UNIT-4

Deep Learning: Basics of Deep Learning, Machine Learning Vs Deep Learning, Fundamental Deep Learning Algorithm-Convolution Neural Network (CNN).

Q) Describe the Motivation for Deep Learning.

The simple machine learning algorithms work very well on a wide variety of important problems. However, they have not succeeded in solving the central problems in AI, such as recognizing speech or recognizing objects.Deep learning was designed to overcome these and other obstacles.

Q) Define Deep Learning(DL).

Deep learning is an aspect of artificial intelligence (AI) that is to simulate the activity of the human brain specifically, pattern recognition by passing input through various layers of the neural network.

Deep-learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs.

Q)Give brief historical background of Deep Learning.

All the algorithms that are used in deep learning are largely inspired by the way neurons and neural networks function and process data in the brain. This image is one of the very first pictures of aneuron. It was drawn by Santiago Ramon y Cajal, back in 1899 based on what he saw after placing a pigeon's brain under the microscope. He is now known as the fatherof modern neuroscience.

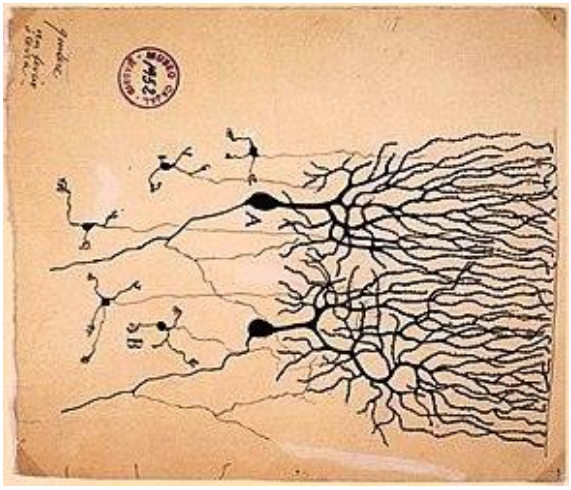


Fig. Human Neural Functioning

It is possible to mimic certain parts of neurons, such as dendrites, cell bodies and axons using simplified mathematical models of what limited knowledge we have on their inner workings: signals can be received from dendrites, and sent down the axon once enough signals were received. This outgoing signal

can then be used as another input for other neurons, repeating the process. Some signals are more important than others and can trigger some neurons to fire easier. Connections can become stronger or weaker, new connections can appear while others can cease to exist.

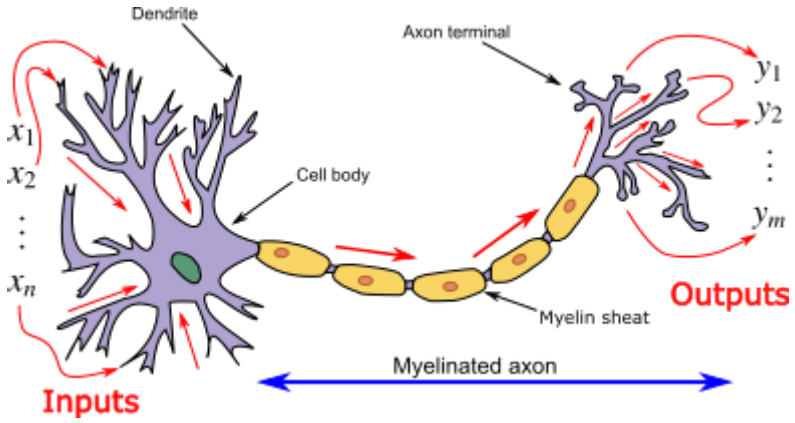


Fig. Biological Neuron

An artificial neuron behaves in the same way as a biological neuron. So it consists of a **soma**(cell body for processing information), **dendrites**(input), and an **axon terminal** to pass on the **output** of this neuron to other neurons. The end of the axon can branch off to connect to many other neurons.

Q) Differentiate a Biological neuron and and Artificial neuron.

Biological neuron	**Artificial neuron**
dendrites	inputs
synapses	weight or inter connection
axon	output
cell body (Soma)	summation and threshold

Q) Define Artificial Neuron(AN). Explain the computation/processing of AN with an example.

Neural Networks are networks used in Deep Learning that work similar to the human nervous system.

An **artificial neuron** is a mathematical function conceived as a model of biological neurons, a neural network. Artificial neurons are elementary units in an artificial neural network.

The artificial neuron receives one or more inputs and sums them to produce an output by applying some activation function.

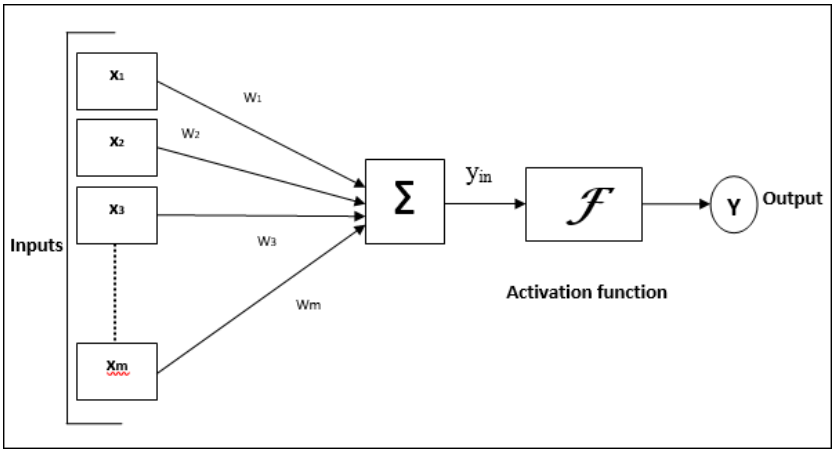


Fig. Artificial Neuron

For the above general model of artificial neural network, the net input can be calculated as follows:

$$y_{in} = (x_1.w_1 + x_2.w_2 + x_3.w_3 \ldots xm.wm) + \text{bias}$$

$$\text{i.e., } y_{in} = \sum_i^m x_i.w_i + \text{bias}$$

where, X_i is set of features and W_i is set of weights.

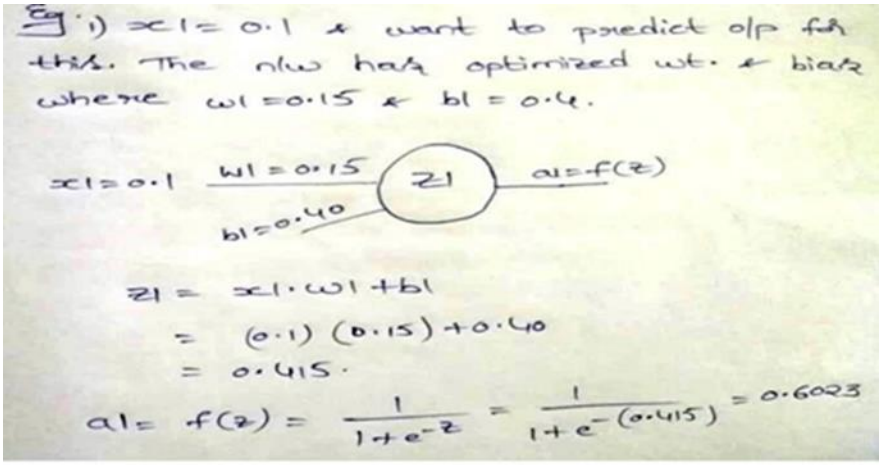
Bias is the information which can impact output without being dependent on any feature.

The output can be calculated by applying the activation function over the net input.

$$Y = F(y_{in})$$

Each AN has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

Eg.

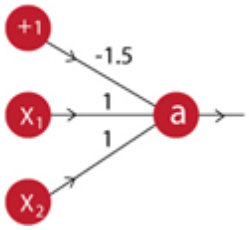


Q) Construct a single layer neural network for implementing OR, AND, NOT gates.

Let us take the activation function as:

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

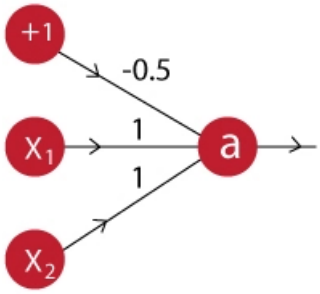
The **AND function** can be implemented as:



The output of this neuron is:**a = f(-1.5*1 + x1*1 + x2*1)**
Calculation for summation:
X1 = 0, x2 = 0 => f(-1.5 + 0 + 0) = f(-1.5) = 0
0 1 =>f(-1.5 + 0 + 1) = f(-0.5) = 0
1 0 =>f(-1.5 + 1 + 0) =f(-0.5) = 0
1 1 =>f(-1.5 + 1 +1) = f(+0.5) = 1
The truth table for this implementation is:

X1	X2	X1 AND X2	(-1.5+X1+X2)	a
0	0	0	-1.5	0
0	1	0	-0.5	0
1	0	0	-0.5	0
1	1	1	0.5	1

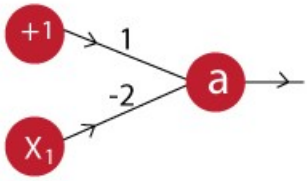
The **OR function** can be implemented as:



The output of this neuron is:a = f(-0.5 + x1 + x2)
The truth table for this implementation is:

X1	X2	X1 OR X2	(-0.5+X1+X2)	a
0	0	0	-0.5	0
0	1	1	0.5	1
1	0	1	0.5	1
1	1	1	1.5	1

The **NOT function** can be implemented as:



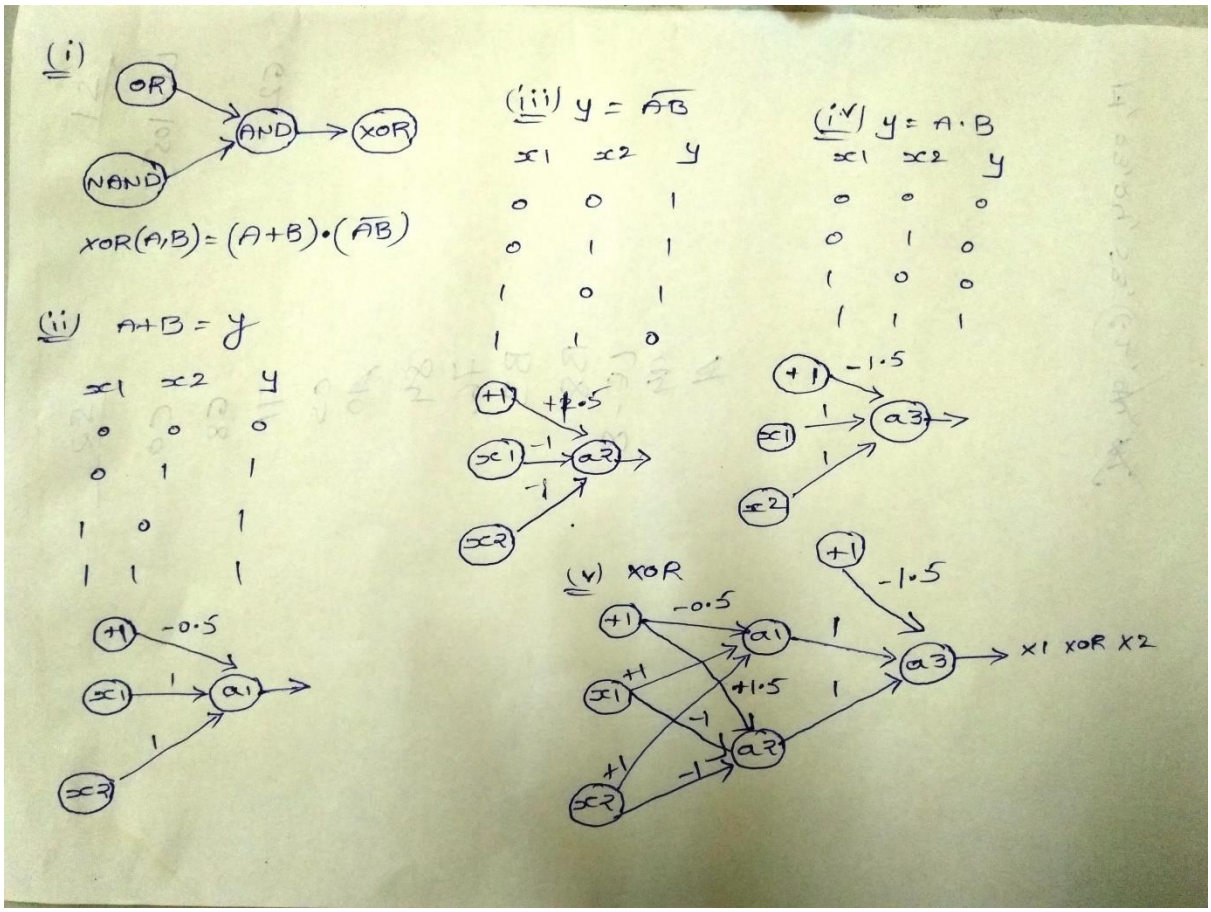
The output of this neuron is:**a = f(1 – 2*x1)**
The truth table for this implementation is:

X1	NOT X1	(1-2*X1)	a
0	1	1	1
1	0	-1	0

Q) Explain the need for multi-layered neural network with an example.

1. **XOR:**

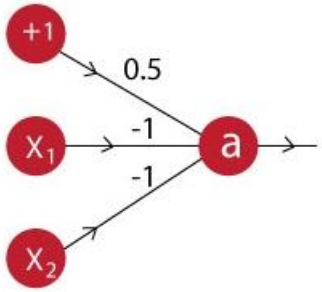
XOR(A,B) = (A+B)*(AB)|
This sort of a relationship cannot be modeled using a single neuron. Thus we will use a multi-layer network.
The idea behind using multiple layers is that complex relations can be broken into simpler functions and combined.



2. XNOR function looks like:

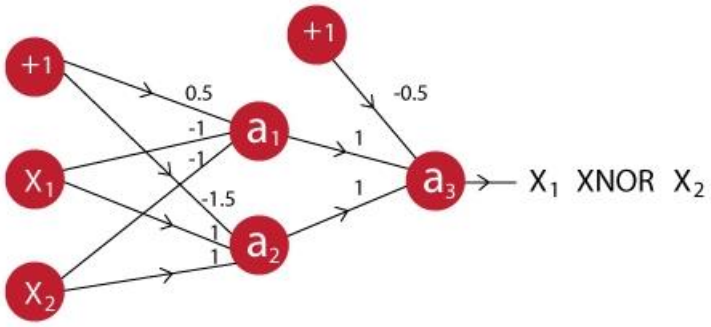
X1	X2	X1 XNOR X2
0	0	1
0	1	0
1	0	0
1	1	1

Lets break down the XNOR function.
X1 XNOR X2 = NOT (X1 XOR X2)
= NOT [(A+B).(A'+B')]
= (A+B)' + (A'+B')'
= (A'.B') + (A.B)
a neuron to model **A'.B'**:



The output of this neuron is:**a = f(0.5 – x1 – x2)**
The truth table for this function is:

X1	X2	X1' AND X2'	(0.5-X1-X2)	a
0	0	1	0.5	1
0	1	0	-0.5	0
1	0	0	-0.5	0
1	1	0	-1.5	0



The different outputs represent different units:
a1: implements A'.B'
a2: implements A.B
a3: implements OR which works on a1 and a2, thus effectively (A'.B' + A.B)
The functionality can be verified using the truth table:

X1	X2	a1	a2	a3
0	0	1	0	1
0	1	0	0	0
1	0	0	0	0
1	1	0	1	1

Q) Define activation function. Explain different types of activation functions.

- Activation Functions are extremely important feature of the Artificial Neural Network. They basically decide whether a neuron should be activated or not. It limits the output signal to a finite value.
- **Activation Function does the non-linear transformation** to the input making it capable to learn more complex relation between input and output. It make the network capable of learning more complex pattern.
- Without an activation function, the neural network is just a linear regression model as it performs only summation of product of input and weights.

Eg. In the below image 2 requires a complex relation which is curve unlike a simple linear relation in image 1.

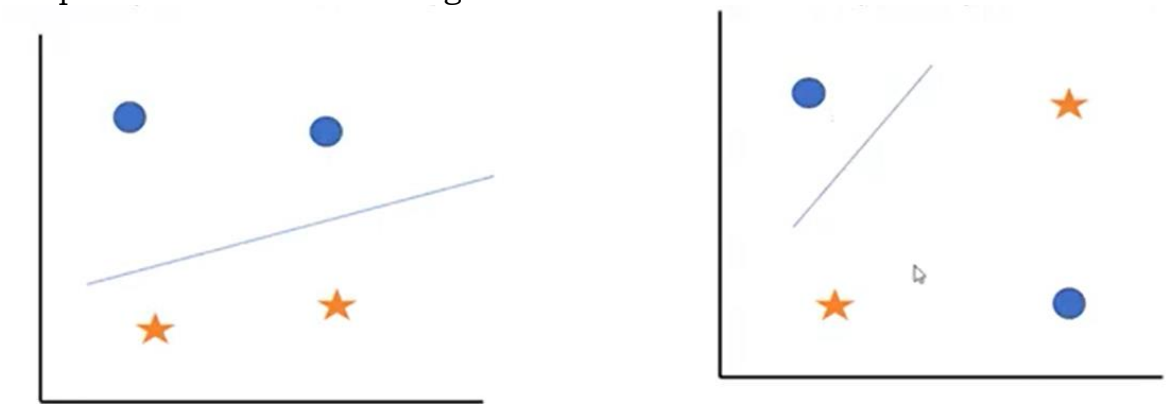


Fig. Illustrating the need of Activation Function for a complex problem.

Activation function must be efficient and it should **reduce the computation** time because the neural network sometimes trained on millions of data points.

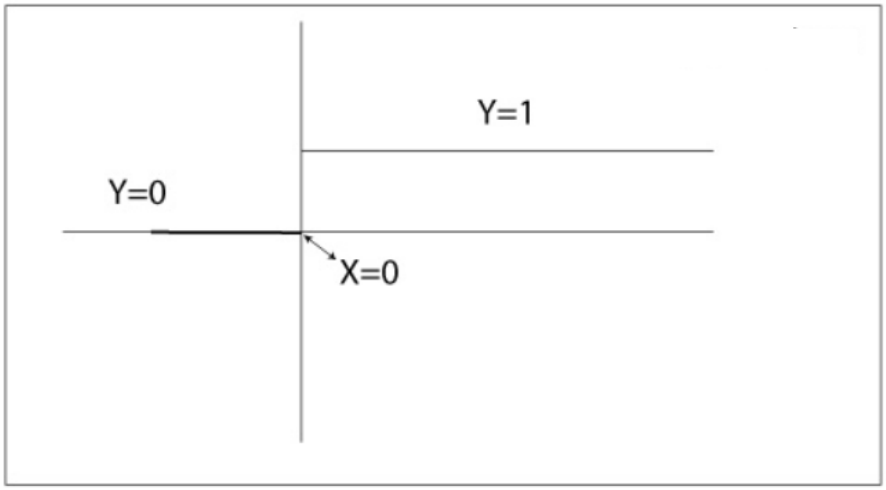
Types of AF:
The Activation Functions can be basically divided into 3 types-

1. Binary step Activation Function
2. Linear Activation Function
3. Non-linear Activation Functions

1. Binary Step Function

A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and

sends exactly the same signal to the next layer.We decide some threshold value to decide output that neuron should be activated or deactivated.It is very simple and useful to classify binary problems or classifier.
Eg.f(x) = 1 if x > 0 else 0 if x <= 0



2. **Linear or Identity Activation Function**

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

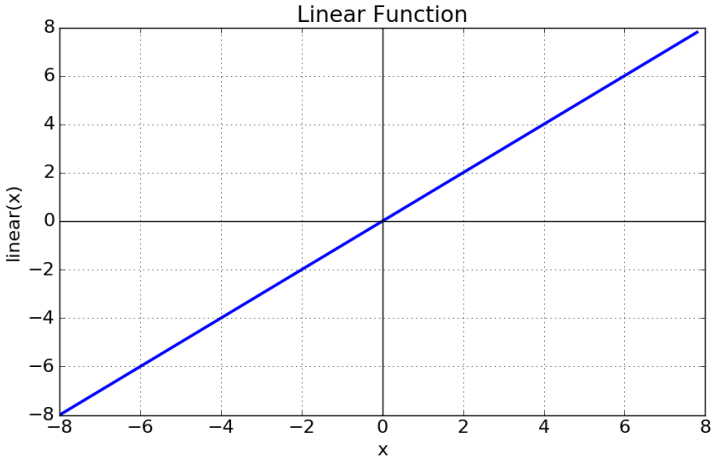


Fig: Linear Activation Function

Equation: f(x) = x
Range : (-infinity to infinity)
It doesn't help with the complexity or various parameters of usual data that is fed to the neural networks

3. **Non-linear Activation Function**

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this.

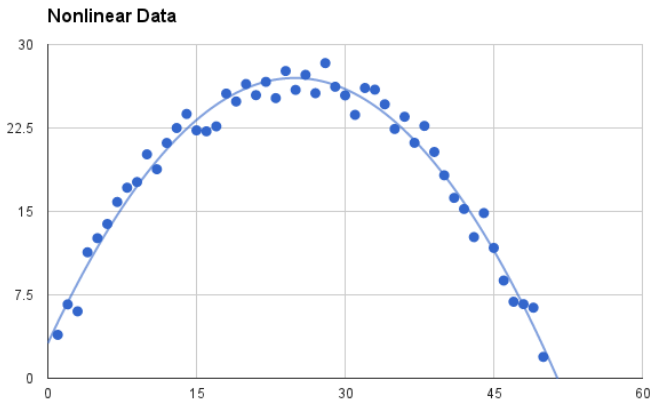


Fig: Non-linear Activation Function

The main terminologies needed to understand for nonlinear functions are:
Derivative or Differential: Change in y-axis w.r.t. change in x-axis.It is also known as slope.
Monotonic function: A function which is either entirely non-increasing or non-decreasing.

The Nonlinear Activation Functions are mainly divided on the basis of their range or curves-

Advantage of Non-linear function over the Linear function :
Differential is possible in all the non -linear function.
Stacking of network is possible, which helps us in creating deep neural nets.
It makes it easy for the model to generalize

3.1 Sigmoid(Logistic AF)(σ):

The main reason why we use sigmoid function is it exists between **0 to 1.**
It is especially used for models where we have to predict the probability as output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

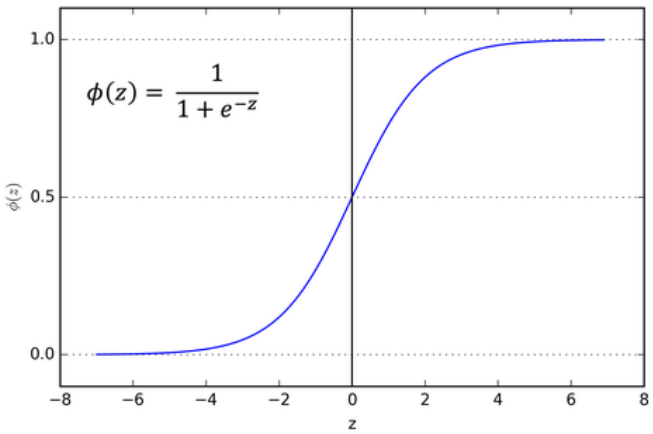


Fig: Sigmoid Function (S-shaped Curve)

The function is **differentiable and monotonic**. But function derivative is not monotonic.
The logistic sigmoid function can cause a neural network to get stuck at the training time.

Advantages

1. Easy to understand and apply
2. Easy to train on small dataset
3. Smooth gradient, preventing "jumps" in output values.
4. Output values bound between 0 and 1, normalizing the output of each neuron.

Disadvantages:

- Vanishing gradient—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- Outputs not zero centered.
- Computationally expensive

3.2 **TanH(Hyperbolic Tangent AF):**

TanH is also like logistic sigmoid but in better way. The range of the TanHfunction is from **-1 to +1.**

TanH is often preferred over the sigmoid neuron because it is zero centred. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in tanh graph.

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

tanh(x) = 2 * sigmoid(2x) - 1

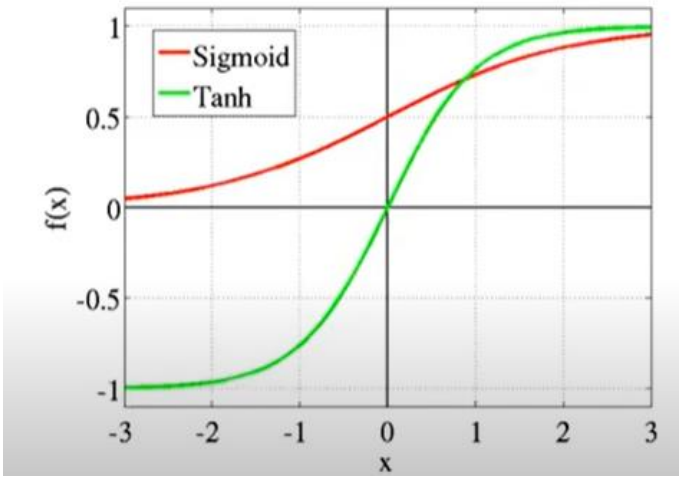


Fig. Sigmoid Vs Tanh

The function is **differentiable** and **monotonic**. But function derivative is not monotonic.

Advantages

- **Zero centered**—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.

Disadvantages

- Like the Sigmoid function is also suffers from vanishing gradient problem
- hard to train on small datasets

3.3 **ReLU(Rectified Linear Unit):**

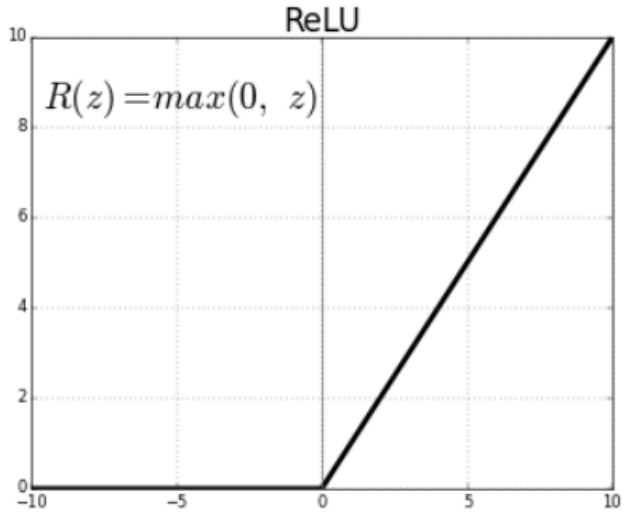
The ReLU is the most used activation function. It is used in almost all convolution neural networks in hidden layers only.
The ReLU is half rectified(from bottom). f(z) = 0, if z < 0
= z, otherwise
R(z) = max(0,z)
The range is **0 to inf.**

Advantages

- **Avoids vanishing gradient problem.**
- **Computationally efficient**—allows the network to converge very quickly
- **Non-linear—**although it looks like a linear function, ReLU has a derivative function and allows for backpropagation

Disadvantages

- Can only be used with a hidden layer
- hard to train on small datasets and need much data for learning non-linear behavior.
- **The Dying ReLU problem**—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.

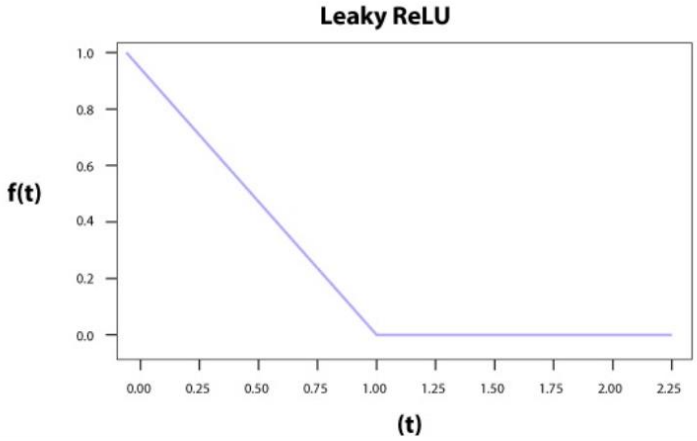


The function and its derivative **both are monotonic**.
All the negative values are converted into zero, and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem.

Leaky ReLU Activation Function

We needed the **Leaky ReLU activation** function to solve the '*Dying ReLU*' problem.

Leaky ReLU we do not make all negative inputs to zero but to a value near to zero which solves the major issue of ReLU activation function.



R(z) = max(0.1*z,z)

Advantages

- **Prevents dying ReLU problem**—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values
- Otherwise like ReLU

Disadvantages

- **Results not consistent**—leaky ReLU does not provide consistent predictions for negative input values.

3.4 **Softmax:**

- Sigmoid able to handle more than two cases(class label).
- Softmax can handle multiple cases. Softmax function squeeze the output for each class between 0 and 1 with sum of them is 1.
- It is ideally used in the final output layer of the classifier, where we are actually trying to attain the probabilities.
- Softmax produces multiple outputs for an input array. For this reason, we can build neural network models that can classify more than 2 classes instead of binary class solution.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

sigma = softmax
z_i = input vector
e^{z_i} = standard exponential function for input vector
K = number of classes in the multi-class classifier
e^{z_i} = standard exponential function for output vector
e^{z_j} = standard exponential function for output vector

Advantages

Able to handle multiple classes only one class in other activation functions—normalizes the outputs for each class between 0 and 1with the sum of the probabilities been equal to 1, and divides by their sum, giving the probability of the input value being in a specific class.

Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

Q) Explain about Deep feedforward networks or feedforward neural networks or multilayer perceptron (MLP).

A deep neural network is a neural network with atleast two hidden layers. Deep neural networks use sophisticated mathematical modeling to process data in different ways.Traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy.

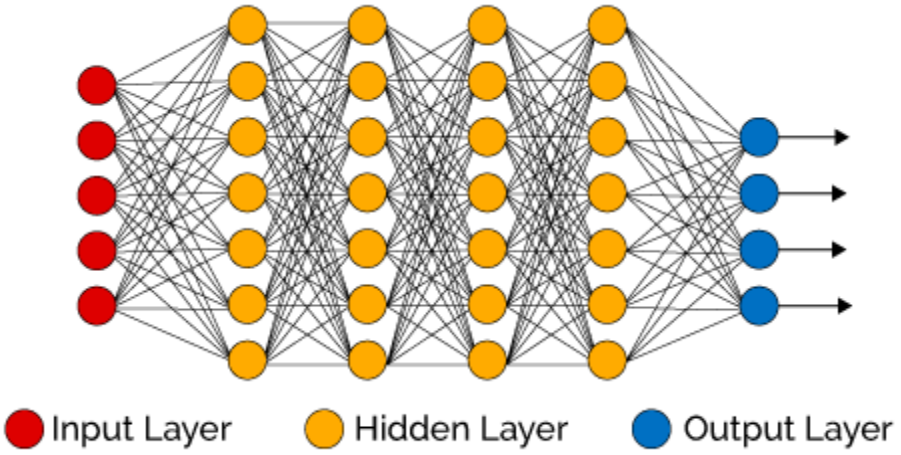
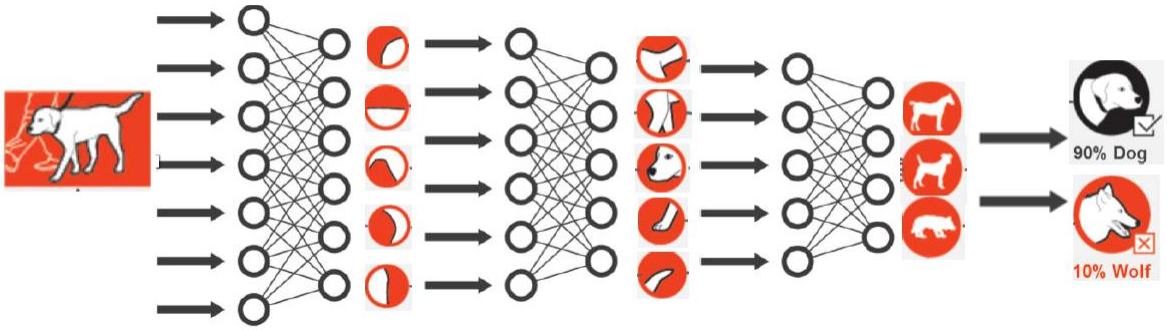


Fig. Deep Feedforward Network

Deep learning creates many layers of neurons, attempting to learn structured representation, layer by layer.



The goal of a feedforward network is to approximate some function f^*. For example,for a classifier, $y = f^*(x)$ maps an input x to a category y.

A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation.

These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations

used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.

When feedforward neural networks are extended to include feedback connections, they are called recurrent neural networks.

Feedforward networks are of extreme importance to machine learning practitioners.They form the basis of many important commercial applications. Forexample, the convolutional networks used for object recognition from photos are aspecialized kind of feedforward network.

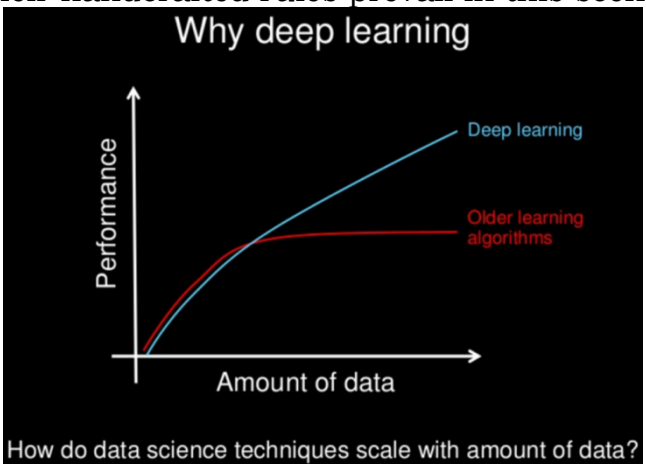
Feedforward neural networks are called networks because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together.

For example, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $\mathbf{f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))}$. This chain structure is most commonly used structure of neural networks. In this case, $f^{(1)}$ is called the first layer of the network called **input layer** used to feed the input into the network; $f^{(2)}$ is called the second layer called **hidden layer** used to train the neural network, and so on. The final layer of a feedforward network is called the **output layer** that provides the output of the network. The overall length of the chain gives the **depth** of the model and **width** of the model is number of neurons in the input layer. It is from this terminology that the name "deep learning" arises.

Q) Differentiate ML & DL.

1. **Data dependencies for Performance:**

When the data is small, deep learning algorithms don't perform that well. This is because deep learning algorithms need a large amount of data to understand it perfectly. On the other hand, traditional machine learning algorithms with their handcrafted rules prevail in this scenario.

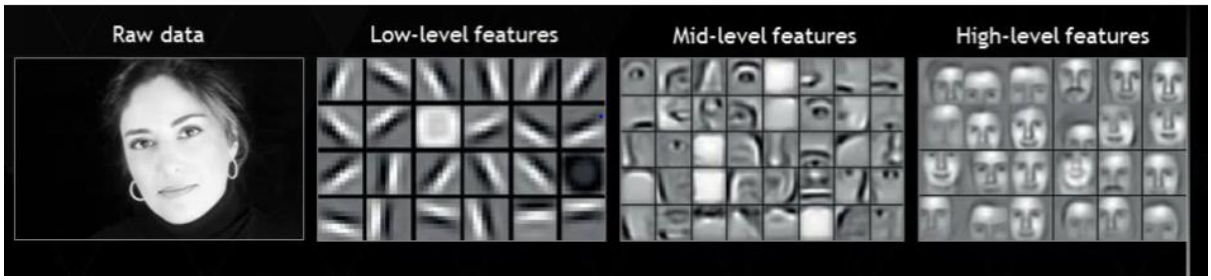


2. **Hardware dependencies**
Deep learning algorithms heavily depend on high-end machines, contrary to traditional machine learning algorithms, which can work on low-end machines. Deep learning algorithms inherently do a large amount of matrix multiplication operations. These operations can be efficiently optimized using a GPU.

3. Feature engineering:
Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.Feature engineering turn your inputs into things the algorithm can understand.

In Machine learning, most of the applied features need to be identified by an expert and then hand-coded as per the domain and data type. Features can be pixel values, shape, textures, position and orientation. The performance of most of the Machine Learning algorithm depends on how accurately the features are identified and extracted.

Deep learning algorithms try to learn high-level features from data. Deep learning reduces the task of developing new feature extractor for every problem. Like, Convolutional NN will try to learn low-level features such as edges and lines in early layers then parts of faces of people and then high-level representation of a face.



4. Problem Solving approach
When solving a problem using traditional machine learning algorithm, it is generally recommended to break the problem down into different parts, solve them individually and combine them to get the result. Deep learning in contrast advocates to solve the problem end-to-end.
Eg. Suppose you have a task of multiple object detection. The task is to identify what is the object and where is it present in the image.
In a typical ML approach, you would divide the problem into two steps, object detection and object recognition
On the contrary, in deep learning approach, you would do the process end-to-end.

5. Execution time
Usually, a deep learning algorithm takes a long time to train. This is because there are so many parameters in a deep learning algorithm that

training them takes longer than usual. Whereas machine learning comparatively takes much less time to train, ranging from a few seconds to a few hours.

This is turn is completely reversed on testing time. At test time, deep learning algorithm takes much less time to run. Whereas, if you compare it with k-nearest neighbors (ML algorithm), test time increases on increasing the size of data. Although this is not applicable on all machine learning algorithms, as some of them have small testing times too.

6. Interpretability:

Suppose we use deep learning to give automated scoring to essays. The performance it gives in scoring is quite excellent and is near human performance. But there's is an issue. It does not reveal why it has given that score. Indeed mathematically you can find out which nodes of a deep neural network were activated, but we don't know what there neurons were supposed to model and what these layers of neurons were doing collectively. So we fail to interpret the results.

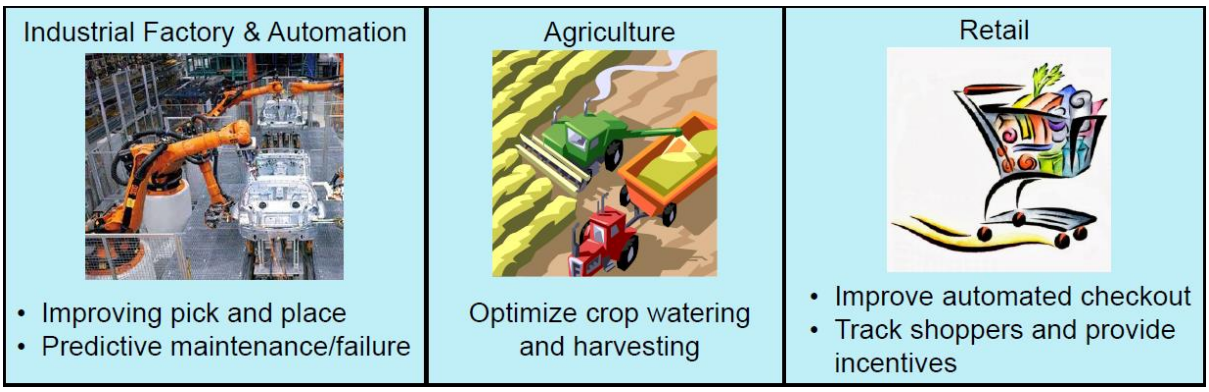
On the other hand, machine learning algorithms like decision trees give us crisp rules as to why it chose what it chose, so it is particularly easy to interpret the reasoning behind it. Therefore, algorithms like decision trees and linear/logistic regression are primarily used in industry for interpretability.

Characteristic	**ML**	**DL**
Data dependencies for Performance	requires less amount of data for identifying rules	requires large amount of data for better performance
Hardware dependencies	work on low-end machines	heavily depend on high-end machines
Feature engineering	features need to be identified by an expert and then hand-coded as per the domain and data type	Deep learning algorithms try to learn high-level features from data. Deep learning reduces the task of developing new feature extractor for every problem.
Problem Solving approach	Break the problem into parts, finds and combines the solution	Solves the problem end-to-end
Execution time	Takes more time for training and less time for testing	Takes much small time for training but may take more time for testing depending on the algorithm like KNN
Interpretability	Fails to interpret the results	Easy to interpret the results

Q) Explain various applications of Deep Learning.

There are various interesting applications for Deep Learning that made impossible things before a decade into reality. Some of them are:

1. Color restoration, where a given image in greyscale is automatically turned into a colored one.
2. Recognizing hand written message.
3. Adding sound to a silent video that matches with the scene taking place.
4. Self-driving cars
5. Computer Vision: for applications like vehicle number plate identification and facial recognition.
6. Information Retrieval: for applications like search engines, both text search, and image search.
7. Marketing: for applications like automated email marketing, target identification
8. Medical Diagnosis: for applications like cancer identification, anomaly detection
9. Natural Language Processing: for applications like sentiment analysis, photo tagging
10. Online Advertising, etc



Q) Briefly explain about loss function in neural networks.

Neural Network uses optimising strategies to minimize the error in the algorithm. The way we actually compute this error is by using a Loss Function. It is used to quantify how good or bad the model is performing. These are divided into two categories i.e. Regression loss and Classification Loss.

1. **Regression Loss Function**

Regression Loss is used when we are predicting continuous values like the price of a house or sales of a company.

Eg. **Mean Squared Error**
Mean Squared Error is the mean of squared differences between the actual and predicted value. If the difference is large the model will penalize it as we are computing the squared difference.

2. **Binary Classification Loss Function**

Suppose we are dealing with a Yes/No situation like "a person has diabetes or not", in this kind of scenario Binary Classification Loss Function is used.
Eg. Binary Cross Entropy Loss
It gives the probability value between 0 and 1 for a classification task. Cross-Entropy calculates the average difference between the predicted and actual probabilities.

3. **Multi-Class Classification Loss Function**

If we take a dataset like Iris where we need to predict the three-class labels: Setosa, Versicolor and Virginia, in such cases where the target variable has more than two classes Multi-Class Classification Loss function is used.
Eg. Categorical Cross Entropy Loss:
These are similar to binary classification cross-entropy, used for multi-class classification problems.

Q) Explain briefly about gradient descent algorithm.
A deep learning neural network learns to map a set of inputs to a set of outputs from training data. We cannot calculate the perfect weights for a neural network.
Gradient descent is an iterative optimization algorithm for finding the minimum of a function.

To find the minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.

The "*gradient*" in gradient descent refers to an error gradient. The model with a given set of weights is used to make predictions and the error for those predictions is calculated.
Eg.

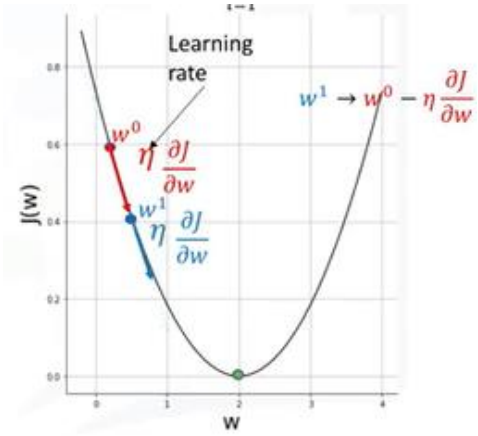


Fig. Gradient Descent

The gradient is given by the slope of the tangent at w = 0.2, and then the magnitude of the step is controlled by a parameter called the learning rate. The larger the learning rate, the bigger the step we take, and the smaller the learning rate, the smaller the step we take. Then we take the step and we move to w1.
Now when choosing the learning rate, we have to be very careful as a large learning rate can lead to big steps and eventually missing the minimum.
On the other hand, a small learning rate can result in very small steps and therefore causing the algorithm to take a long time to find the minimum point.

Q) Explain about Back propagation algorithm.

Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

The algorithm is used to effectively train a neural network through a method called **chain rule.** In simple terms, after each forward pass through a network, back propagation performs a backward pass while adjusting the model's parameters (weights and biases).

Algorithm:

1. Initialize the weights and biases.
2. Iteratively repeat the following steps until defined number of times or threshold value is reached:
   i. Calculate network output using forward propagation.
   ii. Calculate error between actual and predicted values.
   iii. Propagate the error back into the network and update weights and biases using the equations:

$$w^{i+1} \rightarrow w^{i} - \eta \frac{\partial J}{\partial w}$$

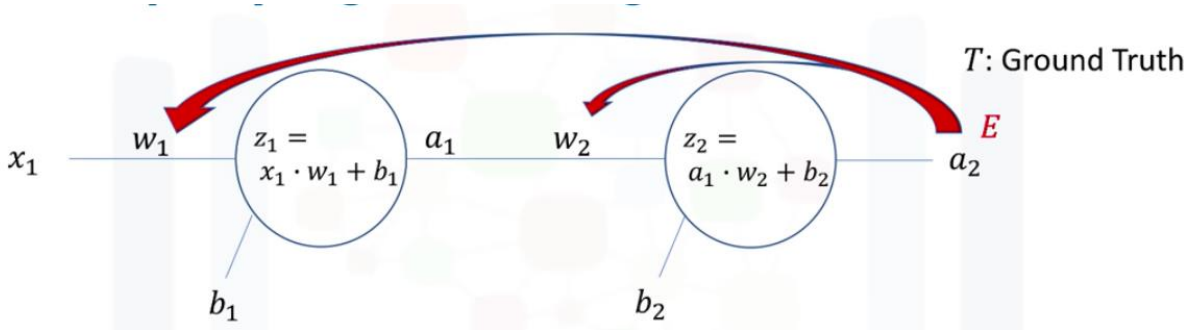
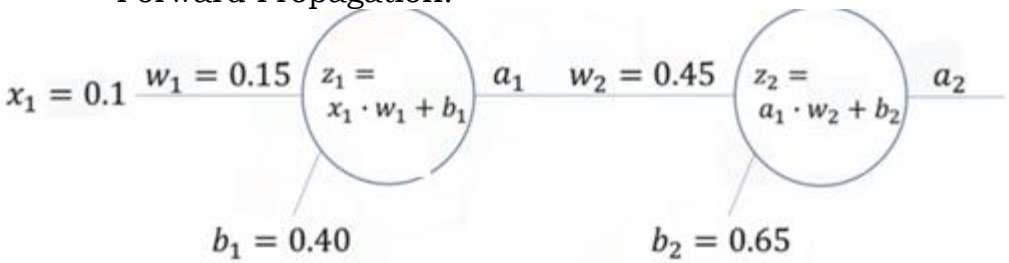


Fig. illustrating BP

Example:

Forward Propagation:



$$z_1 = w_1 \cdot x_1 + b_1 = 0.15 \cdot 0.1 + 0.4 = 0.415$$

$$z_2 = w_2 \cdot a_1 + b_2 = 0.45 \cdot 0.6023 + 0.65 = 0.9210$$

$$a_1 = f(z_1) = \frac{1}{1+e^{-(0.415)}} = 0.6023$$

$$a_2 = f(z_2) = \frac{1}{1+e^{-(0.9210)}} = 0.7153$$

Therefore,

$z_1 = 0.415$ $a_1 = 0.6023$ $z_2 = 0.9210$ $a_2 = 0.7153$

Let us consider,

epochs = 1000 threshold = 0.001

learning rate = 0.4 T = 0.25

4.

$E = 1/2(T-a2)^2 = 0.1083$

Eqn # 1: $z1 = x1 \cdot w1 + b1$

Eqn # 2: $a1 = (z1) = 1/(1+ e^{-z_1})$

Eqn # 3: $z2 = a1 \cdot w2 + b2$

Eqn # 4: $a2 = (z2) = 1/(1+ e^{-z_2})$

Eqn # 5: $E = 1/2 (T - a2)^2$

Updating w2:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2}$$

$$= -(T - a_2) \cdot a_2(1 - a_2) \cdot a_1$$

= 0.45 - 0.4(-(0.25-0.7153))*(0.7153(1-0.7153))*(0.6023)

= 0.45 - 0.4*0.05706

= 0.427

Updating b2:

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2}$$

$$= -(T - a_2) \cdot a_2(1 - a_2) \cdot 1$$

= 0.65 - 0.4*(-(0.25-0.7153))*(0.7153(1-0.7153))*1
= 0.65 - 0.4*0.0948
= 0.612

Updating w1:

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \qquad \#\text{chain rule}$$

$$= -(T - a_2) \cdot a_2(1 - a_2) \cdot w_2 \cdot a_1(1 - a_1) \cdot x_1$$

= 0.15 - 0.4*(-(0.25-0.7153))*(0.7153(1 -0.7153))*0.45*0.6023(1-0.6023)*0.1
= 0.15 - 0.4*0.001021
= 0.1496

Updating w2:

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$$

$$= -(T - a_2) \cdot a_2(1 - a_2) \cdot w_2 \cdot a_1(1 - a_1) \cdot 1$$

= 0.40 - 0.4*(-(0.25-0.7153))*(0.7153(1-0.7153))*0.45*0.6023(1-0.6023)*1
= 0.40-0.4*0.01021
= 0.3959

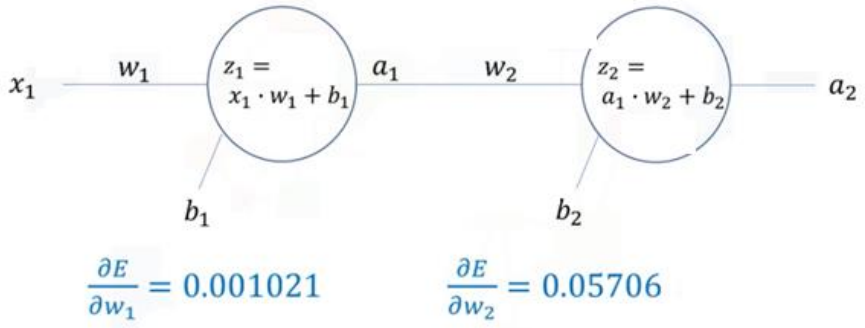
Therefore we continue next iteration(feedforward) with the update values of w_1,b_1,w_2 and b_2.

w_1 = 0.1496 **b_1** = 0.3959 w_2 = 0.427 b_2 = 0.612
x_1 = 0.1.

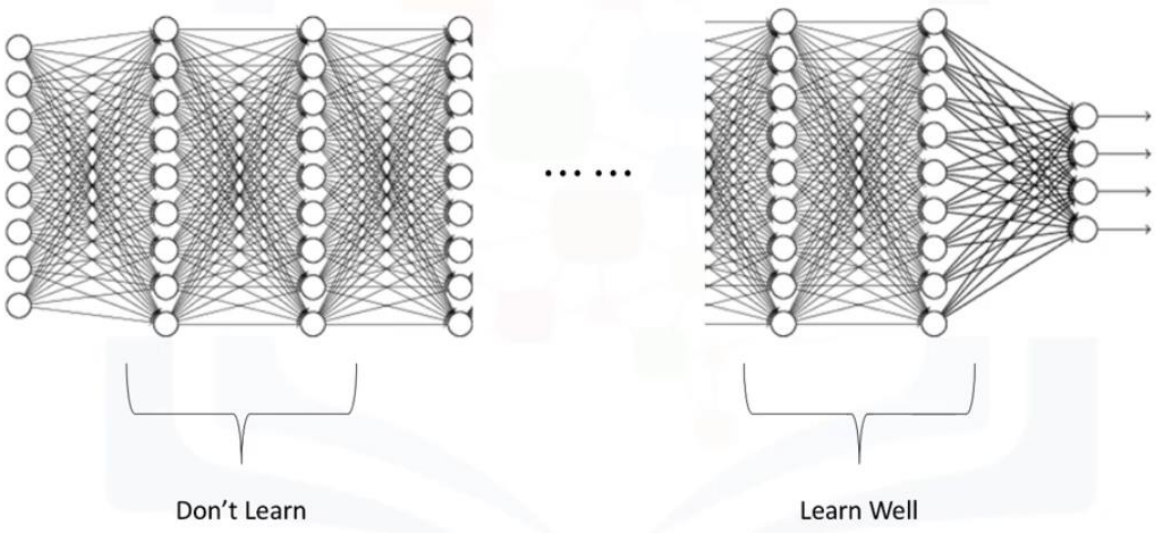
Q) What is Vanishing Gradient problem?

As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train.

Eg. In the below problem the derivatives with respect to weights are very small.



So when we do back propagation, we keep multiplying the factors that are less than 1 by each other and gradients tend to smaller and smaller by moving backward in the network.
This means the neurons in the earlier layers learn very slowly. The result is a training process that takes too long and prediction accuracy is compromised.



Q) Explain indetail about CNN model.

MLP's use one perceptron for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images. For a 224 x 224 pixel image with 3 color channels there are around 150,528 weights that must be trained! As a result, difficulties arise whilst training and overfitting can occur.

A **Convolutional neural network (CNN)** is a neural network that has one or more convolutional layers and is used mainly for image processing, classification, segmentation.

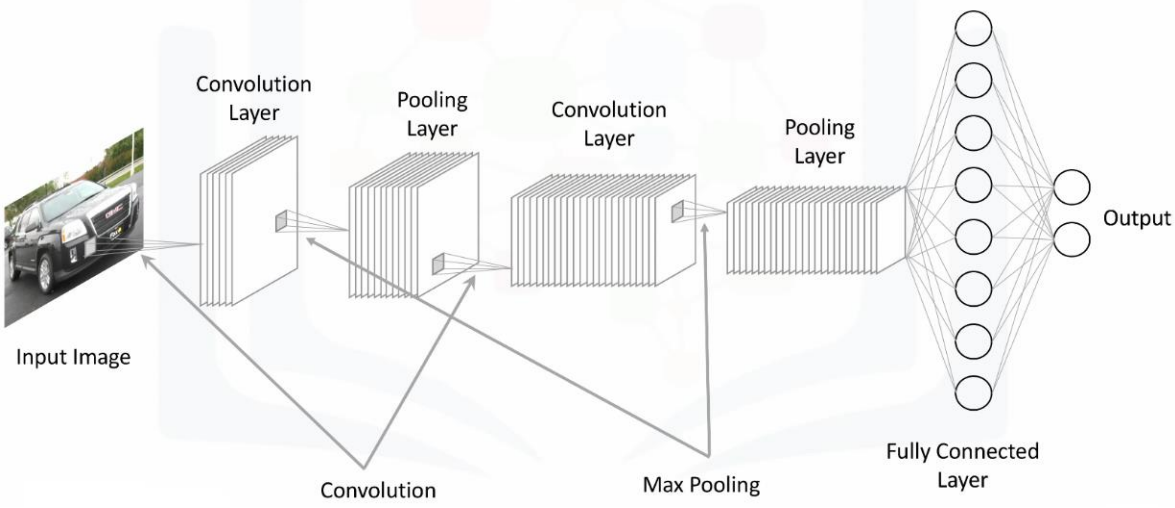


Fig. CNN Architecture

Input layer:
The input to a cnn, is mostly an image(nxmx1-gray scale image and nxmx3-colored image)

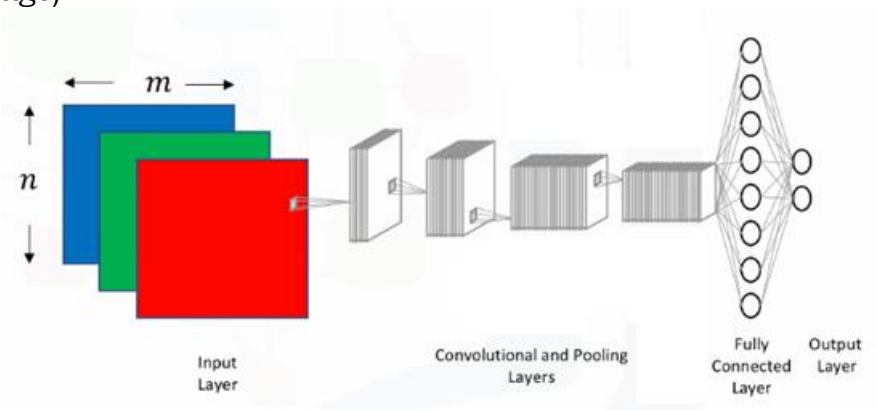


Fig. RGB image as input

Convolution layer:
Here, we basically define filters and we compute the convolution between the defined filters and each of the 3 images.

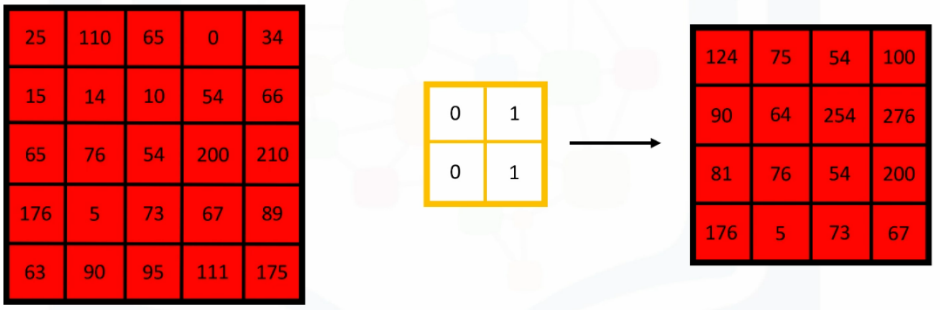


Fig. convolution operation

In the same way we apply to remaining (above is for red image, then we do same for green and blue) images. We can apply more than one filter. More filters we use, we can preserve spatial dimensions better.

We use convolution instead of considering flatten image as input as we will end up with a massive number of parameters that will need to be optimized and computationally expensive.
Eg. We require 25 weights if we take 5x5x1 image with out convolution.
We require 16 weights**(n-f+1 x n-f+1)** if we take 5x5x1 image with 2x2 convolution filter.

By using convolution we can prevent overfitting of the model.

It is worth to have ReLU activation function in convolution layer which passed only positive values and make negative values to zeros.

Pooling layer:
Pooling layer objective is to reduce the spatial dimensions of the data propagating through the network.

1. Max Pooling is the most common, for each section of the image we scan and keep the highest value.

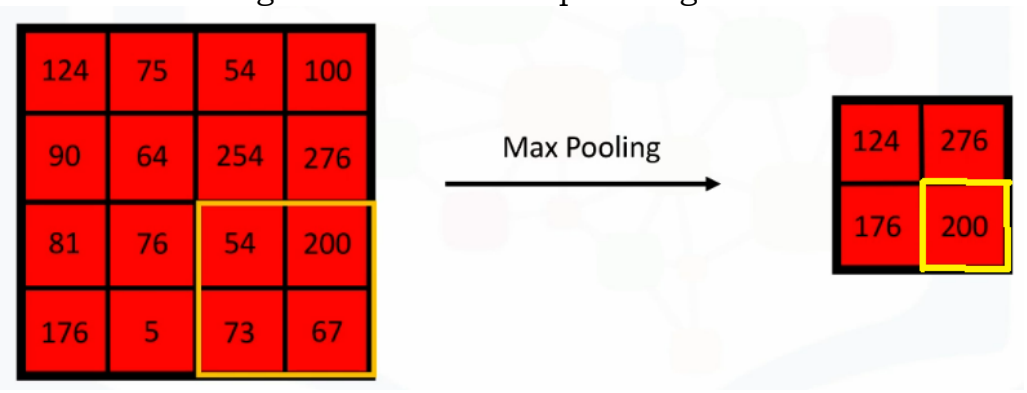


Fig. Max Pooling with stride = 2

Max. pooling provides spatial variance which enables the neural network to recognize objects in an image even if the object does not exactly resemble the original object.

2 Average Pooling: Here, we take average of area we scan.

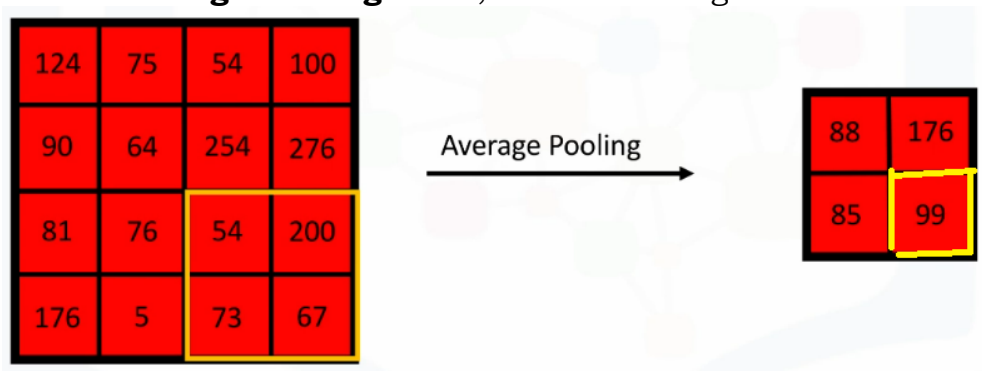


Fig. Average Pooling with stride = 2

Fully Connected Layer:
Here, we flatten the output of the last convolutional layer and connect every node of the current layer with every other node of the next layer.

This layer basically takes output of the preceding layer, whether it is a convolutional layer, ReLU or Pooling layer and outputs an n-dimensional vector, where n is number of classes pertaining to the problem.

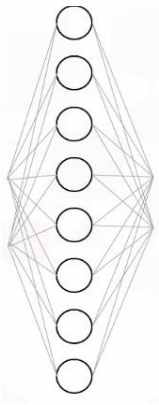


Fig. Fully Connected Layer

Q) Differentiate Shallow NN and Deep NN.

Shallow Neural Network	**Deep Neural Network**
It consists of one hidden layer	It consists of more than one hidden layer
It takes input as vectors only	It takes raw data like images and text as input.