Speech Command Recognition

Report by

Adina Dingankar

The George Washington University

Author Note

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

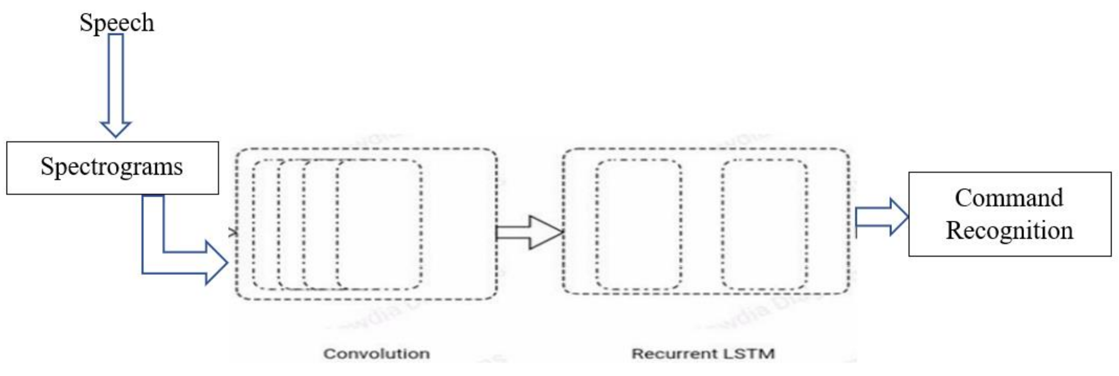
TABLE OF CONTENTS

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)

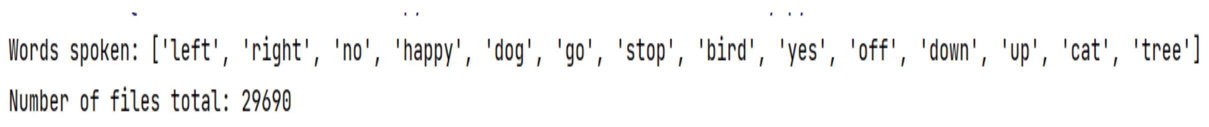
Problem Statement:

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 fine tune the base model and add LSTM layer. After finding the best model I have tested the model performance on an unseen test data.

Information on Algorithm development:

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.

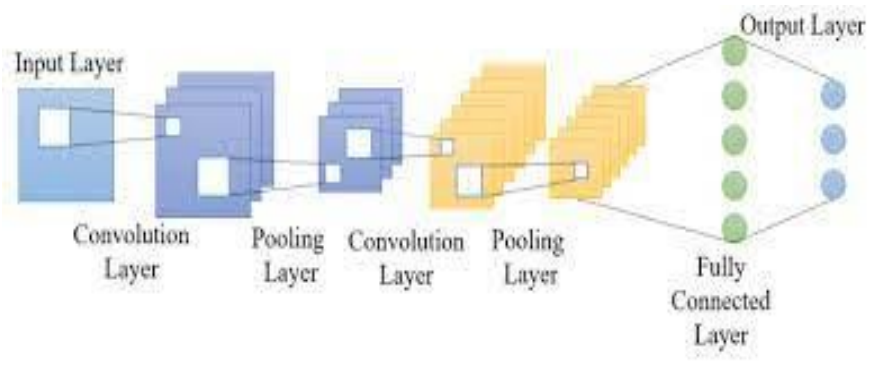


Fig: Basic Architecture of CNN (Source)

LSTM:

Recurrent Neural Network is a neural network in which the previous steps output is fed as input to the current step. But these recurrent neural networks have a limitation that they can look back only a few steps. For understanding problems such as speech recognition, a system is required to store and use context information[5]. A LSTM is a type of recurrent neural network that learns order dependencies in sequence prediction problems are known as LSTM networks. To rectify the vanishing gradient problem, endless efforts were employed, and LSTM is one such solution. The speech signal is continuous in the time domain, so that each frame function only represents the emotional characteristics in a single frame. LSTM increases the information between adjacent frames which helps to reflect temporal continuity of the features. Therefore, LSTM obviously supports speech recognition.

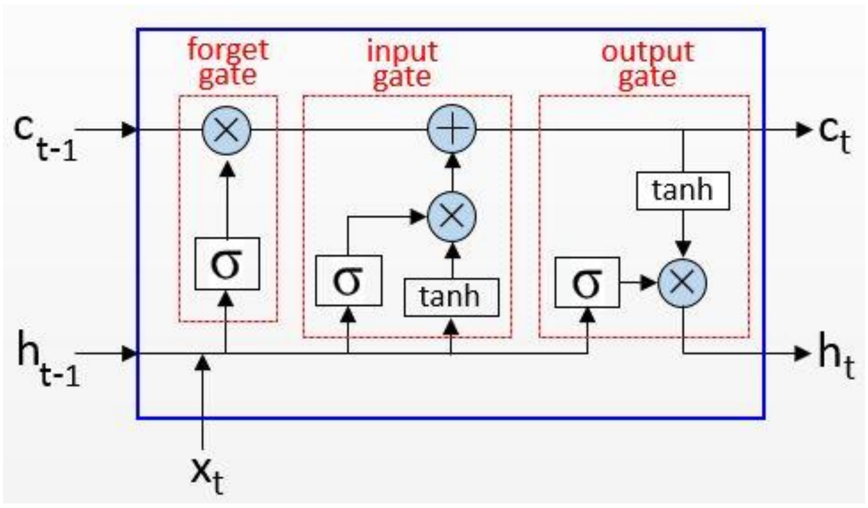


Fig: LSTM architecture (source)

CNN+LSTM model:

A convolutional neural network being a feed-forward network, filters spatial data whereas the recurrent neural network (LSTM) feeds data back into itself. Thus, recurrent neural networks are better suited for sequential data. A convolutional neural network can perceive patterns across space, LSTM can see them over time. Since speech signal is sequential, so LSTM is best suited for speech processing.

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,

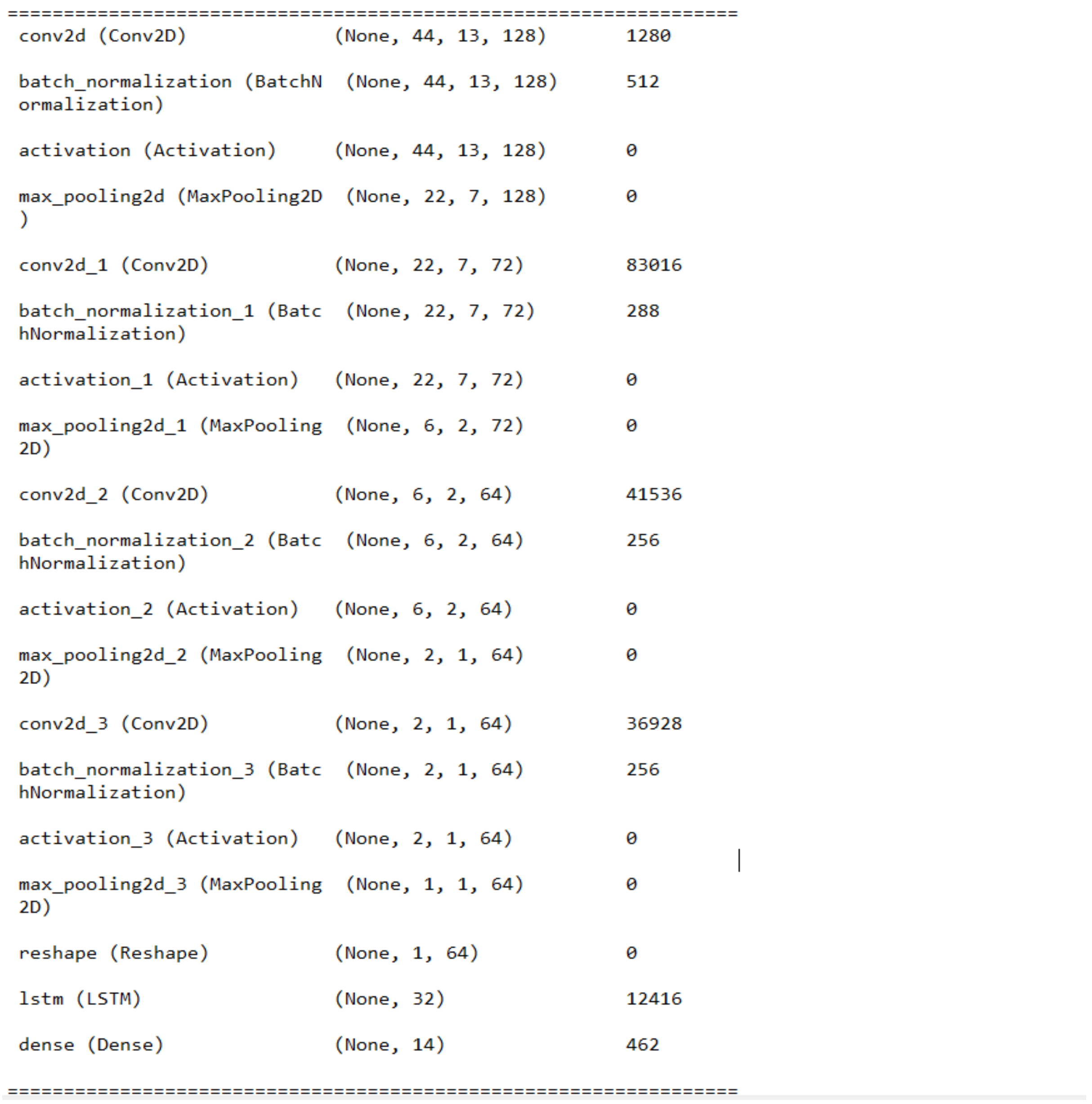
17. Create a SSH session in mobaxterm
18. Host: External IP from any one of the instances
19. Name: ubuntu
20. Add the private key downloaded from the instances
21. Click ok.
22. Create a folder in the cloud

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.

CNN+LSTM:

This architecture is like the base model with an update in convolutional layers along with a LSTM layer stacked on it. It has four convolution layers; LSTM layer and a dense layer is taken as the model architecture. Batch normalization layer, activation layer and max pooling layer are next to each convolution layer. In the convolution layer, rectified linear units (ReLU) activation functions were used. A 3 * 3 kernel size is employed for the convolution layer and a 2 * 2 and 4*4 kernel size of max pooling layer. The convolution layers‟ filter sizes vary as follows- filter size of 128for 1st convolution layer, filter size of 72 for 2nd convolution layer, filter size of 64 for 3rd and 4th convolution layers. Then we resize our output before feeding it into LSTM layer as CNN works on 4D while LSTM works on 2D. And finally, we used the dense layer which is used for classification. This model uses a batch size of 64 and learning rate of 0.01 with Adam optimizer and sparse categorical entropy loss. The dataset is separated as train:validation:test set in the ratio of 70:15:15.



```
=================================================================
 conv2d (Conv2D)             (None, 44, 13, 128)       1280

 batch_normalization (BatchN  (None, 44, 13, 128)      512
 ormalization)

 activation (Activation)     (None, 44, 13, 128)       0

 max_pooling2d (MaxPooling2D  (None, 22, 7, 128)       0
 )

 conv2d_1 (Conv2D)           (None, 22, 7, 72)         83016

 batch_normalization_1 (Batc  (None, 22, 7, 72)        288
 hNormalization)

 activation_1 (Activation)   (None, 22, 7, 72)         0

 max_pooling2d_1 (MaxPooling  (None, 6, 2, 72)         0
 2D)

 conv2d_2 (Conv2D)           (None, 6, 2, 64)          41536

 batch_normalization_2 (Batc  (None, 6, 2, 64)         256
 hNormalization)

 activation_2 (Activation)   (None, 6, 2, 64)          0

 max_pooling2d_2 (MaxPooling  (None, 2, 1, 64)         0
 2D)

 conv2d_3 (Conv2D)           (None, 2, 1, 64)          36928

 batch_normalization_3 (Batc  (None, 2, 1, 64)         256
 hNormalization)

 activation_3 (Activation)   (None, 2, 1, 64)          0

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 reshape (Reshape)           (None, 1, 64)             0

 lstm (LSTM)                 (None, 32)                12416

 dense (Dense)               (None, 14)                462

=================================================================
```

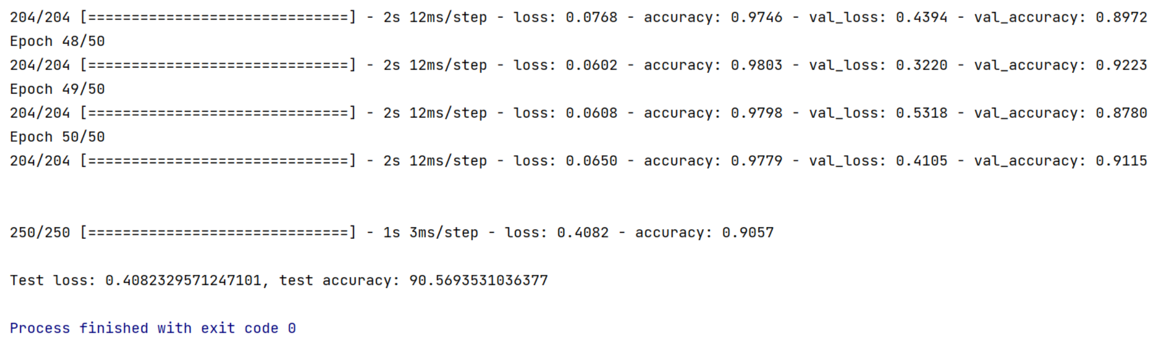
Fig: Model summary for CNN+LSTM model

Out of two models developed, the CNN+LSTM model has better accuracy. Hence the above model is selected to validate its performance on test set.

RESULTS

CNN+LSTM model:

CNN +LSTM model on training on 50 epochs resulted in accuracy of 90% on the test set. Not a much difference from our base model but has shown improvements on the validation 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. Hence this model is chosen as our best model.



```
204/204 [==============================] - 2s 12ms/step - loss: 0.0768 - accuracy: 0.9746 - val_loss: 0.4394 - val_accuracy: 0.8972
Epoch 48/50
204/204 [==============================] - 2s 12ms/step - loss: 0.0602 - accuracy: 0.9803 - val_loss: 0.3220 - val_accuracy: 0.9223
Epoch 49/50
204/204 [==============================] - 2s 12ms/step - loss: 0.0608 - accuracy: 0.9798 - val_loss: 0.5318 - val_accuracy: 0.8780
Epoch 50/50
204/204 [==============================] - 2s 12ms/step - loss: 0.0650 - accuracy: 0.9779 - val_loss: 0.4105 - val_accuracy: 0.9115


250/250 [==============================] - 1s 3ms/step - loss: 0.4082 - accuracy: 0.9057

Test loss: 0.4082329571247101, test accuracy: 90.5693531036377

Process finished with exit code 0
```

Fig Training results

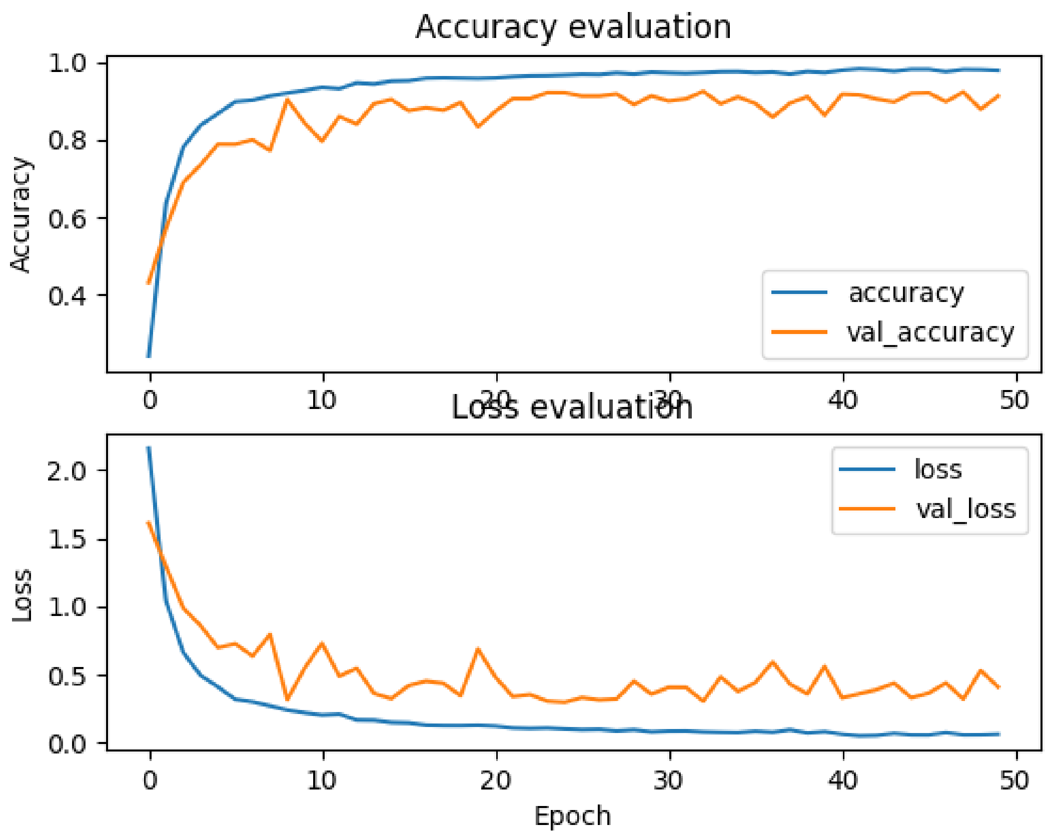
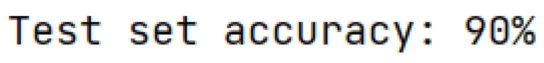


Fig: Train and validation set evaluation

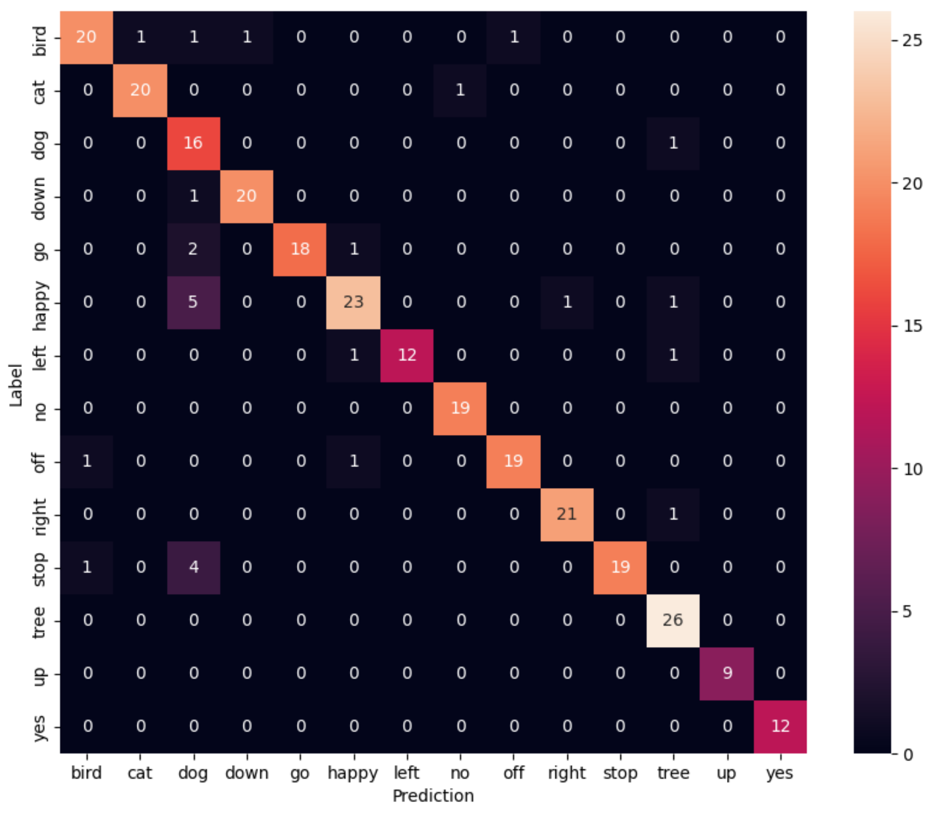
Performance on test set:

The above CNN+LSTM model is chosen as the best model based on the accuracy and loss plots.



```
Test set accuracy: 90%
```

Fig: Test set accuracy



Label \ Prediction	bird	cat	dog	down	go	happy	left	no	off	right	stop	tree	up	yes
bird	20	1	1	1	0	0	0	0	1	0	0	0	0	0
cat	0	20	0	0	0	0	0	1	0	0	0	0	0	0
dog	0	0	16	0	0	0	0	0	0	0	0	1	0	0
down	0	0	1	20	0	0	0	0	0	0	0	0	0	0
go	0	0	2	0	18	1	0	0	0	0	0	0	0	0
happy	0	0	5	0	0	23	0	0	0	1	0	1	0	0
left	0	0	0	0	0	1	12	0	0	0	0	1	0	0
no	0	0	0	0	0	0	0	19	0	0	0	0	0	0
off	1	0	0	0	0	1	0	0	19	0	0	0	0	0
right	0	0	0	0	0	0	0	0	0	21	0	1	0	0
stop	1	0	4	0	0	0	0	0	0	0	19	0	0	0
tree	0	0	0	0	0	0	0	0	0	0	0	26	0	0
up	0	0	0	0	0	0	0	0	0	0	0	0	9	0
yes	0	0	0	0	0	0	0	0	0	0	0	0	0	12

Fig: Confusion matrix of test predictions

The test accuracy is almost like the accuracy of model's performance on training phase. This might be because the test set was generated manually picking files from the train set. Even though the model hasn't seen the test data before we assume that files under same category might have similar modulations on audio format. As we can see on confusion matrix, the model hardly classifies the input category as wrong one.

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 its performance on test data was 90% similar to the training phase.

Future Scope:

The attempt on training Wide Resnet and VGG19 models failed. The future scope of the project can involve usage of deep pretrained models by handling the data pre-processing 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 219 lines of code inclusive of CNN+ LSTM. Out of all the codes for model building the model definition is been referred to the source research papers marked in references.

Code percentage = 100-80 / (100+100) *100 = 10%

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