Austin Dunn
Oct 21, 2016
MUMT 502
Prof. Ichiro Fujinaga

Midterm Report

My senior project in music technology involves a machine learning voice recognition problem, in which I am designing a neural network to recognize the difference in the voices of Hillary Clinton and Donald Trump. Though for now the two current presidential candidates are my focus, my solution more generally could be used to collect data on conversations between a pre-defined set of people of any size, including how long each person spoke and how many times each person interrupted another.

As previously mentioned, I am currently focused on a two-class problem between Hillary Clinton and Donald Trump. From preliminary research, particularly details provided in [1] [2] and [3], I learned that the standard way of feeding audio data into neural nets is by spectrogram image of the audio. From here, many multi-layered architectures use Convolution and Pooling layers (such architectures are called Convolutional Neural Networks, or ConvNets), a technique inspired by the animal visual cortex, to analyze the image for important features.

To get myself started, I wrote a script to create spectrograms on *n*-sample frames of audio (hop size of 256) taken from the first presidential debate. The debate provided good data because of a 'hold-your-applause' rule, resulting in unadulterated vocal data from each candidate. Each spectrogram was then split into a directory using the following convention:

```
[directory]/[training|testing]/[hillary | trump | class3 |
class4 | ...]/s.png
```

where *s* is just an index number to identify each sample. This structure puts all the meta-data in the file path, and makes it easy to programmatically find the number of classes, the names of those classes, and each sample to be used for testing and training. I put one of every seven samples in the testing directory, to ensure a larger amount of training data.

I installed TensorFlow, a library developed by Google's Brain Team and released under open source Apache 2.0 license in November 2015. I then replicated TensorFlow's beginner tutorial, which sets up a single-layer neural net to recognize to a low degree of accuracy the digits printed in the MNIST data set, using my own spectrogram data. Initial tests have yielded results ranging from 50-70% accuracy, for longer training periods (i.e. more data), the results tend to settle right around 65%. Increasing the amount of data fed to the network during training doesn't ever seem to improve things beyond this limit.

Looking forward, I have plans to 'deepen' my model with convolution and pooling layers. To do this, I'll be replicating the code given in TensorFlow's Deep MNIST tutorial, which also makes it easy to modify the number of convolution/pooling layers and the operations performed by each of these layers. This will require a good deal of data modification, as the images I'm using now are quite large, which will create serious problems to do with memory and time when fed with a multi-layered network. I'll also be refining my data to only include frames that have a amplitude of a certain threshold - in other words, frames that actually contain vocal audio data. Beyond these modifications, there's always more - from modifications to the number of layers and operations performed in each layer, to data augmentation as seen in [2], I have many things I'm anticipating trying in order to improve-and perfect-my results.

Bibliography

1. Keunwoo Choi, Gyorgy Fazekas, Mark Sandler. Automatic Tagging Using Deep Convolutional
Neural Networks. Queen Mary University of London, 2016.

2. Jan Schlüter, Thomas Grill. Exploring Data Augmentation For Improved Singing Voice Detection With Neural Networks. Austrian Research Institute for Artificial Intelligence, Vienna, 2015.

3. Philippe Hamel, Simon Lemieux, Yoshua Bengio, Douglas Eck. Temporal Pooling And Multiscale Learning For Automatic Annotation And Ranking Of Music Audio. DIRO, Université de Montréal. CIRMMT.