

Tensorflow를 이용한  
Deep Learning의 기초

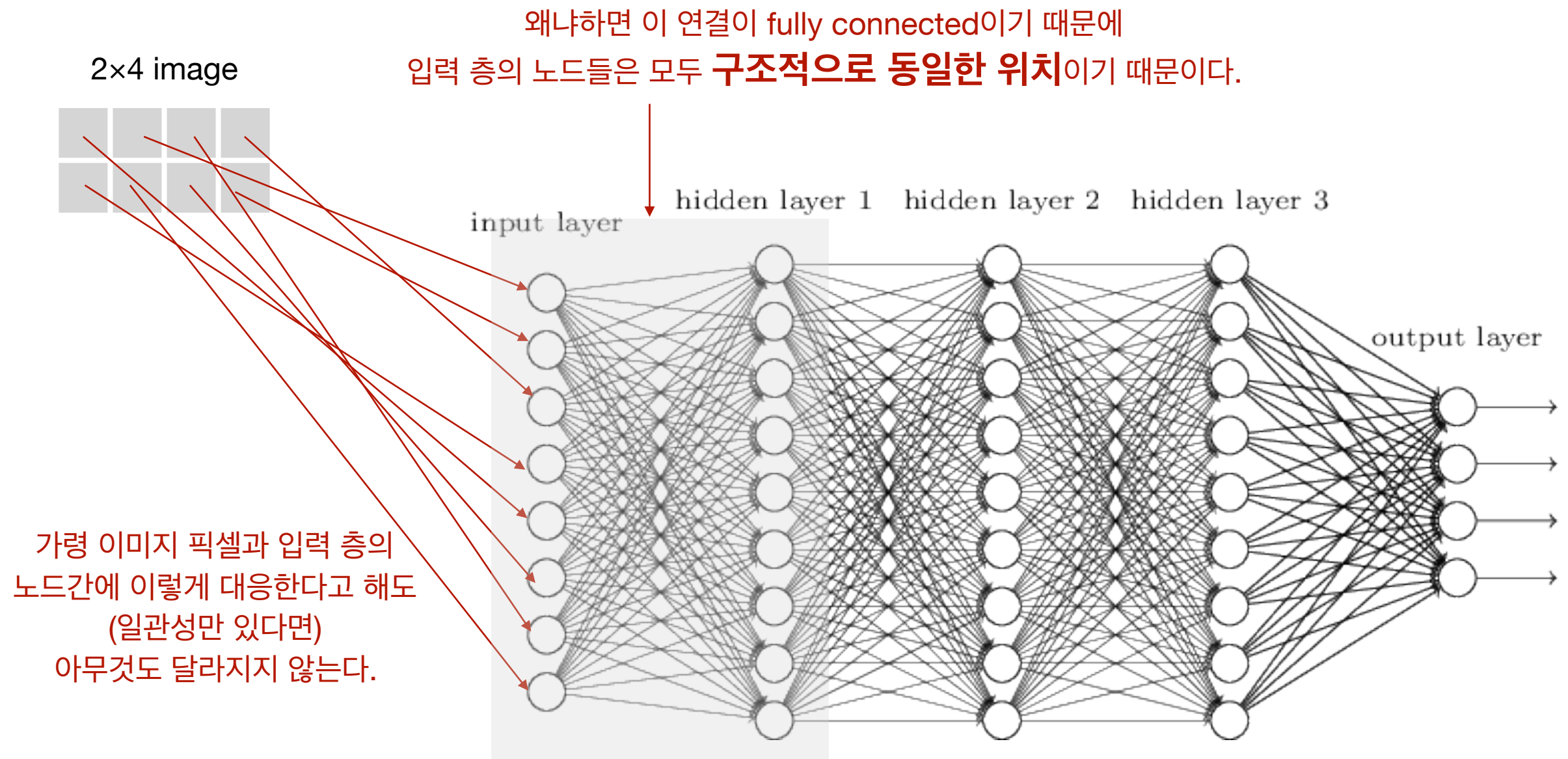
## 제3강: Convolutional Neural Network (CNN)

부경대학교 IT융합응용공학과  
권 오 흠



- 이미지는 다른 유형의 데이터와 달리 데이터 내에서 각 구성요소의 상대적 위치가 중요한 의미를 가진다.
- Fully connected layer는 이미지가 가진 지역적 특성(local feature)를 효과적으로 추출하지 못한다.

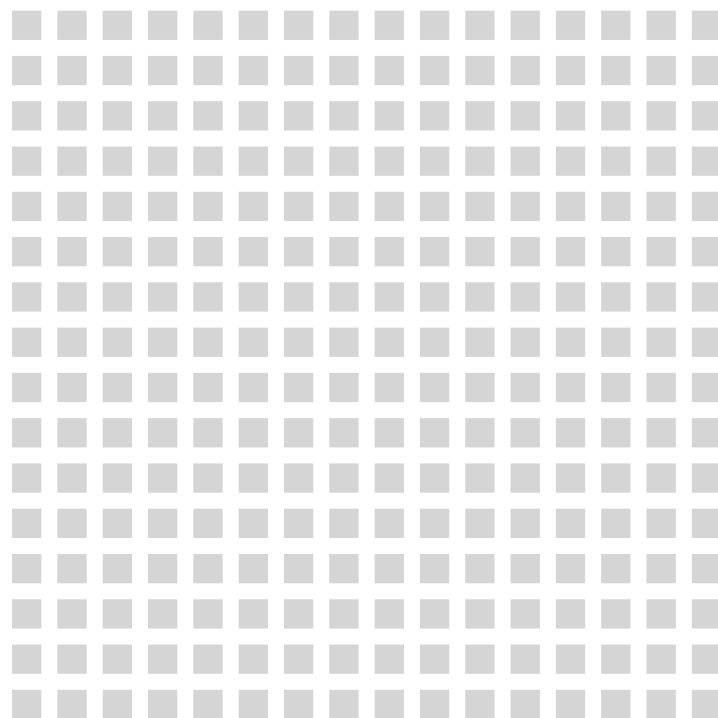
# Fully Connected Layers



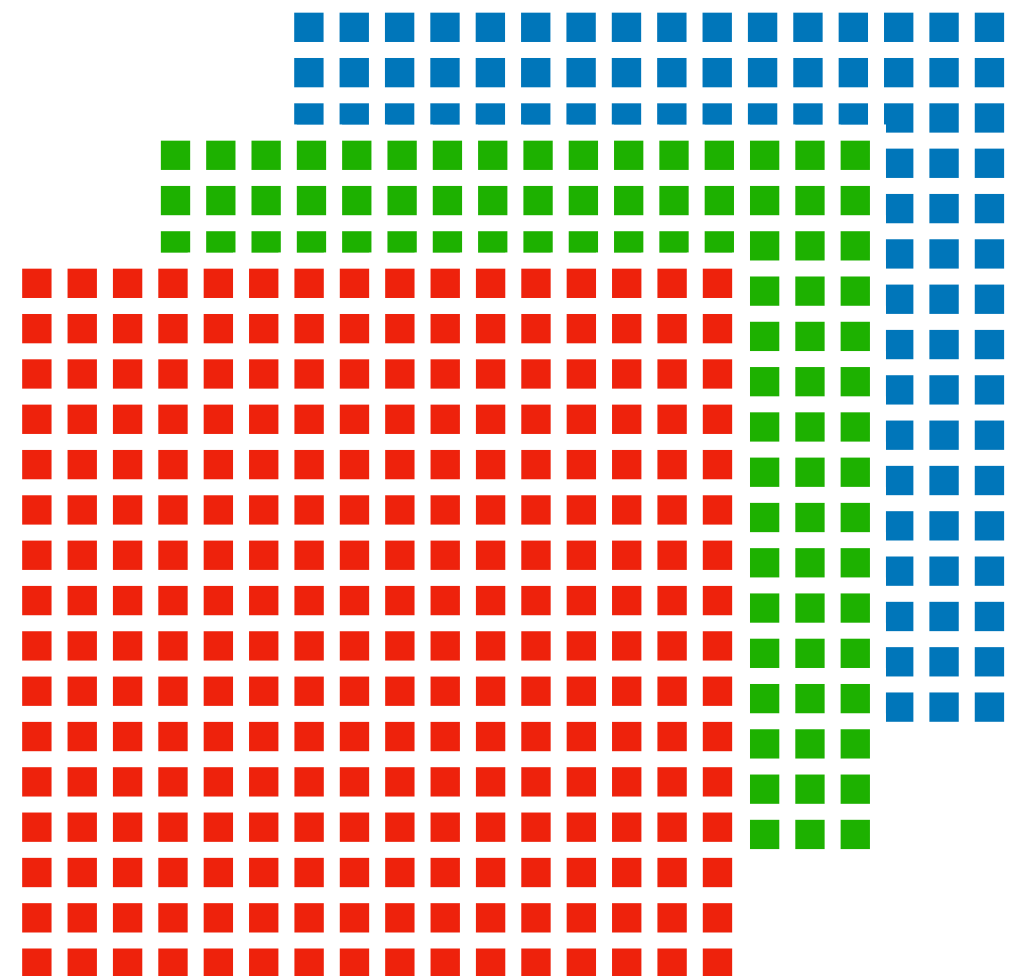
- Fully connected 네트워크는 이미지가 가진 **국부적(local) 특성**을 효과적으로 추출하지 못한다.



- 입력 이미지를 1차원 벡터로 변환하지 않고 2차원 (혹은 RGB이미지의 경우 3차원) 구조를 그대로 유지한다.

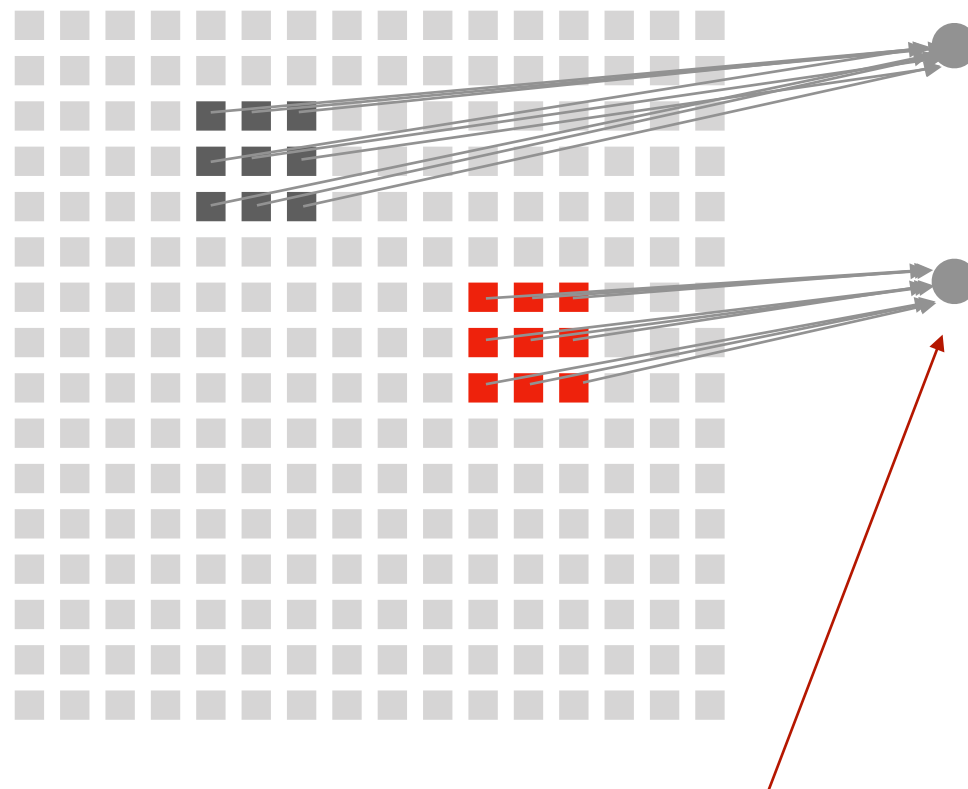


1 채널 이미지의 경우

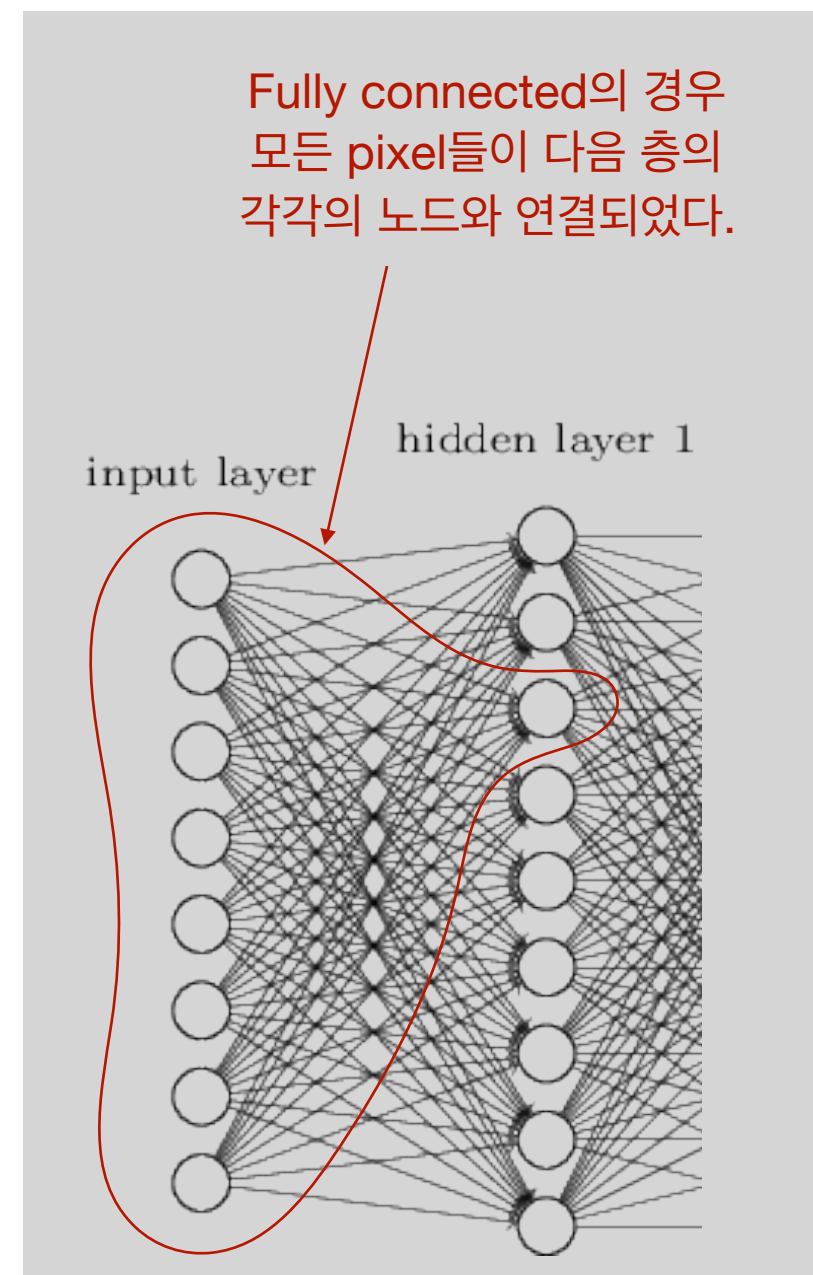


3 채널 이미지의 경우

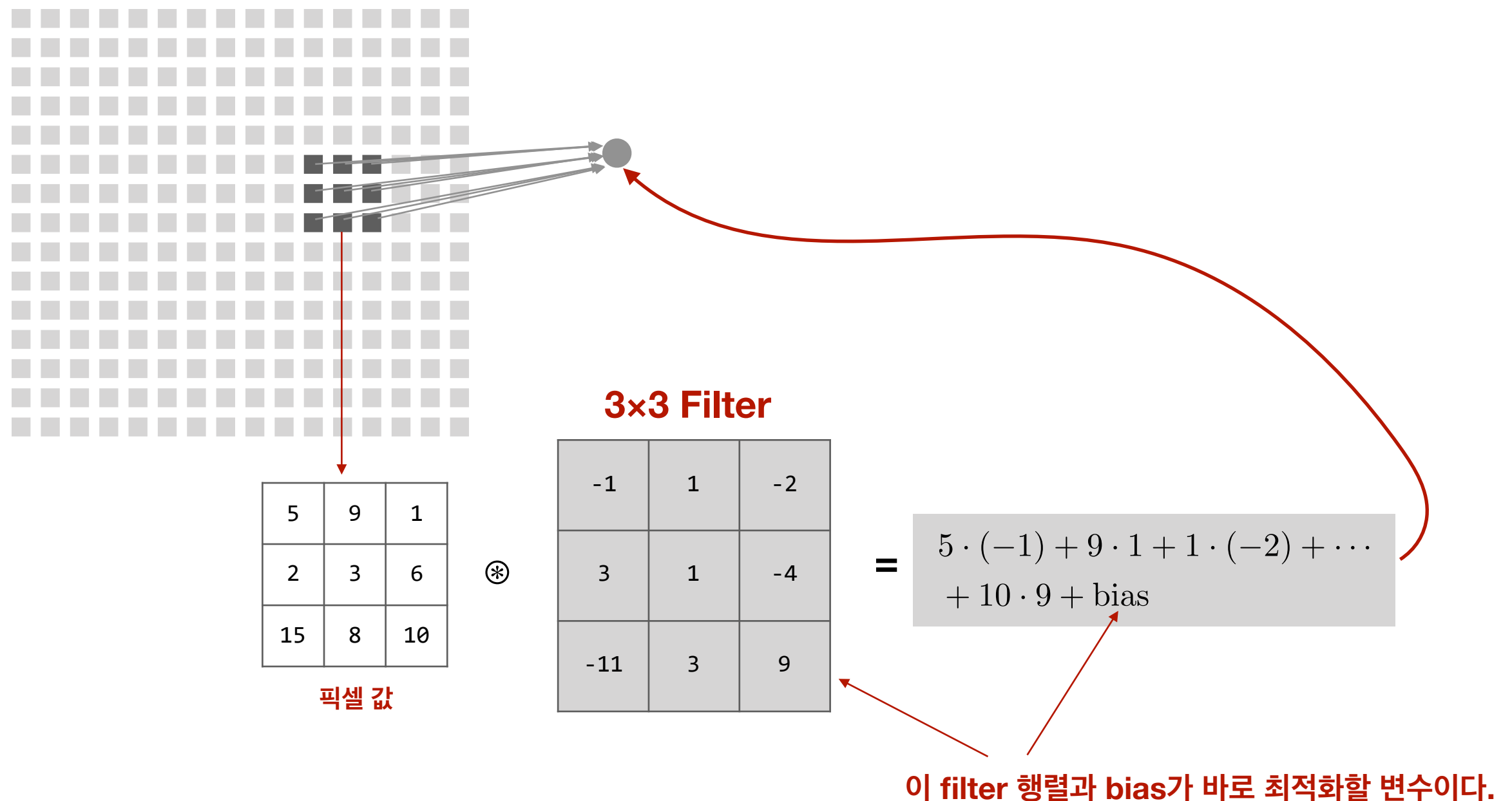
- 모든 픽셀들을 다음 layer의 각 노드에 연결하는 대신 “작은 지역적인 영역 (local receptive field)”만을 다음 층의 한 노드에 연결한다.



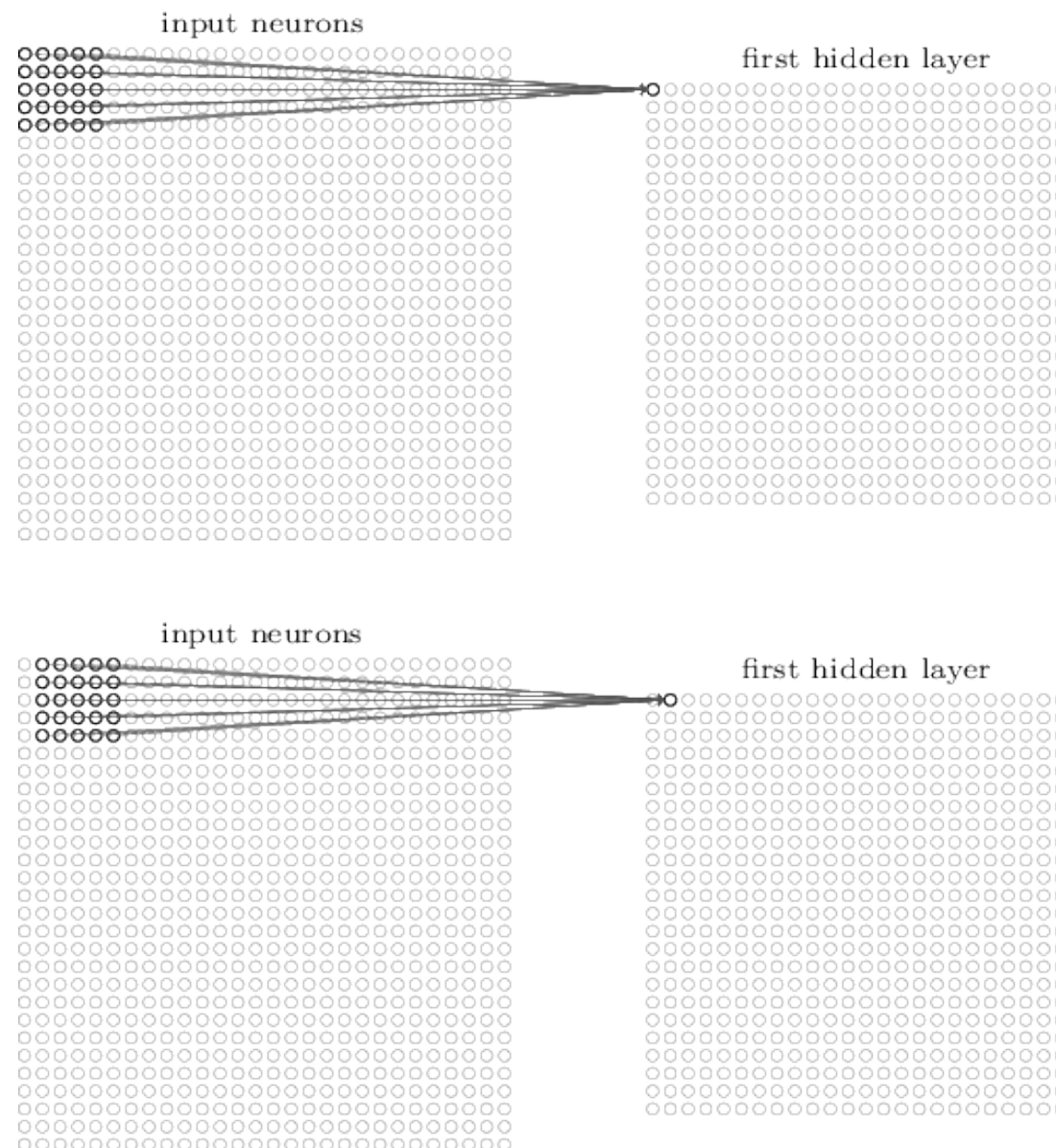
이 노드에는 빨간 9개의 픽셀만이 연결된다. 즉 이 노드는 9개의 픽셀로 구성된 국부적인 한 영역의 특징을 추출하는 역할을 한다.  
이 예에서는 3x3 크기의 영역을 사용하였다.  
보통 3x3 혹은 5x5 크기가 주로 사용된다.



- “작은 지역적인 영역(local receptive field)”의 특징을 추출하기 위해서 하나의 작은 가중치 행렬을 사용한다. 이 행렬을 보통 **필터(filter)** 혹은 **커널(kernel)**이라고 부른다.
- 이 행렬은 모든 receptive field에 대해서 **공유(shared)**된다.

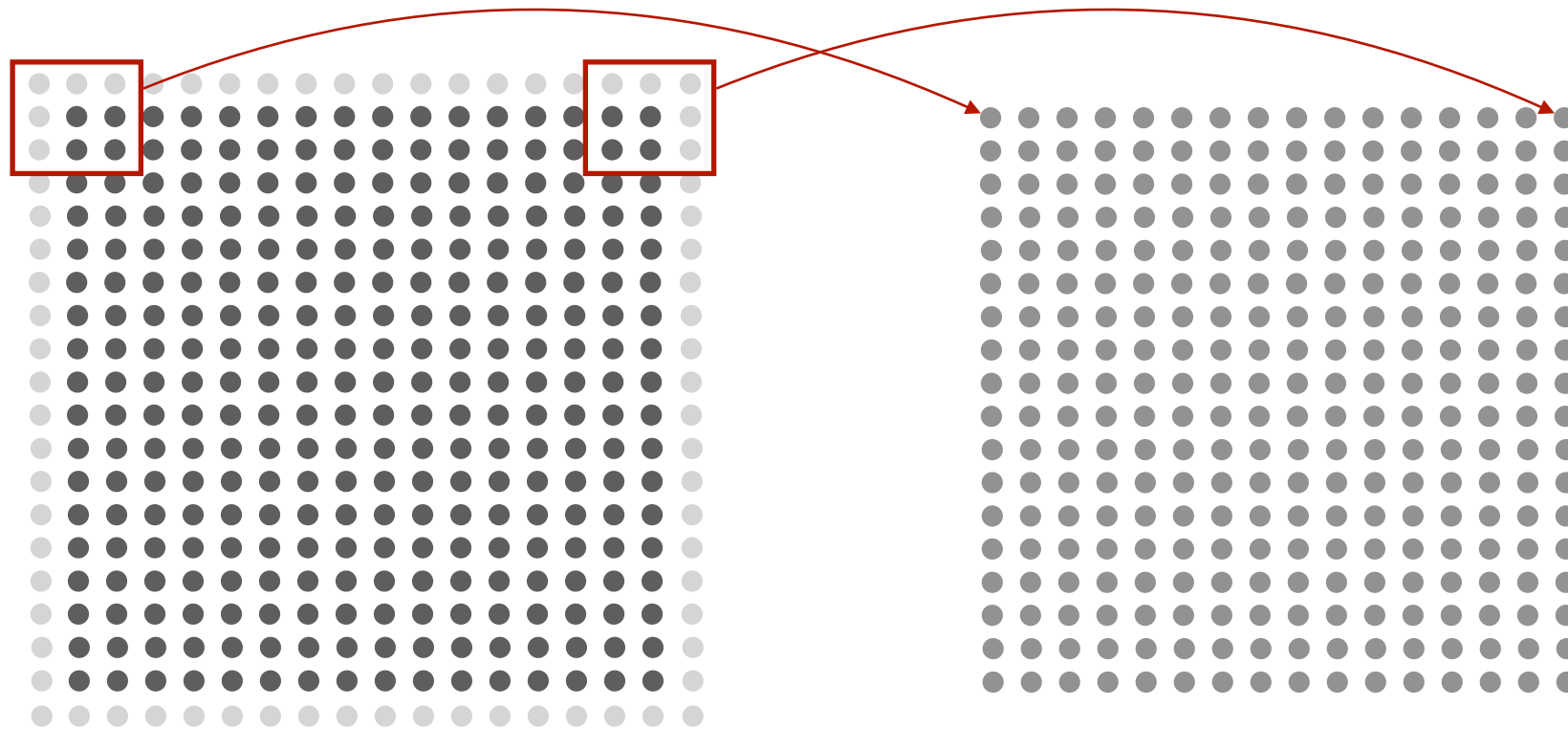


- **Receptive field를 슬라이딩 시키면서 공유된 filter를 사용해 계산한다.** 아래의 그림은 필터의 크기가 5×5인 경우이다.



원래 이미지가  $N \times M$  크기이면  
 $(N-4) \times (M-4)$  크기로 줄어든다.  
만약 필터가  $3 \times 3$ 이면  $(N-2) \times (M-2)$   
로 줄어든다.  
이것은 약간 성가시다.

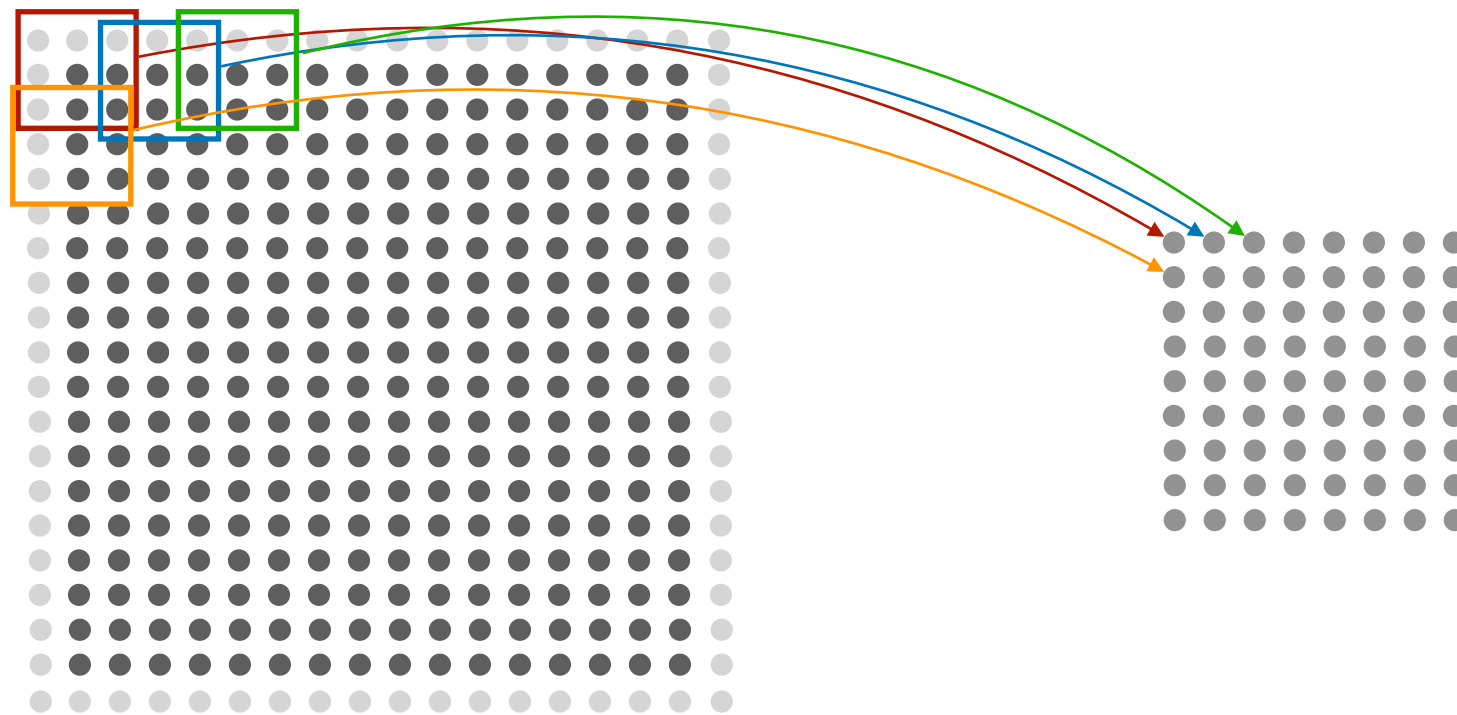
- 이미지의 상하좌우에 가상의 픽셀을 추가(padding)하여 크기가 줄어드는 것을 방지한다.  $3 \times 3$  filter를 사용한다면 1줄씩,  $5 \times 5$  filter를 사용한다면 2줄씩 추가한다. 추가된 행과 열의 픽셀값은 보통 0으로 가정하거나 혹은 인접한 실제 픽셀의 값을 사용하기도 한다.



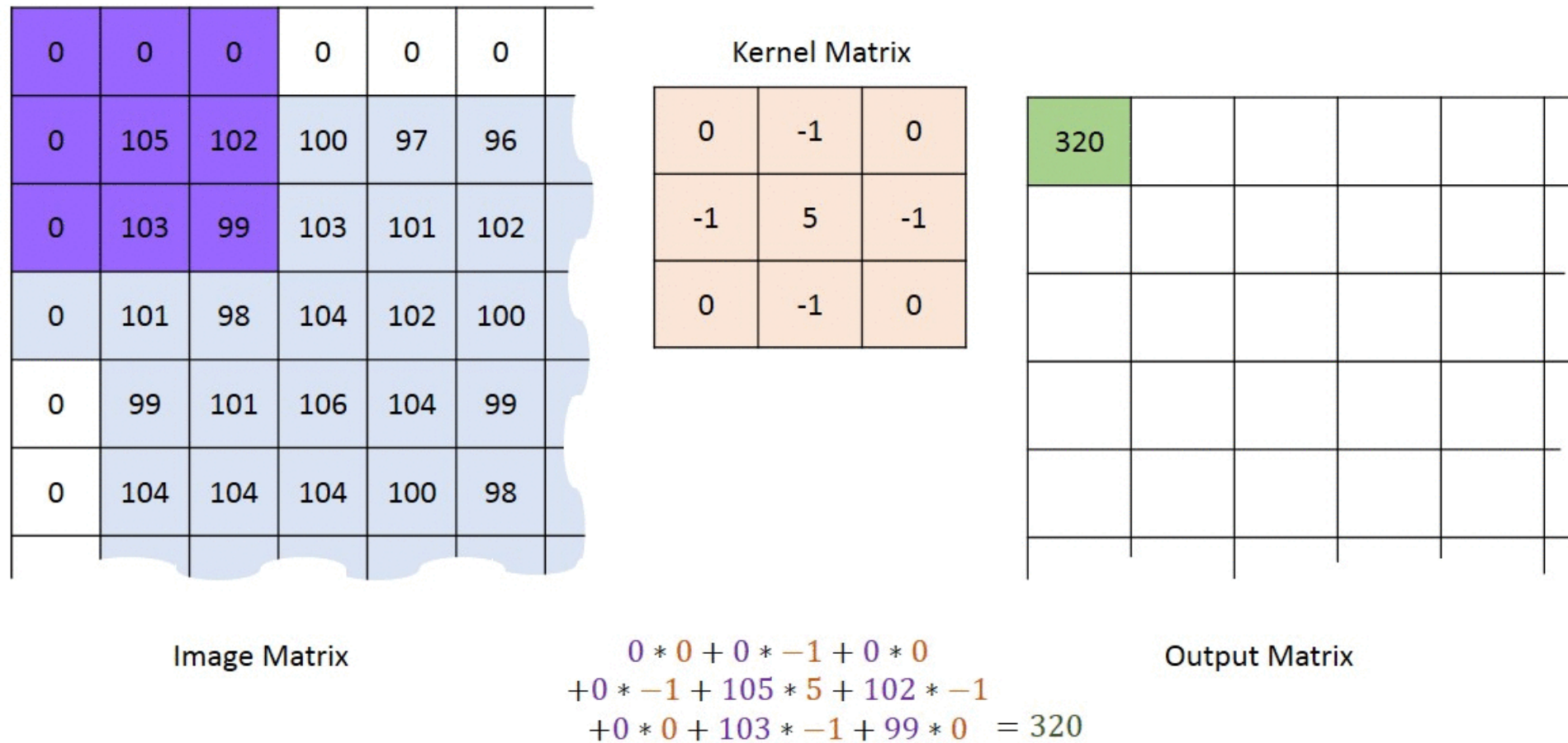
이미지의 크기가 그대로 유지된다.



- 원도우를 슬라이딩할 때 반드시 한 픽셀씩 슬라이딩 해야하는 것은 아니다. 가령 2칸씩 슬라이딩 할 수도 있다. 이 경우 이미지의 크기는 1/2로 줄어든다. 이렇게 슬라이딩하는 단위를 stride라고 부른다. 가로와 세로의 stride 값을 다르게 할 수도 있다.

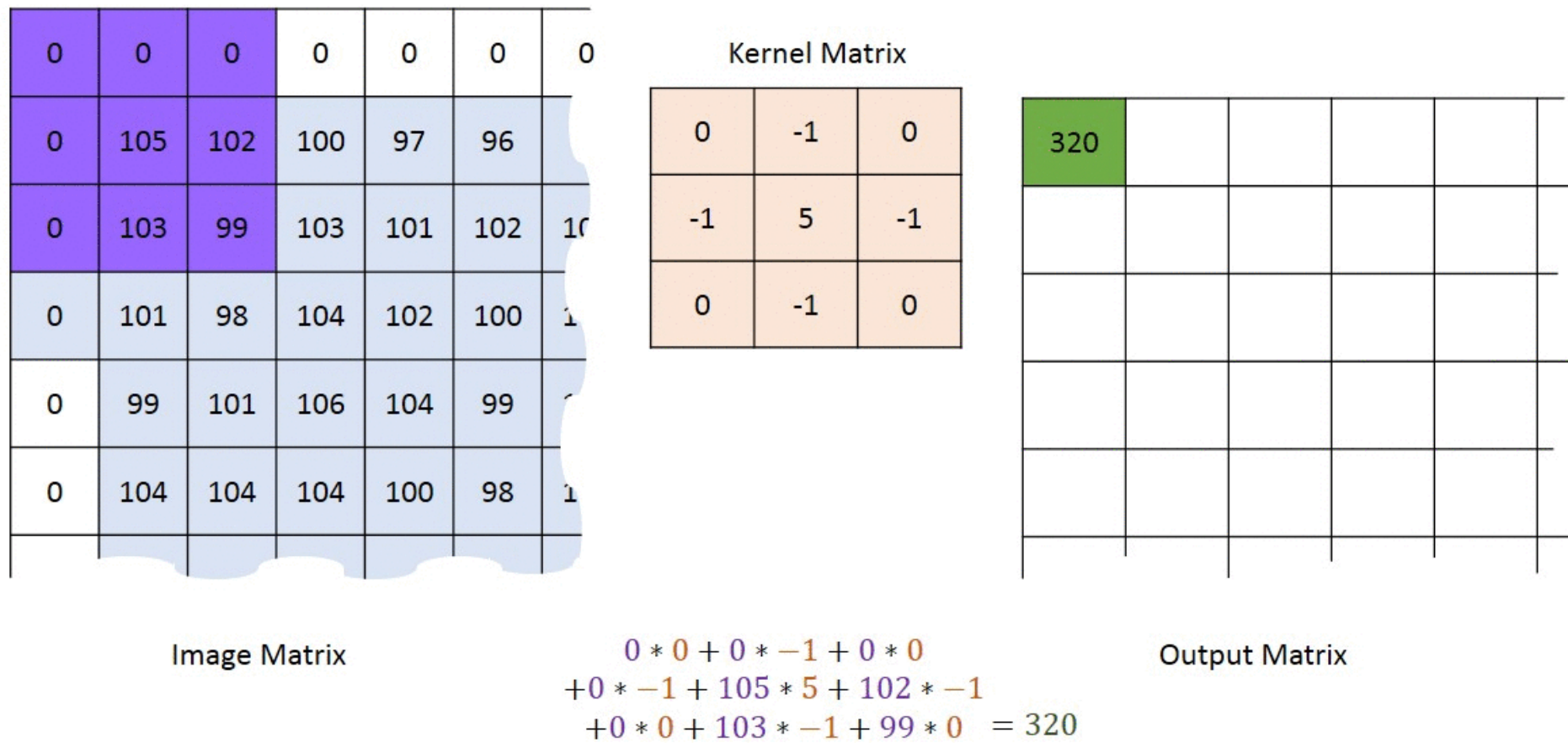


가로/세로 stride 값이 2라면 이미지의 크기가 1/2로 줄어든다.



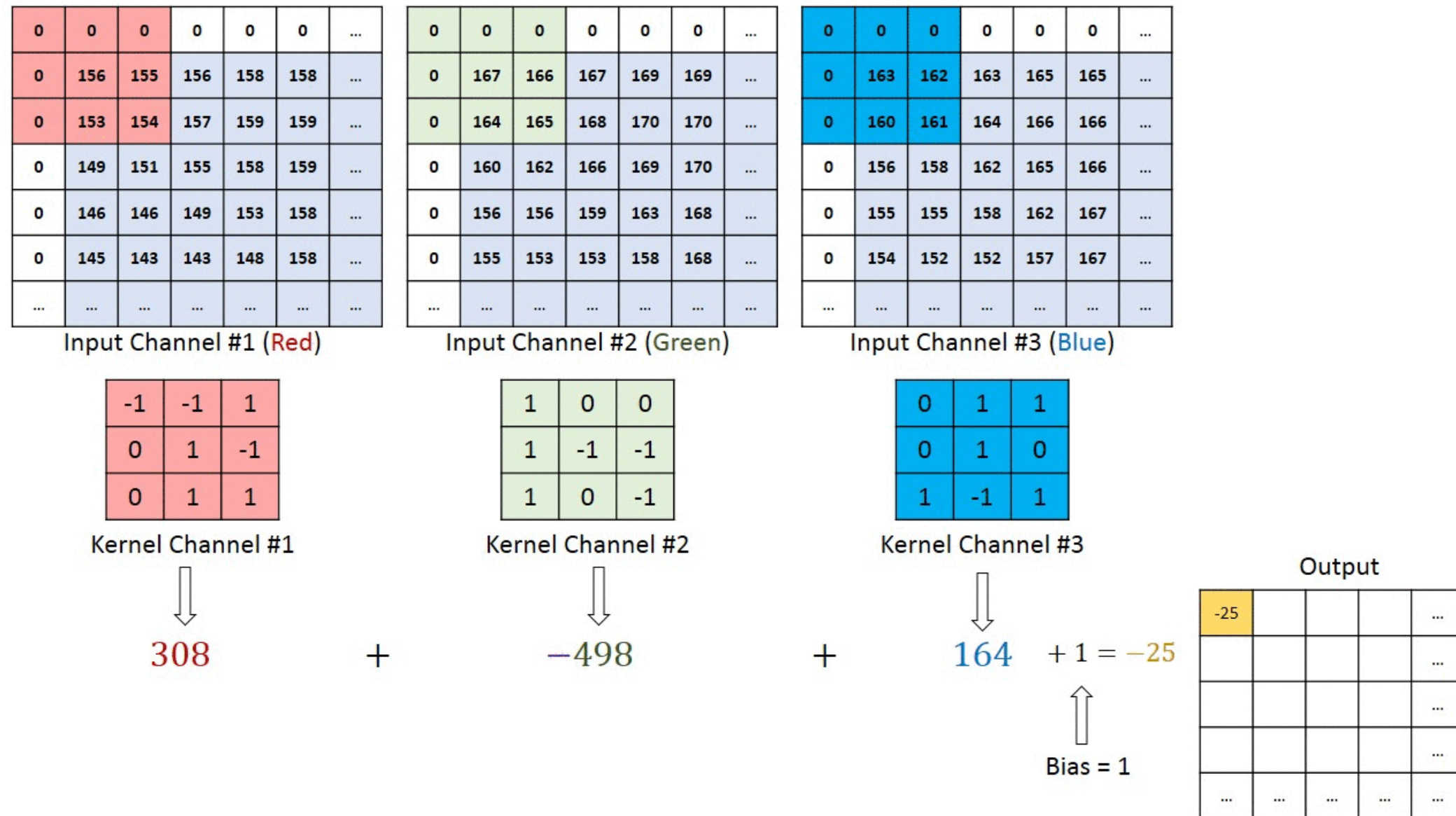
Convolution with horizontal and  
vertical strides = 1

Download and watch [this](#) GIF animation



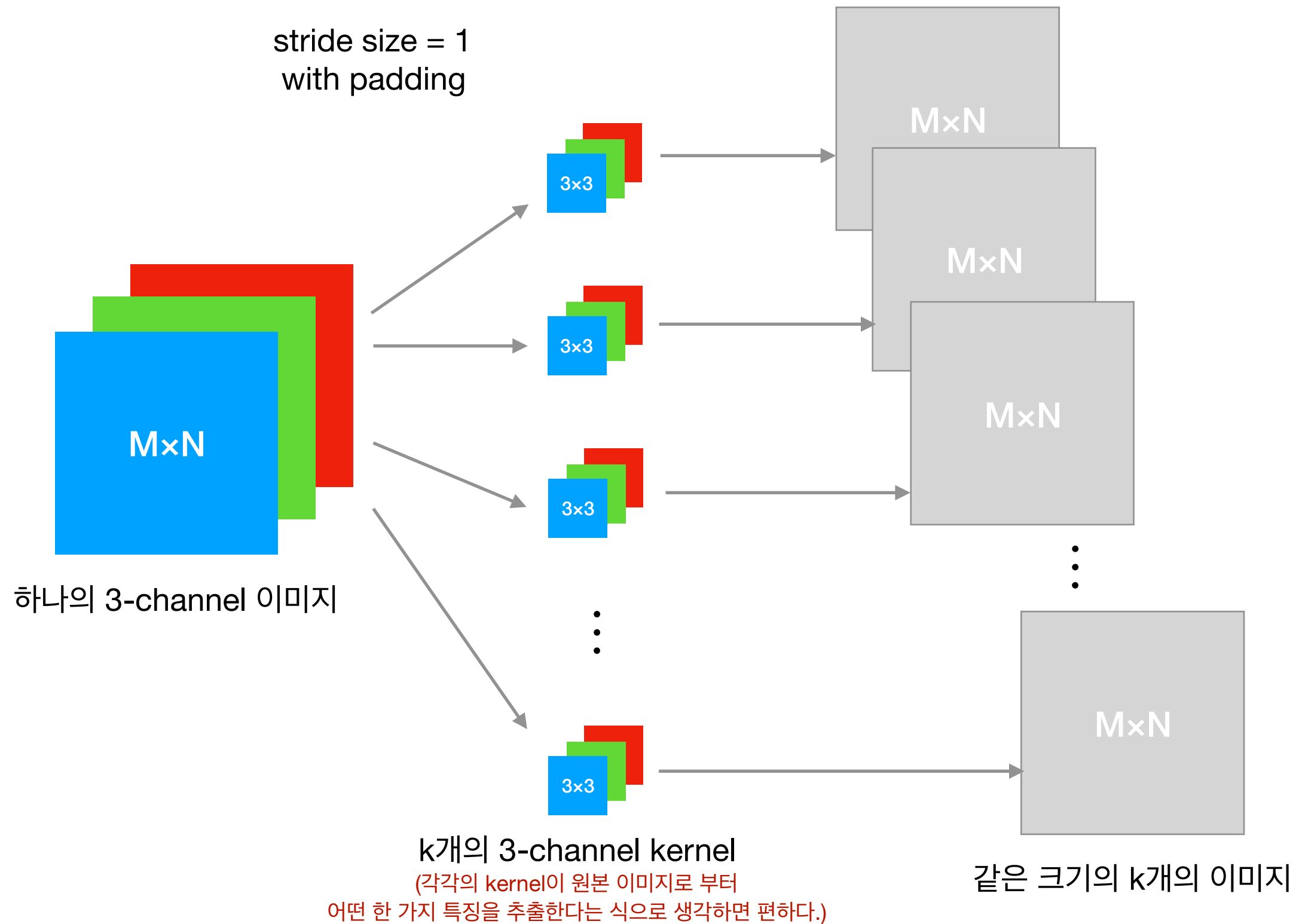
Convolution with horizontal and  
vertical strides = 2

Download and watch [this](#) GIF animation



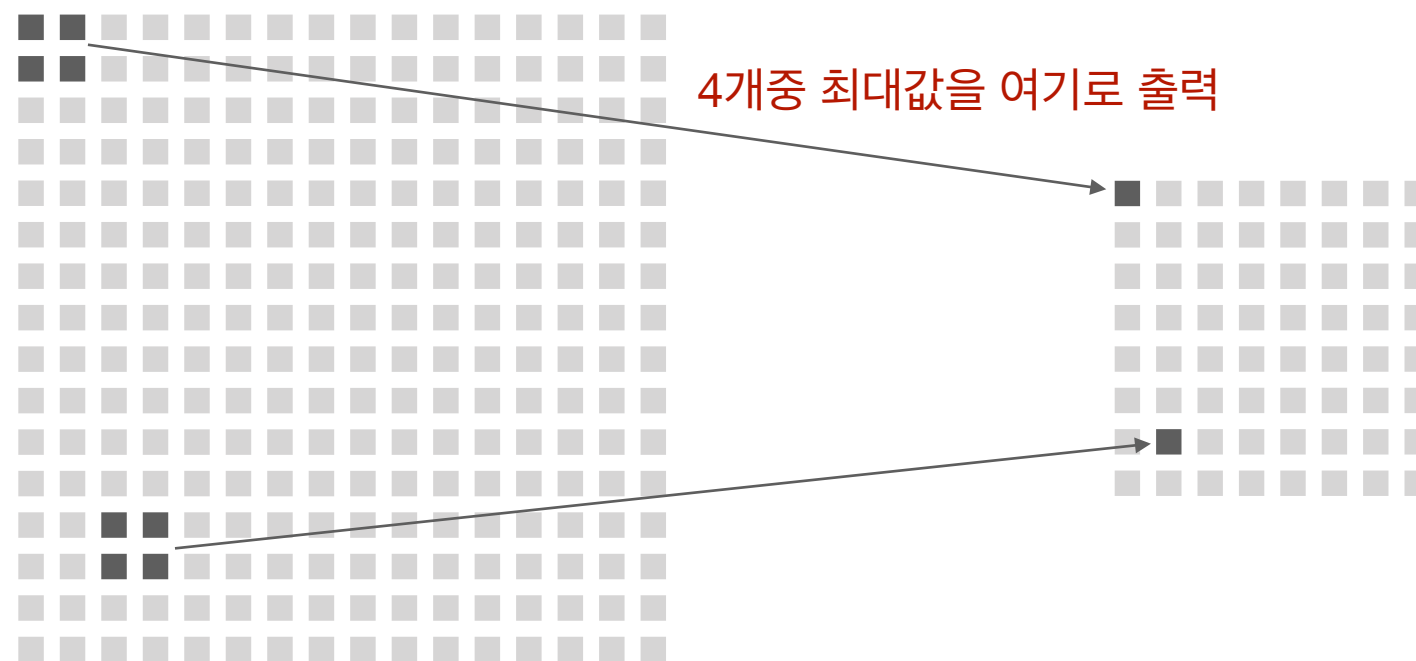
Download and watch [this](#) GIF animation

# Single Convolution Layer



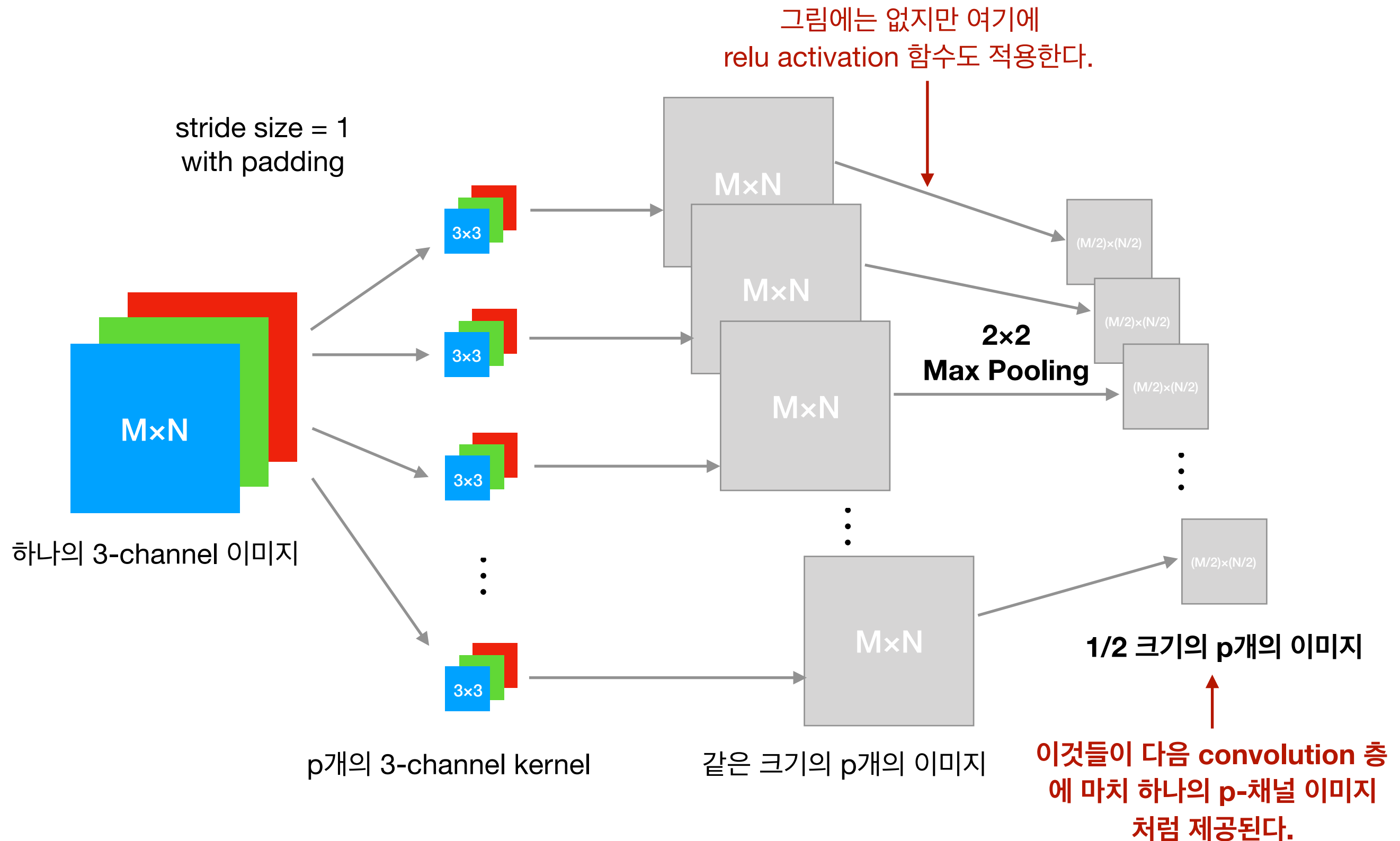


- 일반적으로 convolution layer에 뒤이어 pooling layer를 적용한다.
- Pooling layer의 역할은 convolution layer의 출력을 압축(혹은 단순화)하는 것이다.
- Pooling에는 max-pooling, L2-pooling 등이 있으며, **max-pooling**이 가장 흔하게 사용된다.
- 2x2 max pooling에서는 2x2 영역의 값 중에서 최대값을 선택한다.

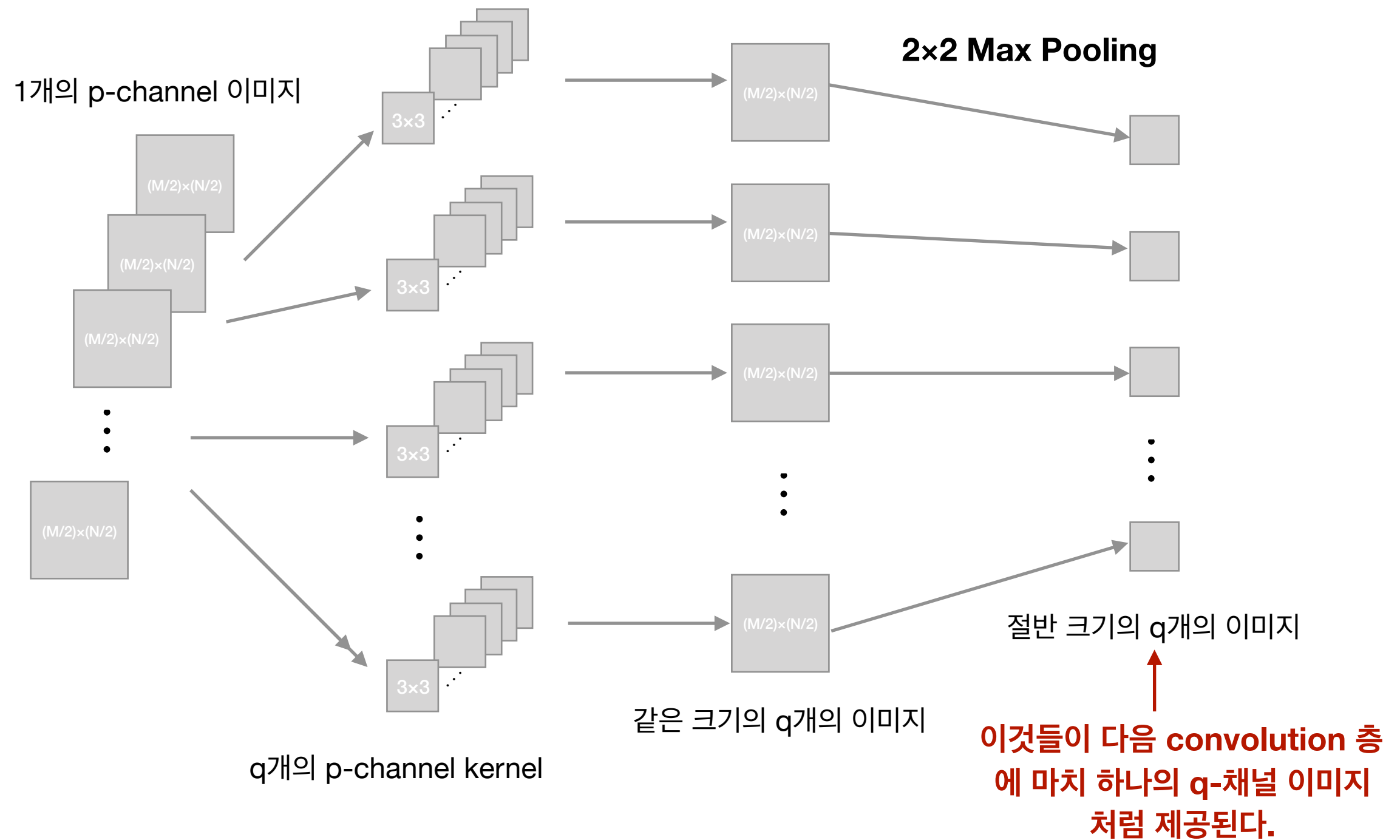


2x2 max pooling을 적용하면 이미지의 크기가 가로/세로 각각 반으로 줄어든다.

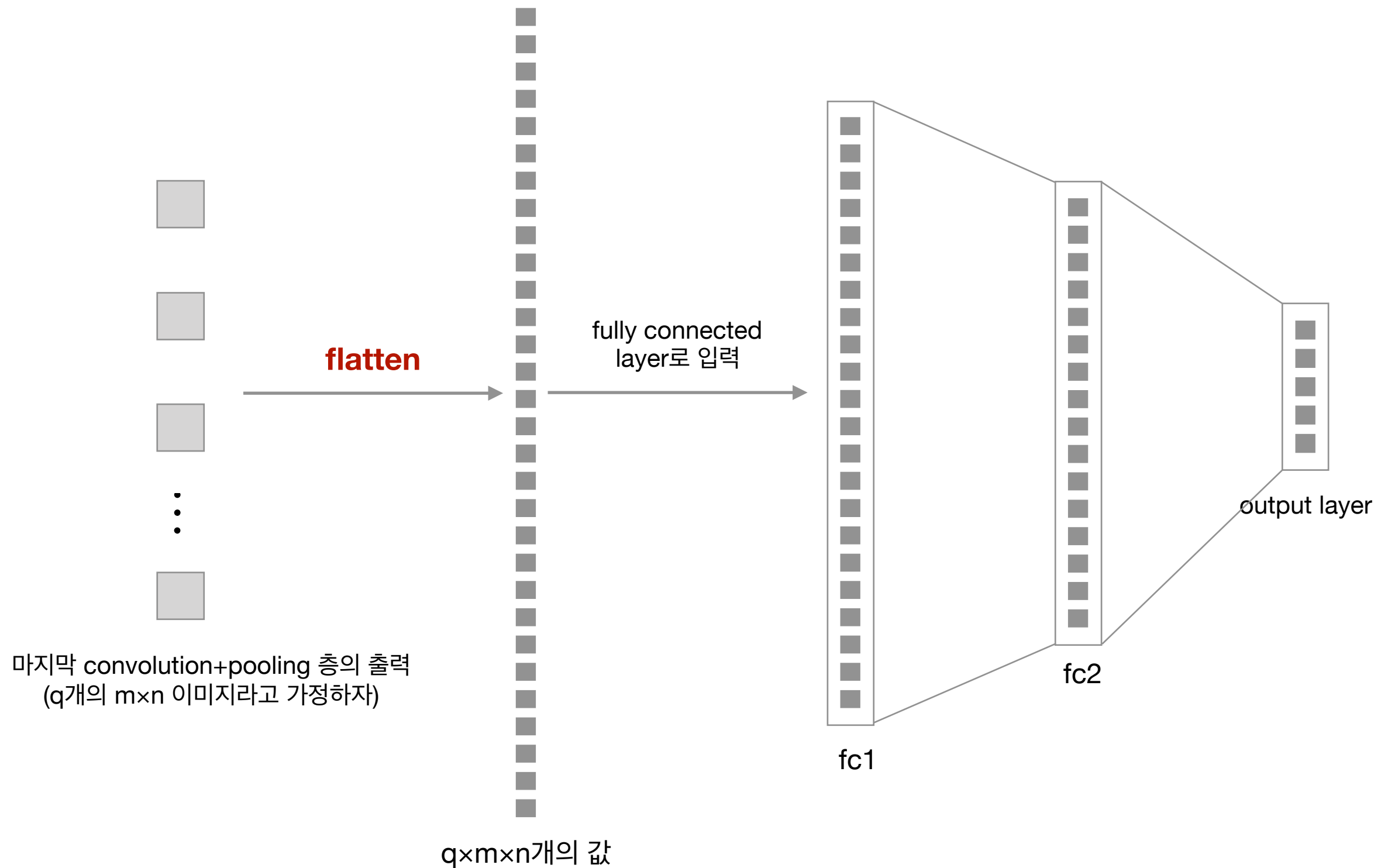
# Convolution + Max Pooling Layer

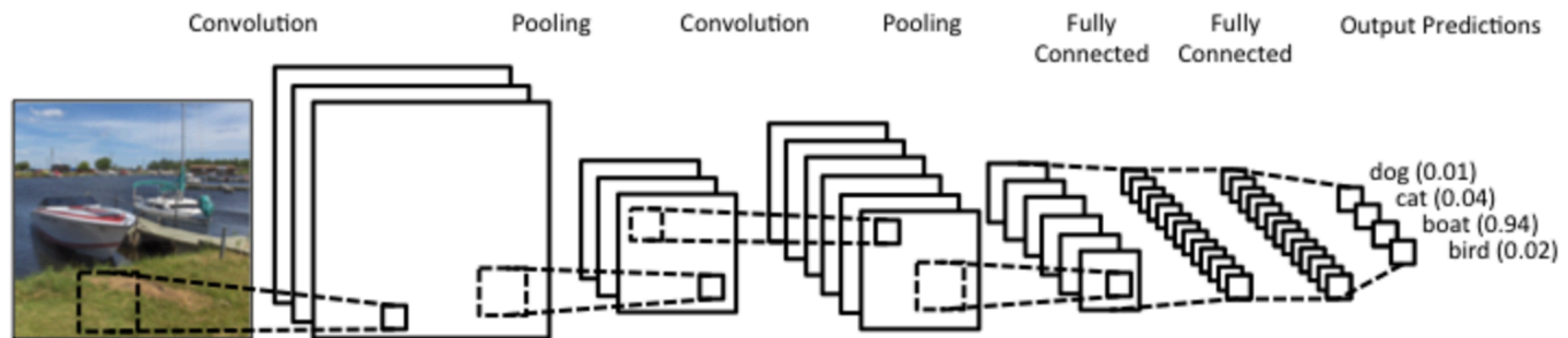


## Next (Convolution + Pooling) Layer



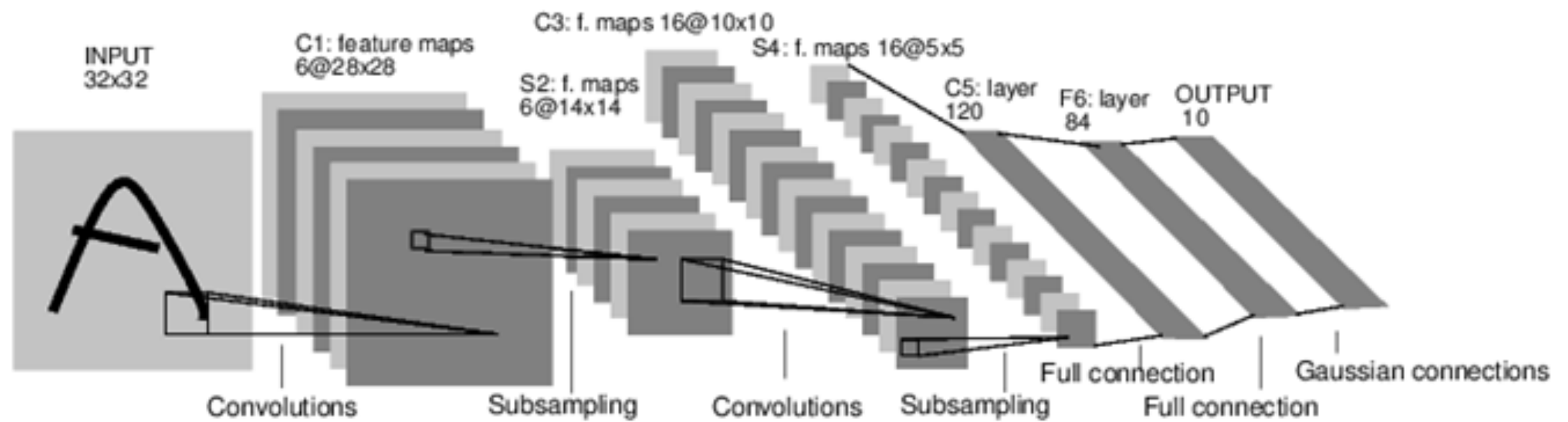
- 일반적으로 convolution + pooling 층들 뒤에 몇 개의 fully connected 층을 배치한다.





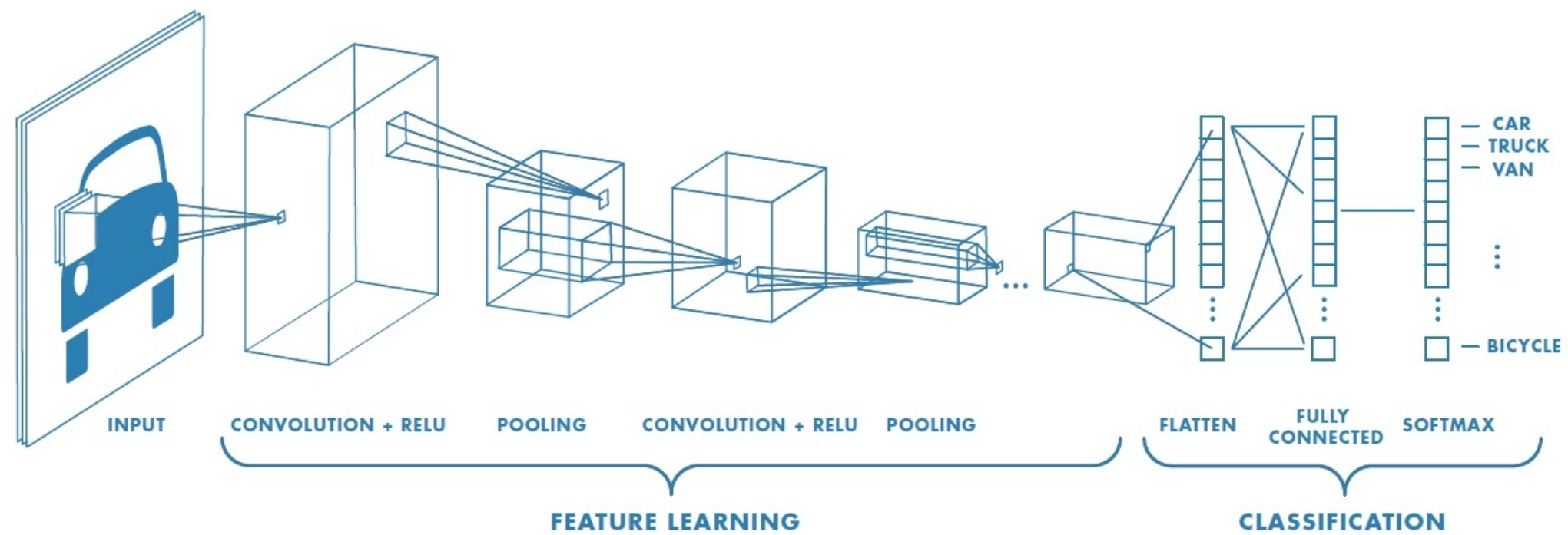
<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>





A Full Convolutional Neural Network (LeNet)

<https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>



<https://www.mathworks.com/discovery/convolutional-neural-network.html>

# CNN 구현

- 3 convolution layers with 32, 64, and 128 filters each
- one fully-connected hidden layer of 512 nodes with dropout
- output layer with softmax activation function

```
import tensorflow as tf
from tensorflow import keras
from HandleInput import *
import numpy as np

def create_model():
    model = keras.Sequential([
        keras.layers.Conv2D(32, kernel_size=3, activation='relu',
                             padding='same', input_shape=(IMG_HEIGHT, IMG_WIDTH, NUM_CHANNEL)),
        keras.layers.MaxPool2D(pool_size=(2,2), strides=(2,2), padding='same'),
        keras.layers.Conv2D(64, kernel_size=3, activation='relu', padding='same'),
        keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same'),
        keras.layers.Conv2D(128, kernel_size=3, activation='relu', padding='same'),
        keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same'),
        keras.layers.Flatten(),
        keras.layers.Dense(512, activation=tf.nn.relu),
        keras.layers.Dropout(0.2),
        keras.layers.Dense(NUM_CLASS, activation=tf.nn.softmax)
    ])

    model.compile(optimizer=tf.train.AdamOptimizer(),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    model.summary()
    return model
```



```
def train(model, train_features, train_labels, val_features, val_labels):  
    checkpoint_path = "training_cnn/cp-{epoch:04d}.ckpt"  
    cp_callback = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,  
                                                    save_weights_only=True,  
                                                    verbose=1)  
  
    model.fit(train_features, train_labels, epochs=50,  
              validation_data = (val_features, val_labels),  
              callbacks=[cp_callback])
```

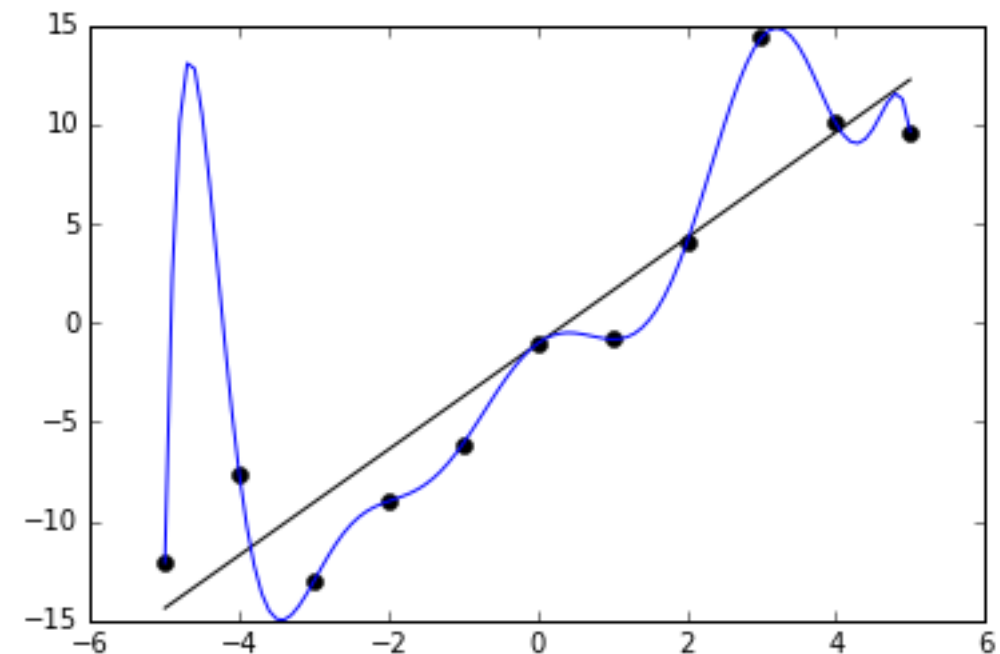
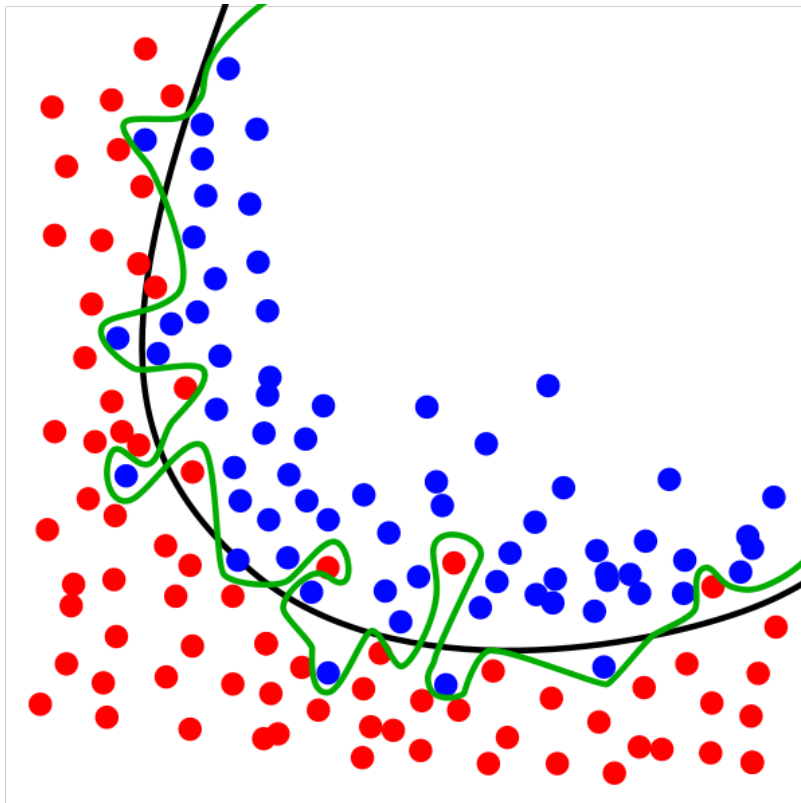
```
def train_from_scratch():  
    class_names = ['cat', 'cow', 'dog', 'pig', 'sheep']  
    train_features, train_labels, test_features, test_labels = load_all_data()  
    model = create_model()  
    train(model, train_features, train_labels, test_features, test_labels)  
  
    test_loss, test_acc = model.evaluate(test_features, test_labels)  
    print('Test accuracy:', test_acc)  
  
    predictions = model.predict(test_features)  
    print(predictions[0])  
    print(np.argmax(predictions[0]))  
    print(test_labels[0])
```

```
def load_weights_and_predict():
    model = create_model()
    model.load_weights('training_cnn/cp-0020.ckpt')

    my_test_img = load_image('test_image01.jpeg')
    my_test_img = np.reshape(my_test_img, [1, IMG_HEIGHT, IMG_WIDTH, 3])
    # my_test_img = (np.expand_dims(my_test_img, 0))
    print(my_test_img.shape)
    my_prediction = model.predict(my_test_img)
    print(my_prediction[0])
    print(np.argmax(my_prediction[0]))

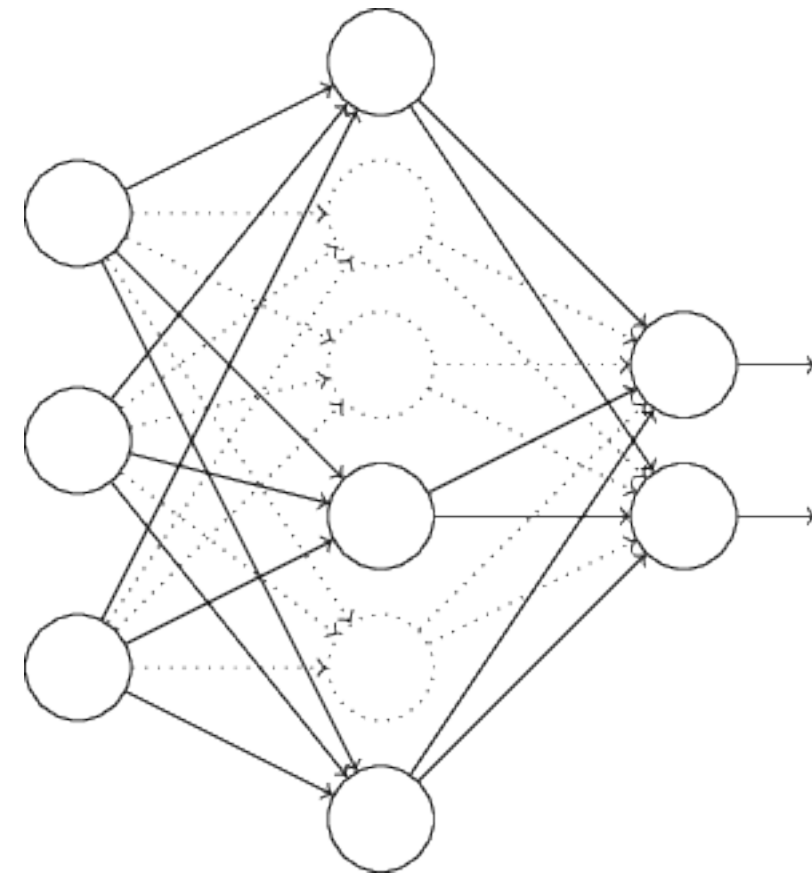
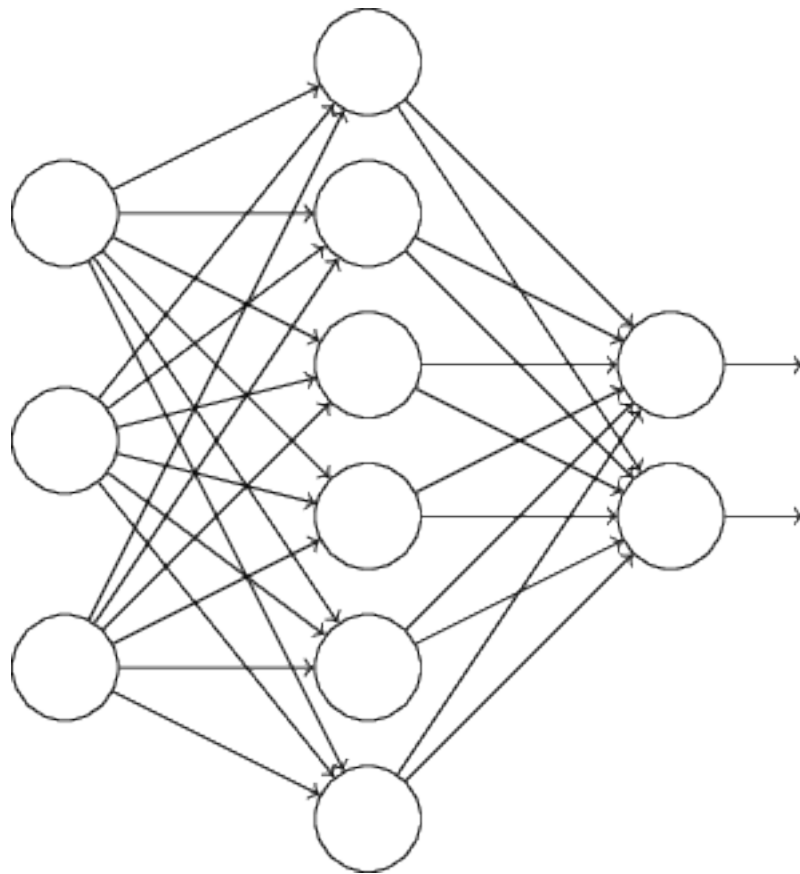
if __name__ == '__main__':
    train_from_scratch()
    # load_weights_and_predict()
```

- Overfitting은 “특정한 데이터 집합에 지나치게 정확하게 맞춰짐으로써 오히려 새로운 추가 데이터에 잘 맞지 않게 되고 따라서 예측력이 떨어지게 된 상태”를 의미한다.



- Dropout은 overfitting을 억제하기 위한 기법의 하나이다.

training batch마다 랜덤하게 일정한 비율의 노드를 네트워크에서 제거하고 training 한다.  
보통 fully connected layer에 적용한다.



마치 여러 개의 신경망을 동시에 training하고  
나중에 weight들을 평균하는 것과 유사한 효과를 얻는다.



dropout 확률이다. 0.5라고 가정하자.

```
keras.layers.Dense(512, activation=tf.nn.relu),  
keras.layers.Dropout(0.5),
```

▶ Dropout은 overfitting을 억제하기 위한 대표적인 기법들 중 하나이다.

