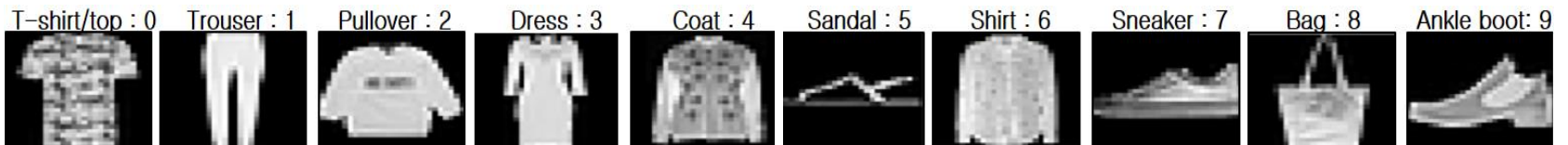
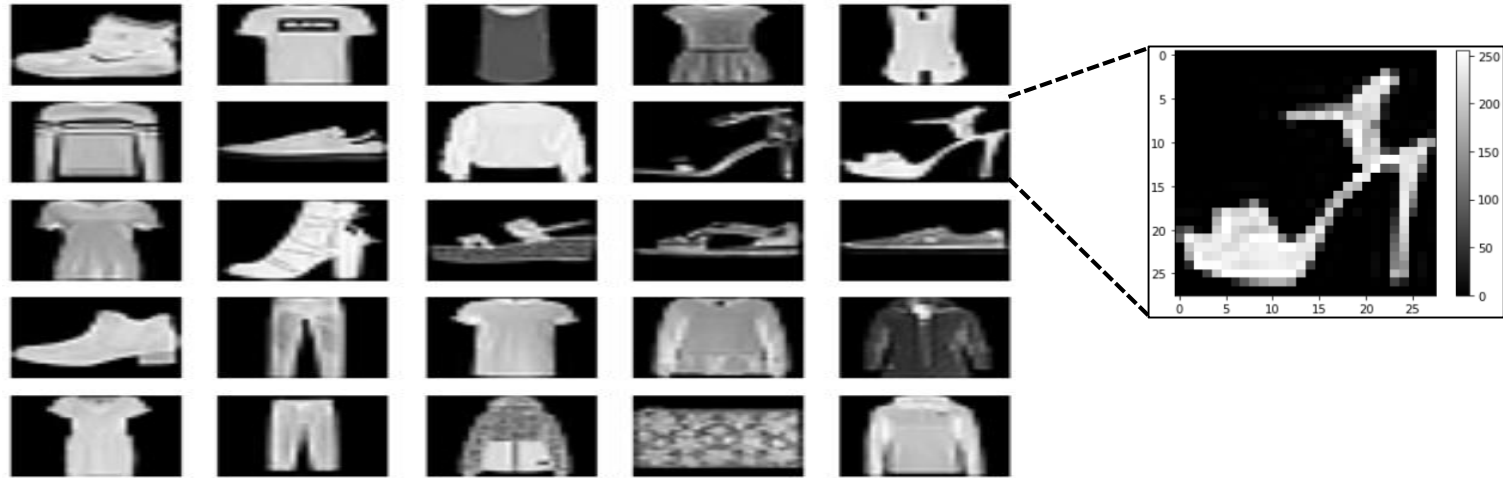


Fashion MNIST Example

<https://youtu.be/AyvicBsP8tE>



Fashion MNIST는 MNIST에 영향을 받아서 만들어진 데이터이며, 손 글씨가 아닌 옷과 신발 등의 흑백 이미지로서 MNIST보다는 좀 더 어려운 문제로 평가 되고 있음

MNIST Example 에서 다음과 같이 변경하여 정확도와 오버피팅을 MNIST와 비교하시오

```
from tensorflow.keras.datasets import fashion_mnist
```

```
fashion_mnist.load_data()
```

loss 값은 증가하는데, 정확도는 일정하게 유지되는 현상을 설명하시오



– Flatten / Dense / Dropout / BatchNormalization –

박성호 (neowizard2018@gmail.com)

TensorFlow 2.x Layer API	Description
<code>tf.keras.layers.Flatten()</code>	입력데이터(텐서)를 1차원 vector로 만들어주는 역할을 수행함
<code>tf.keras.layers.Dense(100, activation='relu')</code> <code>tf.keras.layers.Dense(100, activation='sigmoid')</code> <code>tf.keras.layers.Dense(10, activation='softmax')</code>	1 st 파라미터는 출력 노드 수이며, 활성화 함수는 <code>activation='...'</code> 나타냄
<code>tf.keras.layers.Dropout(rate=0.2)</code>	rate에 지정된 비율만큼 랜덤하게 층과 층 사이의 연결을 끊어서 네트워크의 overfitting을 막는 역할 수행
<code>tf.keras.layers.BatchNormalization()</code>	데이터 평균을 0, 표준편차를 1로 분포 시킴. 높은 학습율을 사용하여 빠른 속도로 학습하면서 overfitting을 줄이는 효과가 있다고 알려져 있음(?)

```
model = Sequential()          # model 생성
model.add(Flatten(input_shape=(28, 28, 1)))
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.25))      # Dropout() 추가
model.add(Dense(10, activation='softmax'))
```

[Self Study 1]

MNIST 에 대해서 SGD(), loss='sparse_categorical_entropy', epochs=50 환경에서 Dropout(..) 비율을 0.25, 0.5, 0.75 등으로 변경하여 오버피팅을 확인하시오

[Self Study 2]

MNIST 에 대해서 Dropout(...) 아닌 BatchNormalization() 레이어를 이용하여 후 학습하시오



– TensorFlow Callback –

박성호 (neowizard2018@gmail.com)

TensorFlow Callback Function <https://youtu.be/5cmPhp0Kz9s>

- TensorFlow 콜백(callback)은 모델의 학습 방향, 저장 시점, 학습 정지 시점 등에 관한 상황을 모니터링 하기 위해 주로 사용됨.
 - 즉 모델이 학습을 시작하면 학습이 완료될 때까지 사람이 할 수 있는게 없음. 따라서 이를 해결하고자 존재하는 것이 콜백 함수임.
 - 예를 들어, 학습 도중에 학습율(learning rate)을 변화시키거나 val_loss가 개선되지 않으면 학습 도중에 학습을 멈추게 하는 등의 작업을 할 수 있음
- TensorFlow 에서 사용되는 대표적인 콜백 함수는
ReduceLROnPlateau, ModelCheckpoint, EarlyStopping 등이 있음

ModelCheckpoint

- 모델이 학습하면서 정의한 조건을 만족했을 때 Model의 weight 값을 중간 저장함.
- 학습시간이 오래 걸린다면, 모델이 개선된 validation score를 도출해낼 때마다 weight를 중간 저장함으로써, 혹시 중간에 memory overflow나 crash가 나더라도 다시 weight를 불러와서 학습을 이어나갈 수 있기 때문에, 시간을 save해 줄 수 있음.

```
from tensorflow.keras.callbacks import ModelCheckpoint

file_path = './modelchpoint_test.h5'           # 저장할 file path

checkpoint = ModelCheckpoint(file_path,         # 저장할 file path
                             monitor='val_loss', # val_loss 값이 개선되었을때 호출
                             verbose=1,          # log 출력
                             save_best_only=True, # best 값만 저장
                             mode='auto')        # auto는 자동으로 best를 찾음

hist = model.fit(x_train, t_train,
                 epochs=50, validation_split=0.2,
                 callbacks=[checkpoint])
```

tensorflow.org 에서 API 사용되는 default parameter 확인해 볼것

EarlyStopping

- 모델 성능 지표가 설정한 epoch동안 개선되지 않을 때 조기 종료할 수 있음.

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

file_path = './modelchpoint_test.h5'           # 저장할 file path

checkpoint = ModelCheckpoint(file_path,         # 저장할 file path
                             monitor='val_loss', # val_loss 값이 개선되었을때 호출
                             verbose=1,         # log 출력
                             save_best_only=True, # best 값만 저장
                             mode='auto')       # auto는 자동으로 best를 찾음

stopping = EarlyStopping(monitor='val_loss',    # 관찰대상은 val_loss
                         patience=5)           # 5 epoch 동안 개선되지 않으면 조기종료

hist = model.fit(x_train, t_train,
                 epochs=50, validation_split=0.2,
                 callbacks=[checkpoint, stopping])
```

tensorflow.org 에서 API 사용되는 default parameter 확인해 볼것

Callback Example

은닉층은 100개 노드를 가지는 신경망을 가지고서 FashionMNIST 데이터를 학습하는 경우,
다음 조건을 만족하는 코드를 각각 구현하고 결과를 확인 하시오

[1] SGD(learning_rate=0.1), loss='sparse_categorical_crossentropy', epochs=50,
val_loss 에 대해서 ModelCheckpoint 콜백 구현

[2] SGD(learning_rate=0.1), loss='sparse_categorical_crossentropy', epochs=50,
val_loss 에 대해서 ModelCheckpoint, EarlyStopping 콜백 구현

머신러닝/딥러닝을 위한

CNN (I)

- 컨볼루션(Convolution) 개념 -

박성호 (neowizard2018@gmail.com)

[Insight] 컨볼루션 (Convolution)

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$



적분변수 τ 를 친숙한 x 로 바꾸고
대칭이동도 하지 않음

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(x)g(x - t)dx$$


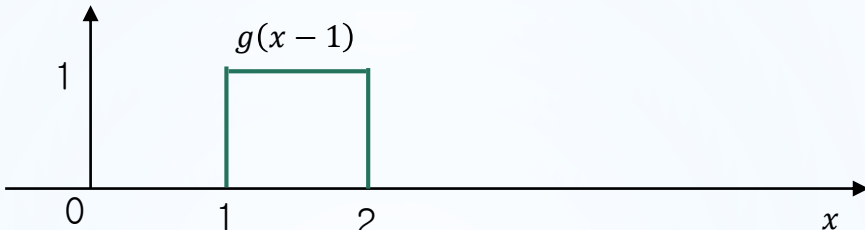
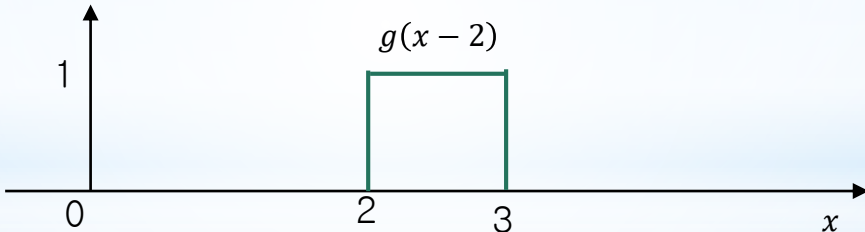

[insight 1] ① time shift ② data variation ③ average

[insight 2] 컨볼루션은 시간 t 함수.

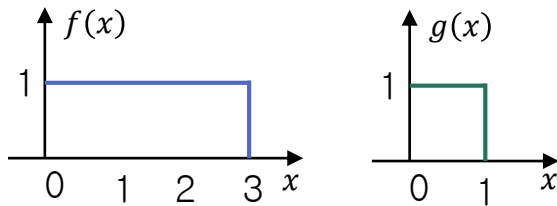
[예] $f(x) = x$, $g(x-t) = x-t$ 라고 가정하면 컨볼루션 결과는 시간 t 함수임

참고 및 복습 영상: https://youtu.be/63Y4tP_soXc

time shift

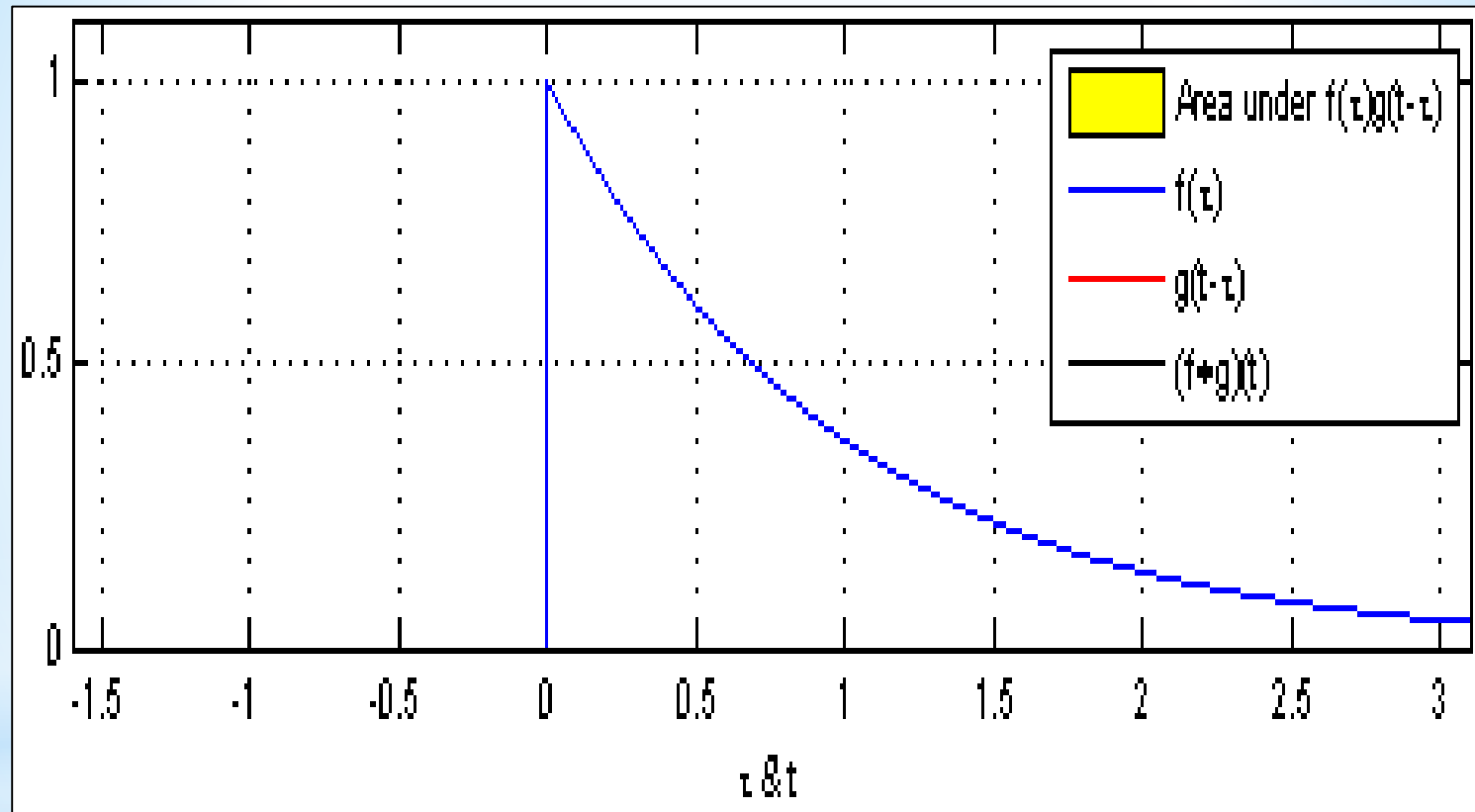
$g(x)$	 <p>A graph of the function $g(x)$ on a Cartesian coordinate system. The horizontal axis is labeled x and has tick marks at 0 and 1. The vertical axis has a tick mark at 1. The function $g(x)$ is represented by a green rectangle starting at $x=0$ and ending at $x=1$, with a constant height of 1. The label $g(x)$ is placed above the rectangle.</p>
$g(x - 1)$	 <p>A graph of the function $g(x - 1)$ on a Cartesian coordinate system. The horizontal axis is labeled x and has tick marks at 0, 1, and 2. The vertical axis has a tick mark at 1. The function $g(x - 1)$ is represented by a green rectangle starting at $x=1$ and ending at $x=2$, with a constant height of 1. The label $g(x - 1)$ is placed above the rectangle.</p>
$g(x - 2)$	 <p>A graph of the function $g(x - 2)$ on a Cartesian coordinate system. The horizontal axis is labeled x and has tick marks at 0, 2, and 3. The vertical axis has a tick mark at 1. The function $g(x - 2)$ is represented by a green rectangle starting at $x=2$ and ending at $x=3$, with a constant height of 1. The label $g(x - 2)$ is placed above the rectangle.</p>
$g(x - t)$	 <p>A graph of the function $g(x - t)$ on a Cartesian coordinate system. The horizontal axis is labeled x and has tick marks at 0, t, and $t+1$. The vertical axis has a tick mark at 1. The function $g(x - t)$ is represented by a green rectangle starting at $x=t$ and ending at $x=t+1$, with a constant height of 1. The label $g(x - t)$ is placed above the rectangle.</p>

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(x)g(x - t)dx$$



$f(x)$					$g(x)$		
1	2	3	0	*	2	0	1
0	1	2	3		0	1	2
3	0	1	2		1	0	2
2	3	0	1				

시간 t	연속함수 $f(t) * g(t)$	계산 결과	이산데이터 $f(t) * g(t)$	계산 결과																									
t = 0			<table><tr><td>1</td><td>2</td><td>3</td><td>0</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>3</td><td>0</td><td>1</td><td>2</td></tr><tr><td>2</td><td>3</td><td>0</td><td>1</td></tr></table> *	1	2	3	0	0	1	2	3	3	0	1	2	2	3	0	1	<table><tr><td>2</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>2</td></tr></table>	2	0	1	0	1	2	1	0	2
1	2	3	0																										
0	1	2	3																										
3	0	1	2																										
2	3	0	1																										
2	0	1																											
0	1	2																											
1	0	2																											
t = 1			<table><tr><td>1</td><td>2</td><td>3</td><td>0</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>3</td><td>0</td><td>1</td><td>2</td></tr><tr><td>2</td><td>3</td><td>0</td><td>1</td></tr></table> *	1	2	3	0	0	1	2	3	3	0	1	2	2	3	0	1	<table><tr><td>2</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>2</td></tr></table>	2	0	1	0	1	2	1	0	2
1	2	3	0																										
0	1	2	3																										
3	0	1	2																										
2	3	0	1																										
2	0	1																											
0	1	2																											
1	0	2																											
t = 2			<table><tr><td>1</td><td>2</td><td>3</td><td>0</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>3</td><td>0</td><td>1</td><td>2</td></tr><tr><td>2</td><td>3</td><td>0</td><td>1</td></tr></table> *	1	2	3	0	0	1	2	3	3	0	1	2	2	3	0	1	<table><tr><td>2</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>2</td></tr></table>	2	0	1	0	1	2	1	0	2
1	2	3	0																										
0	1	2	3																										
3	0	1	2																										
2	3	0	1																										
2	0	1																											
0	1	2																											
1	0	2																											



※ 출처 : <https://en.wikipedia.org/wiki/Convolution>

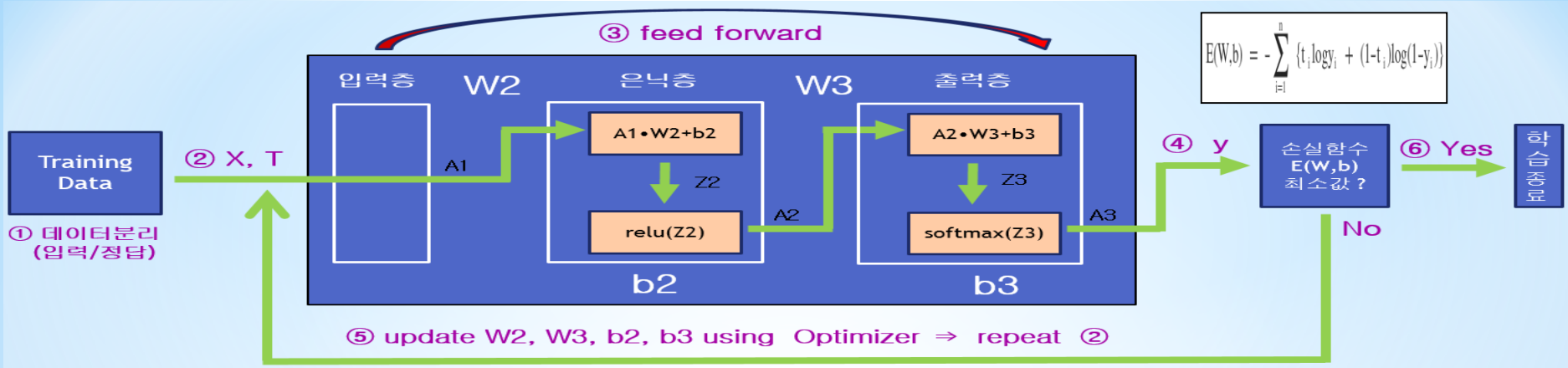


컨볼루션 연산 • 풀링 • 패딩

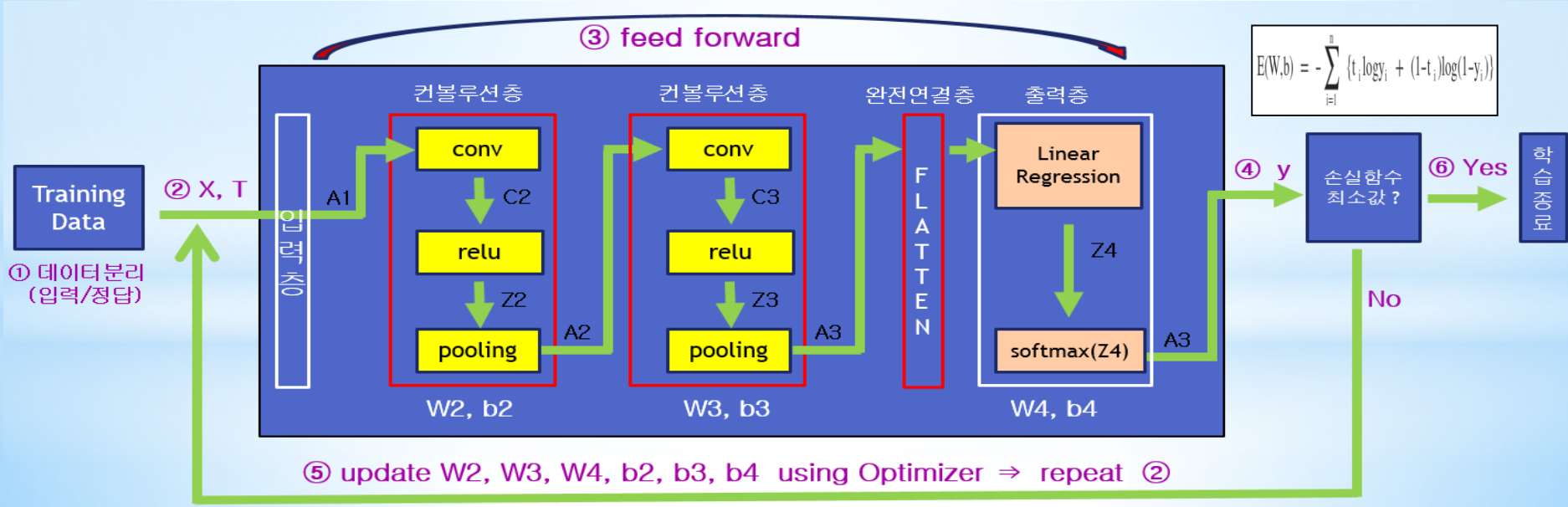
(참고 학습 영상: https://youtu.be/1_70qe1XBV8)

박성호 (neowizard2018@gmail.com)

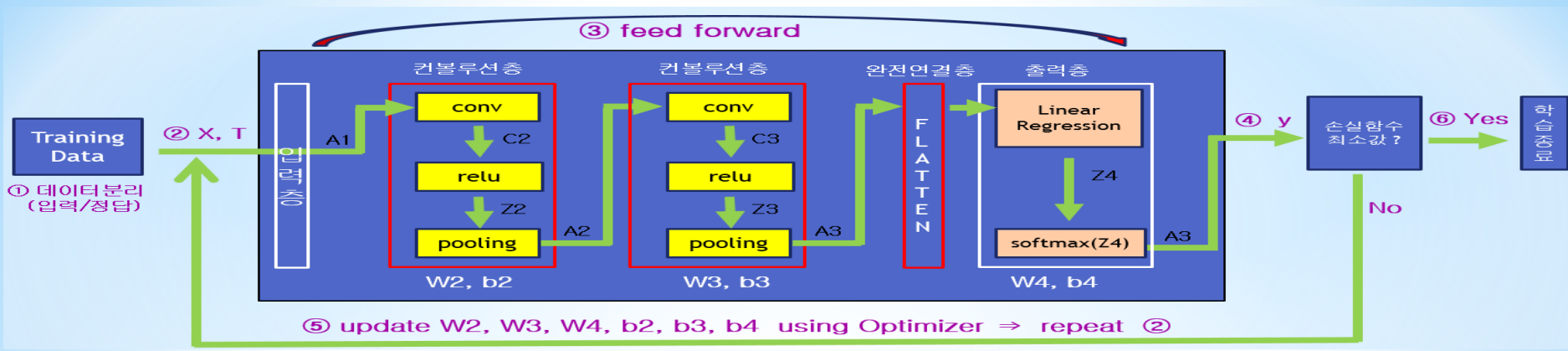
Overview – 아키텍처 비교 (NN vs. CNN)



↓ CNN 아키텍처로 변경



컨볼루션층 개요 - conv / pooling

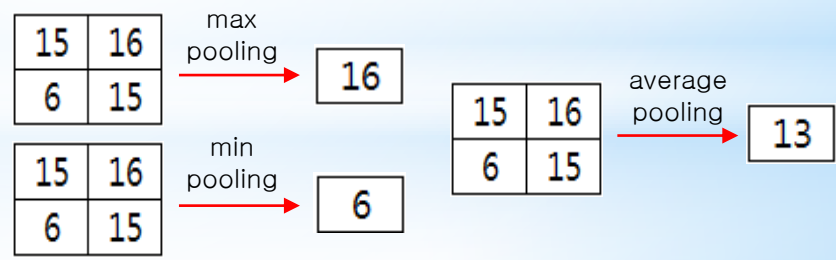


➤ conv (컨볼루션, convolution)

- 입력데이터(A1, A2...)와 가중치들의 집합체인 다양한 필터(filter)와의 컨볼루션 연산을 통해 **입력데이터의 특징(feature)을 추출**하는 역할을 수행함

➤ pooling (풀링)

- 입력 정보를 최대값 • 최소값 • 평균값 등으로 압축하여 데이터 연산량을 줄여주는 역할 수행



컨볼루션 (convolution) 연산 - 특징 추출 (특징 맵, feature map)

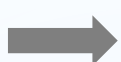
1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

입력데이터

⊛

2	0	1
0	1	2
1	0	2

필터
(가중치 집합체)



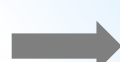
15	16
6	15

컨볼루션 연산 결과

+

3

바이어스



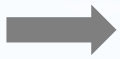
18	19
9	18

특징 맵(feature map)

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

⊛

2	0	1
0	1	2
1	0	2



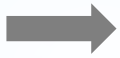
15	

스트라이드(필터의 이동간격)

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

⊛

2	0	1
0	1	2
1	0	2

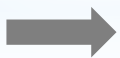


15	16

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

⊛

2	0	1
0	1	2
1	0	2



15	16
6	

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

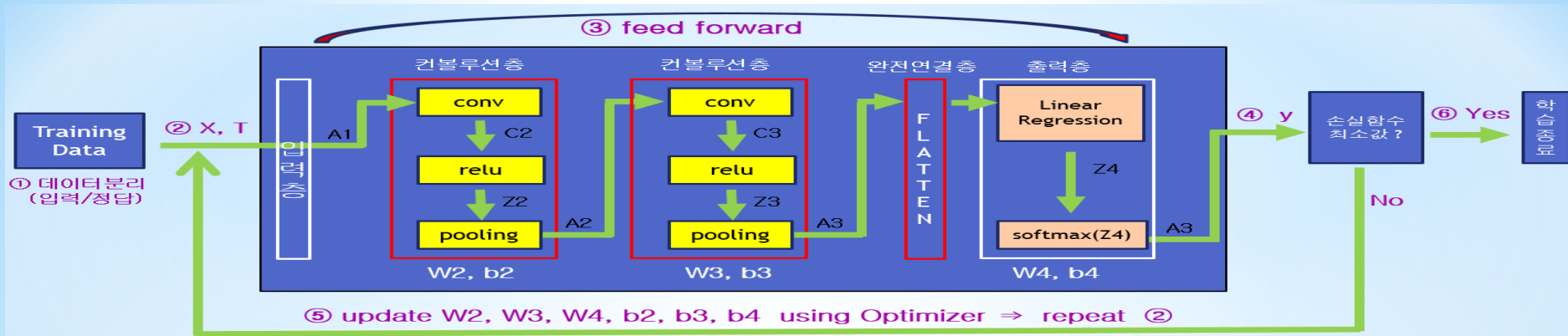
⊛

2	0	1
0	1	2
1	0	2

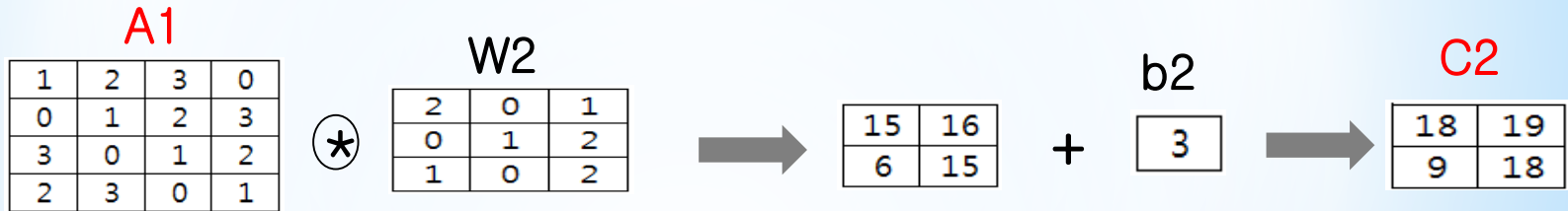


15	16
6	15

relu 연산 / pooling 연산



conv



relu



pooling



패딩 (padding)

➤ 패딩(padding)이란 컨볼루션 연산을 수행하기 전에 **입력 데이터 주변을 특정 값** (예를들면 0)으로 **채우는 것**을 말하며, 컨볼루션 연산에서 자주 이용되는 방법

- 컨볼루션 연산을 수행하면 데이터 크기(shape)이 줄어드는 단점을 방지하기 위해 사용

[패딩: 0] 원본 데이터 크기 (4x4) \Rightarrow 컨볼루션 결과 데이터 크기 (2x2)

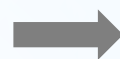
입력데이터 (4x4)

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

⊛

필터 (3x3)

2	0	1
0	1	2
1	0	2



결과 (2x2)

15	16
6	15

[패딩: 1] 원본 데이터 크기 (4x4) \Rightarrow 컨볼루션 결과 데이터 크기 (4x4)

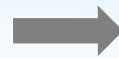
입력데이터 (6x6)

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

⊛

필터 (3x3)

2	0	1
0	1	2
1	0	2



결과 (4x4)

7	12	10	2
4	15	16	10
10	6	15	6
8	10	4	3

컨볼루션 연산을 통한 출력 데이터 크기(shape) 계산

입력 데이터 크기 (H, W), 필터 크기 (FH, FW), 패딩 P, 스트라이드 S 일 때 **출력 데이터 크기 (OH, OW)**

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

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

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

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

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

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

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

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

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

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

※ 소수점 아래는 버림하기 때문에, 입력데이터 크기에 맞게 필터크기 / 패딩 / 스트라이드 등의 설정을 알맞게 해야함

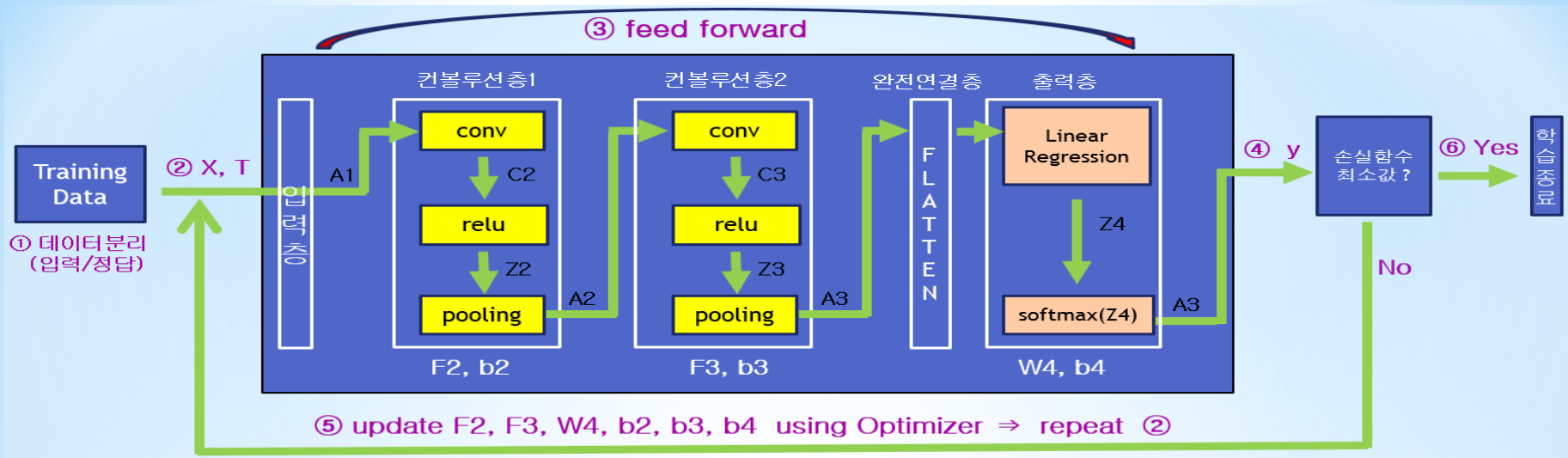


특징 추출 원리

(참고 학습 영상: <https://youtu.be/yUPRprrewhY>)

박성호 (neowizard2018@gmail.com)

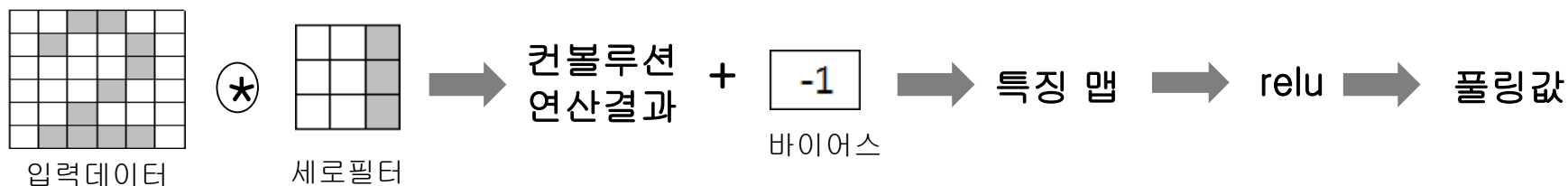
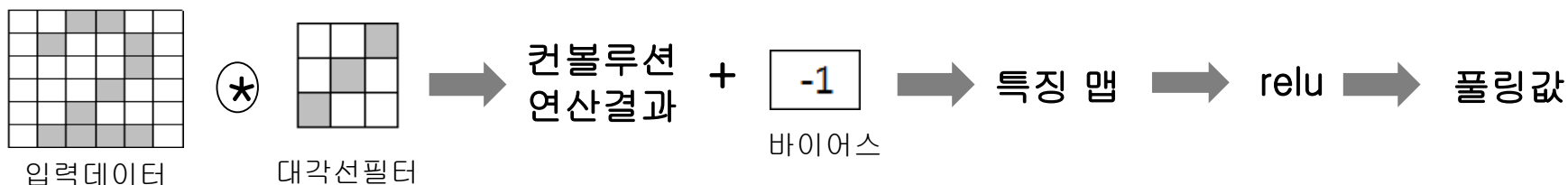
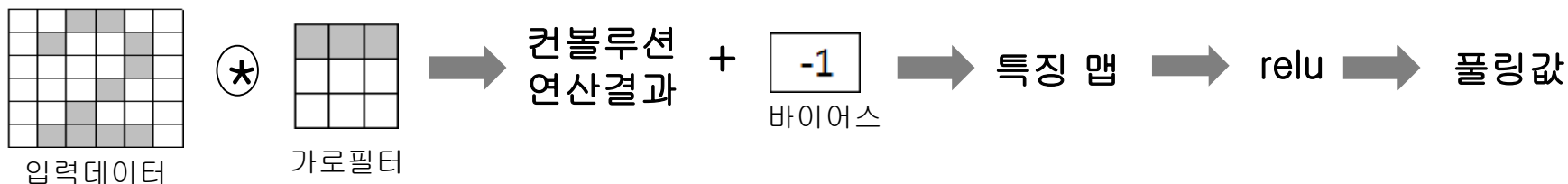
Review – 컨볼루션 층 (convolution layer) 역할



필터를 통해 데이터 특징을 추출 ?

특징 추출 과정 - 입력 데이터 1개 (숫자 2) / 필터 3개 (가로, 대각선, 세로)

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



[참고] 입력데이터와 필터

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

입력데이터 (숫자 2)

1	1	1
0	0	0
0	0	0

가로필터

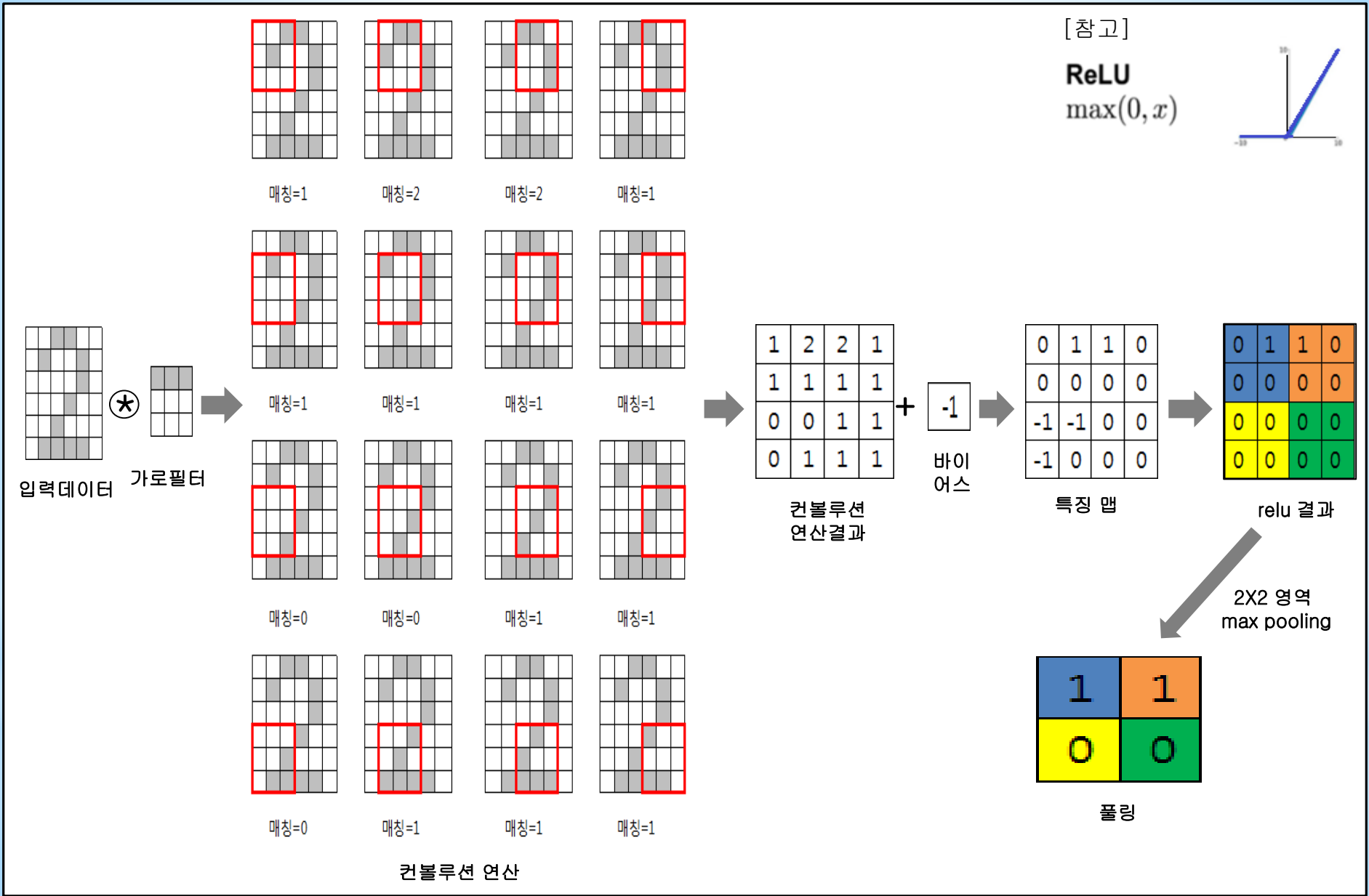
0	0	1
0	1	0
1	0	0

대각선필터

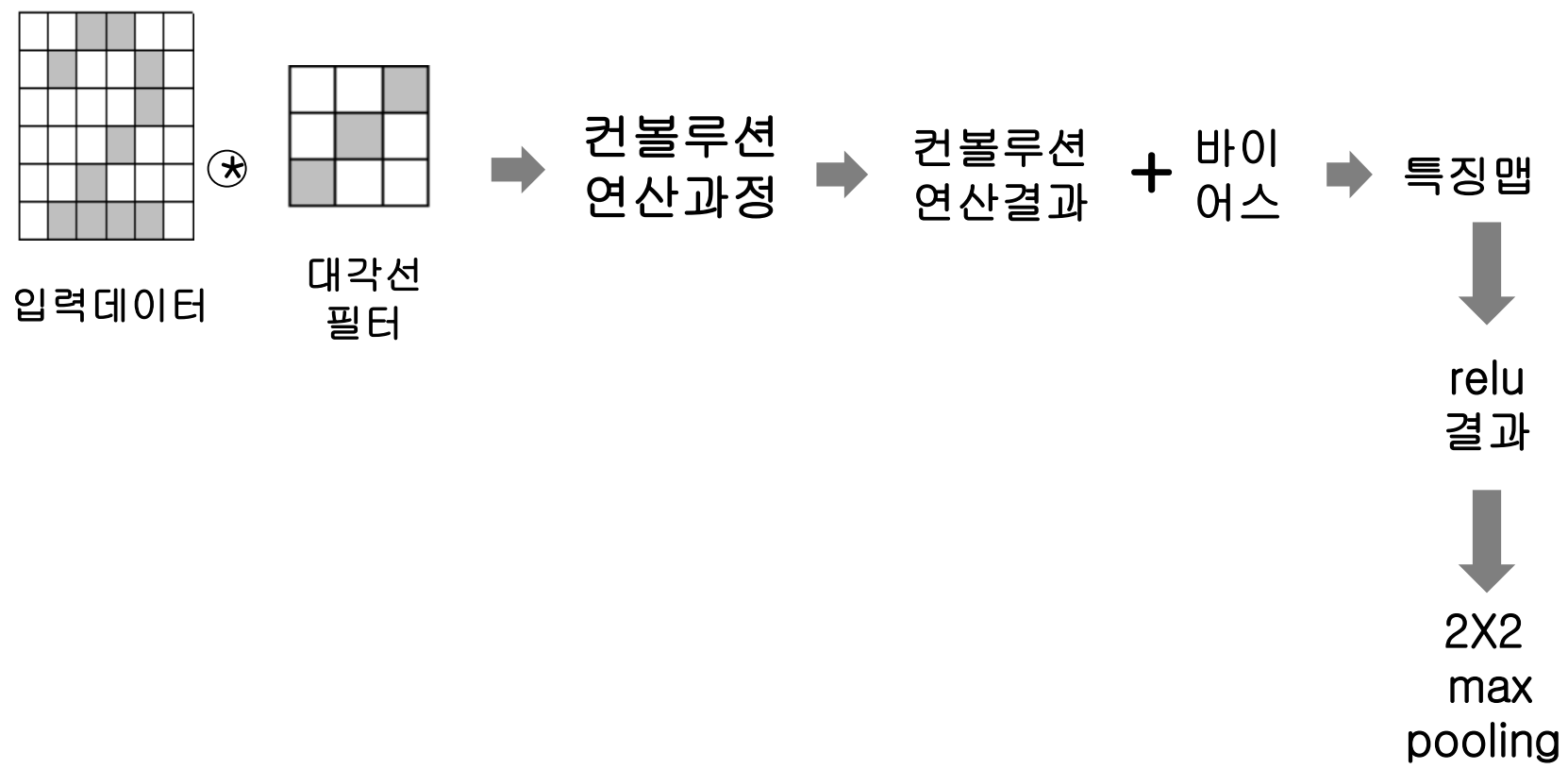
0	0	1
0	0	1
0	0	1

세로필터

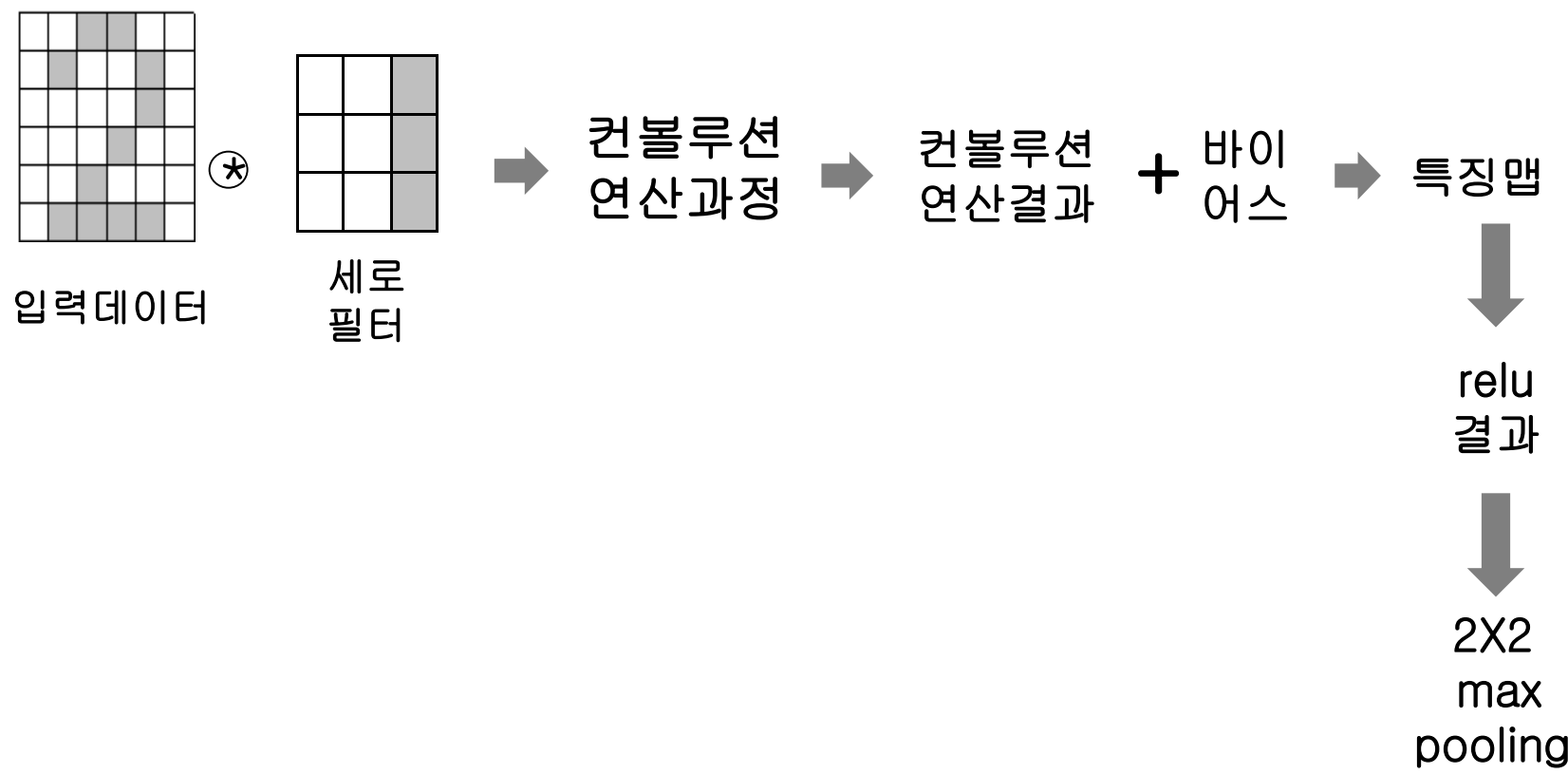
가로필터를 통한 입력 데이터 특징 추출 (스트라이드 1, 패딩 없음, 바이어스 -1)



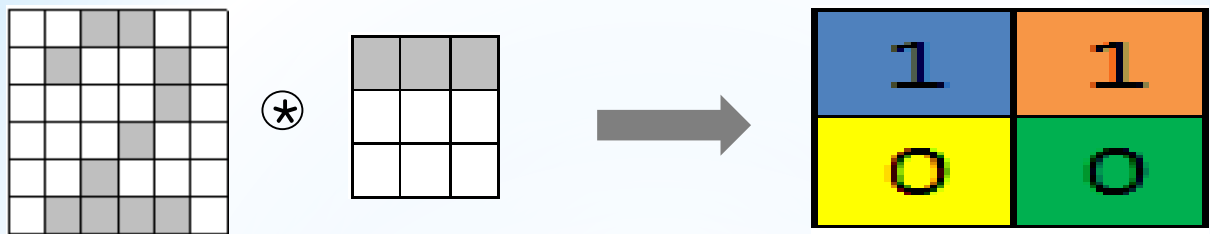
대각선필터를 통한 입력 데이터 특징 추출 (스트라이드 1, 패딩 없음, 바이어스 -1)



대각선필터를 통한 입력 데이터 특징 추출 (스트라이드 1, 패딩 없음, 바이어스 -1)

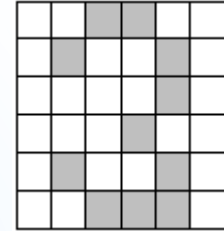


필터를 통한 입력 데이터 특징 추출 원리 - 특징 맵이 압축된 풀링 값

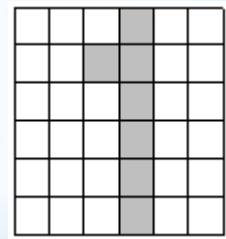


[예제 1] 필터를 통한 입력 데이터 특징 추출 (스트라이드 1, 패딩 없음, 바이어스 -1)

[1] 다음과 같은 숫자 3 에 적용할 수 있는 필터를 최소 5개 만들고,
그 가운데 특징(feature)를 가장 잘 추출 할 수 있는 필터를 검증하시오



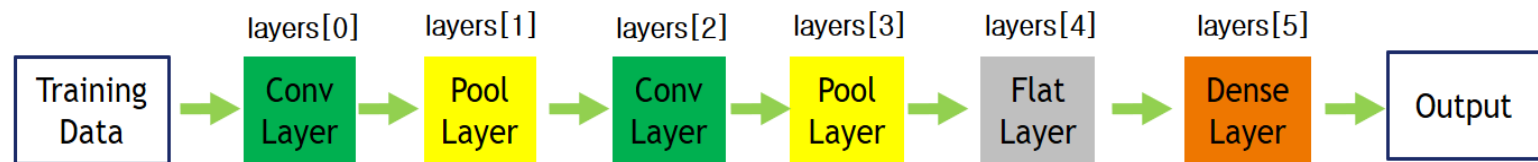
[2] 다음과 같은 숫자 1 에 적용할 수 있는 필터를 최소 5개 만들고,
그 가운데 특징(feature)을 가장 잘 추출 할 수 있는 필터를 검증하시오



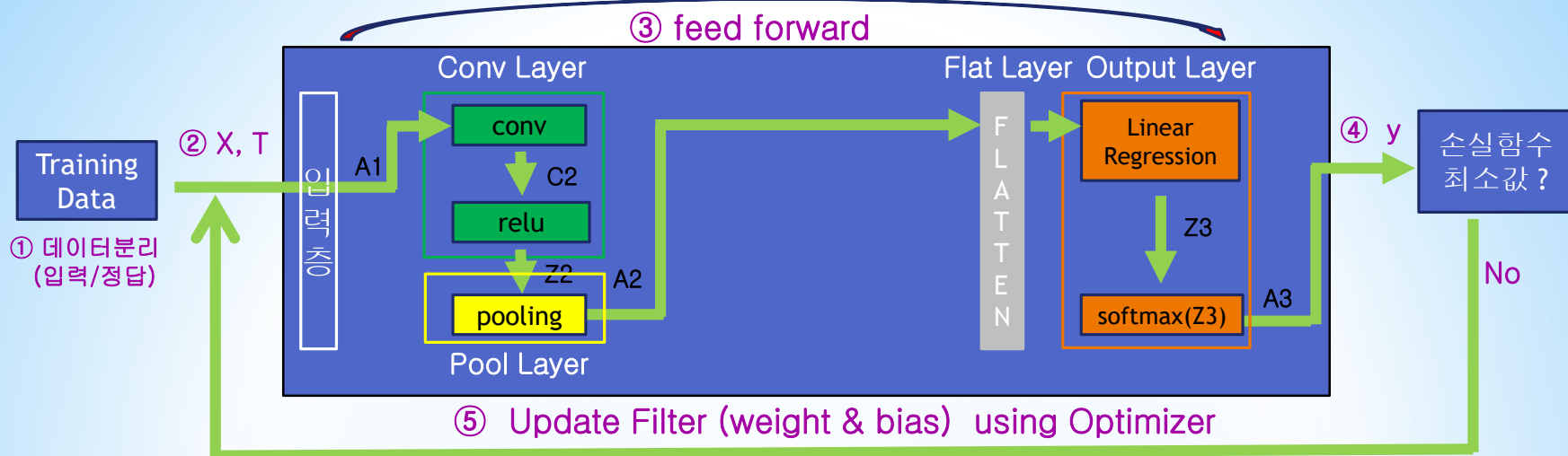


– CNN Basic Architecture (Conv / Pool / Flat) –

박성호 (neowizard2018@gmail.com)



Layer	TensorFlow 2.x Layer API
Conv	<code>Conv2D(input_shape=(28, 28, 1), kernel_size=3, filters=32, strides=(1, 1), activation='relu', use_bias=True, padding='SAME')</code>
	<code>Conv2D(kernel_size=3, filters=32, strides=(1, 1), activation='relu', use_bias=True, padding='SAME')</code>
Pool	<code>MaxPool2D(pool_size=(2, 2), padding='SAME')</code>
Flat	<code>Flatten()</code>
Dropout	<code>Dropout(rate=0.2)</code>
Dense	<code>Dense(10, activation='softmax')</code>



```
import tensorflow as tf

from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPool2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.datasets import mnist

import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt

print(tf.__version__)
```

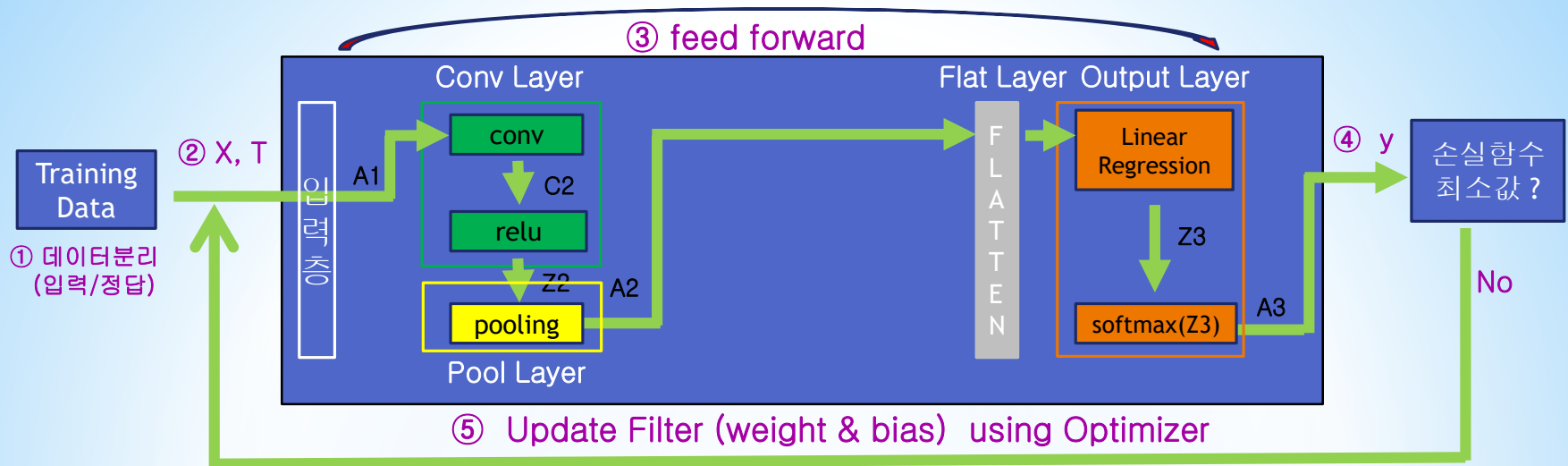
2.2.0

```
(x_train, t_train), (x_test, t_test) = mnist.load_data()

x_train = x_train / 255.0
x_test = x_test / 255.0

print('x_train.shape = ', x_train.shape, ' , x_test.shape = ', x_test.shape)
print('t_train.shape = ', t_train.shape, ' , t_test.shape = ', t_test.shape)

x_train.shape = (60000, 28, 28) , x_test.shape = (10000, 28, 28)
t_train.shape = (60000,) , t_test.shape = (10000,)
```



```

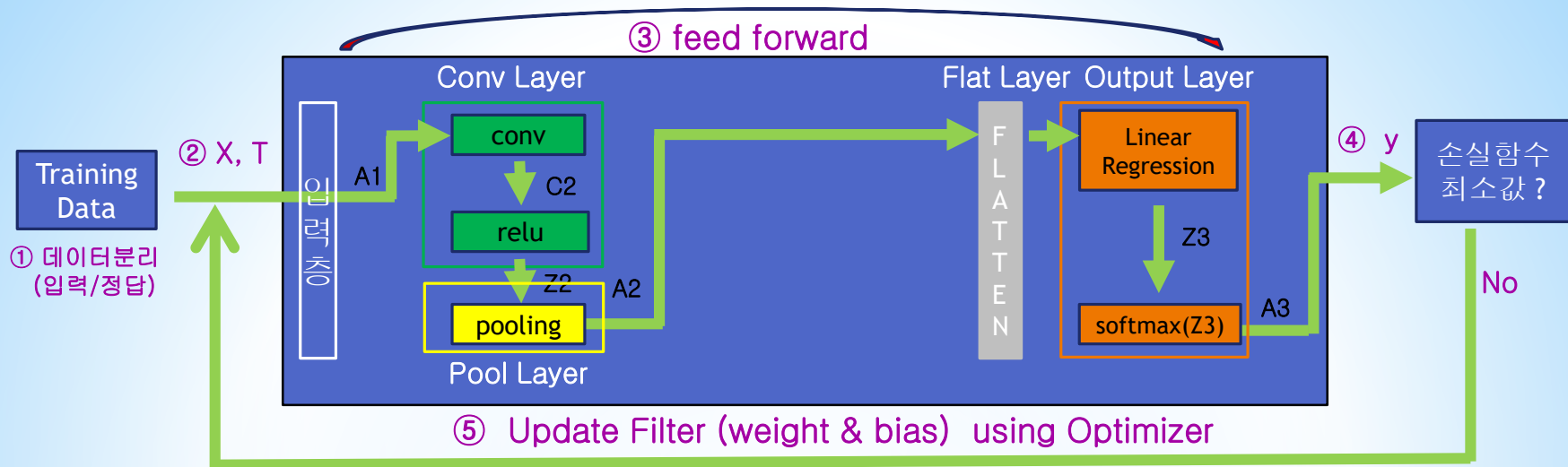
model = Sequential()

model.add(Conv2D(input_shape=(28,28,1),
                  kernel_size=3, filters=32,
                  strides=(1,1), activation='relu', use_bias=True, padding='SAME'))

model.add(MaxPool2D(pool_size=(2,2), padding='SAME'))

model.add(Flatten())

model.add(Dense(10, activation='softmax'))
  
```



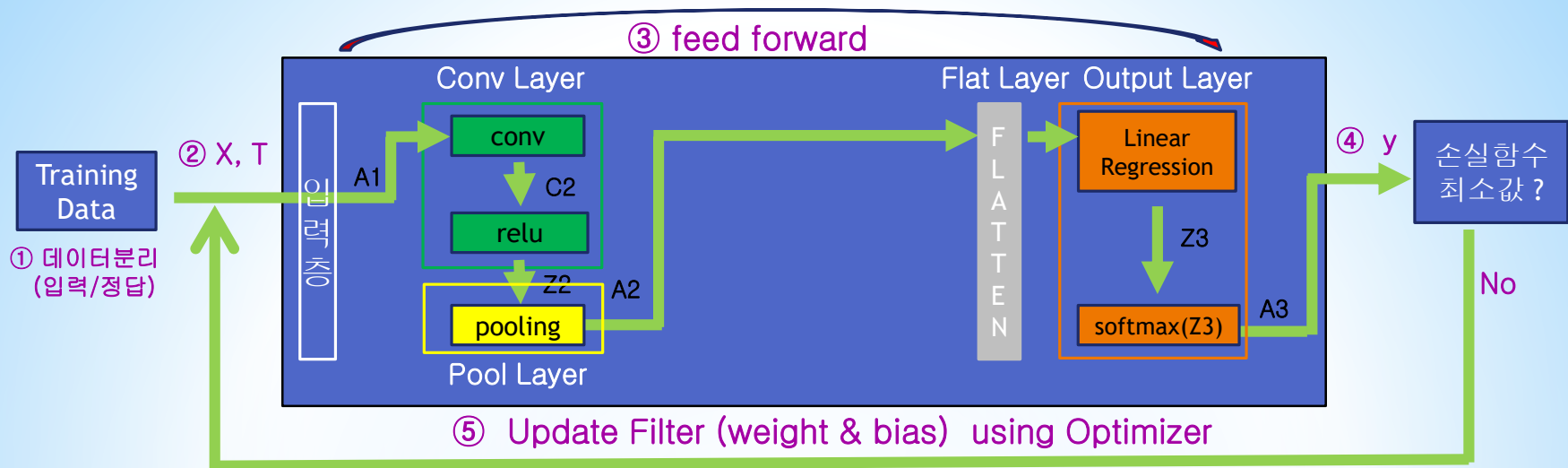
```
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 10)	62730

=====
 Total params: 63,050
 Trainable params: 63,050
 Non-trainable params: 0

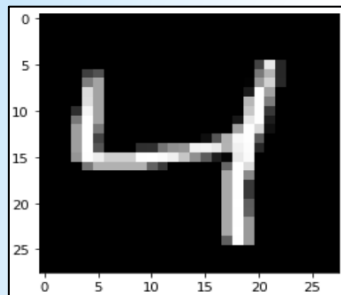


```
hist = model.fit(x_train.reshape(-1,28,28,1), t_train,
                 batch_size=50, epochs=50, validation_split=0.2)
```

```
model.evaluate(x_test.reshape(-1,28,28,1), t_test)
```

```
313/313 [=====] - 1s 2ms/step - loss: 0.1051 - accuracy: 0.9826
[0.10507538914680481, 0.9825999736785889]
```

[예제 0] MNIST 3번째 데이터인 x_train[2] 데이터에 대해서, 다음과 같은 3 x 3 필터에 대해 컨볼루션 연산을 하는 conv2d_simple(input_image, filter, filter_size) 구현하고 컨볼루션 결과를 이미지로 출력하시오



x_train[2]

$$\text{horizontal filter} = \begin{pmatrix} 1.0 & 1.0 & 1.0 \\ 0.0 & 0.0 & 0.0 \\ -1.0 & -1.0 & -1.0 \end{pmatrix}$$

$$\text{vertical filter} = \begin{pmatrix} 1.0 & 0.0 & -1.0 \\ 1.0 & 0.0 & -1.0 \\ 1.0 & 0.0 & -1.0 \end{pmatrix}$$

$$\text{blur filter} = \begin{pmatrix} 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \end{pmatrix}$$

$$\text{sharpen filter} = \begin{pmatrix} 0.0 & -1.0 & 0.0 \\ -1.0 & 5.0 & -1.0 \\ 0.0 & -1.0 & 0.0 \end{pmatrix}$$

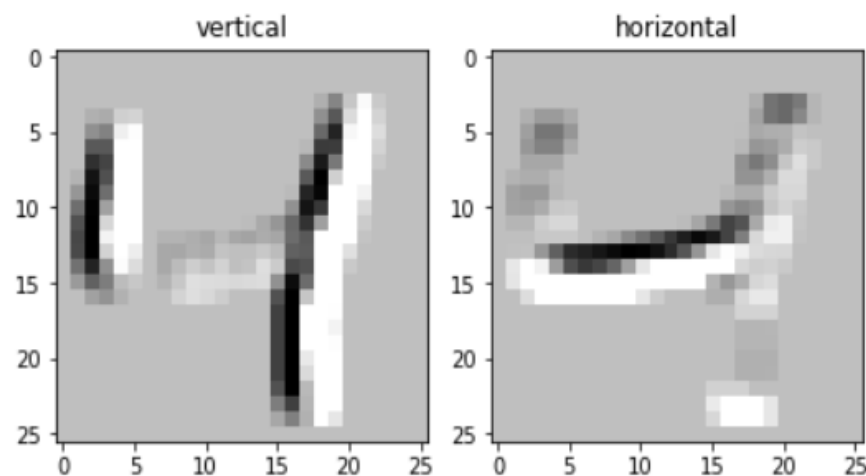
```
vertical_filtered_image = conv2d_simple(x_train[2], vertical_filter, 3)
horizontal_filtered_image = conv2d_simple(x_train[2], horizontal_filter, 3)
```

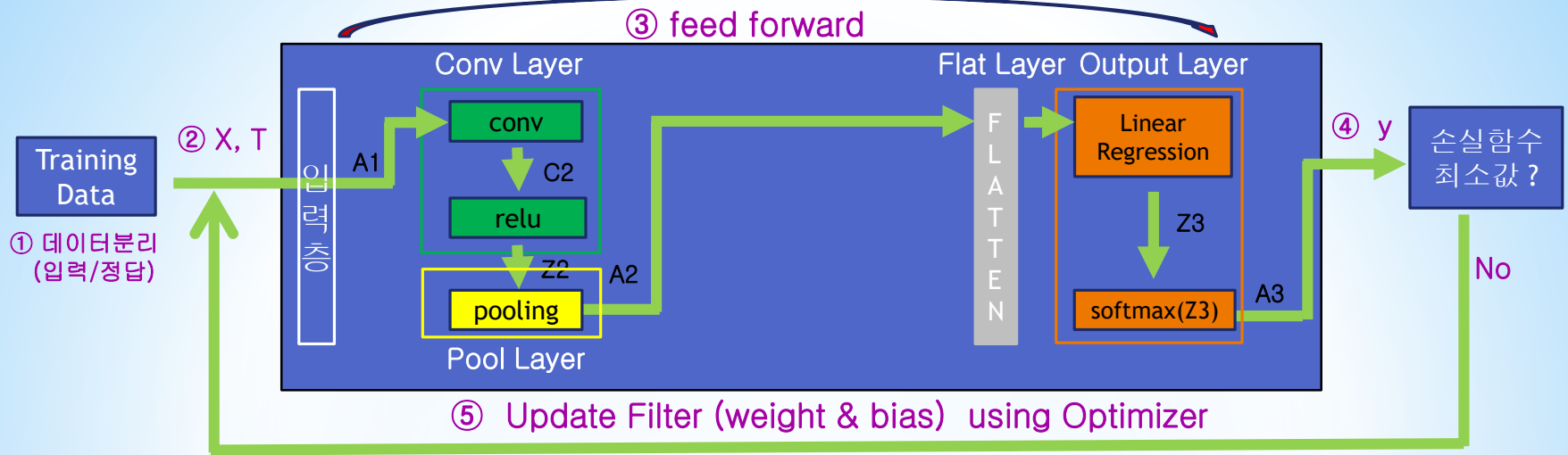
```
plt.figure(figsize=(6,4))

plt.subplot(1, 2, 1)
plt.title('vertical')
plt.imshow(vertical_filtered_image, cmap='gray')

plt.subplot(1, 2, 2)
plt.title('horizontal')
plt.imshow(horizontal_filtered_image, cmap='gray')

plt.tight_layout()
plt.show()
```



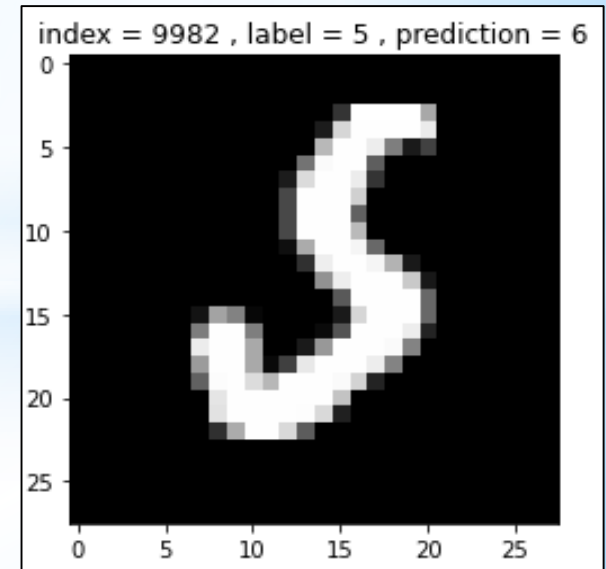


[예제 3]

1 conv / 1 flatten CNN 구조를 이용하여 test data 에 대해 predict 수행 후에 , index_label_prediction 값을 출력하는 코드를 구현하고, 임의의 false prediction 이미지를 다음과 같이 출력하시오

1 conv layer

=> 5 x 5 크기의 32개 필터, 1 stride, 2 x 2 max pooling, padding 있음



[예제 3] MNIST 예제 변경

기존에 구현하였던 MNIST 에서 정답을 one-hot encoding 방식으로 변환시키지 않고 입력으로 넣어주는 경우,

기존 소스에서 변경이 필요한 부분을 재작성해서 구현하시오