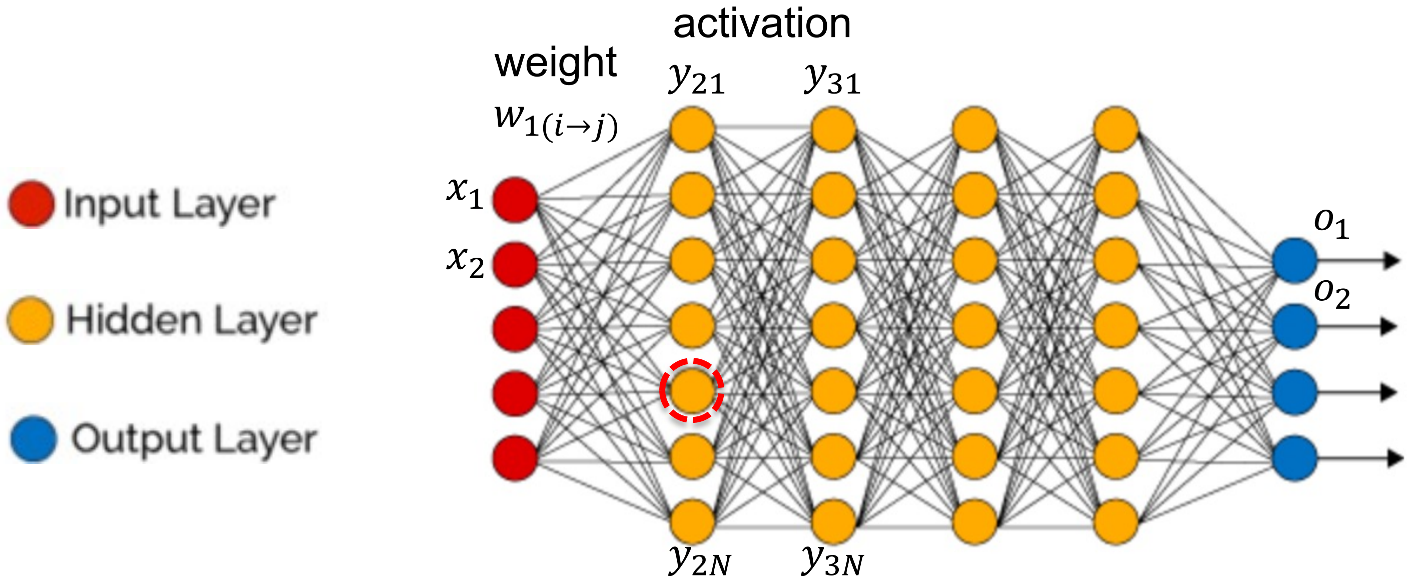
ECE284 Fall 21 W2S1

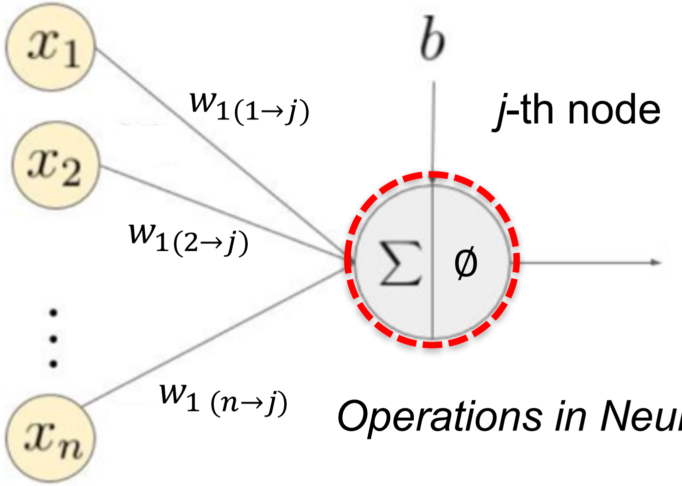
Low-power VLSI Implementation for Machine Learning

Prof. Mingu Kang

UCSD Computer Engineering

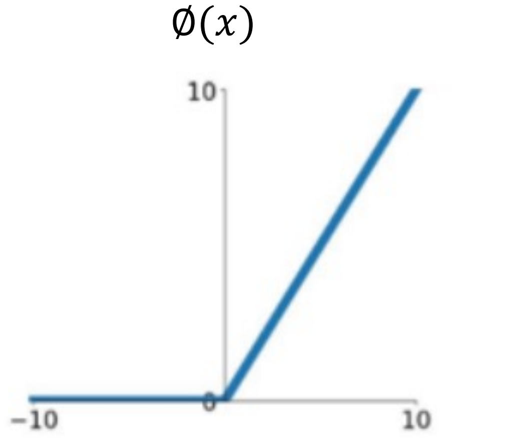
Deep Neural Network – Multi layer Perceptron (Inference)





$$\emptyset\left(b+\sum_{i=1}^{n} x_i w_{1(i\to j)}\right)$$

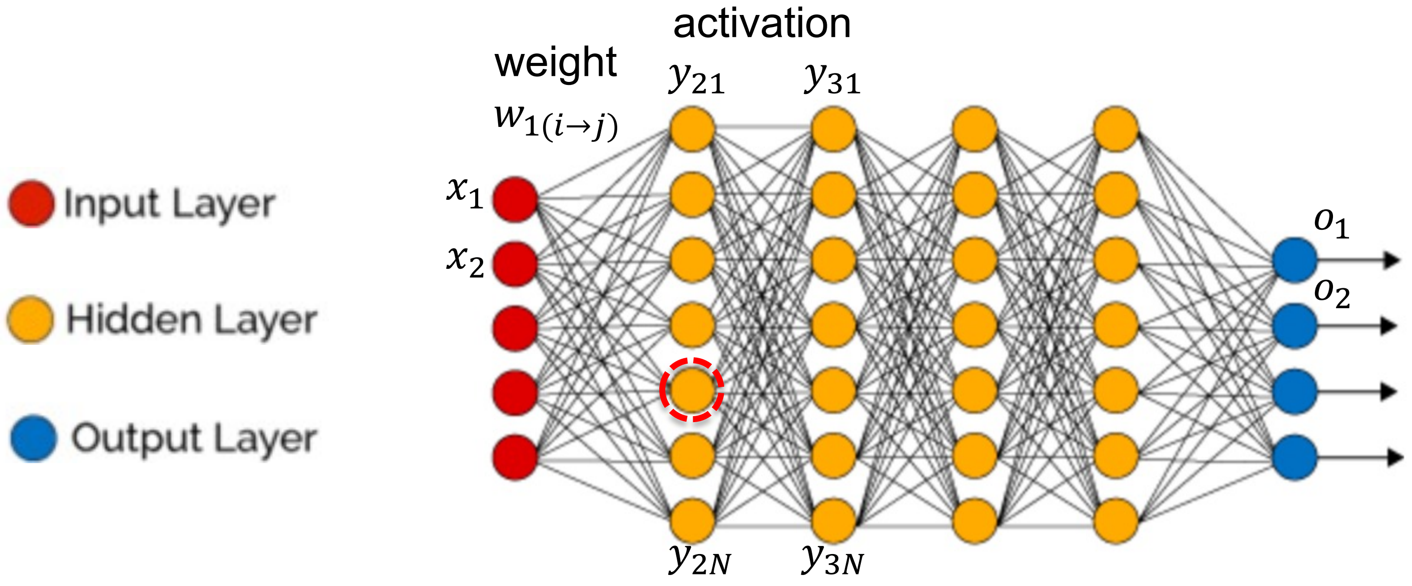
Operations in Neuron

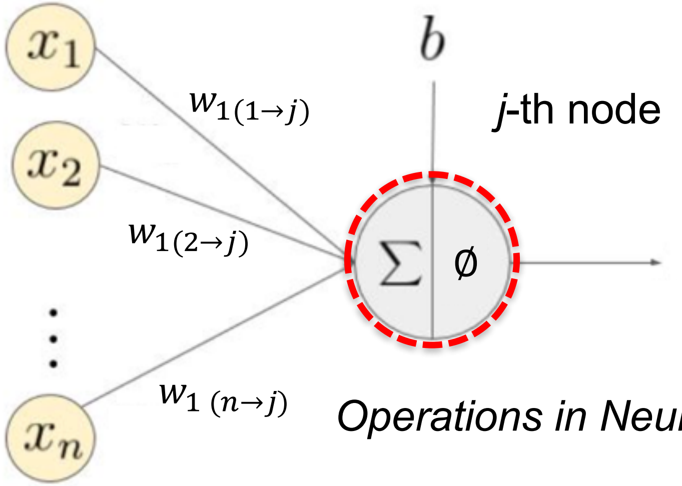


Activation function (ReLu)

Figures from https://becominghuman.ai/, https://www.learnopencv.com/ https://medium.com/

Deep Neural Network – Multi layer Perceptron (Inference)





$$\emptyset\left(b + \sum_{i=1}^{n} x_i w_{1(i\to j)}\right)$$

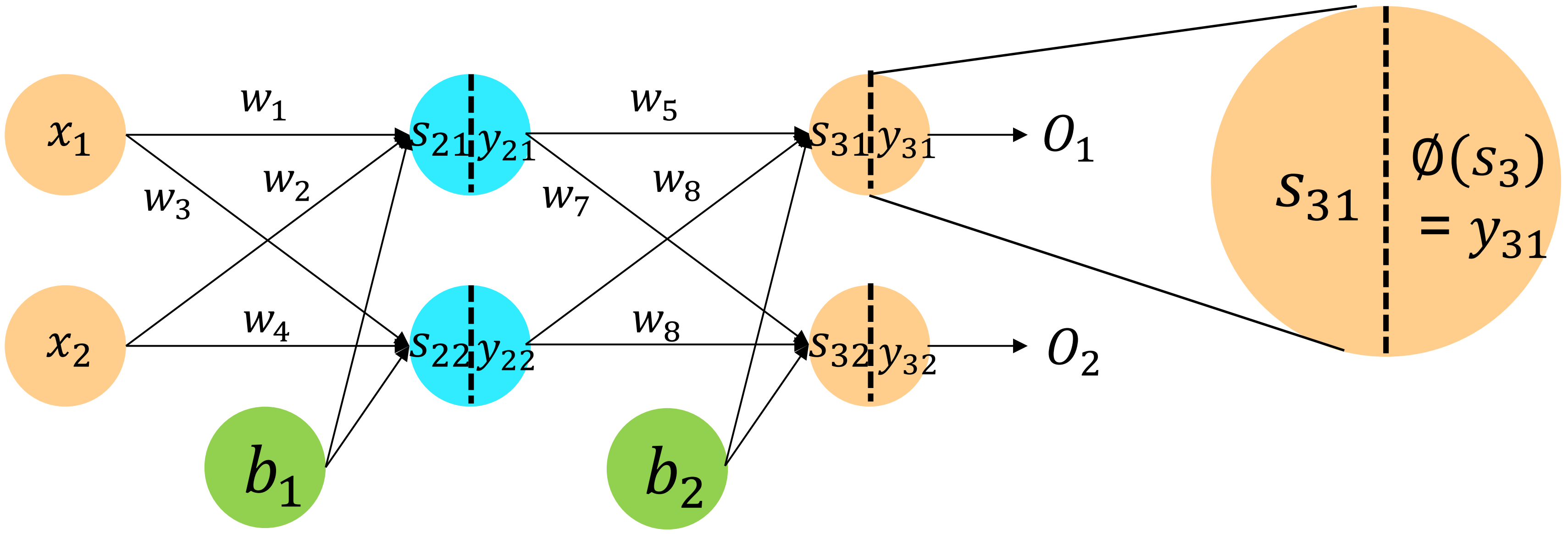
Operations in Neuron

$$\begin{bmatrix} w_{1(1\to 1)} & \cdots & w_{1(5\to 1)} \\ \vdots & \ddots & \vdots \\ w_{1(1\to N)} & \cdots & w_{1(5\to N)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y_{21} \\ y_{22} \\ y_{23} \\ \cdots \\ y_{2N} \end{bmatrix}$$

Matrix multiplication
(bias omitted for simplicity)

Figures from https://becominghuman.ai/, https://www.learnopencv.com/ https://medium.com/

Back Propagation (Example)



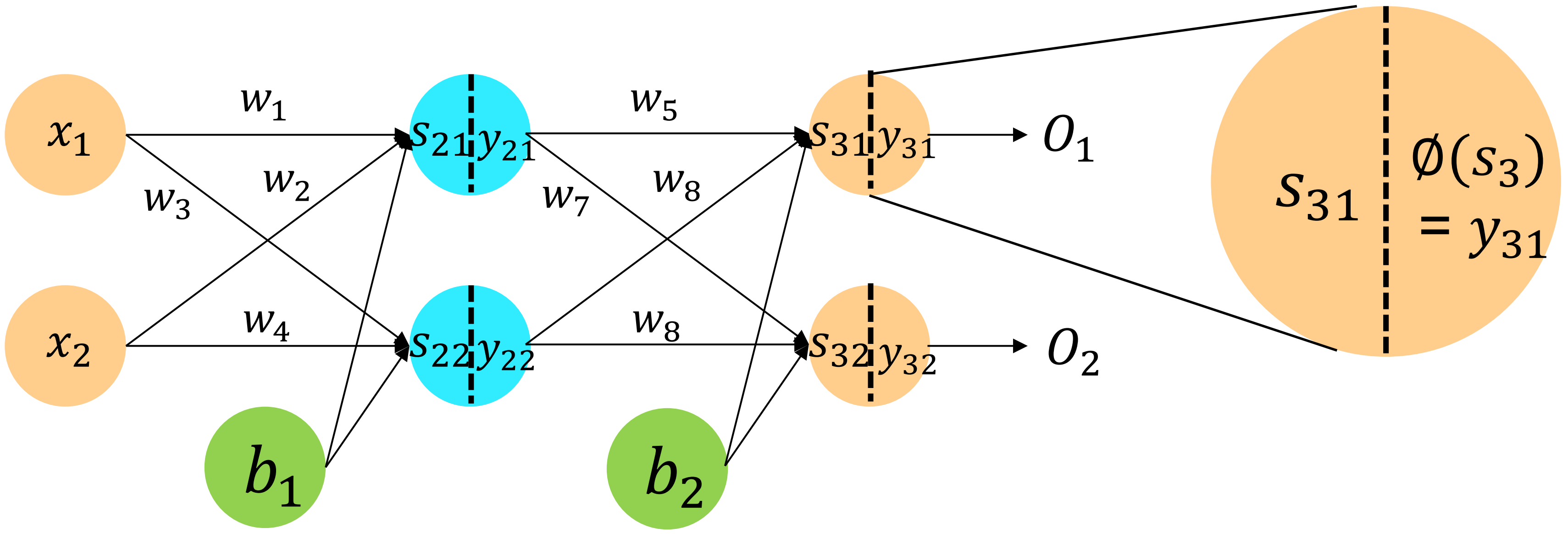
$$L = \sum_{i=1}^{2} \frac{1}{2}(T_i - y_{3i})^2$$

T_i: target value

$$\emptyset(x) = x \;@(x > 0)$$
$$0, \text{otherwise}$$

(ReLU)

Back Propagation (Example)

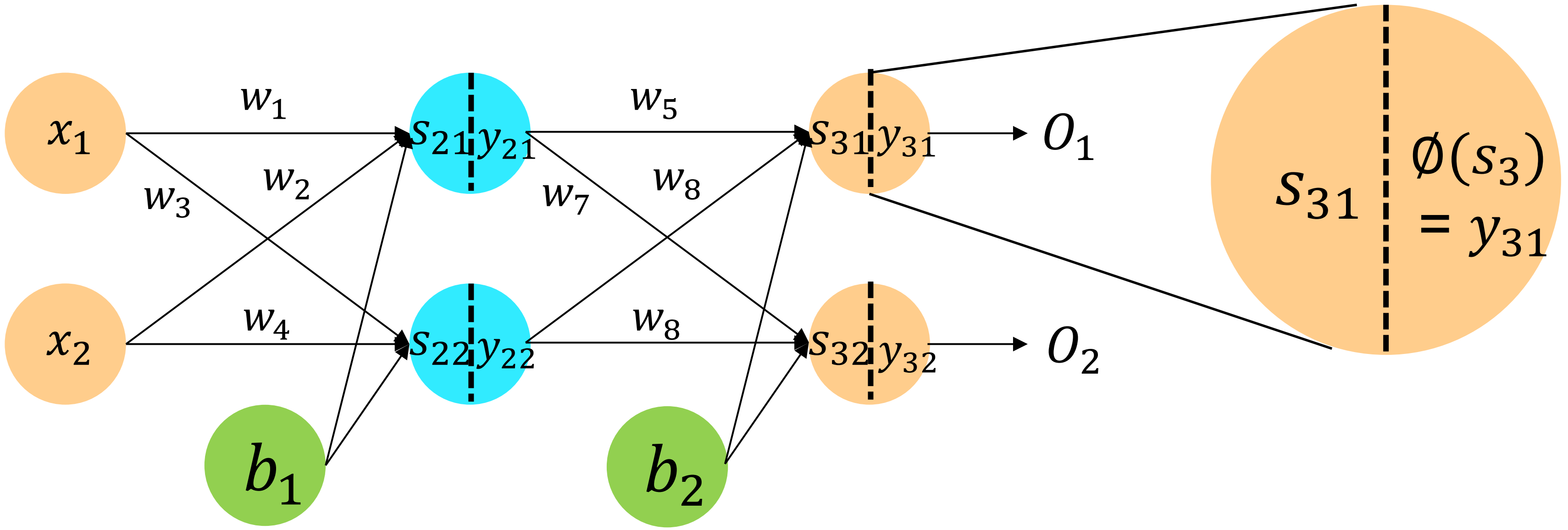


Purple box means typo has been corrected

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial y_{31}} * \frac{\partial y_{31}}{\partial s_{31}} * \frac{\partial s_{31}}{\partial w_5} = \boxed{(y_{31} - T_{31})} * \cancel{\emptyset'(s_{31})} * y_{21}$$

$$= \boxed{\frac{\partial L}{\partial y_{31}} * y_{21}}$$

Back Propagation (Example)

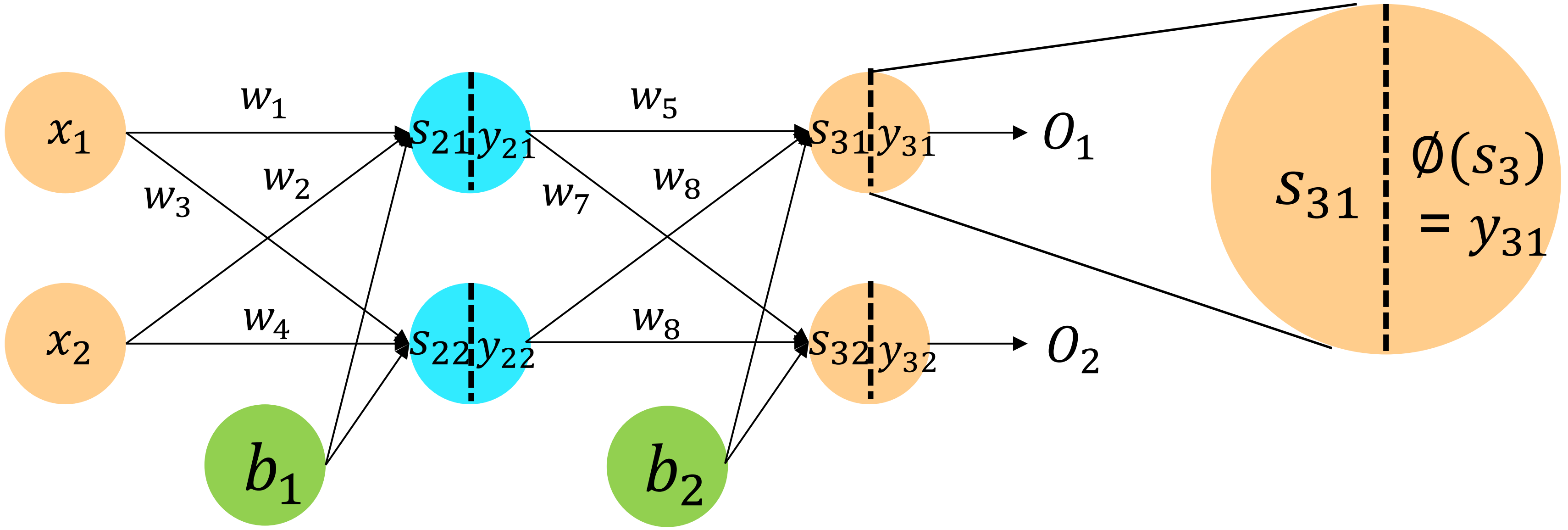


in the next page

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial s_{21}} * \frac{\partial s_{21}}{\partial w_1} = \frac{\partial L}{\partial y_{21}} * \frac{\partial y_{21}}{\partial s_{21}} * \frac{\partial s_{21}}{\partial w_1} = \boxed{\frac{\partial L}{\partial y_{21}}} * \cancel{\emptyset'(s_{21})} * x_1$$

$$\boxed{\frac{\partial L}{\partial w_{L,i\to j}} = y_{L,i} * \frac{\partial L}{\partial y_{(L+1),j}}}$$

Back Propagation (Example)

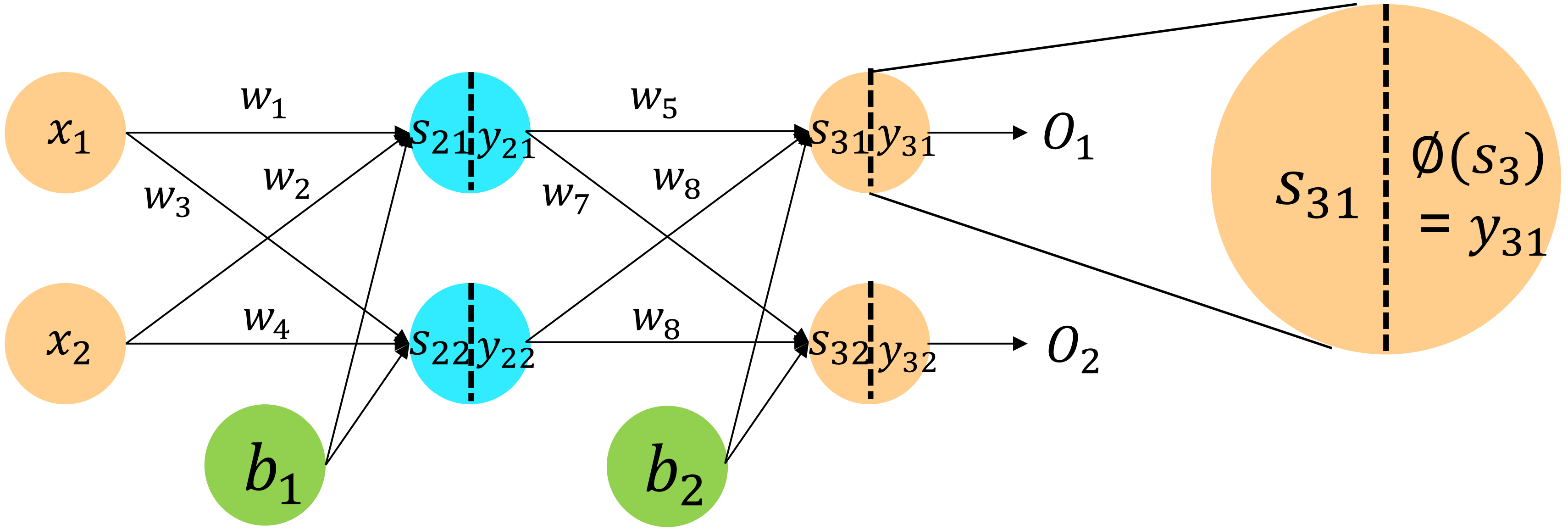


$$\frac{\partial L}{\partial y_{21}} = \frac{\boxed{\partial[0.5(T_1 - y_{31})^2] + \partial[0.5(T_2 - y_{32})^2]}}{\partial y_{21}}$$

$$\boxed{\frac{\partial[0.5(T_1 - y_{31})^2]}{\partial y_{21}}} = \frac{\partial[0.5(T_1 - y_{31})^2]}{\partial y_{31}} * \frac{\partial y_{31}}{\partial s_{31}} * \frac{\partial s_{31}}{\partial y_{21}} = \boxed{-1 * (T_1 - y_{31})} * \cancel{\emptyset'(s_{31})} * w_5$$

$$= \boxed{(y_{31} - T_1) * w_5}$$

Back Propagation (Example)

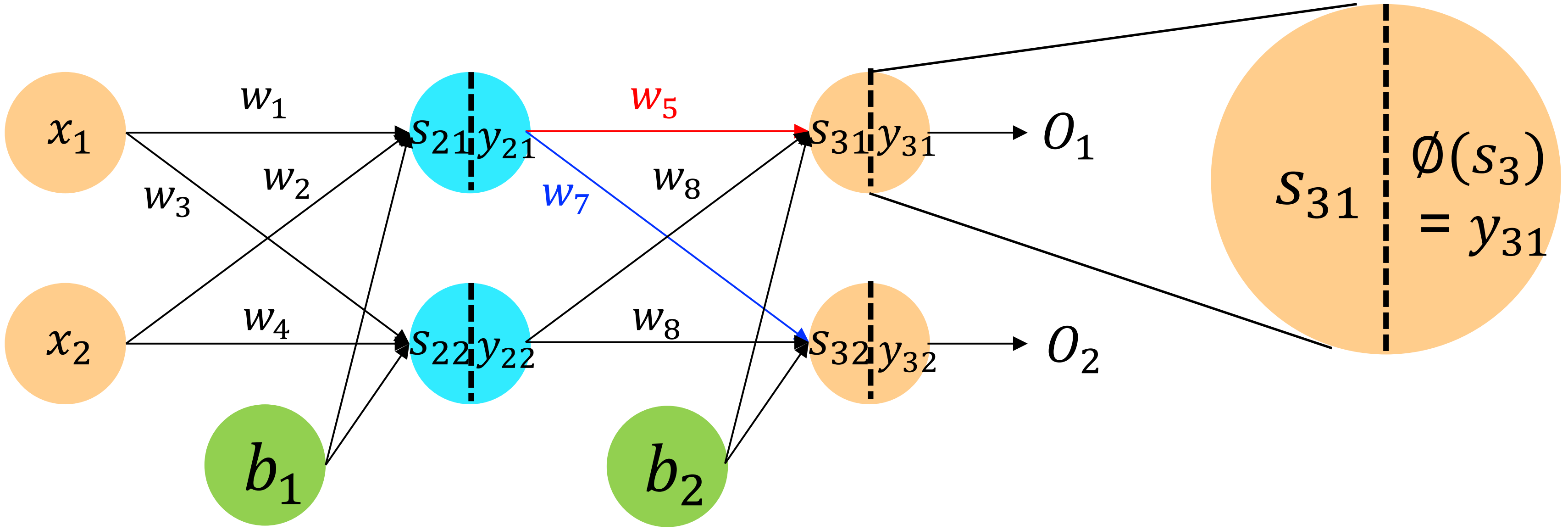


$$\frac{\partial L}{\partial y_{21}} = \frac{\boxed{\partial[0.5(T_1 - y_{31})^2] + \partial[0.5(T_2 - y_{32})^2]}}{\partial y_{21}}$$

$$\boxed{\frac{\partial[0.5(T_2 - y_{32})^2]}{\partial y_{21}}} = \frac{\boxed{\partial[0.5(T_2 - y_{32})^2]}}{\partial y_{32}} * \frac{\partial y_{32}}{\partial s_{32}} * \frac{\partial s_{32}}{\partial y_{21}} = \boxed{-1(T_2 - y_{32})} * \cancel{\emptyset'(s_{31})} * w_7$$

$$= \boxed{(y_{32} - T_{32}) * w_7}$$

Back Propagation (Example)

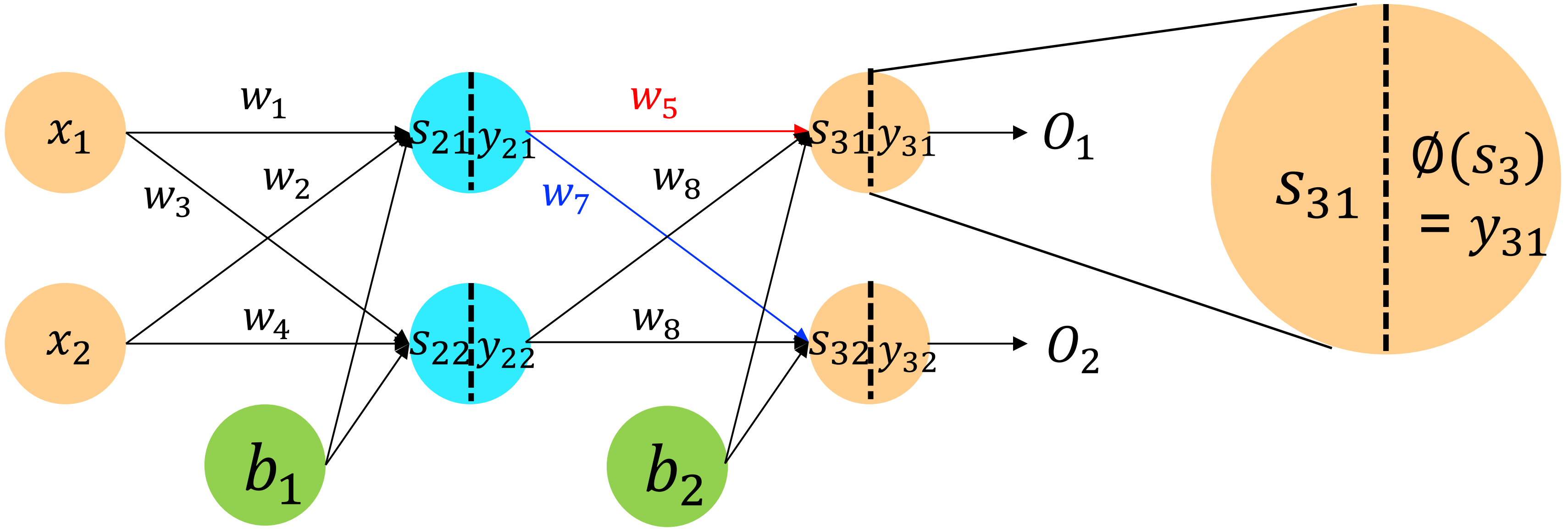


$$\frac{\partial L}{\partial y_{21}} = \frac{\boxed{\partial[0.5(T_1 - y_{31})^2] + \partial[0.5(T_2 - y_{32})^2]}}{\partial y_{21}}$$

$$= \boxed{(y_{31} - T_1) * w_5 + (y_{32} - T_2) * w_7}$$

$$\boxed{\frac{\partial L}{\partial y_{Li}} = \sum_j \frac{\partial L}{\partial y_{(L+1)j}} * w_{L(i \to j)}}$$

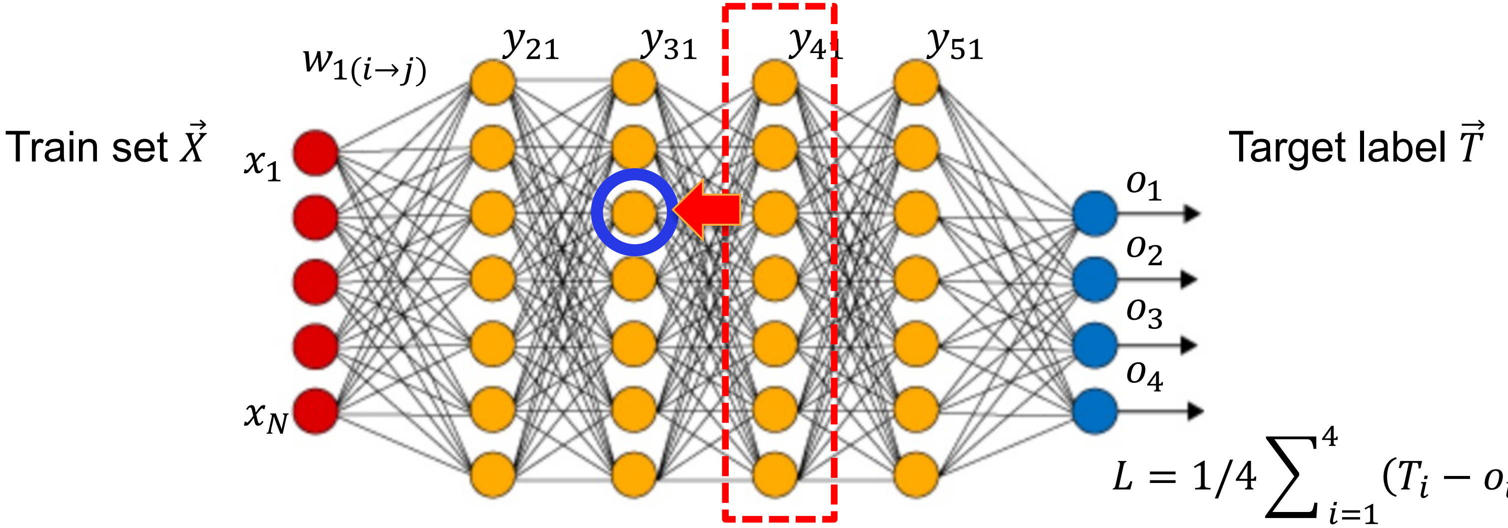
Back Propagation (Example)



$$\boxed{\frac{\partial L}{\partial w_{L,(i\to j)}} = y_{L,i} * \frac{\partial L}{\partial y_{(L+1),j}}}$$

$$\boxed{\frac{\partial L}{\partial y_{L,i}} = \sum_j \frac{\partial L}{\partial y_{(L+1),j}} * w_{L,(i\to j)}}$$

Back Propagation (Example)



$$L = 1/4 \sum_{i=1}^{4} (T_i - o_i)^2$$

Error: $$\frac{\partial L}{\partial y_{Li}} = \sum_{j=1}^{N} \frac{\partial L}{\partial y_{(L+1),j}} w_{L,(i\to j)}$$

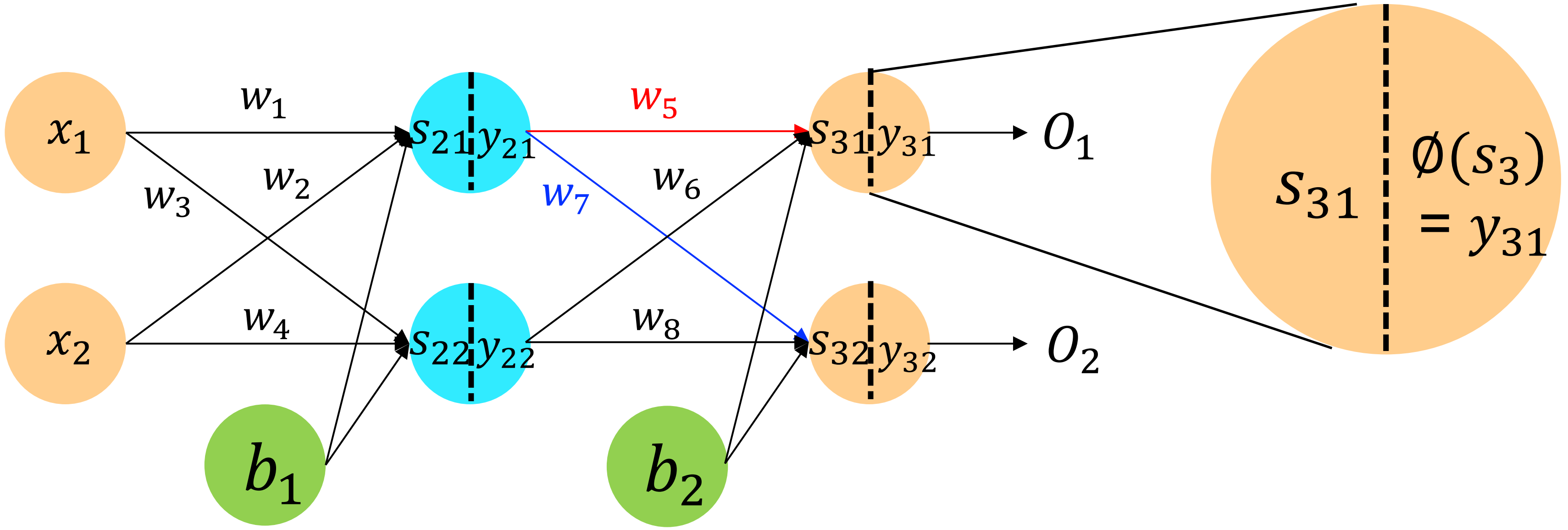
Gradient: $$\frac{\partial L}{\partial w_{L,(i\to j)}} = y_{Li} \frac{\partial L}{\partial y_{(L+1),j}}$$

Floating point computation required

Update: $$w'_{L,(i\to j)} = w_{L,(i\to j)} - \mu \frac{\partial L}{\partial w_{L,(i\to j)}}$$

μ: Learning rate updated per mini-batch

DNN Back Propagation



$$[x_1 \quad x_2]\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = [s_{21} \quad s_{22}]$$

$$[y_{21} \quad y_{22}]\begin{bmatrix} w_5 & w_7 \\ w_6 & w_8 \end{bmatrix} = [s_{31} \quad s_{32}]$$

Let:

$$\boldsymbol{W_1} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \qquad \boldsymbol{X} = [x_1 \quad x_2] \qquad \boldsymbol{S_2} = [s_{21} \quad s_{22}]$$

$$\boldsymbol{W_2} = \begin{bmatrix} w_5 & w_7 \\ w_6 & w_8 \end{bmatrix} \qquad \boldsymbol{Y_2} = [y_{21} \quad y_{22}] \qquad \boldsymbol{S_3} = [s_{31} \quad s_{32}]$$

[Example 1 & HW_Prob2(a)] Two-layer Perceptron Back Propagation

- HW_Prob2(a): calculate the gradients of W1 and W2 and submit the manually (hand-written or typed) calculated solution.
- running Jupyter Notebook to calculate the gradients of W1 and W2, and check your manual solution is correct.

[HW_Prob2(b)] ReLU with negative values

- Open “[HW2_prob2(b)]_DNN_Back_Propagation_with_Negative_Values_at_ReLU”
- This time, there are negative values in the matrices
- Repeat the same process with prob1(a)

- Manual calculation

- Run program (1st cell) to check your manual calculation with pytorch results

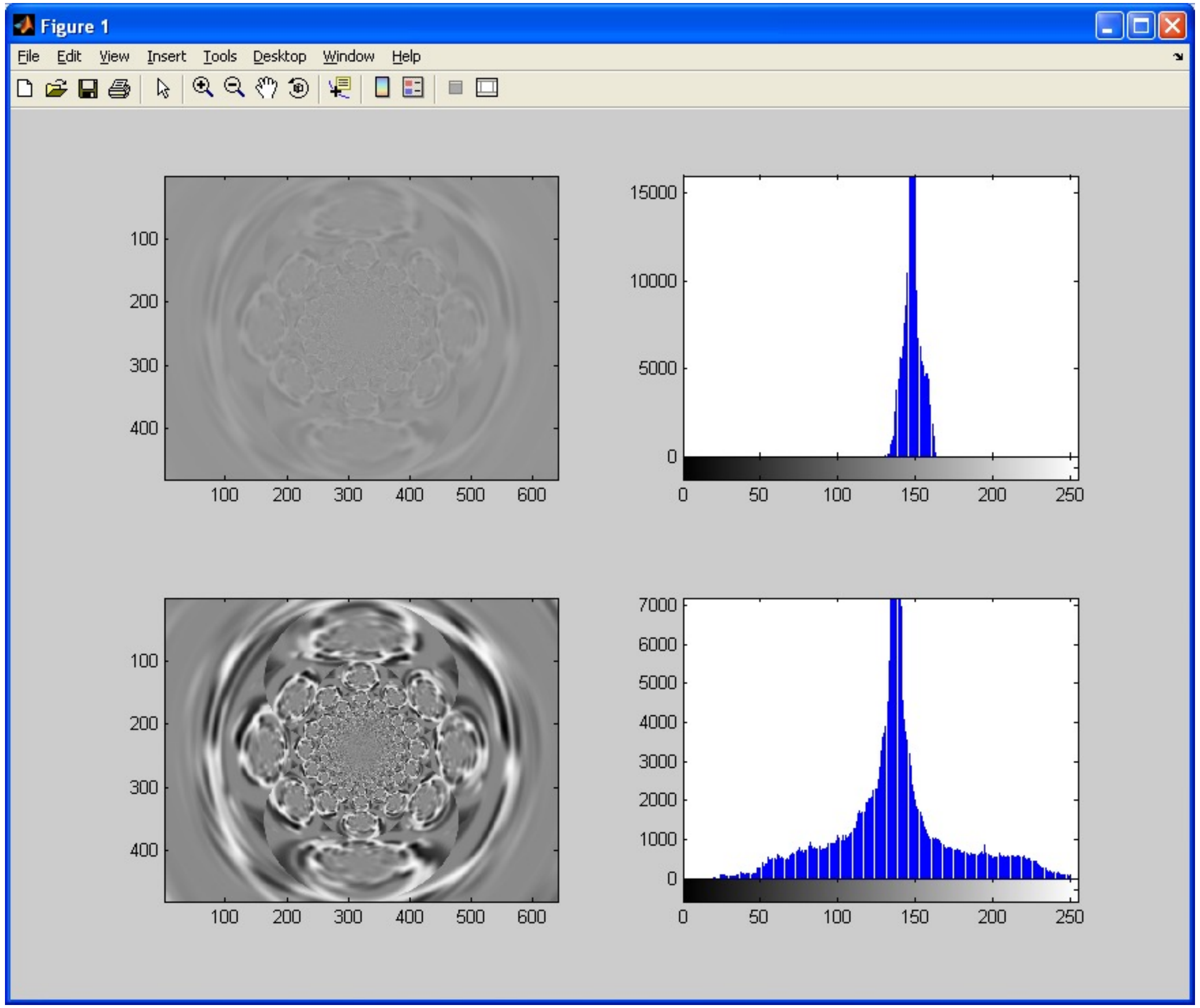
[Example2 & HW_Prob3]_Perceptron_batch_vs_SGD

- Replace the batch-based training part with SGD training

  - i.e., update should happen after each data point

- Write down your observation, e.g., noisy vs. smooth ?
- Write down why it is based on your intuition.

Data Normalization



Example of Image normalization

https://www.giassa.net/?page_id=472

Multi-Layer Perceptron for MNIST (Example 2)

- Data loading
- Normalization
- How label and image data looks

https://github.com/iam-mhaseeb/Multi-Layer-Perceptron-MNIST-with-PyTorch/blob/master/mnist_mlp_exercise.ipynb