파이널 프로젝트 2조

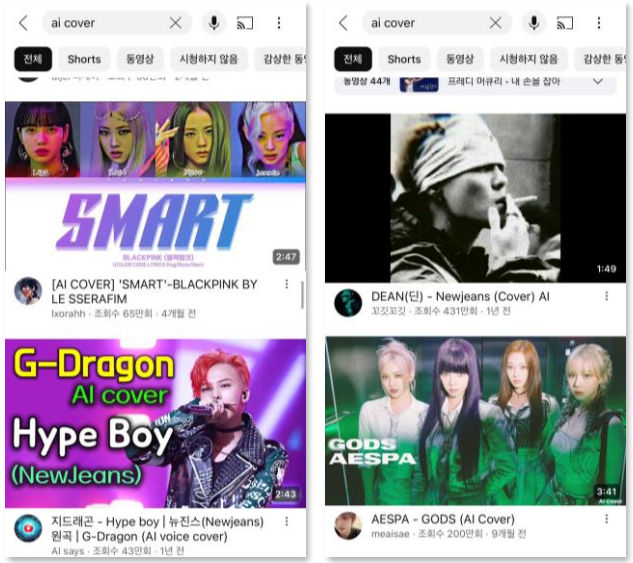
AI generated music 판별 모델

조원: 김민정, 배진우, 유현종, 정태웅, 최지혜, 한성필

Skills : Python/Umap/MFCC/Tensorflow

주제 선정 배경

- AI 음원 생산 사례 증가

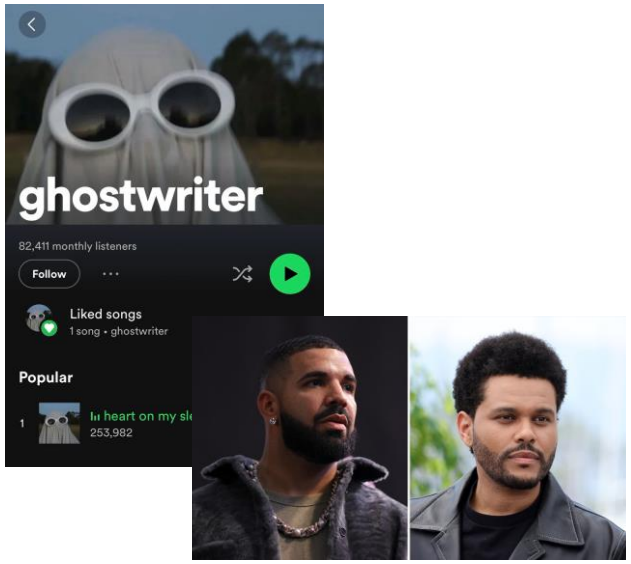


유튜브 'AI cover' 검색 결과
다량의 조회 수 발생

- 관련 규제 미비

학습 시 음원 / 목소리 무단 사용
창작자에 대한 공정한 보상 x
창작물 수집 윤리 무시

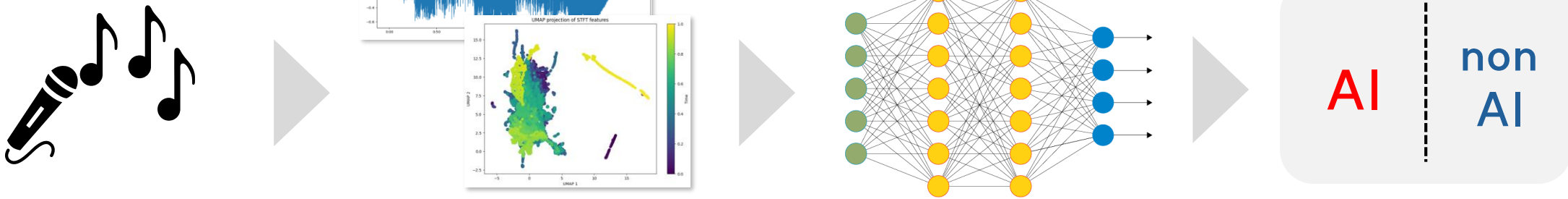
- 저작권 문제



현행법상 목소리에 대한
저작권 침해 주장 성립 어려움

AI 생성 음원인지 구분하는 것을 목적으로 하여
'AI 음원 판별 모델'을 주제로 선정

프로젝트 개요



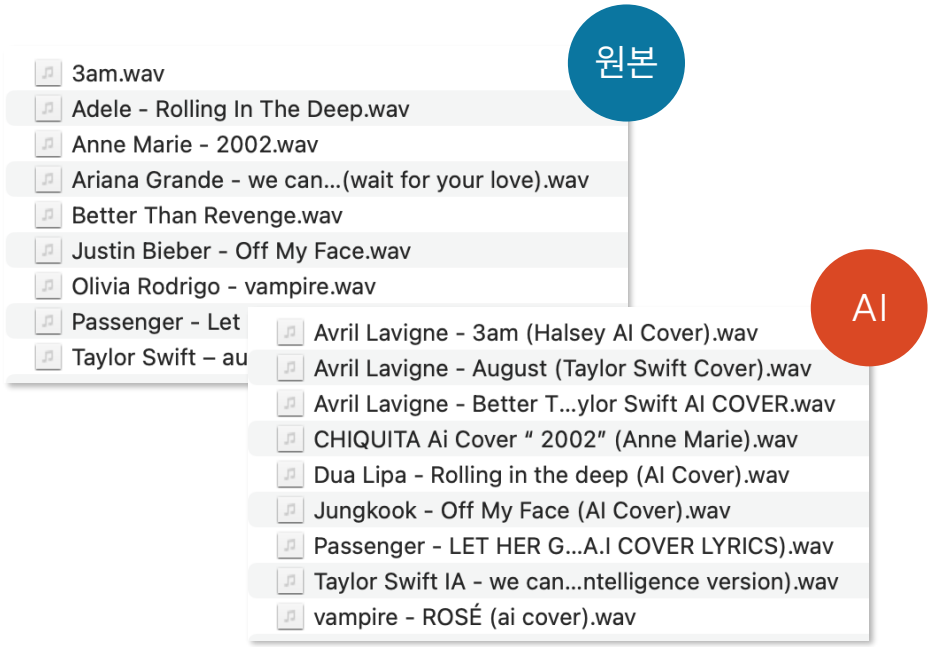
음원 데이터 수집 | 파형 및 umap 시각화 | 모델링 | 분류 결과 출력

딥러닝을 통해 AI와 원본 음원 파일을 학습
이를 통해 wav 파일을 분류하는 모델 생성

데이터 수집 및 시각화

#데이터 수집 기준

- 사람의 귀로 들었을 때(즉, 파형을 비교했을 때) AI 구분이 어려운 음원들 위주 수집
  *높은 난이도로 학습 시 모델 성능 향상
- 기계음이 거의 포함되지 않은 솔로 가수 팝송으로 범주 한정



mp4

- 압축으로 인한 데이터 손실 발생
- 신호 처리 및 특성 추출에 부적합

wav

- 원시 오디오 데이터 제공
- 샘플링 레이트, 뎁스 조정 용이

수집한 데이터
wav 형식으로 변환

&

파형 및 umap 시각화

데이터 수집 및 시각화

Which one is AI?

Jungkook - Standing Next To You (15sec)

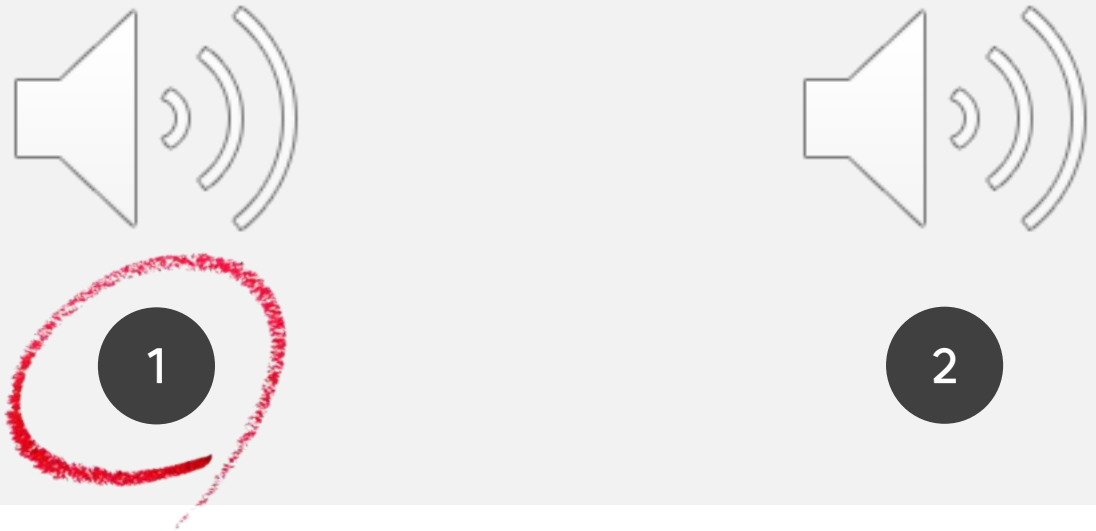
1

2

데이터 수집 및 시각화

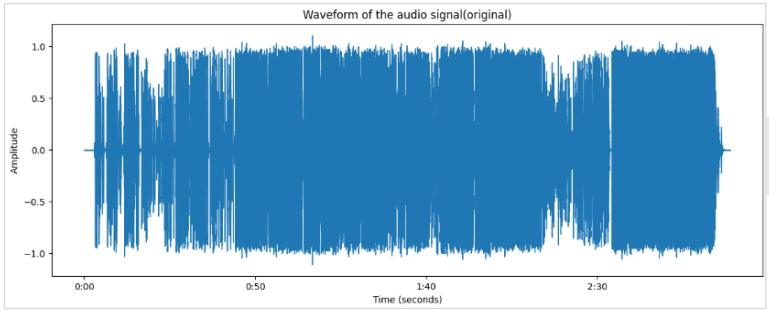
Which one is AI?

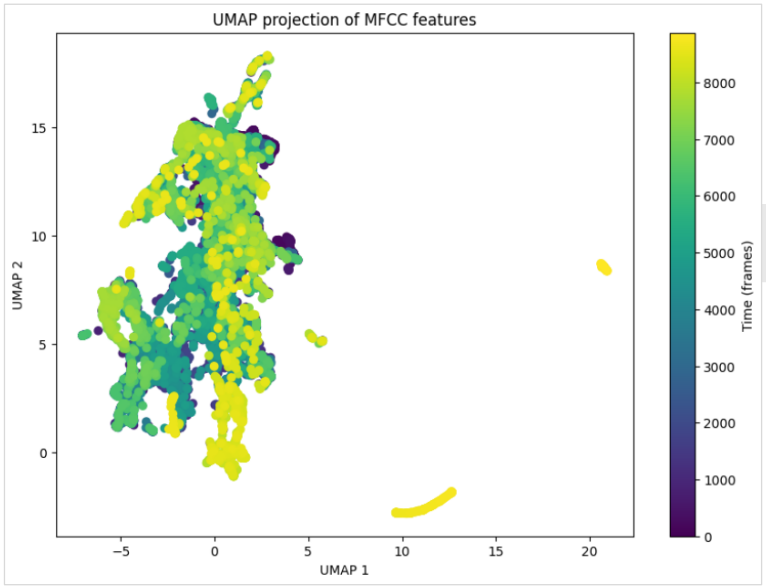
Jungkook - Standing Next To You (15sec)



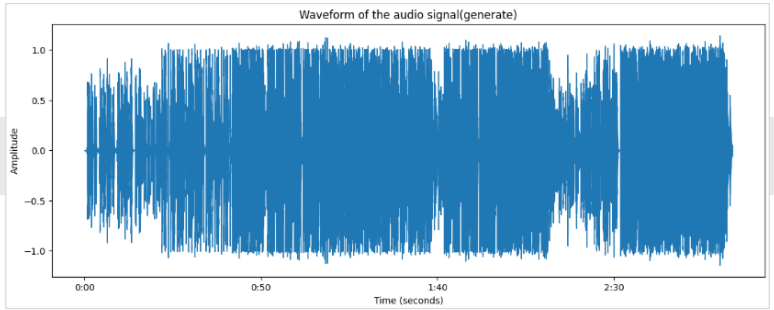
데이터 수집 및 시각화

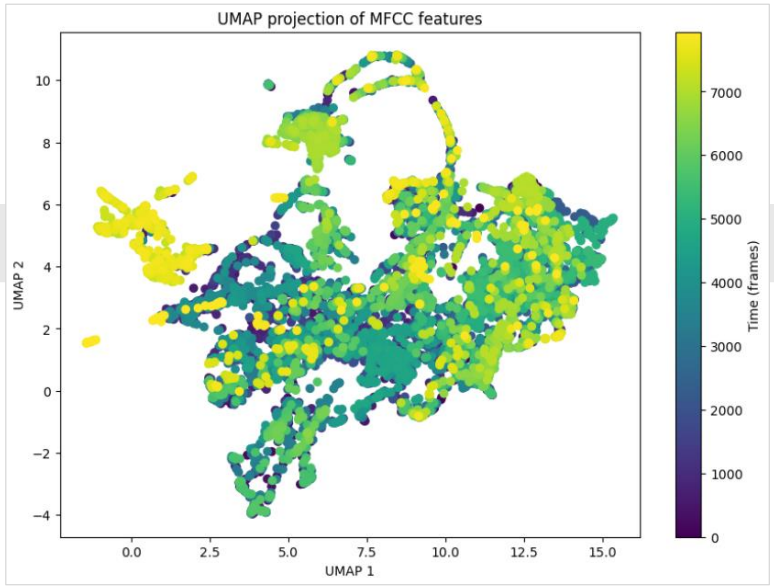
원본





AI





차이 확인 어려움

umap 시각화 시
명확한 차이 확인

1) 음성데이터 세션 나누기

```python
#음원 구간별 추출
from pydub import AudioSegment
import os

for a in X_path:
  # wav 파일 로드
  audio = AudioSegment.from_wav(a)

  # 분할 간격 설정 (밀리초 단위)
  interval = 10 * 1000  # 10초 = 10,000 밀리초
  step = 5 * 1000  # 5초 = 5,000 밀리초

  # 전체 길이
  length = len(audio) #밀리초로 환산됨


  # 저장할 디렉토리 설정
  output_dir = r'/gdrive/MyDrive/빅데이터 45기_파이널 프로젝트/음원데이터/save'

  # 분할 및 저장
  for i in range(0, length, step):
      start_time = i  # 시작 시간 (밀리초 단위)
      end_time = i + interval  # 종료 시간 (밀리초 단위)
      split_audio = audio[start_time:end_time]
      start_sec = start_time // 1000  # 시작 시간 (초 단위)
      end_sec = end_time // 1000  # 종료 시간 (초 단위)
      filename = os.path.join(output_dir, f"output_{start_sec}-{end_sec}_{os.path.basename(a)}.wav")
      split_audio.export(filename, format="wav")
      print(f"Saved {filename}")

  print("분할 완료")
```

- interval = 10초 단위 분할
- step = 5초 단위 중첩

분할된 오디오 데이터 별도 저장

2) 오디오 특징 추출

```python
def extract_features(audio_samples, sample_rate):
    extracted_features = np.empty((0, 101))
    if not isinstance(audio_samples, list):
        audio_samples = [audio_samples]

    for sample in audio_samples:
        zero_cross_feat = librosa.feature.zero_crossing_rate(sample).mean()
        mfccs = librosa.feature.mfcc(y=sample, sr=sample_rate, n_mfcc=100)
        mfccsscaled = np.mean(mfccs.T,axis=0)
        mfccsscaled = np.append(mfccsscaled, zero_cross_feat)
        mfccsscaled = mfccsscaled.reshape(1, 101)
        extracted_features = np.vstack((extracted_features, mfccsscaled))
    return extracted_features
```

```python
# 오디오 특징 추출

X_train_features = extract_features(X_train2, sample_rate)
X_test_features  = extract_features(X_test2, sample_rate)
```

| MFCC 방식으로 오디오 특징 추출

3) 인공신경망 구축

```python
# 데이터셋 나누기
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_path_new, y, test_size=0.25, random_state=42)
print(len(X_train)) # 600
print(len(X_test)) # 200
```

```python
# 음성파일 읽기 & 디지털 변환 함수
def librosa_read_wav_files(wav_files):
    if not isinstance(wav_files, list):
        wav_files = [wav_files]
    return [librosa.load(f, sr = 16000)[0] for f in wav_files] # 음성파일 읽기
```

```python
# train/test 음성파일 읽기
X_train2 = librosa_read_wav_files(X_train)
X_test2  = librosa_read_wav_files(X_test)
print(len(X_train2)) # 600
print(len(X_test2)) # 200
```

| Train, Test split
| 저장한 wav 파일 read

Dataset	
Train 75%	Test 25%

3) 인공신경망 구축

```python
# 신경망 구축
model = Sequential()
model.add(Conv1D(64, kernel_size=3, activation='relu', input_shape=(101,1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(128, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(256, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

*model.summary( )

Layer (type)	Output Shape	Param #
conv1d_3 (Conv1D)	(None, 99, 64)	256
max_pooling1d_3 (MaxPooling1D)	(None, 49, 64)	0
conv1d_4 (Conv1D)	(None, 47, 128)	24704
max_pooling1d_4 (MaxPooling1D)	(None, 23, 128)	0
conv1d_5 (Conv1D)	(None, 21, 256)	98560
max_pooling1d_5 (MaxPooling1D)	(None, 10, 256)	0
flatten_1 (Flatten)	(None, 2560)	0
dense_2 (Dense)	(None, 64)	163904
dropout_1 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Total params: 287489 (1.10 MB)
Trainable params: 287489 (1.10 MB)
Non-trainable params: 0 (0.00 Byte)

4) 학습

```
# 학습환경
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

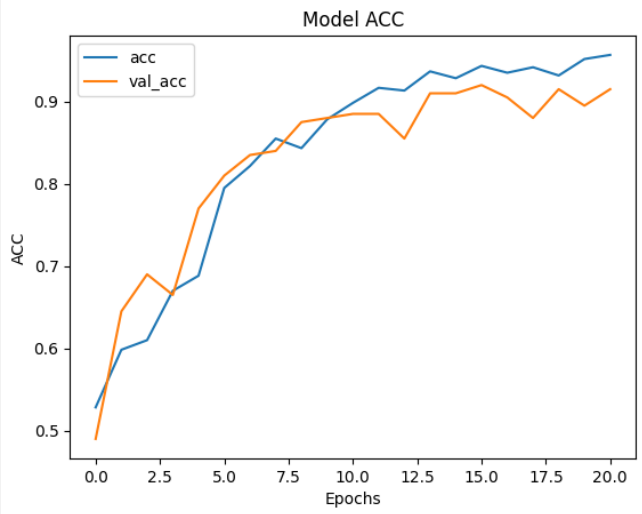
```
# 여기서는 val_accuracy 모니터링해서 성능이 좋아지지 않으면 조기 종료 하게 함.
early_stop = EarlyStopping(monitor='val_accuracy', verbose=1, patience=5)
```

```
check_point = ModelCheckpoint('best_model.h5', verbose=1,
                              monitor='val_loss', save_best_only=True)
```

```
history = model.fit(X_train_features, y_train, epochs=50, batch_size=32, validation_data=(X_test_features,y_test), verbose=1,
          callbacks=[early_stop, check_point])
```

- optimizer = 'adam'
- loss='binary_crossentropy'
- metrics=['accuracy']
- epochs=50
- batch_size=32

**EarlyStopping(조기 종료) 적용



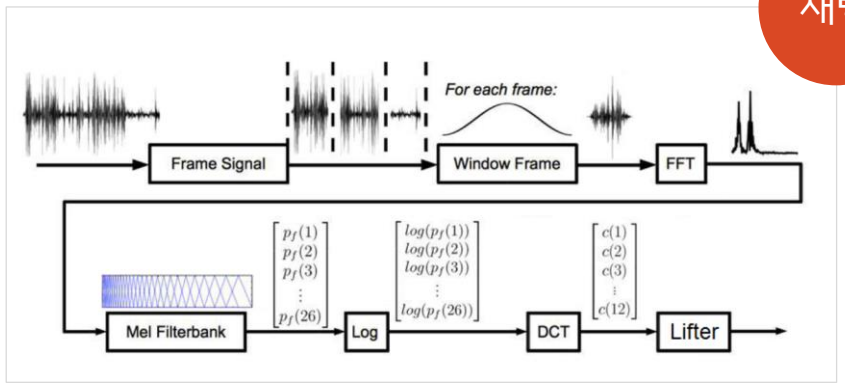
EarlyStopping 적용해 epoch 21번째에서 종료
모델 accuracy 91.5% 확보

방법론 적용 – 1) 오디오 특징 추출

#MFCC

(Mel-Frequency Cepstral Coefficients)

채택

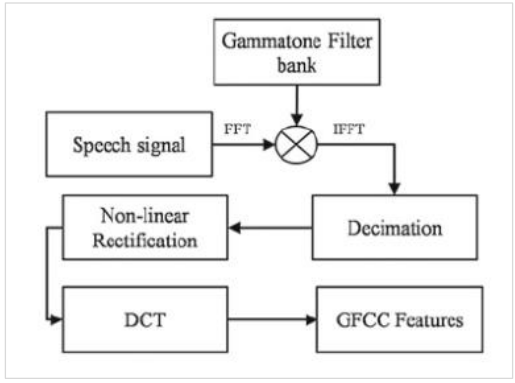


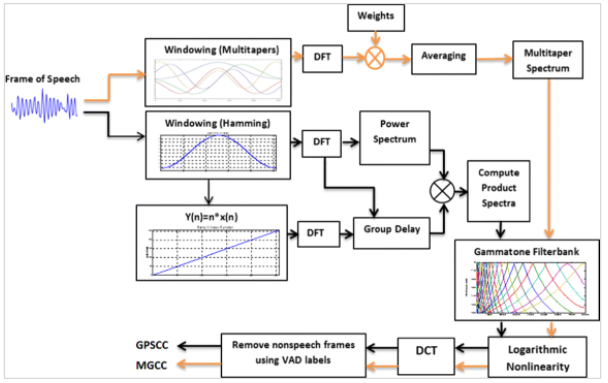
주파수 영역에서 오디오 신호를 분석해 멜 스케일로 변환
사람의 청각에 더 적합한 주파수 대역 강조

*특징
- 사람이 소리를 듣는 방식과 유사하게 분석
- 음성 인식, 음악 장르 분류 등에 주로 사용

#GFCC

(Gammatone Frequency Cepstral Coefficients)



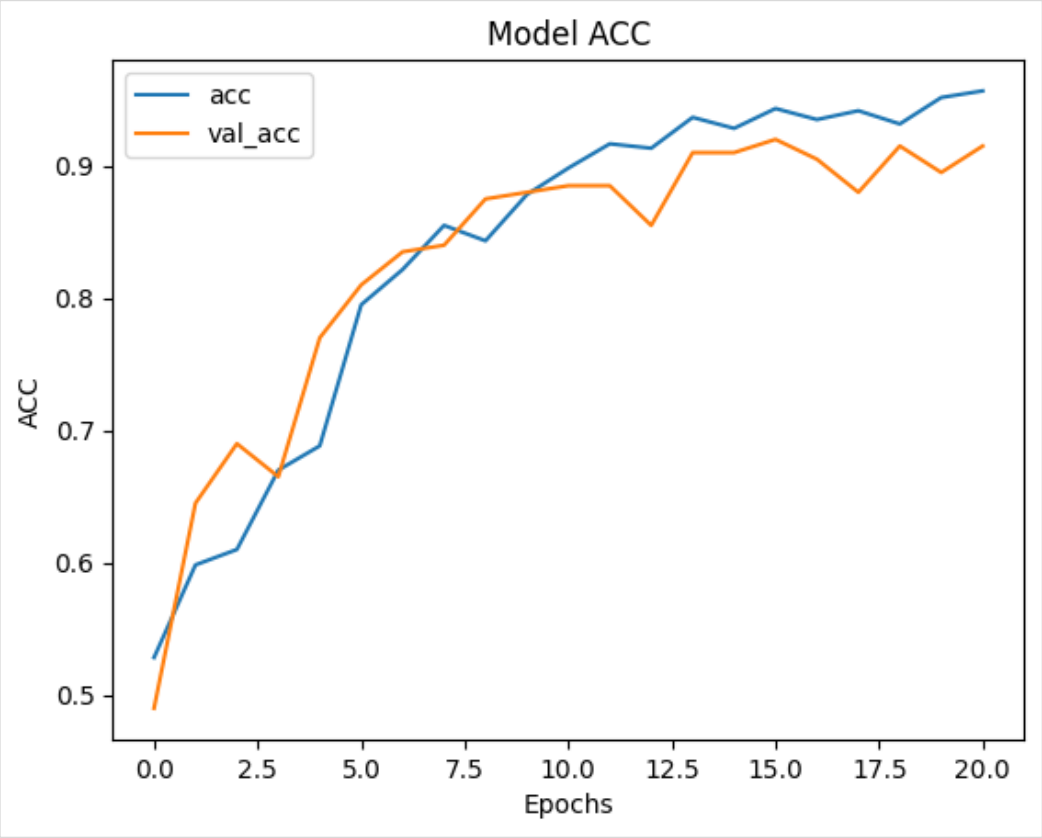


청각 시스템의 비선형 특성을 모방하는 감마톤 필터 사용
귀의 기계적 반응을 모델링하는 데 효과적

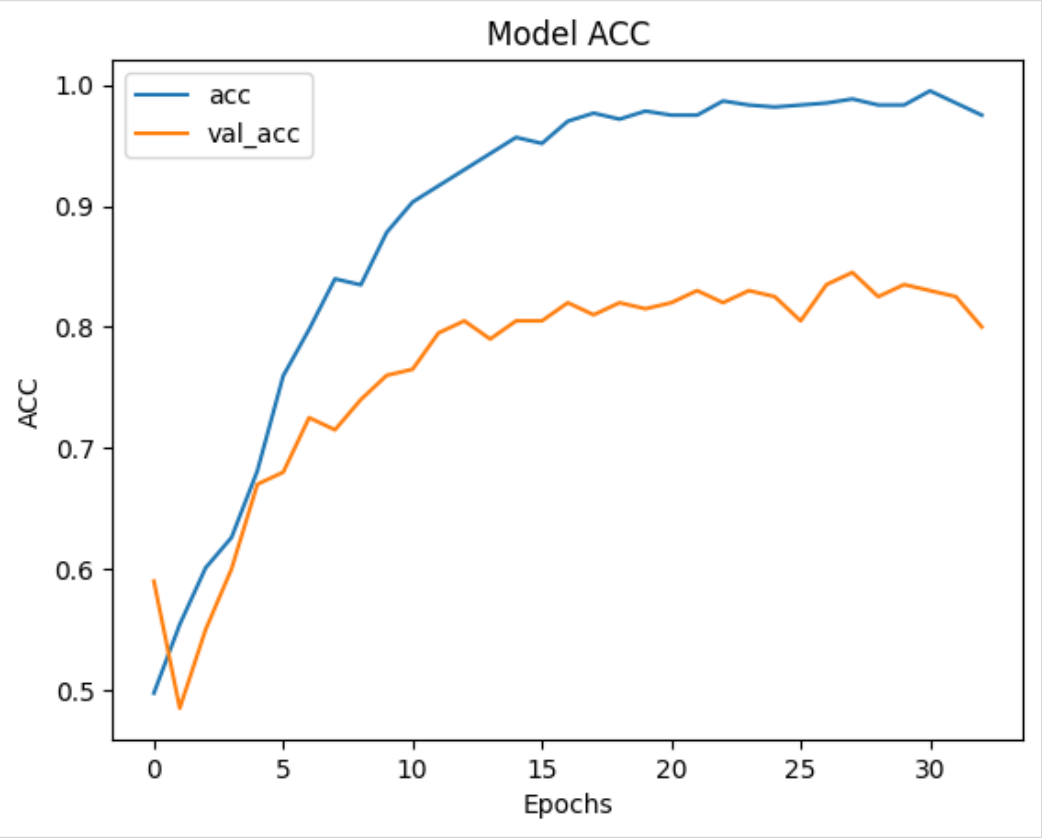
*특징
- 소리의 세밀한 특징을 분석해 잡음 구분에 용이
- 노이즈가 많은 환경에서 음성 인식을 위해 사용

방법론 적용 – 1) 오디오 특징 추출

#MFCC



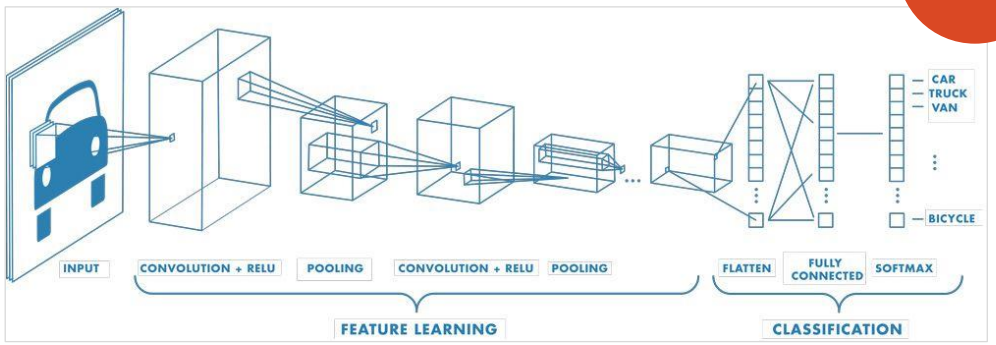
#GFCC



방법론 적용 – 2) 딥러닝 알고리즘

#CNN

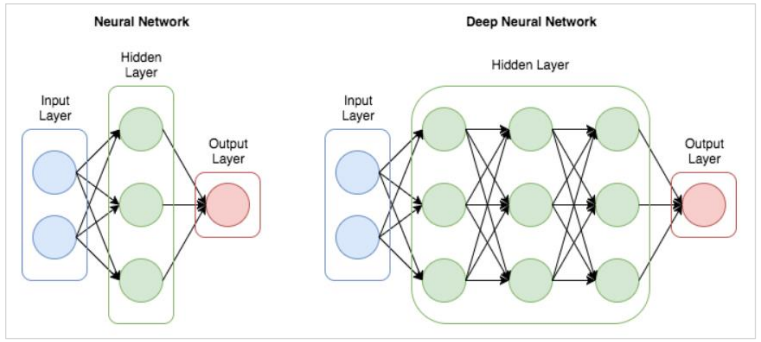
채택



주로 이미지 및 비디오 데이터의 처리에 사용
이미지 인식, 객체 탐지, 영상 분석 등에서 우수한 성능

*특징
- 합성곱(Convolution) 과 풀링(Pooling) 연산
- 최종적으로 완전 연결층을 통해 분류 작업 수행

#DNN

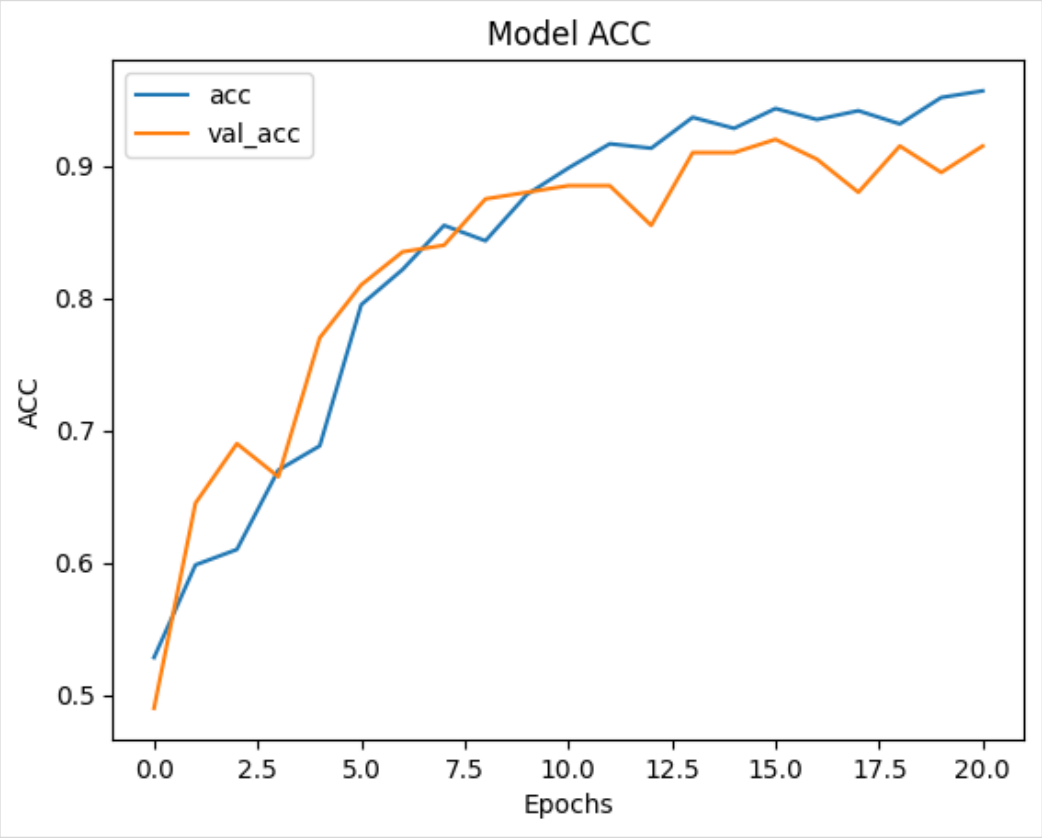


여러 개의 은닉층을 가지고 있는 신경망
주로 비선형적인 문제를 해결하는 데 사용

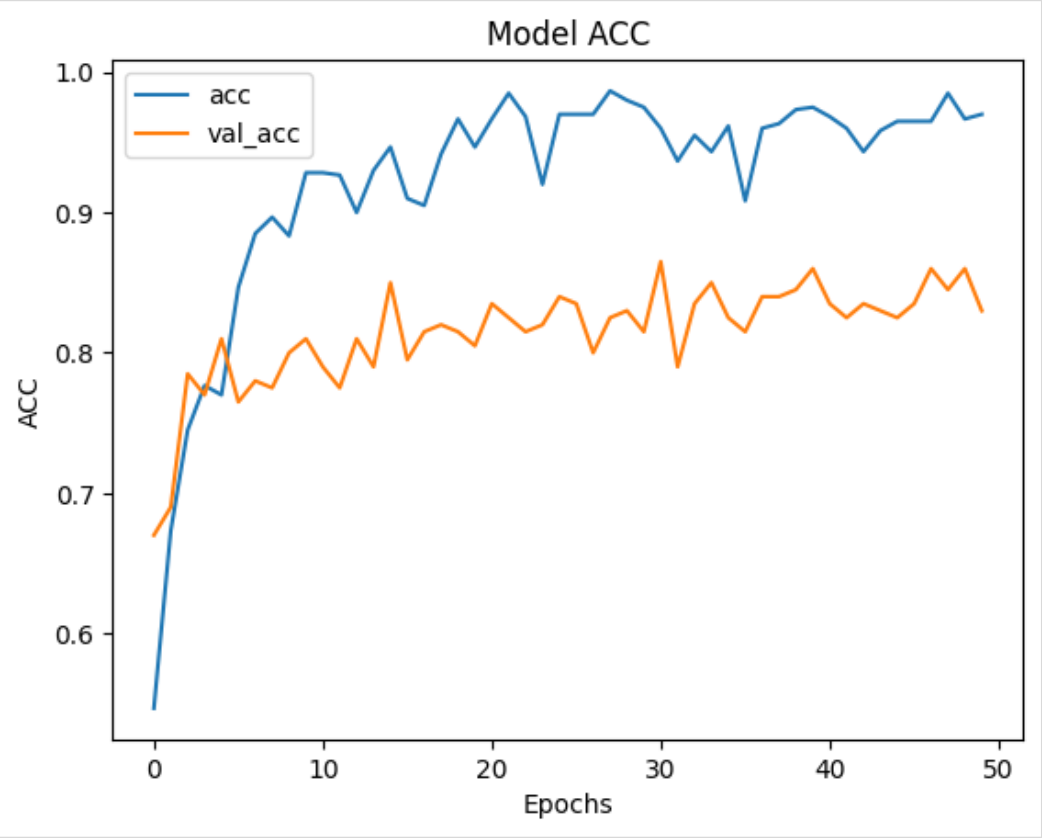
*특징
- 입력층, 은닉층, 출력층으로 구성돼 복잡한 패턴 학습
- 활성함수 사용, 계층적 특징 학습

방법론 적용 – 2) 딥러닝 알고리즘

#CNN

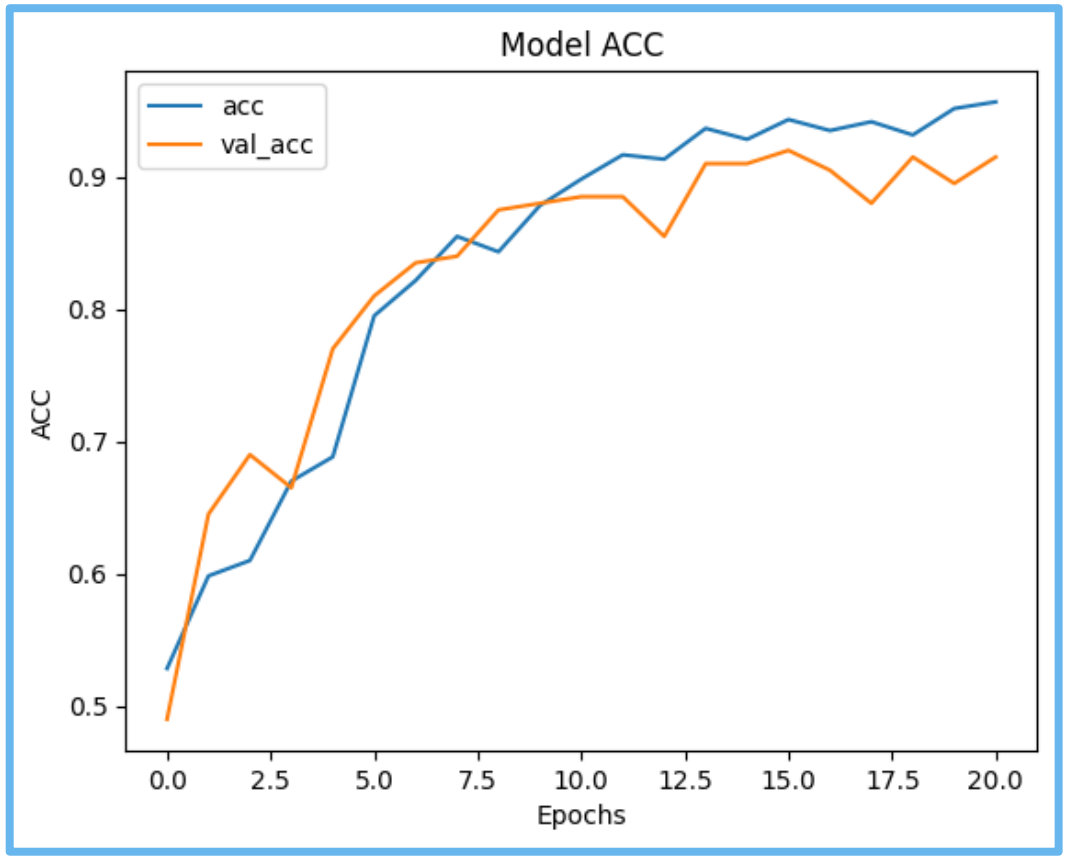


#DNN

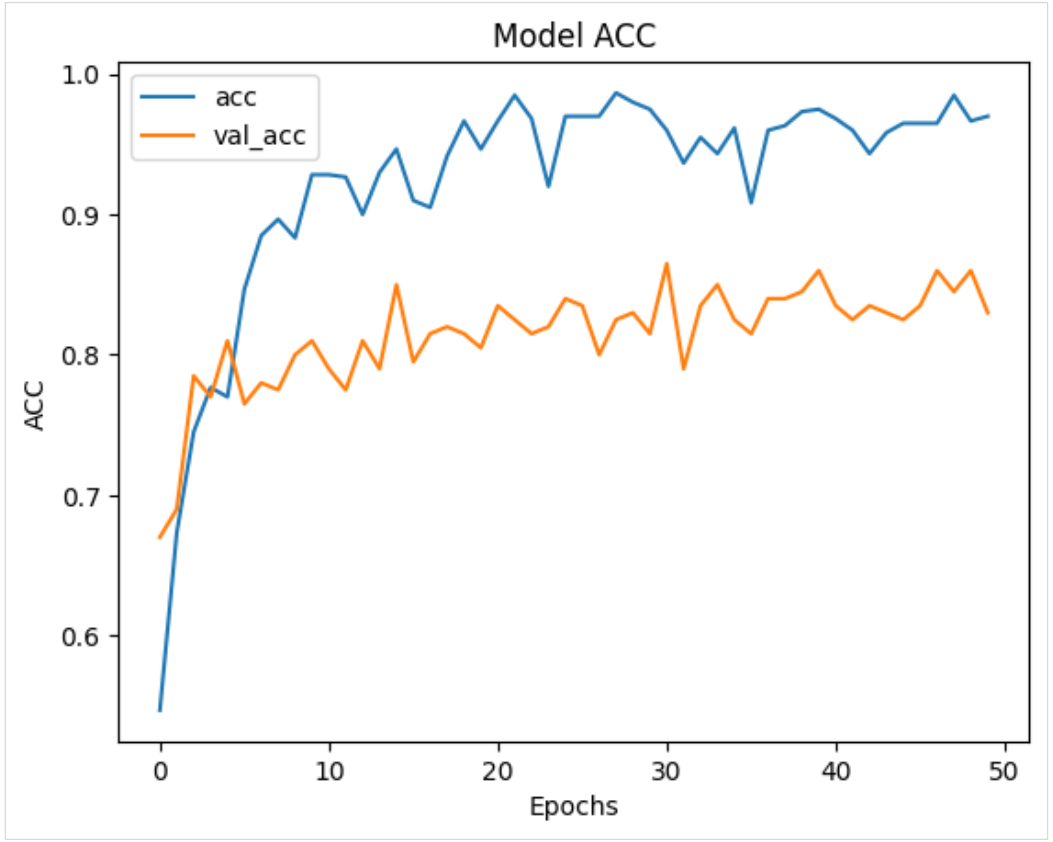


방법론 적용 – 2) 딥러닝 알고리즘

#CNN



#DNN



한계점

- 음성 관련 데이터 확보의 어려움
- 방법론적, 시간적 한계
- 사용 가능한 음성 데이터가 한정적

  ex) 이미 튜닝된 음원, 혼성/그룹 음원

향후 발전 가능성 및 시사점

- 장르나 카테고리 추가 학습시켜 사용 범위 확대
- AI음원 구분 필요성 확인

  ex) AI 워터마크 의무화 추진, 인스타그램 AI 제작 라벨

  ➔ AI 음원에 대한 수익 분배나 유료 api 연동 등 사업화 가능성

  ➔ 콘텐츠 게시 시 AI 여부 자동 판별 시스템화

감사합니다.

파이널 프로젝트 2조