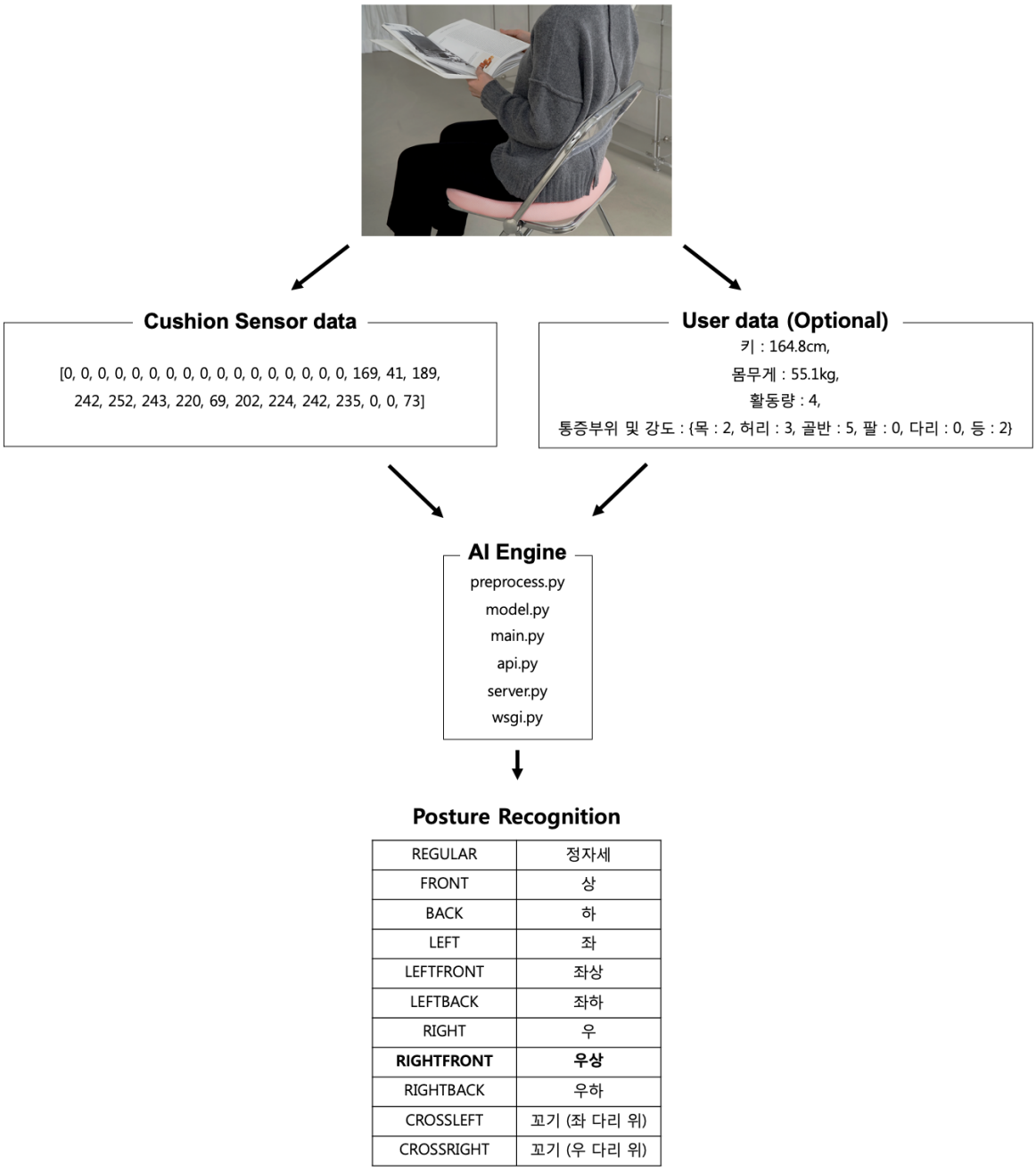
HCLab 앉은 자세 예측 AI 엔진 가이드

0. 엔진 개요

- 스마트 방석의 센서 값을 바탕으로 앉은 자세를 예측하기 위한 AI 엔진
- 엔진 Architecture



REGULAR	정자세
FRONT	상
BACK	하
LEFT	좌
LEFTFRONT	좌상
LEFTBACK	좌하
RIGHT	우
RIGHTFRONT	**우상**
RIGHTBACK	우하
CROSSLEFT	꼬기 (좌 다리 위)
CROSSRIGHT	꼬기 (우 다리 위)

1. 파일 구성

- hclab-smart-cushion-ai-api 폴더는 아래와 같이 구성

```
hclab-smart-cushion-ai-api/
    - README.md
    - conf/
    - logs/
    - run/
```

```
    - test/
    - Makefile [API 서버 실행에 필요한 명령어 집합]
    - requirements.txt [필요 패키지 리스트]
    - resources/ [모델 파일 위치]
    - datasets/
        - sensor_data.csv [Timestamp, Sensor Data, Label로 정의된 데이터]
        - user_data.csv [Timestamp, 키, 몸무게, 통증부위 및 강도, 활동량, 사용 목적으로 정의된 데
이터]
    - src/
        - README.md
        - preprocess.py [데이터 전처리 코드]
        - model.py [모델 학습 코드]
        - main.py [유저와 상호 작용하는 코드]
        - api.py [API 서버 관련 코드]
        - server.py [API 서버 관련 코드]
        - wsgi.py [API 서버 관련 코드]
```

* API 서버 관련 코드는 하단에 설명을 기재하지 않음

sensor_data.csv

- 각각의 데이터는 데이터가 관측된 시각(Timestamp), Sensor Data, Label로 구성됨.

Timestamp	Sensor Data	Label
10/15/2020 11:28:04 AM +09:00	0,198,218,61,236,226,202,155,138,239,245,236,195,102,205,0,0,153,50,160,236,235,223,33,0,53,148,192,54,52,0,0	CROSSRIGHT
10/15/2020 11:28:01 AM +09:00	0,198,218,61,236,227,202,155,138,239,245,236,195,102,205,0,0,152,50,161,236,235,223,33,0,54,148,193,52,53,0,0	CROSSRIGHT
10/15/2020 11:27:58 AM +09:00	0,198,218,62,236,227,202,155,139,240,245,237,195,102,205,0,0,151,48,161,236,235,222,31,0,51,148,192,52,51,0,0	CROSSRIGHT
10/15/2020 11:27:55 AM +09:00	0,198,218,62,236,227,202,155,138,239,245,236,195,102,205,0,0,149,50,160,235,235,222,32,0,51,149,193,54,52,0,0	CROSSRIGHT
10/15/2020 11:27:52 AM +09:00	0,198,218,61,236,227,202,155,137,239,245,236,195,102,205,0,0,151,49,159,236,235,223,33,0,53,148,192,53,53,0,0	CROSSRIGHT
10/15/2020 11:27:49 AM +09:00	0,198,218,63,236,226,202,155,138,239,245,236,195,102,205,0,0,149,49,157,235,235,222,33,0,53,145,192,52,53,0,0	CROSSRIGHT
10/15/2020 11:27:46 AM +09:00	0,198,218,63,236,227,202,155,138,240,245,237,194,102,203,0,0,139,45,156,235,234,221,0,0,51,146,192,54,52,0,0	CROSSRIGHT
10/15/2020 11:27:43 AM +09:00	0,198,218,56,237,226,201,154,137,239,245,237,195,97,205,0,0,148,50,160,234,235,222,32,0,51,143,192,52,52,0,0	CROSSRIGHT
10/15/2020 11:27:40 AM +09:00	0,198,217,55,236,226,201,154,137,239,245,236,194,97,204,0,0,148,50,160,235,235,223,31,0,51,144,193,54,52,0,0	CROSSRIGHT
10/15/2020 11:27:37 AM +09:00	0,198,217,55,236,226,201,155,137,239,245,236,194,97,204,0,0,148,52,160,234,235,223,31,0,51,144,193,54,52,0,0	CROSSRIGHT
10/15/2020 11:27:34 AM +09:00	0,198,218,58,237,226,202,155,138,239,245,236,195,100,204,0,0,148,48,160,234,235,222,31,0,50,143,192,52,51,0,0	CROSSRIGHT
10/15/2020 11:27:31 AM +09:00	0,197,217,61,236,227,201,155,137,239,245,236,195,97,204,0,0,149,47,160,235,235,223,33,0,53,143,192,52,53,0,0	CROSSRIGHT
10/15/2020 11:27:28 AM +09:00	0,197,217,63,236,227,202,155,137,239,245,236,194,96,204,0,0,148,49,160,235,235,222,31,0,51,144,192,55,54,0,0	CROSSRIGHT
10/15/2020 11:27:25 AM +09:00	0,197,217,63,236,227,202,155,137,239,245,236,194,97,204,0,0,148,49,160,235,235,223,32,0,51,144,192,55,53,0,0	CROSSRIGHT
10/15/2020 11:27:22 AM +09:00	0,197,217,63,236,227,202,157,137,239,245,237,195,97,204,0,0,149,47,161,235,235,223,34,0,53,143,192,53,55,0,0	CROSSRIGHT
10/15/2020 11:27:19 AM +09:00	0,198,218,64,236,227,202,157,137,239,245,237,195,97,204,0,0,149,48,161,236,235,223,35,0,53,141,192,58,55,0,0	CROSSRIGHT
10/15/2020 11:27:16 AM +09:00	0,197,218,69,236,227,202,159,137,240,247,237,196,97,204,0,0,149,50,161,236,235,224,36,0,52,142,193,58,58,0,0	CROSSRIGHT
10/15/2020 11:27:13 AM +09:00	0,198,218,90,237,228,204,161,140,239,248,237,200,102,199,0,0,147,50,158,235,236,223,36,0,48,148,193,60,56,0,0	CROSSRIGHT
10/15/2020 11:27:10 AM +09:00	0,197,221,71,237,229,205,160,137,239,248,239,194,88,204,0,0,150,49,165,236,235,226,38,0,53,148,194,60,58,0,0	CROSSRIGHT

user_data.csv

- 각각의 데이터는 데이터가 생성된 시각(Timestamp), 키, 몸무게, 통증부위 및 강도, 활동량, 사용 목적으로 구성됨.

수정일시	키	몸무게	통증부위 및 강도	활동량	사용 목적
2020-10-05 12:34:25	158	53	neck:4,shoulder:4,back:4,arms:2,waist:4,pelvis:4,legs:4	2	허리 통증 완화,자세 교정

preprocess.py

- 데이터 전처리에 필요한 코드가 정의되어 있는 파일
- 주요 함수는 아래와 같음

```
get_all_sensor_data_dir(data_dir)
: data_dir 내부에 존재하는 모든 xlsx 파일들의 경로를 리스트 형식으로 반환하는 함수
[파라미터]
data_dir(str) : xlsx 파일들을 모아놓은 폴더의 경로

[반환 값]
data_dir_list(list) : data_dir 내에 존재하는 xlsx 파일들의 경로가 포함된 리스트

[주의사항]
모든 xlsx 파일은 data_dir 바로 아래에 위치해야 함. 에를 들어 아래와 같은 형식의 구조는 가능하지만,
- data_dir/
  - tester1_sensor_data.xlsx
  - tester2_sensor_data.xlsx

아래와 같은 형식의 구조는 불가능함.
- data_dir/
  - data_sub_dir/
    - tester1_sensor_data.xlsx
    - tester2_sensor_data.xlsx

excel_to_pickle(data_base_dir, data_dir_list)
: data_dir_list에 존재하는 모든 xlsx 파일들을 읽어들여 전처리한 뒤 data_base_dir에 pickle 형태로
 저장하는 함수
[파라미터]
data_base_dir(str) : pickle을 저장할 경로
data_dir_list(str) : xlsx 파일들의 경로가 포함된 리스트

[반환 값]
pickle_dir(str) : 전처리된 pickle의 경로

get_all_sensor_and_user_data_dir(data_dir)
: data_dir 내부에 존재하는 csv 형식의 센서 데이터와 유저 데이터의 경로를 리스트 형식으로 반환하는 함수
[파라미터]
data_dir(str) : csv 파일들을 모아놓은 폴더의 경로

[반환 값]
data_dir_list(list[list]) : [센서 데이터 경로, 유저 데이터 경로]의 리스트

[주의사항]
유저별로 디렉토리를 아래와 같이 분리한 뒤 센서 데이터의 경우 raw_testerX.csv, 유저 데터의 경우 profile
_testerX.csv의 형식으로 csv 파일명을 지정해야 함. 에를 들어 아래와 같은 형식의 구조는 가능하지만,
- data_dir/
  - tester1/
    - raw_tester1.csv
    - profile_tester1.csv
  - tester2/
    - raw_tester2.csv
    - profile_tester2.csv

아래와 같은 형식의 구조는 불가능함.
- data_dir/
  - raw_tester1.csv
  - profile_tester1.csv
  - raw_tester2.csv
  - profile_tester2.csv
```

```
data_dir_list_to_pickle(data_base_dir, data_dir_list)
: data_dir_list에 존재하는 모든 csv 파일들을 읽어들여 전처리한 뒤 data_base_dir에 pickle 형태로 저장하는 함수
[파라미터]
data_base_dir(str) : pickle을 저장할 경로
data_dir_list(str) : [센서 데이터 경로, 유저 데이터 경로]의 리스트

[반환 값]
pickle_dir(str) : 전처리된 pickle의 경로
```

model.py

- 모델 학습에 필요한 코드가 정의되어 있는 파일
- PostureRecognizer 클래스로 정의되어 있으며 파라미터와 주요 함수는 아래와 같음

```
PostureRecognizer(batch_size, epochs, pickle_dir, output_dir, weight_dir)
[파라미터]
mode : 엔진을 실행할 때의 모드 "train", "test", "predict", "debug"가 존재
method : 학습 방법으로 "use_user_data", "sensor_data_only"가 존재
batch_size : 학습 과정 중 한 번에 네트워크에 넘겨 줄 데이터의 수
epochs : 전체 데이터에 대해 학습을 반복할 횟수
pickle_dir : 전처리된 pickle의 경로
output_dir : 학습된 모델이 저장되는 위치로 src 폴더 내부를 기준으로 했을 때 resources
             폴더의 경로
weight_dir : "test", "predict" 모드에서 이용할 모델의 경로

[함수]
train
: 모델을 학습시키는 함수
test
: 학습된 모델의 성능을 평가하는 함수
predict(sensor_data: np.array, user_data: np.array = None)
: 센서 데이터와 유저 데이터(optional)를 입력받아 예측된 자세의 라벨과 확률을 반환하는 함수
debug
: 모델의 architecture를 출력하는 함수
cnn_model
: 센서 데이터만을 이용하는 모델을 정의하고 반환하는 함수
sensor_with_user_data_model
: 센서 데이터와 유저 데이터를 이용하는 모델을 정의하고 반환하는 함수
generate_data(pickle_dir: str)
: pickle의 경로를 바탕으로 sensor data를 normalize, label을 categorical variable로 만들어 반환하는 함수
```

main.py

- 엔진 실행시 유저와 직접적으로 상호 작용하는 코드
- 주요 함수는 아래와 같음

```
main(mode, data_dir, output_dir, weight_dir, method, batch_size, epochs)
: 파라미터를 바탕으로 model.py에 존재하는 PostureRecognizer를 선언하고, data_dird에 존재하는 데이터
를 method에 맞게 전처리한 뒤 mode에 따라 PostureRecognizer 내부의 함수를 호출하는 함수
[파라미터]
mode(str) : 엔진을 실행할 때의 모드
  - "train" : 모델 학습, "test" : 모델 성능 평가, "debug": 모델 architecture 출력
data_dir(str) : csv 파일들이 존재하는 폴더의 경로
output_dir : 학습된 모델이 저장되는 위치로 src 폴더 내부를 기준으로 했을 때 resources
             폴더의 경로
weight_dir : "test", "predict" 모드에서 이용할 모델의 경로
method : 학습 방법
  - "use_user_data" : 센서 데이터만 이용, "sensor_data_only" : 센서 데이터와 유저 데이터
    를 이용
batch_size : 학습 과정 중 한 번에 네트워크에 넘겨 줄 데이터의 수
epochs : 전체 데이터에 대해 학습을 반복할 횟수
```

2. 실행 환경

- 본 코드는 python 3.8 에서 최적화되어 있음
- 본 코드를 수행하기 위해서는 다음과 같은 패키지가 설치되어야 함

```
[필요 패키지]
Flask==1.1.1
Flask-HTTPAuth==3.3.0
gunicorn==20.0.4

tensorflow
sklearn
numpy<1.19.0,>=1.16.0
scipy
matplotlib
pandas
datetime
xlrd==1.2.0
h5py==2.10.0
```

3. 실행 방법

Main 함수 호출

1. hclab-smart-cushion-ai-api 폴더를 준비함
2. 콘솔 창을 열고 위의 폴더로 이동함

```
cd path/to/hclab-smart-cushion-ai-api
```

3. Python과 필요 패키지들을 설치함 (macOS 기준)

```
[Python 설치]
brew install python3

[필요 패키지 설치]
pip3 install -r requirements.txt
```

4. src 폴더로 이동함

```
cd src
```

5. 아래의 방법으로 main 함수를 호출하여 원하는 실행 모드를 호출함

   5-1. 학습(train) 실행

   - 먼저 Timestamp, Sensor Data, Label이 포함된 csv 형태의 학습 데이터를 준비함.
   - 아래의 명령어를 콘솔 창에 입력하여 학습이 실행됨.

```
python3 main.py --mode train --data_dir [DATA_DIR] --output_dir [OUTPUT_DIR]
 --method [METHOD] --batch_size [BATCH_SIZE] --epochs [EPOCHS]
```

   - DATA_DIR : csv 혹은 xlsx 파일들이 저장된 폴더의 경로 (ex : ../datasets)
   - OUTPUT_DIR : 모델 저장 경로 (ex : ../resources)
   - METHOD : 학습 방법 (ex : use_user_data or sensor_data_only)
   - BATCH_SIZE : 학습 과정 중 한 번에 네트워크에 넘겨 줄 데이터의 수. (ex : 4)
   - EPOCHS : 전체 데이터에 대해 학습을 반복할 횟수. (ex : 30)
   - 입력 예시

```
python3 main.py --mode train --data_dir ../datasets --output_dir ../resou
rces --method use_user_data --batch_size 4 --epochs 30
```

- 학습이 완료된 후에는 OUTPUT_DIR에 학습된 모델들과 학습 결과 그래프가 생성됨.

5-2. 테스트(test) 실행

- 학습에 이용한 데이터와 메타데이터가 존재하는 상황에서 OUTPUT_DIR에 생성된 모델 중 테스트할 모델을 선택함.
- 아래의 명령어를 콘솔 창에 입력하여 테스트가 실행됨.

```
python3 main.py --mode test --data_dir [DATA_DIR] --weight_dir [WEIGHT_DIR] -
-method [METHOD] --batch_size [BATCH_SIZE]
```

  - DATA_DIR : csv 파일들이 저장된 폴더 경로 (ex : ../datasets)
  - WEIGHT_DIR : 모델 저장 경로 (ex : ../resources/weights_best.h5)
  - METHOD : 모델 학습시 이용했던 학습 방법 (ex : use_user data or sensor_data_only)
    - 주의사항 : WEIGHT_DIR을 학습시 이용했던 학습 방법과 METHOD가 일치해야 함.
  - BATCH_SIZE : 학습 과정 중 한 번에 네트워크에 넘겨 줄 데이터의 수. (ex : 32)
  - 입력 예시

```
python3 main.py --mode test --data_dir ../datasets --weight_dir ../resour
ces/best_use_user_data.h5 --method use_user_data --batch_size 32
```

5-3. 디버깅(debug) 실행

- 모델의 내부 구조를 출력할 수 있는 기능.
- 아래의 명령어를 콘솔 창에 입력하여 설명이 실행됨.

```
python3 main.py --mode debug --method [METHOD]
```

  - METHOD : 내부 구조 출력을 원하는 모델 (ex : use_user data or sensor_data_only)
  - 입력 예시

```
python3 main.py --mode debug --method use_user_data
```

Local API 호출

1. hclab-smart-cushion-ai-api 폴더를 준비함
2. 콘솔 창을 열고 위의 폴더로 이동함

```
cd path/to/hclab-smart-cushion-ai-api
```

3. Python, Anaconda, make와 필요 패키지들을 설치함 (macOS 기준)

```
[Python 설치]
brew install python3

[Anaconda 설치]
brew install --cask anaconda

[make 설치]
brew install make
```

4. Anaconda 환경을 설정해줌

```
make conda_setup
```

5. 환경 변수를 설정해줌

```
export HCLAB_SC_API_TOKEN=A0E1E013-A2FC-47F7-8751-A14E37C1C054
export HCLAB_SENSOR_MODEL=resources/best_sensor_data_only.h5
export HCLAB_USER_MODEL=resources/best_use_user_data.h5
export HCLAB_SC_LOG_FILE=logs/server.log
export PYTHONPATH=${PYTHONPATH}:${PWD}/src
```

6. Flask server를 실행시킴

```
make flask_run_server
```

7. 콘솔 창을 하나 더 열고 아래와 같이 API Request를 보냄

```
curl --location --request POST 'http://0.0.0.0:22038/analyze/' \
 --header 'Authorization: Token A0E1E013-A2FC-47F7-8751-A14E37C1C054' \
 --header 'Content-Type: application/json' \
 --data-raw '{
    "sensor_data": SENSOR_DATA,
    "user_data": USER_DATA
 }'
```

- 입력 예시(유저 정보와 존재할 때와 존재하지 않을 때 서로 다른 모델을 이용함)

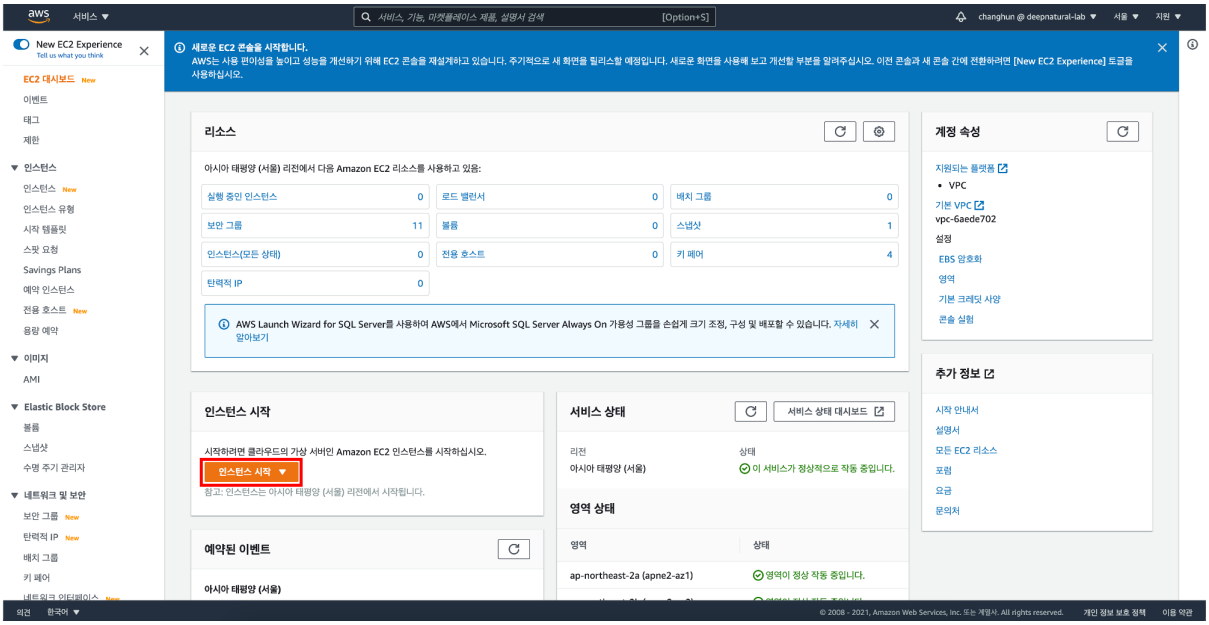
```
curl --location --request POST 'http://0.0.0.0:22038/analyze/' \
 --header 'Authorization: Token A0E1E013-A2FC-47F7-8751-A14E37C1C054' \
 --header 'Content-Type: application/json' \
 --data-raw '{
   "sensor_data": [0,0,0,48,216,225,165,0,0,222,243,242,174,88,177,0,59,199,1
39,178,217,231,223,121,81,157,214,219,160,0,0,0],
   "user_data": {
      "height": 169.1,
      "weight": 50.6,
      "amount_of_activity": 1,
      "pain_info": {
            "neck": 3,
            "waist": 3
      }
   }
}'
```

```
curl --location --request POST 'http://0.0.0.0:22038/analyze/' \
 --header 'Authorization: Token A0E1E013-A2FC-47F7-8751-A14E37C1C054' \
 --header 'Content-Type: application/json' \
 --data-raw '{
    "sensor_data": [0,0,0,48,216,225,165,0,0,222,243,242,174,88,177,0,59,199,
139,178,217,231,223,121,81,157,214,219,160,0,0,0]
}'
```

- 주의사항 : user_data에서 height, weight, amount_of_activity, pain_info 중 누락된 데이터가 있을 경우 sensor_data만을 이용하는 모델을 호출함.

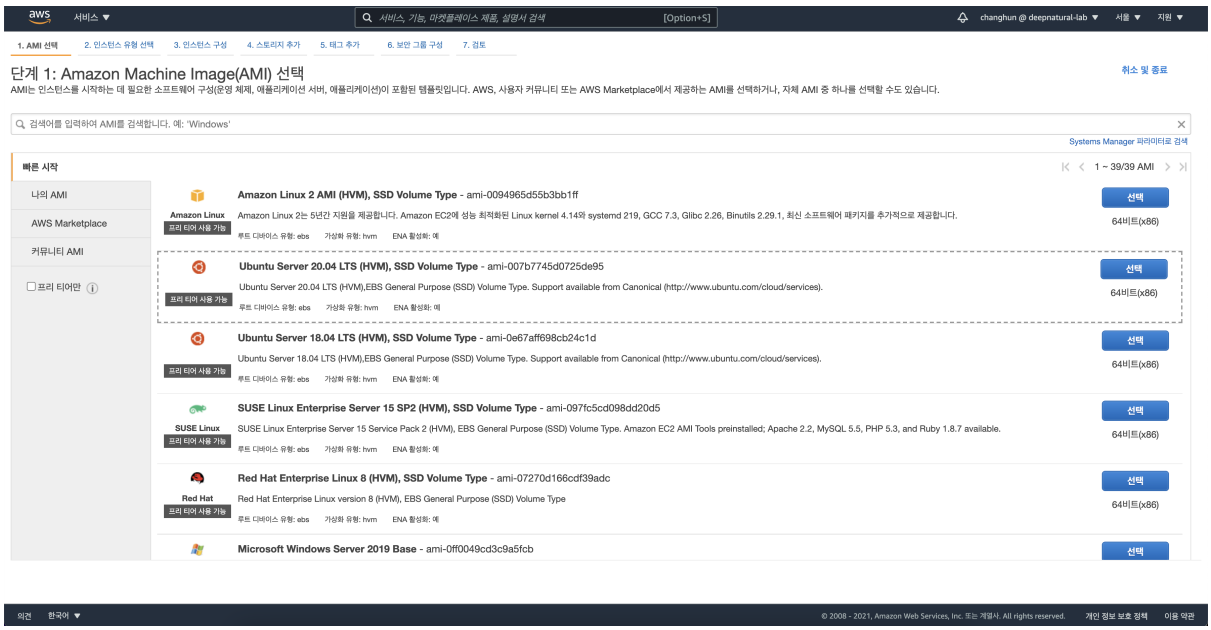
EC2 Instance Server API 호출

1. AWS 가입, 로그인 및 EC2 인스턴스 시작 클릭



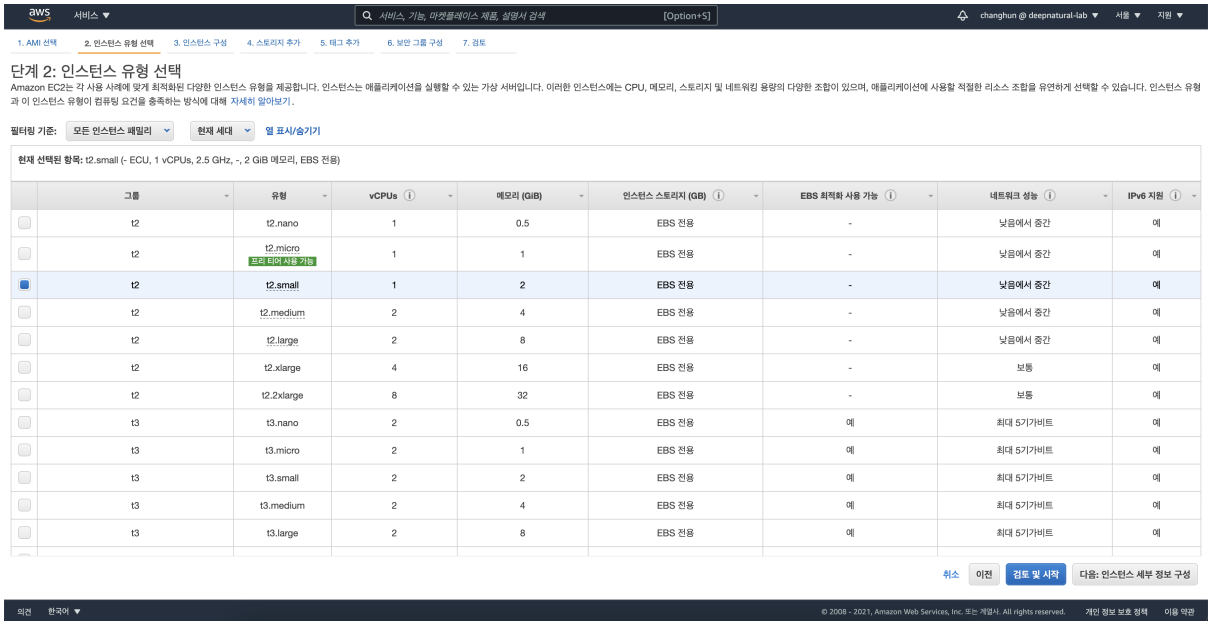
2. AMI 선택

- Ubuntu 20.04 Image 선택



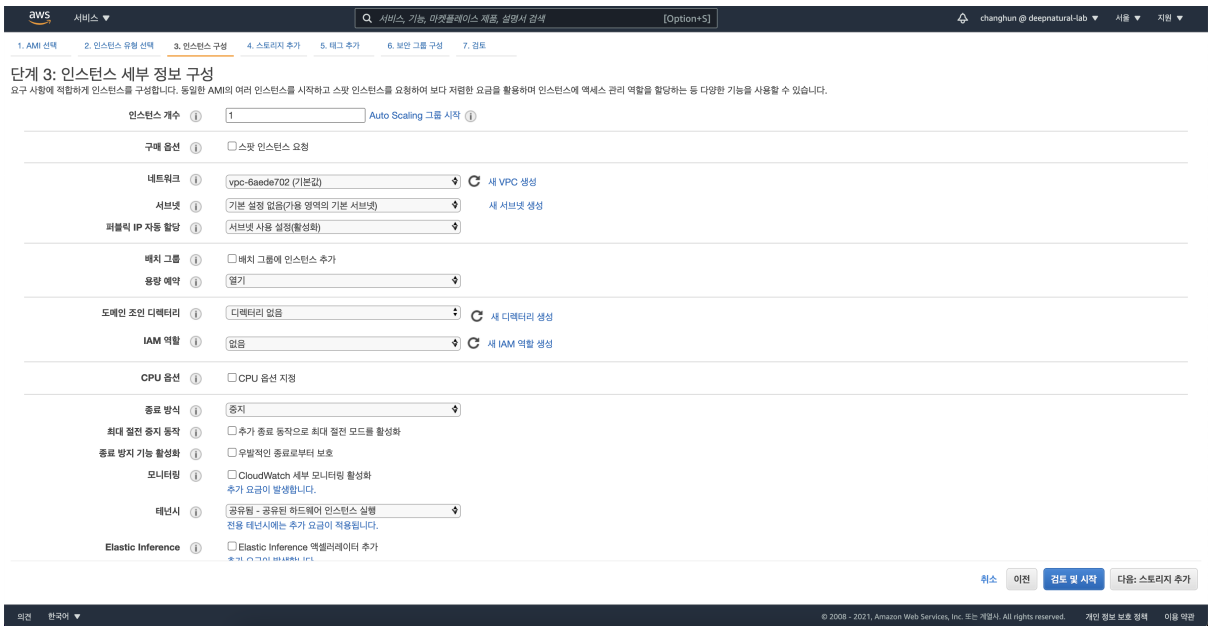
3. 인스턴스 유형 선택

- t2.small 선택
- 다음: 인스턴스 세부 정보 구성 클릭



4. 인스턴스 세부 정보 구성

- 다음: 스토리지 추가 클릭



5. 스토리지 추가

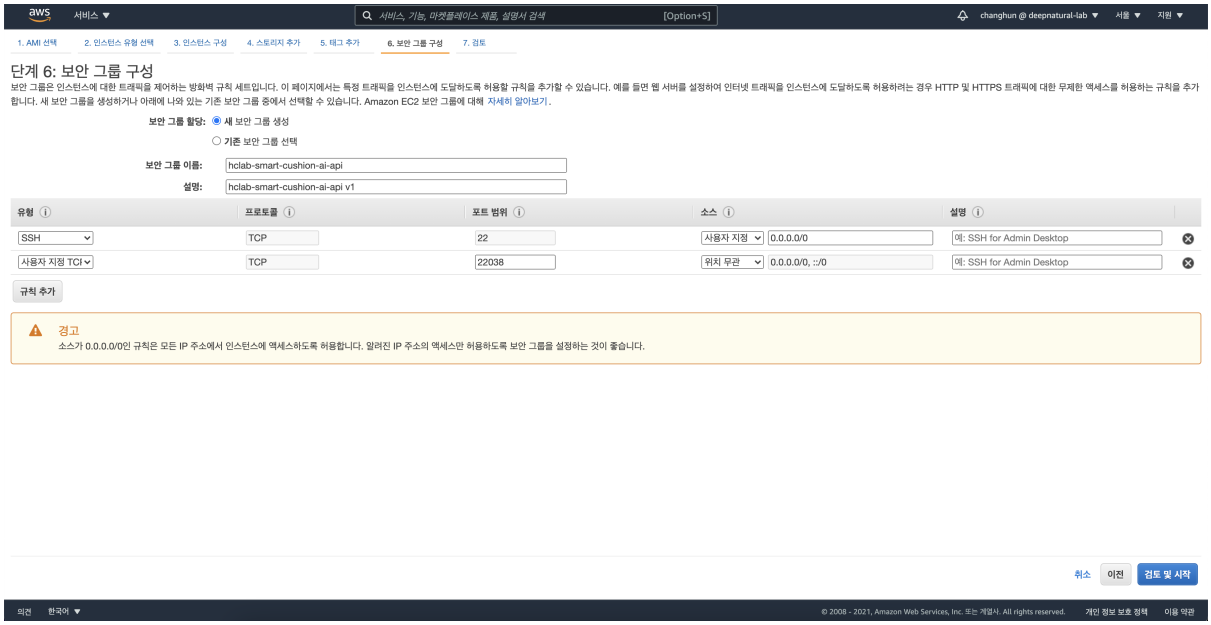
- 64GB 이상의 볼륨 사이즈 입력
- 다음: 태그 추가 클릭

6. 태그 추가

- 다음: 보안 그룹 구성 클릭

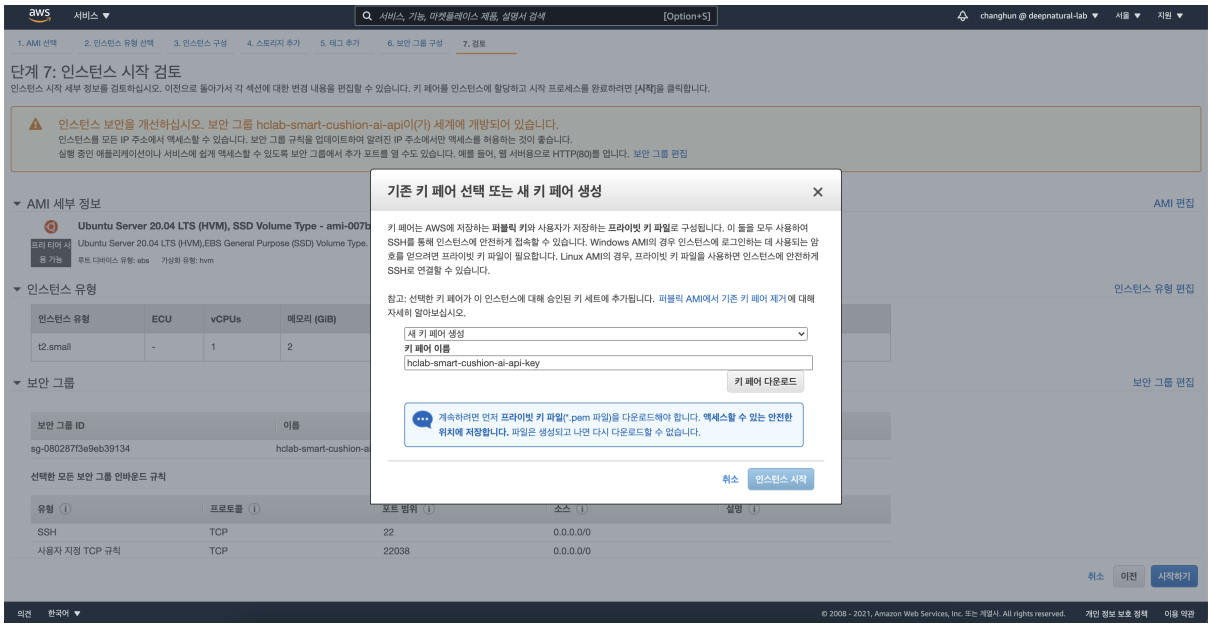
7. 보안 그룹 구성

- 새 보안 그룹 생성에서 규칙 추가를 클릭하여 API Request/Response를 위한 포트 (22038) 오픈
- 검토 및 시작 클릭



8. 인스턴스 시작 검토

- 기존에 생성한 프라이빗 키 파일이 존재하는 경우 `기존 키 페어 선택`, 존재하지 않는 경우 `새 키 페어 생성` 및 `키 페어 다운로드` 클릭
- `시작하기` 클릭



9. SSH를 이용해 서버에 연결

```
chmod 400 hclab-smart-cushion-ai-api-key.pem
ssh -i hclab-smart-cushion-ai-api-key.pem ubuntu@your_ec2_public_domain.compute.amazonaws.com
```

- 입력 예시

```
chmod 400 ~/Desktop/hclab-smart-cushion-ai-api-key.pem
ssh -i ~/Desktop/hclab-smart-cushion-ai-api-key.pem ubuntu@ec2-3-36-132-244.ap-nor
theast-2.compute.amazonaws.com
```

10. hclab-smart-cushion-ai-api 폴더를 준비함

11. 콘솔 창을 열고 위의 폴더로 이동함

```
cd path/to/hclab-smart-cushion-ai-api
```

12. make를 설치함

```
sudo apt-get install make
```

13. 학습을 진행할 경우 datasets 폴더에 데이터셋을 준비하고, 예측을 진행할 경우 resources 폴더에 모델을 준비함

14. API Server를 셋업해줌

```
make setup
```

15. 로컬 환경에서 아래와 같이 API Request를 보냄

```
curl --location --request POST 'http://your_ec2_public_domain.compute.amazonaws.com:22
038/analyze/' \
 --header 'Authorization: Token A0E1E013-A2FC-47F7-8751-A14E37C1C054' \
 --header 'Content-Type: application/json' \
 --data-raw '{
    "sensor_data": SENSOR_DATA,
    "user_data": USER_DATA
 }'
```

- 입력 예시(유저 정보와 존재할 때와 존재하지 않을 때 서로 다른 모델을 이용함)

```
curl --location --request POST 'http://ec2-3-36-132-244.ap-northeast-2.compute.ama
zonaws.com:22038/analyze/' \
 --header 'Authorization: Token A0E1E013-A2FC-47F7-8751-A14E37C1C054' \
 --header 'Content-Type: application/json' \
 --data-raw '{
   "sensor_data": [0,0,0,48,216,225,165,0,0,222,243,242,174,88,177,0,59,199,139,17
```

```
8,217,231,223,121,81,157,214,219,160,0,0,0],
   "user_data": {
      "height": 169.1,
      "weight": 50.6,
      "amount_of_activity": 1,
      "pain_info": {
            "neck": 3,
            "waist": 3
      }
   }
}'
```

```
curl --location --request POST 'http://ec2-3-36-132-244.ap-northeast-2.compute.ama
zonaws.com:22038/analyze/' \
 --header 'Authorization: Token A0E1E013-A2FC-47F7-8751-A14E37C1C054' \
 --header 'Content-Type: application/json' \
 --data-raw '{
   "sensor_data": [0,0,0,48,216,225,165,0,0,222,243,242,174,88,177,0,59,199,139,17
8,217,231,223,121,81,157,214,219,160,0,0,0]
}'
```

- 주의사항 : user_data에서 height, weight, amount_of_activity, pain_info 중 누락된 데이터가 있을 경우 sensor_data만을 이용하는 모델을 호출함.

---

4. 모델 및 성능 평가

모델

- Simple Convolutional Neural Network
  - 방석의 센서 데이터만 이용하는 모델
  - Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 4, 10, 1)]        0
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 6, 12, 1)          0
_________________________________________________________________
conv2d (Conv2D)              (None, 5, 11, 32)         160
_________________________________________________________________
batch_normalization (BatchNo (None, 5, 11, 32)         128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 6, 32)          0
_________________________________________________________________
dropout (Dropout)            (None, 3, 6, 32)          0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 5, 8, 32)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 7, 32)          4128
_________________________________________________________________
batch_normalization_1 (Batch (None, 4, 7, 32)          128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 4, 32)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2, 4, 32)          0
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 4, 6, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 5, 32)          4128
_________________________________________________________________
batch_normalization_2 (Batch (None, 3, 5, 32)          128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 3, 32)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 3, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 192)               0
_________________________________________________________________
dense (Dense)                (None, 11)                2123
=================================================================
Total params: 10,923
Trainable params: 10,731
Non-trainable params: 192
_________________________________________________________________
```

- Convolutional Neural Network with User Input
  - 방석의 센서 데이터와 함께 유저 데이터를 이용하는 모델
  - Architecture

```
Model: "functional_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
sensor_input (InputLayer)       [(None, 4, 10, 1)]   0
__________________________________________________________________________________________________
zero_padding2d (ZeroPadding2D)  (None, 6, 12, 1)     0           sensor_input[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 5, 11, 32)    160         zero_padding2d[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 5, 11, 32)    128         conv2d[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 3, 6, 32)     0           batch_normalization[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 3, 6, 32)     0           max_pooling2d[0][0]
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, 5, 8, 32)     0           dropout[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 4, 7, 32)     4128        zero_padding2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 4, 7, 32)     128         conv2d_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 2, 4, 32)     0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 2, 4, 32)     0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
zero_padding2d_2 (ZeroPadding2D (None, 4, 6, 32)     0           dropout_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 3, 5, 32)     4128        zero_padding2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 3, 5, 32)     128         conv2d_2[0][0]
__________________________________________________________________________________________________
user_input (InputLayer)         [(None, 10)]         0
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 2, 3, 32)     0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 64)           704         user_input[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 2, 3, 32)     0           max_pooling2d_2[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           4160        dense[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 192)          0           dropout_2[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 64)           4160        dense_1[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 256)          0           flatten[0][0]
                                                                 dense_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          32896       concatenate[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 11)           1419        dense_3[0][0]
==================================================================================================
Total params: 52,139
Trainable params: 51,947
Non-trainable params: 192
__________________________________________________________________________________________________
```

성능 평가

Model	Hyperparameter	Accuracy
Simple Convolutional Neural Network	Batch size : 16, Epochs : 30	Train : 75% Validation : 94% Total : 86%
Simple Convolutional Neural Network	Batch size : 4, Epochs : 30	Train : 83% Validation : 93% Total : 92.7%
Convolutional Neural Network with User Input	Batch size : 4, Epochs : 30	Train : 81% Validation : 95% Total : 94.2%

- Epochs이 30 이상이 되면 Overfitting에 의해 성능 향상이 거의 없는 것을 확인함.
- 실험 결과, 아래와 같은 Hyperparameter 조합이 최적임을 확인함.

```
BATCH_SIZE = 4
Epochs = 30
```