Machine Learning Engineer Nanodegree
Capstone Proposal
Geoffrey Hung

Domain Background

In this Kaggle competition, I will need to use the Speech Commands Dataset from Google to build an algorithm that understands simple spoken commands such as "yes", "no", "up", "down". By improving the recognition accuracy of open-sourced voice interface tools, we can improve product effectiveness and their accessibility.

This problem becomes more relevant to IoT products such as Google Home and Amazon Echo. Most of the in-room device requires the products to understand the human voice in order for them. As such, this project will try to build the algorithms to predict the words based on audio data.

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

Problem Statement

The problem is to train the model to classify the audio to different short commands, in short, this is multi-class classification problems, the model needs to classify the audio commands to 12 classes and output the accuracy as result.

There are different approachs to this problem, I prefer to try it out with deep learning where I can use what I learned in lessons to do prediction. Also, audio data can be transferred to spectrogram which I can combine CNN as part of the toolkit models.

Datasets and Inputs

Datasets:
The audio files were collected using crowdsourcing by Google, see aiyprojects.withgoogle.com/open_speech_recording for some of the open source audio collection code they used. The goal was to gather examples of people speaking single-word commands, rather than conversational sentences, so they were prompted for individual words over the course of a five minute session.

Twenty core command words were recorded, with most speakers saying each of them five times. The core words are "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", and "Nine". To help distinguish unrecognized words, there are also ten auxiliary words, which most speakers only said once. These include "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow".

There are total 64,727 different audio files, since I don't have the labels in leaderboard set, I will focus on these audio files and split them in ratio of 8-1-1 as training, validation and test set according to the text files provided.

The data distributions are similar except for “noise” and “silence” which they provided longer clips unlike other audio data with exactly one second long. I will sample the clips of “noise” and “silence” to create sets of one second long audio as part of dataset.

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

Solution Statement
The problem will be to train the model to classify the audio to different short commands, there are different approaches to this problem, I prefer to try it out with deep learning where audio data can be transferred to spectrogram and I can use CNN as part of the toolkit models.

Benchmark Model
According to Google tutorial, their models after hours of training on GPU can achieve accuracy between 85% and 90%, the goal for this project will be to achieve the similar accuracy level as Google did with the model I developed myself.

reference: https://www.tensorflow.org/versions/master/tutorials/audio_recognition

With some of vanilla fully connected NN for spectrogram, I can obtain around 30% accuracy after around 50 epochs.

Evaluation Metrics
Metric will be classification accuracy, this is to count percentage of correctness in classifying the short audio. Accuracy = #(classify correctly) / #(test size)

Note that the dataset maybe unbalanced for some set of data and I will also consider F1 score as another evaluation metrics (this part will be discussed when I do my exploration in final projects)

Project Design
My approach to this problem will be divided into following steps,

1. data
In this part, I will refer to different online materials on how to handle audio data. Audio data is different from image or text which I need another representation to convert audio data. I will also explore and visualize the patterns of different words at this stage.

Refer to different kernels, I can convert audio data to another representation
Example kernel:
https://www.kaggle.com/davids1992/speech-representation-and-data-exploration

Audio data will then be splitted to training, validation and test set following ratio of 8-1-1 roughly

2. Model
I will then explore different model architecture with small dataset and train the full set after I picked the final model architecture and evaluate them by different metrics mentioned above (accuracy / F1 if needed)

3. error analysis and conclusion
In this part, I will analyse the error source, are they distributed evenly and is there any correlation b/w them. I will compare my result with Google tutorial benchmark at last as a final comparison