

캡스톤 디자인 5월 3주차 보고서

주제명	얼굴인식 기반 퍼스널 브랜딩 앱	
4조	팀장: 17011599 안정연 팀원: 17011467 김가영, 17011539 최민경, 17011600 안지송	2021.05.14 ~ 2021.05.20

기본 계획	1. 개별 개발 일정 (정연)					
	구현	진행률	시작일	종료일	총 기간	진척도
	Android, ios 개발 환경 구축	완료	21/03/18	21/03/24	5일	100%
	UI 화면 개발	완료	21/04/16	21/05/03	15일	100%
	카메라 구현	완료	21/05/04	21/05/08	4일	100%
	서버 및 API 연동	완료	21/05/09	21/05/18	9일	100%
	Unit 테스트	진행중	21/05/18	21/05/23	5일	80%
	2. 개별 개발 일정 (가영)					
	구현	진행률	시작일	종료일	총 기간	진척도
	데이터베이스 설계	완료	21/04/01	21/04/10	10일	100%
	서버 및 DB 구축	완료	21/04/15	21/05/04	19일	100%
	앱과 서버 연동	완료	21/05/09	21/05/18	14일	100%
	디버깅 및 테스트	진행중	21/05/12	21/05/23	12일	80%
	3. 개별 개발 일정 (민경, 지송)					
	구현	진행률	시작일	종료일	총 기간	진척도
	얼굴 데이터 구축	완료	21/03/28	21/04/01	5일	100%
	정면 얼굴 영역 검출 (랜드마크검출)	완료	21/04/12	21/04/16	5일	100%
	측면 얼굴 영역 검출 (랜드마크검출)	완료	21/05/01	21/05/12	11일	100%
	얼굴 특징 분류	완료	21/04/26	21/05/10	14일	100%
	얼굴 인식 시스템 테스트	진행중	21/05/13	21/05/23	11일	80%

<p>금주 계획</p>	<ol style="list-style-type: none"> 1. 정면 및 측면 얼굴 특징 검출 세분화 2. 얼굴형 분류 모델에 Face alignment 적용 및 모델 추출 후 저장 3. 얼굴 특징에 따른 추천 스타일 정리 4. 앱과 얼굴 분석 시스템간의 실시간 통신 구현 5. 앱 내 카메라 페이스라인 적용
<p>금주의 수행 내용 및 이슈 해결 방안</p>	<ol style="list-style-type: none"> 1. 정면 및 측면 얼굴 특징 검출 세분화 <ul style="list-style-type: none"> • 정확도 개선 및 특징 검출 세분화를 위해 실제 사진 촬영 후 특징 검출 실행 • 특징 검출 후 결과 판별 기준 수정 • 눈꼬리, 입술 크기 판별 등을 추가하여 특징 검출 세분화 <div data-bbox="480 604 1115 1205" data-label="Text"> <pre> 얼굴형_인덱스 4 얼굴 비 3 얼굴 옆광대 1 얼굴 앞광대 1 코볼 1 0.20769230769230784 미간 0 0 눈 세로 0 0 눈 가로 1 0.5159090909090907 눈꼬리 0 꼬막눈 0 입술 1 </pre> </div> 2. 얼굴형 분류 모델에 Face alignment 적용 및 모델 추출 후 저장 <ul style="list-style-type: none"> • Face alignment 적용 <div data-bbox="499 1431 1361 1861" data-label="Image"> </div> <p>다음과 같이 눈을 가지고 삼각형을 만든 다음 삼각형 에서 제일 작은 각의 값을 세타라 두고 arc tan 세타 만큼 회전 시키면 face alignment 적용 가능.</p>

이슈 : 훈련 시켜야 할 데이터 사진들에 face alignment를 적용할 때 모든 간혹 이상하게 회전하는 경우가 존재 해 정확도가 이전보다 떨어짐.
해결방안 : 일단 훈련데이터 셋을 모두 face alignment를 적용한 다음 일부 이상한 사진들은 따로 하나씩 직접 face alignment 적용.

- 모델 추출하여 저장

Save on GPU, Load on CPU

Save:

```
torch.save(model.state_dict(), PATH)
```

Load:

```
device = torch.device('cpu')
model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load(PATH, map_location=device))
```

GPU가 지원되는 Colab 환경에서 모델을 학습 시키고 모델을 추출함.
CPU에서 모델 로드를 해야하는 상황이여서 PyTorch.org에서 나온 방법으로 모델을 추출함.

```
class_names = ["각진형", "계란형", "둥근형", "마름모형", "하트형"]

transforms_test = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

device = torch.device("cpu")

# 모델 불러오기
model = torch.load(PATH, map_location=torch.device('cpu'))

# 드롭아웃 및 배치 정규화를 평가 모드로 설정
model.eval()

# 이미지 불러오기

image = transforms_test(image_front).unsqueeze(0).to(device)

# 불러온 이미지를 얼굴형 분류 모델에 집어넣기
with torch.no_grad():
    outputs = model(image)
    _, preds = torch.max(outputs, 1)
    num = preds[0].tolist()
```

추출한 모델 로드 후 실행을 시켜 사용자의 얼굴형 판단.

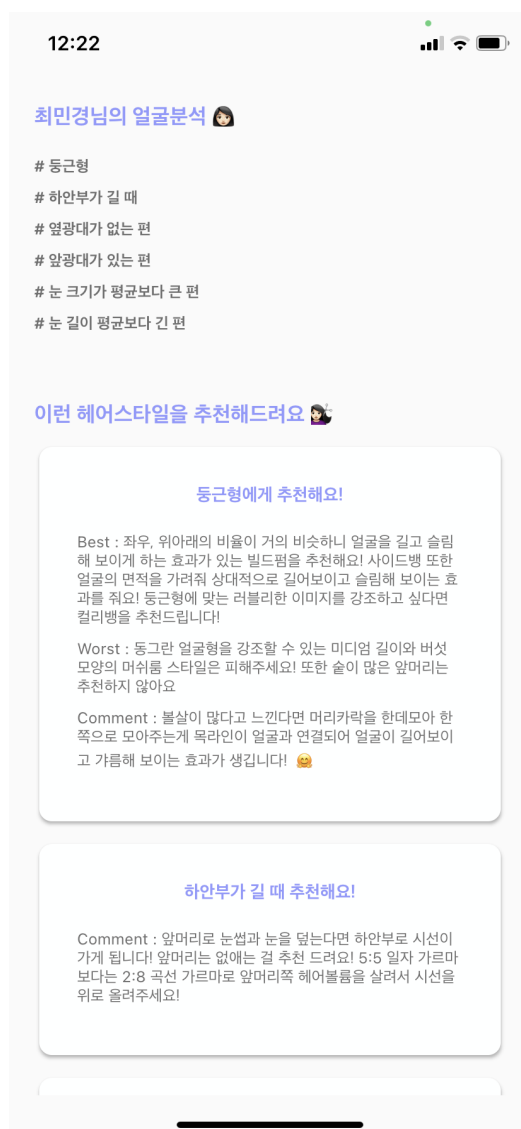
3. 얼굴 특징에 따른 추천 스타일 정리

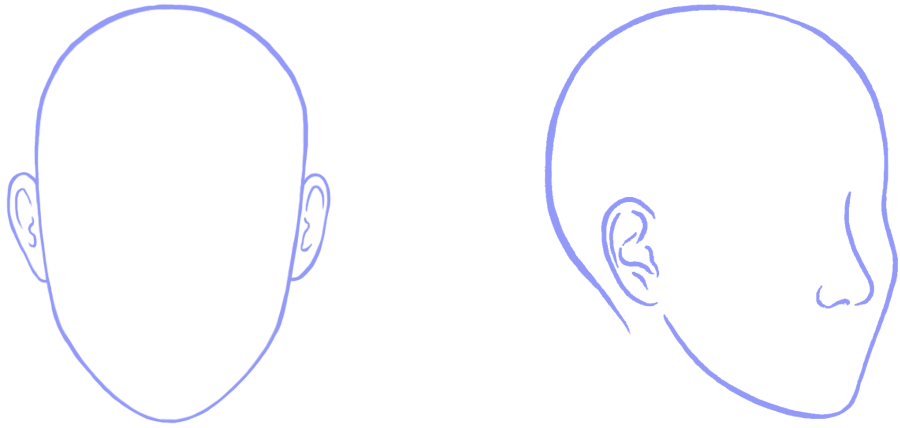
- 얼굴형에 맞는 추천 헤어스타일 정리
- 얼굴 상/중/하안부 비율에 따른 추천 헤어 스타일 정리

- 얼굴의 앞/옆 광대 유무에 따른 추천 헤어 스타일 정리
- 눈 가로, 세로 길이, 눈꼬리, 미간, 콧볼, 얼굴형 등
사용자에 얼굴분석 결과에 따른 메이크업 스타일 정리
- 얼굴형에 따른 눈썹 모양 추천 정리
- 얼굴분석을 한 결과를 토대로 추천해줄 헤어&메이크업 관련 스타일링 관련해
FireBase에 저장 완료

4. 앱과 얼굴 분석 시스템간의 HTTP 통신 구현

- 이슈 : 기존에 하려고 했던 socket 통신이 필요하지 않다고 판단됨. 그에따라
통신 방법을 비동기식으로 변경함
- 분석하기 버튼을 누르면 얼굴인식시스템에 POST방식으로 사진 URL을 전송
후 결과값을 GET방식으로 가져와 데이터베이스에 저장 후 결과에 표시되도록
구현



	<p>5. 앱 내 카메라 페이스라인 제작</p> <ul style="list-style-type: none"> • 저작권 이슈를 피하기 위해 페이스라인 이미지를 직접 제작함. 
차주 수행 계획	<ol style="list-style-type: none"> 1. 얼굴 인식을 개선 2. 예외 처리 및 테스트 3. 앱 내 카메라 페이스라인 적용
참고 문헌	<p>PyTorch model saving (Saving and Loading Models — PyTorch Tutorials 1.8.1+cu102 documentation)</p> <p>스타일 추천 관련 (https://www.youtube.com/channel/UC2viazvlrvicBWCIWHKd70O, https://www.youtube.com/channel/UCqwur42zcSwaGzfLpnEjHIA)</p> <p>firebase documentation (https://firebase.google.com/docs)</p>