Blackboard Writing Audio Recognizer

- Term project Proposal -

20140486 임휘준
20160042 구인용
Group5 20186126 김현지

Motivation

" Can we know what letter is he / she writing, without looking? "

Procedure

1

Data Collection

2

Pre-Processing

3

Feature Extraction

4

Sound Classification

5

Interface

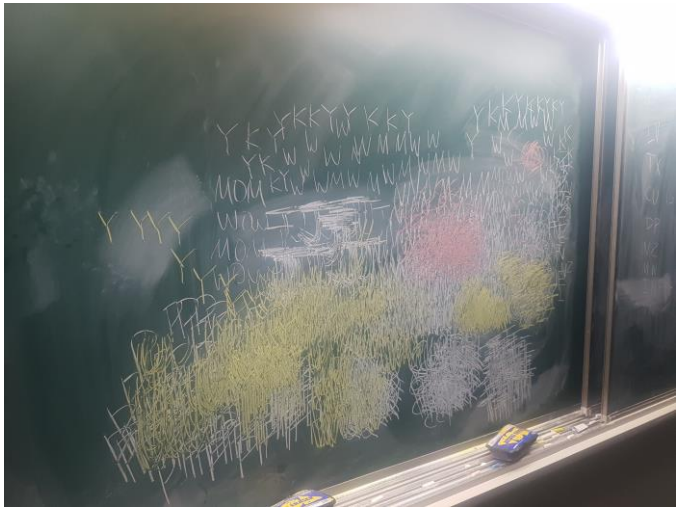
6

Word Validation

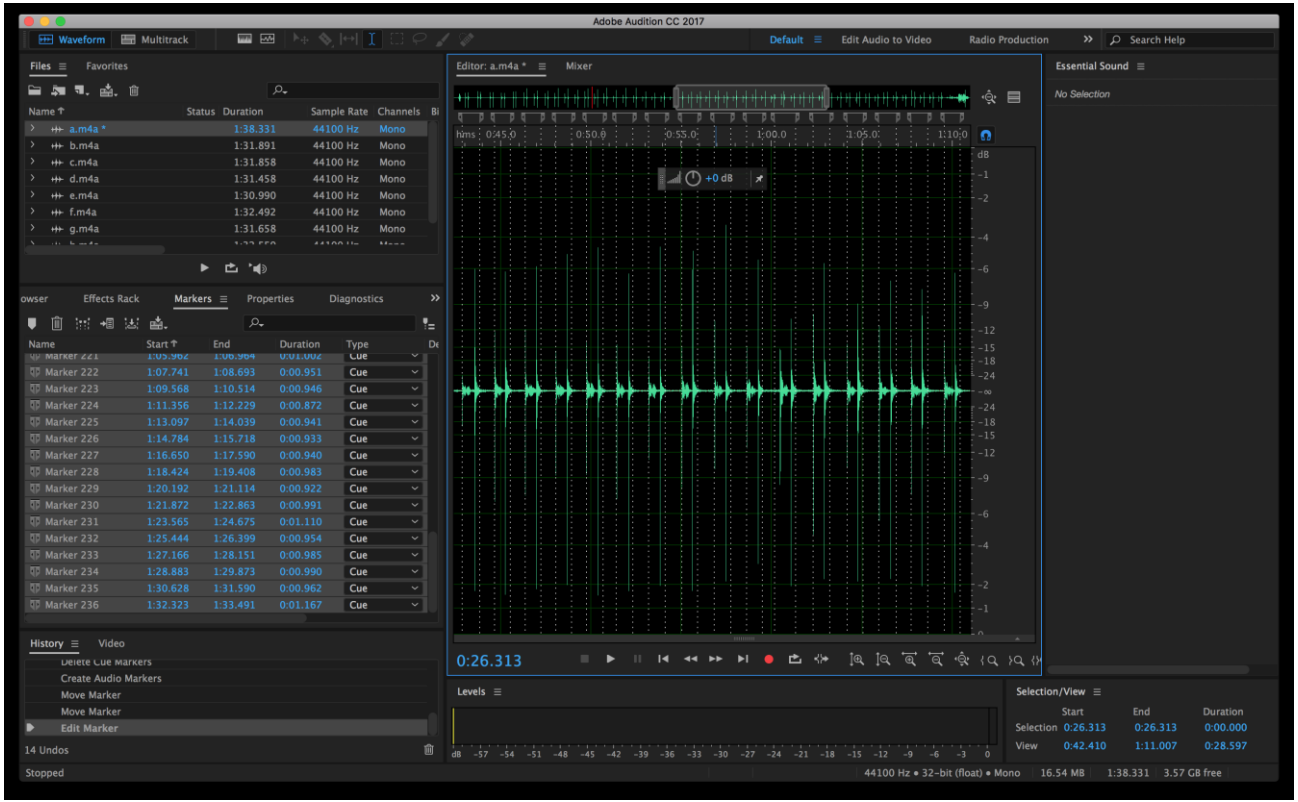
Task done by 구인용 임휘준 이현지

Data collection

1. Specify on upper case alphabet (‘A’, ‘B’, ‘C’ …)
2. Collect 5 minutes of writing sound data per letter
   → Acquires about 160 sound samples per letter (Total 4,179 files)

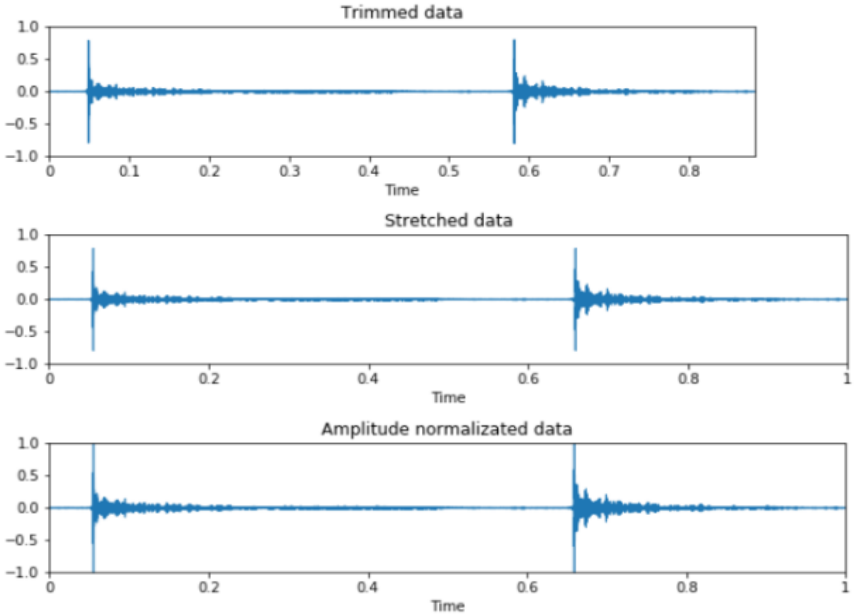


Pre-processing – Sound clipping



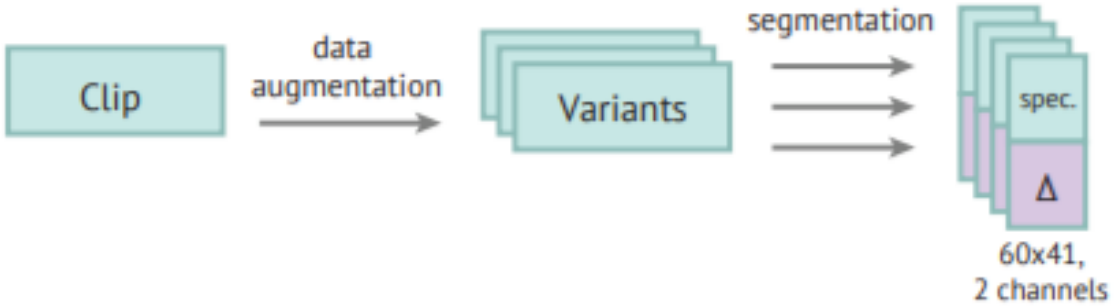
Preprocessing - Data normalization

1. Temporal normalization to 1 second by resampling (via python *librosa* library)
2. Amplitude normalization to range [-1, 1]



Feature extraction

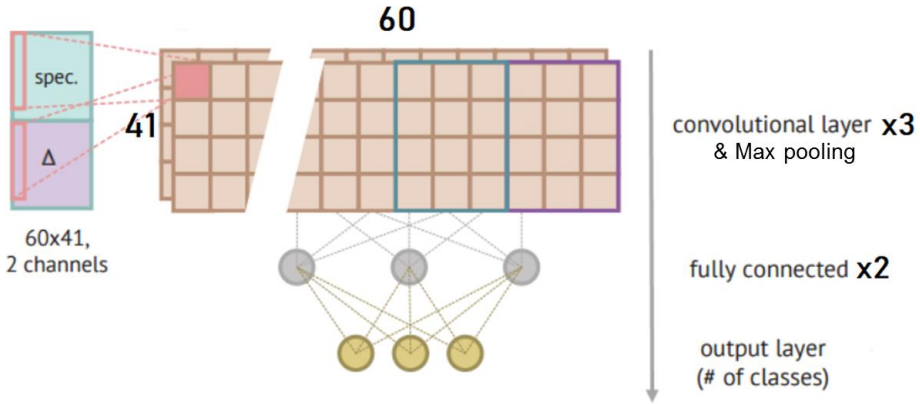
1. Divide each sound clip into (60, 41) windows
2. Calculate log scaled mel-spectrograms and their corresponding deltas from a sound clip (via python *librosa* library) → 2 channels
3. CNN input: (?, 60, 41, 2)
4. Train data: 70 % of total data, Test data: 30 % of total data

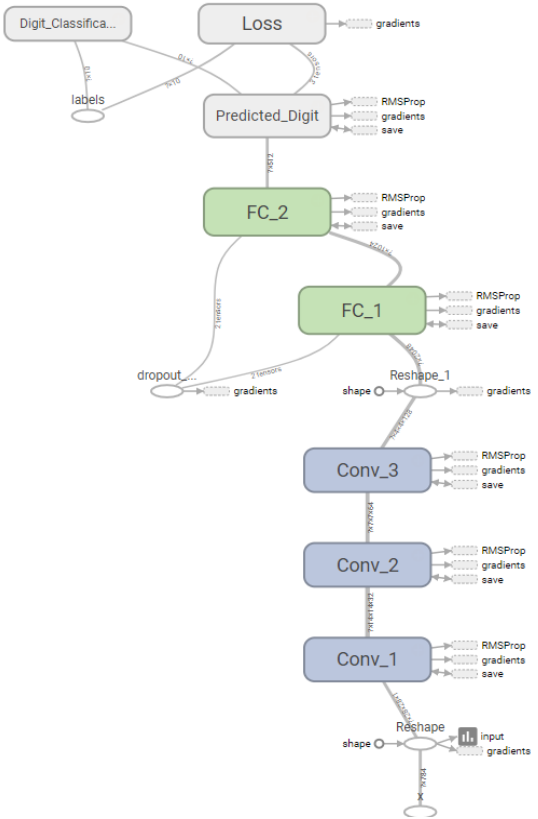


Karol J. P. (2015). Environmental sound classification with convolutional neural networks. *IEEE International workshop on machine learning for signal processing*, 2015, *17-15*

Sound classification

Using Convolutional Neural Network (CNN)

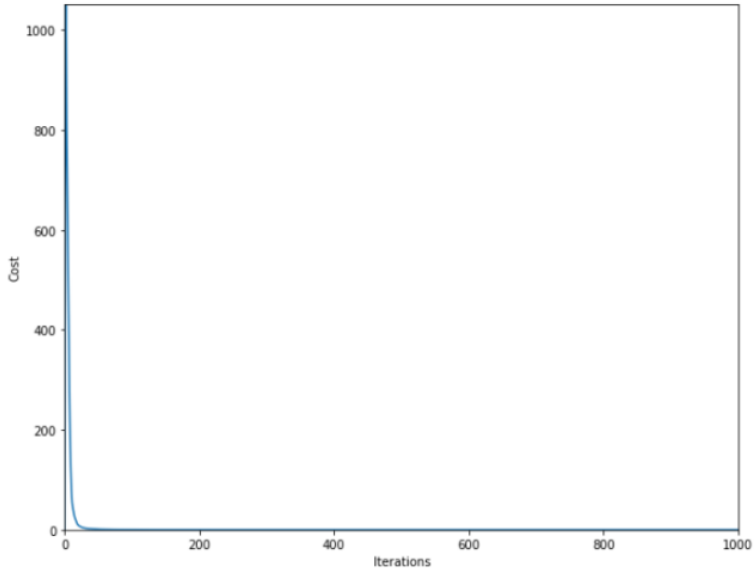




Sound classification - Result

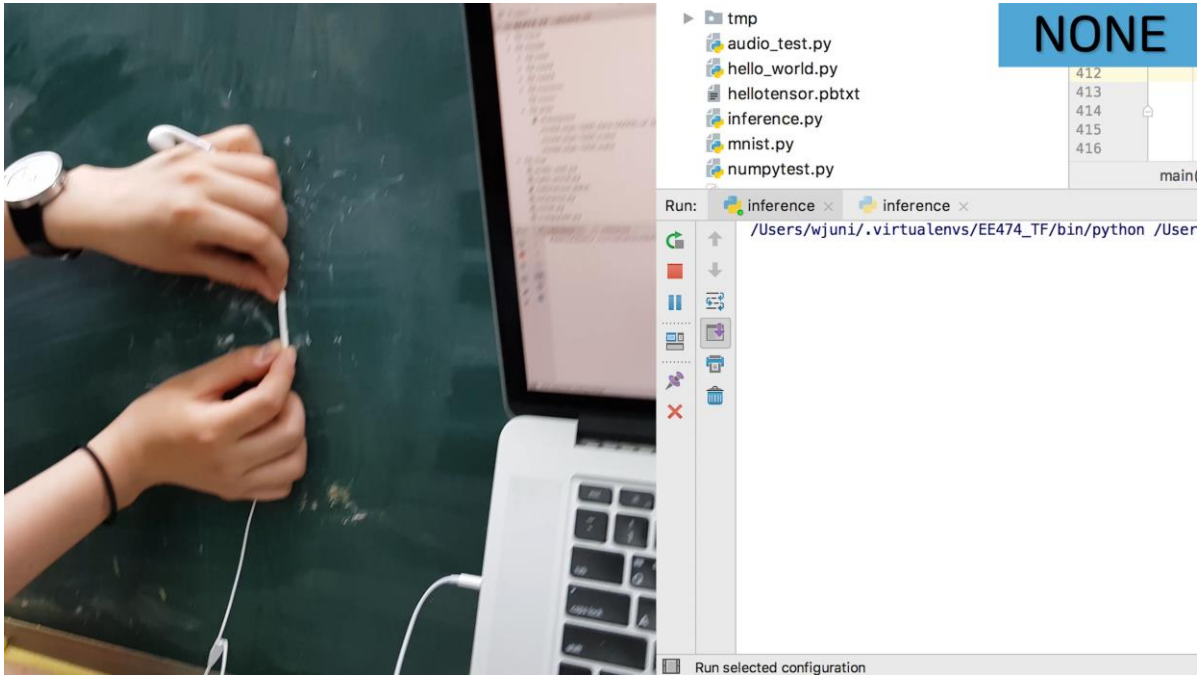
```
@ iteration: 900, Training Acc. = 1.000000, Test Acc. = 0.866821
@ iteration: 910, Training Acc. = 1.000000, Test Acc. = 0.865281
@ iteration: 920, Training Acc. = 1.000000, Test Acc. = 0.862971
@ iteration: 930, Training Acc. = 1.000000, Test Acc. = 0.862971
@ iteration: 940, Training Acc. = 1.000000, Test Acc. = 0.862202
@ iteration: 950, Training Acc. = 1.000000, Test Acc. = 0.866821
@ iteration: 960, Training Acc. = 1.000000, Test Acc. = 0.866821
@ iteration: 970, Training Acc. = 1.000000, Test Acc. = 0.860662
@ iteration: 980, Training Acc. = 1.000000, Test Acc. = 0.862202
@ iteration: 990, Training Acc. = 1.000000, Test Acc. = 0.868360
@ iteration: 1000, Training Acc. = 1.000000, Test Acc. = 0.864511
```

A to Z (26 classes)



Cost graph

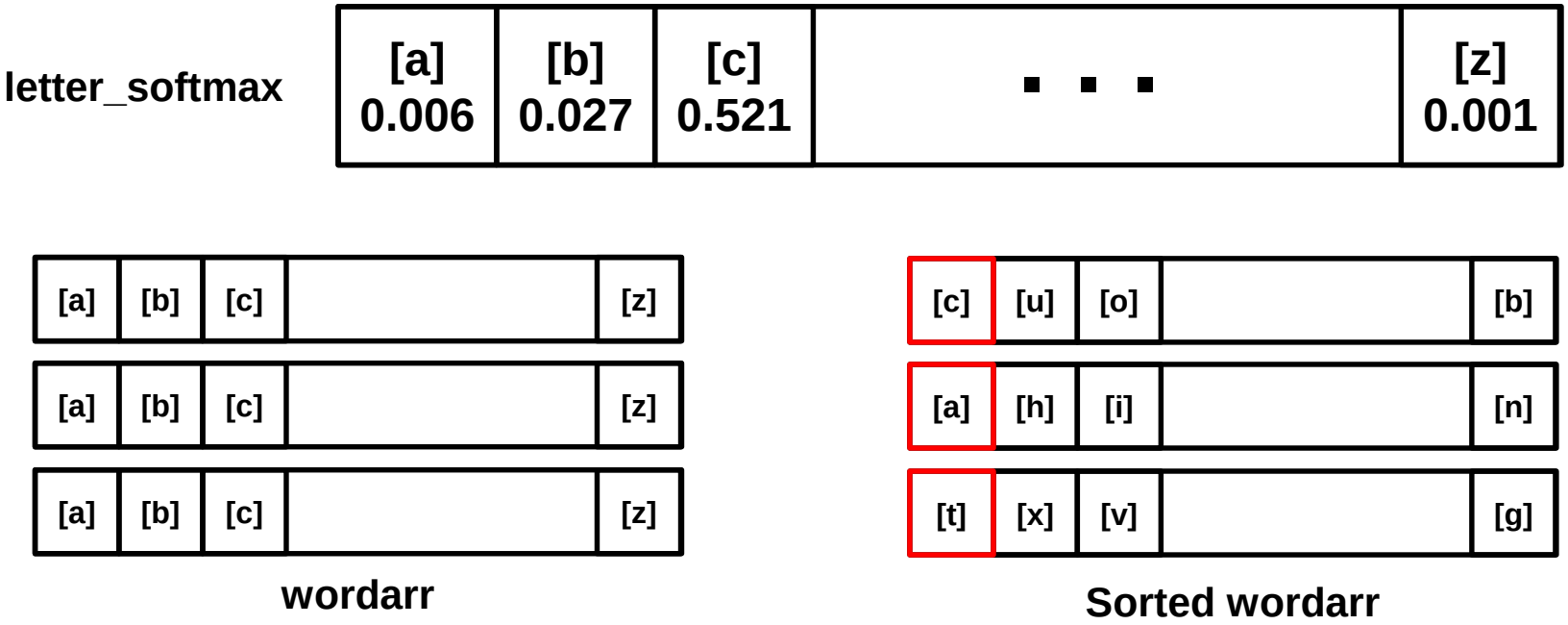
Word validation



- Written word : ‘NONE’
- Prediction : ‘Z’, ‘U’, ‘Z’, ‘E’
- Word validation result : [‘DOZE’, ‘NODE’, ‘NONE’] ← Found

Word validation – Searching rule

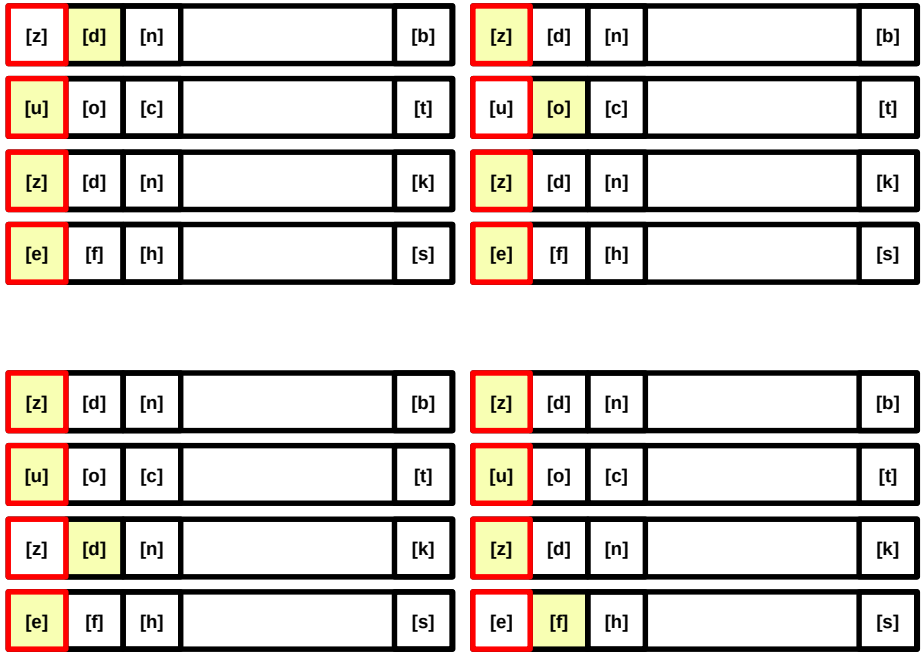
- Pyenchant package : a spell checking library



Word validation – Searching rule

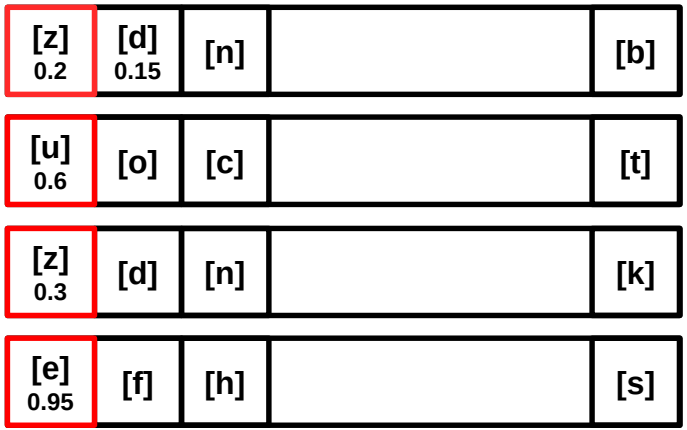
1. Look up the most probable letter combination
2. Try combinations by changing each letter to next probable letter

Sorted wordarr



Word validation – Searching rule

1. Look up the most probable letter combination
2. Try combinations by changing each letter to next probable letter
3. Replace the most probable letter (a) that has least probability to next probable letter (b).
   (add probability of (a) to probability of (b), and set probability of (a) to zero.)



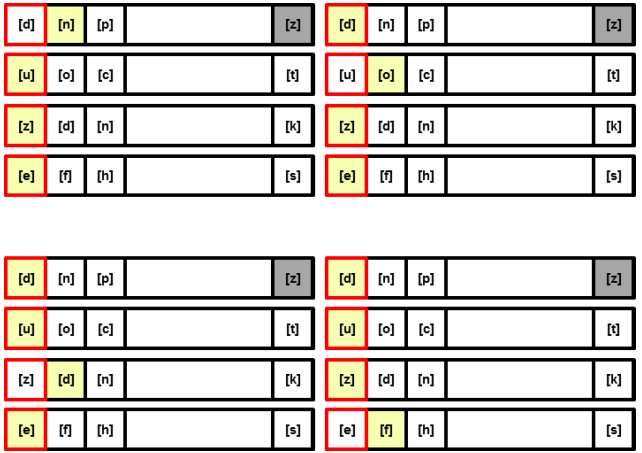
Sorted wordarr

Sorted wordarr

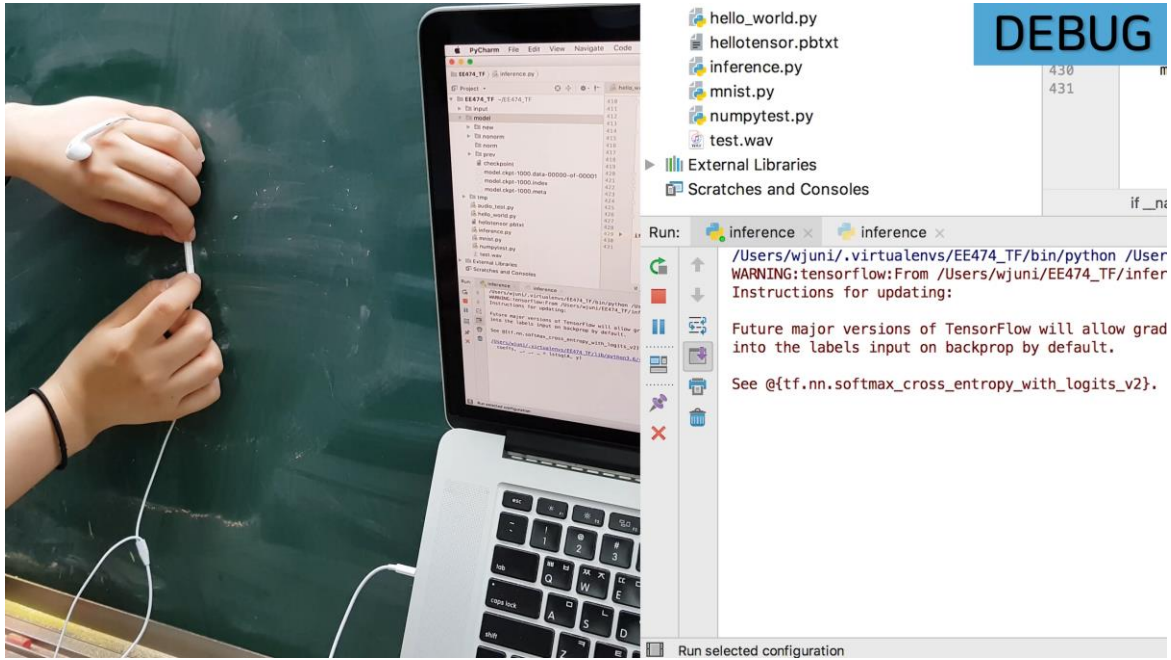
Word validation – Searching rule

1. Look up the most probable letter combination
2. Try combinations by changing each letter to next probable letter
3. Replace the most probable letter (a) that has least probability to next probable letter (b)
   (add probability of (a) to probability of (b), and set probability of (a) to zero)
4. Repeat 1-3 until there are 5 valid words

Sorted wordarr



Word validation



- Written word : ‘DEBUG’
- Prediction : ‘G’, ‘E’, ‘E’, ‘O’, ‘A’
- Word validation result : [‘DEBUG’, ‘DEWED’] ← Found

Future work

- In real-time, classification is not more accurate than expected.
- Collect more data
- Improve the performance of the dictionary

  Search from list containing less words

  • Select the alphabet randomly as softmax probability

Future work

label	incorrect prediction
a	{'g': 1, 't': 1, 'e': 1}
b	{'y': 1, 't': 1}
c	{'u': 1, 'o': 1}
d	{'j': 1, 'b': 4, 'a': 1, 'g': 1, 'r': 1}
e	{'a': 1, 'f': 2}
f	{'i': 2}
g	{'a': 1}
h	{'i': 6, 't': 1}
i	{'h': 2, 'f': 1}
j	{'l': 1, 't': 1, 'p': 2, 'k': 3, 'i': 1, 'd': 1}
k	{'j': 1, 'p': 3, 'y': 4, 't': 1}
l	{'s': 3, 'w': 1, 'u': 2, 'o': 1}
m	{'j': 1, 'n': 2}
n	{'w': 3, 'a': 1, 'v': 2, 'm': 1, 's': 1}
o	{'z': 1, 'v': 3, 's': 3, 'u': 3, 'l': 1}
p	{'x': 2, 'k': 4, 'q': 1, 'd': 1, 'j': 3, 'y': 2, 'i': 1, 'r': 1, 't': 2}
q	{'l': 1}
r	{'n': 2, 't': 1, 'm': 1, 'q': 1}
s	{'o': 7, 'm': 2, 'l': 3, 'n': 2, 'c': 1, 'k': 1}
t	{'r': 2, 'c': 1, 'j': 2, 'n': 1, 'a': 1}
u	{'n': 3, 'o': 4, 'v': 11, 'l': 2, 'm': 1}
v	{'l': 5, 'u': 3, 'x': 1, 'y': 1}
w	{'m': 2, 'v': 1}
x	{'t': 3, 'y': 1, 'p': 2, 'q': 1}
y	{'k': 1, 'i': 1, 'r': 2, 'm': 2, 't': 1}
z	{'w': 2, 'n': 1, 'm': 1}

- Letters with similar writing structure -> one group -> reduce # of classes
  - F, H, I
  - S, O
  - V, U, L
  - K, Y

Reference

- Feature extraction:
- Karol J. P. (2015). Environmental sound classification with convolutional neural networks. *IEEE International workshop on machine learning for signal processing*, 2015, *17-15*
- *https://github.com/aqibsaeed/Urban-Sound-Classification*
- Python Library / Packages
- *https://librosa.github.io/librosa/*
- *https://faculty.math.illinois.edu/~gfrancis/illimath/windows/aszgard_mini/movpy-2.0.0-py2.4.4/manuals/PyEnchant/PyEnchant%20Tutorial.htm*