国際融合科学論/先端融合科学論

LECTURE 03

Machine Learning II: modern style machine learning

Dr. Suyong Eum

OSAKA UNIVERSITY

Lecture Outline

1) Neural networks

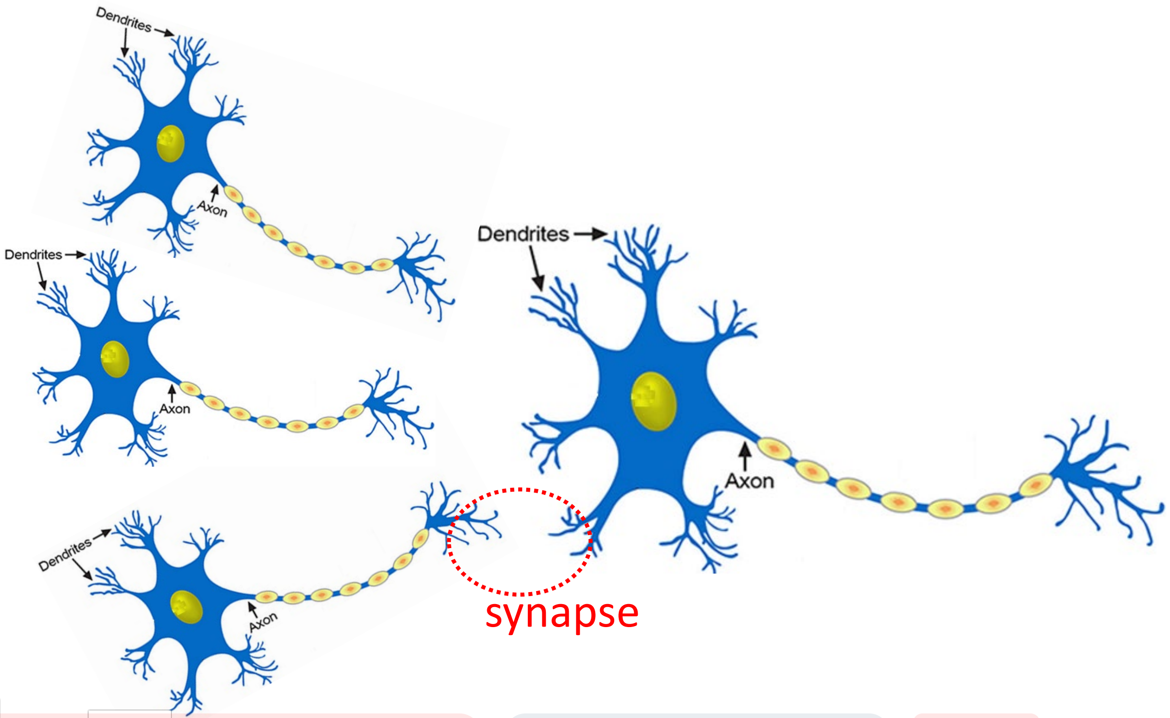
   - Architecture and its operation in detail such as backpropagation, activation functions and weight setting.

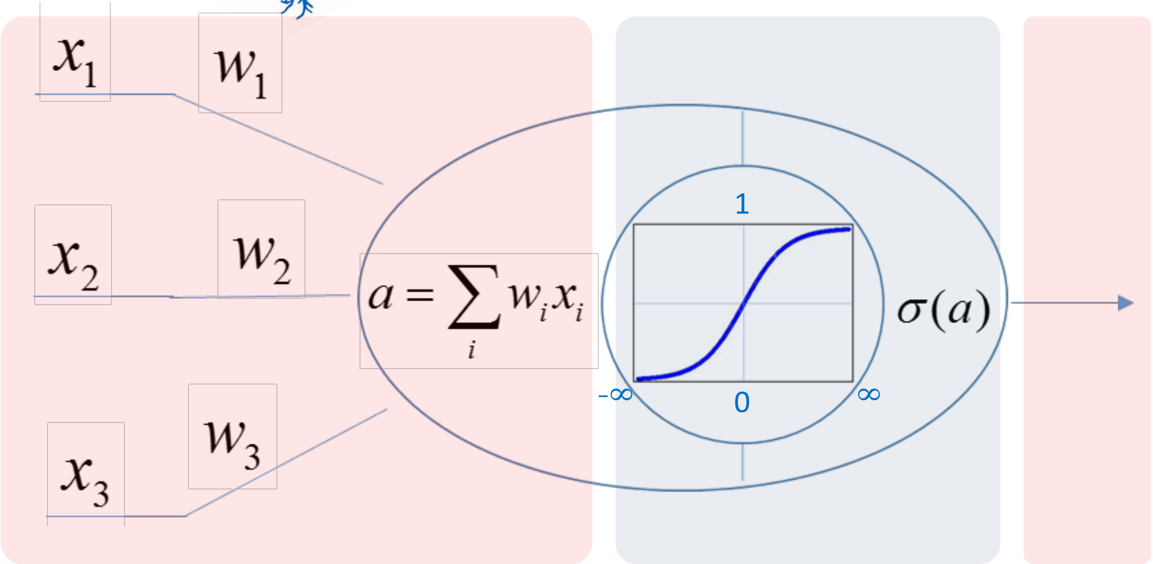
2) Convolutional Neural Networks: CNN

   - Architecture and its operation

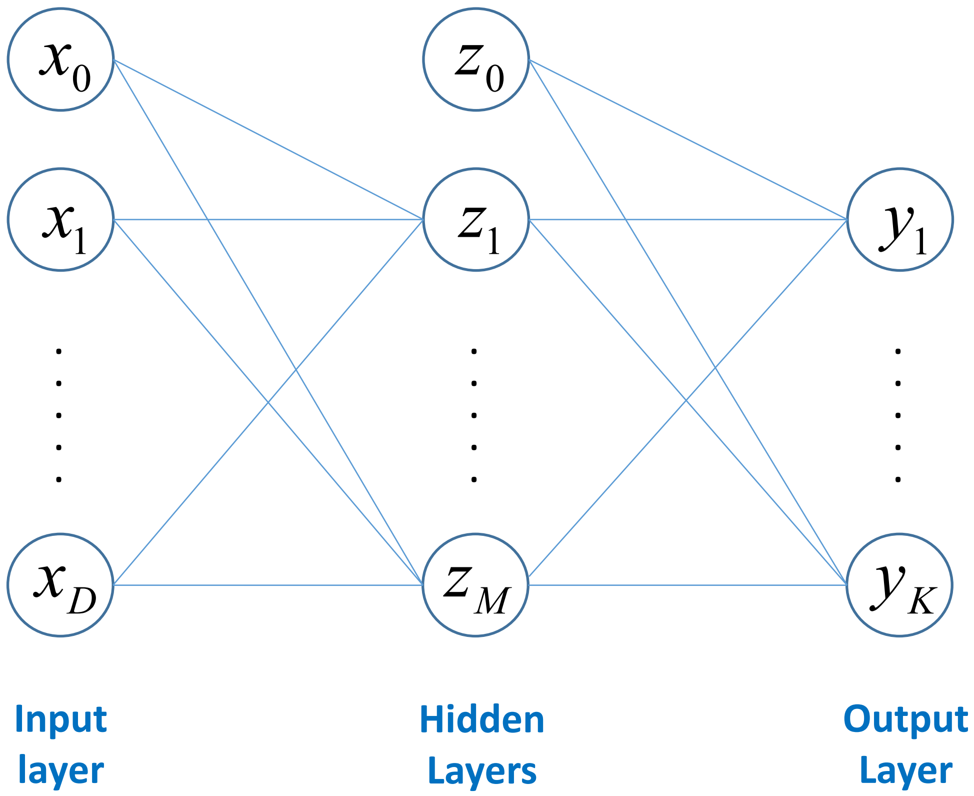
Neural Networks

Neural networks: A bio-inspired approach



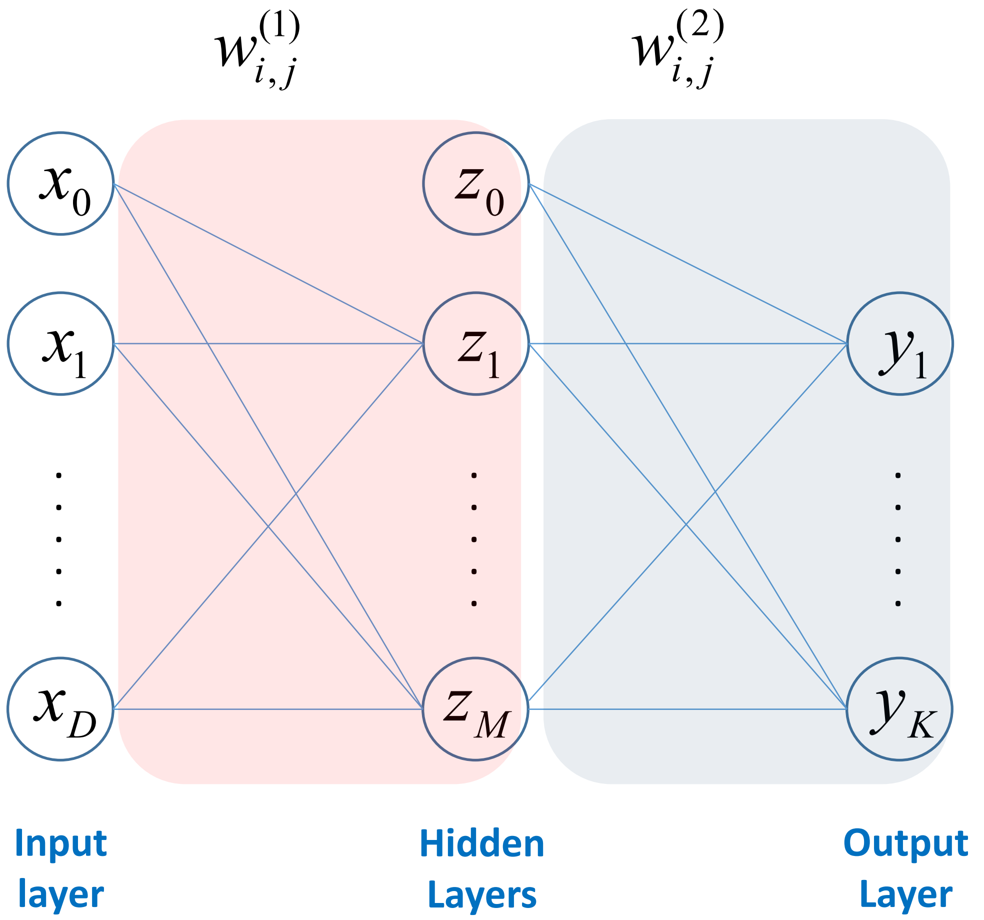


Neural networks: Terminology in neural networks



How many layers it has?

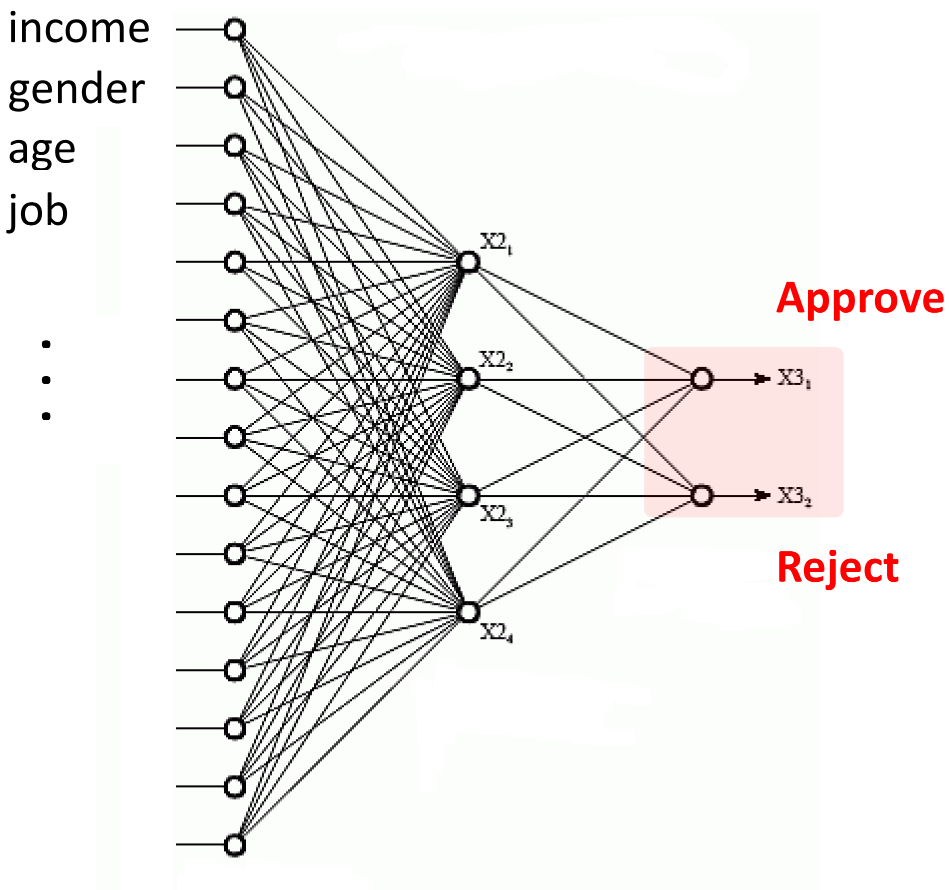
Neural networks: Terminology in neural networks



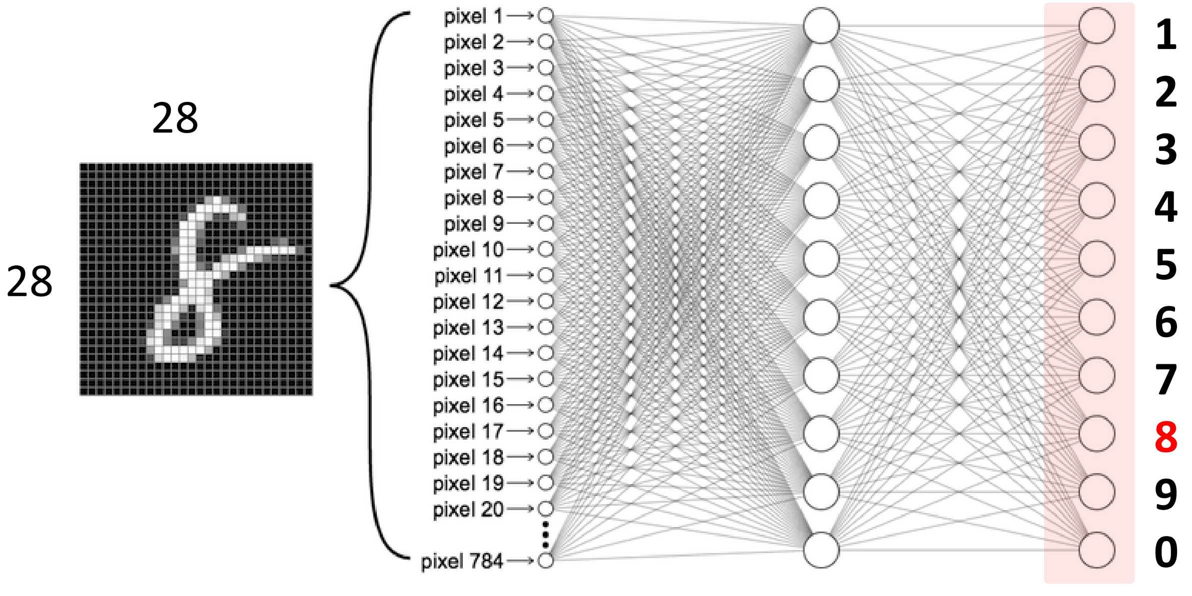
$w_{i,j}^{(\ell)}$: weight on a link at layer (ℓ) between node i and j

- In general, a standard L-layer neural network consists of
  - an input layer,
  - (L-1) hidden layers,
  - an output layer.

Neural networks: Example structures of neural networks

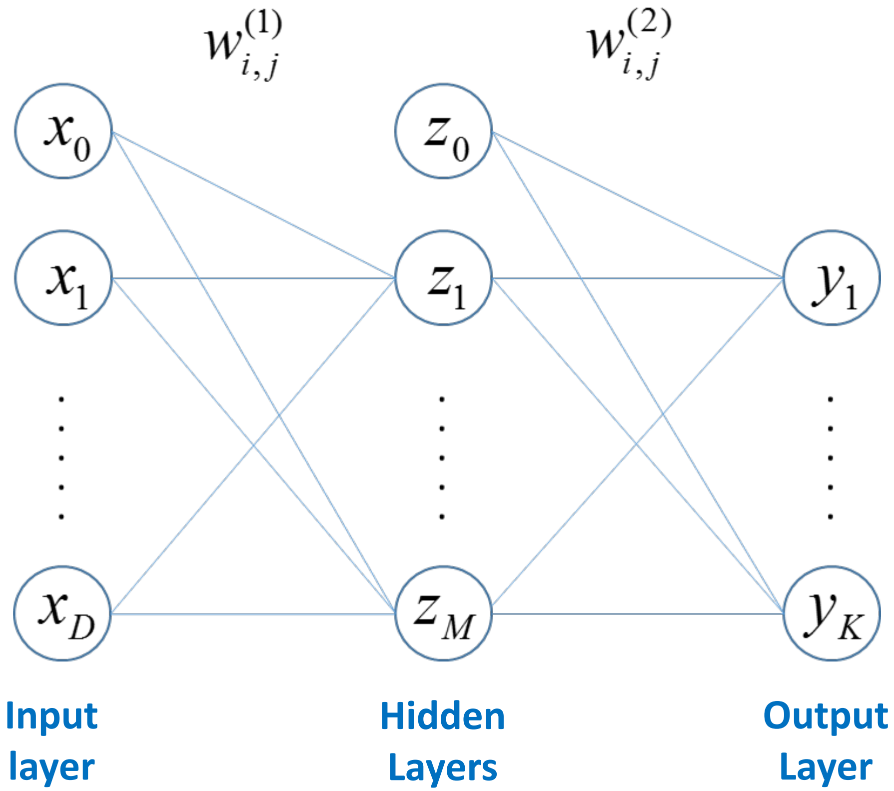


Credit card approval

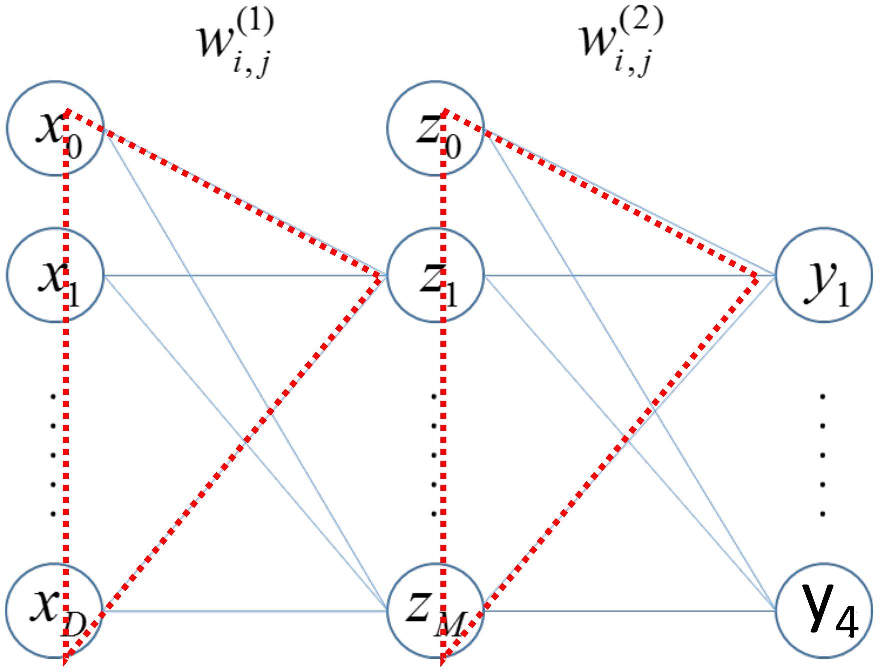


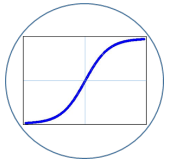
Digit recognition

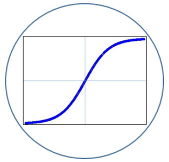
A neural network model



Neural networks: Cross entropy with Softmax







$$w^{(1)}_{1,1}x_1 + w^{(1)}_{2,1}x_2 + \cdots + w^{(1)}_{0,1}x_0$$

$$w^{(2)}_{1,1}z_1 + w^{(2)}_{2,1}z_2 + \cdots + w^{(2)}_{0,1}z_0$$

output		Sigmoid	Normalization	y: Softmax
-5	frog	0.00669	0.00334	0.00004
-1	bird	0.26894	0.13447	0.00243
1	dog	0.73106	0.36553	0.01794
5	cat	0.99331	0.49666	0.97959

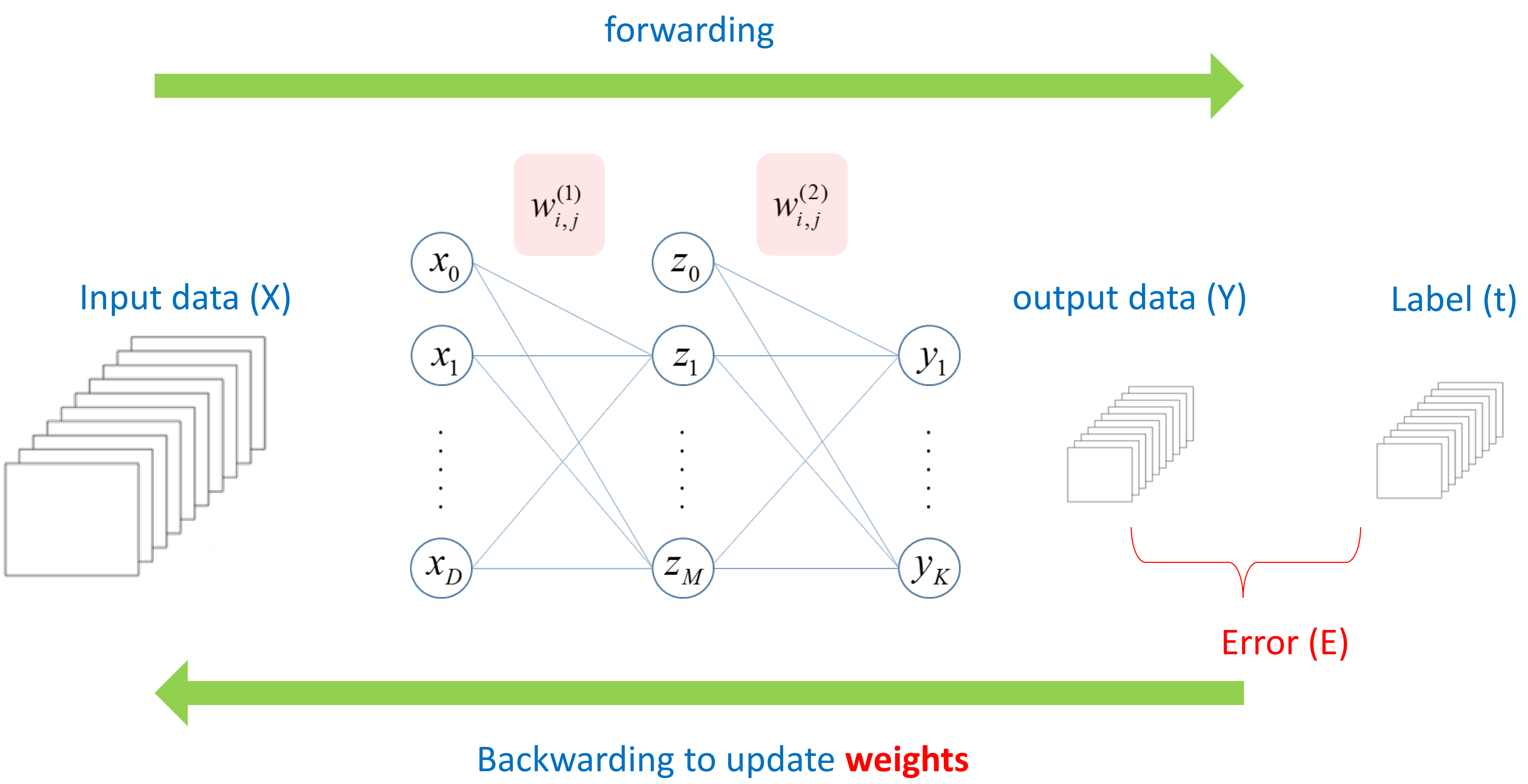
t

Label
0
0
0
1

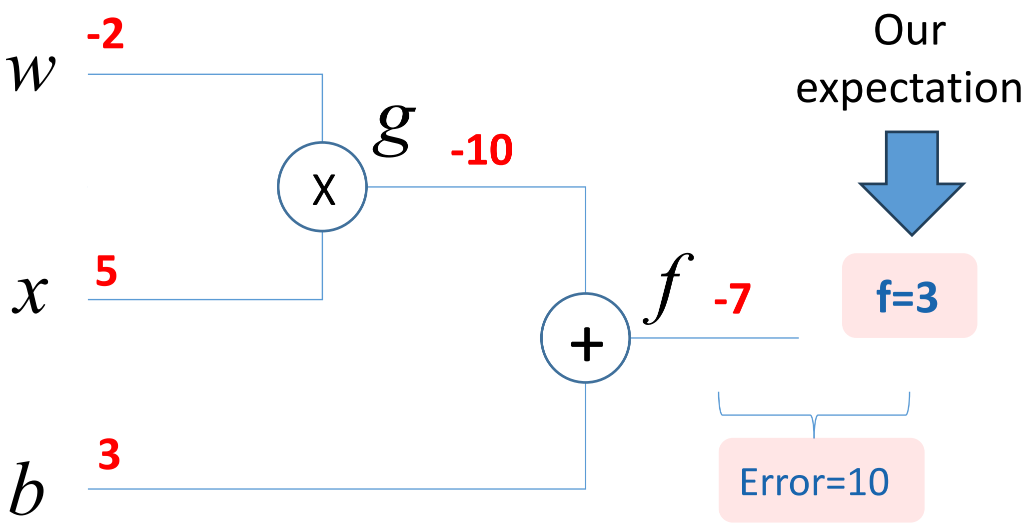
$$H(y) = -\sum_{i} t_i \ln(y_i) = 0.020621$$

ERROR between output (y) and label (t)

Overview of the operation



Backpropagation: a toy example



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

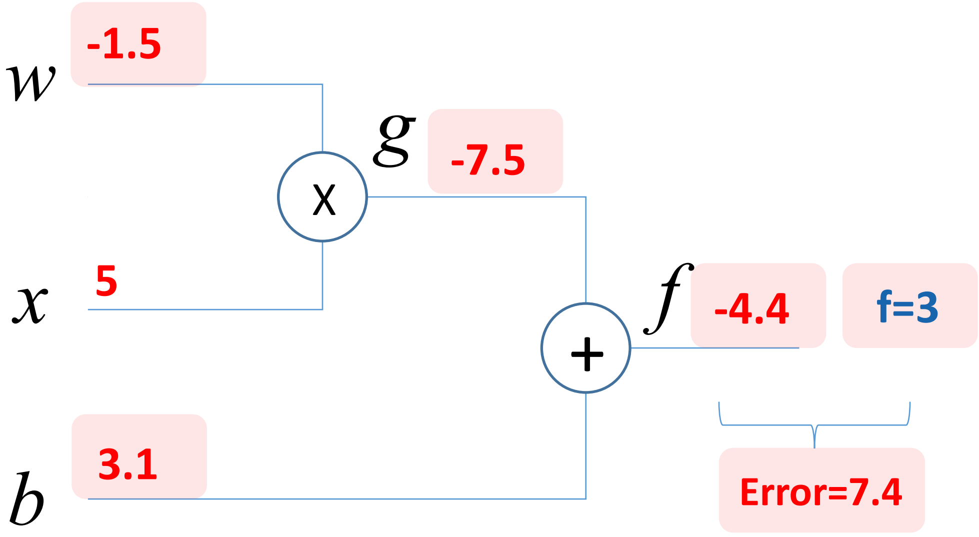
$$g = wx$$

- Assuming that the value of f should be “3”.
- How to update variables which you are interested?

$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

Backpropagation: a toy example: $\eta = 0.1$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

- ❑ Assuming that the value of f should be "3".
- ❑ How to update variables which you are interested?

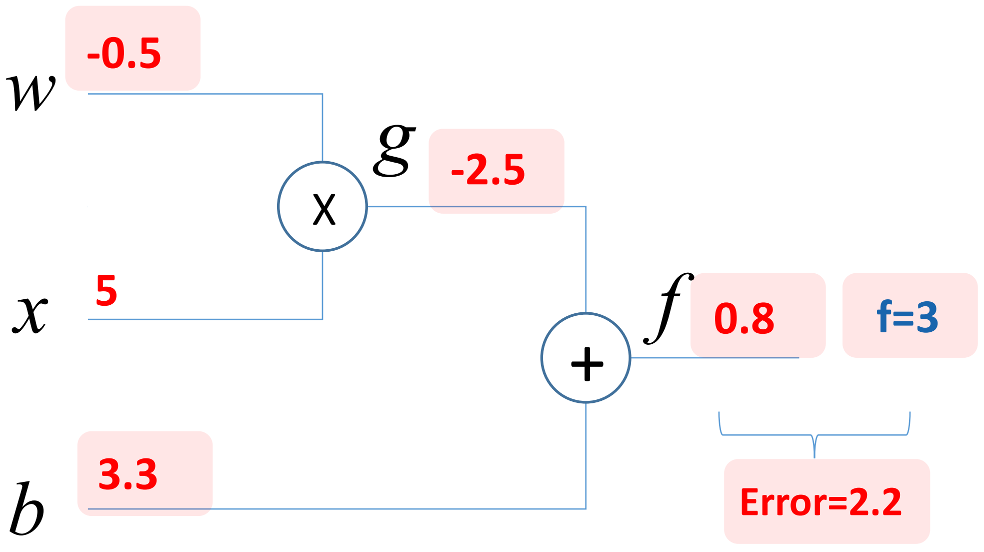
$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

$$W_{new} = -2 + 0.1 \times 5 = -1.5$$

$$b_{new} = 3 + 0.1 \times 1 = 3.1$$

Backpropagation: a toy example: $\eta = 0.3$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

- ❑ Assuming that the value of f should be "3".
- ❑ How to update variables which you are interested?

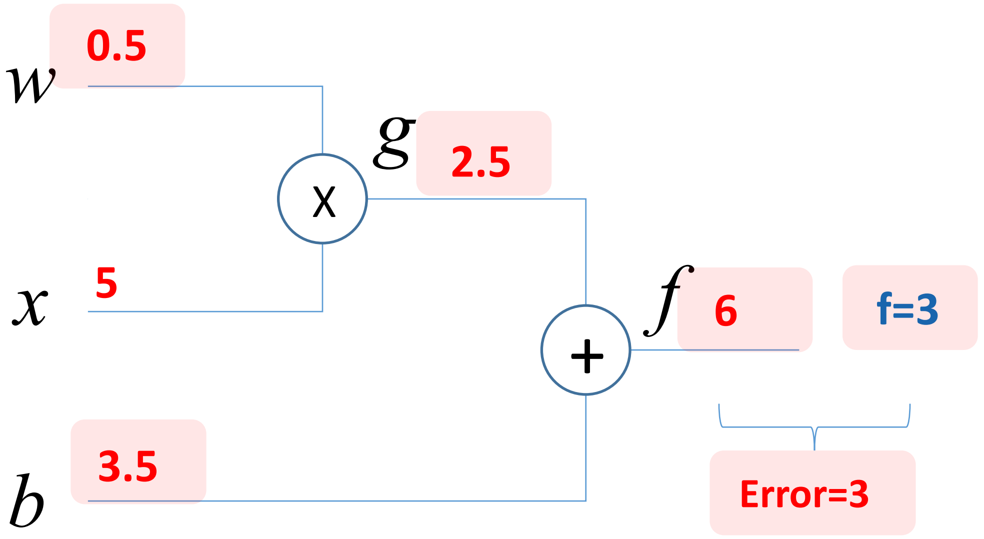
$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

$$W_{new} = -2 + 0.3 \times 5 = -0.5$$

$$b_{new} = 3 + 0.3 \times 1 = 3.3$$

Backpropagation: a toy example: $\eta = 0.5$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

- ❑ Assuming that the value of f should be "3".
- ❑ How to update variables which you are interested?

$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

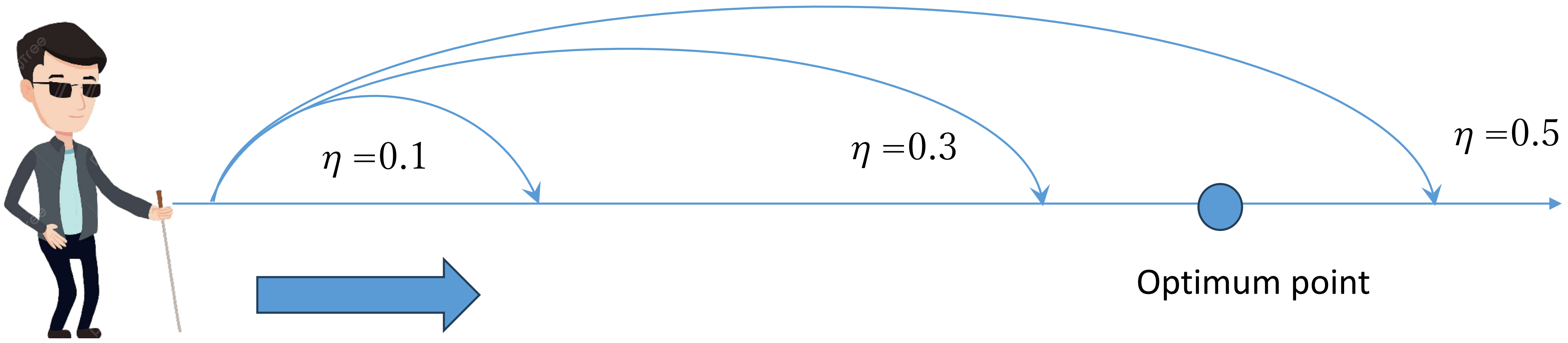
$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

$$W_{new} = -2 + 0.5 \times 5 = 0.5$$

$$b_{new} = 3 + 0.5 \times 1 = 3.5$$

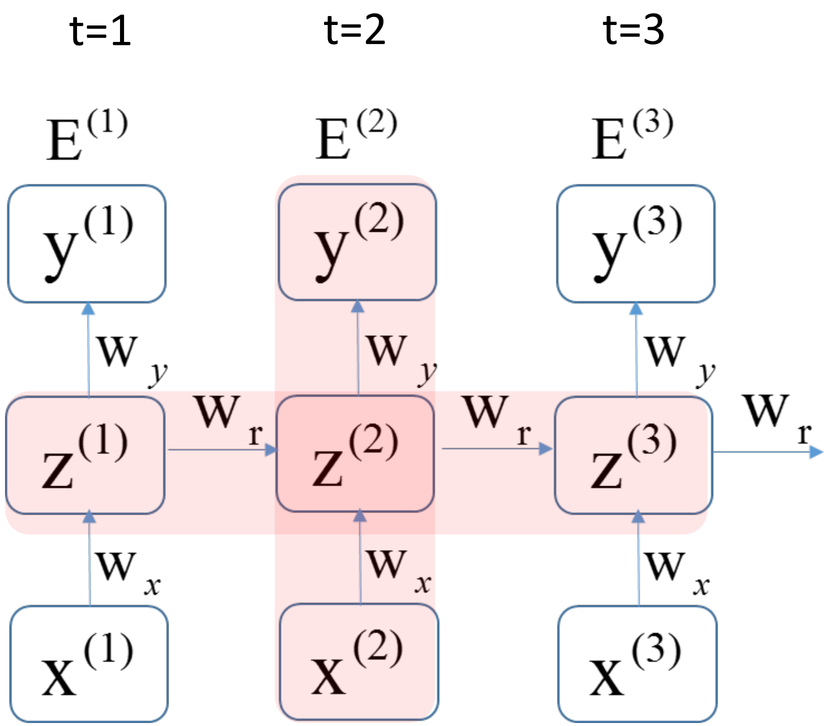
Backpropagation: a toy example

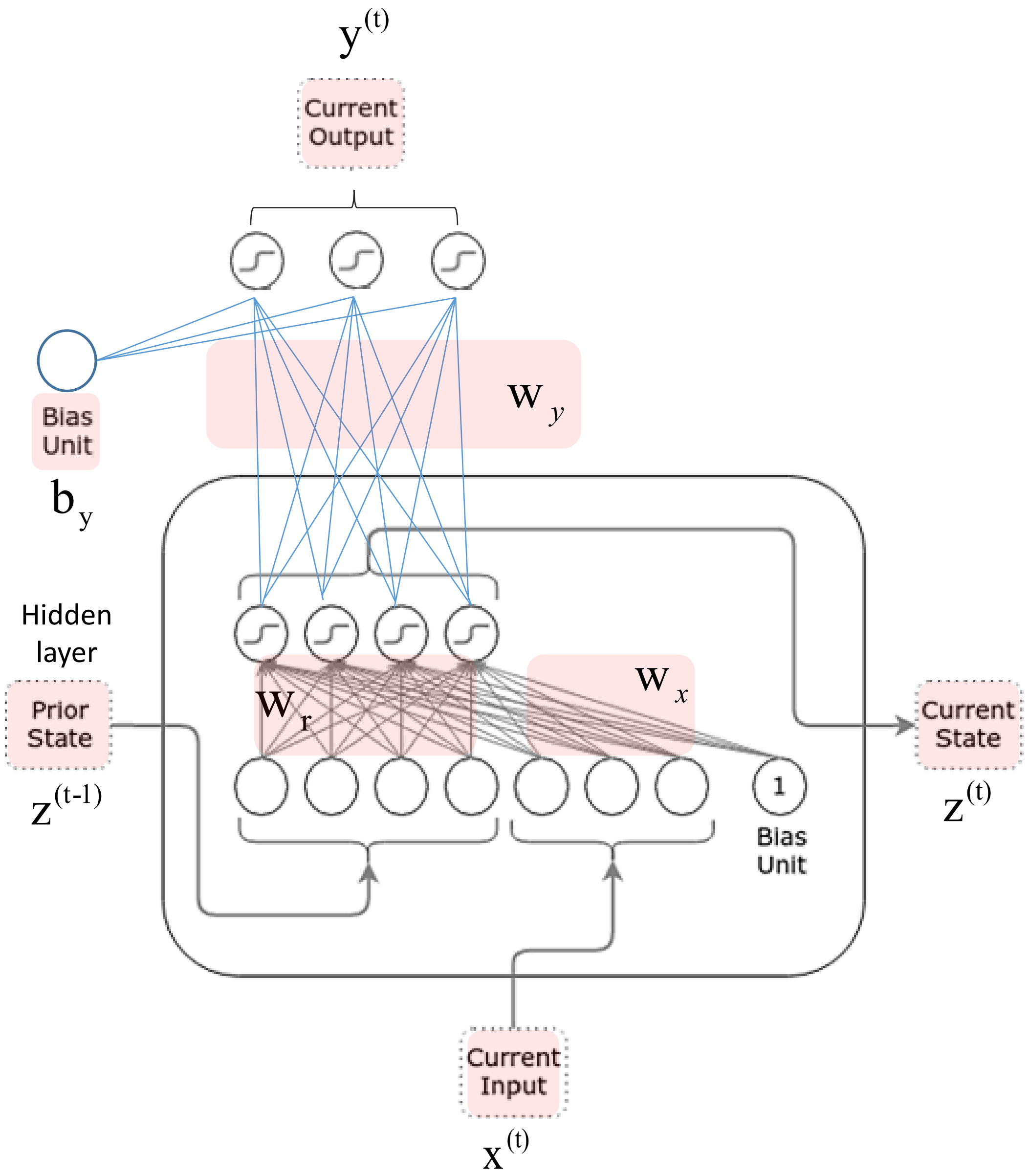
$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i} \qquad b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$



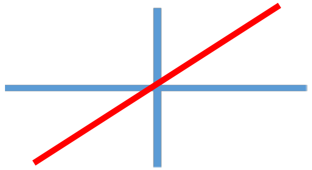
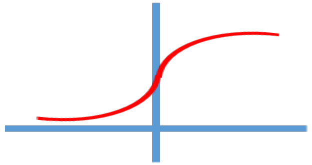
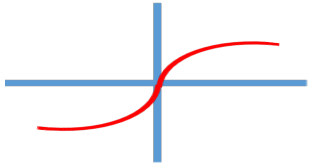
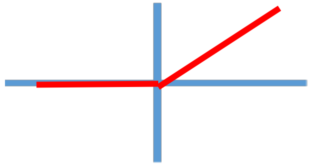
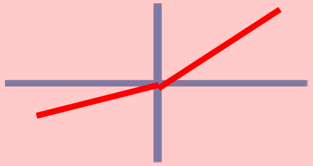
$\frac{\partial f}{\partial w_i}$ $\frac{\partial f}{\partial b}$ Direction !

Another type of neural network: Vanilla RNN





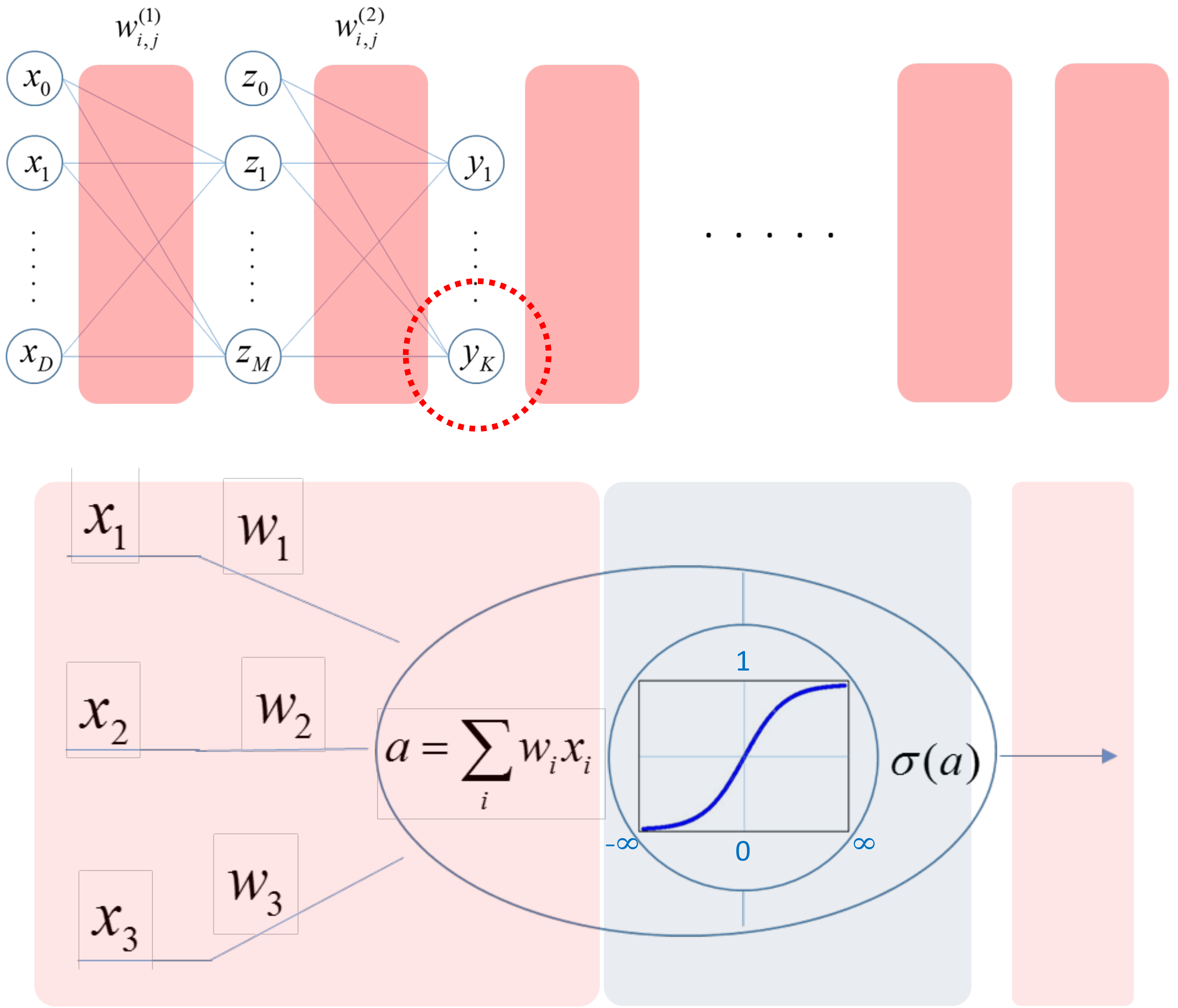
Neural networks: Activation function

Linear	
Sigmoid	
TanH	
ReLU	
Leaky ReLU	

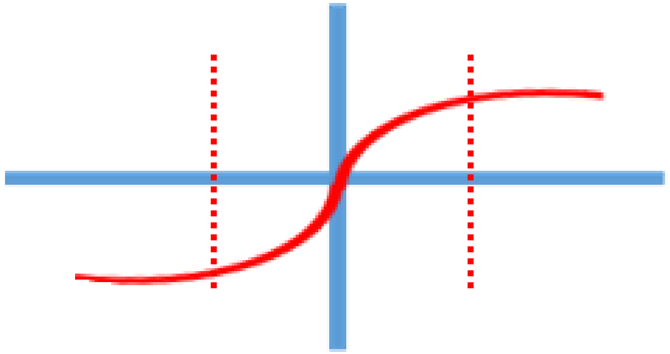
- ❑ Cannot apply backpropagation to find how neural weights should change to reduce the error found.
- ❑ Saturated neuron stops the backpropagation due to the zero gradient at both ends.
- ❑ Non-zero centered: data coming into a neuron is always positive.
- ❑ Zero centered… but the computation of exp() is expensive.
- ❑ The convergence speech with ReLU is 6 times faster than TanH [1]
- ❑ Zero centered and fast convergence …

[1] ImageNet Classification with Deep Convolutional Neural Networks

- How do we set the weight of each link initially?

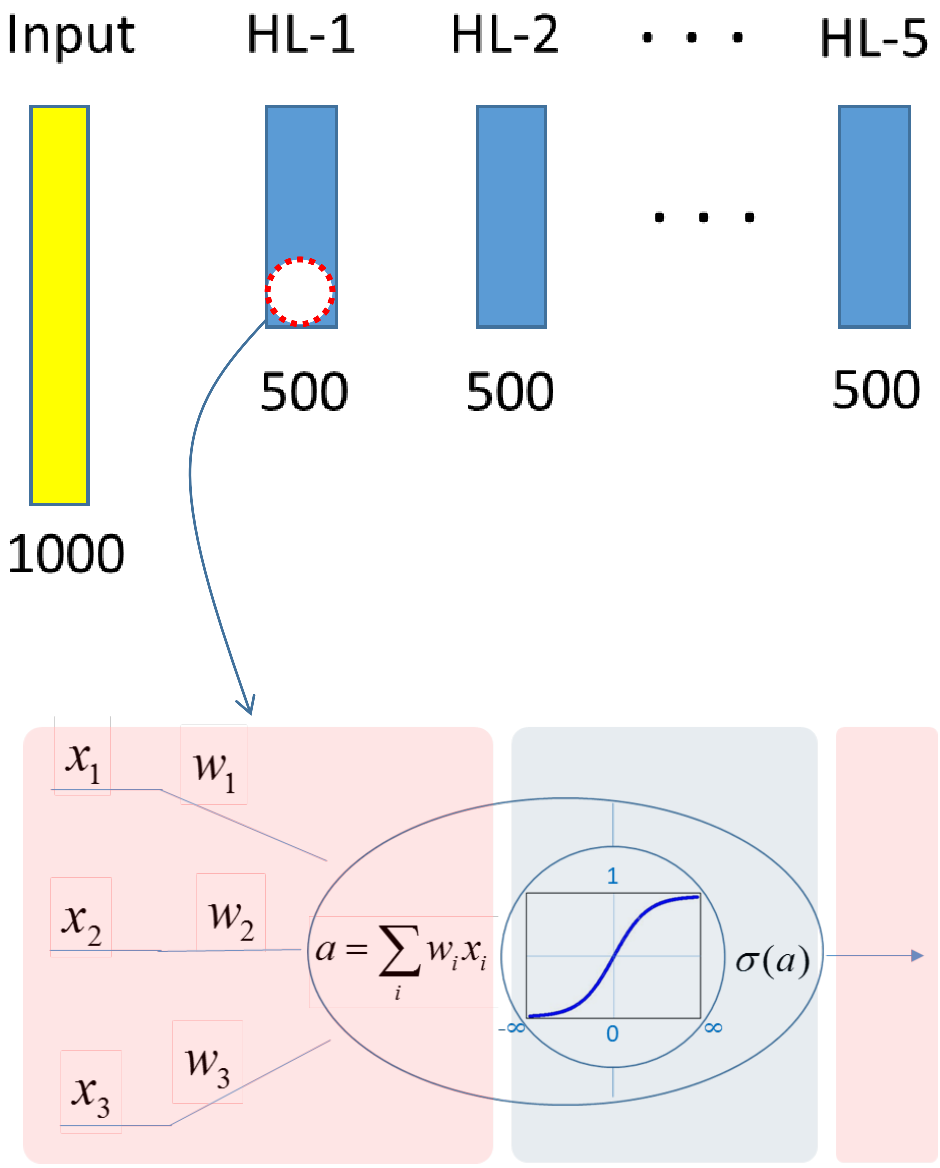


- An input to an activation function had better be within a certain range rather than either an extremely large or small value.



Neural networks: Weight initialization experiment

- A neural network is created as shown in the right, e.g., with 5 layers.
  - Each of the 1000 inputs is drawn from N(0, 1) and goes through the 5 hidden layers,
  - Then, the outputs of each hidden layer, e.g., after activation function, are plotted.



Neural networks: Weight initialization experiment

- ❑ How about random setting?   **Tanh**
  - N (mean=0, std=1)

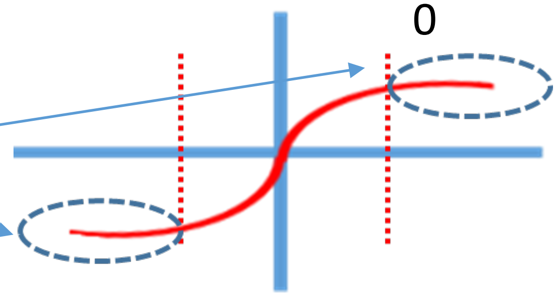
- ❑ Random setting with smaller std?
  - N (mean=0, std=0.01)

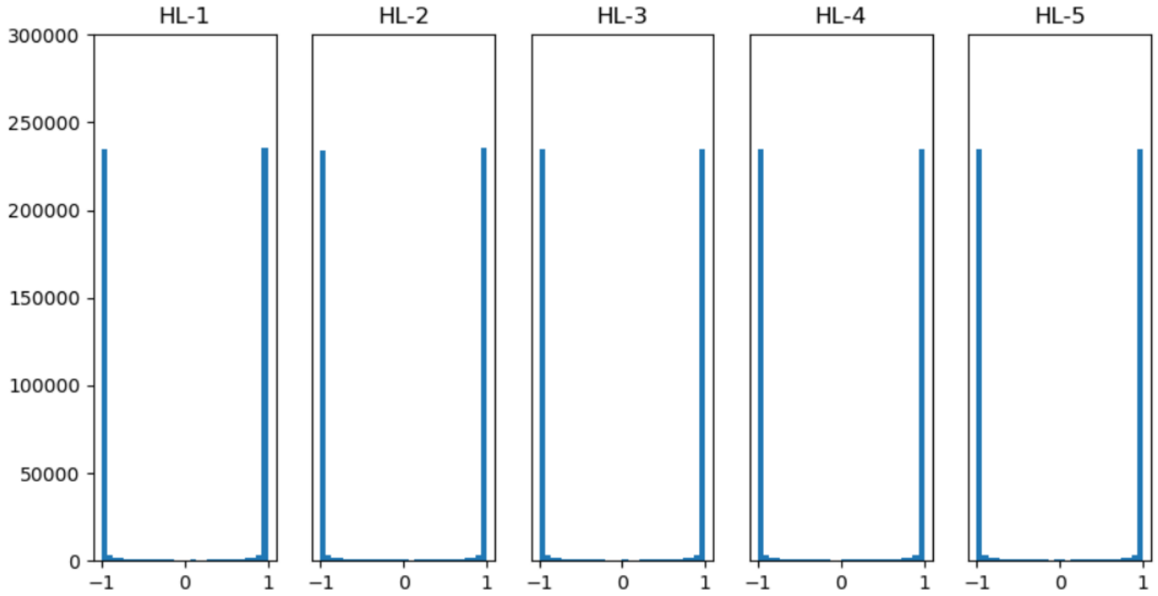
- ❑ How about Xavier initialization?
  - N (mean=0, std= $\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- ❑ How about He initialization?
  - N (mean=0, std= $\sqrt{\frac{4}{fan\ in + fan\ out}}$)

- ❑ The output values of the activation functions, tanh(), in each hidden layer are mostly distributed at -1 and 1
- ❑ Vanishing gradient problem

$$\frac{\partial E(w)}{\partial w_1} = \frac{\partial E(\sigma(a))}{\partial \sigma} \times \frac{\partial \sigma}{\partial a} \times \frac{\partial a}{\partial w_1}$$





Neural networks: Weight initialization experiment

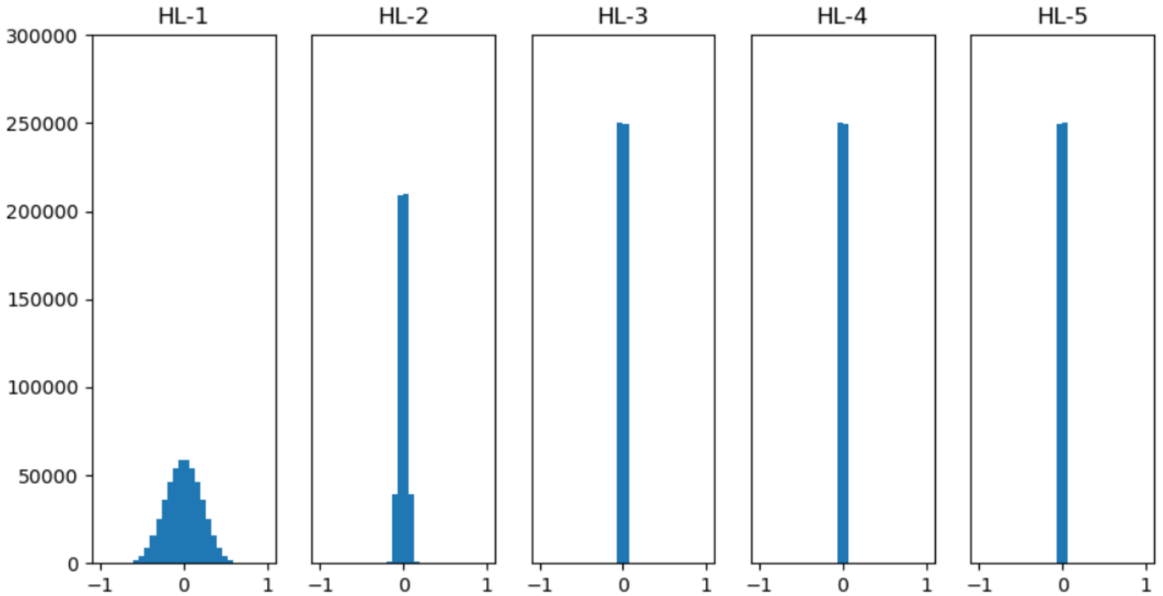
- ❑ How about random setting? **Tanh**
  - N (mean=0, std=1)

- ❑ Random setting with smaller std? **Tanh**
  - N (mean=0, std=0.01)

- ❑ How about Xavier initialization?
  - N (mean=0, std= $\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- ❑ How about He initialization?
  - N (mean=0, std= $\sqrt{\frac{4}{fan\ in + fan\ out}}$)

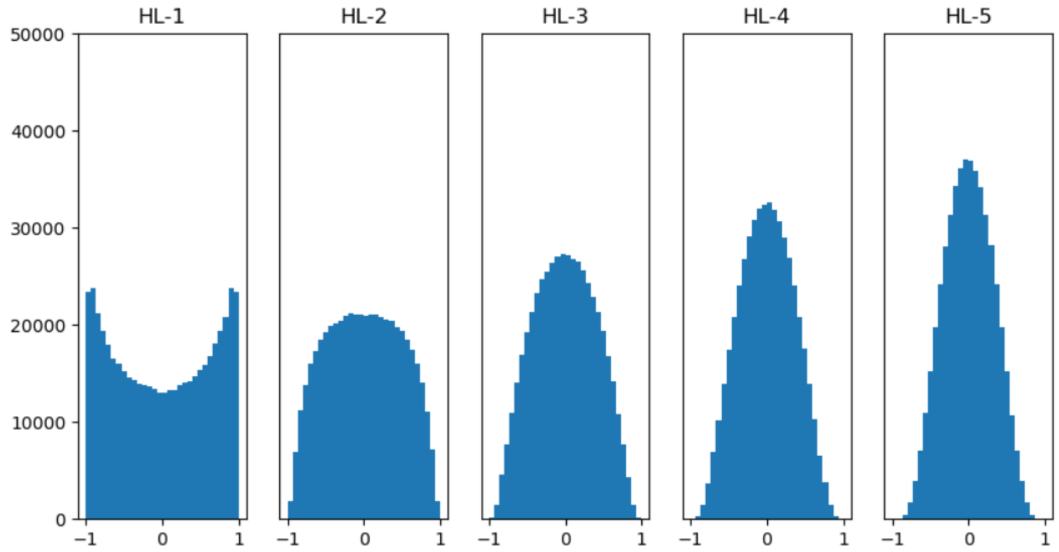
- ❑ It solves the vanishing gradient problem but each weight tends to have same value,
- ❑ which implies some learning problem.



Neural networks: Weight initialization experiment

- ❑ How about random setting? **Tanh**
  - N (mean=0, std=1)
- ❑ Random setting with smaller std? **Tanh**
  - N (mean=0, std=0.01)
- ❑ How about Xavier initialization? **Tanh**
  - N (mean=0, std= $\sqrt{\frac{2}{fan\ in+fan\ out}}$)
- ❑ How about He initialization?
  - N (mean=0, std= $\sqrt{\frac{4}{fan\ in+fan\ out}}$)

- ❑ If S-curve function, e.g., sigmoid or tanh, is used as an activation function, Xavier is a way to initialize weight
- ❑ Solving the vanishing gradient and learning issue shown previously, e.g., well distributed
- ❑ STD is a function of the number of neurons in each hidden layer



Neural networks: Weight initialization experiment

- How about random setting? **Tanh**
  - N (mean=0, std=1)

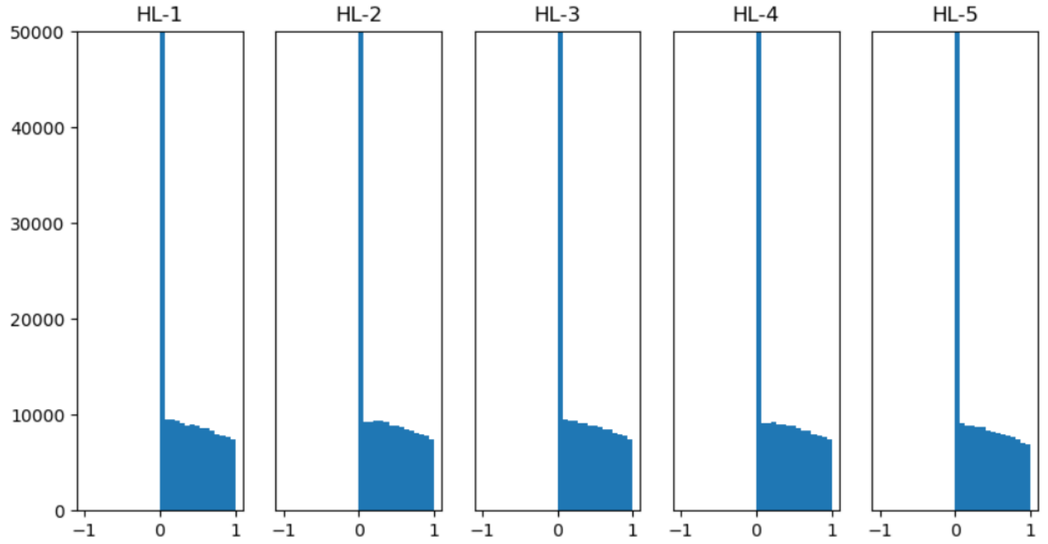
- Random setting with smaller std? **Tanh**
  - N (mean=0, std=0.01)

- How about Xavier initialization? **Tanh**
  - N (mean=0, std= $\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- How about He initialization? **Relu**
  - N (mean=0, std= $\sqrt{\frac{4}{fan\ in + fan\ out}}$)

- As mentioned previously, Relu is 6 times faster than s-curve function.

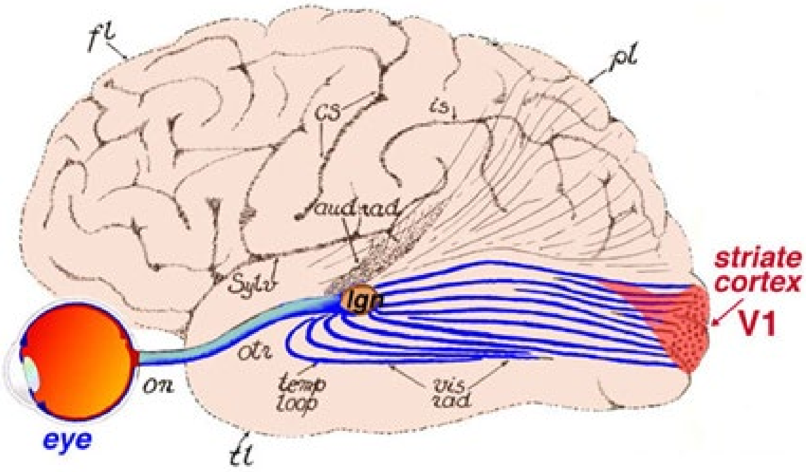
- "He" is a choice for weight initialization when Relu is used as an activation function.

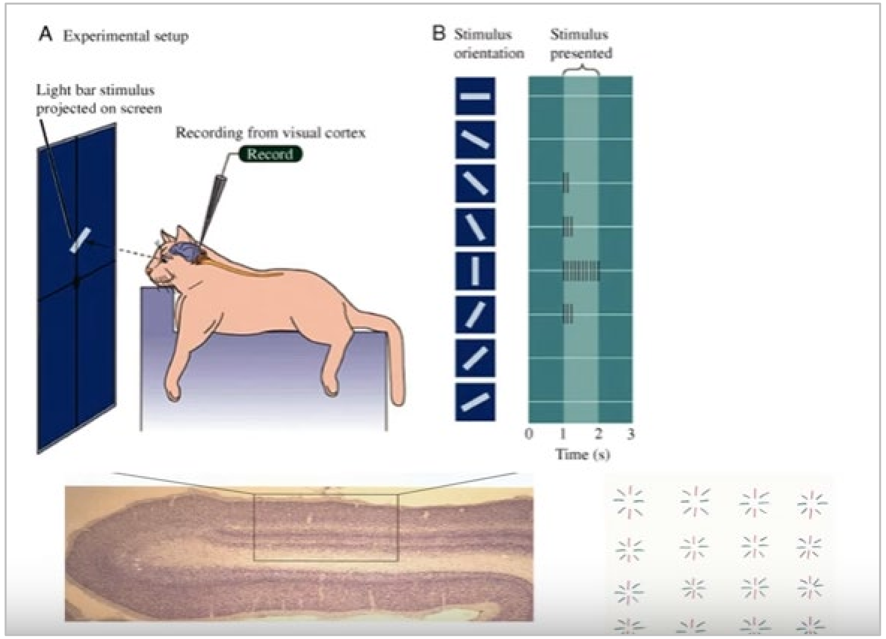


Convolutional Neural Networks (CNN)

CNN: Its History

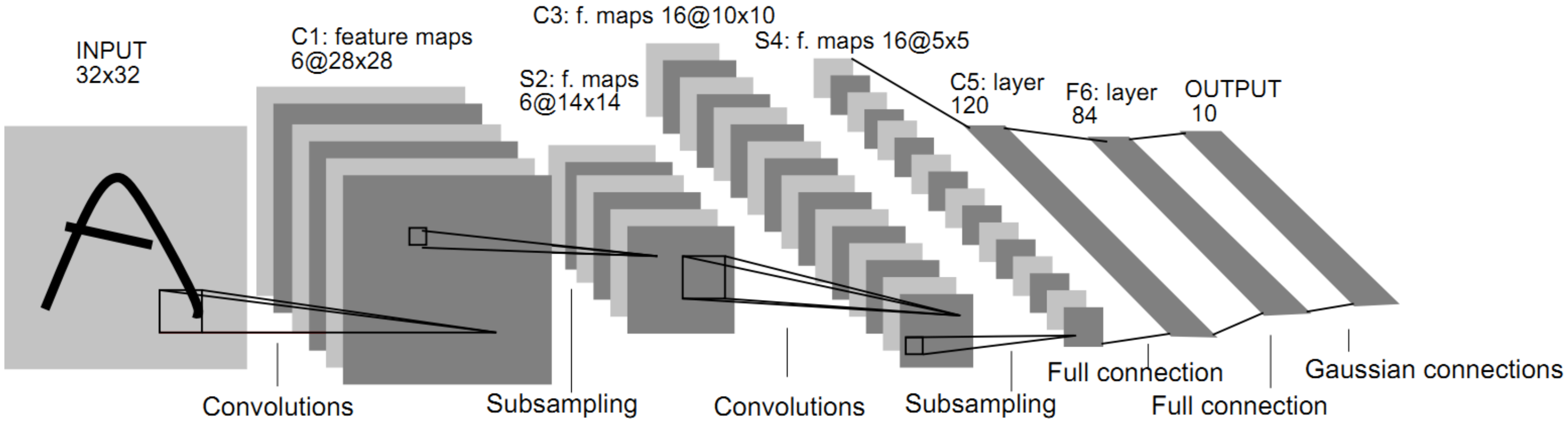
- Hubel and Wiesel in 1962 showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation.
- They found out that these neurons were organized in such an architecture and that together, they were able to produce visual perception.
- In other words, the neuronal cells in the visual cortex look for specific characteristics, edges of an object, and use them to recognize the object
- This is the basis behind CNNs.



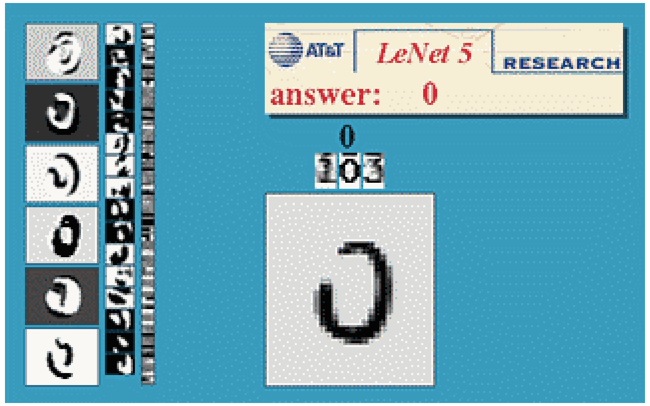


https://www.youtube.com/watch?v=v20-E_2bT2c

- LeCun et al, "Convolutional Networks for Images, Speech, and Time-Series"
  - http://yann.lecun.com/exdb/publis/pdf/lecun-bengio-95a.pdf



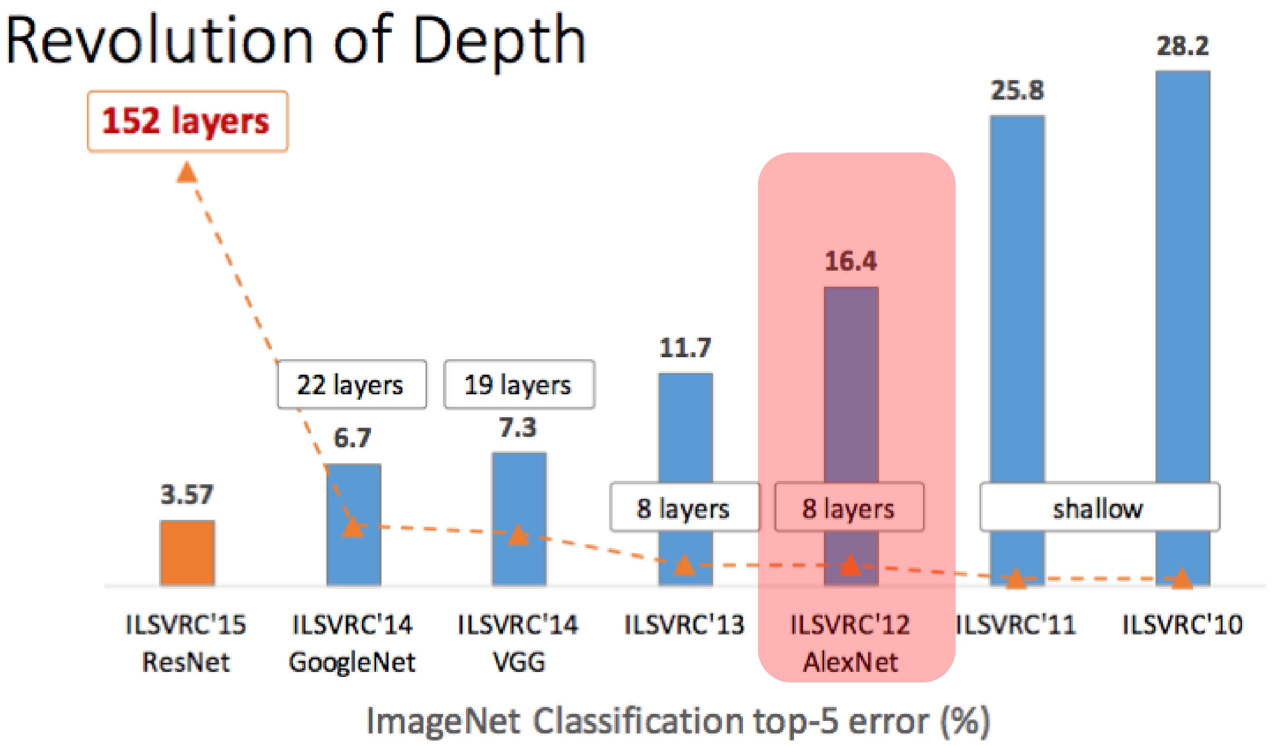
Lecun Yann is the former director of AI Research, Facebook

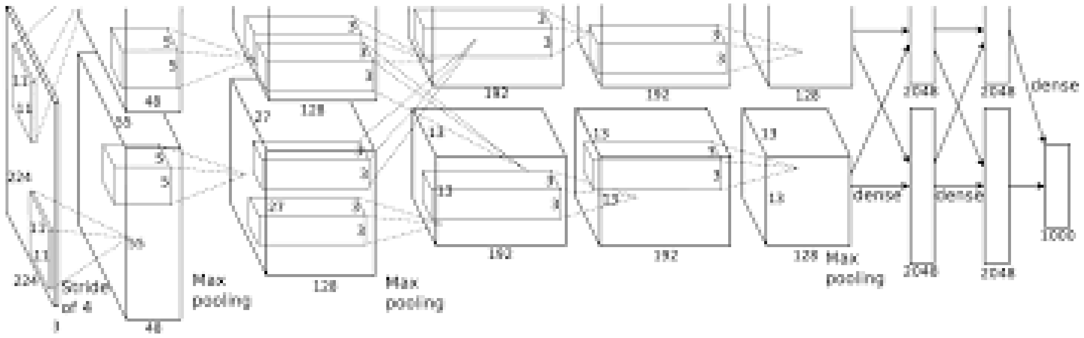


http://yann.lecun.com/

CNN: Its History – 2012

- Alex Krizhevsky et al, "*ImageNet Classification with Deep Convolutional Neural Networks*" ('12)
  - Win the imageNet competition: annual Olympics of computer vision with astounding results compared to previously existing approaches (26% to 15%).

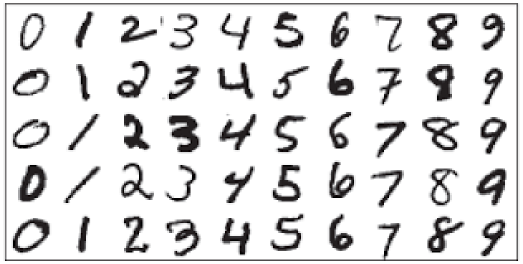




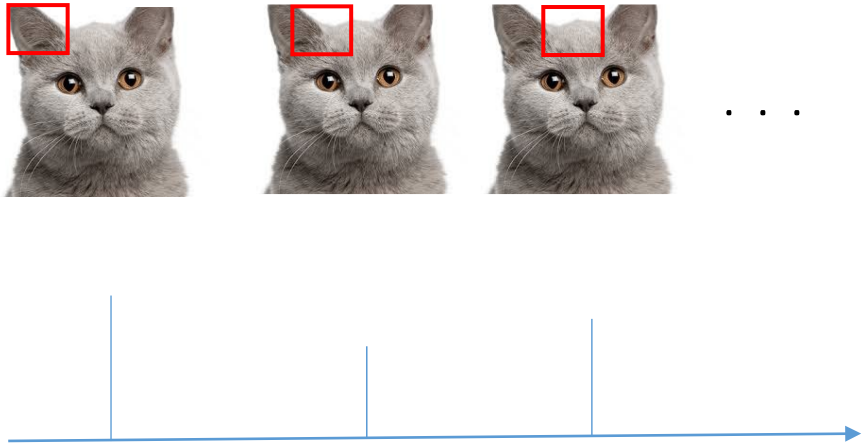
https://medium.com/@Lidinwise/the-revolution-of-depth-facf174924f5

CNN: Why Convolutional Neural Networks?

- Invariance to rotation, transformation, size, etc.

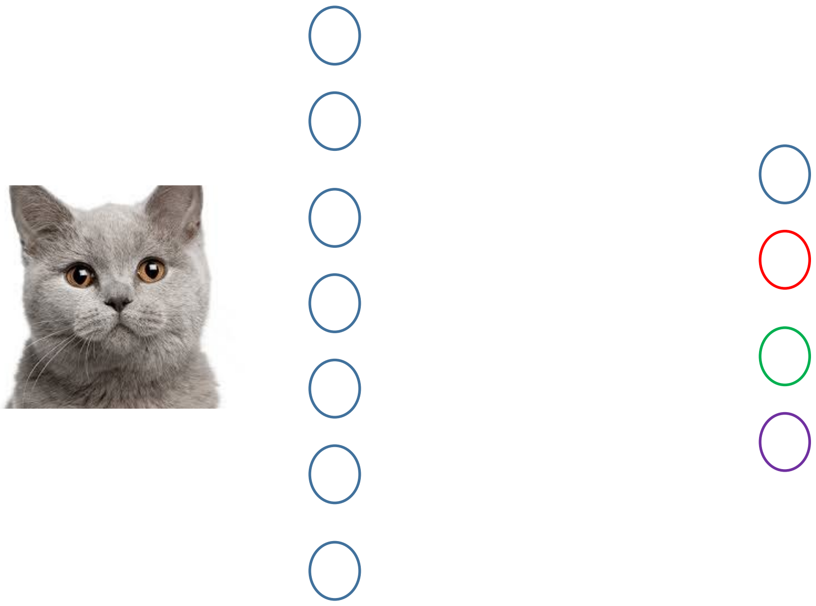


- Three architectural ideas of CNN
  1. Local receptive fields
  2. Weight sharing
  3. Spatial or temporal subsampling

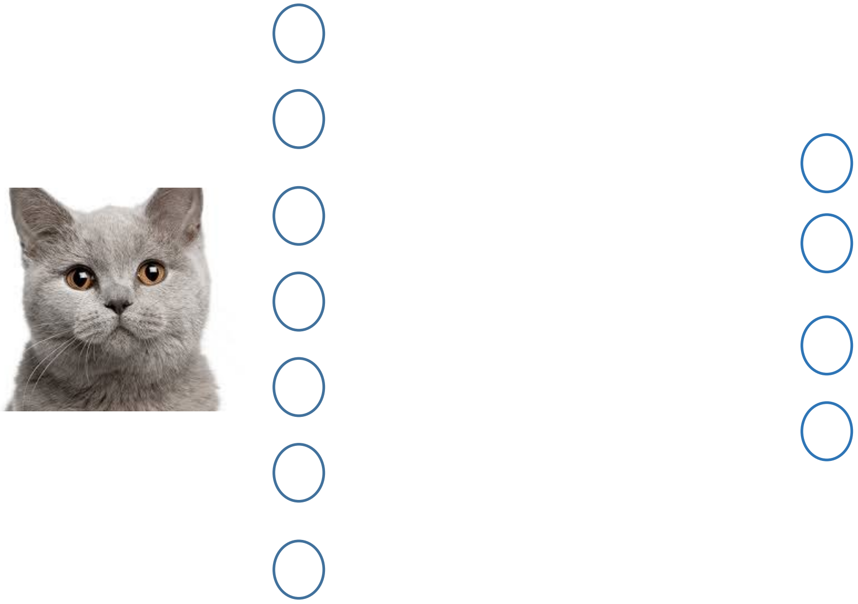


CNN: Its operational idea

- A node takes responsibility for a portion of input data.



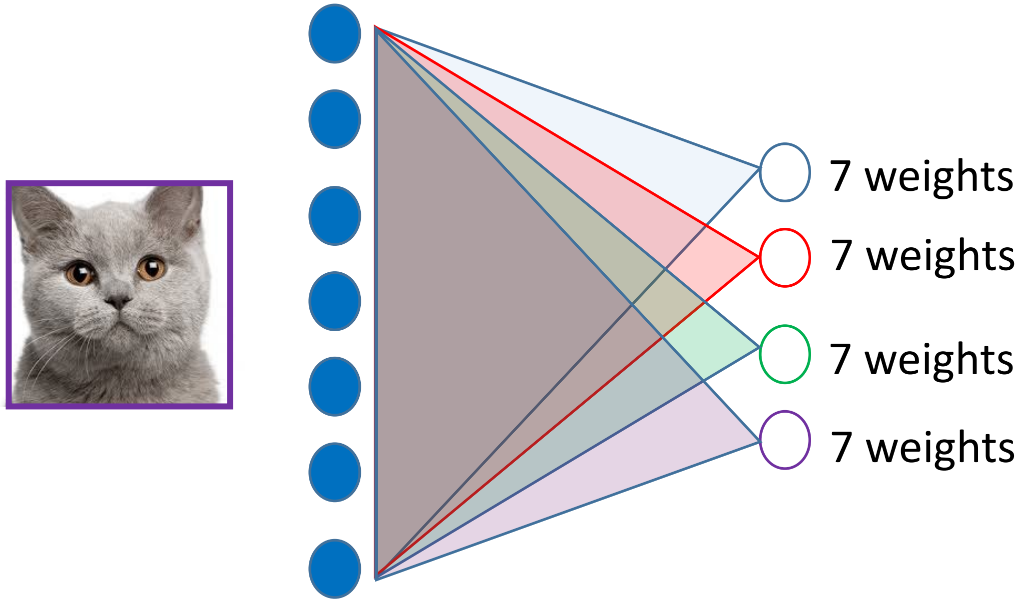
Fully Connected Layer



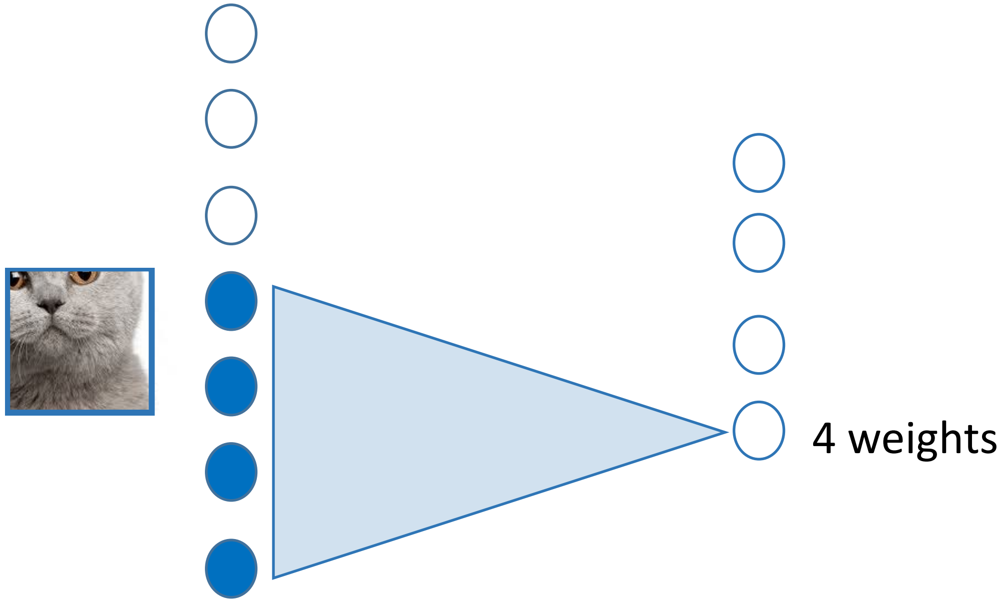
Convolution Layer

CNN: Its operational idea

- A node takes responsibility for a portion of input data.

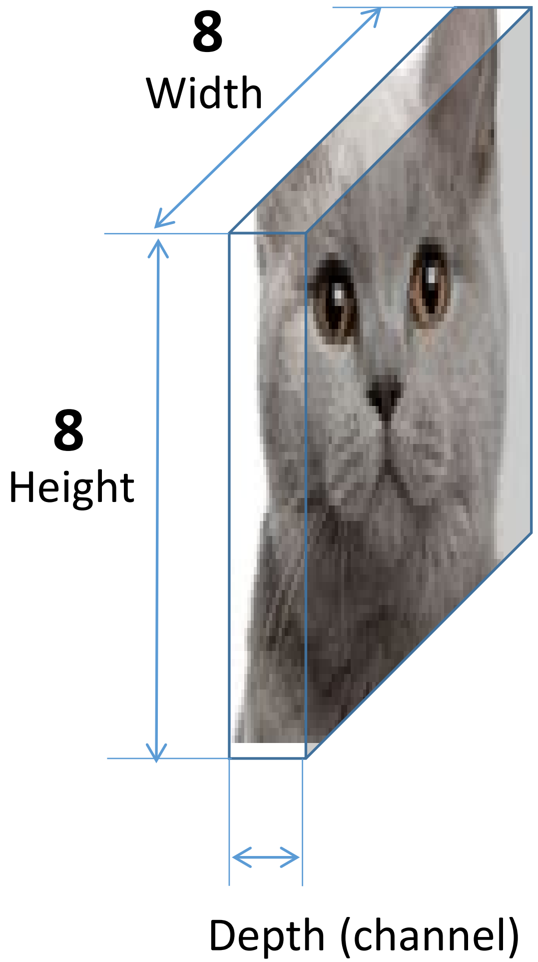


Fully connected Layer
- 7 x 4 = 28 weights

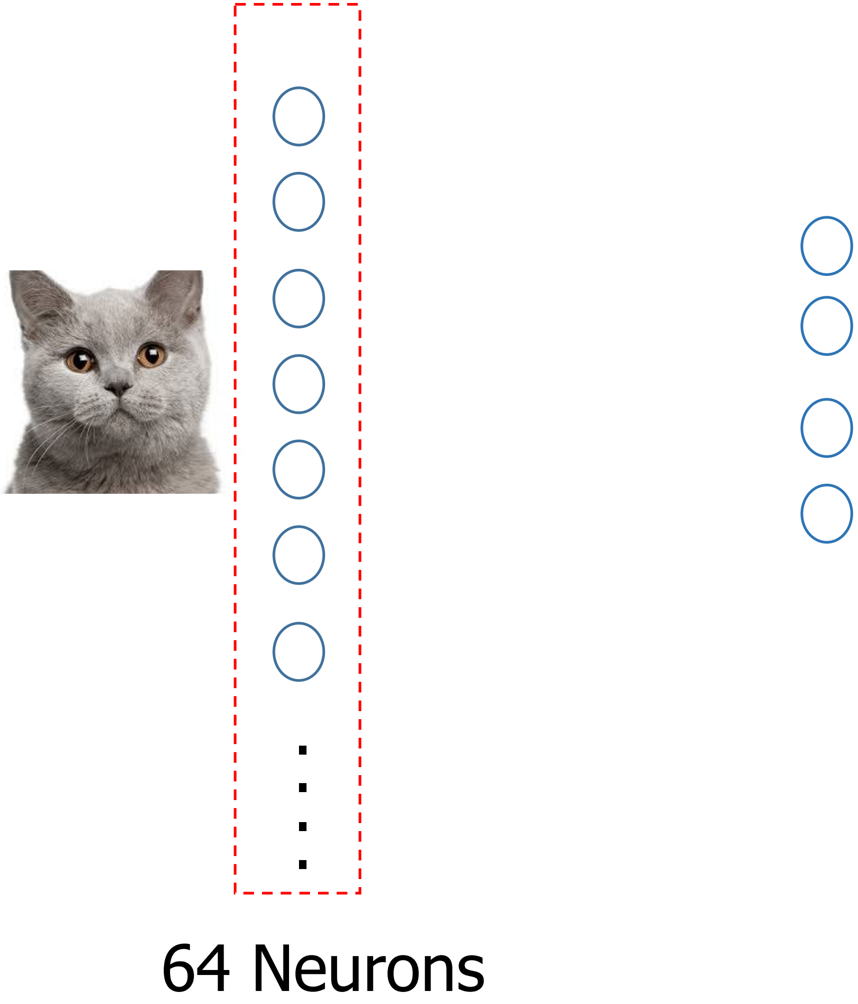


Convolutional Layer
- 4 weights

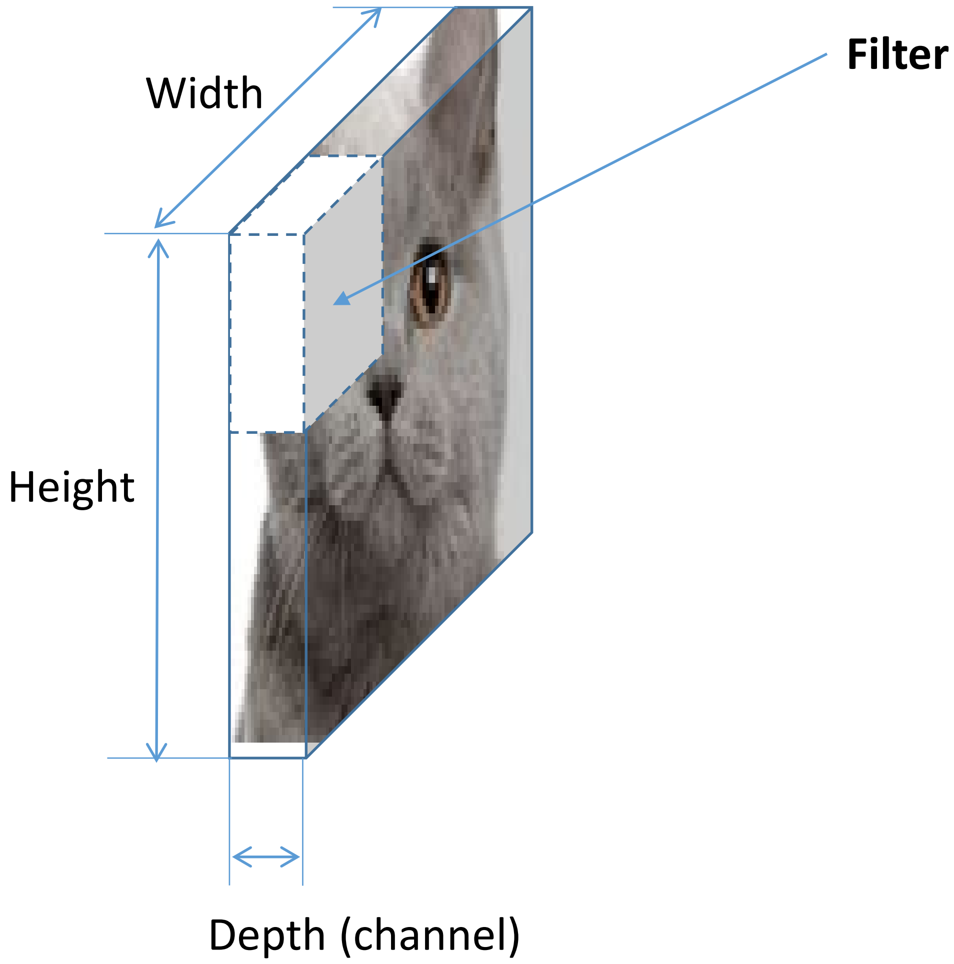
CNN: 2-D representation vs 1-D representation

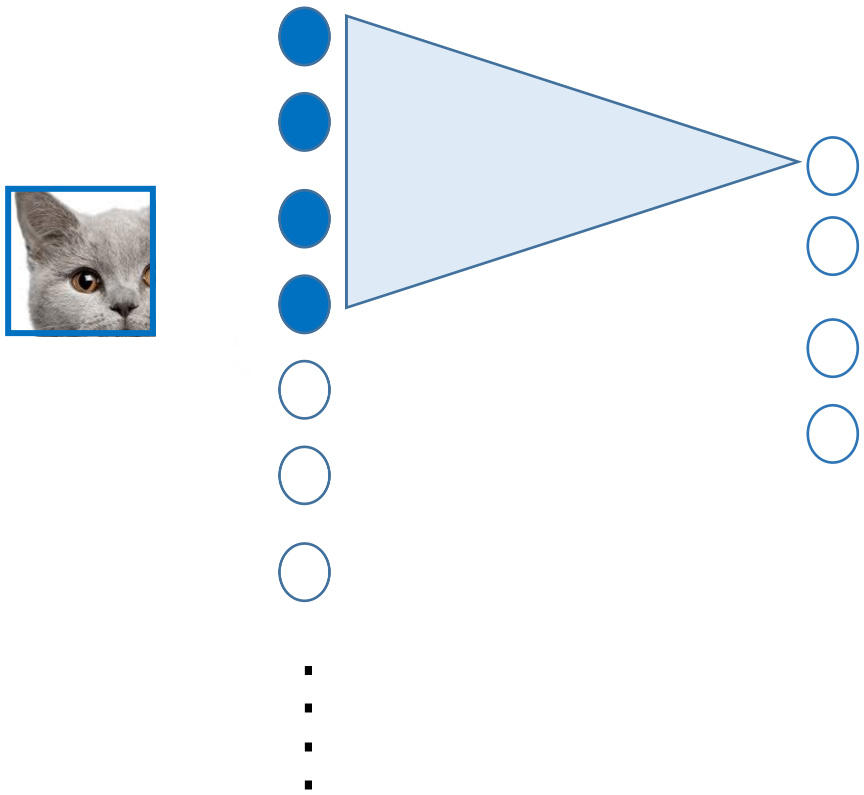


64 PIXEL image

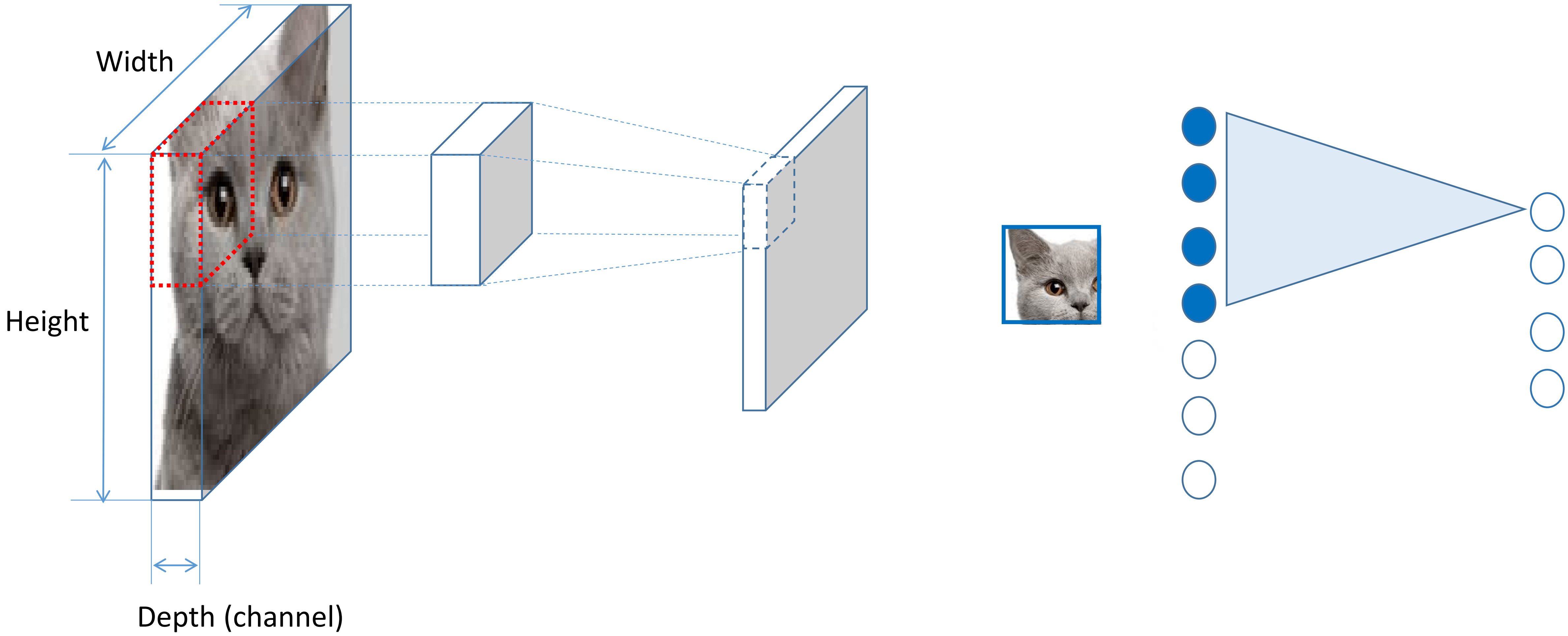


CNN: 2-D representation vs 1-D representation

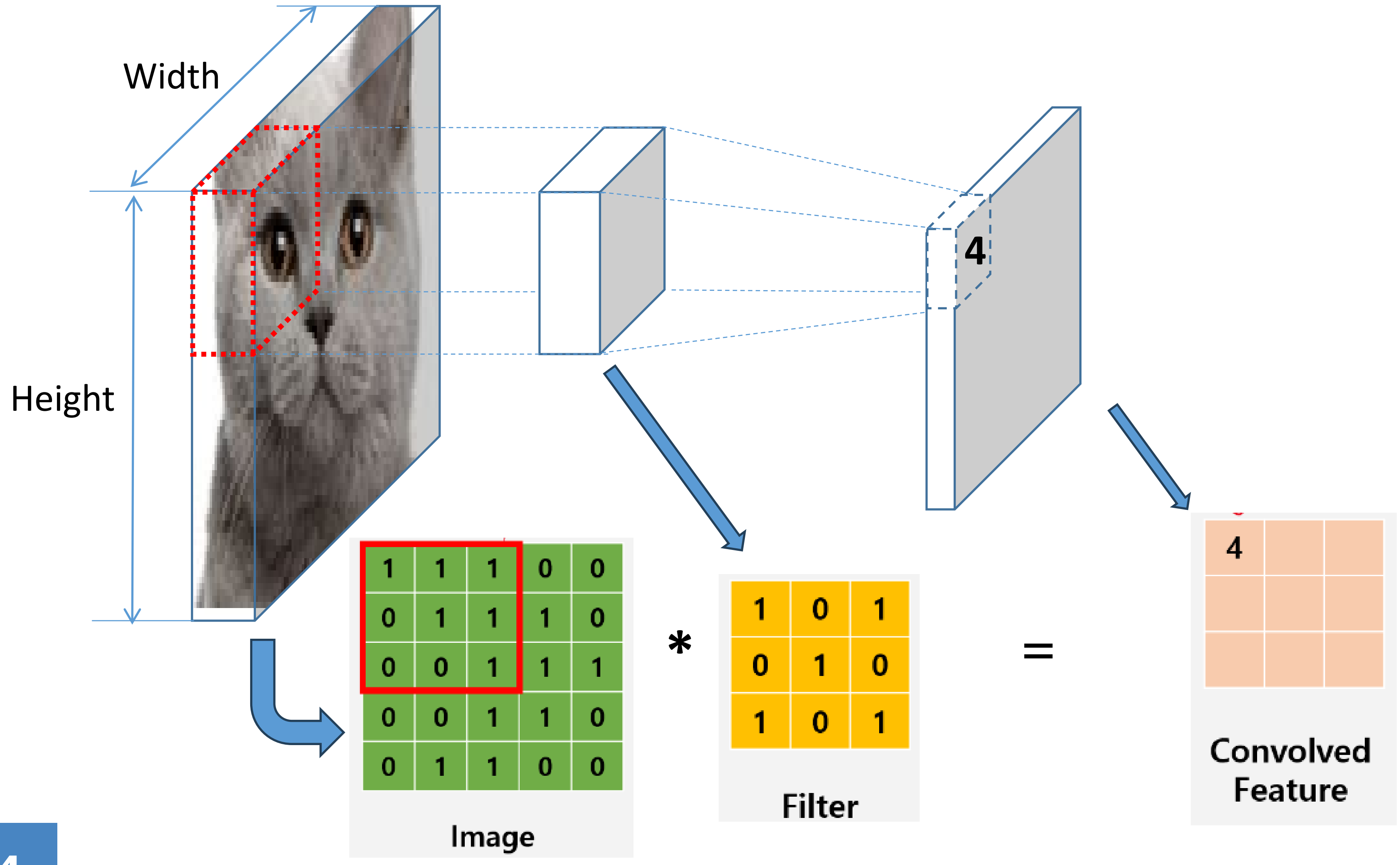




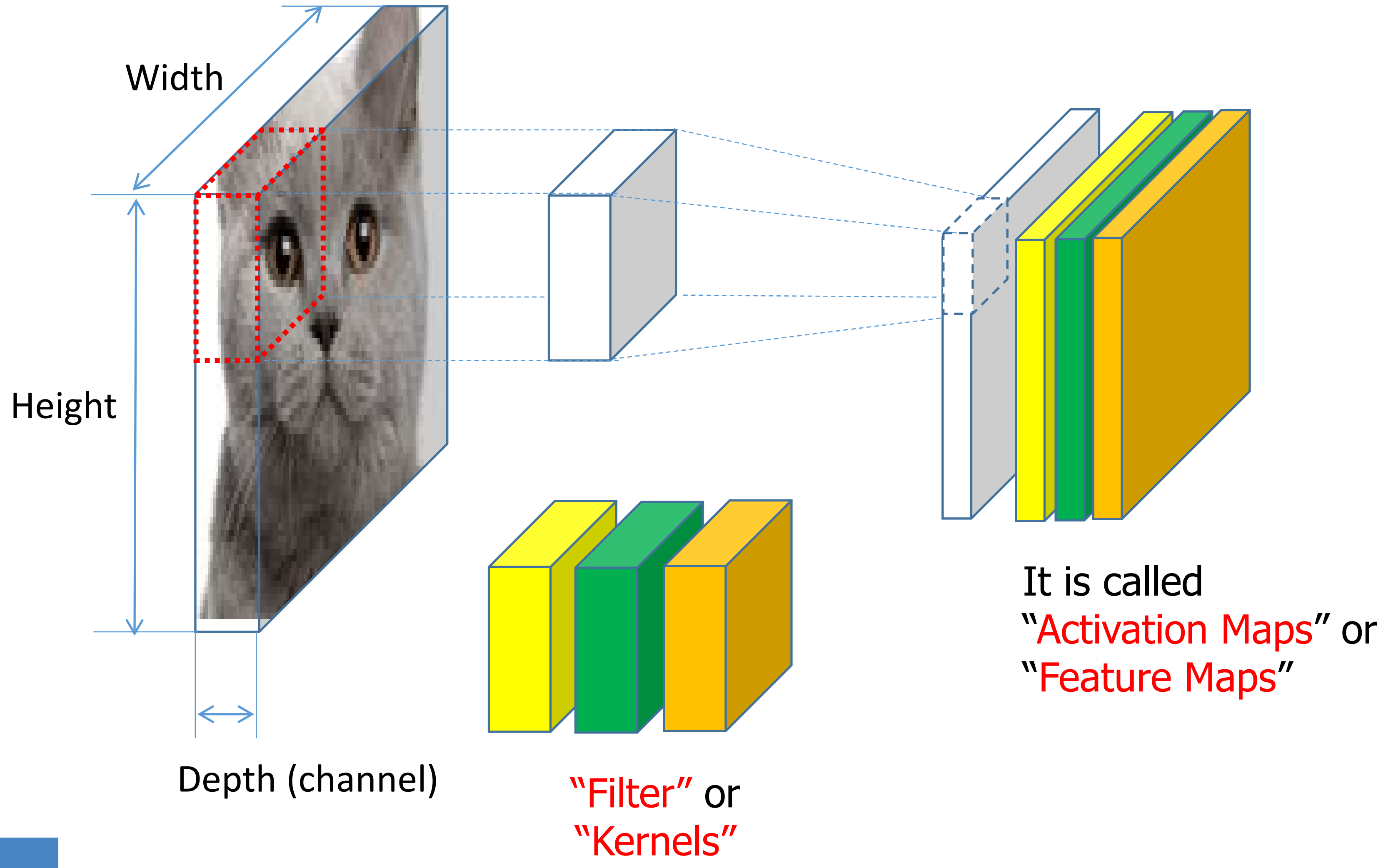
CNN: 2-D representation vs 1-D representation



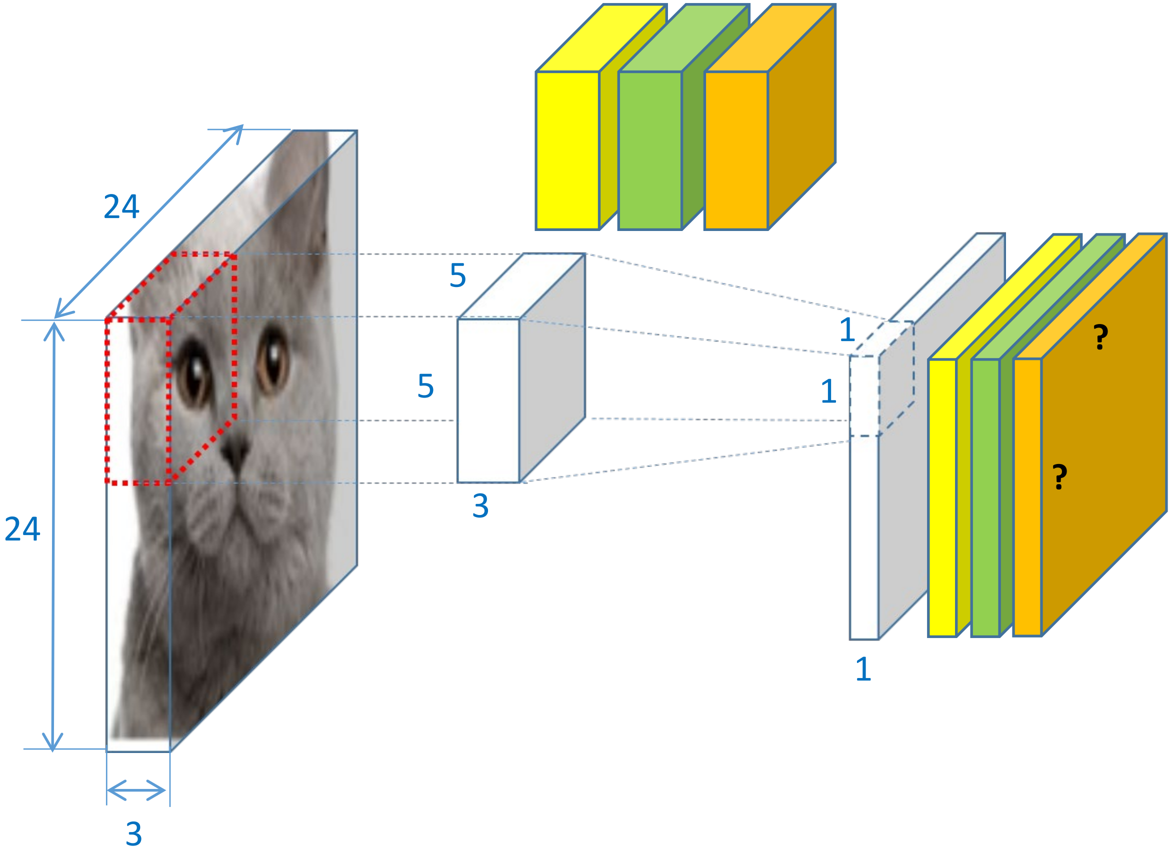
CNN: 2-D representation vs 1-D representation



CNN: How does it work?



CNN: How does it work?

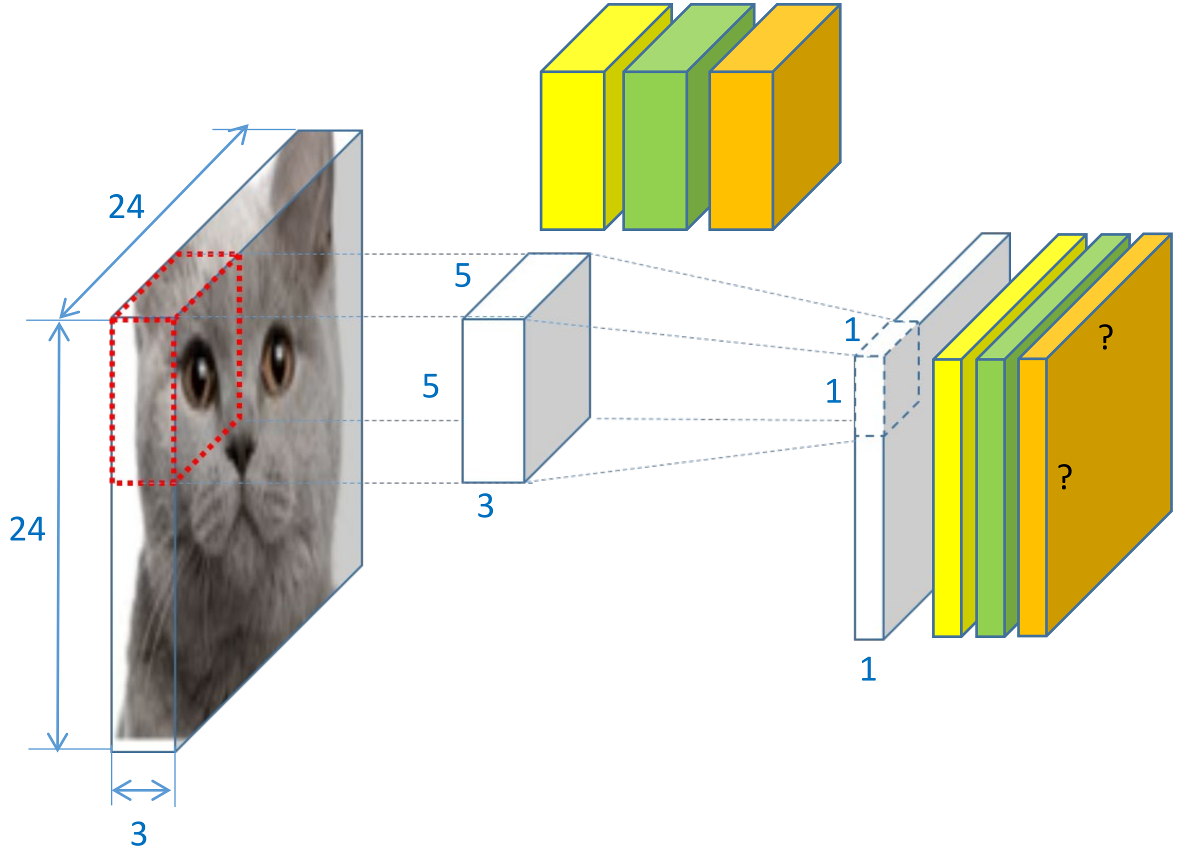


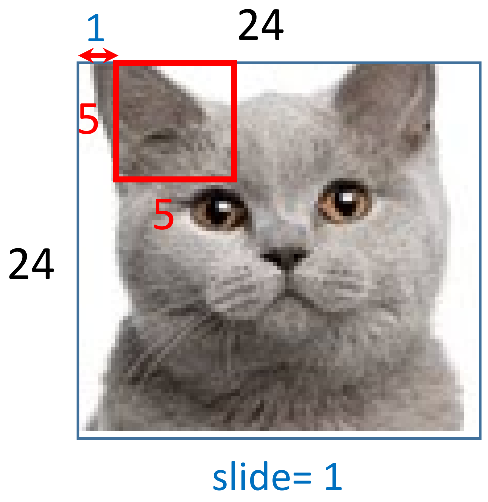
Input layer

Filters
Kernels

Activation maps
Feature maps

CNN: How does it work?



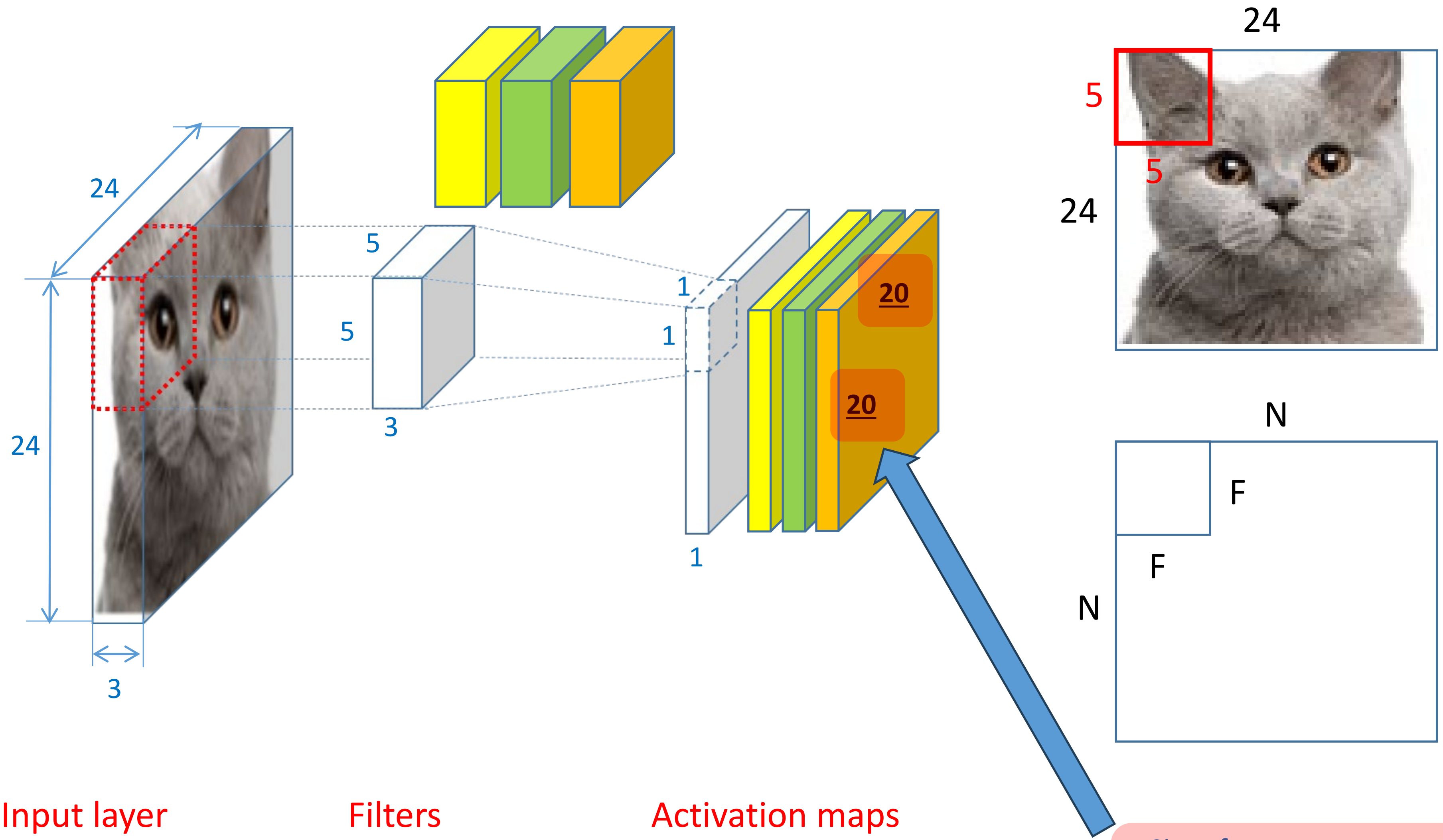


Input layer

Filters
Kernels

Activation maps
Feature maps

CNN: How does it work?



Input layer

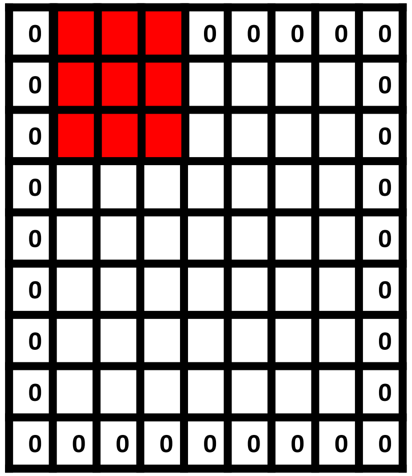
Filters
Kernels

Activation maps
Feature maps

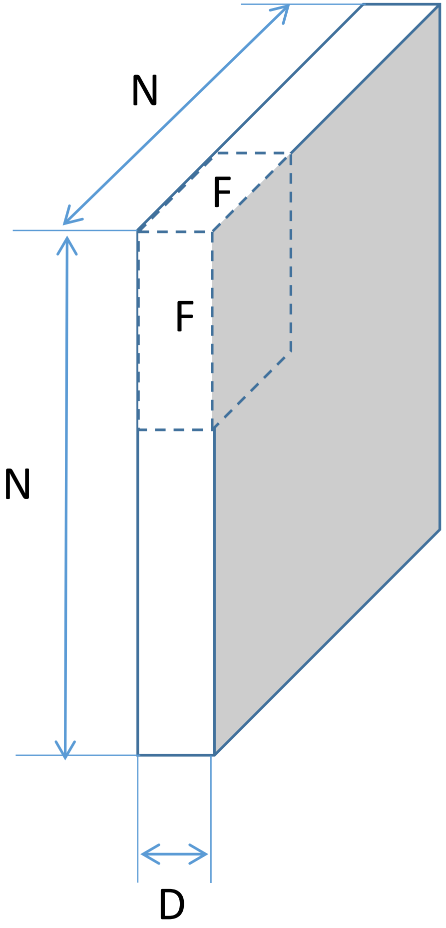
Size of activation map $= \frac{(N-F)}{stride} + 1 = \frac{(24-5)}{1} + 1 = 20$

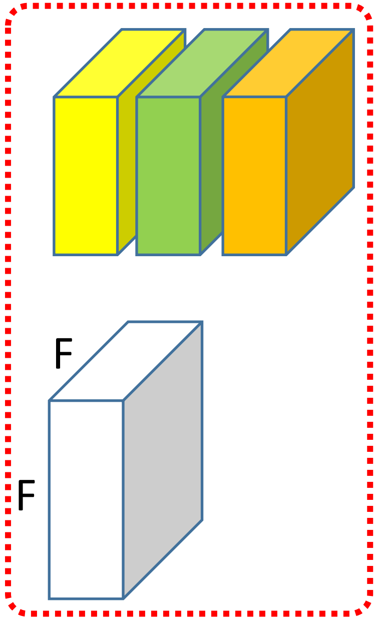
CNN: How does it work?

Hyper parameters	Symbols
Number of filters	K
Size of the filter	F
Stride	S
Padding	P

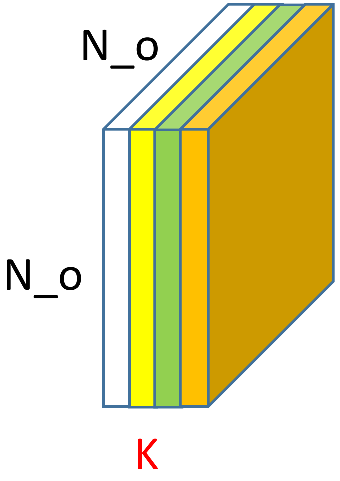


- Padding: P=1
- Stride: S=1
- activation map becomes is 7 x 7 matrix





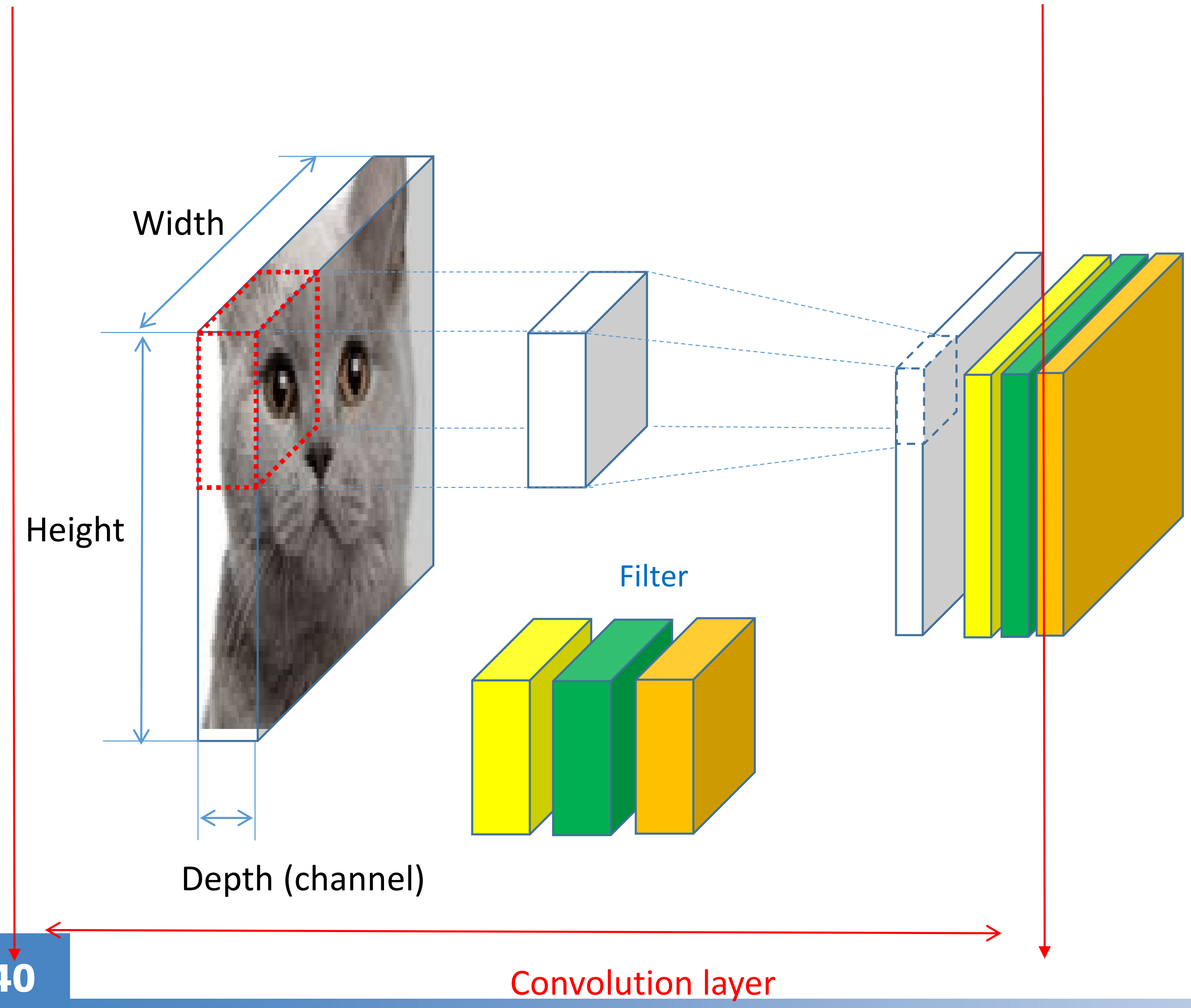
Number of filters: K



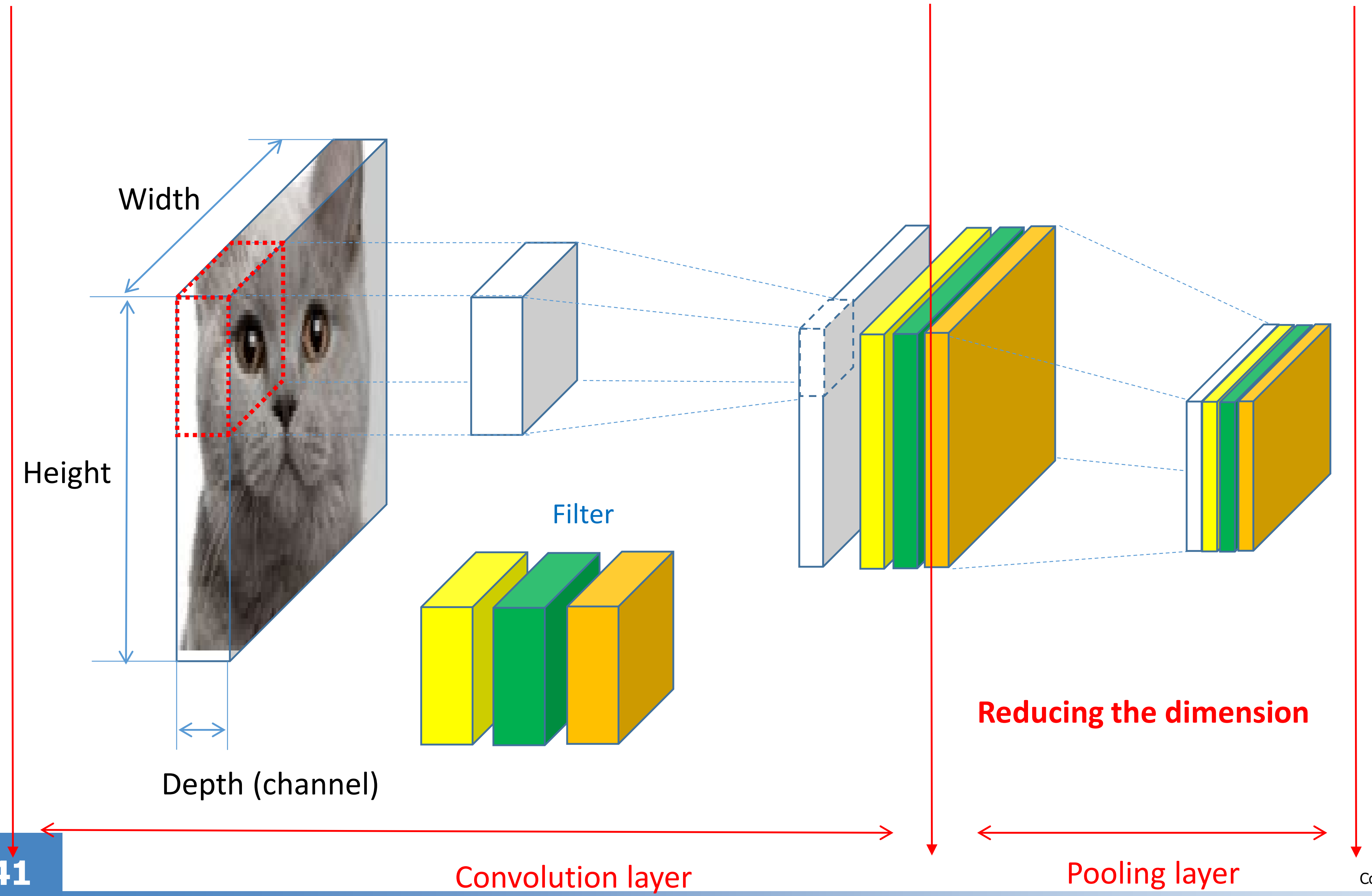
$$N\_o = \frac{(N - F + 2P)}{S} + 1$$

- Padding aims to maintain the original dimension of the original data.

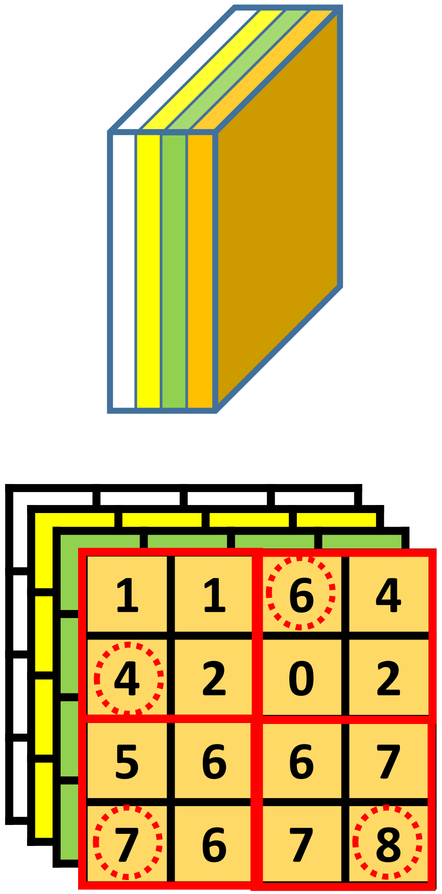
CNN: How does it work?

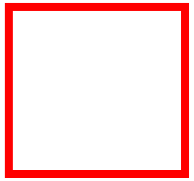


CNN: How does it work?



- ❑ This layer aims to reduce the dimension



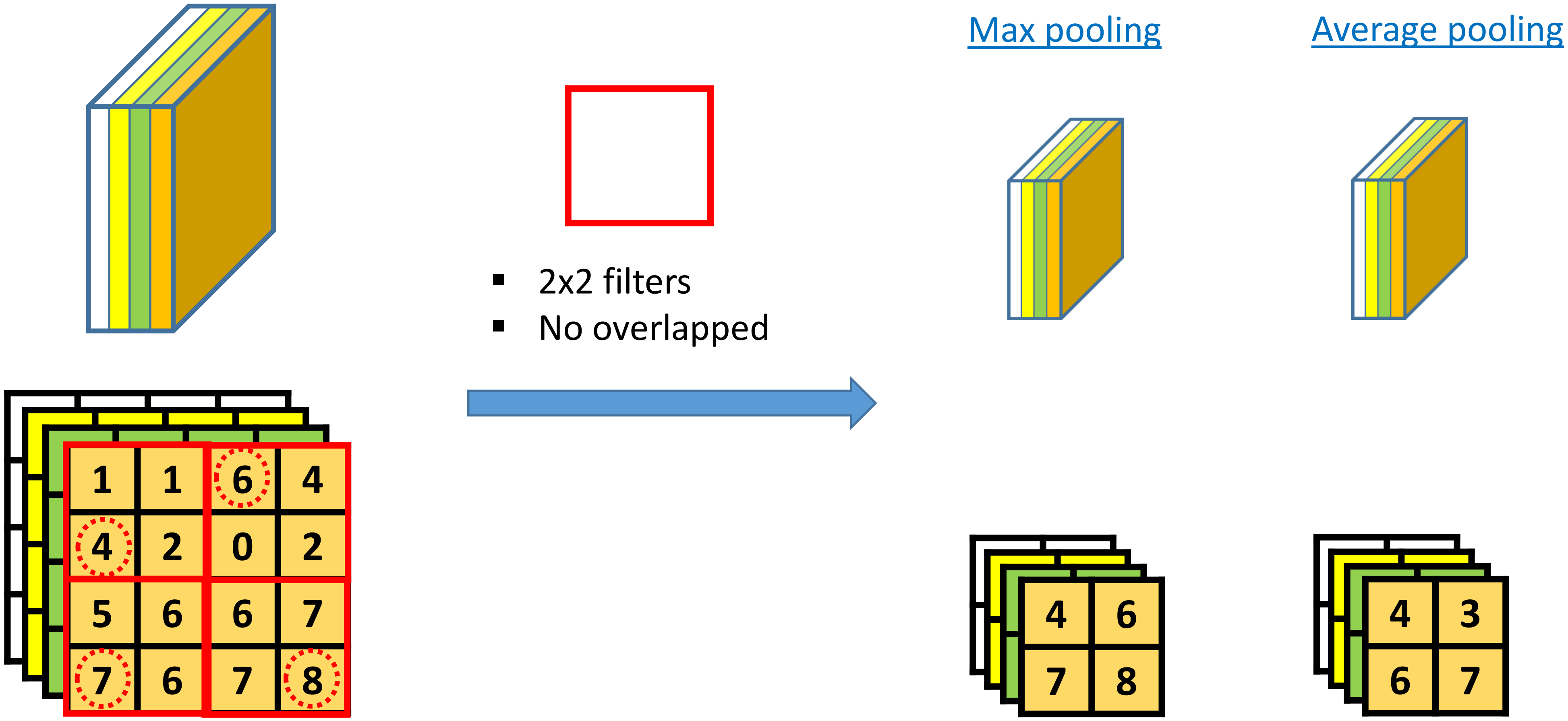


- 2x2 filters
- No overlapped

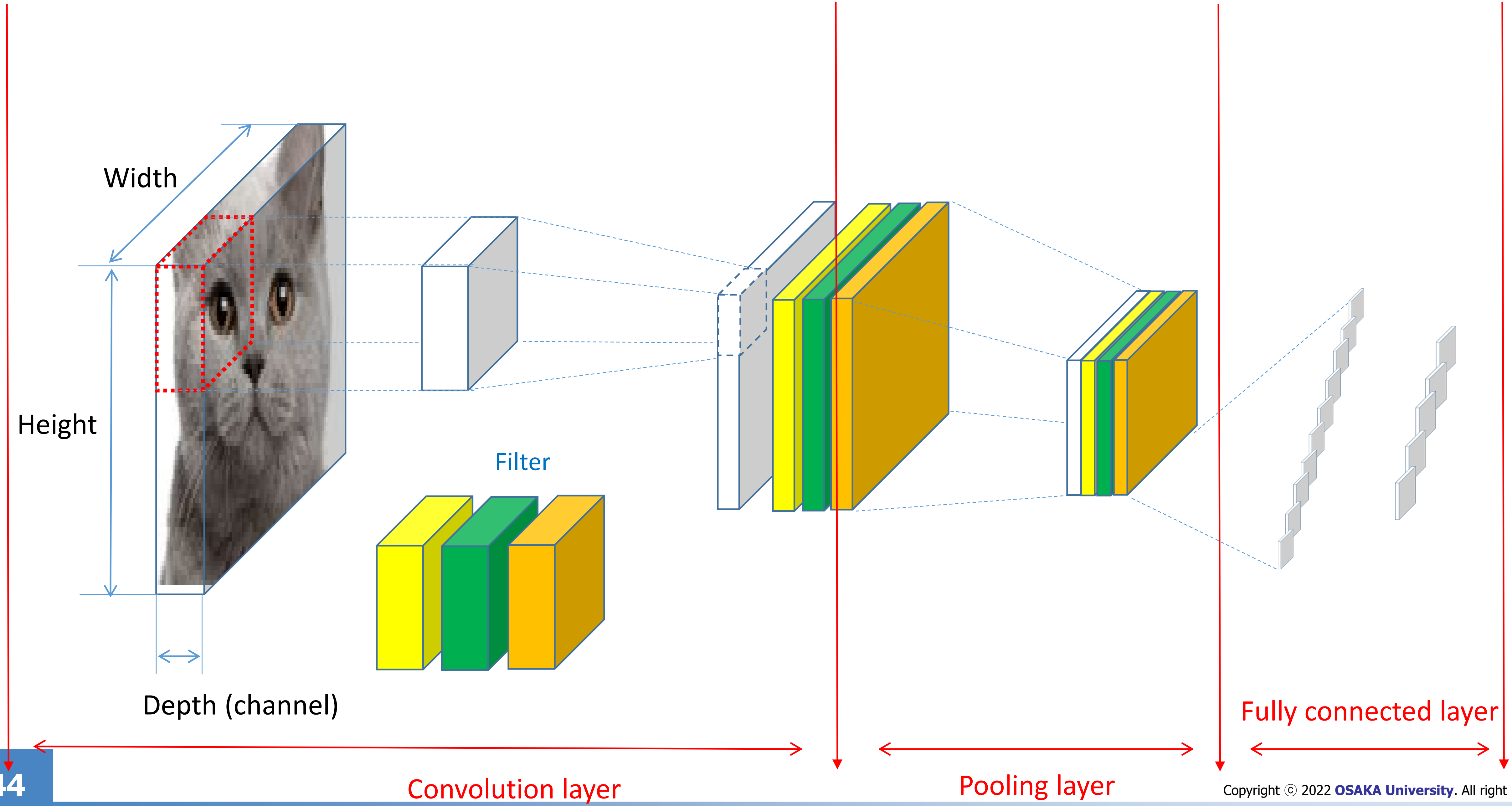
Max pooling

CNN: How does it work?

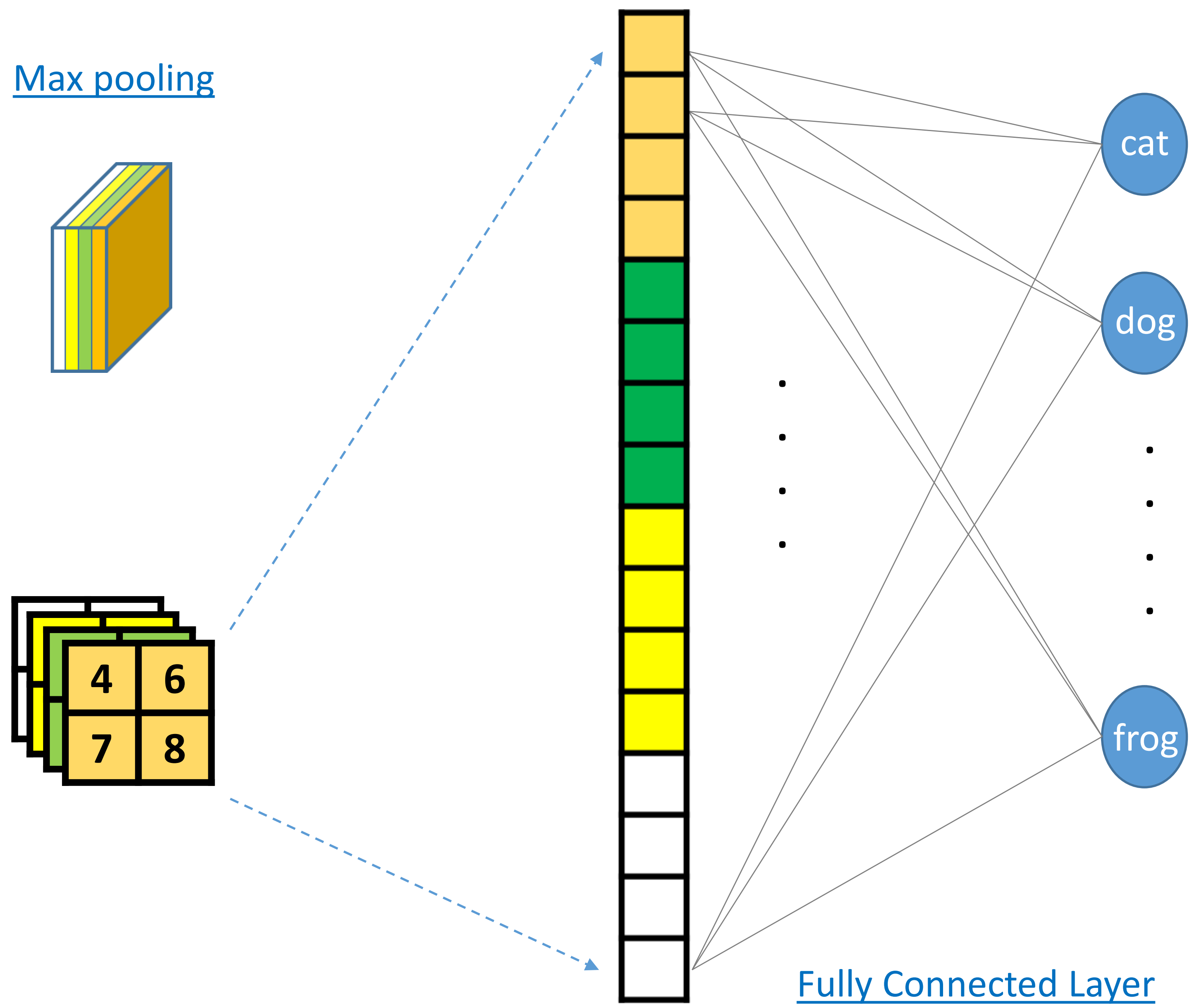
❑ This layer aims to reduce the dimension



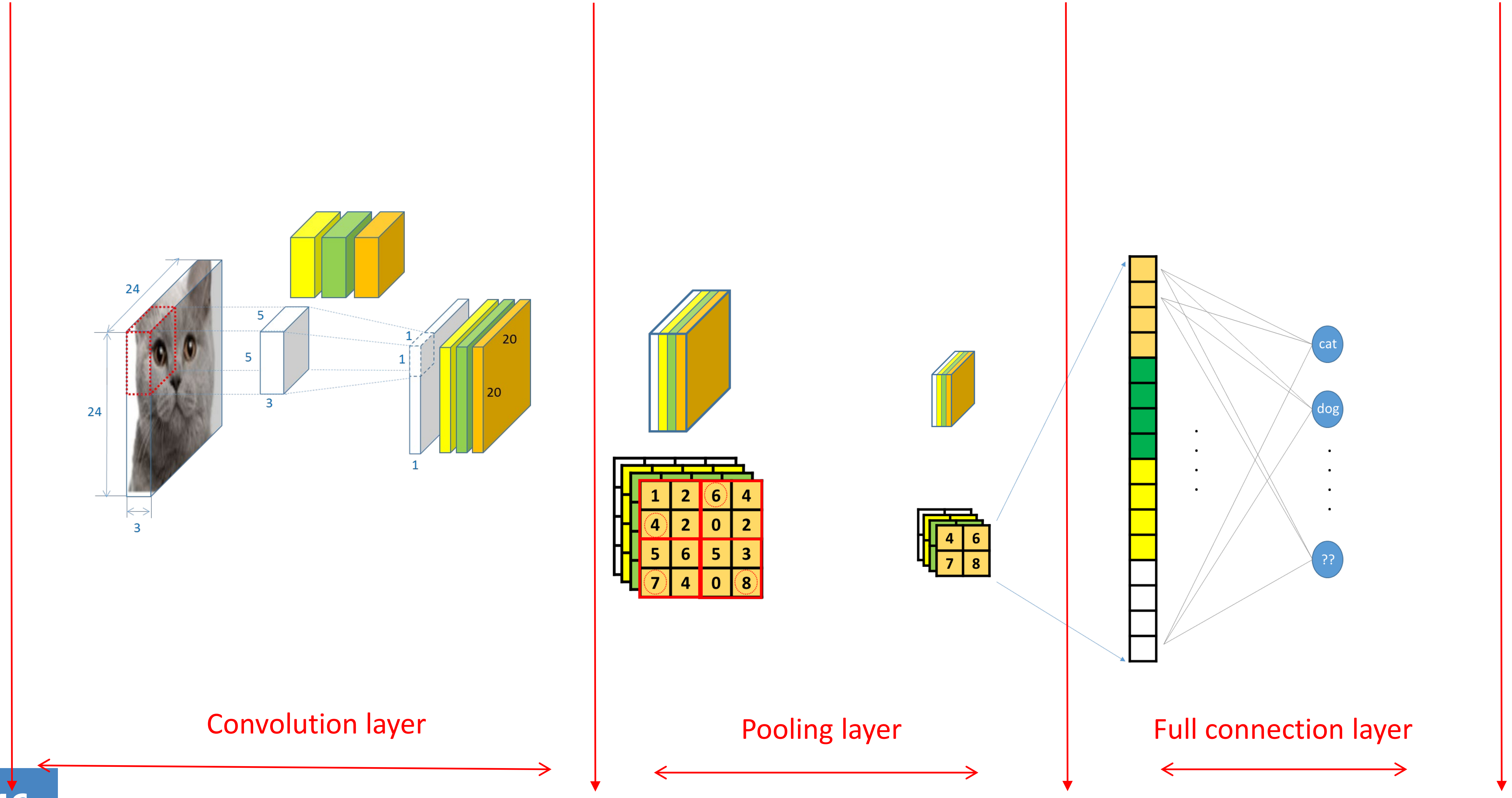
CNN: How does it work?



CNN: How does it work?



Summary of CNN operation

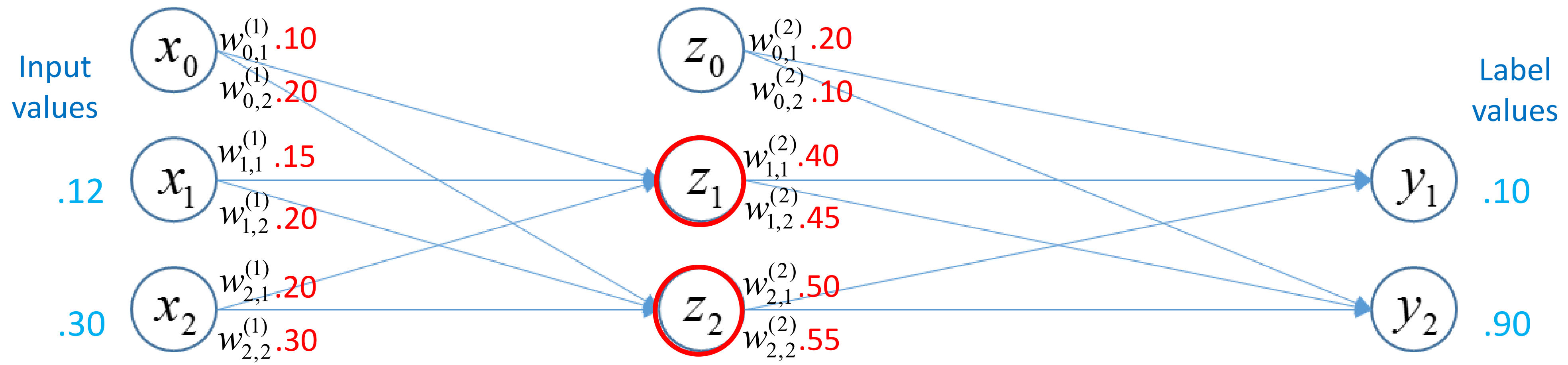


Summary

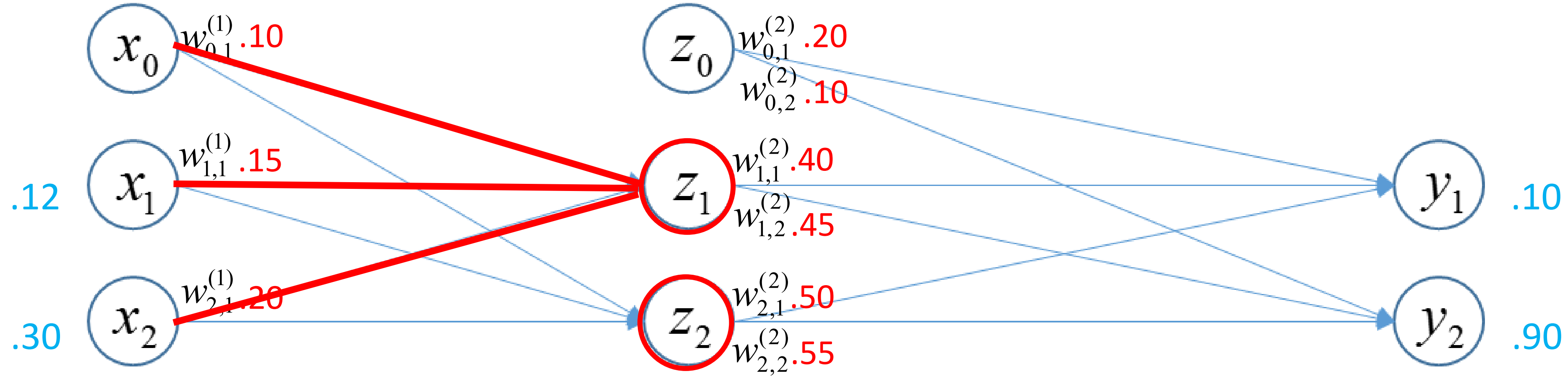
- A deep neural network is a class of neural networks inspired by the human brain's structure and functioning.
- We call a deep neural network as modern style machine learning because it can be operable now due to the abundant data, and powerful machines, etc.
- The backpropagation algorithm of neural networks was explained.
- Several design issues of neural networks such as activation functions, initial link weight setting, were explored.
- One of widely used deep neural networks called Convolutional Neural Network (CNN) was introduced.

Backup slides

Backpropagation algorithm - Forwarding



Backpropagation algorithm - Forwarding

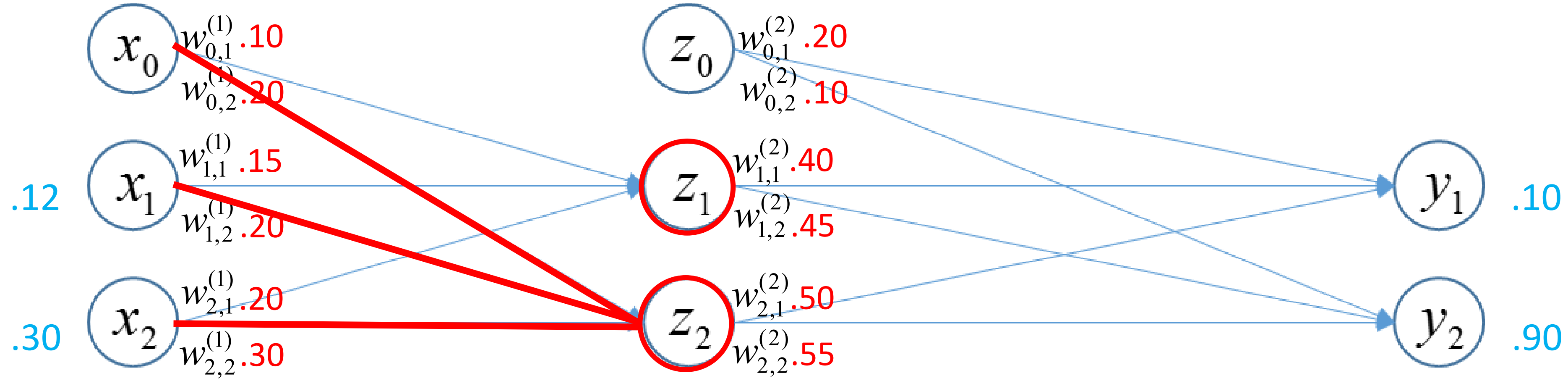


$$a_1 = w_{1,1}^{(1)} x_1 + w_{2,1}^{(1)} x_2 + w_{0,1}^{(1)}$$

$$a_1 = 0.15 \times 0.12 + 0.2 \times 0.3 + 0.1$$

$$a_1 = 0.178$$

Backpropagation algorithm - Forwarding



$$a_1 = w_{1,1}^{(1)} x_1 + w_{2,1}^{(1)} x_2 + w_{0,1}^{(1)}$$

$$a_1 = 0.15 \times 0.12 + 0.2 \times 0.3 + 0.1$$

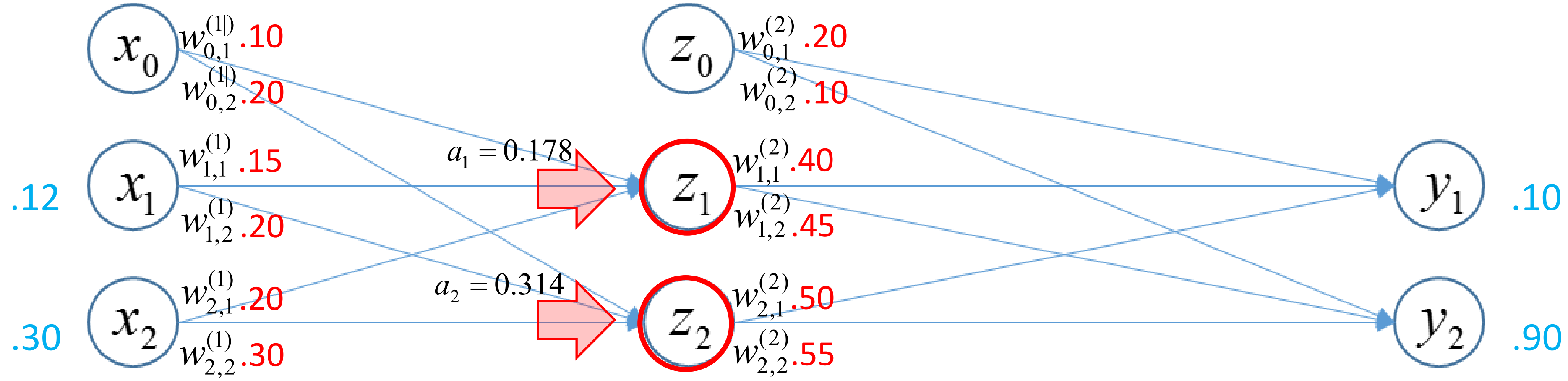
$$a_1 = 0.178$$

$$a_2 = w_{1,2}^{(1)} x_1 + w_{2,2}^{(1)} x_2 + w_{0,2}^{(1)}$$

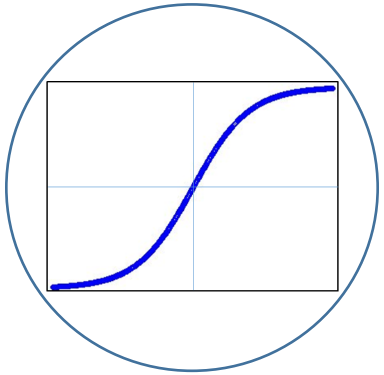
$$a_2 = 0.2 \times 0.12 + 0.3 \times 0.3 + 0.2$$

$$a_1 = 0.314$$

Backpropagation algorithm - Forwarding



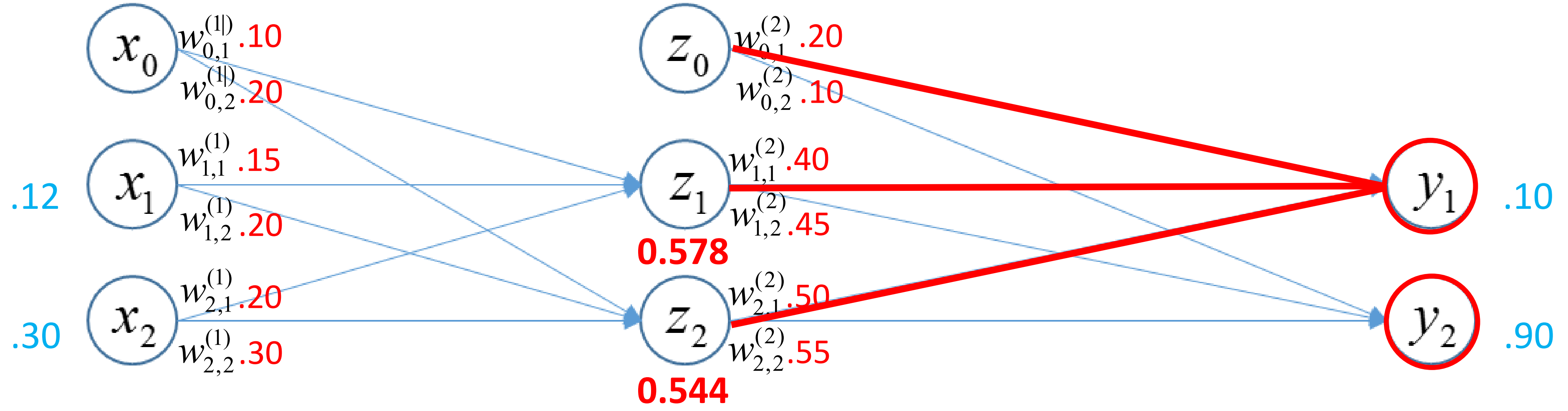
$$a_1 = 0.178$$
$$a_2 = 0.314$$



$$\sigma(a_1) = 0.544$$
$$\sigma(a_2) = 0.578$$

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Backpropagation algorithm - Forwarding

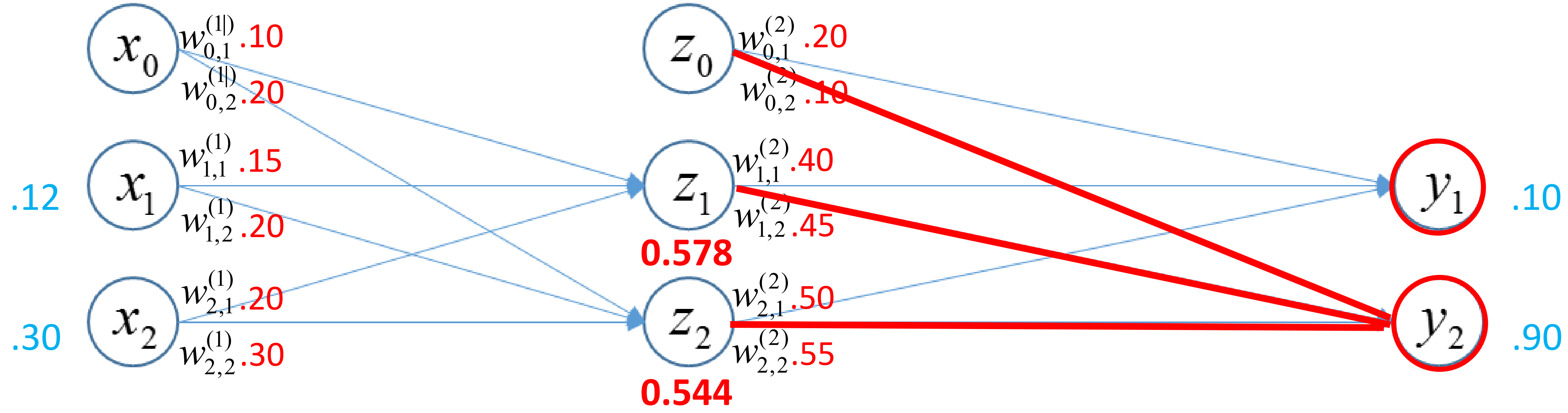


$$a_1 = w_{1,1}^{(2)} z_1 + w_{2,1}^{(2)} z_2 + w_{0,1}^{(2)}$$

$$a_1 = 0.40 \times 0.578 + 0.5 \times 0.544 + 0.2$$

$$a_1 = 0.703$$

Backpropagation algorithm - Forwarding



$$a_1 = w_{1,1}^{(2)} z_1 + w_{2,1}^{(2)} z_2 + w_{0,1}^{(2)}$$

$$a_1 = 0.40 \times 0.578 + 0.5 \times 0.544 + 0.2$$

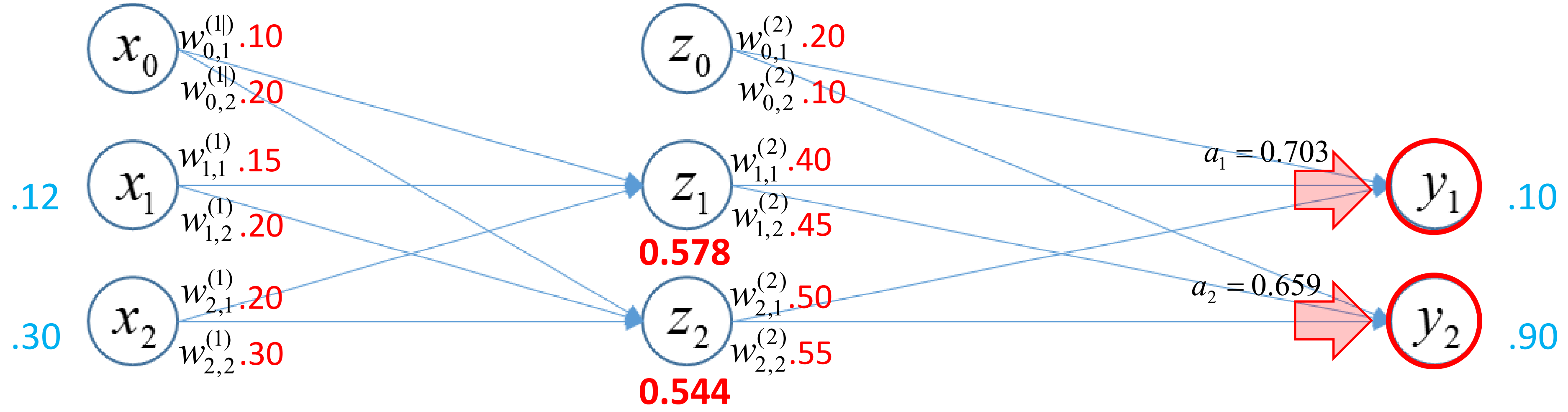
$$a_1 = 0.703$$

$$a_2 = w_{1,2}^{(2)} z_1 + w_{2,2}^{(2)} z_2 + w_{0,2}^{(2)}$$

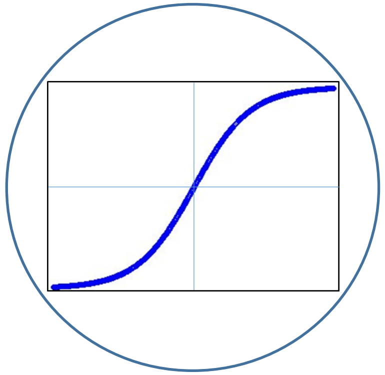
$$a_2 = 0.45 \times 0.578 + 0.55 \times 0.544 + 0.1$$

$$a_2 = 0.659$$

Backpropagation algorithm - Forwarding



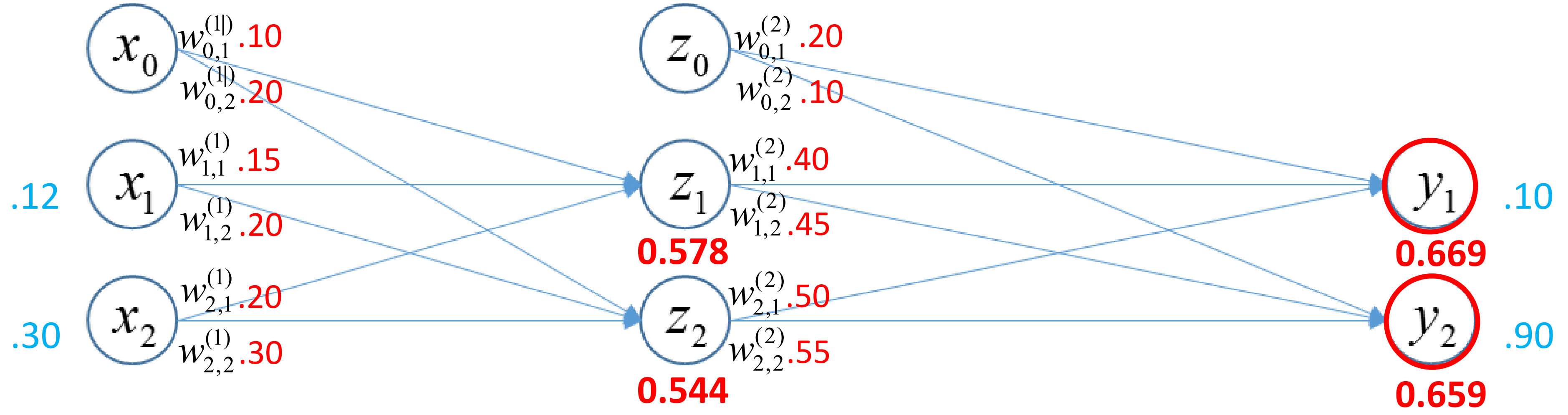
$$a_1 = 0.703$$
$$a_2 = 0.659$$



$$\sigma(a_1) = 0.669$$
$$\sigma(\alpha_2) = 0.659$$

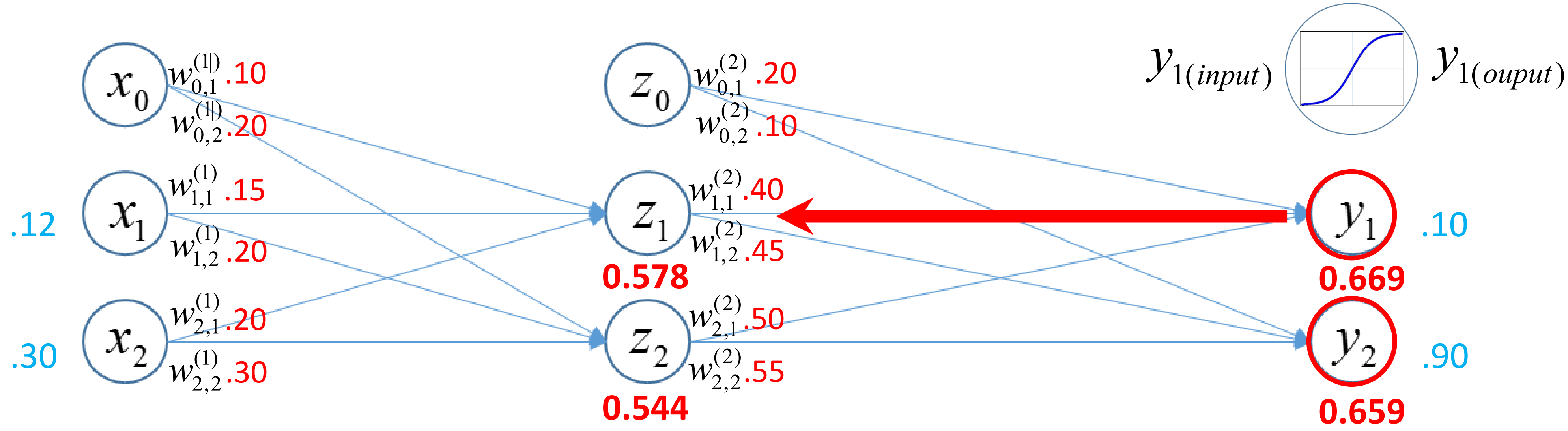
$$\sigma(a) = \frac{1}{1+e^{-a}}$$

Backpropagation algorithm - Forwarding



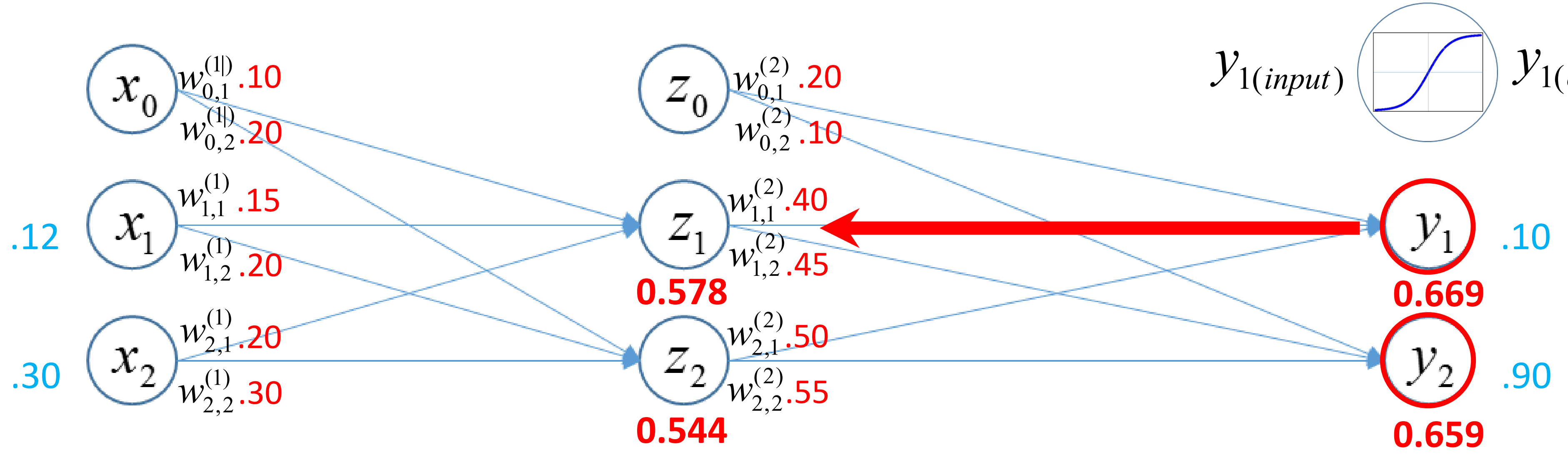
$$E(\mathrm{w}) = \frac{1}{2}\left((0.1 - 0.669)^2 + (0.9 - 0.659)^2\right) = 0.191$$

Backpropagation algorithm - Backwarding



$$\frac{\partial y_{1(input)}}{\partial w_{1,1}^{(2)}} \times \frac{\partial y_{1(ouput)}}{\partial y_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial y_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(2)}}$$

Backpropagation algorithm - Backwarding



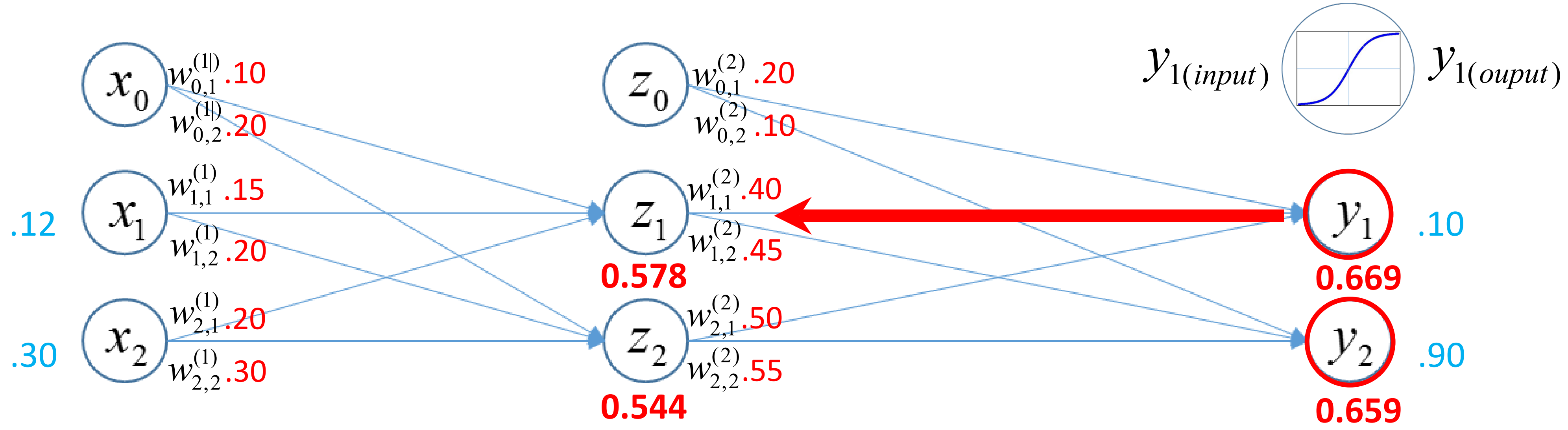
$y_{1(input)}$ $y_{1(ouput)}$

$$\frac{\partial y_{1(input)}}{\partial w_{1,1}^{(2)}} \times \frac{\partial y_{1(ouput)}}{\partial y_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial y_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(2)}}$$

0.569

$$E(\mathrm{w}) = \frac{1}{2}\left((0.1 - y_{1(output)})^2 + (0.9 - y_{2(output)})^2\right)$$

$$\frac{\partial E(\mathrm{w})}{\partial y_{1(output)}} = -(0.1 - y_{1(output)}) = -(0.1 - 0.669) = 0.569$$



$$y_{1(ouput)} = \frac{1}{1+e^{-y_{1(input)}}}$$

$$\frac{\partial y_{1(input)}}{\partial w_{1,1}^{(2)}} \times \frac{\partial y_{1(ouput)}}{\partial y_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial y_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(2)}}$$

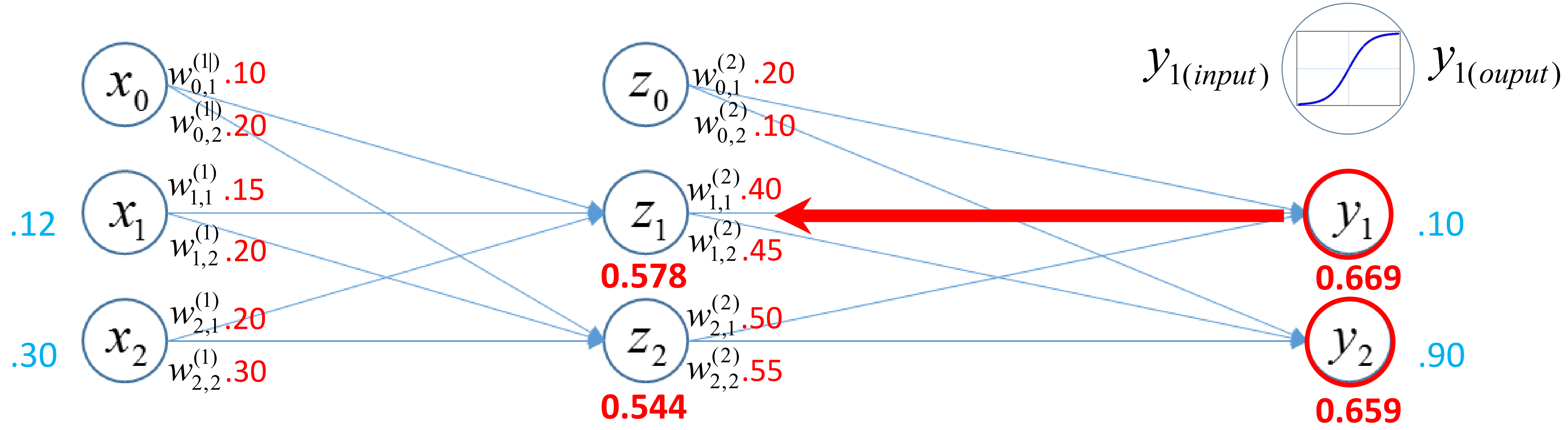
0.221 **0.569**

$$\frac{\partial y_{1(output)}}{\partial y_{1(input)}} = \sigma(y_{1(input)})\left(1-\sigma(y_{1(input)})\right)$$

$$= y_{1(output)}(1-y_{1(output)})$$

$$= 0.669 \times (1-0.669) = 0.221$$

Backpropagation algorithm - Backwarding



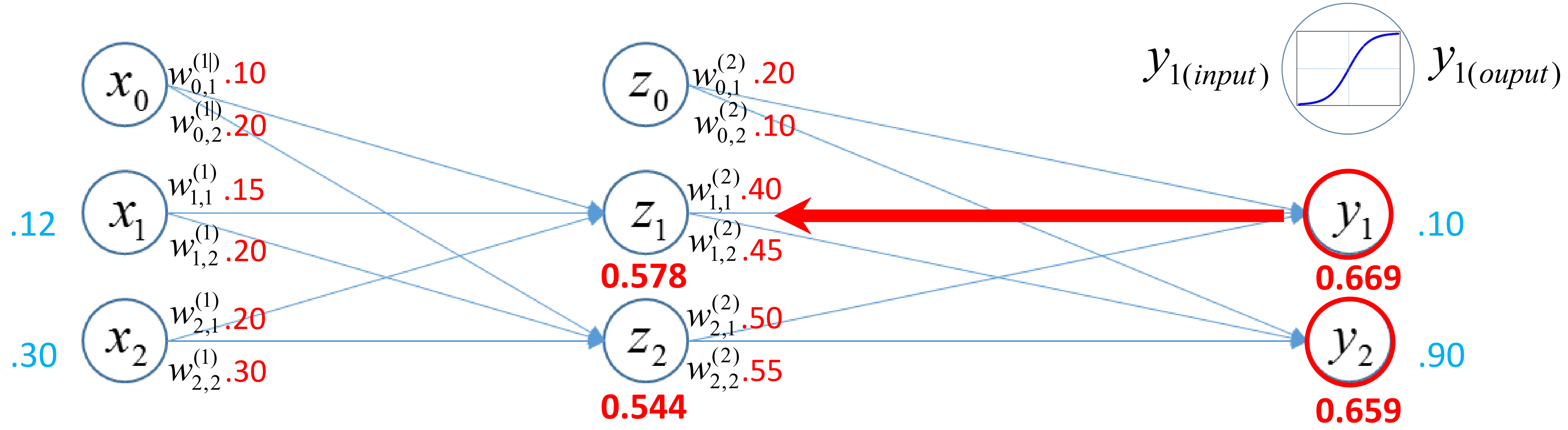
$$y_{1(input)} = w_{1,1}^{(2)} z_1 + w_{2,1}^{(2)} z_2 + w_{0,1}^{(2)}$$

$$\frac{\partial y_{1(input)}}{\partial w_{1,1}^{(2)}} = z_1 = 0.578$$

$$\frac{\partial y_{1(input)}}{\partial w_{1,1}^{(2)}} \times \frac{\partial y_{1(ouput)}}{\partial y_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial y_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(2)}}$$

0.578 **0.221** **0.569** **0.0727**

Backpropagation algorithm - Backwarding



$$w^{(2)}_{1,1}{}^{*} = w^{(2)}_{1,1} - \eta \frac{\partial E(\mathrm{w})}{\partial w^{(2)}_{1,1}}$$

Learning rate

$$= 0.4 - 0.5 \times 0.0727$$

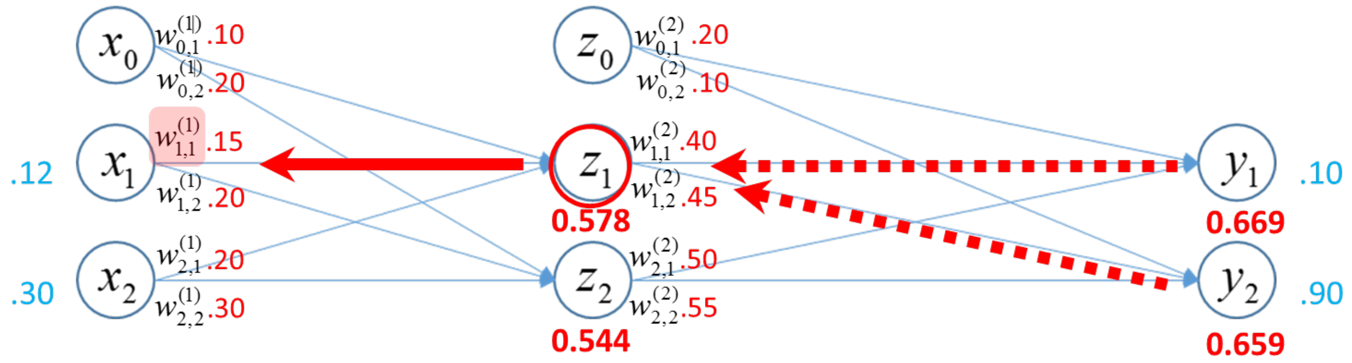
$$= \mathbf{0.36365}$$

$$\frac{\partial y_{1(input)}}{\partial w^{(2)}_{1,1}} \times \frac{\partial y_{1(ouput)}}{\partial y_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial y_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w^{(2)}_{1,1}}$$

0.578 **0.221** **0.569** **0.0727**

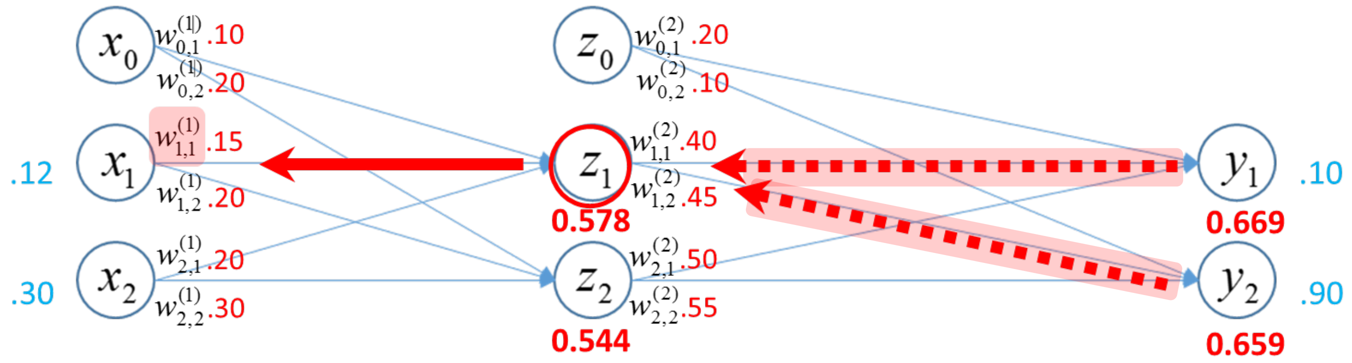
Same procedures are applied for $w^{(2)}_{1,2}, w^{(2)}_{2,1}, w^{(2)}_{2,2}$

Backpropagation algorithm - Backwarding



$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

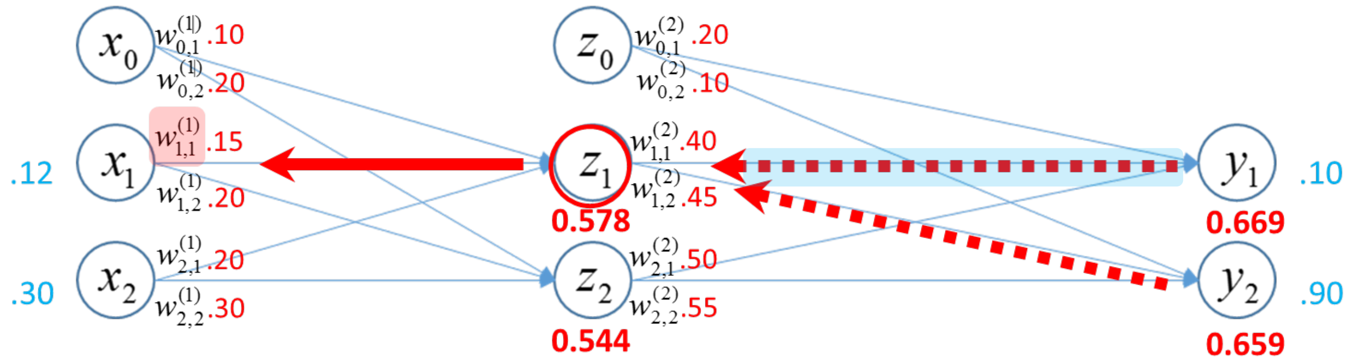
Backpropagation algorithm - Backwarding



$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

$$\frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} + \frac{\partial E(\mathrm{w})_{y_2}}{\partial z_{1(ouput)}}$$

Backpropagation algorithm - Backwarding



$$y_{1(input)} = w_{1,1}^{(2)} z_1 + w_{2,1}^{(2)} z_2 + w_{0,1}^{(2)}$$

$$\frac{\partial y_{1(input)}}{\partial z_{1(output)}} = w_{1,1}^{(2)} = 0.4$$

$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

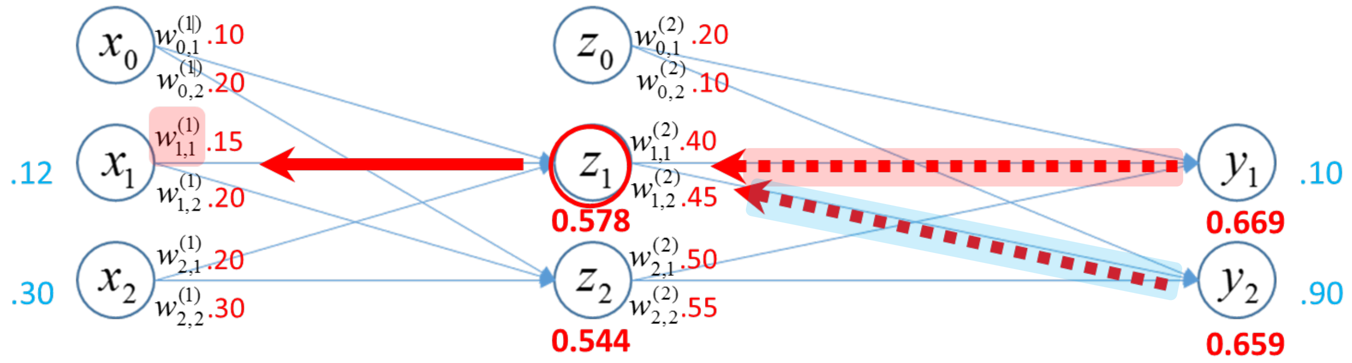
Previously calculated

$$\frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial y_{1(output)}} \times \frac{\partial y_{1(output)}}{\partial y_{1(input)}} \times \frac{\partial y_{1(input)}}{\partial z_{1(ouput)}}$$

0.0503 = 0.569 × 0.221 × 0.4

$$\frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} + \frac{\partial E(\mathrm{w})_{y_2}}{\partial z_{1(ouput)}}$$

Backpropagation algorithm - Backwarding



$$y_{2(input)} = w_{1,2}^{(2)} z_1 + w_{2,2}^{(2)} z_2 + w_{0,2}^{(2)}$$

$$\frac{\partial y_{2(input)}}{\partial z_{1(output)}} = w_{1,2}^{(2)} = 0.45$$

$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

$$\frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} + \frac{\partial E(\mathrm{w})_{y_2}}{\partial z_{1(ouput)}}$$

Previously calculated

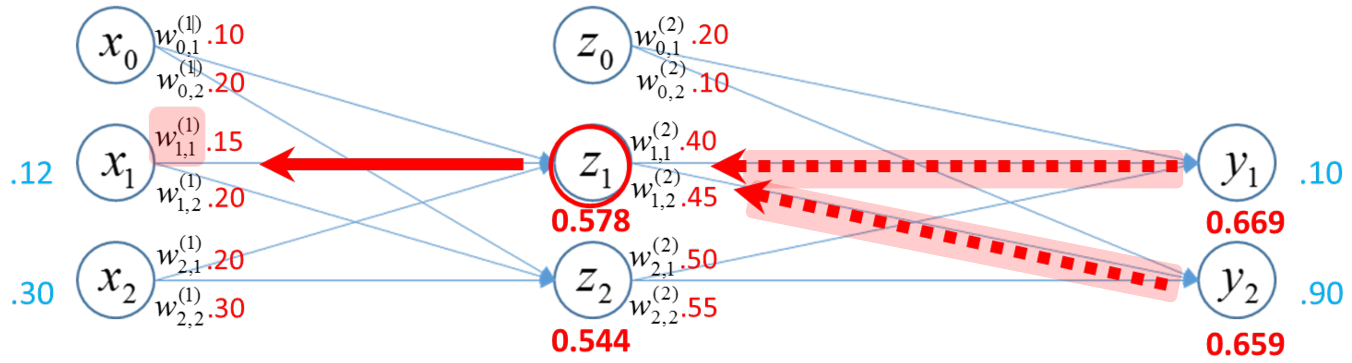
$$\frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial y_{1(output)}} \times \frac{\partial y_{1(output)}}{\partial y_{1(input)}} \times \frac{\partial y_{1(input)}}{\partial z_{1(ouput)}}$$

0.0503 0.569 0.221 0.4

$$\frac{\partial E(\mathrm{w})_{y_2}}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_2}}{\partial y_{2(output)}} \times \frac{\partial y_{2(output)}}{\partial y_{2(input)}} \times \frac{\partial y_{2(input)}}{\partial z_{2(ouput)}}$$

-0.0244 -0.241 0.225 0.45

Backpropagation algorithm - Backwarding



$$y_{2(input)} = w_{1,2}^{(2)} z_1 + w_{2,2}^{(2)} z_2 + w_{0,2}^{(2)}$$

$$\frac{\partial y_{2(input)}}{\partial z_{1(output)}} = w_{1,2}^{(2)} = 0.45$$

$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

0.0259

$$\frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} + \frac{\partial E(\mathrm{w})_{y_2}}{\partial z_{1(ouput)}}$$

$$= 0.0503 - 0.0244 = 0.0259$$

Previously calculated

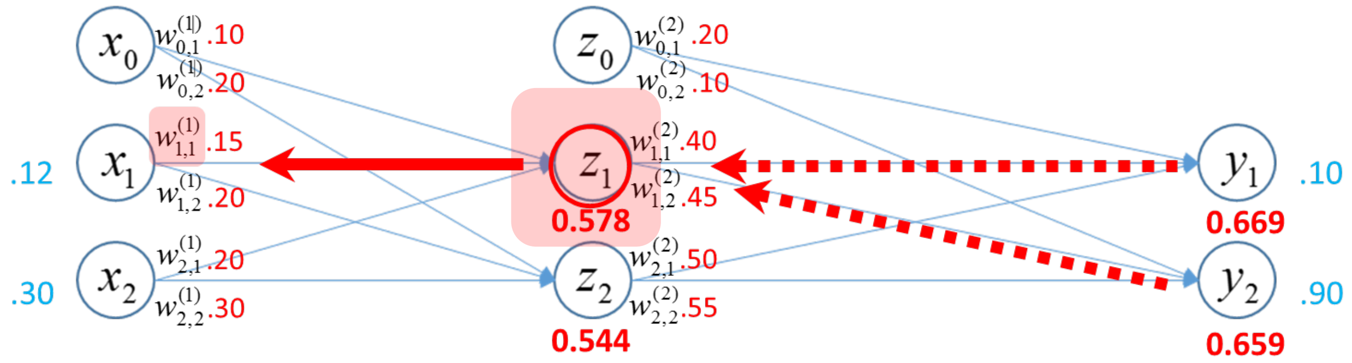
$$\frac{\partial E(\mathrm{w})_{y_1}}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_1}}{\partial y_{1(output)}} \times \frac{\partial y_{1(output)}}{\partial y_{1(input)}} \times \frac{\partial y_{1(input)}}{\partial z_{1(ouput)}}$$

0.0503 = **0.569** × **0.221** × **0.4**

$$\frac{\partial E(\mathrm{w})_{y_2}}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})_{y_2}}{\partial y_{2(output)}} \times \frac{\partial y_{2(output)}}{\partial y_{2(input)}} \times \frac{\partial y_{2(input)}}{\partial z_{2(ouput)}}$$

-0.0244 = **-0.241** × **0.225** × **0.45**

Backpropagation algorithm - Backwarding



$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

0.2439 0.0259

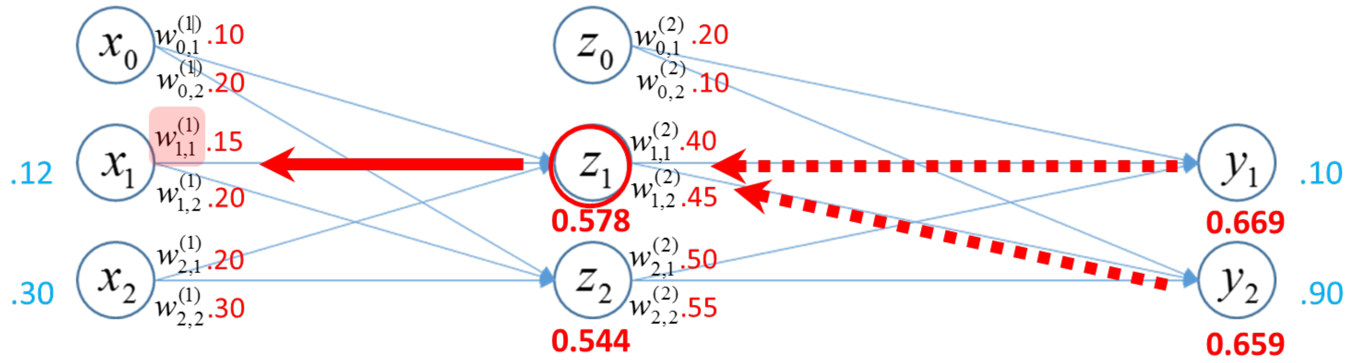
$$z_{1(ouput)} = \frac{1}{1+e^{-z_{1(input)}}}$$

$$\frac{\partial z_{1(output)}}{\partial z_{1(input)}} = \sigma(z_{1(input)})\left(1-\sigma(z_{1(input)})\right)$$

$$= z_{1(output)}(1 - z_{1(output)})$$

$$= 0.578 \times (1 - 0.578) = 0.2439$$

Backpropagation algorithm - Backwarding



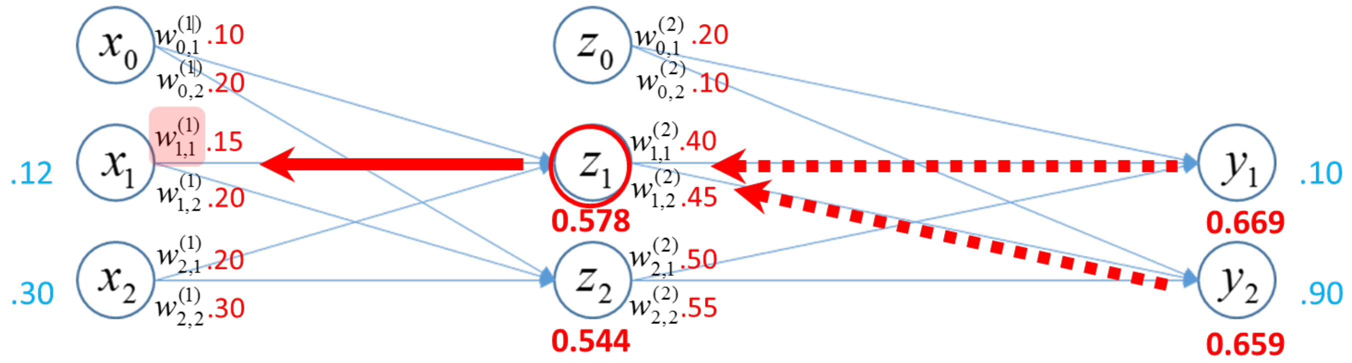
$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

0.12 **0.2439** **0.0259**

$$z_{1(input)} = w_{1,1}^{(1)} x_1 + w_{2,1}^{(1)} x_2 + w_{0,1}^{(1)}$$

$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} = x_1 = 0.12$$

Backpropagation algorithm - Backwarding



$$\frac{\partial z_{1(input)}}{\partial w_{1,1}^{(1)}} \times \frac{\partial z_{1(ouput)}}{\partial z_{1(input)}} \times \frac{\partial E(\mathrm{w})}{\partial z_{1(ouput)}} = \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

0.12 **0.2439** **0.0259**

$$\frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}} = \mathbf{0.00075}$$

$$w_{1,1}^{(1)*} = w_{1,1}^{(1)} + \eta \frac{\partial E(\mathrm{w})}{\partial w_{1,1}^{(1)}}$$

$$= 0.15 - 0.5 \times 0.00075$$

= **0.149625**

Same procedures are applied for $w_{1,2}^{(1)}, w_{2,1}^{(1)}, w_{2,2}^{(1)}$