딥러닝을 활용한
디지털 영상처리
Digital Image Processing via Deep Learning

Lecture 8 – Introduction to Convolutional Neural Network

목차

- 인공신경망 복습
- 신경망 예제
- CNN 모티브
- CNN 구조
- Convolution
- Pooling

복습 1

Input vector, $\boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$ ← Bias - 편향

Then,

$$\boldsymbol{h} = a(\boldsymbol{w^{(1)}x})$$

Where

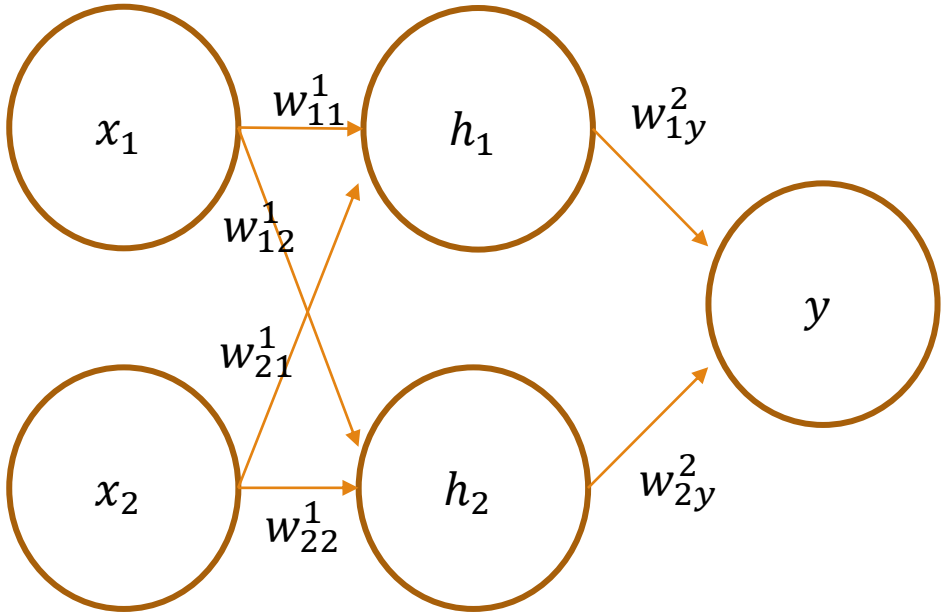
$$\boldsymbol{w^{(1)}} = \begin{pmatrix} w_{b1}^1 & w_{11}^1 & w_{21}^1 \\ w_{b2}^1 & w_{12}^1 & w_{22}^1 \end{pmatrix}$$

Then,

$$y = a(\boldsymbol{w^{(2)}h})$$

Where

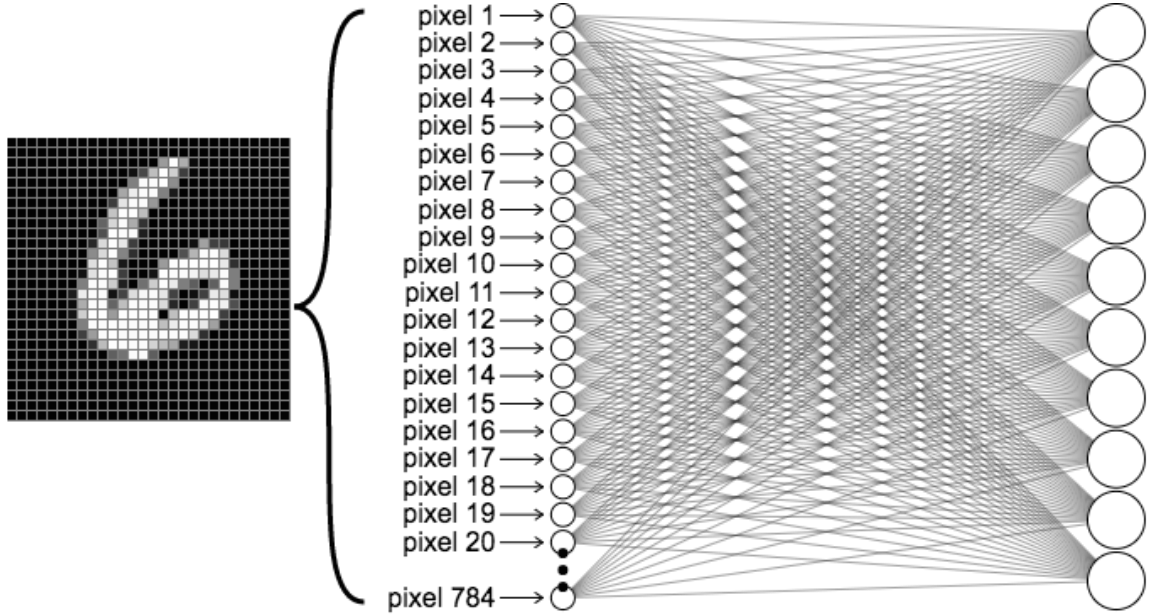
$$\boldsymbol{w^{(2)}} = \begin{pmatrix} w_{by}^2 & w_{1y}^2 & w_{2y}^2 \end{pmatrix}$$



$a()$: activation function – 활성화 함수

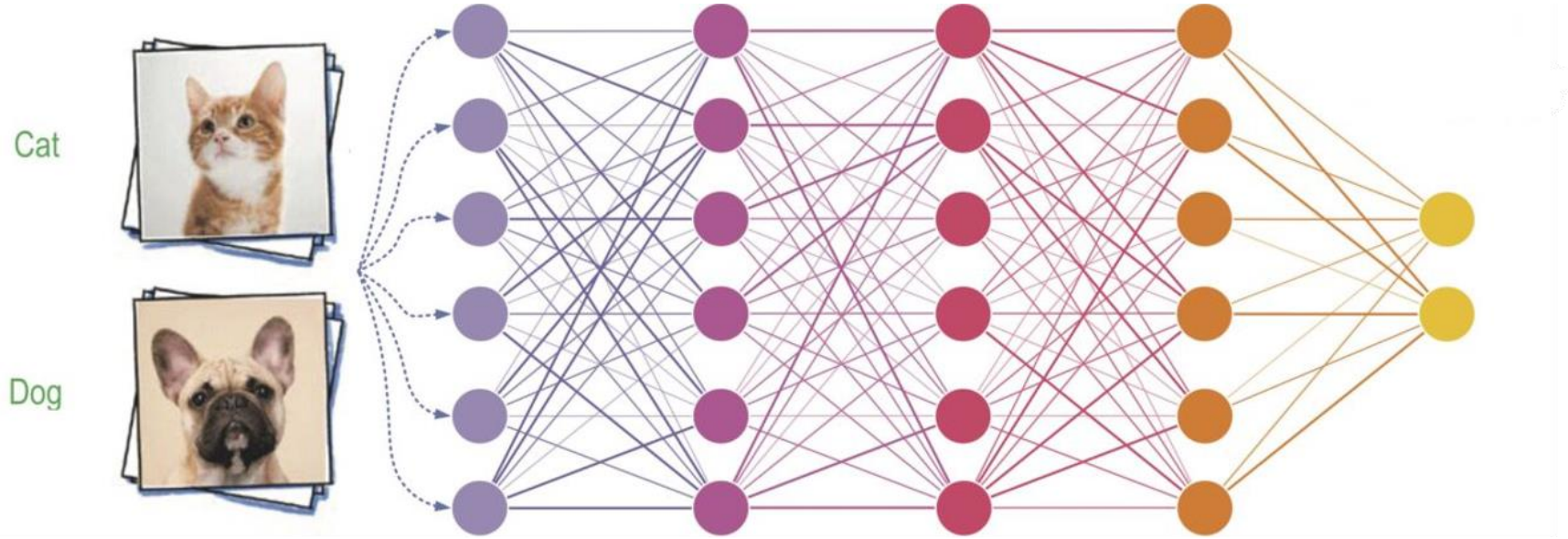
신경망 예제 1: MNIST hand-written numbers

MNIST Data Set: Input Vector가 사람이 쓴 손 글씨 형태이고, 해당 이미지를 알맞은 숫자에 **다중 분류(Multi Classification)**하는 신경망:



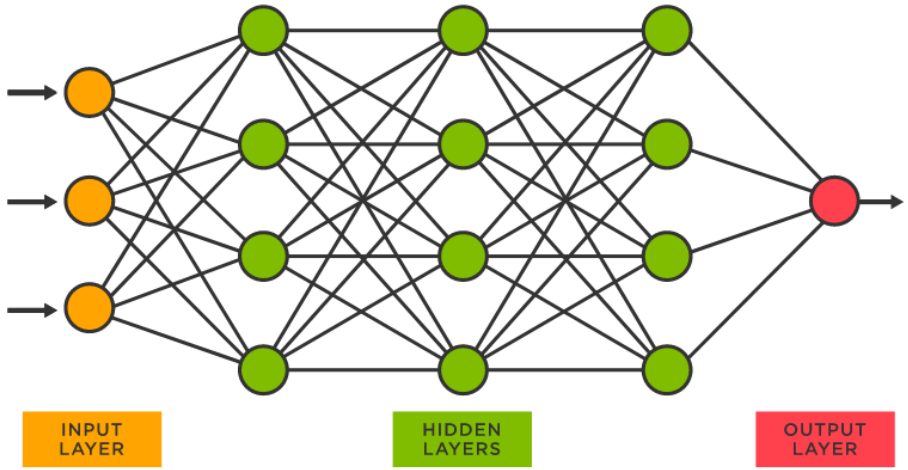
신경망 예제 2: Animal Classification

Input Vector가 동물 이미지의 형태이고, 해당 이미지를 고양이, 혹은 강아지로 **이진 분류(Binary Classification)**하는 신경망



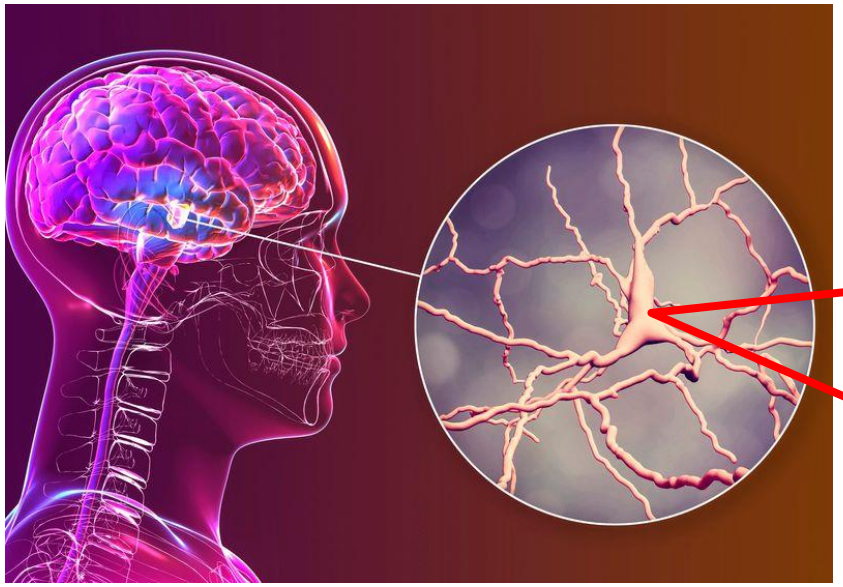
신경망 예제 3: Human Classification

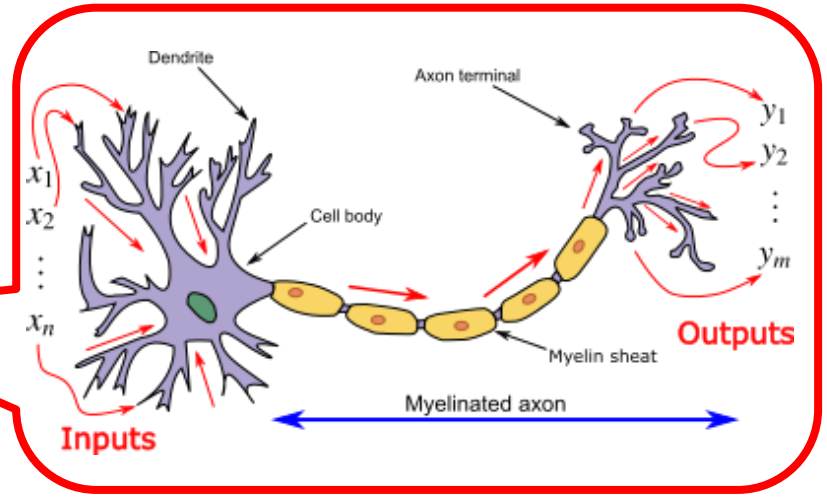
Input Vector가 사람 이미지의 형태이고, 해당 이미지를 **다중 분류(Multi Classification)**하는 신경망:

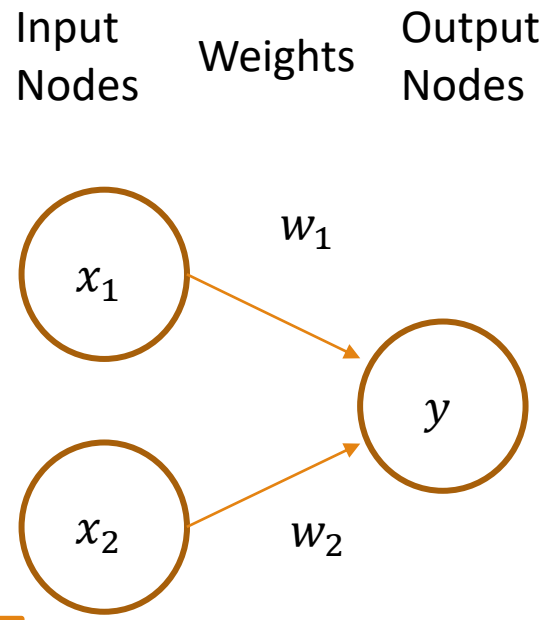


$$\begin{pmatrix} P(x = "Karina") \\ P(x = "Winter") \\ P(x = "NingNing") \\ P(x = "Giselle") \end{pmatrix}$$

Neuron이란?



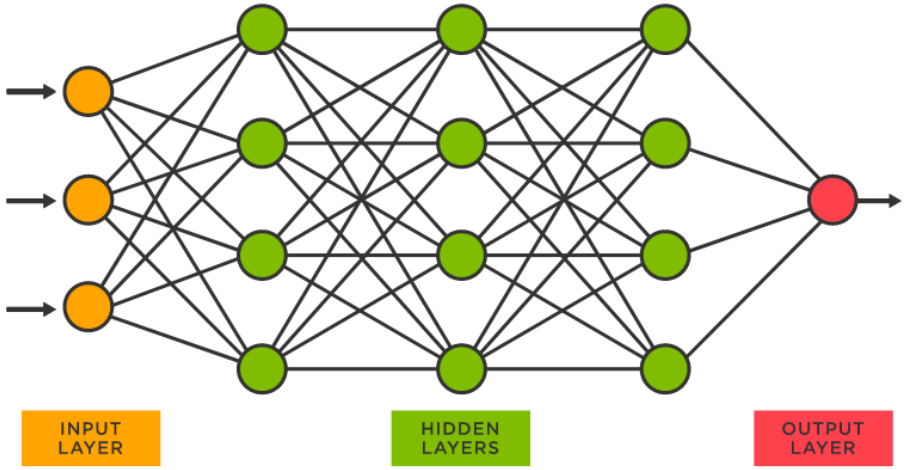




- 우리의 뇌는 Neuron이라는 신경세포로 이루어져 있다.
- Neuron은 dendrite가지돌기로 자극을 받아 axon축삭 을 통해 신호를 전달한다.
- 이 현상을 수학적으로 모델링한 것이 Perceptron이다.

Neural Network란?

- 우리의 뇌 처럼, Perceptron을 여러 번 연결하면 Multi-layer Perceptron이 된다 (Fully Connected Neural Network, Linear module 등, 다양한 이름으로 불린다).
- 이는 Perceptron만으로는 표현 불가능 한 non-linear classification을 가능하게 해준다.



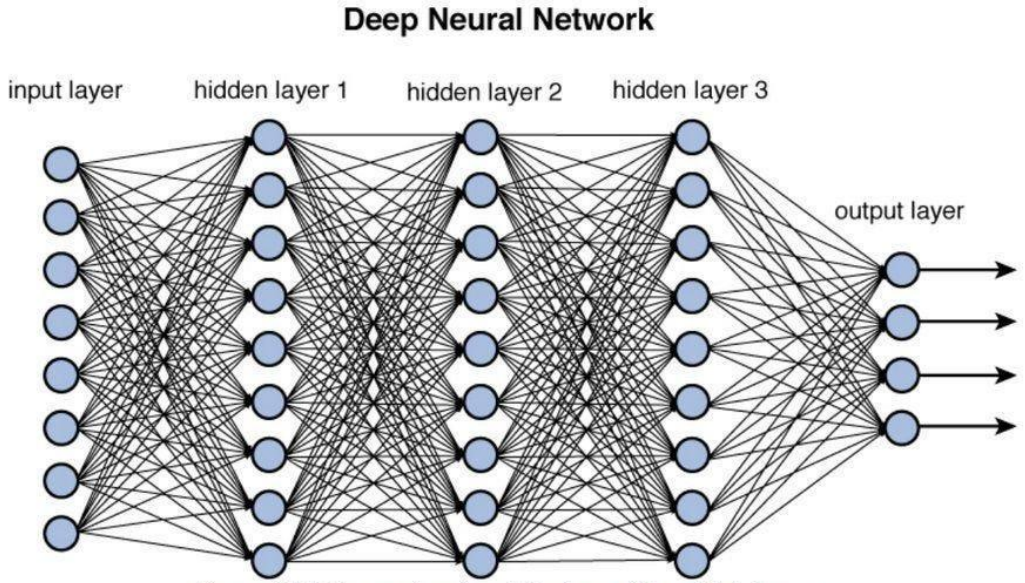
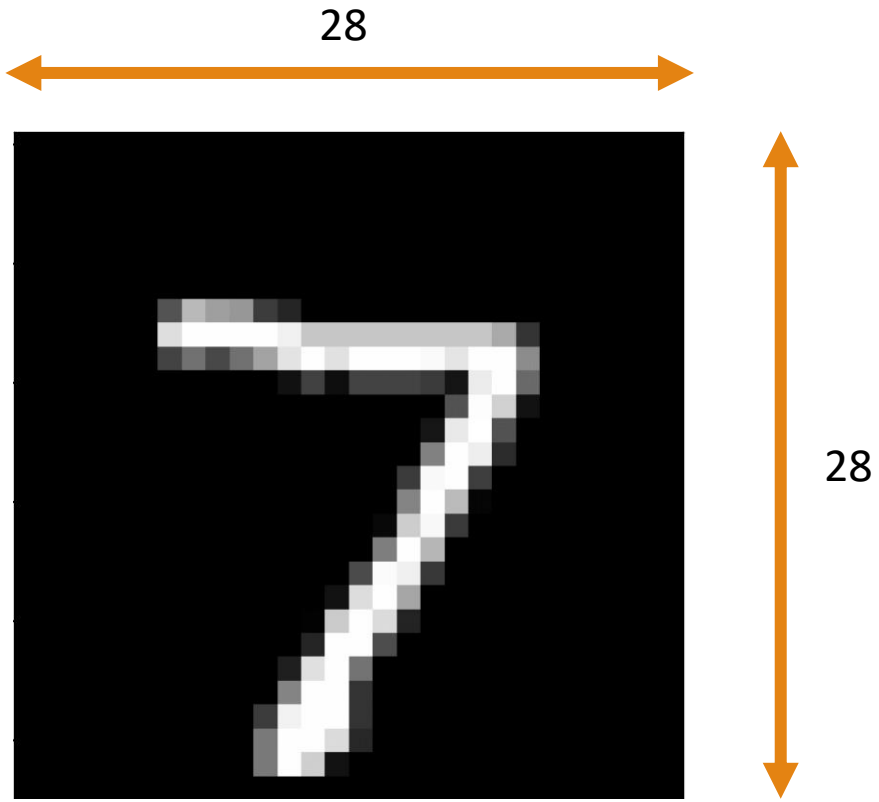
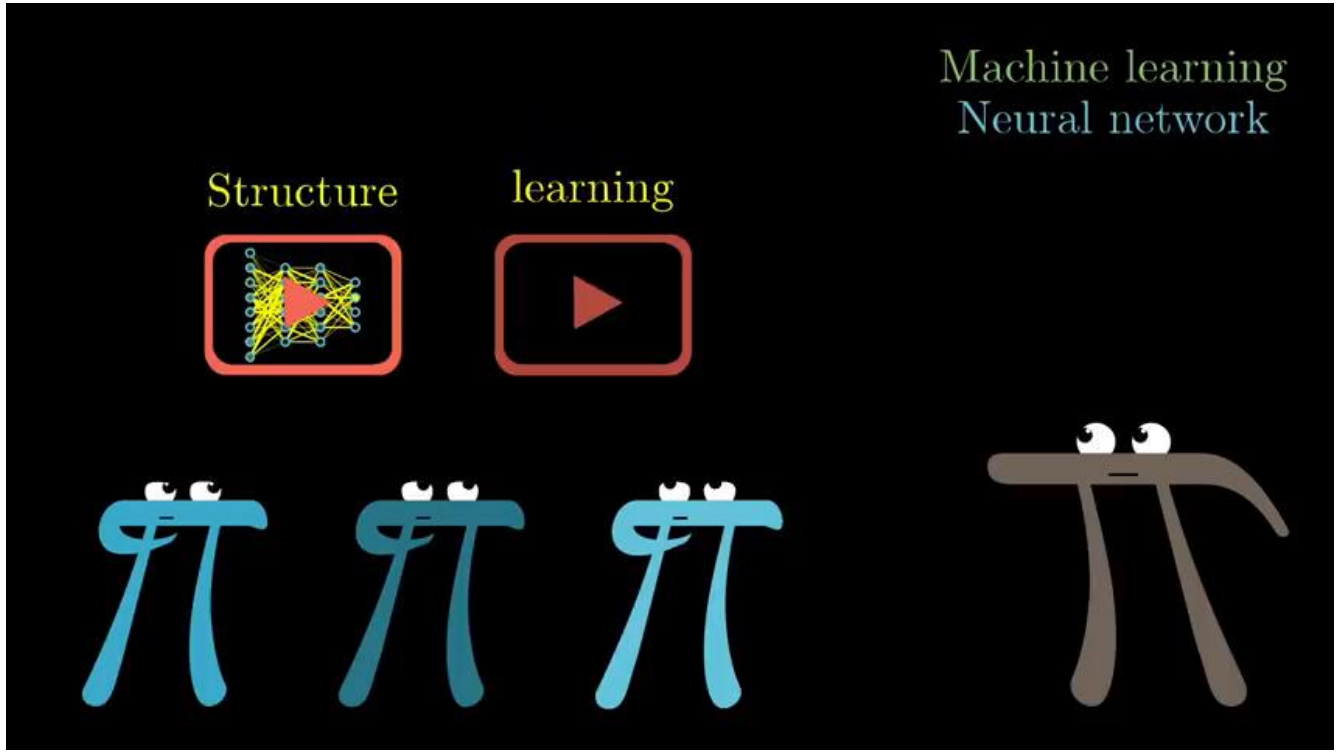


Figure 12.2 Deep network architecture with multiple layers.

Image Representation

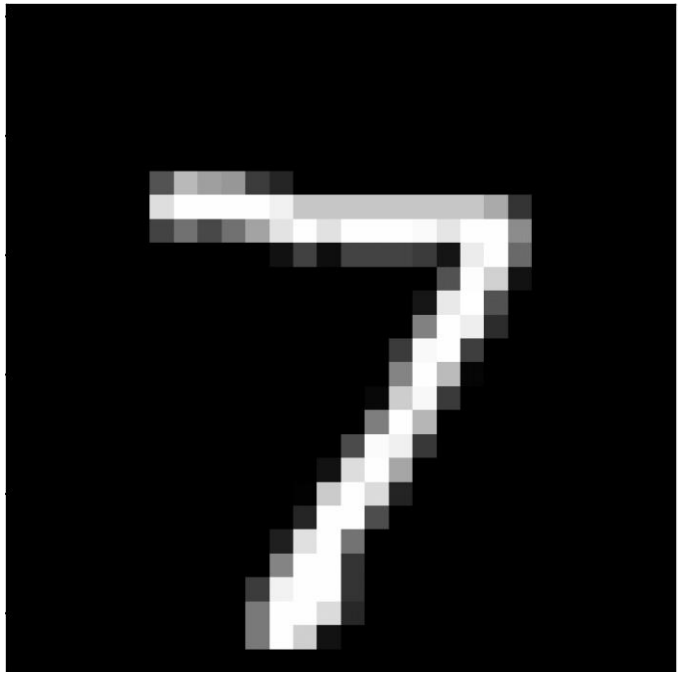




https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=1

Image Representation

Pixel-wise Representation



$$\begin{pmatrix} 0.2342 \\ 0.1235 \\ 1.3456 \\ 4.3156 \\ 0.0024 \\ 0.0255 \\ 1.1032 \\ 2.3109 \\ \vdots \\ 0.8724 \end{pmatrix}$$

- Pixel information(grey scale/RGB)이 하나의 column vector로 정렬된다.

장점:

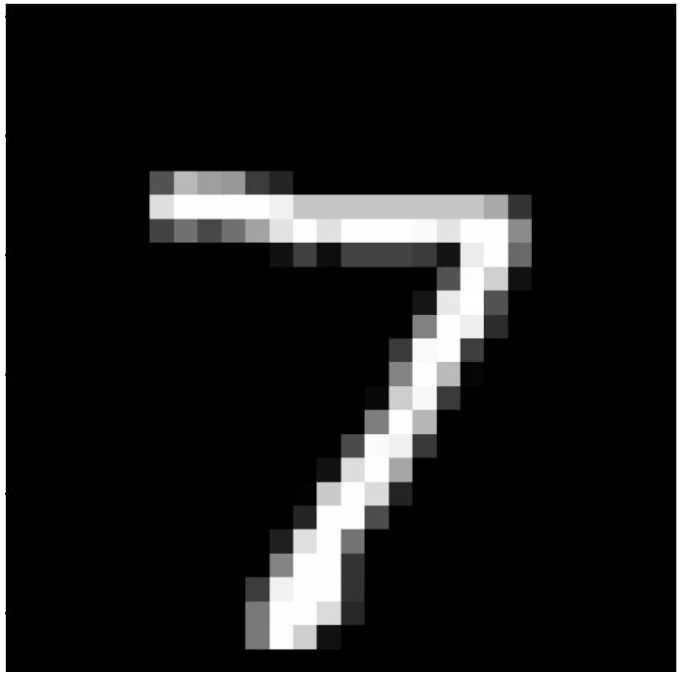
- 이미지로부터 추출하기가 매우 쉽다.
- 같은 row에 대한 정보는 vector상에서 근접한 요소들에 위치해 있다.

단점:

- 이미지가 가지고 있는 다양한 features에 대한 정보가 없다 – 정보의 손실.
  - Edges
  - 2D Shapes
  - Location of particular objects

Image Representation

Feature Representation



$$\begin{pmatrix} Horizontal\ Edges \\ Vertical\ Edges \\ Triangles \\ Circles \\ Sharp\ Corners \\ \vdots \\ Background \end{pmatrix}$$

- 이미지의 feature[특징]이 하나의 column vector로 정렬된다.

장점:

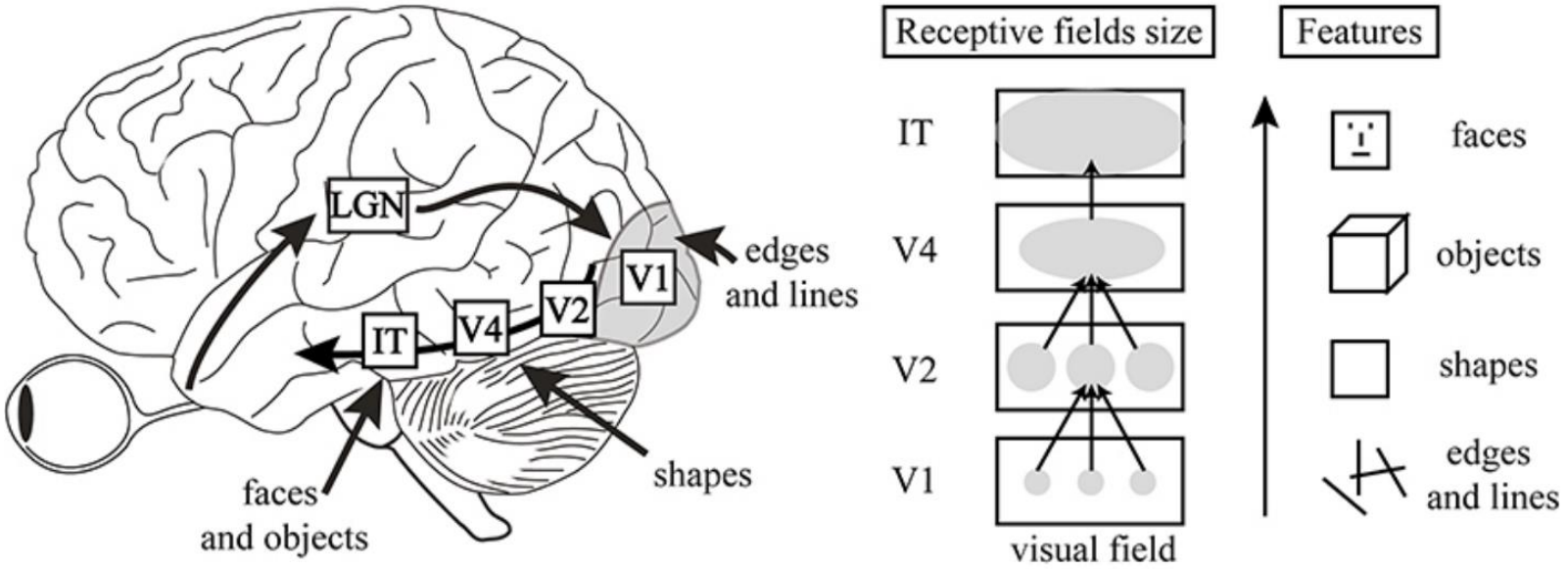
- 이미지가 가지고 있는 다양한 features에 대한 정보가 있다
  - Edges
  - 2D Shapes
  - Location of particular objects

단점:

- 해당 vector를 추출하기가 어렵다

How Human observe the world

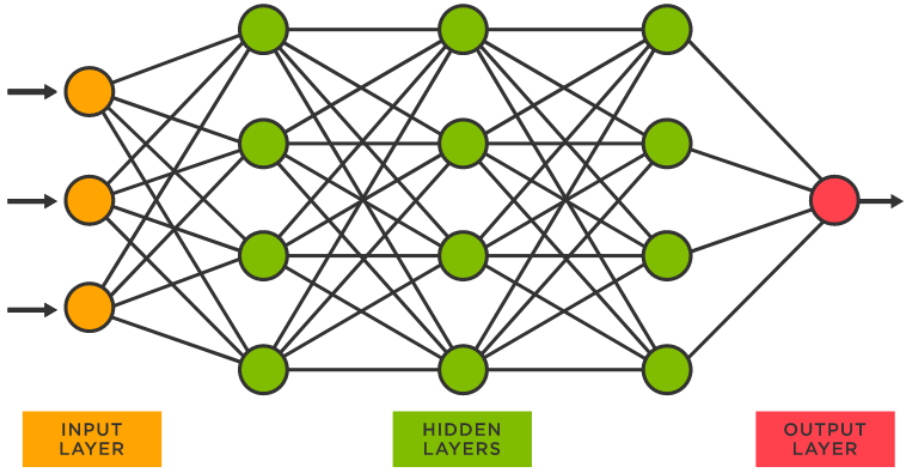
- 인간은 2차원 이미지/영상을 볼 때 자연스럽게 특징을 추출한다.
- Receptive Field수용장 이라는 작은 영역에서 특징을 추출한다.



Convolutional Neural Network (CNN)

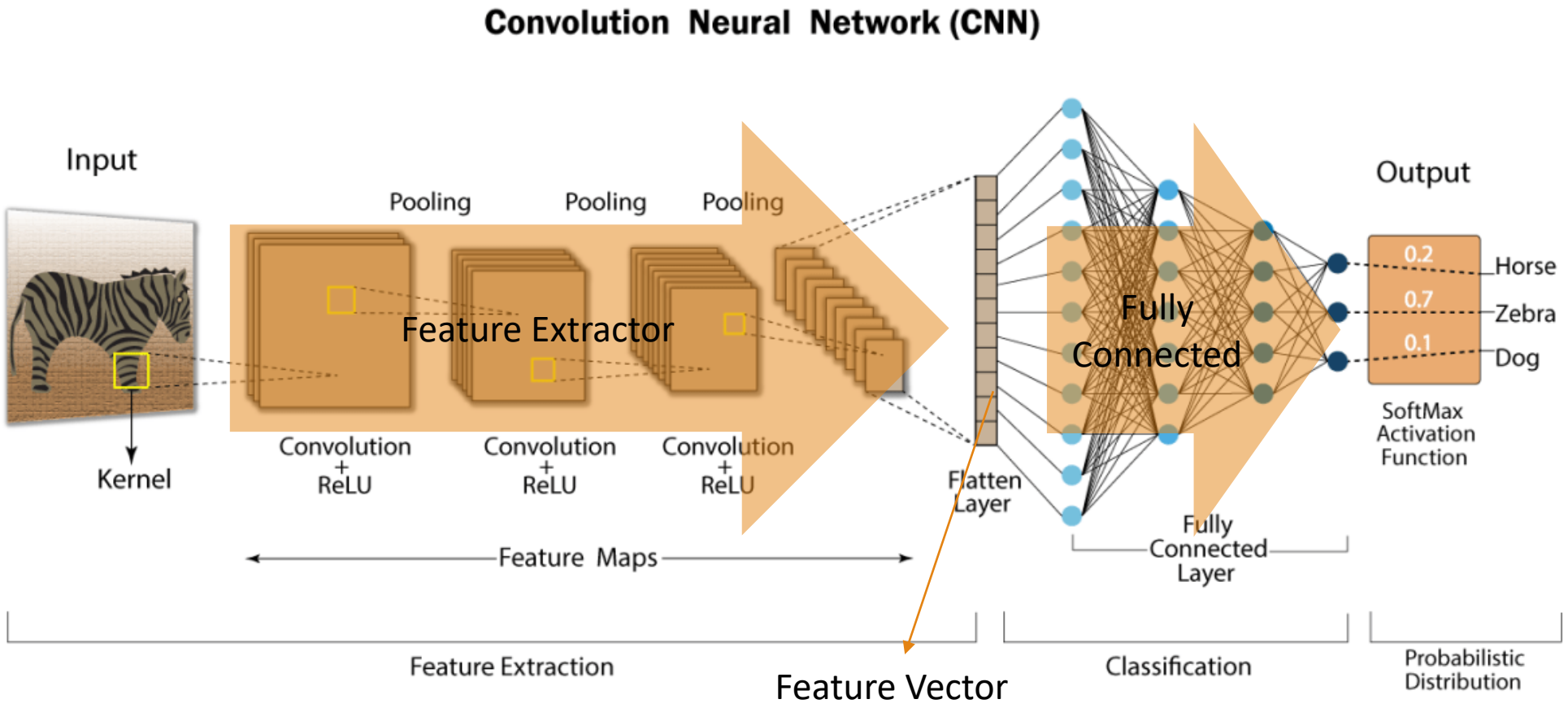
Convolutional Neural Network합성곱 신경망(CNN): 기존의 fully connected network에 입력 이미지의 전처리를 해주는 **feature extractor**를 추가한 인공신경망이다.

Feature Extractor



$$\begin{pmatrix} P(x = "Karina") \\ P(x = "Winter") \\ P(x = "NingNing") \\ P(x = "Giselle") \end{pmatrix}$$

Convolutional Neural Network (CNN)



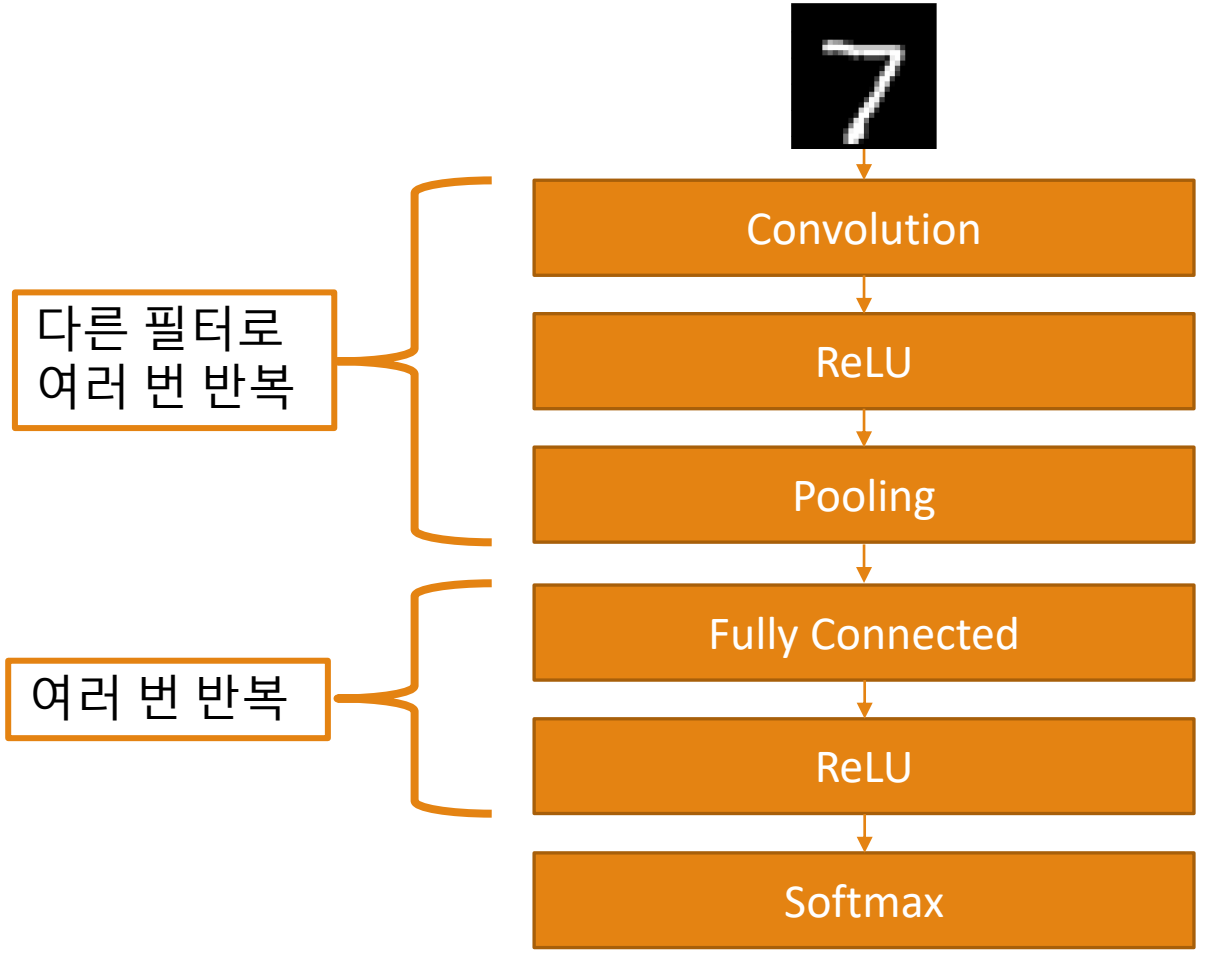
CNN Structure

CNN의 기본 구조:

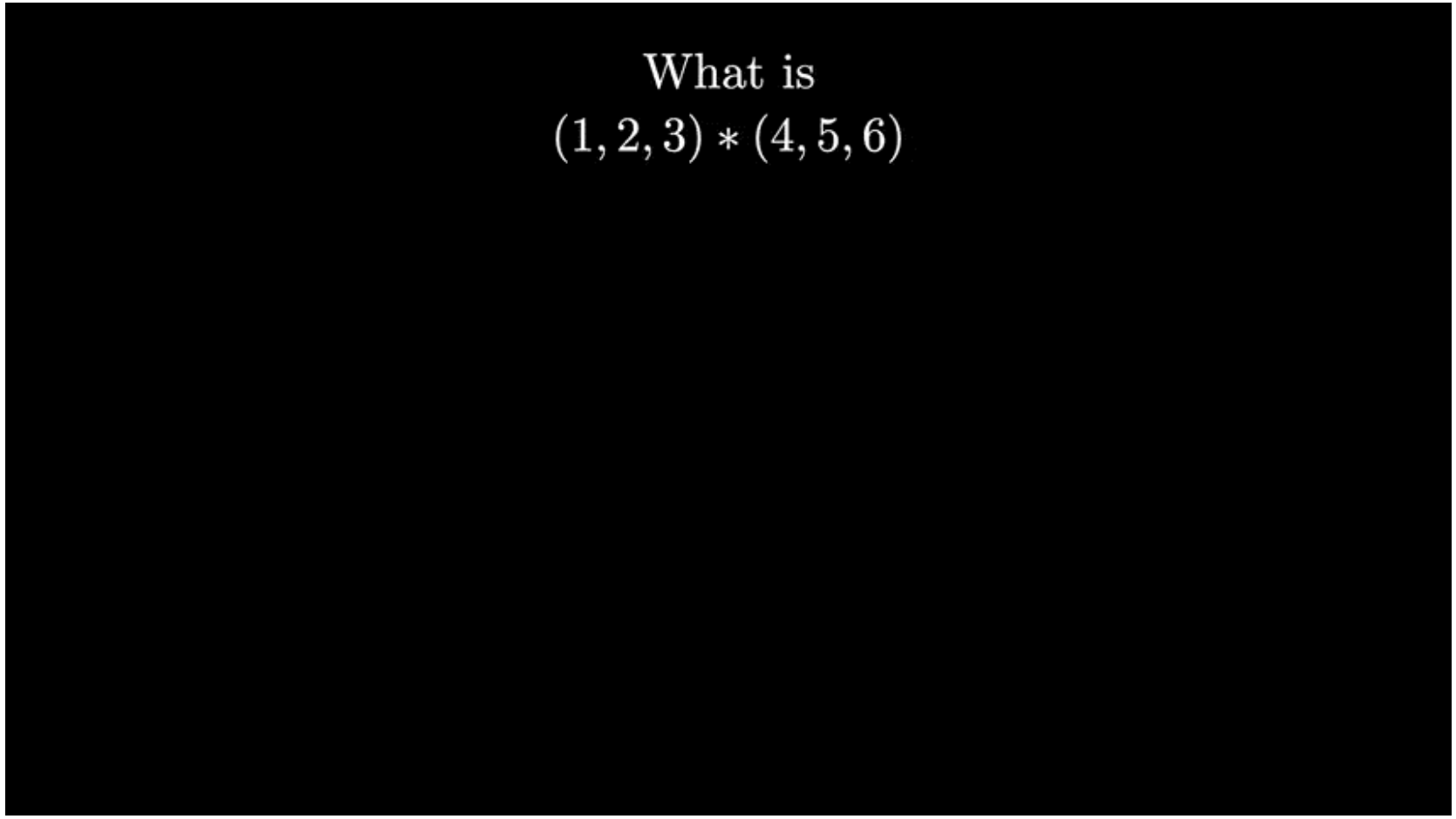
Feature Extractor:
Convolution → ReLU → Pooling(optional) 로 구성
Fully Connected:
Linear Layer → ReLU 로 구성

Convolution: 다양한 필터들로 이미지를 convolution하여 이미지의 각종 특징들을 추출.

Pooling(optional): Downsampling/subsampling의 일종 → 필요 이상으로 높은 이미지의 화질을 낮추고 계산을 가속화하는 연산 단계

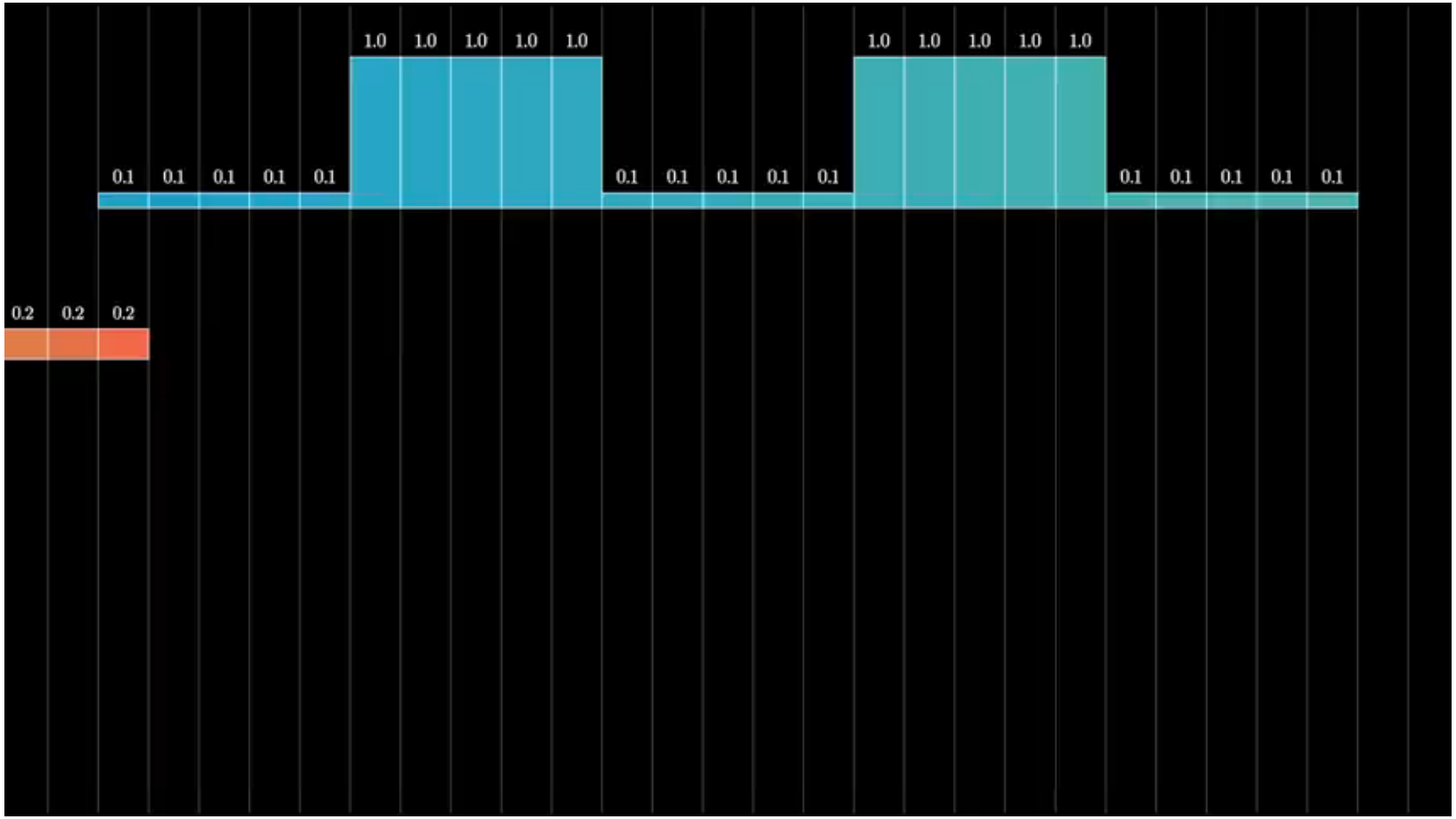


복습: Convolution



https://www.youtube.com/watch?v=KuXjwB4LzSA

복습: Convolution



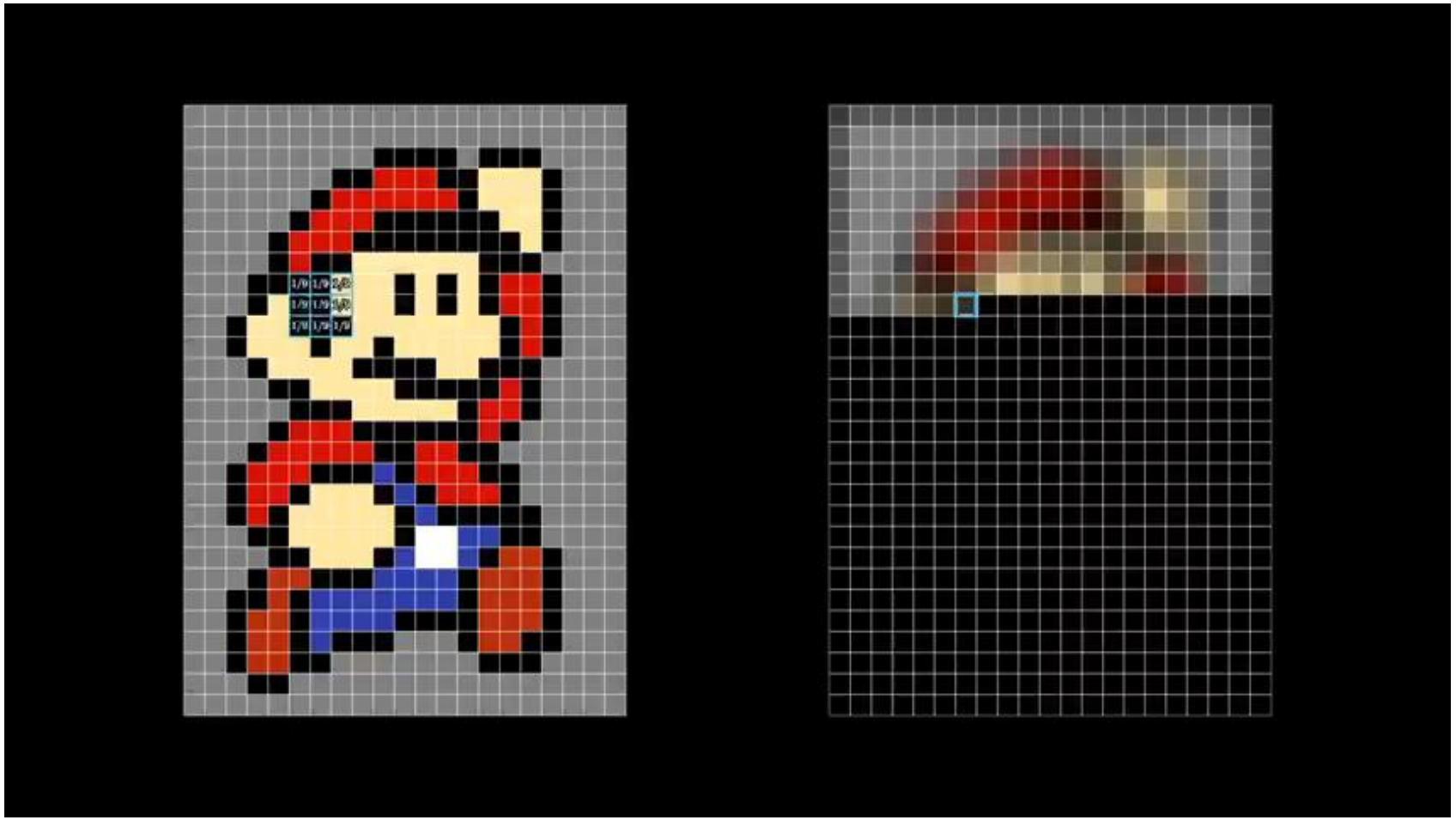
https://www.youtube.com/watch?v=KuXjwB4LzSA

Image Convolution

Convolution

$$* \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Image Convolution – Blurring



https://www.youtube.com/watch?v=KuXjwB4LzSA

Image Convolution – Blurring

Convolution

$$* \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

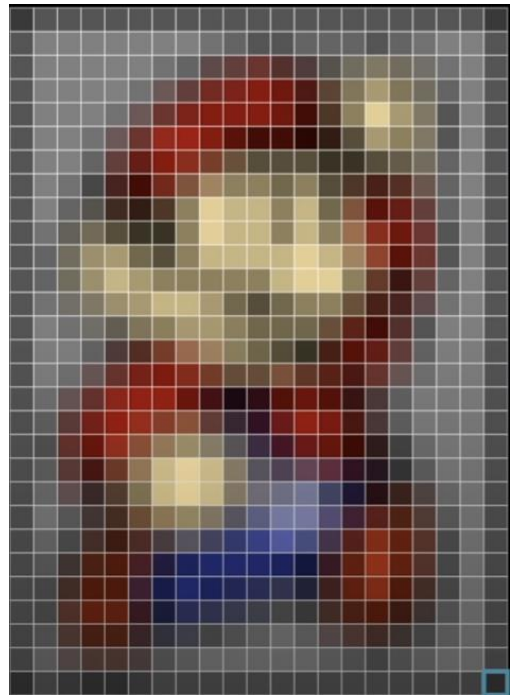


Image Convolution

$$* \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

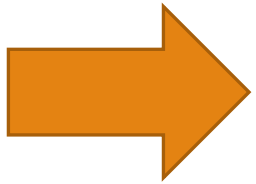


Image Convolution – Vertical Edge

$$* \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

이미지의 Vertical Edge 부분이 추출된다!

Image Convolution – Horizontal Edge

$$* \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

이미지의 Horizontal Edge 부분이 추출된다!

Image Convolution – Vertical Edge

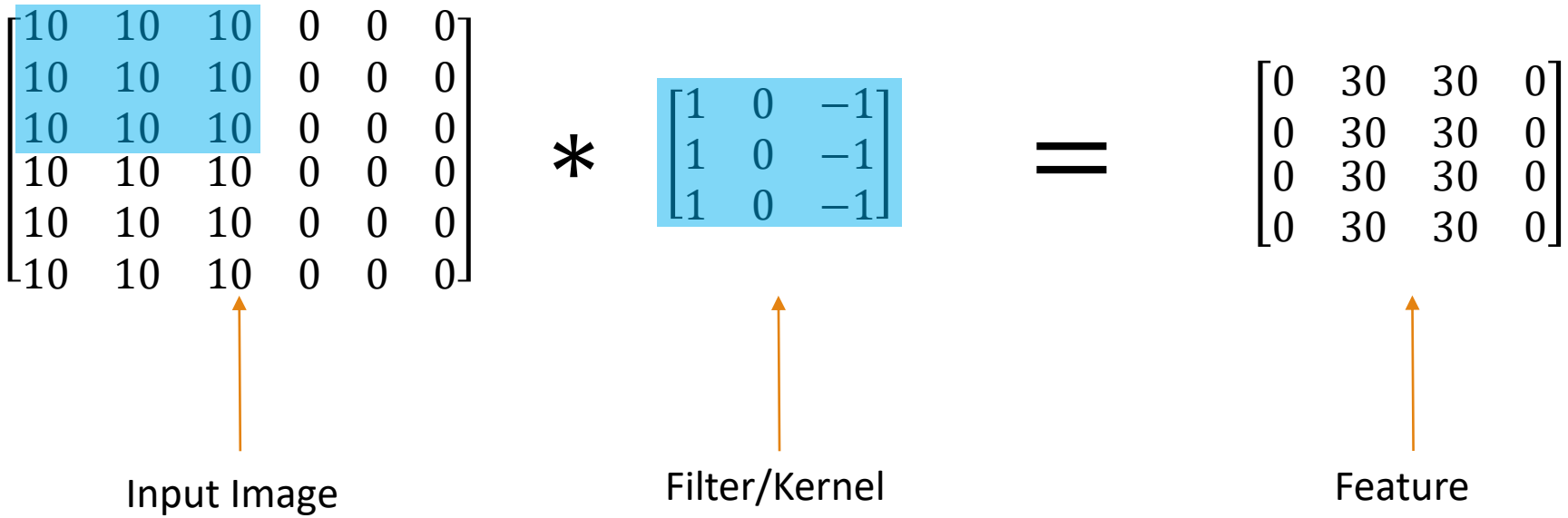


Image Convolution – Vertical Edge

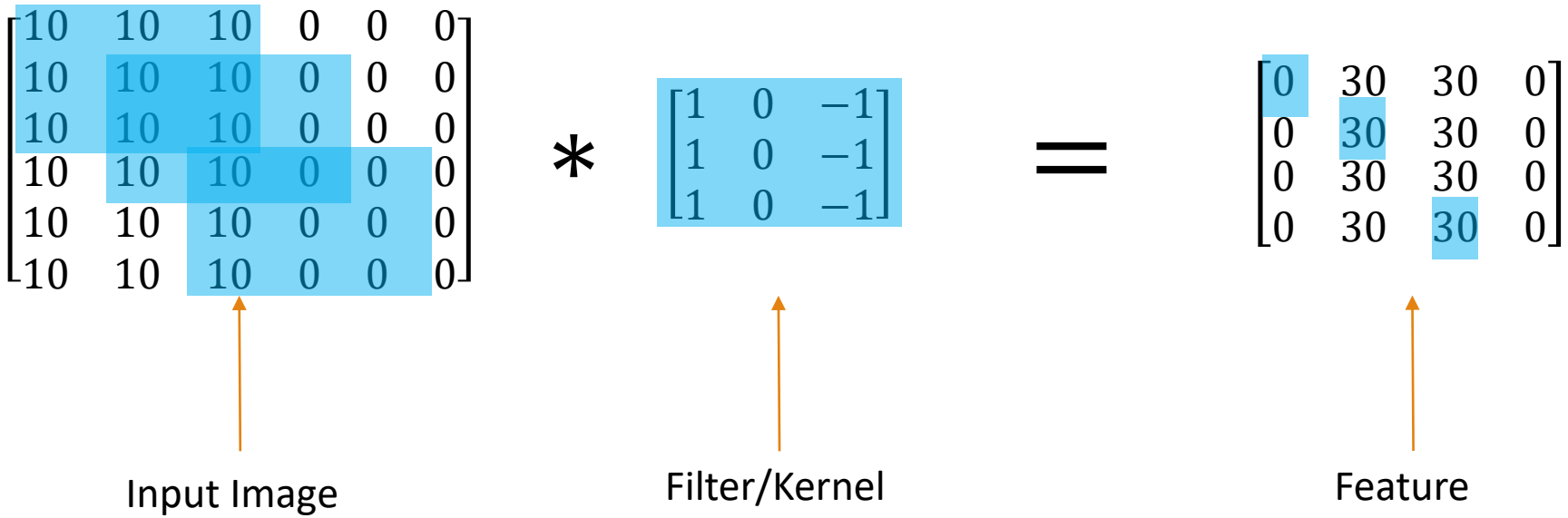


Image Convolution – Padding

간혹 우리는 출력의 형상을 조정 할 필요가 있다. 예: 입력 이미지와 같은 형상
Solution 1: 입력 이미지 외곽을 일정한 크기의 층으로 감싼다 → **Padding**
예: **Zero-Padding:** Padding된 층에 모든 값을 0으로 한다.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -20 & 0 & 20 & 20 & 0 & 0 \\ -30 & 0 & 30 & 30 & 0 & 0 \\ -30 & 0 & 30 & 30 & 0 & 0 \\ -30 & 0 & 30 & 30 & 0 & 0 \\ -30 & 0 & 30 & 30 & 0 & 0 \\ -20 & 0 & 20 & 20 & 0 & 0 \end{bmatrix}$$

Image Convolution – Padding

간혹 우리는 출력의 형상을 조정 할 필요가 있다. 예: 입력 이미지와 같은 형상
Solution 1: 입력 이미지 외곽을 일정한 크기의 층으로 감싼다 → **Padding**
예: **Replication-Padding / Symmetrical-Padding:** 가장 근접한 값으로 padding된다

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \end{bmatrix}$$

Image Convolution – Stride

간혹 우리는 출력의 형상을 조정 할 필요가 있다. 예: 작은 형상의 행렬
Solution 2: 필터의 적용 간격을 조정한다 → **Stride**
예: **Stride = 3**
경우에 따라 조정하는 Hyperparameter!

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Shape of the Output Image

Input Image Shape: (H, W)

Filter Shape: (FH, FW)

Padding Size: P

Stride: S

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

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

예: Input (4,4), Padding 1, Stride 1, Filter (3,3)

$$OH = \frac{H + 2P - FH}{S} + 1 = \frac{4 + 2 - 3}{1} + 1 = 4$$

$$OW = \frac{W + 2P - FW}{S} + 1 = \frac{4 + 2 - 3}{1} + 1 = 4$$

Shape of the Output Image

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \end{bmatrix}$$

예: Input (6,6), Padding 1, Stride 1, Filter (3,3)

$$OH = \frac{H + 2P - FH}{S} + 1 = \frac{6 + 2 - 3}{1} + 1 = 6$$

$$OW = \frac{W + 2P - FW}{S} + 1 = \frac{6 + 2 - 3}{1} + 1 = 6$$

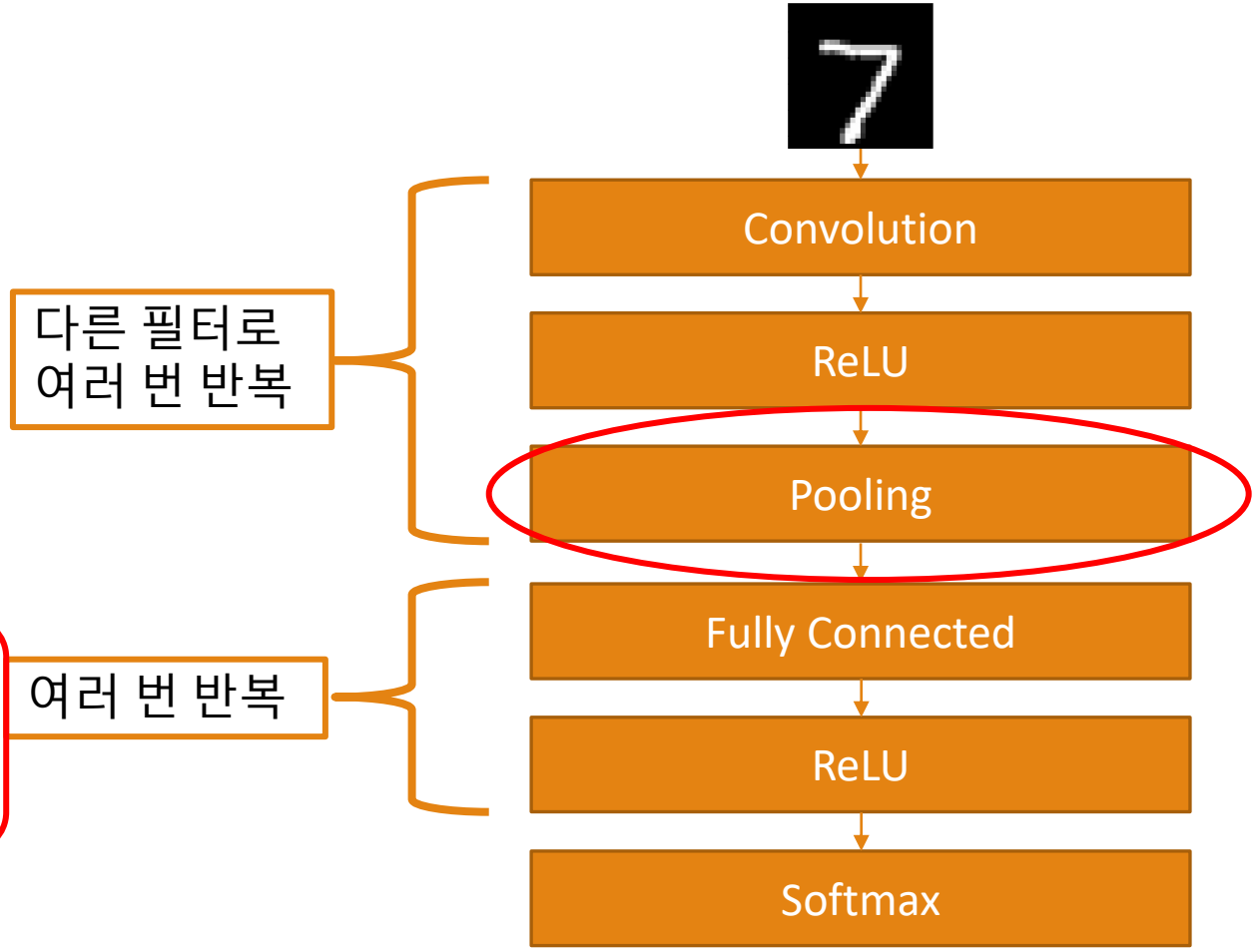
CNN Structure

CNN의 기본 구조:

Feature Extractor:
Convolution → ReLU → Pooling(optional) 로 구성
Fully Connected:
Linear Layer → ReLU 로 구성

Convolution: 다양한 필터들로 이미지를 convolution하여 이미지의 각종 특징들을 추출.

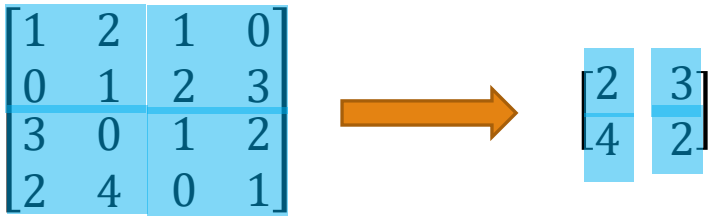
Pooling(optional): Downsampling/subsampling의 일종 → 필요 이상으로 높은 이미지의 화질을 낮추고 계산을 가속화하는 연산 단계



Pooling

Pooling: 세로/가로 방향의 공간을 줄이는 연산
Max Pooling: 특정 영역의 Maximum 값을 구하는 연산
Average Pooling: 특정 영역의 평균을 구하는 연산 (이미지 분야에서는 잘 안 쓰임)

예: Max Pooling



특징:
- 학습해야 할 parameter가 없다
- 입력의 변화(eg, noise)에 강건하다

Filters

Filter의 종류는 어떤 특징을 추출하고 싶은가에 따라서 종류가 너무나도 많고 다양하다.

그럼 어떤 필터들을 사용해야 할까?

정답: 우리가 직접 정할 필요가 없다! 컴퓨터가 알아서 학습하게 만들면 된다!

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\ x_{18} & x_{19} & x_{20} & x_{21} & x_{22} & x_{23} \\ x_{24} & x_{25} & x_{26} & x_{27} & x_{28} & x_{29} \\ x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} + [b]$$

Filter Weight Parameters

Bias Parameter

Summary

- 이미지를 잘 분석 하는 인공신경망을 만들기 위해서는 인간의 수용장처럼 특징을 추출하는 구조가 필요하다.
- 특징 추출은 convolution이라는 수학적 계산을 통해 이루어진다.
- Convolution에 필요한 필터는 인간이 직접 디자인 하는 것이 아닌 컴퓨터가 직접 학습을 하게 만든다. → Next Lecture