Capstone Project abdalla Shaaban

Machine Learning Engineer Nanodegree September 27st, 2018

Simple Speech Command Recognition

I. Definition

Project Overview

Within the past few years, voice command and activation has become a popular trend in user experience and design. Voice command allows for technologies to meet specific consumer demands, such as providing a user interface when hands or vision is occupied- a reason 61% of users state why they use voice command- or when one is in the car or on the go – the primary setting for voice usage in 55 percent of instances (Young,2016). According to Google CEO Sundar pichai, 20% of all queries made in 2016 on Google apps and Android devices were voice search (sterling, 2016).

The future looks bright for voice command. Comscore estimates that 50 percent of searches will be conducted by voice by 2020(Young, 2016). While that figure may be highly optimistic, a study of 39 leading SEO experts identified voice usage as one of the top 3 important trends in search (Alameda Internet Marketing, 2016). While these estimates speak to the Internet search industry exclusively, the enthusiasm for voice activation in this industry portends demand for adoption in other industries.

Problem Statement

- The goal is to recognize a simple speech commands, the tasks involved are the following:
  - Download the speech commands data set.
  - Represent audio using Mel-frequency cepstrum
  - Build the network, the network consists of several convolution layer which followed by pooling layer, followed by fully connected layers and finally softmax classifier.
  - Train the model, once the model has been fully trained and evaluated, the weights will be frozen and extracted.

- The final project is expected to be able to recognize the new audio input.

Metrics

- Based on the balanced distribution of the samples to the target labels, The evaluation metric for the model will be evaluated on Multiclass Accuracy.

$$\boldsymbol{accuracy} = \frac{\sum \textbf{true predicted smaples of each label}}{\textbf{test size}}$$

II. Analysis

Data Exploration

- This project will use audios from the publicly available Speech Commands Data Set v0.01 dataset. This is a set of one-second .wav audio files, each containing a single spoken English word. These words are from a small set of commands, and are spoken by a variety of different speakers. The audio files are organized into folders based on the word they contain, and this data set is designed to help train simple machine learning model.

- There are more labels than the labels should be predicted. The labels you will need to recognize in Test are yes, no, up, down, left, right, on, off, stop, go. Everything else should be considered either unknown or silence.

- This table shows how many recordings of each word are present in the dataset..

Word	Number of utterances
- Down	2359
- Go	2372
- Left	2353
- No	2375
- Off	2357
- On	2367
- Right	2367
- Silence	2015
- Stop	2380
- Unknown	2418
- Up	2375
- Yes	2377

Figure 1: How many recordings of each word are present in the dataset

- So the dataset is balanced.

Exploratory Visualization

The data set contains 65000 audio files of a second of 30 short words, pronounced by thousands of different people. The competition consists then of classifying them in 12 categories. The raw recordings do not make it possible to make a classification easily. It is necessary to transform and rework the data.

So, the first action on these data was to seek to display the wave and the spectrogram of certain words, if only to have a more concrete idea of the way in which they are represented.

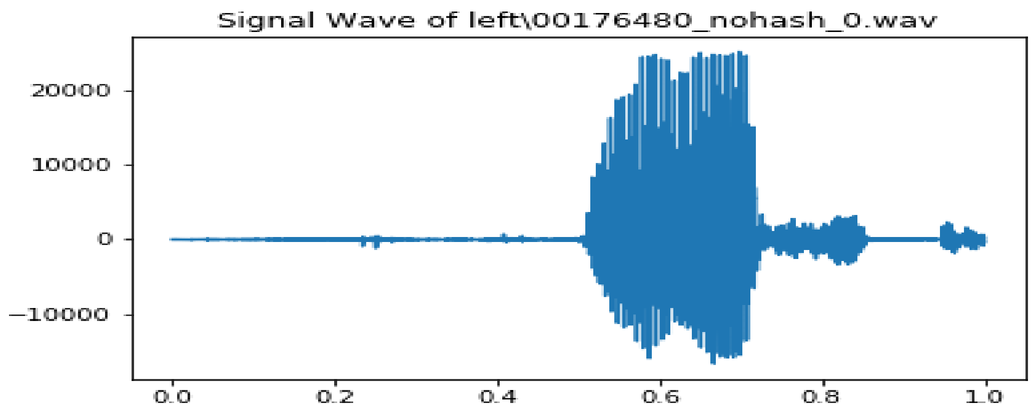

Figure 2: wave of sample\left

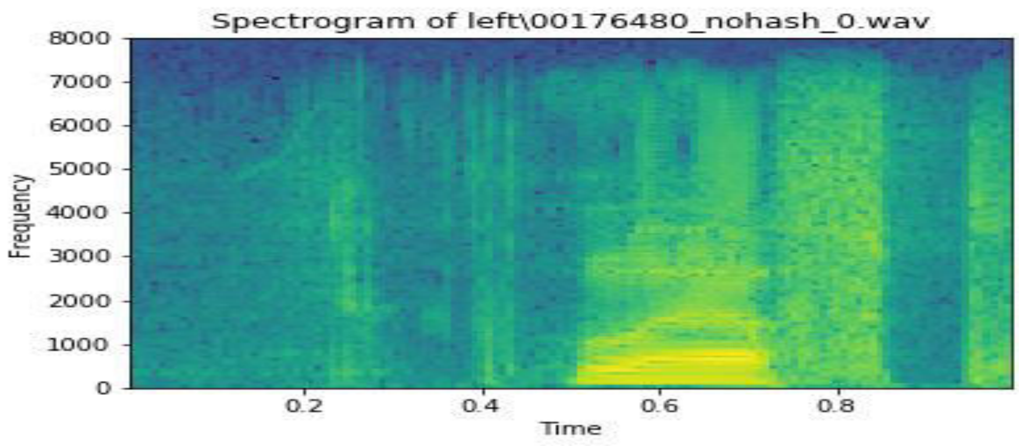

Figure 3: spectrogram of sample\left

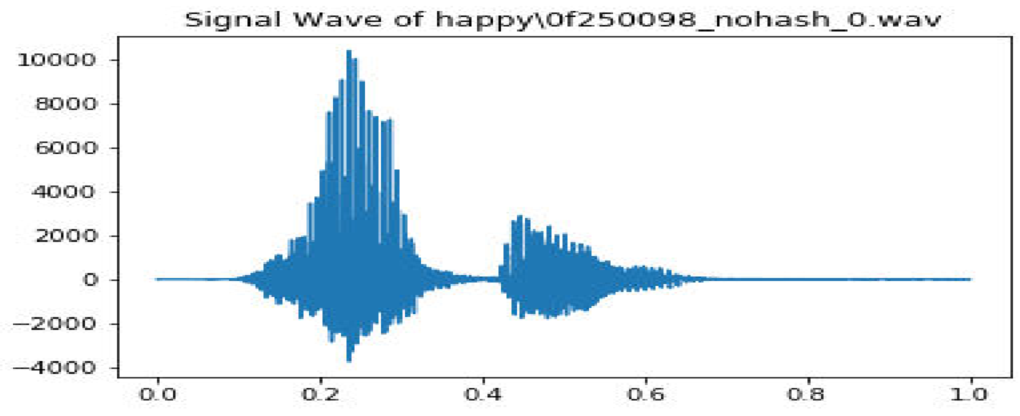

Figure 3: wave of sample\happy

We notice that it is already possible to get an idea of the word pronouns while observing the wave of a word. For example, for "happy", there will be a strong peak at the beginning for the initial 'h'.

We notice silences before and / or after the pronunciation of words.

We have also observed the class distribution, which is relatively homogeneous here, as shown in the graph below; there is no class too over-represented or under-represented, except for the background_noise, that is a class a little apart.

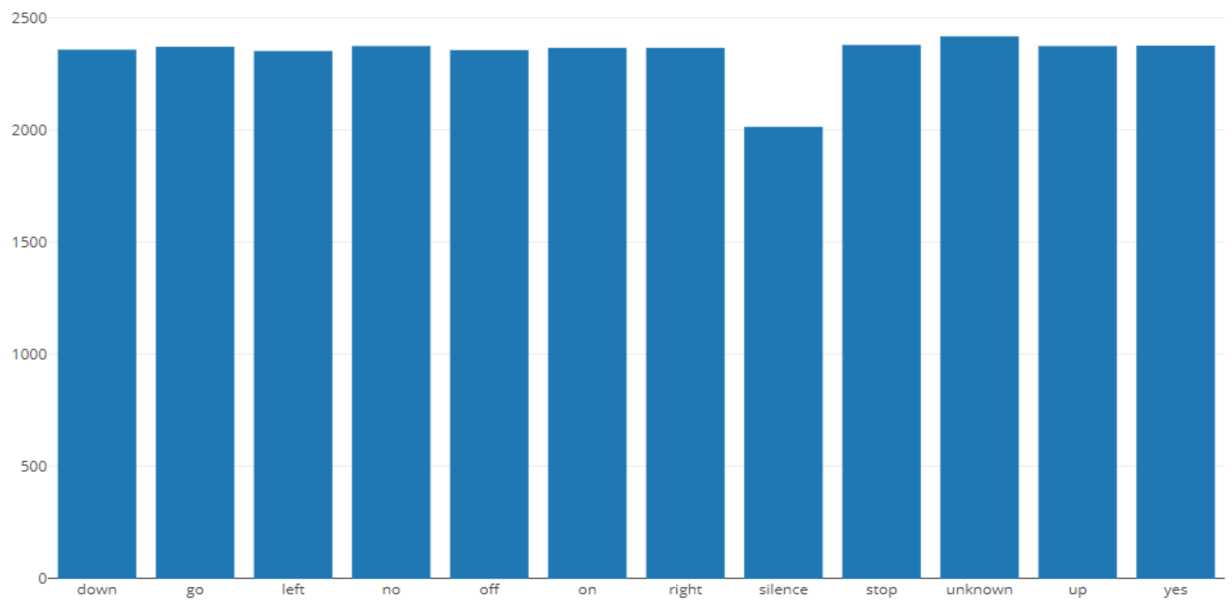

Figure 4: Distribution of classes

The metric used to measure the rate of performance of a model on these data is simply the accuracy rate of the predictions made by this model

Algorithms and Techniques

This is classifier problem; the classifier is a Convolutional Neural Network, they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. It needs a large amount of training data compared to other approaches

Architecture Overview:

A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer.** We will stack these layers to form a full ConvNet **architecture**.
Example Architecture: Overview. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. In more detail:

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- RELU layer will apply an elementwise activation function, such as the max (0, x) max (0, x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image.

Benchmark

I will use the tensorflow audio recognition model [2] as a benchmark. The accuracy of this model is between 85% and 90%

III. Methodology

Data Preprocessing

The preprocessing steps are:

1- Convert the audios to numerical representation “mfcc” for each file.

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

Mel Frequency Cepstral Coefficents (MFCCs) are a feature widely used in automatic speech and speaker recognition.

How to calculate MFCCs.

- Frame the signal into short frames.
- For each frame calculate the periodogram estimate of the power spectrum.
- Apply the Mel filterbank to the power spectra; sum the energy in each filter.
- Take the logarithm of all filterbank energies.
- Take the DCT of the log filterbank energies.
- Keep DCT coefficients 2-13, discard the rest.

Pros of their usage are as follows:

- using of the spectrum of the signal (i.e. expansion by the orthogonal basis of (co)sine functions), which takes into account the “wave” nature of the signal;
- the spectrum is projected onto a special mel-scale, which allows us to extract the most valuable for human perception frequencies;

2- Padding the output vector to become all vectors with the same length.
3- Save the representation of audios for each file in “.npy” file.
4- Concatenate all .npy files for generating training and testing sets use the train_test_split function.

Implementation

The implementation process is the classifier building and training:

1- Load the training data into memory.

2- Define the network architecture and training parameters(batch_size=256,epochs=400,dropout=.2 ,activations functions = 'relu')
3- Define the loss function(categorical_crossentropy).
4- Train the network and save best weights based on validation loss.
5- Save and freeze the trained network.

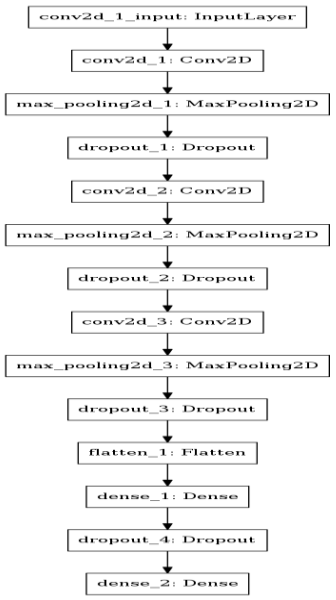

Figure 2: The CNN architecture.

The network consist of input and three convolution layer which followed by Max pooling layer and dropout layer, followed by fully connected layers and dropout layer, and finally softmax classifier

Refinement

In the initial result, the accuracy was around 79%. This was improved upon by using the following techniques:

- Change rmsprop optimizer to adadelta optimizer.
- Add another convolution layer followed by maxpooling layer and dropout layer
- Decrease the dropout probability to .2.
- Change the activation from tanh to relu.
- Increase the number of epochs.

The final model was derived by training in an interactive fashion, adjusting the parameters. The final model has an accuracy of 91.9%.

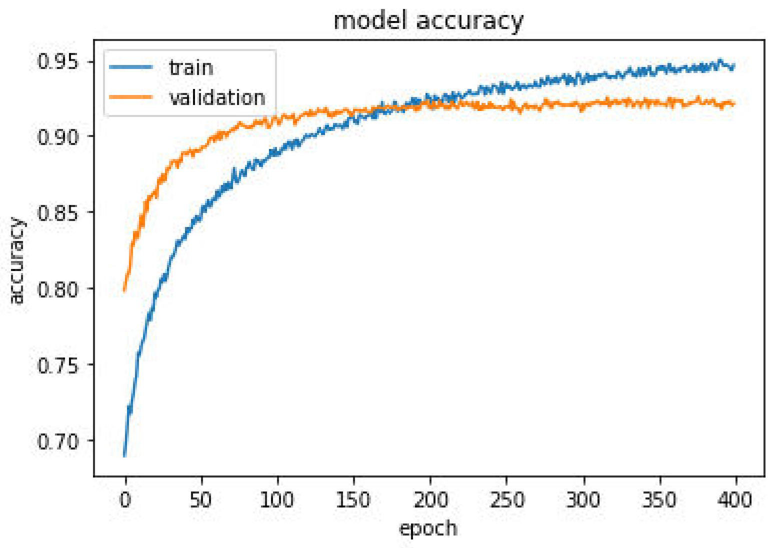

Figure 3: A plot of the training/validation accuracies.

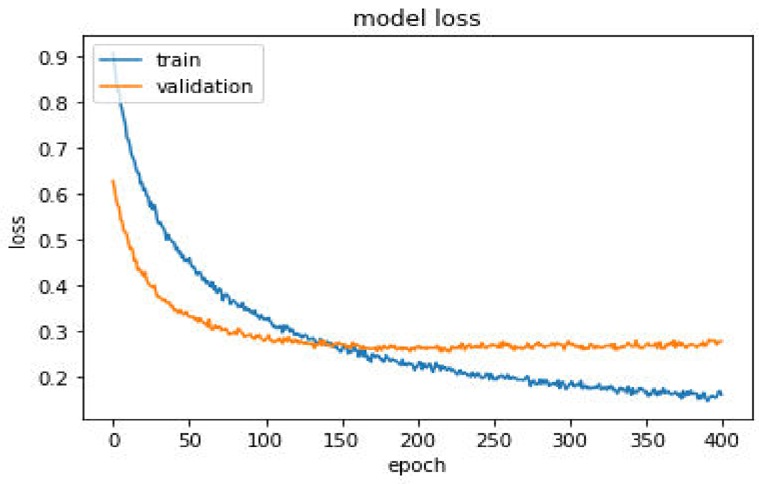

Figure 4: A plot of the training/validation losses

IV. Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The final architecture and hyperparameters were chosen because they performed the best among the tried combinations.

For a complete description of the final model and the training process, refer to Figure 4 along with the following architecture:

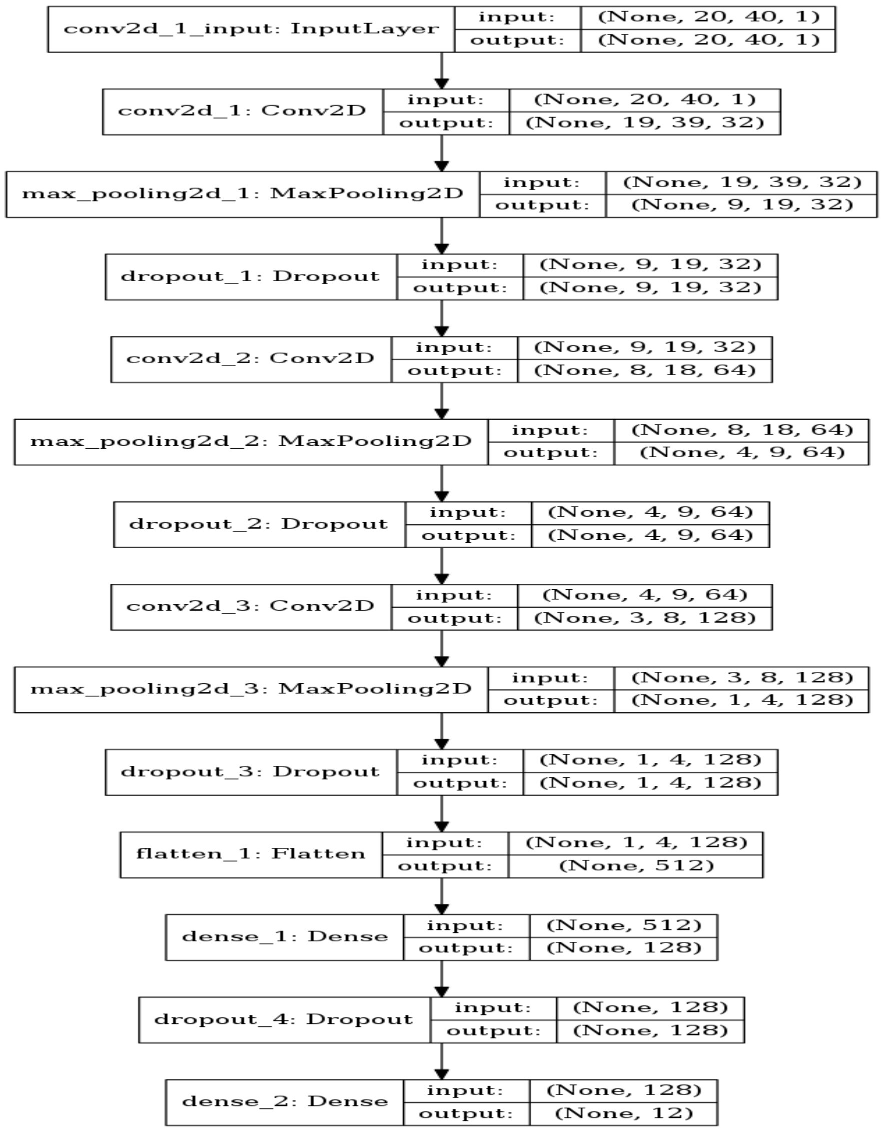

Figure 5: Final model architecture.

To verify robustness I train the model with stratifiedKFold with K=11 and test the model each time and the results

Test accuracy of 1 :	92.3889%
Test accuracy of 2 :	92.4718%
Test accuracy of 3 :	92.3533%
Test accuracy of 4 :	92.7801%
Test accuracy of 5 :	92.7801%
Test accuracy of 6 :	92.7564%
Test accuracy of 7 :	93.1239%
Test accuracy of 8 :	93.0053%
Test accuracy of 9 :	92.7208%

Test accuracy of 10 :	92.8038%
Test accuracy of 11 :	91.8909%
The mean and std 92.64%(+/- .33%)	

Justification

The accuracy of the final model is 91.96% which is better than that of the benchmark.

V. Conclusion

Free-Form Visualization

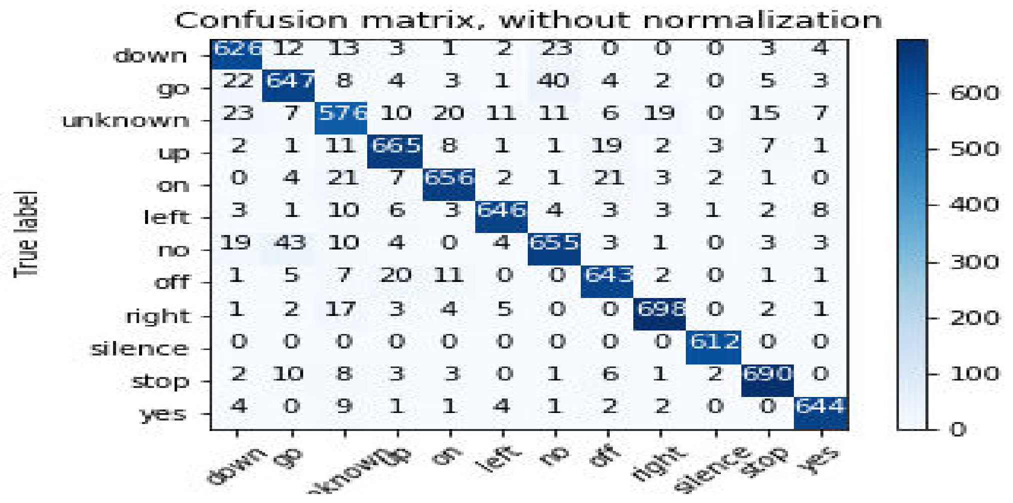

Figure 6: confusion matrix

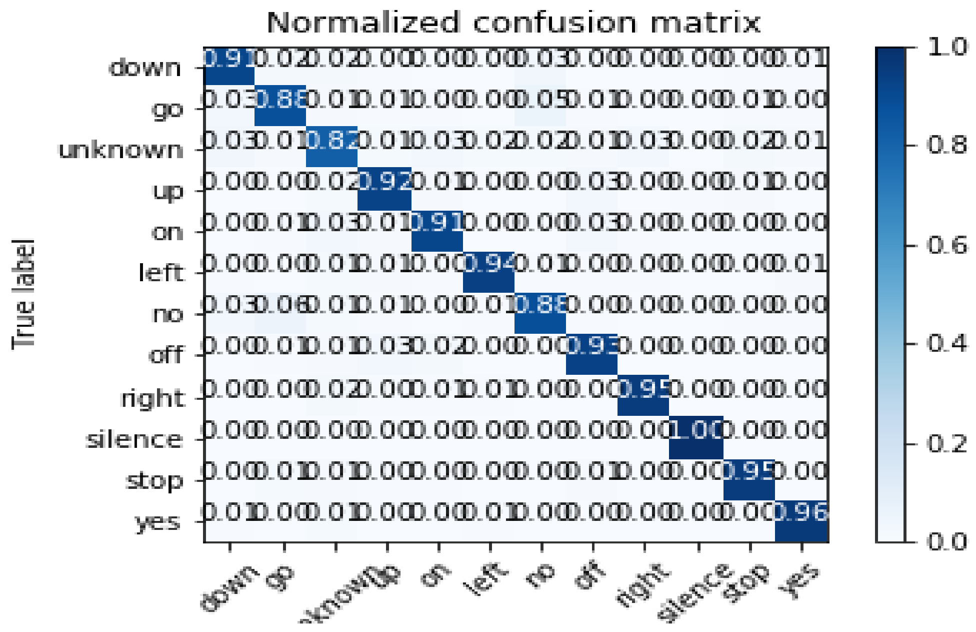

Figure 7: Normalized confusion matrix

The model overall accuracy on the test set was 0.919 (Accurate predictions/All or True Positives/All).

However, the confusion matrix gives additional insight into accuracy by class and intuition for precision and recall efficiency.

An insight we can get from the matrix is that the model was very accurate at classifying silence (True Positive/All = 1.0). However, accuracy for unknown was lower (576/705 = 0.82).

Reflection

The process used for this project can be summarized using the following steps:

- An initial problem and relevant, public datasets were found
- The data was downloaded and preprocessed
- Convert the audio to a numerical representation
- The classifier was built.
- The classifier was trained using the data(multiple times, until a good set of parameters were found)

As for the most interesting aspects of the project, I'm very glad that I found the speech command data set, as this first time I deal with audio files.

Improvement

There are a few ideas which may improve the solution:

- Use networks, or long-short term memory networks, to process the MFCCS. While I used convolutional neural networks to achieve my goal, perhaps other network architectures would be more applicable such as RNNs and LSTMs which are widely used in signal processing and speech recognition.
- Augment and distort signals for testing. While I originally intended to add some random digital distortion to my input data to account for more realistic use cases and to aid against overfitting the models to only the noiseless cases,.
- Use all command from the dataset “30 commands”.

Reference

1- https://www.kaggle.com/c/tensorflow-speech-recognition-challenge

2- https://www.tensorflow.org/tutorials/sequences/audio_recognition

3- http://cs231n.github.io/convolutional-networks/