Deep Learning

Artificial Intelligence

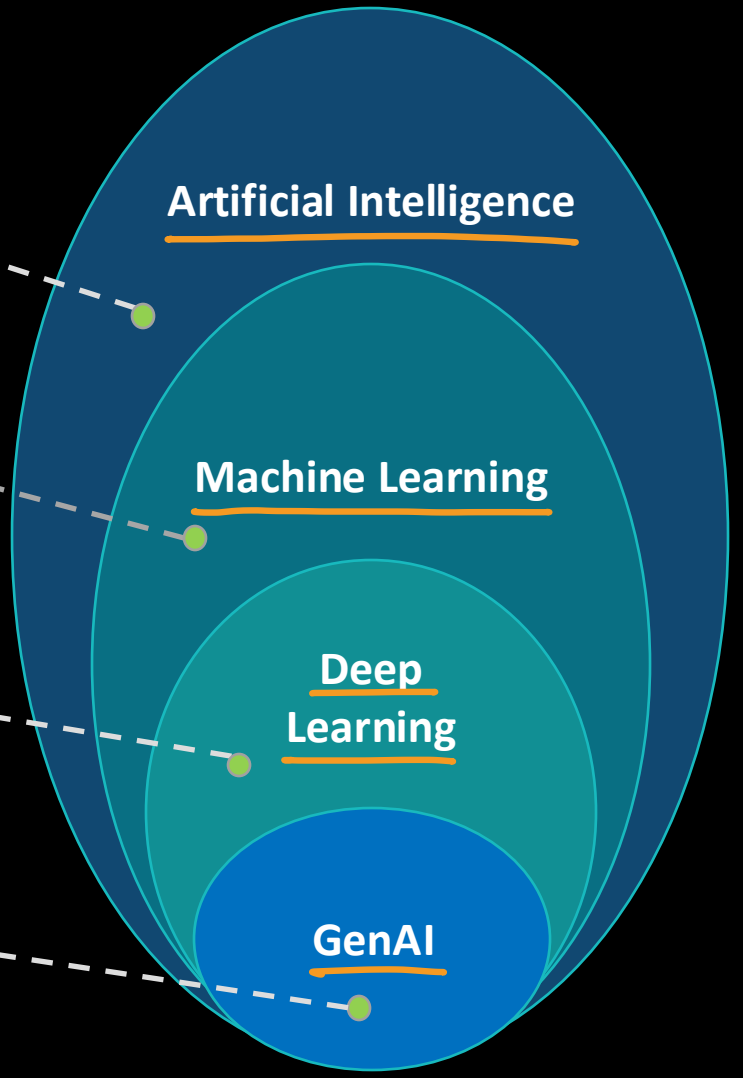
thinking
vision
Speech

Artificial Intelligence
- A technique which enables machine to mimic human behavior

Machine Learning
- Subset of AI which uses statistical methods to enable machines to improve the experience

Deep Learning
- Subset of ML which makes the computation of multi-layer neural network feasible

Generative AI
- Subset of AI that can created new content like text, images and music



Artificial Intelligence

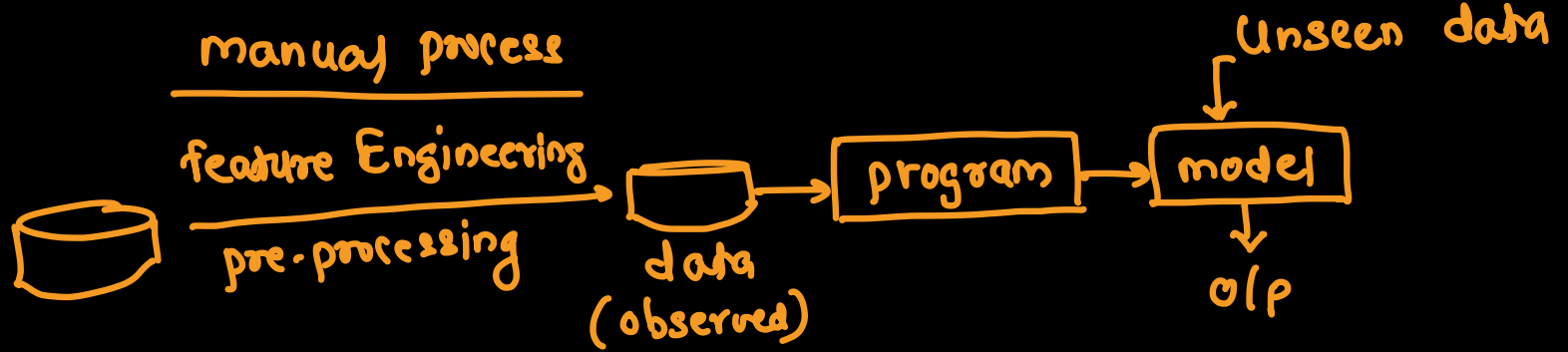
- **Definition**
  - Artificial Intelligence (AI) is the broad concept of enabling machines to perform tasks that typically require human intelligence
  - Artificial Intelligence (AI) is a broad field of computer science focused on creating systems that can perform tasks that typically require human intelligence
- **Examples**
  - Self-driving cars, virtual assistants (Siri, Alexa), spam filters, recommendation systems.
- **Different Approaches**
  - AI encompasses various techniques including rule-based systems, machine learning (ML), and deep learning.

Explicit programming

Machine Learning

data → labelled → supervised
data → unlabelled → unsupervised

- **Definition**
  - Machine Learning (ML) is a branch of Artificial Intelligence (AI) that focuses on enabling computers to learn from data
  - Instead of being explicitly programmed with instructions for every possible scenario, ML algorithms analyze large datasets to identify patterns and make predictions or decisions
- **Types**
  - **Supervised Learning**: Training with labeled data (e.g., classifying images as cats or dogs).
  - **Unsupervised Learning**: Discovering patterns in unlabeled data (e.g., customer segmentation).
  - **Reinforcement Learning**: Training agents to make decisions in an environment to maximize a reward.



Deep Learning

data < huge / unstructured → text / images / audio / video

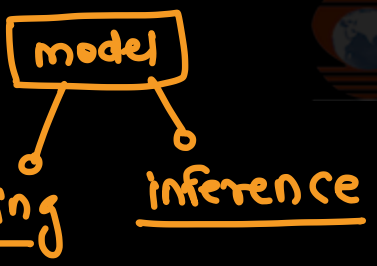
- **Definition**
  - Deep Learning (DL) is a subfield of Machine Learning that utilizes artificial neural networks with multiple layers to analyze data and extract features automatically → Feature Engineering
  - It's inspired by the structure and function of the human brain, specifically how neurons are interconnected and process information
    - BNN → Biological Neural Network
    - ANN → Artificial Neural Network
- **Examples**
  - Computer Vision: Image recognition, Object Detection, Image Generation, Facial Recognition
  - NLP: Chatbot, Text Summarization, Sentiment Analysis, Spam Detection
  - Speech Recognition: Virtual Assistants, Transcription, Voice search, Music Generation
  - Healthcare & Medicine: Drug Discovery, Personalized Medicines, Disease Prediction
  - Finance & Banking: Fraud Detection, Algorithmic Trading, Credit Risk Assessment
  - Gaming & Entertainment: AI powered characters, Procedural Content Generation
  - Robotics & Autonomous Systems: Robot Navigation, Object Manipulation

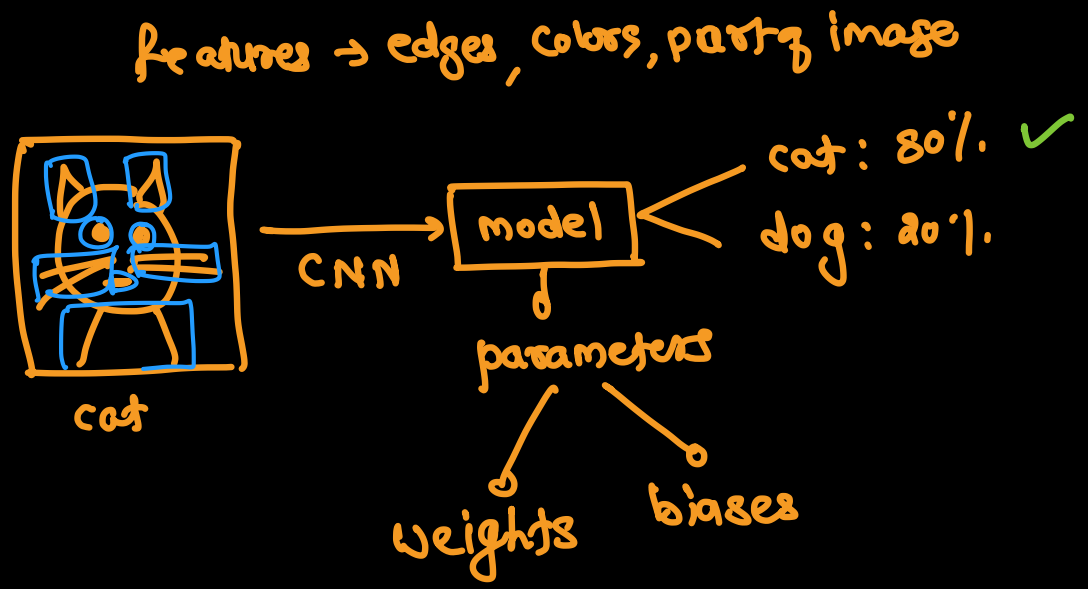
Advantages of Deep Learning

- **High Accuracy**: Achieves state-of-the-art results in many tasks
- **Automatic Feature Learning**: Eliminates the need for manual feature engineering
- **End-to-End Learning**: Simplifies development by training the entire system as a single unit → ANN
- **Scalability**: Improves with more data
- **Handles Unstructured Data**: Effectively processes images, text, and audio
- **Transfer Learning**: Leverages knowledge from one task to another
- **Versatility**: Applicable across a wide range of industries and tasks

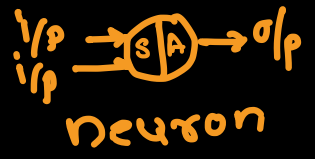
Disadvantages of Deep Learning

- Massive Data Dependency → huge data
- Data Bias
- High Computational Cost
- Long Training Times
- Explainability & Interpretability ↗ black box
- Lack of Transparency
- Difficulty in Generalization
- Complex Architecture Design
- Requires Specialized Knowledge
- High Deployment Costs





Components of Deep Learning



- **Artificial Neural Networks (ANNs) - The Foundation**
  - **Neurons/Nodes**: These are the fundamental building blocks, inspired by biological neurons. They receive inputs, perform a calculation (weighted sum and activation function), and produce an output.
  - **Weights**: Numerical values assigned to each connection between neurons. Weights determine the strength of the connection and influence the output. The learning process involves adjusting these weights to minimize errors.
  - **Biases**: Added to the weighted sum of inputs before applying the activation function. Biases allow neurons to activate even when all inputs are zero, providing more flexibility in learning patterns.
  - **Activation Functions**: Non-linear functions applied to the output of a neuron. They introduce non-linearity into the model, allowing it to learn complex patterns that linear models cannot capture.

↳ unstructured data

- **Layers - Organizing Neurons** layer = collection of neurons
  - **Input Layer:** Receives the raw data as input.
  - **Hidden Layers:** Intermediate layers between the input and output layers. Deep learning models have *multiple* hidden layers (hence "deep" learning). These layers learn increasingly abstract representations of the data.
  - **Output Layer:** Produces the final prediction or output of the model.

Components of Deep Learning

- **Architectures - Specific Network Structures (Different Types of Deep Learning)**
  - **Convolutional Neural Networks (CNNs):** **Used for** Image recognition, object detection, video analysis.
  - **Recurrent Neural Networks (RNNs):** **Used for** Natural language processing (text generation, machine translation), speech recognition.
  - **Transformers:** **Used for** Natural language processing (machine translation, text summarization), image recognition.
- **Training Process - How Models Learn**
  - Loss Function (Cost Function): Measures the difference between the model's predictions and the actual values. The goal is to minimize this loss.
    - Mean Squared Error (MSE): For regression tasks.
    - Cross-Entropy: For classification tasks.
  - Optimizer: An algorithm that adjusts the weights & biases of the network to minimize the loss function
    - Stochastic Gradient Descent (SGD): Basic optimization algorithm.
    - Adam: Adaptive learning rate optimizer, often performs well in practice.
  - Backpropagation: An algorithm that calculates the gradients of the loss function with respect to each weight and bias in the network. These gradients are used by the optimizer to update the weights and biases.
  - Epochs: One complete pass through the entire training dataset.
  - Batch Size: The number of samples used in one iteration of the training process.

Components of Deep Learning

- Regularization Techniques (Preventing Overfitting)
  - Dropout: Randomly deactivates neurons during training, forcing the network to learn more robust features
  - L1/L2 Regularization: Adds a penalty term to the loss function that discourages large weights
  - Early Stopping: Monitors performance on a validation set and stops training when performance starts to degrade

Core Components of Deep Learning

- Neurons/Nodes: Basic units of computation
- Weights & Biases: Parameters learned during training
- Activation Functions: Introduce non-linearity
- Layers (Input, Hidden, Output): Organize neurons into a hierarchical structure
- Architectures (CNN, RNN, Transformers): Specific network structures for different tasks
- Loss Function: Measures prediction error
- Optimizer: Adjusts parameters to minimize loss
- Backpropagation: Calculates gradients for parameter updates
- Regularization Techniques (Dropout, L1/L2): Prevent overfitting

Libraries/Frameworks

- **TensorFlow (Google)** → production
  - One of the most widely used frameworks, known for its flexibility and scalability. Excellent for production deployment
  - **Key Features**: Keras (high-level API integrated within TensorFlow), distributed training, graph execution
  - **Language**: Python (primarily), C++, Java, Go
- **PyTorch (Facebook)** → Researchers
  - Gaining immense popularity, known for its dynamic computation graph and ease of use, Favored by researchers
  - **Key Features**: Dynamic computation graph (easier debugging), Pythonic API, strong community support
  - **Language**: Python (primarily), C++
- **Keras**
  - A high-level API that can run on top of TensorFlow, Theano, or CNTK (though TensorFlow is now the primary backend), focuses on rapid prototyping and ease of use
  - **Key Features**: Simple API, modular design, supports various neural network architectures
  - **Language**: Python

Machine Learning Vs Deep Learning

Feature	Machine Learning (ML)	Deep Learning (DL)
Definition	Algorithms that learn from data without explicit programming.	A subset of ML using deep neural networks to automatically extract features.
Feature Engineering	Manual; requires domain expertise.	Automatic; learned by the network.
Data Requirements	Can work with smaller datasets.	Requires massive amounts of data.
Computational Resources	Less computationally intensive.	Highly computationally intensive (GPUs/TPUs).
Interpretability	Generally more interpretable.	Often considered "black boxes."
Algorithms	Linear Regression, Logistic Regression, Decision Trees, SVMs, KNN, Naive Bayes.	CNNs, RNNs, Transformers.
Complexity	Less complex models. → formula	Highly complex models with many layers and parameters. → weights & biases

When to use Which?

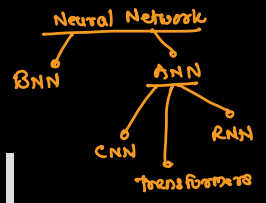
- Use Machine Learning (ML) when
  - You have a smaller dataset
  - You need to understand why the model is making certain predictions (interpretability)
  - You have domain expertise to engineer relevant features
  - Computational resources are limited
- Use Deep Learning (DL) when
  - You have a very large dataset
  - Automatic feature extraction is desired
  - You're dealing with complex data like images, audio, or text
  - High accuracy is paramount and you're willing to sacrifice some interpretability

Neural Network

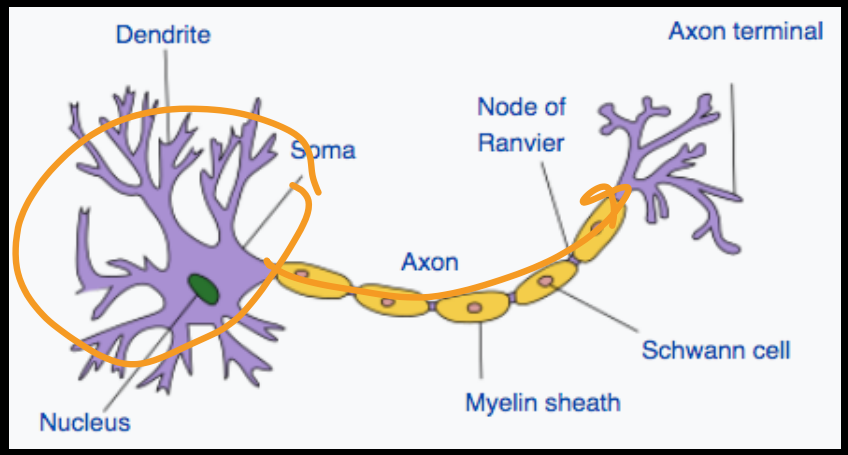
Introduction

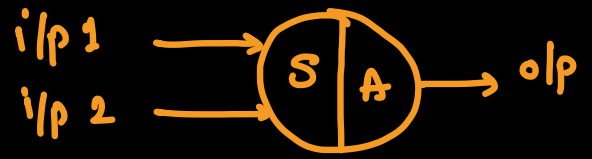
- A neural network is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain
- It is a type of machine learning (ML) process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain
- It creates an adaptive system that computers use to learn from their mistakes and improve continuously
- Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy
- Neural networks can help computers make intelligent decisions with limited human assistance
- This is because they can learn and model the relationships between input and output data that are nonlinear and complex

Structure of Neural Network



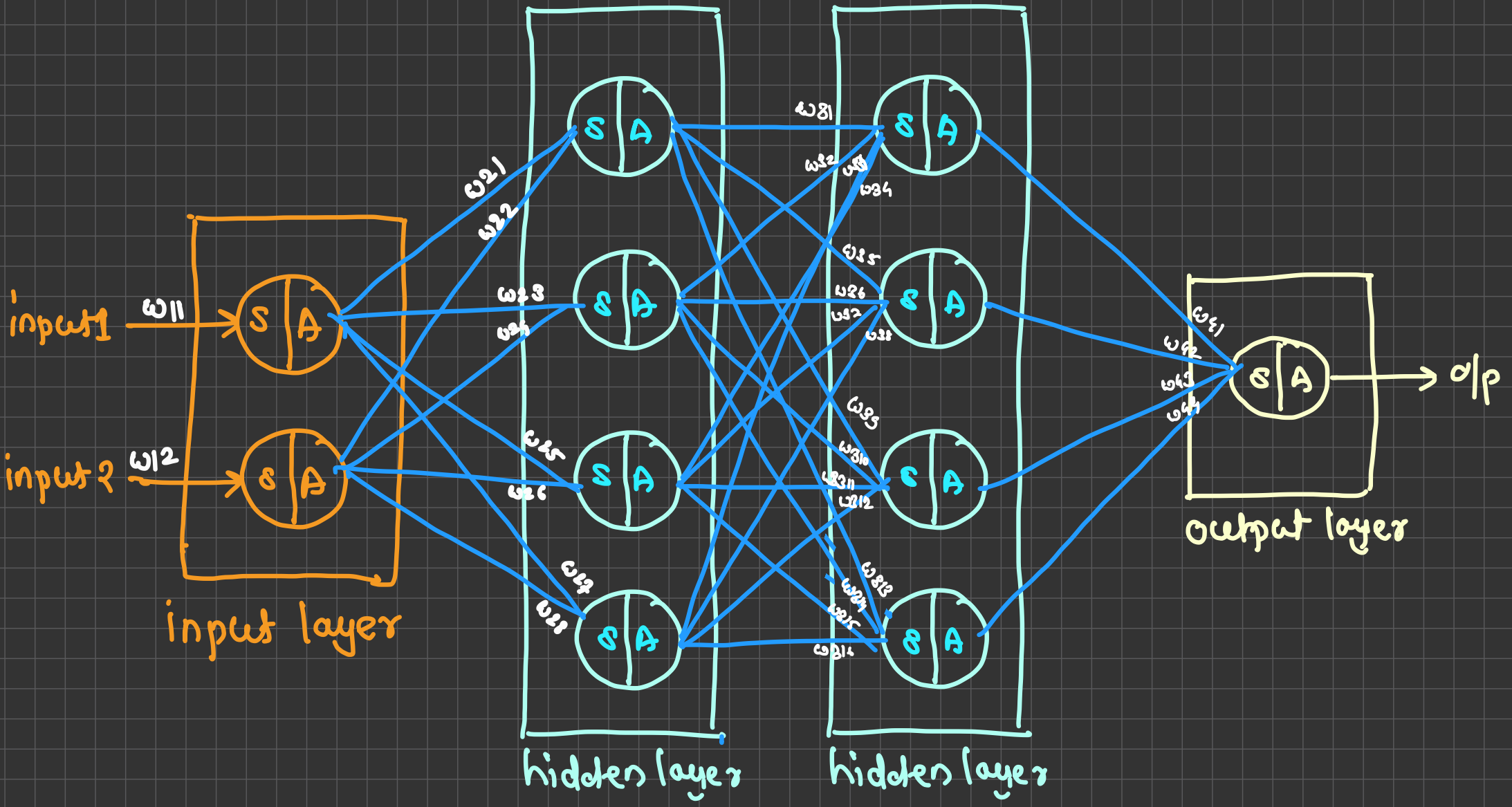
- The human brain is the inspiration behind neural network architecture
- Human brain cells, called neurons, form a complex, highly interconnected network and send electrical signals to each other to help humans process information
- Similarly, an artificial neural network is made of artificial neurons that work together to solve a problem
- Artificial neurons are software modules, called nodes, and artificial neural networks are software programs or algorithms that, at their core, use computing systems to solve mathematical calculations





artificial Neuron

Artificial Neural Network → fully connected N/W



Architecture of Simple Neural Network

- A basic neural network has interconnected artificial neurons in three layers
- **Input Layer**
  - Information from the outside world enters the artificial neural network from the input layer
  - Input nodes process the data, analyze or categorize it, and pass it on to the next layer
- **Hidden Layer(s)** More #hidden layers = better results but more resources are required
  - Hidden layers take their input from the input layer or other hidden layers
  - Artificial neural networks can have a large number of hidden layers
  - Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer → optional layer → a NN may have 0 or more hidden layers
- **Output Layer**
  - The output layer gives the final result of all the data processing by the artificial neural network
  - It can have single or multiple nodes (neurons)
  - For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

Architecture of Deep Neural Network

- Deep neural networks, or deep learning networks, have several hidden layers with millions of artificial neurons linked together
- A number, called weight, represents the connections between one node and another
- The weight is a positive number if one node excites another, or negative if one node suppresses the other
- Nodes with higher weight values have more influence on the other nodes
- Theoretically, deep neural networks can map any input type to any output type
- However, they also need much more training as compared to other machine learning methods
- They need millions of examples of training data rather than perhaps the hundreds or thousands that a simpler network might need

Perceptron

- A single-layer feedforward neural network was introduced in the late 1950s
- It was the starting phase of Deep Learning and Artificial neural networks
- Perceptron is one of the simplest Artificial neural network architectures
- It consists of a single layer of input nodes that are fully connected to a layer of output nodes
- Types of Perceptron
  - Single-Layer Perceptron
    - This type of perceptron is limited to learning linearly separable patterns
    - They are effective for tasks where the data can be divided into distinct categories through a straight line
  - Multilayer Perceptron
    - Multilayer perceptrons possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data

Components of Perceptrons

- **Input Features:** The perceptron takes multiple input features, each input feature represents a characteristic or attribute of the input data
- **Weights:** Each input feature is associated with a weight, determining the significance of each input feature in influencing the perceptron's output. During training, these weights are adjusted to learn the optimal values.
- **Summation Function:** The perceptron calculates the weighted sum of its inputs using the summation function. The summation function combines the inputs with their respective weights to produce a weighted sum.
- **Activation Function:** The weighted sum is then passed through an activation function. Perceptron uses Heaviside step function functions. which take the summed values as input and compare with the threshold and provide the output as 0 or 1.
- **Output:** The final output of the perceptron, is determined by the activation function's result. For example, in binary classification problems, the output might represent a predicted class (0 or 1).

Components of Perceptrons

- **Bias:** The bias is an additional parameter that allows the perceptron to shift its activation function. It helps the model learn patterns even when all inputs are zero.
- **Learning Algorithm (Weight Update Rule)**: During training, the perceptron learns by adjusting its weights and bias based on a learning algorithm. A common approach is the perceptron learning algorithm, which updates weights based on the difference between the predicted output and the true output.

Forward Propagation

- **Input Layer:** Each feature in the input layer is represented by a node on the network, which receives input data
- **Weights and Connections:** The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed
- **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.
- **Output:** The final result is produced by repeating the process until the output layer is reached

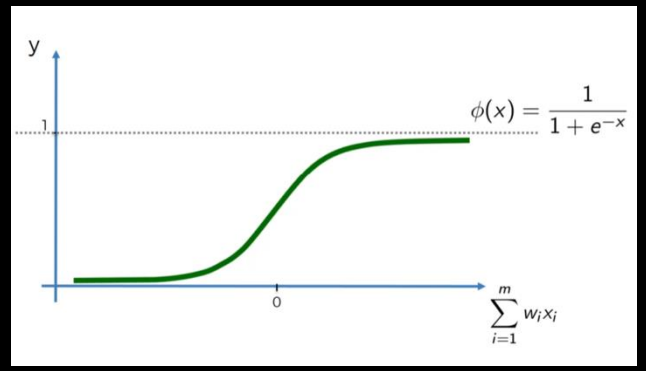
Activation Function

- An activation function in an artificial neural network (ANN) is a mathematical function applied to the output of each neuron (or node) to introduce non-linearity into the model
- This non-linearity allows the network to learn complex patterns and relationships in the data, enabling it to perform tasks such as classification and regression
- Merits
  - Introduce Non-Linearity
    - Without activation functions, a neural network would essentially behave like a linear model, regardless of how many layers it has. Activation functions allow the model to learn non-linear mappings
  - Control Output
    - They determine the output of neurons, which can be used in further computations or as final predictions
  - Enable Learning
    - Activation functions help in the optimization process by providing gradients during backpropagation, which are essential for adjusting weights

Sigmoid Function

- The sigmoid activation function is a mathematical function that maps any real-valued number into a value between 0 and 1. It's commonly used in binary classification tasks.

- **sigmoid**$(z) = \frac{1}{1+ e^{z}}$

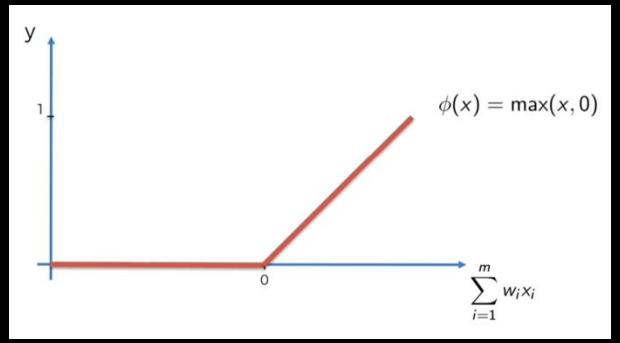


- Properties
  - Output Range: The output of the sigmoid function is always between 0 and 1, which makes it particularly useful for models that predict probabilities.
  - S-shaped Curve: The graph of the sigmoid function has an "S" shape, which makes it smooth and differentiable
  - Non-linear: The sigmoid function introduces non-linearity, allowing the neural network to learn complex patterns
- Use Cases
  - Binary Classification: Often used in the output layer of binary classifiers.
  - Logistic Regression: Commonly applied in logistic regression models.

Relu Function

- The ReLU (Rectified Linear Unit) activation function is one of the most widely used activation function. It introduces non-linearity into the model while maintaining computational efficiency.

- ReLU (x) = max(0, x)

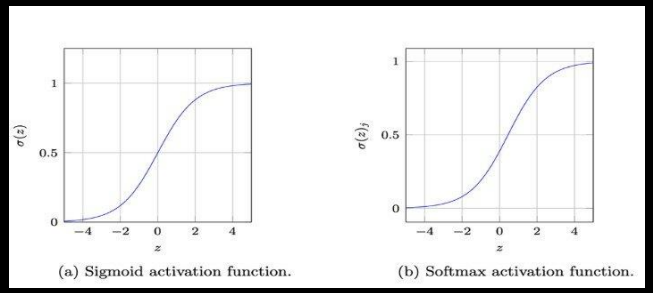


- Properties
  - Non-linearity: ReLU allows the model to learn complex patterns by introducing non-linearity
  - Output Range: The output ranges from 0 to infinity. It outputs zero for any negative input, which can help in preventing the vanishing gradient problem.
  - Sparsity: Because it outputs zero for half of the input space, ReLU can lead to sparse representations, which can improve model efficiency.
  - Computational Efficiency: ReLU is computationally efficient, as it requires only a simple thresholding at zero
- Use Cases
  - ReLU is commonly used in hidden layers of deep neural networks due to its advantages in training and efficiency, making it a standard choice in modern architectures.

Softmax Function

- The Softmax function takes a vector of real numbers and converts it into a probability distribution
- Each value is transformed into a number between 0 and 1, and the sum of all output values equals 1
- Particularly it is used in multi-class classification

- $softmax(z) = \frac{e^z}{\sum e^z}$

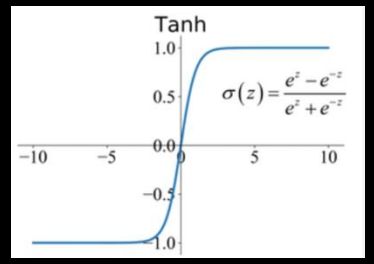


(a) Sigmoid activation function. (b) Softmax activation function.

- Properties
  - Output Range: The output values are in the range (0, 1), which makes them interpretable as probabilities.
  - Sum to One: The sum of all output probabilities is 1, satisfying the property of a probability distribution.
  - Sensitivity to Differences: Softmax emphasizes the largest logits, making it more likely for the model to predict the class with the highest score
- Use Cases
  - Multi-Class Classification: It is commonly used in the output layer of neural networks when there are multiple classes
  - Reinforcement Learning: Often used in policy networks to select actions based on their probabilities

Tanh Function

- The tanh (hyperbolic tangent) activation function is another widely used activation function. It maps input values to a range between -1 and 1, making it useful for models that need to output zero-centered values.
- $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$



- Properties
  - Output Range: The output values range from -1 to 1, which means it is zero-centered. This helps in faster convergence during training, as the outputs are balanced around zero.
  - S-shaped Curve: The tanh function has an "S" shaped curve similar to the sigmoid function, but it is steeper, which can lead to better gradients for training.
  - Non-linearity: Like other activation functions, tanh introduces non-linearity, enabling the network to learn complex patterns.
- Use Cases
  - Hidden Layers: Tanh is often used in the hidden layers of neural networks, especially in recurrent neural networks (RNNs) and other architectures where zero-centered outputs are beneficial.
  - Situations Needing Non-linear Transformations: It's suitable for problems requiring a non-linear transformation with outputs centered around zero.

Backpropagation

- **Loss Calculation:** The network's output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function.
- **Loss Function:** MSE, MAE etc.
- **Gradient Descent:** Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.
- **Adjusting weights:** The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.
- **Training:** During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.
- **Activation Functions:** Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to "fire" a neuron is based on the whole weighted input.

Gradient Descent

- Gradient descent is a widely used optimization algorithm in machine learning and deep learning for minimizing the loss function of a model. The primary goal of gradient descent is to find the optimal parameters (weights and biases) that minimize the difference between the predicted outputs and the actual outputs.
- How Gradient Descent Works
  - **Initialization:** Start with random values for the model parameters
  - **Compute the Loss:** Calculate the loss (or cost) using a loss function, which measures how well the model's predictions match the actual data.
  - **Calculate the Gradient:** Compute the gradient of the loss function with respect to each parameter. The gradient is a vector that points in the direction of the steepest increase of the loss function.
  - **Update Parameters:** Adjust the parameters in the opposite direction of the gradient to reduce the loss
    - $\theta = \theta - \alpha \Delta L(\theta)$
  - **Iterate:** Repeat the process (compute loss, calculate gradient, update parameters) until the loss converges to a minimum or until a predefined number of iterations is reached

Types of Neural Network

- Convolutional neural networks (CNN)
  - It can input images, identify the objects in a picture, and differentiate them from one another
  - Their real-world applications include pattern recognition, image recognition, and object detection
  - A CNN's structure consists of three main layers
    - First is the convolutional layer, where most of the computation occurs
    - Second is the pooling layer, where the number of parameters in the input is reduced
    - Lastly, the fully connected layer classifies the features extracted from the previous layers
- Recurrent neural networks(RNN)
  - It can translate language, speech recognition, natural language processing, and image captioning
  - Examples of products using RNNs include smart home technologies and voice command features on mobile phones
  - Feedback loops in the structure of RNNs allow information to be stored similarly to how your memory works
- Radial basis functions networks (RBF)
  - Radial basis function (RBF) networks differ from other neural networks because the input layer performs no computations
  - Instead, it passes the data directly to the hidden layer. As a result, RBFs have a faster learning speed.
  - Applications of RBF networks include time series prediction and function approximation

Types of Neural Network

- **Long short-term memory networks (LSTM)**
  - These networks are unique and can sort data into short-term and long-term memory cells depending on whether or not the data needs to be looped back into the network as data points or entire sequences
  - LSTM can also be used in handwriting recognition and video-to-text conversion

- **Multilayer perceptrons (MLP)**
  - These are a neural network capable of learning the relationship between linear and non-linear data
  - Through backpropagation, MLPs can reduce error rates
  - Applications that benefit from MLPs include face recognition and computer vision

- **Generative adversarial networks (GAN)**
  - They can generate new data sets that share the same statistics as the training set and often pass as actual data
  - An example of this you've likely seen is art created with AI
  - GANs can replicate popular art forms based on patterns in the training set, creating pieces often indistinguishable from human artwork

Types of Neural Network

- **Deep belief networks (DBN)**
  - They are unique because they stack individual networks that can use each other's hidden network layers as the input for the next layer
  - This allows for the neural networks to be trained faster. They are used to generate images and motion-capture data

- **Self-organising maps (SOM)**
  - Self-organising maps (SOMs), or Kohonen maps, can transform extensive complex data sets into understandable two-dimensional maps where geometric relationships can be visualized
  - This can happen because SOMs use competitive learning algorithms in which neurons must compete to be represented in the output
  - This is decided by which neurons best represent the input
  - Practical applications of SOMs include displaying voting trends for analysis and organizing complex data collected by astronomers so it can be interpreted

Advantages of Neural Network

- **Ability to Learn Complex Patterns**
  - Neural networks can model intricate relationships in data, making them effective for tasks like image recognition, natural language processing, and speech recognition.
- **Non-Linearity**
  - With activation functions, neural networks can capture non-linear relationships, allowing them to learn more complex functions than linear models.
- **Feature Extraction**
  - Deep neural networks, especially convolutional neural networks (CNNs), can automatically extract relevant features from raw data, reducing the need for manual feature engineering.
- **Scalability**
  - Neural networks can scale well with large amounts of data. They tend to perform better as the dataset size increases, leveraging more data to improve model performance.
- **Generalization**
  - When properly trained, neural networks can generalize well to unseen data, making them suitable for real-world applications.

Advantages of Neural Network

- **Versatility**
  - Neural networks can be adapted to a wide range of tasks, from classification and regression to generative modeling and reinforcement learning. They can be used in various domains, including finance, healthcare, and robotics.
- **Parallel Processing**
  - Neural networks can take advantage of parallel processing, particularly when implemented on GPUs, significantly speeding up training times.
- **Robustness to Noise**
  - They can be resilient to noisy inputs, which is beneficial in many real-world applications where data is imperfect.
- **Transfer Learning**
  - Pre-trained neural networks can be fine-tuned for specific tasks, enabling efficient training on smaller datasets while leveraging knowledge from larger, related datasets.
- **Continuous Learning**
  - Neural networks can be designed to learn continuously from new data, allowing them to adapt to changing environments over time.

Disadvantages of Neural Network

- **Complexity**
  - Neural networks can be highly complex, making them difficult to design, tune, and interpret. Choosing the right architecture and hyperparameters often requires significant expertise and experimentation.
- **Data Requirements**
  - They typically require large amounts of labeled data to train effectively. Insufficient data can lead to overfitting, where the model learns noise rather than meaningful patterns.
- **Computational Cost**
  - Training neural networks can be resource-intensive, requiring significant computational power and memory, especially for deep architectures. This can lead to long training times.
- **Overfitting**
  - Without proper regularization techniques, neural networks can overfit to the training data, especially when the model is too complex relative to the amount of training data.
- **Lack of Interpretability**
  - Neural networks are often considered "black boxes." Understanding how they make decisions can be challenging, which is a critical concern in applications like healthcare and finance where interpretability is essential.