UNIT - 4

NEURAL NETWORKS

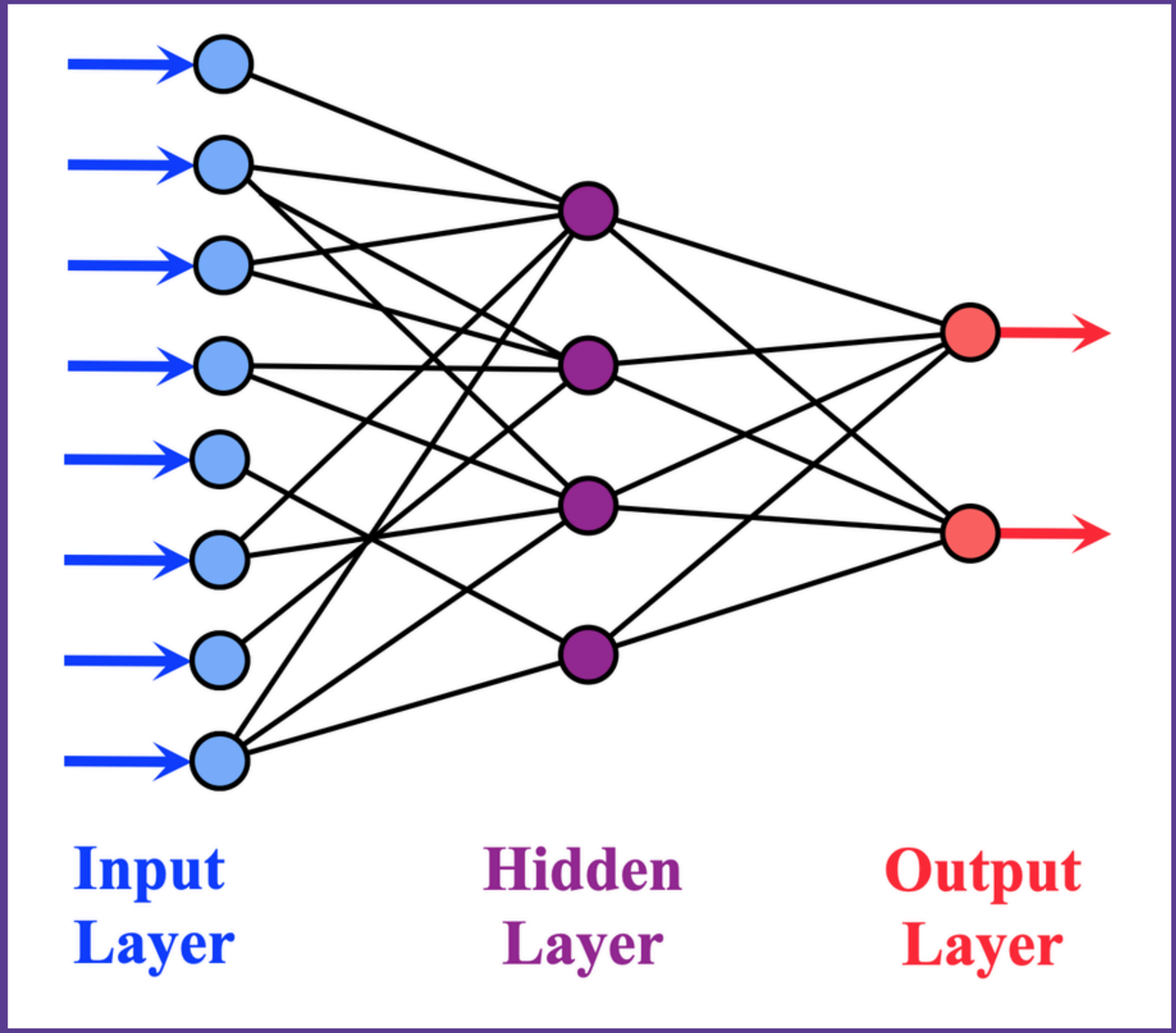
UNIT IV NEURAL NETWORKS 9

Multilayer perceptron, activation functions, network training – gradient descent optimization – Stochastic gradient descent, error backpropagation, from shallow networks to deep networks – Unit saturation (aka the vanishing gradient problem) – ReLU, hyperparameter tuning, batch normalization, regularization, dropout

From Shallow networks to deep networks

- Shallow Neural Network has only one (or just a few) hidden layers between the input and output layers.
- The input layer receives the data, the hidden layer(s) process it, and the final layer produces the output.
- Shallow neural networks are simpler, more easily trained, and have greater computational efficiency than deep neural networks.
- Shallow networks are typically used for simpler tasks such as linear regression, binary classification, or low-dimensional feature extraction.

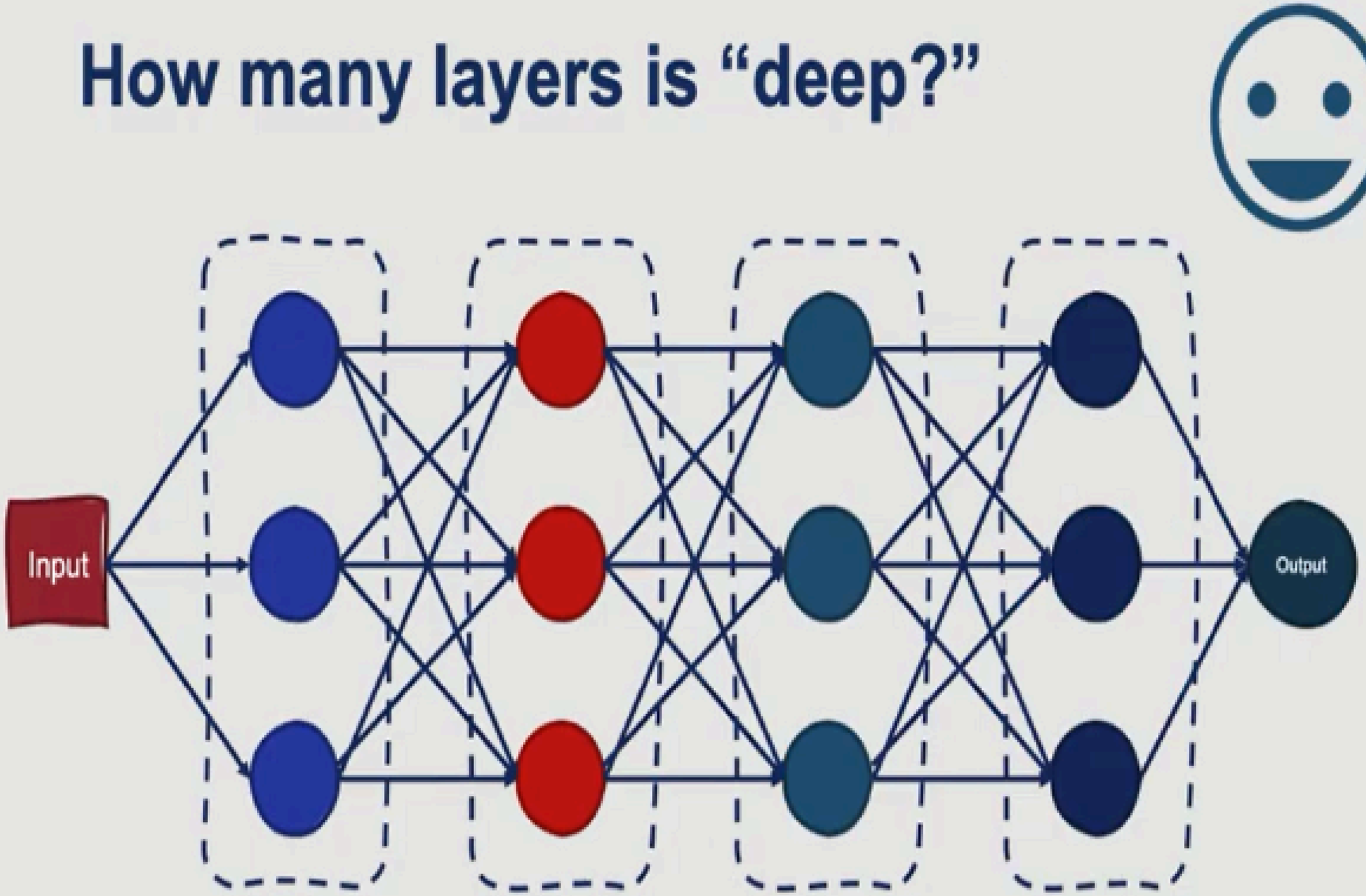
Shallow Neural Network

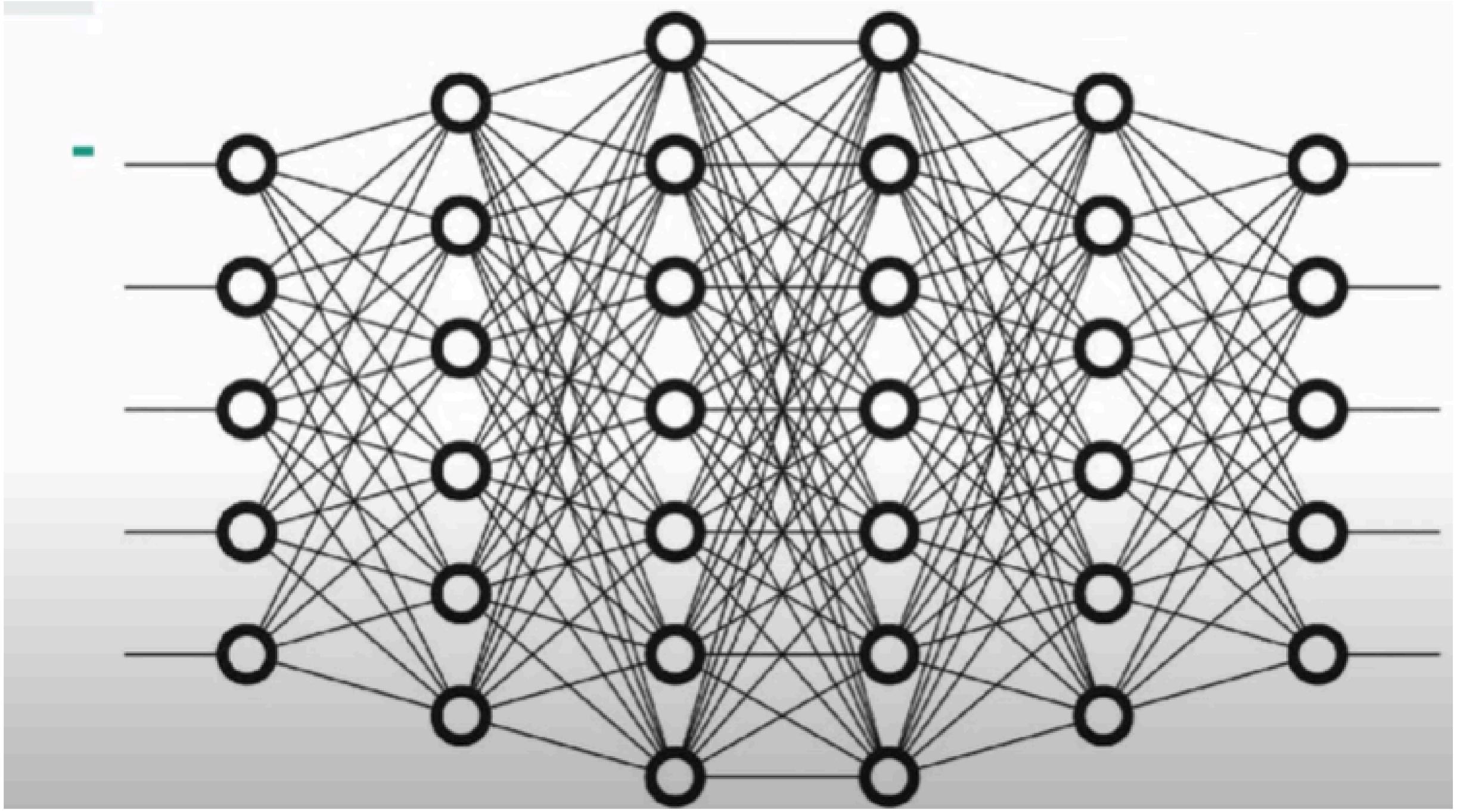


Deep Neural Networks

- Deep Neural Networks have multiple hidden layers stacked between the input and output layers.
- These networks can be quite deep, with tens or even hundreds of layers.

How many layers is “deep?”





Why it's significant:

- For shallow network with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively.
- Gradients of neural networks are found using backpropagation. Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one. By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.
- However, when *n* hidden layers use an activation like the sigmoid function, *n* small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.
- A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

Unit saturation (aka the vanishing gradient problem)

- The vanishing gradient problem is a challenge faced in training artificial neural networks, particularly deep feedforward and recurrent neural networks.
- This issue arises during the backpropagation process, which is used to update the weights of the neural network through gradient descent.

Unit saturation (aka the vanishing gradient problem)

- The gradients are calculated using the chain rule and propagated back through the network, starting from the output layer and moving towards the input layer.
- However, when the gradients are very small, they can diminish as they are propagated back through the network, leading to minimal or no updates to the weights in the initial layers. This phenomenon is known as the vanishing gradient problem.

Causes of Vanishing Gradient Problem

- The vanishing gradient problem is often attributed to the choice of activation functions and the architecture of the neural network.
- Activation functions like the sigmoid or hyperbolic tangent (tanh) have gradients that are in the range of 0 to 0.25 for sigmoid and -1 to 1 for tanh.

Causes of Vanishing Gradient Problem

- When these activation functions are used in deep networks, the gradients of the loss function with respect to the parameters can become very small, effectively preventing the weights from changing their values during training.
- Another cause of the vanishing gradient problem is the initialization of weights.
- If the weights are initialized too small, the gradients can shrink exponentially as they are propagated back through the network, leading to vanishing gradients.

Consequences of Vanishing Gradient Problem

- The vanishing gradient problem can severely impact the training process of a neural network.
- Since the weights in the earlier layers receive minimal updates, these layers learn very slowly, if at all.
- This can result in a network that does not perform well on the training data, leading to poor generalization to new, unseen data.
- In the worst case, the training process can completely stall, with the network being unable to learn the complex patterns in the data that are necessary for making accurate predictions.

Solutions to this problem

Long Short-Term Memory(LSTM)

- So as of now, we have seen there are two major factors that affect the gradient size – weights and their derivatives of the activation function. A simple LSTM helps the gradient size to remain constant. The activation function we use in the LSTM often works as an identity function which is a derivative of 1. So in gradient backpropagation, the size of the gradient does not vanish.

Residual Neural Network

- The skip or bypass connection inr residual network is useful in any network to bypass the data from a few layers. Basically, it allows information to skip the layers. Using these connections, information can be transferred from layer n to layer n+t. Here to perform this thing we need to connect the activation function of layer n to the activation function of n+t. This causes the gradient to pass between the layers without any modification in size.

ReLu Activation Function

- If the ReLU function is used for activation in a neural network in place of a sigmoid function, the value of the partial derivative of the loss function will be having values of 0 or 1 which prevents the gradient from vanishing. The use of ReLU function thus prevents the gradient from vanishing