

페이스 인식 기능을 활용한 감정분석 기능

[최종보고서]



2023.12.19

인하대학교 컴퓨터공학과

캡스톤 설계 - 005분반

지도 교수 : 신병석

팀명:CODETRIO

팀원: 전예준 12181676, 김성아 12181581,최선아 12191714

목차

1. 서론

1.1. 목적

1.2. 개요

1.3. 개발 환경

2. 개발 내용

2.1 시스템 설계

2.2. AI

2.2.1. face detection model

2.2.2. face emotion recognition model

2.2.3. REST API

2.2.4. Dataset

2.3. Backend

2.3.1. Module 개발

2.3.2. DB 개발

2.4. Frontend

2.4.1. 이미지 샘플, 문구 제시

2.4.2. 사용자 얼굴 촬영

2.4.3. 사용자 실제 감정 선택

2.4.4. 사용자 얼굴 사진, 사용자 실제 감정 서버에 전송(병렬화)

2.4.5. 결과 출력

3. 개발 결과물

3.1. 결과물 사진

3.2. 시연영상 링크

4. 결론

5. 참고 문헌

1. 서론

1.1. 목적

본 문서는 페이스 인식 기능을 활용한 감정분석 기능에 대한 최종보고서이다.

이하대학교 컴퓨터 공학과 'CODETRIO' 팀의 프로젝트를 위한 것으로, 이를 바탕으로 시스템을 설계 및 구현한다.

1.2. 개요

본 연구에서는 FER 기술을 활용하여 얼굴 표정에서 감정이 어떻게 드러나는지에 대한 연구를 진행하고자 한다. 특히, 어떠한 상황에서 표정이 더 정확하게 감정을 전달하는지, 그리고 실제로 느끼는 감정과 얼굴 표정을 통해 인식된 감정 간의 일치도를 조사함으로써, 감정 인식 기술의 정확성과 유효성에 대한 통찰을 얻고자 한다.

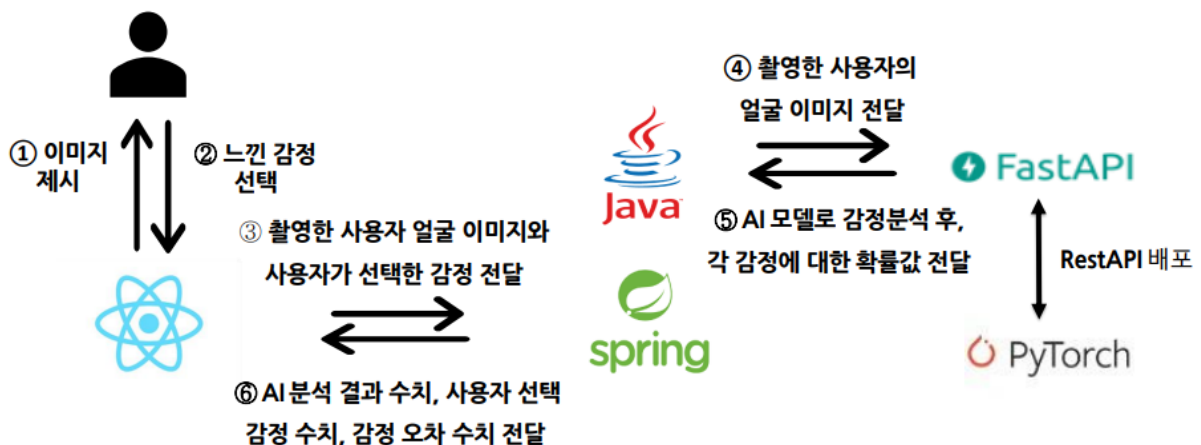
1.3. 개발 환경

- OS : macOS 13.4, window11

- 개발 도구 : Google Colab, VS code, IntelliJ
- 개발 언어: Python (Pytorch), React, Typescript, Java
- 데이터베이스: MySQL
- 개발 기간 : 3개월
- 개발 인원 : 3명

2. 개발 내용

2.1. System 설계



2.2. ai

2.2.1. face detection model

Yolov5 사용 (<https://github.com/ultralytics/yolov5>)

library
yaml
glob
IPython.display
os

model	description
yolov5	yolov5모 델을 활용하여 사람만 인식하도록 학습함

yolov5 modify	description
data.yaml	데이터 정보를 담고있는 파일
data['nc']=1	클래스 수를 하나로 변경(nc=number of class)함
data['names']='person'	클래스 이름을 담고있는 배열에 person만 남김
train(이미지,라벨) 폴더	라벨이 사람인 train데이터셋만 남김
detect.py	모델 로드 및 데이터셋 결과출력 함수

largest_box_index= torch.argmax(torch.tensor([(box[2] - box[0]) * (box[3] - box[1]) for box in det]))	이미지마다 인식된 사람이 여러명 있을 수 있으므로 그 중 가장 크게 인식된 얼굴의 바운딩 박스 인덱스를 찾기 위해 사용함
if idx != largest_box_index: continue	해당 인덱스가 가장 큰 바운딩 박스일 때만 저장하기 위해 사용함

개발 내용

1. yolov5모델을 클론한 후 사람만 인식하도록 클래스 수와 클래스 이름을 변경함
2. 모든 train데이터셋과 valid데이터셋에 대해 라벨이 'person'인 데이터셋만
남기고 나머지는 삭제함
3. yolov5의 train.py를 이용해 batch=16, epochs=5으로 하여 학습시킴

2.2.2. face emotion recognition model

2.2.2.1. face emotion recognition dataset info

library
os
torch.utils.data.DataLoader

model
ResNet32(3,3,3,3,3)

class	description
CustomTrainDataset	Yolov5를 통해 크롭된 이미지들 데이터셋으로 만들기 위해 사용
function	description
__init__	이미지 파일, 이미지 경로, transform정의

__len__	이미지 개수 반환
__getitem__	이미지를 가져와 전처리한 후 이미지와 라벨 반환
LABELS	감정 별 라벨, 값 저장
datasets	CustomTrainDataset에서 나온 데이터셋 배열로 저장
DataLoader	데이터를 배치단위로 나누기 위해 사용
train_loader	train데이터셋 저장
valid_loader	valid데이터셋 저장

2.2.2.1. face emotion recognition model info

class	description
BasicBlock	Yolov5를 통해 크롭된 이미지들 데이터셋으로 만들기 위해 사용
function	description
__init__	Convolution, batch normalization, optimizer 등 함수 정의
forward	Layer마다 수행할 함수 정의 <ul style="list-style-type: none"> 1. Convolution 2. Batch normalization 3. Relu(optimizer) 4. Convolution 5. Batch normalization (Option) downsampler
class	description
ResNet	Resnet모델 정의
function	description
__init__	Convolution, batch normalization, optimizer, pooling, layer 등 정의
_make_layer	Down sampling 필요할 경우 downsample

	layer생성, 레이어 배열 생성
forward	모델의 학습 순서 정의

2.2.2.3. face emotion recognition model train

1) image crop

variable	description
label=['기쁨','당황','분노','불안','슬픔','중립']	감정 라벨을 저장 하는 배열, 라벨 별로 데이터셋이 저장되어있기 때문에 파일 경로를 지정할 때 사용
folder_path	이미지 데이터셋을 저장하고 있는 폴더
crop_folder_path	크롭된 이미지 데이터셋을 저장하고 있는 폴더

개발 내용

1. 감정 라벨을 label 배열에 저장함
2. 각 라벨별 데이터셋의 경로를 가져와 folder_path변수에 할당하고 yolov5.py의 detect.py를 이용하여 이미지를 크롭하고 저장함

2) face emotion recognition model train

variable	description
batch_size=128	배치 사이즈 정의
dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True)	dataset을 미니배치 단위로 만들어 저장
num_class = 6	감정 클래스 수 정의
model = ResNet	사용할 모델 정의
criterion = nn.CrossEntropyLoss	교차 엔트로피 손실함수 사용

optimizer = Adam(model.parameters(), lr=0.001)	Adam 옵티마이저 함수 사용
num_epochs =20	epoch 수 정의
device = torch.device('cuda')	gpu 사용
total_loss	epoch마다 총 오차를 저장
output	모델 output값 저장

개발 내용

1. num_epoch만큼 Resnet모델 학습함
2. optimizer.step()함수를 통해 가중치를 업데이트 함
3. 평균 오차를 구함

3) face emotion recognition model valid

variable	description
correct_predictions	예측값과 라벨이 일치하는 데이터셋 수 저장
total_samples	검증 데이터셋 수 저장
accuracy	전체 이미지 중 라벨을 정확하게 예측한 비율

개발 내용

1. 데이터 셋을 모델에 입력 후 출력 값을 클래스별 확률로 받음
2. 확률이 가장 큰 값을 predicted 변수에 저장함
3. predicted와 라벨이 일치하는 데이터셋 수를 correct_predictions변수에 저장함
4. correct_predictions변수를 valid데이터셋 수로 나눈 후 accuracy 변수에 저장 후

출력함

2.2.2.4. 평가지표: confusion matrix - accuracy

confusion matrix: 모델의 성능을 평가할 때 사용되는 지표로 예측값이 실제 관측값을 얼마나 정확히 예측했는지 보여주는 행렬임

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

accuracy: 모델이 입력된 데이터에 대해 얼마나 정확하게 예측하는지를 나타냄

정확도 = 예측값결과와 실제값이 동일한 건수 / 전체 데이터수

$$= (TP+TN) / (TP+TN+FN+FP)$$

2.2.3. REST API

2.2.3.1. classify_face 함수

library
torch
torch.nn
torchvision

variable	description
model	Resnet32 모델
transform	이미지 전처리를 위한 변환 정의
output	model 출력값 저장

classify_faces 함수 개발 내용

1. 이미지 resize, 텐서 변환 등 이미지 전처리 과정을 transform에 저장함
2. 입력 이미지에 대해 모델을 통과시켜 예측값을 계산한 후 각 클래스에 대한 확률값을 output에 저장함
3. output을 반환함

2.2.3.3. detect_classify 함수

library
flask
io
PIL

variable	description
image_file	클라이언트로부터 받은 이미지파일 저장
image_byte	이미지를 바이트 형식으로 저장
image	Image.open을 사용하여 jpg 형식으로 저장
predictions	classify_faces 모델의 출력값 저장

detect_classify 함수 개발 내용

1. 클라이언트로부터 전송된 이미지 파일을 받아 image_file에 저장함
2. 이미지 파일에서 바이트 데이터를 읽어와 image_bytes에 저장함

3. PIL을 사용하여 바이트 데이터를 이미지로 엮
4. detect_crop_face 함수를 호출하여 이미지 데이터에서 얼굴 이미지를 추출함
5. classify_face 함수를 호출하여 얼굴을 분류하고 예측값을 얻음
6. 결과를 JSON형식으로 반환함

2.2.4. Dataset 소개

한국인 감정인식을 위한 복합영상

본 데이터셋은 한국인의 '안면이미지'를 통해 '감정인식' AI 학습용 모델 구축의 목적으로 구성되며 특정 장소/배경, 특정 감정이 나타난 '1인 셀프카메라(Selfie)' 50만장으로 구축되어 있음. AI 학습 모델의 정확도 증대를 위해, 특정 감정을 잘 표현할 수 있는 '전문인'을 별도 모집하여 25만장을 수집, 자연스러운 일반인의 표정을 묘사할 수 있는 '일반인'을 따로 모집하여 25만장을 수집하였으며, 이는 데이터의 포괄성과 품질에 대한 신뢰성을 확보할 수 있도록 구성되어 있다고 할 수 있음. 7가지로 감정을 분류하였는데 감정체계를 분류하는 기준은 Susan David의 감정분류체계에 따랐으며, 현재 세계적인 감정 이미지를 활용한 학습용 데이터 구축 흐름에 발맞추어 '중립(무표정)' 감정을 별도 추가하였음

Angry	Sad	Anxious	Hurt	Embarrassed	Happy
Grumpy	Disappointed	Afraid	Jealous	Isolated	Thankful
Frustrated	Mournful	Stressed	Betrayed	Self-conscious	Trusting
Annoyed	Regretful	Vulnerable	Isolated	Lonely	Comfortable
Defensive	Depressed	Confused	Shocked	Inferior	Content
Spiteful	Paralyzed	Bewildered	Deprived	Guilty	Excited
Impatient	Pessimistic	Skeptical	Victimized	Ashamed	Relaxed
Disgusted	Tearful	Worried	Aggrieved	Repugnant	Relieved
Offended	Dismayed	Cautious	Tormented	Pathetic	Elated
Irritated	Disillusioned	Nervous	Abandoned	Confused	Confident

<출처: AI Hub 한국인 감정인식을 위한 복합 영상 데이터 설명서>

표 | 감정 분류체계 7종

No.	대감정	세부감정
1	분노	툼툼대는, 좌절한, 짜증내는, 방어적인, 악의적인, 안달하는, 구역질 나는, 노여워하는, 성가신
2	슬픔	실망한, 비통한, 후회되는, 우울한, 마비된, 염세적인, 눈물나는, 낭패한, 환멸을 느끼는
3	불안	두려운, 스트레스 받는, 취약한, 헛갈리는, 당혹스러운, 회의적인, 걱정스러운, 조심스러운, 신경쓰이는
4	상처	질투하는, 배신당한, 격리된, 충격 받은, 궁핍한, 희생된, 억울한, 괴로워하는, 버려진
5	당황	격리된, 시선 의식하는, 외로운, 열등한, 죄책감의, 부끄러운, 혐오스러운, 한심한, 헛갈리는
6	기쁨	감사하는, 믿는, 편안한, 만족한, 흥분한, 느긋한, 안도하는, 신이 난, 자신하는
7	중립	무표정, 감정이 0인 상태

<출처: AI Hub 한국인 감정인식을 위한 복합 영상 데이터 설명서>

카테고리(종)	이미지 개수(개)	분포(%)
기쁨	70,735	14.47%
당황	70,457	14.41%
분노	68,835	14.08%
불안	69,965	14.31%
상처	70,103	14.34%
슬픔	70,508	14.42%
중립	68,173	13.94%
평균	69,825	14.28

<출처: AI Hub 한국인 감정인식을 위한 복합 영상 데이터 설명서>

데이터 셋 예시

jpg, jpeg 형태 이미지 약 50만 장

(1) 인물의 감정 상태 포함

(2) 다양한 장소/배경 포함



2.3. 백엔드

2.3.1. Module 개발

1) EmotionTest Package

전반적인 테스트 과정을 진행하는 작업을 수행

EmotionTestController	
프론트단에서 받은 전반적인 테스트 관련 요청을 처리	
emotionTestService	객체를 생성자를 통해 주입
sampleService	
feedbackService	
resultService	
testStart()	감정 분석 정보를 관리하기 위한 새로운 테스트를 등록함
testResult()	테스트 결과를 종합해서 프론트단으로 보냄

EmotionTestService	
테스트 진행 관련 핵심 비즈니스 로직을 처리	
emotionTestRepository	객체를 생성자를 통해 주입
sampleRepository	

feedbackRepository	
resultRepository	
generateTest()	새 테스트를 생성하고, DB에 저장
setTestSample()	테스트 엔티티에 감정 유발 샘플을 등록
getTest()	테스트 id를 통해 테스트 엔티티를 반환
getImage()	이미지 경로를 받아 저장된 이미지 파일을 찾아서 Byte로 반환
calculateError()	테스트 id를 받아 AI 분석 결과와 사용자 의견 간의 오차를 반환
updateFaceUrl()	DB에 저장된 테스트에 대표사진 경로 추가

EmotionTestRepository	
테스트 정보를 담는 DB에 접근	
save()	새로운 테스트 정보를 DB에 추가 후 부여한 테스트 번호를 반환
findOne(id)	테스트 번호를 이용해 전반적인 테스트 정보를 반환

2) Result Package

테스트 과정에서 AI 측과 통신하는 데이터를 관리하는 작업을 수행

ResultController	
표정 분석 관련 요청을 처리	
resultService	객체를 생성자를 통해 주입
emotionTestService	
receivePhoto()	웹에서 받은 사진을 저장하고 AI 측으로 전송한 뒤, 분석 결과 처리

ResultService

표정 분석 관련 핵심 비즈니스 로직을 처리	
resultRepository	객체를 생성자를 통해 주입
emotionTestRepository	
savePhoto()	웹캠으로 촬영한 BASE64로 인코딩 된 5장의 사진 파일을 서버 내부 저장소에 저장
sendPhoto()	촬영한 5장의 사진을 analyzeImage()를 통해 전송한 후 결과값을 DB에 저장
analyzeImage()	사진 파일을 AI 서버에 전송하고 결과를 받아 처리
selectCentralSequence()	촬영한 5장의 사진에 대해 감정 값의 편차 제곱합의 수치가 가장 적은 사진을 선택

ResultRepository 표정 분석 정보를 담는 DB에 접근	
save()	AI 분석 결과를 DB에 저장
findByTestId()	테스트 번호를 이용하여 감정 분석 결과를 반환

3) Sample Package

테스트에 사용할 샘플을 관리하는 작업을 수행

SampleService 감정 추출용 샘플에 대한 비즈니스 로직을 처리	
sampleRepository	객체를 생성자를 통해 주입
randomSample()	테스트에 사용할 샘플을 무작위로 결정함
getSampleImage()	샘플 이미지의 경로를 반환
getSampleComment()	샘플 문구를 반환

SampleRepository

샘플 정보를 담는 DB에 접근	
findOne()	샘플의 id값을 이용하여 해당 샘플의 데이터를 반환
getSampleCount()	총 샘플의 개수를 반환

4) Feedback Package

테스트 도중 받는 사용자의 피드백을 관리하는 작업을 수행

FeedbackController 사용자 피드백에 대한 요청을 처리	
feedbackService	객체를 생성자를 통해 주입
saveFeedback()	해당 테스트 id에 대한 사용자의 감정 피드백 값을 확률값으로 변환 후, DB에 저장

FeedbackService 사용자 피드백에 대한 비즈니스 로직을 처리	
feedbackRepository	객체를 생성자를 통해 주입
saveFeedback()	사용자가 느낀 감정을 확률값으로 변환한 후 저장
getFeedback()	테스트의 사용자의 의견을 DB에서 찾아 DTO로 반환
convertToDecimal()	사용자가 보낸 Neutral을 제외한 5가지의 감정 수치를 가지고 Neutral까지 계산하여 6개의 확률값 수치 6개의 응답 수치(0~5) 중 최대값에 따라서 1~0을 Neutral에 부여하고 나머지는 수치에 비례해서 배분

FeedbackRepository 사용자 피드백 정보를 담는 DB에 접근	
save()	계산된 피드백 수치를 DB에 저장
findByTestId()	테스트 번호를 이용하여 해당 감정 피드백을 반환

2.3.2. DB 개발

1) sample

사용자의 감정이 포함된 사진을 추출하기 위한 샘플의 데이터를 저장할 테이블

sample	
Primary key	id BIGINT
	face_url VARCHAR(256) comment VARCHAR(256)

id : Primary Key 역할, 샘플 식별 번호

img_url : 샘플로 사용할 사진 파일의 경로

comment : 샘플과 같이 띄울 문구.

2) emotion_test

테스트 데이터를 관리하기 위한 테이블

emotion_test	
Primary key	id BIGINT
Foreign Key	sample_id BIGINT face_url VARCHAR(256)

id : Primary Key 역할, 테스트 번호

sample_id : 얼굴 사진 추출에 사용된 샘플 번호, 1)의 id를 참조함.

face_url : 샘플을 활용하여 촬영한 얼굴 사진이 찍힌 파일의 경로

3) result

페이스 분석으로 나온 테스트 결과물을 저장할 테이블

result	
Primary key	id BIGINT
Foreign Key	angry DECIMAL(4,3) fear DECIMAL(4,3) happy DECIMAL(4,3) sad DECIMAL(4,3) surprise DECIMAL(4,3) neutral DECIMAL(4,3)

id : Primary Key 역할, 2)의 id 참조

angry, fear, happy, sad, surprise, neutral : AI로부터 받은 6가지 감정 수치,
0~1 사이의 확률 값으로 표현됨

4) feedback

사용자의 피드백을 통해 받은 감정값

feedback	
Primary key	id BIGINT
Foreign Key	angry DECIMAL(4,3) fear DECIMAL(4,3)

	happy DECIMAL(4,3)
	sad DECIMAL(4,3)
	surprise DECIMAL(4,3)
	neutral DECIMAL(4,3)

id : Primary Key 역할 , 2)의 id를 참조함

angry, fear, happy, sad, surprise, neutral : 사용자가 느낀 6가지 감정 수치.

0~1 사이의 확률 값으로 표현됨

2.4. 프론트엔드

함수명 또는 컴포넌트명	
variable	description

2.4.1. 샘플 이미지/문구 제시

전역 상태	
samplesAtom	서버에서 받은 Sample 문구/ 이미지를 저장
setSamples	SamplesAtom을 변화시키는 함수
idsAtom	서버에서 받은 id를 저장
setIds	IdsAtom을 변화시키는 함수

getSamples({setSamples,setIds})	
data	서버로 부터 받은 샘플 데이터

개발 내용

1. 서버의 `api/test/start` 엔드포인트에 GET요청을 보냄
2. 요청 성공 시 서버의 응답 데이터인 `res`를 받고 실제 필요한 데이터인 `res.data.tests`를 `data` 변수에 할당함
3. `data.map`으로 `data`의 요소 하나씩 돌면서 `samplesAtom` 형식에 맞게 각각의 샘플 데이터를 `{id: ..., sampleImg: ..., comment: ...}` 형식으로 가공해줌. 서버에서 URL을 받아오므로 `sampleImg`는 ``data:image/jpeg;base64,${item.sampleImg}``로 가공해줌.
4. `SetSamples` 함수를 사용하여 `samplesAtom`을 가공한 결과로 변화시켜줌
5. `data.map`으로 `data`의 요소를 하나씩 돌면서 각 샘플의 `id`를 추출함. `setIds` 함수를 사용하여 `idsAtom`을 변화시켜줌
6. `Try catch` 문을 사용하여 에러를 잡음
7. `TestSample` 컴포넌트에서 `samplesAtom` 상태값을 사용하여 `sample.comment`와 `sample.sampleImg`로 서버에서 받은 샘플 문구/이미지들을 화면에 나타냄

2.4.2. 사용자 얼굴 촬영

2.4.1.2. 웹캠 띄우기

Webcam 컴포넌트	
Video 태그	웹캠을 나타냄
Canvas 태그	웹캠을 촬영본을 나타냄
videoRef	Video 요소에 접근하게 해줌
canvasRef	Canvas 요소에 접근하게 해줌

StartWebcam()

stream	웹캠 비디오 스트림
videoRef	Video 요소에 접근하게 해줌

개발 내용

1. navigator.mediaDevices.getUserMedia({video:true}) 브라우저의 미디어 디바이스 API를 사용해 사용자의 비디오 스트림을 요청함. GetUserMedia 함수를 호출하여 사용자에게 카메라에 대한 액세스 권한을 요청하고, 성공적으로 허용되면 비디오 스트림이 반환됨
2. videoRef.current.srcObject 에 stream을 할당하여 비디오 요소의 srcObject 속성에 웹캠 비디오 스트림을 할당함. 이를 통해 비디오 요소가 웹캠에서 캡처된 비디오를 표시하게 됨
3. videoRef.current.onloadeddata = () => {vedeoRef.current?.play();};를 하여 웹캠 비디오 데이터가 로드되면 비디오 요소의 play 메서드가 호출되어 웹캠 비디오를 재생하도록 함
4. try catch 문을 이용하여 웹캠을 띄우는 과정에서의 오류를 잡아냄

2.4.2.2. 사용자 얼굴 촬영 타이머

setTimer()	
repeatCount	캡처 횟수
repeatInterval	촬영 간격
lastTimestamp	직전 촬영 초(ms)

setTimeInterval(timestamp)	
elapsedTime	촬영 후 지난 초(ms)
timestamp	현재 초(ms)
lastTimestamp	직전 촬영 초(ms)

repeatCount	캡처 횟수
-------------	-------

개발 내용

1. 웹캠 촬영이 시작되면 setTimer 함수의 repeatCount를 0으로 초기화, repeatInterval을 3000/5(3초 동안 5번 캡처하므로)로 초기화, lastTimestamp를 0으로 초기화 함
2. requestAnimationFrame(setTimeInterval)을 실행하여 콜백함수인 setTimeInterval가 실행될 때 setTimeInterval 함수의 인자로 setTimeInterval 함수의 실행 시점(ms)을 전달함
3. setTimeInterval 함수가 실행되면 인자로 함수의 실행 시점(timestamp)이 들어오고, elapsedTime에 timestamp - lastTimestamp를 할당함
4. 만약 elapsedTime >= repeatInterval 이면 촬영해야 하므로 repeatCount를 1 증가, lastTimestamp에 현재 timestamp를 할당해주고 captureWebcam(사용자 얼굴 촬영 함수)를 실행함
5. 만약 repeatCount가 5 미만이면 촬영 횟수가 남아있으므로 requestAnimationFrame(setTimeInterval)을 똑같이 진행해줌

2.4.2.3. 사용자 얼굴 촬영

Webcam 컴포넌트	
Video 태그	웹캠을 나타냄
Canvas 태그	웹캠을 촬영본을 나타냄
videoRef	Video 요소에 접근하게 해줌
canvasRef	Canvas 요소에 접근하게 해줌

captureWebcam()	
videoRef	Video 요소에 접근하게 해줌

canvasRef	Canvas 요소에 접근하게 해줌
video	Video 요소
canvas	Canvas 요소
context	2D 그래픽 컨텍스트
imageDataUrl	사용자의 얼굴을 촬영한 사진 URL

개발 내용

1. `video = videoRef.current`를 하여 `video` 변수가 Video DOM 요소에 접근할 수 있게 하고, `canvas = canvasRef.current`를 하여 `canvas` 변수가 Canvas DOM 요소에 접근할 수 있게함
2. `context = canvas.getContext("2d")`를 하여 2D 그래픽 컨텍스트를 얻어 캔버스에 그림을 그릴 수 있게 함
3. `canvas.width = video.videoWidth`를 하여 `canvas`의 `width`를 비디오 DOM 요소의 `videowidth`로 설정하고, `canvas.height = video.videoHeight`를 하여 `canvas`의 `height`를 video DOM 요소의 `videoHeight`로 설정함
4. `context.scale(-1,1)`을 통해 X축 방향으로 -1로 스케일을 조정함. 이를 통해 캔버스에 그림을 그릴 때 좌우 반전 효과를 얻을 수 있음
5. `context.drawImage(video, -canvas.width, 0, canvas.width, canvas.height)`를 통해 웹캠 비디오를 캡처한 이미지를 좌우 반전하여 캔버스에 그림
6. `canvas.toDataURL("image/jpeg")`를 통해 캔버스에 그려진 이미지를 JPEG 형식의 데이터 URL로 변환하고 변환한 이미지를 `imageDataUrl`에 저장함
7. `setCapturedImages((prevImages) => [...prevImages, imageDataUrl])`를 통해 방금 `imageDataUrl`에 저장한 이미지를 이미지 목록에 추가함. `SetCapturedImages`는 5장의 사용자 얼굴 사진을 저장하는 전역 상태인 `capturedImagesAtom`을 변화시키는 함수임

2.4.3. 사용자 실제 감정 선택

전역 상태	
detailEmotionAtom	사용자가 선택한 감정을 저장
setDetailEmotion	detailEmotionAtom 을 변화시키는 함수

개발 내용

1. DetailEmotionAtom의 기쁨, 당황, 분노, 불안, 슬픔 6가지 감정에 대해 수치 0으로 초기화함
2. 감정 없음을 선택하면 detailEmotionAtom 6가지 감정의 수치를 다 0으로 할당함
3. 감정 있음을 선택하면 detailEmotionAtom의 6가지 감정에 대해 각각 "매우 조금"을 선택했으면 1, "조금"을 선택했으면 2, "보통"을 선택했으면 3, "많이"를 선택했으면 4, "매우 많이"를 선택했으면 5로 할당해줌
4. 감정은 여러 개 선택할 수 있지만 하나의 감정에 대해서는 하나의 수치만 선택 가능함.
5. 선택했던 감정/수치를 한 번 더 클릭하면 선택 해제됨

2.4.4. 사용자 얼굴 사진/ 사용자 실제 감정 서버에 전송

2.4.4.1. 사용자 얼굴 사진 서버에 전송

인자	
capturedImages	사용자의 얼굴 사진들 저장
id	샘플 id 저장

postCapturedImages({capturedImages,id})	
formdata	이미지 데이터 담기 위한 FormData 객체
blob	Blob 저장
response	요청 성공 시, 서버로 부터 받는 응답

개발 내용

1. Const formData = new FormData()를 통해 이미지 데이터를 담는 FormData 객체를 생성함
2. capturedImages 배열을 forEach로 돌면서 각 사용자 얼굴 이미지를 dataUriToBlob(image.split("")[1]) 함수를 사용해 URI에서 Blob으로 변환한 후, formData.append(`file\${index+1}`, blob)을 하여 `file\${index+1}` 이름으로 FormData에 추가함
3. 서버의 api/test/camera/\${id} 엔드포인트에 POST요청을 보냄. 이미지 데이터는 FormData로 전달되며 요청 헤더는 "Content-Type: multipart/form-data"로 설정함
4. 서버로부터 성공적으로 응답을 받으면 response를 반환함
5. try catch를 통해 에러를 잡아냄

dataUriToBlob(dataURI)	
byteString	Base64 디코딩 데이터(원시 이진 데이터)
arrayBuffer	ArrayBuffer 객체
int8Array	8비트 부호 없는 정수 배열

개발 내용 - URI to Blob

1. URI를 Blob으로 변환하는 dataURItoBlob 함수는 dataURI라는 문자열을 매개변수로 받음
2. `byteString = atob(dataURI)`: atob 함수를 사용하여 base64로 인코딩된 데이터를 디코딩하여 얻은 원시 이진 데이터를 ByteString에 할당함
3. `arrayBuffer = new ArrayBuffer(byteString.length)`: 디코딩된 이진 데이터의 길이를 사용하여 ArrayBuffer를 생성함
4. `init8Array = new Uint8Array(arrayBuffer)`: ArrayBuffer를 기반으로 하는 8비트 부호 없는 정수 배열을 생성함. 이 배열은 데이터를 바이트 단위로 다룰 수 있게 함
5. for문으로 init8Array 배열을 돌면서 `int8Array[i] = byteString.charCodeAt(i)`를 해줌으로써 디코딩된 문자열 byteString의 각 문자의 아스키 코드 값을 int8Array[i]에 할당함
6. Int8Array를 사용하여 Blob을 생성하고 Blob의 MIME 타입을 "image/jpeg"로 설정한 후, Blob을 반환함

2.4.4.2. 사용자 실제 감정 서버에 전송

인자	
id	샘플 id 저장
detailEmotion	사용자가 선택한 감정 저장

PostDetailEmotions({id, detailEmotion})	
response	요청 성공 시, 서버로 부터 받는 응답

개발 내용

1. 서버의 `api/test/feedback?id=${id}` 엔드포인트에 POST 요청을 보냄
2. 요청 본문에 `JSON.stringify(detailEmotion)`을 사용하여 `detailEmotion`을 JSON 형식으로 변환하여 포함시킴
3. 요청 헤더에 JSON 형식의 데이터를 전송하는 것을 명시하기 위해 `"Content-Type": "application/json"`을 설정함
4. 서버로 부터 성공적으로 응답을 받으면 응답을 반환함
5. try catch를 통해 에러를 잡아냄

2.4.4.3. 사용자 얼굴 사진/사용자 실제 감정 서버에 전송 병렬화

사용자 얼굴 사진을 서버에 보내는 것과 사용자의 실제 감정을 서버에 보내는 것은 Promise 작업 간의 순서가 상관이 없음. 따라서 병렬로 Promise를 처리하는 것이 좋아보임.

개발 내용

1. Promise.all의 인자로 `postCapturedImages`와 `postDetailEmotions` 함수를 배열 형태로 주어 두 개의 비동기 함수를 동시에 실행시킨 후 모든 프로미스가 완료될 때 까지 기다리게하고 하나의 프로미스로 반환하게 함

2.4.5. 결과 출력

2.4.5.1. 서버에서 결과 가져오기

인자	
ids	샘플 id들 저장
setResult	ResultAtom(결과 데이터 저장) 상태를

	변화시키는 함수
--	----------

getResults({ids,setResult})	
data	서버로 부터 결과 데이터 저장

개발 내용

1. 서버의 api/test/result 엔드포인트에 POST 요청을 보냄
2. 요청 본문에 ids를 JSON.stringify를 사용하여 JSON 형태로 포함시킴
3. 요청 헤더에는 JSON 형식의 데이터를 전송한다는 것을 명시하기 위해 "Content-Type": "application/json"을 설정함
4. 요청 성공 시 서버의 응답 데이터인 res를 받고 실제 필요한 데이터인 res.data.results를 data 변수에 할당함
5. data를 map으로 돌면서 각 요소를 기존 resultAtom의 type에 맞게 가공함
6. 가공한 데이터를 setResult를 통해 resultAtom 상태를 변화시켜줌
7. try catch를 통해 에러를 잡아냄

2.4.5.2. 결과 화면에 출력하기

ResultStepRadio 컴포넌트	
resultStep	몇 번째 샘플에 대한 결과 tab인지 저장

UserImage 컴포넌트	
resultStep	몇 번째 샘플에 대한 결과 tab인지 저장
image	사용자 얼굴 사진

TestSample 컴포넌트	
resultStep	몇 번째 샘플에 대한 결과 tab인지 저장
image	샘플 사진

Graph 컴포넌트 - 오픈 소스 사용	
resultStep	몇 번째 샘플에 대한 결과 tab인지 저장
results	특정 샘플에 대한 결과 데이터 저장

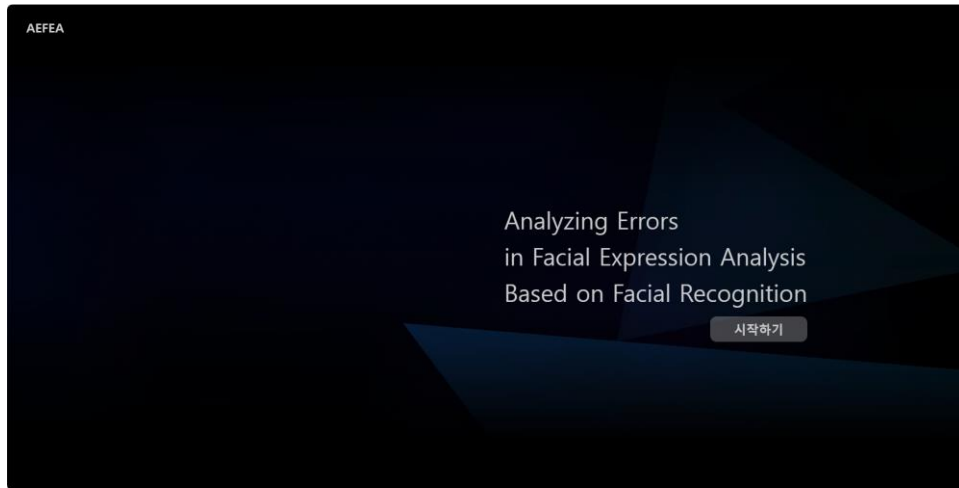
개발 내용

1. ResultStepRadio 컴포넌트에서 특정 step의 결과 탭을 클릭할 수 있는 라디오 버튼을 띄움
2. UserImage 컴포넌트에서 image 변수에 resultsAtom[resultStep-1].faceImg를 할당하고 img 태그를 이용해 사용자의 얼굴 사진을 띄움
3. TestSample 컴포넌트에서 image 변수에 resultsAtom[resultStep-1].sampleImg를 할당하고 img 태그를 이용해 샘플 사진을 띄움
4. Graph 컴포넌트에서 'recharts' 오픈 소스를 사용하여 AI 분석 결과와 사용자의 실제 감정을 그래프로 나타냄

3. 개발 결과물

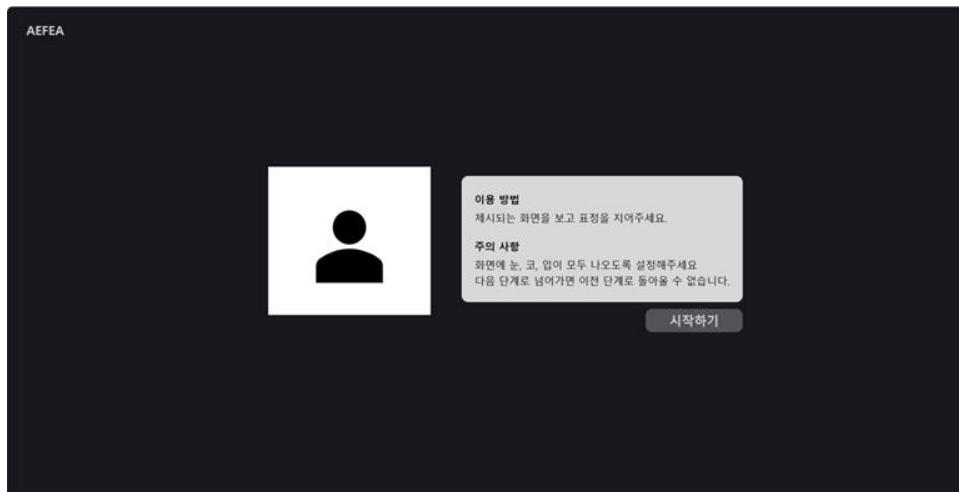
3.1. 결과물 사진

3.1.1. 메인 페이지



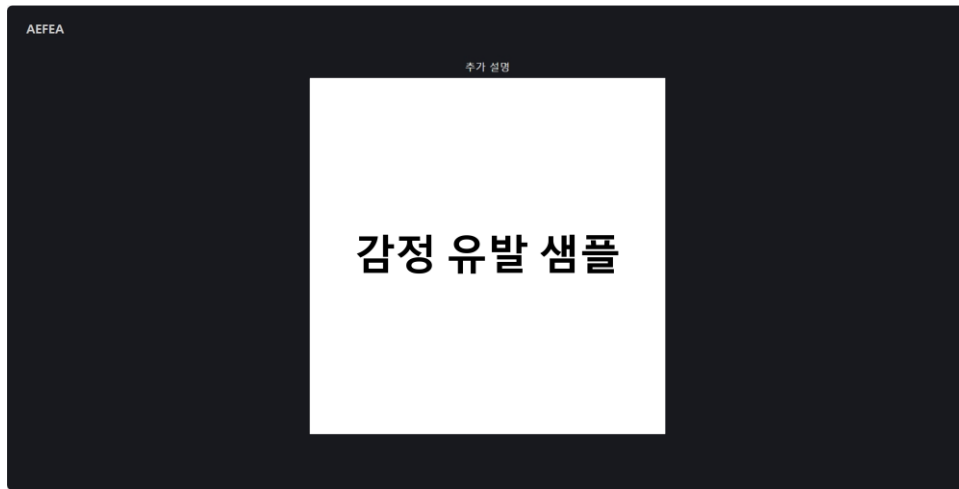
프로젝트를 실행하면 가장 먼저 보게 되는 페이지임. 여기서 '시작하기' 버튼을 누르면 테스트 전 주의사항 확인 및 테스트 환경을 조정하기 위한 테스트 준비 페이지로 이동하면서 감정 분석 테스트를 시작할 수 있음.

3.1.2. 테스트 준비 페이지

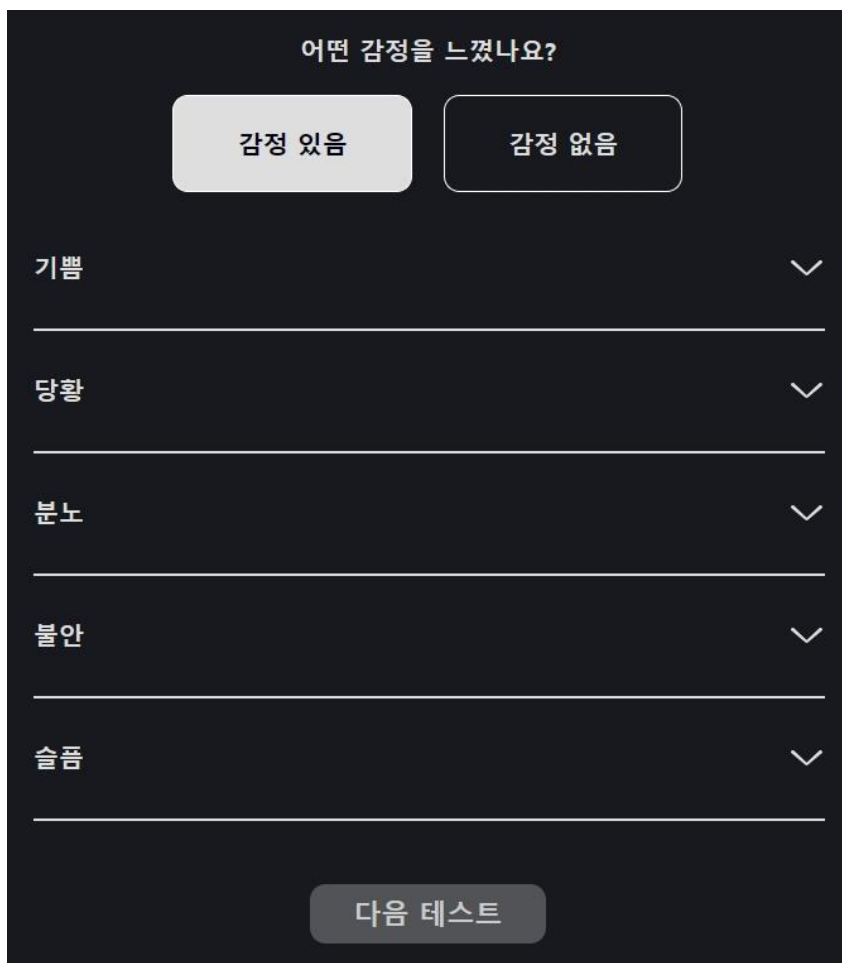


테스트 준비 페이지에서는 카메라로 사용자의 얼굴을 촬영하기 위해 미리 웹캠 화면을 띄워 사진이 잘 촬영되고 있는 지 확인할 수 있음. '시작하기' 버튼을 누르면 테스트 페이지로 이동하면서 WAS에 테스트를 생성하도록 요청을 보냄.

3.1.3. 테스트 페이지



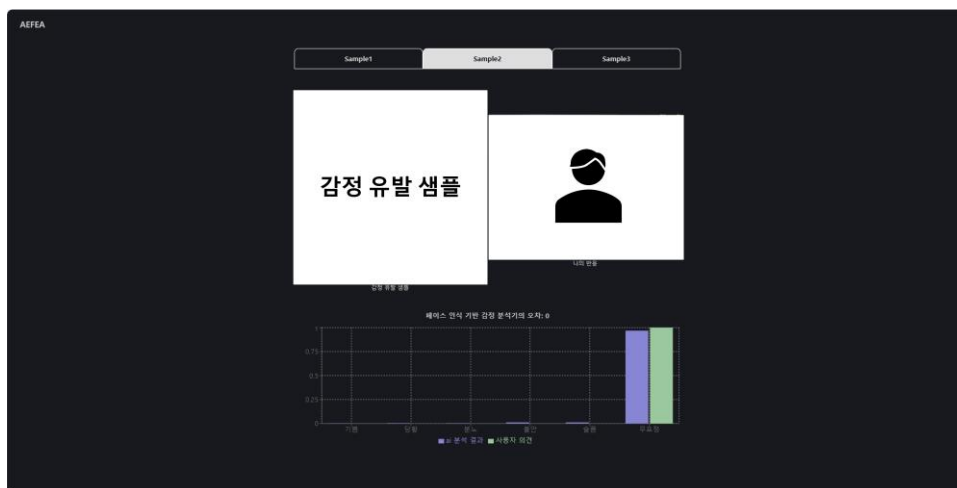
테스트 페이지에서는 감정 유발 샘플을 3초 동안 화면에 표시를 함. 이 3초 동안 연결된 웹캠을 활용해 사용자의 얼굴 사진을 5장 촬영함.



3초가 지난 후, 위의 감정 유발 샘플 밑에 어떤 감정을 느꼈는지 선택할 수 있는

창이 제시됨. 여기서 각 5가지 감정 부분을 누르면 각각 아주 조금, 조금, 보통, 많이, 아주 많이 중 하나를 선택할 수 있음. 테스트는 총 3번 진행하기 때문에 '다음 테스트' 버튼을 눌러 다음 테스트를 진행할 수 있음. 마지막 테스트인 경우 '결과 출력' 버튼을 눌러 결과 출력 페이지로 이동함.

3.1.4. 결과 출력 페이지



결과 출력 페이지에서는 앞서 했던 3개의 테스트 데이터를 DB로부터 받아와서 각 테스트 별로 한 눈에 볼 수 있게 정리해서 화면에 띄움. 각 테스트 결과를 상단의 탭을 통해 이동하면서 볼 수 있으며, 중단에서는 테스트에서 사용된 감정 유발 샘플 사진과 앞서 촬영한 5장의 사진 중 감정 분석 결과의 평균에 가장 근접한 사진을 보여줌. 하단에는 5장의 사진의 AI 모델의 감정 분석 결과의 평균값과 사용자가 느꼈던 감정을 일정한 수식을 통해 확률로 변환한 값을 막대 그래프를 통해 보여주며, 두 값의 전체적인 차이를 표준편차 값으로 보여줌.

3.2. 시연영상 링크

<https://youtu.be/C5-9rM3Zwxc>

4. 결론

본 프로젝트에서는 Fer 데이터셋을 활용하여 학습된 ResNet 모델이 78%의 정확도를 보였으며, 특히 놀라거나 기쁜 감정을 유발하는 사진에서 모델의 정확도가 높게 나오는 것을 확인하였다. 그러나 실제 사용자의 테스트 결과를 통해, 이러한 감정을 유발하는 상황에서 모델의 성능과 사용자의 실제 감정 사이에 차이가 있음을 발견할 수 있었다.

실제 감정을 직접 수집하고 딥러닝 모델의 결과값과 비교함으로써, Fer 모델의 한계를 명확히 이해할 수 있었다. 모델은 특정 감정을 인식하는 데 있어서는 뛰어나지만, 실제 감정과의 정확한 일치를 달성하는 데에는 한계가 있음을 확인했다. 특히, 놀라거나 기쁨과 같은 강렬한 감정을 유발하는 상황에서 모델의 예측이 상대적으로 정확했지만, 다른 감정이나 복잡한 감정의 경우 모델의 성능이 떨어졌다.

이러한 결과로 보아, Fer 모델은 일부 감정에 대해서는 높은 정확도를 보이지만, 감정의 다양성과 실제 감정과의 정확한 일치에는 한계가 있음을 확인할 수 있었다. 향후 연구에서는 모델의 성능을 향상시키기 위해 더 다양한 데이터셋을 활용하거나, 복잡한 감정을 인식하는 데에 더욱 특화된 모델 개발에 초점을 맞추고자 한다.

5. 참고 문헌

[프론트엔드]

<https://recharts.org/en-US/>

<https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/toDataURL>

<https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia>