BACK PROPAGATION

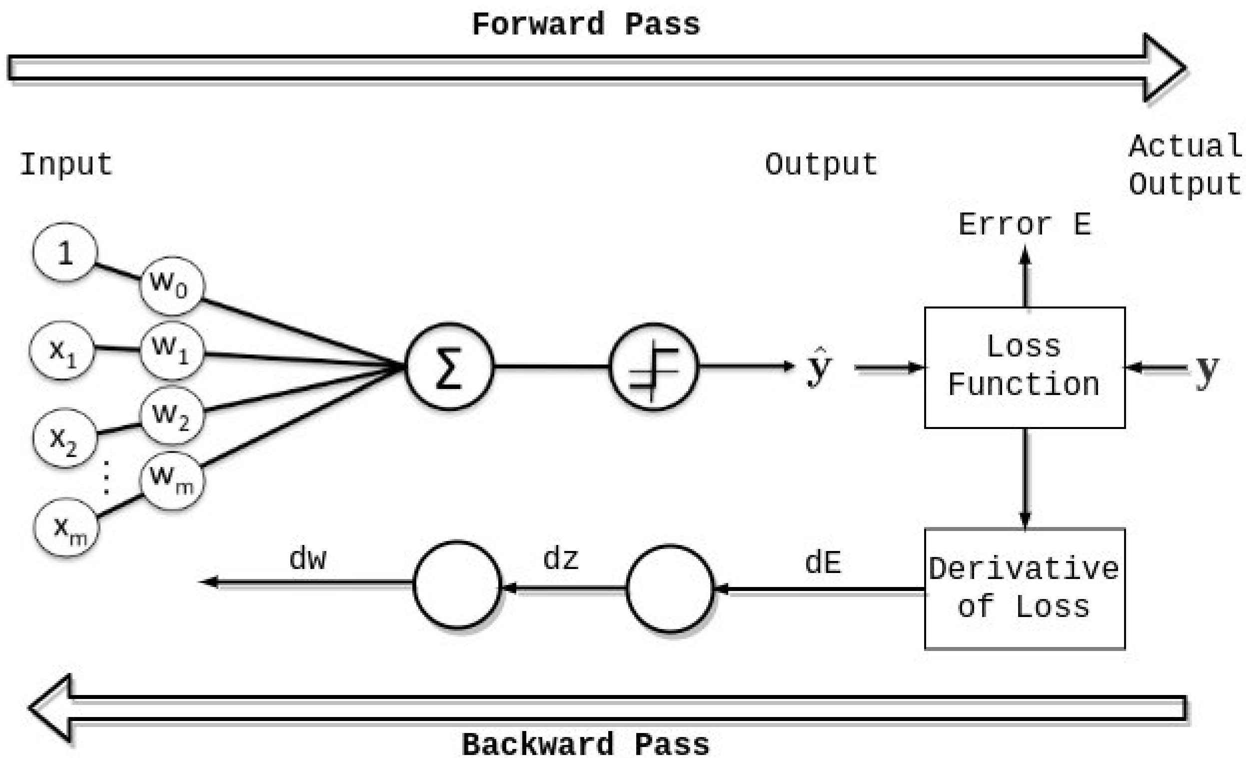
MUKESH KUMAR

WHAT IS LOSS FUNCTION?

Definition

- Backpropagation, short for "backward propagation of errors," is a fundamental algorithm used in training artificial neural networks.

- It is a supervised learning technique that adjusts the weights of the network to minimize the error between the predicted output and the actual target output.



Forward Pass
Input
Output
Actual
Output
Error E
1
w_0
x_1
w_1
x_2
w_2
w_m
x_m
Σ
$\hat{y}$
Loss
Function
y
dw
dz
dE
Derivative
of Loss
Backward Pass

BackProp Algorithm

1. **Forward Pass**:
   – The input data is passed through the network layer by layer to compute the output. Each neuron performs a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.
2. **Loss Calculation**:
   – The loss function, such as Mean Squared Error (MSE) for regression or Cross-Entropy Loss for classification, computes the difference between the predicted output and the actual target values.

3. **Backward Pass (Backpropagation)**:
   - The error is propagated backward through the network to update the weights.

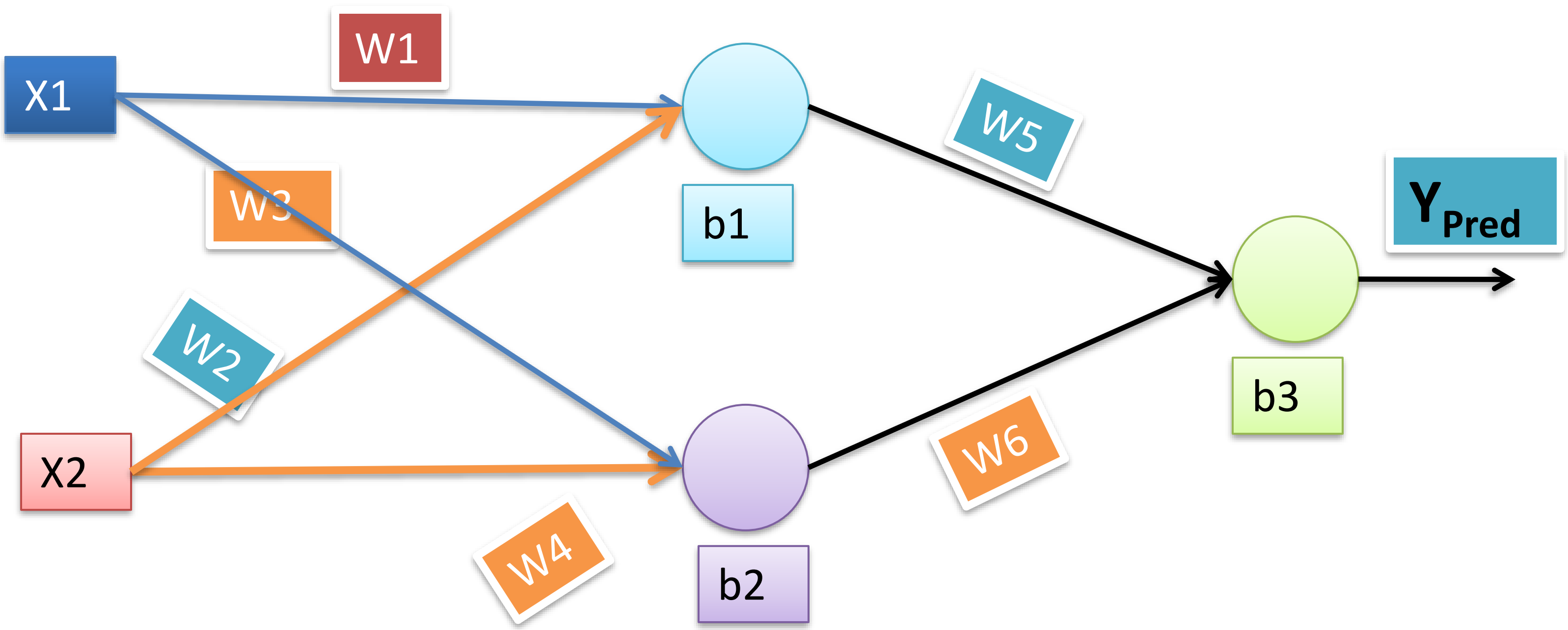
Detailed Backward Pass Process

1. **Calculate the Gradient of the Loss with Respect to the Output**:
   - Compute how the loss changes with respect to the output of each neuron in the output layer.
2. **Propagate the Gradient Backwards through the Network**:
   - Using the chain rule, compute the gradient of the loss with respect to the weights in each layer. This involves:
     - Calculating the gradient of the loss with respect to the output of the previous layer.
     - Multiplying this gradient by the derivative of the activation function to get the gradient with respect to the weighted input to the neurons.
     - Calculating the gradient with respect to the weights and biases.

Detailed Backward Pass Process

3. **Update the Weights and Biases**:
   - Adjust the weights and biases using a learning rate to minimize the loss. This can be done using optimization algorithms like Stochastic Gradient Descent (SGD), Adam, RMSprop, etc.

Weight	Shoulder	Height(Y)
68	14	5.7
89	18	6.0



STEP1:FORWARD PASS

Layer1

Forward Pass:

1. **Input Layer**: The inputs are $X1$ (Weight) and $X2$ (Shoulder).
2. **First Hidden Layer**: The first hidden layer has two neurons.

- Neuron 1:

$$Z1 = W1 \cdot X1 + W2 \cdot X2 + b1$$

- Neuron 2:

$$Z2 = W3 \cdot X1 + W4 \cdot X2 + b2$$

Layer1

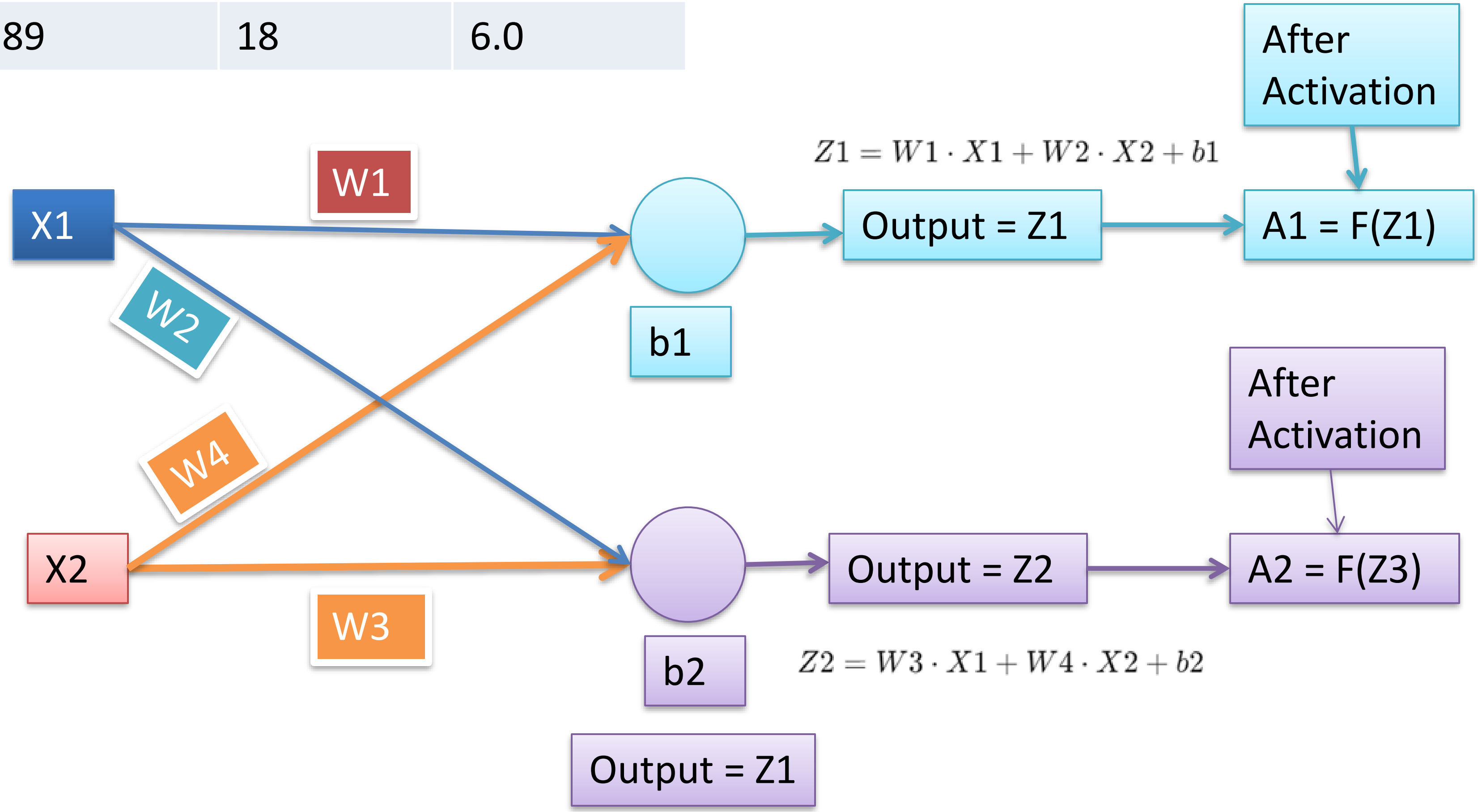
3. **Activation Function**: Apply an activation function (e.g., ReLU, sigmoid) to $Z1$ and $Z2$ to get $A1$ and $A2$.
   - Neuron 1:

$$A1 = f(Z1)$$

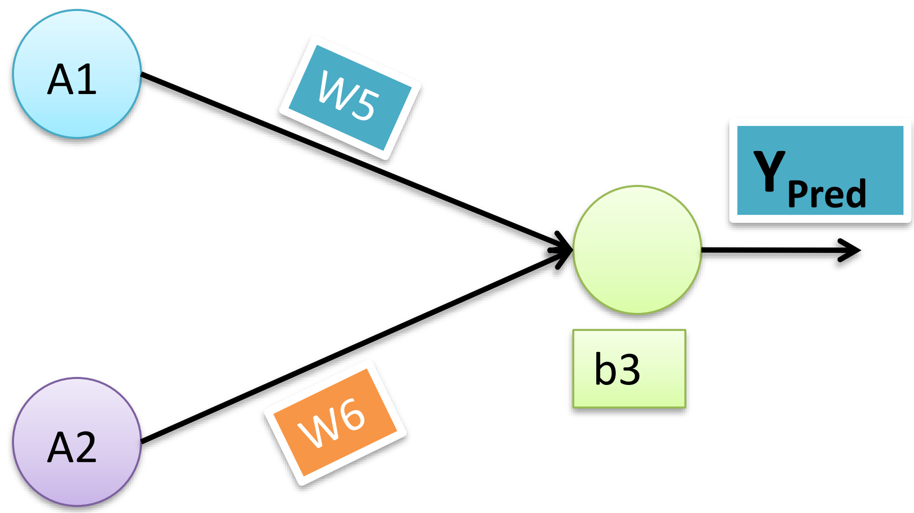
   - Neuron 2:

$$A2 = f(Z2)$$

Weight	Shoulder	Height(Y)
68	14	5.7
89	18	6.0



Weight	Shoulder	Height(Y)
68	14	5.7
89	18	6.0



$$Y_{pred} = W5 \cdot A1 + W6 \cdot A2 + b3$$

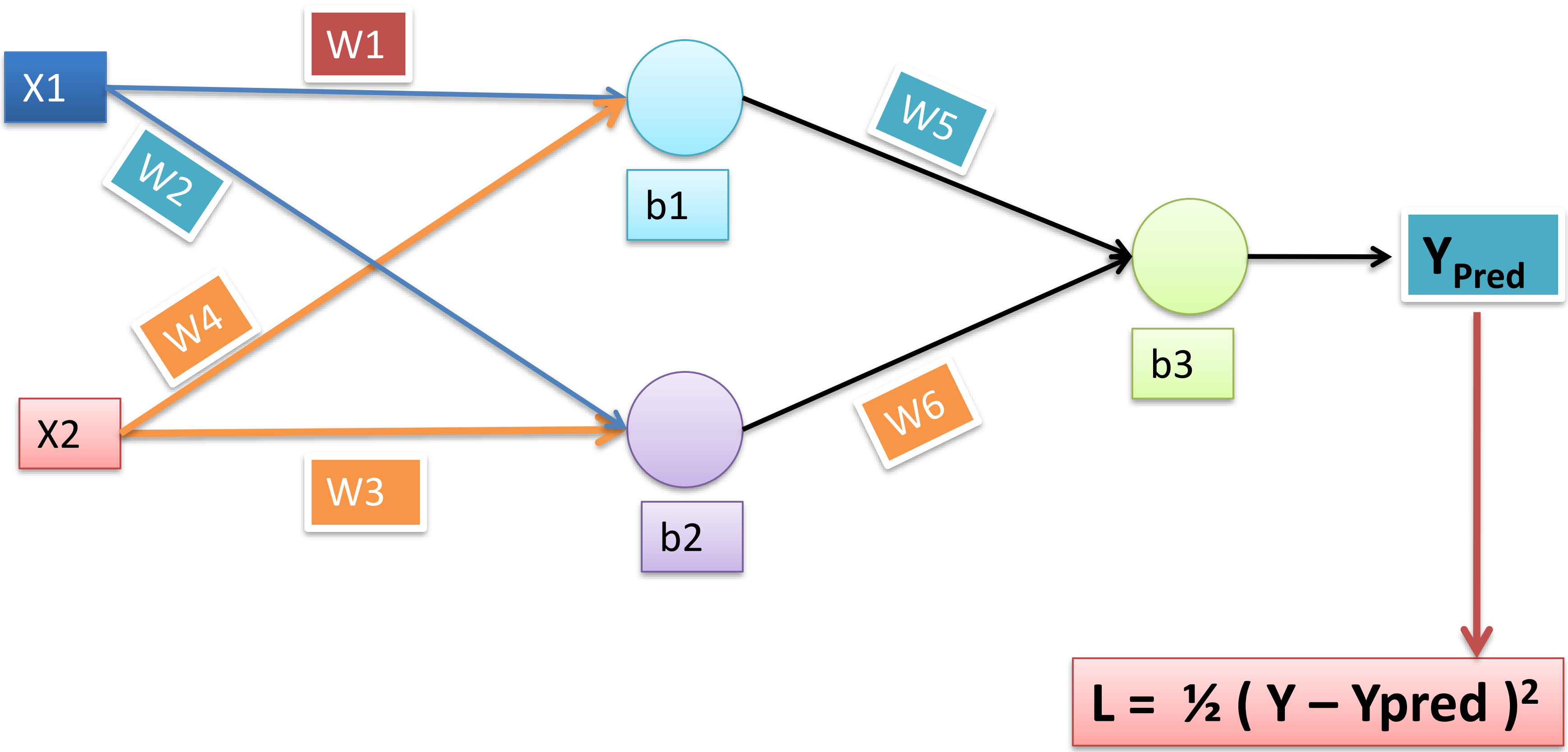
4. **Output Layer**: The output layer has one neuron that combines $A1$ and $A2$ to produce Y_{pred}.

- Output:

$$Y_{pred} = W5 \cdot A1 + W6 \cdot A2 + b3$$

BACKWARD PASS (BACKPROPAGATION):

Weight	Shoulder	Height(Y)
68	14	5.7
89	18	6.0



1. **Loss Function**: Calculate the loss function (e.g., Mean Squared Error) between the predicted output Y_{pred} and the actual output Y.

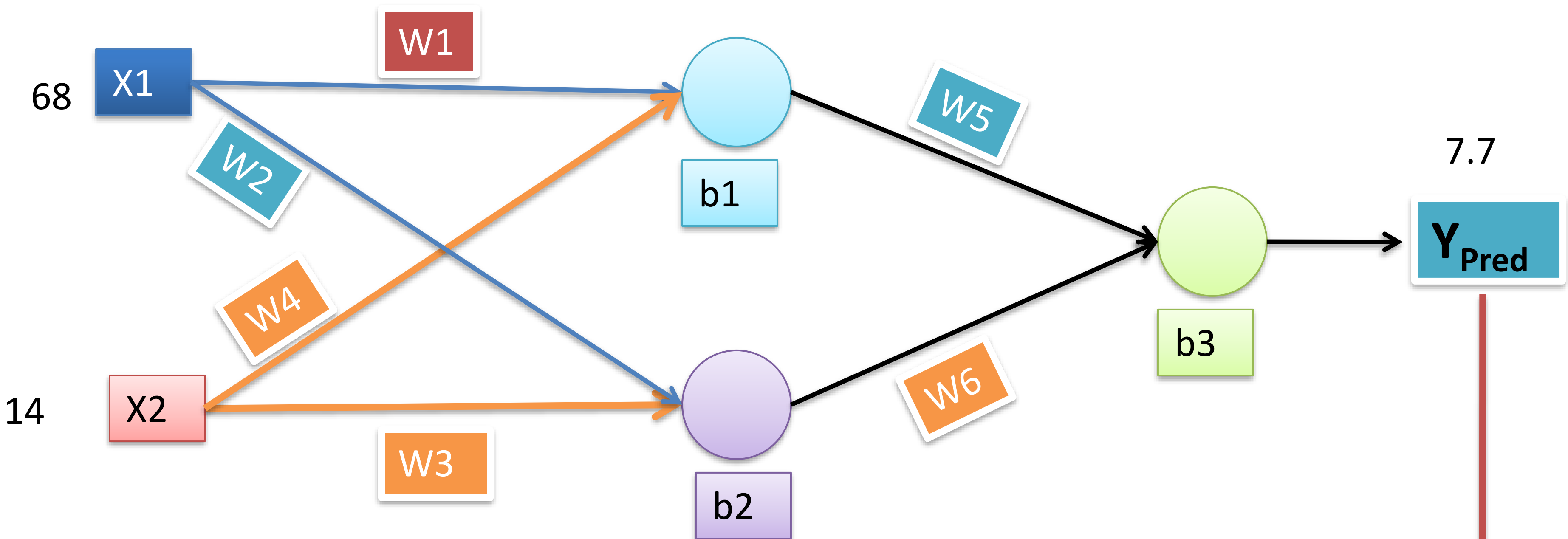
   - Loss:

$$L = \frac{1}{2}(Y - Y_{pred})^2$$

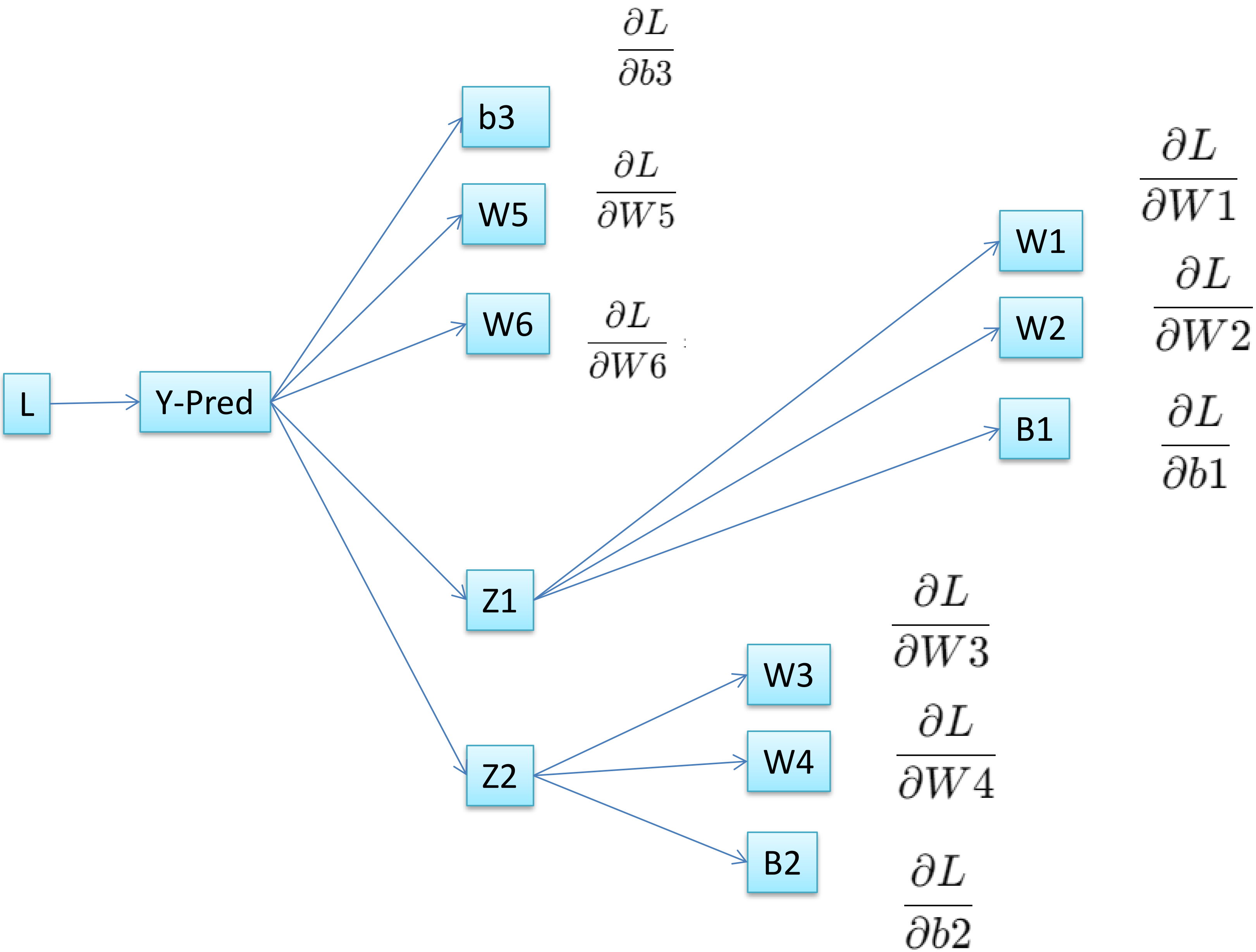
- Derivative
- Partial Differentiation

Explain with example record

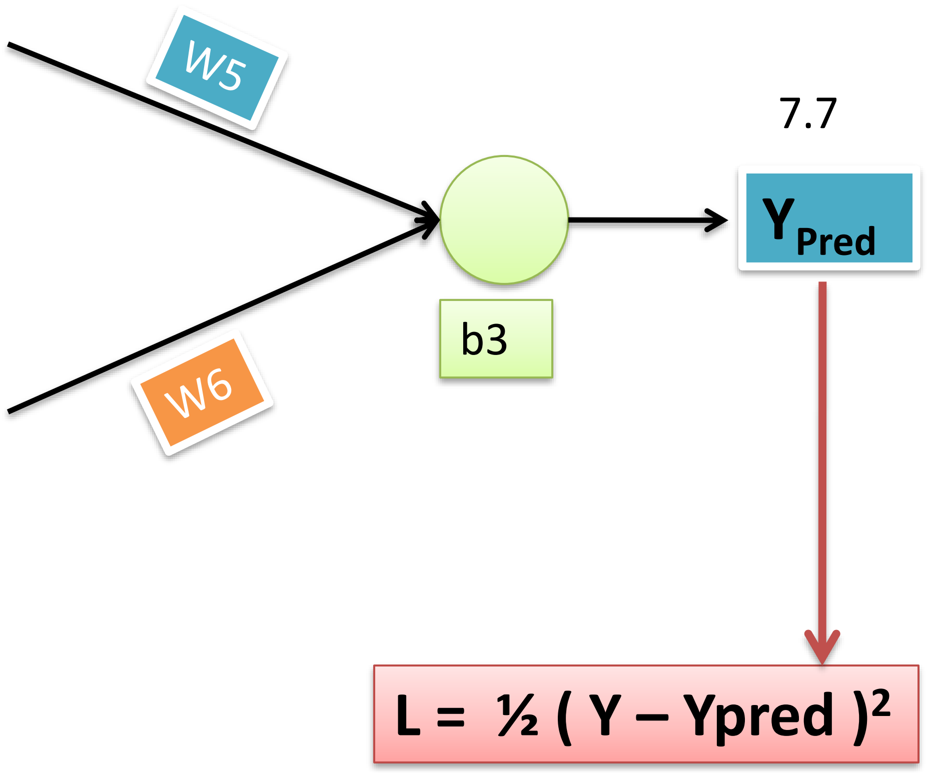
Weight	Shoulder	Height(Y)
68	14	5.7
89	18	6.0



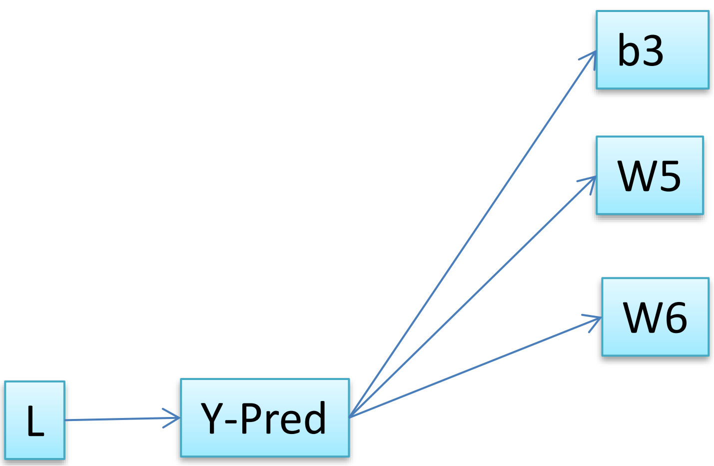
L = ½ (Y – Ypred)²


L
Y-Pred
b3
W5
W6
Z1
Z2
W1
W2
B1
W3
W4
B2
$\frac{\partial L}{\partial b3}$
$\frac{\partial L}{\partial W5}$
$\frac{\partial L}{\partial W6}$
$\frac{\partial L}{\partial W1}$
$\frac{\partial L}{\partial W2}$
$\frac{\partial L}{\partial b1}$
$\frac{\partial L}{\partial W3}$
$\frac{\partial L}{\partial W4}$
$\frac{\partial L}{\partial b2}$

Lets find gradients at output layer



Lets find gradients at output layer



$$\frac{\partial L}{\partial b3}$$

$$\frac{\partial L}{\partial W5}$$

$$\frac{\partial L}{\partial W6}$$

2. **Output Layer**: Calculate the gradient of the loss with respect to the weights $W5$ and $W6$, and bias $b3$.

$$\boxed{Y_{pred} = W5 \cdot A1 + W6 \cdot A2 + b3}$$

- Gradient w.r.t $W5$:

$$\frac{\partial L}{\partial W5} = \frac{\partial L}{\partial Y_{pred}} \cdot \frac{\partial Y_{pred}}{\partial W5} = (Y_{pred} - Y) \cdot A1$$

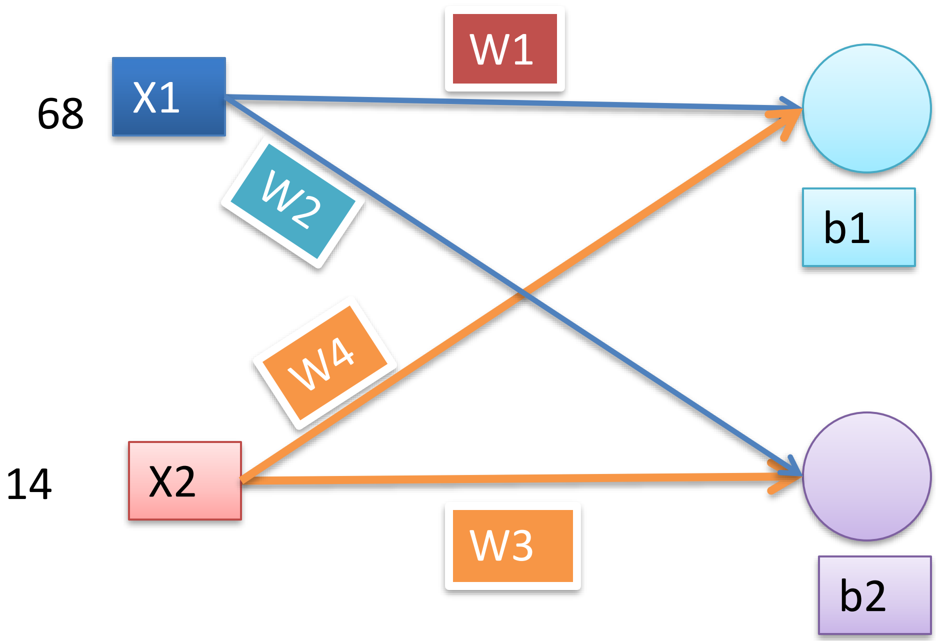
- Gradient w.r.t $W6$:

$$\frac{\partial L}{\partial W6} = (Y_{pred} - Y) \cdot A2$$

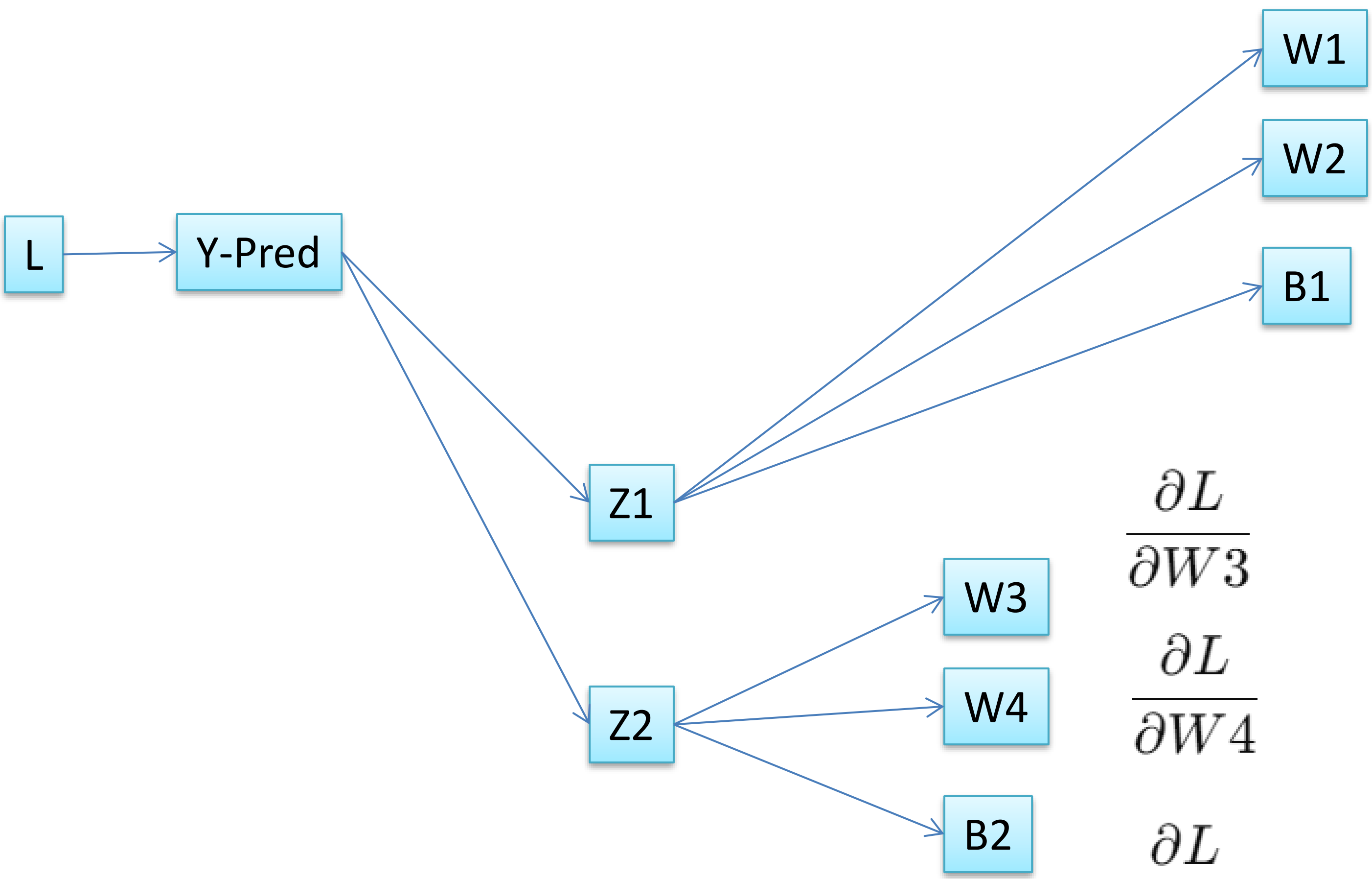
- Gradient w.r.t $b3$:

$$\frac{\partial L}{\partial b3} = (Y_{pred} - Y)$$

Lets find the gradients at first hidden layer



Lets find the gradients at first hidden layer



$$\frac{\partial L}{\partial W1}$$

$$\frac{\partial L}{\partial W2}$$

$$\frac{\partial L}{\partial b1}$$

$$\frac{\partial L}{\partial W3}$$

$$\frac{\partial L}{\partial W4}$$

$$\frac{\partial L}{\partial b2}$$

3. **First Hidden Layer**: Calculate the gradient of the loss with respect to the weights $W1, W2, W3, W4$ and biases $b1, b2$.

$\boxed{Y_{pred} = W5 \cdot A1 + W6 \cdot A2 + b3}$ $\boxed{Z1 = W1 \cdot X1 + W2 \cdot X2 + b1}$

- For Neuron 1:
  - Gradient w.r.t $Z1$:

$$\frac{\partial L}{\partial Z1} = \frac{\partial L}{\partial Y_{pred}} \cdot \frac{\partial Y_{pred}}{\partial A1} \cdot \frac{\partial A1}{\partial Z1} = (Y_{pred} - Y) \cdot W5 \cdot f'(Z1)$$

  - Gradient w.r.t $W1$:

$$\frac{\partial L}{\partial W1} = \frac{\partial L}{\partial Z1} \cdot \frac{\partial Z1}{\partial W1} = (Y_{pred} - Y) \cdot W5 \cdot f'(Z1) \cdot X1$$

  - Gradient w.r.t $W2$:

$$\frac{\partial L}{\partial W2} = (Y_{pred} - Y) \cdot W5 \cdot f'(Z1) \cdot X2$$

  - Gradient w.r.t $b1$:

$$\frac{\partial L}{\partial b1} = (Y_{pred} - Y) \cdot W5 \cdot f'(Z1)$$

- For Neuron 2: $\boxed{Y_{pred} = W5 \cdot A1 + W6 \cdot A2 + b3}$ $\boxed{Z2 = W3 \cdot X1 + W4 \cdot X2 + b2}$
  - Gradient w.r.t $Z2$:

$$\frac{\partial L}{\partial Z2} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2)$$

  - Gradient w.r.t $W3$:

$$\frac{\partial L}{\partial W3} = \frac{\partial L}{\partial Z2} \cdot \frac{\partial Z2}{\partial W3} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2) \cdot X1$$

  - Gradient w.r.t $W4$:

$$\frac{\partial L}{\partial W4} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2) \cdot X2$$

  - Gradient w.r.t $b2$:

$$\frac{\partial L}{\partial b2} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2)$$

- For Neuron 2:
  - Gradient w.r.t $Z2$:

$$\frac{\partial L}{\partial Z2} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2)$$

  - Gradient w.r.t $W3$:

$$\frac{\partial L}{\partial W3} = \frac{\partial L}{\partial Z2} \cdot \frac{\partial Z2}{\partial W3} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2) \cdot X1$$

  - Gradient w.r.t $W4$:

$$\frac{\partial L}{\partial W4} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2) \cdot X2$$

  - Gradient w.r.t $b2$:

$$\frac{\partial L}{\partial b2} = (Y_{pred} - Y) \cdot W6 \cdot f'(Z2)$$

4. **Update Weights and Biases**: Use the gradients to update the weights and biases using a learning rate η.

$$W5 = W5 - \eta \cdot \frac{\partial L}{\partial W5}$$

$$W6 = W6 - \eta \cdot \frac{\partial L}{\partial W6}$$

$$b3 = b3 - \eta \cdot \frac{\partial L}{\partial b3}$$

$$W1 = W1 - \eta \cdot \frac{\partial L}{\partial W1}$$

$$b1 = b1 - \eta \cdot \frac{\partial L}{\partial b1}$$

$$W2 = W2 - \eta \cdot \frac{\partial L}{\partial W2}$$

$$W3 = W3 - \eta \cdot \frac{\partial L}{\partial W3}$$

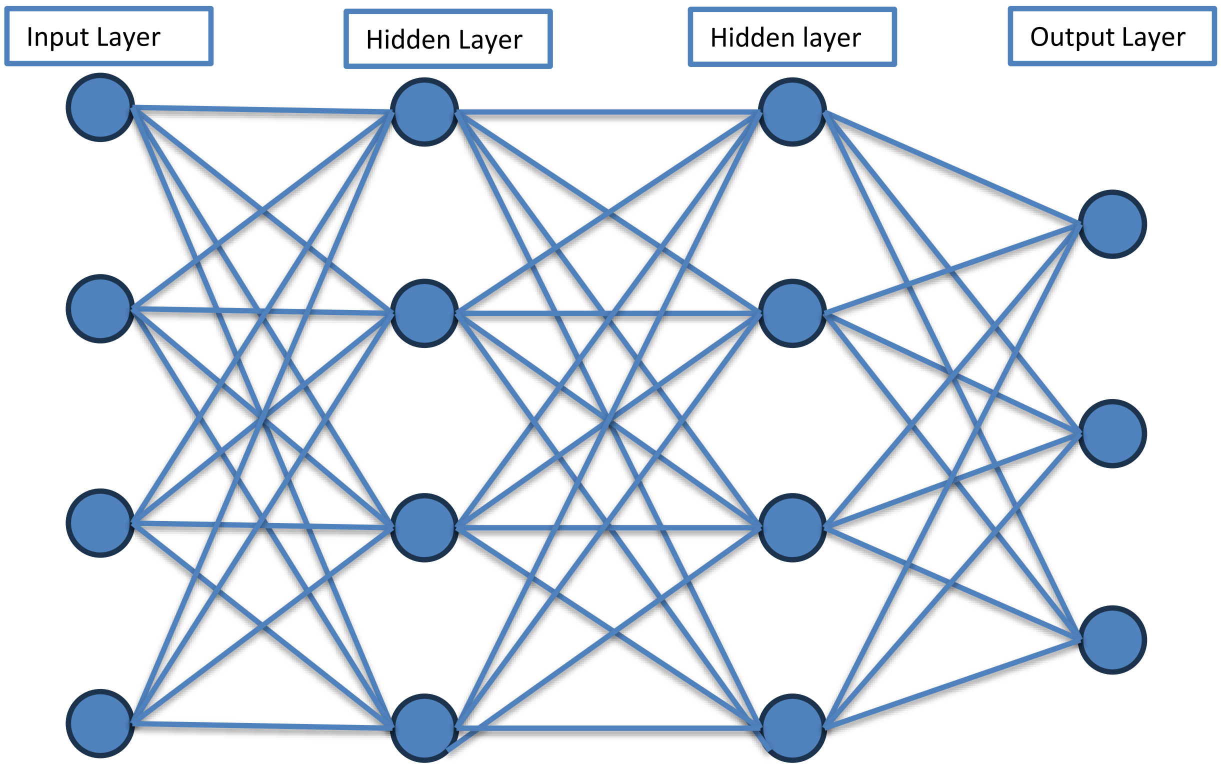
$$b2 = b2 - \eta \cdot \frac{\partial L}{\partial b2}$$

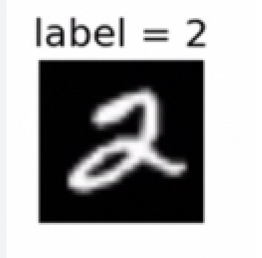
$$W4 = W4 - \eta \cdot \frac{\partial L}{\partial W4}$$

This process iterates for a number of epochs or until the loss converges to a minimum value, effectively training the neural network to predict the height Y_{pred} from weight $X1$ and shoulder $X2$

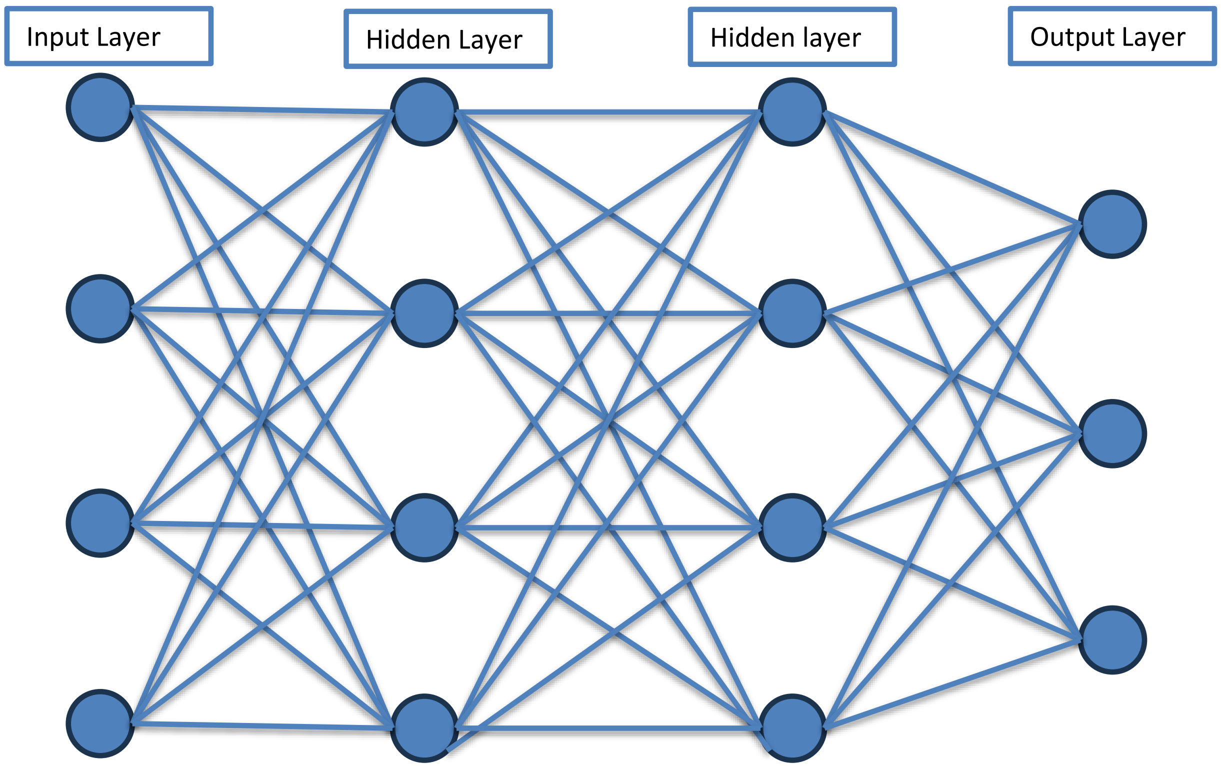
Back Propagation Intuition

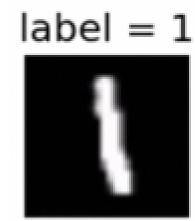
Let's build a NN that identifies 2 digits

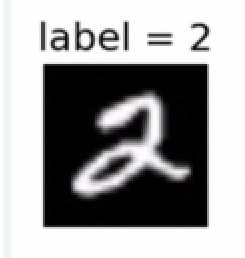




Let's build a NN that identifies 2 digits



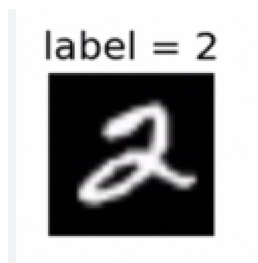




There are three classes in output hence we have 3 neurons in output layer one for each class

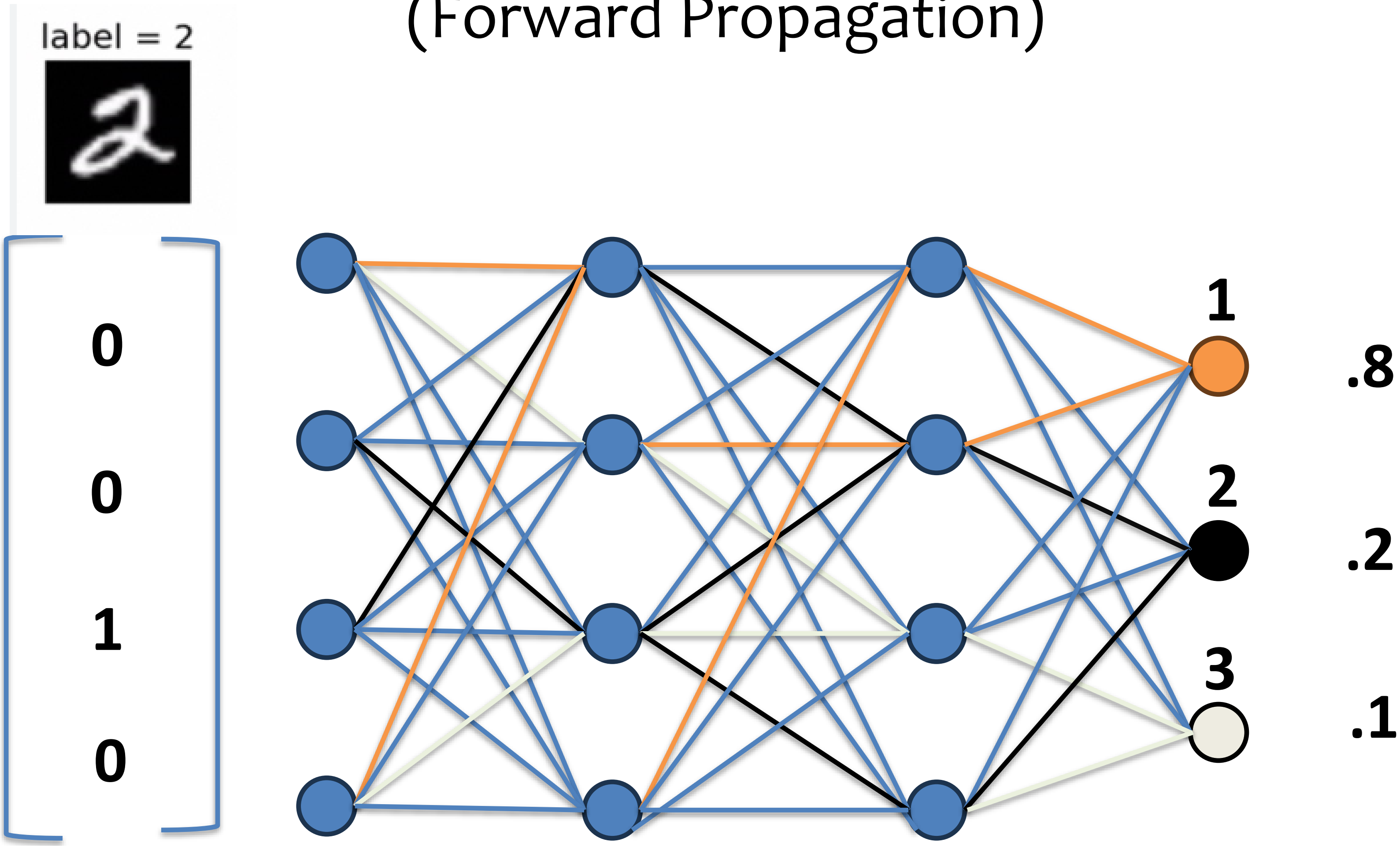
Start Training

- Let's feed first record to the network



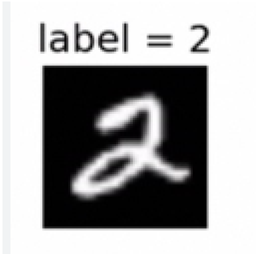
Feeding digit ‘2’ in the network

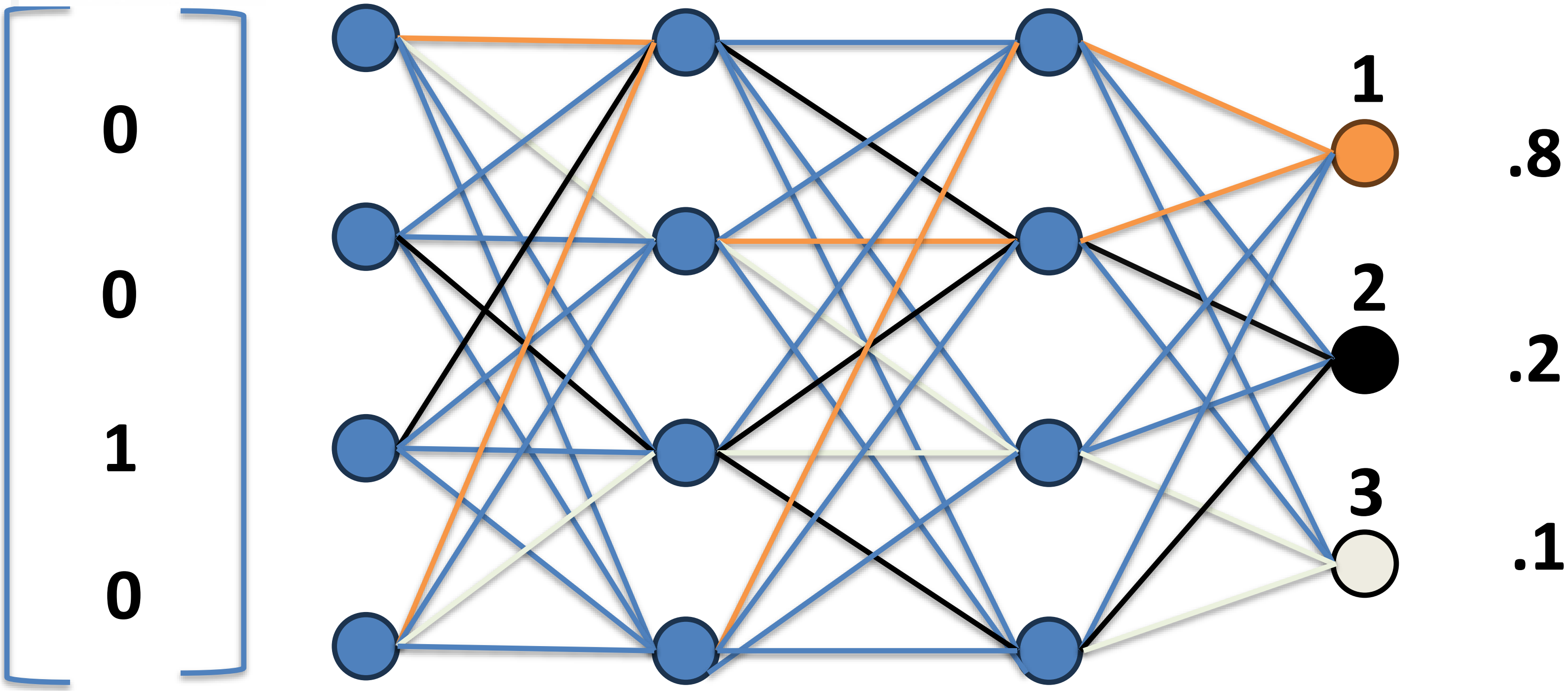
(Forward Propagation)



Model doesn’t understand anything so it spits out random numbers

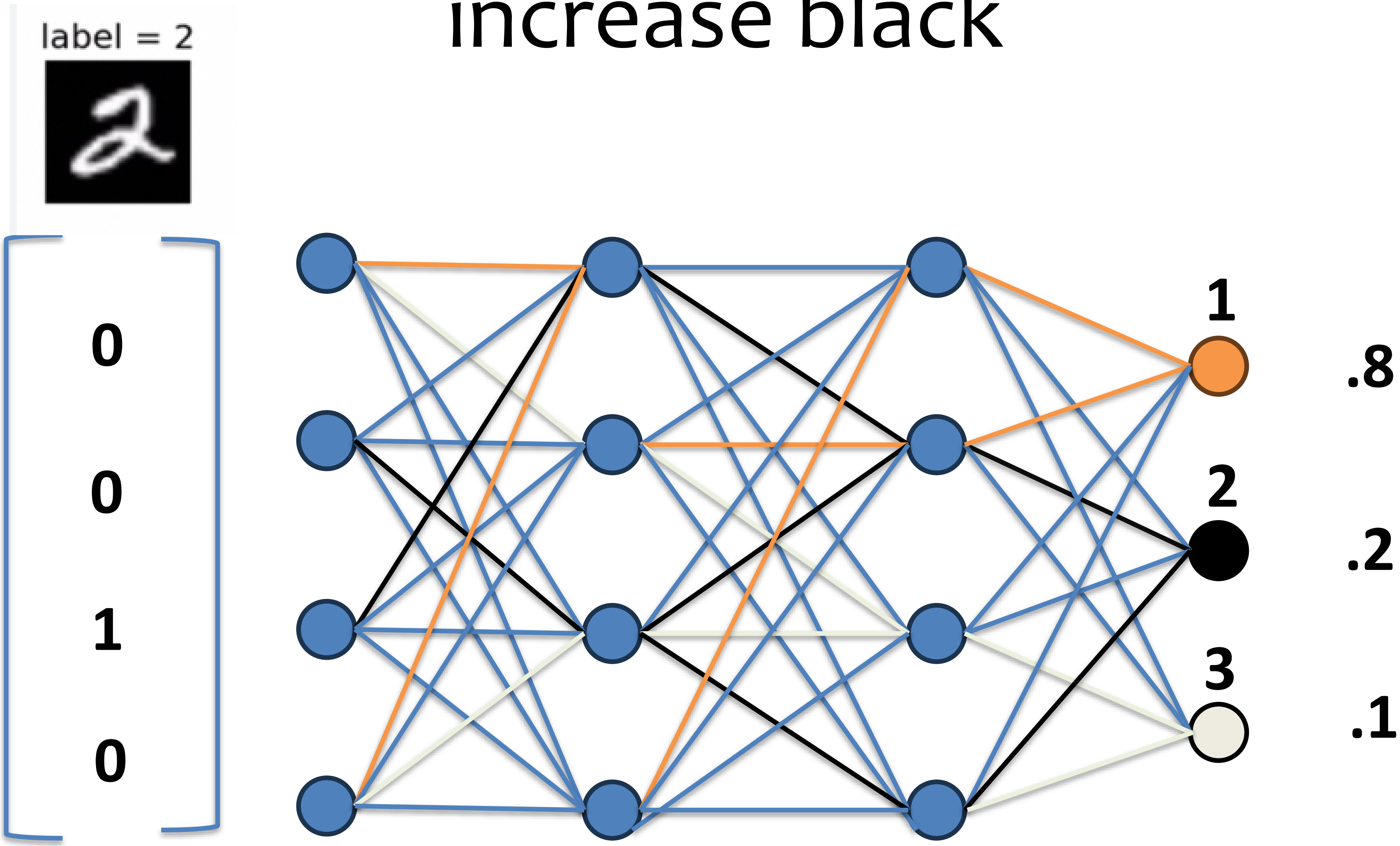
Feeding 2 in the network
Forward Propagation





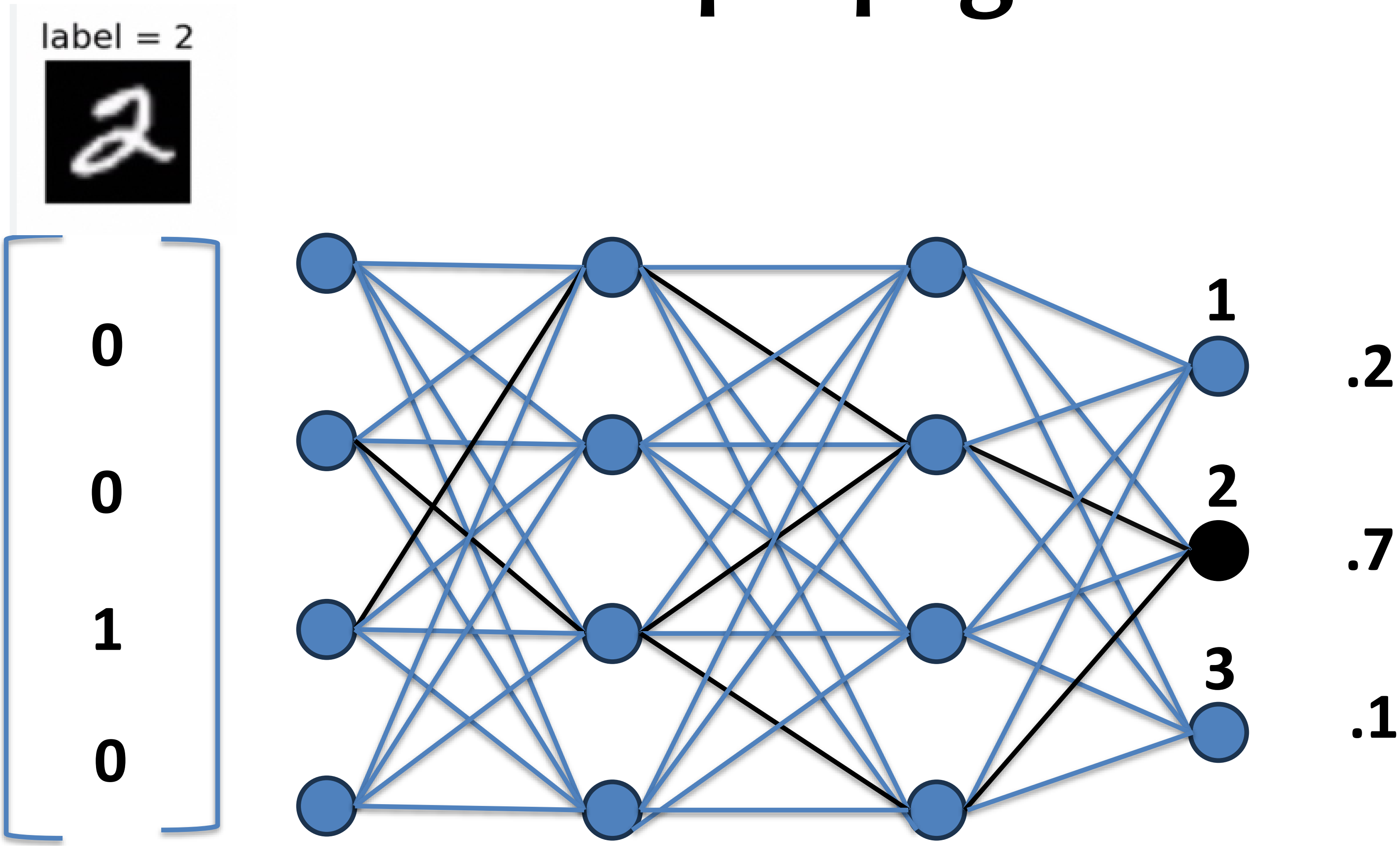
Right now, as per the output Model says it's a 1 not 2

Reduce orange and white and increase black



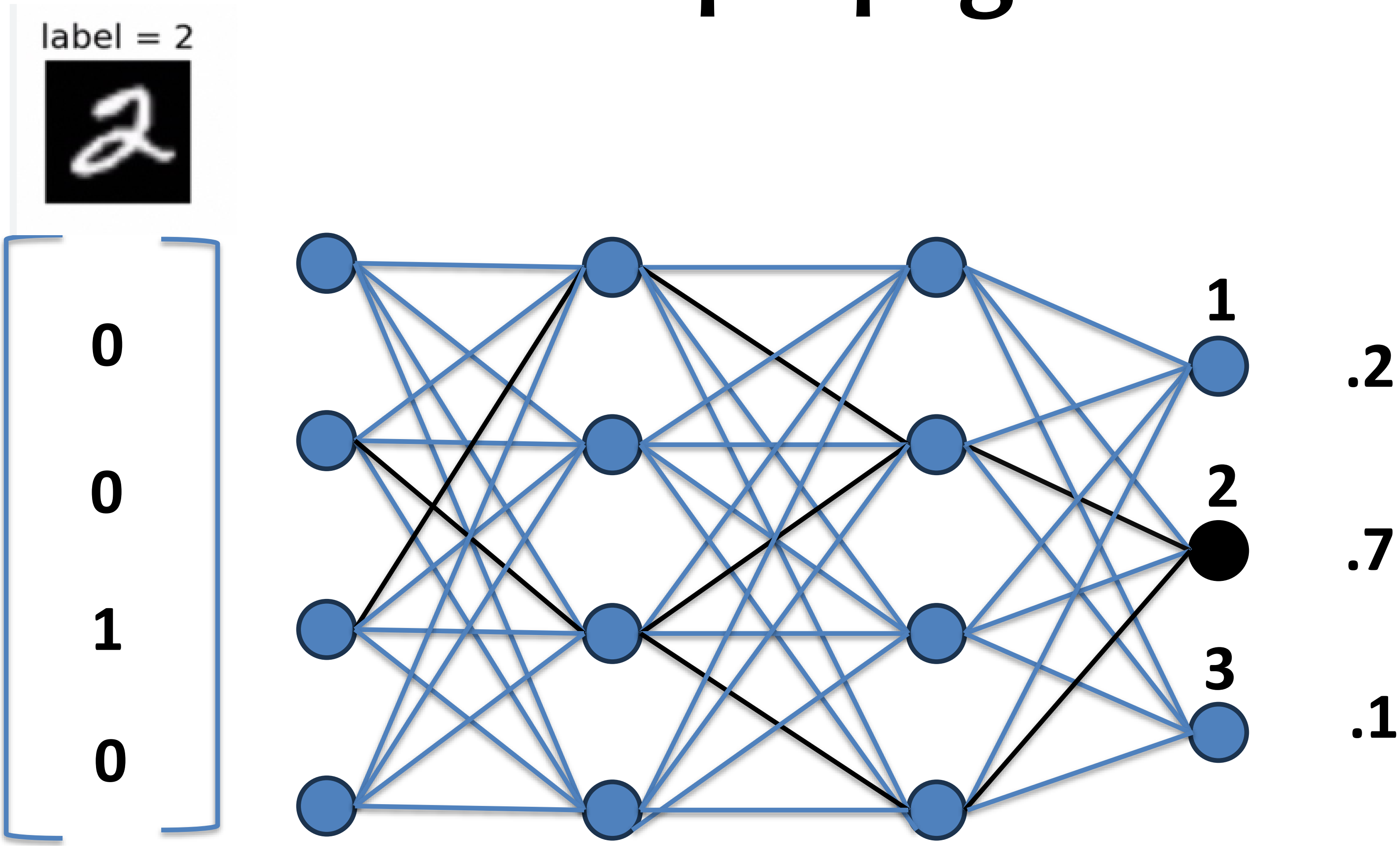
During back prop network adjusts weights to correct the output

After Back propagation



Model adjust the weights such that only the neuron corresponding to output 2 has highest output

After Back propagation



This is repeated for all the records in the training data , then model recognizes all the letters correctly