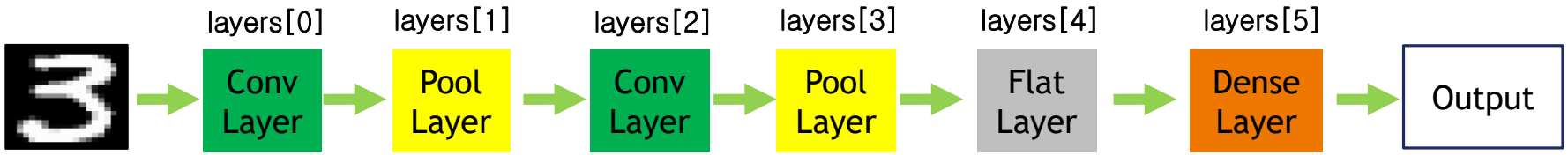
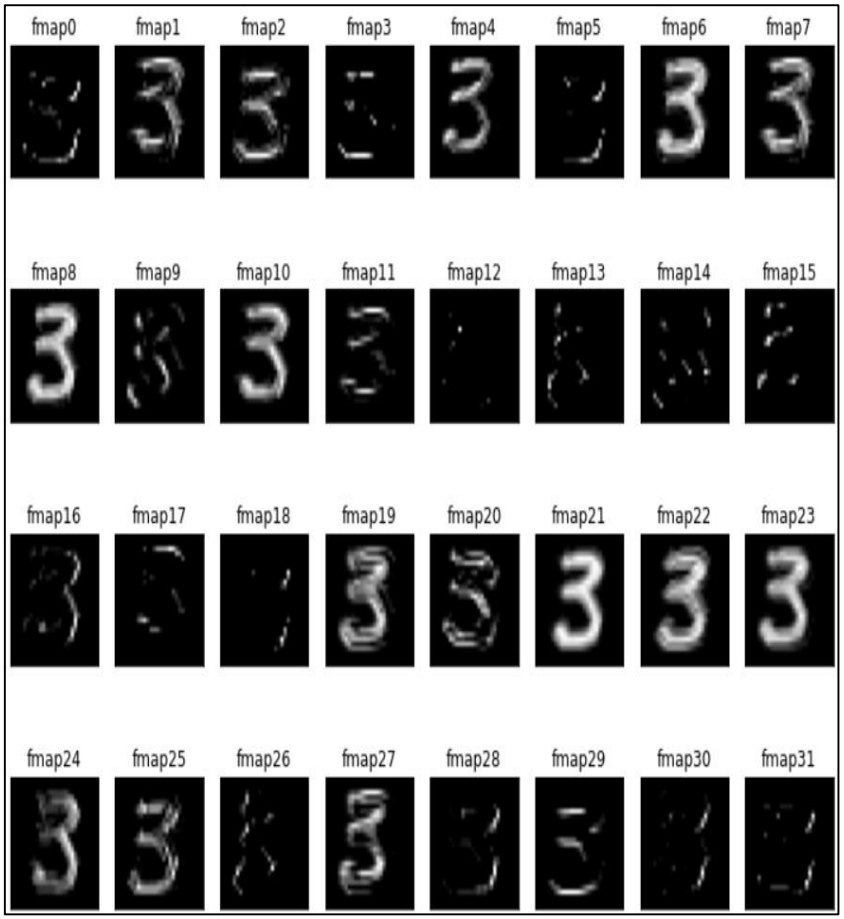
- CNN 특징맵 · 풀링맵 시각화 -

박성호 (neowizard2018@gmail.com)

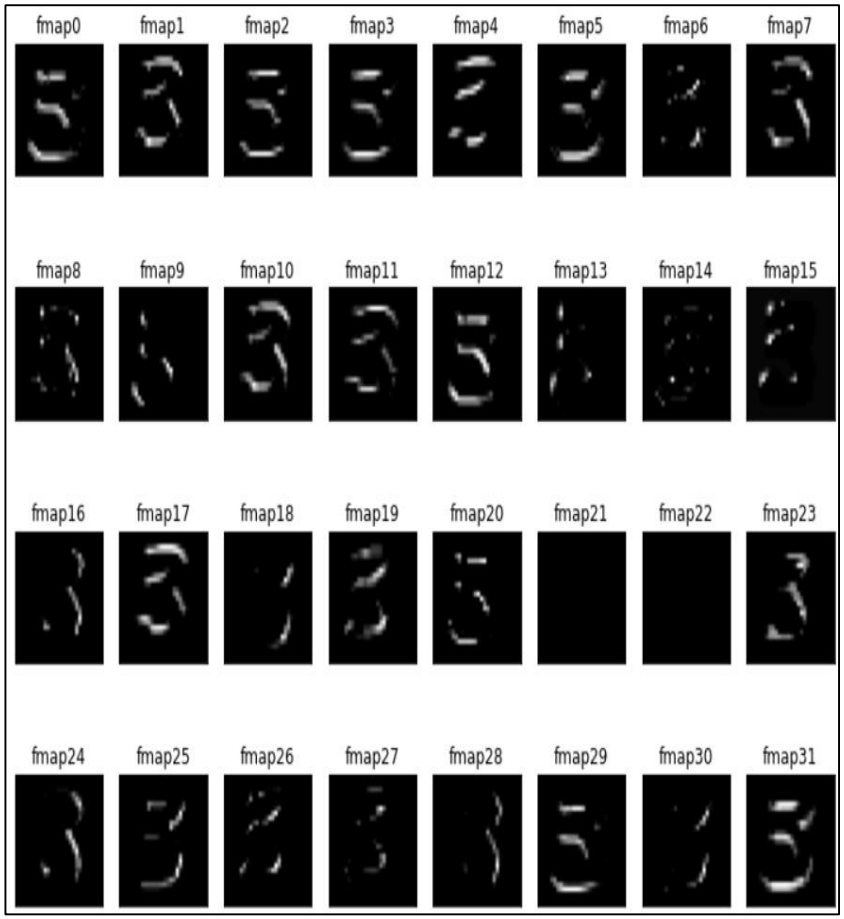
CNN 구조에서 컨볼루션 층의 특징맵(feature map)시각화



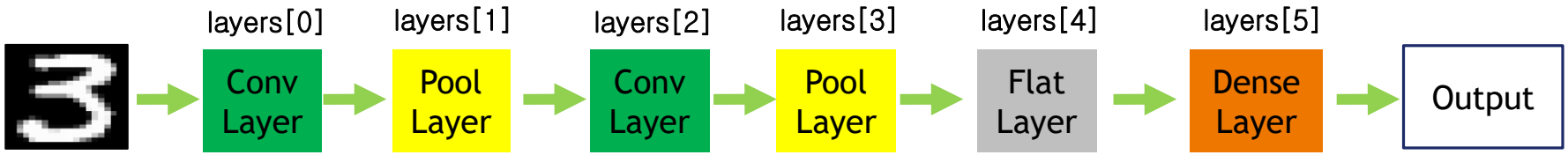
학습전 layer[0] 특징맵 시각화



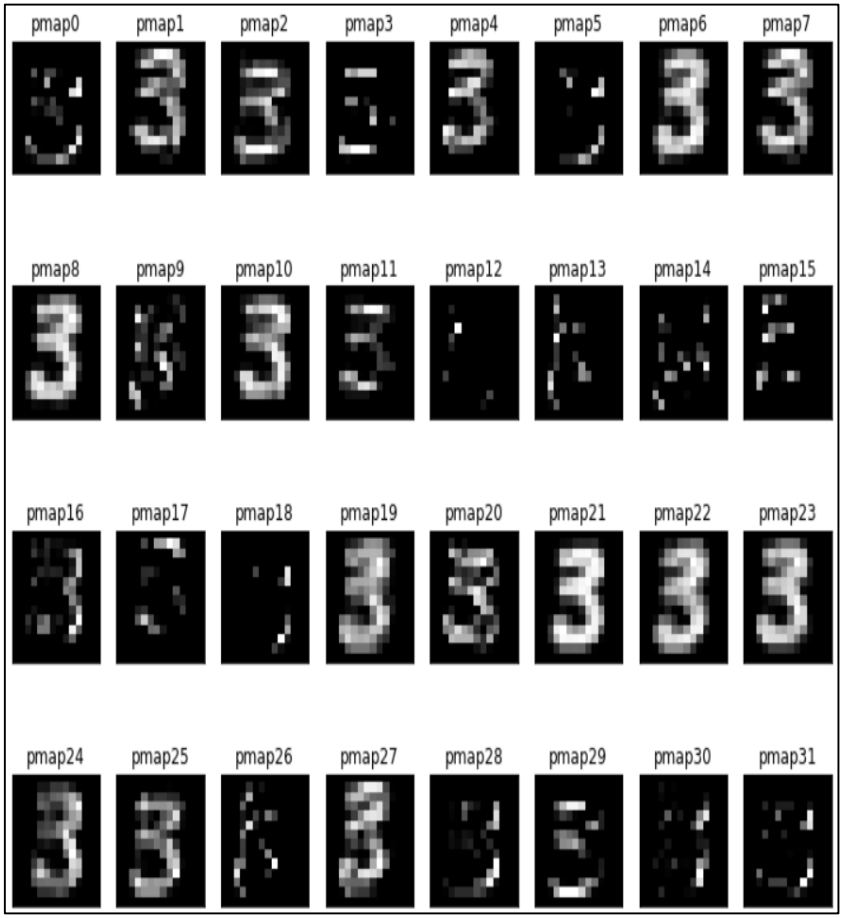
학습후 layer[0] 특징맵 시각화



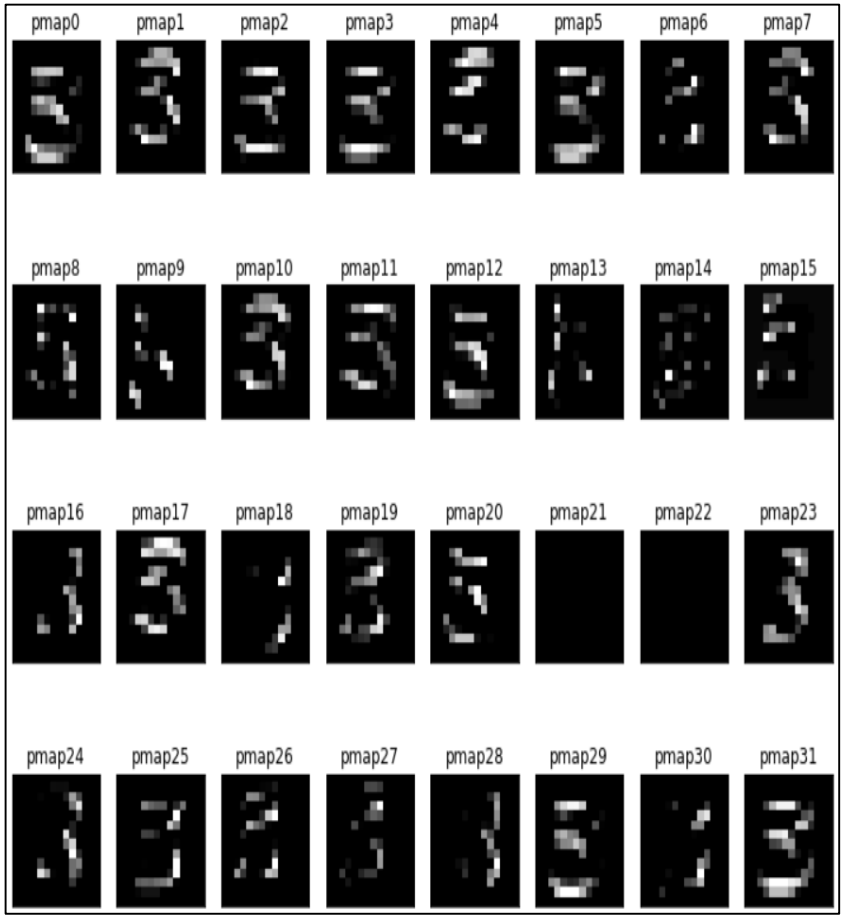
CNN 구조에서 풀링층의 풀링맵(pooling map)시각화



학습전 layer[1] 풀링맵 시각화



학습후 layer[1] 풀링맵 시각화



MNIST Dataset에 대한 CNN 시각화

```
import tensorflow as tf

from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D, Dropout
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
```

```
(x_train, y_train), (x_test, y_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 = ', y_train.shape, ' , t_test.shape = ', y_test.shape)

x_train = x_train.reshape(-1,28,28,1)
x_test = x_test.reshape(-1,28,28,1)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train.shape =  (60000, 28, 28)  , x_test.shape =  (10000, 28, 28)
t_train.shape =  (60000,)  , t_test.shape =  (10000,)
```

MNIST Dataset에 대한 CNN 시각화

MNIST Data를 학습 조건이 다음과 같을 때,

sequential model / functional model 을 각각 구축 하시오

---

feature extractor 조건

1st conv : 3 x 3 x 32 filter, padding='SAME', activation='relu', stride=(1,1)
1st pooling : 2 x 2 maxpooling, padding='SAME'
Dropout(0.25)

2nd conv : 3 x 3 x 64 filter, padding='SAME', activation='relu', stride=(1,1)
2nd pooling : 2 x 2 maxpooling, padding='SAME'
Dropout(0.25)

3rd conv : 3 x 3 x 128 filter, padding='SAME', activation='relu', stride=(1,1)
3rd pooling : 2 x 2 maxpooling, padding='SAME'
Dropout(0.25)

---

compile 조건

optimizer는 Adam(), loss function 은 sparse_categorical_crossentropy

MNIST Dataset에 대한 feature map 시각화

```
for idx in range(len(model.layers)):

    print('model.layers[%d] = %s, %s' % (idx, model.layers[idx].name, model.layers[idx].output.shape))
```

```
model.layers[0] = conv2d, (None, 28, 28, 32)
model.layers[1] = max_pooling2d, (None, 14, 14, 32)
model.layers[2] = dropout, (None, 14, 14, 32)
model.layers[3] = conv2d_1, (None, 14, 14, 64)
model.layers[4] = max_pooling2d_1, (None, 7, 7, 64)
model.layers[5] = dropout_1, (None, 7, 7, 64)
model.layers[6] = conv2d_2, (None, 7, 7, 128)
model.layers[7] = max_pooling2d_2, (None, 4, 4, 128)
model.layers[8] = dropout_2, (None, 4, 4, 128)
model.layers[9] = flatten, (None, 2048)
model.layers[10] = dense, (None, 10)
```

즉 위와 같은 모델이 생성되었을때, 1st conv 층 출력을 가지는 partial_model 을 sequential model 과 functional model을 바탕으로 각각 생성하시오

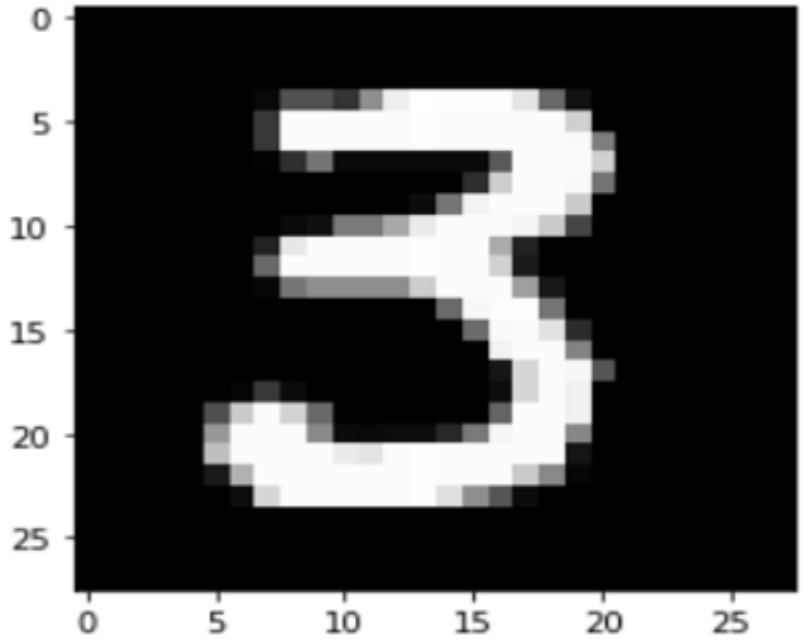
즉 partial_model = Model(inputs=..., outputs=....) 형태로 각각 생성하시오

MNIST Dataset에 대한 CNN 시각화

아래와 같이 임의의 MNIST data 1개를 임의로 선택하시오

```
random_idx = np.random.randint(0, len(x_test))

print(random_idx)
plt.imshow(x_test[random_idx].reshape(28,28), cmap='gray')
```

```
8856
<matplotlib.image.AxesImage at 0x7f395cff4490>
```

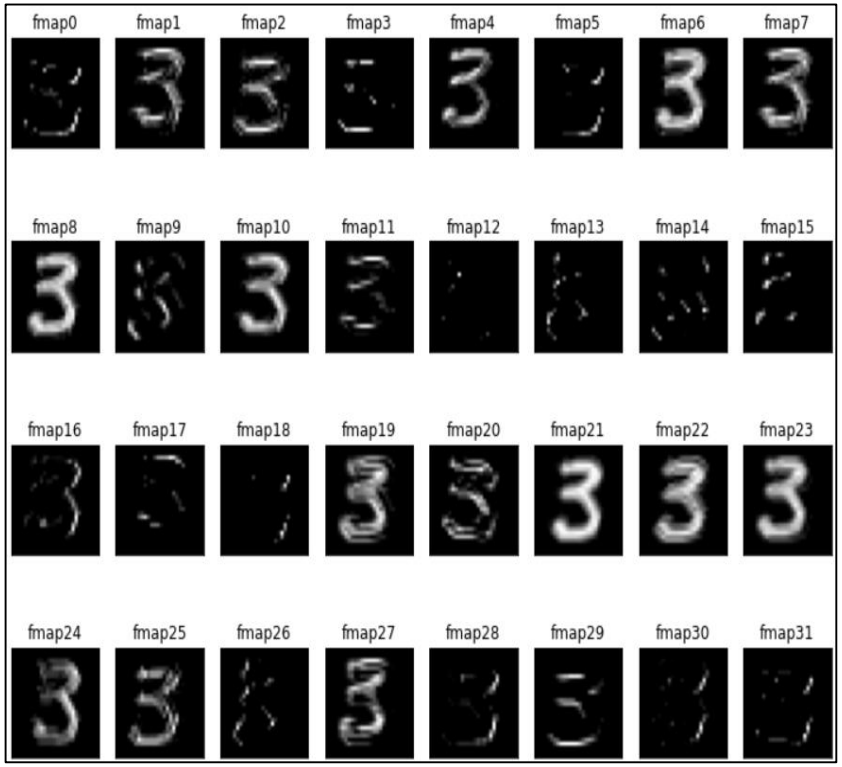


학습 전 layer[0] 특징맵 시각화

```
feature_map = partial_model.predict(x_test[random_idx].reshape(-1,28,28,1))

print(feature_map.shape)

fm = feature_map[0]  # 0번 이미지의 특징 맵을 시각화

print(fm.shape)
```

```
(1, 28, 28, 32)
(28, 28, 32)
```

```
plt.figure(figsize=(10, 8))

for i in range(32): # i번째 특징 맵

    plt.subplot(4,8,i+1)

    plt.imshow(fm[:,:,i], cmap='gray')

    plt.xticks([]); plt.yticks([])
    plt.title("fmap"+str(i))

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



MNIST Dataset에 대한 pooling map 시각화

```
for idx in range(len(model.layers)):

    print('model.layers[%d] = %s, %s' % (idx, model.layers[idx].name, model.layers[idx].output.shape))
```

```
model.layers[0] = conv2d, (None, 28, 28, 32)
model.layers[1] = max_pooling2d, (None, 14, 14, 32)
model.layers[2] = dropout, (None, 14, 14, 32)
model.layers[3] = conv2d_1, (None, 14, 14, 64)
model.layers[4] = max_pooling2d_1, (None, 7, 7, 64)
model.layers[5] = dropout_1, (None, 7, 7, 64)
model.layers[6] = conv2d_2, (None, 7, 7, 128)
model.layers[7] = max_pooling2d_2, (None, 4, 4, 128)
model.layers[8] = dropout_2, (None, 4, 4, 128)
model.layers[9] = flatten, (None, 2048)
model.layers[10] = dense, (None, 10)
```

즉 위와 같은 모델이 생성되었을때 1st pooling 층 출력을 가지는 partial_model 을 sequential model 과 functional model을 바탕으로 각각 생성하시오

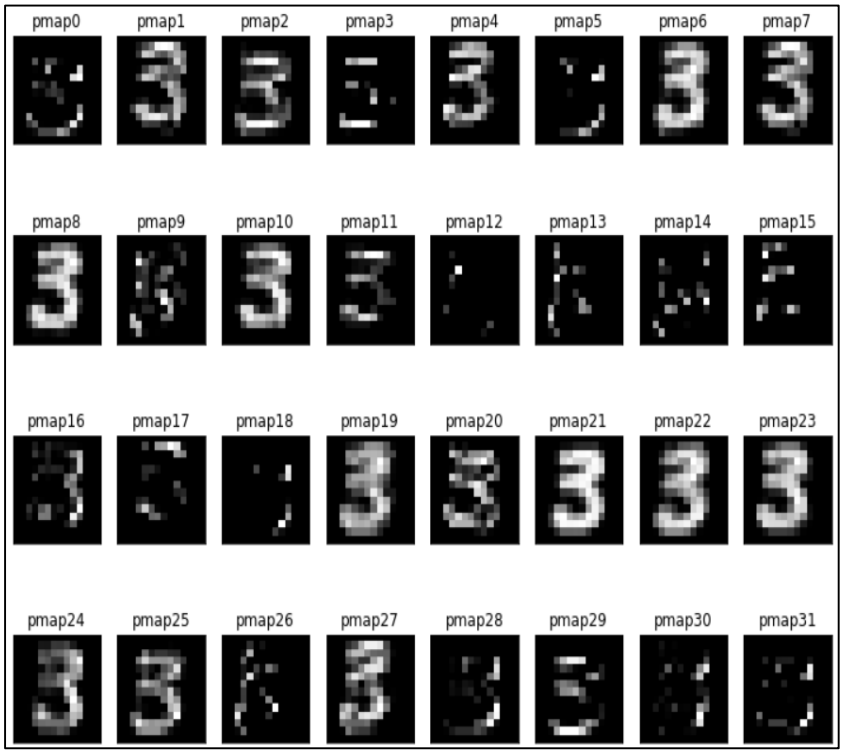
즉 partial_model = Model(inputs=..., outputs=....) 형태로 각각 생성하시오

학습 전 layer[1] 풀링맵 시각화

```
pooling_map = partial_model.predict(x_test[random_idx].reshape(-1,28,28,1))

print(pooling_map.shape)

pm = pooling_map[0]  # 0번 이미지의 풀링 맵을 시각화

print(pm.shape)
```

```
(1, 14, 14, 32)
(14, 14, 32)
```

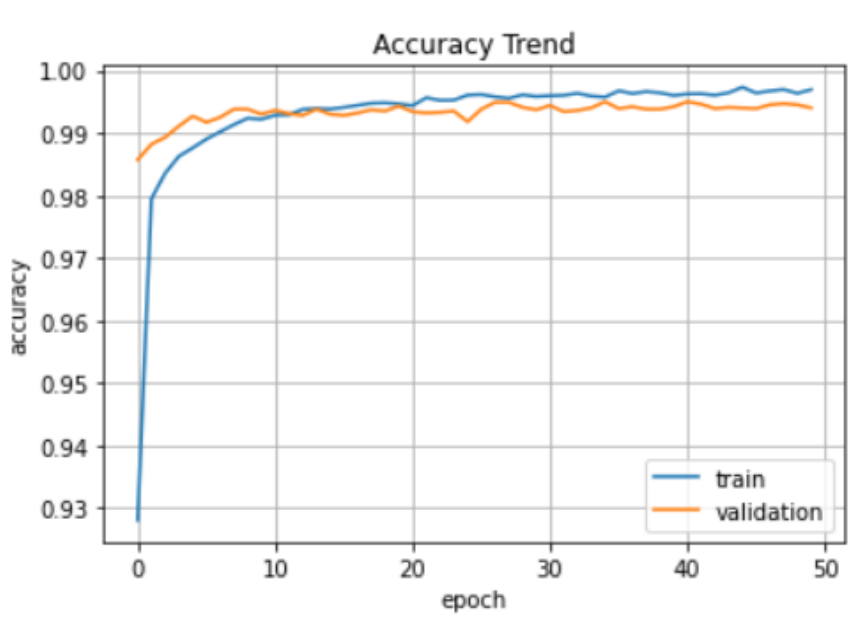
```
plt.figure(figsize=(10, 8))

for i in range(32): # i번째 풀링 맵

    plt.subplot(4,8,i+1)

    plt.imshow(pm[:,:,i], cmap='gray')

    plt.xticks([]); plt.yticks([])
    plt.title("pmap"+str(i))

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



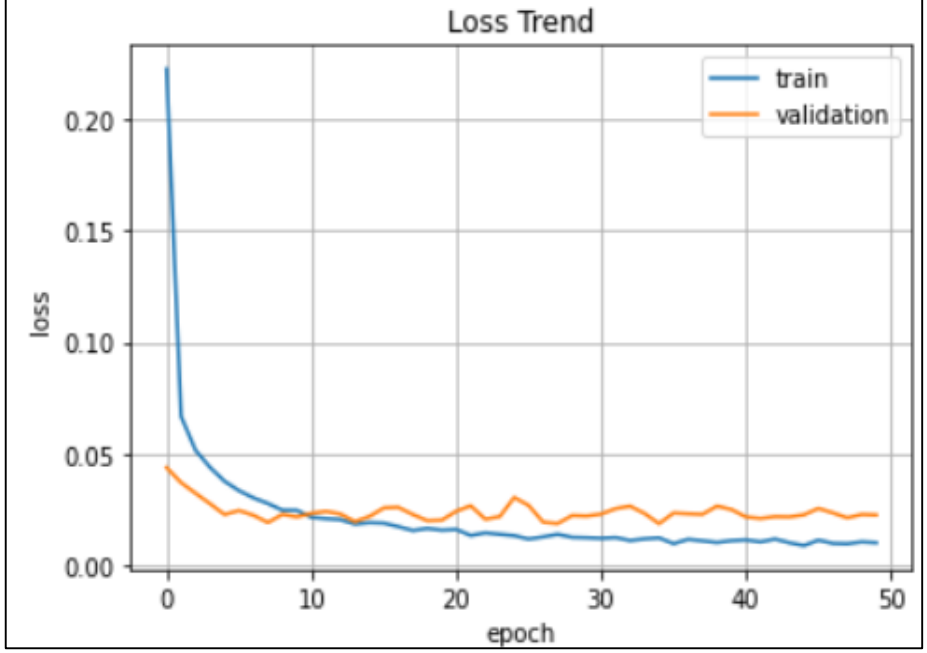
학습 및 오버피팅 확인

```
hist = model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_test, y_test))
```

```
import matplotlib.pyplot as plt

plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Accuracy Trend')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='best')
plt.grid()
plt.show()
```




```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Loss Trend')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='best')
plt.grid()
plt.show()
```



[Self Study 1] 학습을 모두 마친 후, 1st feature map, 1st pooling map 시각화 하는 코드를 작성하시오

[Self Study 2] 학습을 모두 마친 후, 2nd feature map, 2nd pooling map 시각화 하는 코드를 작성하시오