**1**. A neural network is a computational model inspired by the way biological neural systems (like the human brain) process information. It consists of layers of interconnected nodes (neurons), where each neuron processes input data and produces an output. Neural networks are used in machine learning to recognize patterns, classify data, and make predictions by learning from examples. They can be applied to tasks such as image recognition, speech processing, and natural language understanding.

**2**. Neurons in neural networks are computational units that take inputs, apply a weighted sum, and pass the result through an activation function to produce an output. They simulate the function of biological neurons by processing information and transmitting it to the next layer in the network. Neurons are the fundamental building blocks of neural networks, allowing them to learn complex relationships between input data and output predictions.

**3**. An activation function is a mathematical function that determines the output of a neuron based on its input. It introduces non-linearity into the network, enabling the model to learn complex patterns and make more accurate predictions. Without an activation function, a neural network would simply behave like a linear model, limiting its ability to solve complex problems. Common activation functions include ReLU, Sigmoid, and Tanh.

**4**. Backpropagation is a training algorithm used to optimize the weights of a neural network. It works by calculating the error between the network’s predicted output and the actual target output, then propagating this error backward through the network to adjust the weights. This process is repeated iteratively during training, helping the network learn by minimizing the error using gradient descent or other optimization methods.

**5**. Layers in a neural network refer to the groupings of neurons that process information at different stages of the model. The input layer receives the raw data, the hidden layers process this data through various transformations, and the output layer produces the final result. Each layer’s neurons are connected to the neurons in the previous and subsequent layers, allowing information to flow through the network.

**6**. Weights and biases control how much influence each neuron has on the final output. Weights are the parameters that scale the inputs to a neuron, while biases allow the activation function to shift its threshold. By adjusting the weights and biases during training, the network can learn the optimal mappings between inputs and outputs, thereby improving its predictions.

**7**. Overfitting occurs when a neural network learns the training data too well, including noise and irrelevant patterns, causing it to perform poorly on unseen data. This happens when the model becomes too complex relative to the amount of training data available. To prevent overfitting, techniques such as cross-validation, dropout, and regularization can be used to ensure the model generalizes well to new data.

Part 2: Activation Functions

**1**. Mathematical Formula:

f(x) =

\begin{cases}

x, & \text{if } x \geq 0 \\

\alpha x, & \text{if } x < 0

\end{cases}

**2**. Behavior of the Activation Function:

The Leaky ReLU function behaves similarly to the standard ReLU function but with a modification for negative inputs. While ReLU outputs 0 for any negative input, Leaky ReLU allows a small negative slope, determined by the parameter . This prevents the "dying ReLU" problem, where neurons become inactive and stop learning during training. The function is piecewise linear, providing non-linearity for better learning.

**3**. Where and Why It’s Used:

Leaky ReLU is used in situations where ReLU might cause neurons to become inactive during training (e.g., when the input is always negative). It is particularly useful in deep networks, where a large number of layers may cause the vanishing gradient problem. Since Leaky ReLU ensures that gradients flow even for negative values, it allows for better optimization in deep networks. It is also preferred over the Sigmoid or Tanh functions in hidden layers due to its simpler computation and avoidance of gradient saturation.

**4**. Advantages and Disadvantages:

Advantages:

1. Helps mitigate the dying ReLU problem by allowing small negative values.
2. Faster computation compared to Sigmoid or Tanh.
3. Can handle deep networks more effectively due to improved gradient flow.

Disadvantages:

The negative slope parameter might need tuning for optimal performance. If is too large, it can still allow too much negative activation, which might hurt performance. It can still lead to some issues with optimization in certain cases (e.g., if neurons are heavily negative).

**5**. Real-World Application:

Leaky ReLU is commonly used in deep convolutional neural networks (CNNs) for image recognition tasks. For instance, in a network trained to classify images, Leaky ReLU allows the model to learn from both positive and negative feature activations, preventing dead neurons from hindering learning. This has been shown to improve model performance in tasks such as object detection, where complex and varied features are extracted from images.