

주 제 분 석

너  의
목소리가
보  여

목차



01 주제 소개

02 개발 환경 세팅

03 음성 데이터 소개

04 분석 과정

05 교수님 피드백

01

주제 소개



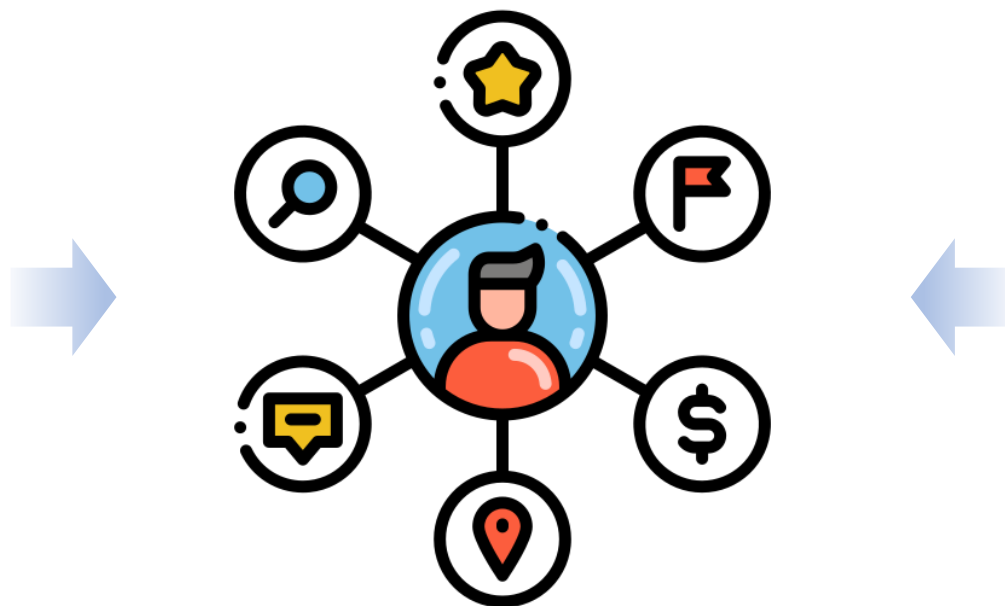
01 주제 소개

딥러닝을 활용한 화자 프로파일링

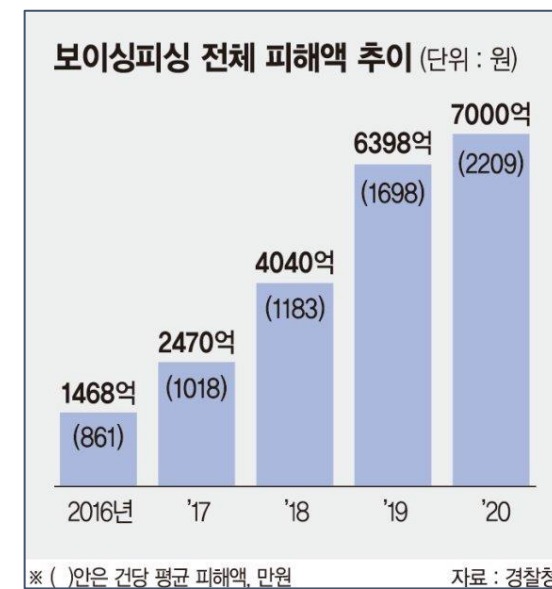
주제 선정 배경



미제 사건




보이스 프로파일링



보이스 피싱

01 주제 소개

딥러닝을 활용한 화자 프로파일링

 주제 선정 배경

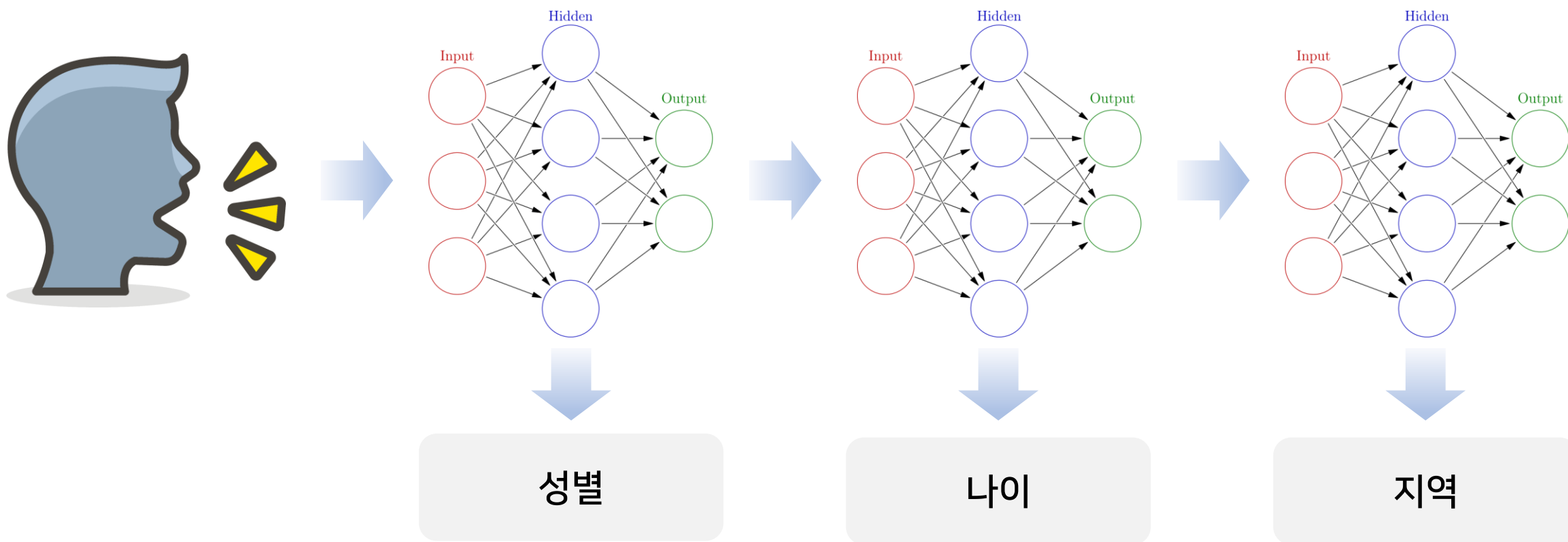


사람의 음성을 바탕으로 성별, 나이, 출신 지역, 최종 학력 등을 유추할 수 있는 모델 구현

01 주제 소개

딥러닝을 활용한 화자 프로파일링

모델 개요



01 주제 소개

딥러닝을 활용한 화자 프로파일링

기대 효과

음성 인식을 통한 범죄사건 해결



발화자 정보 추정을 통한 개인화 마케팅



02

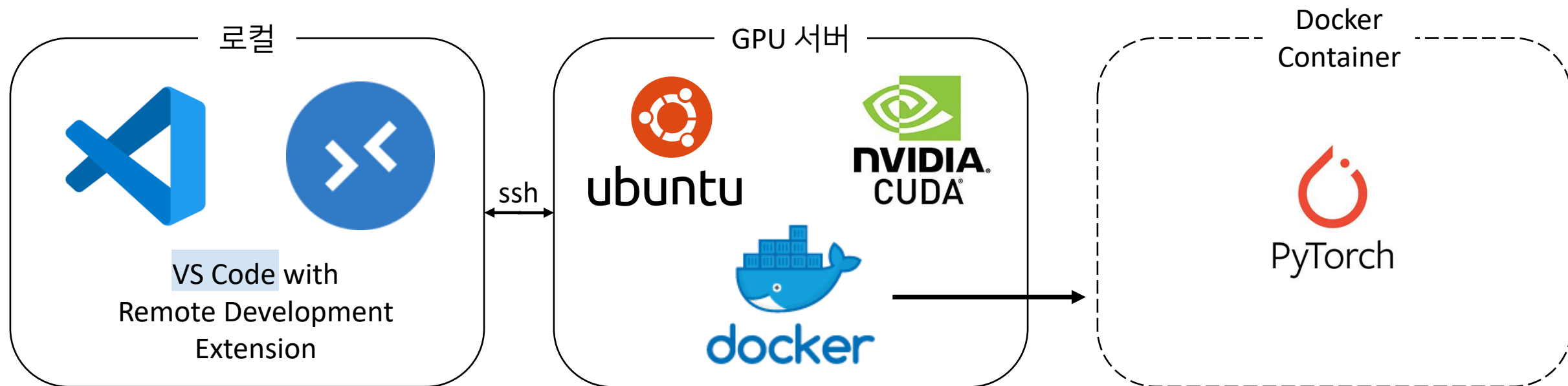
개발 환경



02 개발 환경

개발 환경

초기 개발 환경



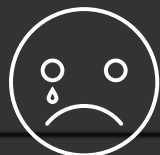
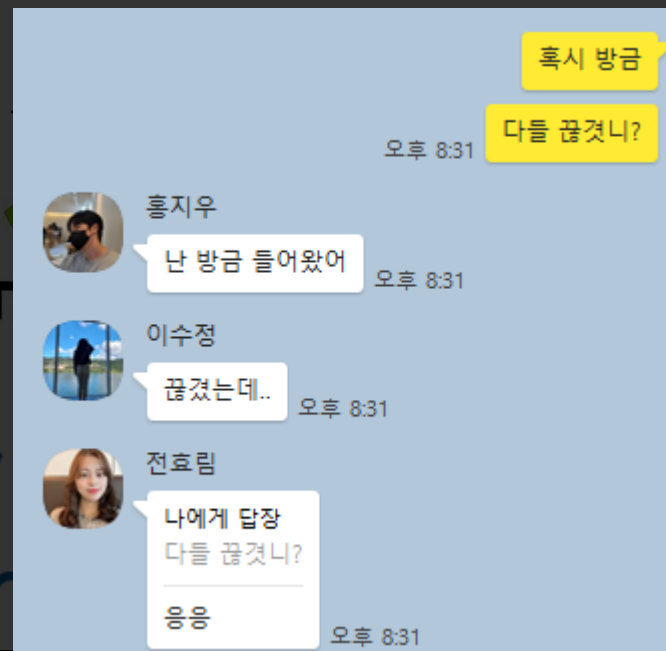
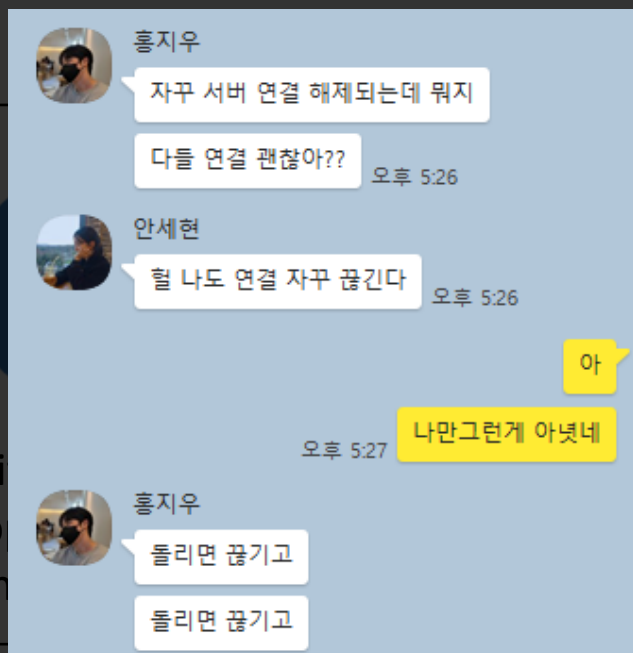
초기에는 VS Code를 이용하여 서버 접속

02 개발 환경



개발 환경

초기 개발 환경

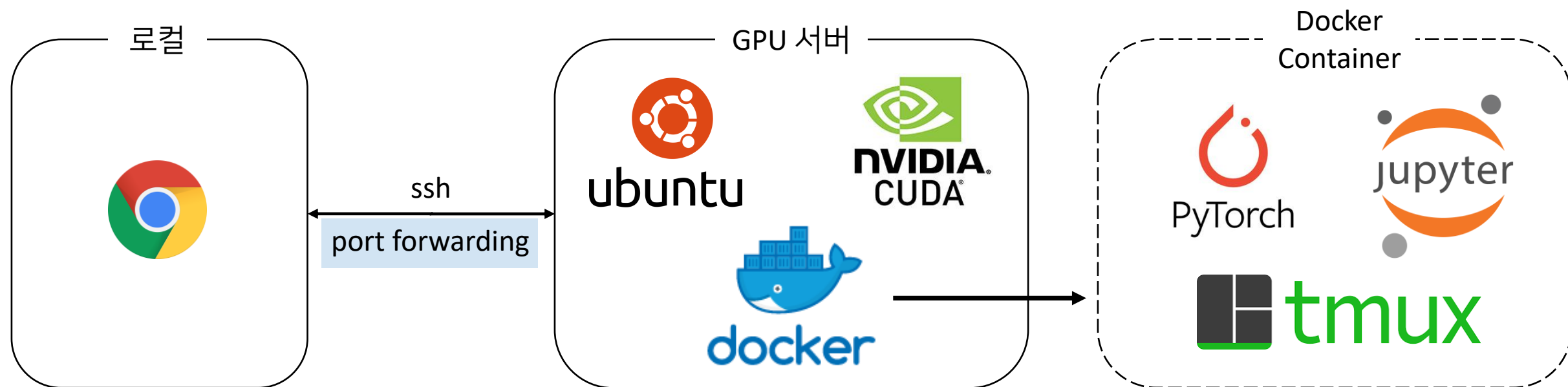


초기에는 VS Code를 이용하여 서버에 접속
특정 상황에서 서버의 연결이 끊기는 현상 자주 발생

02 개발 환경

🎧 개발 환경

▮ 현재 개발 환경



서버에서 주피터 노트북을 실행해 서버를 연 후
포트 포워딩을 통해 로컬에서 직접 접속

02 개발 환경

구성 요소

우분투 (Ubuntu)



ubuntu

서버에서 사용하는 OS

오픈소스 OS인 Linux의 일종으로
가장 인기있는 Linux 배포판

Linux는 다중 사용자 시스템으로
한 컴퓨터를 여러 사용자가 동시에 사용할 수 있어
서버 측 OS로 자주 사용

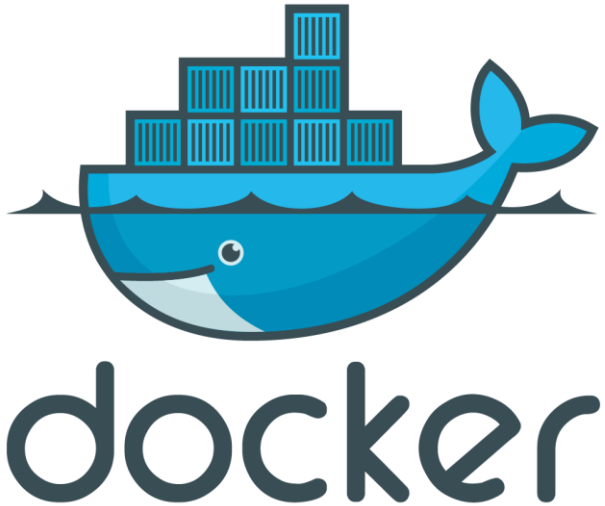
기본적으로 CLI (Command-Line Interface) 환경

02 개발 환경



구성 요소

도커 (Docker)



컨테이너 기반의 가상화 플랫폼

컨테이너란 격리된 공간에서 프로세스가 동작하는 기술
(가상 환경의 일종)

OS나 다른 컨테이너와 독립적인 컨테이너를
생성, 사용, 공유 가능

컨테이너는 이미지의 형태로 저장, 공유되며
이미지를 이용해 컨테이너 생성 가능

02 개발 환경



구성 요소



tmux



터미널에 세션(session)과 윈도우(window) 기능을 더해
생산성을 높여주는 도구

세션을 종료하지 않는다면 백그라운드에서 지속적으로 작업 가능

현재 tmux를 통해 주피터 서버를 열어 둔 상태

02 개발 환경

구성 요소

PyTorch



PyTorch는 Tensorflow와 함께 가장 널리 쓰이는 딥러닝 프레임워크

CUDA는 딥러닝 프레임워크가 연산을 할 때
그래픽 카드 사용을 위해 필요로 하는 프로그램

도커를 이용해 PyTorch image를 container로 만들어 작업 중

02 개발 환경

 구성 요소

 SSH



Secure Shell의 약자

원격 호스트에 접속하기 위해 사용되는 보안 프로토콜

SSH로 서버에 접속해 컨테이너 생성 및 주피터 서버 실행



포트 포워딩

로컬에서 서버의 주피터 노트북 사용 가능

03

음성 데이터



03 음성 데이터

🎧 음성 분석

음성 분석 종류

음성인식
(Speech Recognition)



사람이 말하는 음성 언어를 컴퓨터가 해석해 문자 데이터로 전환하는 처리
STT(Speech-to-Text)

오디오 분류
(Audio Classification)



도시에서 발생하는 소음 종류

오디오 캡셔닝
(Audio Captioning)



자막 자동 생성

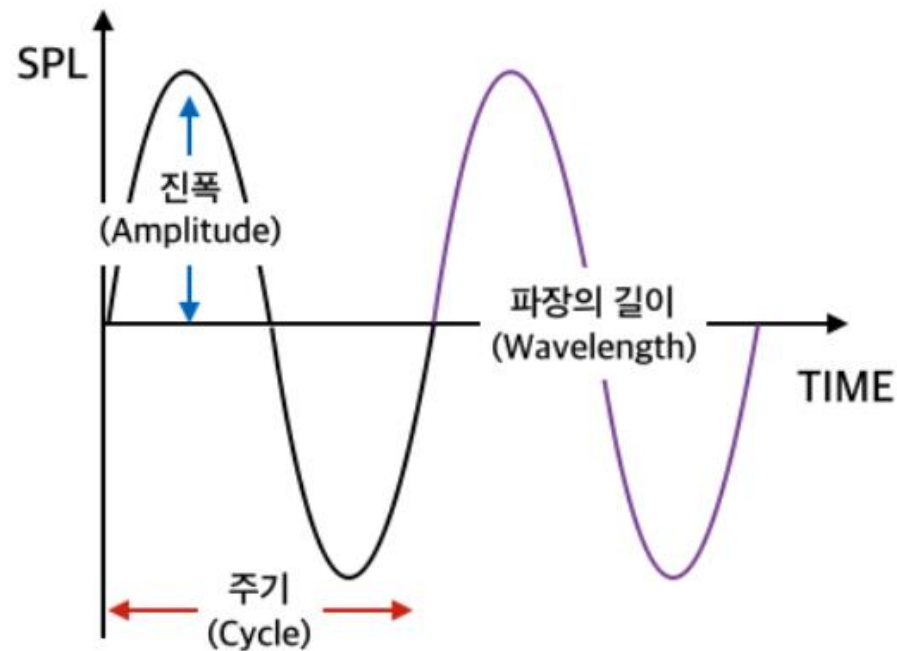
03 음성 데이터



소리의 정의

소리

- 공기 속을 전해오는 파동
- 소리의 3요소
 1. 세기
 - 파동의 진폭, 소리의 크기
 2. 높낮이
 - 파동의 주파수, 소리의 높낮이
 3. 맵시
 - 파동의 파형, 소리의 색상(피아노, 바이올린)



03 음성 데이터



소리의 형태

소리

- 공기 속을 전해오는 파동
- 소리의 3요소

1. 세기

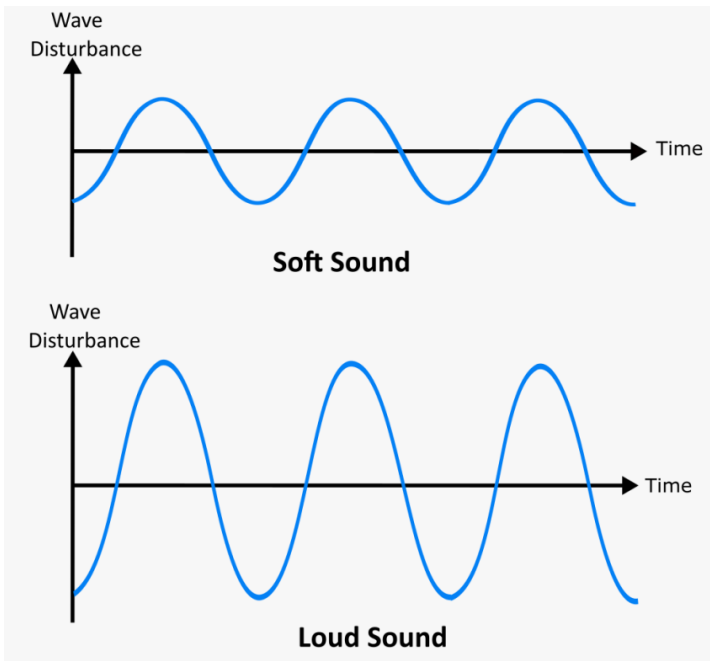
- 파동의 진폭, 소리의 크기

2. 높낮이

- 파동의 주파수, 소리의 높낮이

3. 맵시

- 파동의 파형, 소리의 색상(피아노, 바이올린)



진폭: 파형의 기준선에서 최고점까지의 거리
소리의 세기는 진폭에 따라 달라지며,
진폭이 크면 큰 소리, 작으면 작은 소리

03 음성 데이터

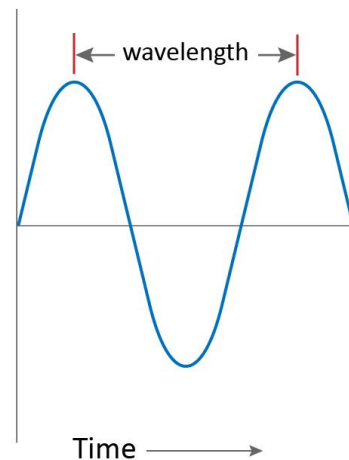


소리의 형태

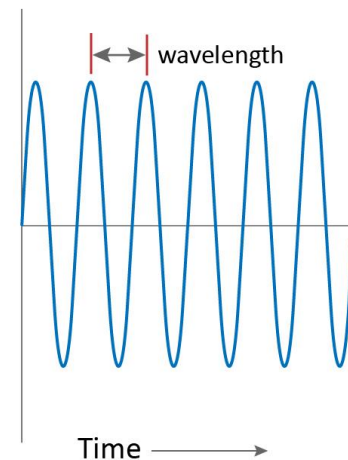
소리

- 공기 속을 전해오는 파동
- 소리의 3요소
 1. 세기
 - 파동의 진폭, 소리의 크기
 2. 높낮이
 - 파동의 주파수, 소리의 높낮이
 3. 맵시
 - 파동의 파형, 소리의 색상(피아노, 바이올린)

Low pitch



High pitch



주파수: 전파나 음파가 1초 동안에 진동하는 횟수
주파수가 높으면 고음, 낮으면 저음

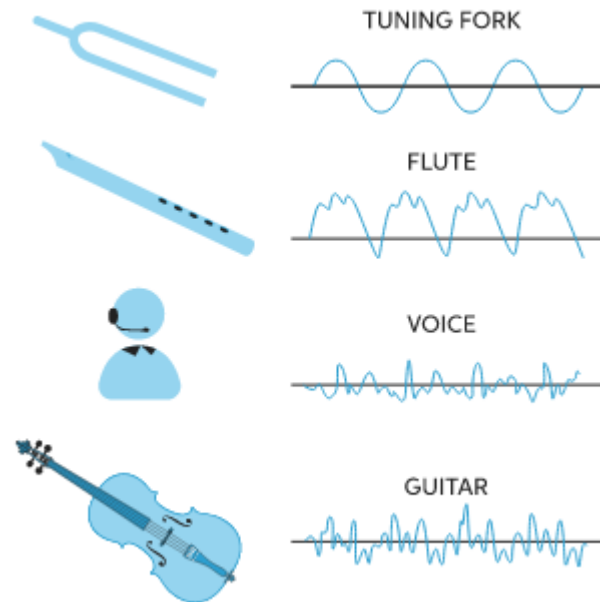
03 음성 데이터



소리의 형태

소리

- 공기 속을 전해오는 파동
- 소리의 3요소
 1. 세기
 - 파동의 진폭, 소리의 크기
 2. 높낮이
 - 파동의 주파수, 소리의 높낮이
 3. 맵시
 - 파동의 파형, 소리의 색상(피아노, 바이올린)



음색: 음의 높이, 크기가 같아도 가지는 고유한 특징
물체마다 발생하는 파동 모양으로 결정,
중요한 요인은 파장구조

03 음성 데이터



소리의 샘플링 레이트

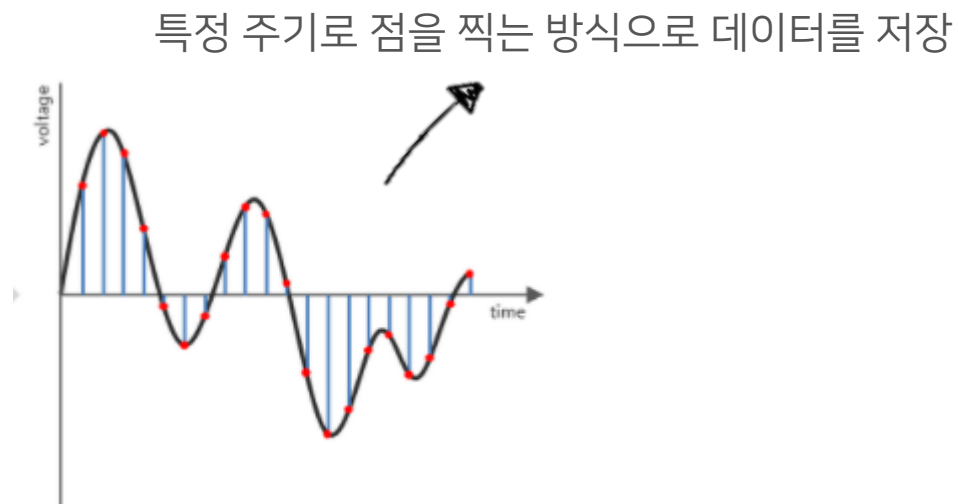
음성 데이터 저장 방식



연속적인 아날로그 소리



디지털화



불연속적인 디지털 소리

03 음성 데이터



소리의 샘플링 레이트

샘플링 레이트 (Sampling Rate)



1초에 몇 개의 샘플을 추출할 것인가



샘플링 레이트가 높을수록,
아날로그와 유사한 디지털 값을 얻음

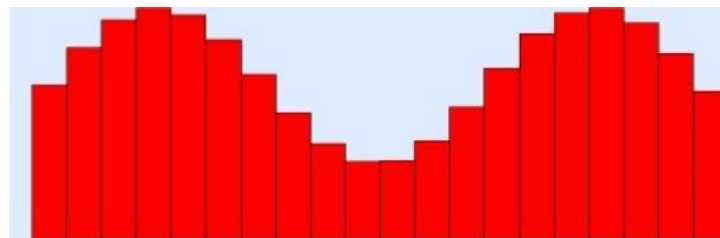


Fig 4: 100 Samples per Second (100 Hz)

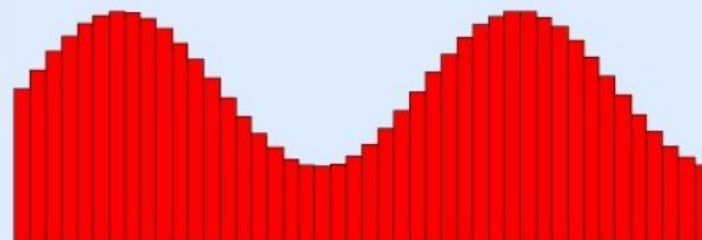


Fig 5: 200 Samples per Second (200 Hz)

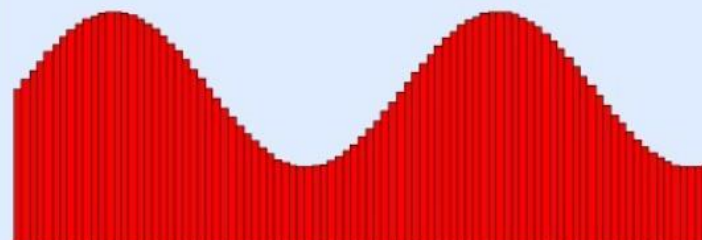


Fig 6: 400 Samples per Second (400 Hz)

03 음성 데이터

데이터 소개

음성 분석 연구



AI Hub

비전

음성/자연어

교육

국토환경

능축수산

안전

자율주행

헬스케어

인공지능 학습용 데이터
다운로드 프로그램 설치

Windows & Mac용

> 간단 사용설명서

> 맥용 설치&삭제 가이드

개방 데이터 ▾ 외부 데이터 ▾ 활용 사례 ▾ 개발 지원 ▾ 경진대회 ▾ 게시판 ▾ 마이페이지 로그아웃

음성/자연어

음성/자연어 감성 대화 말뭉치 텍스트 오디오 2020	음성/자연어 고객 응대 음성 텍스트 오디오 2020	음성/자연어 고서 한자 인식 OCR 이미지 텍스트 2020
음성/자연어 공공행정문서 OCR 이미지 2020	음성/자연어 기계독해 텍스트 2018	음성/자연어 논문자료 요약 텍스트 2020
음성/자연어 다양한 형태의 한글 문자 OCR 이미지 2020	음성/자연어 도서자료 기계독해 텍스트 2020	음성/자연어 도서자료 요약 텍스트 2020
음성/자연어 명령어 음성(노인남녀) 텍스트 오디오 2020	음성/자연어 명령어 음성(소아, 유아) 텍스트 오디오 2020	음성/자연어 명령어 음성(일반남녀) 텍스트 오디오 2020

03 음성 데이터

데이터 소개

음성 분석 연구

데이터는 얼마나 크냐면요...

이승우 17 통계

데이터가 총 5TB정도 한다는것도

어디 써주라

자랑하게

오후 3:51

03 음성 데이터



데이터 소개

음성 분석 연구

데이터는 얼마나 크냐면요...

5 TB

03 음성 데이터



데이터 소개

데이터 소개

음성/자연어 명령어 음성(노인남녀) 텍스트 오디오 2020	음성/자연어 명령어 음성(소아, 유아) 텍스트 오디오 2020	음성/자연어 명령어 음성(일반남녀) 텍스트 오디오 2020
음성/자연어 자유대화 음성(일반남녀) 텍스트 오디오 2020	음성/자연어 자유대화 음성(노인남녀) 텍스트 오디오 2020	음성/자연어 자유대화 음성(소아, 유아) 텍스트 오디오 2020
음성/자연어 한국어 방언 발화(강원도) 텍스트 오디오 2020	음성/자연어 한국어 방언 발화(경상도) 텍스트 오디오 2020	음성/자연어 한국어 방언 발화(전라도) 텍스트 오디오 2020
음성/자연어 한국어 방언 발화(제주도) 텍스트 오디오 2020	음성/자연어 한국어 방언 발화(충청도) 텍스트 오디오 2020	

data

- > 명령어 음성(노인남녀)
- > 명령어 음성(소아, 유아)
- > 명령어 음성(일반남녀)
- > 자유대화 음성(노인남녀)
- > 자유대화 음성(소아, 유아)
- > 자유대화 음성(일반남녀)
- > 한국어 방언 발화(강원도)
- > 한국어 방언 발화(경상도)
- > 한국어 방언 발화(전라도)
- > 한국어 방언 발화(제주도)
- > 한국어 방언 발화(충청도)

03 음성 데이터



데이터 소개

데이터 소개



모든 데이터 셋은
Training과 Validation으로 구분 후,
그 안에 [라벨]과 [원천]으로 구분

03 음성 데이터



데이터 소개

명령어 음성

```
data > 명령어 음성(노인남녀) > Training > [라벨]1.AI비서_라벨링_명령어(노년)_training > n_0879 > {} n_0879-12001-02-01-KAJ-F-09-A.json > ...
1  {
2    "기본정보":{"Language":"KOR","Version":"N/A","ApplicationCategory":"N/A","NumberOfSpeaker":"N/A",
3      "NumberOfUtterance":"N/A","DataCategory":"AI 비서","RecordingDate":"2021-01-13 05:15:39",
4      "FillingDate":"N/A","RevisionHistory":"N/A","Distributor":"Mediazen"},
5    "음성정보":{"SamplingRate":"48000","ByteOrder":"N/A","EncodingLaw":"SignedIntegerPCM",
6      "NumberOfBit":"16","NumberOfChannel":"1","SignalToNoiseRatio":"N/A"},
7    "전사정보":{"LabelText":"운동으로 체조 하려는데 도와 주면 좋겠네."},
8    "화자정보":{"SpeakerName":"KAJ","Gender":"Female","Age":"over70","Region":"서울/인천/경기","Dialect":"경기/서울"},
9    "환경정보":{"RecordingEnviron":"가정","NoiseEnviron":"가정","RecordingDevice":"휴대폰"},
10   "파일정보":{"FileCategory":"Audio","FileName":"n_0879-12001-02-01-KAJ-F-09-A.wav",
11     "DirectoryPath":"/mnt/data1/namz/nia/metrixA/data/2021-01-13/n_0879","HeaderSize":"44","FileLength":"4.38",
12     "FileFormat":"PCM","NumberOfRepeat":"1","TimeInterval":"0","Distance":"30"},
13   "기타정보":{"QualityStatus":"Good"}
14 }
```

JSON 파일

기본정보 / 음성정보 / 전사정보 / 화자정보 / 환경정보 / 파일정보 / 기타정보

03 음성 데이터



데이터 소개

자유대화 음성

```
data > 자유대화 음성(노인남녀) > Training > [라벨]1.AI챗봇 > 노인남여_노인대화07_F_1522434093_60_경상_실내 > {} 노인남여_노인대화07_F_1522434093_60_경상_실내_08580.json >
1  {
2    "발화정보" : { "stt" : "밥 한끼를 제대로 된 밥 한끼를 먹을 수 있다고 생각하면서", "scriptId" : "노인대화-08580",
3                  "fileName" : "노인남여_노인대화07_F_1522434093_60_경상_실내_08580.wav", "recrdTime" : "4.520",
4                  "recrdQuality" : "16K", "recrdDt" : "2020-11-21 20:44:24", "scriptSetNo" : "T_노인대화_7" },
5    "대화정보" : { "recrdEnvrn" : "실내", "colctUnitCode" : "AI 챗봇",
6                  "cityCode" : "경상", "recrdUnit" : "AndroidOS", "convrsThema" : " 방송/연예 " },
7    "녹음자정보" : {
8                  "gender" : "여", "recorderId" : "1522434093", "age" : 60 }
9  }
```

JSON 파일

발화정보 / 대화정보/ 녹음자정보

03 음성 데이터



데이터 소개

한국어 방언 발화

```
data > 한국어 방언 발화(강원도) > Training > [라벨]강원도_학습데이터_1 > 강원도_학습데이터_1 > {} DGDQ20000020.json > [ ] speaker
1 {
2   "id": "DGDQ20000020",
3   "metadata": {
4     "title": "강원방언 AI 학습데이터 DGDQ20000020", "creator": "디큐", "distributor": "디큐",
5     "year": "2020", "category": "강원방언 > 사적 대화 > 일상 대화", "annotation_level": [ "" ],
6     "sampling": "본문 전체", "author": "개인 발화자", "publisher": "개인 발화 녹음",
7     "date": "20201208", "topic": "여행지(국내/해외)" },
8   "speaker": [
9     { "id": 1, "name": "이*레", "age": "50대", "occupation": "농업/임업/어업 종사자",
10      "sex": "여성", "birthplace": "강원", "principal_residence": "강원",
11      "current_residence": "강원", "education": "고졸" },
12   ],
13   "setting": {
14     "relation": "이웃사촌" },
15   "utterance":
16     {
17       "id": "DGDQ20000020.1.1.1", "form": "글쎄요", "standard_form": "글쎄요",
18       "dialect_form": "글쎄요", "speaker_id": "1", "start": 1.35,
19       "end": 2.22, "note": "",
20       "eojeolList": [
21         { "id": 1, "eojeol": "글쎄요", "standard": "글쎄요", "isDialect": false }
22       ]
23     }
24 }
```

JSON 파일

Id / metadata / speaker /
Setting /utterance

03 음성 데이터



데이터 소개

한국어 방언 발화

```
{
  "id": "DGDQ20000020.1.1.3",
  "start": 3.94,
  "end": 5.35,
  "speaker_id": "2",
  "form": "원 얘기부터 (할까요?)/(할까요?)",
  "standard_form": "원 얘기부터 할까요?",
  "dialect_form": "원 얘기부터 할까요?",
  "note": "",
  "eojeolList": [
    {
      "id": "DGDQ20000020.1.1.4",
      "form": "바다 얘기 좀 해주세요 @웃음",
      "standard_form": "바다 얘기 좀 해주세요 {laughing}",
      "dialect_form": "바다 얘기 좀 해주세요 {laughing}",
      "speaker_id": "1",
      "start": 5.35,
      "end": 6.98,
      "note": "",
      "eojeolList": [
```

```
{
  "id": "DGDQ20000020.1.1.5",
  "form": "바다에요?",
  "standard_form": "바다에요?",
  "dialect_form": "바다에요?",
  "speaker_id": "2",
  "start": 7.03,
  "end": 7.72,
  "note": "",
  "eojeolList": [
    {
      "id": "DGDQ20000020.1.1.7",
      "form": "바다의 무슨 얘기를 해야 되나?",
      "standard_form": "바다의 무슨 얘기를 해야 되나?",
      "dialect_form": "바다의 무슨 얘기를 해야 되나?",
      "speaker_id": "2",
      "start": 8.54,
      "end": 10.49,
      "note": "",
      "eojeolList": [
```

한 파일 내에 여러 음성 존재

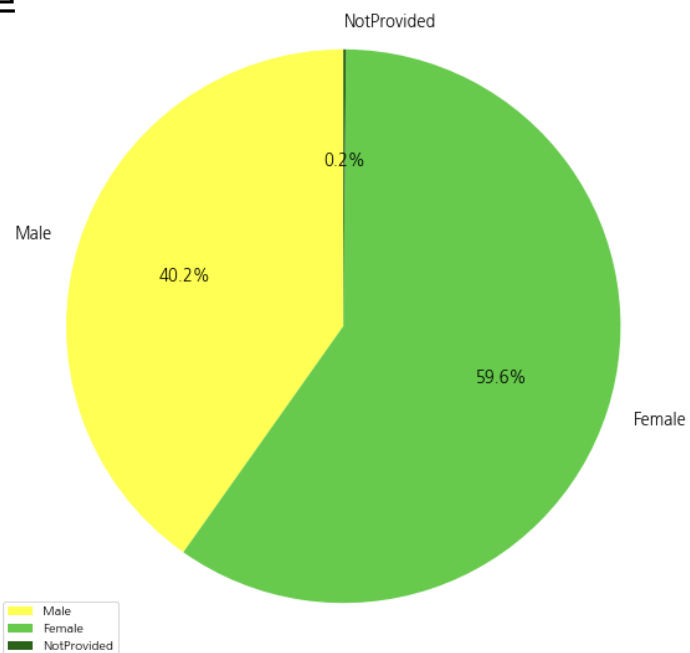
03 음성 데이터



EDA

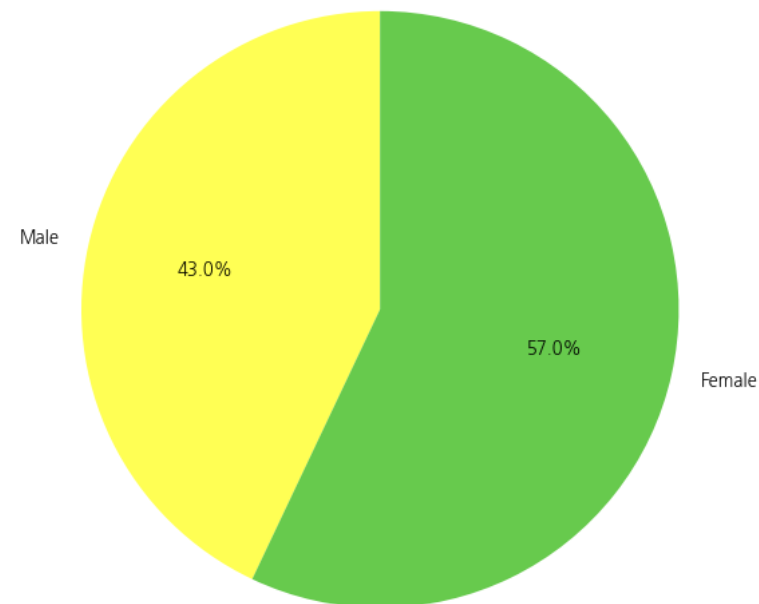


성별



명령어 음성

여성이 20%p 정도 더 많음



자유 대화 음성

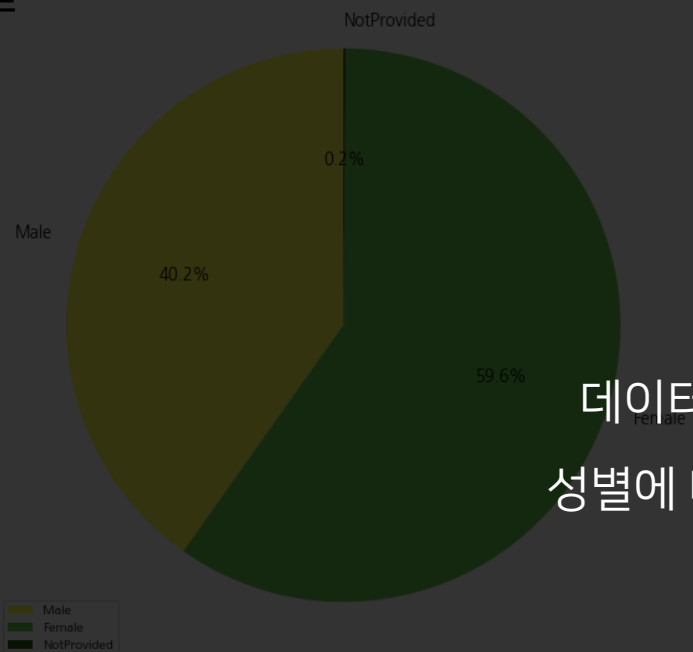
여성이 12%p 정도 더 많음

03 음성 데이터



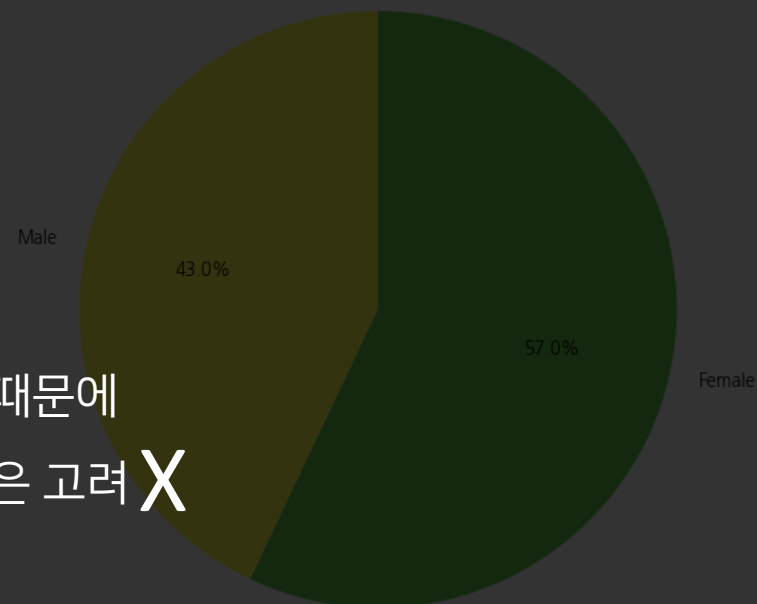
EDA

성별



명령어 음성

여성이 20%p 정도 더 많음



자유 대화 음성

여성이 12%p 정도 더 많음

데이터의 수가 아주 많기 때문에
성별에 대한 데이터 불균형은 고려 X

03 음성 데이터

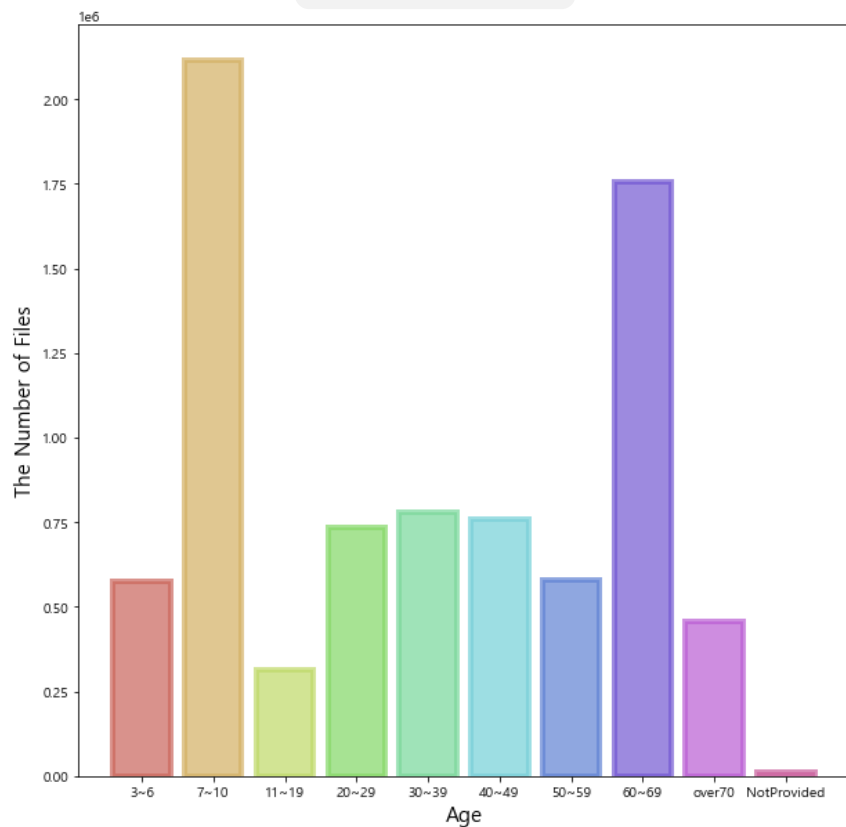


EDA

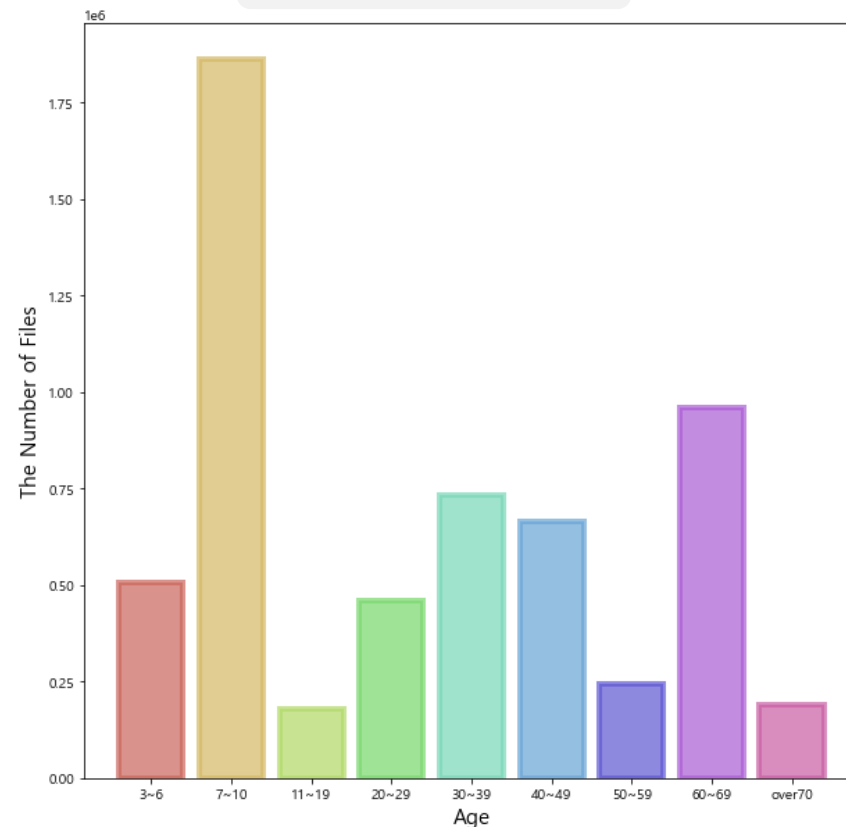


연령

명령어 음성



자유 대화 음성



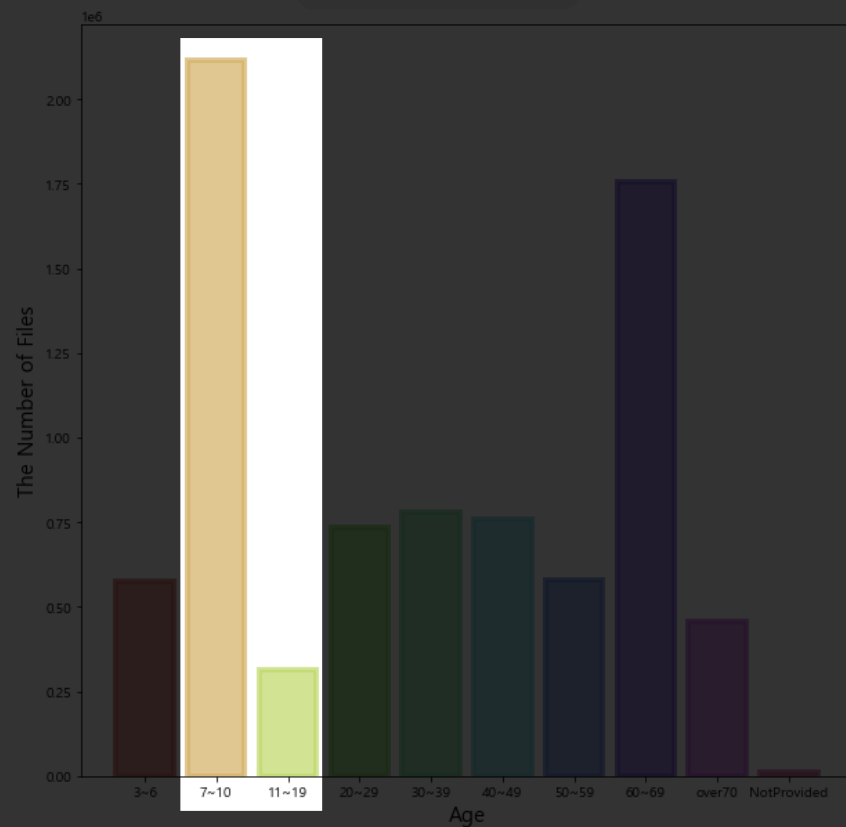
03 음성 데이터



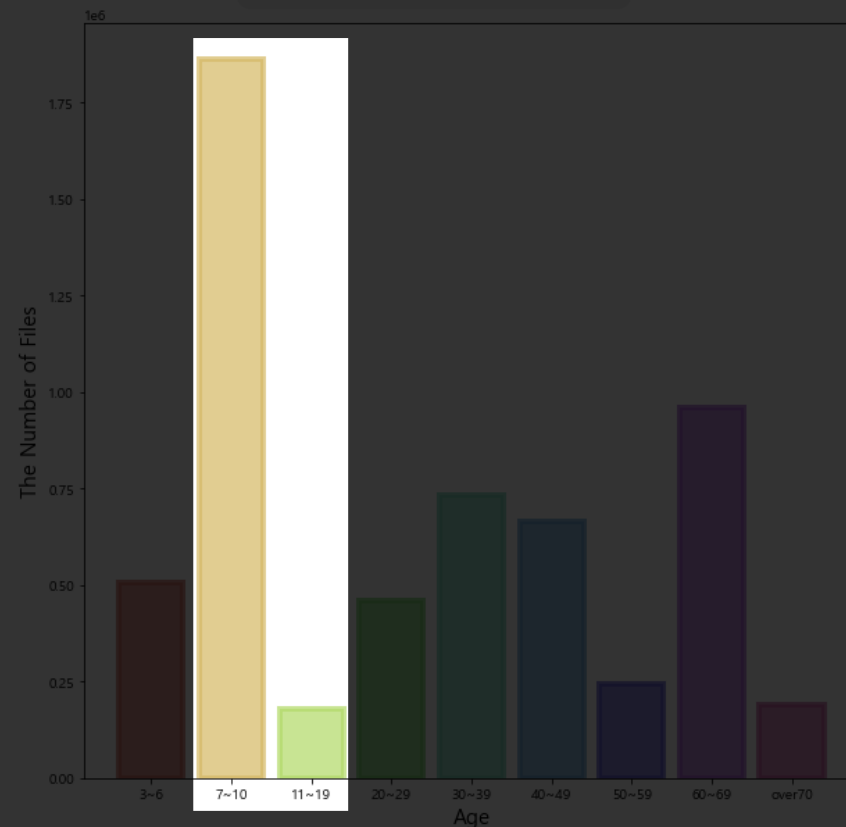
EDA

연령

명령어 음성



자유 대화 음성



연령에 대한 데이터 불균형 고려 필요

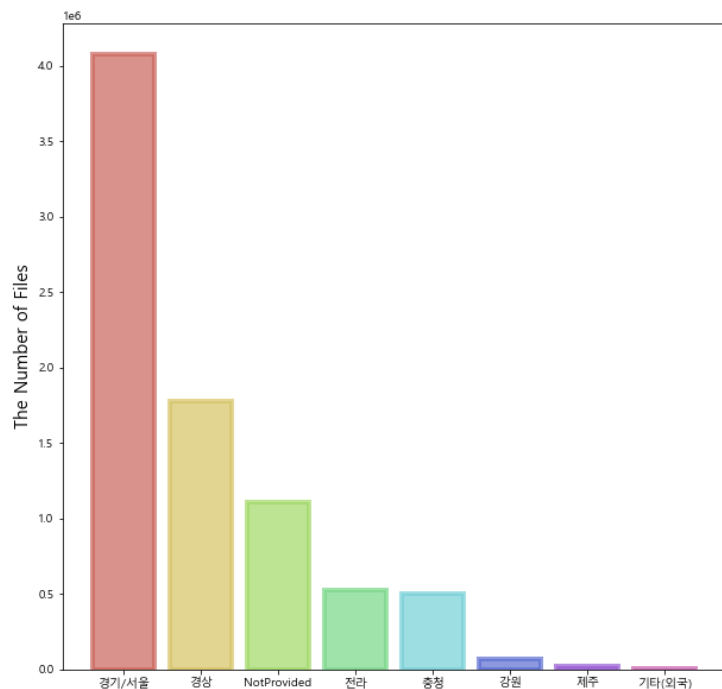
03 음성 데이터



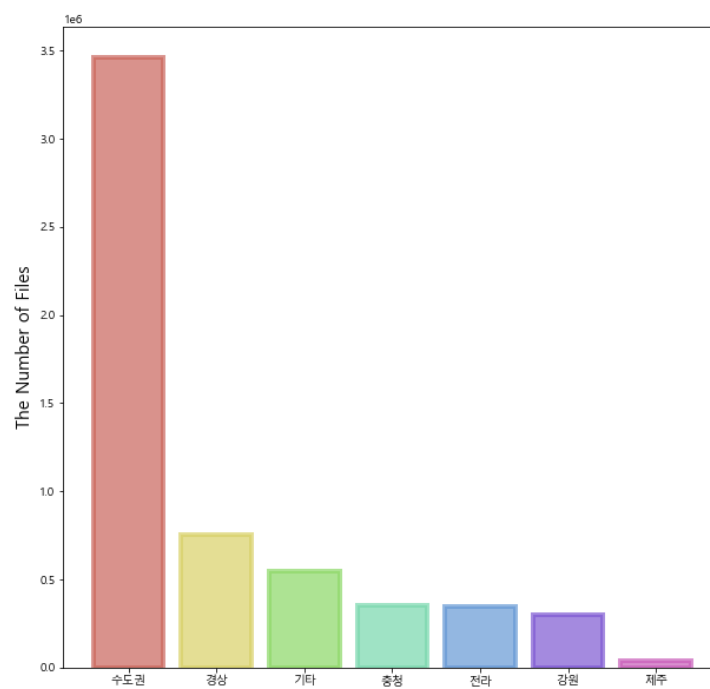
EDA



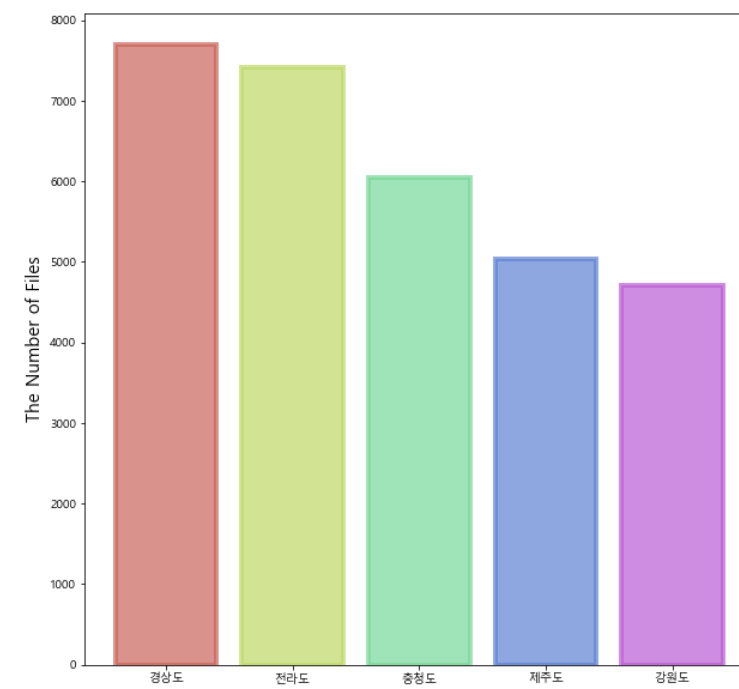
방언



명령어 음성



자유 대화 음성



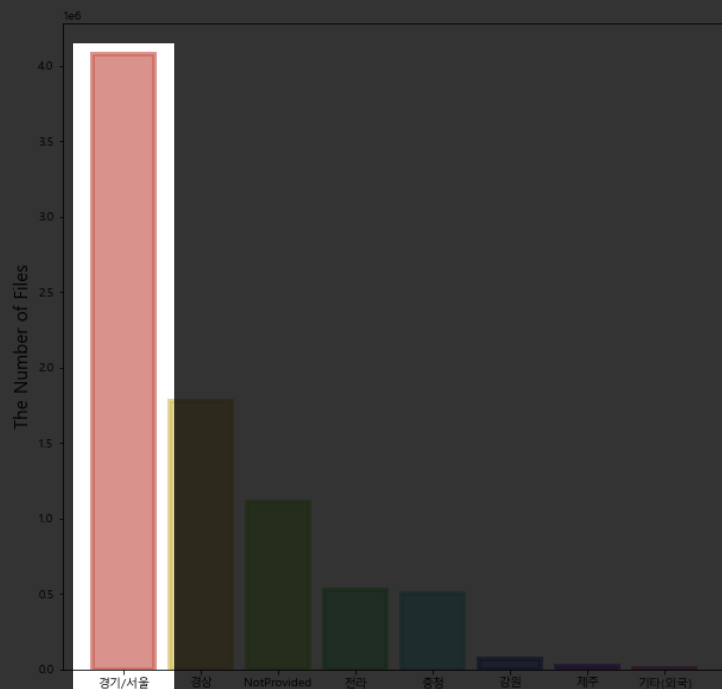
한국어 방언 발화

03 음성 데이터

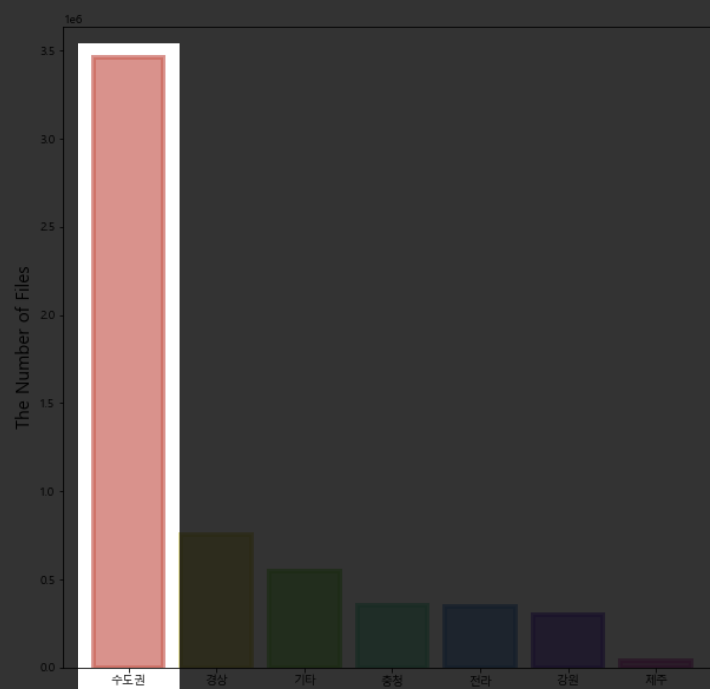


EDA

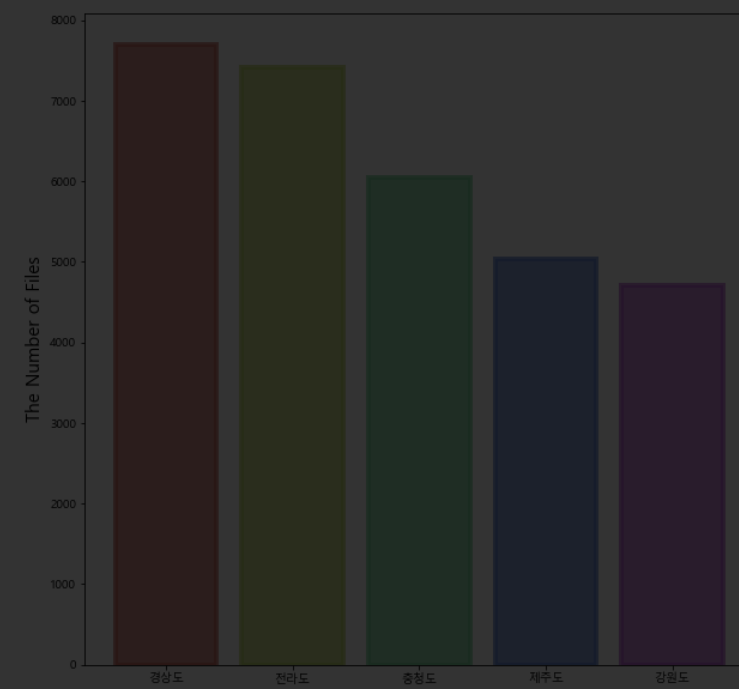
방언



명령어 음성



자유 대화 음성



한국어 방언 발화

방언에 대한 데이터 불균형 고려 필요

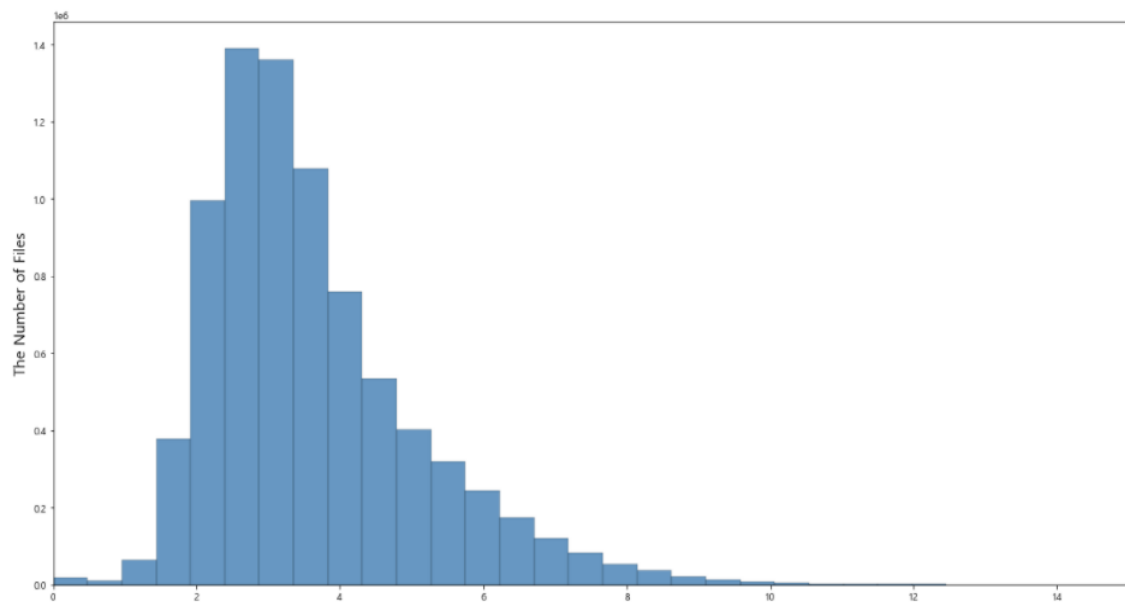
03 음성 데이터



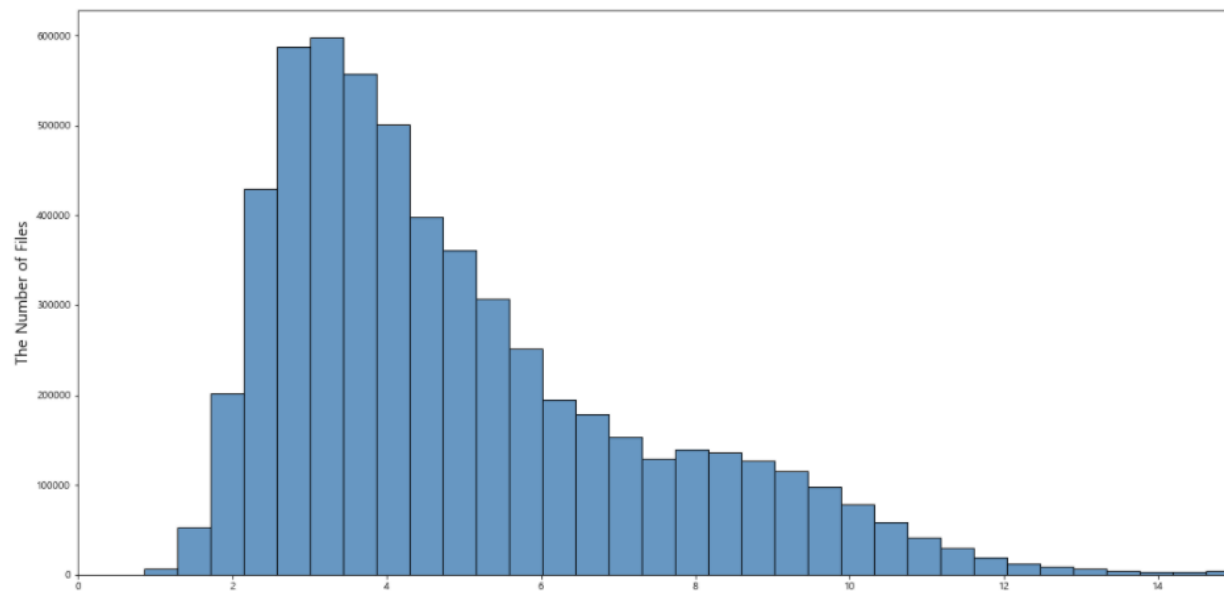
EDA



데이터 길이



명령어 음성



자유 대화 음성

대부분 10초 내의 짧은 음성 데이터

03 음성 데이터



EDA



명령어 음성

```
FileLength
2.76      164549
2.88      160417
2.58      155666
2.46      155170
2.82      152579
...
22.38      1
14.77      1
21.78      1
45.06      1
156.42     1
Name: FileLength, Length: 1664, dtype: int64
Unique한 값 자체의 개수: 1664
```

File Length

길이가 굉장히 긴 경우도 존재
파일을 들어보니 녹음자의 실수의 결과
해당 데이터 삭제

03 음성 데이터



EDA

오디오 파일 분할

```
"speaker": [  
  {  
    "id": "1",  
    "name": "이*례",  
    "age": "50대",  
    "occupation": "농업/임업/어업 종사자",  
    "sex": "여성",  
    "birthplace": "강원",  
    "principal_residence": "강원",  
    "current_residence": "강원",  
    "education": "고졸"  
  },  
  {  
    "id": "2",  
    "name": "김*희",  
    "age": "60대 이상",  
    "occupation": "농업/임업/어업 종사자",  
    "sex": "여성",  
    "birthplace": "강원",  
    "principal_residence": "강원",  
    "current_residence": "강원",  
    "education": "고졸"  
  },  
  {  
    "id": "3",  
    "name": "D*5004",  
    "age": "50대",  
    "occupation": "주부",  
    "sex": "여성",  
    "birthplace": "강원",  
    "principal_residence": "강원",  
    "current_residence": "강원",  
    "education": "고졸"  
  }  
]
```

[방언 발화 데이터]

발화자 수가 여러 명 이고 대화 형식의 12~13분 데이터



발화자 기준으로 나눈 후, 시간 단위로 자를 예정

04

분석 과정



04 분석 과정

 전처리

JSON 파일 CSV 변환



여러 개의 JSON파일을 한 개의 CSV파일로 변환

04 분석 과정



JSON 파일 CSV 변환

data

명령어 음성(노인남녀)

Training

[라벨]1.이비서_라벨링_명령어(노년)_training

[라벨]2.이로봇_라벨링_명령어(노년)_training

[라벨]3.키오스크_라벨링_명령어(노년)_training

[라벨]4.비정형_라벨링_명령어(노년)_training

[원전]1.이비서_원전_1_명령어(노인)_training

[원전]1.이비서_원전_2_명령어(노인)_training

[원전]1.이비서_원전_3_명령어(노인)_training

[원전]1.이비서_원전_4_명령어(노인)_training

[원전]1.이비서_원전_5_명령어(노인)_training

[원전]1.이비서_원전_6_명령어(노인)_training

[원전]1.이비서_원전_7_명령어(노인)_training

[원전]1.이비서_원전_8_명령어(노인)_training

[원전]2.이로봇_원전_1_명령어(노인)_training

[원전]2.이로봇_원전_2_명령어(노인)_training

[원전]2.이로봇_원전_3_명령어(노인)_training

[원전]2.이로봇_원전_4_명령어(노인)_training

[원전]2.이로봇_원전_5_명령어(노인)_training

[원전]2.이로봇_원전_6_명령어(노인)_training

[원전]2.이로봇_원전_7_명령어(노인)_training

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

"기본정보":

{

"Language": "KOR",

"Version": "N/A",

"ApplicationCategory": "N/A",

"NumberOfSpeaker": "N/A",

"NumberOfUtterance": "N/A",

"DataCategory": "AI 키오스크",

"RecordingDate": "2021-01-14 05:04:21",

"FillingDate": "N/A",

"RevisionHistory": "N/A",

"Distributor": "Mediazen"

}

"음성정보":

{

"SamplingRate": "48000",

"ByteOrder": "N/A",

"EncodingLaw": "SignedIntegerPCM",

"NumberOfBit": "16",

"NumberOfChannel": "1",

"SignalToNoiseRatio": "N/A"

}

"전사정보":

{

"LabelText": "주차 요금 계산할게."

}

"화자정보":

{

"SpeakerName": "KHS",

"Gender": "Female",

"Age": "60-69",

"Region": "대전/세종/충청/강원",

}

Language	Version	Application	NumberOfSpeaker	NumberOfUtterance	DataCategory	RecordingDate
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21
KOR	N/A	N/A	N/A	N/A	AI 키오스크	2021-01-14 05:04:21

데이터셋 하나당 하나의 csv파일 생성 후 아래 목적으로 이용 예정

① EDA

② DataLoader

04 분석 과정



전처리

데이터 문제

Json파일의 문제

1. 코드 수정

`Text": "\아침 식사 시간 맞춰서 커피 미리 내려 줘"},`

`JSONDecodeError: Invalid \escape:`

```
class LazyDecoder(json.JSONDecoder):
    def decode(self, s, **kwargs):
        regex_replacements = [
            (re.compile(r'([^\w])\w+([^\w])'), r'\1#####2'),
            (re.compile(r',(\w+)'), r'\1'),
        ]
        for regex, replacement in regex_replacements:
            s = regex.sub(replacement, s)
        return super().decode(s, **kwargs)
```

2. 파일 수정(+try, except 사용)

`Text": "\아침 식사 시간 맞춰서 커피 미리 내려 줘"}.`

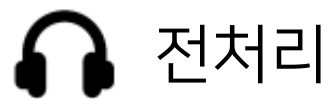
`JSONDecodeError: Expecting ',' delimiter:`

코드 수정으로 json 파일 로드 불가 -> 수작업 수정

But, 너무 많은 양으로 로드되지 않는 것은 넘기기로 결정

```
except JSONDecodeError as e:
    print(e)
except:
    print('error')
```

04 분석 과정



전처리

데이터 문제

Wav 파일의 문제

```
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710826.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710814.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710852.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710783.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710784.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710716.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710848.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710813.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710714.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710766.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710811.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710804.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710792.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710817.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710801.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710777.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710859.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710820.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710753.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710805.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710846.wav  
/root/data/한국어 방언 발화(강원도)/Training/[원천]강원도_10/DGIN20710800.wav
```

RIFF Header가 없어 사용이 불가능한
wav 파일들이 존재



사용 가능한 파일의 리스트 뽑기

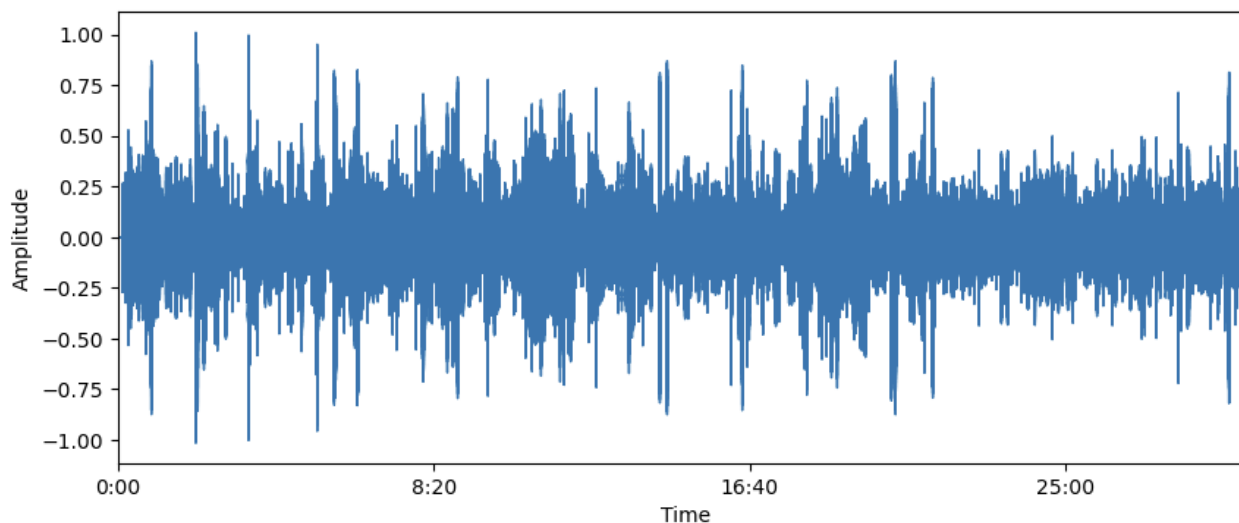
04 분석 과정



Feature engineering



Waveform



Waveform: 음파의 형태

Waveform을 활용하여 음성 데이터 구조 확인

방법: Librosa 또는 torchaudio 패키지를 사용하여 waveform 확인 가능

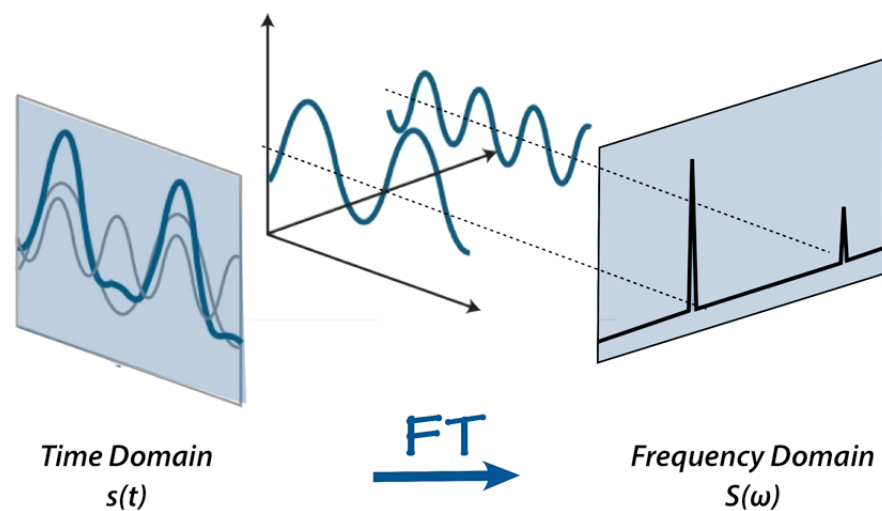
04 분석 과정



Feature engineering



Fourier Transformation



Fourier Transformation

오디오 데이터를 몇 가지 Frequency의 합으로 표현하여

시간 영역의 함수를 주파수 영역의 함수로 변환

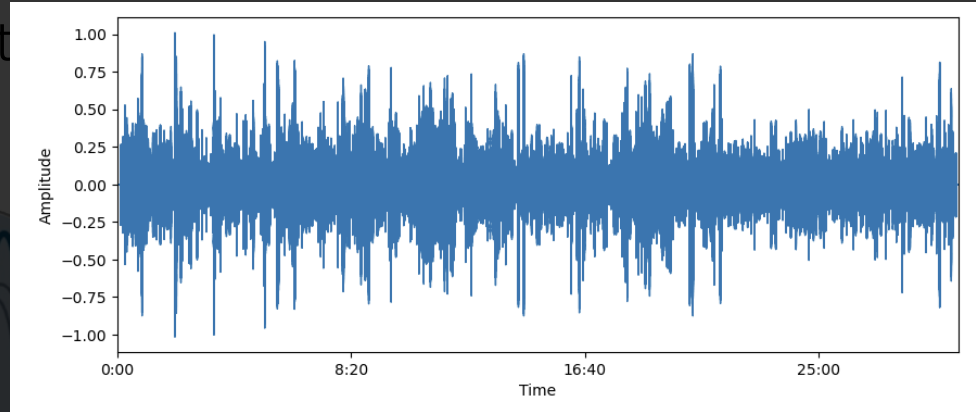
단점: 시간 정보 유실

04 분석 과정



Feature engineering

Fourier Transform



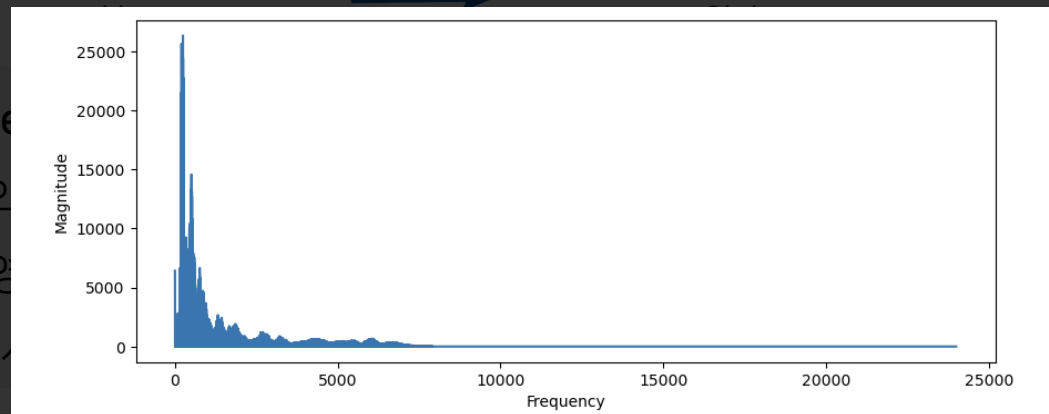
Waveform

Time Domain

FT



Fourier Transformation
Frequency Domain



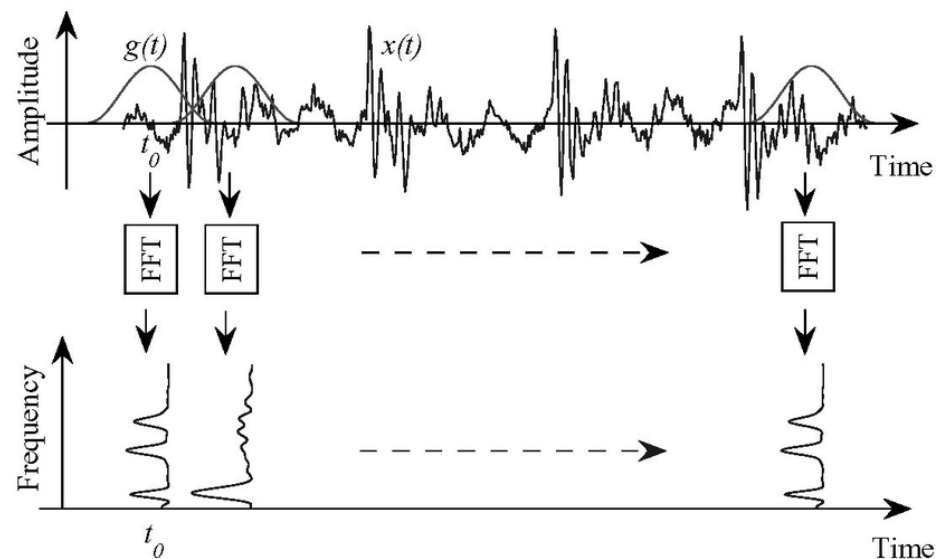
Fourier
오디오
시간 영
단점:

04 분석 과정



Feature engineering

STFT(Short-Time Fourier Transformation)



STFT

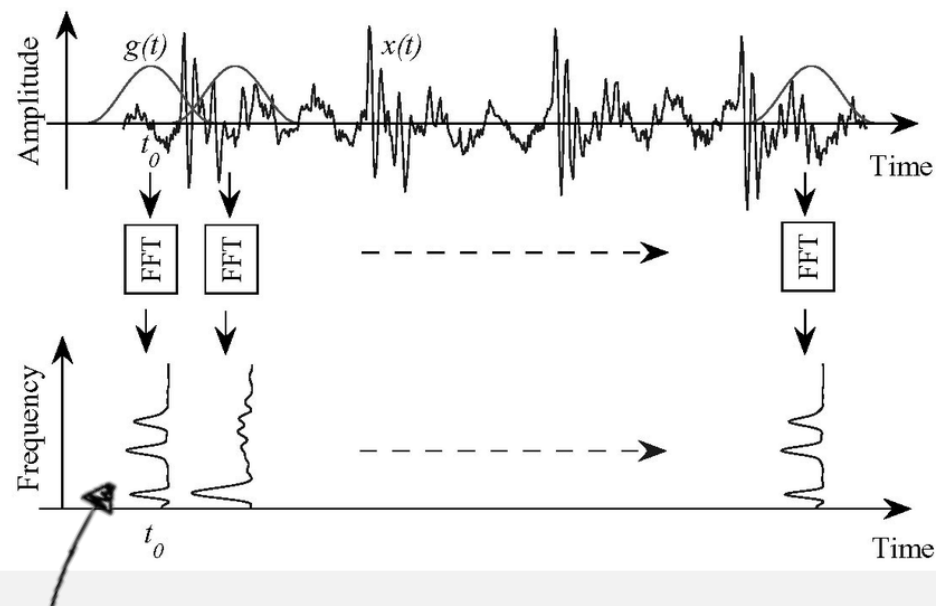
시간 정보를 잃는 Fourier Transformation의 보완을 위해
일정한 간격(frame)마다 Fourier Transformation 진행

04 분석 과정



Feature engineering

STFT(Short-Time Fourier Transformation)



Spectrum

각 프레임에서 Fourier Transformation을 거친 결과
x축은 주파수(frequency), y축은 진폭(magnitude)을 의미

04 분석 과정



Feature engineering

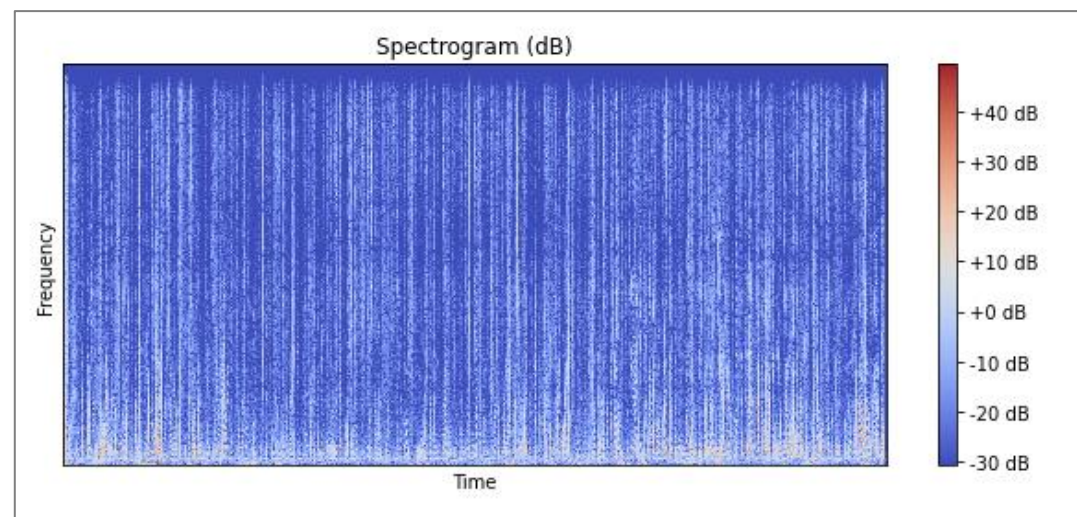


Spectrogram

Spectrogram

각 frame마다 나오는 Spectrum을 이어 붙여 time domain 복원

일반적으로 log scale인 dB단위를 적용해 log-Spectrum 이용

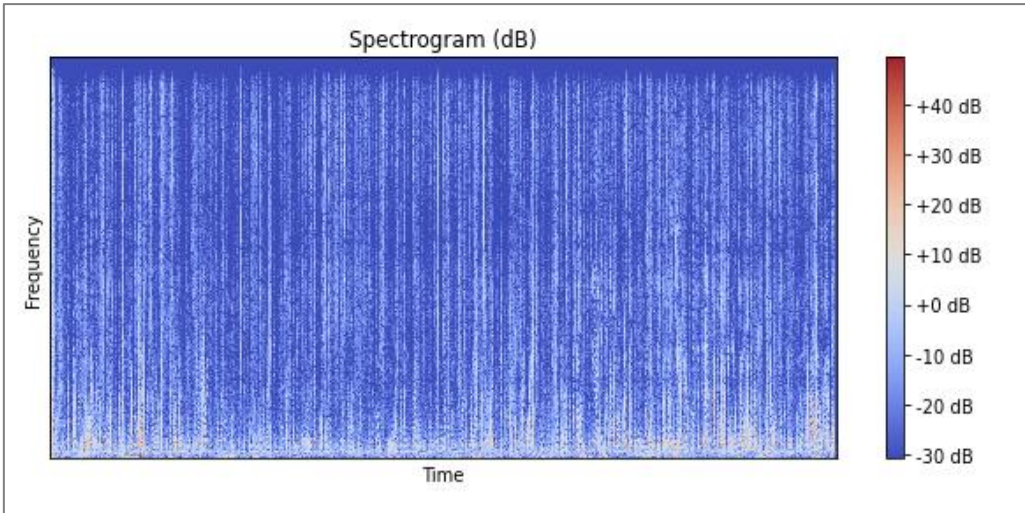


04 분석 과정



Feature engineering

Spectrogram



파형: x축은 시간, y축은 진폭
스펙트럼: x축은 주파수, y축은 진폭

Spectrogram: 파형과 스펙트럼의 특징이 결합된 것

x축은 시간(Time), y축은 주파수(Frequency),

z축은 진폭(Amplitude)

Sampling rate: 1초에 몇 번이나
data point를 찍을지

문제점: 데이터셋마다 데이터의 sampling rate가 다름



데이터 구조가 달라져 모델의 input으로 사용 어려움

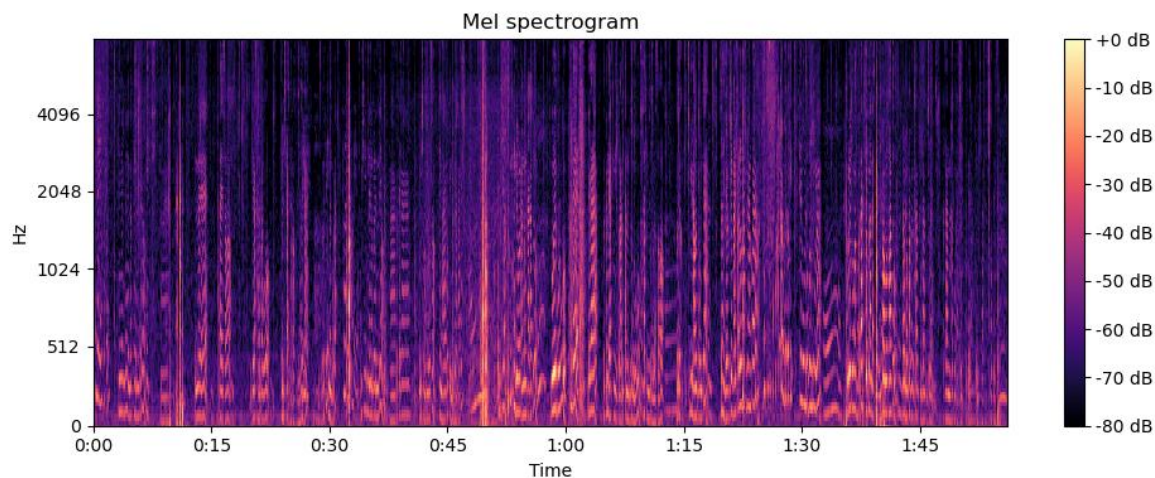
04 분석 과정



Feature engineering



Mel Spectrogram



Mel spectrogram

Spectrogram에 사람의 귀를 반영하는

mel-scale(log-scale)을 적용하는 것

사람 귀의 민감도

저주파수



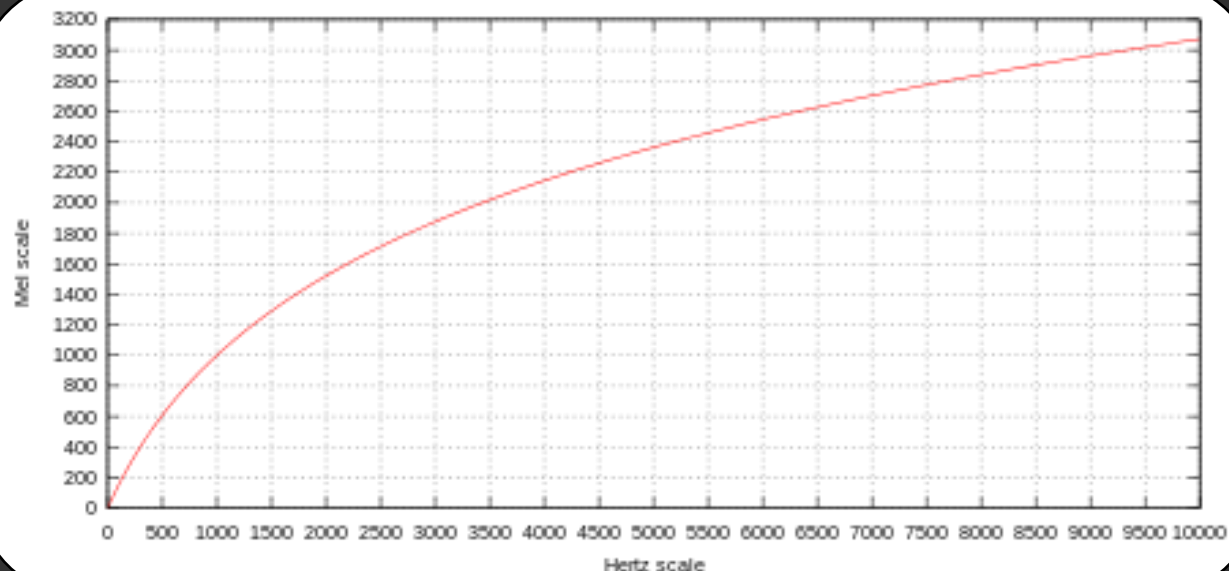
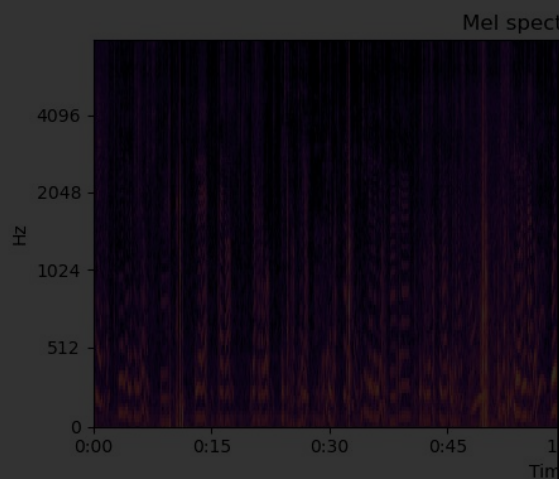
고주파수

04 분석 과정



Feature engineering

Mel Spectrogram



사람의 귀 작동 원리를 고려하여 스케일 단위 변환



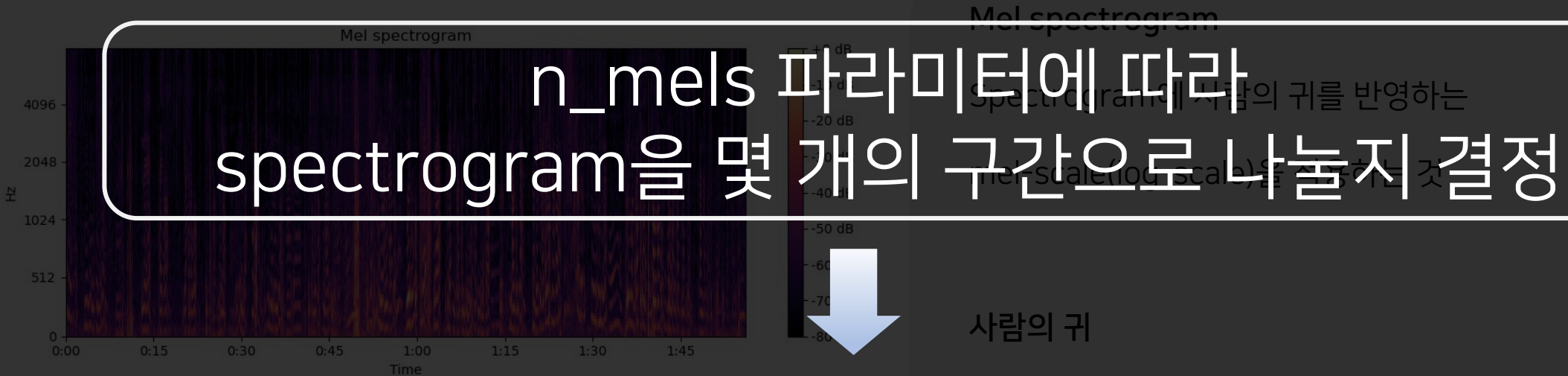
04 분석 과정



Feature engineering

Mel Spectrogram

Mel-Spectrogram



Sampling rate와 관계 없이 생성 가능

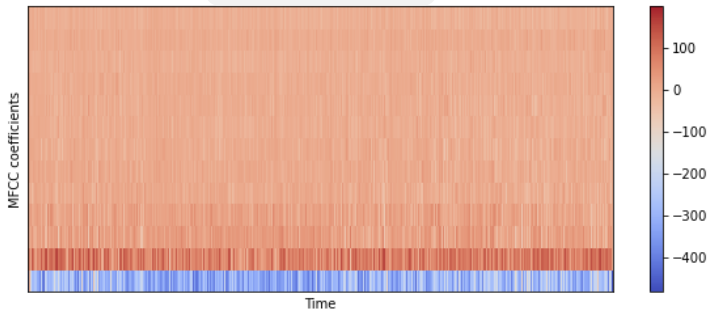
04 분석 과정



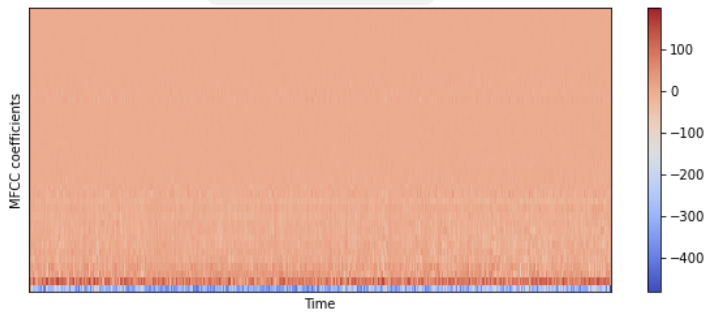
Feature engineering

MFCC(Mel Frequency Cepstral Coefficient)

차분값 추가 전



차분값 추가 후



MFCC

사람은 음성 신호를 linear scale로 받아들이는 것이 아니기에
사람의 청각이 예민하게 반응하는 정보를 강조하여
특징값을 추출하는 방법

보통 13개의 MFCC를 추출하며, 1차 차분, 2차 차분한 값을
포함해 총 39차원 벡터로 MFCC를 가장 많이 사용

04 분석 과정

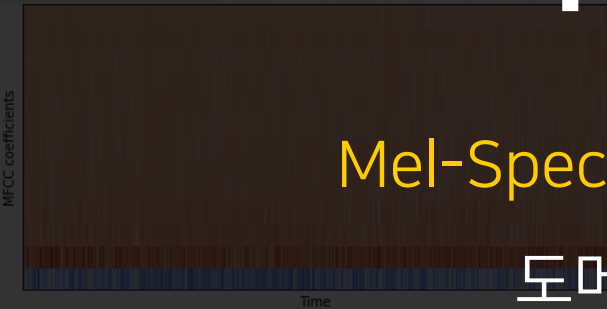


Feature engineering

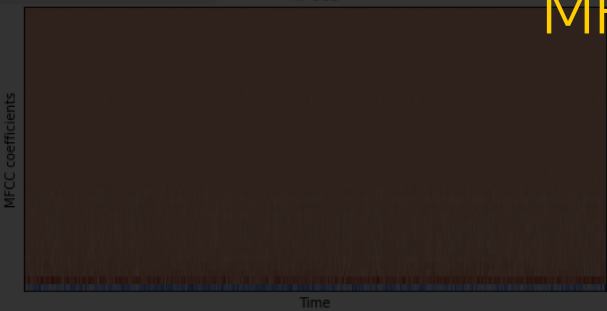
MFCC(Mel Frequency Cepstral Coefficient)

Mel-Spectrogram VS MFCC

차분값 추가 전



차분값 추가 후



MFCC

Mel-Spectrogram: 주파수끼리 Correlate하기 때문에

도메인이 한정적인 문제에서 더 좋은 성능

MFCC: De-Correlate를 해주기 때문에

일반적인 상황에서 더 좋은 성능

포함해 총 39차원 벡터로 MFCC를 가장 많이 사용

*컴퓨팅 파워가 부족할 때는 연산량이 적은 MFCC 선호

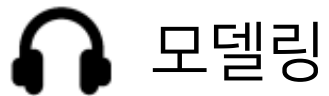
04 분석 과정

모델링

모델 개요

모델 유형	Input 유형	Input Size	모델 구성
CNN (Convolutional NN)	MFCC	<code>torch.size([3, 13, 126000])</code>	[Conv * Pool] Layer 2개 + FC Layer 2개
Vanilla RNN (Recurrent NN)	MFCC	<code>torch.size([126000, 39])</code>	RNN Hidden Layer 2개 + Sequence 길이 126000
CNN	Mel Spectrogram	<code>torch.size([2, 2000, 2000])</code>	[Conv * BN * Pool] Layer 4개 + FC Layer 2개
Vanilla RNN	Mel Spectrogram	<code>torch.size([5000, 256])</code>	RNN Hidden Layer 2개 + Sequence 길이 5000

04 분석 과정

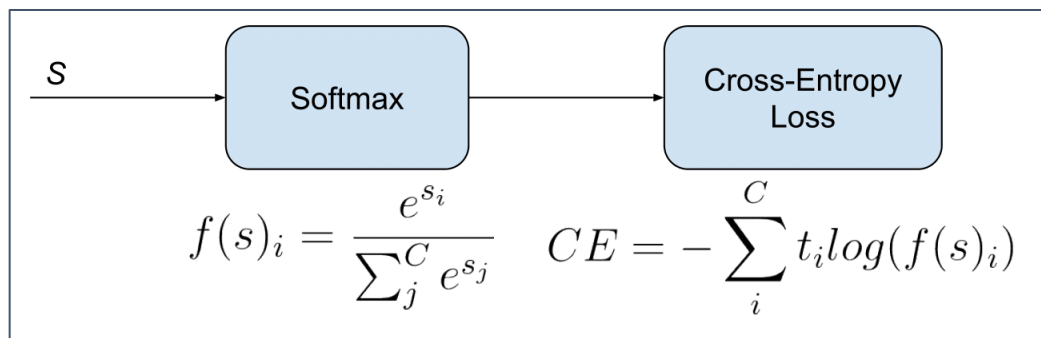


모델링



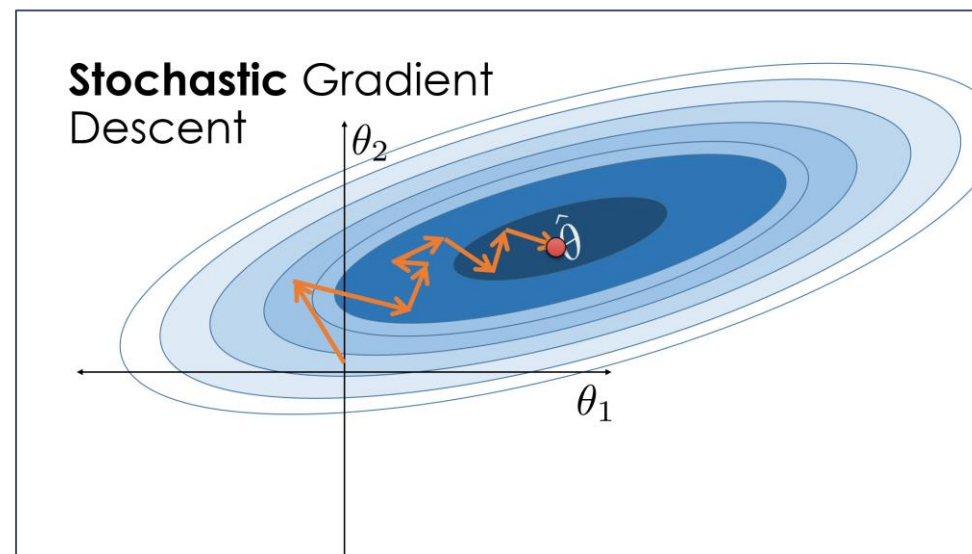
모델 개요

Loss Function



`nn.CrossEntropyLoss()`

Optimizer

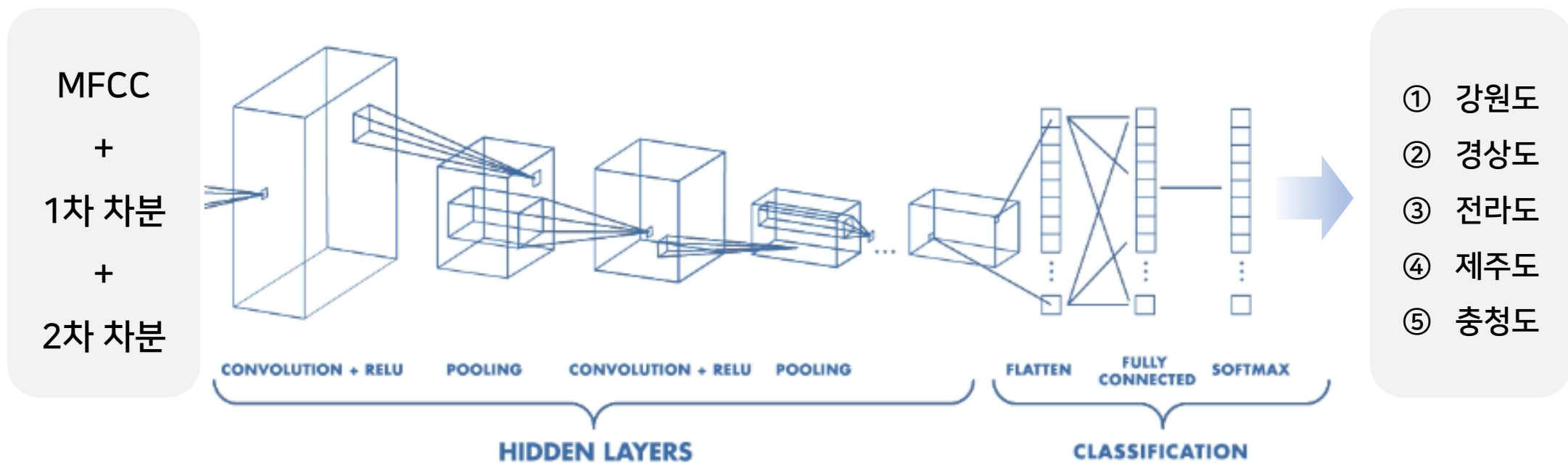


SGD

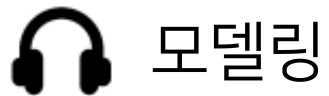
04 분석 과정

모델링

MFCC를 활용한 모델 - CNN

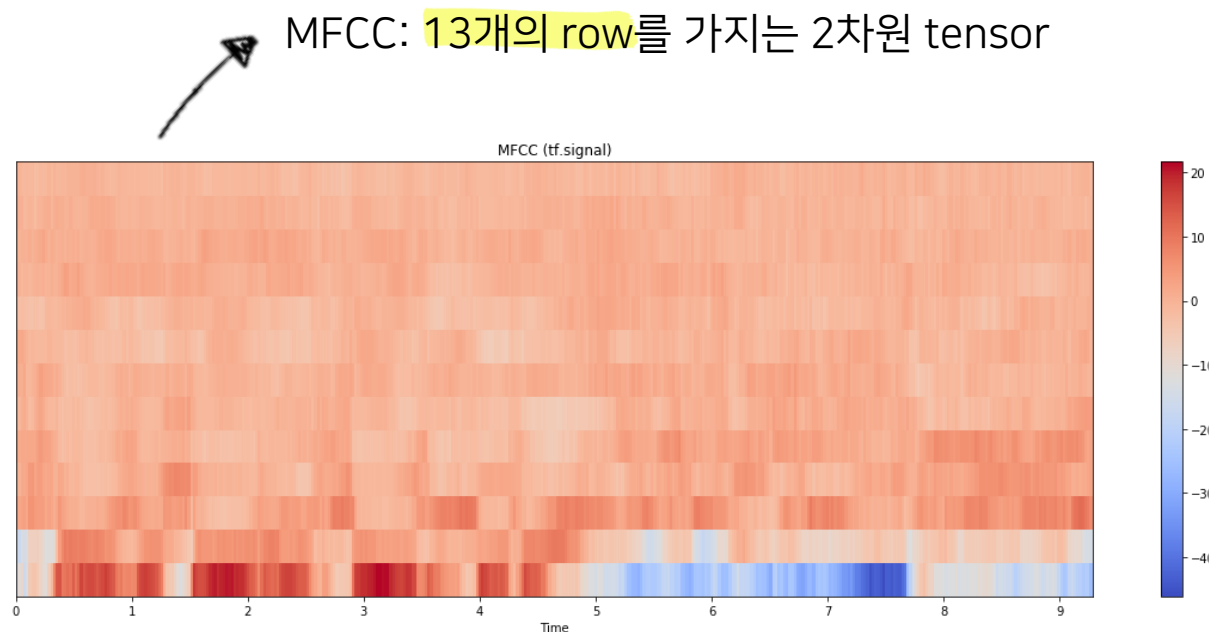


04 분석 과정

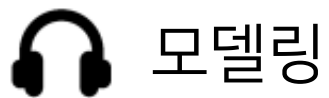


MFCC를 활용한 모델 - CNN

MFCC
+
1차 차분
+
2차 차분



04 분석 과정



MFCC를 활용한 모델 - CNN

MFCC
+
1차 차분
+
2차 차분



```
class ToMFCC(object):
    def __init__(self):
        pass

    def __call__(self, data):
        signal = data['signal']
        sr = data['sample_rate']

        self.n_fft = int(np.ceil(0.025 * sr))
        self.win_length = int(np.ceil(0.025 * sr))
        self.hop_length = int(np.ceil(0.01 * sr))

        audio_mfcc = torch.FloatTensor(librosa.feature.mfcc(y=signal.numpy().reshape(-1),
                                                            sr=sr,
                                                            n_mfcc=13,
                                                            n_fft=self.n_fft,
                                                            hop_length=self.hop_length))

        delta1 = torch.FloatTensor(librosa.feature.delta(audio_mfcc))
        delta2 = torch.FloatTensor(librosa.feature.delta(audio_mfcc, order=2))
        # mfcc_result = np.concatenate((audio_mfcc, delta1, delta2), axis=0)

        mfcc_result = torch.stack([audio_mfcc, delta1, delta2])
        mfcc_result = mfcc_pad1(mfcc_result)

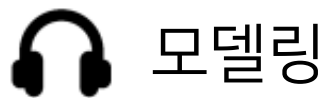
        data['MFCC'] = mfcc_result
        data['input'] = mfcc_result

    return data
```

librosa 라이브러리
feature.mfcc 함수

MFCC 추출

04 분석 과정



MFCC를 활용한 모델 - CNN

MFCC
+
1차 차분
+
2차 차분



```
class ToMFCC1(object):
    def __init__(self):
        pass

    def __call__(self, data):
        signal = data['signal']
        sr = data['sample_rate']

        self.n_fft = int(np.ceil(0.025 * sr))
        self.win_length = int(np.ceil(0.025 * sr))
        self.hop_length = int(np.ceil(0.01 * sr))

        audio_mfcc = torch.FloatTensor(librosa.feature.mfcc(y=signal.numpy().shape(-1),
                                                            sr=sr,
                                                            n_mfcc=13,
                                                            n_fft=self.n_fft,
                                                            hop_length=self.hop_length))

        delta1 = torch.FloatTensor(librosa.feature.delta(audio_mfcc))
        delta2 = torch.FloatTensor(librosa.feature.delta(audio_mfcc, order=2))
        # mfcc_result = np.concatenate((audio_mfcc, delta1, delta2), axis=0)

        mfcc_result = torch.stack([audio_mfcc, delta1, delta2])
        mfcc_result = mfcc_pad1(mfcc_result)

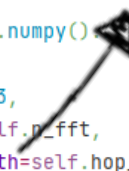
        data['MFCC'] = mfcc_result
        data['input'] = mfcc_result

    return data
```

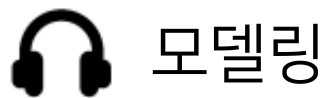
librosa 라이브러리
feature.delta 함수



1차 차분값 추출



04 분석 과정



MFCC를 활용한 모델 - CNN

MFCC
+
1차 차분
+
2차 차분

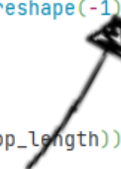


```
class ToMFCC(object):  
    def __init__(self):  
        pass  
  
    def __call__(self, data):  
        signal = data['signal']  
        sr = data['sample_rate']  
  
        self.n_fft = int(np.ceil(0.025 * sr))  
        self.win_length = int(np.ceil(0.025 * sr))  
        self.hop_length = int(np.ceil(0.01 * sr))  
  
        audio_mfcc = torch.FloatTensor(librosa.feature.mfcc(y=signal.numpy().reshape(-1),  
                                                             sr=sr,  
                                                             n_mfcc=13,  
                                                             n_fft=self.n_fft,  
                                                             hop_length=self.hop_length))  
  
        delta1 = torch.FloatTensor(librosa.feature.delta(audio_mfcc))  
        delta2 = torch.FloatTensor(librosa.feature.delta(audio_mfcc, order=2))  
        # mfcc_result = np.concatenate((audio_mfcc, delta1, delta2), axis=0)  
  
        mfcc_result = torch.stack([audio_mfcc, delta1, delta2])  
        mfcc_result = mfcc_pad1(mfcc_result)  
  
        data['MFCC'] = mfcc_result  
        data['input'] = mfcc_result  
  
    return data
```

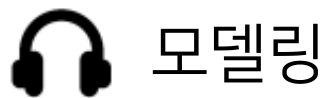
librosa 라이브러리
feature.delta 함수



2차 차분값 추출



04 분석 과정



MFCC를 활용한 모델 - CNN

MFCC
+
1차 차분
+
2차 차분



```
class ToMFCC1(object):
    def __init__(self):
        pass

    def __call__(self, data):
        signal = data['signal']
        sr = data['sample_rate']

        self.n_fft = int(np.ceil(0.025 * sr))
        self.win_length = int(np.ceil(0.025 * sr))
        self.hop_length = int(np.ceil(0.01 * sr))

        audio_mfcc = torch.FloatTensor(librosa.feature.mfcc(y=signal.numpy().reshape(-1),
                                                            sr=sr,
                                                            n_mfcc=13,
                                                            n_fft=self.n_fft,
                                                            hop_length=self.hop_length))

        delta1 = torch.FloatTensor(librosa.feature.delta(audio_mfcc))
        delta2 = torch.FloatTensor(librosa.feature.delta(audio_mfcc, order=2))
        # mfcc_result = np.concatenate((audio_mfcc, delta1, delta2), axis=0)

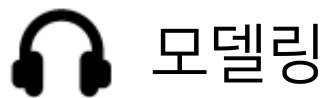
        mfcc_result = torch.stack([audio_mfcc, delta1, delta2])
        mfcc_result = mfcc_pad1(mfcc_result)

        data['MFCC'] = mfcc_result
        data['input'] = mfcc_result

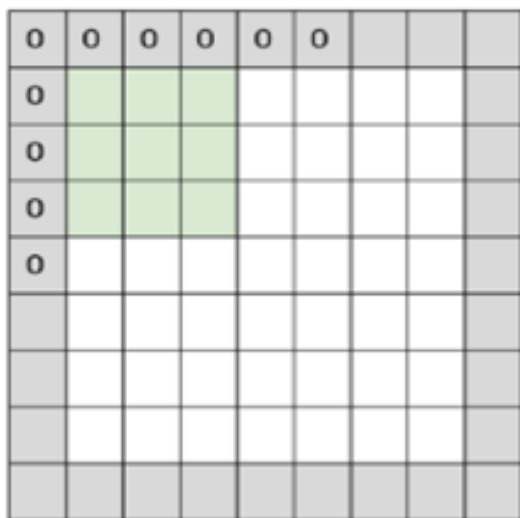
    return data
```

[3, 13, 126000]의
tensor 준비!

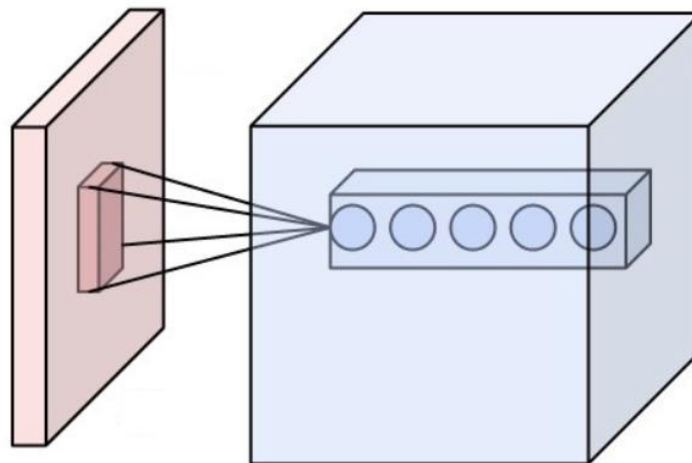
04 분석 과정



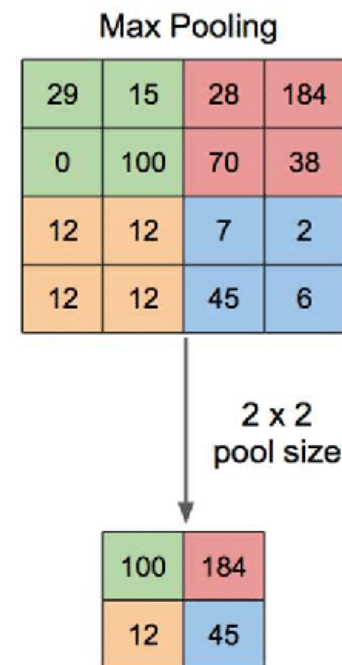
MFCC를 활용한 모델 - CNN



Input 위, 아래
2칸씩 Zero Padding

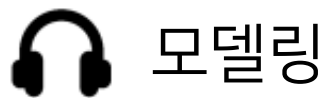


Convolutional Layer



Max Pooling

04 분석 과정

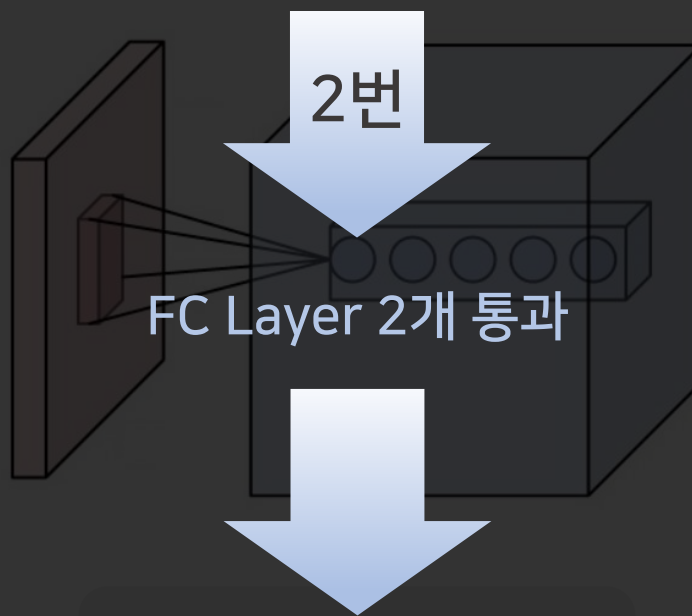


MFCC를 활용한 모델 - CNN

Padding → Convolutional Layer → Max Pooling



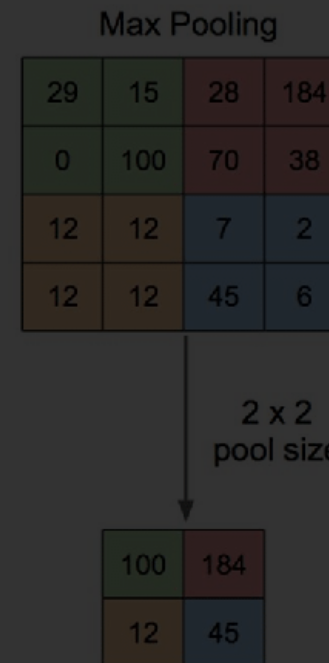
Input 위, 아래
2칸씩 Zero Padding



FC Layer 2개 통과

2번

Output
= [batch_size, 5가지 class의 score]
= [batch_size, 5]

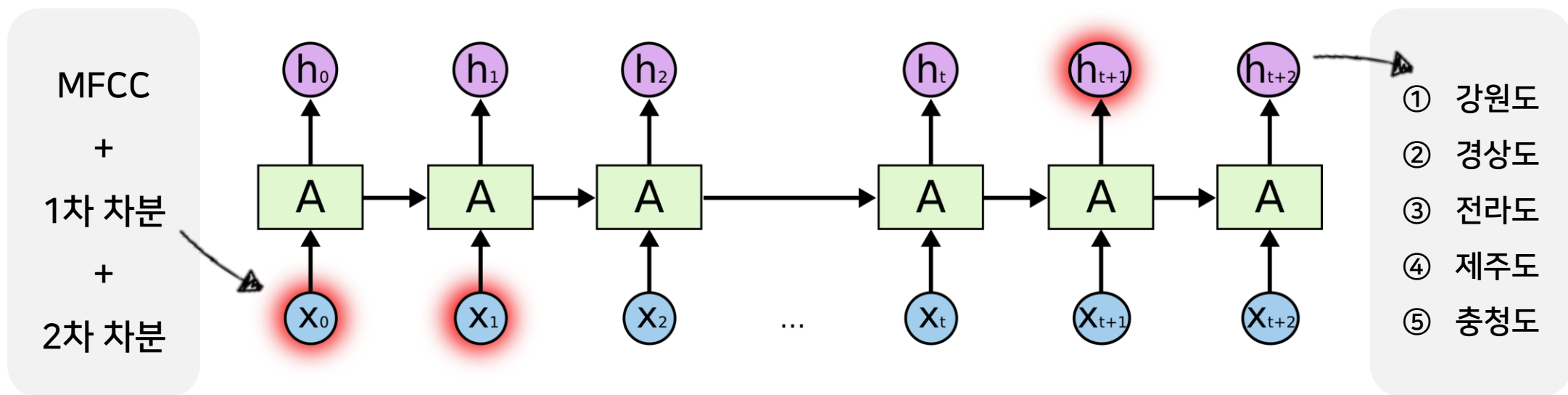


Max Pooling

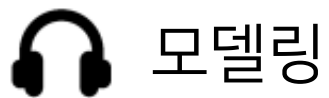
04 분석 과정



MFCC를 활용한 모델 - RNN



04 분석 과정



MFCC를 활용한 모델 - RNN

MFCC
+
1차 차분
+
2차 차분



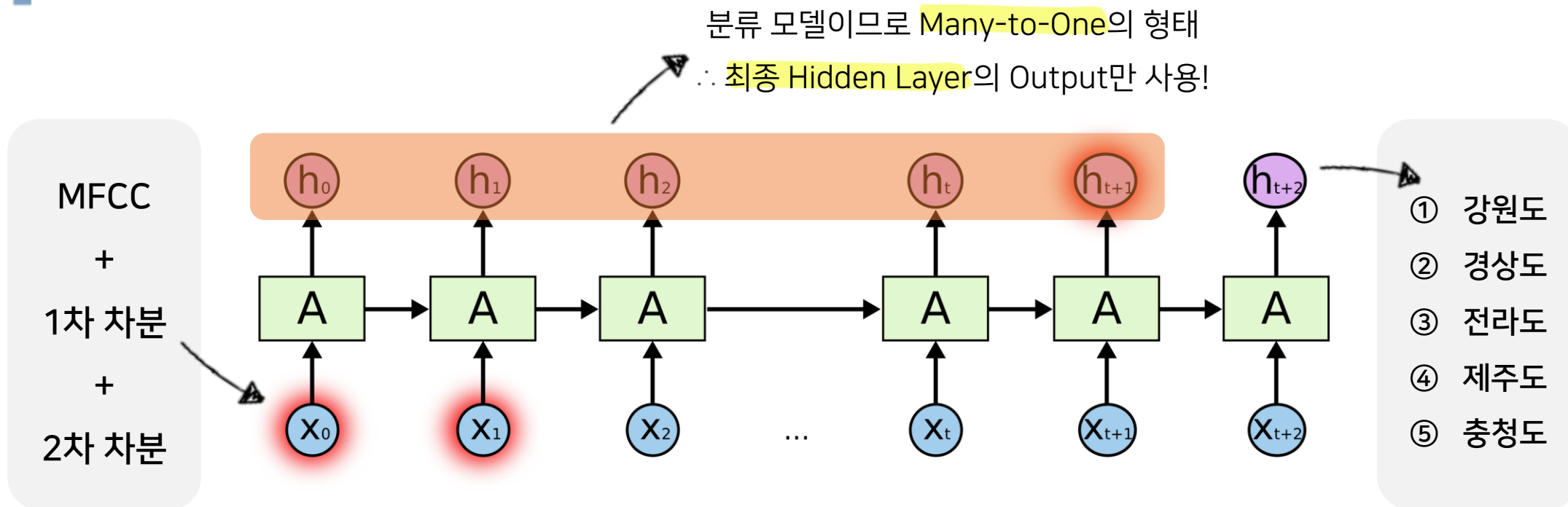
```
class ToMFCC2(object):  
    def __init__(self):  
        pass  
  
    def __call__(self, data):  
        signal = data['signal']  
        sr = data['sample_rate']  
  
        self.n_fft = int(np.ceil(0.025 * sr))  
        self.win_length = int(np.ceil(0.025 * sr))  
        self.hop_length = int(np.ceil(0.01 * sr))  
  
        audio_mfcc = librosa.feature.mfcc(y=signal.numpy().reshape(-1),  
                                          sr=sr,  
                                          n_mfcc=13,  
                                          n_fft=self.n_fft,  
                                          hop_length=self.hop_length).transpose()  
  
        delta1 = librosa.feature.delta(audio_mfcc).transpose()  
        delta2 = librosa.feature.delta(audio_mfcc, order=2).transpose()  
        mfcc_result = np.concatenate((audio_mfcc, delta1, delta2), axis=1)  
  
        mfcc_result = torch.from_numpy(mfcc_result)  
        mfcc_result = mfcc_pad2(mfcc_result)  
  
        data['MFCC'] = mfcc_result  
        data['input'] = mfcc_result  
  
    return data
```

[126000, 39]의
tensor 준비!

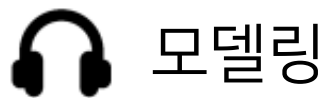
04 분석 과정

모델링

MFCC를 활용한 모델 - RNN

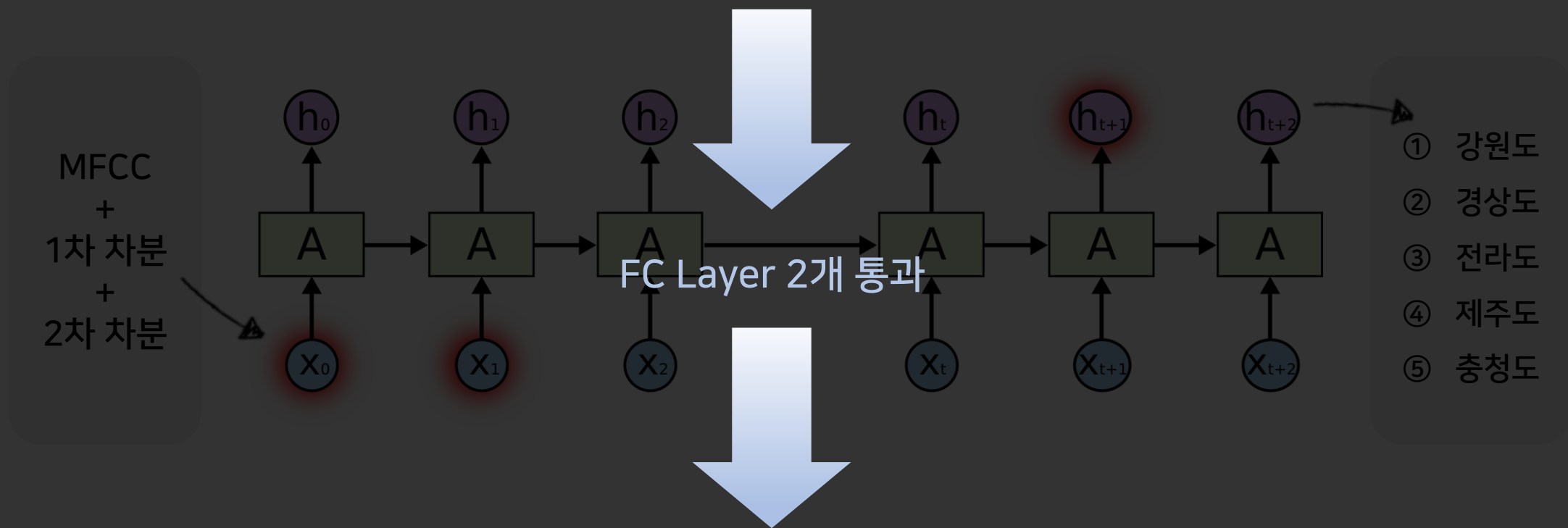


04 분석 과정



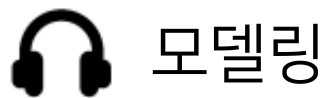
MFCC를 활용한 모델 - RNN

Layer 2개 + Hidden Size 16



CNN과 동일한 Output = [batch_size, 5]

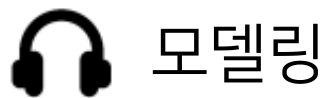
04 분석 과정



모델 개요

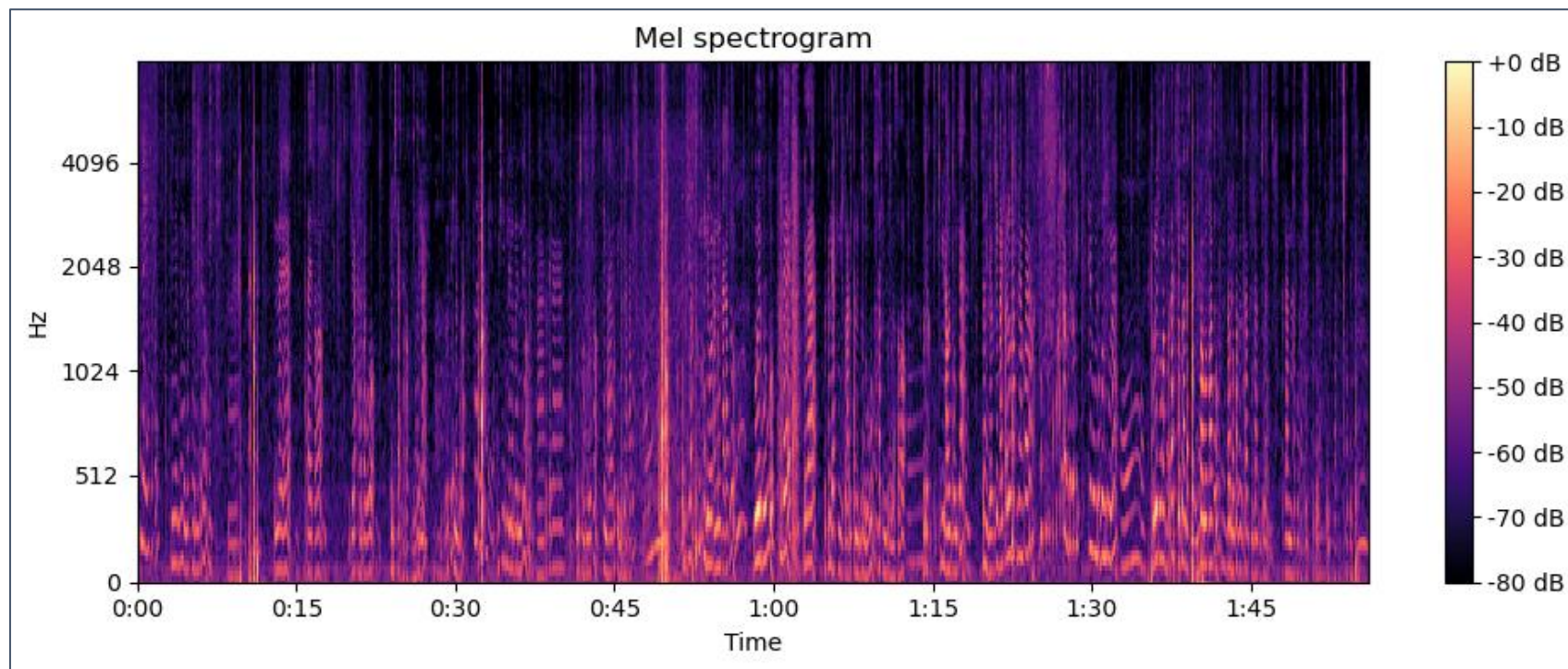
모델 유형	Input 유형	Input Size	모델 구성
CNN (Convolutional NN)	MFCC	<code>torch.size([3, 13, 126000])</code>	[Conv * Pool] Layer 2개 + FC Layer 2개
Vanilla RNN (Recurrent NN)	MFCC	<code>torch.size([126000, 39])</code>	RNN Hidden Layer 2개 + Sequence 길이 126000
CNN	Mel Spectrogram	<code>torch.size([2, 2000, 2000])</code>	[Conv * BN * Pool] Layer 4개 + FC Layer 2개
Vanilla RNN	Mel Spectrogram	<code>torch.size([5000, 256])</code>	RNN Hidden Layer 2개 + Sequence 길이 5000

04 분석 과정

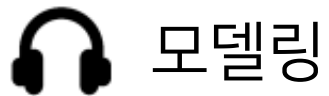


Mel Spectrogram을 활용한 모델 - CNN

Mel
Spectrogram
+
1차 차분



04 분석 과정



Mel Spectrogram을 활용한 모델 - CNN

Mel
Spectrogram
+
1차 차분



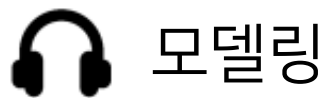
```
class MelSpec1(object):  
    def __init__(self):  
        pass  
  
    def __call__(self, data):  
        signal = data["signal"]  
        sr = data['sample_rate']  
  
        self.n_fft = 20480 # int(np.ceil(0.025 * sr))  
        self.win_length = 20480 # int(np.ceil(0.025 * sr))  
        self.hop_length = 5120 # int(np.ceil(0.01 * sr))  
  
        # MelSpectrogram --> tensor 반환  
        mel_spec = nn.Sequential(T.MelSpectrogram(  
            sample_rate=sr,  
            n_fft=self.n_fft,  
            win_length=self.win_length,  
            hop_length=self.hop_length,  
            n_mels=2000),  
            T.AmplitudeToDB())  
  
        # 차분값 계산  
        mel_delta = T.ComputeDeltas(win_length=self.win_length)  
        mel_out = mel_spec(signal)  
        mel_delta = mel_delta(mel_out)  
  
        output = torch.stack([mel_out[0], mel_delta[0]])  
        output = mel_pad1(output) # [2, 80, 80]  
  
        data['MelSpectrogram'] = output  
        data['input'] = output  
  
        return data
```

feature.melspectrogram 함수



Mel Spectrogram 추출

04 분석 과정



Mel Spectrogram을 활용한 모델 - CNN

Mel
Spectrogram
+
1차 차분



```
class MelSpec1(object):  
    def __init__(self):  
        pass  
  
    def __call__(self, data):  
        signal = data["signal"]  
        sr = data['sample_rate']  
  
        self.n_fft = 20480 # int(np.ceil(0.025 * sr))  
        self.win_length = 20480 # int(np.ceil(0.025 * sr))  
        self.hop_length = 5120 # int(np.ceil(0.01 * sr))  
  
        # MelSpectrogram --> tensor 반환  
        mel_spec = nn.Sequential(T.MelSpectrogram(  
            sample_rate=sr,  
            n_fft=self.n_fft,  
            win_length=self.win_length,  
            hop_length=self.hop_length,  
            n_mels=2000),  
            T.AmplitudeToDB())  
  
        # 차분값 계산  
        mel_delta = T.ComputeDeltas(win_length=self.win_length)  
        mel_out = mel_spec(signal)  
        mel_delta = mel_delta(mel_out)  
  
        output = torch.stack([mel_out[0], mel_delta[0]])  
        output = mel_pad1(output) # [2, 80, 80]  
  
        data['MelSpectrogram'] = output  
        data['input'] = output  
  
        return data
```

feature.delta 함수



1차 차분 추출

04 분석 과정



Mel Spectrogram을 활용한 모델 - CNN

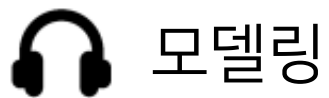
Mel
Spectrogram
+
1차 차분



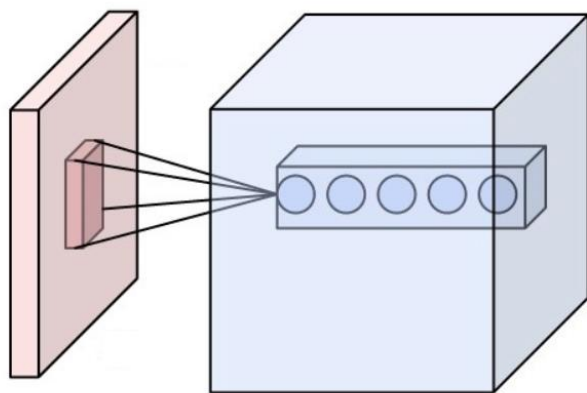
```
class MelSpec1(object):  
    def __init__(self):  
        pass  
  
    def __call__(self, data):  
        signal = data["signal"]  
        sr = data['sample_rate']  
  
        self.n_fft = 20480 # int(np.ceil(0.025 * sr))  
        self.win_length = 20480 # int(np.ceil(0.025 * sr))  
        self.hop_length = 5120 # int(np.ceil(0.01 * sr))  
  
        # MelSpectrogram --> tensor 반환  
        mel_spec = nn.Sequential(T.MelSpectrogram(  
            sample_rate=sr,  
            n_fft=self.n_fft,  
            win_length=self.win_length,  
            hop_length=self.hop_length,  
            n_mels=2000),  
            T.AmplitudeToDB())  
  
        # 차분값 계산  
        mel_delta = T.ComputeDeltas(win_length=self.win_length)  
        mel_out = mel_spec(signal)  
        mel_delta = mel_delta(mel_out)  
  
        output = torch.stack([mel_out[0], mel_delta[0]])  
        output = mel_pad1(output) # [2, 80, 80]  
  
        data['MelSpectrogram'] = output  
        data['input'] = output  
  
        return data
```

[2, 2000, 2000]의
tensor 준비!

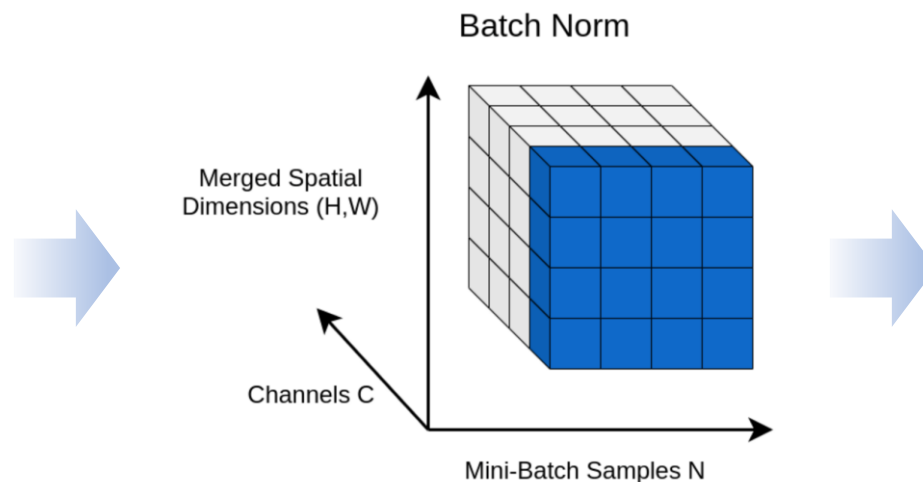
04 분석 과정



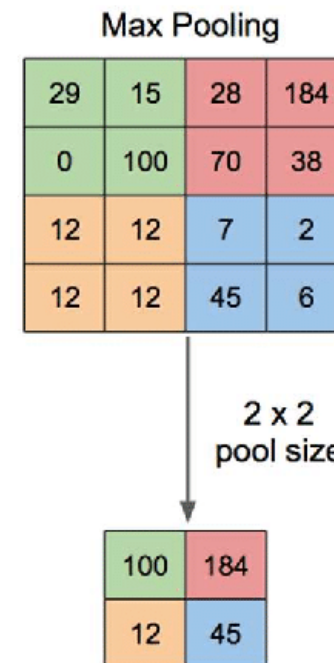
Mel Spectrogram을 활용한 모델 - CNN



Convolutional Layer

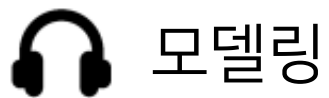


Batch Normalization



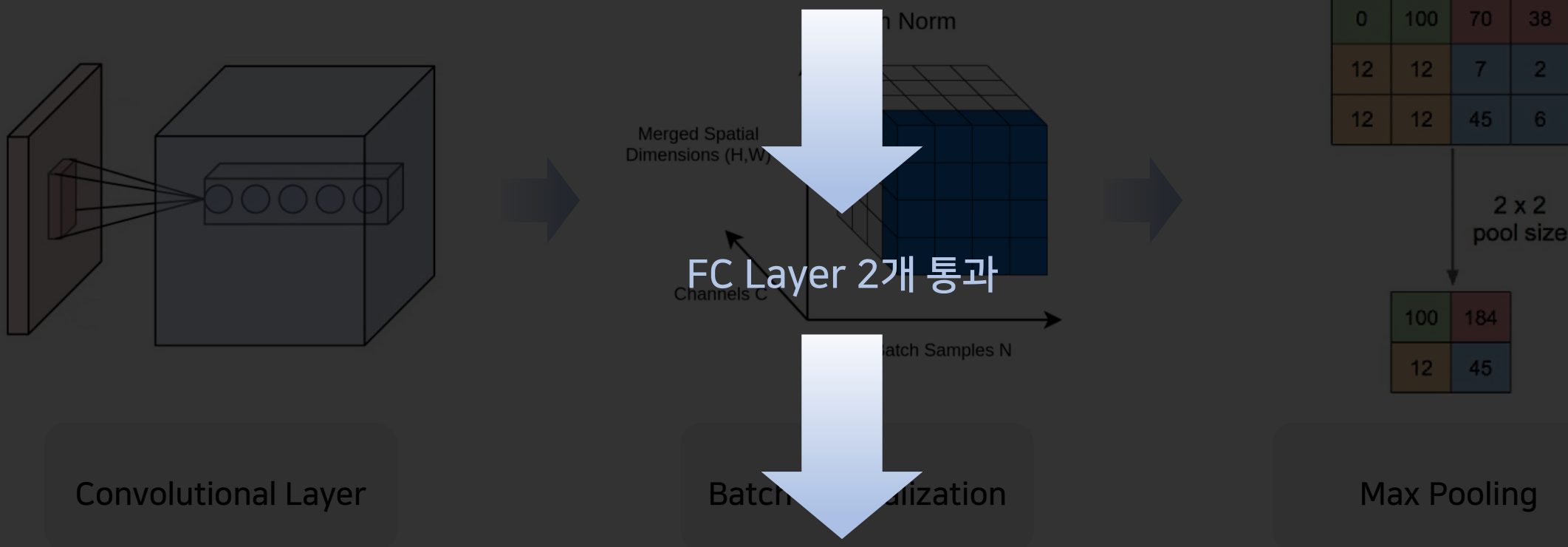
Max Pooling

04 분석 과정



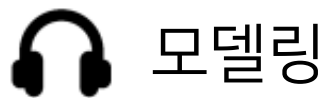
Mel Spectrogram을 활용한 모델 - CNN

[Conv * BN * Pool] Layer 4개

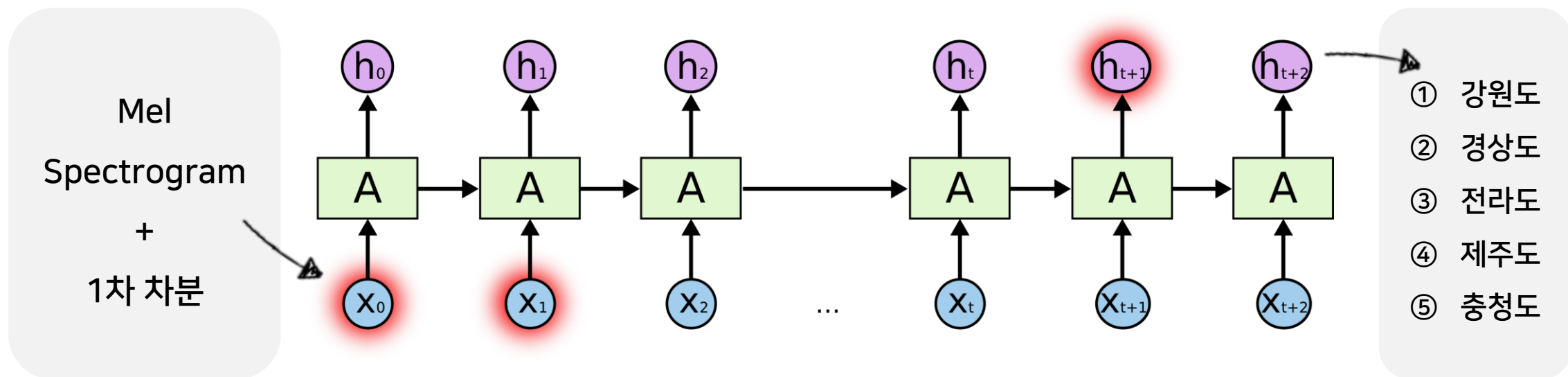


MFCC CNN과 동일한 Output = [batch_size, 5]

04 분석 과정



Mel Spectrogram을 활용한 모델 - RNN



04 분석 과정



Mel Spectrogram을 활용한 모델 - RNN

Mel
Spectrogram
+
1차 차분



```
class MelSpec2(object):  
    def __init__(self):  
        pass  
  
    def __call__(self, data):  
        signal = data["signal"]  
        sr = data['sample_rate']  
  
        mel_input = signal[0].numpy()  
  
        mel_spec = librosa.power_to_db(  
            librosa.feature.melspectrogram(y=signal[0].numpy(),  
                                           sr=sr, n_mels=128,  
                                           fmax=8000, fmin=100)[: , 1000:6000],  
            ref=np.max) # [128, 5000]  
  
        mel_delta = librosa.feature.delta(mel_spec)  
  
        output = torch.Tensor(np.concatenate([mel_spec, mel_delta], axis=0)).transpose(0, 1)  
        # torch.Size([5000, 256])  
  
        data['MelSpectrogram'] = output  
        data['input'] = output  
  
        return data
```

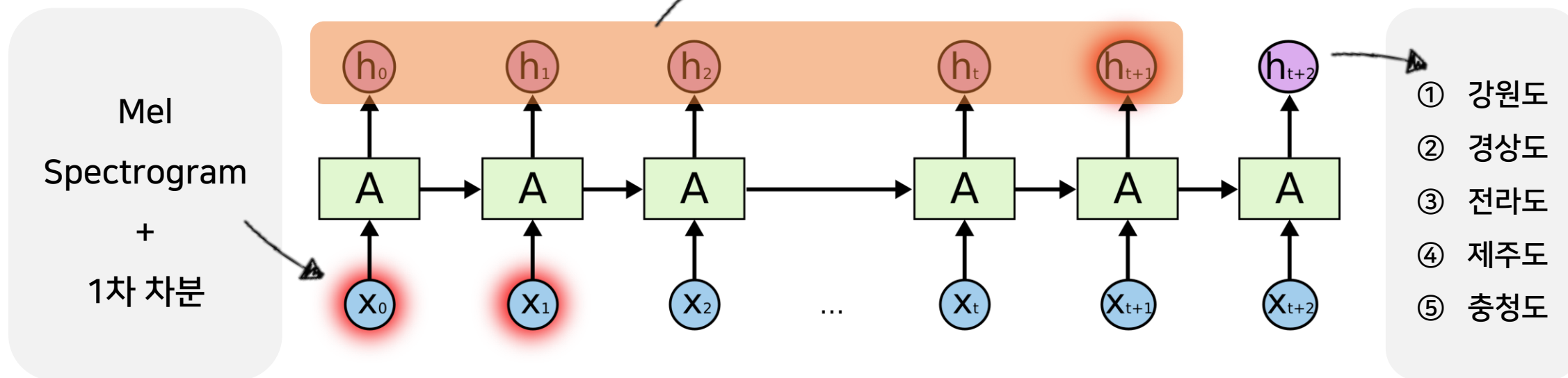
[5000, 256]의
tensor 준비!

04 분석 과정

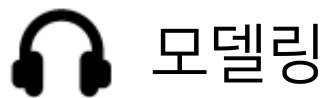
모델링

Mel Spectrogram을 활용한 모델 - RNN

분류 모델이므로 Many-to-One의 형태
∴ 최종 Hidden Layer의 Output만 사용!

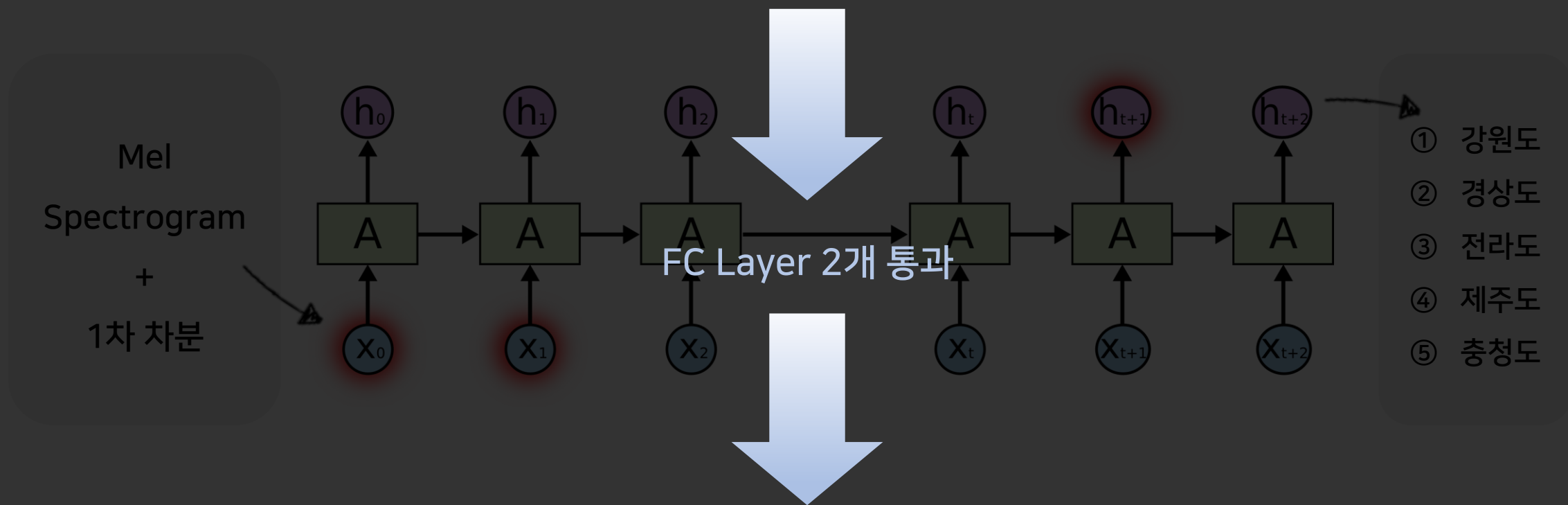


04 분석 과정



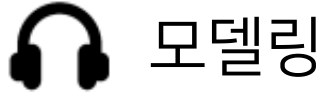
Mel Spectrogram을 활용한 모델 - RNN

Layer 2개 + Hidden Size 64



CNN과 동일한 Output = [batch_size, 5]

04 분석 과정



모델 성능 평가

모델의 유형과 무관하게 학습이 제대로 이루어지지 않음

모델 유형	모델 성능 평가	문제점
CNN (Convolutional NN)	<div>① Loss (Cross Entropy Loss)<ul style="list-style-type: none">▪ Learning Rate: 5e-6~1e-9 적용▪ Batch Size: 10~25 적용▪ Epochs: 1~20 적용▪ 1.5000 ~ 1.7000 사이 진동</div> <div>② Train Accuracy<ul style="list-style-type: none">▪ Sample: 100개~1000개 사용▪ 20% ~ 40% 사이 진동</div>	<div>① Loss 수렴하지 않음</div> <div>② Train Set에 대한 정확도 낮음</div> <div>↓</div> <div>데이터 전처리 방법 근본적인 개선 필요!</div>
Vanilla RNN (Recurrent NN)		
CNN		
Vanilla RNN		

05

교수님 피드백



05 교수님 피드백



데이터관련

한 사람 당 다수의 음성 파일이 있는데 과적합이 발생하지 않을까?



10초 정도의 짧은 분량이라면 같은 화자가 큰 문제 **X**
단, training과 validation data에 같은 화자가 나타나지 않도록 주의



Training data와 Validation data 확인 후 처리 작업



05 교수님 피드백



데이터관련

데이터가 너무 길어서 학습이 원활하지 않는데 잘라서 사용해도 될까요?



잘라서 사용해도 되지만 길이가 같게 해주어야 함
padding을 넣을 때는 EOS 토큰을 꼭 넣어주어야
모델이 padding을 데이터로 인식 **X**



데이터를 일정하게 잘라서 사용
padding 코드 다시 점검



05 교수님 피드백



데이터관련

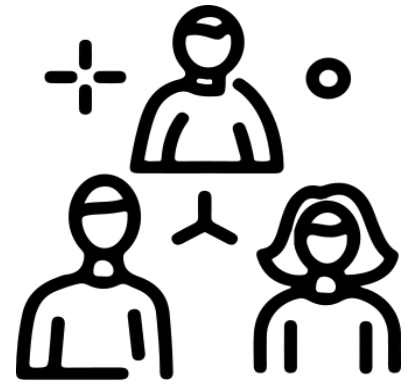
음성 파일에 배경 소음이 존재하는데 지워야 할까요?



배경 소음의 유무는 데이터 자체에 큰 결함 X
배경 소음이 있는 데이터와 없는 데이터의 학습 결과를 비교하는 것 추천



배경 소음을 지웠을 때와 지우지 않았을 때 모델의 성능 차이 확인하기



05 교수님 피드백

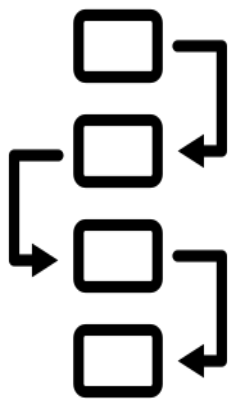


모델링관련

순차적으로 분류를 수행하는 것이 합리적인 방법일까요?

아이디어 자체는 좋으니

성별/ 연령/ 지역 분류의 baseline 모델을 각각 확보 후
순차적으로 분류했을 때의 결과와 비교하는 것이 바람직



독립적으로 분류하는 모델과 순차적으로 분류하는 모델 성능 차이 확인하기



감사합니다

