## Capstone Project abdalla Shaaban

## Machine Learning Engineer Nanodegree September 27st, 2018

## 

# Simple Audio Recognition

## I. Definition

### Project Overview

In this section, look to provide a high-level overview of the project in layman’s terms. Questions to ask yourself when writing this section: - *Has an overview of the project been provided, such as the problem domain, project origin, and related datasets or input data?* - *Has enough background information been given so that an uninformed reader would understand the problem domain and following problem statement?*

### 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.

## 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

In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section: - *Have you visualized a relevant characteristic or feature about the dataset or input data?* - *Is the visualization thoroughly analyzed and discussed?* - *If a plot is provided, are the axes, title, and datum clearly defined?*

### 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.
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.
3. Define the loss function, accuracy.
4. Train the network and save best weights based on validation loss.
5. Save and freeze the trained network.

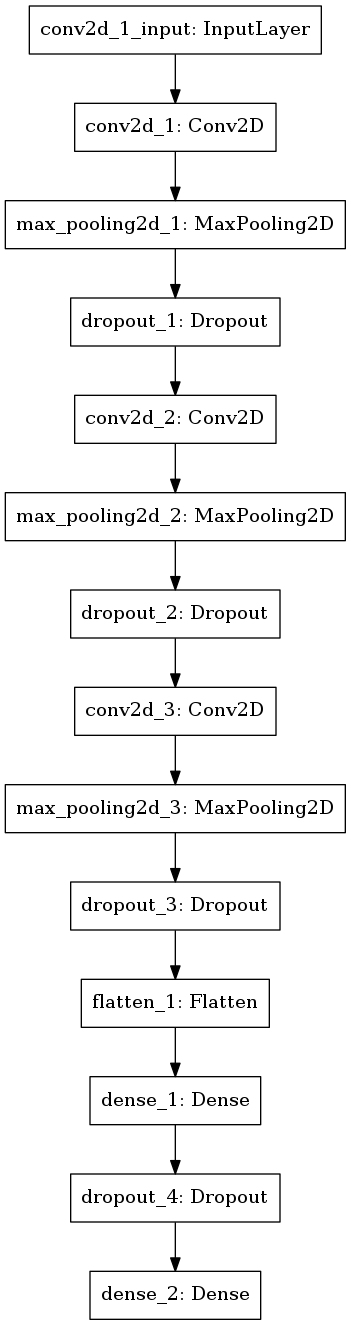


Figure 2: The CNN architecture.

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

### Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section: - *Has an initial solution been found and clearly reported?* - *Is the process of improvement clearly documented, such as what techniques were used?* - *Are intermediate and final solutions clearly reported as the process is improved?*

## 

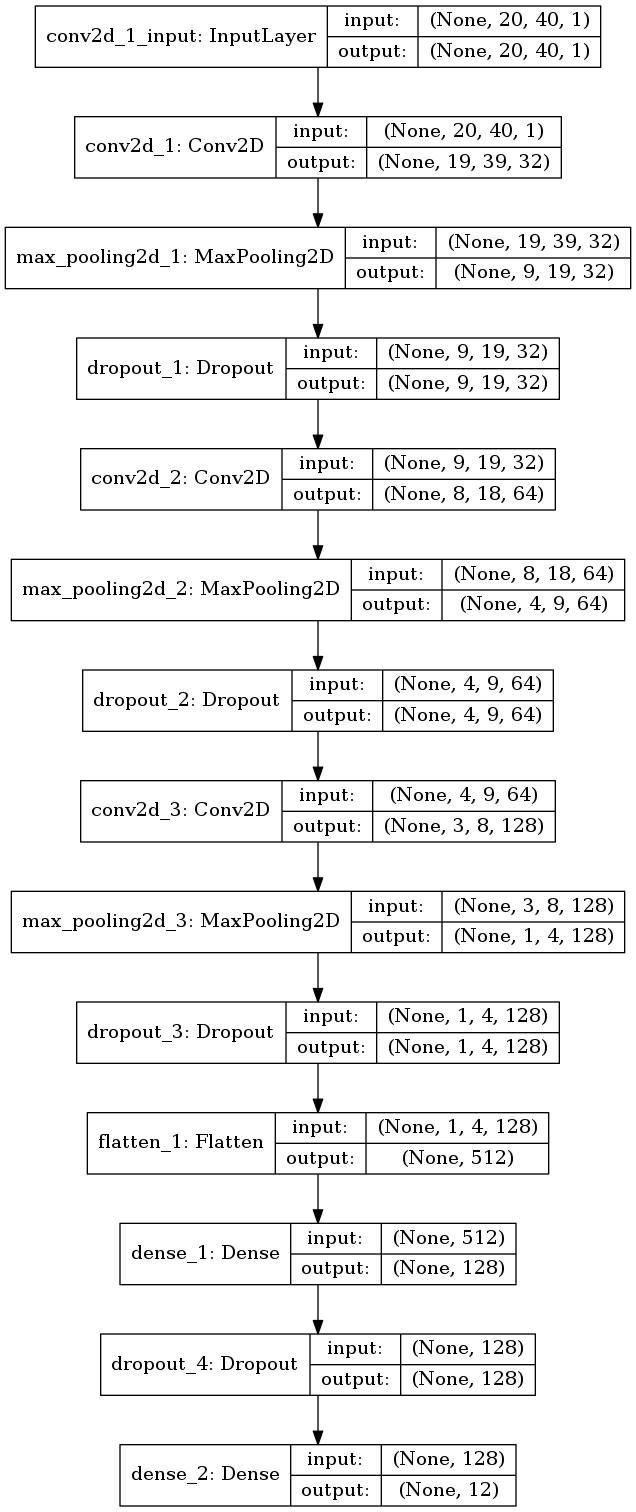
## IV. Results

### 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



### Justification

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

## V. Conclusion

### Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section: - *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?* - *Is the visualization thoroughly analyzed and discussed?* - *If a plot is provided, are the axes, title, and datum clearly defined?*

### 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/