



SAPIENZA
UNIVERSITÀ DI ROMA

NEURAL NETWORKS PROJECT REPORT

1022870

NEURAL NETWORKS

CNN for Keyword Spotting

Author:

Adil Chakkala Paramba

Student Number:

1817501

March 16, 2018

Contents

1	ConvNet for Keyword Spotting	2
1.1	Original Study	2
1.2	Introduction	2
1.3	Used Models and Dataset	3
2	Audio Classification	4
3	Implementation Example	5
3.1	Tensorflow-Simple Audio Recognition	5
3.2	PyTorch Honk	5
3.3	Wake-Up-Word Keyword Spotting implemented in Keras	6
4	Implementation	6
4.1	Tensorflow Transfer learning	6
4.2	Tensorflow Simple CNN	6
4.3	Keras Implementation with Max-Pooling	7
4.3.1	Data Pre-Processing Model	7
4.3.2	Sequential Model	9
4.4	Keras Implementation without Max-Pooling	10
5	Demo	11
5.1	Python	11
5.2	Android	12
6	Results	13
7	How to Run the Code	13
7.1	Required libraries	13
7.2	Getting the dataset	14
7.3	Building it From Scratch	14
7.4	Using the Pre-trained Model	14

1 ConvNet for Keyword Spotting

1.1 Original Study

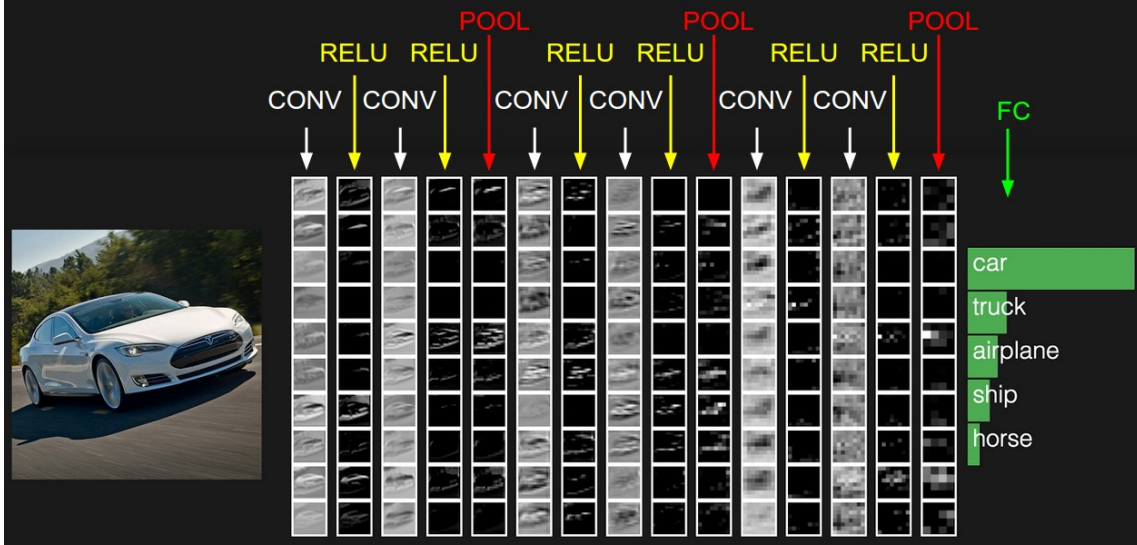
This is a Keras implementation of the Google's research **Convolutional Neural Networks for Small-footprint Keyword Spotting**[1] by Tara N. Sainath and Carolina Parada

In this research paper authors talk about 4 different kinds of networks and I choose one among them (cnn-one-fstride4).

1.2 Introduction

Speech has been a major part of human life from the beginning of time and it is the major means of communication for humans. Speech recognition system is an important technology in the modern world. Nowadays with the state of the art speech recognition systems like OK Google, Amazon Alexa, Apple Siri and Microsoft Cortana, the speech recognition has become a part of our daily life. In the form of personal assistant, all these services help us to do day-to-day activities easily and very effectively. This paper deals with the hot-word detection in a speech recognition system. In simple words, when we say OK Google, the Google assistant will start listening to us, so here Ok Google is the hot-word. There are several methods out there to detect the hot-word and most of them use an RNN or LSTM to do this, but here we are going to deal with Convolutional neural networks.

Convolutional Neural Networks (ConvNets or CNN) are a category of Neural Networks that is very useful and provides good results in the field of image recognition and classification. This is widely used in technologies like self driving cars. A mixture of mathematics and computer science inspired by biological science, since the introduction of CNN, this have revolutionized the field of computer vision. In 2012 Alex Krizhevsky used CNN to win that year's ImageNet competition, drastically reducing the error rate from 26% to 15%.



The image is taken from <http://cs231n.github.io/convolutional-networks/>

CNN is basically used for the classification in the field of images (2-D or 3-D array) but here we are going to deal with the voice (1-D array), so the straightforward question is, can we use a 1-D data for a CNN?. Normally we use an RNN or LSTM for voice data. But using a various mathematical technique we are going to transform the voice/audio data into a 2-D data and then feed it into the convolutional neural network layer and get the prediction.

As building a CNN is an easy task with high-end deep learning libraries like Tensorflow, Keras, PyTorch etc. But the biggest challenge is getting ready with the input data to be fed into the network.

1.3 Used Models and Dataset

In the original paper, the researchers have used the mel frequency model to reconstruct the input data, so we go with this as the technique of converting 1-D audio data to a 2-D data. The Dataset for this is Speech command Dataset from Google Research Blog [6]

2 Audio Classification

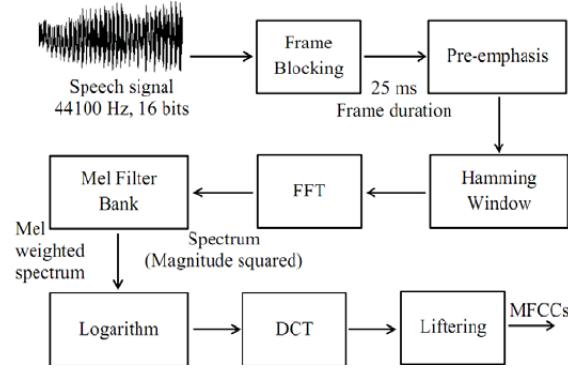
The sound/audio, a single dimensional data, which can be further converted or can be represented in the form of a two-dimensional data using the frequency and related energies[2]. We can do this by using different kinds of representations. We can plot as a Frequency vs Time representation, or Amplitude vs Time etc. The ability of the computer to understand the spoken human language has a great impact on the recent history. Speech analysis and speech detection are one of the major fields of Artificial Intelligence research community.

A computer understanding human speech will be a very close step in solving the intelligence or achieving a conscious Artificial Intelligence. Because once the computer is able to understand and process what we speak, thus it is likely to behave like a human. But unfortunately as the computer only understand binary so we need to represent these data in some other form. There are several methods to do this, which includes, converting these audio files into one of the following form and then feed to the computer:-

- Spectrogram
- MFCC
- Frequency Matching
- Amplitude Matching
- Frequency Power distribution
- Factor analysis

Audio analysis is normally a difficult task compared to Image analysis/Classification, this is because of many parameters including noise, no.of samples(1-second audio normally have 16000 sample points) and many other parameters.

Here we are going to discuss only MFCC, as it had proved very effective over the years.



The above image[7] shows how the MFCC is extracted from a raw voice. Because of the well-structured values of MFCC, these give very good result compared to any other audio processing format. Spectrogram also gives a good quality representation of the audio data but not as detailed as MFCC

3 Implementation Example

3.1 Tensorflow-Simple Audio Recognition

In the tensorflow website [9], under the category tutorials, there is the implementation of this very paper but a different method as well as if we change an argument we can run the very same CNN-one-fstride4, which I have implemented, but I did not use this method and am using keras for building this. So as to get an idea of how does the model work, I actually tried to execute this program and understood the model used, training time, how to build a CNN in tensorflow and a lot more but the training time was nearly 5 hours in my GTX 1070 graphics card, so this is the main reason I wanted to find a different method to implement this so that it takes less time and I can actually tune the hyperparameter by trial and error methods..

3.2 PyTorch Honk

This is another method, which I found on GitHub[4] by Castorini from Deep learning and information retrieval at the University of Waterloo. I tried to implement this, and it was interesting but a bit complicated, so I thought to move ahead and I found another paper.

3.3 Wake-Up-Word Keyword Spotting implemented in Keras

This is a work done by Rajath Kumar at Learning and Extraction of Acoustic Pattern Lab, IISc India, under the guidance of Prof. Sriram Ganapathy. I was impressed by this model, it is a simple model and they used keras which made it even simpler, but still, there was no much information about the preprocessing and also here is the point I decided to write code my own, inspired from Tensorflow blog, keras blog and Wake-Up-Word Keyword Spotting implemented in Keras paper. [10]

4 Implementation

I tried a total of 4 methods and chose the best one. I will explain all the models below, and the best model will be discussed in depth:

4.1 Tensorflow Transfer learning

Here I used my code from previous project which I have optimized and have changed the forked code from tensorflow. So basically what I did here was converted all the audio waves into corresponding spectrogram image as mentioned in the tensorflow tutorial and used inception v3 for re-training.

Accuracy:60%

4.2 Tensorflow Simple CNN

In this method, I converted all the images into 28 x 28 and then made the dataset into a tfrecord format and then applied a CNN with the following layers

- Convolution
- MaxPooling
- Convolution
- MaxPooling
- Droupout = 0.4
- Fully Connected(Softmax)

So I got an output of **Accuracy:76%** and the normal model that is trained with added background noise and audio mixing gives an **Accuracy of 90%**

4.3 Keras Implementation with Max-Pooling

This is the time I Shifted from tensorflow to keras because of the simplicity of implementing the network, the main reason for using keras is because of the tagline of keras **"Being able to go from idea to result with the least possible delay is key to doing good research."** So I wanted to change the hyperparameter and see how the network behaves so I started with keras.

Here keras is used on top of tensorflow, so tensorflow will run in the backend so this makes it easy to convert the keras model to tensorflow model(I converted the keras model to a tensorflow model and have made an Android app of the Same which I will discuss at the end of this).

4.3.1 Data Pre-Processing Model

So as to input the data, first thing I did was Converting the wav file to a Mel-frequency cepstral coefficients, which is one of the best ways to represent voice models, this is the one they used in the main paper of Google, so rather than going with the images I decided to stick to the paper and go with MFCC. MFCC or we can rather call it as Audio embeddings[8]

To obtain MFCC by hard coding, we would need to do five steps [3][5]:

- Take the Fourier transform of (a windowed excerpt of) a signal.
- Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
- Take the logs of the powers at each of the mel frequencies.
- Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
- The MFCCs are the amplitudes of the resulting spectrum.

But thanks to the librosa library, we can use this library and generate the MFCC and I have taken the wav2mfcc function snippet from the example library program. So the main parameters used in these are :

- MaxLength = 11, this was chosen by experimentation basics from one of the reference paper.
- Audio Channel = 1(mono)
- sampling rate is 16000

Now, If maximum the length exceeds MFCC lengths, then pad the remaining ones with zero.

Once we have this function we will run it for all files and save all the MFCC into an array, so each label will have one array which contains all its wav file's MFCC so using the `save_to_array()`, which I wrote, will do this and save all of them as (.npy) files.

Now we have each voice as a 2-D array and as the keras will return the label as numbers or one hot encoded vector, we use the inbuilt function of keras `to_categorical` to convert it into label we want. Now we are ready to train the model, so as we have already converted the model into (.npy) files, we don't have to calculate MFCC every time we run the program, instead we will just load them from the disk.

Now using the train test split function of sklearn library I split the data into 80% training and 20% testing.

According to 3.3.2, now I built a keras sequential model with the following attributes:

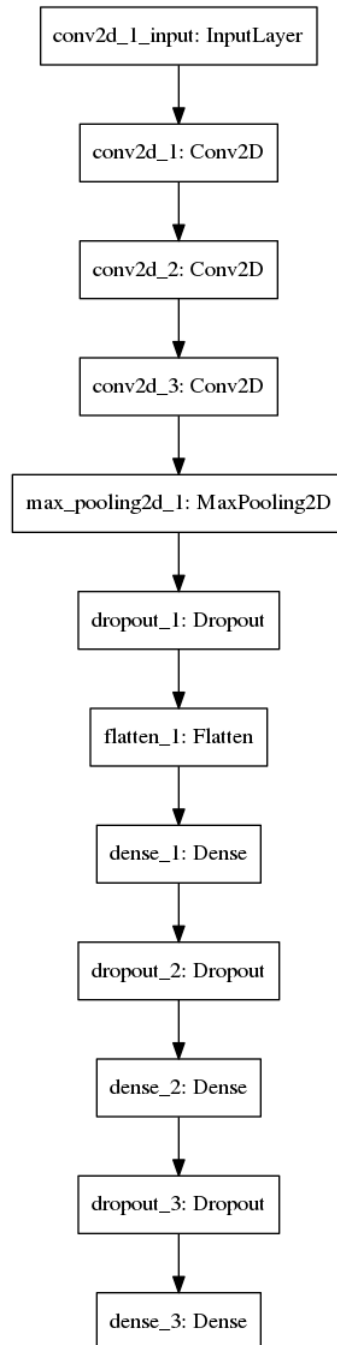
- epoch = 100 (the entire dataset is passed 100 times back and forth the network)
- batch_size = 100 (as we can't pass all the data points at one time, we divide it into smaller portions and then pass)

Accuracy : 89.02, that is a pretty good accuracy.

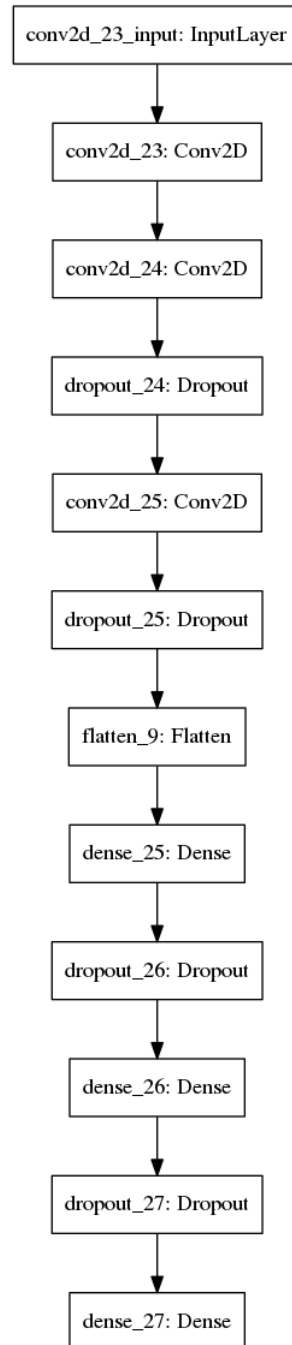
Now we are ready with the trained model, using `save.model` in keras I saved this, so further it can be used easily without any training. To predict a new voice, first we convert the voice to MFCC then pass it to the model using `predict` function, so I run this in a loop, record, convert to MFCC, predict. I will attach the video of the prediction according to voice input as well as I have everything uploaded to GitHub.

4.3.2 Sequential Model

The sequential model used in this model generated through `model.save()` of keras



4.4 Keras Implementation without Max-Pooling



In the above model, I removed the max pooling layer, this was just a try to see how will the network behave if I remove the max-pooling layer, and yes! it worked, the new **accuracy: 92.30** there was a slight increase in the accuracy.

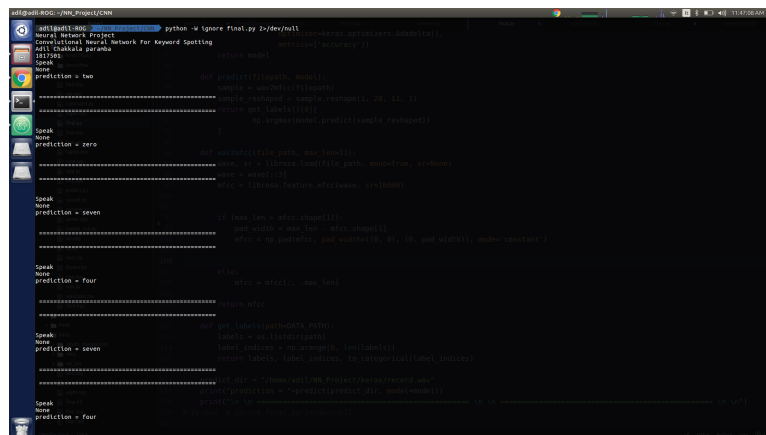
5 Demo

I have made two methods to check the model

- Python program Running in a loop and Displaying the prediction according to the voice input
- Android app Forked from Tensorflow Repository and modified code according to my model(this is just demoed with existing code I just changed the input parameters and the MFCC function and added a home page and a details page).

5.1 Python

So I have this final program which uses the alsaaudio library to record voice and then I feed this recorded voice to the model and get the prediction and in the attached recordings you can find the demo of this.



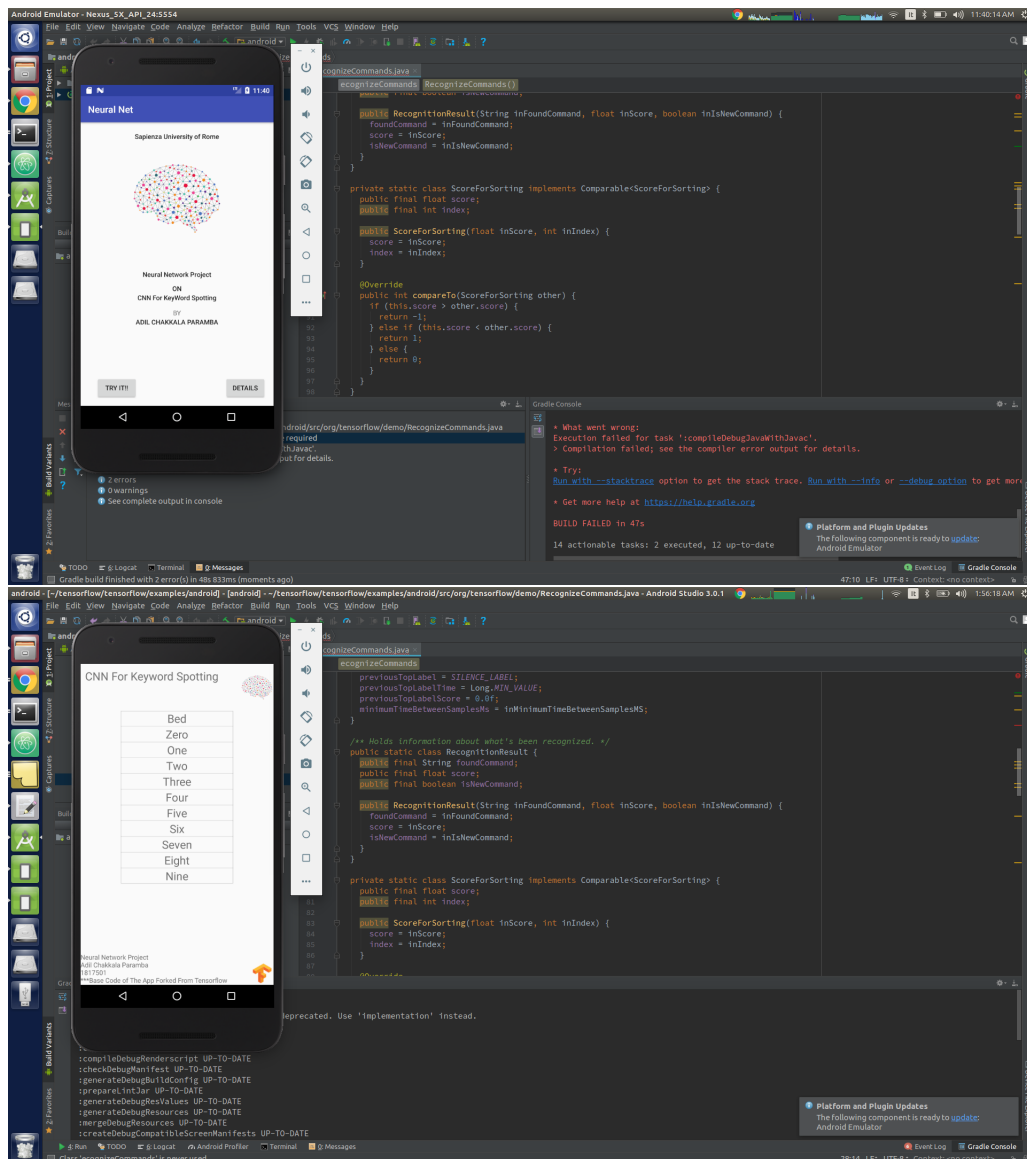
```

$ cd /home/paramba/Projects/CNN
$ python -W ignore final.py >/dev/null
=====
Speak
None
prediction = two
=====
Speak
None
prediction = zero
=====
Speak
None
prediction = seven
=====
Speak
None
prediction = four
=====
Speak
None
prediction = seven
=====
Speak
None
prediction = four
=====
Speak
None
prediction = seven
=====
Speak
None
prediction = four
=====

```

5.2 Android

Base app code is taken from the tensorflow repository and had few edits and tweaks to make it work with my model so basically the saved model I converted using a program from StackOverflow so I got the .pb file which is recommended tensorflow model format and then I used it in the app. I added few more activities to the App, a welcome page, details of the project page and then the demo app.



6 Results

With the final model used without a max-pooling layer, I have achieved over 92.30%. So without just checking with the dataset from google I tried working with my own voice as well as I collected the 11 words from few of my friends and I tested those too. It produced a very good output for all the samples and hence I understood:-

- How to build a CNN from scratch
- How to prepare the dataset for Neural Networks
- How to transport the trained model to an android platform
- Keras
- Tensorflow
- Tensorboard
- Tensorflow Projector
- Converting keras model to tensorflow model
- Preparing dataset in tfrecord format
- Embeddings

7 How to Run the Code

7.1 Required libraries

- Tensorflow
- Keras
- librosa
- sklearn
- numpy
- tqdm
- h5py
- pyaudio

- time
- wave

7.2 Getting the dataset

Download the dataset from Google research blog or use your own dataset. Once you have downloaded the dataset copy it into the data folder in the repository's root directory or you can keep it anywhere, but remember to change the data directory in the program.

7.3 Building it From Scratch

So as to build the model from scratch, change the `DATA_PATH` variable in the `nn_final_full.py` to the main folder of your dataset and on the last line in the `predict` function, change it to any of the audio files in the `validation_data` folder or you can use one of your recordings.

7.4 Using the Pre-trained Model

Note in the pre-trained model I have only trained 11 speeches, they are :

- bed
- zero
- one
- two
- three
- four
- five
- six
- seven
- eight
- nine

So to use this, you can Run the program `final.py` and change model to `my_model.h5` in the model directory of the repository and also `WAVE_OUTPUT_FILENAME` to `<filename.wav>` and also copy paste the same directory path to the third last line `predict_dir`

References

- [1] Convolutional neural networks for small-footprint keyword spotting. <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43969.pdf>.
- [2] Machine learning for audio,image and video analysis. <http://www.springer.com/it/book/9781447167341>.
- [3] Min xu; et al. (2004). "hmm-based audio keyword generation". in kiyoharu aizawa; yuichi nakamura; shin'ichi satoh. <https://web.archive.org/web/20070510193153/http://cemnet.ntu.edu.sg/home/asltchia/publication/AudioAnalysisUnderstanding/Conference/HMM-Based%20Audio%20Keyword%20Generation.pdf>.
- [4] Pytorch honk. <https://github.com/castorini/honk>.
- [5] Sahidullah, md.; saha, goutam (may 2012). <http://www.sciencedirect.com/science/article/pii/S0167639311001622>.
- [6] Speech command dataset from google research blog. https://storage.cloud.google.com/download.tensorflow.org/data/speech_commands_v0.01.tar.gz.
- [7] Speech recognition(feature extraction). <http://recognize-speech.com/feature-extraction/mfcc>.
- [8] Tensorflow embeddings. https://www.tensorflow.org/programmers_guide/embedding.
- [9] Tensorflow simple audio recognition. https://www.tensorflow.org/versions/master/tutorials/audio_recognition.
- [10] Wake-up-word keyword spotting implemented in keras. <https://github.com/rajathkmp/keyword-spotting>.