

# 5

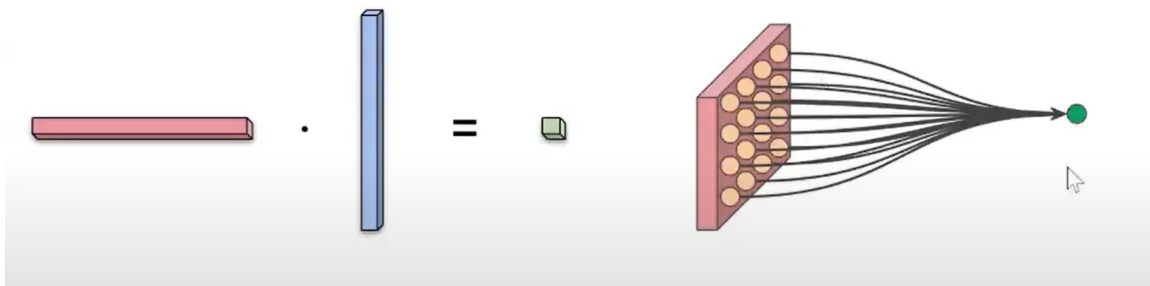
## Lecture 5

### Fully-connected layer (FC layer)

- input으로 들어온 모든 값들이 어떤 식으로든 output에 영향을 미침  
→ 모든 픽셀에 있는 값들이 각각의 가중치와 곱한 뒤 더해져 하나의 값을 내놓음

FC layer models relationships **from every** input value **to every** output value.

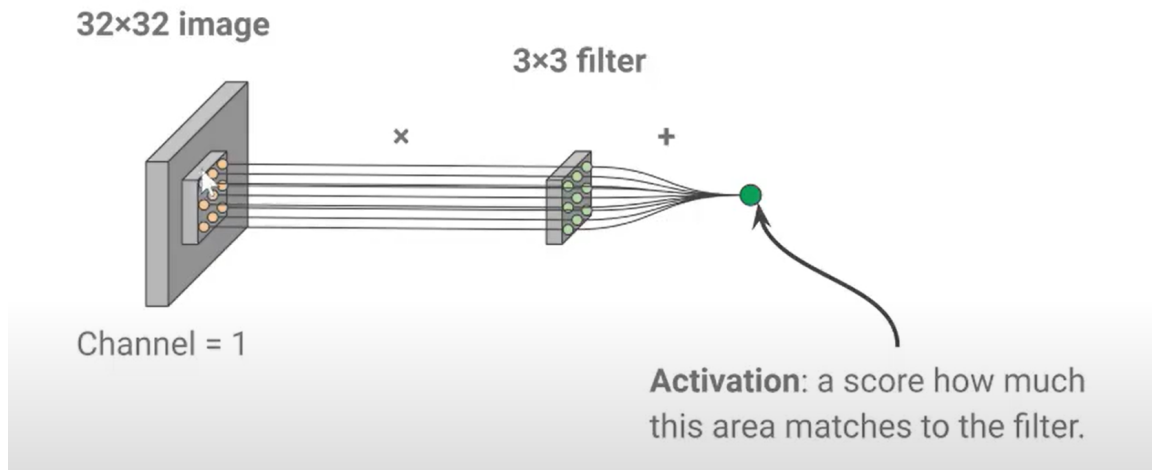
That is, it is assumed that **any output value can be affected by any input value**.



- spatial locality
  - 위치와 사이즈를 모르기 때문에, 전체 이미지에 대한 multi-sized filter들을 적용해 score를 계산해야 함.
  - threshold 이상인 score, 혹은 가장 높은 score를 가진 filter의 위치에 존재한다고 추론

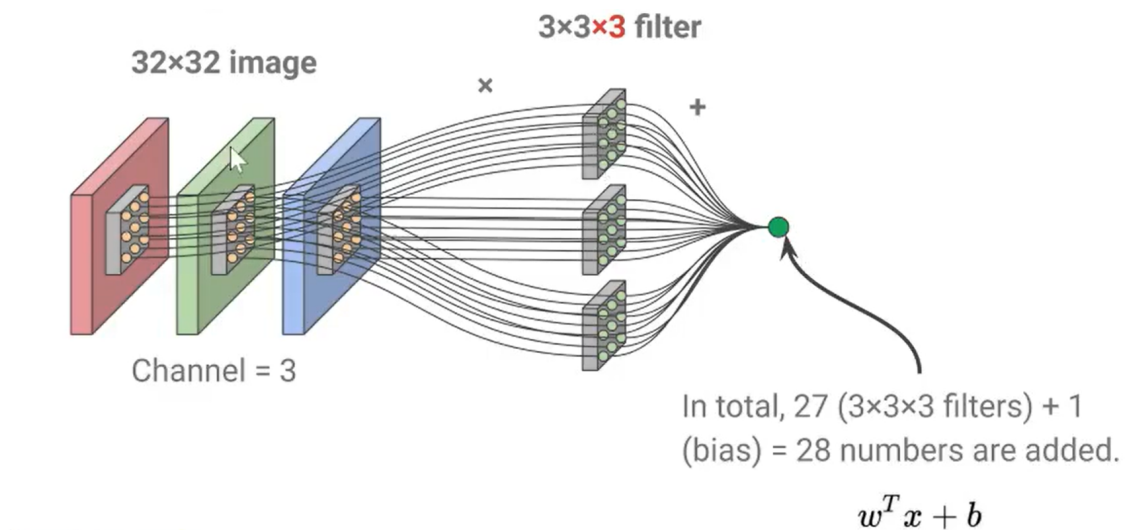
### Convolutional Layer

- monotone image



- filter를 모든 자리에 옮겨가면서 계산한다

- RGB color image



- monotone과 달리 channel만 3개가 된 것.
- 모든 자리에 대해서 이 과정이 이루어짐

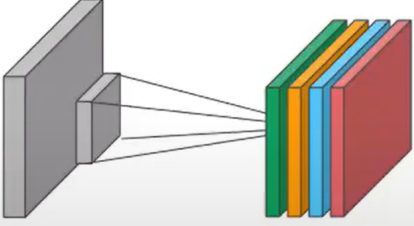
- Convolution over entire image

- 32X32 image에 3X3 filter로 convolution 진행하면 30X30 나온다.
- 32X32 image에 5X5 filter로 convolution 진행하면 28X28 나온다

→ 이런 식으로 convolution을 진행할수록 사이즈가 줄어들고, 우리가 지정한 만큼 channel 개수가 변경된다.

We may have multiple filters, or activation maps.

- Each map does exactly same operations.
- The only difference is actual weights, learned from data.

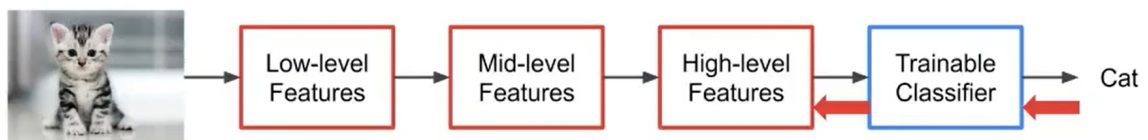


Q. For  $32 \times 32 \times 3$  image and with 4  $5 \times 5 \times 3$  filters, what is the size of output activation map?

A.  $28 \times 28 \times 4$

- 이때 4개는 channel 수

- Nested Conv-layers



- high-level feature : 모델 class와 가장 직접적인 피쳐들을 학습 (가장 중요)
- mid-level feature : high-level feature을 구분하는데 중요한 피쳐들을 학습
- low-level feature : mid-level feature을 구분하는데 필요한 피쳐들을 학습

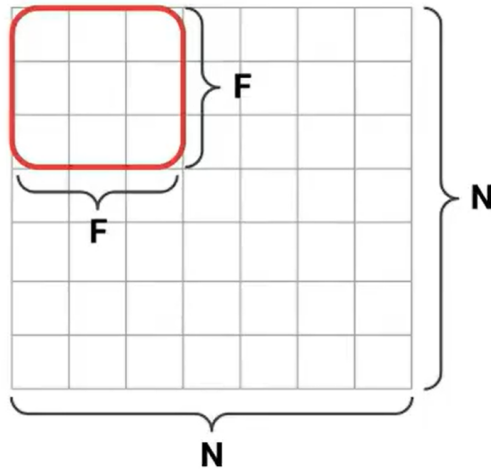
## Convolutional Layer : Deeper

- Nested Conv-layers
  - 2 Questions
    - 쌓으면 쌓을수록 이미지가 축소됨
    - resolution 높아지면 computation 복잡해짐. 어떻게 해결??

### Convolution 진행할 때 이미지 사이즈 줄어드는 것에 대한 해결 방법

- Stride

- 다음 column으로 convolution 넘어갈 때 몇몇 column 건너뛰는 것
- output size



Output size:

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

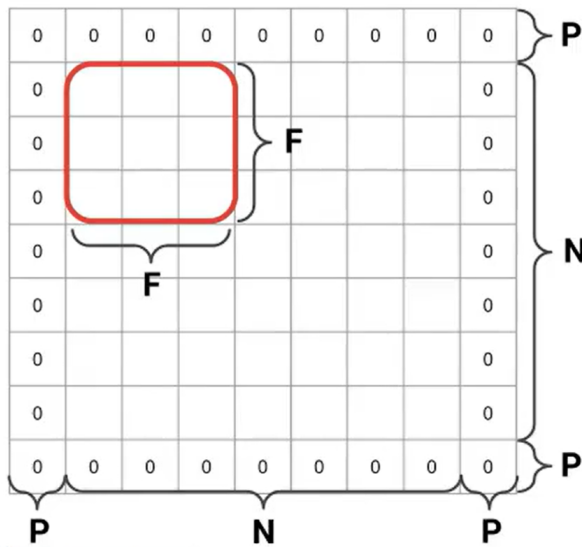
$$\text{stride}=1: (7 - 3) / 1 + 1 = 5$$

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

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

- 이때 이미지 사이즈와 필터 사이즈에 따라 계산 불가능한 stride 존재한다는 것 유의
  - padding 해줌
- Padding
  - 경계에 0을 넣어줌 (zero padding)
    - 사이즈가 input 이미지와 동일해짐
- stride와 padding을 했을 때의 output size

In practice, it is common to zero pad the border.



Output size:

$$(N - F + 2P) / \text{stride} + 1$$

With  $P = (F - 1) / 2$ , the map size will be preserved.

$$F = 3 \rightarrow P = 1$$

$$F = 5 \rightarrow P = 2$$

$$F = 7 \rightarrow P = 3$$

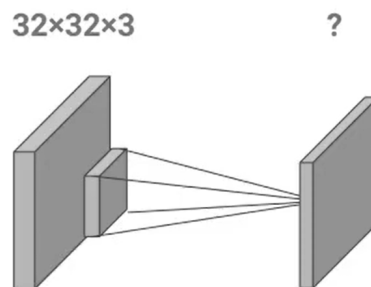
- convolutional network를 사용하는 이유

Given an input volume of  $32 \times 32 \times 3$ , apply 10  $5 \times 5$  filters with stride 1, pad 2.

What is the output size?  $32 \times 32 \times 10$

Number of parameters?  $10 (5 \times 5 \times 3 + 1) = 760$

cf. Number of parameters if fully-connected?  $32 \times 32 \times 10 \times 32 \times 32 \times 3 = 31,457,280$



- filter가 어떤 위치에 있던 간에 항상 같은 weight을 사용함
- 즉 spatial locality와 positional invariance를 가정함으로써 parameter의 수를 줄인다는 장점
  - 이때 이 가정을 충족하지 못하는 이미지 task가 있지 않을까? 일반적으로는 거의 만족함

- 정리

Given an input volume of  $W \times H \times C$ , a convolutional layer needs 4 hyperparameters:

- Number of filters  $K$
- The filter size  $F$
- The stride  $S$
- The zero padding  $P$

This will produce an output of size  $W' \times H' \times K$ , where,

- $W' = (W - F + 2P) / S + 1$
- $H' = (H - F + 2P) / S + 1$

Number of parameters:  $K(F^2C + 1)$

- Fully-connected vs Conv
  - convolutional layer은 fully-connected layer의 special case (반대도 가능! 노트 참고)

## Pooling Layer

- Pooling layer
  - downsampling하는 layer
  - learning은 이루어지지 않고, max/min 등의 정해진 일만 수행함
  - overfitting을 방지 + 메모리상 문제가 생기지 않도록 해줌
- pooling 방법
  - max pooling : max값만 갖고 옴
  - average pooling : mean 값 갖고 옴
- Pooling layer이 필수적인가?
  - 최근 논문에서는 convolutional layer의 파라미터들을 잘 조절하면 pooling layer 이 없어도 된다고 결론.

- 정리

Given an input volume of  $\mathbf{W \times H \times C}$ , a pooling layer needs 2 hyperparameters:

- The spatial extent  $\mathbf{F}$
- The stride  $\mathbf{S}$

This will produce an output of size  $\mathbf{W' \times H' \times K}$ , where,

- $\mathbf{W' = (W - F) / S + 1}$
- $\mathbf{H' = (H - F) / S + 1}$

Number of parameters:  $\mathbf{0}$  (no “learning” happens)