

CNN

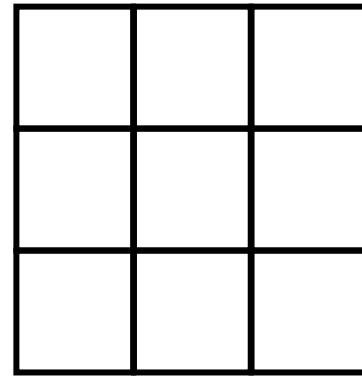
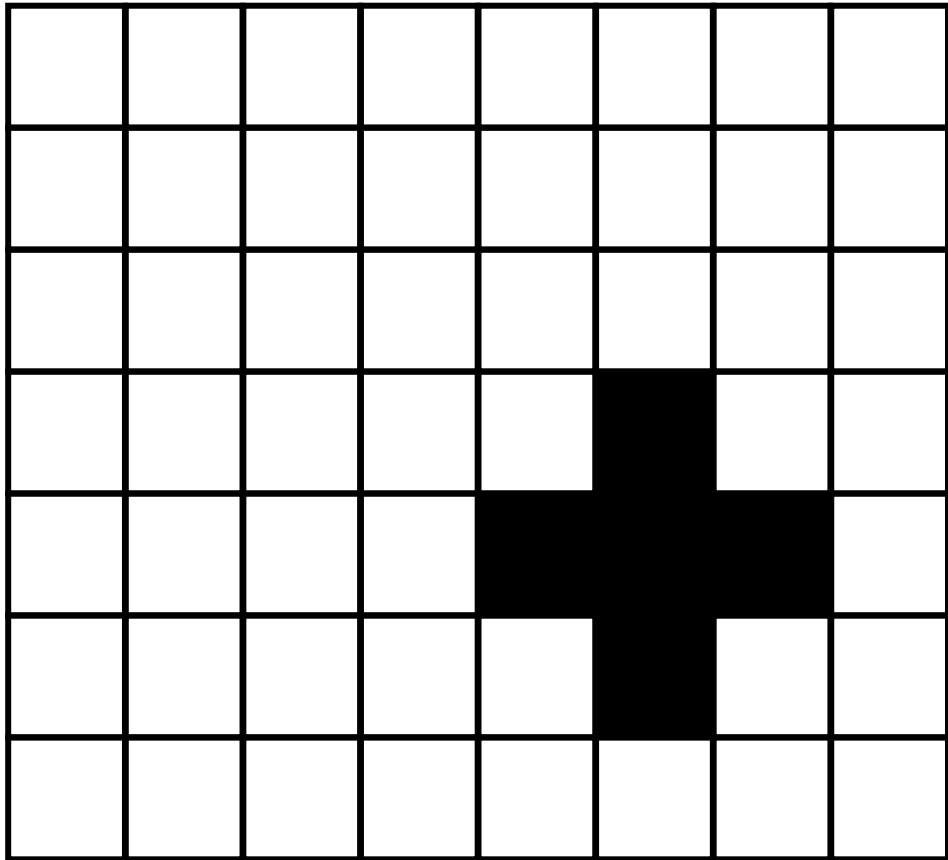
(Convolution Neural Network)

개요

- 영상처리 복습
- CNN의 동작원리
- CNN 프로그래밍 (Conv2D(), MaxPooling2D()...)

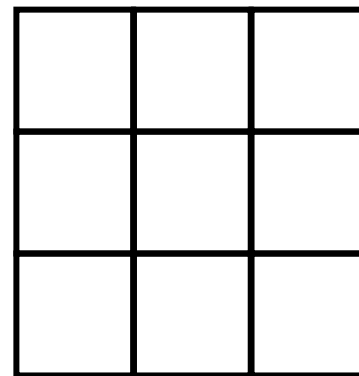
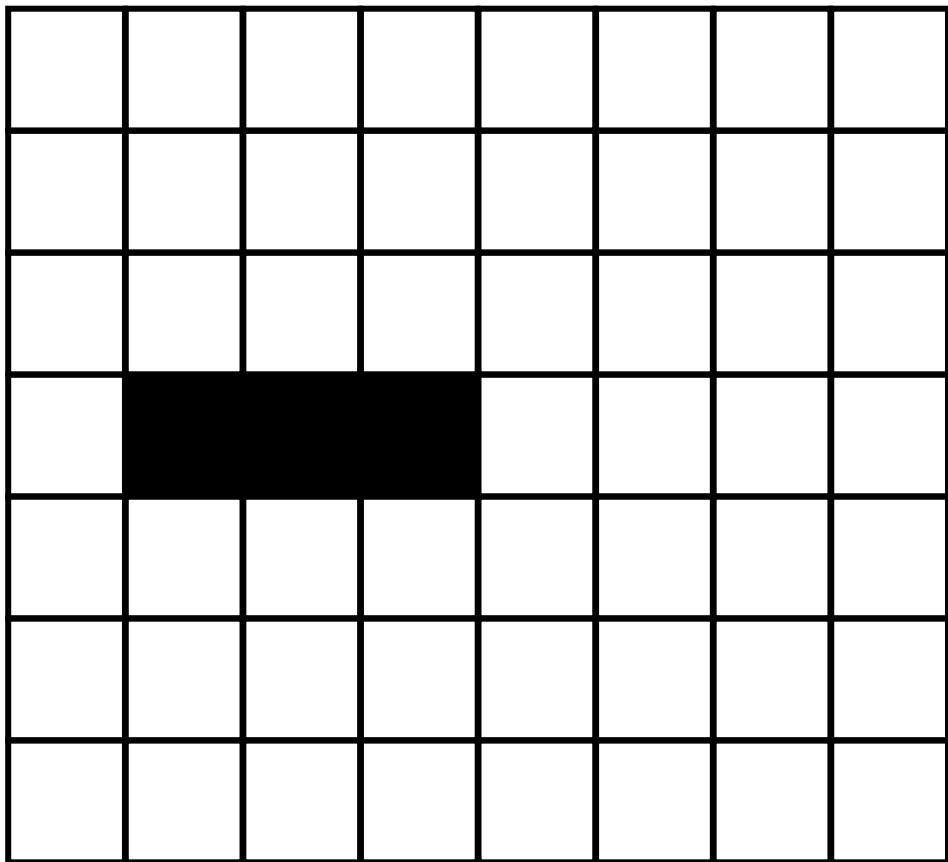
복습 : 필터(Filter)

- 사진에서 + 를 찾으려면 어떤 필터를 쓰면 될까?



복습 : 필터(Filter)

- 사진에서 --- 를 찾으려면 어떤 필터를 쓰면 될까?



Convolution 연산

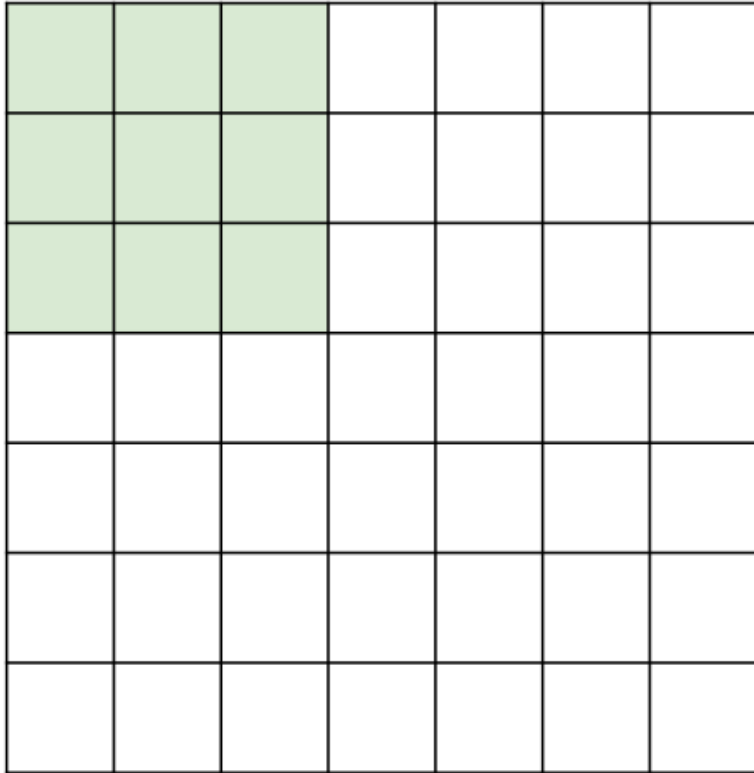
입력 크기와 출력 크기 관점에서

축구공 찾기



세부 개념도

7

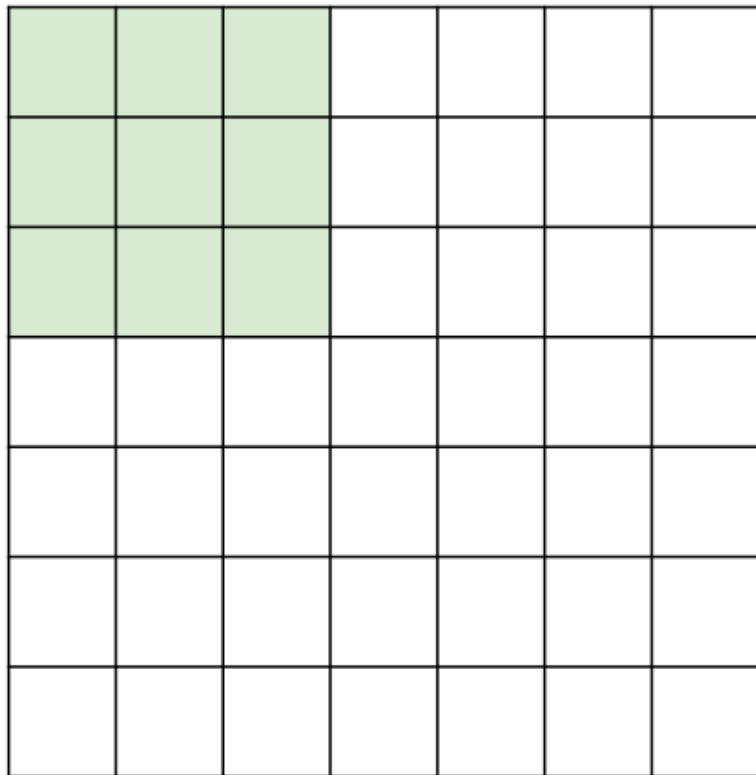


7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

세부 개념도

7



7

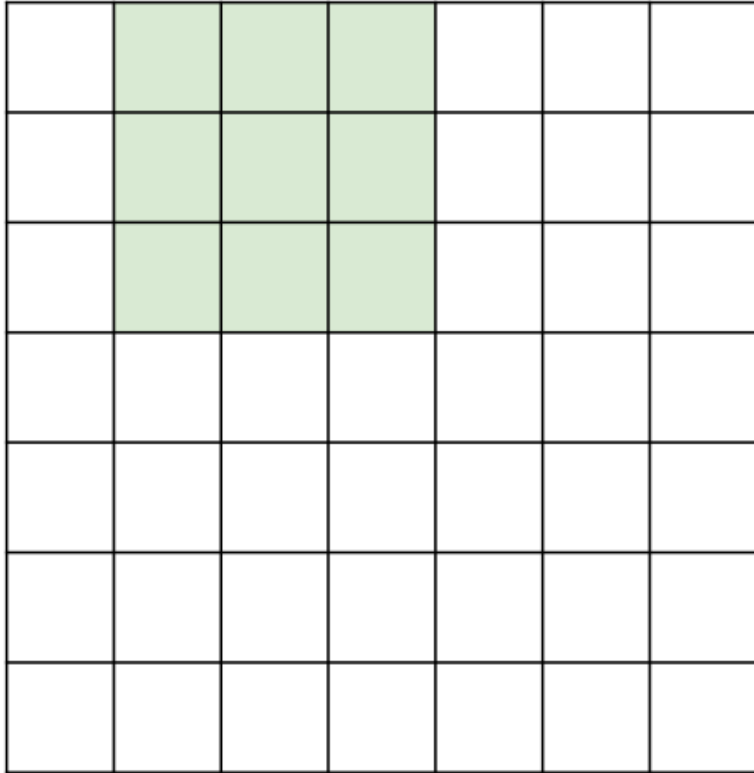
7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

$$\begin{aligned} &w_{11} * p_{x_{i-1}, x_{j-1}} + w_{12} * p_{x_i, x_{j-1}} + w_{13} * p_{x_{i+1}, x_{j-1}} + \\ &w_{21} * p_{x_{i-1}, x_j} + w_{22} * p_{x_i, x_j} + w_{23} * p_{x_{i+1}, x_j} + \\ &w_{31} * p_{x_{i-1}, x_{j+1}} + w_{32} * p_{x_i, x_{j+1}} + w_{33} * p_{x_{i+1}, x_{j+1}} \end{aligned}$$

= 새로운 특징점 하나 생성

세부 개념도

7

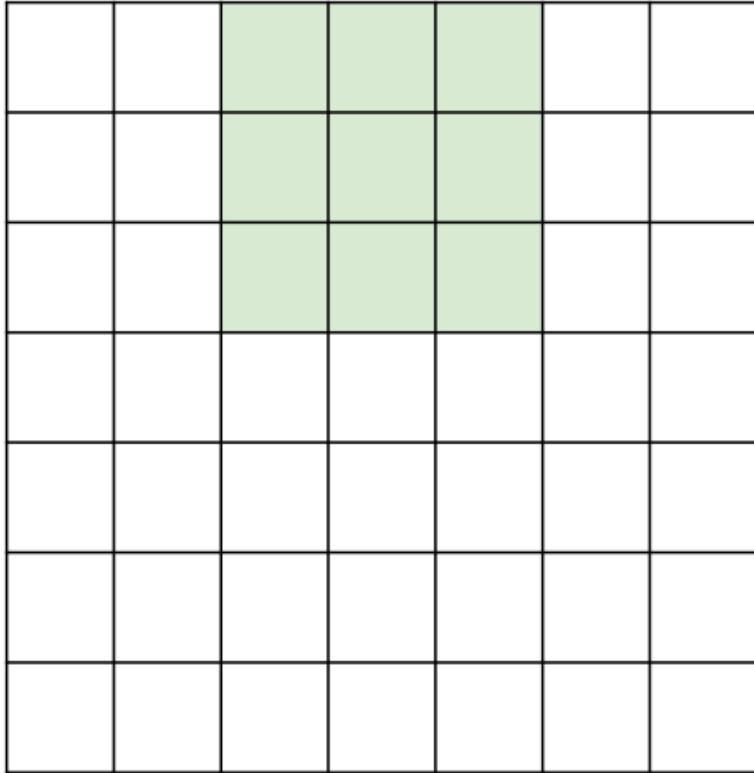


7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

세부 개념도

7

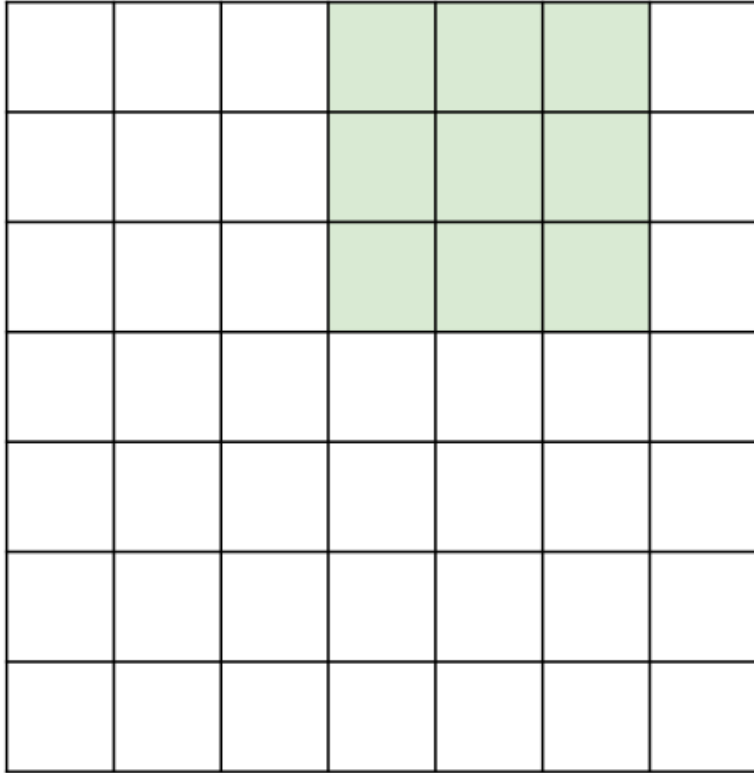


7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

세부 개념도

7



7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

세부 개념도

7

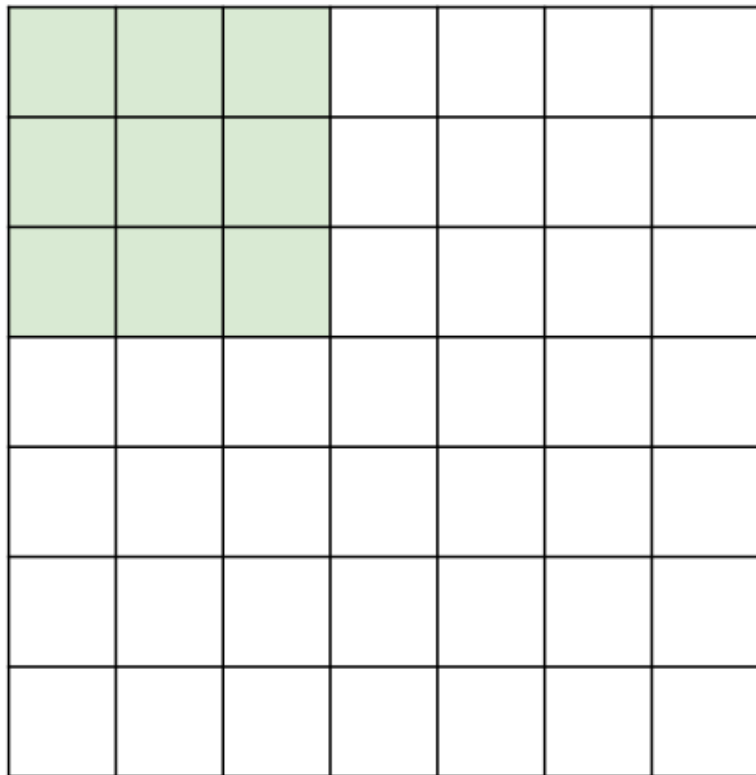
7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

=> 5 x 5의 출력결과

세부 개념도

7



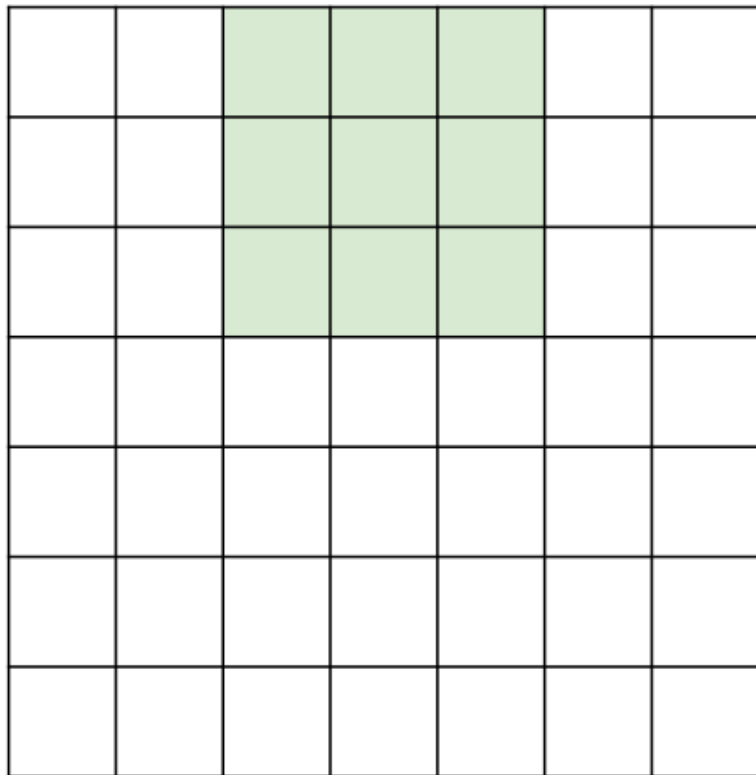
7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

두 칸씩 이동하면 stride 2

세부 개념도

7



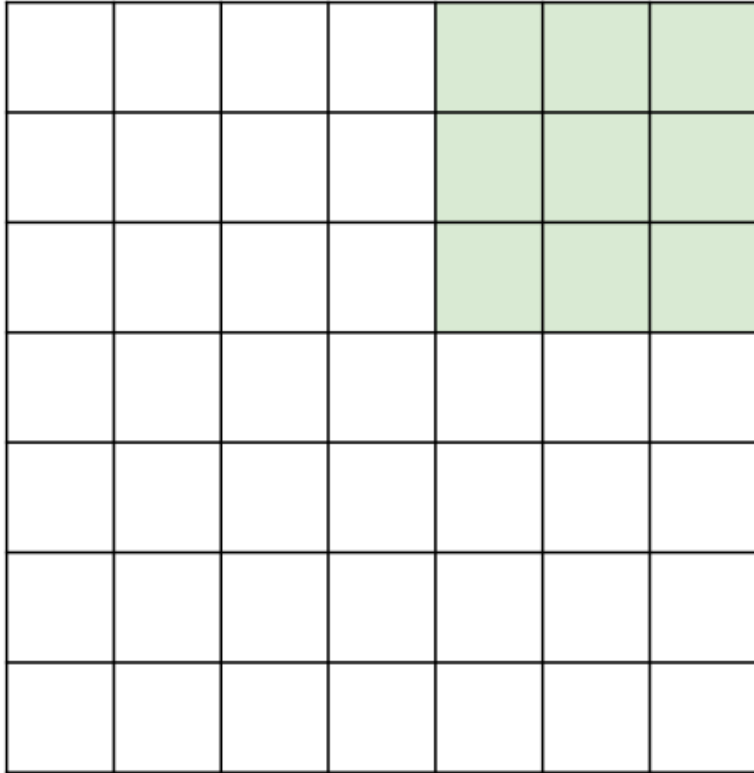
7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

두 칸씩 이동하면 stride 2

세부 개념도

7



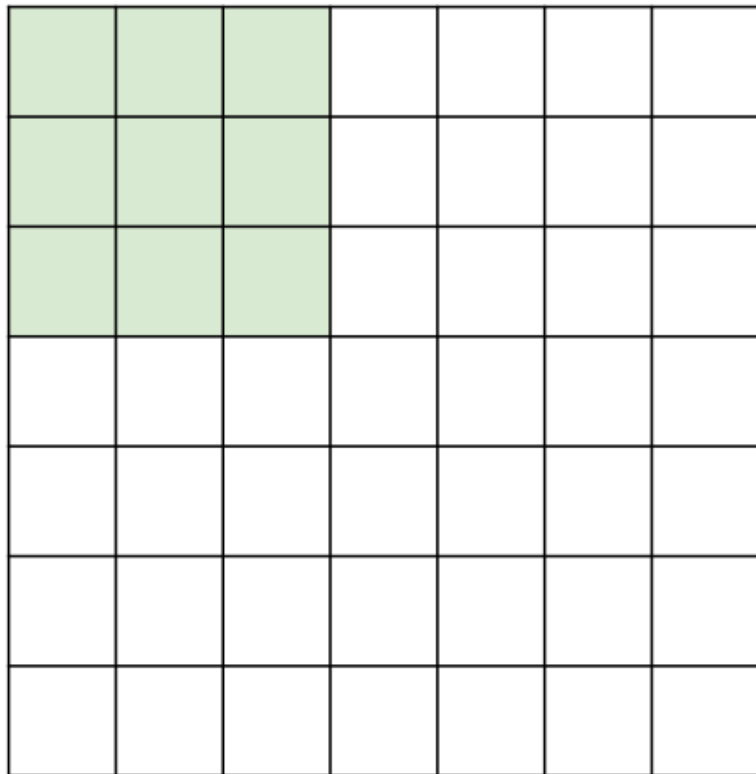
7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

두 칸씩 이동하면 stride 2

세부 개념도

7



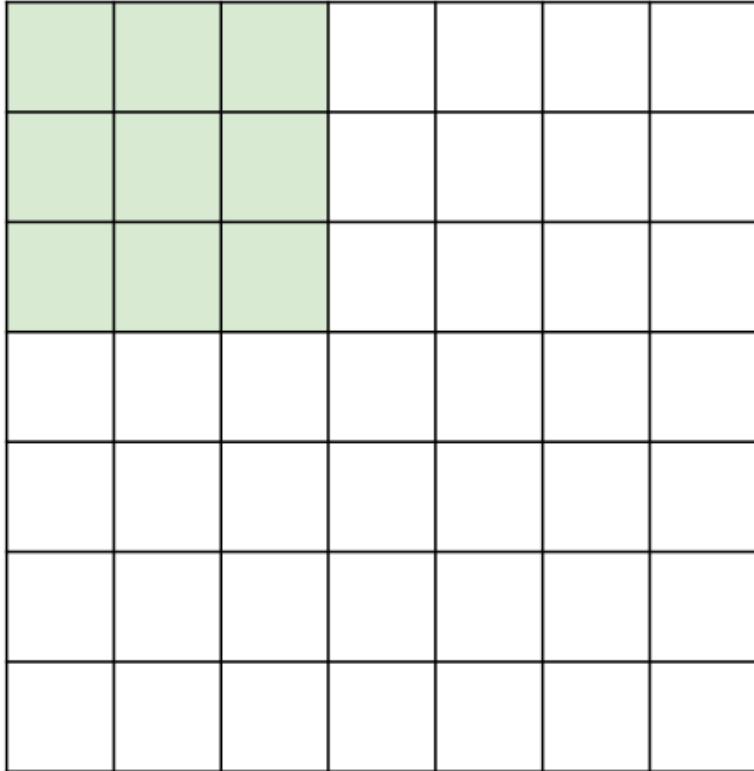
7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

세 칸씩 이동하면 stride 3

세부 개념도

7



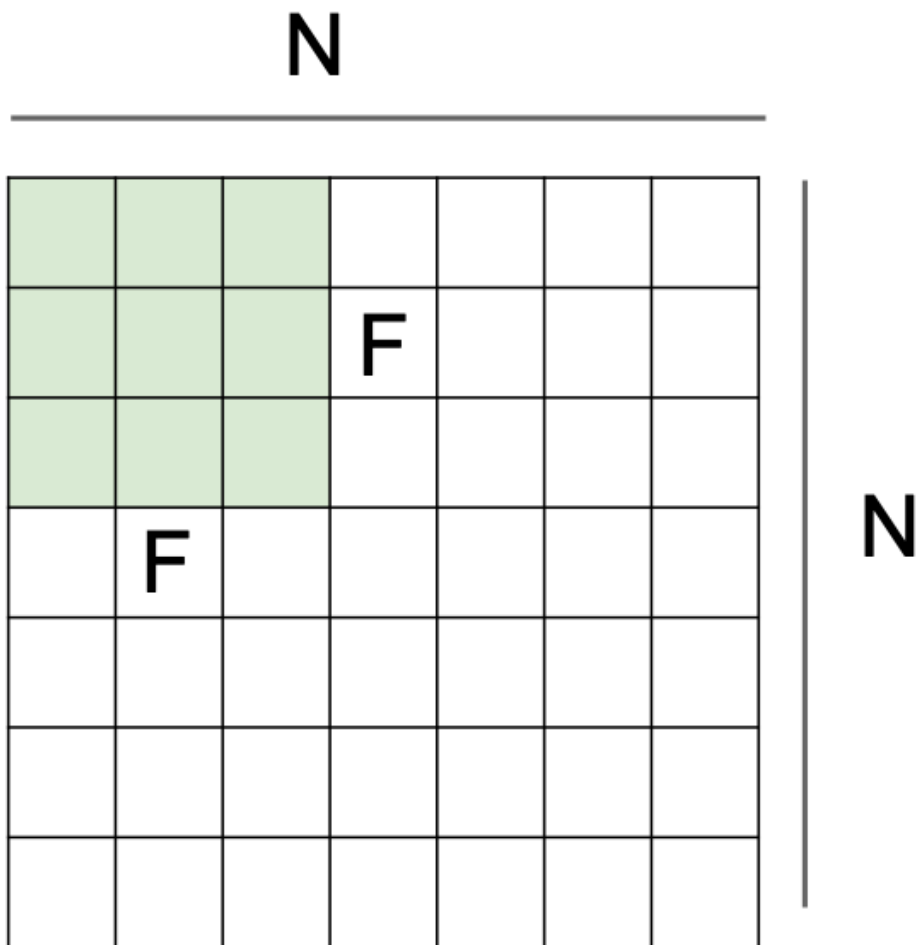
7

7 x 7로 이루어진 이미지와
3 x 3 필터가 주어진 경우

세 칸씩 이동하면 stride 3

7x7 이미지
3x3 필터에서는
3칸 이동은 불가

세부 개념도



일반식으로 나타내면

Output size:

$$(N - F) / \text{stride} + 1$$

e.g. $N = 7, F = 3$:

$$\text{stride } 1 \Rightarrow (7 - 3) / 1 + 1 = 5$$

$$\text{stride } 2 \Rightarrow (7 - 3) / 2 + 1 = 3$$

$$\text{stride } 3 \Rightarrow (7 - 3) / 3 + 1 = 2.33 : \backslash$$

동일 크기 만들기 : Zero Padding을 경계선에 적용

0	0	0	0	0	0			
0								
0								
0								

7 x 7로 이루어진 이미지에
Zero padding 크기 1 적용

3 x 3 필터가 주어진 경우

한 칸씩 이동하면 (stride 1)

(recall:)

$(N - F) / \text{stride} + 1$

동일 크기 만들기 : Zero Padding을 경계선에 적용

0	0	0	0	0	0			
0								
0								
0								
0								

7 x 7로 이루어진 이미지에
Zero padding 크기 1 적용

3 x 3 필터가 주어진 경우

한 칸씩 이동하면 (stride 1)

최종적으로 7 x 7

동일 크기 만들기 : Zero Padding을 경계선에 적용

0	0	0	0	0	0			
0								
0								
0								
0								

필터가 $F \times F$ 인 경우, padding 크기 $(F - 1)/2$ 설정 시,
입력과 동일한 크기의 특징 생성

$F = 3$ 이면 padding 크기 1

$F = 5$ 이면 padding 크기 2

$F = 7$ 이면 padding 크기 3

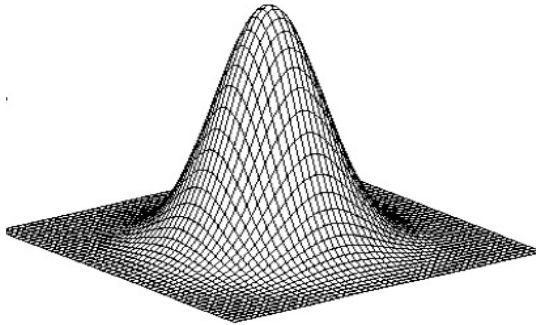
Convolutional Neural Networks

Convolution Neural Network

- 이미지에서 특징을 찾기 위한 기존의 방법 중 하나 (검증된 방법)
- 학습을 통해 데이터에서 목적에 맞는 필터(Filter)를 찾아냄
 - 기존에는 필터를 연구자들이 수학적으로 찾아냄
- 특징1 : 물체가 이미지에 어디에 있던지 탐지 가능
- 특징2 : Dense대비 파라미터의 수가 아주 적음
- 특징3 : 해당 데이터셋에만 특화된 특징을 추출 (장점? 단점?)

Convolution Neural Network (CNN)

연구자들이 고안한 필터

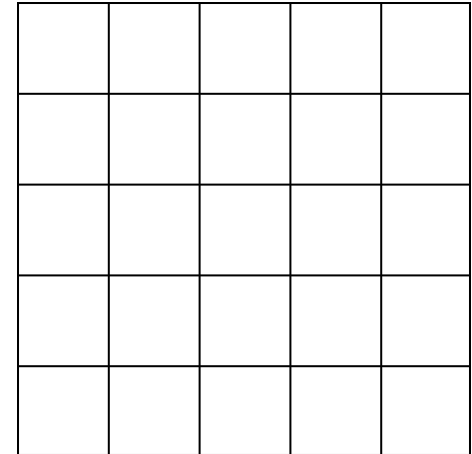


Gaussian

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

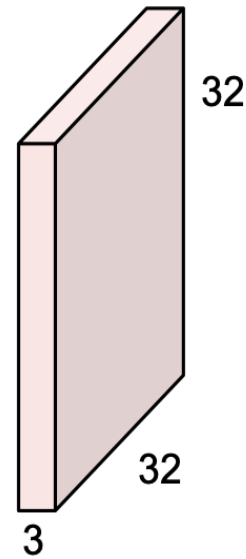
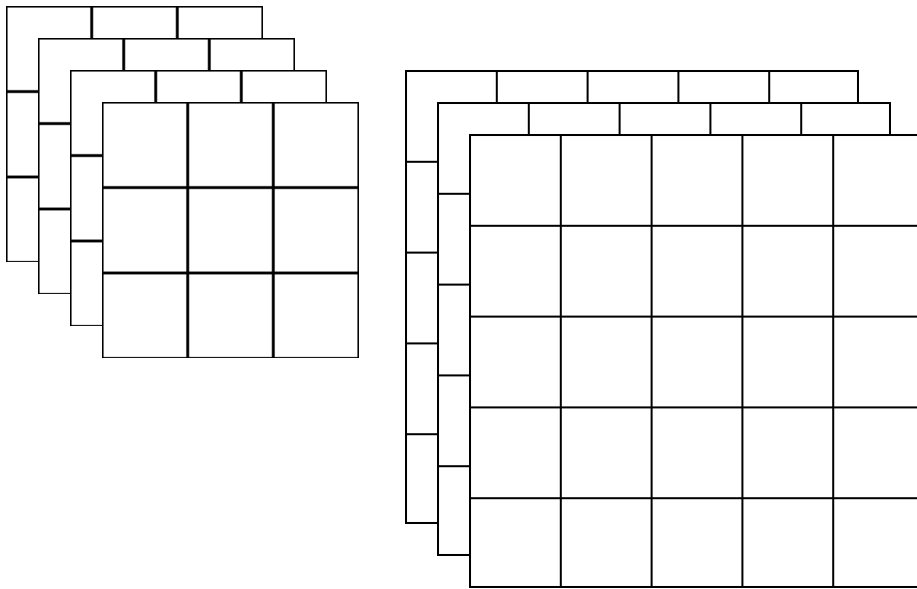
필터를 데이터에서 학습 ?



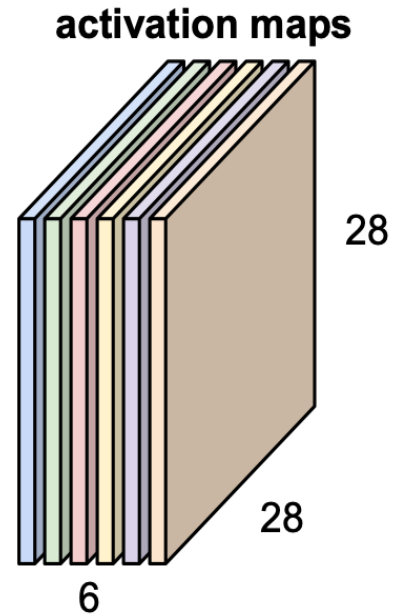
Convolution Neural Network (CNN)

데이터셋에 특화된 학습 필터
다양한 필터로 구성

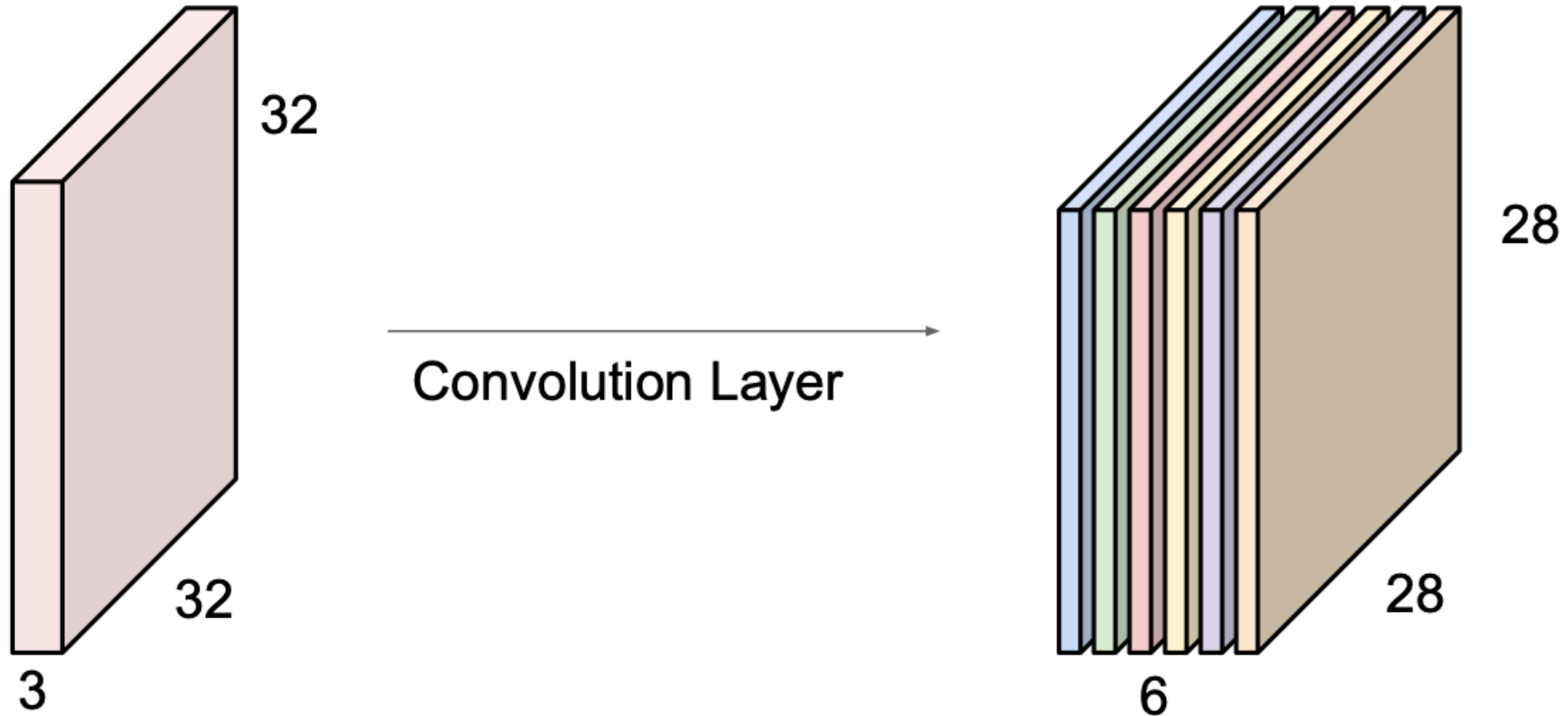
이미지를 다양한 특징 벡터로 변경



Convolution Layer



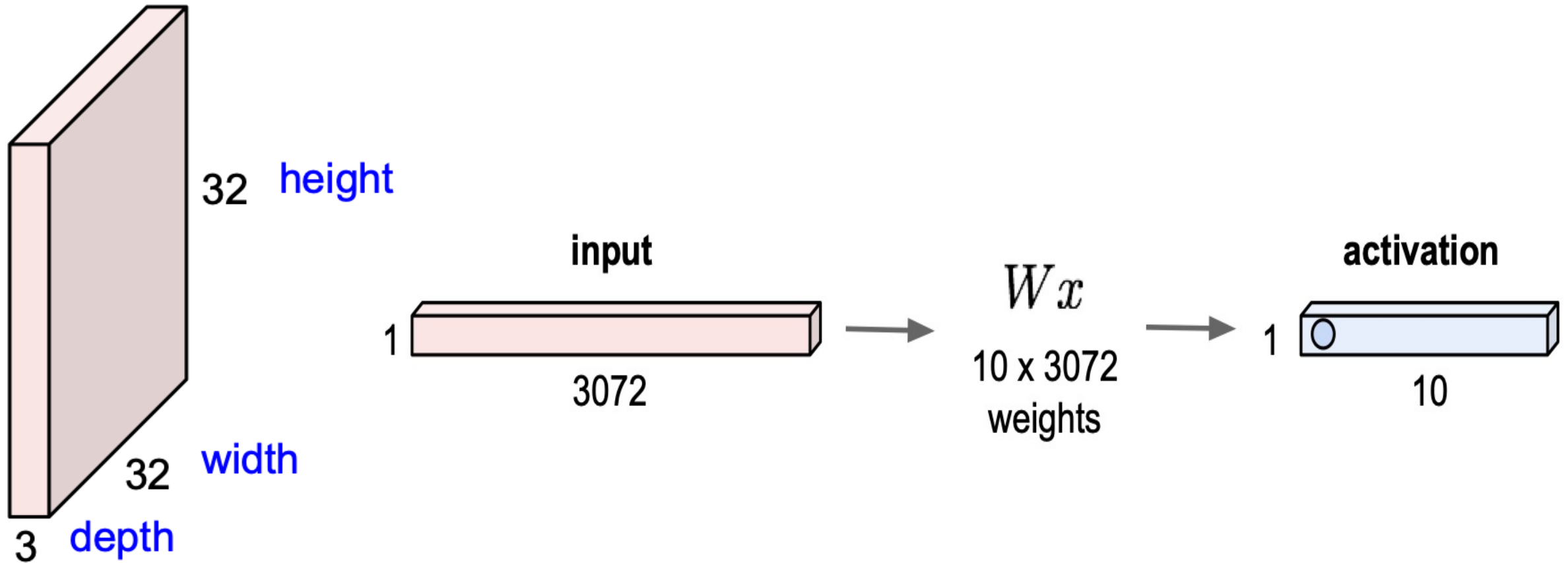
예를 들어, 6개의 5×5 (가로 5 * 세로 5)의 필터가 있다고 하면,
각 필터에 해당하는 6개의 서로다른 특징 추출
activation maps



이 6개의 추출 특징들을 하나의 새로운 이미지 생성 (크기 $28 * 28 * 6$)

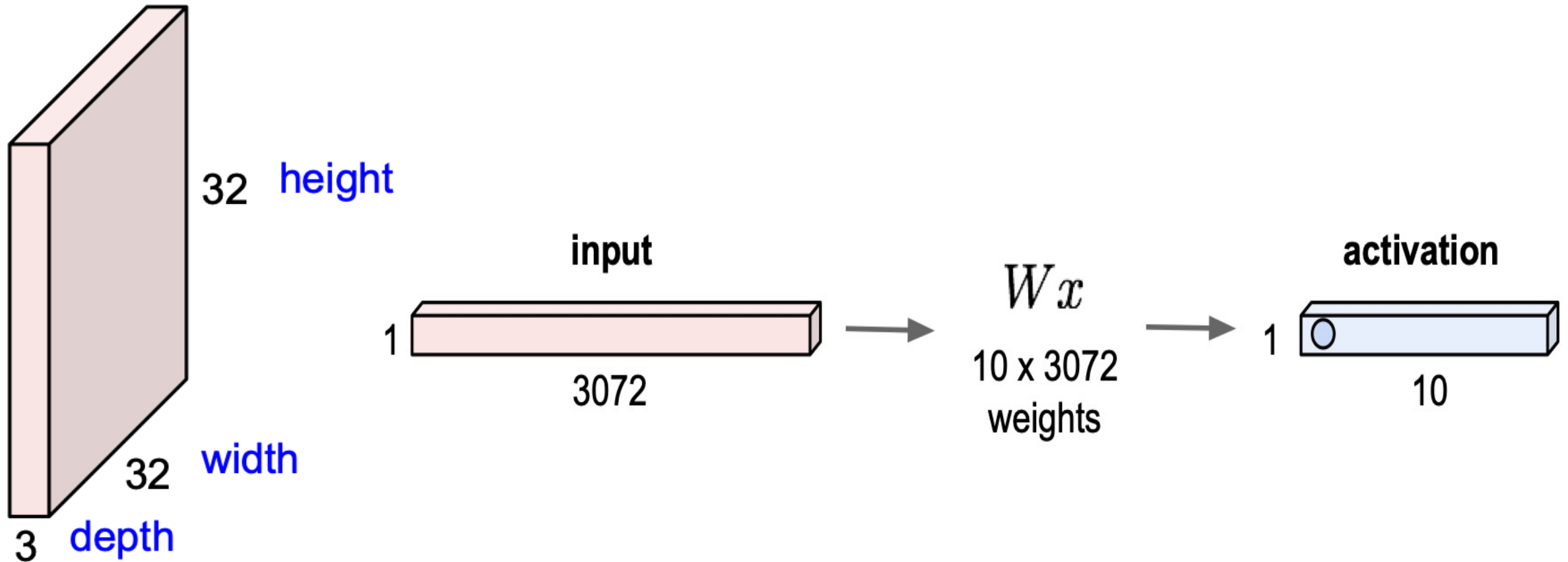
Dense Layer

32x32x3 image -> stretch to 3072 x 1



Dense Layer

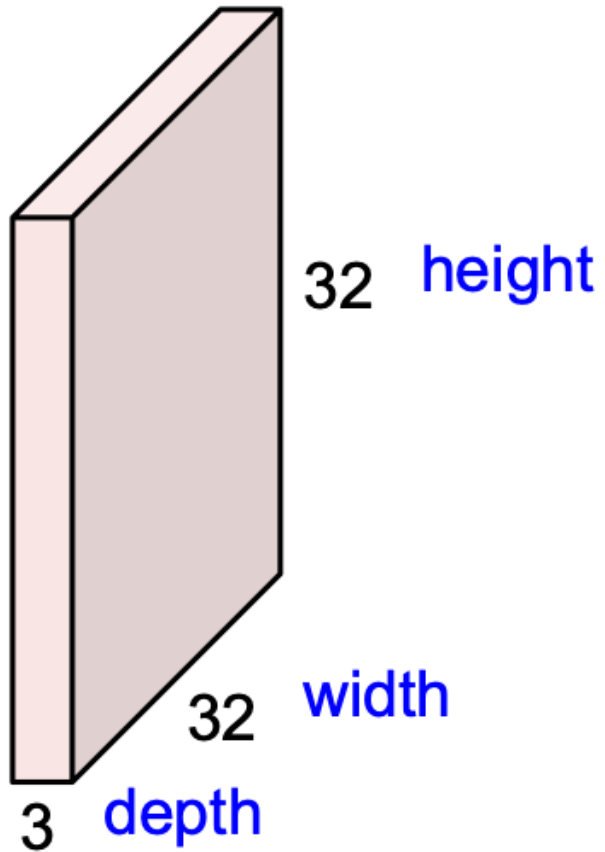
32x32x3 image -> stretch to 3072 x 1



하나의 점 생성시 3072개의 입력 모두 이용

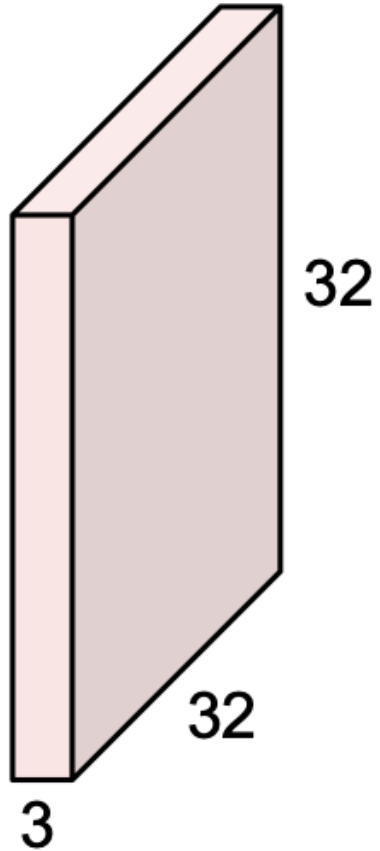
Convolution Layer

32x32x3 image -> 공간 구조 정보가 유지됨



Convolution Layer

32x32x3 image

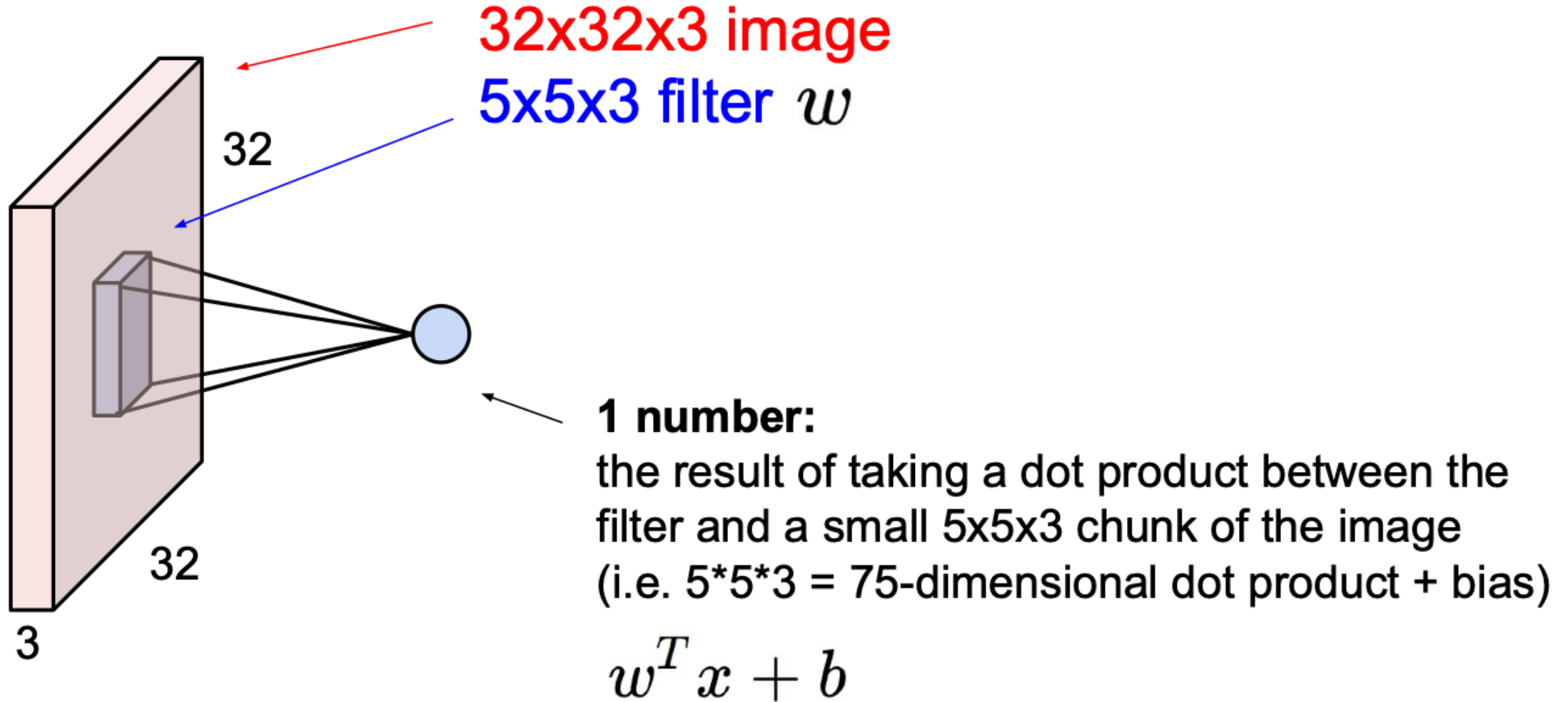


5x5x3 filter



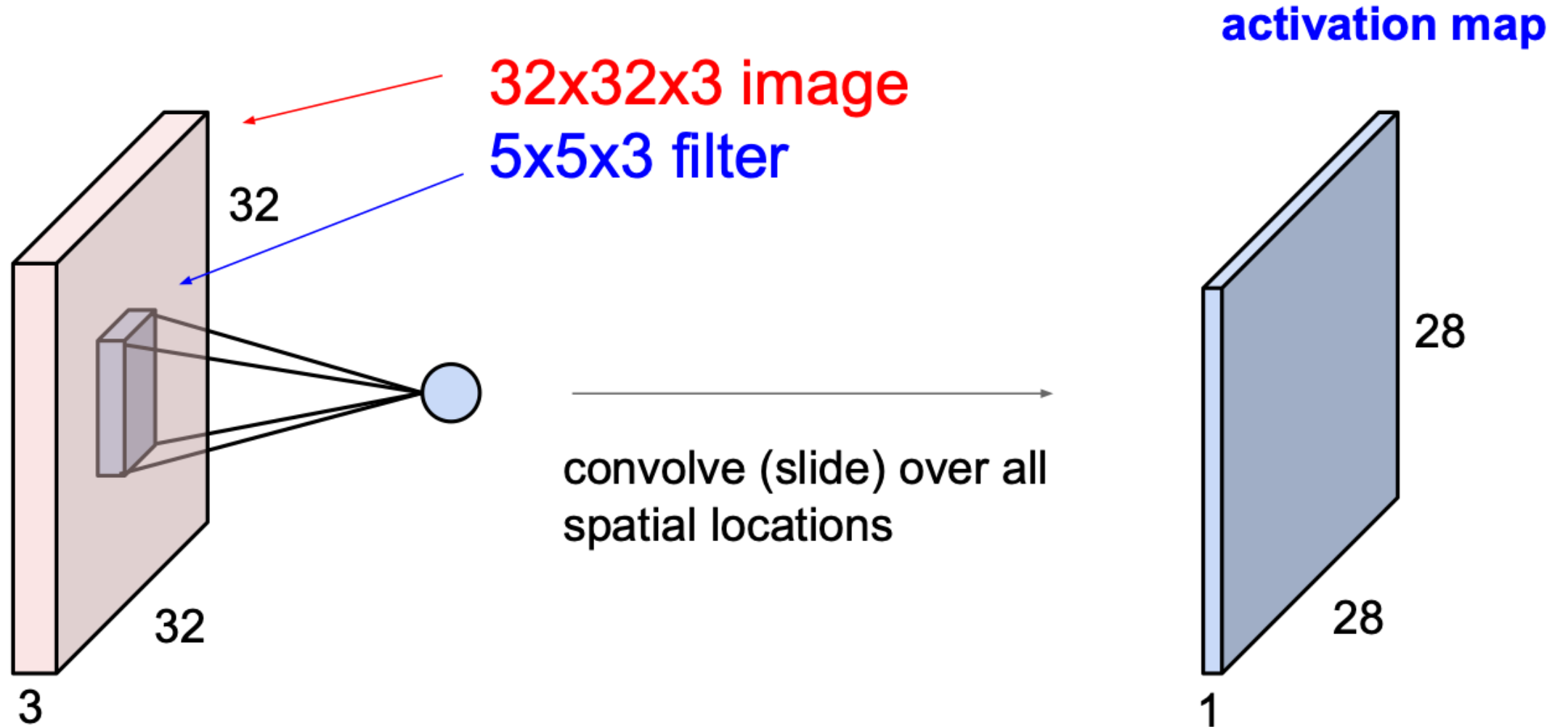
7 x 7로 이루어진 이미지와
Convolve란 필터가 이미지 위를
공간적으로 슬라이딩 하면서
내적(dot product)를 수행

Convolution Layer



5x5x3의 작은 청크이미지마다 하나의 점 생성

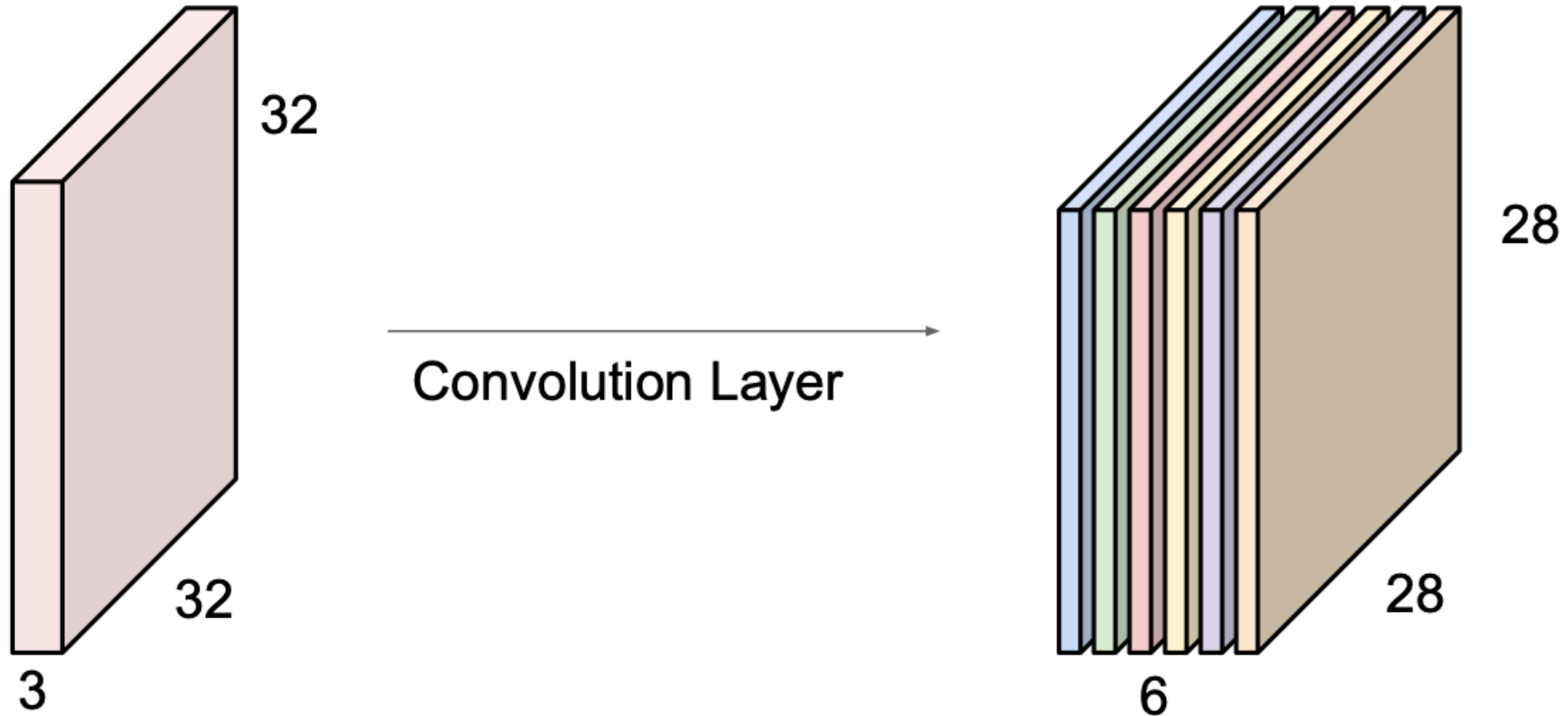
Convolution Layer



Convolution Layer

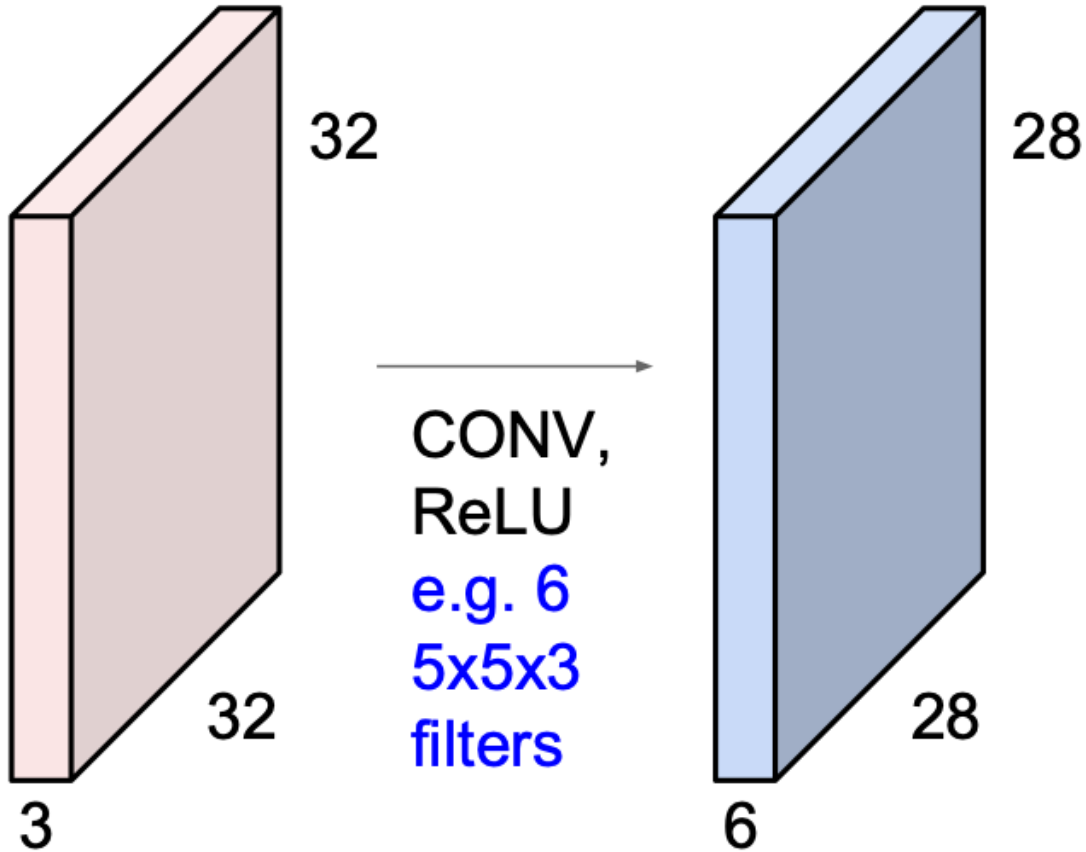


예를 들어, 6개의 5×5 (가로 5 * 세로 5)의 필터가 있다고 하면,
각 필터에 해당하는 6개의 서로다른 특징 추출
activation maps

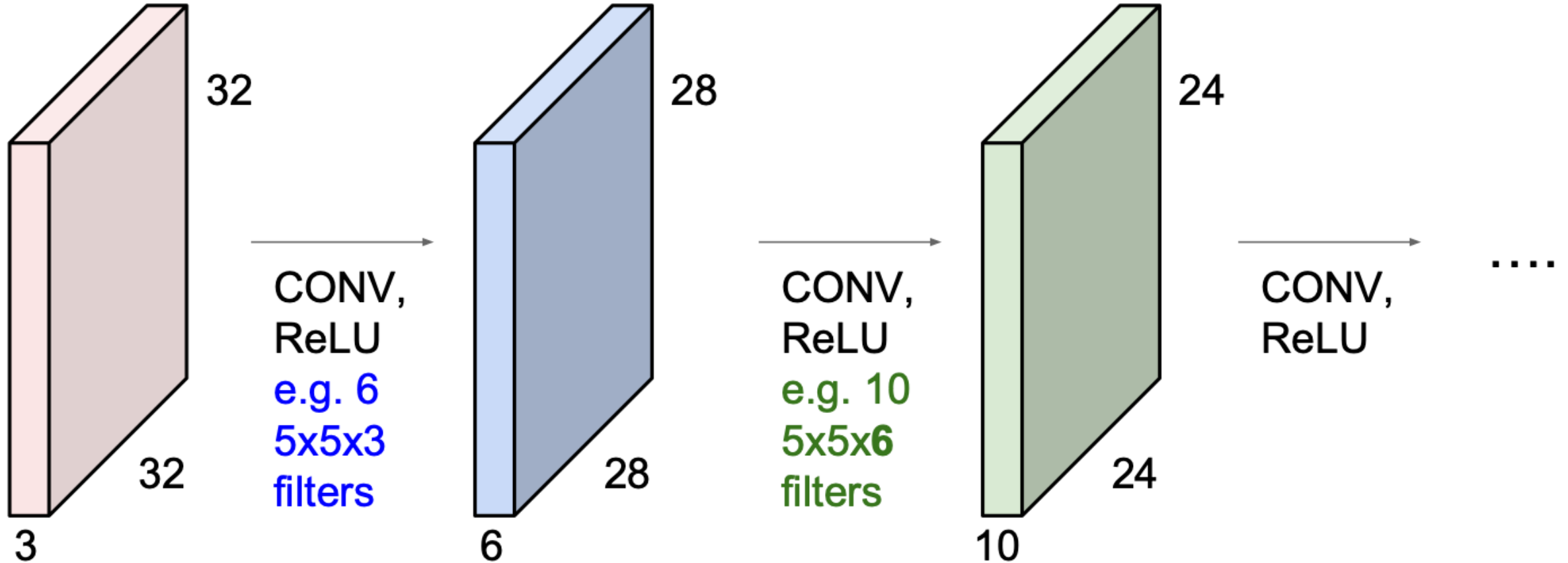


이 6개의 추출 특징들을 하나의 새로운 이미지 생성 (크기 $28 * 28 * 6$)

ConvNet : Convolution Layer의 시퀀스 (+ Activation functions)



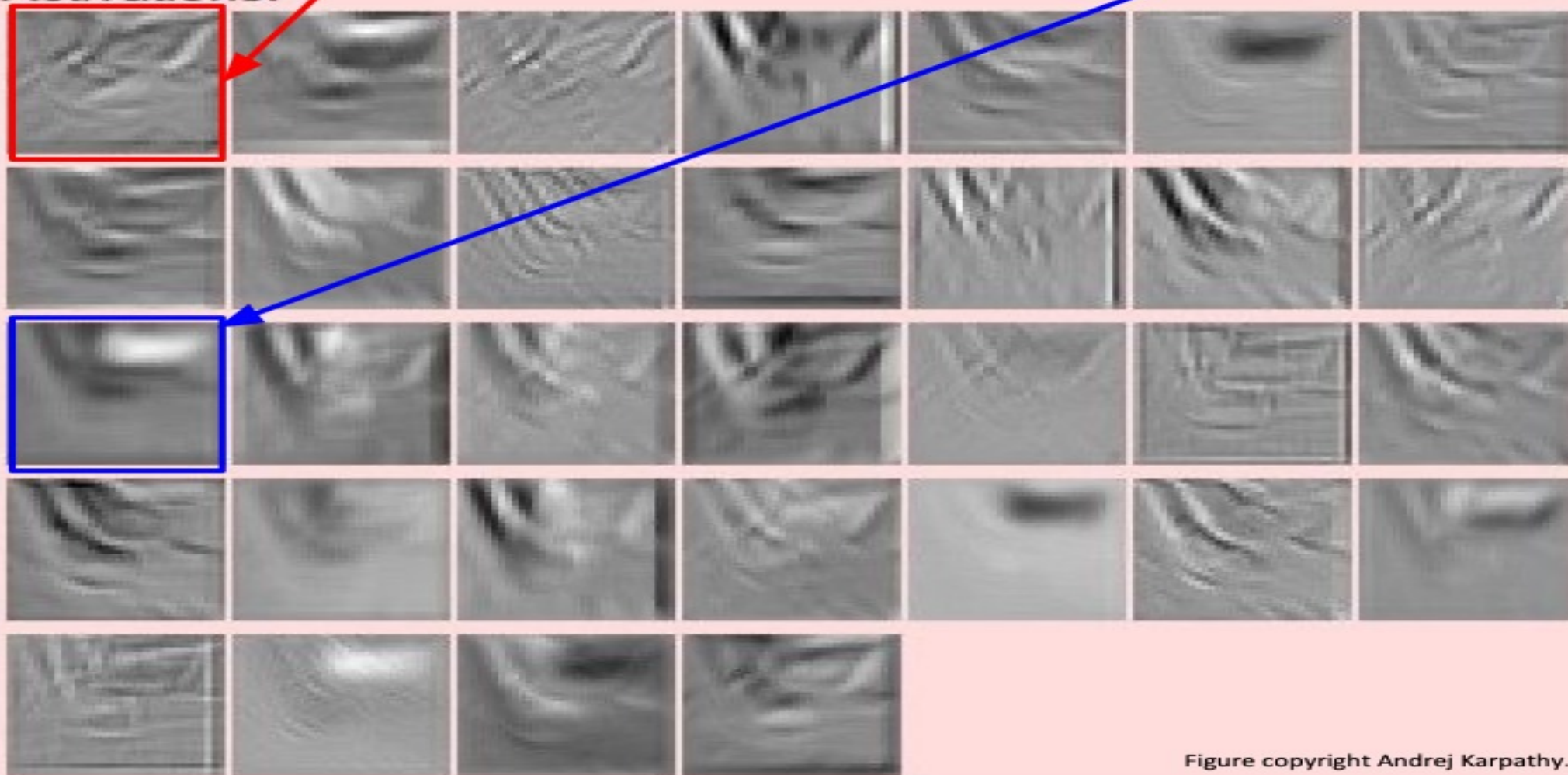
ConvNet : Convolution Layer의 시퀀스 (+ Activation functions)





one filter =>
one activation map

Activations:



$f[l]$

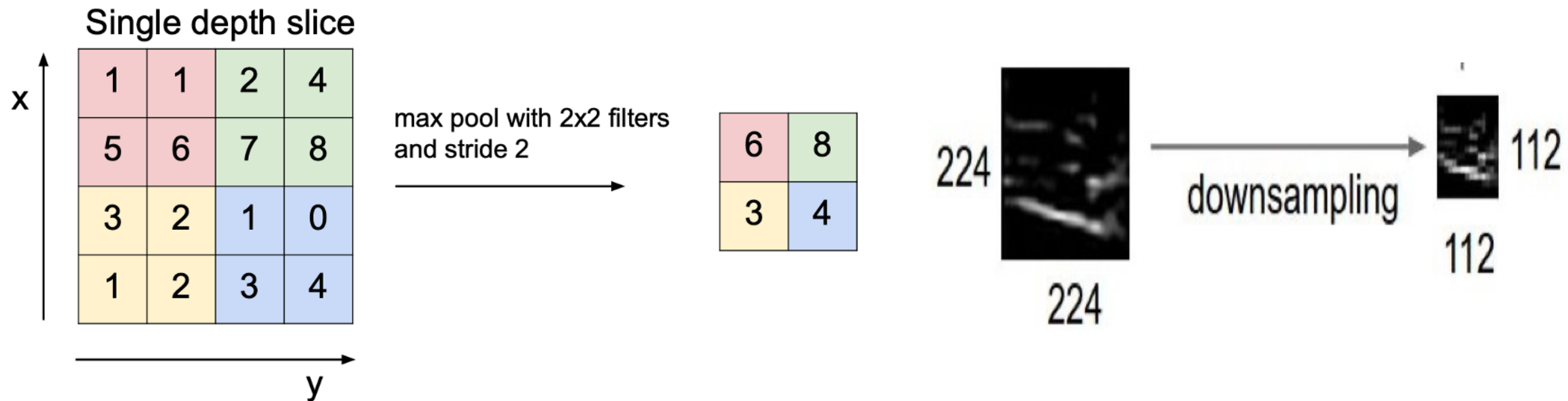
CNN 프로그래밍

API 설명 : Conv2D

- Conv2D(filters, kernel_size, strides, padding, ...)
 - - filters : 필터 개수. 즉, 추출하고자 하는 서로 다른 특징의 개수
 - - kernel_size : 필터 크기 [예: (3,3)]
 - - strides : Convolution 동작시 건너뛰는 크기 [예 : (2,2)]
 - - padding : 데이터 주면에 0을 padding 할지 여부 [예: 'same', 'valid']
- 입력 (None, W, H, C) -> 출력 (None, W', H', filters)

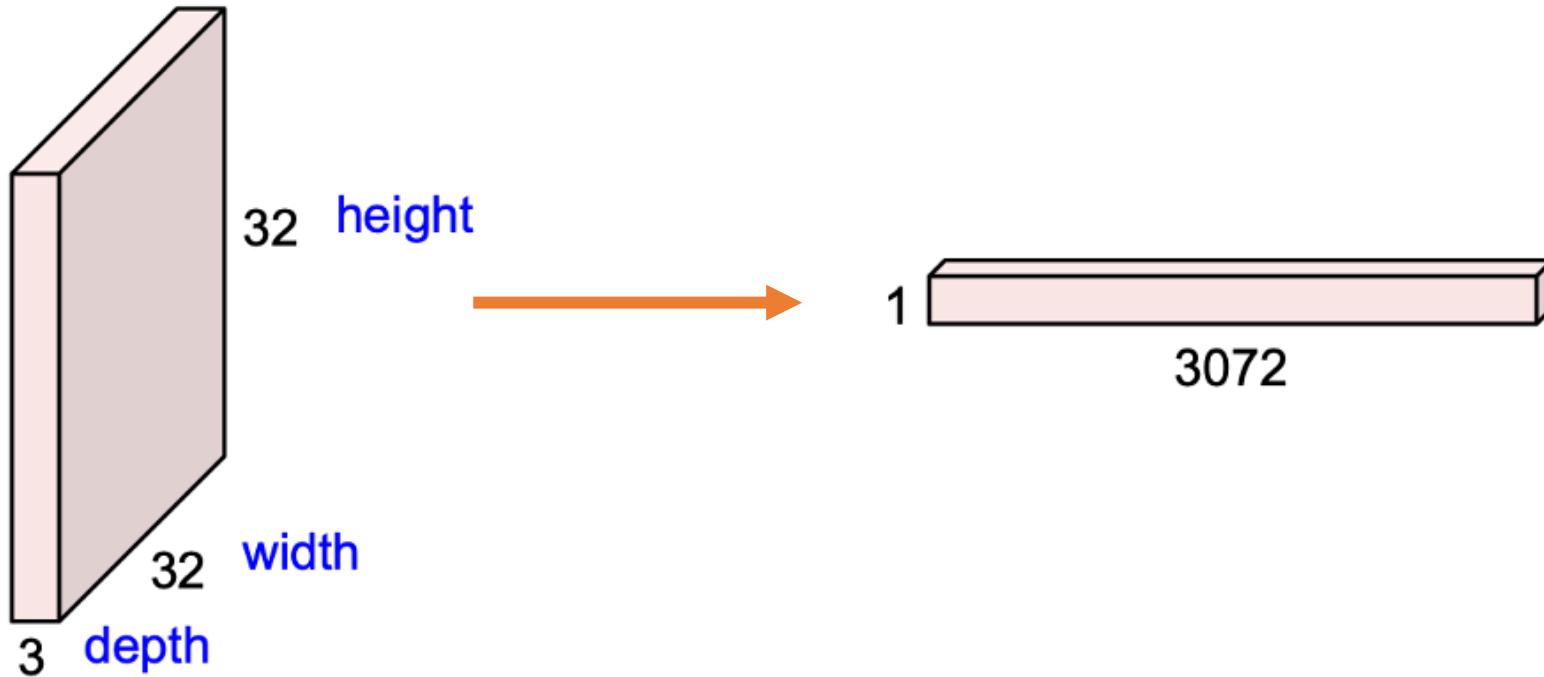
API 설명 : MaxPooling2D

- `MaxPooling2D(pool_size, strides)`
 - `pool_size` : pooling할 대상의 크기 (예: (2,2))
 - `strides` : 동작시 건너뛰는 크기 [예 : (2,2)]



API 설명 : Flatten()

- 추가 파라미터 없이 단독으로 사용
- 다차원의 데이터를 1차원 데이터로 변환



예제 : MNIST with CNN



참고 : <https://github.com/rickiepark/deep-learning-with-python-notebooks/blob/master/5.1-introduction-to-convnets.ipynb>

예제 : MNIST

```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

예제 : MNIST

```
model = models.Sequential()  
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))  
model.add(layers.Conv2D(32, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Flatten())  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.Dense(10, activation='softmax'))
```

model.summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 24, 24, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 12, 12, 32)	0
conv2d_2 (Conv2D)	(None, 10, 10, 64)	18496
conv2d_3 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 64)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65600
dense_1 (Dense)	(None, 10)	650

Total params: 131,242

Trainable params: 131,242

Non-trainable params: 0

예제 : MNIST

```
model.compile(optimizer='rmsprop',  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])  
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```



Terminal output showing training progress for 5 epochs on MNIST data. The output includes epoch numbers, progress bars, time per step, loss, and accuracy.

```
Epoch 1/5  
938/938 [=====] - 5s 6ms/step - loss: 0.1428 - accuracy: 0.9557  
Epoch 2/5  
938/938 [=====] - 5s 6ms/step - loss: 0.0409 - accuracy: 0.9875  
Epoch 3/5  
938/938 [=====] - 5s 6ms/step - loss: 0.0285 - accuracy: 0.9914  
Epoch 4/5  
938/938 [=====] - 5s 5ms/step - loss: 0.0219 - accuracy: 0.9934  
Epoch 5/5  
938/938 [=====] - 5s 5ms/step - loss: 0.0178 - accuracy: 0.9950  
<tensorflow.python.keras.callbacks.History at 0x7f55da0746a0>
```

Dense VS CNN

- Dense의 경우 파라미터 개수가 CNN대비 엄청 큼

```
network = models.Sequential()  
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))  
network.add(layers.Dense(10, activation='softmax'))  
network.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
=====		
dense_4 (Dense)	(None, 512)	401920
=====		
dense_5 (Dense)	(None, 10)	5130
=====		
Total params: 407,050		
Trainable params: 407,050		
Non-trainable params: 0		

CNN의 경우

Total params: 131,242
Trainable params: 131,242
Non-trainable params: 0