합성곱 신경망(CNN)

2021

■ 심층 신경망(DNN: Deep Neural Network)

❖ 딥러닝 학습의 일반적 절차

- 적절한 네트워크
  - 구조
  - 비선형성 획득 방법: 활성 함수
- 그래디언트 체크
- 학습 파라메터 초기화
- 학습 파라메터 최적화
- 과적합 방지
- 기타
  - 학습률 감소

■ MNIST 손글씨 데이터

- 미국 국립표준기술원(NIST)이 고등학생과 인구조사국 직원 등이 쓴 손글씨를 이용해 만든 데이터
- 70,000개의 글자 이미지에 각각 0부터 9까지 이름표를 붙인 데이터셋

■ MNIST 손글씨 데이터

❖ 데이터 전처리

- MNIST 데이터 불러오기

```
from tensorflow.keras.datasets import mnist

(X_train, Y_class_train), (X_test, Y_class_test) = mnist.load_data()

print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
```

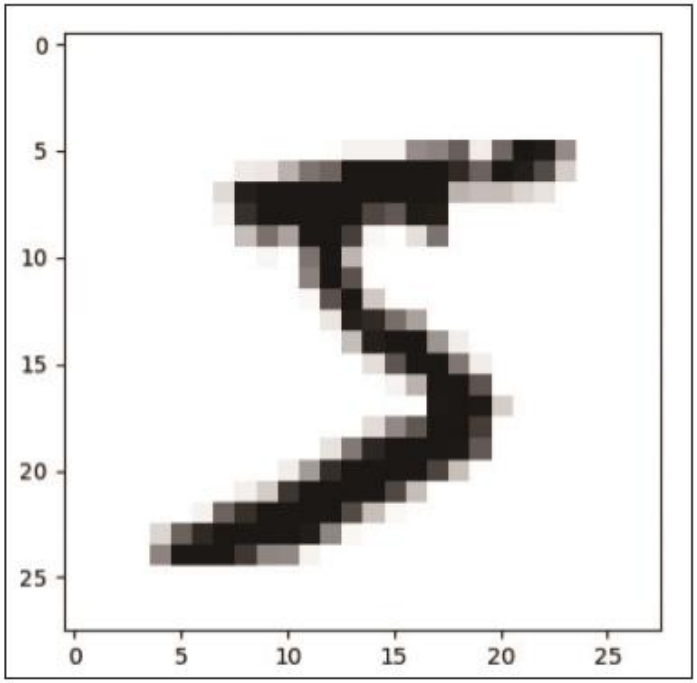
```
학습셋 이미지 수: 60000 개
테스트셋 이미지 수: 10000 개
```

■ MNIST 손글씨 데이터

❖ 데이터 전처리

- MNIST 데이터 이미지 보기

```
import matplotlib.pyplot as plt
plt.imshow(X_train[0], cmap='Greys')
plt.show()
```



■ MNIST 손글씨 데이터

❖ 데이터 전처리: 데이터 구조

- 가로 28 × 세로 28 = 총 784개의 픽셀
- 밝기: 0(흰색) ~ 255(검은색)

```
for x in X_train[0]:
    for i in x:
        sys.stdout.write('%d\t' % i)
    sys.stdout.write('\n')
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	14	1	154	253	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	253	249	64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	136	253	253	253	212	135	132	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

■ MNIST 손글씨 데이터

❖ 데이터 전처리: 정규화(Normalization)

- 케라스는 데이터가 0에서 1 사이의 값일 때 최적의 성능을 보임
- 0~255 사이의 값으로 이루어진 값을 0~1 사이의 값으로 바꿔야 함

```
X_train = X_train.astype('float64')
X_train = X_train / 255

X_test = X_test.astype('float64') / 255
```

■ MNIST 손글씨 데이터

❖ 데이터 전처리: 원 핫 인코딩(One-hot encoding)

- 0~9까지의 정수형 값을 갖는 현재 형태에서 0 또는 1로만 이루어진 벡터로 값을 수정해야 함
- 예를 들어 class가 '3'이라면, [3]을 [0,0,0,1,0,0,0,0,0,0]로 바꿔 주어야 함

```
Y_train = tf.keras.utils.to_categorical(Y_class_train, 10)
Y_test = tf.keras.utils.to_categorical(Y_class_test, 10)
```

■ MNIST 손글씨 인식을 DNN으로 해결

❖ 1차원 배열로 전환

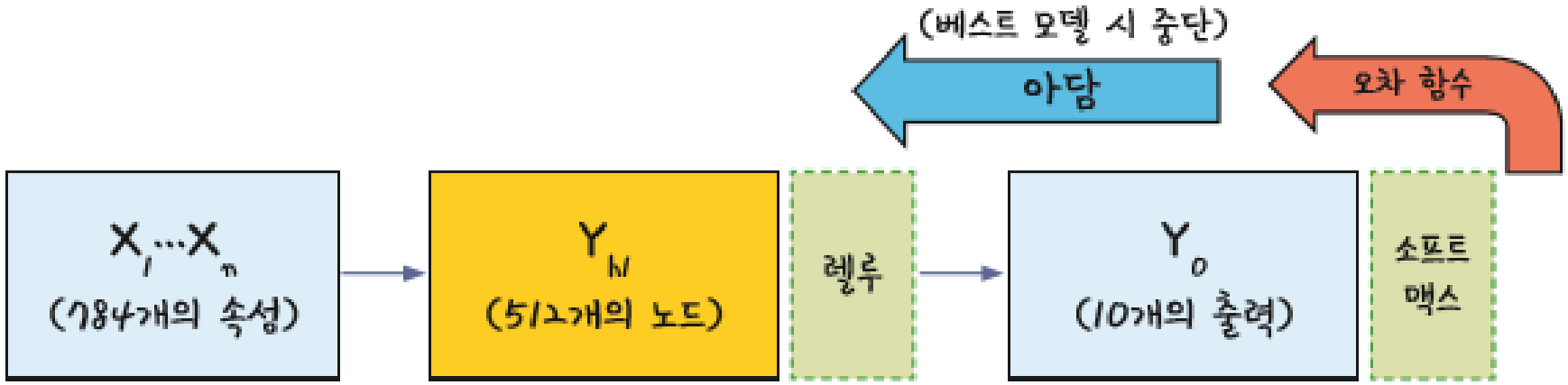
- 가로 28, 세로 28의 2차원 배열을 784개의 1차원 배열로 바꿔 주어야 함

```
X_train = X_train.reshape(X_train.shape[0], 784)
```

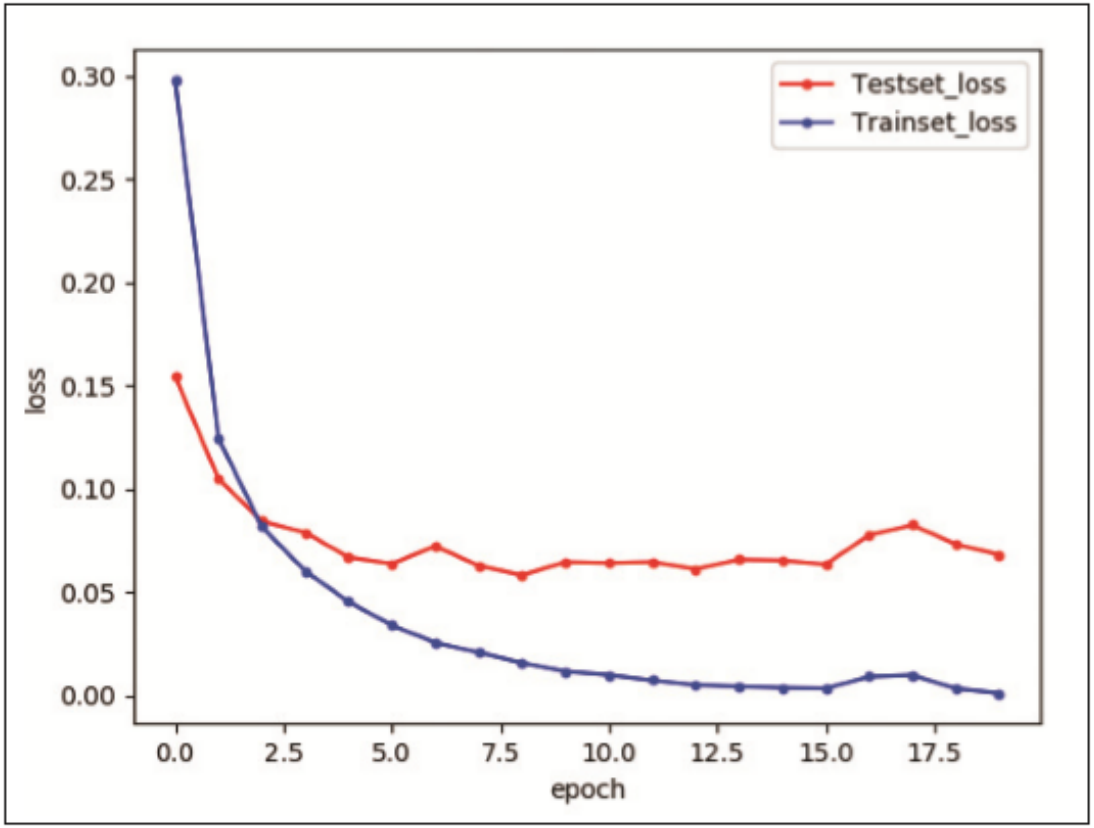
- 1차원 배열로 전환하고 정규화

```
X_train = X_train.reshape(X_train.shape[0], 784).astype('float64') / 255
```

■ MNIST 손글씨 인식을 DNN으로 해결

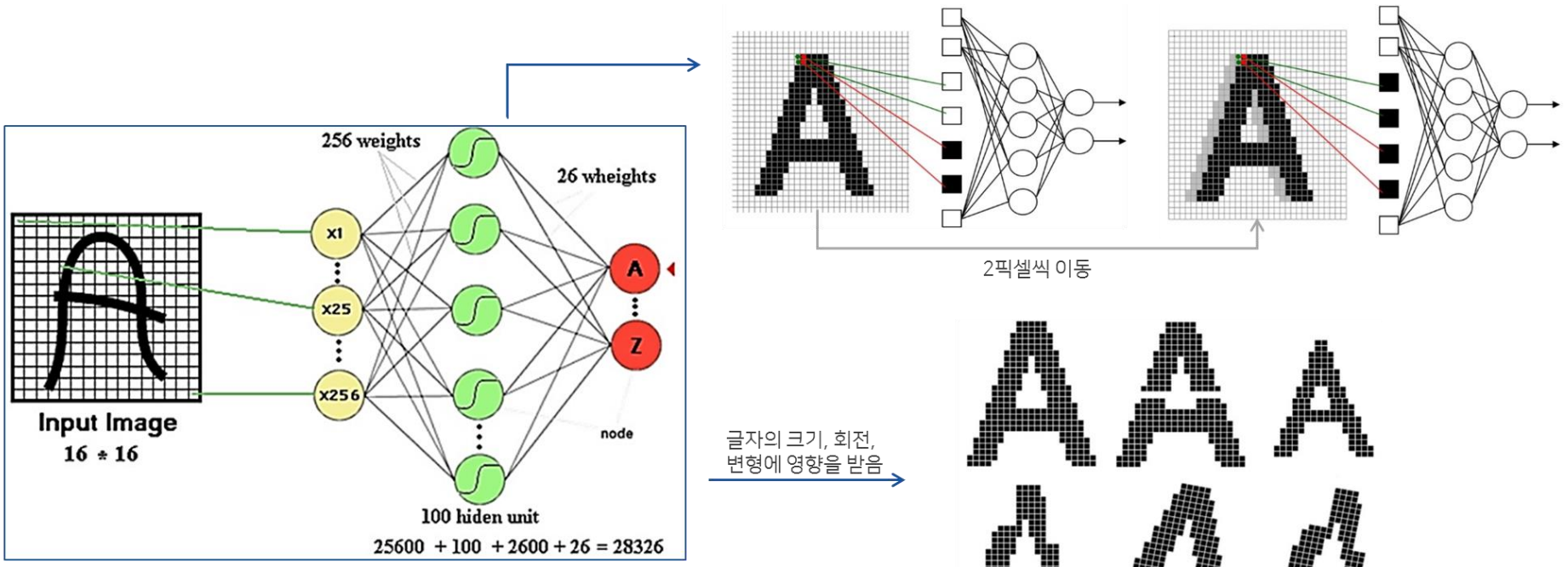


■ MNIST 손글씨 인식을 DNN으로 코딩



■ 심층 신경망(DNN)으로 구현했을 때 문제점

- 변수의 개수
- 네트워크의 크기
- 학습 시간
- 글자의 형상은 고려하지 않고, 글자의 크기, 회전, 변형에 취약함



→ *글자의 형상은 고려하지 않고, raw data를 직접 처리하기 때문에 많은 양의 학습 데이터가 필요하고, 따라서 학습 시간이 길어짐*

■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Mask(Filter, Window, Kernel)

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

주어진 이미지

x1	x0
x0	x1

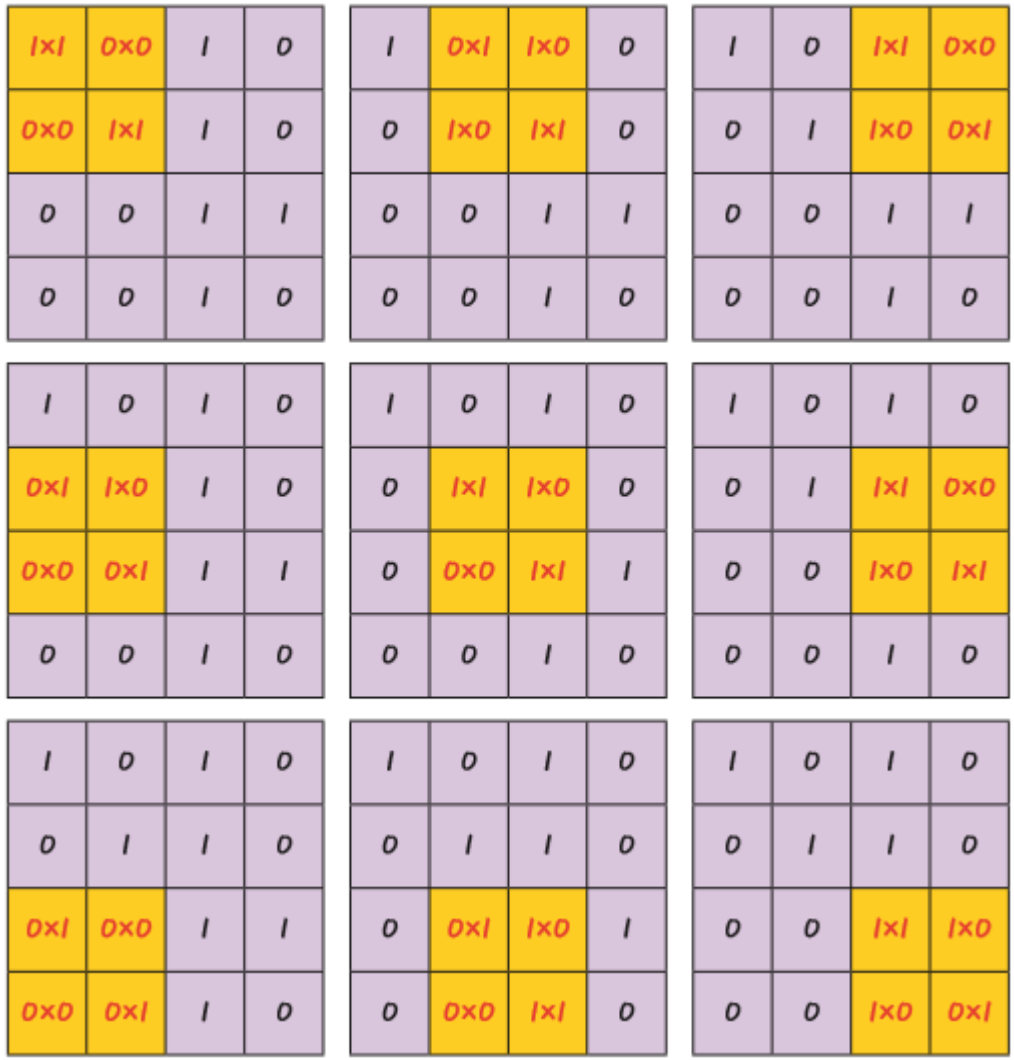
필터

■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Convolution 과정

1×1	0×0	1	0
0×0	1×1	1	0
0	0	1	1
0	0	1	0

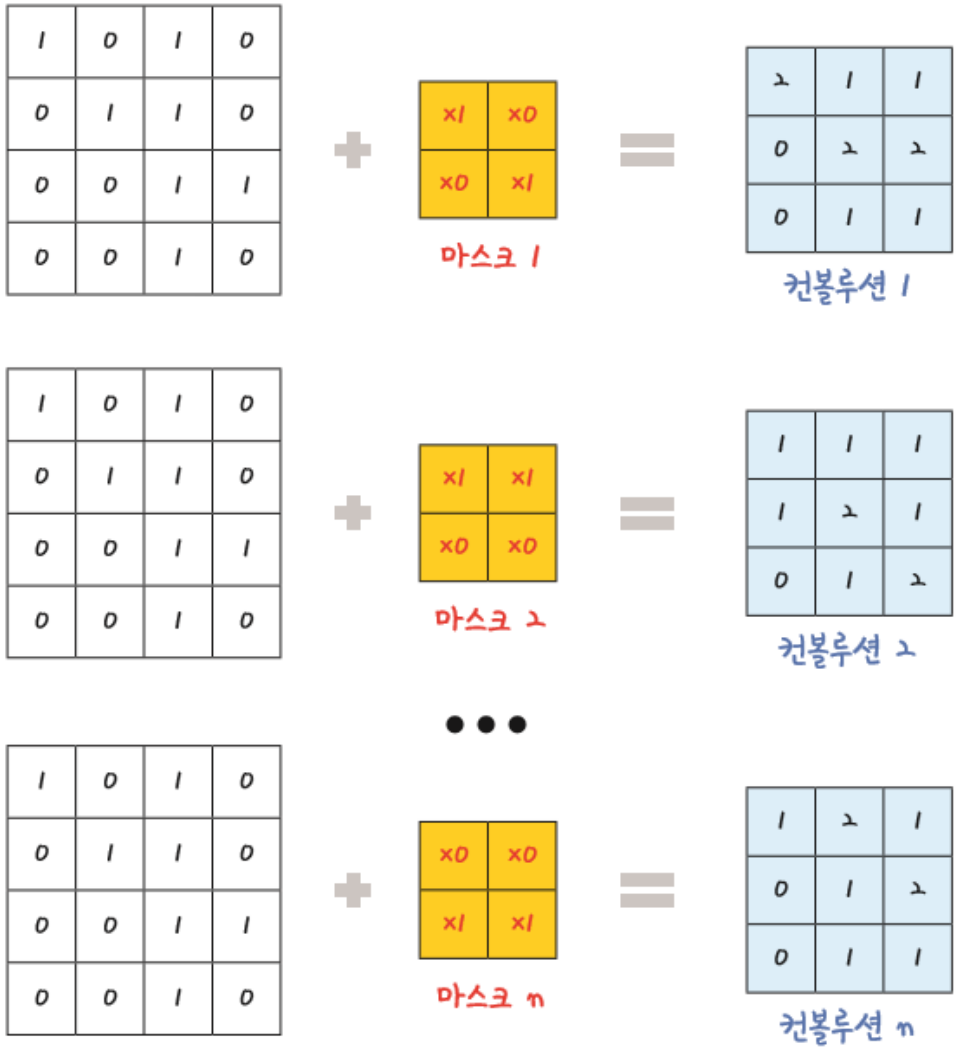
$$(1\times1)+(0\times0)+(0\times0)+(1\times1)=2$$



■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Convolution 과정

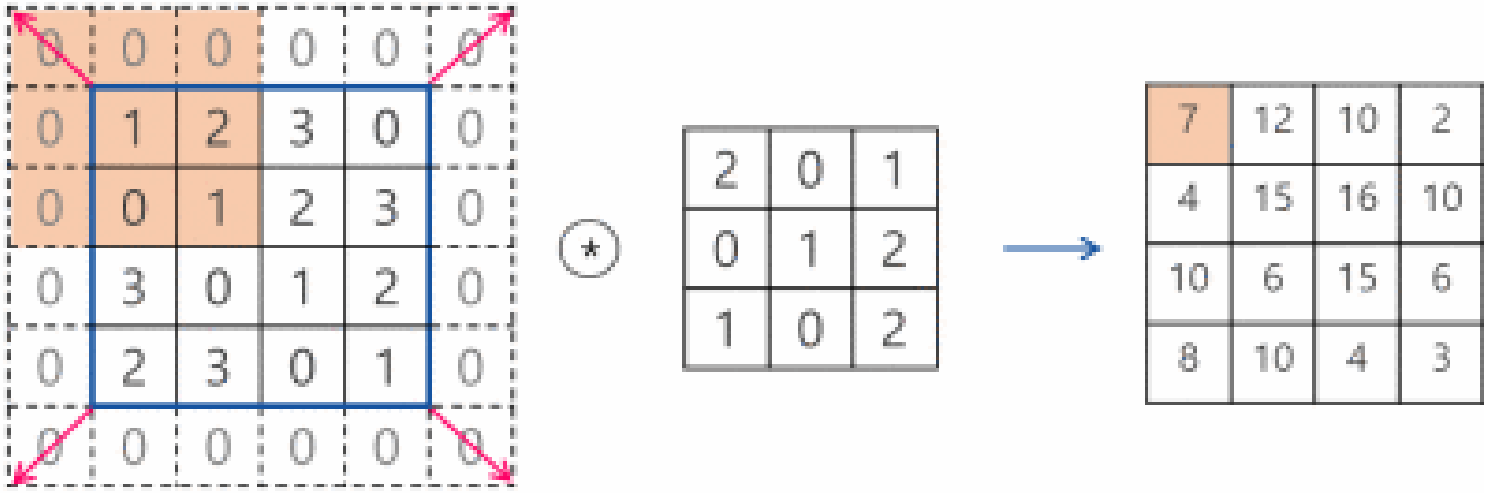
- 컨볼루션을 만들면 입력 데이터로부터 더욱 정교한 특징을 추출할 수 있음
- 이러한 마스크를 여러 개 만들 경우 여러 개의 컨볼루션이 만들어짐 (Feature Map)



■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Padding, Stride

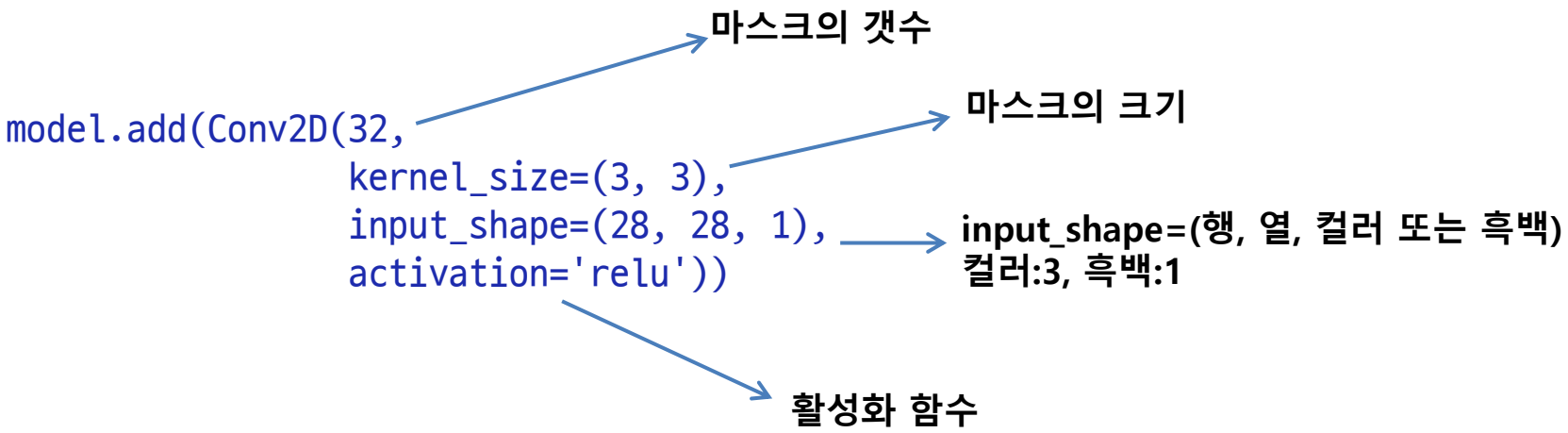
- 패딩: 합성곱 연산을 수행하기 전, 입력데이터 주변을 특정값으로 채워 늘리는 것
- 스트라이드: 입력데이터에 필터를 적용할 때 이동할 간격을 조절하는 것



■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Convolution 층 추가

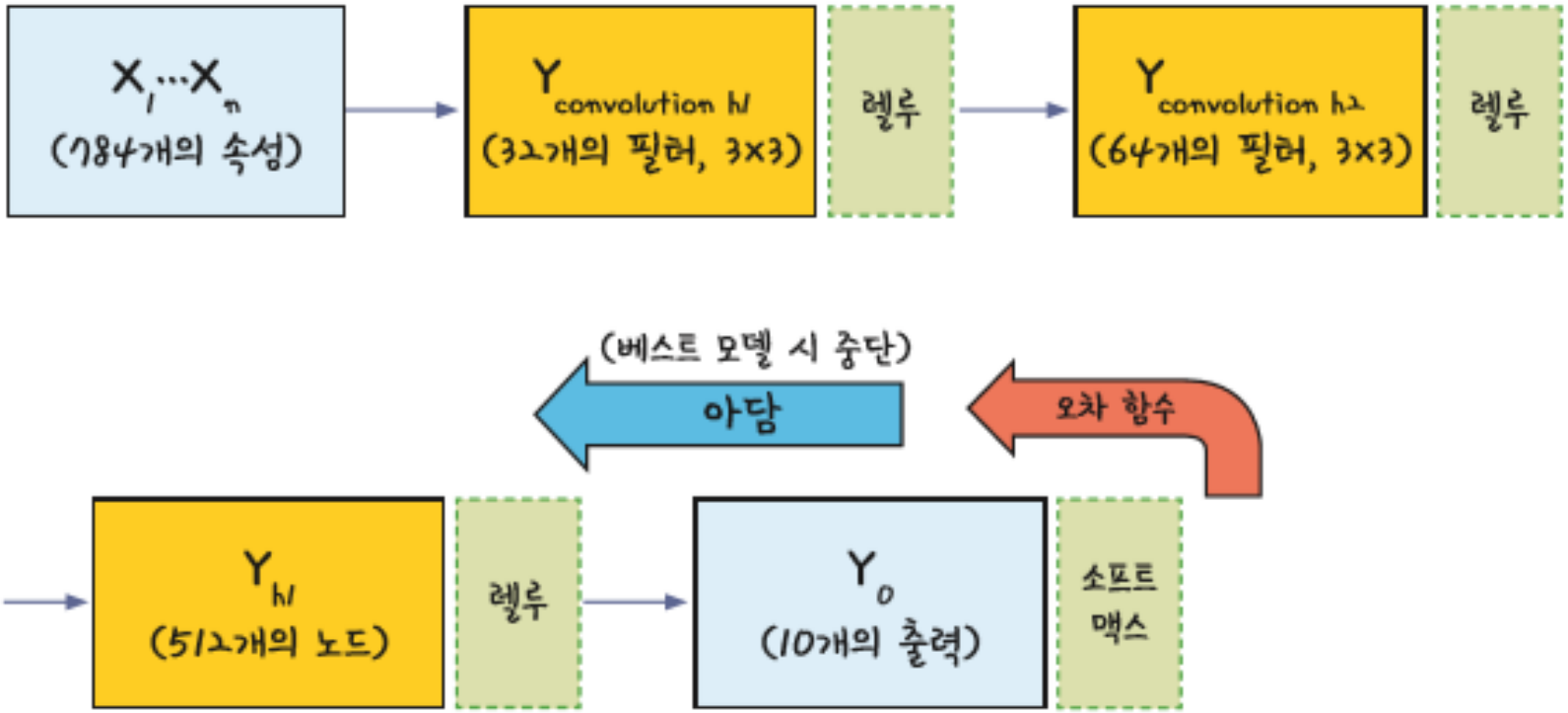
- Conv2D()



- 컨볼루션 층을 하나 더 추가

```
model.add(Conv2D(64, (3, 3), activation='relu')))
```

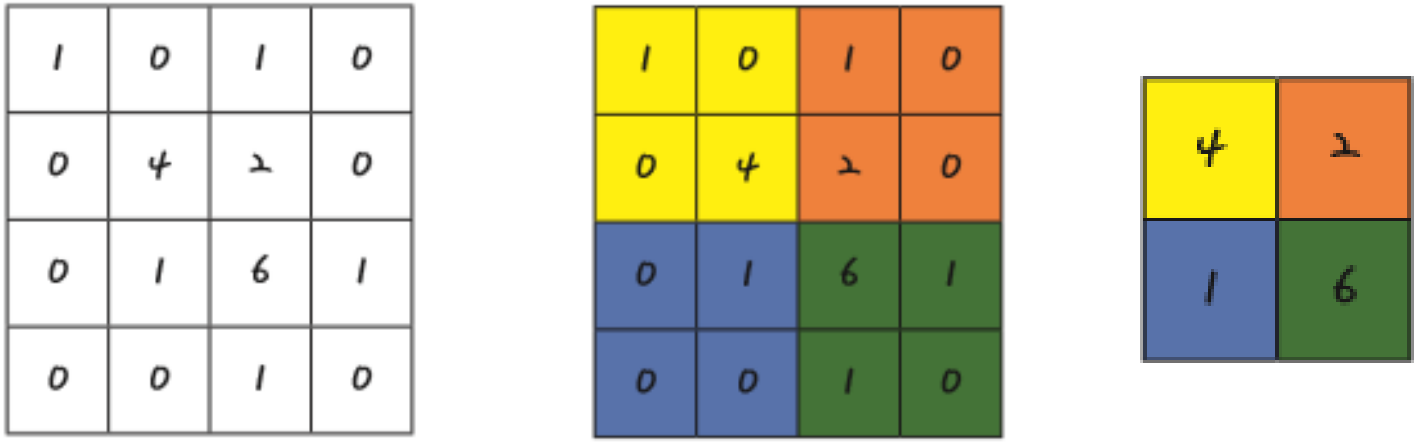
■ 컨볼루션 신경망(Convolutional Neural Network, CNN)



■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 풀링(Pooling)

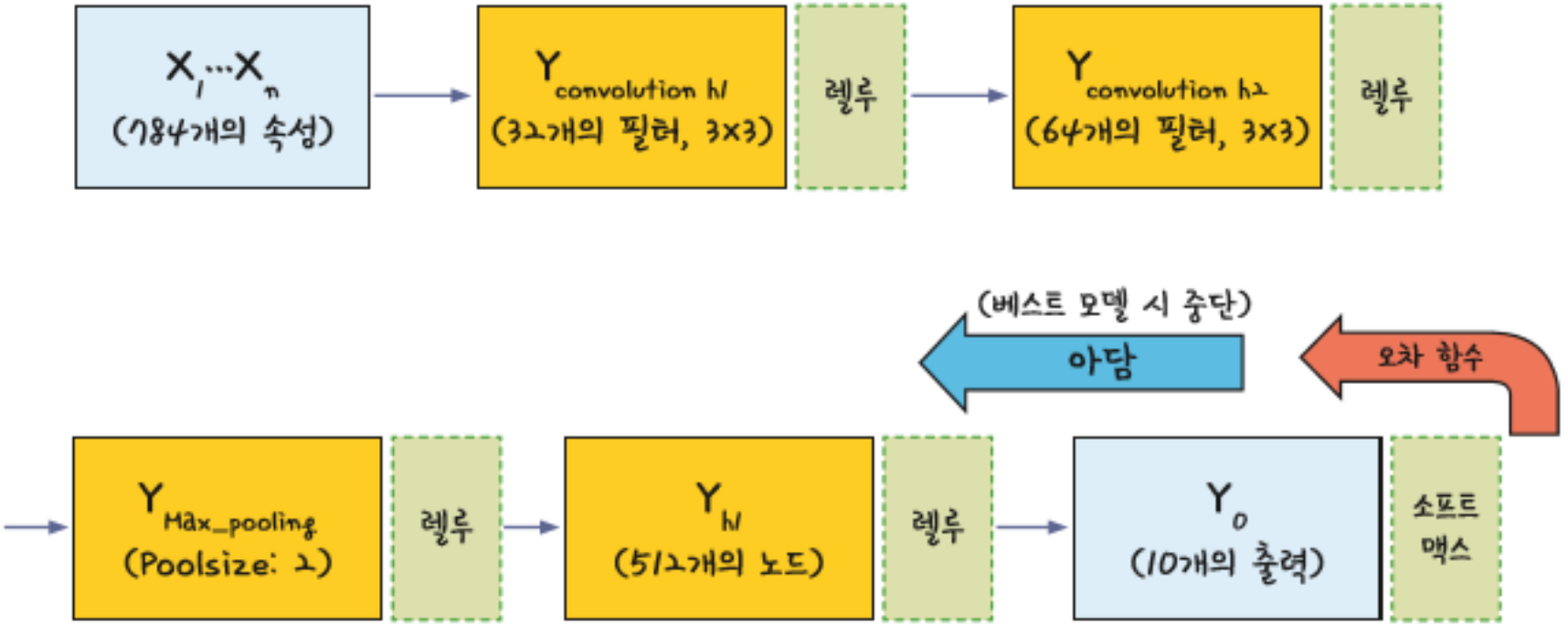
- Convolution 결과를 축소하는 것
- 풀링 기법 중 가장 많이 사용되는 방법이 맥스 풀링(max pooling)
- 맥스 풀링은 정해진 구역 안에서 가장 큰 값만 다음 층으로 넘기고 나머지는 버림



- 불필요한 정보를 간추림

```
model.add(MaxPooling2D(pool_size=2))
```

■ 컨볼루션 신경망(Convolutional Neural Network, CNN)



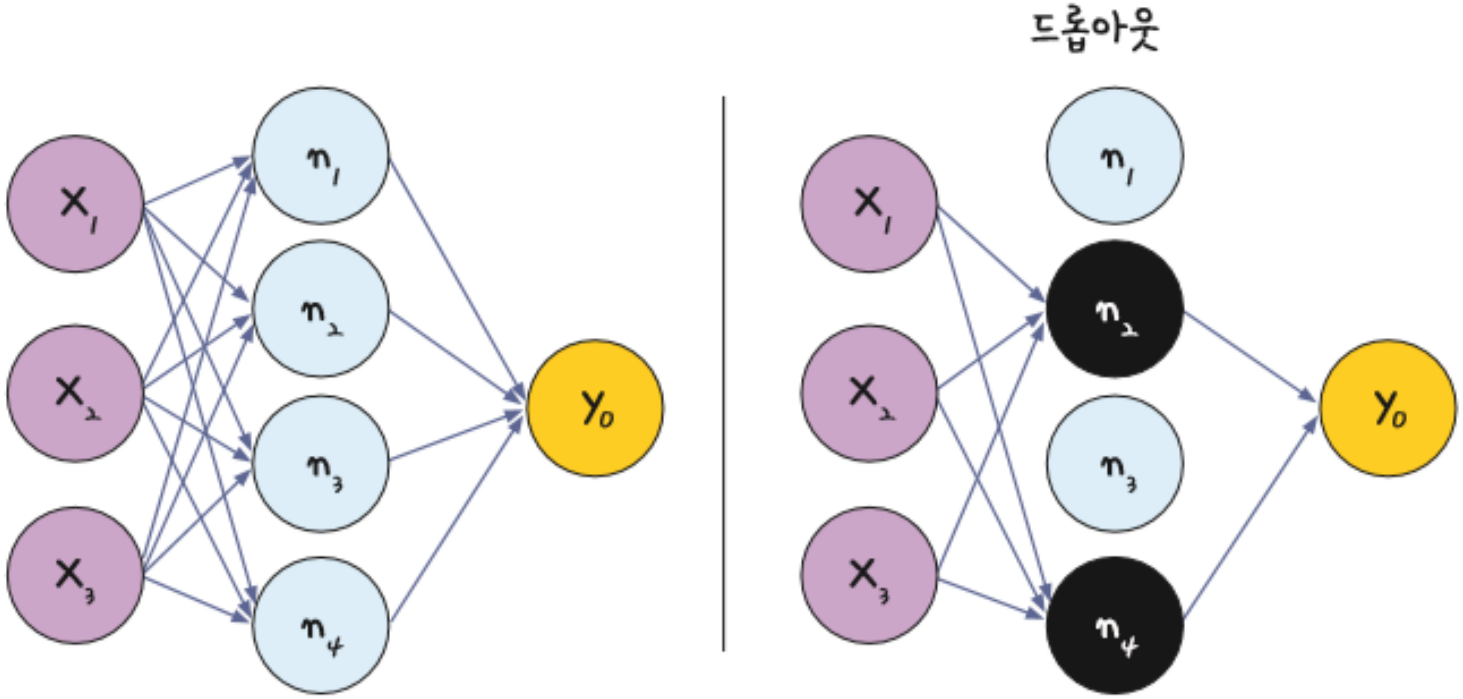
■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 드롭아웃(Drop out) , 플래튼(Flatten)

- 노드가 많아지거나 층이 많아진다고 해서 학습이 무조건 좋아지는 것이 아니다
  → 과적합 발생
- 과접합을 피하는 간단하지만 효과가 큰 기법이 바로 드롭아웃(drop out) 기법
- 드롭아웃은 은닉층에 배치된 노드 중 일부를 임의로 꺼주는 것

■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 드롭아웃(Drop out) , 플래튼(Flatten)



- 25%의 노드를 끄려면 다음과 같이 코드를 작성

```
model.out(Dropout(0.25))
```

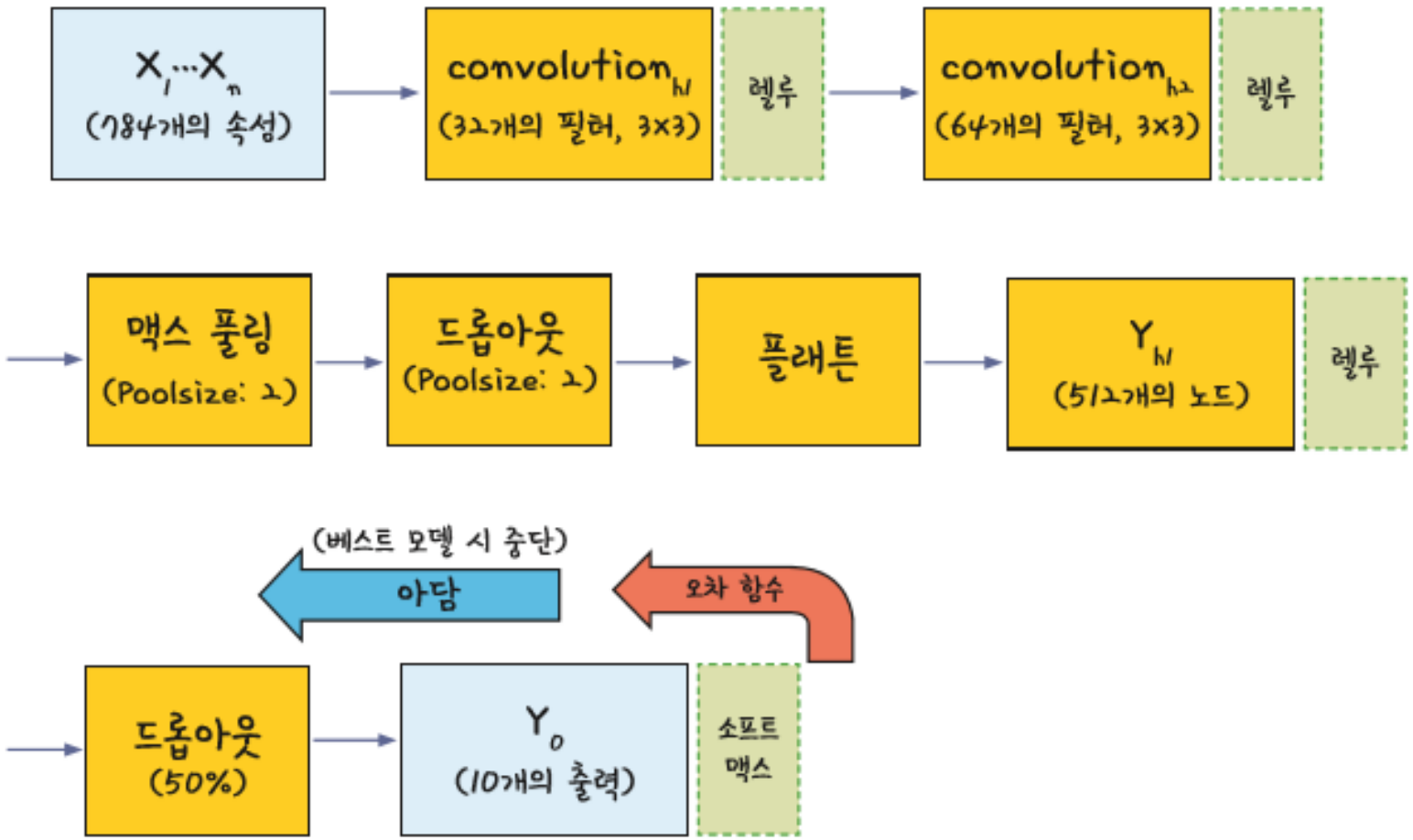
■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 드롭아웃(Drop out) , 플래튼(Flatten)

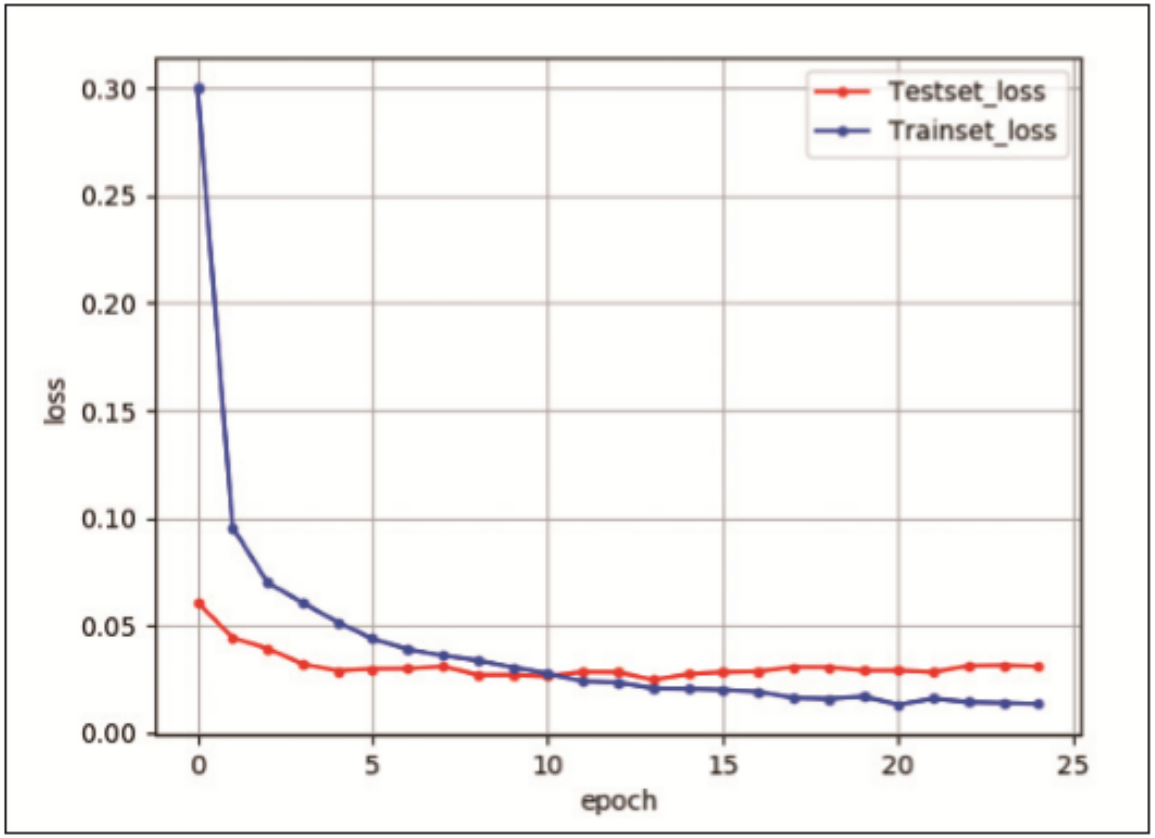
- 콘볼루션, 맥스풀링, 드롭아웃 층을 거친 후 기본 층에 연결
  - 콘볼루션, 맥스풀링: 2차원
  - 기본 층: 1차원
- 2차원 → 1차원 변환

```
model.add(Flatten())
```

■ 컨볼루션 신경망(Convolutional Neural Network, CNN)



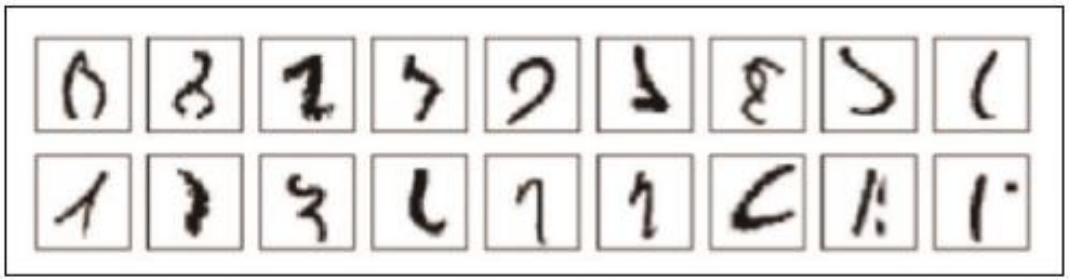
■ 코딩으로 확인하는 MNIST 이미지 인식



■ 코딩으로 확인하는 MNIST 이미지 인식

❖ 결과 리뷰

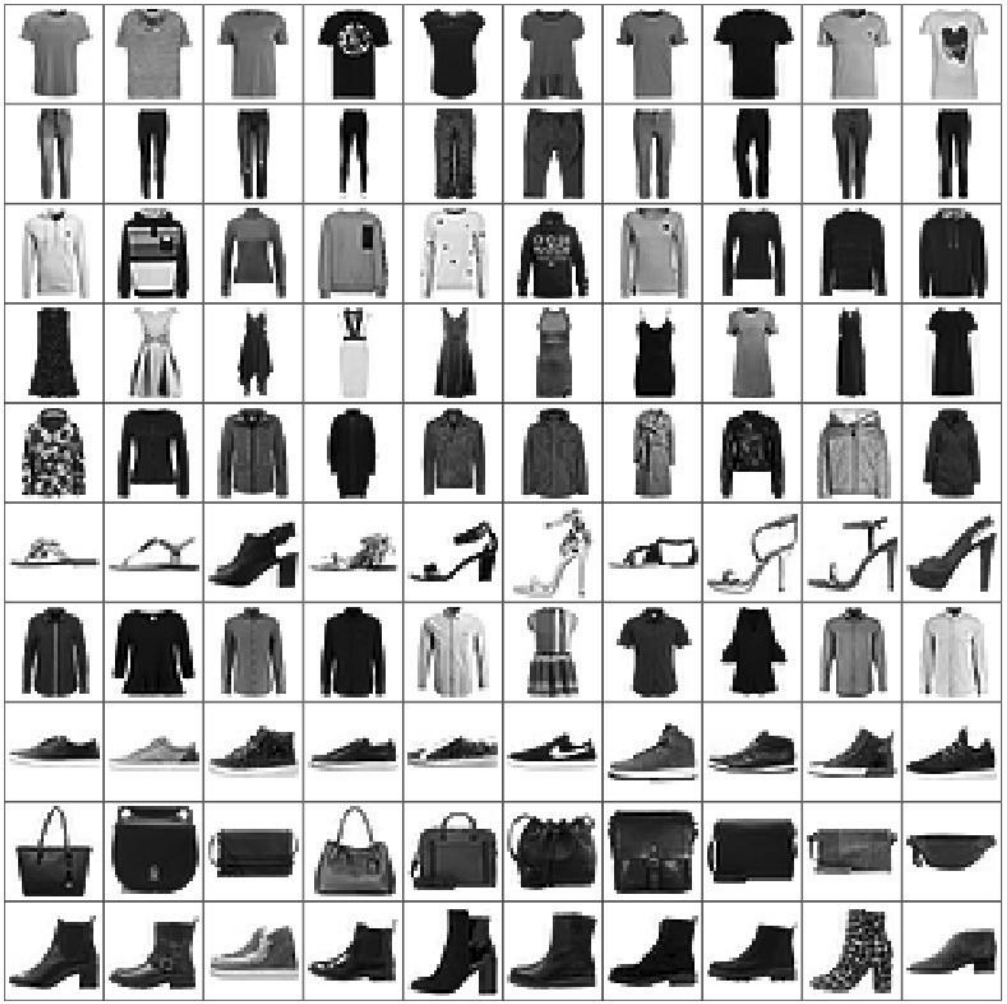
- 0.9901, 즉 99.01%의 정확도
- 심층 신경망 코드에서는 정확도가 97.86%
- 100% 다 맞히지 못한 이유는 데이터 안에 다음과 같이 확인할 수 없는 글씨가 들어있었기 때문



■ Fashion MNIST

❖ 데이터 셋

- 손글씨 데이터셋 대용으로 사용 가능



■ 개, 고양이 구분

❖ Kaggle site

- 2010년 설립된 빅데이터 솔루션 대회 플랫폼 회사
- 2017년 Google이 인수 (ZDNet 기사)
- 기업 및 단체에서 Prize를 걸고 데이터와 해결 과제를 등록하면, 데이터 사이언티스트들이 이를 해결하기 위해 모델을 개발하고 경쟁하게 되는 시스템

❖ Kaggle 구성 요소

- Overview: 문제에 대한 간략한 소개와 문제 정의
- Dataset: 예측 모델을 만들기 위해 필요한 데이터셋 및 field에 대한 설명
- Kernels: 다른 사람들이 어떤 모델을 써서 구현을 했는지 힌트를 얻을 수 있고, 또한 내가 구현한 모델이 과연 올바른지에 관해서 코멘트를 주고받을 수 있음
- Discussion: 게시판 역할
- Leaderboard: 모델 예측 정확도 랭킹

■ 개, 고양이 구분

❖ 데이터 셋

- 훈련 셋: 개, 고양이 사진 각각 12,500개, 총 25,000개
- 테스트 셋: 개, 고양이 사진 합쳐서 12,500개

```
datasets
└── dogs-vs-cats
    ├── train
    │   ├── cat.0.jpg
    │   │    ...
    │   ├── cat.12499.jpg
    │   ├── dog.0.jpg
    │   │    ...
    │   └── dog.12499.jpg
    └── test1
        ├── 1.jpg
        │    ...
        └── 12500.jpg
```

■ 개, 고양이 구분

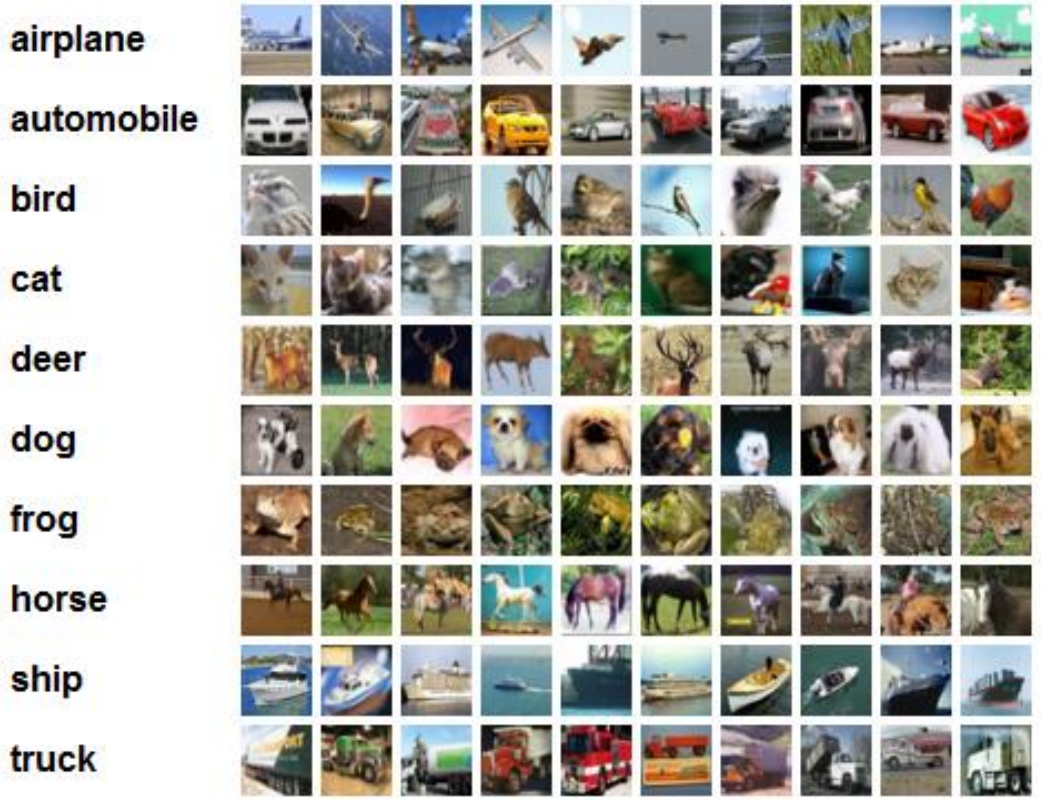
❖ 데이터 셋

- 훈련 셋: 개, 고양이 사진 각각 12,500개, 총 25,000개
- 테스트 셋: 개, 고양이 사진 합쳐서 12,500개

■ Cifar 10

❖ 데이터 셋

- 32 x 32 크기의 컬러 이미지
- 훈련 셋: 10가지 종류의 50,000개
- 테스트 셋: 10가지 종류의 10,000개



4. 데이터 부풀리기

❖ 데이터 부풀리기(Data Augmentation)

- 원본 이미지에 인위적인 변화를 주어
- 변화된 이미지는 충분히 학습에 활용될 수 있는 데이터가 됨
- 적당한 힘으로 학습 면적을 아주 조금 골고루 넓히자는 의미
- 대부분의 경우 인식의 정확도가 올라감

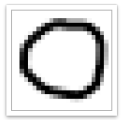
❖ ImageDataGenerator 클래스

- Keras에서 제공
- 파라메터는 객체 생성시 전달
- *flow_from_directory* 메소드를 활용하면 폴더 형태로된 데이터 구조를 바로 가져와서 사용할 수 있음

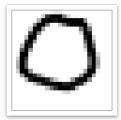
❖ ImageDataGenerator 클래스 사용 사례

```
datagen = ImageDataGenerator(
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of dataset
    samplewise_std_normalization=False,  # divide each input by its std
    zca_whitening=False,  # apply ZCA whitening
    zca_epsilon=1e-06,  # epsilon for ZCA whitening
    rotation_range=0,  # randomly rotate images in the range (deg 0 to 180)
    width_shift_range=0.1,  # randomly shift images horizontally
    height_shift_range=0.1,  # randomly shift images vertically
    shear_range=0.,  # set range for random shear
    zoom_range=0.,  # set range for random zoom
    channel_shift_range=0.,  # set range for random channel shifts
    fill_mode='nearest',  # set mode for filling points outside the input boundaries
    cval=0.,  # value used for fill_mode = "constant"
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False,  # randomly flip images
    rescale=None,  # set rescaling factor (applied before any other transformation)
    preprocessing_function=None,  # set function that will be applied on each input
    data_format=None,  # image data format, either "channels_first" or "channels_last"
    validation_split=0.0  # fraction of images reserved for validation
)
```

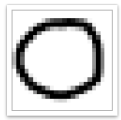
❖ 훈련 셋

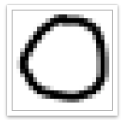

circle001.png

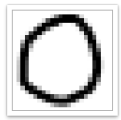

circle002.png

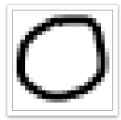

circle003.png

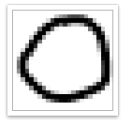

circle004.png

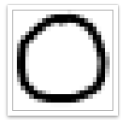

circle005.png

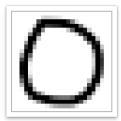

circle006.png

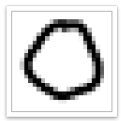

circle007.png


circle008.png

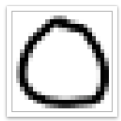

circle009.png

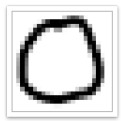

circle010.png

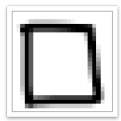

circle011.png

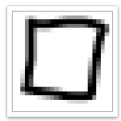

circle012.png

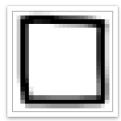

circle013.png

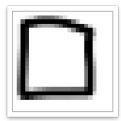

circle014.png

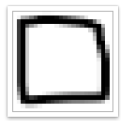

circle015.png

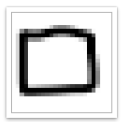

rectangle001.png

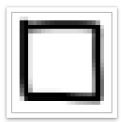

rectangle002.png

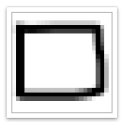

rectangle003.png

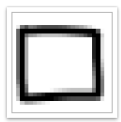

rectangle004.png

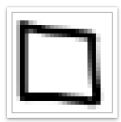

rectangle005.png

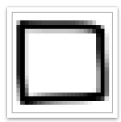

rectangle006.png

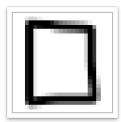

rectangle007.png

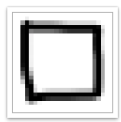

rectangle008.png

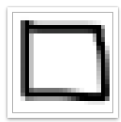

rectangle009.png

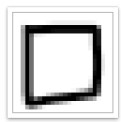

rectangle010.png


rectangle011.png


rectangle012.png


rectangle013.png


rectangle014.png


rectangle015.png


triangle001.png


triangle002.png


triangle003.png


triangle004.png


triangle005.png


triangle006.png


triangle007.png


triangle008.png


triangle009.png


triangle010.png

triangle011.png

triangle012.png

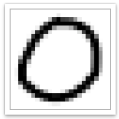
triangle013.png

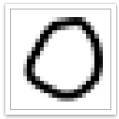
triangle014.png

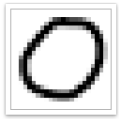
triangle015.png

❖ 테스트 셋

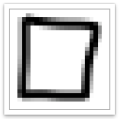

circle016.png

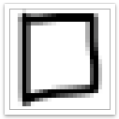

circle017.png

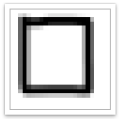

circle018.png

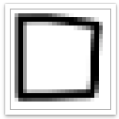

circle019.png

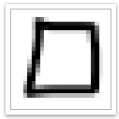

circle020.png


rectangle016.png


rectangle017.png


rectangle018.png


rectangle019.png


rectangle020.png


triangle016.png


triangle017.png


triangle018.png


triangle019.png


triangle020.png

❖ 도전 테스트 셋

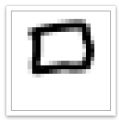

circle021.png


circle022.png


circle023.png


circle024.png


circle025.png


rectangle021.png


rectangle022.png


rectangle023.png


rectangle024.png


rectangle025.png


triangle021.png

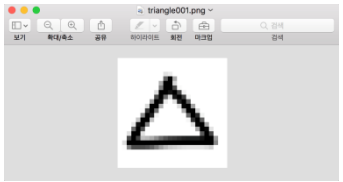

triangle022.png


triangle023.png


triangle024.png


triangle025.png

- 원본 이미지:



- rotation_range = 90, 지정된 각도 범위(90도)내에서 임의로 원본이미지를 회전

- width_shift_range = 0.1, 지정된 수평방향 이동 범위(10%)내에서 임의로 원본이미지를 이동

- height_shift_range = 0.1, 지정된 수직방향 이동 범위(10%)내에서 임의로 원본이미지를 이동

4. 데이터 부풀리기

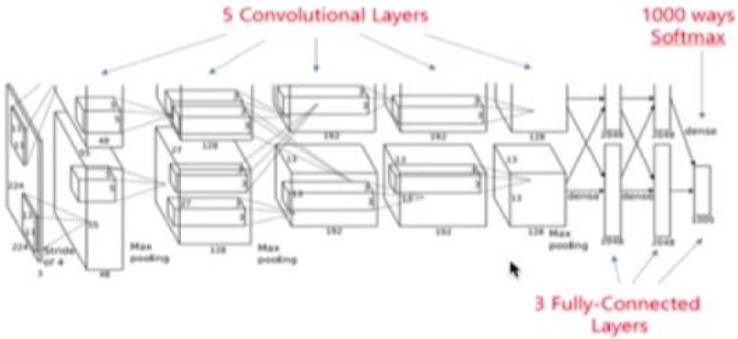
- zoom_range = 0.3, 지정된 확대/축소 범위(0.7 ~ 1.3배)내에서 임의로 원본이미지를 확대/축소

- horizontal_flip = True, 수평방향으로 뒤집기

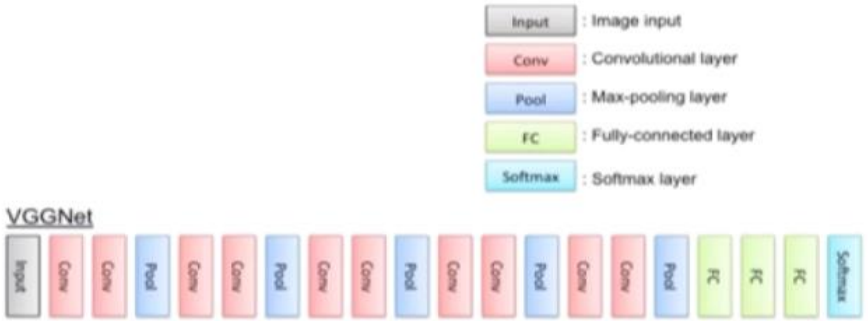
- vertical_flip = True, 수직방향으로 뒤집기

CNN Architectures

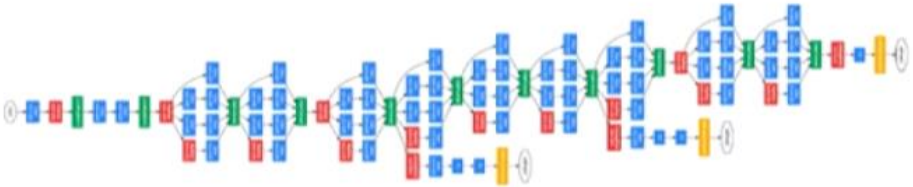
AlexNet



VGG



GoogLeNet



ResNet

■ AlexNet

❖ 개요

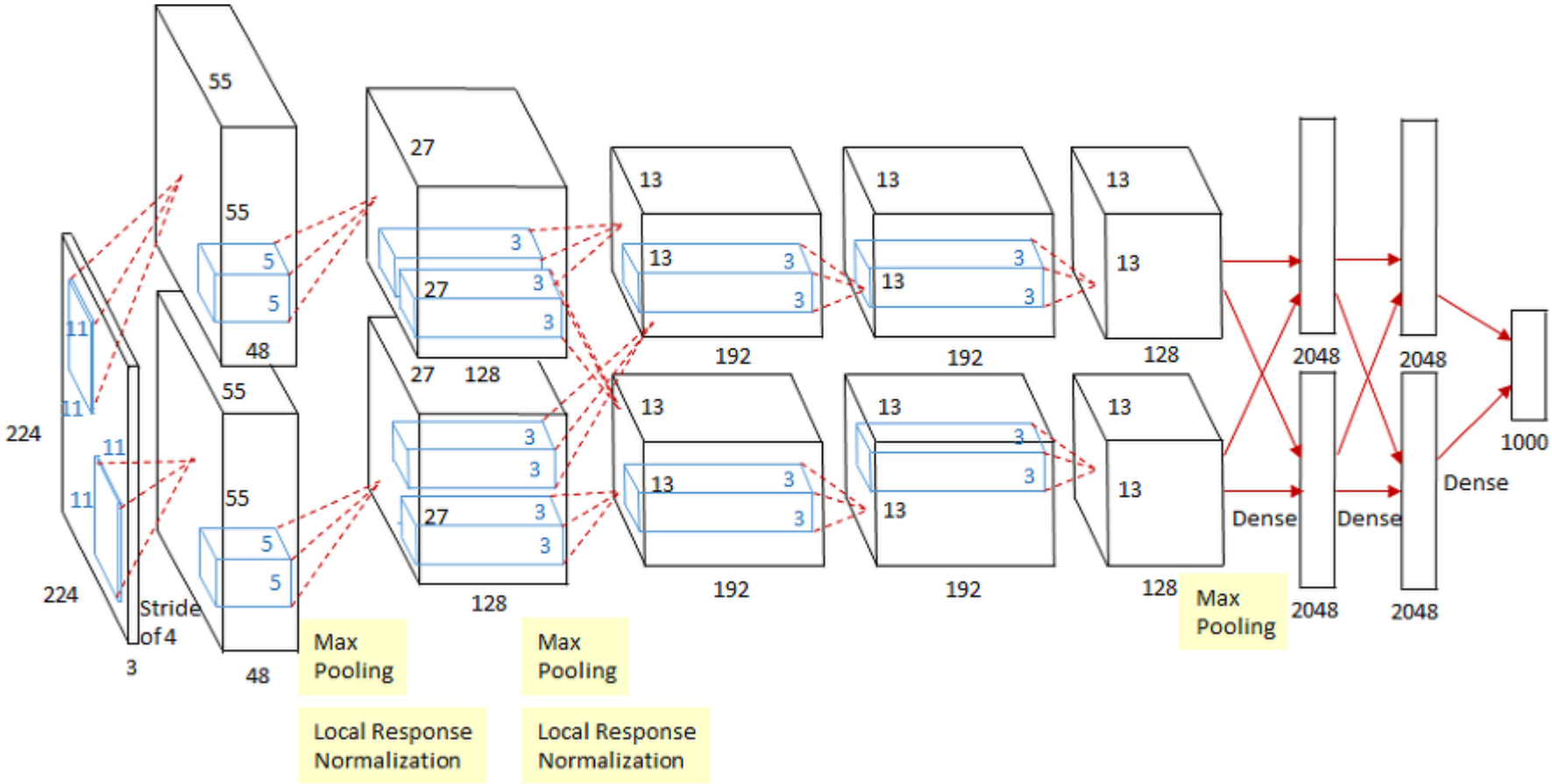
- ImageNet에서 주관하는 ILSVRC (Large Scale Visual Recognition Competition) 대회에서, 2012년 제프리 힌튼 교수팀의 AlexNet이 top 5 test error(5개의 예측값 중에 정답이 없는 경우) 기준 15.4%를 기록해 2위(26.2%)를 큰 폭으로 이기고 1위를 차지함.

- 이 대회는 1000개의 클래스를 가진 120만장의 이미지를 학습하고 15만장의 이미지로 테스트하여 정답률을 겨루는 대회

- AlexNet의 등장은 딥러닝, 특히 CNN이 본격적으로 주목받게 되는 계기가 되었고 여기서 소개된 ReLU, Dropout 등은 지금도 표준으로 사용되고 있음.

■ AlexNet

❖ 구조

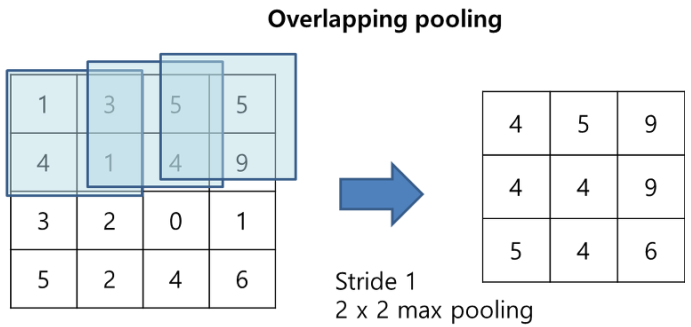

224
224
3
11
11
11
11
Stride of 4
55
55
48
48
5
5
5
5
55
55
27
27
27
27
128
128
3
3
3
3
3
3
3
3
Max Pooling
Max Pooling
Local Response Normalization
Local Response Normalization
13
13
13
13
13
13
13
13
13
13
13
13
192
192
192
192
128
128
3
3
3
3
3
3
3
3
3
3
3
3
Max Pooling
2048
2048
2048
2048
Dense
Dense
Dense
1000

■ AlexNet

❖ 특징

- 5개의 컨볼루션 레이어, 3개의 Fully Connected 레이어로 구성
- 2개의 GPU로 병렬연산 수행

- ReLU 활성화 함수 사용
- Dropout 사용
- Max pooling, Overlapping pooling
- LRN(Local Response Normalization)
- Data Augmentation
- Stochastic Gradient Descent



■ VGGNet

❖ 개요

- 2014년 ILSVRC 대회에서, 2등을 한 모델

- GoogLeNet에 밀려 2위를 했지만, 훨씬 간단한 구조로 이해와 변형이 쉽다는 장점이 있어 많이 응용되는 모델

- 깊이에 따른 변화를 비교하기 위해, 3x3의 작은 필터 크기를 사용했고, 모델 깊이와 구조에 변화를 주어 실험. (6가지 모델)

■ VGGNet

❖ 구조

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					

■ VGGNet

❖ 특징

- Small filters, Deeper networks

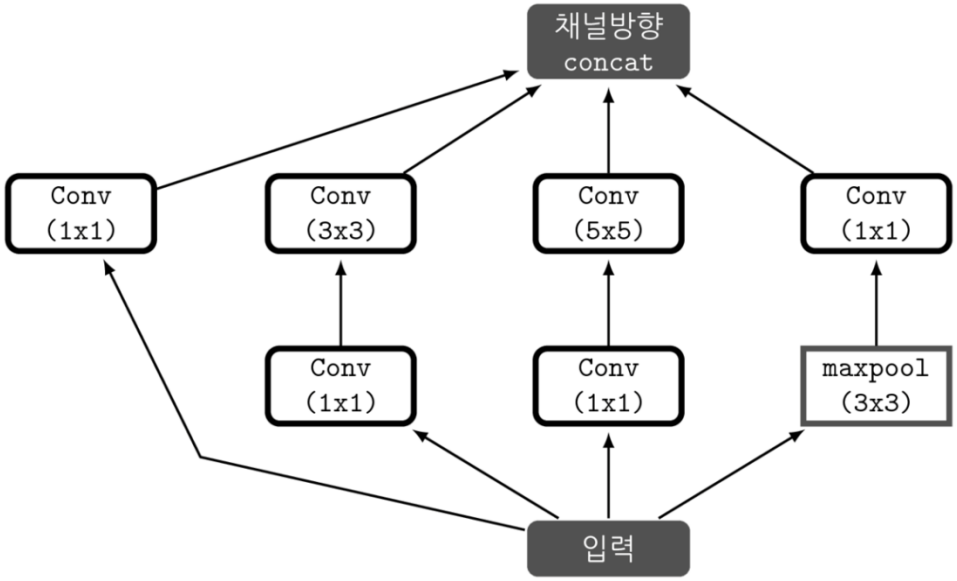
- 8개의 layer를 가지는 AlexNet에서 16~19개의 layer를 가지는 VGGNet으로 발전
- 3x3의 크기를 가지는 filter를 사용
- stride=1, padding=1 인 convolution layer
- 2x2 max pooling with stride=2인 pooling layer

- 3x3 을 깊게 쌓게 되면, 우선 비선형성을 더 많이 반영할 수 있으며, 실제로 필요한 parameter 수도 적게된다는 장점이 있음

■ GoogLeNet (Inception)

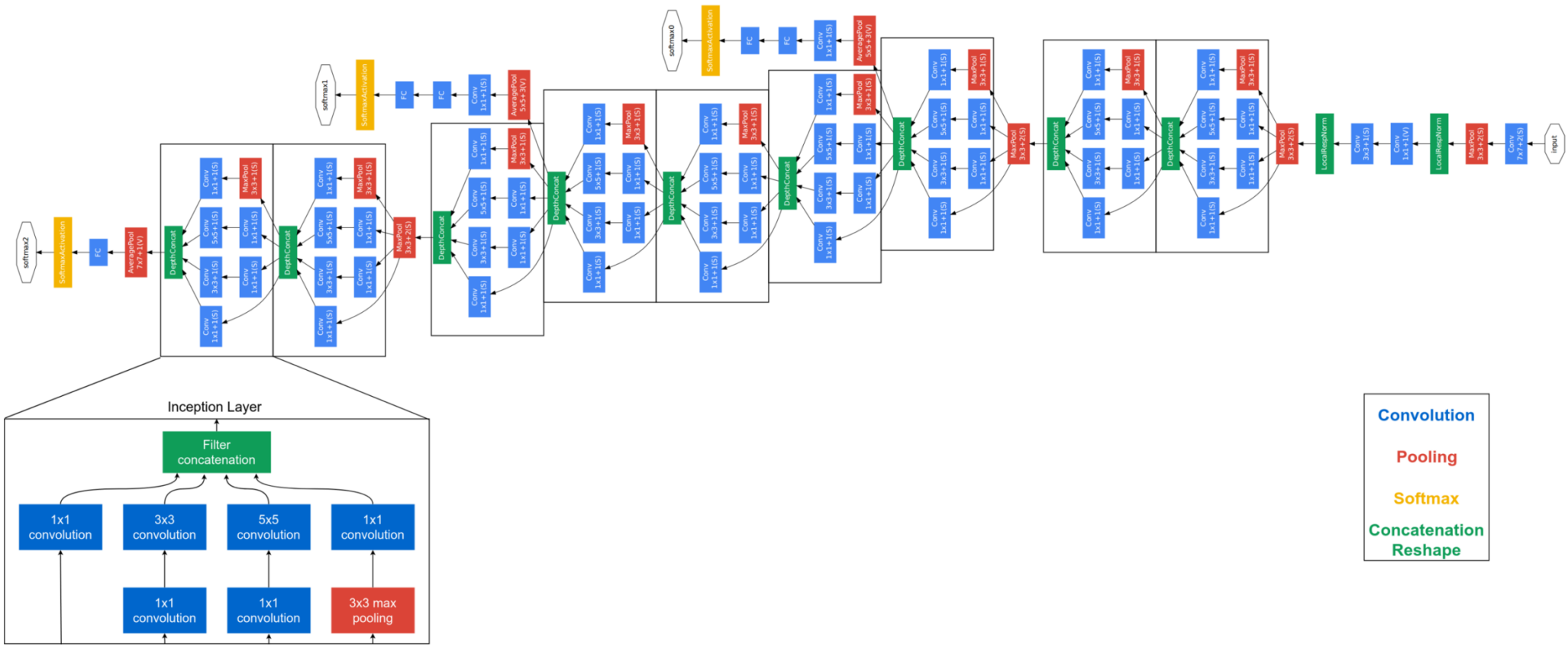
❖ 개요

- 2014년 ILSVRC 대회에서, 1등을 한 모델
- 22개의 레이어
- 노드 간의 연결을 줄이면서(Sparse connectivity), 행렬 연산은 Dense 연산을 하도록 처리하는가 ➔ Inception module



■ GoogLeNet

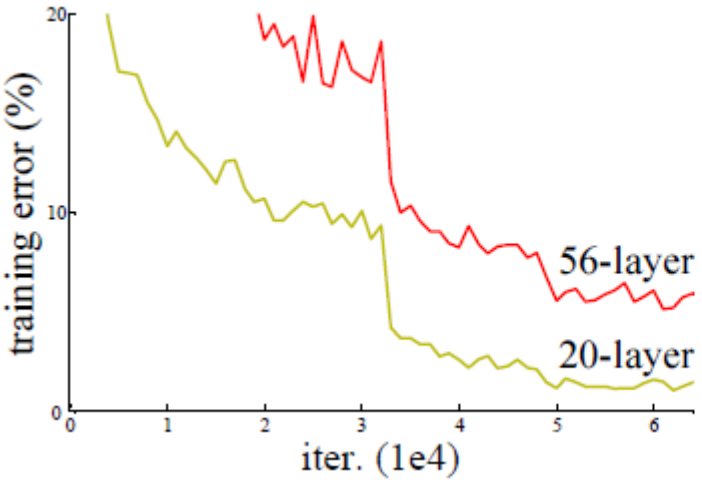
❖ 구조

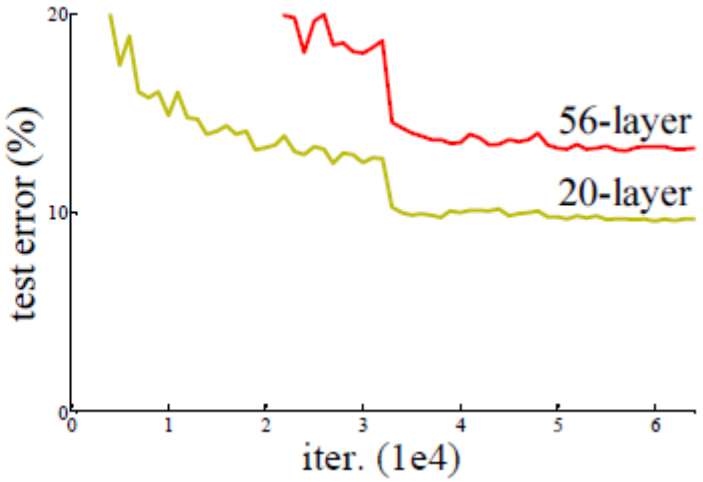


■ ResNet (Residual Network)

❖ 개요

- 2015년 ILSVRC 대회에서, 1등을 한 모델 (Microsoft)
- 152개 층
- 망을 깊게하면 무조건 성능이 좋아질까?

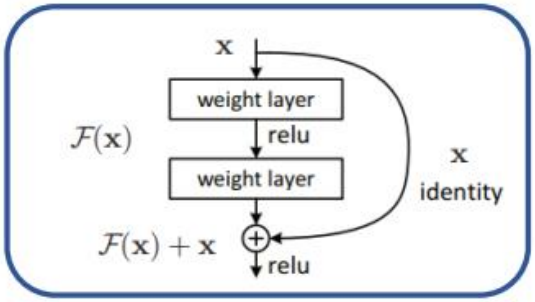


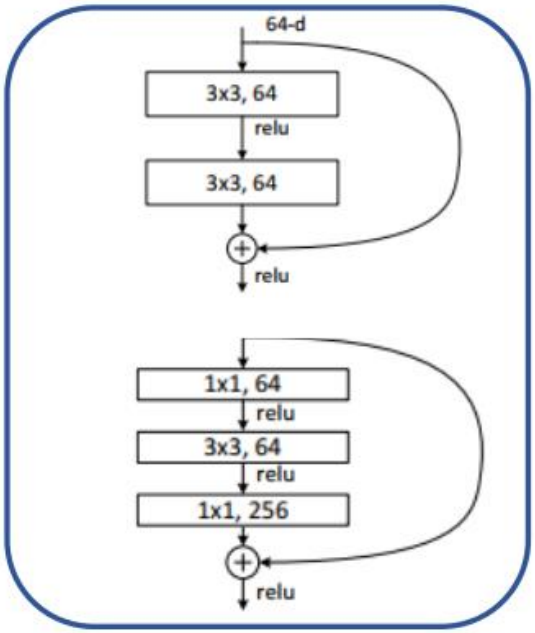


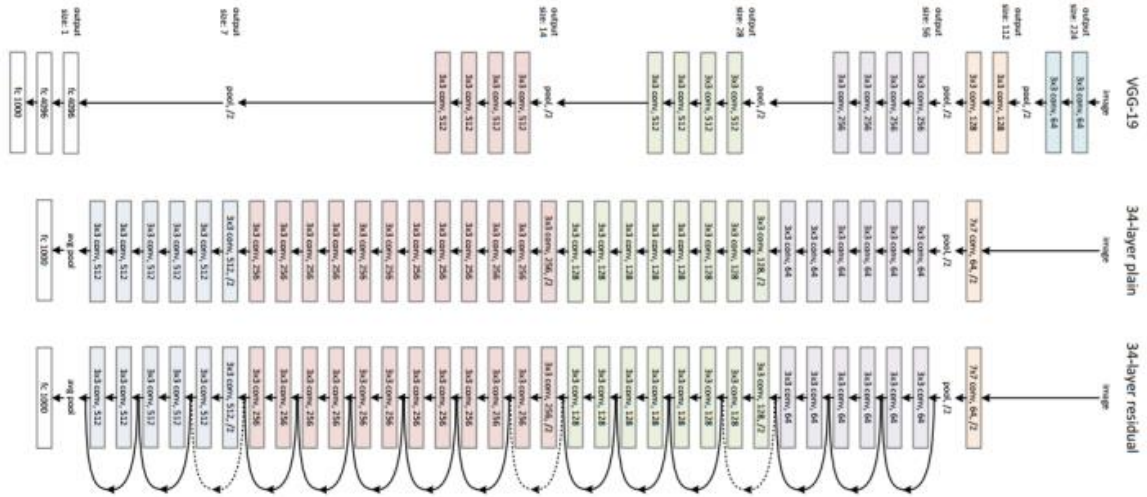
- Degradation 문제 해결

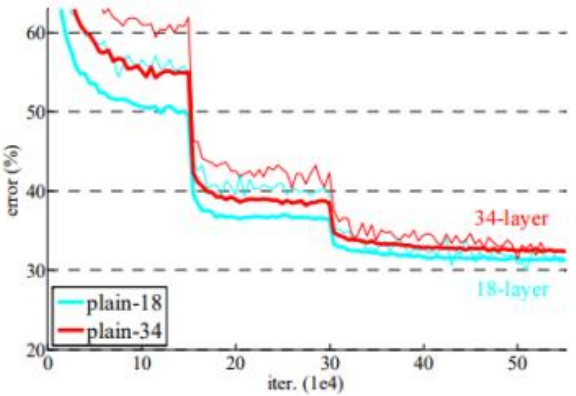
■ ResNet

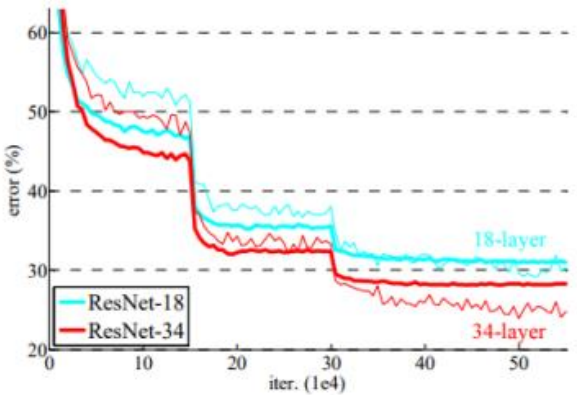
❖ 구조

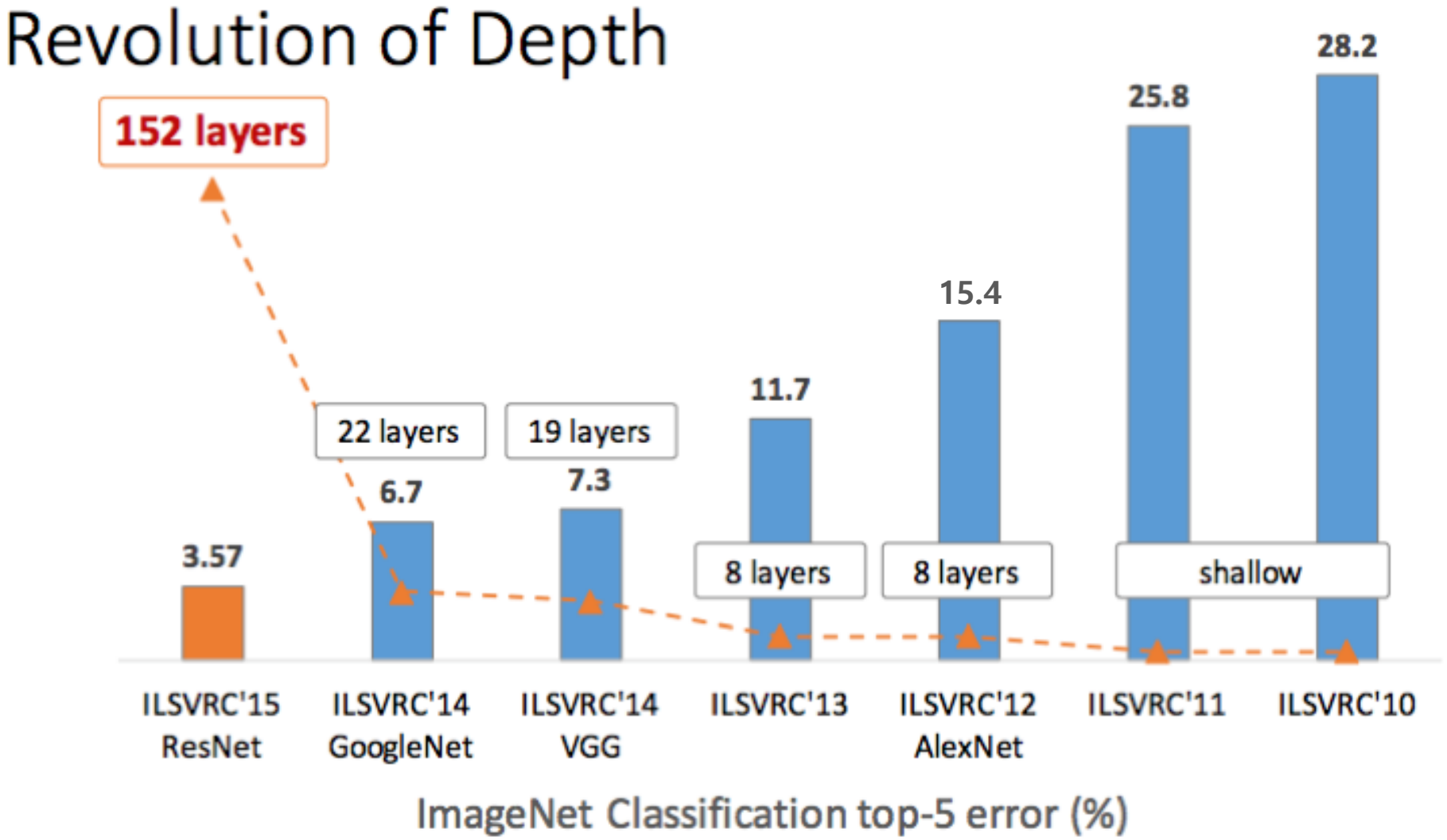

x
weight layer
$\mathcal{F}(x)$
relu
weight layer
x
identity
$\mathcal{F}(x) + x$
relu


64-d
3x3, 64
relu
3x3, 64
relu
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu


VGG-19
34-layer plain
34-layer residual


error (%)
iter. (1e4)
plain-18
plain-34
34-layer
18-layer


error (%)
iter. (1e4)
ResNet-18
ResNet-34
18-layer
34-layer


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
15.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)