1) **Derive the back propagation rule considering the training rule for output unit weights and training rule for hidden unit weights.**
https://www.i2tutorials.com/machine-learning-tutorial/machine-learning-derivativation-of-back-propagation-rule/

2) **Explain activation functions with neat diagrams.**
https://www.geeksforgeeks.org/activation-functions-neural-networks/

3) **Discuss linear neurons and their limitations.**
Linear neurons, also known as linear activation functions, are the simplest type of activation functions used in artificial neural networks. They compute a linear combination of the input values and their corresponding weights, optionally adding a bias term. Mathematically, the output of a linear neuron can be represented as:

Output = (Weighted Sum of Inputs) + Bias

Here, the weighted sum of inputs refers to the dot product of input values and their respective weights, and the bias is an additional constant term added to the output.

However, linear neurons have several limitations that make them unsuitable for many tasks and network architectures:

1. *Lack of Non-Linearity*: The most significant limitation of linear neurons is their inability to introduce non-linearity into the network. Neural networks derive their power from their capacity to model complex relationships in data, which often involve non-linear patterns. Using only linear neurons results in a network that can only represent linear transformations of the input data, severely limiting its expressive capability.

2. *Inability to Approximate Complex Functions*: Many real-world problems require the approximation of complex functions with non-linear behavior. Linear neurons are incapable of capturing these complex relationships, making them unsuitable for tasks like image and speech recognition, natural language processing, and more.

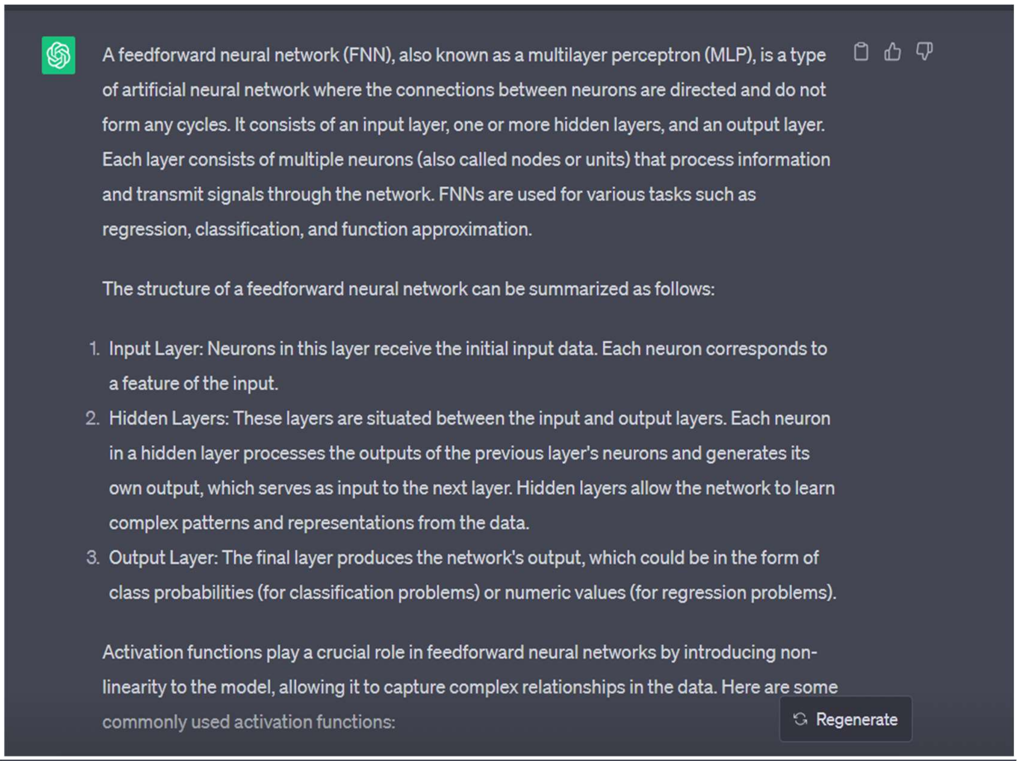
3. *Vanishing Gradient Problem*: When linear neurons are used in deep networks (networks with many layers), they suffer from the vanishing gradient problem. The gradients during backpropagation become extremely small as they are multiplied by the weights in each layer. This can lead to slow convergence or even the inability of the network to learn effectively.

4. *Limited Representation Power*: Linear neurons can only perform linear transformations, such as scaling, rotation, and translation, on the input data. They cannot handle more intricate transformations required for tasks like feature extraction, hierarchical pattern recognition, and feature learning.

5. *Difficulty in Learning Complex Patterns*: Since linear neurons can't capture complex patterns, networks with only linear neurons would require an enormous number of neurons and layers to approximate even relatively simple non-linear functions. This would result in increased computational complexity and training time.

To overcome these limitations, neural networks incorporate non-linear activation functions, such as ReLU, sigmoid, tanh, and their variants. These activation functions introduce the necessary non-linearity, enabling neural networks to approximate complex functions and learn intricate patterns from data. Non-linear activation functions play a pivotal role in the success of deep learning, allowing networks to model a wide range of functions and solve a variety of tasks effectively.

4) Describe feed forward neural networks and different activation functions that can be used in neurons.

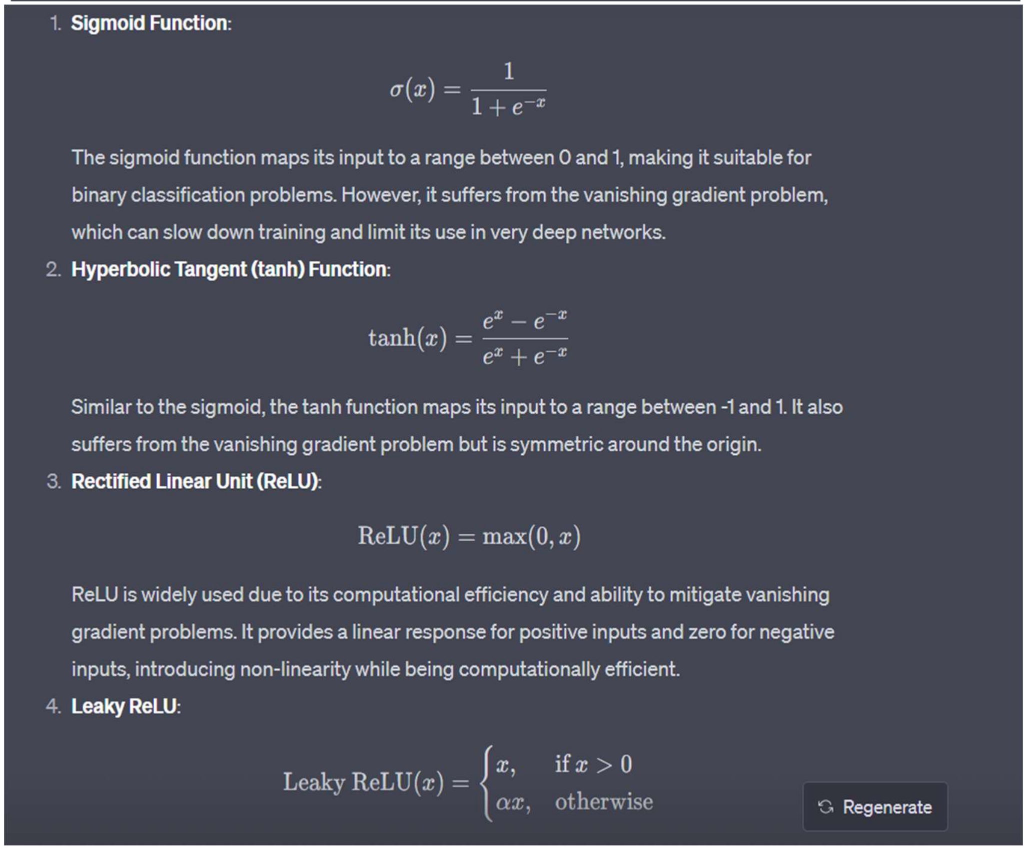


A feedforward neural network (FNN), also known as a multilayer perceptron (MLP), is a type of artificial neural network where the connections between neurons are directed and do not form any cycles. It consists of an input layer, one or more hidden layers, and an output layer. Each layer consists of multiple neurons (also called nodes or units) that process information and transmit signals through the network. FNNs are used for various tasks such as regression, classification, and function approximation.

The structure of a feedforward neural network can be summarized as follows:

1. Input Layer: Neurons in this layer receive the initial input data. Each neuron corresponds to a feature of the input.
2. Hidden Layers: These layers are situated between the input and output layers. Each neuron in a hidden layer processes the outputs of the previous layer's neurons and generates its own output, which serves as input to the next layer. Hidden layers allow the network to learn complex patterns and representations from the data.
3. Output Layer: The final layer produces the network's output, which could be in the form of class probabilities (for classification problems) or numeric values (for regression problems).

Activation functions play a crucial role in feedforward neural networks by introducing non-linearity to the model, allowing it to capture complex relationships in the data. Here are some commonly used activation functions:



1. **Sigmoid Function**:

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

The sigmoid function maps its input to a range between 0 and 1, making it suitable for binary classification problems. However, it suffers from the vanishing gradient problem, which can slow down training and limit its use in very deep networks.

2. **Hyperbolic Tangent (tanh) Function**:

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Similar to the sigmoid, the tanh function maps its input to a range between -1 and 1. It also suffers from the vanishing gradient problem but is symmetric around the origin.

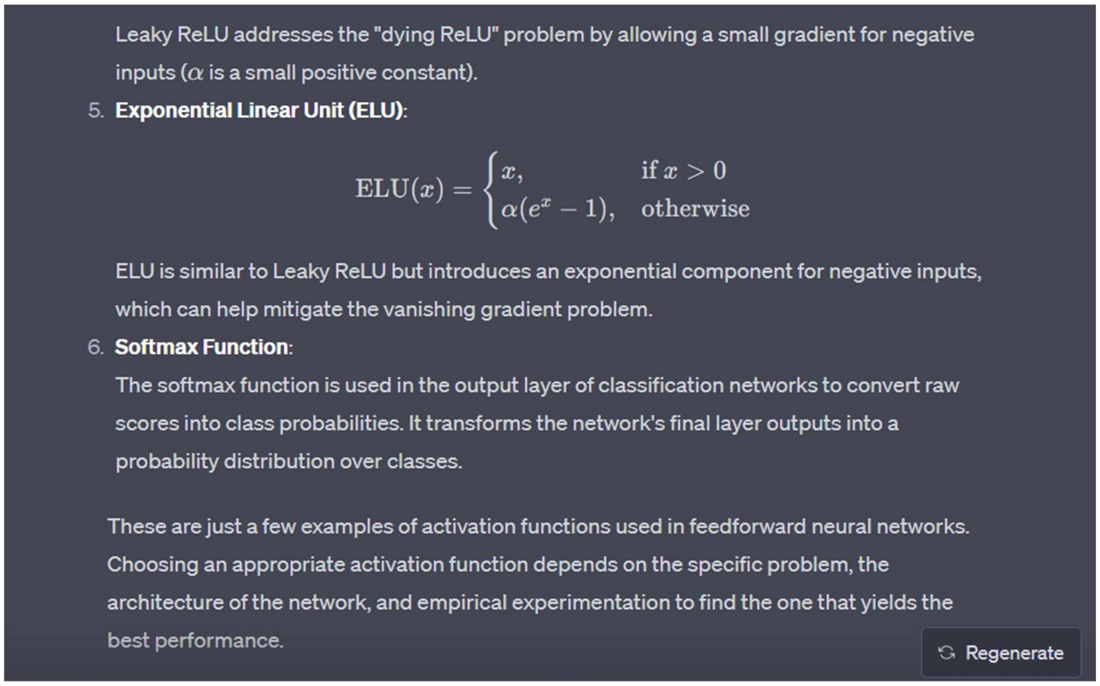
3. **Rectified Linear Unit (ReLU)**:

$$\mathrm{ReLU}(x) = \max(0, x)$$

ReLU is widely used due to its computational efficiency and ability to mitigate vanishing gradient problems. It provides a linear response for positive inputs and zero for negative inputs, introducing non-linearity while being computationally efficient.

4. **Leaky ReLU**:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{otherwise} \end{cases}$$


Leaky ReLU addresses the "dying ReLU" problem by allowing a small gradient for negative inputs (α is a small positive constant).

5. **Exponential Linear Unit (ELU)**:

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

   ELU is similar to Leaky ReLU but introduces an exponential component for negative inputs, which can help mitigate the vanishing gradient problem.

6. **Softmax Function**:
   The softmax function is used in the output layer of classification networks to convert raw scores into class probabilities. It transforms the network's final layer outputs into a probability distribution over classes.

These are just a few examples of activation functions used in feedforward neural networks. Choosing an appropriate activation function depends on the specific problem, the architecture of the network, and empirical experimentation to find the one that yields the best performance.