

# 추억 사진관

team : HI-AI



Album

사용자가 사랑하는 사람들과의 소중한 기억을 모아서 앨범 형태로 보관할 수 있도록 도와줍니다.

[View details »](#)



Photo Upload

사용자가 편리하게 사진을 업로드하고 자신만의 앨범을 만들어 소중한 순간들을 보관하도록 도와줍니다.

[View details »](#)



Memory Search

사용자가 업로드한 사진을 기반으로 유사한 사진들을 자동으로 찾아주어 소중한 기억을 다시 찾아볼 수 있게 합니다.

[View details »](#)

## ☰ 프로젝트 Intro

- "추억 사진관"은 사용자의 사진을 저장하고, 이를 관리하며, 특정 얼굴의 사진을 검색하는 기능을 제공하는 웹 기반 애플리케이션입니다.
- 사용자는 웹 페이지를 통해 사진을 업로드하고, 업로드한 사진들을 앨범으로 볼 수 있습니다.
- 특정 얼굴 사진을 Key로 사용하여 기존 사진 중 유사한 얼굴을 가진 사진을 찾을 수 있습니다.



# 기술 스택

- Frontend: HTML, CSS, JavaScript, Bootstrap
- Backend: Node.js, Express
- AI 모델 서버 : Flask
- Database(DB): MongoDB
- 얼굴 인식 및 임베딩: MTCNN, FaceNet(InceptionResNetv1)
- 서버 배포: AWS Elastic Beanstalk
- Storage: AWS S3



## 사진 분류과정

총 3가지의 step으로 이루어집니다.

### 1. Face Detection



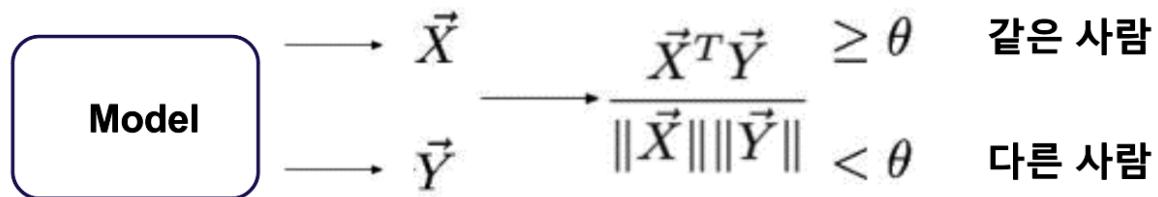
- Face Detection 모델(MTCNN)을 활용하여 사진에서 얼굴의 영역을 얻습니다.

### 2. Feature Extraction



- Feature Extraction 과정을 통해 얼굴의 임베딩 벡터를 추출합니다.

### 3. Similarity & Threshold



Feature Extraction	Cosine similarity	Threshold
--------------------	-------------------	-----------

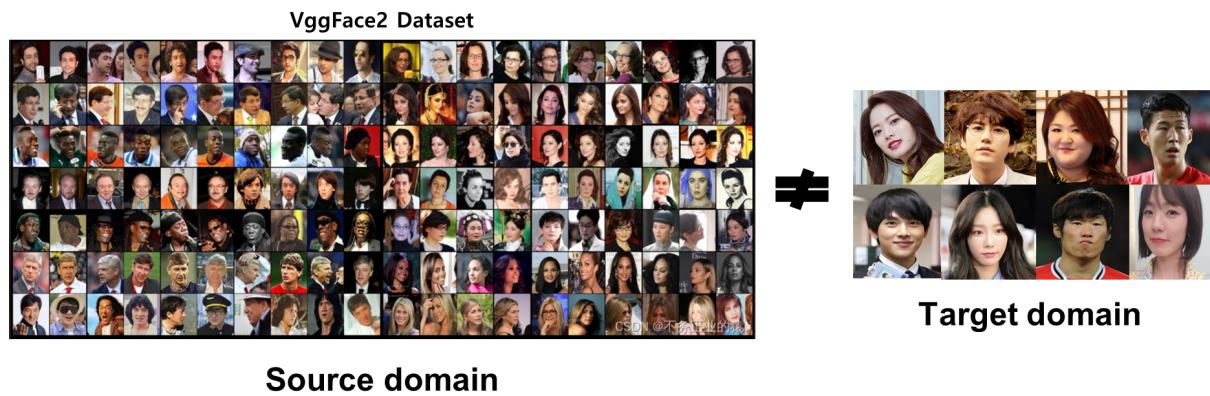
- 추출한 임베딩 벡터와 key인물의 임베딩 벡터의 유사도를 계산하여 같은 사람인지 판단합니다.



### 문제 정의

- 얼굴 인식을 위한 라이브러리가 대부분 서양인의 데이터셋으로 학습되어 있었습니다.
- 학습된 데이터셋에 저희의 예상 사용자인 한국인(동아시아)인의 비중이 작았습니다.
- 아래 표를 보면 사전 학습된 모델 중에서 얼굴인식 분야의 'East Asian'의 정확도가 다른 인종에 비해 낮은 것을 확인할 수 있습니다.

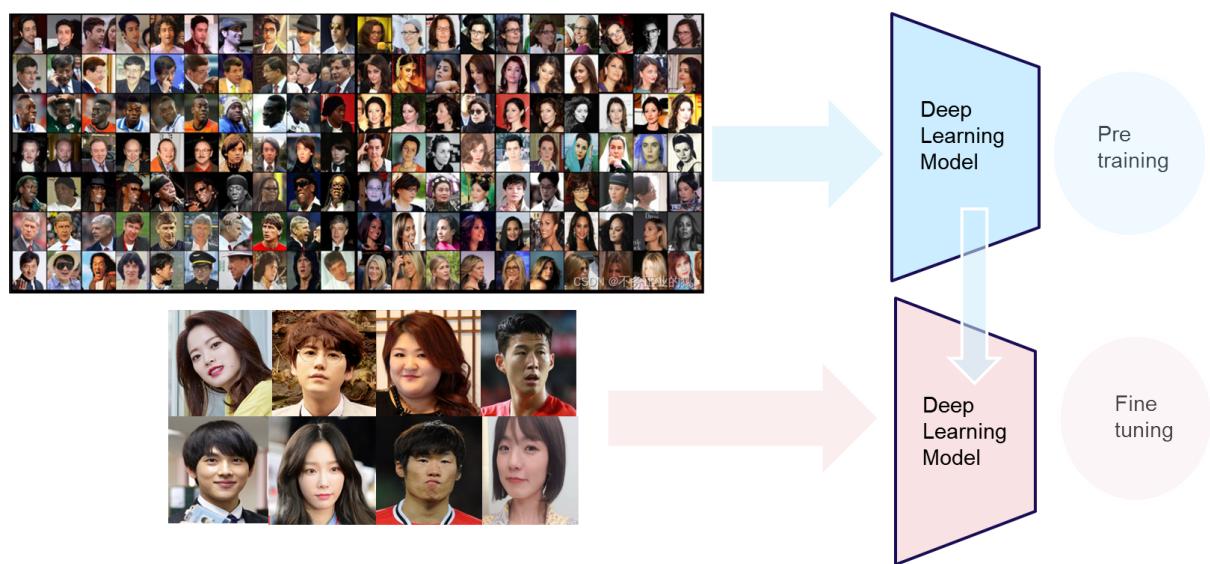
Backbone	Dataset	MR-ALL	African	Caucasian	South Asian	East Asian	Link(onnx)
R100	Casia	42.735	39.666	53.933	47.807	21.572	GDrive
R100	MS1MV2	80.725	79.117	87.176	85.501	55.807	GDrive
R18	MS1MV3	68.326	62.613	75.125	70.213	43.859	GDrive
R34	MS1MV3	77.365	71.644	83.291	80.084	53.712	GDrive
R50	MS1MV3	80.533	75.488	86.115	84.305	57.352	GDrive
R100	MS1MV3	84.312	81.083	89.040	88.082	62.193	GDrive
R18	Glint360K	72.074	68.230	80.575	75.852	47.831	GDrive
R34	Glint360K	83.015	79.907	88.620	86.815	60.604	GDrive
R50	Glint360K	87.077	85.272	91.617	90.541	66.813	GDrive
R100	Glint360K	90.659	89.488	94.285	93.434	72.528	GDrive



→ Pretraining에 활용된 데이터셋과 Target 데이터셋의 분포가 다르다, **Domain Shift** 발생

**Domain Shift**로 문제를 정의하고, 문제 해결을 위해 **Domain Generalization**을 진행했습니다.

**Domain Generalization** 방법 중에서 Target 데이터셋을 직접 구해, Fine-tuning 하는 방법을 선택했습니다.



- Domain에 적합한 데이터 Crawling & Labeling
- 적합한 Metric 선정
- 선정한 Metric을 이용하여 모델 탐색(관련 SOTA모델 탐색, 모델 특징 분석)
- 적합한 데이터로 Fine-tuning

**Pre-Trained** 모델에 **Target domain**에 맞는 데이터셋을 **Fine-Tuning** 시키기



## 1. Dataset

- 한국인 유명인 데이터셋 2000장  
구축(크롤링)
  - Train : 1500장, Test : 500장 구분
  - 데이터 정보:
    - 100명 (남 : 50명, 여 : 50명)
    - label하나당 20장

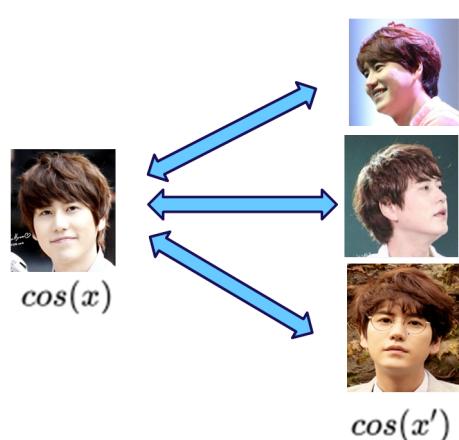
## 2. Metric

### a. Distance : 코사인 유사도(Cosine similarity)

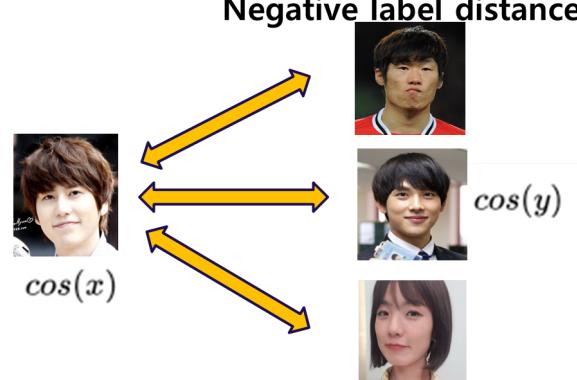
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

### b. Positive distance & Negative distance

Positive label distance

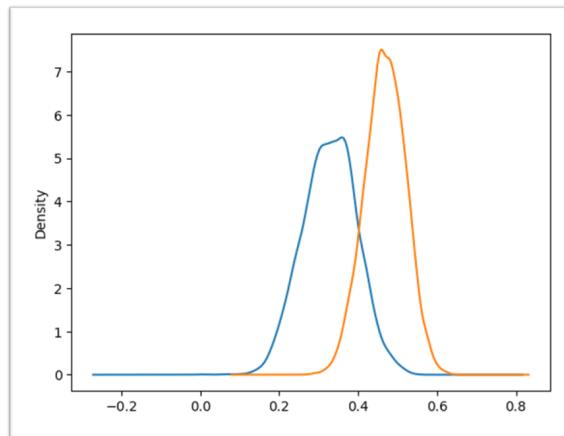


Negative label distance



### c. KDE(Kernel Density Estimation)

- Positive distance
- Negative distance
- X 축 : Distance
- Y 축 : 확률 밀도



1. 수치적 Metric은 Positive mean과 Negative mean 사이를 탐색하여, 정확도(Accuracy)의 고점 기준으로 Threshold 를 정한다.  
=> Threshold로 정확도, 정밀도, 재현율, F1 스코어를 정한다.
2. 두 분포 사이에 겹치는 부분이 최소화 되는 것이 이상적이다.

### 3. Pre-train Model 탐색 & 선정

1. 선정 방법
  - a. 테스트 데이터 셋 500장 (Crawling한 데이터 셋 25명, 각 20장)
  - b. 확인하고자 하는 Pre-train Model로 테스트 데이터 셋의 Embedding vector를 추출
  - c. 추출된 Embedding vector로 위의 Metric을 이용하여 정확도(Accuracy)를 구하여 선정
2. Pre-train Model 탐색

Model(Backbone)	dataset	Accuracy	F1-score
Arcface(ResNet-18)	MS1MV3	0.5485	0.4579
Arcface(MobileNet)	face-emore	0.5321	0.4410
Arcface(ResNet-50)-scratch	Web-face	0.7812	0.7644

Facenet(Inception-ResNet)	VggFace2	0.8810	0.8262
Facenet(ResNet-50)-scratch	Web-face	0.7286	0.6227

### 3. Pre-train Model 설정

- Facenet(Inception-ResNet) 설정

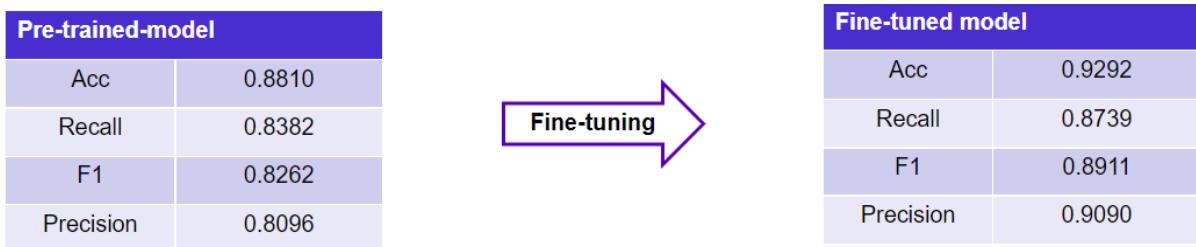
Facenet(Inception-ResNet)	
Accuracy	0.8810
Recall	0.8382
F1	0.8262
Precision	0.8096

## Fine-Tuning

- 모델설정 : VGGFace2 데이터셋으로 Pretrain된 **FaceNet(InceptionResNetv1)** 채택
- 모델성능 : test 데이터셋 기준 accuracy : **0.8810%**
- lr, Optimizer, layer Freezing, Resize 등 하이퍼파라미터 변경 실험

LEARNING RATE		OPTIMIZER		LAYER FREEZE		RESIZE		FINAL		
lr	Acc	Optimizer	Acc	freeze	Acc	Resize	Acc	Fine-tuned model		
1e-4	0.8945	SGD	0.8943	No-freeze	0.8945	112	0.8943	Acc	0.9292	
5e-5	0.8815	Adam	0.8234	Bn	0.8942		160	Recall	0.8739	
1e-5	0.6652		conv	0.8628	F1			0.8911		
5e-6	0.5662							Precision	0.9090	
1e-6	0.4531									

- 최종 결과(accuracy) : 0.8810 → 0.9292



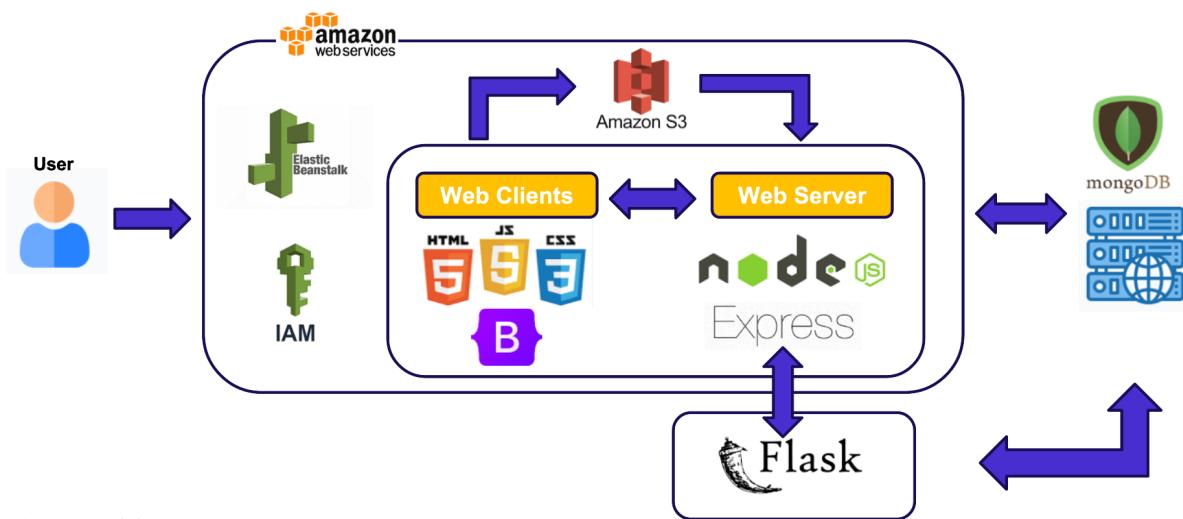
## Product Serving

### 1. Service Architecture



본 프로젝트는 마이크로서비스 아키텍처를 사용하였습니다. 사용자와 상호작용을 하는 웹 UI와 웹 서버는 Node.js로, AI와 관련된 작업을 수행하는 API 서버는 Flask로 구성하였습니다. Node.js와 Flask 사이의 데이터 전달은 RESTful API를 통해 이루어졌습니다. 이를 통해 서비스의 모듈화와 확장성을 보장하였습니다.

## 2. 사용된 기술 및 도구



### a. Frontend

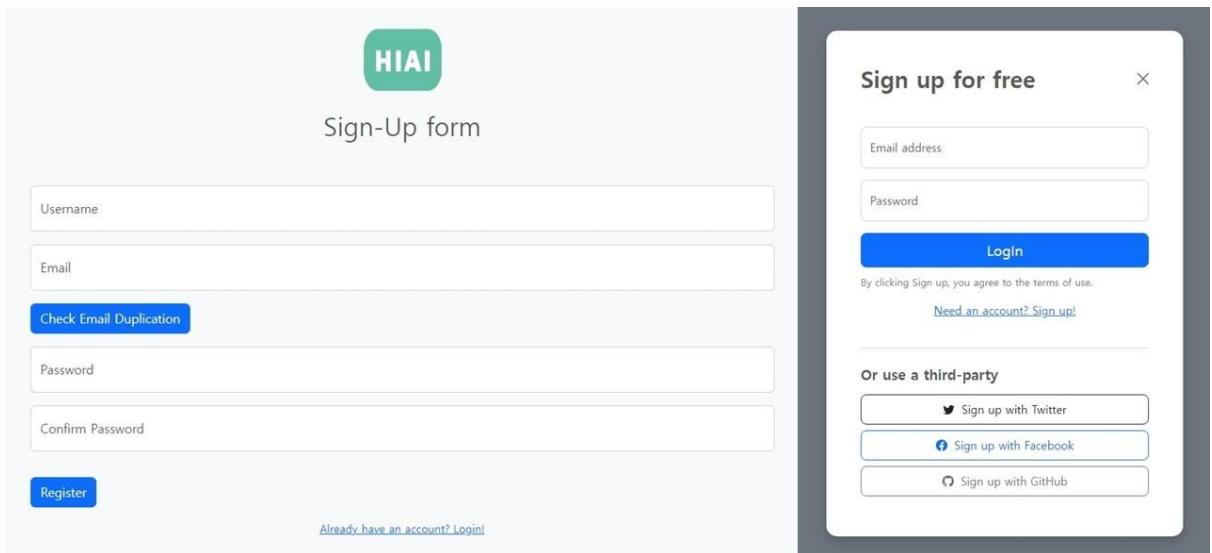
Frontend에서는 사용자와 상호작용을 담당하는 웹 UI를 구성하기 위해 Bootstrap, JavaScript, CSS, HTML을 사용하였습니다. 특히, Bootstrap은 UI 컴포넌트를 쉽게 만들 수 있게 해주는 프레임워크로서, 이를 통해 사용자 친화적인 웹 페이지를 제작하였습니다.

### b. Backend

Backend는 사용자의 요청을 처리하고 결과를 반환하는 역할을 담당하였습니다. 이를 위해 JavaScript로 서버 사이드 애플리케이션을 개발할 수 있게 해주는 플랫폼인 Node.js와 Node.js 위에서 동작하는 웹 애플리케이션 프레임워크인 Express를 사용하였습니다. 이러한 기술의

선택은 웹 서버 구축과 RESTful API 제공에 필요한 유연성과 확장성을 제공하기 때문입니다.

서버의 주요 기능 중 하나는 사용자 인증입니다. 이를 위해 우리는 bcrypt와 passport, connect-mongodb-session 라이브러리를 사용하였습니다.



## i. bcrypt

bcrypt는 사용자의 비밀번호를 안전하게 저장하기 위해 사용되었습니다. 사용자가 회원가입을 진행하거나 로그인할 때, 입력한 비밀번호는 bcrypt를 통해 해시화되어 데이터베이스에 저장됩니다. 이렇게 해시화된 비밀번호는 원본 비밀번호를 추측하는 것을 매우 어렵게 만들어, 사용자의 정보 보호에 도움을 줍니다.

## ii. Passport

Passport는 인증을 처리하는 프레임워크로, 이 프로젝트에서는 세션 기반 인증을 관리하는데 사용되었습니다. 세션 기반 인증에서는 사용자가 로그인을 하면 서버는 세션 ID를 생성하고 클라이언트에게 이를 전달합니다. 클라이언트는 이후 요청마다 이 세션 ID를 포함하여 서버에 전송하게 되며, 서버는 이를 통해

사용자를 인증합니다. Passport는 이러한 세션 ID의 생성과 관리, 사용자 인증 과정을 처리해 줍니다.

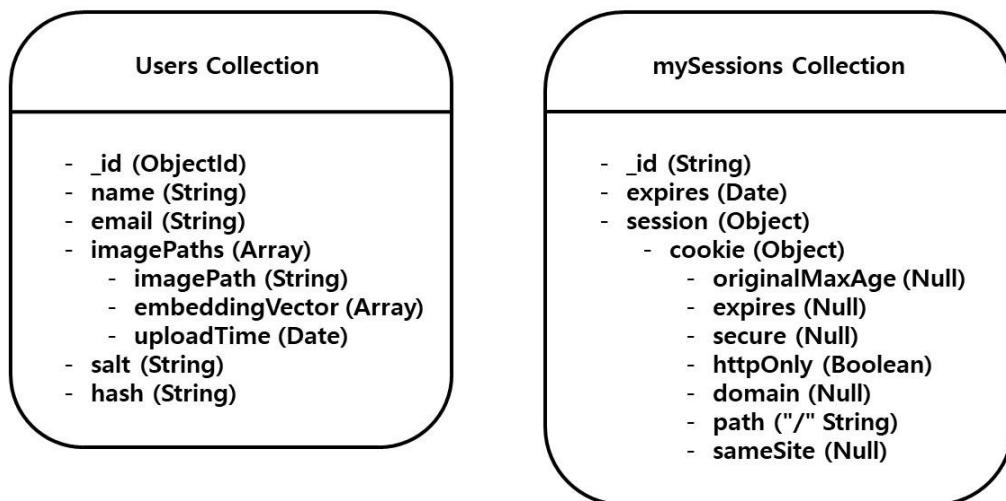
### iii. connect-mongodb-session

connect-mongodb-session는 세션 데이터를 MongoDB에 저장하기 위해 사용했습니다. 이 패키지는 MongoDB를 세션 저장소로 사용하는 데 필요한 모든 설정을 제공하며, 이를 통해 Express 애플리케이션의 세션 데이터를 MongoDB에 안전하게 저장하고 조회할 수 있습니다. 이렇게 하면 서버가 재시작되거나 클러스터 환경에서 여러 인스턴스 간에 세션을 공유할 수 있게 됩니다.

## c. AI 모델 서버

AI 모델 서버에서는 얼굴의 임베딩 벡터를 계산하고 유사도를 구하여 결과를 반환하는 역할을 담당하는 API 서버를 구성하기 위해 Flask를 사용하였습니다. Flask는 Python으로 웹 서버를 간단히 구현할 수 있게 해주는 라이브러리로, 본 프로젝트에서는 이를 통해 AI 관련 작업을 처리하는 API를 제공하였습니다.

## d. Database



데이터베이스로는 MongoDB Atlas를 사용하였습니다. MongoDB Atlas는 클라우드 환경에서 MongoDB를 사용할 수 있게 해주는 서비스로, 본 프로젝트에서는 이를 통해 사용자의 이미지 URL과 해당 이미지의 임베딩 벡터를 저장하였습니다.

## e. Storage

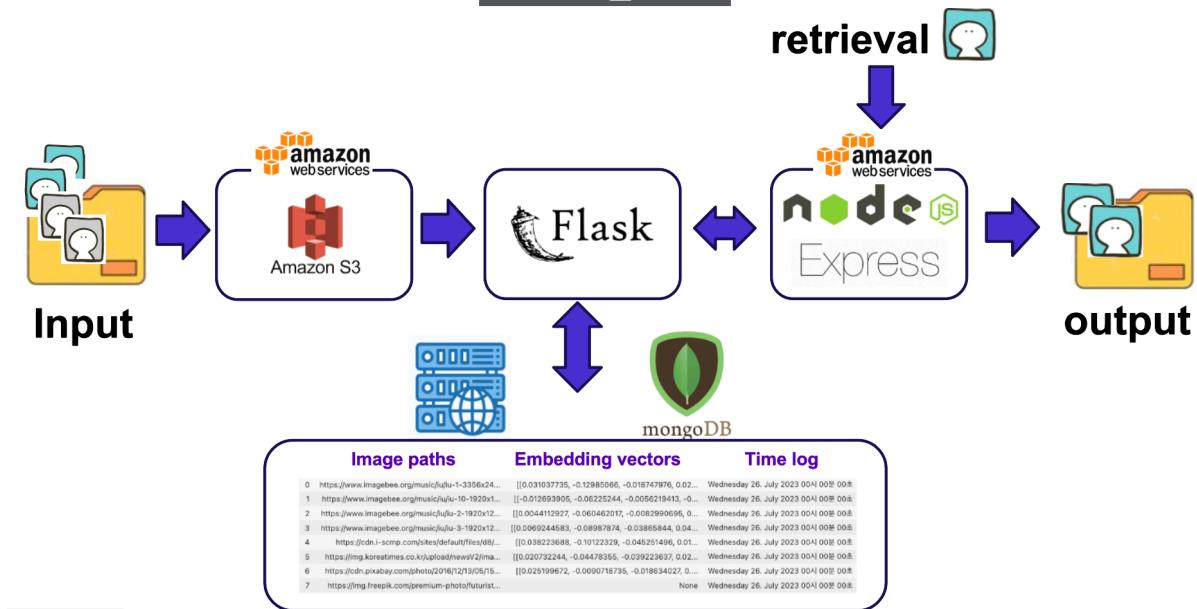
□	이름	▲   유형	□	1690784047312_5a5103bd-42b4-4a5f-9529-ea47dc205b00.jpg	jpg
□	5555@naver.com/	폴더	□	1690784139400_54c50dd4-f6a3-4d9b-90c9-71d0a0e45f23.jpg	jpg
□	admin@admin.com/	폴더	□	1690784139400_be0e8c2c-6df3-4db6-971a-88fd4767627a.jpg	jpg
□	hong@naver.com/	폴더	□	1690784139406_cfb949df-1013-4d1a-b2d0-16e02417f939.jpg	jpg
□	jhl9804@naver.com/	폴더	□	key/	폴더
□	jhshin1030@naver.com/	폴더	□		
□	sgh00@naver.com/	폴더	□		

이미지는 AWS S3에 저장하였습니다. S3는 AWS에서 제공하는 스케일링이 용이한 객체 storage로, 본 프로젝트에서는 이를 통해 사용자의 이미지를 안정적으로 저장하고 관리하였습니다.

## f. Product Serving

프로젝트를 AWS Elastic Beanstalk을 통해 배포하였습니다. AWS Elastic Beanstalk은 애플리케이션 배포를 단순화해주는 서비스로, 인프라 구성, resource provisioning, 애플리케이션 health monitoring 등을 자동으로 처리해줍니다. 본 프로젝트에서는 이를 통해 안정적으로 서비스를 운영할 수 있었습니다.

# 3. 주요 기능 및 구현 방법



### a. 사진 업로드 기능

The screenshot shows the **Memory Studio** web application. At the top, there is a navigation bar with links: **Memory Studio**, **Home**, **Album**, **Find**, and **Profile**. Below the navigation bar, the main content area has a title **FILE UPLOAD** with a star icon.

The upload interface includes a dashed box for dragging files, a **Drag files to upload** placeholder, a **Choose File** button, and an **Upload** button.

To the right, a progress bar table lists three files being uploaded:

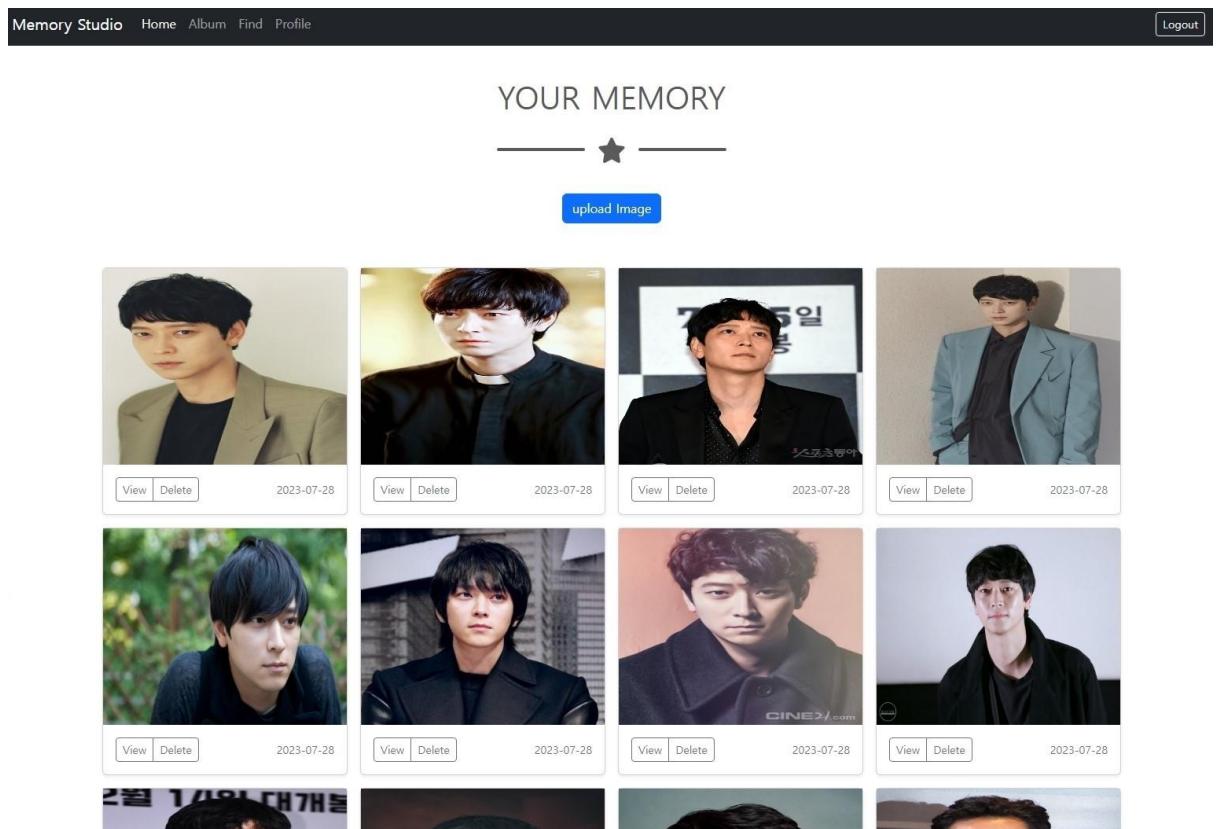
File Name	Status	Progress	Speed
Hangain_6.j...	100% done	X	317475 90KB/sec
Hangain_7.j...	100% done	X	72570 90KB/sec
Hangain_8.j...	100% done	X	35582 90KB/sec

사용자는 웹 페이지를 통해 사진을 업로드할 수 있습니다. 사용자가 사진을 업로드하면, Frontend에서 Backend로 사진 저장 요청이 발생합니다. Backend에서는 이 요청을 처리하여 사진의 원본을 S3에 저장하고, S3에 접근할 수 있는 URL을 반환합니다. 그 후, 이 URL을 Flask API로 요청하여, Flask API에서는 이 URL의 이미지를 기반으로

임베딩 벡터를 계산하고, 이 임베딩 벡터와 이미지 URL을 MongoDB에 저장합니다.

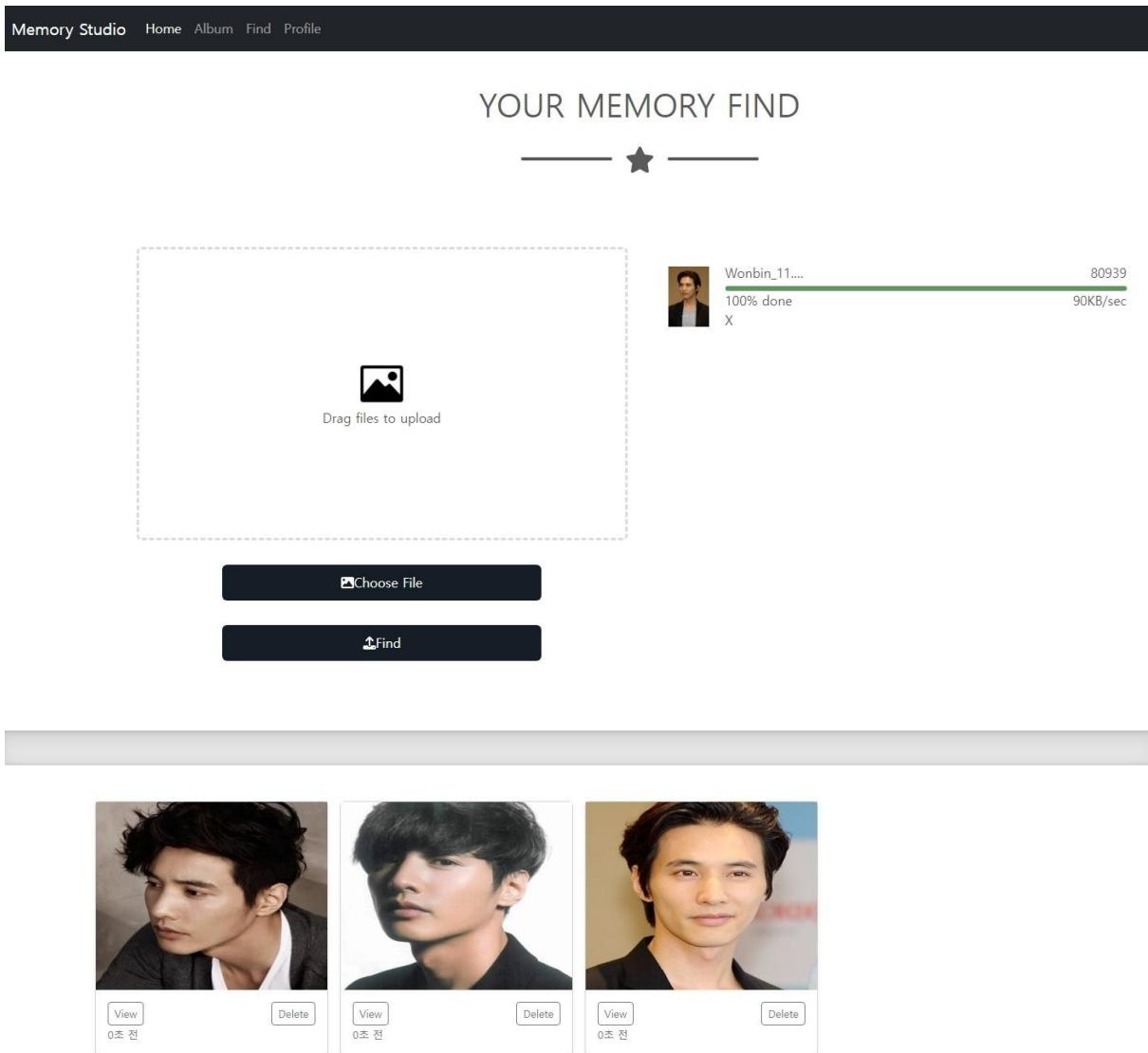
이 과정에서는 사용자가 업로드한 사진을 기반으로 얼굴 인식을 수행하고, 인식된 얼굴을 기반으로 임베딩 벡터를 생성하는 작업이 포함됩니다. 이 작업을 통해 사용자가 업로드한 사진의 얼굴 정보를 벡터 형태로 변환하여 데이터베이스에 저장할 수 있습니다.

## b. 사진 앨범 기능



사용자는 웹 페이지를 통해 업로드한 사진들을 앨범 형태로 볼 수 있습니다. 사용자 별로 다른 앨범 화면을 보여주기 위해, S3에는 사용자의 이메일 별로 폴더를 생성하여 이미지를 저장하였습니다. 사용자가 로그인하면, 해당 이메일에 해당하는 S3의 폴더를 찾아 해당 폴더에 저장된 사진들을 웹 페이지에 출력하여 보여줍니다.

### c. 사진 검색 기능



사용자는 웹 페이지를 통해 특정 얼굴 사진을 검색할 수 있습니다.

사용자는 키로 사용할 얼굴의 사진을 업로드하면, 이를 기반으로 기존에 업로드되었던 사진들 중에서 키 사진과 가장 유사한 얼굴을 가진 사진을 찾아 반환합니다.

Key로 사용할 사진을 업로드하면, Backend에서는 이를 S3에 저장하고, 저장된 이미지의 URL을 Flask API로 요청합니다. Flask API에서는 이 URL을 기반으로 이미지의 임베딩 벡터를 계산하고, 이를 MongoDB에서 사용자가 업로드했던 이미지들의 임베딩 벡터들과 비교하여 유사도를 계산합니다. 이 과정에서 가장 높은 유사도를 가진 이미지를 찾아, 이 이미지의 URL을 반환합니다. 반환된 URL은

Backend에서 Frontend로 전달되어, Frontend에서는 이 URL의 이미지를 사용자에게 보여줍니다.

이 기능을 통해 사용자는 특정 얼굴 사진을 업로드하면, 그와 유사한 얼굴을 가진 이미지를 쉽게 찾을 수 있습니다. 이는 얼굴 인식, 얼굴 임베딩, 이미지 유사도 계산 등의 AI 기술을 활용하여 구현되었습니다.

## 후속 개발 예정

### 지속적인 재학습

- 고객 데이터셋 확보
  - 고객의 동의를 받아 수집한 데이터를 학습에 활용 ( 맞추지 못한 데이터 위주)
- 주기적 모델 재학습을 통한 일반화 성능 향상
  - ex) 고객 평가의 잃은 별점 n개 누적시마다 재학습
- 모델 경량화(Pruning, Quantization)를 통한 빠른 Inference

### 클라우드 연동 & 배포

- 클라우드를 연동하여 고객이 일일히 사진을 업로드 하는 불편을 제거
- Docker 활용한 환경관리 & 배포

### 애플리케이션 서비스 개발

- 웹서비스 → 애플리케이션 서비스 전환을 통한 사용자의 편의성 향상
- IOS, Android 어플 개발

### 기능 확장

- 인물 + 사물, 동물 등의 검색 기능 확장
- Chat-GPT API 연결하여 추천 분류 태그 얻기
- Fine-Tuning model 활용하여 추가기능 확장(주민등록증 사진 신뢰도, 보정 정도 확인 서비스)

## Reference

---

1. <https://github.com/deepinsight/insightface>
2. <https://github.com/paul-pias/Face-Recognition>
3. <https://github.com/1adrianb/unsupervised-face-representation>
4. <https://github.com/ronghuaiyang/arcface-pytorch/tree/master>

# 개인 회고(김우진)

## 1. 학습목표 달성을 위해 노력한 것

이번 프로젝트에서 제가 주로 학습하고자 했던 부분은 백엔드 개발, MongoDB, 그리고 처음으로 도전한 AI 모델 **product serving** 였습니다. 따라서, 공식 문서와 관련 온라인 자료를 통해 MongoDB와 Node.js, Express에 대해 깊게 학습하였습니다. 실제 코드 작성에서 발생하는 문제점을 해결하기 위해 Stack Overflow와 같은 커뮤니티에서도 도움을 얻으려 노력하였습니다.

## 2. 프로젝트를 개선 시킨 방법과 시도

사용자의 보안과 편의성을 높이기 위해 bcrypt와 passport 라이브러리를 도입하여 인증 시스템을 개선하였습니다. 또한, 세션 데이터를 MongoDB에 저장하기 위해 connect-mongodb-session 라이브러리를 활용함으로써, 서버가 재시작되거나 여러 서버 인스턴스 간에 세션을 공유할 수 있게 하였습니다.

## 3. 마주한 한계와 아쉬웠던 점

이번 프로젝트에서 가장 큰 어려움은 AI 모델을 웹 서버에 적용하고 그 결과를 처리하여 반환하는 부분이었습니다. 이 과정에서 다양한 어려움을 겪었으나, 그만큼 많은 것을 배울 수 있었습니다. 또한, 백엔드 개발의 복잡성과 어려움을 깨닫게 되었습니다. 특히, 비동기 처리와 에러 핸들링, 서버와 클라이언트 간의 통신에 대한 깊은 이해가 필요했다는 것을 알게 되었습니다. 시간 부족으로 홈페이지 기능 구현을 끝까지 완성 못 시켰던 점이 아쉬웠습니다.

## 4. 프로젝트를 하면서 느낀점

백엔드 개발과 AI 모델 서빙에 대해 많이 공부할 수 있었습니다. 처음 접하는 개념과 기술이 많았지만, 그만큼 새로운 학습의 기회가 있었다고 생각합니다. 그리고, 이러한 과정을 통해 개발은 팀원들과의 협업과 의사소통이 중요하다는 것을 다시 한번 깨닫게 되었습니다.

## 5. 다음 프로젝트에서 시도해 볼 점

다음 프로젝트에서는 초기 설계 단계에 더 많은 시간을 투자하려고 합니다. 그리고 이번에 어려움을 겪었던 모델 서빙 부분에 대해 더 깊이 있게 학습하고, 이를 더욱 효율적으로 구현할 방법을 찾아보려고 합니다. 또한, 백엔드 개발에 있어서 중요한 비동기 처리와 에러 핸들링에 대해 더 깊게 학습하고, 이를 프로젝트에 적용해보고 싶습니다. 마지막으로, 코드 리뷰와 피드백을 통해 끊임없이 코드를 개선해나가는 경험을 해보고 싶습니다.

## 개인 회고(신건희)

### 1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

프로젝트와 관련된 논문과 정리된 글을 보면서 흐름을 파악하고, 방향성을 잡았다. 그리고 방향성에 맞춰서 코드 구현과 모델링, fine-tuning을 진행했다.

### 2. 나는 어떤 방식으로 모델을 개선했는가?

우리 task에 맞는 데이터셋을 찾고, Domain Generalization을 위해 적절한 데이터셋으로 Finetuning을 진행했다. 그리고 Finetuning에 필요한 평가 메트릭을 찾아서 구현하고, 그 메트릭을 기준으로 목표한 정확도에 맞도록 지속하여 모델을 개선할 수 있었다.

### 3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

기존 대회들은 어느정도 Baseline과 문제 가이드라인이 주어진 상태에서 진행하여 그 흐름만 파악하고 있을 뿐 다 알지는 못했지만, 최종프로젝트를 거치면서 직접 Dataset을 구하고, 코드를 짜면서 Fine-tuning 파이프라인을 구현해보니, 확실히 와닿았다.

그리고 직접 자료조사와 논문 공부하고 난 후, 근거를 바탕으로 코드를 짜보니, 이해하는 깊이와 코드를 짜는 재미가 증가했다.

### 4. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

확실히 논문을 읽을 때 중요한 것은 논문이 말하고자 하는 방향성도 있지만, 읽기 위해서 필요한 배경의 흐름을 파악하고 있어야 한다는 것을 다시한번 느꼈다. 처음에 급하게 얼굴 인식 모델의 최신 동향위주로 공부하려 했는데, 그렇게 하다 보니 놓치는 개념이 너무 많았다.

Domain Generalization도 여러 기법이 있었고, 특히 Domain adaptation쪽으로 더 공부하면서 구현해 보고 싶었는데 아직까지 코드 구현 실력이 부족하고, 시간도 없어서 실행으로 옮기지 못했다.

추가로 Metric Learning과 관련하여 다양하게 Fine-tuning을 해보고 싶었는데, Loss 구현도 어렵고 수학적으로 모델에 미치는 영향을 완전히 이해하지 못하여 아쉽게도 Arc Margin Loss로 고정하고 진행하였다.

### 5. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

이렇게 end to end로 경험하다 보니, 어느정도 그림이 잡혀서 해보고 싶은 것들이 마구 생겨났다. 특히 모델링 쪽으로 확장을 해본다면, 앞서 말한 Domain Generalization 쪽으로 요즘 활발히 연구가 진행되고 있는데, 그 부분의 스킬들을 다양하게 시도해 보면서 구현 실력도 늘리고 성능 차이도 확인해 보고 싶다. 또, Loss도 아직 Triplet Loss 부분을 못해 봤는데, Triplet Loss 데이터셋도 구현하여 Fine-tuning 해보고 싶다.

그리고 아직 프로젝트 서빙쪽 이해도가 낮은데, 굉장히 재밌어 보여서 다음 프로젝트에서는 하나하나 직접 서빙까지 진행하고 싶다.

## 개인 회고(신중현)

### 1. 학습목표 달성을 위해 노력한 것

관련 주제에 대해서 survey하고, face recognitino에 관한 유명한 논문 Facenet, Arcface, 경량화된 모델들, mobilenet, ghostnet 등을 읽으면서 모델적인 부분에서 어떻게 접근해야 할지 loss를 어떻게 설정해야 할지 방향성을 찾았고, 다른 팀원들에게 도움이 되도록 front end 와 docker 공부를 추가적으로 더 했던 것 같다.

### 2. 개선 시킨 방법과 시도

사실상 모델구조를 구체적으로 바꾸거나 그렇게 하지는 못하였고, 우리에게 적합하다고 생각하는 평가 dataset을 만들어서, loss function을 조절하거나, 하이퍼 파라미터 튜닝을 하며 모델의 성능을 올렸던 것 같다.

### 3. 마주한 한계와 아쉬웠던 점

domain adaptation에 해당하는 방법이 여러가지가 있는데, 그중 유명한 Domain adversarial training neural network 방법을 적용하지 못한 것이 아쉽다. 그리고 경량화된 모델을 적용시키니 성능이 나와주지 않아서, 다른 task인 경우에 어떻게 성능을 높이는 방법을 찾지 못한 것이 아쉽다.

### 4. 프로젝트를 하면서 느낀점

product serving까지 하면서, 사실 모델을 잘 만드는 것도 중요하지만 다른 더 큰 부분들이 많다는 것을 알게되었다. dataset의 중요성, 평가 metric, 어떻게 지속적으로 학습 시킬 것인지. test domain에 맞는 dataset과 평가는 어떤 것일지, 서버에 저장하는 데이터는 어떤 형식으로 할지, docker를 써서 동일한 환경 설정을 하고 배포를 하는 것 등등 모델 뒤에 필요한 수많은 것들이 존재한다고 생각이 들었다.

### 5. 다음 프로젝트에서 시도해 볼 점

이번 프로젝트에서 배운 것을 바탕으로 feature embedding에 더 관심이 생겼고, information extraction하는 방법을 조금 더 구체적으로 공부해서 프로젝트를 하나 만들 것이다. 내가 처음부터 끝까지 완성한, 그리고 누구나 쉽게 그 환경을 세팅할 수 있는 프로젝트를 만들고 싶다고 생각했다.

## 개인 회고(이종휘)

### 1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

모델선정 : 사진 앨범 분류가 실시간 수준의 성능을 요구하는 서비스는 아니라고 판단했고, 모델들의 추론속도가 느리지 않다고 판단하여 모델의 성능을 기준으로 VGGFACE-2 데이터셋으로 학습된 IResNet50모델을 선정했다.

**Fine-Tuning** : 모델을 학습시킨 VGGFace2 데이터셋이 대부분 백인, 흑인이었는데 서비스의 타겟인 한국인과 분포가 다르다고 판단하여 이 문제를 해결하기 위해 한국인 연예인 데이터셋을 직접 구축하여 Fine-Tuning하는 과정을 거쳐서 성능을 끌어올렸다.

### 2. 나는 어떤 방식으로 모델을 개선했는가?

Fine-Tuning은 Learning rate, Optimizer, input size, layer freeze등의 하이퍼 파라미터를 변경하면서 실험하여 가장 성능이 높은 결과를 활용했다.

### 3. 마주한 한계와 아쉬웠던 점

얼굴 데이터셋을 얻는 과정이 어려웠고, 직접 구축한 데이터셋의 수가 부족했을 수도 있을 거라는 생각이 든다. 더 많은 데이터셋을 하고 Data Augmentation을 적용하여 경향을 확인했다면 좋았을 것 같다.

학습데이터와 타겟 데이터가 다른 문제를 해결하기 위해 Domain-Generalization이나 Domain-Adaptation의 방식으로 접근하여 이 방식들을 공부하고 새로운 해결 방안들을 적용해 보았다면 좋았을 것 같다.

### 4. 프로젝트를 하면서 느낀 점

데이터셋을 구하거나 제작하는 과정이 중요하고 어려운 과정이라는 사실을 알게 되었고, 프로덕트 서빙을 위해 사용할 도구들을 선정하는 과정도 시간이 많이 들었는데, 프로덕트 서빙의 지식이 부족해서 시간이 많이 걸린 것 같다.

### 5. 다음 프로젝트에서 시도해 볼 점

문제를 꼼꼼히 분석하여 정의하고 이 문제를 해결하기 위한 방법은 어떤 것들이 있는지 공부하여 많은 방법들을 적용시켜서 결과를 확인해 보면 좋을 것 같다.

모델링뿐만 아니라 프로덕트 서빙의 역할도 알아서 경험해보면 좋을 것 같다.