Smithers

Project Update

Jacob Jones, ECE 551

Outline

- Introduction – *What's this all about?*
- Motivation – *Why am I doing this?*
- Background – *Contemporary methods*
- Proposed Method – *What I will be doing*
- Preliminary Results – *What I have done so far*
- Proposed Research – *What still needs to be done*
- Conclusions – *Wrap up this presentation*

Introduction

- *Smithers* is a machine learning model used to detect a wake word
- This model uses mel-frequency cepstral coefficients and a convolutional neural net to make predictions on real-time audio
- This presentation reports on the progress made towards this effort

Motivation

- Smart devices are everywhere
- Smart devices respond to voice commands
- Three big ones:
  - Google Assistant, Alexa, Siri
- I am interested in embedded devices
  - How does voice recognition work on resource-constrained hardware?
- Are these devices always listening?
  - Short answer: No
  - Long answer: Sort of…

Background

- Are these devices always listening?
- How do you respond to commands without always listening?
  - Voice command recognition too computationally intensive
- Amazon, Apple, and Google have researched this topic extensively
- Common solution: “wake words”
- Idea: train model to recognize only one command
  - “Hey Alexa”, “Hey Siri”, “Hey Google”
  - Once command is recognized, activate larger model
- Some devices place model on separate dedicated processor

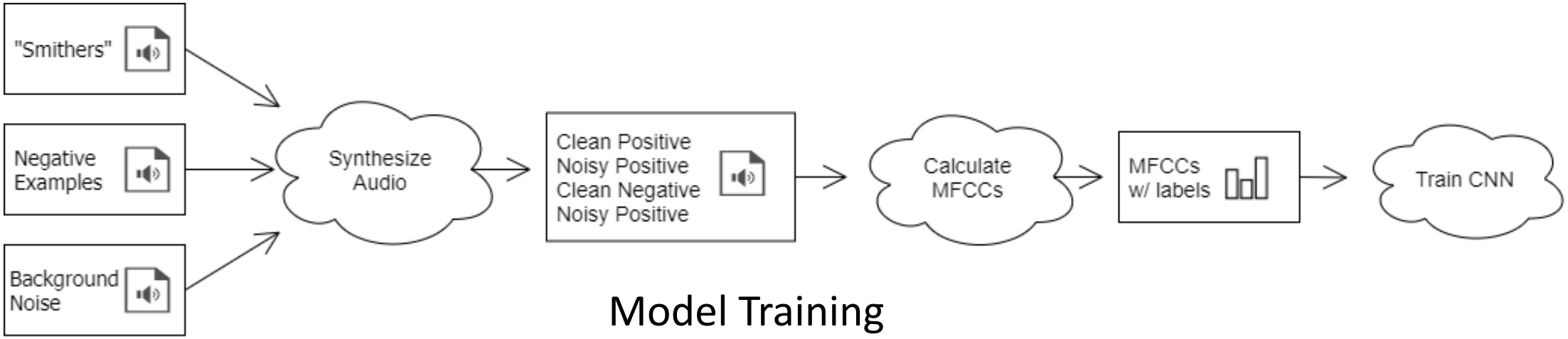
Background (cont.)

- Common approaches:
  - Convolutional Neural Nets (CNNs)
  - Mel-frequency cepstral coefficients (MFCC) – more on these later
- Papers referenced:
  - C. Jose, Y. Mishchenko, T. Senechal, A. Shah, A. Escott, and S. Vitaladevuni, “Accurate detection of wake word start and end using a cnn,” 2020 (Amazon)
  - G. Chen, C. Parada, and G. Heigold, “Small-footprint keyword spotting using deep neural networks,” in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 4087–4091 (Google)
  - S. Sigtia, R. Haynes, H. Richards, E. Marchi, and J. Bridle, “Efficient voice trigger detection for low resource hardware,” in INTERSPEECH, 2018 (Apple)

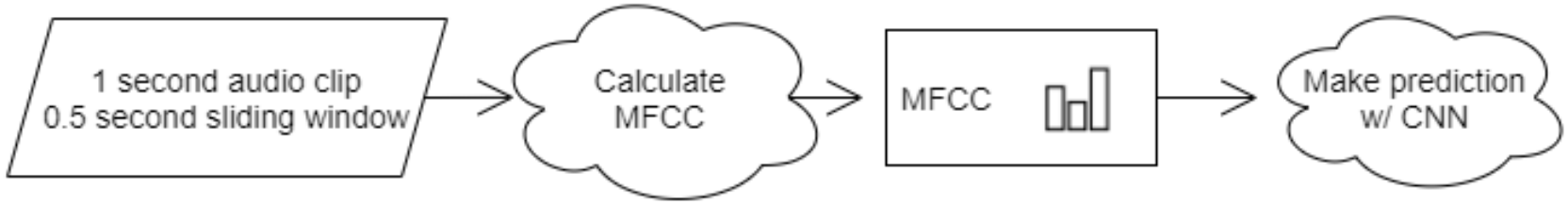
Proposed Method – "Smithers"

- Develop my own CNN capable of recognizing the wake word "Smithers"
- Why "Smithers"? Smithers is the faithful assistant to the crotchety Mr. Burns in *The Simpsons*
- Use methods proposed by the referenced research paper

Proposed Method (cont.)



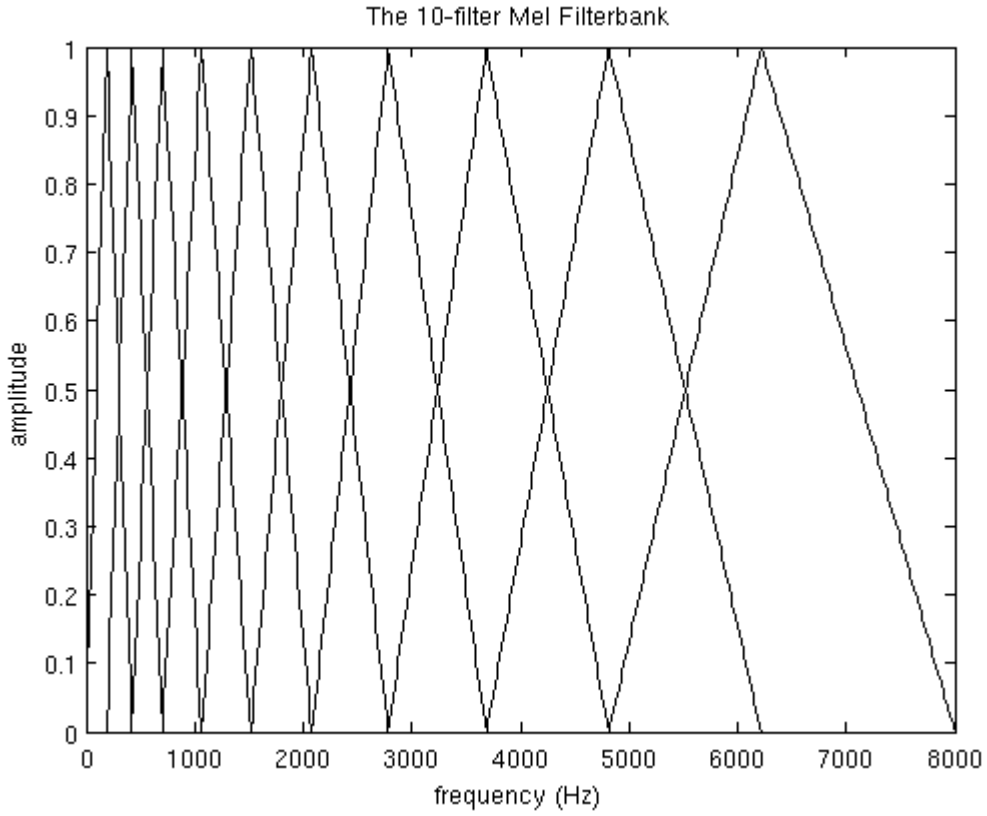
Model Training



Implementation

Proposed Method (cont.) - MFCCs

- Human ear frequency response is not linear
- Mel scale better mimics human hearing
- MFCCs – take FFT, apply mel filter bank, take logs of powers at each mel frequency, take discrete cosine transform
- MFCCs are amplitudes of spectrum
- Idea: distil spectrum into only relevant information
- Train CNN with mel-frequency cepstral coefficients (MFCCs)



Source: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

Proposed Method (cont.) - CNN

Training:

- Training audio will be trimmed into 1 second clips – important!
- MFCC coefficients will be calculated for each clip
- Convolutional neural net will train on the MFCC coefficients

Implementation:

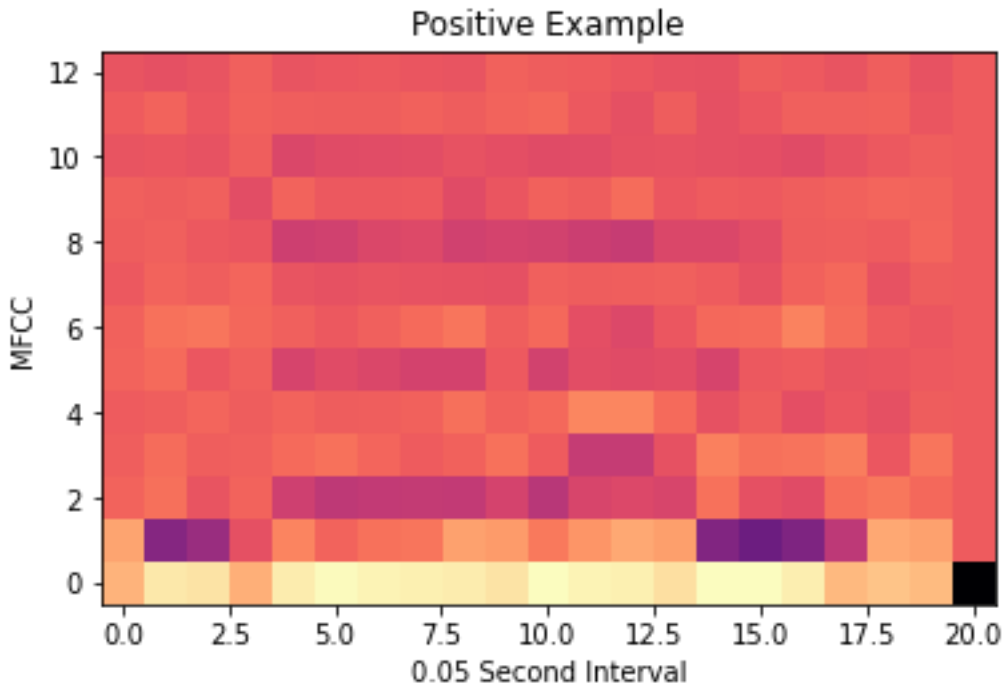
- Two 1-second input audio buffers will be used, spaced 0.5 seconds apart
- This sliding window will help when trigger word is split between two 1-second intervals

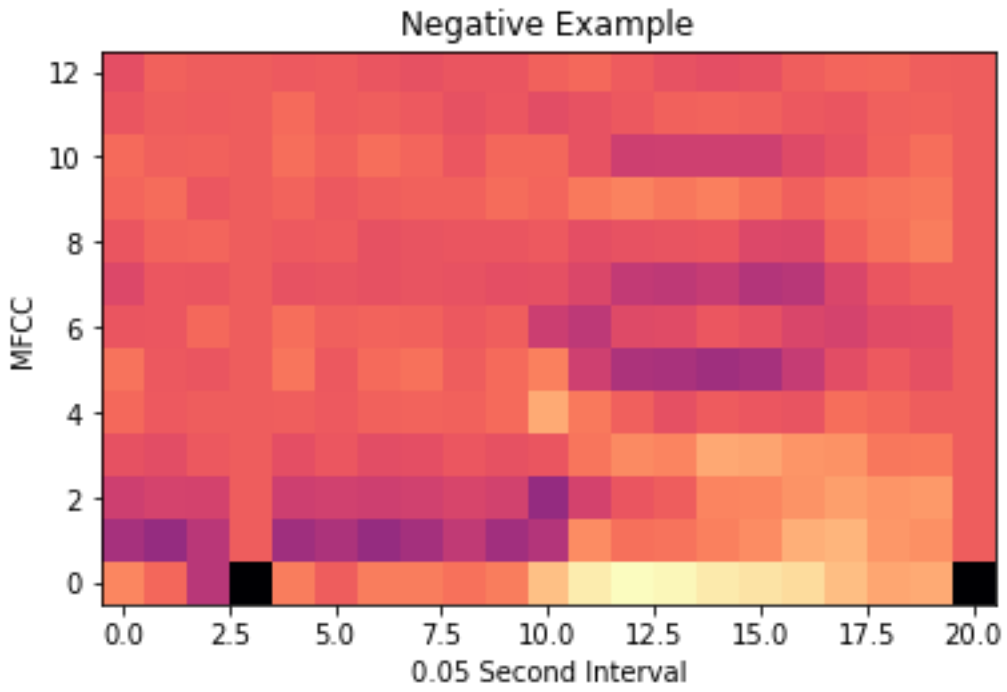
Preliminary Results – Data Synthesis

- Recorded 30 examples of the wake word “Smithers”
- Used 130 negative examples from Google’s Speech Commands Dataset
- Used 200 noise clips from Harvard’s ESC-50 dataset
- Synthesized 400 examples each of positive and negative examples
  - For each loop, choose one positive example, one negative example, and one background noise
  - Randomly select 1 second interval of noise
  - Insert each positive example, negative example into random start point of background noise
  - Save noisy positive, clean positive, noisy negative, and clean negative
  - Half of dataset is clean, half is noisy

Preliminary Results (cont.) - MFCCs

- Audio was downsampled to 10khz
- MFCCs were calculated at 0.05 second intervals – 20 total for each 1 second clip





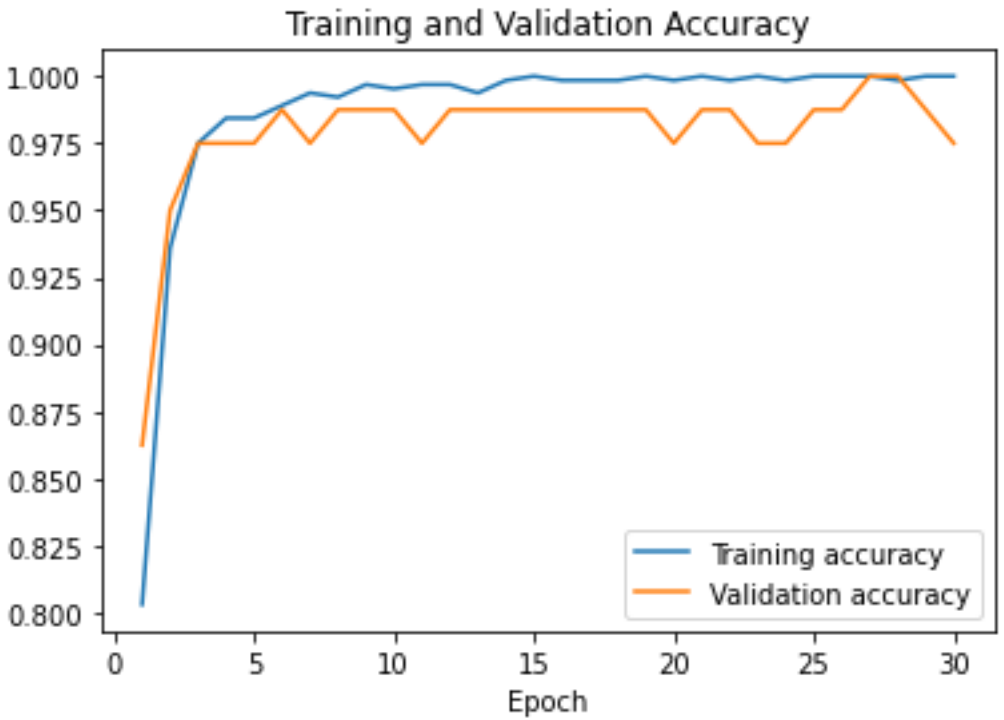
Preliminary Results (cont.) - CNN

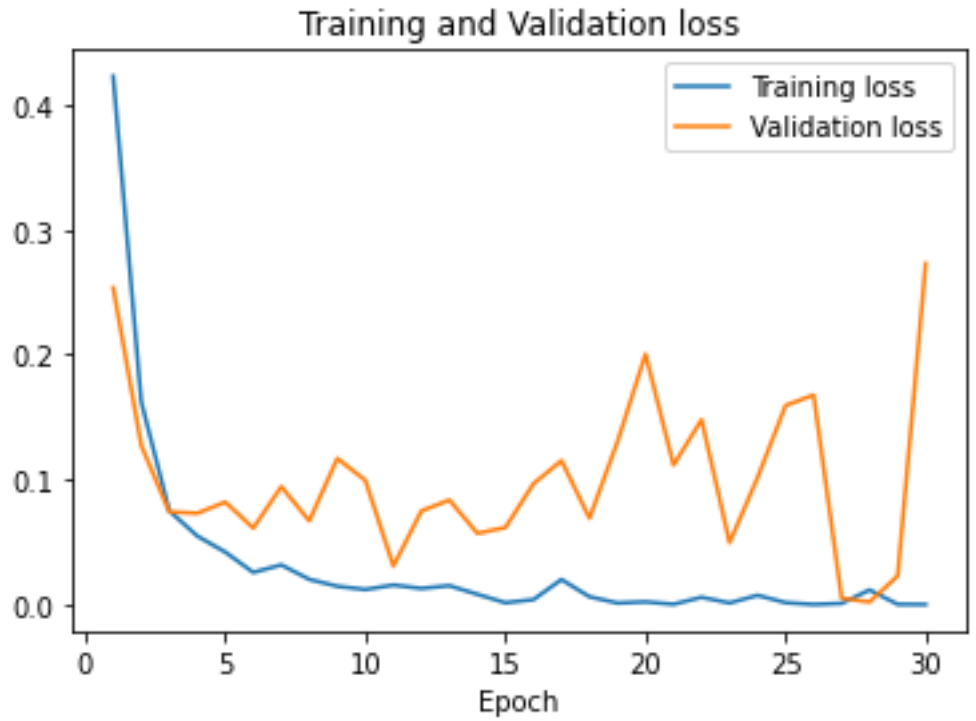
- CNN architecture ripped from the internet
- Need to play around with layers
- Used RelU activation as suggested by Apple

```
Model: "Smithers"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 13, 21, 1)]       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 20, 32)        160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 10, 32)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 9, 32)          4128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 5, 32)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 4, 64)          8256
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 2, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 20,865
Trainable params: 20,865
Non-trainable params: 0
_________________________________________________________________
```

Preliminary Results (cont.) - Validation

- Model works!
- Works well?





Preliminary Results (cont.) - Discussion

- Tested the model on new audio recordings
  - Model recognized when I said “Smithers”
  - Key words there: “I said”, not “Smithers”
  - Model also false-positived on “banana”, “Alexa”, and “dinosaur”
- Does model recognize “Smithers” or does model recognize my voice?
  - Only positive examples came from my voice
  - Negative examples all came from external datasets
  - How do you tell model what to look for?

Proposed Research

- Record negative example using my voice
- Improve example synthesis to better match real audio
  - Synthesized clips need to be similar to real-time audio
- Play around with CNN architecture
- Implement real-time predictions
- Explore effects of reducing sample rate, quantizing MFCCs
  - Interested in applying to embedded hardware
  - How do I quantify the tradeoff on these sacrifices?

Conclusions

- *Smithers* is a machine learning model used to detect a wake word
- MFCCs and CNNs are the key components to this model
- Current model works, has issues