week7 . 인공 신경망

합성곱

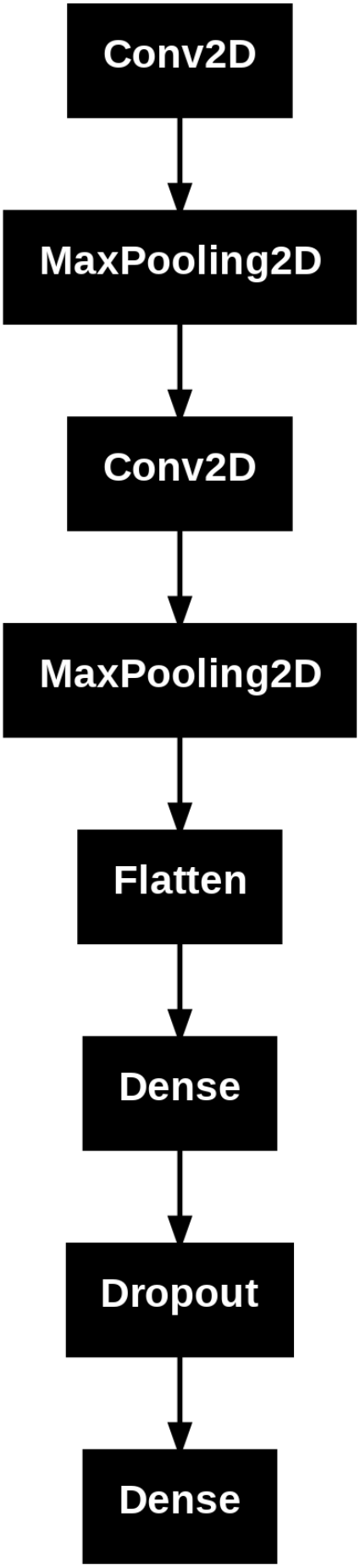
: 밀집층과 비슷하게 입력과 가중치를 곱하고 절편을 더하는 선형 계산.

- 밀집층과 달리 합성곱은 입력 전체가 아닌 일부만 사용하여 선형 계산 수행.
- 합성곱 층의 필터는 밀집층의 뉴런에 해당.
- 필터의 가중치와 절편을 종종 '커널' 이라 부르는데, 자주 사용되는 '커널'의 크기는 (3,3) 또는 (5,5).
- 커널의 깊이는 입력의 깊이와 같음

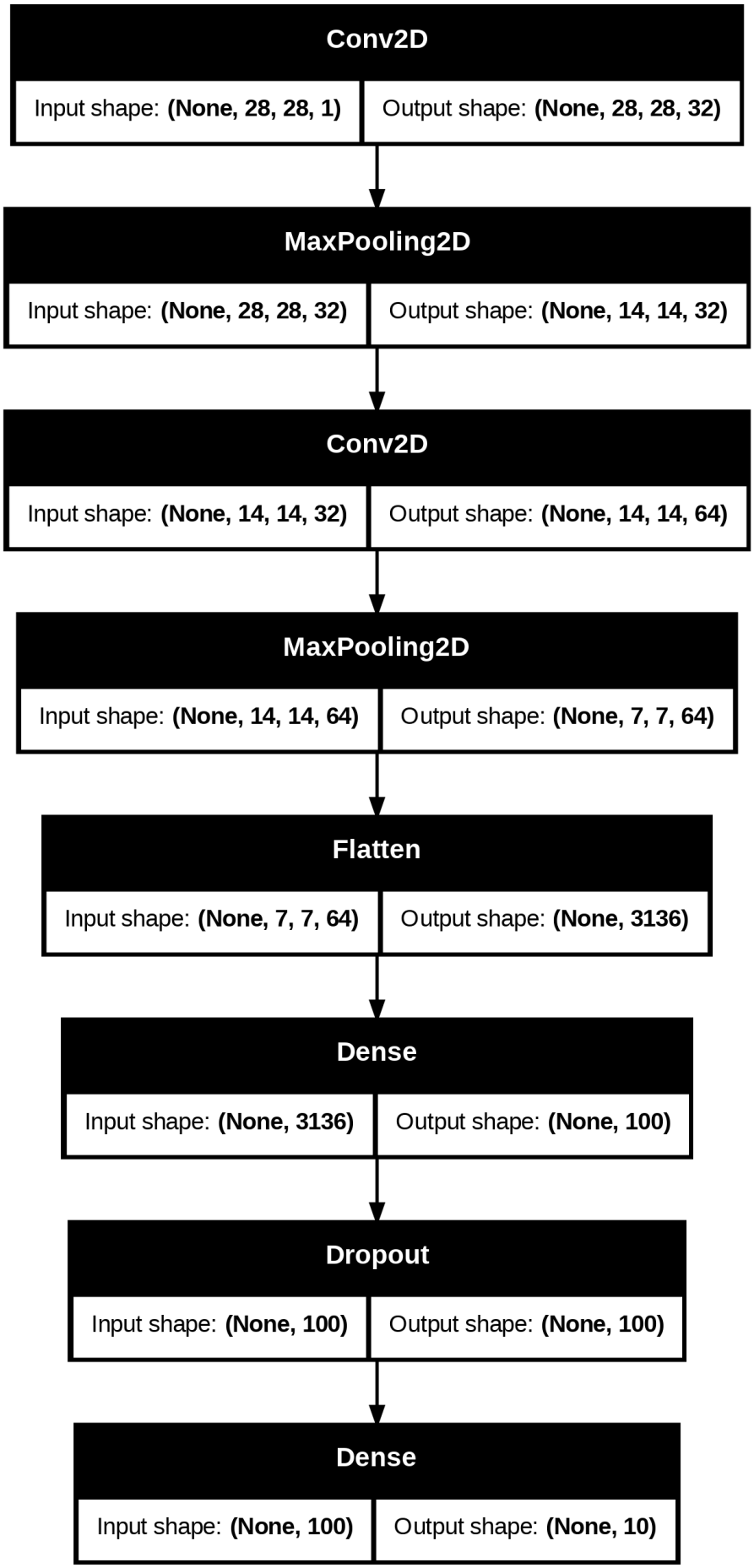
```
model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=3, activation='
                              padding='same', input_shape=(28

model.add(keras.layers.MaxPooling2D(2))

model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activati
                              padding='same'))
model.add(keras.layers.MaxPooling2D(2))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(100, activation='relu'))
model.add(keras.layers.Dropout(0.4))
model.add(keras.layers.Dense(10, activation='softmax'))

model.summary()

-> Model: "sequential"
 ┌─────────────────────────────────────────────────────────────
 |  Layer (type)                          |  Output Shape
```

```
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
│ conv2d (Conv2D)                      │ (None, 28, 28, 32)
├──────────────────────────────────────┼───────────────────────
│ max_pooling2d (MaxPooling2D)         │ (None, 14, 14, 32)
├──────────────────────────────────────┼───────────────────────
│ conv2d_1 (Conv2D)                    │ (None, 14, 14, 64)
├──────────────────────────────────────┼───────────────────────
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 7, 7, 64)
├──────────────────────────────────────┼───────────────────────
│ flatten (Flatten)                    │ (None, 3136)
├──────────────────────────────────────┼───────────────────────
│ dense (Dense)                        │ (None, 100)
├──────────────────────────────────────┼───────────────────────
│ dropout (Dropout)                    │ (None, 100)
├──────────────────────────────────────┼───────────────────────
│ dense_1 (Dense)                      │ (None, 10)
└──────────────────────────────────────┴───────────────────────
 Total params: 333,526 (1.27 MB)
 Trainable params: 333,526 (1.27 MB)
 Non-trainable params: 0 (0.00 B)
```

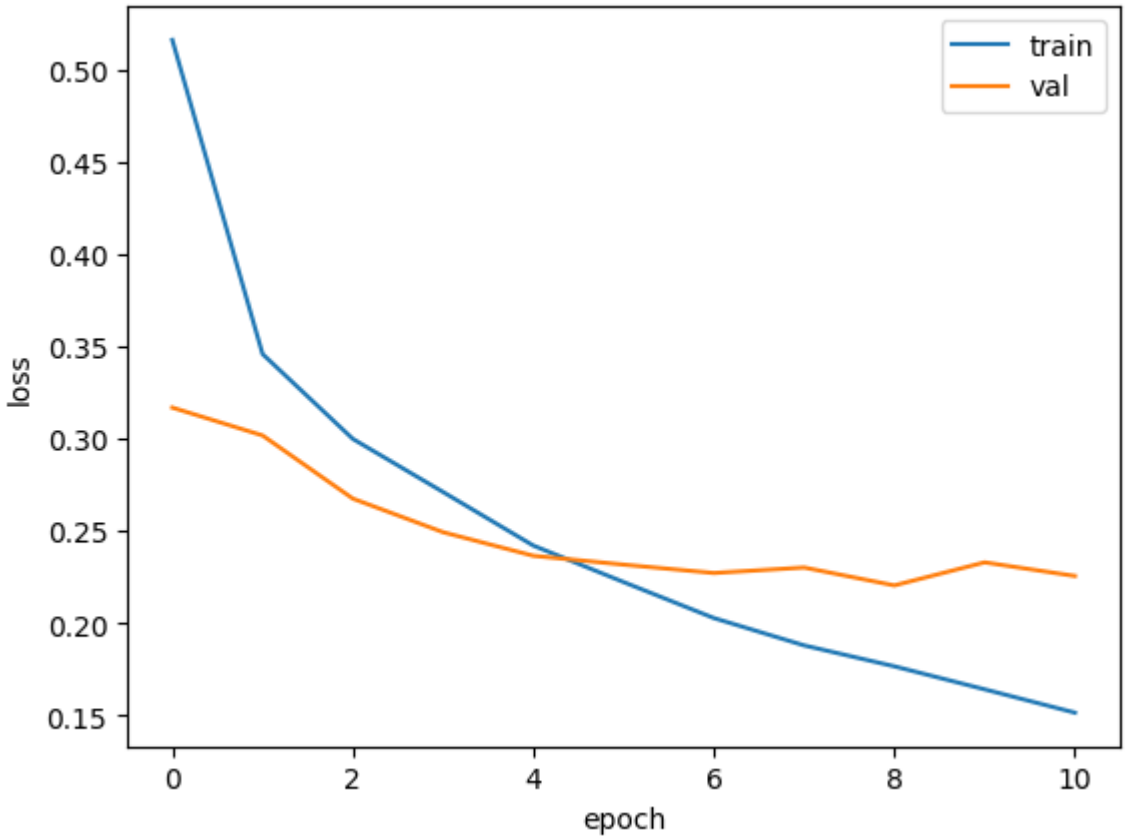
```
keras.utils.plot_model(model)
```


Conv2D
MaxPooling2D
Conv2D
MaxPooling2D
Flatten
Dense
Dropout
Dense

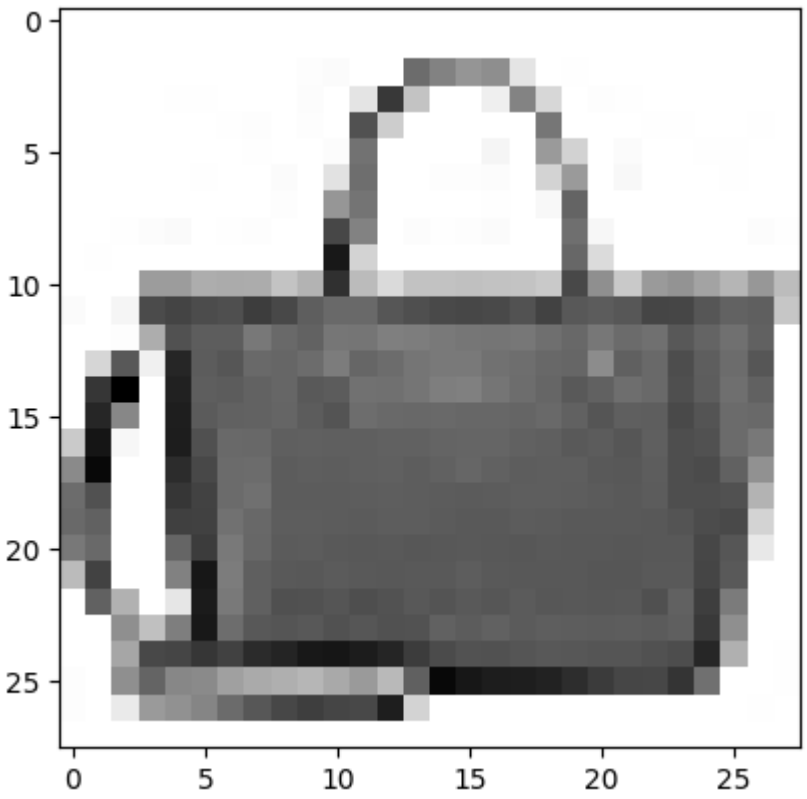
```
keras.utils.plot_model(model, show_shapes=True)
```


Conv2D
Input shape: (None, 28, 28, 1)
Output shape: (None, 28, 28, 32)
MaxPooling2D
Input shape: (None, 28, 28, 32)
Output shape: (None, 14, 14, 32)
Conv2D
Input shape: (None, 14, 14, 32)
Output shape: (None, 14, 14, 64)
MaxPooling2D
Input shape: (None, 14, 14, 64)
Output shape: (None, 7, 7, 64)
Flatten
Input shape: (None, 7, 7, 64)
Output shape: (None, 3136)
Dense
Input shape: (None, 3136)
Output shape: (None, 100)
Dropout
Input shape: (None, 100)
Output shape: (None, 100)
Dense
Input shape: (None, 100)
Output shape: (None, 10)

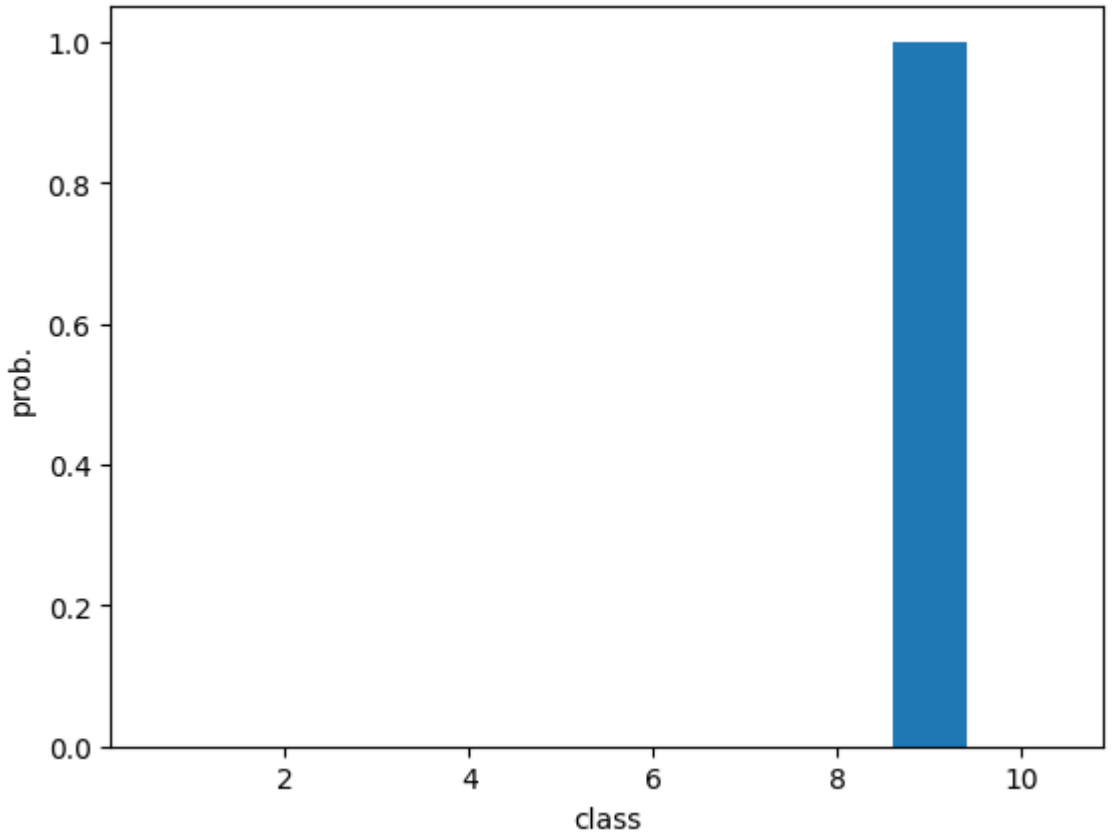
```
import matplotlib.pyplot as plt

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




```
plt.imshow(val_scaled[0].reshape(28, 28), cmap='gray_r')
plt.show()
```




```
plt.bar(range(1, 11), preds[0])
plt.xlabel('class')
plt.ylabel('prob.')
plt.show()
```




```
classes = ['티셔츠', '바지', '스웨터', '드레스', '코트',
           '샌달', '셔츠', '스니커즈', '가방', '앵클 부츠']

import numpy as np
print(classes[np.argmax(preds)])
-> 가방
```

특성 맵

: 합성곱 층이나 풀링 층의 출력 배열.

: 필터 하나가 하나의 특성 맵을 만들고, 합성곱 층에서 5개의 필터를 적용하면 5개의 특성 맵이 만들어짐.

패딩

: 합성곱 층의 입력 주위에 추가한 0으로 채워진 픽셀. 패딩을 사용하지 않는 것을 '밸리드 패딩' 이라고 하며, 합성곱 층의 출력 크기를 입력과 동일하게 만들기 위해 입력에 패딩을 추가하는 것을 '세임 패딩' 이라고 함.

스트라이드

: 합성곱 층에서 필터가 입력 위를 이동하는 크기.

: 일반적으로 스트라이드는 1픽셀을 사용.

풀링

: 가중치가 없고 특성 맵의 가로세로 크기를 줄이는 역할을 수행.

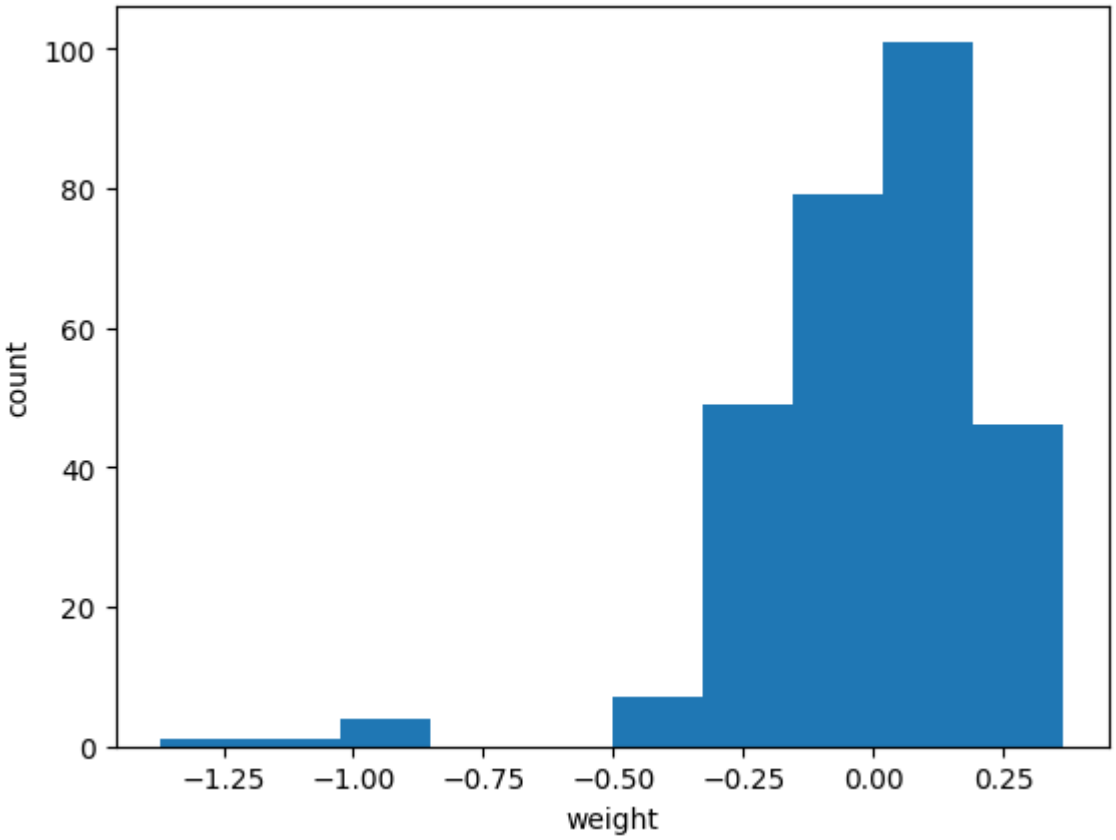
- 대표적으로 최대 풀링과 평균 풀링이 있으며, (2,2) 풀링으로 입력을 절반으로 줄임.

```
model = keras.models.load_model('best-cnn-model.keras')
model.layers
-> [<Conv2D name=conv2d, built=True>,
 <MaxPooling2D name=max_pooling2d, built=True>,
 <Conv2D name=conv2d_1, built=True>,
 <MaxPooling2D name=max_pooling2d_1, built=True>,
 <Flatten name=flatten, built=True>,
 <Dense name=dense, built=True>,
 <Dropout name=dropout, built=True>,
 <Dense name=dense_1, built=True>]
```

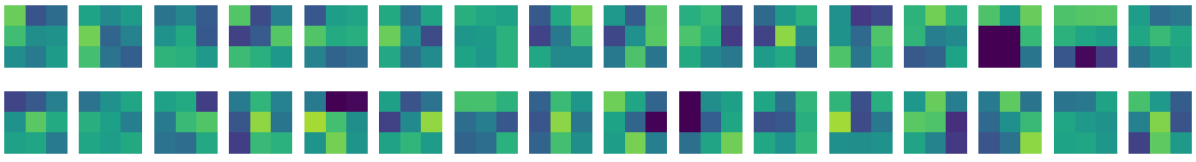
```
conv = model.layers[0] #첫번째 합성곱 층의 가중치와 두번째 원소(절편)의

print(conv.weights[0].shape, conv.weights[1].shape)
-> (3, 3, 1, 32) (32,)

conv_weights = conv.weights[0].numpy()

print(conv_weights.mean(), conv_weights.std())
#평균은 0에 가깝고 표준편차는 0.27정도
-> -0.014383553 0.23351653
```

```
import matplotlib.pyplot as plt
```

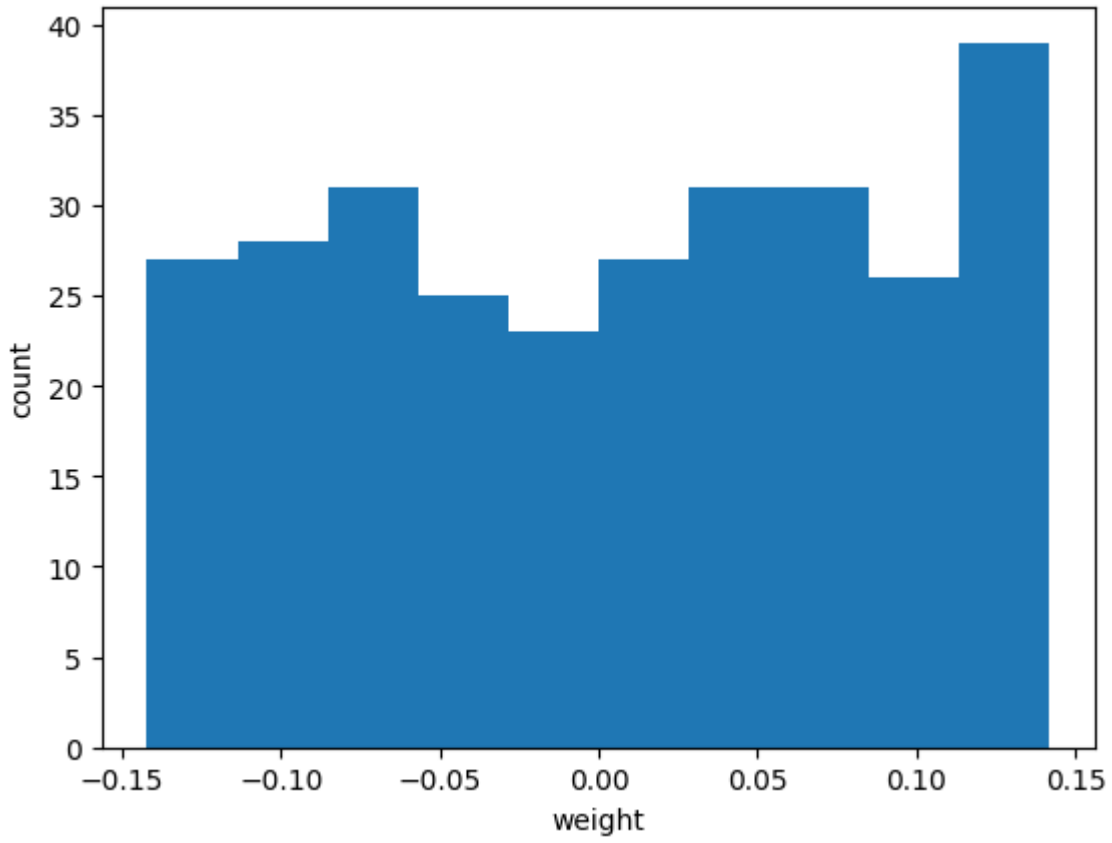
```
plt.hist(conv_weights.reshape(-1, 1))
plt.xlabel('weight')
plt.ylabel('count')
plt.show()
```




```
fig, axs = plt.subplots(2, 16, figsize=(15,2))

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

plt.show()
```



```
# Conv2D 층의 가중치를 no_training_conv 변수에 저장
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())
-> 0.0053191613 0.08463709

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))

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

plt.show()
```

Conv2D

: 입력의 너비와 높이 방향의 연산을 구현한 클래스

- 첫 번째 매개변수는 합성곱 필터의 개수이며, kernel_size 매개변수는 필터의 커널 크기를 지정함.
- 가로세로 크기가 같은 경우 정수 하나로, 다른 경우 튜플로 지정 가능.
- 일반적으로 커널의 가로세로의 크기는 동일하며, 커널의 깊이는 입력의 깊이와 동일하기 때문에 따로 지정하지 않음.
- strides 매개변수는 입력의 패딩 타입을 지정하고, 기본값 'valid'는 패딩을 하지 않음.
- 'same'은 합성곱 층의 출력의 가로세로 크기를 입력과 동일하게 맞추도록 입력에 패딩을 추가함.
- activation 매개변수는 합성곱 층에 적용할 활성화 함수를 지정.

MaxPooling2D

: 입력의 너비와 높이를 줄이는 풀링 연산을 구현한 클래스

- 첫 번째 매개변수는 풀링의 크기를 지정, 가로세로 크기가 같은 경우 정수 하나로, 다른 경우 정수의 튜플로 지정.
- 일반적으로 풀링의 가로세로 크기는 같게 지정.

- strides 매개변수는 풀링의 이동 간격을 지정.
- 기본값은 풀링의 크기와 동일. 즉, 입력 위를 겹쳐서 풀링하지 않음.
- padding 매개변수는 입력의 패딩 타입을 지정. 기본값 'valid'는 패딩을 하지 않으며, 'same'은 합성곱 층의 출력의 가로세로 크기를 입력과 동일하게 맞추도록 입력에 패딩을 추가.

plot_model()

: 케라스 모델 구조를 주피터 노트북에 그리거나 파일로 저장.

- 첫 번째 매개변수에 케라스 모델 객체를 전달.
- to_file 매개변수에 파일 이름을 지정 → 그림을 파일로 저장.
- show_shapes 매개변수를 True 로 지정하면 층의 입력, 출력 크기를 표시. 기본값은 False.
- show_layer_names 매개변수를 True 로 지정하면 층 이름을 출력. 기본값은 True

matplotlib

: bar() 은 막대그래프를 출력.

- 첫번째 매개변수에 x축의 값을 리스트나 넘파이 배열로 전달.
- 두번째 매개변수에 막대의 y축 값을 리스트나 넘파이 배열로 전달.
- width 매개변수에서 막대의 두께 지정 가능. 기본값은 0.8

가중치 시각화

: 합성곱 층의 가중치를 이미지로 출력하는 것.

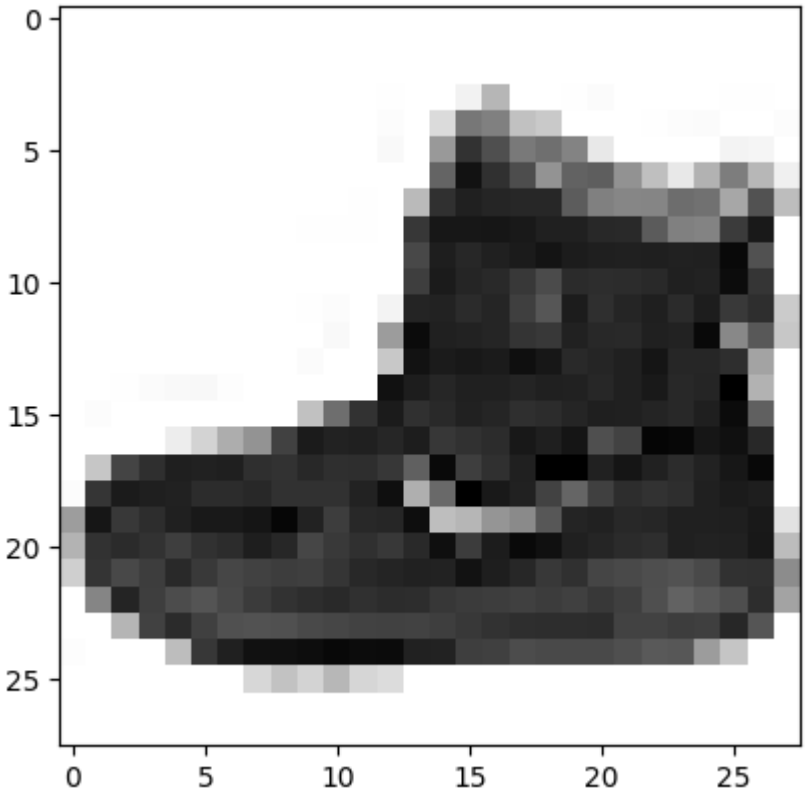
특성 맵 시각화

: 합성곱 층의 활성화 출력을 이미지로 그리는 것.

- 가중치 시각화와 함께 비교하여 각 필터가 이미지의 어느 부분을 활성화시키는지 확인할 수 있음.

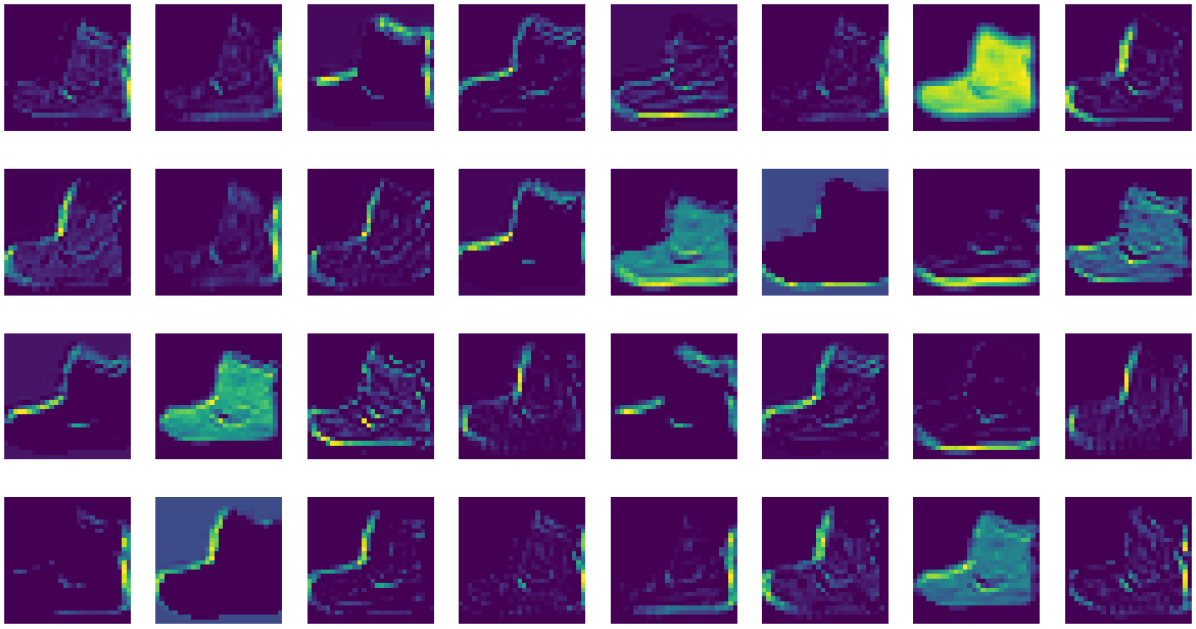
```
(train_input, train_target), (test_input, test_target) = kera
-> Downloading data from https://storage.googleapis.com/tenso
```

```
29515/29515 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorfl
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/st
Downloading data from https://storage.googleapis.com/tensorfl
5148/5148 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorfl
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

plt.imshow(train_input[0], cmap='gray_r')
plt.show()
```

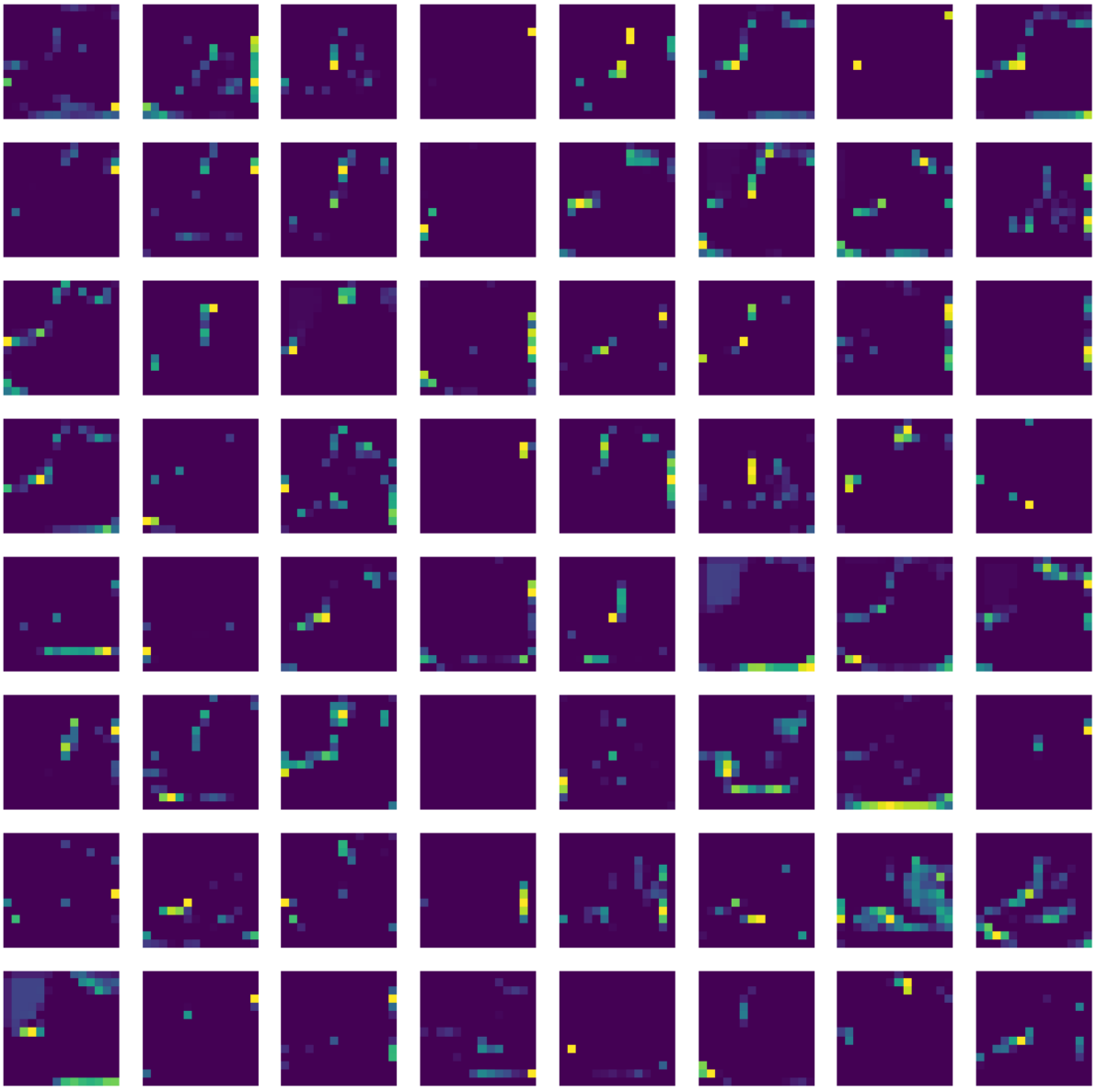



```
print(feature_maps.shape)
-> (1, 28, 28, 32)

fig, axs = plt.subplots(4, 8, figsize=(15,8))

for i in range(4):
```

```
    for j in range(8):
        axs[i, j].imshow(feature_maps[0,:,:,i*8 + j])
        axs[i, j].axis('off')

plt.show()
```



```
print(feature_maps.shape)
-> (1, 14, 14, 64)

fig, axs = plt.subplots(8, 8, figsize=(12,12))

for i in range(8):
    for j in range(8):
        axs[i, j].imshow(feature_maps[0,:,:,i*8 + j])
        axs[i, j].axis('off')

plt.show()
```



함수형 API

: 신경망 모델을 만드는 방법 중 하나. Model 클래스에 모델의 입력과 출력을 지정함.

- 전형적으로 입력은 Input() 함수를 사용하여 정의하고, 출력은 마지막 층의 출력으로 정의

```
print(model.inputs)
-> [<KerasTensor shape=(None, 28, 28, 1), dtype=float32, spar

conv_acti = keras.Model(model.inputs, model.layers[0].output)
```

TensorFlow

- Mode : 케라스 모델을 만드는 클래스.
- 첫 번째 매개변수인 inputs에 모델의 입력 또는 입력의 리스트를 지정.
- 두 번째 매개변수인 outputs에 모델의 출력 또는 출력의 리스트를 지정.
- name 매개변수에 모델의 이름 지정.