인공지능 (Artificial Intelligence)

심층 신경망

Hyuntae Cho

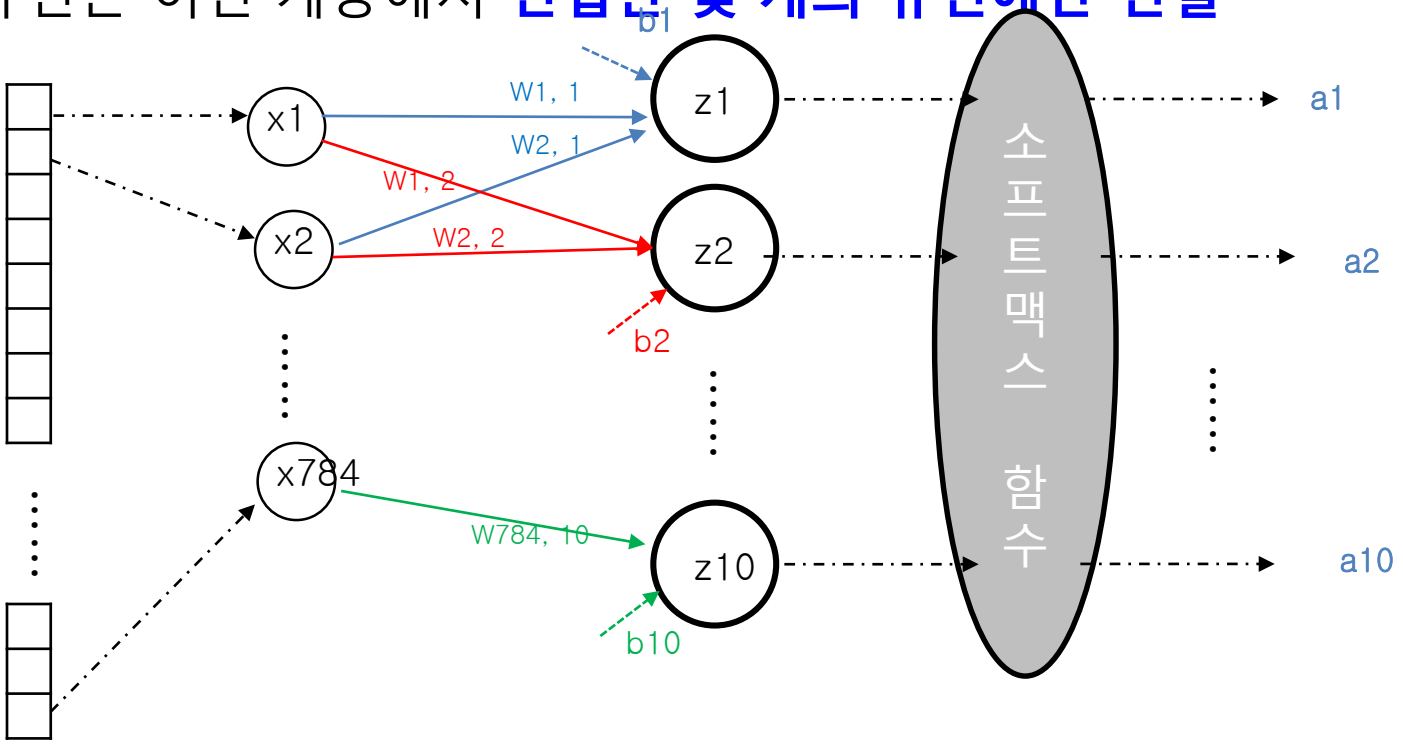
Dept. of Digital Media Engineering

Tongmyong University

동명대학교
TONGMYONG UNIVERSITY

합성곱 (convolution) 신경망

- **완전 연결층 (fully connected) 신경망**
  - 인접하는 계층의 모든 뉴런과 결합되어 있는 신경망
- **합성곱 신경망 – 1차원 데이터**
  - 각각의 뉴런은 이전 계층에서 **인접한 몇 개의 뉴런에만 연결**

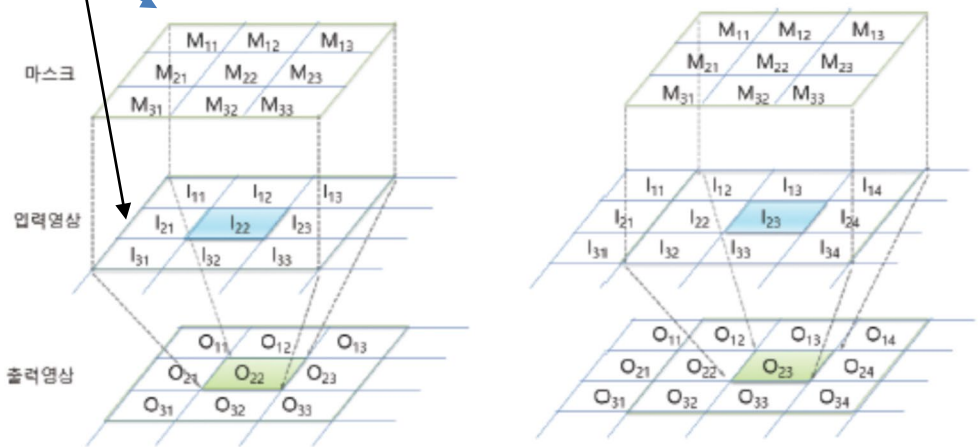


합성곱 (convolution) – 2차원 데이터

- 출력 화소 O_{22} = 입력 화소 I_{22} 와 mask 크기만큼의 주위 화소들을 이용해 계산
  - *즉, mask의 M_{ij} 영상의 I_{ij}* 를 곱한 다음, 모두 더함

$$
\begin{aligned}
O_{22} &= \sum\sum M_{ij} \cdot I_{ij} \\
&= M_{11} \cdot I_{11} + M_{12} \cdot I_{12} + M_{13} \cdot I_{13} \\
&+ M_{21} \cdot I_{21} + M_{22} \cdot I_{22} + M_{23} \cdot I_{23} \\
&+ M_{31} \cdot I_{31} + M_{32} \cdot I_{32} + M_{33} \cdot I_{33}
\end{aligned}
$$

O23 은 mask와 입력 영상을 이동 후, 같은 연산을 반복함.



$$
\begin{aligned}
O_{22} &= \sum\sum M_{ij} \cdot I_{ij} \\
&= M_{11} \cdot I_{11} + M_{12} \cdot I_{12} + M_{13} \cdot I_{13} \\
&+ M_{21} \cdot I_{21} + M_{22} \cdot I_{22} + M_{23} \cdot I_{23} \\
&+ M_{31} \cdot I_{31} + M_{32} \cdot I_{32} + M_{33} \cdot I_{33}
\end{aligned}
$$

$$
\begin{aligned}
O_{23} &= \sum\sum M_{ij} \cdot I_{ij} \\
&= M_{11} \cdot I_{12} + M_{12} \cdot I_{13} + M_{13} \cdot I_{14} \\
&+ M_{21} \cdot I_{22} + M_{22} \cdot I_{23} + M_{23} \cdot I_{24} \\
&+ M_{31} \cdot I_{32} + M_{32} \cdot I_{33} + M_{33} \cdot I_{34}
\end{aligned}
$$

〈그림 7.1.1〉 회선의 과정에 대한 이해

결과적으로 회선(convolution)으로 생성되는 영상은 mask의 원소값에 따라 결정됨.

필터개수: the dimensionality of the output space (i.e. the number of output filters in the convolution)

케라스에서 합성곱 사용

- 케라스의 층은 모두 keras.layers 패키지 아래 구현되어 있음 → 합성곱은 Conv2D() 메소드

```
from tensorflow import keras
keras.layers.Conv2D(10, kernel_size=(3,3), activation='relu')
```

필터의 개수 (마스크 개수)

커널의 크기 (마스크 크기)

활성화 함수

```
from tensorflow import keras
keras.layers.Conv2D(10, kernel_size=(3,3), activation='relu',
padding='same')
```

Same padding 지정 (입력 주위를 0으로 채움)

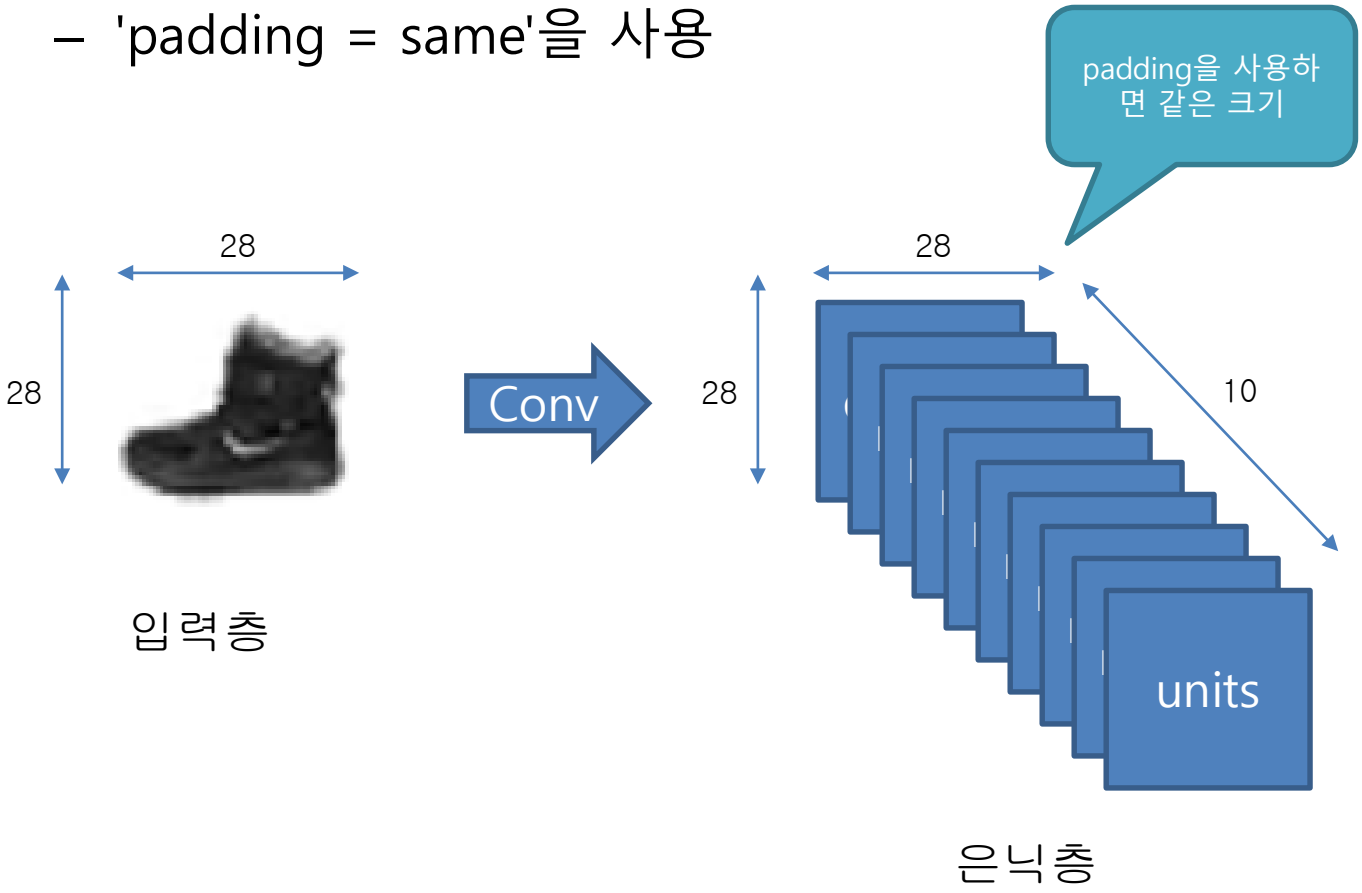
- Keras API를 사용하면 합성곱 신경망(CNN)을 쉽게 구현 가능

```
from tensorflow import keras
keras.layers.Conv2D(10, kernel_size=(3,3), activation='relu')
```

케라스에서 합성곱 사용

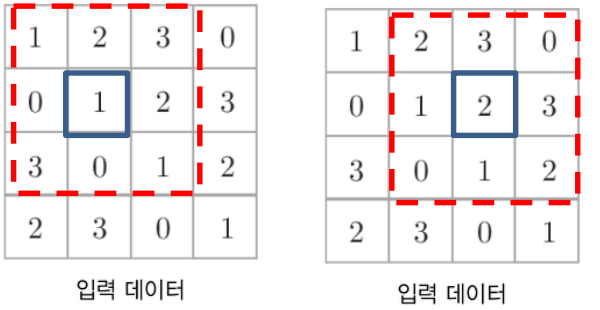
*패딩을 하지 않으면 24x24

- **앞장의 코드는?**
  - 커널 개수가 10개
  - 'padding = same'을 사용

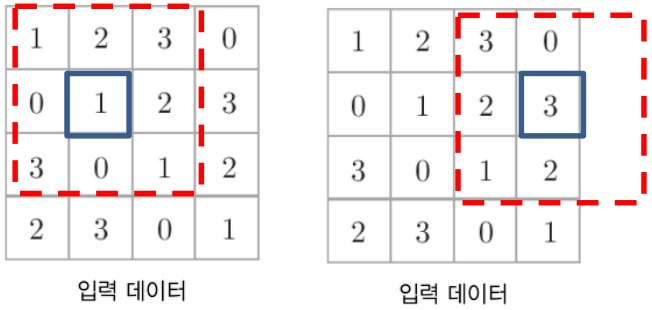


스트라이드 (strides)

- **필터(마스크)의 이동 간격을 설정**
  - 앞서 합성곱에서 1칸씩 이동한다고 배웠다.
  - 필터 (마스크)의 이동 거리를 설정



Strides = 1



Strides = 2

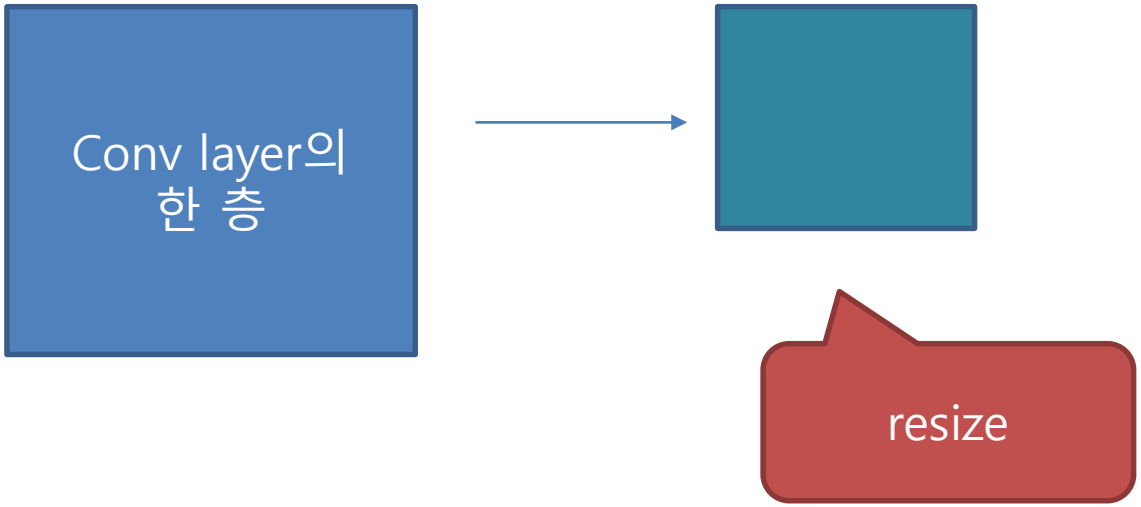
- **사용**

```
from tensorflow import keras
keras.layers.Conv2D(10, kernel_size=(3,3), activation='relu',
padding='same', strides=2)
```

필터 이동 거리

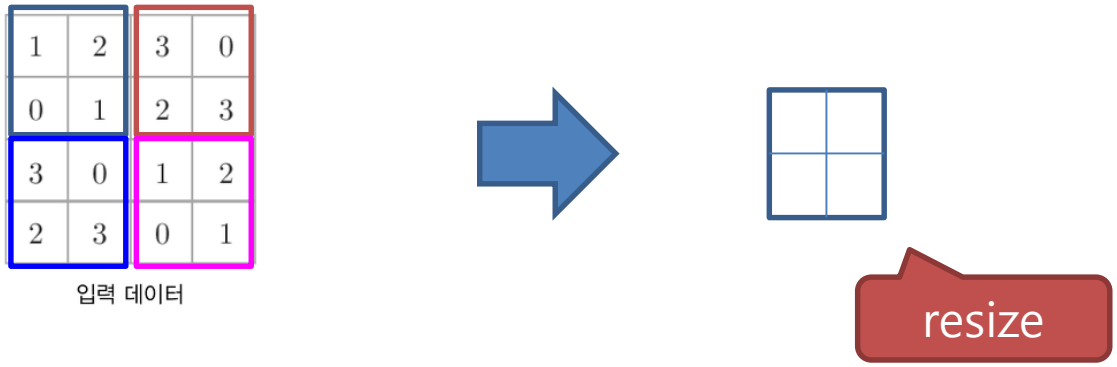
Pooling layer (풀링 층)

- **Sampling 또는 resizing 의 개념임**
  - 풀링 레이어는 콘볼루션 레이어의 **출력 값을 단순하게 압축**해주고
  - 콘볼루션 레이어가 생산해낸 정보를 컴팩트하게 만들어 줌



Pooling layer (풀링 층)

- **풀링:**
  - 합성곱 층에서 만든 특성 맵의 가로/세로 크기를 줄이는 역할을 수행
  - **최대 풀링(MaxPooling)**
    - 마스크에서 최대 값을 선택
  - 최소 풀링
    - 마스크에서 최소 값을 선택
- **Ex**
  - mask size = 2x2, strides = 2



Pooling layer (풀링 층)

- **Max Pooling 사용**

```
from tensorflow import keras

keras.layers.MaxPooling2D(2)
        or
keras.layers.MaxPooling2D (2, strides=2, padding='same')
```

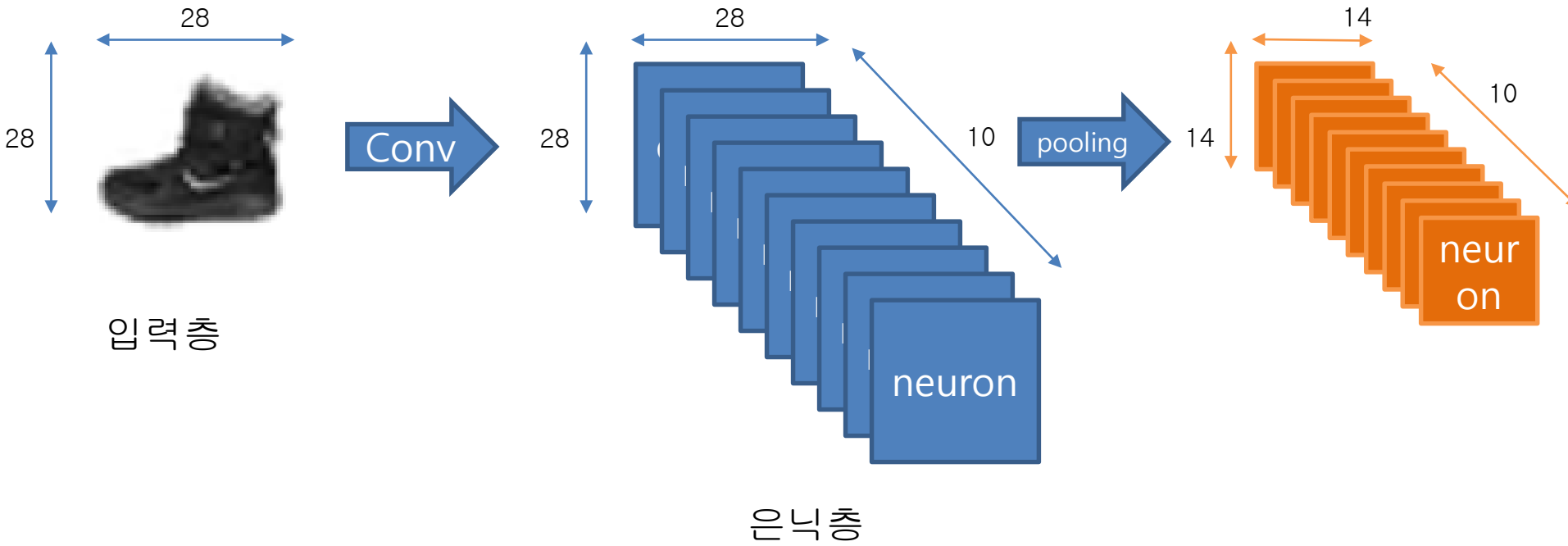
Pooling의 크기: 2 → 가로 세로 크기를 절반으로 줄임

이동 간격

Valid padding → 외곽 채우기 있음

케라스에서 합성곱 사용

- pooling 결과는?



요약

- **합성곱 (convolution):**
  - 입력과 가중치를 곱하고 필터 범위를 다 더함 → 그후, 절편을 취함
- **특성 맵**
  - 합성곱 층이나 풀링 층의 **출력 배열**
- **패딩**
  - 합성곱 층의 입력 주위에 추가항 0으로 채워진 픽셀
  - 출력의 크기를 입력과 같게 유지 하게 해줌
- **스트라이드**
  - 합성곱 층에서 필터(마스크)가 입력(이미지) 위를 이동하는 크기
- **풀링**
  - 가중치가 없고 특성맵의 가로/세로 크기를 줄이는 역할을 수행
  - resizing

합성곱 신경망 (CNN) 실습

- 1) import 텐서플로

```
#!pip install tensorflow-gpu==2.0.0-rc1
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
```

- 2) MNIST 데이터 셋 준비하기

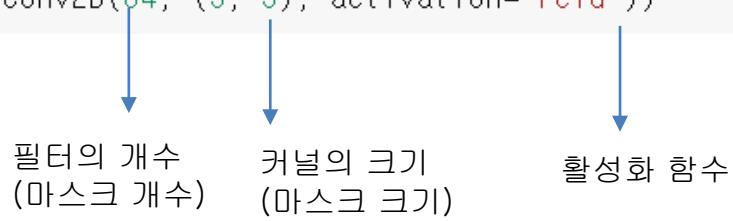
```
(train_input, train_target), (test_input, test_target) = keras.datasets.fashion_mnist.load_data(

train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# 픽셀 값을 0~1 사이로 정규화합니다.
train_images, test_images = train_images / 255.0, test_images / 255.0
```

출처: https://colab.research.google.com/github/tensorflow/docs-l10n/blob/master/site/ko/tutorials/images/cnn.ipynb?hl=ko#scrollTo=TzMWsTmkUiLY

합성곱 신경망 (CNN) 실습

- 3) 합성곱 층 만들기

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```



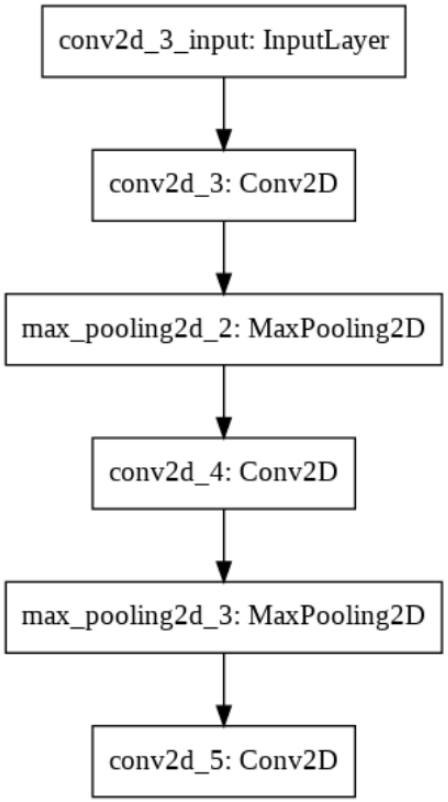
필터개수: the dimensionality of the output space (i.e. the number of output filters in the convolution)

합성곱 신경망 (CNN) 실습

- 만든 층 확인

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```



합성곱 신경망 (CNN) 실습

- 4) 마지막에 dense (밀집층) 추가하기

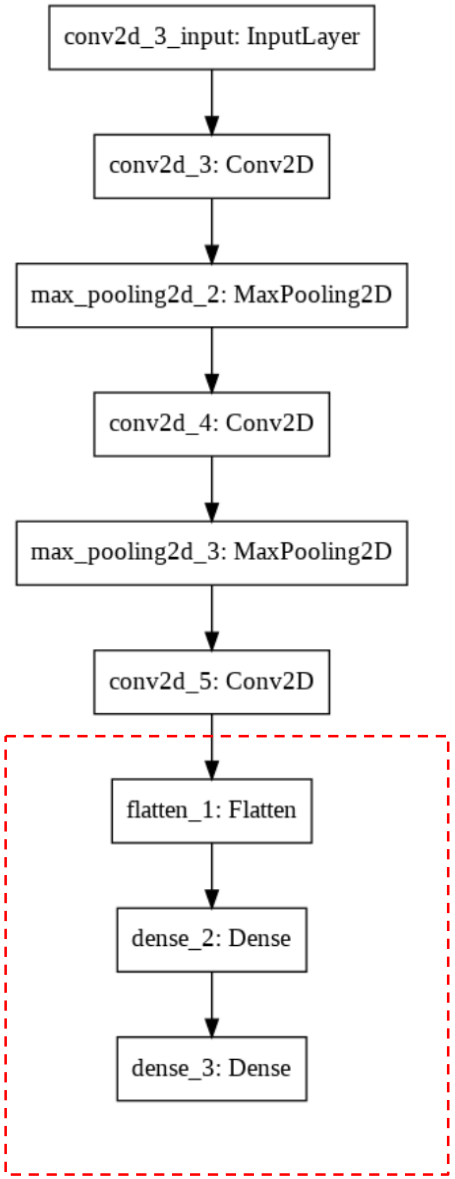
```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

합성곱 신경망 (CNN) 실습

- 만든 층 다시 확인

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```



합성곱 신경망 (CNN) 실습

- 5) 모델 컴파일 및 훈련 하기

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

model.fit(train_images, train_labels, epochs=5)
```

- 6) 모델 평가

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
```

```
313/313 - 3s - loss: 0.0255 - accuracy: 0.9920 - 3s/epoch - 9ms/step
```

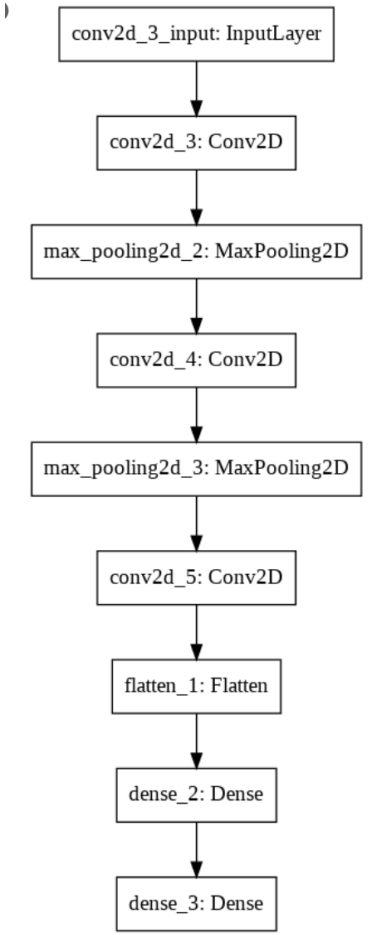
```
print(test_acc)
```

```
0.9919999837875366
```

Appendix

- 신경망 구조를 다음의 명령을 통해 그림으로 볼수 있다.

```
from tensorflow import keras
keras.utils.plot_model(model)
```



컨퓨전 메트릭스란?

- **얼마나 예측된 것이 실제 (참)와 잘 맞는가를 보여주는 척도**
  - True/False (참 거짓)은 **분류가 맞았다/틀렸다**
  - Positive/Nagative (양성/음성)은 **예측이 양성**으로 나왔다 **음성**으로 나왔다

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

https://blog.naver.com/sillllver/222889540993

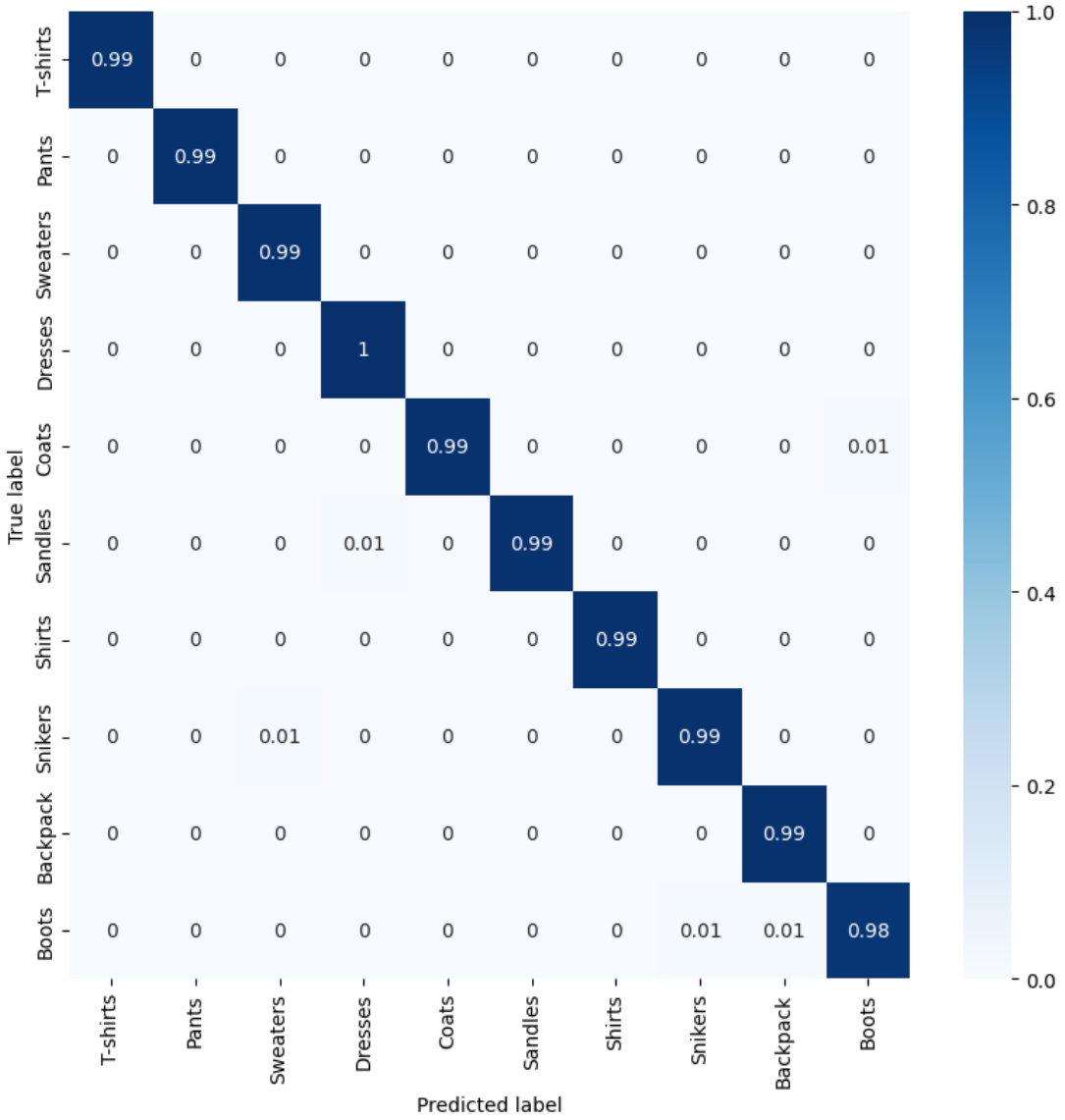
컴퓨전 I

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

IMAGES = [
    "T-shirts",
    "Pants",
    "Sweaters",
    "Dresses",
    "Coats",
    "Sandles",
    "Shirts",
    "Snikers",
    "Backpack",
    "Boots",
]

y_true = test_labels

predict_x = model.predict(test_images)
y_pred = np.argmax(predict_x, axis=1)


#classes=[0,1,2,3,4,5,6,7]
classes = IMAGES

con_mat = tf.math.confusion_matrix(labels=y_true, predictions=y_pred).numpy()
con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis], decimals=2)

con_mat_df = pd.DataFrame(con_mat_norm,
                     index = classes,
                     columns = classes)

figure = plt.figure(figsize=(8, 8))
sns.heatmap(con_mat_df, annot=True,cmap=plt.cm.Blues)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

Label	0	1	2	3	4	5	6	7	8	9
item	티셔츠	바지	스웨터	드레스	코트	샌달	셔츠	스니커즈	가방	앵클부츠

결과

- Y = 참값
- X = 예측값


True label
T-shirts
Pants
Sweaters
Dresses
Coats
Sandles
Shirts
Snikers
Backpack
Boots
0.99 0 0 0 0 0 0 0 0 0
0 0.99 0 0 0 0 0 0 0 0
0 0 0.99 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0.99 0 0 0 0 0.01
0 0 0 0.01 0 0.99 0 0 0 0
0 0 0 0 0 0 0.99 0 0 0
0 0 0.01 0 0 0 0 0.99 0 0
0 0 0 0 0 0 0 0 0.99 0
0 0 0 0 0 0 0 0.01 0.01 0.98
T-shirts
Pants
Sweaters
Dresses
Coats
Sandles
Shirts
Snikers
Backpack
Boots
Predicted label
1.0
0.8
0.6
0.4
0.2
0.0