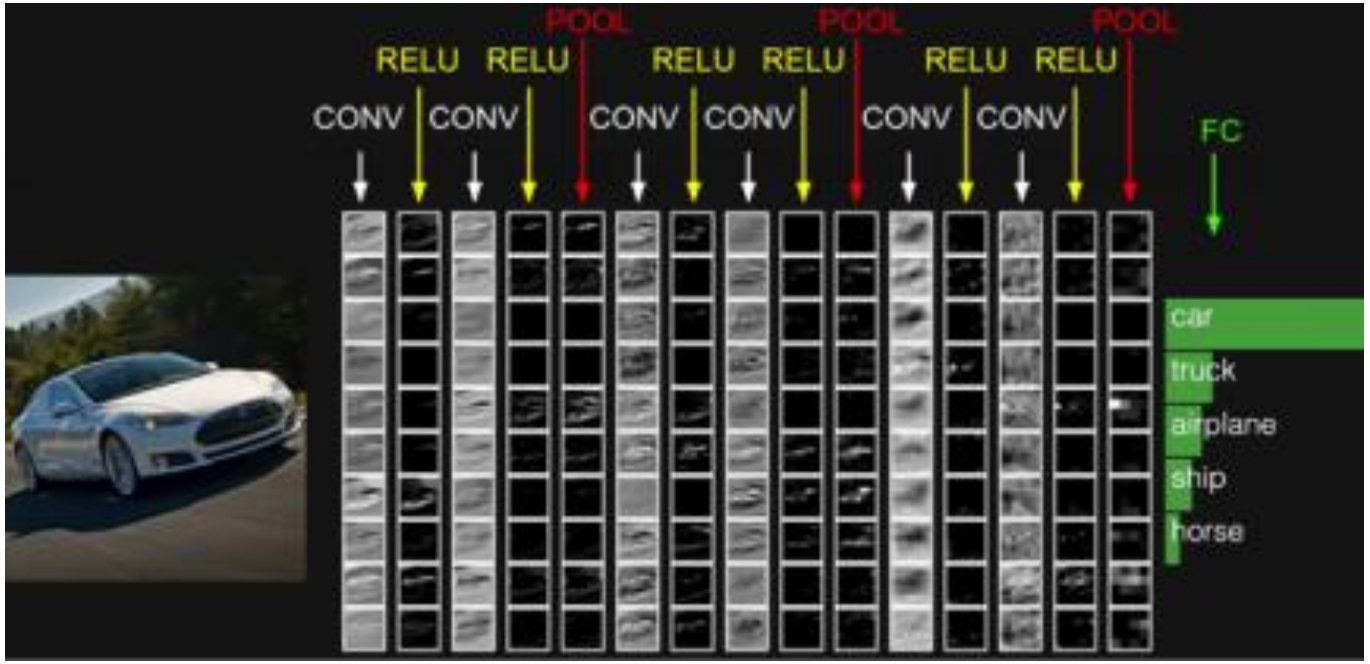
9장. 합성곱 신경망
(Convolutional Neural Network: CNN)

1. CNN 개요
2. 컨보루션 연산. 폴링. 패딩
3. 필터를 통해 데이터 특징을 추출하는 원리
4. 4. CNN 모델 실습

1. CNN 개요

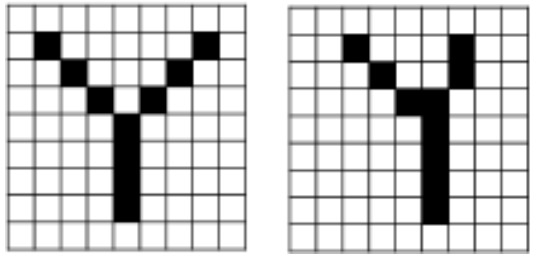
❖ 합성곱 신경망(Convolutional Neural Network)?

- 이미지 처리에 탁월한 성능을 보이는 신경망
- 합성곱층과(Convolution layer)와 풀링층(Pooling layer)으로 구성

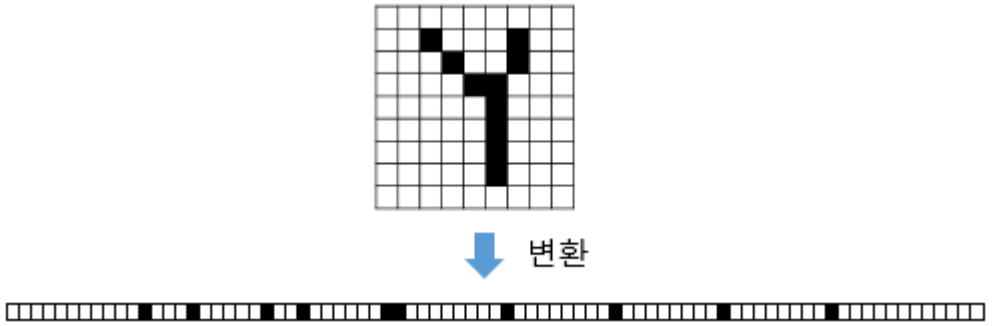


1. CNN 개요

❖ 1. CNN 개요



Y를 정자로 쓴 손글씨와 휘갈겨 쓴 손글씨를 2차원 텐서인 행렬로 표현한 것



이미지를 1차원 텐서인 벡터로 변환하고 다층 퍼셉트론의 입력층으로 사용

두 이미지를 기계가 같은 글자(Y)로 인식하는가?

공간적 정보를 잃은 1차원 벡터로 된 정보로 기계나 사람이 Y로 인식하는가?

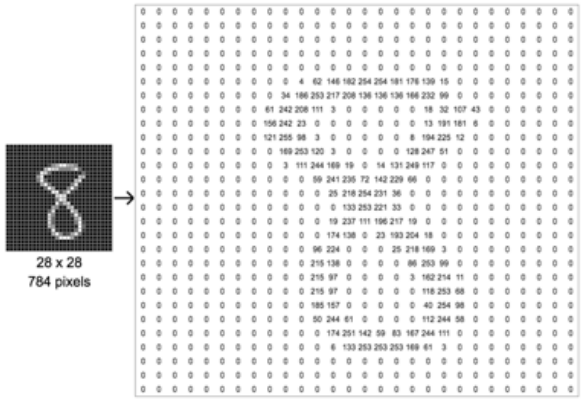
이미지의 공간적인 구조 정보를 보존하면서 학습할 수 있는 방법이 필요하며, 이를 위해 사용하는 것이 합성곱 신경망 임

공간적 정보란 :거리가 가까운 어떤 픽셀들끼리는 어떤 연관이 있고, 어떤 픽셀들끼리는 값이 비슷하거나 등을 포함

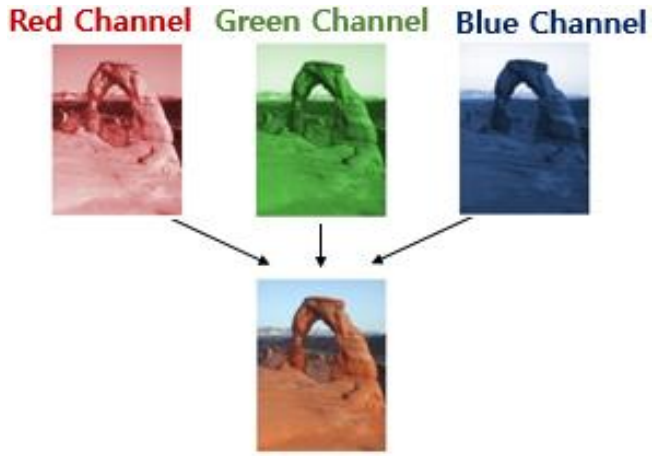
1. CNN 개요

❖ 채널(Channel)

- 기계는 글자나 이미지보다 숫자(텐서)를 더 잘 처리함
- 이미지는 **(높이, 너비, 채널)**이라는 3차원 텐서
- 높이: 이미지의 세로 방향 픽셀 수, 너비: 이미지의 가로 방향 픽셀 수, 채널: 색 성분을 의미
- 흑백 이미지는 채널 수가 1이며, 각 픽셀은 0부터 255 사이의 값을 가집니다. 아래는 28 × 28 픽셀의 손글씨 데이터를 보여줌



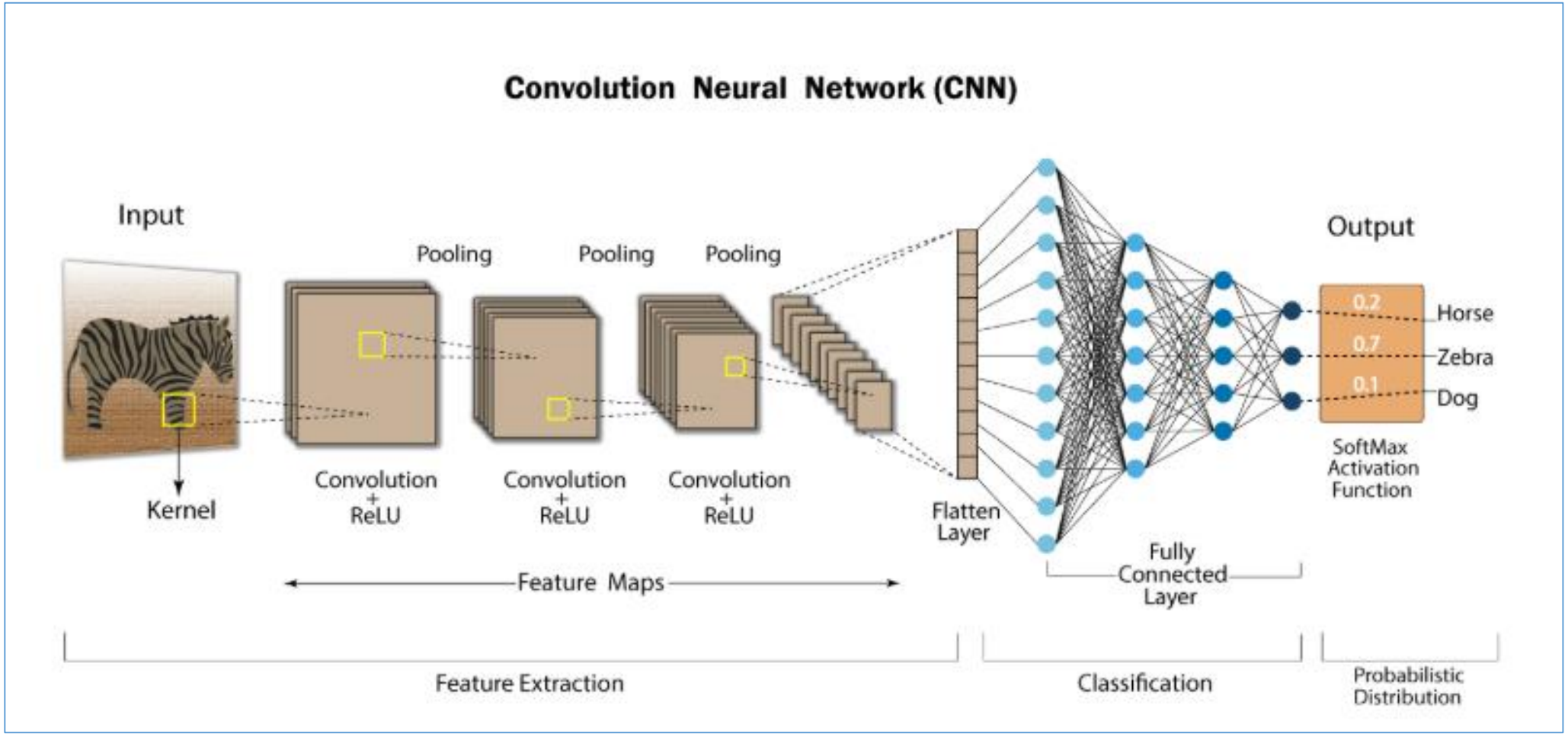
(28 × 28 × 1)의 크기를 가지는 3차원 텐서.



(28 × 28 × 3)의 크기를 가지는 3차원 텐서

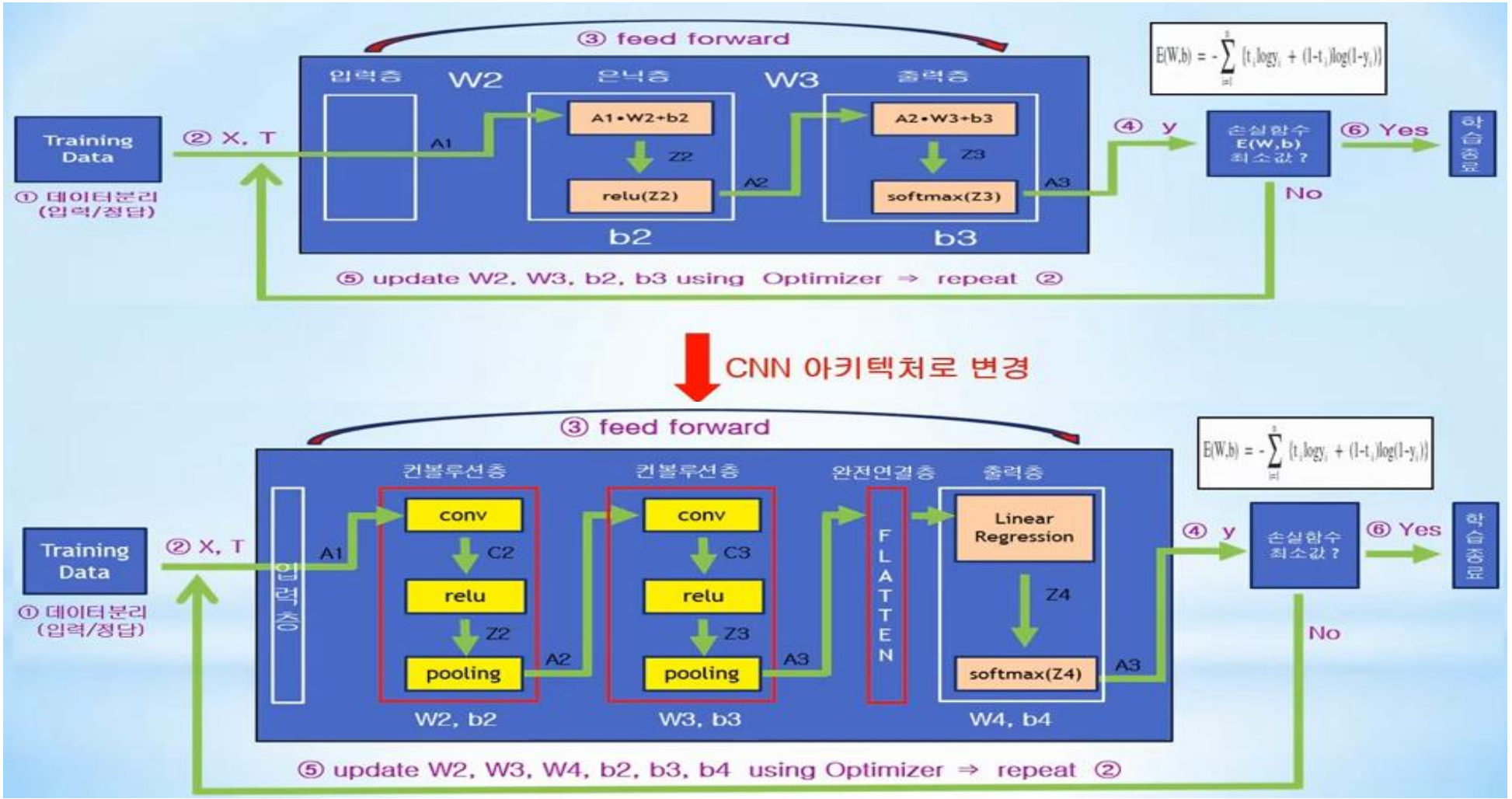
1. CNN 개요

❖ CNN 전체 구조와 역할



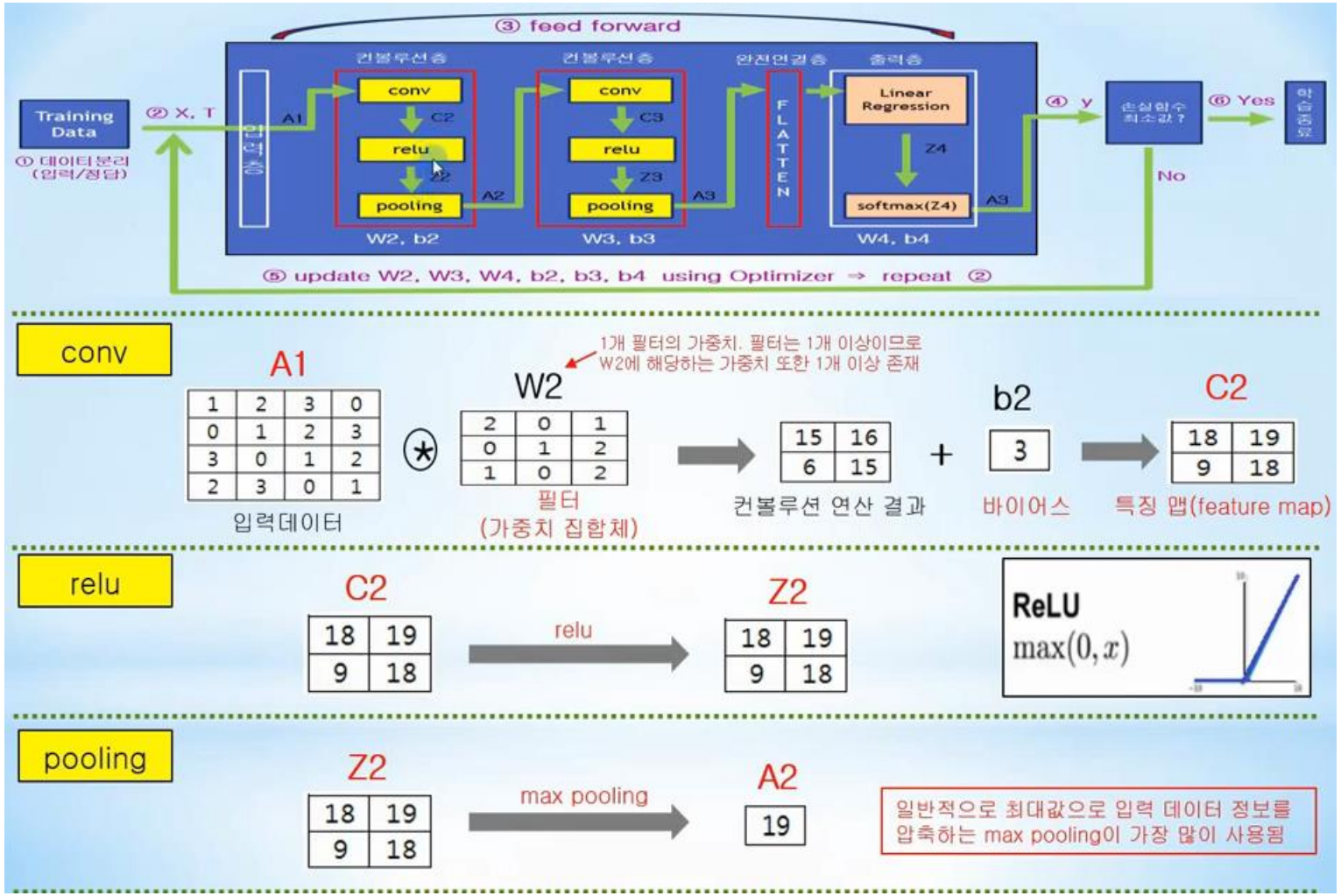
1. CNN 개요

❖ 아키텍처 비교(ANN vs. CNN)



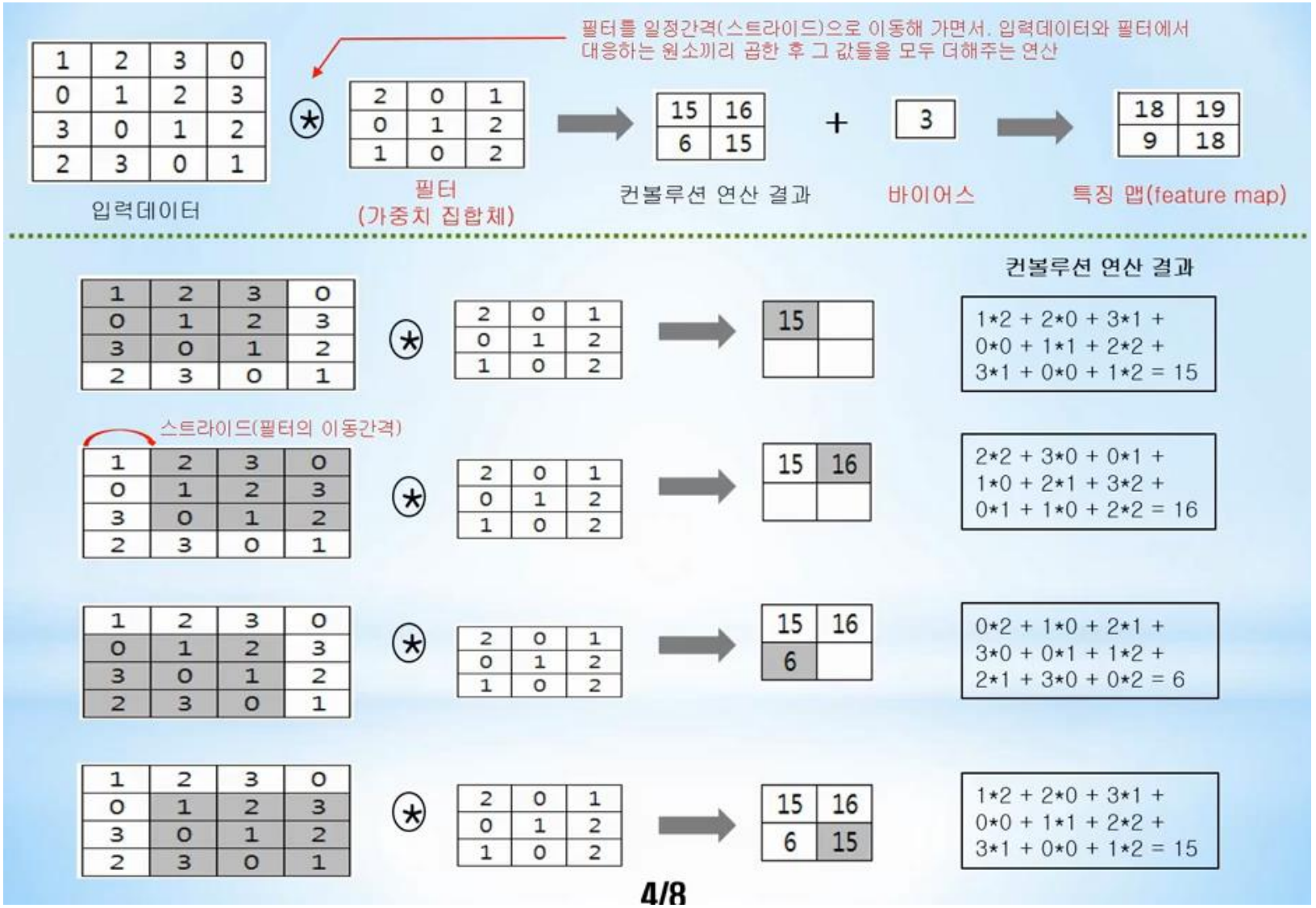
2. 컨보루션 연산. 폴링. 패딩

❖ 합성곱/폴링 연산(Convolution/Pooling operation)



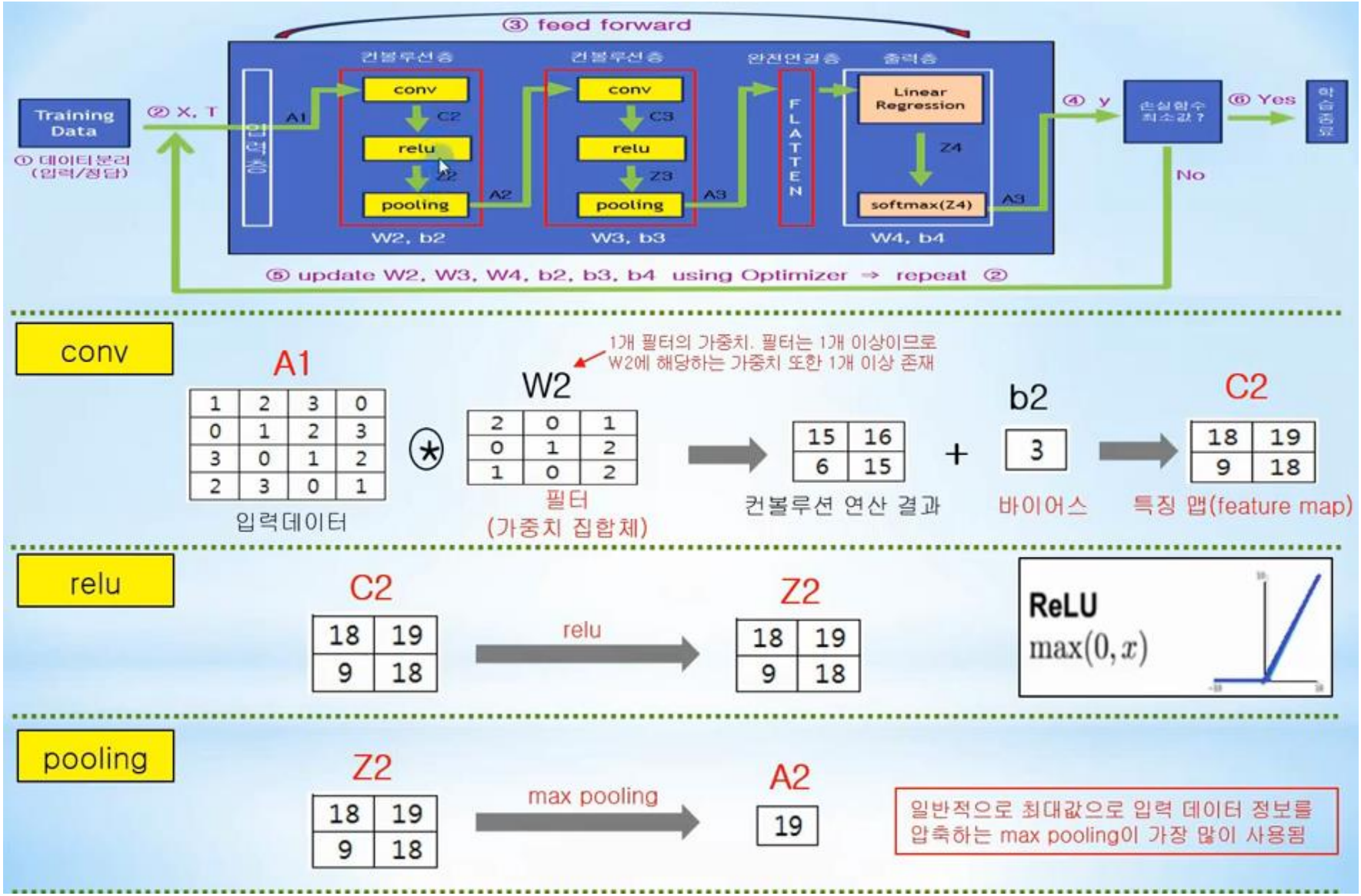
2. 컨보루션 연산. 폴링. 패딩

❖ 합성곱 연산



2. 컨보루션 연산. 폴링. 패딩

❖ **ReLU/Pooling 연산**



2. 컨보루션 연산. 폴링. 패딩

❖ 패팅(Padding)

- 컨볼루션 연산을 수행하기 전에 입력 데이터 주변을 특정 값(예를 들면 0)으로 채우는 것, 컨볼루션 연산에서 자주 이용되는 방법
  - 컨볼루션 연산을 수행하면 데이터 크기(shape)이 줄어드는 단점을 방지

2. 컨보루션 연산. 폴링. 패딩

❖ 컨볼루션 연산을 통한 출력 데이터 크기 계산

- 입력 데이터 크기(H,W), 필터 크기(FH,FW), 패딩 P, 스트라이드 S일 때 출력 데이터 크기 (OH, OW)

$$OH = \frac{H + 2P - FH}{S} + 1 \qquad OW = \frac{W + 2P - FW}{S} + 1$$

[예1] 입력 (4, 4), 필터 (3, 3), 패딩 1, 스트라이드 1 ⇒ 출력 (4, 4)

$$OH = \frac{4 + 2*1 - 3}{1} + 1 = 4 \qquad OW = \frac{4 + 2*1 - 3}{1} + 1 = 4$$

[예2] 입력 (7, 7), 필터 (3, 3), 패딩 0, 스트라이드 2 ⇒ 출력 (3, 3)

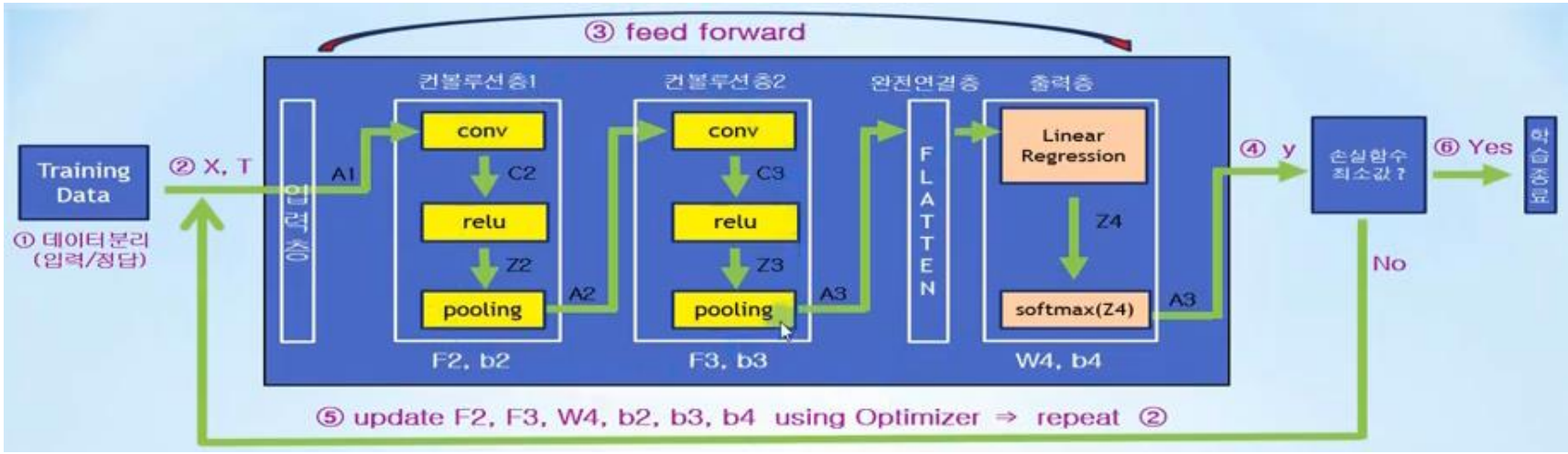
$$OH = \frac{7 + 2*0 - 3}{2} + 1 = 3 \qquad OW = \frac{7 + 2*0 - 3}{2} + 1 = 3$$

[예3] 입력 (28, 31), 필터 (5, 5), 패딩 2, 스트라이드 3 ⇒ 출력 (10, 11)

$$OH = \frac{28 + 2*2 - 5}{3} + 1 = 10 \qquad OW = \frac{31 + 2*2 - 5}{3} + 1 = 11$$

3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 컨볼루션 층(convolution layer) 역할 => 입력 데이터 의 특징 추출



	conv 출력	relu 출력	pooling 출력	컨볼루션 층 역할
컨볼루션층1	A1 ⊛ F2 + b2 = C2 입력 데이터 / 필터 (가중치 집합체) / 바이어스 / 특징 맵 (feature map)	→ Z2 C2>0, C2 C2<=0, 0	→ A2 max pooling	입력데이터 A1과 가중치들의 집합체인 1개 이상의 필터 F2와 컨볼루션 연산을 통해 입력데이터 A1의 특징(feature)을 추출하는 역할을 수행함
컨볼루션층2	A2 ⊛ F3 + b3 = C3 입력 데이터 / 필터 (가중치 집합체) / 바이어스 / 특징 맵 (feature map)	→ Z3 C3>0, C3 C3<=0, 0	→ A3 max pooling	입력데이터 A2과 가중치들의 집합체인 1개 이상의 필터 F3와 컨볼루션 연산을 통해 입력데이터 A2의 특징(feature)을 추출하는 역할을 수행함

3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 컨볼루션 층(convolution layer) 역할

① 입력데이터와 1개 이상의 필터들과 컨볼루션 연산을 통해서

② 입력데이터 특징(feature)을 추출하여 특징맵(feature map)을 만들고

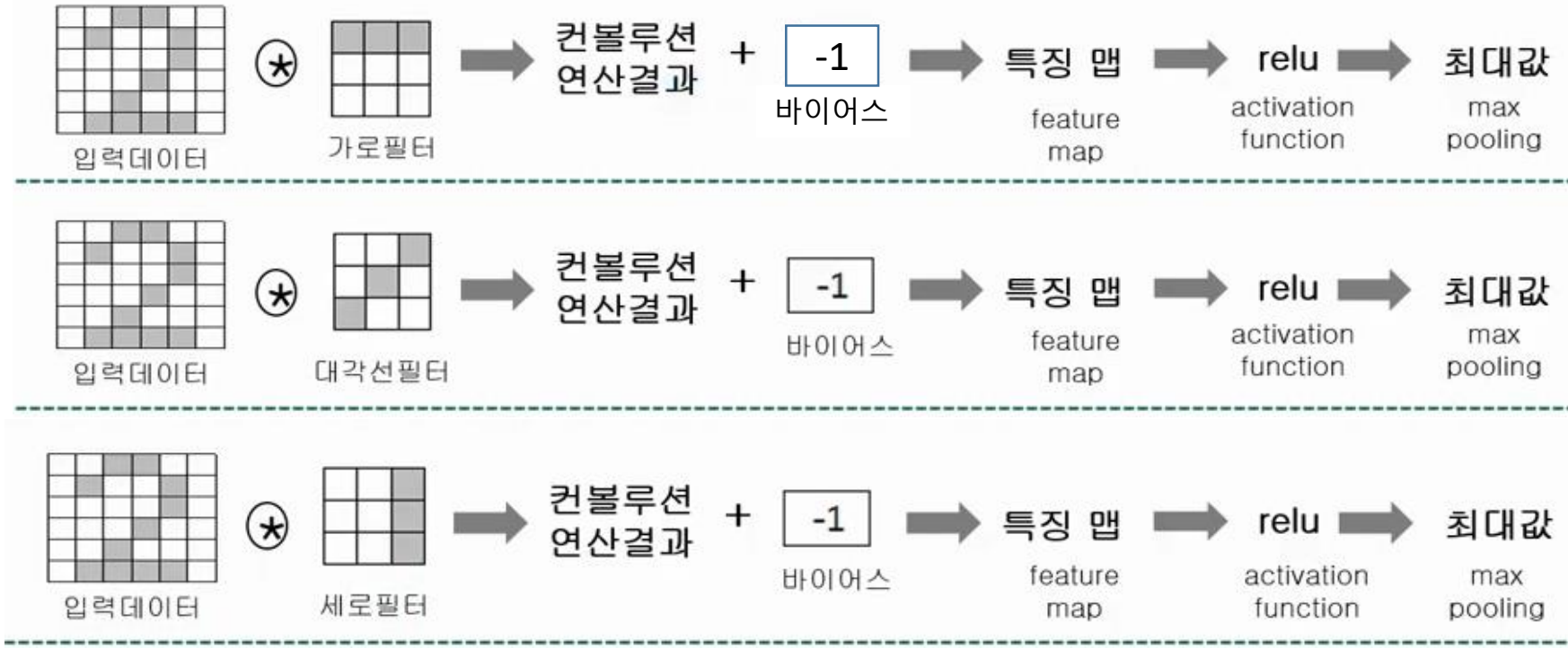
③ 특징맵에서 초대 값을 뽑아내서 다음 층으로 전달

필터를 통해 데이터 특징을 추출 ?

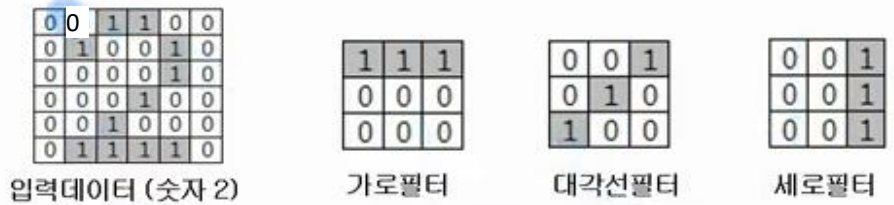
3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 특징추출 과정

➢ 입력데이터 1개 (숫자 2)에 필터 3개 (가로, 대각선, 세로 필터) 적용 (계산 편의를 위해 패딩 적용하지 않음)



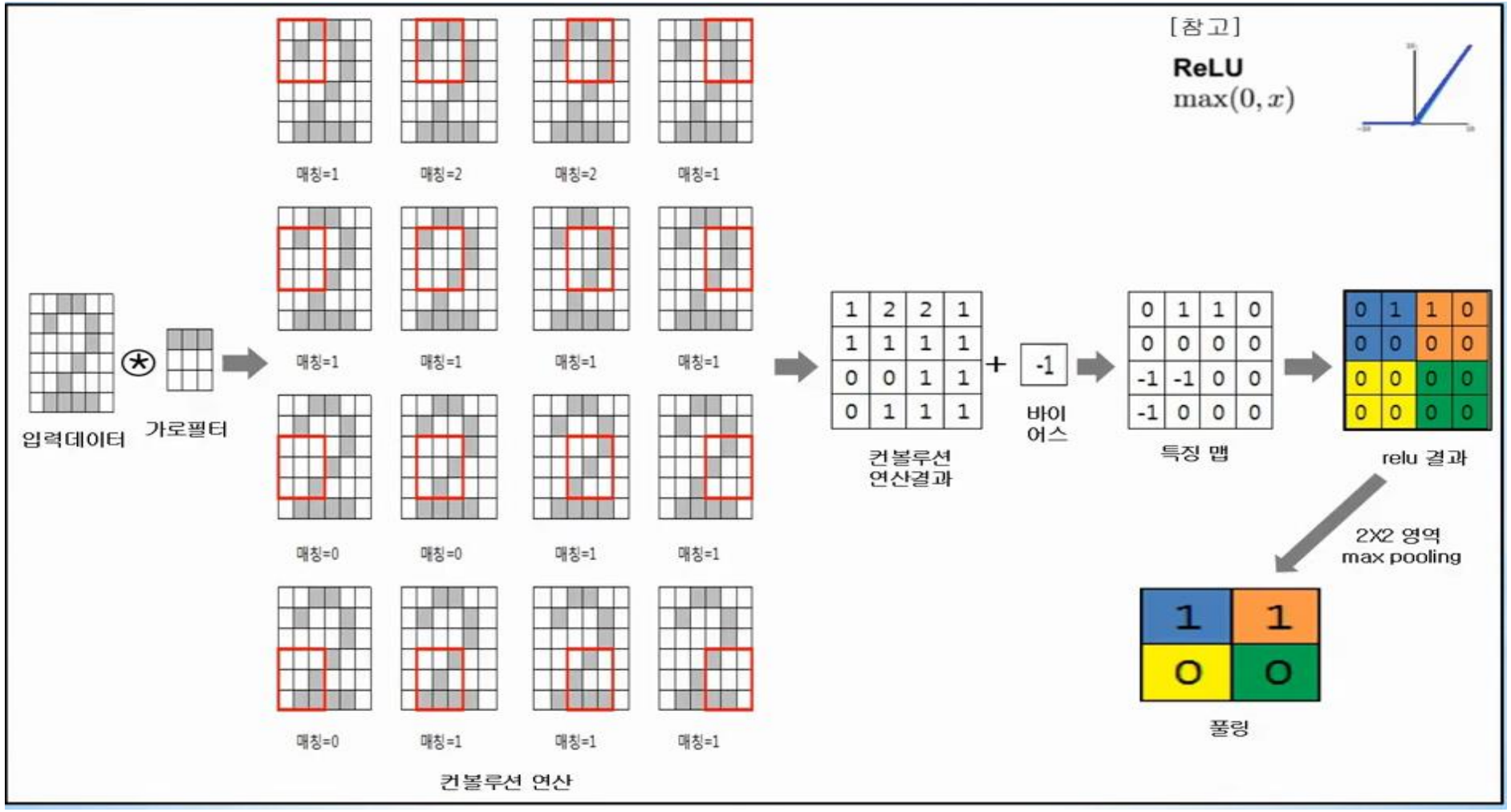
[참고] 입력데이터와 필터



이미지참조(재구성) : 처음 배우는 딥러닝 수학

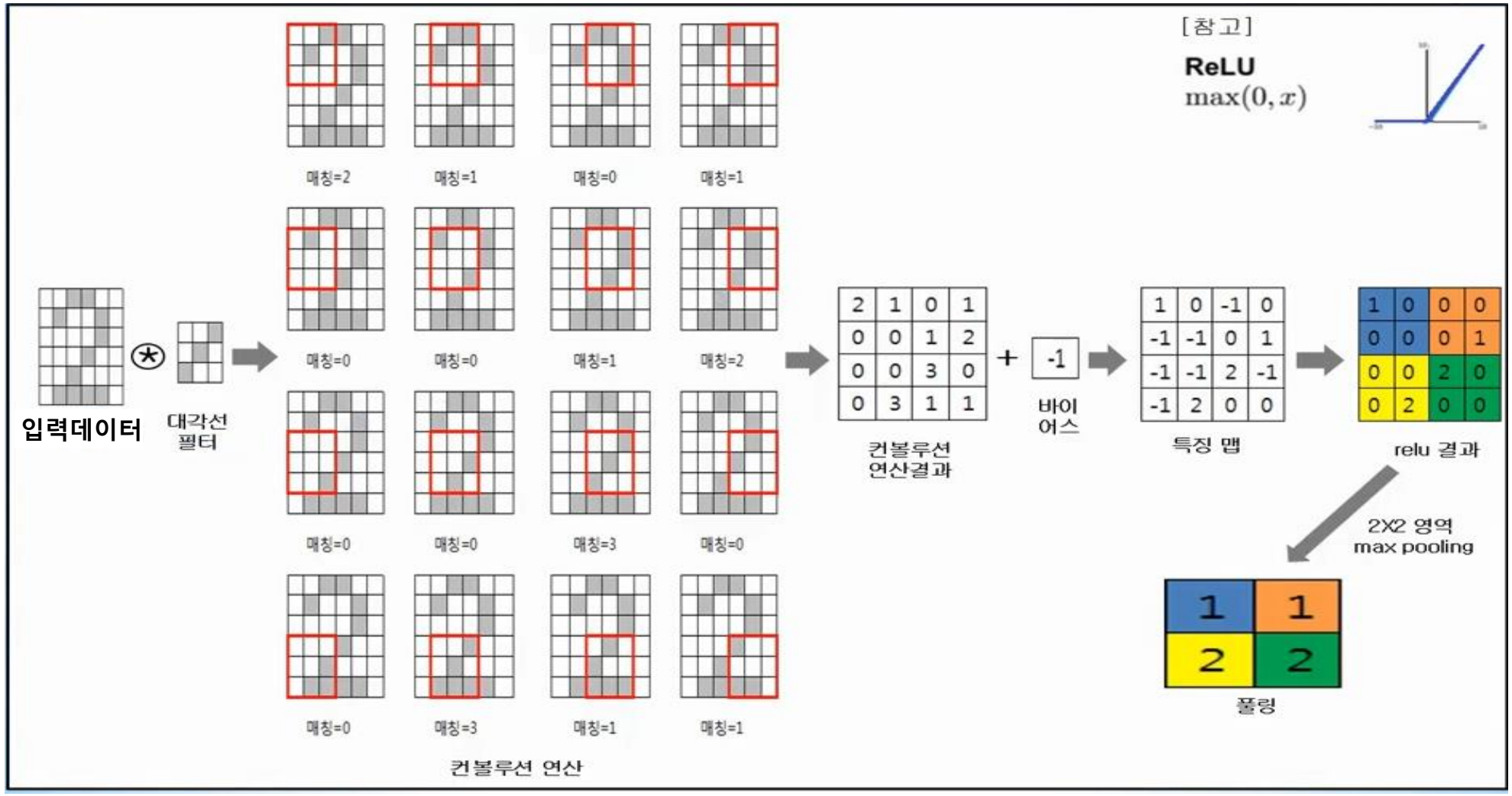
3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 가로 필터를 통한 입력 데이터 특징 추출(스트라이트 1,패딩 없음)



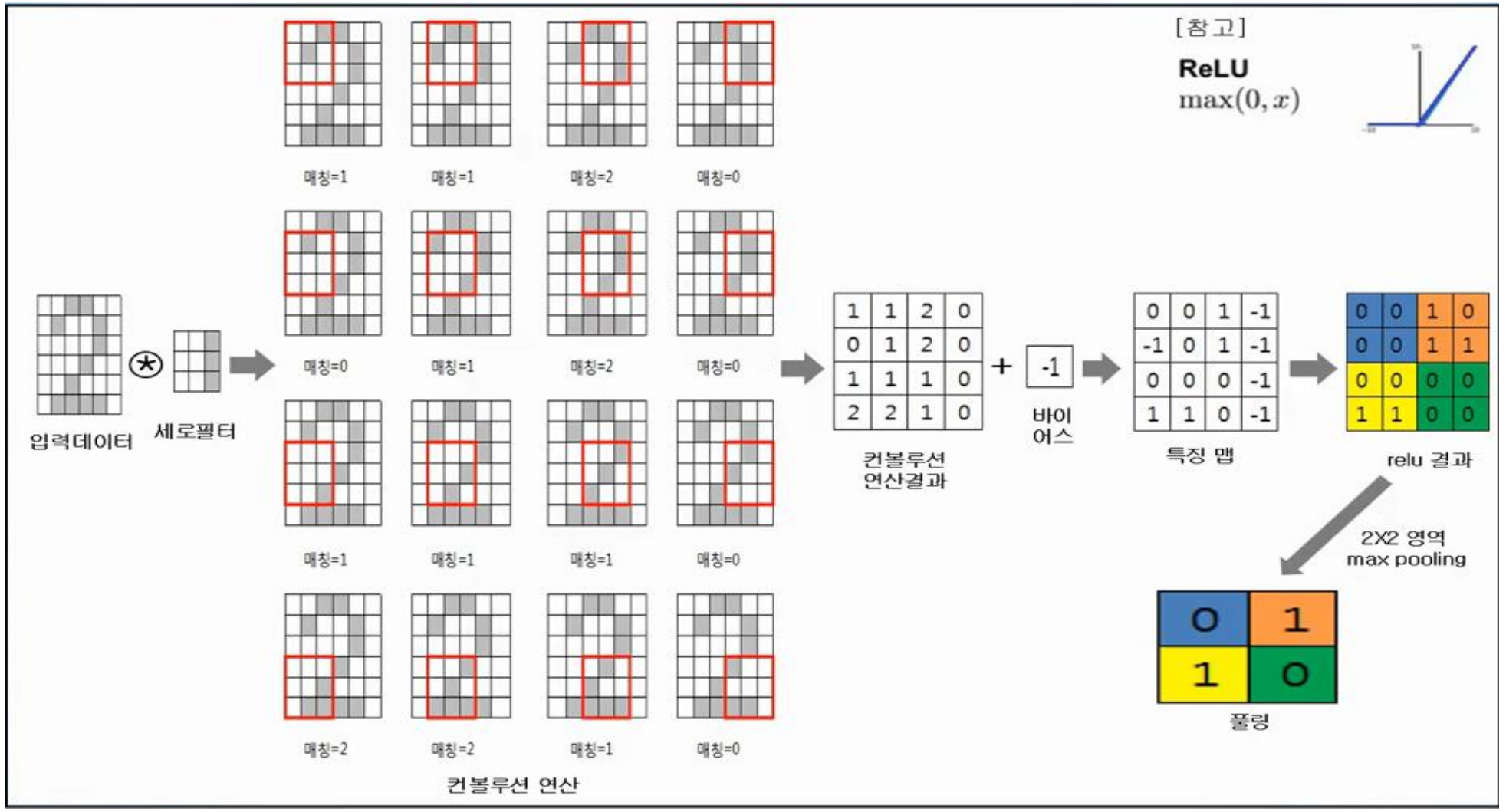
3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 대각선필터를 통한 데이터 특징 추출



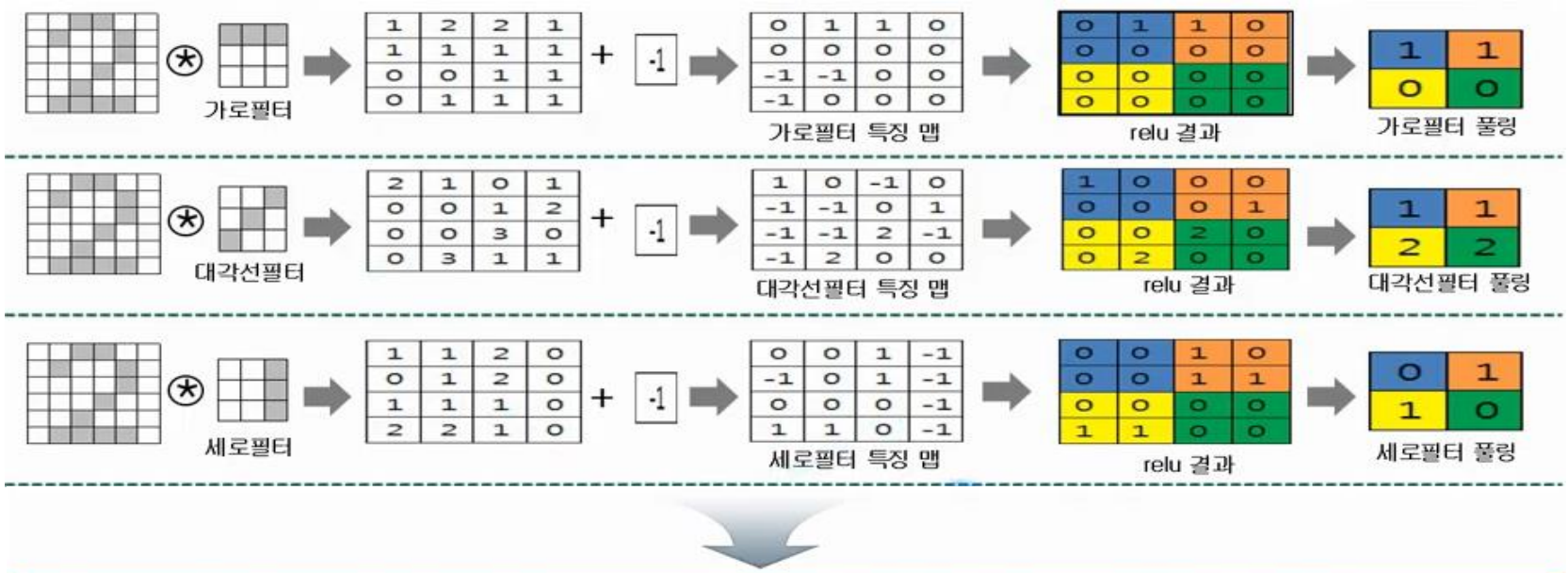
3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 새로 필터를 통한 입력 데이터 특징 추출



3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 필터를 통한 입력데이터 특징 추출원리-특징 맵이 압축된 풀링 값

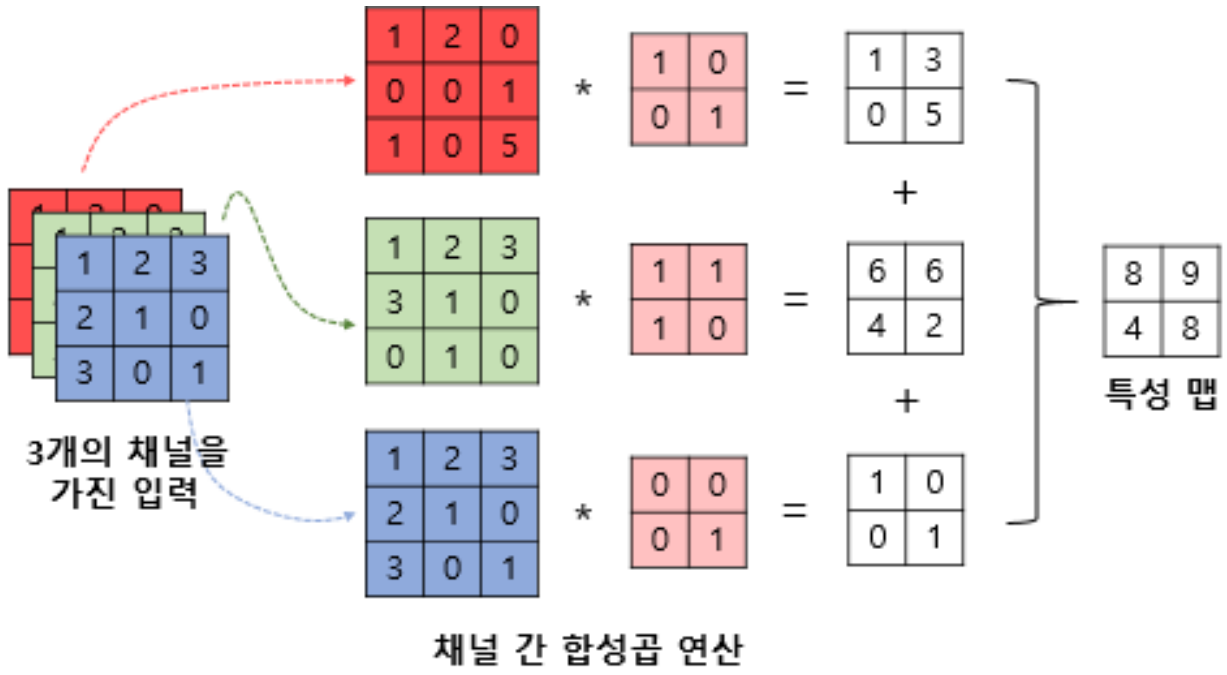


➢ 컨볼루션 연산 결과인 특징 맵(feature map) 값을 압축하고 있는 풀링 값을 보면,

- 대각선 필터에 대한 풀링 값이 가로와 세로필터의 풀링 값 보다 큰 값으로 구성되어 있는데,
- 풀링 값이 크다는 것은, 데이터 안에 해당 필터의 특징(성분)이 많이 포함되어 있는 것을 의미함. 즉, 특징 맵 값이 압축되어 있는 풀링 결과 값을 통해 데이터의 특징(성분)을 추출 할 수 있음
- 위의 예제를 보면, 입력 데이터 '2' 는 대각선 특징이 가로나 세로 특징보다 더욱 많이 포함되어 있으며 이러한 특징을 추출하는데 대각선 필터가 가로나 세로보다 유용하다는 것을 알 수 있음

3. 필터를 통해 데이터 특징을 추출하는 원리

❖ 다수의 채널을 가질 경우의 합성곱 연산(3차원 텐서의 합성곱 연산)

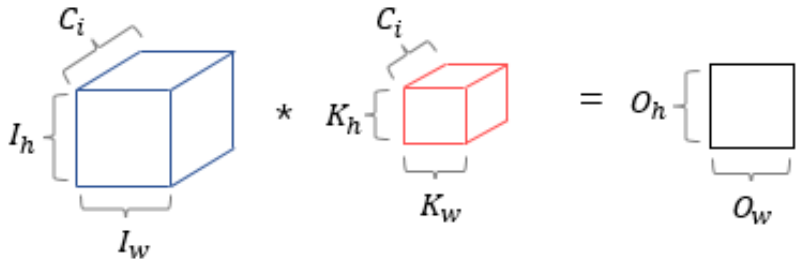


3. 필터를 통해 데이터 특징을 추출하는 원리

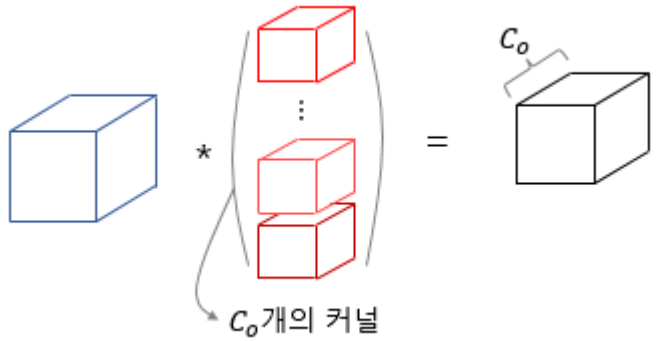
❖ 3차원 텐서의 합성곱 연산

- 일반화를 위해 사용하는 각 변수가 의미하는 바는 다음과 같다

- I_h : 입력의 높이
- I_w : 입력의 너비
- K_h : 커널의 높이
- K_w : 커널의 너비
- O_h : 특성 맵의 높이
- O_w : 특성 맵의 너비
- C_i : 입력 데이터의 채널



1개 커널 3차원 텐서의 합성곱 연산



다수 커널 사용 3차원 텐서의 합성곱 연산

4. CNN 모델 실습 : CNN 모델 살펴보기

```
import torch
import torch.nn as nn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
import torch.nn.init
```

```
#배치크기(한번에 로딩하는 이미지 수), 채널, 높이, 너비
input=torch.Tensor(1, 1, 28, 28)
print(input.size())
print(input)
```

```
conv1=nn.Conv2d(1, 32, 3, padding=1)
# Conv2d(입력 채널 수, 출력 채널 수, kernal_size(필터의 크기),
stride=1, padding=1)
print(conv1)
```

```
conv2=nn.Conv2d(32, 64, kernel_size=3, padding=1)
print(conv2)
```

```
pool=nn.MaxPool2d(2)
print(pool)
```

```
out1=conv1(input)
out2=pool(out1)
print(out1.size())
print(out2.size())
```

```
out3=conv2(out2)
out4=pool(out3)
print(out3.size())
print(out4.size())
```

```
# 분류를 위한 Flatten
out=out4.view(out4.size(0), -1)
print(out.size())
```

```
fc=nn.Linear(3136, 10)
outf=fc(out)
print(outf.size())
print(outf)
```

4. CNN 모델 실습 : MNIST

```
# cuda 사용설정
device='cuda' if torch.cuda.is_available() else 'cpu'
torch.manual_seed(777)

if device=='cuda':
    torch.cuda.manual_seed_all(777)
```

```
learing_rate=0.001
epochs=15
batch_size=100
```

```
mnist_train=dsets.MNIST(root='MNIST_data',
                        train=True,
                        transform=transforms.ToTensor(),
                        download=True)

mnist_test=dsets.MNIST(root='MNIST_data',
                        train=False,
                        transform=transforms.ToTensor(),
                        download=True)
```

```
print(mnist_train)
print(mnist_test
```

```
train_loader=DataLoader(dataset=mnist_train,
                        batch_size=batch_size,
                        shuffle=True,
                        drop_last=False)

test_loader=DataLoader(dataset=mnist_test,
                        batch_size=batch_size,
                        shuffle=True,
                        drop_last=False)
```

```
for X, Y in train_loader:
    print(X.size())
    print(Y.size())
    break

for Y,Y in test_loader:
    print(X.size())
    print(Y.size())
    break
```

4. CNN 모델 실습 : MNIST

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        #conv layer1
        #image in shape(100, 1, 28, 28)
        # conv -> (?, 32, 28, 28)
        # pool ->(?, 32, 14,14)

        self.layer1=nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3,stride=1,padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        #conv layer1
        #image in shape(?, 32, 14, 14)
        # conv -> (?, 64, 14, 14)
        # pool ->(?, 64, 7, 7)
        self.layer2=nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3,stride=1,padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        #완전 연결층 (100,64*7*7)
        self.fc=nn.Linear(64*7*7, 10, bias=True)
        #완전 연결층 한정으로 가중치 초기화
        nn.init.xavier_uniform_(self.fc.weight)
```

```
    def forward(self, x):
            out=self.layer1(x)
            out=self.layer2(out)
            out=out.view(out.size(0),-1)
            out=self.fc(out)
            return out
```

```
model=CNN().to(device)
criterion=nn.CrossEntropyLoss().to(device)
optimizer=torch.optim.Adam(model.parameters(),
lr=learing_rate)
```

```
print(model)
print(list(model.parameters()))
```

```
train_total_batch=len(train_loader)
test_total_batch=len(test_loader)
print(train_total_batch)
print(test_total_batch)
```

4. CNN 모델 실습 : MNIST

```
for epoch in range(epochs):
    avg_cost=0

    for X, Y in train_loader:
        X=X.to(device)
        Y=Y.to(device)

        optimizer.zero_grad()
        y_hat=model(X)
        cost=criterion(y_hat, Y)
        cost.backward()
        optimizer.step()

        avg_cost+=cost/train_total_batch
    print('Epoc:', epoch, 'cost:', avg_cost)
```

```
with torch.no_grad():
    avg_accuracy=0

    for X,Y in test_loader:
        X=X.to(device)
        Y=Y.to(device)
        pred=model(X)
        correct_prde=torch.argmax(pred, -1)==Y
        accuracy=correct_prde.float().sum()
        avg_accuracy+=accuracy
    avg_accuracy=avg_accuracy/test_total_batch

print('Accuracy:', avg_accuracy)
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

```
import torch
import torch.nn as nn
import numpy as np
from PIL import Image #
from torchvision import transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader, TensorDataset
from torchvision.utils import save_image
import matplotlib.pyplot as plt

#from google.colab import drive # 구글드라이브 연결 라이브러리
#drive.mount('/content/drive') #구글 드라이브 연결
```

```
torch.manual_seed(777)
IMAGE_SIZE=128

device='cuda' if torch.cuda.is_available() else 'cpu'
if device=='cuda':
    torch.cuda.manual_seed_all(777)
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

```
original_datasets=ImageFolder(root='flower_photos/',
                              transform=transforms.Compose([
                                  transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
                                  transforms.ToTensor()
                              ]))
print(original_datasets)
```

```
original_loader=DataLoader(original_datasets,
                           batch_size=2313,
                           shuffle=True,
                           drop_last=False,
                           num_workers=1) # 데이터 로드시 사용할 병렬 프로세스 수, 단일 프로세스 사용
```

```
for X, Y in original_loader:
    print(X.size(), Y.size())
    print(Y)
    break
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

```
# 이미지 로더로 부터 이미지와 label을 추출
total_batch=len(original_loader)
print(total_batch)

# loader로부터 새로운 이미지와 레이블 추출
original_images, labels=next(iter(original_loader))

print(original_images.shape)
print(labels.shape)

print(labels[:10])
```

```
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
#import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(8,10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.title(labels[i].item())
    plt.imshow(original_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

```
X1=original_images #3,128,128 : 채널, 행, 열
Y1=labels
print(X1.size(), Y1.size())
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 이미지 증강(이미지 확대 후 센터 크롭)하여 이미지 개수 늘리기

```
Image_transform=transforms.Compose([
    transforms.Resize((256,256)),
    transforms.CenterCrop((IMAGE_SIZE, IMAGE_SIZE)),
    transforms.ToTensor()
])
```

```
transforms_datasets=ImageFolder(root='flower_photos/',
                                transform=Image_transform)
transforms_loader=DataLoader(transforms_datasets,
                             batch_size=1000,
                             shuffle=True,
                             num_workers=1)
```

```
transforms_images, transforms_labels=next(iter(transforms_loader))
```

```
print(transforms_images.size(), transforms_labels.size())
X2=torch.cat([X1, transforms_images], dim=0)
Y2=torch.cat([Y1, transforms_labels],dim=0)
print(X2.size(), Y2.size())
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 이미지 변형 후 원본 이미지와 비교

```
none_datasets=ImageFolder(root='flower_photos/',
                          transform=transforms.Compose([
                            transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
                            transforms.ToTensor()
                            ]))
none_loader=DataLoader(none_datasets, batch_size=100, shuffle=False, num_workers=1)
```

```
none_images, none_labels=next(iter(none_loader))
```

```
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.title(transforms_labels[i].item())
    plt.imshow(transforms_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()

for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.title(none_labels[i].item())
    plt.imshow(none_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 이미지 증강 함수 활용

```
def create_loader(transform):
    transforms_datasets=ImageFolder(root='flower_photos',
                                    transform=transform)
    transform_loader=DataLoader(transforms_datasets,
                                batch_size=1000,
                                shuffle=True,
                                num_workers=1)

    transform_images, labels=next(iter(transform_loader))
    return transform_images, labels
```

```
plt.figure(figsize=(16,4))
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.title(color_tf_labels[i].item())
    plt.imshow(color_tf_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

```
Image_transform=transforms.Compose([
    transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
    transforms.ColorJitter(brightness=(0.7,0.9), # 이미지 밝기설정
                           contrast=(1,1),
                           saturation=(0.7,0.9), # 이미지 대비 설정
                           #hue=(-0.2,0.2)),
    transforms.ToTensor()
])
color_tf_images, color_tf_labels=create_loader(Image_transform)
```

```
X3=torch.cat([X2, color_tf_images], dim=0)
Y3=torch.cat([Y2, color_tf_labels], dim=0)
print(X3.size(), Y3.size())
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

```
image_transform = transforms.Compose([
    transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
    # RandomHorizontalFlip 적용
    transforms.RandomHorizontalFlip(p=0.8),  # 수평 뒤집기
    transforms.ToTensor()
])

flip_tf_images, flip_tf_labels=create_loader(image_transform)

X4=torch.cat([X3, flip_tf_images], dim=0)
Y4=torch.cat([Y3, flip_tf_labels],dim=0)
print(X4.size(), Y4.size())
```

```
plt.figure(figsize=(16,4))
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.title(flip_tf_labels[i].item())
    plt.imshow(flip_tf_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

```
image_transform = transforms.Compose([
    transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
    # RandomRotation : 이미지 회전 적용
    transforms.RandomRotation(degrees=(-15, 15),
        interpolation=transforms.InterpolationMode.BILINEAR, fill=0),
    transforms.ToTensor()
])

rot_tf_images, rot_tf_labels=create_loader(image_transform)
X5=torch.cat([X4, rot_tf_images], dim=0)
Y5=torch.cat([Y4, rot_tf_labels], dim=0)
print(X5.size(), Y5.size())
```

```
plt.figure(figsize=(16,4))
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.title(rot_tf_labels[i].item())
    plt.imshow(rot_tf_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

```
image_transform = transforms.Compose([
    transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
    # GaussianBlur 적용
    transforms.GaussianBlur(kernel_size=(3, 3), sigma=(1.0, 2.0)),
    transforms.ToTensor()
])

GB_tf_images, GB_tf_labels=create_loader(image_transform)
X6=torch.cat([X5, GB_tf_images], dim=0)
Y6=torch.cat([Y5, GB_tf_labels], dim=0)
print(X5.size(), Y5.size())
```

이미지 블러링 하기

```
image_transform = transforms.Compose([
    transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
    # RandomAdjustSharpness 적용
    transforms.RandomAdjustSharpness(sharpness_factor=0.5, p=0.9),
    transforms.ToTensor()
])

sp_tf_images, sp_tf_labels=create_loader(image_transform)
X7=torch.cat([X6, sp_tf_images], dim=0)
Y7=torch.cat([Y6, sp_tf_labels], dim=0)
print(X6.size(), Y6.size())
```

이미지 샤프닝 하기

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 최종 데이터 셋으로 train data/test data 만들기

```
cnt0=(Y7==0).sum()
cnt1=(Y7==1).sum()
cnt2=(Y7==2).sum()
cnt3=(Y7==3).sum()
cnt4=(Y7==4).sum()
print(cnt0, cnt1, cnt2, cnt3, cnt4)
```

```
X=X7
Y=Y7
ratios=[0.8,0.2]
print(Y.size())
train_cnt=int(Y.size(0)*ratios[0])
test_cnt=int(Y.size(0)*ratios[1])
cnts=[train_cnt, test_cnt]
print(train_cnt, test_cnt)
```

```
indices=torch.randperm(X.size(0))
print(indices[:10])
```

```
x=torch.index_select(X, dim=0, index=indices)
y=torch.index_select(Y, dim=0, index=indices)
x_train=x[:cnts[0]]
x_test=x[cnts[0]:]
y_train=y[:cnts[0]]
y_test=y[cnts[0]:]

print(x_train.size(), y_train.size())
print(x_test.size(), y_test.size())
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 모델 작성을 위한 데이터셋 및 데이터 로더 구성

```
train_dataset=TensorDataset(x_train, y_train)
test_dataset=TensorDataset(x_test, y_test)
```

```
train_loader=DataLoader(train_dataset,
                        batch_size=100,
                        shuffle=True,
                        drop_last=False)
test_loader=DataLoader(test_dataset)
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 모델정의 클래스

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()

        # 입력(3, 128, 128)
        self.conv1=nn.Conv2d( #3, 128,128
            in_channels=3,
            out_channels=8,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.conv2=nn.Conv2d( #8, 64,64
            in_channels=8,
            out_channels=16,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.conv3=nn.Conv2d( #16, 32,32
            in_channels=16,
            out_channels=32,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.conv4=nn.Conv2d( #32, 16,16
            in_channels=32,
            out_channels=64,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.conv5=nn.Conv2d( #64, 8,8
            in_channels=64,
            out_channels=128,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.pool=nn.MaxPool2d(kernel_size=2,stride=2)
        self.fc1=nn.Linear(128*4*4, 128)
        self.fc2=nn.Linear(128,64)
        self.fc3=nn.Linear(64,5)

    def forward(self, x): #(3,128,128)
        x=self.conv1(x)
        x=torch.relu(x)
        x=self.pool(x) #(8,64,64)
        x=self.conv2(x)
        x=torch.relu(x)
        x=self.pool(x) #(16,32,32)
        x=self.conv3(x)
        x=torch.relu(x)
        x=self.pool(x) #(32,16,16)
        x=self.conv4(x)
        x=torch.relu(x)
        x=self.pool(x) #(64,8,8)
        x=self.conv5(x)
        x=torch.relu(x)
        x=self.pool(x) #(128,4,4)

        x=x.view(-1, 4*4*128)
        x=self.fc1(x)
        x=self.fc2(x)
        x=self.fc3(x)
        x=torch.softmax(x, dim=1)
        return x
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 모델 작성/ 최적화 설정/ 학습

```
model =CNN().to(device)
print(model)
optimizer=torch.optim.Adam(model.parameters(),lr=0.001)
criterion=nn.CrossEntropyLoss()
```

```
def train(model, train_loader, optimizer, interval):
    model.train() #모델의 훈련 mode로 설정

    for batch_index,(image, label) in enumerate(train_loader):
        image=image.to(device)
        label=label.to(device)

        optimizer.zero_grad()
        y_hat=model(image)
        loss=criterion(y_hat, label)
        loss.backward()
        optimizer.step()

        if batch_index % interval==0:
            print('train Epoch:{}, Train Loss:{}'.format(epoch, loss.item()))
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 모델 평가/ 학습

```
def evaluate(model, test_loader):
    model.eval() #모델을 평가모드로 설정
    test_loss=0
    correct=0

    with torch.no_grad():
        for image, label in test_loader:
            image=image.to(device)
            label=label.to(device)
            y_hat=model(image)
            test_loss+=criterion(y_hat, label).item()
            predition=y_hat.argmax(dim=1)
            correct+=(predition==label).sum()

    test_loss=test_loss/len(test_loader.dataset)
    test_accuracy=correct/len(test_loader.dataset)*100
    return test_loss, test_accuracy
```

```
epochs=200
for epoch in range(1, epochs+1):
    train(model, train_loader, optimizer, interval=100)
    test_loss, test_accuracy=evaluate(model, test_loader)
    print('epoch:{}, test loss:{}, test accuracy:{}'
          .format(epoch, test_loss, test_accuracy))
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 모델 저장/ 저장된 모델 로드/ 예측/ 이미지 시각화

```
path='model.pt'
torch.save(model.state_dict(), path)
```

```
model2=CNN().to(device)
model2.load_state_dict(torch.load(path))
model2.eval()
```

```
predict=model2(x_test[1].to(device)).argmax(dim=1)
print(predict.item(), y_test[1].item())
```

```
plt.figure(figsize=(3, 3))
# permute로 이미지의 shape를 다음과 같이 변경합니다
# (height, width, channel)
plt.imshow(x_test[1].permute(1, 2, 0))
plt.axis('off')
plt.show()
```

4. CNN 모델 실습 : 저장된 모델 활용하기

❖ 새로운 노트북(ipynb) 파일을 작성하여 새로운 이미지를 사용하여 분류하기

```
import torch
import torch.nn as nn
import numpy as np
from PIL import Image
from torchvision import transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader, TensorDataset
from torchvision.utils import save_image
import matplotlib.pyplot as plt
```

```
torch.manual_seed(777)
# 이미지 크기를 128 x 128 로 조정합니다
IMAGE_SIZE = 128
device = 'cuda' if torch.cuda.is_available() else 'cpu'
# GPU 사용 가능일 경우 랜덤 시드 고정
if device == 'cuda':
  torch.cuda.manual_seed_all(777)
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 예측을 위한 데이터 저장/ 이미지 데이터 셋/로더/로더로부터 이미지, label 추출

- 이미지 저장 폴더 : test_image/test 폴더에 분류에 사용할 이미지 저장

```
test_dataset=ImageFolder(root='test_image/',
                         transform=transforms.Compose([
                             transforms.Resize((IMAGE_SIZE,IMAGE_SIZE)),
                             transforms.ToTensor()
                         ]))
```

```
test_loader=DataLoader(test_dataset,
                       batch_size=10,
                       shuffle=False,
                       num_workers=1)
```

```
test_images, labels=next(iter(test_loader))
print(test_images.size(), labels.size())
```

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 새로운 테스트 이미지 출력

```
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
import matplotlib.pyplot as plt

plt.figure(figsize=(16,4))
for i in range(10):
    plt.subplot(1,10, i+1)
    plt.imshow(test_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

CNN 모델 정의를 위한 클래스 작성 : 생략

4. CNN 모델 실습 : 사용자 이미지 데이터 셋(flower)

❖ 모델 객체 생성 및 저장된 모델 로드/ 이미지 분류/ 테스트이미지 출력하기

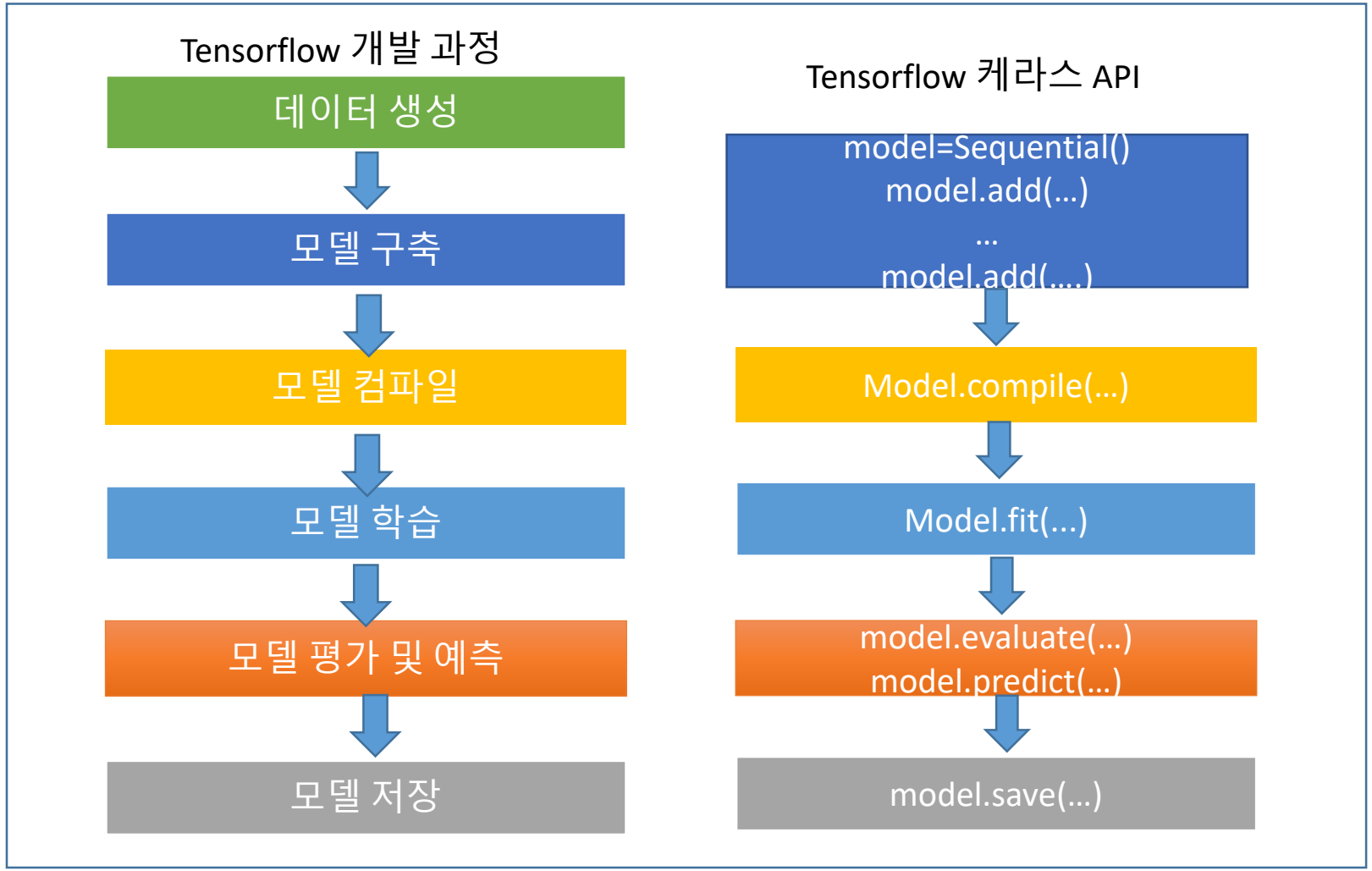
```
model2=CNN().to(device)
model2.load_state_dict(torch.load('model.pt'))
model2.eval()
```

```
predict=model2(test_images.to(device)).argmax(dim=1)
print(predict)
```

```
plt.figure(figsize=(16,4))
for i in range(10):
    plt.subplot(1,10, i+1)
    plt.imshow(test_images[i].permute(1,2,0))
    plt.axis('off')
plt.show()
```

4. CNN 모델 실습 : Tensorflow

❖ Tensorflow



4. CNN 모델 실습 : Tensorflow

❖ MNIST dataset 활용

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

plt.rc('font', family='Malgun Gothic')  # 또는 'AppleGothic', 'NanumGothic'
plt.rcParams['axes.unicode_minus'] = False  # 마이너스 부호 깨짐 방지


# 1. 데이터 로딩 및 전처리
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# CNN 입력을 위해 (28, 28, 1) 형태로 reshape
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0

# One-hot 인코딩
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

4. CNN 모델 실습 : Tensorflow

❖ MNIST dataset 활용-2

```
# 2. CNN 모델 구성
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')  # 출력층: 클래스 10개
])

# 3. 모델 컴파일
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 4. 모델 학습
model.fit(x_train, y_train, epochs=5, batch_size=64, validation_split=0.1)
```

4. CNN 모델 실습 : Tensorflow

❖ MNIST dataset 활용3

```
# 5. 모델 평가
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'\n 테스트 정확도: {test_acc:.4f}')

# 6. 예측 및 시각화
def predict_and_plot(index):
    img = x_test[index].reshape(1, 28, 28, 1)
    prediction = model.predict(img)
    predicted_label = np.argmax(prediction)

    plt.imshow(x_test[index].reshape(28, 28), cmap='gray')
    plt.title(f'예측: {predicted_label}')
    plt.axis('off')
    plt.show()

# 예: 0번째 이미지 예측 및 시각화
predict_and_plot(0)
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-1

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import os
import matplotlib as mpl
from tensorflow.keras.callbacks import EarlyStopping

# ✅ 한글 폰트 설정 (그래프 제목 등에서 깨짐 방지)
mpl.rc('font', family='Malgun Gothic')  # 또는 'AppleGothic', 'NanumGothic'
plt.rcParams['axes.unicode_minus'] = False

# ✅ 경로 설정: 이미지 폴더 경로
data_dir = "flower_photos"  # flower/ 안에 클래스별 폴더 존재

# ✅ 데이터셋 설정(이미지 크기, 배치 사이즈, 시드)
img_height, img_width = 128, 128
batch_size = 32
seed = 123
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-2

```
# ✅ 데이터셋 설정(train data/validation data)

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=seed,
    image_size=(img_height, img_width),
    batch_size=batch_size
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=seed,
    image_size=(img_height, img_width),
    batch_size=batch_size
)
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-3

```
# ✅ 클래스 이름 확인
class_names = train_ds.class_names
print("클래스:", class_names)

# ✅ 성능 최적화를 위한 데이터셋 캐싱
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

# ✅데이터 증강(Data Augmentation)
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),
])
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-4

```
# ✅ CNN 모델 구성
model = tf.keras.Sequential([
    data_augmentation,

    tf.keras.layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),

    tf.keras.layers.Conv2D(32, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Conv2D(64, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Conv2D(128, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(len(class_names), activation='softmax')
])
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-5

```
# ✅ 모델 컴파일
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# ✅ 학습
epochs = 10

#EarlyStopping 콜백으로 과적합 방지
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
history = model.fit(train_ds, validation_data=val_ds, epochs=epochs,callbacks=[early_stopping])

# 모델 저장
model.save("flower_classifier_model.h5")
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-6

```
# ✅ 정확도 시각화
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.plot(range(len(acc)), acc, label='학습 정확도')
plt.plot(range(len(val_acc)), val_acc, label='검증 정확도')
plt.title('정확도 변화')
plt.xlabel('에포크')
plt.ylabel('정확도')
plt.legend()
plt.grid(True)
plt.show()
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-7

```
# ✅ 예측 함수 정의
def predict_from_dataset(dataset, class_names, num_images=5):
    for images, labels in dataset.take(1):
        predictions = model.predict(images)

        for i in range(num_images):
            plt.imshow(images[i].numpy().astype("uint8"))
            pred_label = class_names[np.argmax(predictions[i])]
            true_label = class_names[labels[i]]

            plt.title(f"예측: {pred_label} / 실제: {true_label}")
            plt.axis("off")
            plt.show()

# ✅ 예측 결과 시각화
predict_from_dataset(val_ds, class_names, num_images=5)
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-8

```
# 저장된 모델 불러오기
loaded_model = tf.keras.models.load_model("flower_classifier_model.h5")

# ✅ 저장된 모델을 사용한 예측 함수
def predict_with_loaded_model(dataset, class_names, model, num_images=5):
    for images, labels in dataset.take(1):
        predictions = model.predict(images)
        for i in range(num_images):
            plt.imshow(images[i].numpy().astype("uint8"))
            pred_label = class_names[np.argmax(predictions[i])]
            true_label = class_names[labels[i]]
            plt.title(f"[저장 모델] 예측: {pred_label} / 실제: {true_label}")
            plt.axis("off")
            plt.show()

# ✅ 예측 실행
predict_with_loaded_model(val_ds, class_names, loaded_model, num_images=5)
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower)-9

```
# Confusion Matrix 추가 (정밀도, 재현율 확인용)
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

y_true, y_pred = [], []
for images, labels in val_ds.unbatch():
    preds = model.predict(tf.expand_dims(images, axis=0))
    y_pred.append(np.argmax(preds))
    y_true.append(labels.numpy())

cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('예측 클래스')
plt.ylabel('실제 클래스')
plt.title('Confusion Matrix')
plt.show()
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower): (새파일에서 저장된 모델 로더하여 예측)-1

```
# predict_custom_image.py

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import load_img, img_to_array
import matplotlib.font_manager as fm

plt.rc('font', family='Malgun Gothic')  # 또는 'AppleGothic', 'NanumGothic'
plt.rcParams['axes.unicode_minus'] = False  # 마이너스 부호 깨짐 방지

# 모델 및 클래스 이름 불러오기
model = tf.keras.models.load_model("flower_classifier_model.h5")
class_names =['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
#np.load("class_names.npy").tolist()

# 이미지 사이즈
img_height = 128
img_width = 128
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower): (새파일에서 저장된 모델 로더하여 예측)-2

```
# 예측 함수
def predict_custom_image(img_path):
    img = load_img(img_path, target_size=(img_height, img_width))
    img_array = img_to_array(img)  # ! 정규화 하지 마세요
    img_array = tf.expand_dims(img_array, 0)

    prediction = model.predict(img_array)[0]
    predicted_class = class_names[np.argmax(prediction)]
    confidence = np.max(prediction)

    # 시각화
    plt.imshow(img)
    plt.title(f"예측: {predicted_class} ({confidence*100:.2f}%)")
    plt.axis("off")
    plt.show()

    print("\n📊 클래스별 확률:")
    for cls, prob in zip(class_names, prediction):
        print(f"{cls}: {prob*100:.2f}%")
```

4. CNN 모델 실습 : Tensorflow

❖ 사용자 이미지 활용(Flower): (새파일에서 저장된 모델 로더하여 예측)-1

```
# 1개 테스트 이미지 예측
predict_custom_image('test_image/test/01.jpg')  # 여기에 테스트 이미지 경로 지정
```

```
# 폴더내의 이미지 모두 예측하기
test_folder='test_image/test'

for filename in os.listdir(test_folder):
        img_path = os.path.join(test_folder, filename)
        predict_custom_image(img_path)
```