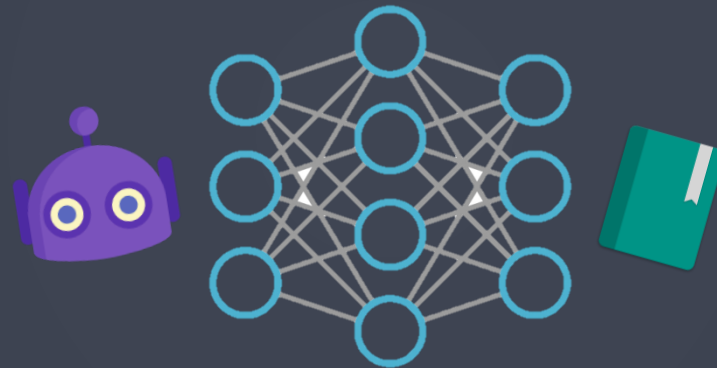


Deep Learning

Chapter 4 합성곱 신경망, 신경망 성능 개선, 전이학습 (Convolutional Neural Network, Transfer Learning)



START

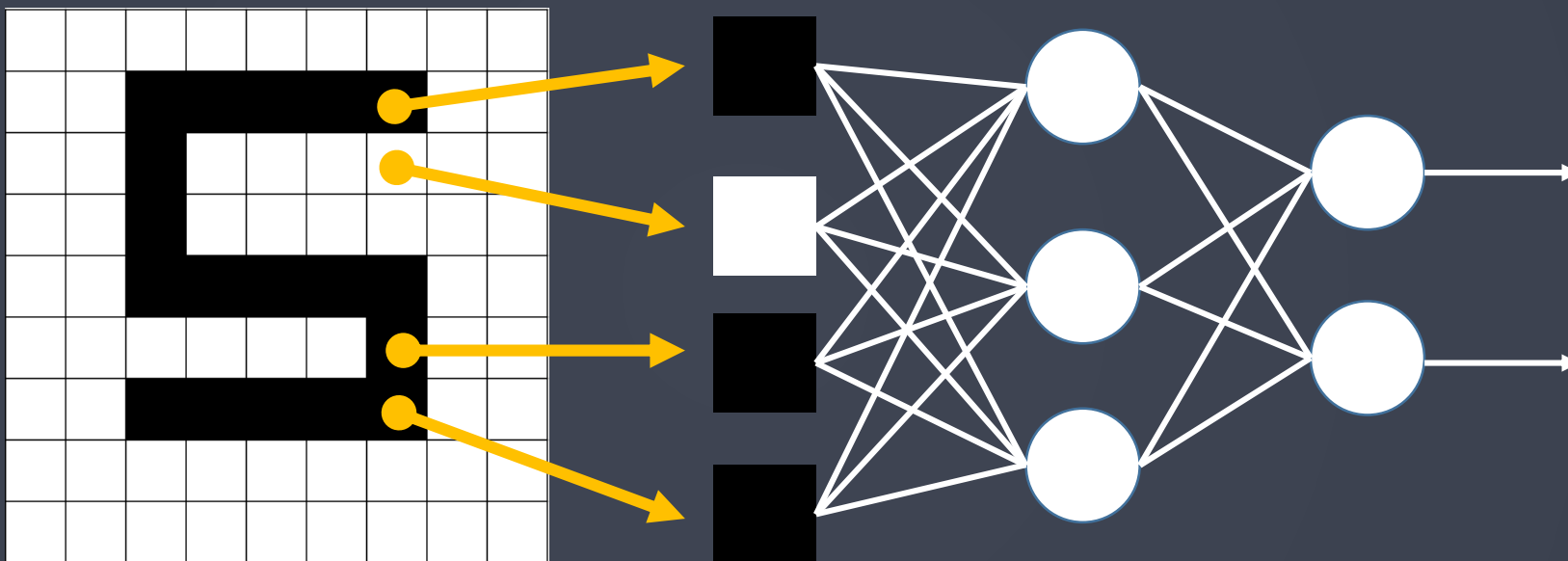
- 합성곱 신경망에 대해 알 수 있다.
- Keras를 활용해 합성곱 신경망을 구성 할 수 있다.
- 신경망 성능 개선을 위한 방법을 이해 할 수 있다.

MLP 이미지 분석

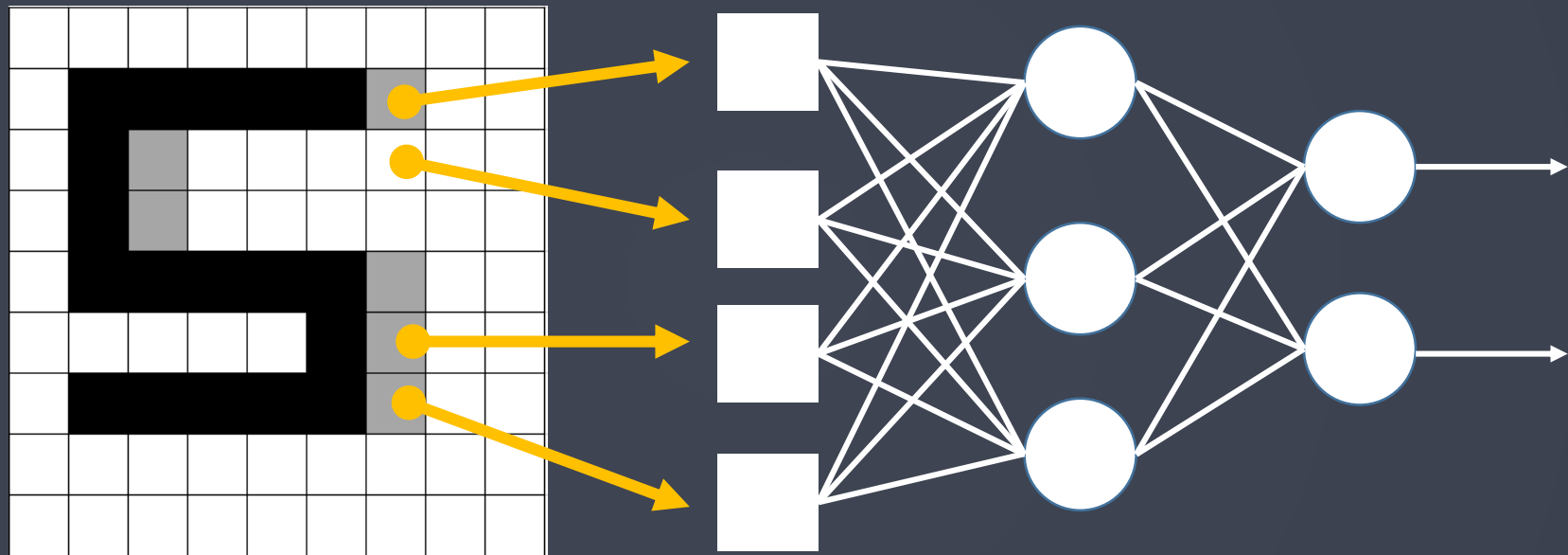
- MLP은 층이 깊어지고 뉴런의 수가 많아지면 가중치 수가 급격히 늘어난다.
- MLP 신경망을 이미지 처리에 사용한다면 이미지의 어떤 특정 패턴이 존재하는 위치에 민감하게 동작하며 패턴의 위치에 종속적인 결과를 얻는다.
- MLP는 아래 세 개의 5는 패턴이 다르다고 판단한다.
- MLP로 이러한 숫자 인식을 하려면 숫자의 크기를 비슷하게 맞추어야 한다.



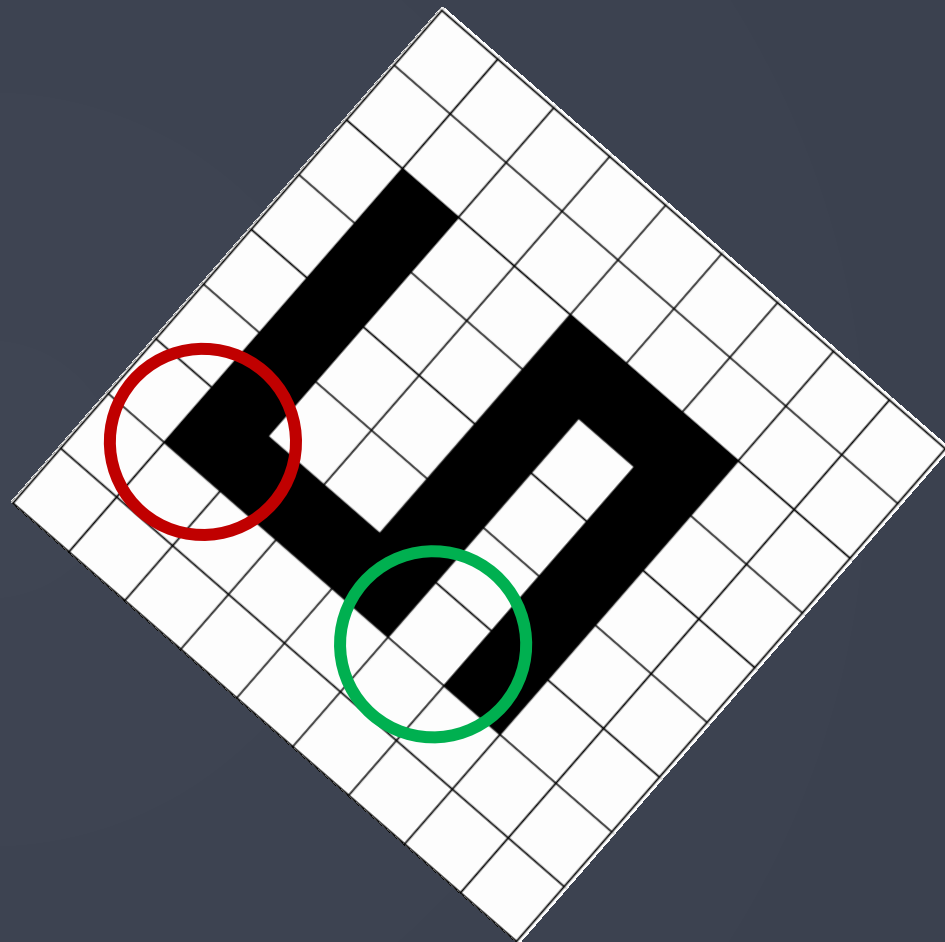
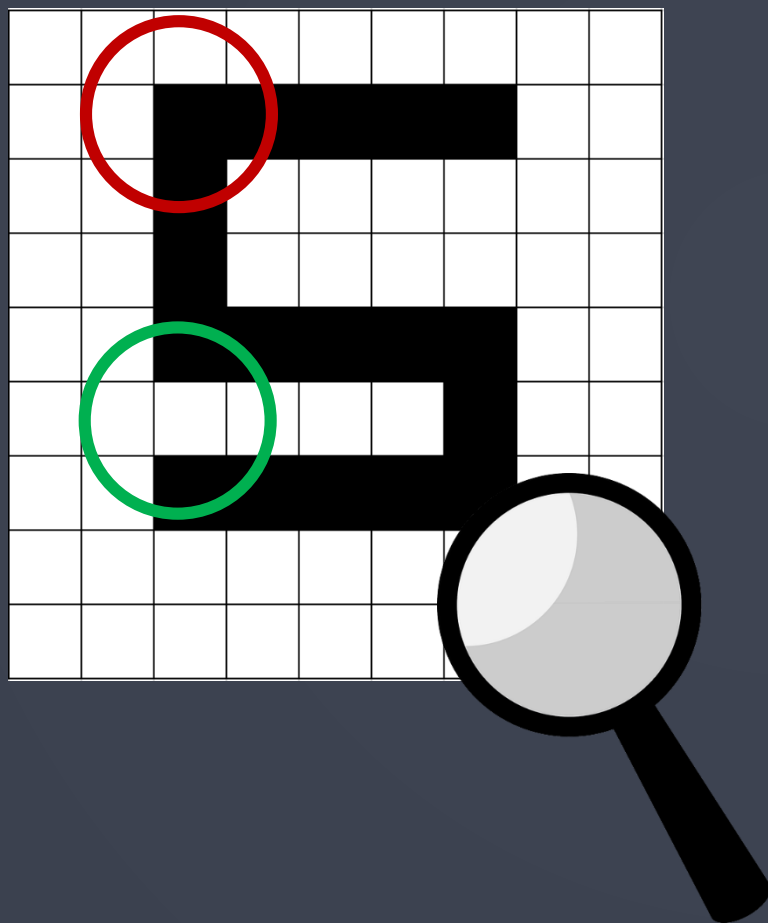
CNN(Convolutional Neural Network)

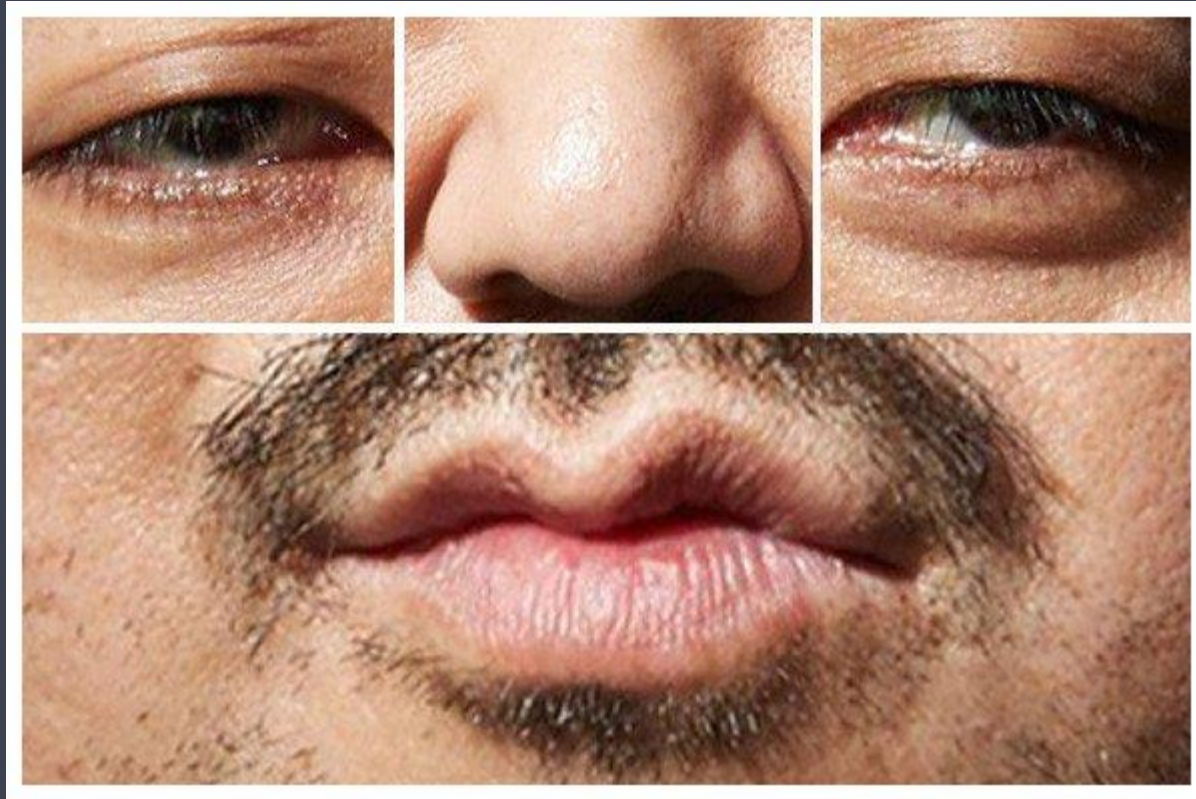


CNN(Convolutional Neural Network)



특징을 추출하자

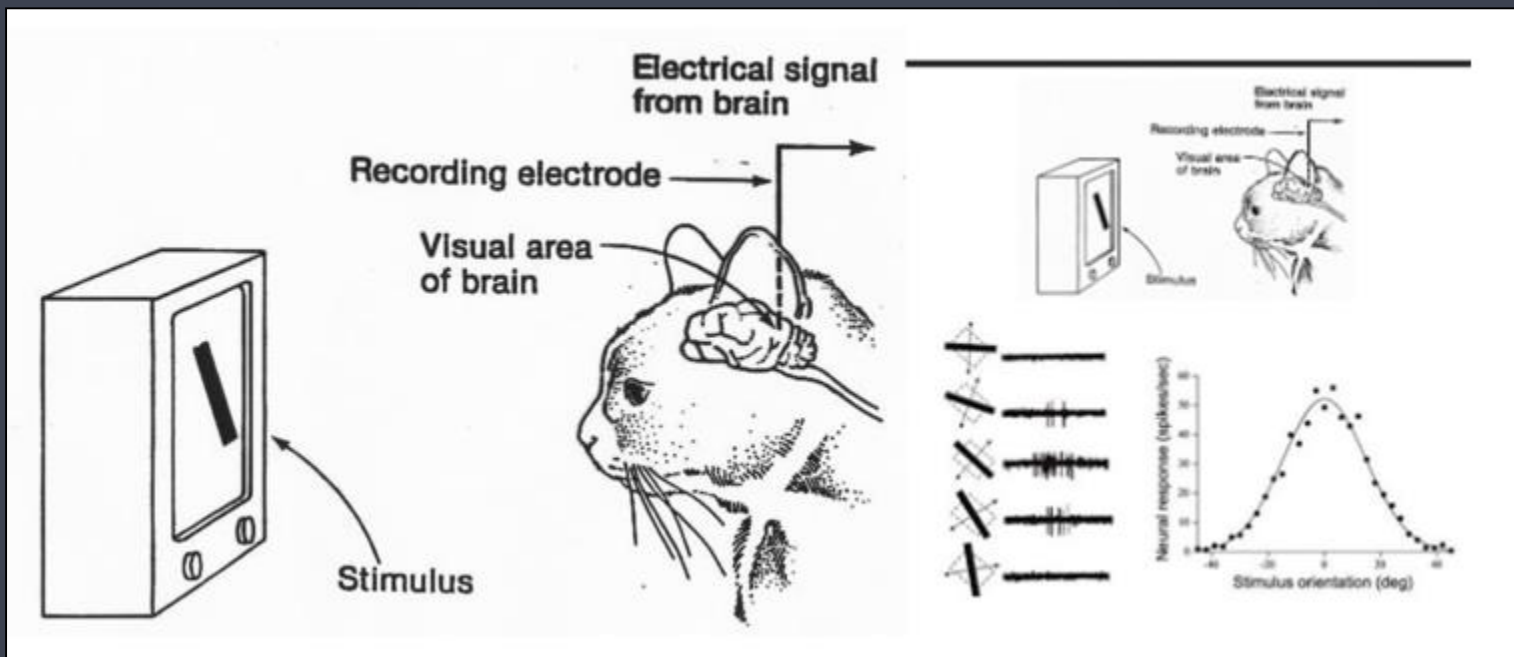






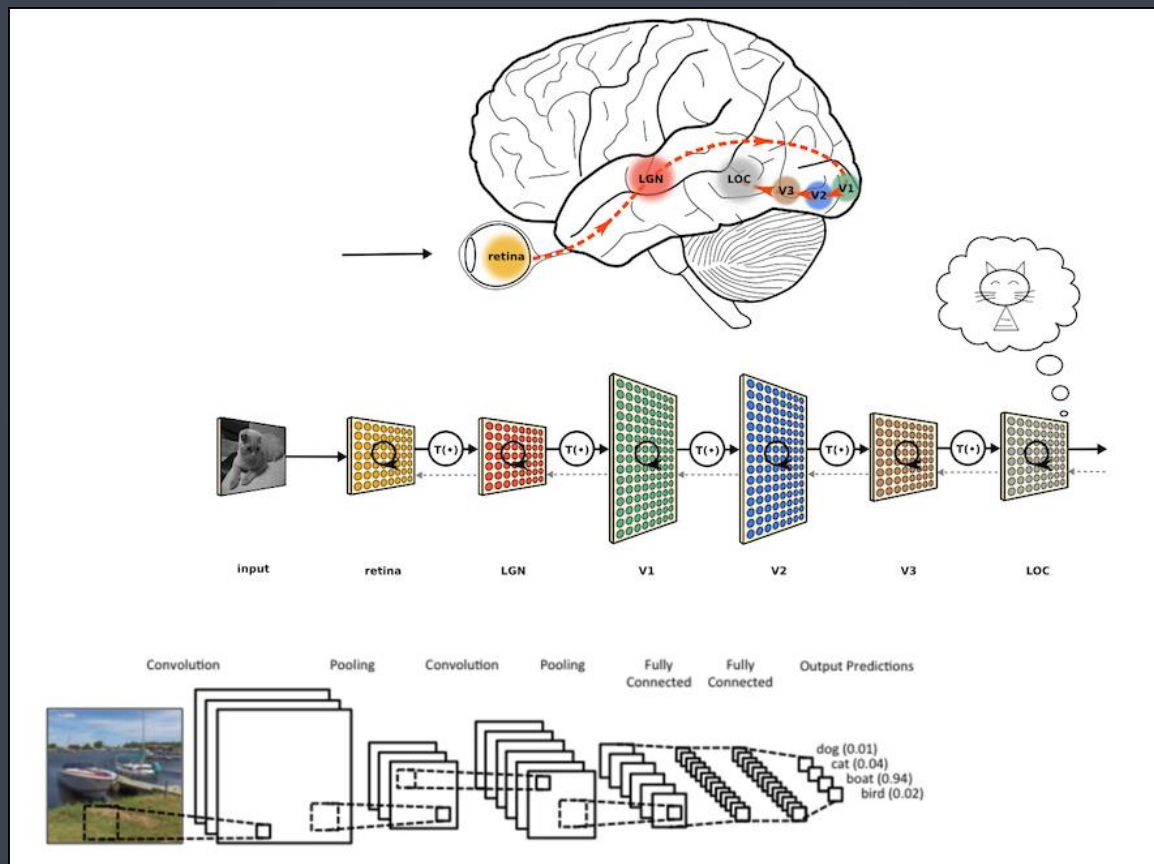
CNN (Convolution Neural Network)

- 1998년 Yann Lecun 교수에 의해 1950년 대 수행했던 고양이의 뇌 파 실험에 영감을 얻어 이미지 인식을 획기적으로 개선 할 수 있는 CNN 제안

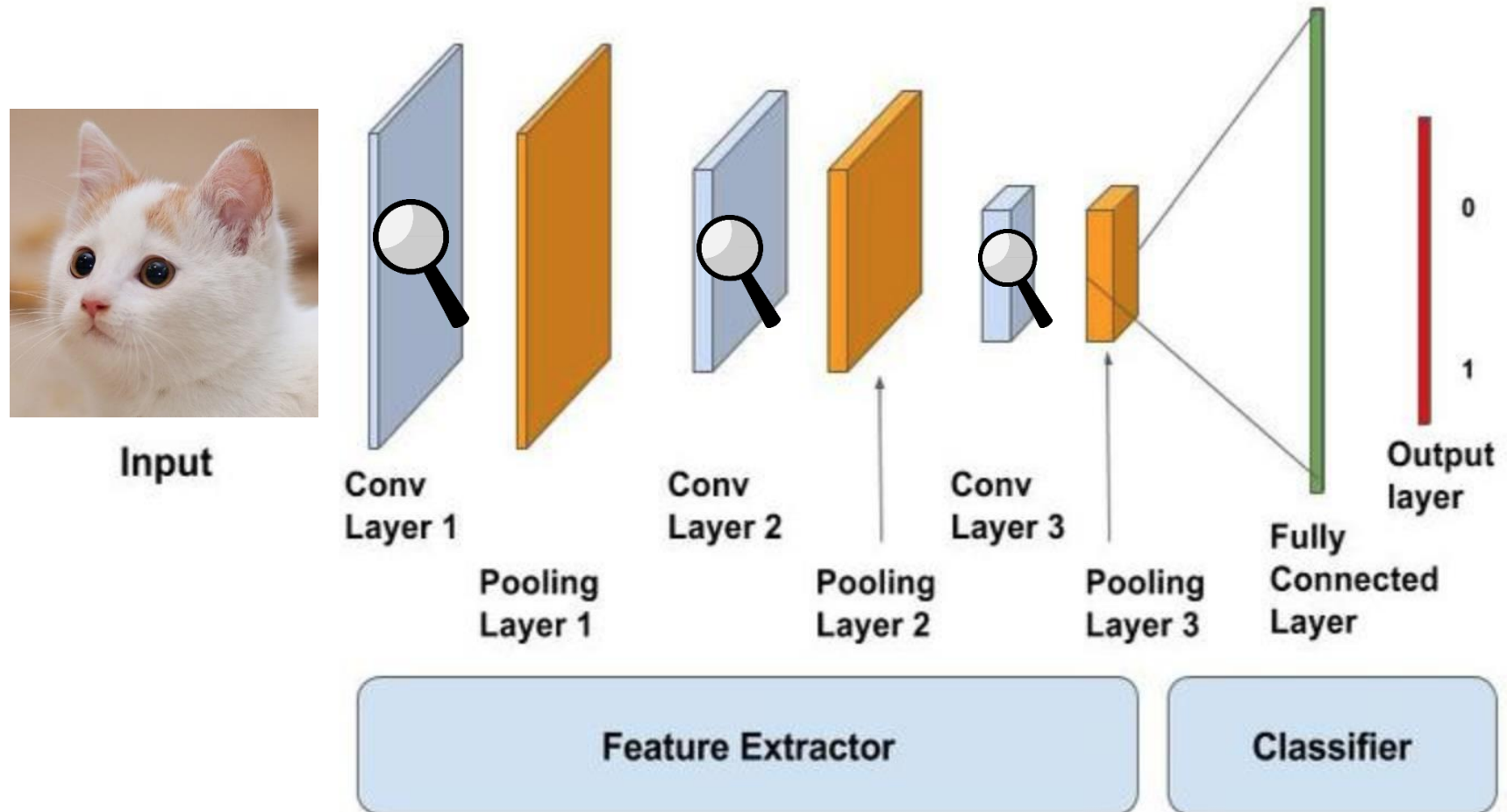


CNN (Convolution Neural Network)

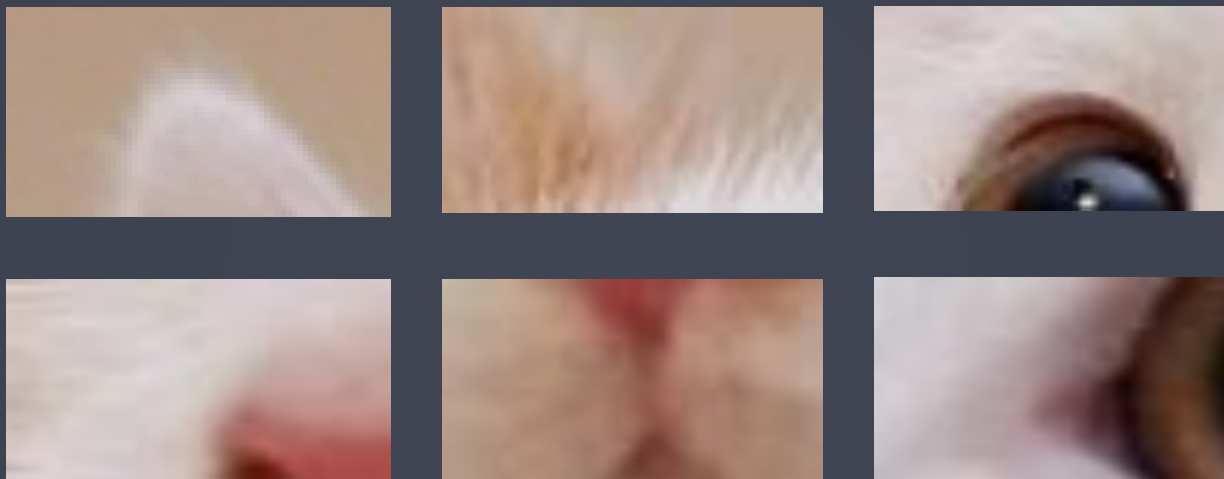
- 고양이의 눈으로 보는 사물의 형태에 따라 뇌의 특정영역 (뉴런)만 활성화 된다는 결과를 기반으로 제안



CNN 구조

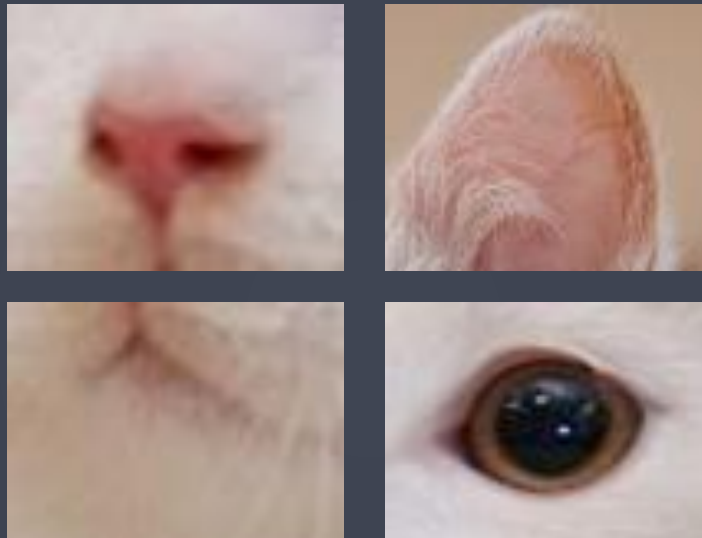


이미지를 가까운 거리에서 본다면



1 단계 : 세모, 동그라미, 가로

조금 더 떨어져서 보면



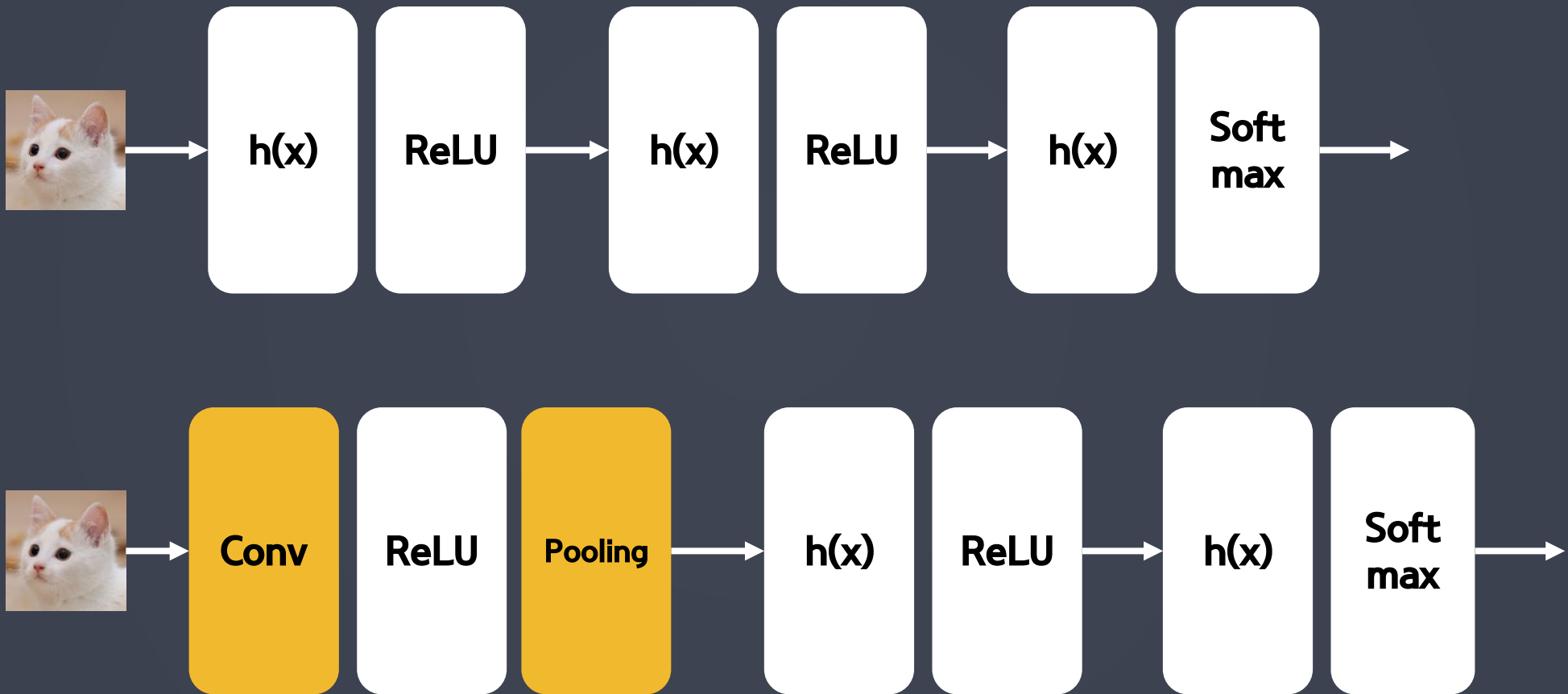
2 단계 : 코, 귀, 입, 눈

더 떨어져서 보면

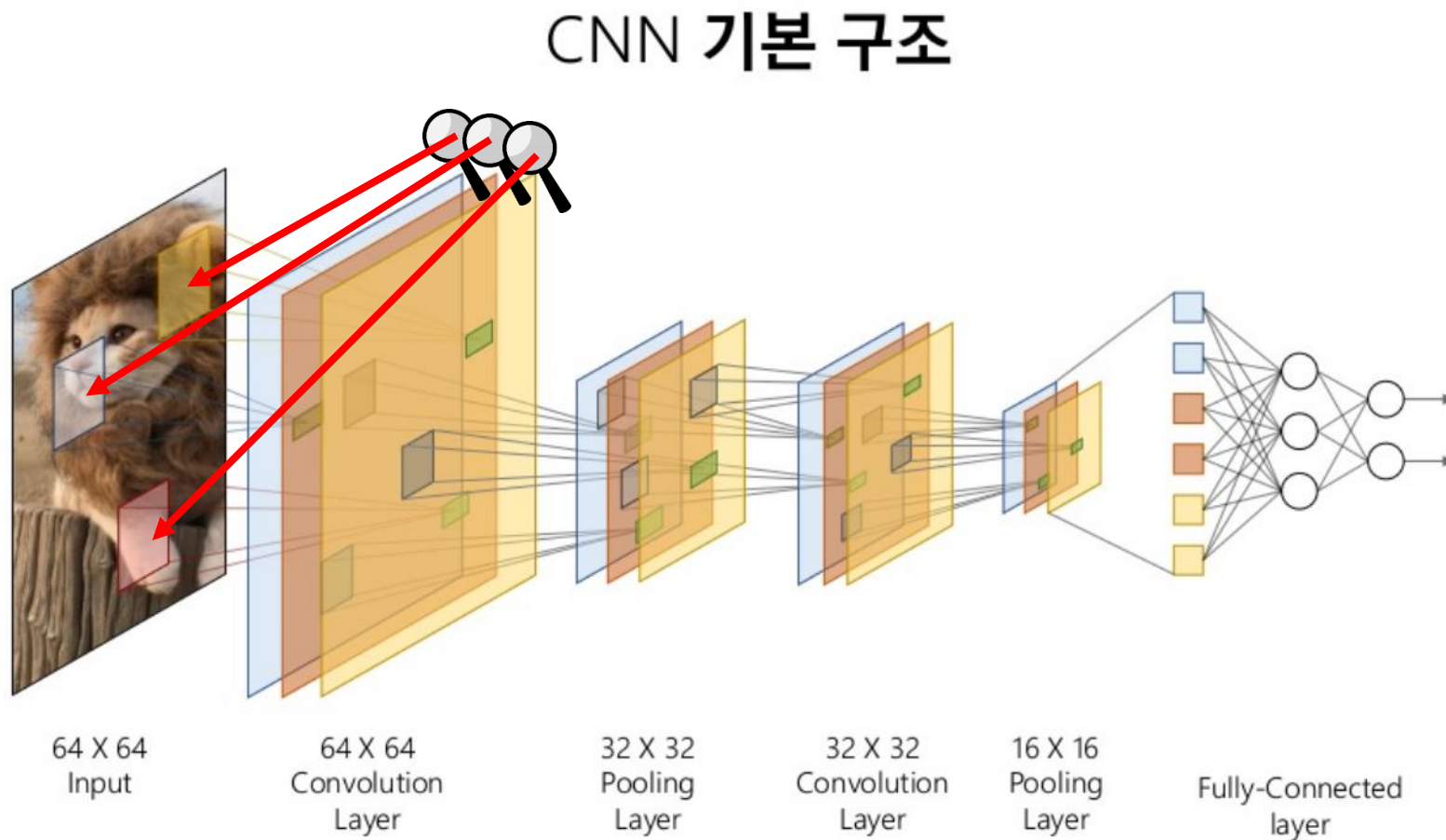


3 단계 : 고양이

CNN 구조



CNN 구조



CNN (Convolution Neural Network)

- CNN(합성곱)은 입력된 이미지에서 특징을 추출하기 위해 마스크(필터)를 도입하는 방법
- 이미지 전체 영역에 대해 서로 동일한 연관성(중요도)로 처리하는 대신 특정 범위에 한정해 처리한다면 훨씬 효과적일 것이라는 아이디어 제시

1	0	1	0
1	0	1	0
1	1	1	1
0	0	1	0

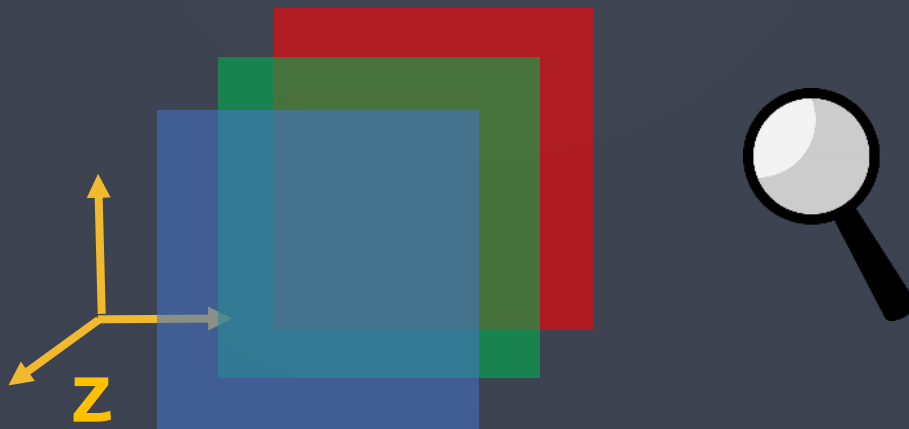
이미지

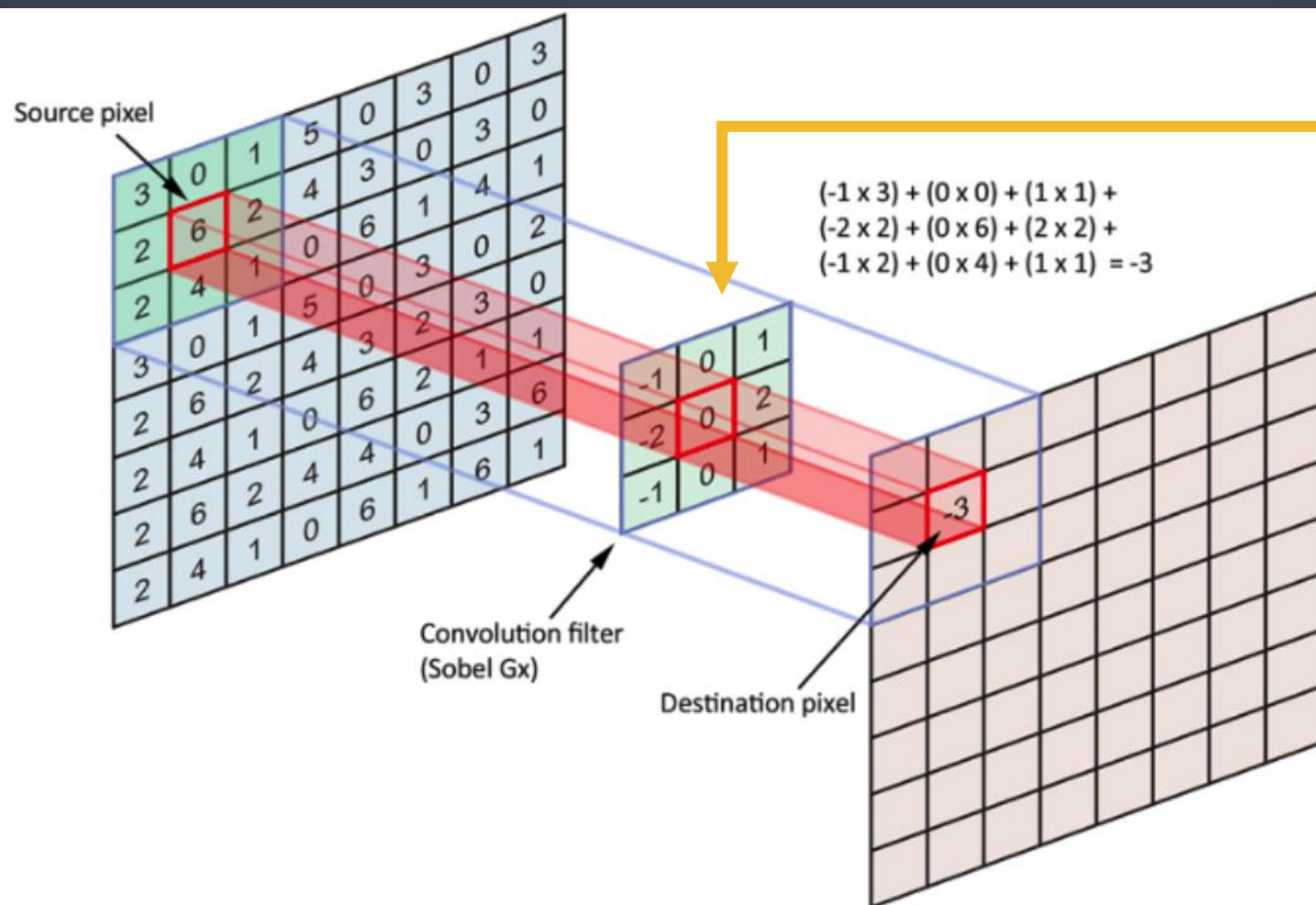
1	0
0	1

마스크
(필터)

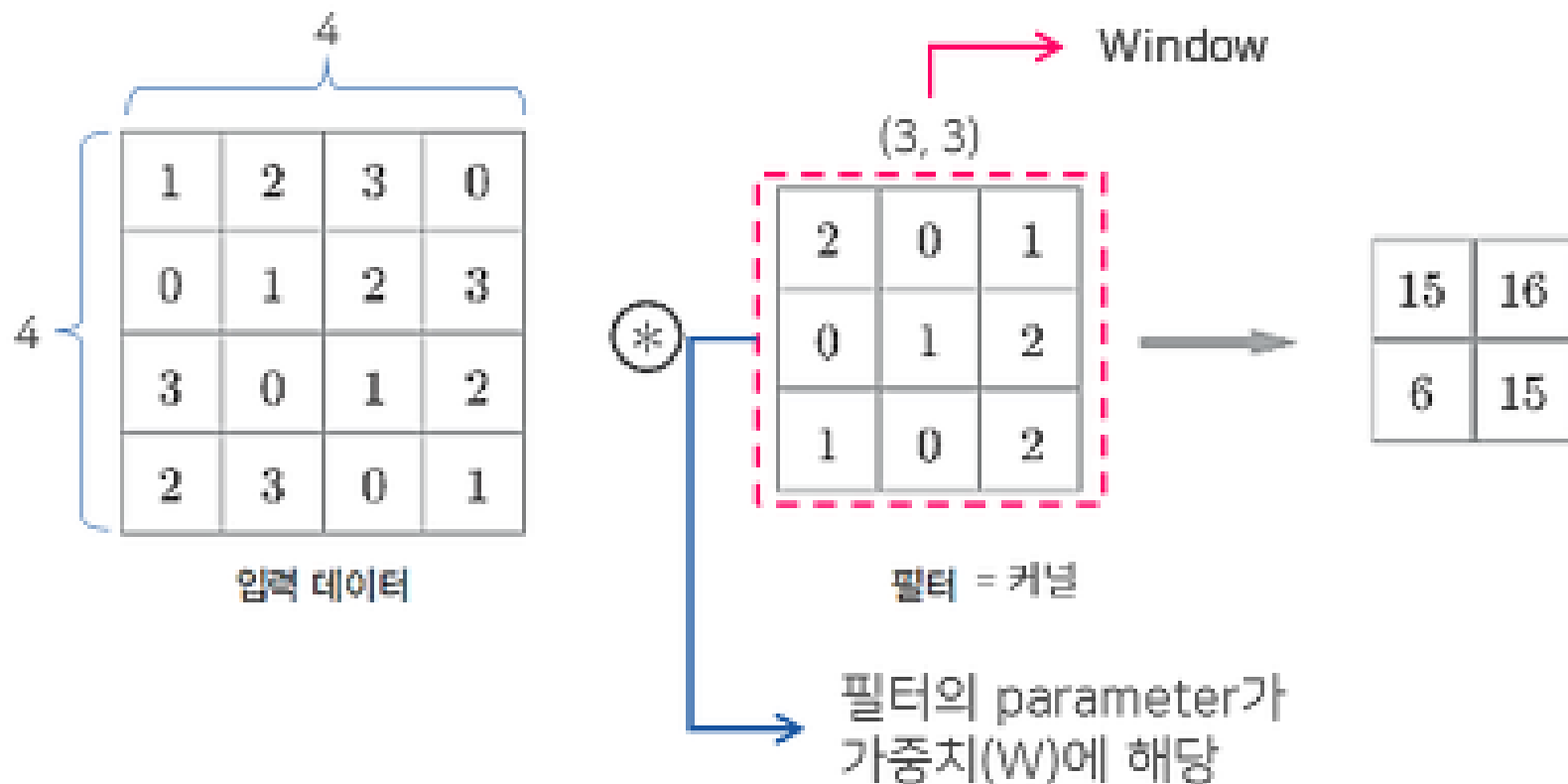
CNN (Convolution Neural Network)

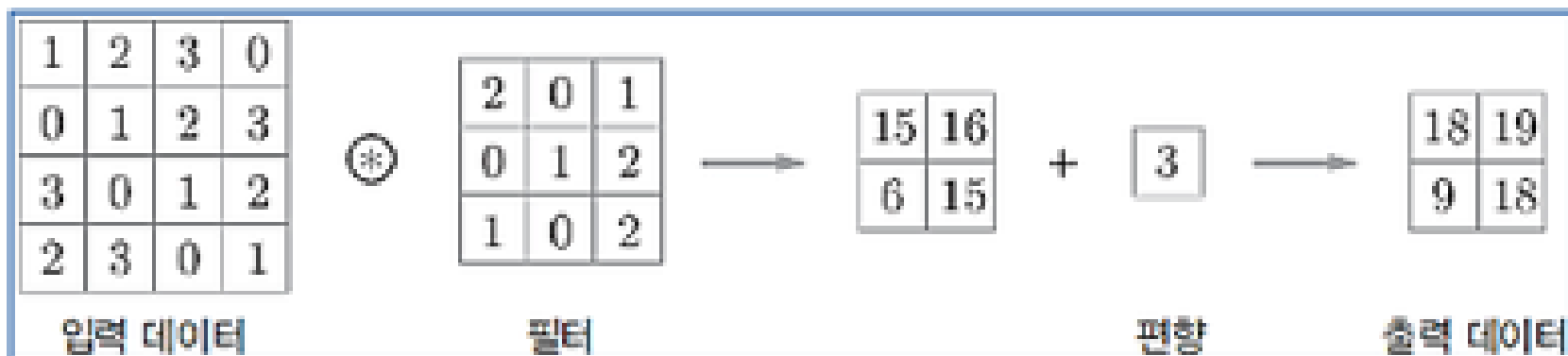
- 합성곱 계층의 입출력 데이터를 특성맵(feature map)이라고 한다.
- 특성맵은 평면을 구성하는 2차원으로 구성된다(채널이라고도 부른다).
- 흑백으로 코딩된 경우 (MNIST처럼) 흑백의 그레이 스케일만 나타내면 되므로 깊이는 1이 된다.
- 입력신호가 RGB 신호로 코딩된 경우, 깊이 축의 차원은 세가지 색을 각각 나타내는 3이 된다. (데이터의 형상 정보를 유지 할 수 있다.)

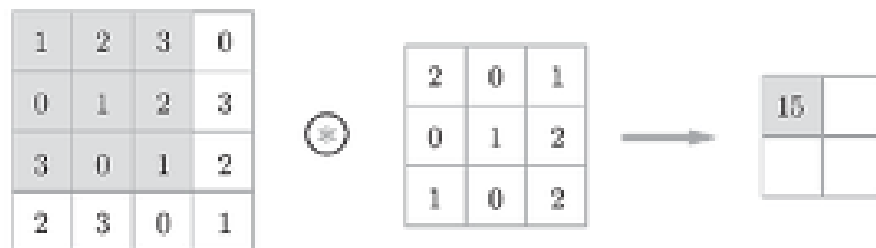




필터(filter)

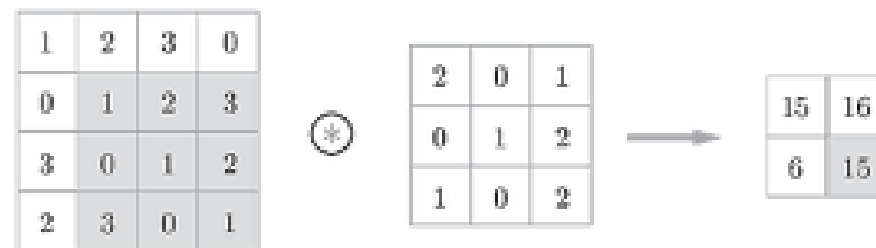
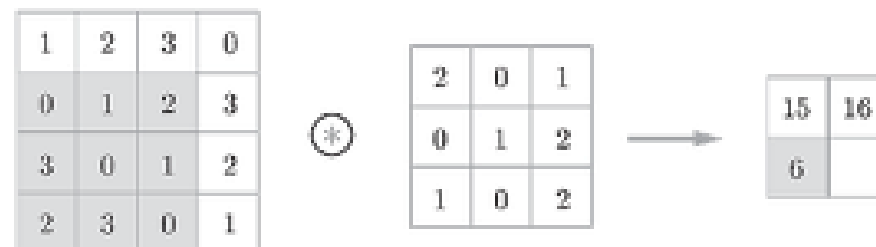
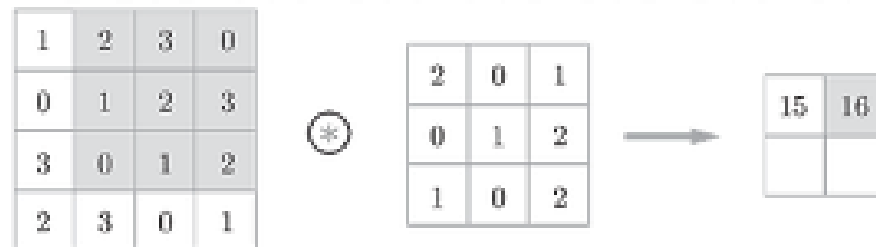






- 단일 곱셈-누산(FMA, Fused Multiply-Add)

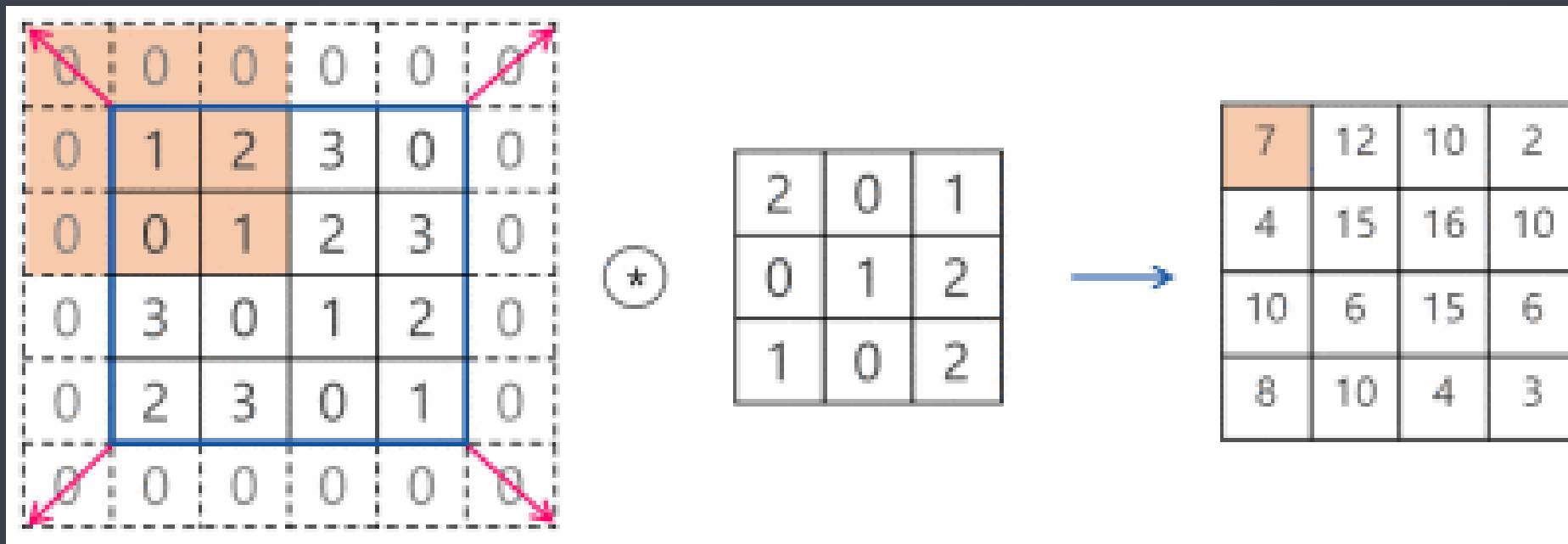
$$1 * 2 + 2 * 0 + 3 * 1 + 0 * 0 + 1 * 1 + 2 * 2 + 3 * 1 + 0 * 0 + 1 * 2 = 15$$



패딩(Padding)

- 필터의 크기는 5x5나 7x7 등 임의의 크기로 정할 수 있다.
- 필터의 크기로 인해 가장자리 부분의 데이터가 부족해서 입력과 출력의 크기가 달라지게 되는데
- 이를 보정하기 위해서 입력신호의 가장자리 부분에 0을 미리 채워넣는 것을 패딩(padding) 이라고 한다.
- Conv2D 계층에서는 padding 인자를 사용하여 패딩을 지정할 수 있다. valid로 설정하면 패딩을 사용하지 말라는 뜻이다.
 - same을 지정하면 출력의 차원이 입력과 같아지도록 적절한 수의 패딩을 자동으로 입력하라는 것이다.

패딩(Padding)



축소샘플링

- 합성곱을 수행한 결과 신호를 다음 계층으로 전달할 때, 모든 정보를 전달하지 않고 일부만 샘플링하여 넘겨주는 작업을 축소 샘플링(subsampling)이라고 한다.
- 축소 샘플링을 하는 이유는 좀 더 가치 있는 정보만을 다음 단계로 넘겨주기 위해서이다.
- 커널 수를 늘리면 특성맵의 숫자가 점차 커지게 된다. 머신러닝의 최종 목적은 정보를 결국 줄여나가야 하며 따라서 핵심 정보만 다음 계층으로 전달하는 장치가 필요하다.

스트라이드(Stride)

- 축소 샘플링에는 크게 나누어 스트라이드(stride)와 풀링(pooling) 등 두 가지 기법이 있다.
- 스트라이드는 합성곱 필터링을 수행할 때 패치를 (예를 들면 3x3 크기)를 한 픽셀씩 옆으로 이동하면서 출력을 얻지 않고, 2 픽셀씩 또는 3 픽셀씩 건너 뛰면서 합성곱을 수행하는 방법이다.
 - 이를 스트라이드 2 또는 스트라이드 3이라고 하는데, 이렇게 하면 출력 특성맵의 크기를 1/4 또는 1/9로 줄일 수 있다.

스트라이드(Stride)

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

⊗

2	0	1
0	1	2
1	0	2



15		

스트라이드 : 2

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

⊗

2	0	1
0	1	2
1	0	2



15	17	

풀링(Pooling)

- 풀링이란 CNN에서 합성곱 수행 결과를 다음 계층으로 모두 넘기지 않고, 일정 범위내에서 (예를 들면 2x2 픽셀 범위) 가장 큰 값을 하나만 선택하여 넘기는 방법을 사용한다.
- 이렇게 지역내 최대 값만 선택하는 풀링을 최대풀링(max pooling)이라고 한다.
 - 최대 풀링을 하면 작은 지역 공간의 대표 정보만 남기고 나머지 신호들을 제거하는 효과를 얻는다.

1	2	0	7	1	0
0	9	2	3	2	3
3	0	1	2	1	2
2	4	0	1	0	1
6	0	1	2	1	2
2	4	0	1	8	1

pooling



9	7
6	8

pooling



1	1	2	0	7	1
3	0	9	2	3	2
3	0	1	1	2	1
3	2	4	0	1	0
2	6	0	1	2	1
1	2	4	0	1	8

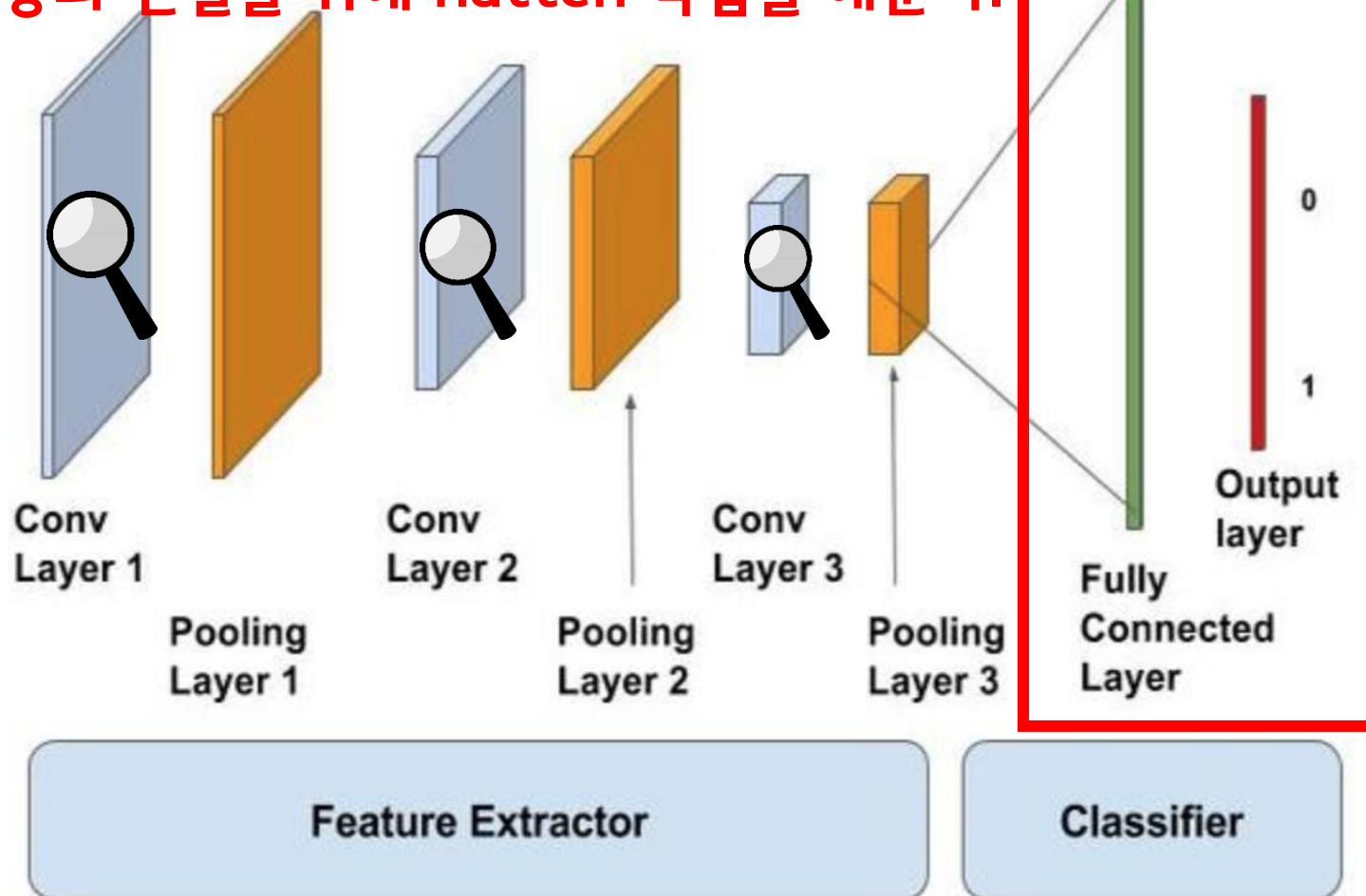
입력의 변화에 영향을 적게 받는다

CNN(Convolutional Neural Network)

전결합층과 연결을 위해 flatten 작업을 해준다.



Input



```
Conv2D ( filters = 32,  
         kernel_size = (5, 5),  
         padding='valid',  
         input_shape=(28, 28, 1),  
         activation='relu',  
         strides = (2, 2) )
```

- 첫번째 인자 : Convolution 필터의 수.
- 두번째 인자 : Convolution 커널의 (행, 열).

```
Conv2D ( filters = 32,  
        kernel_size = (5, 5),  
        padding='valid',  
        input_shape=(28, 28, 1),  
        activation='relu',  
        strides = (2, 2) )
```

- padding : 경계 처리 방법을 정의.
 - 'valid' : 유효한 영역만 출력. 따라서 출력 이미지 사이즈는 입력 사이즈보다 작다.
 - 'same' : 출력 이미지 사이즈가 입력 이미지 사이즈와 동일하다.


```
Conv2D ( filters = 32,  
        kernel_size = (5, 5),  
        padding='valid',  
        input_shape=(28, 28, 1),  
        activation='relu',  
        strides = (2, 2) )
```

- `input_shape` : 샘플 수를 제외한 입력 형태를 정의.
 - 모델에서 첫 레이어일 때만 정의.(행, 열, 채널 수)로 정의.
 - 흑백영상인 경우에는 채널이 1이고, 컬러(RGB)영상인 경우에는 채널을 3으로 설정.

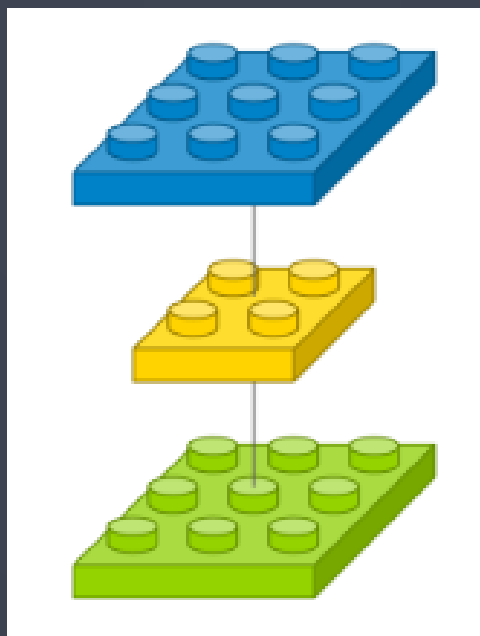
```
Conv2D ( filters = 32,  
         kernel_size = (5, 5),  
         padding='valid',  
         input_shape=(28, 28, 1),  
         activation='relu',  
         strides = (2, 2) )
```

- activation : 활성화 함수 설정.
 - 'relu'
 - 'sigmoid'
 - 'softmax'
 - 'linear'

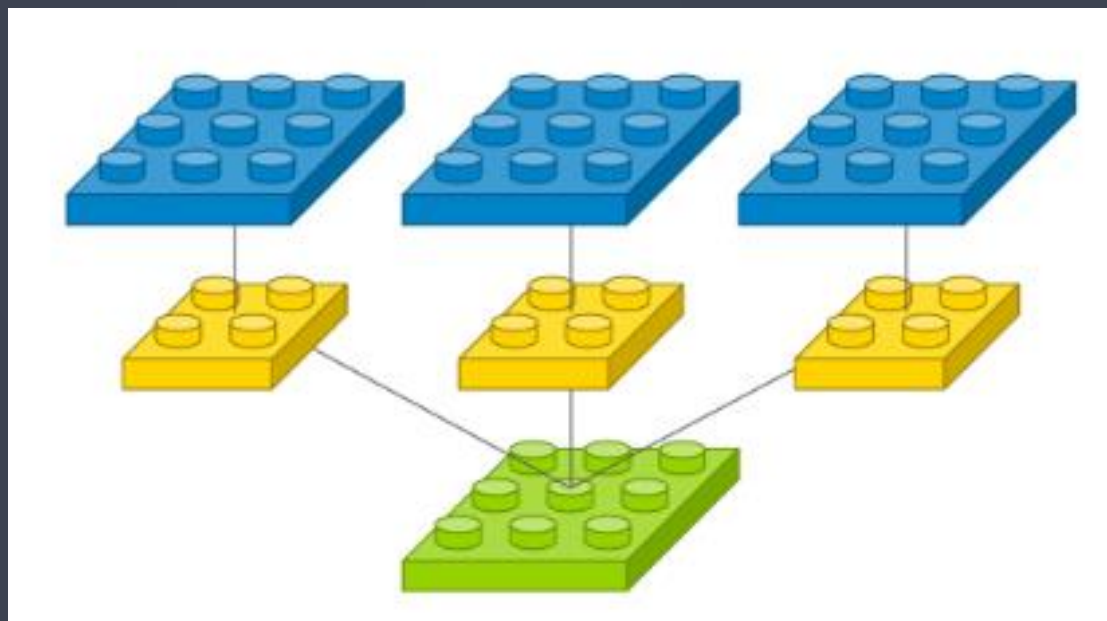
```
Conv2D ( filters = 32,  
        kernel_size = (5, 5),  
        padding='valid',  
        input_shape=(28, 28, 1),  
        activation='relu',  
        strides = (2, 2) )
```

- stride : stride 크기 지정 (행, 열).

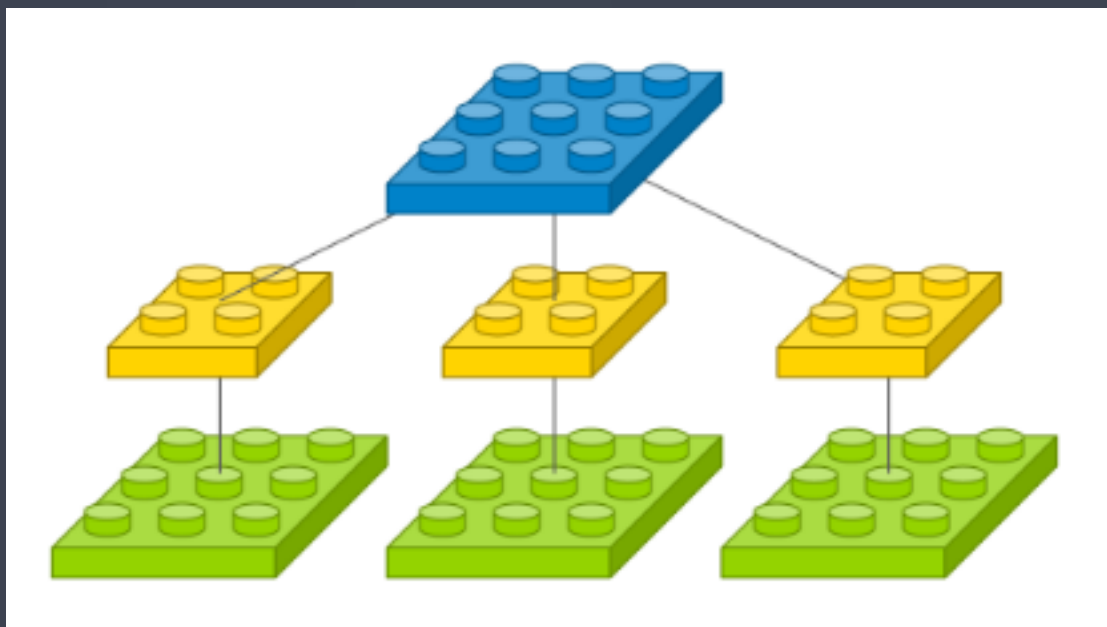
Conv2D (filters = 1,
kernel_size = (2, 2),
padding='same',
input_shape=(3, 3, 1))



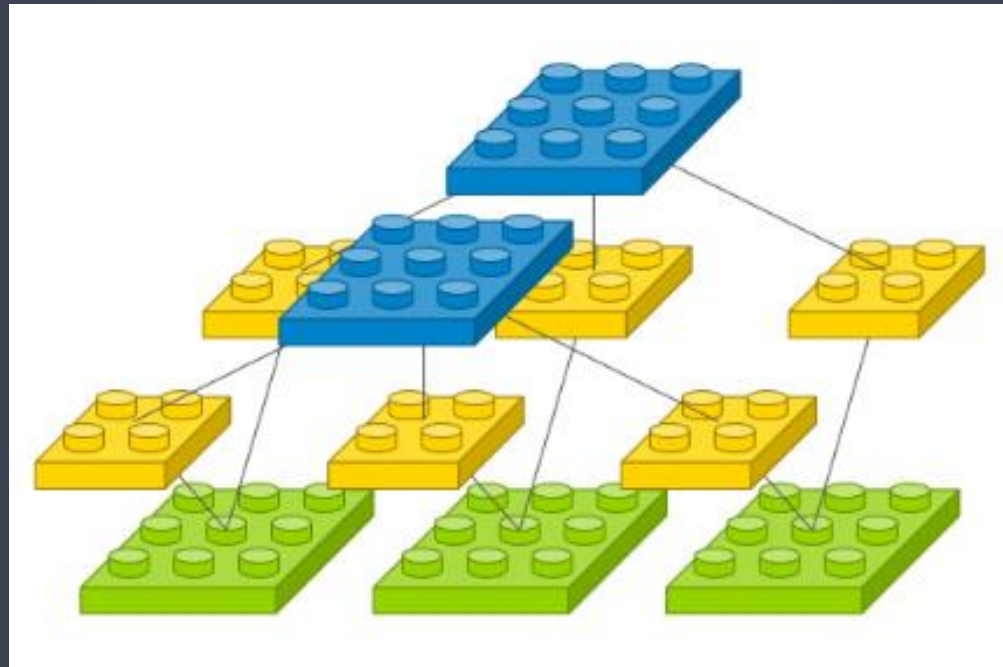
Conv2D (filters = 3,
kernel_size = (2, 2),
padding='same',
input_shape=(3, 3, 1))



Conv2D (filters = 1,
kernel_size = (2, 2),
padding='same',
input_shape=(3, 3, 3))

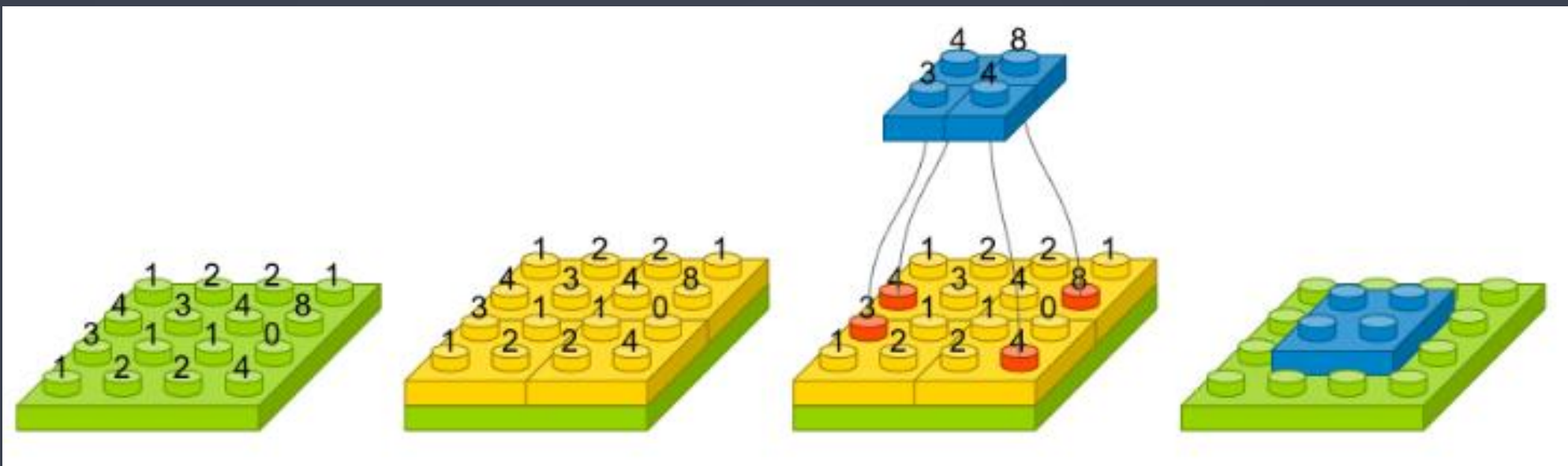


Conv2D (filters = 2,
kernel_size = (2, 2),
padding='same',
input_shape=(3, 3, 3))



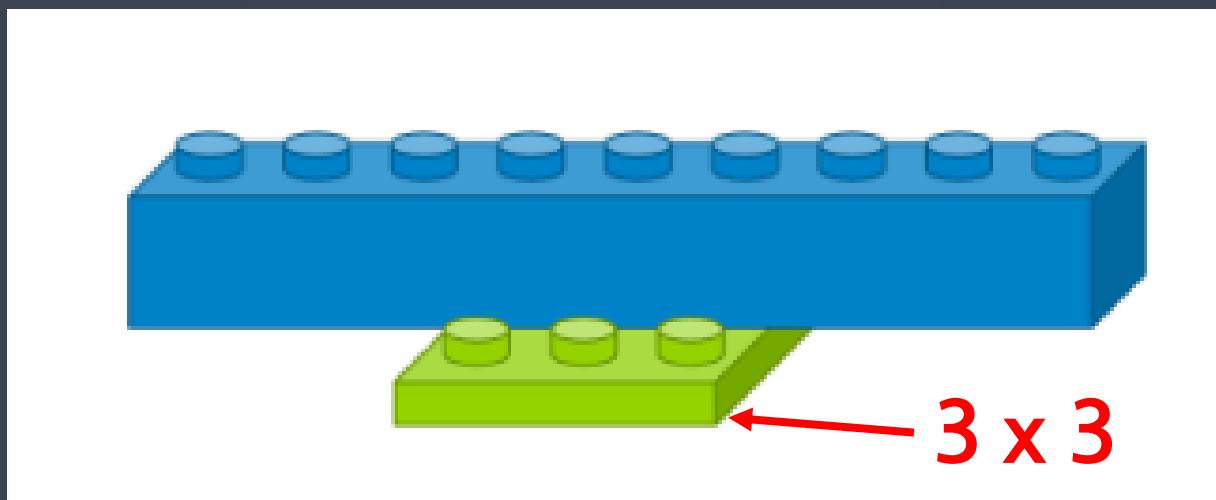
MaxPooling2D (**pool_size** = (2, 2), **stride** = (2, 2))

- **pool_size** : max pooling 크기 지정 (행, 열).
- **stride** : stride 크기 지정 (행, 열).



Flatten ()

- 1차원으로 변경하는 함수



Keras활용 MNIST CNN 실습

Keras활용 고양이, 강아지 분류 실습

과대적합 피하는 방법

✓ 학습조기중단 (early stopping)

✓ L1 이나 L2 규제

```
from keras import regularizers
model.add(Dense(64, input_dim=64,
                 kernel_regularizer=regularizers.l2(0.01)))
```

keras.regularizers.l1(0.)

keras.regularizers.l2(0.)

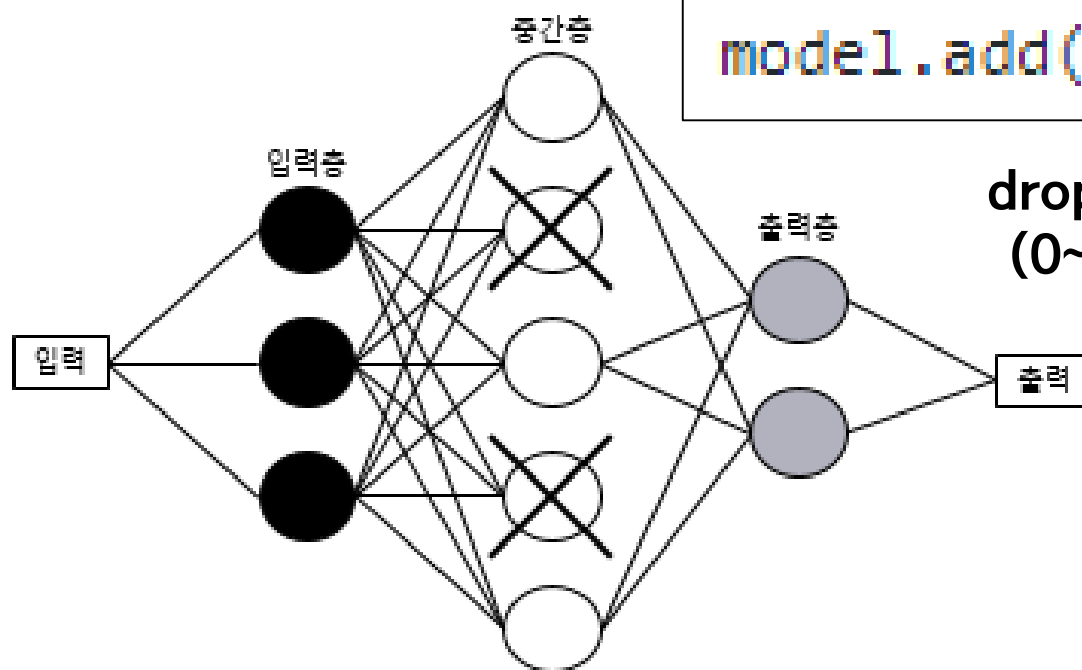
keras.regularizers.l1_l2(l1=0.01, l2=0.01)

과대적합 피하는 방법

- ✓ 학습조기중단 (early stopping)
- ✓ L1 이나 L2 규제
 - 드롭아웃(dropout)
 - 데이터 확장(data augmentation)

과대적합 피하는 방법 - dropout

- 드롭아웃은 학습을 하는 동안에만 적용되고 학습이 종료된 후 예측을 하는 단계에서는 모든 유닛을 사용하여 예측한다.



```
model.add(Dropout(0.3))
```

drop 노드 비율
(0~1 사이 값)

과대적합 피하는 방법 - 데이터 확장

- 과대적합이 일어나는 이유 중 하나는 훈련데이터가 부족하기 때문이다.
- 훈련 데이터가 충분히 많다면 과대적합을 줄일 수 있다.
- 데이터 확장이라 훈련 데이터를 다양하게 변형하여 변형된 새로운 훈련 데이터처럼 사용함으로써 마치 훈련 데이터 수가 늘어난 효과를 얻는 것이다.

과대적합 피하는 방법 - 데이터 확장

- `rotation_range = 360 : 0°` 에서 360° 사이에서 회전
- `width_shift_range = 0.1` : 전체에서 10% 내외 수평이동
- `height_shift_range = 0.1` : 전체에서 10% 내외 수직이동
- `shear_range = 0.5` : 0.5라디안 내외 시계반대방향으로 변형
- `zoom_range = 0.3` : 0.7~1.3배로 축소/확대
- `Horizontal_flip = True` : 수평방향으로 뒤집기
- `Vertical_flip = True` : 수직방향으로 뒤집기

https://tykimos.github.io/2017/06/10/CNN_Data_Augmentation/

과대적합 피하는 방법 - 데이터 확장

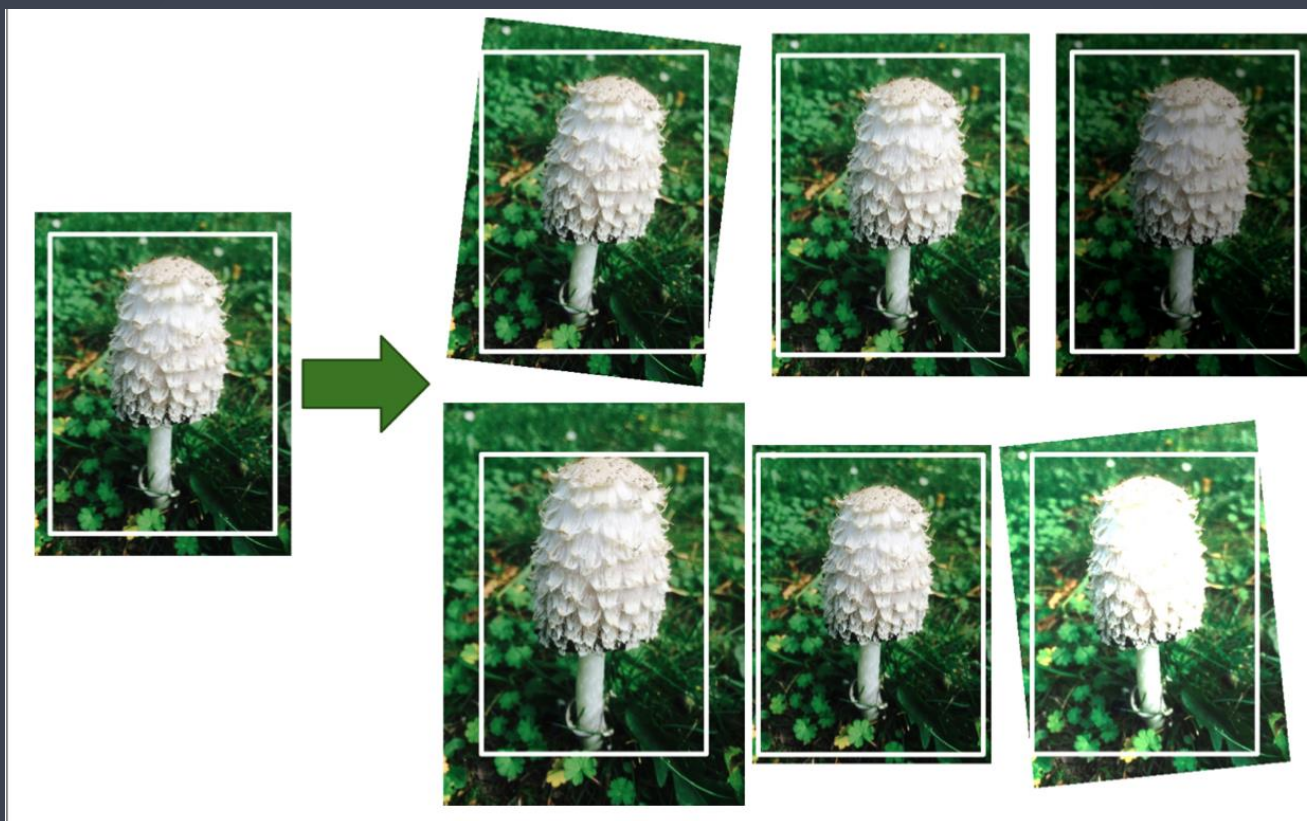
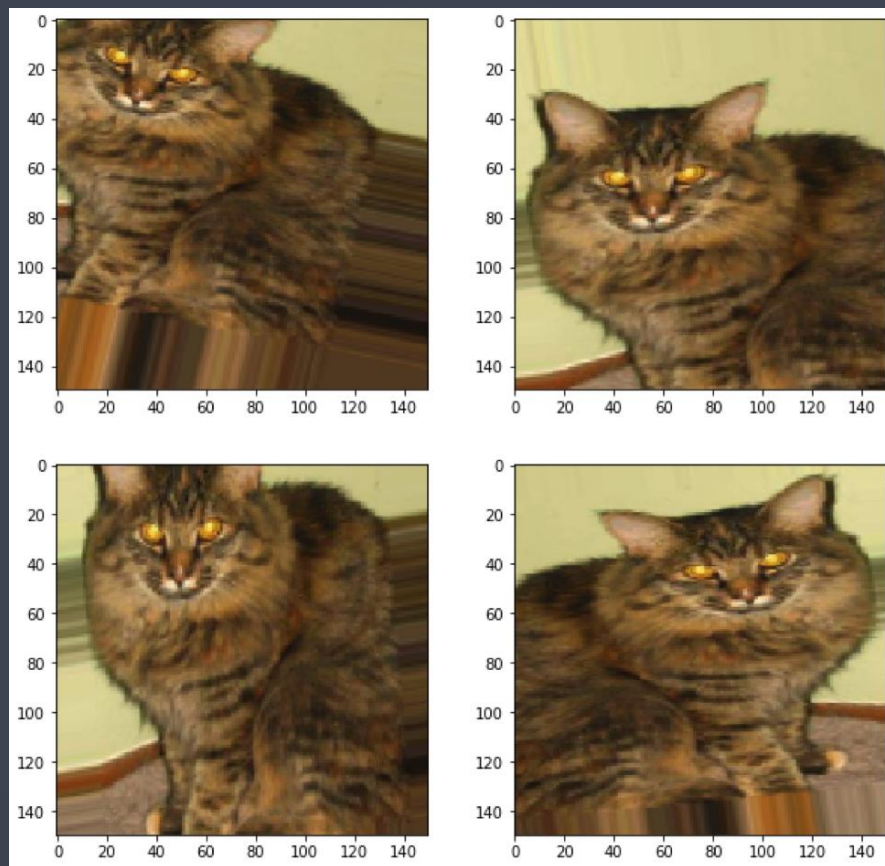


Figure 11-10. Generating new training instances from existing ones

과대적합 피하는 방법 - 데이터 확장



기타 성능 개선

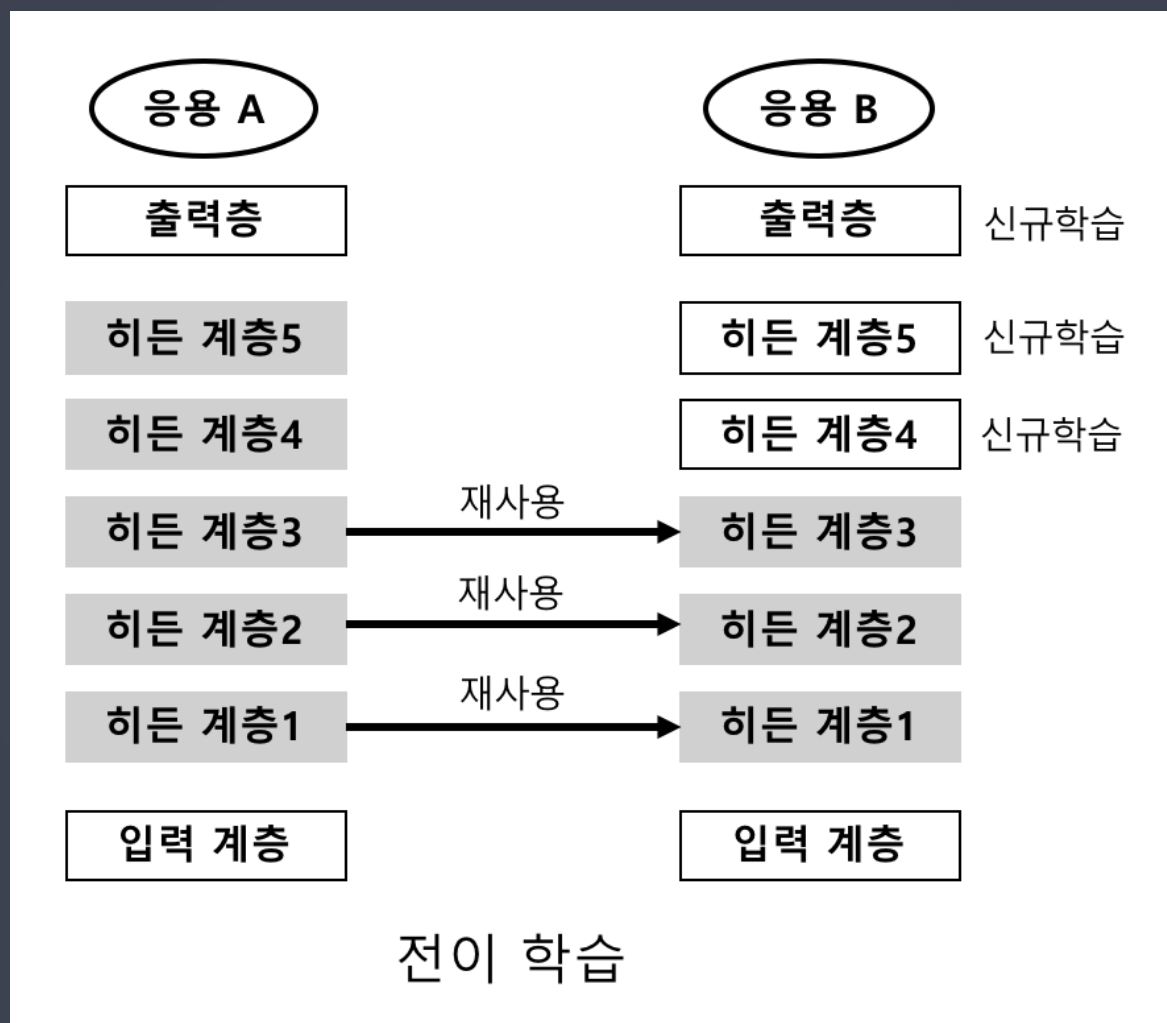
- 가중치 초기값 설정 (Xavier초깃값, He초깃값)
 - 앞 계층의 노드를 이용해 표준편차가 $1 / \sqrt{n}$ 인 정규분포로 초기화하는 방법
- 배치정규화(Batch Normalization)
 - 활성화 함수 앞 또는 뒤에서 평균 0, 분산 1로 정규화하는 방법
 - 각 층에서 값들이 적당히 분포되도록 조정하는 것

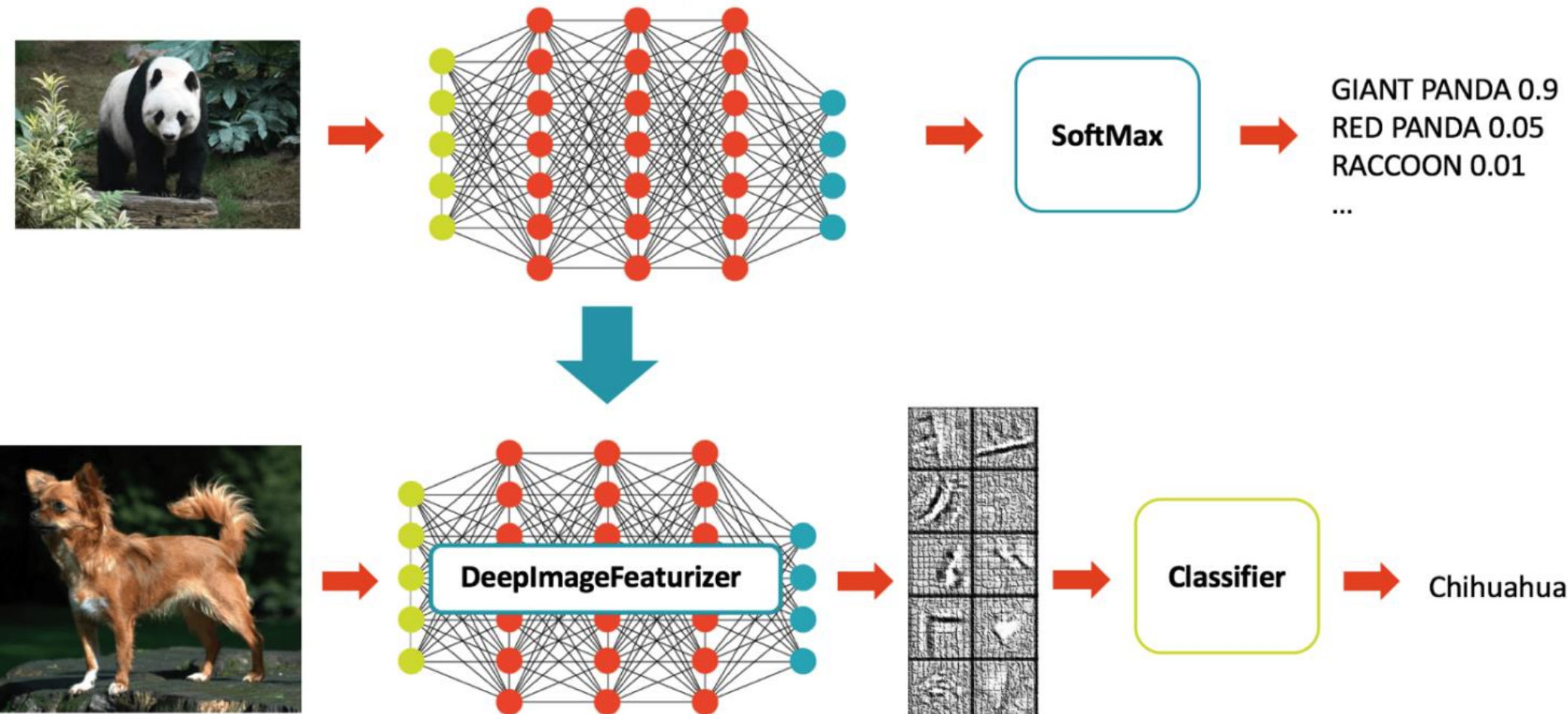
Keras활용 고양이, 강아지 분류 실습 + 데이터 부풀리기

전이학습

- 전이학습이란 다른 데이터 셋을 사용하여 이미 학습한 모델을 유사한 다른 데이터를 인식하는데 사용하는 기법이다.
- 이 방법은 특히 새로 훈련시킬 데이터가 충분히 확보되지 못한 경우에 학습 효율을 높여준다.
- 사전학습모델을 이용하는 방법은 특성 추출(feature extraction) 방식과 미세조정(fine-tuning) 방식이 있다

전이학습(Transfer Learning)







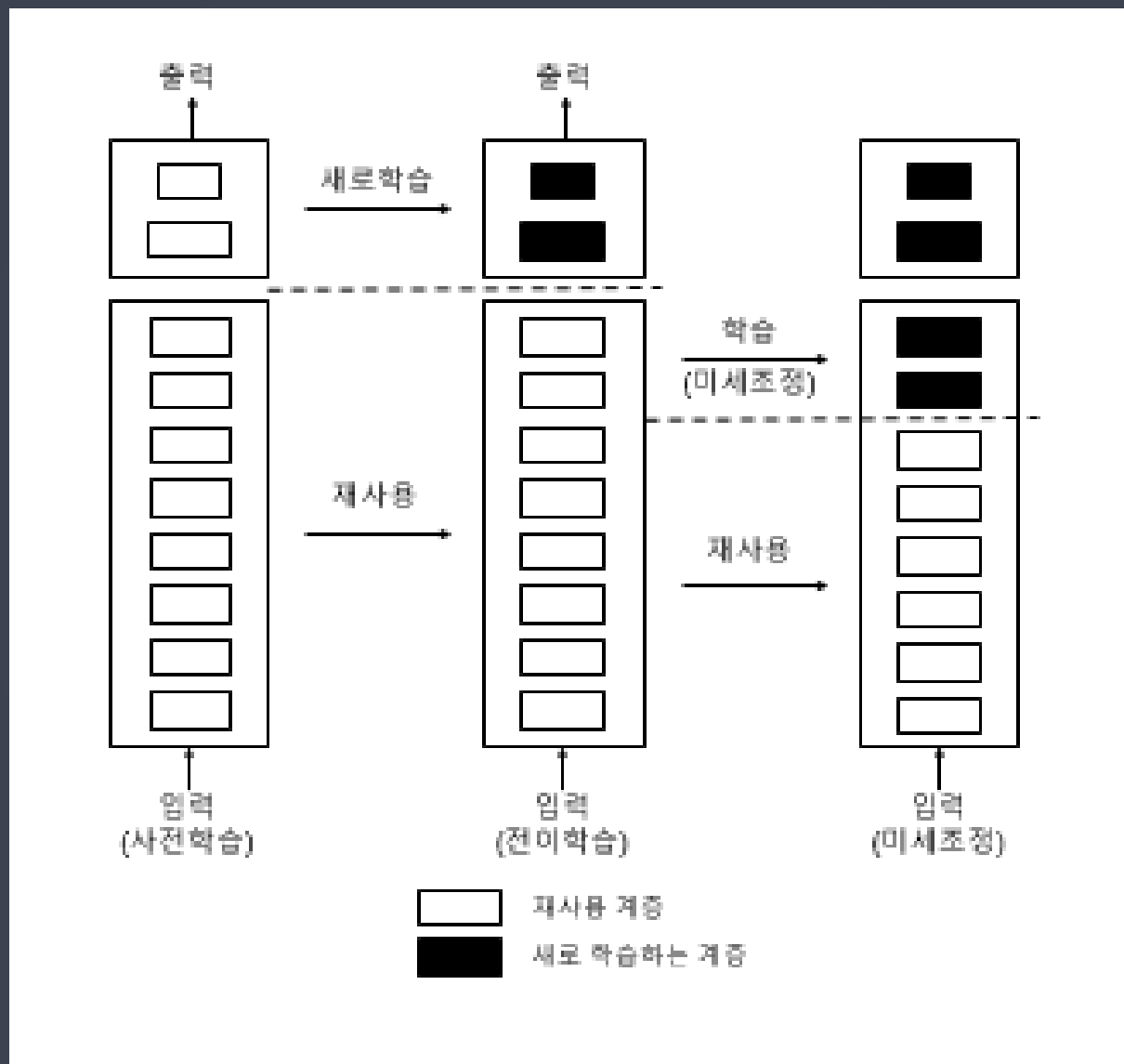
Input



전이학습 - 특성추출방식

- 컨볼루션 베이스 부분만 재사용하는 이유는 이 부분은 상당히 일반적인 학습정보를 포함하고 있기 때문이다.
- 컨볼루션 계층에서도 재사용할 수 있는 정보의 수준은 몇 번째 계층인지에 따라 다르다. 모델의 앞 단의 계층일수록 에지, 색상, 텍스처 등 일반적인 정보를 담는다.
- 반면에 뒤 부분의 깊은 계층일수록 추상적인 정보를 담는다 (예를 들어 고양이 귀, 강아지 귀 등).
- 새롭게 분류할 클래스의 종류가 사전 학습에 사용된 데이터와 특성이 매우 다르면, 컨볼루션 베이스 전체를 재사용해서는 안되고 앞단의 일부 계층만을 재사용해야 한다.

전이학습(Transfer Learning) - 미세조정방식



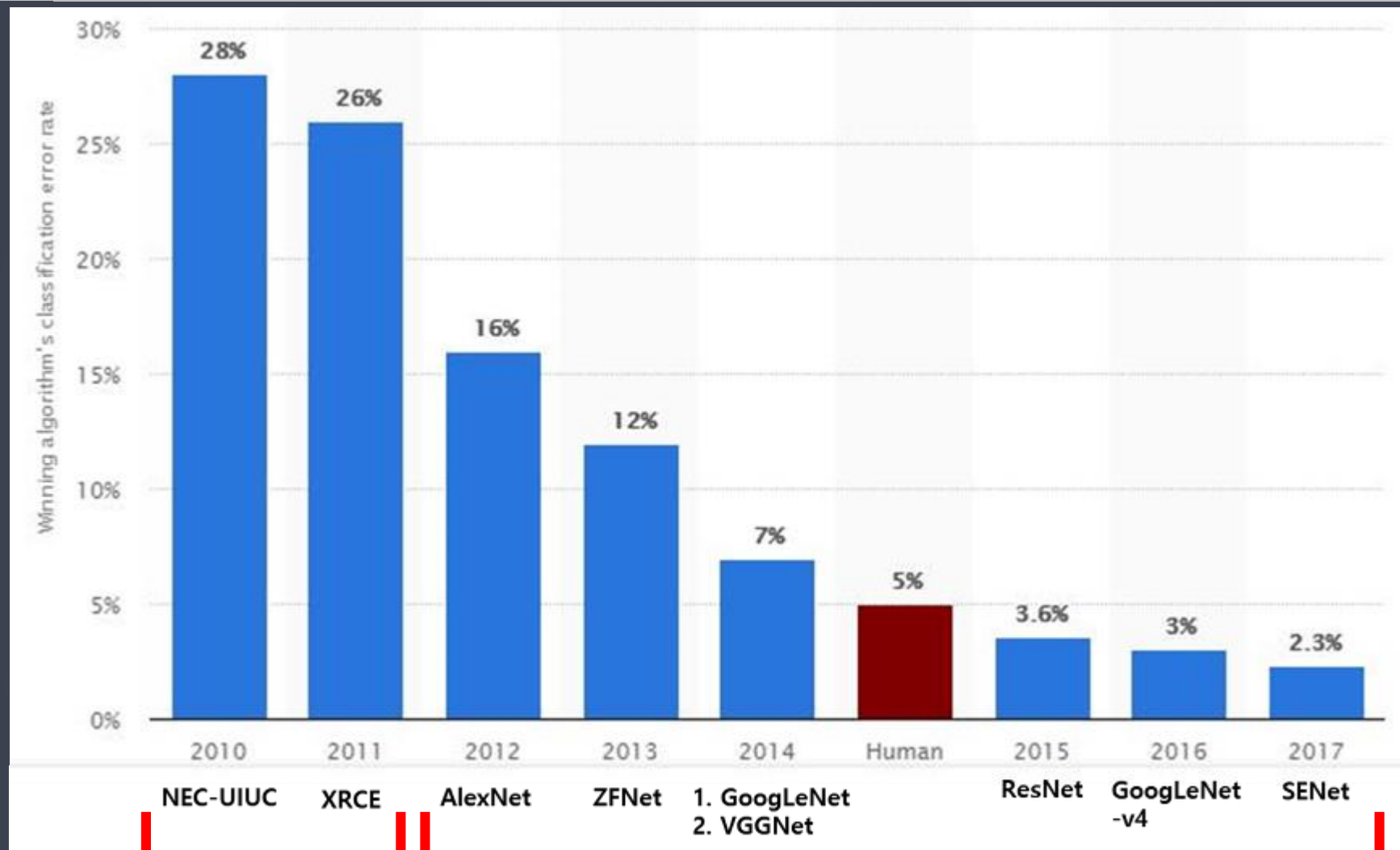
전이학습 - 미세조정방식

- 모델 베이스 중 상위 몇개의 계층은 전결합층 분류기와 함께 새로 학습시키는 방식이다.
- 최종 분류기의 계수가 랜덤하게 초기화 되어 있으므로 이를 먼저 학습시키며 이때 사전학습 모델의 컨볼루션 베이스를 초기에는 고정해야 한다.
- 먼저 분류기를 계수를 학습시킨 다음에 (즉, 이 동안은 미세조정을 하지 않도록 상위계층의 계수를 고정시켜 두고), 그 이후에 미세조정을 해야 한다.
- 처음부터 베이스 상위계층의 계수를 같이 훈련시키면 분류기에서 발생하는 큰 에러 값으로 인해, 사전 학습된 정보가 많이 손실된다.

전이학습 - 미세조정방식

- 1) 사전 학습된 기본 네트워크 상단에 새로운 네트워크를 추가한다.
 - 2) 기본 네트워크를 고정시킨다.
 - 3) 새로 추가한 부분을 학습시킨다.
 - 4) 기본 계층중에 학습시킬 상위 부분의 고정을 푼다
 - 5) 고정을 푼 계층과 새로 추가한 계층을 함께 훈련시킨다.
- 미세 조정을 천천히 수행하기 위해서 느린 학습 속도를 선택한다. 갑자기 큰 변화를 주면 사전 학습된 내용이 훼손되기 때문이다.

Keras활용 고양이, 강아지 분류 전이학습 실습



Feature
Engineering

Convolutional Neural Networks

VGG-16

Architectures

ImageNet Challenge 2014

(3x3 filter)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Classification

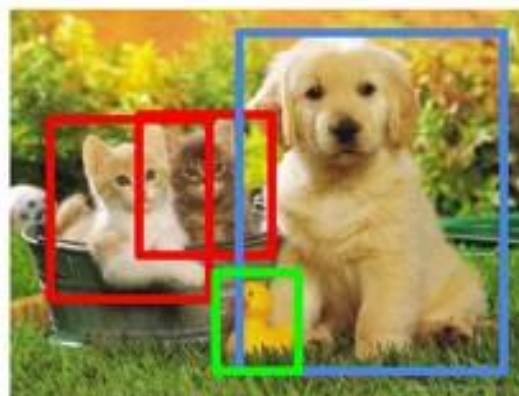


CAT

Classification
+ Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



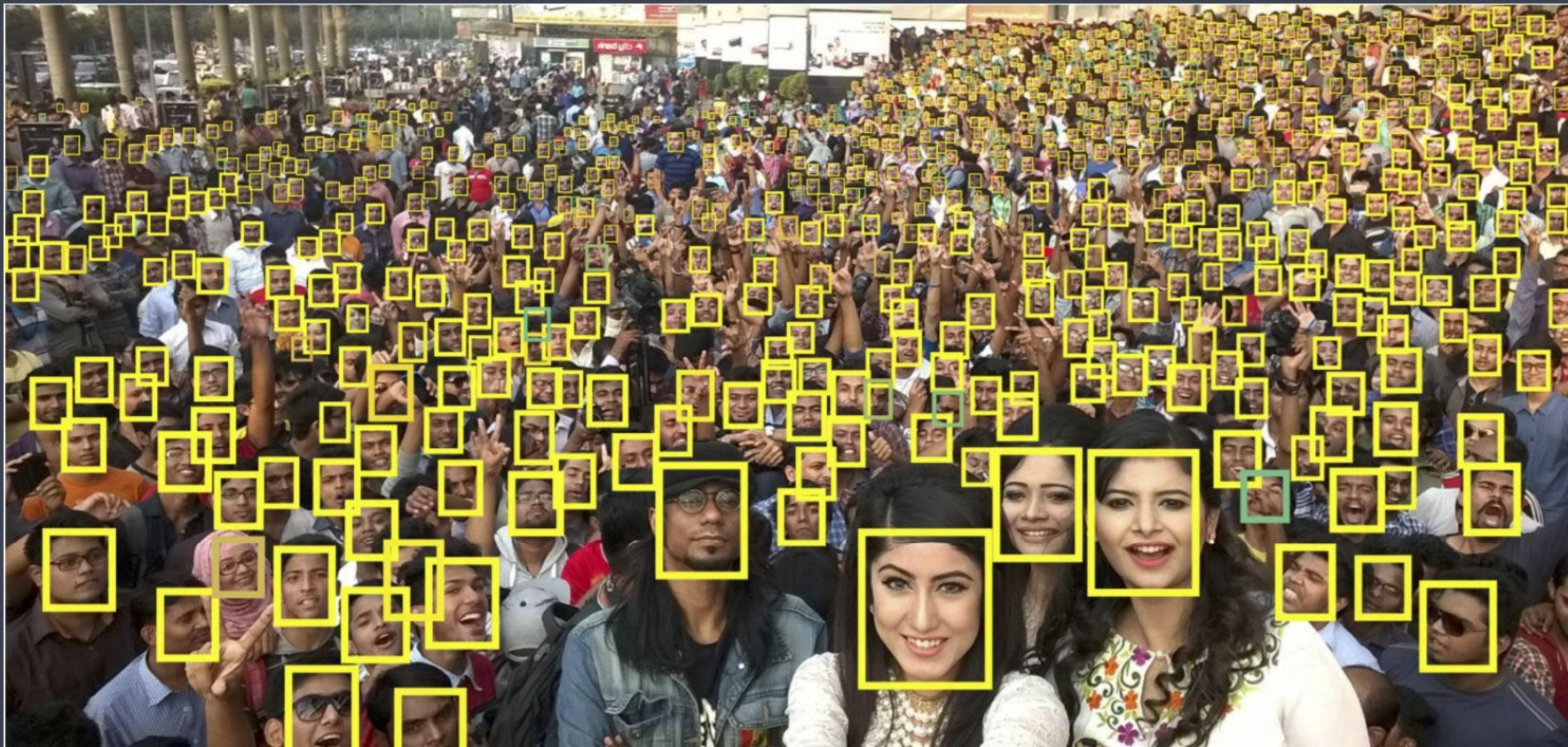
CAT, DOG, DUCK

Single object

Multiple objects

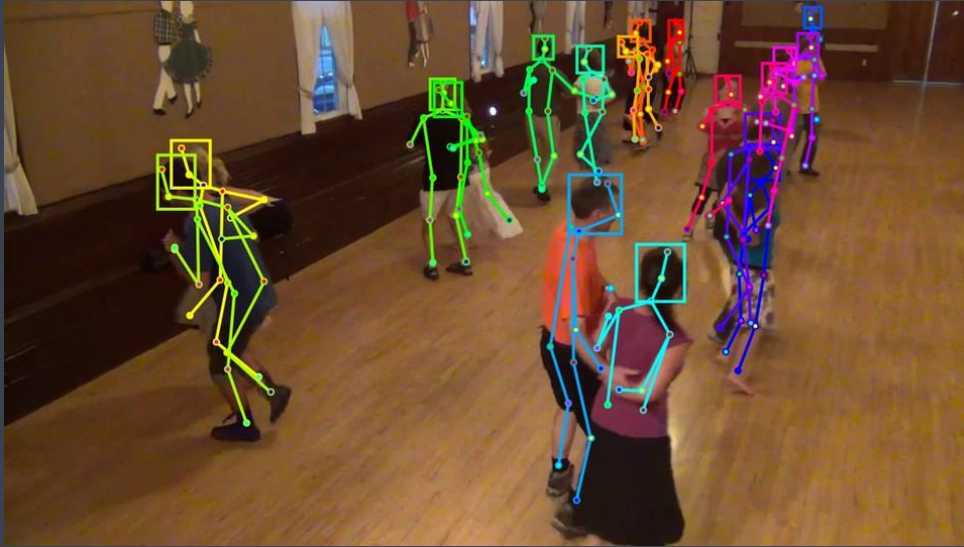
- RCNN, SPPnet, FastRCNN, FasterRCNN, Mask-RCNN
- AttentionNet, SSD, YOLO, YOLOv2, YOLOv3

Face detection

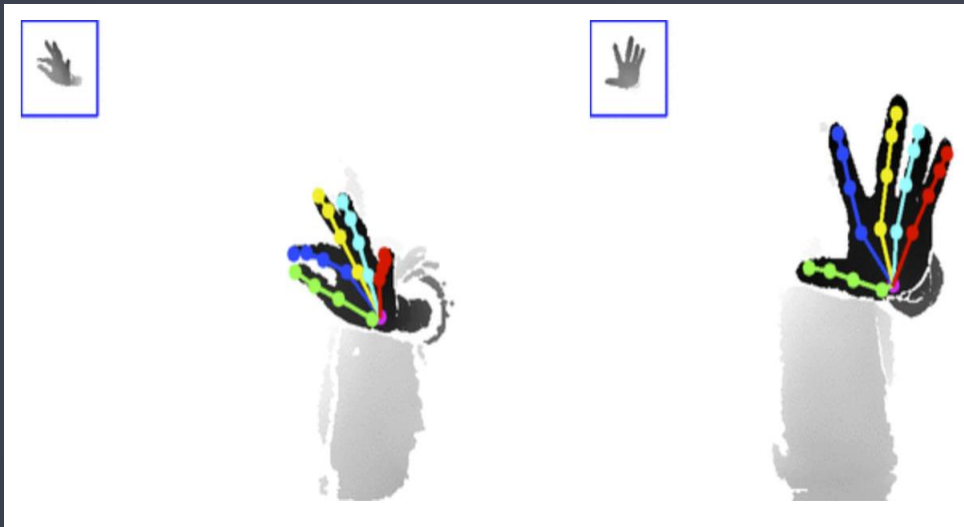


- OpenCV(Haar Cascades / HOG), Dlib(Face Recognition)
- MTCNN(Multi-Task Cascaded Convolutional Neural Network)

Pose Estimation



- Tensorflow Lite(Pose estimation)
- ml5js(poseNet)



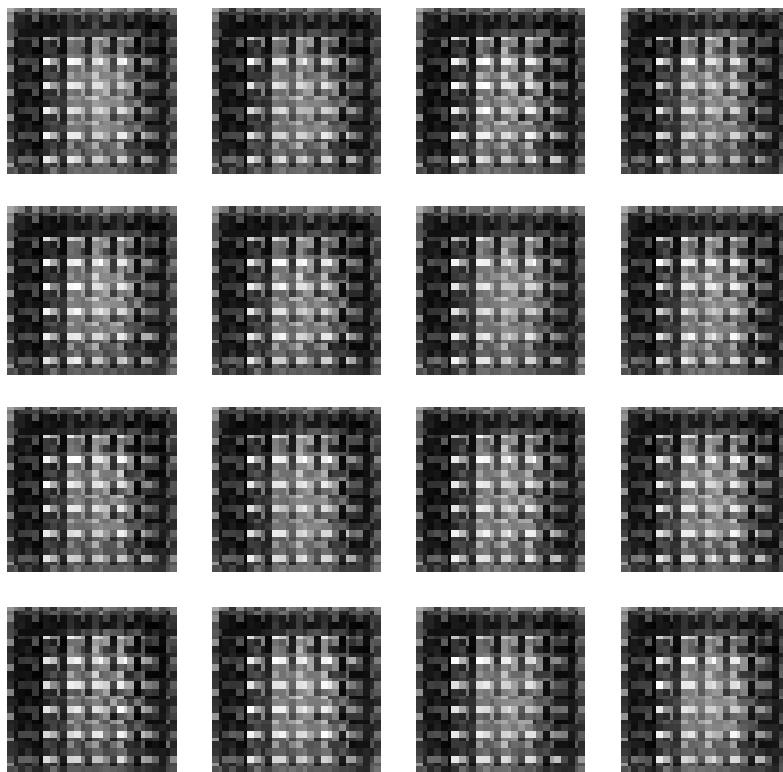
- Caffe-pose(Pose-REN)

생성모델 - Style Transfer



- Tensorflow
- Pytorch

생성모델 - DCGAN



- Tensorflow
- Pytorch

생성모델 - CycleGAN

Input Image



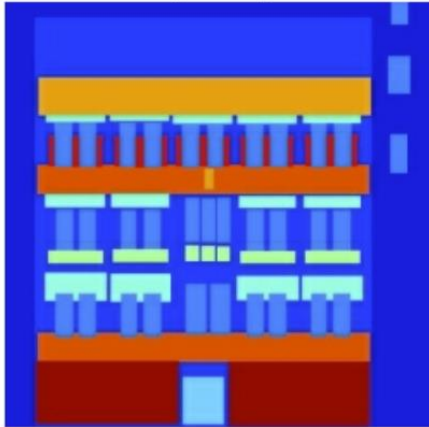
Predicted Image



- Tensorflow
- Pytorch

생성모델 - Pix2Pix

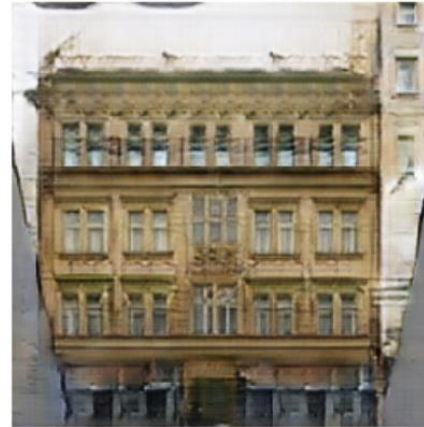
Input Image



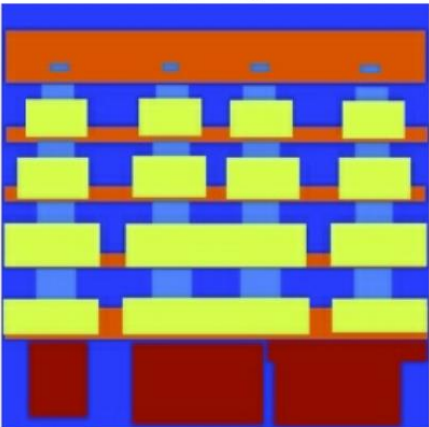
Ground Truth



Predicted Image



Input Image



Ground Truth



Predicted Image



- Tensorflow
- Pytorch