한국소프트웨어산업협회
실시간 빅데이터분석 및 시각인지 시스템 개발자 양성과정

얼굴 인식을 이용한
특정 인물 탐지 및 피대상자 비식별 조치

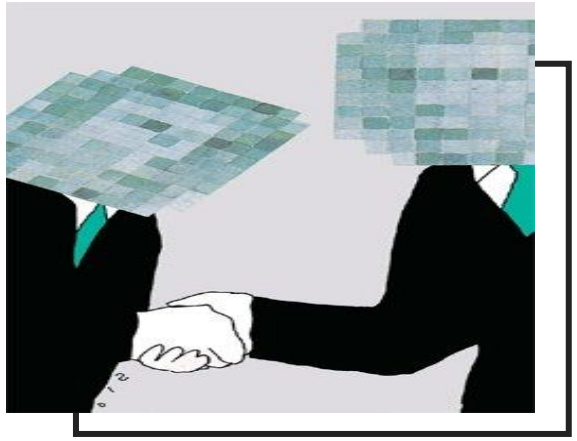
윤다영 김다솜 김다영 원석민

INDEX

프로젝트 개요

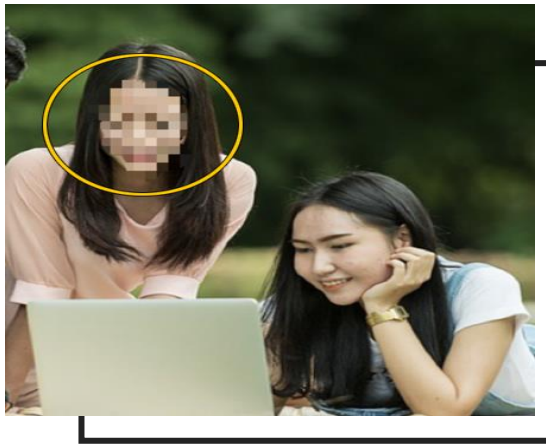
1. 주제 선정 배경
2. 프로젝트 일정
3. 프로젝트 흐름도
4. 개발환경

주제 선정 배경



- **초상권**
  : 본인의 초상이 허가 없이 촬영되거나 공표되지 않을 권리

- 기술 고도화에 시민들 노출 심화
- 1인 생중계 방송, 실시간 생중계 방송 등 초상권이 보호 받지 못함
- 원격, 비대면 수업등의 활성화로 초상권 문제 화제



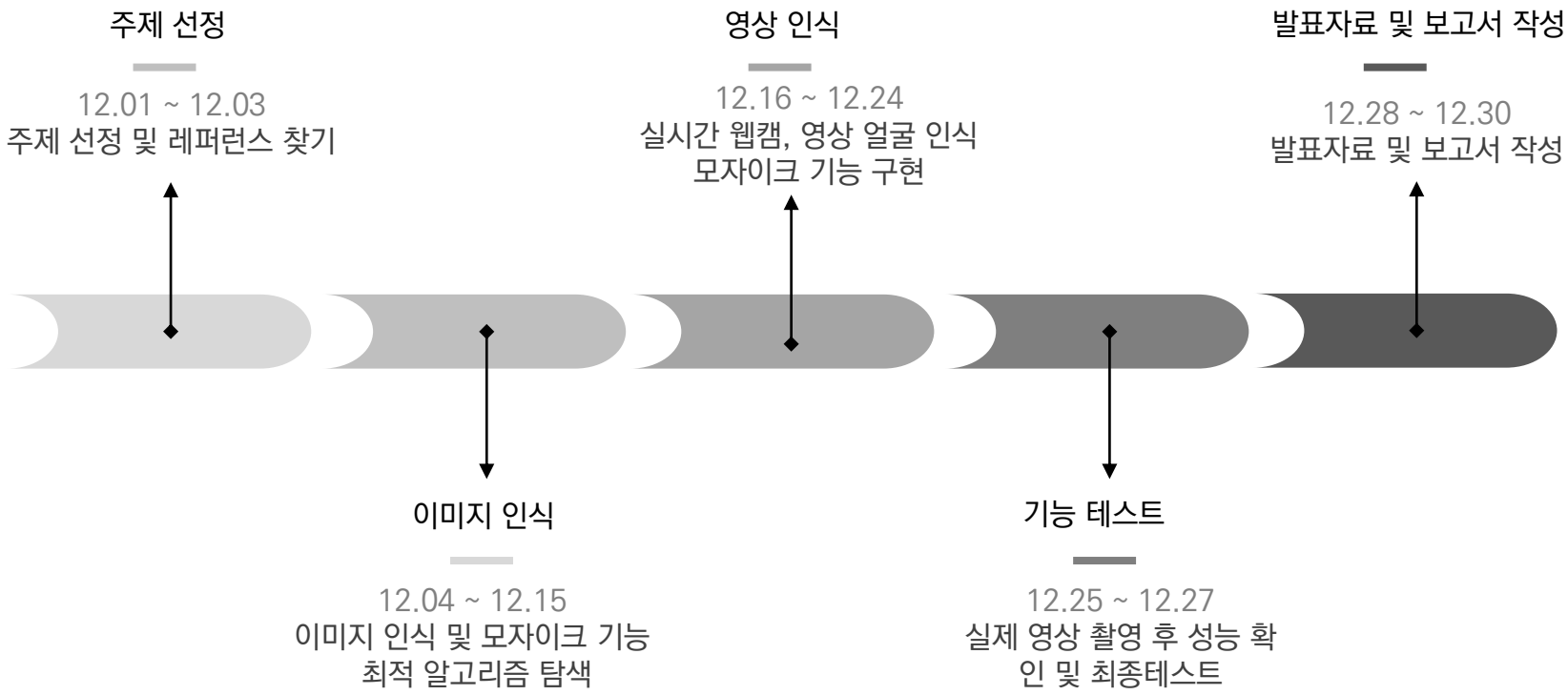
- 컨텐츠 제작자를 위한

  초상권 보호 목적

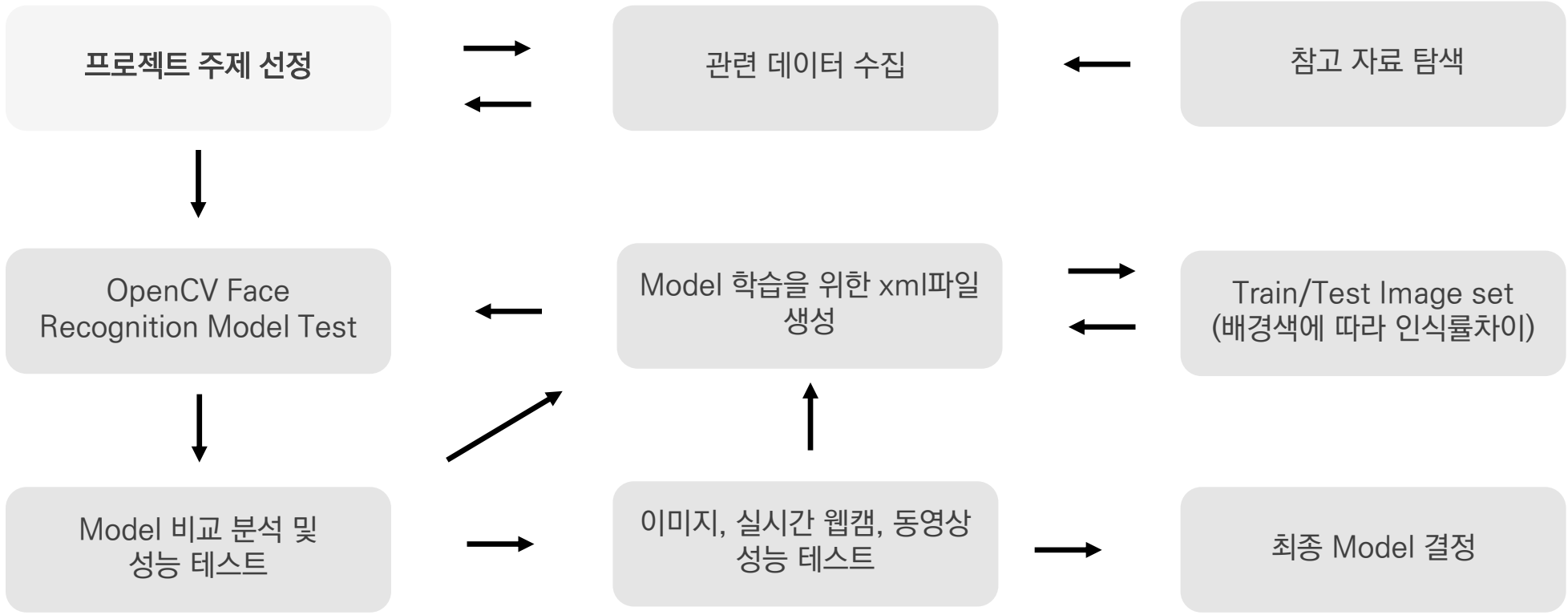
  자동 모자이크 처리 프로그램

프로젝트 일정

목표 : 컨텐츠 제작자를 위한 초상권 보호 목적 자동 모자이크 처리 프로그램 제작



프로젝트 흐름도


프로젝트 주제 선정
관련 데이터 수집
참고 자료 탐색
OpenCV Face Recognition Model Test
Model 학습을 위한 xml파일 생성
Train/Test Image set (배경색에 따라 인식률차이)
Model 비교 분석 및 성능 테스트
이미지, 실시간 웹캠, 동영상 성능 테스트
최종 Model 결정

개발환경

OS

Windows 10
Pro 64bit(x64 processor)

HW

Intel ® Core™
i7-4790 CPU @ 3.60GHz

Dev. Tools

Python 3.8.5

Anaconda 4.10.1

Jupyter Notebook

Google Colab

Python packages

OpenCV 4.5.3

Numpy

Matplotlib 3.5.0

Tensorflow 2.4.0

Keras 2.4.0

ImageFont
ImageDraw
Image

AI Models

Haar_Cascade	cvlib	Face_recognition
Dlib_Hog	MTCNN	
Dlib_ResNet	Pickle	
DNN_caffemodel	Deepface	

Communication Tool

Kakao Talk

GitHub

Google Shared Drive

프로젝트 구현

1. 모델 비교 분석
2. 이미지 인식
3. 실시간 웹캠 인식
4. 동영상 인식

모델 비교 분석_ 이미지인식Test

7개의 얼굴이 포함된 img_1(모두 정면)과 img_2(정면, 측면)로 얼굴 인식률을 비교

〈img_1(정면)〉

〈img_2(정면, 측면)〉

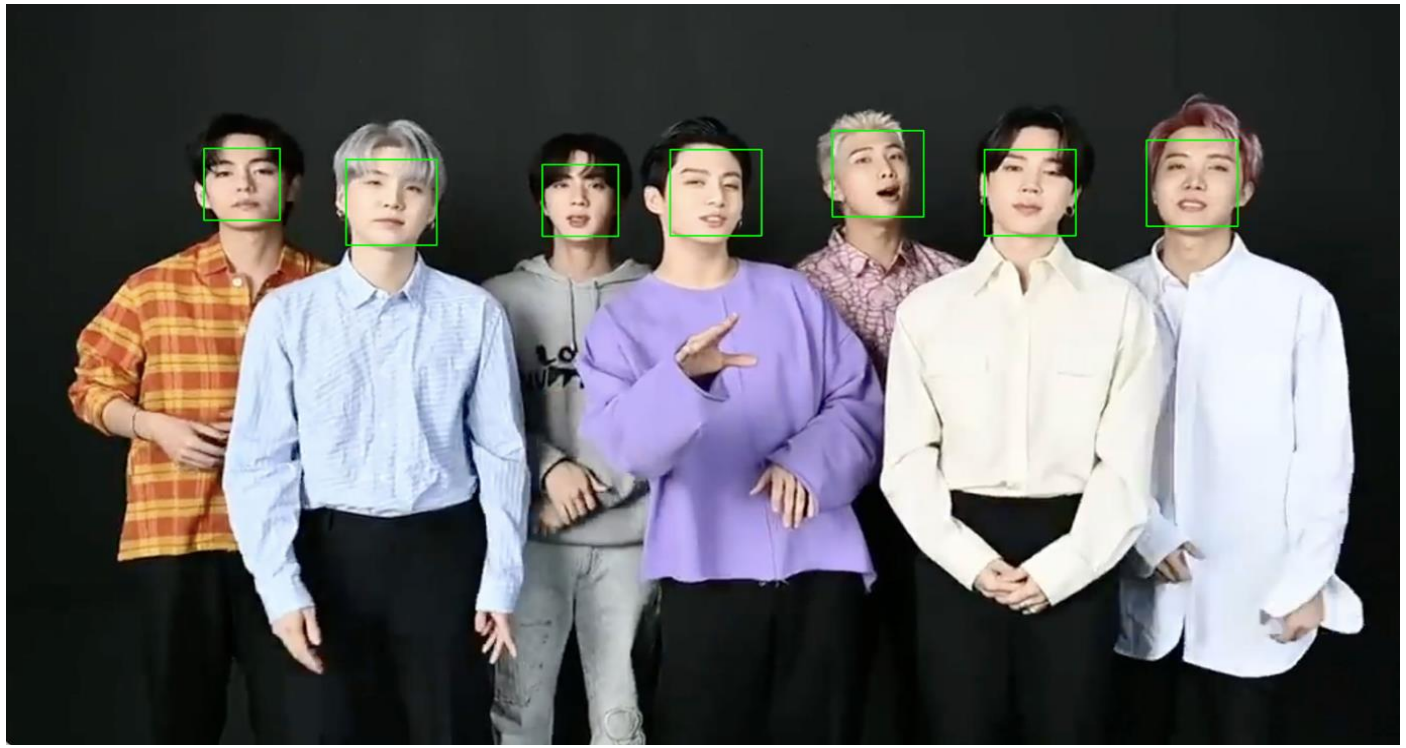
모델 비교 분석_ 이미지 인식 Test

7개의 얼굴이 포함된 img_1(모두 정면)과 img_2(정면, 측면)로 모델 7개 얼굴 인식률을 비교

Model	img_1(정면)	img_2(정면+측면)	비고
Haar_Cascade	7명 (100%)	5명 (42.9%)	기타 분류기 적용 : 2명 (28.6%) 3명 (43.0%)
Dlib_Hog	7명 (100%)	5명 (71.4%)	측면 2명 제외 모두 인식
Dlib_ResNet	7명 (100%)	5명 (71.4%)	측면 2명 제외 모두 인식
DNN_caffemodel	7명 (100%)	0명 (0.0%)	전원 인식 불가
Deepface	7명 (100%)	7명 (100%)	개별 인식 불가
Face_recognition	7명 (100%)	5명 (71.4%)	학습데이터 오류 발견 (오분류/미분류)
cvlib	7명 (100%)	0명 (0%)	전원 인식 불가

모델 비교 분석_ 이미지 인식 Test

img_1(모두 정면) 모든 모델이 7개 얼굴 인식



모델 비교 분석_ 이미지 인식 Test

img_2(정면+측면) 최대 5개 얼굴 인식

0개

2개

3개

5개

모델 비교 분석_ Haar_cascade

img_2(정면+측면) 측면 인식 불가
: ‘haarcascade_frontface.xml’ 분류기 –〉 2명 인식
: ‘haarcascade_frontface_default.xml’ 분류기 –〉 3명 인식

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontface.xml')
img = cv2.imread('bts_test.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

〈 2명 인식〉

〈 3명 인식〉

모델 비교 분석_ Haar_cascade

img_2(정면+측면) 측면 인식 불가
: ‘haarcascade_frontalface_default.xml’ 분류기와 ‘haarcascade_profile.xml 분류기를 동시에 사용하여 인식률 개선 -> 5명까지 인식 가능

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
#eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
profile_cascade = cv2.CascadeClassifier('haarcascade_profileface.xml')
```

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

cv2.imshow('img', img)
```

〈 5명 인식〉

이미지인식_Dlib

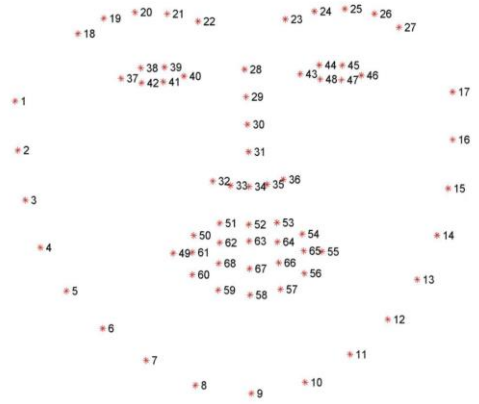
Dlib

c++ 로 개발된 이미지 처리 및 기계 학습, 얼굴인식 등을 할 수 있는 고성능의 라이브러리

특히, 얼굴인식에 관한 강력한 기능을 보이는 라이브러리라고 할 수 있음

파이썬 패키지 dlib에는 이 과정이 구현되어 있음

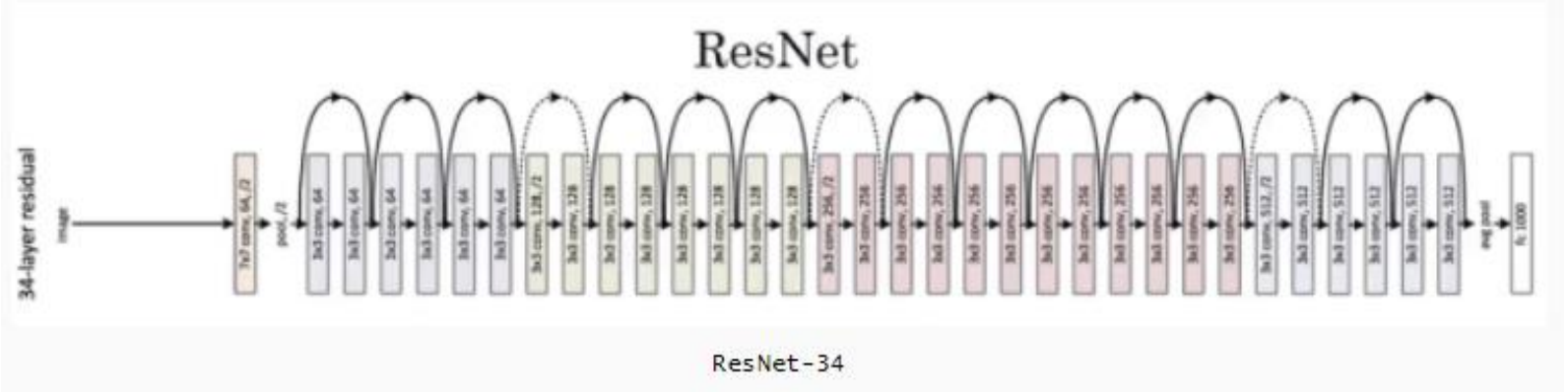
- 그림에서 얼굴이 있는 영역을 알아낸다. (face location)
- 얼굴 영역에서 눈, 코, 입 등 68개의 주요 좌표를 추출한다. (facial landmarks)
- 68개의 좌표를 128개의 숫자(벡터)로 변환한다. (face encoding)



이미지인식_Dlib_ResNet

Dlib_ResNet

ResNet-34 model 에서 몇개의 레이어를 제거하여 29개의 합성곱 레이어로 구성된 신경망을 재구성한다. 150x150x3 크기의 입력을 받고 128차원 벡터로 얼굴 이미지를 표현한다. 다양한 데이터셋으로 훈련한 결과 사람의 정확도(97.53%) 보다 높은 99.35%의 정확도 구현 가능



ResNet-34

-> 보다 높은 정확도를 가지고 있고, 비교한 모델 중 속도가 가장 빨라 이 모델을 사용하여 구현

이미지인식_얼굴인식코드

객체 정의 -> 얼굴 탐지 함수 정의(find_faces) -> 학습한 이미지의 랜드마크 거리 계산(encode_faces)

-> 이미지에서 특정 얼굴 탐지 및 피대상자 비식별조치(모자이크 처리)

1. 객체 정의

dlib 라이브러리 내장 함수 shape_predictor 및 resnet 모델로 객체를 정의
: 68개의 랜드마크를 이용하여 얼굴을 인식하는 shape_predeictor_68_face_landmarks 와 dlib_face_recognition_resnet_model_v1을 사용

```
detector = dlib.get_frontal_face_detector()
sp = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
facerec = dlib.face_recognition_model_v1('dlib_face_recognition_resnet_model_v1.dat')
```

이미지인식_얼굴인식코드

2. 얼굴 탐지 함수 정의(find_faces)

```
def find_faces(img):
    dets = detector(img, 1)

    if len(dets) == 0:
        return np.empty(0), np.empty(0), np.empty(0)

    rects, shapes = [], []
    shapes_np = np.zeros((len(dets), 68, 2), dtype= int)
    for k, d in enumerate(dets):
        rect = ((d.left(), d.top()), (d.right(), d.bottom()))
        rects.append(rect)

        shape = sp(img, d)

        # convert dlib shape to numpy array
        for i in range(0, 68):
            shapes_np[k][i] = (shape.part(i).x, shape.part(i).y)

        shapes.append(shape)

    return rects, shapes, shapes_np
```

- 입력 이미지에서 detector로 얼굴을 검출하여 dets로 받음
- 이미지에 얼굴이 없을 때 비어있는 0의 np 값을 반환(예외 처리)
- (len(dets), 68,2) shape의 데이터타입이 정수인 0으로 채운 array인 shapes_np 생성

  → len(dets)는 이미지에서 찾은 얼굴의 갯수
- 검출된 얼굴들에서 반복해서 좌표left, top, right, bottom 값을 뽑아 rects 리스트에 append
- shape_predictor 함수로 얻은 68개의 값을 넘파이 array 로 변환하고 shapes 리스트에 append
- rects, shapes, shapes_np 반환

이미지인식_얼굴인식코드

3. 얼굴의 랜드마크 추출 함수 정의(encode_faces)

```
def encode_faces(img, shapes):
    face_descriptors = []
    for shape in shapes:
        face_descriptor = facerec.compute_face_descriptor(img, shape)
        face_descriptors.append(np.array(face_descriptor))

    return np.array(face_descriptors)
```

- 탐지한 얼굴에서 특징점의 랜드마크 68개를 추출(find_faces 함수에서 반환된 shape)하여 128개의 벡터로 변환해 얼굴 판별하는 함수임.
- facerec 는 전달된 이미지를 128차원 벡터로 변환하는데 사용된다. 29 개의 합성곱 레이어가 있는 사전 훈련된 ResNet 네트워크 모델로 약 3백만 개의 얼굴로 구성된 데이터셋에서 학습되었다.
- encode_faces 함수는 128차원의 벡터의 얼굴 랜드마크의 넘파이 배열을 반환한다.

이미지인식_얼굴인식코드

4. 이미지 학습

```
# 이미지학습 train
img_paths = {'정국1':'정국1.jpg', '정국2':'정국2.jpg', '정국3':'정국3.jpg'}
```

〈학습 이미지 3장(정면, 정면, 측면)〉

- 학습 시키고 싶은 이미지를 img_paths에 넣어서 지정

이미지인식_얼굴인식코드

5. 학습이미지 DB생성

```
descs = []

for name, img_path in img_paths.items():
    img = cv2.imread(img_path)
    _, img_shapes, _ = find_faces(img)
    descs.append([name, encode_faces(img, img_shapes)[0]])

np.save('descs.npy', descs)
```

```
[['정국1', array([-6.42565116e-02,  1.67656913e-02,  1.09558143e-02, -1.03970215e-01,
       -6.32530674e-02, -4.79219817e-02, -1.03603899e-01, -1.01171561e-01,
        1.23213857e-01, -1.25224471e-01,  2.15959981e-01, -9.87264812e-02,
       -1.82047710e-01, -5.88646606e-02, -7.53460005e-02,  1.89463228e-01,
       -1.64266288e-01, -1.61002964e-01, -5.99844828e-02,  6.47784770e-03,
        1.00731090e-01,  4.02793065e-02, -4.91798334e-02,  8.15239325e-02,
       -1.18247569e-01, -2.68381476e-01, -1.00011528e-01, -4.03784588e-02,
```

- 앞에서 정의한 encode_faces함수를 이용하여 랜드마크를 추출하고 학습된 이미지 DB를 생성한다.
- 각 이미지 당 128개의 값으로 학습된 것을 확인할 수 있다.
- 128개 숫자는 deep learning의 결과물이다.
- 같은 사람의 얼굴을 입력하면 비슷한 숫자가 나온다는 점
  → 얼굴을 구분하는 특징점 이라고 볼 수 있음

이미지인식_얼굴인식코드

6. 이미지에서 특정 얼굴 탐지 및 피대상자 비식별조치(모자이크 처리)

```
img = cv2.imread('bts_1.jpeg')

rects, shapes, _ = find_faces(img) # 얼굴 찾기
descriptors = encode_faces(img, shapes) # 인코딩

for i, desc in enumerate(descriptors):
    x = rects[i][0][0] # 얼굴 X 좌표
    y = rects[i][0][1] # 얼굴 Y 좌표
    w = rects[i][1][1]-rects[i][0][1] # 얼굴 너비
    h = rects[i][1][0]-rects[i][0][0] # 얼굴 높이

    # 추출된 랜드마크와 데이터베이스의 랜드마크들 중 제일 짧은 거리를 찾는 부분
    descs1 = sorted(descs, key=lambda x: np.linalg.norm([desc] - x[1]))
    dist = np.linalg.norm([desc] - descs1[0][1], axis=1)
```

- 사용할 이미지를 img로 읽어오기
- find_face 함수로 얼굴을 찾기
- encode_faces 함수로 얼굴 특징점 인코딩

- 새로 찾은 얼굴의 x,y,w,h 값에 대하여 DB 얼굴 특징점 벡터와의 유클리디안 거리 값으로 유사도를 계산해 얼굴을 확인

이미지인식_얼굴인식코드

6. Test 이미지에서 특정 얼굴 탐지 및 피대상자 비식별조치(모자이크 처리)

- 유클리디안 거리 값이 작을 수록 유사도 큼(학습데이터와 같은 사람으로 판단)
- Threshold 값을 0.45로 설정하고 거리 값이 그 이하일 때 특정 인물로 인식하고
- 0.45를 초과하면 비식별 처리(모자이크/ 바운딩 박스와 함께 표기)

→ 임계값은 테스트를 과정에서 정확도, 특이도, 정밀도 모두 안정적이면서 비교적 엄격한 수치로 결정함[1)]

1) 허석렬, “딥러닝 얼굴인식 기술을 활용한 방문자 출입관리 시스템 설계와 구현”, 디지털융복합연구 제19권 제2호(2021), pp.245-251.

이미지인식_얼굴인식코드

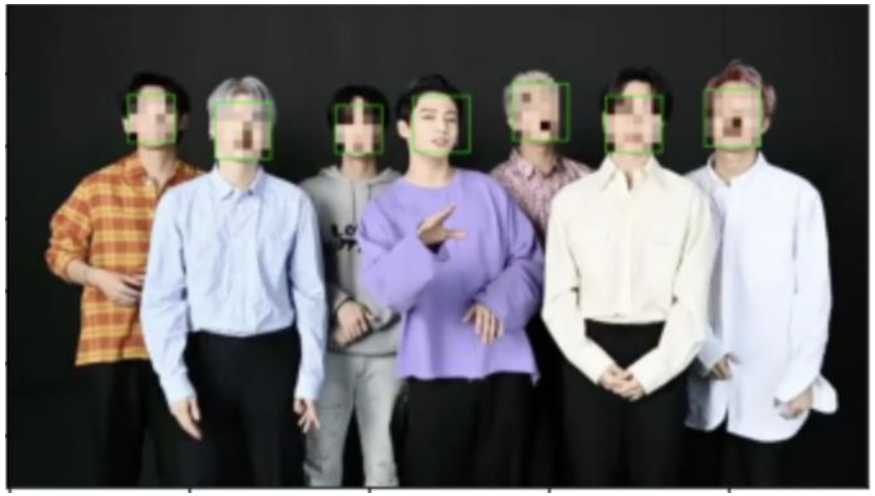
6. Test 이미지에서 특정 얼굴 탐지 및 피대상자 비식별조치(모자이크 처리)

```
    if dist < 0.45: # 그 거리가 0.45보다 작다면 그 사람으로 판단
        name = descs1[0][0]
    else:           # 0.45보다 크다면 모르는 사람으로 판단 -> 모자이크 처리
        mosaic_img = cv2.resize(img[y:y+h, x:x+w], dsize=(0, 0), fx=0.04, fy=0.04) # 축소
        mosaic_img = cv2.resize(mosaic_img, (w, h), interpolation=cv2.INTER_AREA) # 확대
        img[y:y+h, x:x+w] = mosaic_img # 인식된 얼굴 영역 모자이크 처리

    cv2.rectangle(img, (x, y), (x+w, y+h), (0,255,0), 2) # 얼굴 영역 박스
    #cv2.putText(img, str(dist)[1:6], (x+5,y+h-5),  2, (0,0,255), 4) # 사진에 거리 출력


#     한글
    img = Image.fromarray(img)
    draw = ImageDraw.Draw(img)
   # draw.text((x+5,y-50), name, font=ImageFont.truetype("./batang.ttc", 60), fill=(255,255,255))
    img = np.array(img)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

이미지인식_얼굴인식코드

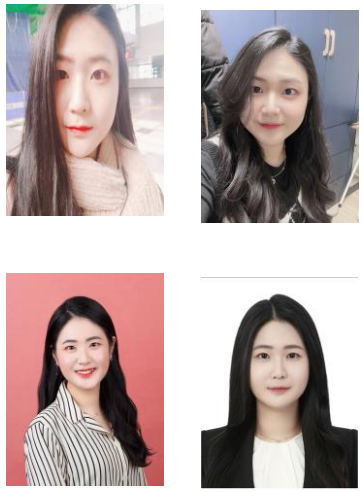
7. 구현결과

: 학습된 특정 얼굴 정국 외의 6인 모두 비식별 조치(모자이크 처리)

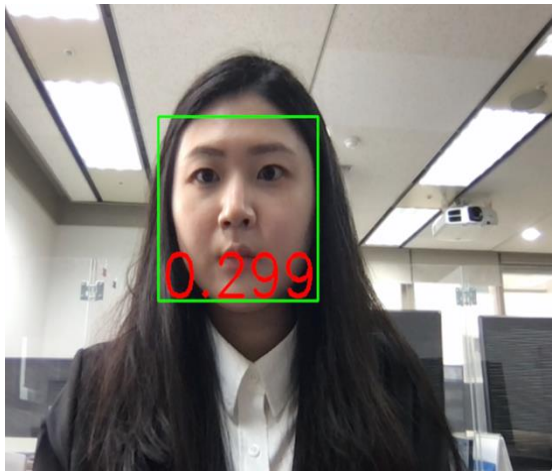


실시간 웹캠_Dlib

배경색이 모두 다른 4장의 이미지 학습 후 테스트 결과



〈학습 이미지 4장〉

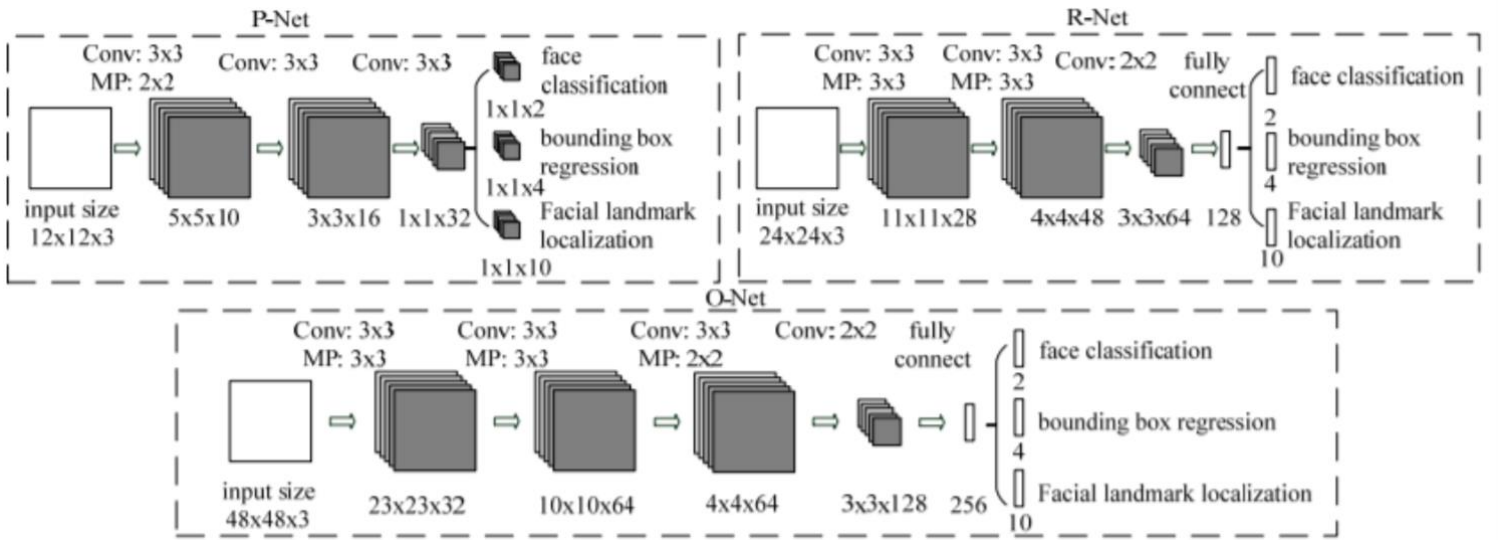


〈특정인물 인식〉

〈특정인물 외 비식별조치〉

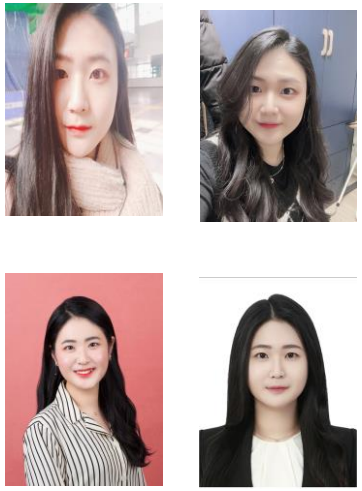
MTCNN(Multi-task Cascaded Convolutional Network)란?

P-Net(Proposal Network) : 이 네트워크는 12 X 12 크기의 프레임에서 얼굴을 찾는다. 이 네트워크의 목적은 빠른 결과를 만드는 것이다.
R-Net(Refine Network) : 이 네트워크는 P-Net보다 더 깊은 구조를 갖는다. 이전 네트워크인 P-Net에서 전달받은 모든 후보가 R-Net으로 전달된다. R-Net은 여기서 다량의 후보를 탈락시킨다.
O-Net(Output Network) : 바운딩 박스(bounding box - 얼굴영역)와 얼굴 랜드마크 위치를 반환한다.

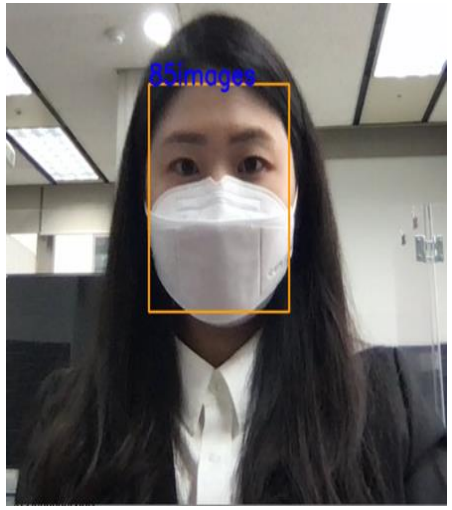


실시간 웹캠_MTCNN

실시간 웹캠을 활용한 마스크 착용 후 특정인물 인식_속도보다는 정확도에 초점 **MTCNN모델** 사용



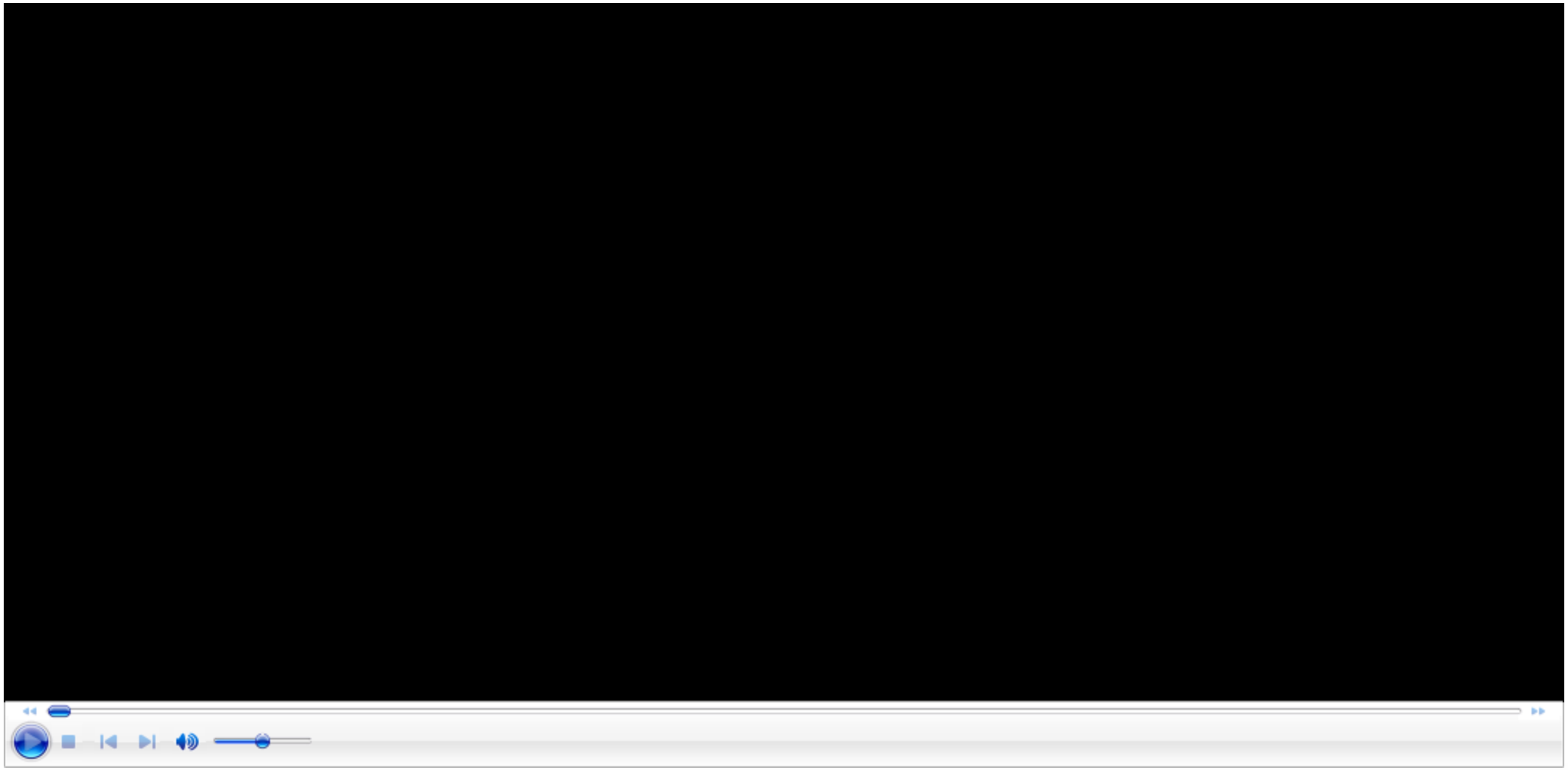
〈학습 이미지 4장〉



〈특정인물 인식〉

〈특정인물 외 비식별조치〉

동영상 인식



BLUR_ABLE

프로젝트 결론

1. 기대효과
2. 향후과제
3. 질의 응답

기대효과

- 초상권 침해 방지 효과 및 인식개선
- 외부 촬영, 브이로그 촬영 유튜버 등 작업시간 감소로 인한 진입장벽을 낮출 수 있음
- 사진이나 영상 속 군중에서 범죄자 식별 탐지 및 추적 기능
- 자동차 표지판, 도로, 가게 간판 등 비식별 조치화 응용

향후과제

특정 얼굴 인식 불가 -> 눈썹이 드러나지 않고, 그에 따라 얼굴 인식 불가

BLUR_ABLE

향후과제

인식하지 못한 좌, 우측의 측면 얼굴은 머리카락으로 얼굴 일부분을 가리고 있어 이목구비를 잘 인식하지 못한 것으로 추측 됨

향후과제

학습/테스트 사진의 배경에 따라 인식률 차이가 큼

향후과제

GPU의 문제로 다양한 영상을 테스트 하지 못해, 영상에서의 얼굴 인식률의 정도를 정확히 파악하지 못함

Q & A

THANK YOU