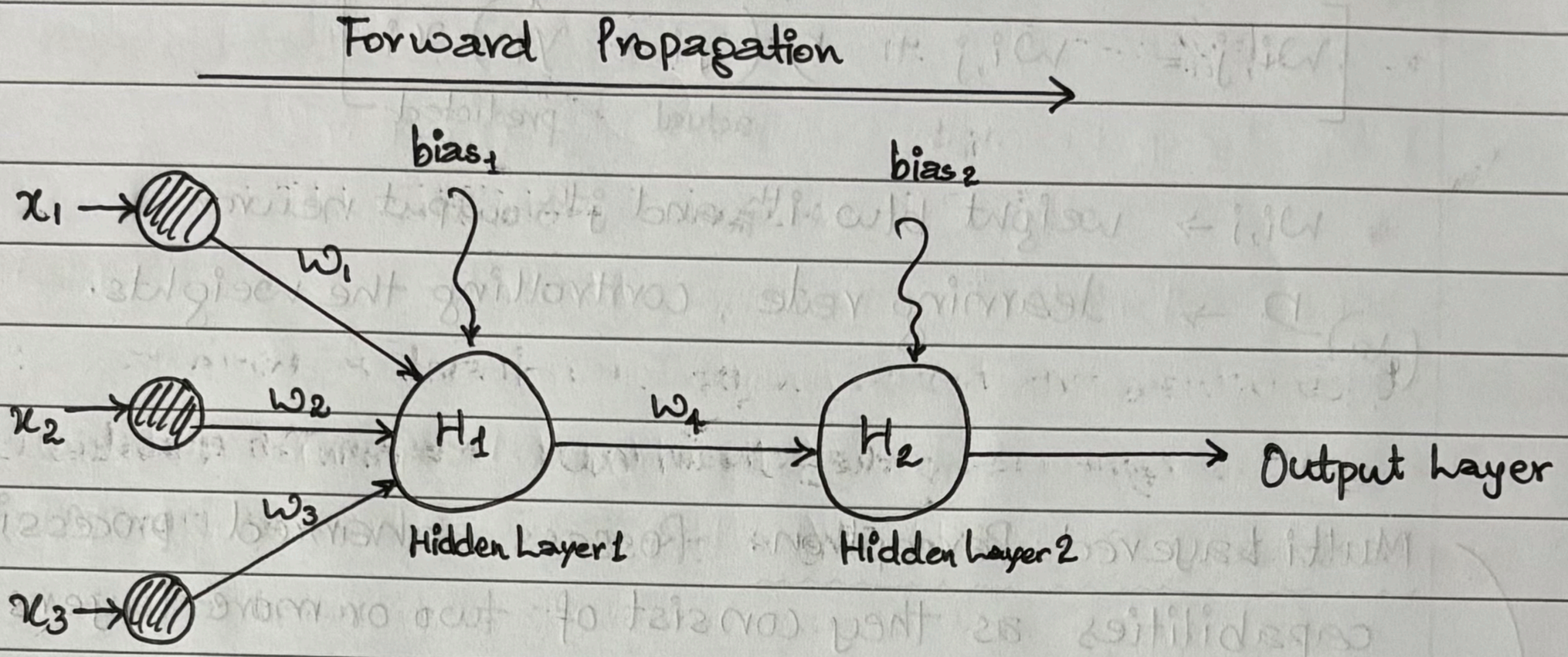
__/__/____

① **Forward Propagation**: The input data moves through each layer of neural network where each neuron applies weighted sum, add bias, passes the result through an activation function and making predictions.
This process is crucial before backpropagation updates weights



→ Hidden Layer 1: Step 1: $z = \sum_{i=1}^{n} x_i w_i + bias_1$

Forward pass }

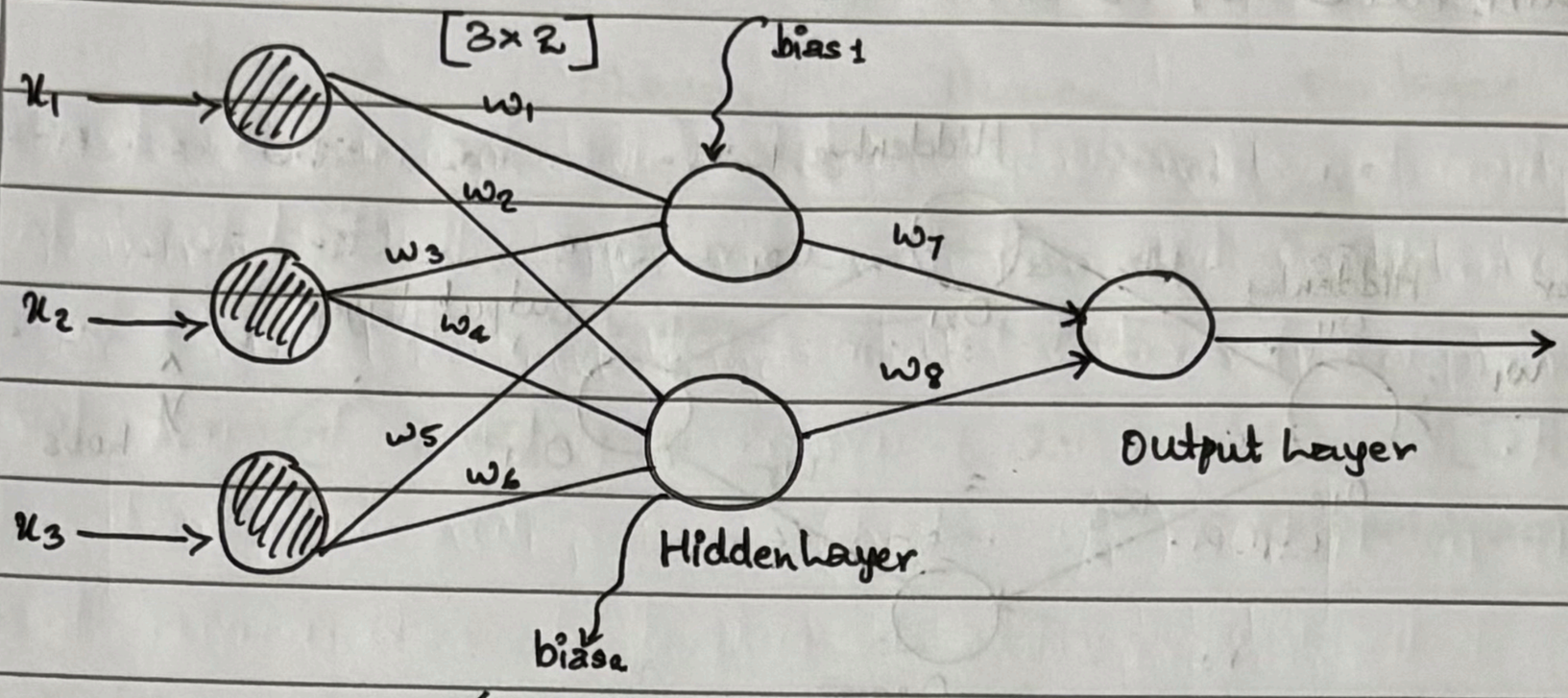
Step 2: Activation (z)

→ Hidden Layer 2: Step 1: z = Output H_1 * w + $bias_2$

Step 2: Activation (z) → $\hat{y}$ output predicted

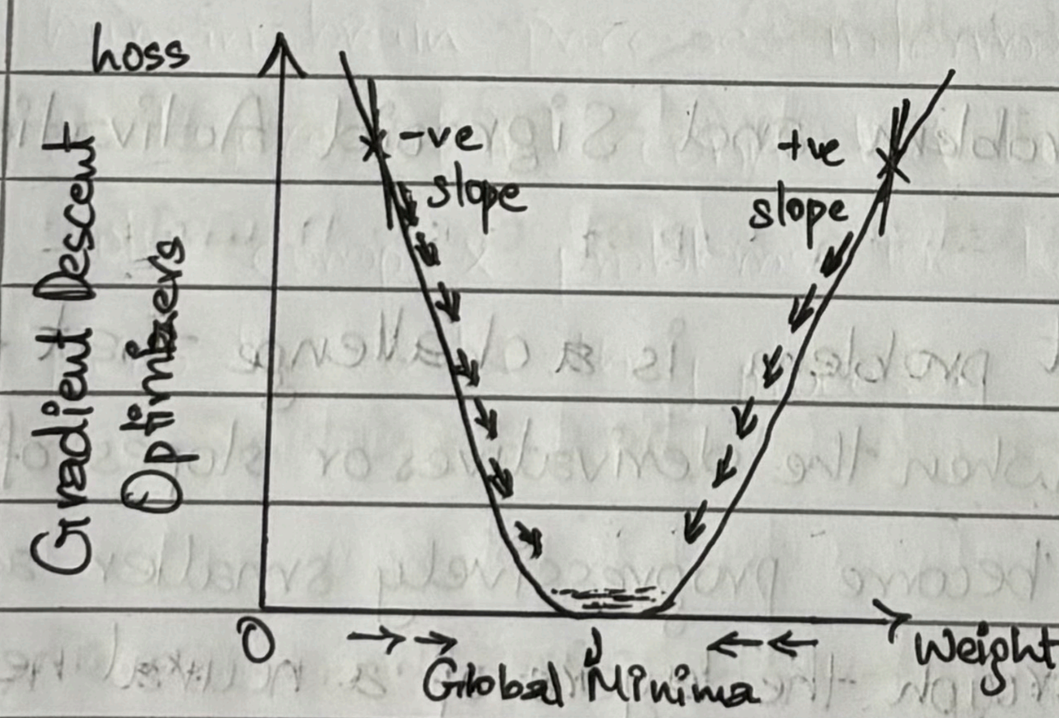
→ Calculate Loss function → (Actual output - predicted output)
If loss function is high we will update weights and continue to reduce loss function.

② **Backpropagation**: Also known as Backward Propagation of Errors is a method used to train neural network.
It's goal is to reduce the difference between the model's predicted output and actual output by adjusting weights and biases.



It computes the gradient of loss function with respect to each weight using the chain rule making it possible to update weights efficiently. These gradients indicate how much each weight and bias should be adjusted to minimize the error in the next iteration. The activation function through its derivative plays a crucial role in computing these gradients during Back Propagation.

→ Weight Updation formula: $W_{new} = W_{old} - \eta \frac{\partial Loss}{\partial W_{old}}$ (η → learning Rate)



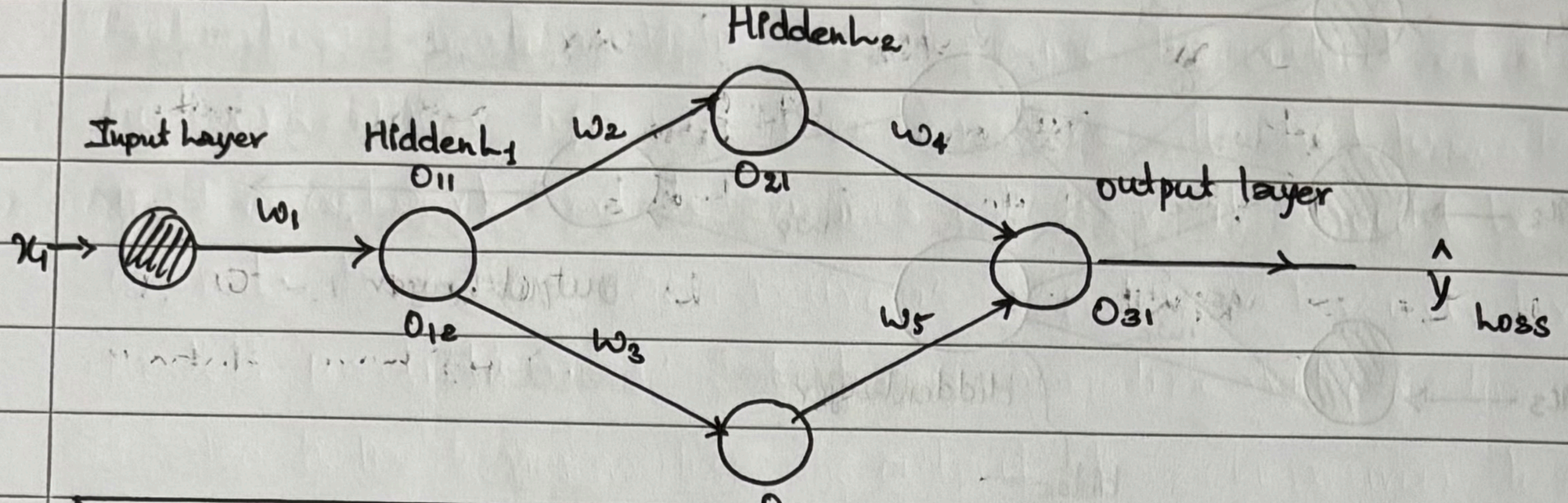
* We use optimizers to reduce the loss va

* η → should be always a small value (0.00

* When $W_{new} = W_{old}$, we reach global minim

for -ve slope, $W_{new} = W_{old} - \eta(-ve)$	for +ve slope, $W_{new} = W_{old} - \eta(+ve)$
$= W_{old} + \eta(+ve)$	$= W_{old} - \eta(+ve)$
$W_{new} >> W_{old}$	$W_{new} << W_{old}$

③ *Chain Rule of Derivatives*



$$w_{new} = w_{old} - \eta \frac{\partial Loss}{\partial w_{old}}$$

Calculating $w_{1new} = w_{1old} - \eta \frac{\partial Loss}{\partial w_{1old}}$

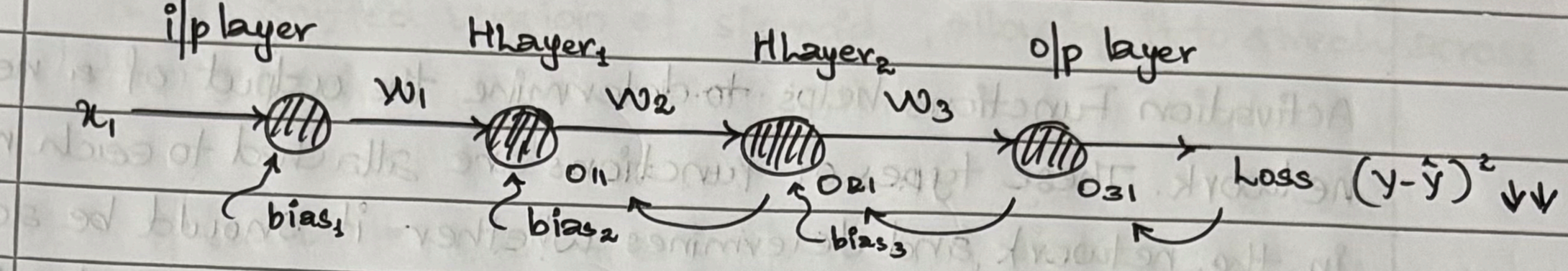
$$\Rightarrow \frac{\partial loss}{\partial w_{1old}} = \left[\frac{\partial Loss}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1old}}\right]$$

$$+$$

$$\left[\frac{\partial loss}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1old}}\right]$$

④ *Vanishing Gradient Problem and Sigmoid Activation*

The vanishing gradient problem is a challenge that emerges during backpropagation when the derivatives or slopes of the activation functions become progressively smaller as we move backward through the layers of a neural network.

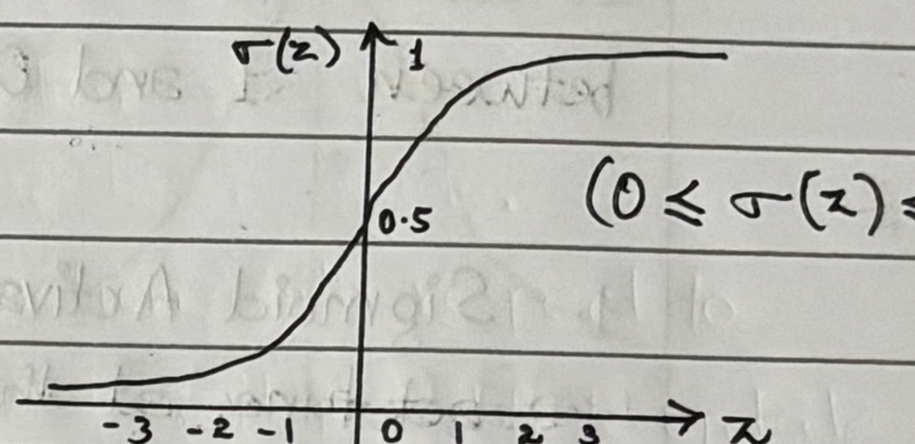


1. $z = \sum_{i=1}^{n} x_i w_i + b$

2. $\sigma(z) \rightarrow [0 \text{ to } 1]$

↳ sigmoid activation function

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



$(0 \leq \sigma(z) \leq 1)$

w.r.t $W_{1new} = W_{1old} - \eta \frac{\partial loss}{\partial W_{1old}}$ → smaller value

η → 0.01

Derivative of $\sigma(z)$

$(0 \leq \partial(\sigma(z)) \leq 0.$

expanding

$$\frac{\partial loss}{\partial W_{1old}} = \frac{\partial loss}{\partial O_{31}} * \frac{\partial O_{31}}{\partial O_{21}} * \frac{\partial O_{21}}{\partial O_{11}} * \frac{\partial O_{11}}{\partial W_{11}}$$

0-0.25 0-0.25 0-0.25

$$O_{31} = \sigma(W_3 * O_{21} + b_3)$$

$$O_{31} = \sigma(z)$$

→ z (input going before the activation)

$$\frac{\partial O_{31}}{\partial O_{21}} = \frac{\partial(\sigma(z))}{\partial(z)} * \frac{\partial z}{\partial O_{21}}$$

→ (Derivative of sigmoid) ⇒ $0 \leq \sigma(z) \leq 0.25$

$$\Rightarrow 0 \leq \sigma(z) \leq 0.25 * \partial((W_3 * O_{21}) + b_3) / \partial(O_{21})$$

So, as η is around 0.01 and $\partial Loss / \partial W_{1old} \approx 0.25$,

$W_{1new} \approx W_{1old}$, the weights are never getting updated and we are not moving to the convergance zone. It is because of sigmoid.