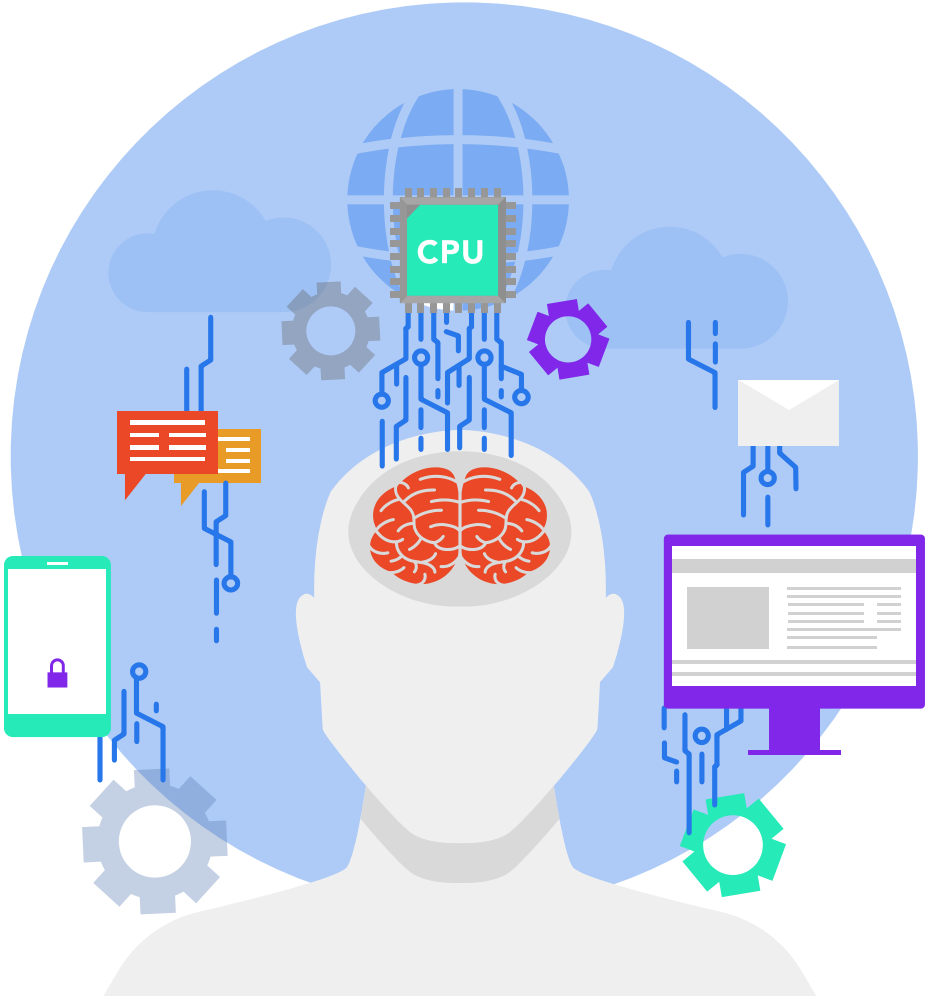
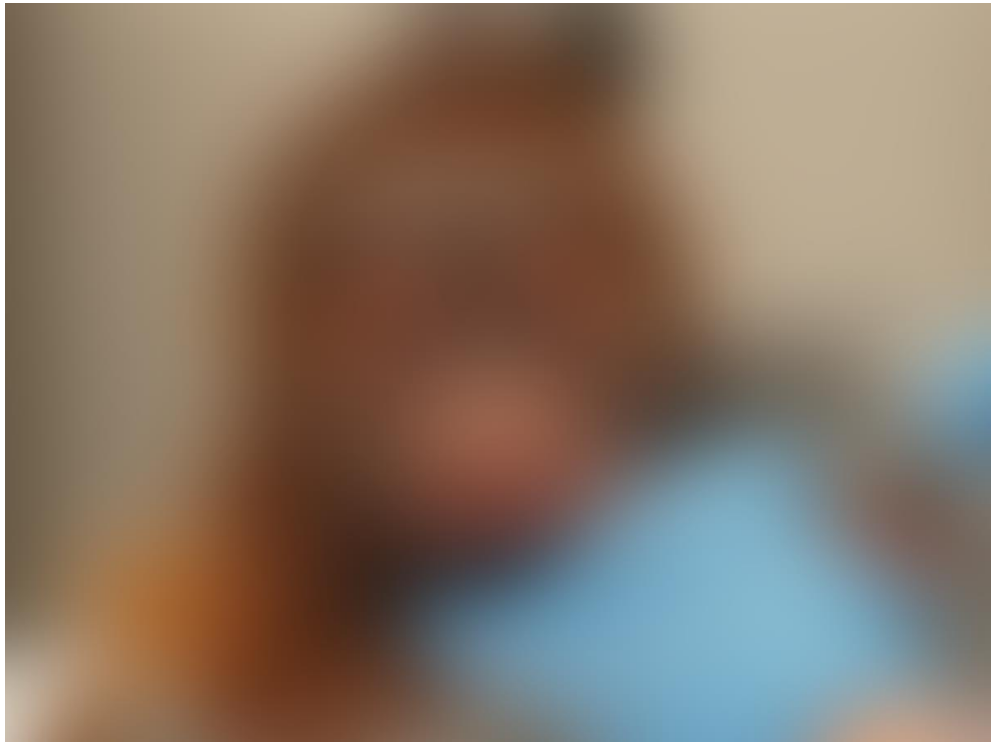


Artificial Neural Networks

Dr. Hemraj S. L.
Cryptologist.

Lets understand Preprocessing



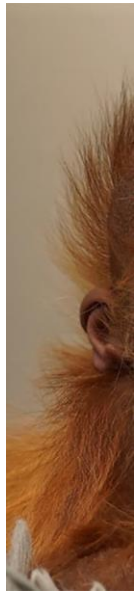
Lets understand Preprocessing

Lets understand Preprocessing

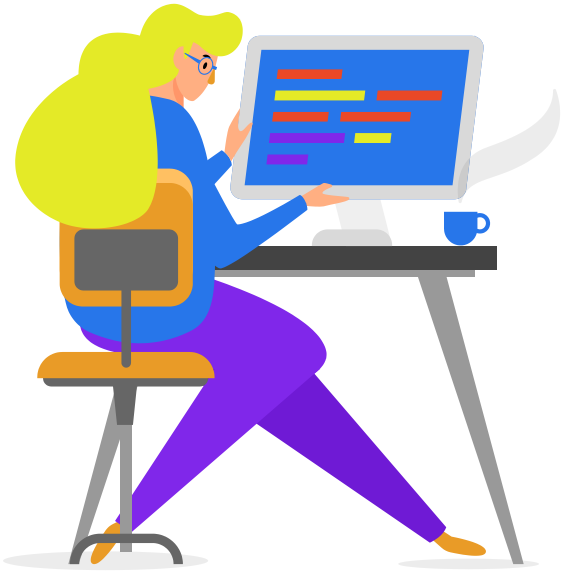
Lets understand Preprocessing

Lets understand Preprocessing

Lets understand Preprocessing



Artificial Neural Networks



01 ANN History

02 Taxonomy of neural network

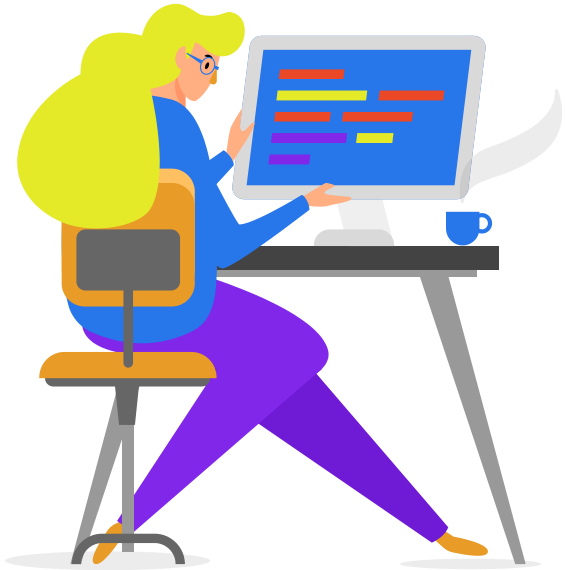
03 Characteristics and applications of ANN

04 definition of ANN

05 Difference between ANN and human brain

06 Single layer network

Artificial Neural Networks



07 Perceptron training algorithm

08 Widrow Hoff Learning Algorithm

09 Delta Rule Learning Algorithm

10 ADALINE & MADALINE Linear Neuron

11 AI v/s ANN

12 Application of ANN

ANN History

1943 01

Neural Network

Warren McCulloch modeled a simple neural network using electrical circuits in order to describe how neurons in the brain might work.

1949 02

Organization Behaviour

activation of one neuron by another increases its strength each time they are used – Donald Hebb's

1958 03

Perceptron

A learning method for McCulloch and Pitts neuron model named Perceptron was invented by Rosenblatt.

1960 04

Adaline & Madaline

Bernard Widrow and Marcian Hoff developed models called "ADALINE" and "MADALINE."

1961 05

Back Propagation

Rosenblatt proposed the "backpropagation" scheme for multilayer networks

1969 06

Multi-layer perceptron

Multilayer perceptron MLP was invented by Minsky and Papert

1976 07

Resonance Theory

Stephen Grossberg and Gail Carpenter developed Adaptive resonance theory.

ANN History

1982 08

Hopfield Net

The major development was Hopfield's Energy approach.

1985 09

Boltzmann

Boltzmann machine was developed by Ackley, Hinton, and Sejnowski.

1986 10

Delta Rule

Rumelhart, Hinton, and Williams introduced Generalised Delta Rule.

1988 11

Binary A.M.

Kosko developed Binary Associative Memory BAM and also gave the concept of Fuzzy Logic in ANN.

1989 12

Back Propagation - Redefined

neural networks were used to recognize handwritten digits based on a technique proposed in 1989 by Yann LeCun

2010 13

Rise of Deep learning

better optimization algorithms, helped to train deeper neural networks efficiently.

2012 14

Visual Recognition

Using GPU to trained neural network for large scale visual recognition

ANN History

2016 15

Power of AI

AlphaGo — an AI developed by DeepMind — beat the Go world champion (with 10^170 possible combinations in chess)

2016 16

Google Acquired DeepMind

Google acquired DeepMind for more than $500 million.

2020 17

Journal Article

GPT-3 is now writing journal articles that are often barely recognizable from human ones

2021 18

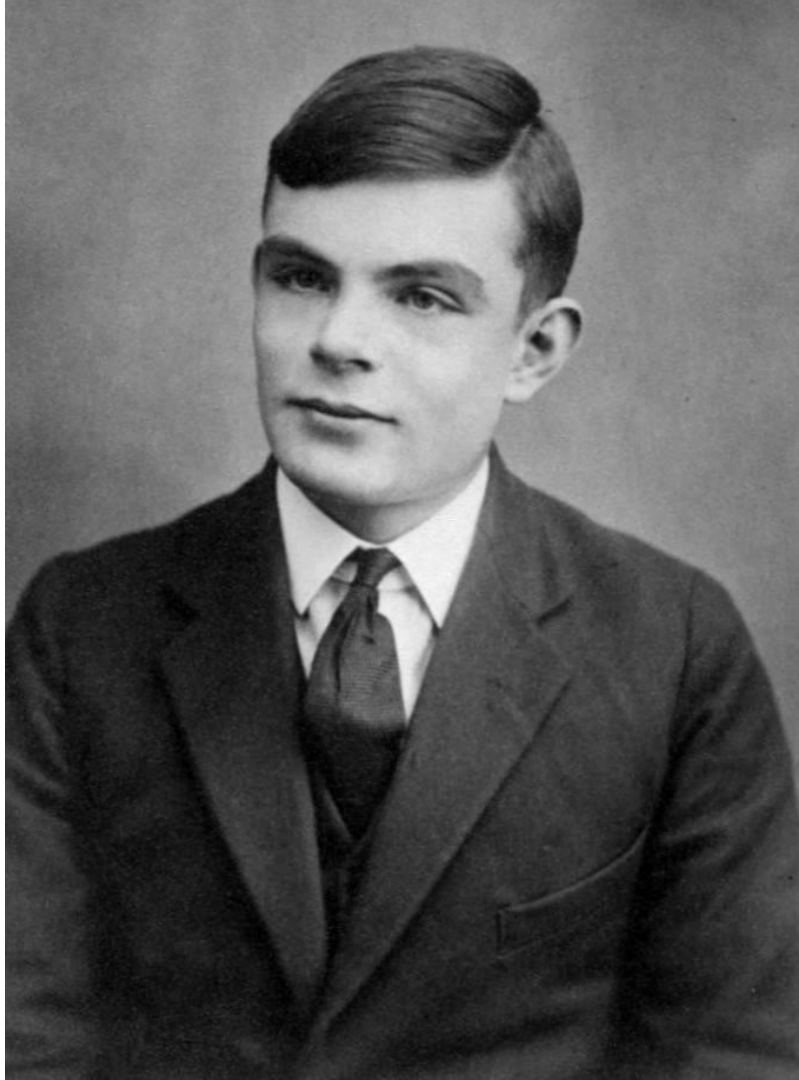
OpenAI

OpenAI — funded by Elon Musk - $1 billion investmen

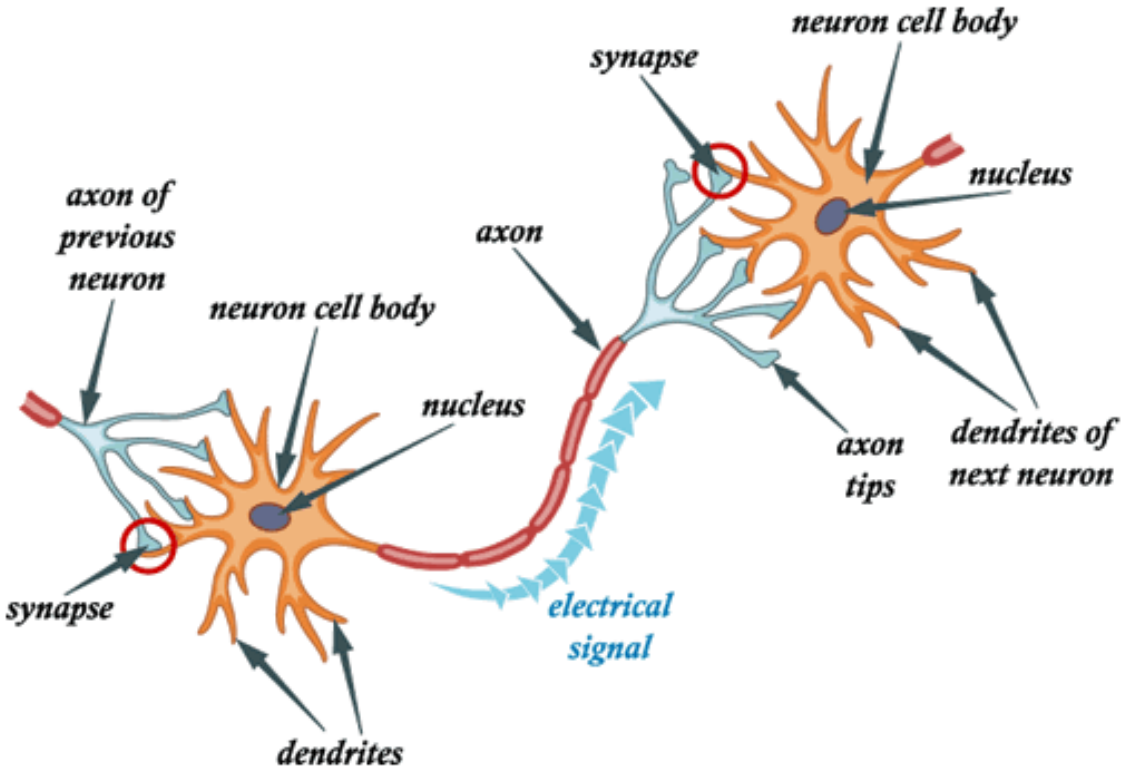
Alan Turing

Turing was a founding father of artificial intelligence and of modern cognitive science, and he was a leading early exponent of the hypothesis that the human brain is in large part a digital computing machine

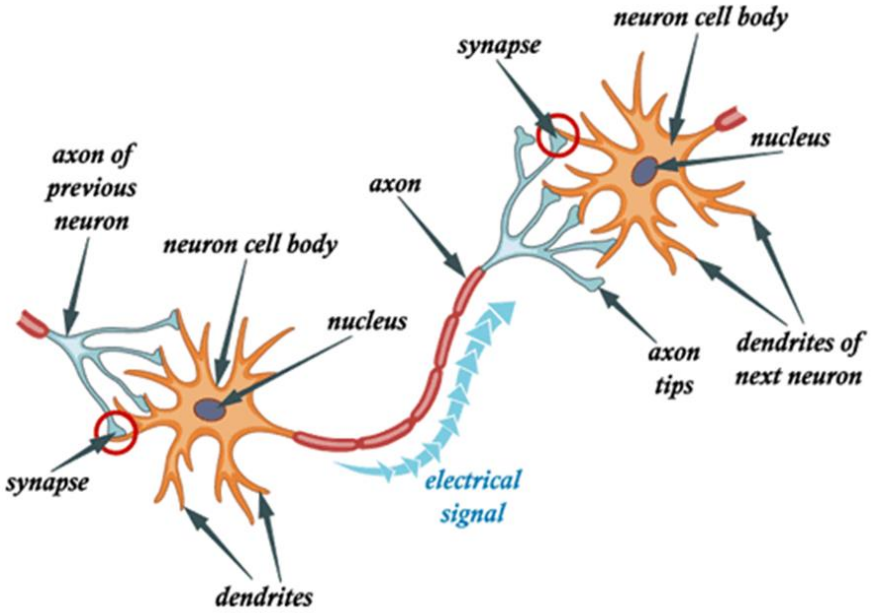
Alan Turing (1912–1946, famous English computer scientist) proposed the “Turing test” to assess if a machine could imitate human language well enough to be confused for a human.



Neuron: Basic unit of nervous system



Neuron: Basic unit of nervous system



A schematic of a biological neuron consists of variousparts in it : *dendrite, soma, axon and synapse*.

Dendrite : A bush of very thin fibre.

Axon : A long cylindrical fibre.

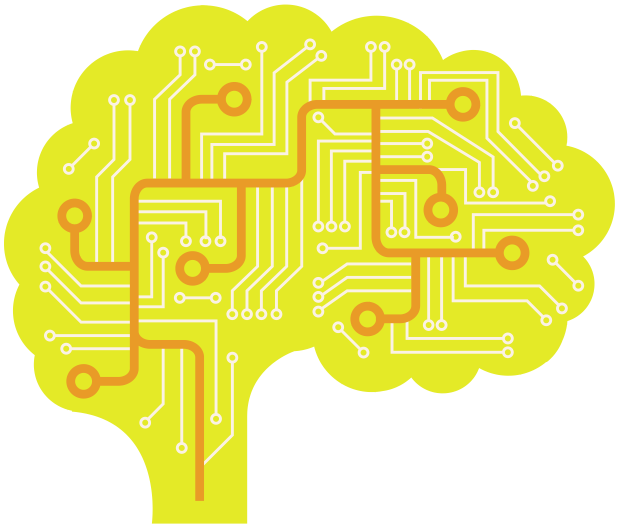
Soma : It is also called a cell body, and just like as a nucleus of cell.

Synapse : It is a junction where axon makes contact with the dendrites of neighboring dendrites.

Neuron and its working

Biological Neural network

A nerve cell ***neuron*** is a special biological cell that processes information



There is a chemical in each neuron called neurotransmitter.

A signal (also called sense) is transmitted across neurons by this chemical.

That is, all inputs from other neuron arrive to a neurons through dendrites.
These signals are accumulated at the synapse of the neuron and then serve as the output to be transmitted through the neuron.

An action may produce an electrical impulse, which usually lasts for about a millisecond.

Note that this pulse generated due to an incoming signal and all signal may not produce pulses in axon unless it crosses a **threshold value**.
Also, note that an action signal in axon of a neuron is commutative signals arrive at dendrites which summed up at **soma**.

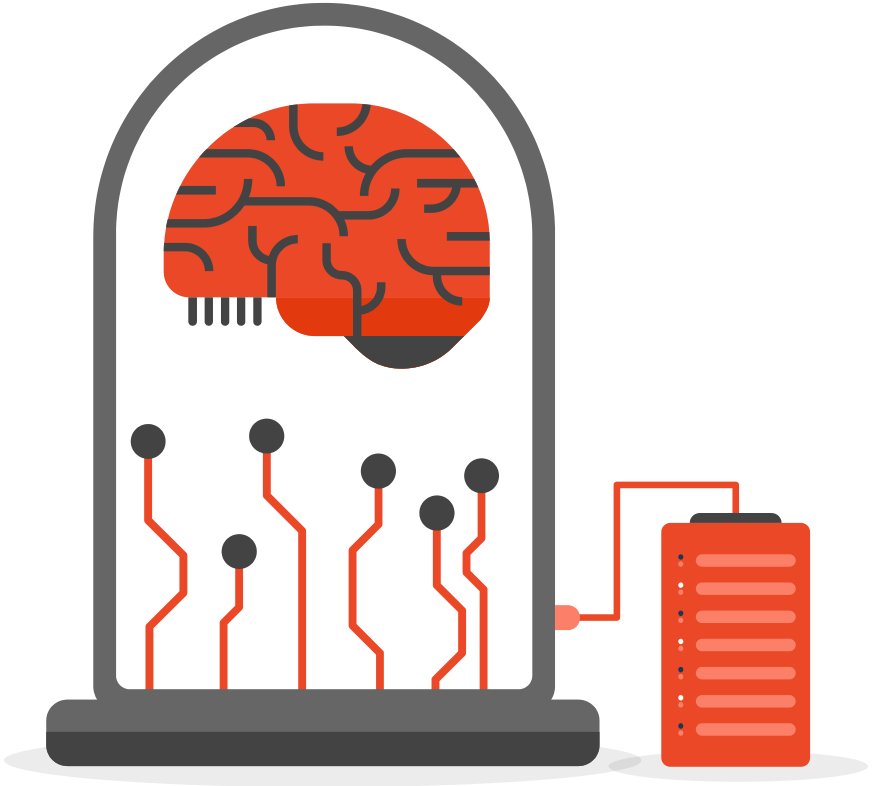
Artificial Neural Network

Artificial neural networks are one of the main tools used in machine learning. As the *“neural”* part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

While neural networks (also called “*perceptron's*”), it is only in the last several decades where they have become a major part of artificial intelligence. This is due to the arrival of a technique called “backpropagation,” which allows networks to adjust their hidden layers of neurons in situations where the outcome doesn’t match what the creator is hoping for. Example: a network designed to recognize dogs.

Another important advance has been the arrival of deep learning neural networks, in which different layers of a multilayer network extract different features until it can recognize what it is looking for.

Artificial Neural Network

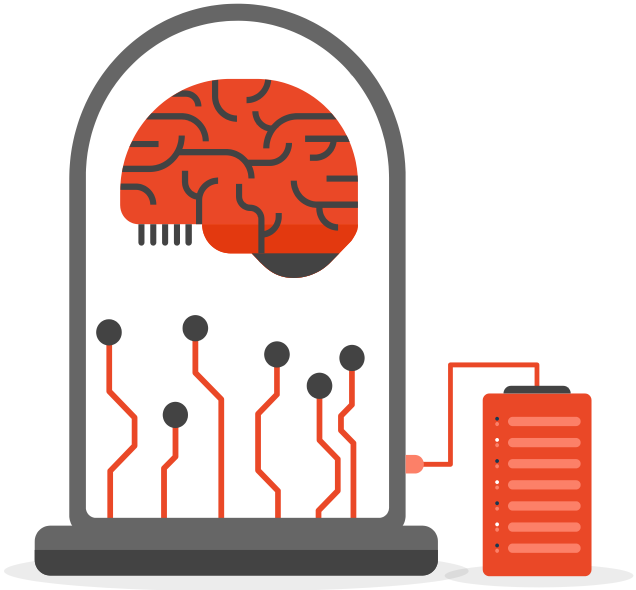


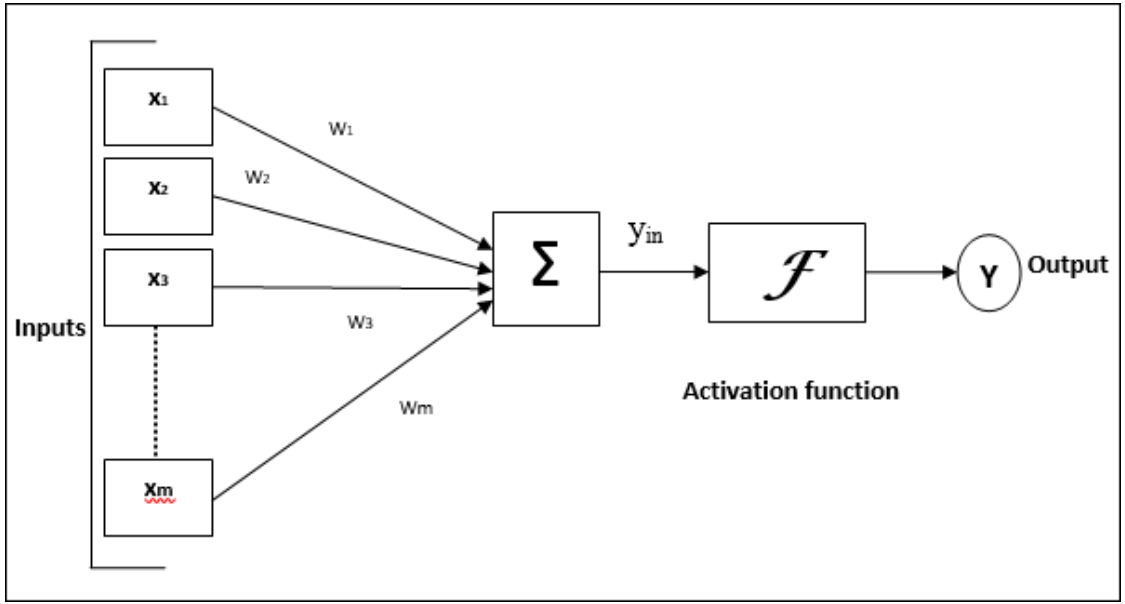
In fact, the human brain is a highly complex structure viewed as a massive, highly interconnected network of simple processing elements called **neurons**.

Artificial neural networks (ANNs) or simply we refer it as neural network (NNs), which are simplified models (i.e. imitations) of the biological nervous system, and obviously, therefore, have been motivated by the kind of computing performed by the human brain.

The behavior of a biolgical neural network can be captured by a simple model called artificial neural network.

Artificial Neural Network



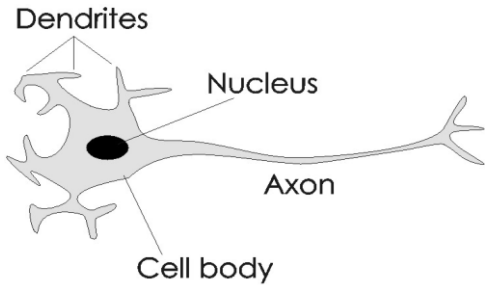


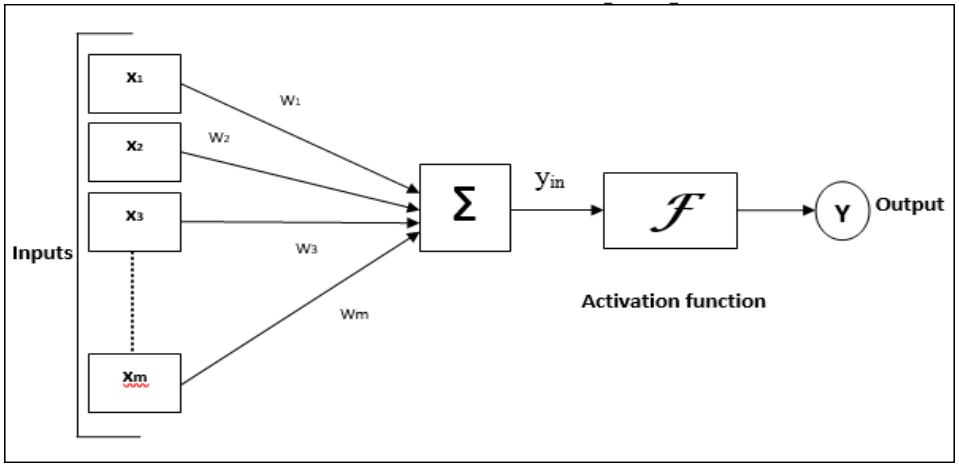
To generate the final output y, the sum is passed to a filter Φ called Transfer/Activation function, which releases the output.

Here, $x_1, x_2, \cdots, x_n$ are the n inputs to the artificial neuron.

$w_1, w_2, \cdots, w_n$ are weights attached to the input links.

Artificial Neural Network





Here, $x_1, x_2, \cdots, x_n$ are the n inputs to the artificial neuron.

$w_1, w_2, \cdots, w_n$ are weights attached to the input links.

We may note that a neuron is a part of an interconnected network of nervous system and serves the following.

Compute input signals

Transportation of signals (at a very high speed)

Storage of information

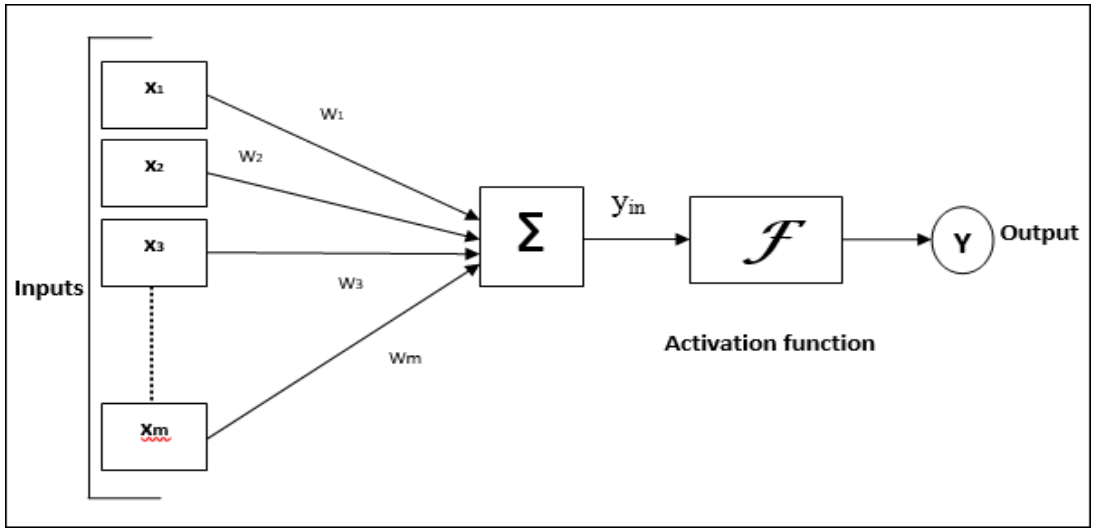
Perception, automatic training and learning

We also can see the analogy between the biological neuron and artificial neuron. Truly, every component of the model (i.e. artificial neuron) bears a direct analogy to that of a biological neuron. It is this model which forms the basis of neural network (i.e. artificial neural network).

Perceptron

Developed by Frank Rosenblatt by using McCulloch and Pitts model, perceptron is the basic operational unit of artificial neural networks. It employs supervised learning rule and is able to classify the data into classes.

Operational characteristics of the perceptron: It consists of a single neuron with an arbitrary number of inputs along with adjustable weights, but the output of the neuron is 1 or 0 depending upon the threshold. It also consists of a bias whose weight is always 1. Following figure gives a schematic representation of the perceptron.



Perceptron

Perceptron thus has the following three basic elements −

•**Links** − It would have a set of connection links, which carries a weight including a bias always having weight 1.

•**Adder** − It adds the input after they are multiplied with their respective weights.

•**Activation function** − It limits the output of neuron. The most basic activation function is a *Heaviside step function* that has two possible outputs. This function returns 1, if the input is positive, and 0 for any negative input.

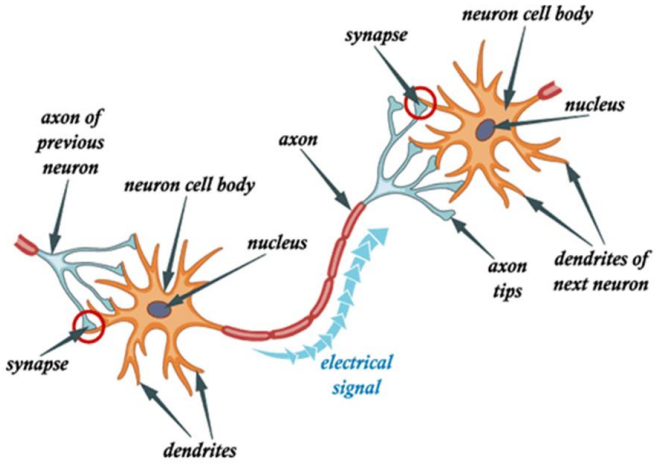
Artificial neural network

A very commonly known transfer function.

In this transfer function, sum (i.e. ***I***) is compared with a threshold value θ.

If the value of ***I*** is greater than θ, then the output is **1** else it is **0** (this is just like a simple linear filter).

Biological Neural Network	Artificial Neural Network
Soma	*Node*
Dendrites	*Input*
Synapse	*Weights or Interconnections*
Axon	*Output*



What exactly happens in BNN

A biological neuron receives all inputs through the dendrites, sums them and produces an output if the sum is greater than a threshold value.

The input signals are passed on to the cell body through the synapse, which may accelerate or retard an arriving signal.

It is this acceleration or retardation of the input signals that is modeled by the **weights**.

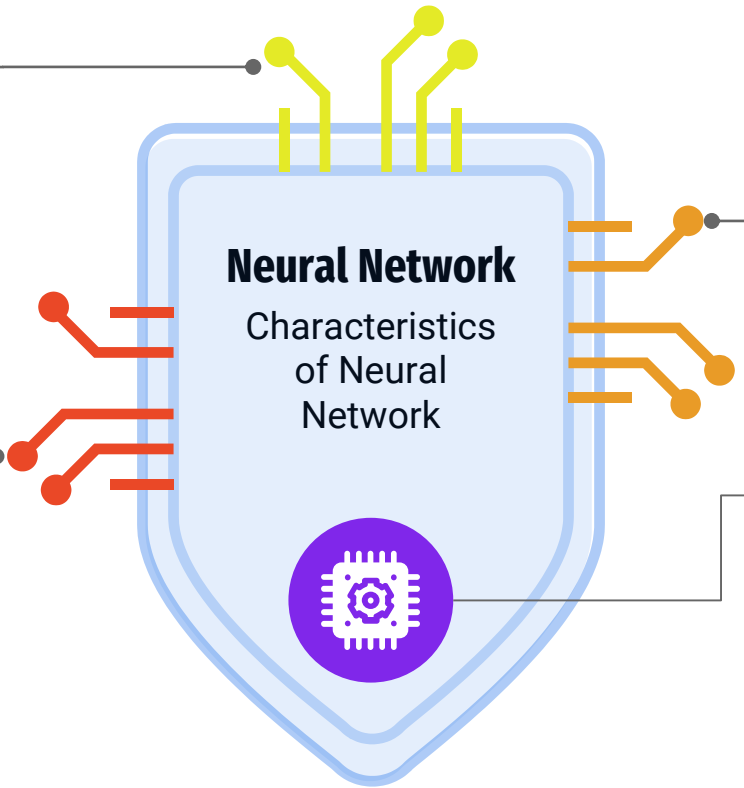
An effective synapse, which transmits a stronger signal will have a correspondingly larger weights while a weak synapse will have smaller weights.

Thus, weights here are multiplicative factors of the inputs to account for the strength of the synapse.

BNN vs ANN

Biological Neuron	Artificial Neuron
It is made of cells.	The cells correspond to neurons.
It has dendrites which are interconnections between cell body.	The connection weights correspond to dendrites.
Soma receives the input.	Soma is similar to net input weight.
The axon receives the signal.	The output of ANN corresponds to axon.

Characteristics of Neural Network



01

Non Linearity

The mechanism followed in ANN for the generation of the input signal is nonlinear.

02

Supervised Learning

The input and output are mapped and the ANN is trained with the training dataset.

03

Unsupervised Learning

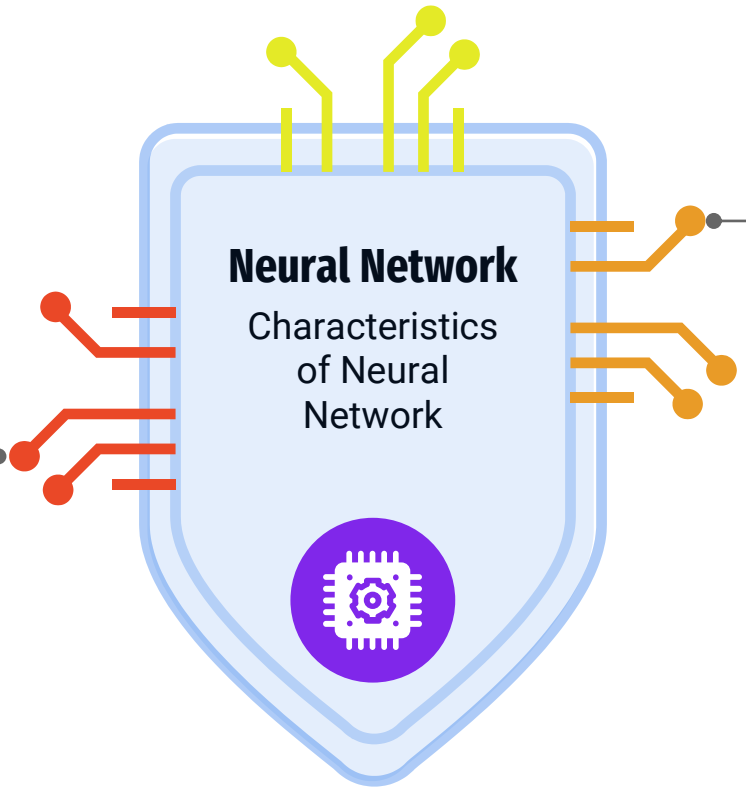
The target output is not given, so the ANN will learn on its own by discovering the features in the input patterns.

04

Adaptive Nature

The connection weights in the nodes of ANN are capable to adjust themselves to give the desired output

Characteristics of Neural Network



05

Biological Neuron Analogy

The ANN has a human brain-inspired structure and functionality.

06

Fault Tolerance

These networks are highly tolerant as the information is distributed in layers and computation occurs in real-time.

Structure of ANN

Artificial Neural Networks are processing elements either in the form of algorithms or hardware devices modeled after the neuronal structure of a human brain cerebral cortex.

These networks are also called as **Neural Networks**. The NN which is formed by using many layers which are interconnected are often called "***Multilayer Perceptron***".

The neurons in one layer are called "Nodes". These nodes have an "Activation function".

Structure of ANN

Three main layers in ANN

The hidden layers calculate the output based on the "***weights***" which is the "*sum of weighted synapse connections*". The hidden layers refine the input by removing redundant information and send the information to the next hidden layer for further processing.

01

Input Layer

The input patterns are fed to the input layers.

02

Hidden Layers

There can be one or more hidden layers where processing takes place

03

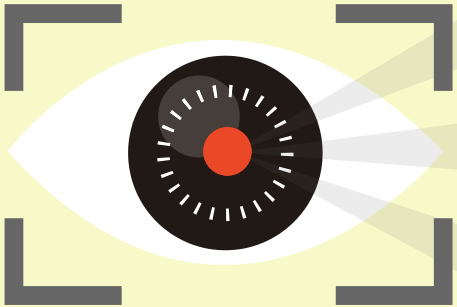
Output Layer

Output is taken from this layer

Taxonomy of Artificial Neural Network

Artificial neural networks (ANN) are adaptive models that can establish almost any relationship between data. It allow a mappings between a set of input and output vectors. ANNs are quite promising in solving problems where traditional models fail, especially for modeling complex phenomena which show a ***non-linear relationship.***

3 network categories in ANN



Signal Transfer — the input signal is transformed into an output signal

State Transition — Hopfield networks, and Boltzmann machines

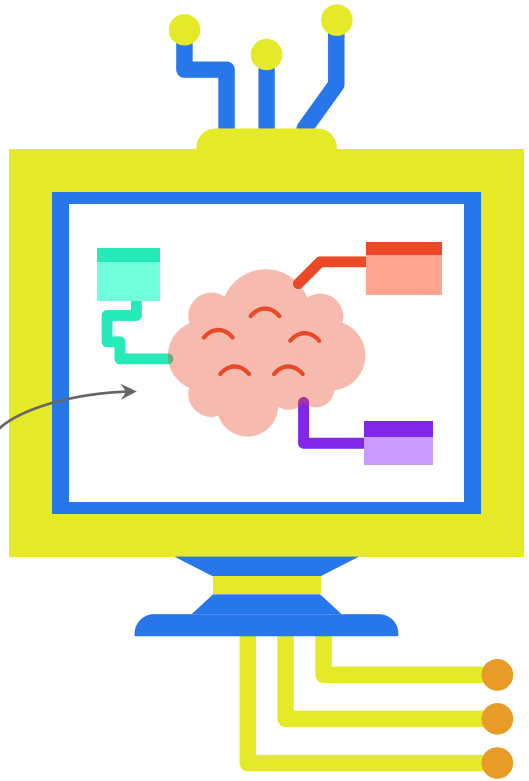
Competitive Learning — all the neurons of the network compete for the input signal. The neuron which "wins" gets the chance to move towards the input signal in n-dimensional space

Activation Function in ANN

Inputs

The input patterns are fed to the input layers. There is one input layer

- Input 1
- Input 2
- Input 3



Outputs

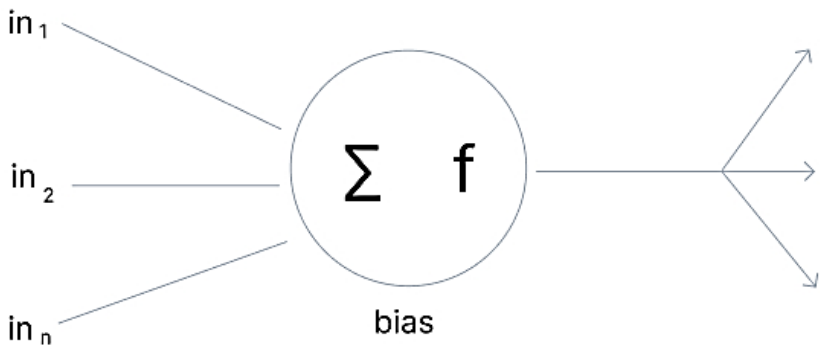
This hidden layer connects to the “output layer” where the output is shown

- Output 1
- Output 2
- Output 3

What is a Neural Network Activation Function?

An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.

The role of the Activation Function is to derive output from a set of input values fed to a node (or a layer).



What is a Neural Network Activation Function?

So, What exactly is a node?

Well, if we compare the neural network to our brain, a node is a replica of a neuron that receives a set of input signals.

Depending on the nature and intensity of these input signals, the brain processes them and decides whether the neuron should be activated (“fired”) or not.

The activation function also called as Transfer function in Artificial Neural Network.

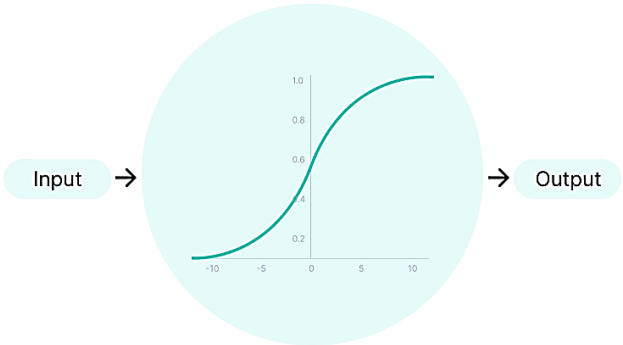
The primary role of the Activation Function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output.

Why do Neural Networks Need an Activation Function?

So we know what Activation Function is and what it does, but

Why do Neural Networks need it?

Well, the purpose of an activation function is to add non-linearity to the neural network.



Activation Function

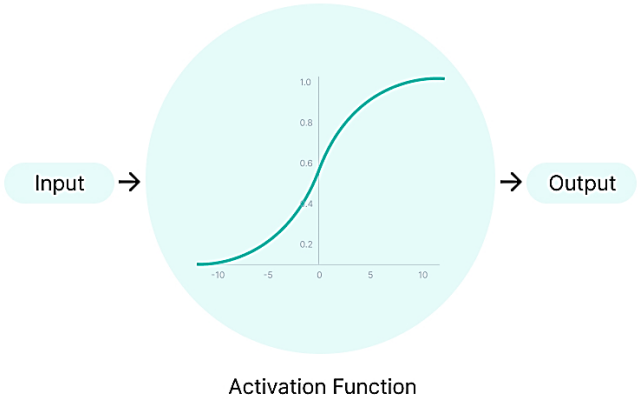
Why do Neural Networks Need an Activation Function?

Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it.

Let’s suppose we have a neural network working without the activation functions.

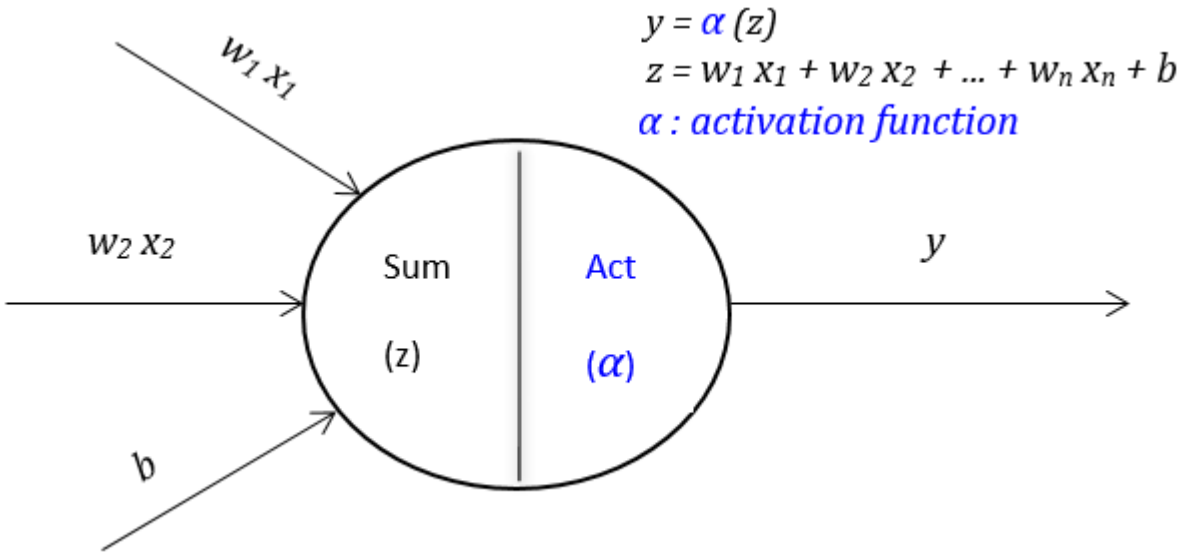
In that case, every neuron will only be performing a linear transformation on the inputs using the *weights and biases*. It’s because it doesn’t matter how many hidden layers we attach in the neural network; all layers will behave in the same way because *the composition of two linear functions is a linear function itself*.

Although the neural network becomes simpler, learning any complex task is impossible, and our model would be just a linear regression model.



Activation Function

Why Activation Functions



The need for these activation functions includes converting the linear input signals and models into non-linear output signals, which aids the learning of high order polynomials for deeper networks.

Types of Neural Network Activation functions

Binary Step function

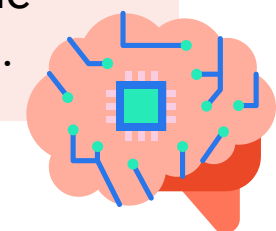
Binary step function depends on a threshold value that decides whether a neuron should be activated or not

Linear Activation Function

The linear activation function is also known as Identity Function where the activation is proportional to the input.

Non-Linear Activation Function

The most used function, it makes it easy for neural network model to adapt with a variety of data and to differentiate between the outcomes.

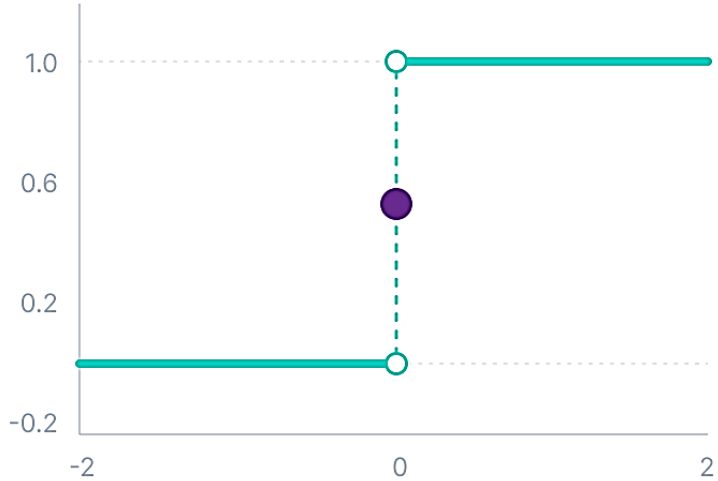


Binary Step Function

Binary step function depends on a threshold value that decides whether a neuron should be activated or not.

The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.

Here are some of the limitations of binary step function:
•It cannot provide multi-value outputs—for example, it cannot be used for *multi-class classification problems*.
•The gradient of the step function is zero, which causes a hindrance in the backpropagation process.



Binary step

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

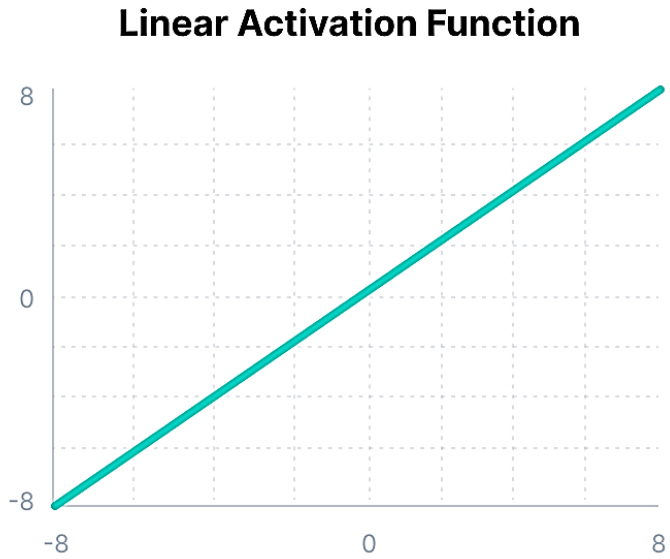
Linear Activation Function

The linear activation function is also known as Identity Function where the activation is proportional to the input.

However, a linear activation function has two major problems :

It's not possible to use backpropagation as the derivative of the function, and has no relation to the input x.

All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer



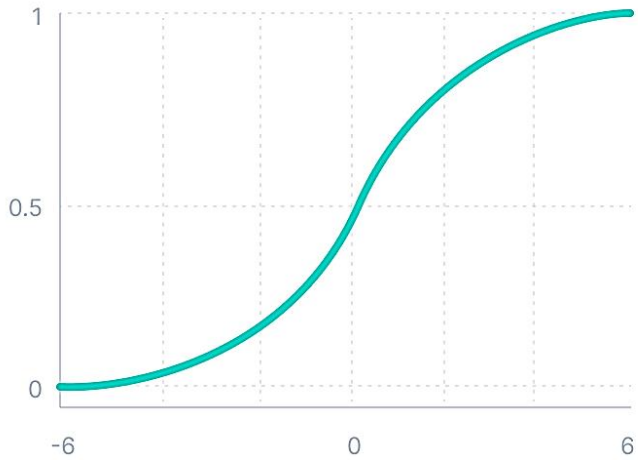
Non-Linear Activation Function

The linear activation function have limited power, this does not allow the model to create complex mappings between the network's inputs and outputs.

However, a Non-linear activation function solve following limitations of linear function :

They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.

They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

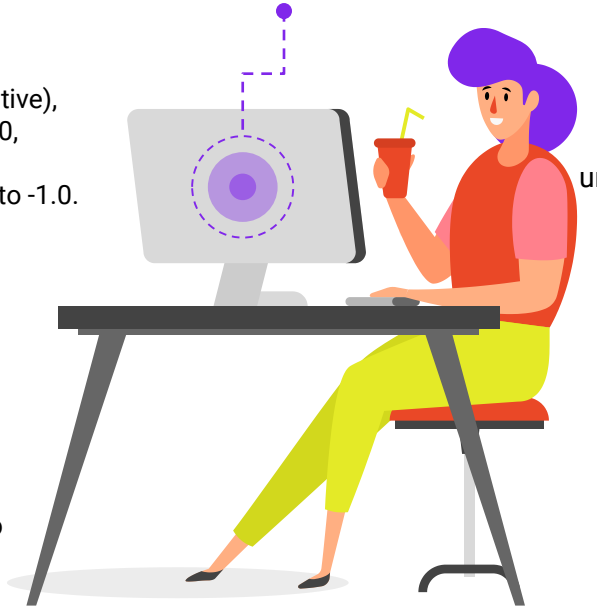


$$f(x) = \frac{1}{1 + e^{-x}}$$

Various Non-Linear Activation Functions

ANN

Non-Linear Activation Functions



01 Logistic Activation Function

This function takes any real value as input and outputs values in the range of 0 to 1.

02 Tanh Function

In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

03 ReLU Function

Rectified Linear Unit: The neurons will only be deactivated if the output of the linear transformation is less than 0.

04 Leaky ReLU Function

improved version of ReLU function to solve the Dying ReLU problem as it has a small positive slope in the negative area

05 Parametric ReLU Function

aims to solve the problem of gradient's becoming zero for the left half of the axis.

06 ELU Function

Exponential linear units uses a log curve to define the negative values unlike the leaky ReLU and Parametric ReLU functions with a straight line.

07 Softmax Function

It calculates the relative probabilities and return probability of each class

08 Swish Function

Developed by google, it enhances the expression of input data and weight to be learnt

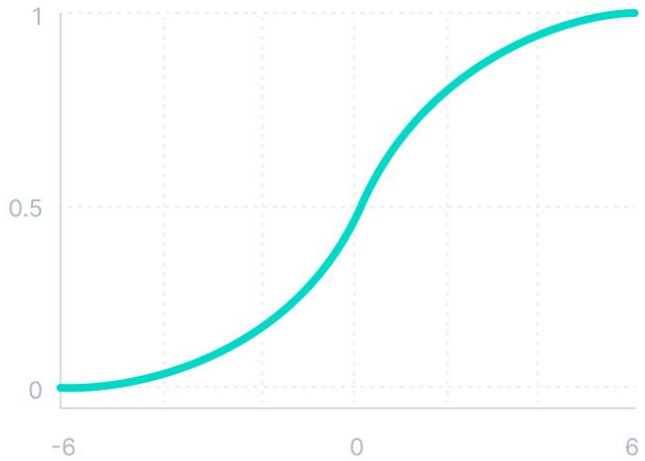
Logistic Activation Function

This function takes any real value as input and outputs values in the range of 0 to 1.

The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below.

Here's why sigmoid/logistic activation function is one of the most widely used functions:

•It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
•The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.



Tan-H (Hyperbolic Tangent) Function

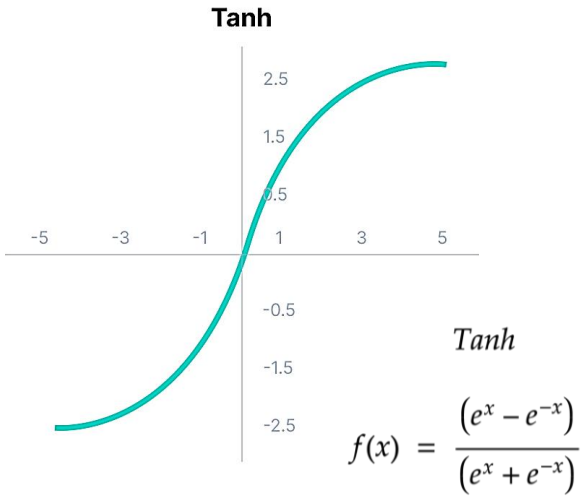
Tanh function is very similar to the ***logistic activation function***, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

Advantages of using this activation function are:

The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or positive.

Usually used in hidden layers of a neural network. It helps in centering the data and makes learning for the next layer much easier.

But problem of vanishing gradient descent issue still persists

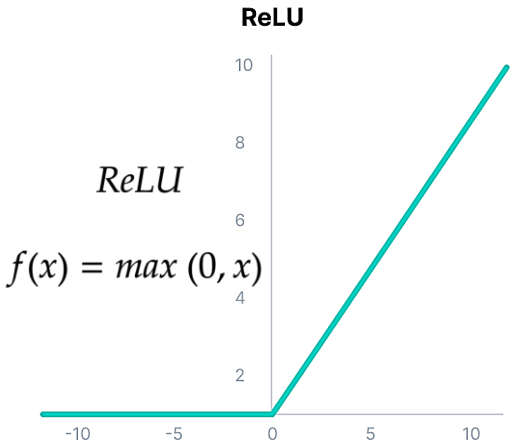


ReLU Function

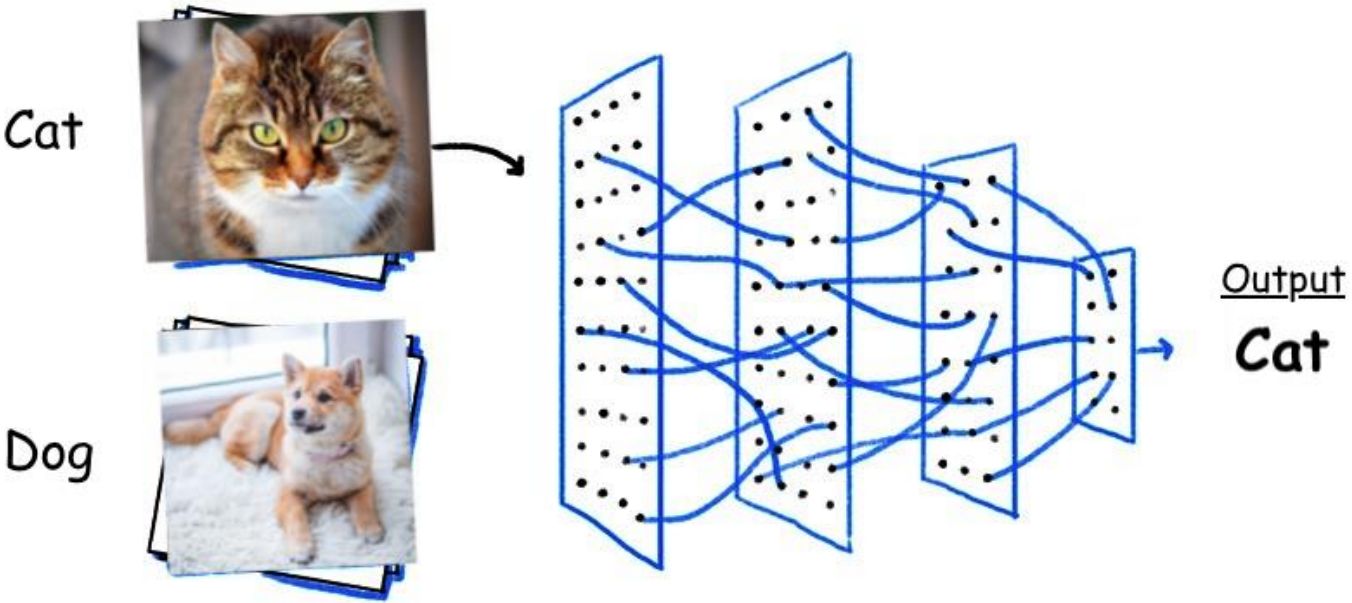
ReLU is an activation function more deeply it is a linear activation function. Which is like a sigmoid and tanh activation function but better than them, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. ReLU function does not activate all the neurons at the same time.

The neurons will only be deactivated if the output of the linear transformation is less than 0

Where the ReLU function is its derivatives are constant. If in input the function gets a negative it returns 0 or if the input is greater than 0 it returns a similar value back. That is why we can say the output from the ReLU has ranged between 0 to infinity.

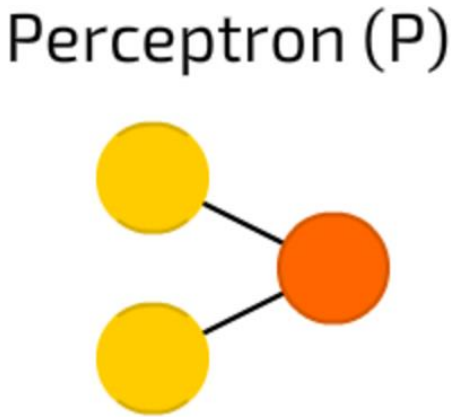


Classification, Detection and Segmentation



Classification of Neural Networks

The simplest and oldest model of Neuron, as we know it. Takes some inputs, sums them up, applies activation function and passes them to output layer. That’s it.



Classification of Neural Networks

01 Perceptron

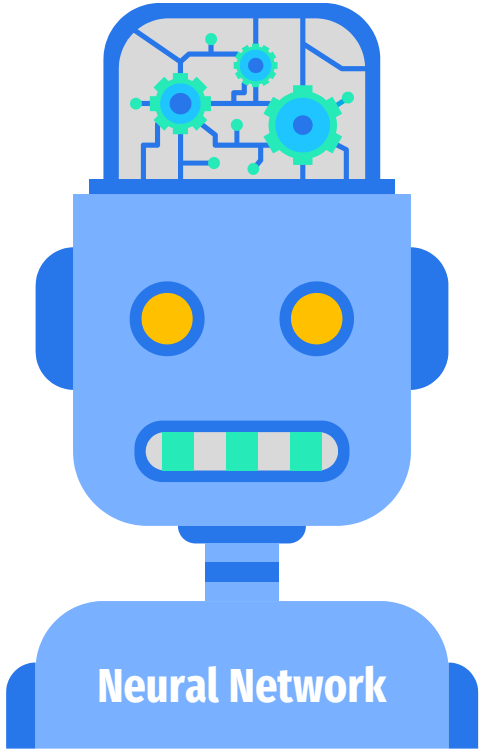
02 Feed Forward

03 Multilayer Perceptron

04 Convolutional NN

05 Recurrent NN

06 Long Short Term Memory

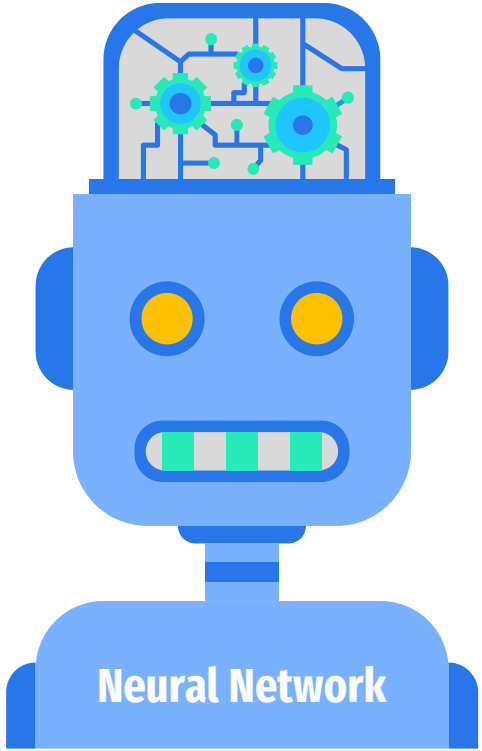


Classification of Neural Networks

Attention Based 07

Auto Encoders 09

Denoising Auto Encoders 11



08 Gated Recurrent Unit

10 Variational Auto Encoders

12 Sparse Auto Encoders

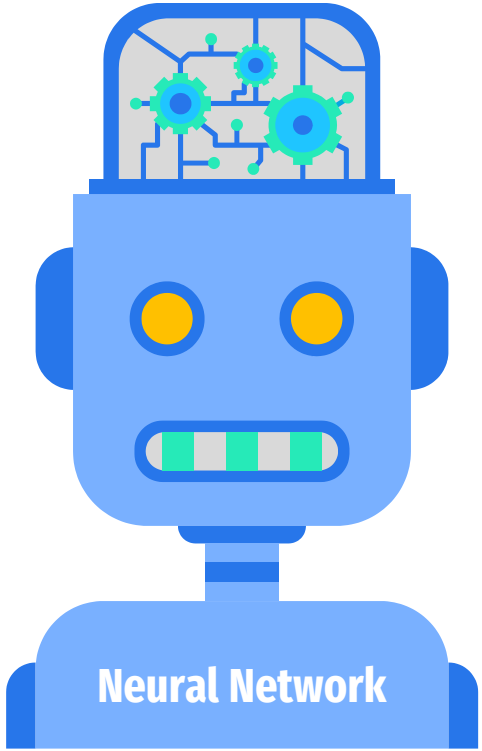
Classification of Neural Networks

Markov Chain 13

Boltzmann Machine 15

Deep Convolutional Network 17

Kohonen Network 19



14 Hopfield Network

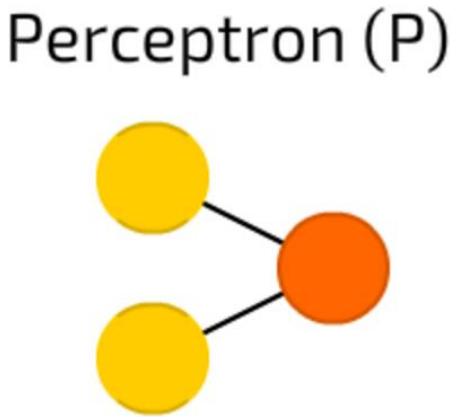
16 Restricted BM

18 Deconvolutional Network

20 Support Vector Machine

Perceptron

The simplest and oldest model of Neuron, as we know it. Takes some inputs, sums them up, applies activation function and passes them to output layer. That’s it.

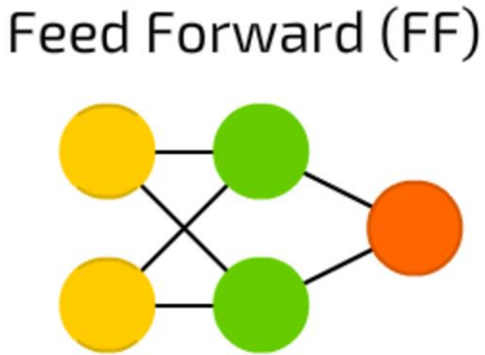


Feed Forward (Backpropagation)

- all nodes are fully connected
- activation flows from input layer to output, without back loops
- there is one layer between input and output (hidden layer)

In most cases this type of networks is trained using Backpropagation method. The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule;

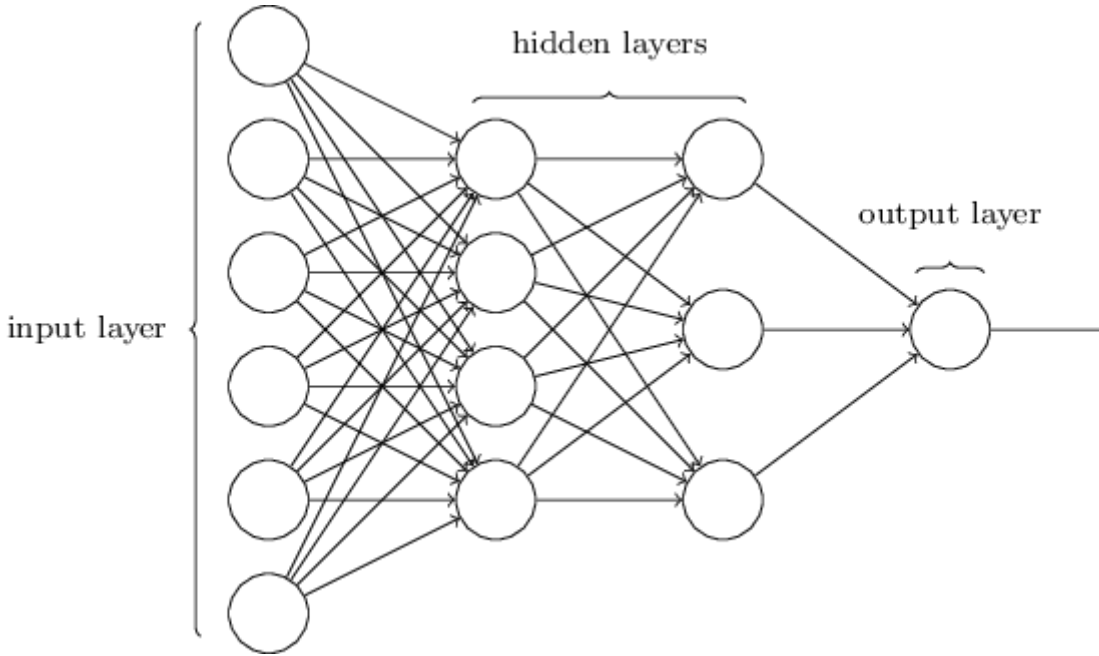
this is an example of dynamic programming.



Multilayer Perceptron (Deep Neural Network)

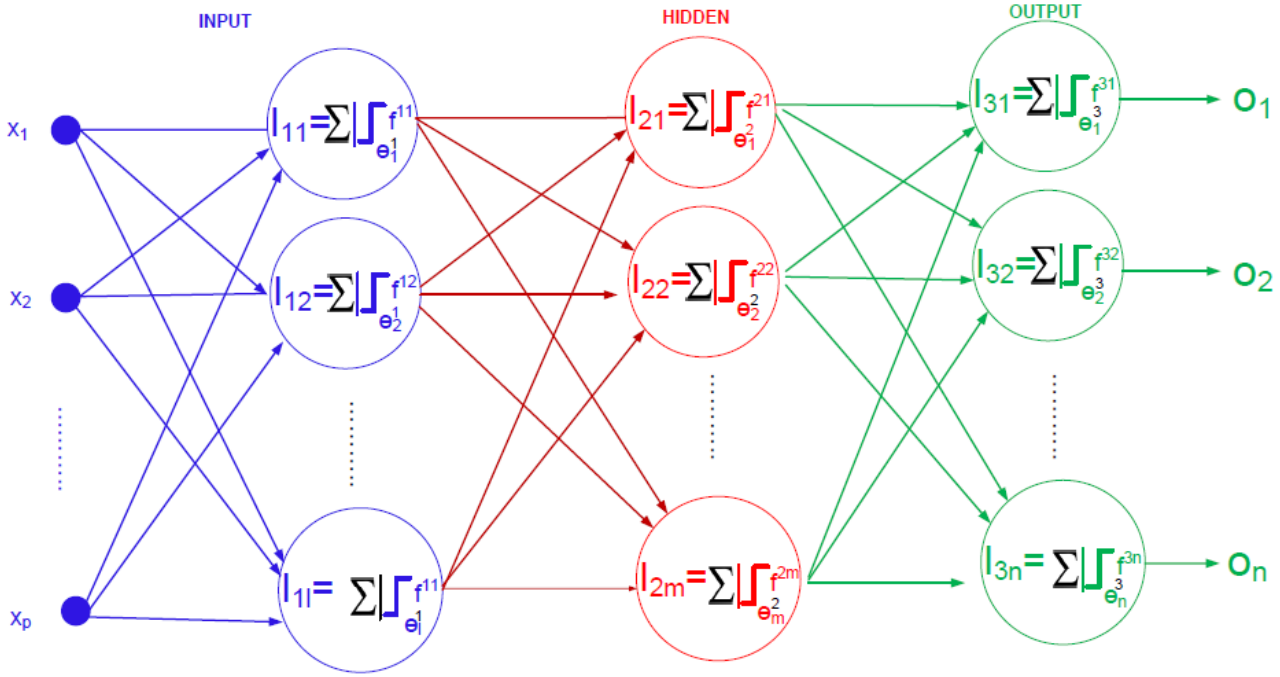
This NN consists of more than one hidden layer. It helps us to solve complex real world problems. A multilayer feedforward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The goal of a feedforward network is to approximate some function.

There are no feedback connections in which outputs of the model are fed back into itself.



Convolutional Neural Network - CNN

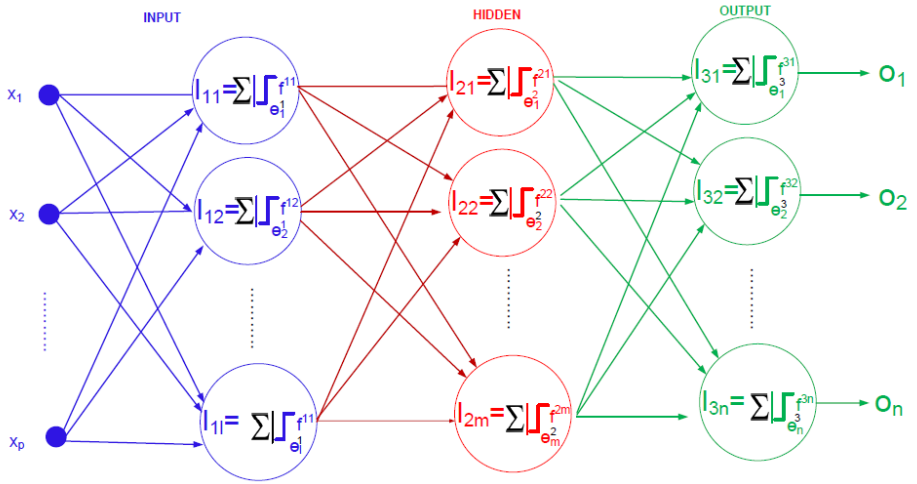
CNN's are the most mature form of deep neural networks to produce the most accurate i.e. better than human results in computer vision.



Convolutional Neural Network - CNN

CNN's are made of layers of Convolutions created by scanning every pixel of images in a dataset. As the data gets approximated layer by layer, CNN's start recognizing the patterns and thereby recognizing the objects in the images. These objects are used extensively in various applications for identification, classification, recognition, etc.

Example: Google Translator and Google Lens.

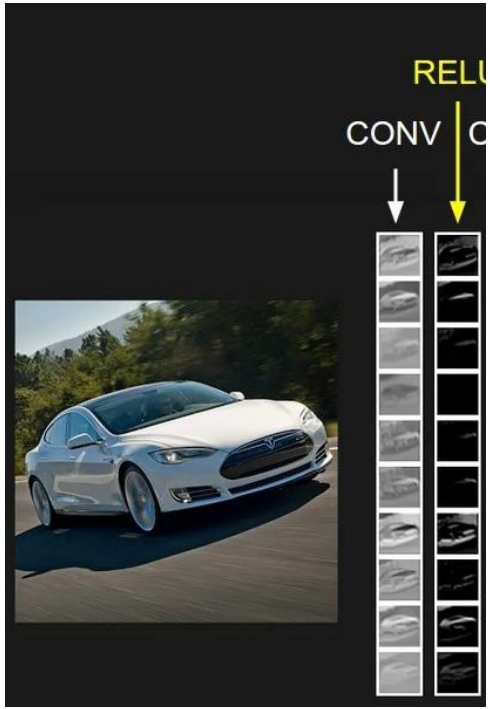


Convolutional Neural Network - CNN

The application of CNNs is exponential as they are even used in solving problems that are primarily not related to computer vision. A very simple but intuitive explanation of CNNs can be found here.

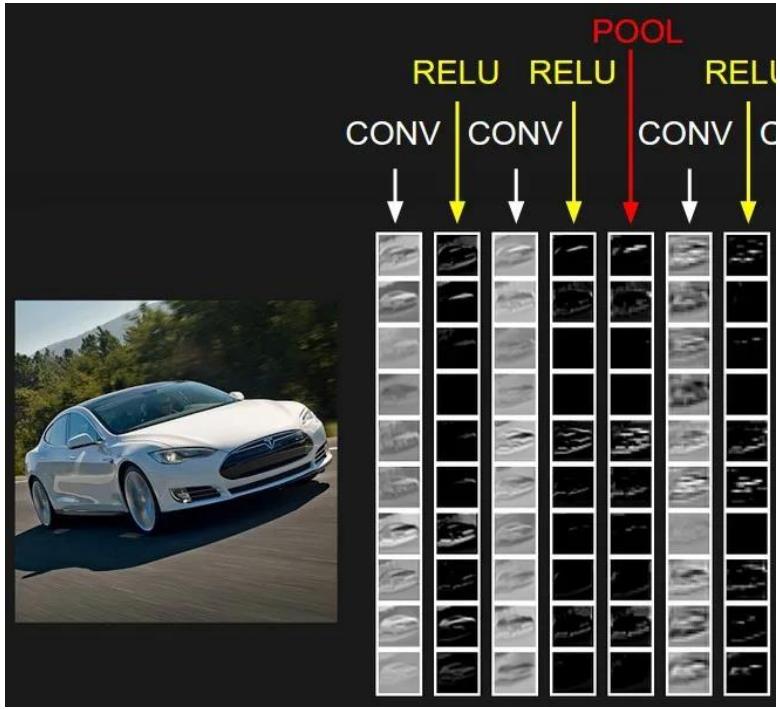
Convolutional Neural Network - CNN

The application of CNNs is exponential as they are even used in solving problems that are primarily not related to computer vision. A very simple but intuitive explanation of CNNs can be found here.



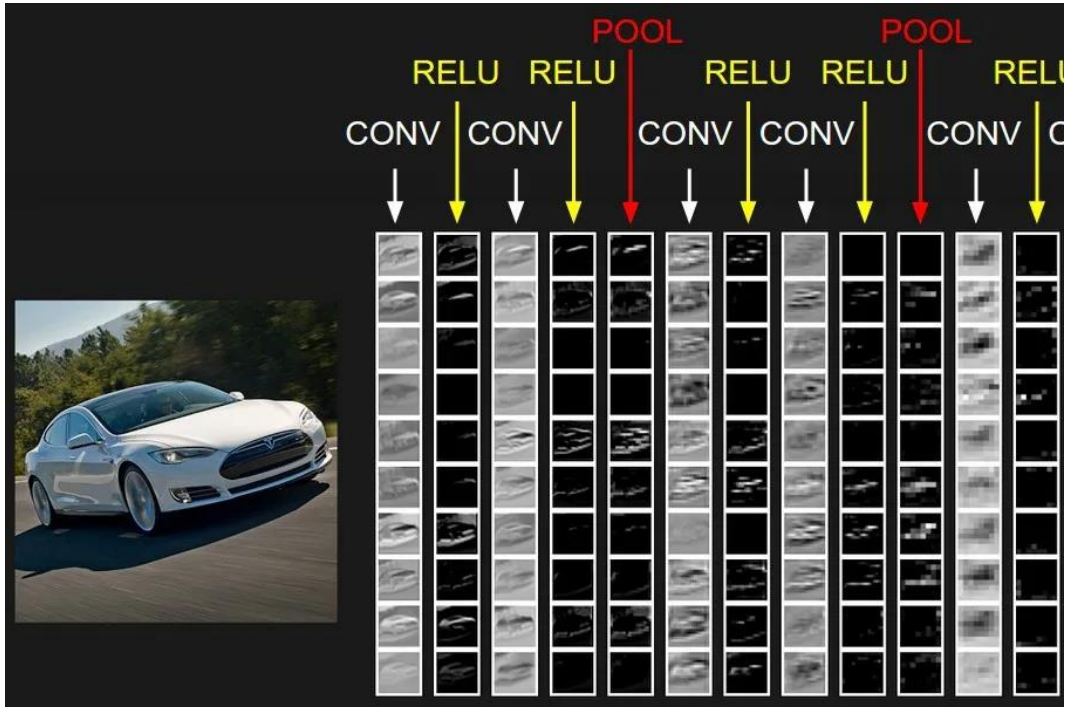
Convolutional Neural Network - CNN

The application of CNNs is exponential as they are even used in solving problems that are primarily not related to computer vision. A very simple but intuitive explanation of CNNs can be found here.



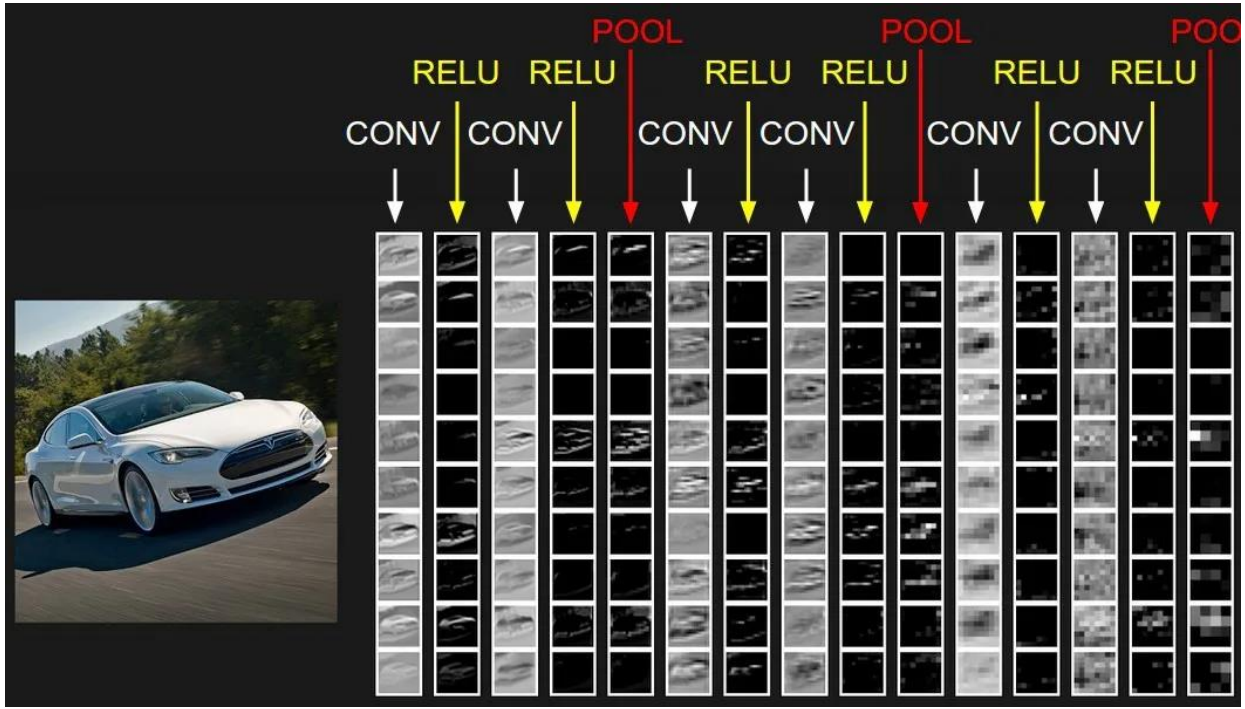
Convolutional Neural Network - CNN

The application of CNNs is exponential as they are even used in solving problems that are primarily not related to computer vision. A very simple but intuitive explanation of CNNs can be found here.



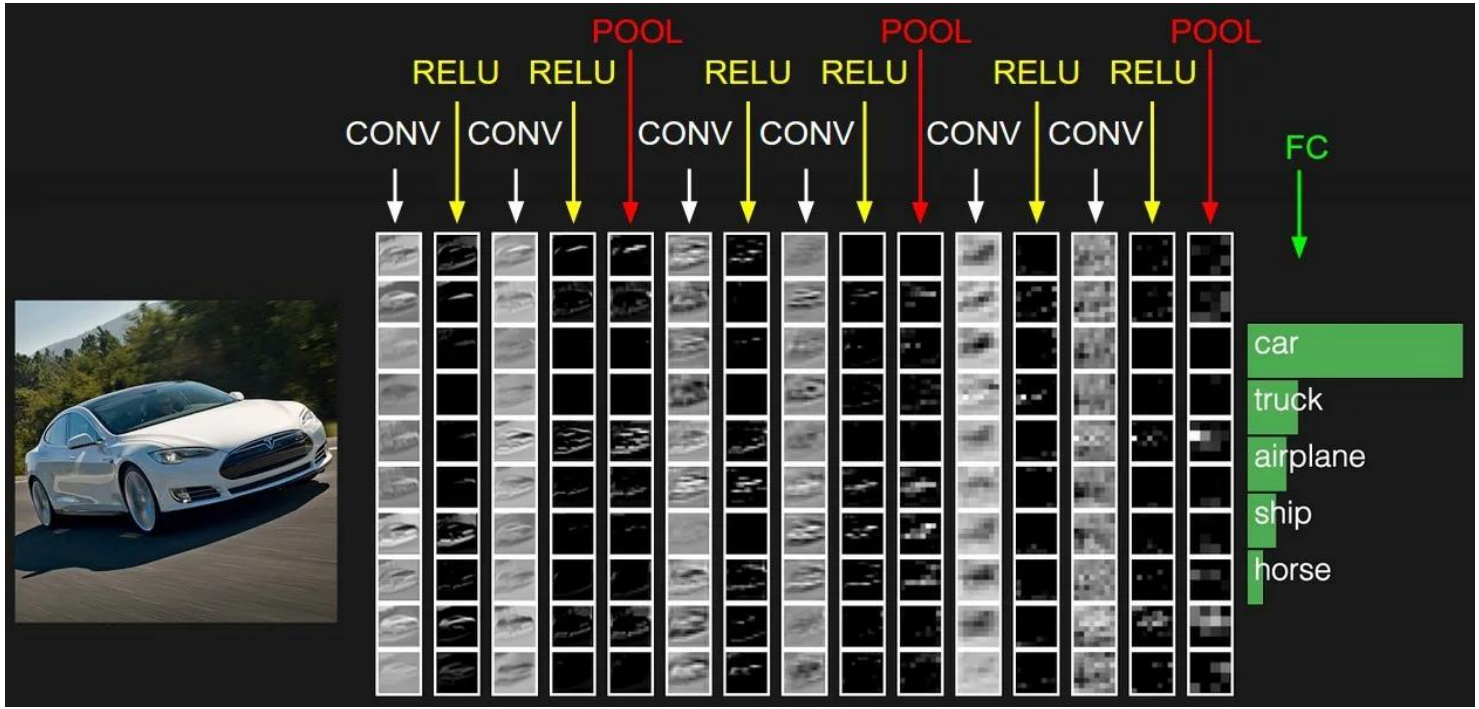
Convolutional Neural Network - CNN

The application of CNNs is exponential as they are even used in solving problems that are primarily not related to computer vision. A very simple but intuitive explanation of CNNs can be found here.



Convolutional Neural Network - CNN

The application of CNNs is exponential as they are even used in solving problems that are primarily not related to computer vision. A very simple but intuitive explanation of CNNs can be found here.



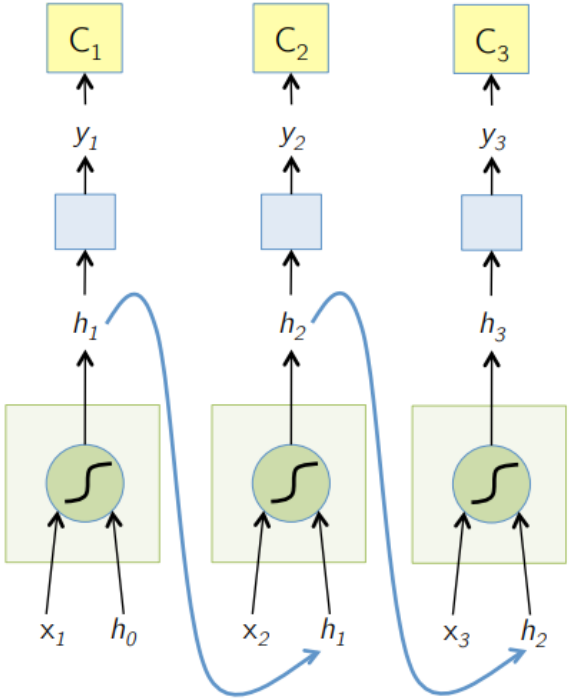
Tesla

Recurrent Neural Network (RNN)

RNN help us to solve problems related to NLP. RNNs feed the output of a few hidden layers back to the input layer to aggregate and carry forward the approximation to the next iteration of the input dataset.

It also helps the model to self-learn and corrects the predictions faster to an extent. Such models are very helpful in understanding the semantics of the text in NLP operations. There are different variants of RNNs like *Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU),* etc.

Here in the diagram, the activation from h1 and h2 is fed with input x2 and x3 respectively.



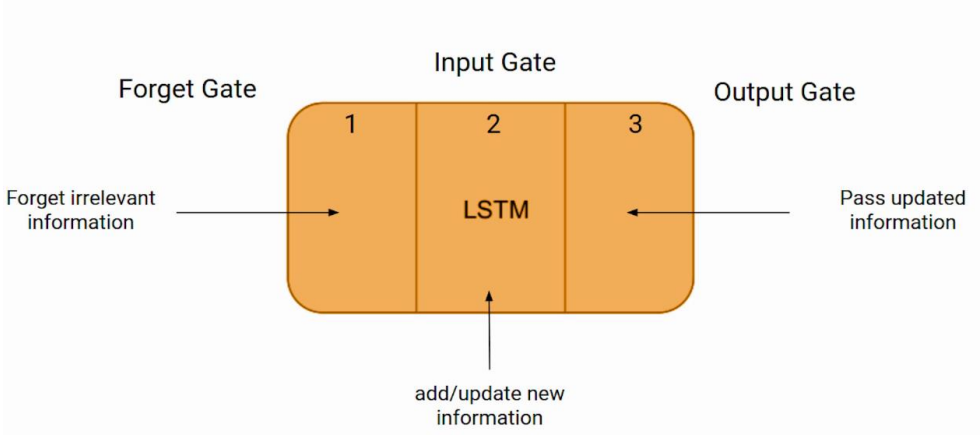
Long Short Term Memory (LSTM)

Let’s say while watching a video you remember the previous scene or while reading a book you know what happened in the earlier chapter. Similarly RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they can not remember Long term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

Long Short Term Memory (LSTM)

LSTMs are designed specifically to address the vanishing gradients problem with the RNN. Vanishing Gradients happens with large neural networks where the gradients of the loss functions tend to move closer to zero making pausing neural networks to learn.

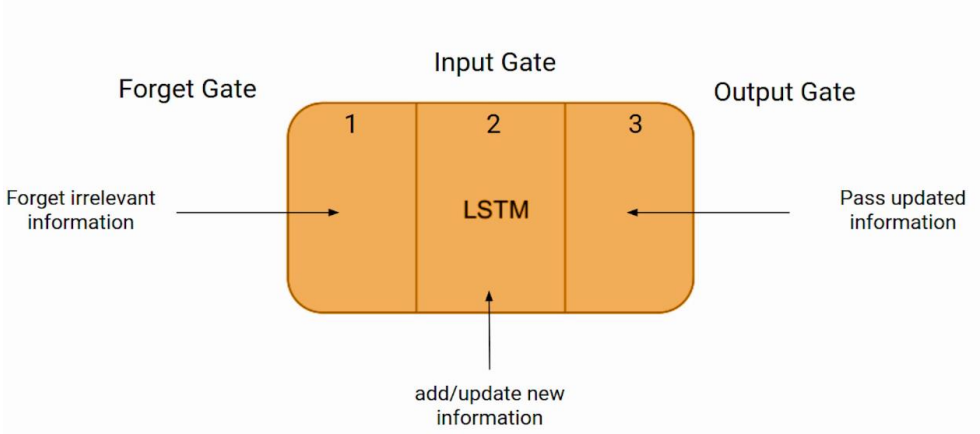
LSTM solves this problem by preventing activation functions within its recurrent components and by having the stored values unmutated. This small change gave big improvements in the final model resulting in tech giants adapting LSTM in their solutions.



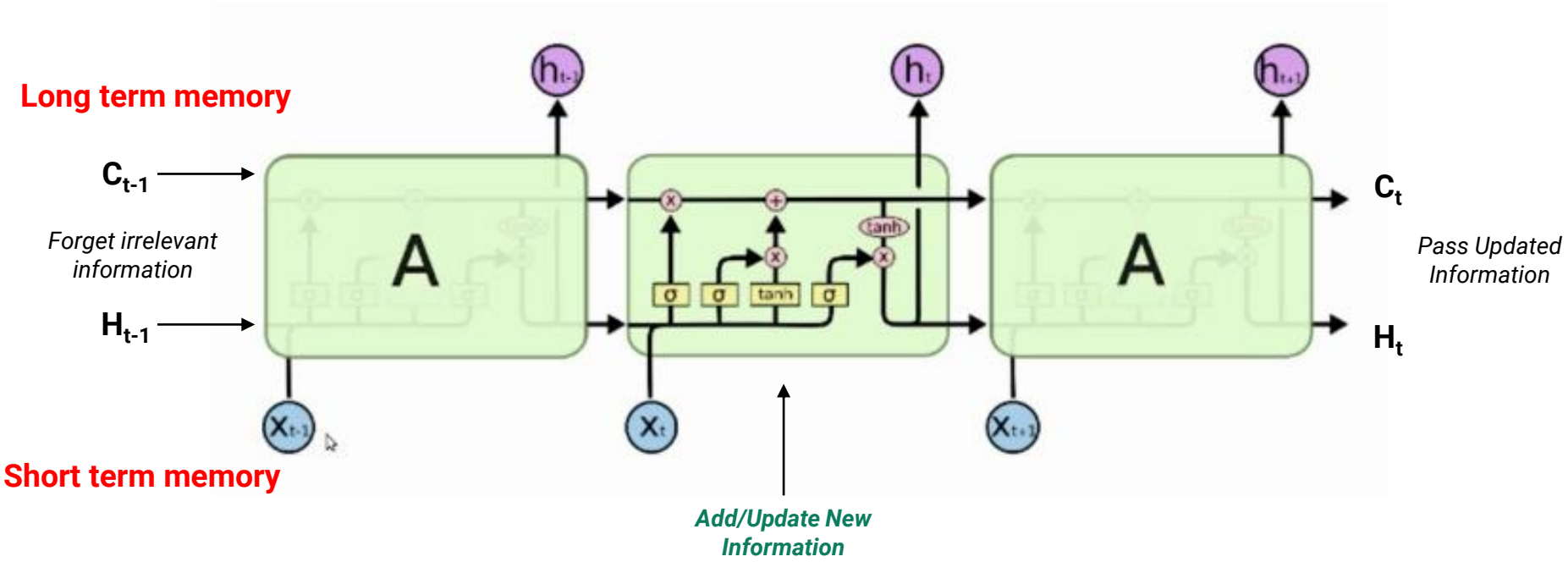
These three parts of an LSTM cell are known as gates. The first part is called **Forget gate, the** second part is known as **the Input gate** and the last one is **the Output gate**.

Long Short Term Memory (LSTM)

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.



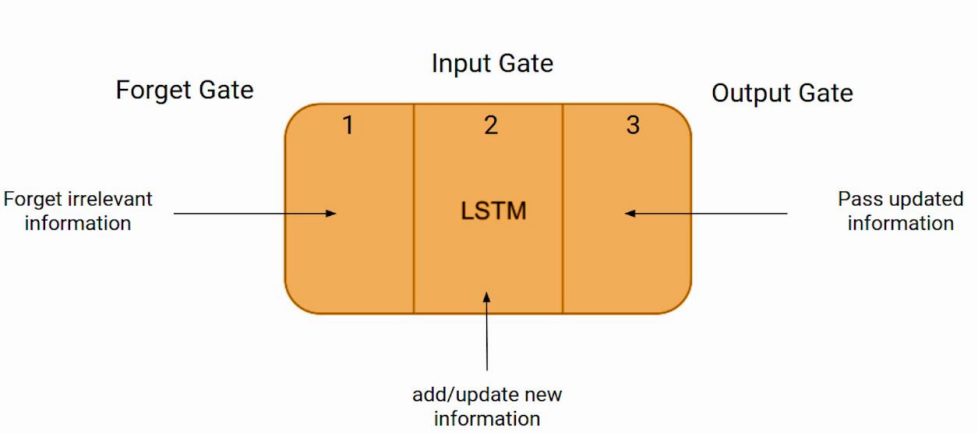
Long Short Term Memory (LSTM)



Just like a simple RNN, an LSTM also has a hidden state where H_{t-1} represents the hidden state of the previous timestamp and H_t is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by C_{t-1} and C_t for previous and current timestamp respectively.

Long Short Term Memory (LSTM)

RISHI is a *NICE* person. **MAX** is an *EVIL*.



It is very clear, in the first sentence we are talking about **RISHI** and as soon as we encounter the full stop(.) we started talking about **MAX**. As we move from the first sentence to the second sentence, our network should realize that we are no more talking about **RISHI**. Now our subject is **MAX**. Here, the Forget gate of the network allows it to forget about it.

Long Short Term Memory (LSTM)

Rishi is a *NICE* person. **MAX** is an *EVIL*.

Forget Gate: the first step is to decide whether we should keep the information from the previous timestamp or forget it. In this case our network should forget about RISHI.

$C_{t-1} * f_t = 0$ ...if $f_t = 0$ (forget everything)

$C_{t-1} * f_t = C_{t-1}$ ...if $f_t = 1$ (forget nothing)

INPUT Gate: is used to quantify the importance of the new information carried by the input.

Rishi *knows swimming. He told me over the phone that he had served the navy for four long years*

Now just think about it, based on the context given in the first sentence, which information of the second sentence is critical. First, *he used the phone to tell or he served in the navy*. In this context, it doesn't matter whether he used the phone or any other medium of communication to pass on the information. The fact *that he was in the navy* is important information and this is something we want our model to remember. This is the task of the **Input gate.**

Long Short Term Memory (LSTM)

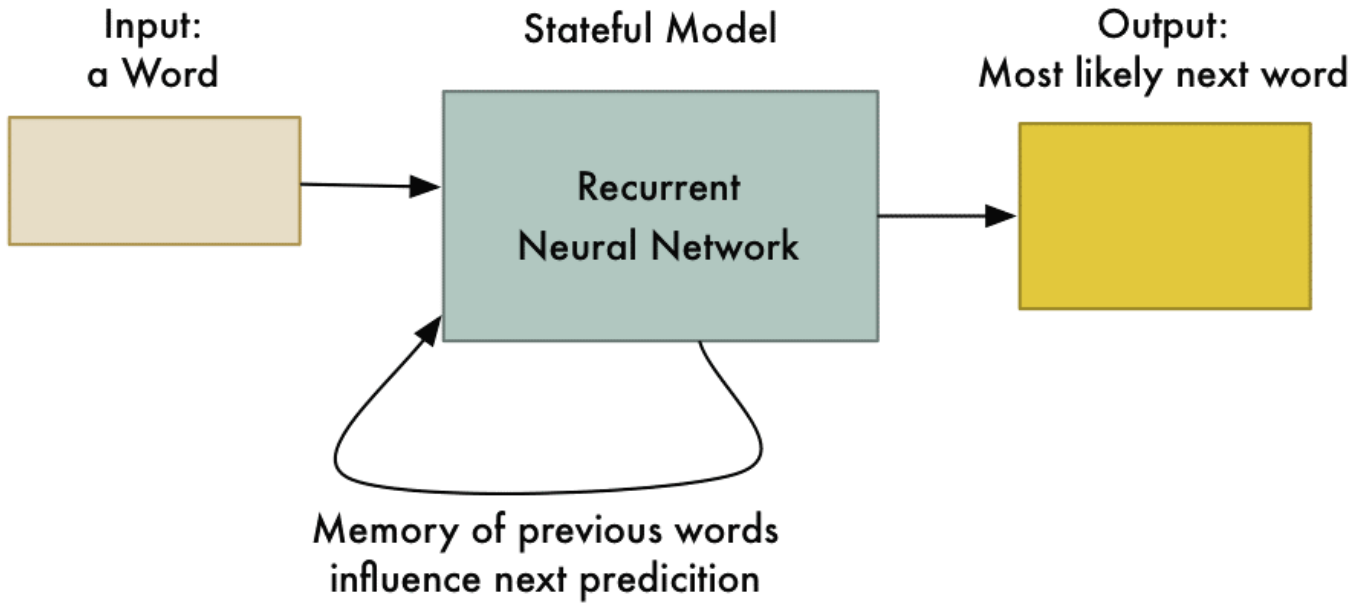
ROCKY BHAI single-handedly fought the enemy (*Adheera*) at KGF and died for his KGF. For his contributions, brave_____ .

OUTPUT Gate: Its value will also lie between 0 and 1 because of this sigmoid function. It turns out that the hidden state is a function of Long term memory (C_t) and the current output. If you need to take the output of the current timestamp just apply the *SoftMax activation* on hidden state H_t.

$$\text{Output} = \text{Softmax}(H_t)$$

During this task, we have to complete the second sentence. Now, the minute we see the word brave, we know that we are talking about a person (i.e. *ROCKY*). In the sentence only ROCKY is brave, we can not say the enemy (*Adheera*) is brave or the KGF is brave. So based on the current expectation we have to give a relevant word to fill in the blank. That word is our output and this is the function of our Output gate.

Long Short Term Memory (LSTM)



Output so far:

Machine

Attention-based Networks

The Attention models are built by focusing on part of a subset of the information they're given thereby eliminating the overwhelming amount of background information that is not needed for the task at hand.

Attention models are built with a combination of soft and hard attention integrated with back-propagating.

Multiple attention models stacked hierarchically is called Transformer. These transformers are more efficient to run the stacks in parallel so that they produce state of the art results with less time for training the model.

An attention distribution becomes very powerful when used with CNN/RNN and can produce text description to an image as follow.

Attention-based Networks

The Attention models are built by focusing on part of a subset of the information they're given thereby eliminating the overwhelming amount of background information that is not needed for the task at hand.

Attention models are built with a combination of soft and hard attention integrated with back-propagating.

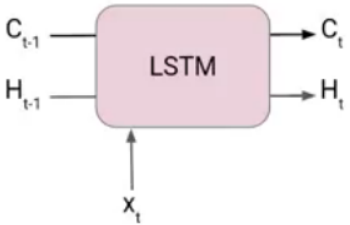
Multiple attention models stacked hierarchically is called Transformer. These transformers are more efficient to run the stacks in parallel so that they produce state of the art results with less time for training the model.

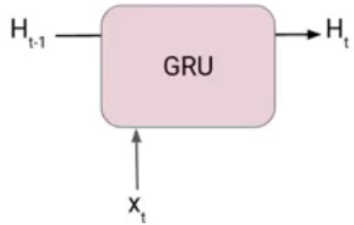
An attention distribution becomes very powerful when used with CNN/RNN and can produce text description to an image as follow.

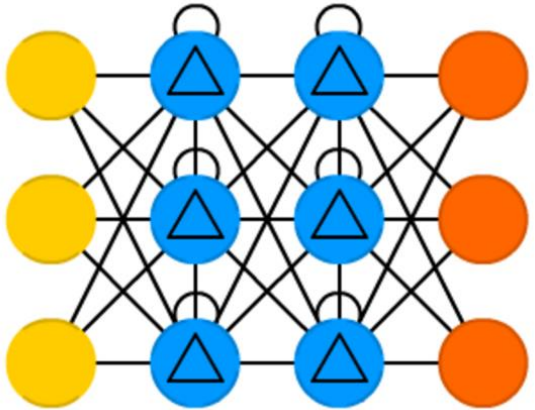
Gated Recurrent Unit (GRU)

GRUs are very similar to Long Short Term Memory(LSTM). Just like LSTM, GRU uses gates to control the flow of information. GRU does not have a separate cell state (Ct). It only has a hidden state(Ht). Due to the simpler architecture, GRUs are faster to train. GRU consumes less gates then LSTM and almost the same effective.

Currently used the most in sound (music) and speech synthesis.







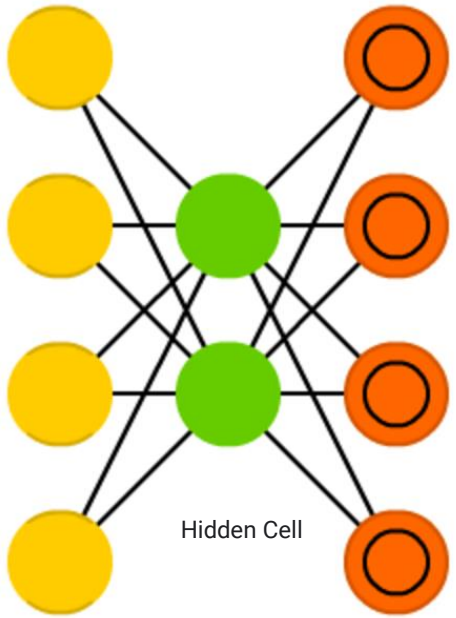
AutoEncoders

Autoencoders are used for classification, clustering and feature compression.

When you train FF neural networks for classification you mostly feed input and expect one of output cells to be activated. This is called “supervised learning”.

AEs, on the other hand, can be trained without supervision. Their structure — when number of hidden cells is smaller than number of input cells (and number of output cells equals number of input cells)

when the AE is trained the way the output is as close to input as possible, forces AEs to generalise data and search for common patterns.

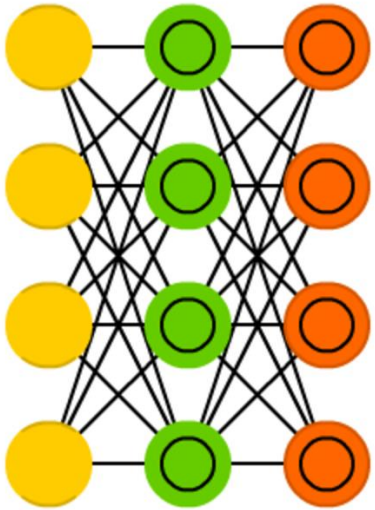


Variational AutoEncoders

VAEs, comparing to AE, compress probabilities instead of features.

Despite that simple change, when AEs answer the question *“how can we generalise data?”,* VAEs answer the question *“how strong is a connection between two events?*

a variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.



Probabilistic Hidden Cell

Input Cell

Match Input Output Cell

Perceptron Training Algorithm

Frank Rosenblatt invented the perceptron at the Cornell Aeronautical Laboratory in 1957. A perceptron has one or more than one inputs, a process, and only one output.

The concept of perceptron has a critical role in machine learning. It is used as an algorithm or a linear classifier to facilitate supervised learning of binary classifiers. Supervised learning is amongst the most researched of learning problems. A supervised learning sample always consists of an input and a correct/explicit output. The objective of this learning problem is to use data with correct labels for making predictions on future data, for training a model.

A linear classifier that the perceptron is categorized as is a classification algorithm, which relies on a linear predictor function to make predictions. Its predictions are based on a combination that includes weights and feature vector.

The perceptron algorithm, in its most basic form, finds its use in the binary classification of data.

Components of a Perceptron

Input: x1, x2, x3, x4, ...xn

Weights: w1, w2, w3, w4, ..wn
These are values that are calculated during the training of the model.

Bias: It allows the classifier to move the decision boundary around from its original position. Bias allows for higher quality and faster model training.

Activation/step function: Activation or step functions are used to create non-linear neural networks.

Weighted summation: $\sum w_i x_i$ for all i -> [1 to n].

Categorization of a Perceptron

Perceptron algorithms can be categorized into:

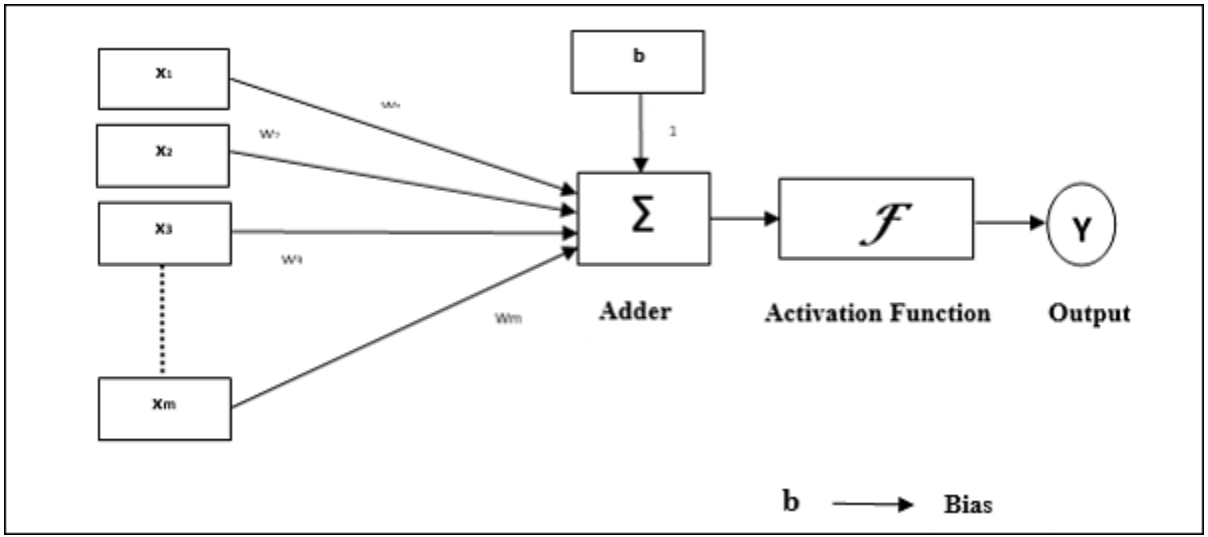
single-layer
multi-layer perceptron’s.

The single-layer type organizes neurons in a single layer.

The multi-layer type arranges neurons in multiple layers.

In the multi-layer scenario, each neuron of the first layer takes inputs and gives a response to the group of neurons present in the second layer. This process continues until the last layer is reached.

Training Algorithm for Single Layer Perceptron

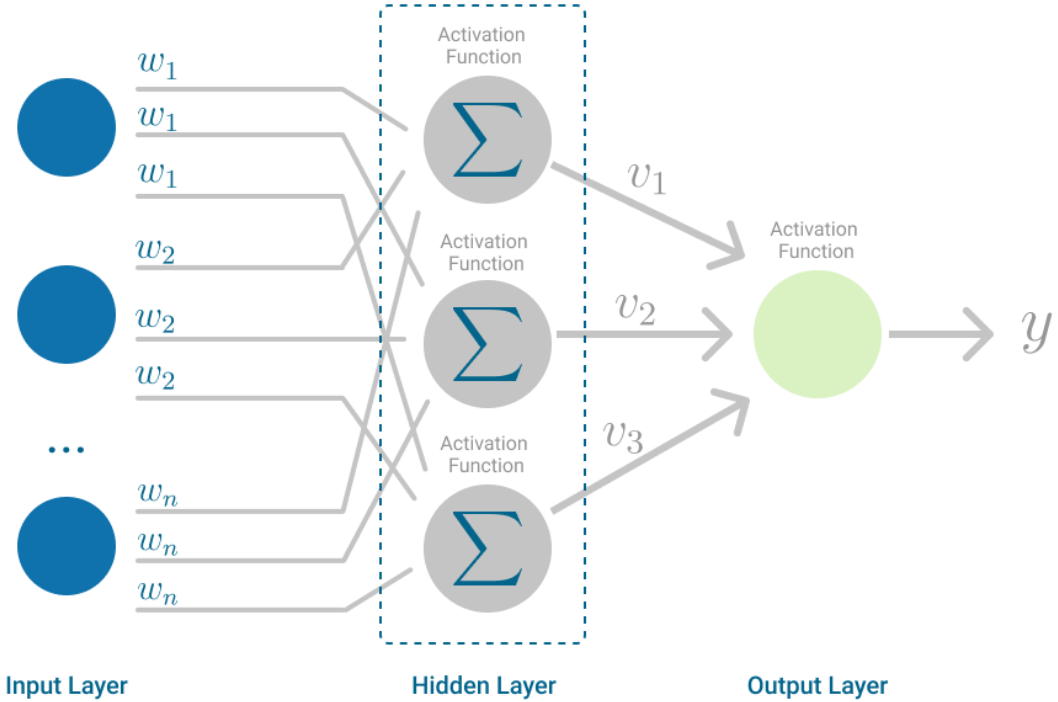


It consists of a single neuron with an arbitrary number of inputs along with adjustable weights, but the output of the neuron is 1 or 0 depending upon the threshold. It also consists of a bias whose weight is always 1.

Training Algorithm for Multi-Layer Perceptron

Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

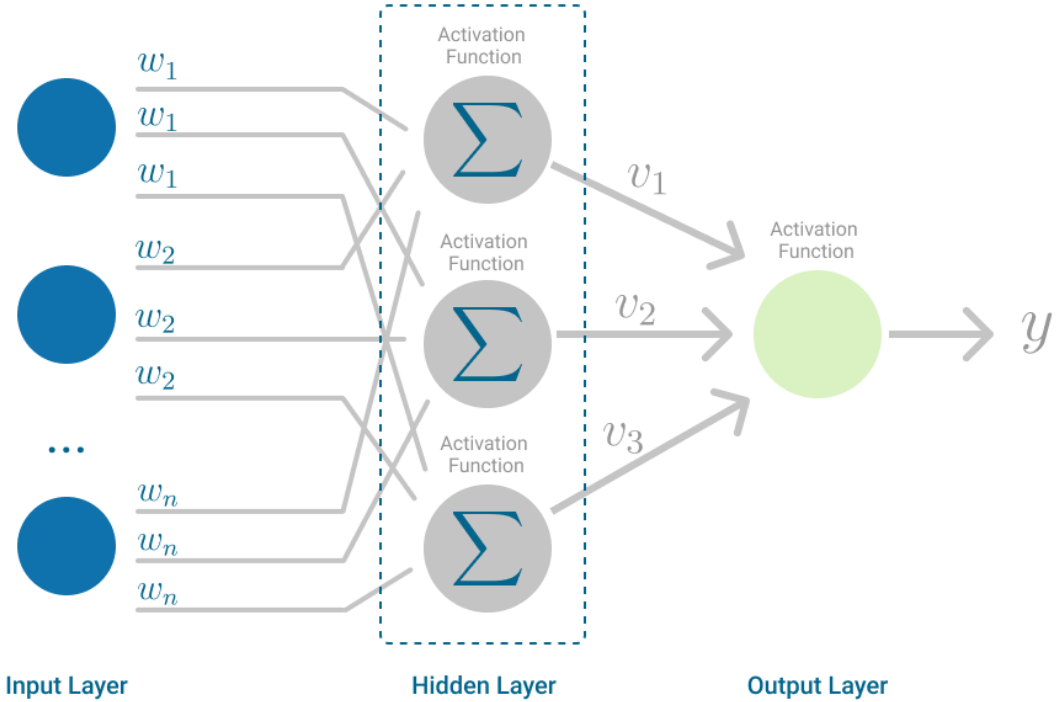


Training Algorithm for Multi-Layer Perceptron

But it has more to it,

If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning.

This is where we can use Backpropagation approach to solve several problems.

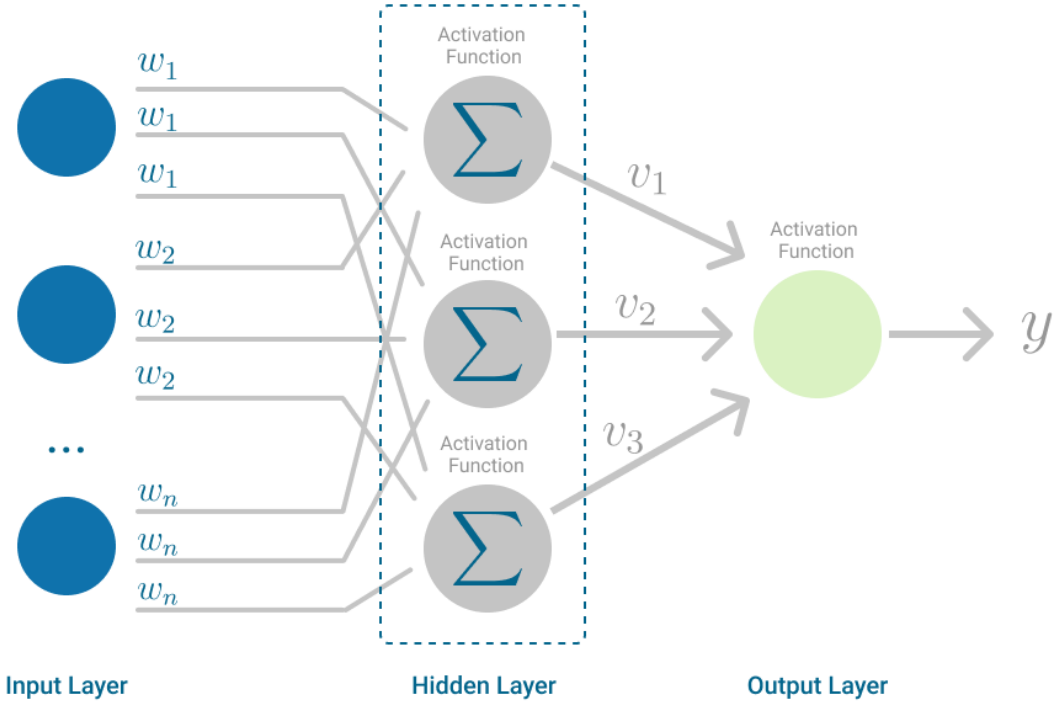


Training Algorithm for Multi-Layer Perceptron

In feed forward, we usually compute MSE i.e. Mean Squared Error.

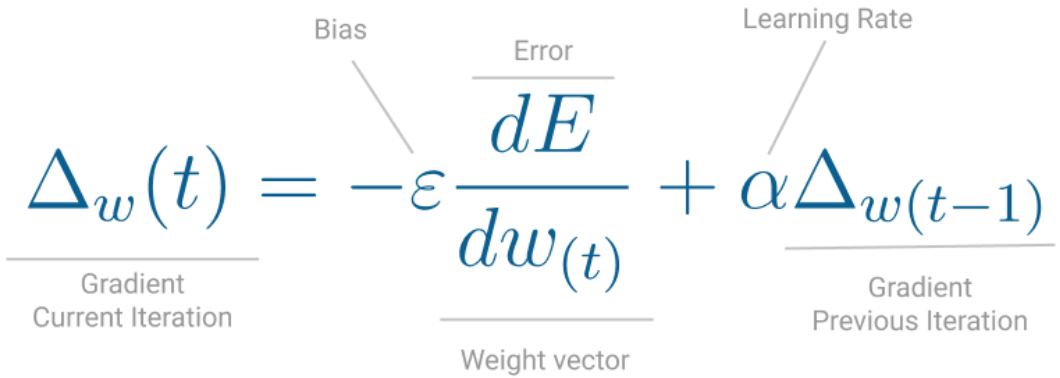
Where as in Backpropagation, we compute gradient.

In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That's how the weights are propagated back to the starting point of the neural network.



Training Algorithm for Multi-Layer Perceptron

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn't changed more than a specified convergence threshold, compared to the previous iteration.



Best example: Using Perceptron for Sentiment Analysis (Binary classifier task).

Steps involve in performing sentiment analysis

1. Collect text (with labels in corpus)
2. Convert free text into format that our machine learning model could work on.
3. Find Term Frequency (encodes any kind of text as a statistic of how frequent each word, or term, is in each sentence and the entire document).
4. Remove English stop words
5. Apply L1 Normalization
6. Test on sentences the model have never seen before
7. Find accuracy of predictions
8. Use multi-layer perceptron (Activation function, Optimization function (Stochastic gradient descent), learning rate, number of iterations.
9. Calculate value of of the loss function at each iteration.
10. Add more neurons to hidden layer (will improve accuracy)

Learning Rules in Neural Network

Learning rule is a method or a mathematical logic. It helps a Neural Network to learn from the existing conditions and improve its performance. It is an iterative process.

It improves the Artificial Neural Network's performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment. There are different learning rules in the Neural network:

- **Hebbian learning rule –** It identifies, how to modify the weights of nodes of a network.
- **Perceptron learning rule –** Network starts its learning by assigning a random value to each weight.
- **Delta learning rule –** Modification in weight of a node is equal to the multiplication of error and the input.
- **Correlation learning rule –** The correlation rule is the supervised learning.
- **Outstar learning rule –** We can use it when it assumes that nodes or neurons in a network arranged in a layer

Hebbian Learning Rule

The Hebbian rule was the first learning rule. In 1949 Donald Hebb developed it as learning algorithm of the unsupervised neural network. We can use it to identify how to improve the weights of nodes of a network.

The Hebb learning rule assumes that – If two neighbor neurons activated and deactivated at the same time. Then the weight connecting these neurons should increase.

For neurons operating in the opposite phase, the weight between them should decrease. If there is no signal correlation, the weight should not change.

When inputs of both the nodes are either positive or negative, then a strong positive weight exists between the nodes. If the input of a node is positive and negative for other, a strong negative weight exists between the nodes.

This learning rule can be used for both soft- and hard-activation functions.

$$W_{ij} = x_i * x_j$$

Perceptron Learning Rule

As you know, each connection in a neural network has an associated weight, which changes in the course of learning.

the network starts its learning by assigning a random value to each weight.

Calculate the output value on the basis of a set of records for which we can know the expected output value (this is called as learning sample)

The network then compares the calculated output value with the expected value.

Next calculates an error function ∈, which can be the sum of squares of the errors occurring for each individual in the learning sample.

$$\sum_i \sum_j \left(E_{ij} - O_{ij} \right)^2$$

Perceptron Learning Rule

$$\sum_i \sum_j \left(E_{ij} - O_{ij} \right)^2$$

Perform the first summation on the individuals of the learning set, and perform the second summation on the output units.

Eij and Oij are the expected and obtained values of the jth unit for the ith individual.

The network then adjusts the weights of the different units, checking each time to see if the error function has increased or decreased. As in a conventional regression, this is a matter of solving a problem of least squares.

Delta Learning Rule / Widrow & Hoff Rule

The delta rule, is one of the most common learning rules. It depends on supervised learning.

This rule states that the modification in weight of a node is equal to the multiplication of error and the input.

1.The perceptron learning rule originates from the Hebbian assumption while the delta rule is derived from the gradient- descent method (it can be generalized to more than one layer).
2.The delta rule updates the weights between the connections so as to minimize the difference between the net input to the output unit and the target value.
3.The major aim is to minimize all errors over all training patterns. This is done by reducing the error for each pattern, one at a time
4.The delta rule for adjusting the weight of i^{th} pattern (i =1 to n) is

Delta Learning Rule / Widrow & Hoff Rule

For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference.

The delta rule or Adaline rule is the most commonly used learning rule in NN, it adjust the weights so as to minimize the mean squared error. The method is based on the steepest descent algorithm.

We can use the delta learning rule with both single output unit and several output units.
While applying the delta rule assume that the error can be directly measured.

The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

Correlation Learning Rule

The correlation learning rule based on a similar principle as the Hebbian learning rule. It assumes that weights between responding neurons should be more positive, and weights between neurons with opposite reaction should be more negative.

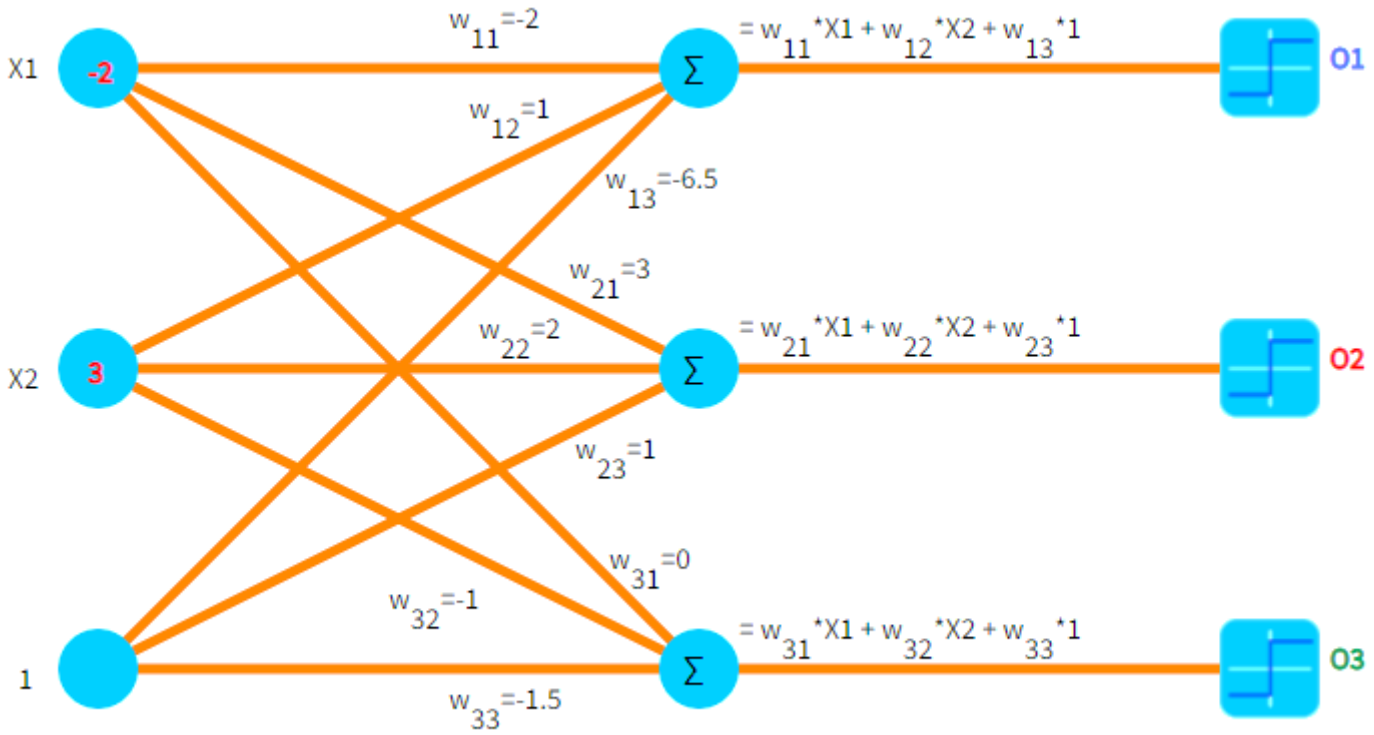
Contrary to the Hebbian rule, the correlation rule is the supervised learning. Instead of actual response, Oi , the desired response, Di, is used for the weight-change calculation.

$$\Delta w_{ij} = \eta \, (D_i) \, X_j$$

where $\mathbf{D_i}$ is the desired value of output signal. This training algorithm usually starts with initialization of weights to zero.

the corresponding weight increase is proportional to their product. The rule typically applies to recording data in memory networks with binary response neurons. It can be interpreted as a special case of the Hebbian rule with a binary activation function and for Oi = Di.

Correlation Learning Rule



Out Star Learning Rule

We use the Out Star Learning Rule when we assume that nodes or neurons in a network arranged in a layer.

Here the weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights.

The out start rule produces the desired response t for the layer of n nodes.

This is a supervised training procedure because desired outputs must be known.

Adaptive Linear Neuron (Adaline)

Adaline which stands for Adaptive Linear Neuron, is a network having a single linear unit. It was developed by Widrow and Hoff in 1960. Some important points about Adaline are as follows −

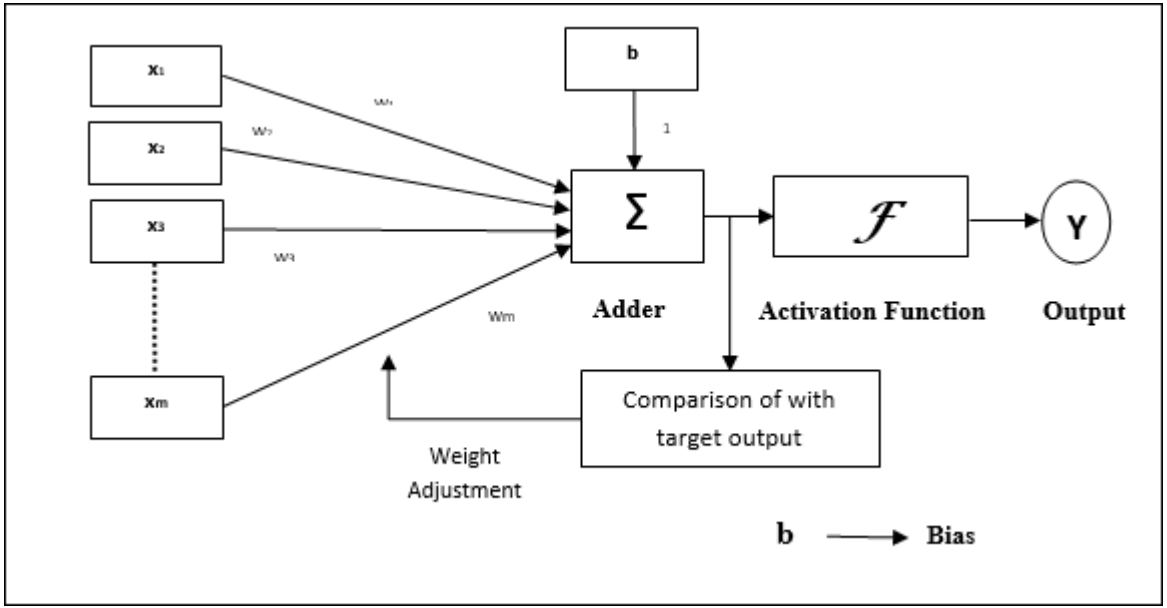
•It uses bipolar activation function.

•It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.

•The weights and the bias are adjustable.

Adaptive Linear Neuron (Adaline)

The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/target output. After comparison on the basis of training algorithm, the weights and bias will be updated.



Multiple Adaptive Linear Neuron(Madaline)

Madaline which stands for Multiple Adaptive Linear Neuron, is a network which consists of many Adalines in parallel. It will have a single output unit. Some important points about Madaline are as follows −

•It is just like a multilayer perceptron, where Adaline will act as a hidden unit between the input and the Madaline layer.

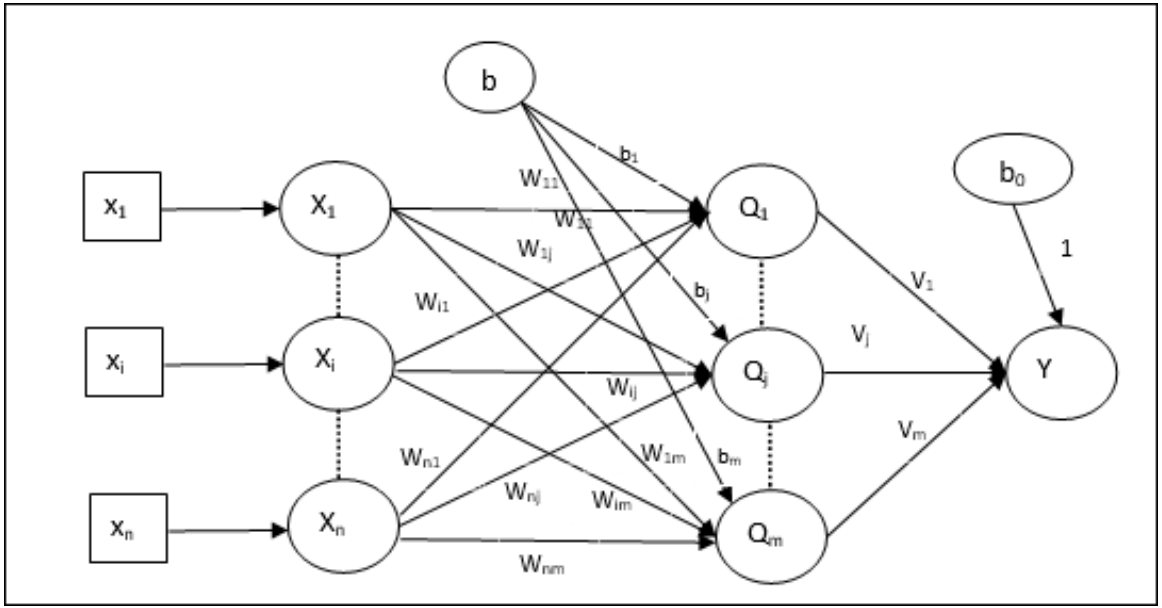
•The weights and the bias between the input and Adaline layers, as in we see in the Adaline architecture, are adjustable.

•The Adaline and Madaline layers have fixed weights and bias of 1.

•Training can be done with the help of Delta rule.

Multiple Adaptive Linear Neuron (Madaline)

The architecture of Madaline consists of “n” neurons of the input layer, “m” neurons of the Adaline layer, and 1 neuron of the Madaline layer. The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.



Kohonen Neural Network

Self Organizing Map (or Kohonen Map) is a type of Artificial Neural Network which is also inspired by biological models of neural systems. It follows an unsupervised learning approach and trained its network through a competitive learning algorithm.

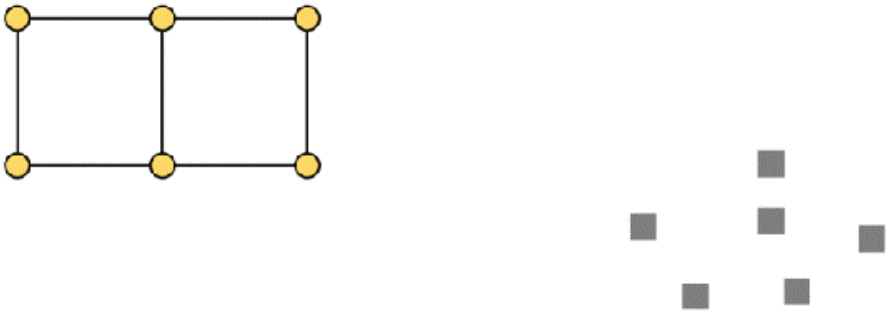
Kohonen NN is used for clustering and mapping techniques to map multidimensional data onto lower-dimensional which allows people to reduce complex problems for easy interpretation. It comprises of two layers, one is the Input layer and the other one is the Output layer. The objective is to map input vectors of arbitrary dimension to discrete map comprised of neurons. The map needs to be trained to create its own organization of the training data.

When training the map, the location of the neuron remains constant but the weights differ depending on the value.

Kohonen Neural Network

In the first phase: every neuron value is initialized with a small weight and the input vector.

In the second phase: the neuron closest to the point is the 'winning neuron' and the neurons connected to the winning neuron will also move towards the point like in the graphic below



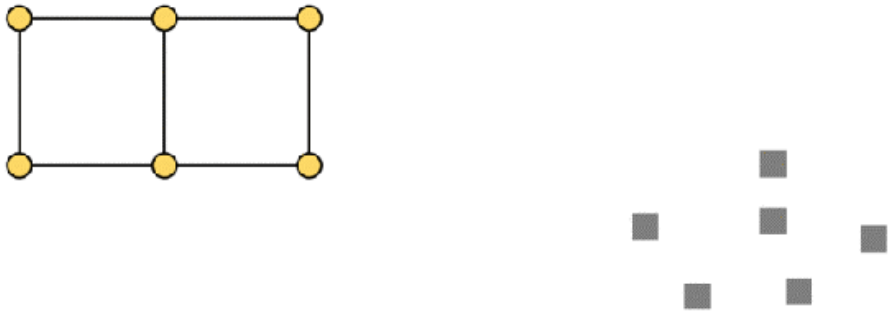
Step 0: Position neurons (orange) in data space.

BMU – Best matching unit

The BMU is selected according to the similarity, between the current input values and all the nodes in the grid.

Kohonen Neural Network

The distance between the point and the neurons is calculated by the Euclidean distance, the neuron with the least distance wins. Through the iterations, all the points are clustered and each neuron represents each kind of cluster. The similar nodes grouped together towards one area, and dissimilar ones separated.



Step 0: Position neurons (orange) in data space.

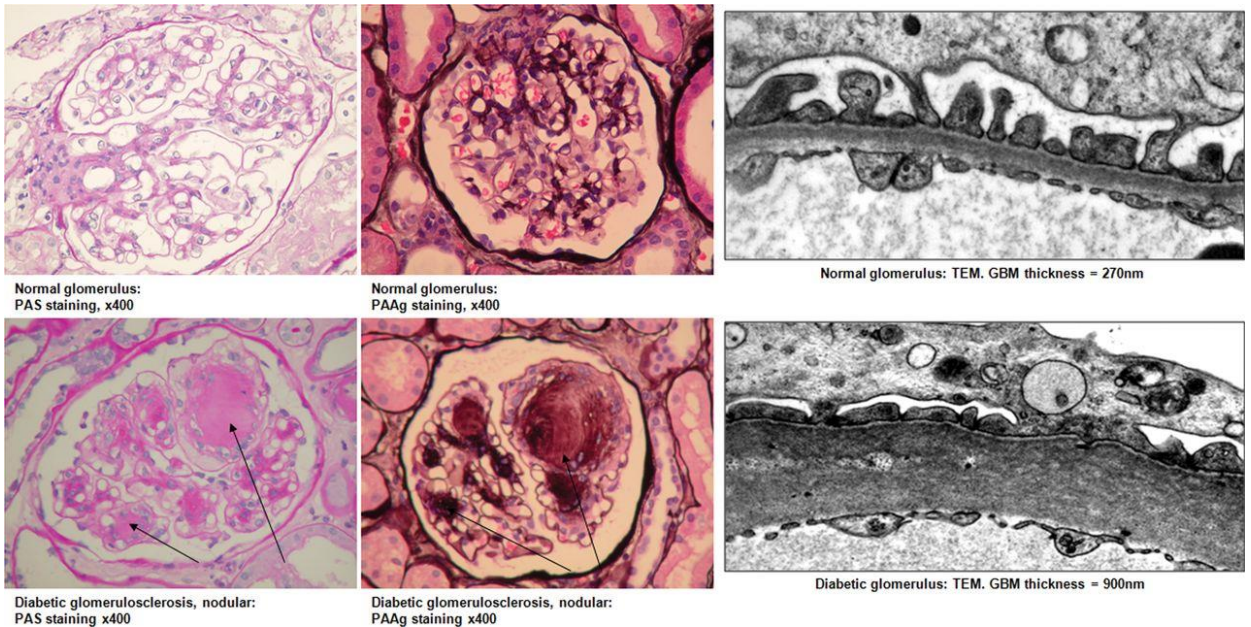
– Best matching unit

Kohonen Neural Network

Kohonen Neural Network is used to recognize patterns in the data. Its application can be found in medical analysis to cluster data into different categories.

Kohonen map was able to classify patients having glomerular or tubular with an high accuracy.

Image displaying a comparison between a healthy and a diseased glomerular using Kohonen Neural Network



AI vs Neural Network

	AI	Neural Network
Meaning	It is a layer of neural networks that smart machines processes.	It is a system of artificial nodes that are used together to solve problems.
Nature	It refers to machines possessing their own intelligence.	It mimics the intelligence that an human brain possesses.
Dependency	It is dependent on artificial neural networks.	It is not dependent on AI.
Applications	It is used in machine learning, machine vision, knowledge reasoning, clinical diagnosis, and much more.	It is used for categorization, classification, pattern recognition, language processing, named entity recognition, and much more.
Training	It can be trained very quickly.	It takes a relatively longer duration to train neural networks.
Performance	It shows very high performance.	It shows low performance.

Application of ANN

•**Aerospace:** Aircraft component fault detectors and simulations, aircraft control systems, high-performance auto-piloting, and flight path simulations.

•**Automotive**: Improved guidance systems, development of power trains, virtual sensors, and warranty activity analyzers

•**Electronics:** Chip failure analysis, circuit chip layouts, machine vision, non-linear modeling, prediction of the code sequence, process control, and voice synthesis.

•**Manufacturing**: Chemical product design analysis, dynamic modeling of chemical process systems, process control, process and machine diagnosis, product design and analysis, paper quality prediction, project bidding, planning and management, quality analysis of computer chips, visual quality inspection systems, and welding quality analysis.

•**Mechanics:** Condition monitoring, systems modeling, and control.

•**Robotics**: Forklift robots, manipulator controllers, trajectory control, and vision systems.

Application of ANN

Telecommunications: ATM network control, automated information services, customer payment processing systems, data compression, equalizers, fault management, handwriting recognition, network design, management, routing and control, network monitoring, real-time translation of spoken language, and pattern recognition (faces, objects, fingerprints, semantic parsing, spell check, signal processing, and speech recognition).

Banking: Credit card attrition, credit and loan application evaluation, fraud and risk evaluation, and loan delinquencies.

Business Analytics: Customer behavior modelling, customer segmentation, fraud propensity, market research, market mix, market structure, and models for attrition, default, purchase, and renewals

Defense: Counterterrorism, facial recognition, feature extraction, noise suppression, object discrimination, sensors, sonar, radar and image signal processing, signal/image identification, target tracking, and weapon steering.

Application of ANN

Education: Adaptive learning software, dynamic forecasting, education system analysis and forecasting, student performance modeling, and personality profiling.

Financial: Corporate bond ratings, corporate financial analysis, credit line use analysis, currency price prediction, loan advising, mortgage screening, real estate appraisal, and portfolio trading.

Medical: Cancer cell analysis, ECG and EEG analysis, emergency room test advisement, expense reduction and quality improvement for hospital systems, transplant process optimization, and prosthesis design.

Securities: Automatic bond rating, market analysis, and stock trading advisory systems.

Transportation: Routing systems, truck brake diagnosis systems, and vehicle scheduling.

Discrete Hopfield Network

Discrete Hopfield Network is a type of algorithms which is called - ***Autoassociative memories*** (Autoassociative memory networks is a possibly to interpret functions of memory into neural network model) . The idea behind this type of algorithms is very simple. It can store useful information in memory and later it is able to reproduce this information from partially broken patterns.

Example: Image you look at an old picture where you were been for a long time.

Discrete Hopfield Network

By looking at the picture you manage to recognize a few objects or places that make sense to you and form some objects even though they are blurry.

Your brain will start bring out some associations about the place.

With these details that you got from your memory so far other parts of picture start to make even more sense.

Though you don’t clearly see all objects in the picture, you start to remember things and withdraw from your memory some images, that cannot be seen in the picture, just because of those very familiarly-shaped details that you’ve got so far.

This phenomena is called as ***Autoassociative memories***

Discrete Hopfield Network

So, In discrete Hopfield network, Discrete means that network only works with binary vectors. But for this network we wouldn't use binary numbers in a typical form. Instead, we will use bipolar numbers. They are almost the same, but instead of 0 we are going to use -1 to decode a negative state. We can't use zeros. And there are two main reasons for it. The first one is that zeros reduce information from the network weight, later in this article you are going to see it. The second one is more complex, it depends on the nature of bipolar vectors.

The simplest associative memory is just a **sum of outer products** of the ***N*** patterns $\{x_i\}_{i=1}^{N}$ that we want to store (Hebbian learning rule). In classical Hopfield Networks these patterns are polar (binary), i.e. $\mathbf{x_i} \in \{-1,1\}^{\mathbf{d}}$, where ***d*** is the length of the patterns. The corresponding weight matrix ***W*** is:

$$W = \sum_{i}^{N} x_i x_i^T$$

Discrete Hopfield Network

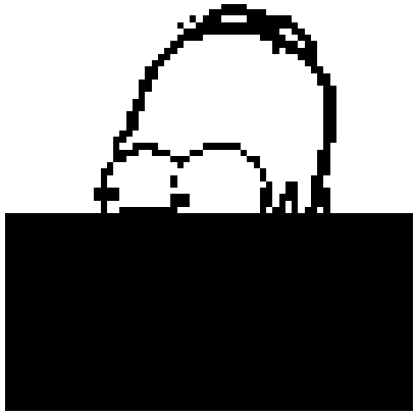
Example: Input image

Converts

Black and white image as an associative memory with polar state and pattern.

Discrete Hopfield Network

Here weight W is added

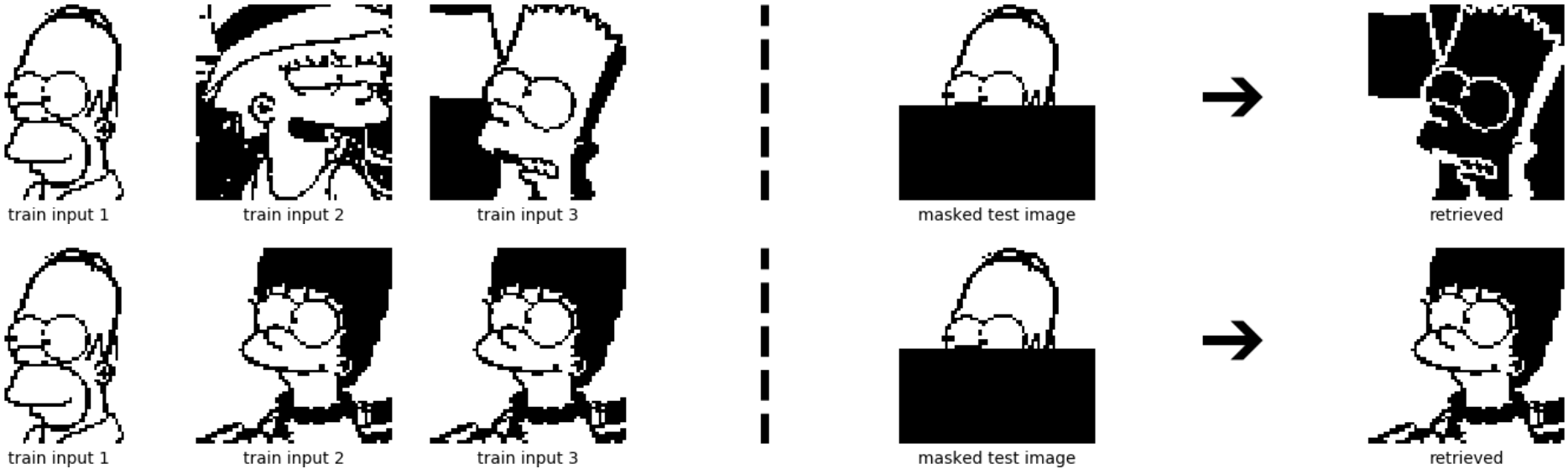


Converts

Original image is restored by minimizing weight

Discrete Hopfield Network

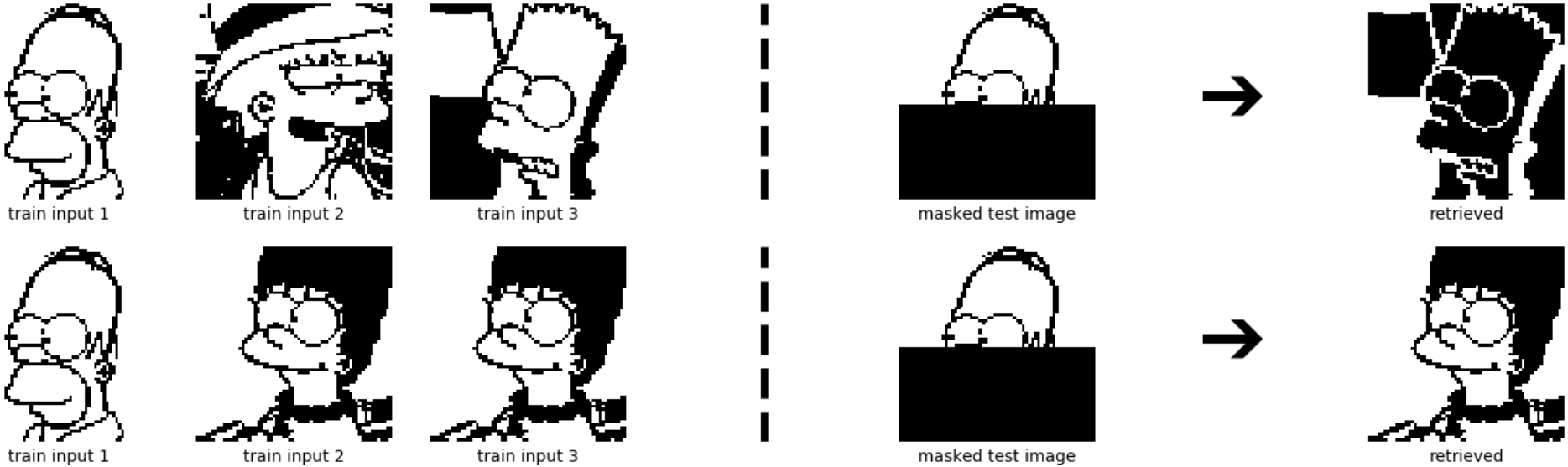
What happens if we store **more than one pattern**? The weight matrix is then built from the sum of outer products of **three stored patterns** (three input images)



the retrieval of the image looks incorrect

Discrete Hopfield Network

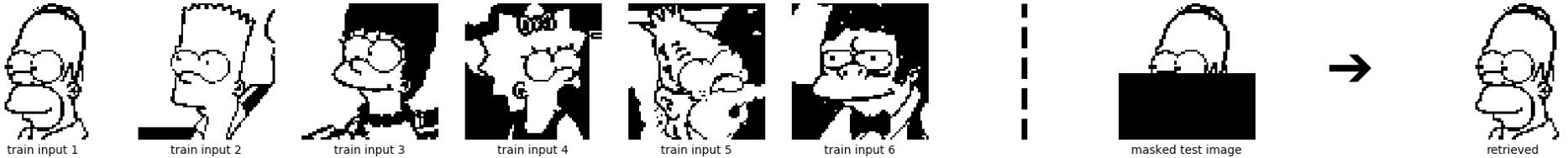
So, we need a model which **allows pulling apart close patterns**, such that (strongly) **correlated patterns can be distinguished**.



Modern Hopfield Network

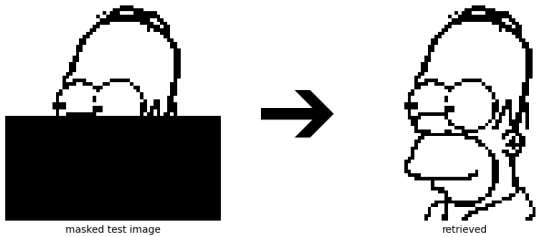
In modern Hopfield Networks, they do not have a weight matrix. Instead, the **energy function** is the ***sum of a function of the dot product*** of every stored pattern $\boldsymbol{X_i}$ with the state pattern.

By using modern Hopfield network help us to retrieve Homer Simpson image out of the 6 stored patterns. Compared to the classical Hopfield Network, it now works smoothly.



Modern Hopfield Network

Compared to the traditional Hopfield Networks, the increased storage capacity now allows pulling apart close patterns. We are now able to distinguish (strongly) correlated patterns, and can retrieve one specific pattern out of many.



Modern Hopfield Network

The most important properties of our new ***energy function*** are:

1. Global convergence to a local minimum: means that all limit points that are generated by the iteration.
2. Exponential storage capacity
3. Convergence after one update step

Here in modern Hopfield network, the SoftMax activation function helps to increase accuracy in pattern recognition.

Boltzmann Machine (BM)

Boltzmann Machine (*named after Ludwig Boltzmann*) is a kind of recurrent neural network where the nodes make binary decisions and are present with certain biases. Several Boltzmann machines can be collaborated together to make even more sophisticated systems such as a deep belief network.

It consists of a network of symmetrically connected, neurons that make decisions stochastically whether to be active or not.

Boltzmann machines are typically used to solve different computational problems such as, for a search problem, the weights present on the connections can be fixed and are used to represent the cost function of the optimization problem.

Boltzmann Machine (BM)

Similarly, for a learning problem, the Boltzmann machine can be presented with a set of binary data vectors from which it must find the weights on the connections so that the data vectors are good solutions to the optimization problem defined by those weights.

Boltzmann machines make many small updates to their weights, and each update requires solving many different search problems.

unsupervised learning methods for this model are considered to be more useful. For examples: *Clustering, Dimensionality reduction, Anomaly detection and Creating generative models.*

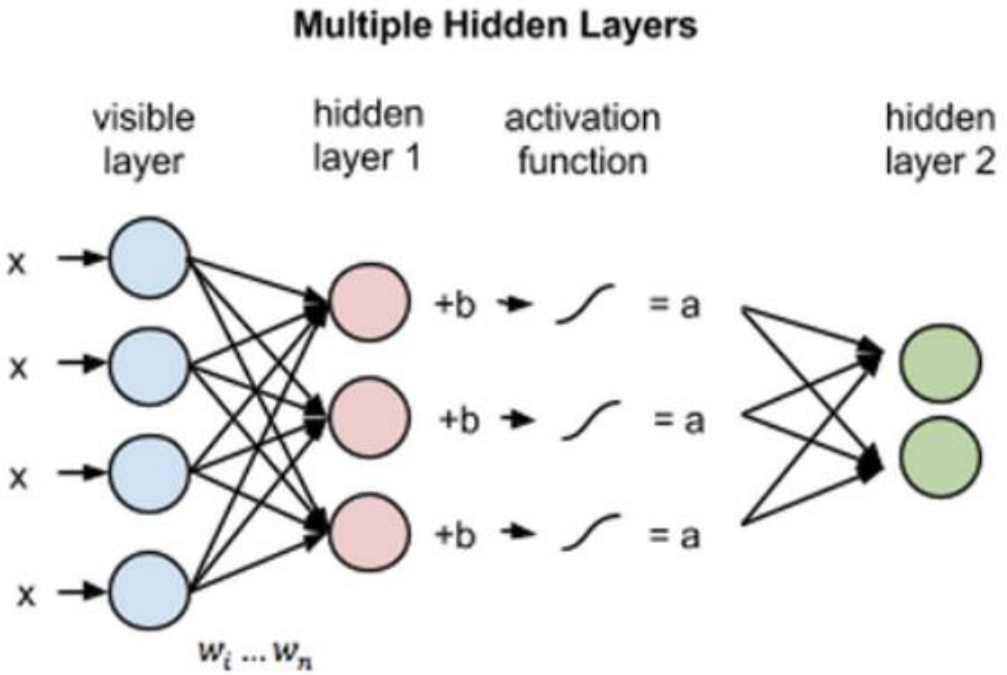
This model can then be trained to get ready to monitor and study abnormal behaviour depending on what it has learnt.

Boltzmann Machine (BM)

The coefficients that modify the inputs are randomly initialized. Each visible node takes a low-level feature from an item present in the dataset to be learned. The result of those two operations is fed into an activation function which in turn produces the node's output also known as the strength of the signal passing through it.

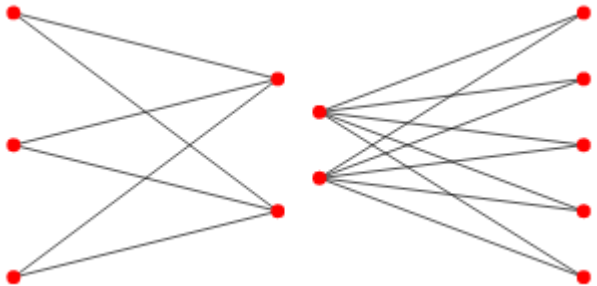
The outputs of the first hidden layer would be passed as inputs to the second hidden layer, and from there through as many hidden layers created, until they reach a final classifying layer.

Learning is typically very slow in Boltzmann machines with a high number of hidden layers



Credit/Debit card fraud detection using Boltzmann Machine (BM)

Restricted Boltzmann machines (RBM) are a special type of Boltzmann Machines that fall under the category of energy based models. It is a generative stochastic model which consists of two layers of visible and hidden layers, they are connected by symmetrical bipartite graph (a bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets and such that every edge connects a vertex in to one in).



bipartite graph

Credit/Debit card fraud detection using Boltzmann Machine (BM)

Restricted Boltzmann machines (RBM) are a special type of Boltzmann Machines that fall under the category of energy based models. It is a generative stochastic model which consists of two layers of visible and hidden layers, they are connected by symmetrical bipartite graph (a bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets and such that every edge connects a vertex in to one in).

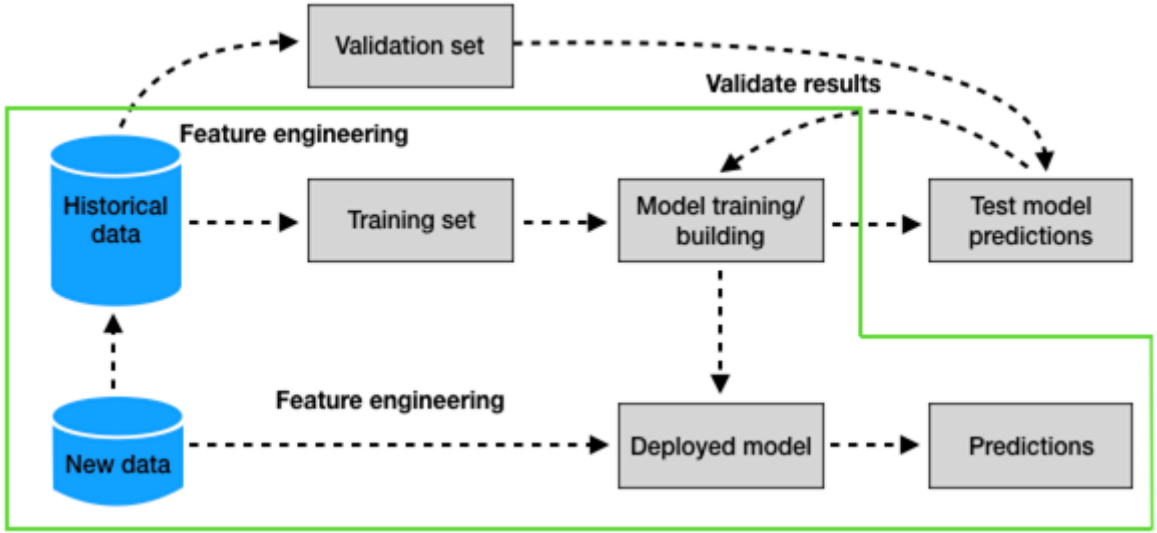
The name “Restricted” came from the fact that there is no intra-layer connection, every node in the visible layer is connected to every node in the hidden layer but no two nodes in the same layer are connected to each other.

The point of RBM is the way in which they learn by themselves for data reconstruction.

RBM uses all transactions that transfer from acquiring bank as visible input and then that goes to the hidden node, and after the calculation of the activation function, the RBM reconstructs the model by transferring the new input from the activation function back to the output or visible function.

Credit/Debit card fraud detection using Boltzmann Machine (BM)

we need to use unsupervised learning because some fraudsters commit frauds once through online mediums and then switch to other techniques also the model learn from data and not from labels

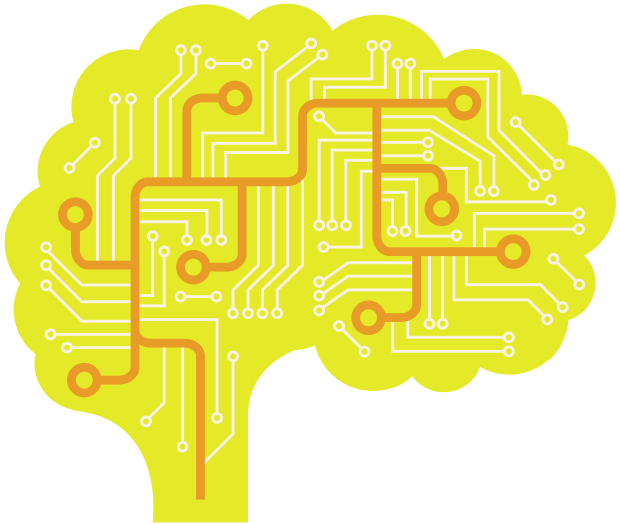


Steps involve in Credit/Debit card fraud detection using Boltzmann Machine (BM) / RBM

1. Import Dataset / using existing dataset.
2. Data Exploration
3. Apply preprocessing and data manipulation on existing dataset
4. Data modelling: split our dataset: 80% of our data will be attributed to the train_data whereas 20% will be attributed to the test data.
5. Apply Logistic Regression, which help us to find outcome probability of a class such as fraud/not fraud, positive/negative.
6. Use Decision Tree Model to plot the outcomes of a decision. These outcomes are basically a consequence through which we can conclude as to what class the object belongs to.
7. Apply ANN, set threshold to 0.5 and check the values above 0.5 will correspond to 1 and the rest will be 0 in fraud detection.
8. Apply Gradient Boosting to perform classification and regression tasks (to boost accuracy).
9. Apply BM/RBM to increase accuracy too.

Research Dimensions in Neural Network

1. Autoencoders based on neural networks: *handwriting recognition tool*
2. Convolutional neural network model: *image processing in self-driving cars*
3. Recurrent neural network model: *chatbots, text mining, video processing*
4. Cryptographic applications using artificial neural networks: *avoiding data leakages in communication*
5. Credit scoring system: *credit risks and classifying applications*
6. Web-based training environment: *User modelling to personalize content for users*
7. Vehicle security system using facial recognition: *identification and authentication of individuals*
8. Automatic music generation: *discovered patterns of rhythm and predict the next tokens*
9. Application for cancer detection: *to classify cancer cells*
10. Text summarizer: *text into a shorter version, like paraphrasing*
11. Intelligent chatbot: *identify the context of the queries and then respond with relevant answers*
12. Human pose estimation project: *Snapchat and Instagram use to fix face filters on a person*
13. Human activity recognition project: *sitting on a chair, falling, picking something up*



Thank You

Dr. Hemraj S. L.
Cryptologist