Convolutional Neural Network

2025 Summer DL Session 5주차

KOREA
UNIVERSITY

목차

01
들어가기 전에,,

01. 들어가기 전에,,

4주차 과제 우수자

22기 이은서 님 **22기 이세훈 님**

모두 과제하느라 수고많으셨습니다 ! :)

01. 들어가기 전에,,

4주차 과제의 의도

1 . Define Network (TwoLayerNet → MultiLayerNet: 가중치 초기화 및 감소 weight_decay)
2 . Layers(Affine, Relu, Softmax) + Sigmoid
3 . Optimizers (SGD, Momentum, AdaGrad) + Adam
4 . Regularization (BatchNormalization, Dropout)
5 . MNIST 데이터에 정규화 기법 적용별 효과 비교 (정규화 기법 적용 전 vs 후) (정규화 기법별 비교)

5주차 학습 목표

MLP와 CNN의 차이를 알고, 이미지 처리에 강점을 가지는 CNN이 무엇인지 알아보자.

02
MLP vs CNN

02. MLP vs CNN

Convolutional Neural Network(CNN)

Image Recognition Task에 사용
특히, Image Classification에 많이 쓰임.

CNN의 가장 큰 특징

1. **Weight Sharing** by Convolution

Input image 사이즈가 너무 큰 경우 weight 수가 너무 많이 필요함. Input dimension이 너무 큰 경우 weight sharing을 하면 weight의 수를 어느정도 고정시킬 수 있음.

2. Solve **Translation invariance** by Max pooling

그림을 이동시켜도 실제로는 같은 그림.
하지만 컴퓨터 데이터는 픽셀별로 데이터를 인식하기에, 회정이나 변형이 조금만 가해져도 컴퓨터는 전혀 다른 이미지로 해석

02. MLP vs CNN

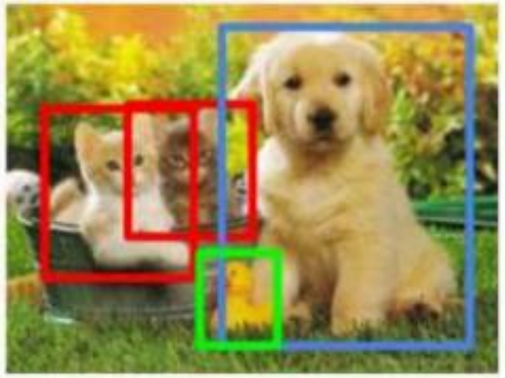
Convolutional Neural Network(CNN)

Classification

CAT

requires invariance

Object Detection



CAT, DOG, DUCK

requires equvariance

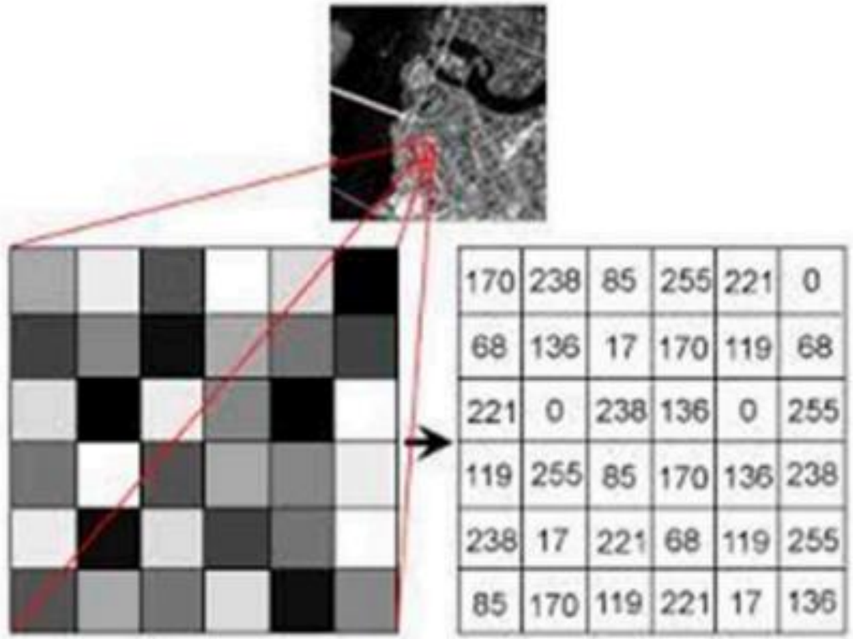
Instance Segmentation

CAT, DOG, DUCK

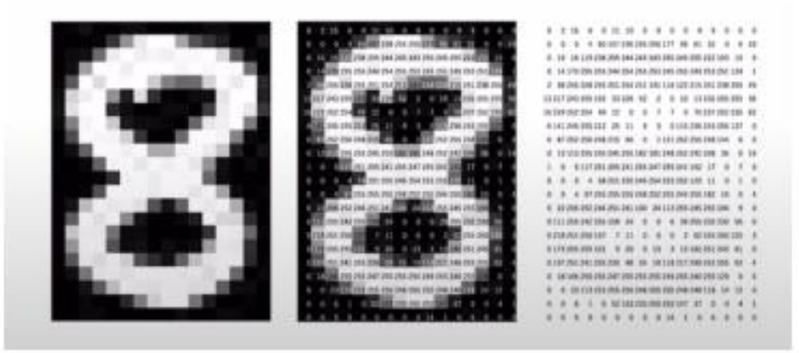
requires equvariance

02. MLP vs CNN

Image data



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

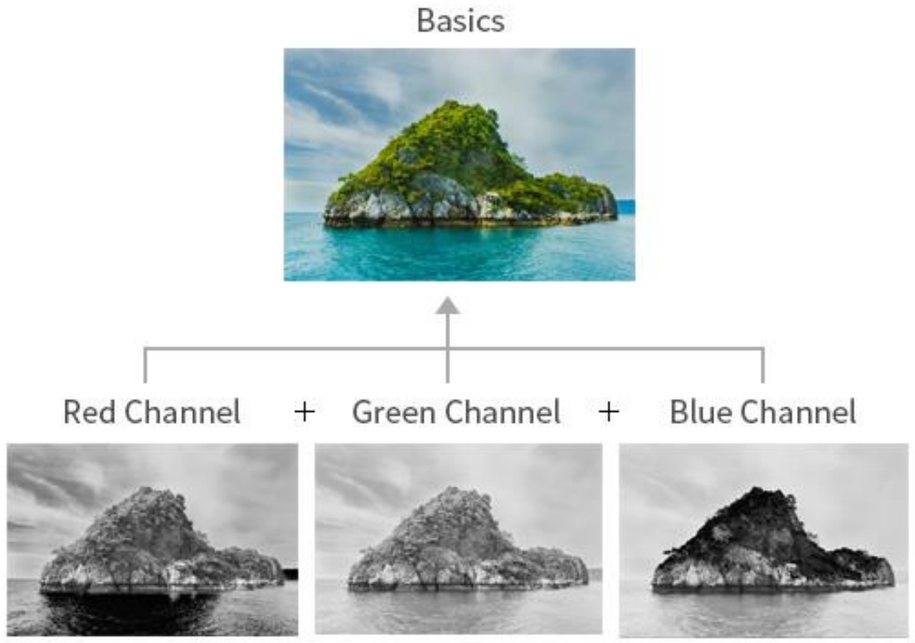


픽셀로 따지면 가로 W개, 높이가 H개

- #(pixel) = H * W
- 픽셀 수 ↑, 화질(resolution) ↑
- 픽셀 값 = 밝기 값
- 0~255 사이, 0: 검정 255: 하양

02. MLP vs CNN

Image data



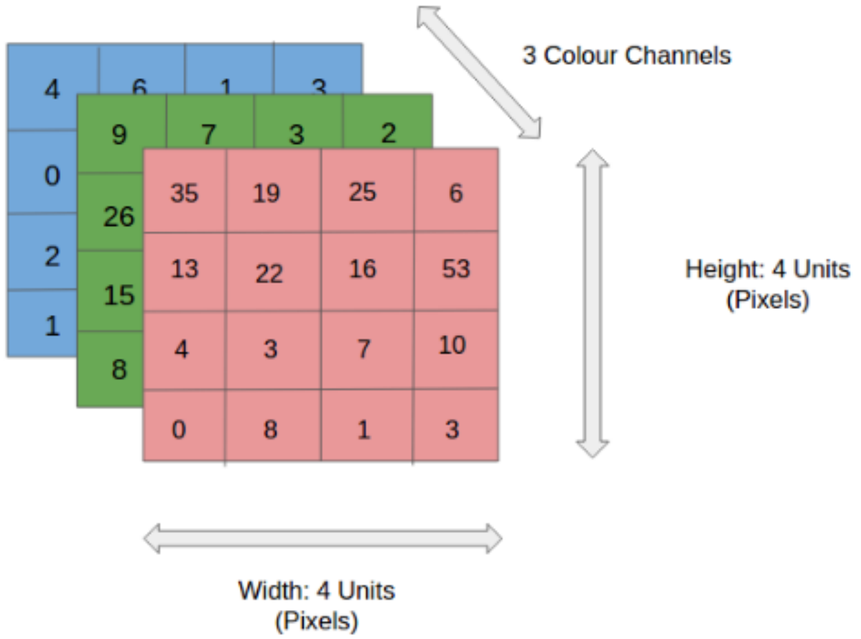
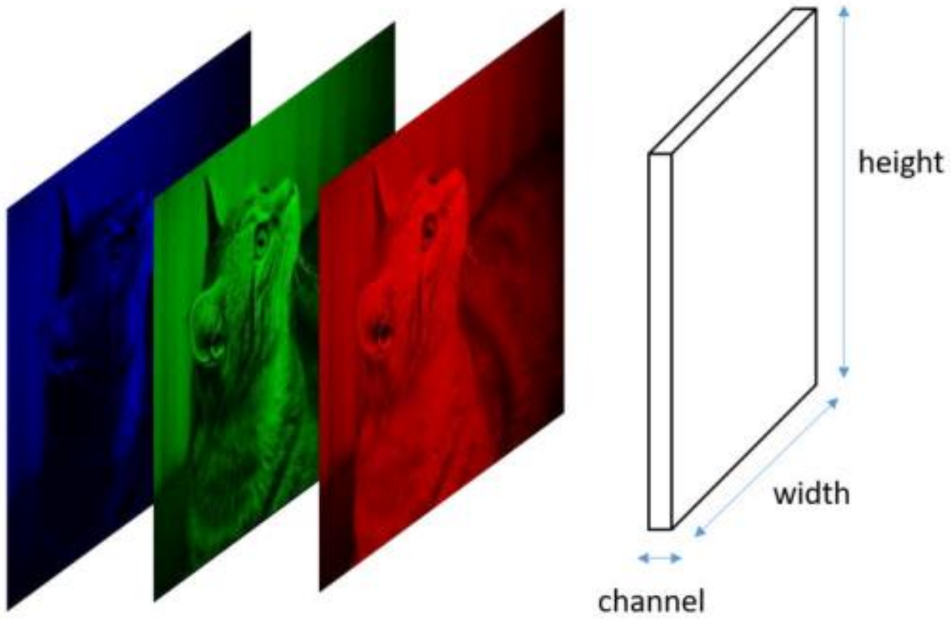


Image data는 각 pixel당 3개의 RGB값을 가지고 있다

Image data



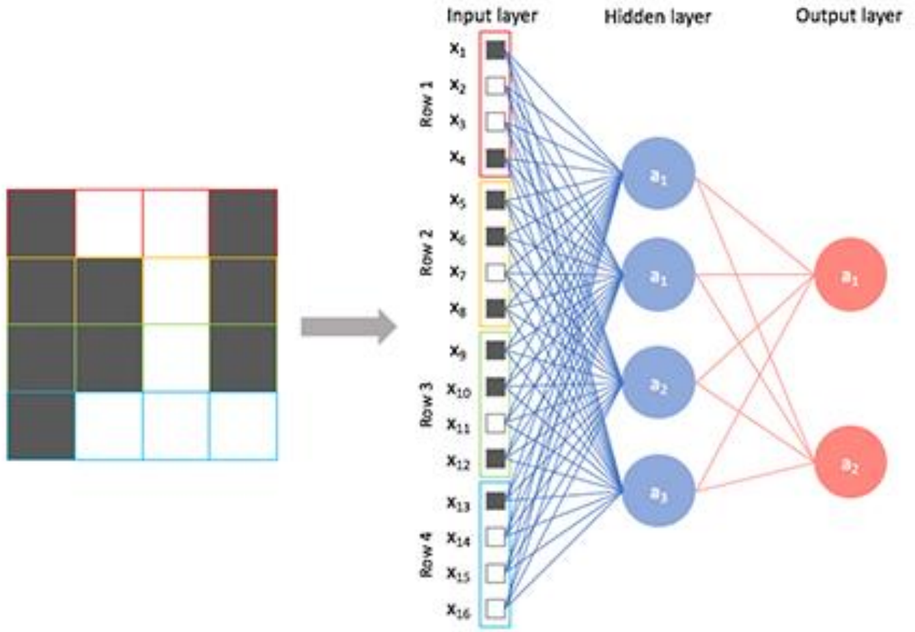
이미지 텐서의 크기

H * W * C

#(Channel) = 3 : RGB 채널 이미지
#(Channel) = 1 : 흑백 이미지

02. MLP vs CNN

Image data



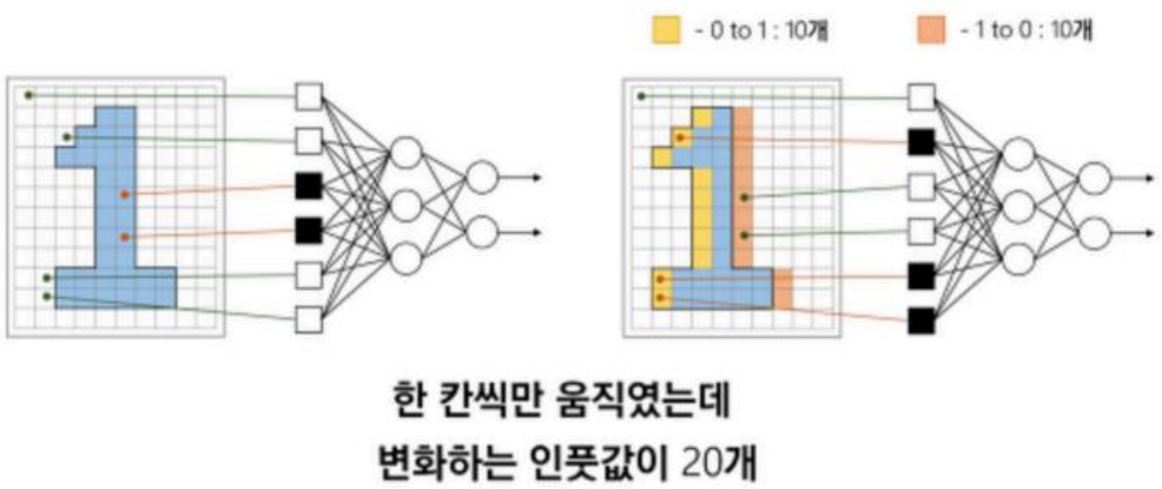
Why difficult?

각 텐서(RGB와 Pixel)의 위치가
곧 feature로 다루어지기 때문

기존의 MLP(Fully Connected Layer)로
처리하기엔 parameter수 가 기하급수적으로
커지게 된다.

또한 각 Pixel의 주변부 정보와의 관련성을
학습시키기 어렵다.

Image data

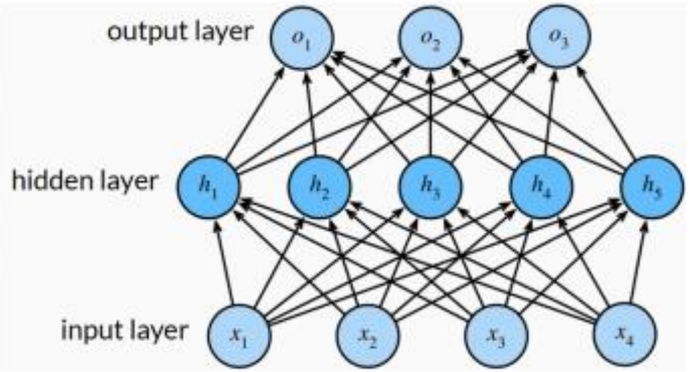


Why difficult?

이미지가 조금 차이 나더라도 input의 변화가 매우 크게 발생함.

02. MLP vs CNN

$$h_i = f\left(\sum_{j=1}^{m}(w_{ij} \cdot x_j) + b_i\right)$$



02. MLP vs CNN

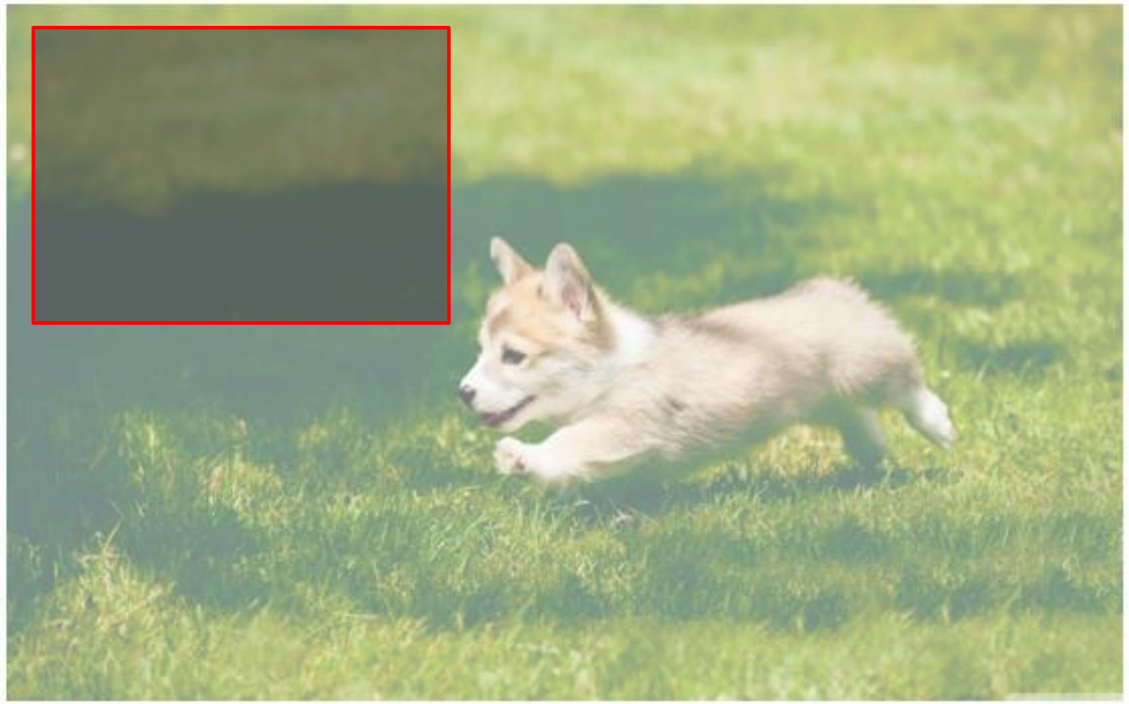
Locality principle

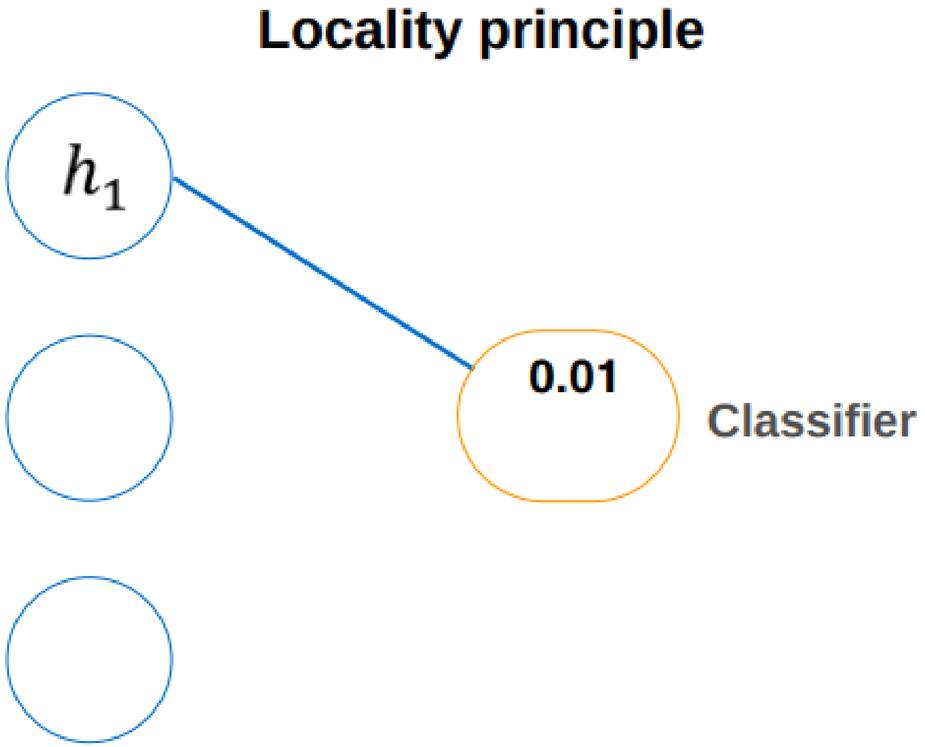
1. 먼 지역의 내용(특징)은 무관하다.
2. 이미지의 특정 부분을 인식할 때,
   그 부분과 멀리 떨어진 다른 지역의 정보는 중요하지 않다고 간주된다.

이미지를 스캔하면서 각 Image patch(조각)에 점수를 매긴다.
= 이미지의 구조는 국소적인 정보를 기반으로 처리될 수 있다.
= Local Info(특정 영역의 특징)을 활용한 효율적인 학습을 목표.

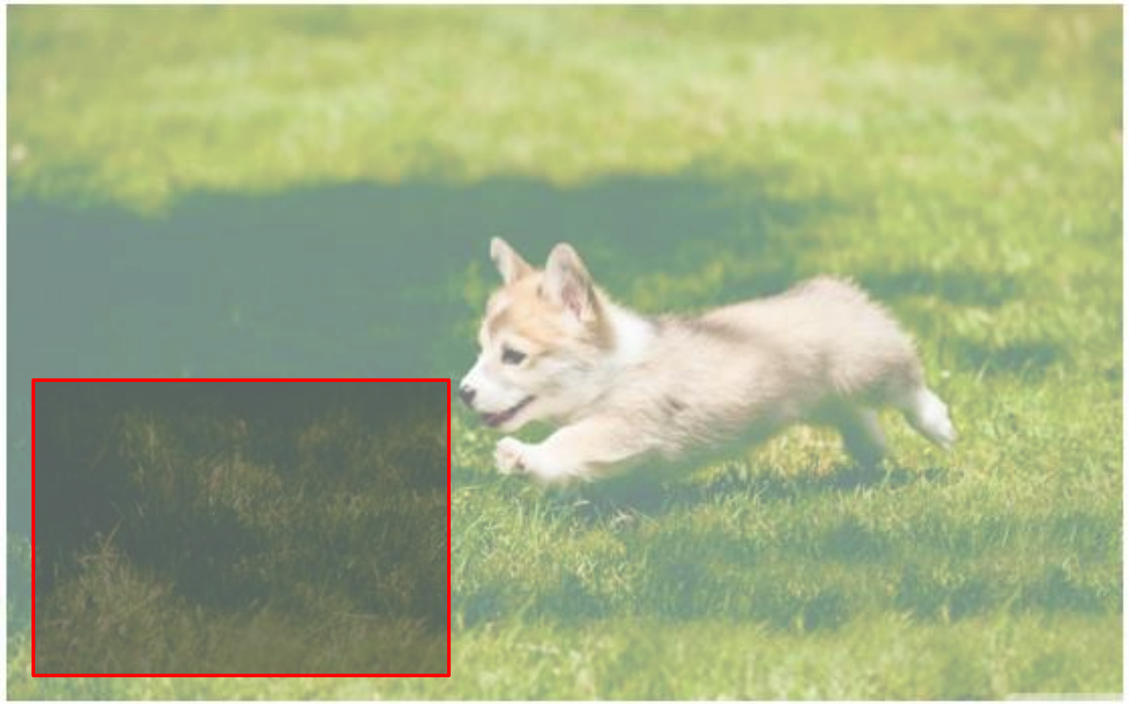
➡ **CNN(Convolutional Neural Network)**은 Locality Principle에 기반하여 이미지를 처리!

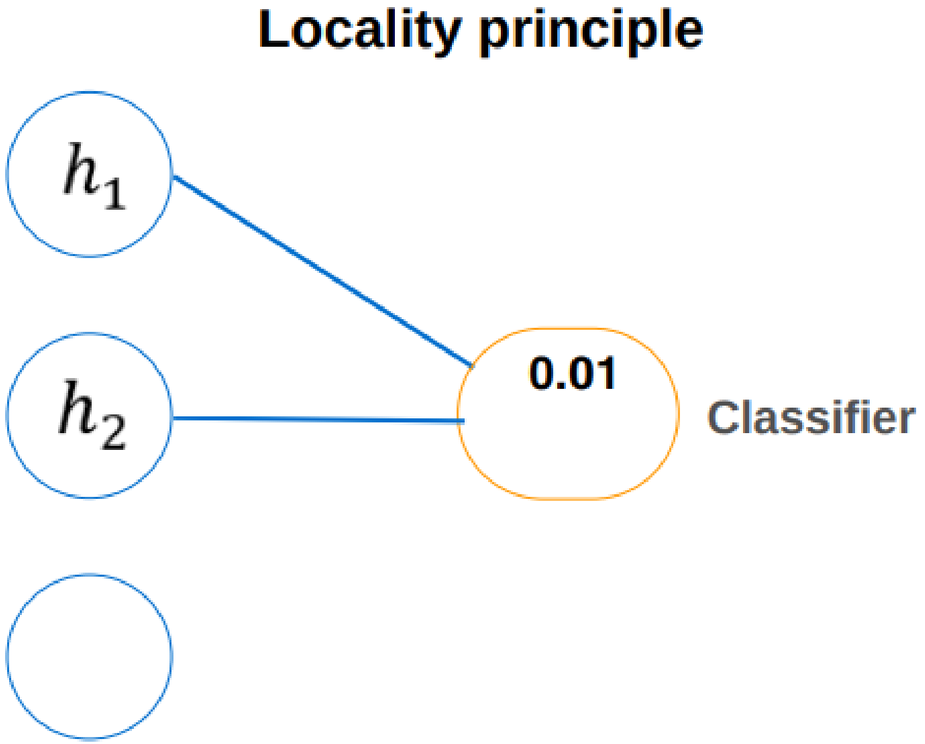
02. MLP vs CNN



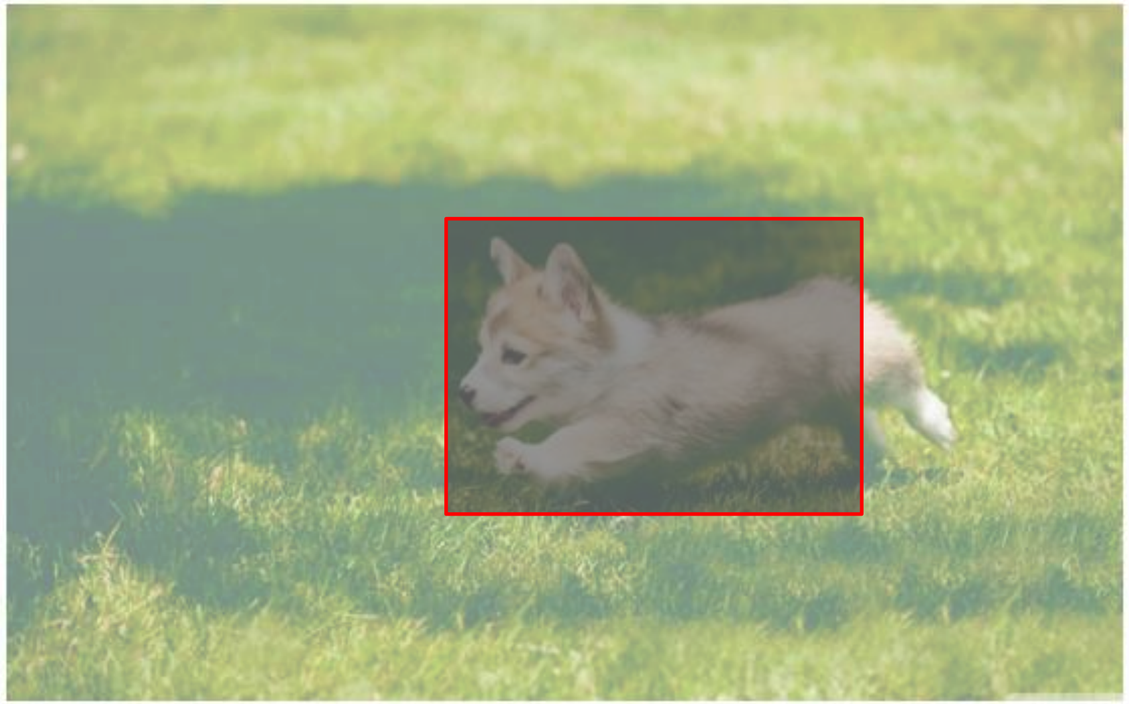


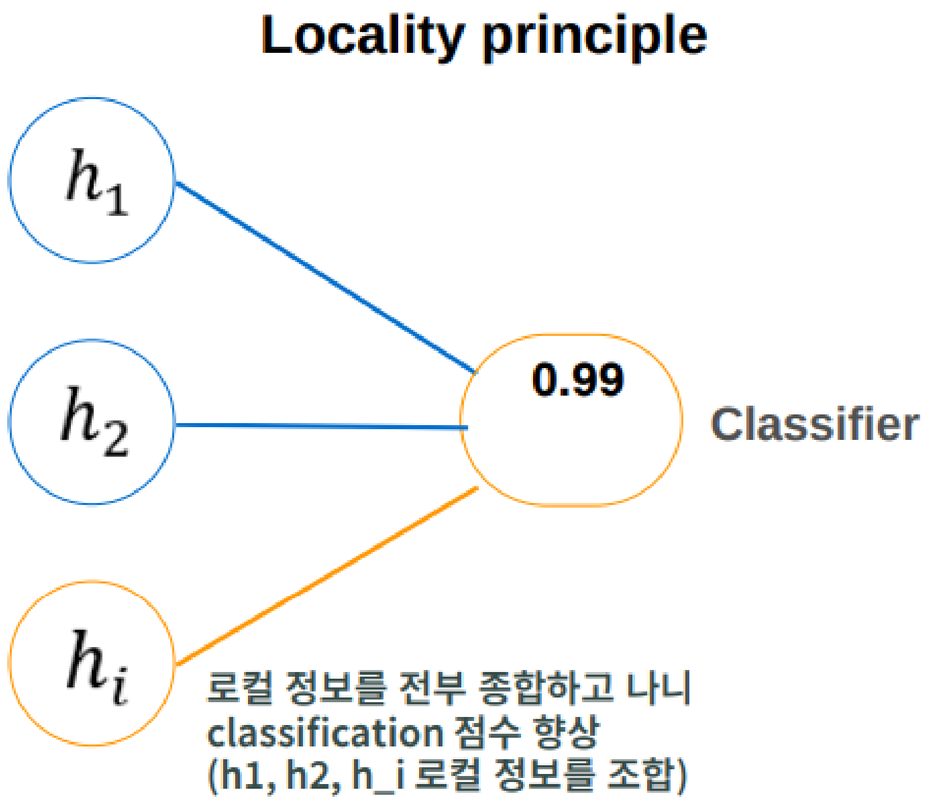
02. MLP vs CNN





02. MLP vs CNN





02. MLP vs CNN

Spatial Invariance

1. 객체의 모습은 그 위치에 의존하지 않는다.
2. 같은 객체가 이미지의 어느 위치에 있든지 간에 시스템이 일관되게 그 객체를 인식할 수 있어야 한다.

Classification
translation invariance

$$g_\phi(\mathcal{T}[\mathbf{x}]) = g_\phi(\mathbf{x})$$

$\mathcal{T}$ translation operation

Object detection 등
translation equivariance

$$f_\theta(\mathcal{T}[\mathbf{x}]) = \mathcal{T}[f_\theta(\mathbf{x})]$$

f_θ, g_ϕ models

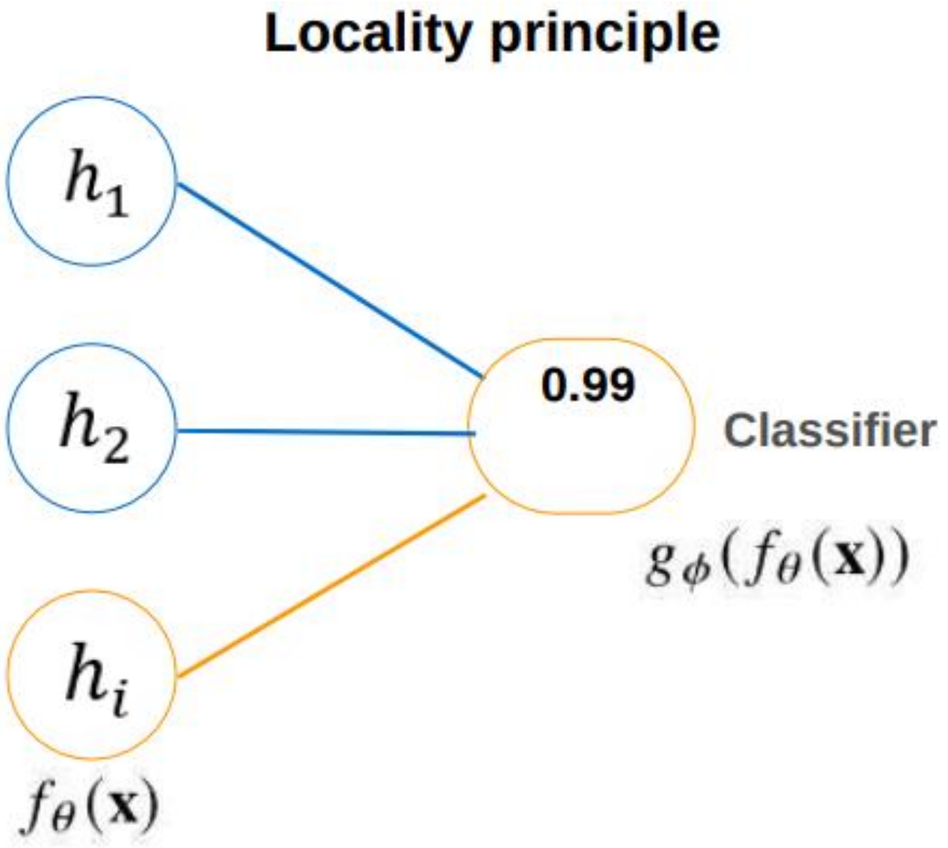
평행이동 후에도 출력 결과가 동일 (Pooling)

➡ **사진 속 강아지가 이미지의 중앙, 왼쪽, 오른쪽 어디에 있든 강아지로 분류**

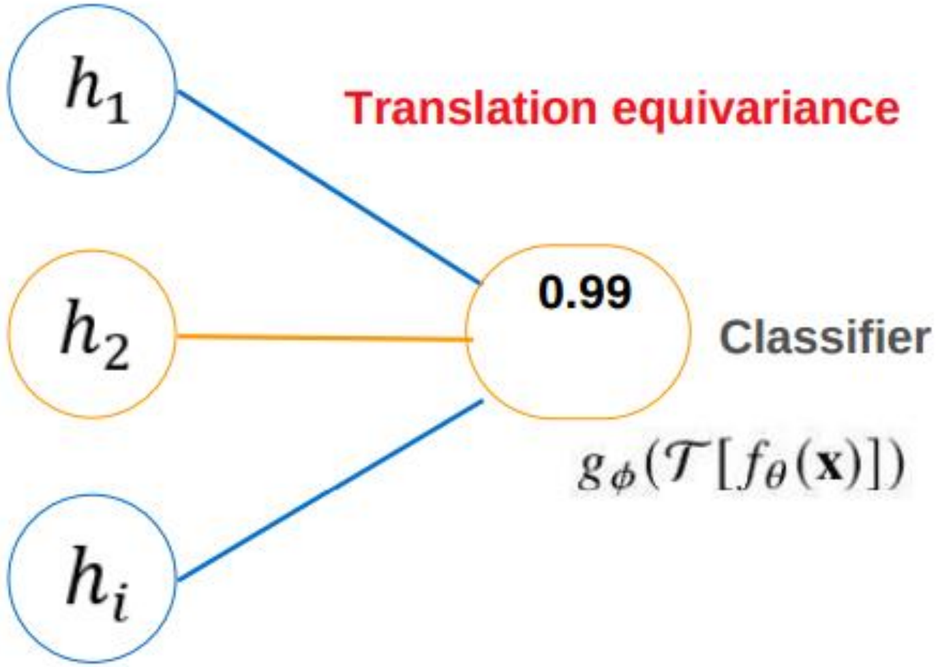
평행이동이 입력과 출력에 동일하게 반영 (Convolution)

➡ **입력을 평행 이동하면, 출력도 동일한 방식으로 평행이동 (특정 map에서 동일한 이동 발생)**

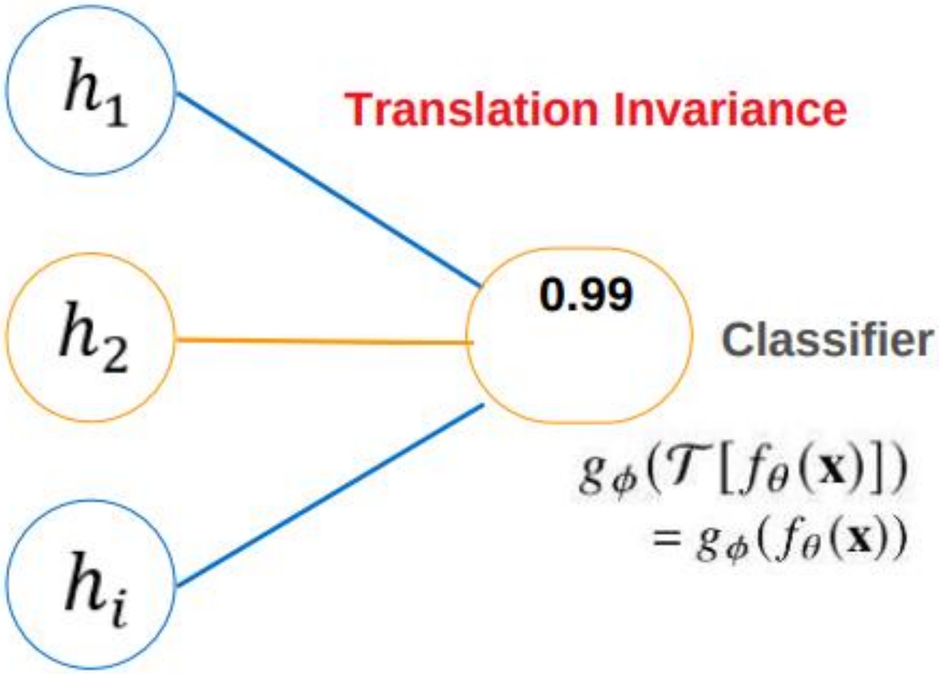
02. MLP vs CNN



02. MLP vs CNN



02. MLP vs CNN



02. MLP vs CNN

MLP가 이미지 처리에 가지는 한계

1. 계산 효율성 (학습 parameter의 수)

= input data size의 수가 너무 많아서 학습 parameter가 너무 많아짐.

2. Locality 문제

pixel 하나는 정보값 X, pixel은 독립적으로 정보를 주는 게 아니라 주변의 pixel과 합쳐져서 정보를 주게 됨 = Spatial relation

→ Flatten해서 나열해버리면 위치정보가 사라지게 됨. (Serialization의 2차원 벡터로 표현)

→+여기에다가 weight sum을 하면, 위치정보가 또 사라지게 됨.

3. Translation 문제

Translation = 일부 변형을 주더라도 같은 그림으로 인식하는 것 (예시: 한쪽에 치우쳐진 사진 -> 같은 그림으로 인식 X)

if) 10000장의 데이터에 대해 학습을 시킨다면

→ 각각의 case에 대해 일일이 학습을 시켜야 함. (한쪽에 치우쳐진 거, 사이즈 바꾼 거, noise 있는 거, 회전된 거, 등등)

= 각 augmentation에 대해 각 데이터를 모두 학습을 시켜야 함.

→ 학습이 너무 어렵기 때문에, Translation을 할 수 있는 모델을 써야 함.

02. MLP vs CNN

1. 계산 효율성 (학습 parameter의 수)

= input data size의 수가 너무 많아서 학습 parameter가 너무 많아짐.

Convolution에서는 filter를 사용

* filter : surrounding pixel을 같이 학습

Parameter 수가 filter에 dependent함

2. Locality 문제

pixel 하나는 정보값 X, pixel은 독립적으로 정보를 주는 게 아니라 주변의 pixel과 합쳐져서 정보를 주게 됨 = Spatial relation

→ Flatten해서 나열해버리면 위치정보가 사라지게 됨. (Serialization의 2차원 벡터로 표현)

→+여기에다가 weight sum을 하면, 위치정보가 또 사라지게 됨.

3. Translation 문제

Translation = 일부 변형을 주더라도 같은 그림으로 인식하는 것 (예시: 한쪽에 치우쳐진 사진 -> 같은 그림으로 인식 X)

if) 10000장의 데이터에 대해 학습을 시킨다면

→ 각각의 case에 대해 일일이 학습을 시켜야 함. (한쪽에 치우쳐진 거, 사이즈 바꾼 거, noise 있는 거, 회전된 거, 등등)

= 각 augmentation에 대해 각 데이터를 모두 학습을 시켜야 함.

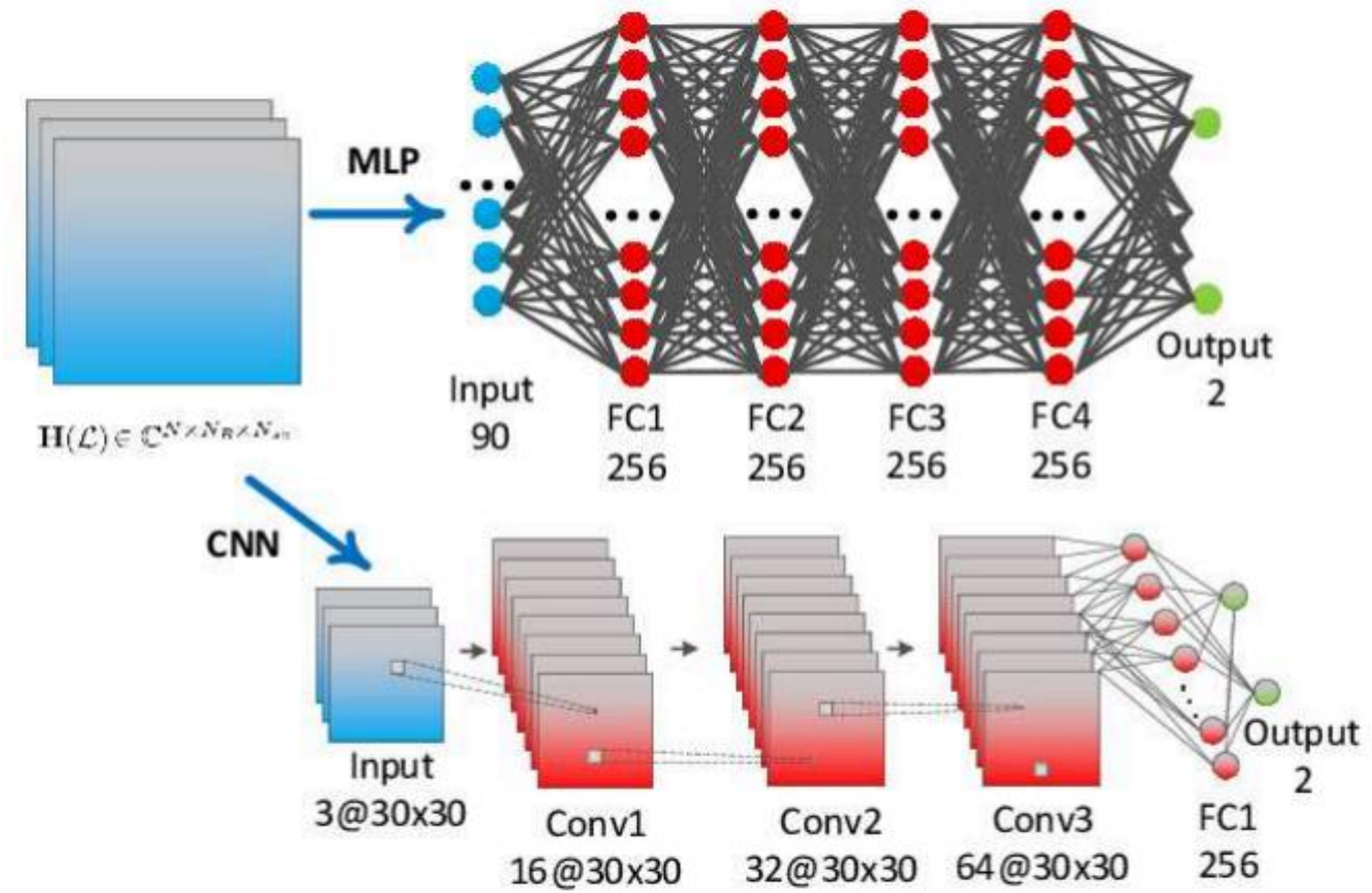
→ 학습이 너무 어렵기 때문에, Translation을 할 수 있는 모델을 써야 함.

Max pooling으로 해결

(translation invariance 특성을 사용)

03
Convolution Layer

03. Convolution Layer

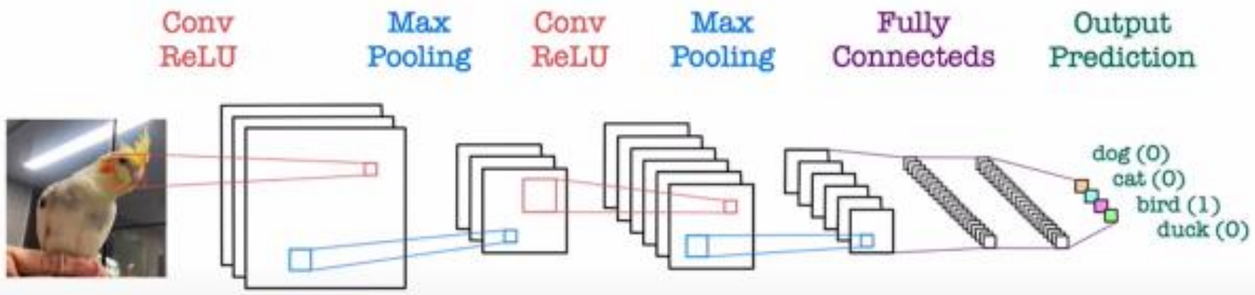


03. Convolution Layer

CNN의 Architecture

CNN architecture

- LeCun creates the first CNN LeNet for character recognitions at 1998.
- AlexNet which is ILSVRC 2012 winner has similar structure as LeNet.



* Convolution (shared weights, dimension reduction)
* Nonlinearity (ReLU)
* Max pooling (translation invariance, dimension reduction for FC layer)
* Fully connected layer (classification)

03. Convolution Layer

CNN이 이미지를 인식하는 단계

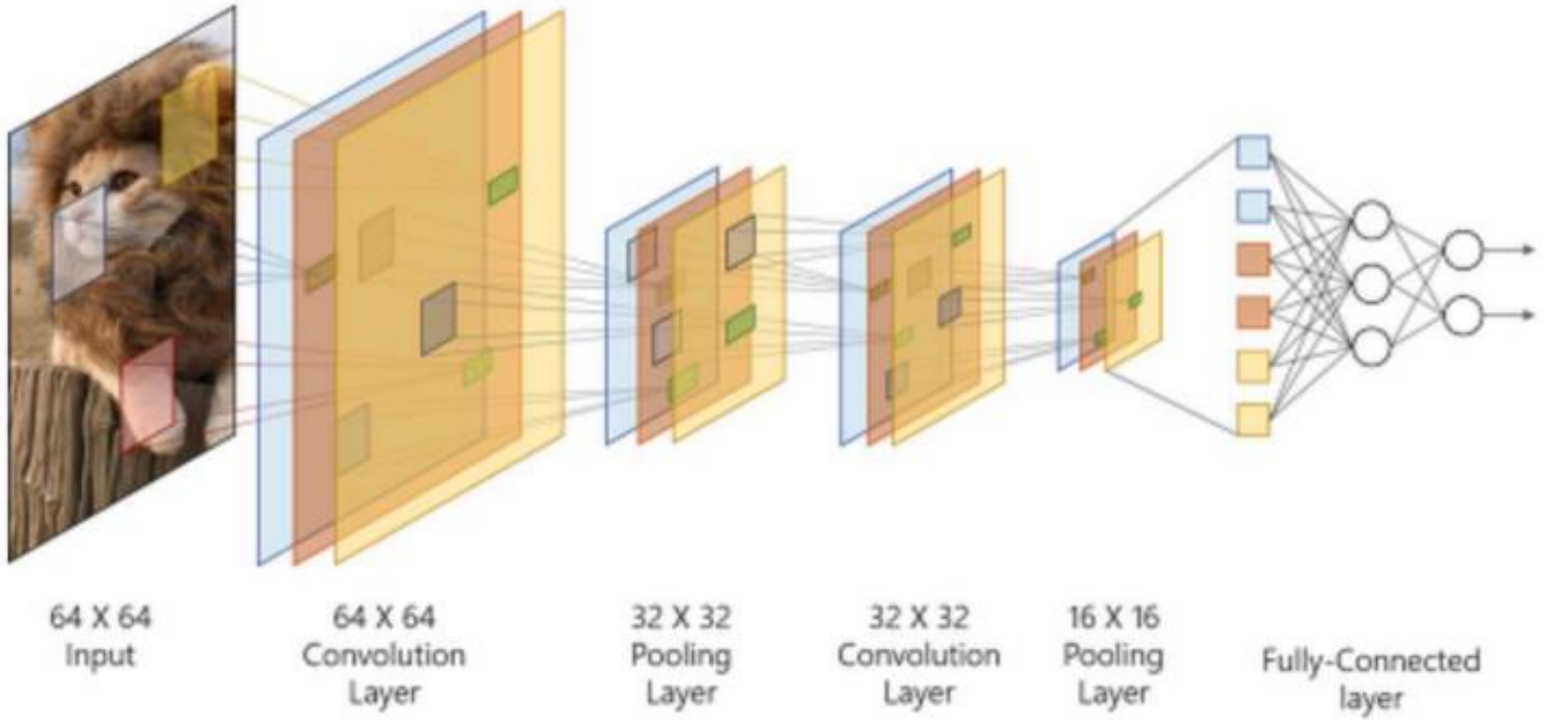
1단계 : 가로, 동그라미, 세모, 부드러움

2단계 :눈, 코, 귀, 발

3단계 :고양이!

03. Convolution Layer

CNN이 이미지를 인식하는 단계



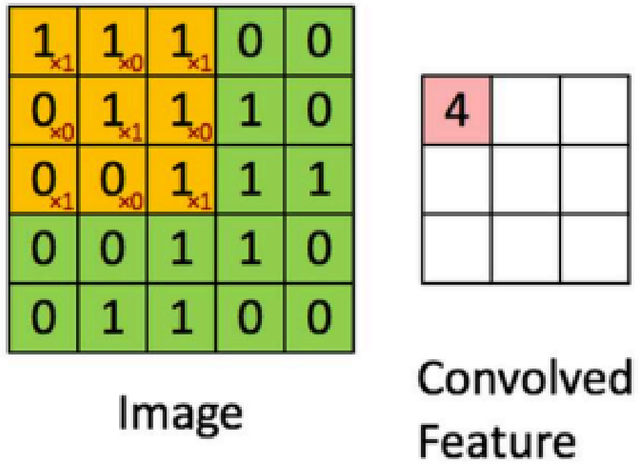
03. Convolution Layer

What is convolution?

what is convolution? cross correlation?

합성곱: $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$

교차상관: $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t+\tau)\,d\tau$



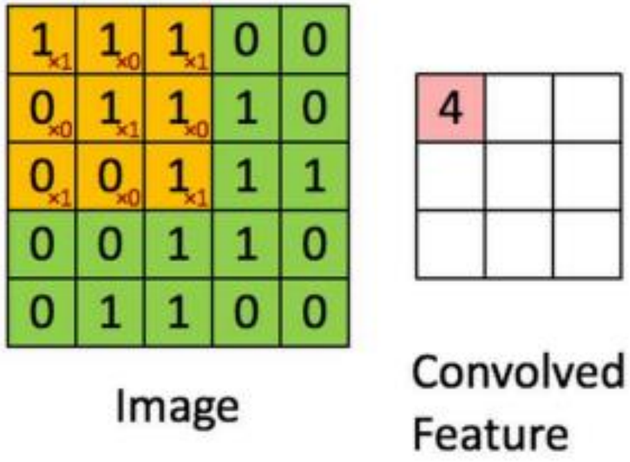
03. Convolution Layer

What is convolution?

1D Conv $$[\mathbf{f} * \mathbf{x}](i) = \sum_{p} \mathbf{f}(p)\mathbf{x}(i+p)$$

2D Conv $$[\mathbf{f} * \mathbf{x}](i,j) = \sum_{p,q} \mathbf{f}(p,q)\mathbf{x}(i+p,j+q)$$

3D Conv $$[\mathbf{f} * \mathbf{x}](i,j,k) = \sum_{p,q,r} \mathbf{f}(p,q,r)\mathbf{x}(i+p,j+q,k+r)$$



03. Convolution Layer

$$h_1 = \sum_{|p|\le\Delta} \mathbf{f}(p)\mathbf{x}(1+p) \in \text{1st patch}$$

$$h_2 = \sum_{|p|\le\Delta} \mathbf{f}(p)\mathbf{x}(2+p) \in \text{2nd patch}$$

$$\vdots \qquad\qquad \vdots$$

$$h_i = \sum_{|p|\le\Delta} \mathbf{f}(p)\mathbf{x}(i+p) \in i\text{-th patch}$$

03. Convolution Layer

$$h_1 = \sum_{|p|\le\Delta} \mathbf{f}(p)\mathbf{x}(1+p) \in \text{1st patch}$$

$$h_2 = \sum_{|p|\le\Delta} \mathbf{f}(p)\mathbf{x}(2+p) \in \text{2nd patch}$$

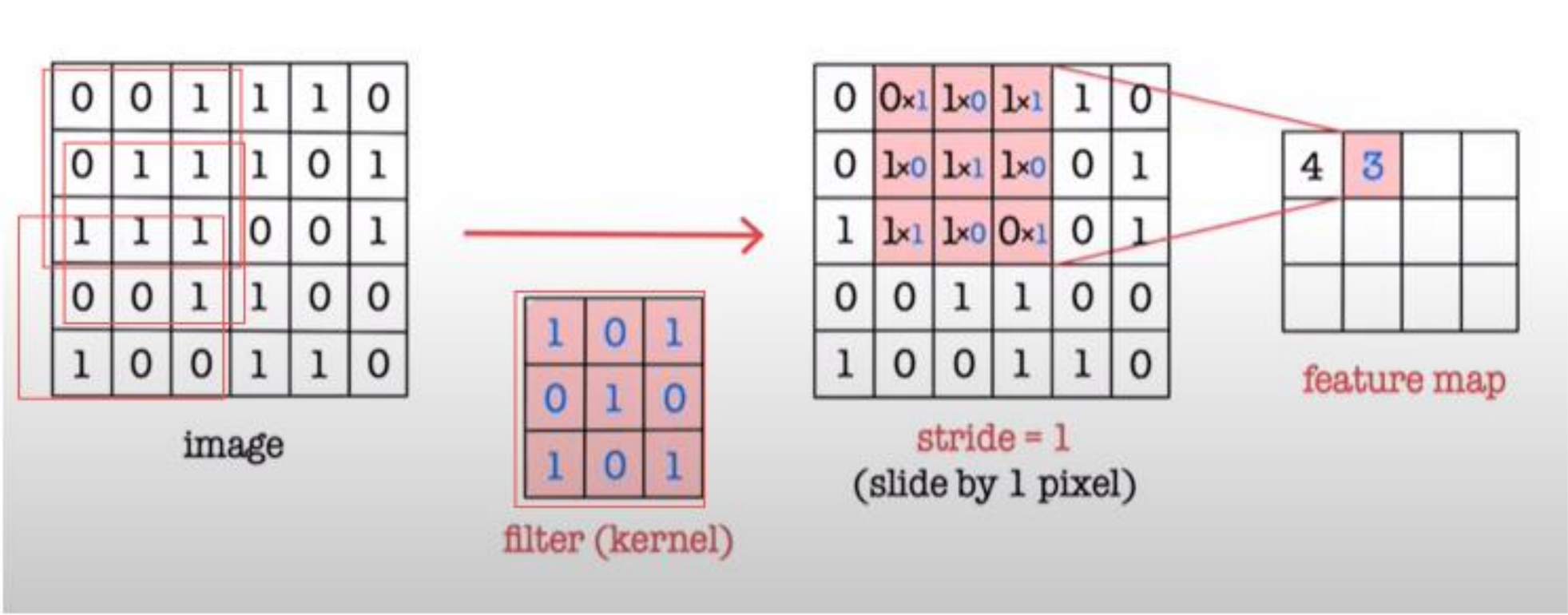
$$\vdots \qquad \vdots$$

$$h_i = \sum_{|p|\le\Delta} \mathbf{f}(p)\mathbf{x}(i+p) \in i\text{-th patch}$$

filters

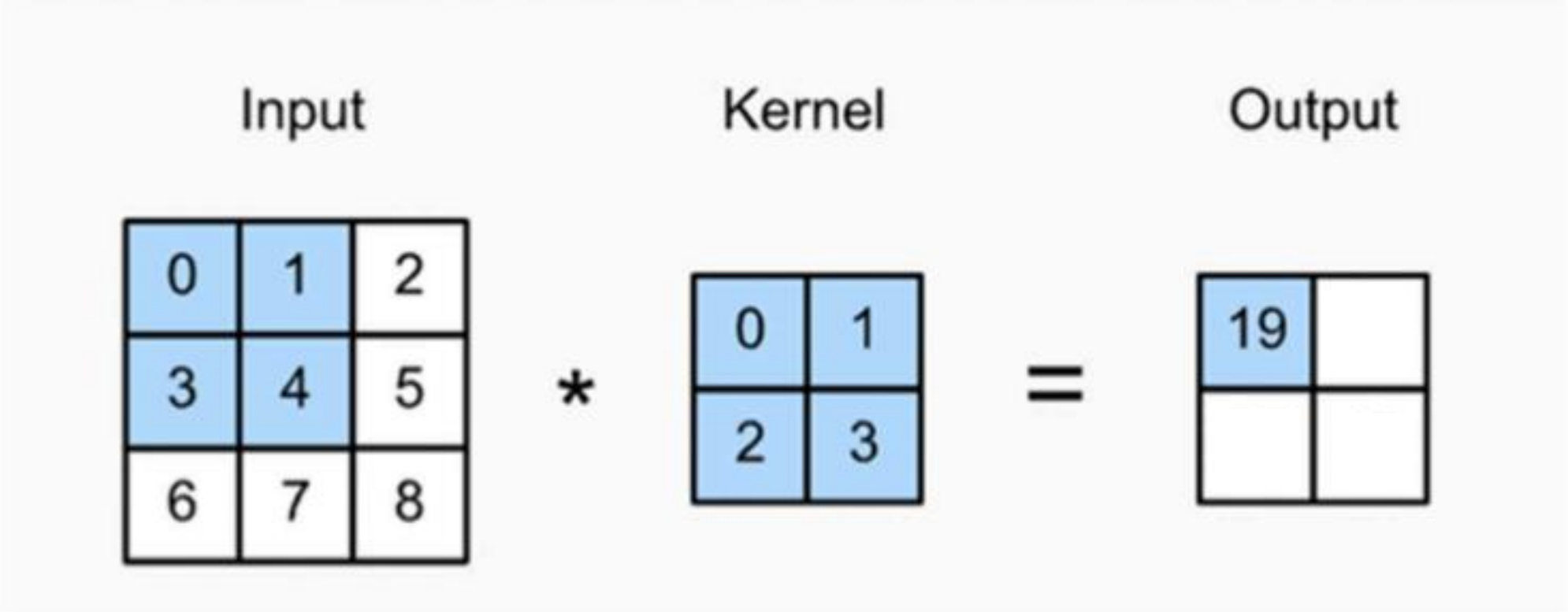
03. Convolution Layer

Convolution 연산



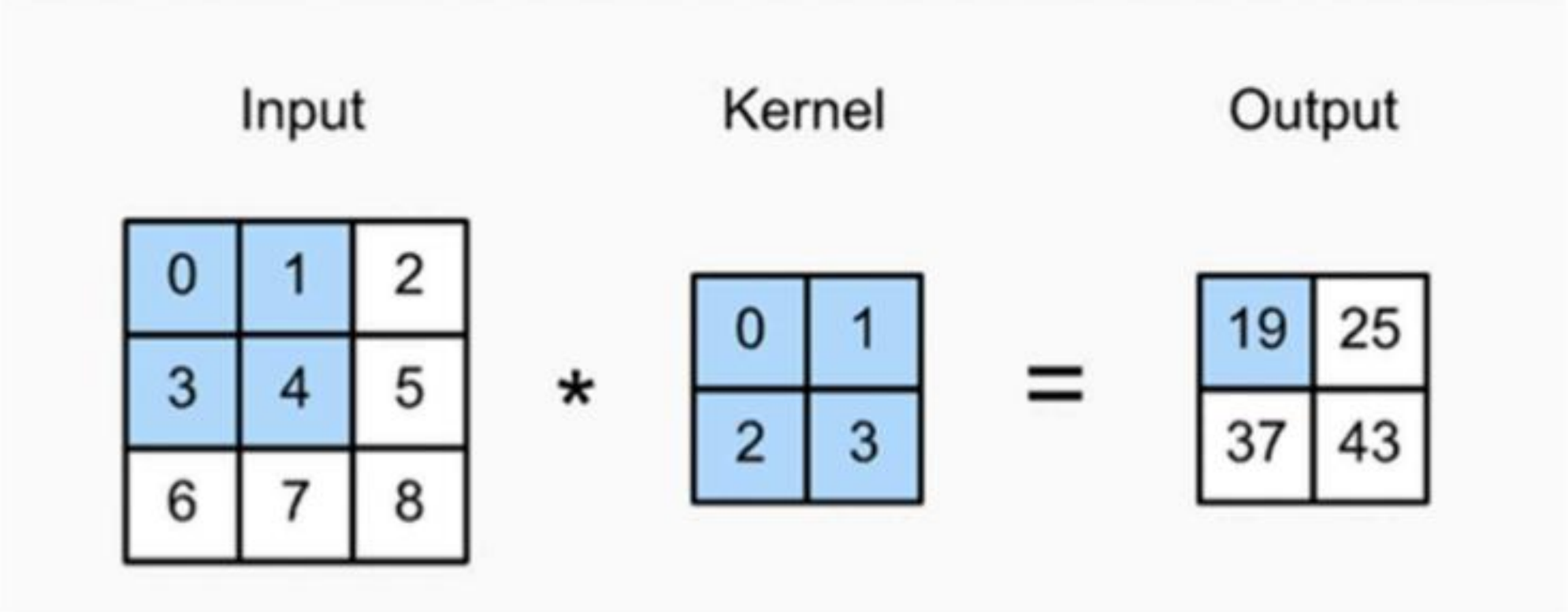
03. Convolution Layer

Convolution 연산



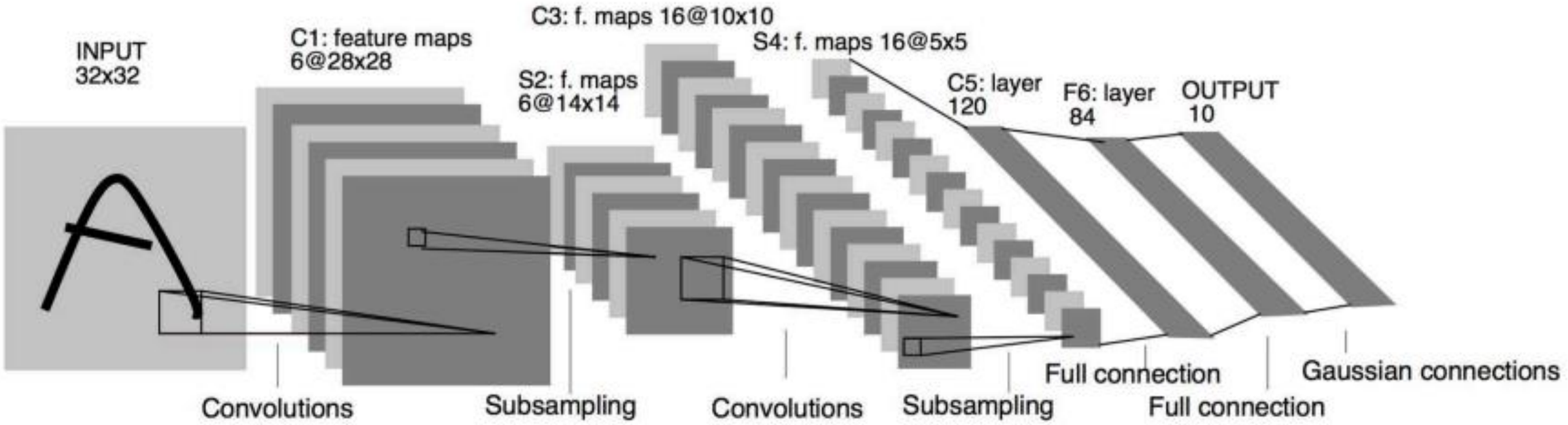
03. Convolution Layer

Convolution 연산

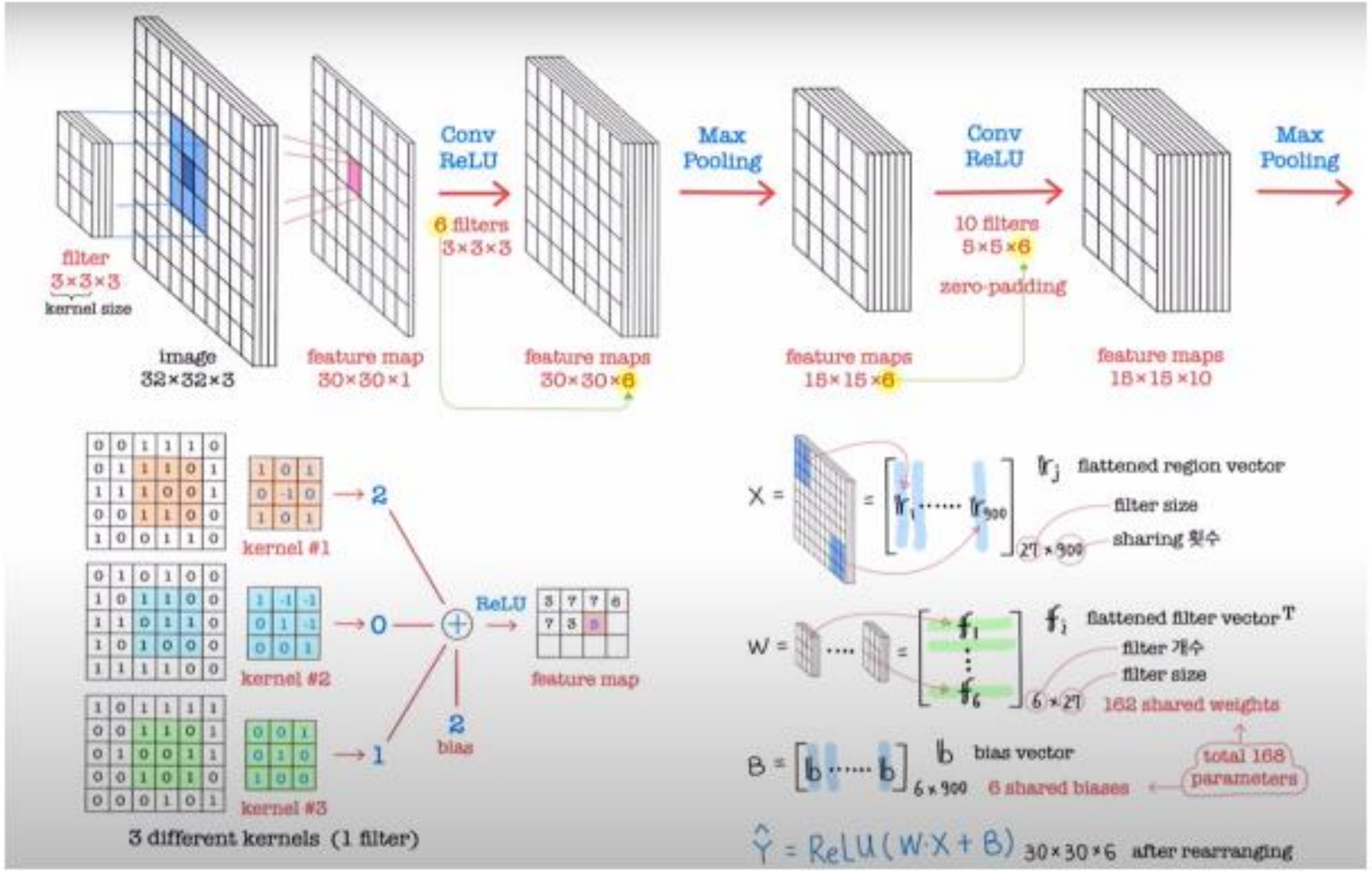


03. Convolution Layer

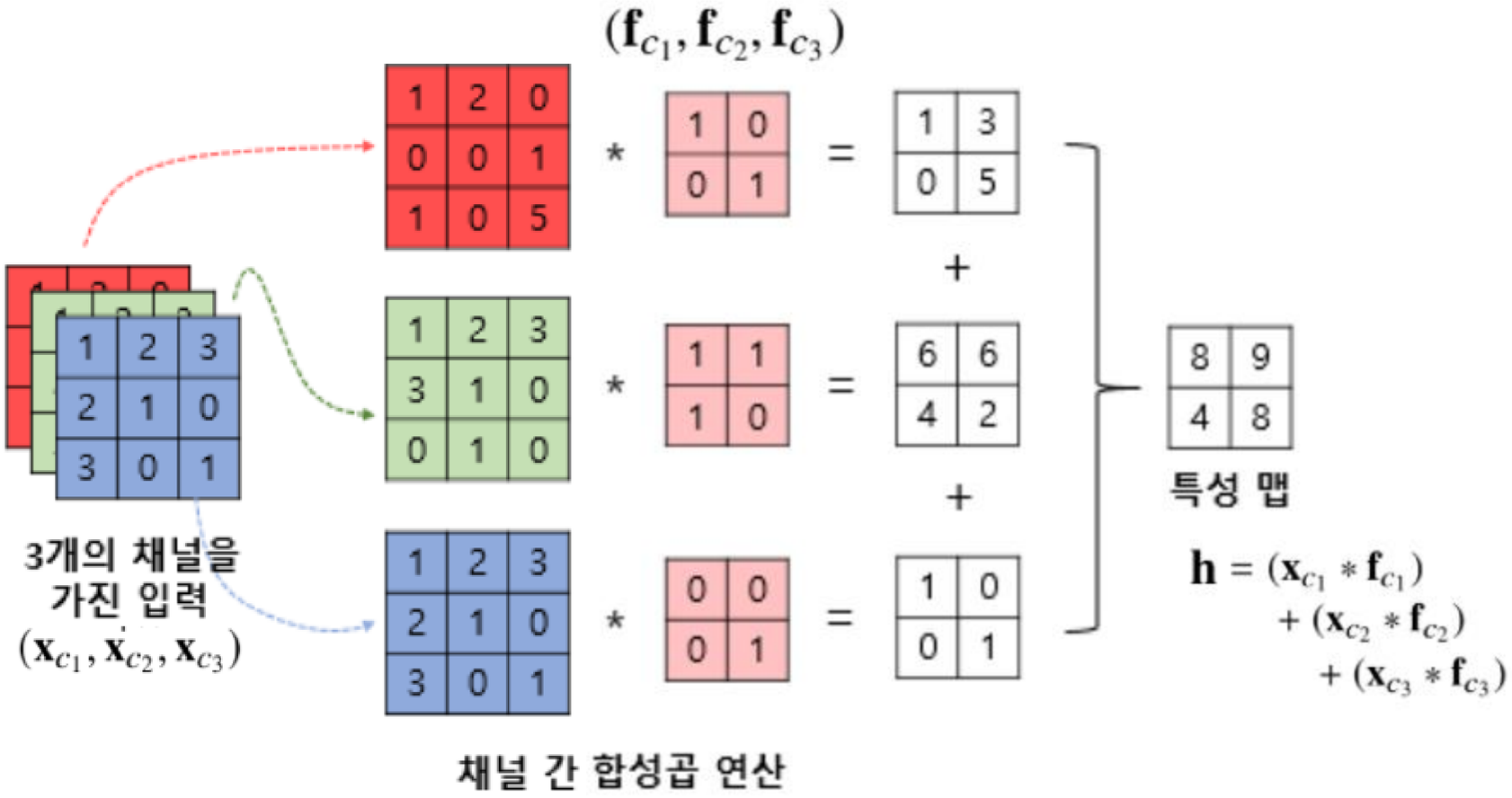
Convolution 연산



03. Convolution Layer



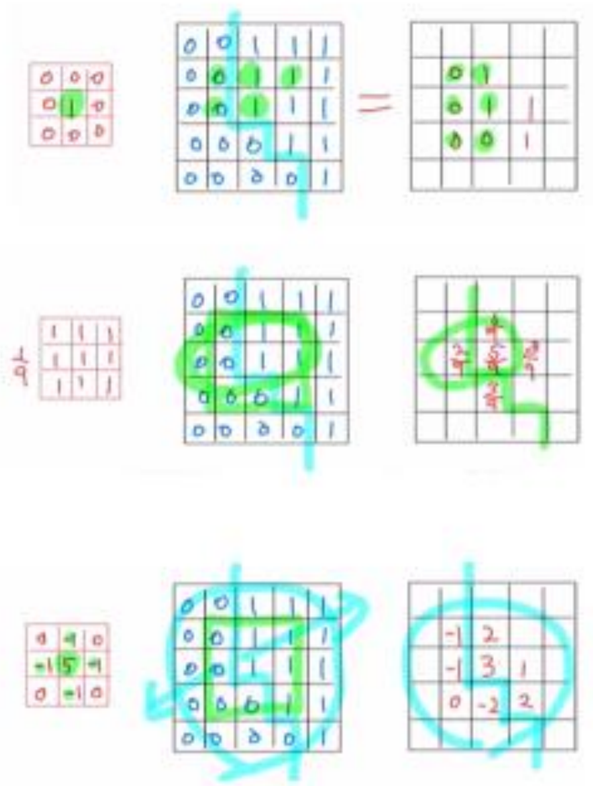
03. Convolution Layer



04

Padding & Stride & Pooling Layer

04. Padding & Stride & Pooling Layer



같은 input image에 다른 filter 적용하면
다른 종류의 feature map이 나옴.

Depending on the element values, a kernel can cause a wide range of effects:

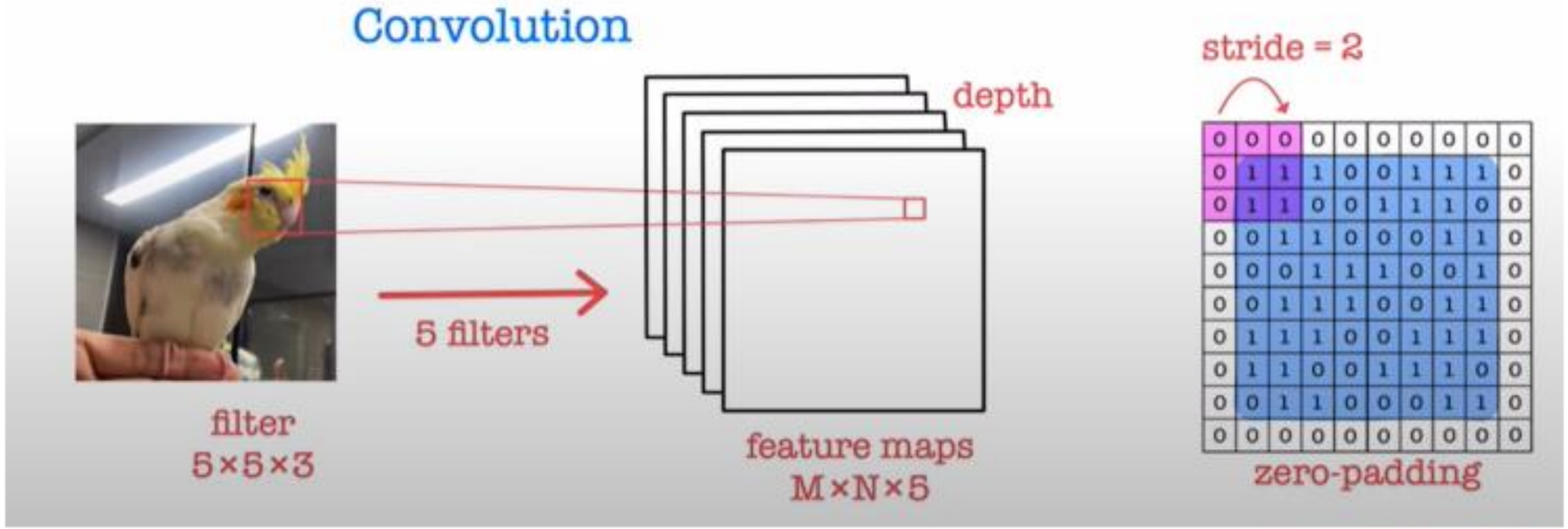
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

04. Padding & Stride & Pooling Layer

Convolutional layer의 4가지 hyperparameter

1. filters (filter 개수)
2. kenerl_size (convolution window: h*w)
3. strides (kernel 간 간격)
4. padding (valid=no padding, same= zero-padding, input과 같은 size로 ouput을 만들어줌)

Convolution2D(64, (3,3), strides=1, padding="same", ...)



3*3 kernel 사용하면 stride 1여도 되는데
5*5 kernel 사용하면 stride 2여야 함.

04. Padding & Stride & Pooling Layer

Padding?

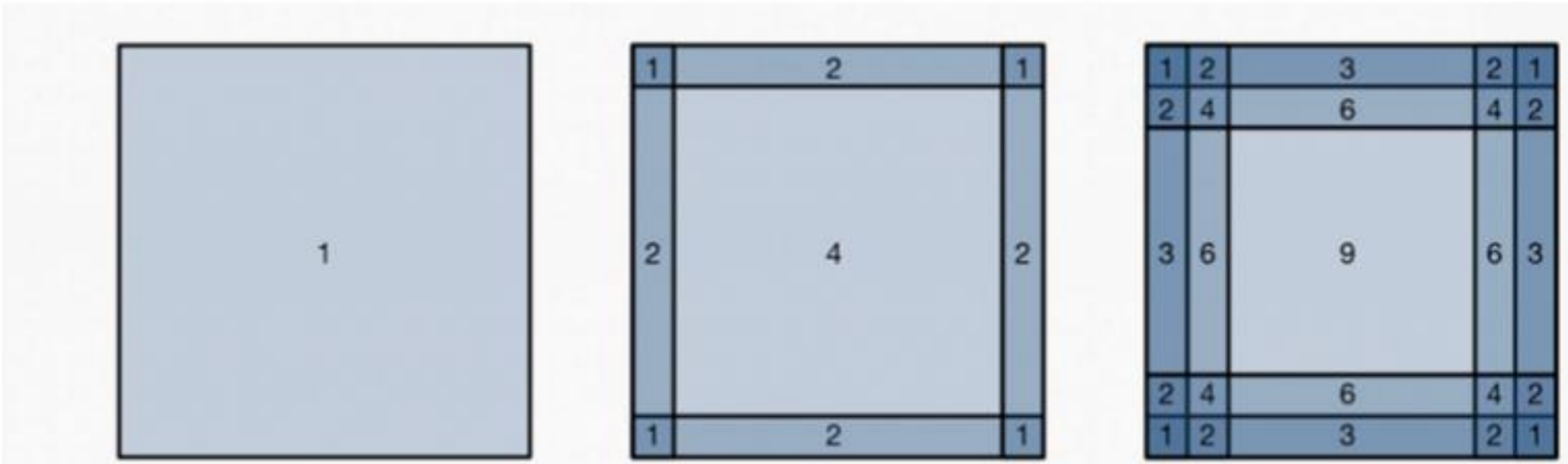
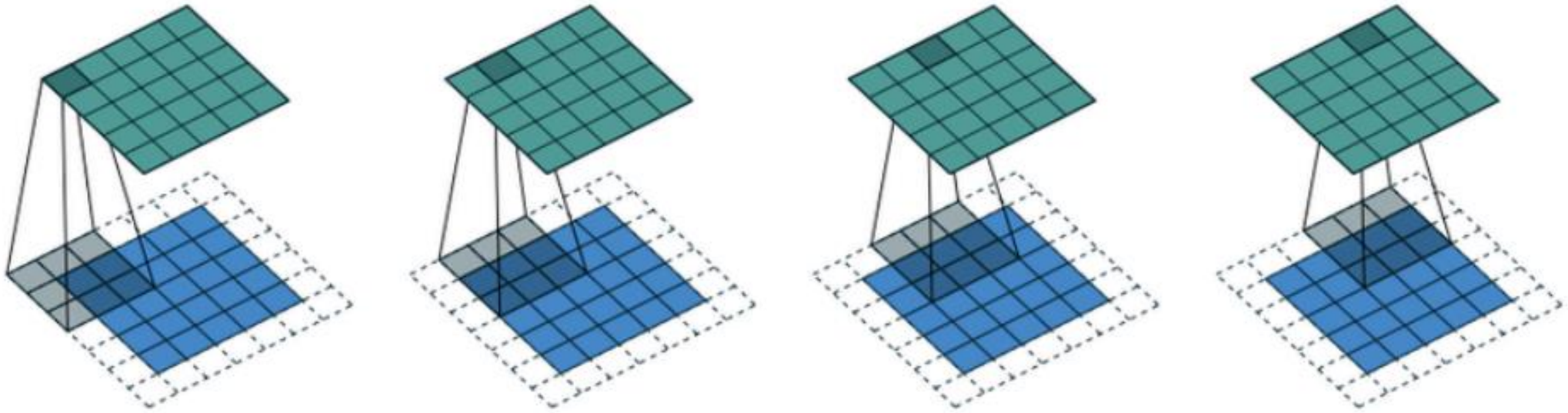
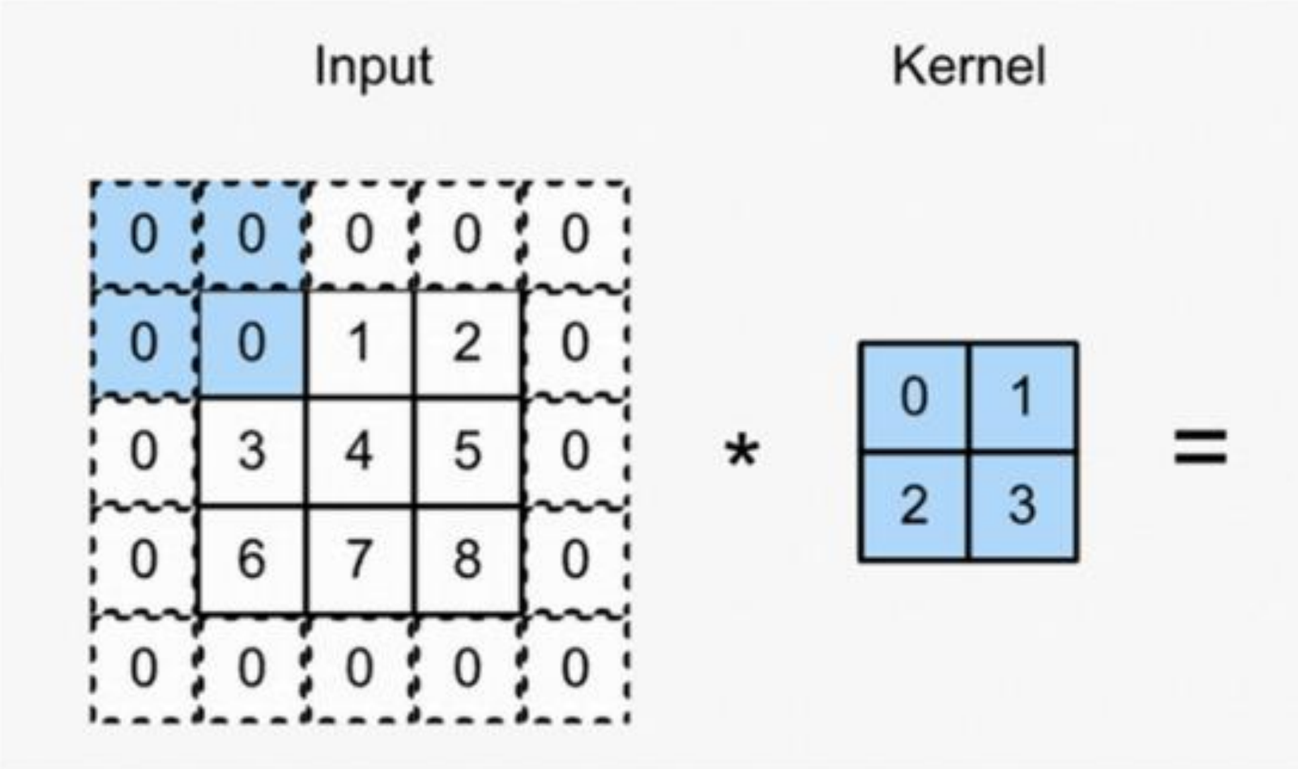


Fig. 7.3.1 Pixel utilization for convolutions of size 1×1, 2×2, and 3×3 respectively.

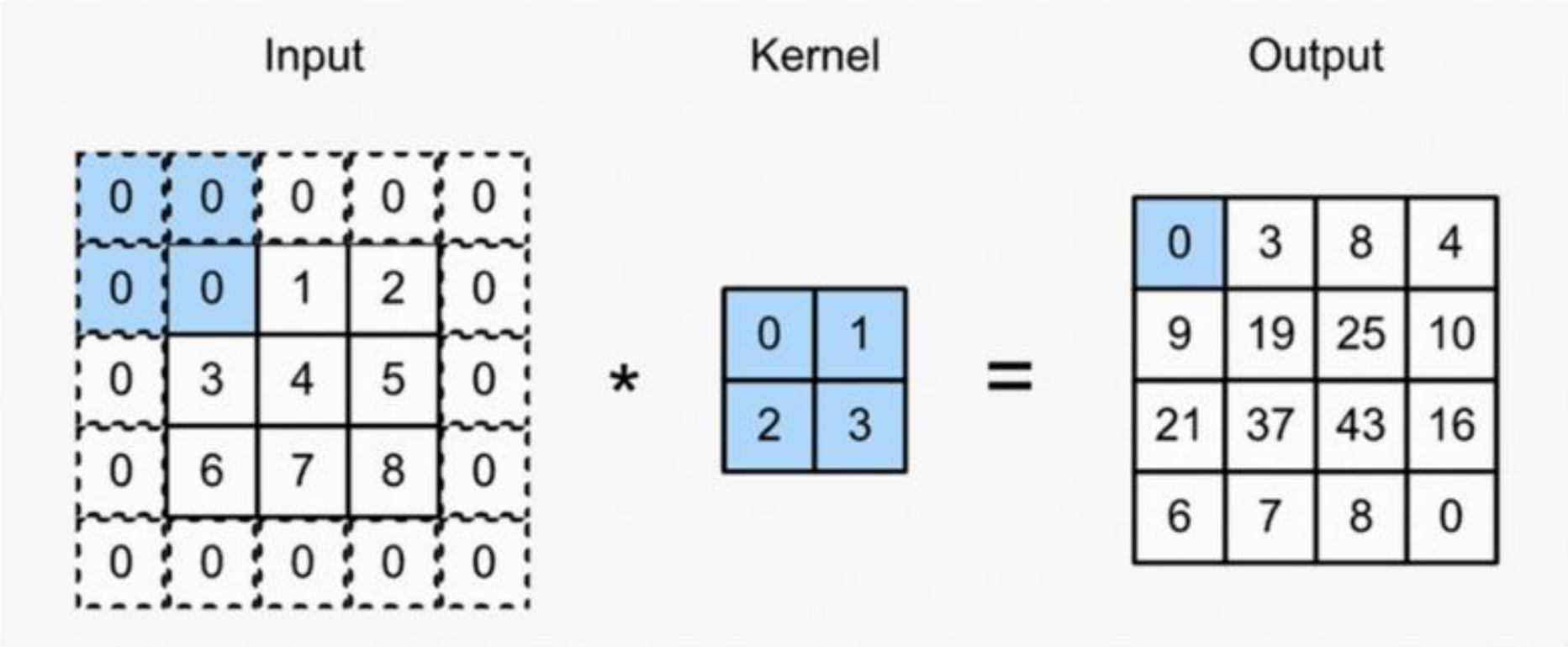
04. Padding & Stride & Pooling Layer



04. Padding & Stride & Pooling Layer

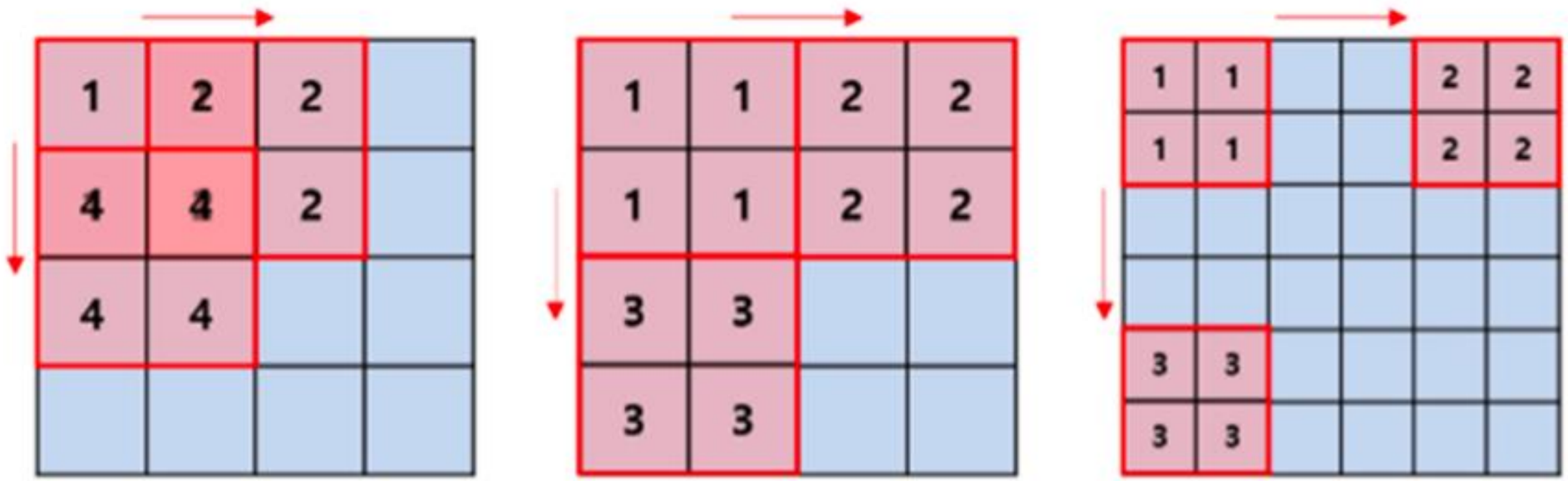


04. Padding & Stride & Pooling Layer



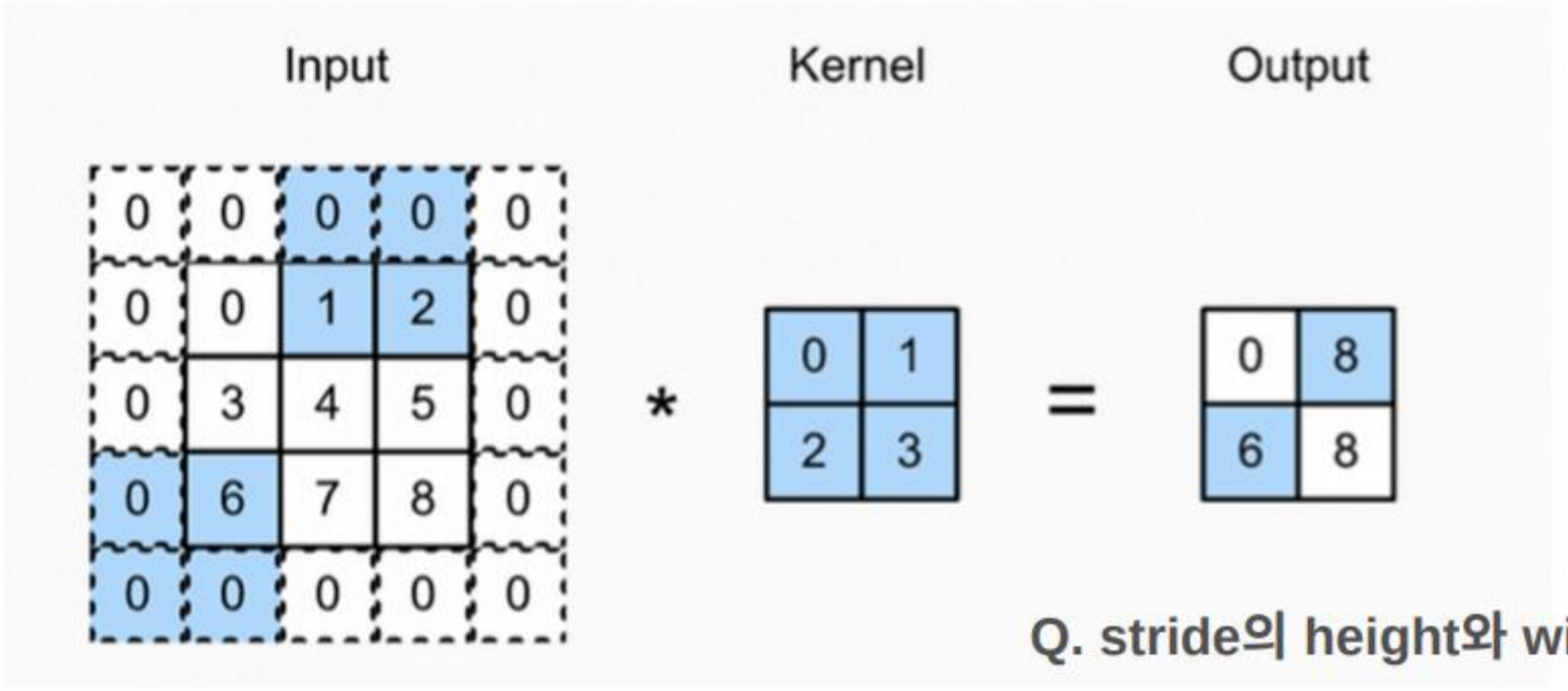
04. Padding & Stride & Pooling Layer

Stride?



[Stride: 1] [Stride: 2] [Stride: 3]

04. Padding & Stride & Pooling Layer



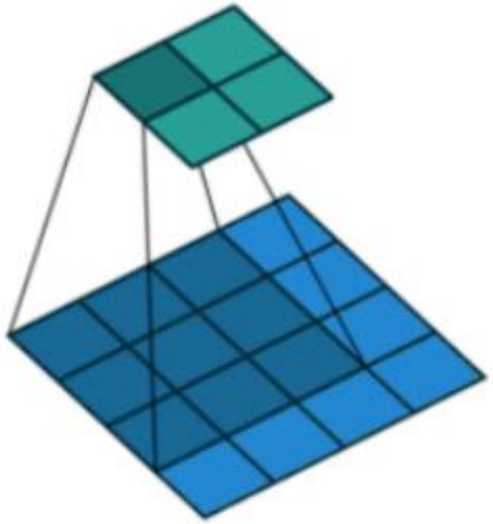
04. Padding & Stride & Pooling Layer

$$O_h = \left\lfloor \frac{I_h - K_h + 2P}{S_h} + 1 \right\rfloor$$

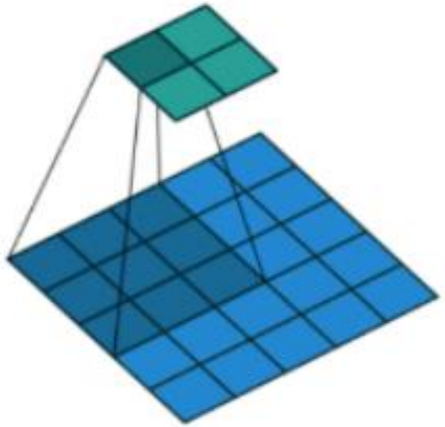
$$O_w = \left\lfloor \frac{I_w - K_w + 2P}{S_w} + 1 \right\rfloor$$

O_h: 출력 이미지의 높이(Height)
O_w: 출력 이미지의 너비(Width)
I_h: 입력 이미지의 높이
I_w: 입력 이미지의 너비
K_h: 컨볼루션 커널의 높이
K_w: 컨볼루션 커널의 너비
P: 패딩(padding)의 양
S_h: 스트라이드(stride)의 높이
S_w: 스트라이드(stride)의 너비

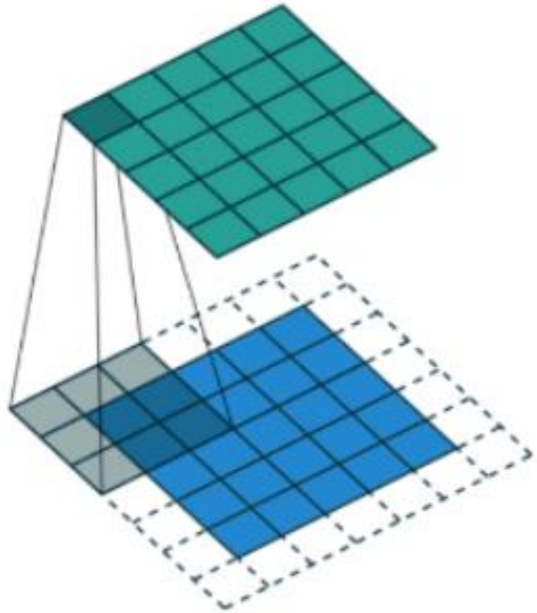
04. Padding & Stride & Pooling Layer



Vanilla convolution



convolution + stride



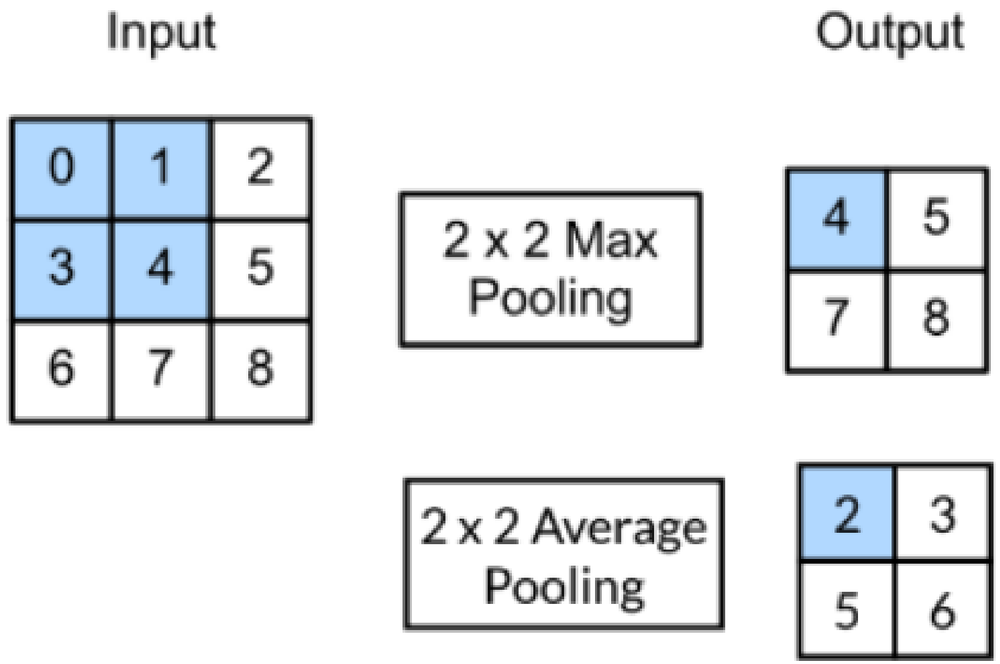
convolution + padding

04. Padding & Stride & Pooling Layer

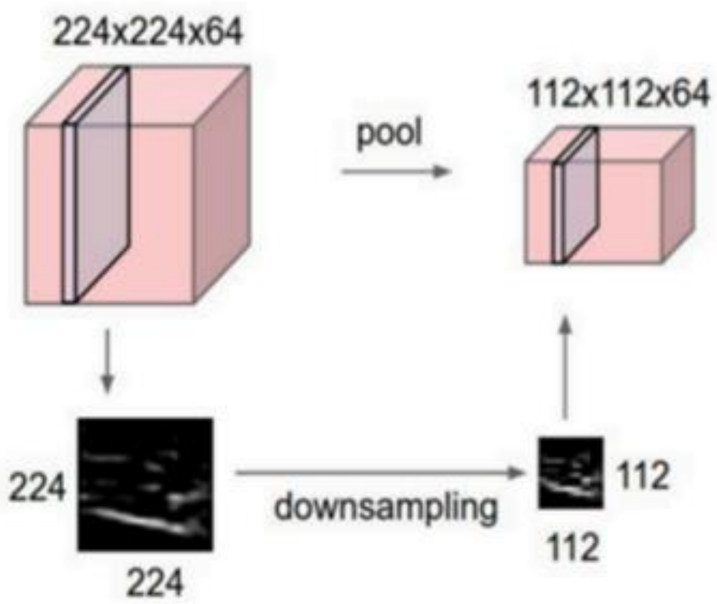
Pooling ?

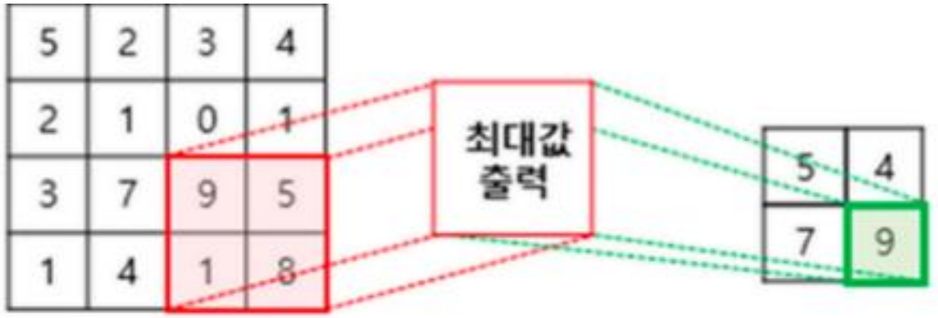
점진적으로 spatial resolution 을 줄여준다.

- aggregate information
- downsampling
- parameter-free operator
- locally translation invariance



04. Padding & Stride & Pooling Layer





Pooling ?

점진적으로 spatial resolution 을 줄여준다.

- aggregate information
- downsampling
- parameter-free operator
- locally translation invariance

Pooling을 통해서 차원을 축소

Max Pooling, Mean Pooling

Conv 계산처럼 Stride가 존재

04. Padding & Stride & Pooling Layer

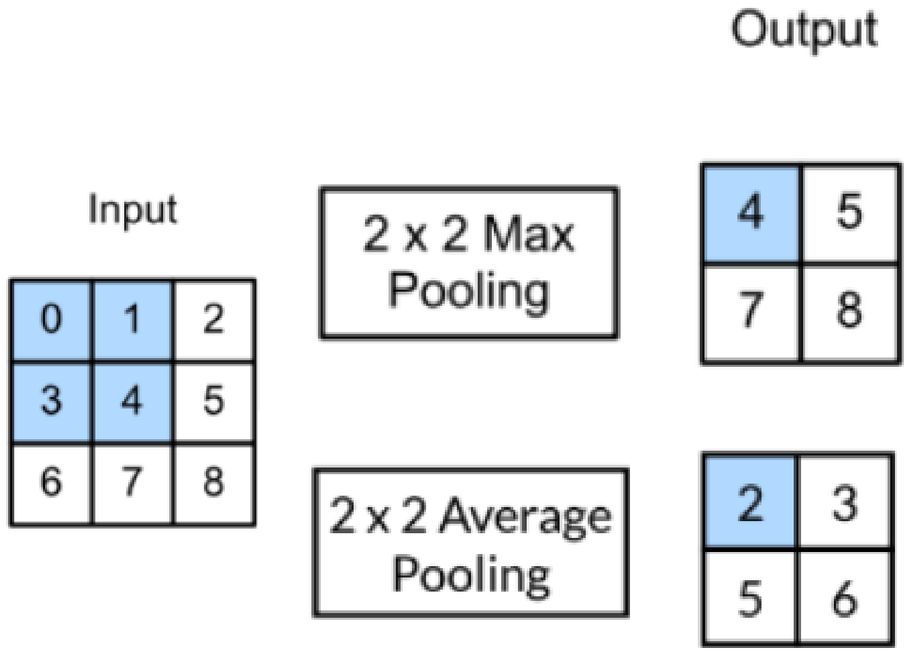
Pooling의 종류

1. Maximum pooling

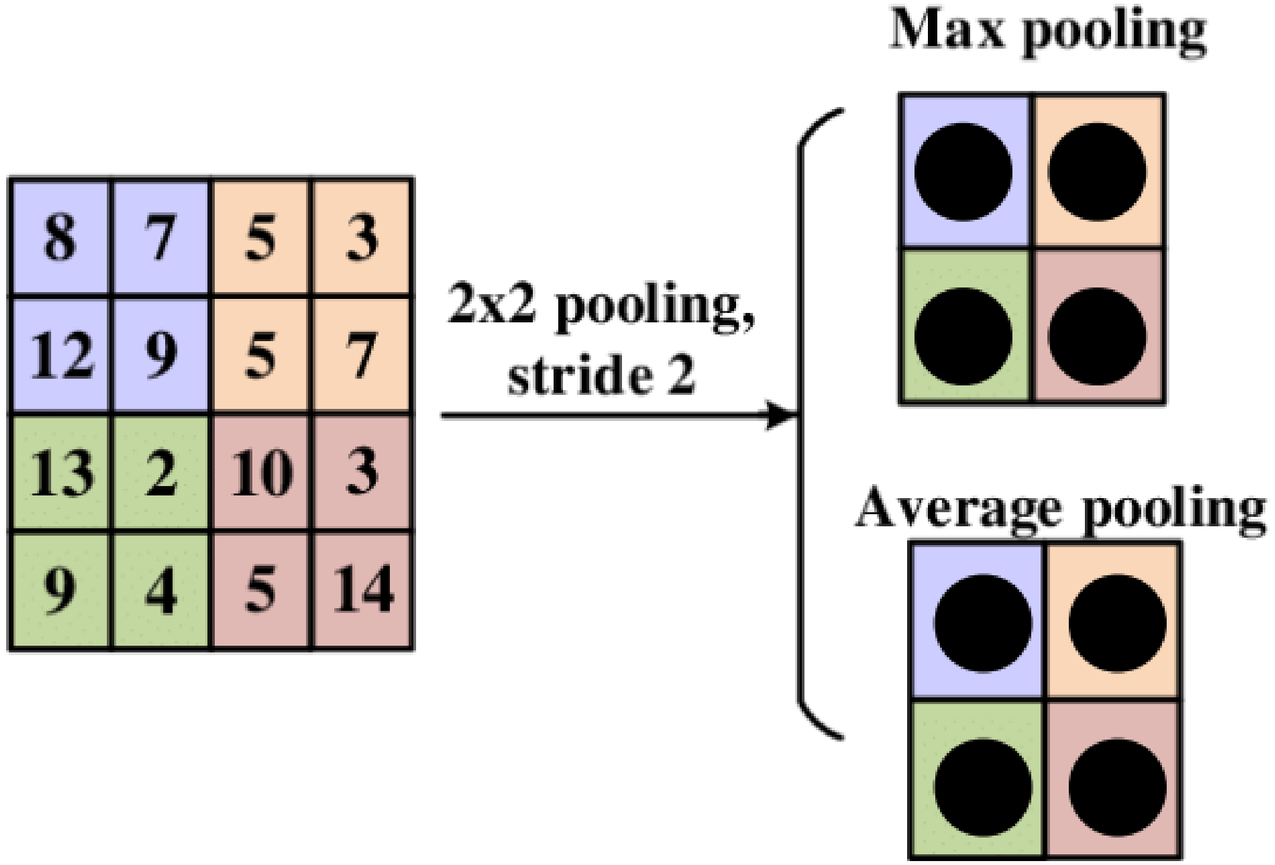
```
X = torch.tensor([[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]])
pool2d(X, (2, 2))
```

2. Average pooling

```
pool2d(X, (2, 2), 'avg')
```



04. Padding & Stride & Pooling Layer



ETC.

Parameter 개수 비교

시나리오

- 입력 이미지의 크기 : **32x32** 픽셀
- 입력 이미지의 채널 수 : **3 (RGB)**
- **MLP**의 **hidden layer: 1**개, 뉴런의 수는 **128**개
- **CNN**의 **convolution layer: 32ro** 필터, 각 필터의 크기 **3x3**

MLP의 파라미터 계산 :

CNN의 파라미터 계산 :

ETC.

ResNet

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

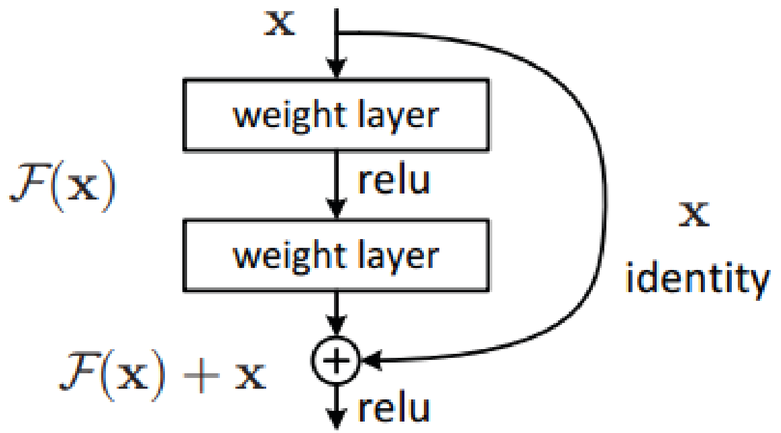


Figure 2. Residual learning: a building block.

Summary

MLP

- 전통적인 신경망 구조로, 입력층, 하나 이상의 은닉층, 그리고 출력층으로 구성된다.
- 각 층의 모든 뉴런은 이전 층의 모든 뉴런과 연결되어 있다.
- 이미지와 같은 고차원 데이터에서는 매개변수의 수가 매우 많아질 수 있어 비효율적일 수 있다.

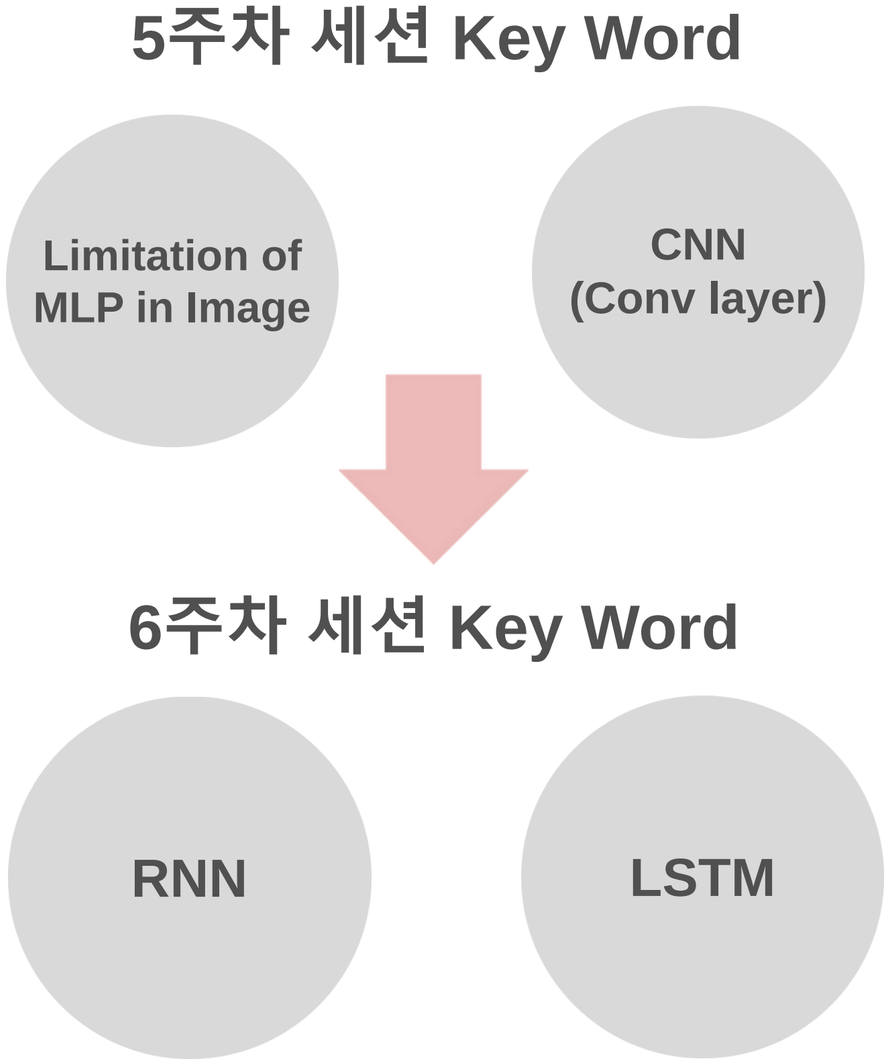
CNN

- 이미지 처리에 최적화된 신경망 구조로, 합성곱층과 풀링층을 통해 특징을 추출한다.
- 가중치 공유를 통해 훨씬 적은 수의 매개변수를 사용한다.
- 이미지의 공간적 계층 구조를 인식할 수 있어 이미지 인식과 분류에 매우 효과적이다.

05

다음 주차 예고

05. 다음 주차 예고



05. 다음 주차 예고

5주차 과제

공부 과제

d2l

9장, **10**장

RNN, LSTM

논문 리뷰

ResNet

코드 과제

슬랙 공지로
추후 전달 예정

Thank You

2025 DL Session 5주차 끝!

KOREA
UNIVERSITY