Speech Recognition

Swapnil Sirsat

February 2021

Introduction

"Okay Google", "Alexa", "Siri", all these calls are not for some human or a biological living being, well not that you don't know that already, this shows that how common and indulged machines have been in our lives lately. Its not that our computer somehow miraculously started to understand our language but some code that we simply know as Speech command recognition today based on some deep learning and neural networks.
The concept of Speech command recognition is based on the same principal of the way we humans learn i.e. getting trained first and then applying it. Here too we use some data to train our device for that and based on that data the model or device tries to predict.

Feed Forward Architecture for Speech command recognition

The main goal of a feed-forward network is to approximate some function f. As the name defines, the flow of information takes place in the forward direction. The architecture comprises of various layers including a input layer and an output layer. If there are 'N' features as an input then the input layer would have 'N' numbers of neurons. Between input layer and output layer there are hidden layers. there could be any number of hidden layers which can be changed accordingly as per requirements.
Feed forward architecture takes specific features as input and based on that This feed forward architecture can be used to develop a speech command recognition system.
The architecture attaches some weights to each neuron in input layer for the corresponding feature and then passes it to the next layer which applies the activation function on the summation of the dot products of the data from input neuron. and this goes on until the output layer. The below figure can show the working of a basic feed forward architecture.

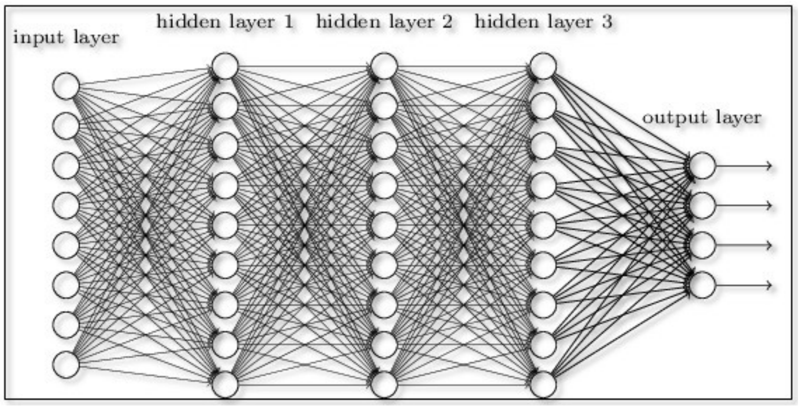

Figure 1: Various layers of a Feed Forward architecture

Distinguishing Various audio commands

Now, as we already know that computer can't understand audio as we do. Thus here we need some kind of numerals which can distinguish the audio commands. going with amplitudes or any such features won't actually help as most of the time amplitude vs time plot looks all the same for all the commands. Therefore we need some better feature for distinguishing between different commands.
In this model we are using MFCC which stands for Mel Frequency Cepstral Coefficient.

What is MFCC?

Mel Frequency Cepstral Coefficient is an feature extraction technique based on cepstral representation of audio. They are excellent tools for getting information out of audio. We would be calculating this MFCCs wiht the help of a predefined function in librosa library.

Feature Extraction and Accuracy

Now as we know differentiating audios just on the basis of amplitude won't be enough. Feeding our model this info would be trivial. To achieve this goal we shall be using MFCCs which as discussed is an excellent tool for getting necessary data out from various. Also on top of that the preprocessing part would add to the accuracy by removing all the trivial information from our audios before we begin the feature extraction part.
Feature extraction part involves getting the required features out of the audios and that would be done by calculation of MFCCs. Though the mathematical calculation to calculate the MFCCs is quite complex we would be using librosa library for this purpose.

Visualization

Audio signals as data are not easy to processed having some factors we know nothing about. Machine doesn't understand human voice or texts as some ordinary human would think. As data analyst, we need a mechanism which would unravel its properties.

Spectrogram : A spectrogram is a visual way of representing the signal strength, or "loudness", of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time.
A spectrogram is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the color or brightness.

In our code, we used matplotlib(python library used for visualization) to get our desired spectrograms. Code for required purpose is provided.

Description

- before we begin going anywhere in our model, we would start by visualization, noting beneficial to the computer itself but it would help us humans to get a rough idea of what our computer would be taking as input and by what means would it be doing the predictions thus helping us to write the best layers for our model
- Initially we would be performing the preprocessing part so that we could remove all the unnecessary noises so that when we perform the feature extraction parts we would only get features for the given data. Here we are performing normalization and preemphasis
- after pre-processing we perform the feature extraction using MFCC
- adding Layers to the feed forward model
  1 **Input layer**: the first layer of our model which takes in our input features of the audio. the number of neurons in this layers are equal to the number of input feature data items. The input neurons aren't really neuron's tho they act as if they are.
  2 **Adding hidden layer**: A Dense layer is added to which each neuron receives inputs from each neuron in previous layer
  3 the activation function for the layers would be **rectified linear unit**. It is a common activation function
  4 **Adding output layer**: the final layer of our model is the output layer which gives out the final output for our feed forward architecture
  5 the activation function in our output layer would be soft max. **Soft max** is applies the rule of "winner takes it all" and thus our output is only one output i.e. for any input the output can only be only one command which can be only one all the commands, soft max would do the job for getting any one 'on' of all output and all else 'off'.
- Back propagation: after getting output while training the model checks for the actual and if prediction was incorrect the model back-propagates such that it adjusts the weight accordingly to get most justified output

Data and Execution

5×80 samples

- of "Forward" , "Back", "Left", "Right" , "Stop"
- of 1 second each
- of sampling rate 16000

Epochs

There are 500 epochs trained in which accuracy gradually increases with each new epoch

Loss function

The loss Function used is Categorical crossentropy. We used the Categorical Cross entropy loss function. Categorical-crossentropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which o

Optimization Algorithm

The Adam Optimization algorithm is used. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Recording and prediction

Recording

I've used Soundevice library to record voices and wavio to save them (either, tomporalily or permanently) , the sampling rate would be 16000 and the duration would be of 1 sec. The Voice recorder accepts audio after a beep sound, thus wait for a beep and then speak up.

Prediction

The libroasa library is used for the conversion of audio to numpy arrays. The library also allows to set the sampling rate accordingly which I've set to 16000. Further for prediction purposes this audio also undergoes the same stages as the training data for preprocessing and feature extraction. and finally the output is predicted using the predict which is originally in the numerical categorical format.

Getting the output which we can understand

We've now the obtained the output which is in the form of zeroes and one which of course we can't understand thus we need to convert it to the original form. for this we've used the numpy.unique funtion which allots the mapping for the original data and the converted data which can be later used for reversetransforming.