Classifying Audio using Neural nets

In this module we will discuss how to classify audio signals by using Deep Neural Networks.It was already proven that Deep Neural networks will work well in case of Computer Vision and NLP and gave good results.Now we will try to classify audio signals by using Convolution Neural Networks.

I have built an Audio Classifier Neural Network using Tensorflow framework and I will discuss the process in following steps

1. Data Collection and Analysing Data.
2. Data Preprocessing
3. Defining a model
4. Evaluating results
5. Conclusion

Data Collection and Analysing Data:

Initially I have collected Data from “Tensorflow Speech Recognition Challenge” conducted in kaggle and the dataset contains around 64,727 audio files which contains human voices based on different classes such as “Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", and "Nine”.The data was captured in a variety of formats, and then converted to a 16-bit little-endian PCM-encoded WAVE file at a 16000 sample rate. The audio was then trimmed to a one second length to align most utterances. The audio files were then screened for silence or incorrect words, and arranged into folders by label.An Inside look into data was shown in Fig1.

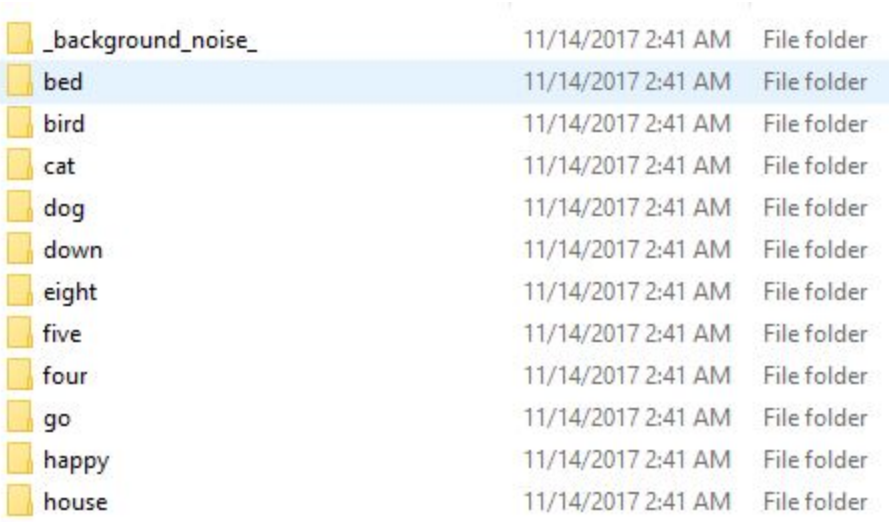

Fig 1.Dataset

Data Preprocessing:

After doing a lot of research on how to feed audio data into Deep neural networks.I found a way to convert audio data into spectrogram(an Image) which can easily fed into Convolution Neural networks.

For converting audio data into spectrogram we consider following steps:

- Converting audio data into Frequency-Time domain by Fourier Transformation
- Converting Frequency-Time domain into spectrograms
- Later passing this spectrograms to Mel-Frequency filter

Converting audio data into Frequency-Time domain by Fourier Transformation:

Initially audio data will be in the form of fig3 in which we cannot get any kind of intuition or any kind of features.So we apply Fourier Transformation on the audio data which gives Frequency-Time domain for audio data.Fourier function typically in fig2

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

Fig 2[1]

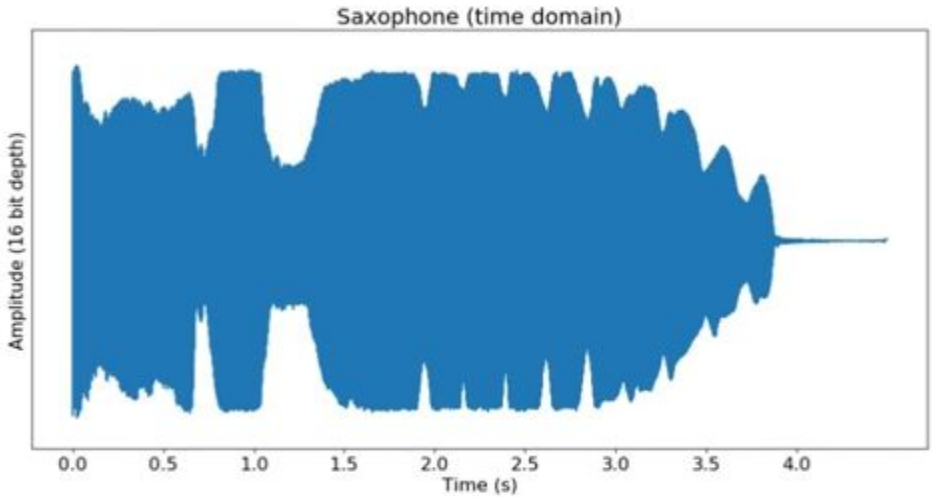

Fig 3[1]

Converting Frequency-Time domain into spectrograms:

We later convert frequency-Time domains into spectrograms based on the frequency at particular second from which we can extract features from.Converted graph will be in the form of fig 4.Fig 5 shows spectrograms at particular seconds and bright lines represent frequency is at high rate.They will be stacked upon one another to feed into our convolutional neural network.

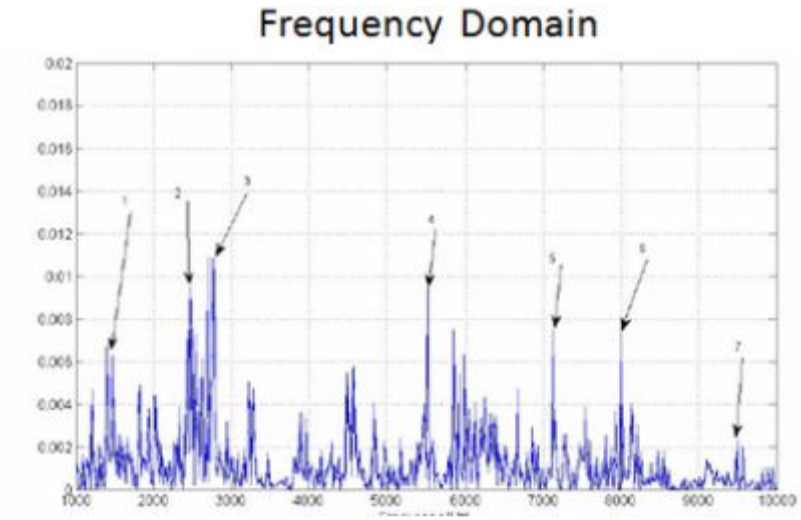

Fig 4[2]

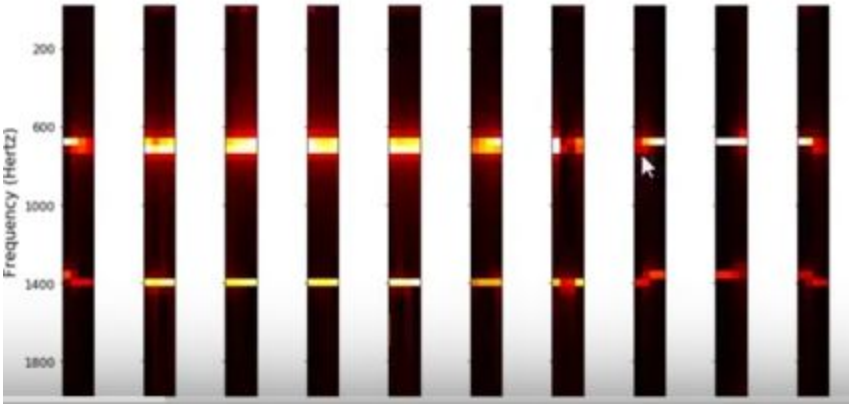

Fig 5[1]

Later passing this spectrograms to Mel-Frequency filter :

From early 2000s it was known that the best way a machine can understand audio signals is through MFCCs[4]

MFCCs are commonly derived as follows:

1. Takes the spectrogram of the signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

This Mel-Frequency spectrum returns an array of integers for the wave file which can be fed into convolution layers.

For preprocessing the entire process I have gone through several libraries such as sound file,sound and Librosa in python.After doing a lot of research I found **librosa** is good library for preprocessing audio files which have inbuilt functions for every step Mentioned above.

Defining Model:

The model I have used is typical CNN and the architecture of the model is shown below in fig6.

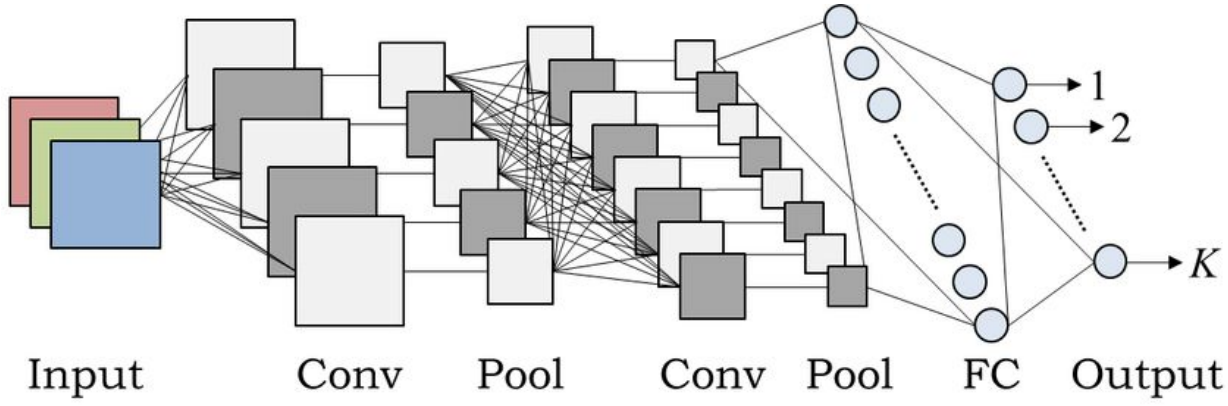

Fig 6[3]

Layers used in the model:

1. Three Convolution layers
2. One Flatten Layer
3. Two Dense Layers,one fully connected and other for getting probabilities.

The Convolution layer:

Convolutional layer are the major building blocks used in convolutional neural networks.A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

Why have I chosen Convolutional neural network:

After the entire preprocessing is done for the audio data it gives a spectrogram typically an image format.It was proven that CNNs will work well for image data and I even tried LSTMs for solving the same use case but found that CNNs show better results than RNNs.

Softmax Layer:

Softmax calculates a probability for every possible class. In our case, it returns a 10-dimensional vector representing the probability of each class.

Tensorflow Model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 126, 42, 64)       640
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 126, 42, 64)       0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 124, 40, 32)       18464
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 62, 20, 32)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 60, 18, 16)        4624
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 30, 9, 16)         0
_________________________________________________________________
flatten_8 (Flatten)          (None, 4320)              0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               553088
_________________________________________________________________
dense_17 (Dense)             (None, 10)                1290
=================================================================
Total params: 578,106
Trainable params: 578,106
Non-trainable params: 0
_________________________________________________________________
```

Fig 7

Evaluating results:

I trained the model around 19000 samples and tested against 4000 samples.I got training accuracy around 95% and testing accuracy around 90% which shows little overfitting but that isn't our cause.

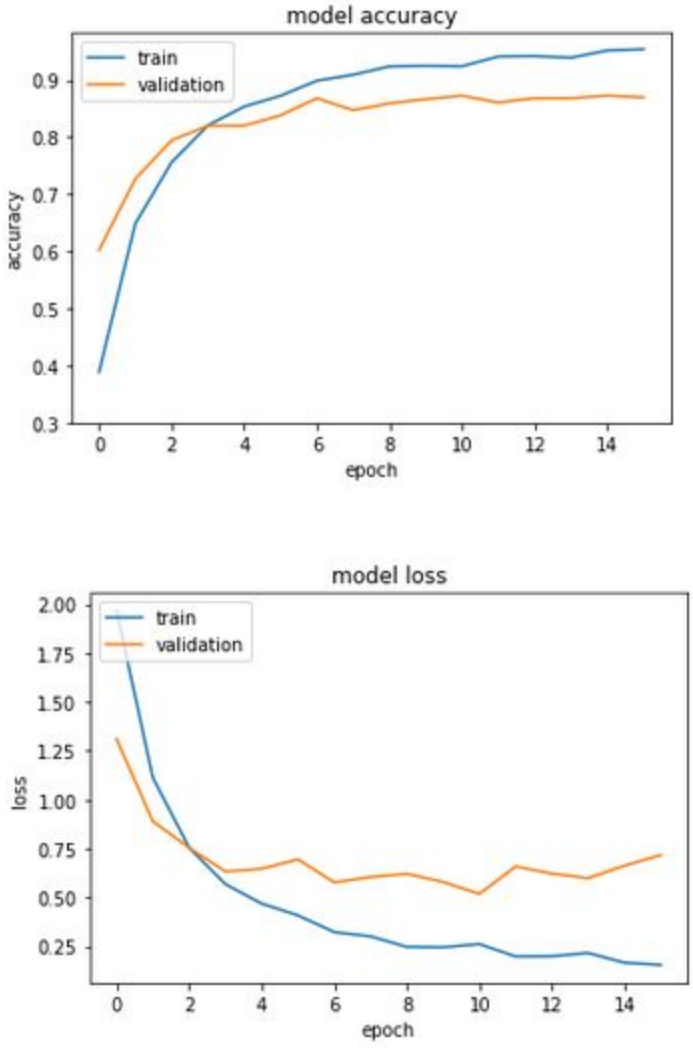

Fig 8

Conclusion:

The model achieved accuracy of 90% which is good and we can achieve better accuracy by fine tuning the model.

REFERENCES

1. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
2. https://www.researchgate.net/post/how_to_convert_time_domain_data_into_frequency_domain_data
3. Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks
4. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/