

FIND MY FACE

내 얼굴을 찾아조

인공지능 프로그래밍 경진대회



윤한조(20191789)

강연승(20161886)

류상배(20191796)

CONTENTS

개요

데이터 저장과 전처리

데이터와 훈련과정

분류

딥러닝 모델

예측 결과의 시각화



개요

배경



학생 수가 많아서 매번 출석
부르기 힘들었던 교수님이
출석 부르는 영상을 띄우고
체크하셨다는 얘기를 듣고
출석체크를 쉽게 할 수 있는
방법을 구상하게 되었음

목적



웹캠이나 영상으로 본인 얼굴 인식

기대효과



A

코로나 19로 인한 실시간 강의 시 번거롭게 출석을
부르지 않아도 화상 출석체크에 사용 가능



B

웹캠을 집의 현관이나 복도 등 주요 장소에 설치하면
스마트 초인종, 스마트 CCTV 등으로 사용 가능

분류

이진분류

“이진분류란? 데이터 값에 따라 참은 1, 거짓은 0 으로 참 거짓 두가지로 판별하는 방식입니다.”

저희 프로젝트에서 원하는 참의 기준은 다음과 같습니다.

- 거리가 0.6 이면 다른 사람의 얼굴 → 이런 경우의 이름은 Unknown
- 거리가 0.6 이하이고, 최소값을 가진 사람의 이름을 찾음

분류

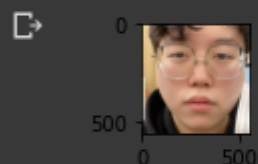
이미지 인코딩

```
▶ image_path = '/gdrive/My Drive/Colab Notebooks/ob/20191796류상배.jpg'
image = fr.load_image_file(image_path)

face_locations = fr.face_locations(image)

for (top, right, bottom, left) in face_locations:
    ryu = image[top:bottom, left:right]

plt.rcParams['figure.figsize'] = (1, 1)
plt.imshow(ryu)
plt.show()
```



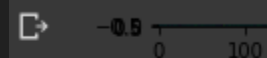
face_encodings 함수를 이용한 인코딩

```
[74] enc_ryu_face = fr.face_encodings(ryu)
```

인코딩 결과

```
[75] plt.imshow(enc_ryu_face)
plt.show()
```

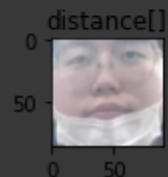
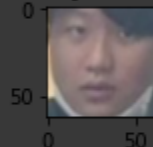
```
▶ plt.imshow(enc_ryu_face)
plt.show()
```



+ 코드

```
[80] for face in my_faces:
    # 자신의 얼굴 인코딩하기
    enc_my_face = fr.face_encodings(face)
    # 자신의 얼굴과의 거리 비교
    distance = fr.face_distance(enc_my_face, enc_ryu_face[0])
    # 사진 타이틀 지정 및 출력
    plt.title('distance' + str(distance))
    plt.imshow(face)
    plt.show()
```

distance[0.48080214]



분류

조건에 따라서 분류해주는 웹캠 .py 코드

```
if self.process_this_frame:
    # Find all the faces and face encodings in the current frame of video
    self.face_locations = face_recognition.face_locations(rgb_small_frame)
    self.face_encodings = face_recognition.face_encodings(rgb_small_frame, self.face_locations)

    self.face_names = []
    for face_encoding in self.face_encodings:
        # See if the face is a match for the known face(s)
        distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        min_value = min(distances)

        # tolerance: How much distance between faces to consider it a match. Lower is more strict.
        # 0.6 is typical best performance.
        name = "Unknown"
        if min_value < 0.6:
            index = np.argmin(distances)
            name = self.known_face_names[index]

        self.face_names.append(name)

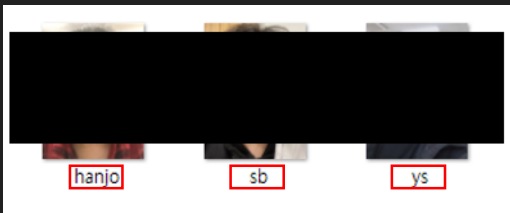
self.process_this_frame = not self.process_this_frame
```

데이터 저장과 전처리



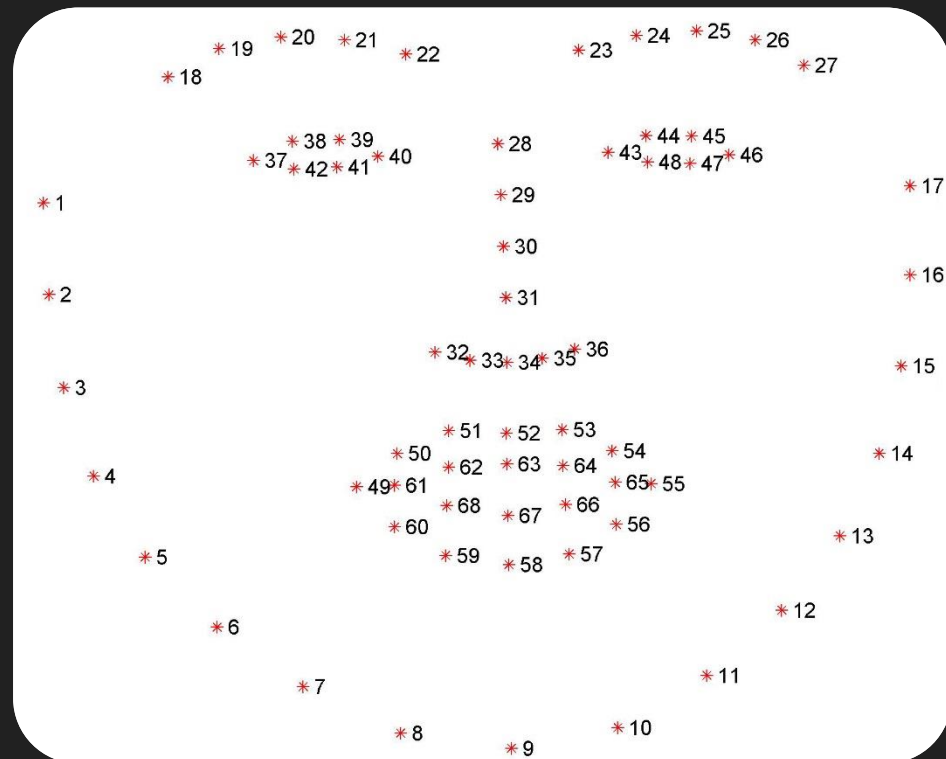
데이터 읽어오기

테스트 데이터를 읽어오는 파일은 knowns 디렉토리를
설정하였고 사진 파일과 파일 이름을 추출

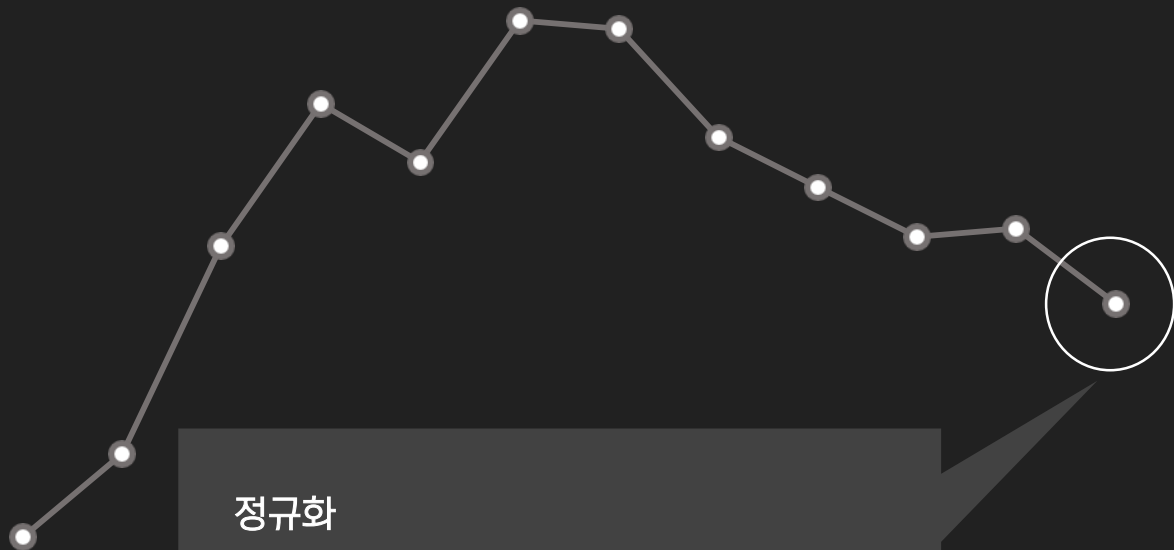


데이터 전처리

사진에서 얼굴 영역을 특정해, face landmarks라 불리는 68개의 얼굴의 특징을 분석한
데이터를 known_face_encodings에 저장한 후 계산량을 줄이기 위해 크기 조절을 하고
읽은 프레임에서 얼굴 영역과 특징을 추출



데이터 저장과 전처리



정규화

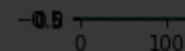
표본 사진과 웹캠의 거리(일치율%)를 0 - 1 사이의 값으로 나타내 조건을 넣어줌

face_encodings 함수를 이용한 인코딩

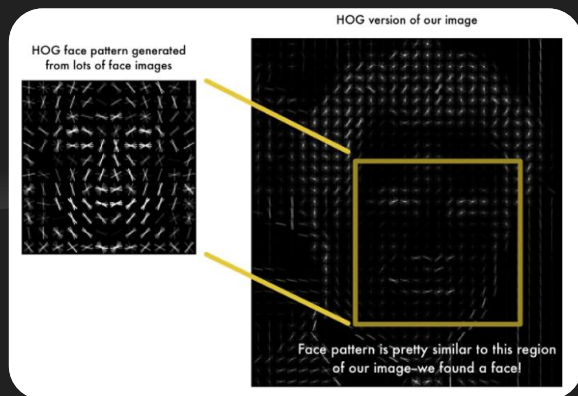
```
▶ enc_ryu_face = fr.face_encodings(ryu)
```

인코딩 결과

```
[75] plt.imshow(enc_ryu_face)  
plt.show()
```

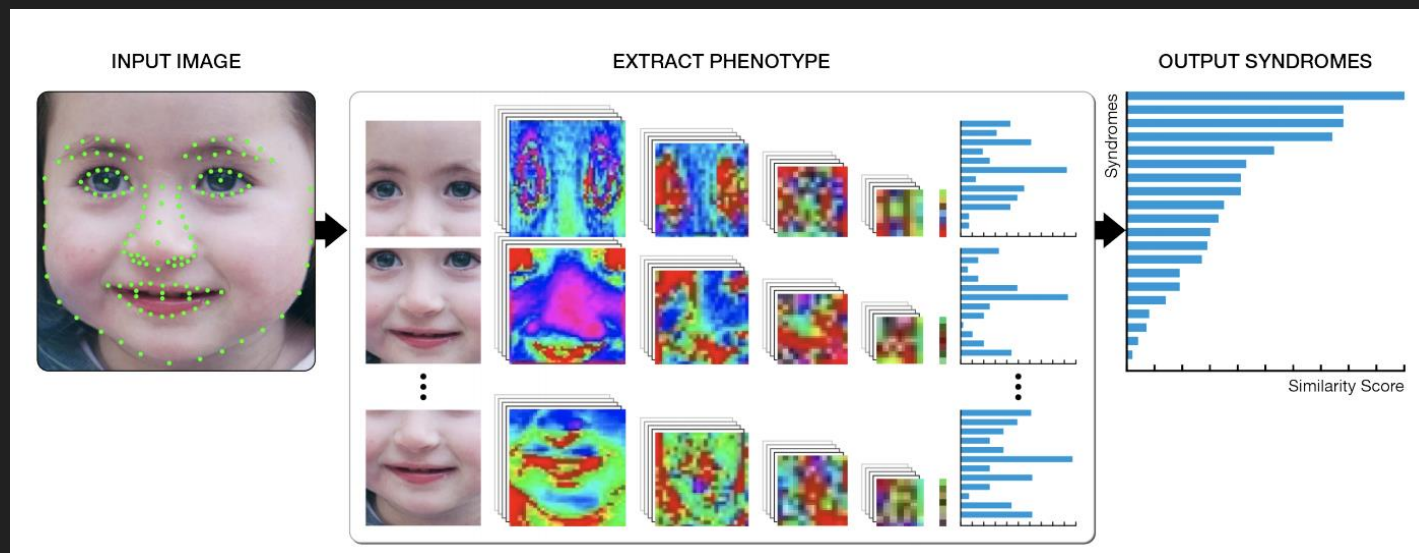


딥러닝 모델



DCNN(Deep Convolutional Neural Network)

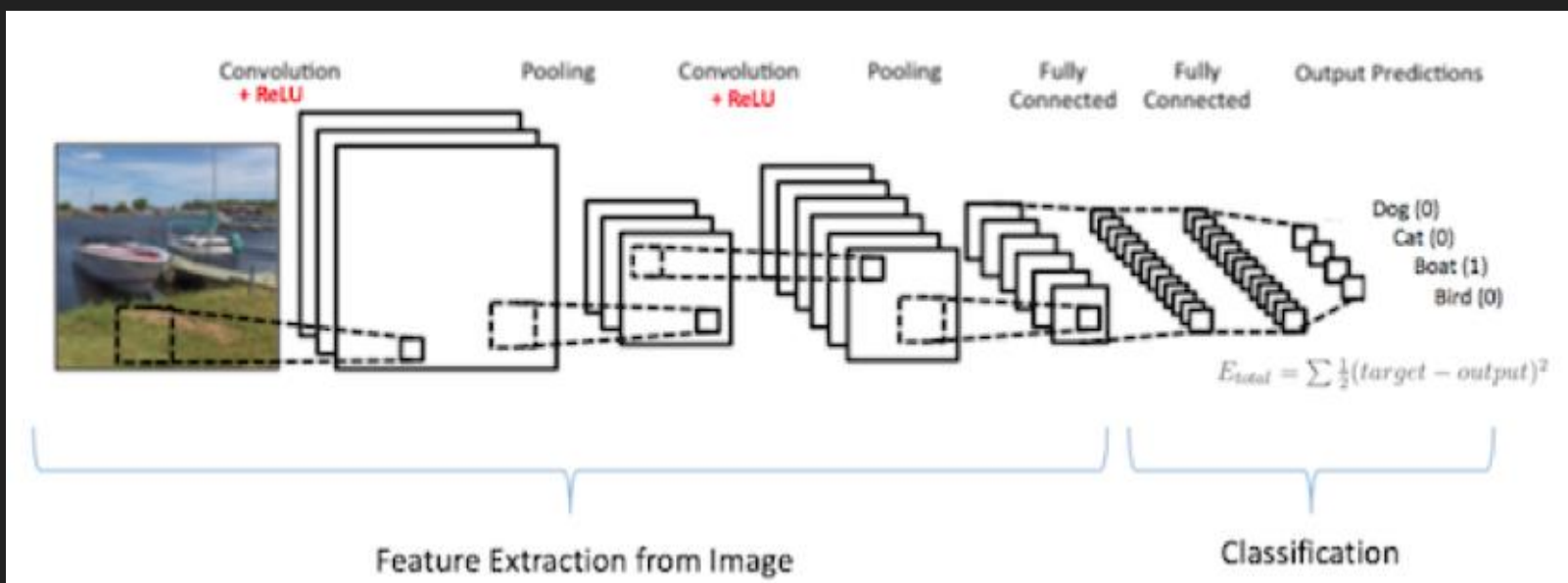
CNN 보다 더 추상적이고 인간의 통찰력에 가까운 신경망



딥러닝 모델

DCNN 의 기본적인 구조는 CNN과 동일 하게 이루어져 있습니다.

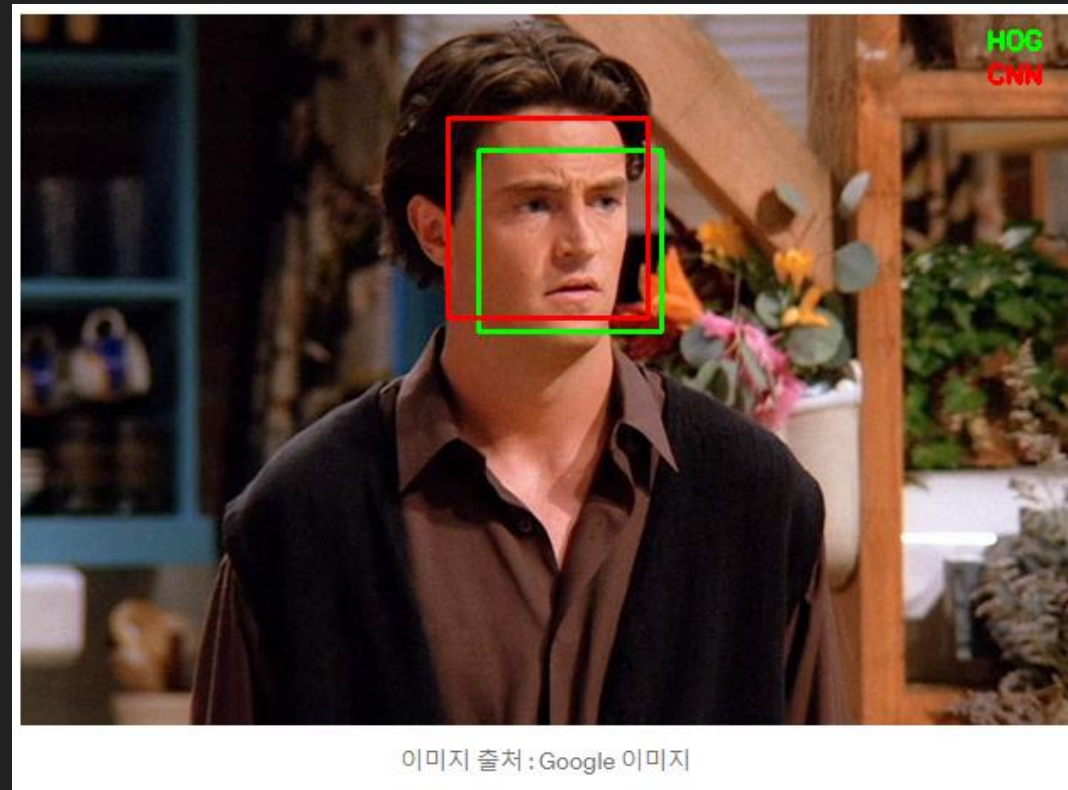
DCNN 장점은 CNN의 구조적 한계인 방향과 비율 바라보는 시점의 한계를 보완해줍니다.



딥러닝 모델

입력층

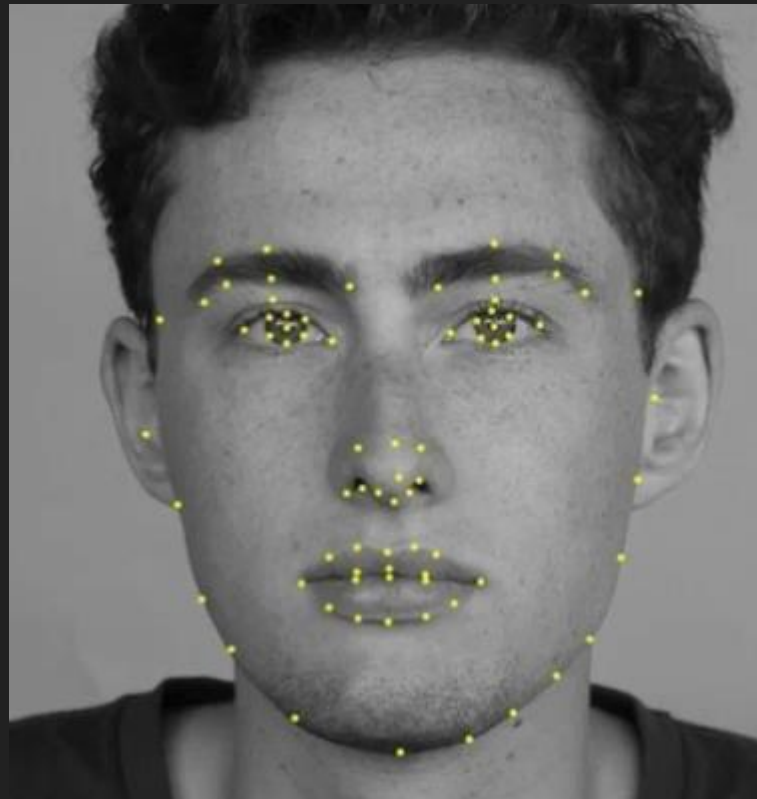
- HOG 알고리즘을 이용해 이미지를 단순화
- 이미지와 얼굴의 가장 유사해 보이는 부분을 탐색



딥러닝 모델

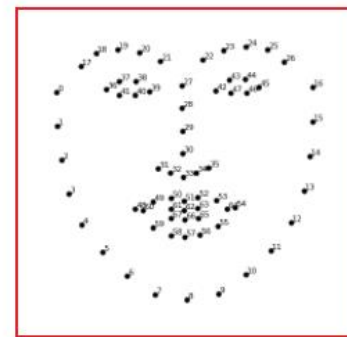
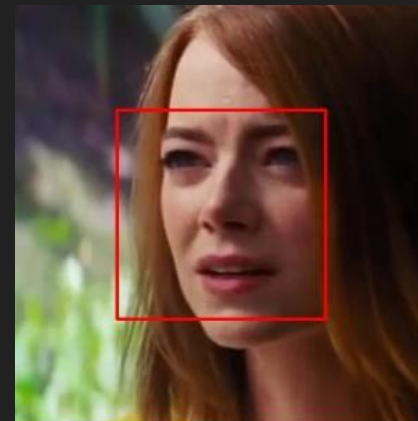
중간층

- 랜드마크를 찾아 얼굴의 모양을 알아냄
- 눈과 입이 중앙에 오도록 이미지를 변형



얼굴 인식의 과정

1. 그림에서 얼굴이 있는 영역을 알아낸다. (face location)
2. 얼굴 영역에서 눈, 코, 입 등 68개의 주요 좌표를 추출한다. (facial landmarks)
3. 68개의 좌표를 128개의 숫자로 변환한다. (face encoding)
 - 128개 숫자는 deep learning의 결과물이기 때문에 각 숫자의 의미는 알지 못함
 - 하지만 같은 사람의 얼굴을 입력하면 비슷한 숫자가 나옴



[0.45, 0.19, 0.84, 0.66,
0.05, 0.39, 0.96, 0.81,
...
0.72, 0.31, 0.57, 0.49]

얼굴 간의 거리 (유사성) 구하기

사진에서 얼굴을 인식



그 얼굴의 face_encoding을 학습한 사람의 face_encoding과 비교



가장 거리가 가까운 (가장 비슷한) 사람 찾기 가능

face_encoding은 128 차원의 벡터



수학적으로 L2 norm (유클리드 거리)를 이용하면 두 벡터의 거리를



구할 수 있기 때문에 서로 얼마나 가까운지 비교 가능



파이썬에서는 numpy의 linalg.norm() 함수를 이용

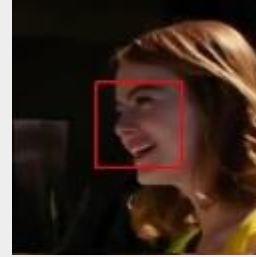
파이썬 패키지 face_recognition의 face_distance() 함수는
numpy의 linalg.norm() 함수의 wrapper로 구현



두 face_encoding의 거리는 0 ~ 1 사이의 값으로 구해지고,
이 값은 두 얼굴이 얼마나 비슷한 지에 대한 척도로 사용 가능

데이터와 훈련과정

1. 훈련용 아는 사람의 얼굴 사진 적재(load)



2. 동일한 아는 사람의 다른 사진 적재



데이터와 훈련과정

3. 전혀 다른 사람의 사진 적재



4. 알고리즘에서 세 개의 이미지 각각에 대해
현재 생성되는 측정값을 확인해 신경망을 조정하고,
각 얼굴을 러닝해 128개의 측정값(임베딩)을 제공합니다.

```
img_paths = {
    'sb': 'img/20191796류삼배.jpg',
    'hanjo': 'img/20191789윤한조.jpg',
    'ys': 'img/20161886강연승.jpg',
}

descs = {
    'sb': None,
    'hanjo': None,
    'ys': None,
}

for name, img_path in img_paths.items():
    img_bgr = cv2.imread(img_path)
    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)


    _, img_shapes, _ = find_faces(img_rgb)
    descs[name] = encode_faces(img_rgb, img_shapes)[0]

np.save('img/descs.npy', descs)
print(descs)
```

```
{'sb': array([-0.03382163,  0.03314117,  0.09757136, -0.02186488, -0.07401694,
               0.00197663, -0.06304663, -0.09394763,  0.12042058, -0.12319717,
               0.2397434 , -0.01145236, -0.22679706, -0.05021239, -0.04714309,
               0.17340891, -0.17401531, -0.11677527, -0.03247834,  0.00042242,
               0.04534199,  0.01471337,  0.01340745,  0.02557805, -0.12723488,
              -0.31796709, -0.10631758, -0.07291916, -0.01339537, -0.03060094,
              -0.0755947 ,  0.06667549, -0.12024336,  0.01443091,  0.02599729,
               0.0927707 , -0.0094328 , -0.10635617,  0.16524452,  0.01346461,
              -0.24365737,  0.07444928,  0.00588227,  0.21299095,  0.17716983,
               0.03906694,  0.02704768, -0.10180095,  0.1351902 , -0.20301616,
               0.05311631,  0.15842807,  0.08455649,  0.06899515, -0.00352532,
              -0.11239804,  0.05787743,  0.05354468, -0.08016118, -0.00591741,
               0.05957318, -0.0938555 ,  0.03395691, -0.04148825,  0.15083361,
               0.07726467, -0.10051125, -0.15891972,  0.08976453, -0.17030264,
              -0.12776943,  0.06709476, -0.19295801, -0.15012974, -0.29909796,
               0.02058021,  0.41944516,  0.10636707, -0.19115287,  0.03494427,
              -0.0222014 ,  0.01525955,  0.14086106,  0.18502381, -0.00869346,
              -0.01438482, -0.11289093, -0.0085019 ,  0.27435806, -0.08492574,
              -0.06242473,  0.21796882,  0.01440977,  0.10285294,  0.0345906 ,
               0.04255563,  0.01013615,  0.0375254 , -0.06051267,  0.02188758,
               0.09135295,  0.09301541,  0.09459959,  0.11954299,  0.17183718,
```


프로젝트 코드

1. 머신러닝을 이용해 학습이 끝난 랜드마크 모델 파일을 사용

 shape_predictor_68_face_landmarks.dat

 dlib_face_recognition_resnet_model_v1.dat

이 모델 파일의 역할은 적은 표본으로도 높은 정확도를 보여줄 수 있게 도움

2. 모델을 사용해 68개의 점으로 표본이미지를 분석할 수 있게 한 코드 

```
import matplotlib.path as path_effects

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

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=np.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

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)
```

프로젝트 코드

3. 웹캠을 사용해 얼굴을 인식할 수 있도록 하는 코드

```
class FaceReog():
    def __init__(self):
        # 웹캠이 설치되어 있는지 확인
        self.camera = camera.VideoCamera()

        self.known_face_encodings = []
        self.known_face_names = []

        # 샘플 이미지 로드
        dirname = 'knowns'
        files = os.listdir(dirname)
        for filename in files:
            name, ext = os.path.splitext(filename)
            if ext == '.jpg':
                self.known_face_names.append(name)
                pathname = os.path.join(dirname, filename)
                img = face_recognition.load_image_file(pathname)
                face_encoding = face_recognition.face_encodings(img)[0]
                self.known_face_encodings.append(face_encoding)

        self.face_locations = []
        self.face_encodings = []
        self.face_names = []
        self.process_this_frame = True
```

```
def __del__(self):
    del self.camera

def get_frame(self):
    frame = self.camera.get_frame()

    # 샘플링 파일을 리사이즈 해줍니다 (용량 때문)
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    rgb_small_frame = small_frame[:, :, ::-1]

    if self.process_this_frame:
        self.face_locations = face_recognition.face_locations(rgb_small_frame)
        self.face_encodings = face_recognition.face_encodings(rgb_small_frame, self.face_locations)

        self.face_names = []
        for face_encoding in self.face_encodings:
            distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
            min_value = min(distances)

            # 프른이미지와 웹캠에 비춘 얼굴의 거리가 0.6 미만이면 unknown이 뜨도록 했습니다.
            name = "Unknown"
            if min_value < 0.6:
                index = np.argmin(distances)
                name = self.known_face_names[index]

            self.face_names.append(name)

        self.process_this_frame = not self.process_this_frame

    for (top, right, bottom, left), name in zip(self.face_locations, self.face_names):
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

    return frame

def get_jpg_bytes(self):
    frame = self.get_frame()
    ret, jpg = cv2.imencode('.jpg', frame)
    return jpg.tobytes()
```

예측 결과의 시각화



얼굴을 인식하여 하단에 파일명이 출력되고, 학습하지 않은 얼굴은 unknown으로 출력

Thank you 😊