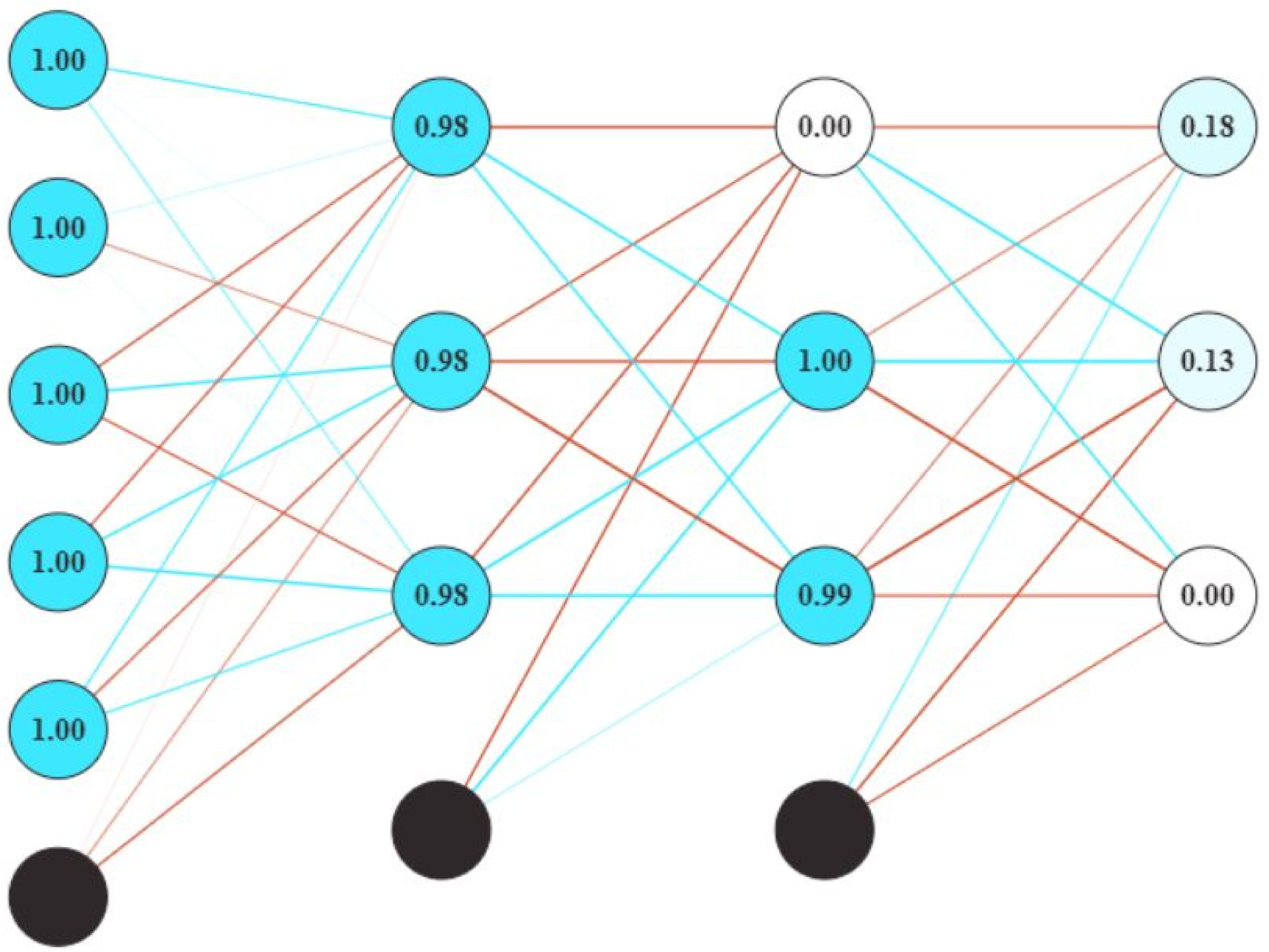
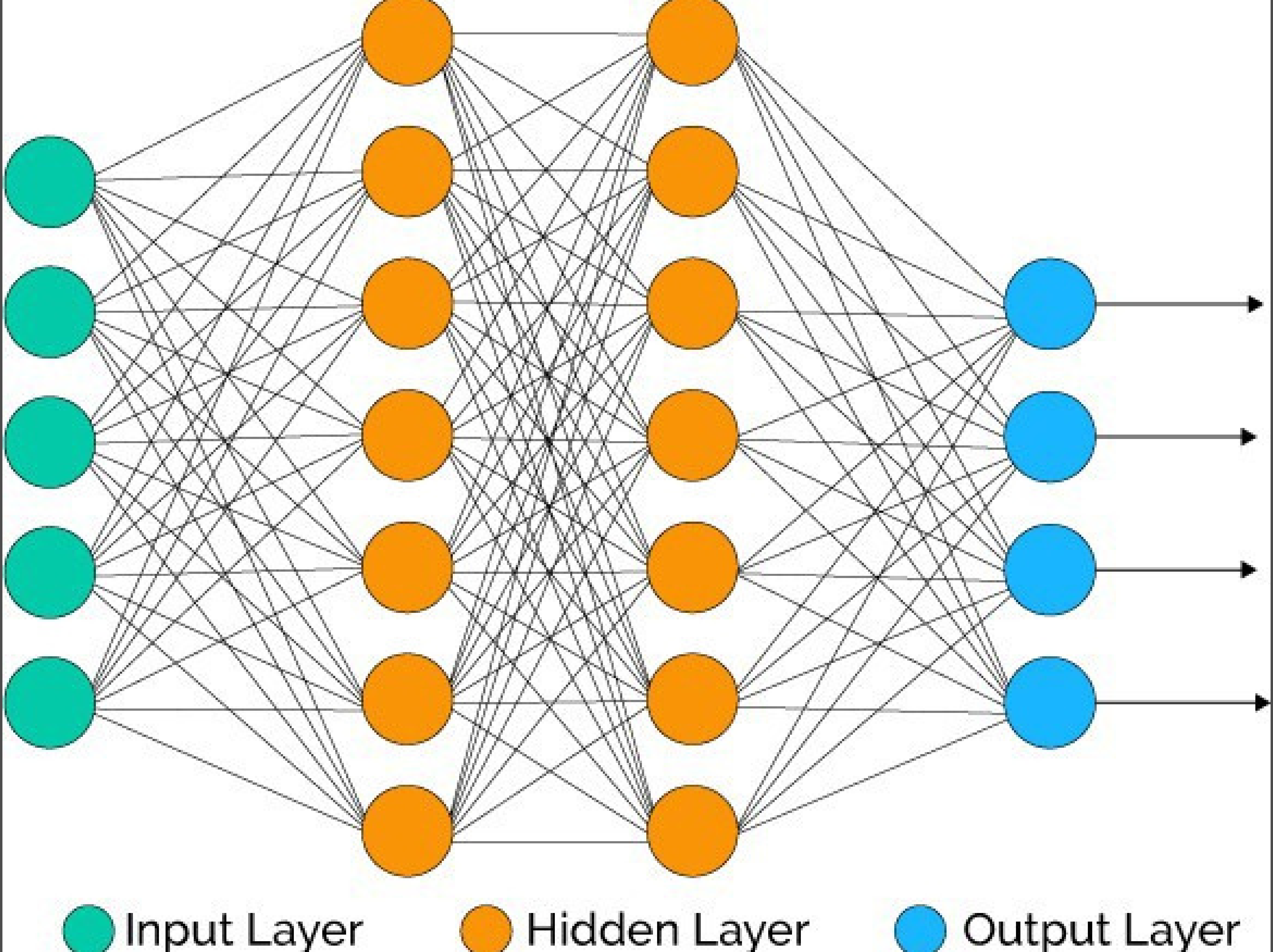

1.00
1.00
1.00
1.00
1.00
0.98
0.98
0.98
0.00
1.00
0.99
0.18
0.13
0.00

Symbol	Definition
L	Number of layers in the network
l	Layer index
j	Node index for layer l
k	Node index for layer $l-1$
y_j	The value of node j in the output layer L for a single training sample
C_0	Loss function of the network for a single training sample
$w_j^{(l)}$	The vector of weights connecting all nodes in layer $l-1$ to node j in layer l
$w_{jk}^{(l)}$	The weight that connects node k in layer $l-1$ to node j in layer l
$z_j^{(l)}$	The input for node j in layer l
$g^{(l)}$	The activation function used for layer l
$a_j^{(l)}$	The activation output of node j in layer l


Input Layer
Hidden Layer
Output Layer

ALGORITHM

STEP 1 : Initialize all the weights and biases in the network

STEP 2 : While termination condition is not satisfied

STEP 3 : For each training instance X

STEP 4 : **FORWARD PROPAGATE**

FORWARD PROPAGATION

$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at $Hidden_1$ layer*

$a^{(2)} = f(z^{(2)})$ *activation value at $Hidden_1$ layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at $Hidden_2$ layer*

$a^{(3)} = f(z^{(3)})$ *activation value at $Hidden_2$ layer*

$s = W^{(3)}a^{(3)}$ *Output layer*

STEP 5 : Calculate Cost (Error) Function

Loss C_0

Observe that the expression

$$\left(a_j^{(L)} - y_j\right)^2$$

is the squared difference of the activation output and the desired output for node j in the output layer L. This can be interpreted as the loss for node j in layer L.

Therefore, to calculate the total loss, we should sum this squared difference for each node j in the output layer L.

This is expressed as

$$C_0 = \sum_{j=0}^{n-1} \left(a_j^{(L)} - y_j\right)^2.$$

Input $z_j^{(l)}$

We know that the input for node j in layer l is the weighted sum of the activation outputs from the previous layer $l-1$.

An individual term from the sum looks like this:

$$w_{jk}^{(l)} a_k^{(l-1)}$$

So, the input for a given node j in layer l is expressed as

$$z_j^{(l)} = \sum_{k=0}^{n-1} w_{jk}^{(l)} a_k^{(l-1)}.$$

Activation Output $a_j^{(l)}$

We know that the activation output of a given node j in layer l is the result of passing the input, $z_j^{(l)}$, to whatever activation function we choose to use $g^{(l)}$.

Therefore, the activation output of node j in layer l is expressed as

$$a_j^{(l)} = g^{(l)}\left(z_j^{(l)}\right).$$

COMPOSITION OF FUNCTIONS

$$C_{0_j} = \left(a_j^{(L)} - y_j\right)^2.$$

We see that C_{0_j} is a function of the activation output of node j in layer L, and so we can express C_{0_j} as a function of $a_j^{(L)}$ as

$$C_{0_j}\left(a_j^{(L)}\right).$$

$$a_j^{(L)} = g^{(L)}\left(z_j^{(L)}\right).$$

$$z_j^{(L)}\left(w_j^{(L)}\right).$$

Composition of Functions

$$C_{0_j} = C_{0_j}\left(a_j^{(L)}\left(z_j^{(L)}\left(w_j^{(L)}\right)\right)\right).$$

STEP 5 : DERIVATE OF THE LOSS FUNCTION

$$\frac{\partial C_0}{\partial w_{12}^{(L)}}.$$

Since C_0 depends on $a_1^{(L)}$, and $a_1^{(L)}$ depends on $z_1^{(L)}$, and $z_1^{(L)}$ depends on $w_{12}^{(L)}$, the chain rule tells us that to differentiate C_0 with respect to $w_{12}^{(L)}$, we take the product of the derivatives of the composed function.

This is expressed as

$$\frac{\partial C_0}{\partial w_{12}^{(L)}} = \left(\frac{\partial C_0}{\partial a_1^{(L)}}\right)\left(\frac{\partial a_1^{(L)}}{\partial z_1^{(L)}}\right)\left(\frac{\partial z_1^{(L)}}{\partial w_{12}^{(L)}}\right).$$

STEP X : Update weights and bias

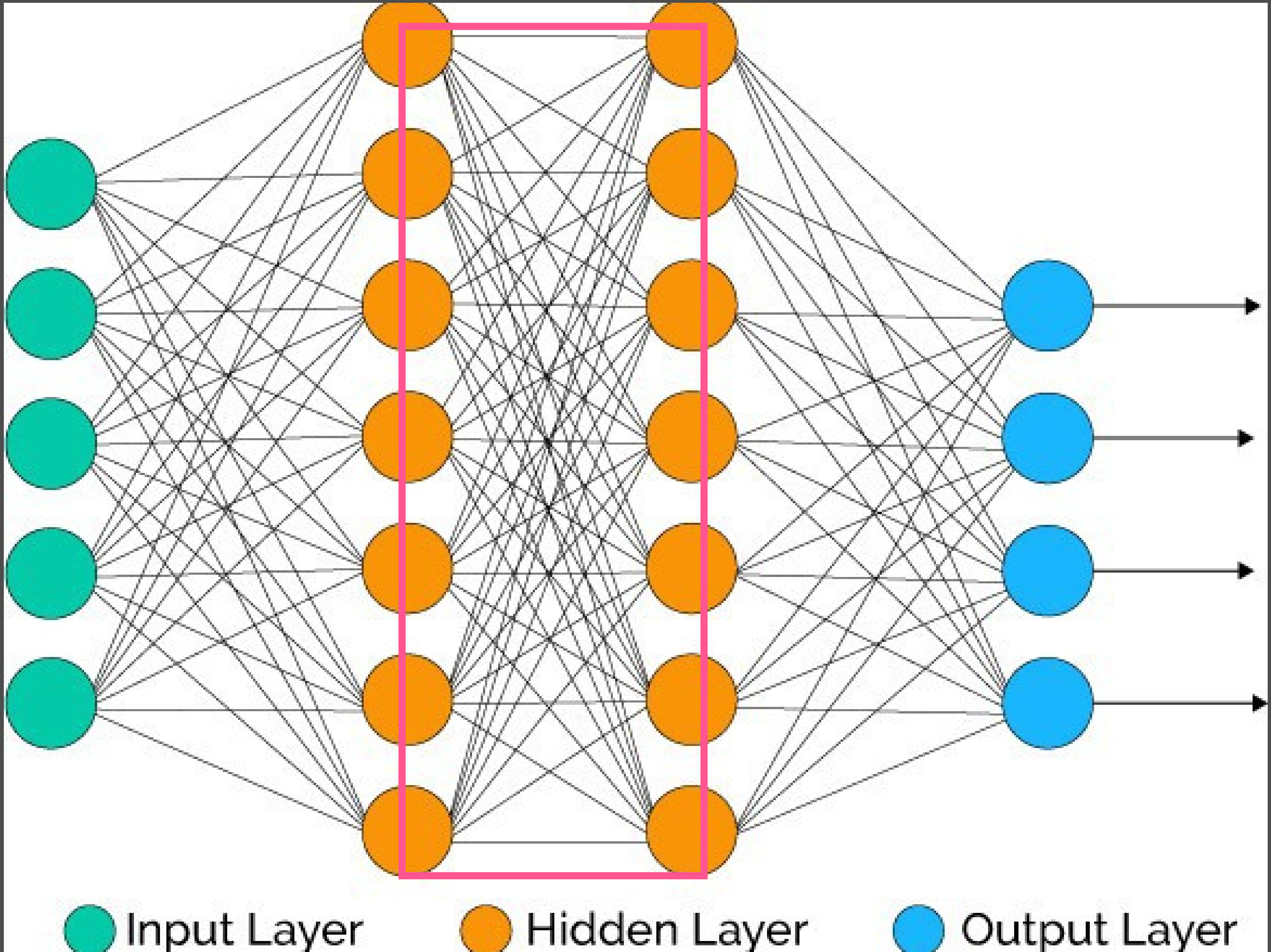
while (termination condition not met)

$$w := w - \epsilon \frac{\partial C}{\partial w}$$

$$b := b - \epsilon \frac{\partial C}{\partial b}$$

end

STEP 6 : SIMILARLY FIND LOSS AND DERIVATE FOR HIDDEN LAYERS



STEP 7 : Update weights and bias

$$while\ (termination\ condition\ not\ met)$$

$$w := w - \epsilon \frac{\partial C}{\partial w}$$

$$b := b - \epsilon \frac{\partial C}{\partial b}$$

$$end$$