Pusan National University
Computer Science and Engineering
Technical Report 2021-53

소음 환경에 강한 음성인식 모델 개발

포네틱 코드
201524495 안준수
201661701 강동민

지도교수 백윤주

목 차

1. 서론

1.1 연구 배경

음성은 사람 간의 가장 자연스러운 의사소통 방식이다. 음성인식 기술은 이미 50년 전부터 지속적인 연구가 이루어지고 있는 분야이다. 특히 종래의 음성인식 기술은 낭독체 음성인식 기술이 주로 연구 대상이었으나, 딥러닝 및 잡음처리 기술의 발전으로 인해 현재는 사람 간의 자연스러운 대화 음성을 대상으로 기술 고도화가 이루어지고 있다. 특히 최근 몇 년 동안 보편화 되는 클라우드 서버 및 고성능 GPU와 같은 하드웨어의 눈부신 발전에 힘입은 딥러닝 기술에 의해, 혁신적인 성능을 보이는 음성인식 기술이 등장하고 있다. 음성인식 기술은 스마트폰, 자동차, 콜 센터 등 현재 우리 생활의 많은 부분에 녹아들어서 서비스화되고 있다.

1.2 기존 문제점

이러한 음성인식 기술의 비약적 발전에도 불구하고 여전히 음성인식은 사람들이 웅성거리는 식당, 회의실, 버스나 지하철 등과 같은 소음이 발생하는 환경에서는 매우 낮은 정확도를 보이는 등 개선할 점이 많다. 실생활에서 음성인식 기술이 원활하게 사용되려면 실생활에서 자주 발생하는 소음에 대한 내성이 있어야 할 것이다.

아래의 [표 1]은 우리가 직접 녹음 및 소음 합성한 파일을 Kakao STT REST API 서비스를 이용하여 인식시켜 본 결과이다. 인식 음성은 적절한 길이를 갖춘 문장으로 택하였다. 구체적인 문장 내용은 '국가는 재해를 예방하고 그 위험으로부터 국민을 보호하기 위하여 노력하여야 한다.'이다.

이렇듯 원본 음성은 완벽하게 잘 인식하였지만, 소음이 있는 환경에서는 음성인식에 오인식 및 미인식 등의 상당한 오류가 발생한다. 이에 소음 환경에서의 원활한 음성인식을 제공하는 기술 및 서비스를 개발하고자 한다.

비고	출력내용
원본	국가는 재해를 예방하고 그 위험으로부터 국민을 보호하기 위하여 노력하여야 한다.
대화 도중 양치질하는 소음	국가는 재해를 예방하고 그 위험으로부터 국민을 보호하기 위하여 노력하여야 한다.
샤워장의 물이 떨어지는 소음	북한의 대회로 예방하고 그 위험으로부터 국민을 보험위하여 노역하여야 한다.
스포츠 경기장의 환호 소음	인식 불능
대중들이 야유하는 소음	국가는 재해를 예방하고 위험도 있고 민의 고향이에요.거려야 한다.

[표 1] Kakao STT REST API 소음 환경에서의 테스트 결과
(node.js REST fefch 이용)
원본과 다른 출력 부분은 빨강으로 표시했다.

1.3 연구 목표

본 졸업과제는 소음이 학습된 음성인식 모델을 사용해 다양한 소음 환경에서도 정상적으로 작동하는 음성인식 프로그램을 만드는데 목표를 둔다.

- 소음이 있는 다양한 파일 제작
  - ▶ 음성 파일과 소음 파일을 합성시킨다.
  - ▶ 소음 파일은 실생활에서 얻을 수 있는 소음이거나 인위적으로 만든 소음으로 한다. (Google Audio Set 등의 Dataset 이용)
  - ▶ 자연어 처리가 아닌 명령어 처리로 진행한다.

- 다양한 소음 환경 중 어떤 소음이 음성인식을 가장 어렵게 하는지 판단
  - ▶ Kakao STT 등 음성 파일을 텍스트로 변환시켜주는 API 및 서비스를 이용해 합성된 음성 파일을 얼마나 잘 인식하는지 확인한다.
  - ▶ 결과를 확인하고 인식에 가장 취약한 소음을 찾는다.

- 취약한 소음 환경에 내성을 가진 모델을 제작
  - ▶ TensorFlow를 이용해 다양한 합성 데이터 파일로 학습을 시키고 모델을 제작한다.

- 음성인식 프로그램을 만들어 실시간으로 사용할 수 있게끔 구현
  - ▶ 학습된 모델을 임베디드 기기에 적용하여 실시간 소음 내성 음성인식이 가능하도록 구현한다.
  - ▶ 실생활에서 인식이 잘 되는지 확인한다.

2. 과제 배경 지식

2.1 Tensorflow

TensorFlow는 Google 브레인 팀에서 개발한 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 End-to-End 오픈소스 소프트웨어 플랫폼이다. 심볼릭 수학 라이브러리이자, 딥러닝, 머신러닝 등의 기계학습 응용 프로그램에도 사용되는 TensorFlow는 Python, C++, Java 등 다양한 언어에 대해 API를 지원하기 때문에 사용 언어에 크게 제약을 받지 않아서 전 세계적으로 활발하게 쓰이고 있다.

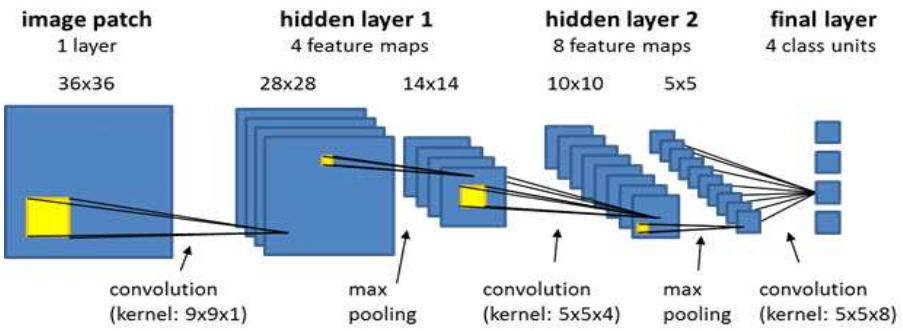
2.1.1 Keras Module

즉각적인 '모델 반복' 및 쉬운 디버깅 기능을 가능하게 하는, 즉시 실행 기능이 포함된 Keras와 같은 API를 사용하여 ML 모델을 쉽게 빌드하고 학습시킬 수 있다. 데이터 플로우 그래프를 사용하여 시각화하기 편하고 다양한 추상화 라이브러리와 혼용해서 사용 가능하다는 장점이 있다. 또한, 유연한 아키텍처로 구성되어있어 코드의 수정 없이 데스크톱, 서버 혹은 모바일 디바이스에서 CPU나 GPU를 사용하여 연산을 구동할 수 있다.

2.1.2 Neural Network

(1) CNN (Convolution Neural Network)

CNN은 DNN(Deep Neural Network)의 문제점을 해결하기 위해 고안되었다. 이미지 분석에서 CNN는 훌륭한 성과를 보여준다. 이미지를 날것 그대로 받음으로써 공간적/지역적 정보를 유지한 채 특성들의 계층을 빌드업 한다. 이미지 전체를 보는 것, 이미지의 한 픽셀과 주변 픽셀들의 연관성을 살리는 특징이 있다.



[그림 1] CNN 모델 예시

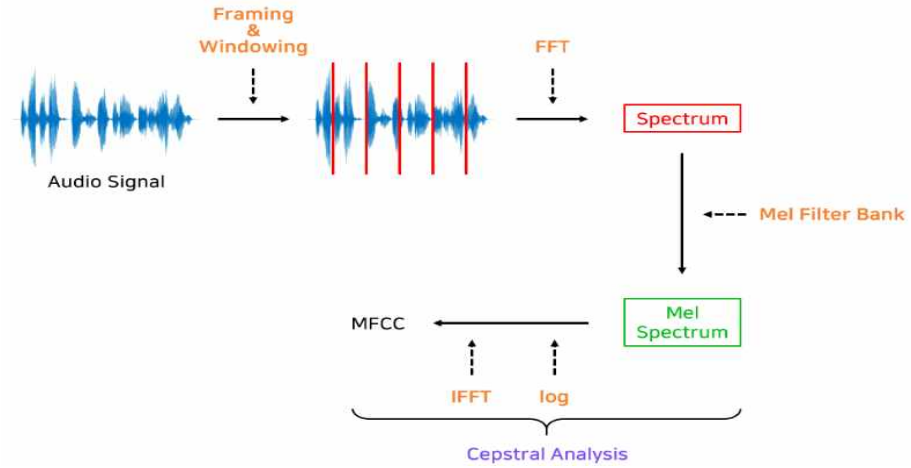
[그림 1]은 특정 네트워크 아키텍처를 가진 신경망 클래스를 가지며, 각각의 은닉층에는 일반적으로 두 개의 별개의 레이어가 있다. 첫 번째 단계는 이전 레이어의 Local Convolution의 결과다. 두 번째 단계는 Max-Pooling 단계로, 첫 번째

단계의 여러 단위의 최대 응답만 유지하여 단위 수를 크게 줄인다. 여러 은닉층 이후에 최종 레이어는 일반적으로 완전히 연결된 레이어다. 그것은 네트워크가 예측하는 각 클래스에 대한 단위를 가지며, 이러한 각 단위는 이전 계층의 모든 단위로부터 입력을 받는다.

2.2 MFCC (Mel-Frequency Cepstral Coefficient)

Mel-Frequency Cepstral Coefficient는 오디오 신호에서 추출할 수 있는 feature로, 소리의 고유한 특징을 나타내는 수치다. 주로 음성인식, 화자 인식, 음악 장르 분류 등 오디오 도메인의 문제를 해결하는 데 사용된다.

먼저 오디오 신호를 프레임별로 나누어 고속 푸리에 변환을 적용해 Spectrum을 구한다. 다음 Spectrum에 Mel Filter Bank를 적용해 Mel Spectrum을 구한다. 마지막으로 Mel Spectrum에 Cepstral 분석을 적용해 MFCC를 구한다. [그림 2]



[그림 2] MFCC 추출 과정

3. 연구 내용

3.1 시나리오 구성 및 데이터 수집

3.1.1 시나리오 구성

어떤 환경에서 사용하고, 어떤 소음에 강한 모델을 만들어야 할지 고민을 했다. 최근 이슈에 맞춰 3가지 조건에 부합하는 주제를 선정하기로 했다.

- 공공장소를 비롯해 범용적으로 활용 가능한 모델
- 최근 코로나 사태로 인한 비대면 서비스 인프라 수요에 부합
- 신체 접촉 없이 사용 가능

키오스크와 자판기 같은 벤딩 머신은 위 조건을 만족하는 것으로 생각되기 때문에 주제로 선정했다.

3.1.2 개발환경

- 실행 환경 : Windows 10, Raspberry Pi
- 실행 언어 : Python, TypeScript, JavaScript, C++
- 개발 도구 : Tensorflow, Cloud STT API, AudioSegment, PyQt
- 데이터 샘플 : Google Speech Command set, Free Restaurants Sound Effects, 직접 녹음

3.1.3 데이터 수집

- Google_Speech_Commands_v0.02
- 키오스크가 있을 환경과 비슷한 식당 소음 사용

모델에 이용될 Data Set의 수집 및 활용은 최대한 데이터의 객관성 및 대표성, 비 편향성, 선형성, 효율성 등을 고려한다. 이에 우리는 Google Audio Set 등의 상대적으로 앞서 언급한 데이터의 객관성 등이 상당히 확보된 Data Set을 소음 데이터로 사용하고, 마찬가지로 Google Command Data Set과 같은 Data Set을 음성 데이터를 사용한다. 이를 통해 개발의 편의성 및 데이터의 객관성 모두 확보할 수 있다. 또, 학습된 모델을 바탕으로 실제 테스트를 수행 후, 결과에 따라 Data Set의 투입 수준 등을 조정하여 모델을 수정 및 개선한다.

3.1.4 데이터 분석

one, two, three, four, five, six, seven, eight, nine, yes, no, unknown, silence로 명령어를 구성하고, 총 72,090개의 음성 파일을 사용해 모델 제작을 진행

했다.

모든 음성 파일의 시간을 1초, Sampling Rate를 16,000으로 고정한 뒤 unknown과 silence를 제외한 명령어에 40초짜리 식당 소음을 자르는 위치와 소리의 크기를 무작위로 overlay 했다.

모든 파일을 무작위로 섞은 뒤 train set 57,000개, validation set 7,200개, train set과 validation set에 사용되지 않은 파일 4,520개를 test set으로 사용했다.

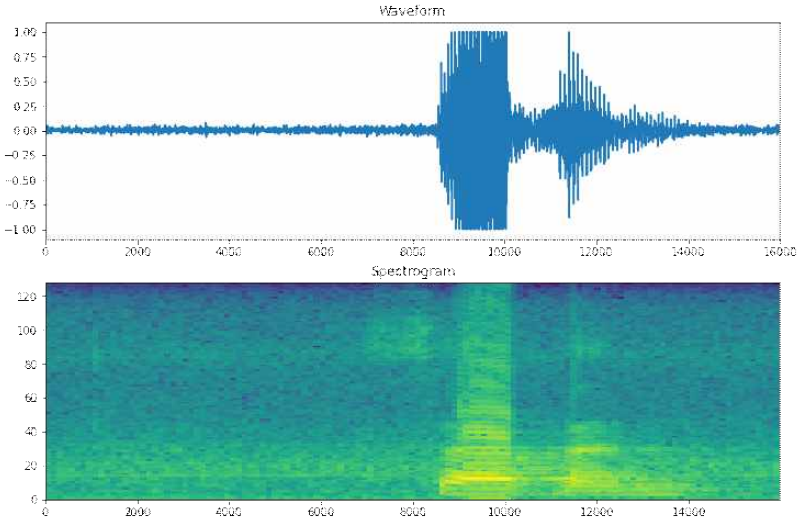
3.1.5 데이터 정규화

[그림 3]과 같이 파형을 시간 경과에 따른 주파수 변화를 보여주고 2D 이미지로 나타낼 수 있는 Spectrogram으로 변환한다. 이는 오디오를 시간-주파수 영역으로 변환하기 위해 단시간 푸리에 변환을 적용하여 수행할 수 있다.

푸리에 변환 성분의 주파수로 신호를 변환하고 있지만, 모든 시각 정보를 잃는다. 단시간 푸리에 변환 시간의 창에 신호를 나누고 푸리에 시간 정보를 보존하고 표준 회선을 실행할 수 있는 2차원 텐서를 반환, 각 창에 변환을 실행한다.

이후 생성된 Spectrogram은 2.2와 같이 MFCC를 구하는 데 사용된다. 3.1.4에서 1초의 시간과 16,000 Sampling Rate를 만족하지 않으면 Argument Error를 불러낸다.

이후 음성을 실시간으로 받을 때도 1초와 16,000 Sampling rate를 맞춰주는 정규화를 진행했다.



[그림 3] Waveform을 Spectrogram으로 변환한 모습

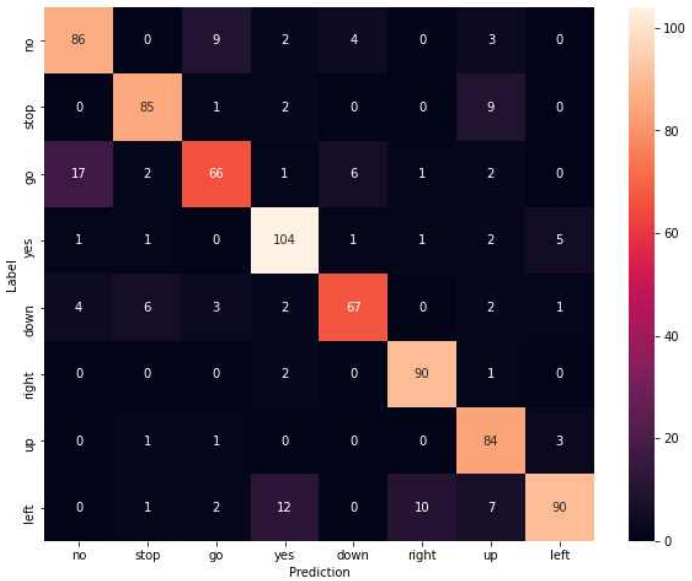
3.2 분석 모델링

3.2.1 초기 모델

(1) CNN 모델

```
model = models.Sequential([
    layers.Input(shape=input_shape),
    preprocessing.Resizing(32, 32),
    norm_layer,
    layers.Conv2D(32, 3, activation='relu'),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_labels),
])
```

[그림 4] CNN 모델



[그림 5] 초기 CNN 모델의 혼동 행렬 결과

[그림 4]를 보면 Resizing 층이 더 빨리 훈련 모델을 사용하려면 입력을 다운 샘플링 했다. Convolution 2D Layer를 추가하고 MaxPooling 2D Layer를 이용해 최대값을 구하는 필터를 통과, 25%의 확률로 Dropout을 하는 layer를 추가했

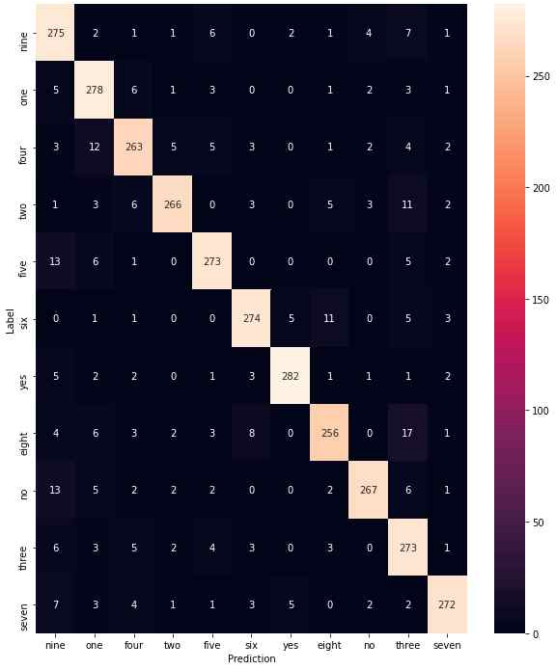
다. Convolution 2D Layer의 필터 개수를 32, 64개로 하며 MaxPooling2D의 default 값은 pool_size는 (2,2)가 되고 strides는 pool_size가 된다.

해당 모델의 데이터 세트는 train set 6,400개, validation set 800개, test set 800개로 총 8:1:1의 비율을 보인다. 초기 Tensorflow의 예제에 있는 코드를 돌리면 [그림 5]와 같은 혼동 행렬을 볼 수 있다. 혼동 행렬은 모델이 테스트 세트의 각 명령에 대해 얼마나 잘 수행했는지 확인하는 데 도움이 된다. 초기 코드의 정확도는 약 84%의 정확도를 보인다. 하지만 실생활에 적용하기 위해선 명령어 외의 단어나 침묵에 대해서 결과를 알 수 있는 모델이 필요하다고 생각이 들어 모델의 성능을 개선하고자 했다.

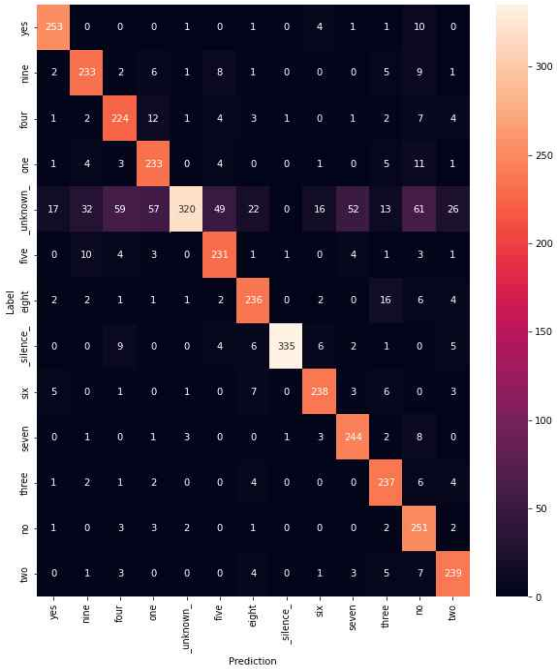
3.2.2 최종 모델

(1) CNN 모델

데이터 세트의 크기가 부족하다고 생각되어 3.1.4를 따른 뒤 수행했다. 이후 모델을 불러오는 과정에 있어 체크포인트를 사용했다. 체크포인트는 모델 트레이닝 도중 weight를 저장하는 것을 뜻한다, 파일을 저장하고 불러옴으로써 학습된 모델을 재사용하고 지난 학습을 이어서 더하고 하는 작업을 가능하게 한다.



[그림 6] 향상된 CNN 혼동 행렬 결과



[그림 7] silence와 unknown을 추가 결과

[그림 6]과 같이 정확도가 향상된 모델을 얻을 수 있었다. [그림 7]과 같이 silence와 unknown을 추가했을 때 정확도가 조금 떨어지는 모습을 보였지만 키오스크 환경을 위해 포함해야만 했다. 우리는 위 모델을 임베디드 기기에 넣은 뒤

결과를 확인하고자 했다.

3.3 임베디드 적용

3.3.1 라즈베리 파이

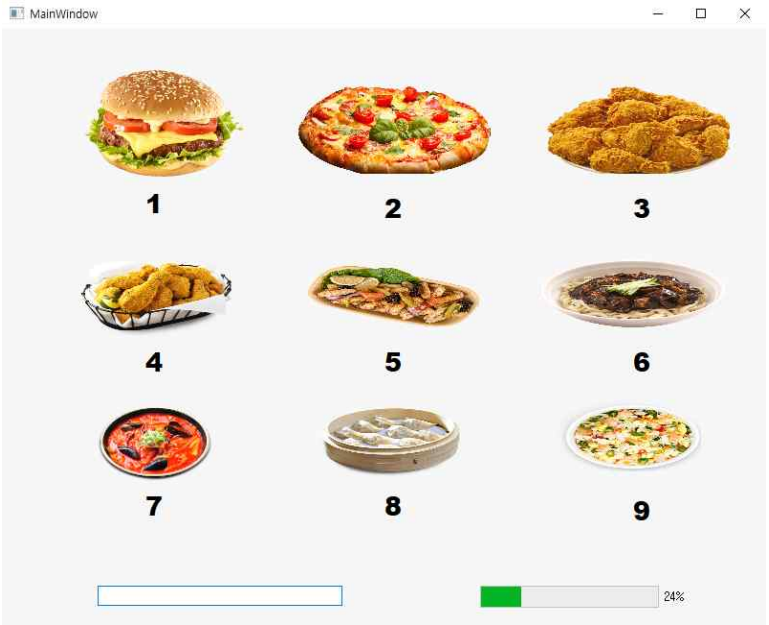
임베디드 기기의 특징은 오랫동안 오류 없이 안정적으로 동작하도록 설계가 되고 물리적 손상에서 비교적 자유로운 칩을 사용한다. 우리는 임베디드 기기 중 라즈베리 파이를 이용했다. 라즈베리 파이의 장점은 단순한 기능과 5V의 저전력, 작은 크기, 저렴한 비용 등이 있다.

3.4 UI/UX 제작

마지막으로 상기한 모델을 좀 더 직관적이고 편하게 사용하기 위해 GUI 기반의 프로그램을 제작했다. PyQt와 STT API를 이용해 키오스크와 유사한 UI/UX를 만들고자 했다.

3.4.1 UI/UX 기능

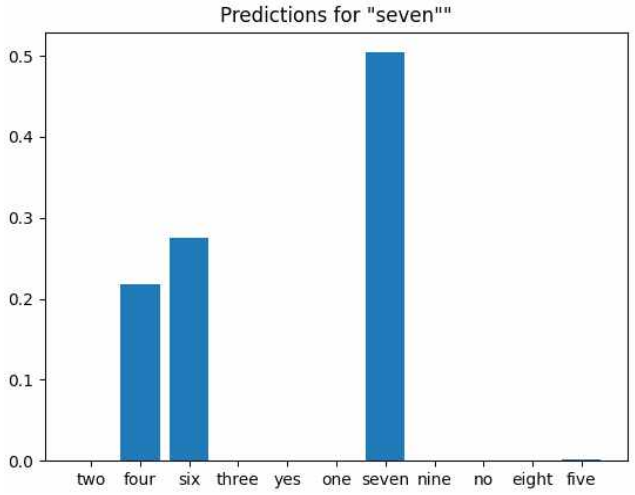
1번부터 9번까지 메뉴가 있고 음성에 따라 UI가 변경된다. 아래 progress bar는 시간을 의미하고 음성을 녹음하면 발화한 명령어를 인식하여 표시해준다. [그림 8]



[그림 8] 키오스크 메뉴 선택 UI 화면
음성으로 1~9(영어), yes, no를 말해 메뉴를 선택한다.

4. 연구 결과 분석 및 평가

[그림 9]는 소음 내성 음성인식 모델의 인식 성능을 보여준다. 단어 'seven'을 발화하였을 때, 모델은 해당 단어가 seven임을 잘 예측하였다.



[그림 9] "seven"을 말하는 음성 파일을 넣은 결과

[그림 10]와 [그림 11]을 보면 깨끗한 음성 파일로 만든 모델과 노이즈 음성 파일로 만든 모델을 비교해보았다. 각각 깨끗한 음성 테스트 세트와 노이즈 음성 테스트 세트로 정확도를 측정했을 때 기존에 있던 예제 코드보다 정확도가 상당히 높아졌지만, [그림 11]을 보면 깨끗한 음성 파일로 만든 모델에 노이즈 음성 테스트 세트를 대입해보니 정확도가 상당히 낮아졌다. 그럼에도 73%라는 생각보다 높은 정확도를 나타냈다. 그 이유를 생각해보면 데이터의 개수가 많고 소음의 크기도 무작위기 때문에 소음이 없는 데이터가 생겨 정확도가 높은 것 같다고 예상한다.

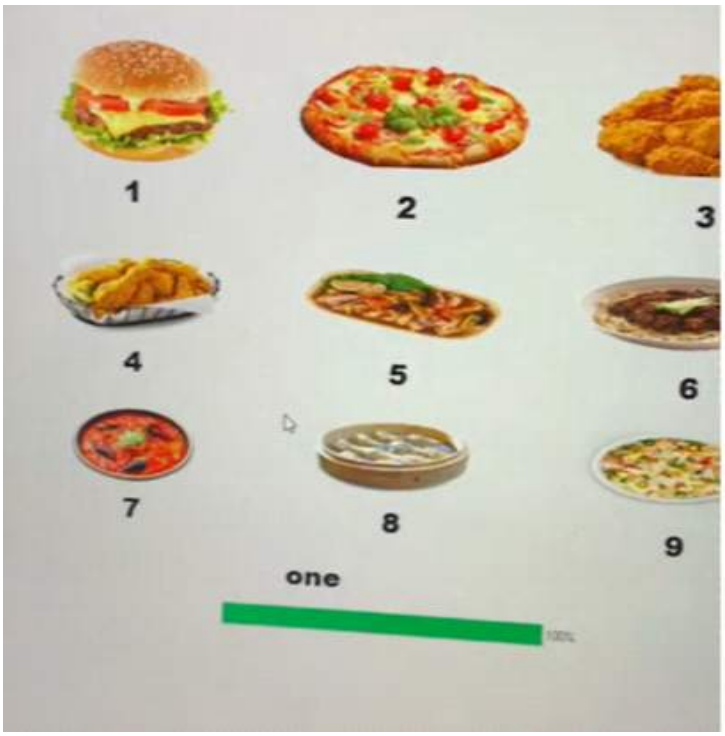
```
[67]  1 y_pred = np.argmax(Noise_model.predict(Clear_test_audio), axis = 1)
      2 y_true = Clear_test_labels
      3
      4 test_acc = sum(y_pred == y_true) / len(y_true)
      5 print(f'Noise Model + Clear Data set accuracy: {test_acc:.0%}')

      Noise Model + Clear Data set accuracy: 87%

[68]  1 y_pred = np.argmax(Noise_model.predict(Noise_test_audio), axis = 1)
      2 y_true = Noise_test_labels
      3
      4 test_acc = sum(y_pred == y_true) / len(y_true)
      5 print(f'Noise Model + Noise Data set accuracy: {test_acc:.0%}')

      Noise Model + Noise Data set accuracy: 86%
```

[그림 10] 소음이 포함된 음성으로 만든 Noise_모델
각각 소음 파일과 깨끗한 파일로 테스트했을 때 결과
[67]: Noise_모델과 깨끗한 음성, [68]: Noise_모델과 소음 음성

```
[71]  1 y_pred = np.argmax(Clear_model.predict(Clear_test_audio), axis = 1)
      2 y_true = Clear_test_labels
      3
      4 test_acc = sum(y_pred == y_true) / len(y_true)
      5 print(f'Clear Model + Clear Data set accuracy: {test_acc:.0%}')

      Clear Model + Clear Data set accuracy: 83%

[70]  1 y_pred = np.argmax(Clear_model.predict(Noise_test_audio), axis = 1)
      2 y_true = Noise_test_labels
      3
      4 test_acc = sum(y_pred == y_true) / len(y_true)
      5 print(f'Clear Model + Noise Data set accuracy: {test_acc:.0%}')

      Clear Model + Noise Data set accuracy: 73%
```

[그림 11] 깨끗한 음성으로 만든 Clear_모델
각각 소음 파일과 깨끗한 파일로 테스트했을 때 결과
[71]: Clear_모델과 깨끗한 음성, [70]: Clear_모델과 소음 음성

[그림 12]을 보면 시중에 있는 키오스크와 유사하게 만들었다. 마우스 클릭이 아닌 음성으로 메뉴를 선택할 수 있다. 메뉴를 말한 뒤 yes/no로 확인을 한 다음 계속 반복해서 메뉴를 선택할 수 있다.



[그림 12] 실제 UI 화면 캡처
음성으로 1~9(영어), yes, no를 말해 메뉴를 선택한다.
위 화면은 “one”을 말했을 때 상황이다.

5. 팀원별 역할 분담

팀원	역할	
공통	● 음성 데이터 수집 ● Git을 이용한 버전 관리	
안준수	● 음성 자료 합성 ● 딥러닝 모델 설계	- Python의 AudioSegment 모듈을 사용해 소음과 음성 합성 - 데이터 정규화 진행 - Google Colab과 Tensorflow를 사용해 딥러닝 모델 설계 및 제작
강동민	● STT API 적용 ● UI/UX 개발 ● 임베디드 환경 적용	- Kakao STT API를 이용해 음성인식에 취약한 소음 판단 - PyQt 및 TTS API를 이용해 키오스크와 유사한 UI/UX 제작 - 라즈베리 파이에 모델 적용

[표 2] 역할 분담표

6. 개발 일정

	6월				7월				8월				9월			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
기초 교육 및 사전 조사																
시나리오 구상																
음성 데이터 및 소음 수집																
음성 데이터와 소음 합성																
Google Colab 모델 학습																
데이터 학습																
모델 성능 테스트 및 수정																
임베디드 환경 적용																
GUI 프로그램 작성 및 테스트																
발표 및 시연 준비																

[표 3] 개발 일정표

7. 결론 및 향후 연구 방향

기존 산업에서 활용 및 이용되는 음성인식 모델의 경우, 소음 환경에서의 음성의 인식률은 그렇지 않은 환경에 비해 현저히 낮은 결과를 보였다. 이에 우리는 어떠한 소음 종류가 특히 음성인식의 정확도에 크게 영향을 미치는지 조사하고, 그러한 소음 환경을 극복하여 실용적으로 소음 내성 음성 인식모델을 활용할 다양한 시나리오를 구성해보았다. 또한, 소음 내성 음성인식 모델을 구축하기 위하여 이용되는 머신러닝 기술, 특히 그중에서도 CNN 학습 모델 등의 기술 사항 등을 공부하고 소음 내성 음성인식 모델 구축에 적용하였다. 또한, 이러한 모델을 임베디드에서 동작하도록 하여 상대적으로 다양한 환경에서의 모델 활용을 꾀했다.

이에, 우리는 COVID-19으로 인한 비대면, 비접촉 형태의 제품 및 서비스에 대한 높은 시장 수요와 그러한 시대정신을 고려하여 소음 내성 음성인식 기술 모델을 구축하고, 그러한 모델을 비접촉 음성인식 방식의 키오스크에 적용하는 활용안을 설계하였다. 이러한 설계로, 우리는 소음 환경 음성인식 모델의 시장에서의 무한한 활용 가능성을 엿볼 수 있었다. 우리가 구성한 음식점의 소음뿐만 아니라, 다양한 산업에서 발생하는 소음을 극복한 다양한 형태의 제품 및 서비스로 활용될 수 있을 것이다.

향후, 모델에 더욱 다양한 명령어를 지원하도록 데이터 세트의 크기를 늘린다거나, 여러 초매개변수(Hyper-parameters)를 수정 및 강화하여 모델의 정확도를 정교하게 향상하고자 할 때는 모델의 하드웨어적 요구사항이 상대적으로 상향될 것이다. 이러한 경우, 상대적으로 다소 하드웨어적 한계를 지닌 임베디드 기기에서의 동작은 어려워질 수 있다. 향후, 향상된 모델의 임베디드 기기에서의 구현을 위해서, CNN 학습 모델과 유사하지만 임베디드 기기에 좀 더 적합한 DS-CNN 학습 모델을 적용할 수 있다.또는, 현재 산업에서 전반적으로 통용되는 키오스크의 경우, 카드리더기, NFC, 영수증 출력을 위한 프린터기 등 각종 드라이버와의 원활한 통신을 위해 Windows OS 및 클라우드 환경 아래서 구현된 경우가 많다. 이러한 하드웨어적 성능 확보 및 OS 등의 최적화를 함께 고려하여 향후 모델 구축 및 연구가 이루어져야 한다.

8. 산업체 멘토링 내용

2021년 전기 산학협력 프로젝트 활동보고서

팀 명	포넥틱 코드			
팀 원	201524495_안준수, 201661701_강동민			
과 제 명	소음 환경에 강한 음성인식 모델 개발			
산 업 체 멘 토	기 업 명	㈜에프에스알엔티		
	성 명	박세영	직 위	대표이사
	연 락 처		E-MAIL	sypark@fsrnt.com

세부 활동 내용

1. 산학협력 프로젝트 멘토링 내용

첫 미팅에선 오프라인으로 만나 멘토링을 진행했습니다. 과제를 진행하기에 앞서 어느 정도 수준인지 이야기했고 어떤 방향으로 문제를 해결할지 같이 고민했습니다. 음성인식과 관련된 시중에 있는 제품, 회사 이야기를 들려주면서 다양한 선택지를 제시해 주셨습니다. 또한, 음성인식의 중요함과 발전 가능성을 들었습니다.

이후 첫 번째 보고서를 바탕으로 멘토 의견서를 받아 방향을 수정했습니다. 모든 단어보다는 특정 단어를 분석하고, 간단한 음향학 도서를 읽어보기도 했습니다.

두 번째 미팅에선 온라인으로 진행했습니다. 그동안 했던 자료들을 보고 계획대로 잘 진행되고 있는지, 진행 중 어려운 점 및 도움이 필요한 것들을 이야기했습니다.

이후 두 번째 보고서를 바탕으로 고민 중이었던 시나리오를 수정했습니다. 코로나 상황과 수요가 있을만한 키오스크 제작으로 바꾼 뒤 거기에 알맞은 명령어 샘플을 찾았습니다.

2. 산학협력 프로젝트 멘토링 후기

처음 만났을 때 음성인식, 텐서플로우, 인공지능에 대해 기초가 부족한 상태였습니다. 물론 공부는 조금 해갔지만, 정확한 목표가 없어 정확하게 어떤 방식으로 진행하겠다고 확답을 주지 못하고 우물쭈물했습니다. 그런데도 같이 커피를 마시면서 생각해보고, 전혀 엉뚱한 내용을 질문했을 때도 친절하게 대답해 주셨습니다.

멘토 의견서를 봤을 때, 우리가 제출한 보고서를 대충 훑어본 것이 아닌 꼼꼼히 본 것이 느껴졌습니다. 보고서 내용 중 궁금하거나 의문스러운 내용이 있으면 의견서에 피력하셨고, 저희가 과제를 진행하면서 놓치고 있었던 문제에 대해 자세하고 친절하게 설명해주셨습니다.

힘들었을 수도 있는 졸업과제를 좋은 멘토님을 만나 잘 끝낼 수 있었던 것 같아 다행이라고 생각합니다. 졸업과제와 관련된 내용뿐만 아니라 대화하면서 많은 것을 배울 수 있는 시간이었습니다.

[그림 13] 산업체 멘토링 내용

9. 참고 문헌

<관련 논문>

[1] Within Google, Andrew Howard [6] has introduced efficient mobile models called MobileNets using depthwise separable convolutions.

[2] Xception: Deep Learning with Depthwise Separable Convolutions

[그림 1] 출처: https://docs.ecognition.com/v9.5.0/eCognition_documentation/Reference%20Book/23%20Convolutional%20Neural%20Network%20Algorithms/Convolutional%20Neural%20Network%20Algorithms.htm

[그림 2] 출처: Li Yang, Binarized Depthwise Separable Neural Network for Object Tracking in FPGA, 2019. 05.

[그림 3] 출처: https://brightwon.tistory.com/11

<Python Library>

[1] librosa: https://librosa.org/

[2] AudioSegment: https://audiosegment.readthedocs.io/en/latest/audiosegment.html

<Commands Set>

[1] Google Speech Command Set v2 : http://download.tensorflow.org/data/speech_commands_test_set_v0.02.tar.gz

<Data Samples>

[1] PNL 100 Nonspeech Sounds: http://web.cse.ohio-state.edu/pnl/corpus/HuNonspeech/HuCorpus.html

[2] AudioSet: https://research.google.com/audioset/

[3] FSD50K: https://zenodo.org/record/4060432#.YJTrJrUzaUl

[4] Restaurants Sound: https://www.fesliyanstudios.com/royalty-free-sound-effects-download/restaurants-179

<표 1 에서 사용한 Cloud STT API>

[1] KAKAO NEWTON STT: https://speech-api.kakao.com/

<참고 웹 사이트>

[1] https://www.tensorflow.org/tutorials/audio/simple_audio

[2] https://yunmorning.tistory.com/58

[3] https://yeomko.tistory.com/45

[4] https://m.blog.naver.com/sooftware/221661644808

[5] https://hichoe95.tistory.com/48

[6] https://halfundecided.medium.com/%EB%94%A5%EB%9F%AC%EB%8B%9D-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-cnn-convolutional-neural-networks-%EC%89%BD%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-836869f88375