# <학습데이터>

## 1. 데이터 구축 설명서

### 1.1 데이터 수집 방법

#### 데이터 출처

* 공개 데이터 셋.
* **AI HUB**에서 제공하는 한국인 안면 이미지 공개 데이터 셋.
* 한국인 안면 이미지 URL : https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=83\n

#### 데이터 수집 주기

* 데이터 수집은 한번 진행 되었으며, 이후 추가적인 데이터 수집은 없음.

### 1.2 데이터 전처리 과정

#### 데이터 정제

* 이미지 정보
  + 저, 중, 고화질 이미지가 각각 동일 인물 400명의 데이터 셋.
  + 얼굴 각도 20종，조명 30종，액세서리 6종，표정 3종, 해상도 3종으로 **1명당 약 32,400장**의 데이터 셋으로 구축되어 있음.
  + 해상도 : 고해상도(864x576)，중해상도(346x230)，저해상도 (173x115)
* 이미지 분류
  + 400명의 class가 있고 1 class당 32,400개 사진 중 1,700장의 사진을 랜덤 추출
  + 1 class당 1,700장의 사진을 추출한 후 train : valid : test = 6 : 2 : 2 비율로 분할 (대략 1000 : 350 : 350)
* 중복데이터 처리
  + 한 ID값을 가진 이미지 화질이 저, 중, 고화질로 나뉘어져 있어 **중복 처리는 시행하지 않음**.

### 1.3 데이터 변환

* 이미지 크기 조정
  + 모든 이미지를 size를 160x160 으로 조정.
  + 모든 이미지를 파이토치 tensor로 변환함. **(Channel, Height, Width)** 형태의 텐서로 변환.
  + 모든 이미지를 평균 = [0.5, 0.5, 0.5]와 표준편차 = [0.5, 0.5, 0.5]로 정규화. (**픽셀값 => [-1,1]사이**)
* 라벨링
  + 처음 압축 해제할 때 가지고 있던 고유 ID값을 라벨로 붙여서 사용.

### 1.4 데이터 구조

* 디렉터리 구조
* modeling  
  ├─ .ipynb\_checkpoints  
  │ ├─ datacleansing-checkpoint.\*\*ipynb\*\*  
  │ └─ 데이터정제보고서-checkpoint.md  
  ├─ datacleansing.ipynb  
  ├─ datasets  
  ├─ facenet\_pytorch.ipynb  
  ├─ low\_data  
  ├─ middle\_data  
  ├─ high\_data  
  ├─ test  
  │ ├─ 19070922  
  │ │ ├─ 19070922\_S001\_L1\_E01\_C1.jpg  
  │ │ ├─ 19070922\_S001\_L1\_E01\_C2.jpg  
  │ ├─ 19071032  
  │ │ ├─ 19071032\_S003\_L8\_E02\_C1.jpg  
  │ │ ├─ 19071032\_S003\_L8\_E02\_C2.jpg  
  ├─ train  
  │ ├─ 19070922  
  │ │ ├─ 19070922\_S001\_L1\_E01\_C1.jpg  
  │ │ ├─ 19070922\_S001\_L1\_E01\_C2.jpg  
  │ ├─ 19071032  
  │ │ ├─ 19071032\_S003\_L8\_E02\_C1.jpg  
  │ │ ├─ 19071032\_S003\_L8\_E02\_C2.jpg  
  ├─ valid  
  │ ├─ 19070922  
  │ │ ├─ 19070922\_S001\_L1\_E01\_C1.jpg  
  │ │ ├─ 19070922\_S001\_L1\_E01\_C2.jpg  
  │ ├─ 19071032  
  │ │ ├─ 19071032\_S003\_L8\_E02\_C1.jpg  
  │ │ ├─ 19071032\_S003\_L8\_E02\_C2.jpg  
  └─ 데이터 정제 보고서.md
* 파일명 규칙
  + 파일명은 '**고유ID값\_액세서리\_조명\_표정\_인덱스.jpg**' 형식으로 구성되어 있으며 '*19070922\_S001\_L1\_E01\_C1.jpg*' 는 **'19070922' ID값**을 가진 사람의 첫번째 얼굴 이미지.
  + test, train, valid 폴더 안의 파일은 이름만 같고 다른 이미지로 구성.

#### 데이터 저장 형식

* 이미지 형식
  + 모든 이미지는 JPG 형식

#### 데이터 개인정보 보호

* 데이터는 익명화된 형태로 저장됨.
* 출처 : AI HUB , 한국과학기술연구원 , 한국지능정보사회진흥원

#### 데이터 수집 및 전처리 과정에 사용된 모든 코드는 GitHub 리포지토리에 공개하였으며, README 파일에 상세한 설명을 포함했습니다.

## 2. 학습 환경 설명서

### 2.2 학습 환경 개요

#### 목적

* + AI 모델을 학습시키기 위해 사용된 하드웨어 , 소프트웨어 환경의 정보를 제공

#### 하드웨어

* + 데스크탑 : CUDA를 지원하는 NVIDA GPU가 장착된 데스크탑 환경.
    - GPU 모델 : RTX 2070 SUPER , RTX 2080 SUPER
    - CPU : Intel i5-10600 , AMD RYZEN 5 3600
    - RAM : 16GB , 16GB
  + Macbook : Apple Silicon M2 칩을 탑재한 MacBook 환경.
    - GPU : 10코어
    - CPU : 8코어
    - RAM : 16GB

#### 소프트웨어

* + 운영체제
    - 데스크탑 : Windows 11
    - MacBook : mac os somona 14.5
  + python version
    - 3.10.4 , 3.11.7

### 환경설정

#### 데스크탑(CUDA사용)

* + CUDA 및 드라이버 설치
    - CUDA 설치: [NVIDIA CUDA Toolkit 설치 가이드](https://developer.nvidia.com/cuda-downloads)에 따라 CUDA Toolkit 설치.
    - 설치 확인: nvcc --version 명령어로 CUDA 설치 확인.
  + python 가상환경 설정
* # 데스크탑  
  python -m venv env  
  env\Scripts\activate  
    
  # Mac  
  python -m venv env  
  source env/bin/activate
  + PyTorch 설치
* # 데스크탑 CUDA Version ==12.1  
  pip install torch==2.3.0 torchvision==0.18.0 torchaudio==2.3.0 --index-url https://download.pytorch.org/whl/cu121  
    
  # Mac  
  pip install torch torchvision torchaudio
  + 테스트
* # 데스크탑  
  import torch  
  device = 'cuda' if torch.cuda.is\_available() else 'cpu'  
  print(device) # "cuda" 출력확인  
    
  # Mac  
  import torch  
  device = "mps" if torch.backends.mps.is\_available() else "cpu"  
  print(device) # "mps" 출력확인

#### 모델 학습 중간 결과는 주기적으로 saved\_model 폴더에 저장

#### Google Colab

* + A100 GPU
    - .

### 2.2 AWS - Elastic Beanstalk

1. 콘솔 홈에서 Elastic Beanstalk을 검색하여 로그인을 하면 콘솔 홈으로 넘어감. 그 후 상단 검색창에 Elastic Beanstalk을 검색
2. 좌측 목록아이콘 클릭, 애플리케이션 클릭, 애플리케이션 생성 클릭
   * 애플리케이션과 환경의 관계는 하나의 애플리게이션 아래에 여러 환경이 존재할 수 있음.
   * 애플리케이션을 먼저 생성한 후 환경을 만들어야 함
3. 애플리케이션 이름을 입력.
   * 설명 및 태그는 생략 가능
4. 생성후 자동으로 환경 설정화면으로 넘어감.
   * 우측 상단 새 환경 생성 클릭
5. 환경구성
   * 환경 티어: 웹 서버 환경 선택.
   * 애플리케이션 정보: 설정한 앱 이름이 입력되어있으면 태그 추가 필요 X
   * 환경 정보: 작성하지 않고 있는 그대로 두어도 무방.
   * 플랫폼 유형 : 관리형 플랫폼 선택
   * 플랫폼 : 본인이 사용하는 언어환경 선택
   * 플랫폼 브랜치, 버전 : 제약사항이 없는 이상 최신버전 선택
   * 애플리케이션 코드 : 바로 소스코드를 올릴 수 있지만, 올리기 전에 환경부터 만들기 -> 샘플 애플리케이션 선택
   * 프리셋 : 프로젝트 규모가 크지 않은 이상 다일 인스턴스 선택
     + 여러개의 컴퓨터가 병렬적으로 작동하도록 하고 싶다면 고가용성 선택
6. 서비스 액세스 구성
   * 서비스 역할 : 기존 서비스 역할 사용
   * 기존 서비스 역할 : 초기값 그대로 사용

#### EC2 keypair, EC2 인스턴스 프로파일

* 처음 앱을 만들 경우에 아무 것도 없을 수 있음.  
  각각 생성 해주어야함

1. 네트워킹, 데이터베이스 및 태그 설정
   * 선택 사항
2. 인스턴스 트래픽 및 조정 구성
   * 인스턴스 : 그대로 두어도 무방
   * 용량 : 아래 항목만 수정.
     1. 환경 유형 => 단일 인스턴스
     2. 플릿 구성 => 온디맨드 인스턴스
     3. 인스턴스 => 기본적으로 세팅되어 있는 것들을 X를 눌러 삭제 후 t2..nano를 선택(제일 저렴함)
3. 업데이트, 모니터링 및 로깅 구성
   * 선택 사항
4. 검토
   * 이상 없으면 제출
5. 코드 업로드
   * 생성 후 도메인 클릭하면 기본 화면이 출력됨
   * 환경 우측 상단의 업로드 및 배포 클릭
   * node 서버폴더를 폴더째로 zip 확장자로 압축 후 업로드(참고자료 사용 언어)

#### 출처

https://velog.io/@bcl0206/AWS-Elastic-Beanstalk-배포-방법-DB-연결