1. **데이터 늘림**

인당 5장(정면 웃음 안경 좌측아래 우측아래)으로 데이터를 늘리고 총 20명의 사람을 등록시켰다.

사람 수 늘리고 각도 늘어나니 **측면 정확도가 월등히 올라감**

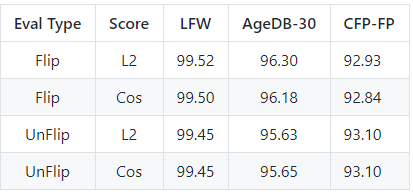
1. **한사람당 학습 사진을 늘렸는데 사진마다 피쳐값이 달라 어떻게 할지 고민**

각 피쳐 값 마다 사람지정(기존방법) vs 피쳐값들을 계산해서 한 사람은 하나의 값으로 나타내기(새로운방법)

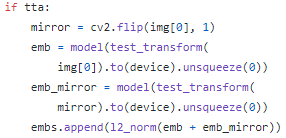
* 각 사진에 대한 피쳐값의 평균을 내어 그 사람의 피쳐값으로 하기로 함.
* 실제로 많은 코드들에서 사용되고 있는 기법으로 일반적으로 한사람을 다른 각도에서 본다고 해도 다른 사람으로 보이는 것이 말이 안되기 때문에 가능한듯?

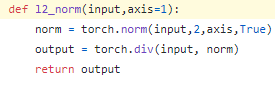
1. **그리고 깃허브에서 얼굴인식 과정에서 mtcnn을 거친 얼굴만 있는 사진을 더 효과적으로 학습시키는 방법을 찾아 도입함.**

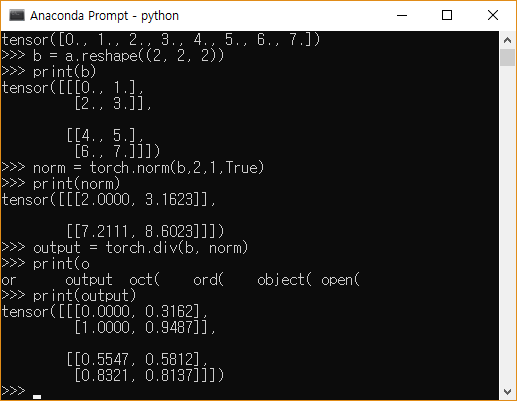
* MTCNN을 거쳐 나온 얼굴만 있는 사진(a)을 좌우반전 해서 한 장 더 만든 사진(b) 두개를 계산식을 사용해서 합침(거울 본다고 다른 사람으로 보이지 않는다고 직관적 이해 가능

이건 아이디어 만든 사람 실험결과 > flip이 이 사람 아이디어인데 가져다 쓰기로 함

* 정확한 과정은 아래와 같음

원본사진 모델에 돌리고, 좌우반전 한 사진 모델 돌리고, 나온 값을 더해서 l2\_norm일라는 식에 넣음

l2\_norm식은 옆에 랑 같은데 흔히 알고 있는 l2놈 식이 아님 그냥 이름을 저렇게 한거 같은데 왜 그랫는지 모르겠음. L2놈을 구하고(norm) 아래에 output = torch.div(input, norm) 에서 처음에 넣었던값 input(emb + emb\_mirror)에 norm을 나눠준 값을 사용함



저 식 과정 이해하기 쉽게 그냥 예시를 들면 위와 같음

* 결론은 이런 식 도입하니 성능이 좋아짐

1. 기존에 만들었던 model-at, model-fr을 Flask를 사용하여 서버에 올림.

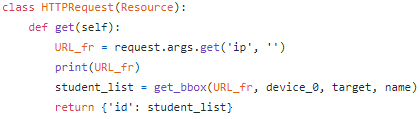
* 코드

Model-at(<https://github.com/KHU-Face-ID/taeho_repository/blob/864a562728b1778ecb0f94845df6d750c2fe5281/modelat_flask.py>)

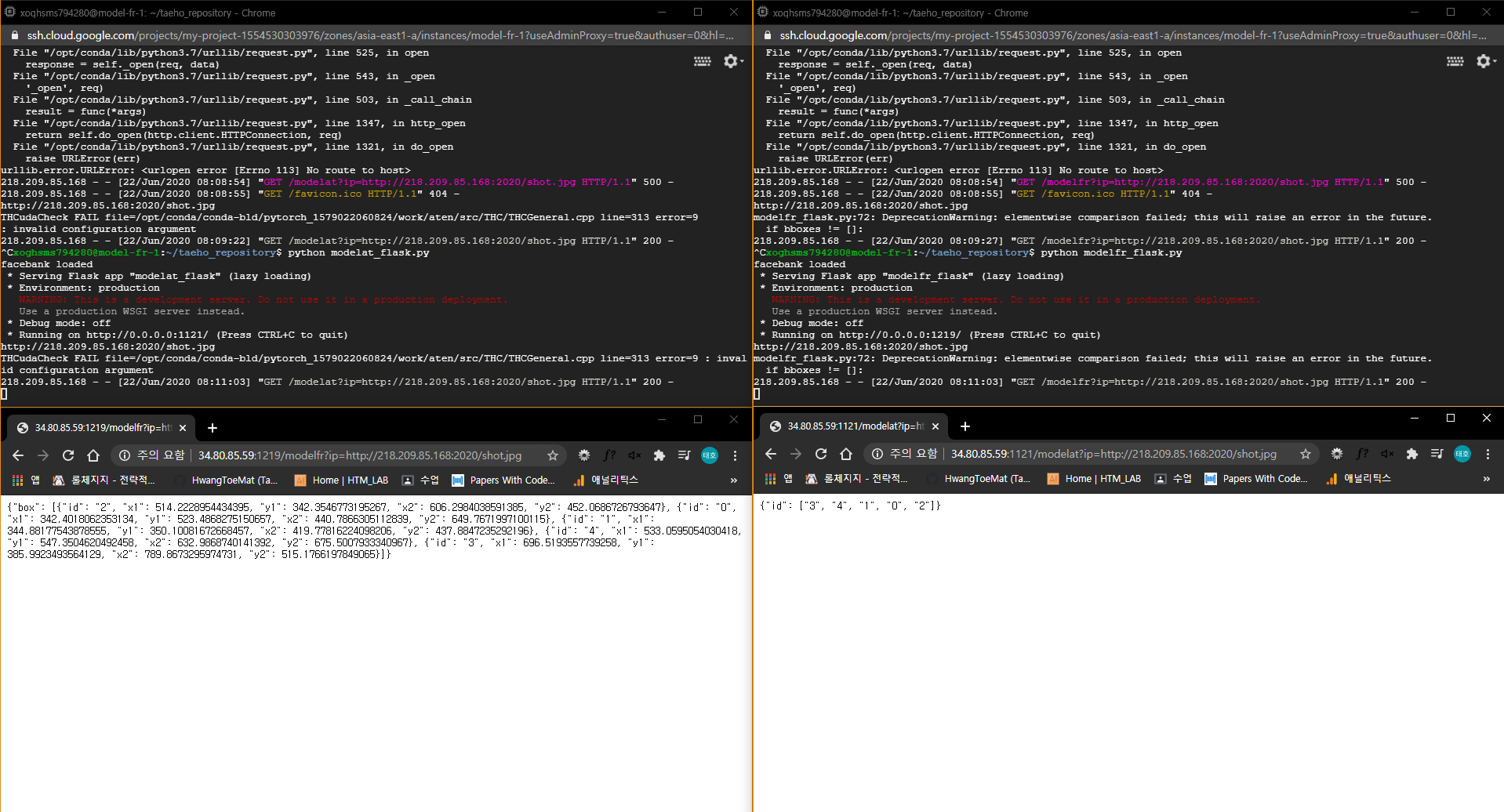
Model-fr(<https://github.com/KHU-Face-ID/taeho_repository/blob/864a562728b1778ecb0f94845df6d750c2fe5281/modelfr_flask.py>)

각각 modelat\_flask.py, modelfr\_flask.py 가 최종 실행 파일임

* Flask를 통해 rest API방식(Get만 사용함)으로 모델이 서버에서 돌아갈 수 있도록 구현함.



* 구현방식은 간단함. 먼저 얼굴인식 모델 외부 요청 Get으로 { ‘ip’ : XXX.XXX.XXX.XXX:OOO } 와 같은 형태로 아이피 카메라의 아이피 값이 들어온다면 카메라에서 촬영되고 있는 그 순간의 프레임을 가져와 학생들의 위치를 파악하여 {"box": [{"id": "JS", "x1": 17, "y1": 86, "x2": 19, "y2": 100}, {"id": "TH", "x1": 11, "y1": 66, "x2": 91, "y2": 100}]} 이런 식으로 값을 반환함
* 출석체크 모델의 경우 위와 같은 형태로 ip가 들어오면 {"id": [“TH”, “EH”, “QW”]} 이런 식으로 이름만 나옴



GCP에 올리고 돌려본 모습 왼쪽(위, 아래)가 modelfr이고 오른쪽이 modelat이다.

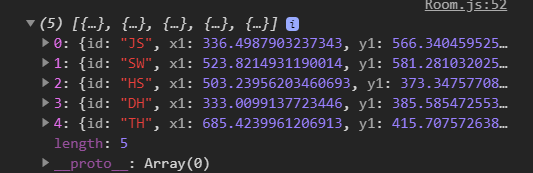
1. 프론트 서버와 모델서버가 통신하는 과정에서 CORS 문제가 발생

* CORS란?( <https://velog.io/@wlsdud2194/cors>)

CORS는 Cross Origin Resource Sharing의 약자로 **도메인** 또는 **포트**가 **다른 서버의 자원을 요청하는 매커니즘을** 말합니다. 이때 요청을 할때는 **cross-origin HTTP 에 의해 요청**됩니다. 하지만 **동일 출처 정책(same-origin policy)** 때문에 CORS 같은 상황이 발생 하면 **외부서버에 요청한 데이터를 브라우저에서 보안목적으로 차단**합니다. 그로 인해 정상적으로 데이터를 받을 수 없습니다.

라는 플라스크 자체의 cors를 다루는 라이브러리를 통해 해결함. >> 이거 때문에 하루날림……

이를 통해 결국 프론트 서버와 통신할 수 있게됨.



프론트에서 Get으로 모델서버에 요청 후 값을 받은 모습 (통신완료)