# Chapter 4

# Neural network

# Artificial neural network (ANN)/digital brain

- Your brain controls everything you do, and it is much more powerful than any computer you can find.
- This complex organ sends messages using cells called neurons, and it never stops analyzing data, even as you sleep.
- Scientists are trying to understand the brain to create a digital version.
- For computers to do so, we need to create something called an artificial neural network, which has digital neurons connected into a complex net that resembles the structure of the brain.
- ANN is a ML program, or model, that makes decisions in a manner similar to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions.

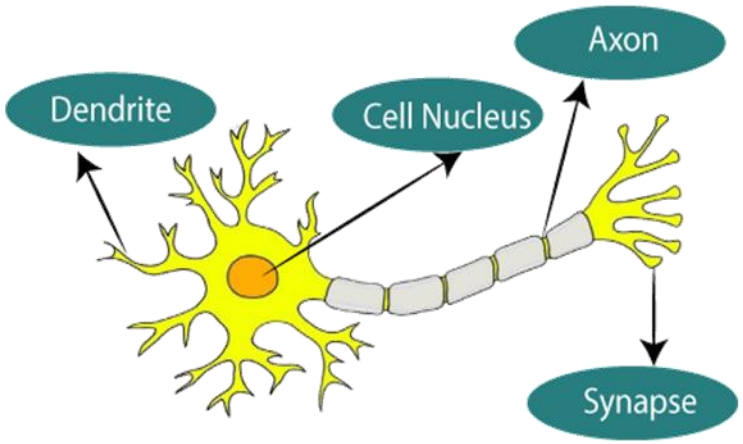
# How brain work?

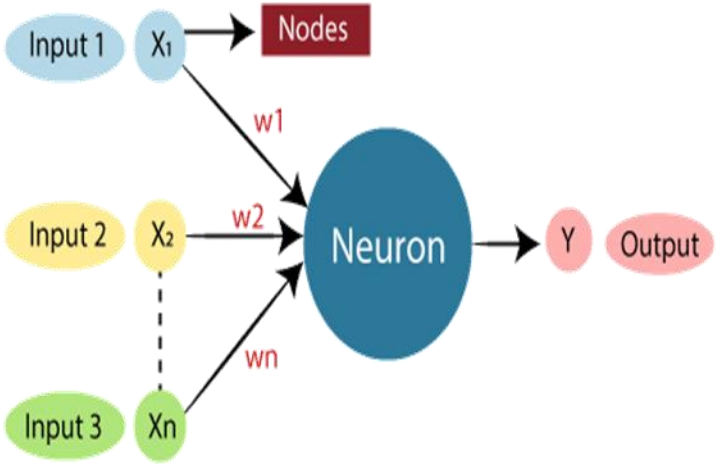
- The brain is composed of many neurons connected in a complex neural network.
- Each neuron has a cell body, containing the nucleus, and extensions from the cell body called the axon and dendrites.
- Dendrites receive electrical signals coming into the cell, while the axon transmits an electrical signal away from the cell, toward other neurons.
- The connection between each neurons is represented by synapse
- The place where two neurons meet to transfer signals from one to the other is called the synapse.

# Con't…

- Even when neurons are not sending an electrical impulse to other neurons, there is always a low, background level of electrical activity traveling along neurons.
- Scientists call this background electricity noise.
- When the neuron transmits an electrical signal, this is seen as a sharp spike or a peak on the graph.
- Thus, we can think of the neuron as existing in one of two states, either "off" (noise) or "on" (sending a sharp electrical signal).
- These states can be represented in the language of mathematics with two symbols: "0" (off) and "1" (on).
- The language of 1 and 0s is known as binary language which is a machine language that represent data using two symbol system(0 and 1) and is is language of computers

# Comparison of Ann and human brain




| Biological Neural Network | Artificial Neural Network |
|---|---|
| Dendrites | Inputs |
| Cell nucleus | Nodes |
| Synapse | Weights |
| Axon | Output |

# Creating an Artificial Neural Network

- Imagine creating a large, 3D structure from pipes of various shapes and sizes. Each pipe can be connected to many other pipes and has a valve that can be opened or closed.
- As a result, you end up with a million combinations of pipe connections.
- Now, let is connect the pipe contraption to a water tap.
- Pipes of different sizes will allow the water to flow at different speeds, and if the valves are closed, the water will not flow.
- The water represents the data that is transferred in the brain, while the pipes represent neurons.
- The valves represent the connections between neurons—the synapses/weights

# Con’t…

- Scientists are trying to create a digital brain that connects digital neurons like our imaginary water pipes.
- Based on the binary coding (0, 1)of neurons  they hope to create a thinking machine that is an accurate electronic version of a brain, full of digital neurons working together in a large, efficient, reliable network: an artificial neural network.

# ANN

- The digital neurons that make up an artificial neural network are called nodes.
- There are 3 layers in Ann(input(accept data), hidden (process)and output(return result/prediction))
- Each node has a special feature, called  weight, which is programmed by the developers.
- The weight of a node can be compared to the valves in our imaginary pipe structure, or the synapses in the brain—valves regulate the strength of the incoming signal.
- Now let is imagine that the pipes in our structure lead to a tank.
- The tank represents an artificial neuron.
- Each valve regulates the amount of water entering the tank.

# Con’t…

- The total amount of water coming in from the various pipes is the “input” to the tank, which is called the input signal in an ANN.
- The valves represent the weights of nodes, which regulate the strength of the signals coming into the nodes from the environment and other neurons.
- Meanwhile, the tank filled with water is like output data
- Each node in the ANN has multiple inputs, representing inputs signals from the environment or other neurons.

# Con't…

- When the network is active, the node receives different data (signals, which can be represented by the numbers) through each input and multiplies the numbers by their assigned weight.
- The node then sums up all the input signals to obtain the total, which is the output signal.
- Neurons regularly experience a low level of electrical noise, which is not strong enough to transmit a signal i.e. noise
- Well, in the ANN, if the output signal is below a predefined threshold, the node does not pass data on to the next layer—it is considered noise.
- If the number exceeds the threshold, the node sends the output signal to the next layer—the same way a signal is sent across a synapse when the electrical activity is high enough.
- This all happens in the binary language of 1 and 0s.

# Activation function

- While building a NN, one key decision is selecting the *Activation Function* for both the hidden layer and the output layer.
- Activation functions decide whether a neuron should be activated.
- An activation function is a mathematical function applied to the output of a neuron.
- It introduces non-linearity into the model, allowing the network to learn and represent complex patterns in the data.
- Without non-linearity feature a neural network would behave like a linear regression model no matter how many layers it has
- The activation function decides whether a neuron should be activated by calculating the weighted sum of inputs and adding a bias term.
- This helps the model make complex decisions and predictions by introducing non-linearity to the output of each neuron.

# Importance of Non-Linearity in NNs

- Neural networks consist of neurons that operate using weights, biases, and activation functions.
- In the learning process, these weights and biases are updated based on the error produced at the output—a process known as back propagation.
- Activation functions enable back propagation by providing gradients that are essential for updating the weights and biases.
- Without non-linearity, even deep networks would be limited to solving only simple, linearly separable problems.
- Activation functions empower neural networks to model highly complex data distributions and solve advanced deep learning tasks.
- Adding non-linear activation functions introduce flexibility and enable the network to learn more complex and abstract patterns from data.

# Types of Activation Functions in NN

- Linear Activation Function
- Non-Linear Activation Functions
  - Relu
  - Sigmoid
  - Tanh

# Back propagation

- **Back propagation** is a powerful algorithm in deep learning, primarily used to train ANN, particularly feed-forward networks.
- It works iteratively, minimizing the cost function by adjusting weights and biases.
- In each epoch, the model adapts these parameters, reducing loss by following the error gradient.
- Back propagation often utilizes optimization algorithms like gradient descent  or stochastic gradient descent
- The algorithm computes the gradient using the chain rule from calculus, allowing it to effectively navigate complex layers in the neural network to minimize the cost function.

# Why is Back propagation Important?

- Back propagation plays a critical role in how neural networks improve over time. Here's why:
- **Efficient Weight Update**: It computes the gradient of the loss function with respect to each weight using the chain rule, making it possible to update weights efficiently.
- **Scalability**: It scales well to networks with multiple layers and complex architectures, making deep learning feasible.
- **Automated Learning**: With back propagation, the learning process becomes automated, and the model can adjust itself to optimize its performance.

# Types of NN

**Feed-forward neural networks**

- Simple and pass information in 1 direction, through various input nodes, until it makes it to the output node.
- The network might or might not have many hidden node layers, making their functioning more interpretable.
- It's prepared to process large amounts of noise.

**Recurrent neural networks**

- Complex in nature and save the output of processing nodes and feed the result back into the model.
- This is how the model learns to predict the outcome of a layer.
- Each node in the RNN model acts as a memory cell, continuing the computation and execution of operations.

# Con’t…

**Convolutional neural networks**

- A network architecture for deep learning that learns directly from data.
- Useful for finding patterns in images to recognize objects, classes, and categories.

**De-convolutional neural networks**

- Use a reversed CNN learning process and try to find lost features or signals that might have originally been considered unimportant to the CNN system's task.
- This network model can be used in image synthesis and analysis.

# Con’t…

**Modular neural networks**

- These contain multiple neural networks working separately from one another.
- The networks don't communicate or interfere with each other's activities during the computation process.
- Consequently, complex or big computational processes can be performed more efficiently.

**Perceptron neural networks**

- Represent the most basic form of NNs and were introduced in 1958 by Frank Rosenblatt, an American psychologist, considered to be the father of deep learning.
- The perceptron is specifically designed for binary classification tasks, enabling it to differentiate between two classes based on input data.

# Con’t…

**Multilayer perceptron networks**

- MPN consist of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer.
- Each layer is fully connected to the next, meaning that every neuron in one layer is connected to every neuron in the subsequent layer.

**Radial basis function networks**

- Use radial basis functions as activation functions.
- They're typically used for function approximation, time series prediction and control systems.

# The perceptron

- Perceptron is a type of NN that performs binary classification that maps input features to an output decision, usually classifying data into one of two categories, such as 0 or 1.
- Perceptron consists of a single layer of input nodes that are fully connected to a layer of output nodes.
- It is particularly good at learning linearly separable patterns.

# Types of perceptron

- Single-Layer Perceptron is limited to learning linearly separable patterns.
- Effective for tasks where the data can be divided into distinct categories through a straight line and powerful in its simplicity, it struggles with more complex problems where the relationship between inputs and outputs is non-linear.
- Multi-Layer Perceptron possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

# How does Perceptron work?

- A weight is assigned to each input node of a perceptron, indicating the importance of that input in determining the output.
- The Perceptron's output is calculated as a weighted sum of the inputs, which is then passed through an activation function to decide whether the Perceptron will fire.
- The weighted sum is computed as:

$z=w1x1+w2x2+…+wnxn=XTW$$z=w1x1+w2x2+…+wnxn=XTW$

- The step function compares this weighted sum to a threshold.
- If the input is larger than the threshold value, the output is 1; otherwise, it's 0.

# Con't…

- In a fully connected layer, also known as a dense layer, all neurons in one layer are connected to every neuron in the previous layer.
- The output of the fully connected layer is computed as:

$f_{W,b}(X)=h(XW+b)$

- where X is the input *W* is the weight for each inputs neurons and b is the bias and h is the step function.
- During training, the Perceptron's weights are adjusted to minimize the difference between the predicted output and the actual output.
- This is achieved using supervised learning algorithms like the delta rule or the Perceptron learning rule.

# Con’t…

- The weight update formula is:

$w_{i,j} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$

Where:

- $w_{i,j}$ is the weight between the *ith* input and *jth* output neuron,
- $x_i$ is the ith input value,
- $y_j$ is the actual value, and $\hat{y}_j$ is the predicted value,
- $\eta$ is the **learning rate**, controlling how much the weights are adjusted.
- This process enables the perceptron to learn from data and improve its prediction accuracy over time.

# Non linear regression

- Sometimes linear models are not sufficient to capture the real-world phenomena, and thus nonlinear models are necessary.
- Non-linear regression in ANNs leverages the network's ability to model complex relationships in data using its layered structure and non-linear activation functions.
- Unlike linear regression, which assumes a straight-line relationship, ANNs can identify approximate highly non-linear patterns by passing inputs through multiple layers of neurons.
- These neurons, activated by functions like ReLU or sigmoid, enable the network to capture intricate dependencies between variables.
- During training, the network adjusts its weights via optimization algorithms like gradient descent to minimize prediction errors.
- This makes ANNs highly effective for tasks involving non-linear and high-dimensional data.

# Two-Class Discrimination

- Two-class discrimination refers to the task of classifying data points into two distinct categories, such as "yes" or "no" or "positive" and "negative.“
- The network typically uses one output neuron with a sigmoid activation function, which outputs a probability between 0 and 1, with a threshold (e.g., 0.5) determining the final classification.
- During training, the ANN adjusts its weights using a loss function like binary cross-entropy to minimize the error between predicted probabilities and actual labels.
- By learning patterns in the input data, the ANN establishes a decision boundary—linear or non-linear—depending on the network's design, enabling it to distinguish between the two classes effectively.

# Multiclass Discrimination

- The task of classifying data points into one of three or more distinct categories or classes called **Multiclass discrimination**.
- This involves designing a network with an output layer that has one neuron per class, where each neuron outputs a score or probability representing the likelihood of the input belonging to that class.
- A common approach is to use the softmax activation function in the output layer to convert these scores into probabilities that sum to 1.
- The class with the highest probability is selected as the prediction.
- Multiclass discrimination is trained using a loss function like categorical cross-entropy, and it can handle tasks such as recognizing digits (0–9) in handwritten images or categorizing objects in images (e.g., cats, dogs, and birds).

# Multiple Hidden Layers

- **Multiple hidden layers** refer to the presence of more than one intermediate layer between the input and output layers.
- These layers allow the network to learn and represent increasingly complex patterns and hierarchical features in the data.
- Each hidden layer transforms the input it receives using weights, biases, and activation functions, passing the processed information to the next layer.
- Networks with multiple hidden layers are called **deep neural networks** and are particularly powerful for tasks like image recognition, natural language processing, and time-series prediction.
- By stacking layers, the network can model highly non-linear relationships, with early layers capturing basic features (e.g., edges in images) and deeper layers learning higher-level abstractions (e.g., object shapes or concepts).

# Training procedures

- The training procedure of ANN involves initializing weights and biases with small random values, followed by forward propagation, where input data passes through the layers to produce predictions.
- The predicted outputs are compared to true labels using a loss function to calculate the error.
- Through back propagation, the gradients of the loss with respect to the weights and biases are computed using the chain rule.
- These gradients are then used to update the weights and biases via an optimization algorithm like stochastic gradient descent (SGD) or Adam.
- This process is repeated for multiple epochs, with the network continuously improving its ability to minimize the loss and make accurate predictions.
- Validation on a separate dataset is often used to monitor performance and prevent over-fitting.

# Improving convergence

- Improving convergence in NN training involves using strategies that accelerate learning and help the model reach an optimal solution efficiently.
- Key techniques include adjusting the learning rate with methods like Adam, which adapt the rate dynamically.
- Proper weight initialization (e.g., Xavier or He initialization) prevents slow convergence or vanishing/exploding gradients.
- Batch normalization stabilizes learning by normalizing layer inputs, while regularization techniques like dropout and L2 regularization help prevent over fitting.

# Over-training

- Overtraining, also known as **over-fitting**, occurs when a neural network learns the training data too well, including its noise and fluctuations, at the expense of generalizing to new, unseen data.
- This typically happens when the model is too complex relative to the amount of training data, or when training goes on for too many epochs.
- As a result, the network performs well on the training set but fails to make accurate predictions on the validation or test sets, as it has essentially memorized the training data rather than learning the underlying patterns.
- Techniques like **regularization**, **early stopping**, and using a larger training dataset can help mitigate overtraining and ensure that the model generalizes effectively.

# Structuring the network

- Structuring the network in ANNs refers to designing the architecture, including decisions on the number of layers, the number of neurons in each layer, and the types of activation functions to be used.
- A well-structured network should strike a balance between complexity and performance, ensuring it is capable of capturing the underlying patterns in the data without being too simple (under fitting) or too complex (over fitting).
- The input layer corresponds to the features of the dataset, and the output layer corresponds to the predicted results, whether it be a classification label or a continuous value.

# Con’t…

- Hidden layers, which lie between the input and output, allow the network to learn hierarchical representations of data.
- Choosing the right structure depends on the problem, such as whether to use a shallow or deep network, or whether to use convolutional layers for image tasks or recurrent layers for sequence data.
- The structure also includes deciding on hyper-parameters like learning rate, batch size, and the optimizer used for training.

# Tuning the network size

- Tuning the network size involves adjusting the number of layers and neurons in each layer to find the optimal architecture that balances model complexity and performance.
- A network that is too small (with fewer neurons or layers) may under fit, failing to capture important patterns in the data, while a network that is too large may over fit, memorizing the training data without generalizing well to new data.
- Tuning the network size requires experimentation to determine the right balance for the specific problem, often guided by cross-validation, where different network configurations are tested to evaluate their performance.
- The choice of network size also depends on factors like the amount of available data, the complexity of the task, and computational resources, with larger networks typically needing more data and processing power.

Thanks