



# OpenCV와 딥러닝을 이용한 얼굴 인식

최석재 [lingua@naver.com](mailto:lingua@naver.com)

# Face Recognition

- AI에서 얼굴 인식은 중요한 기술 중 하나
- 보안 및 감시, 개인화된 서비스, 건강 관리 등에 인식된 정보가 활용된다
- 여기서는 얼굴 인식 기술로 자주 사용되는 라이브러리인
- OpenCV와 Dlib의 사용 방법을 알아본다

# 구글 드라이브 연결

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

*# 경로 변경*

```
%cd /content/gdrive/MyDrive/pytest_img/opencv/
```

# 얼굴 인식하기 1/3

※ 다음은 Colab에서의 코드이나, cv2.imshow() 부분만 수정하여 Local에서도 동일한 코드로 수행할 수 있다

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
```

```
img = cv2.imread('/content/gdrive/MyDrive/pytest_img/img_align_celeba_small/202485.jpg')
```

```
# 이미지 그리기
cv2_imshow(img)
```

```
# 사용자의 반응을 기다리는 함수
# 파이썬 커널과의 충돌을 막기 위해 필요
cv2.waitKey(0)
```

```
# 모든 윈도우를 닫기
cv2.destroyAllWindows()
```



# 얼굴 인식하기 2/3

얼굴과 눈을 찾는 알고리즘 파일은 OpenCV가 제공하는 파일이나,  
아래 경로의 data 폴더에서도 찾을 수 있다  
<https://github.com/opencv/opencv>

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_eye.xml')
faces = face_cascade.detectMultiScale(image=gray, scaleFactor=1.2, minNeighbors=1)
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]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
```

얼굴을 쉽게 찾도록 이미지를 회색조로 변환

얼굴을 찾는 알고리즘 파일 로드

눈을 찾는 알고리즘 파일 로드

얼굴 인식 감도(사각형 크기)      인식해야 하는 최소 얼굴의 수  
이미지를 20%씩 줄이며 검색      인식할 얼굴의 수를 입력  
너무 빨리 줄면 놓치므로 1.1~1.4를 사용

얼굴 위치선      얼굴 사각형의 색상과 크기

회색조와 컬러로 얼굴 영역을 (height, width)로 입력

회색조 결과를 이용하여 얼굴 영역에서 눈 인식

눈 인식 결과 x, y, w, h 의 네 개 position이 나온다

눈 사각형의 색상과 크기를 결정해  
컬러 얼굴 이미지에 그림

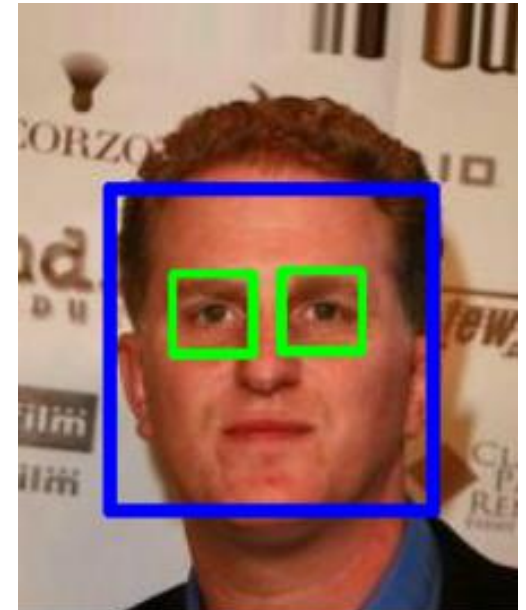
눈 위치선. x에 width를, y에 height를 더함  
(start point, end point)

# 얼굴 인식하기 3/3

```
cv2_imshow(img)
```

```
# 사용자의 반응을 기다리는 함수  
# 파이썬 커널과의 충돌을 막기 위해 필요  
cv2.waitKey(0)
```

```
# 모든 윈도우를 닫기  
cv2.destroyAllWindows()
```



Dlib

# Dlib

- 딥러닝 기반의 오픈소스 라이브러리로
- 최근 얼굴 인식 분야에서 가장 많이 사용되고 있다
- 높은 정확도를 가지며 다양한 OS에서 사용할 수 있는 점이 큰 장점
- 그러나 설치가 조금 어려운 편이다



# 얼굴 인식 모델

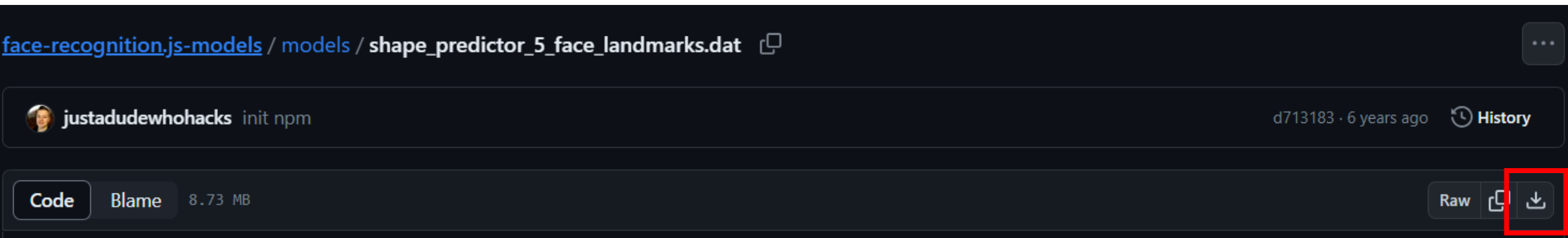
- 얼굴 인식에 사용되는 모델을 다운로드해야 한다
- 이 모델은 얼굴 인식에 필요한 5개의 주요 지점을 탐지한다

- ① 왼쪽 눈의 중심
- ② 오른쪽 눈의 중심
- ③ 코의 끝
- ④ 왼쪽 입술 모서리
- ⑤ 오른쪽 입술 모서리

※ 따라서 정면 얼굴을 잘 인식한다

# 모델 다운로드

- 다음의 깃헙 경로에서 모델을 다운로드 할 수 있다
- [https://github.com/justadudewhohacks/face-recognition.js-models/blob/master/models/shape\\_predictor\\_5\\_face\\_landmarks.dat](https://github.com/justadudewhohacks/face-recognition.js-models/blob/master/models/shape_predictor_5_face_landmarks.dat)



※ /content/gdrive/MyDrive/pytest\_img/dlib/shape\_predictor\_5\_face\_landmarks.dat 에도 있음

# Colab에서 수행

- 먼저 Colab에서의 진행 방법을 알아본다
- Colab은 이미 Dlib을 설치해놓은 상태이기 때문에 쉽게 사용할 수 있다
- `from google.colab import drive`
- `drive.mount('/content/gdrive')`

# 이미지 출력

- import dlib
- import matplotlib.pyplot as plt
- import matplotlib.patches as patches

- img\_path = '/content/gdrive/MyDrive/pytest\_img/img\_align\_celeba\_small/202485.jpg'

# 얼굴의 주요 지점 인식 모델 로드

- sp = dlib.shape\_predictor("/content/gdrive/MyDrive/pytest\_img/dlib/shape\_predictor\_5\_face\_landmarks.dat")
- detector = dlib.get\_frontal\_face\_detector() # 얼굴 인식 함수
- img = dlib.load\_rgb\_image(img\_path) # 이미지 로드
- plt.figure(figsize=(8, 5)) # 이미지 사이즈 설정
- plt.imshow(img)



# 올곧은 영역 인식

- `img_result = img.copy()`
- `dets = detector(img, 1)`
- `if len(dets) == 0:`  
    `print('cannot find faces!')`
- `fig, ax = plt.subplots(1, figsize=(8, 5))`
- `for det in dets:`  
    `x, y, w, h = det.left(), det.top(), det`  
    `rect = patches.Rectangle((x,y), w, h`  
    `ax.add_patch(rect)`
- `ax.imshow(img_result)`

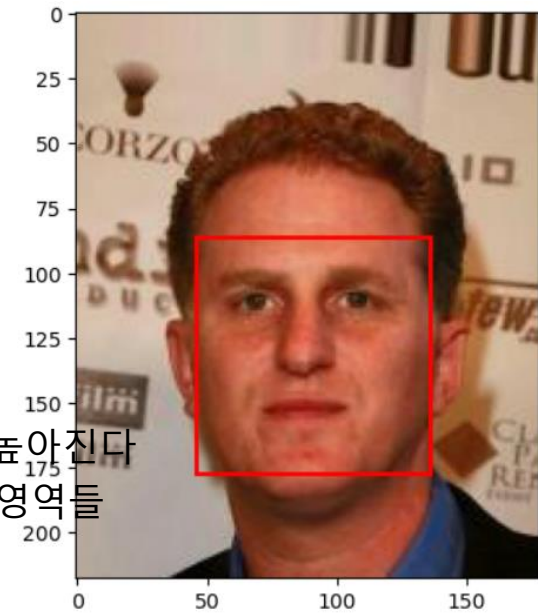
# 1은 업샘플링 횟수. 횟수를 늘릴수록 해상도가 높아진다  
# dets는 얼굴 검출기가 이미지에서 검출한 얼굴 영역들  
# 얼굴영역의 갯수가 0일 경우

# 이미지 사이즈 설정. figure는 전체 캔버스, axes는 부분 영역(여기서는 1개)

# 인식된 얼굴 영역 개수에 따라  $x, y, w, h$  위치 파악

```
x, y, w, h = det.left(), det.top(), det.width(), det.height()
rect = patches.Rectangle((x,y), w, h, linewidth=2, edgecolor='r', facecolor='none') # 사각형 옵션
ax.add_patch(rect) # 이미지에 사각형 부착
```

facecolor는 객체 내부 색상을 지정하는 데 사용  
red, blue, green, yellow 등



# 얼굴 주요 지점 파악

- `fig, ax = plt.subplots(1, figsize=(8, 5))`

- `for det in dets:`

- `s = sp(img, det)`

- `for point in s.parts():`

- `circle = patches.Circle((point.x, point.y), radius=3, edgecolor='r', facecolor='r')`

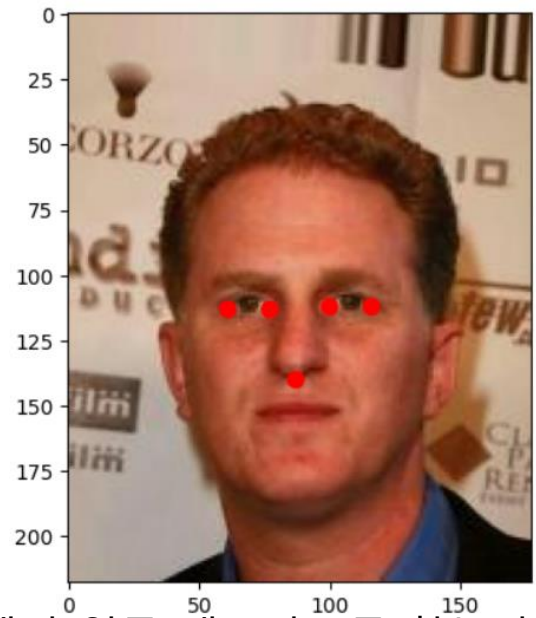
- `ax.add_patch(circle)`

- `ax.imshow(img_result)`

# 검출된 얼굴 영역들

# 얼굴 주요 지점 인식 모델로 각 얼굴영역에서 얼굴 랜드마크를 찾는다

# 5개의 점에 대한 for문



# Local PC에 설치

- Local에 설치하는 것은 조금 어렵지만,
- Python 및 텐서플로 설치에 성공하였다면 비교적 쉽게 진행할 수 있다
- <http://dlib.net/> 에서 dlib을 다운로드한다

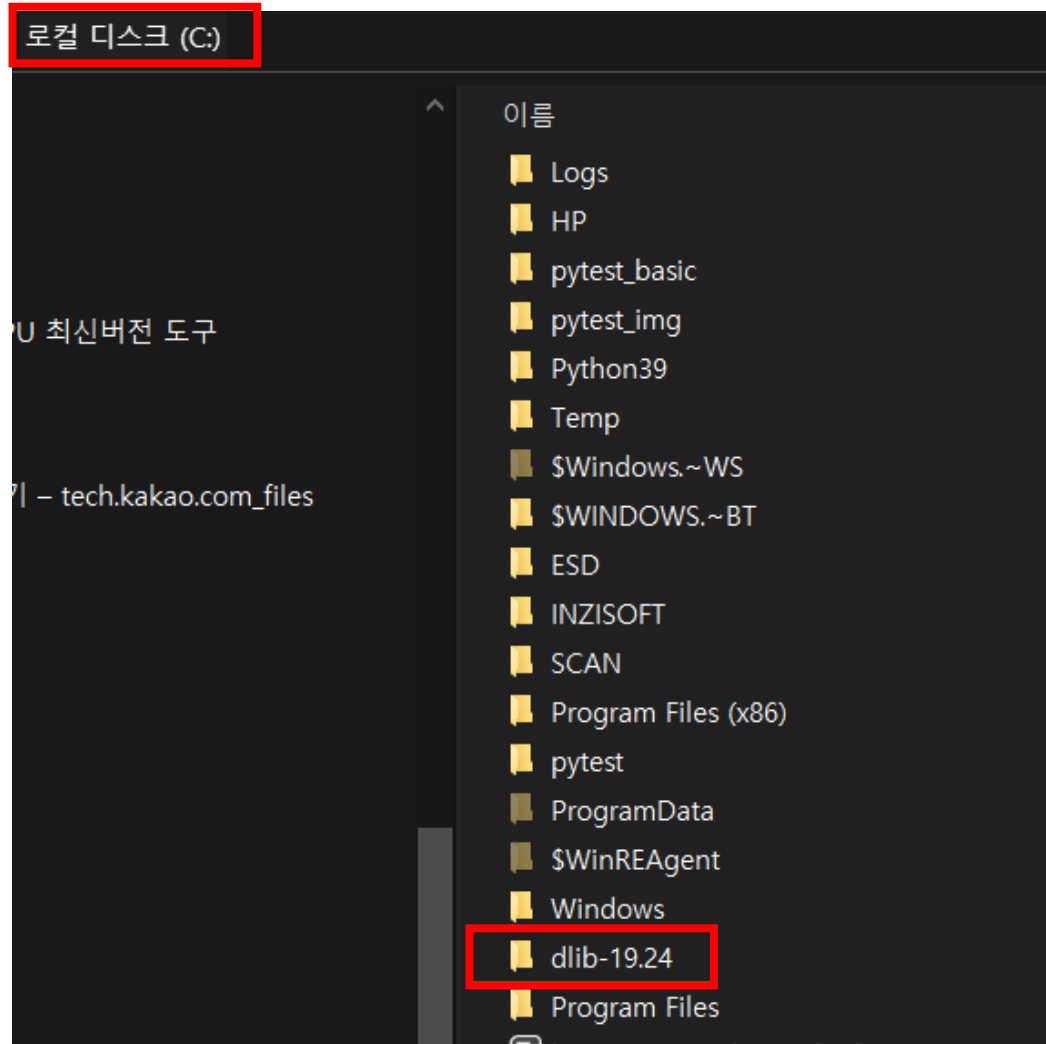
The screenshot shows the dlib website. On the left, there is a sidebar with navigation links: 'Help/Info' (with sub-links: Dlib Blog, Examples: C++, Examples: Python, FAQ, Home, How to compile, How to contribute, Index, Introduction, License, Python API, Suggested Books, Who uses dlib?), 'Current Release' (with sub-links: Change Log, Release Notes), and a red-bordered button labeled 'Download dlib ver.19.24'. On the right, under the heading 'Major Features', there are three bullet points: 'Documentation' (with sub-points: Unlike a lot of c every class and preconditions fc calling functions, Lots of example, I consider the a that isn't docum), 'High Quality Portab' (with sub-points: Good unit test, The library is te work on any PC, No other packa OS are needed, There is no inst: compile page fc, All operating sy small as possibl pure ISO standa), and 'Machine Learning A' (with sub-points: Deep Learning, Conventional SM, Reduced-rank n, Relevance vecto).

**Major Features**

- **Documentation**
  - Unlike a lot of c every class and preconditions fc calling functions
  - Lots of example
  - I consider the a that isn't docum
- **High Quality Portab**
  - Good unit test
  - The library is te work on any PC
  - No other packa OS are needed.
  - There is no inst: compile page fc
  - All operating sy small as possibl pure ISO standa
- **Machine Learning A**
  - Deep Learning
  - Conventional SM
  - Reduced-rank n
  - Relevance vecto

# C 드라이브에 저장

- 다운로드 받은 압축파일을 풀고,
- C:\W 루트에 넣는다
- 다른 위치에서는 실행이 잘 안될 수 있다





# CMake 다운로드

- 소스코드 관리 프로그램인 CMake를 다운로드한다
- <https://cmake.org/download>
- 자신의 OS에 맞는 버전을 찾아 다운로드

Source distributions:

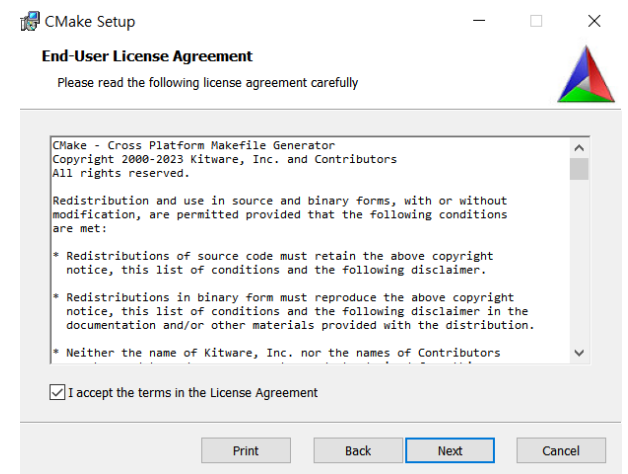
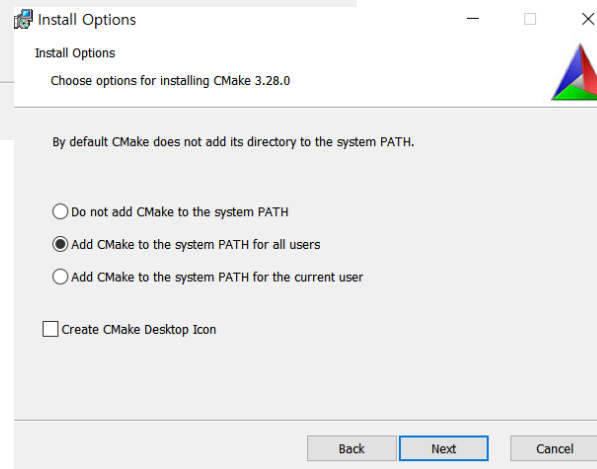
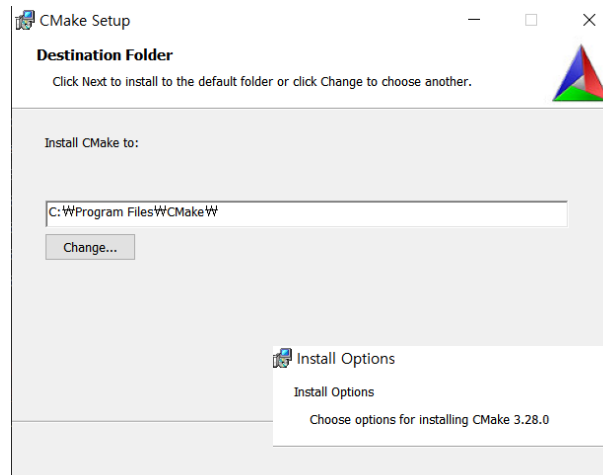
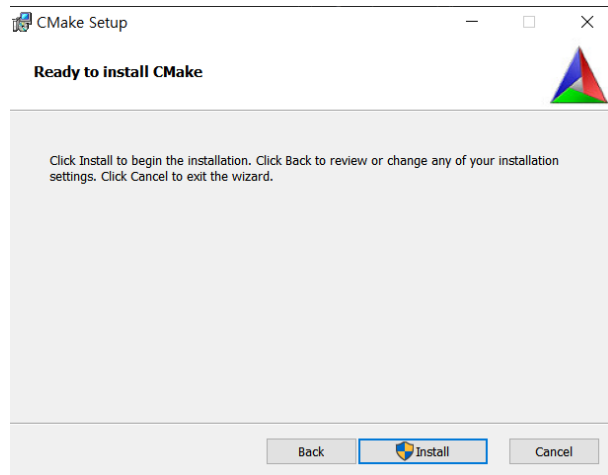
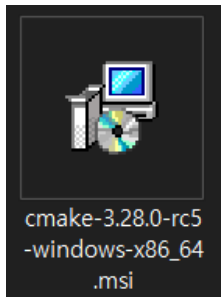
Platform	Files
Unix/Linux Source (has \n line feeds)	<a href="#">cmake-3.28.0-rc5.tar.gz</a>
Windows Source (has \r\n line feeds)	<a href="#">cmake-3.28.0-rc5.zip</a>

Binary distributions:

Platform	Files
Windows x64 Installer:	<a href="#">cmake-3.28.0-rc5-windows-x86_64.msi</a>
Windows x64 ZIP	<a href="#">cmake-3.28.0-rc5-windows-x86_64.zip</a>
Windows i386 Installer:	<a href="#">cmake-3.28.0-rc5-windows-i386.msi</a>
Windows i386 ZIP	<a href="#">cmake-3.28.0-rc5-windows-i386.zip</a>
Windows ARM64 Installer:	<a href="#">cmake-3.28.0-rc5-windows-arm64.msi</a>
Windows ARM64 ZIP	<a href="#">cmake-3.28.0-rc5-windows-arm64.zip</a>
macOS 10.13 or later	<a href="#">cmake-3.28.0-rc5-macos-universal.dmg</a>

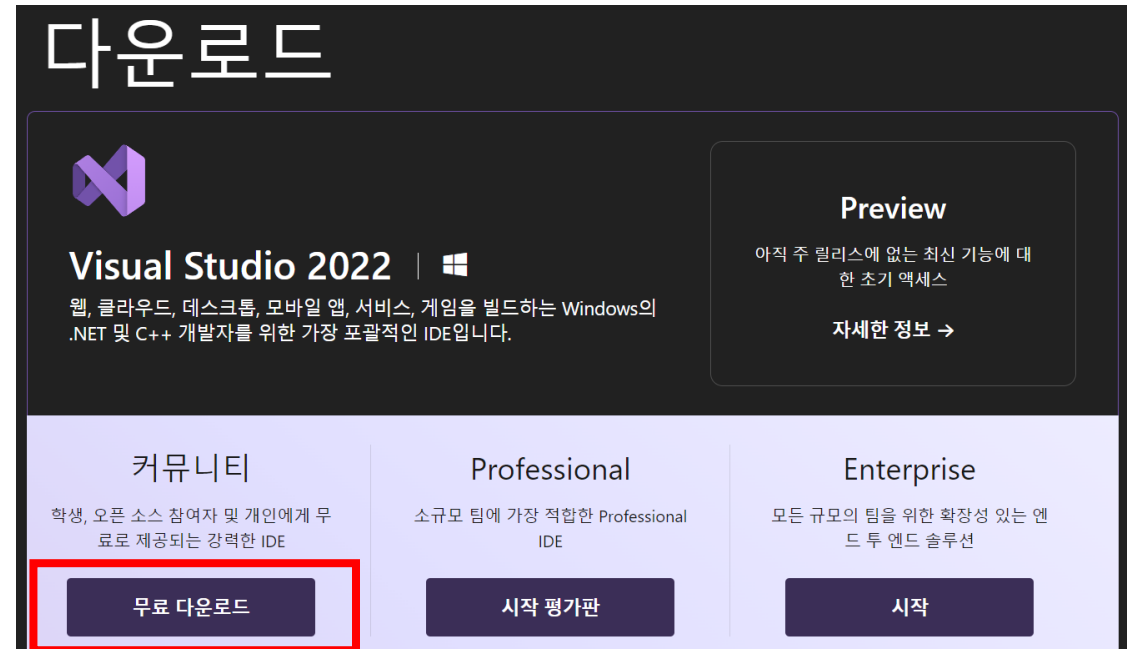
# CMake 설치

- 다운로드받은 파일을 실행하여 설치한다



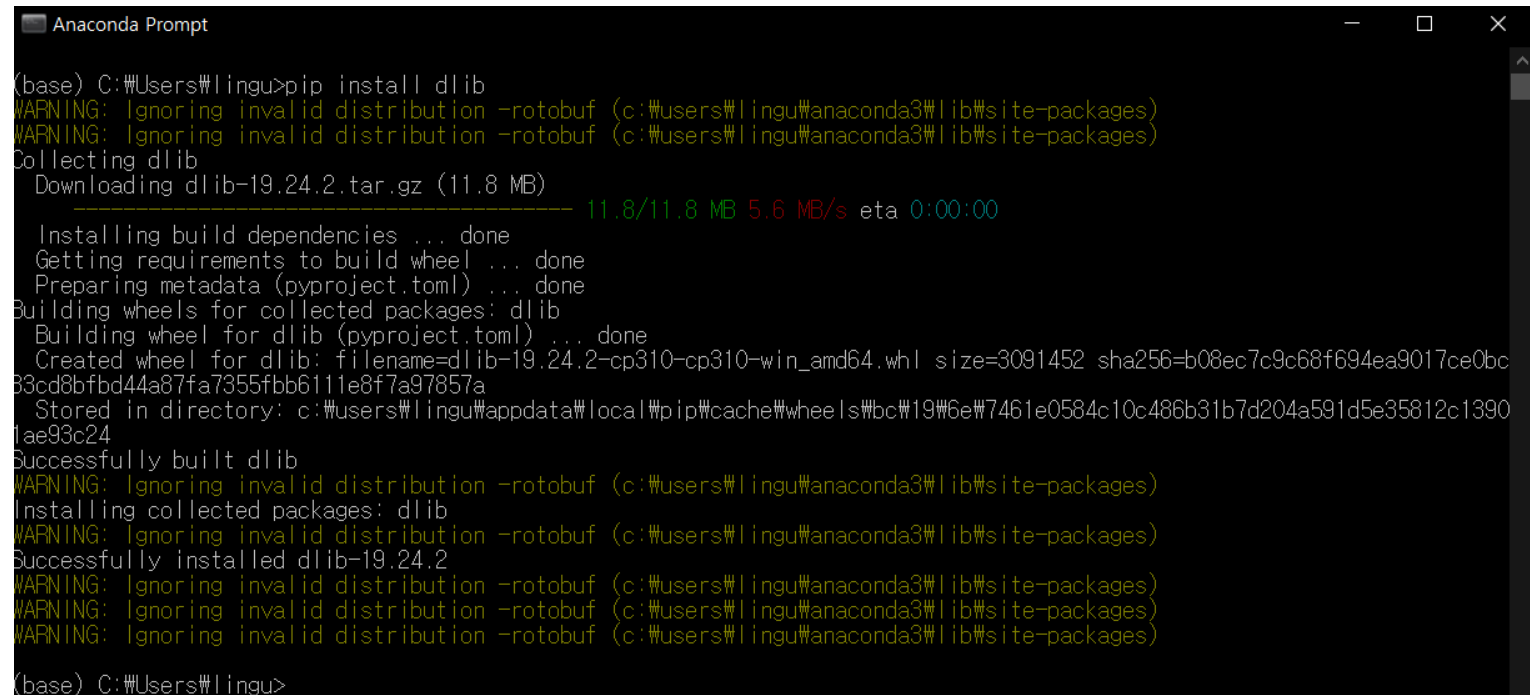
# Visual Studio 설치

- 다음의 경로에서 Visual Studio 커뮤니티 버전을 설치한다
- <https://visualstudio.microsoft.com/ko/downloads/>



# dlib 라이브러리 설치

- Anaconda Prompt에서 pip 명령어로 dlib을 설치한다
- pip install dlib
- 이 과정은 30분 이상 소요될 수 있다

A screenshot of the Anaconda Prompt terminal window. The window title is "Anaconda Prompt". The terminal shows the command "pip install dlib" being executed. The output includes several warning messages about ignoring invalid distributions, followed by the collection and download of dlib-19.24.2.tar.gz (11.8 MB). A progress bar shows the download is complete at 11.8/11.8 MB with a speed of 5.6 MB/s. The terminal then shows the installation of build dependencies, getting requirements, preparing metadata, and building wheels for dlib. The final output is "Successfully installed dlib-19.24.2".

```
(base) C:\Users\lingu>pip install dlib
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
Collecting dlib
  Downloading dlib-19.24.2.tar.gz (11.8 MB)
    ----- 11.8/11.8 MB 5.6 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: dlib
  Building wheel for dlib (pyproject.toml) ... done
  Created wheel for dlib: filename=dlib-19.24.2-cp310-cp310-win_amd64.whl size=3091452 sha256=b08ec7c9c68f694ea9017ce0bc
  Stored in directory: c:\users\lingu\appdata\local\pip\cache\wheels\bc\19\6e\7461e0584c10c486b31b7d204a591d5e35812c1390
  1ae93c24
Successfully built dlib
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
Installing collected packages: dlib
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
Successfully installed dlib-19.24.2
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lingu\anaconda3\lib\site-packages)
(base) C:\Users\lingu>
```

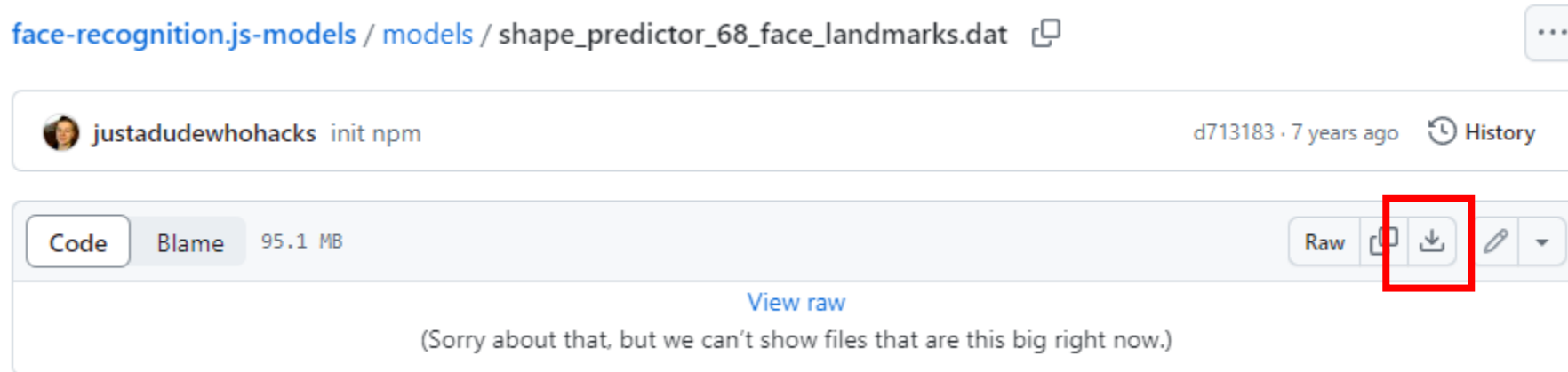
# Jupyter 실행

- Dlib은 IPython을 사용하는 Jupyter 또는 Visual Studio Code에서 더 잘 실행된다
- Jupyter에서 앞과 동일한 코드로 진행한다

68개 지점 사용하기

# Face Landmark

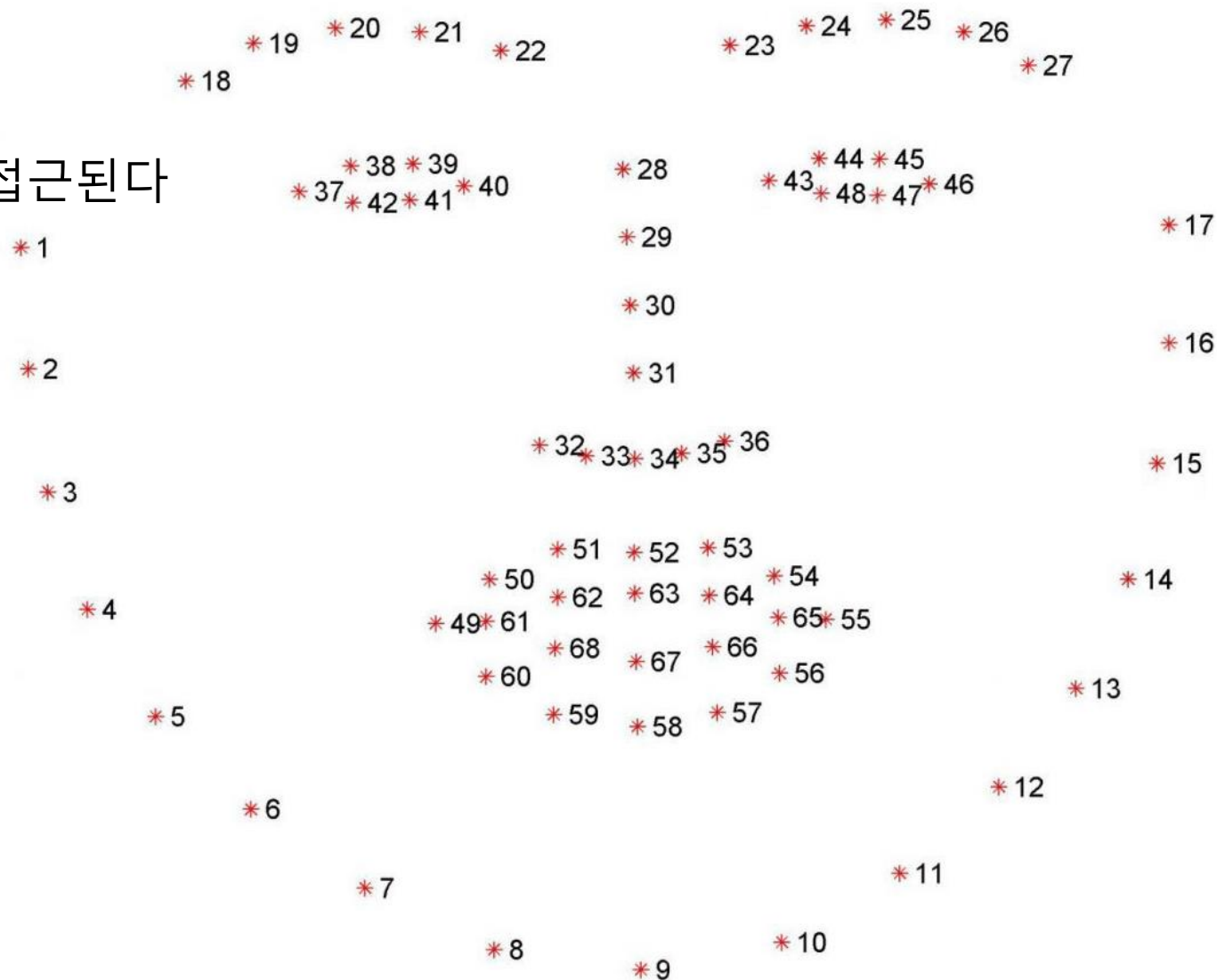
- dlib의 사용법을 좀 더 알아본다
- 이번에는 68개 지점을 사용해본다
- [https://github.com/justadudewhohacks/face-recognition.js-models/blob/master/models/shape\\_predictor\\_68\\_face\\_landmarks.dat](https://github.com/justadudewhohacks/face-recognition.js-models/blob/master/models/shape_predictor_68_face_landmarks.dat)



※ /content/gdrive/MyDrive/pytest\_img/dlib/shape\_predictor\_68\_face\_landmarks.dat 에도 있음

# Face Landmark

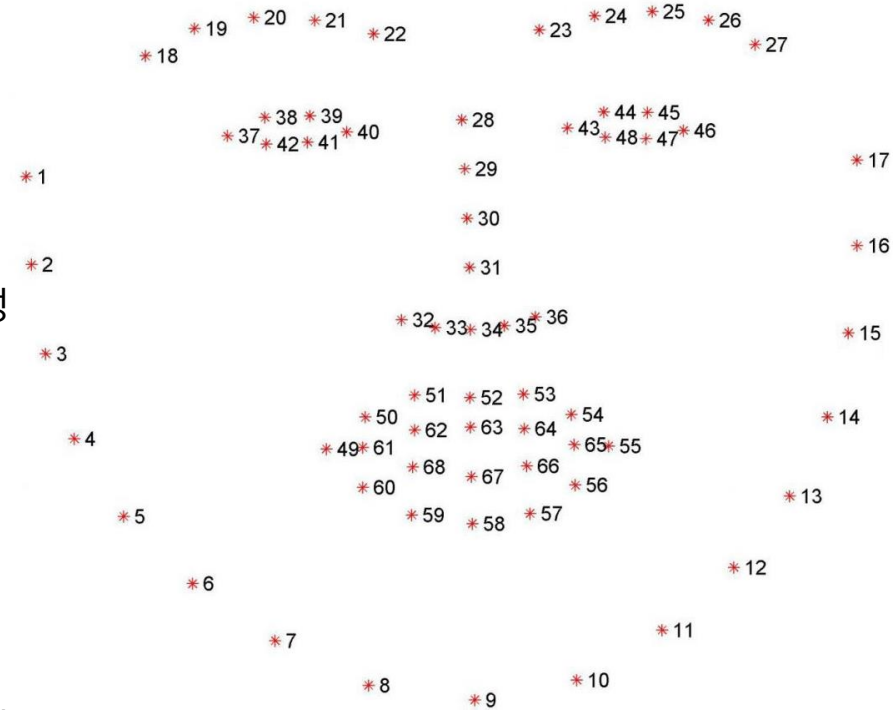
- dlib이 사용하는 얼굴의 68개 지점
- 1부터 시작하므로 index로는 0~67로 접근된다
- RIGHT\_EYE = 36~41
- LEFT\_EYE = 42~47
- EYES = 36~47
- MOUTH = 48~67
- NOSE = 27~35
- EYEBROWS = 17~26
- JAWLINE = 0~16
- ALL = 0~67 로 접근된다





# 주요 지점 설정

- 68개 지점을 이용하여 얼굴의 부위를 직접 코딩한다
- RIGHT\_EYE = list(range(36, 42))  
# 오른쪽 눈. 36~41까지 생성
- LEFT\_EYE = list(range(42, 48))  
# 왼쪽 눈. 42~47까지 생성
- EYES = list(range(36, 48))  
# 양쪽 눈. 36~47까지 생성
- MOUTH = list(range(48, 68))  
# 입. 48~67까지 생성
- NOSE = list(range(27, 36))  
# 코. 27~35까지 생성
- EYEBROWS = list(range(17, 27))  
# 양쪽 눈썹. 17~26까지 생성
- JAWLINE = list(range(0, 17))  
# 턱선. 0~16까지 생성
- ALL = list(range(0, 68))  
# 얼굴. 0~67까지 생성



# dlib으로 이미지 읽기

- 이번에는 dlib으로 이미지를 읽어서 OpenCV로 이미지를 출력해 본다
- dlib은 이미지를 RGB로 읽는데, OpenCV는 BGR로 처리하므로 변환해주어야 한다
- import dlib
- import matplotlib.pyplot as plt
- import matplotlib.patches as patches
- img\_path = '/content/gdrive/MyDrive/pytest\_img/img\_align\_celeba\_small/202485.jpg'
- sp = dlib.shape\_predictor("/content/gdrive/MyDrive/pytest\_img/dlib/shape\_predictor\_68\_face\_landmarks.dat")
- detector = dlib.get\_frontal\_face\_detector()      # 얼굴 영역 인식 함수      # 얼굴의 68개 지점 인식 모델 로드

# dlib으로 이미지 읽기

# dlib으로 이미지 읽기

- `img = dlib.load_rgb_image(img_path)`
- `img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)`

- `cv2.imshow(img)`
- `cv2.waitKey(0)`
- `cv2.destroyAllWindows()`

# 이미지를 RGB로 읽음

# CV2에서 처리하기 위해 다시 BGR로 변환

# CV2로 이미지 출력



# 크기 조절 및 얼굴 개수 파악

- 원본 이미지가 작기 때문에 사이즈를 키우고,
- 이미지의 얼굴이 몇 개인지 파악한다

- `img_result = img.copy()` # 사본 생성
- `img_resize = cv2.resize(img_result, dsize=(350, 450))` # 크기 조절
- `gray = cv2.cvtColor(img_resize, cv2.COLOR_BGR2GRAY)` # 더 잘 찾도록 흑백 이미지로 변환
- `dets = detector(gray, 1)` # 이미지에서 얼굴 영역을 찾는다
- `if len(dets) == 0:` # 얼굴영역의 갯수가 0일 경우  
    `print('발견된 얼굴이 없습니다!')`
- `print("발견된 얼굴의 수:", len(dets))` # 1

# 얼굴 좌표 출력

- 이미지에서 얼굴의 주요 부위를 찾아 좌표를 출력한다
- `import numpy as np`
- ```
for det in dets:                                # 얼굴 감지 결과(dets)에서 각 검출(det) 결과를 순회
    # shape predictor에 의해 감지된 랜드마크 포인트를 x, y 좌표로 계산하여 points 변수에 저장
    points = np.matrix([[p.x, p.y] for p in sp(gray, det).parts()])
    show_parts = points[ALL]                      # 0~67의 얼굴 포인트를 show_parts에 저장
```

얼굴 주요 지점 인식 모델(sp)를 사용하여 각 얼굴영역에서 얼굴 랜드마크를 찾고, 이 포인트들의 x, y 좌표를 NumPy 행렬로 생성
- ```
for (i, point) in enumerate(show_parts):
    x = point[0, 0]                              # x 좌표 (각 행의 첫 번째)
    y = point[0, 1]                              # y 좌표 (각 행의 두 번째)
    cv2.circle(img_resize, (x, y), 1, (0, 0, 255), -1) # 빨간색으로 점을 찍는다
    cv2.putText(img_resize, f"{i+1}", (x, y-2), cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0, 255, 0), 1) # 점에 녹색으로 숫자 표시
```

image, text, text position, font, font scale, color, thickness

# 얼굴 좌표 출력

- cv2.imshow(img\_resize)
- cv2.waitKey(0)
- cv2.destroyAllWindows()
- points 변수는 다음과 같이 x, y 좌표로 되어 있다

```
matrix([[ 82, 239],  
        [ 85, 265],  
        [ 88, 289],  
        [ 91, 314],  
        [ 98, 338],  
        [110, 358],  
        [127, 374],  
        [147, 385],  
        [171, 387],  
        [195, 384],  
        [218, 375],  
        [239, 360],  
        [254, 342])
```



# 눈을 통한 졸음 상태 확인

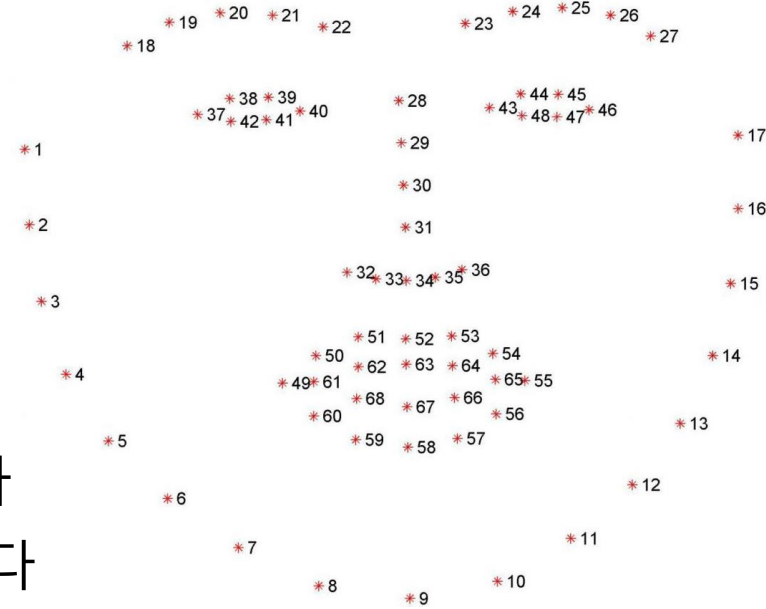
- 얼굴 좌표가 확인이 되었다면 얼굴의 상태를 알 수 있다
- 오른쪽 눈과 왼쪽 눈 좌표를 사용해 이들의 개방 정도를 측정한다
- 눈을 감고 있는지 여부는 눈 종횡비(EAR; Eye Aspect Ratio)를 사용한다
- 눈 종횡비는 눈의 세로 길이와 가로 길이의 비율로 계산되며, 공식은 다음과 같다

$$EAR = \frac{\|p2 - p6\| + \|p3 - p5\|}{2 * \|p1 - p4\|}$$

p1~p6은 눈 위치 랜드마크

- 일반적으로 EAR 값이 0.2~0.3 이하로 떨어지면 눈을 감은 것으로 간주된다

오른쪽 눈의 경우 p2(38), p6(42), p3(39), p5(41), p1(37), p4(40)

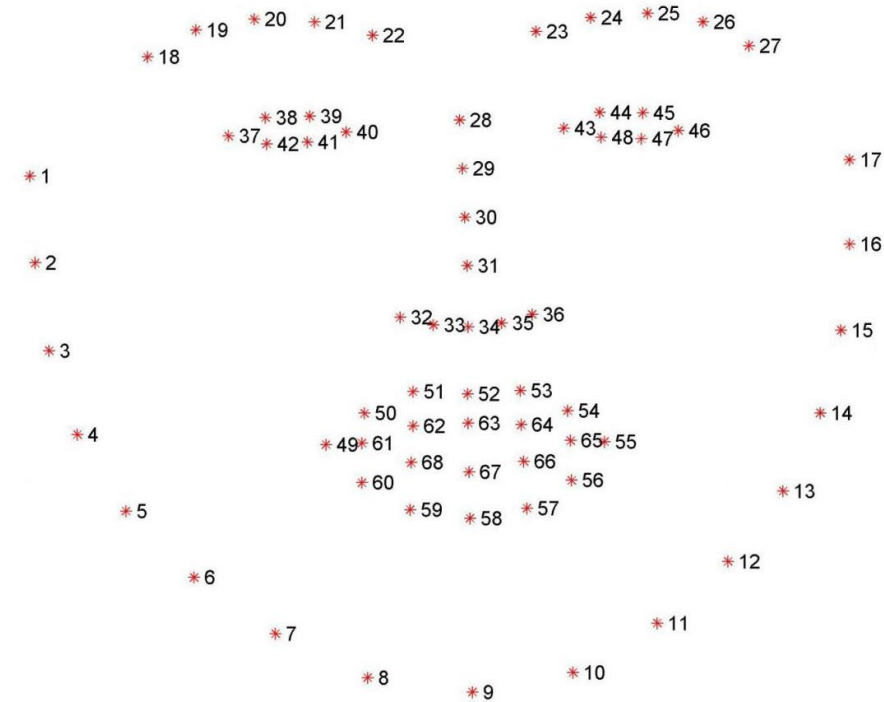


# EAR 값 계산 함수

- EAR 계산 공식은 다음과 같이 구현할 수 있다
- `def eye_aspect_ratio(eye_points):`  
    `A = np.linalg.norm(eye_points[1] - eye_points[5])`  
    `B = np.linalg.norm(eye_points[2] - eye_points[4])`  
    `C = np.linalg.norm(eye_points[0] - eye_points[3])`  
    `EAR = (A + B) / (2.0 * C)`  
  
    `return EAR`

`np.linalg.norm`은 벡터의 길이를 계산한다

오른쪽 눈의 경우 1-(38), 5-(42), 2-(39), 4-(41), 0-(37), 3-(40)





# 상태 판단

- for det in dets:  
    points = np.array([[p.x, p.y] for p in sp(gray, det).parts()])

얼굴 주요 지점 인식 모델(sp)를 사용하여 각 얼굴영역에서 얼굴 랜드마크를 찾고,  
이 포인트들의 x, y 좌표를 NumPy 배열로 생성

left\_eye\_EAR = eye\_aspect\_ratio(points[LEFT\_EYE])

# 왼쪽 눈 EAR

right\_eye\_EAR = eye\_aspect\_ratio(points[RIGHT\_EYE])

# 오른쪽 눈 EAR

- if left\_eye\_EAR < 0.25:  
    print("왼쪽 눈을 감고 있음")  
else:  
    print("왼쪽 눈을 뜨고 있음")
- if right\_eye\_EAR < 0.25:  
    print("오른쪽 눈을 감고 있음")  
else:  
    print("오른쪽 눈을 뜨고 있음")

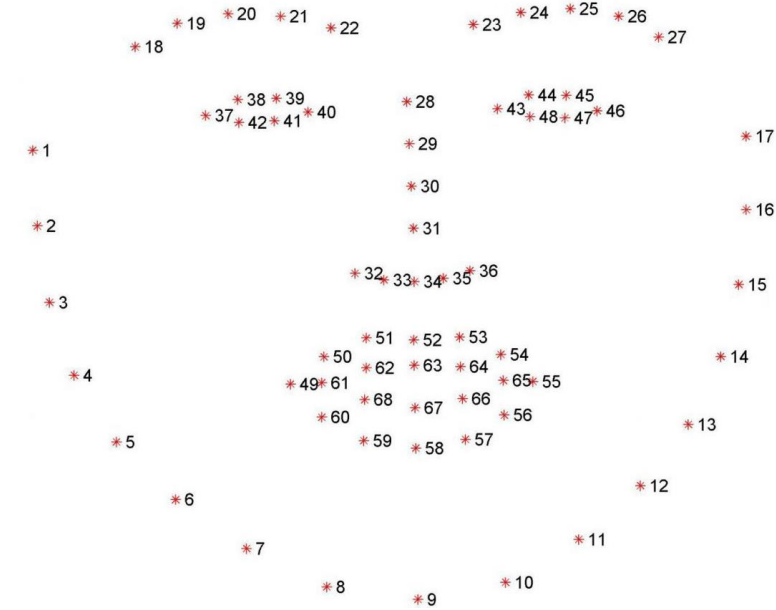
왼쪽 눈을 뜨고 있음  
오른쪽 눈을 뜨고 있음

# 입을 통한 졸음 상태 확인

- 입을 얼마나 벌리고 있는지도 알 수 있다
- 입의 상단과 하단 사이의 거리를 입의 높이로 계산한다
- 입의 양끝 사이의 거리를 입의 너비로 계산한다
- 입 종횡비(MAR; Mouth Aspect Ratio) 공식은 다음과 같다

$$MAR = \frac{\|p_{top} - p_{bottom}\|}{\|p_{left} - p_{right}\|}$$

- MAR 값은 약 0.3~0.4 이상이면 벌리고 있는 것으로 간주할 수 있다



top(52), bottom(58), left(49), right(55)

# MAR 값 계산 함수

- MAR 계산 공식은 다음과 같이 구할 수 있다

- def mouth\_aspect\_ratio(mouth\_points):

top = mouth\_points[3]

bottom = mouth\_points[9]

left = mouth\_points[0]

right = mouth\_points[6]

MAR = np.linalg.norm(top - bottom) / np.linalg.norm(left - right)

return MAR

# 랜드마크 52 표시

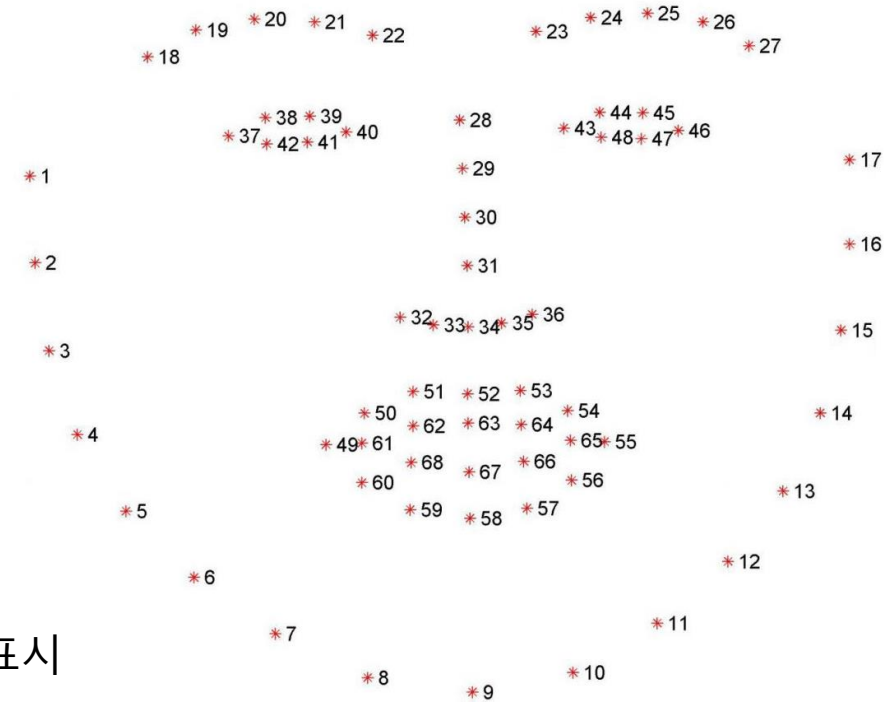
# 랜드마크 58 표시

# 랜드마크 49 표시

# 랜드마크 55 표시

MOUTH = list(range(48, 68))  
52가 index 3 (48, 49, 50, 51, 52)

# 입. 48~67까지 생성



# 상태 판단

- for det in dets:

```
    points = np.array([[p.x, p.y] for p in sp(gray, det).parts()])
```

```
    mar = mouth_aspect_ratio(points[MOUTH])           # 입의 MAR 계산
```

- if mar > 0.35:

```
    print("입을 벌리고 있음")
```

```
else:
```

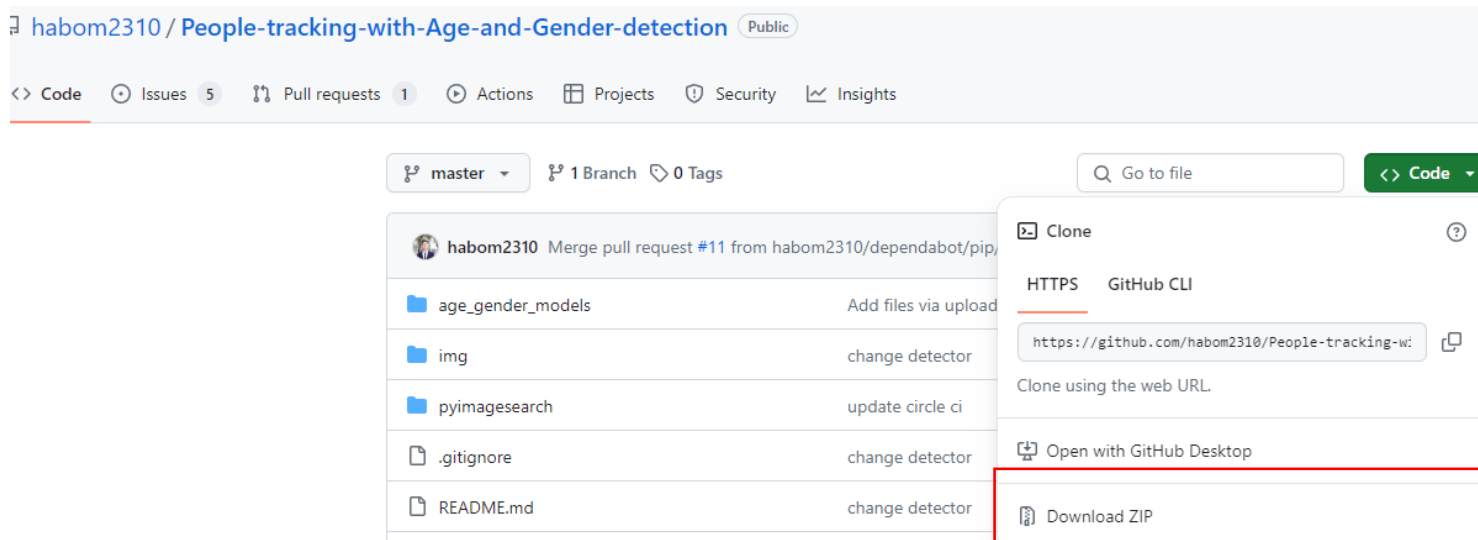
```
    print("입을 다물고 있음")
```

입을 다물고 있음

나이와 성별

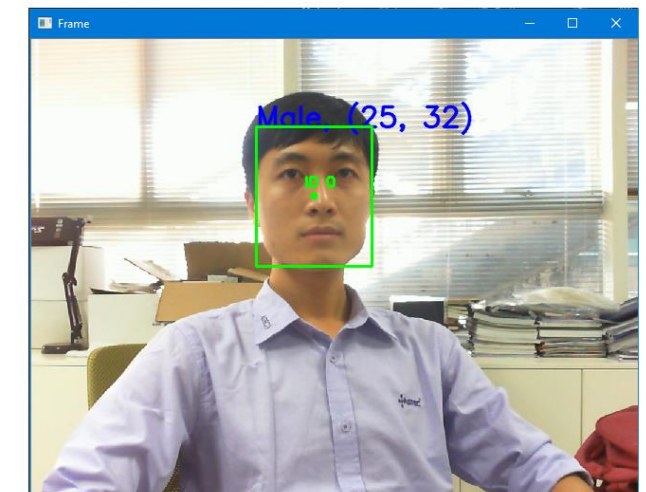
# 나이와 성별 예측 모델

- 다음의 깃헙에 나이와 성별을 예측할 수 있는 모델이 있다
- <https://github.com/habom2310/People-tracking-with-Age-and-Gender-detection>
- 이것을 cv2.dnn 모듈을 사용하여 로드하면 이미지의 나이와 성별을 예측할 수 있다
- cv2.dnn 모듈은 OpenCV에서 딥러닝 네트워크를 로드하고 실행할 수 있는 모듈








People-tracking-with-Age-and-Gender-detection

A combination between people tracking and age and gender detection



# 모델 파일

- 다음의 네 개 파일을 저장한다
- age\_gender\_models 폴더에 있는,

 age_net.caffemodel	2022-07-18 오전 10:25	CAFFEMODEL 파일	44,592KB
 deploy_age.prototxt	2022-07-18 오전 10:25	PROTOTXT 파일	3KB
 deploy_gender.prototxt	2022-07-18 오전 10:25	PROTOTXT 파일	3KB
 gender_net.caffemodel	2022-07-18 오전 10:25	CAFFEMODEL 파일	44,580KB
 model_CNN_V2.h5	2022-07-18 오전 10:25	H5 파일	22,461KB

※ /content/gdrive/MyDrive/pytest\_img/dlib/ 에도 위의 4개 파일이 있음

# 구간 정의

- import dlib
- import cv2

# 나이 구간 정의

- age\_list = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']

# 성별 구분 정의

- gender\_list = ["Male", "Female"]



# 인식 함수

프레임워크별 모델 로드 함수:  
cv2.dnn.readNetFromCaffe()  
cv2.dnn.readNetFromTensorflow()  
cv2.dnn.readNetFromTorch()

# 얼굴 영역 인식 함수

- detector = dlib.get\_frontal\_face\_detector()

# 나이 인식 함수. Caffe 프레임워크에서 훈련된 모델을 로드

- age\_detector =  
cv2.dnn.readNetFromCaffe("/content/gdrive/MyDrive/pytest\_img/dlib/deploy\_age.proto  
txt", "/content/gdrive/MyDrive/pytest\_img/dlib/age\_net.caffemodel")

# 성별 인식 함수

- gender\_detector =  
cv2.dnn.readNetFromCaffe("/content/gdrive/MyDrive/pytest\_img/dlib/deploy\_gender.pr  
ototxt", "/content/gdrive/MyDrive/pytest\_img/dlib/gender\_net.caffemodel")

# 원본 이미지 출력

- 원본 이미지를 출력해본다
- `from google.colab.patches import cv2_imshow`
- `img = cv2.imread('/content/gdrive/MyDrive/pytest_img/img_align_celeba_small/202485.jpg')`
- `img_resize = cv2.resize(img, dsize=(350, 450))`      # 크기 조절
- `cv2_imshow(img_resize)`      # 이미지 그리기
- `cv2.waitKey(0)`
- `cv2.destroyAllWindows()`



# 흑백 이미지로 변환

- `gray = cv2.cvtColor(img_resize, cv2.COLOR_BGR2GRAY)` # 정확한 인식을 위해 흑백 이미지로 변환
- `dets = detector(gray, 1)`
- `if len(dets) == 0:` # 얼굴영역의 갯수가 0일 경우  
    `print('발견된 얼굴이 없습니다!')`
- `print("발견된 얼굴의 수:", len(dets))` # 1

# 예측 수행 (1/3)

- for det in dets:    # 모든 얼굴에 대하여
  - x1, y1, x2, y2 = det.left(), det.top(), det.right(), det.bottom()    # 얼굴 box 좌표
  - face\_img = img\_resize[y1:y2, x1:x2].copy()                            # 원본 이미지에서 얼굴 영역만 copy
  - 
  - # 이미지 전처리를 수행하여 blob(Binary Large Object) 타입으로 변환
  - # scale factor: 이미지의 픽셀값 조정을 위한 스케일링 인자. 1은 원본 픽셀값. 0.00392를 사용하면 이미지가 0~255일 때 0~1 범위로 조정됨
  - # 신경망 입력을 위해 조정하는 고정된 이미지 크기. (227, 227)
  - # mean: 각 색상 채널(BRG)에 대해 빼줄 평균값. 모델 학습 시 사용된 평균값과 일치해야 하므로 고정값
  - # swapRB: 딥러닝 모델이 사용한 것과 같게 BGR 순서를 사용. True는 RGB.
  - blob = cv2.dnn.blobFromImage(face\_img, scalefactor=1, size=(227, 227), mean=(78.4263377603, 87.7689143744, 114.895847746), swapRB=False)

# 예측 수행 (2/3)

# 계속 for문 안쪽

# 나이 예측

age\_detector.setInput(blob)

age\_preds = age\_detector.forward()

age = age\_list[age\_preds[0].argmax()]

# 나이 예측 모델의 입력으로 설정

# 모델을 실행하여 예측 결과를 얻음

# 가장 높은 확률을 선택

# 성별 예측

gender\_detector.setInput(blob)

gender\_preds = gender\_detector.forward()

gender = gender\_list[gender\_preds[0].argmax()]

# 예측 수행 (3/3)

# 계속 for문 안쪽

```
cv2.rectangle(img_resize, (x1, y1), (x2, y2), (255, 255, 255), 2)
```

```
text = f'{gender} {age}' # 텍스트 설정
```

```
cv2.putText(img_resize, text, (x1, y1), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 0, 0),  
thickness=10) # 글자 배경 (검정)
```

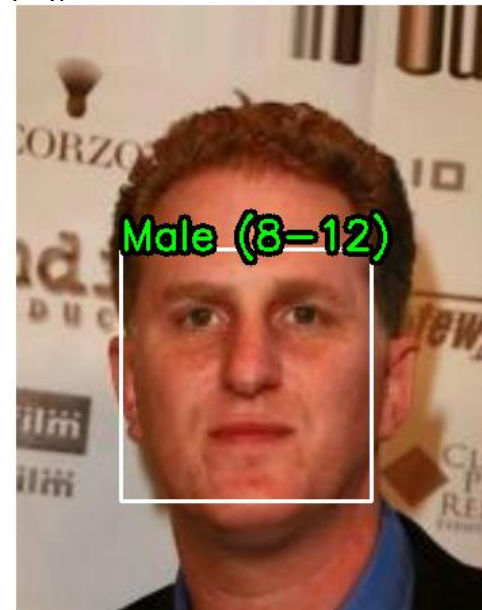
```
cv2.putText(img_resize, text, (x1, y1), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0),  
thickness=2) # 글자 (녹색)
```

```
cv2.imshow(img_resize)
```

```
print("\n\n")
```

# for문 나옴

- cv2.waitKey(0)
- cv2.destroyAllWindows()



성별은 맞았으나, 나이는 맞지 않았다

# 예측 수행 부분 전체 코드

```
for det in dets: # 모든 얼굴에 대하여
    x1, y1, x2, y2 = det.left(), det.top(), det.right(), det.bottom() # 얼굴 box 좌표

    face_img = img_resize[y1:y2, x1:x2].copy() # 원본 이미지에서 얼굴 영역만 copy

    # scale factor: 이미지의 픽셀값 조정을 위한 스케일링 인자. 1은 원본 픽셀값
    # 신경망 입력을 위해 조정하는 고정된 이미지 크기. (227, 227)
    # mean: 각 색상 채널(BRG)에 대해 빼줄 평균값. 모델 학습 시 사용된 평균값과 일치해야 하므로 고정값
    # swapRB: 딥러닝 모델이 사용한 것과 같게 RGB 순서를 사용
    blob = cv2.dnn.blobFromImage(face_img, scalefactor=1, size=(227, 227), mean=(78.4263377603, 87.7689143744, 114.895847746), swapRB=False)

    # 나이 예측
    age_detector.setInput(blob)
    age_pres = age_detector.forward()
    age = age_list[age_pres[0].argmax()]

    # 성별 예측
    gender_detector.setInput(blob)
    gender_preds = gender_detector.forward()
    gender = gender_list[gender_preds[0].argmax()]

    cv2.rectangle(img_resize, (x1, y1), (x2, y2), (255, 255, 255), 2)
    text = f'{gender} {age}'
    cv2.putText(img_resize, text, org=(x1, y1), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 0, 0), thickness=10) # 글자 배경 (검정)
    cv2.putText(img_resize, text, org=(x1, y1), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0), thickness=2) # 글자 (녹색)

    cv2.imshow(img_resize)
    print("\n\n")

cv2.waitKey(0)
cv2.destroyAllWindows()
```