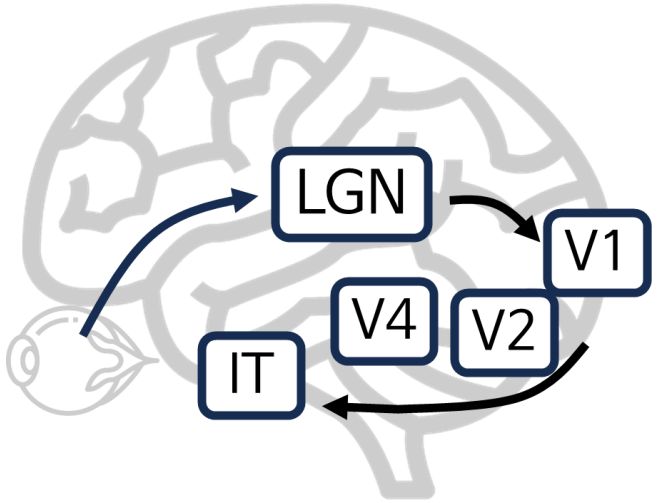
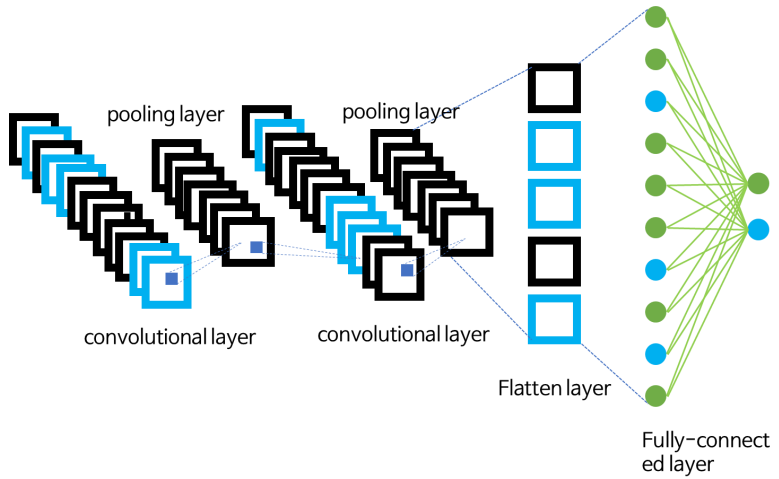
Deep Learning101

CNN (합성곱신경망)

Convolutional Neural Network





안녕하세요 신박AI입니다

신박AI

오늘은 딥러닝의 중요한 구성 요소 중 하나인
CNN에 대해 알아보려고 합니다

신박AI

CNN은 Convolutional Neural Network의
약자로,

신박AI

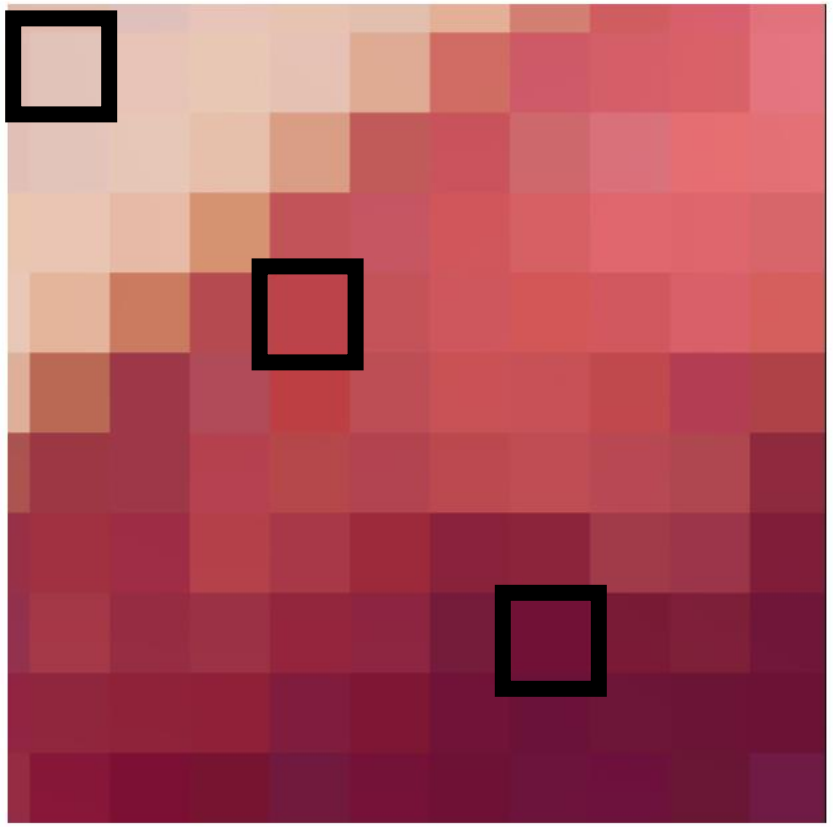
이미지 처리와 패턴 인식에 탁월한 성능을
보여주는 신경망입니다

신박AI

이미지image는 픽셀로 이루어진 격자 형태의 데이터입니다

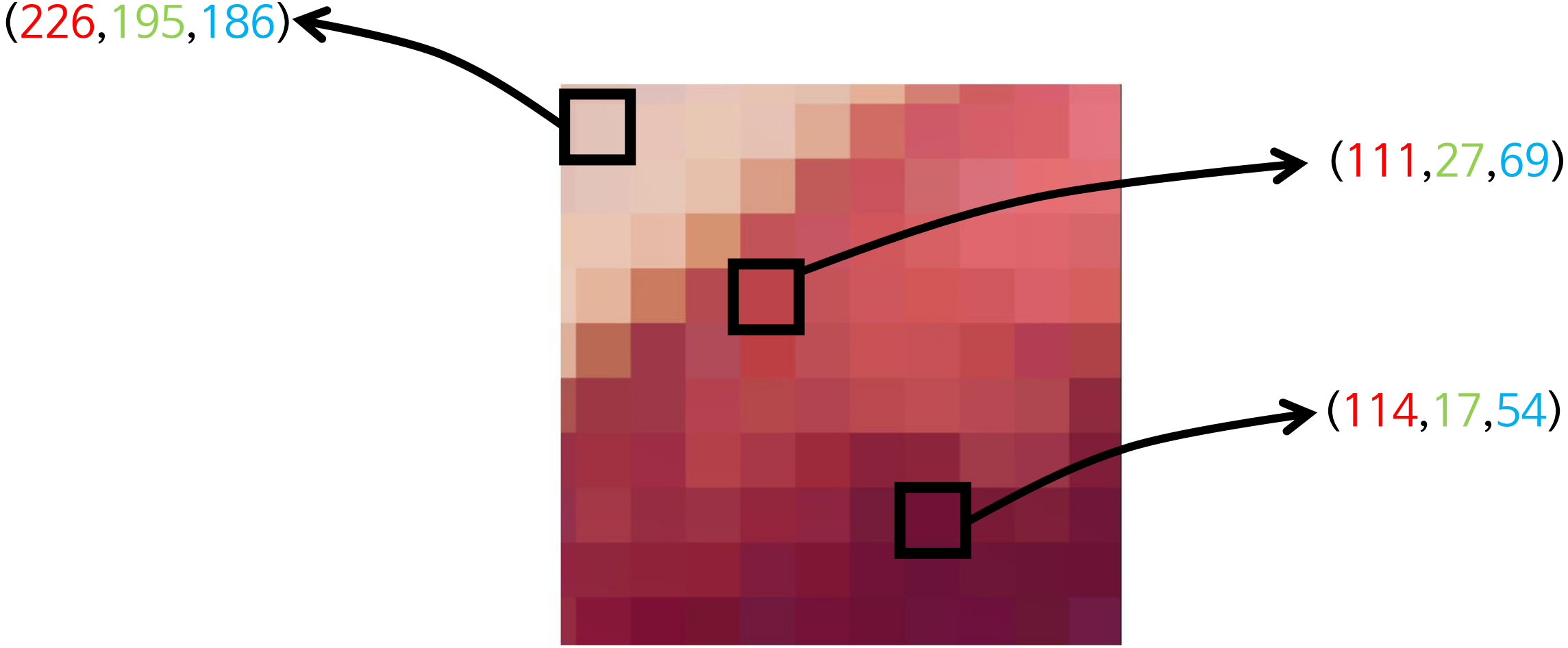
신박AI

이미지image는 픽셀로 이루어져 있고



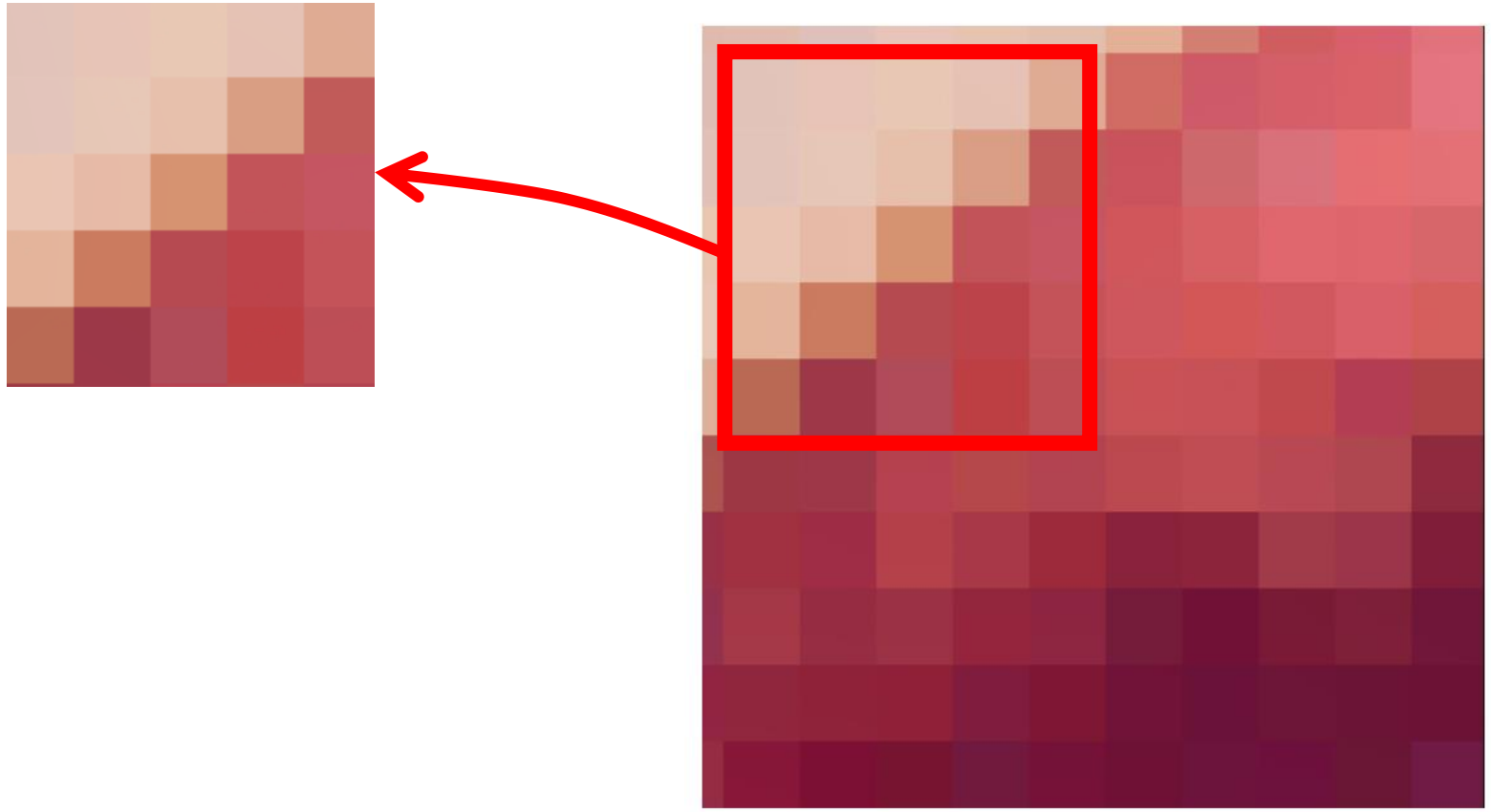
신박AI

각각의 픽셀은 rgb값을 가진 데이터입니다



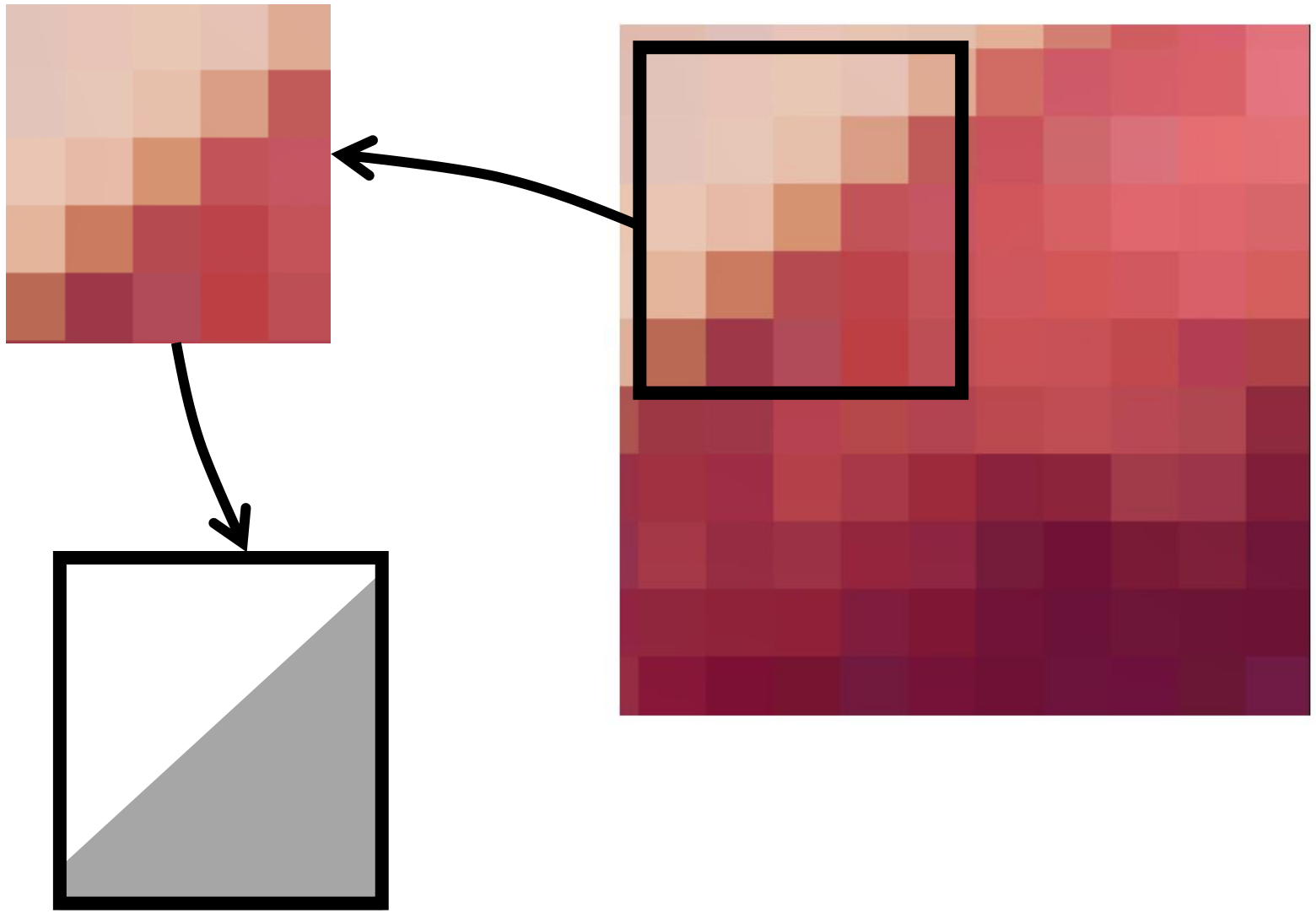
신박AI

CNN은 이러한 이미지 데이터의 공간적 특징을 추출하여 학습하고

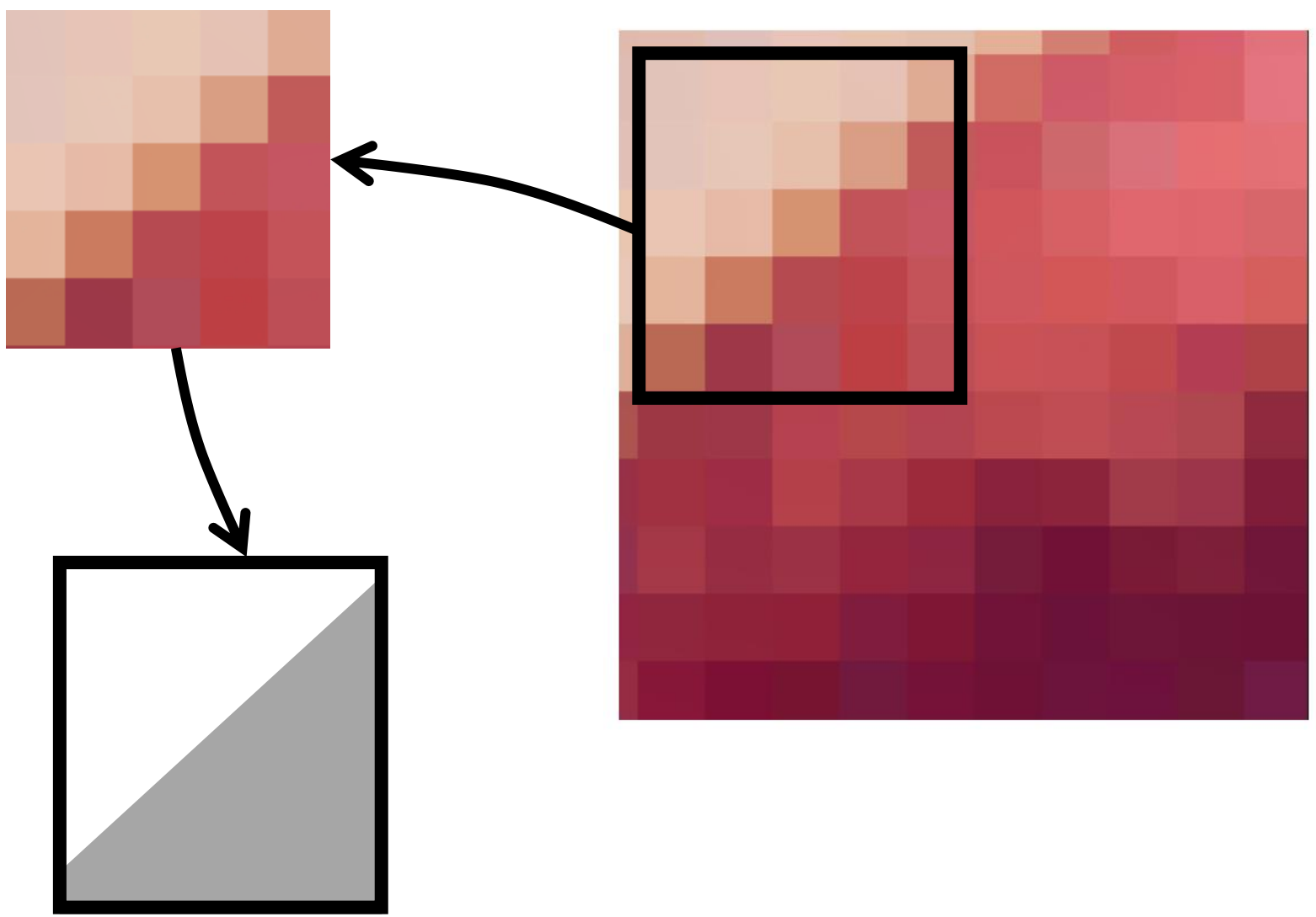


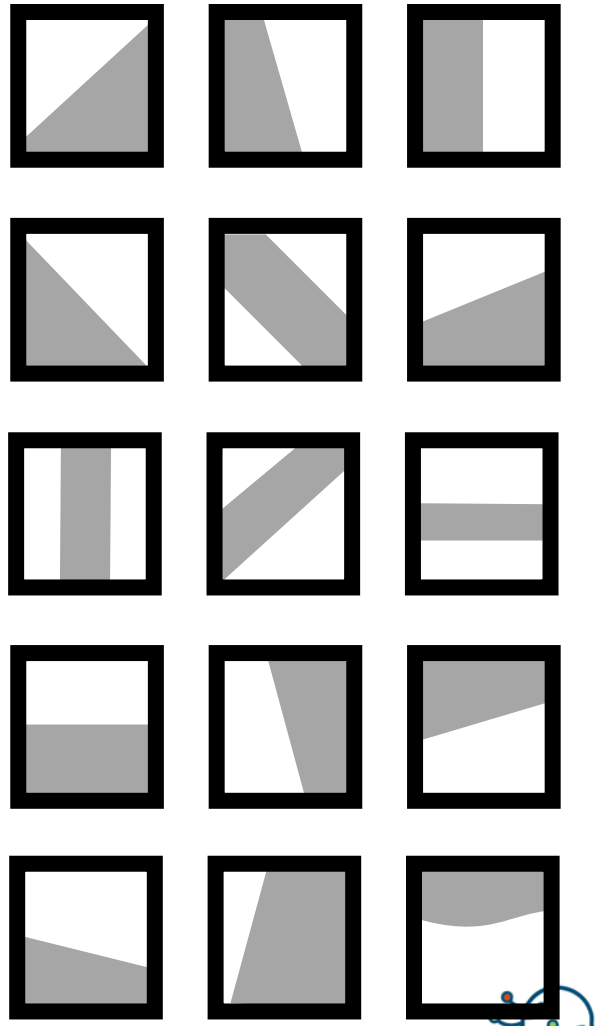
신박AI

이를 기반으로 패턴을 인식하는 데 사용됩니다

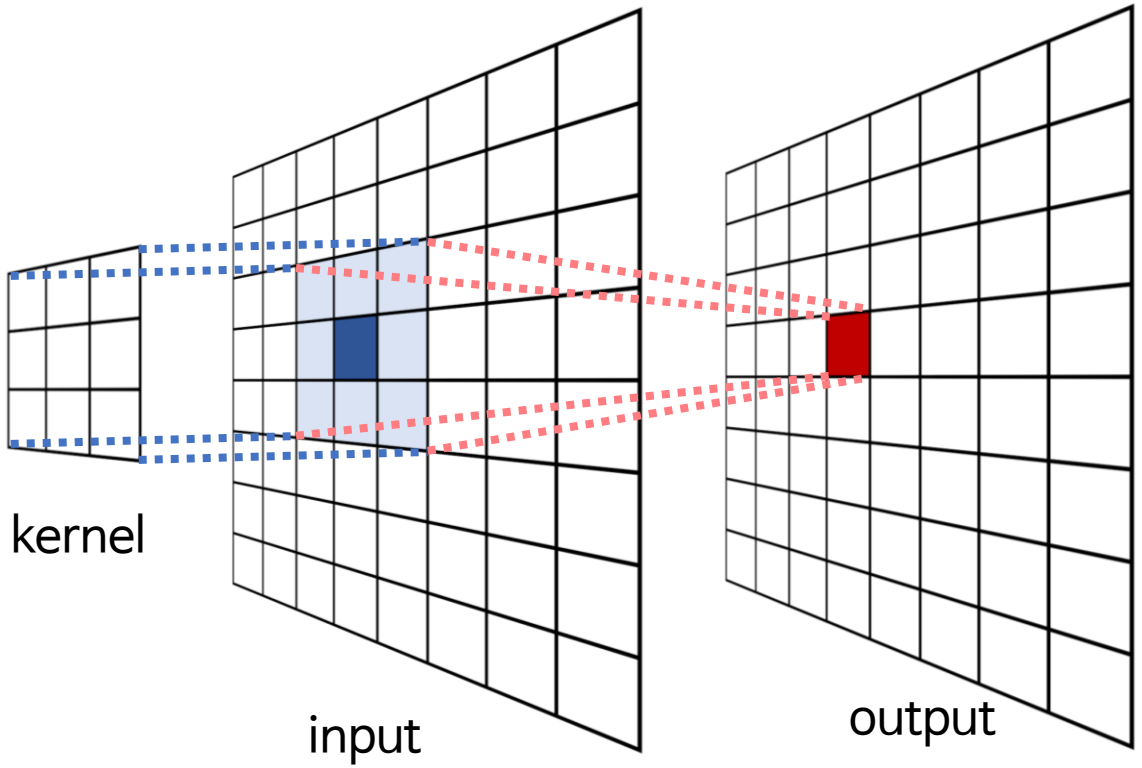


이를 기반으로 패턴을 인식하는 데 사용됩니다



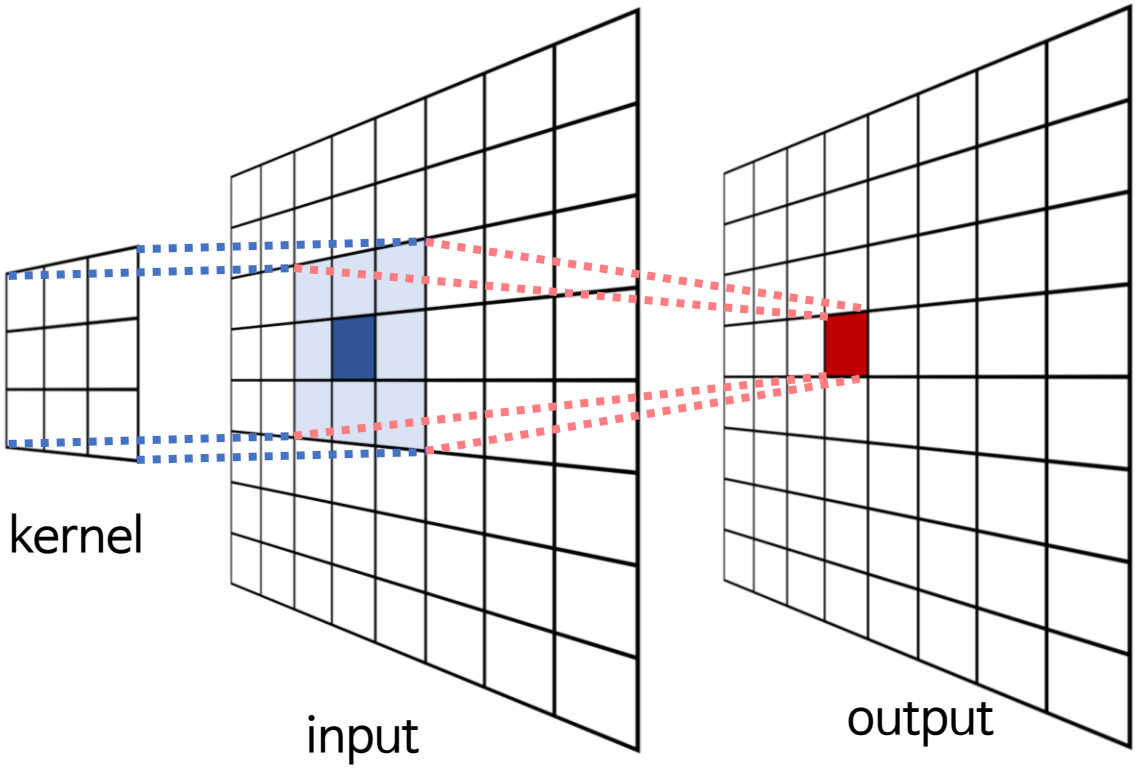


이를 위해 CNN은 주로

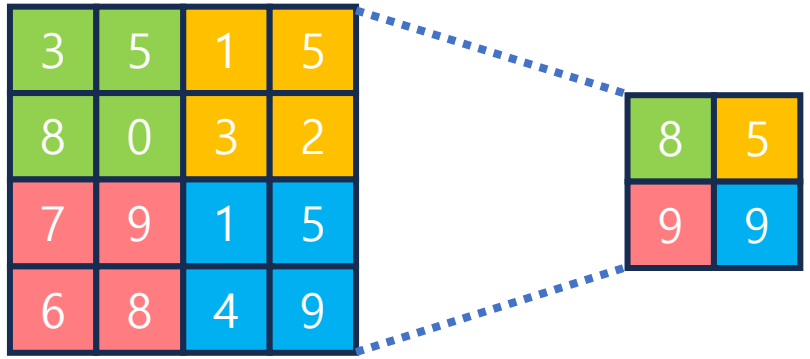


합성곱 층 Convolutional layer

신박AI

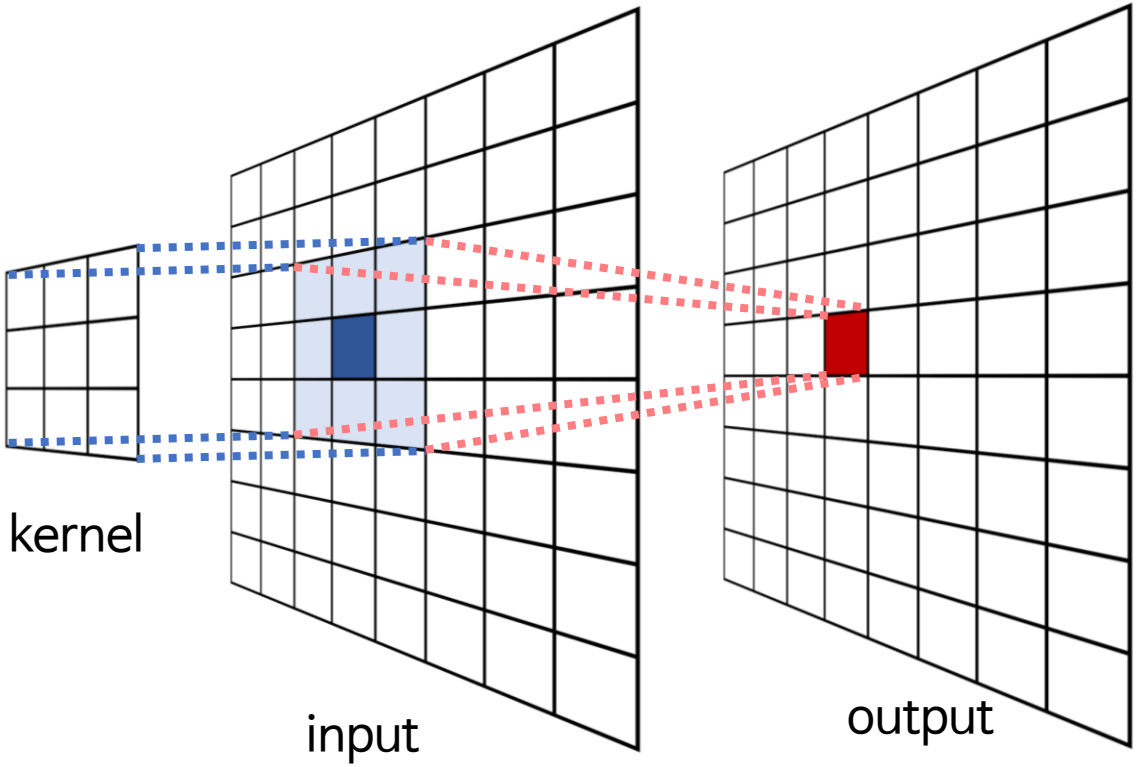


합성곱 층 Convolutional layer

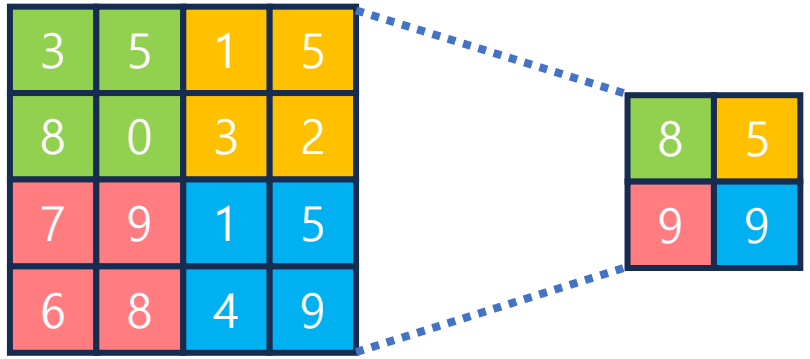


풀링 층 Pooling layer

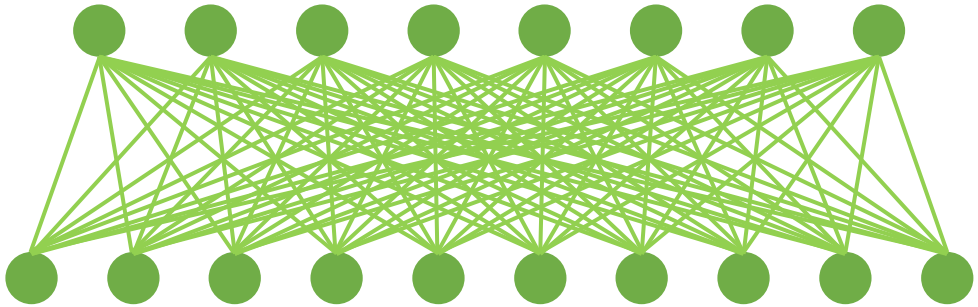
신박AI



합성곱 층 Convolutional layer



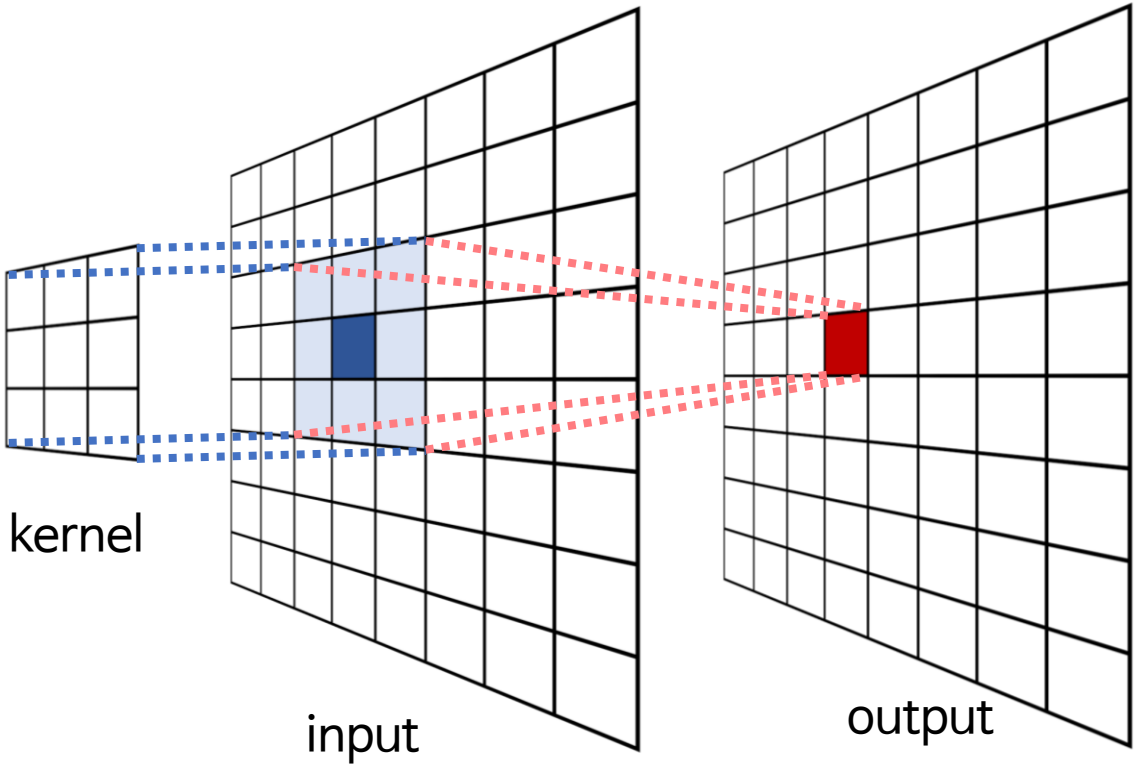
풀링 층 Pooling layer



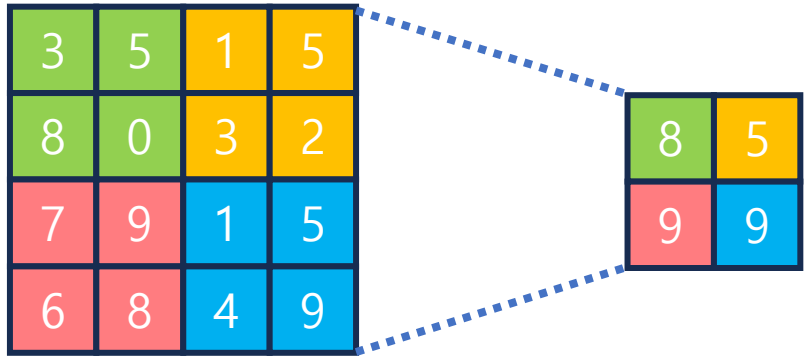
밀집 층 Fully-connected layer
Dense layer

신박AI

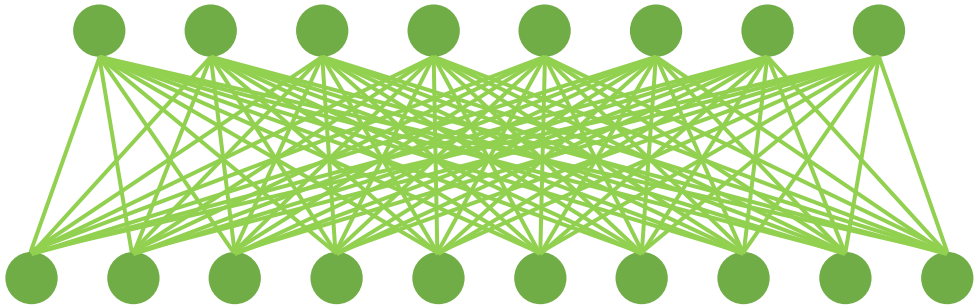
CNN은 이러한 층들로 구성됩니다



합성곱 층 Convolutional layer



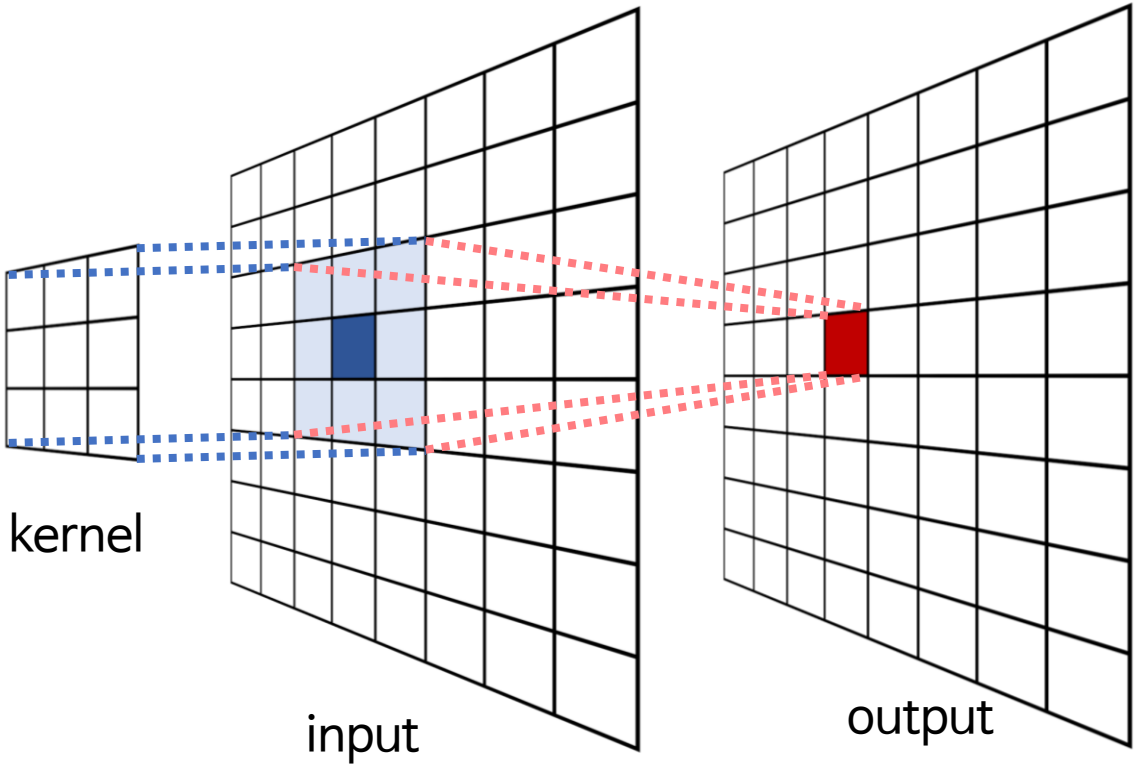
풀링 층 Pooling layer



밀집 층 Fully-connected layer
Dense layer

신박AI

먼저, 합성곱 층은 CNN에서 이미지 처리와 패턴 인식을 위해 주로 사용되는 중요한 구성 요소입니다

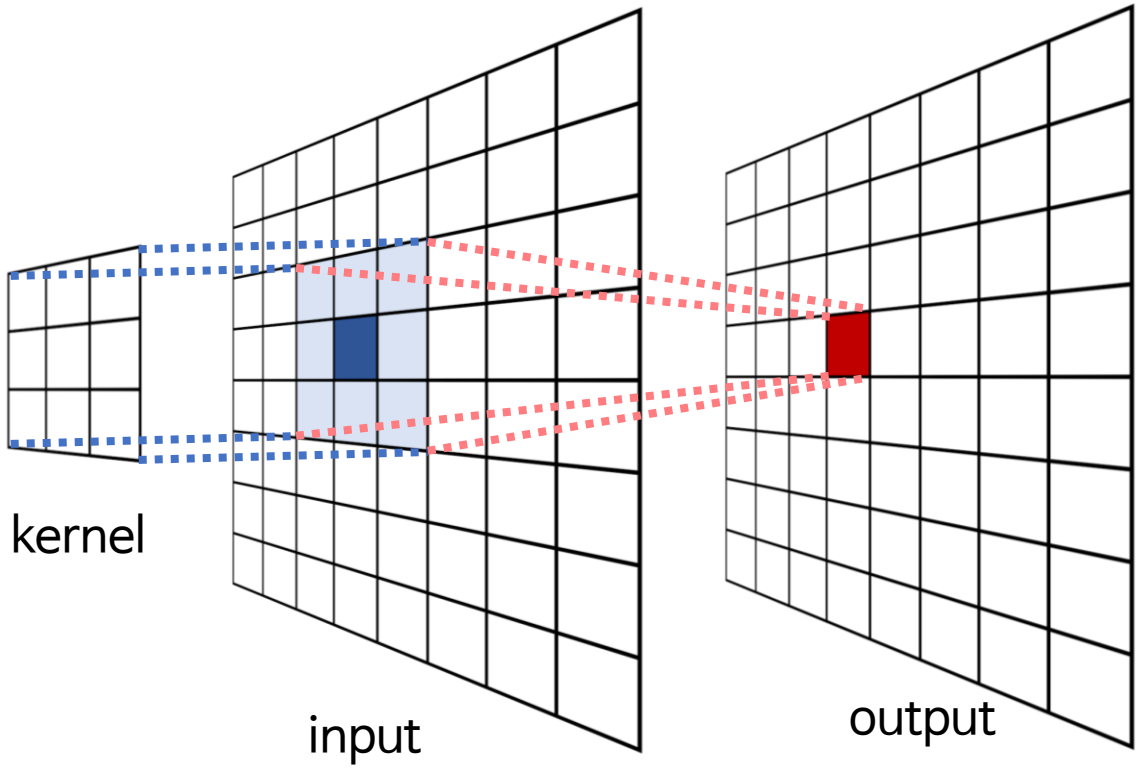


합성곱 층 Convolutional layer

풀링 층 Pooling layer

밀집 층 Fully-connected layer
Dense layer

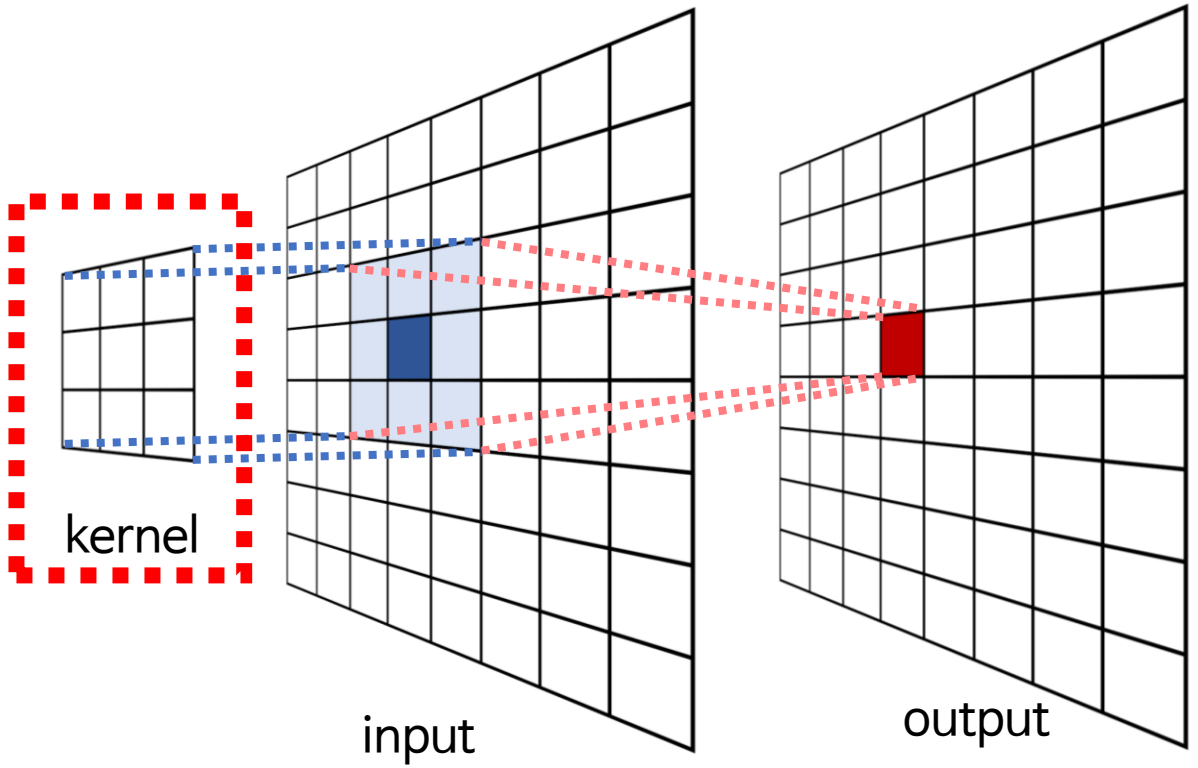
먼저, 합성곱 층은 CNN에서 이미지 처리와 패턴 인식을 위해 주로 사용되는 중요한 구성 요소입니다



합성곱 층 Convolutional layer

신박AI

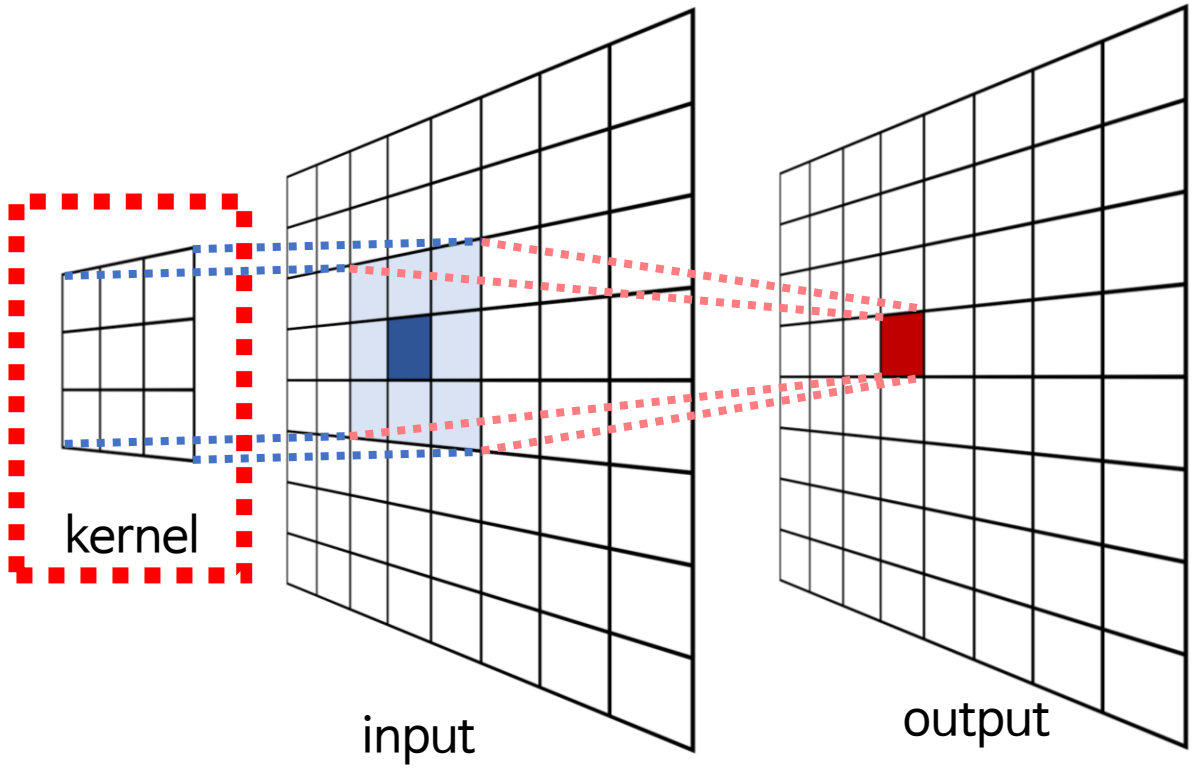
합성곱 층은 입력 이미지와 작은 크기의 필터(kernel) 간의 합성곱 연산을 통해 새로운 특성 맵(feature map)을 생성합니다



합성곱 층 Convolutional layer

신박AI

이를 통해 이미지의 지역적인 패턴을 감지하고 추출할 수 있습니다



합성곱 층 Convolutional layer

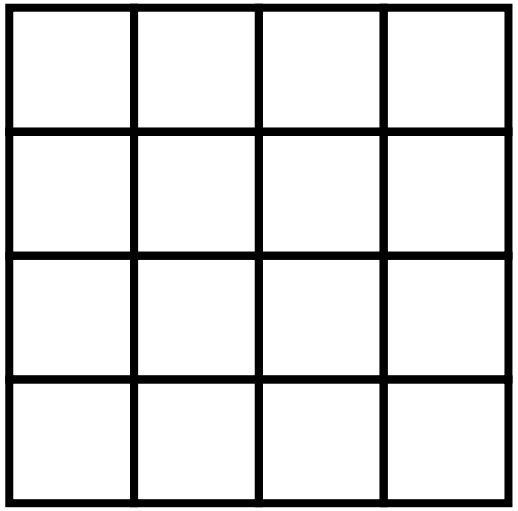
신박AI

합성곱 연산은 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정을 말합니다

0	0	1
-1	1	-1
1	-1	0

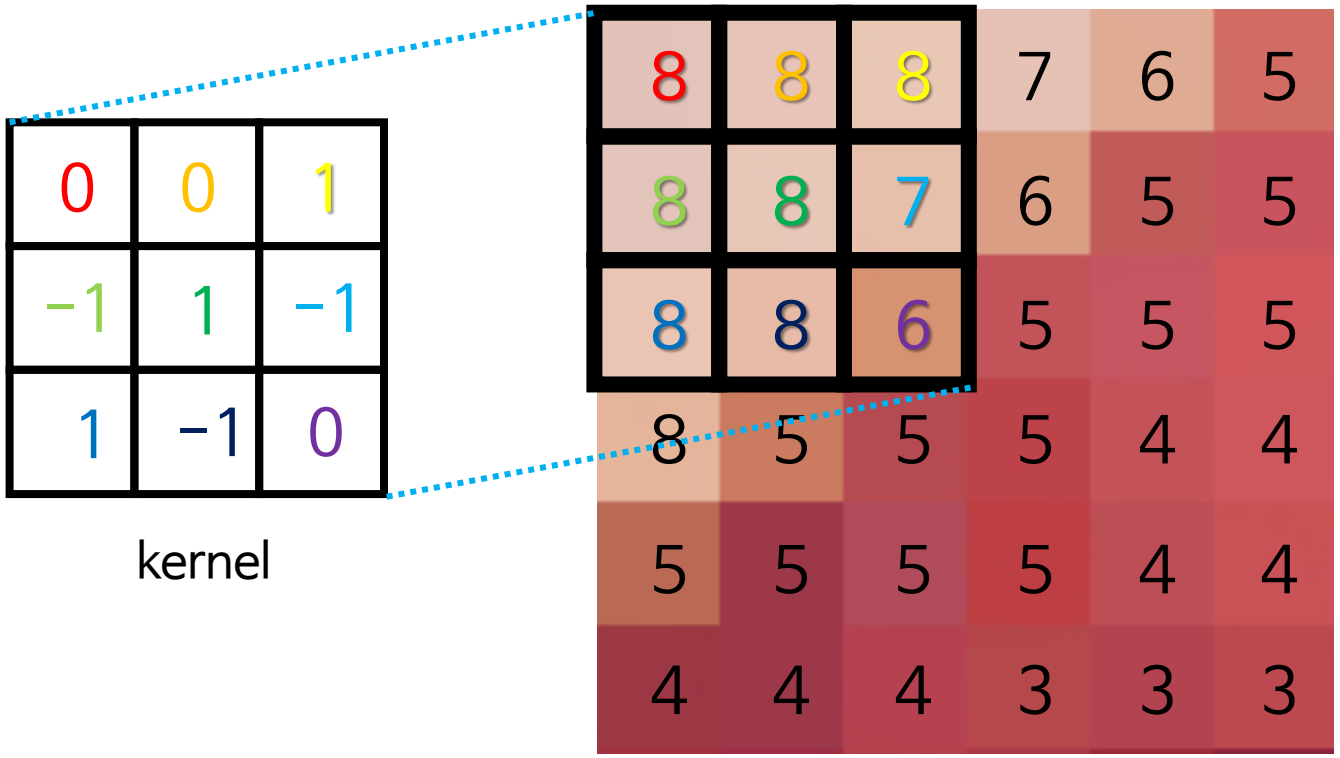
kernel

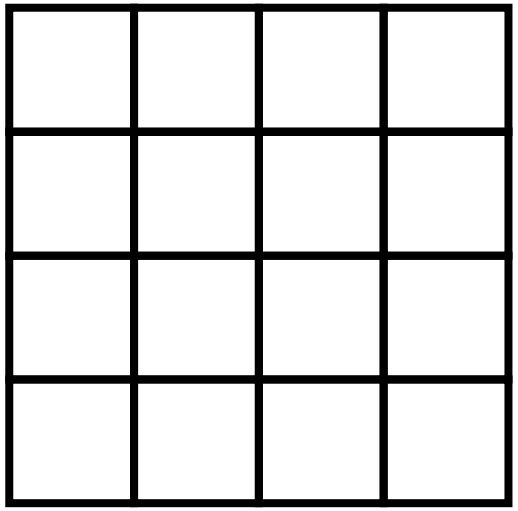
8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3



신박AI

합성곱 연산은 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정을 말합니다



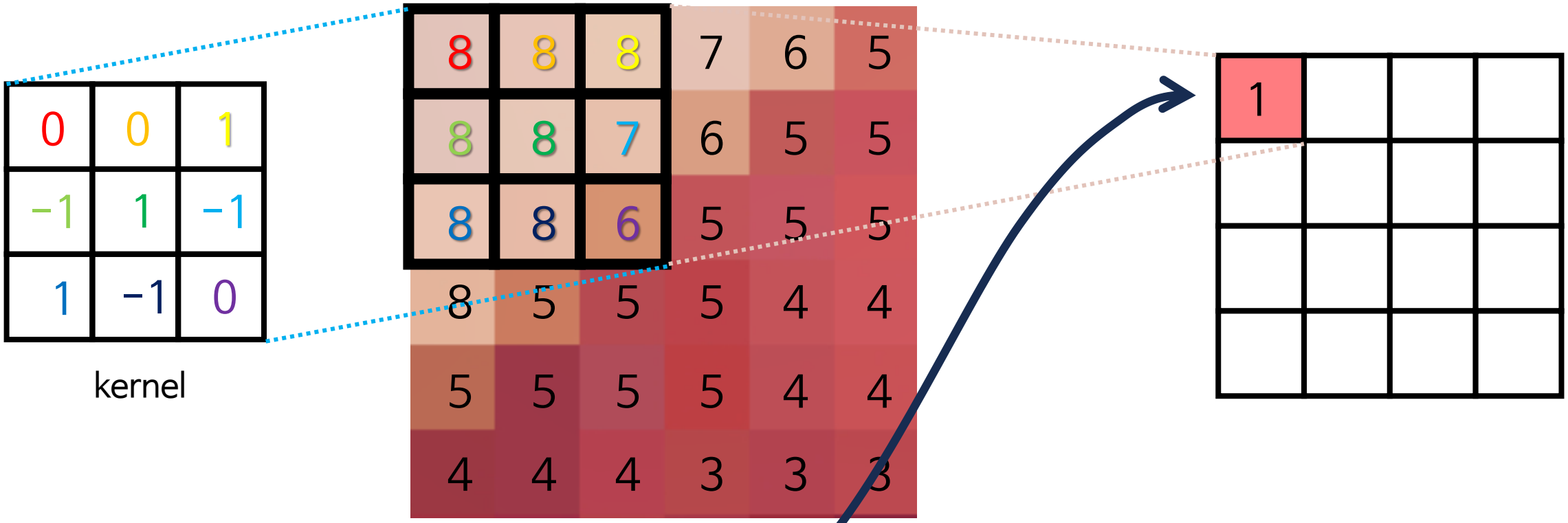


8 x 0 + 8 x 0 + 8x1 + 8x(−1) + 8x1 + 7x(−1) + 8x1 + 8x(−1) + 6 x 0

= 1

신박AI

합성곱 연산은 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정을 말합니다

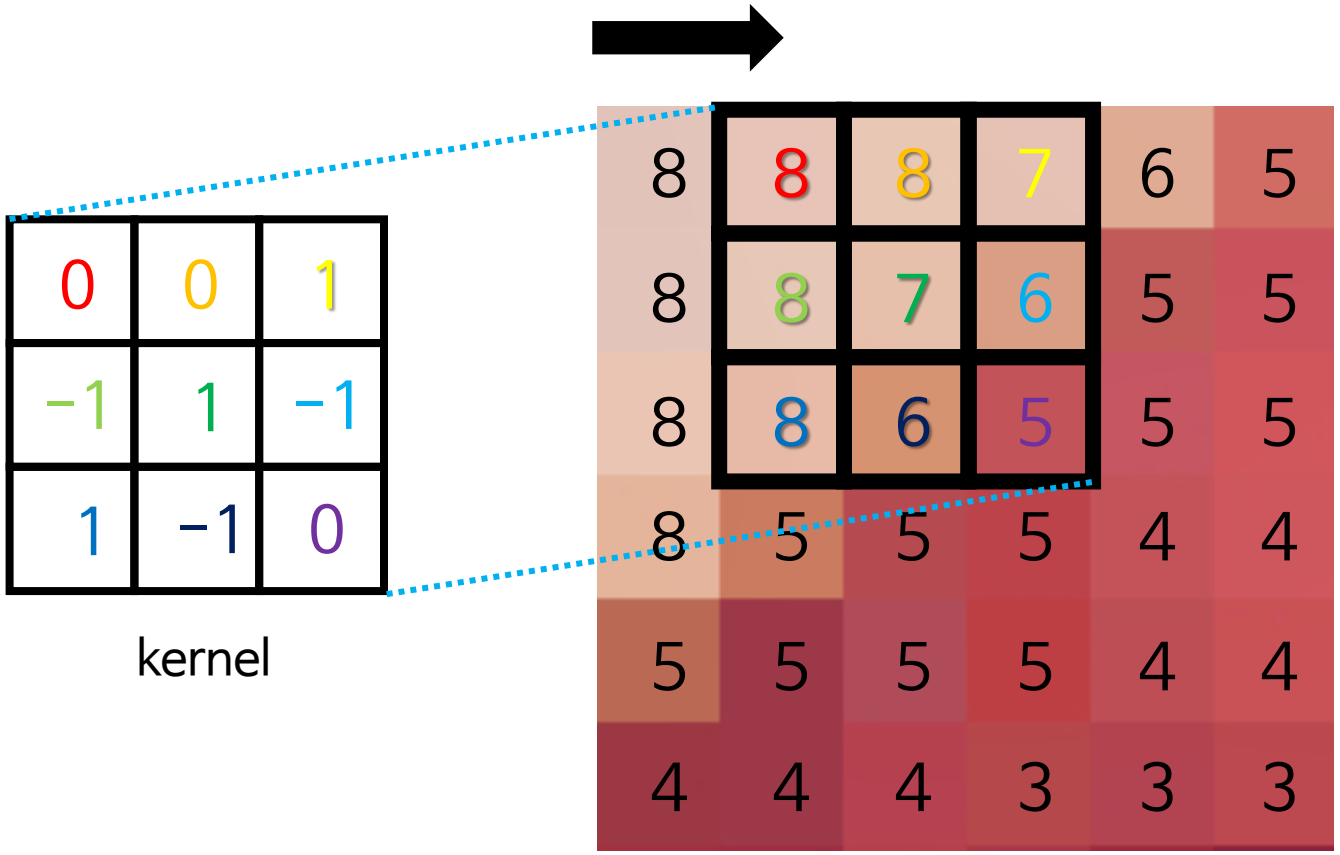


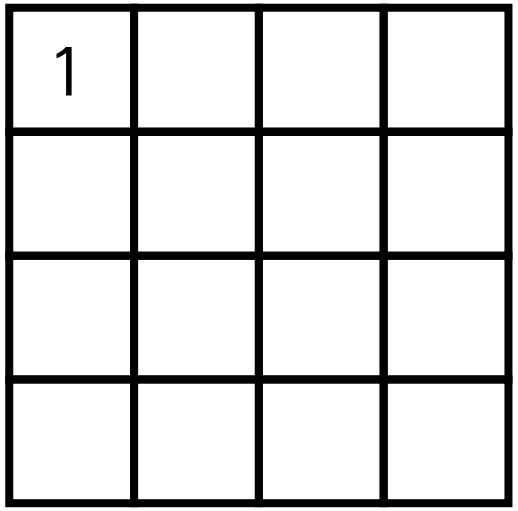
8 x 0 + 8 x 0 + 8x1 + 8x(−1) + 8x1 + 7x(−1) + 8x1 + 8x(−1) + 6 x 0

= 1

신박AI

이미지 위에서 슬라이딩하면서 필터와 이미지의 요소별 곱을 계산합니다



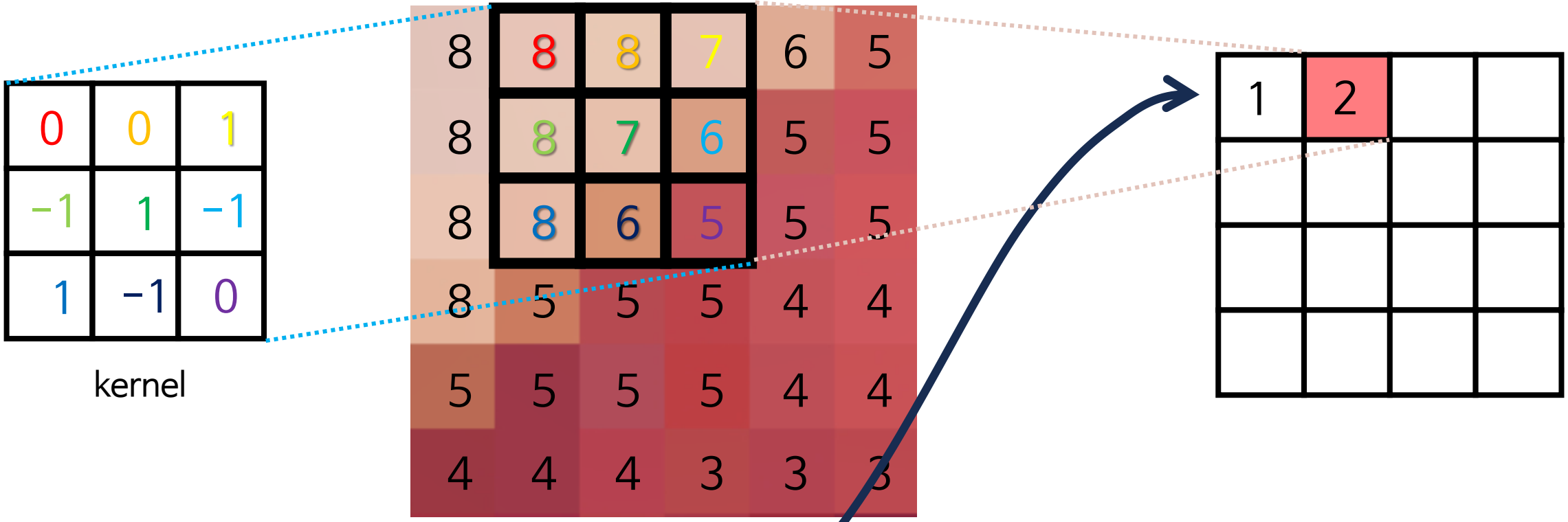


$8 \times 0 + 8 \times 0 + 7 \times 1 + 8 \times (-1) + 7 \times 1 + 6 \times (-1) + 8 \times 1 + 6 \times (-1) + 5 \times 0$

$= 2$

신박AI

이미지 위에서 슬라이딩하면서 필터와 이미지의 요소별 곱을 계산합니다

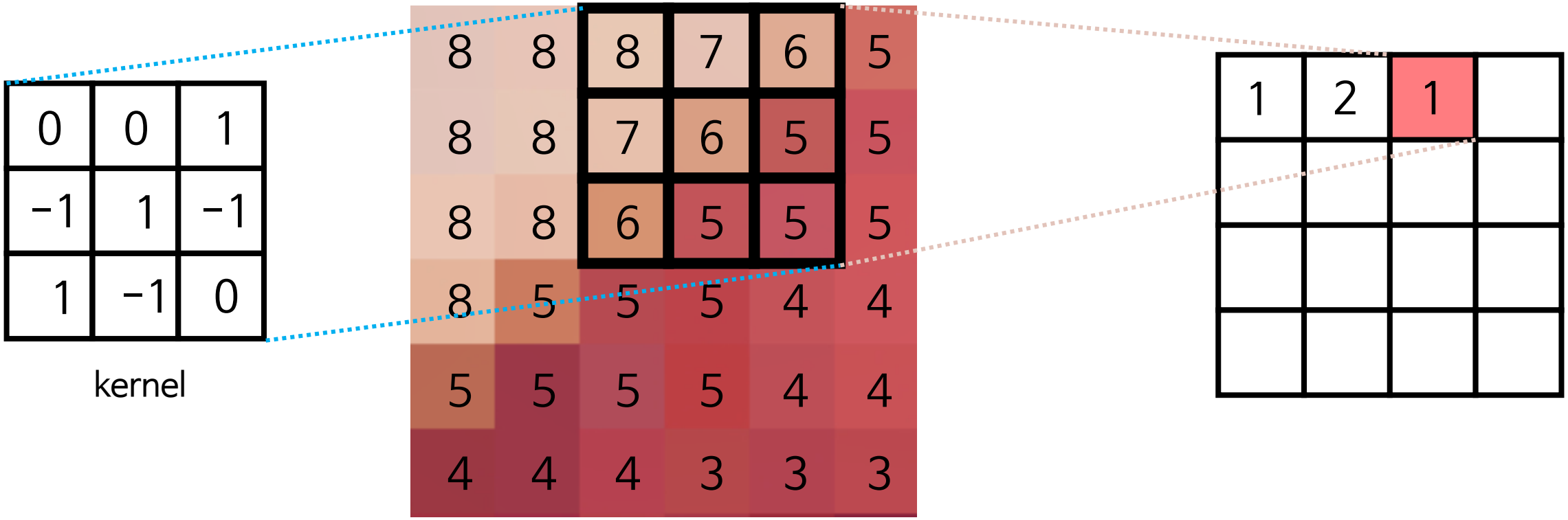


$8 \times 0 + 8 \times 0 + 7 \times 1 + 8 \times (-1) + 7 \times 1 + 6 \times (-1) + 8 \times 1 + 6 \times (-1) + 5 \times 0$

$= 2$

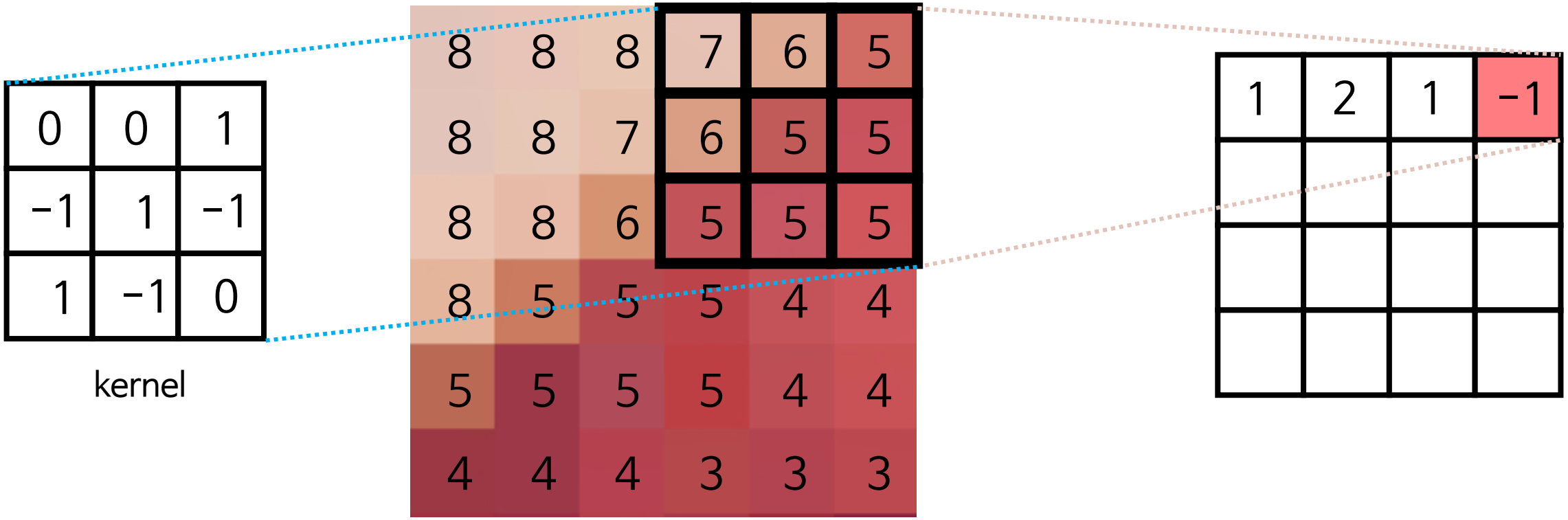
신박AI

같은 방식으로 커널을 이동해가며 합성곱의 값을 구해줍니다



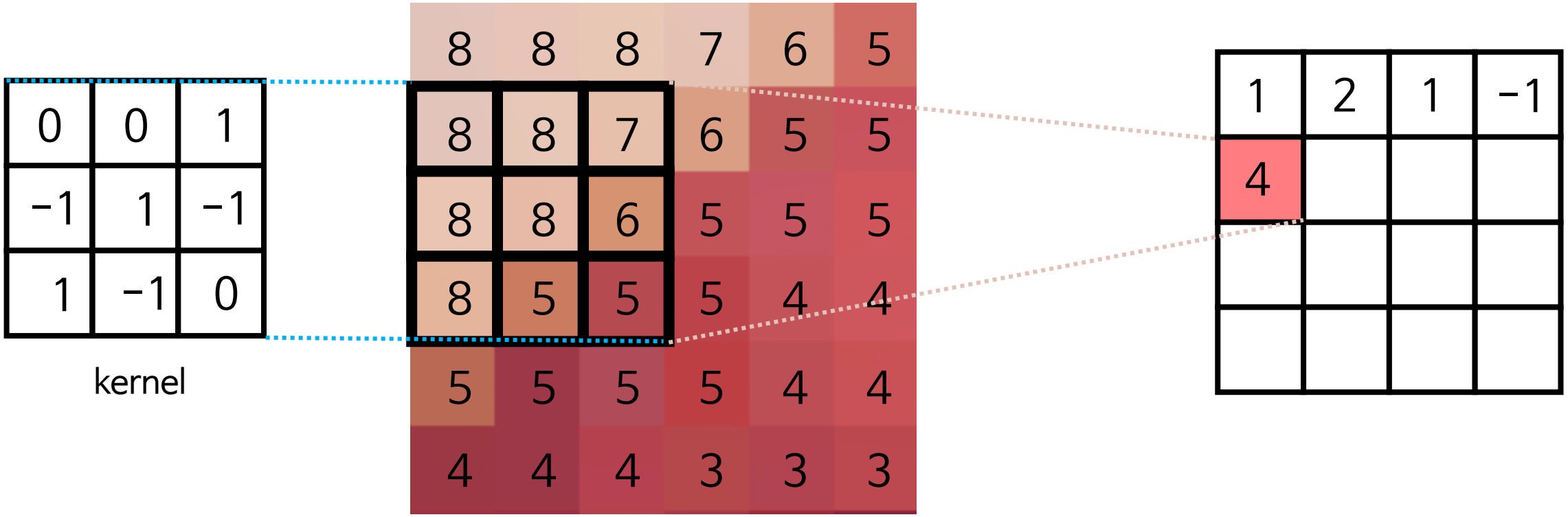
신박AI

같은 방식으로 커널을 이동해가며 합성곱의 값을 구해줍니다



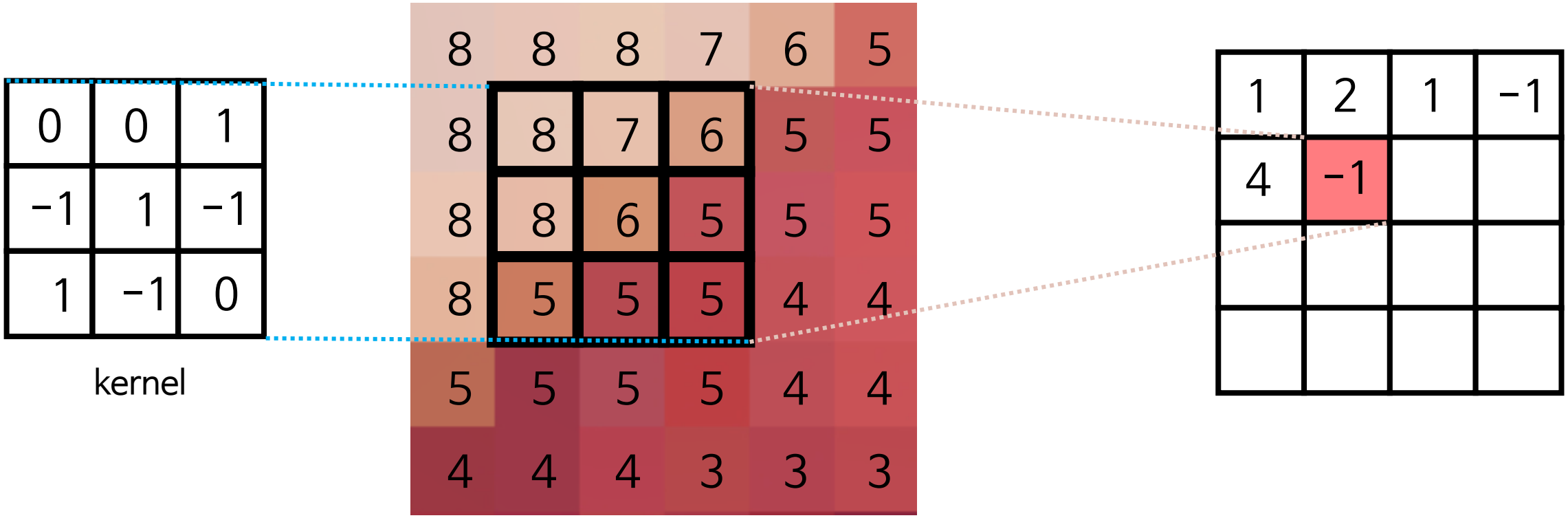
신박AI

같은 방식으로 커널을 이동해가며 합성곱의 값을 구해줍니다



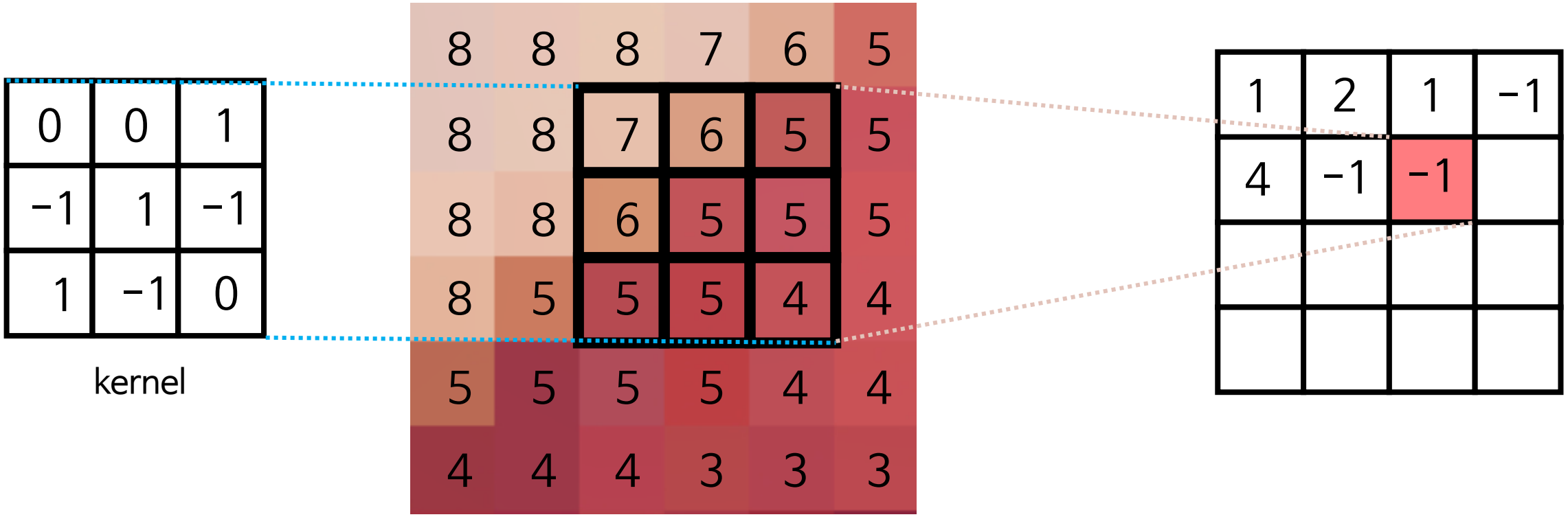
신박AI

같은 방식으로 커널을 이동해가며 합성곱의 값을 구해줍니다



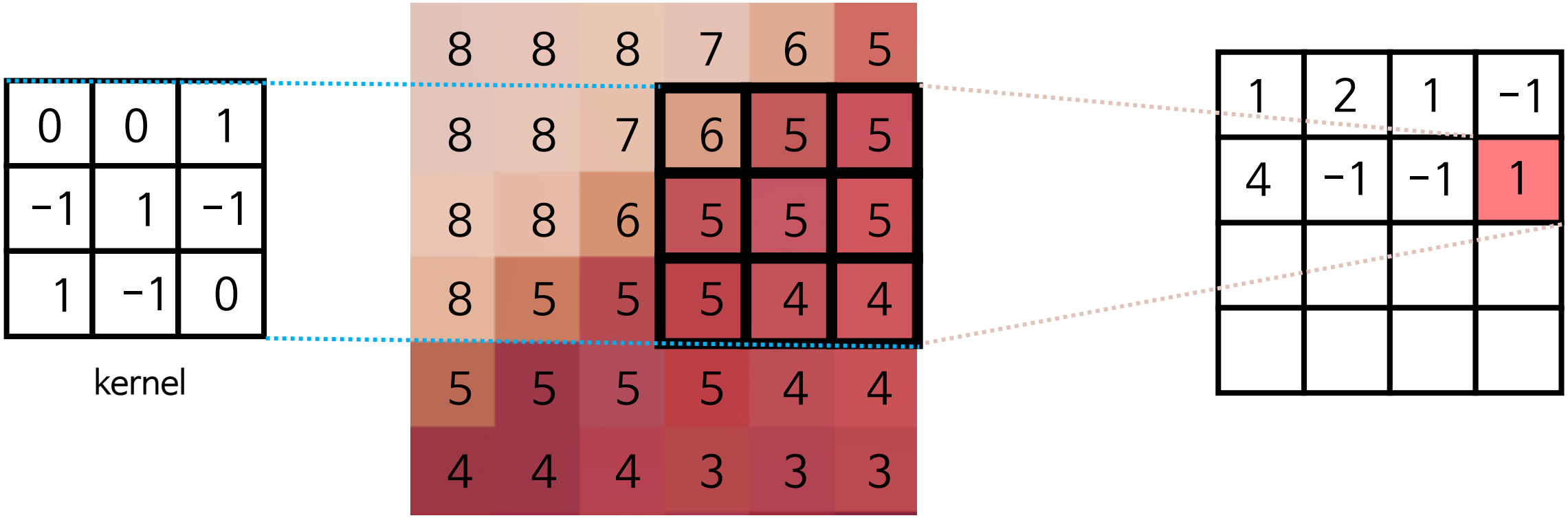
신박AI

같은 방식으로 커널을 이동해가며 합성곱의 값을 구해줍니다



신박AI

같은 방식으로 커널을 이동해가며 합성곱의 값을 구해줍니다



신박AI

이렇게 계산된 결과를 모아 새로운 특성 맵 feature map을 생성합니다

0	0	1
-1	1	-1
1	-1	0

kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

1	2	1	-1
4	-1	-1	1
-2	0	1	1
0	0	1	-1

feature map

신박AI

이것이 합성곱 연산의 핵심입니다

0	0	1
-1	1	-1
1	-1	0

kernel

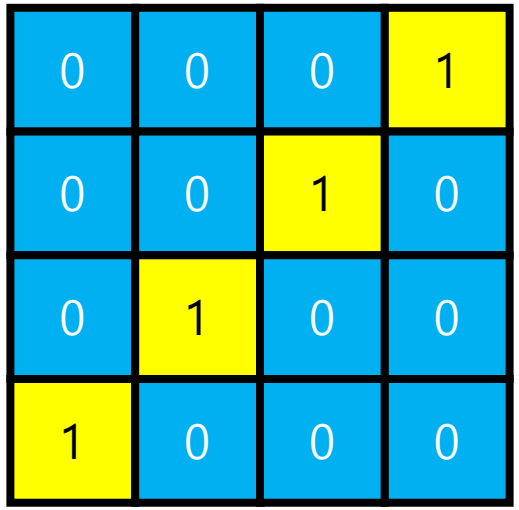
8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

1	2	1	-1
4	-1	-1	1
-2	0	1	1
0	0	1	-1

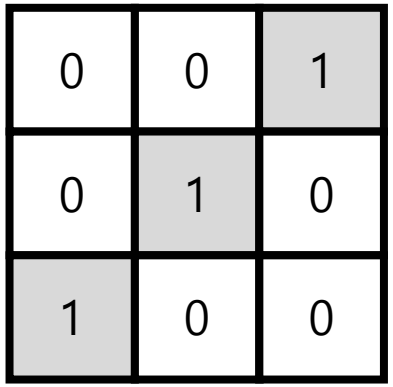
feature map

그러면 합성곱 연산이 갖는 의미는 무엇일까요?

예를들어 다음과 같은 이미지와 커널이 있다고 가정해봅시다



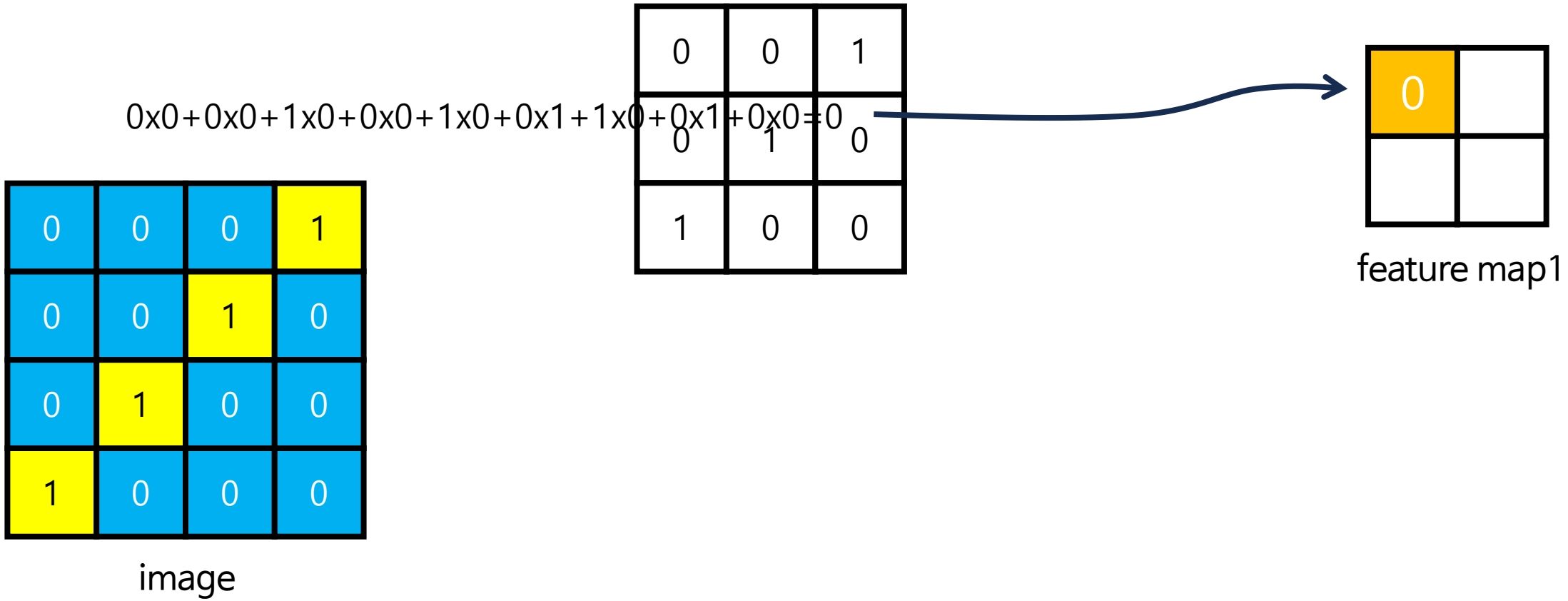
image



kernel1

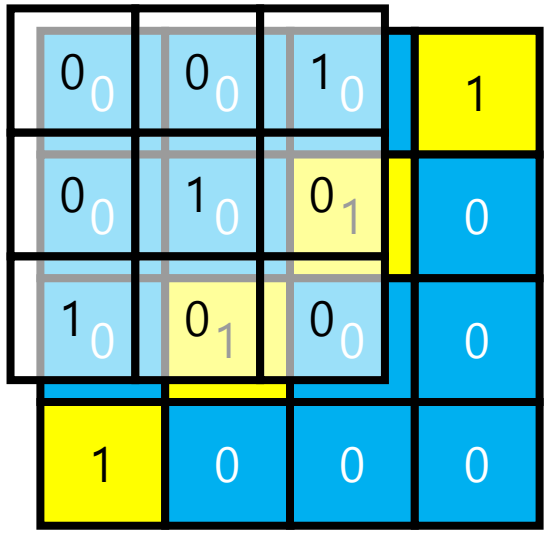
신박AI

그러면 합성곱 연산을 수행하면..

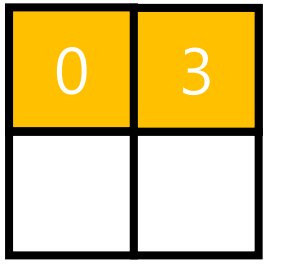


그러면 합성곱 연산을 수행하면..

0x0+0x0+1x1+0x0+1x1+0x0+1x1+0x0+0x0=3



image

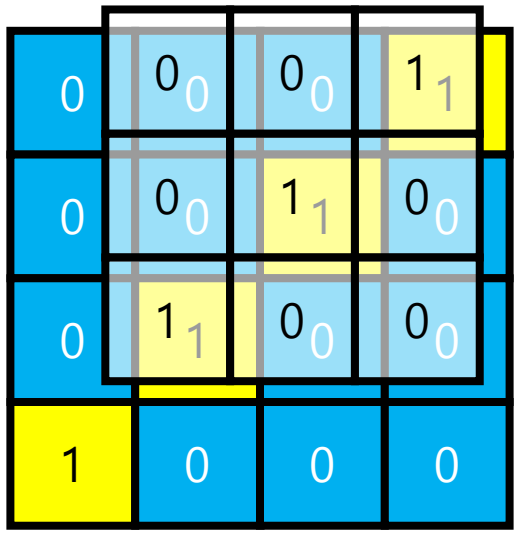


feature map1

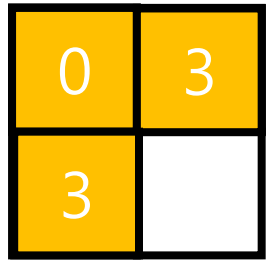
신박AI

그러면 합성곱 연산을 수행하면..

0x0+0x0+1x1+0x0+1x1+0x0+1x1+0x0+0x0=3



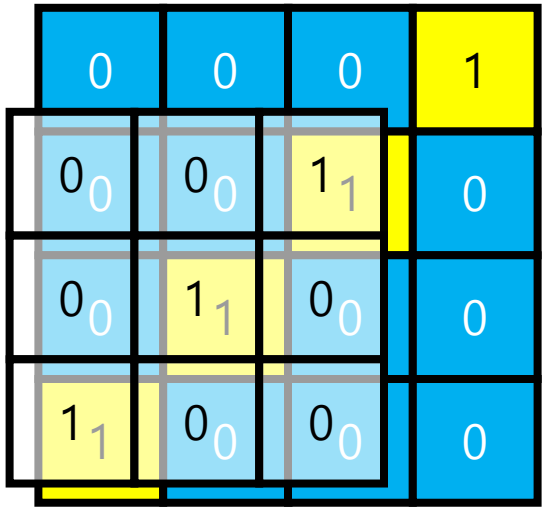
image



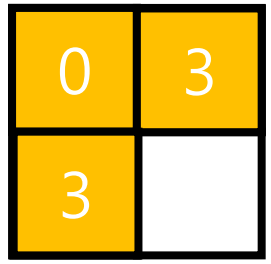
feature map1

신박AI

그러면 합성곱 연산을 수행하면..



image

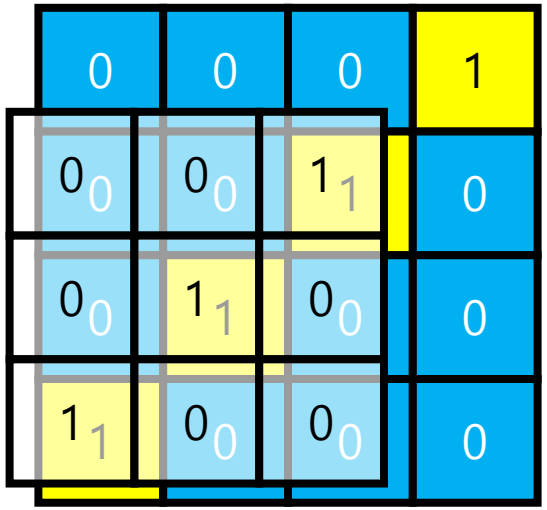


feature map1

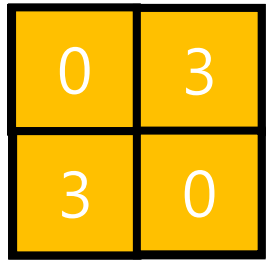
신박AI

그러면 합성곱 연산을 수행하면..

0x0+0x0+1x0+0x0+1x0+0x1+1x0+0x1+0x0=0



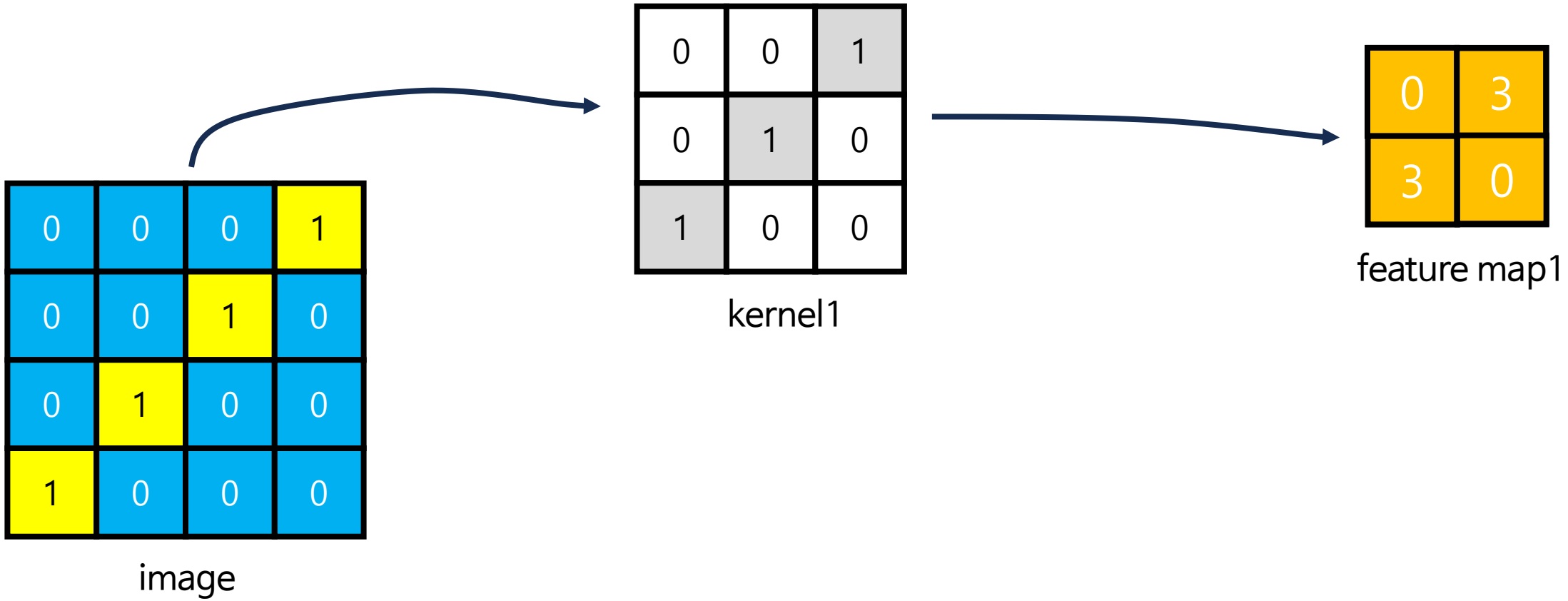
image



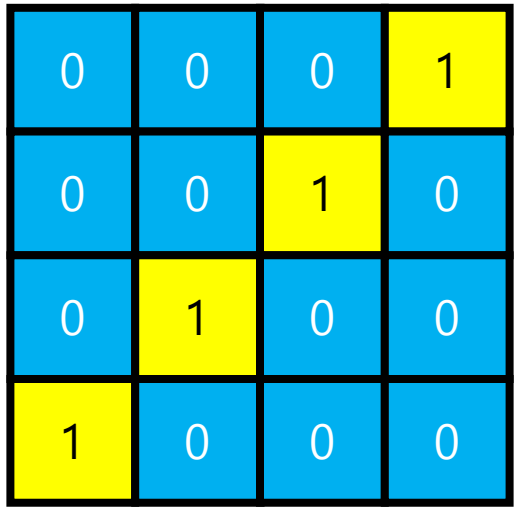
feature map1

신박AI

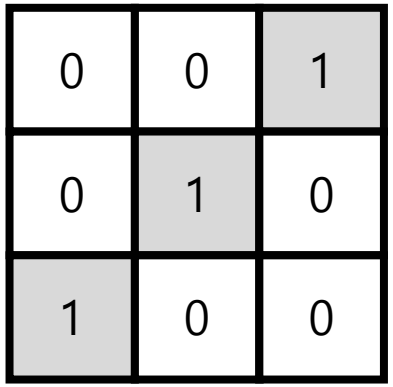
kernel1이 만드는 feature map1은 다음과 같이 생성됩니다



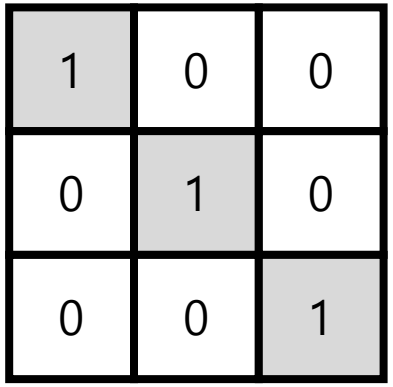
그러면 이런 kernel2는 어떨까요?



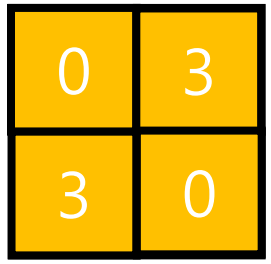
image



kernel1



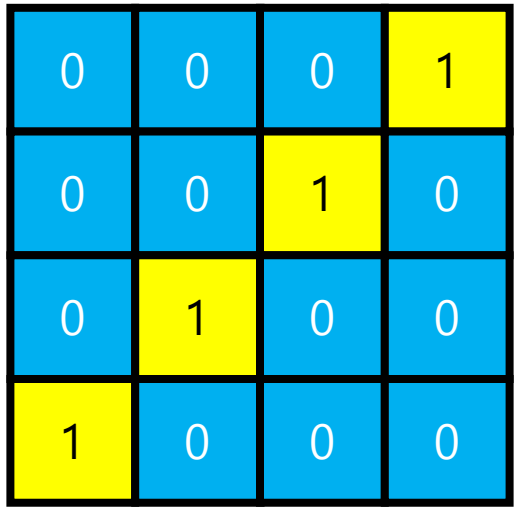
kernel2



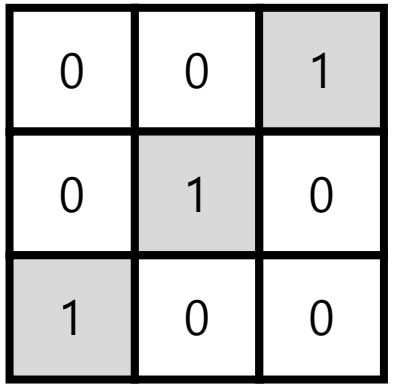
feature map1

신박AI

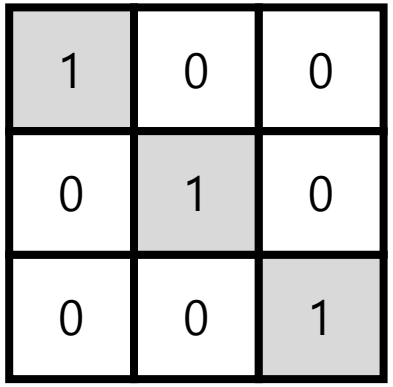
그러면 이런 kernel2는 어떨까요?



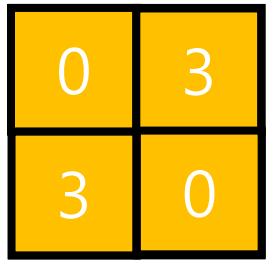
image



kernel1



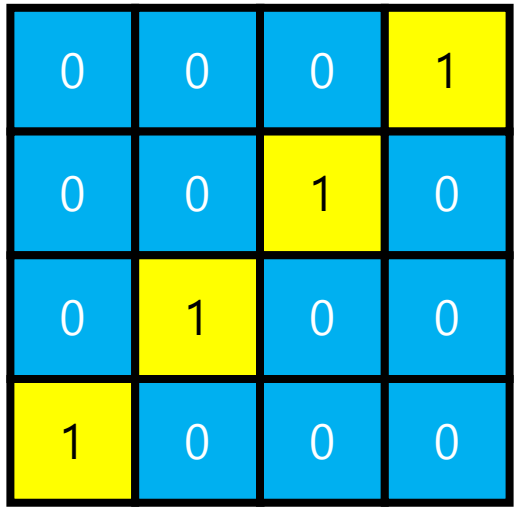
kernel2



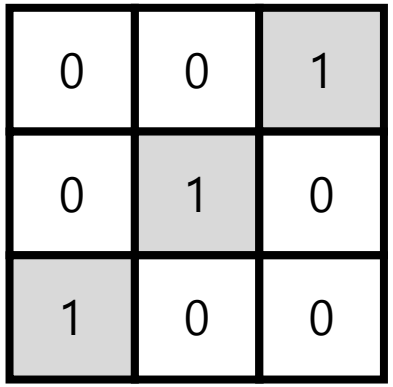
feature map1

신박AI

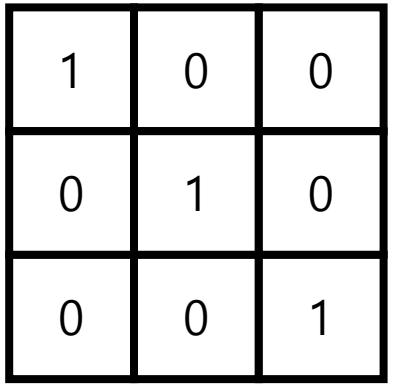
똑같이 합성곱 연산을 수행해보겠습니다

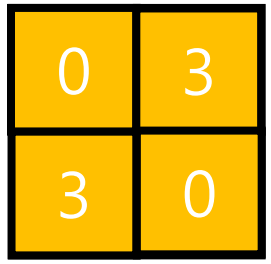


image

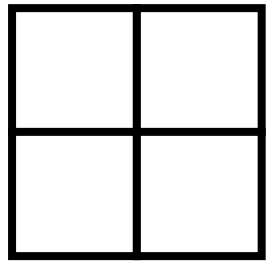


kernel1





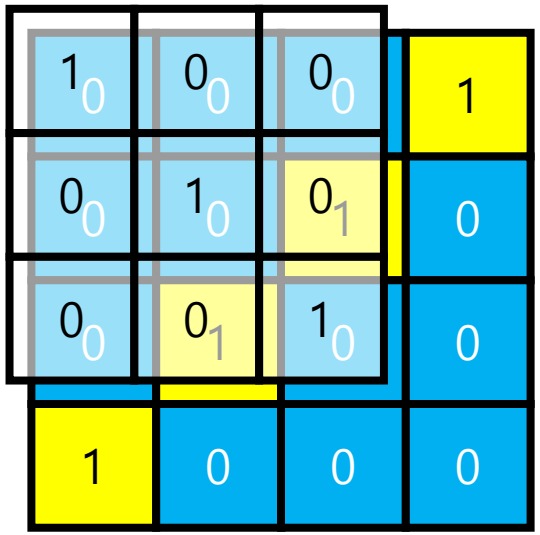
feature map1



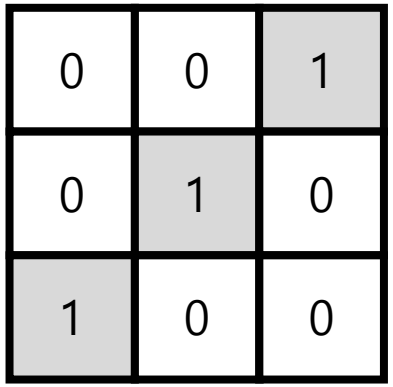
feature map2

신박AI

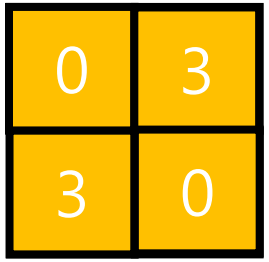
똑같이 합성곱 연산을 수행해보겠습니다



image

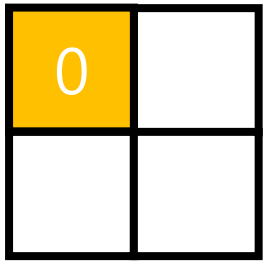


kernel1



feature map1

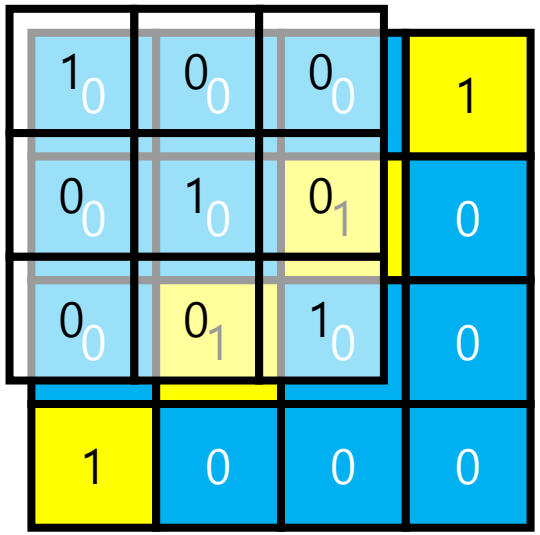
1x0+0x0+0x0+0x0+1x0+0x1+0x0+0x1+1x0=0



feature map2

신박AI

똑같이 합성곱 연산을 수행해보겠습니다



image

0	0	1
0	1	0
1	0	0

kernel1

0	3
3	0

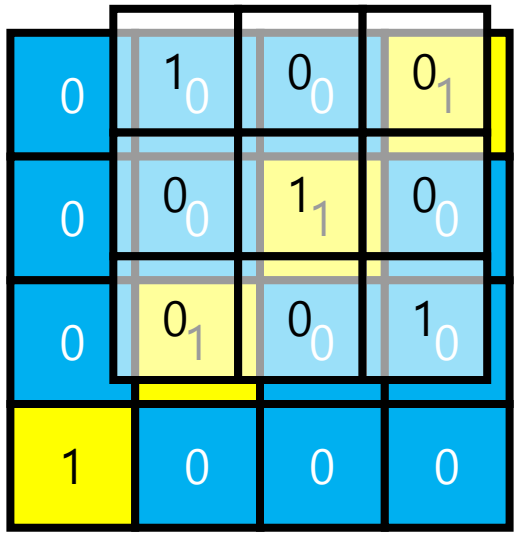
feature map1

0	

feature map2

신박AI

똑같이 합성곱 연산을 수행해보겠습니다



image

0	0	1
0	1	0
1	0	0

kernel1

0	3
3	0

feature map1

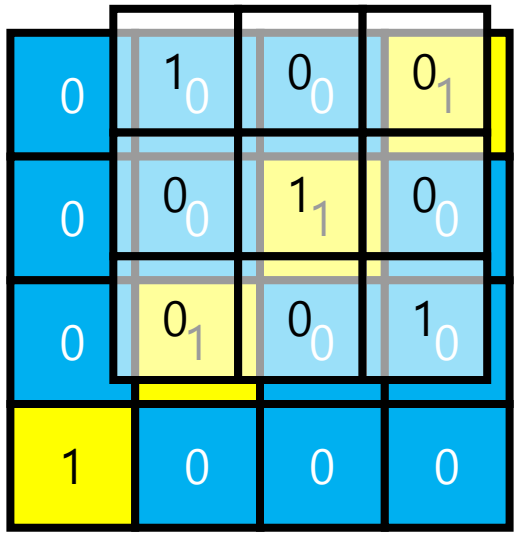
1x0+0x0+0x0+0x0+1x1+0x0+0x1+0x0+1x0=0

0	1

feature map2

신박AI

똑같이 합성곱 연산을 수행해보겠습니다



image

0	0	1
0	1	0
1	0	0

kernel1

0	3
3	0

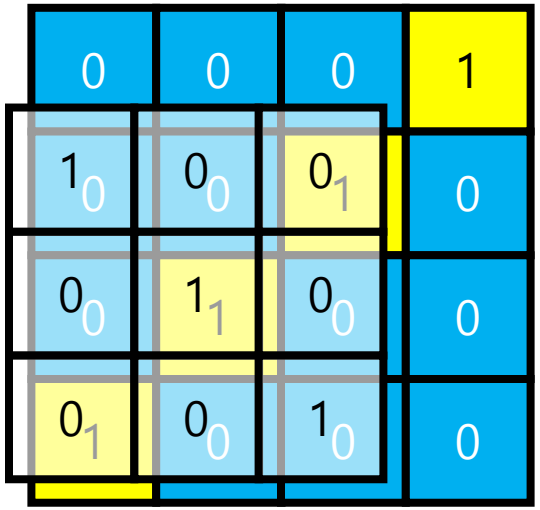
feature map1

0	1

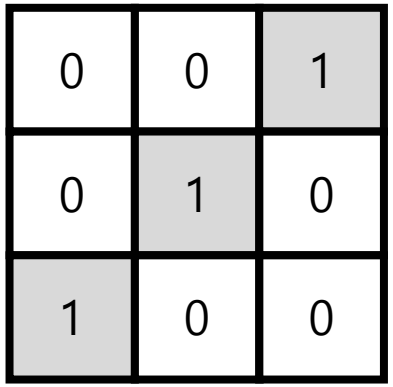
feature map2

신박AI

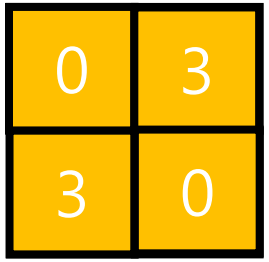
똑같이 합성곱 연산을 수행해보겠습니다



image

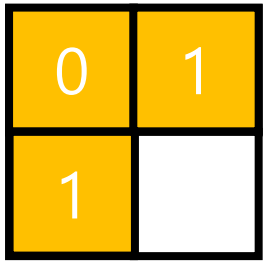


kernel1



feature map1

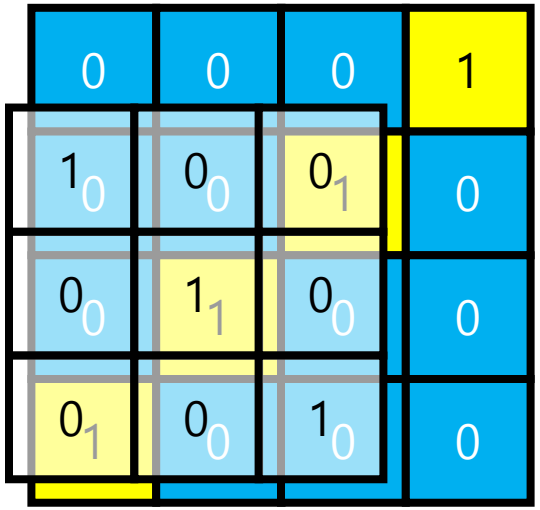
1x0+0x0+0x1+0x0+1x1+0x0+0x1+0x0+1x0=0

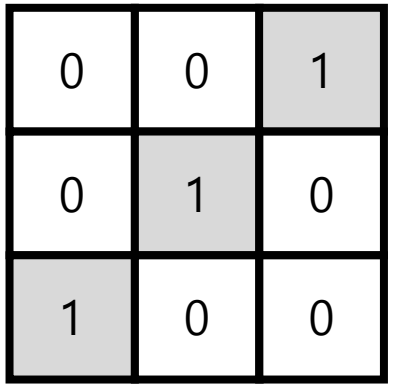


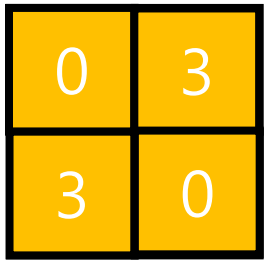
feature map2

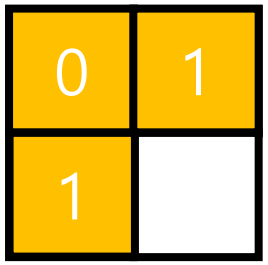
신박AI

똑같이 합성곱 연산을 수행해보겠습니다



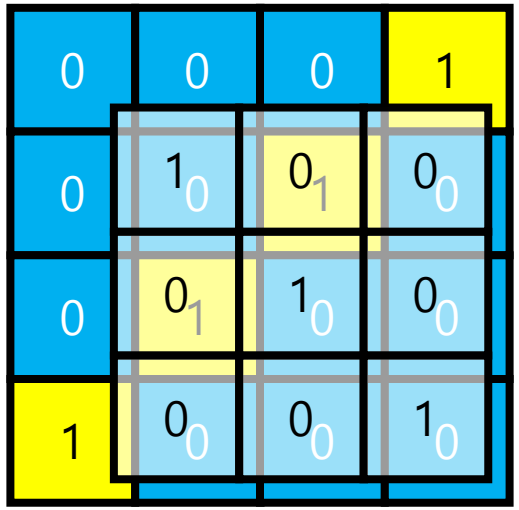




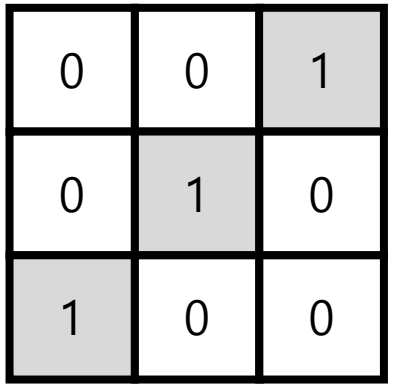


신박AI

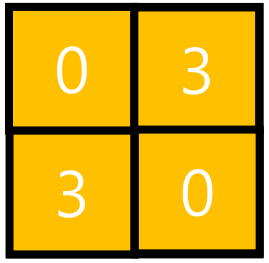
똑같이 합성곱 연산을 수행해보겠습니다



image

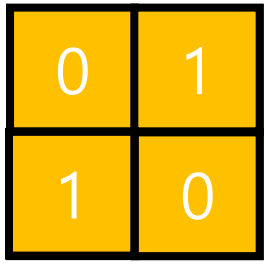


kernel1



feature map1

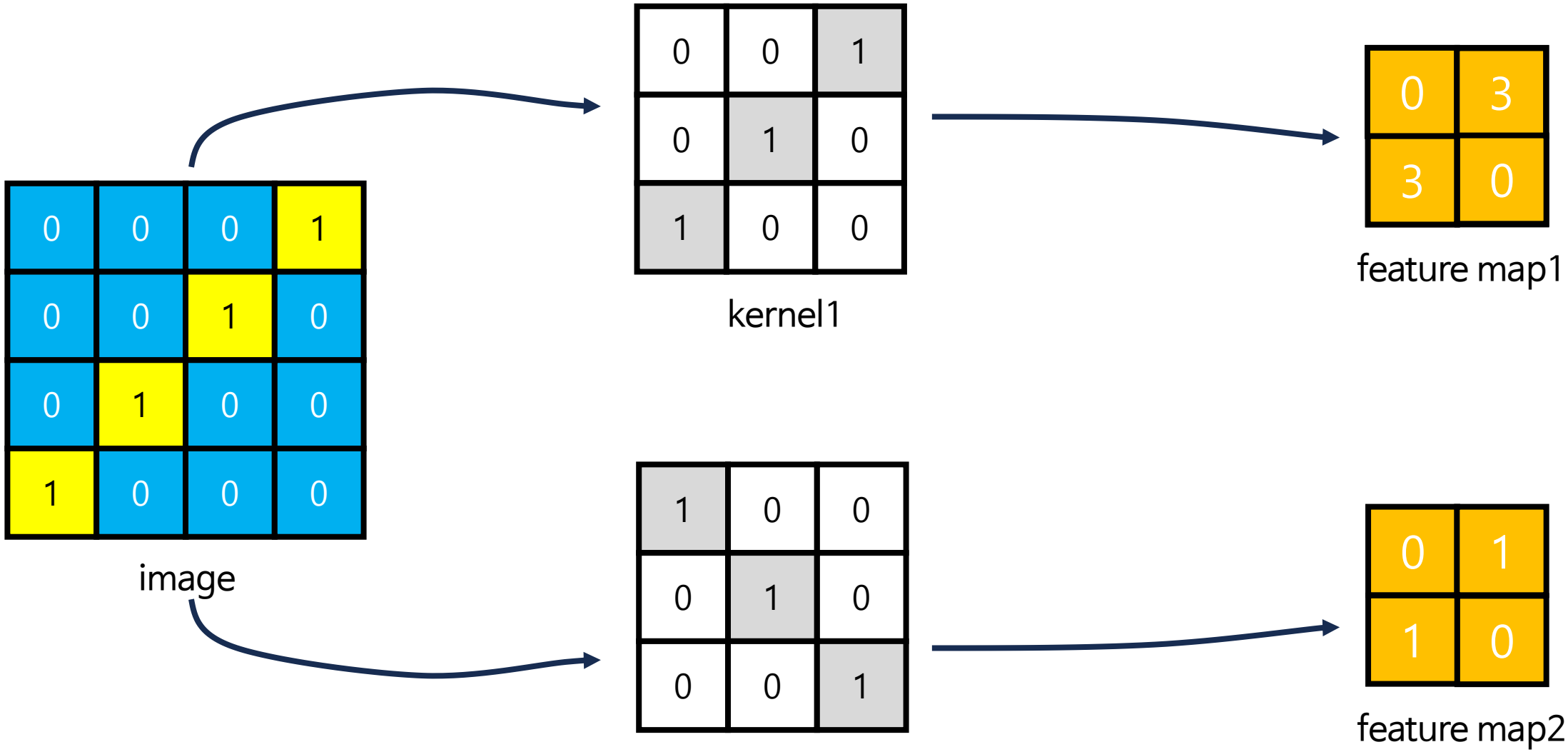
1x0+0x1+0x0+0x1+1x0+0x0+0x0+0x0+1x0=0



feature map2

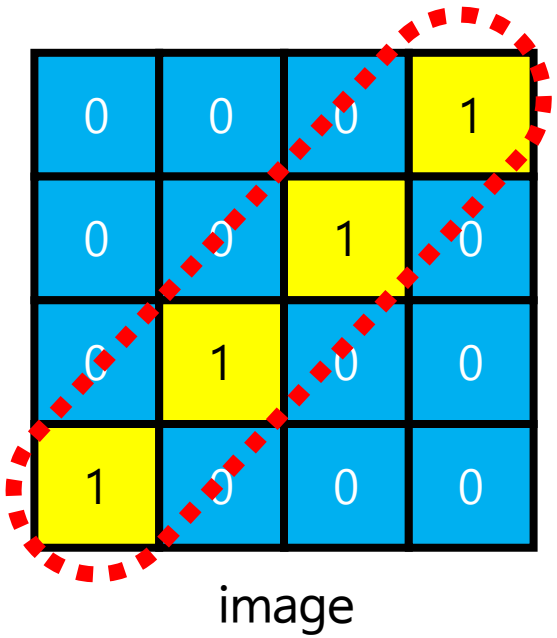
신박AI

각각의 kernel이 만드는 feature map은 다음과 같이 생성됩니다

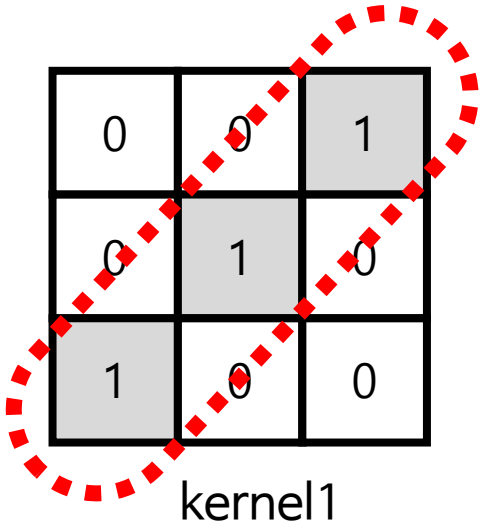


신박AI

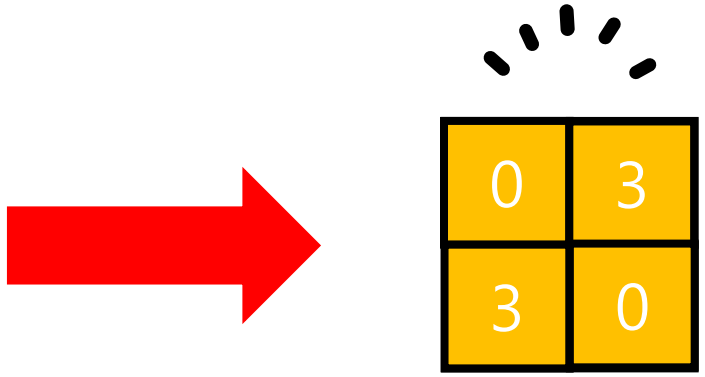
보시다시피, 이미지의 특성과 유사한 커널은 출력값이 높고,



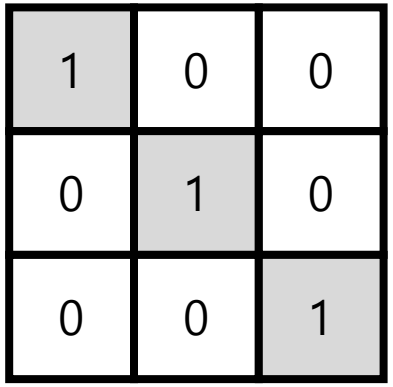
image

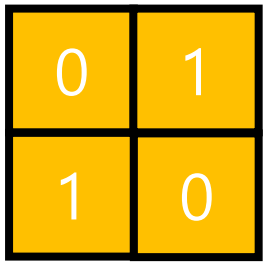


kernel1



feature map1

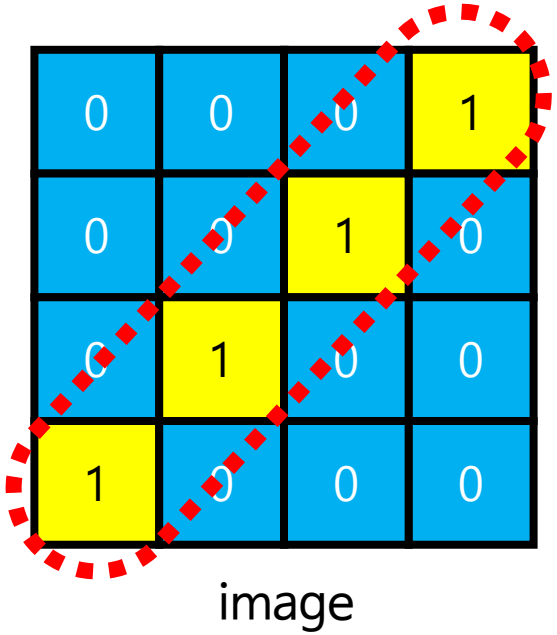




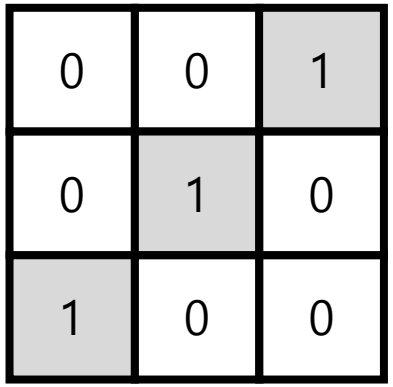
feature map2

신박AI

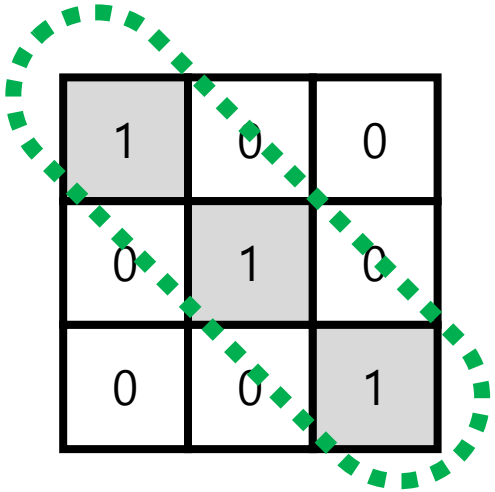
이미지의 특성을 반영하지 않는 커널은 출력값이 낮았습니다

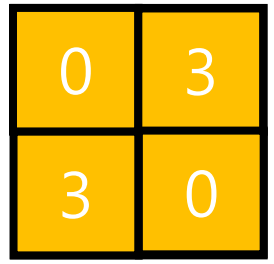


image

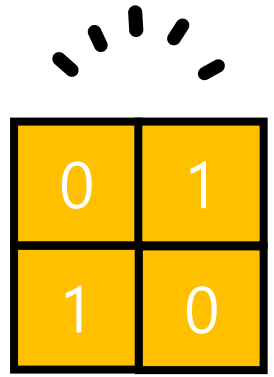


kernel1





feature map1



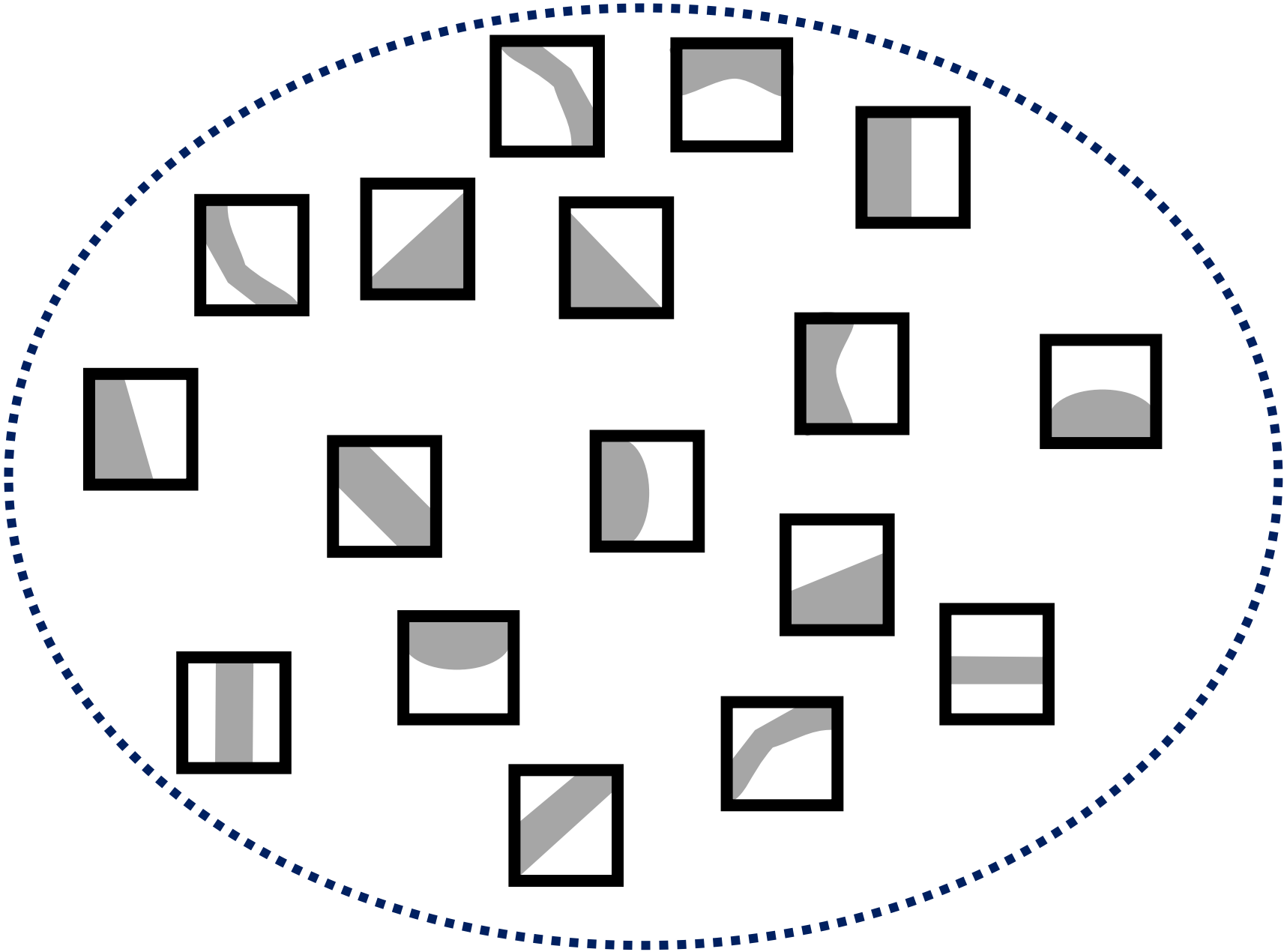
feature map2

신박AI

좀 더 확장하여 생각해본다면..

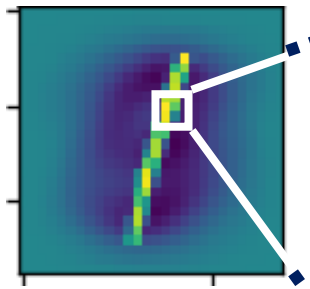
신박AI

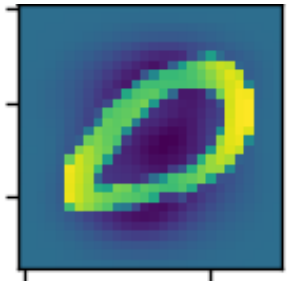
이런 여러 종류의 kernel들이 있다고 가정해볼때

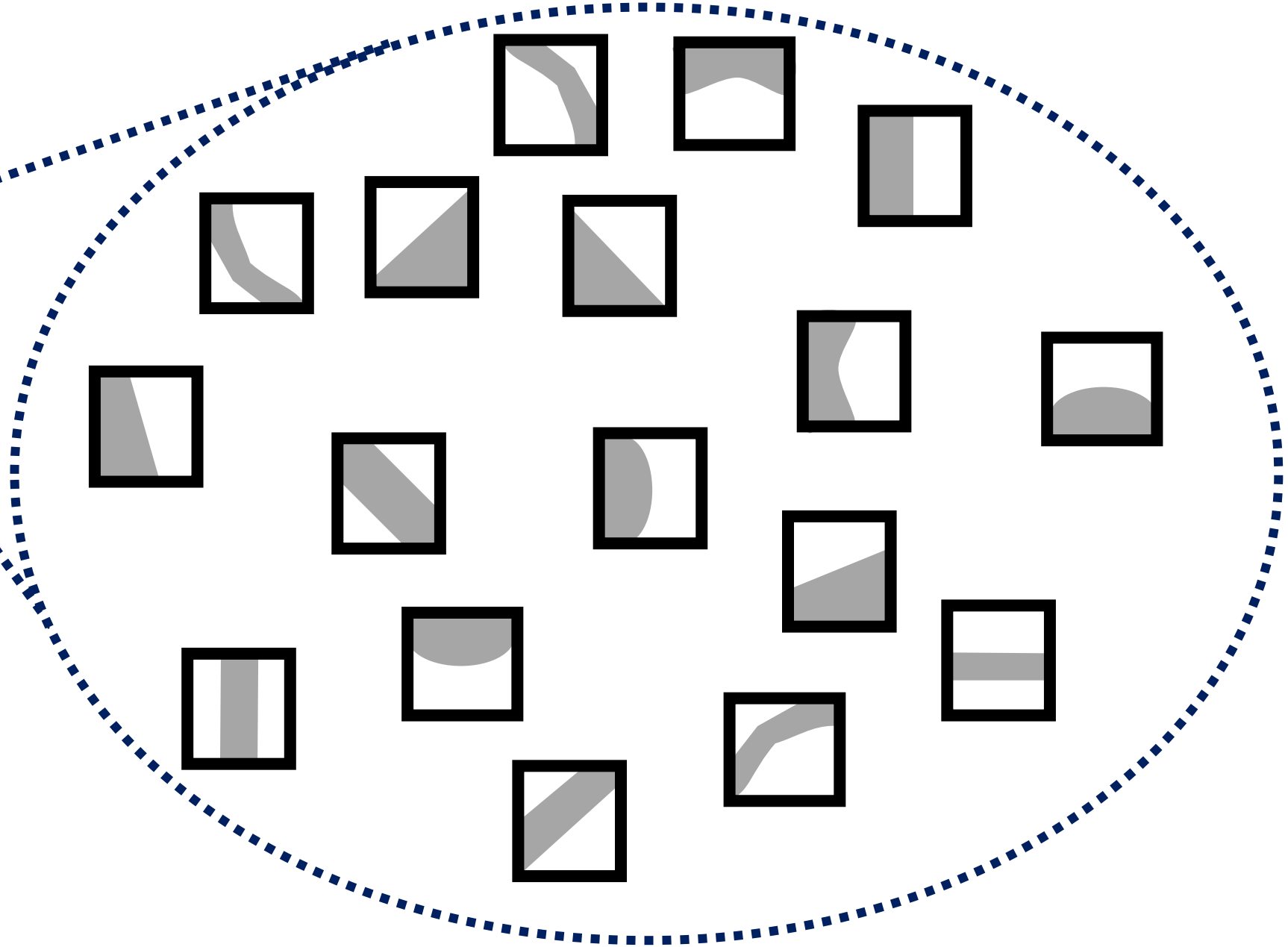


신박AI

MNIST ‘1’이라는 이미지가 입력값이 될 경우

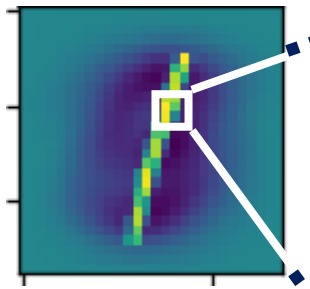


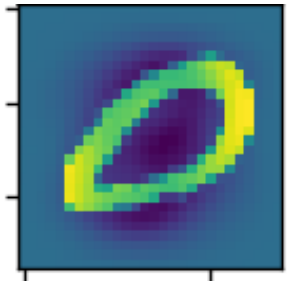


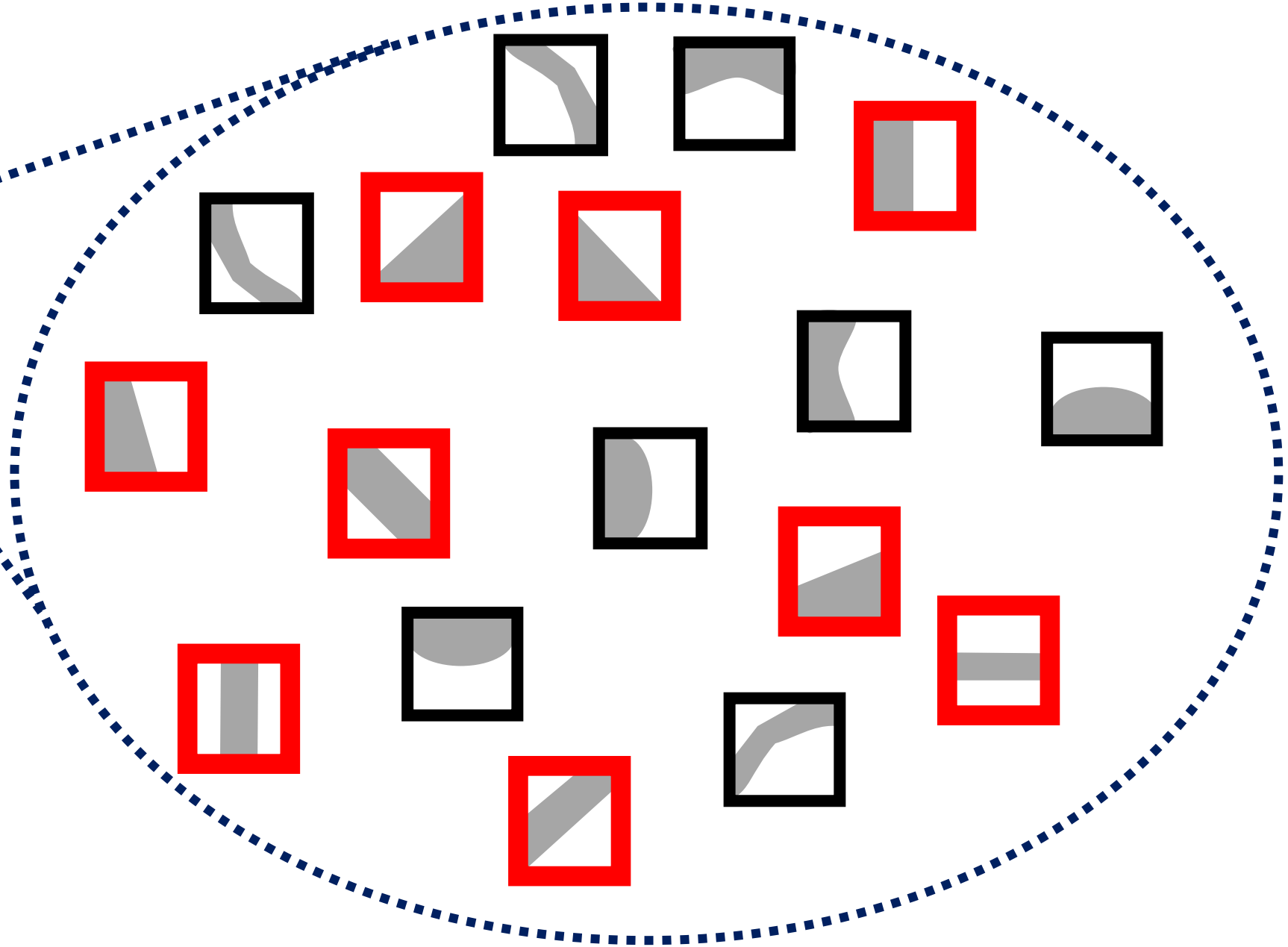


신박AI

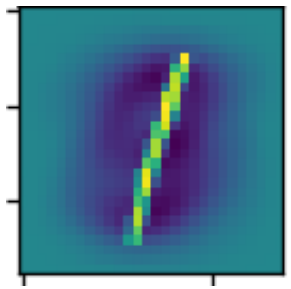
직선 형태의 kernel들의 합성곱 출력값이 높아지고,

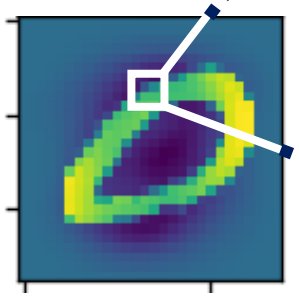


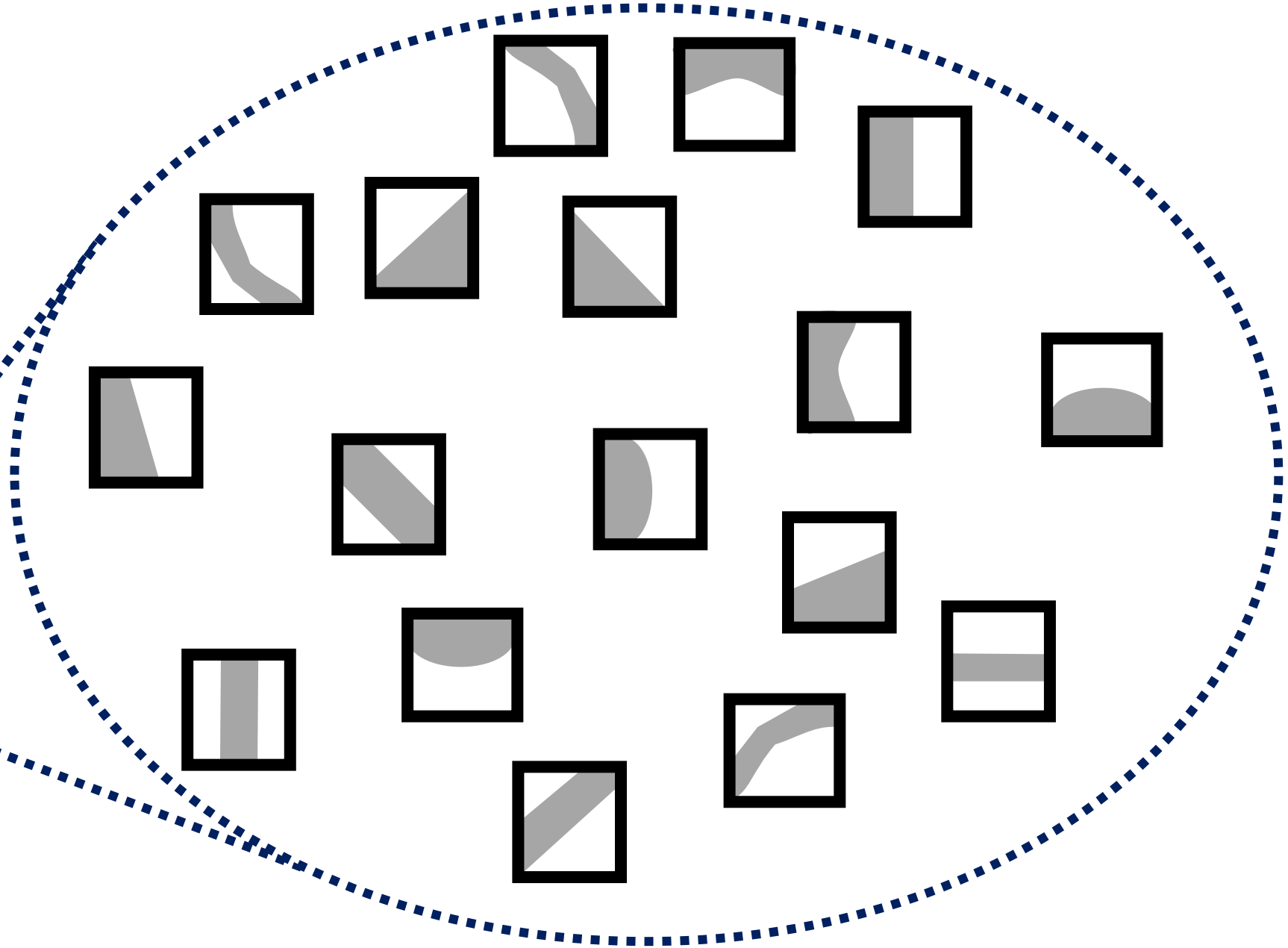




MNIST ‘0’이라는 이미지가 입력값이 될 경우

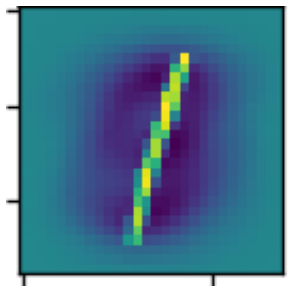


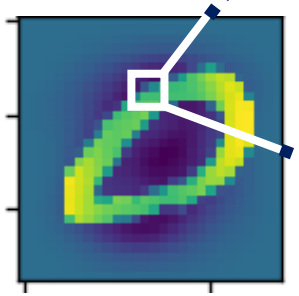


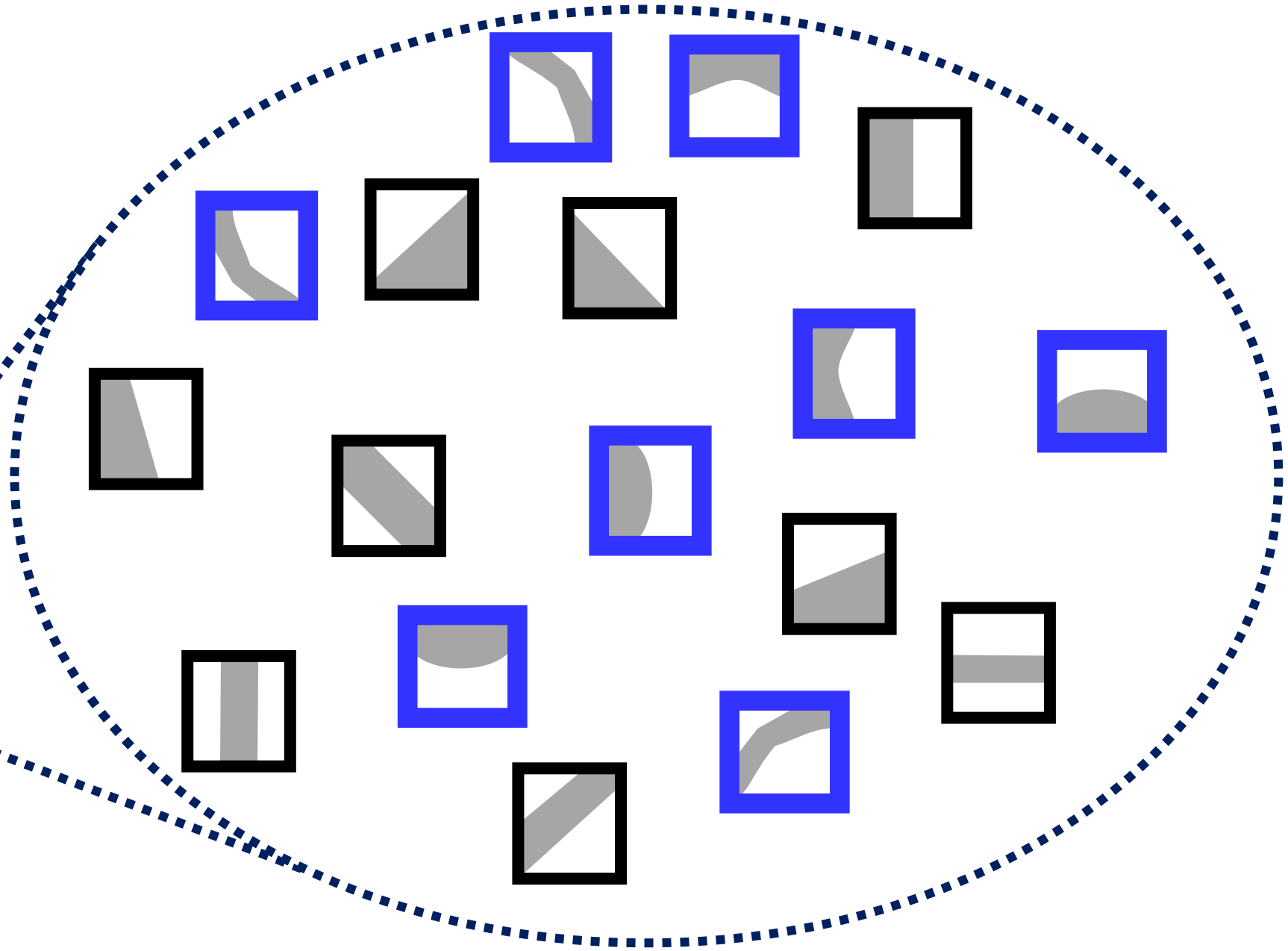


신박AI

곡선 형태의 kernel들의 합성곱 출력값이 높아지게 됩니다

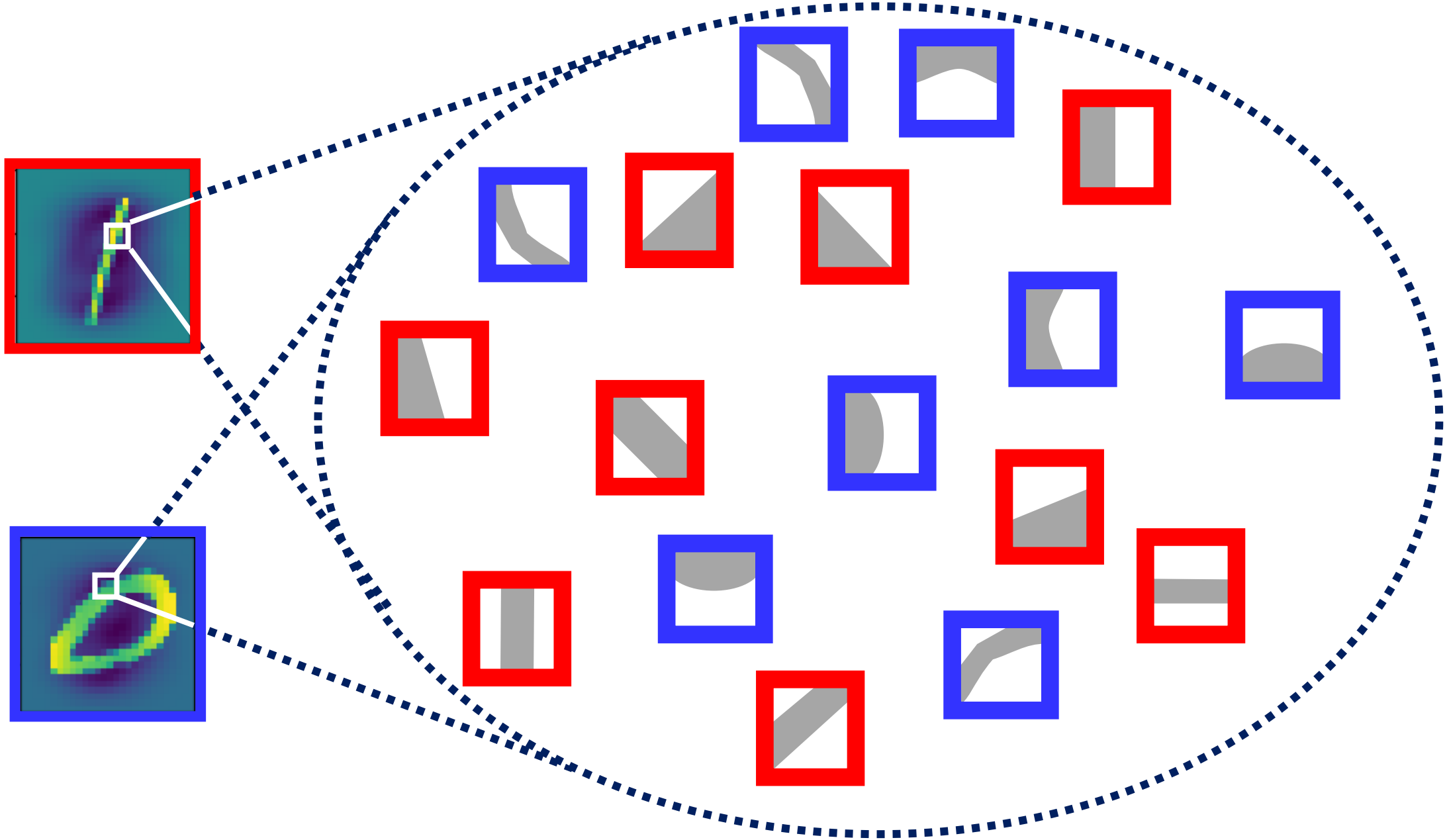






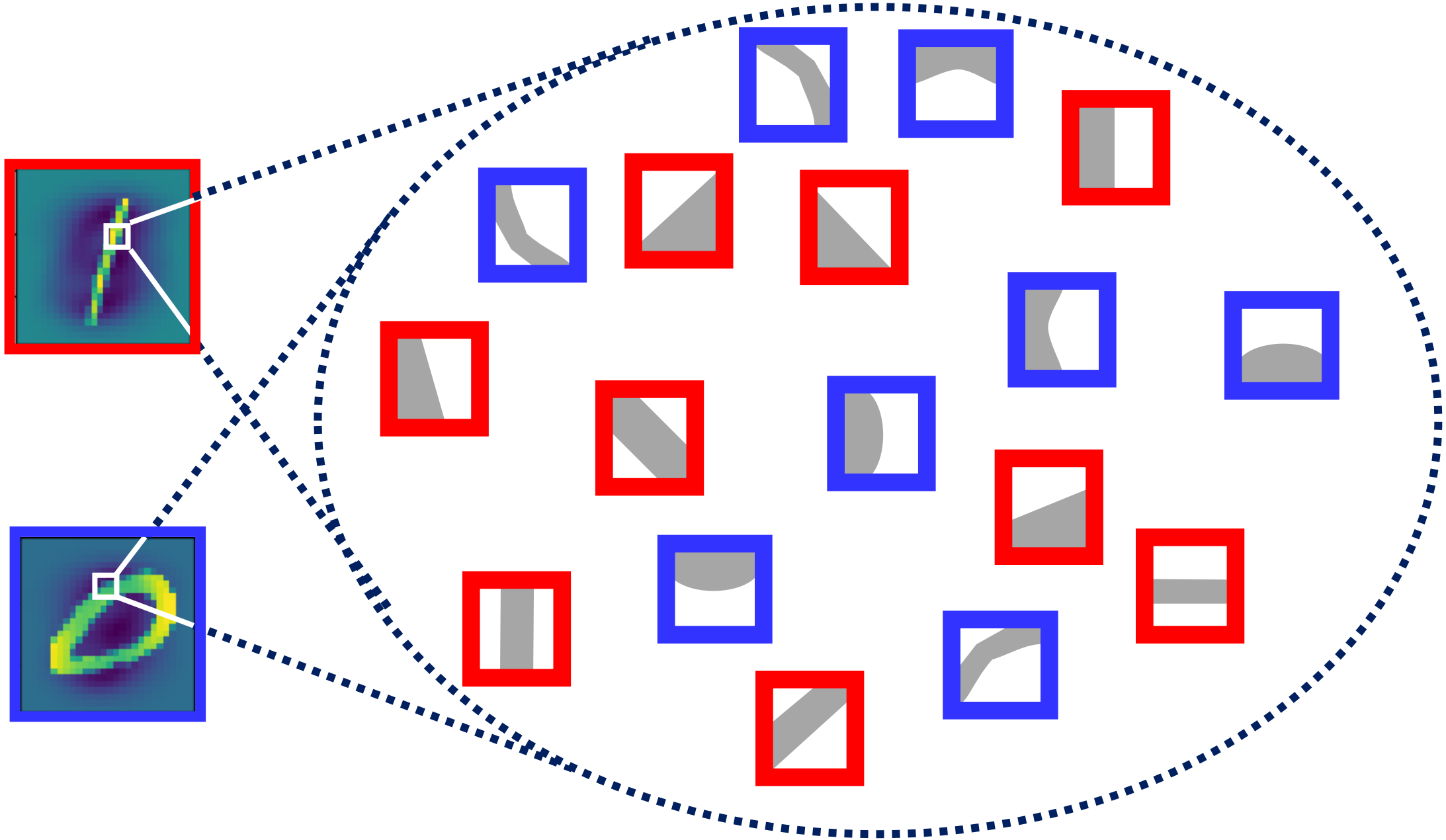
신박AI

합성곱의 출력값을 일종의 확률로 생각해보면,



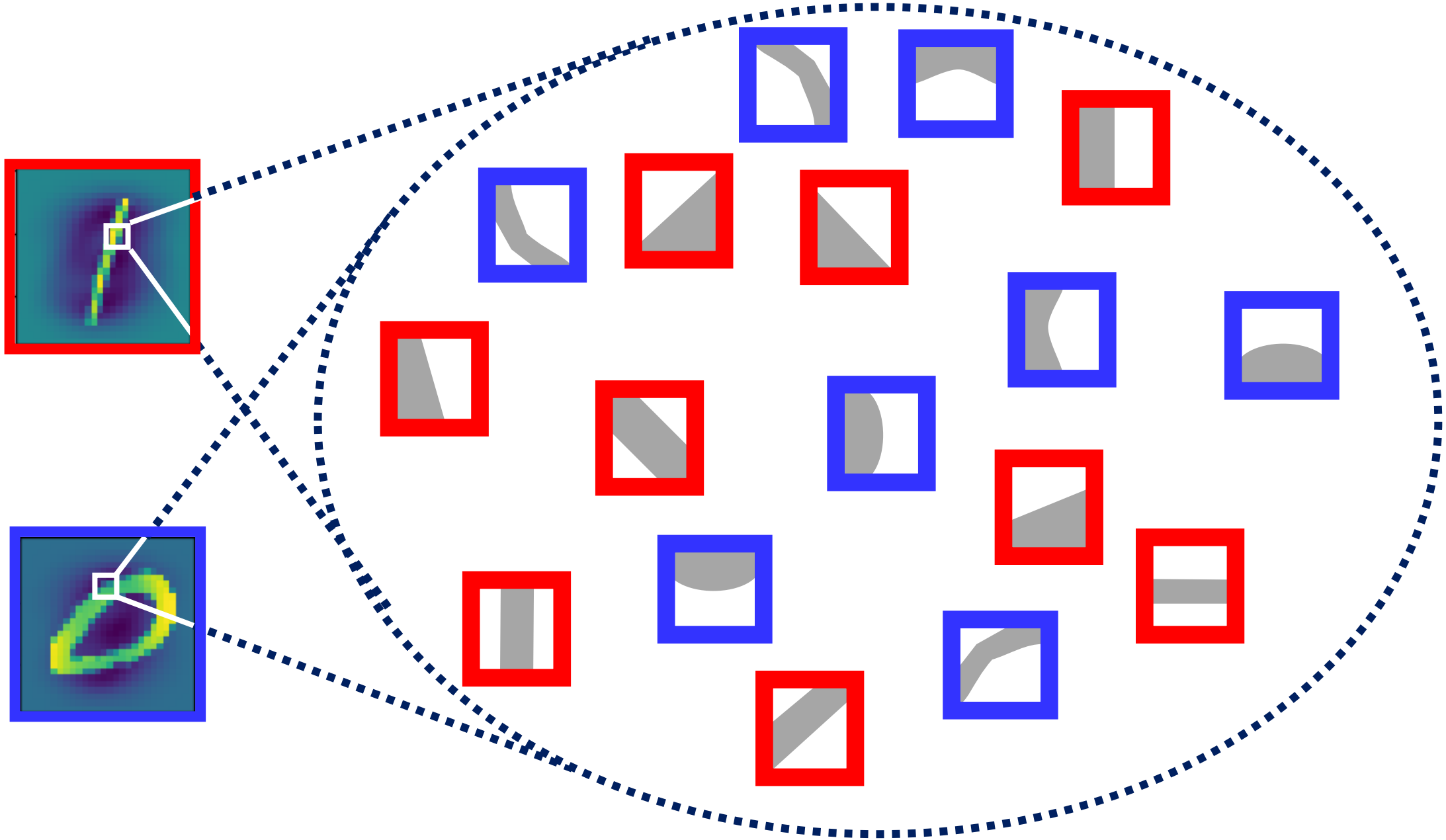
신박AI

합성곱 레이어의 역할은 주어진 이미지의 지역적 특성 (feature)을



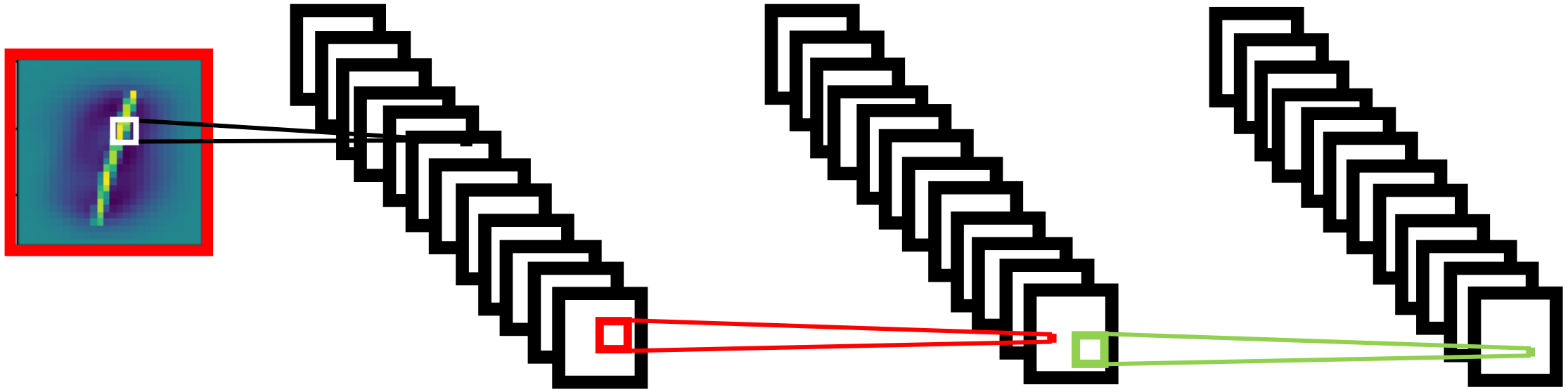
신박AI

분석하여 확률로 리턴하는 역할을 수행한다고 보시면 되겠습니다

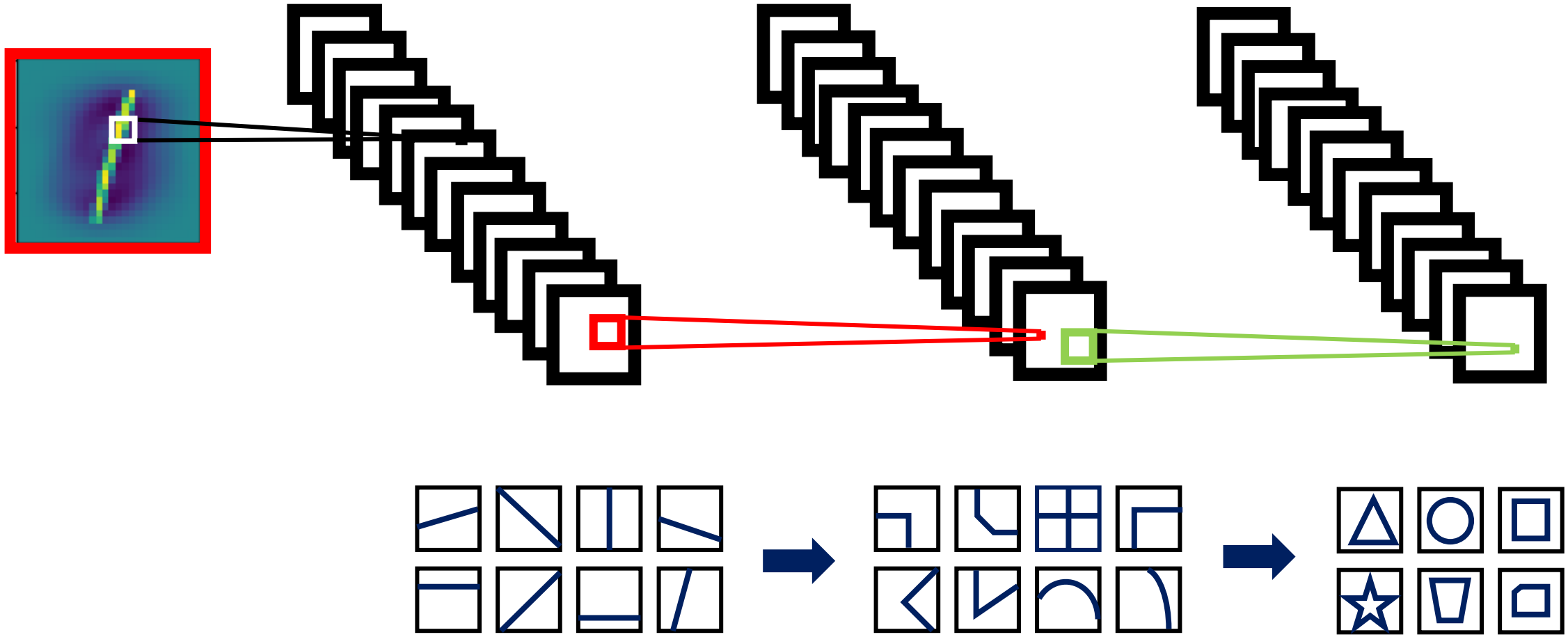


신박AI

뿐만 아니라, 이렇게 합성곱 층이 깊어질 수록,

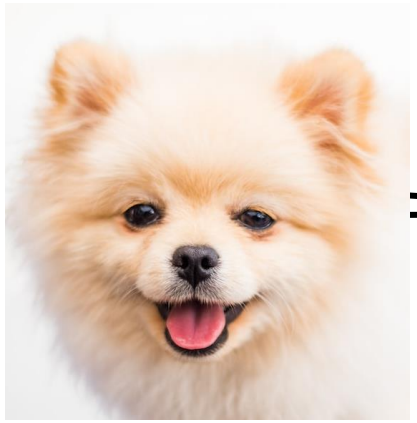


복잡한 형태의 local feature도 처리할 수 있게 됩니다



신박AI

그래서 개와 고양이같은 복잡한 이미지도 합성곱층이 깊어지면 이미지를 분류해 낼 수가 있게 되는 것입니다



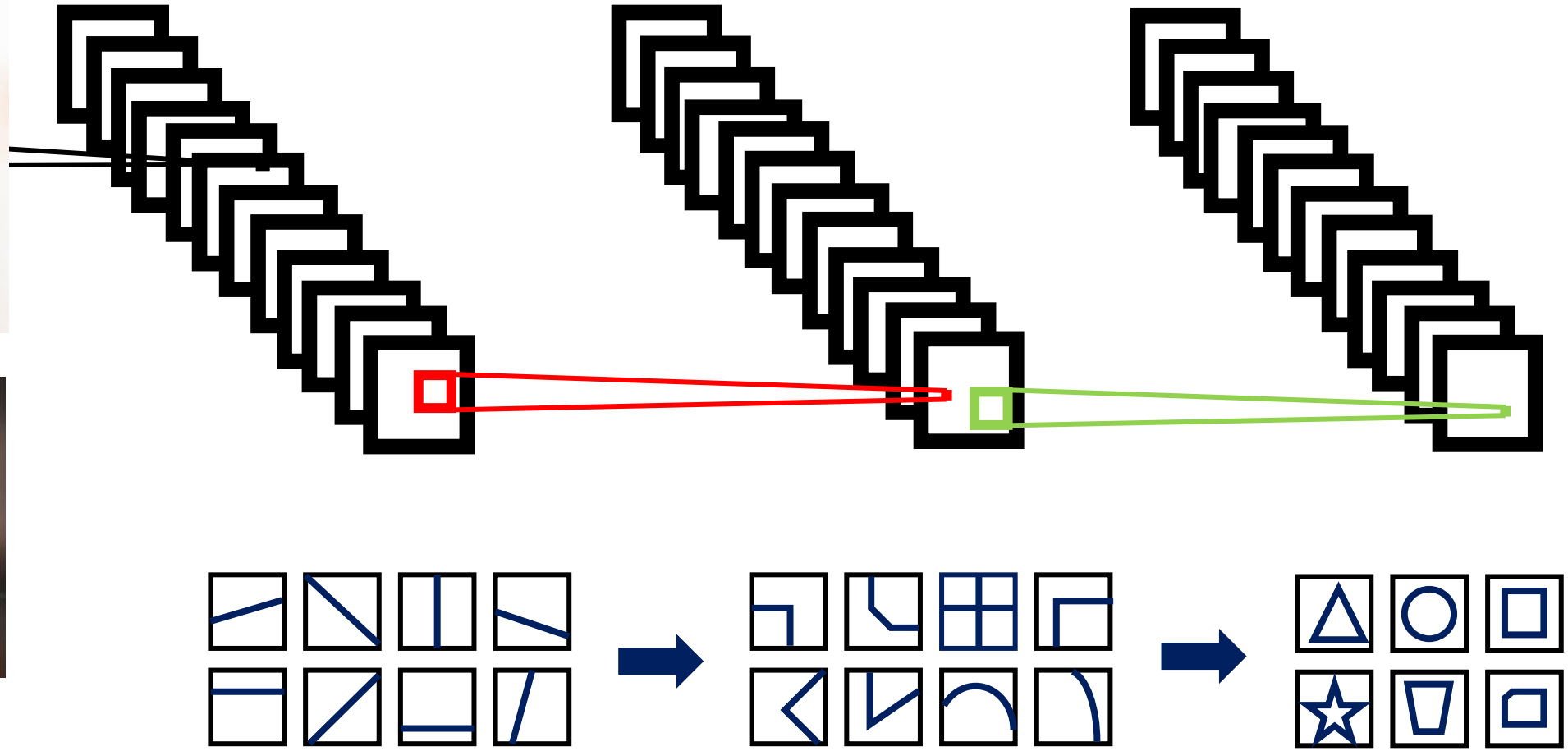


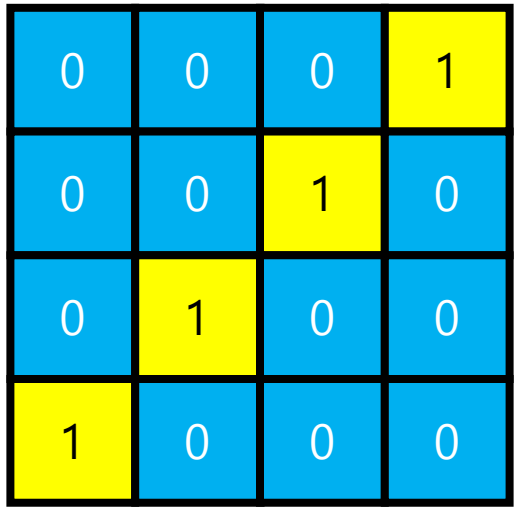
Photo by Onur Binay on Unsplash

Photo by Denys Sergushkin on Unsplash

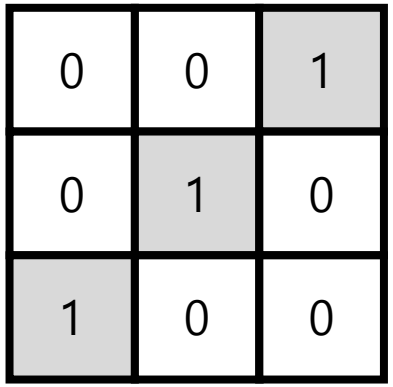
신박AI

또한, kernel의 크기와 보폭(stride)은 합성곱 층의 출력층의 크기를 조절하는 중요한 요소입니다

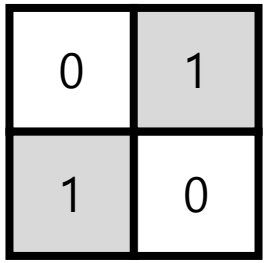
이렇게 크기가 서로 다른 두개의 커널은,



image



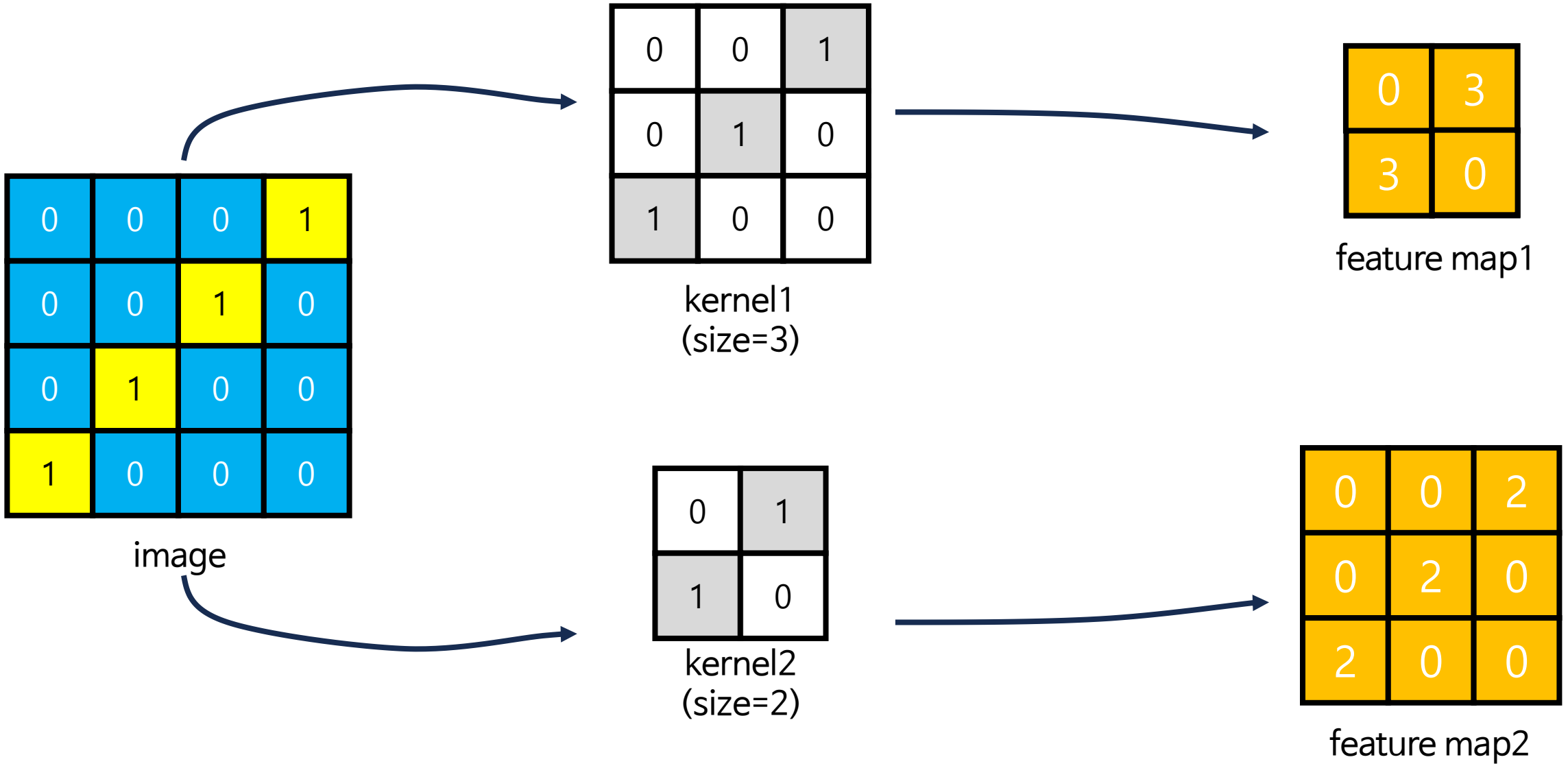
kernel1
(size=3)



kernel2
(size=2)

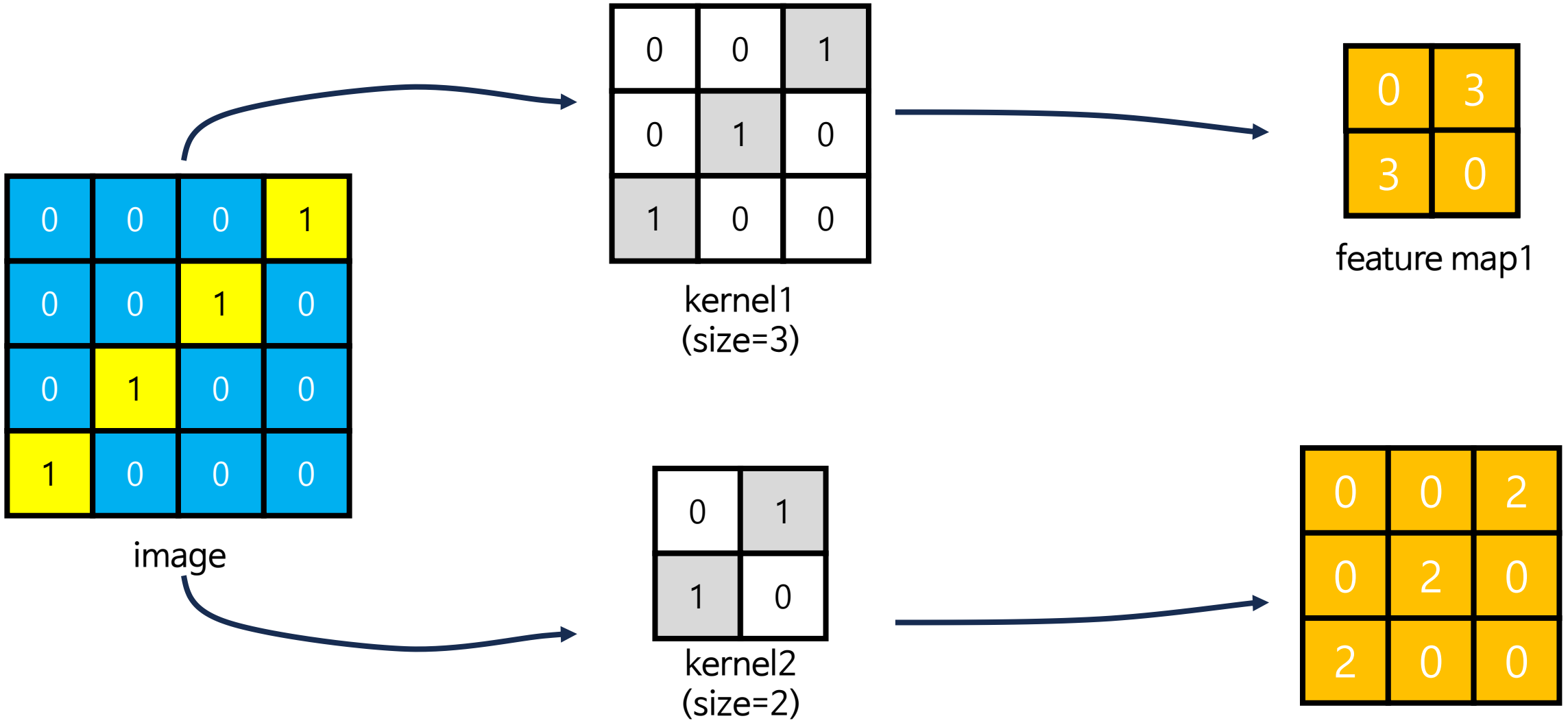
신박AI

이렇게 크기가 서로 다른 두개의 커널은,



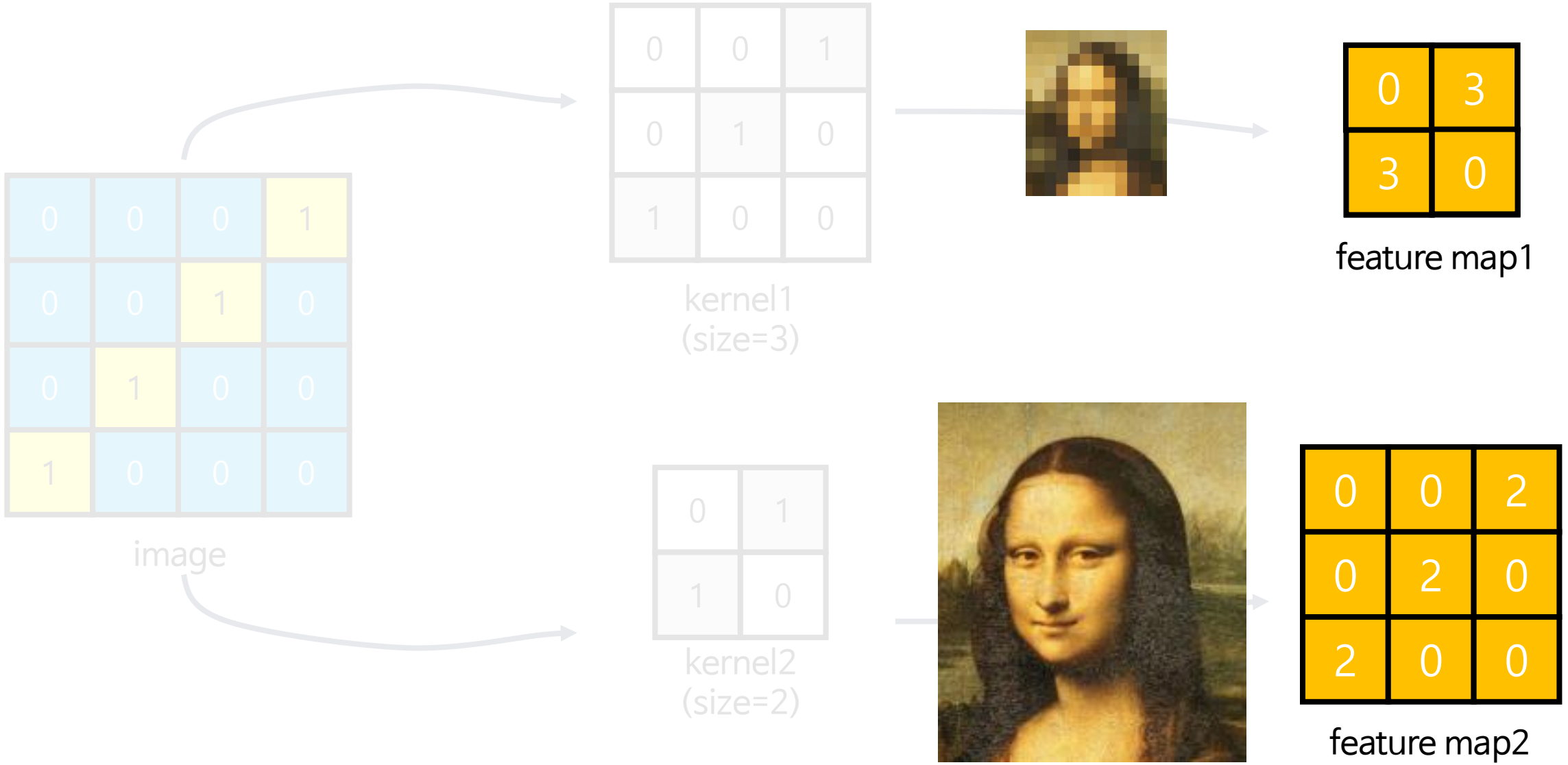
신박AI

서로 크기가 다른 feature map을 생성하게 됩니다



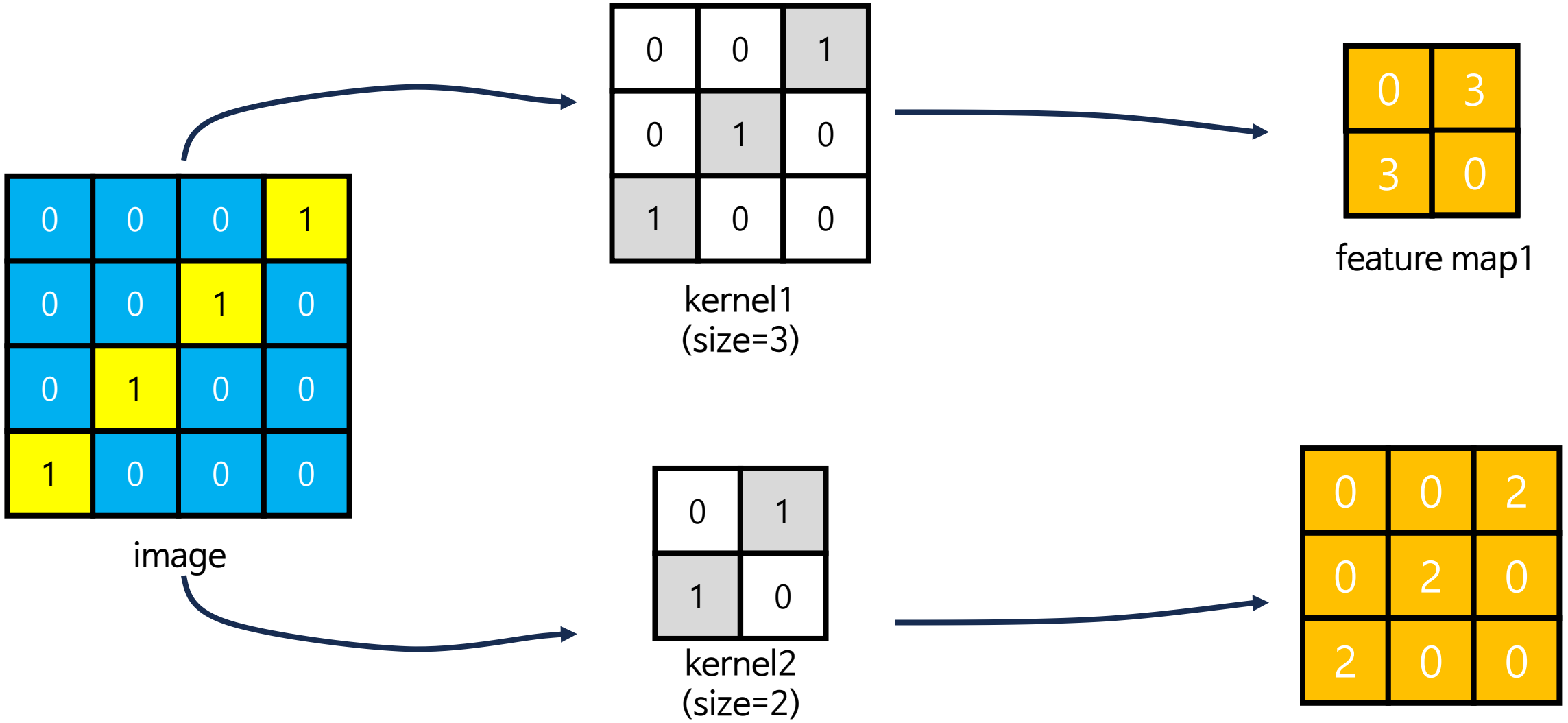
신박AI

이미지의 픽셀수가 많을 수록 디테일한 이미지가 되듯,



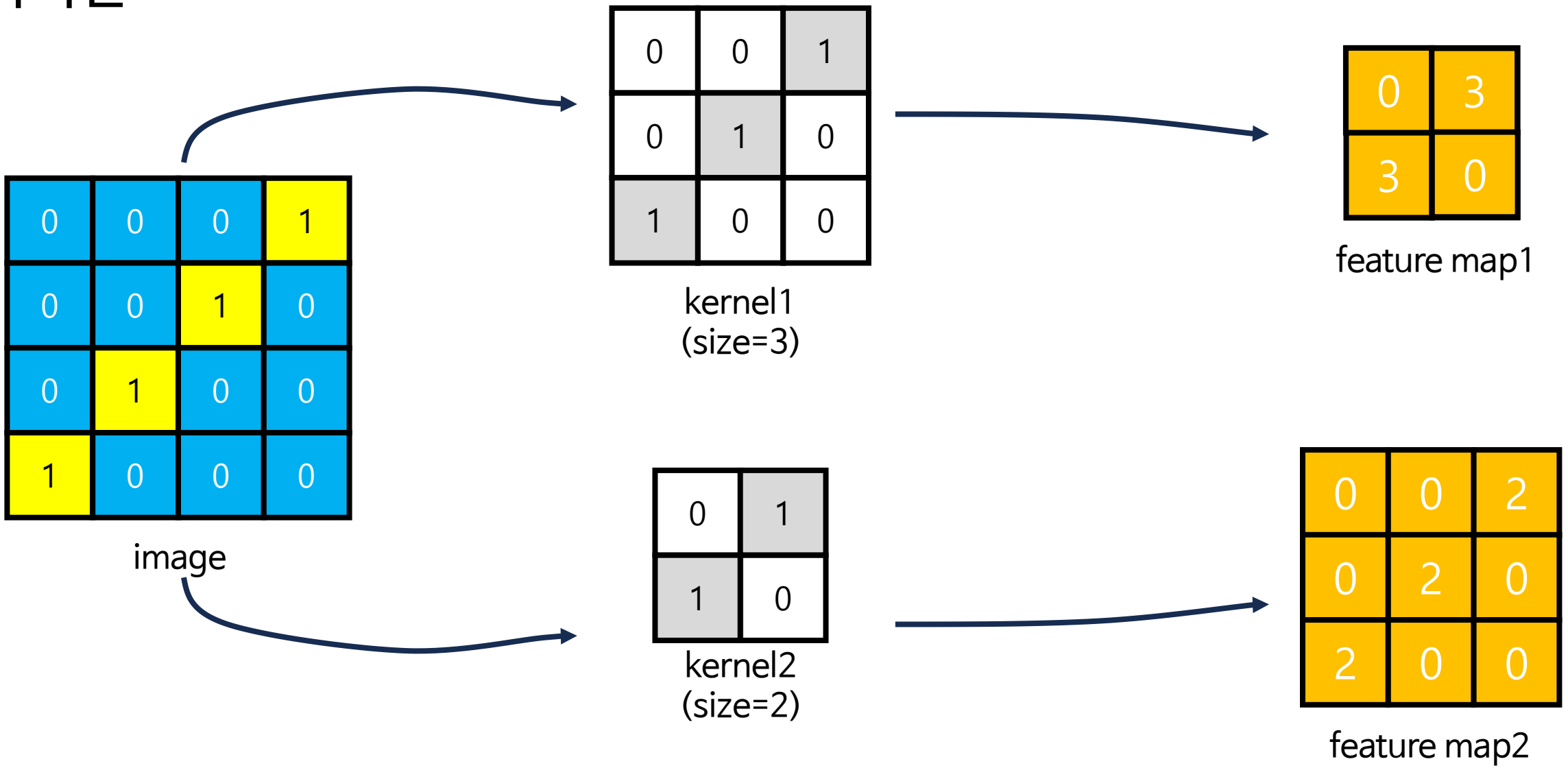
신박AI

feature map의 크기가 클 수록 feature의 디테일을 표현할 수 있습니다



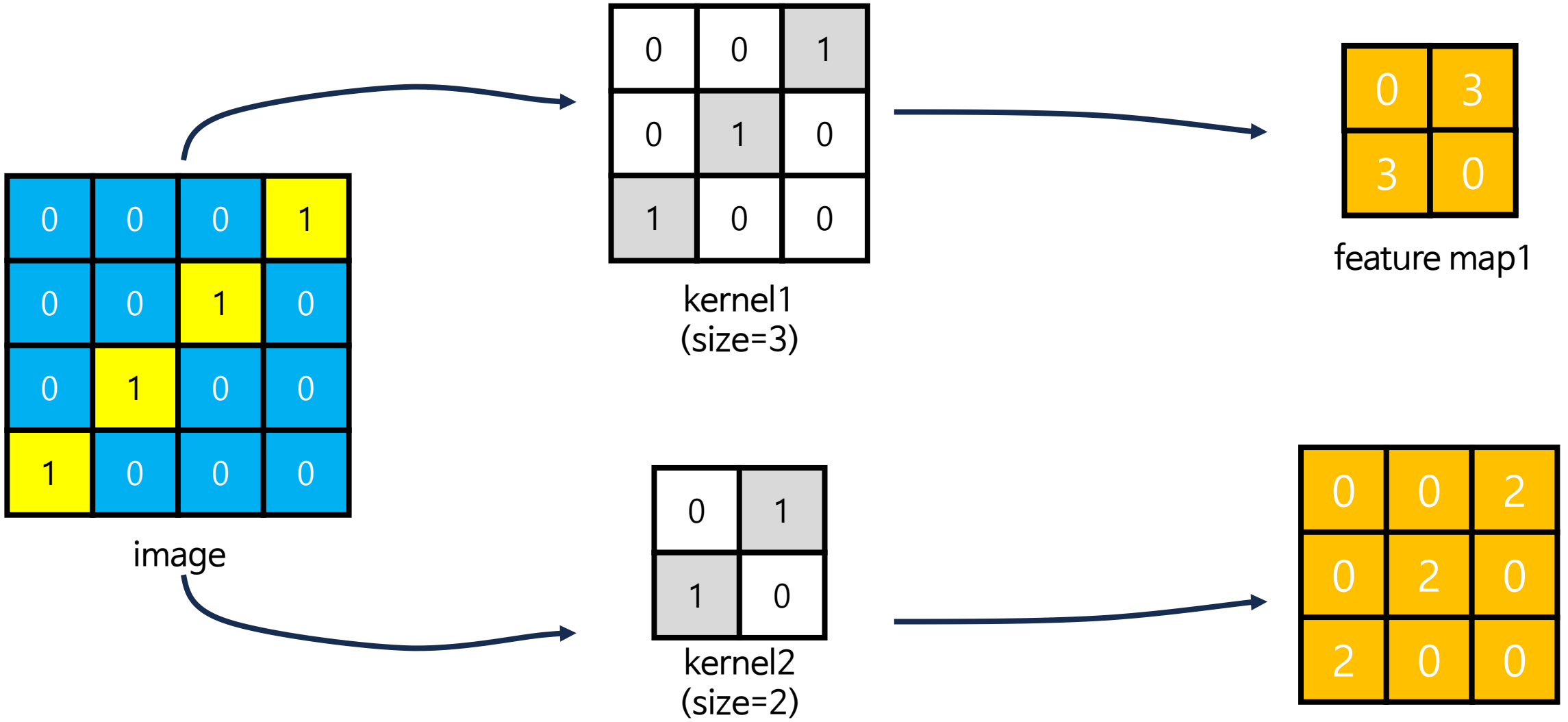
신박AI

그러므로 kernel의 사이즈가 작으면 디테일한 feature도 처리할 수 있게 되지만



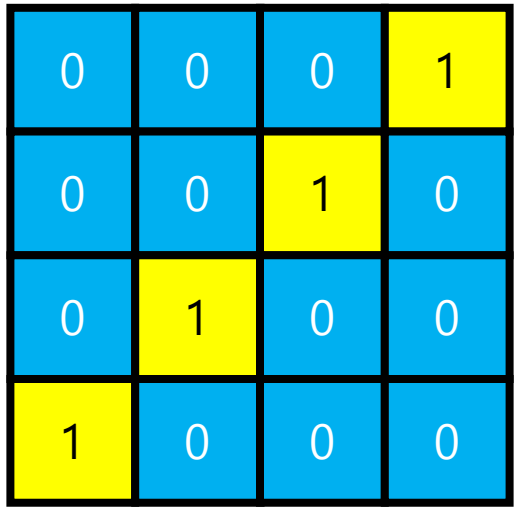
신박AI

그만큼 연산량도 증가하기 때문에 적절한 크기를 찾는 것이 중요합니다

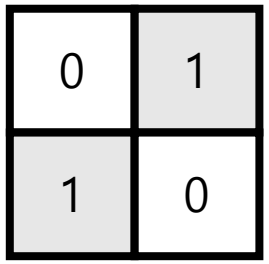


신박AI

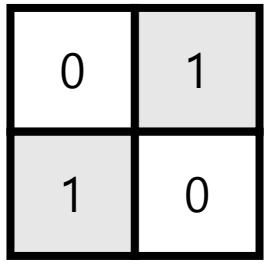
또한 보폭 (stride)도 feature map의 크기와 연산량을 결정짓는 중요한 요소입니다



image



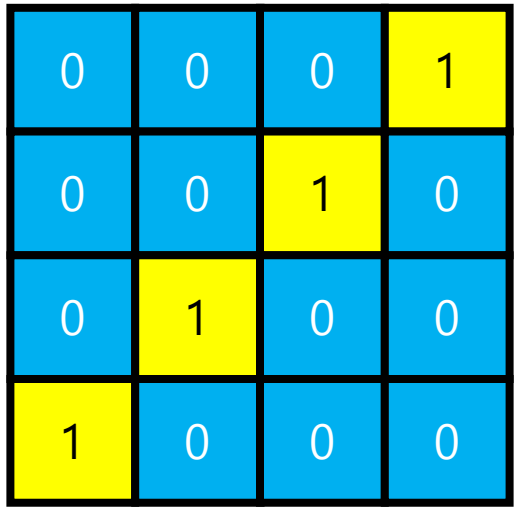
kernel1
(stride = 1)



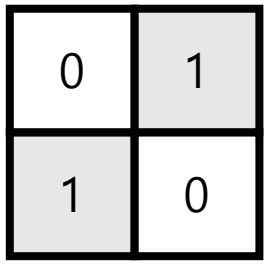
kernel2
(stride = 2)

신박AI

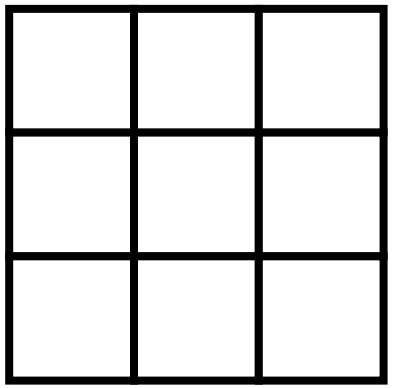
보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다



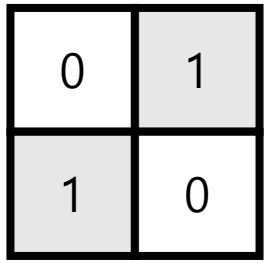
image



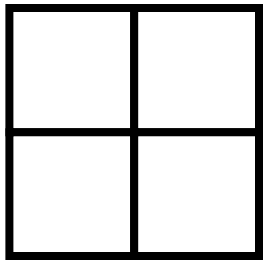
kernel1
(stride = 1)



feature map1



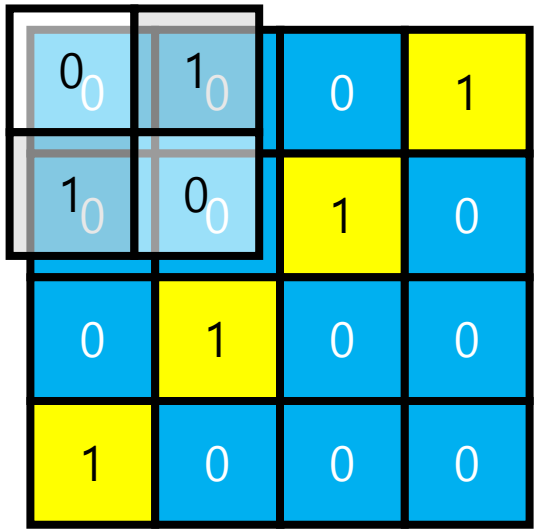
kernel2
(stride = 2)



feature map2

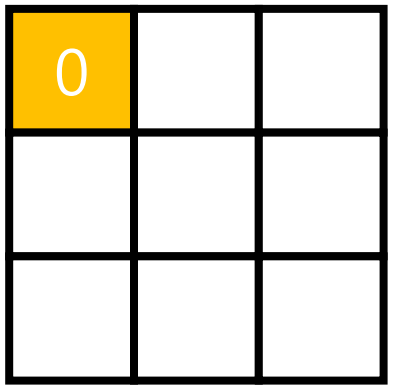
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

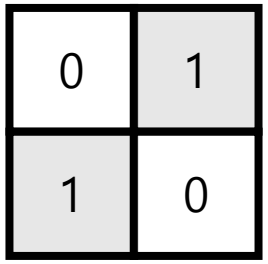


image

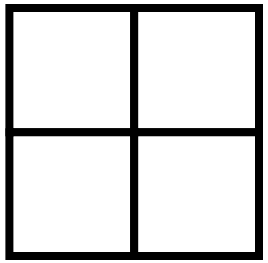
kernel1
(stride = 1)



feature map1



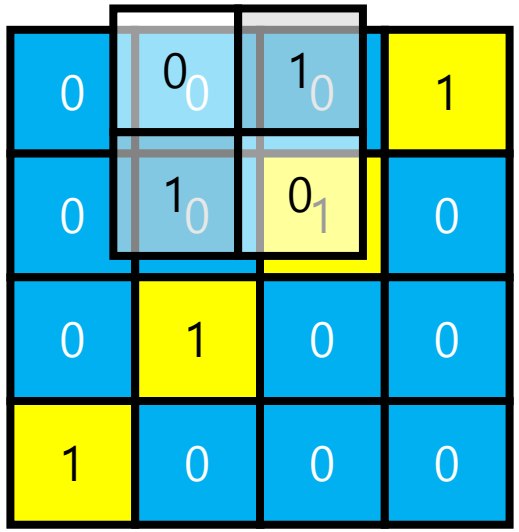
kernel2
(stride = 2)



feature map2

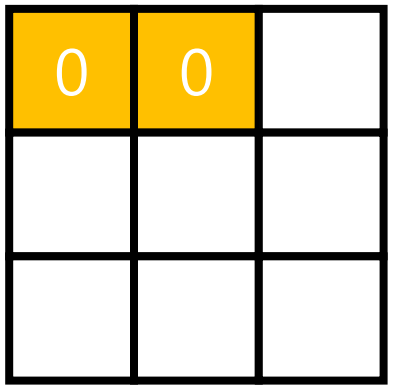
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

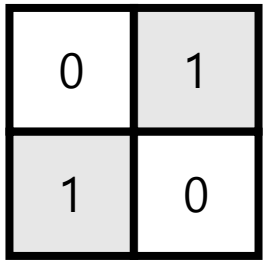


image

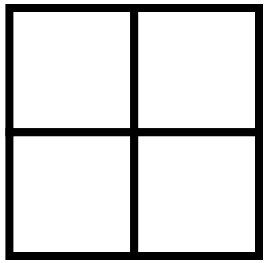
kernel1
(stride = 1)



feature map1



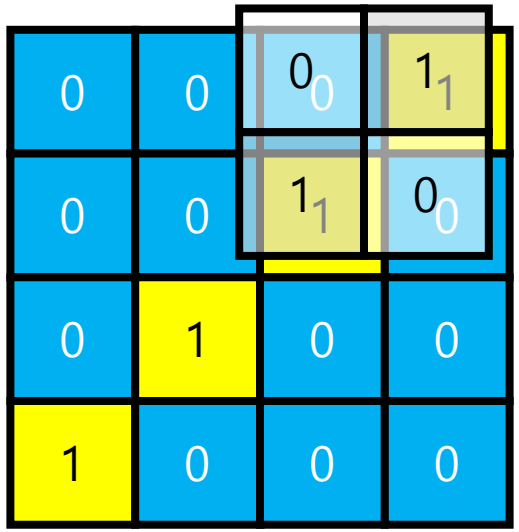
kernel2
(stride = 2)



feature map2

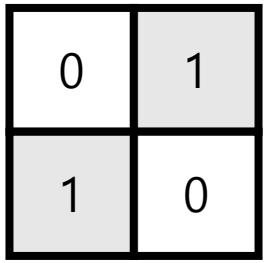
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

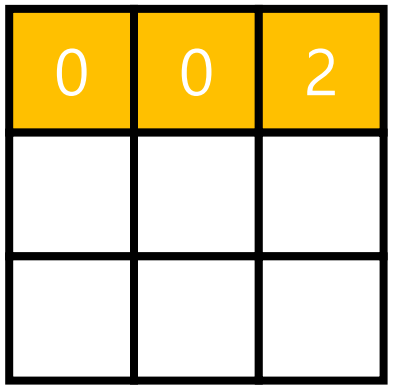


image

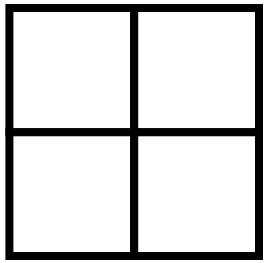
kernel1
(stride = 1)



kernel2
(stride = 2)



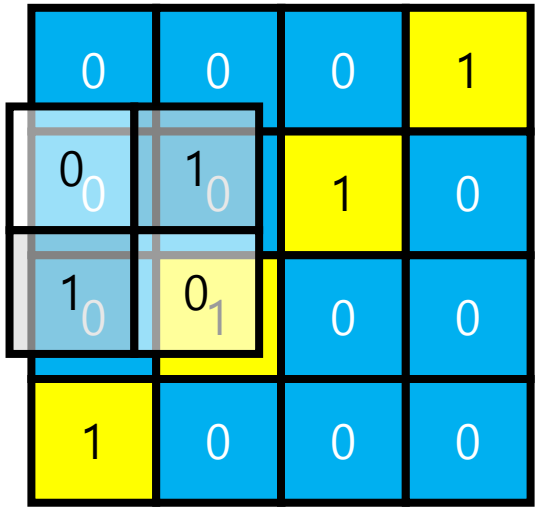
feature map1



feature map2

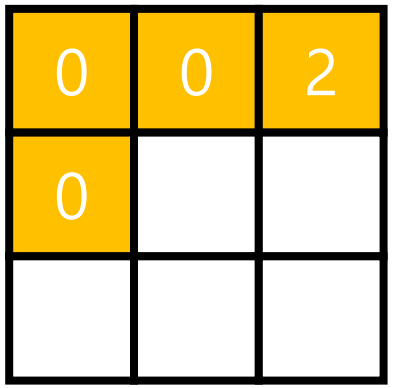
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

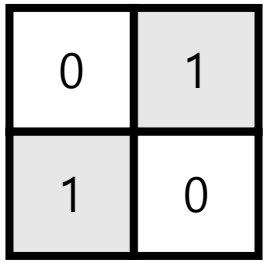


image

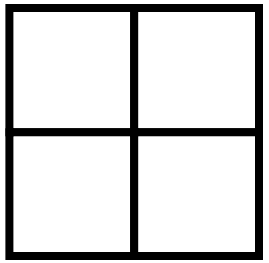
kernel1
(stride = 1)



feature map1



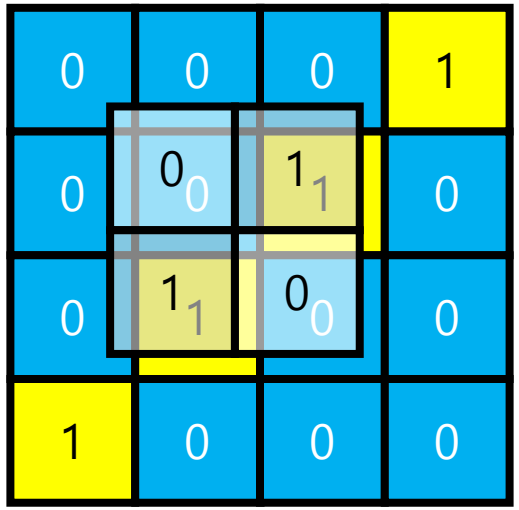
kernel2
(stride = 2)



feature map2

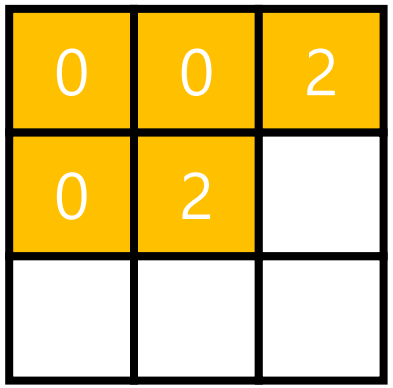
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

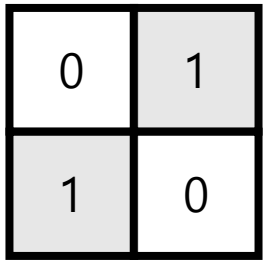


image

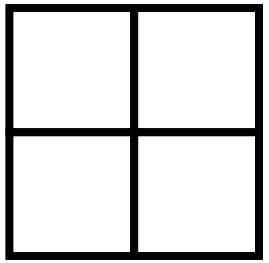
kernel1
(stride = 1)



feature map1



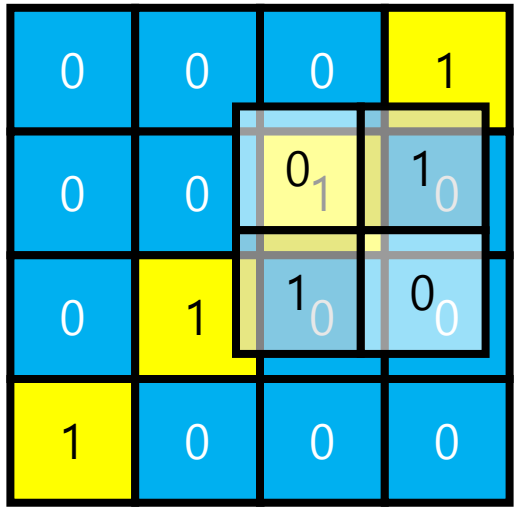
kernel2
(stride = 2)



feature map2

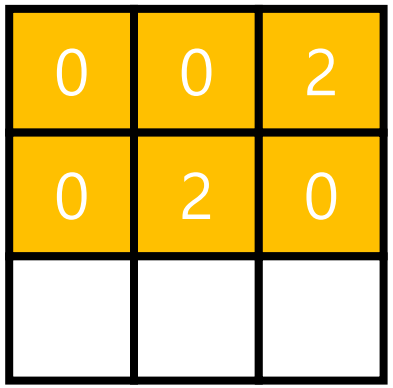
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

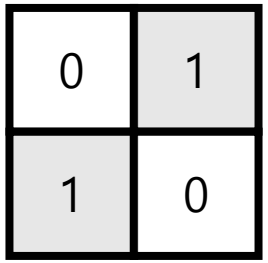


image

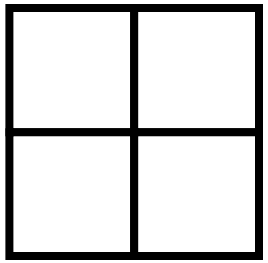
kernel1
(stride = 1)



feature map1



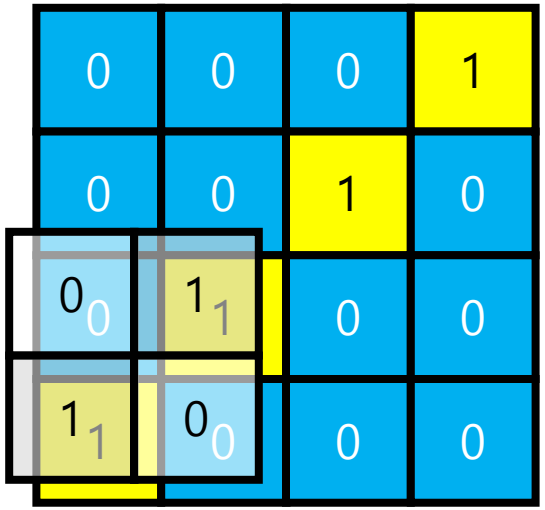
kernel2
(stride = 2)



feature map2

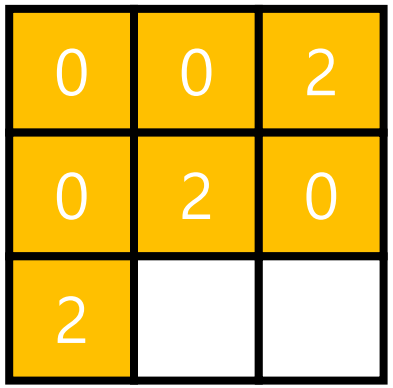
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

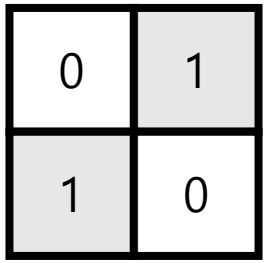


image

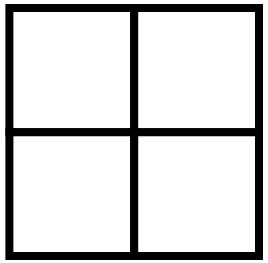
kernel1
(stride = 1)



feature map1



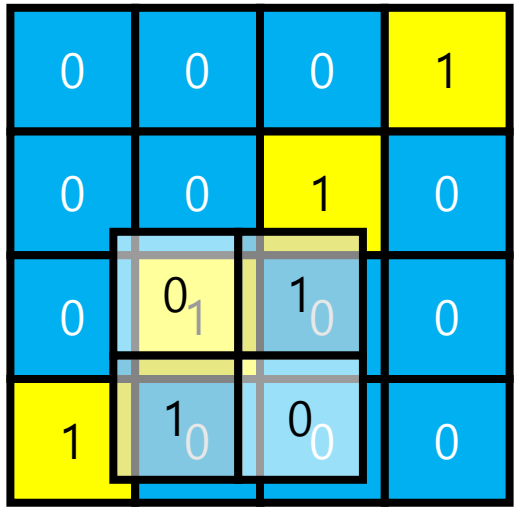
kernel2
(stride = 2)



feature map2

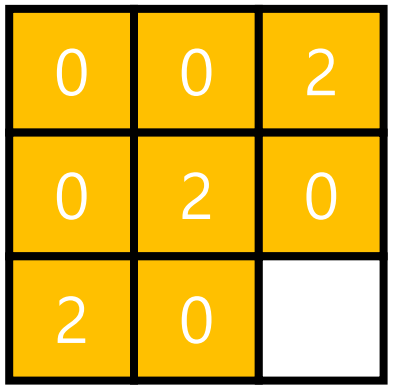
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

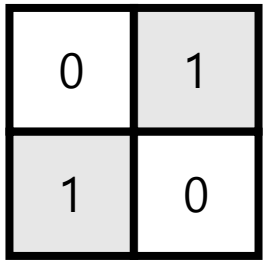


image

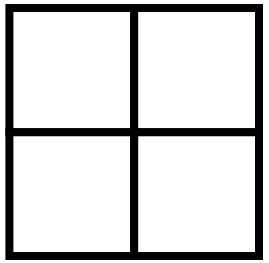
kernel1
(stride = 1)



feature map1



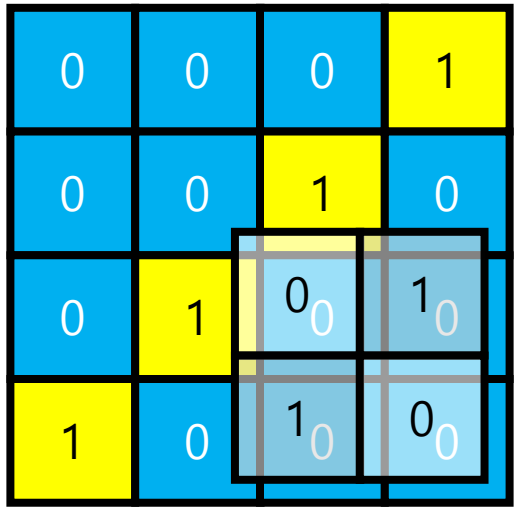
kernel2
(stride = 2)



feature map2

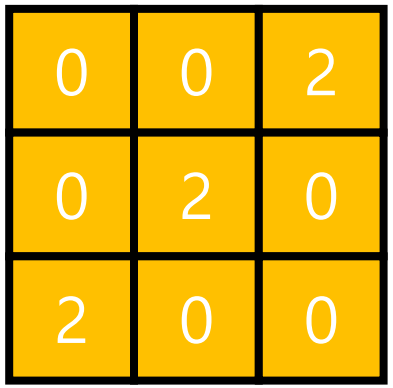
신박AI

보폭 stride가 1일 경우는 한칸 씩 움직이며 합성곱연산을 합니다

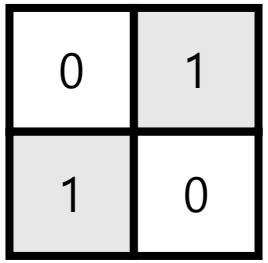


image

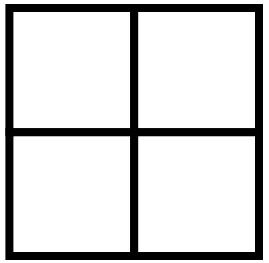
kernel1
(stride = 1)



feature map1



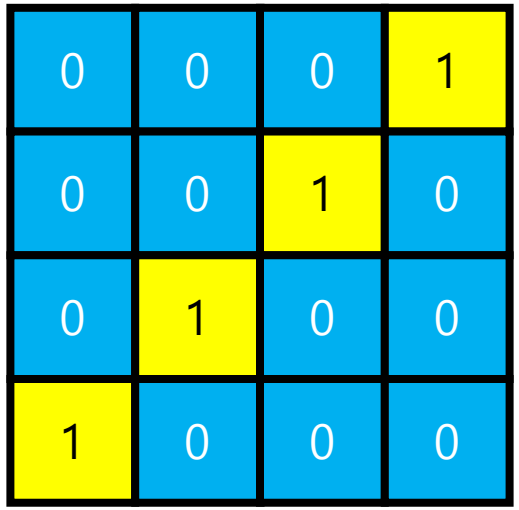
kernel2
(stride = 2)



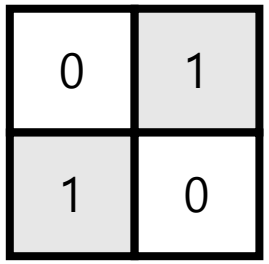
feature map2

신박AI

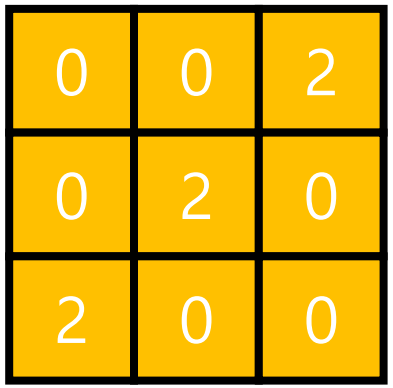
보폭 stride가 2일 경우는 두칸 씩 움직이며 합성곱연산을 합니다



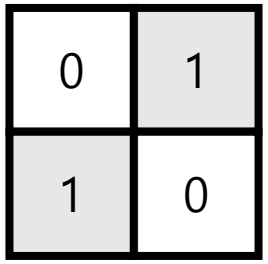
image



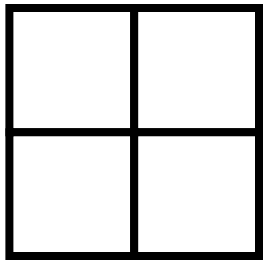
kernel1
(stride = 1)



feature map1



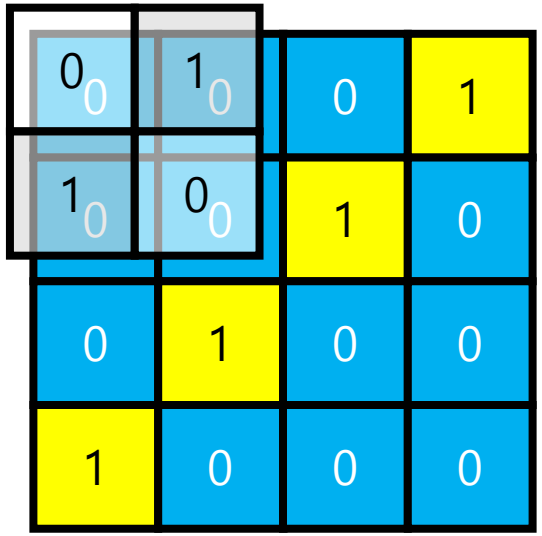
kernel2
(stride = 2)



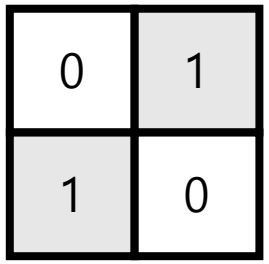
feature map2

신박AI

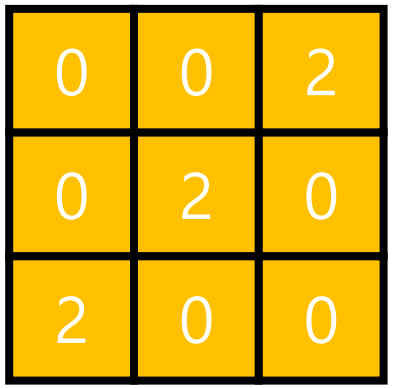
보폭 stride가 2일 경우는 두칸 씩 움직이며 합성곱연산을 합니다



image

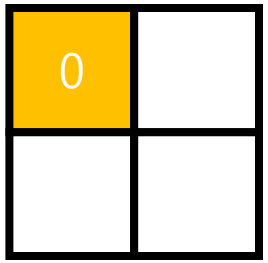


kernel1
(stride = 1)



feature map1

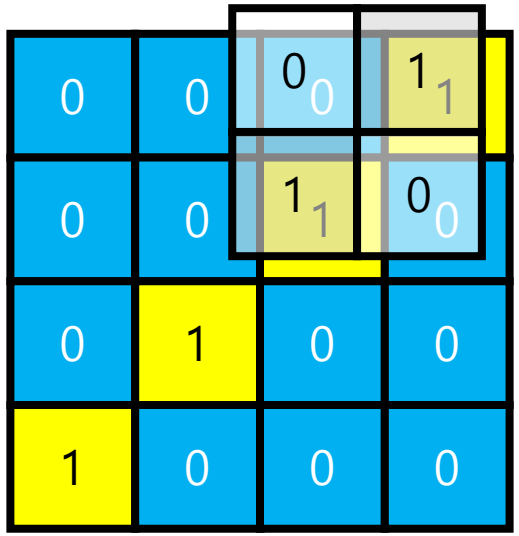
kernel2
(stride = 2)



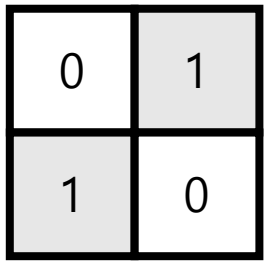
feature map2

신박AI

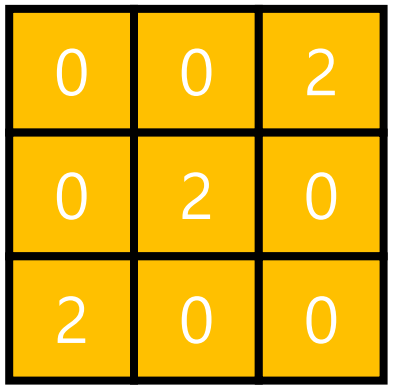
보폭 stride가 2일 경우는 두칸 씩 움직이며 합성곱연산을 합니다



image

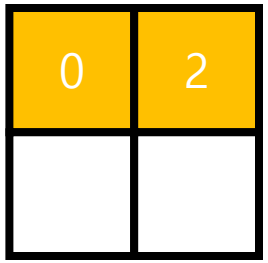


kernel1
(stride = 1)



feature map1

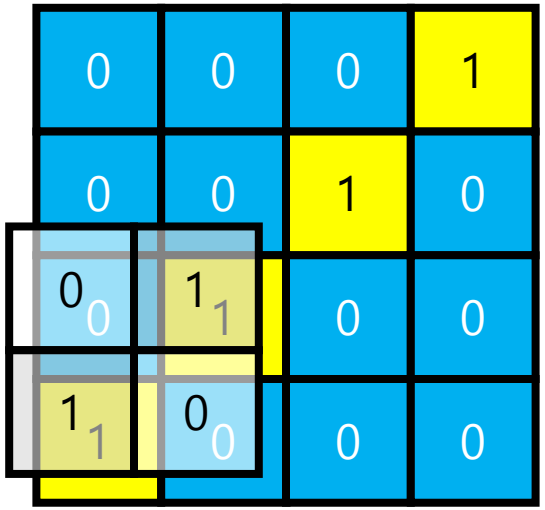
kernel2
(stride = 2)



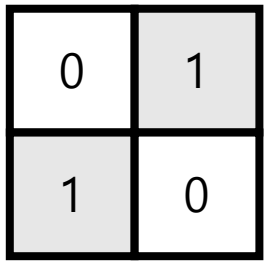
feature map2

신박AI

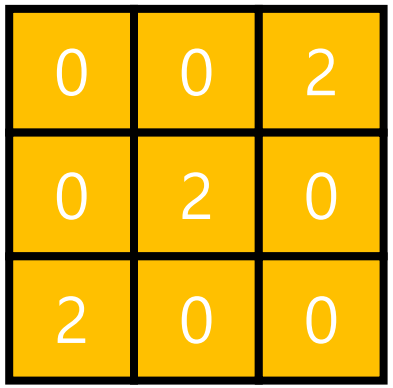
보폭 stride가 2일 경우는 두칸 씩 움직이며 합성곱연산을 합니다



image

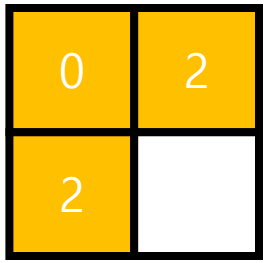


kernel1
(stride = 1)



feature map1

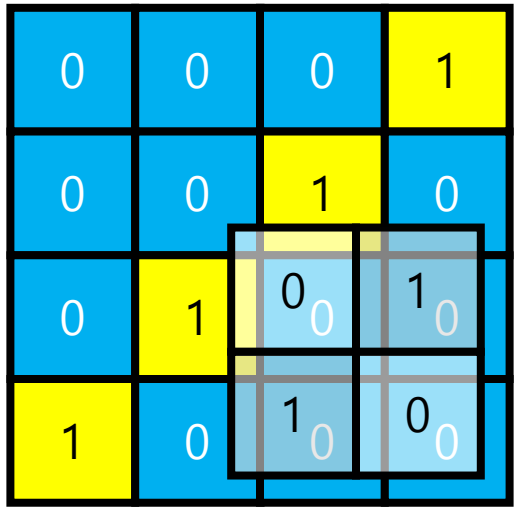
kernel2
(stride = 2)



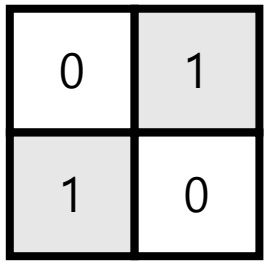
feature map2

신박AI

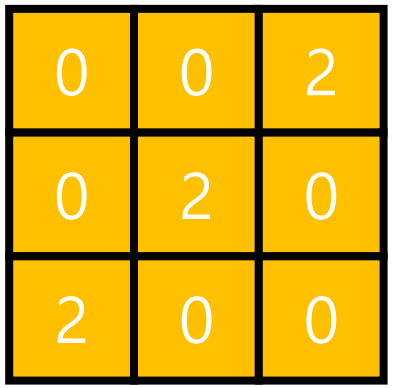
보폭 stride가 2일 경우는 두칸 씩 움직이며 합성곱연산을 합니다



image

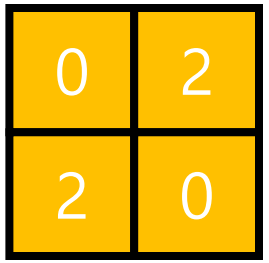


kernel1
(stride = 1)



feature map1

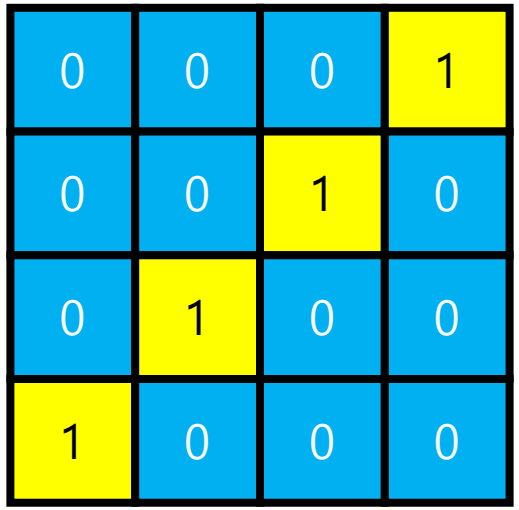
kernel2
(stride = 2)



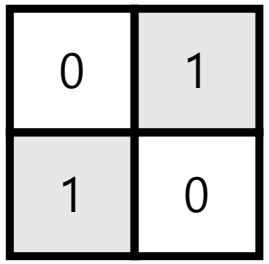
feature map2

신박AI

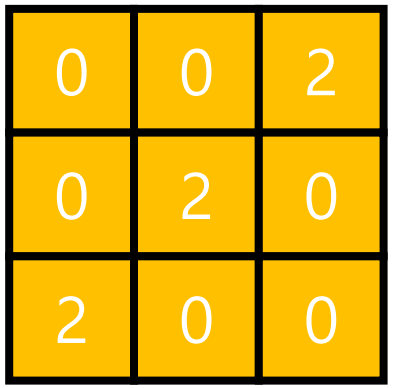
보폭 stride가 2일 경우는 두칸 씩 움직이며 합성곱연산을 합니다



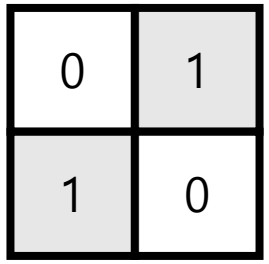
image



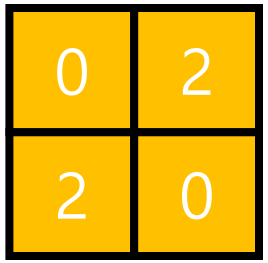
kernel1
(stride = 1)



feature map1



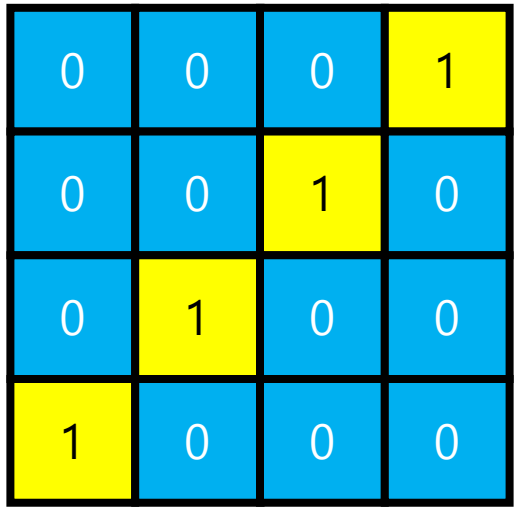
kernel2
(stride = 2)



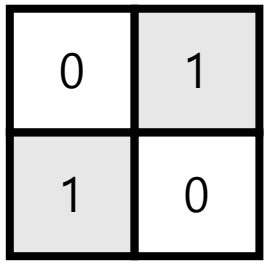
feature map2

신박AI

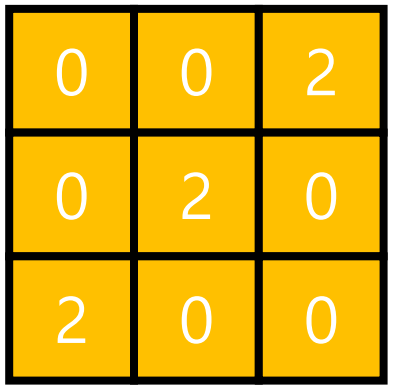
이처럼 kernel 크기가 같아도 보폭 stride에 따라 feature map의 크기가 달라질 수 있습니다



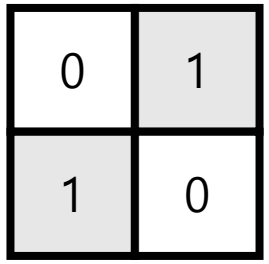
image



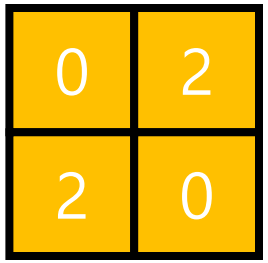
kernel1
(stride = 1)



feature map1



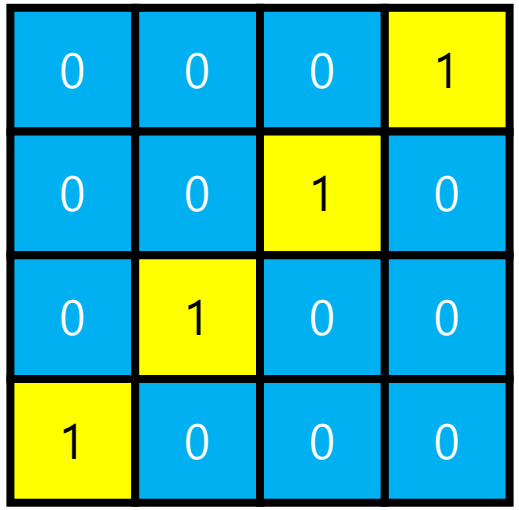
kernel2
(stride = 2)



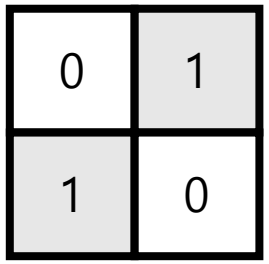
feature map2

신박AI

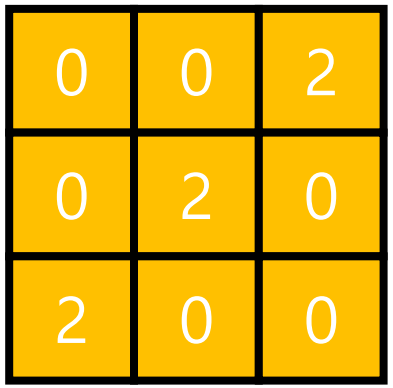
kernel의 크기와 보폭은 연산량과 모델의 정확성 사이에서 밸런스를 맞추어 설정하시면 되겠습니다



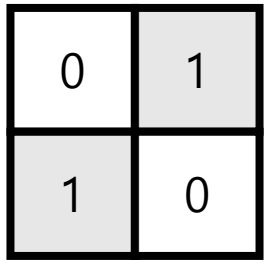
image



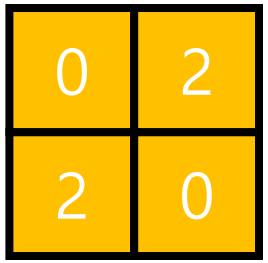
kernel1
(stride = 1)



feature map1



kernel2
(stride = 2)

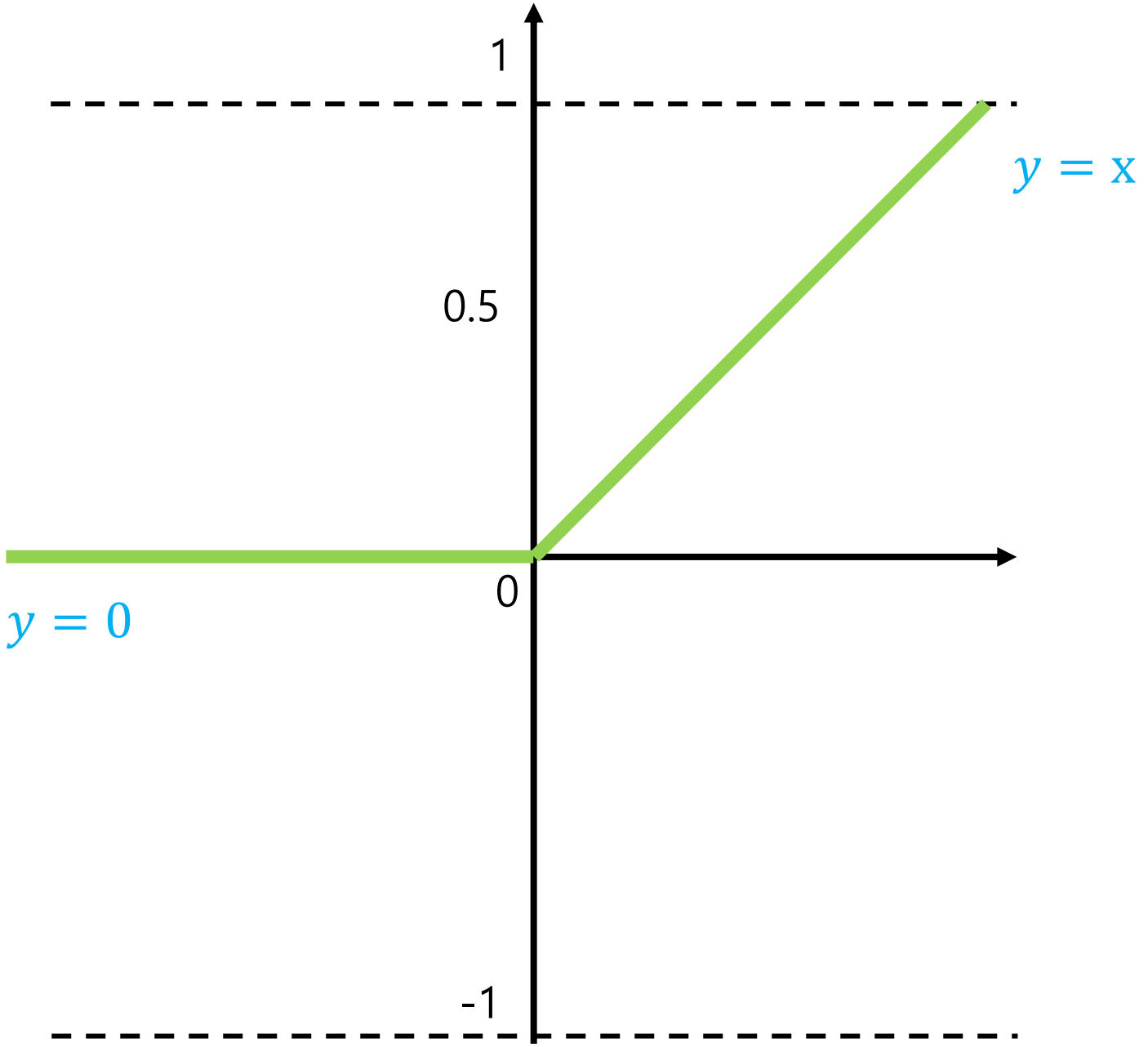


feature map2

신박AI

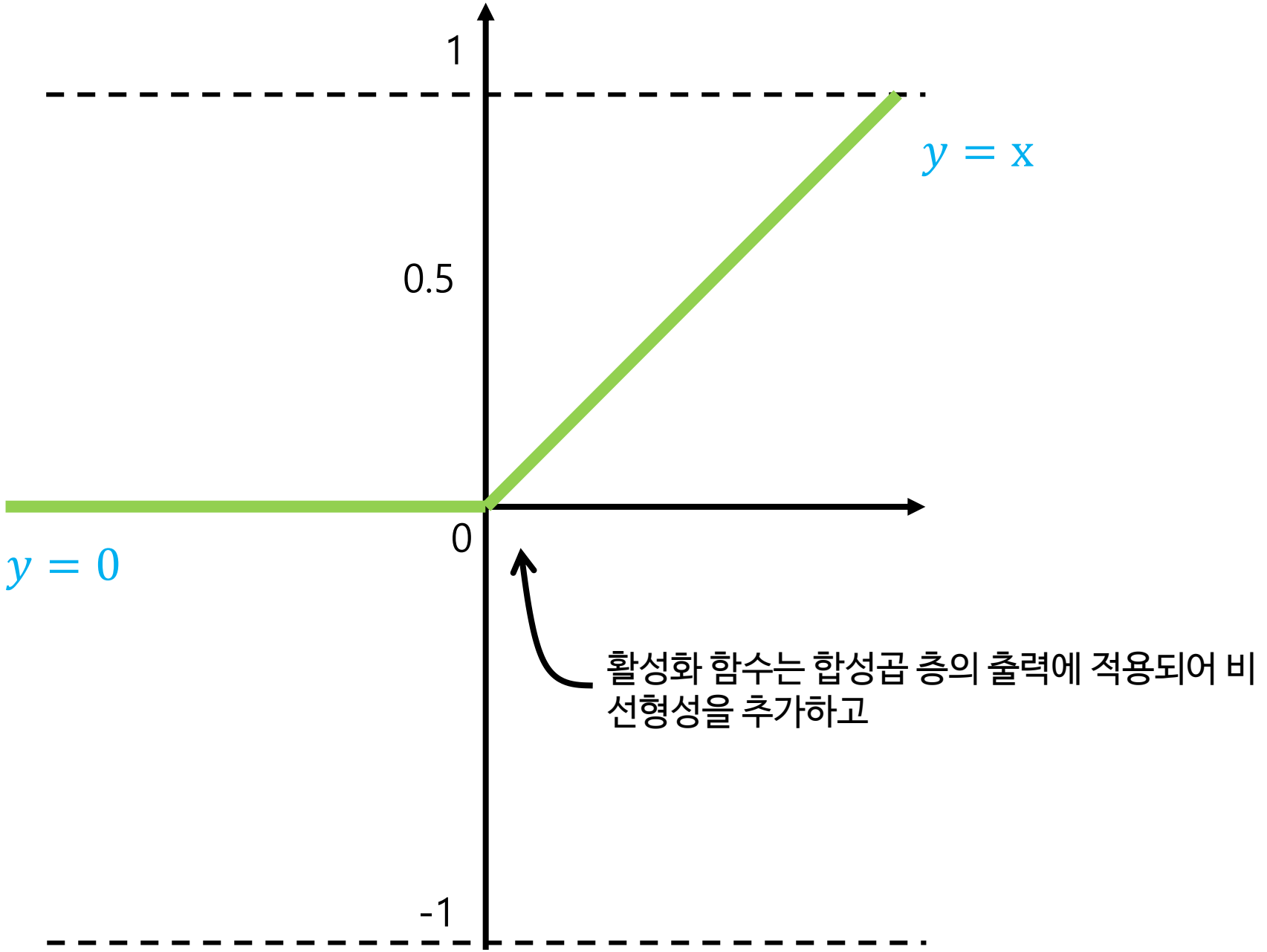
또한 합성곱 층은 보통 활성화 함수(activation function)를 사용하여 비선형성을 도입합니다

대표적인 활성화 함수로는 ReLU(Rectified Linear Unit) 함수가 있습니다



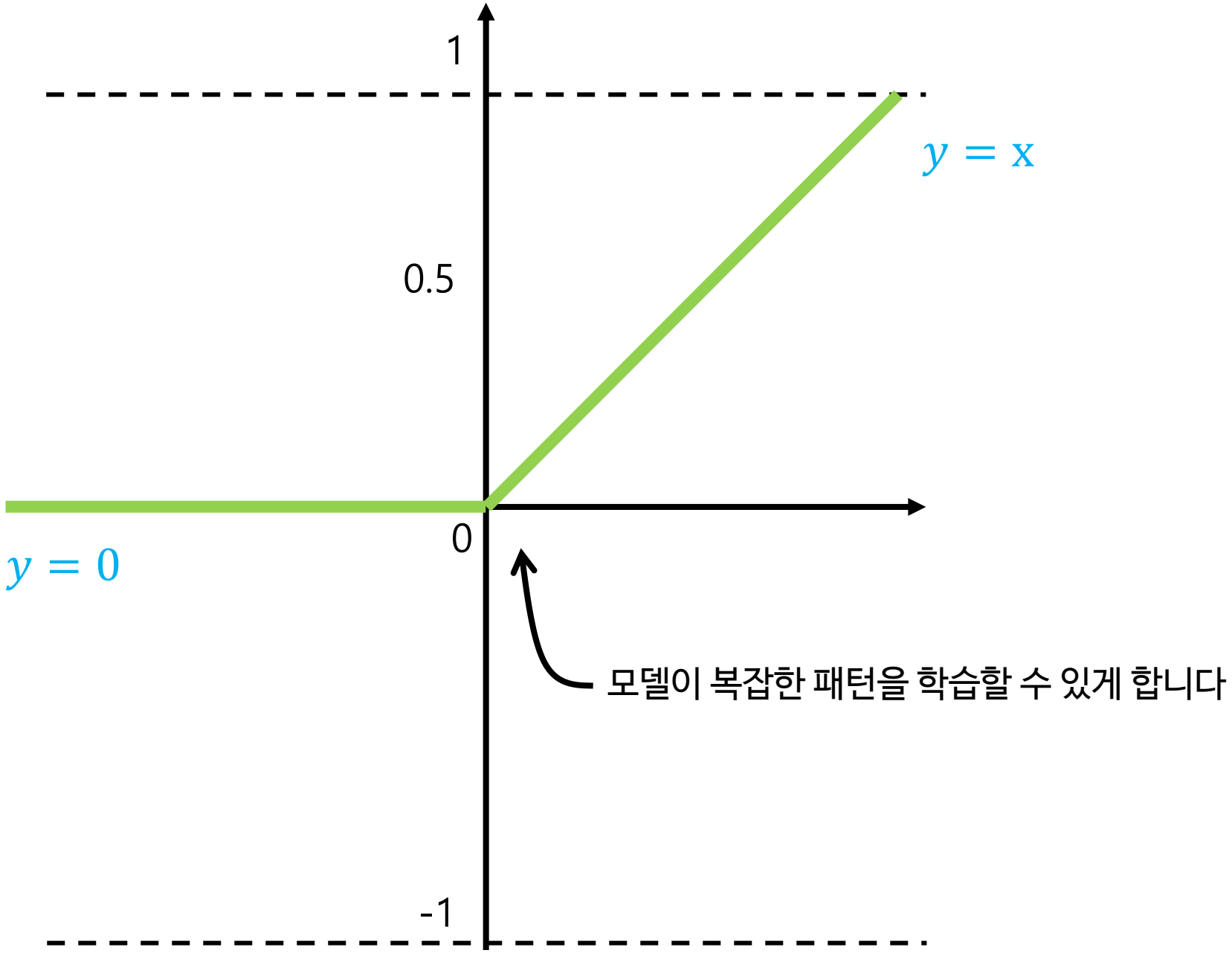
신박AI

대표적인 활성화 함수로는 ReLU(Rectified Linear Unit) 함수가 있습니다



신박AI

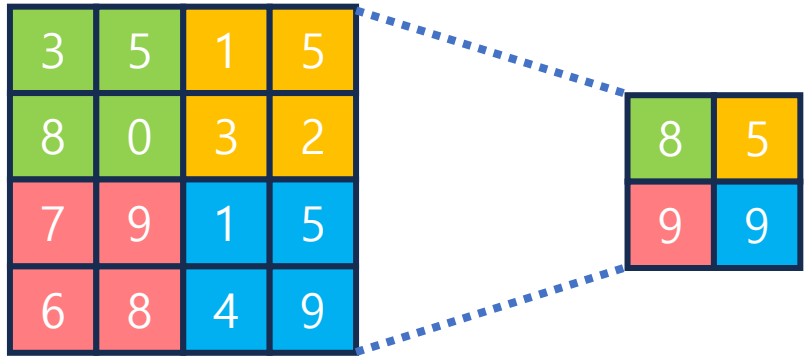
대표적인 활성화 함수로는 ReLU(Rectified Linear Unit) 함수가 있습니다



신박AI

다음으로, 풀링 층은 합성곱 층에서 추출된 feature map 의 크기를 줄이는 역할을 합니다

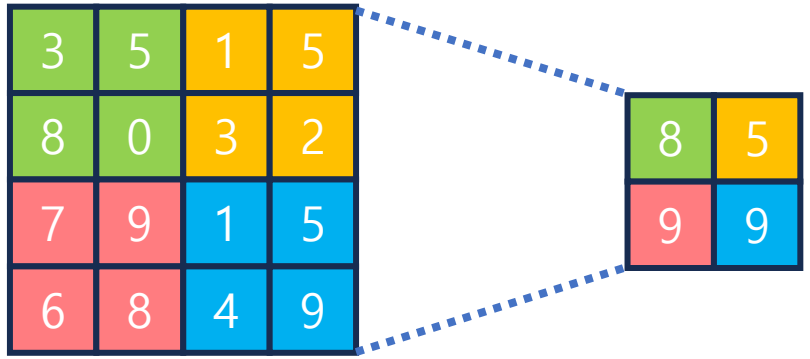
합성곱 층 Convolutional layer



풀링 층 Pooling layer

밀집 층 Fully-connected layer
Dense layer

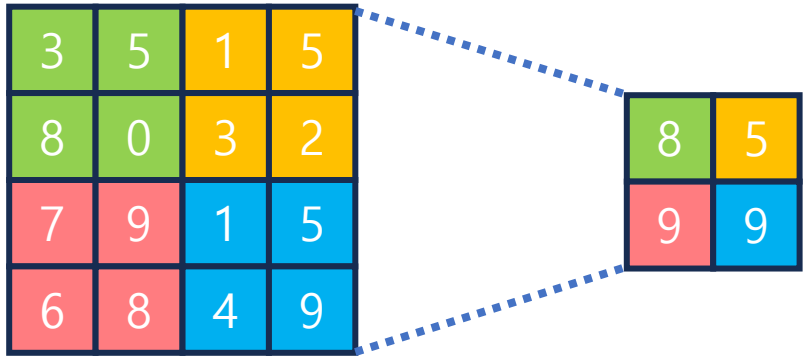
CNN에서 사용하는 풀링층은 크게 두가지가 있습니다



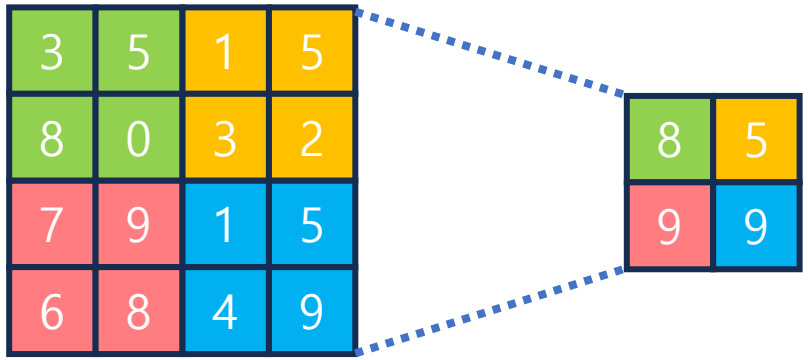
풀링 층 Pooling layer

신박AI

하나는 최대 풀링이고



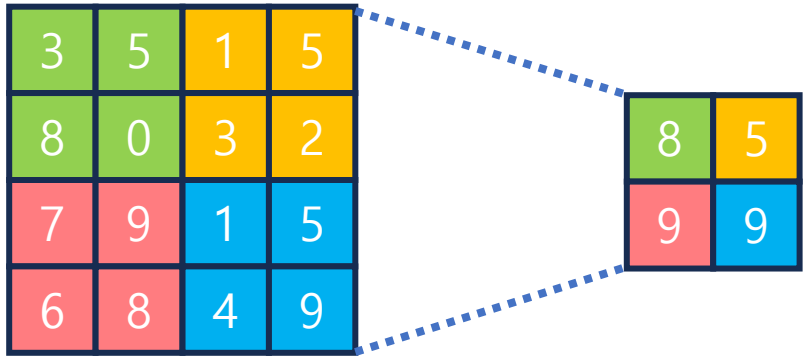
풀링 층 Pooling layer



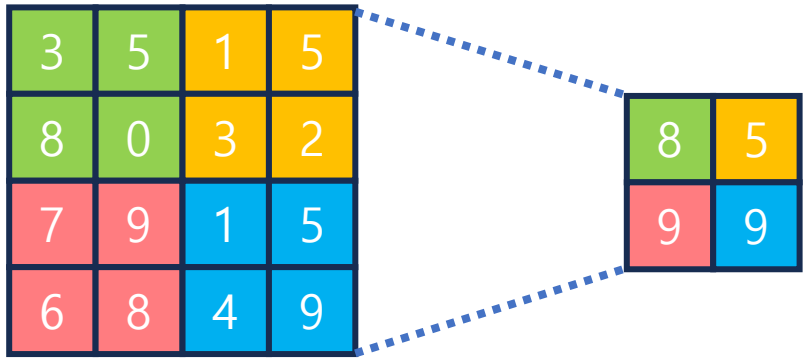
최대 풀링 층 Max Pooling layer

신박AI

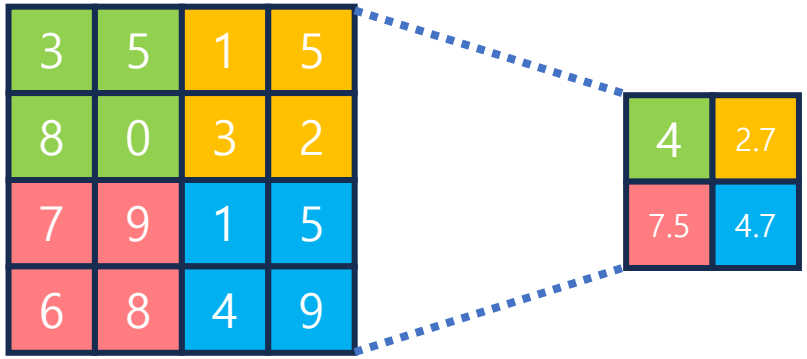
또 다른 하나는 평균 풀링입니다



풀링 층 Pooling layer



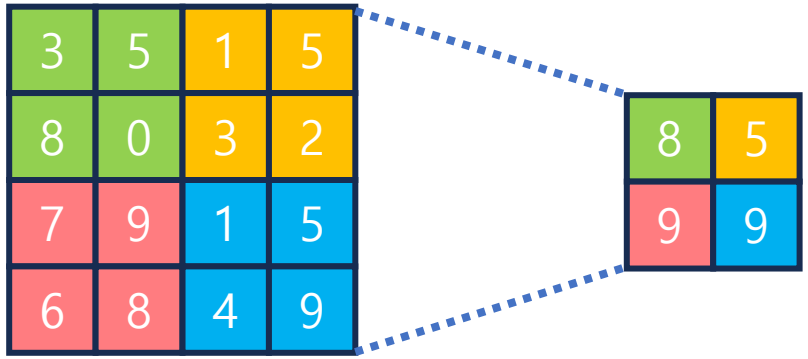
최대 풀링 층 Max Pooling layer



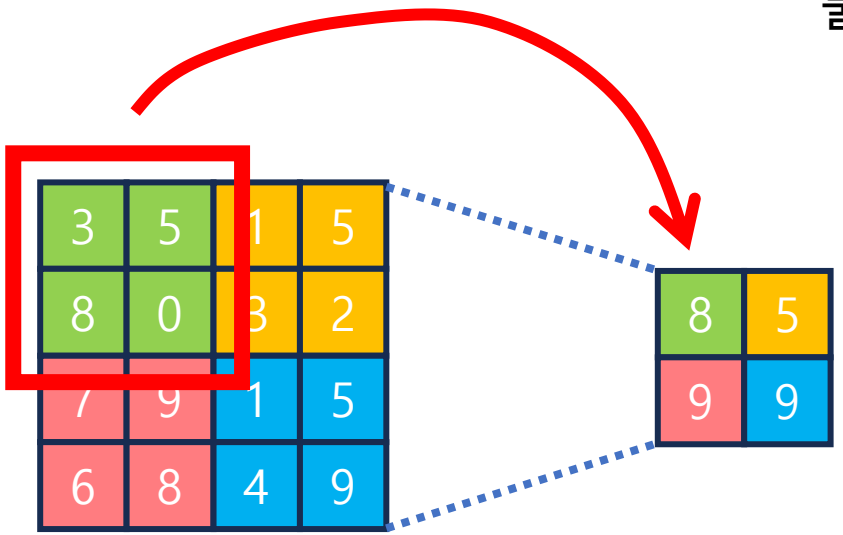
평균 풀링 층 Average Pooling layer

신박AI

최대풀링층은 주어진 풀링 윈도우 내의 값중 최대 값을 리턴하고



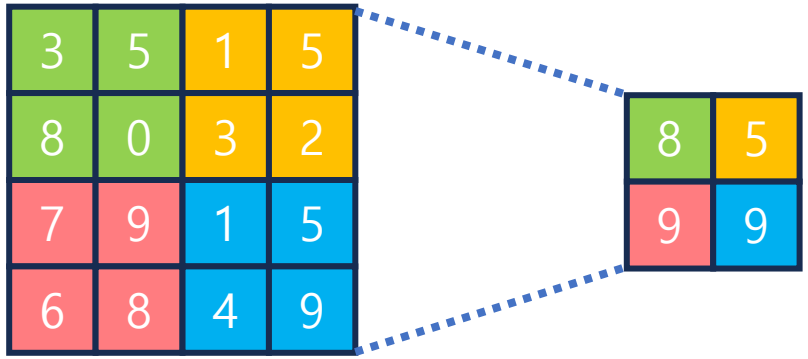
풀링 층 Pooling layer



최대 풀링 층 Max Pooling layer

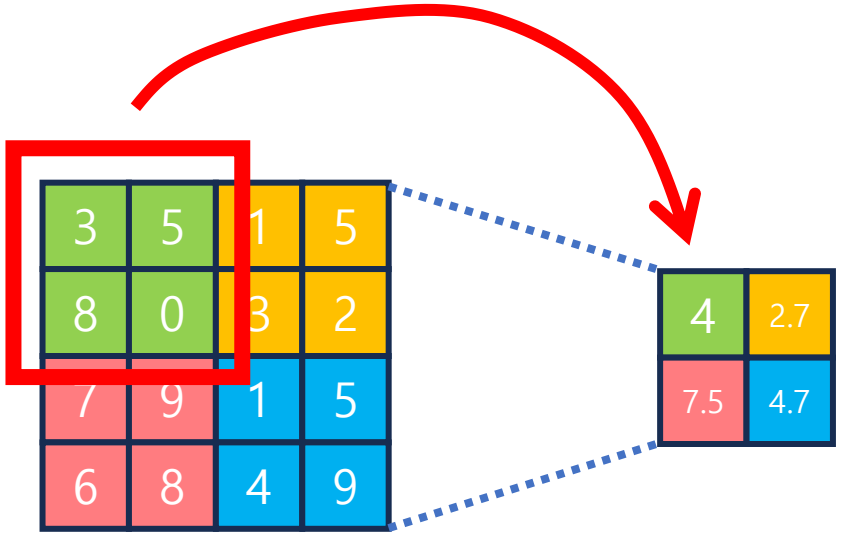
평균 풀링 층 Average Pooling layer

평균풀링층은 주어진 풀링 윈도우 내의 값의 평균값을 리턴합니다



풀링 층 Pooling layer

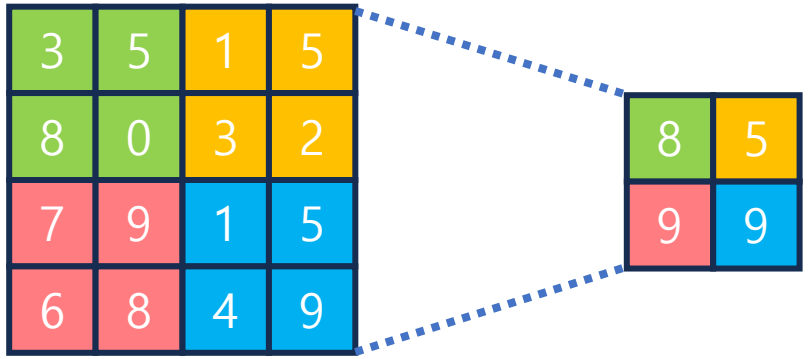
최대 풀링 층 Max Pooling layer



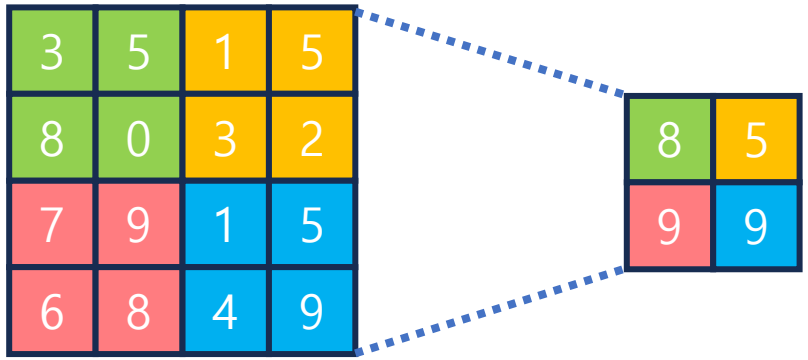
평균 풀링 층 Average Pooling layer

신박AI

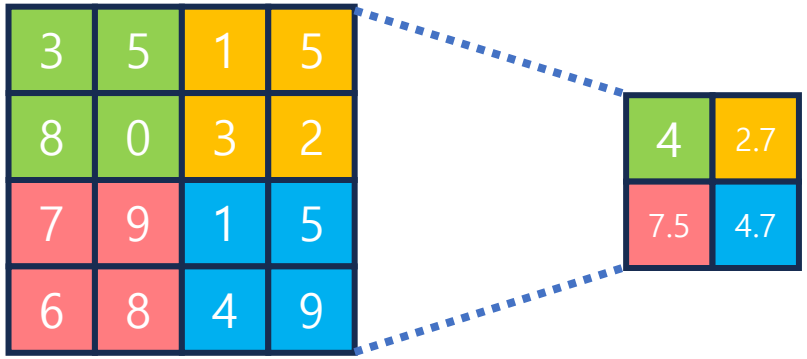
이런 풀링층을 통해 공간적인 크기를 줄이고



풀링 층 Pooling layer



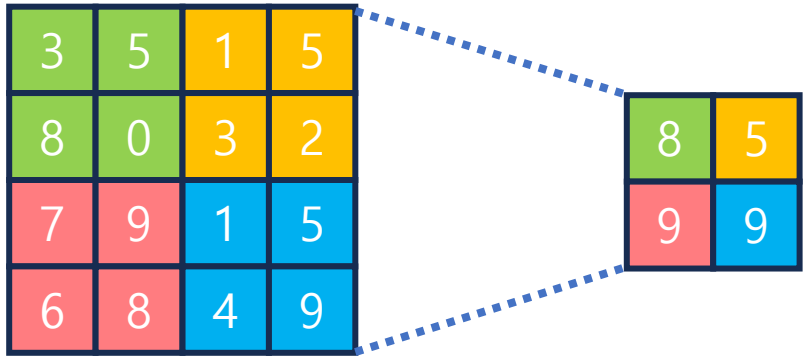
최대 풀링 층 Max Pooling layer



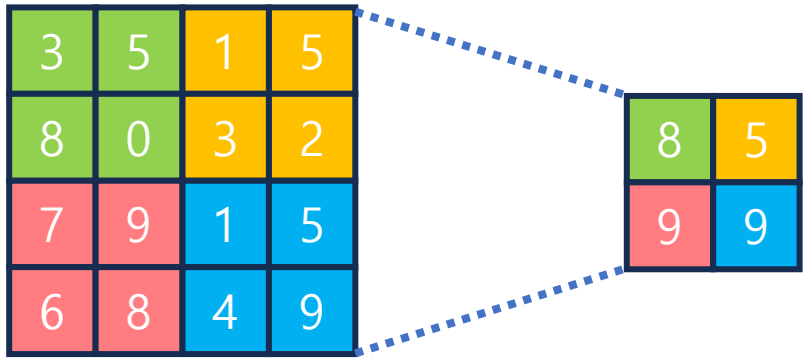
평균 풀링 층 Average Pooling layer

신박AI

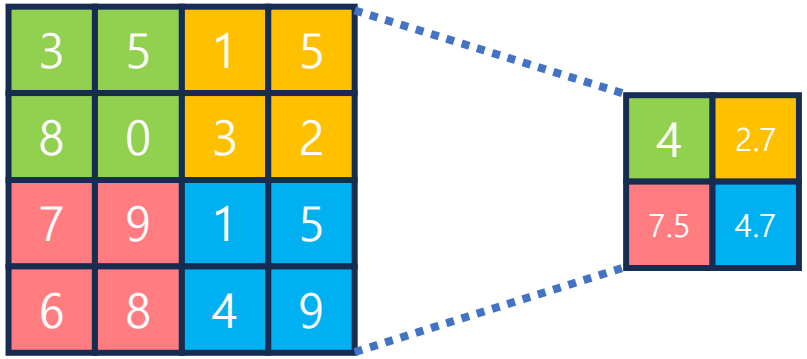
계산 비용을 줄이면서 중요한 특징을 보존할 수 있습니다



풀링 층 Pooling layer



최대 풀링 층 Max Pooling layer



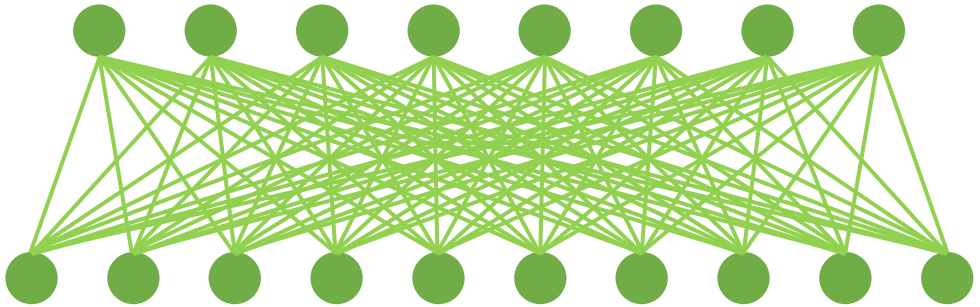
평균 풀링 층 Average Pooling layer

신박AI

마지막으로, 밀집 층은 추출된 feature들을 기반으로 최종 출력을 생성하는 역할을 합니다

합성곱 층 Convolutional layer

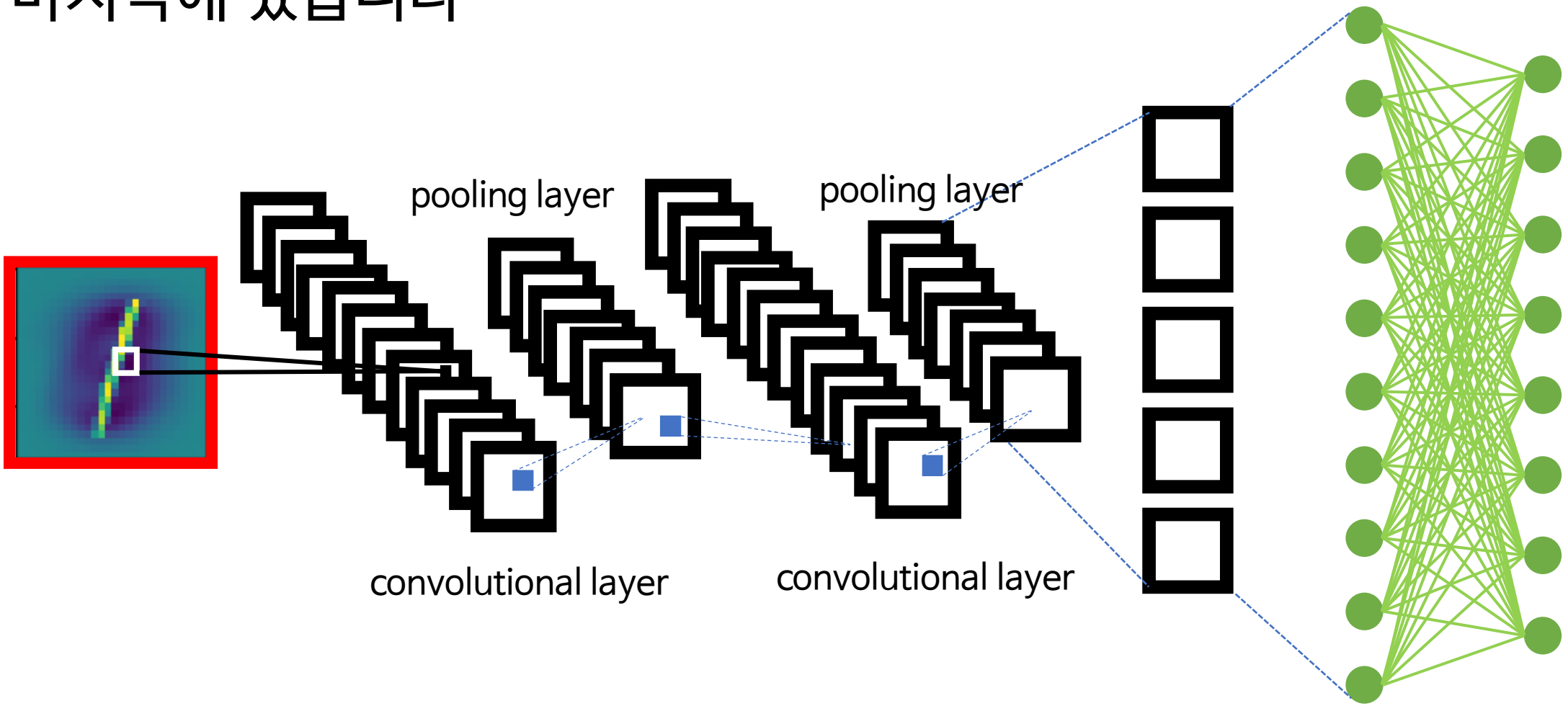
풀링 층 Pooling layer



밀집 층 Fully-connected layer
Dense layer

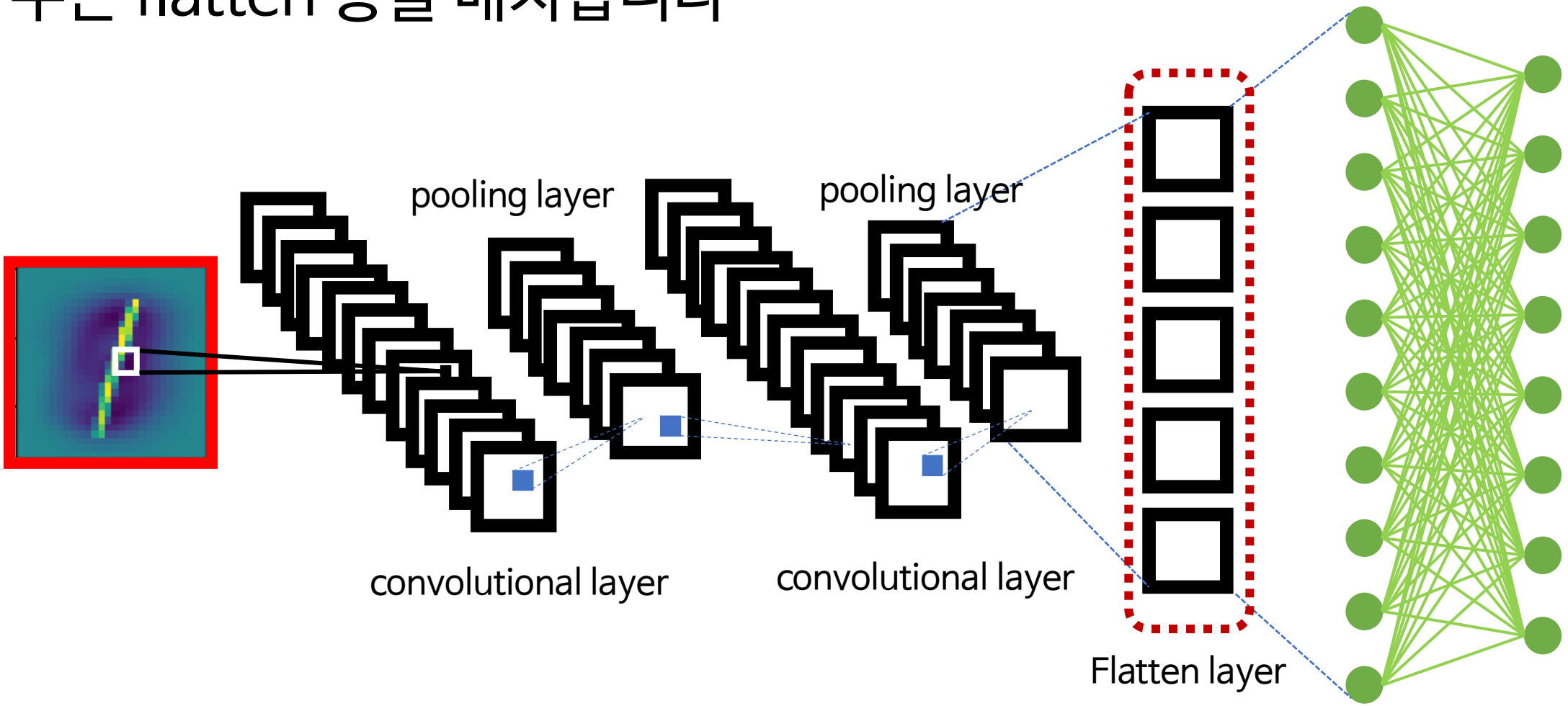
신박AI

최종 출력을 생성하기 위해 보통 fully-connected layer는 CNN모델의 마지막에 있습니다



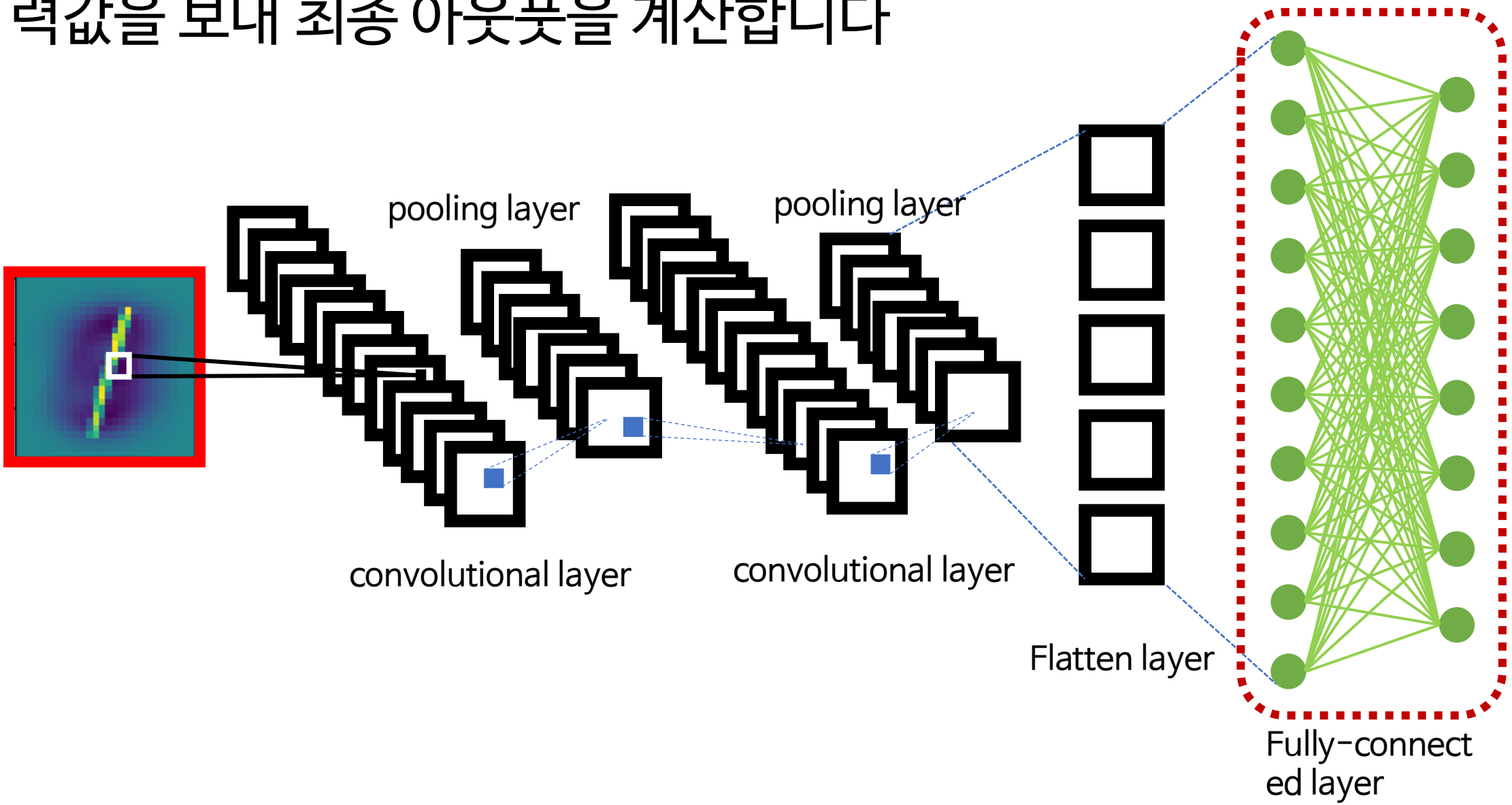
신박AI

fully-connected layer 전에 모든 local feature를 1차원으로 만들어 주는 flatten 층을 배치합니다



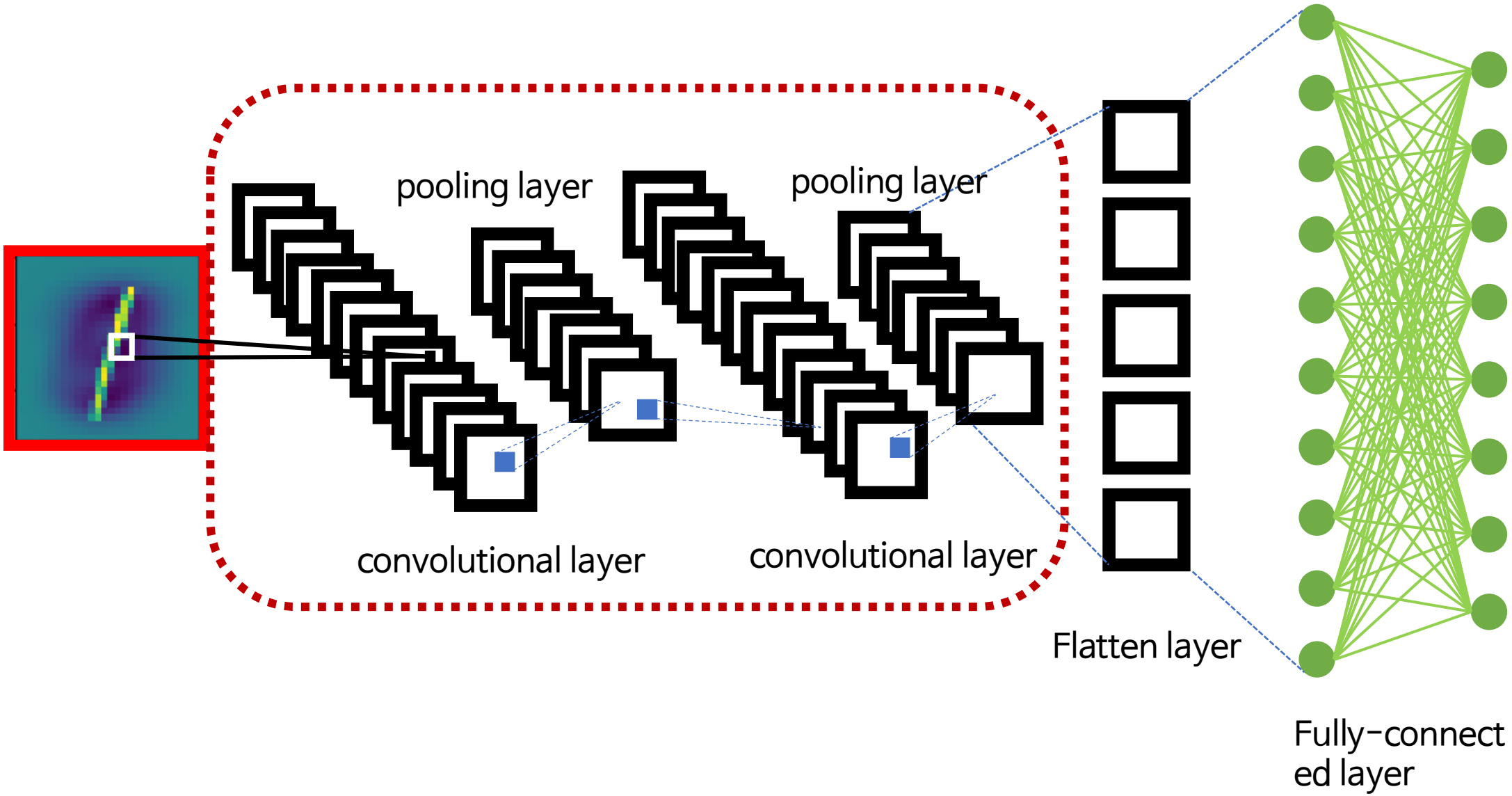
신박AI

그런 다음 모든 local feature들과 연결된 fully-connected layer로 출력값을 보내 최종 아웃풋을 계산합니다



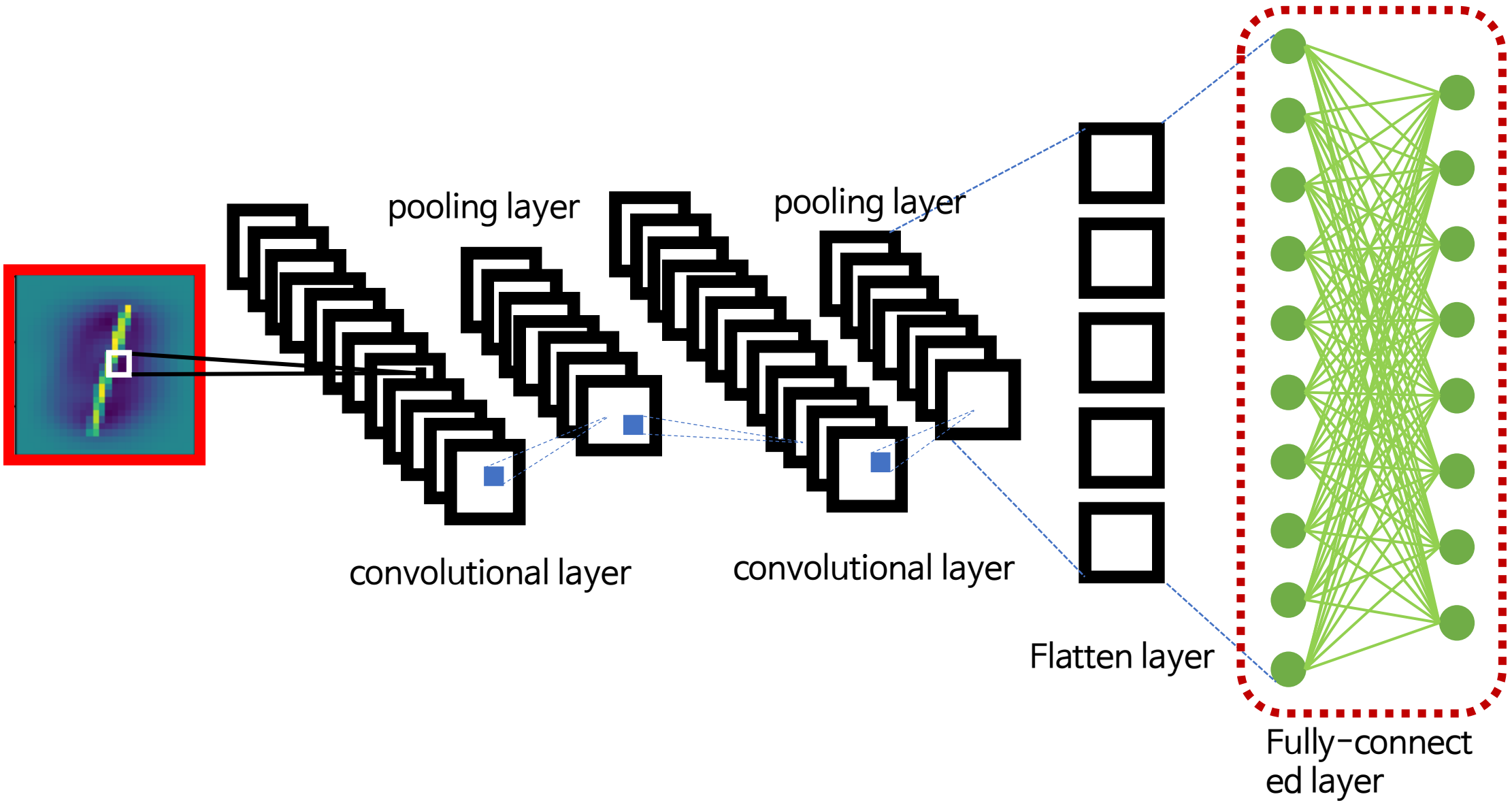
신박AI

Local feature들은 어디까지나 '지역적'인 확률이기 때문에..



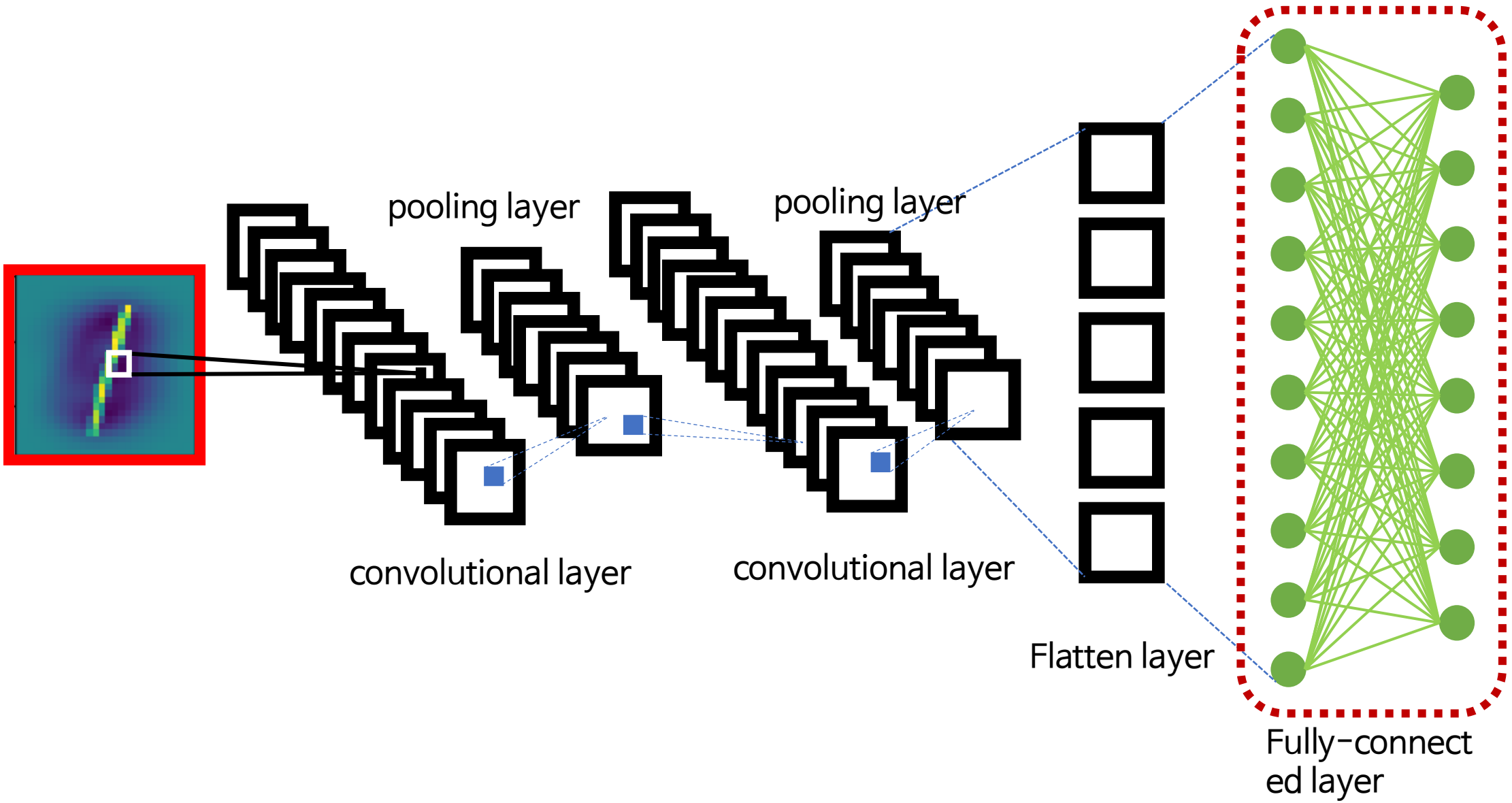
신박AI

최종 확률을 계산하기 위해서는 반드시 전체를 고려해야 합니다



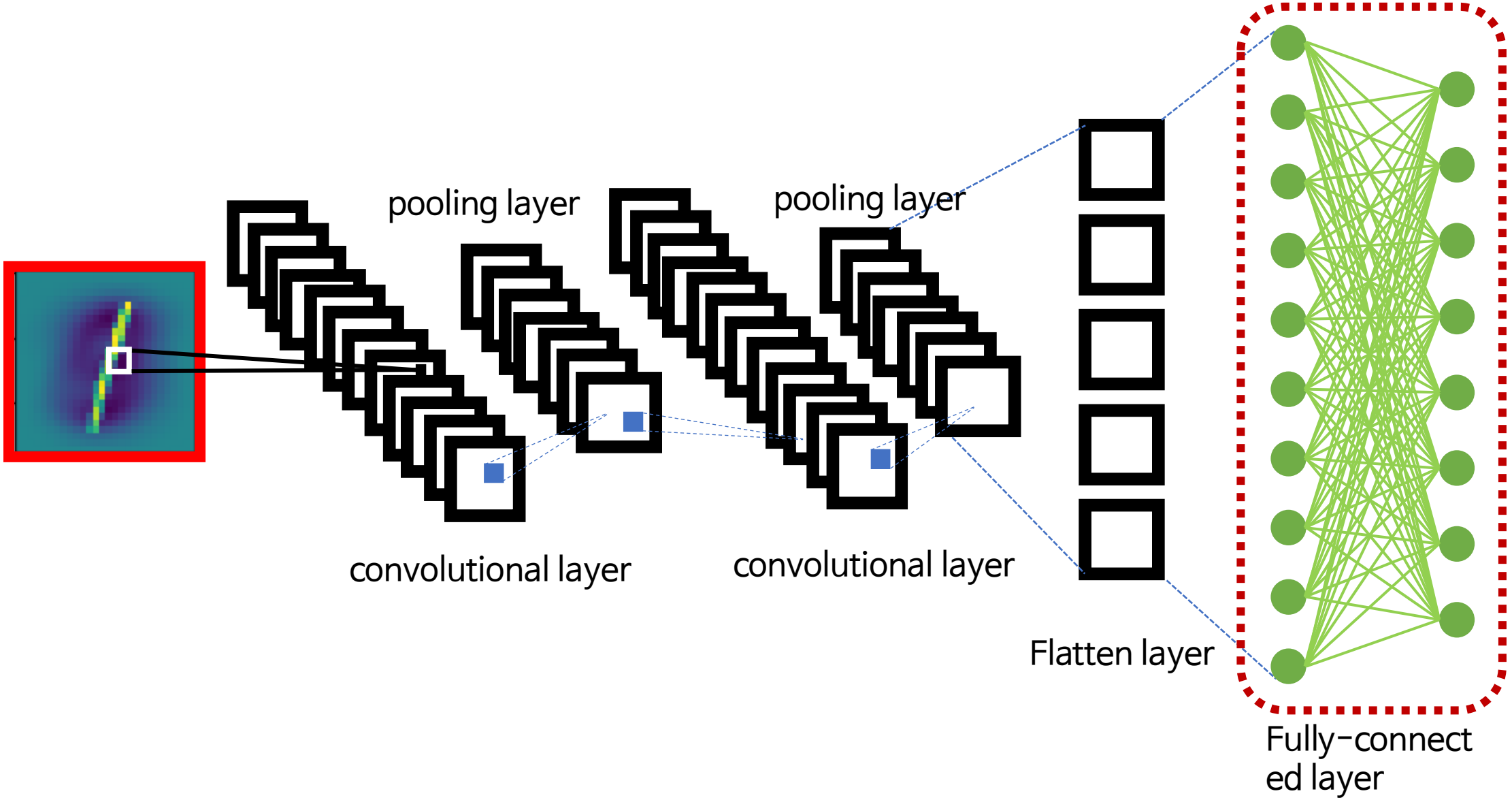
신박AI

그래서 모든 local feature의 입력을 받는 fully-connected layer가



신박AI

마지막 단계에서 필요한 것입니다



신박AI

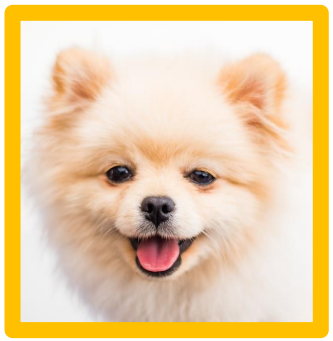
자 그러면 이런 각각의 층들이 모인 CNN이 어떻게 학습하는지

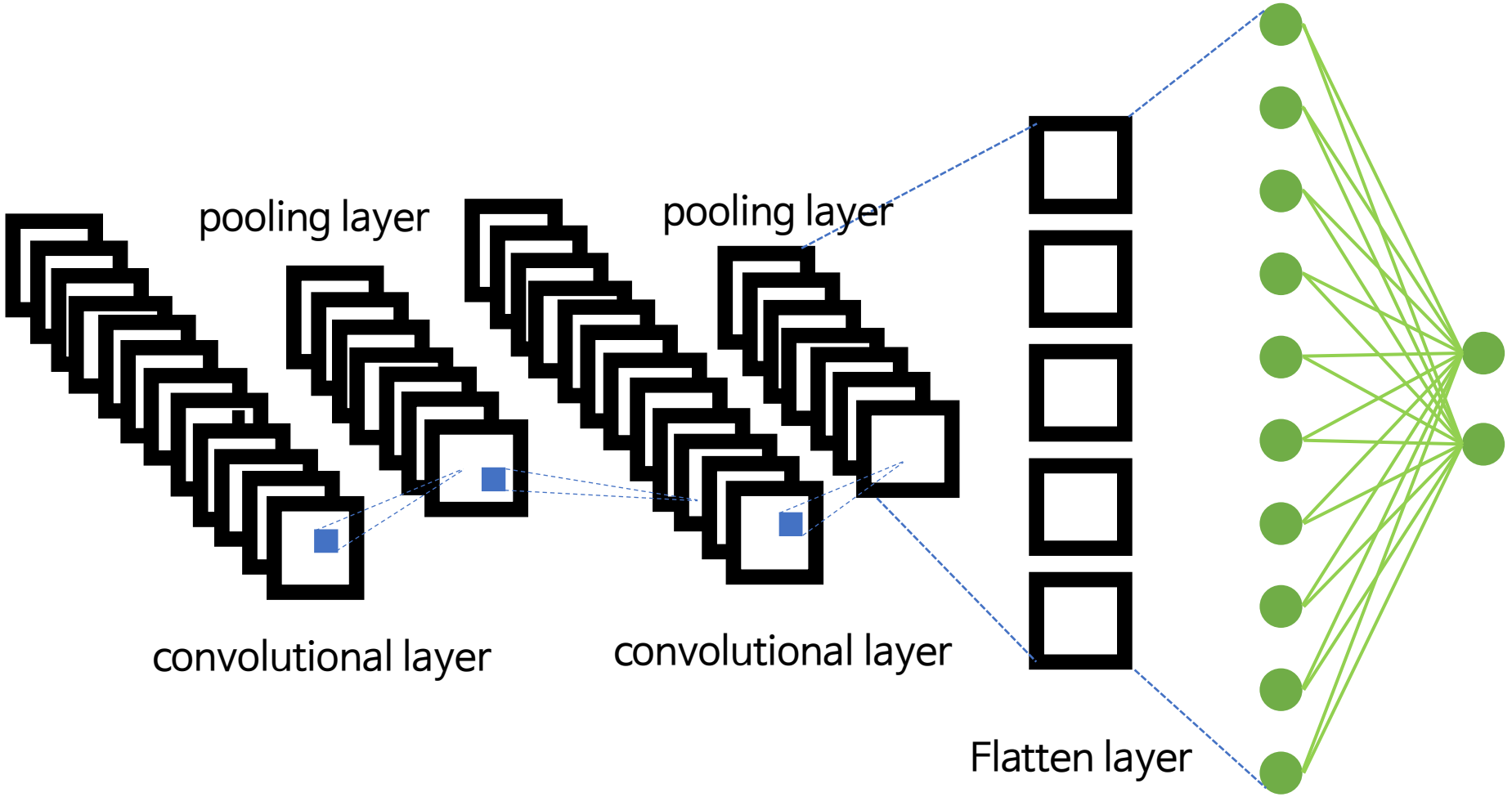
신박AI

개념적으로 한번 살펴보겠습니다

신박AI

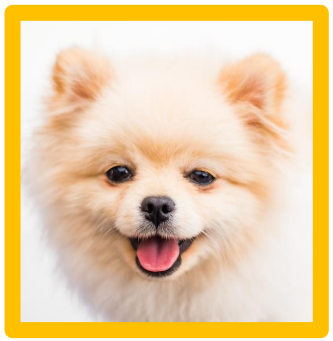
개념적 설명을 위해 이러한 CNN을 가정해봅시다

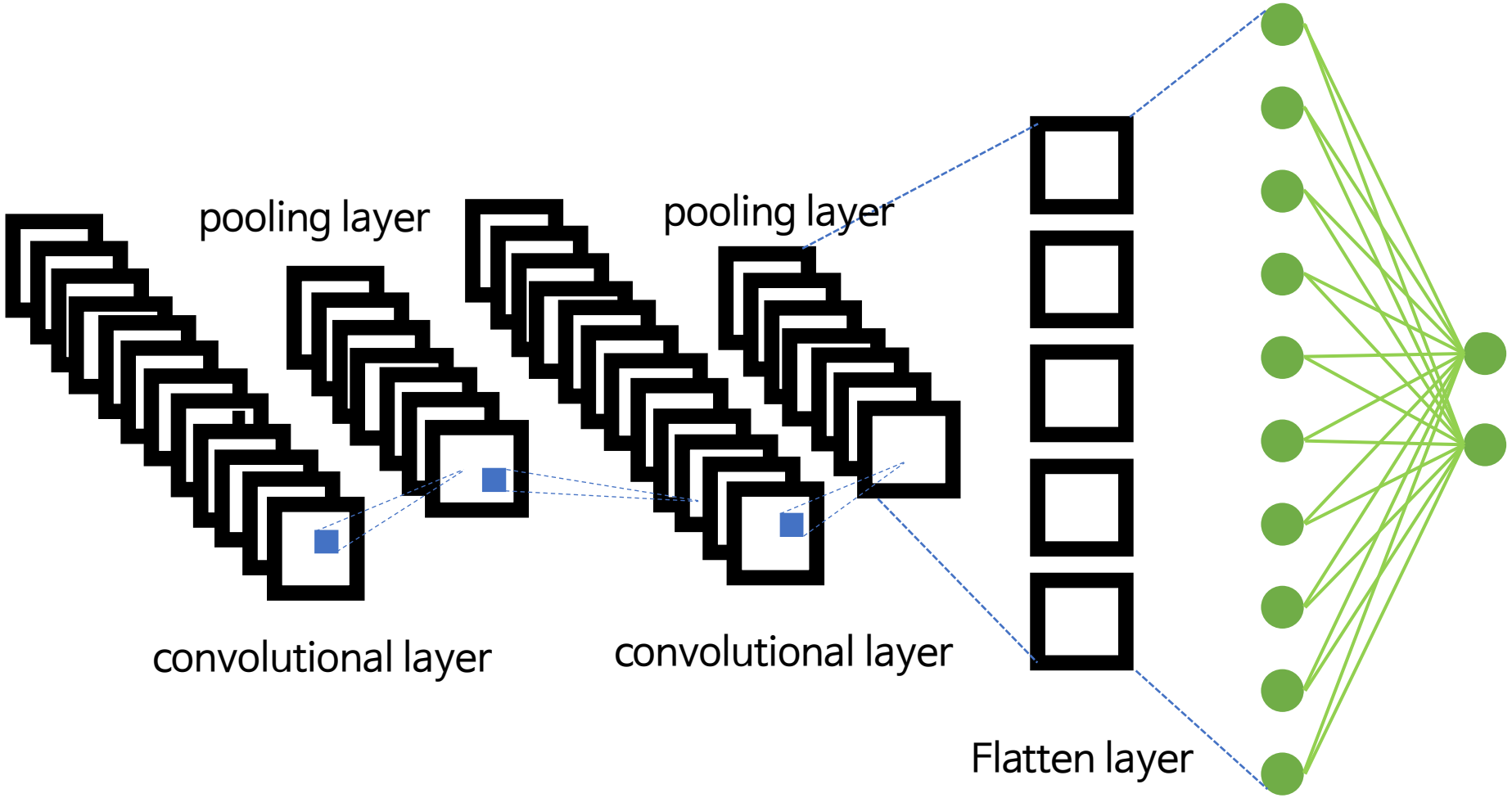




신박AI

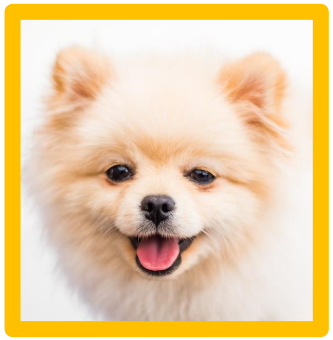
이 CNN모델은 강아지와 고양이를 구분하는 딥러닝 모델이라고

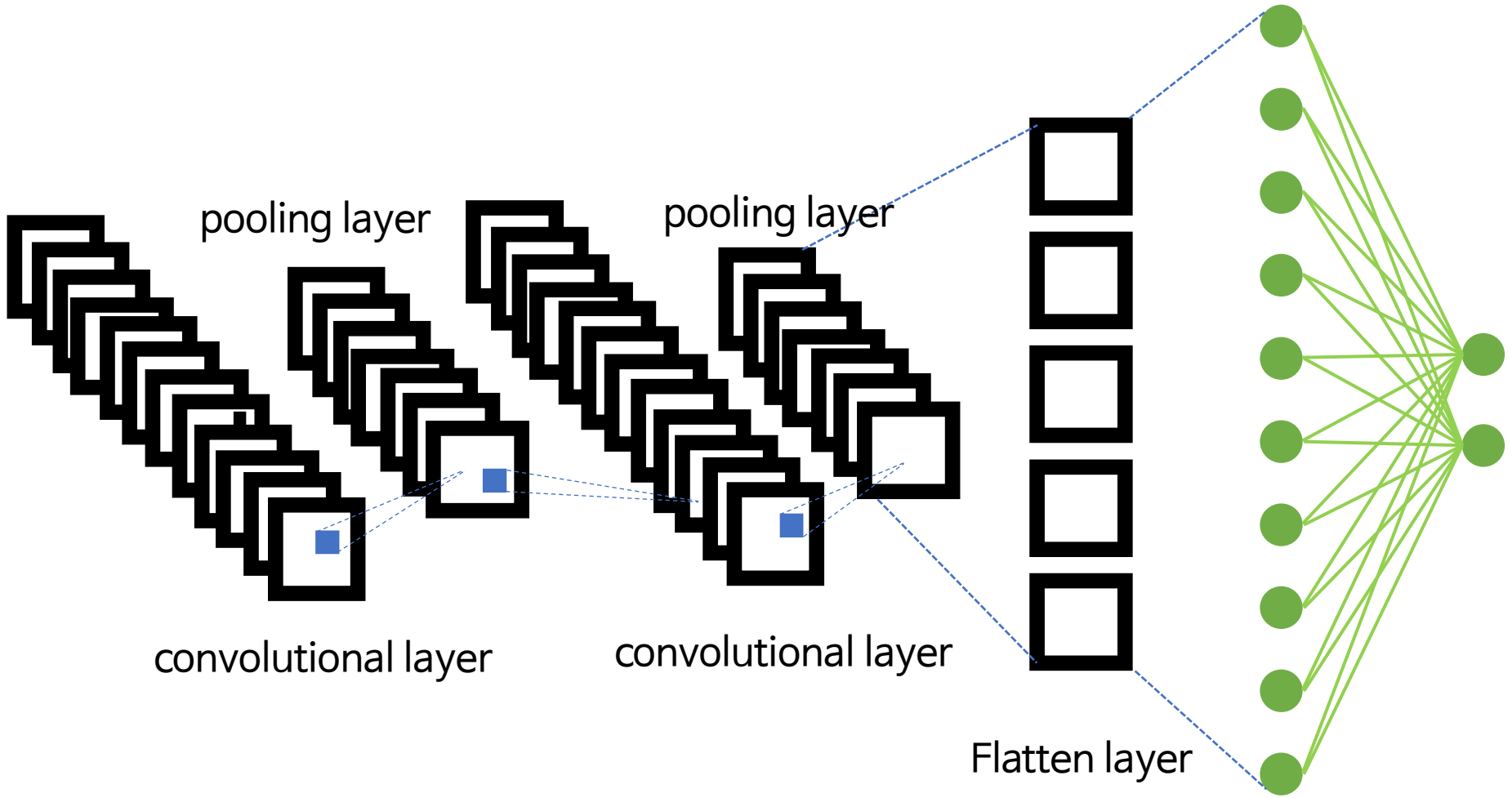




신박AI

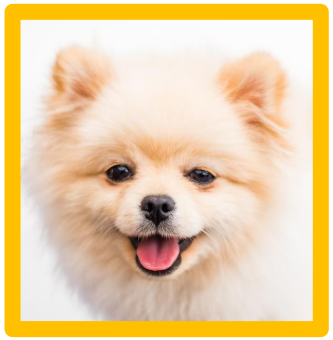
생각해봅시다

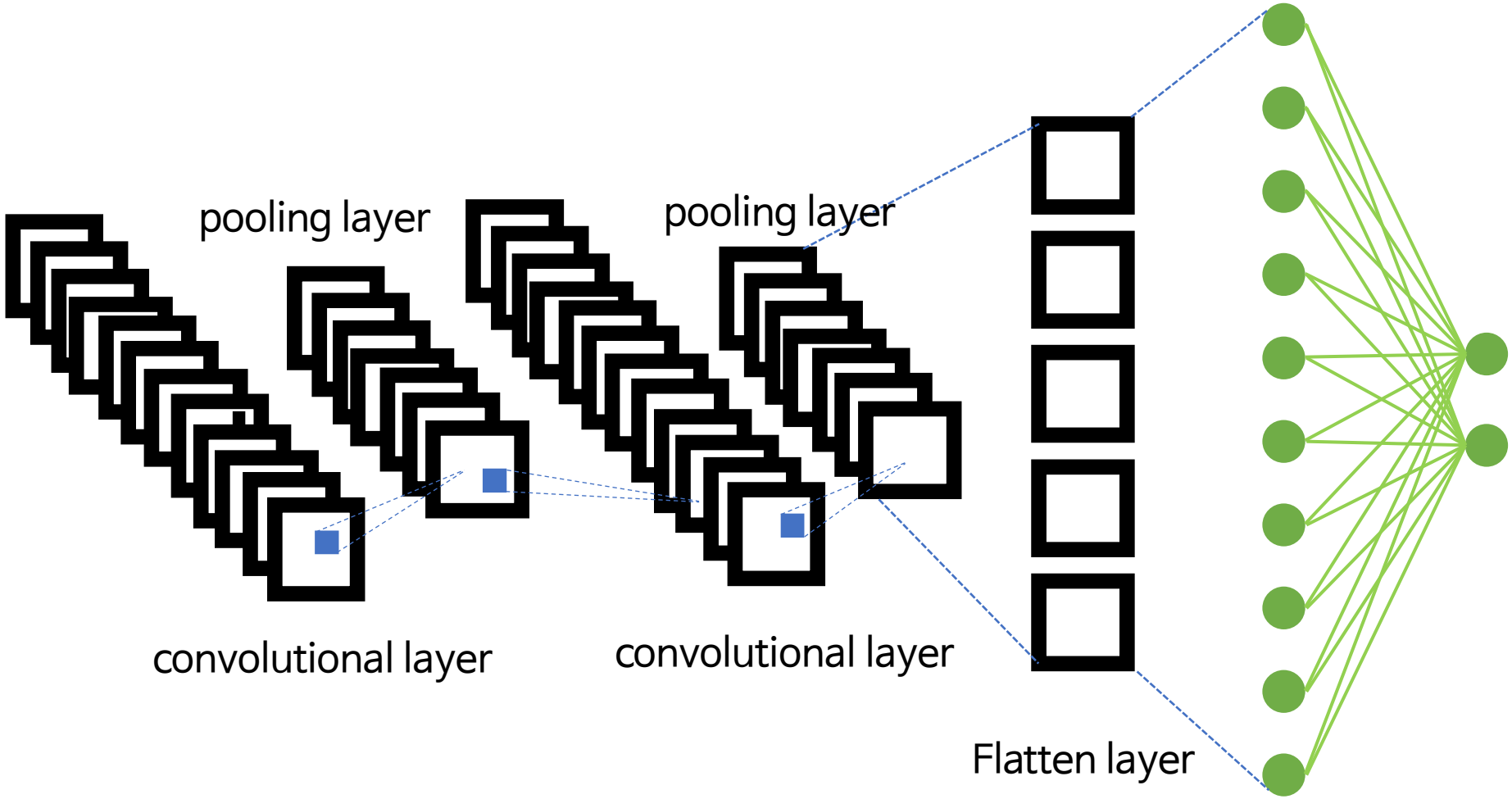



pooling layer
pooling layer
convolutional layer
convolutional layer
Flatten layer
Fully-connect
ed layer

신박AI

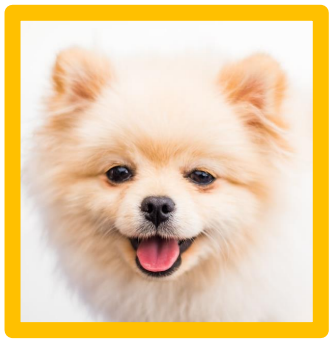
그래서 강아지가 입력값으로 들어올 경우..

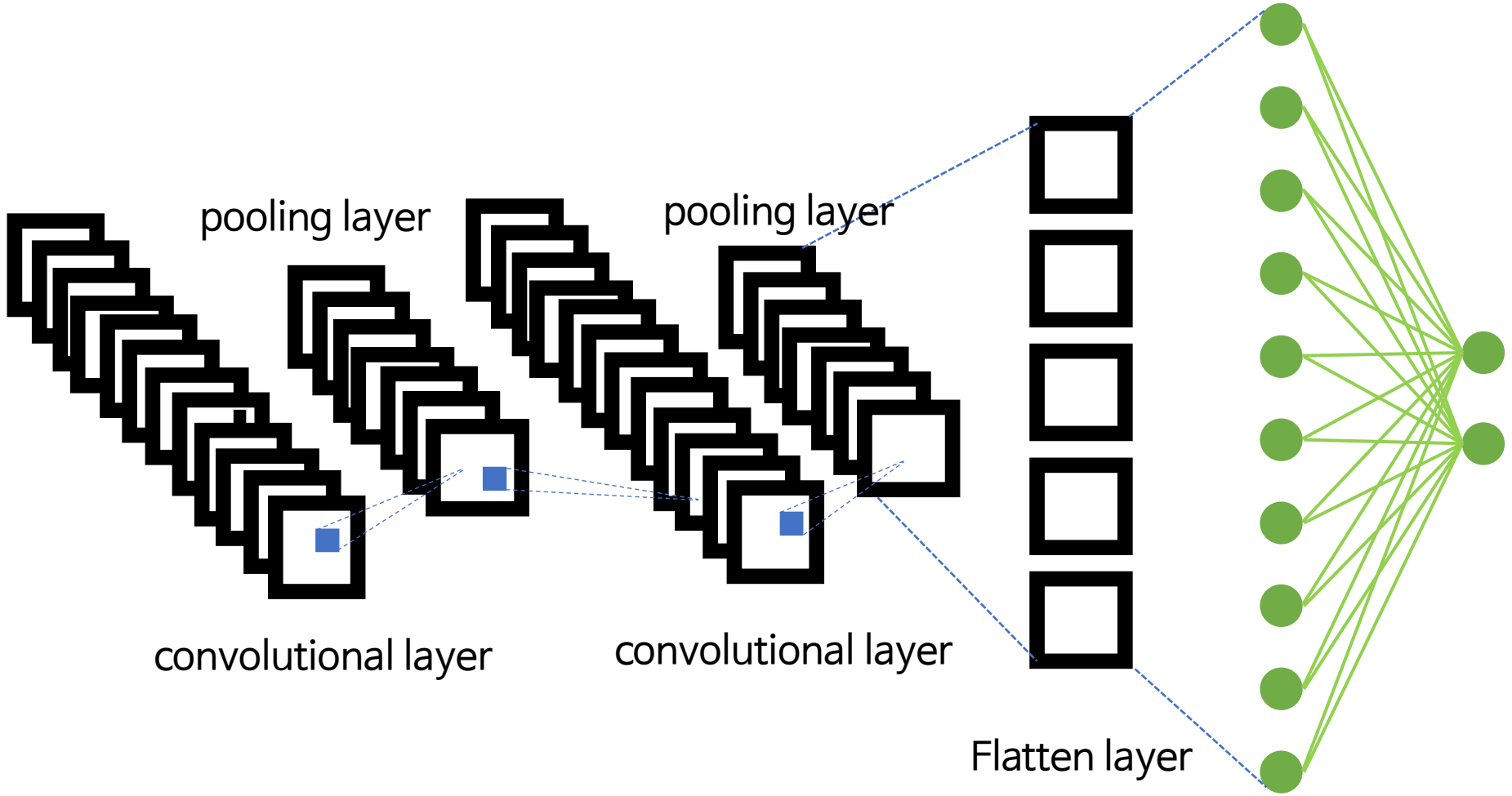




신박AI

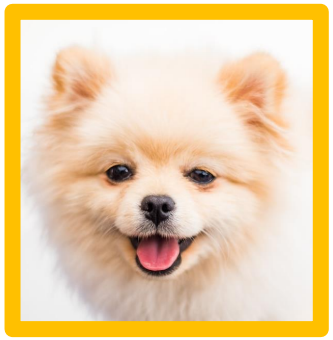
출력은 1, 0이 되고

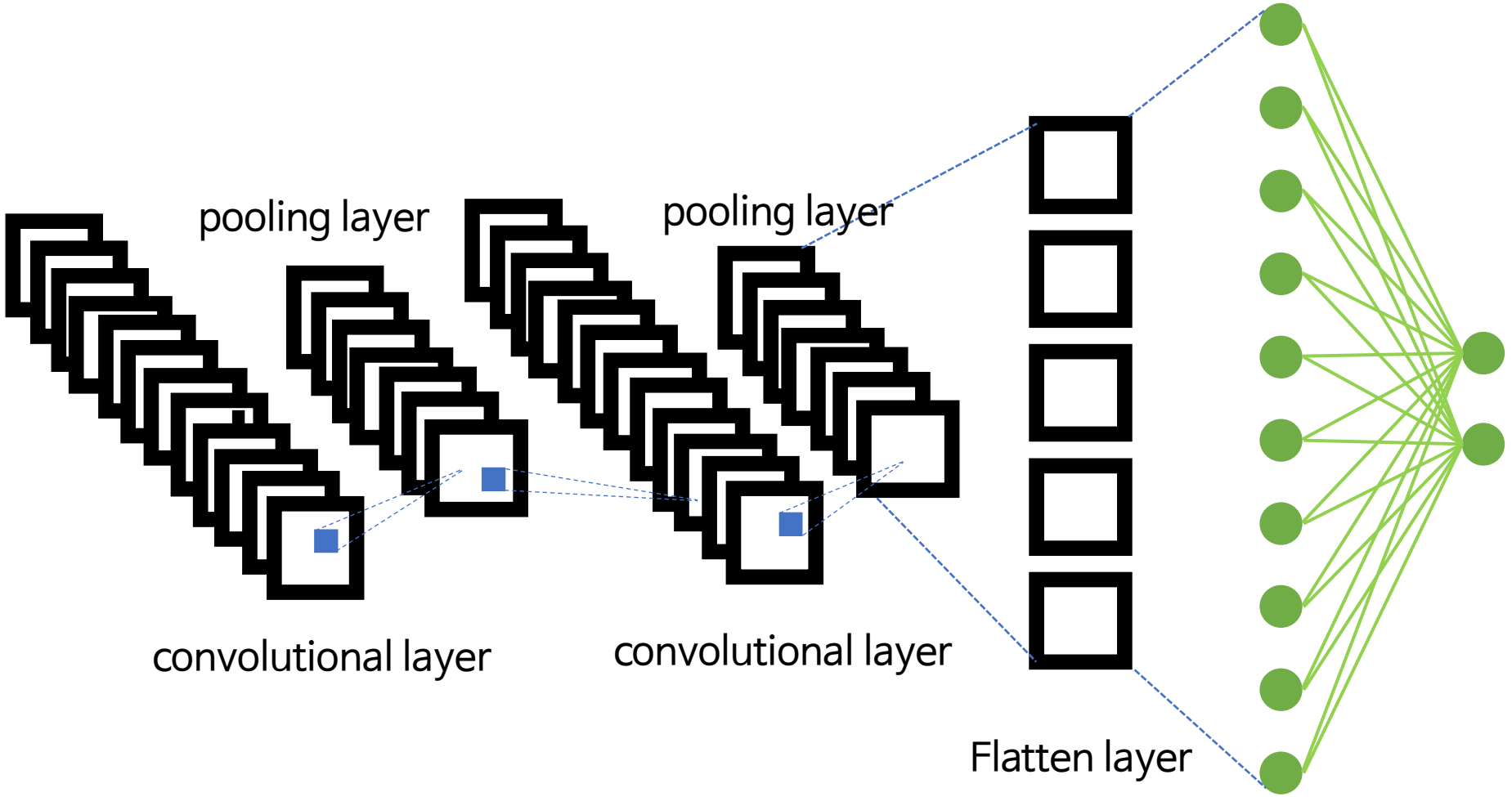




신박AI

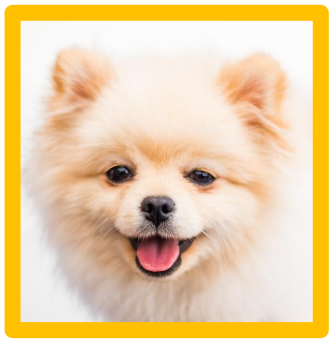
만약 고양이가 입력값으로 들어올 경우..





신박AI

출력은 0, 1이 되어야한다고 가정해봅시다



pooling layer

pooling layer

convolutional layer

convolutional layer

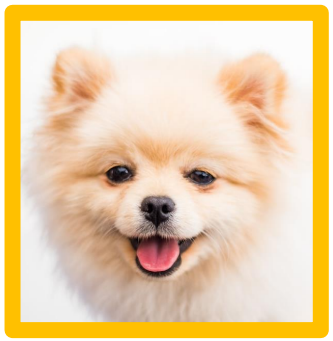
Flatten layer

Fully-connect
ed layer

0
1

신박AI

그래서 만약 강아지 사진이 들어온다면



pooling layer

pooling layer

convolutional layer

convolutional layer

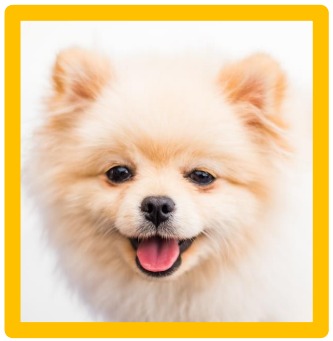
Flatten layer

Fully-connect
ed layer

1
0

신박AI

순전파 feedforward의 결과 최종 출력값이



pooling layer

pooling layer

convolutional layer

convolutional layer

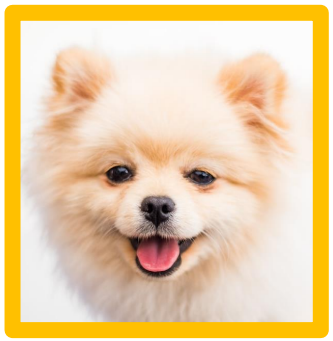
Flatten layer

Fully-connect
ed layer

1
0

신박AI

순전파 feedforward의 결과 최종 출력값이 (0.7,0)이 나왔다고 합시다



pooling layer

pooling layer

convolutional layer

convolutional layer

Flatten layer

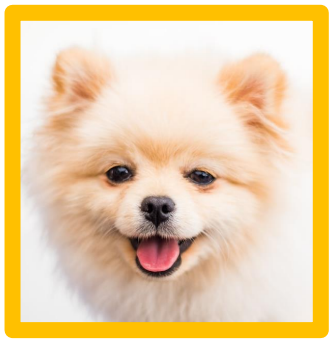
0.7
0

1
0

Fully-connect
ed layer

신박AI

그러면 오차인 0.3을 줄여나가기 위하여…



pooling layer

pooling layer

convolutional layer

convolutional layer

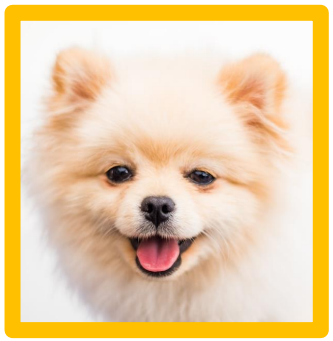
Flatten layer

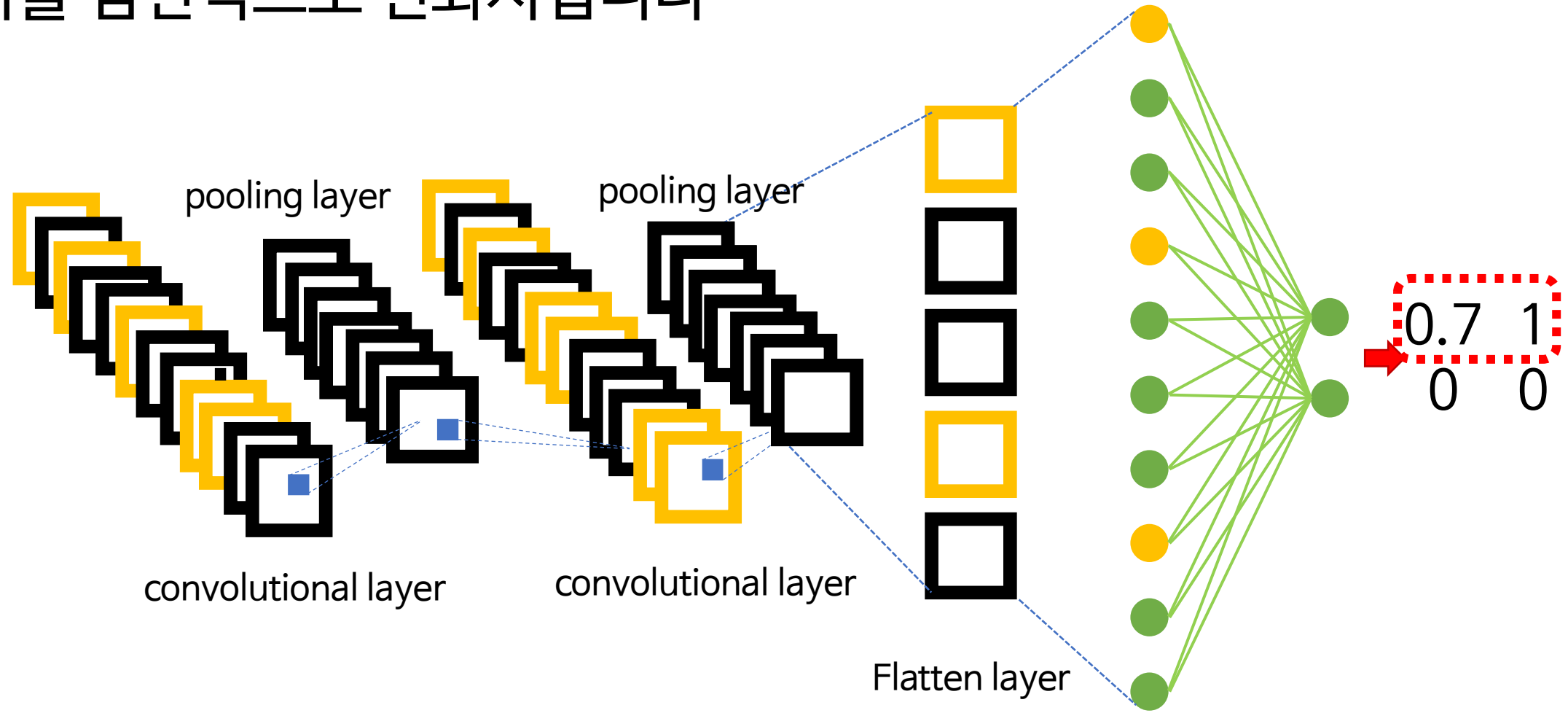
Fully-connect
ed layer

0.7 1
0 0

신박AI

강아지 이미지의 local feature들과 유사한 모습이 되어가도록 kernel들의 가중치를 점진적으로 변화시킵니다





신박AI

그리고 고양이 이미지에 대한 최종 출력값이 (0,0.8)이 나왔다고 합시다

pooling layer

pooling layer

convolutional layer

convolutional layer

Flatten layer

Fully-connect
ed layer

0 0

0.8 1

신박AI

그러면 오차인 0.2을 줄여나가기 위하여…

pooling layer

pooling layer

convolutional layer

convolutional layer

Flatten layer

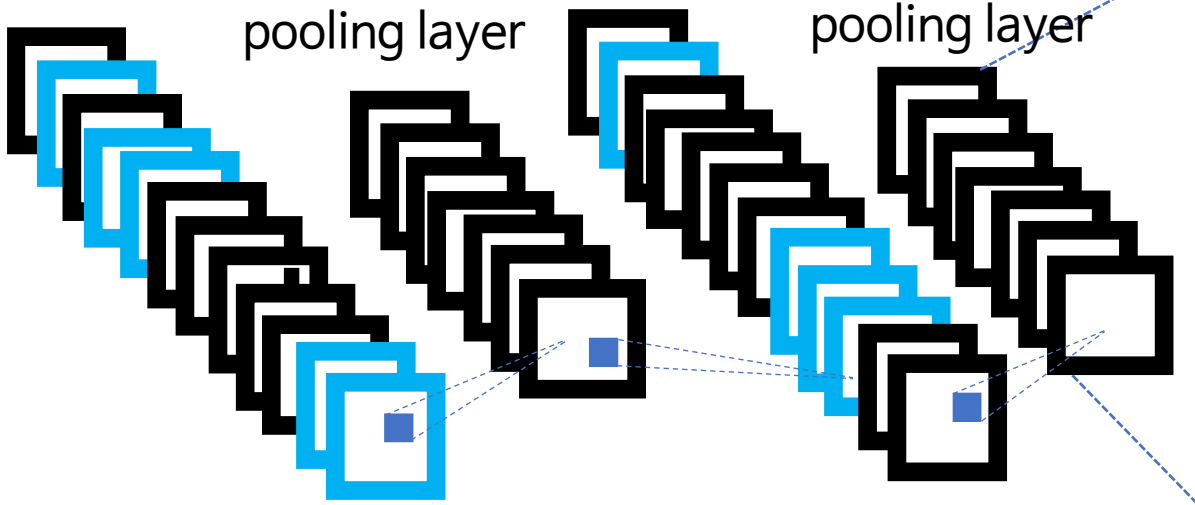
Fully-connect
ed layer

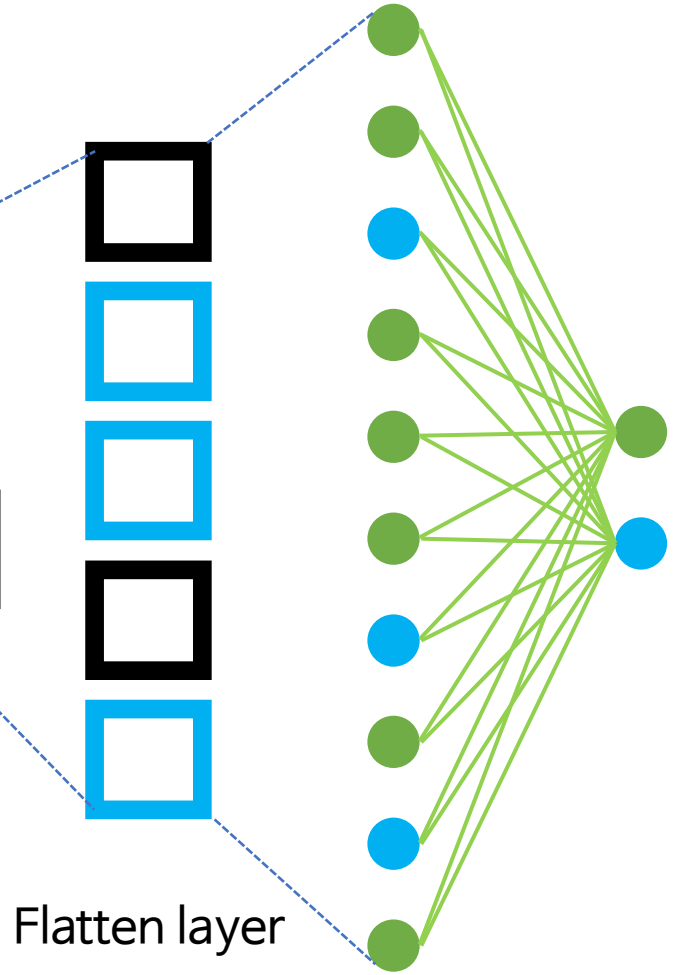
0 0

0.8 1

신박AI

고양이 이미지의 local feature들과 유사한 모습이 되도록 kernel들의 가중치를 점진적으로 변화시킵니다





신박AI

물론 kernel가중치 변화 알고리즘은 역전파와 경사하강법이 사용됩니다

pooling layer

pooling layer

convolutional layer

convolutional layer

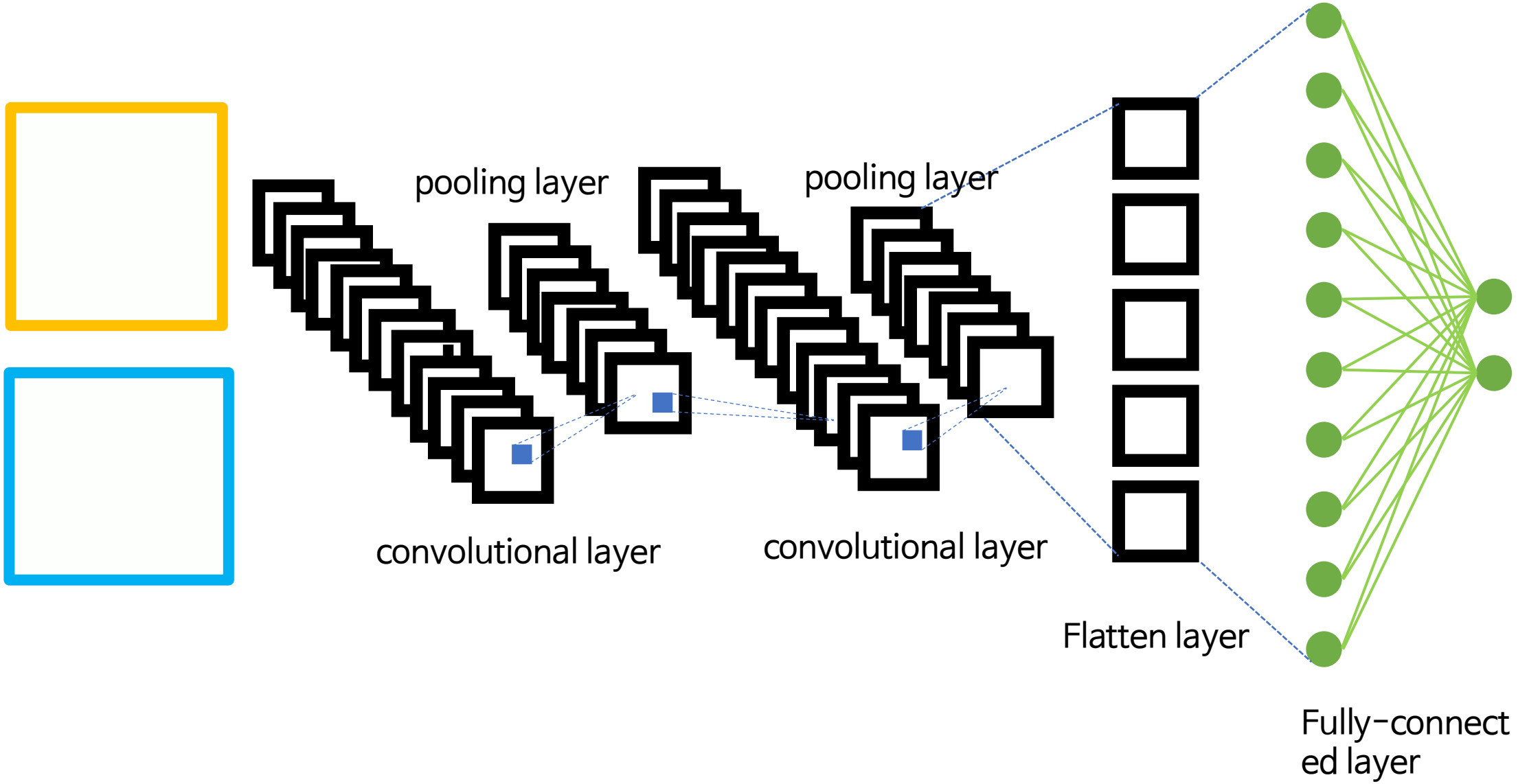
Flatten layer

Fully-connect
ed layer

0 0
0.8 1

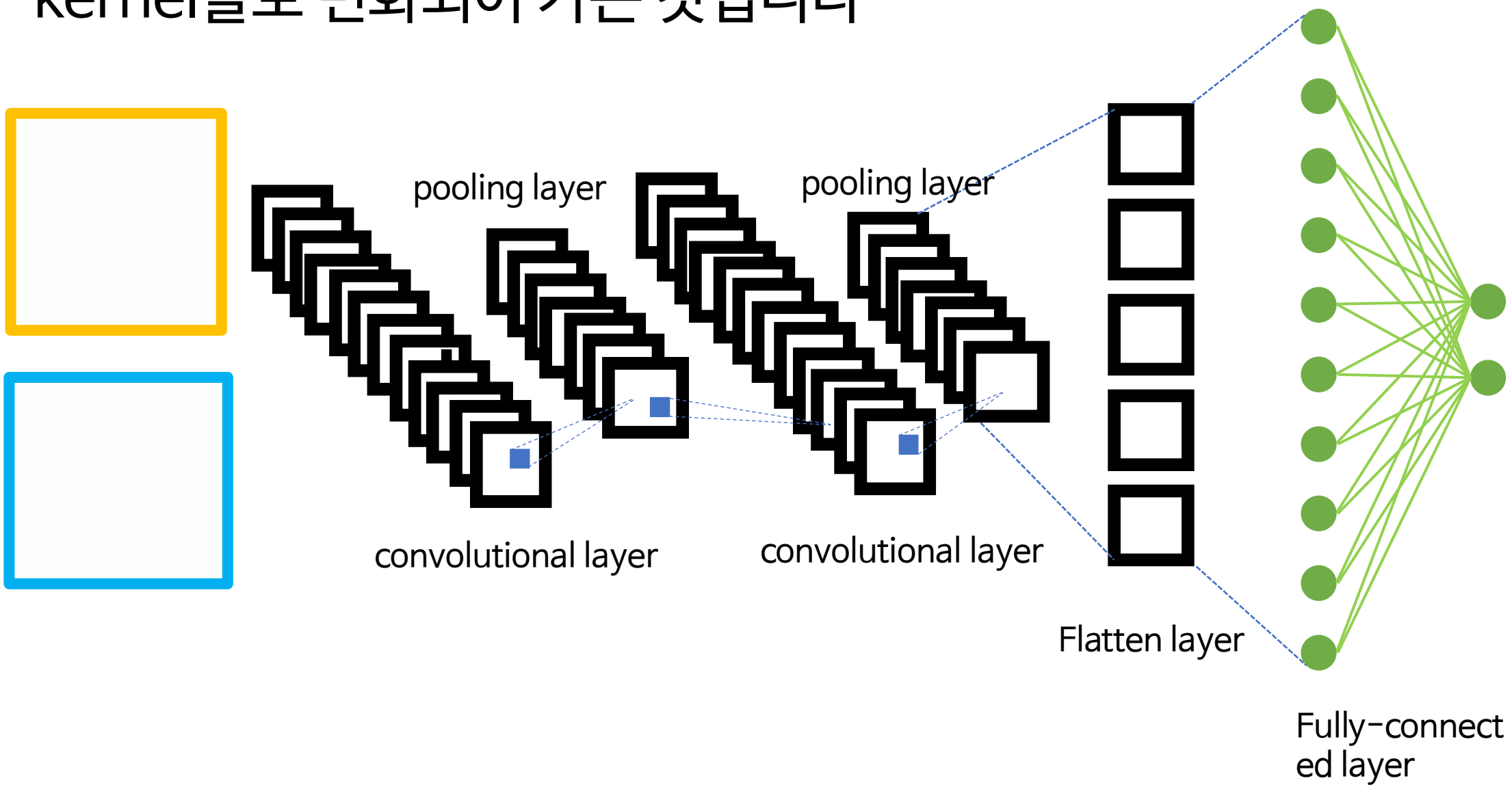
신박AI

이와 같은 과정으로 많은 데이터를 통해 반복 학습하면..



신박AI

각각의 kernel들은 강아지와 고양이의 local feature들을 잘 구분하는 kernel들로 변화되어 가는 것입니다

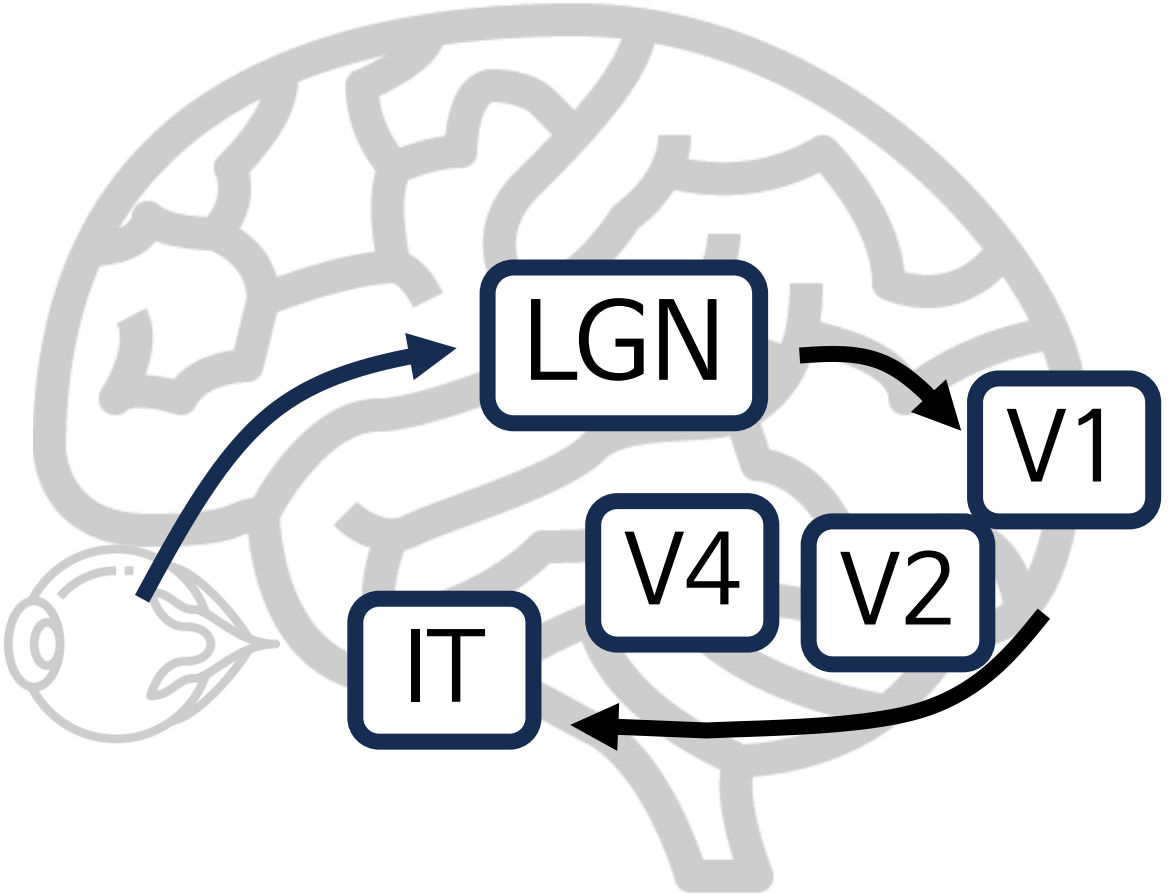


신박AI

그리고 마지막으로, CNN 모델과 인간의 시각정보 처리 과정과의 유사점에 대해 말씀드리고 싶습니다

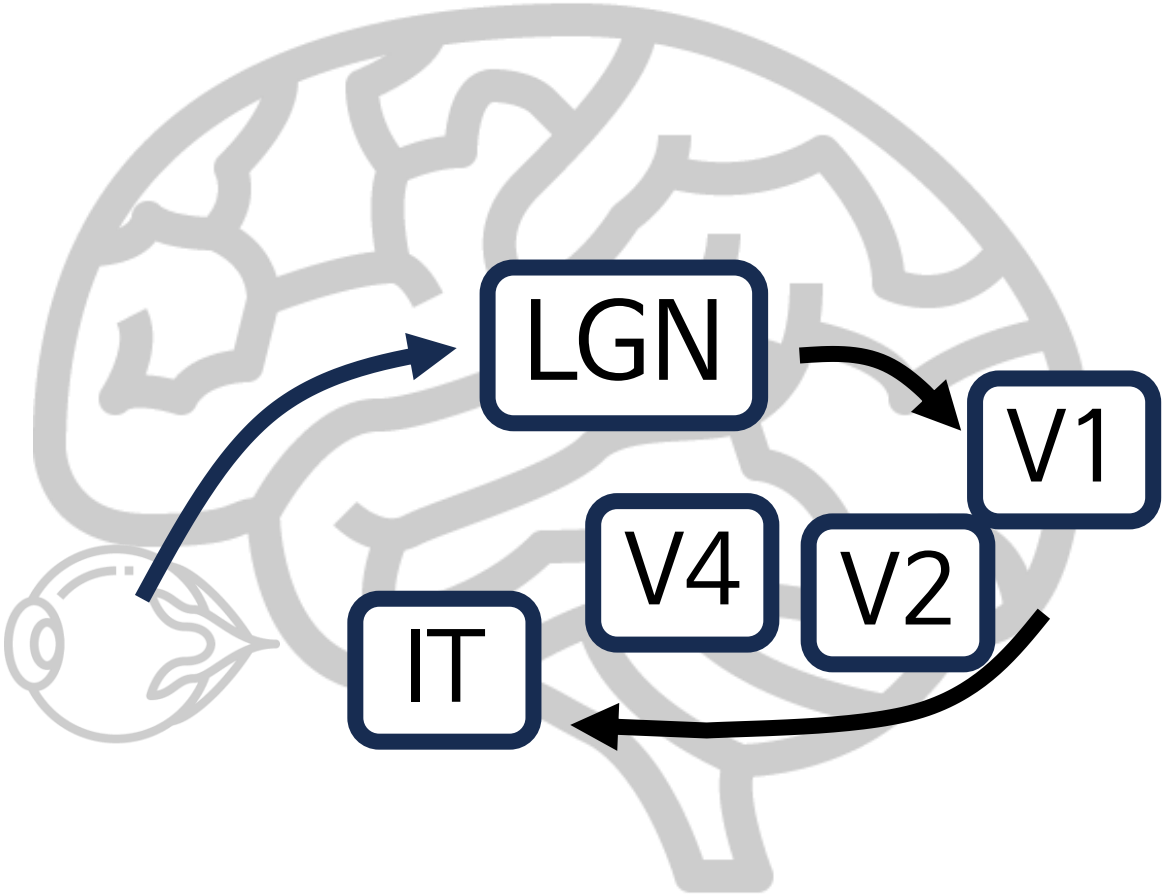
신박AI

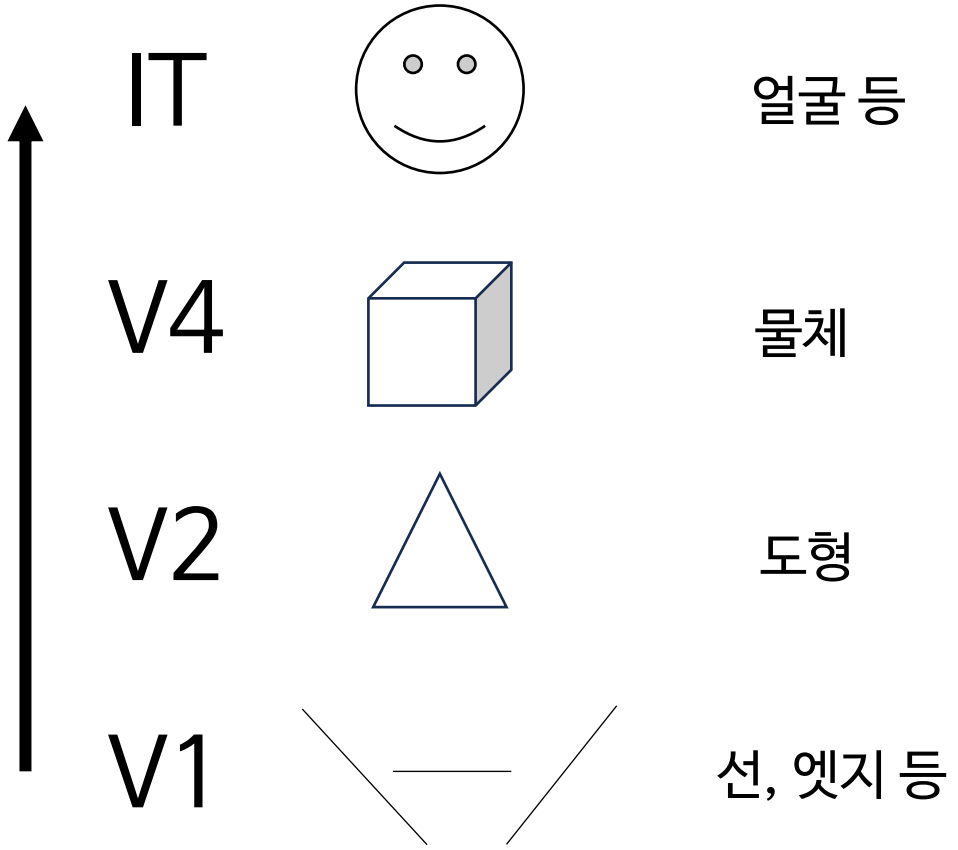
눈을 통해 들어온 시각 정보는 시상에 있는 LGN을 거쳐 일차시각 피질인 V1으로 들어가고 차례로 V2, V4, 그리고 IT지역으로 올라갑니다



신박AI

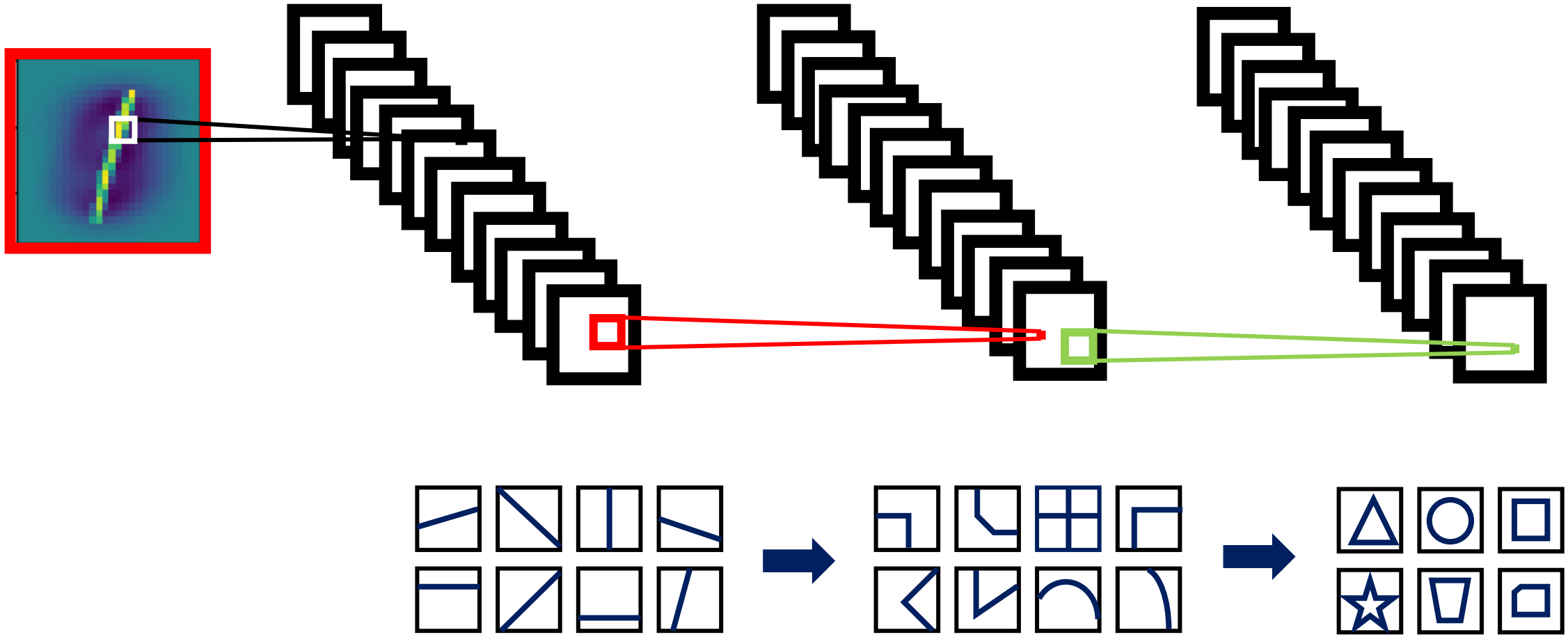
인간의 시각 정보 처리 과정은 상위 영역으로 올라갈 수록 처리하는 feature의 복잡도가 올라간다고 알려져 있습니다





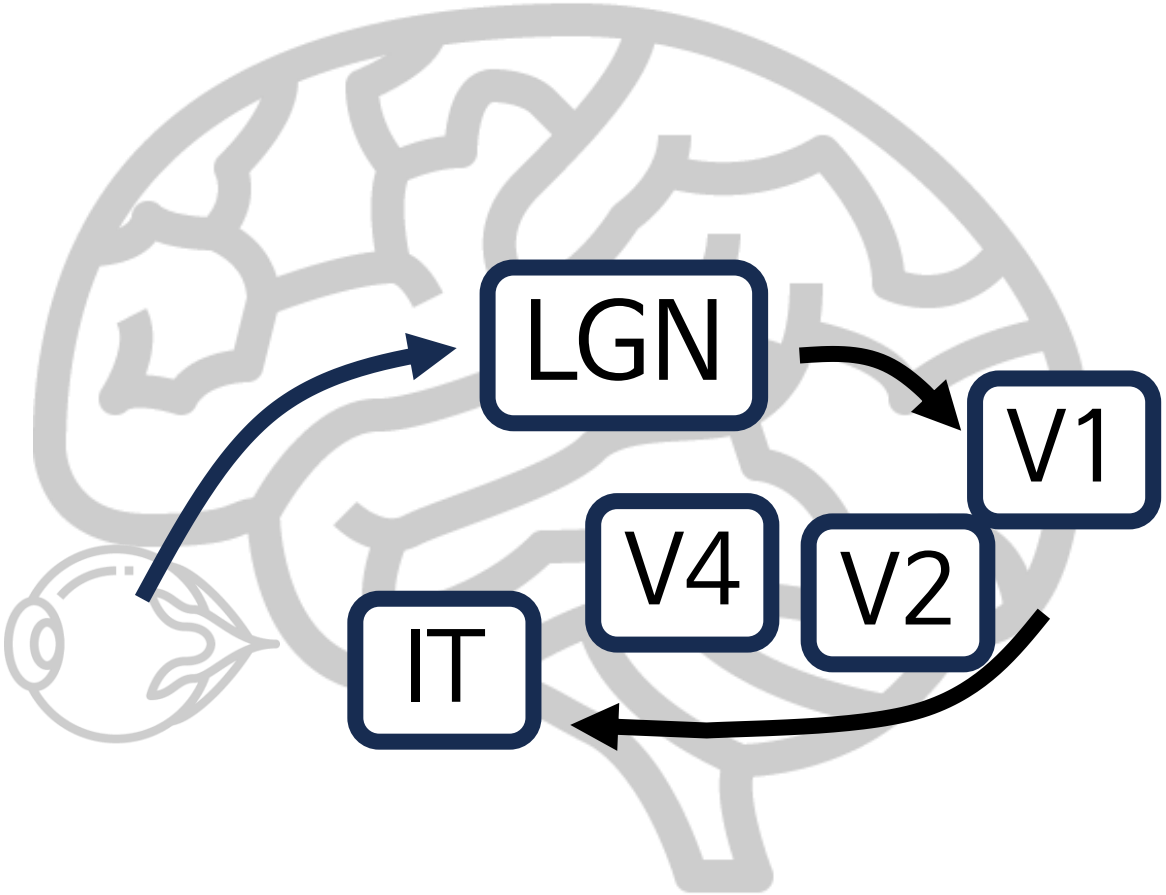
신박AI

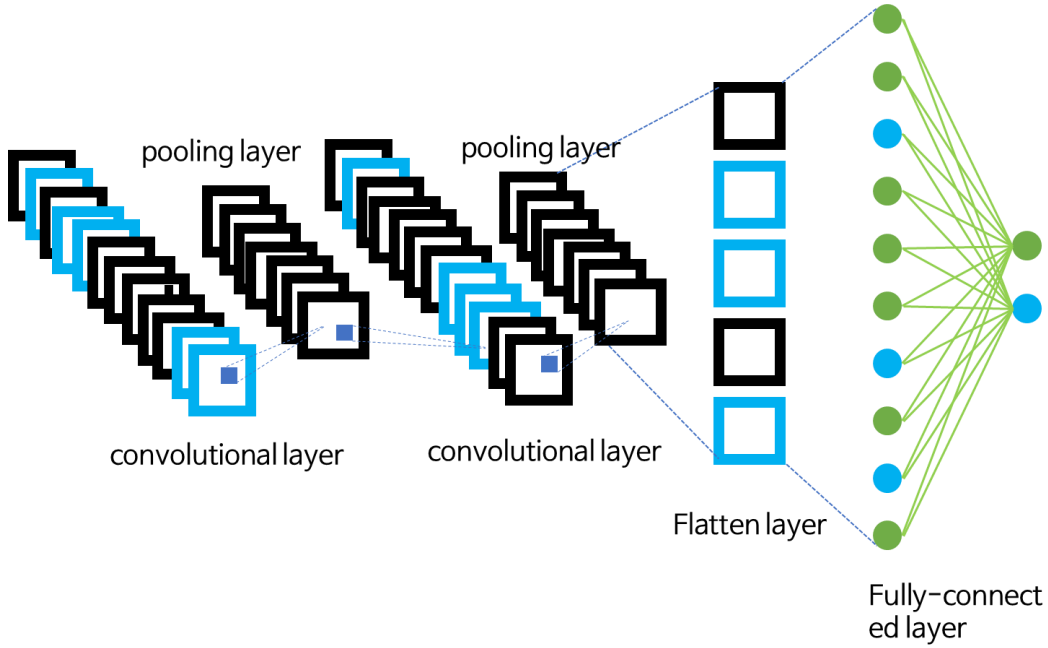
마치 CNN의 합성곱 층이 깊어 질 수록 복잡한 형태의 feature를 처리하는 특성과 아주 유사하다고 볼 수 있습니다



신박AI

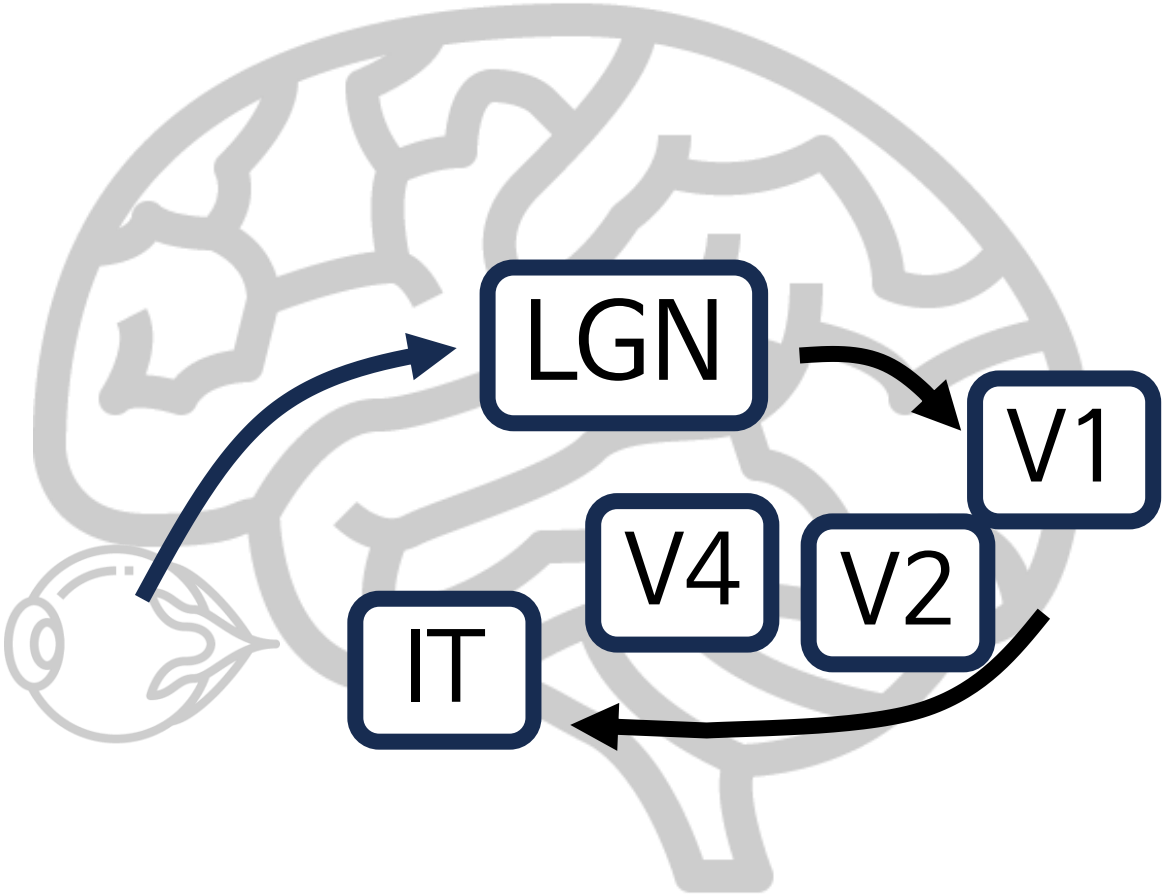
물론 CNN을 두고 사람의 뇌와 같은 메커니즘이라고는 말 할 수 없지만,

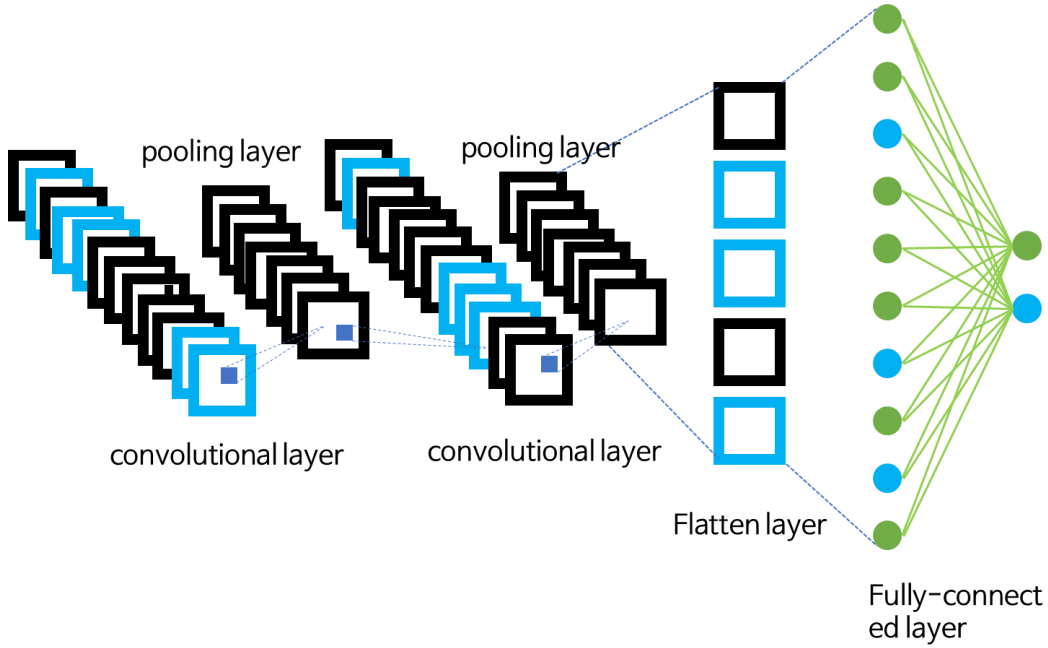




신박AI

큰 틀에서 CNN이 인간의 시각 정보 처리 과정과 어느정도 유사성이 보이는 점이 흥미롭다고 할 수 있을 것입니다





신박AI

이상으로 CNN에 대해 설명과 예제를 통하여
알아 보았습니다

신박AI

다음 영상에서는 오늘 배운 것을 바탕으로 CNN
을 구현해보는 시간을 갖도록 하겠습니다

신박AI

이 채널은 여러분의 관심과 사랑이 필요합니다

신박AI

‘좋아요’와 ‘구독’버튼은 강의 준비에 큰 힘이 됩니다!

신박AI

그리고 영상 자료를 사용하실때는
출처 '신박AI'를 밝혀주세요

신박AI

오늘 긴 시간 시청해 주셔서..

신박AI

감사합니다!

신박AI

신박AI