TensorFlow Speech Recognition Challenge

Petra Brčić
Applied Mathematics
petrabrcic94@gmail.com

Ivan Čeh
Computer Science
ivan.ceh1234@gmail.com

Sandro Lovnički
Computer Science
lovnicki.sandro@gmail.com

June 26, 2018

Abstract: We are witnessing the significant development of the blooming relationship between science and technology. Humanity is striving on making everyday life easier and more controllable by the integration of the intelligent machines. We want to control those machine easier, such as by spoken commands. This is where this project comes in and answers that question.

Keywords: tensorflow, speech recognition, audio recognition, feature extraction, MFCC.

Contents

I. Introduction

. . . for which our code can be found at our GitHub repository[1].

II. Data

This project was given by Kaggle as Tensorflow challenge, where the task is to correctly classify the audio recording. Dataset was given by Tensorflow as one second audio clips of different words spoken by different people. Dataset can automaticly be downloaded when starting the train part of machine learning solution to the problem. The goal is to have a model that tries to classify a one second audio clip as either silence, an unknown word, "yes", "no", "up", "down", "left", "right", "on", "off", "stop" and "go", even though more words can be found in the dataset and included in the training process. Each of the words is separated into files named as the recordings of the word they present. In order to make these clips as realistic as they would be in real life, the background noise is added in preprocess part of the code. There is a file of few background noises within the dataset that is downloaded. Each audio file is encoded with the id of the person who recorded the word, followed by nohash and a number from 0, specifing the time the same person recorded the word. Apart from the audio files that take up the majority of the data file, testing and validation lists can be found too. Each of those text file contains a list of the audio clips so that dataset can be patitioned in training, validation and test set.

i. Preprocessing

In order to feed our neural network with the data, we first need to process it. Neural network is fed by the image, so we need to tranform the audio file to a spectrogram. Spectrogram is a visual representation of the spectrum of frequencies of sound or other signal as they vary in time. Spectrograms can be generated by an optical spectrometer, a bank of band-pass filters or by Fourier transorm. First, let us observe what the data looks like after reading the audio file.

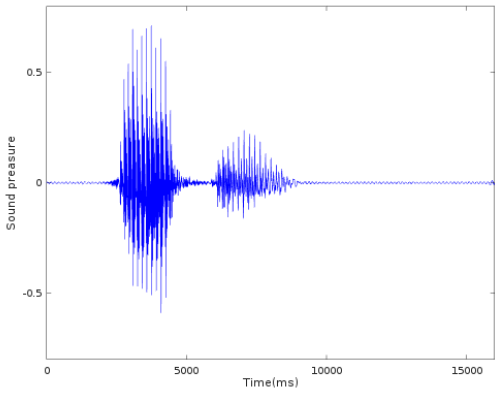

Figure 1: *Waveform for audio 'happy'*

Creating a spectrogram using FFT is a digital process. Digitally sampled data (which ours is), in time domain, is broken up into chunks, which usually overlap, and Fourier transformed to calculate the magnitude of the frequency spectrum for each chunk. Each chunk then corresponds to a vertical line in the image; a measurement of magnitude versus frequency for a specific moment in time (the midpoint of the chunk). These spectrums or time plots are then laid side by side to form the image or a three-dimensional surface, or slightly overlapped in various ways, i.e. windowing. This process essentially corresponds to computing the squared magnitude of the short-time Fourier transform (STFT) of the signal x(t) - that is, for a window ω,

$$spectrogram(t,\omega) = |STFT(t,\omega)|^2,$$

where

$$STFT\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]\omega[n-m]e^{-j\omega n}.$$

The STFT is performed on a computer so it uses the FFT (Fast Fourier Transform), so all the variables are discrete and quantized.
After running our code in Octave to generate

[1] https://github.com/Qkvad/SpeechRecognition

spectrogram yield the figure 2

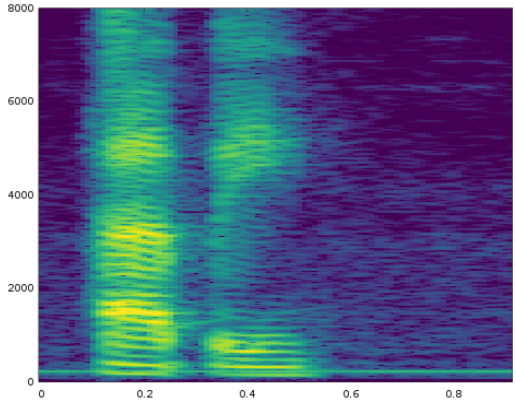

Figure 2: *Spectogram for audio 'happy', 80ms window with 10ms step*

Our code works slightly different, first we scale the volume, then we shift it in time, add background noise and then calculate spectrogram. But for all the features of spectrogram to be easily visible on the paper form, we added the octave function to obtain the standard version of the spectrogram.
There is one more thing we have to dig into, that is getting from spectrogram to the image that is going to feed the machine learning algorithm. Final output of the preprocess is mel-frequency cepstrum [2] coefficients (MFCCs), the coefficients that make up an MFC. MFC is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. This frequency wraping can allow for better representation of sound in audio compression.
MFCCs are derived as follows:

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

III. Machine Learning

i. Convolutional Neural Network

ii. Results

IV. Different Approaches

References

[1] Misha E. Kilmer, Karen Braman, Ning Hao. Third Order Tensors as Operators on Matrices: A Theoretical and Computational Framework with Applications in Imaging

[2] Misha E. Kilmer, Carla D. Martin. Factorization Strategies for Third-order Tensors

2

[2] Cepstrum (the name derived by reversing the first four letters of "spectrum", whereas operations on cepstra are labeled quefrency analysis, liftering or cepstral analysis) is the result of IFT of the logarithm of the estimated spectrum of a signal