**CS624 MIDTERM**

**Q1.** **What is a Big Data System, and how does Apache Spark fit into the Big Data Ecosystem? Give an example to back up your explanation?**

**Solution:-** A Big Data System refers to the infrastructure and technologies designed to handle and process large volumes of data, typically characterized by the three Vs: Volume, Velocity, and Variety. These systems are crucial for organizations dealing with massive amounts of data to extract valuable insights and make data-driven decisions. The Big Data Ecosystem comprises various tools, frameworks, and platforms that work together to manage, process, and analyze data efficiently.

Apache Spark is a powerful open-source, distributed computing system that plays a significant role in the Big Data Ecosystem. Below are the points which helps us to understand the role of Spark in Big Data Eco system.

**Data Processing:** Spark allows parallel data processing across a cluster of computers, enabling efficient handling of large datasets. Its in-memory processing capabilities significantly speed up iterative algorithms and interactive queries compared to traditional batch processing frameworks like Apache Hadoop MapReduce.

**Ease of Use:** Spark provides high-level APIs in languages like Scala, Java, Python, and R, making it accessible to a wide range of developers. It also includes libraries for data analysis (Spark SQL), machine learning (MLlib), graph processing (GraphX), and stream processing (Structured Streaming), making it a versatile tool for various data processing tasks.

**Integration:** Spark seamlessly integrates with other components in the Big Data Ecosystem. It can run on Hadoop Distributed File System (HDFS) and can also read data from other data sources, including HBase, Apache Hive, and Apache Cassandra. Spark can be used in combination with tools like Apache Kafka for real-time data streaming.

**Example:** A financial institution have large number of customer and merchant across the globe, the company wants to analyze customer purchase data to identify trends and improve marketing strategies. The company collects vast amounts of transaction data, including customer information, product details, and purchase history. In this case, Apache Spark can be used to process and analyze the data efficiently. Spark's capabilities enable the company to perform tasks like aggregating sales data, identifying popular products, and running machine learning algorithms to recommend products to customers in real-time. The ability to handle large-scale data processing, support for diverse workloads, and integration with other Big Data tools make Apache Spark a valuable component in the company's Big Data System.

**Q2.** **Explain Mlib from Apache Spark and how it differs from Sklearn's implementation of machine learning techniques.**

**Solution:-** MLlib (Machine Learning Library) is a component of Apache Spark that provides scalable and distributed machine learning algorithms. It is designed to work seamlessly with Spark's core functionality, enabling the processing of large-scale datasets across a cluster of machines. MLlib is written in Scala and offers APIs for multiple programming languages, including Java, Scala, Python, and R.

1.Spark's MLlib is specifically designed for distributed computing. It can efficiently handle large datasets by distributing the computation across a cluster of machines. Scikit-learn, on the other hand, is primarily designed for single-machine environments. While it's powerful for smaller datasets that fit into memory, it may face challenges when dealing with very large datasets that require distributed computing.

2. MLlib benefits from the Apache Spark community and ecosystem, providing a wide range of tools for big data processing and analytics. Scikit-learn has a strong Python machine learning community and is widely used in traditional machine learning environments.

3. While MLlib's primary language is Scala, it provides APIs for Java, Scala, Python, and R, making it accessible to a broader audience. Scikit-learn is primarily a Python library, and its APIs are designed for Python developers.

4. MLlib provides a set of machine learning algorithms for classification, regression, clustering, collaborative filtering, and dimensionality reduction. It includes tools for feature extraction, transformation, and model evaluation. However, the range of algorithms might be more limited compared to scikit-learn. Scikit-learn is a comprehensive machine learning library with a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and more. It offers a rich set of tools for model selection, evaluation, and hyperparameter tuning.

5. MLlib seamlessly integrates with the broader Spark ecosystem, allowing users to leverage Spark's capabilities for data preprocessing, feature engineering, and distributed computing. Scikit-learn is a standalone library and doesn't have native integration with distributed computing frameworks like Spark.

**Q3.What is a cluster? Why do you need a cluster for Apache Spark?**

**Solution:-** A cluster refers to a group of interconnected computers (nodes) that work together to perform computing tasks. Each node in the cluster contributes its processing power, memory, and storage, creating a unified and scalable computing environment. Clusters are widely used in various fields to handle large-scale data processing, parallel computing, and distributed storage.

Apache Spark needs a cluster for the following reasons:

**Scalability:** Clusters provide a scalable infrastructure that allows organizations to expand their computing resources easily. As the volume of data grows or the complexity of computations increases, additional nodes can be added to the cluster to handle the load. This scalability is crucial for processing big data efficiently.

**Parallel Processing:** Apache Spark is designed to perform parallel processing on distributed data. By distributing data and computations across multiple nodes in a cluster, Spark can process large datasets much faster than a single machine. This parallelism is achieved through tasks executed in parallel on different nodes, leading to improved performance and reduced processing times.

**Fault Tolerance:** Clusters provide fault tolerance, ensuring the system's resilience in the face of node failures or other issues. Apache Spark employs a resilient distributed dataset (RDD) abstraction, which allows it to recover lost data by recomputing it from the original source or replicas stored on other nodes. This fault tolerance mechanism enhances the reliability of data processing in a distributed environment.

**Resource Management:** Cluster managers, such as Apache Hadoop YARN or Apache Mesos, help manage resources in a distributed environment. They allocate tasks to nodes, monitor resource usage, and ensure optimal utilization of the cluster's computing resources. This centralized management is critical for efficient and coordinated execution of tasks across the cluster.

**Data Distribution:** Large datasets are distributed across the nodes in a cluster, allowing Spark to leverage data locality. Data locality minimizes data transfer over the network by performing computations on the nodes where the data resides. This results in reduced data movement and faster processing times compared to a scenario where data needs to be transferred between nodes.

**Memory Sharing and Caching:** In-memory processing is a key feature of Apache Spark, allowing it to store intermediate data in memory for faster access. Clusters provide a shared memory space that enables Spark to cache and share data across nodes. This shared memory architecture enhances performance by reducing the need to read data from disk repeatedly.

**Q4.What are the differences between RDD, Spark Dataframe, and Spark SQL?**

Solution:- RDD is the fundamental data structure in Spark, representing an immutable distributed collection of objects that can be processed in parallel. RDDs are fault-tolerant, meaning they can recover from node failures by recomputing lost data using lineage information. RDD operations are evaluated lazily, and transformations on RDDs are only executed when an action is triggered.

DataFrame is a higher-level abstraction built on top of RDD, providing a distributed collection of data organized into named columns. Spark DataFrames offer a more user-friendly API similar to traditional databases or data frames in programming languages like R or Python's Pandas. DataFrames are also fault-tolerant and can leverage Spark's Catalyst optimizer for query optimization.

Spark SQL is a module in Spark for structured data processing, providing a programming interface for querying structured and semi-structured data. It includes a SQL query parser, a built-in DataFrame API, and support for Hive queries, enabling integration with existing Hive-based workflows. Spark SQL allows users to seamlessly mix SQL queries with Spark programs, providing flexibility in how data is processed.

**Key Differences:**

**Abstraction Level:**

RDD is a low-level abstraction, providing fine-grained control over distributed data and computation. DataFrame and Spark SQL offer higher-level abstractions, allowing users to express computations more concisely and in a more SQL-like manner.

**Optimization:** RDD operations are not automatically optimized, and users need to manage optimization explicitly. DataFrames and Spark SQL leverage the Catalyst optimizer to optimize query plans and provide better performance.

**Ease of Use:** RDDs require more manual intervention for optimization and expressiveness. DataFrames and Spark SQL offer a more user-friendly API with a higher level of abstraction, making them easier to use, especially for those familiar with SQL or traditional data frames.

**Q5. What's the difference between a SPARK SQL Group by and window function with Partitioning?**

**Solution:-**  The difference between spark sql and window function with partitioning are as follows:-

The result of GROUP BY is a reduced set of rows where each row represents an aggregated group based on the specified columns. The result of window functions with partitioning maintains the original number of rows, and additional columns are added to each row based on the window function calculations.

Group by typically used for simple aggregations like SUM, AVG, COUNT, etc. Windows functions allow for more complex aggregations and calculations like running totals, rankings, and custom aggregations within a specified window frame.

Group by deduces granularity by grouping rows with similar values into a summarized result set. Window functions maintains the original row-level granularity and adds calculated values based on the window specification within each partition.

Group by produces a new DataFrame or table with aggregated values for each group. Window function retains the original structure of the DataFrame but includes additional columns with the window function results.

Group by suitable for summarizing data and obtaining aggregated statistics for specific groups. Window function Useful for performing row-level calculations within partitions, such as comparing a value to the average within its partition.

Group by can be computationally expensive, especially with large datasets, as it involves shuffling and aggregating data across the entire dataset. Can be more efficient in certain scenarios, as they operate within partitions and may not require shuffling across the entire dataset.

Group by Commonly used for traditional data summarization and aggregation tasks. Winddows function Beneficial for tasks requiring context-aware calculations within specified partitions.

**Q6.Discuss any two performance tuning techniques for Spark?**

**Solution:-**  The two major performance tuning techniques are as follows:

**Partitioning:** Overview: Efficient data partitioning is critical for optimizing Spark job performance. Partitioning determines how data is distributed across the nodes in a cluster, affecting both data processing and communication costs.

**Hash Partitioning:** It involves distributing data based on a hash function applied to a partition key. This method ensures that related data is likely to be present in the same partition, reducing shuffling costs during join operations.

**Range Partitioning:** Data is divided based on a specified range of values. This is useful when the data naturally has a clear ordering, and operations like range queries can benefit from this partitioning strategy.

**Custom Partitioning:** Users can implement their custom partitioning logic by extending the Partitioner class in Spark. This allows for tailored partitioning strategies based on the characteristics of the data and the specific requirements of the job.

**Caching and Persistence:** Spark provides the ability to cache and persist intermediate or final results in memory or on disk. Caching is particularly beneficial when there are multiple stages in a Spark job that require the same dataset, reducing the need to recompute the same data multiple times.

**Memory Storage Levels:** Spark allows you to choose different storage levels for caching, ranging from MEMORY\_ONLY to MEMORY\_ONLY\_SER (serialized) or MEMORY\_AND\_DISK. The choice depends on the size of the data, computation cost, and the available memory on the cluster nodes.

**Unpersisting:** It's important to unpersist or uncache RDDs/DataFrames when they are no longer needed. This releases the memory resources, avoiding potential memory issues on the cluster.

**Broadcast Variables:** Instead of relying on caching, smaller read-only datasets that are used across tasks can be broadcasted to all the worker nodes. This helps in reducing data transfer costs during the execution of tasks.

**Q7. What is Pyspark and how does it is related to the Apache Spark?**

**Solution:-** PySpark is the Python API for Apache Spark, a fast and general-purpose cluster computing system for big data processing. PySpark allows Python developers to harness the power of Spark for large-scale data processing, machine learning, graph processing, and more, using familiar Python programming constructs.

**Python API for Apache Spark:** PySpark enables Python developers to interact with Apache Spark using Python programming language.

It provides a high-level API for distributed data processing, making it easier for Python developers to leverage the capabilities of Spark.

**Integration with Spark Ecosystem:** PySpark seamlessly integrates with the broader Spark ecosystem, which includes components like Spark SQL for structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for real-time data processing. Users can utilize the functionalities of these components within PySpark, making it a comprehensive solution for various data processing tasks.

**Programming Flexibility:** PySpark allows developers to write Spark applications using Python, a popular and widely used programming language known for its readability and ease of use. This is particularly beneficial for data scientists and analysts who are proficient in Python and want to leverage Spark's capabilities without having to learn a new programming language like Scala (which is the native language for Apache Spark).

**DataFrames and RDDs:** PySpark supports both Resilient Distributed Datasets (RDDs) and DataFrames. RDDs are the fundamental data structure in Spark, and PySpark provides Python-friendly APIs for working with RDDs.DataFrames, introduced in Spark 1.3, provide a higher-level abstraction built on top of RDDs, making it easier to express complex data manipulations. PySpark users can work with DataFrames using Python syntax.

**Execution Engine:** Despite using Python for high-level programming, PySpark jobs are executed on the same Spark engine that executes jobs written in Scala or Java. This ensures that PySpark applications can achieve similar performance characteristics as those written in the native Spark languages.

**Q8. What is streaming in the context of big data? What's the difference between watermarking and windowing?**

**Solution:-** Streaming in the context of big data refers to the processing of continuously flowing data in real-time or near-real-time. This data can be generated from various sources, such as sensors, social media feeds, log files, or any other event-driven system. Instead of processing data in traditional batch-oriented fashion, streaming systems handle data as it arrives, enabling rapid analysis, monitoring, and decision-making.In a streaming architecture, data is processed in small, time-bounded units or micro-batches. Streaming frameworks, like Apache Spark Streaming, Apache Flink, and Apache Kafka Streams, provide the infrastructure to process and analyze data in real-time, offering features such as event time processing, windowing, and watermarks to handle the challenges of streaming data.

The difference between watermarking and windowing are as under:-

**Watermarking:-** Watermarking is a technique used in streaming systems to handle event time in data streams. Event time refers to the time when an event actually occurred in the real world, as opposed to the time when it is processed by the system.

Watermarking is used to manage and account for the variability in event arrival times. It helps in addressing issues such as late-arriving events and out-of-order events.

A watermark is essentially a timestamp that represents the progress of event time in the data stream. It acts as an indicator of the completeness of the data processed up to that point.

Watermarks are often used in conjunction with windowing to define the completeness of a time window. Events with timestamps earlier than the watermark are considered complete and can be processed, while events with timestamps after the watermark may still arrive.

**Windowing:-** Windowing is a concept in streaming systems that involves dividing the continuous data stream into finite, overlapping, or non-overlapping time intervals called windows.

Windowing allows the system to group and process data in batches based on time intervals, enabling aggregation, summarization, and analysis over specific periods.

Windows can be tumbling (non-overlapping) or sliding (overlapping), and they are defined by the start and end times. The processing of data within a window is typically carried out using operations like aggregations or computations.

Windowing is useful for various tasks, including computing statistics over time, monitoring trends, and generating results at regular intervals. It is often employed alongside watermarks to handle event time considerations.

**Q9. What is Databricks, and how is it different from Apache Spark?**

Solution:- Databricks is a cloud-based big data analytics platform that is built on top of Apache Spark. It provides an integrated environment for data engineers, data scientists, and analysts to collaborate and work with large-scale data processing and analytics. While Apache Spark is the open-source distributed computing framework, Databricks adds additional features and a user-friendly interface to simplify the deployment, management, and usage of Spark clusters.

The main difference between Databricks and Apache Spark are as follows:

1.Databricks is a cloud-based platform provided as a service (PaaS) on cloud providers such as AWS, Azure, and Google Cloud. Databricks manages the underlying infrastructure, making it easier to deploy and scale Spark clusters without worrying about cluster configuration, maintenance, or optimization. Apache Spark is an open-source distributed computing framework that users need to install, configure, and manage on their own infrastructure or cloud instances.

2.Databricks offers a collaborative workspace where data teams can work together seamlessly. The platform includes features like notebooks for code collaboration, data visualization, and collaboration tools that facilitate teamwork and knowledge sharing. Spark provides APIs in various programming languages (Scala, Java, Python, and R), it doesn't inherently provide a collaborative environment. Users need to set up their own tools for code sharing and collaboration.

3. Databricks includes integrated services like Databricks SQL for SQL-based queries, Databricks ML for machine learning, and Databricks Delta for optimized data management and storage. These services aim to streamline the end-to-end data processing and analytics workflow. Spark, as an open-source project, provides core libraries for distributed data processing, machine learning (MLlib), graph processing (GraphX), and SQL-based querying (Spark SQL). Users can integrate additional libraries as needed.

4.Databricks is designed to provide an easy-to-use interface for users with varying skill levels, making it accessible to data scientists, analysts, and engineers. Databricks notebooks, for example, offer an interactive environment for code development and exploration. Spark's native interface might be considered more complex for users who are not experienced in distributed computing. It requires more manual configuration and management.

5. The Databricks platform often includes optimizations and enhancements beyond the open-source Spark project. Databricks Delta, for instance, provides improved data reliability, performance optimizations, and transactional capabilities on top of Spark. The open-source nature of Spark allows the community to contribute to its development and improvement. However, users need to stay up-to-date with the latest releases and may need to implement additional optimizations themselves.

**Q10. What is the difference between the keywords cache and persist in the context of Spark?**

**Solution:-** CACHE and PERSIST do the same job to help in retrieving intermediate data used for computation quickly by storing it in memory, while by caching we can store intermediate data used for calculation only in memory , persist additionally offers caching with more options/flexibility. Persist can be thought of as flexible caching.

or CACHING = PERSIST in Memory only

Persist has 5 options (DISK,MEMORY,OFF-HEAP,DESERIALIZED,NO\_OF\_REPLICAS)

1) for storage level **(MEMORY\_ONLY,DISK\_ONLY,MEMORY\_AND\_DISK)**

2) **Off-heap option**: imagine we have a worker node of 64GB 16 CPU Cores, which in turn has 3 executors 20GB and 5 CPU cores each. the remainder memory that is: 64GB - (20GB\*3) = 4GB is the off-heap memory (outside the executors/containers but within the node)

3) for the form in which data can be stored in memory and Disk. Cached Data can be stored in binary(**serialized**) or object format(**de-serialized**). Memory can store data in both serialized and de-serialized format , whereas on Disk we can store data only in serialized format. that's why the option provided is always w.r.t Memory

**serialized** data takes lesser space but more CPU cycles whereas **de-serialized** data takes more space but is much quicker for data retrieval

4) **No of replicas** of data you want to have on disk or memory

Persist options example:

**persist(True, True, False, False, 2)** -> store on memory and remainder if any on disk , don't use off-heap memory, store data in serialized form, 2 replicas of data.

.persist() can also be used with dataframes with StorageLevel options:

dataframe.persist(StorageLevel.DISK\_ONLY) --> store on disk only

dataframe.persist(StorageLevel.MEMORY\_ONLY) --> store on memory only

dataframe.persist(StorageLevel.MEMORY\_AND\_DISK\_SER) -> store in memory and then in disk if remaining data in **Serialized** format

dataframe.persist(StorageLevel.MEMORY\_AND\_DISK\_SER\_2) -> same as previous example but with 2 replicase of data

**Q11. What's the difference between Spark Context and Spark Session?**

**Solution:-**  The main difference between SparkContext and SparkSession are as follows:

* SparkContext is the entry point and the main interface for interacting with Spark. It represents the connection to a Spark cluster and coordinates the execution of Spark jobs.
* In earlier versions of Spark, you would create a SparkContext to initialize Spark. However, starting from Spark 2.0, the preferred entry point is to use SparkSession instead of creating a SparkContext directly.
* SparkContext is responsible for tasks such as distributing the application code to the cluster, acquiring resources (executors), managing the execution of tasks, and handling communication with the cluster manager (e.g., Apache Mesos, Apache Hadoop YARN, or Spark's standalone cluster manager).
* SparkSession is a higher-level API introduced in Spark 2.0 to simplify and unify the usage of various Spark components. It encompasses the functionality of SparkContext and adds support for working with structured data using DataFrames and Datasets.
* In Spark 2.0 and later, you typically create a SparkSession as the entry point for your Spark application. It encapsulates the functionality of SparkContext and provides additional features for working with structured and semi-structured data.
* SparkSession simplifies the process of working with Spark by providing a unified interface for creating DataFrames, executing SQL queries, and interacting with various Spark components. It also manages the underlying SparkContext for you.

**Q12. What is a DAG in the context of Spark. What's the difference between a DAG, JOB, STAGE, and TASK?**

**Solution:-** A Directed Acyclic Graph (DAG) represents the logical execution plan of a Spark application. It is a graph of stages and the relationships between them, illustrating the sequence of transformations and actions that Spark will execute to process the data. The DAG is a crucial part of Spark's internal execution engine and is used to optimize and schedule the tasks required to execute a Spark job.

**Difference between DAG, Job, Stage, and Task:**

**Job:**A job in Spark represents the complete computation triggered by an action in the user code. It consists of multiple stages. Each action in Spark triggers the execution of one or more jobs. Jobs are the highest-level unit of work in Spark and are composed of multiple stages.

**Stage:**A stage is a set of parallel tasks that perform the same computation but on different partitions of the data. Stages are determined by the shuffle operations in the DAG.A job is broken down into one or more stages. Stages are units of work that can be executed in parallel, and they are separated by shuffle dependencies. Stages can be further divided into tasks.

**Task:**A task is the smallest unit of work in Spark and represents the execution of a computation on a partition of the data .Each stage consists of multiple tasks, where each task corresponds to the processing of a single partition of the data. Tasks are the actual units of work that are sent to the executors for execution.

**DAG (Directed Acyclic Graph):**The DAG is the logical representation of the entire sequence of transformations and actions in a Spark application. It is a directed acyclic graph where vertices represent RDDs (Resilient Distributed Datasets) and edges represent the transformations between RDDs. The DAG is constructed when transformations are applied to RDDs, and it represents the sequence of transformations that Spark will execute when an action is triggered. The DAG is optimized by Spark's Catalyst optimizer and Tungsten execution engine to improve performance.

**Q13. What is concept drift or data drift, and how does it affect machine learning models?**

**Solution:-** This is the phenomenon where the statistical properties of the target variable or input features in a machine learning model change over time. In other words, the relationships between the model inputs and outputs evolve, making the model's assumptions less valid and potentially leading to a decrease in predictive performance.

**Sudden Concept Drift:-** This occurs when there is an abrupt and significant change in the data distribution. The relationship between input features and the target variable undergoes a sudden shift.

**Incremental Concept Drift:-** This type of concept drift involves a gradual and continuous change in the data distribution over time. The model needs to adapt to the evolving patterns to maintain accurate predictions.

**Effects of Concept Drift on Machine Learning Models are as follows:-**

* **Model Performance Degradation**
* **Increased Prediction Errors**
* **Model Obsolescence**
* **Need for Continuous Monitoring and Adaptation**
* **Decision-Making Impact**

**Q14. What's the difference between a relational database like MySQL, a timeseries database, and a graph database like Neo4J? You only need to explain one or two differences**

**Solution: -**

* Relational Database (e.g., MySQL):
  + Data Model: Relational databases follow the tabular data model. Data is organized into tables with rows and columns, and relationships between tables are established using foreign keys.
  + Use Cases: Relational databases are suitable for applications with structured and well-defined data, where relationships between entities are primarily one-to-one or one-to-many. They are commonly used in traditional business applications and transactional systems.
  + Query Language: SQL (Structured Query Language) is used for defining and manipulating the data in relational databases.
* Timeseries Database:
  + Data Model: Timeseries databases are specialized for handling time-stamped data. They organize data based on timestamps and typically include additional features for handling time-based queries efficiently.
  + Use Cases: Timeseries databases are well-suited for applications dealing with data generated over time, such as sensor data, financial market data, or monitoring systems. They excel at handling sequential data points and time-based analytics.
  + Query Language: Depending on the specific timeseries database, the query language may vary. Some timeseries databases use SQL extensions, while others have specialized query languages.
* Graph Database (e.g., Neo4j):
  + Data Model: Graph databases use a graph data model composed of nodes, edges, and properties. Nodes represent entities, edges represent relationships between entities, and properties provide additional information about nodes and edges.
  + Use Cases: Graph databases are ideal for applications with complex relationships and interconnected data. They excel in scenarios where relationships between entities are equally important as the entities themselves, such as social networks, fraud detection, and network analysis.
  + Query Language: Cypher is a query language specifically designed for graph databases like Neo4j. It allows expressive querying of graph structures, making it easy to traverse and analyze relationships.

**Q15. What's the difference between a perceptron, multilayer perceptron, and a deep neural network?**

**Solution:-**  The difference between a perception, multilayer perceptron and DNN are as follows:

A perceptron is the simplest form of a neural network. It consists of a single layer with one or more input nodes, a weight associated with each input, and an activation function that computes the output based on the weighted sum of inputs.

An MLP is an extension of the perceptron, introducing one or more hidden layers between the input and output layers. Each node in a hidden layer has an associated weight, and the network has non-linear activation functions.

A deep neural network is a neural network with a large number of hidden layers, allowing it to learn intricate hierarchical representations of the input data. DNNs are capable of automatically extracting features at different levels of abstraction.

A perceptron is a single-layer feedforward neural network without any hidden layers. It can be used for binary classification problems where it learns a linear decision boundary.

The presence of hidden layers enables MLPs to learn complex non-linear relationships within the data. The input is passed through the hidden layers before reaching the output layer.

DNNs can have many hidden layers, forming a deep architecture. These layers enable the network to capture and learn complex patterns, making them suitable for tasks like image and speech recognition.

The training of a perceptron involves adjusting the weights based on the error in the predictions. It uses a simple learning rule to update the weights and improve its accuracy.

MLPs use backpropagation and gradient descent algorithms for training. Backpropagation involves propagating errors backward through the network, adjusting weights to minimize the error.

Training deep neural networks can be challenging due to issues like vanishing gradients or overfitting. Techniques such as batch normalization, skip connections, and advanced optimization algorithms are often used to train deep networks effectively.

**Q16. Why utilize a deep neural network when you already have classic machine learning algorithms like logistic regression and random forest?**

**Solution:-**  The main reasons to use DNN instead of machine learning models are as follows:-

* **Complex Non-linear Relationships:**
  + DNNs are well-suited for tasks where the relationships between input features and output are highly non-linear and complex. They can automatically learn and capture intricate patterns and representations at different levels of abstraction.
* **Feature Learning:**
  + DNNs can automatically learn hierarchical representations and features from the data. This is particularly advantageous when dealing with high-dimensional and unstructured data, such as images, audio, or text. The ability to automatically extract relevant features can lead to improved performance.
* **End-to-End Learning:**
  + DNNs support end-to-end learning, allowing the model to learn a hierarchical representation of the input data directly from raw features. This eliminates the need for manual feature engineering, which is often required in traditional machine learning algorithms.
* **Performance on Large Datasets:**
  + DNNs tend to perform well on large datasets. As the amount of labeled data increases, DNNs can leverage their capacity to learn complex relationships and generalize better than simpler models.
* **Representation Learning:**
  + DNNs are capable of learning meaningful and abstract representations from the data. In tasks like image recognition, the early layers of a DNN may learn basic features like edges, textures, and colors, while deeper layers learn more complex and abstract features.
* **Availability of Computational Resources:**
  + With advancements in hardware (e.g., GPUs and TPUs) and distributed computing, training and deploying DNNs have become more feasible and efficient. This enables the use of large-scale models for improved performance.
* **State-of-the-Art Performance:**
  + In certain domains, such as computer vision and natural language processing, DNNs have achieved state-of-the-art performance on benchmark datasets. If achieving the highest accuracy or solving complex problems is a priority, DNNs may be the preferred choice.
* **Transfer Learning and Pre-trained Models:**
  + DNNs support transfer learning, where pre-trained models on large datasets can be fine-tuned for specific tasks with smaller datasets. This is beneficial when labeled data is limited, as the model can leverage knowledge gained from a broader context.

**Q17. Give one application or use of GAN**

**Solution:- Image Synthesis for Data Augmentation**

GANs can be employed to generate new, realistic images for data augmentation, providing additional training examples without the need for collecting more real-world data.

Once the GAN is trained, the generator can be used to generate new, realistic images that share similar characteristics with the training data. These generated images can be mixed with the original dataset to create an augmented dataset for training machine learning models.

In computer vision tasks, such as object detection or image classification, GAN-generated images can be used to augment the dataset, making the trained models more robust and capable of handling a wider range of scenarios.

**Q18. Imagine you're creating a Big data system for a company like Twitter. List at least five requirements that you believe might be needed to build the system and list how different tools or concepts we have been covering in class will help with you with these requirements. You only need to define requirements in the context of Big Data and explain why the tool/concept can help you meet those requirements. You can choose your own company, but the use case should be big data.**

**Solution:-**  The details for the solution are as follows:

**Scalability:-** Scalability is crucial for handling the massive and growing volume of data generated by millions of tweets daily. The system should be able to scale horizontally by adding more resources (nodes) to accommodate increasing data loads efficiently. **Apache Spark can be used here.**

**Durability:-** Durability refers to the ability of the system to ensure data persistence and integrity, preventing data loss even in the face of hardware failures or other issues. **Hadoop can be used here.**

**Availability:-** Availability is essential for ensuring that the Big Data system remains operational and accessible, providing consistent services to users despite potential failures or disruptions. **Zookeeper can be used here.**

**Data Collection:** The big data company continuously collects a massive volume of data in real-time from user interactions, tweets, retweets, likes, follows, etc. This data is often unstructured and includes text, images, videos, and more.

**Data Ingestion:** The collected data is ingested into a distributed computing framework like Apache Hadoop or Apache Spark. These frameworks allow for the parallel processing of large datasets across a cluster of machines.

**Data Storage:** The big data company uses distributed storage systems, such as Hadoop Distributed File System (HDFS) or cloud-based storage solutions like Amazon S3, to store the vast amounts of data.

**Data Processing:** Data is processed in batches or streams. Batch processing involves analyzing data in large chunks at scheduled intervals, while stream processing handles data in real-time as it's generated. **Apache Kafka or Flink can be sued here**

**Data Cleaning and Transformation:** The data may undergo cleaning and transformation to remove noise, handle missing values, and prepare it for analysis.

**Analysis and Machine Learning:** Various analytical techniques and machine learning algorithms are applied to extract insights, detect patterns, and make predictions from the data. This could involve sentiment analysis, trend detection, recommendation systems, and more. **SparkML ,Sklearn and other libraries can be used here.**

**Visualization and Reporting:** The results of the analysis are often visualized using tools like Tableau, Power BI, or custom-built visualization dashboards. This helps in presenting the findings in a user-friendly format.

**Data Storage for Retrieval:** Processed data and results may be stored in databases, data warehouses, or other storage solutions for easy retrieval and further analysis.

**Feedback Loop:** The big data company may use the insights gained from big data analysis to improve user experience, optimize content delivery, and refine algorithms for features like recommendation systems or content ranking

**Data Security and Privacy:** Ensure the security and privacy of user data, adhering to data protection regulations.