DSM(Driver Status Monitoring) 시스템을 위한 얼굴 검출

DSM Summary

- 국내 및 세계 각 국의 교통사고 사망 원인으로 **졸음 운전 및 전방 주시 태만**이 큰 비중을 차지하는 것으로 분석.

- 이를 극복하기 위하여 자동차 제조사들은 운전 보조 시스템(Advanced Driver Assistance System, ADAS)에 운전자 상태 모니터링(Driver Status Monitoring) 시스템을 개발 및 제품으로 출시되고 있으며 ADAS 시장이 확대되는 것과 함께 고객 요구도 및 **시장이 확대 될 것으로 전망**.

- DSM 시스템은 **vision 기술을 이용**하여 얼굴 및 눈 인식, 객체 추적, 상태 판단 등의 알고리즘 처리를 통하여 구현.

- 차량 내 외부 조명 환경 및 야간에 강인하고, 신뢰할 수 있는 인식 성능을 가진 시스템 구현이 필요.

- 현재 기술 수준 및 차량 내 적용 사례는 vision 시스템을 중심으로 각종 센서를 이용하여 운전자의 상태를 모니터링 하고 이상 징후 발생 시 **알람 및 경고를 제공**해 주는 수준.

- 신뢰성 높은 제품을 개발하여 **운전자 인증**을 통한 편의 기능 제공 및 **ADAS 연계**를 통한 운전 환경 개선을 제공 함.

◆ DSM 개발 필요성

– **Causes of Car Accidents**

» 국내: 고속도로 교통사고 사망의 주요 원인 – 졸음운전

▪ 최근 5년간 고속도로 고속도로 교통사고 현황 – 1만2478건 사고 중 1473명 사망 (사망원인: **졸음운전(458명, 31%), 전방 주시 태만(425명, 28.8%)**, 과속(264명, 17.9%) 등 <2014.10.8, 아시아 경제>

» 일본: 교통사고 사망의 주요 원인 – 졸음운전 및 전방 주시 태만

▪ 2시간에 한명 꼴로 교통사고로 사망 – 2012년 년간 4500명 사망. 주요 사망원인: **졸음운전 또는 전방 주시 태만(40%)**, 속도 위반(18%) 등 <2013.12.19, Automotive Report>

» 북미 및 유럽

▪ OECD회원국 교통사고 비교 자료에 따르면 전 세계적으로 교통사고 빈도 수는 전반적으로 낮아지는 추세이며 그 중 북미가 교통사고가 가장 많이 발생하고 있다.(**졸음운전 사고 한해 약 6만 건 발생, 이 중 2만 건 사망사고**, NHTSA, 2002)

▪ 유럽의 국가들 중 독일의 경우 치명적 사고 중 25%가 운전자의 피로에 기인한 것이라는 통계자료가 있다.(독일 보험협회)

– **Trend**

» 주요 자동차 OEM은 ADAS 기능의 추가 및 개발에 투자를 하고 있으며, 졸음 감지 시스템 등의 운전자 모니터링 시스템에 기술 개발 투자 중

◆ DSM (주요 OEM 기술 현황)

– **BOSCH, DDD(Driver Drowsiness Detection) System (Vision system + sensors)**

» EPS 및 steering wheel angle sensor와 연동해 wheel의 움직임을 지속적으로 monitoring (warning signal or information 제공)

– **DENSO, Passenger Eye (Only Vision system)**

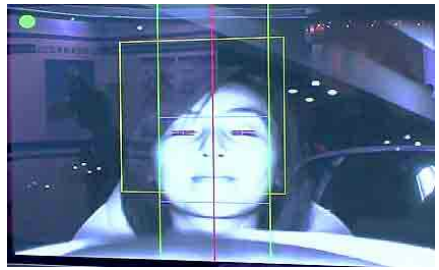
» 핸들 중앙에 설치된 카메라로 운전자 얼굴을 촬영해 얼굴의 방향이나 눈의 열린 상태를 감지하여 졸음운전이나 한눈을 팔면서 하는 운전 등이 벌어지면 운전자에게 경고

– **AISIN SEIKI (Vision system + sensors)**

» 핸들에 카메라 모듈을 설치하고 좌석에 압력 센서와 진동 장치를 내장한 뒤 운전자의 호흡이나 맥박 등을 감지. 카메라와 압력 센서를 통해 종합적인 판단으로 졸음 혹은 실신상태를 구별. 가벼운 졸음의 경우 음성 안내만으로 끝나지만, 깊은 졸음운전 및 실신 등 비상사태가 검출되면 음성 안내뿐만 아니라 좌석을 진동시켜 사고를 일으키기 전에 운전자를 깨우는 기능

<BOSCH>

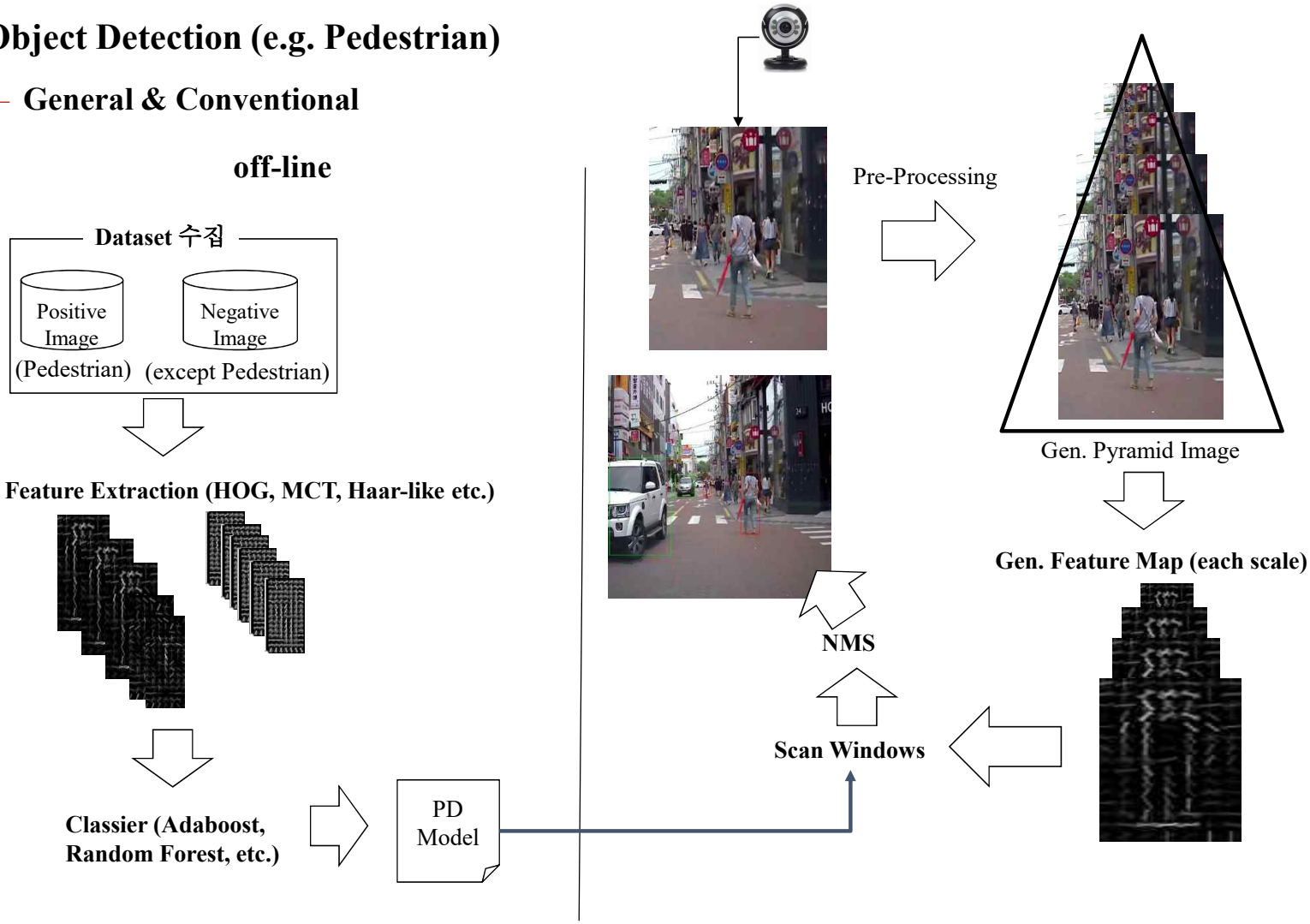
<DENSO>



<AISIN SEIKI>

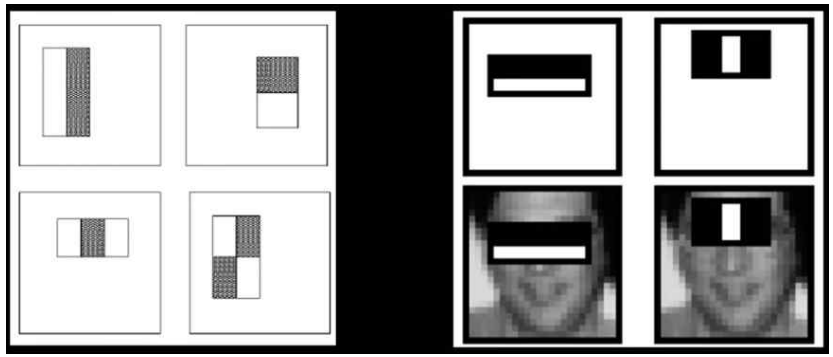
Conventional Method of the Object Detection

- **Object Detection (e.g. Pedestrian)**
  - **General & Conventional**



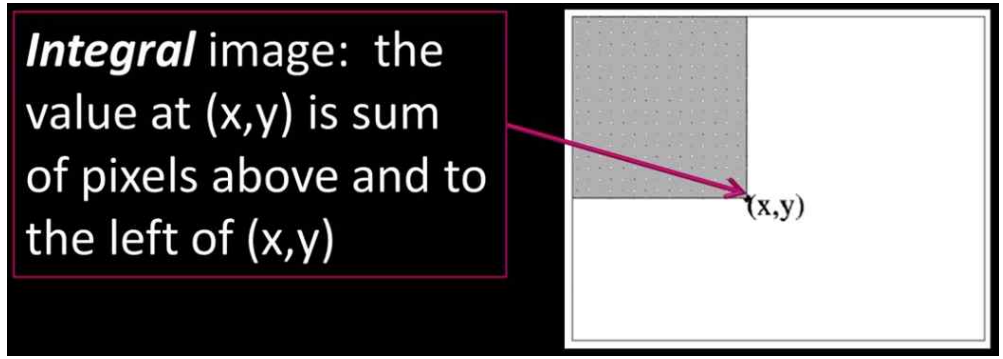
Base Knowledge: Face Detection

- **Face Detection (P. Viola & M. Jones)**
  - **Rapid Object Detection using Boosted Cascade of Simple Features**
    - » Objects: faces
  - **Main Idea**
    - » Features: "Rectangular" Filter (Haar-Like Feature)



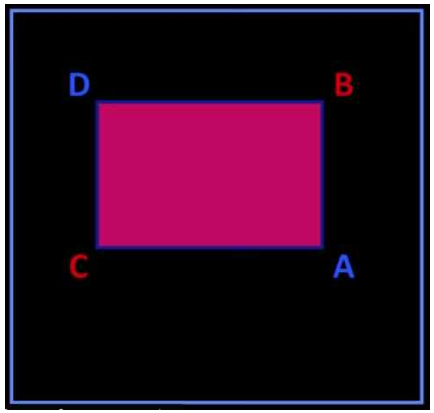
Difference in average intensity of adjacent regions

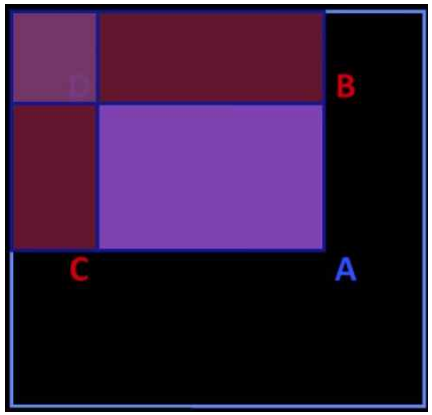
» Integral Image



Base Knowledge: Face Detection

- **Face Detection (P. Viola & M. Jones)**
  - Rapid Object Detection using Boosted Cascade of Simple Features
  - **Main Idea**
    - » Features: "Rectangular" Filter (Haar-Like Feature)
    - » Integral Image: Why this is very useful?
      - Only **2 minus(-) and 1 plus(+) operations are required for any size of rectangle**



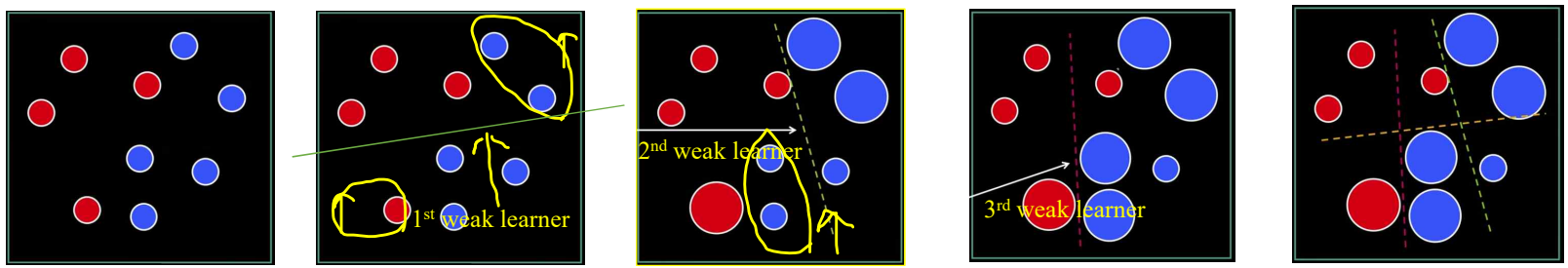


The Sum of original image values within the rectangle can be computed to integral image as:

$$sum = A - B - C + D$$

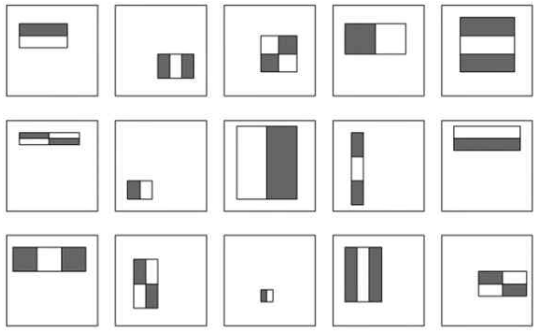
Base Knowledge: Face Detection

- **Face Detection (P. Viola & M. Jones)**
  - Rapid Object Detection using Boosted Cascade of Simple Features
  - **Main Idea**
    - » Features: "Rectangular" Filter (Haar-Like Feature)
    - » Integral Image
    - » Cascade Adaboost
      - ▪ Boosting: Iterative Learning Method
        - ✓every iteration: calculates the weighted training error
        - ✓Initially: weight each training example equally
        - ✓In each boosting rounds
          - • **Find the weak Learner** that achieves the **lowest weighted training error**
          - • Raise weights of training examples misclassified by current weak learner
            - • weak learner: simply a function that partitions space
        - ✓**Combines the weak learner:** Compute final Classifier as linear combination of all weak learners



Base Knowledge: Face Detection

- **Face Detection (P. Viola & M. Jones)**
  - Rapid Object Detection using Boosted Cascade of Simple Features
  - **Main Idea**
    - » Features: "Rectangular" Filter (Haar-Like Feature)
    - » Integral Image
    - » Cascade Adaboost
      - Weak Learner(Classifier)



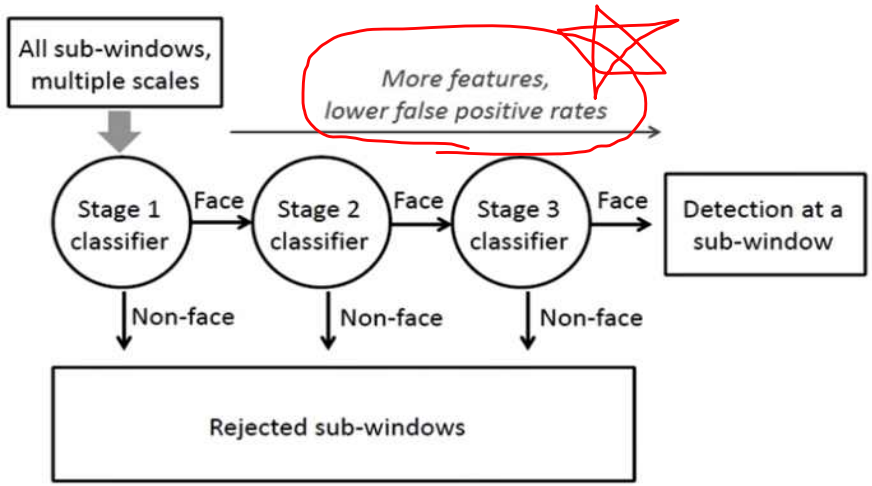
Considering all possible filter parameters – position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Choose discriminative features to be weak classifiers

Base Knowledge: Face Detection

- **Face Detection (P. Viola & M. Jones)**
  - Rapid Object Detection using Boosted Cascade of Simple Features
  - **Main Idea**
    - » Features: "Rectangular" Filter (Haar-Like Feature)
    - » Integral Image
    - » Cascade Adaboost
      - **Rejecting clear negatives quickly**
        - ✓**Key Idea: almost every where is a non-face**
          - **Detect non-faces more frequently than faces**
          - **If we can say it's not a face, be sure and move on at the window scanning**



Face Detection using opencv lib.

- **opencv detectMultiscale Lib.**

◆ detectMultiScale() [1/3]

```
void cv::CascadeClassifier::detectMultiScale ( InputArray          image,
                                               std::vector< Rect > & objects,
                                               double              scaleFactor = 1.1 ,
                                               int                 minNeighbors = 3 ,
                                               int                 flags = 0 ,
                                               Size                minSize = Size() ,
                                               Size                maxSize = Size()
                                             )
```

Python:

```
cv.CascadeClassifier.detectMultiScale(  image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]] ) -> objects
cv.CascadeClassifier.detectMultiScale2( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]] ) -> objects, numDetections
cv.CascadeClassifier.detectMultiScale3( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]] ) -> objects, rejectLevels, levelWeights
```

Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles.

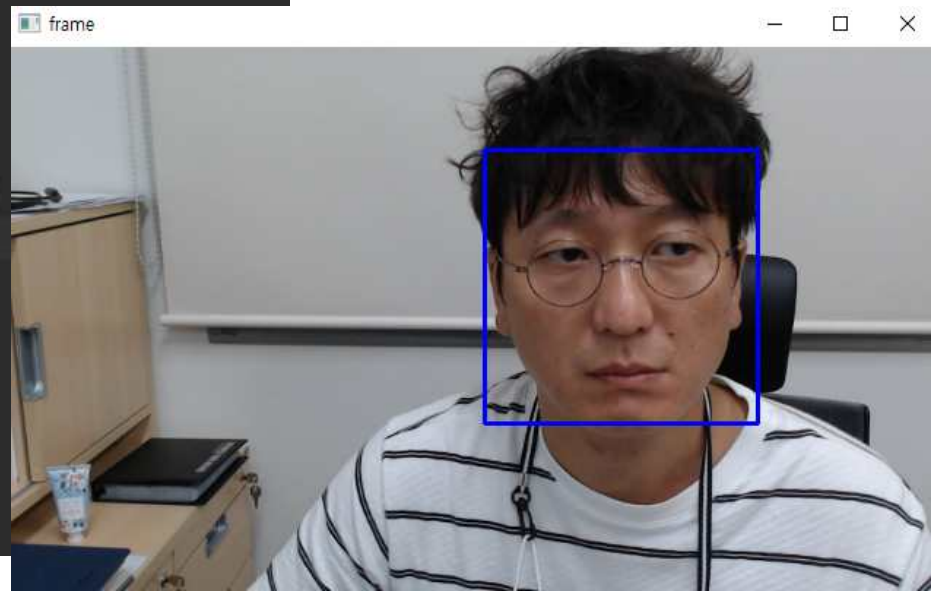
Parameters

image	Matrix of the type CV_8U containing an image where objects are detected.
objects	Vector of rectangles where each rectangle contains the detected object, the rectangles may be partially outside the original image.
scaleFactor	Parameter specifying how much the image size is reduced at each image scale.
minNeighbors	Parameter specifying how many neighbors each candidate rectangle should have to retain it.
flags	Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
minSize	Minimum possible object size. Objects smaller than that are ignored.
maxSize	Maximum possible object size. Objects larger than that are ignored. If `maxSize == minSize` model is evaluated on single scale.

Face Detection using opencv lib.

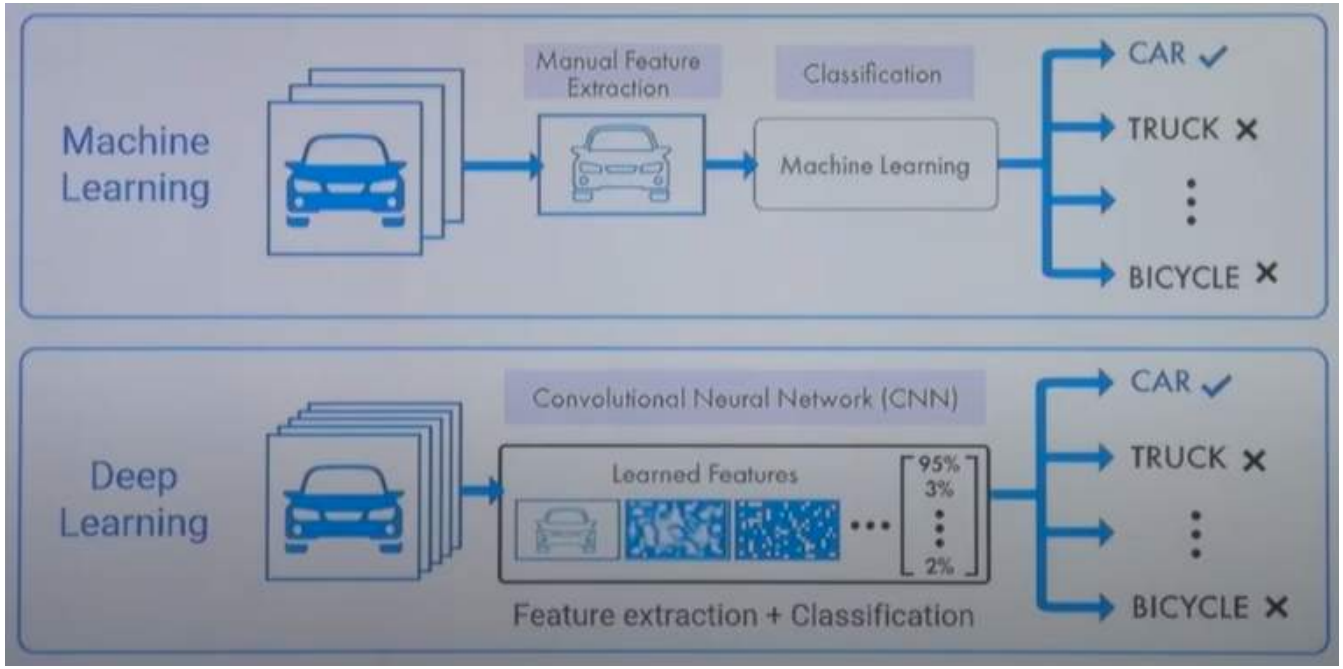
- **opencv detectMultiscale Lib.**

```python
import numpy as np
import cv2

detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

while (True):
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.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)

    cv2.imshow('frame', img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



Conventional Object Detection vs. Deep Learning base-Object Detection

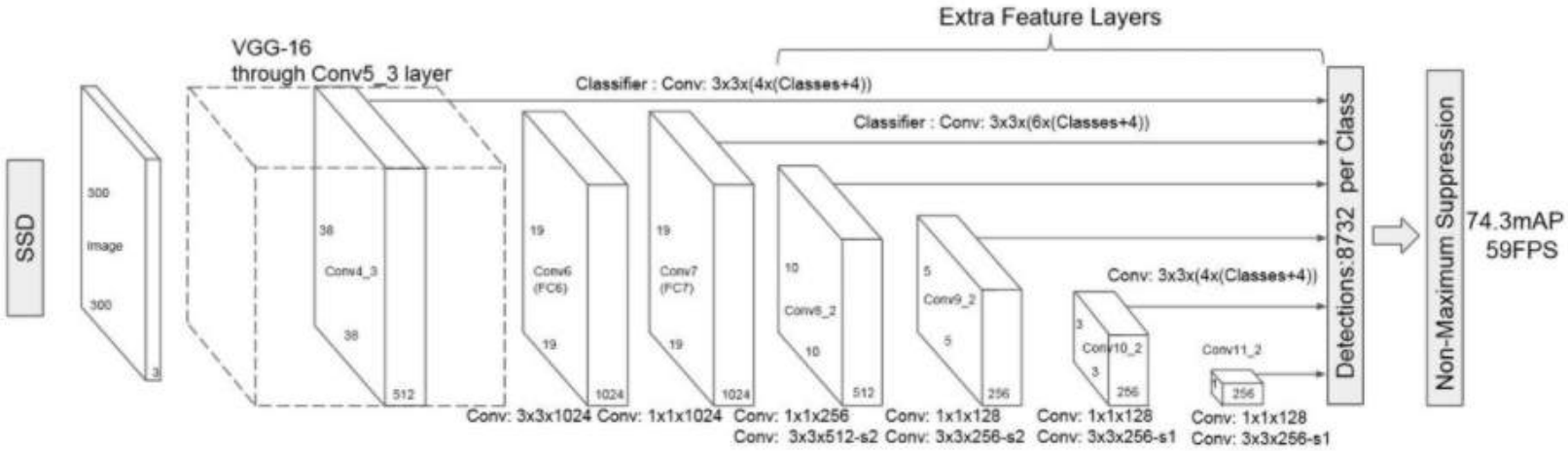
- **Object Detection (e.g. Car)**
  - **General & Conventional**

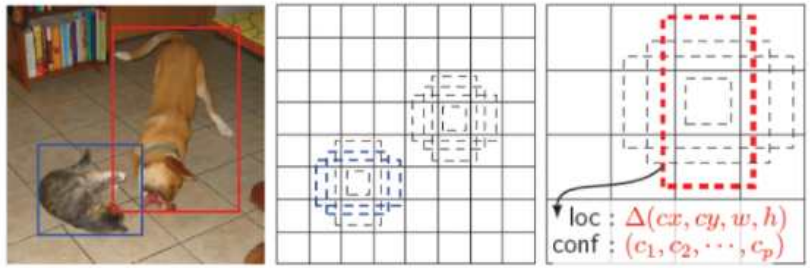


출처: https://www.mathworks.com/discovery/deep-learning.html

SSD(Single-Shot multi-box Detector)

- **SSD**
  - **Architecture**





(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Opencv DNN API

- 네트워크 불러오기 (모델 Read)
  - 학습은 안되지만, Inference를 시킬 수 있음

Python:

cv.dnn.readNet(model[, config[, framework]]) -> retval

cv.dnn.readNet(framework, bufferModel[, bufferConfig]) -> retval

- model: 훈련된 가중치를 저장하고 있는 이진 파일 이름
- config: 네트워크 구성을 저장하고 있는 텍스트 파일 이름
- framework: 명시적인 딥러닝 프레임워크 이름
- retval: cv2.dnn_Net 클래스 객체

딥러닝 프레임워크	model 파일 확장자	config 파일 확장자	framework 문자열
카페	*.caffemodel	*.prototxt	"caffe"
텐서플로우	*.pb	*.pbtxt	"tensorflow"
토치	*.t7 또는 *.net		"torch"
다크넷	*.weights	*.cfg	"darknet"
DLDT	*.bin	*.xml	"dldt"
ONNX	*.onnx		"onnx"

Opencv DNN API

- 네트워크 입력 Blob 만들기 (입력 텐서 만들기)

Python:

cv.dnn.blobFromImage(image[, scalefactor[, size[, mean[, swapRB[, crop[, ddepth]]]]]]) -> retval

- image: 입력 영상
- scalefactor: 입력 영상 픽셀 값에 곱할 값. 기본값은 1.
- size: 출력 영상의 크기. 기본값은 (0, 0).
- mean: 입력 영상 각 채널에서 뺄 평균 값. 기본값은 (0, 0, 0, 0).
- swapRB: R과 B 채널을 서로 바꿀 것인지를 결정하는 플래그. 기본값은 False
- crop: 크롭(crop) 수행 여부. 기본값은 False.
- ddepth: 출력 블롭의 깊이. CV_32F 또는 CV_8U. 기본값은 CV_32F.
- retval: 영상으로부터 구한 블롭 객체. numpy.ndarray. shape=(N,C,H,W), dtype=float32

Opencv DNN API

- 네트워크에 텐서 입력 시키기

Python:

```
cv.dnn_Net.setInput( blob[, name[, scalefactor[, mean]]] ) -> None
```

- blob: 블롭 객체
- name: 입력 레이어 이름
- scalefactor: 추가적으로 픽셀 값에 곱할 값
- mean: 추가적으로 픽셀 값에서 뺄 평균 값

SSD: mean(104, 177, 123)

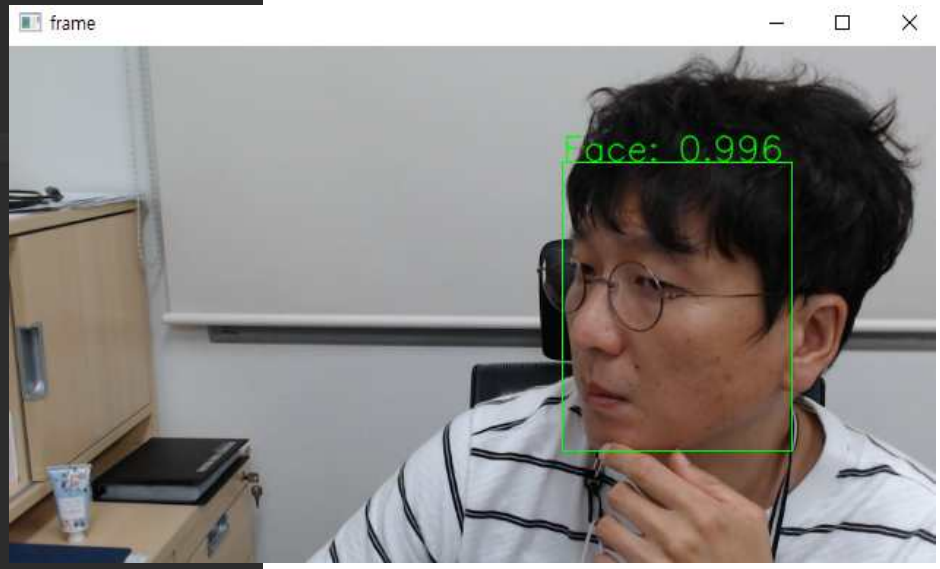
- Forward

Python:

```
cv.dnn_Net.forward(                [, outputName]                ) -> retval
cv.dnn_Net.forward(                [, outputBlobs[, outputName]] ) -> outputBlobs
cv.dnn_Net.forward(                outBlobNames[, outputBlobs]   ) -> outputBlobs
cv.dnn_Net.forwardAndRetrieve( outBlobNames                      ) -> outputBlobs
```

Face Detection using opencv dnn lib.

```
while True:
    _, frame = cap.read()
    if frame is None:
        break

    blob = cv2.dnn.blobFromImage(frame, 1, (300, 300), (104, 177, 123))
    net.setInput(blob)
    detect = net.forward()
    print(frame.shape, frame.shape[:2])
    detect = detect[0, 0, :, :]
    (h, w) = frame.shape[:2]
    print(detect[0:10, 2])
    for i in range(detect.shape[0]):
        confidence = detect[i, 2]
        if confidence < 0.5:
            break

        x1 = int(detect[i, 3] * w)
        y1 = int(detect[i, 4] * h)
        x2 = int(detect[i, 5] * w)
        y2 = int(detect[i, 6] * h)

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0))

        label = 'Face: %4.3f' % confidence
        cv2.putText(frame, label, (x1, y1 - 1), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1, cv2.LINE_AA)

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



Face Detection using opencv dnn lib.

```
    for i in range(detect.shape[0]):
        confidence = detect[i, 2]
        if confidence < 0.5:
            break

        x1 = int(detect[i, 3] * w + 0.5)
        y1 = int(detect[i, 4] * h + 0.5)
        x2 = int(detect[i, 5] * w + 0.5)
        y2 = int(detect[i, 6] * h + 0.5)

        fx = (x2 - x1) / cat.shape[1]
        cat2 = cv2.resize(cat, (0, 0), fx=fx, fy=fx)
        pos = (x1, y1 - (y2 - y1) // 4)

        overlay(frame, cat2, pos)

    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == 27:
        break

cv2.destroyAllWindows()
```

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr

