Introduction to CNN & RNN

2020310634 수학과 이호준

Contents

0. Review

0. Review

Data

$$x^{(1)} = [x_1^{(1)} \quad \cdots \quad x_n^{(1)}]$$
$$x^{(2)} = [x_1^{(2)} \quad \cdots \quad x_n^{(2)}]$$
$$\vdots$$
$$x^{(d)} = [x_1^{(d)} \quad \cdots \quad x_n^{(d)}]$$

Bias

$$b = [b^{(1)} \quad \cdots \quad b^{(m)}]$$

Weights

$$w^{(1)} = \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_n^{(1)} \end{bmatrix}$$
$$w^{(2)} = \begin{bmatrix} w_1^{(2)} \\ \vdots \\ w_n^{(2)} \end{bmatrix}$$
$$\vdots$$
$$w^{(m)} = \begin{bmatrix} w_1^{(m)} \\ \vdots \\ w_n^{(m)} \end{bmatrix}$$

Matrix Form

$$X = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(d)} & \cdots & x_n^{(d)} \end{bmatrix}$$

$$W = \begin{bmatrix} w_1^{(1)} & \cdots & w_1^{(m)} \\ \vdots & \ddots & \vdots \\ w_n^{(1)} & \cdots & w_n^{(m)} \end{bmatrix}$$

$$b = [b^{(1)} \quad \cdots \quad b^{(m)}]$$

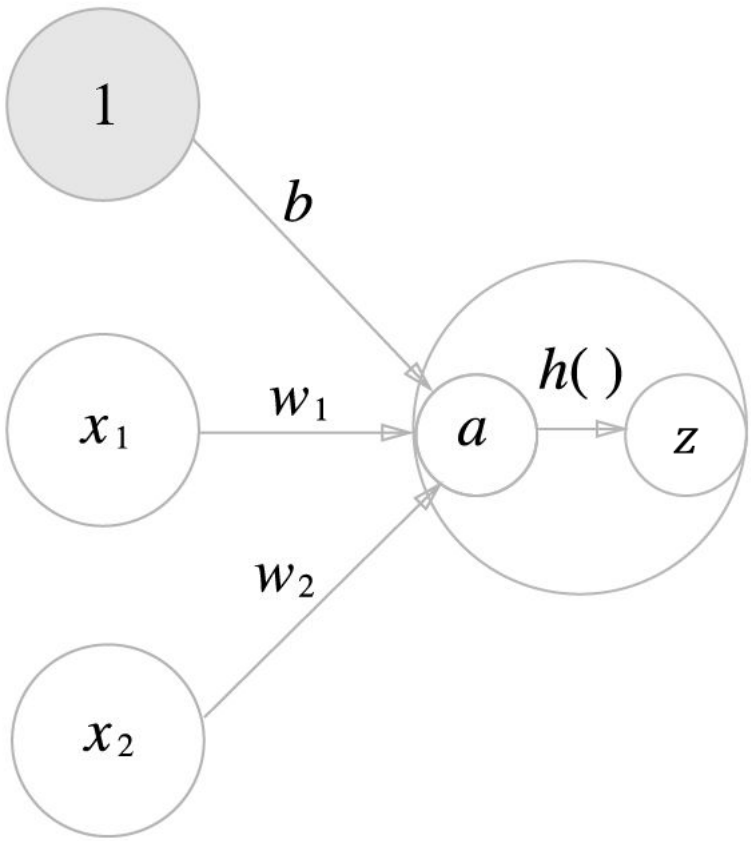
0. Review

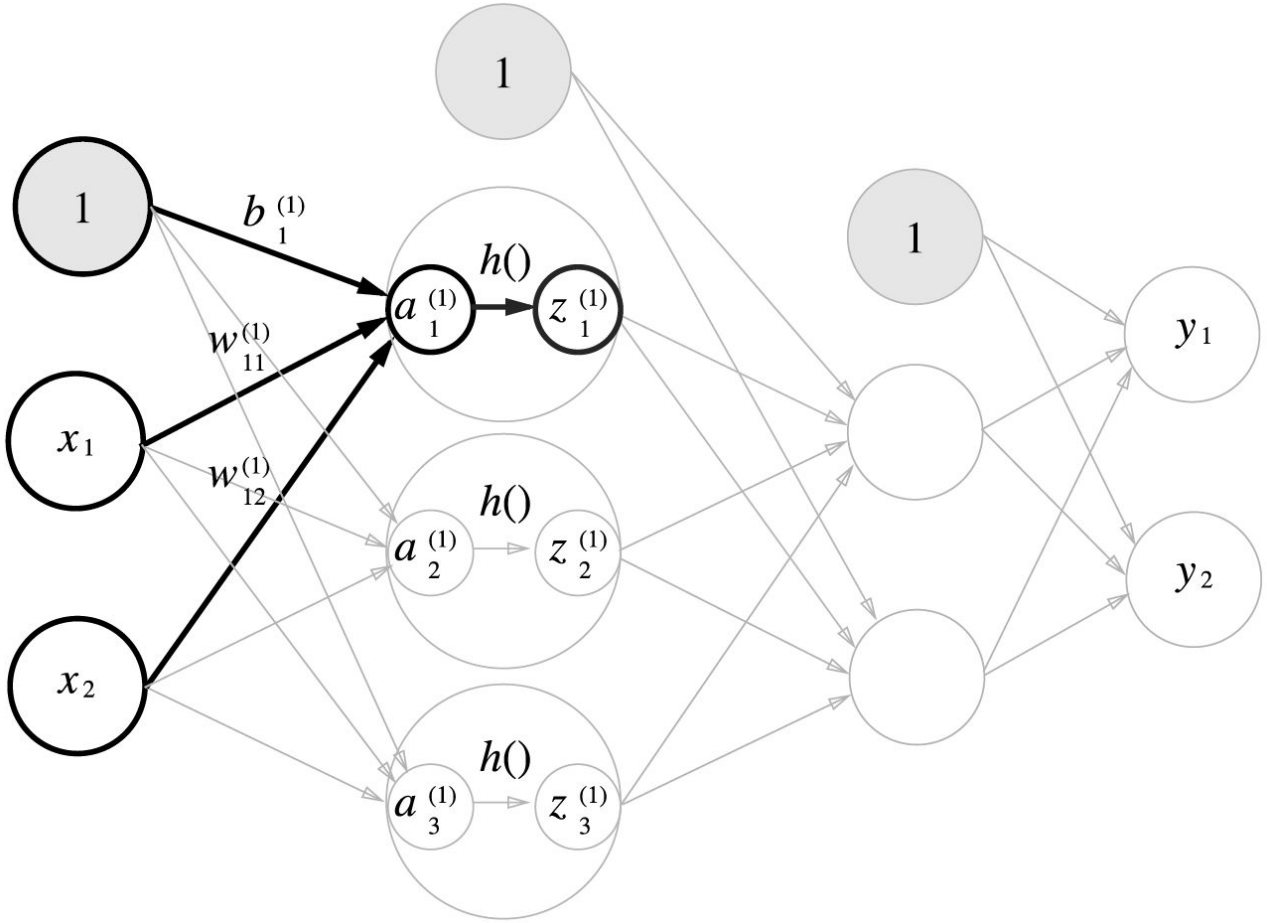
Affine Transform

$$A = XW + b$$

Activation Functions

- Sigmoid: $f(x) = \frac{1}{1 + e^{-x}}$
- ReLU: $f(x) = \max\{0, x\}$
- Softmax: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$


1
b
x1
w1
a
h( )
z
w2
x2


1
1
b (1) 1
h()
a (1) 1
z (1) 1
w(1) 11
x1
w(1) 12
h()
a (1) 2
z (1) 2
x2
h()
a (1) 3
z (1) 3
1
y1
y2

0. Review

Loss function

$$L = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$ Sum of squares for error (SSE)

$$L = -\sum_{k} t_k \log y_k$$ Cross entropy error (CEE)

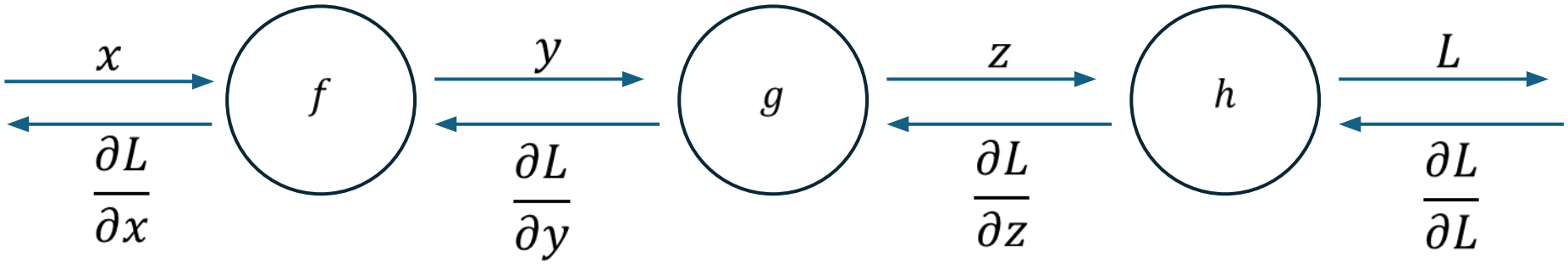
Gradient Descent

$$W = W - \alpha \frac{\partial L}{\partial W}$$

$\alpha: learning\ rate$
$L: Loss\ function$

0. Review

Backpropagation

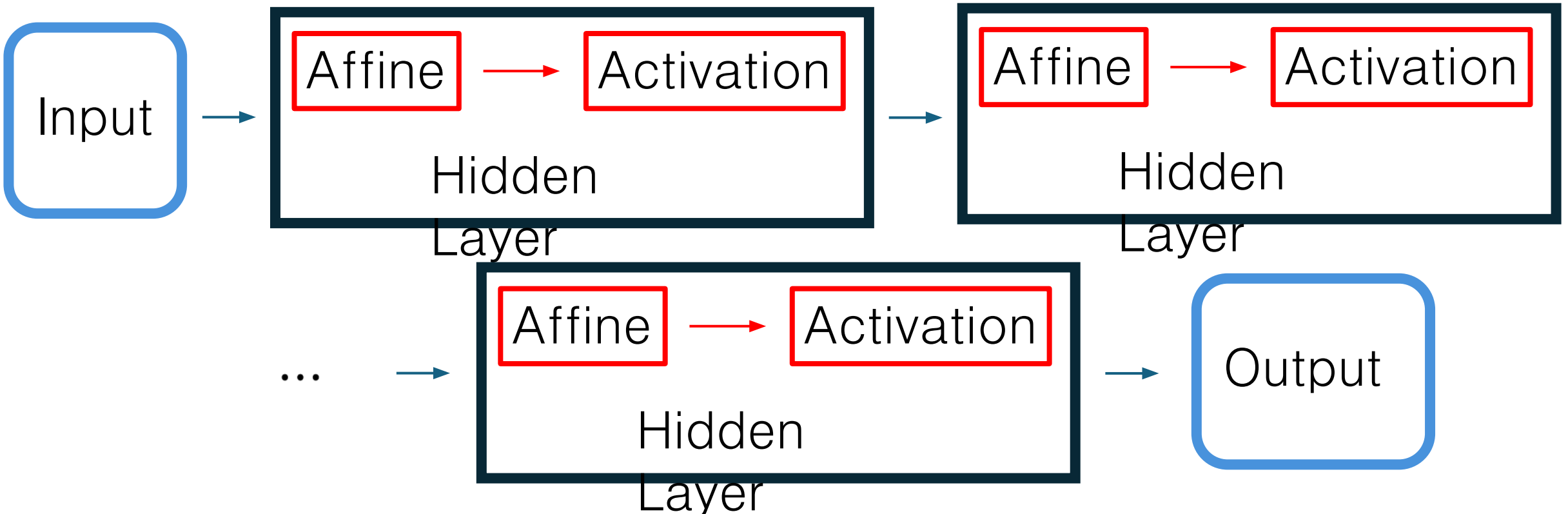


$$= \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \qquad = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$$

0. Review

Fully Connected Neural Network (or MLP: Multi-Layer Perceptron)

Affine Layer -> Activation Function



1. CNN

Convolutional **N**eural **N**etwork

1-1. Convolution Layer

Convolution (1-dim)

$$(y * w)(t) = \sum_{\tau=-\infty}^{\infty} y(\tau)w(t-\tau)$$

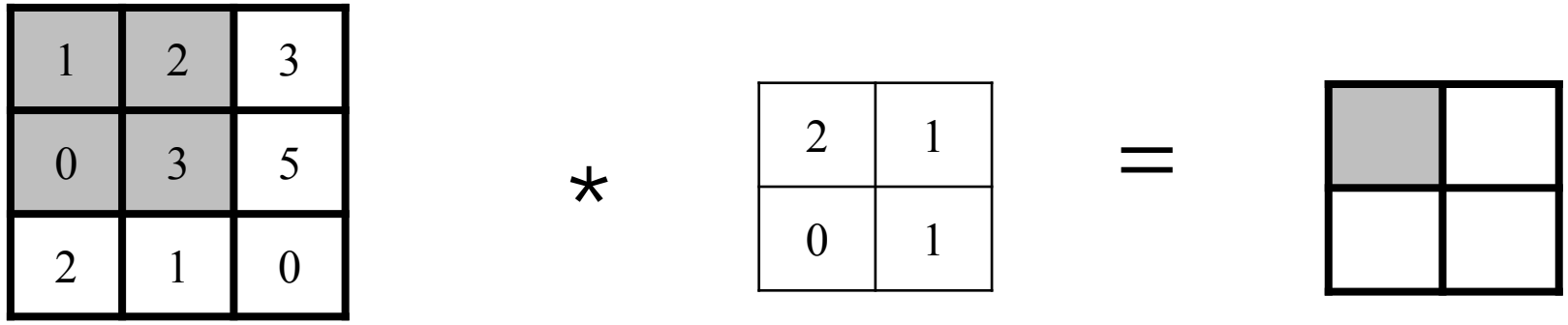
$$(y * w)(t) = \int_{-\infty}^{\infty} y(\tau)w(t-\tau)d\tau$$

We use (discrete) Convolution (2-dim)

$$(X * W)(i,j) = \sum_{u=0}^{FH-1} \sum_{v=0}^{FW-1} X(i+u, j+v)W(u,v)$$

1-1. Convolution Layer

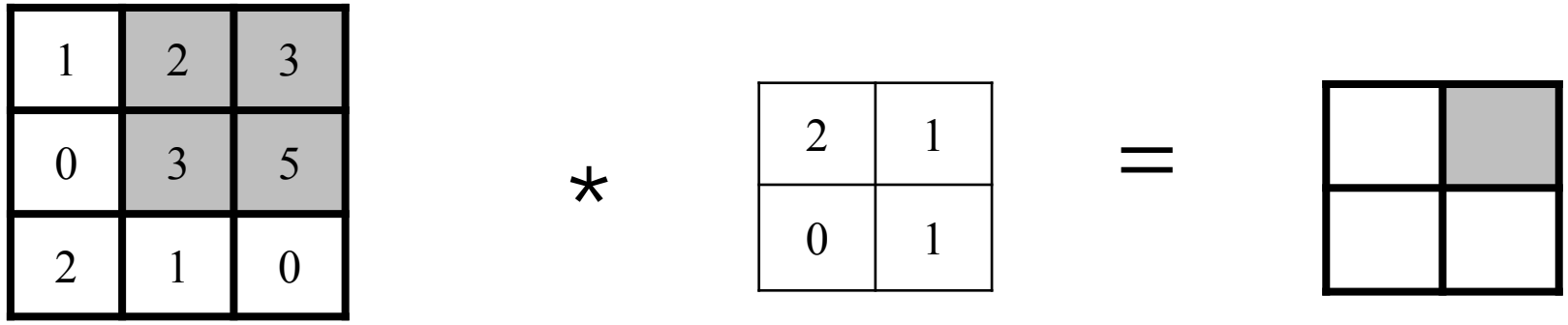
Example: one matrix X and one W



X W

1-1. Convolution Layer

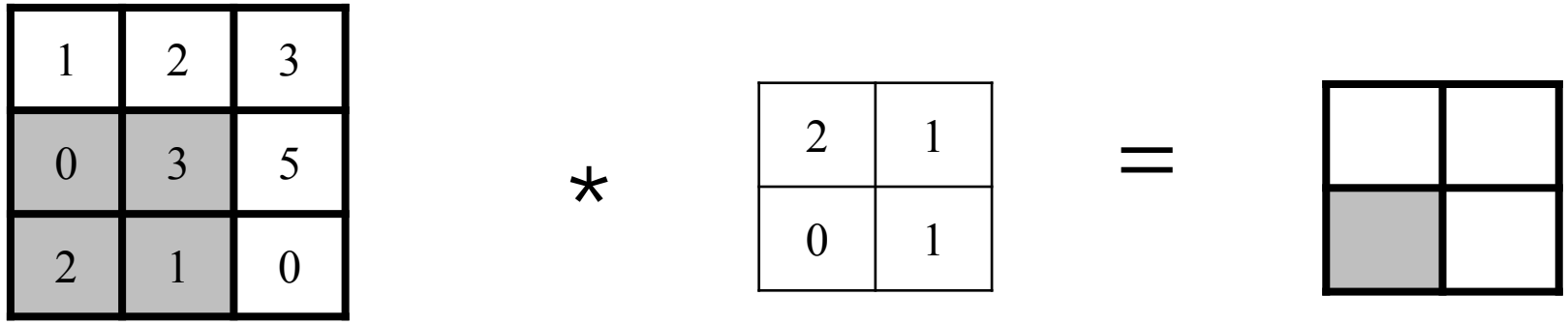
Example: one matrix X and one W



X W

1-1. Convolution Layer

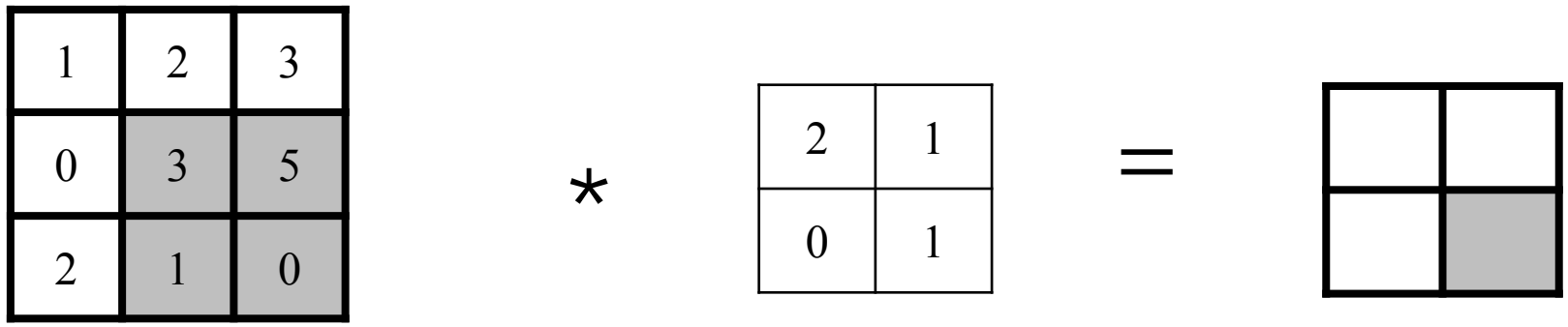
Example: one matrix X and one W



X W

1-1. Convolution Layer

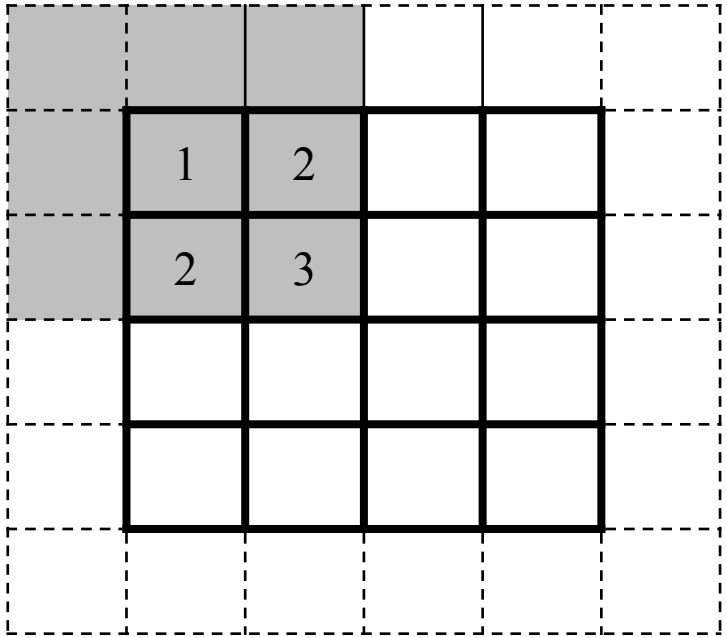
Example: one matrix X and one W



X W

1-1. Convolution Layer

Padding



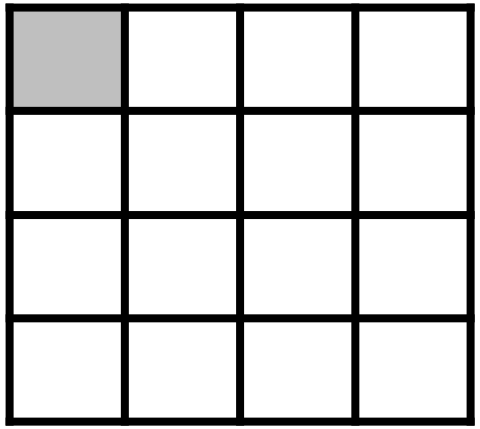
X

$*$

1	2	3
0	1	0
0	2	4

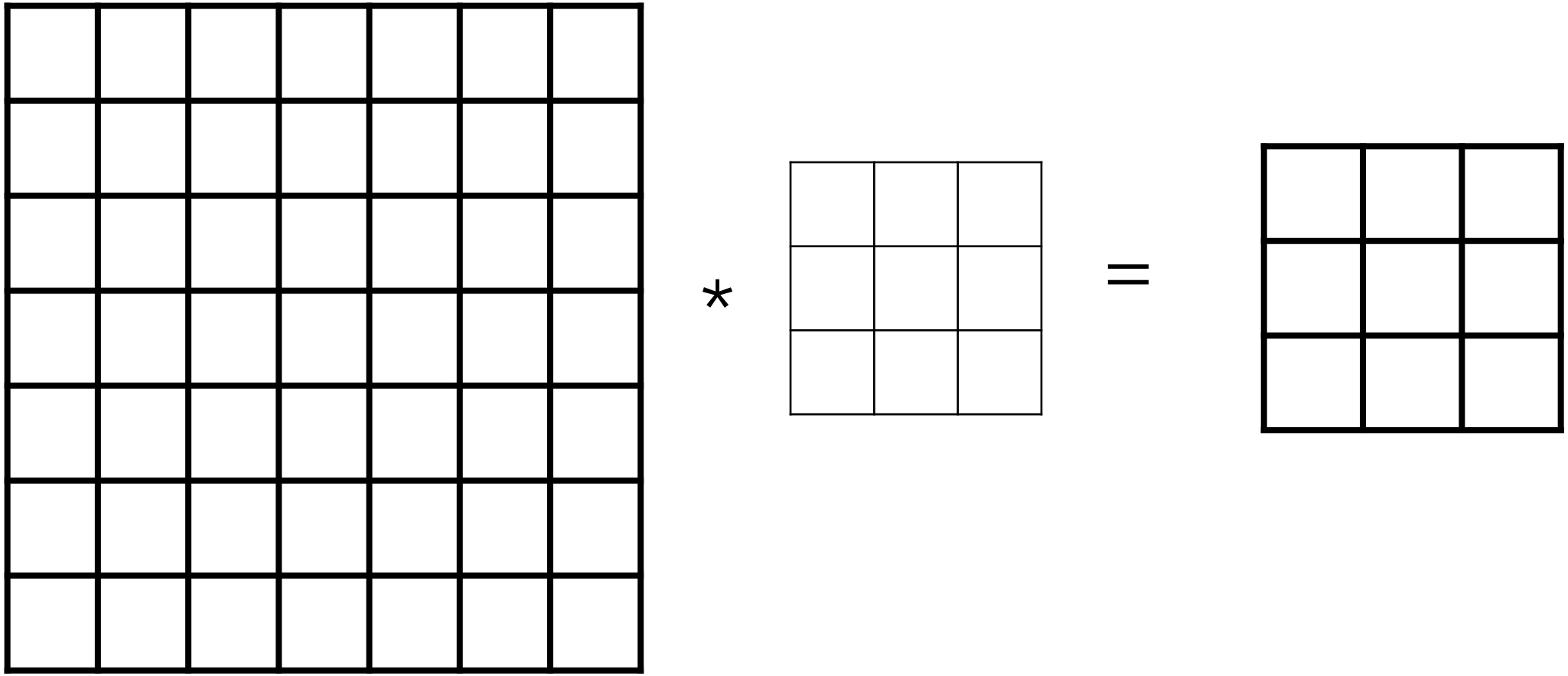
W

$=$



1-1. Convolution Layer

Stride



X

$*$

W

$=$

1-1. Convolution Layer

Size of the Output

$$(X * W)(i,j) = \sum_{u=0}^{FH-1} \sum_{v=0}^{FW-1} X(i+u, j+v) W(u,v)$$

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

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

X is $H \times W$ matrix

W is $FH \times FW$ matrix

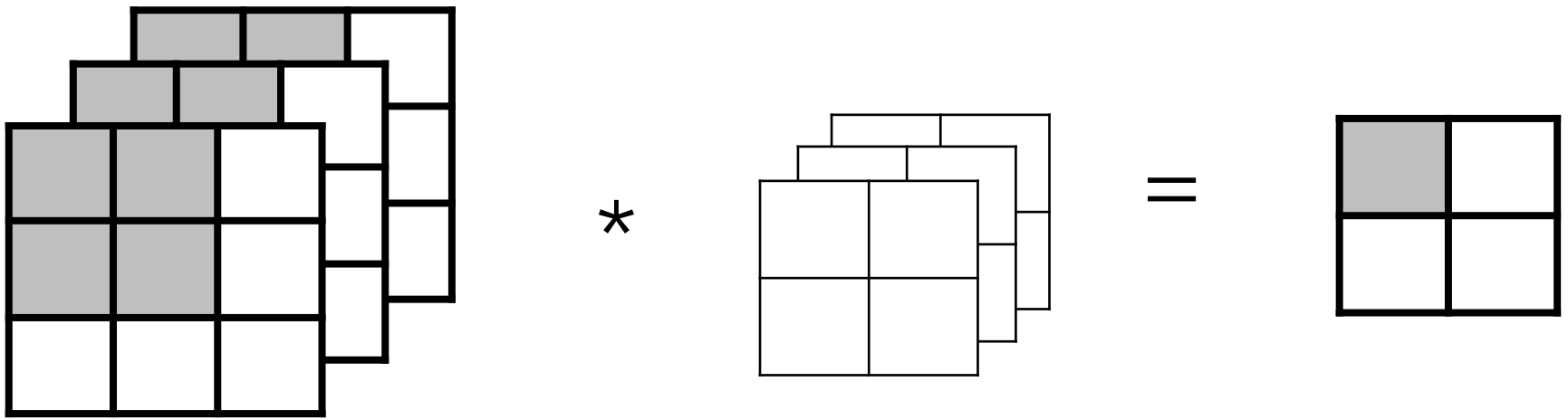
P is Padding

S is Stride.

1-1. Convolution Layer

Increasing the Number of Channels

$$(X * W)(i,j) = \sum_{c=0}^{C-1} \sum_{u=0}^{FH-1} \sum_{v=0}^{FW-1} X(c, i+u, j+v) W(c,u,v)$$

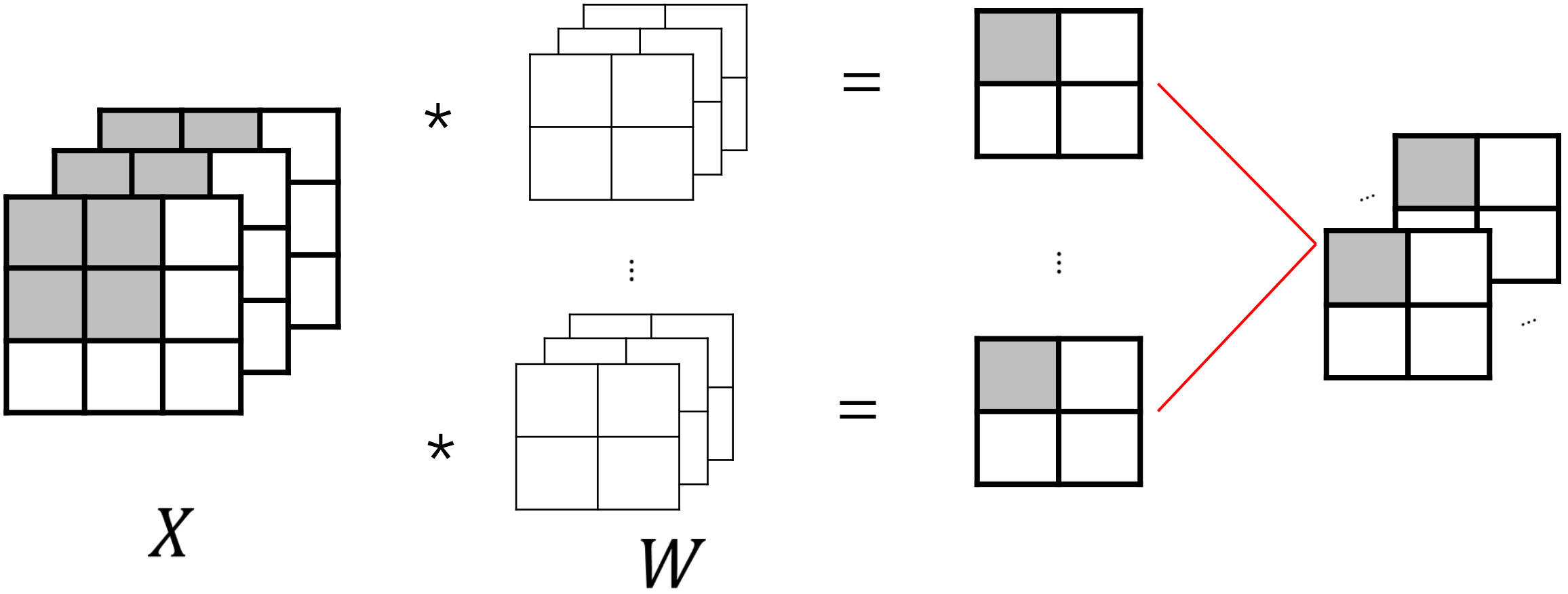


X W

1-1. Convolution Layer

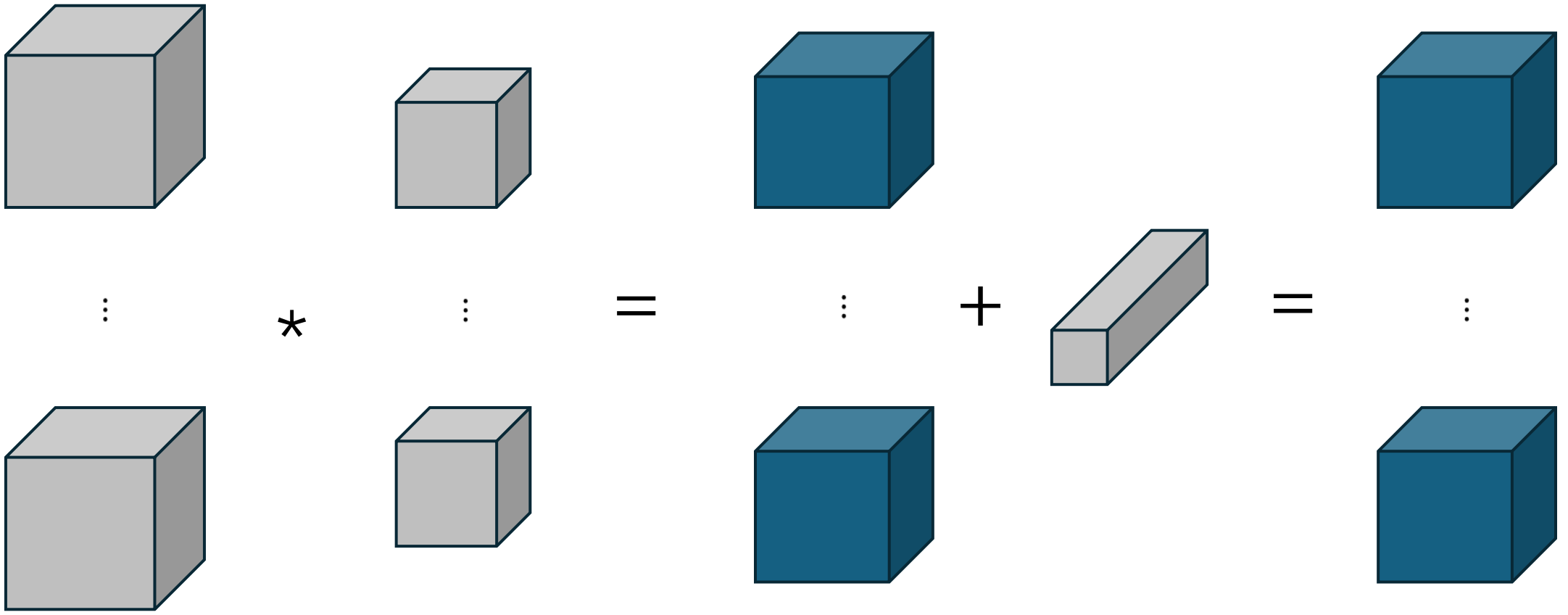
Increasing the Number of Filters

FN-times $(X * W)(i,j) = \sum_{c=0}^{C-1} \sum_{u=0}^{FH-1} \sum_{v=0}^{FW-1} X(c, i+u, j+v) W(c,u,v)$



1-1. Convolution Layer

Batch Processing



X W outputs b outputs

1-2. Pooling Layer

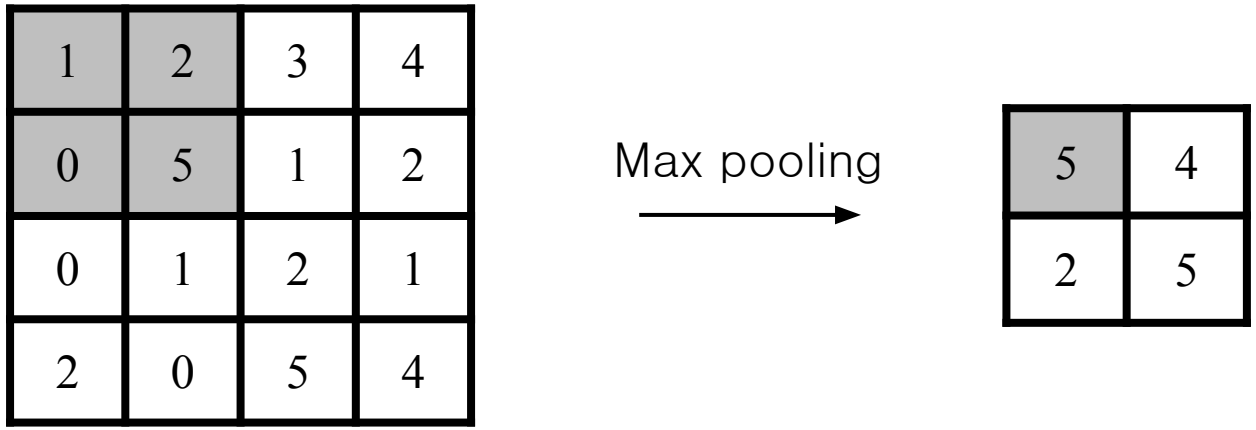
Pooling

- Max Pooling
- Average Pooling

Mathematical effect

- Downsampling
- Translation Invariance

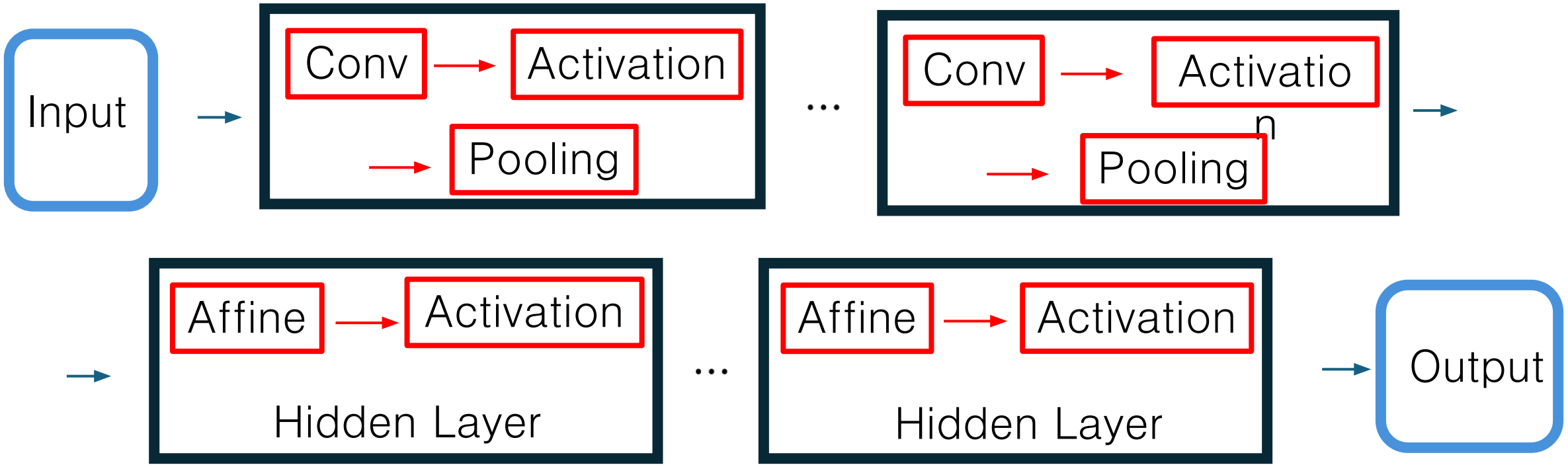
Ex) Pooling size 2x2



1-3. Structure of CNN

Convolutional **N**eural **N**etwork

Convolution Layer -> Activation (-> Pooling): Feature Extraction
+ Affine -> Activation: Classification



1-3. Structure of CNN

Gradient Descent

$$W = W - \alpha \frac{\partial L}{\partial W}$$

$\alpha: learning\ rate$
$L: Loss\ function$

$$\frac{\partial L}{\partial W} = X * \frac{\partial L}{\partial Y}$$

1-3. Structure of CNN

Why do we use convolution?

Ex) 28 x 28 image

- MLP
Input data: 28 x 28 = 784
Output: 100 neurons → 784 x 100 = 78,400 weights + 100 bias

- CNN
Filter: 3 x 3 = 9 weights
# of filter : 100 → 9 x 100 = 900 parameters + 100 bias

# of Parameter is reduced

입력 이미지

필터 1

출력 이미지 1

세로 에지에 반응

필터 2

출력 이미지 2

가로 에지에 반응

2. RNN

Recurrent **N**eural **N**etwork

2-1. Time Series Data

Sentence data -> time series data

“MIMIC is math interchange club.”

“Sungkyunkwan University (SKKU), located in Seoul, South Korea, is a prestigious institution with a history dating back over 600 years. It combines traditional values with modern research, offering a wide range of programs in sciences, humanities, engineering, and business. SKKU is known for its strong academic reputation, innovative research, and global partnerships.”

Dividing sentence into words (Tokenization)
-> Integer Encoding (Index)
-> Embedding Layer
-> RNN input data

What is next word?

2-1. Time Series Data

Word -> vector (Embedding Layer)

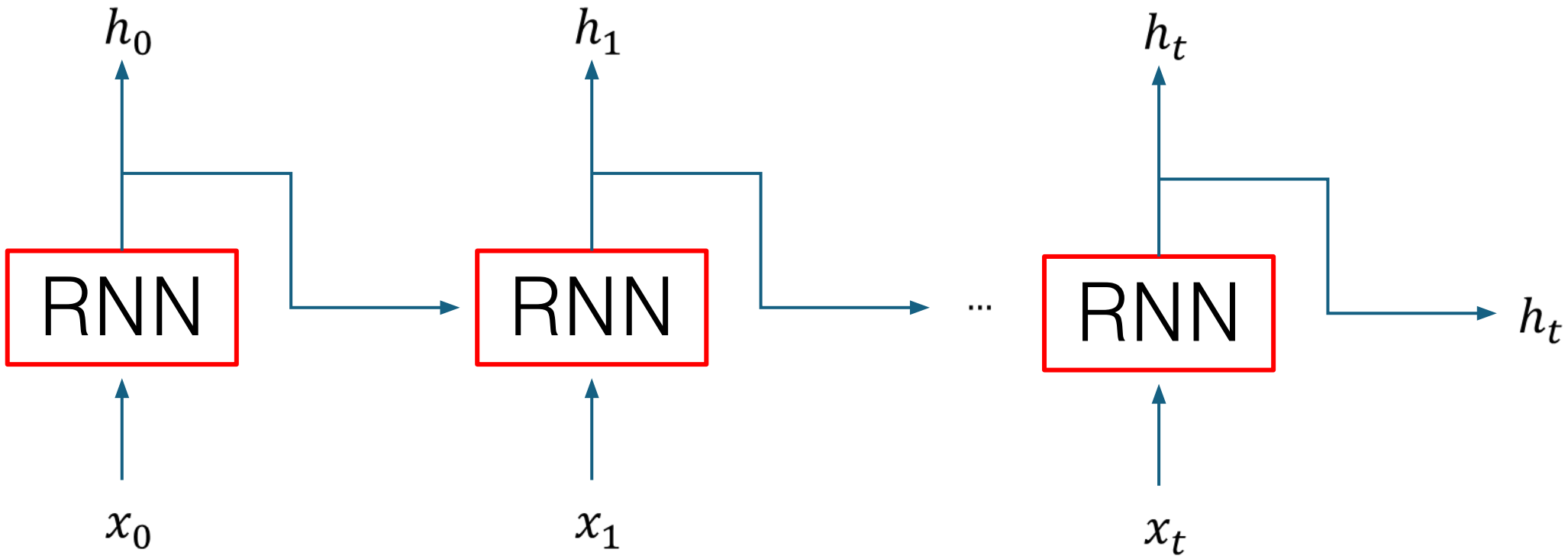
MIMIC / is / math / interchange / club / .

0 1 2 3 4 5 -> indices

Ex) $E = \begin{bmatrix} 0.61 & \cdots & 0.52 \\ \vdots & \ddots & \vdots \\ 0.14 & \cdots & 0.23 \end{bmatrix}$ *where E is* $V \times d$ *matrix*

We will learn matrix E with gradient descent.

2-2. RNN Layer



2-2. RNN Layer

$$h_t = \varphi(h_{t-1}W_h + x_tW_x + b_h) \qquad \varphi : tanh \text{ or } ReLU$$

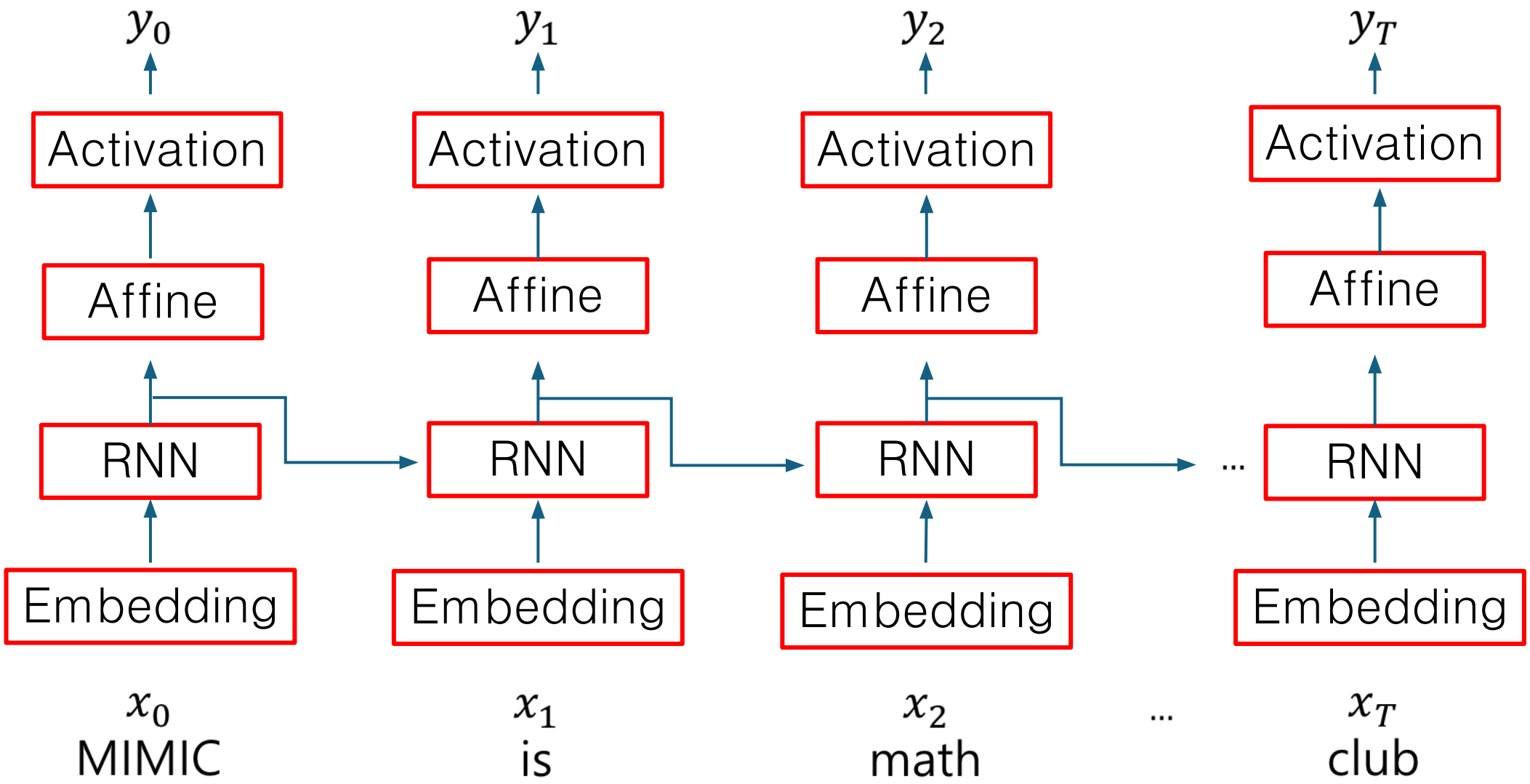
$$h_0 = \varphi(x_0W_x + b_h)$$

$$h_1 = \varphi(h_0W_h + x_1W_x + b_h) \qquad \text{The same wights } W_x, W_h$$

$$\vdots$$

$$h_T = \varphi(h_{T-1}W_h + x_TW_x + b_h)$$

2-3. Structure of RNN



2-3. Structure of RNN

$$y_t = g(W_y h_t + b_y)$$

$g : softmax$ or $sigmoid$

$$y_0 = g(W_y h_0 + b_y)$$
$$y_1 = g(W_y h_1 + b_y)$$

The same wight W_y

$$\vdots$$

$$y_T = g(W_y h_T + b_y)$$

2-3. Structure of RNN

Gradient Descent

$$W_h = W_h - \alpha \frac{\partial L}{\partial W_h}$$

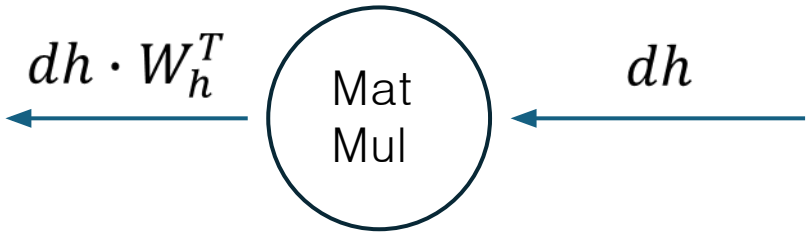
$$W_x = W_x - \alpha \frac{\partial L}{\partial W_x}$$

$$W_y = W_y - \alpha \frac{\partial L}{\partial W_y}$$

$\alpha: learning\ rate$
$L: Loss\ function$

2-3. Structure of RNN

Exploding Gradients & Vanishing Gradients



$$W_h = W_h - \alpha \frac{\partial L}{\partial W_h}$$

LSTM, Gradients clipping

3. Comparison

3. Comparison

	CNN	**RNN**
Data	Spatially structured data	Sequential or time-series data
Information Flow	One-directional	Recurrent
Structure	Repeated block cells [Conv - Act – (Pooling)]	Repeated RNN cell
Main Application	Image classification Object detection Video processing	Natural language processing Speech recognition Time-series forecasting
Parameter	The same filter -> all spatial inputs	The same weights -> all time steps

Q/A

Thank you