
파이썬

OpenCV 프로그래밍 4

OpenCV와 얼굴 검출

□ OpenCV에서 지원하는 얼굴 검출 기법

✓ Haar Cascade 방법

- 2001년 Viola & Jones에 의해 제안된 방법
- Haar-like 특징과 Adaboost 알고리즘, Cascade 구조를 사용하여 빠르고 정확한 얼굴 검출을 수행



✓ DNN(Deep Neural Net) 방법

- OpenCV 3.3.1부터 DNN 모듈을 사용한 얼굴 검출을 기본 예제로 제공 (2017년)
- ResNet-10과 SSD를 기반으로 학습된 얼굴 검출 네트워크 사용

OpenCV와 얼굴 검출

- Haar cascade 얼굴 검출 과정 시각화



<https://www.youtube.com/watch?v=hPCTwxF0qf4>

OpenCV와 얼굴 검출

□ OpenCV DNN 얼굴 검출

- ✓ 기존의 CascadeClassifier보다 대체로 더 좋은 성능을 나타냄

	Haar Cascade	DL
Size on disk	528KB	10MB(fp32), 5MB(fp16)
Efficiency @300x300	30ms	9.34ms

- ✓ 정면 얼굴, 측면 얼굴, 가려짐이 있어도 검출 가능

딥러닝

□ 딥러닝(Deep Learning)이란?

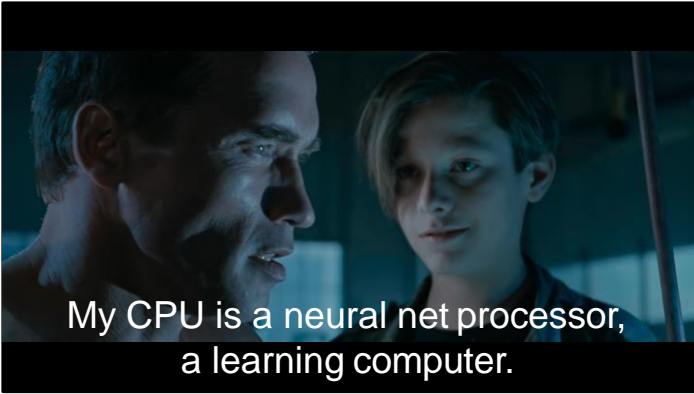
- ✓ 2000대부터 사용되고 있는 심층 신경망(deep neural network)의 또 다른 이름



WIKIPEDIA
The Free Encyclopedia

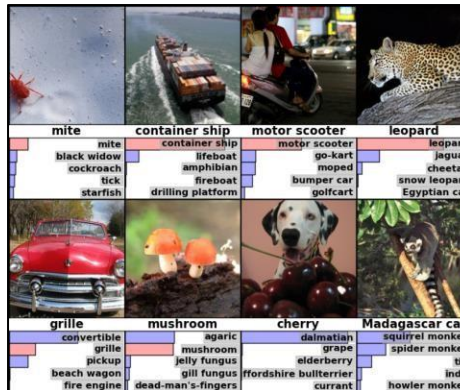
Deep learning is part of a broader family of machine learning methods based on artificial neural networks.

https://en.wikipedia.org/wiki/Deep_learning



My CPU is a neural net processor,
a learning computer.

Terminator 2 (1991)



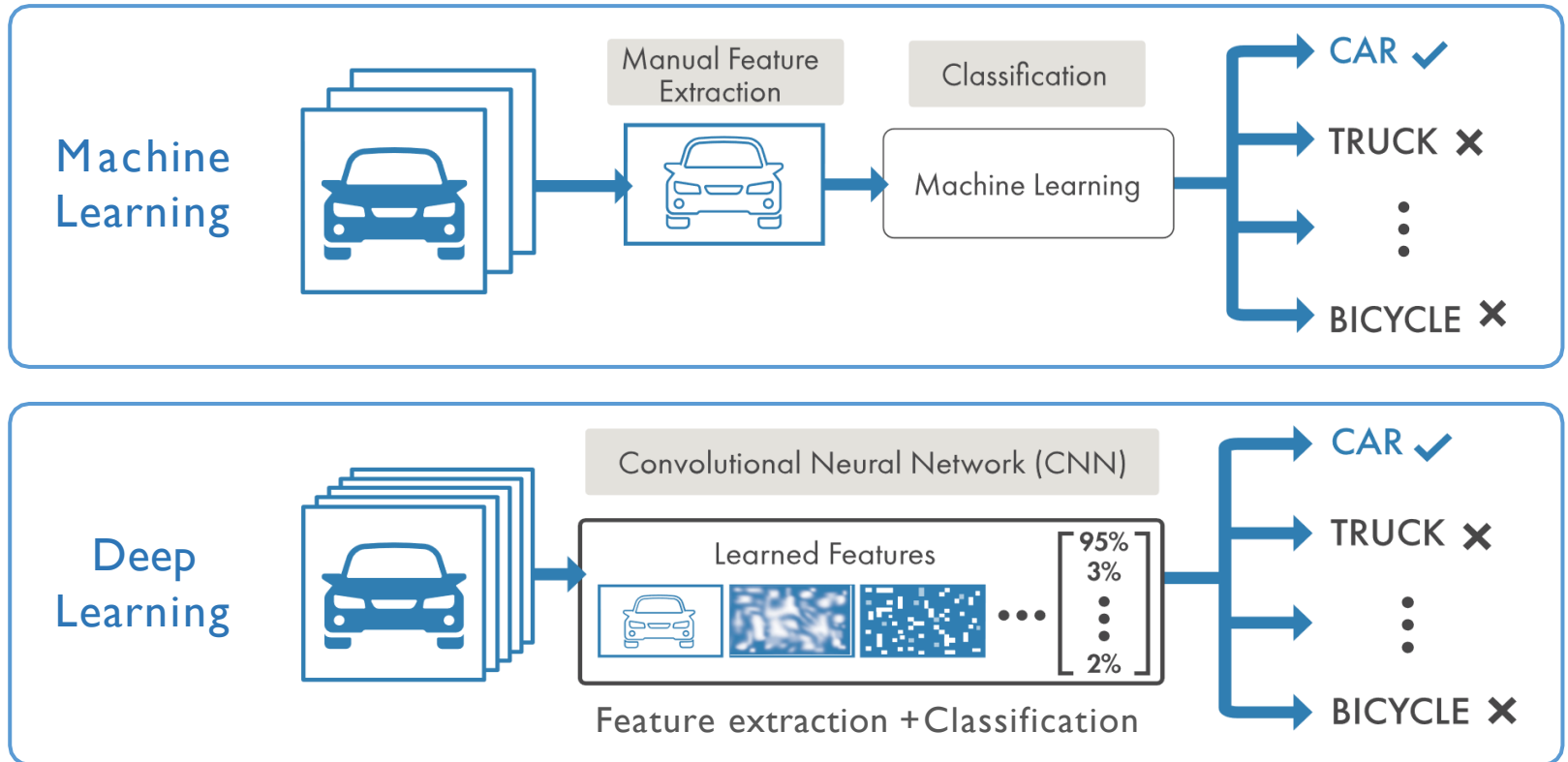
AlexNet (2012)



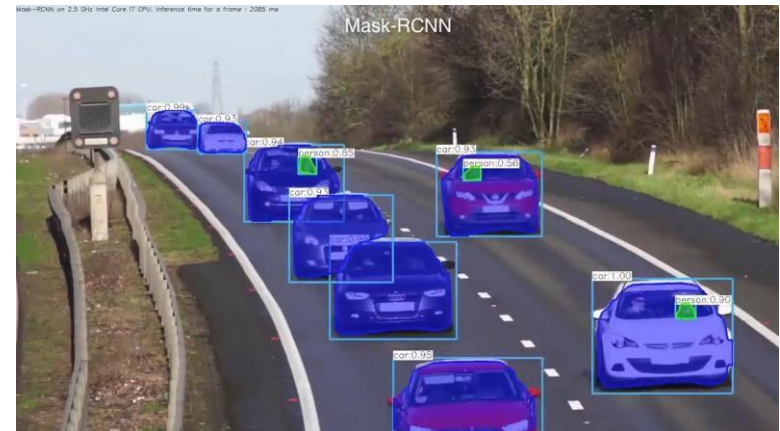
Alpha Go (2016)

딥러닝과 머신 러닝

□ 머신 러닝(ML) vs. 딥러닝(DL)



<https://www.mathworks.com/discovery/deep-learning.html> / <https://youtu.be/-SgLEuhfbg>



딥러닝 활용



Super-Resolution

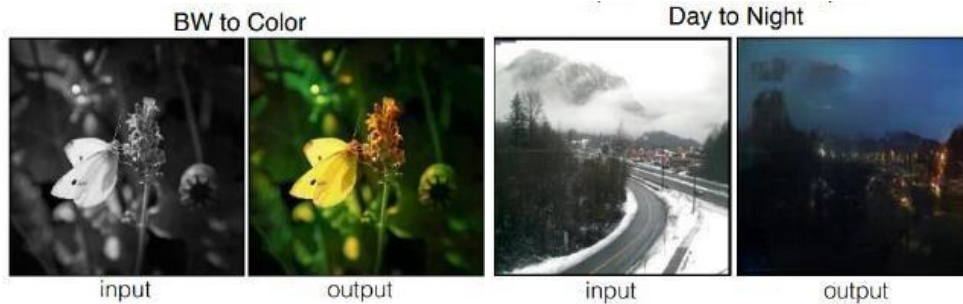


Image-to-Image Translation

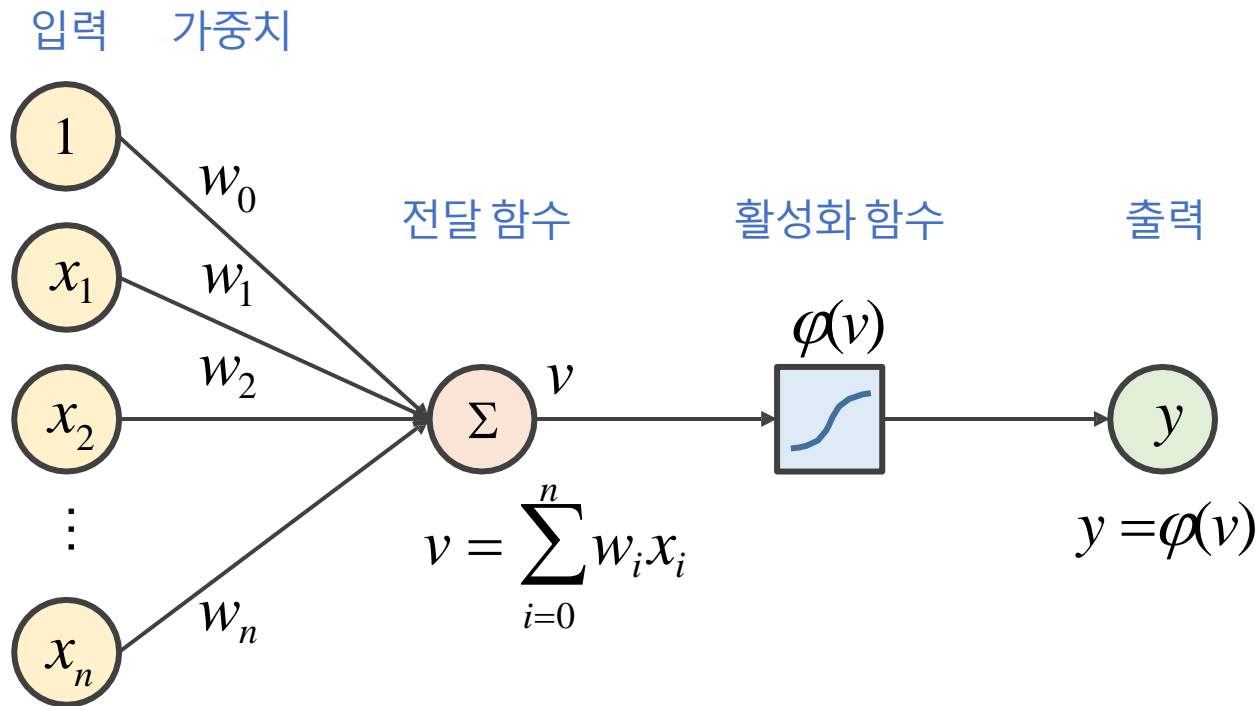


Image Inpainting

신경망 기초 이론

□ 퍼셉트론(Perceptron)

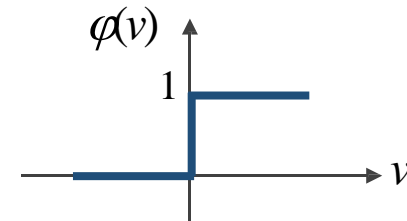
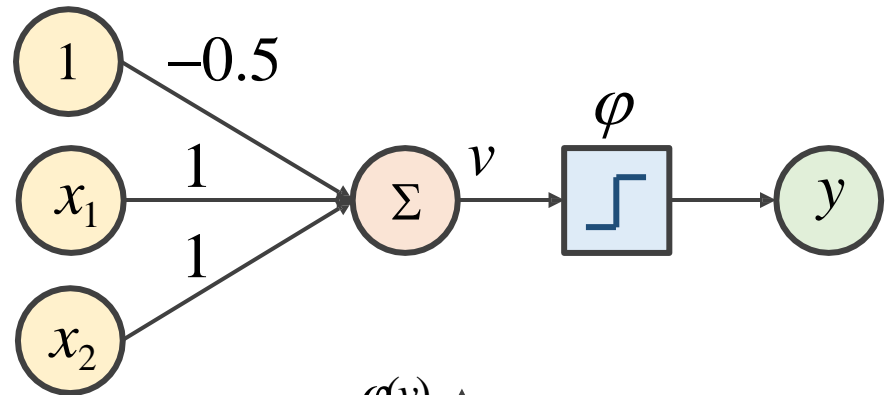
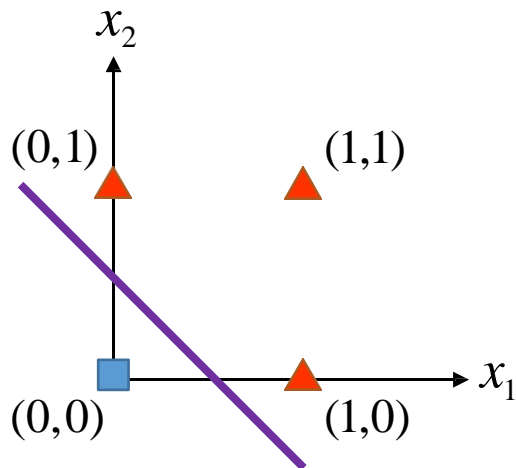
- ✓ 다수의 입력으로부터 가중합을 계산하고, 이를 이용하여 하나의 출력을 만들어내는 구조(1950년대)



신경망 기초 이론

□ 퍼셉트론에 의한 OR 연산 구현

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



$$y = \begin{cases} 1 & \text{if } x_1 + x_2 - 0.5 \geq 0 \\ 0 & \text{if } x_1 + x_2 - 0.5 < 0 \end{cases}$$

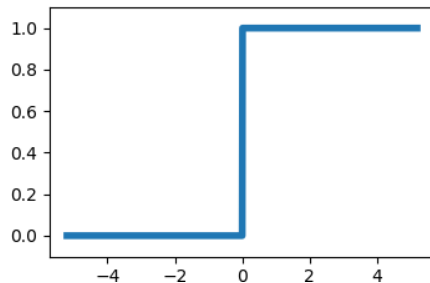
신경망 기초 이론

□ 활성화 함수(activation function)

- ✓ 생물학적 뉴런(neuron)에서 입력 신호가 일정 크기 이상일 때만 신호를 전달하는 메커니즘을 모방한 함수
- ✓ 비선형 함수를 사용

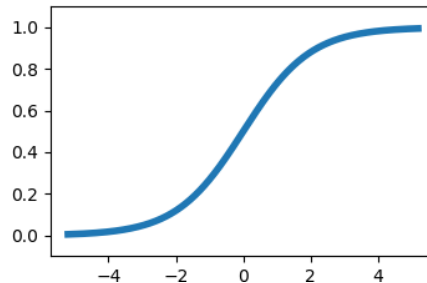
[Step function]

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



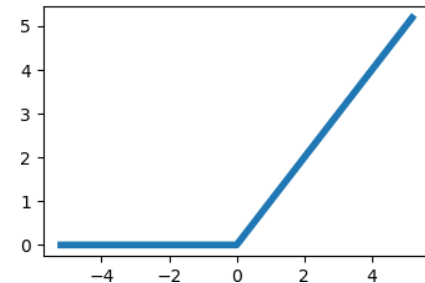
[Sigmoid]

$$f(x) = \frac{1}{1+e^{-x}}$$



[ReLU]

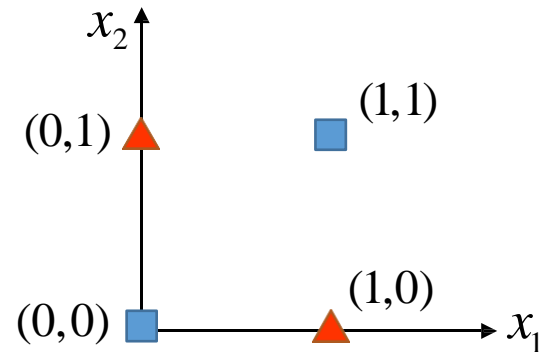
$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



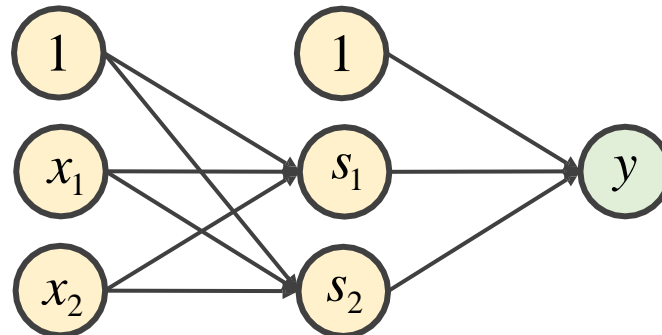
신경망 기초 이론

- 퍼셉트론에 의한 XOR 연산을 구현하려면?
 - ✓ XOR는 단순한 입력-출력 구조의 퍼셉트론으로 구현 불가 (Not linear!)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



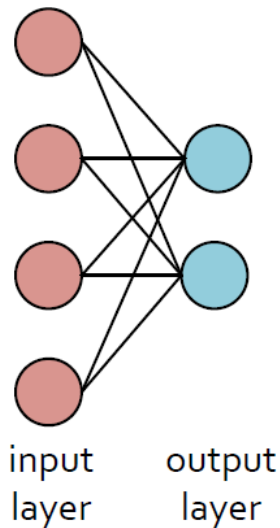
- ✓ Solution?
 - ➔ 다층 퍼셉트론 (Multi-layer perceptron)



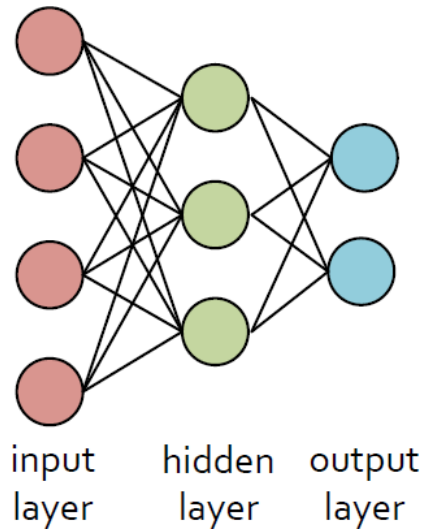
신경망 기초 이론

- 다층 퍼셉트론(MLP, Multi-Layer Perceptron)과 심층신경망(Deep Neural Network)

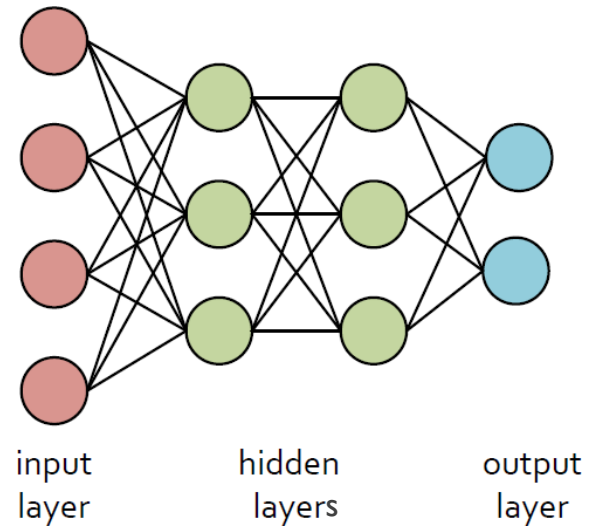
Single-Layer Perceptron



Multi-Layer Perceptron



Deep Neural Network

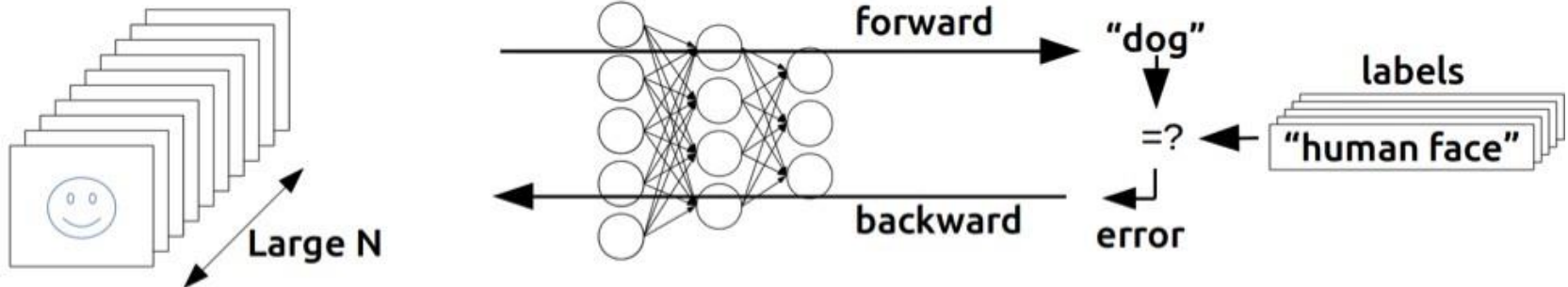


신경망 기초 이론

□ 학습 알고리즘

- ✓ Weight를 조절하는 방법
- ✓ 학습의 기준이 되는 비용(cost) 또는 손실(loss) 함수를 정의한 후, 비용을 최소화하는 방향으로 학습을 진행
- ✓ 대부분의 신경망 모델은 경사 하강법(gradient descent) 또는 오류 역전파(error backpropagation) 등의 알고리즘을 사용

Training



심층 신경망의 문제점과 해결 방안

- 학습이 제대로 안 됨
 - ✓ Vanishing Gradient
 - ✓ Overfitting
 - ✓ Local minima
 - ✓ Weight initialization?

- 학습이 너무 느림
 - ✓ 하드웨어 성능이 낮음
 - ✓ Gradient Decent

심층 신경망의 문제점과 해결 방안

□ 학습이 제대로 안 됨

- ✓ Vanishing Gradient → ReLU(Rectified Linear Units) 사용
- ✓ Overfitting → Regularization (Dropout, Batch Normalization)
- ✓ Local minima → Don't worry!
- ✓ Weight initialization? → 일정 범위 안의 임의의 값. (Xavier method)

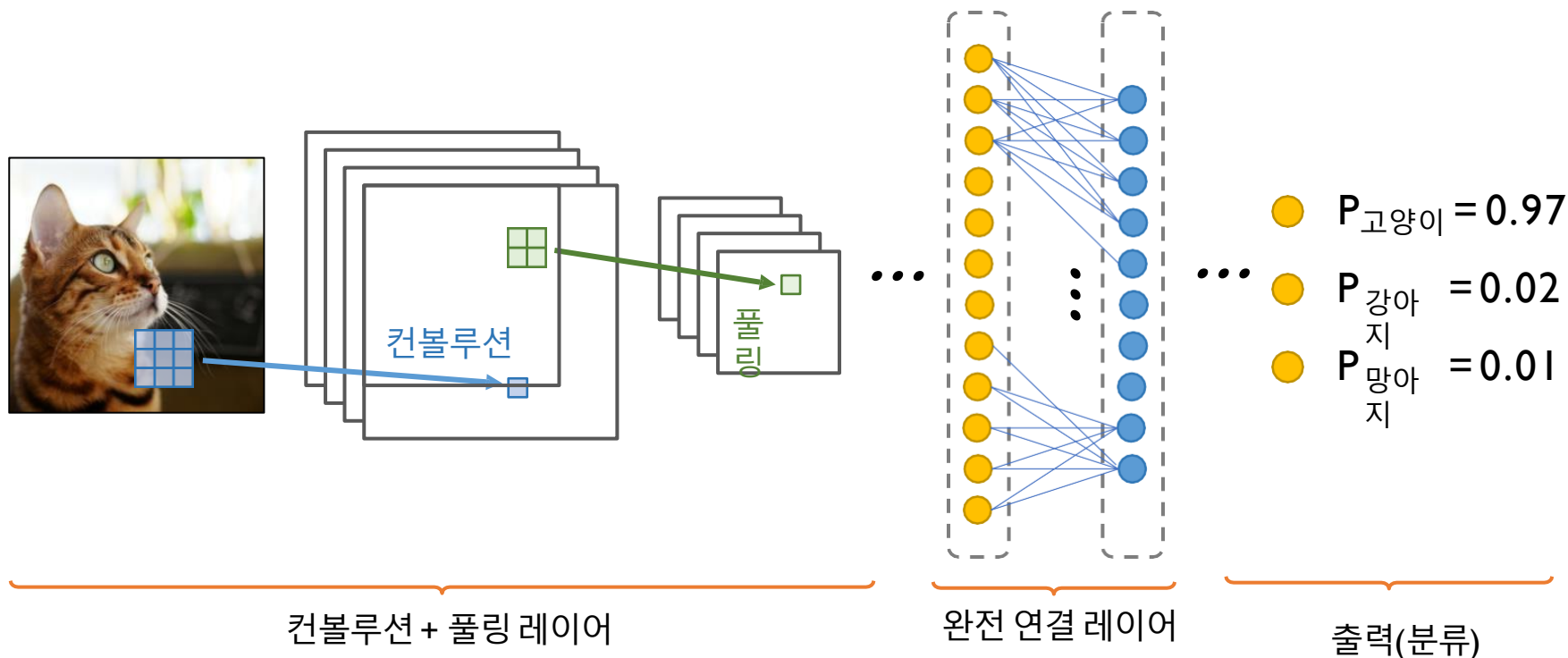
□ 학습이 너무 느림

- ✓ 하드웨어 성능이 낮음 → CPU, GPU 발전
- ✓ Gradient Decent → SGD, Adam method

→ Large dataset (e.g. ImageNet, COCO, etc)

컨볼루션 신경망: CNN

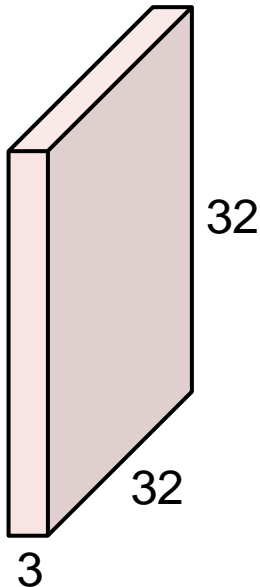
- 컨볼루션 신경망(CNN: Convolutional Neural Network)
 - ✓ 영상 인식 등을 위한 딥러닝에 특화된 네트워크 구조
 - ✓ 일반적 구성: 컨볼루션 + 풀링(pooling) + ... + 완전 연결 레이어(FCN)



컨볼루션 신경망: CNN

- 컨볼루션 레이어(Convolution Layer)
 - ✓ 2차원 영상에서 유효한 특징(feature)를 찾아내는 역할
 - ✓ 유용한 필터 마스크가 학습에 의해 결정됨
 - ✓ 보통 ReLU 활성화 함수를 함께 사용함

32x32x3 image

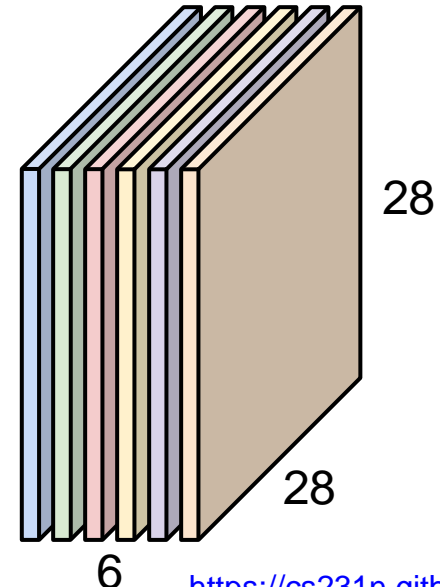


6 (5x5x3) filters



convolve (slide) over all spatial locations

28x28x6 (activation maps)

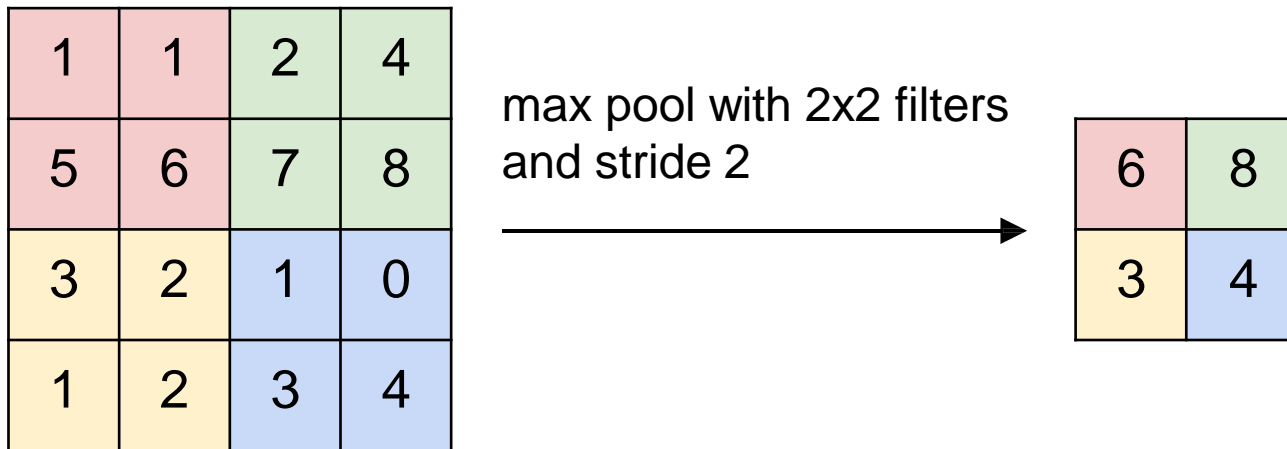


<https://cs231n.github.io/>

컨볼루션 신경망: CNN

□ 풀링 레이어(Pooling Layer)

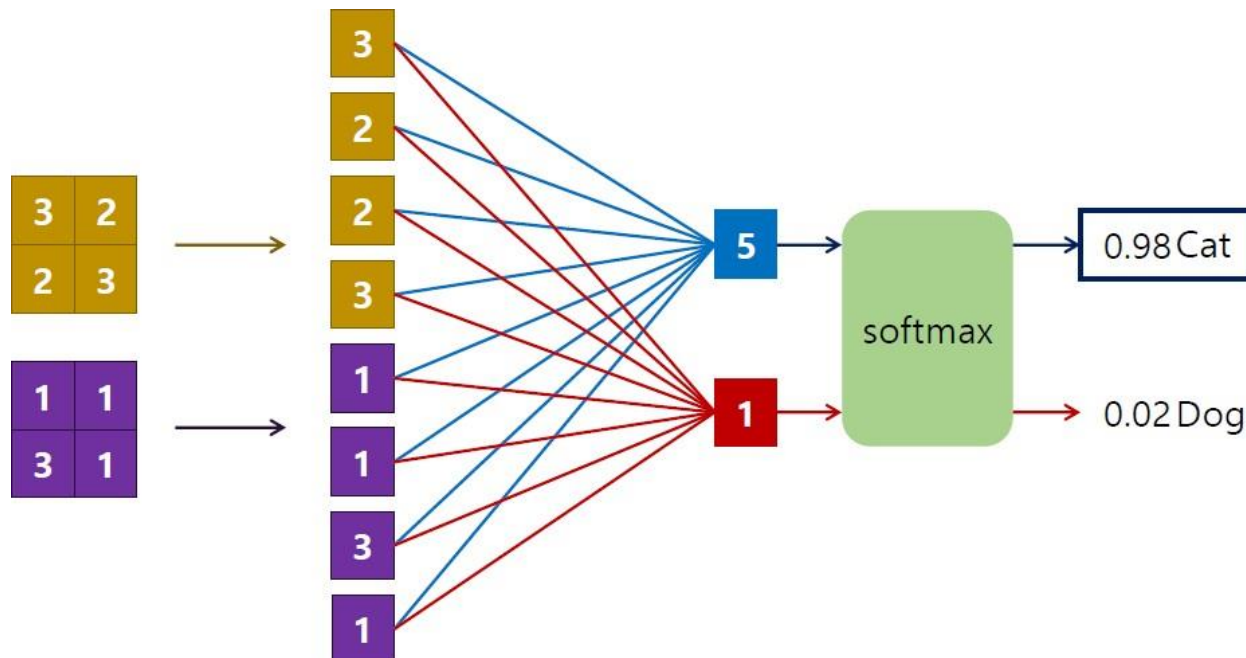
- ✓ 유용한 정보는 유지하면서 입력 크기를 줄임으로써 과적합(overfitting)을 예방하고 계산량을 감소시키는 효과를 가짐
- ✓ 보통 최대 풀링(max pooling) 또는 평균 풀링(average pooling) 사용
- ✓ 학습이 필요 없음



컨볼루션 신경망: CNN

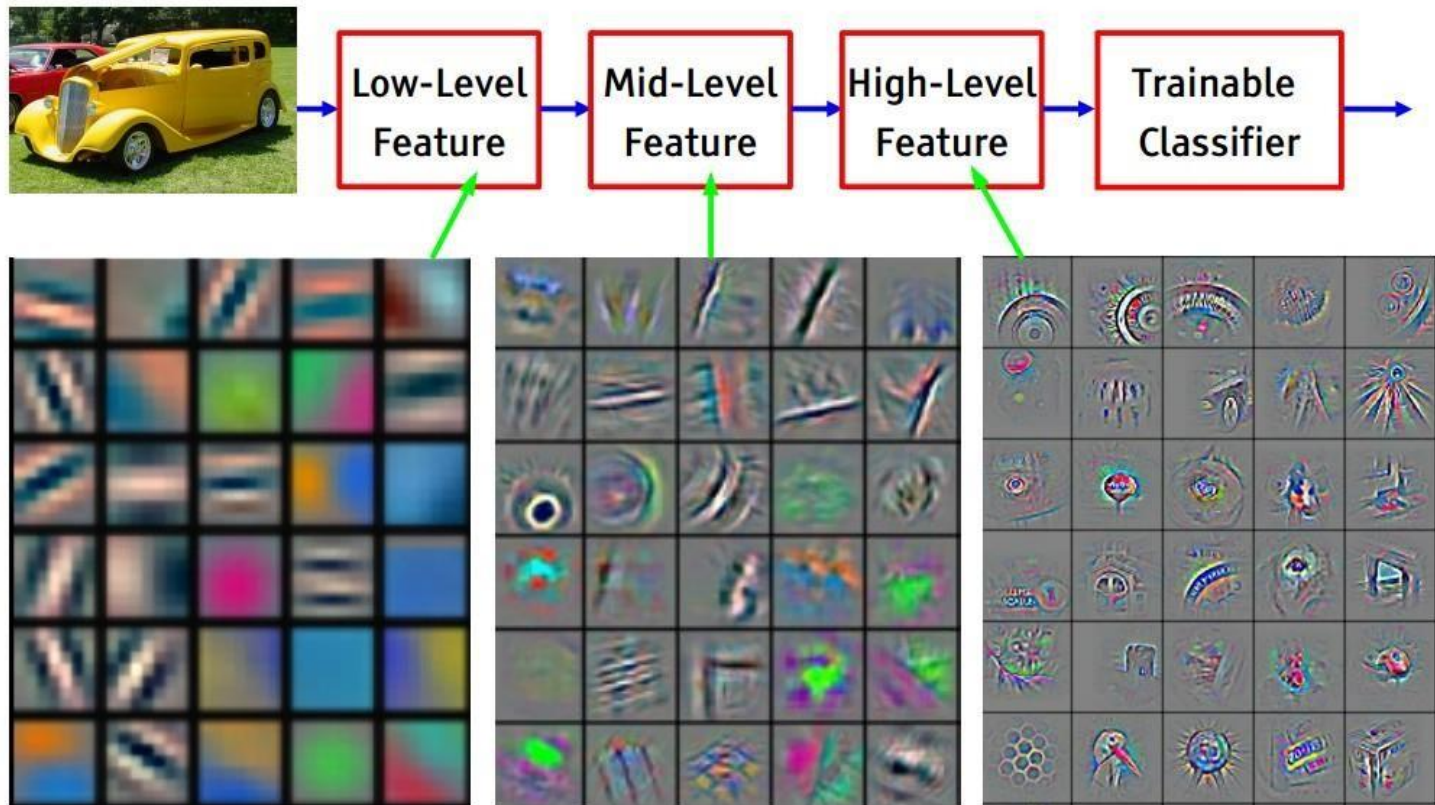
□ 완전 연결 레이어(Fully Connected Layer)

- ✓ 3차원 구조의 activation map($H \times W \times C$)의 모든 값을 일렬로 이어 붙임
- ✓ 인식의 경우, 소프트맥스(softmax) 레이어를 추가하여 각 클래스에 대한 확률 값을 결과로 얻음



컨볼루션 신경망: CNN

□ 학습된 컨볼루션 레이어 필터의 예

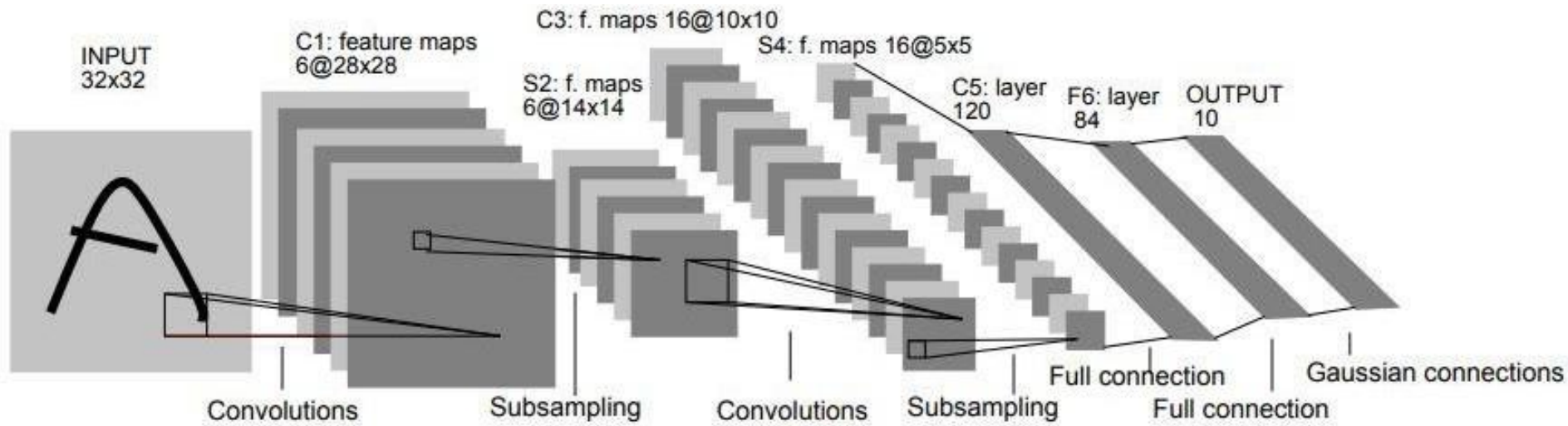


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

컨볼루션 신경망의 예

□ 필기체 숫자 인식을 위한 LeNet-5 (LeCun et al., 1998)

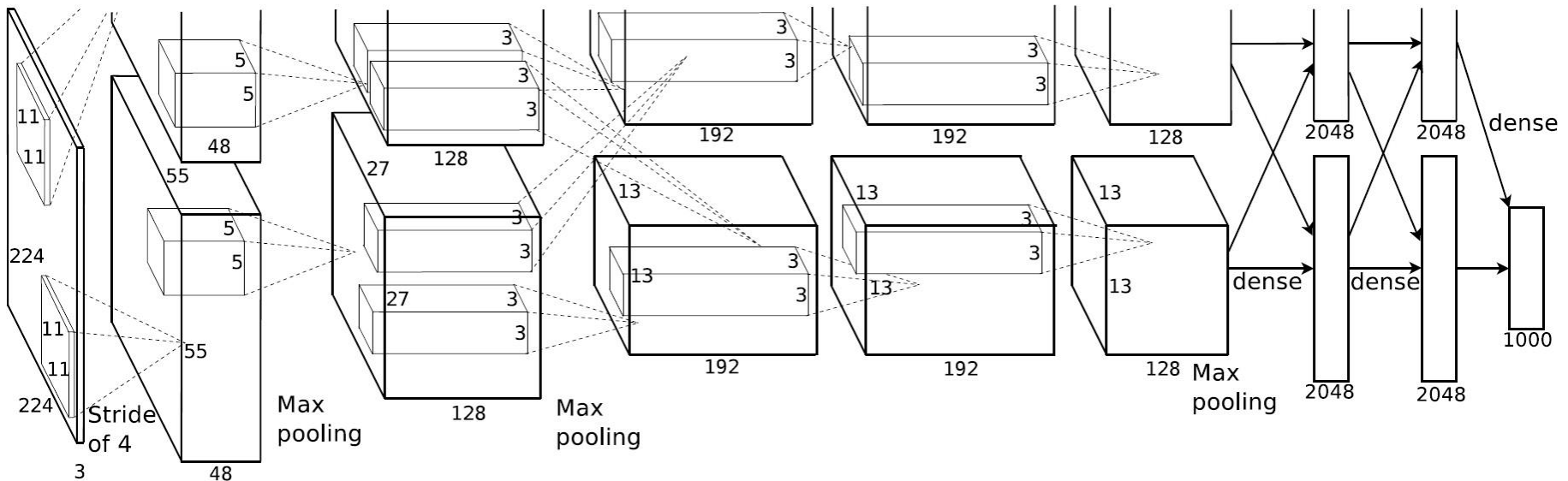
- ✓ CNN 원조
- ✓ 컨볼루션: 5×5 , Stride 1
- ✓ 풀링: 평균, 2×2 , Stride 2
- ✓ 입력: 32×32 $[0, 1]$ 정규화된 영상



컨볼루션 신경망의 예

□ AlexNet (Krizhevsky, 2012)

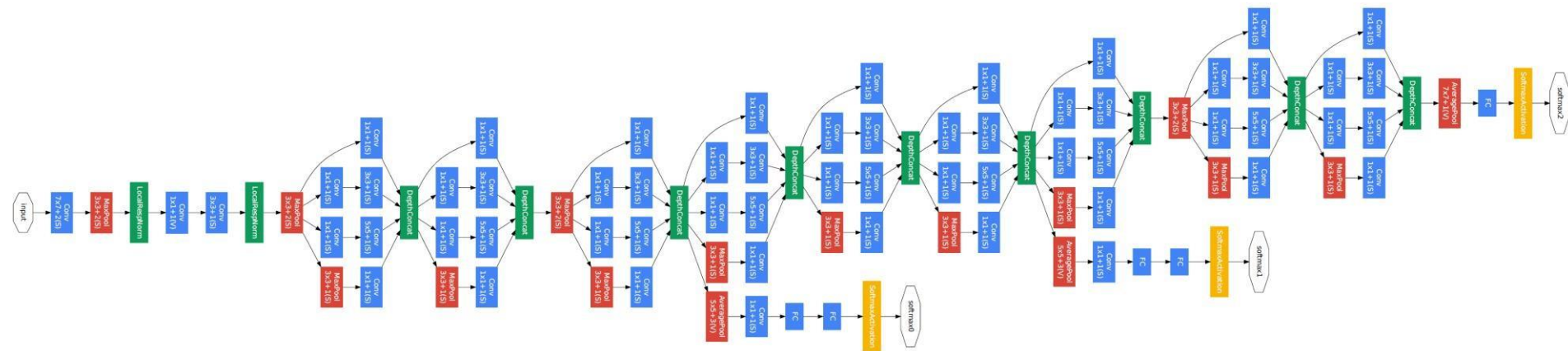
- ✓ 2012년 ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 우승
 - 1000 categories / 1.2M train, 50,000 validation, and 150,000 test images
- ✓ Top-5 error: 15.4% (Other CV-based methods > 25%)
- ✓ ReLU, 2 GPU, 227x227 input, 61M parameters, ...



컨볼루션 신경망의 예

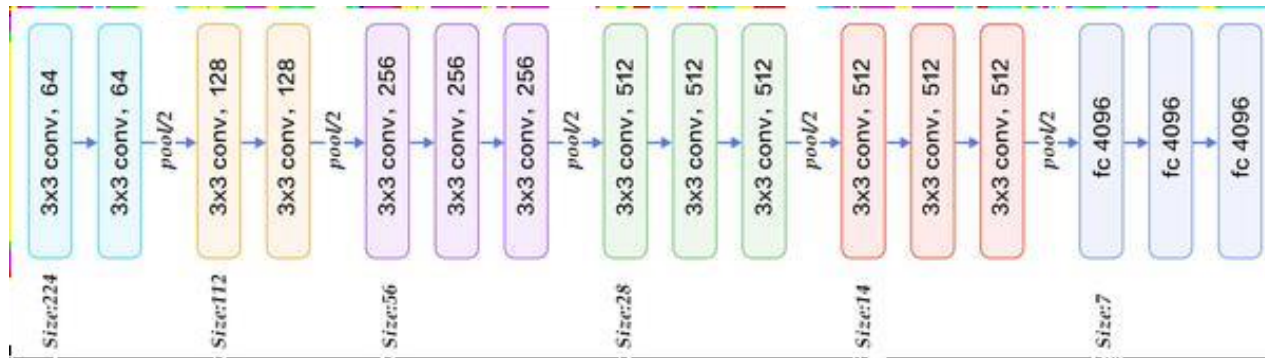
□ GoogLeNet

- ✓ 2014년 ILSVRC 영상 인식 분야에서 1위
 - 1000개 카테고리, 120만개 훈련 영상, 15만개의 테스트 영상
- ✓ Top-5 Error: 6.7% (사람: 5.1%) / 총 22개의 레이어로 구성
- ✓ 입력: 224x224, BGR, mean = (104, 117, 123)
- ✓ 출력: 1x1000, 실수형 행렬

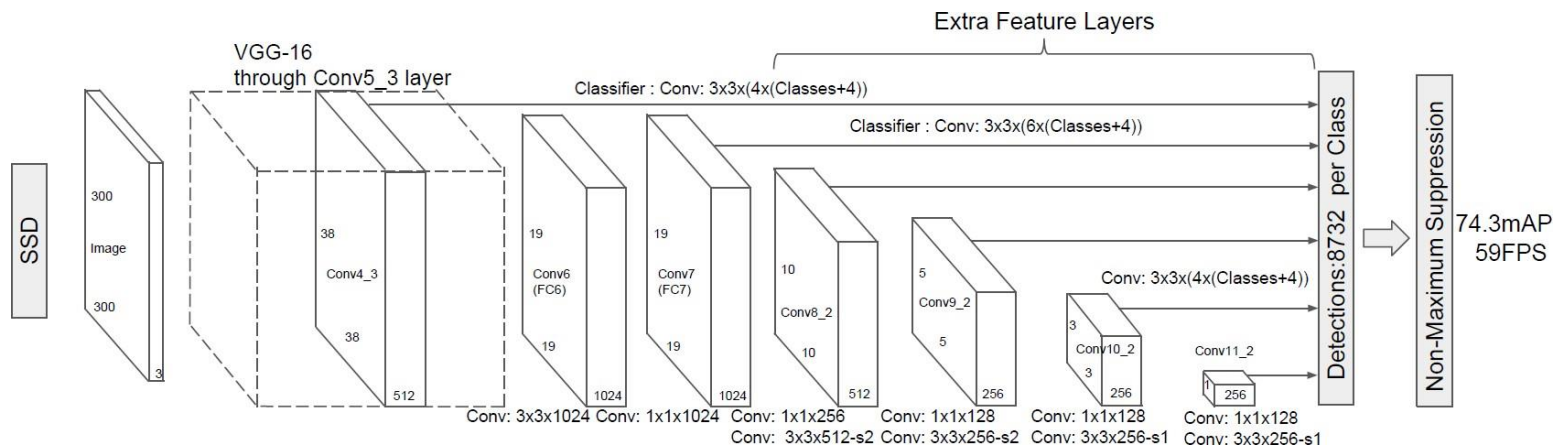


컨볼루션 신경망의 예

□ VGG16 (Simonyan and Zisserman, 2014)



□ 실시간 객체 검출을 위한 SSD (Single Shot Detector) (W. Liu, et. al, 2016)



OpenCV DNN

□ Tested networks

[Image classification]

- [AlexNet](#)
- [GoogLeNet](#)
- [VGG](#)
- [ResNet](#)
- [SqueezeNet](#)
- [DenseNet](#)
- [ShuffleNet](#)
- [Inception](#)
- [MobileNet](#)
- [Darknet](#)

[Object detection]

- [SSD VGG](#)
- [MobileNet-SSD](#)
- [Faster-RCNN](#)
- [R-FCN](#)
- [OpenCV face detector](#)
- [Mask-RCNN](#)
- [EAST](#)
- [YOLOv2, tiny YOLO, YOLOv3](#)

[Semantic segmentation]

- [FCN](#)
- [ENet](#)

[Pose estimation]

- [OpenPose](#)

[Image processing]

- [Colorization](#)
- [Fast-Neural-Style](#)

학습 기능은 없고, 테스트 기능만 있음. 학습된 모델을 가지고 와서 실행만 하는 모듈이 제공됨

[Person identification]

- [OpenFace](#)

<https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV>

OpenCV DNN

□ Net 클래스 정의

```
<opencv-src>\modules\dnn\include\opencv2\dnn\dnn.hpp
```

```
class Net
{
public:
    Net();
    ~Net();

    bool empty() const;

    Mat forward(const String& outputName = String());
    void setInput(InputArray blob, const String& name = "", double
        scalefactor = 1.0, const Scalar& mean = Scalar());

    void setPreferableBackend(int backendId);
    void setPreferableTarget(int targetId);

    int64 getPerfProfile(std::vector<double>& timings);
    ...
};
```

OpenCV DNN API

□ 네트워크 불러오기

```
cv2.dnn.readNet(model, config=None, framework=None) -> 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 블롭(blob) 만들기

```
cv2.dnn.blobFromImage(image, scalefactor=None, size=None,  
                      mean=None, swapRB=None, crop=None,  
                      ddepth=None) -> 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

□ 네트워크 입력 설정하기

```
cv2.dnn_Net.setInput(blob, name=None, scalefactor=None,  
                     mean=None) -> None
```

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

OpenCV DNN API

□ 네트워크 순방향 실행 (추론)

```
cv2.dnn_Net.forward(outputName=None) -> retval
```

- ✓ outputName: 출력 레이어 이름
- ✓ retval: 지정한 레이어의 출력 블록. 네트워크마다 다르게 결정됨.

GoogLeNet 영상 인식

- 미리 학습된 GoogLeNet 학습 모델 및 구성 파일 다운로드
 - ✓ 모델 파일:
 - http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel
 - ✓ 구성 파일:
 - https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet에서 deploy.prototxt 파일 다운로드
 - ✓ 클래스 이름 파일:
 - 1~1000번 클래스에 대한 설명을 저장한 텍스트 파일
 - https://github.com/opencv/opencv/blob/4.1.0/samples/data/dnn/classification_classes_ILSVRC2012.txt
- 입출력 형식
 - ✓ 입력: 224x224, BGR, mean=(104, 117, 123)
 - ✓ 출력: 1000개의 노드

GoogLeNet 영상 인식

```
import sys
import numpy as np
import cv2
```

```
filename = 'space_shuttle.jpg'
```

```
if len(sys.argv) > 1:
    filename = sys.argv[1]
```

명령행 인자 지원

```
img = cv2.imread(filename)
```

```
# Load network
```

```
net = cv2.dnn.readNet('bvlc_googlenet.caffemodel', 'deploy.prototxt')
```

```
# Load class names
```

```
classNames = None
```

```
with open('classification_classes_ILSVRC2012.txt', 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
)
```

GoogLeNet 영상 인식

Inference

```
inputBlob = cv2.dnn.blobFromImage(img, 1, (224, 224), (104, 117, 123))
net.setInput(inputBlob, 'data')
prob = net.forward()
```

Check results & Display

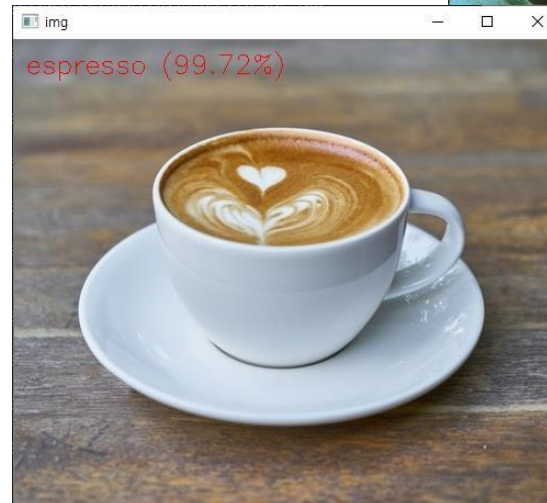
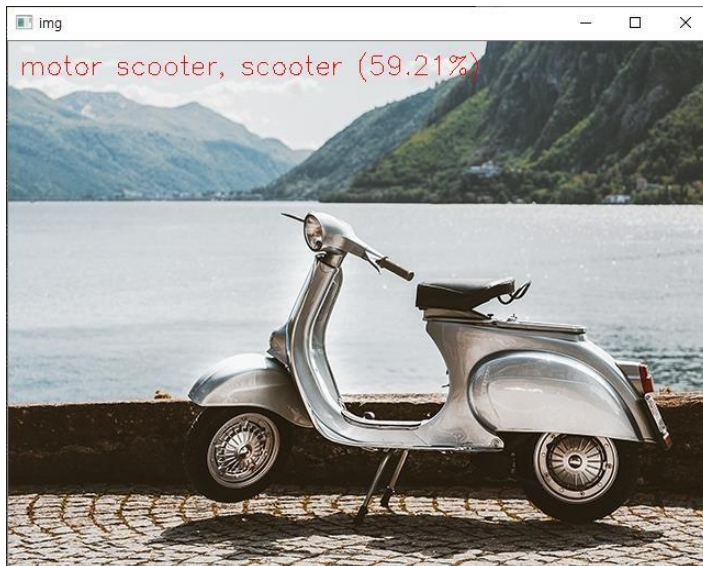
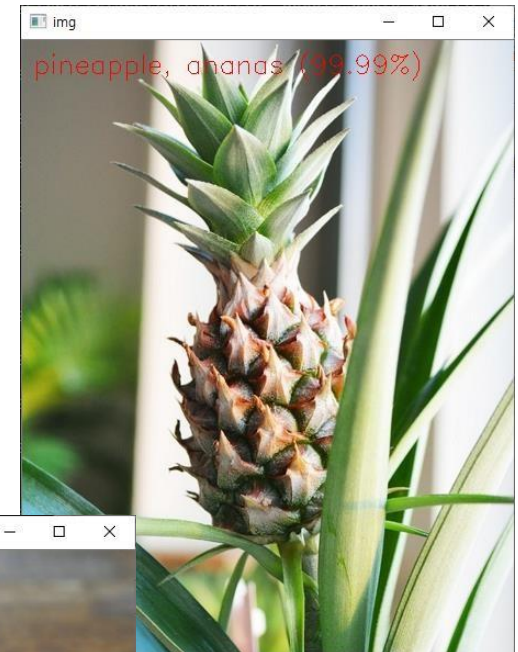
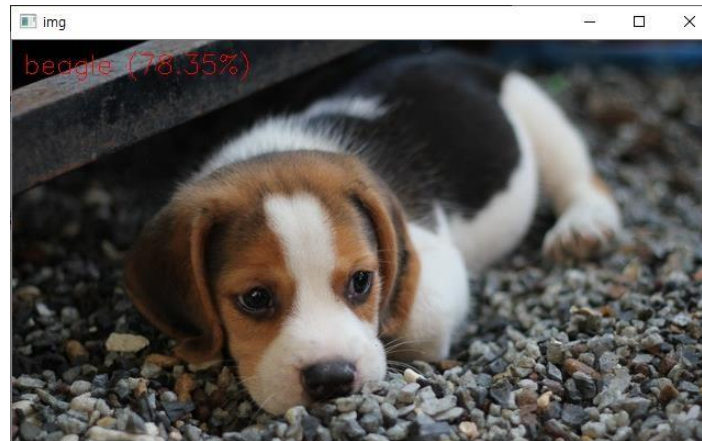
구글넷 출력 행렬 prob은 numpy.ndarray이고, shape=(1, 1000), dtype=float32.

```
out = prob.flatten()
classId = np.argmax(out)
confidence = out[classId]
```

```
text = '%s (%4.2f%%)' % (classNames[classId], confidence * 100)
cv2.putText(img, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255),
            1, cv2.LINE_AA)
```

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

GoogLeNet 영상 인식



OpenCV와 얼굴 검출

□ 모델 & 구성 파일 다운로드

✓ https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector 에서 파일 다운로드 후 아래 명령 실행

```
> python download_weights.py
```

□ 입출력 형식

- ✓ 입력: 300x300, BGR, mean=(104, 177, 123)
- ✓ 출력: class, confidence, coordinate 등의 정보를 담고 있는 4차원 행렬
shape=(1, 1, N, 7)

0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
⋮	⋮	⋮	⋮	⋮	⋮	⋮

OpenCV SSD 얼굴 검출

```
import cv2

model = 'res10_300x300_ssd_iter_140000_fp16.caffemodel'
config = 'deploy.prototxt'

cap = cv2.VideoCapture(0)

net = cv2.dnn.readNet(model, config)

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()
    det = detect[0, 0, :, :]
```

detect.shape=(1, 1, N, 7)이고 이중 뒤쪽 두 개의 차원에 검출 정보가 저장됨.
그러므로 편의상 2차원 행렬로 변환하여 사용.

OpenCV SSD 얼굴 검출

```
(h, w) = frame.shape[: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)

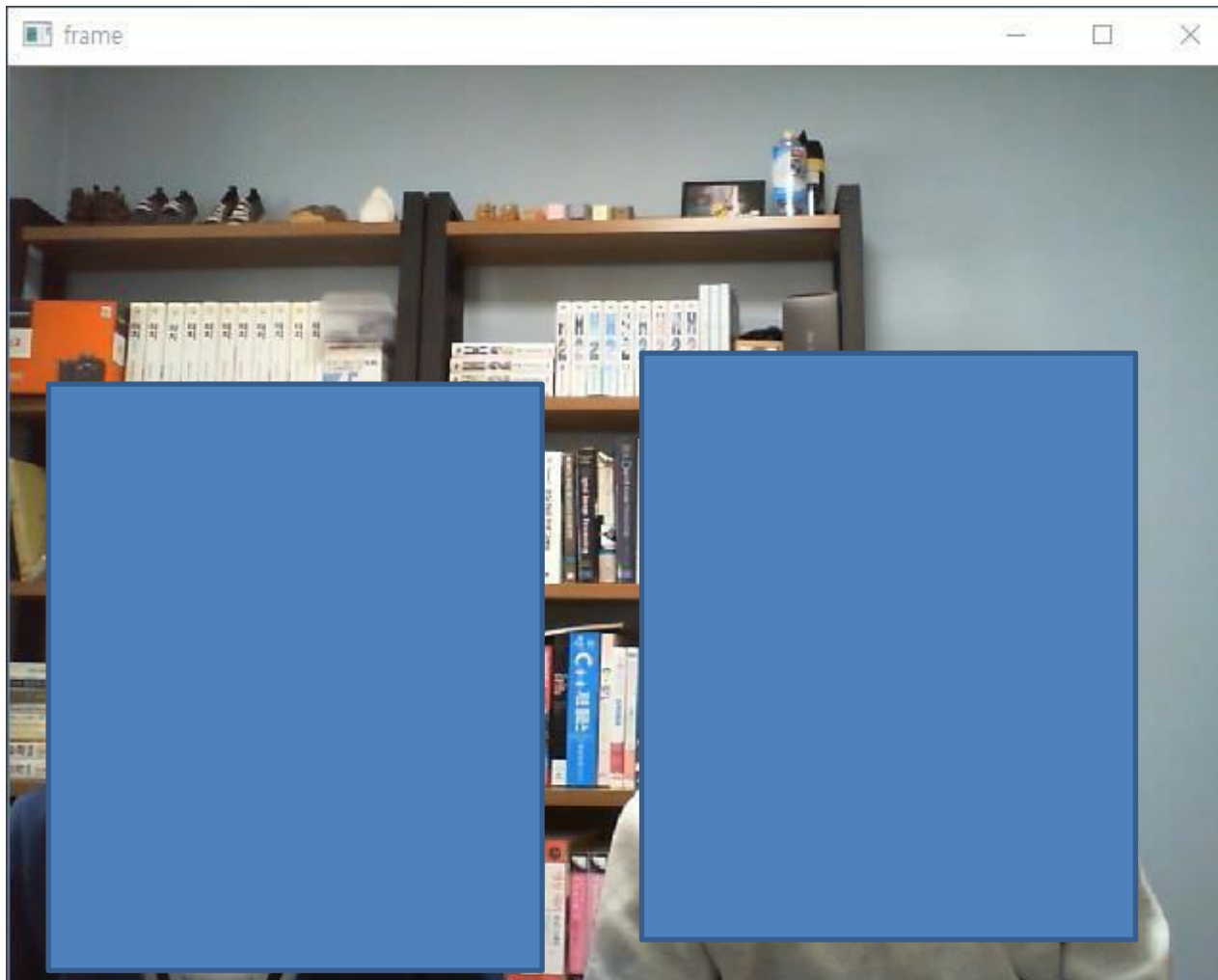
cv2.imshow('frame', frame)
if cv2.waitKey(1) == 27:
    break

cv2.destroyAllWindows()
```

detect:

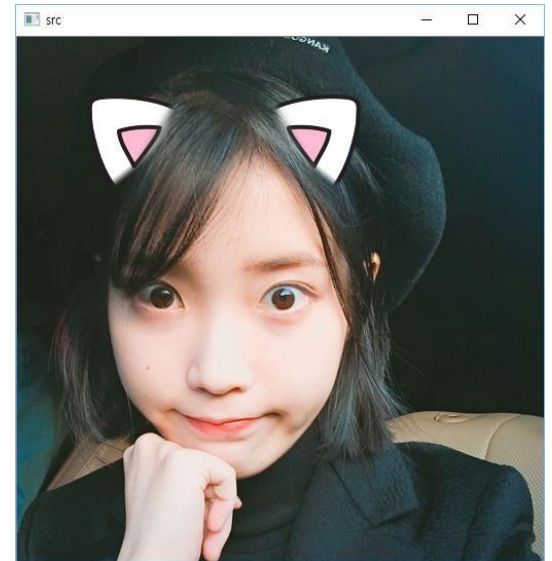
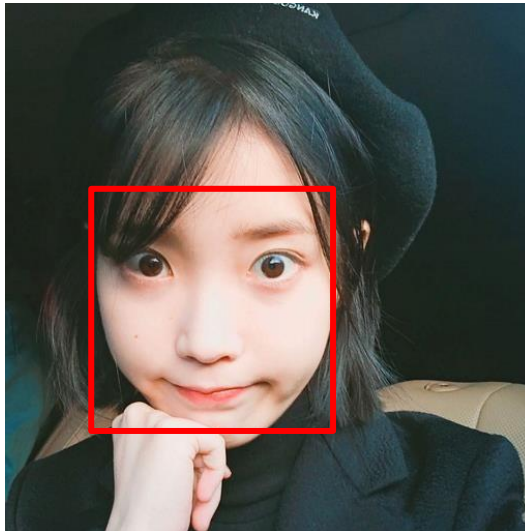
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
⋮	⋮	⋮	⋮	⋮	⋮	⋮

OpenCV SSD 얼굴 검출



얼굴 스티커

- 얼굴을 검출한 후 재미있는 그래픽을 합성
 - ✓ 투명한 PNG 파일 불러오기 ⑦ 4채널 (b, g, r, alpha)
 - ✓ 카메라 입력에서 얼굴 검출 ⑦ 얼굴 위치 & 크기 정보 추출
 - ✓ 두 장의 영상 합성



얼굴 스티커

```
import sys
import numpy as np
import cv2

model = 'opencv_face_detector_uint8.pb'
config = 'opencv_face_detector.pbtxt'

cap = cv2.VideoCapture(0)
net = cv2.dnn.readNet(model, config)
cat = cv2.imread('cat.png', cv2.IMREAD_UNCHANGED)

while True:
    ret, frame = cap.read()

    blob = cv2.dnn.blobFromImage(frame, 1, (300, 300), (104, 177, 123))
    net.setInput(blob)
    detect = net.forward()

    (h, w) = frame.shape[:2]
    detect = detect[0, 0, :, :]
```

얼굴 스티커

```
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()
```

얼굴 스티커

```
def overlay(frame, cat, pos):
    if pos[0] < 0 or pos[1] < 0:
        return

    if pos[0] + cat.shape[1] > frame.shape[1] or pos[1] + cat.shape[0] > frame.shape[0]:
        return

    sx = pos[0]
    ex = pos[0] + cat.shape[1]
    sy = pos[1]
    ey = pos[1] + cat.shape[0]

    img1 = frame[sy:ey, sx:ex] # shape=(h, w, 3)
    img2 = cat[:, :, 0:3]      # shape=(h, w, 3)
    alpha = 1. - (cat[:, :, 3] / 255.) # shape=(h, w)

    img1[:, :, 0] = (img1[:, :, 0] * alpha + img2[:, :, 0] * (1. - alpha)).astype(np.uint8)
    img1[:, :, 1] = (img1[:, :, 1] * alpha + img2[:, :, 1] * (1. - alpha)).astype(np.uint8)
    img1[:, :, 2] = (img1[:, :, 2] * alpha + img2[:, :, 2] * (1. - alpha)).astype(np.uint8)
```