TensorFlow Speech Recognition Kaggle Challenge

그데목 Geudemog

고정욱 노범용 박상훈 양영규 최윤철

Sponsored by TensorFlow-KR

INTRODUCTION

TensorFlow Speech Recognition Challenge

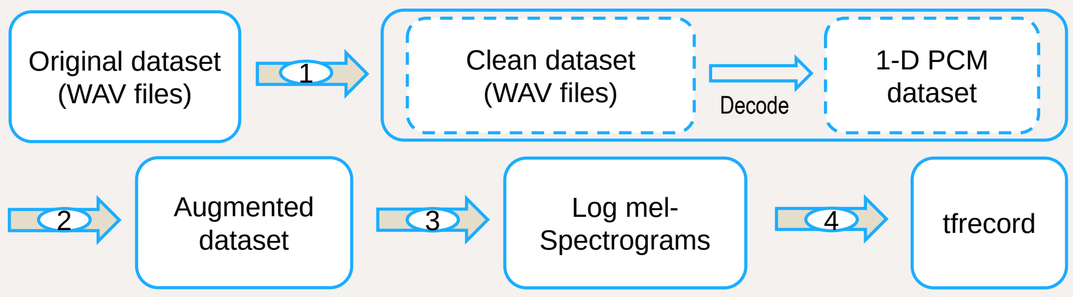
- **Can you build an algorithm that understands simple speech commands?**

Tensorflow에서 제공하는 음성 데이터,
64,728개의 train data(1초 길이의30가지 labeled data와 1분 길이의 6종류의 background data)와 158,538개의 test data를 12가지 label(yes, no, up, down, left, right, on, off, stop, go, silence, unknown)로 분류예측하는 Kaggle challenge project. (Scoring metric: Multiclass Accuracy)

개발 환경

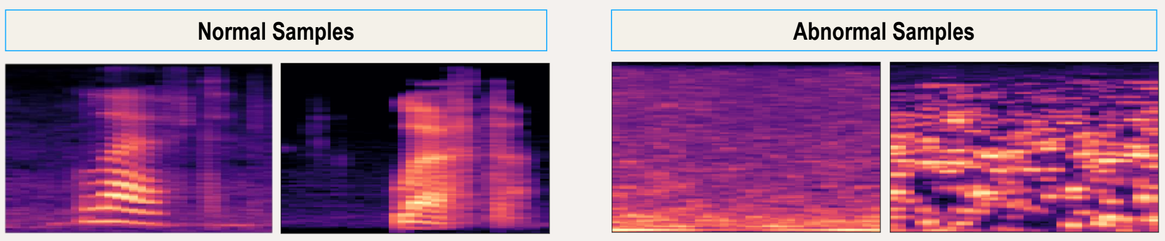
- 메인 서버: ec2 p3.2xlarge
  (GPU: 1, CPU: 8, 메모리: 61G, GPU 메모리: 16G, OS: Ubuntu 16.04.3)
- 서브 서버: gcp
  (GPU: 1 * NVIDIA Tesla K80, CPU: 8, 메모리 : 32G, OS: Ubuntu 16.04)
- 언어: python 3.6.3 Anaconda custom (64-bit)
- 주요 라이브러리: tensorflow 1.4, librosa 0.5.1, numpy 1.13.3, pandas 0.21.0

DATA PREPROCESSING



1. Removing Abnormal Data

- Abnormal Data
  - Definition: label과 일치하지 않는 내용의 data.
    1) Case 1: 특정 label의 dataset에 noise/silence data가 포함된 경우.
    2) Case 2: 특정 label의 dataset에 다른 label의 data가 포함된 경우.
- 1차 Filtering: Mel-spectrogram을 육안 검사. noise/silence data는 차이가 명확하게 나타남.



- 2차 Filtering: 1차 Filtering 결과들을 직접 청음하여 제거.
  - 상기와 같이 Mel-Spectrogram에서 육안으로 명확한 차이가 나타나는 data는 대부분 noise/silence data
  - 1, 2차 Filtering 과정을 통해 case 1에 해당하는 abnormal data는 대부분 제거 가능했으나, case 2의 경우는 완전히 제거하는 것이 불가능.

2. Data augmentation

- **Speed tuning**
  - Speed 변경 범위는 원래 속도의 0.9 – 1.1배 범위에서 random으로 선택.
  - Speed 변경 범위를 더 확대할 경우, speech event가 일어나는 부분의 data의 손실이 커질 수 있다고 판단.
- **Pitch tuning**
  - 한 옥타브를 24개 step으로 나누고 pitch에 [-4, 4] 범위에서 random으로 뽑은 step을 original data에 더해 pitch를 변화시킴.
- **Background noise mixing**
  - 주어진 background noise 6가지 중 1가지를 임의로 선택하고, 선택 noise에서 1초 단위로 임의의 구간을 추출, train data에 더하는 방식으로 mixing.

이상의 3가지 방법을 조합하여 8배로 data augmentation.

조합 순서까지 고려하는 경우 최대 16배까지 augmentation이 가능했으나, 데이터 용량 및 모델 학습 시간을 고려하여 8배 augmentation으로 마무리.

Speech event의 start & end point를 detection 하여 trimming 하는 경우, data 손실 없이 speed tuning 및 time shift가 가능하다고 생각되었으나 실행하지는 못했음.

3. Making log mel-spectrograms

- **STFT parameter setting**
  - 성대의 떨림 정도나 모양이 급격하게 변할 수 없으므로 음성은 짧은 구간(보통 20ms)동안의 주기적인 특성(quasi-stationary)으로 가정함.
  - 초기에는 위 domain 지식이 없어 librosa.feature.melspectrogram의 default setting(win_length 128ms에 해당)을 적용하여 시간에 따른 주파수 특성이 지나치게 smoothing된 mel-spectrogram을 학습에 사용하는 시행착오가 있었음.
  - Frequency 축은 126으로 축소하여 사용 (n_mels = 126)

MODEL

Convolutional + LSTM Model

- Raw Wave를 Mel-spectrogram으로 변환.
- Mel-spectrogram을 convolution layer를 통과시켜 Feature map들을 생성.
- 생성된 feature map들을 시간 기준으로 여러 개의 시계열 vector들로 resize.
  [Amplitude, Time, n_channels] -> [Time, Amplitude x n_channels]
- 시계열 Vector들을 Many to one방식의 RNN-LSTM에 학습.
- 생성된 벡터를 SOFTMAX 함수로 classification.

기대효과

- 시계열 데이터이므로, Convolution layer를 통해 특징들을 잡아내는 것에 그치지 않고, Recurrent layer를 통해 CNN만으로는 잡지 못하는 시계열 특징을 반영

Figure. CNN + LSTM

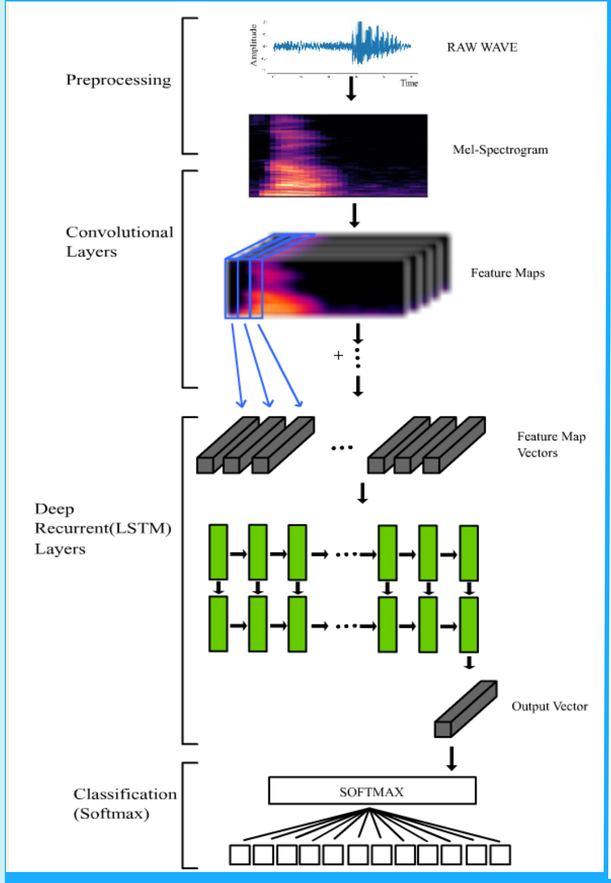
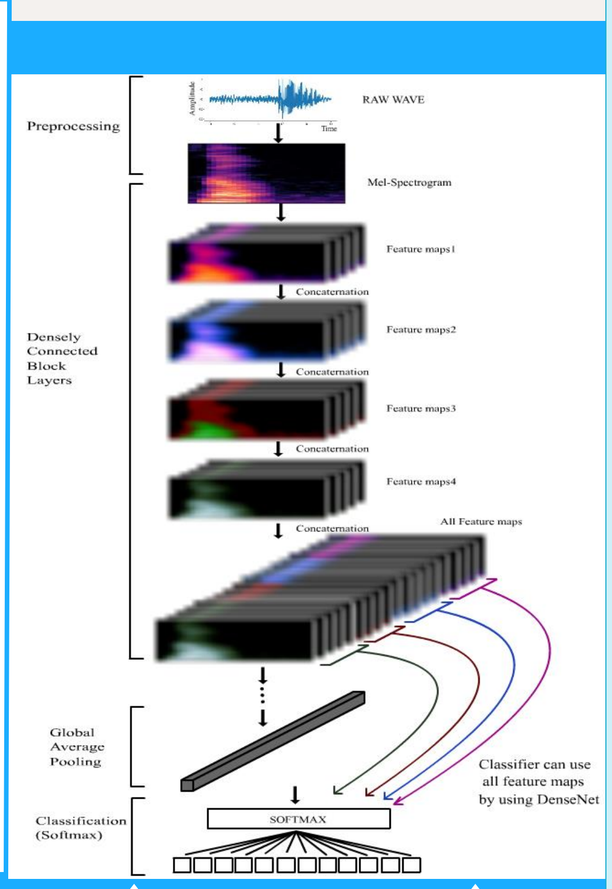


Figure. DenseNet



DenseNet Model

Growth_rate(k):
각 layer가 생성하는 channel 수.

Composite Layer:
계산 효율을 위해 Bottelneck 구조 사용.
[BN → ReLU → conv1x1 (4k로) → BN → ReLU →conv3x3]

Transition Layer:
feature map size줄여주고, compression factor theta에 따라 채널 압축.
[BN --> conv11 --> avg_pool2]

- 여러 개의 composite Layer를 묶어서 하나의 Dense Block 생성.
- 여러 개의 Dense block 생성.
- Dense Block 사이에 Transition Layer 통과.
- 생성된 feature map들을 Global Average Pooling.
- 생성된 벡터를 SOFTMAX 함수로 classification.

기대효과

- 생성되는 Feature map들의 특징들을 고려하며 깊은 신경망 구조의 분류기 생성 가능.
- 이미지 인식에 있어서 가장 좋은 알고리즘 중 하나이므로, Spectrogram 이미지 인식에도 우수한 결과를 기대.

CONCLUSION

모델	설명	점수
CNN	Convolution layer 4층, Dense layer 2층으로 구성된 신경망	0.82767
4Conv + 1LSTM	Convloution layer 4층, LSTM layer 1층으로 구성된 신경망	0.87266
6Conv + 2LSTM	Convloution layer 6층, LSTM layer 2층으로 구성된 신경망	0.87677
Densenet(121)	총 58개의 block으로 구성되어 총 121층인 신경망	0.87231
Bagging	모델들을 단순 평균하여 앙상블한 모델	0.88546
Minmax-median	모델들의 predict_proba가 모두 높거나 낮으면 채택, 그렇지 않으면 중앙값 채택	0.88793

- Convolution layer에서 더 나아가 LSTM layer를 통해 시계열 특징 잡아내는데 성공.
- 튜닝을 하지 않은 Densenet보다 Convolutional + LSTM 모델이 성능이 좋음.
- 기본 Bagging Model보다 우수한 앙상블 기법 사용.
- 모든 모델들이 강하게 확신하고 있는 부분 (0.8이상, 0.2 이하) 은 가장 높거나 낮은 predict_proba 채택, 그렇지 않으면 중앙값(median)을 predict_proba로 사용.
- **최종 점수 : 0.88793, 리더보드 : 132/ 1315 위 달성.**