Speech Command Recognition

Aman Budhraja

Department of ECE, Western University, London, ON N6A 3K7

Abstract

In the past few years there has been a boost in research of machine learning algorithms in the field of audio processing. This has been possible now because of the abundance of audio data available for analysis, specially speech data. This project explores various deep learning algorithms that can be applied to speech processing. In-depth analysis of the feature extraction techniques like Mel-frequency cepstrum coefficients (MFCC), and deep learning algorithms like Convolution Neural Network (CNN), Long short-term memory (LSTM) and hybrid models like CNN + LSTM have been presented. The evaluation of these models have also been provided in the report.

Introduction

Speech recognition using deep learning algorithms has been a struggle for researchers, whether it was getting speech dataset for processing or dealing with huge amount of audio data with limited processing power. This project deals with speech command dataset that includes about 65,000, 1 second long audio file for 30 different words. For simplicity only 4 word, left, up, down and right, out of 30 are used for speech command recognition. By creating a model that can recognize speech commands accurately, one can use it to control robots just with few commands rather than using a complex controller, or build a voice controlled applications.

In the project Mel-frequency cepstrum coefficients (MFCC) features were used to train 3 deep neural networks, CNN, LSTM and a hybrid model(CNN+LSTM). Also one more model, CNN, was trained on raw audio data to recognize the speech command in the audio file.

This report briefs about the algorithm used in ‘Background’ section. ‘Methodology’ section discuss about the approaches used, data pre-processing, tuning of hyperparameters etc. ‘Result’ section provides information on the evaluation of all the models trained, whereas ‘Conclusion’ section concludes the report.

Background

Three deep learning algorithms, CNN, LSTM and CNN+LSTM hybrid, were used to train classifier models that can classify audio/speech files based upon the speech command spoken in the audio files.

Convolution Neural Network (CNN)

CNNs are mainly used in the field of image classification. The basic architecture of CNN includes the following layers :-

1. Input layer – This layer deals with the input data.

2. Feature extraction layer – It is divided into 2 parts :-
    1. Convolution layer – In this layer, the convolution between weights and input neurons is calculated, which generates high order features. Also in this layer all the neurons are collected to all the neurons in the previous layer.
    2. Pooling layer – This layer basically helps in down sampling of the convoluted data. Most common method used is max pooling, in which maximum value in a region is taken.
3. Classification layer – This layer includes 1 to n, fully connected hidden layers and a output layer.

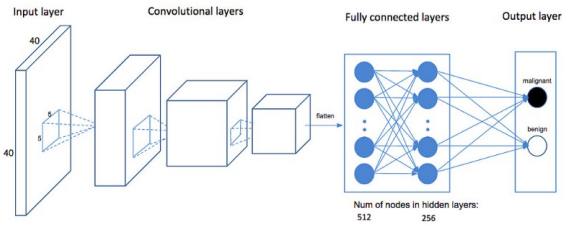

Fig1: Architecture of CNN

For audio processing CNN can be difficult to work with as they are not good at handling time series data. So in order to do data analysis on audio data using CNN, one needs to convert audio data to image data.

Long short-term memory (LSTM)

It is one of the most common type of Recurrent Neural Network (RNN). LSTM has 2 main components: -

- Memory cells – These hold the information from the current and previous steps and/or layers.
- Gates – LSTM has 3 types of gates
    - Input gate – This gate protects the LSTM unit from irrelevant events.
    - Output gate – Releases the information of the memory cell at the output of LSTM unit.
    - Forget gate – For the current LSTM unit to forget previous unit information, this gate is used.

Because of these gates an LSTM model is able to handle vanishing gradient problem in a better way. Hence LSTM are a better option than RNN models for classifying audio data.

Hybrid Model (CNN + LSTM)

In this, the first few input layers are convolution layer, the middle layers are LSTM layers and finally in the end there are fully connected hidden layer and finally an output layer.

This model is theoretically the most apt one, as it takes the advantage of both the CNN and LSTM algorithm, convolution layers helps in creating high order feature values and the LSTM layer in the middle then can deal with the time series part of the audio data.

For calculating the accuracy of the models trained, two accuracy metrics were used : -

Categorical Cross-Entropy

This metric calculates the cross-entropy values for multi-class classification problems.

Categorical Accuracy

This metric calculates the mean accuracy rates across all prediction.

Methodology

Data pre-processing

The audio files in the data-set were down-sampled from 16KHz to 8KHz, the reason being the variation in audio signals is more in lower frequencies (more variation in a signal more features can be extracted from that part of the signal).

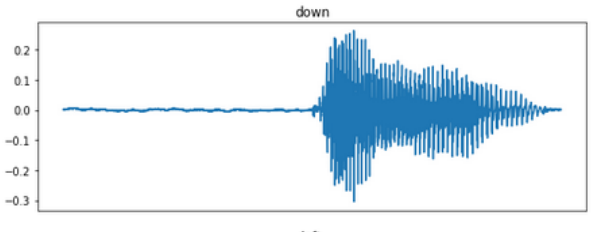

Fig 2: Audio signal in time domain

After down-sampling the audio file , Mel-frequency cepstrum coefficients (MFCC) features for each and every signal is calculated. MFCCs are one of the most widely used feature extraction techniques for audio data. Below steps are followed to find MFCC of an audio signal : -

1. The audio signal is first converted from time domain to frequency domain using fast fourier transform (FFT). This signal obtained after FFT is known as spectrum.

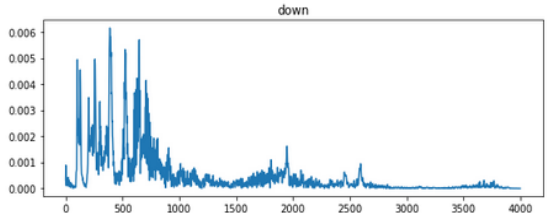

Fig 3: Audio signal in frequency domain

2. Since more variation can be seen for lower frequency, it is best to convert the signal to a log scale so that even for a small amount of change we have a visible output.

   Formula for converting frequency to Mel scale,

   M(f) = 1125*Ln(1+ f/700)

3. The the signal is passed through a filer bank known as Mel scale filter bank. This filter bank can have 26 – 40 filters, 26 being a standard.

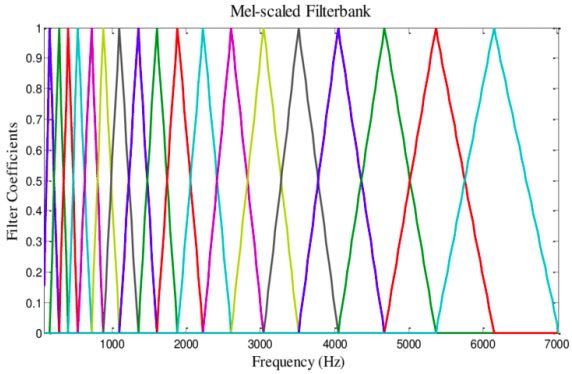

Fig 4: Mel-scaled Filterbank

   Formula for calculating the spacing between filters,

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

   where, M= number of filters we want

   f() = list of M+2 Mel-spaced frequency.

4. The signal received has a lot of correlated data in them as the filters in the Mel-spaced filter bank tend to overlap over one another. To remove this, Discrete Cosine Transform technique is used on the audio data from 26 filters.

5. Finally audio data from 13 out of 26 filters are kept the rest are discarded as they do not contain any useful data.

After extracting features the MFCC features were normalized using the min-max technique and one-hot encoding was used for unrelated categorical data.

Hyper-parameter tuning

For all the models trained, below hyper-parameters were tuned using brute force,

- Number of layers
- Number of neurons in a layer
- Learning rate
- Dropouts

The value for rest of the hyper-parameter were either set to default or given an arbitrary value. The best model chosen after tuning the hyper-parameters was the one with the lowest value for 'loss' on validation dataset.

Validation

Complete data-set was randomized before splitting into train and test dataset. The dataset was divided into 80% train dataset and 20% test dataset and the model was validated on 10% of the train dataset.

Callbacks

- **Earlystopping**

  This callback monitors a particular parameter of the model, for example validation loss. If the parameter does not provide a desired output the model stops training.

- **TensorBoard**

  This callback helps in generating loss and accuracy graphs of the models trained.

Results

The code was developed in python 3.0 using the following libraries: -

- Tensorflow – for machine learning
- Librosa – for audio processing
- Python speech features – for MFCC
- Pickle – for saving preprocessed audio data and for saving trained models

Tuned hyperparameter ranges

Model	No. of layers			No. of neurons	
	CNN	LSTM	Hidden	CNN/ LSTM	Hidden
CNN	1,2,3	-	1,2,3	32,64,128	32,64,128
LSTM	-	1,2,3	1,2,3	32,64,128	32,64,128
CNN+ LSTM	1,2,3	1,2,3	1,2,3	32,64,128	32,64,128
CNN(with raw audio data as inpu)	1,2,3,4	-	1,2,3	8,16,32,64	64,128,256

Learning rate = [0.001, 0.0015, 0.002]

Dropouts = [0.2, 0.3, 0.5]

Learning rate and dropout range was similar for all models trained

Selected models description

Model	No. of layers			No. of neurons	
	CNN	LSTM	Hidden	CNN/ LSTM	Hidden
CNN	3	-	2	64	128
LSTM	-	2	4	64	64
CNN+ LSTM	3	1	1	CNN-64 LSTM-16	128
CNN(with raw audio data as inpu)	4	-	2	[8, 16, 32, 64]	[256, 128]

Learning rate for the models is 0.002

Droupout for all the models is 0.3

Accuracy of the models

The models were evaluated on 20% of complete dataset. Also the models were optimized on 'categorical-crossentropy', loss function, which can also be used as an accuracy measure. Another accuracy measure used is 'categorical-accuracy'.

CNN

Model	Dataset	Loss	Categorical-accuracy
CNN	Train	0.94	0.57
	Validation	0.99	0.55
	Test	0.99	0.55

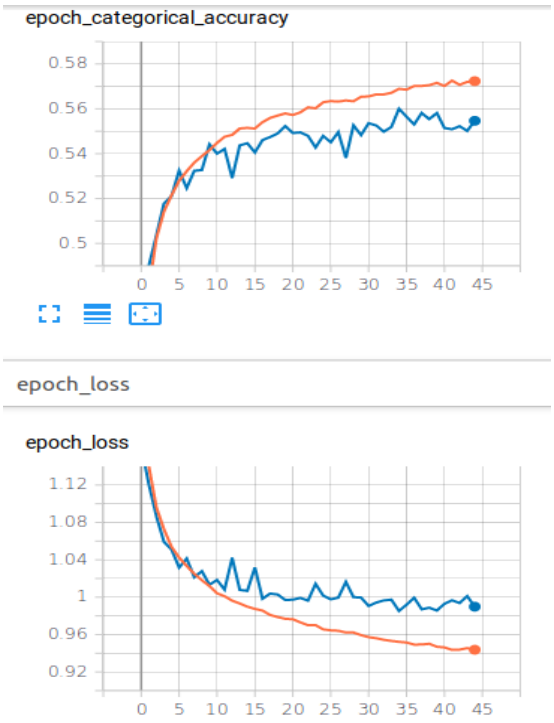

Fig 5 : Accuracy and loss graph for CNN Model

CNN model seems to be overfitting the dataset. The model is not that accurate (only 55% accuracy on test dataset). Also the model took a really long time to converge (about 45 epoch).

LSTM

Model	Dataset	Loss	Categorical-accuracy
LSTM	Train	0.83	0.61
	Validation	0.89	0.59
	Test	0.89	0.59

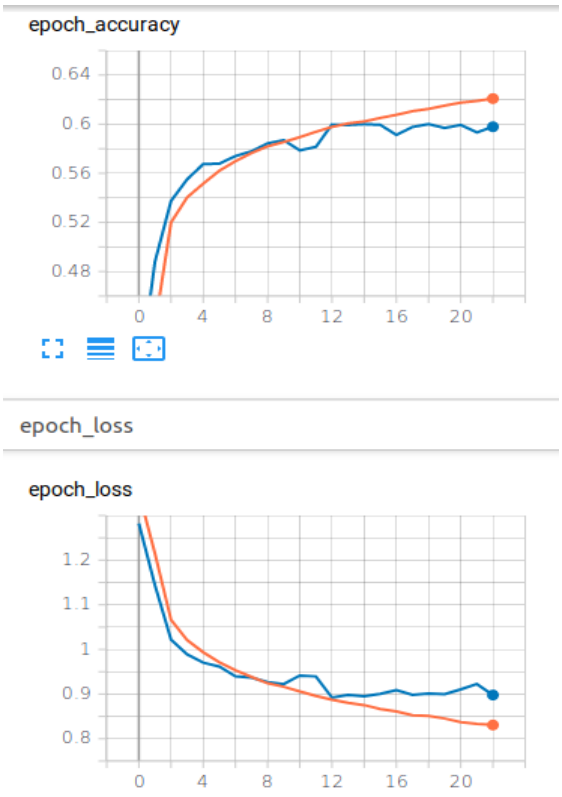

Fig 6 : Accuracy and loss graph for LSTM Model

LSTM model is also overfitting the dataset. Even though the model is more accurate than CNN model, it is still not good for classification (only 59% accuracy on test dataset). The model took a less time to converge (about 25 epoch).

CNN + LSTM

Model	Dataset	Loss	Categorical-accuracy
CNN + LSTM	Train	1.03	0.52
	Validation	1.04	0.52
	Test	1.03	0.53

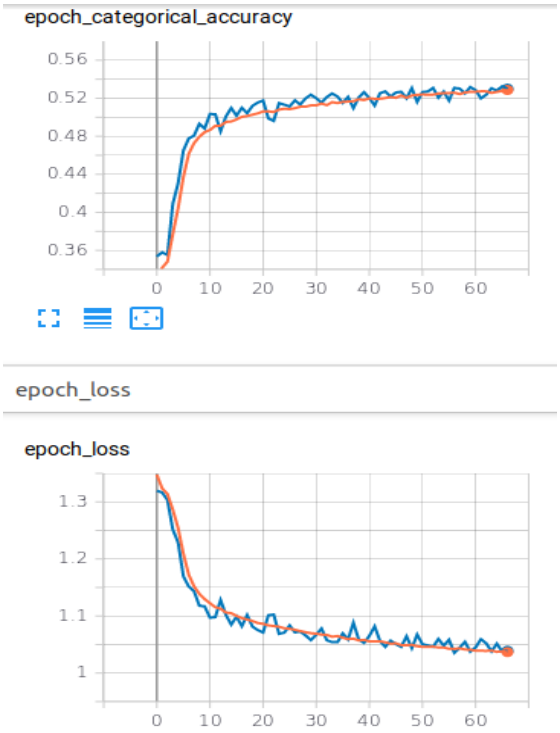

Fig 7: Accuracy and loss graph for Hybrid Model

Hybrid model is not overfitting the dataset. The model has accuracy worst than CNN model (only 53% accuracy on test dataset). The model took the longest to converge (about 66 epoch).

CNN (Raw audio data as input)

Model	Dataset	Loss	Categorical-accuracy
CNN (raw audio data as input)	Train	0.1	0.96
	Validation	0.25	0.93
	Test	0.21	0.93

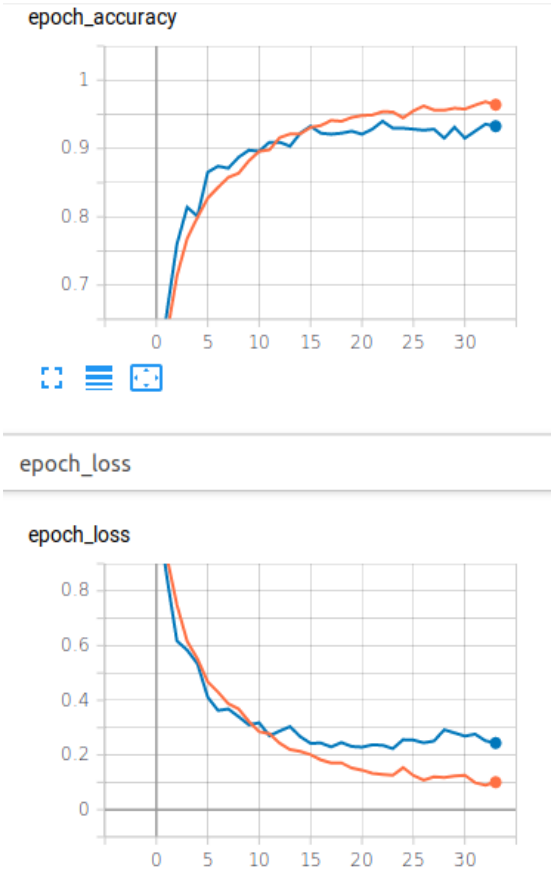

Fig 8: Accuracy and loss graph for CNN Model (Raw audio data as input)

Hybrid model is overfitting the dataset a little bit. The model has accuracy is good enough for classification (93% accuracy on test dataset).

Conclusion

In this project 4 deep neural network models, 2 CNN, 1 LSTM and 1 Hybrid(CNN+LSTM), were trained on speech command dataset, which can classify an audio file based upon the speech command spoken in audio file.

From the results provided it can be following can be concluded: -

- Even after tuning the hyperparameters the models overfit the dataset
- Out of all the models CNN+LSTM took the longest time to converge (about 66 epochs) and even then the accuracy of the model was good.

- CNN model trained with raw audio data performed the best out of all the models, with almost 93% accuracy.
- If a model needs to be trained on only few speech commands then raw audio data can be used.
- For models to be trained on speech recognition then probably MFCC and other audio processing techniques should be used so as to decrease training time of the model (raw audio data had more than 1 million trainable parameters for just 4 words and the training time was more than the models trained with MFCC features)

References

- Aurelien Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, CA: O'Reilly Media, 2017
- The University of Western Ontario. (2019). Data Analytics Foundations. [Online]. Available: https://owl.uwo.ca
- Audio dataset : Google [Online]. Available: https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html

- Keras Library Document. [Online]. https://keras.io/
- Ren Tianping. Application of speech recognition technology [J]. Henan Science and Technology, 2005.
- iang Ming Hu, in the Yuan Baozong, Lin Biqin. Neural networks for speech recognition research and progress. Telecommunications Science,1997,13(7):1-6.