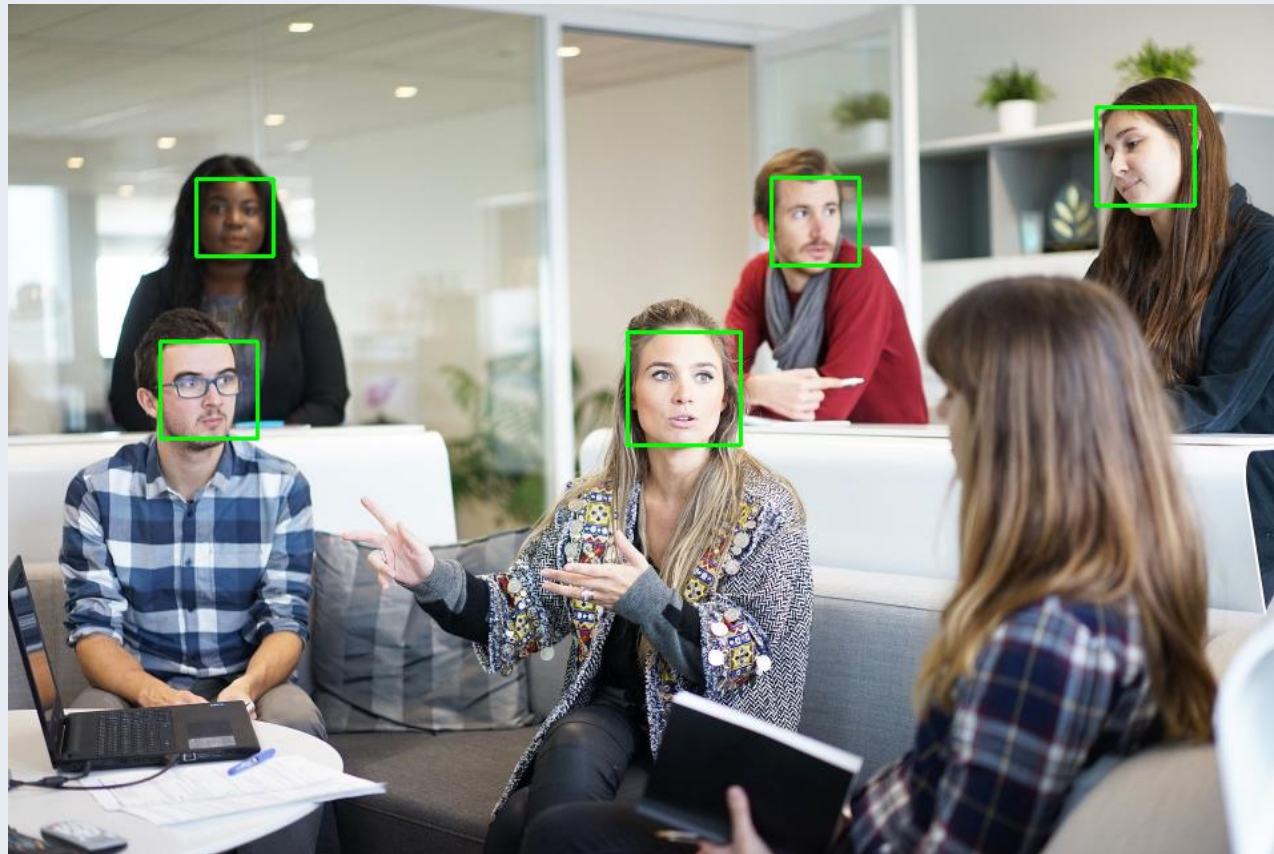
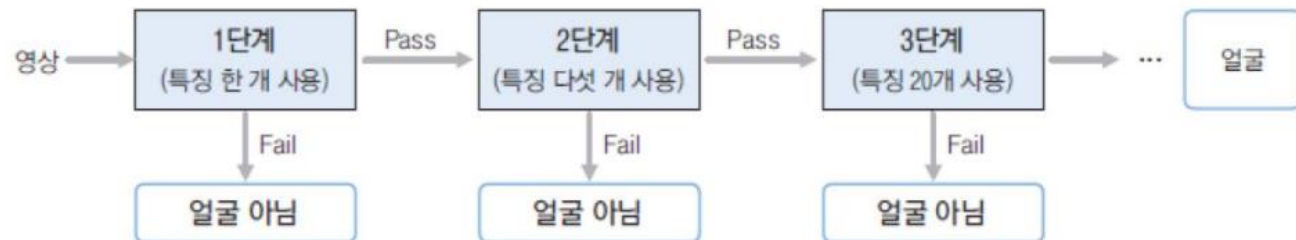
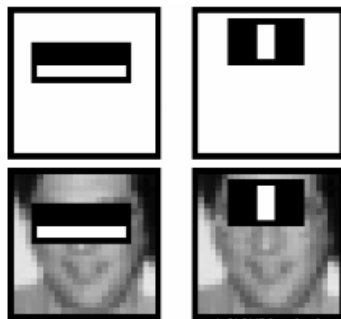
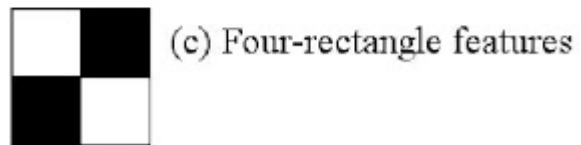
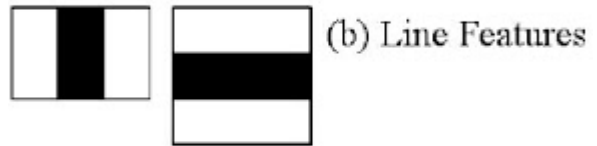


얼굴 탐지 (Face Detection)

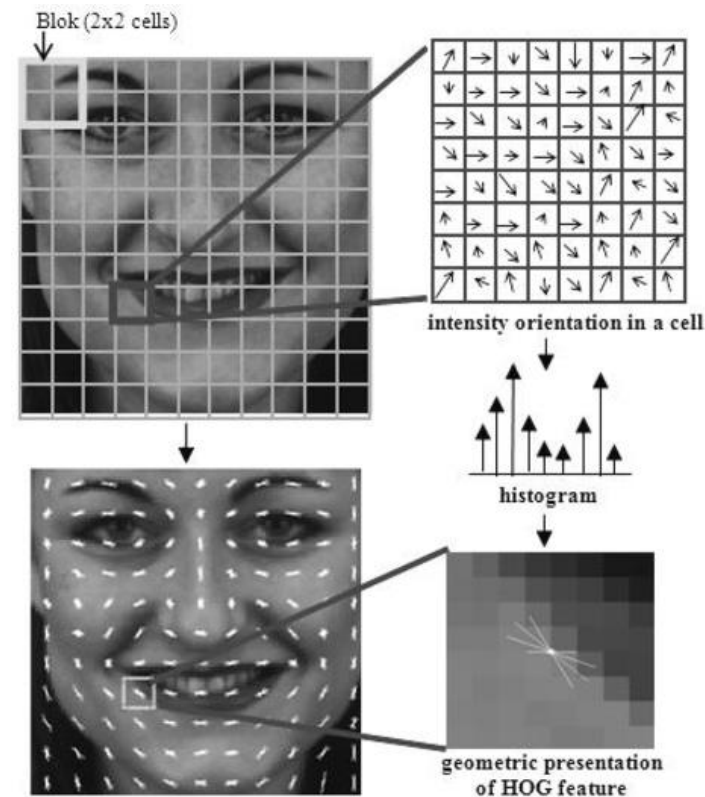
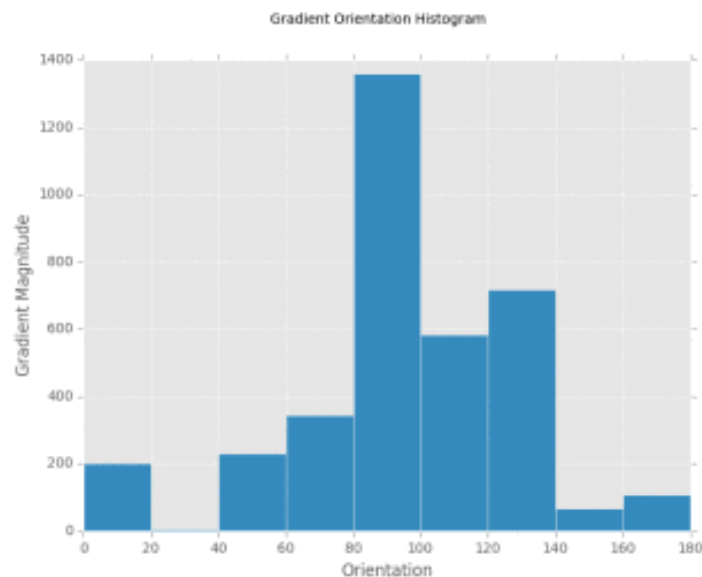
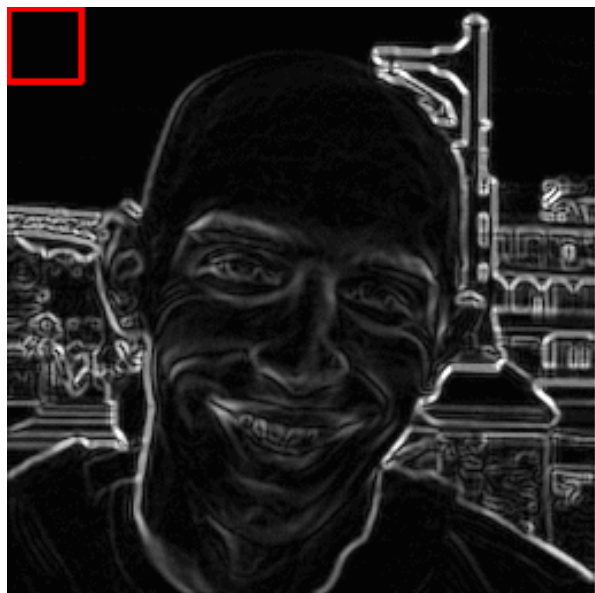


CASCADE CLASSIFIER



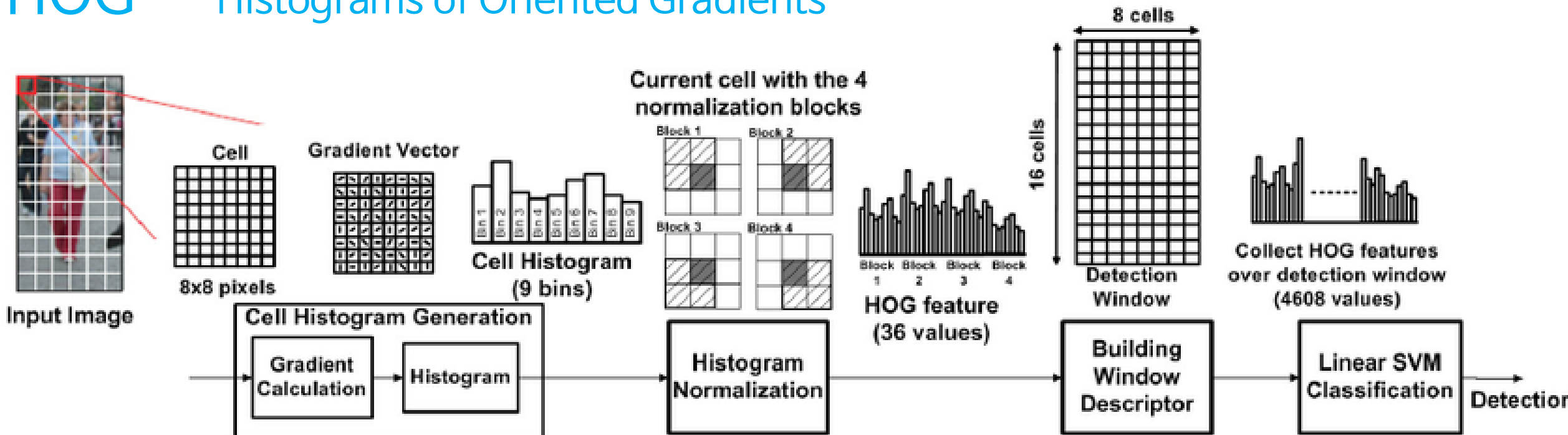
HOG - Histograms of Oriented Gradients

HOG는 픽셀값의 변화로 파악할 수 있는 영상 밝기 변화의 방향을 그래디언트(gradient)로 표현하고, 이로부터 객체의 형태를 찾아낼 수 있습니다. 얼굴 탐색, 보행자 검출 등에 활용할 수 있습니다.



3. Histogram of oriented gradient extraction from face.

HOG - Histograms of Oriented Gradients



1. input image에서 cell을 나누고, cell에서 각 pixel마다 gradient의 크기 값과 방향을 구한다.
2. 구한 gradient 방향을 bin으로 크기 값의 histogram을 만들고
3. cell을 4개를 묶어서 block을 만들어 bin 갯수의 * 4가 되는데, 이것을 vector라고 했을 때, 이 vector의 절대 값을 나누어준다. (L1, L2 norm)으로 즉 정규화시켜준다.
4. 각 block은 36개 feature를 갖게 되고, image 전체에서의 block 갯수 * 36이 되면 이 image를 나타내는 feature vector가 된다.
5. 이미지(feature vector)과 label로 이루어진 쌍(example)은 선형 SVM 분류기를 통해 학습된다.
6. test 이미지를 주면, feature vector를 가지고 detection window에서 내가 찾고자 하는 label이 있는지를 확인하고, 그 window의 위치를 알려주는 것이다.

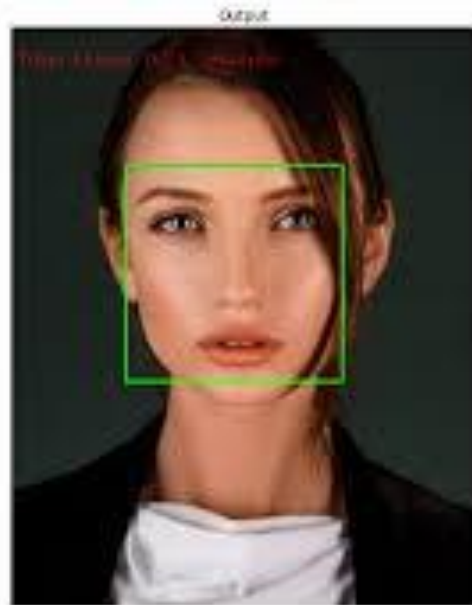


<http://dlib.net/>

Dlib 라이브러리에서는 얼굴 탐색을 위해 HOG특성을 활용하거나 또는 학습된 CNN모델을 사용할 수 있습니다.

기본적으로 HOG 특성을 활용하므로 `dlib.get_frontal_face_detector()`를 사용하여 얼굴 검출 합니다.

CNN 학습 결과를 사용하려면 `dlib.cnn_face_detection_model_v1` 과 같은 클래스를 사용합니다.



(a)

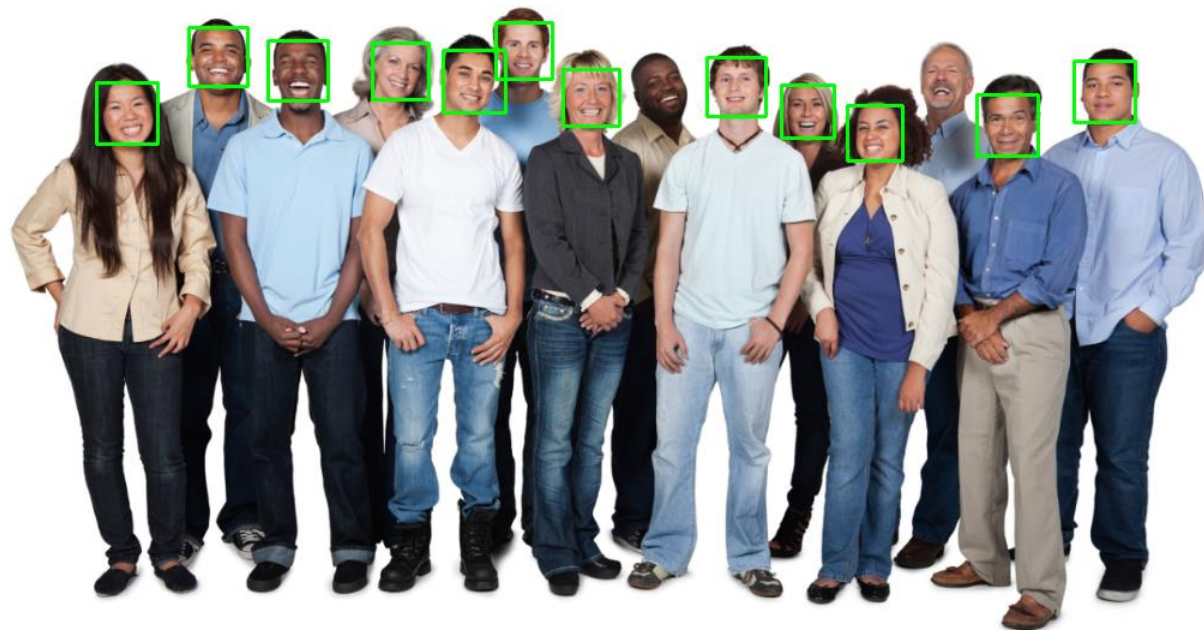


(b)

얼굴 탐지 (Face Detection)



face_detection.ipynb

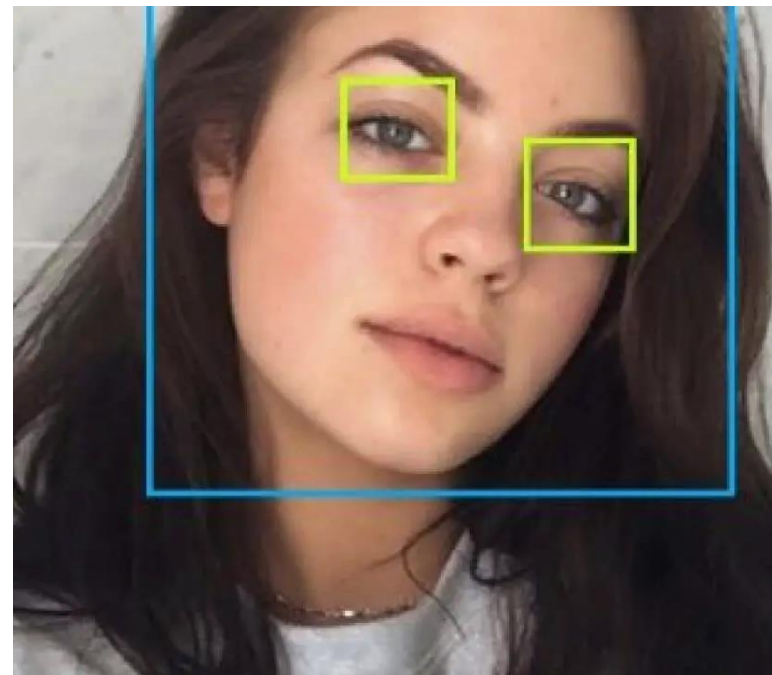


웹캠에서 얼굴 탐지



face_detector.py

```
1 import cv2
2
3 face_detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
4 eye_detector = cv2.CascadeClassifier("haarcascade_eye.xml")
5 cap = cv2.VideoCapture(0)
6
7 while True:
8     # capture video frame
9     ret, frame = cap.read()
10    gray_image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
11    detections = face_detector.detectMultiScale(
12        gray_image, minSize=(100, 100), minNeighbors=5
13    )
14
15    # draw a rectangle around the faces
16    for x, y, w, h in detections:
17        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 255, 0), 2)
18        rec_gray = gray_image[y : y + h, x : x + w]
19        rec_color = frame[y : y + h, x : x + w]
20        eyes = eye_detector.detectMultiScale(rec_gray)
21        for x1, y1, w1, h1 in eyes:
22            cv2.rectangle(rec_color, (x1, y1), (x1 + w1, y1 + h1), (0, 127, 255), 2)
23
24    # display the resulting frame
25    cv2.imshow("Face Recognition", frame)
26    if cv2.waitKey(1) & 0xFF == ord("q"):
27        break
28
29 # release the video capture
30 cap.release()
31 cv2.destroyAllWindows()
```






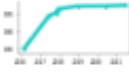












Face Detection Benchmark

Benchmarks

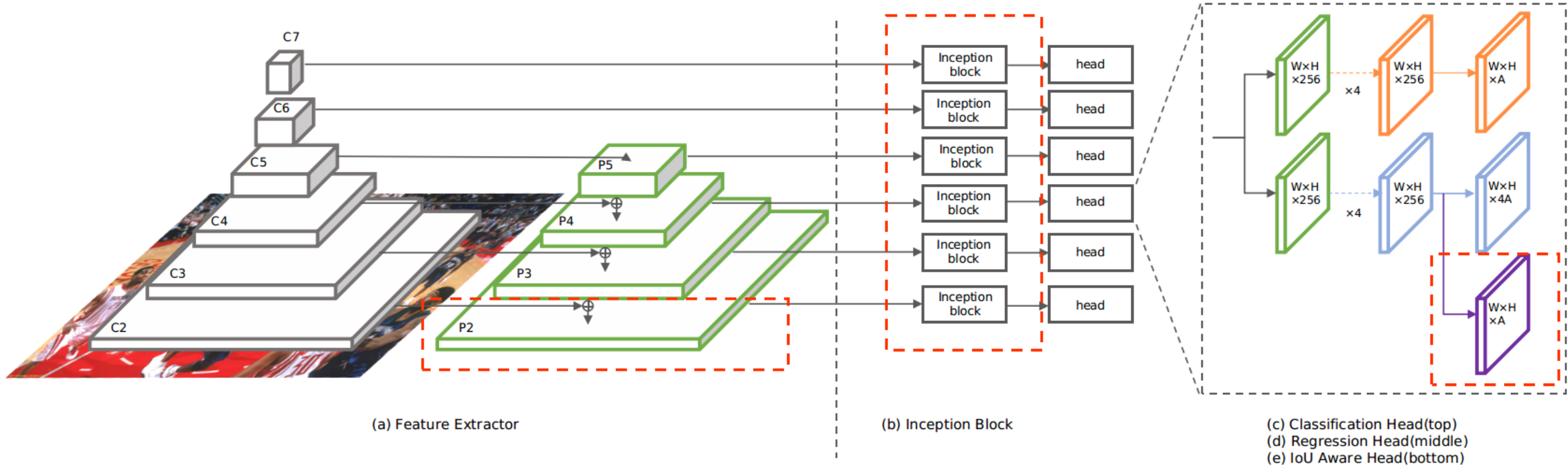
[Add a Result](#)

These leaderboards are used to track progress in Face Detection

Trend	Dataset	Best Model	Paper	Code	Compare
	WIDER Face (Hard)	TinaFace(ResNet-50)			See all
	WIDER Face (Medium)	ASFD			See all
	WIDER Face (Easy)	ASFD			See all
	FDDB	DSFD			See all
	Annotated Faces in the Wild	SRN			See all
	PASCAL Face	SRN			See all

<https://paperswithcode.com/task/face-detection/latest>

Face Detection Benchmark



Face Detection Benchmark

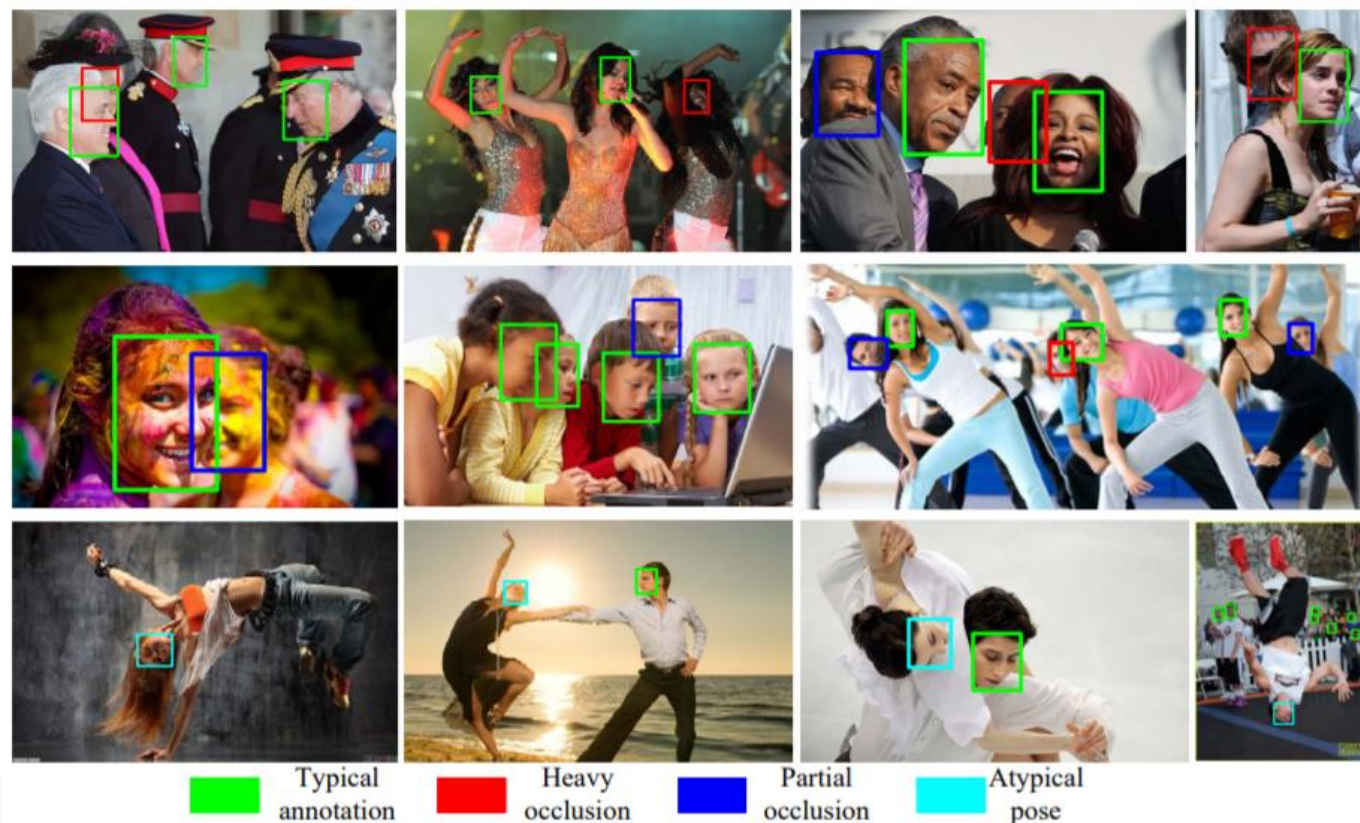
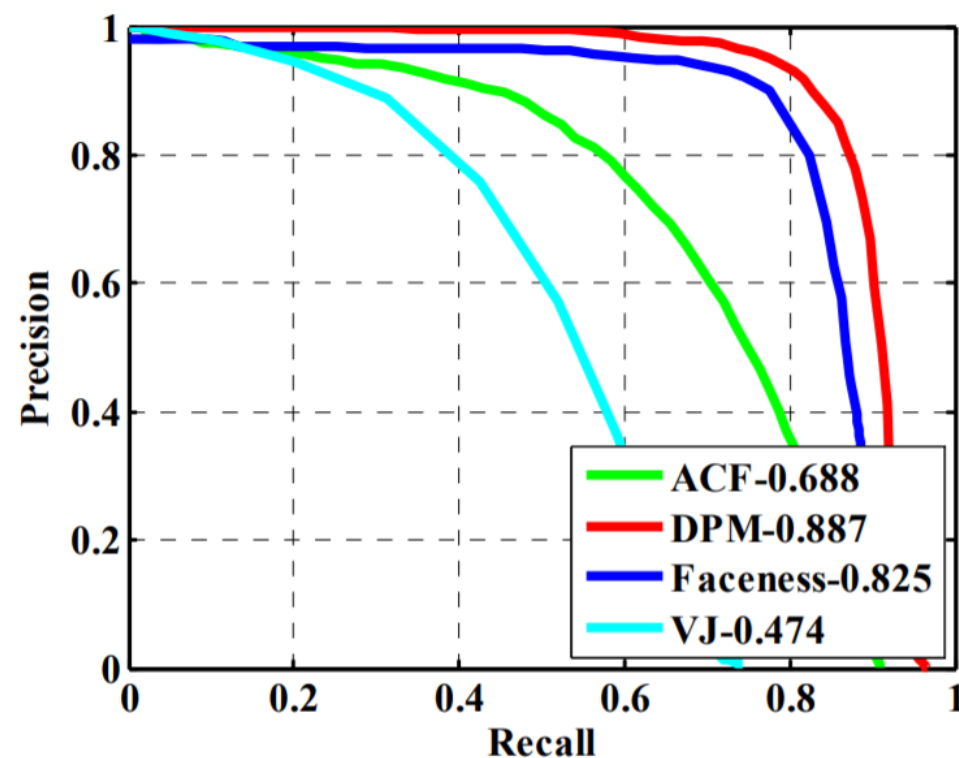


Figure 2. Examples of annotation in WIDER FACE dataset (**Best view in color**).



ACF(A Fast Corner Detector): 카테고리: 코너 탐지 및 특징 검출
DPM(Deformable Parts Model) : 객체 탐지와 위치 정확도 향상
Faceness: 얼굴 탐지 및 얼굴 관련 작업
Viola-Jones : 얼굴 인식 및 일반 객체 탐지

THANK YOU

kgpark88@gmail.com