

23 하계 학부연구생 프로그램

ch08. 합성곱 신경망

12223547 박혜민

합성곱 신경망으로 패션 아이템 분류하기

01

CNN 기본 구조

- filter
- padding
- stride
- pooling

02

실습

- 과정
- 시각화

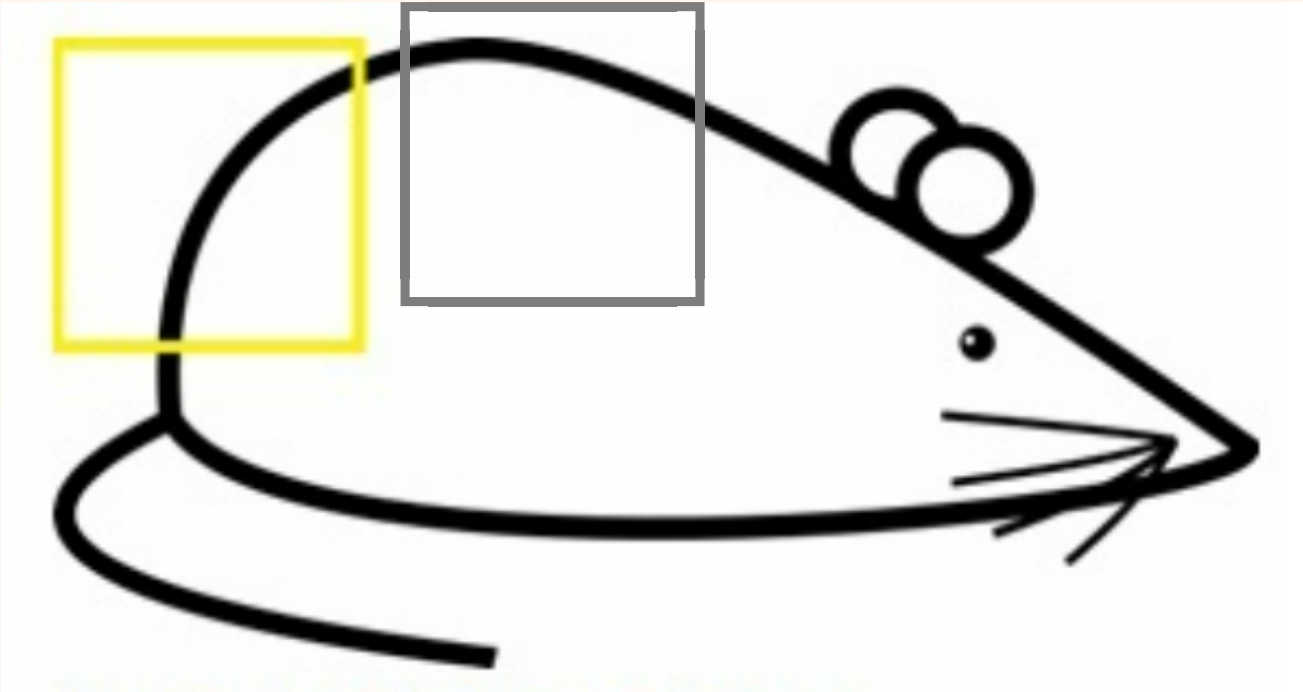
Convolution Neural Network(CNN)



윤곽선(특징) 추출



filter



conv filter의 edge 부분과 일치하는 영역에서
높은 값 도출

<왼쪽 위로 볼록한 곡선을 특징으로 추출하는 filter>

		30
	30	40
30	20	
30		

0	0	30
0	30	0
30	0	0
30	0	0

30		
20	30	
		30
		40

곱과 합의 연산

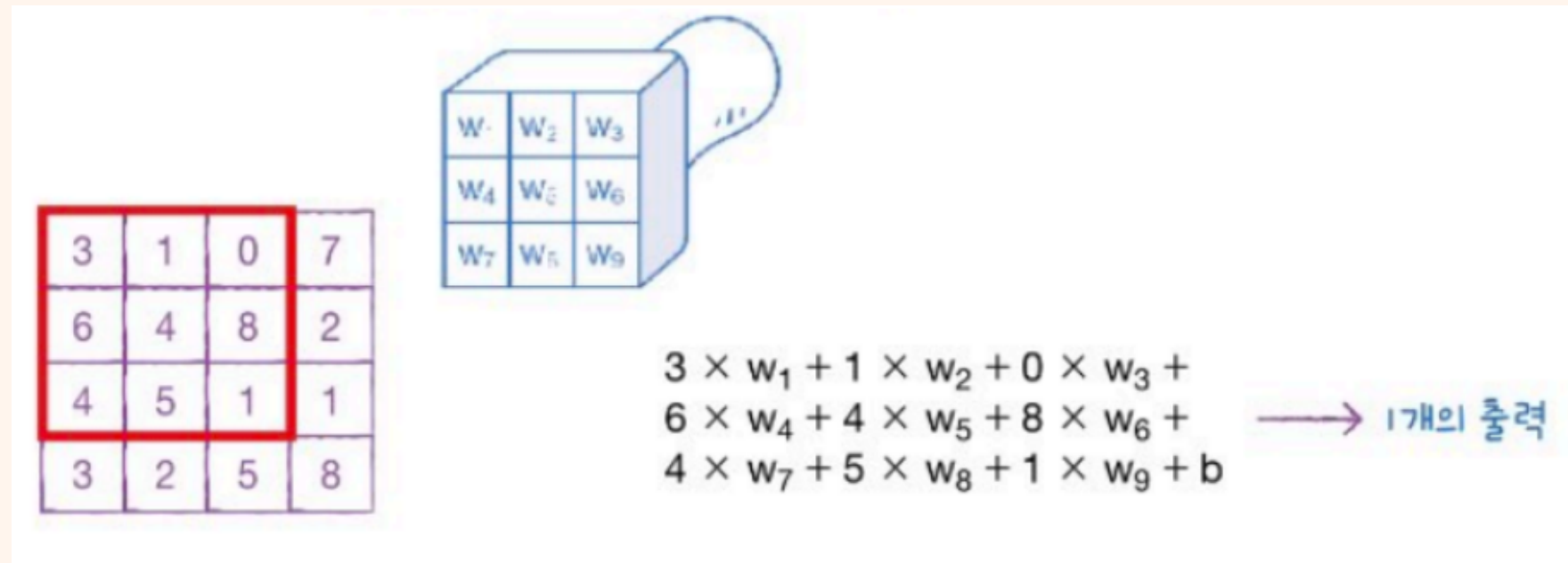
A



B

filter

window sliding

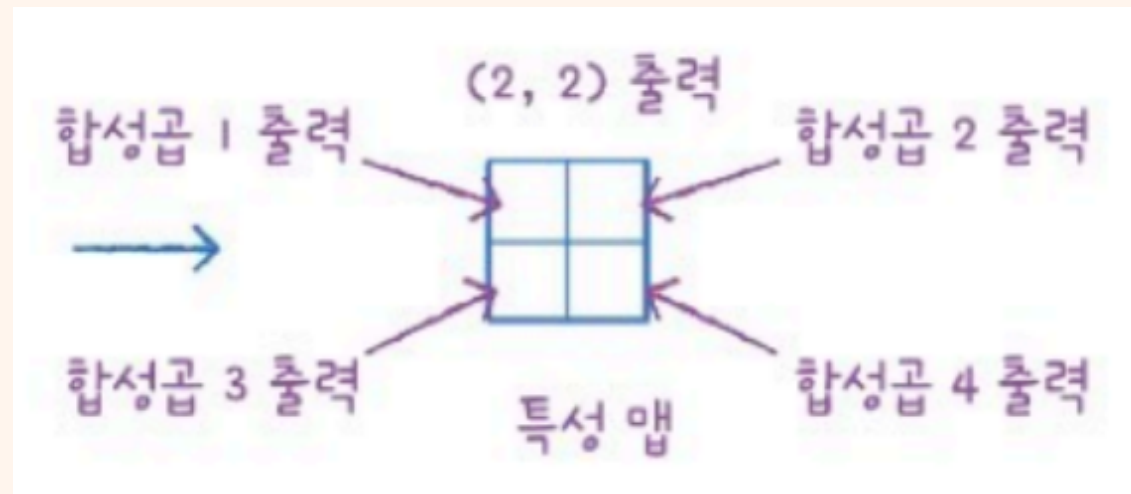


- 왼쪽 위 부터 시작하여 오른쪽으로 이동, 더 이상 이동할 수 없으면 아래로 한 칸 이동한 다음 다시 왼쪽부터 시작

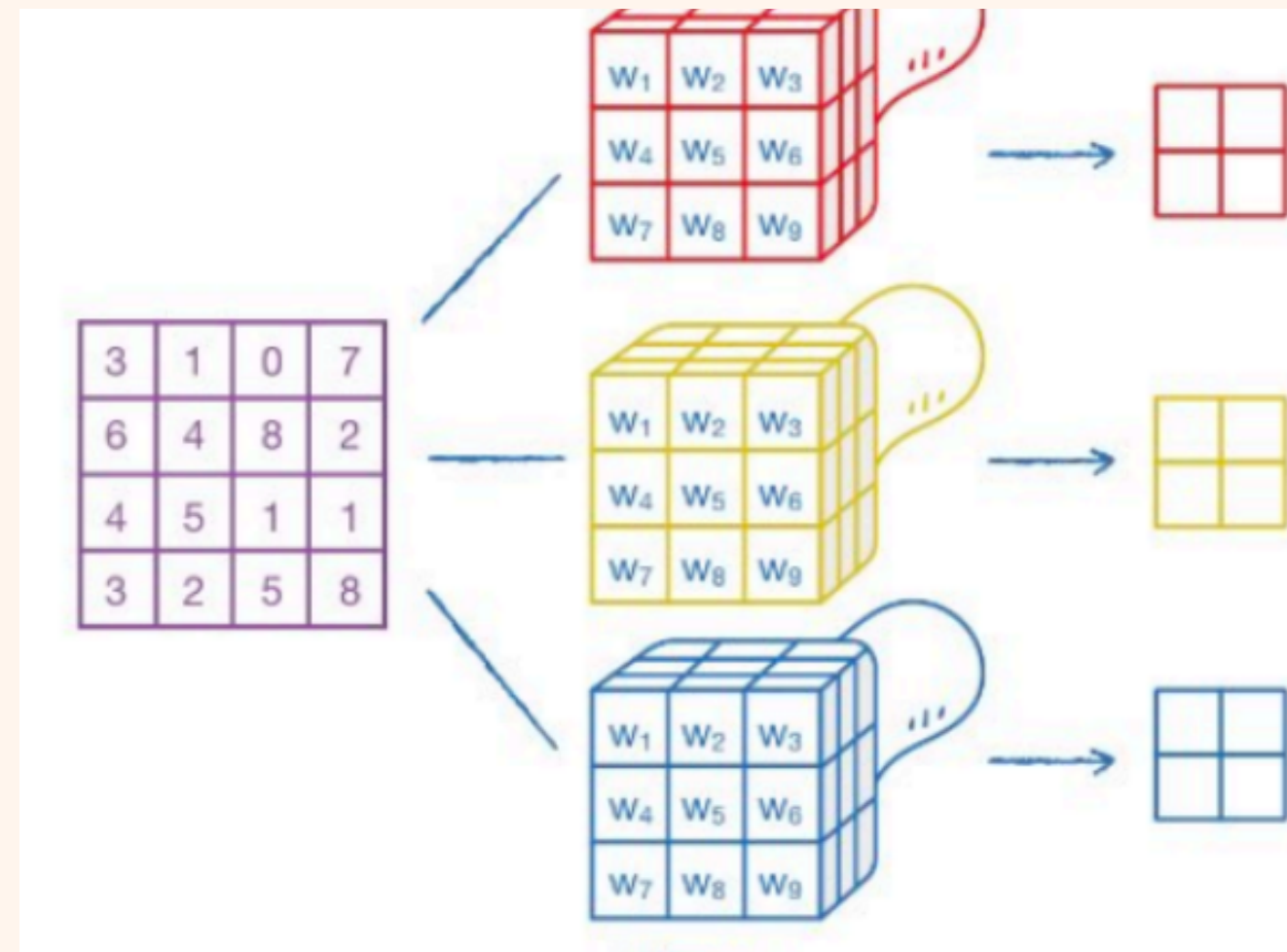


filter

feature map



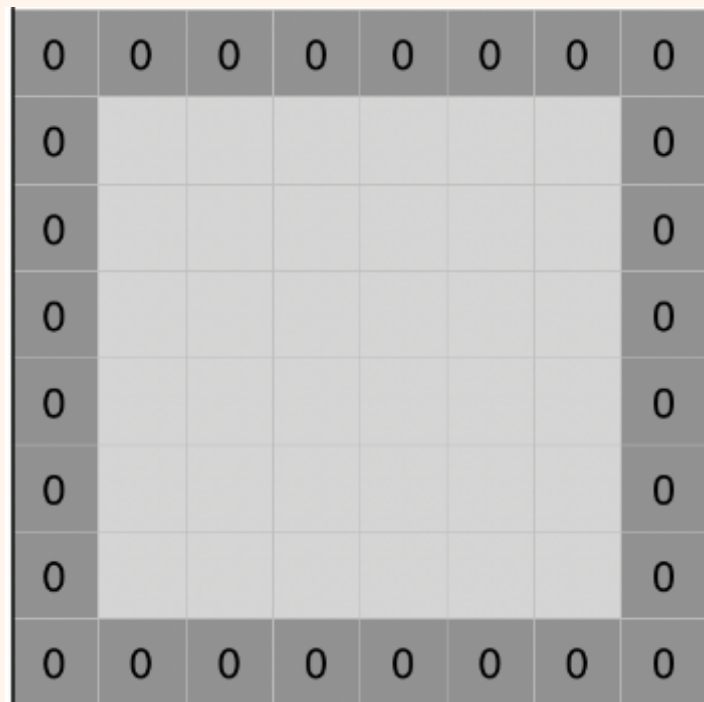
3개의 convolution filter -> 3개의 feature maps



- 여러 개의 서로 다른 필터를 사용하여 서로 다른 feature 뽑아냄
- filter 개수 많아질수록 더 다양한 특징 추출

padding

zero padding



0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

- image 주위를 0으로 둘러주는 과정
- 0으로 패딩하여 계산에 영향을 주지 않음

<padding 종류>

1. same padding

: output image가 input image의 크기와 동일하도록 패딩함

2. valid padding

: Padding을 하지 않음

filter size: (fxf), input size : (nxn)일때,

$[(n + 2p) \times (n + 2p) \text{ image}] * [(f \times f) \text{ filter}] \rightarrow [(n \times n) \text{ image}]$, $p = (f-1)/2$

padding

zero padding

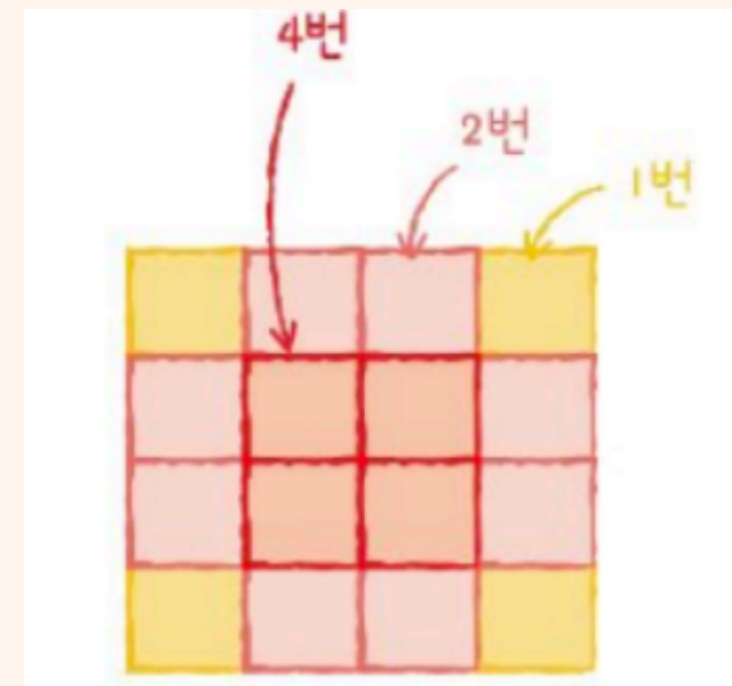
padding이 필요한 이유

1. 이미지의 데이터 축소를 막기 위해

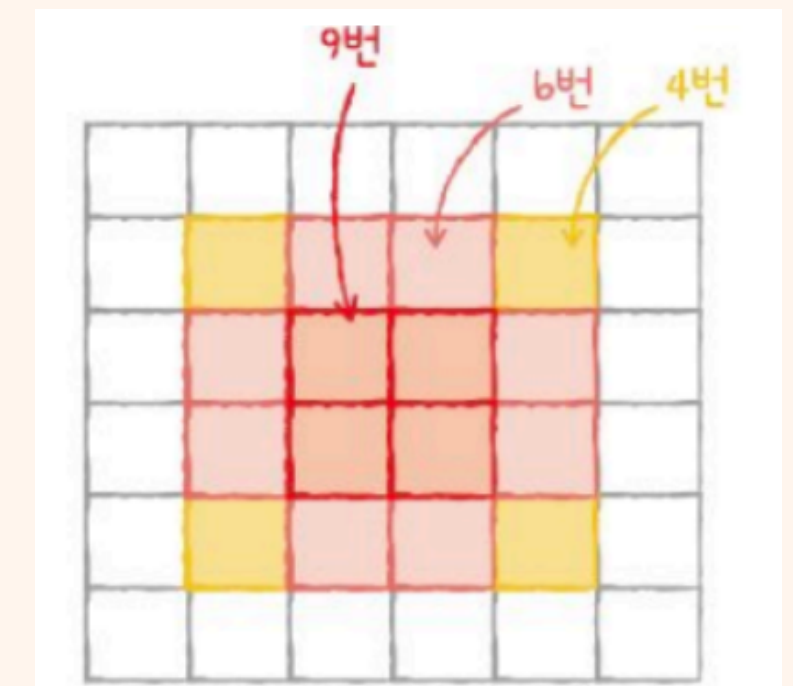
- valid padding에서 filter를 한 번 적용한 경우 사이즈는 $(n-f+1 * n-f+1)$ 로 축소
- 여러 layer를 거쳐 아웃풋 이미지가 작아질수록 정보가 축소되어 성능 감소

2. Edge pixel data를 충분히 활용하기 위해

- window sliding의 특성 상 모서리 쪽 자료가 중앙 쪽 자료보다 훨씬 적게 연산됨
- 계산되는 횟수를 비슷하게 만들어 이미지 주변에 있는 정보를 잃어버리지 않게 함



(3,3) valid padding



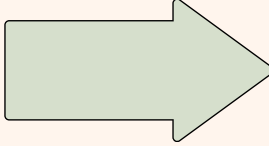
(3,3) 1 pixel padding
4:1 → 9:4

stride

input data에 필터를 적용할 때 이동할 간격

<stride = 2>

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1



1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

- 일반적으로 stride = 1로 하여 1pixel씩 이동

stride

input data에 필터를 적용할 때 이동할 간격

20이상의 stride를 사용하는 이유?

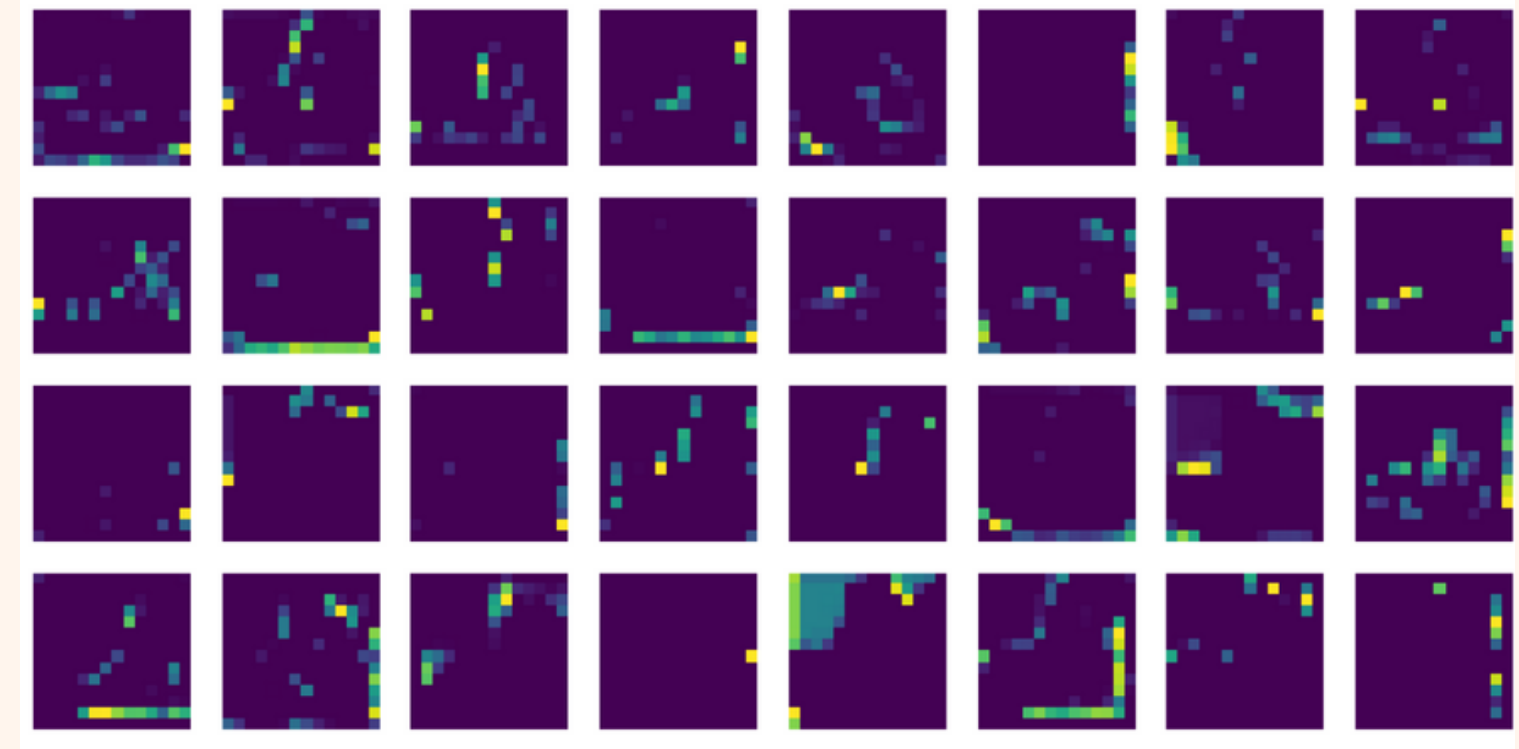
1. 특징의 다운샘플링

- 큰 보폭으로 이동하면 이미지의 작은 특징들을 묶어주어 더 큰 영역에서 하나의 특징으로 인식
- 노이즈 감소 및 로버스트성 향상
- 출력 맵의 크기가 작아져 공간적인 해상도를 줄여줌
- 공간적인 변화 감지

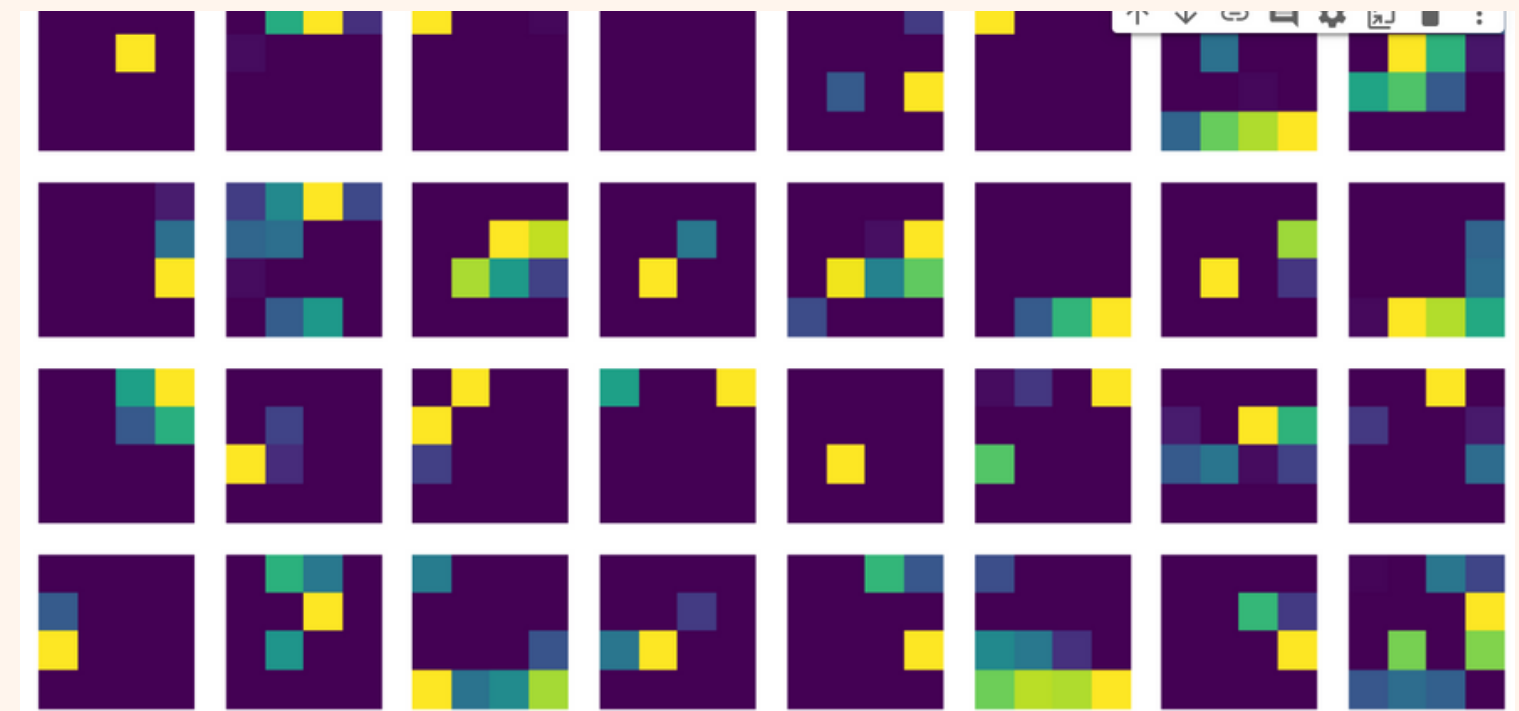
2. 계산량 감소

- 필터 이동 횟수를 줄여 계산량 감소, 연산 속도 증가

stride 1 적용한 결과



stride 2 적용한 결과



pooling

합성곱 층에서 만든 특성 맵의 가로세로 크기를 절반으로 줄임

<장점>

- feature map의 사이즈 감소시켜 메모리 절약
- 중요한 데이터만 추출하여 노이즈 제거

<단점>

- 전체정보를 반영하지 못하므로 중요한 정보가 소실 가능성 있음

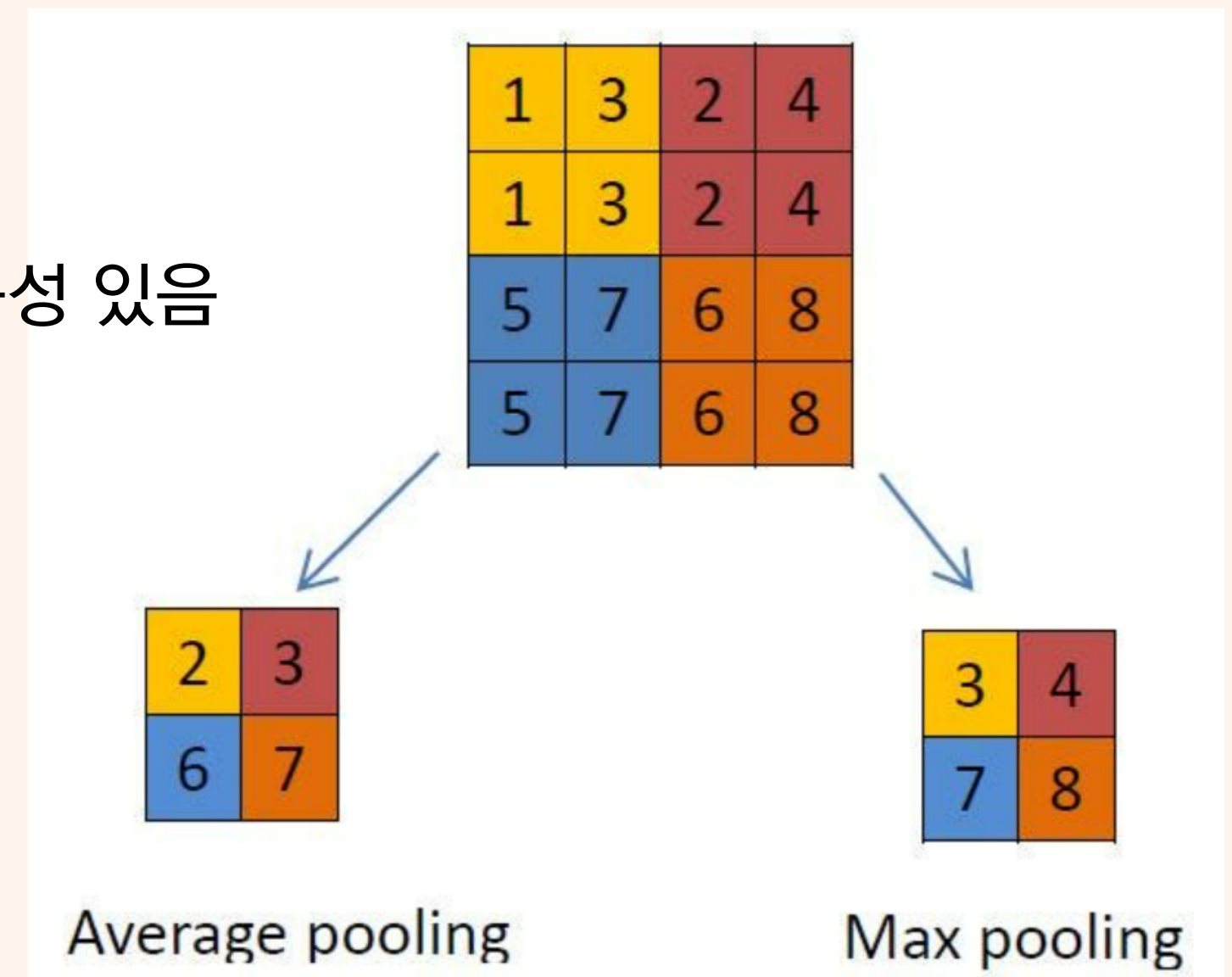
<종류>

1. max pooling

- 인접 유닛 중 가장 큰 값 추출

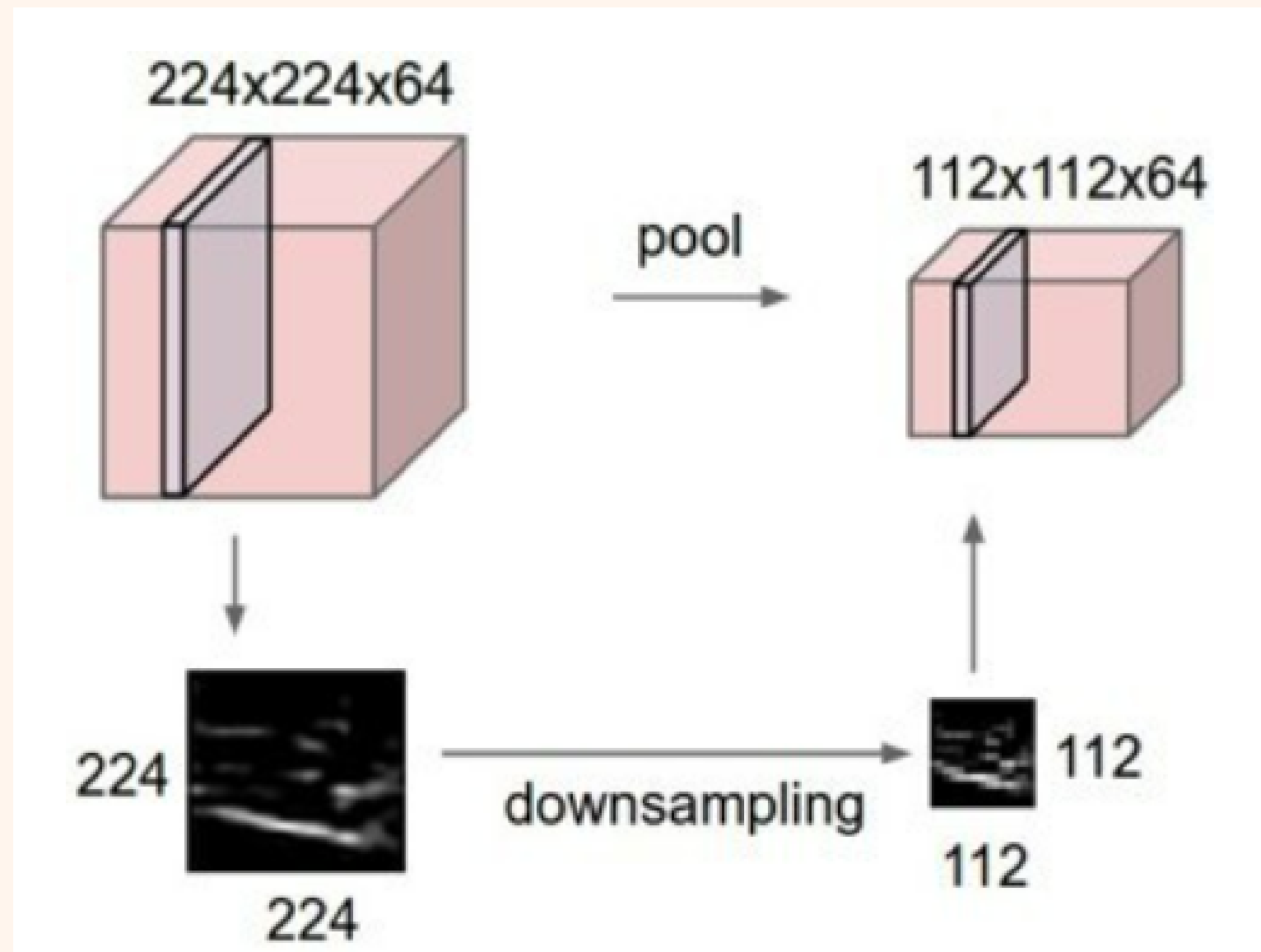
2. average pooling

- 인접 유닛에서의 평균값 추출

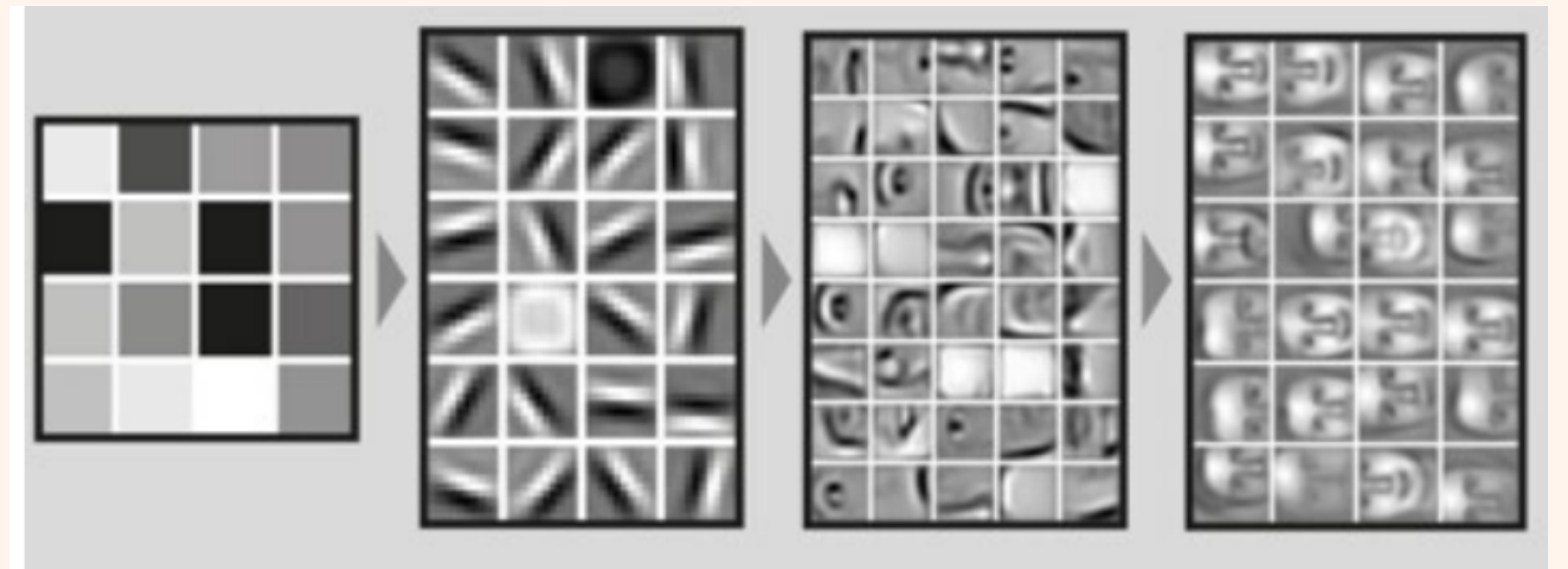


pooling

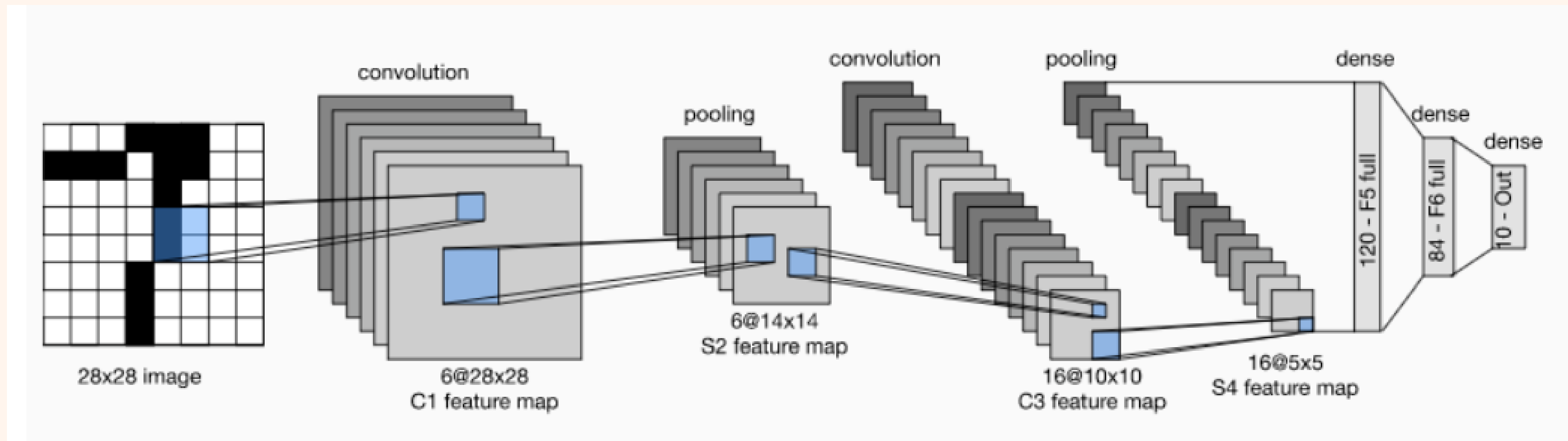
합성곱 층에서 만든 특성 맵의 가로세로 크기를 절반으로 줄임



- 기존 이미지 형태 유지하면서 크기 감소
- downsampling을 통해 conv filter의 크기가 상대적으로 커져 더 추상적인 정보 학습 가능



CNN 전체 구조



- layer 뒤로 갈수록 filter개수 증가
- > feature의 형태가 점점 복잡해지므로 더 다양한 형태의 모양 필요해지기 때문
- 마지막에 FC layer를 추가하여 분류 문제 적용

특성 맵 크기

- 입력 크기=(H, W)
- 필터 크기=(FH, FW)
- 출력 크기=(OH, OW)
- 패딩=P
- 스트라이드=S



$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

- stride가 높아질수록 특성맵의 크기 작아짐
- padding의 pixel수 많아질수록 특성맵 크기 커짐

실습 - keras API 이용해 CNN으로 MNIST 이미지 분류하기

객체 생성, 층 추가

```
In [2]: #객체 생성
model = keras.Sequential()

#합성곱 층 추가 (입력 차원 지)
model.add(keras.layers.Conv2D(32, kernel_size = 3, activation = 'relu', padding = 'same', input_

#풀링층 추가
model.add(keras.layers.MaxPooling2D(2))
```

```
In [3]: #두 번째 합성곱-풀링 층 추가
model.add(keras.layers.Conv2D(64, kernel_size = 3, activation = 'relu', padding = 'same', input_

#풀링층 추가
model.add(keras.layers.MaxPooling2D(2))
```

```
In [4]: #Flatten클래스, Dense은닉층, (dropout층), Dense출력층 추가
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'))
```

conv2d_input	input:	[(None, 28, 28, 1)]
InputLayer	output:	[(None, 28, 28, 1)]

filter 32개, same padding

conv2d	input:	(None, 28, 28, 1)
Conv2D	output:	(None, 28, 28, 32)

max pooling

max_pooling2d	input:	(None, 28, 28, 32)
MaxPooling2D	output:	(None, 14, 14, 32)

filter 64개, same padding

conv2d_1	input:	(None, 14, 14, 32)
Conv2D	output:	(None, 14, 14, 64)

max pooling

max_pooling2d_1	input:	(None, 14, 14, 64)
MaxPooling2D	output:	(None, 7, 7, 64)

flatten	input:	(None, 7, 7, 64)
Flatten	output:	(None, 3136)

dense	input:	(None, 3136)
Dense	output:	(None, 100)

dropout	input:	(None, 100)
Dropout	output:	(None, 100)

dense_1	input:	(None, 100)
Dense	output:	(None, 10)

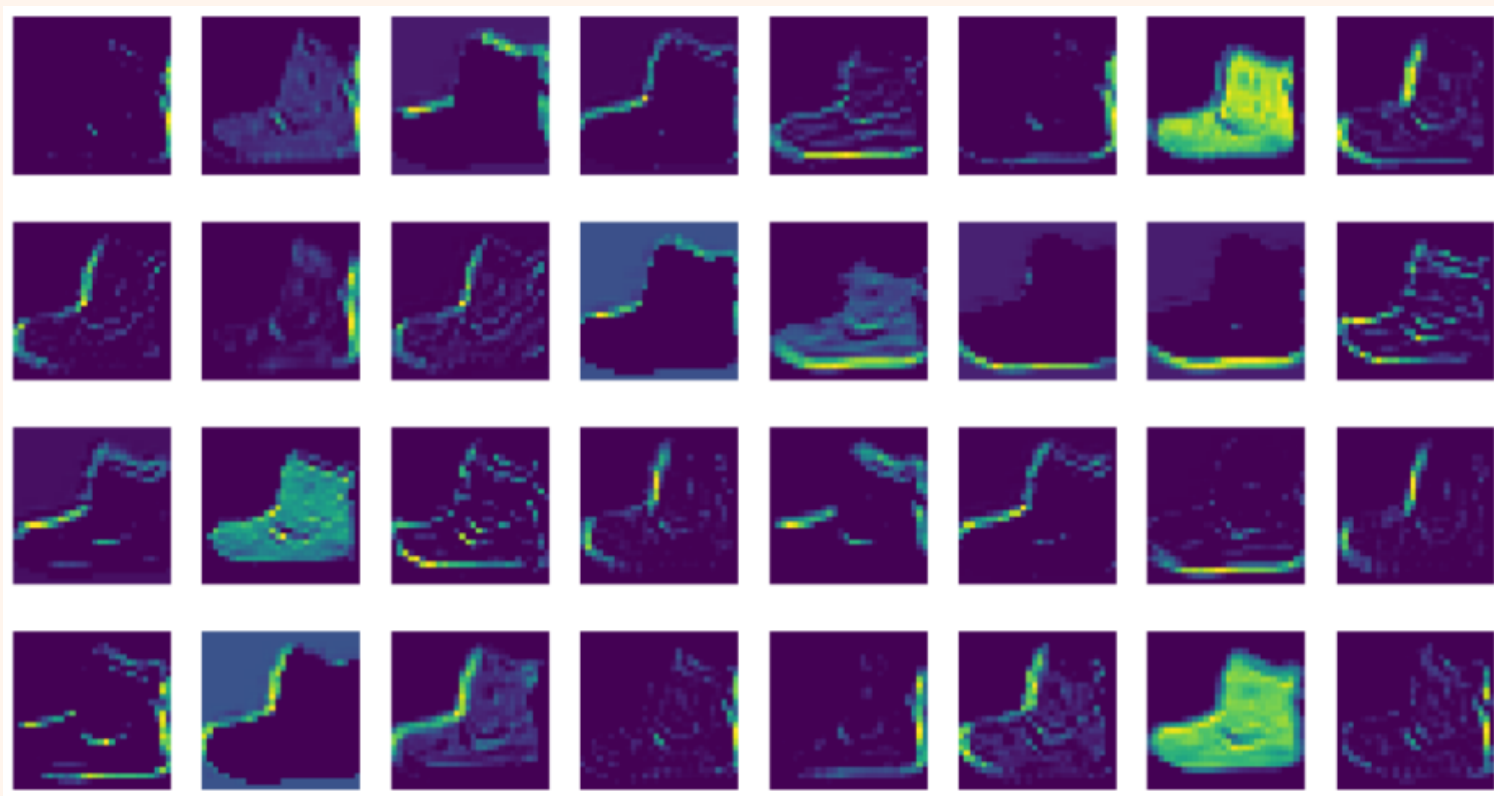
가중치 시각화

- 초기 무작위로 지정되어 학습을 통해 유용한 패턴 학습

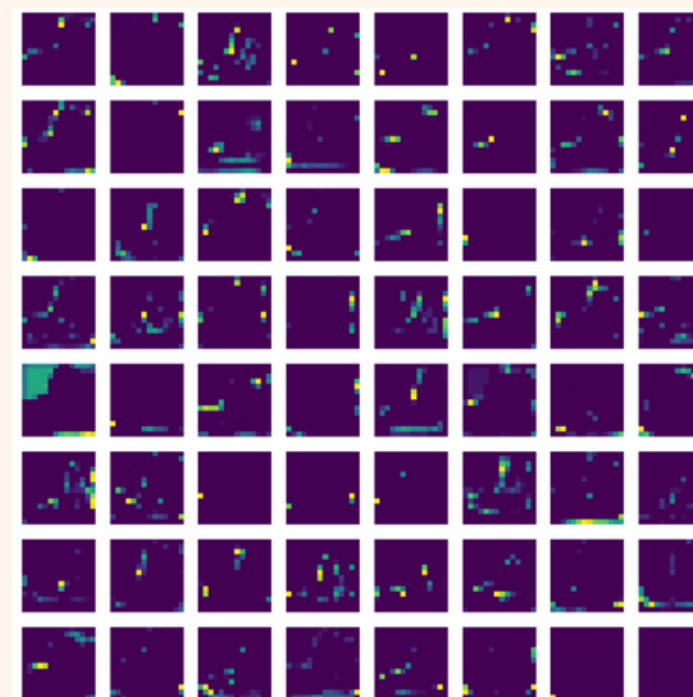


특성맵 시각화

<첫 번째 Cov layer 통과 후>



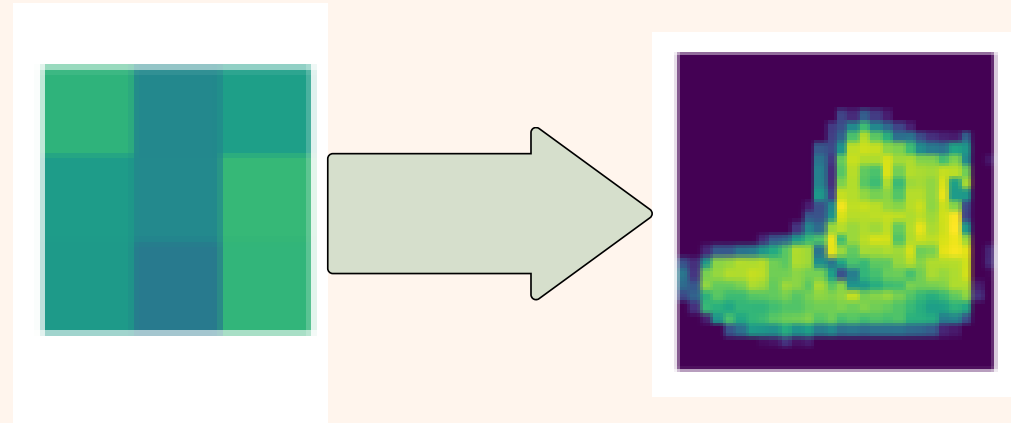
<두 번째 Cov layer 통과 후>



- 보다 추상적인 특징 학습

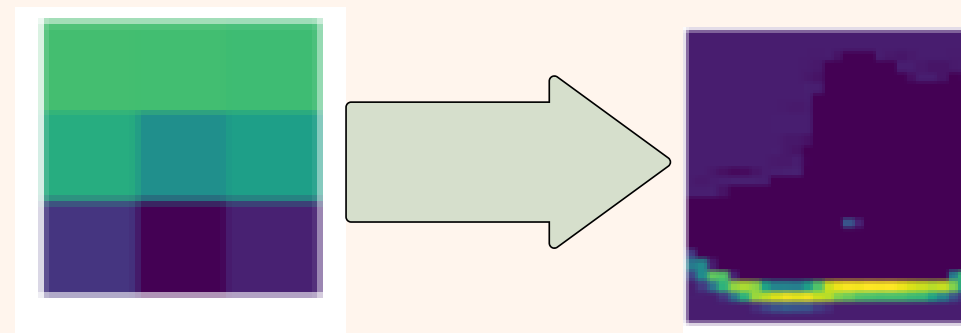
<대체적으로 밝은(높은 값을 가진) 필터>

- 전체적으로 활성화



<수평적으로 밝은(높은 값을 가진) 필터>

- 수평선 감지



<수직적으로 밝은(높은 값을 가진) 필터>

- 수직선 감지

