Data Science and Generative AI

by SATISH @

https://sathyatech.com

Deep Learning (DL)

Deep Learning and Neural Networks

The deep learning revolution started around 2010.

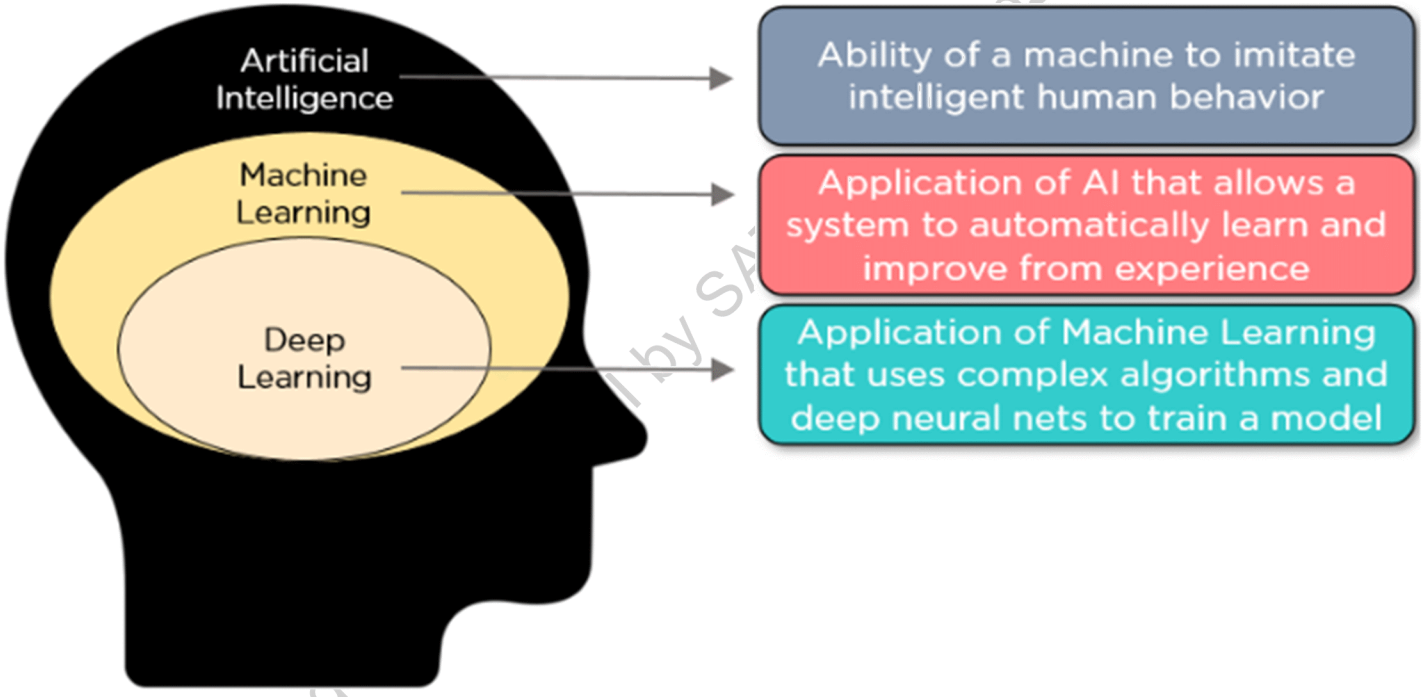
Since then, Deep Learning has solved many "unsolvable" problems.

The deep learning revolution was not started by a single discovery. It more or less happened when several needed factors were ready:

- Computers were fast enough
- Computer storage was big enough
- Better training methods were invented
- Better tuning methods were invented

What is Deep Learning?

Deep Learning is a part of machine learning that deals with algorithms inspired by the structure and function of the human brain. It uses artificial neural networks to build intelligent models and solve complex problems. We mostly use deep learning with unstructured data.



Let's now look understand the basics of neural networks in this Deep Learning with Python article.

Your AI/ML Career is Just Around The Corner!

AI Engineer Master's ProgramExplore Program

Neural Networks (NN)

Neural Networks is:

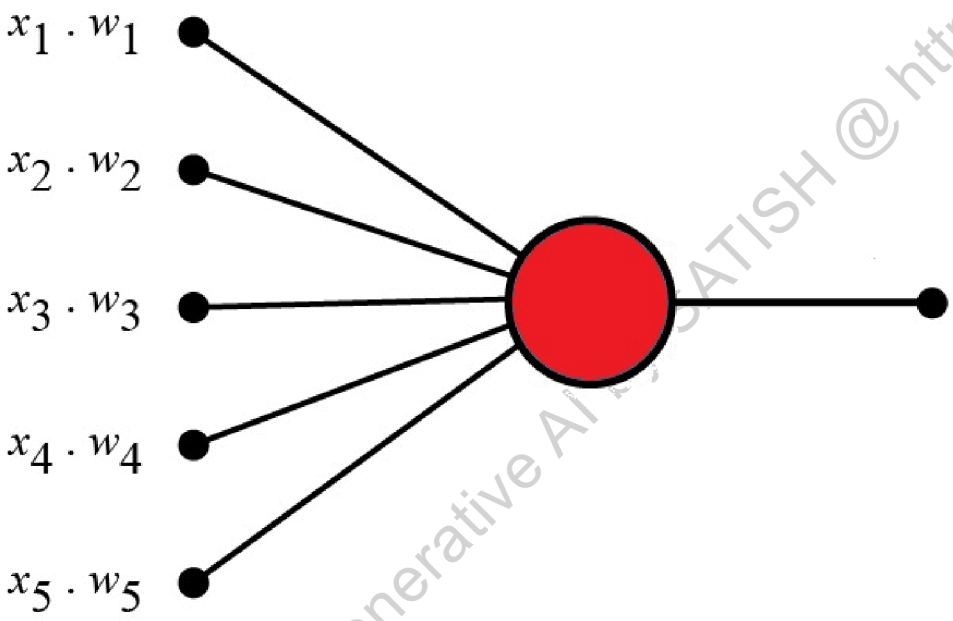
- A programming technique
- A method used in machine learning
- A software that learns from mistakes

Neural Networks are based on how the human brain works:
Neurons are sending messages to each other. While the neurons are trying to solve a problem (over and over again), it is strengthening the connections that lead to success and diminishing the connections that lead to failure.

Perceptrons

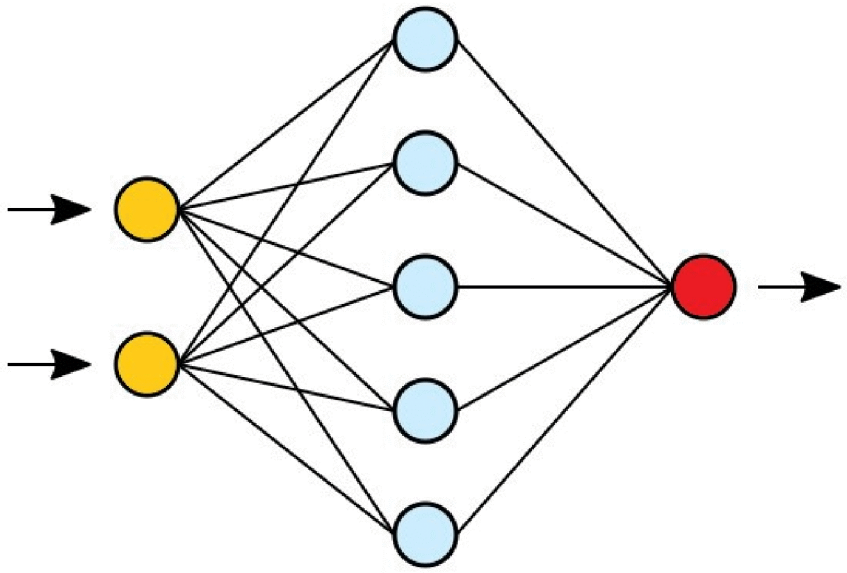
The **Perceptron** defines the first step into Neural Networks.

It represents a single neuron with only one input layer, and no hidden layers.



Neural Networks

Neural Networks are **Multi-Layer Perceptrons**.



In its simplest form, a neural network is made up from:

- An input layer (yellow)
- A hidden layer (blue)
- An output layer (red)

In the **Neural Network Model**, input data (yellow) are processed against a hidden layer (blue) before producing the final output (red).

The First Layer:
The yellow perceptrons are making **simple decisions** based on the input. Each single decision is sent to the perceptrons in the next layer.

The Second Layer:
The blue perceptrons are making decisions by weighing the results from the first layer. This layer make more **complex decisions** at a more abstract level than the first layer.

Neurons

Scientists agree that our brain has between 80 and 100 billion neurons.

These neurons have hundreds of billions connections between them.

Neurons (aka Nerve Cells) are the fundamental units of our brain and nervous system.

The neurons are responsible for receiving input from the external world, for sending output (commands to our muscles), and for transforming the electrical signals in between.

Artificial Neural Networks are normally called Neural Networks (NN).

Neural networks are in fact multi-layer **Perceptrons**.

The perceptron defines the first step into multi-layered neural networks.

Neural Networks are the essence of **Deep Learning**.

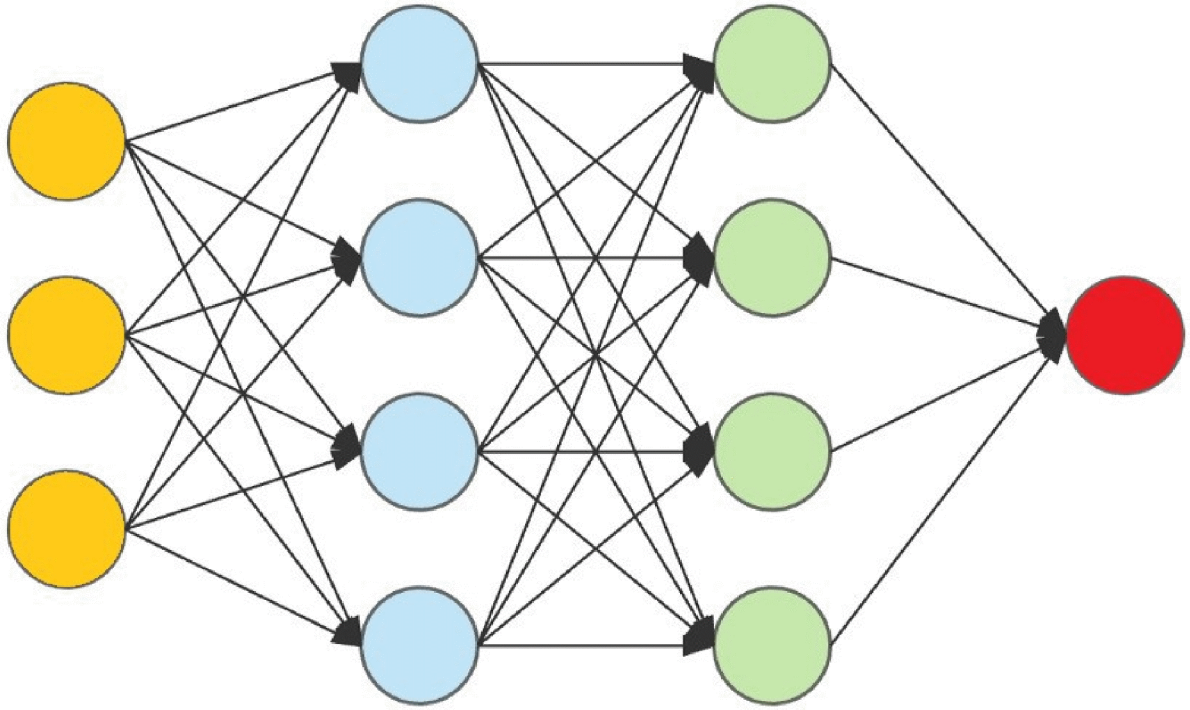
Neural Networks are one of the most significant discoveries in history.

Neural Networks can solve problems that can NOT be solved by algorithms:

- Medical Diagnosis
- Face Detection
- Voice Recognition

The Neural Network Model

Input data (Yellow) are processed against a hidden layer (Blue) and modified against another hidden layer (Green) to produce the final output (Red).



What Is Deep Learning?

Deep learning is a machine learning subset that makes computers do what comes naturally to humans: learn by example.

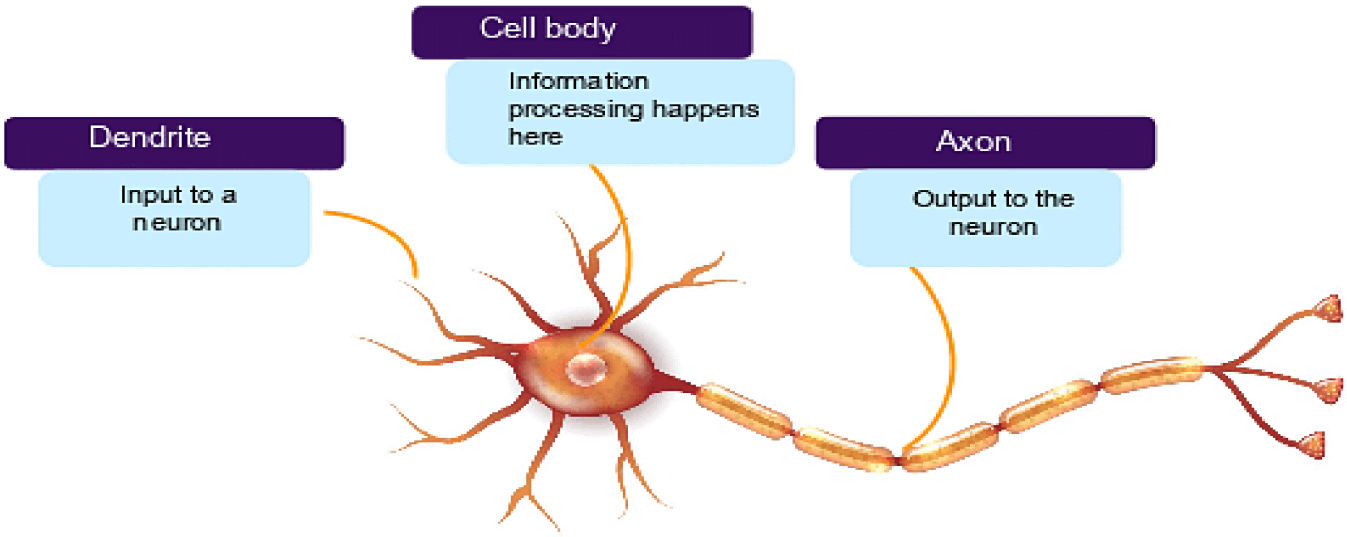
Machines get trained with images as examples, a process very different from hardwiring a computer program to recognize something and learn. You don't control how it knows; you control the aspects that go into it. The computer identifies the object based on the images fed earlier.

Scientists built a synthetic form of a biological neuron that powers any deep learning-based machine.

So, what is a neural network?

What Is a Neural Network?

To understand how an artificial neuron works, we should first understand how a biological neuron works.



Dendrites

These receive information or signals from other neurons that get connected to it.

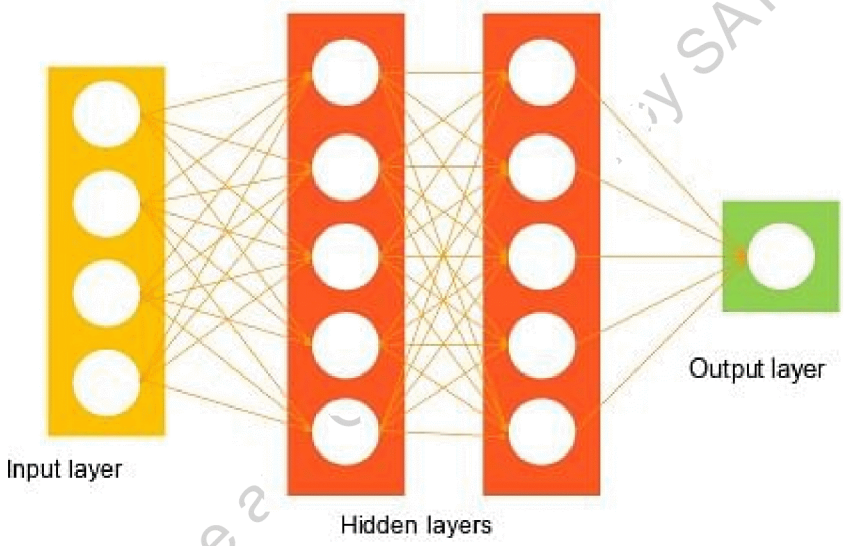
Cell Body

Information processing happens in a cell body. These take in all the information coming from the different dendrites and process that information.

Axon

It sends the output signal to another neuron for the flow of information. Here, each of the flanges connects to the dendrite or the hairs on the next one.

The image shown below depicts an ANN.



The network starts with an input layer that receives input in data form.

The lines connected to the hidden layers are called weights, and they add up on the hidden layers. Each dot in the hidden layer processes the inputs, and it puts an output into the next hidden layer and, lastly, into the output layer.

Looking at the above two images, you can observe how an ANN replicates a biological neuron.

- Input to a neuron - input layer
- Neuron - hidden layer
- Output to the next neuron - output layer

A neural network is a system of hardware or software patterned after the operation of neurons in the human brain. Neural networks, also called artificial neural networks, are a means of achieving deep learning.

When you want to figure out how a neural network functions, you need to look at neural network architecture.

Deep Learning

Classical programming uses programs (algorithms) to create results:

Traditional Computing

Data + Computer Algorithm = Result

Machine Learning uses results to create programs (algorithms):

Machine Learning

Data + Result = Computer Algorithm

Machine Learning

Machine Learning is often considered equivalent with Artificial Intelligence.

This is not correct. Machine learning is a subset of Artificial Intelligence.

Machine Learning is a discipline of AI that uses data to teach machines.

"Machine Learning is a field of study that gives computers the ability to learn without being programmed."

Intelligent Decision Formula

- Save the result of all actions
- Simulate all possible outcomes
- Compare the new action with the old ones
- Check if the new action is good or bad
- Choose the new action if it is less bad
- Do it all over again

Deep Learning is a subset of Machine Learning using neural networks to model and solve complex problems. A complex problem is broken down into smaller parts.

Deep Learning includes neural networks with more than one layer, called “deep learning”. It learns complicated representations of data through hierarchical learning processes. Neural Networks are the backbone of deep learning.

Components of Deep Learning

The following are some of the vital components of Deep Learning:

1. **Neural Networks:** It consists of layers of neurons i.e. nodes. Each neuron takes input, applies a transformation, and passes the output to the next layer.
2. **Layers:** Includes the following three layers:
   - Input Layer: Receives the initial data.

- Hidden Layers: Intermediate computations and feature extraction are performed.
- Output Layer: The final prediction or classification is produced

3. **Activation Functions:** An activation function determines what is to be done with the neurons i.e. it should be activated or not. Example: ReLU, Sigmoid, etc. Here, ReLU is a Rectified Linear Unit, a non-linear activation function used in deep neural networks.
4. **Training:** This adjusts the neurons' weights using backpropagation and optimization algorithms to minimize the error between predicted and actual outcomes.

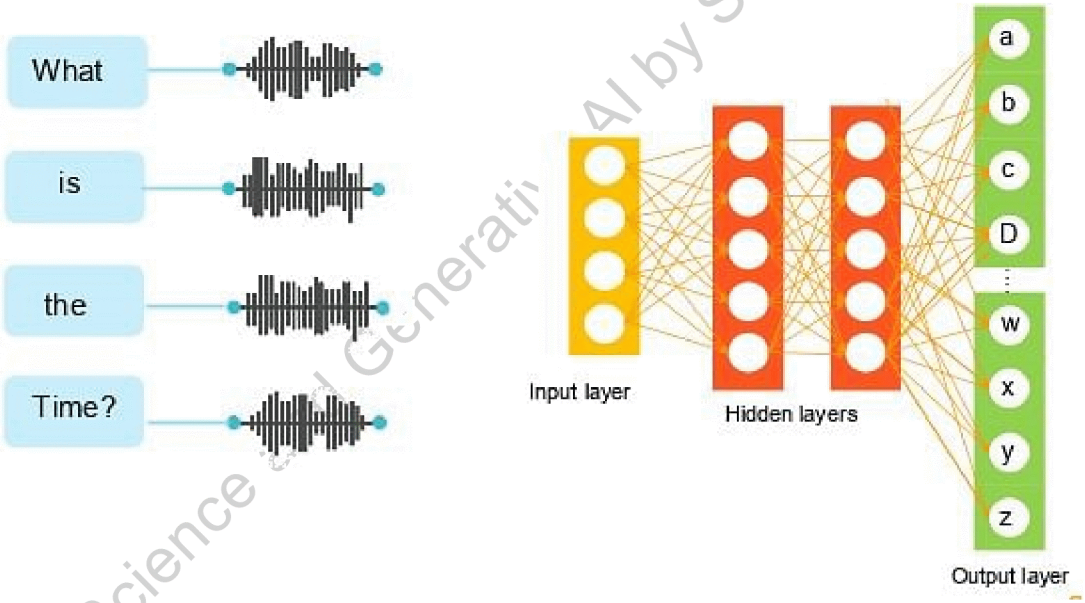
The Architecture of a Neural Network

Have you ever asked Siri a question? The device answers accurately. Let's take a closer look and see how the virtual assistant accomplishes this feat of speech recognition.

There are five recognized types of neural networks.

- Single-layer feed-forward network.
- Multilayer feed-forward network.
- Single node with its own feedback.
- Single-layer recurrent network.
- Multilayer recurrent network.

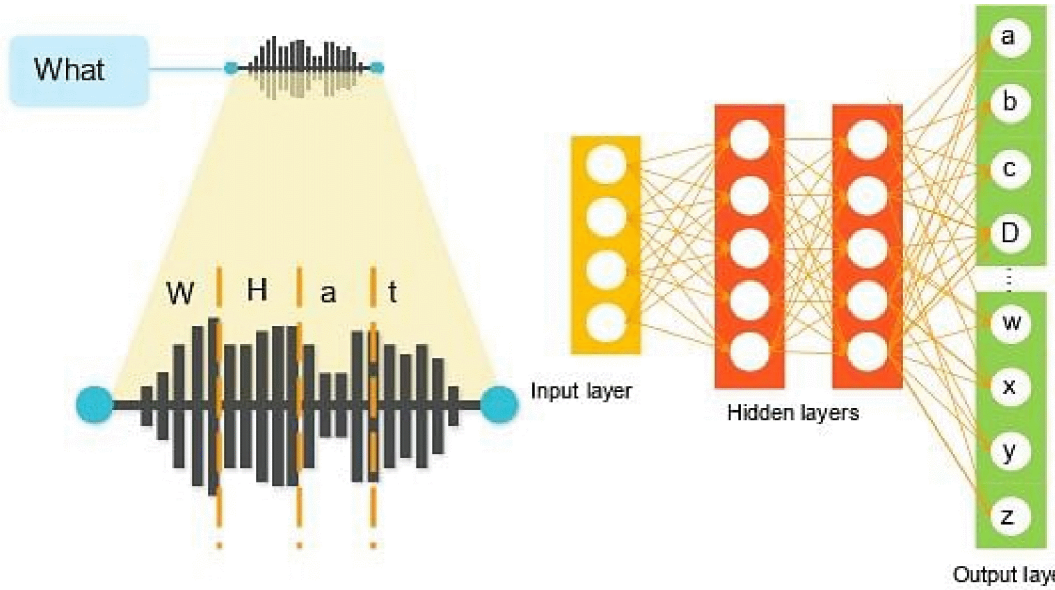
Consider, for instance, the neural network shown below, an example of a single-layer recurrent network:



There are input, hidden, and output layers on the network. But, first, the network needs to recognize the sentence: What is the time?

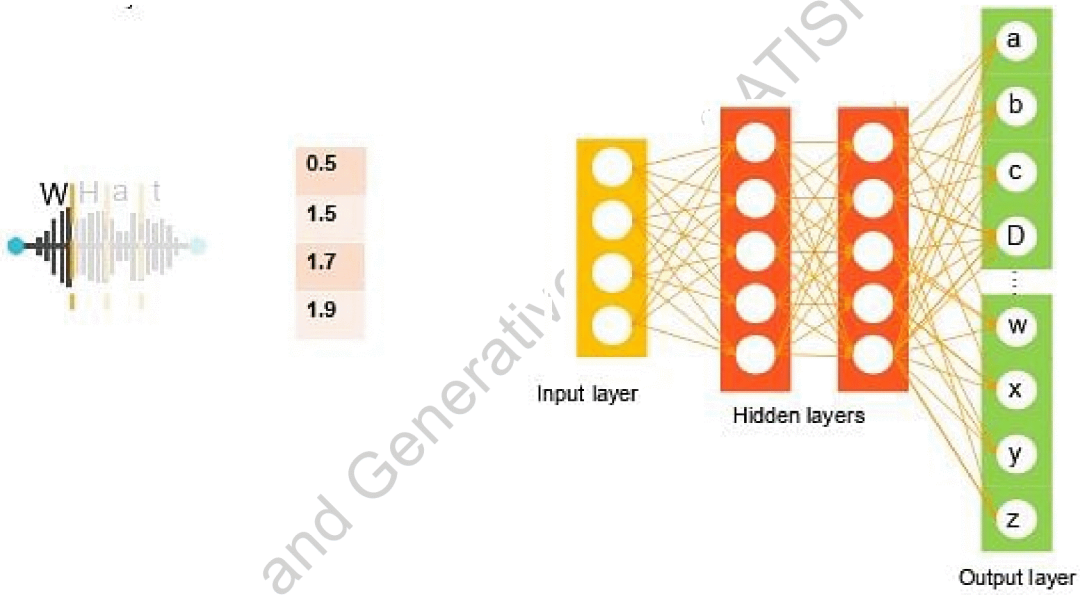
Here, each word comes in as a pattern of sound. Then, the sentence gets sampled into discrete sound waves.

Let's consider the first word: What

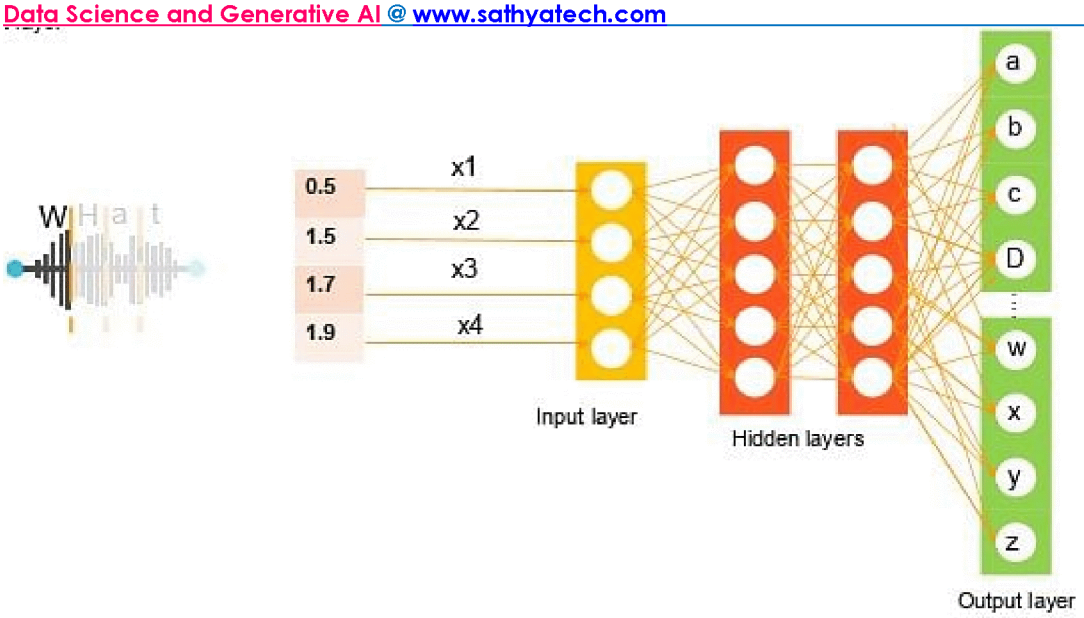


You can see the waveform is split based on every letter. Now we will split the sound wave for the letter W into smaller segments.

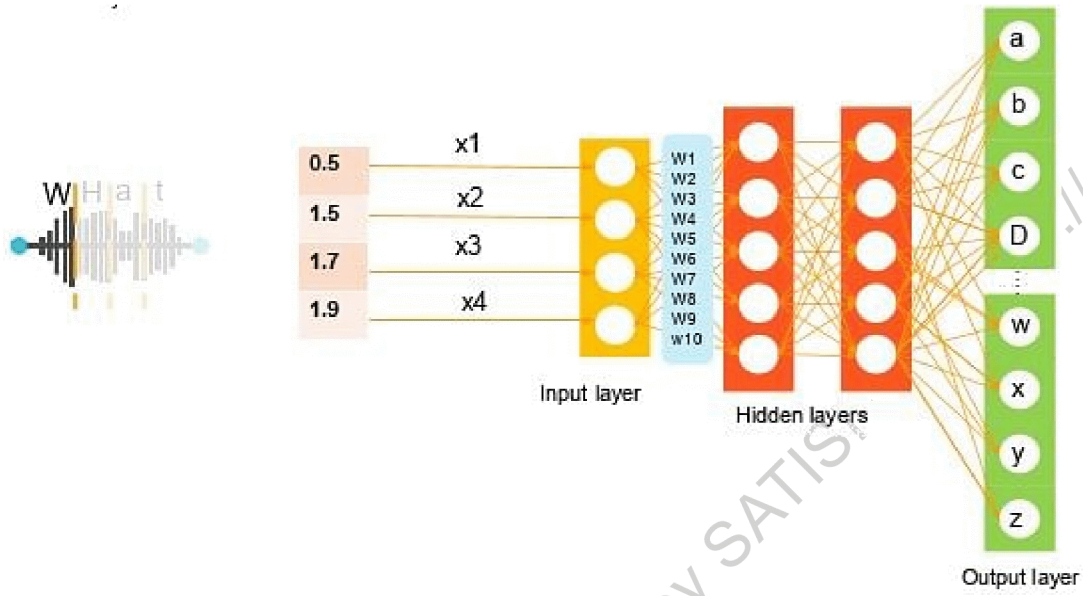
The amplitude varies in the sound wave when we analyze the letter 'W,' as shown below.



We collect the values at intervals and form an array. Then, different amplitudes come in for other letters, and we feed the variety of amplitudes to the input layer.



Random weights get assigned to each interconnection between the input and hidden layers.



We always start with the random key, as assigning a preset value to the weights takes a significant amount of time when training the model.

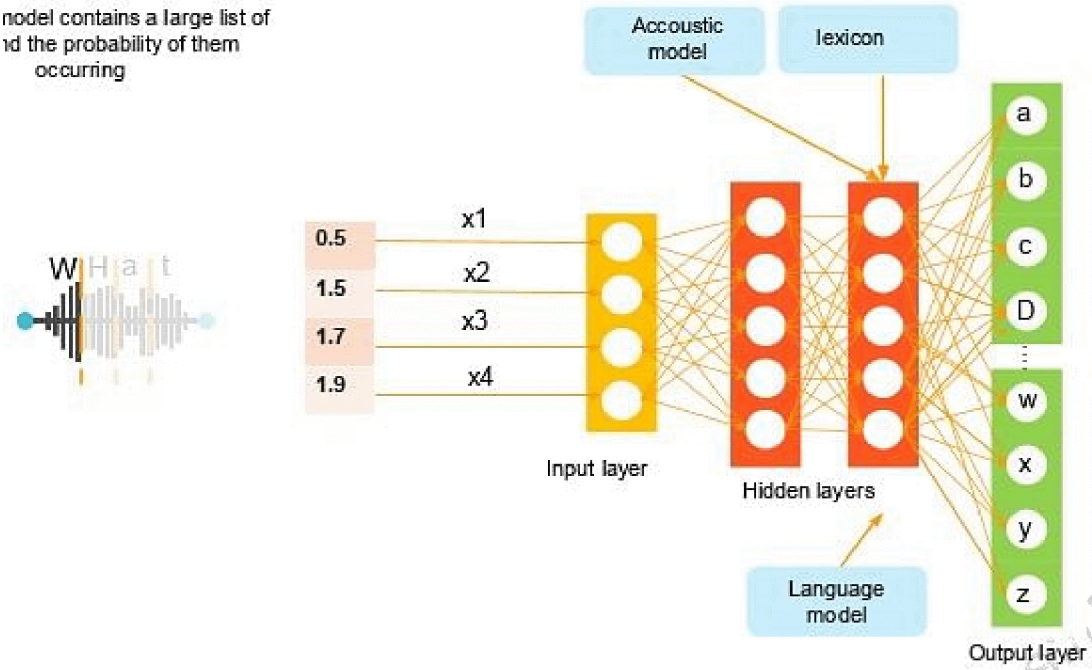
The weights get multiplied with the inputs, and a bias is added to form the transfer function.

$$\sum_{i=1}^{n} w_i x_i + b$$

Weights get assigned to the interconnection between the hidden layers. The output of the transfer function is fed as an input to the activation function. The work from one hidden layer becomes the input to the next.

The acoustic model contains the statistical representation of each sound that makes a word. So we start building these acoustic models, and as these layers separate them, they'll start learning what the different models represent for other letters.

The lexicon contains the data for different pronunciations of every word. The lexicon is at the end, where we end up with the ABCD, identifying the other letters there.



Finally, we get our output letter. Following the same process for every word and letter, the neural network recognizes the sentence you said or your question.

Note that the terms “acoustic model” and “lexicon” are specific to the domain of understanding speech. When dealing with other input formats, you'll have different labels, but the process remains the same.

Advantages of Neural Network

ANN outputs aren't limited entirely by inputs and results given to them initially by an expert system. This ability comes in handy for robotics and pattern recognition systems.

This neural network has the potential for high fault tolerance and can debug or diagnose a network on its own. ANN can go through thousands of log files from a company and sort them out. It is currently a tedious task done by administrators, but it will save a significant amount of time, energy, and resources if it can be automated.

Nonlinear systems can find shortcuts to reach computationally expensive solutions. We see this in the banking industry, for example, where they work on a particular Excel

spreadsheet, and as time goes by, start building codes around it. In over 20 years, they might create a repertoire of all these functions, and the neural network rapidly comes up with the same answers otherwise done in days, weeks, or even a month, when done by a large bank.

So how do neural networks apply today?

Applications of Neural Network

With an enormous number of applications implementations every day, now is the most appropriate time to know about the applications of neural networks, machine learning, and artificial intelligence. Some of them are discussed below:

Handwriting Recognition

Neural networks are used to convert handwritten characters into digital characters that a machine can recognize.

Stock-Exchange prediction

The stock exchange is affected by many different factors, making it difficult to track and difficult to understand. However, a neural network can examine many of these factors and predict the prices daily, which would help stockbrokers.

Currently, this operation is still in its initial phases. However, you should know that over three terabytes of data a day are generated from the United States stock exchange alone. That's a lot of data to dig through, and you must sort it out before you start focusing on even a single stock.

Traveling Issues of sales professionals

This application refers to finding an optimal path to travel between cities in a given area. Neural networks help solve the problem of providing higher revenue at minimal costs. However, the Logistical considerations are enormous, and we must find optimal travel paths for sales professionals moving from town to town.

Image compression

The idea behind neural network data compression is to store, encrypt, and recreate the actual image again. Therefore, we can optimize the size of our data using image compression neural networks. It is the ideal application to save memory and optimize it.

So, what does the future of neural networks look like?

Future of Neural Networks

With the rapid pace that AI and machine learning are being adopted by companies today, we could see more advancements in the applications of neural networks in the foreseeable future. AI and machine learning will offer a wealth of personalized choices for users worldwide. For example, all mobile and web applications try to give you an enhanced customized experience based on your search history, and neural networks can make that possible.

Hyper-intelligent virtual assistants will make life easier. If you have ever used Google assistant, Siri, or any other products, you can see how they're slowly evolving. They may even predict your email responses in the future!

We can also expect intriguing discoveries on algorithms to support learning methods. However, we are just in the infant stage of applying artificial intelligence and neural networks to the real world.

Neural networks will be a lot faster in the future, and neural network tools can get embedded in every design surface. We already have a little mini neural network that plugs into an inexpensive processing board or even into your laptop. Instead of the software, focusing on the hardware would make such devices even faster.

Neural networks will also find their way into the fields of medicine, agriculture, physics, research, and anything else you can imagine. Neural networks will also find its way into the fields of medicine, agriculture, physics, research, and anything else you can imagine.

What is a Neural Network?

You've probably already been using neural networks on a daily basis. When you ask your mobile assistant to perform a search for you—say, Google or Siri or Amazon Web—or use a self-driving car, these are all neural network-driven. Computer games also use neural networks on the back end, as part of the game system and how it adjusts to the players, and so do map applications, in processing map images and helping you find the quickest way to get to your destination.

A neural network is a system or hardware that is designed to operate like a human brain.

Neural networks can perform the following tasks:

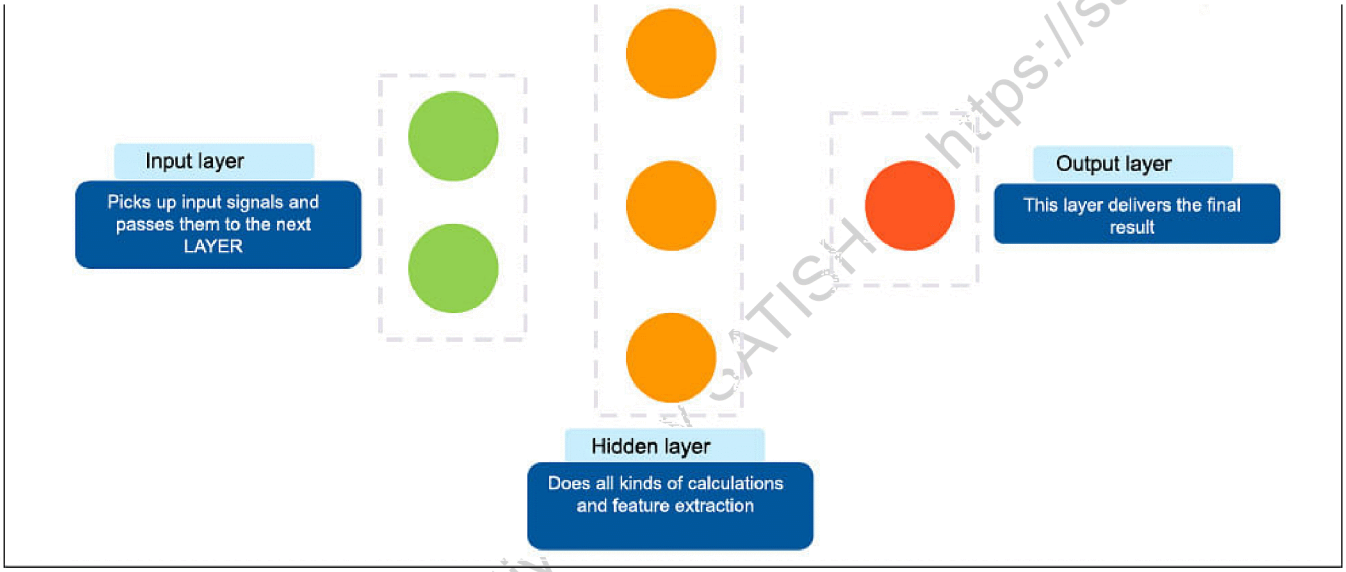
- Translate text

- Identify faces
- Recognize speech
- Read handwritten text
- Control robots
- And a lot more

Let us continue this neural network tutorial by understanding how a neural network works.

Working of Neural Network

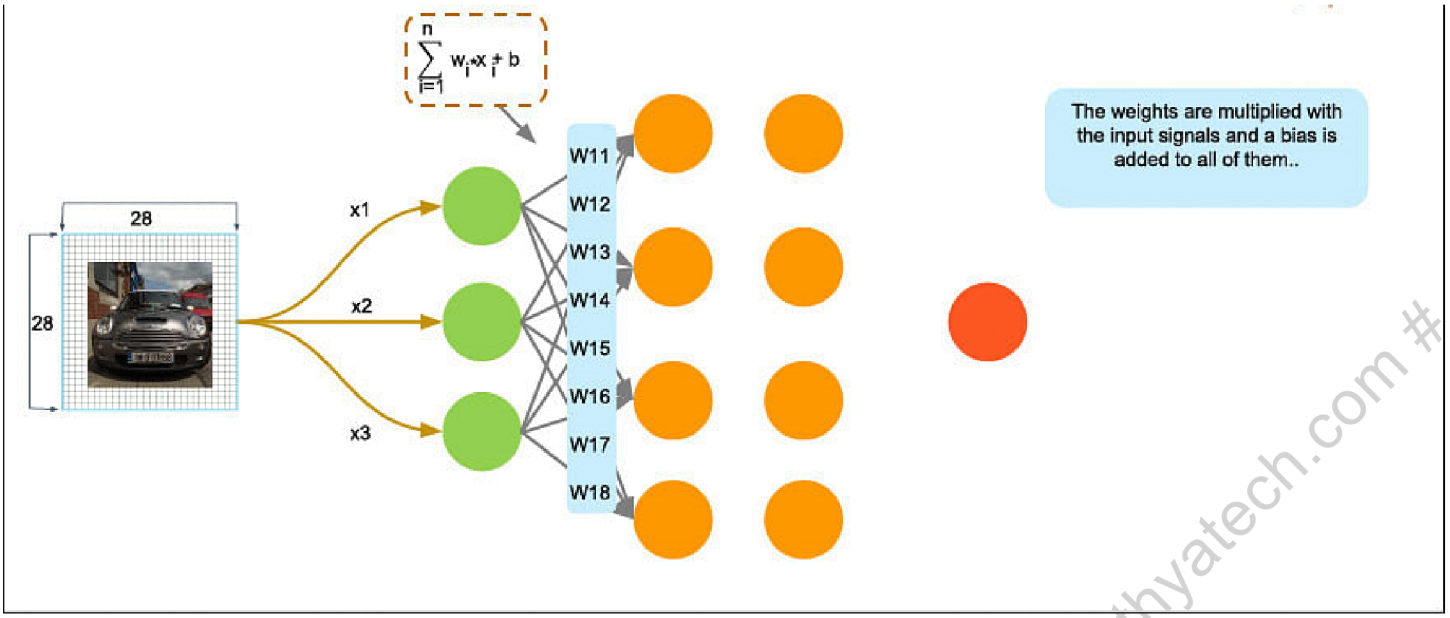
A neural network is usually described as having different layers. The first layer is the input layer, it picks up the input signals and passes them to the next layer. The next layer does all kinds of calculations and feature extractions—it's called the hidden layer. Often, there will be more than one hidden layer. And finally, there's an output layer, which delivers the final result.



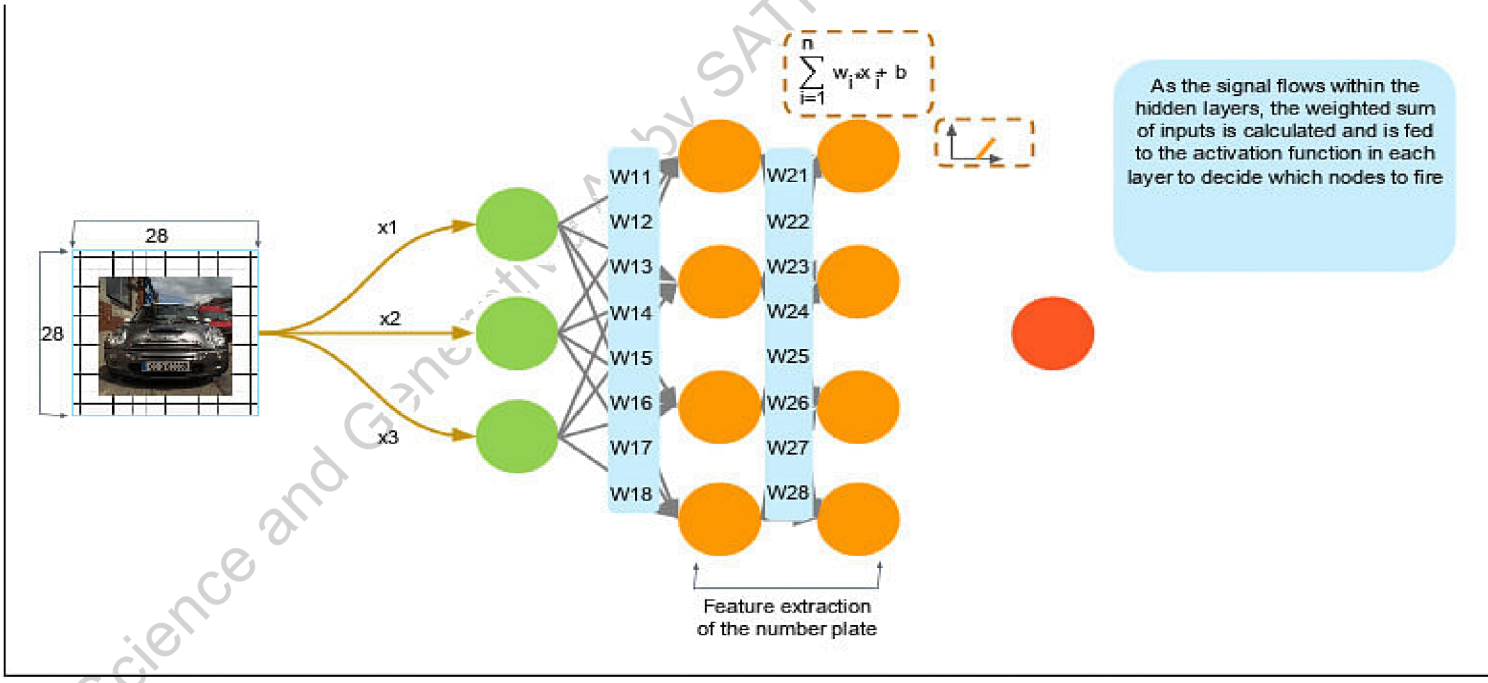
Let's take the real-life example of how traffic cameras identify license plates and speeding vehicles on the road. The picture itself is 28 by 28 pixels, and the image is fed as an input to identify the license plate. Each neuron has a number, called activation, which represents the grayscale value of the corresponding pixel, ranging from 0 to 1—it's 1 for a white pixel and 0 for a black pixel. Each neuron is lit up when its activation is close to 1.

Pixels in the form of arrays are fed into the input layer. If your image is bigger than 28 by 28 pixels, you must shrink it down, because you can't change the size of the input layer. In our example, we'll name the inputs as X1, X2, and X3. Each of those represents one of the pixels

coming in. The input layer then passes the input to the hidden layer. The interconnections are assigned weights at random. The weights are multiplied with the input signal, and a bias is added to all of them.



The weighted sum of the inputs is fed as input to the activation function, to decide which nodes to fire for feature extraction. As the signal flows within the hidden layers, the weighted sum of inputs is calculated and is fed to the activation function in each layer to decide which nodes to fire.



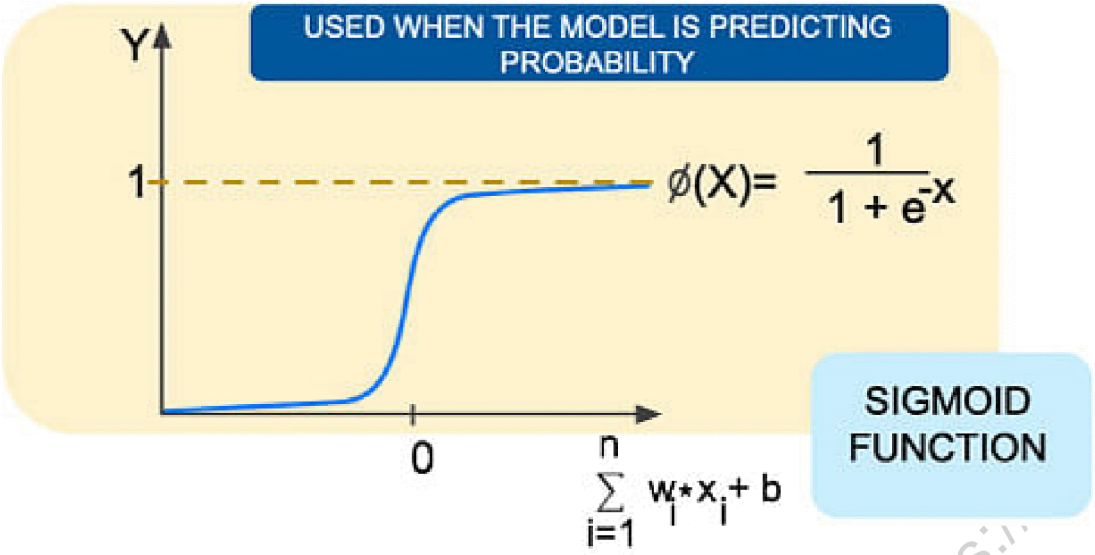
Here we'll take a detour to examine the neural network activation function.

Activation Function:

There are different types of activation functions.

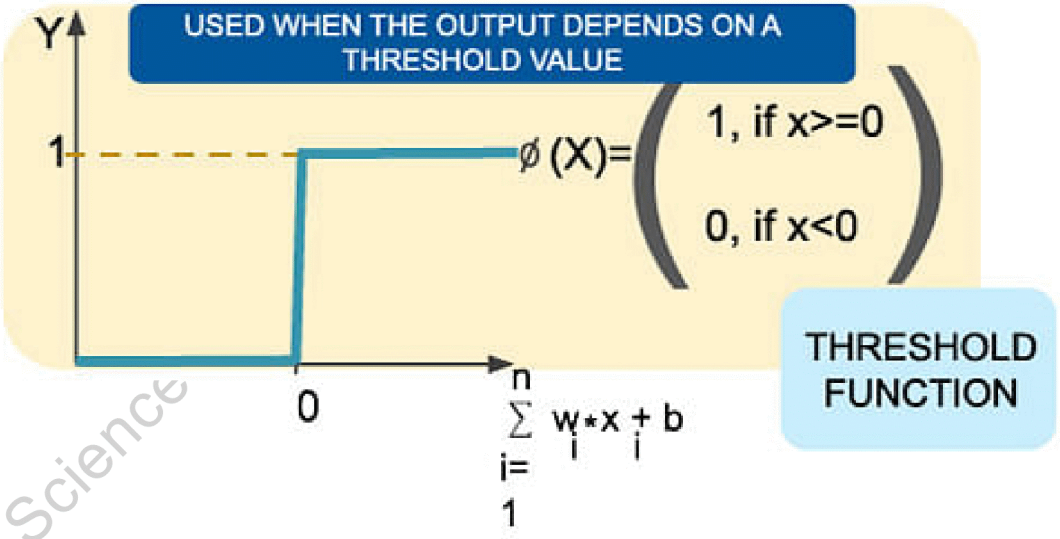
1. Sigmoid Function

The sigmoid function is used when the model is predicting probability.



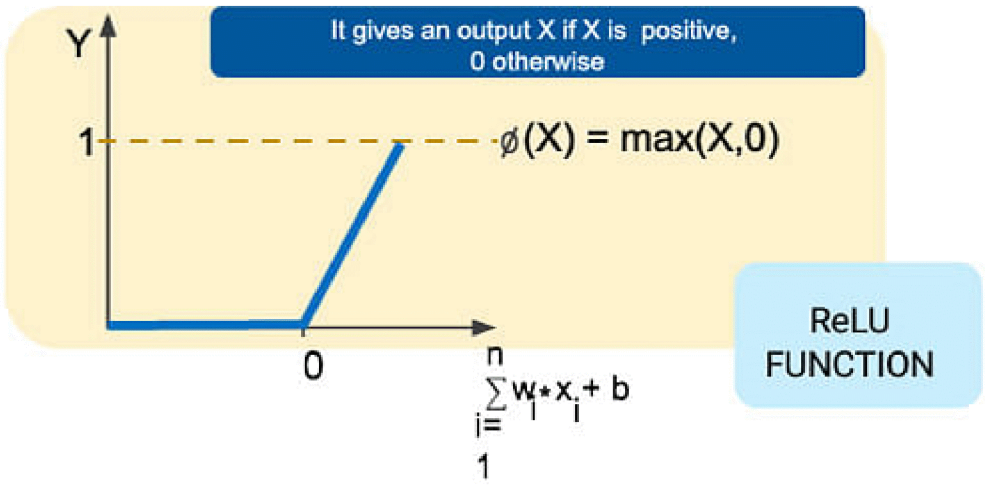
2. Threshold Function

The threshold function is used when you don't want to worry about the uncertainty in the middle.



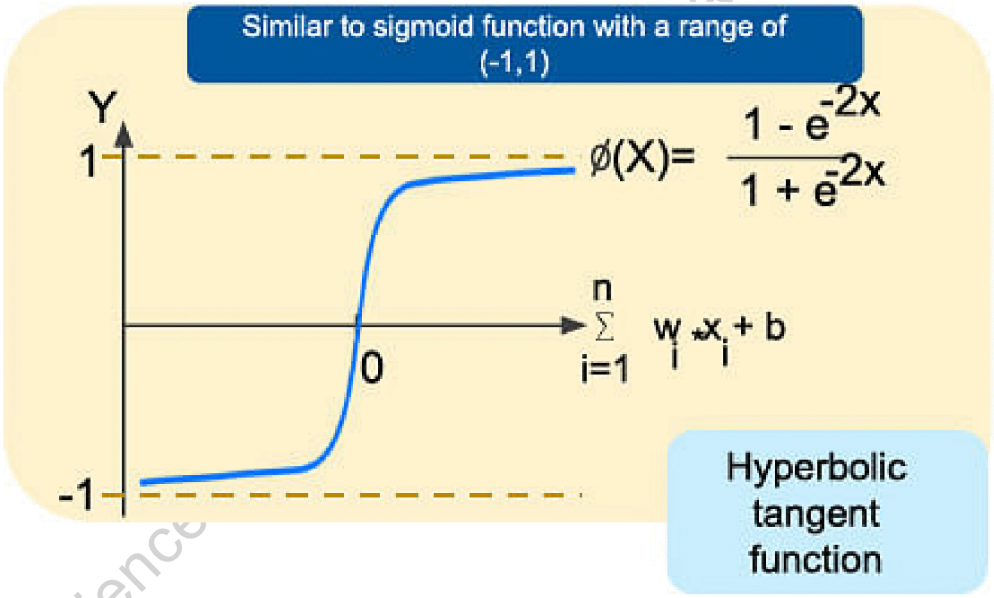
3. ReLU (rectified linear unit) Function

The ReLU (rectified linear unit) function gives the value but says if it's over 1, then it will just be 1, and if it's less than 0, it will just be 0. The ReLU function is most commonly used these days.

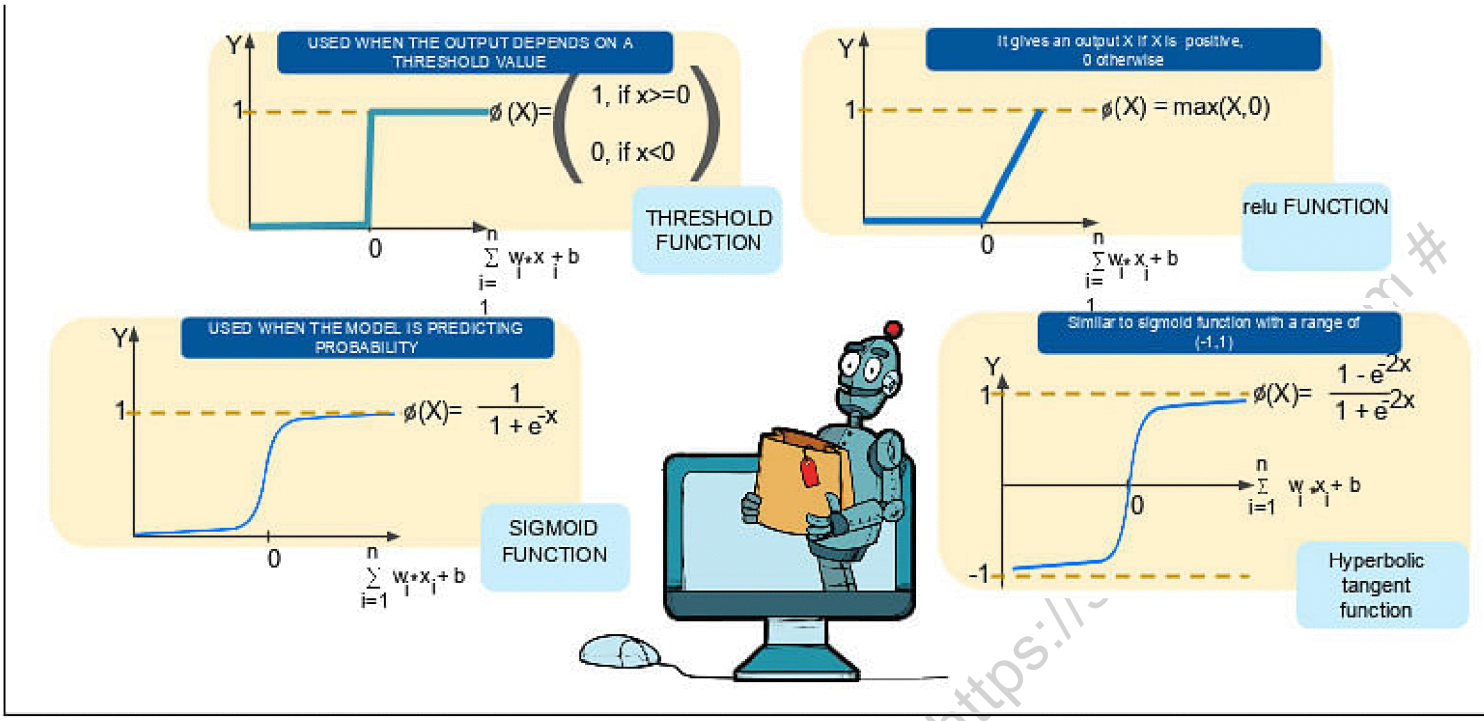


4. Hyperbolic Tangent Function

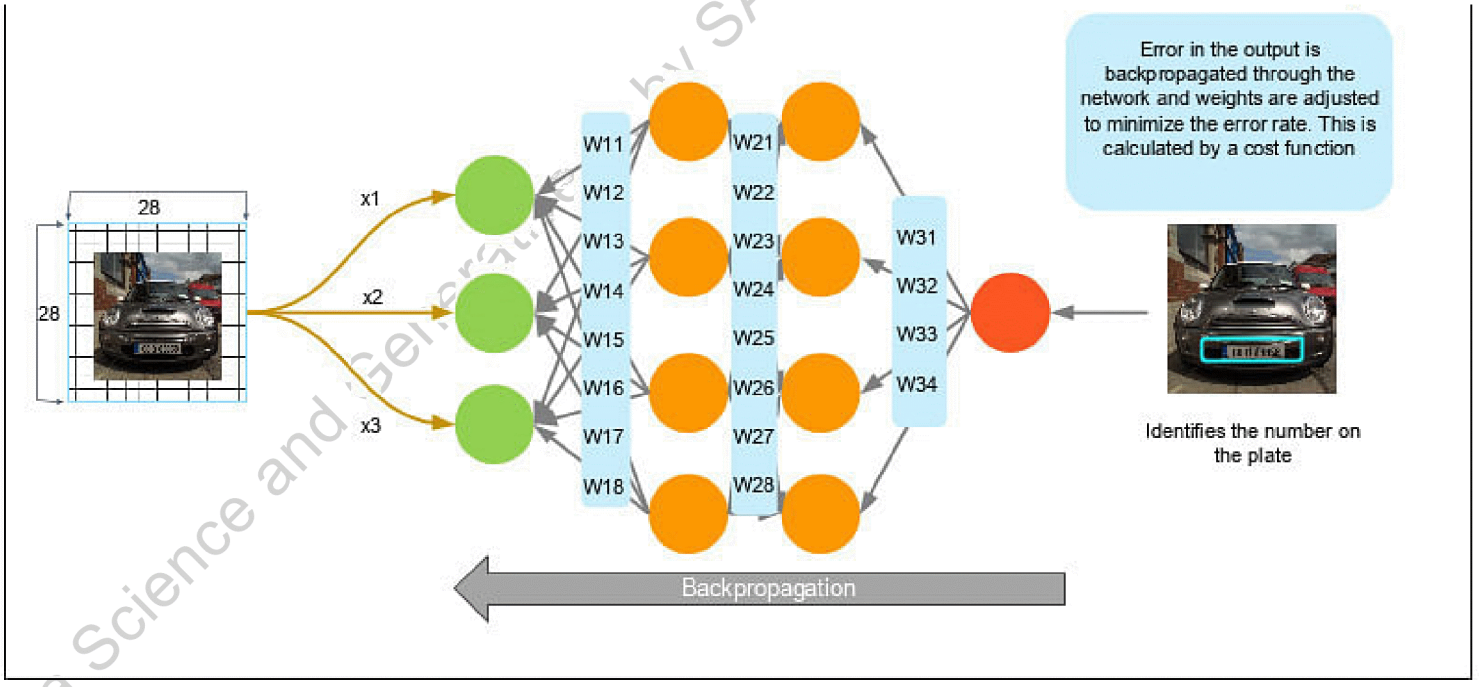
The hyperbolic tangent function is similar to the sigmoid function but has a range of -1 to 1.



Now that you know what an activation function is, let's get back to the neural network. Finally, the model will predict the outcome, applying a suitable application function to the output layer. In our example with the car image, optical character recognition (OCR) is used to convert it into text to identify what's written on the license plate. In our neural network example, we show only three dots coming in, eight hidden layer nodes, and one output, but there's really a huge amount of input and output.



Error in the output is back-propagated through the network and weights are adjusted to minimize the error rate. This is calculated by a cost function. You keep adjusting the weights until they fit all the different training models you put in.



The output is then compared with the original result, and multiple iterations are done for maximum accuracy. With every iteration, the weight at every interconnection is adjusted based on the error. That math gets complicated, so we're not going to dive into it here. But, we would look at how it's being done while executing the code for our use case.

In the following section of the neural network tutorial, let us explore the types of neural networks.

Types of Neural Networks

The different types of neural networks are discussed below:

1. Feed-forward Neural Network

This is the simplest form of ANN (artificial neural network); data travels only in one direction (input to output). This is the example we just looked at. When you actually use it, it's fast; when you're training it, it takes a while. Almost all **vision and speech recognition** applications use some form of this type of neural network.

2. Radial Basis Functions Neural Network

This model classifies the data point based on its distance from a center point. If you don't have training data, for example, you'll want to group things and create a center point. The network looks for data points that are similar to each other and groups them. One of the applications for this is **power restoration systems**.

3. Kohonen Self-organizing Neural Network

Vectors of random input are input to a discrete map comprised of neurons. Vectors are also called dimensions or planes. Applications include using it to recognize patterns in data like a **medical analysis**.

4. Recurrent Neural Network

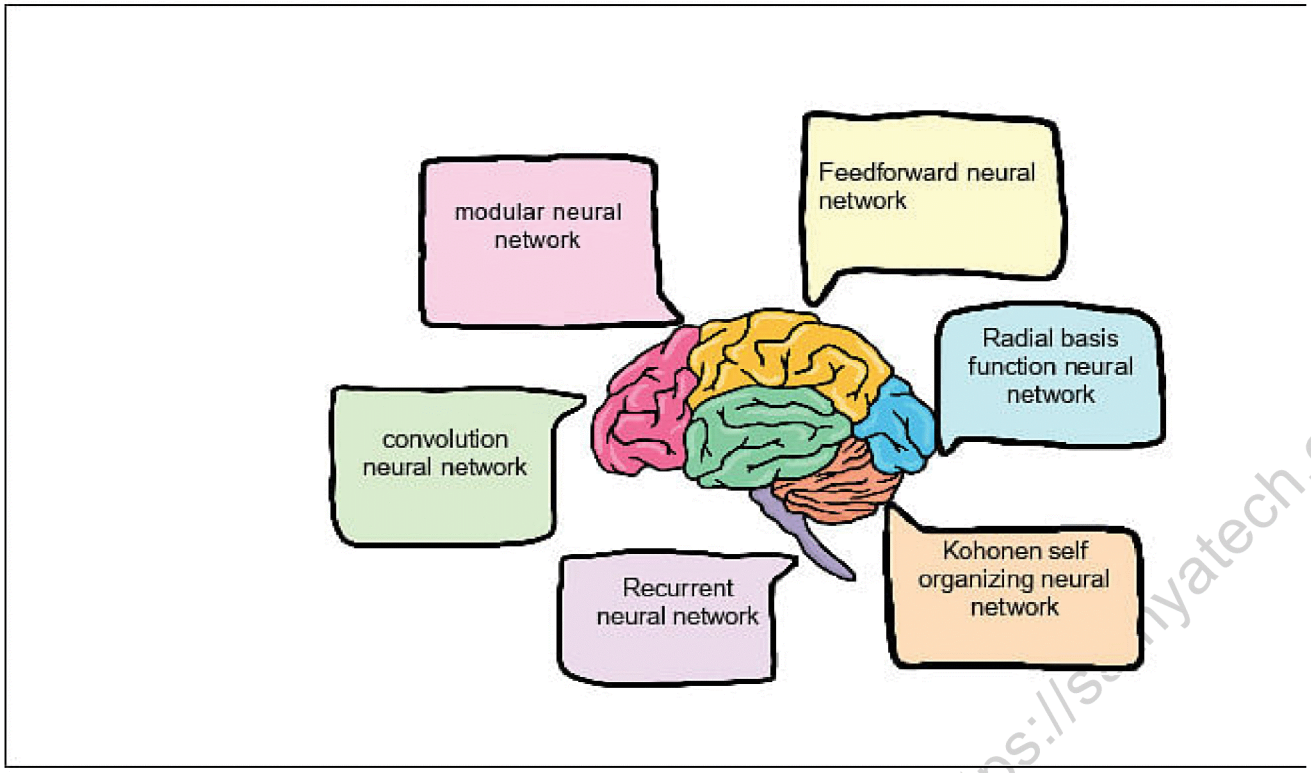
In this type, the hidden layer saves its output to be used for future prediction. The output becomes part of its new input. Applications include **text-to-speech conversion**.

5. Convolution Neural Network

In Convolution Neural Network, the input features are taken in batches—as if they pass through a filter. This allows the network to remember an image in parts. Applications include signal and image processing, such as **facial recognition**.

6. Modular Neural Network

This is composed of a collection of different neural networks working together to get the output. This is cutting-edge and is still in the research phase.



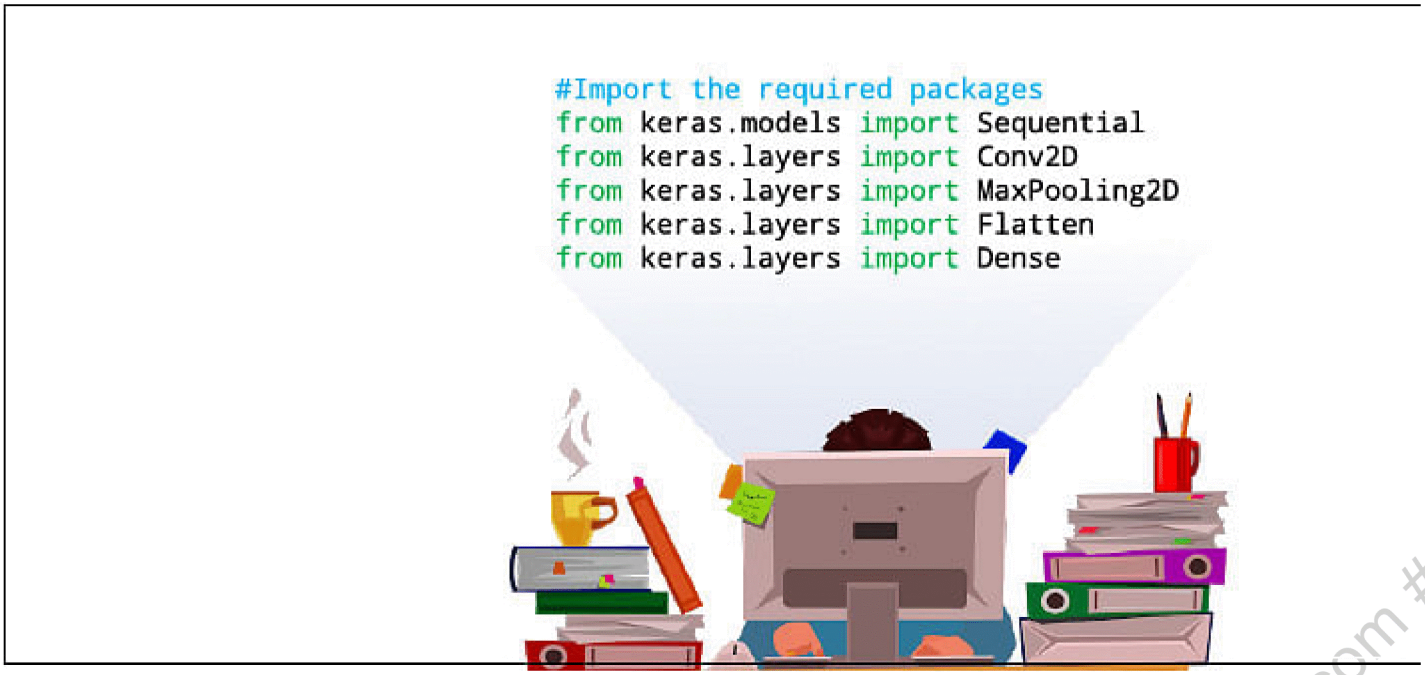
The next section of the neural network tutorial deals with the use of cases of neural networks.

Neural Network - Use Case

Let's use the system to tell the difference between a cat and a dog. Our problem statement is that we want to classify photos of cats and dogs using a neural network. We have a variety of dogs and cats in our sample images, and just sorting them out is pretty amazing!

Coding Language and Environment

We will implement our use case by building a neural network in Python(version 3.6). We're going to start by importing the required packages using Keras:



Let's talk about the environment we're working on. You can visit the official website of Keras and the first thing you'll notice is that Keras operates on top of TensorFlow, CNTK or Theano. TensorFlow is probably one of the most widely used packages with Keras.

Keras is user-friendly and has modularity and extensibility. It also works with Python, which is important because a lot of people in data science now use Python. When you're working with Keras, you can add layer after layer with the different information in each, which makes it powerful and fast.

Side note: Here, we're using Anaconda with Python in it, and we have created our own package called keraspython36. If you're doing a lot of experimenting with different packages, you probably want to create your own environment in there.

Artificial Neural Networks?

In this section, you will learn what artificial neural networks are and how they are inspired by the human brain. You will also learn some basic terminology and concepts related to artificial neural networks.

What are artificial neural networks?

Artificial neural networks, or ANN for short, are a type of machine learning technique that can learn from data and perform various tasks, such as classification, regression, clustering, anomaly detection, and more. Machine learning is a branch of artificial intelligence that aims to create systems that can learn from data and improve their performance without explicit programming.

Artificial neural networks are inspired by the structure and function of the human brain, which consists of billions of interconnected neurons that process and transmit information. Similarly, artificial neural networks are composed of layers of artificial neurons that can learn from data and perform various tasks.

What are artificial neurons?

Artificial neurons, or simply neurons, are the basic units of artificial neural networks. They are mathematical functions that can receive inputs, perform some computation, and produce an output. Each neuron has a set of weights and a bias that determine how the inputs are combined and transformed. The weights and bias are the parameters that the neuron can learn from data and adjust accordingly.

Each neuron also has an activation function that determines the output of the neuron based on the input. The activation function can be linear or nonlinear, depending on the type of problem and the desired output. Some common activation functions are sigmoid, tanh, ReLU, softmax, and more.

Here is a simple example of an artificial neuron that takes two inputs, x1 and x2, and produces an output, y. The neuron has weights w1 and w2, a bias b, and a sigmoid activation function.

```
# Define the sigmoid function
  def sigmoid(x):
    return 1 / (1 + np.exp(-x))

  # Define the neuron function
  def neuron(x1, x2, w1, w2, b):
    # Compute the weighted sum of inputs and bias
    z = w1 * x1 + w2 * x2 + b
    # Apply the activation function
    y = sigmoid(z)
    # Return the output
    return y
```

What are the layers of artificial neural networks?

The layers of artificial neural networks are the groups of neurons that are connected and perform a specific function. There are three types of layers in artificial neural networks: input layer, hidden layer, and output layer.

The input layer is the first layer of the network that receives the data and passes it to the next layer. The input layer does not perform any computation, but simply acts as a placeholder for the data.

The hidden layer is the layer that is between the input and output layers. The hidden layer performs the main computation and learning of the network. The hidden layer can have one or more sub-layers, depending on the complexity and depth of the network. The hidden layer can also have different types of neurons and activation functions, depending on the type of problem and the desired output.

The output layer is the last layer of the network that produces the final output of the network. The output layer can have one or more neurons, depending on the type of problem and the desired output. The output layer can also have different types of neurons and activation functions, depending on the type of problem and the desired output.

Here is a simple example of an artificial neural network that has one input layer, one hidden layer, and one output layer. The input layer has two neurons, the hidden layer has three neurons, and the output layer has one neuron. The network can perform a binary classification task, such as predicting whether a customer will buy a product or not.

```
# Import numpy for numerical computation
  import numpy as np

  # Define the sigmoid function
  def sigmoid(x):
    return 1 / (1 + np.exp(-x))

  # Define the network function
  def network(x1, x2):
    # Define the weights and biases of the network
    w1 = 0.1
    w2 = 0.2
    w3 = 0.3
    w4 = 0.4
    w5 = 0.5
    w6 = 0.6
    w7 = 0.7
    w8 = 0.8
    w9 = 0.9
    b1 = 0.1
```

```
b2 = 0.2
b3 = 0.3
b4 = 0.4

# Compute the output of the input layer
i1 = x1
i2 = x2

# Compute the output of the hidden layer
h1 = sigmoid(w1 * i1 + w2 * i2 + b1)
h2 = sigmoid(w3 * i1 + w4 * i2 + b2)
h3 = sigmoid(w5 * i1 + w6 * i2 + b3)

# Compute the output of the output layer
y = sigmoid(w7 * h1 + w8 * h2 + w9 * h3 + b4)

# Return the output
return y
```

In summary, artificial neural networks are a type of machine learning technique that can learn from data and perform various tasks. Artificial neural networks are composed of layers of artificial neurons that can learn from data and perform various tasks. Artificial neurons are mathematical functions that can receive inputs, perform some computation, and produce an output. The layers of artificial neural networks are the groups of neurons that are connected and perform a specific function.

In the next section, you will learn how artificial neural networks work and how they can learn from data.

3. How do Artificial Neural Networks Work?

In this section, you will learn how artificial neural networks work and how they can learn from data. You will also learn some basic concepts and techniques related to artificial neural networks, such as forward propagation, backward propagation, loss function, gradient descent, and more.

How do artificial neural networks work?

Artificial neural networks work by passing the data through the layers of neurons and producing an output. The output of each neuron depends on the inputs, the weights, the

bias, and the activation function. The output of each layer is the input of the next layer, until the final output is reached.

This process of passing the data from the input layer to the output layer is called forward propagation. Forward propagation is the main computation and prediction phase of the network. Forward propagation can be represented by a mathematical equation that relates the input and output of the network.

For example, suppose we have a simple network that has one input layer with two neurons, one hidden layer with three neurons, and one output layer with one neuron. The network can perform a binary classification task, such as predicting whether a customer will buy a product or not. The network function can be written as:

```
# Define the network function
  def network(x1, x2):
    # Define the weights and biases of the network
    w1 = 0.1
    w2 = 0.2
    w3 = 0.3
    w4 = 0.4
    w5 = 0.5
    w6 = 0.6
    w7 = 0.7
    w8 = 0.8
    w9 = 0.9
    b1 = 0.1
    b2 = 0.2
    b3 = 0.3
    b4 = 0.4

    # Compute the output of the input layer
    i1 = x1
    i2 = x2

    # Compute the output of the hidden layer
    h1 = sigmoid(w1 * i1 + w2 * i2 + b1)
    h2 = sigmoid(w3 * i1 + w4 * i2 + b2)
    h3 = sigmoid(w5 * i1 + w6 * i2 + b3)
```

```
    # Compute the output of the output layer
    y = sigmoid(w7 * h1 + w8 * h2 + w9 * h3 + b4)

    # Return the output
    return y
```

However, the network function is not fixed, but can be adjusted and improved based on the data. The network can learn from the data by comparing the output of the network with the actual output of the data, and updating the weights and bias accordingly. This process of updating the weights and bias based on the data is called learning or training.

How do artificial neural networks learn from data?

Artificial neural networks learn from data by minimizing the difference between the output of the network and the actual output of the data. This difference is called the error or the loss, and it measures how well the network performs on the data. The goal of learning is to find the optimal values of the weights and bias that minimize the loss.

The loss can be calculated by using a loss function, which is a mathematical function that quantifies the error or the loss. There are different types of loss functions, depending on the type of problem and the desired output. Some common loss functions are mean squared error, cross entropy, hinge loss, and more.

For example, suppose we have a binary classification problem, such as predicting whether a customer will buy a product or not. The actual output of the data is either 0 or 1, representing no or yes. The output of the network is a probability between 0 and 1, representing the confidence of the prediction. A common loss function for this type of problem is the cross entropy loss, which is defined as:

```
# Define the cross entropy loss function
  def cross_entropy_loss(y_true, y_pred):
    # Compute the loss for each data point
    loss = - (y_true * np.log(y_pred) + (1 - y_true) * np.log(1 - y_pred))
    # Return the average loss over all data points
    return np.mean(loss)
```

The cross entropy loss function penalizes the network for making wrong predictions, and rewards the network for making correct predictions. The lower the loss, the better the network performs on the data.

However, the loss is not a simple function of the weights and bias, but a complex function of the inputs, the outputs, the activation functions, and the network structure. Therefore,

finding the optimal values of the weights and bias that minimize the loss is not a trivial task, but a challenging optimization problem.

One of the most popular and effective methods to solve this optimization problem is the gradient descent method. The gradient descent method is an iterative algorithm that updates the weights and bias by moving in the opposite direction of the gradient of the loss function. The gradient of the loss function is the vector of partial derivatives of the loss function with respect to the weights and bias, and it indicates the direction and magnitude of the steepest increase of the loss function.

By moving in the opposite direction of the gradient, the gradient descent method can find the local minimum of the loss function, which corresponds to the optimal values of the weights and bias. The size of the step that the gradient descent method takes in each iteration is called the learning rate, and it determines how fast or slow the algorithm converges to the optimal solution. The learning rate is a hyperparameter that can be tuned by the user.

The gradient descent method can be applied to the whole data set at once, which is called batch gradient descent, or to a subset of the data set at a time, which is called mini-batch or stochastic gradient descent. The latter methods are more efficient and effective, especially for large data sets, as they can reduce the computational cost and avoid getting stuck in local minima.

This process of updating the weights and bias by using the gradient descent method is called backward propagation. Backward propagation is the main learning and training phase of the network. Backward propagation can be implemented by using the chain rule of calculus, which allows to compute the gradient of the loss function with respect to the weights and bias by propagating the errors from the output layer to the input layer.

Here is a simple example of how to implement the backward propagation for a simple network that has one input layer with two neurons, one hidden layer with three neurons, and one output layer with one neuron. The network can perform a binary classification task, such as predicting whether a customer will buy a product or not. The network uses the sigmoid activation function and the cross entropy loss function.

```
# Import numpy for numerical computation
  import numpy as np

  # Define the sigmoid function and its derivative
  def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

```
    def sigmoid_derivative(x):
        return sigmoid(x) * (1 - sigmoid(x))

    # Define the cross entropy loss function and its derivative
    def cross_entropy_loss(y_true, y_pred):
        # Compute the loss for each data point
        loss = - (y_true * np.log(y_pred) + (1 - y_true) * np.log(1 - y_pred))
        # Return the average loss over all data points
        return np.mean(loss)

    def cross_entropy_loss_derivative(y_true, y_pred):
        # Compute the derivative of the loss for each data point
loss_derivative = - (y_true / y_pred - (1 - y_true) / (1 - y_pred))
        # Return the average derivative over all data points
        return np.mean(loss_derivative)

    # Define the network function
    def network(x1, x2, w1, w2, w3, w4, w5, w6, w7, w8, w9, b1, b2, b3, b4):
        # Compute the output of the input layer
        i1 = x1
        i2 = x2

        # Compute the output of the hidden layer
        h1 = sigmoid(w1 * i1 + w2 * i2 + b1)
        h2 = sigmoid(w3 * i1 + w4 * i2 + b2)
        h3 = sigmoid(w5 * i1 + w6 * i2 + b3)

        # Compute the output of the output layer
        y = sigmoid(w7 * h1 + w8 * h2 + w9 * h3 + b4)

        # Return the output and the intermediate values
        return y, i1, i2, h1, h2, h3

    # Define the backward propagation function
    def backward_propagation(x1, x2, y_true, w1, w2, w3, w4, w5, w6, w7, w8, w9, b1, b2,
b3, b4, learning_rate):
        # Forward pass
```

```
y_pred, i1, i2, h1, h2, h3 = network(x1, x2, w1, w2, w3, w4, w5, w6, w7, w8, w9, b1, b2, b3, b4)

    # Compute the derivative of the loss with respect to the output
dL_dy = cross_entropy_loss_derivative(y_true, y_pred)

    # Compute the derivative of the output with respect to the input of the output layer
    dy_dz4 = sigmoid_derivative(w7 * h1 + w8 * h2 + w9 * h3 + b4)

    # Compute the derivative of the output with respect to the weights and biases of the output layer
    dz4_dw7 = h1
    dz4_dw8 = h2
    dz4_dw9 = h3
    dz4_db4 = 1

    # Compute the derivative of the output of the hidden layer with respect to the input of the hidden layer
    dh_dz3 = sigmoid_derivative(w5 * i1 + w6 * i2 + b3)
    dh_dz2 = sigmoid_derivative(w3 * i1 + w4 * i2 + b2)
    dh_dz1 = sigmoid_derivative(w1 * i1 + w2 * i2 + b1)

    # Compute the derivative of the output of the hidden layer with respect to the weights and biases of the hidden layer
    dz3_dw5 = i1
    dz3_dw6 = i2
    dz3_db3 = 1
    dz2_dw3 = i1
    dz2_dw4 = i2
    dz2_db2 = 1
    dz1_dw1 = i1
    dz1_dw2 = i2
    dz1_db1 = 1

    # Compute the derivative of the loss with respect to the weights and biases of the output layer
    dL_dw7 = dL_dy * dy_dz4 * dz4_dw7
    dL_dw8 = dL_dy * dy_dz4 * dz4_dw8
```

```
    dL_dw9 = dL_dy * dy_dz4 * dz4_dw9
    dL_db4 = dL_dy * dy_dz4 * dz4_db4

    # Compute the derivative of the loss with respect to the weights and biases of the hidden layer
    dL_dw5 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz3 * dz3_dw5
    dL_dw6 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz3 * dz3_dw6
    dL_db3 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz3 * dz3_db3
    dL_dw3 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz2 * dz2_dw3
    dL_dw4 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz2 * dz2_dw4
    dL_db2 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz2 * dz2_db2
    dL_dw1 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz1 * dz1_dw1
    dL_dw2 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz1 * dz1_dw2
    dL_db1 = dL_dy * dy_dz4 * dz4_dw9 * dh_dz1 * dz1_db1

    # Update the weights and biases of the output layer
    w7 -= learning_rate * dL_dw7
    w8 -= learning_rate * dL_dw8
    w9 -= learning_rate * dL_dw9
    b4 -= learning_rate * dL_db4

    # Update the weights and biases of the hidden layer
    w5 -= learning_rate * dL_dw5
    w6 -= learning_rate * dL_dw6
    b3 -= learning_rate * dL_db3
    w3 -= learning_rate * dL_dw3
    w4 -= learning_rate * dL_dw4
    b2 -= learning_rate * dL_db2
    w1 -= learning_rate * dL_dw1
    w2 -= learning_rate * dL_dw2
```

4. Types of Artificial Neural Networks

In this section, you will learn about different types of artificial neural networks and their applications. You will also learn some advantages and disadvantages of each type of network.

What are the types of artificial neural networks?

Artificial neural networks can be classified into different types based on their structure, function, and learning method. There are many types of artificial neural networks, but some of the most common and popular ones are:

- Feedforward neural networks
- Recurrent neural networks
- Convolutional neural networks
- Generative adversarial networks

What are feedforward neural networks?

Feedforward neural networks, or FNN for short, are the simplest and most basic type of artificial neural networks. They are composed of layers of neurons that are connected in a forward direction, meaning that the data flows from the input layer to the output layer without any feedback loops or cycles. Feedforward neural networks can perform various tasks, such as classification, regression, clustering, and more.

Some advantages of feedforward neural networks are:

- They are easy to understand and implement
- They can learn complex nonlinear functions
- They can be trained efficiently using gradient descent and backpropagation

Some disadvantages of feedforward neural networks are:

- They have a fixed structure and size, which limits their flexibility and scalability
- They cannot handle sequential or temporal data, such as text, speech, or video
- They are prone to overfitting, which means that they memorize the training data and fail to generalize to new data

What are recurrent neural networks?

Recurrent neural networks, or RNN for short, are a type of artificial neural networks that can handle sequential or temporal data, such as text, speech, or video. They are composed of layers of neurons that are connected in a forward and backward direction, meaning that the data flows from the input layer to the output layer and also from the output layer to the input layer. Recurrent neural networks can perform various tasks, such as natural language processing, speech recognition, machine translation, and more.

Some advantages of recurrent neural networks are:

- They can process sequential or temporal data of variable length
- They can capture the long-term dependencies and context of the data
- They can generate new sequences of data, such as text or speech

Some disadvantages of recurrent neural networks are:

- They are difficult to understand and implement
- They are computationally expensive and slow to train
- They suffer from the vanishing or exploding gradient problem, which means that the gradient of the loss function becomes either too small or too large to update the weights and bias effectively

What are convolutional neural networks?

Convolutional neural networks, or CNN for short, are a type of artificial neural networks that can handle spatial data, such as images, video, or audio. They are composed of layers of neurons that are connected in a local and sparse way, meaning that each neuron is only connected to a small region of the previous layer. Convolutional neural networks can perform various tasks, such as image recognition, face detection, object detection, and more.

Some advantages of convolutional neural networks are:

- They can process spatial data of variable size
- They can extract the features and patterns of the data automatically
- They can reduce the number of parameters and computations by using local and sparse connections

Some disadvantages of convolutional neural networks are:

- They are complex and require a lot of data and resources to train
- They are sensitive to the hyperparameters and the architecture of the network
- They are not very good at handling sequential or temporal data, such as text or speech

What are generative adversarial networks?

Generative adversarial networks, or GAN for short, are a type of artificial neural networks that can generate new data that resembles the original data, such as images, video, or audio. They are composed of two networks that compete with each other: a generator network that tries to create fake data, and a discriminator network that tries to distinguish between real and fake data. Generative adversarial networks can perform various tasks, such as image synthesis, image editing, image super-resolution, and more.

Some advantages of generative adversarial networks are:

- They can create realistic and high-quality data
- They can learn from unlabeled or unsupervised data
- They can be applied to various domains and modalities of data

Some disadvantages of generative adversarial networks are:

- They are very hard to train and stabilize
- They are prone to mode collapse, which means that the generator network produces the same or similar data
- They are difficult to evaluate and measure their performance

In summary, artificial neural networks can be classified into different types based on their structure, function, and learning method. Each type of network has its own advantages and disadvantages, and can be used for different tasks and applications.

In the next section, you will learn about some applications of artificial neural networks and how they can solve real-world problems.

5. Applications of Artificial Neural Networks

In this section, you will learn about some applications of artificial neural networks and how they can solve real-world problems. You will also see some examples of artificial neural networks in action and how they can produce amazing results.

What are some applications of artificial neural networks?

Artificial neural networks can be applied to various domains and modalities of data, such as text, speech, image, video, audio, and more. They can perform various tasks, such as classification, regression, clustering, anomaly detection, generation, translation, and more. Here are some examples of applications of artificial neural networks:

- Image recognition: Artificial neural networks can recognize and classify objects, faces, scenes, and more in images. For example, [Google Photos] uses artificial neural networks to organize and search your photos based on the content, such as people, places, things, and events.
- Speech recognition: Artificial neural networks can recognize and transcribe speech into text. For example, [Microsoft Cortana] uses artificial neural networks to understand and respond to your voice commands and queries.
- Natural language processing: Artificial neural networks can process and analyze natural language, such as text or speech. They can perform various tasks, such as sentiment analysis, text summarization, question answering, and more. For example, [Microsoft Bing] uses artificial neural networks to provide relevant and accurate search results and suggestions based on your queries.
- Machine translation: Artificial neural networks can translate text or speech from one language to another. For example, [Microsoft Translator] uses artificial neural networks to translate text, speech, and images across more than 70 languages.
- Image synthesis: Artificial neural networks can generate new images that resemble the original images, such as faces, animals, landscapes, and more. For example, [This Person Does Not Exist] uses artificial neural networks to create realistic and high-quality faces of people that do not exist.
- Image editing: Artificial neural networks can edit images in various ways, such as enhancing, restoring, colorizing, stylizing, and more. For example, [DeepArt] uses artificial neural networks to transform your photos into artworks in the style of famous artists.
- Image super-resolution: Artificial neural networks can increase the resolution and quality of images, such as photos, videos, and medical images. For example, [Let's Enhance] uses artificial neural networks to upscale and improve your images without losing details.
- Self-driving cars: Artificial neural networks can control and navigate self-driving cars, such as detecting and avoiding obstacles, following traffic rules, and planning routes. For example, [Tesla Autopilot] uses artificial neural networks to provide a full self-driving capability for your car.
- Reinforcement learning: Artificial neural networks can learn from their own actions and experiences, and optimize their behavior to achieve a goal. For example,

[AlphaGo] uses artificial neural networks to play and master the game of Go, one of the most complex and challenging board games in the world.

These are just some of the many applications of artificial neural networks, and there are more to be discovered and developed in the future. Artificial neural networks have the potential to revolutionize various fields and industries, and improve our lives and society.

In the next section, you will learn about some challenges and limitations of artificial neural networks and how they can be overcome or mitigated.

6. Challenges and Limitations of Artificial Neural Networks

In this section, you will learn about some challenges and limitations of artificial neural networks and how they can be overcome or mitigated. You will also learn some best practices and tips for designing and training artificial neural networks.

What are some challenges and limitations of artificial neural networks?

Artificial neural networks are powerful and versatile machine learning techniques, but they are not perfect and have some drawbacks and difficulties. Some of the common challenges and limitations of artificial neural networks are:

- Data quality and quantity: Artificial neural networks require a large amount of high-quality and relevant data to learn and perform well. However, data can be scarce, noisy, incomplete, imbalanced, or biased, which can affect the performance and reliability of the network. Therefore, data preprocessing, cleaning, augmentation, and balancing are essential steps before training the network.
- Hyperparameter tuning: Artificial neural networks have many hyperparameters that can affect the performance and behavior of the network, such as the number and size of layers, the type and number of neurons, the activation function, the learning rate, the batch size, the regularization, and more. However, finding the optimal values of these hyperparameters is not easy and often requires trial and error, grid search, or random search. Therefore, hyperparameter tuning is a time-consuming and tedious process that can improve the network.
- Overfitting and underfitting: Artificial neural networks can suffer from overfitting or underfitting, which means that they either memorize the training data and fail to generalize to new data, or fail to capture the complexity and patterns of the data. Overfitting and underfitting can be caused by various factors, such as the data quality and quantity, the network structure and size, the learning rate, the regularization,

and more. Therefore, overfitting and underfitting can be prevented or reduced by using various techniques, such as cross-validation, dropout, early stopping, and more.

- Interpretability and explainability: Artificial neural networks are often considered as black boxes, which means that they can produce accurate and impressive results, but they cannot explain how and why they produce those results. This can be problematic for some applications that require transparency, accountability, and trust, such as medical diagnosis, legal decision, and ethical judgment. Therefore, interpretability and explainability are important aspects that can enhance the understanding and trust of the network.
- Adversarial attacks and robustness: Artificial neural networks can be vulnerable to adversarial attacks, which are malicious inputs that are designed to fool or mislead the network. For example, adding a small and imperceptible noise to an image can cause the network to misclassify the image. Adversarial attacks can pose serious threats and risks for some applications that require security, safety, and reliability, such as self-driving cars, face recognition, and fraud detection. Therefore, adversarial attacks and robustness are critical issues that can affect the performance and security of the network.