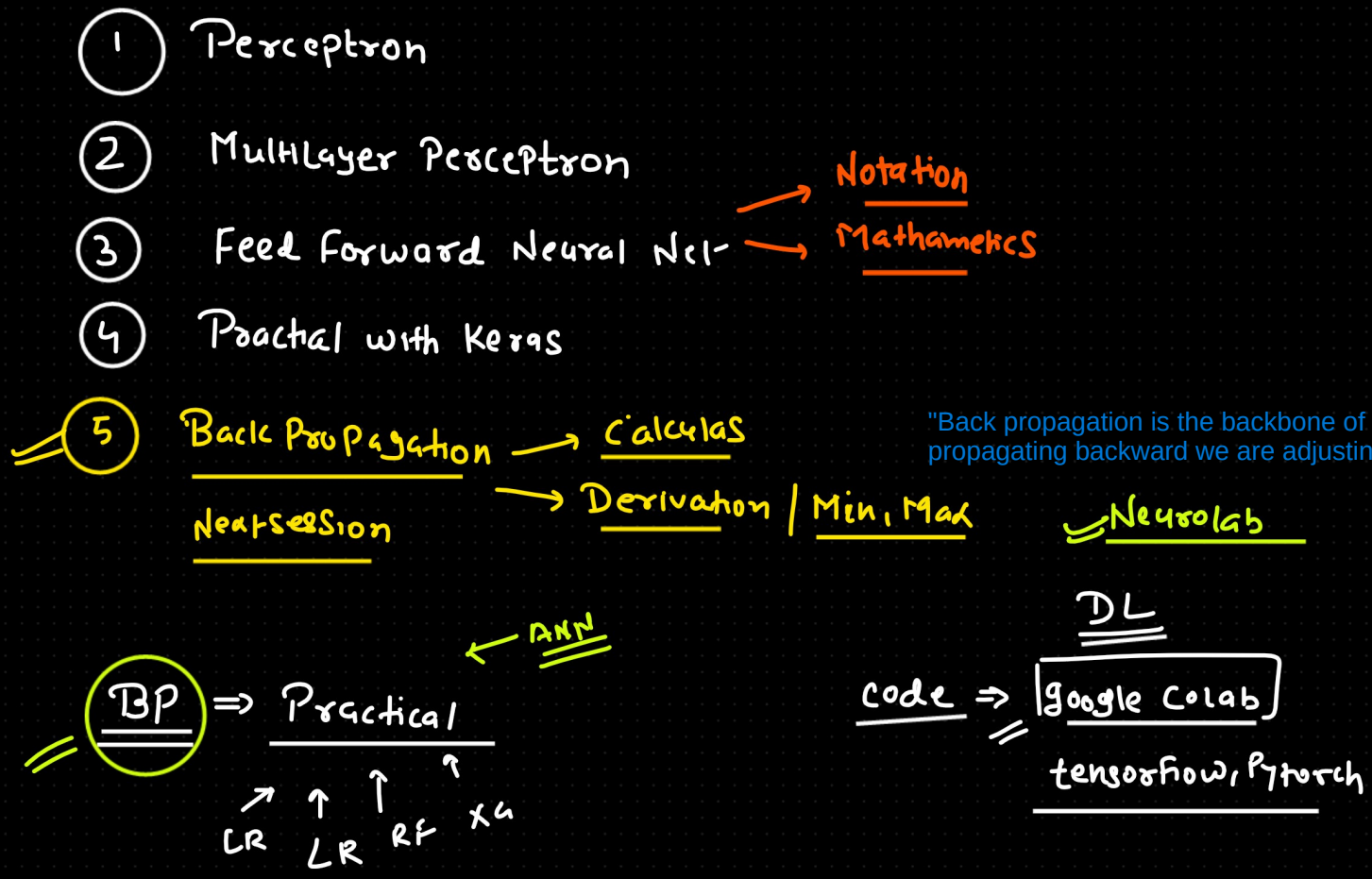
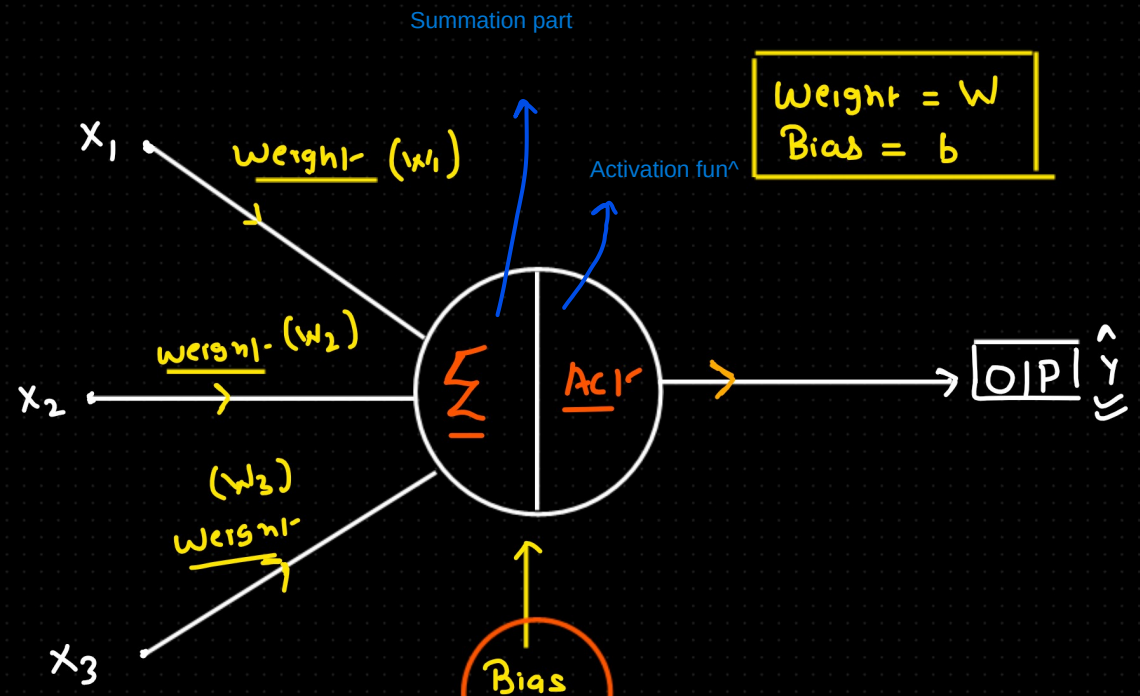
Agenda:



Tip:
Use Google Colab/Neuro Lab for implementing DL since in local system it may require some extra dependency in the form of GPU

Perceptron



(Summation part) Σ $x_1w_1 + x_2w_2 + x_3w_3 + b \Rightarrow W^TX + b$

Where W represents the vector of weights and X represents the vector of inputs

Passing the resultant of summation part to the activation function

Sigmoid fln $\Rightarrow$ Act^ $(W^TX + b) \Rightarrow$ O/P

$Y = mx + C$

LR $\Rightarrow$ $Y = m_1x_1 + m_2x_2 + m_3x_3 + C$

Logreg $\Rightarrow$ Sigmoid fln. $\Rightarrow \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(y)}}$

Acf fln

Types of activation fun^

1. Step fln
2. Sigmoid fln
3. tanh fln

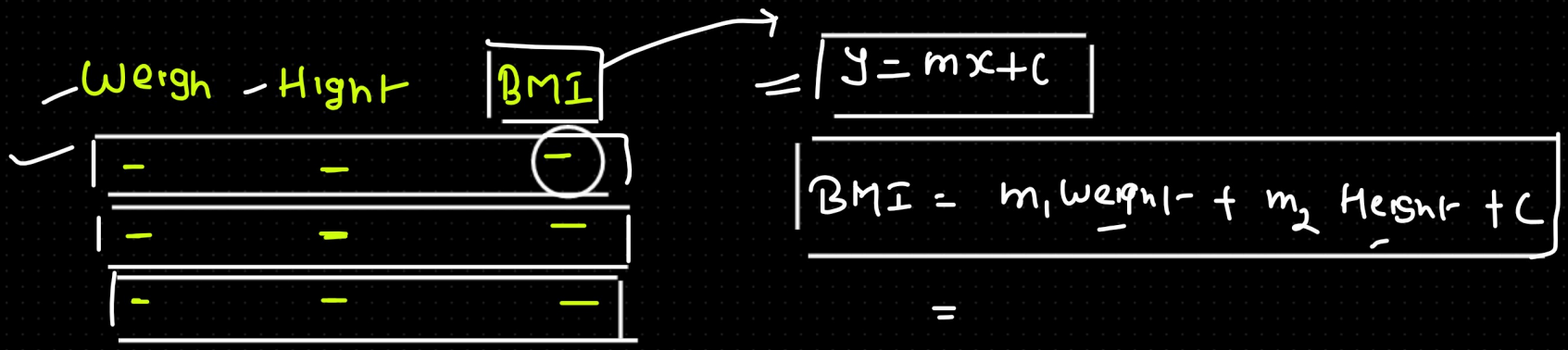
Multilayer Perceptron (MLP)

Input features are passed as nodes in the Input Layer.
Also, each neuron is called a node.

1. Input layer
2. Hidden layer
3. Output layer

Overall requirements to understand neural network better

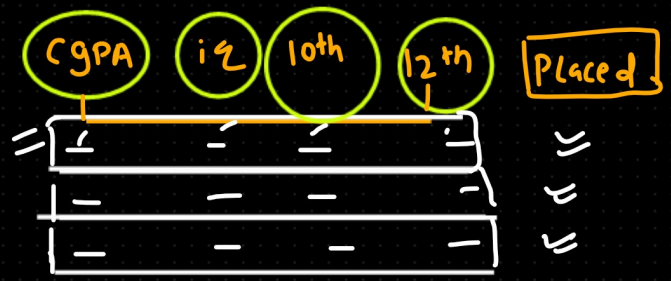
- Feed forward NN => Linear Algebra
  - Dot Product
  - Vector
  - Matrix
  - Matrix operations
- BackPropagation => Calculas
  - derivation, min-max, fln



Understanding the MLP structure through the following example:

Notations

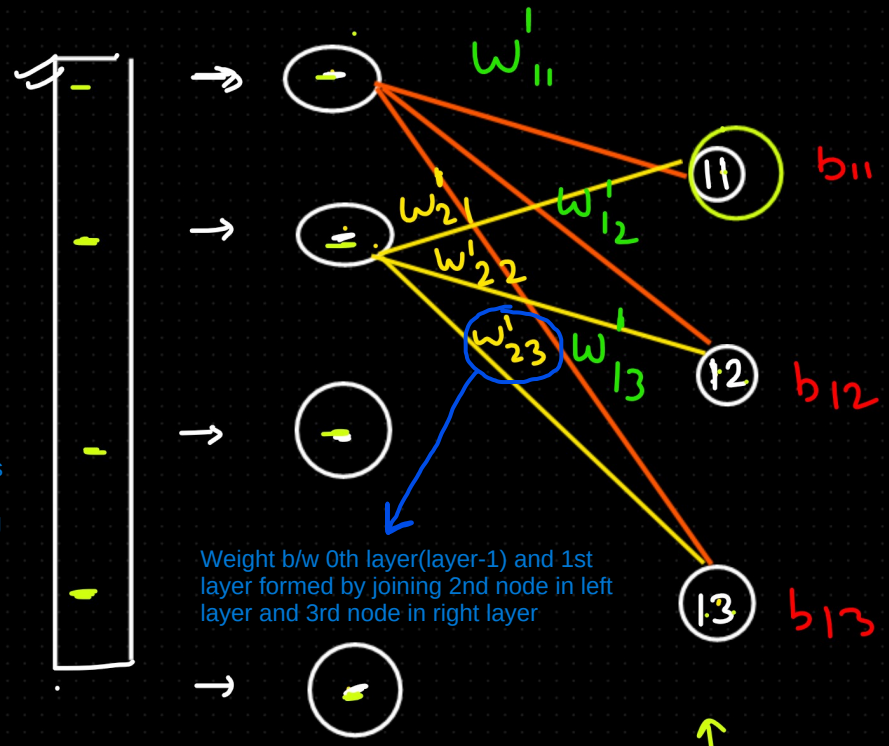
Data ⇒ M x N { M ⇒ rows, N ⇒ Columns } N = 4



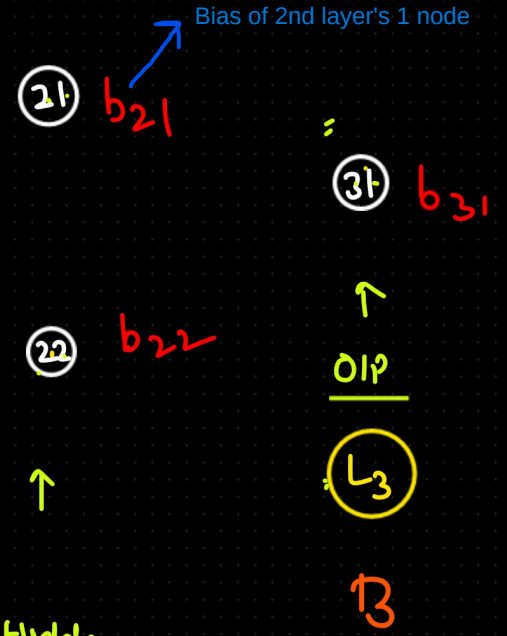
Complexity

Combination

- 4 independent features hence 4 nodes in the input layer.
- There can be as many number of hidden layers. This is a hyperparameter and an optimal value is picked up using hyperparameter tuning.
- Since we have only 1 dependent or target feature hence we have only one node in the output layer.



Weight b/w 0th layer(layer-1) and 1st layer formed by joining 2nd node in left layer and 3rd node in right layer



Input layer can be treated as 0th layer

Input
L_0

Hidden
L_1
B

Hidden
L_2
B

Edge = Weight

H_1 (Hidden Layer)

H_2 (Hidden Layer)

Weight = w_{ij}

bias = b_{ij}

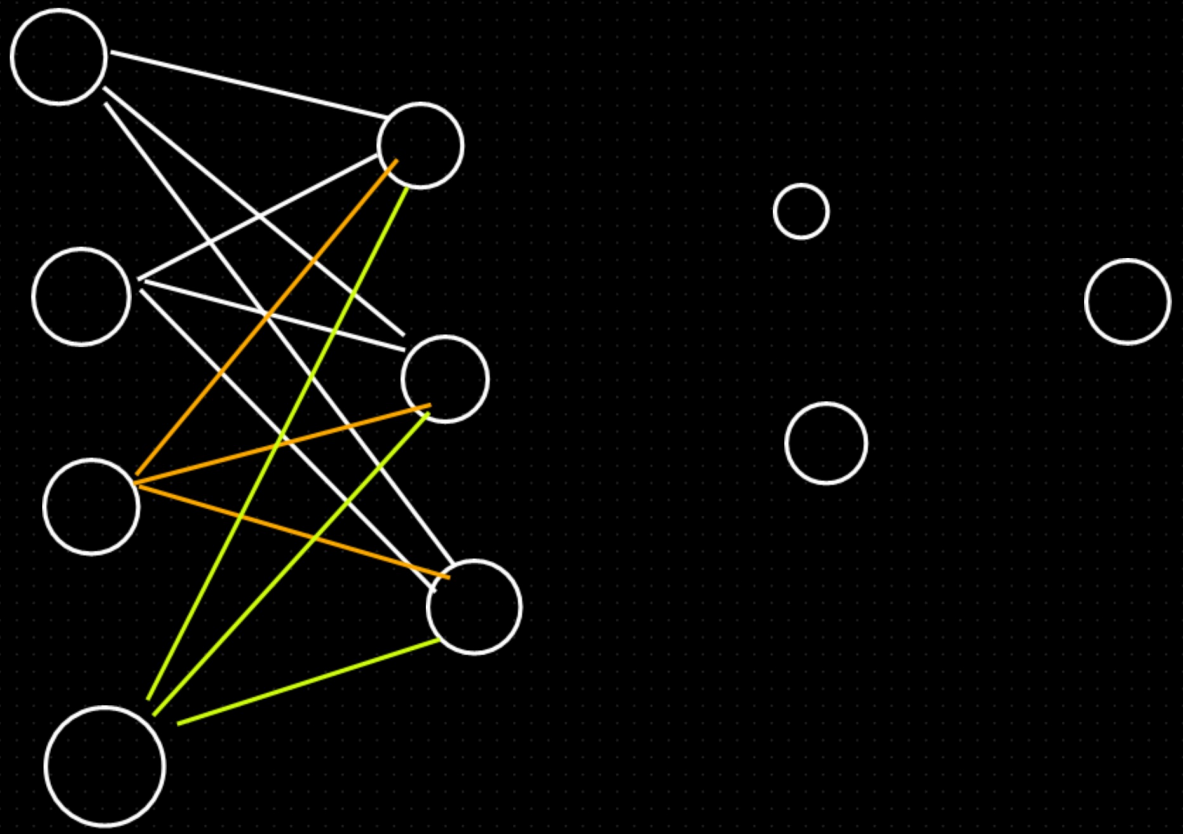
output = o_{ij}

{ i = layer, j = Node No }

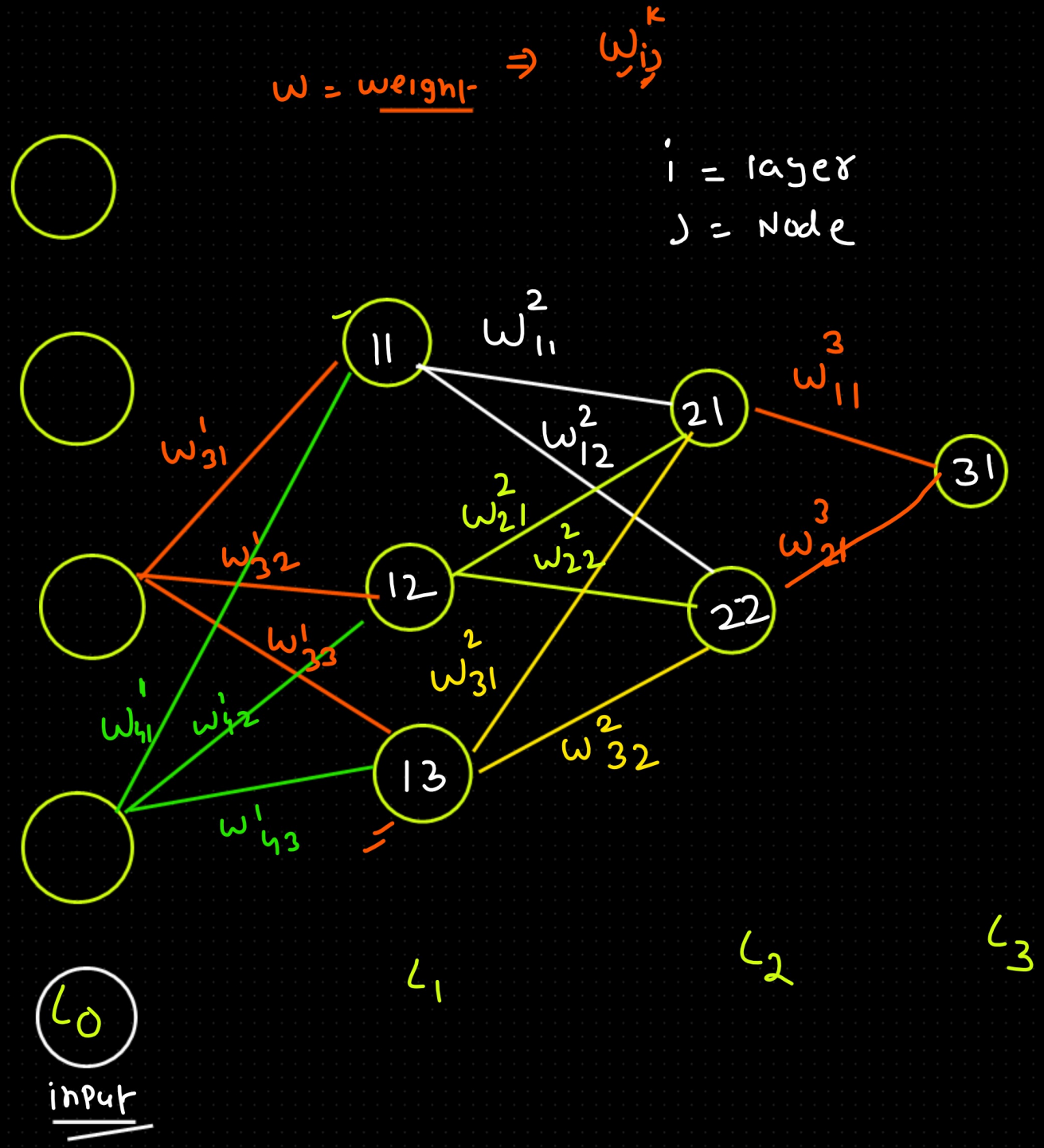
w_{ij}

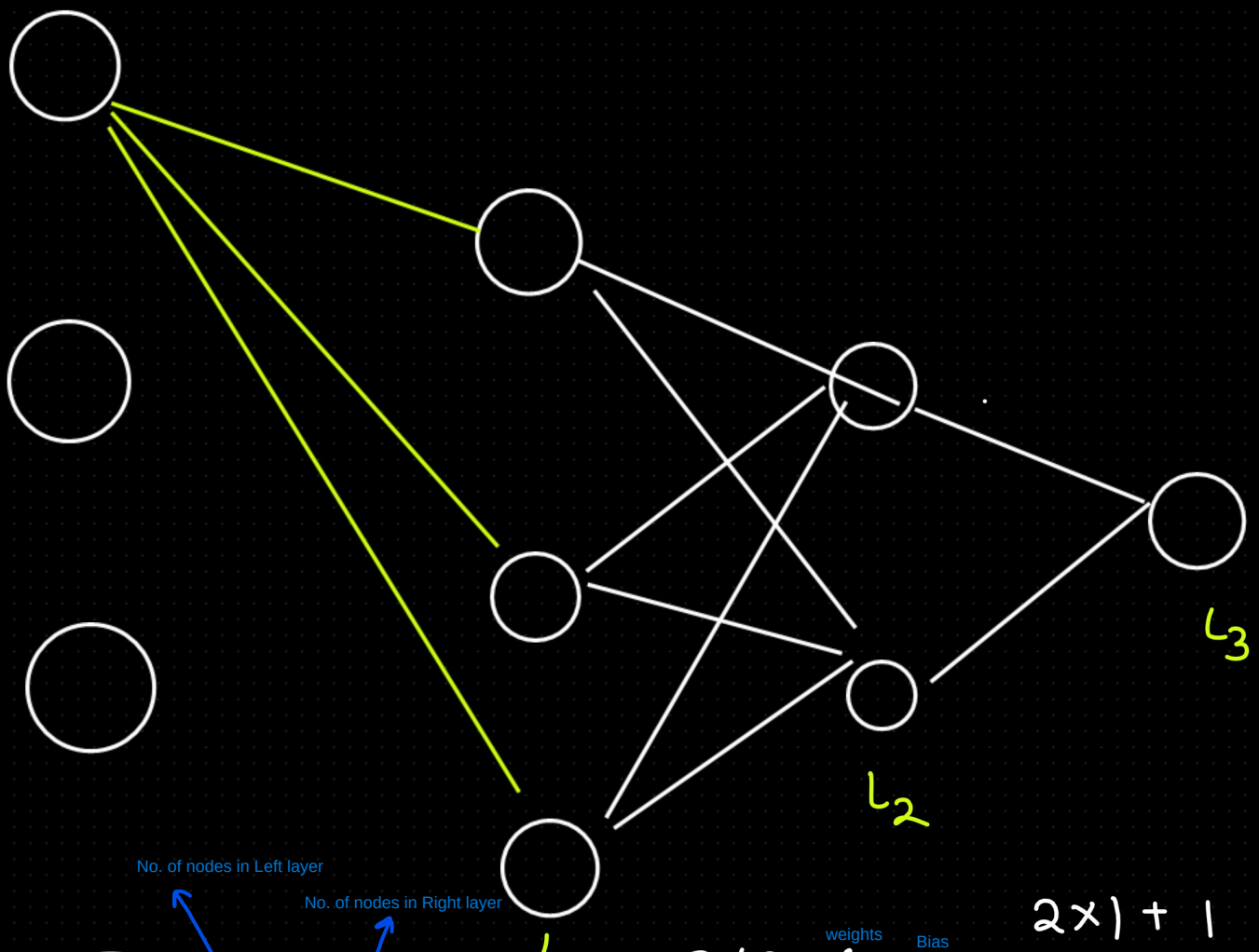
Trainable Parameter

Total number of Weights and Bias in the entire Neural Network is known as Trainable Parameter



$4 \times 3 = \boxed{12}$


w = weight
$\Rightarrow w_{ij}^{k}$
i = layer
j = Node
11
12
13
21
22
31
w_{11}^{2}
w_{12}^{2}
w_{21}^{2}
w_{22}^{2}
w_{31}^{2}
w_{32}^{2}
w_{11}^{3}
w_{21}^{3}
w_{31}^{1}
w_{32}^{1}
w_{33}^{1}
w_{41}^{1}
w_{42}^{1}
w_{43}^{1}
L_0
L_1
L_2
L_3
input



Trainable Parameter

$y = mx + c$

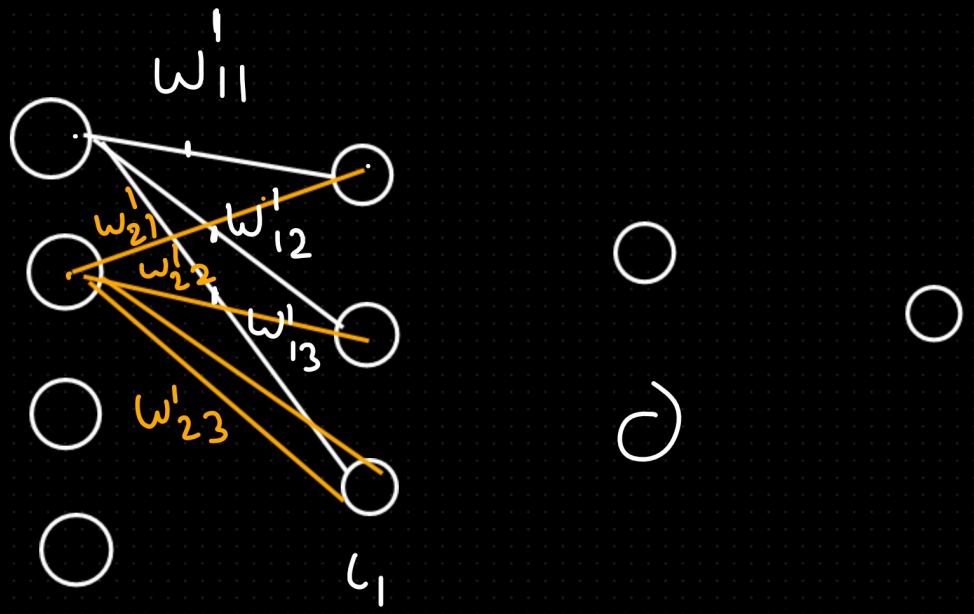
$y = m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4 + c$

G.D. ⇒ optimizer ⑤ $m_1\ m_2\ m_3\ m_4\ c$

↓

trainable

loss ↓

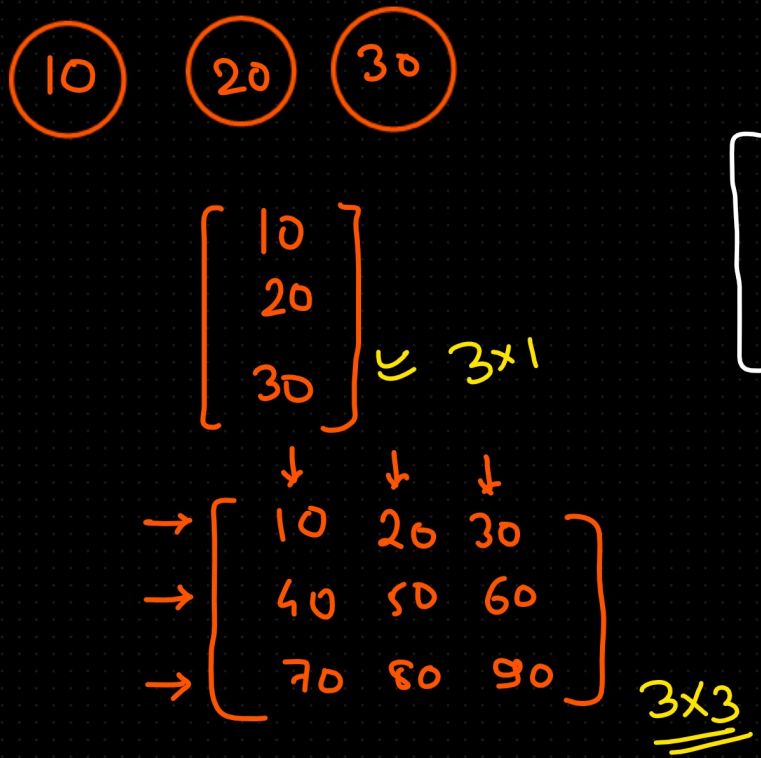


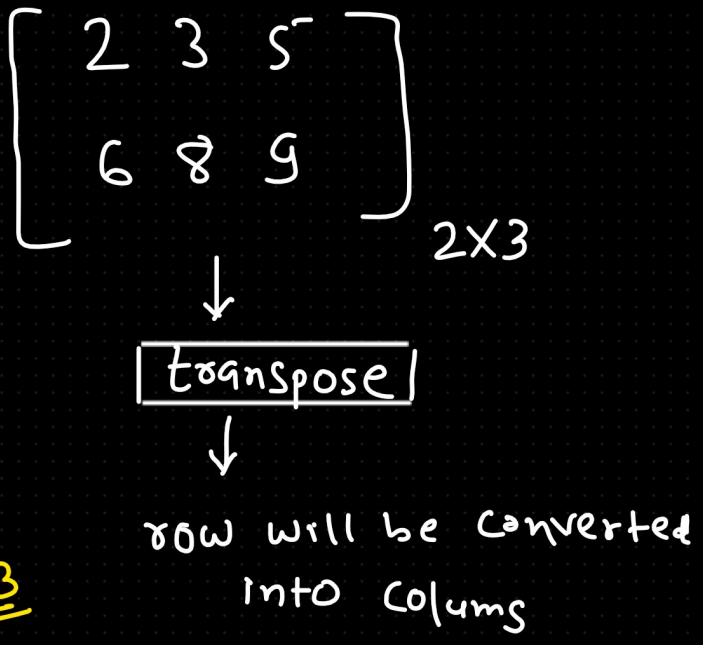
Transpose

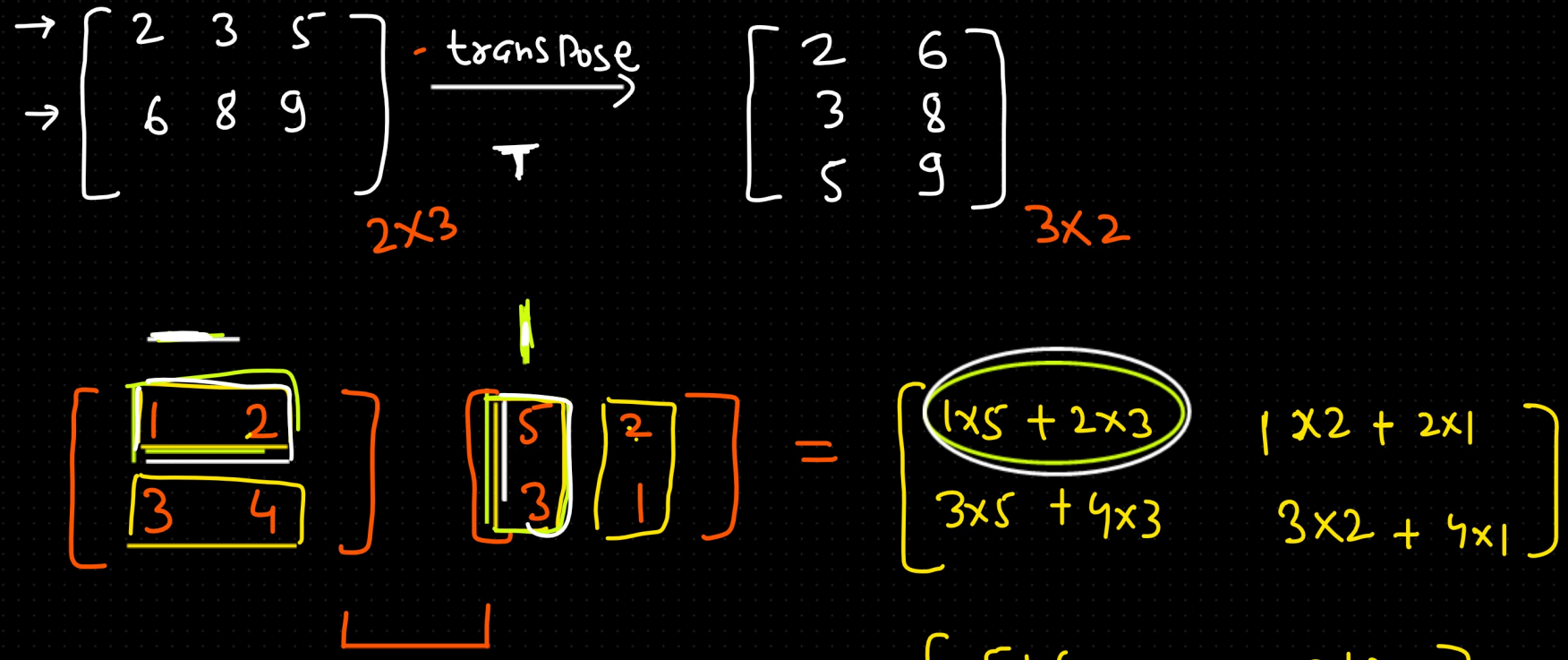
Vector

Matrix

Matrix multiplication operation







Ques: Why are we transposing i.e; $W^T \cdot X$?

Ans: Reason why we are transposing the matrix is because in matrix multiplication no. of column of first matrix needs to be equal to no. of row of second matrix.

Ques: How to know how many hidden layers to consider?

Ans: No. of hidden layers is the Hyperparameter which using hyperparameter tuning can provide optimal no. of hidden layers to be considered.

See the basic concept in the case of Neural Networks is that we use forward propagation to traverse forward in neural network to calculate the loss at the end(Output layer). This loss represents the difference of deviation of predicted value from actual value(loss=y-y^). We attempt to minimize this loss for which by using Back Propagation we traverse backward in the Neural network to optimize the Trainable parameters(Weights and Biases) using an optimization algorithm such as Gradient Descent, Sophistic Gradient Descent etc.

$$y = mx + c$$

$$\equiv \quad y = m_1 x_1 + m_2 x_2 + m_3 x_3 + m_4 x_4 + m_5 x_5 + c$$

$$\left\{ \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + [c] \right\}$$

$$\equiv$$

$$M^T X + C$$

$$\begin{bmatrix} m_1 & m_2 & m_3 & m_4 & m_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$$\equiv$$

Dot Product

Forward Propogation

$L_0 \Rightarrow$ I/P

$L_1 L_2 \Rightarrow$ HL (Hidden Layer)

$L_3 \Rightarrow$ O/P

CgPA	iq	10th	12th	Placed
—	—	—	—	—
—	—	—	—	—

Input layer = No. of Col

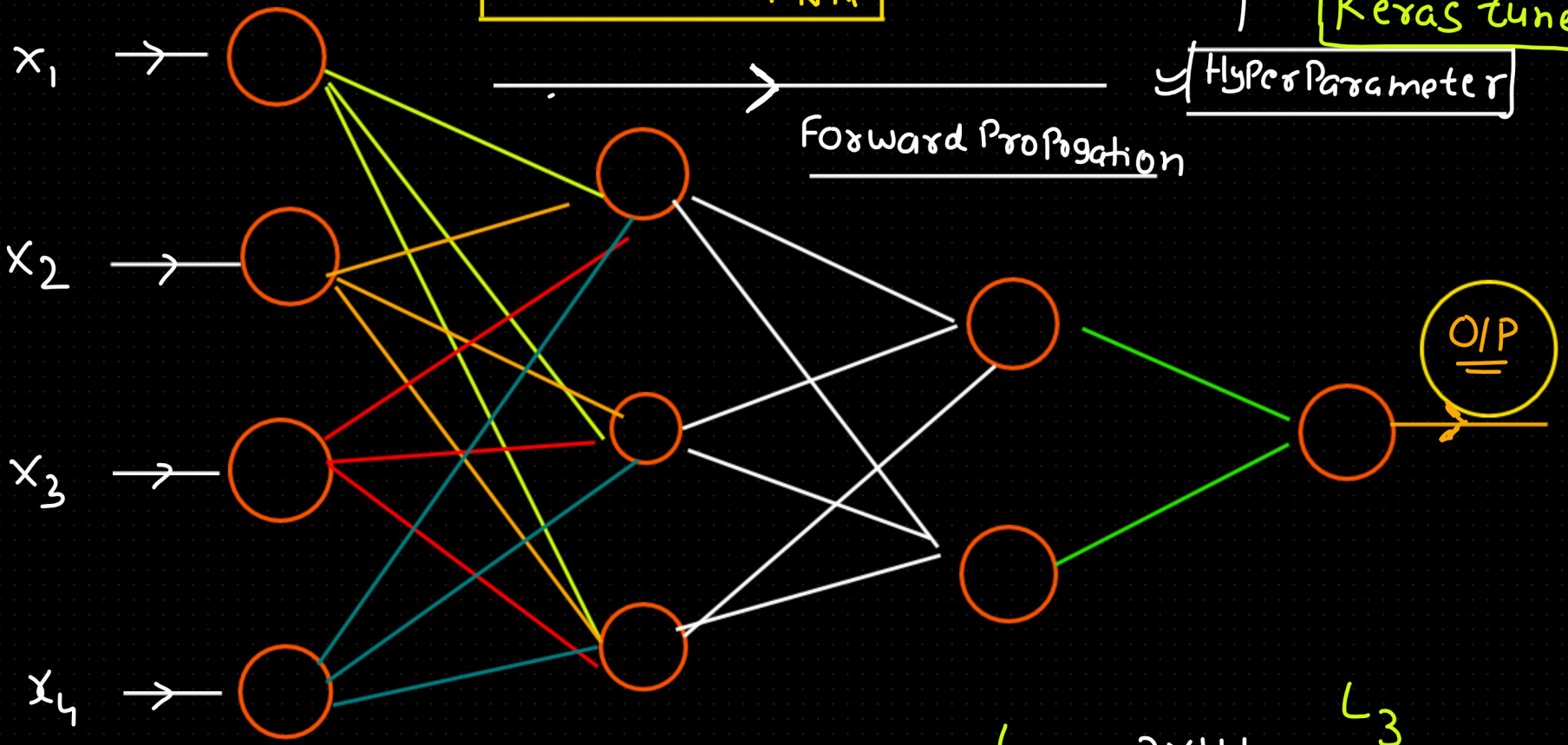
Neural Network where we each node is connected with each other is known as Fully Connected Neural Network

fully Connected NN

Hidden layer, No. of Neuron

↑ Keras tuner

↳ HyperParameter



Forward Propogation

L_0 $4 \times 3 + 3$ L_1 $3 \times 2 + 2$ L_2 $2 \times 1 + 1$ L_3

15 8 3

BackPropogation

Total number of trainable parameters

$\Rightarrow$ 26

$\Rightarrow$ FP $\Rightarrow$ Calculation

Loss

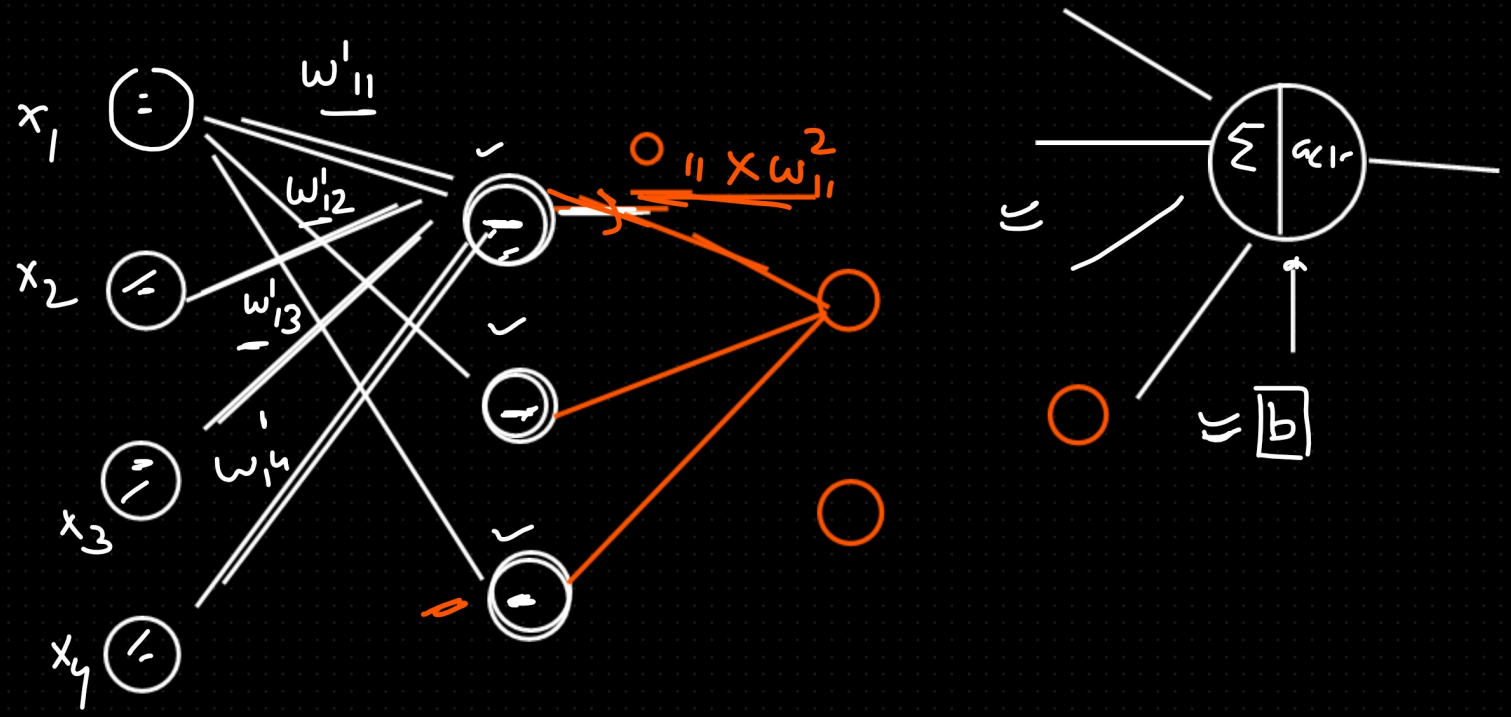
BP $\Rightarrow$ Optimizer $\Rightarrow$ Updating trainable Parameters

Layer1 Weight Matrix

Just observe here that how are the weights arranged in - W's Transpose and how basically a neuron's summation part is performing calculations internally

$$\rightarrow \begin{bmatrix} w_{11}' & w_{12}' & w_{13}' \\ w_{21}' & w_{22}' & w_{23}' \\ w_{31}' & w_{32}' & w_{33}' \\ w_{41}' & w_{42}' & w_{43}' \end{bmatrix}^T \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} w_{11}'x_1 + w_{21}'x_2 + w_{31}'x_3 + w_{41}'x_4 \\ w_{12}'x_1 + w_{22}'x_2 + w_{32}'x_3 + w_{42}'x_4 \\ w_{13}'x_1 + w_{23}'x_2 + w_{33}'x_3 + w_{43}'x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$



Resultant of the summation is passed to the activation function that yields to the Output O11, O12 and O13

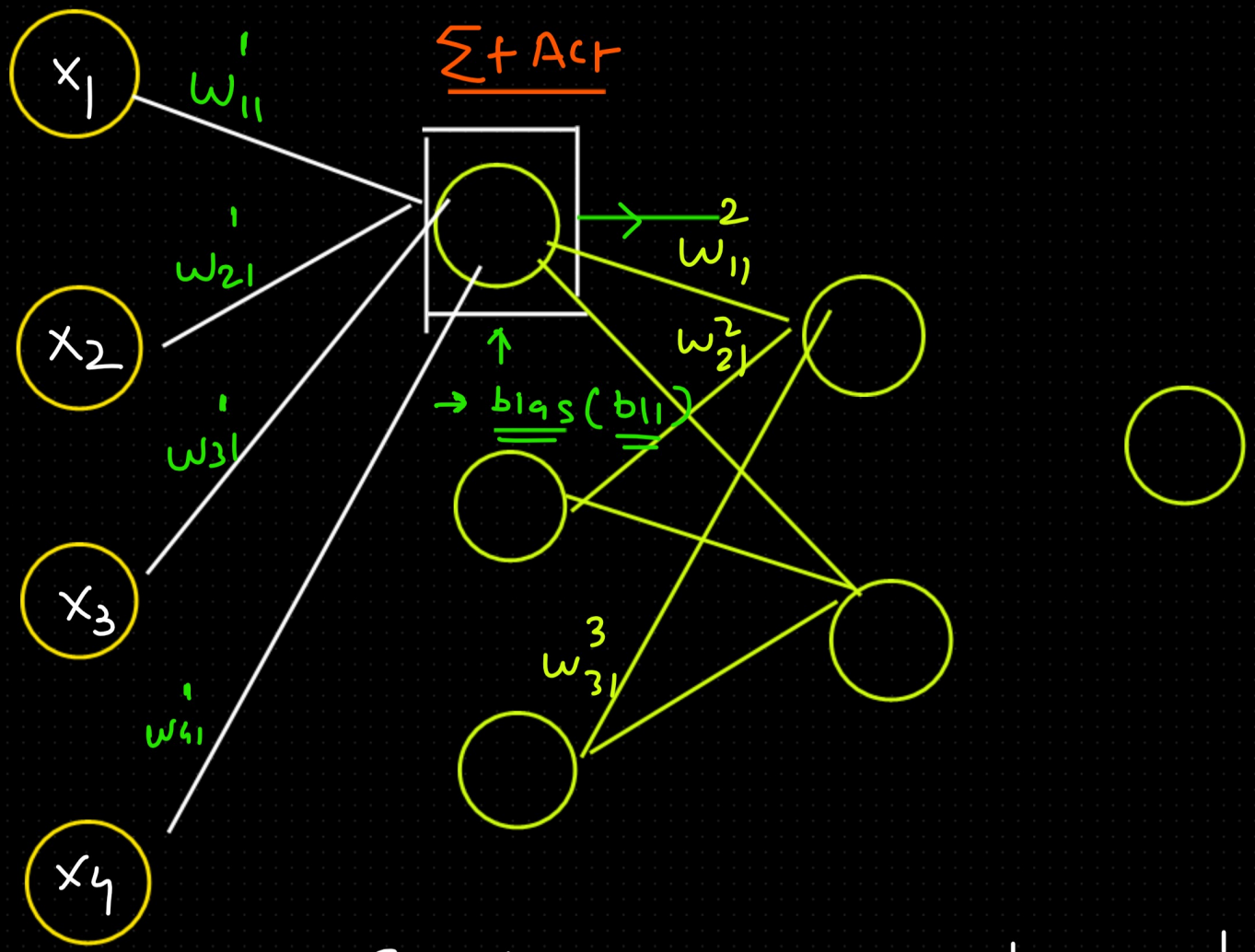
Activation fn

Act

$$\begin{pmatrix} \begin{bmatrix} w'_{11}x_1 + w'_{21}x_2 + w'_{31}x_3 + w'_{41}x_4 + b_{11} \\ w'_{12}x_1 + w'_{22}x_2 + w'_{32}x_3 + w'_{42}x_4 + b_{12} \\ w'_{13}x_1 + w'_{23}x_2 + w'_{33}x_3 + w'_{43}x_4 + b_{13} \end{bmatrix} \end{pmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix}$$



$$act\left(x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + x_4 w^1_{41} + b_{11}\right)$$

$$\begin{bmatrix} w_{11}^{2} & w_{12}^{2} \\ w_{21}^{2} & w_{22}^{2} \\ w_{31}^{2} & w_{32}^{2} \end{bmatrix}^{T} \times \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

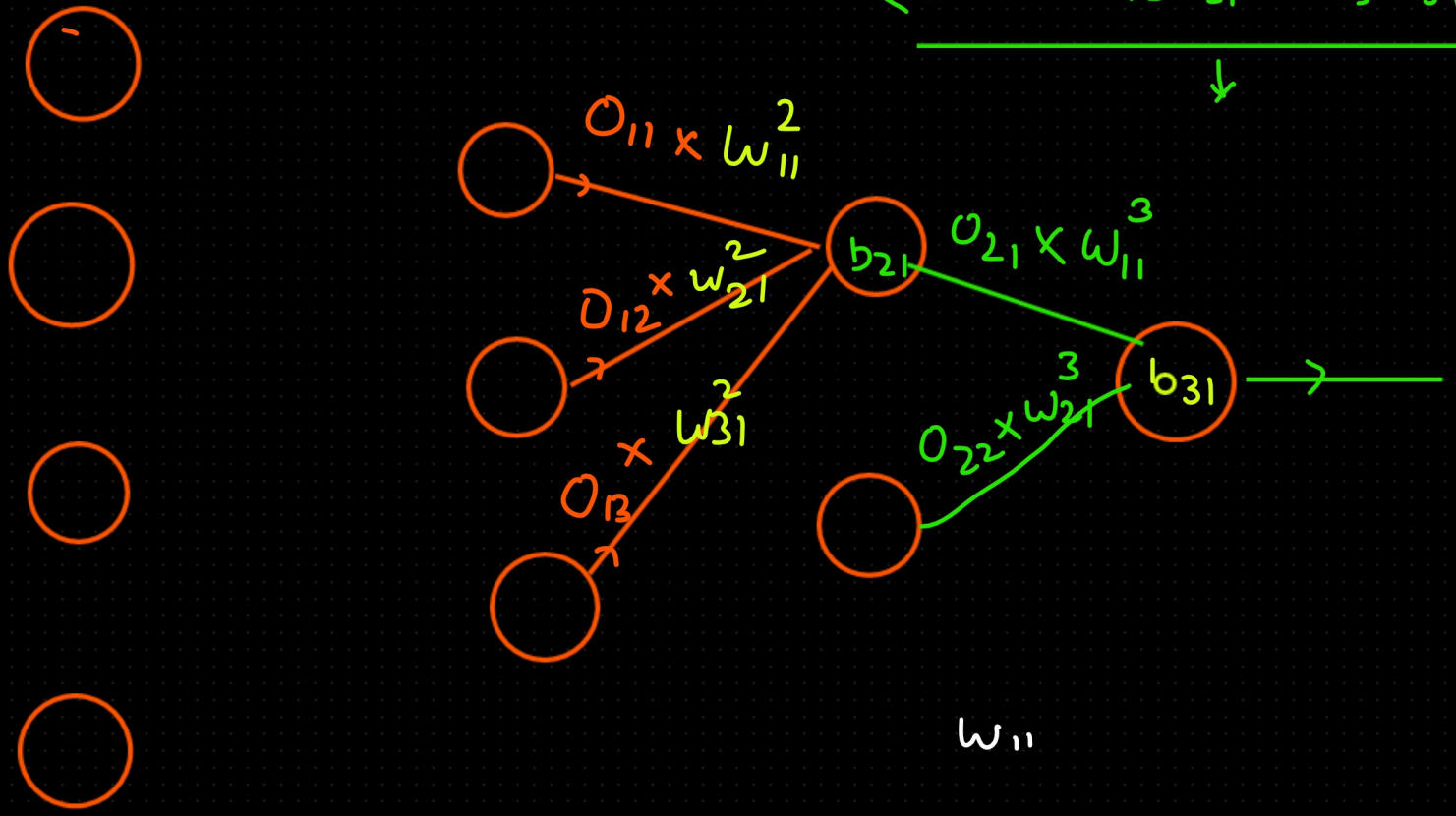
<u>alt</u>

$$\left(\begin{bmatrix} w_{11}^{2} o_{11} + w_{21}^{2} o_{12} + w_{31}^{2} o_{13} + b_{21} \\ w_{12}^{2} o_{11} + w_{22}^{2} o_{12} + w_{32}^{2} o_{13} + b_{2} \end{bmatrix} \right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

It can be clearly observed here that the Output from the previous layer is then multiplied with the respective next weights which then acts as the summation part for the neuron in the next layer. This is how in neural networks this chain keeps on going until output layer

act

$$z = \sigma\left(o_{11}w_{11}^{2} + o_{12}w_{21}^{2} + o_{13}w_{31}^{2} + b_{21}\right)$$



Layer 3

$$\begin{bmatrix} w_{11}^{3} \\ w_{21}^{3} \end{bmatrix}^{T} \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix}$$

2×1 $\quad$ 2×1 $\quad$ 1×1

$\hat{y}$

act $\left(\begin{bmatrix} w_{11}^{3} o_{21} + w_{21}^{3} o_{22} + b_{31} \end{bmatrix} \right) =$ Output

$y = mx + c$

26 ⇒ trainable param

= BP ⇒ GD ⇒ SGD