



Day60; 20221130(보강 시작)

날짜	@2022년 11월 30일
유형	@2022년 11월 30일
태그	

GitHub - u8yes/AI

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

<https://github.com/u8yes/ai>

u8yes/AI



1 Contributor 0 Issues 0 Stars 0 Forks

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8e23efe3-9f93-435b-b21b-4b648b60ca73/07_%ED%95%A9%EC%84%B1%EA%B3%B1_%EC%8B%A0%EA%B2%BD%EB%A7%9D\(CNN\).pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8e23efe3-9f93-435b-b21b-4b648b60ca73/07_%ED%95%A9%EC%84%B1%EA%B3%B1_%EC%8B%A0%EA%B2%BD%EB%A7%9D(CNN).pdf)

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cdf0955d-80e3-434f-9448-b1577ce8841e/01_fashionMNIST_CNN_20221130.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/dc2c8188-6134-4586-8834-0631c230a2ad/02_Best_CNN_Model_Tracer_20221130.ipynb

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b9498409-4368-421c-9eed-e401af8f65ec/best-cnn-model.h5>

RNN - 자연어 처리

8byte가 256.

1byte로는 1색상을 표현하고, 컬러는 4byte(3byte이지만 짝수 바이트가 가장 빨리 처리해서).

정방 행렬 - 행과 열이 같은 것(일반적으로 홀수로 잡음)

필터에 랜덤값을 잡는다. → 입력 데이터에 대봄.

합성곱 연산 – 입력 데이터에 필터를 적용



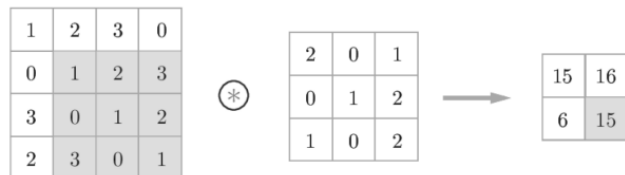
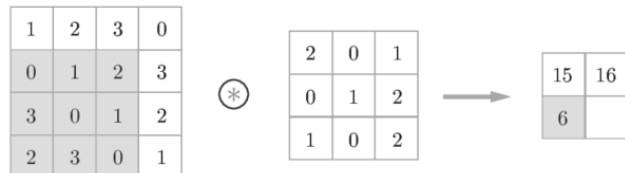
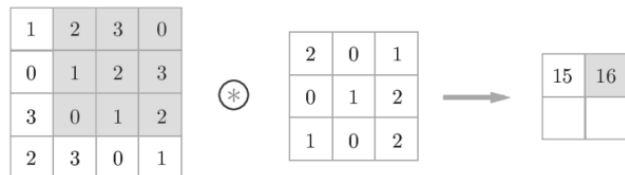
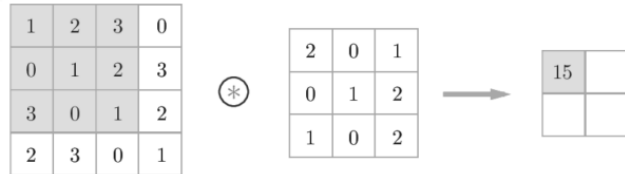
- 합성곱 연산을 $*$ 기호로 표기.
- 이미지 처리에서 말하는 필터 연산에 해당.
- 데이터와 필터의 형상을 (높이, 너비)로 표기.
 - 위 예의 경우 입력은 (4, 4), 필터는 (3, 3), 출력은 (2, 2)가 됨.
- 문헌에 따라 필터를 커널이라 칭하기도 함.

$1 * 2 + 2 * 0 + 3 * 1 + \dots + 1 * 2 = 15$ 가 나옴.

한칸 내려가는 식으로 계산.

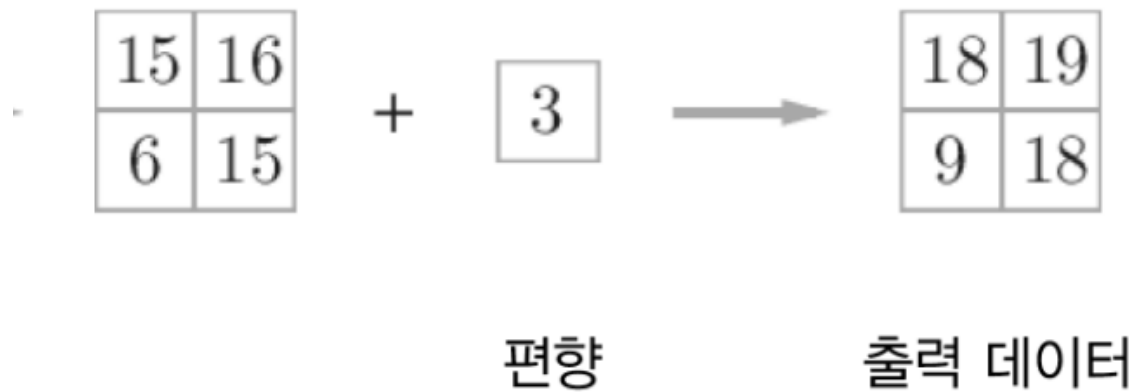
단일 곱셈 - 누산(Fused Multiply-Add, FMA)

- 합성곱 연산의 계산 순서



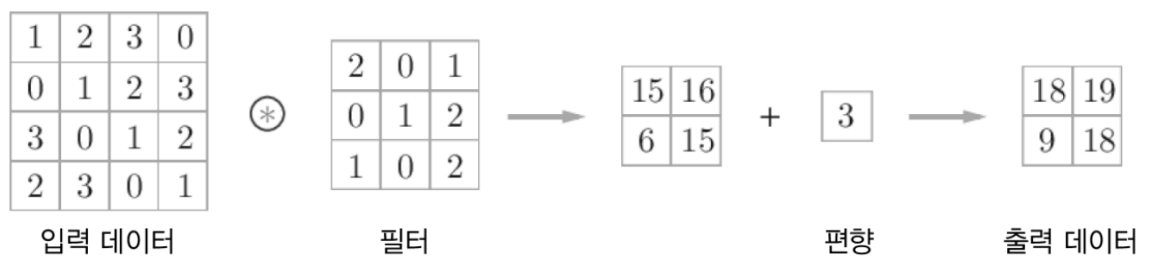
+= 편향

$15 + 3 + \dots 15 + 3 = 18, 19, 9, 18$



기본 개념은 회귀랑 비슷하다. $X \cdot W + \text{bias}$ 와 비슷하다.

합성곱 연산의 편향



- 필터를 적용한 후 데이터에 더해 줌.
- 편향은 항상 (1 x 1)만 존재.

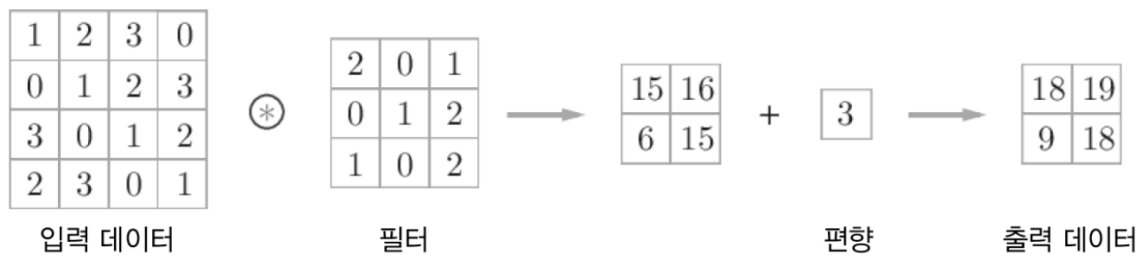
증명사진 같은 경우엔 필터를 크게 잡아도 뒷 배경은 거의 흰색, 회색 등으로 유사하게 나와서 상관없다. 아래의 숫자는 설명한 것과 상관없다.

2	0	1
0	1	2
1	0	2

필터

사이즈는 4X4 → 2X2로 줄어들었다.

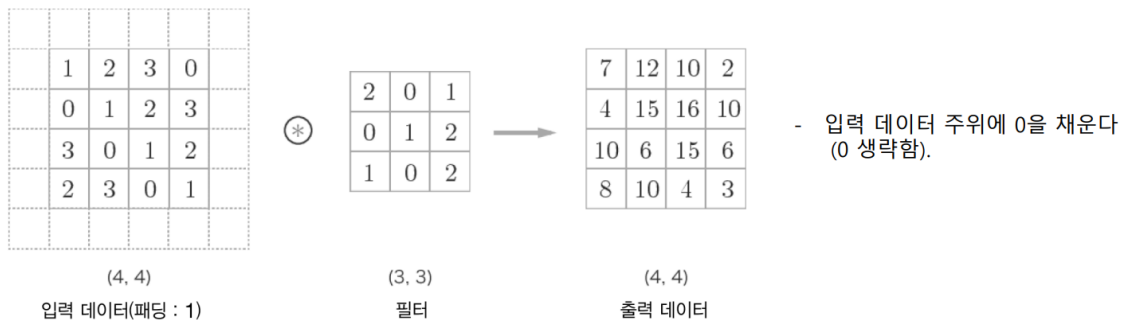
합성곱 연산의 편향



동일한 **Shape**을 유지할 수 있게끔 좌 우 위 아래 가상으로 픽셀을 추가해줌.

패딩(padding)

- 합성곱 연산을 수행하기 전에 입력 데이터 주변을 특정 값(예컨대 0)으로 채우는 것.
- 합성곱 연산에서 자주 이용하는 기법 - 출력 크기를 조정할 목적으로 사용.



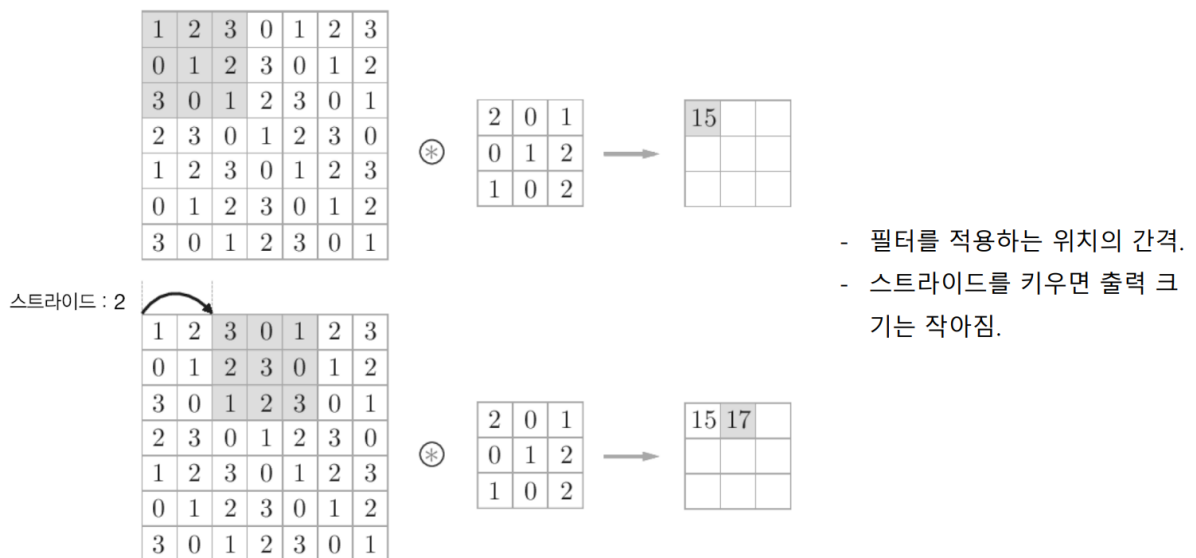
- 처음에 크기가 (4, 4)인 입력 데이터에 패딩이 추가되어 (6, 6)이 됨.
- (3, 3) 크기의 필터를 걸면 (4, 4) 크기의 출력 데이터가 생성.

입력 데이터(패딩 : 1)

Convolutional - padding - Stride <볼 패스>

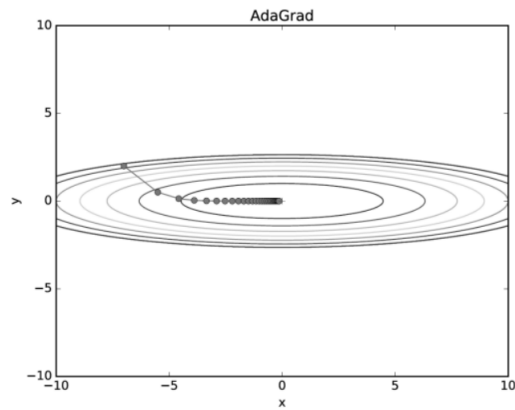
유사성이 있는 사진일 경우에는 스트라이드를 크게.

스트라이드



AdaGrad가 중간에 멈춰버리는 단점이 있어서 보완한 것이 디폴트로 RMSprop()를 맞춤.

AdaGrad



- 최소값을 향해 효율적으로 이동.
- y축 방향은 기울기가 커서 처음에는 크게 움직이지만, 그 큰 움직임에 비례해 갱신 정도도 큰 폭으로 작아지도록 조정.
- 따라서 y축 방향으로 갱신 강도가 빠르게 약해지고, 지그재그 움직임이 줄어듬.

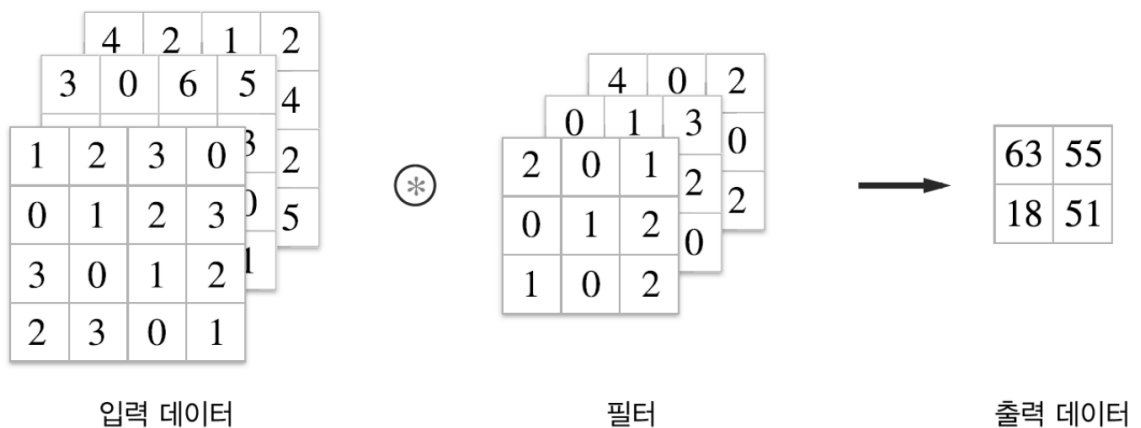
height, width

입력 크기(H, W)

Color일 경우에 3면으로 3차원, R G B.

Filter도 R G B로 구성.

3차원 데이터의 합성곱 연산



면끼리 다 더해 숫자를 넣어주면서 출력.

3차원 데이터 합성곱 연산의 계산 순서

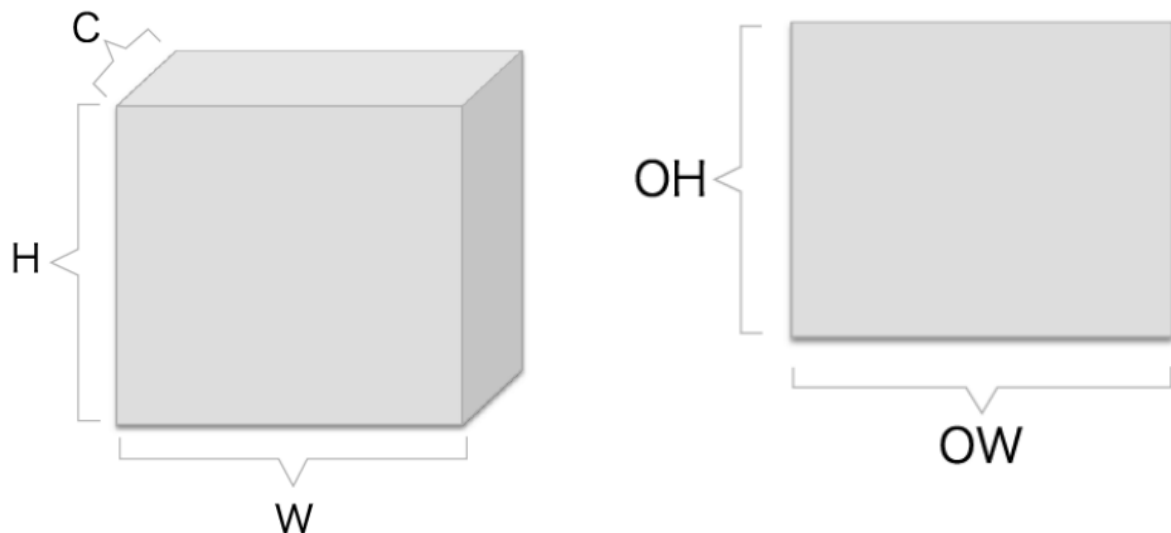


3차원의 합성곱 연산에서 주의할 점.

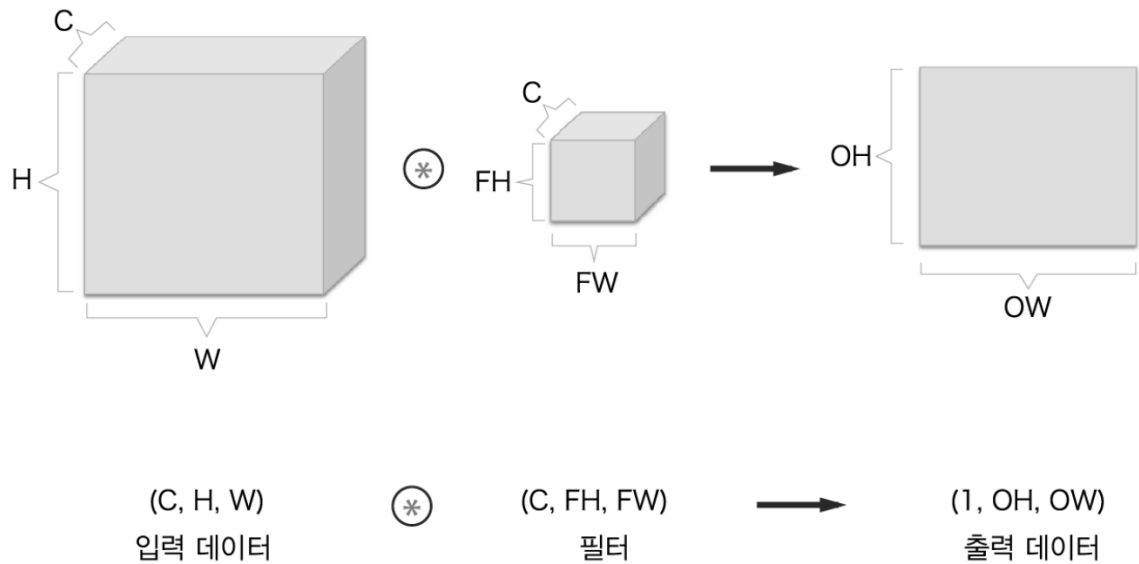
- 입력 데이터의 채널 수와 필터의 채널 수가 같아야 함.

3차원 입력 → 3차원 Filter → 2차원 출력

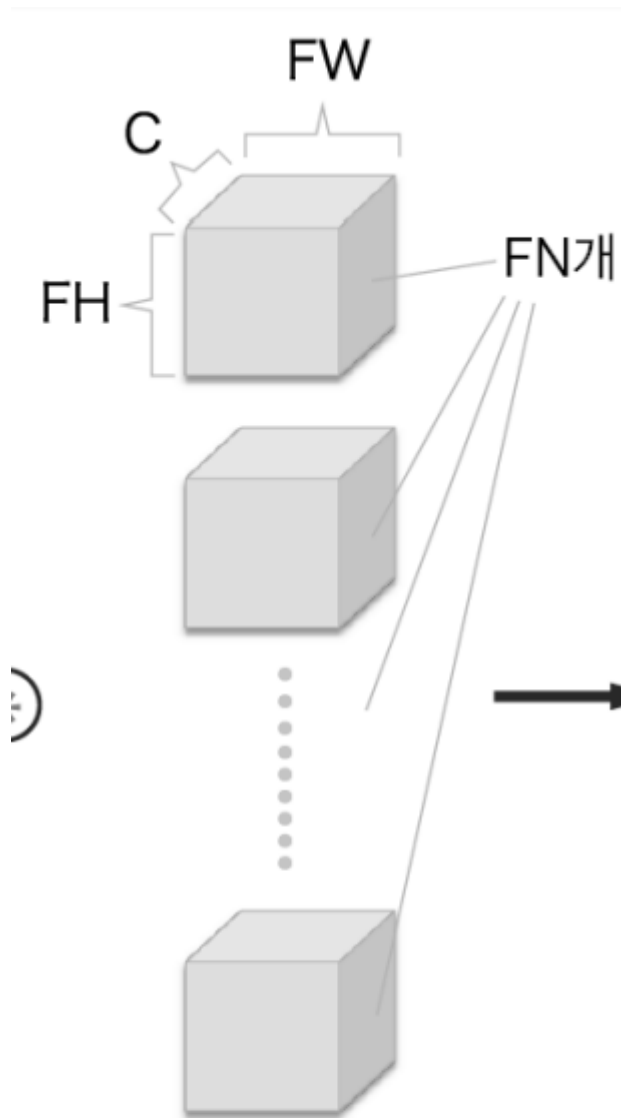
Channel은 R G B를 3차원 3면으로 표현, Height는 행, 높이, Width는 열, 너비 → Output은 2차원



3차원 합성곱 연산 – 블록으로 생각하기



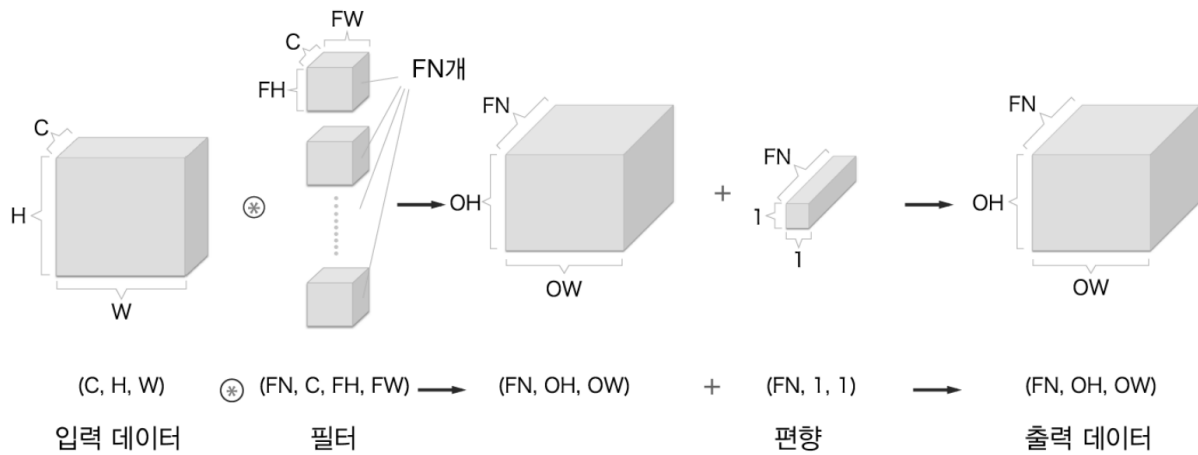
filter가 많으면 n가 있기에 4차원 filter가 됨. (다만 n개의 수를 포함시킨 것뿐) → 출력도 2차원이 아닌 3차원으로



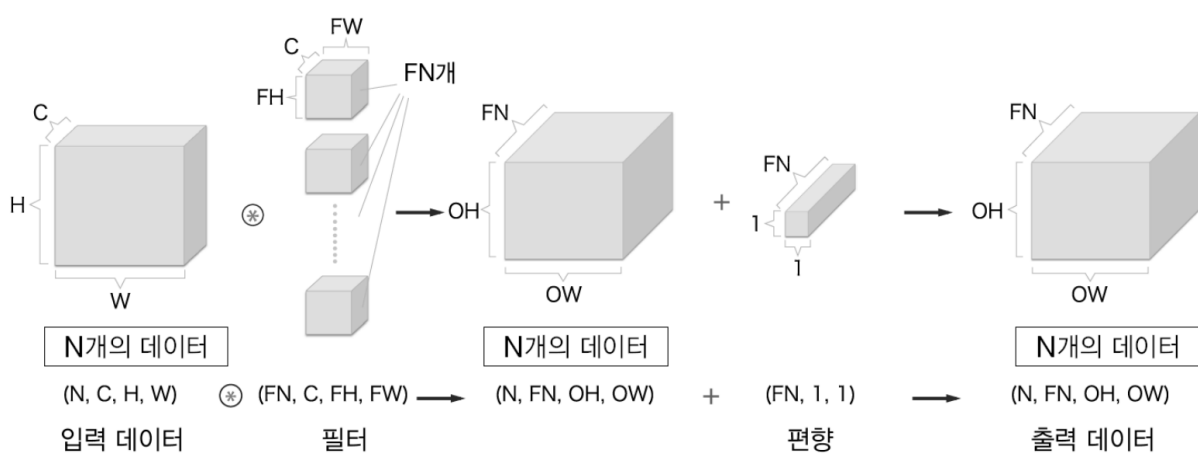
(FN, C, FH, FW) —

필터

합성곱 연산의 처리 흐름(편향 추가)



합성곱 연산의 처리 흐름(배치 처리)

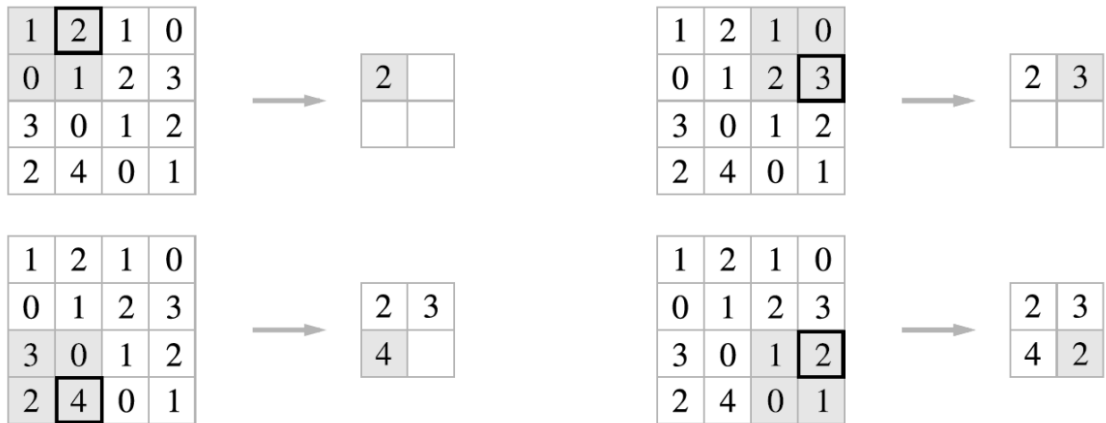


같은 3차원 이미지인데 **흑백은 1면**, **컬러는 3면**이 있는 것.

풀링 계층

풀링은 활성화함수를 거친 후에 넣어줌

풀링 계층



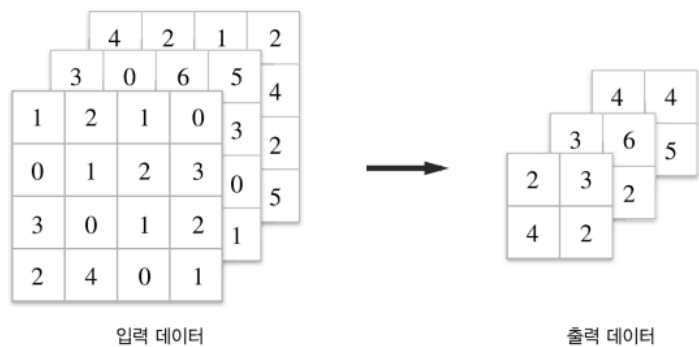
- 세로 · 가로 방향의 공간을 줄이는 연산.
- 위의 예 경우 2 x 2 최대 풀링(max pooling)을 스트라이드 2로 처리하는 순서.
- 풀링의 윈도우 크기와 스트라이드는 같은 값으로 설정하는 것이 일반적.

위의 예 경우 2 x 2 최대 풀링(max pooling)을 스트라이드 2로 처리하는 순서.

학습해야될 가중치가 필요없음.

풀링은 차원(채널)의 변화가 없다.

채널 수가 변하지 않는다.



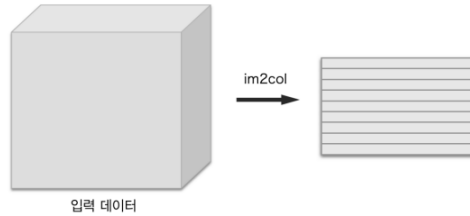
완전연결 계층 뒤에 드롭아웃 계층 사용

지식을 잘못 연계시키면 잘못된 지식이 되어버리니깐 연계도 잘해야한다.

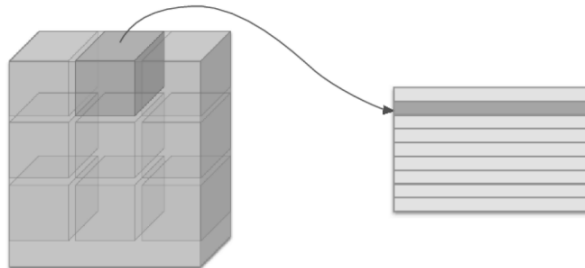
3차원의 세계가 여러 개 있는 것을 상상으로 표현한 것일뿐 정답은 아니다.

합성곱 계층 구현하기

- 4차원 배열 : CNN에서 계층 사이를 흐르는 데이터는 4차원.
- `im2col`로 데이터 전개하기 - 입력 데이터를 필터링(가중치 계산)하기 좋게 전개하는(펼치는) 함수.



- 입력 데이터에서 필터 적용 영역(3차원 블록)을 앞에서부터 순서대로 1줄로 펼친다.

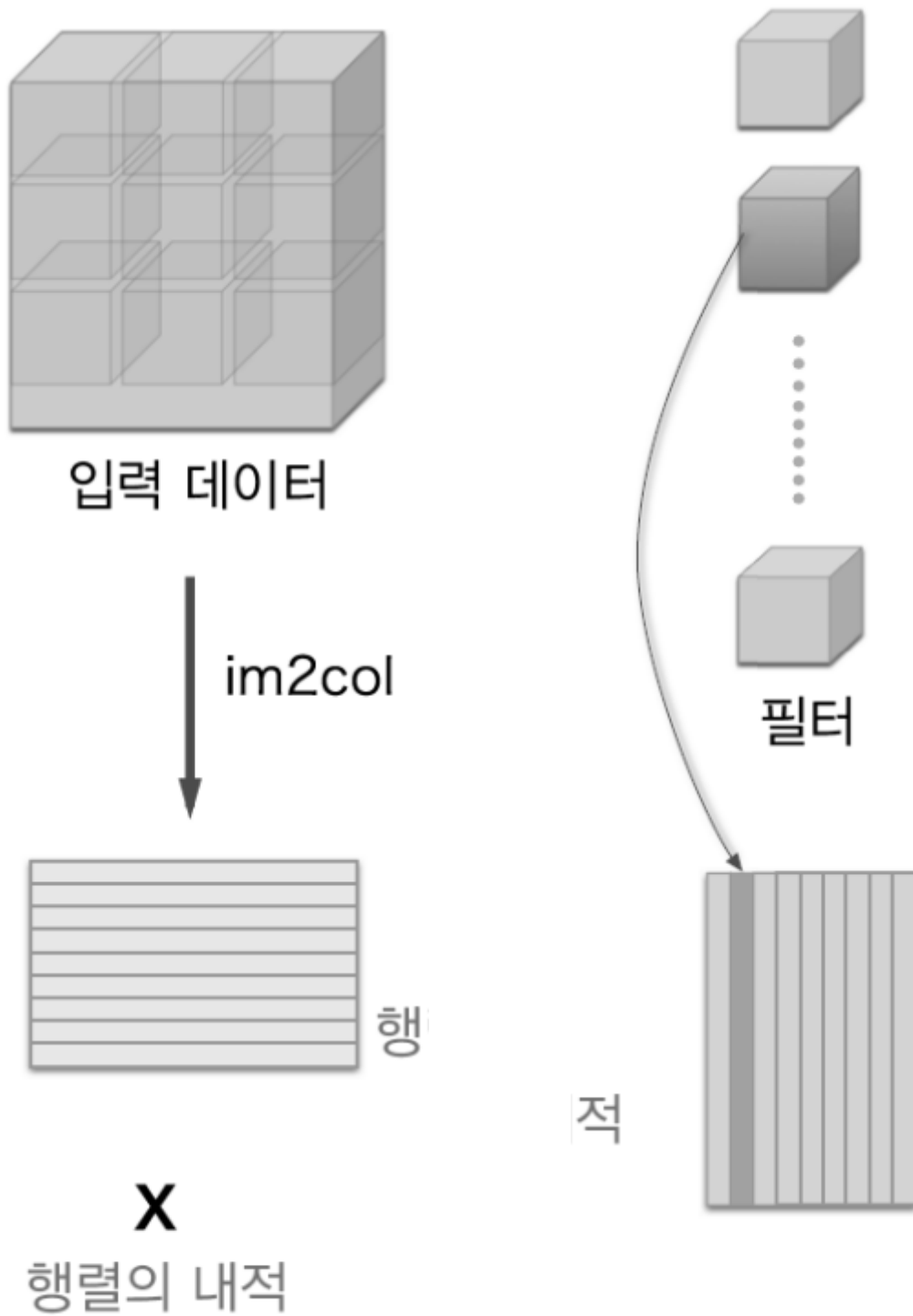


각 한 개의 3차원 → 2차원으로 펼쳐서 알고리즘을 동작시키게끔 함수를 정의해놨음.

$$28 * 28 * 3 = 2352$$

$$784 \times 3 =$$

2,352



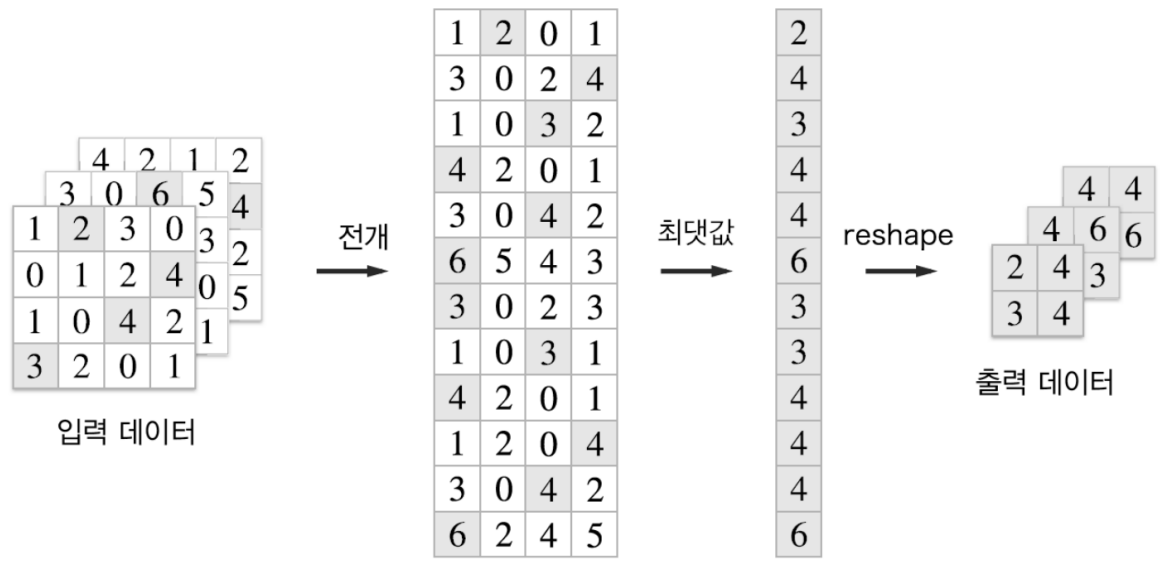
min-max는 axis가 0이면 각 행에서 최대값을 뽑아낸다. axis가 1이면 각 열로 비교를 해서 뽑아줌.

1	2	0	1
3	0	2	4
1	0	3	2
4	2	0	1
3	0	4	2
6	5	4	3
3	0	2	3
1	0	3	1
4	2	0	1
1	2	0	4
3	0	4	2
6	2	4	5

2
4
3
4
4
6
3
3
4
4
4
4
6

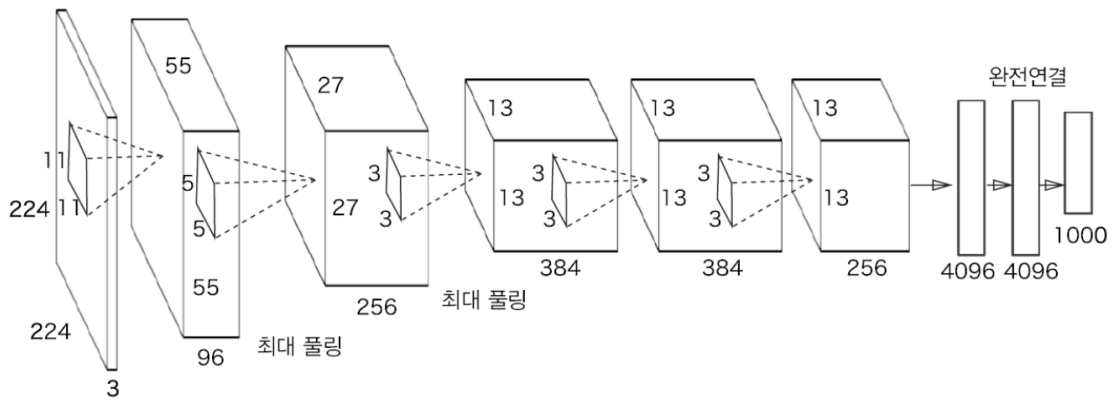
풀링은 axis를 1로 넣어줘야 함.

풀링 계층 구현의 흐름



대표적인 CNN - AlexNet

- 2012년에 발표된 AlexNet은 딥러닝 열풍을 일으키는 데 큰 역할.
- 구성



- 활성화 함수로 ReLU를 이용하고, 드롭아웃을 사용.
- LRN(Local Response Normalization)이라는 국소적 정규화를 실시하는 계층을 이용.

01_fashionMNIST_CNN


```

: from tensorflow import keras
  from sklearn.model_selection import train_test_split

  (train_input, train_target), (test_input, test_target) = \
      keras.datasets.fashion_mnist.load_data()

  train_scaled = train_input.reshape(-1, 28, 28, 1) / 255.0
  # 데이터는 모든 것-1, 행열은 28,28, Channel은 흑백1컬러3
  train_scaled, val_scaled, train_target, val_target = \
      train_test_split(train_scaled, train_target, test_size=0.2, random_state=42)

: print(train_scaled.shape, train_target.shape, val_scaled.shape, val_target.shape)

(48000, 28, 28, 1) (48000,) (12000, 28, 28, 1) (12000,)

: # CNN(합성곱) 신경망 만들기
  # convolutional 형
  model = keras.Sequential()
  model.add(keras.layers.Conv2D(filters=32, kernel_size=(3,3), input_shape=(28,28,1), padding='same',
                                activation='relu')) # Stride는 디폴트가 1씩 옮겨감.
  # input_shape=( ) N데이터, 행, 열, 면(흑백1컬러3)
  # padding='same' - filter가 입력의 shape만큼 똑같이 출력으로 나오게 해줌.

  model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
  # AveragePooling2D() 평균을 선택하겠다. MaxPooling2D 최대값을 선택하겠다.
  # filter가 pool_size 옮길 때 형태들

: model.add(keras.layers.Conv2D(filters=64, kernel_size=(3,3), input_shape=(28,28,1), padding='same',
                                activation='relu'))
  model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

: # 완전 연결형
  model.add(keras.layers.Flatten()) # Flatten은 알아서 펼쳐줌 - 완전 연결형.
  model.add(keras.layers.Dense(units=100, activation='relu'))
  model.add(keras.layers.Dropout(0.4)) # 40%는 출력을 내보내지 말라는 것. units=100개 중 40개를 죽임
  model.add(keras.layers.Dense(units=10, activation='softmax'))

```

Layer(은닉층) - 딥러닝에서 이렇게 부름.

relu는 오차역전파법 중에 0 밑으로 내려가는 것은 다 0으로, 그 위는 그대로 내보낸다는 것

```
: model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d_5 (MaxPooling 2D)	(None, 14, 14, 32)	0
conv2d_6 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_6 (MaxPooling 2D)	(None, 7, 7, 64)	0
flatten_5 (Flatten)	(None, 3136)	0
dense_10 (Dense)	(None, 100)	313700
dropout_5 (Dropout)	(None, 100)	0
dense_11 (Dense)	(None, 10)	1010

```
=====
Total params: 333,526
Trainable params: 333,526
Non-trainable params: 0
=====
```

```
: # 모델 컴파일과 훈련
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
```

```
] checkpoint_cb = keras.callbacks.ModelCheckpoint('best-cnn-model.h5', save_best_only=True) # 전이 학습
# callbacks - 운영체제로부터 전달받으면 오버라이딩 된 것을 실행시킴.
# ModelCheckpoint - filter를 업데이트하다가 최적의 값이 되는 부분을 저장했다가 기울기가 좋은 상태가 아닐 때 내보내줌.

early_stopping_cb = keras.callbacks.EarlyStopping(patience=2, restore_best_weights=True)
# 과대적합을 때 남은 epochs 실행안하고 멈춰라.
# 하지만 2번까지는 참아라.
# 최고의 가중치(기울기)

history = model.fit(train_scaled, train_target, epochs=20,
                    validation_data=(val_scaled, val_target),
                    callbacks=[checkpoint_cb, early_stopping_cb]) # epochs 너무 많이 학습시키면 과대적합된다.
# callbacks=[checkpoint_cb, early_stopping_cb]
```

```

Epoch 1/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.2582 - accuracy: 0.9068 - val_loss: 0.2303 - val_accuracy: 0.9136
Epoch 2/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.2335 - accuracy: 0.9136 - val_loss: 0.2292 - val_accuracy: 0.9161
Epoch 3/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.2121 - accuracy: 0.9213 - val_loss: 0.2209 - val_accuracy: 0.9193
Epoch 4/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.1959 - accuracy: 0.9275 - val_loss: 0.2232 - val_accuracy: 0.9208
Epoch 5/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.1838 - accuracy: 0.9315 - val_loss: 0.2181 - val_accuracy: 0.9223
Epoch 6/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.1663 - accuracy: 0.9377 - val_loss: 0.2287 - val_accuracy: 0.9210
Epoch 7/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.1566 - accuracy: 0.9397 - val_loss: 0.2141 - val_accuracy: 0.9244
Epoch 8/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.1452 - accuracy: 0.9449 - val_loss: 0.2317 - val_accuracy: 0.9237
Epoch 9/20
1500/1500 [=====] - 14s 9ms/step - loss: 0.1377 - accuracy: 0.9478 - val_loss: 0.2323 - val_accuracy: 0.9241

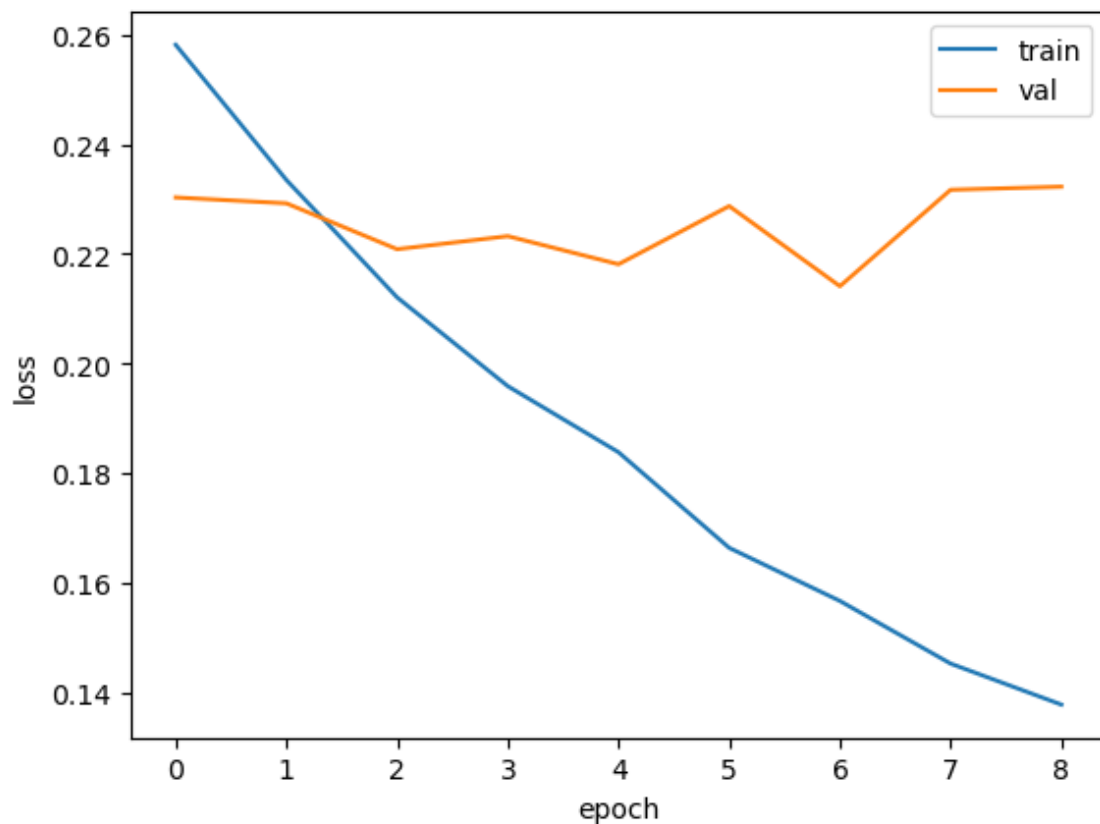
```

```

: import matplotlib.pyplot as plt

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.show()

```



너무 train으로 과대적합되어있음.

02_Best_CNN_Model_Tracer

```
[8]: from tensorflow import keras

model = keras.models.load_model('best-cnn-model.h5')

[9]: model.layers

[9]: [<keras.layers.convolutional.conv2d.Conv2D at 0x1c0227e3ec8>,
<keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x1c021c07548>,
<keras.layers.convolutional.conv2d.Conv2D at 0x1c0227fac08>,
<keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x1c022800948>,
<keras.layers.resizing.flatten.Flatten at 0x1c0227fa448>,
<keras.layers.core.dense.Dense at 0x1c02280dd08>,
<keras.layers.regularization.dropout.Dropout at 0x1c0227f3cc8>,
<keras.layers.core.dense.Dense at 0x1c0213eb5c8>]

10]: conv = model.layers[0]
print(conv.weights[0].shape, conv.weights[1].shape)

(3, 3, 1, 32) (32,)

11]: import numpy

conv_weights = conv.weights[0].numpy()
print(conv_weights.mean(), conv_weights.std())

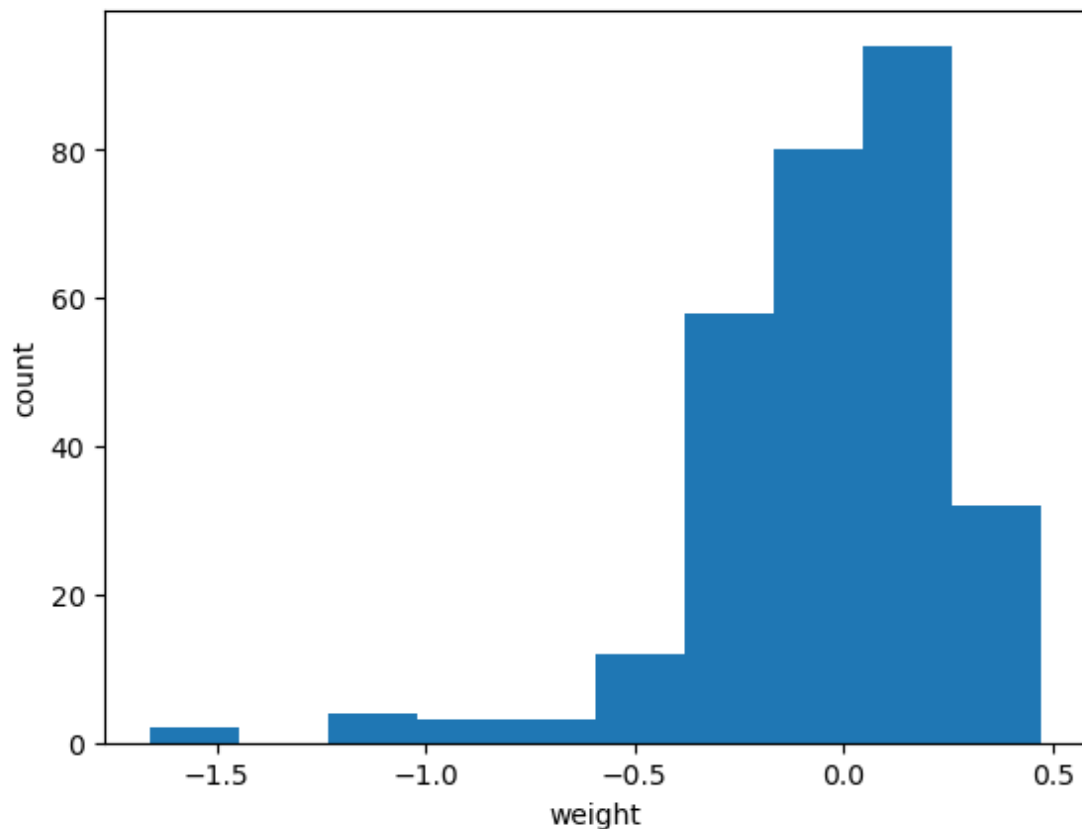
-0.039864466 0.3013197
```

```

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

plt.hist(conv_weights.reshape(-1,1))
plt.xlabel('weight')
plt.ylabel('count')
plt.show()

```



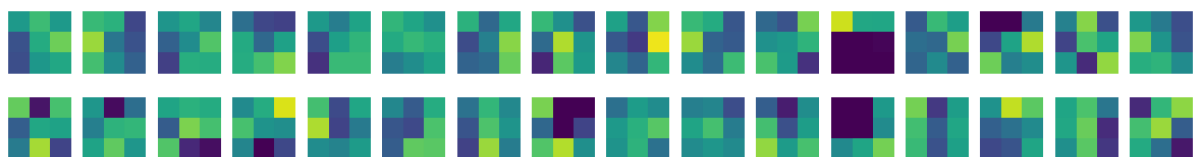
```

[9]: fig, axes = plt.subplots(2,16, figsize=(15,2)) # figsize인 것

for i in range(2):
    for j in range(16):
        axes[i,j].imshow(conv_weights[:, :, 0, i * 16 + j], vmin=-0.5, vmax=0.5)
        axes[i,j].axis('off')

plt.show()

```



```

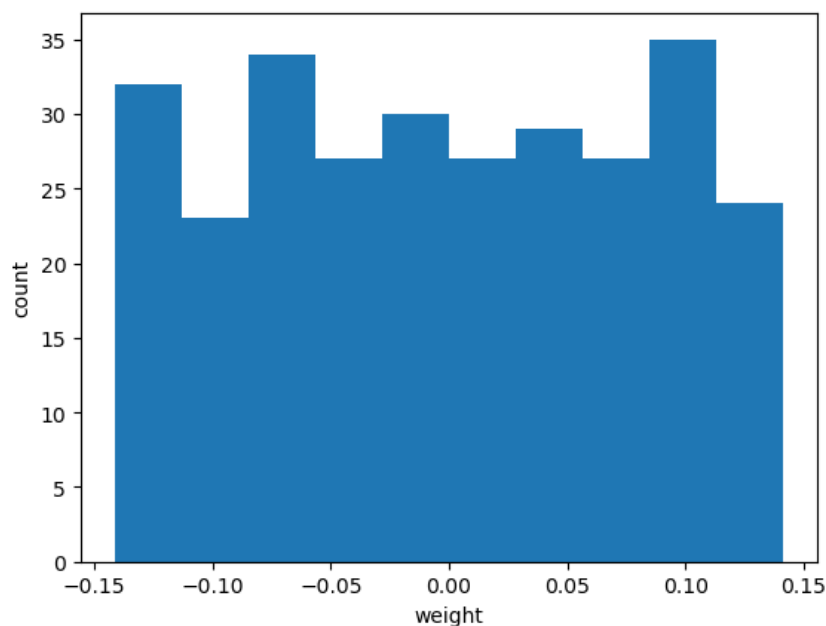
: no_training_model = keras.Sequential()
  no_training_model.add(keras.layers.Conv2D(filters=32, kernel_size=(3,3), input_shape=(28,28,1), padding='same',
                                             activation='relu'))

: no_training_conv = no_training_model.layers[0]
  print(no_training_conv.weights[0].shape)
(3, 3, 1, 32)

: no_training_weights = no_training_conv.weights[0].numpy()
  print(no_training_weights.mean(), no_training_weights.std())
-0.0005667812 0.08153395

: plt.hist(no_training_weights.reshape(-1,1))
  plt.xlabel('weight')
  plt.ylabel('count')
  plt.show()

```



```

fig, axs = plt.subplots(2,16, figsize=(15,2)) # figsize인 것

for i in range(2):
    for j in range(16):
        axs[i,j].imshow(no_training_weights[:, :, 0, i * 16 + j], vmin=-0.5, vmax=0.5)
        axs[i,j].axis('off')

plt.show()

```



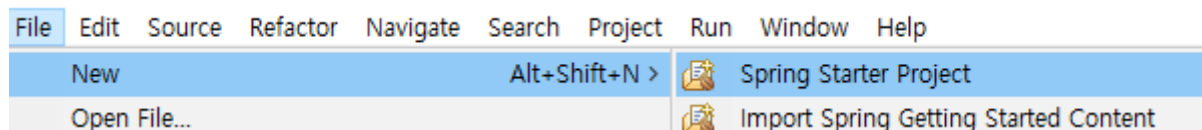
보강(Spring Boot)

흔·하·파)

- **튜플**: 함수와 함께 많이 사용되는 리스트와 비슷한 자료형으로, 리스트와 다른 점은 한번 결정된 요소는 바꿀 수 없습니다.
- **람다**: 매개변수로 함수를 전달하기 위해 함수 구문을 작성하는 것이 번거롭고, 코드 공간 낭비라는 생각이 들 때 함수를 간단하고 쉽게 선언하는 방법입니다.

pom - project object model

Spring Boot를 만들어줌

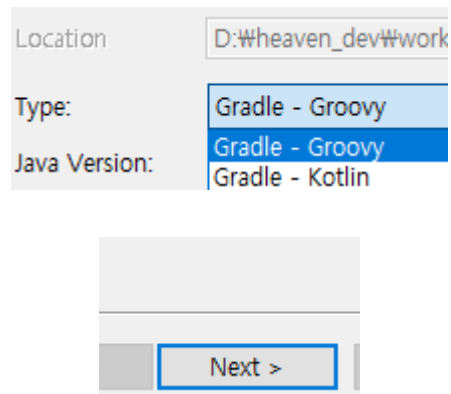


APP에 최적화된 Kotlin

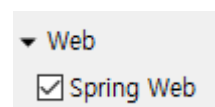
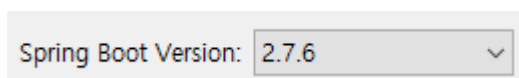
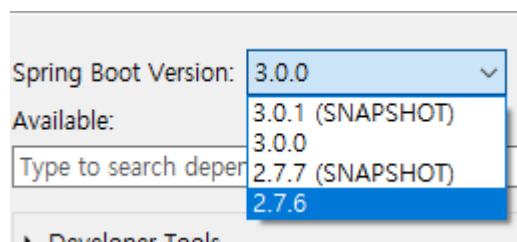
New Spring Starter Project

Service URL	<input type="text" value="https://start.spring.io"/>
Name	<input type="text" value="SpringBootProj"/>
<input checked="" type="checkbox"/> Use default location	
Location	<input type="text" value="D:\#heaven_dev\workspaces\spring"/>
Type:	<div>Gradle - Groovy ▾ Gradle - Groovy Gradle - Kotlin Maven com.example</div>
Java Version:	
Group	

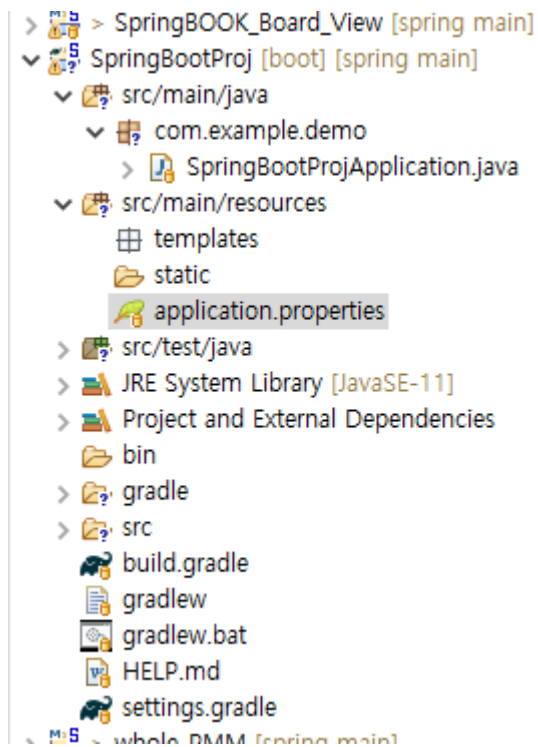
Spring Boot는 Groovy언어를 선택.



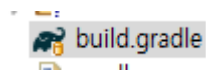
snapshot은 개발중인 거라서 위험



Finish 클릭




maven의 pom.xml 파일과 같은 gradle의 파일

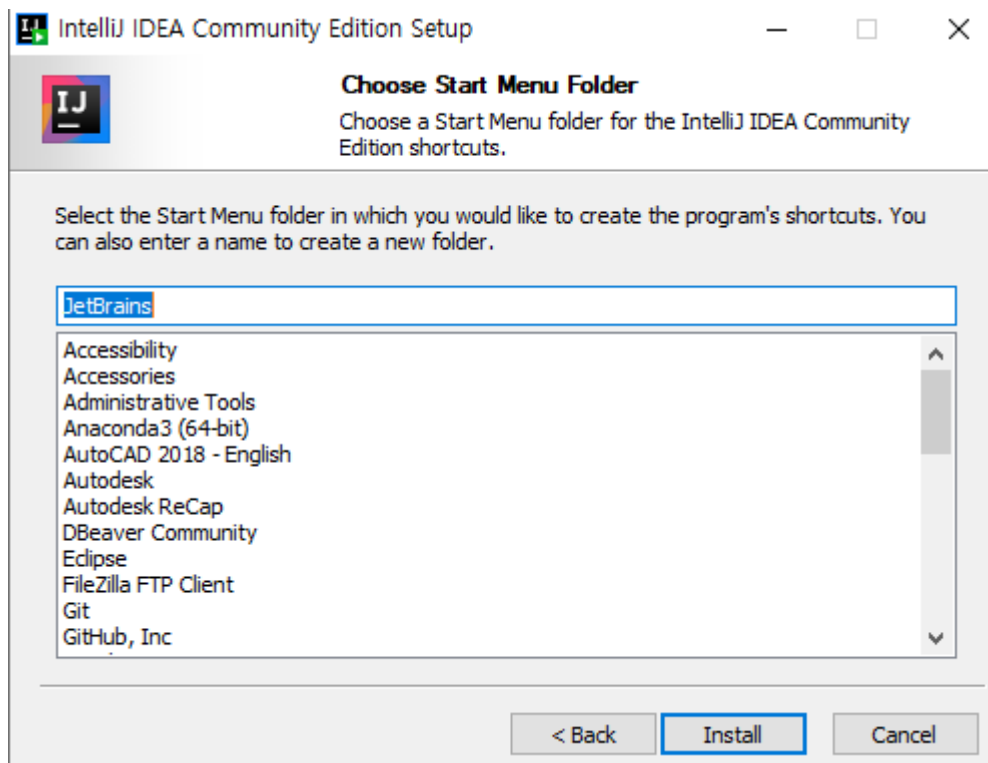
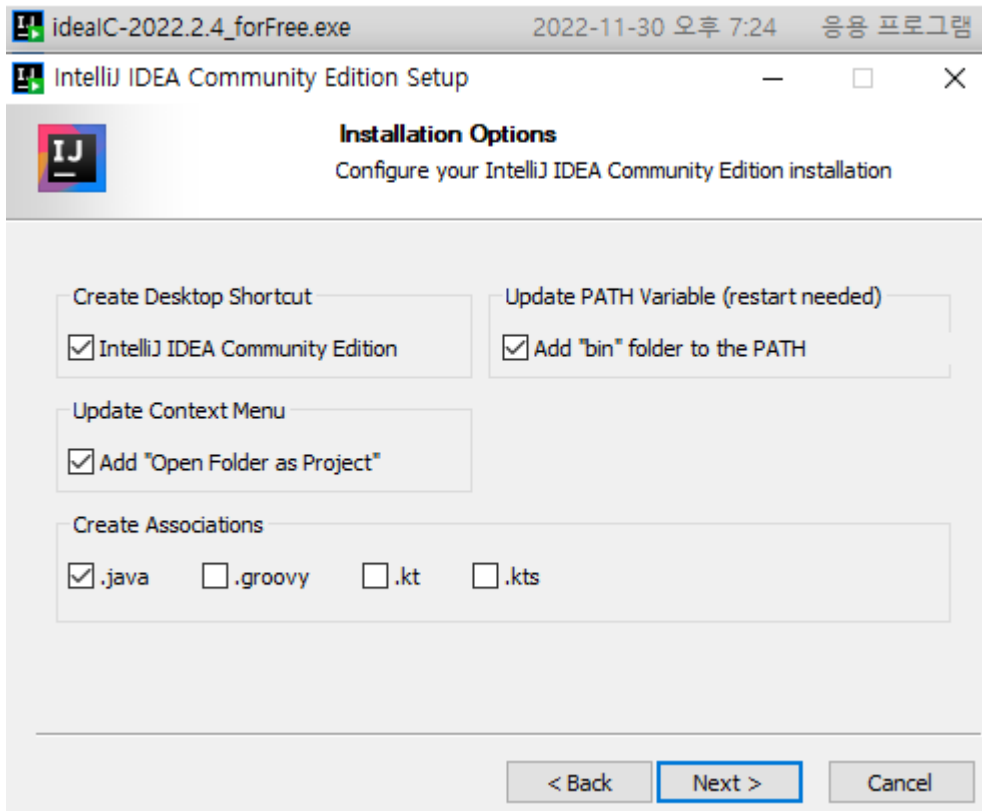


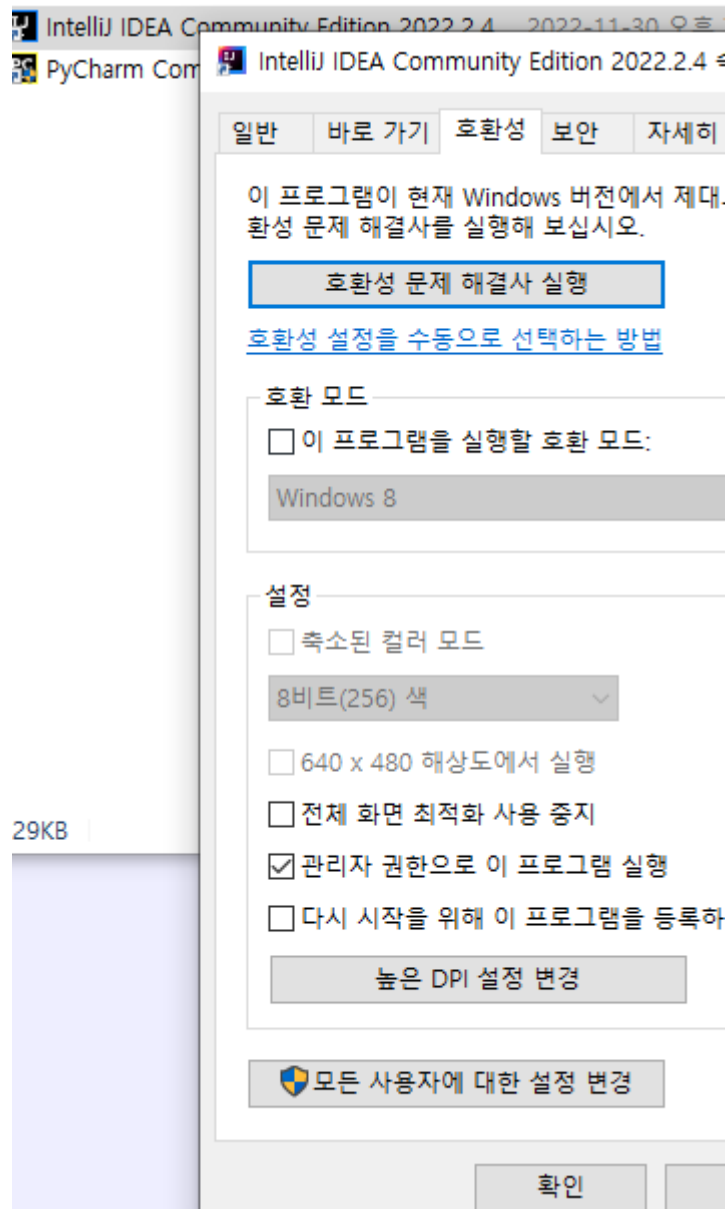
```
build.gradle x
1 plugins {
2     id 'java'
3     id 'org.springframework.boot' version '2.7.6'
4     id 'io.spring.dependency-management' version '1.0.15.RELEASE'
5 }
6
7 group = 'com.example'
8 version = '0.0.1-SNAPSHOT'
9 sourceCompatibility = '11'
10
11 repositories {
12     mavenCentral()
13 }
14
15 dependencies {
16     implementation 'org.springframework.boot:spring-boot-starter-web'
17     testImplementation 'org.springframework.boot:spring-boot-starter-test'
18 }
19
20 tasks.named('test') {
21     useJUnitPlatform()
22 }
```

다운로드 IntelliJ IDEA: 우수성과 인체 공학이 담긴 JetBrains Java IDE

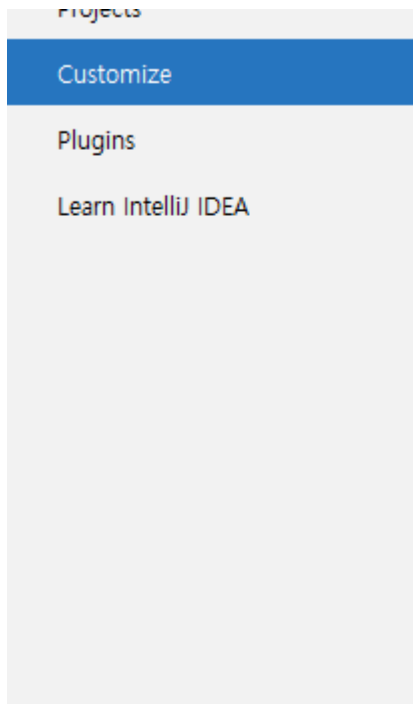
최신 버전 다운로드: IntelliJ IDEA (Windows, macOS, Linux)

 <https://www.jetbrains.com/ko-kr/idea/download/#section=windows>





이클립스와 단축기를 똑같이 쓸 수 있게 설정해줌.



Accessibility

IDE font: ▼

☐ Adjust colors for red-green vision deficiency
Requires restart. For protanopia and deuteranopia.

Keymap

▼

[Configure...](#)

[Import Settings...](#)

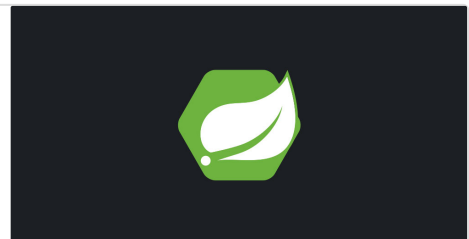
[All settings...](#)

스프링부트 프로젝트를 만드는 사이트까지 만들어봤음.

Spring Initializr

Initializr generates spring boot project with just what you need to start quickly!

 <https://start.spring.io/>



☰

spring initializr

⚙️

🌙

Project

☒ Gradle - Groovy
 ☐ Gradle - Kotlin
 ☐ Maven

Language

☒ Java
 ☐ Kotlin
 ☐ Groovy

Spring Boot

☐ 3.0.1 (SNAPSHOT)
 ☐ 3.0.0
 ☐ 2.7.7 (SNAPSHOT)
 ☒ 2.7.6

Project Metadata

Group

com.example

Artifact

demo

Name

SpringBootProj

Description

Demo project for Spring Boot

Package name

com.example.demo

Packaging

☒ Jar
 ☐ War

Java

☐ 19
 ☐ 17
 ☒ 11
 ☐ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf

TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

👤

🐦

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

packaging ☒ Jar ☐ War

Java ☐ 19 ☐ 17 ☒ 11 ☐ 8

GENERATE CTRL + ⌘

D:\heaven_dev\workspaces\SpringBoot\src\demo

- > java2
- > proj_save
- > Python
- > R
- > spring
- > spring_lecture
- ▼ SpringBoot
 - ▼ src
 - ▼ demo
 - > .gradle
 - > ..

