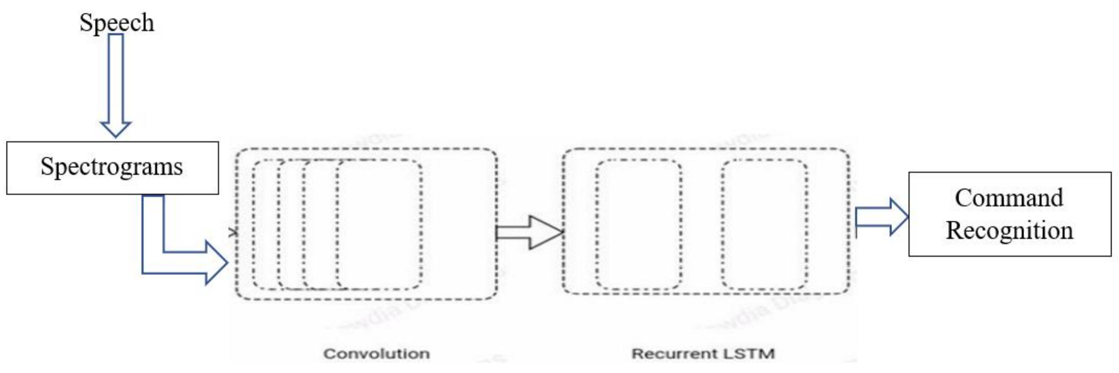
TABLE OF CONTENTS

INTRODUCTION

Automatic speech recognition is a very necessary activity for effective human-computer interaction. Sound is a complex, feature-rich signal, and sound recognition has attracted research interest using a rich portfolio of Machine Learning (ML) methodologies and mechanisms. Such mechanisms include classic ('traditional') ML methods (such as Support Vector Machine, Linear Discriminant Analysis) as well as deep learning (notably Convolutional Neural Networks (CNNs)).

Such sound recognition has extensive real-world applications to identify the import words/phrases in the spoken sentences. One such application is Keyword spotting (KWS) technology, a potential technique to provide fully hands-free interface, and this is especially convenient for mobile devices compared to typing by hands. And it is also the desired technique for situations like driving or some emergency cases. Since speech command recognition system usually runs on smartphones or tablets, it therefore must be low-latency, and must have a very small memory footprint, and require only very small computation.

Implementation of keyword spotting technology involves networks which helps in understanding the specific commands/keywords in the conversations.



Source: (PDF) A Hybrid Technique using CNN+LSTM for Speech Emotion Recognition (researchgate.net)

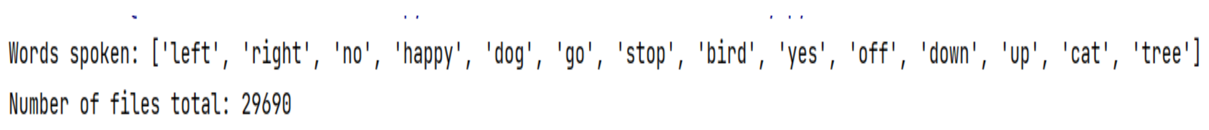
Problem Statement:

The purpose of the project is to develop deep convolutional LSTM network for speech command recognition. The project is motivated by using spectrograms as inputs to the hybrid deep convolutional LSTM for speech emotion recognition. The trained proposed model using four convolutional layers for high-level feature extraction from input spectrograms, LSTM layer for accumulating long-term dependencies and finally dense layer.

DESCRIPTION OF DATASET

The motivation for the customized speech command recognition dataset has been drawn from Google’s speech commands dataset, ESC50, LibriSpeech and various other speech recognition datasets. The dataset used in the project was extracted by a Google Researcher for research purposes from various other similar speech command datasets.

The dataset used in this project is sourced from Kaggle which has audio files (.wav) of duration of one second. The dataset has approximately 30K audio files classified based on 14 major labels. The dataset will be separated into train-test sets manually and processed to be trained on the deep learning models.



```
Words spoken: ['left', 'right', 'no', 'happy', 'dog', 'go', 'stop', 'bird', 'yes', 'off', 'down', 'up', 'cat', 'tree']
Number of files total: 29690
```

Fig: Label categories and total files

The datasets were later preprocessed to produce JSON files containing the following features:

```
data = {
   "mapping": [],
   "labels": [],
   "MFCCs": [],
   "files": []
}
```

Mapping – Has keywords (Eg.: dog, tree)

Labels – Stores the labels encoded (Eg:0,1,2)

MFCCs – Stores the Mel spectrograms in the form of arrays

Files – Have the file paths of the respective audio files

DESCRIPTION OF WORK

My contribution to this project is to pre-process and analyse the data. And to develop CNN base model.

Convolutional Neural Network (CNN):

CNNs initially focused on image classification, object detection, and recognition tasks. Images are employed as input, and ImageNet (http://www.image-net.org/, accessed on 30 June 2021), a 14 million-image database, is used to train and evaluate some of the most known convolutional networks. CNNs are evolving in terms of size (number of layers) and structure (type and connection of layers). Referring to a set of well-known networks, ordered chronologically, of which we select a subset to evaluate in sound recognition.

Thinking of a convolution neural network as an artificial neural network that has some ability to identify patterns and make sense of them. This pattern detection makes neural networks such useful for image analysis. There are mainly four layers in CNN, and we use some additional layers to normalize our network. Each of these are described below:

Convolution Layer: The convolution layer represents CNN's layer one where we interact (images, 1D time series data) with filters or kernels. By using a sliding window, we apply small units across the input and these units are known as filters. In convolution process, element-wise product of filters in the image is taken and then for each sliding action the products are added. We will obtain a 2-D matrix after convolving a 3-D filter as the output.

Pooling layer: Pooling includes the down-sampling of the features with the goal that we must learn less parameters during training. The number of feature the sliding window skips along the width and height is the stride. Over-fitting is reduced by performing pooling as it reduces the number of parameters.

Batch Normalization layer: During training, if there is any instability in any layer of our neural network, we apply batch normalization to that layer. The output from the activation function is normalized using a batch normalization layer and this being the very first thing that this layer does.

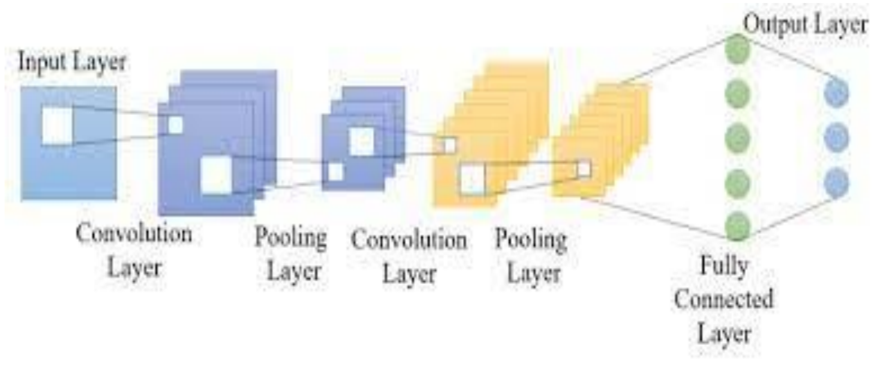


Fig: Basic Architecture of CNN (Source)

EXPERIMENTAL SETUP

AWS_GCP_SETUP:

AWS instance setup:

1. Log in to AWS account and launch console.
2. Launch virtual machine
3. Select AMI if already exists
4. GPU – g3.4xlarge
5. Generate a key pair and download the key in .pem version
6. Launch the instance

GCP instance setup:

7. Launch Google cloud console
8. Launch compute engine
9. Create a project and click create intents
10. Select the following options in the instance creation tab:
11. Zone – US central-a-zone
12. Series- N1
13. Machine type – n1-standard-8(30GB)
14. GPU – nvidia-teslaT4/P4
15. Boot disk – ubuntu, version – 20.04, size – 300
16. Add the public key generated using Puttygen software

For windows,

17. Create a SSH session in mobaxterm
18. Host: External IP from any one of the instances
19. Name: ubuntu
20. Add the private key downloaded from the instances
21. Click ok.
22. Create a folder in the cloud

PyCharm integration:

23. Create a project
24. Tools -> Deployment -> Configuration -> Add SFTP
25. SSH configuration: Host- IP address from instances, name: ubuntu, authorize using the private key pair.
26. Test the connection
27. In mappings, map the remote local path and the cloud folder's path
28. Deployment -> Configuration -> Automatic upload
29. Setup the Interpreter
30. Python interpreter -> SSH interpreter -> Existing server configuration -> Choose the cloud -> click ok.

Data Analysis:

Before processing the data for models, a series of EDA steps were performed to understand the audio signals. One audio file from each subcategory is considered as an example and developed spectrum analysis of the audio files. The audio files are read using ‘librosa’ a python package to visualize the waveform of the respective sounds. Plots for visualizing following spectrums and spectrograms are generated to differentiate and understand sound files.

Fourier_Transform: The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent.

The Fourier transform can be used to construct the power spectrum of a signal by taking the square of the magnitude spectrum. The power spectral curve shows signal power as a function of frequency. The power spectrum is particularly useful for noisy or random data where phase characteristics have little meaning.

STFT: The Short-time Fourier transform (STFT), is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time.

Mel Spectrogram: The Mel spectrogram is used to provide our models with sound information like what a human would perceive. The raw audio waveforms are passed through filter banks to obtain the Mel spectrogram.

MFCCs: 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 is that in the MFC, 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 spectrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

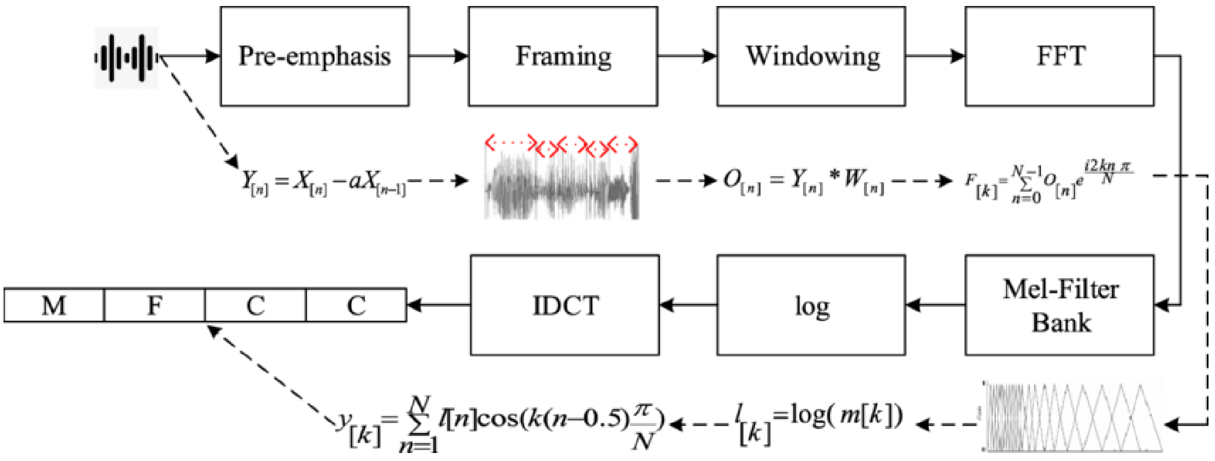

Pre-emphasis
Framing
Windowing
FFT
$Y_{[n]} = X_{[n]} - aX_{[n-1]}$
$O_{[n]} = Y_{[n]} * W_{[n]}$
$F_{[k]} = \sum_{n=0}^{N-1} O_{[n]} e^{\frac{i2kn\pi}{N}}$
M
F
C
C
IDCT
log
Mel-Filter Bank
$y_{[k]} = \sum_{n=1}^{N} l[n]\cos(k(n-0.5)\frac{\pi}{N})$
$l_{[k]} = \log(m[k])$

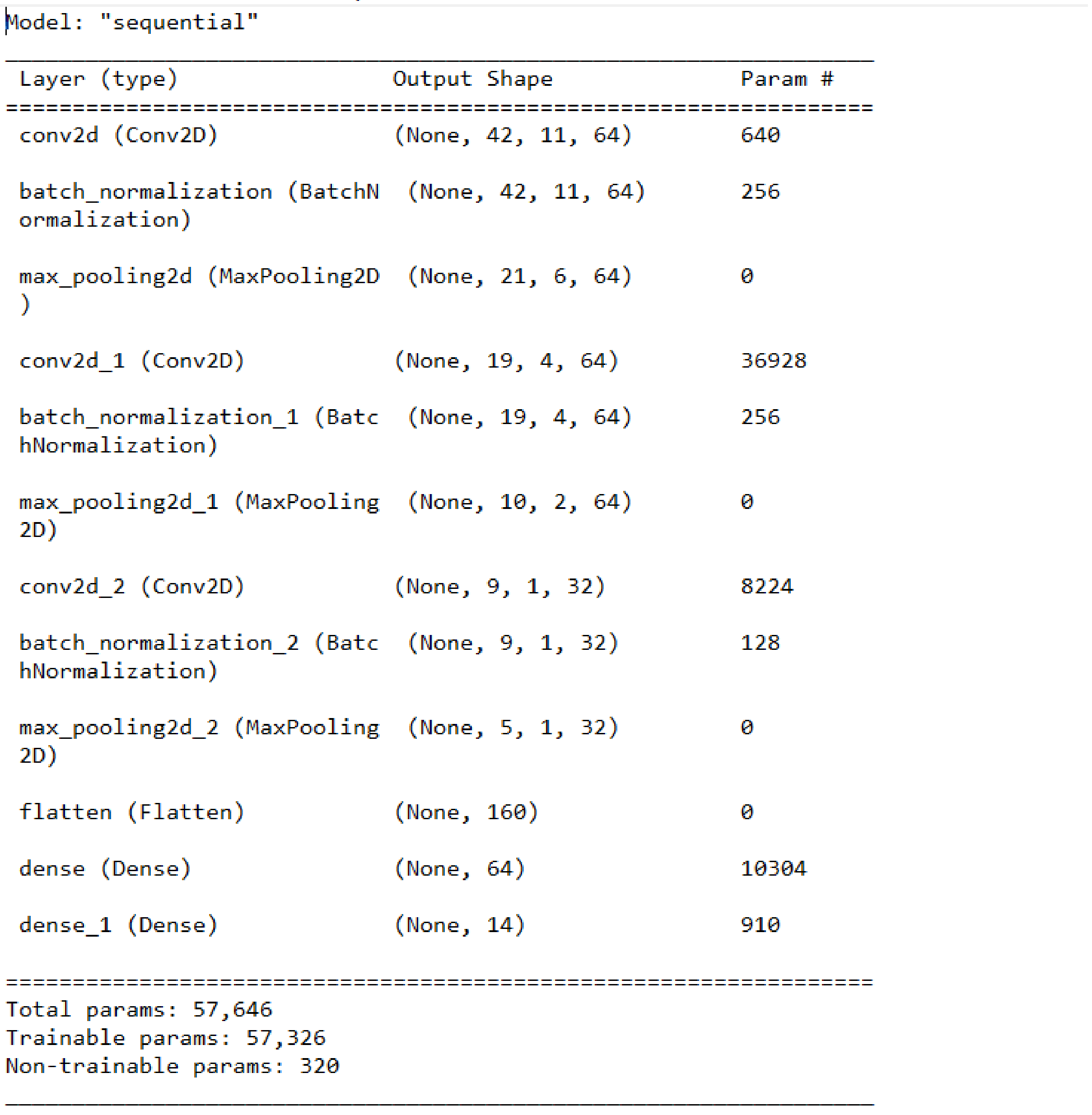
Fig: Processing audio signals to MFCC[6]

Dataset Preprocessing:

The dataset extracted has 14 sub folders titled with keyword containing .wav files. The dataset has been manually segregated into Train and test set to evaluate the model’s performance on the test set. The audio files are preprocessed to create MFCC for each file and the MFCC values with the label and file path to original audio file are stored in JSON file as described in the second chapter of the report.

CNN (Base model):

The base model has three convolutional layers with batch normalization and max pooling layers followed by them. The output from convoluted layers is flattened and fed to a dense layer (with 'relu' activation) and an output layer containing 'softmax' activation. The convolutional layers use activation 'elu'. This model uses a batch size of 32 and learning rate of 0.001 with SGD optimizer and sparse categorical entropy loss. The dataset is separated as train:validation:test set in the ratio of 70:15:15.

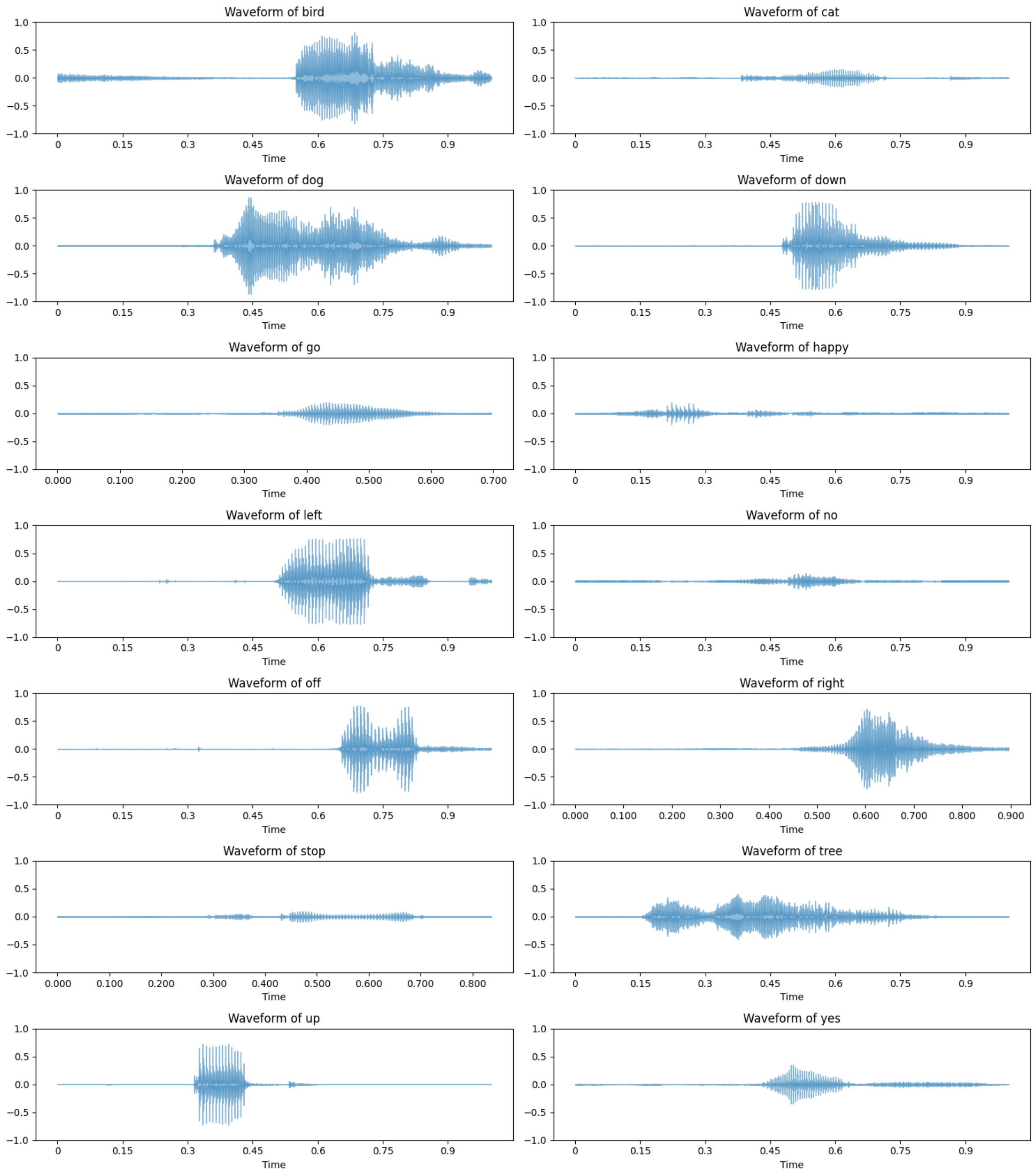


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 42, 11, 64)        640

 batch_normalization (BatchN  (None, 42, 11, 64)       256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 21, 6, 64)        0
 )

 conv2d_1 (Conv2D)           (None, 19, 4, 64)         36928

 batch_normalization_1 (Batc  (None, 19, 4, 64)        256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 10, 2, 64)        0
 2D)

 conv2d_2 (Conv2D)           (None, 9, 1, 32)          8224

 batch_normalization_2 (Batc  (None, 9, 1, 32)         128
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 5, 1, 32)         0
 2D)

 flatten (Flatten)           (None, 160)               0

 dense (Dense)               (None, 64)                10304

 dense_1 (Dense)             (None, 14)                910

=================================================================
Total params: 57,646
Trainable params: 57,326
Non-trainable params: 320
_________________________________________________________________
```

Fig: Model summary of CNN base model

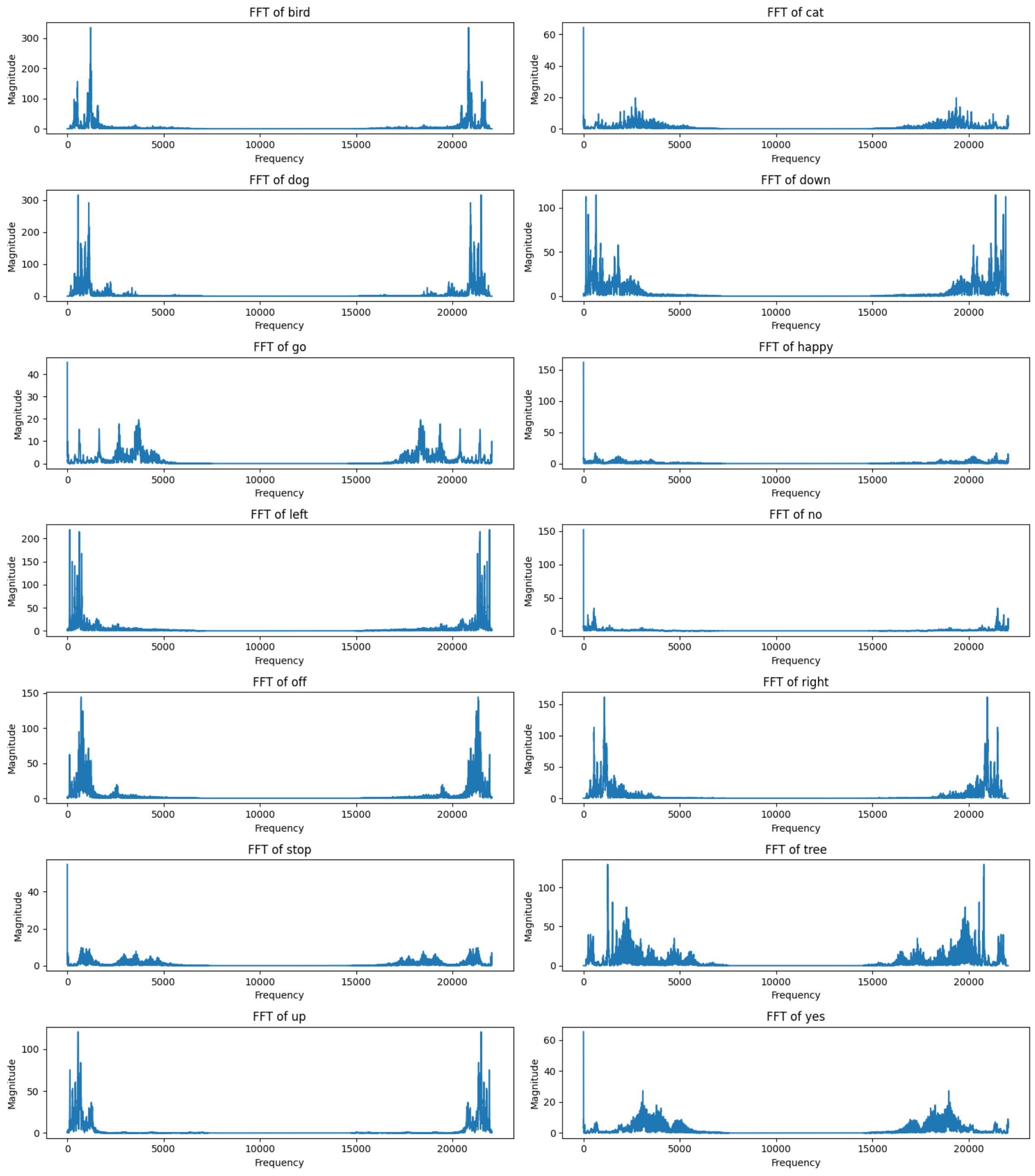
RESULTS

Plot of Waveform of each subcategory:



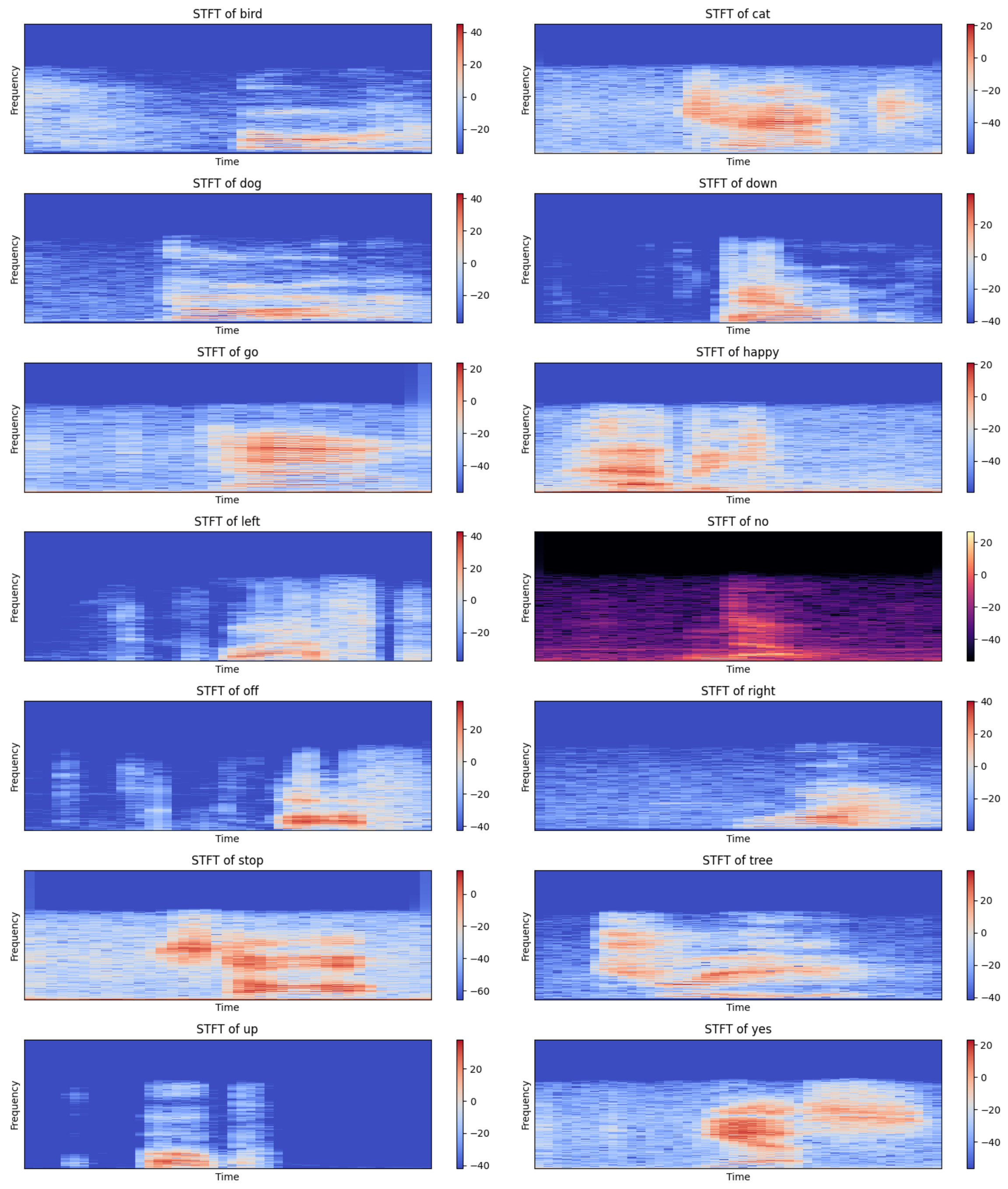
The above waveforms helps in understanding the utterances of each word in the form audio signals.

FFT of each subcategory:



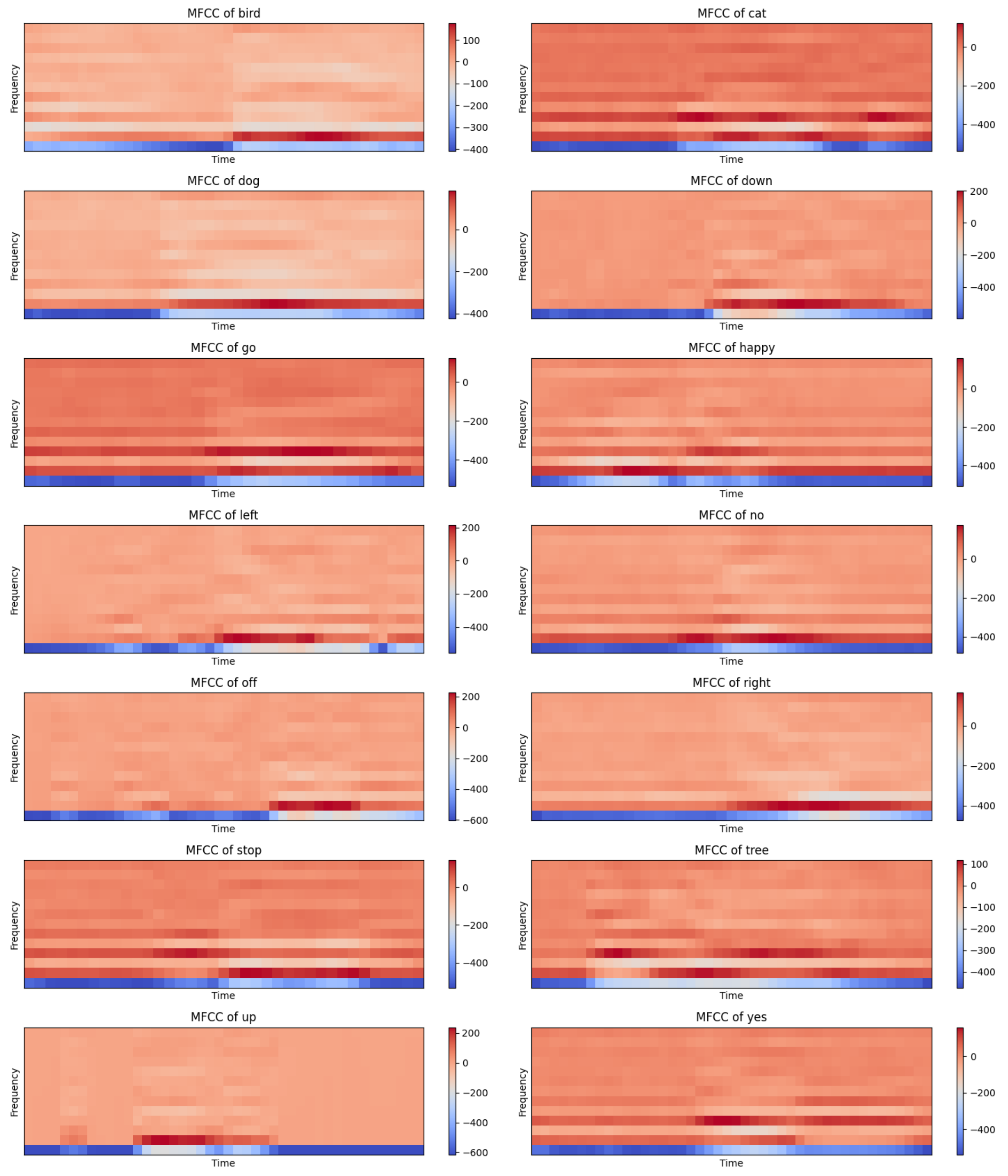
The above FT plot helps in understanding each element of an audio signal. This is a decomposed version of periodic sounds into sum of sine waves oscillating at different frequencies. This helps in visualizing a waveform from time domain to frequency domain. In short it shows the snapshots of all the elements of a waveform.

STFT:



To understand how things, change in time a STFT version of waveform is visualized. STFT computes FFT at different levels of frequency and magnitude using fixed frame size. This helps in preserving the time information too. This in short gives a spectrogram including time, frequency and amplitude.

MFCCs:



Looking at the above MFCCs, we can understand that it helps in capturing the timbral/textural aspects of sound. It also helps in analyzing the audio file in frequency domain. It approximates the audio files similar to human auditory system.

CNN model:

CNN model on training on 50 epochs resulted in accuracy of 87% on the test set. The accuracy and loss evaluation plots shows that the model has been training better on the train data as we can see a smooth curve. When it was validated based on validation set there has been a fluctuations in the loss and accuracy values which shows the model might be overseeing the data.

```
250/250 [==============================] - 1s 2ms/step - loss: 0.3955 - accuracy: 0.8791

Test loss: 0.39549198746681213, test accuracy: 87.9107117652893
```

Fig: Training of model

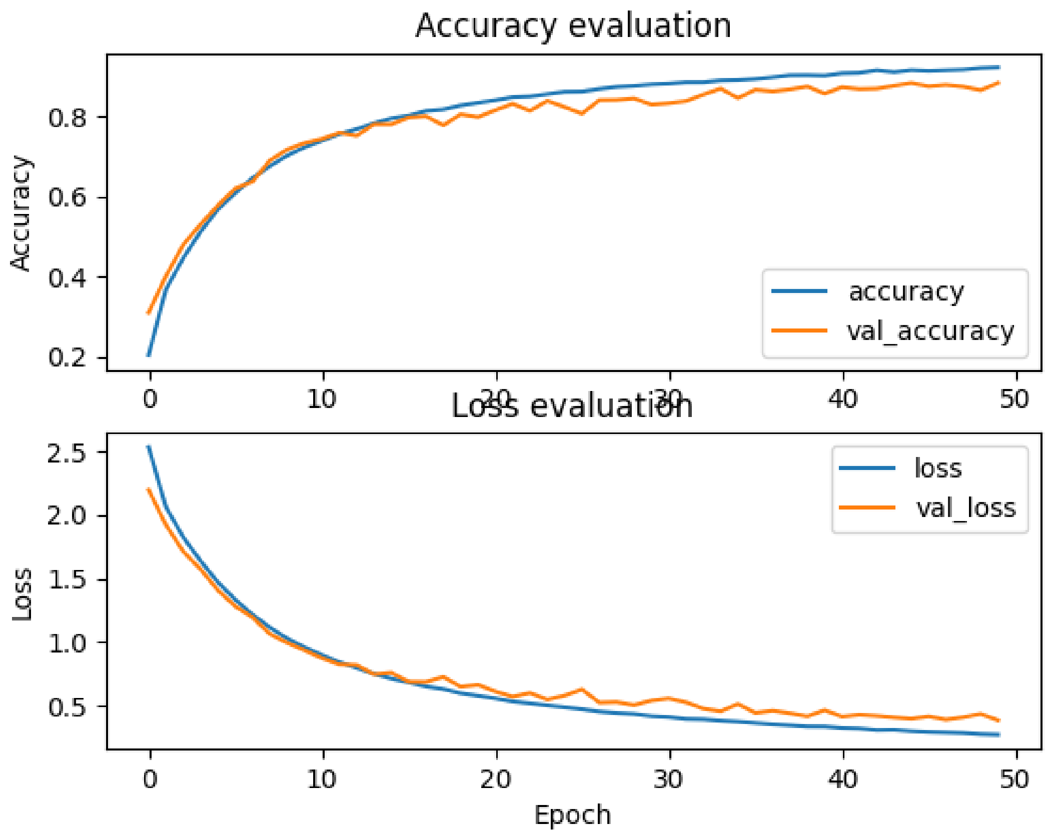


Fig: Train and validation set evaluation

CONCLUSION

The CNN+LSTM model was picked as the best model based on the accuracy. And as we discussed the model performed well on test set since the files are from similar voice modulations. Both the models used in the project doesn’t have major difference in their performances. The table below shows the performance of models in train and test set.

Model	Train accuracy	Train loss	Test accuracy	Test loss
CNN	0.91	0.30	0.87	0.39
CNN+LSTM	0.97	0.06	0.905	0.40

Considering the above table, the CNN+LSTM model was picked to be the best one and it’s performance on test data was 90% similar to the training phase.

Future Scope:

The attempt on training Wideresnet and VGG19 models failed. The future scope of the project can involve usage of deep pretrained models by handling the data preprocessing part much better. And finding a test data with significant modulation differences in the words uttered would help in finding a better and accurate model.

CODE PERCENTAGE

My code part contains 340 lines of code including python scripts for data preprocessing, analysis, and base model. Considering the codes inspired from exam train scripts of ML2 (function definitions), reference paper codes and official documentations of tensorflow, librosa the following will be my code from internet percentage.

Code percentage = 150-100 / (150+190) *100 = 14%

REFERENCES

1. Kaushik, Baij. (2020). A Hybrid Technique using CNN+LSTM for Speech Emotion Recognition. International Journal of Engineering and Advanced Technology. 9. 1126-1130. 10.35940/ijeat.E1027.069520.
2. Tsalera, E.; Papadakis, A.; Samarakou, M. Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning. J. Sens. Actuator Netw. 2021, 10, 72. https://doi.org/10.3390/ jsan10040072
3. Lim, Wootaek, Daeyoung Jang, and Taejin Lee. "Speech emotion recognition using convolutional and recurrent neural networks." 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA). IEEE, 2016.
4. Hajarolasvadi, Noushin, and Hasan Demirel. "3D CNN-Based Speech Emotion Recognition Using K-Means Clustering and Spectrograms." Entropy 21.5 (2019): 479.
5. Zeng, N., Zhang, H., Song, B., Liu, W., Li, Y., Dobaie, A.M., 2018. Facial expression recognition via learning deep sparse autoencoders. Neurocomputing 273, 643–649
6. https://www.kaggle.com/code/venkatkumar001/terminology-of-audio-data-augmentation?scriptVersionId=91620568