NYCU DLP
Lab2 - Backpropagation

TA 陳鵬宇

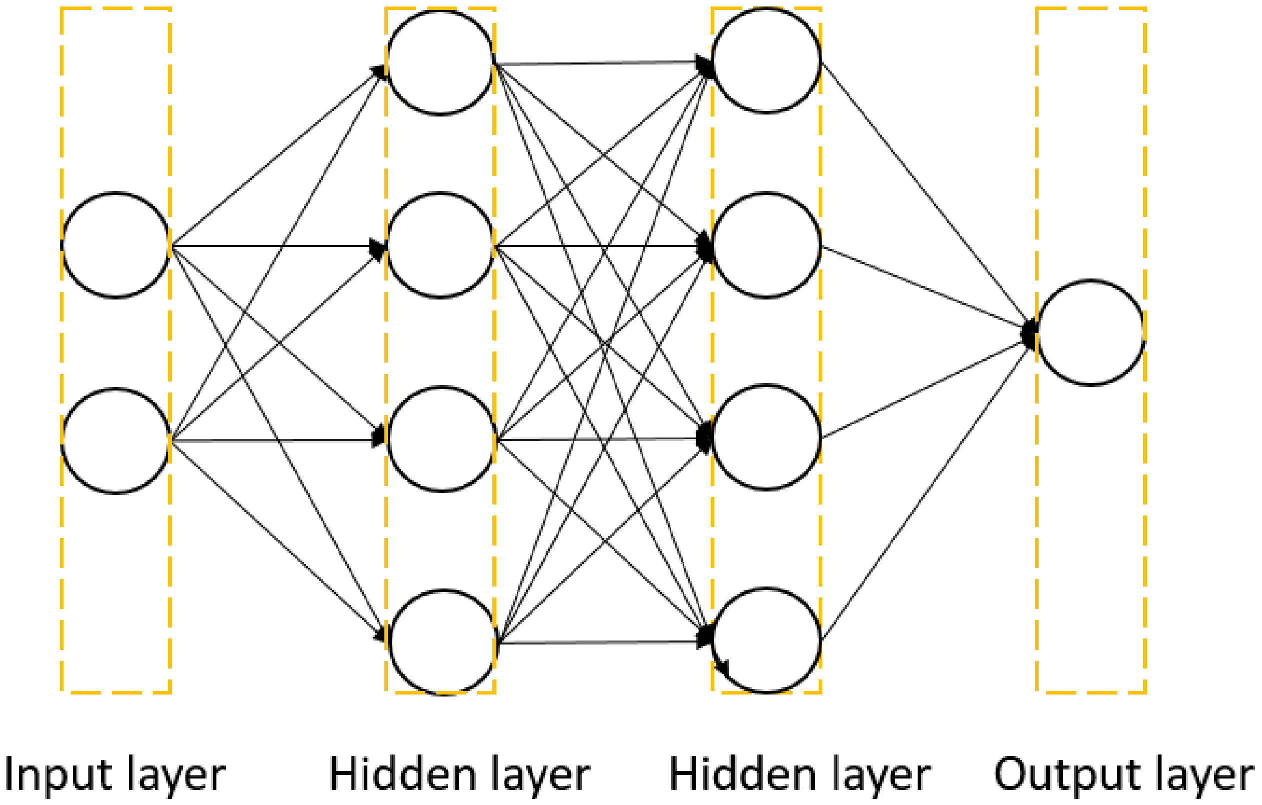
March 15, 2022

Outline

- Lab Objective
- Important Date
- Lab Description
- Scoring Criteria

Lab Objective

- In this lab, you will need to understand and implement a simple neural network with forward and backward pass using two hidden layers



Important Date

- Report Submission Deadline: 3/22 (Tue) 11:59 a.m.

- Demo date: 3/22 (Tue)

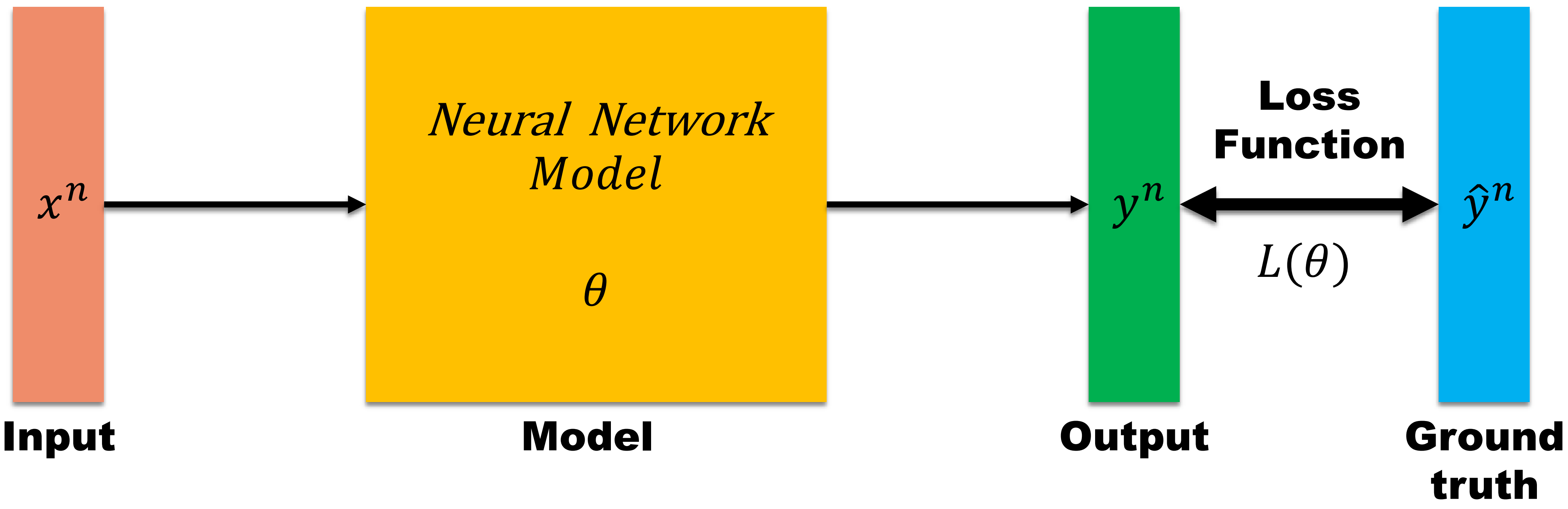
- Zip all files in one file
  - Report (.pdf)
  - Source code

- name it like「DLP_LAB2_yourstudentID_name.zip」
  - ex: 「DLP_LAB2_309551113_陳鵬宇.zip」

Lab Description

- Implement a simple neural network with two hidden layers
- You can only use Numpy and other python standard libraries.
- Plot your comparison figure showing the predictions and ground truth.
- Plot your learning curve (loss, epoch).
- Print the accuracy of your prediction.

Lab Description



$$\theta = \{w_1, w_2, w_3, w_4, \cdots\}$$

$$\nabla \boldsymbol{L}(\boldsymbol{\theta}) = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \partial L(\theta)/\partial w_3 \\ \vdots \\ \vdots \end{bmatrix}$$

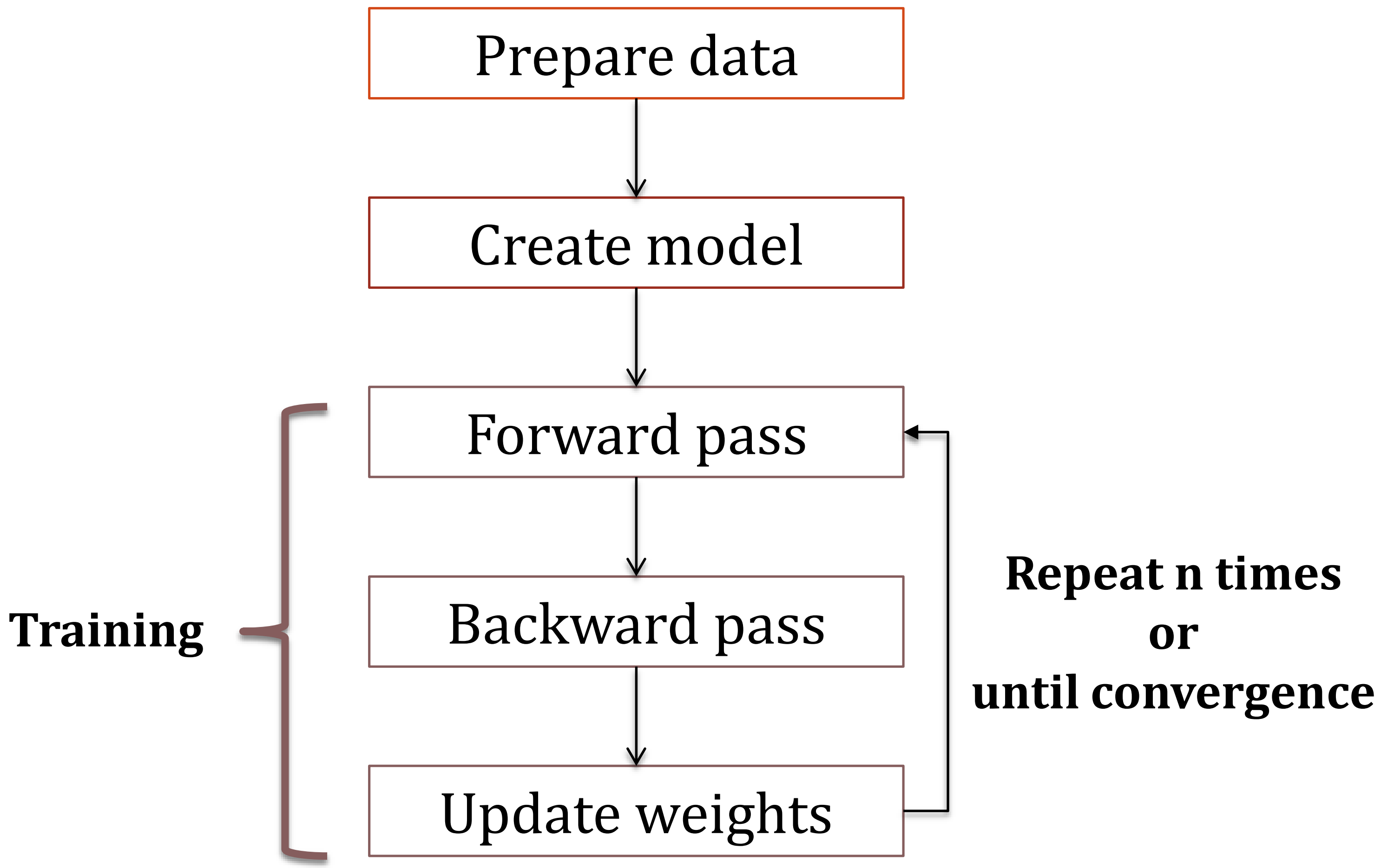
Compute $\nabla L(\theta^0)$ $\quad \theta^1 = \theta^0 - \rho \nabla L(\theta^0)$

Compute $\nabla L(\theta^1)$ $\quad \theta^2 = \theta^0 - \rho \nabla L(\theta^1)$

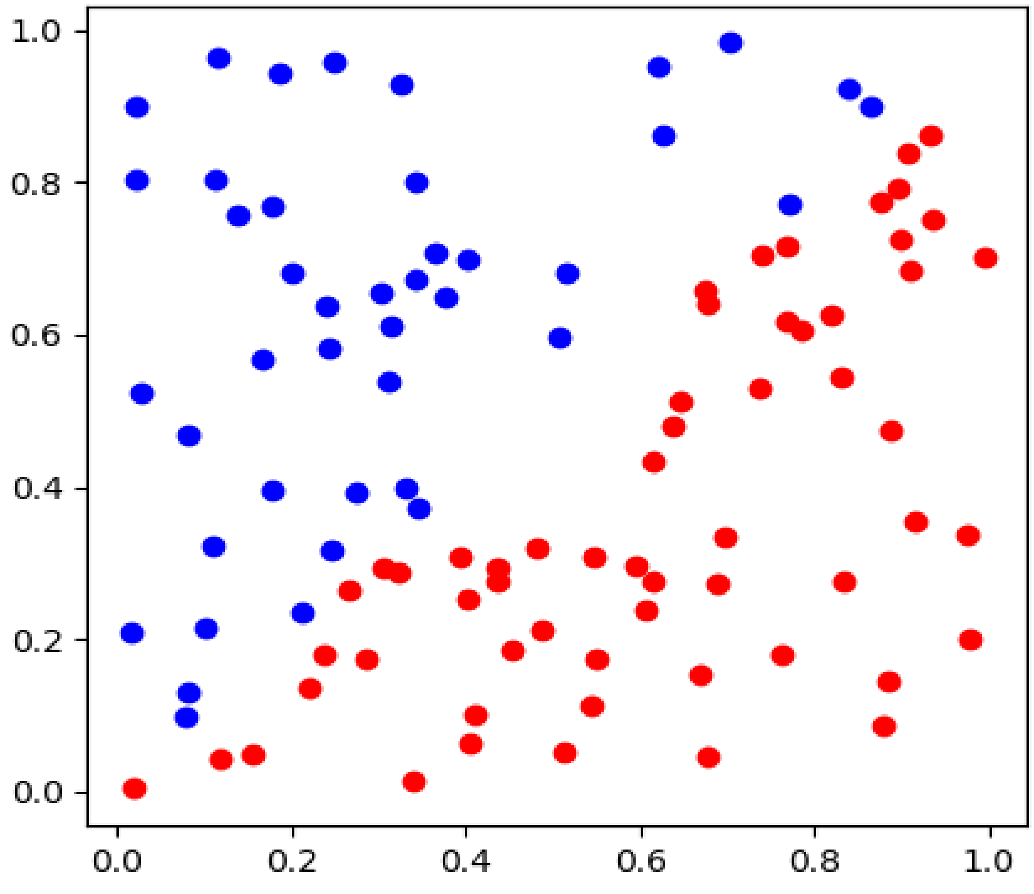
Compute $\nabla L(\theta^2)$ $\quad \theta^3 = \theta^0 - \rho \nabla L(\theta^2)$

ρ : Learning rate

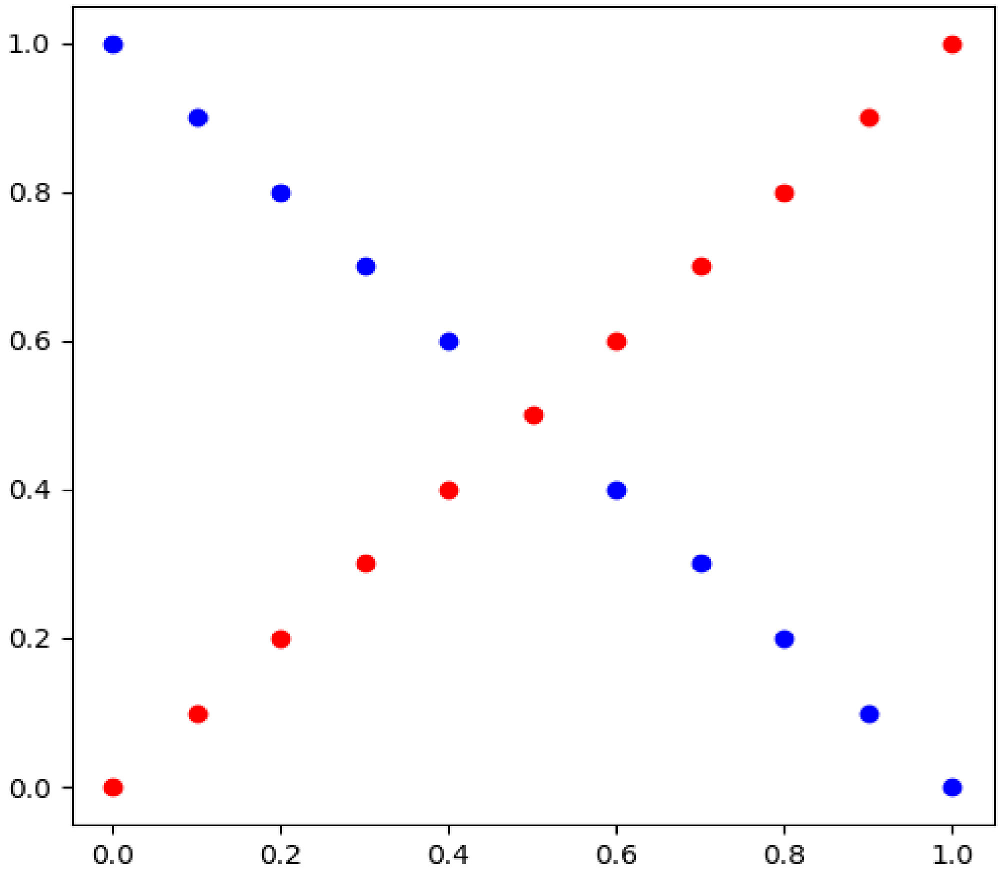
Lab Description – Flowchart



Lab Description - Data

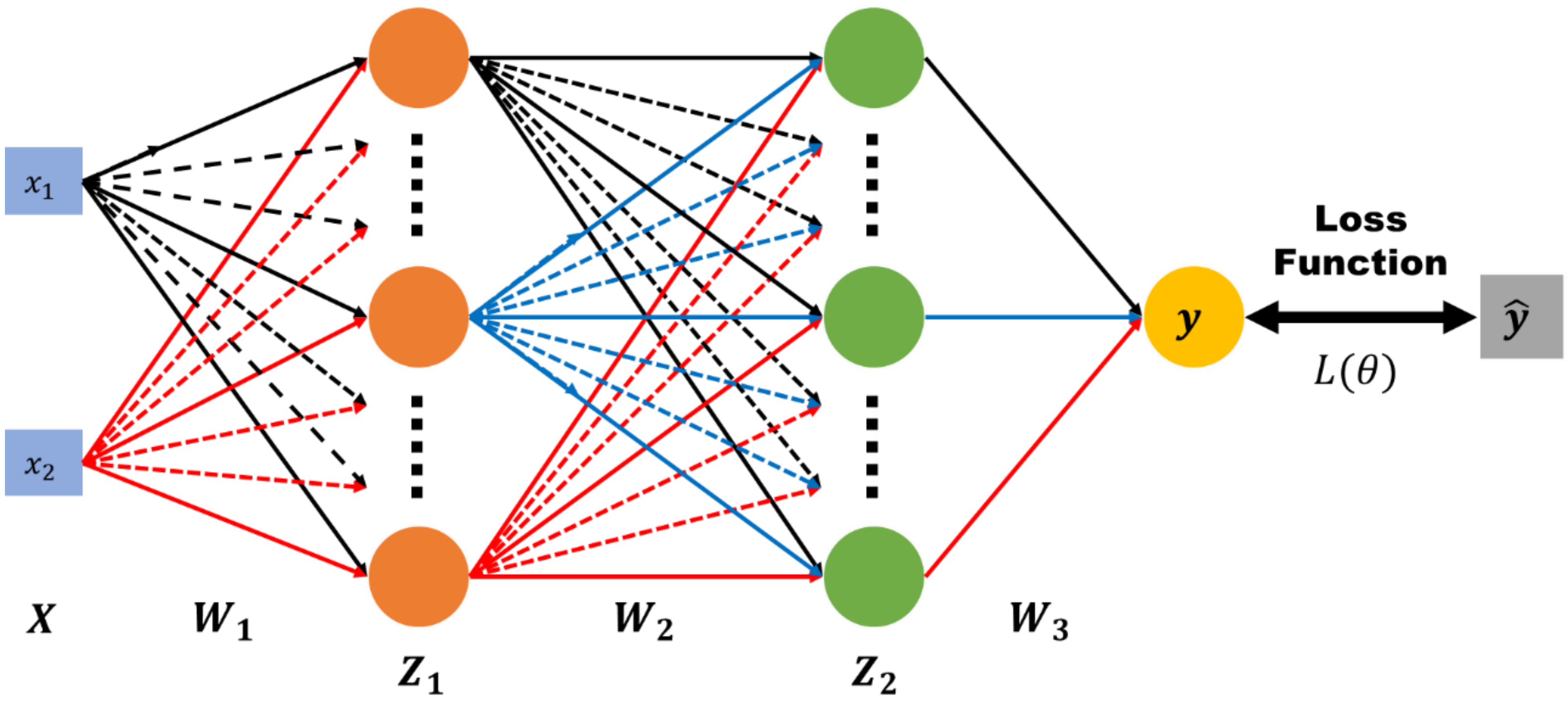


generate_linear()



generate_XOR_easy()

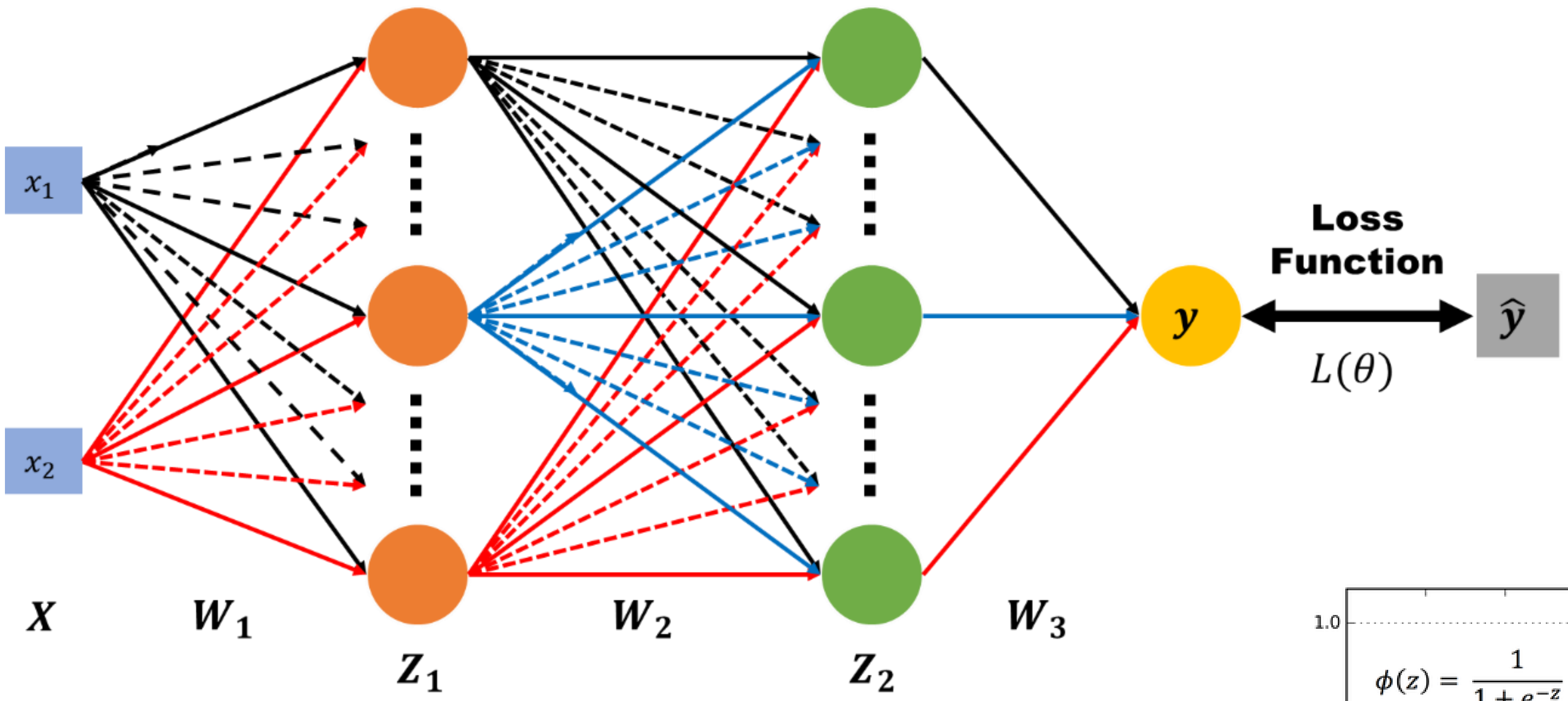
Lab Description – Architecture



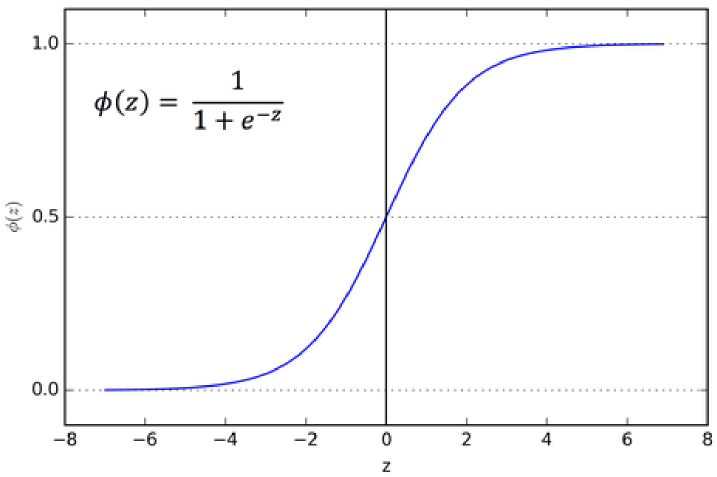
X : $[x_1, x_2]$ $\quad$ y : *outputs* $\quad$ $\hat{y}$: *ground truth*

W_1, W_2, W_3 : *weight matrix of network layers*

Lab Description – Forward

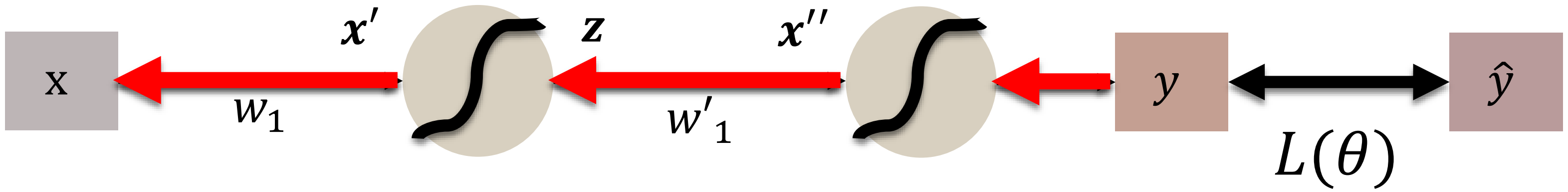


$$Z_1 = \sigma(XW_1) \qquad Z_2 = \sigma(Z_1W_2) \qquad y = \sigma(Z_2W_3)$$



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

Lab Description – Backward



$\boldsymbol{x' = xw_1}$ $\quad$ $\boldsymbol{z = \sigma(x')}$ $\quad$ $\boldsymbol{x'' = zw'_1}$ $\quad$ $\boldsymbol{y = \sigma(x'')}$

Chain rule

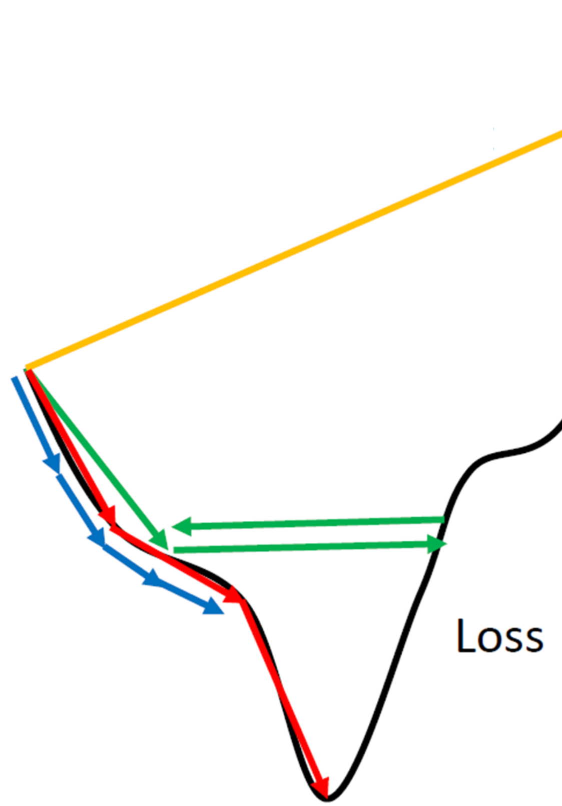
$$y = g(x) \qquad z = h(y)$$

$$\mathbf{x} \overset{\mathbf{g()}}{\longrightarrow} \boldsymbol{y} \overset{\boldsymbol{h}()}{\longrightarrow} \boldsymbol{z} \qquad \frac{\boldsymbol{dz}}{\boldsymbol{dx}} = \frac{\boldsymbol{dz}}{\cancel{\boldsymbol{dy}}} \frac{\cancel{\boldsymbol{dy}}}{\boldsymbol{dx}}$$

$$\begin{aligned}
\frac{\partial L(\theta)}{\partial w_1} &= \frac{\partial y}{\partial w_1} \frac{\partial L(\theta)}{\partial y} \\
&= \frac{\partial x''}{\partial w_1} \frac{\partial y}{\partial x''} \frac{\partial L(\theta)}{\partial y} \\
&= \frac{\partial z}{\partial w_1} \frac{\partial x''}{\partial z} \frac{\partial y}{\partial x''} \frac{\partial L(\theta)}{\partial y} \\
&= \frac{\partial x'}{\partial w_1} \frac{\partial z}{\partial x'} \frac{\partial x''}{\partial z} \frac{\partial y}{\partial x''} \frac{\partial L(\theta)}{\partial y}
\end{aligned}$$

Lab Description – Gradient descent

Network Parameters $\theta = \{w_1, w_2, w_3, w_4, \cdots\}$

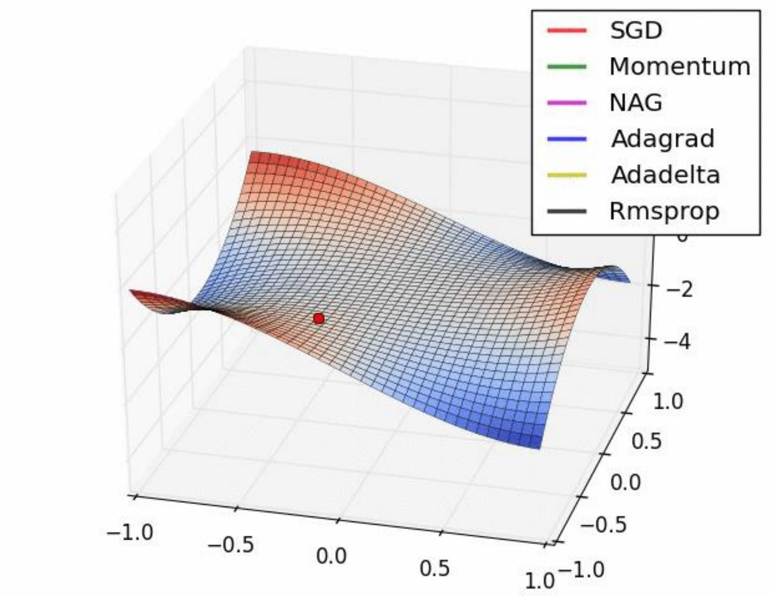


$$\theta^1 = \theta^0 - \rho \nabla L(\theta^0)$$

$$\theta^2 = \theta^0 - \rho \nabla L(\theta^1)$$

$$\theta^3 = \theta^0 - \rho \nabla L(\theta^2)$$

ρ : Learning rate



http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Lab Description - Prediction

- In the training, you need to print loss

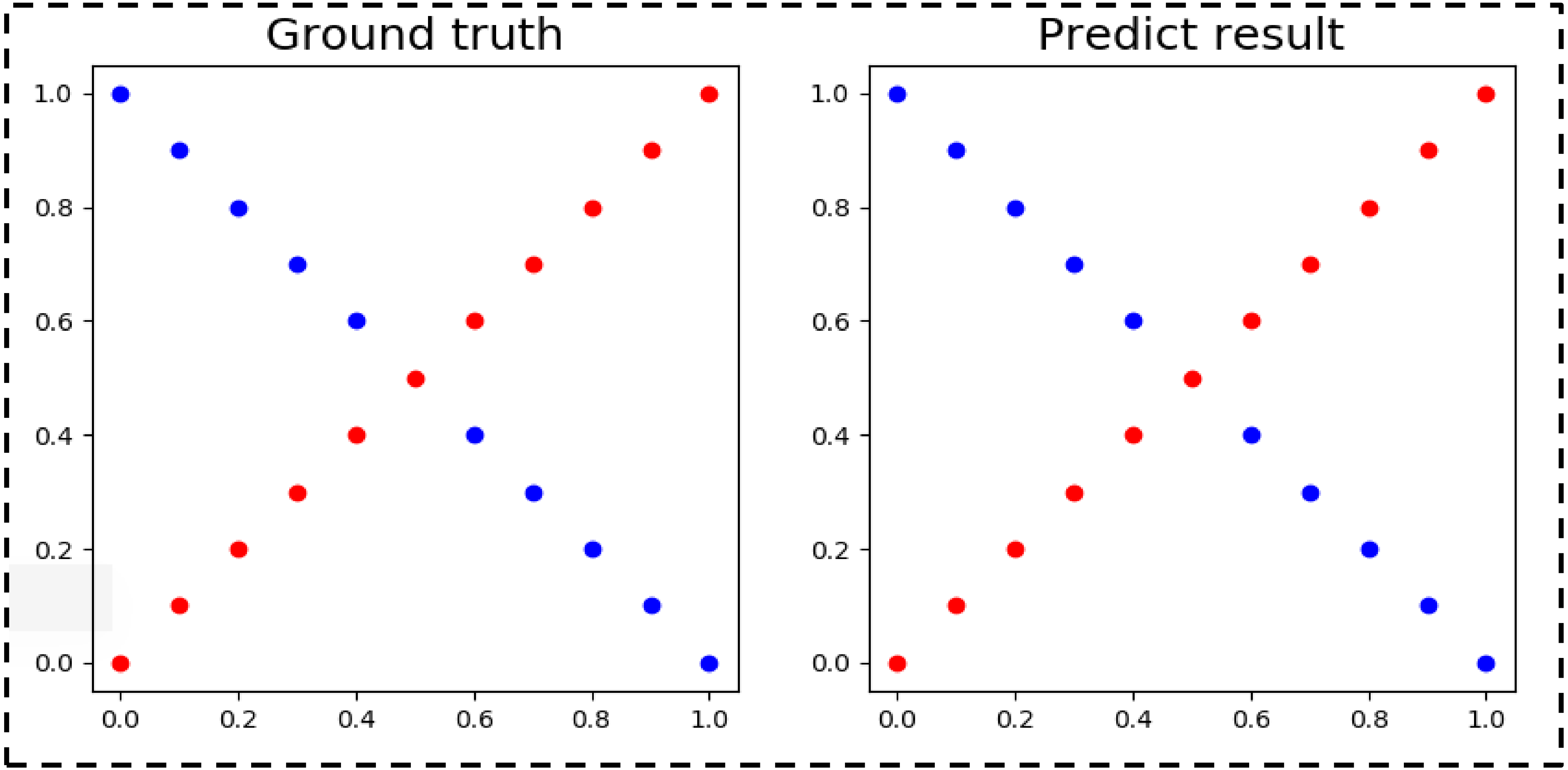
```
epoch 10000 loss : 0.16234523253277644
epoch 15000 loss : 0.2524336634177614
epoch 20000 loss : 0.1590783047540092
epoch 25000 loss : 0.22099447030234853
epoch 30000 loss : 0.3292173477217561
epoch 35000 loss : 0.40406233282426085
epoch 40000 loss : 0.43052897480298924
epoch 45000 loss : 0.4207525735586605
epoch 50000 loss : 0.3934759509342479
epoch 55000 loss : 0.3615008372106921
epoch 60000 loss : 0.33077879872648525
epoch 65000 loss : 0.30333537090819584
epoch 70000 loss : 0.2794858089741792
epoch 75000 loss : 0.25892812312991587
epoch 80000 loss : 0.24119780823897027
epoch 85000 loss : 0.22583656353511342
epoch 90000 loss : 0.21244497028971704
epoch 95000 loss : 0.2006912468389013
```

- In the testing, you need to show your predictions, also the accuracy

```
[[0.01025062]
 [0.99730607]
 [0.02141321]
 [0.99722154]
 [0.03578171]
 [0.99701922]
 [0.04397049]
 [0.99574117]
 [0.04162245]
 [0.92902792]
 [0.03348791]
 [0.02511045]
 [0.94093942]
 [0.01870069]
 [0.99622948]
 [0.01431959]
 [0.99434455]
 [0.01143039]
 [0.98992477]
 [0.00952752]
 [0.98385905]]
```

Lab Description - Prediction

- Visualize the predictions and ground truth at the end of the training process



Scoring Criteria

- Report (40%)
- Demo(60%)
  - Experimental results (40%)
  - Questions (20%)
- Late report or demo
  - Before 6/28.

Reference

1. *http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html*
2. *http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML17_2.html*

Optimizers

- SGD – minibatch
- Momentum

Repeat Until Convergence {

$$\nu_j \leftarrow \eta * \nu_j - \alpha * \nabla_w \sum_1^m L_m(w)$$

$$\omega_j \leftarrow \nu_j + \omega_j$$

}

- Adagrad

Adagrad

$$v_t = v_{t-1} + (\nabla w_t)^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{(v_t)} + \epsilon} \nabla w_t$$

- Adam

For each Parameter w^j

(j subscript dropped for clarity)

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*

g_t : *Gradient at time t along* ω^j

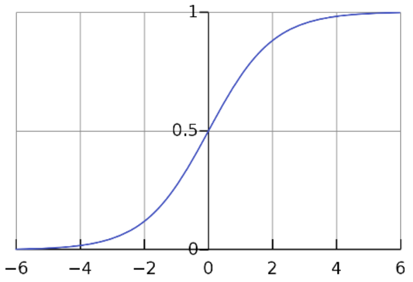
ν_t : *Exponential Average of gradients along* ω_j

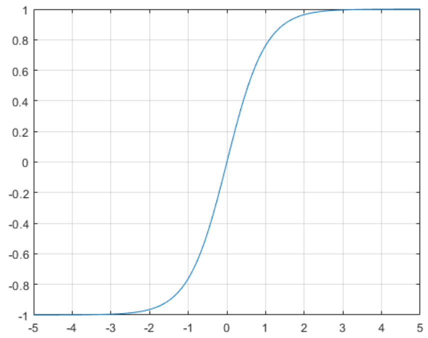
s_t : *Exponential Average of squares of gradients along* ω_j

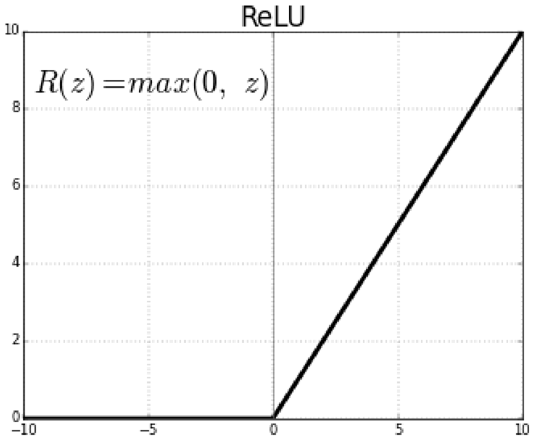
β_1, β_2 : *Hyperparameters*

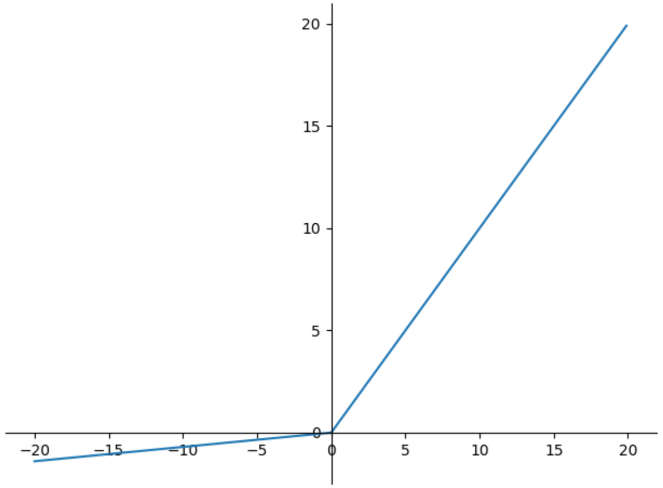
Activation functions

- Sigmoid
- Tanh
- Relu
- Leaky Relu









Backpropagation in Convolutional Neural Network

$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \textbf{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

$$O_{11} = F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22}$$

$$O_{12} = F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23}$$

$$O_{21} = F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32}$$

$$O_{22} = F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}$$

$$\begin{bmatrix} \partial E/\partial X_{11} & \partial E/\partial X_{12} & \partial E/\partial X_{13} \\ \partial E/\partial X_{21} & \partial E/\partial X_{22} & \partial E/\partial X_{23} \\ \partial E/\partial X_{31} & \partial E/\partial X_{32} & \partial E/\partial X_{33} \end{bmatrix} = \textbf{Full\_Convolution}\left(\begin{bmatrix} \partial E/\partial O_{11} & \partial E/\partial O_{12} \\ \partial E/\partial O_{21} & \partial E/\partial O_{22} \end{bmatrix}, \begin{bmatrix} F_{22} & F_{21} \\ F_{12} & F_{11} \end{bmatrix} \right)$$

$$\begin{bmatrix} \partial E/\partial F_{11} & \partial E/\partial F_{12} \\ \partial E/\partial F_{21} & \partial E/\partial F_{22} \end{bmatrix} = \textbf{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \partial E/\partial O_{11} & \partial E/\partial O_{12} \\ \partial E/\partial O_{21} & \partial E/\partial O_{22} \end{bmatrix} \right)$$

Reference : https://medium.com/@2017csm1006/forward-and-backpropagation-in-convolutional-neural-network-4dfa96d7b37e