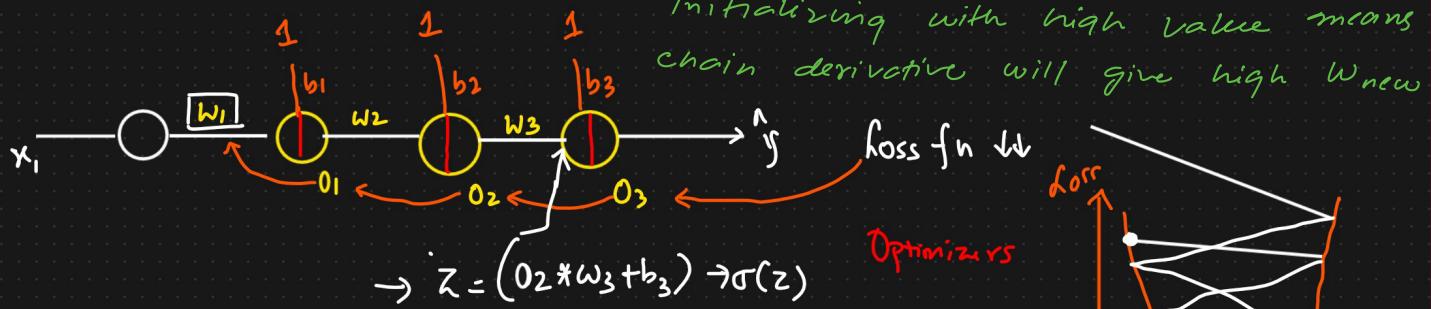


# Exploding Gradient Problem $\Rightarrow$ Weight Initialization Technique



$$w_{1,new} = w_{1,old} - \eta \boxed{\frac{\partial h}{\partial w_{1,old}}}$$

$\left\{ \begin{array}{l} w_{new} \ggg w_{1,old} \\ w_{new} \lll w_{1,old} \end{array} \right.$   
 Chain Rule of Derivatives

$$\frac{\partial h}{\partial w_{1,old}} = \frac{\partial h}{\partial o_3} * \boxed{\frac{\partial o_3}{\partial o_2}} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{1,old}}$$

big \* big \* big \* big  $\rightarrow$  big value

$$\frac{\partial o_3}{\partial o_2} = \boxed{\frac{\partial \sigma(\omega)}{\partial \omega}} * \frac{\partial \omega}{\partial o_2}$$

$$= [0 - 0.25] * \frac{\partial (o_2 * w_3 + b_3)}{\partial o_2} = [0 - 0.25] * w_3 \Rightarrow \underline{\underline{500 - 1000}}$$

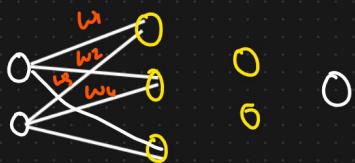
Exploding gradients happen when gradients grow too large during backpropagation, often due to deep networks or bad initialization. They can break the training process. Fix it with good initialization, gradient clipping, and careful architecture design.

Gradient explode = 

- Bad initialization
- Deep networks

## Weight Initialising Techniques

- ① Uniform Distribution ✓
- ② Xavier/Glorot Initialization ✓
- ③ Kaiming/He Initialization ✓

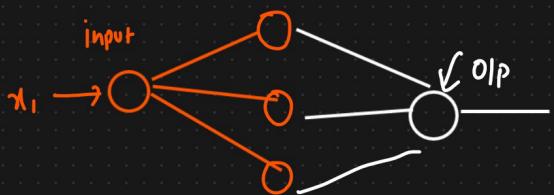


### Solutions:

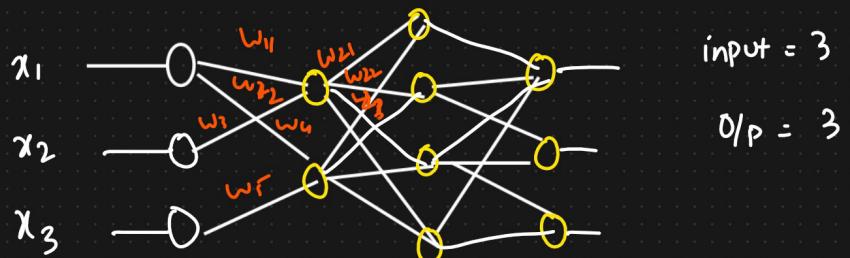
Problem	Solution
Gradients getting too large	Gradient Clipping
Poor initialization	Use He/Xavier Initialization
Very deep network	Use architectures like ResNet (skip connections)
High learning rate	Lower learning rate

## Key Points

- ① Weights should be small ✓
- ② Weights should not be same ✓
- ③ Weights should have good variance



input = 1  
Output = 1



input = 3

O/P = 3

## ① Uniform Distribution

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

$$\left[ \frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}} \right]$$

## ② Xavier/Glorot Initialization

Researcher → Xavier Glorot

### ① Xavier Normal Init

$$W_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{(\text{input} + \text{output})}}$$

### ② Xavier Uniform

$$W_{ij} \sim \text{Uniform Distribution}$$

$$\left[ \frac{-\sqrt{6}}{\sqrt{\text{input} + \text{output}}}, \frac{\sqrt{6}}{\sqrt{\text{input} + \text{output}}} \right]$$

### ③ Kaiming He Initialization

#### ① He Normal

$$W_{ij} \sim N(0, \sqrt{\sigma^2})$$

$$\sigma = \sqrt{\frac{2}{\text{input}}}$$

#### ② He uniform

$$W_{ij} \sim \text{Uniform Distribution} \left[ -\sqrt{\frac{6}{\text{input}}}, \sqrt{\frac{6}{\text{input}}} \right]$$

#### Conclusion:

Q: why do we need weight initialization Techniques?

when training a neural network, how we initialize the weights matters a lot - it can make or break our model's ability to learn.

#### ✓ Key Reasons:

Reason	Explanation
Breaks Symmetry	If all weights are the same (e.g., zeros), neurons learn the same features — no diversity.
Avoids Vanishing/Exploding Gradients	Poor initialization can cause gradients to become too small or too large during backprop.
Speeds Up Convergence	Good initialization helps the model learn faster and more reliably.

Imp

#### ✓ Essential Weight Initialization Techniques

Technique	Use When	Activation Function
Xavier (Glorot)	For balanced input/output layers	tanh, sigmoid
He Initialization	For deep networks with ReLU-based activations	ReLU, Leaky ReLU
Random Uniform	Simple or shallow networks	Any (but not recommended for deep networks)



#### Quick Recap:

- 🎯 Xavier → good for smooth activations
- ⚡ He → best for ReLU and its variants
- ⚠ Avoid zeros or constant initializations