머신러닝 개요

Lecture 8 :Convolution Neural Networks

College of Information and Electronic Engineering

Kyung Hee University

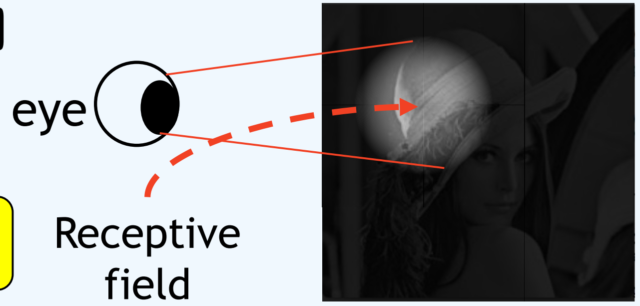
Prof. Wonha Kim

(wonha@khu.ac.kr)

Contents

Convolution for Network (1/8): Receptive field

- Human visual neuron responds to stimuli in a receptive field (= restricted local region).
- Human visual perception responds to local information (not individual pixels)
- Statistically, the pixels are correlated with neighbor pixels.



Extract data information (feature) from a local region (not a pixel).

- **1-D Convolution**

$$s(i) = z \circledast u = \sum_{x=-(h-1)/2}^{(h-1)/2} z(i+x)u(x)$$

0	1	2	3	4	5	6	7
3	2	4	5	1	20	2	2

Kernel

0.3	0.4	0.3

Feature Map

-	2.9	3.7	3.5	7.9	8.9	7.4	-

- **2-D Convolution**

$$s(j,i) = z \circledast u = \sum_{y=-(h-1)/2}^{(h-1)/2} \sum_{x=-(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x)$$

	0	1	2	3	4	5	6	7
0	2	2	2	2	2	1	1	1
1	2	2	2	2	2	1	1	1
2	2	2	2	2	2	1	1	1
3	2	2	2	2	2	1	1	1
4	2	2	2	9	9	9	9	9
5	2	2	2	9	9	9	9	9
6	2	2	2	9	9	9	9	9
7	2	2	2	9	9	9	9	9

Kernel

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

Feature Map

-	-	-	-	-	-	-	-
-	0	0	0	-3	-3	0	-
-	0	0	0	-3	-3	0	-
-	0	7	7	-2	-2	0	-
-	0	14	14	-1	-1	0	-
-	0	21	21	0	0	0	-
-	0	21	21	0	0	0	-
-	-	-	-	-	-	-	-

- 2D- Convolution

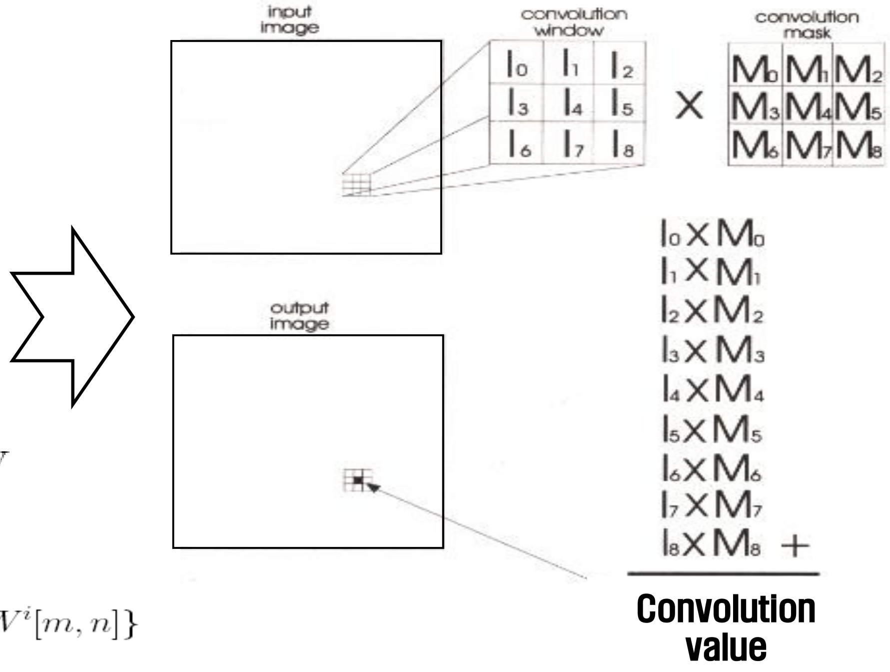
- Pixel value at (i,j) : $I(i,j)$

- Convolution window (Filter): $\mathrm{W} = \begin{bmatrix} W[-M,-N] & \dots & W[-M,N] \\ & \vdots & \\ \cdots & W[0,0] & \cdots \\ & \vdots & \\ W[M,-N] & \dots & W[M,N] \end{bmatrix}$
  (size : (2M+1)x(2N+1))

- Convolution value at (i,j) : $z(i,j) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} I[i-m, j-n] \cdot W[m,n] \equiv I \circledast W$

- Multi-channel (M-channel) convolution : $z(i,j) = \sum_{i=1}^{C} \{ I^i[i-m, j-n] \circledast W^i[m,n] \}$

Repeated for every pixel



2-D input

3	2	1	5
5	1	2	1
1	2	3	0
0	2	1	2

(4x4)

$\circledast$

kernal

1	1	1
1	1	1
1	1	1

(3x3)

=

3	2	1	5
5	1	2	1
1	2	3	0
0	2	1	2

$3 \times 1 + 2 \times 1 + 1 \times 1$
$+ 5 \times 1 + 1 \times 1 + 2 \times 1$
$+ 1 \times 1 + 2 \times 1 + 3 \times 1$
$= 20$

3	2	1	5
5	1	2	1
1	2	3	0
0	2	1	2

= 17

3	2	1	5
5	1	2	1
1	2	3	0
0	2	1	2

$5 \times 1 + 1 \times 1 + 2 \times 1$
$+ 1 \times 1 + 2 \times 1 + 3 \times 1$
$+ 0 \times 1 + 2 \times 1 + 1 \times 1$
$= 17$

3	2	1	5
5	1	2	1
1	2	3	0
0	2	1	2

= 14

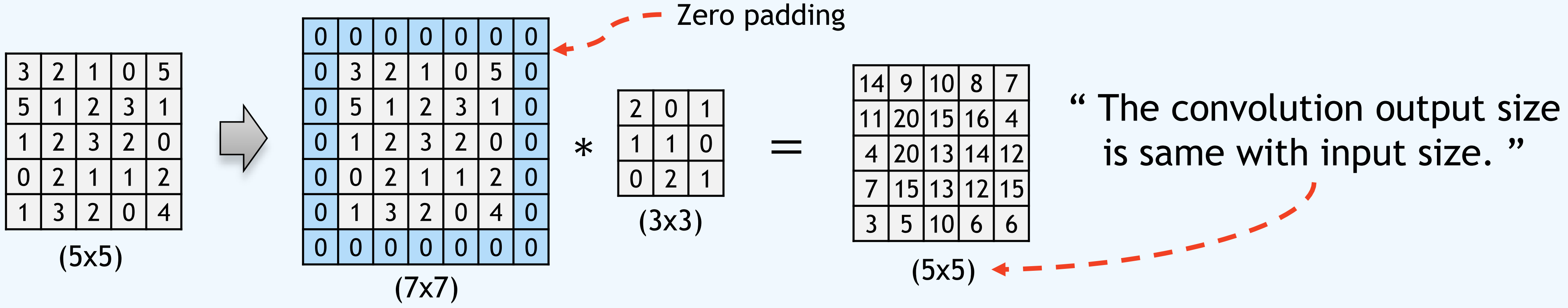
=

20	17
17	14

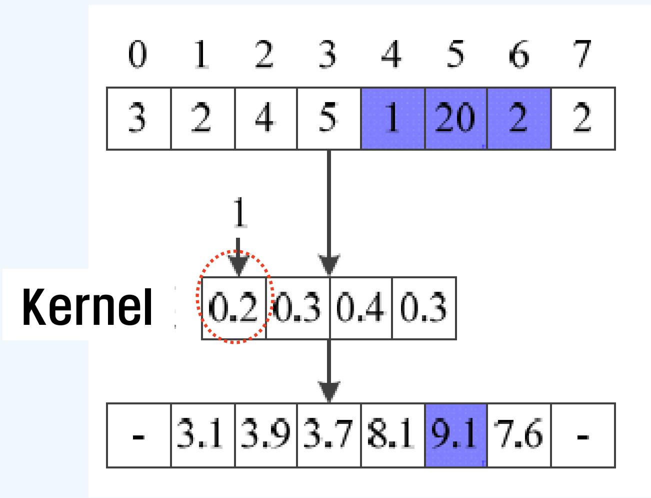
(2x2)

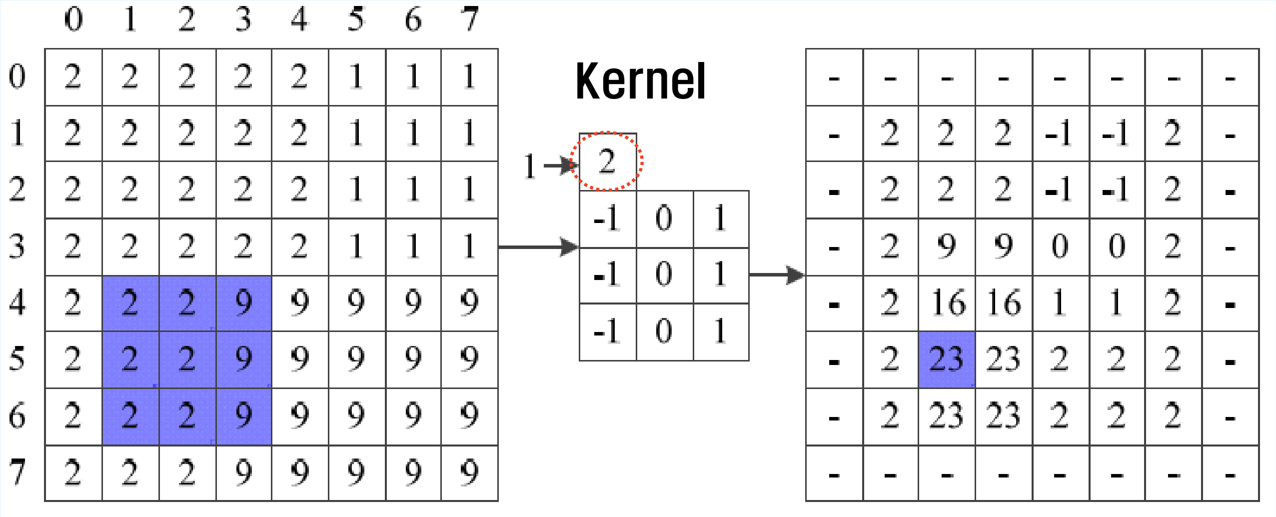
Convolution for Network (3/8) : Zero padding/ Bias

- Padding extends an image with zeros : Compensating the input size reduction at boundaries.



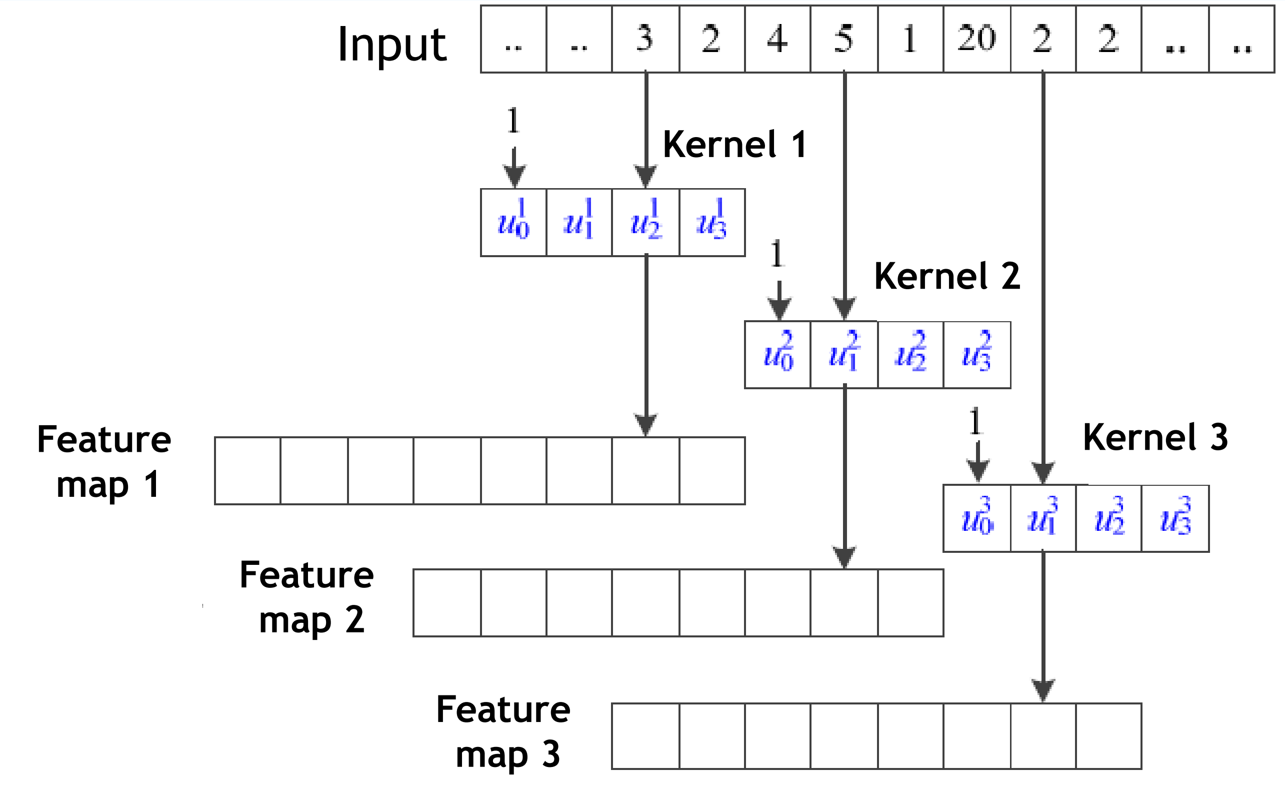
- Bias Addition





Convolution for network (4/8): Multi-kernel structure

- Different kernels extract different features
- In actual applications, 10~100 kernels are used.
- Example : Edge detection
  - Vertical edge extraction : $\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$, Horizontal edge extraction : $\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

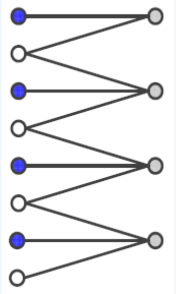


Convolution for Network (5/8) : Stride

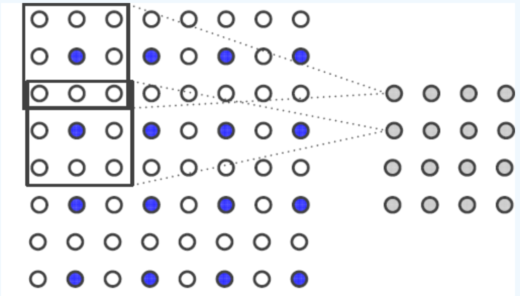
- Stride defines interval of convolution. When stride is k, convolution is performed at every k sample.
  In case of image, feature map is down sized to $1/k^2$.

  - Stride K=2

1-D data



2-D data (ex images)



→ Feature map is down-sampled.

  - Example with Stride K=2

$$\underset{(5\text{x}5)}{\begin{bmatrix} 3 & 2 & 1 & 0 & 5 \\ 5 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 \\ 0 & 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}} * \underset{(3\text{x}3)}{\begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}} = \underset{(2\text{x}2)}{\begin{bmatrix} 20 & 16 \\ 15 & 12 \end{bmatrix}} \Rightarrow$$

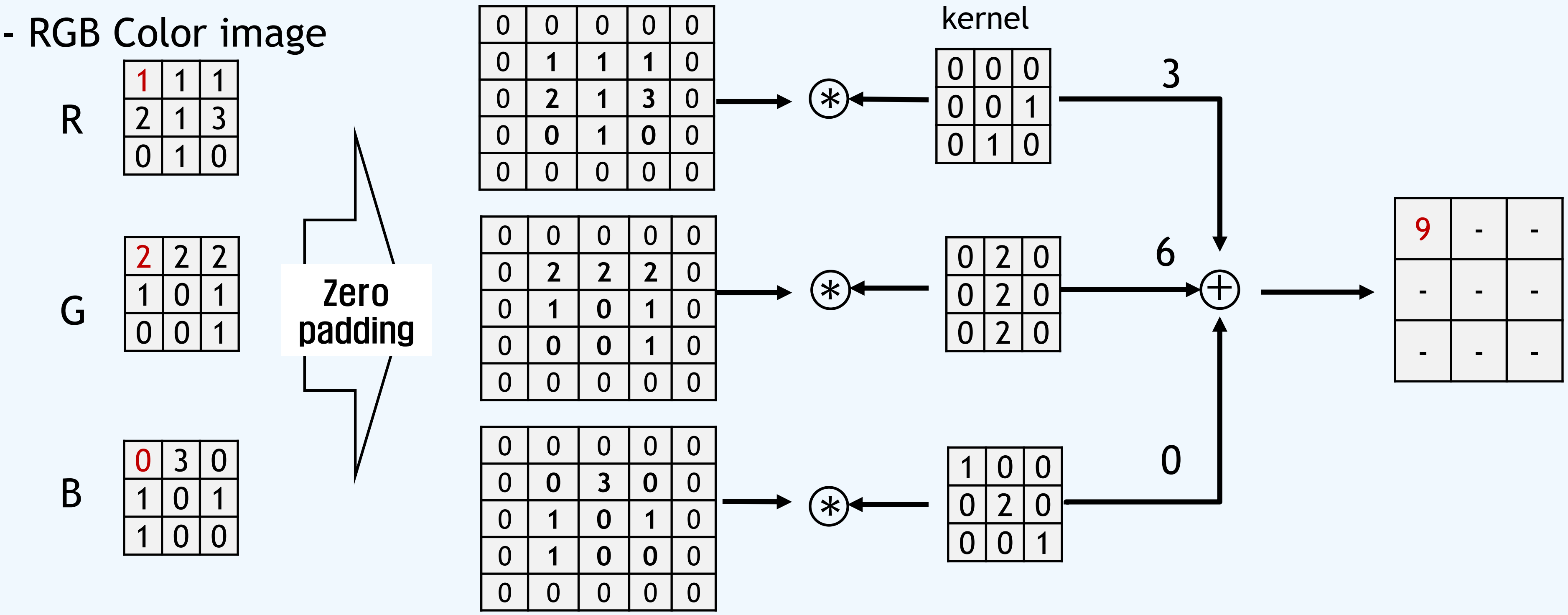
Stride=2

$$\begin{bmatrix} 3 & 2 & 1 & 0 & 5 \\ 5 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 \\ 0 & 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 20 & \\ & \end{bmatrix} \qquad \begin{bmatrix} 3 & 2 & 1 & 0 & 5 \\ 5 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 \\ 0 & 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 16 \\ & \end{bmatrix}$$

Stride=2

$$\begin{bmatrix} 3 & 2 & 1 & 0 & 5 \\ 5 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 \\ 0 & 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 16 \\ 15 & \end{bmatrix} \qquad \begin{bmatrix} 3 & 2 & 1 & 0 & 5 \\ 5 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 \\ 0 & 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 16 \\ 15 & 12 \end{bmatrix}$$

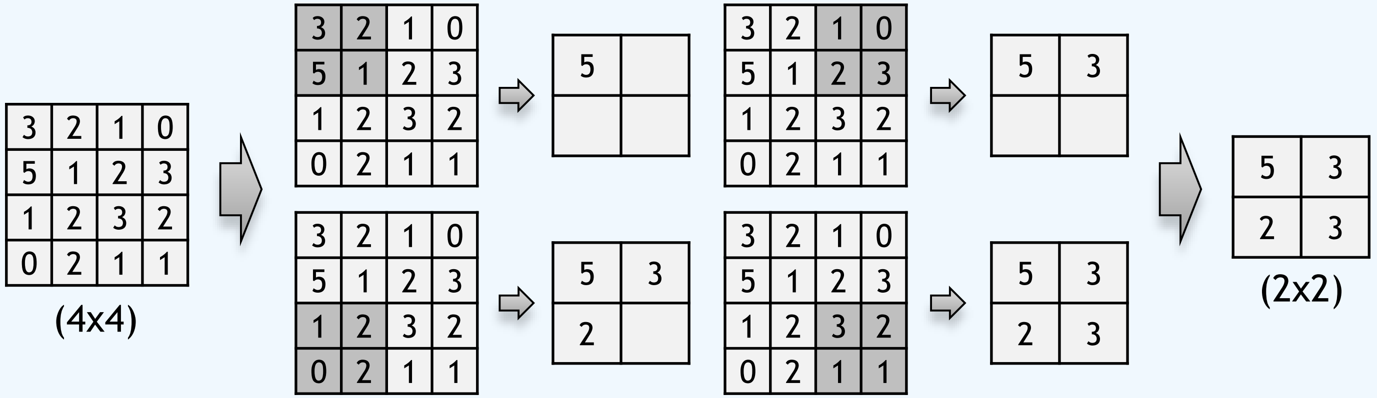
- **Tensor Convolution**



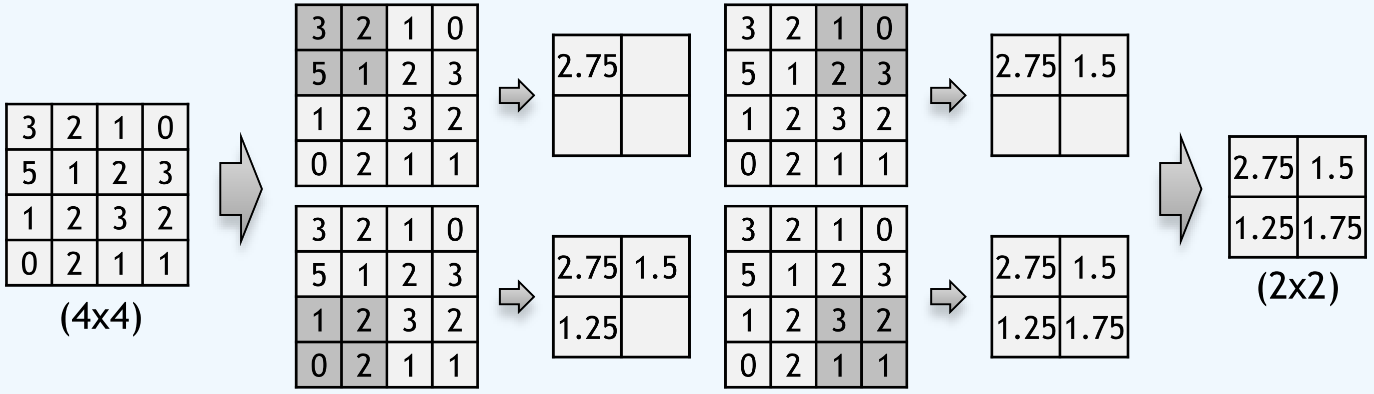
Convolution for Network (7/8) : Pooling

- Pooling : Reducing feature dimension

  - Max pooling (Pooling size : 2x2, Pooling stride : 2)

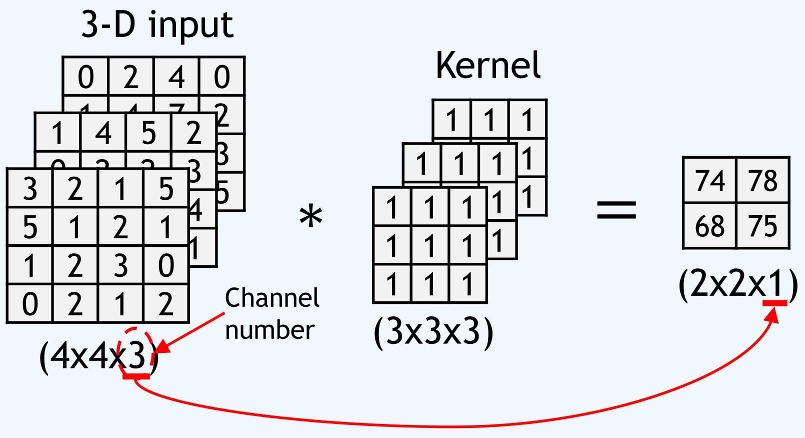


  - Average pooling (Pooling size : 2x2, Pooling stride : 2)

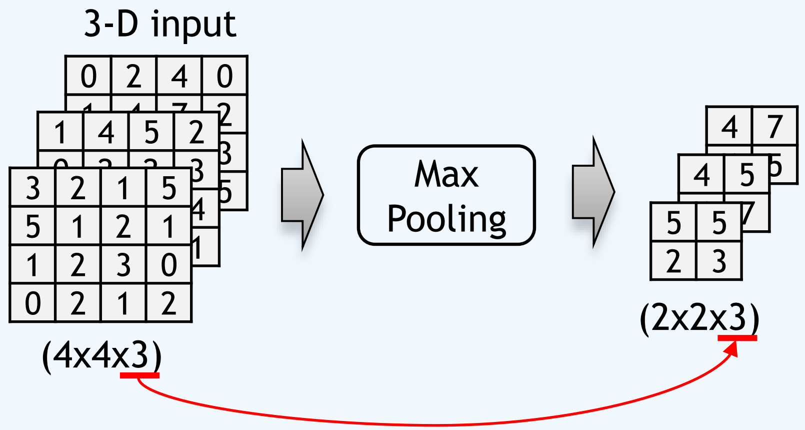


- Pooling & Convolution

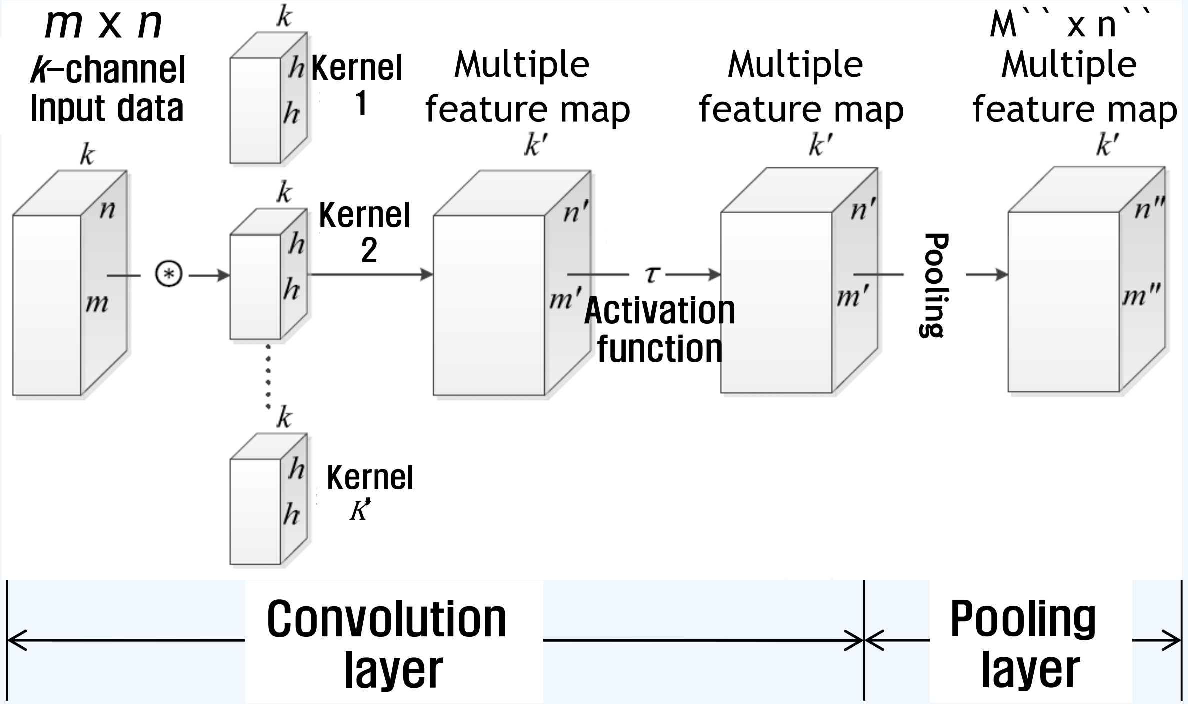
  - Convolution changes channel (feature map) number



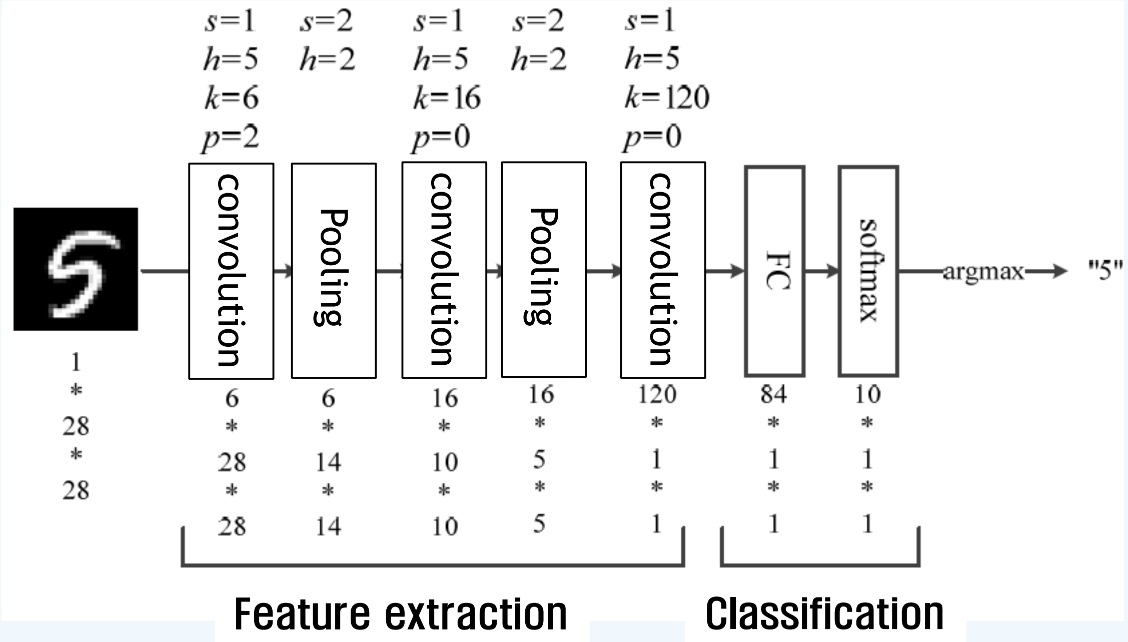
  - Pooling does not changes channel (feature map) number



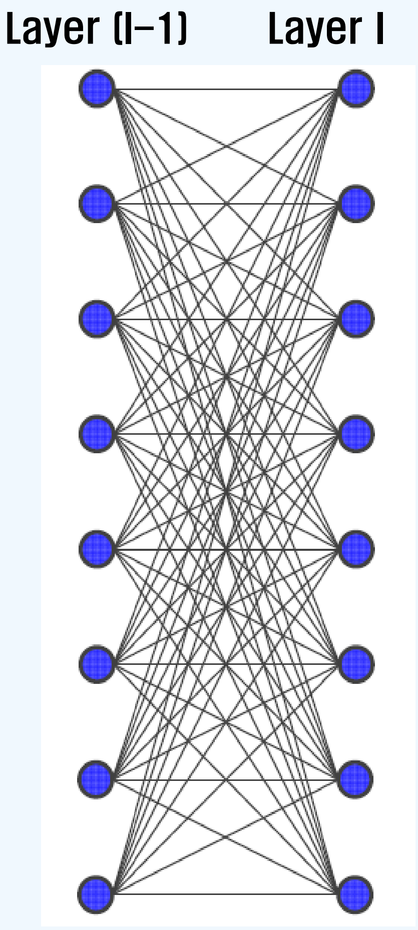
- Typical CNN structure



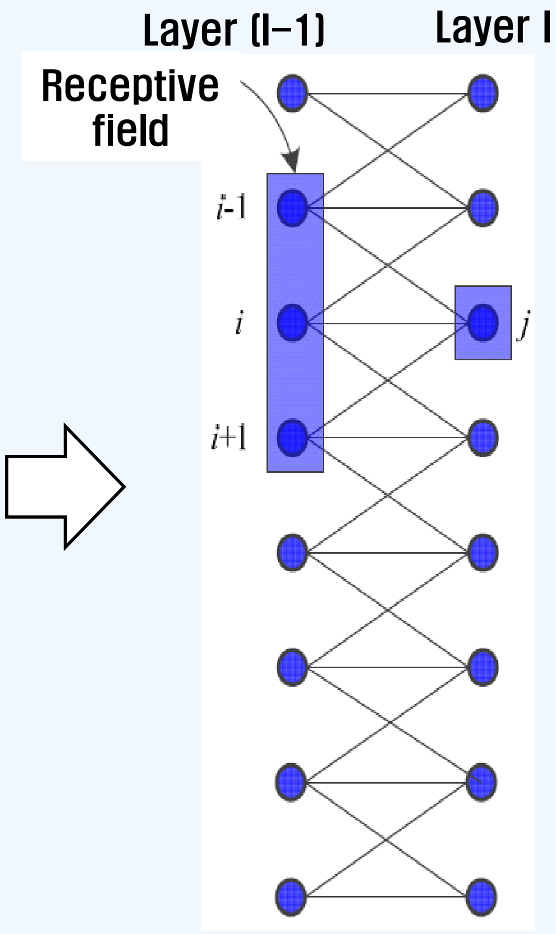
- Example : MNIST recognition



Advantages of Convolution Neural Network (1/2)



Fully connected Network

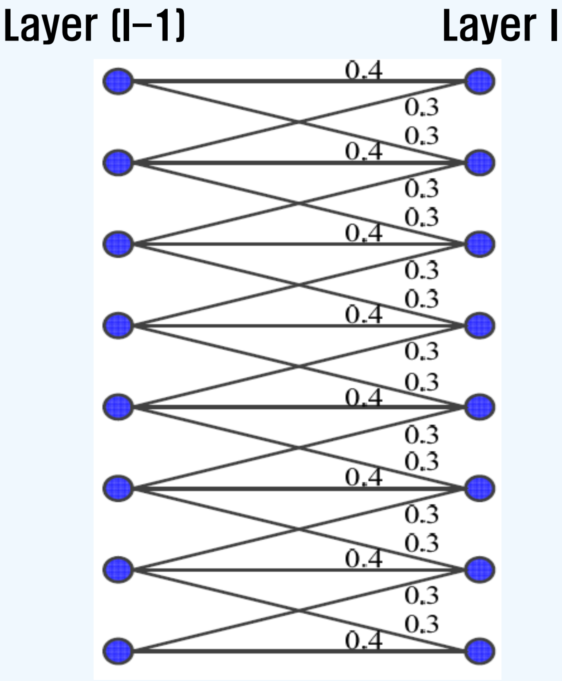


Convolutional Network

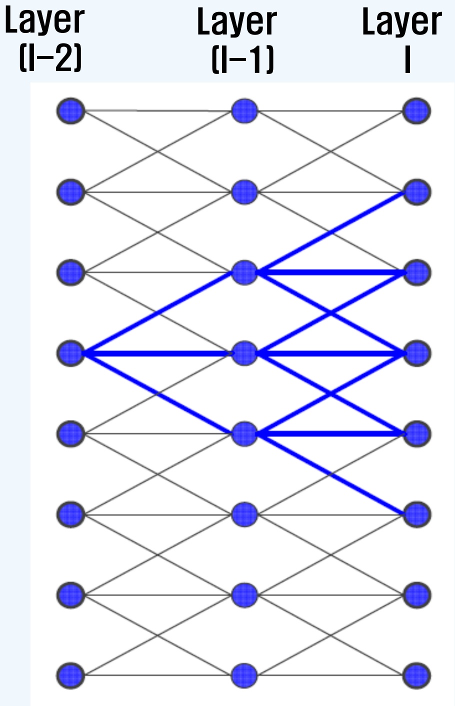
- **Advantages of CNN**
  - Good for lattice (image and speech) data : Translation equivalent.
  - Receptive field well reflects human visual system. So, CNN outperforms the conventional neural network (NN)
  - Variable size input data can be processed.
  - Less memory for parameters than conventional neural network(NN).
  - Less calculation and faster convergence than conventional neural network(NN).

Advantage of Convolution Neural Network (2/2)

- Weight sharing
  - Kernel is weight
  - All nodes use the same kernel.
  - Number of weights is same as kernel size.
  - The complexity of model is greatly reduced.

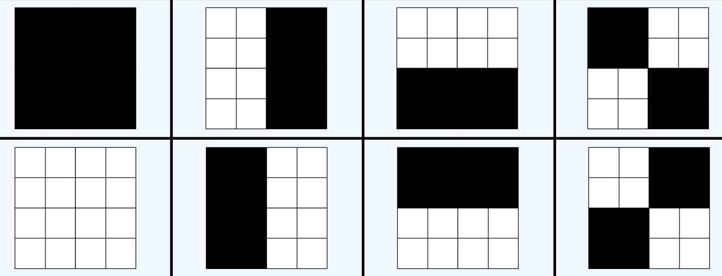


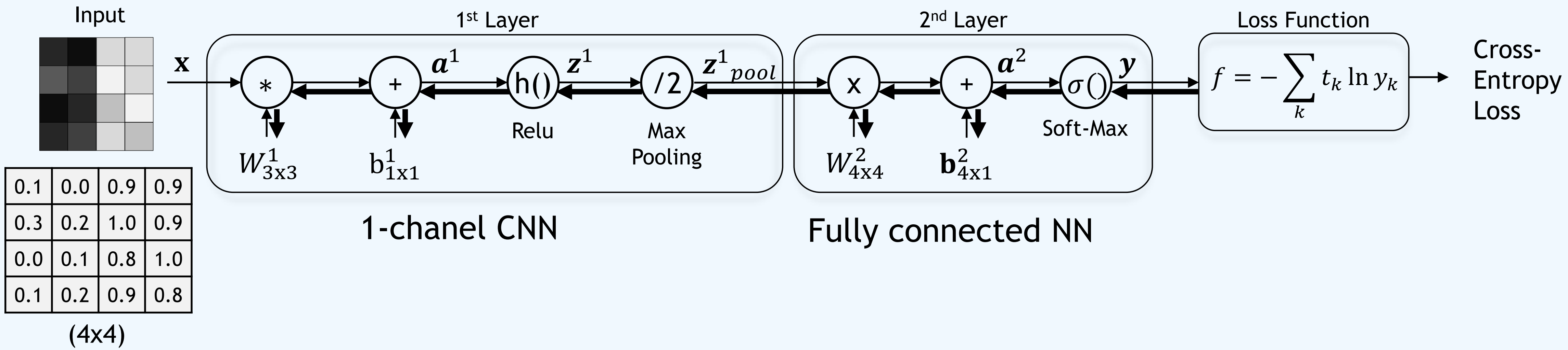
- Parallel and Distributed structure
  - Each node can be calculated independently. So it's a parallel structure.
  - Nodes affect the whole nodes through passing layers. So, it is distributed structure.
  - Good for GPU architectures.



Example -Detecting image direction (1/3)

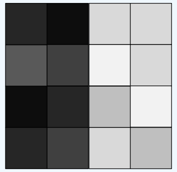
- Detecting image direction (More sophisticated example)
  - 4x4 gray scale image, 4-directions
  - Label 4 directions with binary feature vector
  - Convolution : size = (3x3), stride = 1
  - Pooling : Max pooling, size = (2x2), stride = 2
  - Activation Function : Relu, Soft-max

Direction	Non	vertical	horizontal	diagonal
Image				
Label	[1,0,0,0]	[0,1,0,0]	[0,0,1,0]	[0,0,0,1]



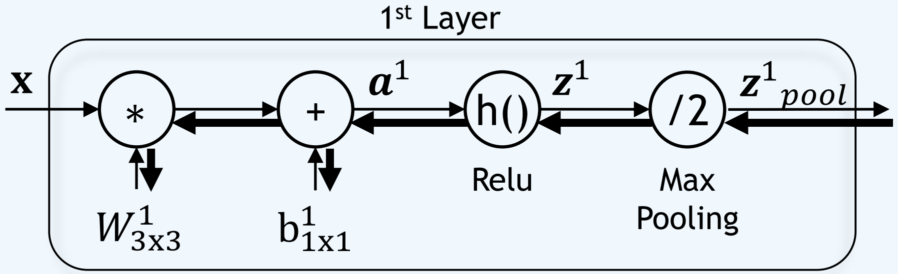
Example : Detecting image direction (2/3)

- 1^{st} Layer (=Convolution Layer)
- Padding : zero-padding,
- Convolution : size = (3x3), stride = 1
- Pooling : max pooling, size = (2x2), stride = 2



0.1	0.0	0.9	0.9
0.3	0.2	1.0	0.9
0.0	0.1	0.8	1.0
0.1	0.2	0.9	0.8

(4x4)



$\mathbf{x}$-padding

0	0	0	0	0	0
0	0.1	0.0	0.9	0.9	0
0	0.3	0.2	1.0	0.9	0
0	0.0	0.1	0.8	1.0	0
0	0.1	0.2	0.9	0.8	0
0	0	0	0	0	0

(6x6)

⇒ $*$ ⇒ $+$ ⇒

W^1_{3x3}

-0.38	-0.80	0.07
-0.34	-0.43	0.98
-0.62	-0.29	-0.94

(3x3)

b^1_{1x1}

0.02

(1x1)

$\boldsymbol{a}^1$

-0.30	-0.32	-0.75	-1.55
-0.09	0.06	-1.49	-2.56
-0.32	-0.41	-1.33	-2.57
0.18	0.76	-0.26	-1.73

(4x4)

⇒ h() Relu ⇒

$\boldsymbol{z}^1$

0	0	0	0
0	0.06	0	0
0	0	0	0
0.18	0.76	0	0

(4x4)

⇒ /2 Max Pooling ⇒

$\boldsymbol{z}^1{}_{pool}$

0.06	0
0.76	0

(2x2)

- Summation of matrix and scalar
- Change scalar to the same size matrix

b^1_{1x1}

0.02

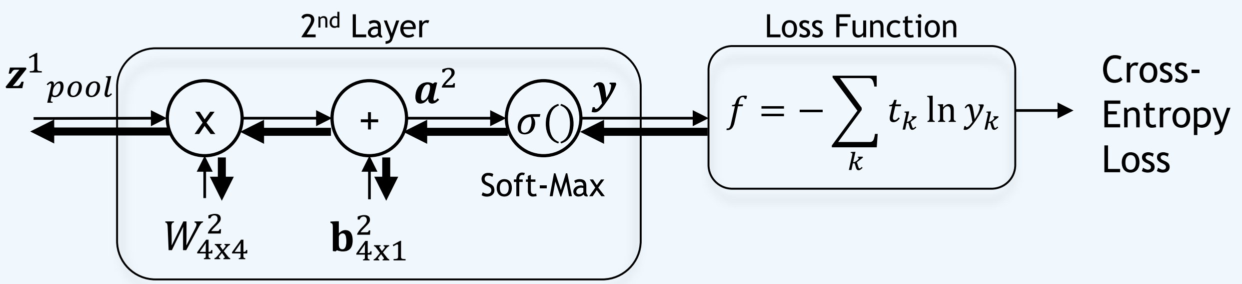
=>

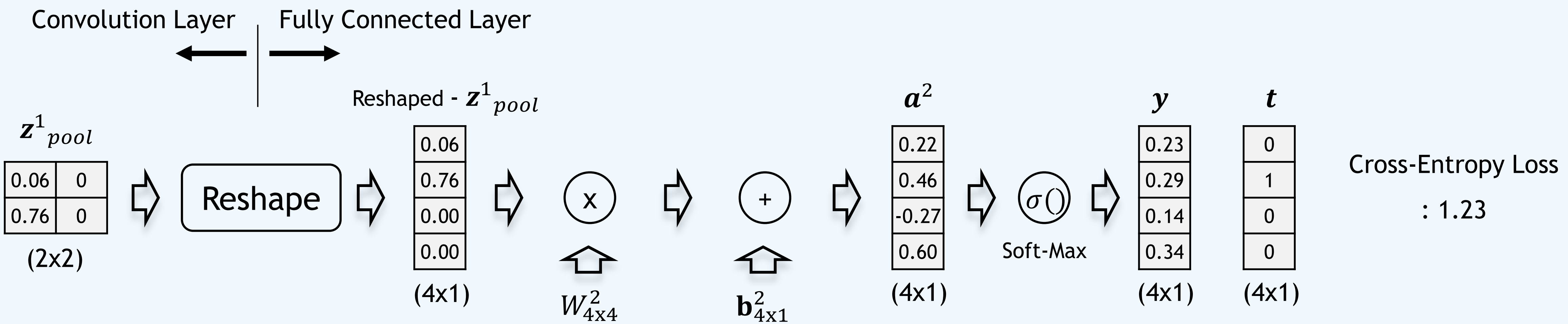
b^1_{4x4}

0.02	0.02	0.02	0.02
0.02	0.02	0.02	0.02
0.02	0.02	0.02	0.02
0.02	0.02	0.02	0.02

Example : Detecting image direction (3/3)

- 2nd Layer (=Fully connected Layer)
  - All nodes are interconnected.
  - So, it is called as "Fully connected layer" .





W^2_{4x4}

-0.78	0.41	0.15	-0.27
0.69	0.58	-0.42	-0.88
0.04	-0.34	-0.47	0.89
0.22	0.83	0.78	0.31

(4x4)

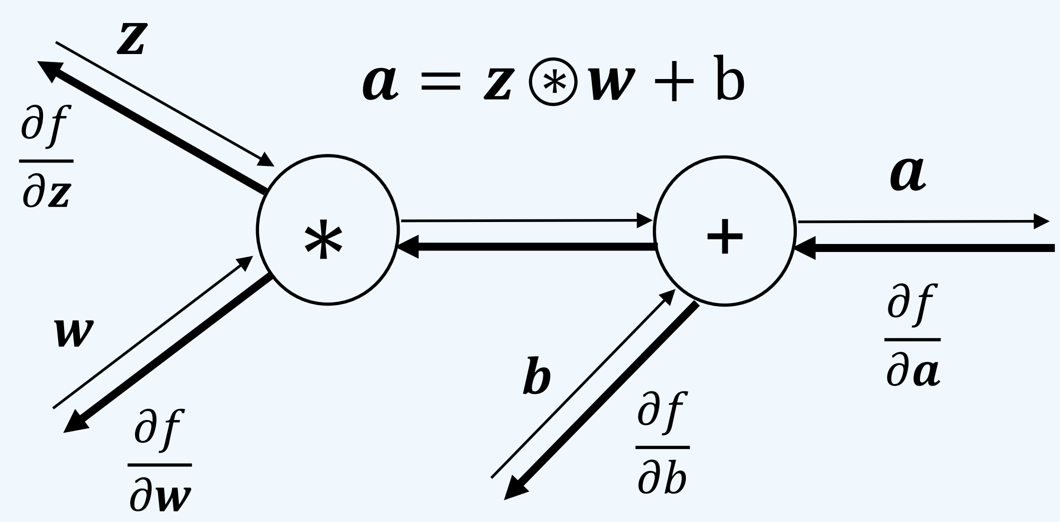
b^2_{4x1}

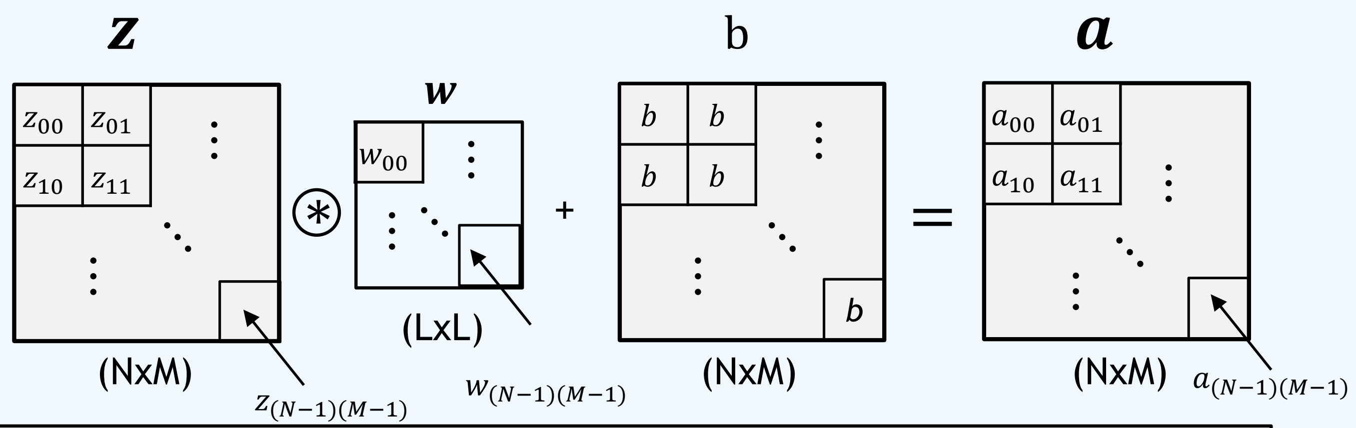
-0.05
-0.02
-0.01
-0.04

(4x1)

" Wrong Prediction "

Backward Propagation of CNN (1/8)





$$\frac{\partial f}{\partial \mathbf{w}} = \frac{\partial f}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial \mathbf{w}} = ? \qquad \frac{\partial f}{\partial \mathbf{z}} = \frac{\partial f}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial \mathbf{z}} = ? \qquad \boldsymbol{b} = \frac{\partial f}{\partial \mathbf{a}}$$

$$\left\{\frac{\partial f}{\partial w_{pq}}\right\} = \left\{\frac{\partial f}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial w_{pq}}\right\} \qquad \left\{\frac{\partial f}{\partial z_{kl}}\right\} = \left\{\frac{\partial f}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial z_{kl}}\right\} \qquad \{b\} = ?$$

$$a_{ij} = \sum_{p=0}^{L-1}\sum_{q=0}^{L-1} w_{pq}\cdot z_{(i+p-L/2)(j+q-L/2)} = w_{00}\cdot z_{(i-L/2)(j-L/2)} + w_{01}\cdot z_{(i-L/2)(j-L/2+1)}$$
$$\cdots + w_{(\frac{L}{2}\frac{L}{2})}\cdot z_{ij} + \cdots + w_{(L-1)(L-1)}\cdot z_{(i+L/2-1)(j+L/2-1)}$$

zero padding :

$$z_{kl} = 0 \text{ for } -\tfrac{L}{2} < k < 0, (N-1) < k < N+\tfrac{L}{2}, -\tfrac{L}{2} < l < 0, (M-1) < l < M+\tfrac{L}{2}$$

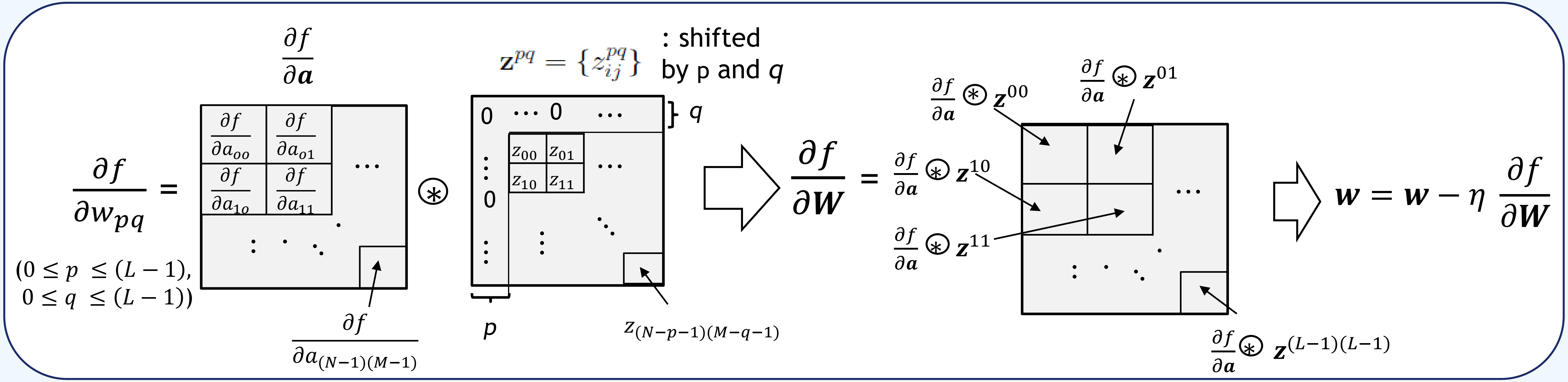
$$\frac{\partial a_{ij}}{\partial w_{pq}} = \begin{cases} z_{(i+p-L/2)(j+p-L/2)} & \text{for } 0 \le p,\ q \le (L-1) \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial a_{ij}}{\partial z_{kl}} = \begin{cases} w_{(k-i-L/2)(l-j-L/2)} & \text{for } i-\frac{L}{2} \le k \le (i+\frac{L}{2}-1),\ j-\frac{L}{2} \le l \le (j+\frac{L}{2}-1) \\ 0 & \text{otherwise} \end{cases}$$

CNN Backward Propagation (2/8): $\frac{\partial f}{\partial w}$

$$\frac{\partial f}{\partial w_{pq}} = \frac{\partial f}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{a}}{\partial w_{pq}} = \frac{\partial f}{\partial a_{00}} \cdot \frac{\partial a_{00}}{\partial w_{pq}} \cdots + \frac{\partial f}{\partial a_{ij}} \cdot \frac{\partial a_{ij}}{\partial w_{pq}} \cdots + \frac{\partial f}{\partial a_{(N-1)(M-1)}} \cdot \frac{\partial a_{(N-1)(M-1)}}{\partial w_{pq}} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \frac{\partial f}{\partial a_{ij}} \cdot \frac{\partial a_{ij}}{\partial w_{pq}}$$

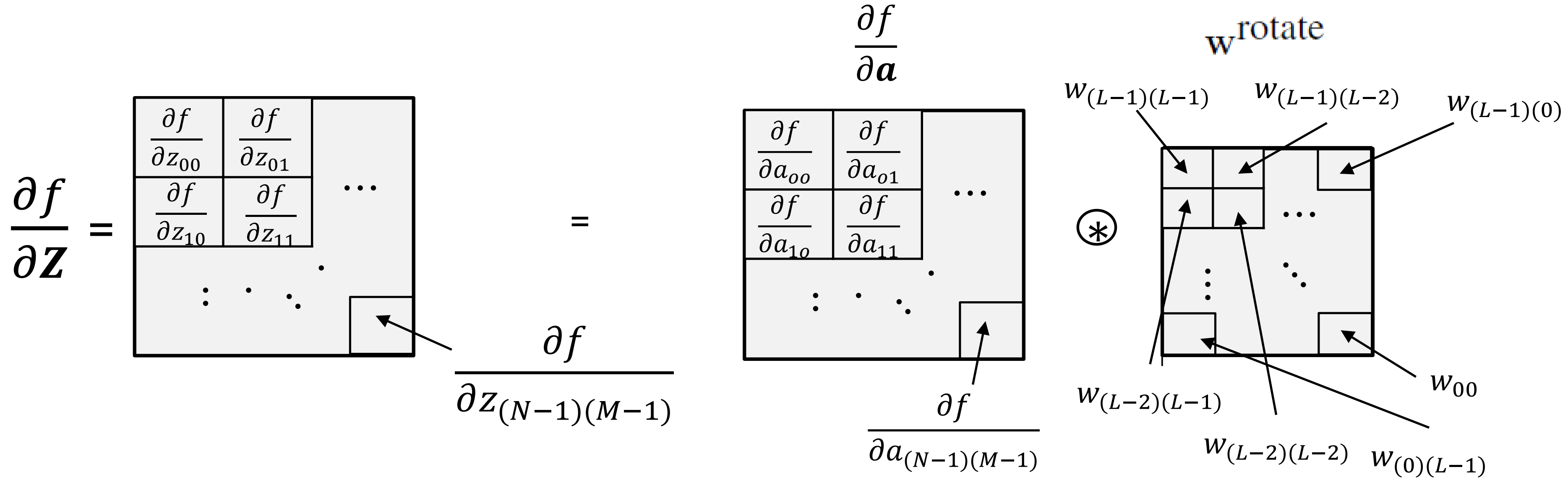
$$= \begin{cases} \displaystyle\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \frac{\partial f}{\partial a_{ij}} \cdot z_{(i+p-\frac{L}{2})(j+p-\frac{L}{2})} & \text{for } 0 \le p,\ q \le (L-1) \\ 0 & \text{otherwise} \end{cases} = \frac{\partial f}{\partial \mathbf{a}} \circledast \mathbf{z}^{pq}$$



Backward Propagation of CNN (3/8) : $\frac{\partial f}{\partial z}$

$$\frac{\partial f}{\partial z_{kl}} = \frac{\partial f}{\partial \mathrm{a}}\cdot\frac{\partial \mathrm{a}}{\partial z_{kl}} = \frac{\partial f}{\partial a_{00}}\cdot\frac{\partial a_{00}}{\partial z_{kl}}\cdots+\frac{\partial f}{\partial a_{ij}}\cdot\frac{\partial a_{ij}}{\partial z_{kl}}\cdots+\frac{\partial f}{\partial a_{(N-1)(M-1)}}\cdot\frac{\partial a_{(N-1)(M-1)}}{\partial z_{kl}} = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\frac{\partial f}{\partial a_{ij}}\cdot\frac{\partial a_{ij}}{\partial z_{kl}}$$

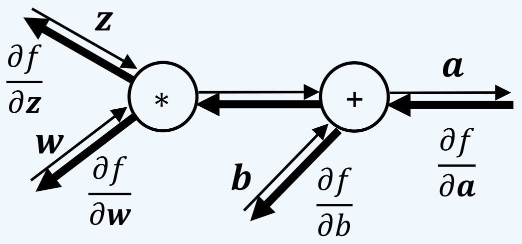
$$= \begin{cases} \displaystyle\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\frac{\partial f}{\partial a_{ij}}\cdot w_{(k-i+\frac{L}{2})(l-j+\frac{L}{2})} & \text{for } i-\frac{L}{2}\le k\le (i+\frac{L}{2}-1),\quad j-\frac{L}{2}\le l\le (j+\frac{L}{2}-1) \\ 0 & \text{otherwise} \end{cases} = \frac{\partial f}{\partial \mathrm{a}} \circledast \mathrm{w}^{\mathrm{rotate}}$$



Backward Propagation of CNN (4/8)

- **Backward Propagation of Convolution Layer**

- Convolution Layer with bias term



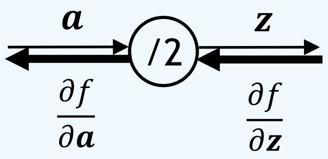
- In two-dimension,

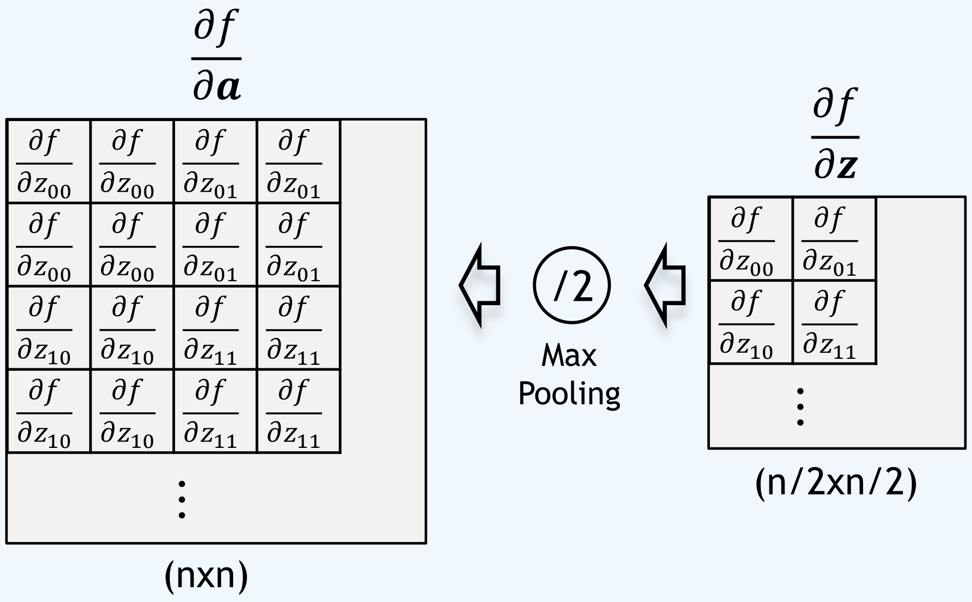
$$\underbrace{\begin{bmatrix} x_{00} & x_{01} & x_{02} & \\ x_{10} & x_{11} & x_{12} & \\ x_{20} & x_{21} & x_{22} & \\ & \vdots & & \end{bmatrix}}_{(n\mathrm{x}n)}^{\boldsymbol{x}} * \underbrace{\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}}_{(3\mathrm{x}3)}^{\boldsymbol{w}} + \underbrace{\begin{bmatrix} b & b & b & \\ b & b & b & \\ b & b & b & \\ & \vdots & & \end{bmatrix}}_{(n\text{-}2\mathrm{x}n\text{-}2)}^{\boldsymbol{b}} = \underbrace{\begin{bmatrix} a_{00} & a_{01} & a_{02} & \\ a_{10} & a_{11} & a_{12} & \\ a_{20} & a_{21} & a_{22} & \\ & \vdots & & \end{bmatrix}}_{(n\text{-}2\mathrm{x}n\text{-}2)}^{\boldsymbol{a}}$$

$$\frac{\partial f}{\partial b} = \sum_{i,j} \frac{\partial f}{\partial a_{i,j}} \qquad b = b - \eta \frac{\partial f}{\partial b}$$

- **Backward Propagation of Pooling Layer**

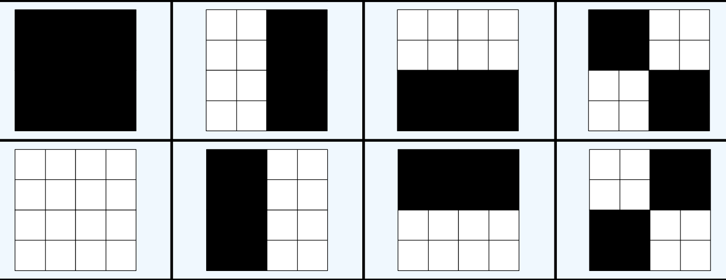
- Pooling(size : 2x2, stride : 2)

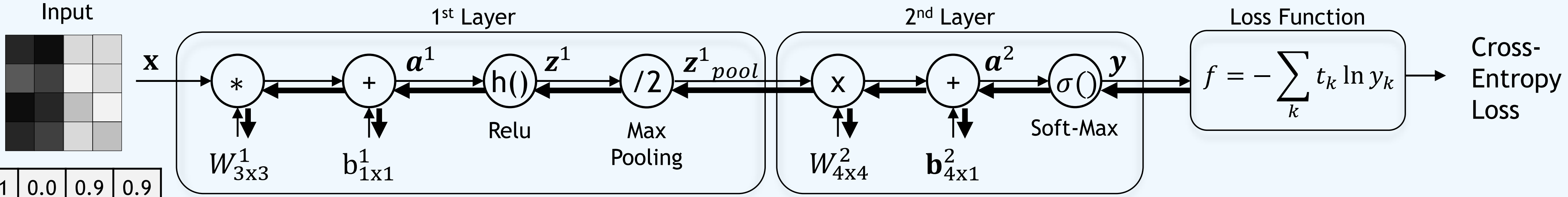




Backward Propagation of CNN (5/8) : Example

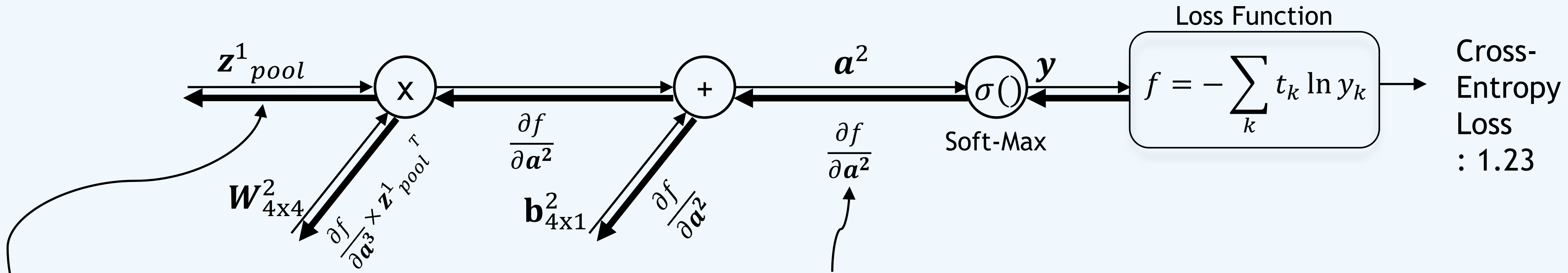
- Detecting image direction (More sophisticated example)
  - 4x4 gray scale image, 4-directions
  - Label 4 directions with binary feature vector
  - Convolution : size = (3x3), stride = 1
  - Pooling : Max pooling, size = (2x2), stride = 2
  - Activation Function : Relu, Soft-max
  - Learning Rate : $\eta = 0.5$

Direction	Non	vertical	horizontal	diagonal
Image				
Label	[1,0,0,0]	[0,1,0,0]	[0,0,1,0]	[0,0,0,1]



0.1	0.0	0.9	0.9
0.3	0.2	1.0	0.9
0.0	0.1	0.8	1.0
0.1	0.2	0.9	0.8

- **2nd Layer (Fully Connected Layer)**



$$\frac{\partial f}{\partial \boldsymbol{z}^1{}_{pool}} = {W^2_{4x4}}^T \times \frac{\partial f}{\partial \boldsymbol{a}^2} = \begin{bmatrix} -0.59 \\ -0.09 \\ 0.53 \\ 0.79 \end{bmatrix}$$

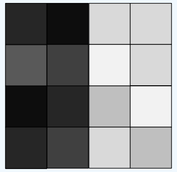
$$\frac{\partial f}{\partial \boldsymbol{a}^2} = \mathbf{y} - \mathbf{t} = \begin{bmatrix} 0.23 \\ -0.71 \\ 0.14 \\ 0.34 \end{bmatrix}$$

$$\underset{\boldsymbol{W}^2_{4x4}}{\begin{bmatrix} -0.79 & 0.32 & 0.15 & -0.27 \\ 0.71 & 0.85 & -0.42 & -0.88 \\ 0.04 & -0.39 & -0.47 & 0.89 \\ 0.21 & 0.70 & 0.78 & 0.31 \end{bmatrix}} = \underset{\boldsymbol{W}^2_{4x4}}{\begin{bmatrix} -0.78 & 0.41 & 0.15 & -0.27 \\ 0.69 & 0.58 & -0.42 & -0.88 \\ 0.04 & -0.34 & -0.47 & 0.89 \\ 0.22 & 0.83 & 0.78 & 0.31 \end{bmatrix}} - \eta \times \underset{\mathbf{y}-\mathbf{t}}{\begin{bmatrix} 0.23 \\ -0.71 \\ 0.14 \\ 0.34 \end{bmatrix}} \times \underset{{\boldsymbol{z}^1{}_{pool}}^T}{\begin{bmatrix} 0.06 & 0.76 & 0.00 & 0.00 \end{bmatrix}}$$

$$\underset{\mathbf{b}^2_{4x1}}{\begin{bmatrix} -0.16 \\ 0.33 \\ -0.08 \\ -0.21 \end{bmatrix}} = \underset{\mathbf{b}^2_{4x1}}{\begin{bmatrix} -0.05 \\ -0.02 \\ -0.01 \\ -0.04 \end{bmatrix}} - \eta \times \underset{\mathbf{y}-\mathbf{t}}{\begin{bmatrix} 0.23 \\ -0.71 \\ 0.14 \\ 0.34 \end{bmatrix}}$$

Backward Propagation of CNN (7/8) : Example

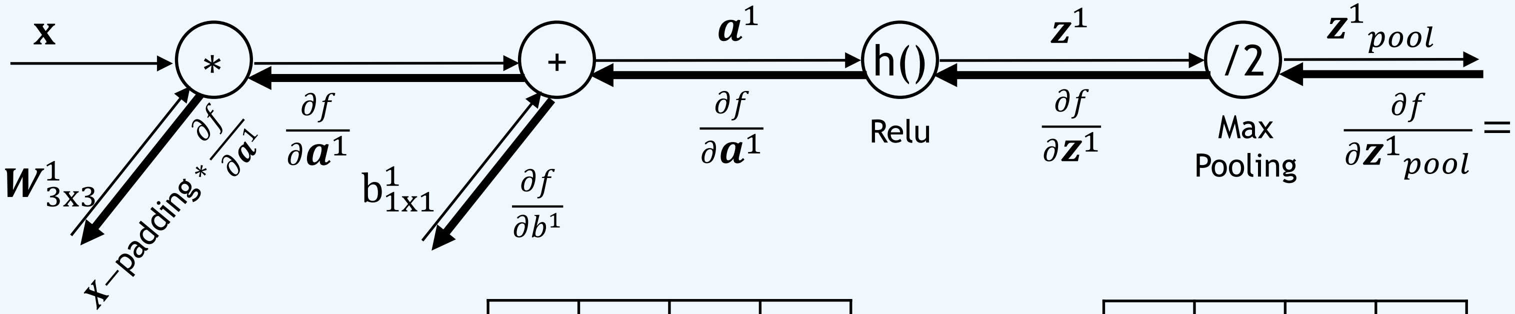
- 1st Layer (Convolution Layer)



0.1	0.0	0.9	0.9
0.3	0.2	1.0	0.9
0.0	0.1	0.8	1.0
0.1	0.2	0.9	0.8

-0.30	-0.32	-0.75	-1.55
-0.09	0.06	-1.49	-2.56
-0.32	-0.41	-1.33	-2.57
0.18	0.76	-0.26	-1.76

: Larger than 0



$$\frac{\partial f}{\partial \boldsymbol{z}^1{}_{pool}} = \begin{bmatrix} -0.59 \\ -0.09 \\ 0.53 \\ 0.79 \end{bmatrix} = \begin{bmatrix} -0.59 & 0.53 \\ -0.09 & 0.79 \end{bmatrix}$$

$$\frac{\partial f}{\partial \boldsymbol{a}^1} = \frac{\partial f}{\partial \boldsymbol{z}^1} \circ (\boldsymbol{a}^1 > 0) =$$

0.00	0.00	0.00	0.00
0.00	-0.59	0.00	0.00
0.00	0.00	0.00	0.00
-0.09	-0.09	0.00	0.00

$$\frac{\partial f}{\partial \boldsymbol{z}^1} =$$

-0.59	-0.59	0.53	0.53
-0.59	-0.59	0.53	0.53
-0.09	-0.09	0.79	0.79
-0.09	-0.09	0.79	0.79

⇐ x2 ⇐

-0.59	0.53
-0.09	0.79

$$\boldsymbol{W}^1_{3x3} = \boldsymbol{W}^1_{3x3} - \eta \times \frac{\partial f}{\partial \boldsymbol{a}^1} \circledast \mathbf{x}\text{-shifting}$$

$\boldsymbol{W}^1_{3x3}$

-0.35	-0.80	0.37
-0.25	-0.36	1.32
-0.62	-0.26	-0.70

=

$\boldsymbol{W}^1_{3x3}$

-0.38	-0.80	0.07
-0.34	-0.43	0.98
-0.62	-0.29	-0.94

$- \eta \times$ $\frac{\partial f}{\partial \boldsymbol{a}^1}$

0.00	0.00	0.00	0.00
0.00	-0.59	0.00	0.00
0.00	0.00	0.00	0.00
-0.09	-0.09	0.00	0.00

$\circledast$ **x**-shifting

0.1	0.0	0.9	0.9
0.3	0.2	1.0	0.9
0.0	0.1	0.8	1.0
0.1	0.2	0.9	0.8

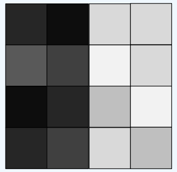
$$\underset{\boxed{0.40}}{\mathrm{b}^1_{1x1}} = \underset{\boxed{0.02}}{\mathrm{b}^1_{1x1}} - \eta \times \sum \frac{\partial f}{\partial \boldsymbol{a}^1}$$

$$0.02 - 0.5 \times (-0.59 - 0.09 - 0.09)$$

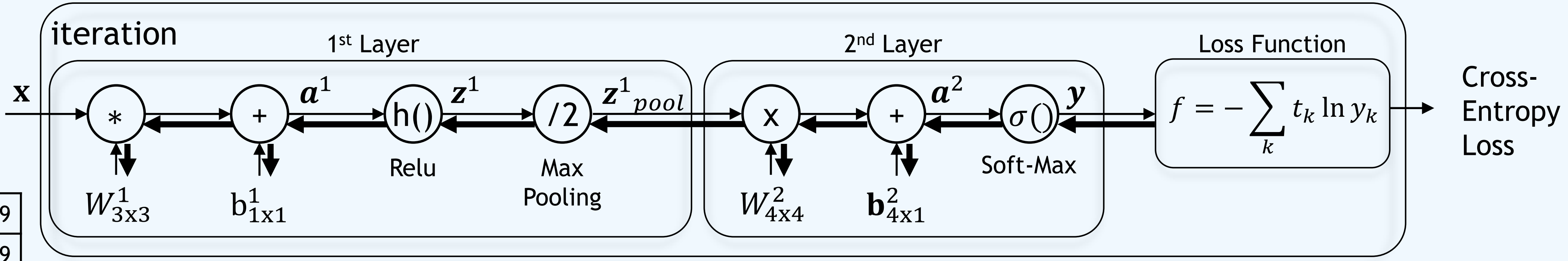
Backward Propagation of CNN (8/8) : Example

- **Training : 7 iteration**

Input



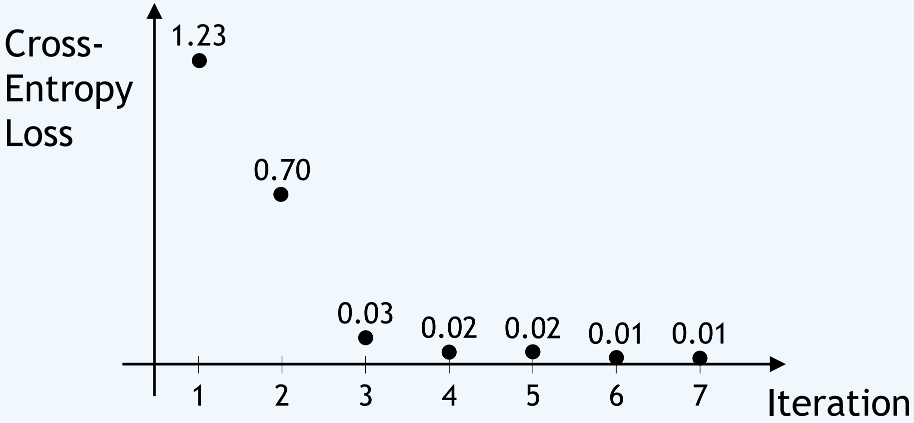
0.1	0.0	0.9	0.9
0.3	0.2	1.0	0.9
0.0	0.1	0.8	1.0
0.1	0.2	0.9	0.8



- **Prediction($\mathbf{y}$)**

Iteration	1	2	3	4	5	6	7	t
Predic-tion	0.23	0.03	0.00	0.00	0.00	0.00	0.00	0
	0.29	0.50	0.97	0.98	0.98	0.99	0.99	1
	0.14	0.07	0.01	0.00	0.00	0.00	0.00	0
	0.34	0.40	0.02	0.02	0.01	0.01	0.01	0

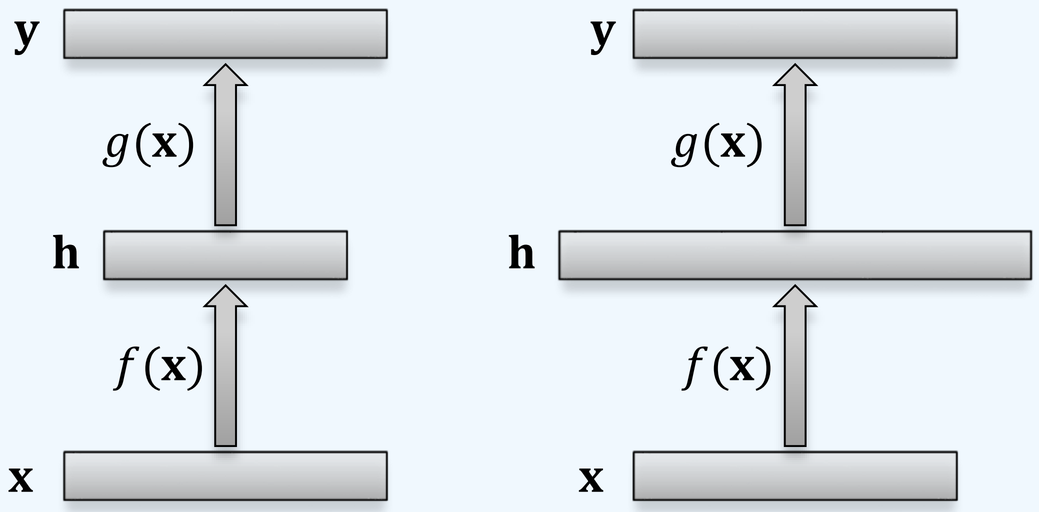
- **Loss Decreasing**



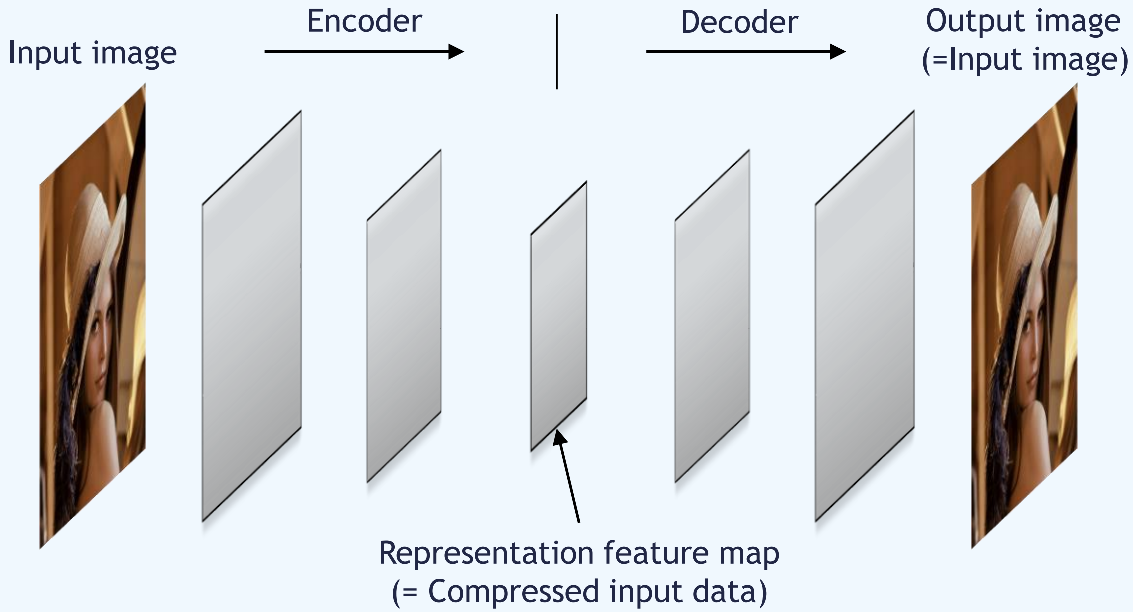
Applications of CNN (1/3) : Auto-encoder/decoder

- The aim is to learn a representation for a set of data.
- In auto-encoder, $\mathbf{y} = g(f(\mathbf{x}))$ where f : encoder, g : decoder , $\mathbf{y} = \mathbf{x}$ *or* $\mathbf{x}'$.
- In under-complete, the representation feature map is smaller than input map.
- In over-complete, the representation feature map is larger than input map.

- Under-complete
- Over-complete

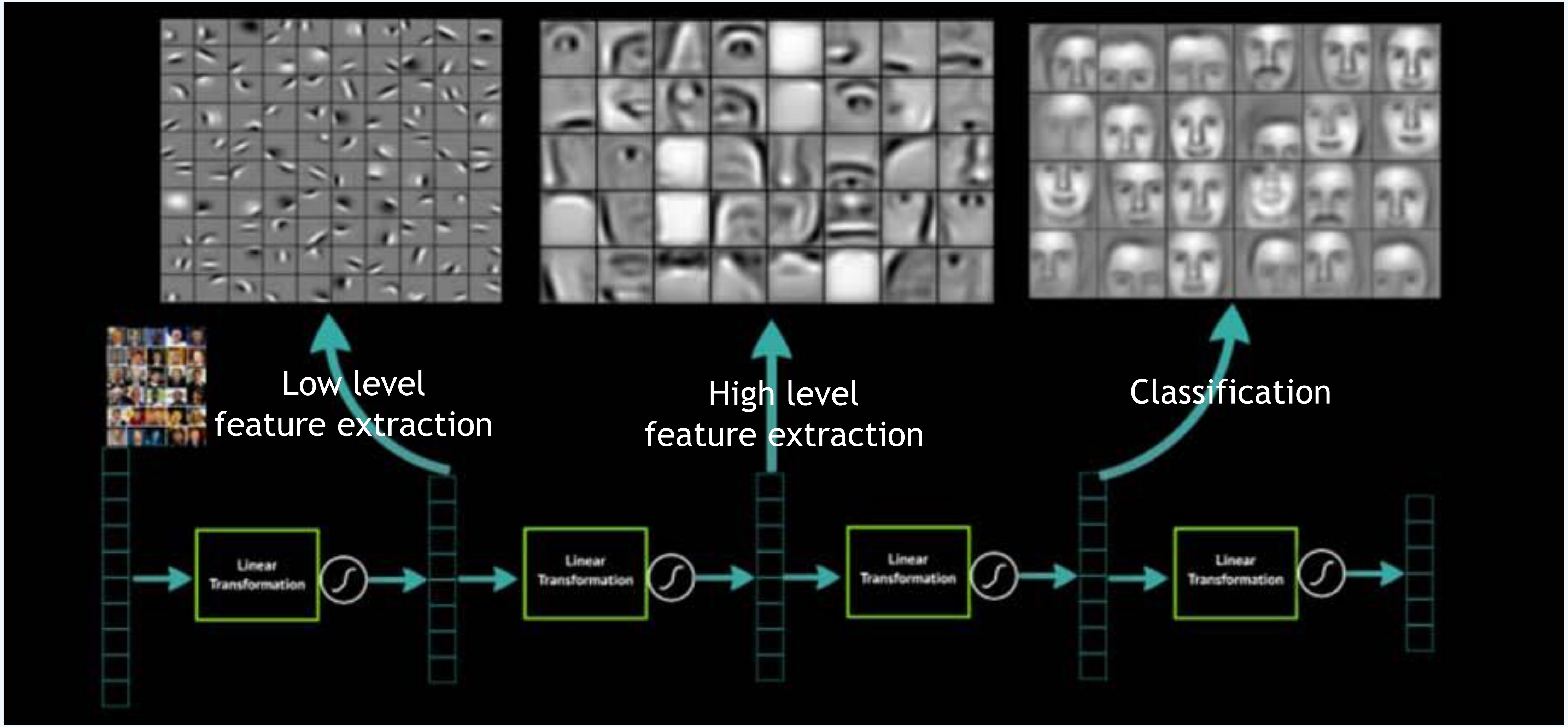


Ex) Image compression(under-complete)



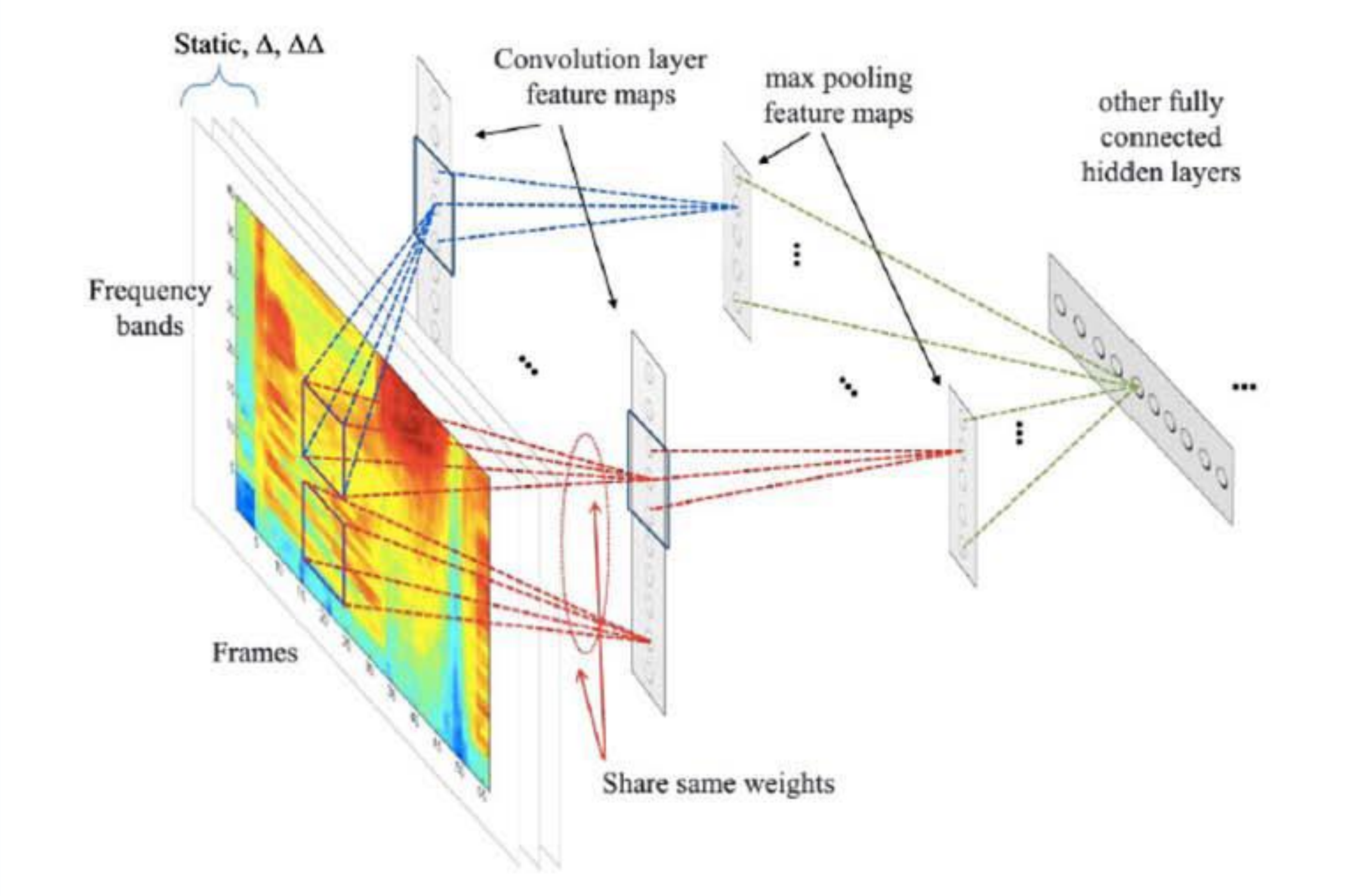
Applications of CNN (2/3) : Face recognition

- Deep Learning learns layers of features



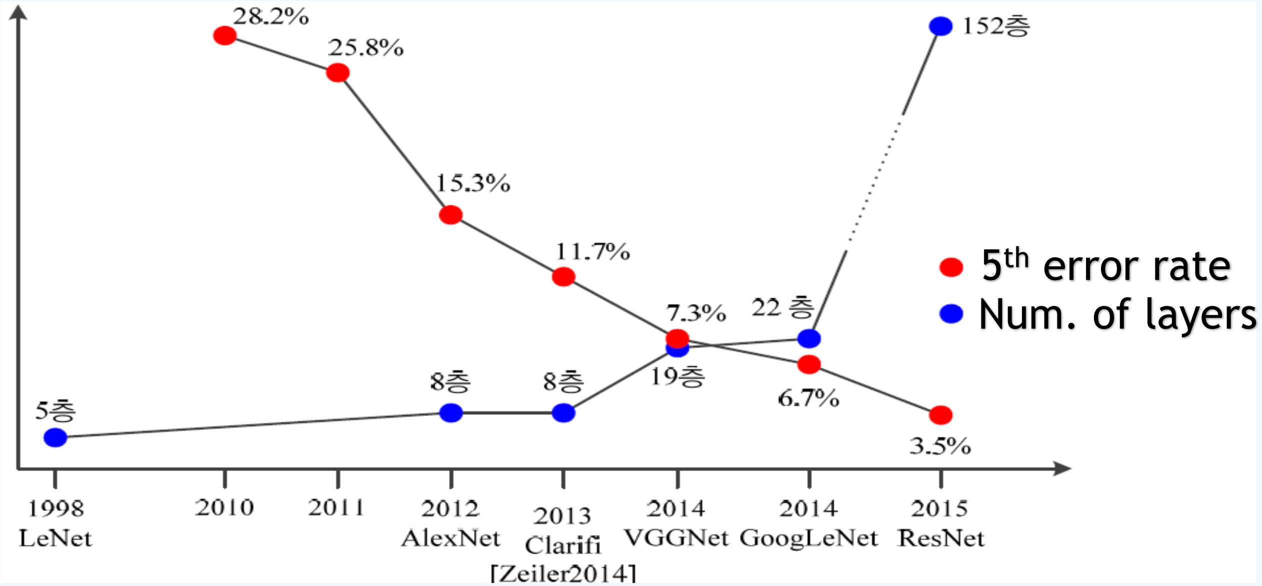
Applications of CNN (3/3) : Speech Recognition

- **Speech Recognition**



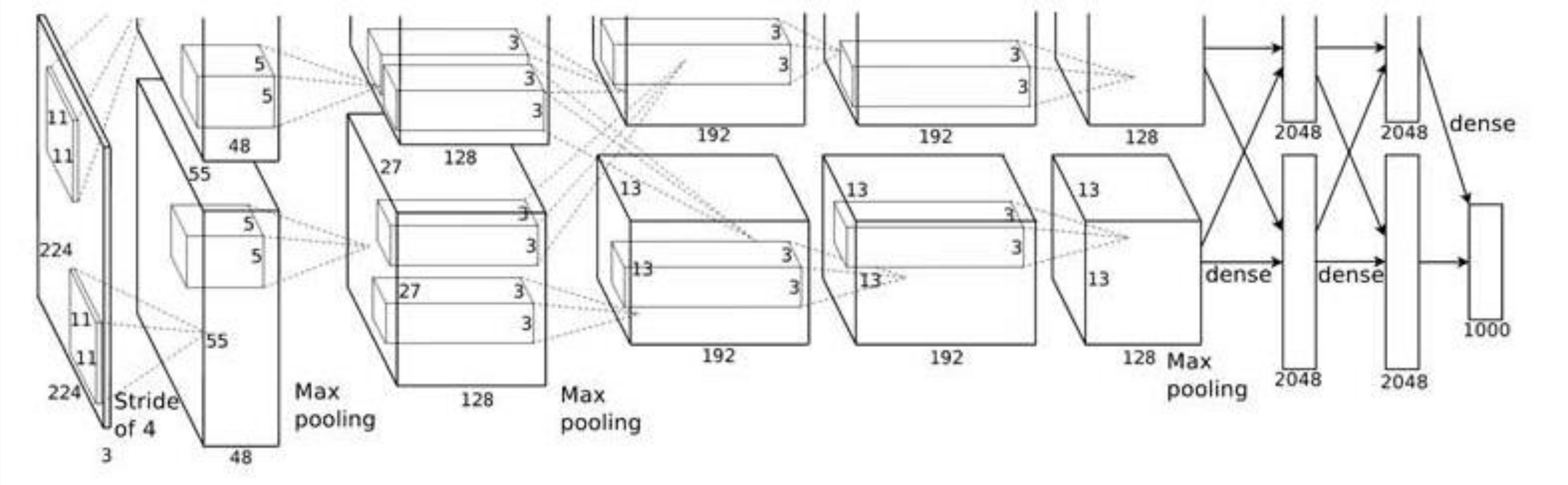
Abdel-Hamid, Ossama, et al. “Convolutional neural networks for speech recognition.” IEEE/ACM Transactions on audio, speech, and language processing 22.10(2014): 1533-1545.

- Image Net Large-Scale Visual Recognition Challenge(ILSVRC)
  - ImageNet DB : Via internet, collect 500 ~ 1000 images for 20,000 categories via the Internet.
  - Human labels via Amazon MTurk. (14 million labeled images, 20k classes.)
  - Challenge by classification, detection and positioning for 1000 categories. Error rate is confirmed by the 1st and 5st error rates.
  - Training images : 1.2 million, Valid images : 50,000, Test images : 150,000
  - By revealing structure and weights, the winning CNN becomes a widely used standard neural network.



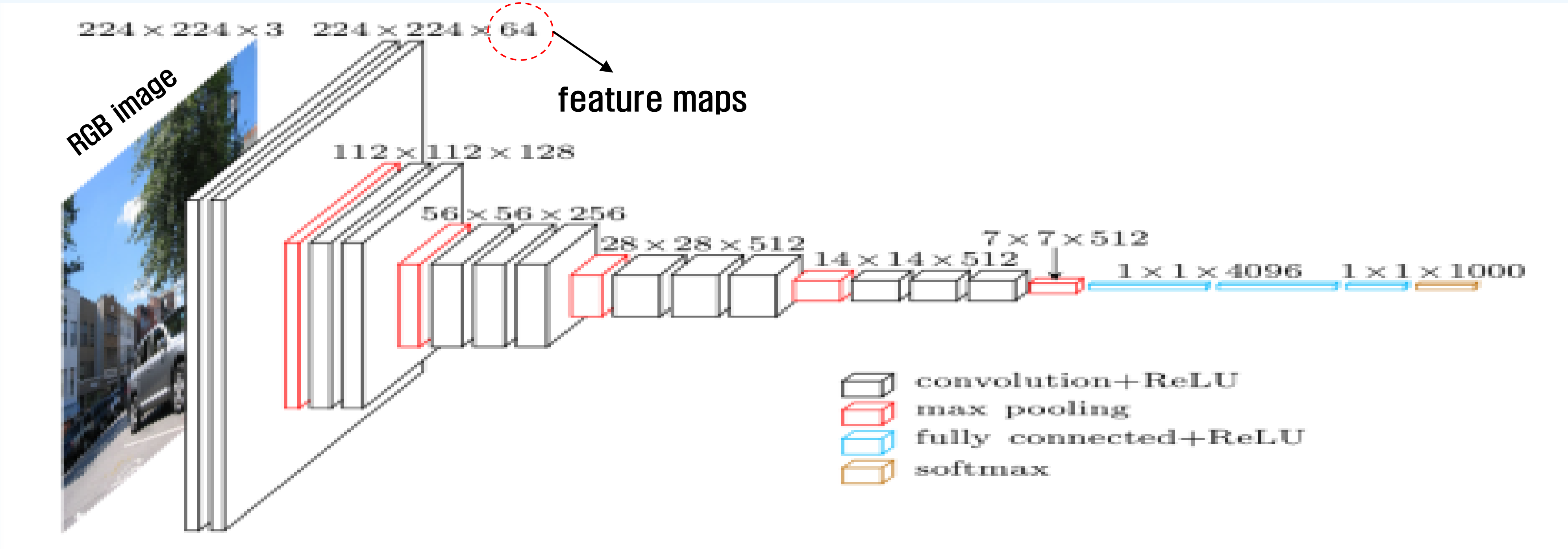
Prevailing CNNS (2/9) : AlexNet (by Alex Krizhevsky)

- 5 CNN layers+3 FC (fully connected) layers
- Node distribution at 8 layers: 290400–186624–64896–43264–4096–4096–1000
- Weights : $2x10^6$ at CNN layers and 6.5x107
- FC layers have 30 times more parameters. Future CNNs were developed to reduce parameters of FC layers
- Parallel processing by GPU
- ReLu, Regularization, Data Augmentation (2048 times), Dropout
- Reduce 2~3% error rate via Ensemble method



Prevailing CNNS (3/9) : VGG-16/19 (by Oxford University)

- Network architecture is simple. So, it is considered as a basic CNN architecture and frequently used.
- Small kernel size : 3x3, Convolution layer : 8~16 (more deep layers than AlexNet).
  ⇒ More layers with small kernels produce better performances.
- VGG-16 : 16 Conv layers [3 x (56x56x256) + 3 x (28x28x512) + 3 x (14x14x512)] + 3 FC layers
- VGG-19 : 19 Conv layers [4 x (56x56x256) + 4 x (28x28x512) + 4 x (14x14x512)] + 3 FC layers



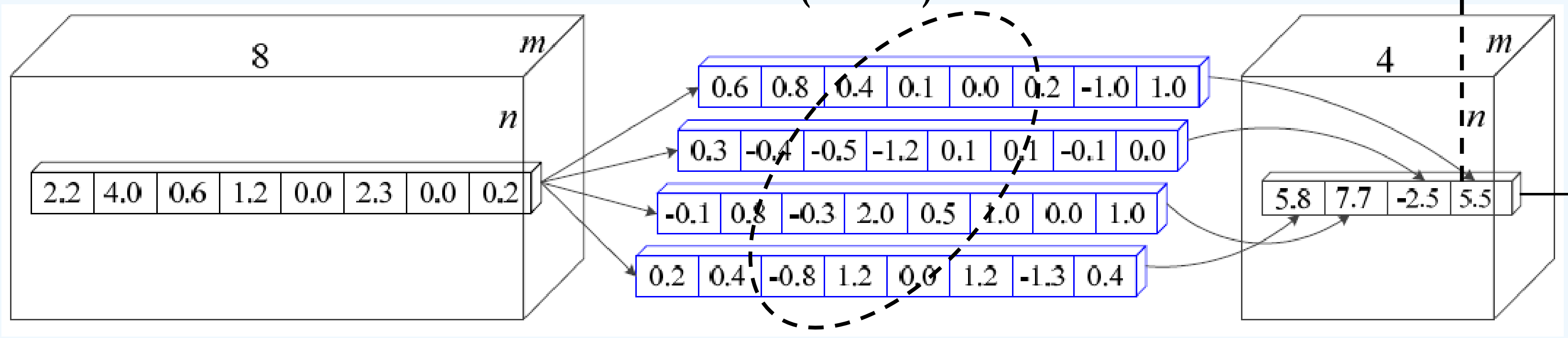
Prevailing CNNS (4/9) : 1x1 Kernel

- Network in Network (NIN) [Lin 2014]
- Reducing dimension :

  Ex: 8x(m*n) => 4x(m*n) (using four 8*1*1 kernels)

2.2*0.6 + 4.0*0.8 +0.6*0.4
+ 1.2*0.1+0.0*0.0+2.3*0.2
+0.0*(−1.0)+0.2*1.0 =5.54

(8*1*1)- 4kernels

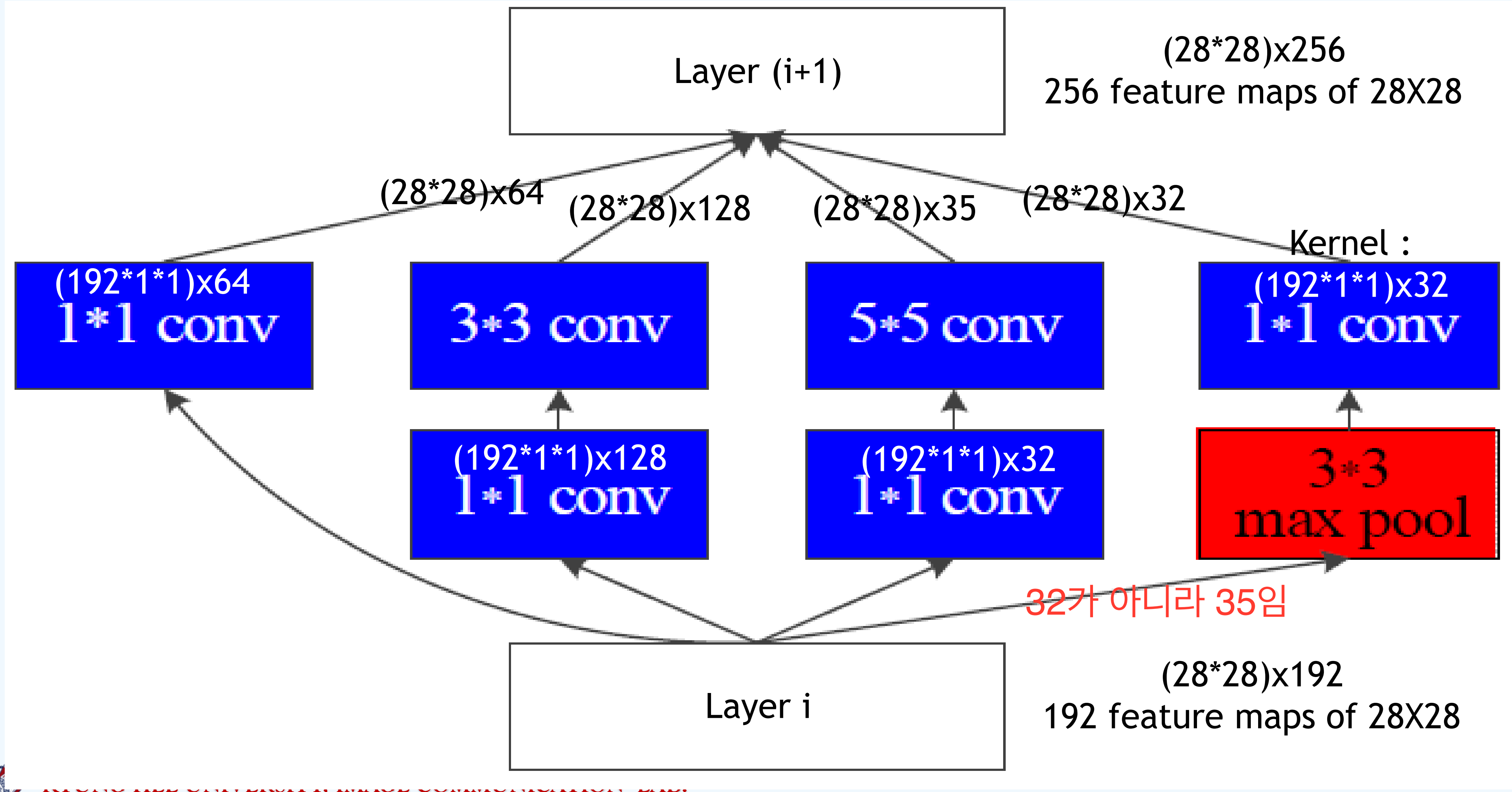


ReLU → [5.8, 7.7, 0.0. 5.5]

- With non-linear activation functions (ex: ReLU), more distinct, discriminative feature maps can be generated.
- VGGNet tried to use, bud did not use. GoogLeNet positively adopts 1x1 kernels.

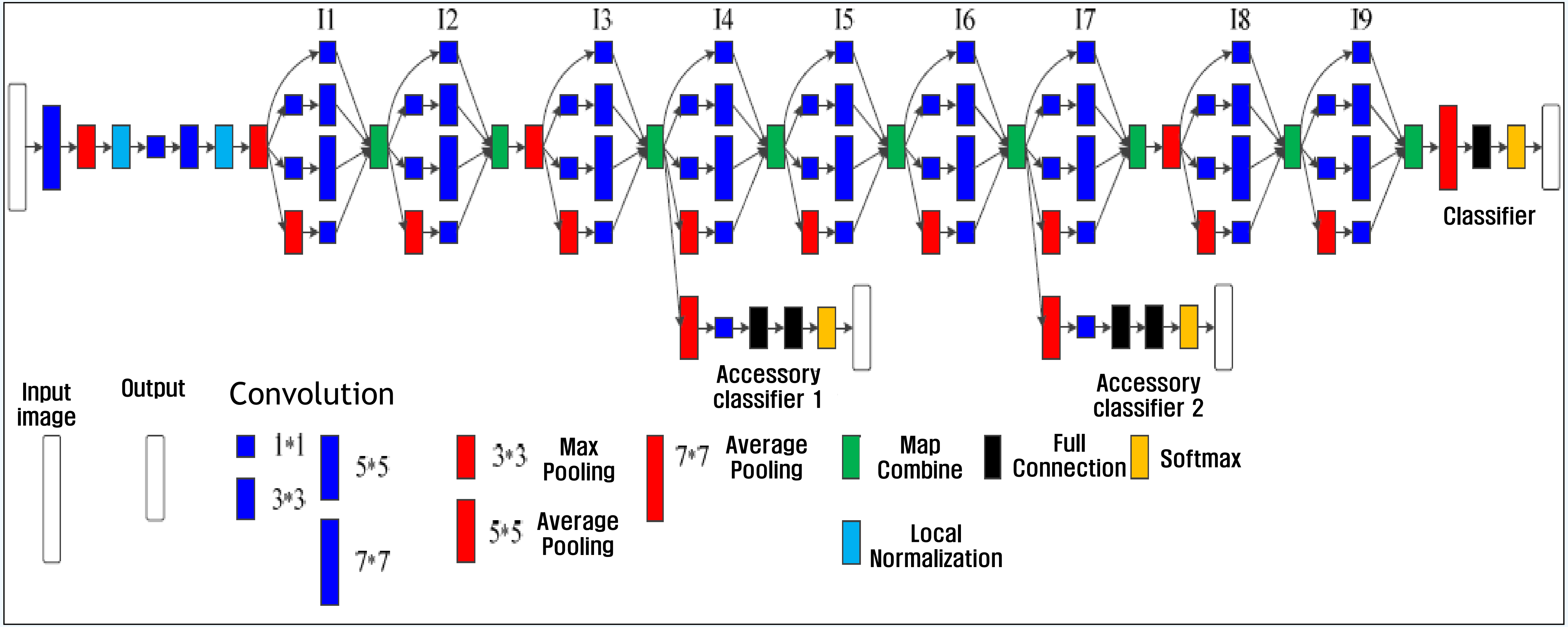
Prevailing CNNS (5/9) : GoogLe Net (1/2)

—Inception Module : NIN



Prevailing CNNS (6/9) : GoogLe Net (2/2)

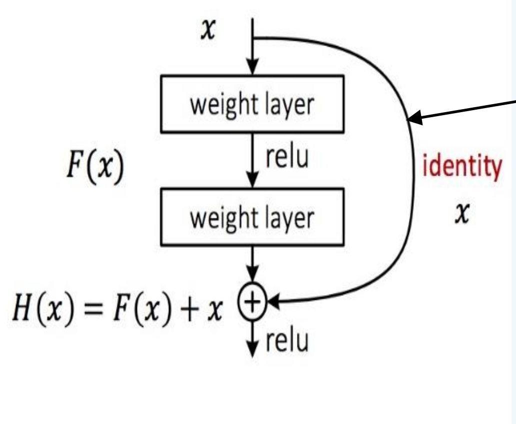
- 9 inception modules
- 27 layers : 22 layers having parameters + 5 layers without parameters
- 1 fully connected layer : 1 million parameters (1% of VGGNet)



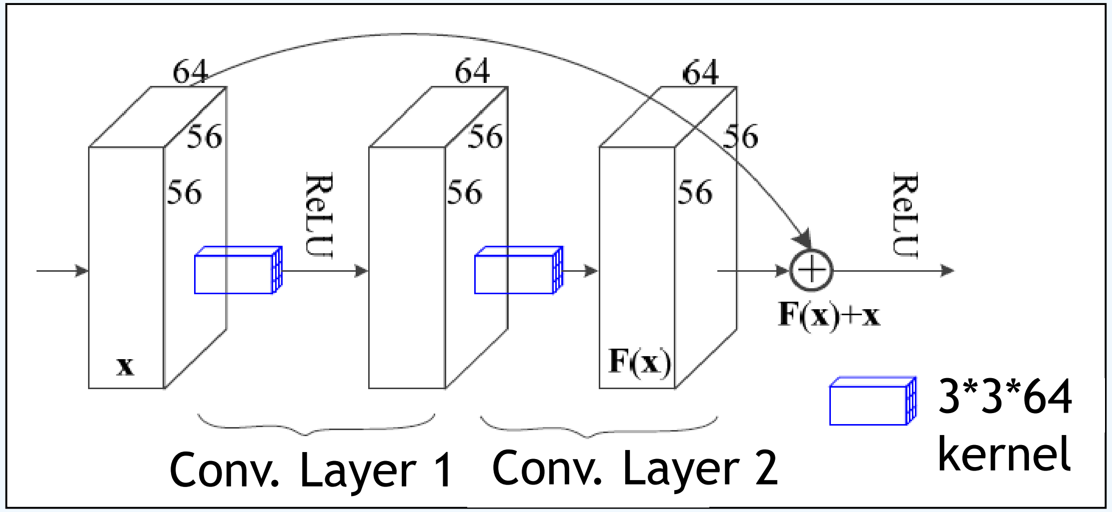
Prevailing CNNS (7/9) : ResNet -(by Microsoft-China) (1/2)

- Convolution Network : $\mathbf{F(x) = \tau(x \circledast w_1) \circledast w_2}$ + $\mathbf{y = \tau(F(x))}$
- Residual Network : $\mathbf{F(x) = \tau(x \circledast w_1) \circledast w_2}$ + $\mathbf{y = \tau(F(x) + x)}$ x : Short cut

Short-cut(or skip connection)



Through short-cut, some information skip hidden layers



-Its short-cut (or skip connection) solves vanishing gradient problem at Deep Neural Network.

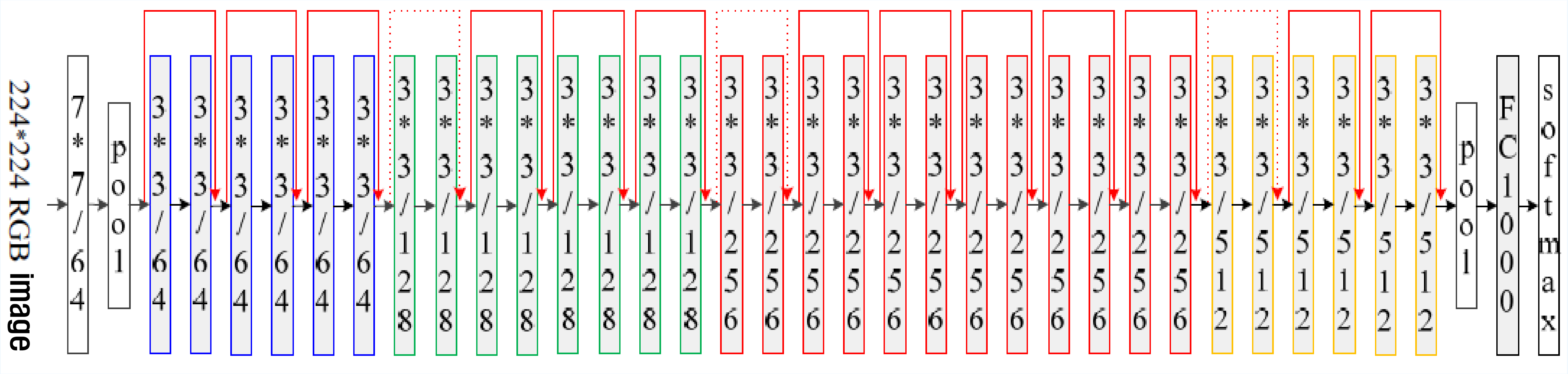
Back propagate Gradient : $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathbf{F}(\mathbf{x}_i)$ As layer deepens, more likely to be 0. => Vanishing momentum.

Residual Gradient : $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathbf{F}(\mathbf{x}_i)\right)$ Almost zero possibility to be -1. => Prevent vanishing momentum.

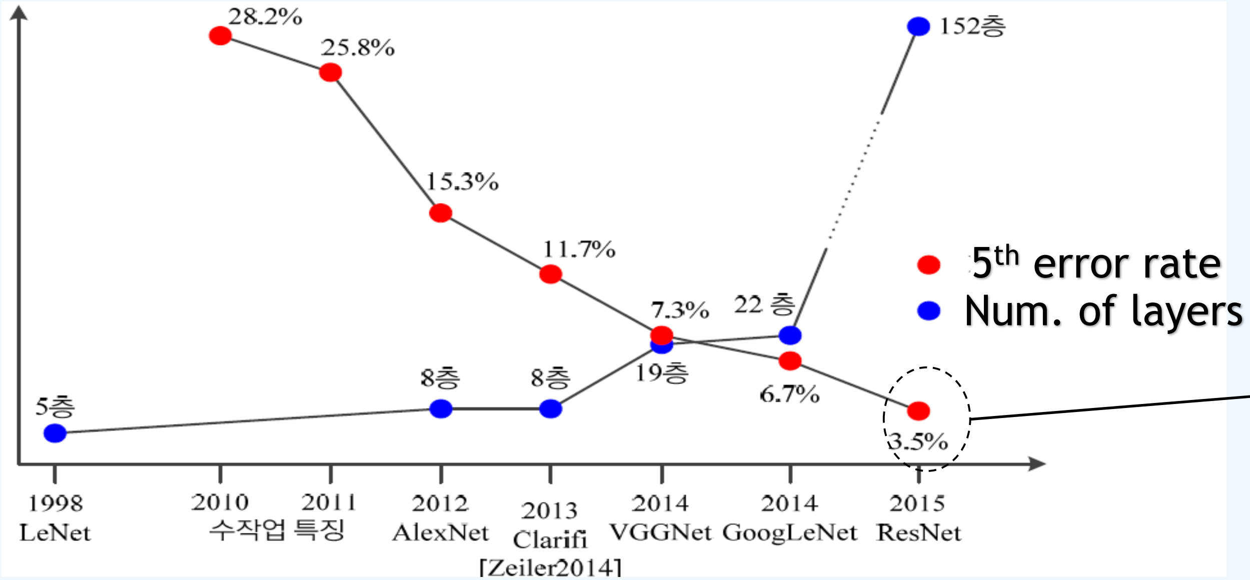
Prevailing CNNS (8/9) : ResNet (2/2)

- Due to short-cut, the network layer number can be 152.
- 3*3 Kernels are used. (same as in VGGNet)
- Not use FC. Use global average pooling. (Not same as in VGGNet)
- Use batch normalization, Not use dropout (Not same as in VGGNet)

- Example : 3*3 kernel, 34 layers



Prevailing CNNS (9/9) : Current Status



Reaches to human perception.

Classification challenge

Object recognition challenge