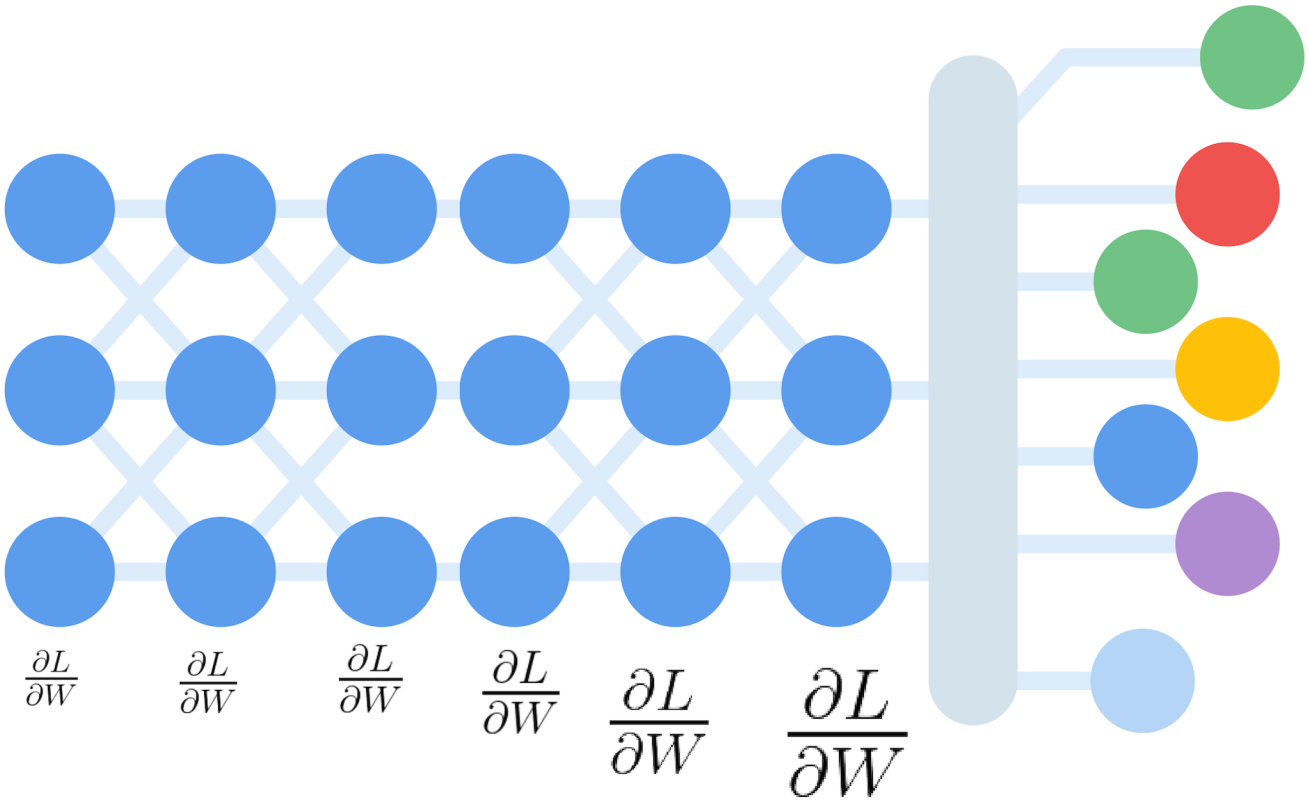
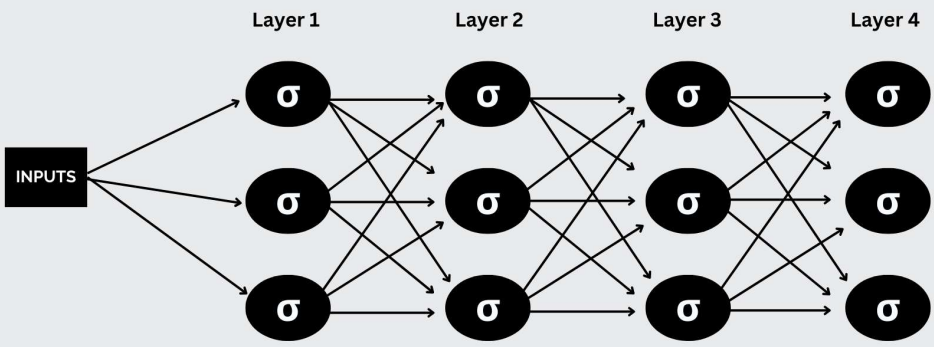
Fundamentals of Deep Learning

The Vanishing Gradient Problem



Shivang Kainthola

Consider this neural network with four layers, with the sigmoid function as its activation function.



Weights	W_1	W_2	W_3	W_4
Linear Transformation	$z(1) = W_1.(a_0) + b_1$	$z(2) = W_2.(a_1) + b_2$	$z(3) = W_3.(a_2) + b_3$	$z(4) = W_4.(a_3) + b_4$
Output	$a(1) = \sigma(z(1))$	$a(2) = \sigma(z(2))$	$a(3) = \sigma(z(3))$	$a(4) = \sigma(z(4))$

Forward Pass

During forward pass, every **layer N** of the neural network calculates its outputs by :

1) Obtaining a **linear transformation Z** of its inputs

(which are obtained from previous layers) which includes the **Weights (W)** , the **Bias (b)** term of that layer, and the output (a) of previous layer :

$$Z_n = W_n.(a_{n-1}) + b_n$$

2) Passing that linear transformation through its activation function to get the output

The activation function for this example is the sigmoid function.

$$a_n = \sigma\left(Z_n\right)$$

And this continues for the next layer :

$$Z_{n+1} = W_{n+1}.(a_n) + b_{n+1}$$
$$a_{n+1} = \sigma\left(Z_{n+1}\right)$$

After calculating the output, the loss function is used to calculate the error.

Backpropagation

The process of training neural networks involves backpropagation and gradient descent.

During backpropagation, starting from the last hidden layer, we take derivatives of the loss function with respect to the weights of the layer.

For a four layer neural network, the training (backpropagation) using the loss function will begin with the final layer. [SEP]

1) Calculate gradient of the Loss function (L) with respect to the Activation function's output for the final output layer.

2) Decompose it using chain rule to derivative of the linear transformation (Z), and then further, to update each Weight (W) of the final layer.

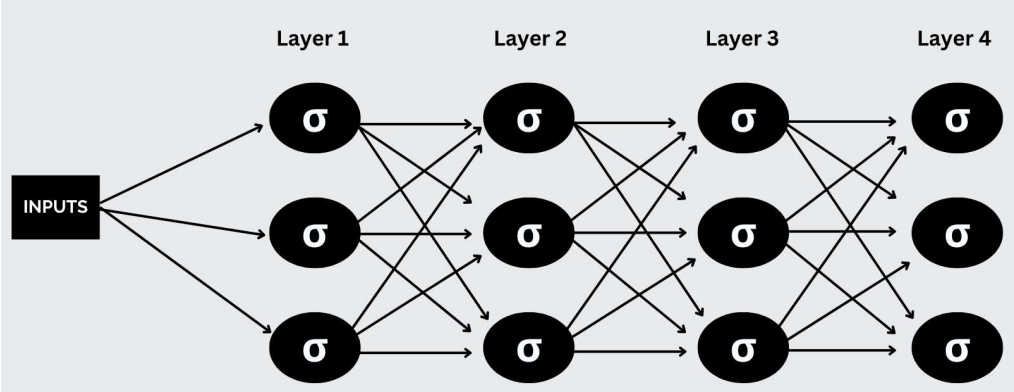
3) Calculate the Error Term.

4) Propagate the error term back to the previous (N-1) layer.

5) Calculate the error term for the (N-1)th layer, use it to update the weights and biases.

6) Propagate the error to (N-2) layer, and repeat.

Now, considering the backpropagation for 4 layer neural network :



We will calculate the Error Term for the Final Layer (Layer 4) :

δ_4

We refer to the gradient of the loss function with respect to the pre-activation values (linear combination) as the Error Term.

This error term is propagated back to other layers.

$$\delta_4 = \frac{\partial L}{\partial z_4}$$

$$\frac{\partial L}{\partial z_4} = \frac{\partial L}{\partial a_4} . \frac{\partial a_4}{\partial z_4}$$

$$= \frac{\partial L}{\partial a_4} . \frac{\partial a_4}{\partial z_4} . \frac{\partial z_4}{\partial W_4}$$

$$= \frac{\partial L}{\partial a_4} . \frac{\partial a_4}{\partial z_4} . \frac{\partial}{\partial W_4} (W_4 . (a_3) + b_4)$$

$$= \frac{\partial L}{\partial a_4} . \frac{\partial}{\partial z_4} (\sigma(z_4)) . (a_3)$$

$$= \frac{\partial L}{\partial a_4} . (\sigma'(z_4)) . (a_3)$$

Given that :

$$a(4) = \sigma(z(4))$$

$$z_4 = W_4 . (a_3) + b_4$$

It is propagated it back to previous layers which update their weights and propagate it back:

For layer 3 :

$$\delta_3 = (W_4)^T.(\delta_4).(\sigma'(z_3))$$

For layer 2 :

$$\delta_2 = (W_3)^T.(\delta_3).(\sigma'(z_2))$$

For layer 1 :

$$\delta_1 = (W_2)^T.(\delta_2).(\sigma'(z_1))$$

(W(l+1)T is the transpose of the weight matrix connecting layer previous layer to next layer.

We propagate the error term δ backward through each layer,
the values get multiplied by
small derivatives σ′(z(l)) and
potentially small weights W(l+1)

As the number of layers increases, the error term becomes smaller and smaller.

By the time it reaches the earlier layers of the network, it diminishes to almost negligible values i.e vanishes
and those layers are barely updated.

Vanishing Gradient

For *deeper* neural networks, the diminishing value of the error term i.e. the vanishing gradient, becomes a major problem.

The derivatives of the activation functions decrease to extremely small values, and decrease the effect of the updates when multiplied to the error term.

As a result, the early layers of the network are hardly trained.

How is it fixed ?

1) Changing the activation functions :

Using activation function like ReLU, whose derivatives do not decrease drastically, can help maintain the magnitude.

2) Initializing weights with appropriate magnitudes, considering the scale of the neural network.

3) Normalizing the inputs for each layers to preserve gradients.

4) Using optimizer algorithms, like Adam, RMSProp etc.