**1. Explain the architecture of Spark**

Apache Spark is an open-source distributed computing system designed for processing and analyzing large-scale data sets. Its architecture is built around the concept of Resilient Distributed Datasets (RDDs), which are distributed collections of data that can be operated on in parallel across a cluster of machines.

**Driver Program:** The Spark driver program is the main control program that coordinates the execution of Spark jobs. It contains the user's SparkContext, which is the entry point to the Spark API. The driver program runs the main function and creates RDDs, applies transformations, and invokes actions on those RDDs.

**Cluster Manager:** Spark can run on various cluster managers like Apache Mesos, Hadoop YARN, or its own built-in standalone cluster manager. The cluster manager allocates resources (CPU, memory) to Spark applications and manages the execution of tasks on the worker nodes.

**Worker Nodes:** Worker nodes are the machines in the Spark cluster where the actual computation takes place. Each worker node runs an executor process that executes tasks and stores data in memory or on disk. Executors are managed by the cluster manager and are responsible for running individual tasks and storing intermediate data.

**RDDs (Resilient Distributed Datasets):** RDDs are the fundamental data abstraction in Spark. They represent distributed collections of objects that can be operated on in parallel. RDDs are immutable and fault-tolerant, meaning that if a partition of an RDD is lost, it can be recomputed using the lineage information stored by Spark.

**Transformations:** Transformations in Spark are operations that create a new RDD from an existing one. Examples of transformations include map, filter, join, and groupByKey. Transformations are lazy, meaning that they do not compute results immediately but instead build up a directed acyclic graph (DAG) of transformations to be executed later.

**Actions:** Actions in Spark are operations that trigger the execution of the DAG of transformations and return results to the driver program or write data to external storage. Examples of actions include collect, count, reduce, and save.

**2. Explain activation function**

An activation function, within the context of artificial neural networks and deep learning, is a mathematical operation applied to the output of a neuron or a layer of neurons. The purpose of an activation function is to introduce non-linearity into the network, enabling it to learn complex patterns in the data.

**Type of activation function are:-**

1. **Linearity vs. Non-linearity:** A linear function produces outputs that are directly proportional to its inputs, following a straight line. In real-world data is non-linear, meaning the relationship between inputs and outputs is not directly proportional and cannot be adequately captured by linear functions alone. Activation functions introduce non-linearity into neural networks, allowing them to model and learn complex relationships in the data.
2. **Sigmoid Activation Function:** One of the earliest activation functions used in neural networks is the sigmoid function, which maps input values to the range (0, 1). It has a characteristic S-shaped curve and is often used in binary classification tasks. However, it suffers from the vanishing gradient problem, where gradients become extremely small for large input values, making training slow and prone to convergence issues.
3. **Hyperbolic Tangent (tanh) Activation Function:** Similar to the sigmoid function, the tanh function maps input values to the range (-1, 1). It is also S-shaped but centered around zero, which helps mitigate the vanishing gradient problem to some extent. Tanh activation functions are commonly used in hidden layers of neural networks.
4. **Rectified Linear Unit (ReLU) Activation Function:** ReLU is one of the most widely used activation functions in deep learning. It returns the input value if it is positive, and zero otherwise. ReLU activation functions introduce sparsity and have faster convergence during training compared to sigmoid and tanh functions. However, they suffer from the "dying ReLU" problem, where neurons can become inactive for certain input ranges, leading to dead neurons that do not contribute to the learning process.
5. **Leaky ReLU:** Leaky ReLU addresses the dying ReLU problem by allowing a small gradient for negative inputs, preventing neurons from becoming completely inactive.
6. **The Exponential Linear Unit (ELU)** activation function is a type of activation function used in artificial neural networks, particularly in deep learning architectures. It is designed to address some of the limitations of other activation functions like ReLU (Rectified Linear Unit), particularly the "dying ReLU" problem, where neurons can become inactive for certain input ranges.

**3. List different types of activation function with their formula**

Here are several different types of activation functions commonly used in neural networks, along with their mathematical formulas:

**Sigmoid Activation Function:**

Formula:*●* X=1/1+e^-x

● RANGE : (0,1)

**Hyperbolic Tangent (tanh) Activation Function:**

Formula:*● TANh(X) = e^x-e^-x / e^x+e^-x*

*● RANGE : (-1,1)*

**Rectified Linear Unit (ReLU) Activation Function:**

Formula:*f*(*x*)=max(0,*x*)

● RANGE : [0,+INFINITY)

**Leaky ReLU Activation Function:**

Formula:● LRELU (X) = X IF X>0 , AX OTHERWISE

● RANGE : (-INFINITY, +INFINITY)

**Exponential Linear Unit (ELU) Activation Function:**

Formula:● ELU(X) = X IF X>0, A(E^X-1) OTHERWISE

● RANGE : (-A, +INFINITY)

**4. Explain Hybrid Inheritance with Code.**

Hybrid inheritance is a combination of multiple types of inheritance, typically combining features of both single and multiple inheritances. In hybrid inheritance, classes inherit from multiple base classes through a combination of single and multiple inheritance.

**5. Explain Neural Networks**

Neural networks are a class of machine learning models inspired by the structure and function of the human brain. They consist of interconnected nodes, called neurons or units, organized into layers. Neural networks are capable of learning complex patterns and relationships within data, making them powerful tools for various tasks such as classification, regression, clustering, and more.

**1.Basic Structure:**

* Neurons: Neurons are the basic building blocks of neural networks. Each neuron receives input signals, performs a computation on these inputs, and produces an output signal. Neurons are typically organized into layers.
* Layers: Neurons within a neural network are organized into layers. The three main types of layers are:
  + Input Layer: The first layer of the neural network that receives input data.
  + Hidden Layers: Intermediate layers between the input and output layers. They perform complex transformations on the input data.
  + Output Layer: The final layer of the neural network that produces the model's output.

**2.Connections:**

* + Connections between neurons carry information in the form of numerical values, often referred to as weights. Each connection has an associated weight that determines the strength of the connection.
  + During the training process, the weights of these connections are adjusted to minimize the difference between the predicted output and the actual output.

**3.Activation Function:**

* + Activation functions introduce non-linearity into the neural network, allowing it to learn complex patterns in the data.
  + Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and variants like Leaky ReLU and ELU (Exponential Linear Unit).

**4.Feedforward and Backpropagation:**

* Feedforward: During the feedforward pass, input data is propagated through the network, layer by layer, to produce the output. Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.
* Backpropagation: Backpropagation is the algorithm used to train neural networks. It involves computing the gradient of a loss function with respect to the network's weights and adjusting the weights using gradient descent or its variants to minimize the loss.