FourthBrain

Spotify Wake Words Project

Monu Singh, Nischal, Nkem Michael Onuorah

Outline

- Problem
- Solution
- Data + Model
- MLE Stack

Problem

- Context - Voice interaction or commands “Hey Google”, or “Hey Siri” rely on keyword spotting to start interaction on local device. It helps people experience “Hands-free” searching and task completion

  Keyword Spotted → Acknowledge → Post “request” → Web service → Get “response”

- What & Why - Opportunities to improve the above technology -
  - Triggers on negative wake words, unrelated speech, background noise, or silence
  - High no. of instance, when device does not trigger on positive wake words
  - Need for quick response & acknowledgment
  - Ability to customise wake word
  - Wake model to be lightweight & energy efficient

Solution

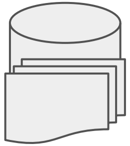
- To design, build and deploy a lightweight Keyword spotting ML model (CNN, SVM) and exposed as a mobile-application that can process a *"custom wake word"*
- Voice response with results by respecting local device resource constraints (low compute) and adhering to ethical challenges (Privacy respecting and non-eavesdropping)
- Model will measure following metrics which tie backs to existing challenges :-

1. Accuracy of the custom wake word detection (Primary Metric)
2. Minimize False Reject Rate per hour of speech
3. Minimize False Alarm Rate per hour of speech
4. Low latency → Measure & Reduce time to acknowledge

Data

<u>Google Speech Command Dataset V2 35 *</u> *(For Model Training)*

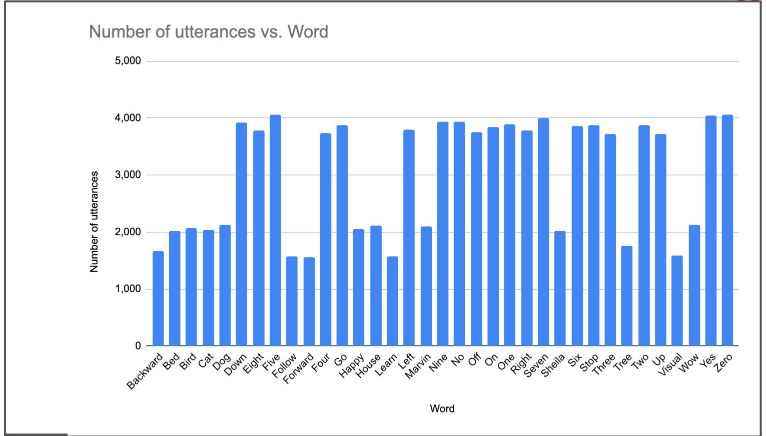
- The dataset consisted of 105,829 utterances of 35 words
- Stored as a one-second (or less) WAVE format file, with the sample data encoded as linear 16-bit single-channel PCM values, at a 16 KHz rate.
- There are 2,618 speakers recorded, assigned unique hex code to each. (All American accents, Language - English)

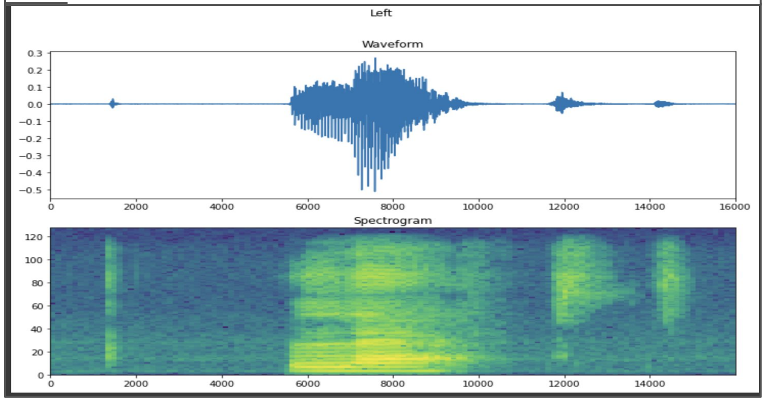


Folder structure in input data with word as parent directory

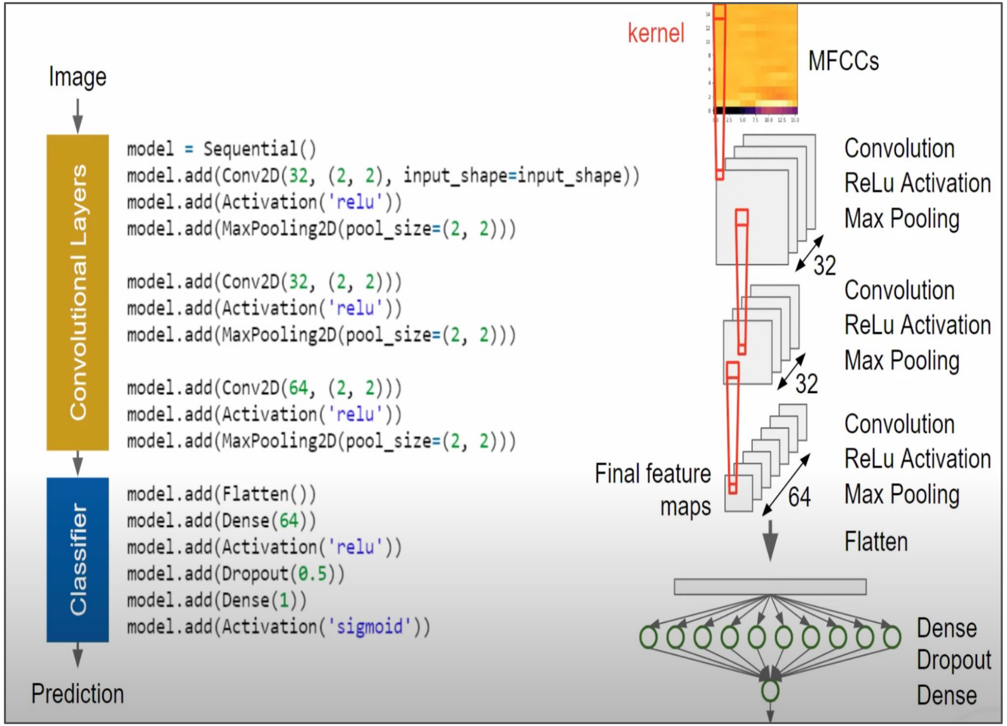
TensorFlow

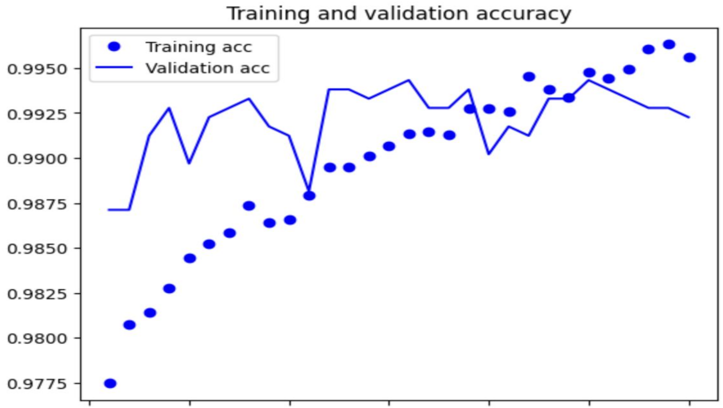
EDA Using Google Research Colab
and TF tutorial on audio word detection. [EDA-Notebook]

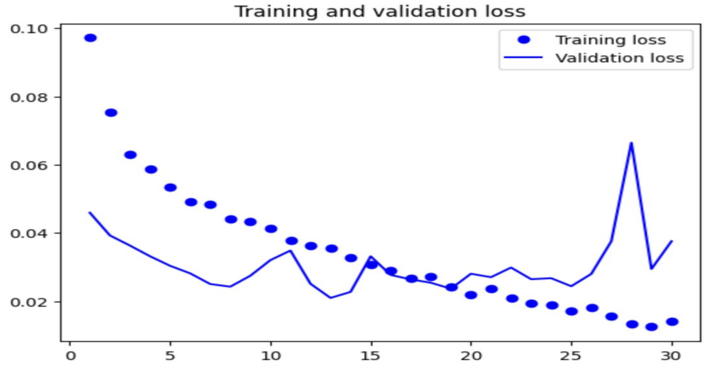




CNN Model



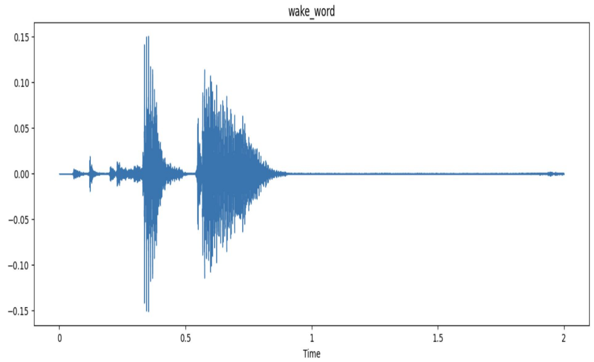




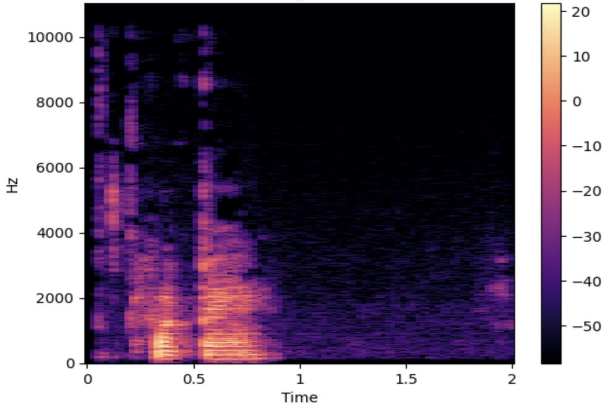
source - https://www.geeksforgeeks.org/python-image-classification-using-keras/

Dataset for DNN and MLP Classifier

- Created our own audio dataset
  - Built the library of background noise and separate library of wake word
  - Preprocess each file to MFCC format before using it to train/test model



- Wake word sample

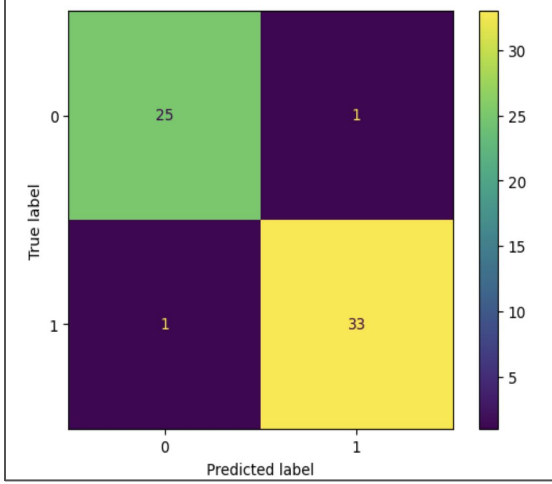


- Background noise sample

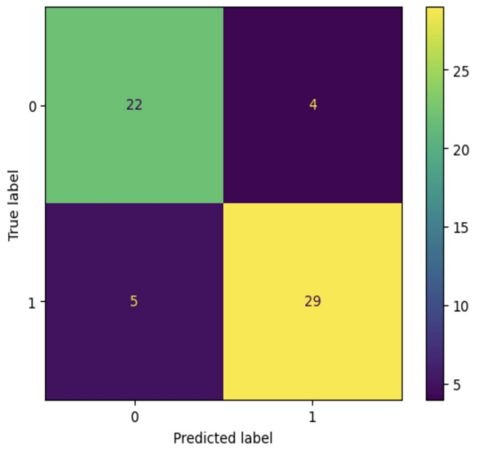
DNN and MLP Classifier - Custom Dataset

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense (Dense)               (None, 256)               10496

activation (Activation)     (None, 256)               0

dropout (Dropout)           (None, 256)               0

dense_1 (Dense)             (None, 256)               65792

activation_1 (Activation)   (None, 256)               0

dropout_1 (Dropout)         (None, 256)               0

dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 76,802
Trainable params: 76,802
Non-trainable params: 0
_________________________________________________________________
```

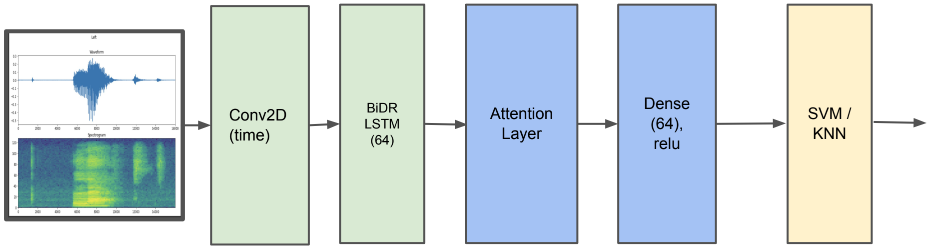
```
Model Classification Report:

2/2 [==============================] - 0s 1ms/step
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        26
           1       0.97      0.97      0.97        34

    accuracy                           0.97        60
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29b61d970>
```



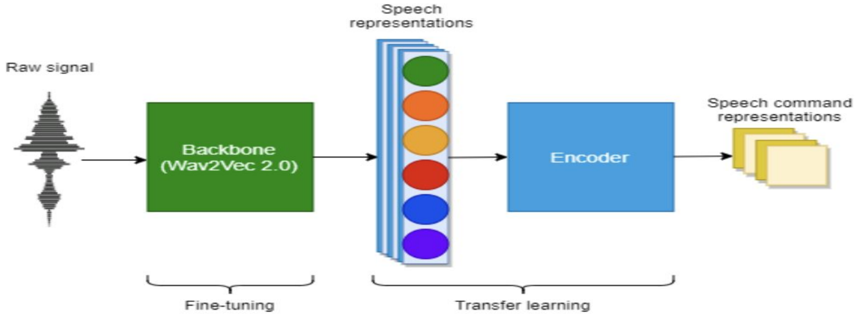

```
MLPClassifier
MLPClassifier(max_iter=500, solver='lbfgs')

The prediction accuracy is:  85.0
              precision    recall  f1-score   support

          No       0.81      0.85      0.83        26
         Yes       0.88      0.85      0.87        34

    accuracy                           0.85        60
   macro avg       0.85      0.85      0.85        60
weighted avg       0.85      0.85      0.85        60

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29357ed00>
```



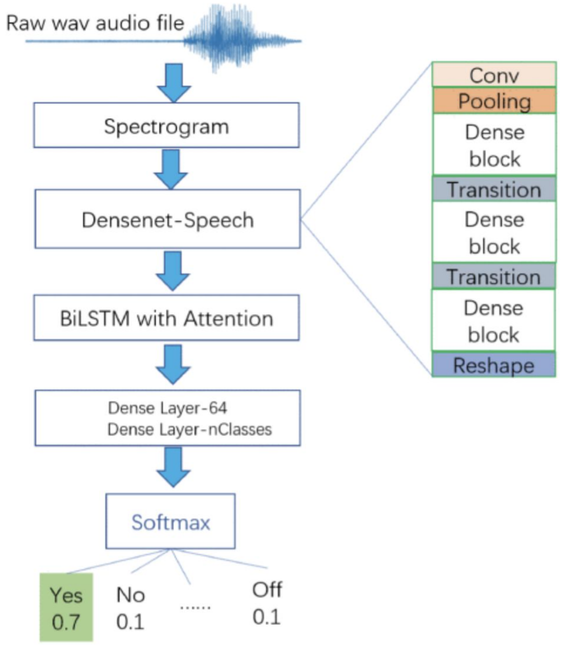
Model



Model 1 - A neural attention model for speech command recognition
https://arxiv.org/pdf/1808.08929v1.pdf



Model 2 - Wav2KWS: Transfer Learning From Speech Representations for Keyword Spotting
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8607038

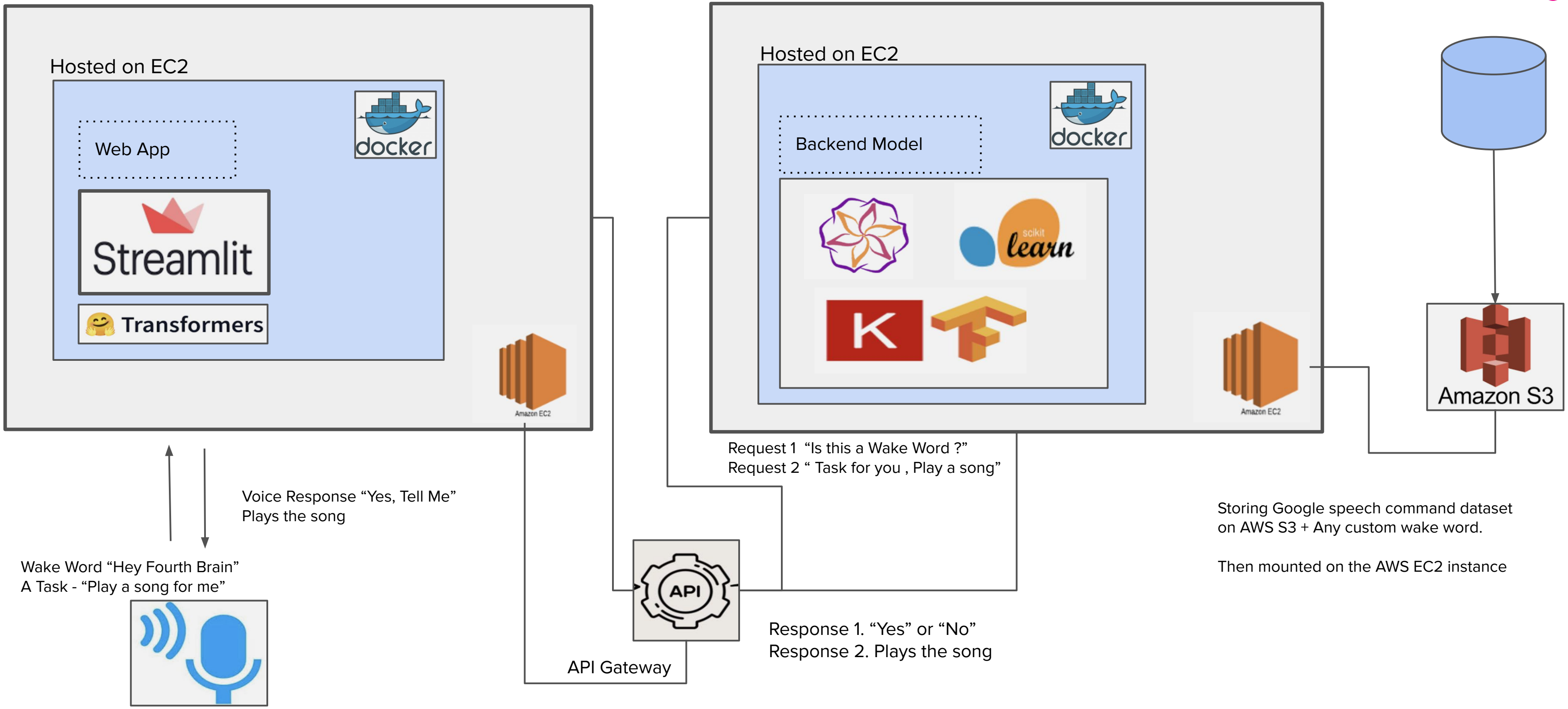


Model 3 - DenseNet and BiLSTM for Keyword Spotting
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8607038

Challenges

- Lack of experience with audio dataset
- Deciding on which model to use
- Inference is not working correctly
- Finalizing inference method - Predict on recorded/streaming audio
- Lack of backend knowledge
- Not enough time

MLE Stack



Hosted on EC2
Web App
docker
Streamlit
Transformers
Amazon EC2
Hosted on EC2
Backend Model
docker
scikit learn
K
Amazon EC2
Amazon S3
Request 1 “Is this a Wake Word ?”
Request 2 “ Task for you , Play a song”
Voice Response “Yes, Tell Me”
Plays the song
Wake Word “Hey Fourth Brain”
A Task - “Play a song for me”
API
API Gateway
Response 1. “Yes” or “No”
Response 2. Plays the song
Storing Google speech command dataset
on AWS S3 + Any custom wake word.
Then mounted on the AWS EC2 instance

Demo (2 min)

… & screen share well!

Conclusions (90 s)

- We learned to do end-to-end ML the easy way, the hard way
- Let us tell you about it!
- Here’s a tip or two for anyone who tries to walk down a similar path!
- And the biggest lesson we’re taking with us into the future is …

Future Work (30 s)

- Given those conclusions and lessons...
- here's what we would do next...
- in rank order...
- and here's why
- That's a wrap!

Thank You! Questions?