# TensorFlow Speech Recognition Challenge

Deepak Sathyanarayanan

**DataSet** - TensorFlow Speech Recognition Challenge ([click here](https://www.kaggle.com/c/tensorflow-speech-recognition-challenge))

**Problem Statement:**

The data set contains 64k audio files, each about 1 second. We’re also given the label for each sample, that is, what’s being said in that one second. There are 31 labels in total:

'right','eight','cat','tree','bed','happy','go','dog','no','wow','nine','left','stop','three','sheila','one','bird','zero','seven','up','noise','marvin','two','house','down','six','yes','on','five','off','four'

Given any audio input, predict the word using the signals/ frequencies extracted

**Where will it be used?**

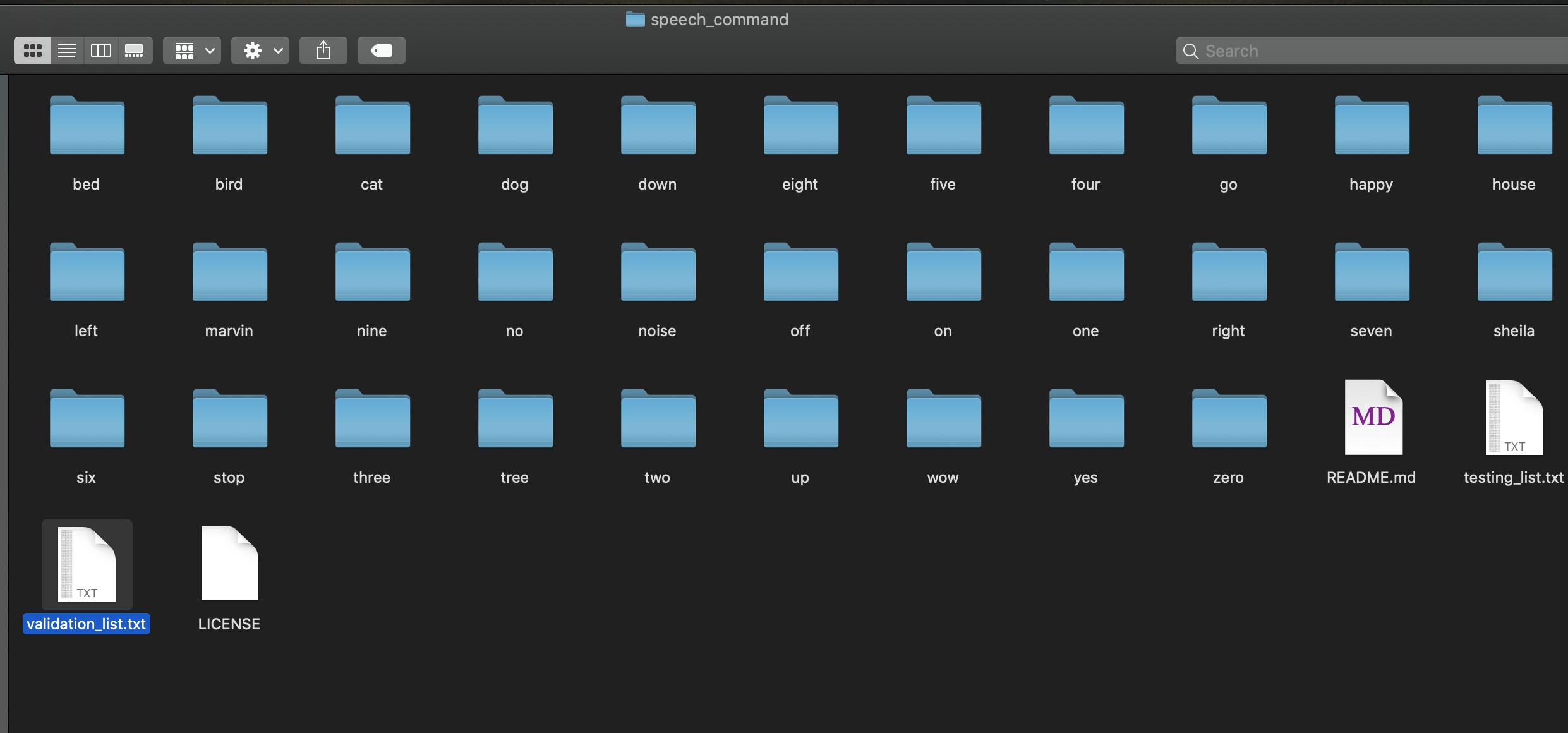
ML can be used to predict the customer’s queries and use the predicted words/ sentences to help the customers get answers for their query.

**About the Data:**

Data is available on World Wide Web. This dataset is provided by Google to use the Speech Commands Dataset to build an algorithm that understands simple spoken commands. By improving the recognition accuracy of open-sourced voice interface tools, it is possible to improve any product’s effectiveness and their accessibility.

The dataset contains 65,000 one-second long utterances of 30 short words, by thousands of different people.

Snapshot of the local repository:



Note: There is also a separate repo which has captured the inputs for some random noises

**Approach:**

First task is to extract spectrograms or MFCC (Mel-Frequency Cepstral Coefficients) out of the raw audio. Python module, Librosa can help in extracting MFCCs

You can break a natural sound wave into its constituent waves at different frequencies (~fourier transform). This is basically what spectrogram tells us: at each point in time what is the intensity of each frequency.

All audio files have 16k sample rate which means they capture up to 8k Hz sound frequency. Mels are bins in which you place certain frequency ranges and using a higher number of mels results in more fine grained spectrograms.

The x-axis is correlated with time and its length for a given sample depends on a few parameters you use for calculating spectrogram, such as hop length.

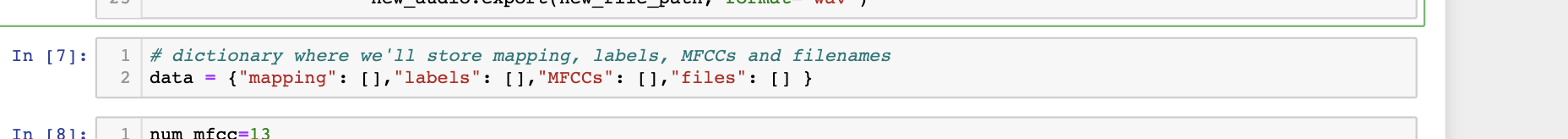
MFCC detailed explanation: <https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd>

Hop\_Length – It is how much we can advance the analysis time origin from frame to frame

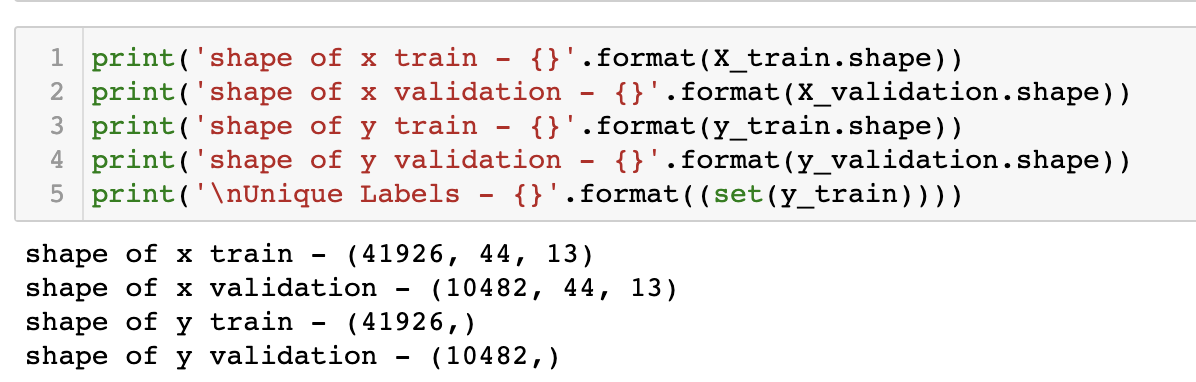
From the MFCCs, we will get numpy array which tells you how strong a certain frequency for a given timestamp is. From there on we can build a DL model on top of it.

Process Involved:

1. Create dataset using input audio files
2. Loop through the repository to extract the labels
3. Loop through the sub-directories to extract the frequencies from the audio files
4. Breakdown the files in “Backgroundnoise” folder and extract consecutive 1 second audio of the files in “Backgroundnoise” folder
5. Create json file with the following inputs from the audio files
   1. Mapping
   2. Labels
   3. MFCCs (Frequencies)
   4. Files (path)

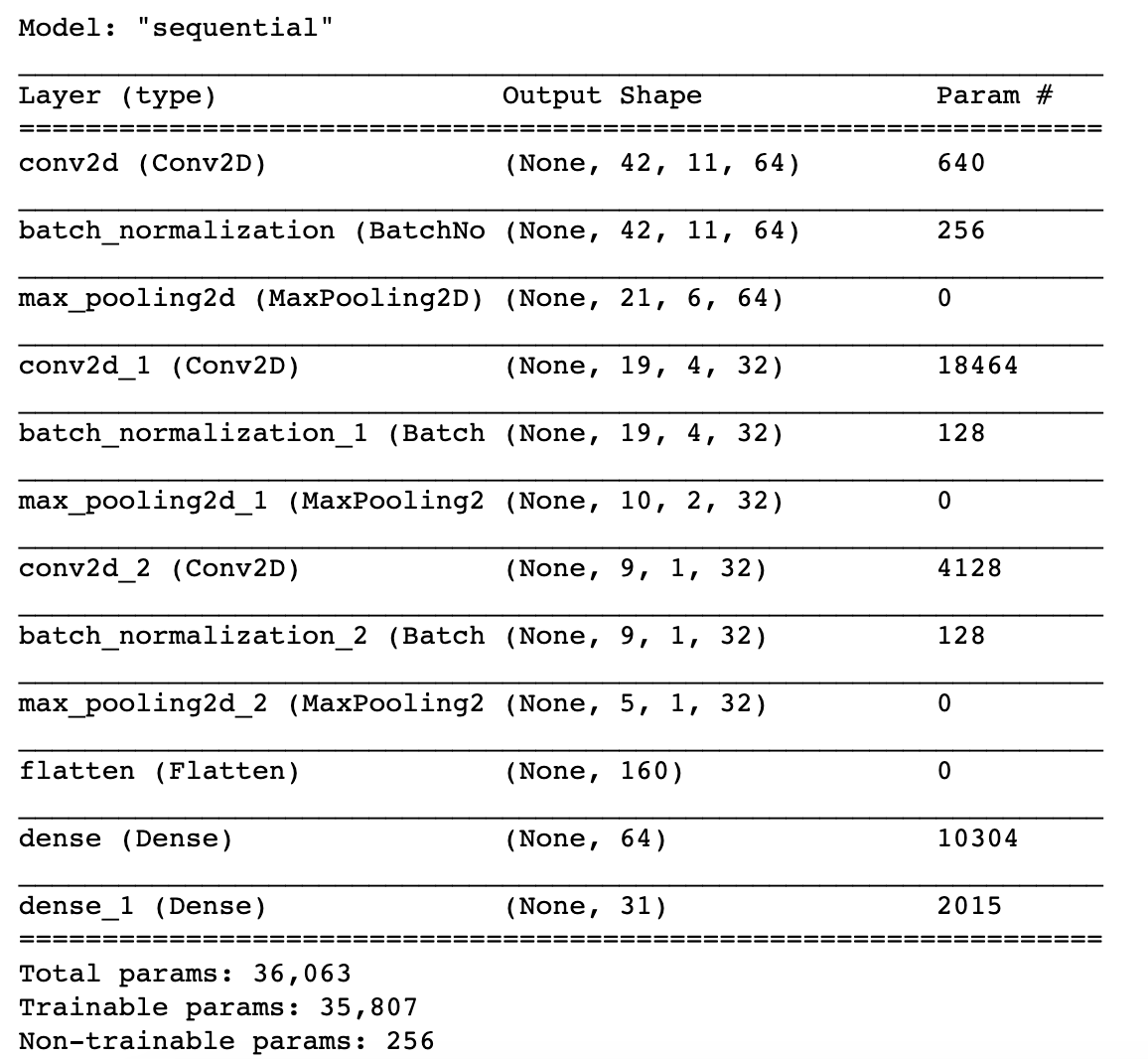


1. Store the json file as the input data for building ML models in-order to predict the audio input
2. Import the model and split it into train/ test dataset

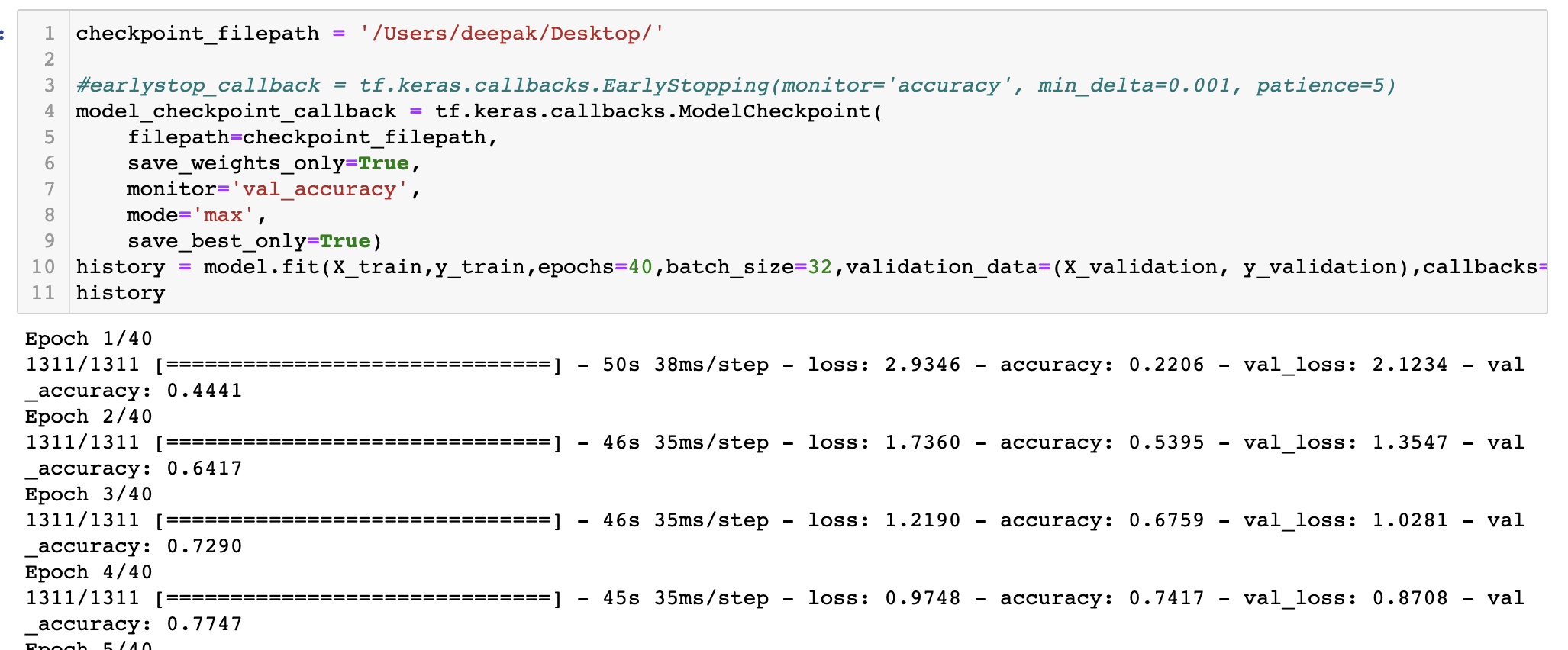


1. Create a sequential layer of deep learning model
   1. 1ST Layer
      1. Convolution Layer
      2. Batch Normalization
      3. Max Pooling
   2. 2nd Layer
      1. Convolution Layer
      2. Batch Normalization
      3. Max Pooling
   3. Flatten Layer
   4. Dense Layer – I
   5. Dense Layer – II

Model Summary

****

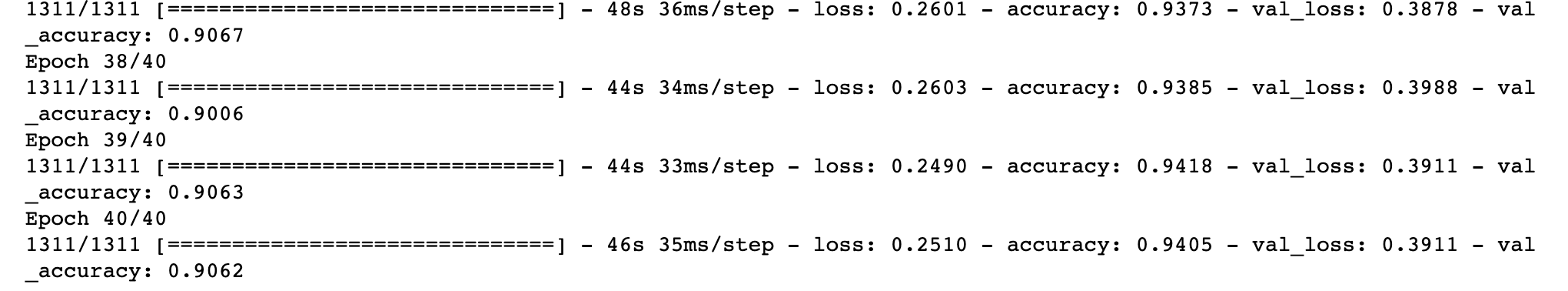
1. Train the model



**.**

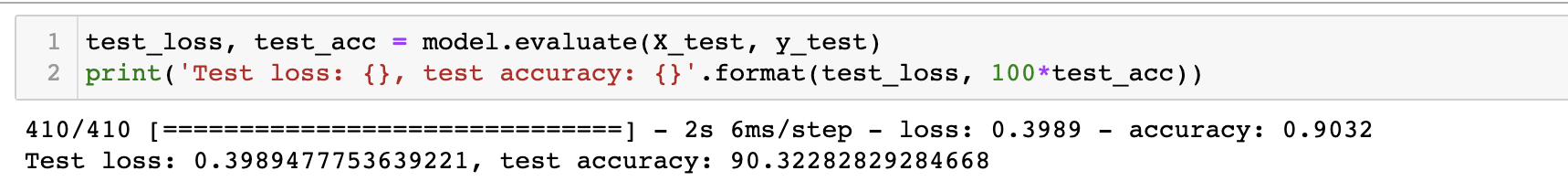
**.**

**.**

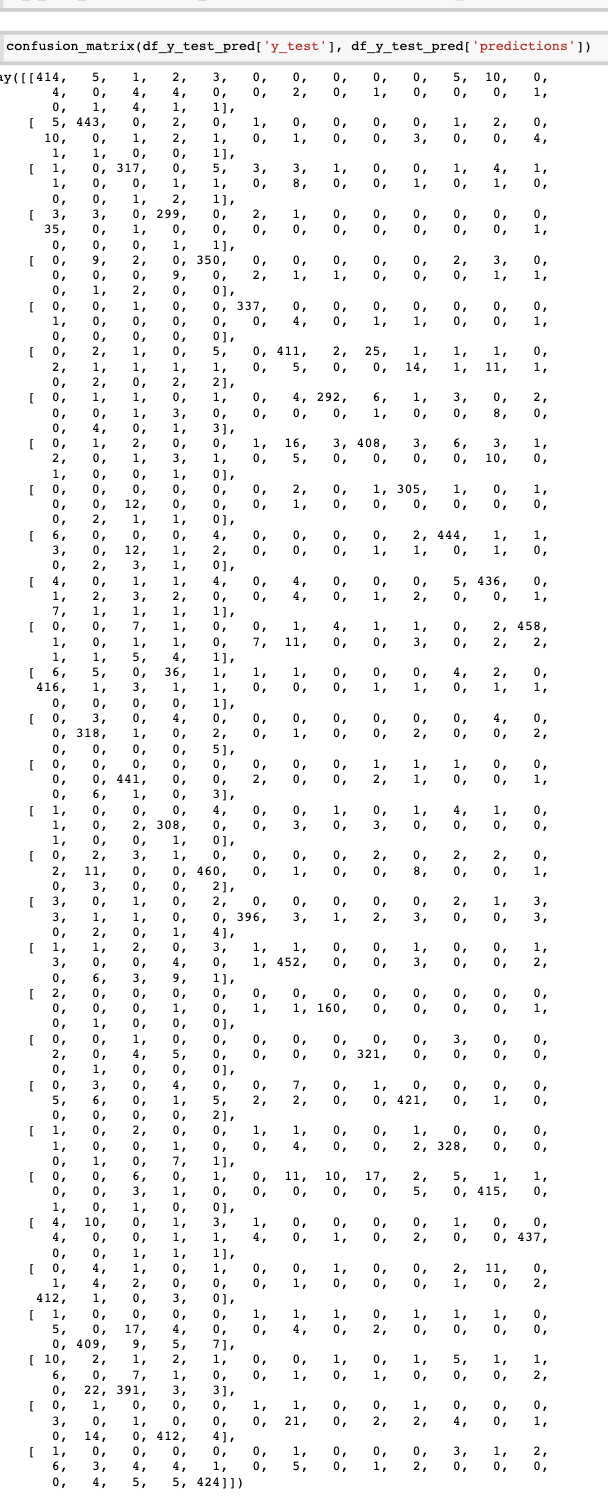


We can see that the training accuracy is 0.9405 and the validation accuracy is 0.9062. Further we can also note that loss is 0.3911

1. On the test dataset the model gave accuracy of ~90%



1. Confusion Matrix



1. Predicting the top 5 probabilities for each audio file

