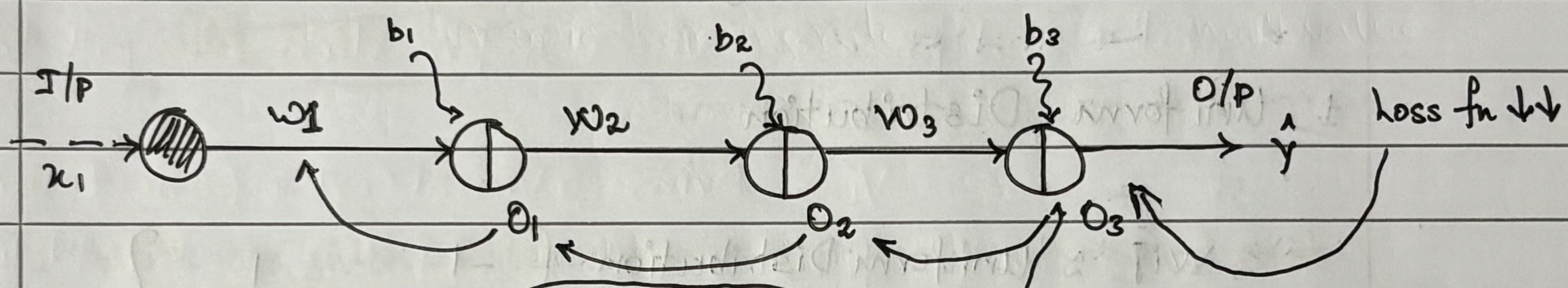
__/__/____

⑧ *Exploding Gradient Problem*

It is a challenge encountered during training deep neural networks. It occurs when the gradients of the networks loss function with respect to the weights become excessively large. The issues arises when, during backpropagation the derivatives or slopes of the neural network is layers grow progressively larger as we move backward.

→ Exact opposite of Vanishing Gradient Problem.



$W_{1new} = W_{1old} - \eta \left(\frac{\partial loss}{\partial W_{1old}} \right)$

$z = \sigma (O_2 * W_3 + b_3)$

$$\frac{\partial loss}{\partial W_{1old}} = \frac{\partial loss}{\partial O_3} * \frac{\partial O_3}{\partial O_2} * \frac{\partial O_2}{\partial O_1} * \frac{\partial O_1}{\partial W_{1old}}$$

$$\frac{\partial O_3}{\partial O_2} = \frac{\partial \sigma(z)}{\partial (z)} * \frac{\partial (z)}{\partial O_2}$$

sigmoid ↖

$$= [0 - 0.25] * W_3$$

If we intialized the weights with a big value,

$\{ W_{inew} >>> W_{iold}$ or $W_{inew} <<< W_{iold} \}$,

there will be scenario that the gradients may go out of bounds.

__/__/__

⑨ *Weight Initialization Techniques*

If the weights are not initialized correctly, it may give rise to the Vanishing Gradient Problem or Exploding Gradient Problem.

1. Weights should be small.
2. Weights should not be same (every neuron should have the capacity to process in a different way)
3. Weights should have good variance.

1. Uniform Distribution

$$W_{ij} \approx \text{Uniform Distribution} \left\{ \frac{-1}{\sqrt{\text{input}}} , \frac{1}{\sqrt{\text{input}}} \right\}$$

2. Xavier / Glorot Initialization

→ Normal Initialization, $W_{ij} \approx N(M=0, \sigma)$

where $\sigma = \sqrt{\frac{2}{(in + out)}}$

→ Uniform Initialization, $W_{ij} \approx \text{Uniform} \left\{ \frac{-\sqrt{6}}{\sqrt{in+out}} , \frac{\sqrt{6}}{\sqrt{in+out}} \right\}$

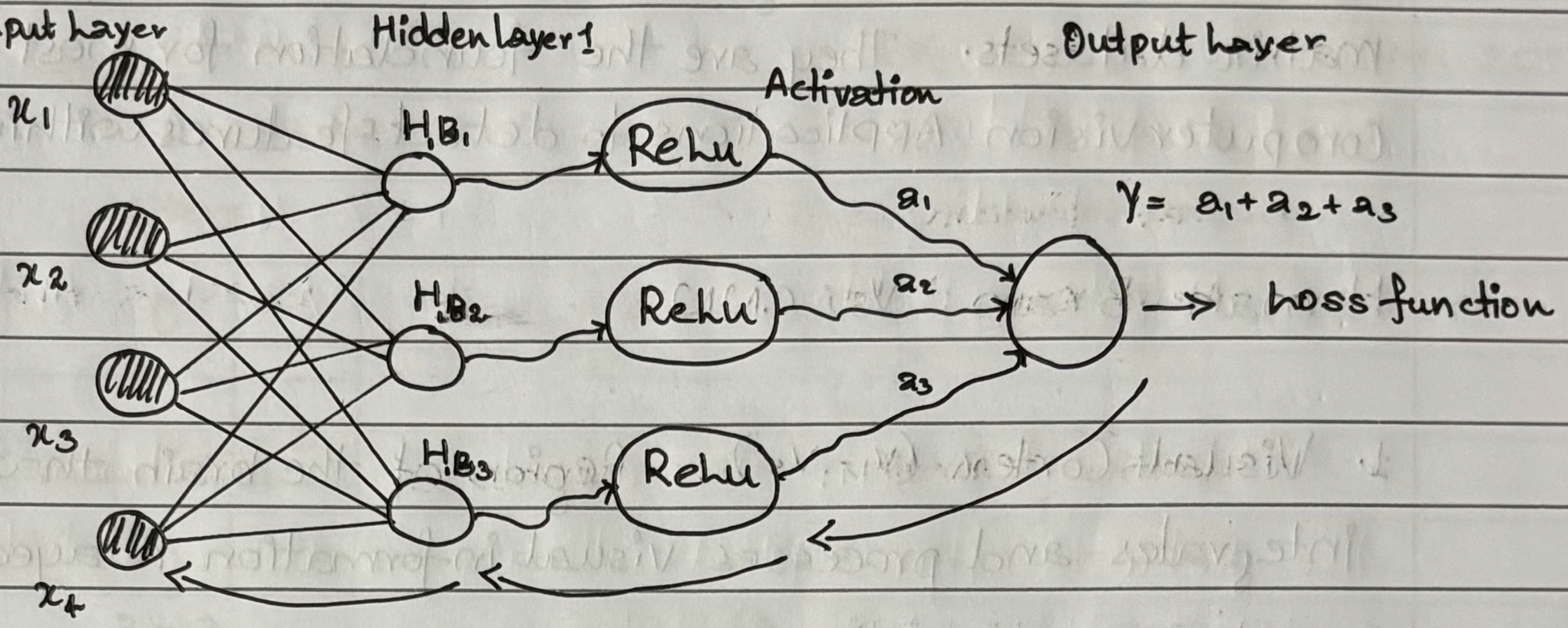
3. Kaiming He Initialization

→ He Normal, $W_{ij} \approx N(\mu=0, \sigma)$, $\sigma = \sqrt{2/\text{input}}$

→ He Uniform, $W_{ij} \approx \text{Uniform} \left\{ -\sqrt{\frac{6}{\text{input}}} , \sqrt{\frac{6}{\text{input}}} \right\}$

__/__/____

→ Complete Working of a Artificial Neural Network.



1. With the given initial parameters, perform forward pass.

layer → Activation Function → Output → Calculate loss function

2. Perform Back propagation → find how much each weight contributed to error, so we can adjust it.

3. Compute Gradients with Chain Rule

4. Update the weights and biases using optimizers

5. Repeat for next epochs until convergance (loss is reduced to the best)