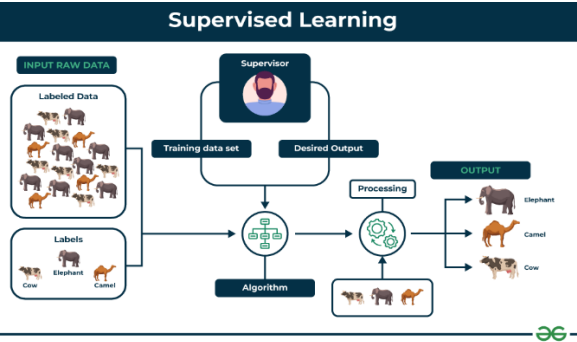
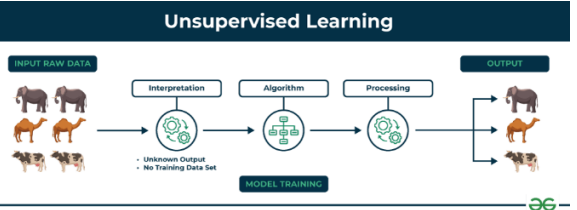
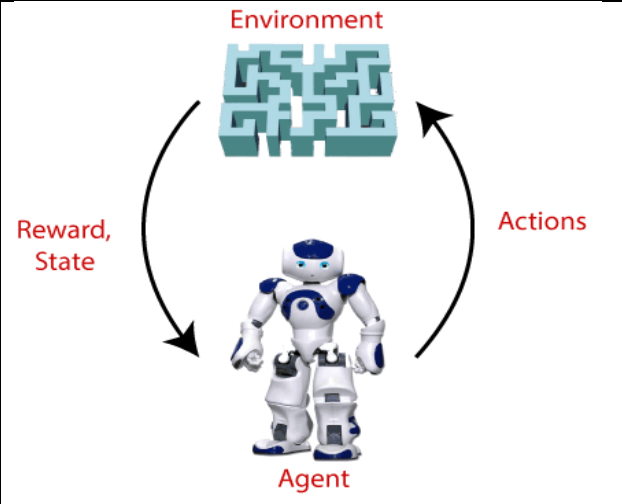
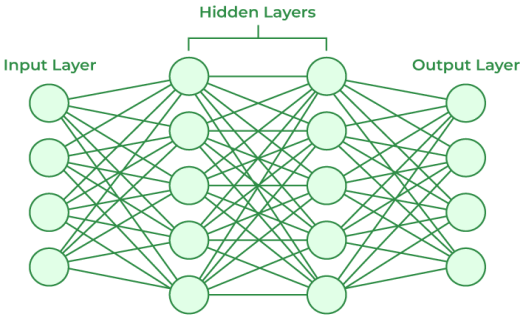


SERIAL NO:	QUESTION:	ANSWERS:
1: Types of Machine Learning (10 Marks)	<p>a) Explain the main differences between Supervised, Unsupervised, and Reinforcement Learning. Provide an example for each type. (6 Marks)</p> <p>b) Discuss one advantage and one limitation of each type of Machine Learning. (4 Marks)</p>	<p>Supervised Learning:</p>  <p>The diagram illustrates the supervised learning workflow. It starts with 'INPUT RAW DATA' (elephants and camels) which is split into 'Labeled Data' (with icons and text labels like 'Elephant', 'Camel', 'Cow') and 'Training data set'. The 'Training data set' is fed into an 'Algorithm' (represented by a brain icon). A 'Supervisor' (person icon) provides a 'Desired Output' to the 'Algorithm'. The 'Algorithm' then goes through 'Processing' to produce the 'OUTPUT' (elephants, camels, and cows). A feedback loop connects the 'OUTPUT' back to the 'Supervisor'.</p> <ul style="list-style-type: none"> Definition: In supervised learning, the model is trained on a labeled dataset, which means that each training example is paired with an output label. The goal is for the model to learn to map inputs to the correct output. Example: Spam email detection, where emails are labeled as "spam" or "not spam." Advantage: High accuracy with labeled data, as the model learns directly from the examples provided. Limitation: Requires a large amount of labeled data, which can be expensive and time-consuming to obtain. <p>Unsupervised Learning:</p>  <p>The diagram shows the unsupervised learning workflow. 'INPUT RAW DATA' (elephants and camels) is processed through 'Interpretation' (with a brain icon) and an 'Algorithm' (brain icon) to produce the 'OUTPUT' (elephants and camels). A 'MODEL TRAINING' box is shown below the algorithm. A note indicates 'Unknown Output' and 'No Training Data Set'.</p> <ul style="list-style-type: none"> Definition: Unsupervised learning involves training a model on data without labeled responses. The model tries to find hidden patterns or intrinsic structures in the input data. Example: Customer segmentation, where customers are grouped into clusters based on purchasing behavior without predefined labels. Advantage: Can find hidden patterns in data, useful for exploratory data analysis. Limitation: Interpretation of results can be challenging, as there is no straightforward way to validate the output. <p>Reinforcement Learning:</p>

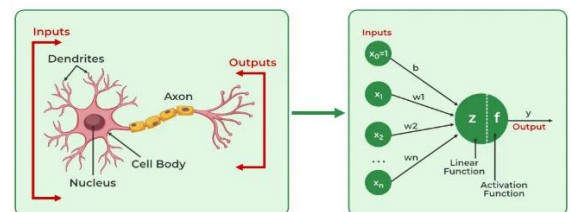
		<div data-bbox="879 192 1503 694">  </div> <ul style="list-style-type: none"> • <i>Definition:</i> Reinforcement learning involves an agent that learns to make decisions by performing actions and receiving feedback from the environment in the form of rewards or penalties. • <i>Example:</i> Game-playing AI like AlphaGo, which learns to play and improve over time through trial and error. • <i>Advantage:</i> Learns optimal actions through trial and error, useful in dynamic environments. • <i>Limitation:</i> Can be computationally intensive and slow to learn, as it requires many iterations to achieve good performance.
<p>2: Old vs. New Machine Learning (4 Marks)</p>	<p>Describe two significant differences between traditional Machine Learning methods and modern Machine Learning approaches like Deep Learning. Provide an example of each to illustrate your points.</p>	<ol style="list-style-type: none"> 1. Feature Engineering: <ul style="list-style-type: none"> ○ Traditional ML: Requires manual feature engineering, where domain experts handcraft features from raw data. This process is often labor-intensive and requires deep domain knowledge. ○ Deep Learning: Automatically extracts features from raw data through multiple layers of abstraction. This allows the model to learn complex representations directly from the data. ○ <i>Example:</i> In image classification, traditional ML might require manual extraction of features like edges or textures, whereas deep learning uses convolutional layers to automatically learn these features from the pixels. 2. Scalability and Complexity: <ul style="list-style-type: none"> ○ Traditional ML: Models like linear regression or decision trees are less complex and often struggle to scale with large datasets and complex patterns.

		<p>They may not capture intricate relationships within the data.</p> <ul style="list-style-type: none"> ○ Deep Learning: Handles large datasets and complex patterns efficiently due to its layered architecture and the ability to model non-linear relationships. It scales well with increased data and computational power. ○ <i>Example:</i> Logistic regression might work for simple classification tasks but would struggle with tasks like image recognition, where Convolutional Neural Networks (CNNs) excel due to their ability to capture spatial hierarchies.
3: Artificial Neural Networks and Activation Functions (10 Marks)	<p>a) What are artificial neural networks (ANNs)? Describe their basic structure and components. (4 Marks)</p>	<p>a)</p> <ul style="list-style-type: none"> ● Artificial Neural Networks (ANNs): Computational models inspired by the human brain, designed to recognize patterns. ANNs consist of layers of interconnected nodes or neurons that process input data and produce an output.  <ul style="list-style-type: none"> ● Basic Structure: <ul style="list-style-type: none"> ○ Input Layer: Receives the initial data. Each neuron in this layer represents a feature of the input data. ○ Hidden Layers: Intermediate layers that process inputs from the input layer through weighted connections and produce outputs for the next layer. These layers extract features and learn complex representations. ○ Output Layer: Produces the final output, such as a classification label or a regression value. ● Components: <ul style="list-style-type: none"> ○ Neurons (Nodes): Basic units that receive inputs, apply a linear transformation

b) Explain the purpose of activation functions in neural networks. Provide three examples of activation functions and their mathematical formulations. (6 Marks)

b)

- **Purpose of Activation Functions:** Activation functions introduce non-linearity into the network, allowing it to learn and model complex patterns in the data. Without activation functions, a neural network would simply be a linear regression model, regardless of the number of layers.



- **Examples:**

1. **Sigmoid:**

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

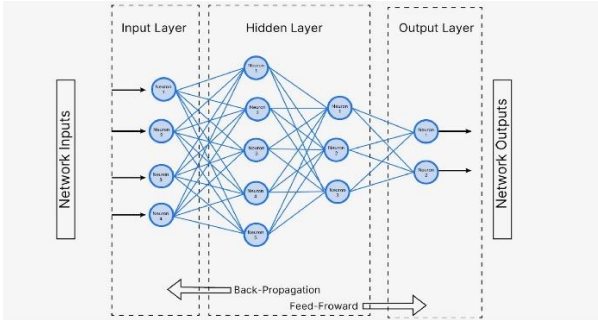
- **Description:** Maps input values to a range between 0 and 1, useful for binary classification tasks.
- **Property:** Smooth gradient, but can suffer from vanishing gradients, making training deep networks difficult.

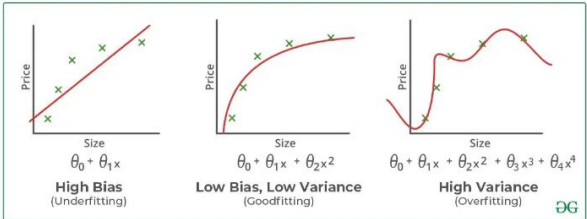
2. **ReLU (Rectified Linear Unit):**

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

- **Description:** Outputs zero for negative inputs and passes positive inputs unchanged. It introduces sparsity in the model, where only a subset of neurons are active, leading to efficient computation.
- **Property:** Fast convergence, but can suffer from dying ReLUs

		<p>where neurons get stuck during training.</p> <p>3. Tanh (Hyperbolic Tangent):</p> <div data-bbox="1177 331 1485 392" data-label="Equation-Block"> $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ </div> <ul style="list-style-type: none"> ▪ <i>Description:</i> Maps input values to a range between -1 and 1, providing better gradient flow than the sigmoid function. ▪ <i>Property:</i> Zero-centered output, which helps in faster convergence, but still can suffer from vanishing gradients for deep networks.
4: The XOR Problem (4 Marks)	<p>a) Describe the XOR problem and explain why it is significant in the context of neural networks. (2 Marks)</p> <p>b) How do neural networks solve the XOR problem? (2 Marks)</p>	<p>a)</p> <ul style="list-style-type: none"> • XOR Problem: The XOR (exclusive OR) problem is a classic problem in machine learning where the output is true if and only if one of the inputs is true. It is a binary classification problem with the following truth table: <ul style="list-style-type: none"> ○ (0, 0) -> 0 ○ (0, 1) -> 1 ○ (1, 0) -> 1 ○ (1, 1) -> 0 <div data-bbox="890 1196 1485 1529" data-label="Figure"> <p style="text-align: center;">Linear separability</p> <p>The figure consists of three 2D plots labeled 'x and y' on the axes. Each plot shows four data points: (0,0), (0,1), (1,0), and (1,1). In the first plot, a diagonal line with a negative slope separates the points (0,1) and (1,0) from (0,0) and (1,1). In the second plot, a different diagonal line is shown. In the third plot, a question mark is placed in the center, indicating that the data is not linearly separable.</p> </div> <ul style="list-style-type: none"> • Significance: The XOR problem is significant because it is not linearly separable, meaning it cannot be solved by a single-layer perceptron or linear classifier. This limitation highlighted the need for multi-layer networks to solve more complex problems. <p>b)</p> <ul style="list-style-type: none"> • Neural Networks Solution: Neural networks solve the XOR problem by using multiple layers (hidden layers) that introduce non-linear transformations. By combining the outputs of these hidden layers, the network can create a decision boundary that separates the XOR classes. Specifically, a network with at least one hidden layer and non-linear

		<p>activation functions can model the XOR function by learning complex representations of the input data.</p> 
<p>5: Training Neural Networks and Backpropagation (10 Marks)</p>	<p>a) Explain the process of training a neural network. What are the key steps involved? (5 Marks)</p> <p>b) Describe the backpropagation algorithm and its significance in training neural networks. (5 Marks)</p>	<p>a) (Diagram same as Q4.b)</p> <ul style="list-style-type: none"> • Initialize Weights and Biases: The parameters (weights and biases) of the network are initialized, often with small random values. • Forward Pass: The input data is passed through the network layer by layer, and each layer's neurons apply their weights, biases, and activation functions to compute their outputs. The final output is compared to the actual target to compute the loss. • Compute Loss: The loss function measures the difference between the predicted output and the actual target. Common loss functions include Mean Squared Error (MSE) for regression and Cross-Entropy Loss for classification. • Backward Pass (Backpropagation): The gradients of the loss with respect to each weight and bias are computed using the chain rule of calculus. These gradients indicate how to adjust the parameters to reduce the loss. • Update Weights and Biases: The weights and biases are updated using an optimization algorithm, such as Stochastic Gradient Descent (SGD) or Adam. The parameters are adjusted in the direction that reduces the loss. <p>b)</p> <ul style="list-style-type: none"> • Backpropagation Algorithm: <ul style="list-style-type: none"> ○ Forward Pass: Calculate the output of the network for the given input and compute the loss. ○ Backward Pass: Compute the gradient of the loss function with respect to each parameter using the chain rule. This involves calculating the derivative of the loss with respect to the output, and then propagating this derivative backward through the network to update the parameters.

		<ul style="list-style-type: none"> ○ Gradient Calculation: For each layer, the gradient of the loss with respect to its weights and biases is calculated by multiplying the gradient from the subsequent layer by the derivative of the activation function of the current layer. ○ Parameter Update: Update the weights and biases using the computed gradients and the learning rate, which controls the step size of the update. ● Significance: Backpropagation is essential because it efficiently computes the gradients needed for the optimization process in training neural networks. It allows for the training of deep networks by propagating the error gradient backward through the network, layer by layer. This process enables the model to learn from the errors and adjust its parameters to minimize the loss, ultimately improving its performance on the training data and its generalization to unseen data.
6: Underfitting vs. Overfitting (6 Marks)	<p>a) Define underfitting and overfitting in the context of machine learning models. (2 Marks)</p> <p>b) Provide two strategies to prevent each issue. (4 Marks)</p>	<p>a)</p> <ul style="list-style-type: none"> ● Underfitting: Occurs when a model is too simple to capture the underlying structure of the data. This results in poor performance on both training and test datasets. The model fails to learn the patterns in the data, leading to high bias. ● Overfitting: Occurs when a model is too complex and captures not only the underlying patterns but also the noise in the training data. This results in excellent performance on the training dataset but poor generalization to new, unseen data, leading to high variance.  <p>b)</p> <ul style="list-style-type: none"> ● Preventing Underfitting: <ol style="list-style-type: none"> 1. Increase Model Complexity: Use a more complex model with additional layers or parameters that can capture the data's underlying patterns. <ul style="list-style-type: none"> ▪ <i>Example:</i> Switch from a linear regression model to a

		<p>polynomial regression model or increase the depth of a neural network.</p> <ol style="list-style-type: none"> 2. Feature Engineering: Add relevant features or transformations of existing features that better represent the underlying patterns in the data. <ul style="list-style-type: none"> ▪ <i>Example:</i> Create interaction terms or polynomial features from the existing features to provide the model with more information. <ul style="list-style-type: none"> • Preventing Overfitting: <ol style="list-style-type: none"> 1. Regularization: Apply techniques such as L2 regularization (Ridge), L1 regularization (Lasso), or Dropout in neural networks to penalize large coefficients and prevent the model from fitting the noise. <ul style="list-style-type: none"> ▪ <i>Example:</i> Adding an L2 penalty term to the loss function in a linear regression model or using dropout layers in a neural network. 2. Cross-Validation: Use techniques like k-fold cross-validation to ensure the model generalizes well to unseen data by evaluating its performance on different subsets of the data. <ul style="list-style-type: none"> ▪ <i>Example:</i> Split the dataset into k folds and train the model on k-1 folds while validating it on the remaining fold, rotating through all folds.
7: Feature Scaling and Fully Connected Layers (6 Marks)	a) Why is feature scaling important in training neural networks? Describe two common methods of feature scaling. (3 Marks)	a) <ul style="list-style-type: none"> • Importance of Feature Scaling: Feature scaling is crucial in training neural networks because it ensures that all features contribute equally to the learning process. It speeds up convergence during training by making the gradient descent steps more uniform and preventing features with larger scales from dominating the model's learning process. Additionally, it helps avoid issues with exploding and vanishing gradients. • Methods: <ol style="list-style-type: none"> 1. Standardization: Transforms data to have a mean of zero and a standard deviation of one.

		<ul style="list-style-type: none"> ▪ Formula: $X' = \frac{X - \mu}{\sigma}$ ▪ Use Case: Useful when the features have different units or scales, ensuring they are comparable. <p>2. Normalization: Scales data to a fixed range, typically [0, 1] or [-1, 1].</p> <ul style="list-style-type: none"> ▪ Formula: $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$ ▪ Use Case: Useful when features have different ranges and the model, such as a neural network, benefits from having input values in a specific range. <p>b)</p> <ul style="list-style-type: none"> • Fully Connected Layers (Dense Layers): In fully connected layers, each neuron is connected to every neuron in the previous and next layers. These layers are responsible for learning global patterns in the data and are typically used in the final stages of a network to combine features learned in previous layers and produce the final output. <ul style="list-style-type: none"> ○ Description: Each neuron applies a linear transformation to the input followed by a non-linear activation function. ○ Use Case: Suitable for tasks where features are not spatially correlated, such as in tabular data or after feature extraction in convolutional networks. • Convolutional Layers: Convolutional layers apply filters (kernels) to local regions of the input, extracting spatial features and preserving the spatial structure of the data. <ul style="list-style-type: none"> ○ Description: Each filter convolves over the input, producing feature maps that highlight local patterns like edges, textures, or objects. ○ Use Case: Ideal for tasks involving spatial data, such as image and video processing, where local patterns are important. <p>Difference:</p> <ul style="list-style-type: none"> • Connection Pattern: Fully connected layers have dense connections, making them computationally intensive, while convolutional layers have sparse, local connections, reducing the number of parameters.
	<p>b) What are fully connected layers in neural networks? How do they differ from convolutional layers? (3 Marks)</p>	

		<ul style="list-style-type: none">• Feature Learning: Fully connected layers learn global patterns, whereas convolutional layers focus on local spatial hierarchies and features.• Use Case: Fully connected layers are used for final decision-making, while convolutional layers are used for feature extraction in spatial data.
--	--	--