Deep Learning

By
Prof(Dr) Premanand P Ghadekar

Outline

- Introduction
- Basic Concepts
- Artificial Intelligence
- Deep Learning
- Neural Network
- Single & Multilayer Perceptron
- Activation Functions
- Loss Functions
- Optimizer
- Hyper Parameter Tuning
- ANN
- Back Propagation
- RNN etc.

Why Artificial Intelligence

Let's understand this with an example:

If a car exceeds the speed limit, then for a human to monitor and note down all the numbers is not possible.

Why Artificial Intelligence

In order to solve it, we can use a machine to capture the number plate picture and covert it into a text format

What is Artificial Intelligence

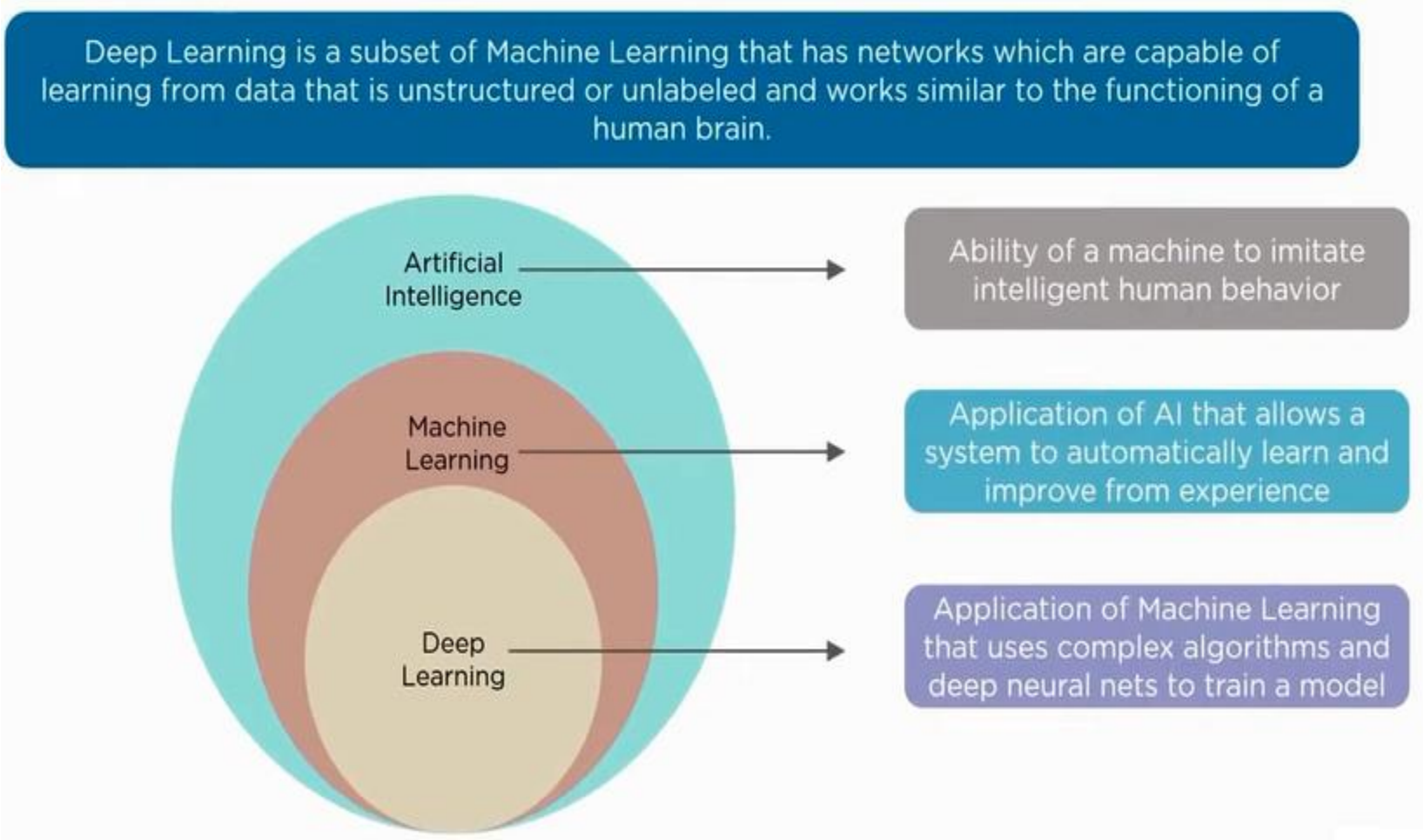
Artificial Intelligence is the capability of a machine to imitate intelligent human behavior.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem

Outcomes of this study is used as a basis of developing intelligent software and systems.

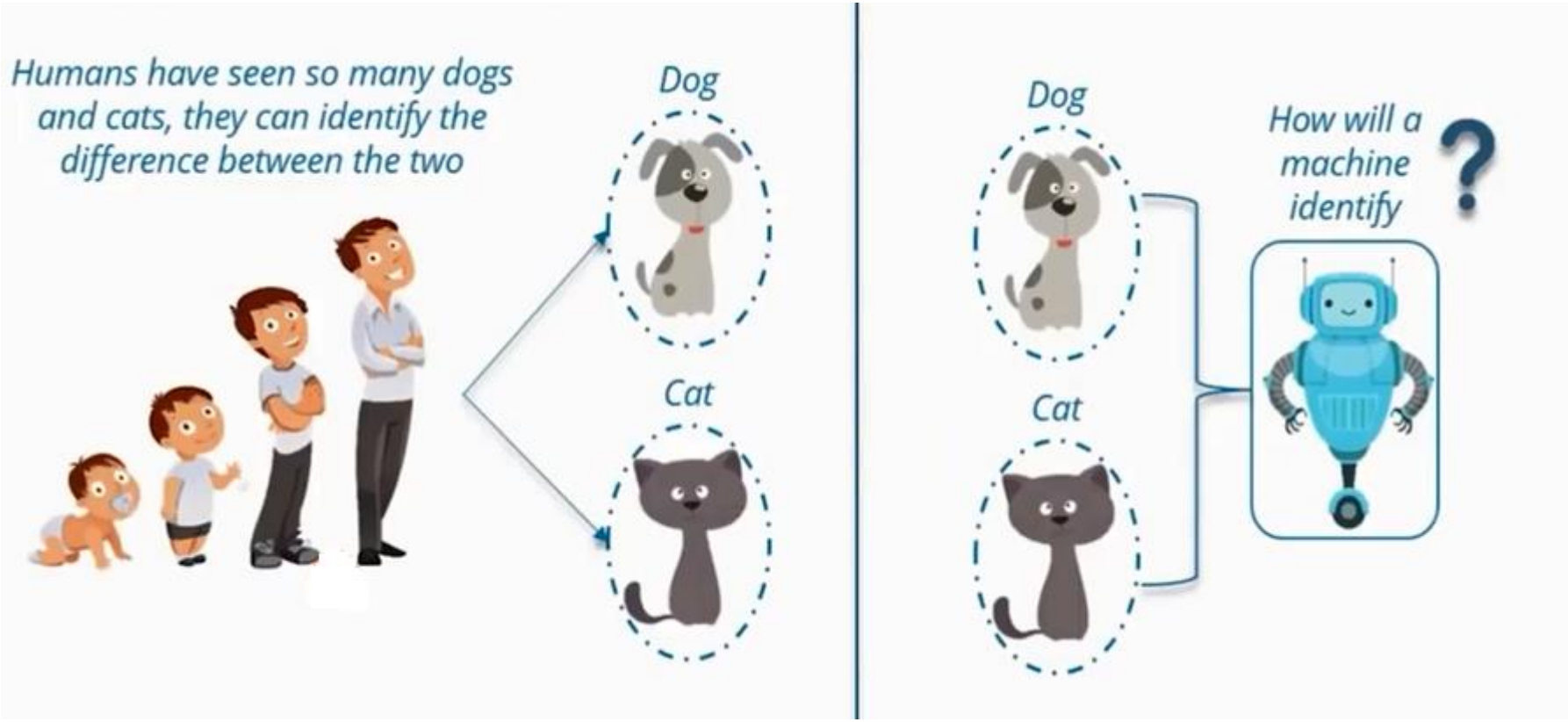
Applications of Artificial Intelligence

Deep Learning

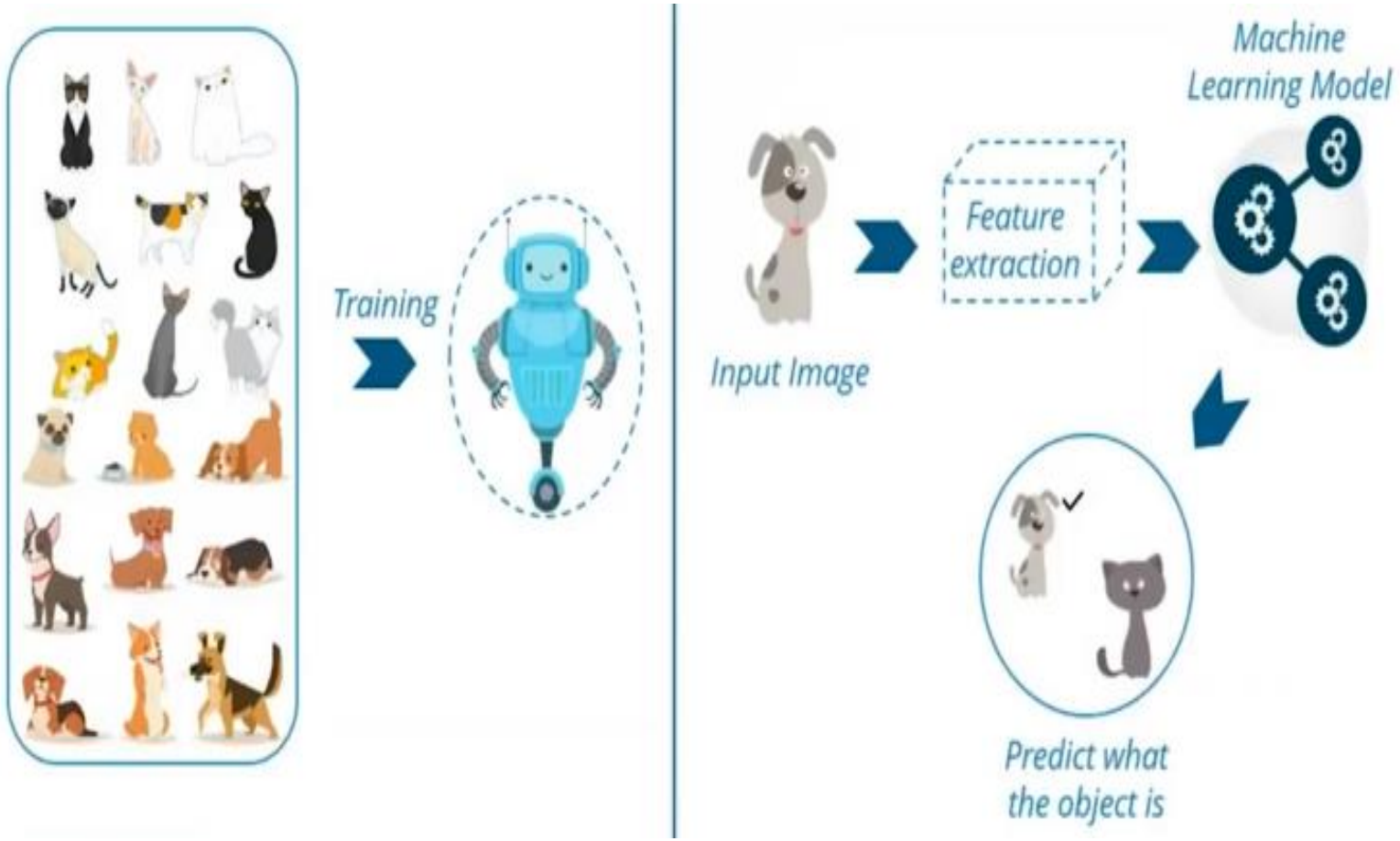


Deep Learning –Techniques that learns features and task from Data Directly
In Deep Learning, features can be learnt just from raw data.

What is Deep Learning

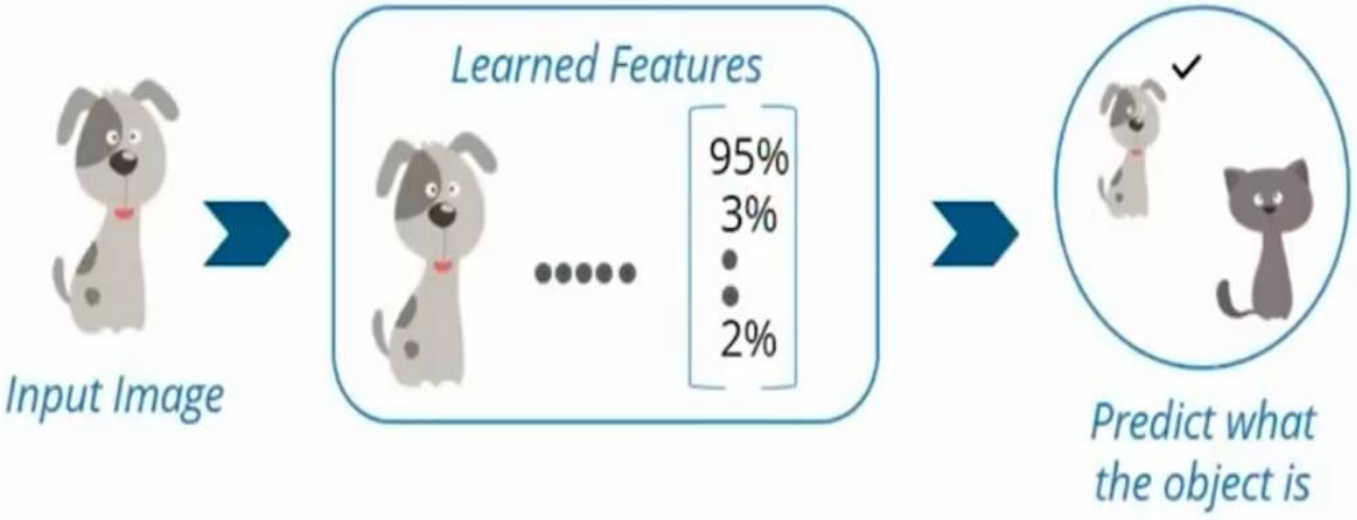


Deep Learning

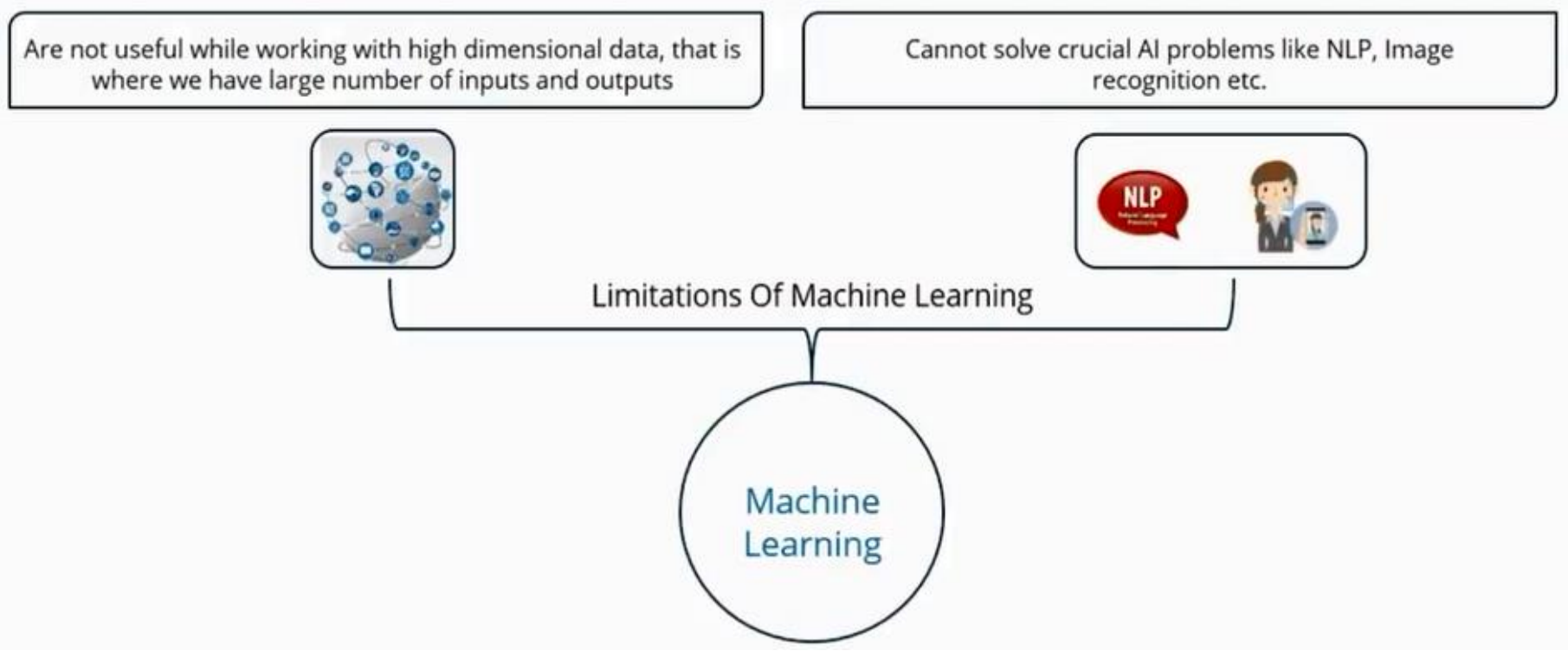


What is Deep Learning

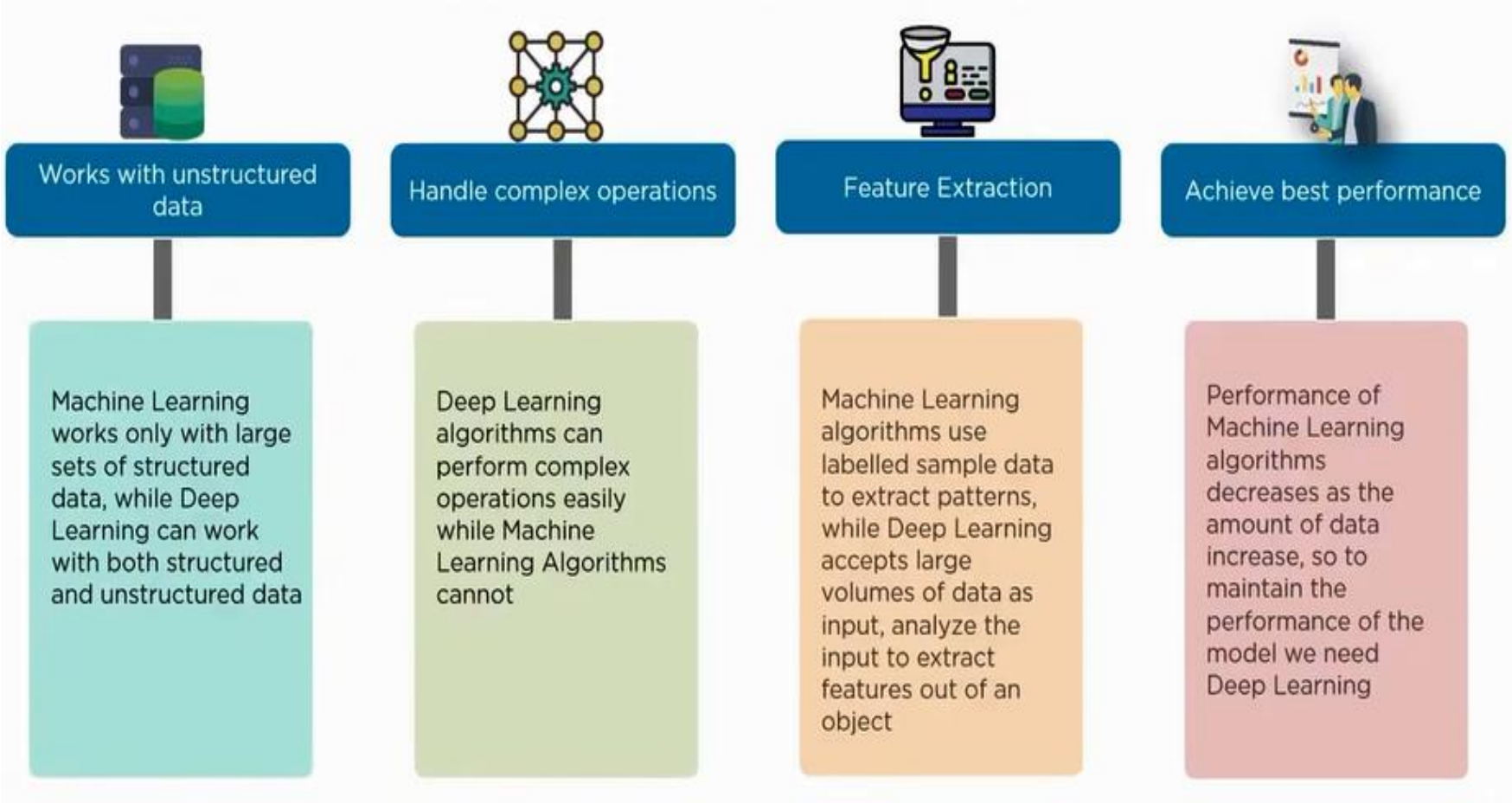
Deep Learning Skips the manual steps of extracting features, you can directly feed images to the Deep learning algorithm, which then predicts the object.



Why do we need Deep Learning

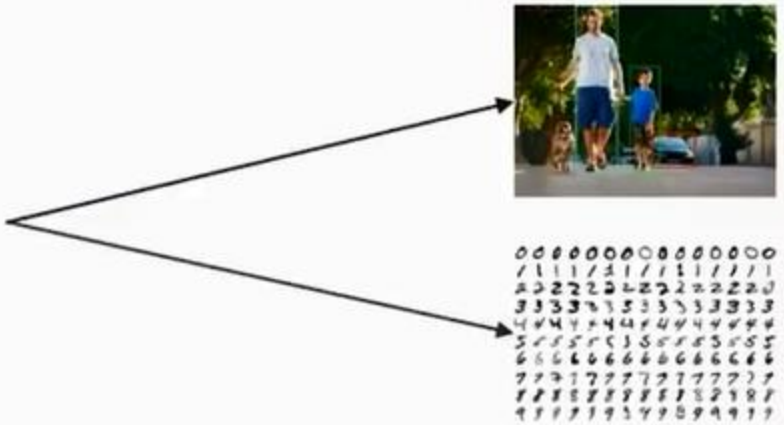


Why do we need Deep Learning



Why do we need Deep Learning

- One of the big challenges with traditional Machine Learning models is a process called feature extraction.
- For complex problems such as object recognition or handwriting recognition, this is a huge challenge.



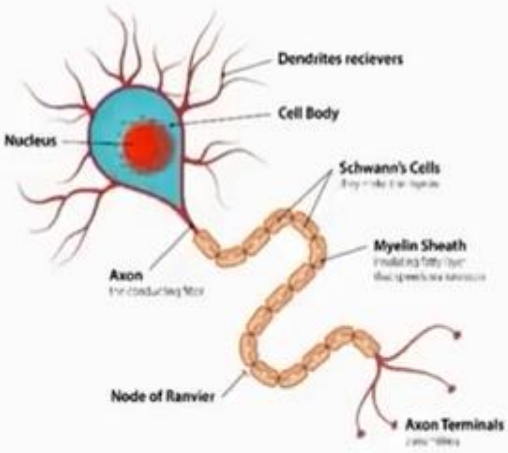
Why do we need Deep Learning

- Deep Learning models are capable to focus on the right features by themselves, requiring little guidance from the programmer.
- These models also partially solve the dimensionality problem.

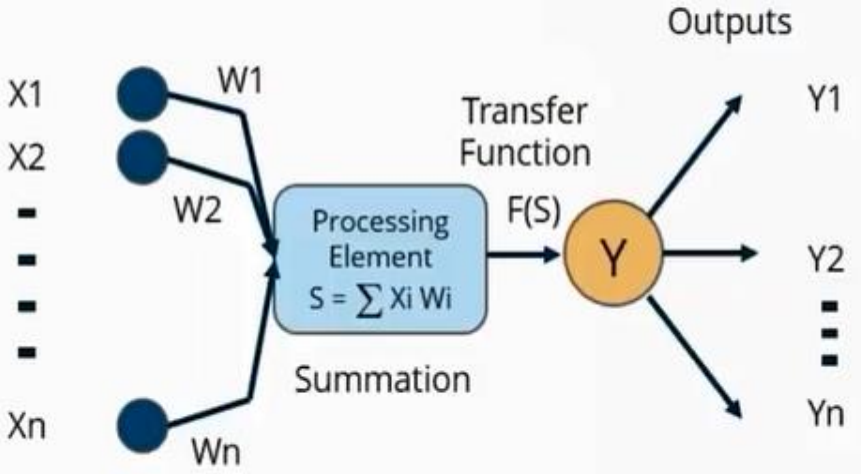
The idea behind Deep Learning is to build learning algorithms that mimic brain.

Why do we need Deep Learning

- Deep Learning is implemented through Neural Networks.
- Motivation behind Neural Networks is the biological Neuron.

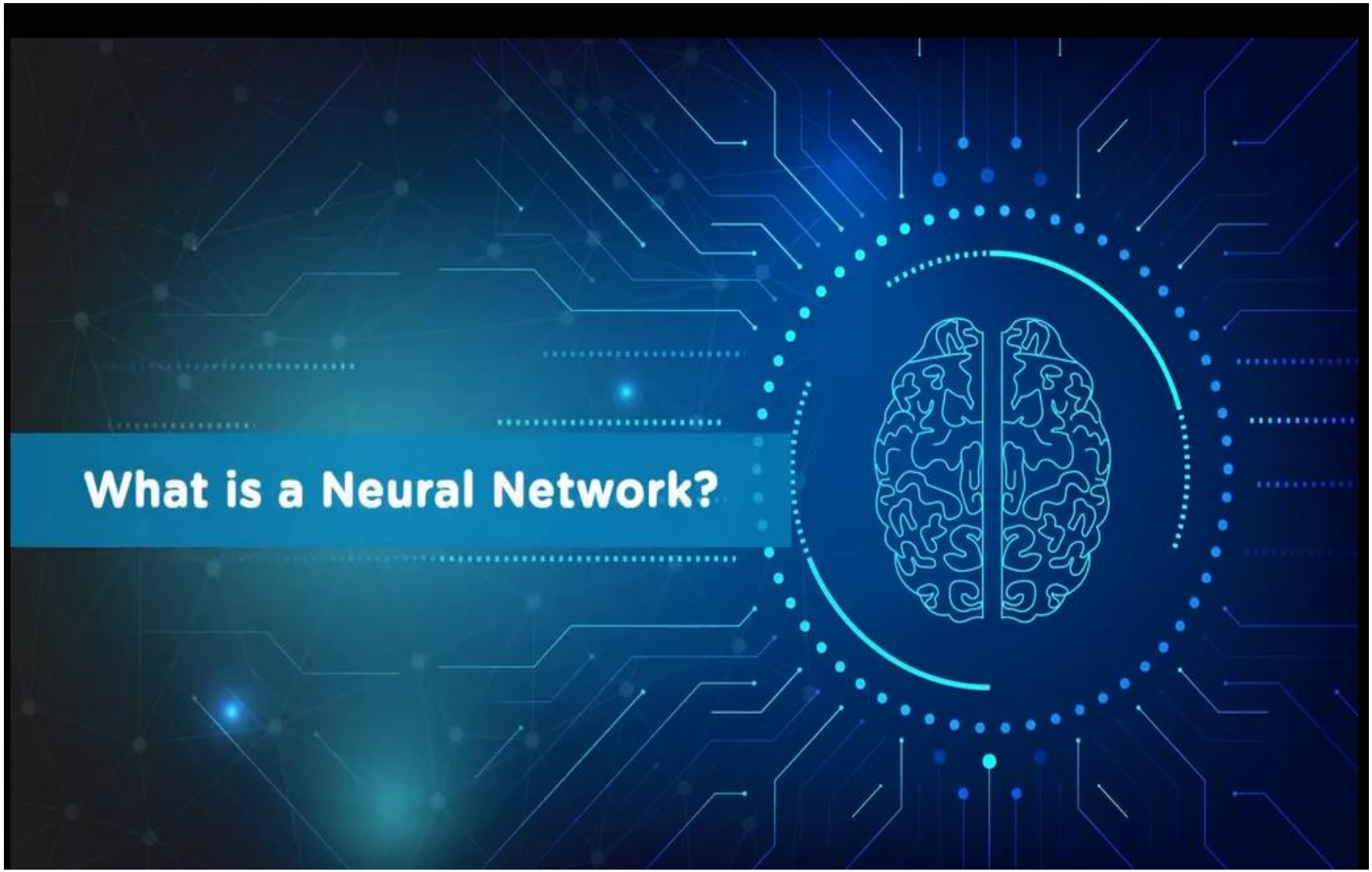


Neuron



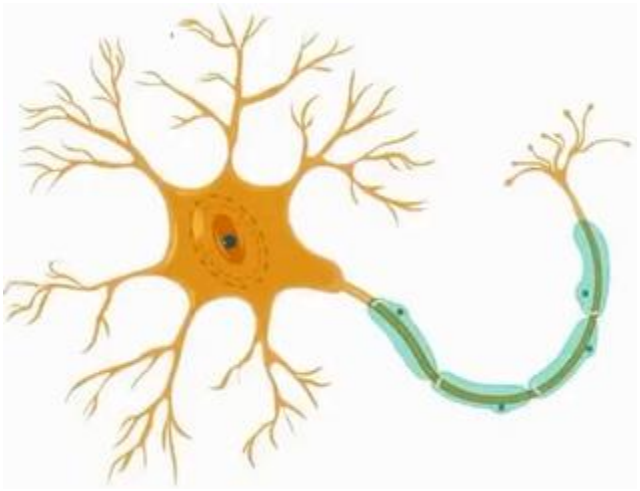
Artificial Neural Network

What is a Neural Network



What is a Neural Network

Neural Networks are modeled after **biological Neural Networks** that allow Computers to learn And interact like humans do. It has **interconnected neurons with dendrites** that receive Inputs , and based on those inputs , it produces an electric signal i.e. **output through the axon**.



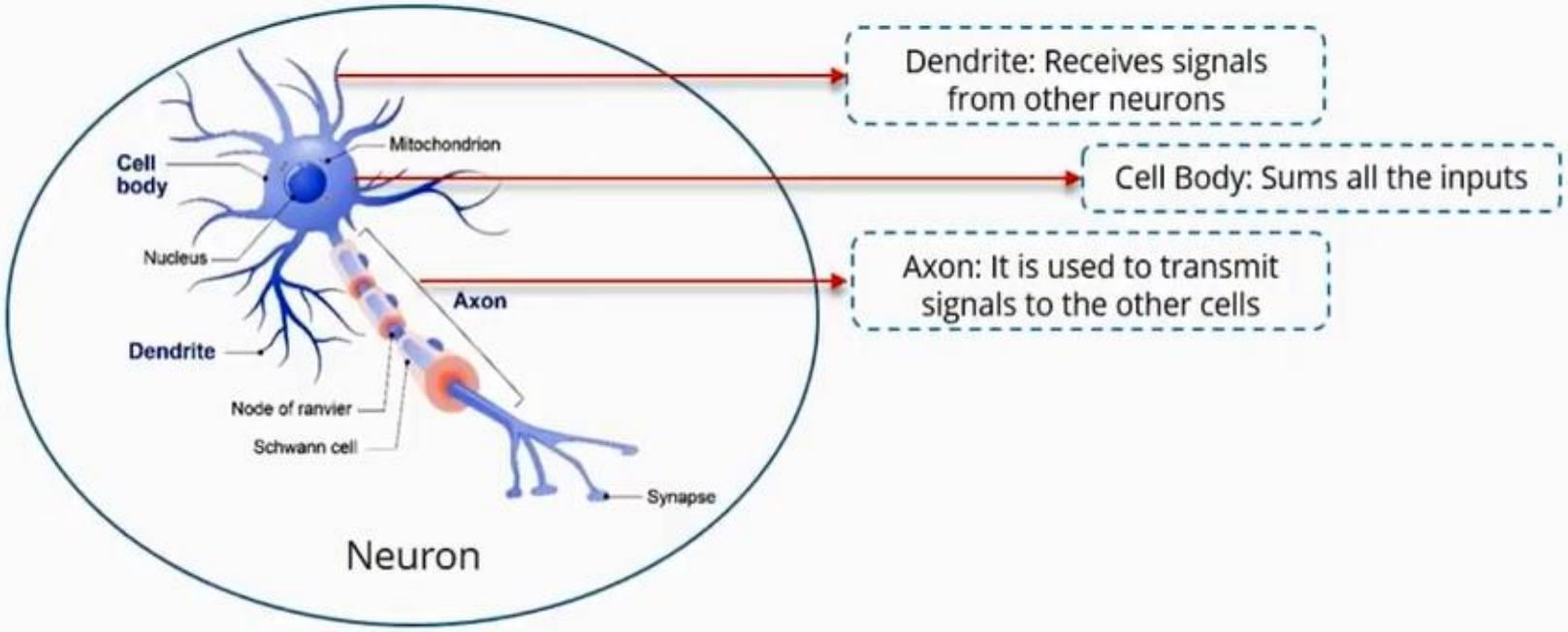
Biological Neuron

Neural Networks

- Take in data as input
- Train themselves to understand pattern in the data
- Output useful predictions

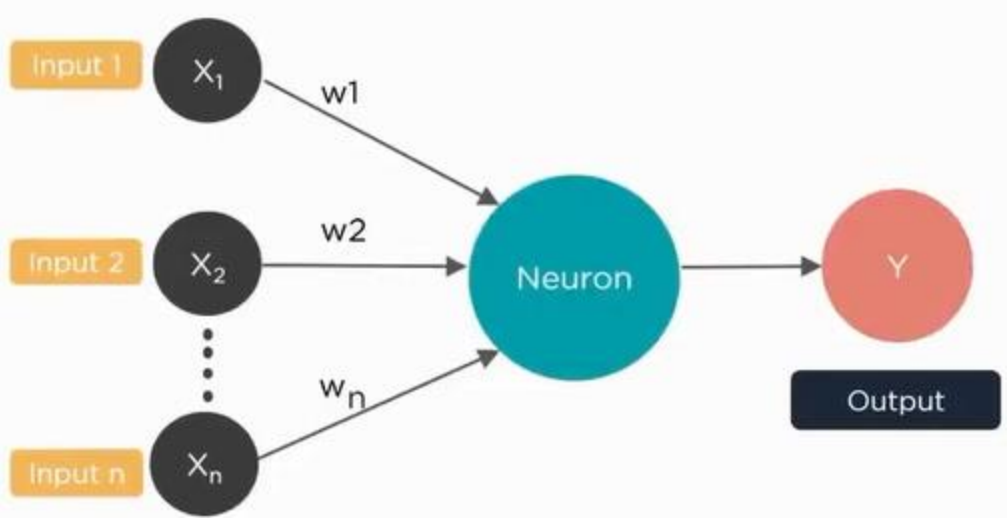
How the Deep Learning works

Deep learning is a form of machine learning that uses a model of computing that's very much inspired by the structure of the brain, so lets understand that first.



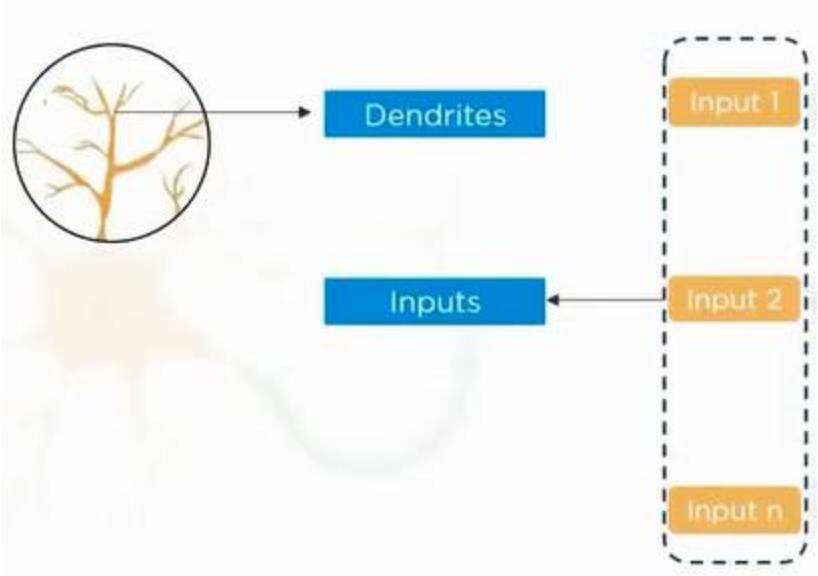
What is a Neural Network

Neural Networks has interconnected neurons that receive some inputs, Processes those inputs in layer to produce the output



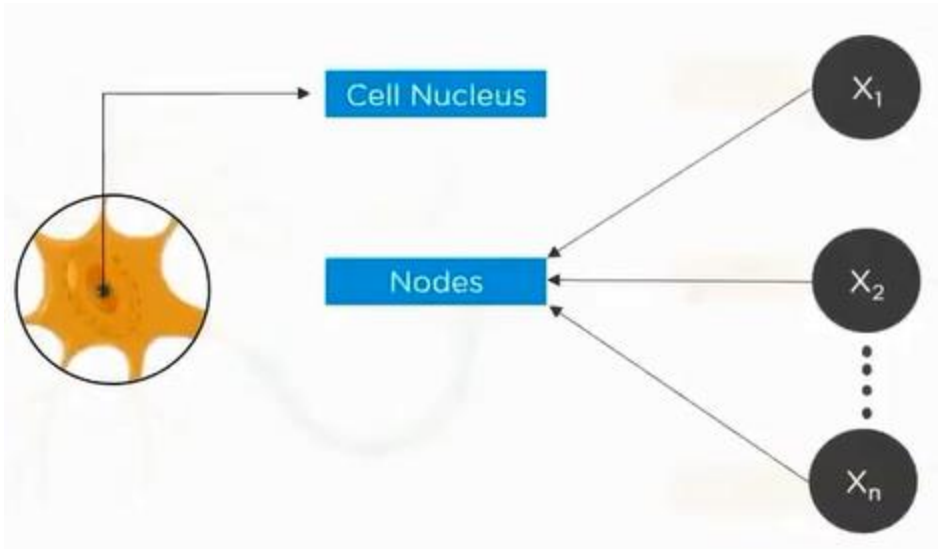
Biological Neurons Vs Artificial Neurons

Dendrites------→ Inputs



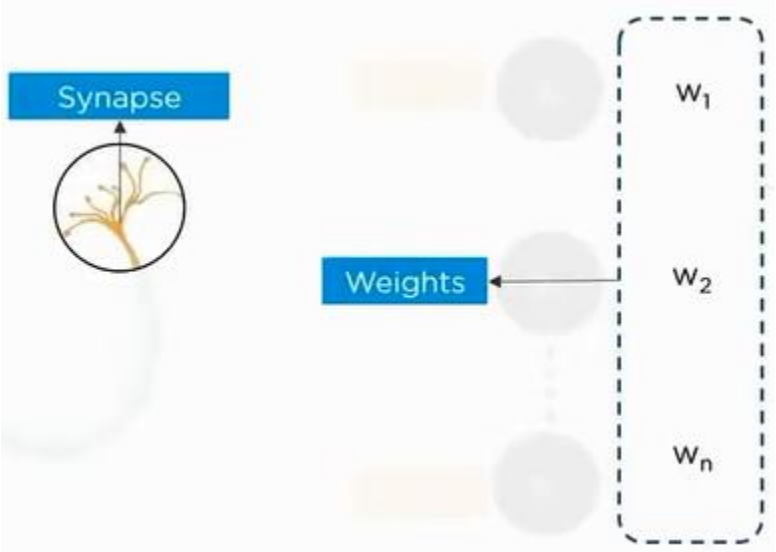
Biological Neurons Vs Artificial Neurons

Cell Nucleus ------ → Nodes



Biological Neurons Vs Artificial Neurons

Synapse ------ → Weights



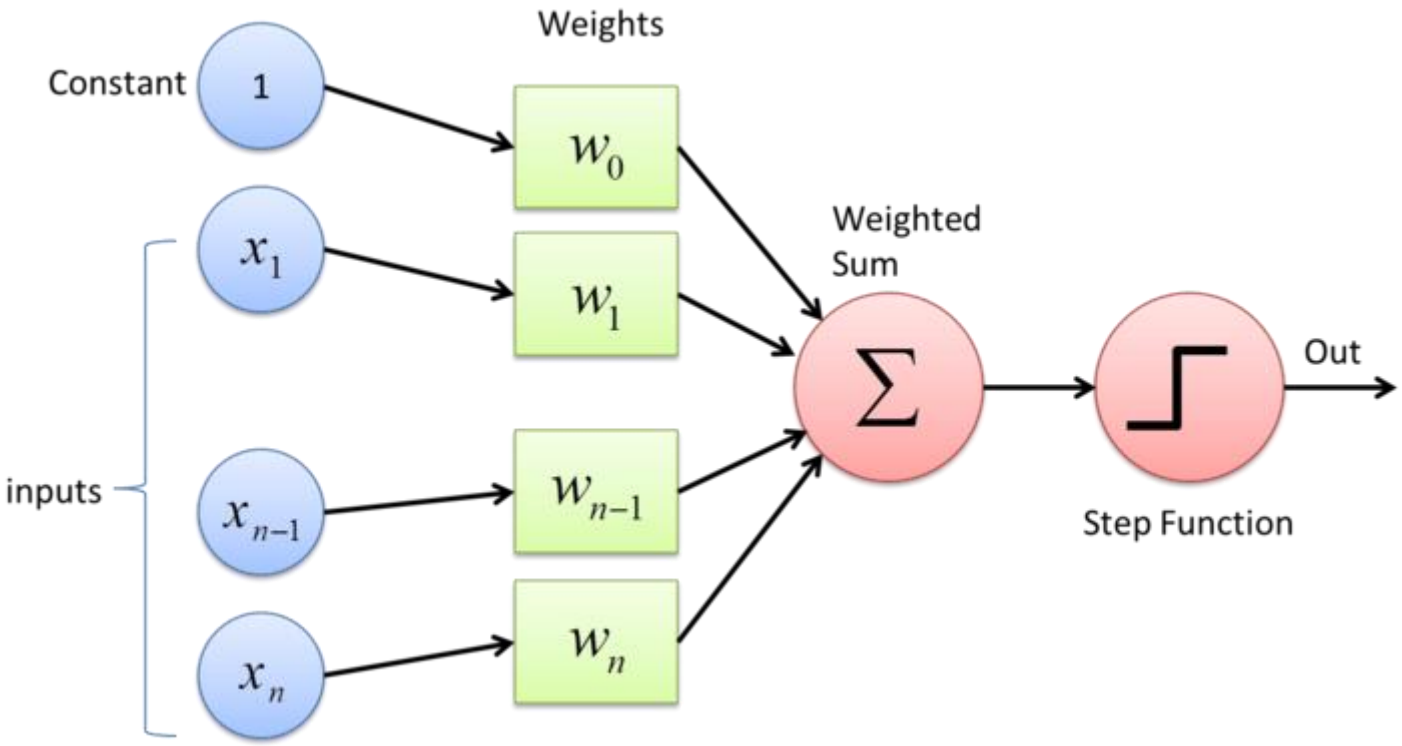
Biological Neurons Vs Artificial Neurons

Axon ------ → Output

Key Components of the Neural Network Architecture

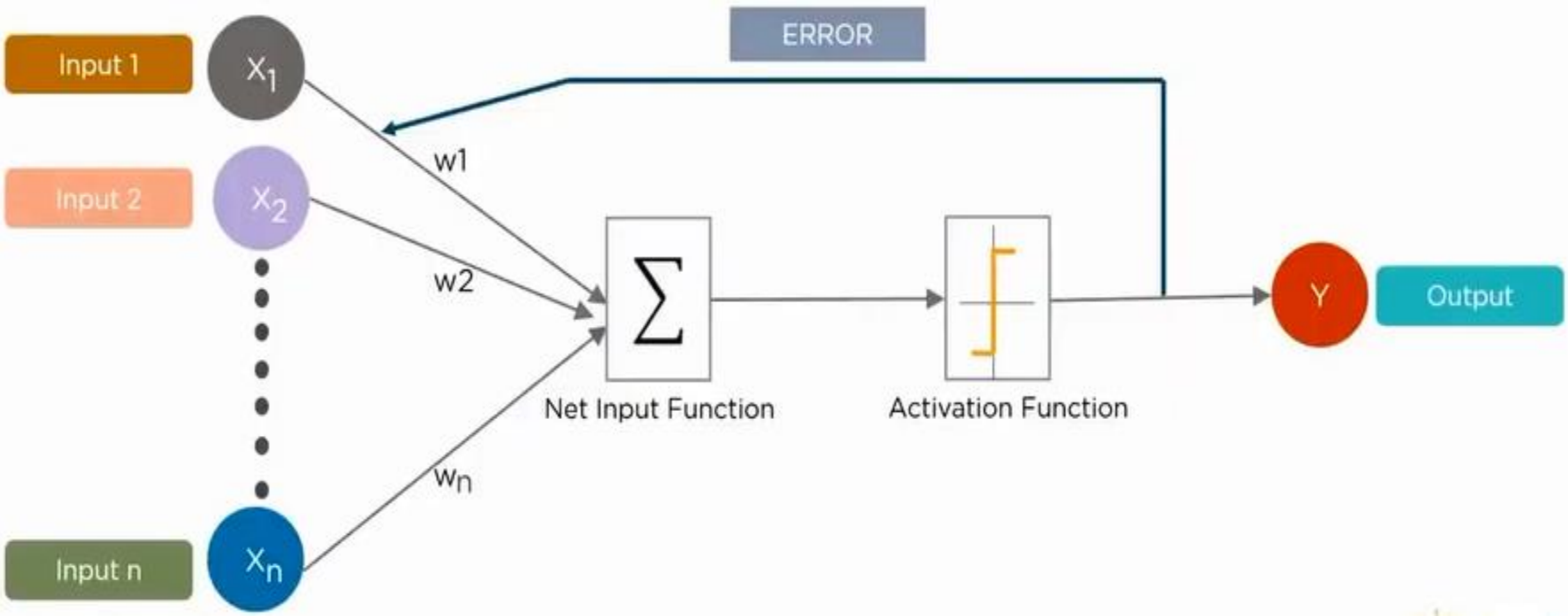
The Artificial Neural Network is made of individual units called neurons. These neurons are designed to mimic the behavior of the human brain.

The key components of a neuron are as follows.

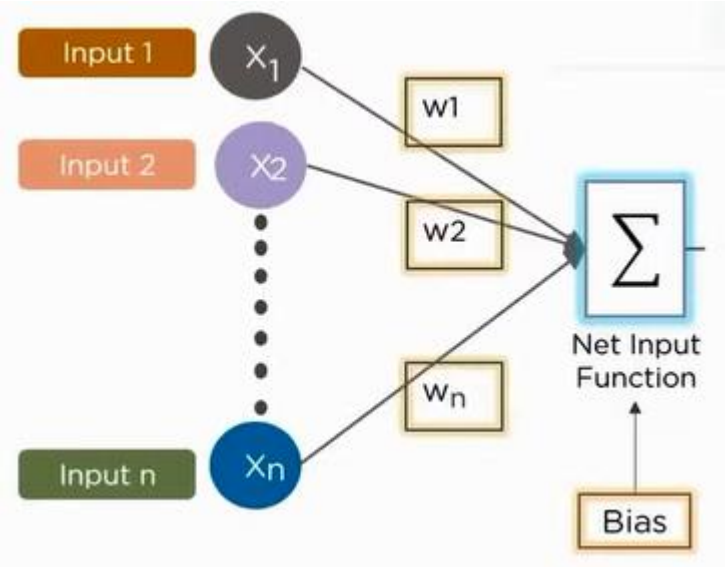


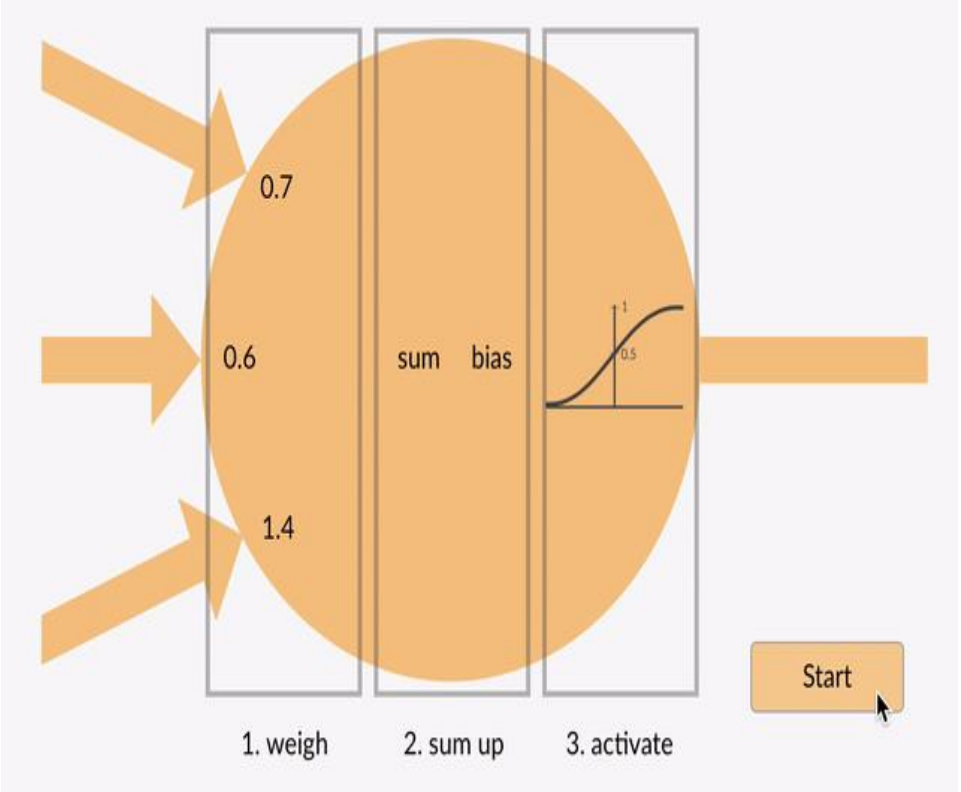
What is a Perceptron

A Perceptron is the basic part of the Neural Network . It represents a single neuron of a Human brain and is used for binary classifiers.
Perceptron is a model for understanding a single-layer ANN.



What is a Perceptron

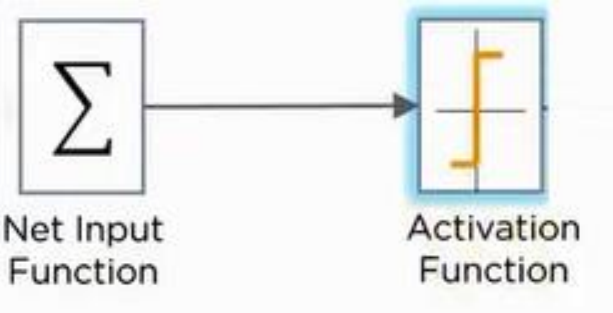




Inputs are multiplied with the Weights and a summation is Performed , plus a bias is added

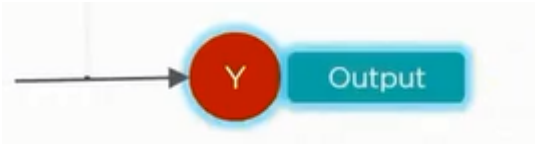
$$\sum_{i=1}^{n} w_i * x_i + b$$

What is a Perceptron

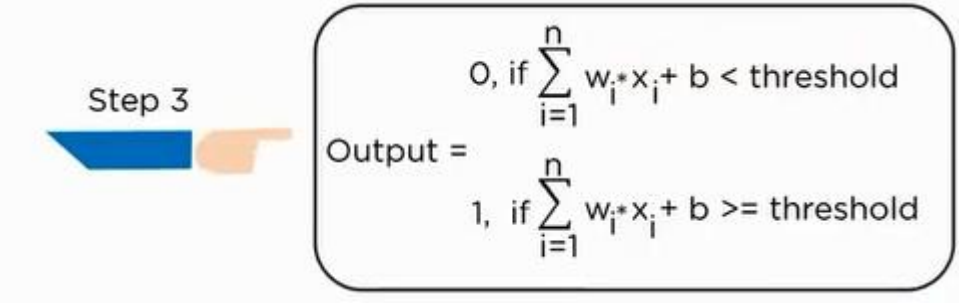


The Weighted sum of inputs is passed to an activation function to determine if a Neuron will fire or not

What is a Perceptron

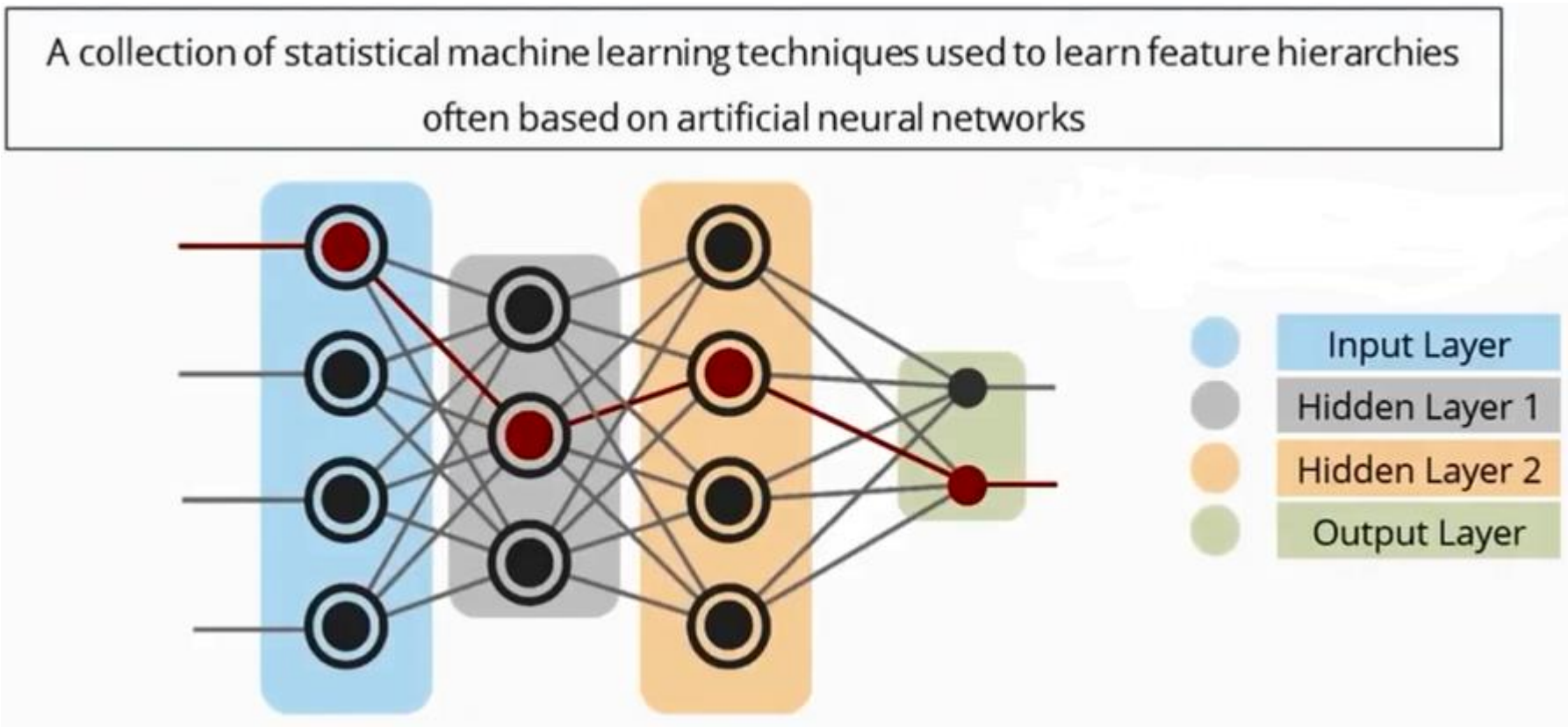


Perceptron receives an input signals if the sum of the Input signals exceed a certain threshold value, it either Outputs a signal or does not return a output.

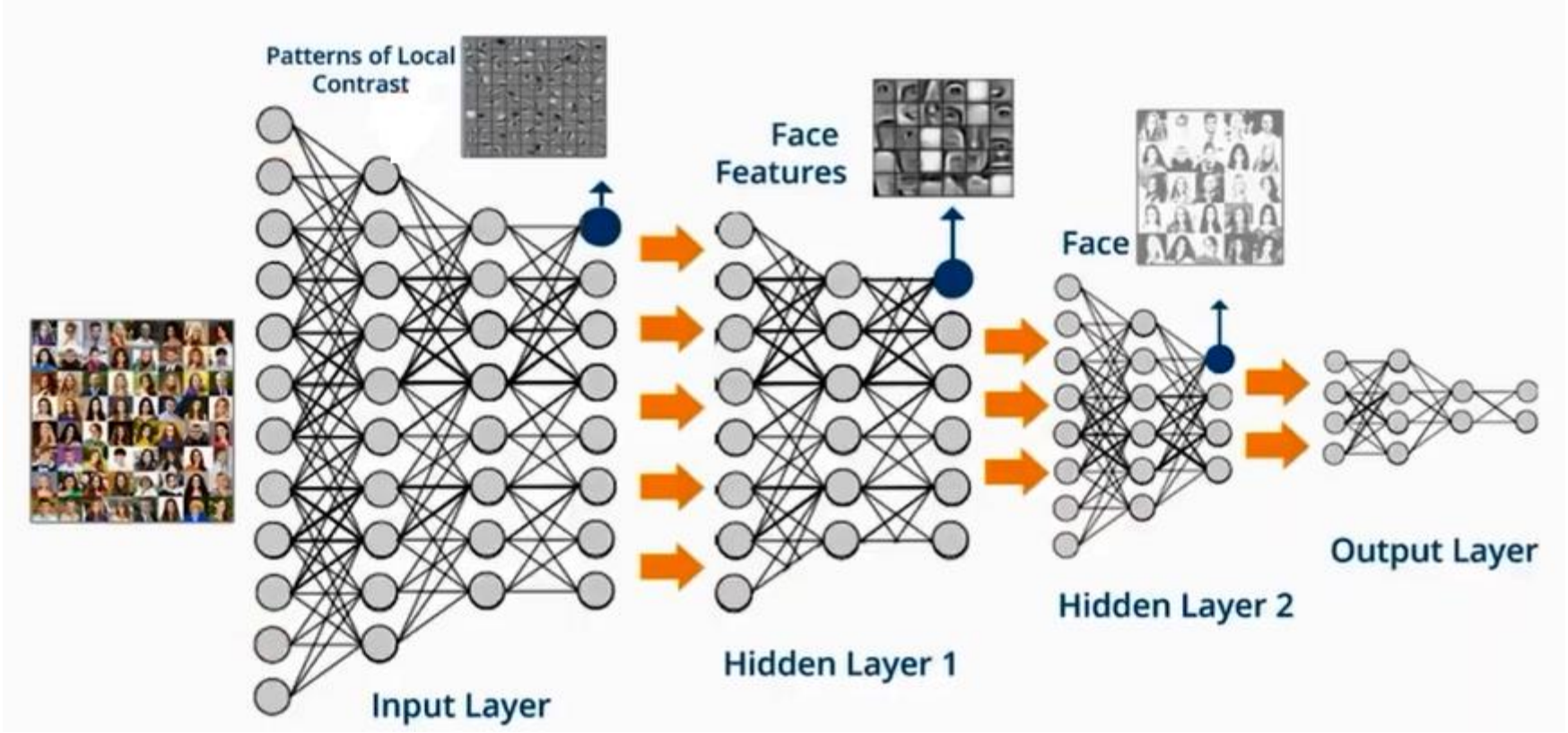


What is Deep Learning

A collection of statistical machine learning techniques used to learn feature hierarchies often based on artificial neural networks



Deep Learning Example



Deep Learning Applications

Self Driving Cars

Voice Controlled Assistance

Automatic Image Caption Generation

Automatic Machine Translation

How the Deep Learning works

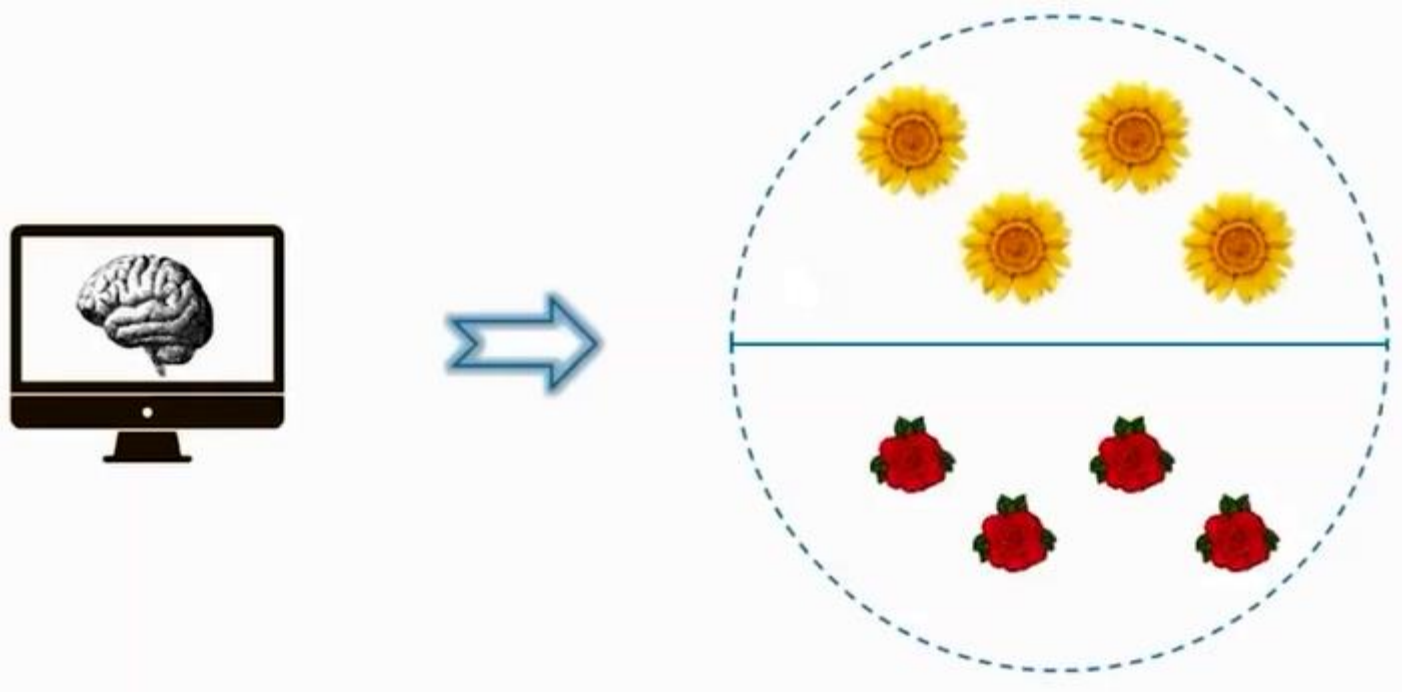
I have a dataset about flowers.

It includes:

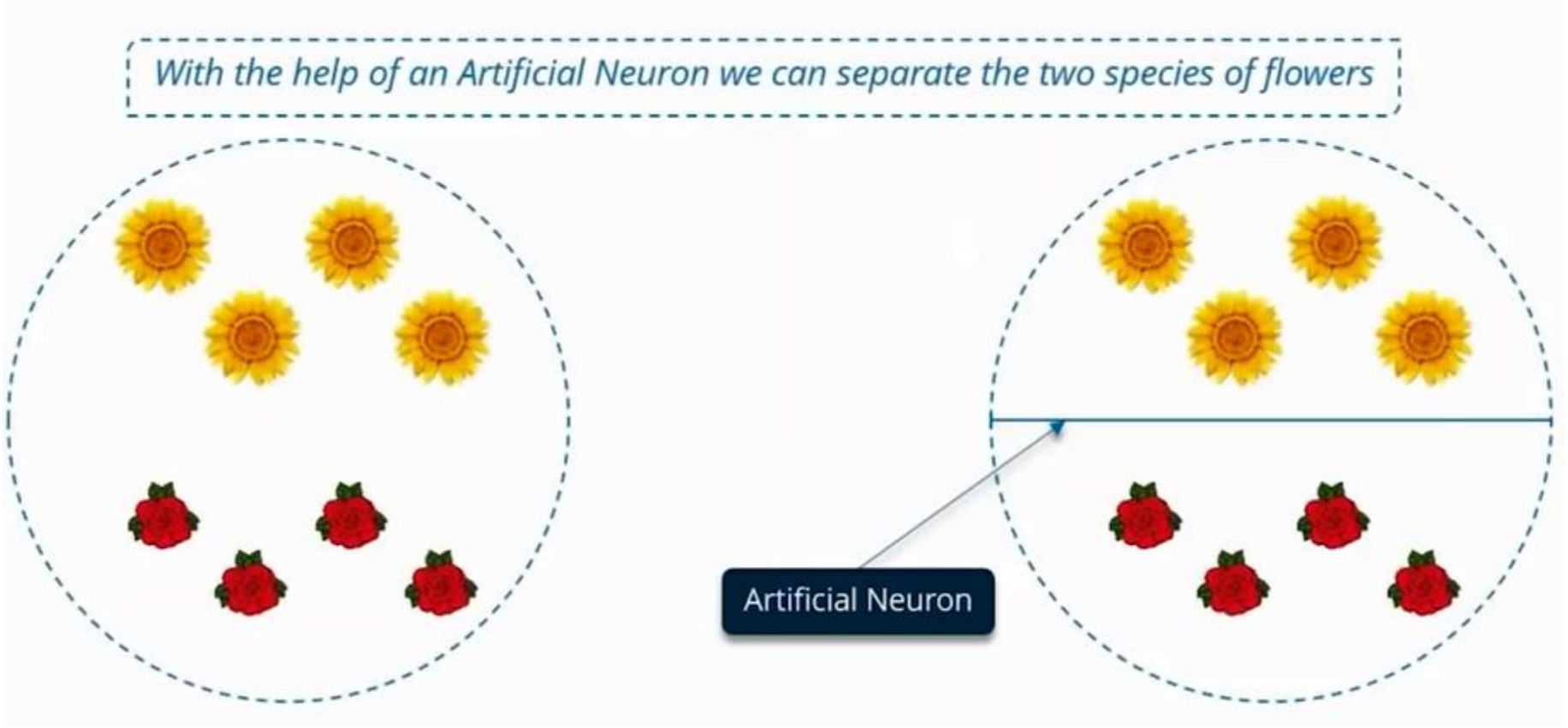
- ❑ Sepal Length
- ❑ Sepal Width
- ❑ Petal Length
- ❑ Petal Width

I want to classify the type of flower on the basis of this dataset.

Why we need Artificial Neuron

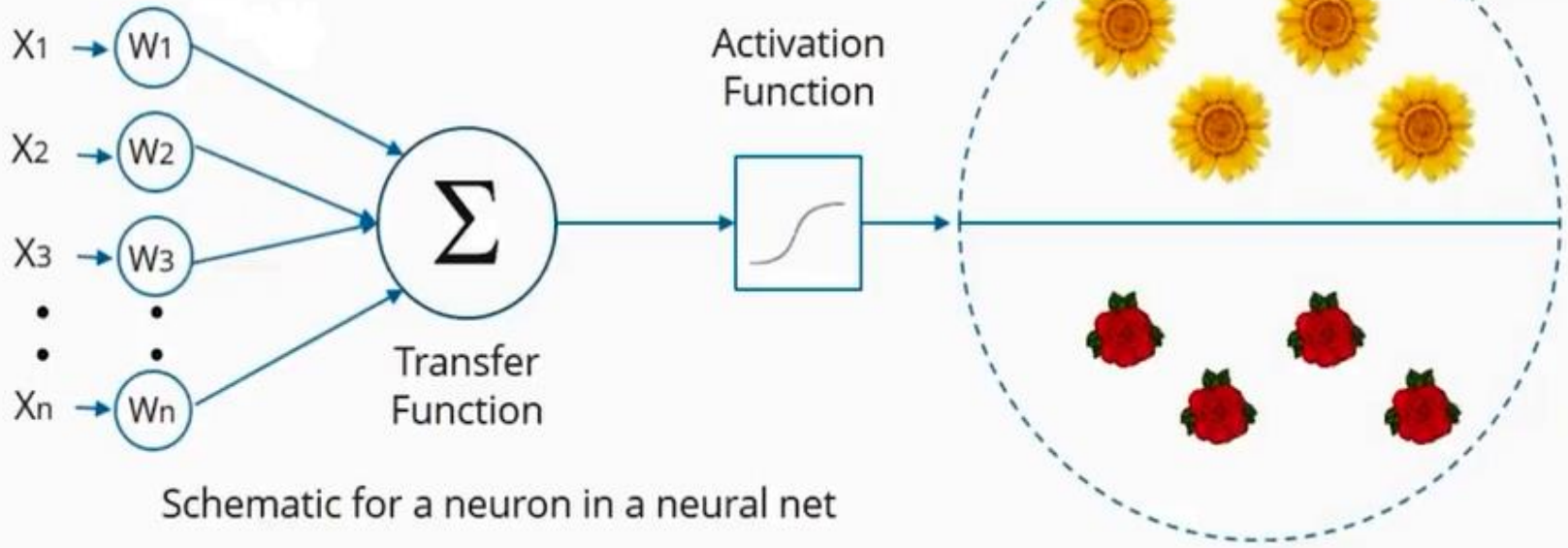


Why we need Artificial Neuron



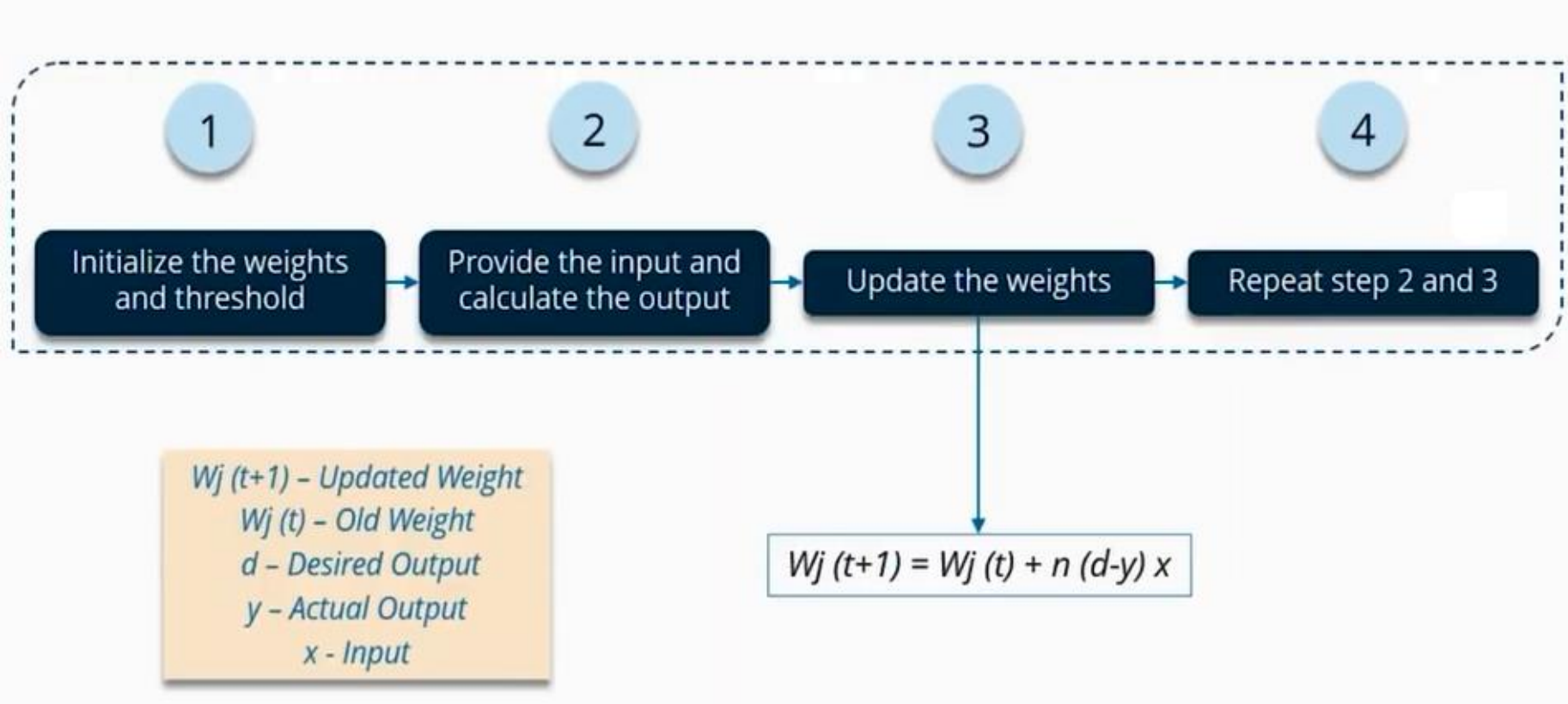
Perceptron Learning Algorithm

To get started, I'll explain a type of artificial neuron called a *perceptron*.



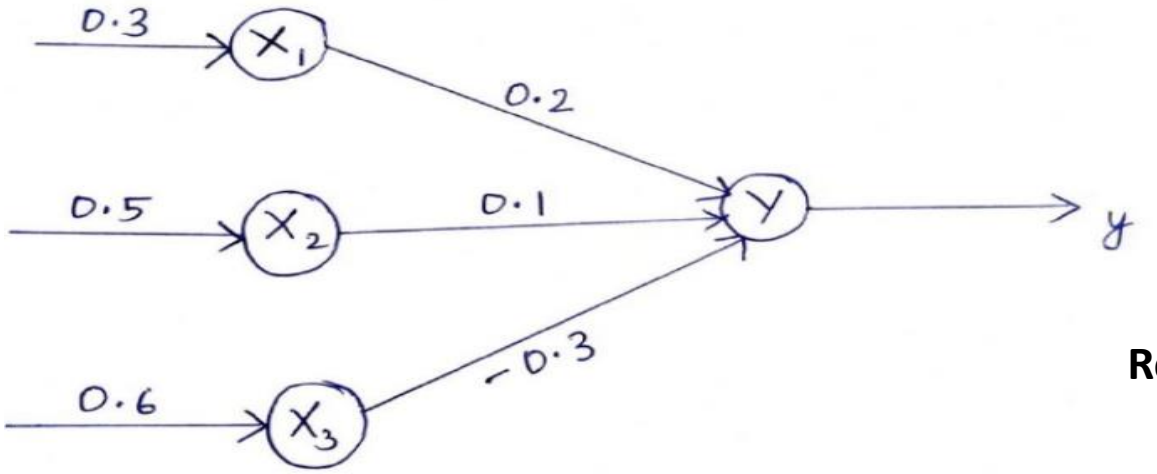
Schematic for a neuron in a neural net

Perceptron Learning Algorithm



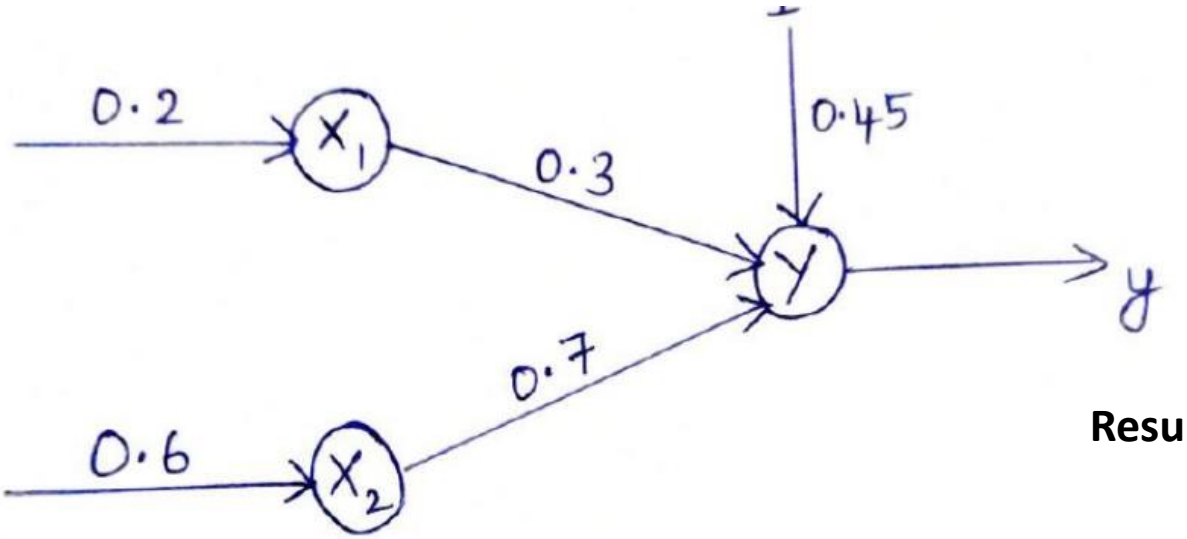
For the network shown, calculate the net input to the output neuron.

1.



Result= -0.07

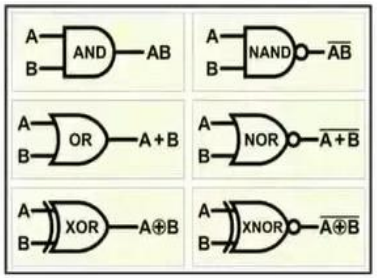
2.



Result= 0.93

Applications

A logic gate is the basic building block of a digital circuit. Most logic gates have 2 inputs and 1 output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels.



Popular Logic Gates

- AND
- OR
- NAND
- NOR
- NOT
- XOR

Applications

It can be used to implement Logic Gates.

OR

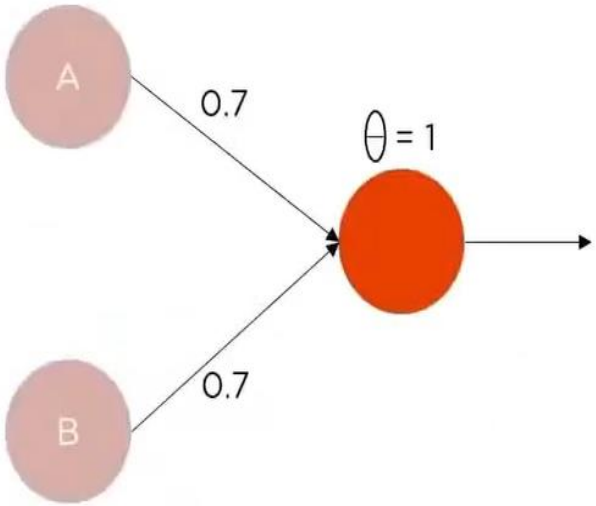
X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

AND

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Applications-AND Gate

A and B are input neurons. The red neuron is our output neuron. The threshold value is 1. If the sum of the weighted input neurons is greater than the threshold, the output is 1 else 0.



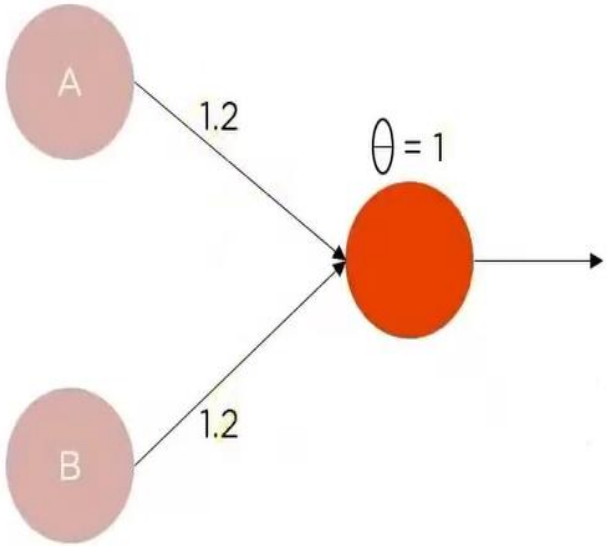
w_1*x_1	w_2*x_2	$w_1*x_1 + w_2*x_2$
Input (A)	Input (B)	Output
0.7*0	0.7*0	0
0.7*0	0.7*1	0.7
0.7*1	0.7*0	0.7
0.7*1	0.7*1	1.4

→ This exceeds the threshold, so output is 1

We have designed a neuron which implements a logical AND gate

Applications-OR Gate

A and B are input neurons. The red neuron is our output neuron. The threshold value is 1. If the sum of the weighted input neurons is greater than the threshold, the output is 1 else 0.



w_1*x_1	w_2*x_2	$w_1*x_1 + w_2*x_2$
Input (A)	Input (B)	Output
1.2*0	1.2*0	0
1.2*0	1.2*1	1.2
1.2*1	1.2*0	1.2
1.2*1	1.2*1	2.4

These exceed the threshold, so output is 1

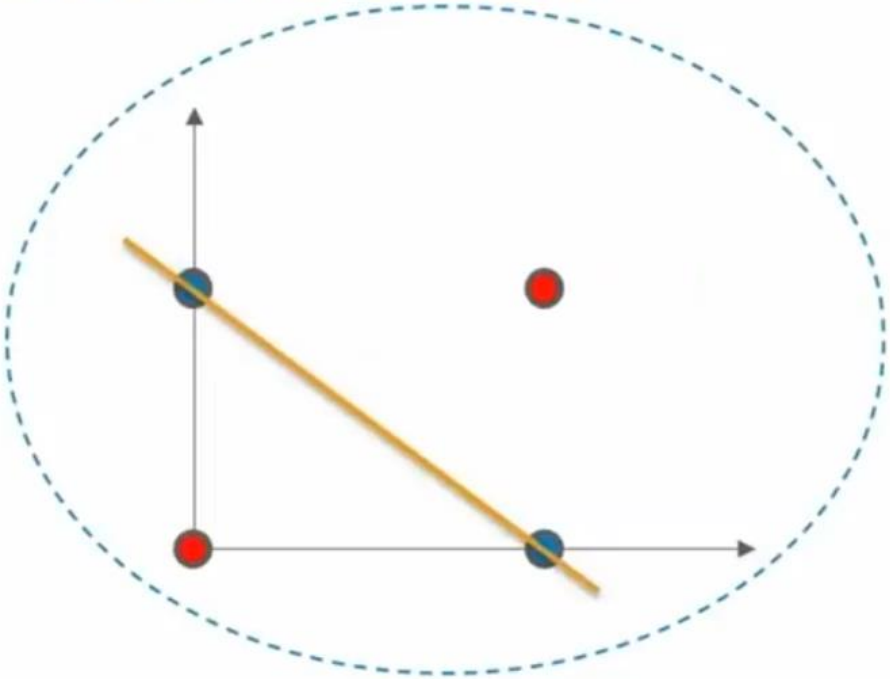
We have designed a neuron which implements a logical OR gate

Applications-XOR Gate

Let us understand this with an example:
How can I implement an XOR gate using Single Layer Perceptron?

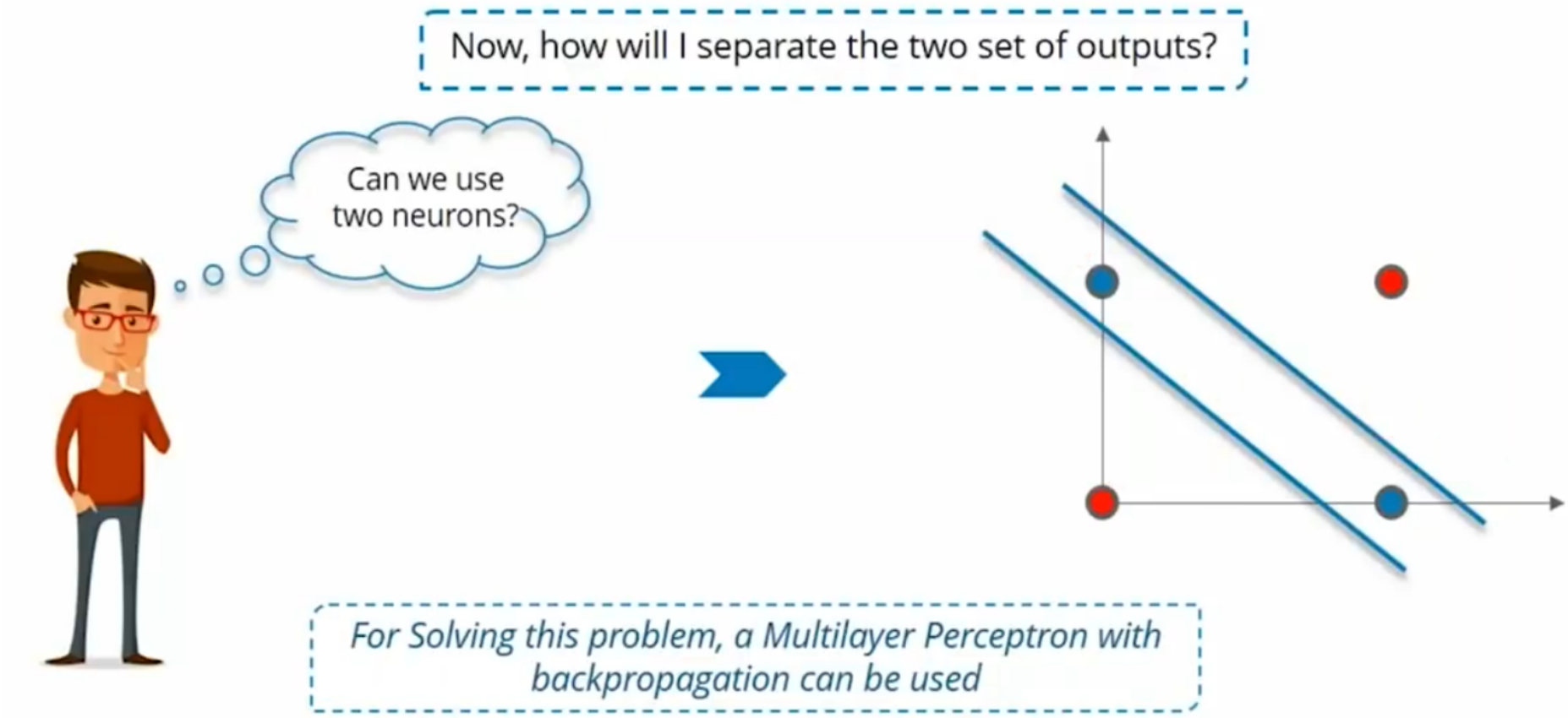
XOR

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0



Limitation of Single Perceptron

Applications-XOR Gate

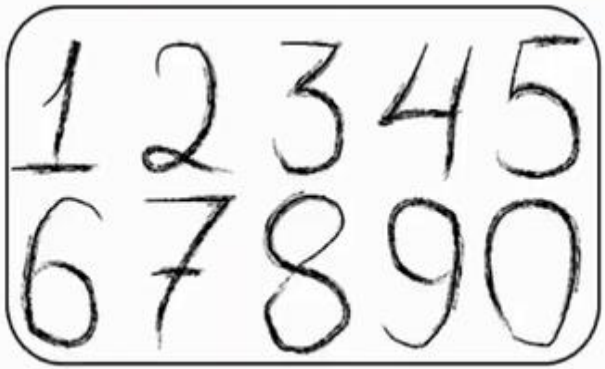


Limitation of Single Perceptron can be overcome by using Multilayer Perceptron-More than one hidden layer.

Single Layer Perceptron-Use Case

Let's now see how to implement a single layer neural network for an image classification problem using TensorFlow.

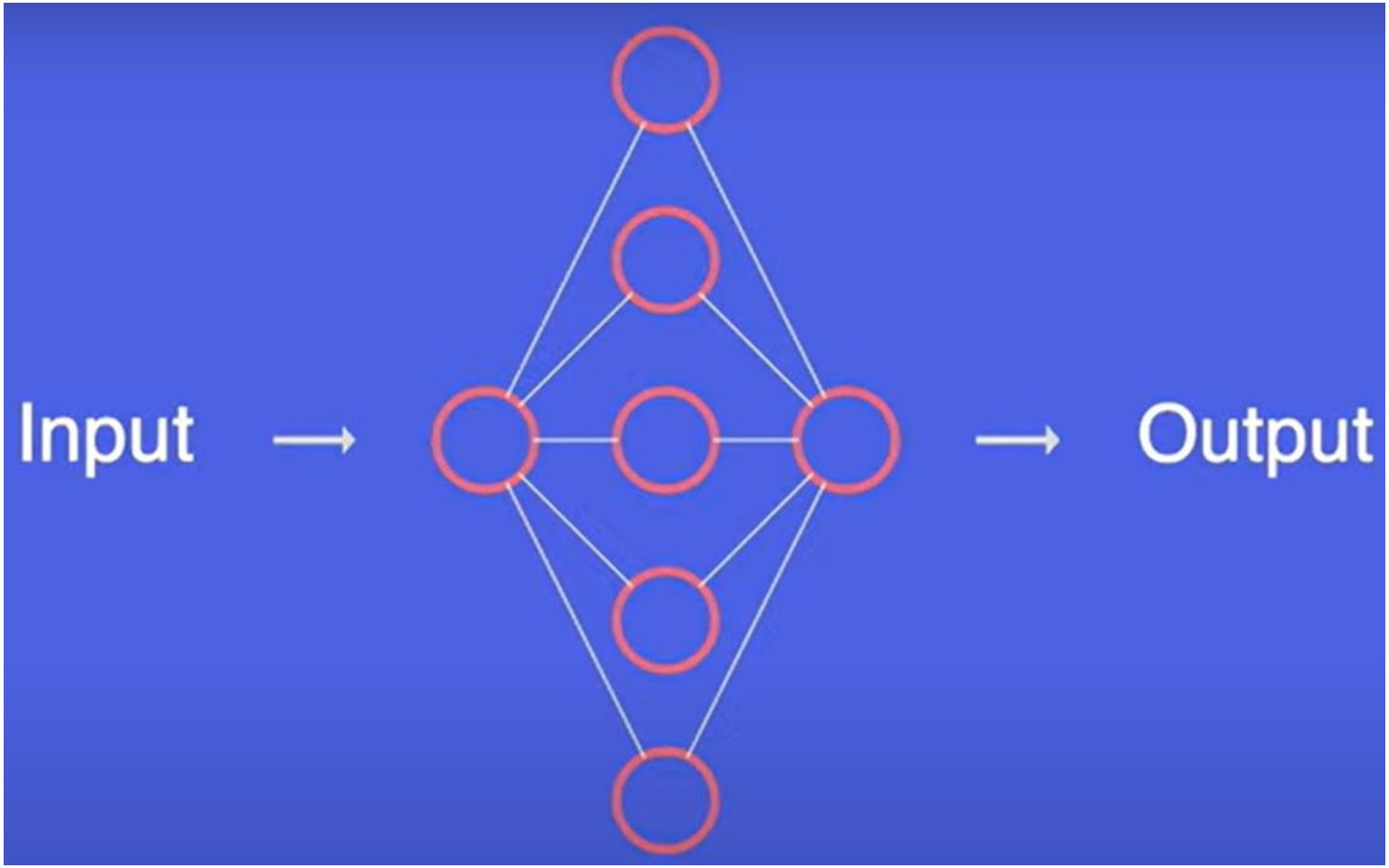
MNIST Dataset



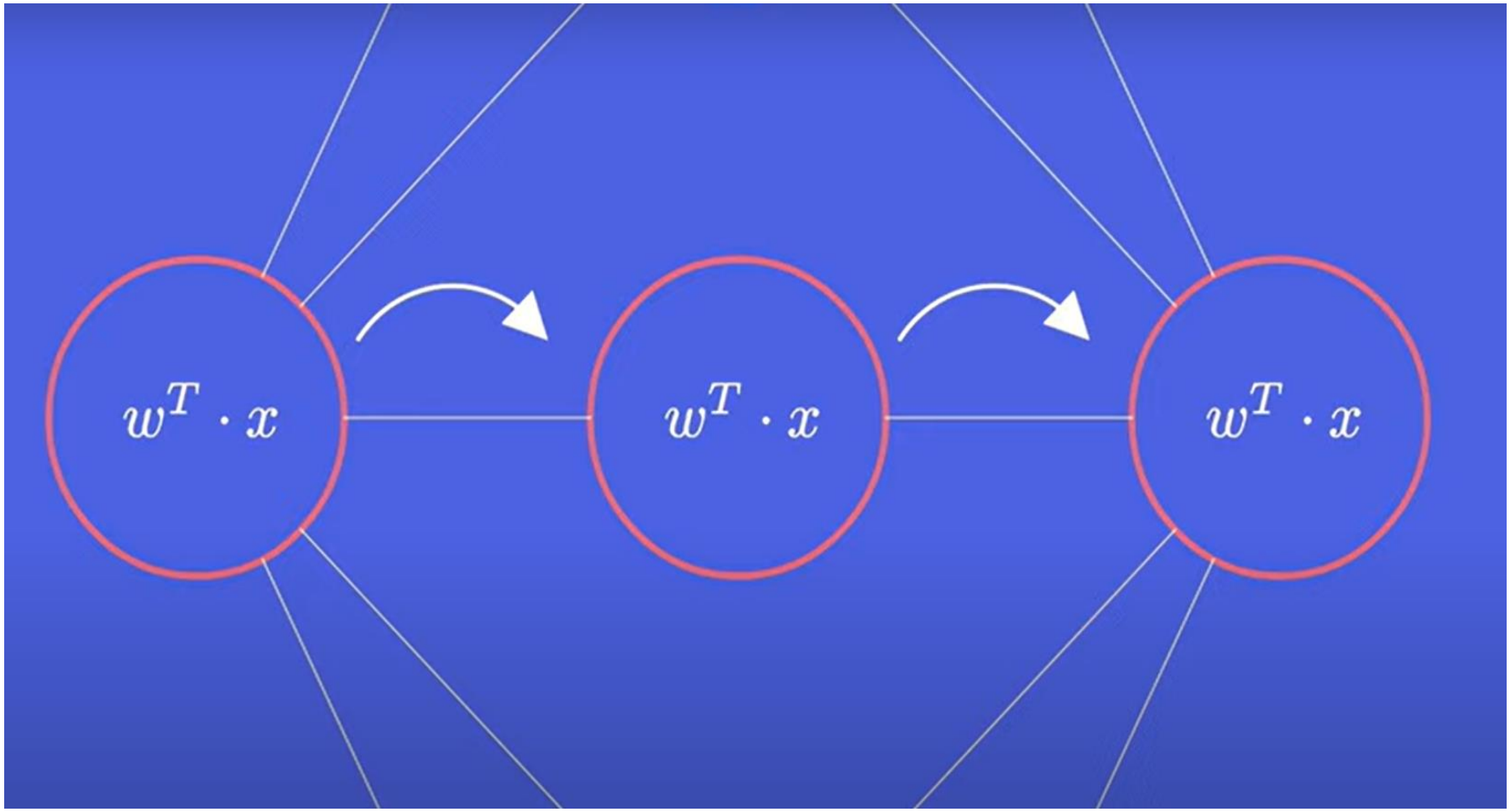
55,000 training images

10,000 testing images

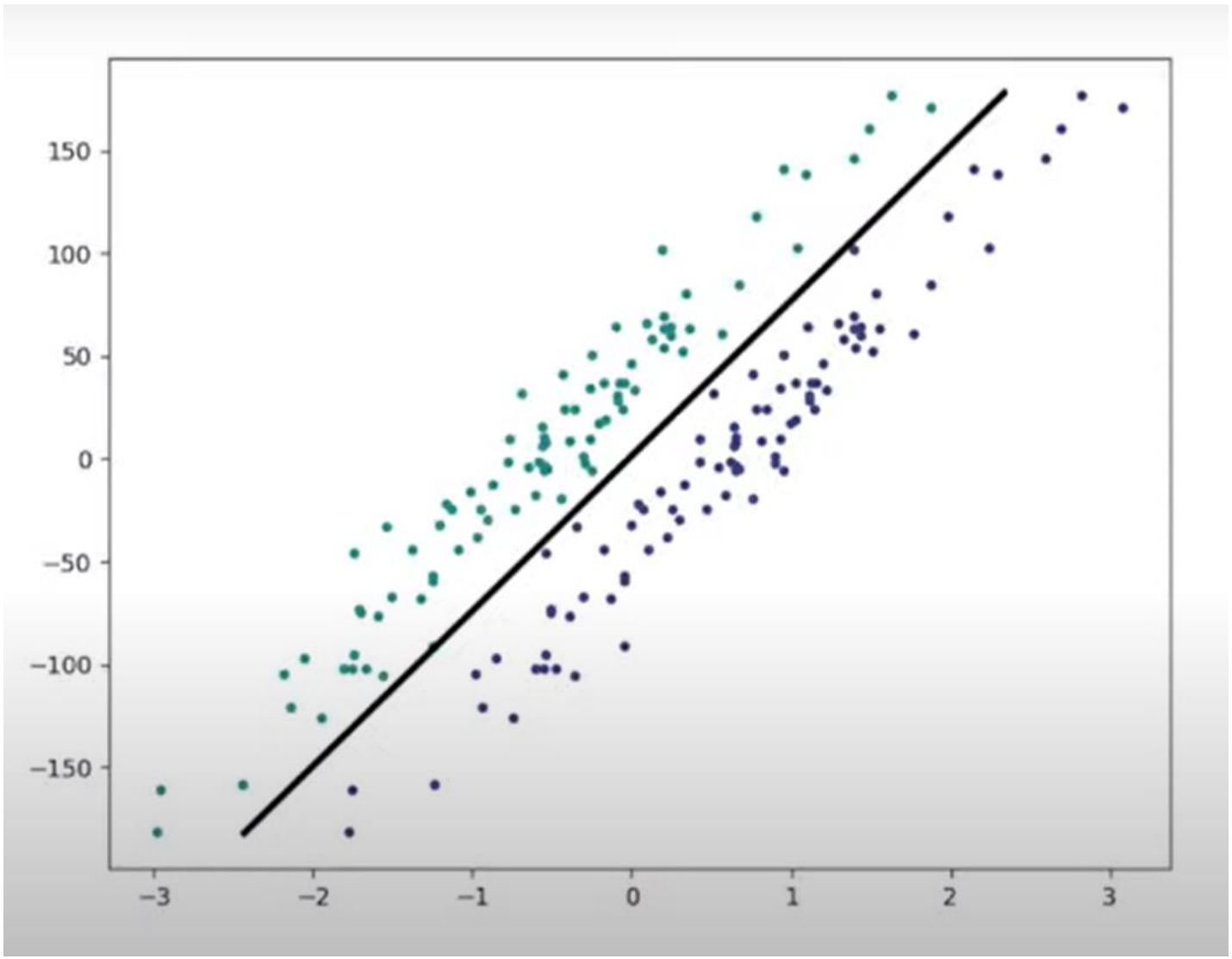
Activation Functions



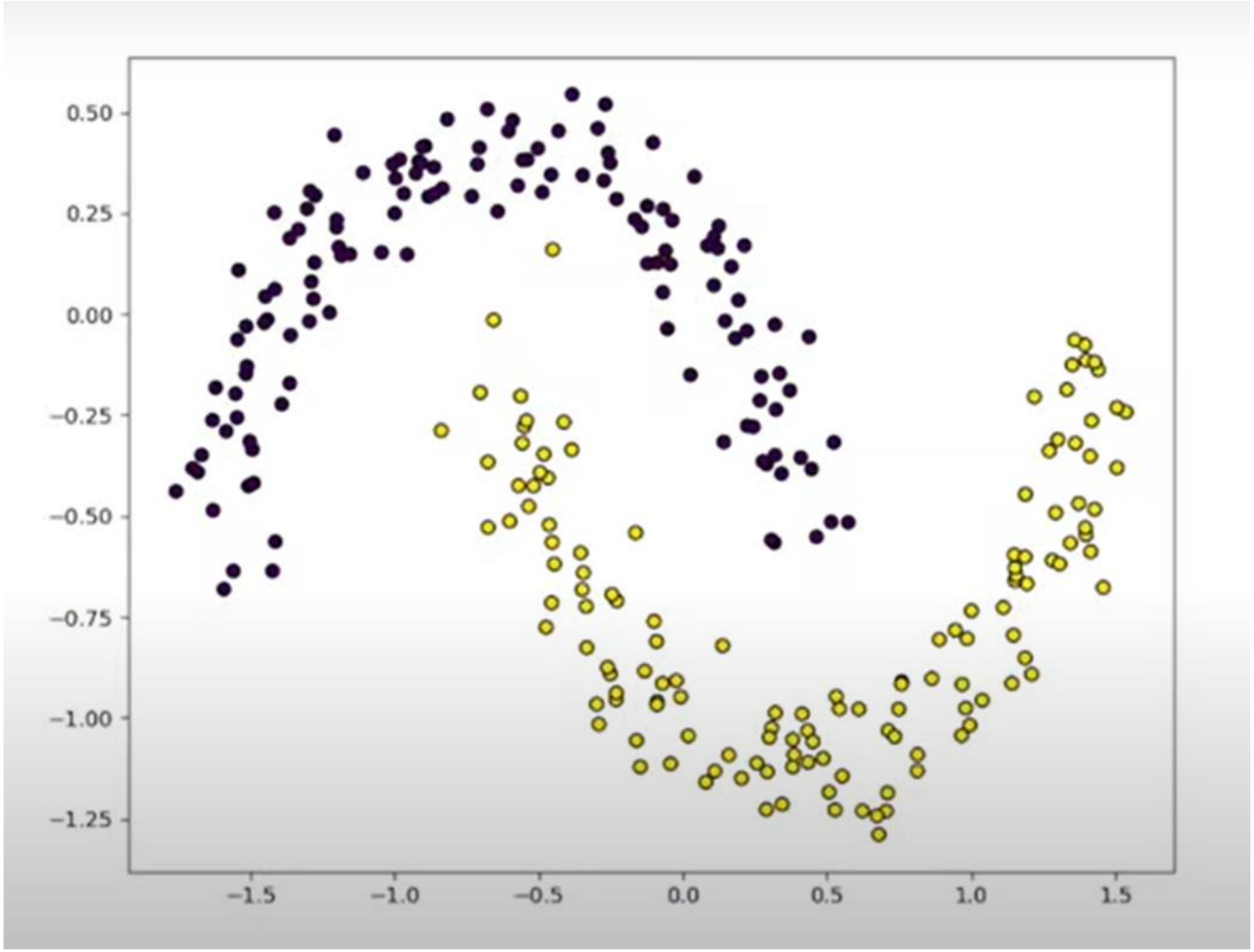
Activation Functions



Activation Functions



Activation Functions



Terms used in Neural Networks

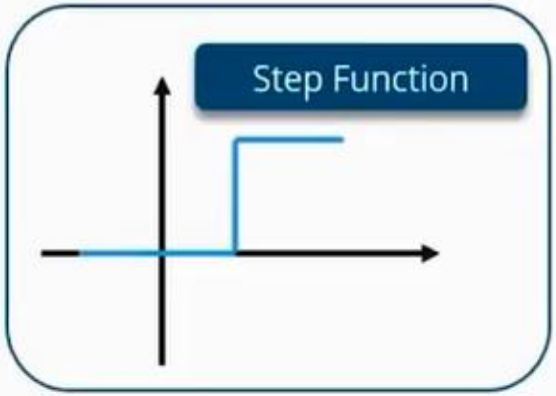
Activation Functions

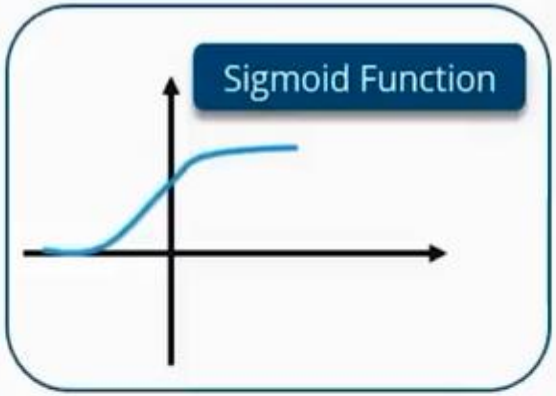
- It's just a thing function that you use to get the output of node.
- It is also known as **Transfer Function**
- **The Activation Functions can be basically divided into 2 types-**

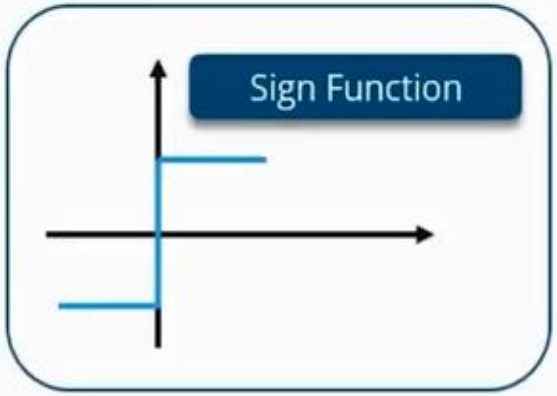
1. Linear Activation Function-Step, Relu, Leaky Relu etc.
2. **Non-linear Activation Functions-Introduce Non-Linearity in the Network-Sigmoid, Tanh, Softmax etc.**

- Decide whether a neuron can contribute to the next layer
- But, how do we decide if a neuron can fire/activate or not?

Activation Functions







Activation Functions

1. Step Function

If Value >0 ⟶ Activate

Else ⟶ Don not activate

'0' is the threshold

Drawback: Doesn't model every situation

Activation Functions-Step Function

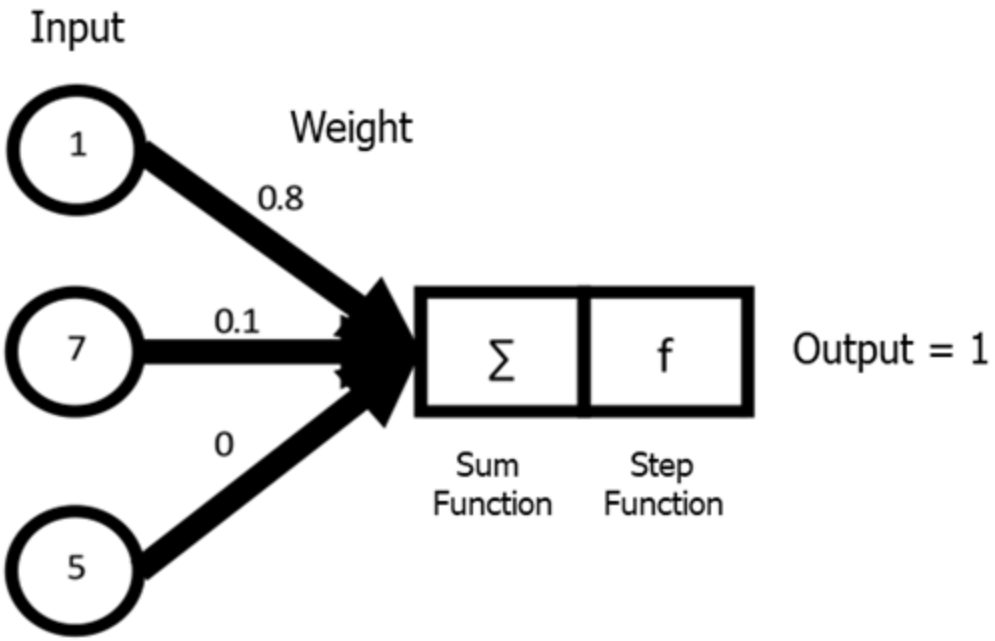
Step Function

$$s = \sum_{i=1}^{n} xi * wi$$

$$s = (1 * 0.8) + (7 * 0.1) + (5 * 0)$$

$$s = 0.8 + 0.7 + 0$$

$$s = 1.5$$



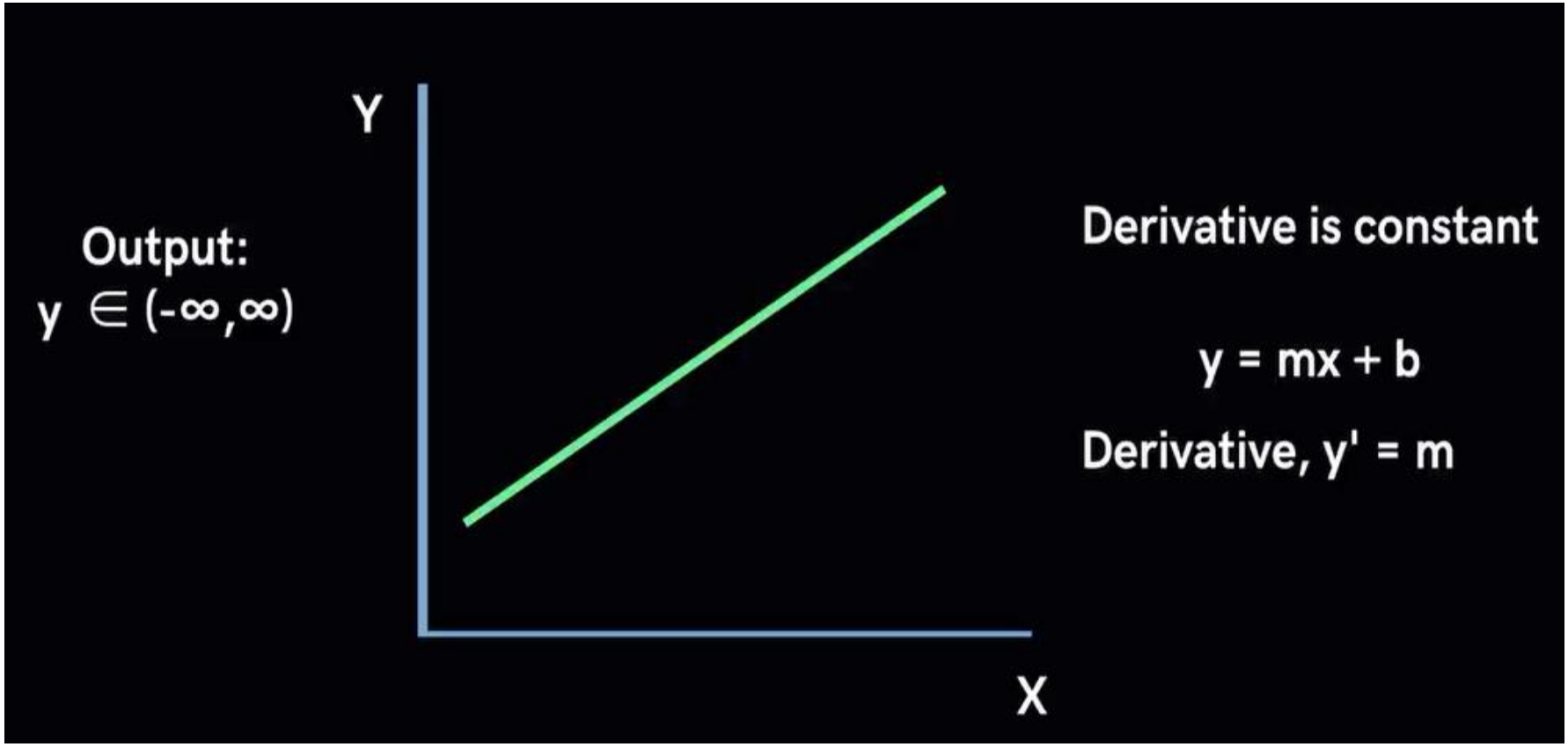
Step function

(If f > 0) = 1

(If f =<0) = 0

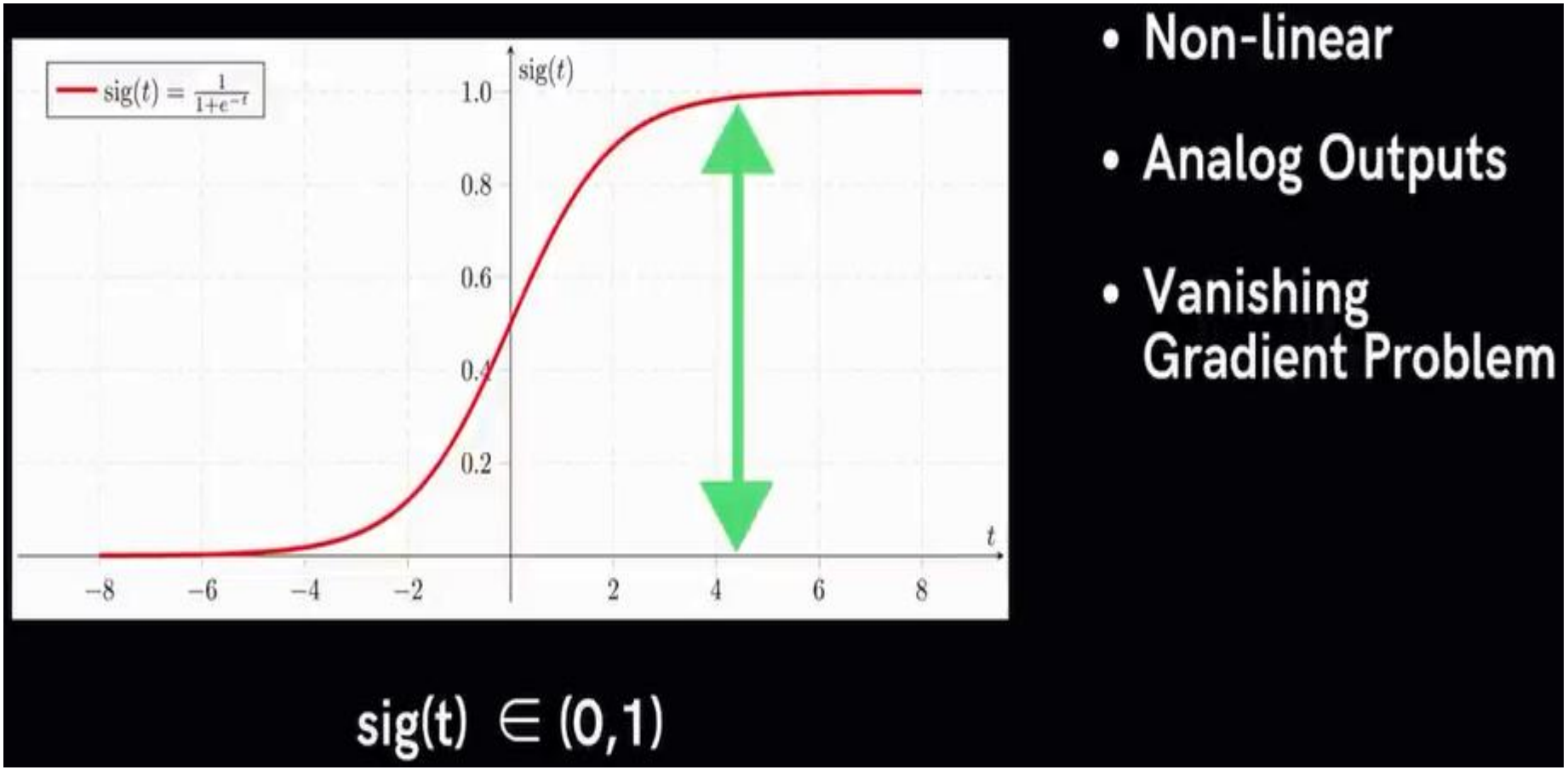
Activation Functions

2. Linear Function



Activation Functions

3. Sigmoid Function



Activation Functions

3. Sigmoid Function-

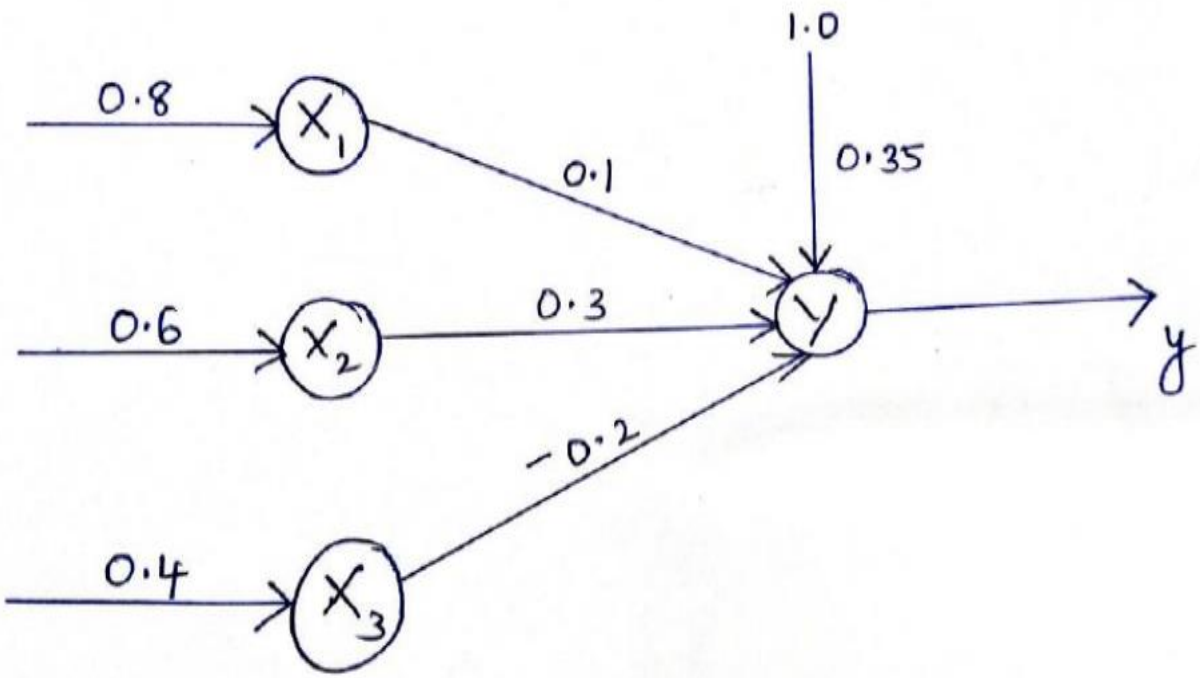
- The main reason why we use sigmoid function is because **it exists between (0 to 1).**
- Therefore, it is especially used for models where we have **to predict the probability as an output.** Since **probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.**
- The **function is differentiable**. That means, we can find the slope of the sigmoid curve at any two points.

Activation Functions

3. Sigmoid Function- Binary Sigmoid function & Bipolar Sigmoid Function

Binary Sigmoidal function: also called logistic sigmoid function or unipolar sigmoid function. It can be defined as:

$f(x) = 1 / 1+e^{-\lambda x}$ where λ is the steepness parameter.



Activation Functions

3. Sigmoid Function

Binary Sigmoidal Function: It can be defined as:
f(x) = 1 /1+$e^{-\lambda x}$ where λ is the steepness parameter.

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

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$= 0.35 + 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times (-0.2)$$

***Yin*=0.53**

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.53}} = 0.625$$

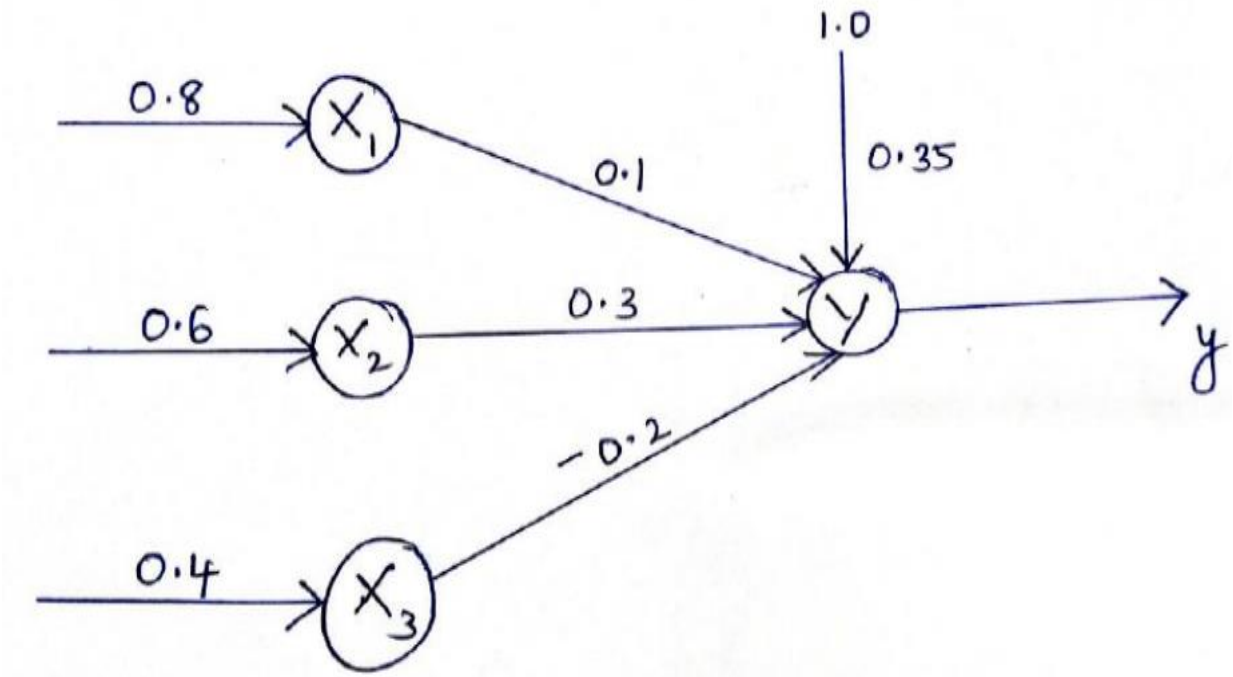
Binary Sigmoidal Function *y* = 0.625

Activation Functions

3. Sigmoid Function

Bipolar Sigmoidal Function: It can be defined as: **Range-(-1,+1). It is a modified Sigmoidal Function**

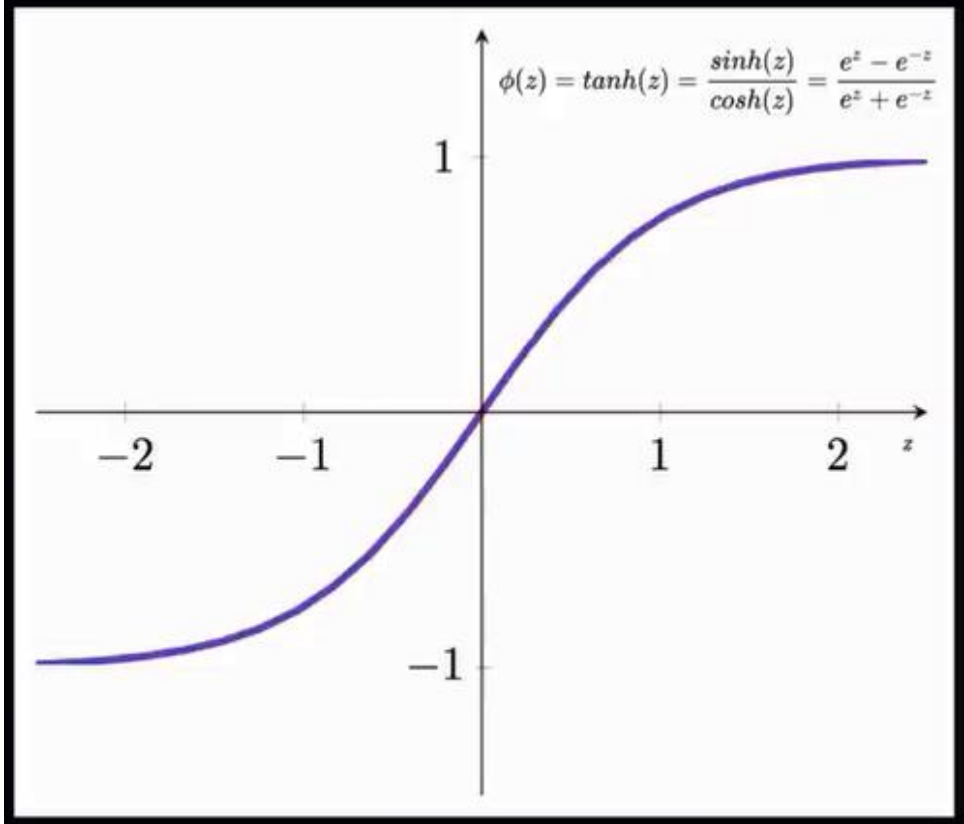
f(x) = 2 /1+$e^{-\lambda x}$ - 1 where λ is the steepness parameter.



$$y = f(y_{in}) = \frac{2}{1+e^{-y_{in}}} - 1 = \frac{2}{1+e^{-0.53}} - 1 = 0.259$$

Activation Functions

4. Tanh Function



tanh(x) ∈ (-1,1)

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

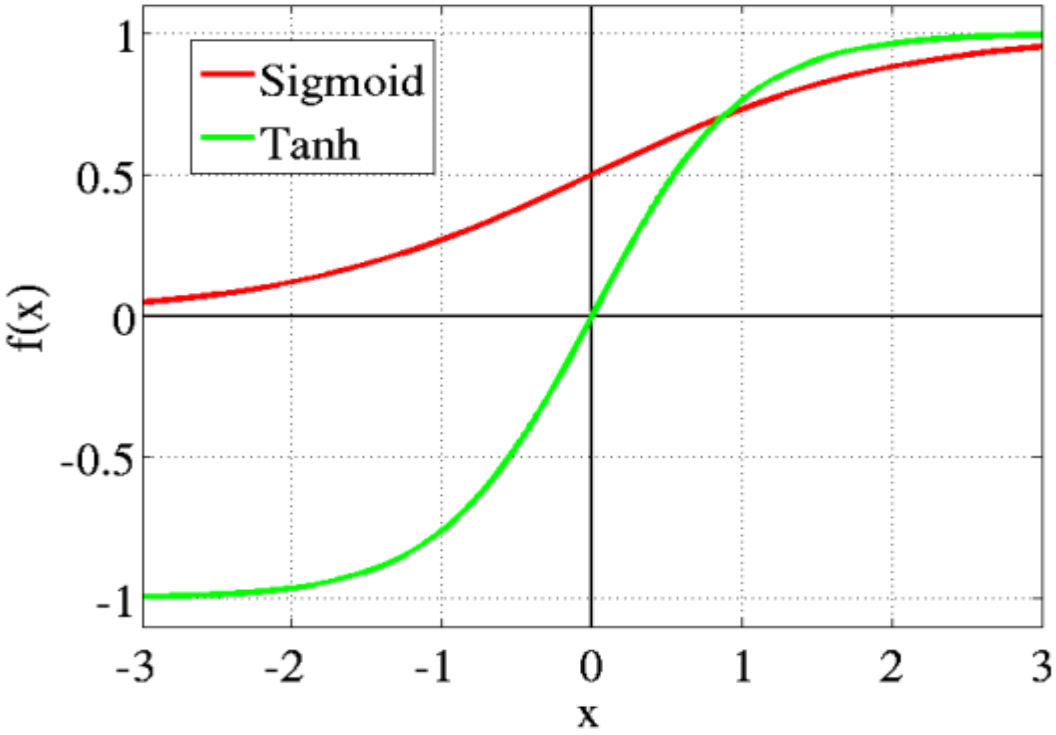
tanh(x) = 2 sigmoid(2x) - 1

- Non-linear
- Derivative Steeper than Sigmoid
- Vanishing Gradient Problem

Activation Functions

4. Tanh Function

tanh is also like **logistic sigmoid** but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).



The advantage is that the negative inputs will be mapped strongly negative, and the zero inputs will be mapped near zero in the tanh graph.

The function is **differentiable**.

Activation Functions-Vanishing Gradient Problem

Vanishing Gradient Problem:

- When the neural networks are trained with gradient-based learning methods and backpropagation, they encounter the 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.**

- The **vanishing gradient problem is caused by the derivative of the activation function used** to create the neural network.

Why:

Certain activation functions, like **the sigmoid function, squishes a large input space into a small input space between 0 and 1.**

Therefore, **a large change in the input of the sigmoid function will cause a small change in the output.** Hence, the **derivative becomes small.**

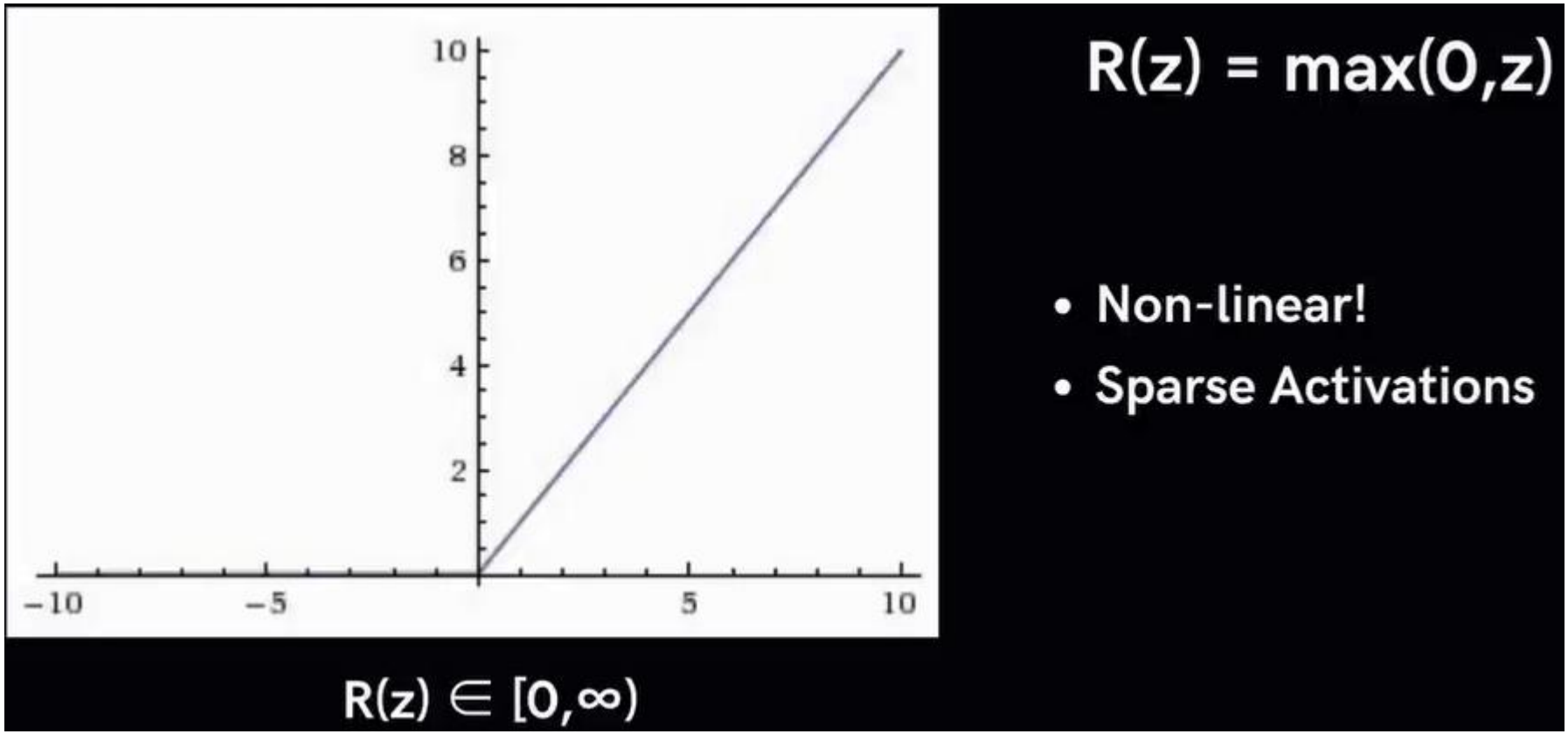
Solutions:

The simplest solution is to use other activation functions, such as **ReLU,** which **doesn't cause a small derivative.**

Activation Functions

4. Relu Function (Rectified Linear Unit)

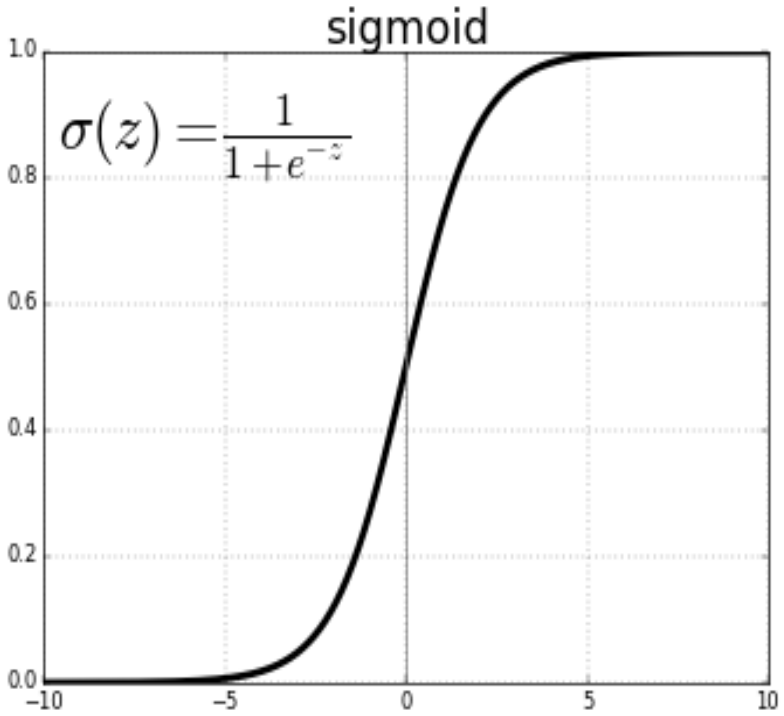
The ReLU is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.

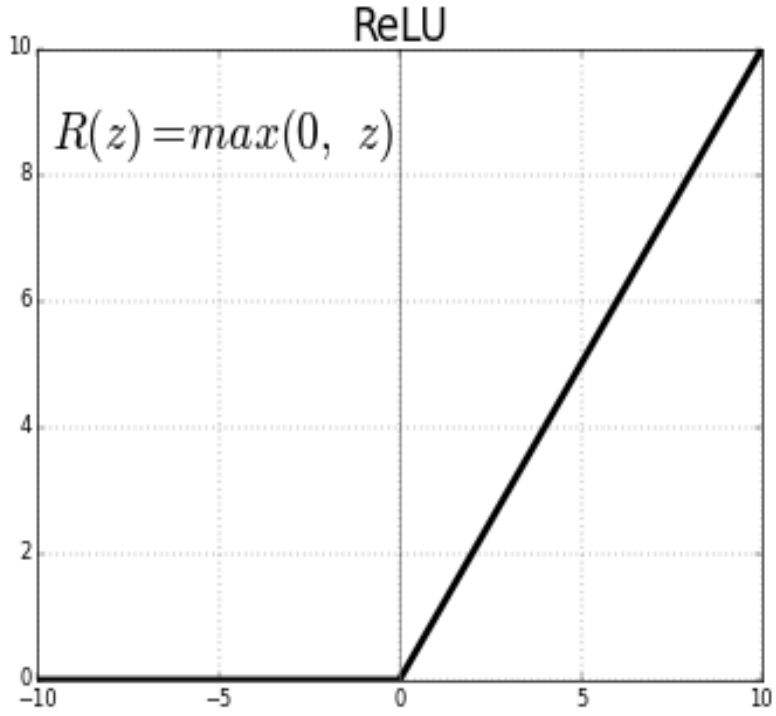


Activation Functions

4. Relu Function (Rectified Linear Unit)

Range: [0 to infinity)



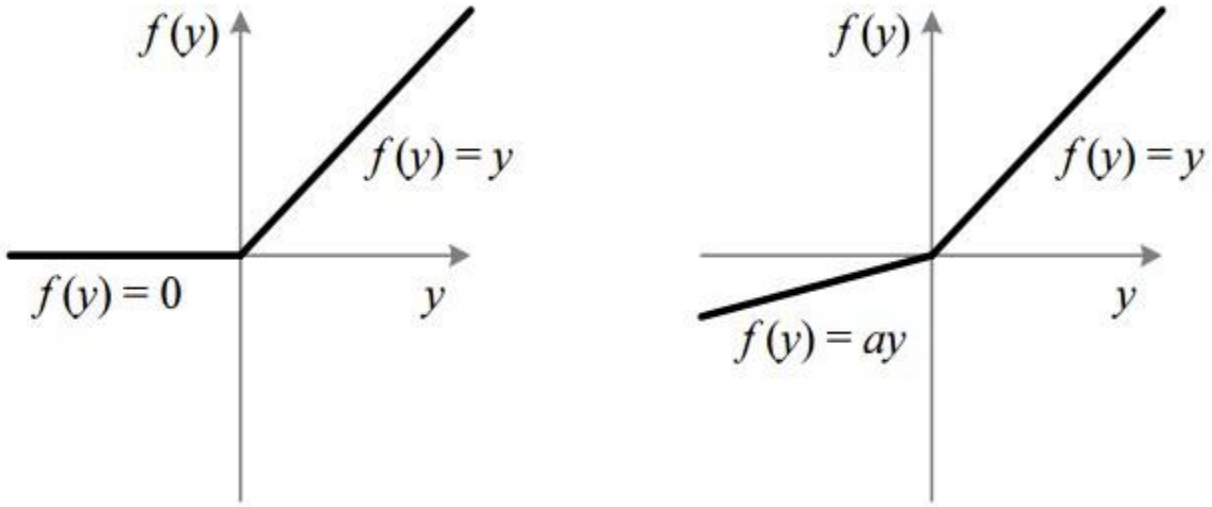


ReLU-The issue is that **all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly.**

Activation Functions

5. Leaky ReLU

It is an attempt **to solve the dying ReLU problem. It has a small slope for negative values instead of a flat slope.**



The **leak helps to increase the range of the ReLU function.** Usually, the value of **a** is 0.01 or so.

When **a is not 0.01** then it is called **Randomized ReLU**.

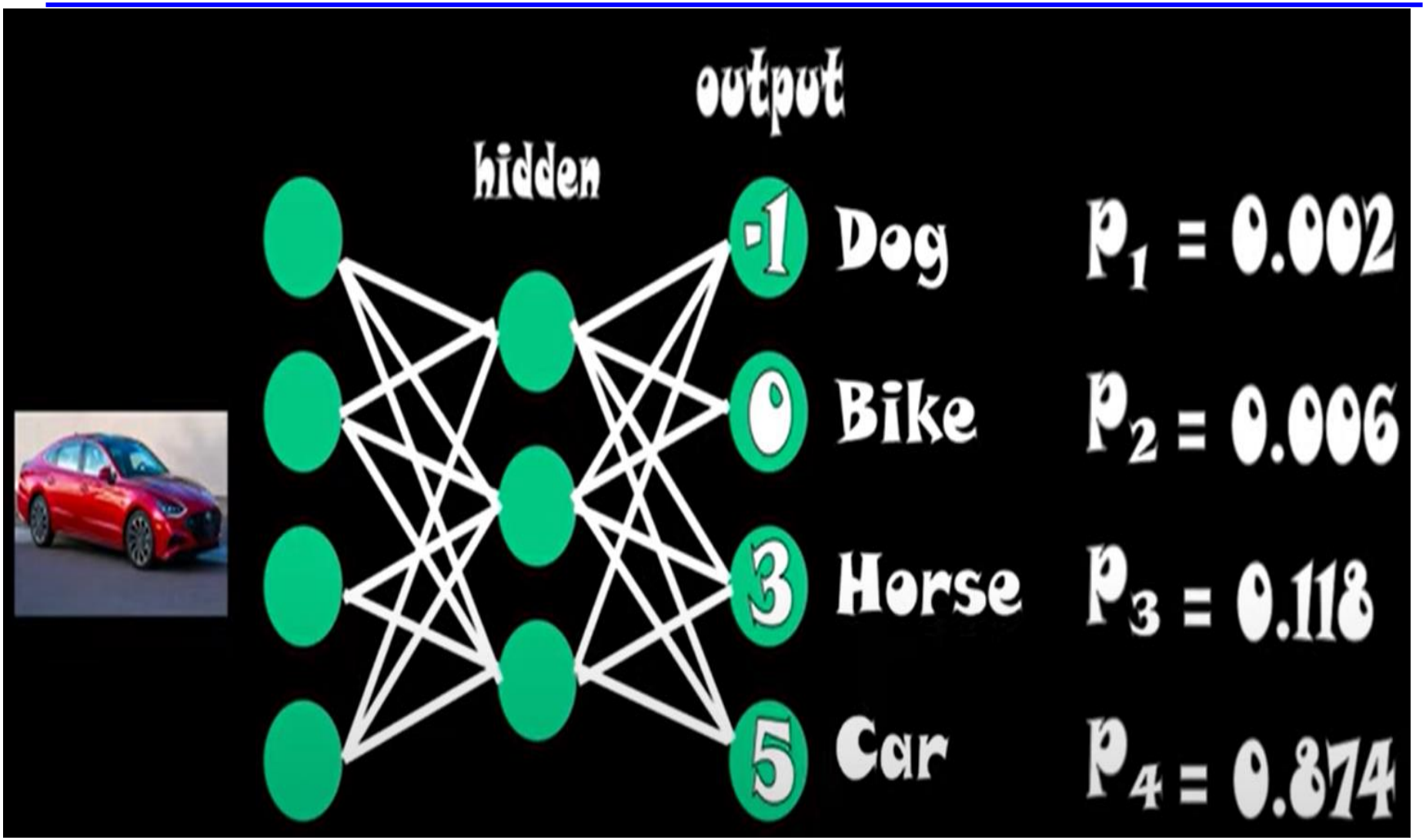
Therefore, the **range** of **the Leaky ReLU is (-infinity to infinity).**

Activation Functions

6. Softmax function

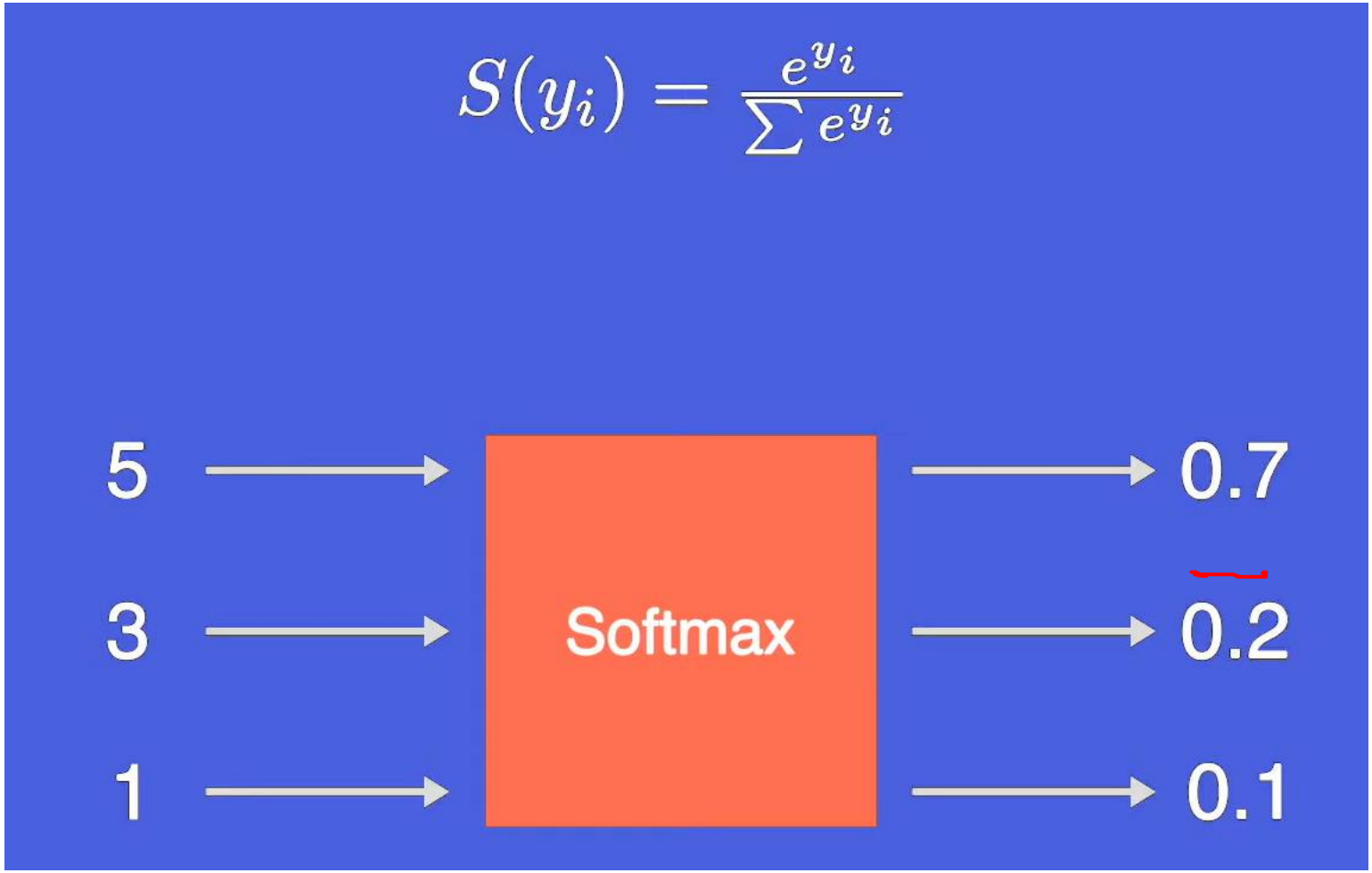
- Softmax is used mainly at the last layer i.e. output layer for decision making the same as sigmoid activation works.
- The **Softmax basically gives value to the input variable according to their weight and the sum of these weights is eventually one.**
- **Softmax** is a mathematical function that converts a vector of numbers into a vector of probabilities.
- You can consider higher probabilities as actual output.
- It is used to predict **a multinomial probability distribution.**
- **Softmax is used as the activation function for multi-class classification problems**

Activation Functions -Softmax



Activation Functions

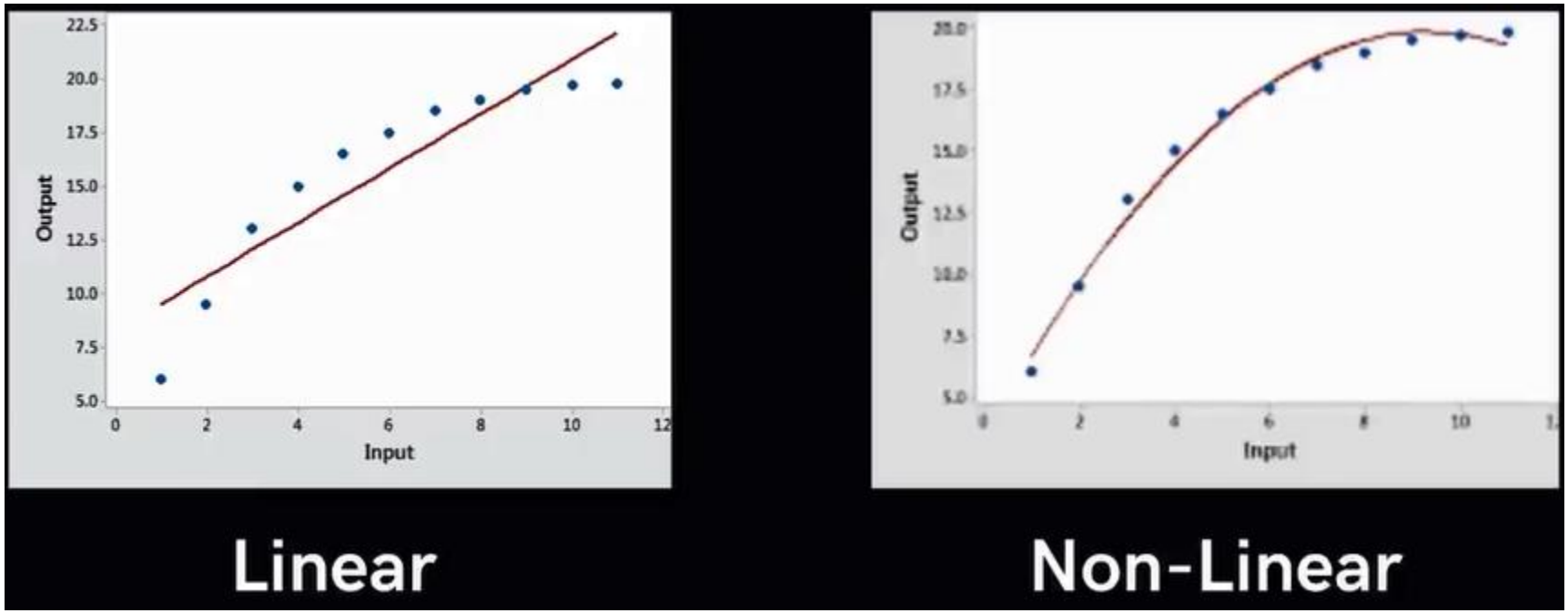
6. Softmax function



The softmax function can be used in a classifier only when the classes are mutually exclusive.

Which Activation Function to Use

Modeling Non-Linear Regression Data with a Linear & Non-Linear Curve



Loss/Cost Function

- After training the model, we need to see how badly our model is performing.
- Accuracy function tell us how well our model is performing, they do not provide us with insight on how to better improve them.
- Hence, **We need a Correctional Function** which can help us compute when our model is most accurate.

Loss Function

Quantify the deviation of the predicted output by the neural network to the expected output.

The way we actually compute this error is by using a Loss Function.

I. **Regression**: Squared Error, Huber Loss

II. **Binary Classification**: Binary Cross-Entropy, Hinge Loss

III. **Multi-Class Classification:** Multi-Class Cross-Entropy, Kullback Divergence

Regression Loss Function

1. Mean Squared Error
2. Mean Squared Logarithmic Error Loss
3. Mean Absolute Error Loss

Regression Loss Function

1. 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.**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Regression Loss Function

2. Mean Squared Logarithmic Error Loss

- It is used **to reduce the difference between the actual and predicted variable by taking the natural logarithm of the predicted variable and then the mean squared error.**
- It will overcome the problem possessed by the Mean Square Error Method.
- The model will now penalize less in comparison to the earlier method.

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2$$

n - the number of data points

y - the actual value of the data point. Also known as true value

ŷ - the predicted value of the data point. This value is returned by model.

Regression Loss Function

2. Mean Squared Logarithmic Error Loss

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2$$

Example- Let's say we have the following sets of numbers:

actual values of y	4	0	5	2
predicted values of ŷ	3.5	1	5	3

$$MSLE = \frac{(\log(4+1) - \log(3.5+1))^2 + (\log(0+1) - \log(1+1))^2 +}{4} =$$

$$= \frac{+(\log(5+1) - \log(5+1))^2 + (\log(2+1) - \log(3+1))^2}{4} \approx 0.1436$$

Regression Loss Function

3. Mean Absolute Error Loss

- Sometimes there may be some data points which far away from rest of the points **i.e. outliers.**
- In such cases **Mean Absolute Error Loss** will be appropriate to use as **it calculates the average of the absolute difference between the actual and predicted values.**

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Binary Classification Loss Function

1. **Binary Cross Entropy Loss**
2. **Hinge Loss**

Binary Classification Loss Function

1. 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.**

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Here, pi is the probability of class 1, and (1-pi) is the probability of class 0.

Binary Classification Loss Function

2. Hinge Loss

- This type of loss is used when **the target variable has 1 or -1 as class labels.**
- It penalizes the model when there is a difference in the sign between the actual and predicted class values.
- These are particularly **used in SVM models.**

It is defined according to the following formula, where t is the actual outcome (either 1 or -1), and y is the output of the classifier.

$$\mathbf{l(y)=Max(0,1-t\cdot y)}$$

Multi-Class Classification Loss Function

Multi-Class Classification Loss function is used, where the **target variable has more than two classes.**

1. **Categorical Cross Entropy Loss**
2. **Kullback Leibler Divergence Loss**

Multi-Class Classification Loss Function

1. Categorical Cross Entropy Loss

These are similar to binary classification cross-entropy, used for multi-class classification problems.

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Multi-Class Classification Loss Function

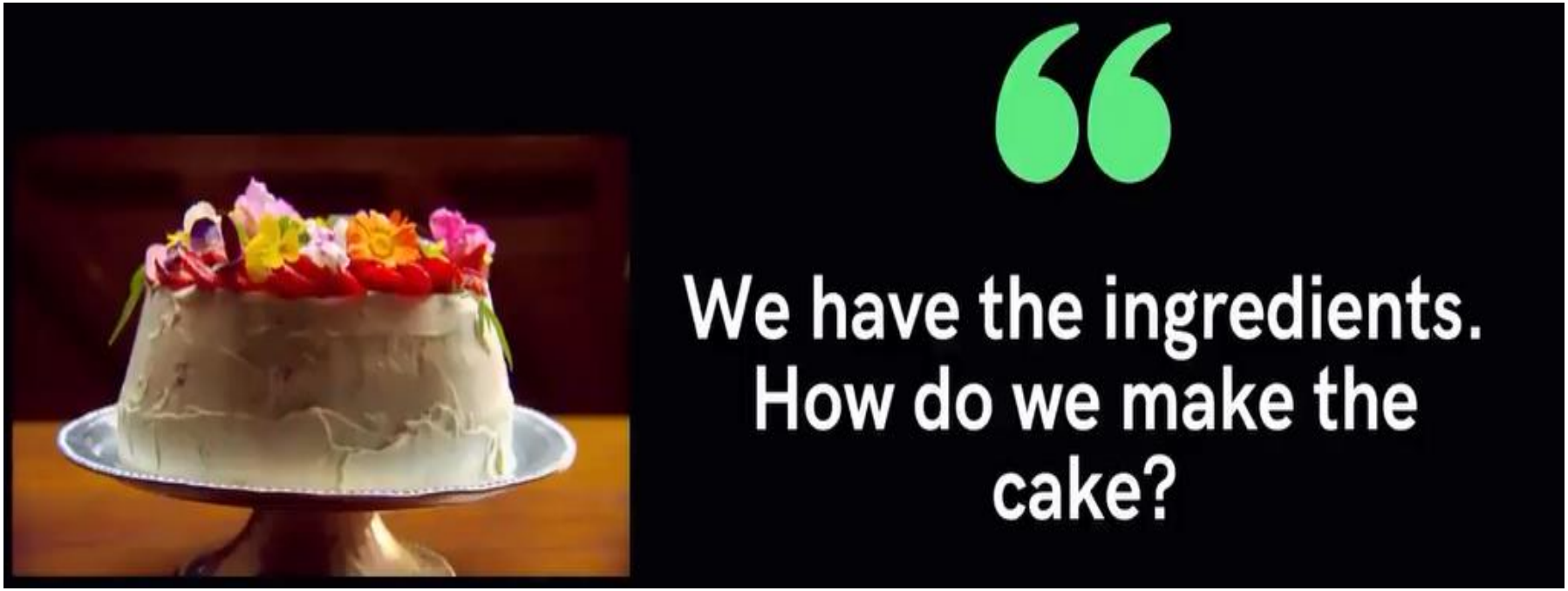
2. Kullback Leibler Divergence Loss

- Kullback Leibler Divergence Loss calculates how much a given distribution is away from the true distribution.
- These are **used to carry out complex operations like autoencoder where there is a need to learn the dense feature representation.**

How to Reduce the Loss

During training, we adjust the parameters to minimize the loss Function and make our model as optimized as possible.

But, How do we do that?



How to Reduce the Loss

Optimizers are used to increase the efficiency, minimize the loss and update the variable.

Optimizer modifies each variable according to the magnitude of the derivative of loss with respect to that variable.

Gradient Descent Optimizer can be used.

Gradient Descent is an iterative optimization algorithm, used to find the minimum value for a function.

(Gradient means-The degree at which a road, etc. goes up or down. Descent means-a movement down to a lower place.)

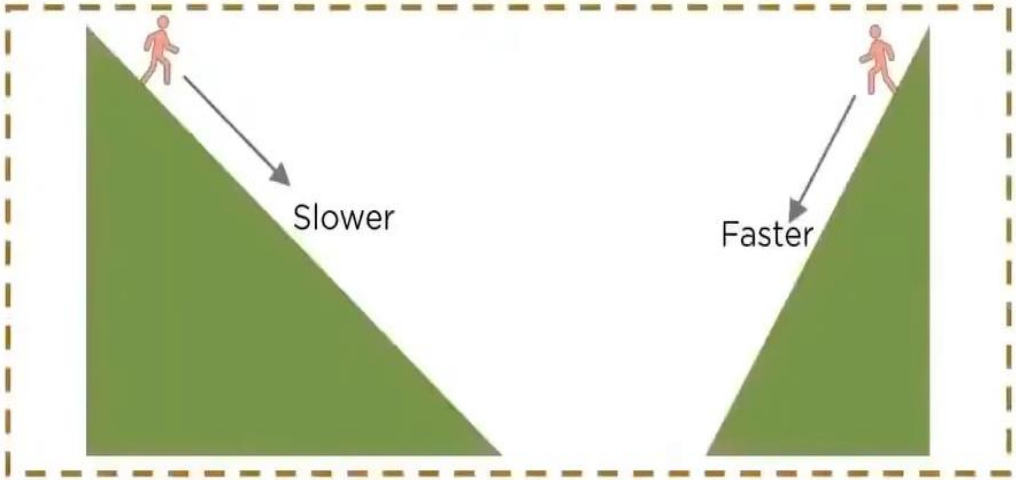
Optimizers

- Tie together the loss function and model parameters by **Updating** the network based on the output of the loss function.
- **Loss Functions guide optimizers.**

Gradient Descent

Gradient Descent is an optimization algorithm for finding the minimum of a function

A person trying to reach the base of a mountain



The person will take more time to reach the base of the mountain if the slope is gentle and will come down faster if the slope is steep. Likewise, a Neural Net will train slowly if the Gradient is small and it will train quickly if the Gradient is large.

Consider **every step as learning rate** and **position as weight.** Find the descent space and start walking. **Optimizer check in which direction we should update the weights to minimize the loss.**

Gradient Descent

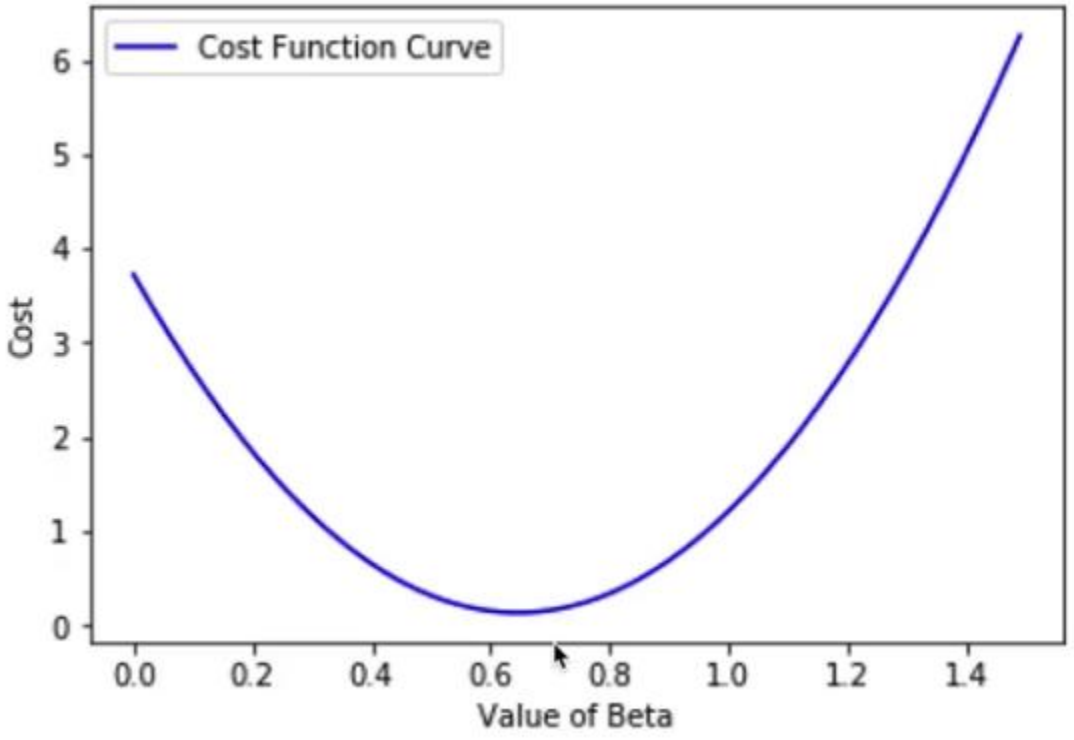
- **Gradient descent is an optimization algorithm** which is **commonly-used to train machine learning models and neural networks.**
- **Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer**, gauging its accuracy with each iteration of parameter updates.
- Until the function is close to or equal to zero, the model will continue to adjust its parameters to yield the smallest possible error.

Gradient Descent

let's consider a linear model, **Y_pred=mx+c. Or Y_pred= ßx+b**

In this equation, Y_pred represents the output. 'c' is the intercept and 'm' is the slope whereas x is the input value. **Cost Function-MSE**

For a linear model, we have a convex cost function as shown in the image below that looks like a bowl.



Gradient Descent

Initially, you can **start with randomly selected values of the parameters c and m** and make the predictions Y_pred.

The next step is to calculate the error i.e the difference between the original and the predictions.

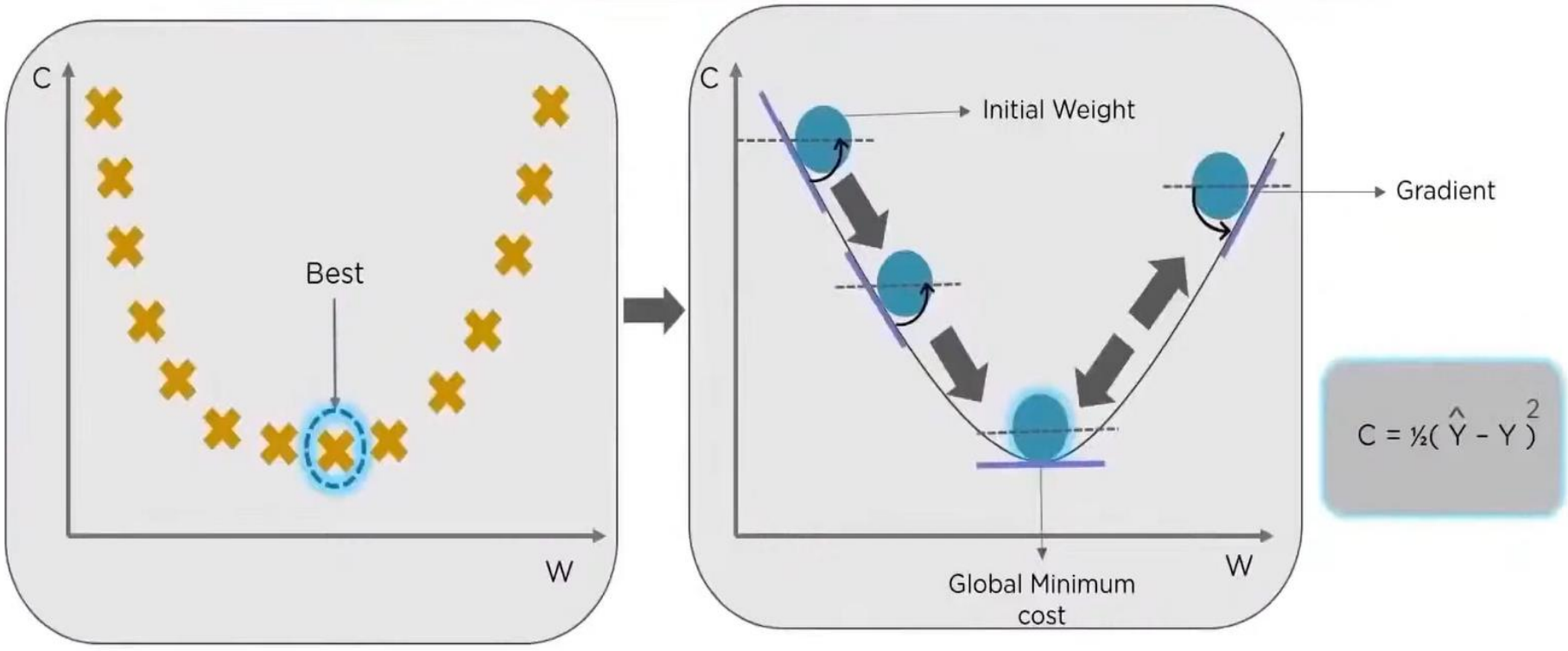
The predicted values with these parameters will land you anywhere on this cost function curve.

Now the task is to update the **value of c and m to minimize the error** i.e coming down on this curve to the lower points.

This process will continue until we reach the lowest point of this curve **i.e the minimum cost**

Gradient Descent

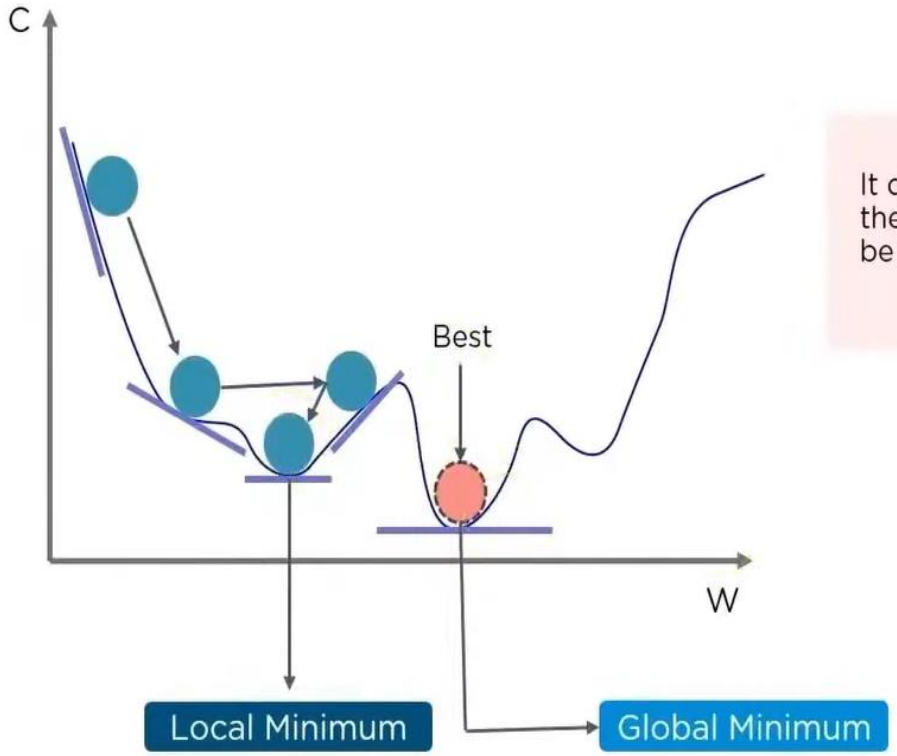
Gradient Descent is an optimization algorithm for finding the minimum of a function



Batch Gradient Descent & Stochastic Gradient

A	B	C
S.NO.	**BATCH GRADIENT DESCENT**	**STOCHASTIC GRADIENT DESCENT**
1	Computes gradient using the whole Training sample	Computes gradient using a single Training sample
2	Slow and computationally expensive algorithm	Faster and less computationally expensive than Batch GD
3	Not suggested for huge training samples.	Can be used for large training samples.
4	Deterministic in nature.	Stochastic in nature.
5	Gives optimal solution given sufficient time to converge.	Gives good solution but not optimal.
6	No random shuffling of points are required.	The data sample should be in a random order, and this is why we want to shuffle the training set for every epoch.
7	Can't escape shallow local minima easily.	SGD can escape shallow local minima more easily.
8	Convergence is slow.	Reaches rthe convergence much faster.

Stochastic Gradient Descent



It does not require the cost function to be convex

Takes the rows one by one, runs the neural network and then adjusts the weights

Helps you avoid the local minimum as it performs 1 iteration at a time

Parameters and Hyperparameters

What are Model Parameters

- Variables internal to the neural network.
- Value can be estimated right from the data.

Model Parameters

- Required by the model to make predictions
- Values define the skill of the model
- **Estimated directly from data**
- **Not set manually**
- Saved as part of the learned model
- Examples: Weights, Biases etc.

What are Model Hyperparameters?

Configurations external to the neural network.

Value cannot be estimated right from the data.

Model Hyperparameters

No clear-cut way to find the best value.

When a DL algorithm is tunes, you are really tuning the Hyperparamters.

If you have to manually specify a parameter, it is a hyperparamter.

Example:-Learning rate, C in SVMs

Summary

Summary

Model Parameters $\longrightarrow$ Estimated from the data

Model Hyperparameters $\longrightarrow$ Can’t be estimated from the data.

Model hyperparameters are often referred to as parameters because they are the parts of the machine learning that must be set manually and tuned.

Epochs, Batches, Batch Sizes and Iterations

Epochs, Batches, Batch Sizes & Iterations

- Need these terms only if the dataset is Large.

Break down the dataset into smaller "chunks" and feed those chunks to the neural network one-by-one.

Epochs, Batches, Batch Sizes and Iterations

Epochs

- When the ENTIRE dataset is passed forward and backward through the neural network only ONCE.
- We use multiple epochs to help our model generalize better.
- No right number of epochs

Iterations

Number of batches needed to complete one epoch.

Alternatively,

Number of Batches = Number of Iterations for one epoch

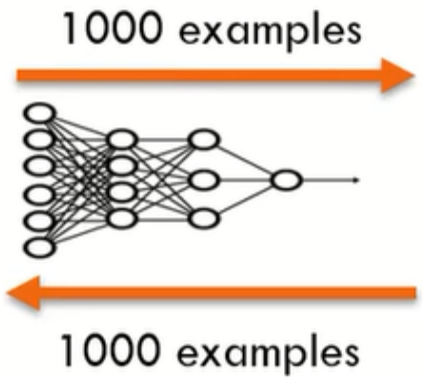
Epochs, Batches, Batch Sizes and Iterations

Suppose you have a dataset of 34000 training examples and you divide The dataset into batches of 500.

To compete 1 epoch, it would have taken 68 iterations.

Total Training examples = 1000

1 Epoch



After the first epoch, the weights will be decent

By feeding the training data To neural network again and again , we can improve the weights further

Epochs, Batches, Batch Sizes and Iterations

Batch size is a hyperparameter

Batch: Since one epoch is too big to feed to the Computer at once we divide it in several smaller batches.

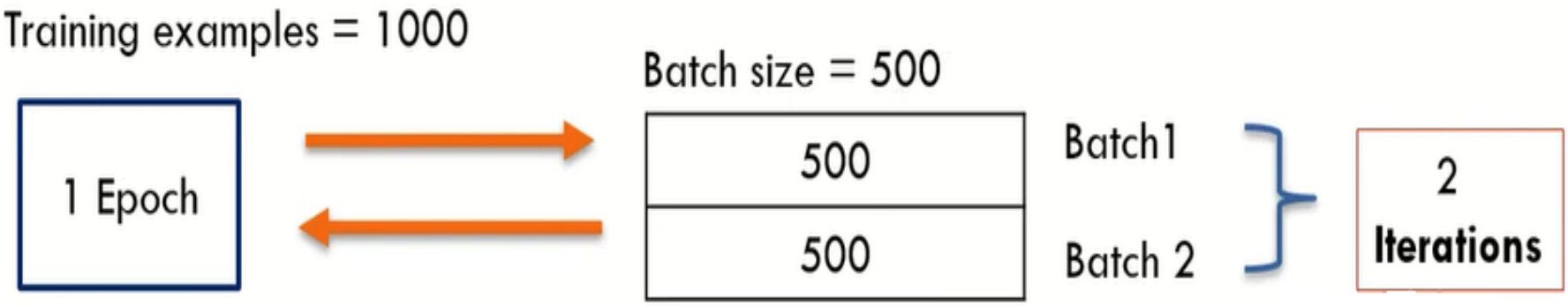
Batch_Size: Total number of training examples present in as single batch.

Iterations

Iterations are the number of batches needed to complete one epoch

Epochs, Batches, Batch Sizes and Iterations

Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.



Hidden Layers and Activation Function

How many hidden layers should I choose?

Which Activation function to pick?

Experiment----Experiment--------Experiment

Core Problem in Deep Learning

Model should perform well on training data AND new test data.

Most common problem faced is Overfitting.

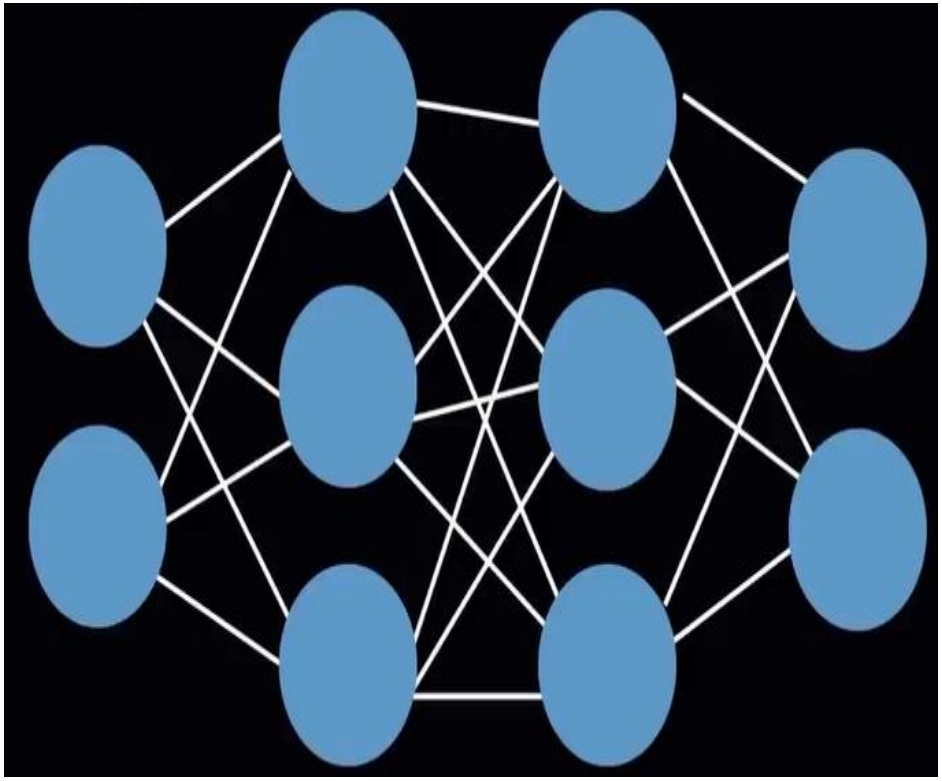
Tackling Overfitting

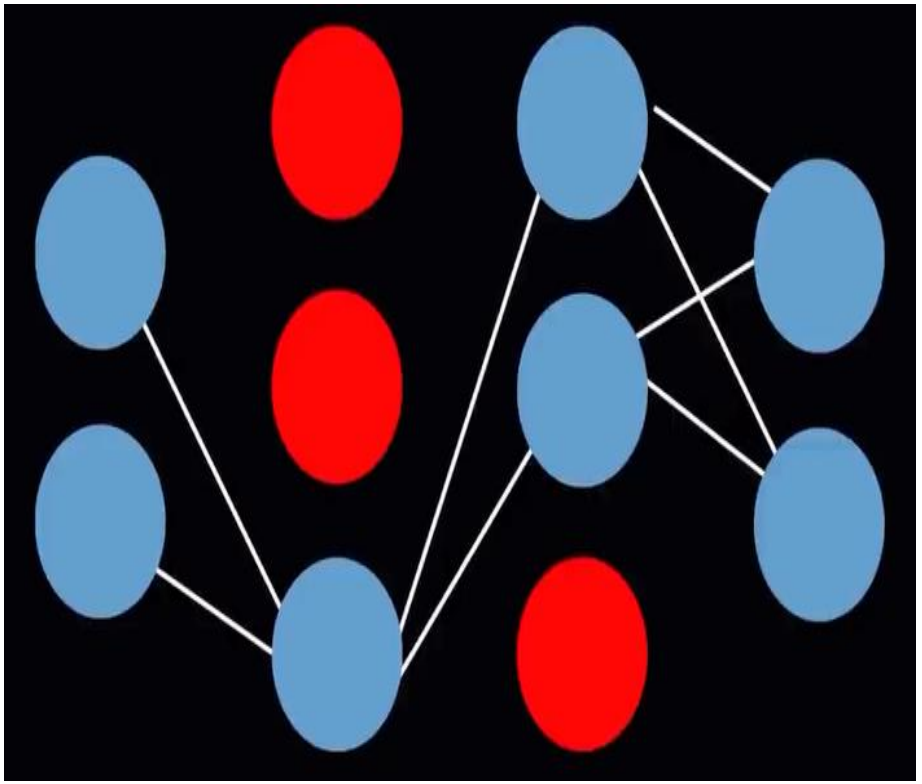
1. **Dropout**
2. **Augmentation**
3. **Early Stopping**

Core Problem in Deep Learning

Dropout

Dropout randomly removes some nodes & their connections





Core Problem in Deep Learning

Dataset Augmentation

More Data $\longrightarrow$ **Better Model**

Apply transformations on the existing dataset to get synthesize more data.

Original Image

Rotated Image

Scaled Image

Core Problem in Deep Learning

Early Stopping

Training error decreases steadily

But, validation error increases after a certain point.

Neural Network Architectures

Fully-Connected Feed Forward Neural Networks

Each Neuron is connected to every subsequent layer with no backward connections.

Neurons have activations functions

- Linear
- Sigmoid
- Tanh
- ReLU

} **Non-Linear**

Neural Network Architectures

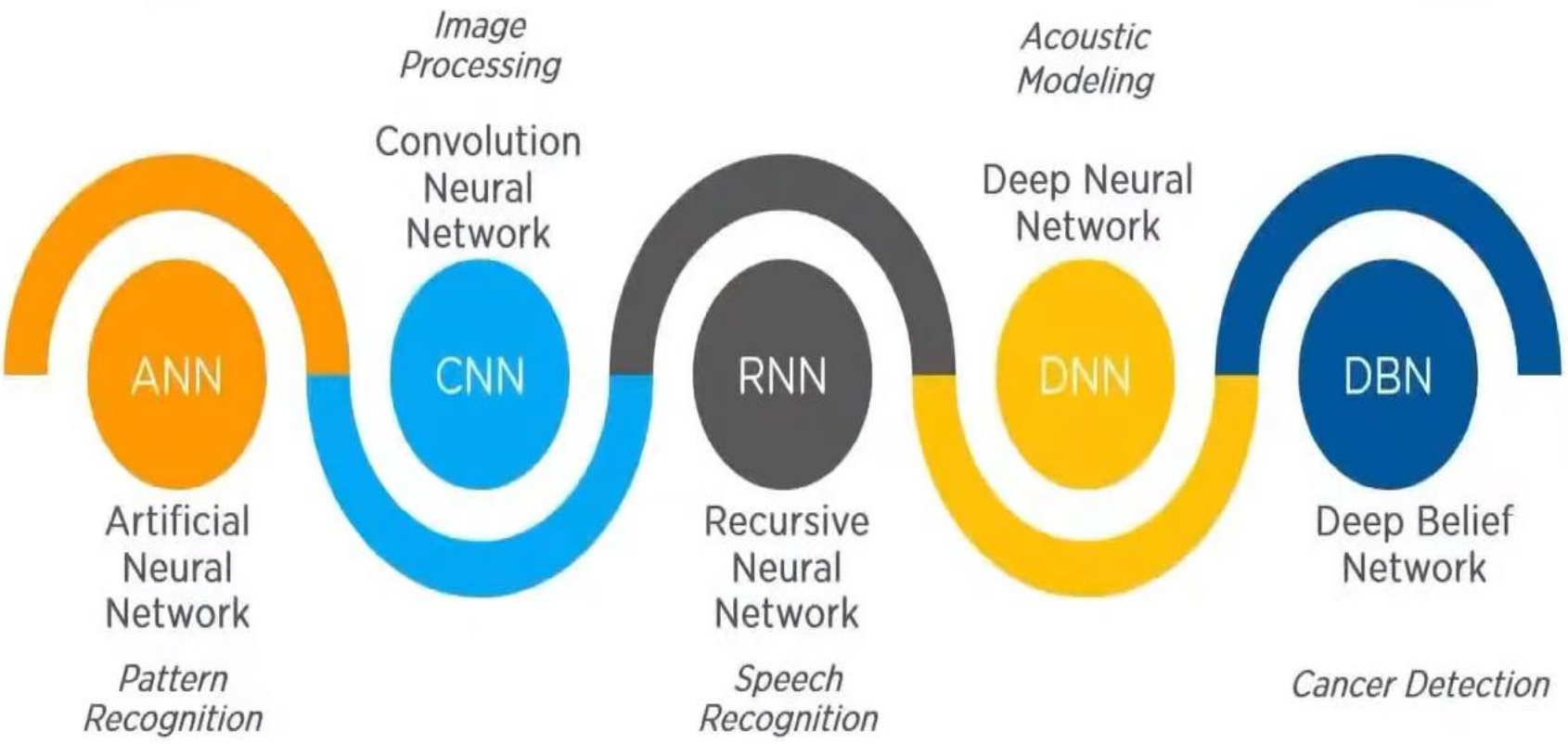
Fully-Connected Feed Forward Neural Networks

- Inputs
- Outputs
- Hidden Layers
- Neurons per hidden Layer
- Activation functions

- **More Neurons** $\longrightarrow$ **larger Computational Resources**

Neural Network Architectures

Neural Networks are mainly classified into Five types.



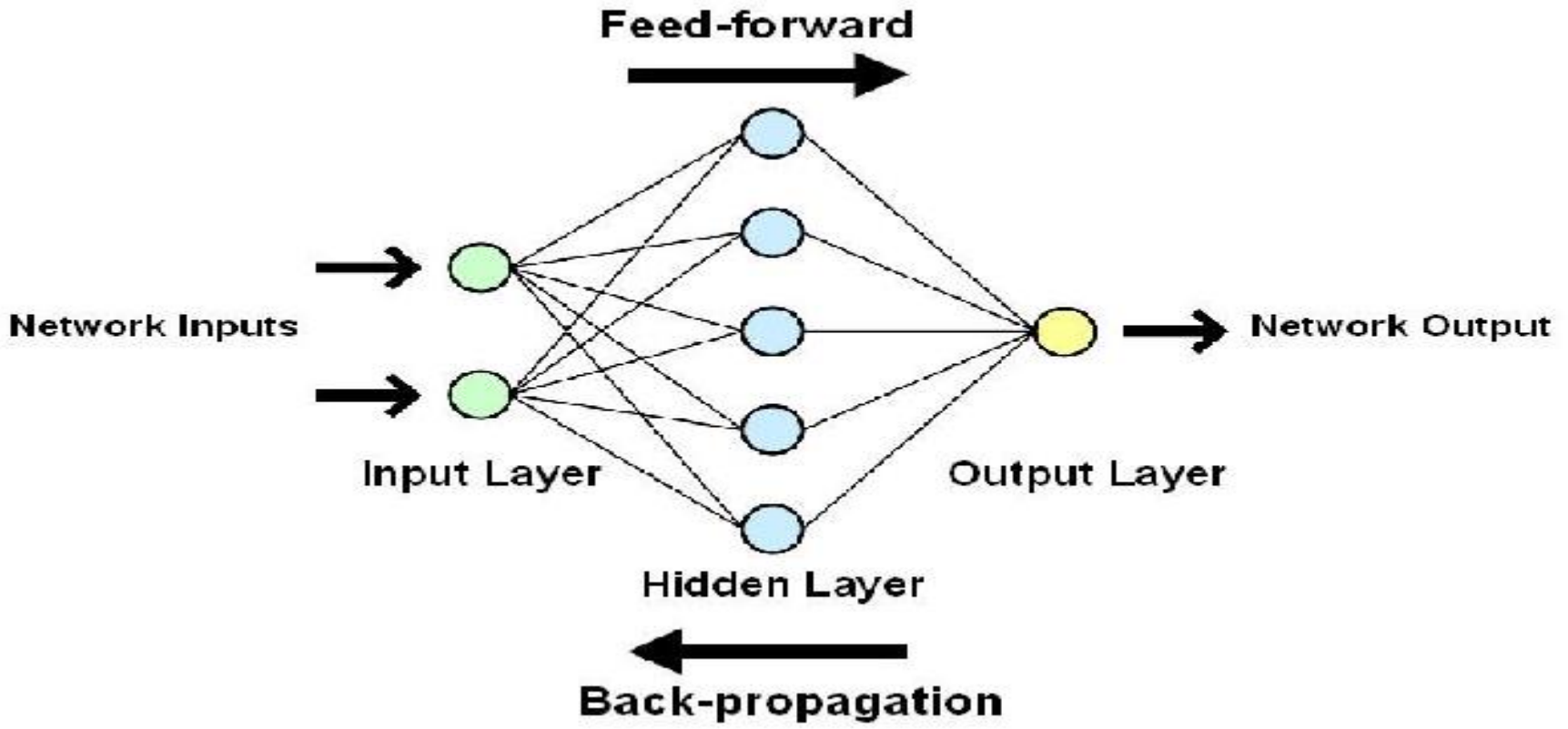
- Deep Belief Networks (DBN) is an unsupervised learning algorithm.
- It consist of two different types of neural networks – Belief Networks and Restricted Boltzmann Machines.
- DBN is also a multi-layer belief network.

Artificial Neural Network

- **Artificial Neural Networks (ANN)** are algorithms **based on brain function** and are used to model complicated patterns and forecast issues.
- The Artificial Neural Network (ANN) is a deep learning method that arose from the concept of the human brain Biological Neural Networks.
- The development of ANN was the result of an attempt to replicate the workings of the human brain.
- The workings of ANN are extremely similar to those of biological neural networks, although they are not identical.
- ANN algorithm **accepts only numeric and structured data.**
- **Convolutional Neural Networks (CNN) and Recursive Neural Networks (RNN)** are used to **accept unstructured and non-numeric data forms such as Image, Text, and Speech.**

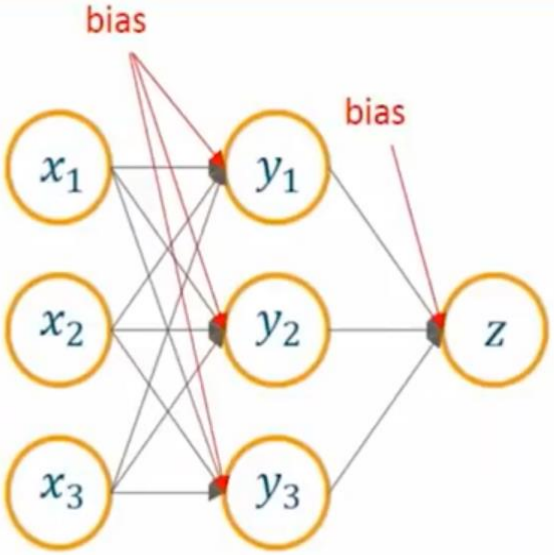
Artificial Neural Networks Architecture

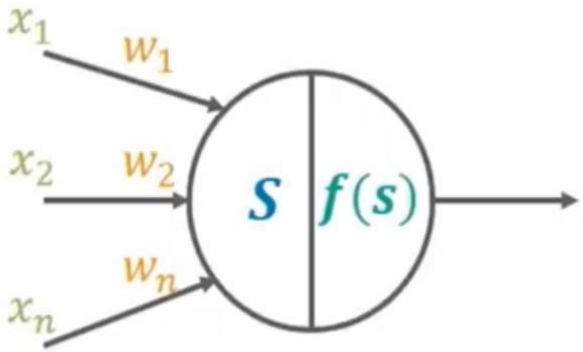
- There are three layers in the network architecture: **the input layer, the hidden layer (more than one), and the output layer.**
- Because of the numerous layers are sometimes referred to as the **MLP (Multi-Layer Perceptron).**



Multilayer perceptron

A Multi-layer Perceptron has the same structure of a single layer perceptron but with one or more hidden layers and is thus considered a deep neural network.





Summation:

$$S = \sum_{i=1}^{n} w_i * x_1$$

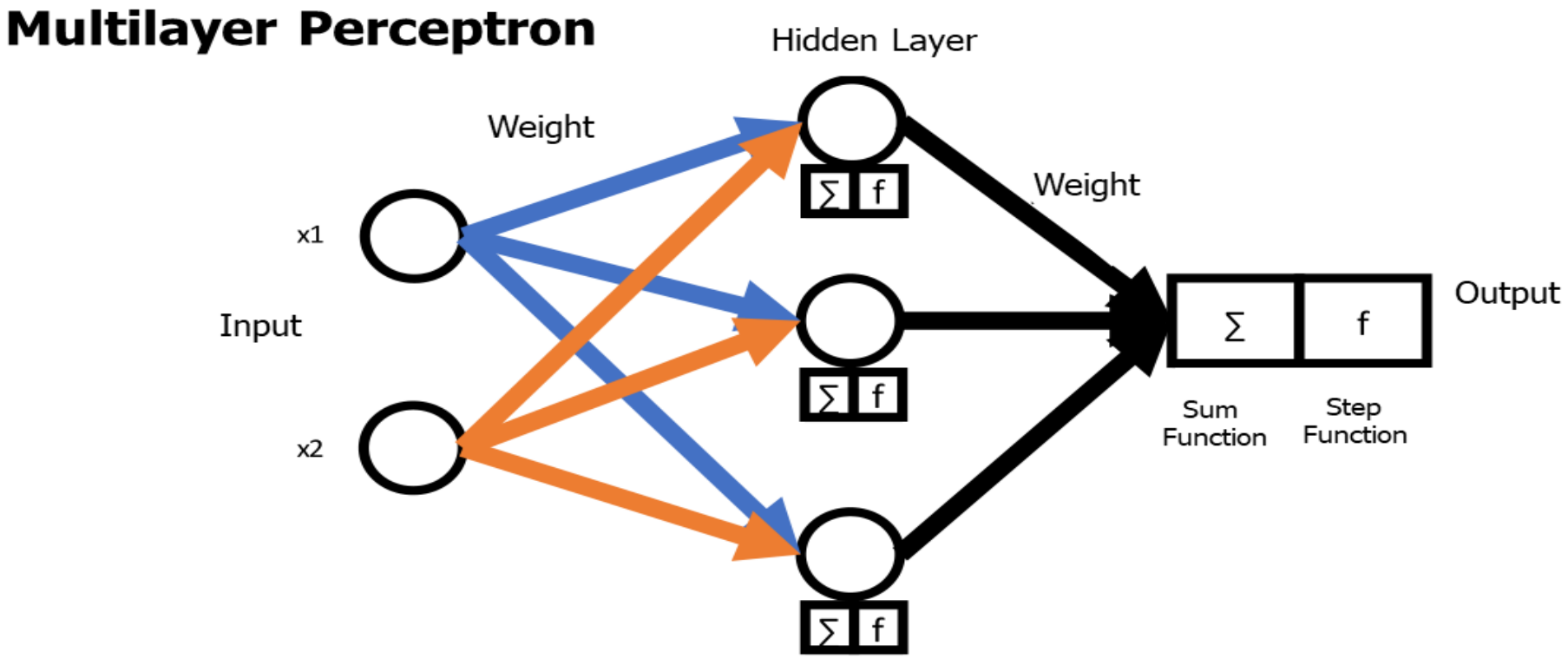
Transformation:

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

Multilayer Perceptron

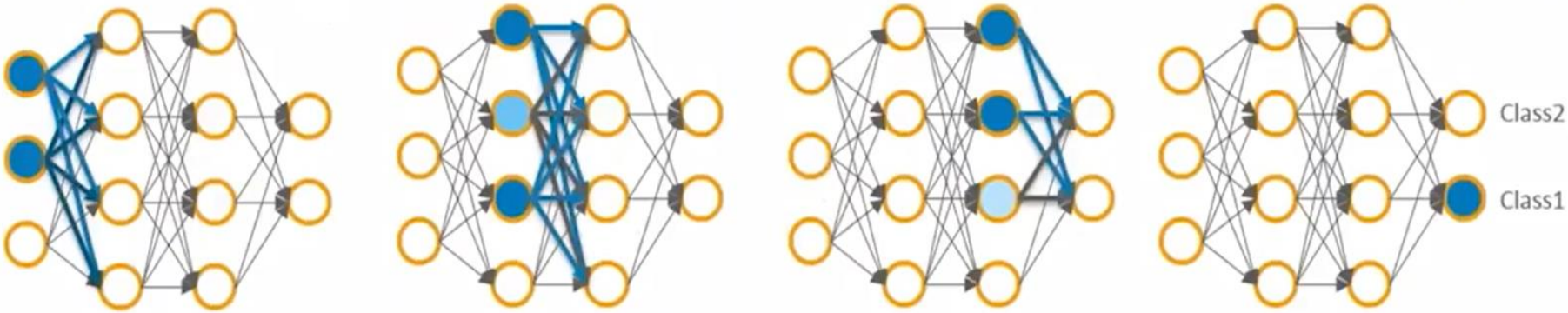
Multilayer networks have **more than one hidden**, which is one of the characteristics of Deep Learning. In conjunction with activation function, sigmoid function.

We visualized below the processing of **a multilayer ANN**, which from the input values in the ANN, these values are multiplied by their respective weights, and the output is the result of processing the weights with the activation function.



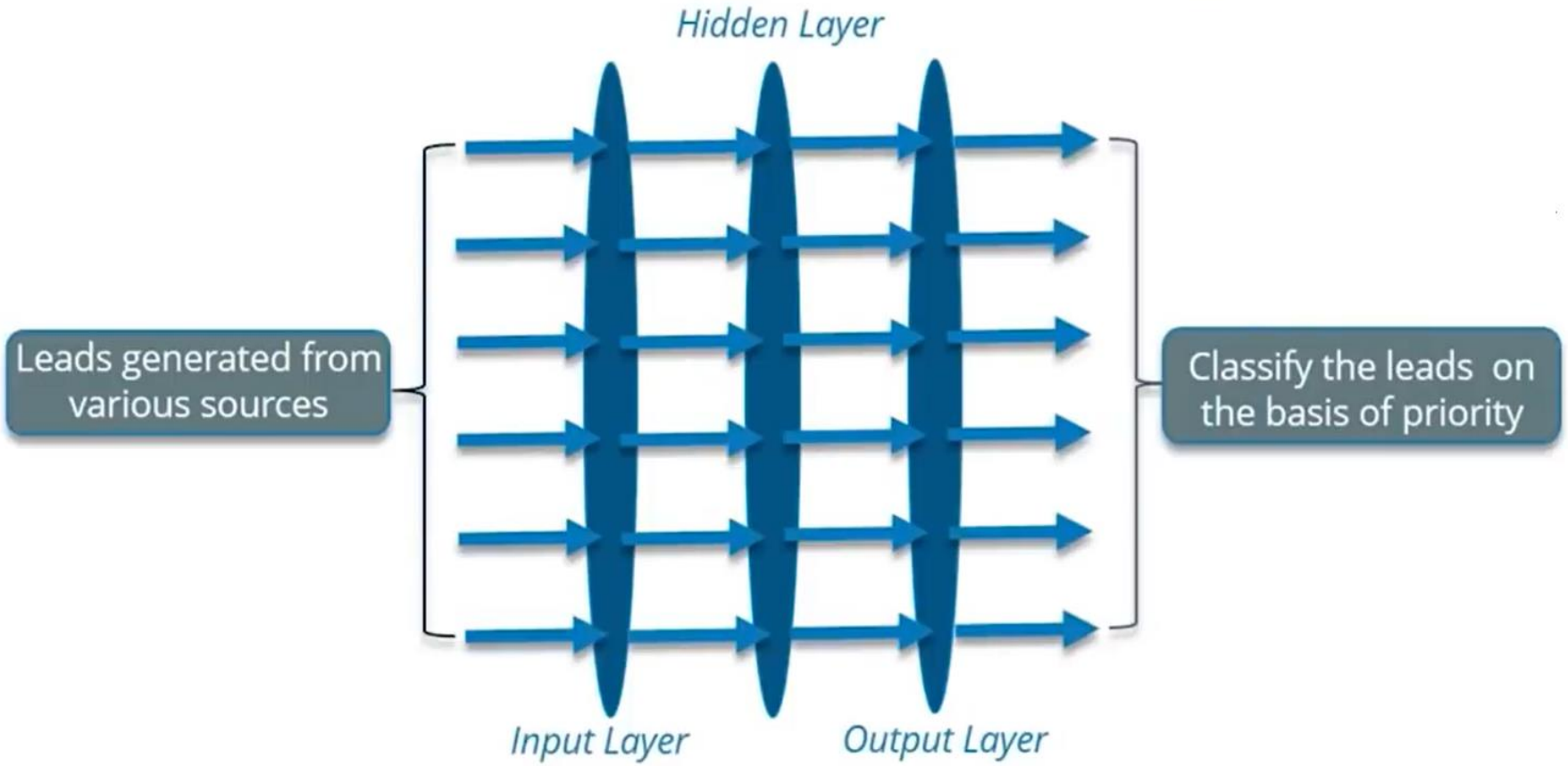
Multilayer perceptron-How it works

- The weights between the units are the primary means of long-term information storage in neural networks
- Updating the weights is the primary way the neural network learns new information



A set of inputs is passed to the first hidden layer, the activations from that layer are passed to the next layer and so on, until you reach the output layer

Multilayer perceptron



Learning Process of Neural Network

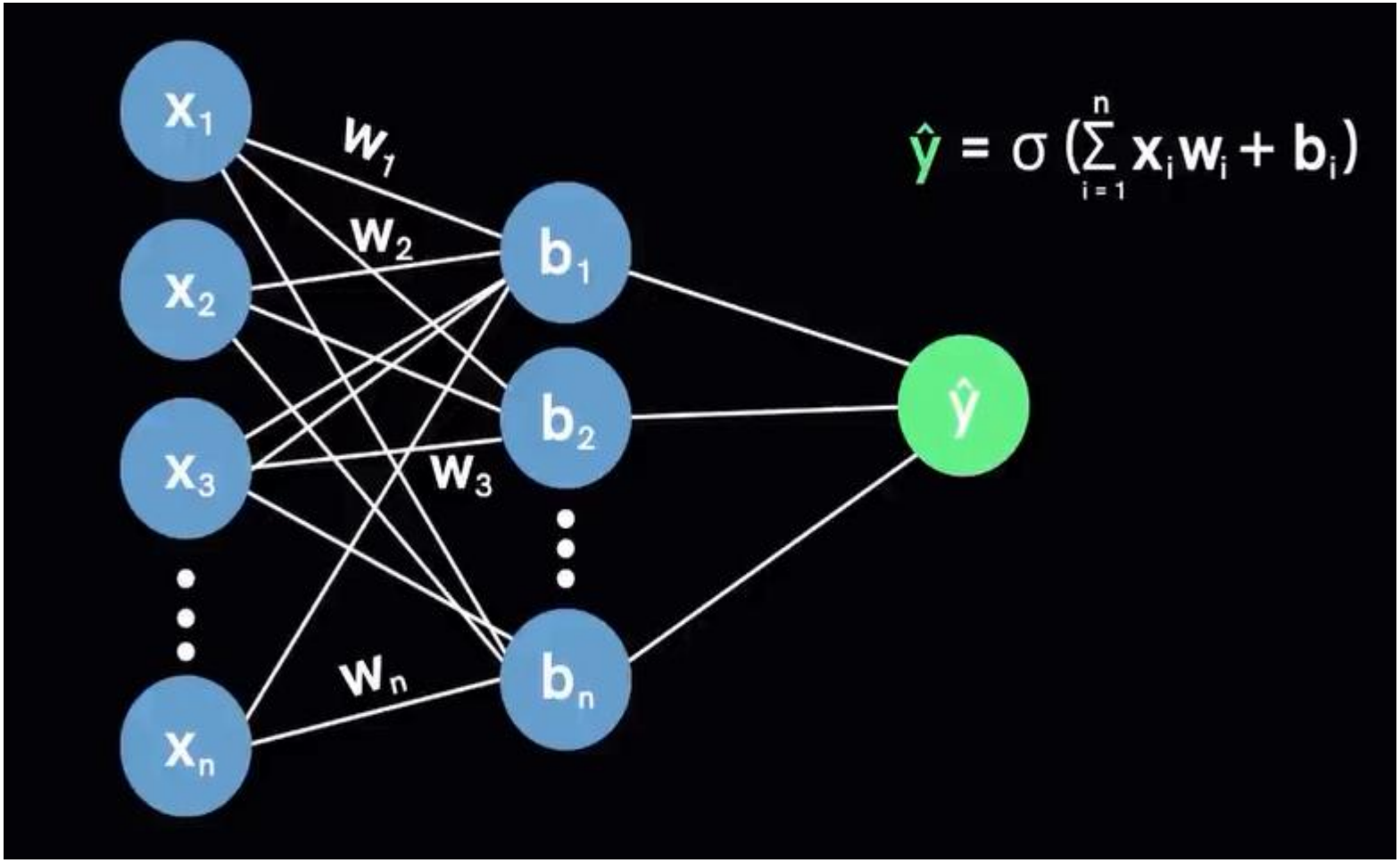
Learning Process

1. Forward Propagation
2. Back Propagation

Learning Process of Neural Network

Learning Process

1. Forward Propagation



Learning Process of Neural Network

Learning Process

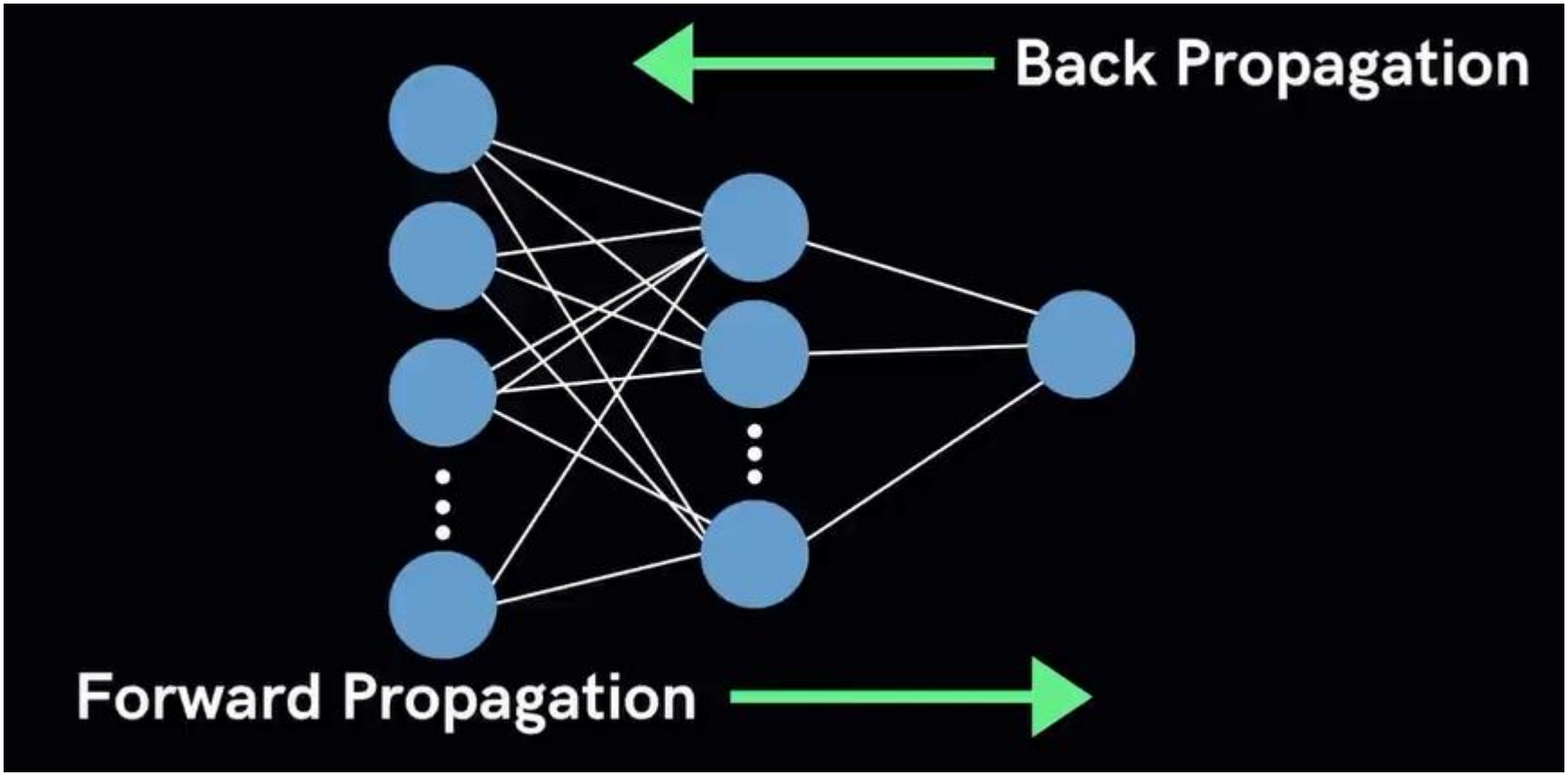
Forward Propagation

1. Weight:-How important is that Neuron
2. Bias:-Allows for the shifting of activation function to the right or left.

Back Propagation

Learning Process

Back Propagation



Back Propagation

Learning Process

Back Propagation

Loss functions help quantify the deviation from the expected output.

What is Backpropagation

In order to classify the leads on the basis of priorities, we need to provide the maximum weight to the most important lead.
For that we can calculate the difference between the actual output and the desired output.
According to that difference we can update the weights.

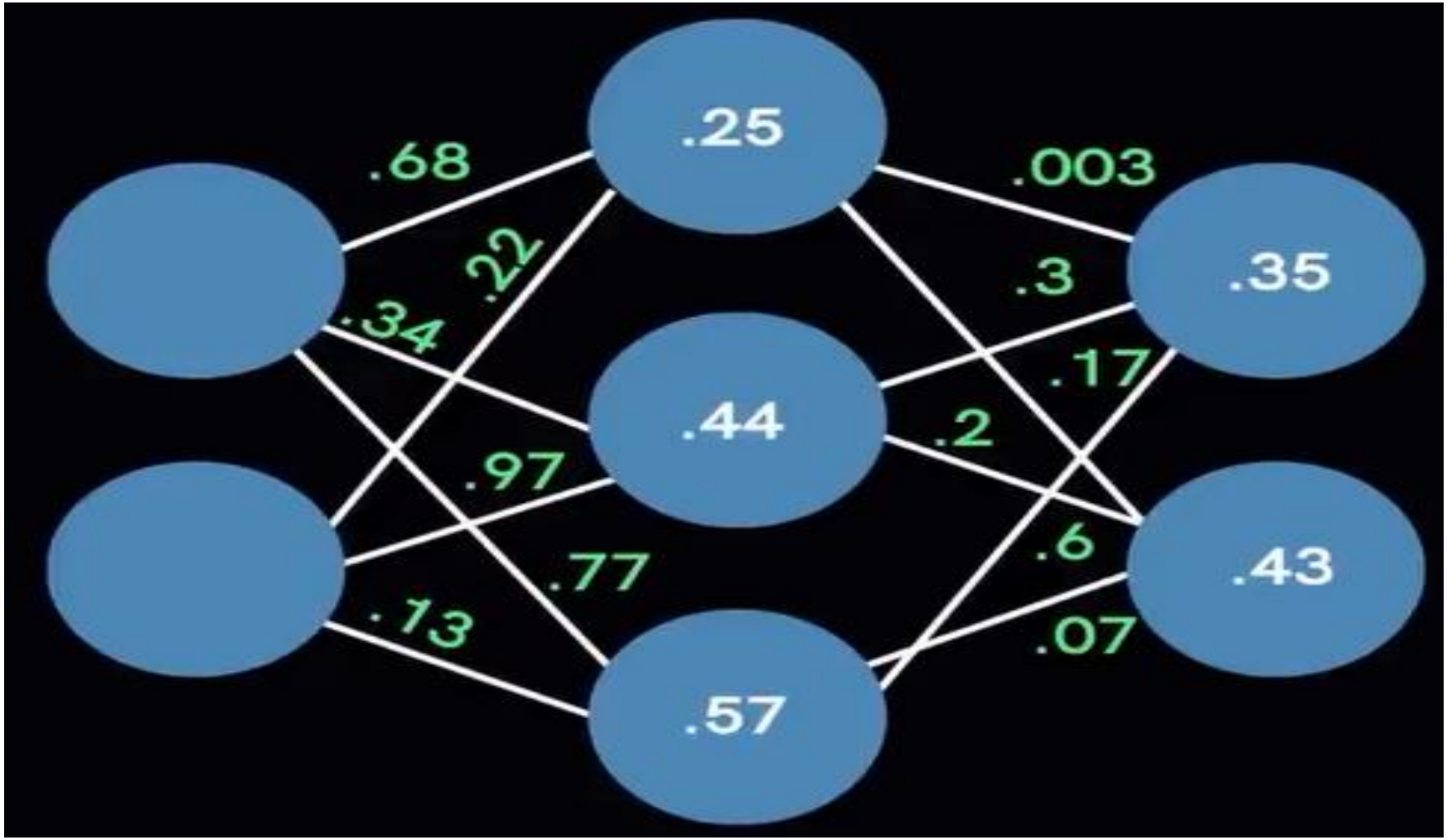
The Backpropagation algorithm is a supervised learning method for Multilayer Perceptron.

Learning Process of Neural Network-Car Truck Data

#	Weight	Goods	Car/ Truck
1	15	02	Car
2	34	67	Truck
3	42	54	Truck

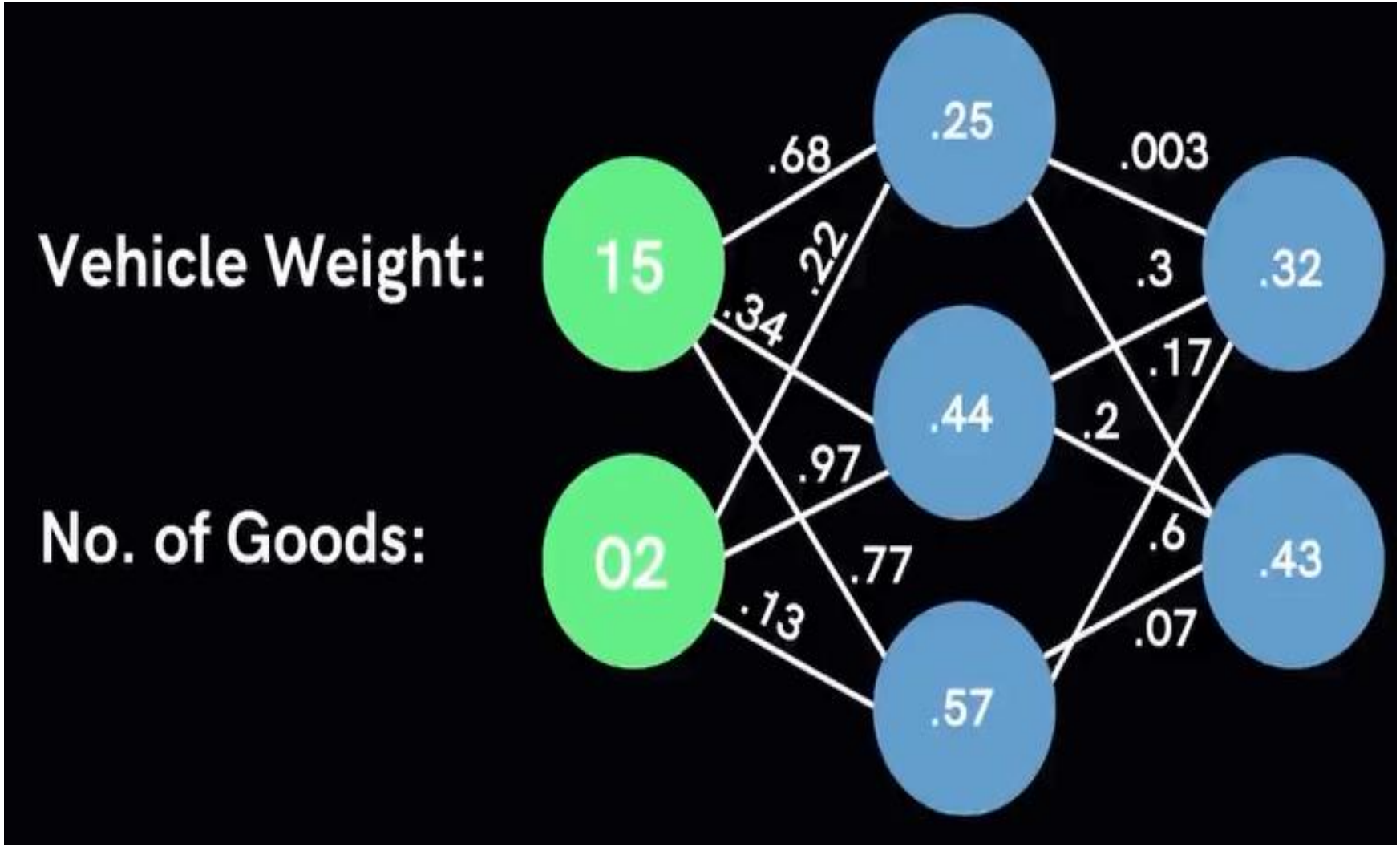
Learning Process of Neural Network-Car Truck Data

Randomly Initialize Parameters (Weight and Bias)



Learning Process of Neural Network-Car Truck Data

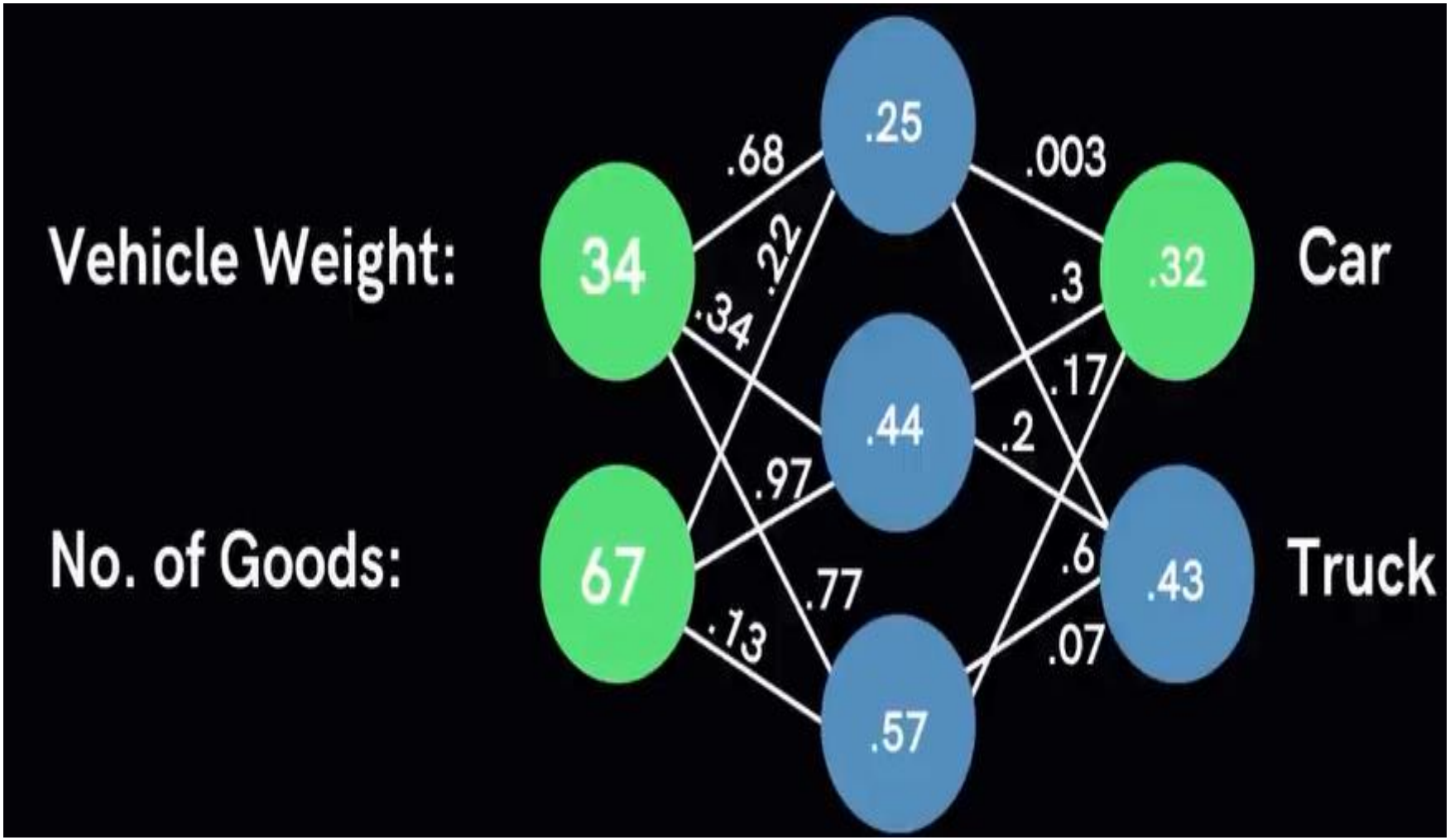
Randomly Initialize Parameters (Weight and Bias)



Learning Process of Neural Network-Car Truck Data

1. Use a Loss Function
2. Go Backwards and adjust weights and biases
3. Values adjusted to better fit model

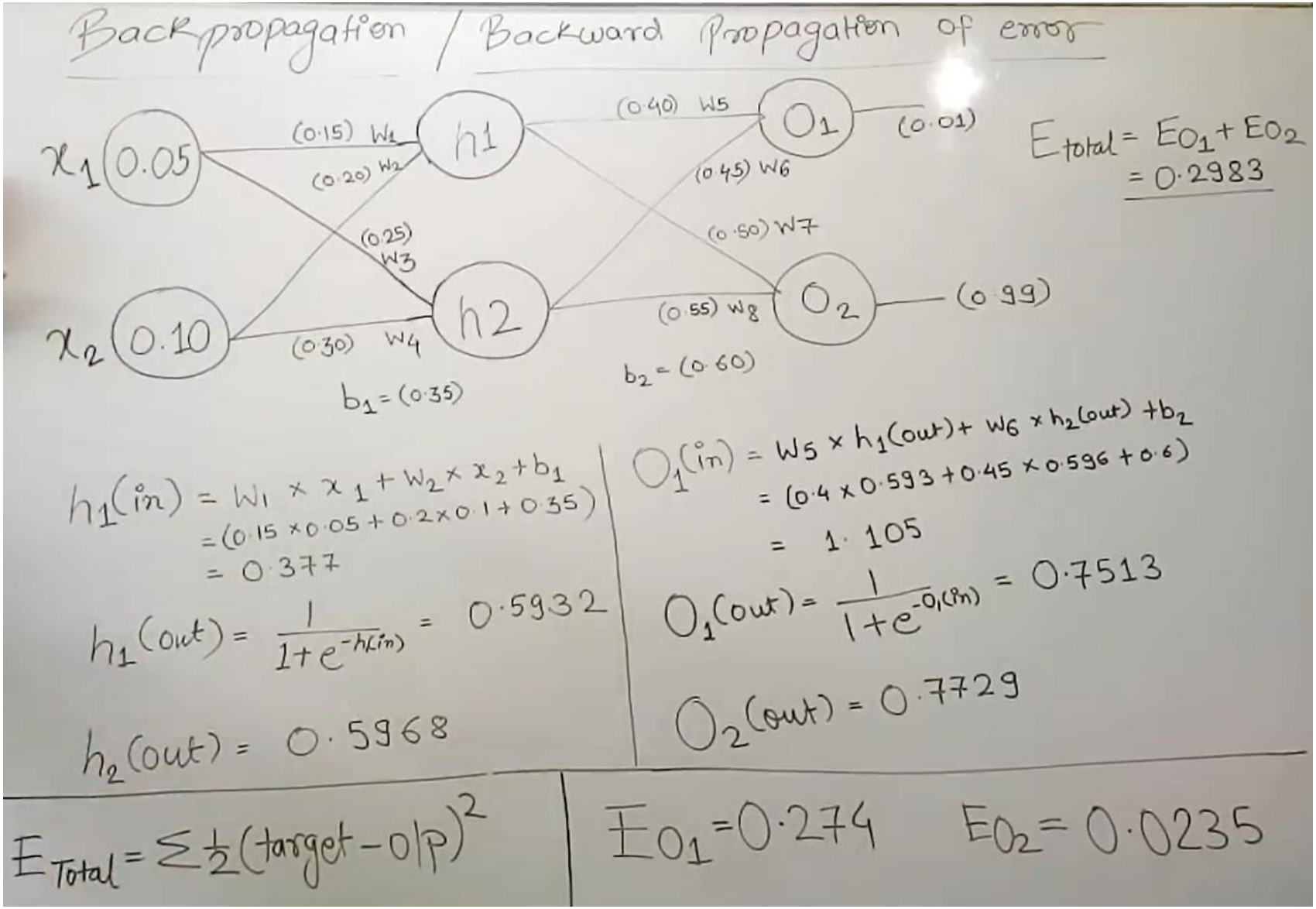
Learning Process of Neural Network-Car Truck Data



Learning Process of Neural Network-Car Truck Data

- Initialize Parameters with Random Values.
- Feed input data to network.
- Compare predicted value with expected value & calculate loss.
- Perform Backpropagation to propagate this loss back through the network.
- Update parameters based on the loss.
- Iterate previous steps till loss is minimized.

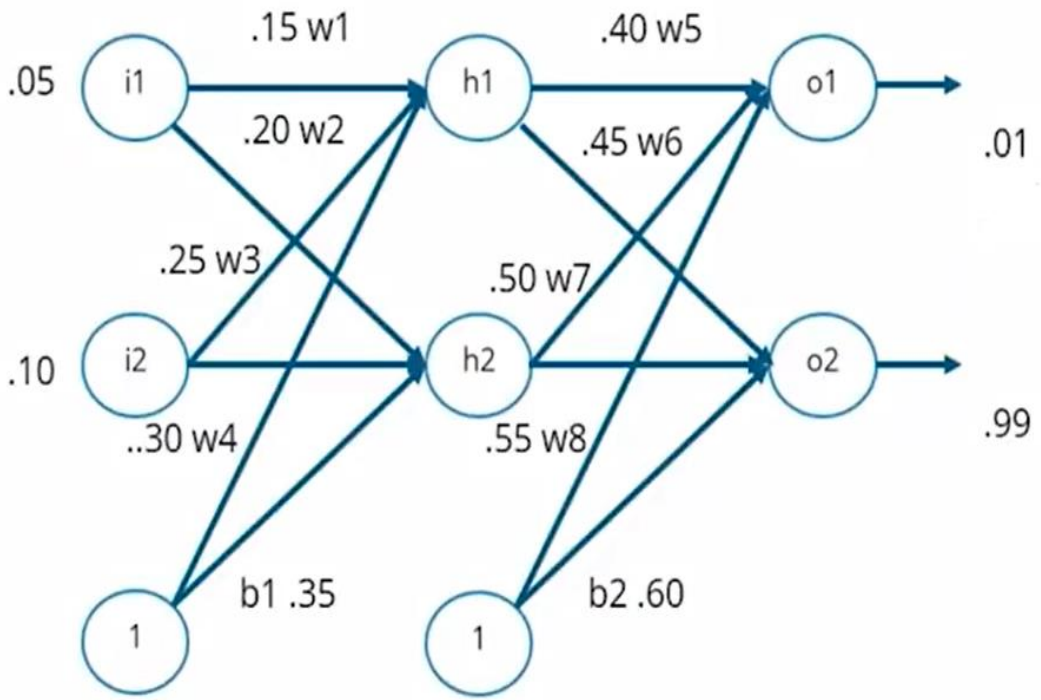
Backpropagation with Example



Backpropagation Learning Algorithm

Consider the following neural network with:

- two inputs
- two hidden neurons
- two output neurons
- two biases



Backpropagation Learning Algorithm

Net Input For h1:

net h1 = w1*i1 + w2*i2 + b1*1 → net h1 = 0.15*0.05 + 0.2*0.1 + 0.35*1 = 0.3775

Output Of h1:

out h1 = $1/1+e^{-\text{net h1}}$ → $1/1+e^{.3775}$ = 0.593269992

Output Of h2:

out h2 = 0.596884378

Backpropagation Learning Algorithm

We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Output For o1:

net o1 = w5*out h1 + w6*out h2 + b2*1 → 0.4*0.593269992 + 0.45*0.596884378 + 0.6*1 = 1.105905967

Out o1 = $1/1+e^{-\text{net o1}}$ → $1/1+e^{-1.105905967}$ = 0.75136507

Output For o2:

Out o2 = 0.772928465

Backpropagation Learning Algorithm

Error For o1:

$E_{o1} = \Sigma 1/2(target - output)^2$ → $\frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$

Error For o2:

$E_{o2} = 0.023560026$

Total Error:

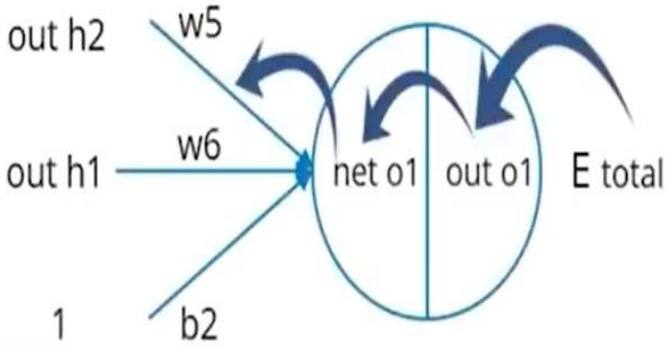
$E_{total} = E_{o1} + E_{o2}$ → $0.274811083 + 0.023560026 = 0.298371109$

Backpropagation Learning Algorithm

Update each of the weights in the network so that they cause the actual output to be closer the target output.

Consider w5, we will calculate the change in total error w.r.t w5: $\frac{\delta Etotal}{\delta w5}$

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



The chain rule *tells us how to find the derivative of a composite function.*

Backpropagation Learning Algorithm

How much does the total error change with respect to the output?

$$E_{total} = 1/2(\text{target o1} - \text{out o1})^2 + 1/2(\text{target o2} - \text{out o2})^2$$

Find out the partial derivative of Etotal w r t outo1.

$$\frac{\delta Etotal}{\delta out\ o1} = -(\text{target o1} - \text{out o1}) = -(0.01 - 0.75136507) = 0.74136507$$

How much does the output o1 change w.r.t its total net input?

$$\text{out o1} = 1/1+e^{-neto1}$$

$$\frac{\delta out\ o1}{\delta net\ o1} = \text{out o1 (1 - out o1)} = 0.75136507\ (1 - 0.75136507) = 0.186815602$$

Find out the partial derivative of out o1 w r t net o1.

Backpropagation Learning Algorithm

How much does the total net input of o1 changes w.r.t w5?

net o1 = w5 * out h1 + w6 * out h2 + b2 * 1

$$\frac{\delta net\ o1}{\delta w5} = 1 * \text{out h1}\ w5^{(1-1)} + 0 + 0 = 0.593269992$$

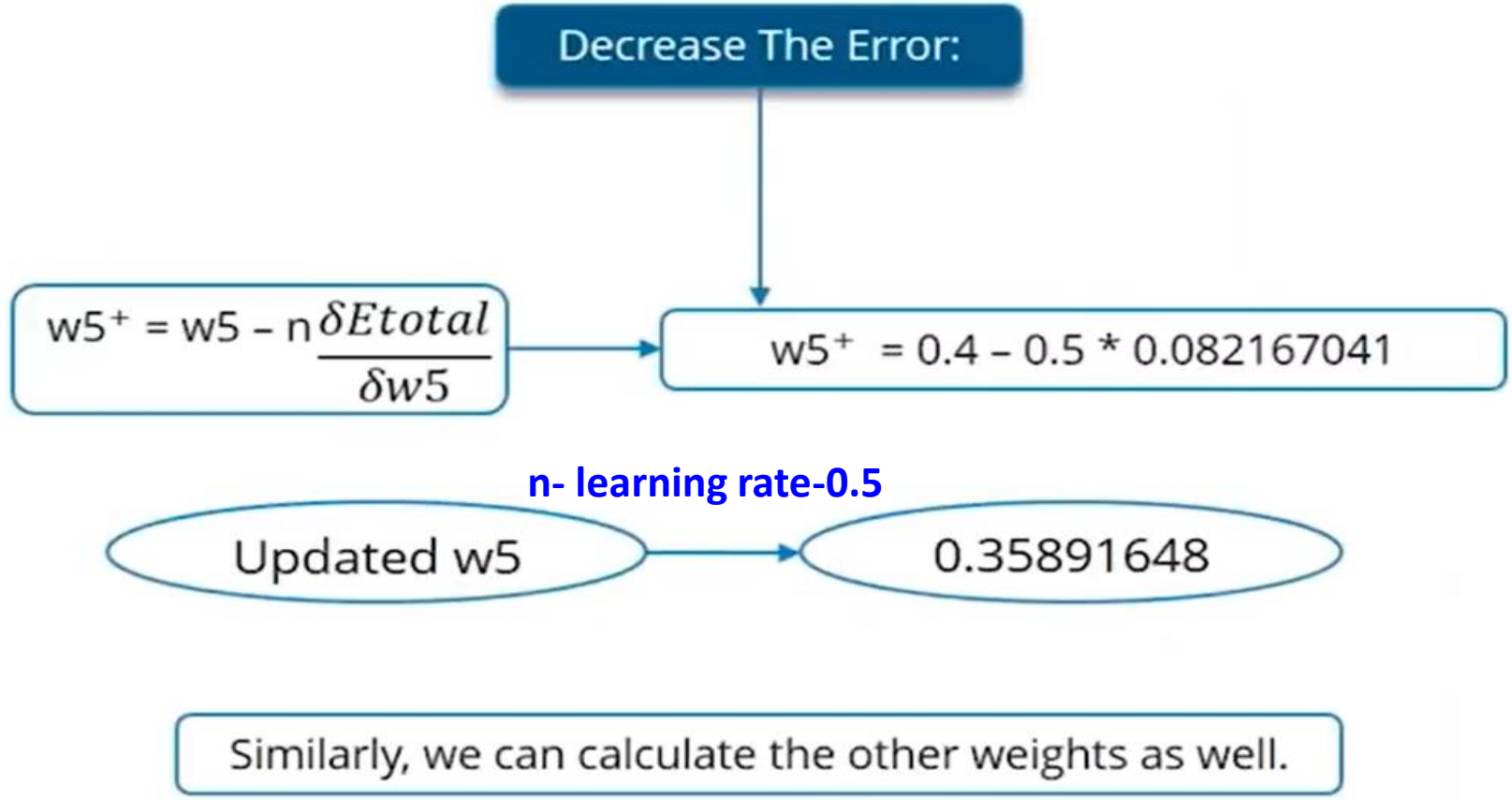
Find out the partial derivative of net o1 w.r.t. w5

Putting all these values together

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$

0.082167041

Backpropagation Learning Algorithm



Learning Rate, is a hyperparameter that determines how quickly a model learns from the data during training.

It controls the size of the step taken in each iteration of the optimization algorithm, **affecting the convergence and accuracy of the model.**

Artificial Neural Networks Architecture

- It is possible to think of the hidden layer as a **"distillation layer,"** which extracts some of the most relevant patterns from the inputs and sends them on to the next layer for further analysis.

- It accelerates and improves the efficiency of the network by recognizing just the most important information from the inputs and discarding the redundant information.

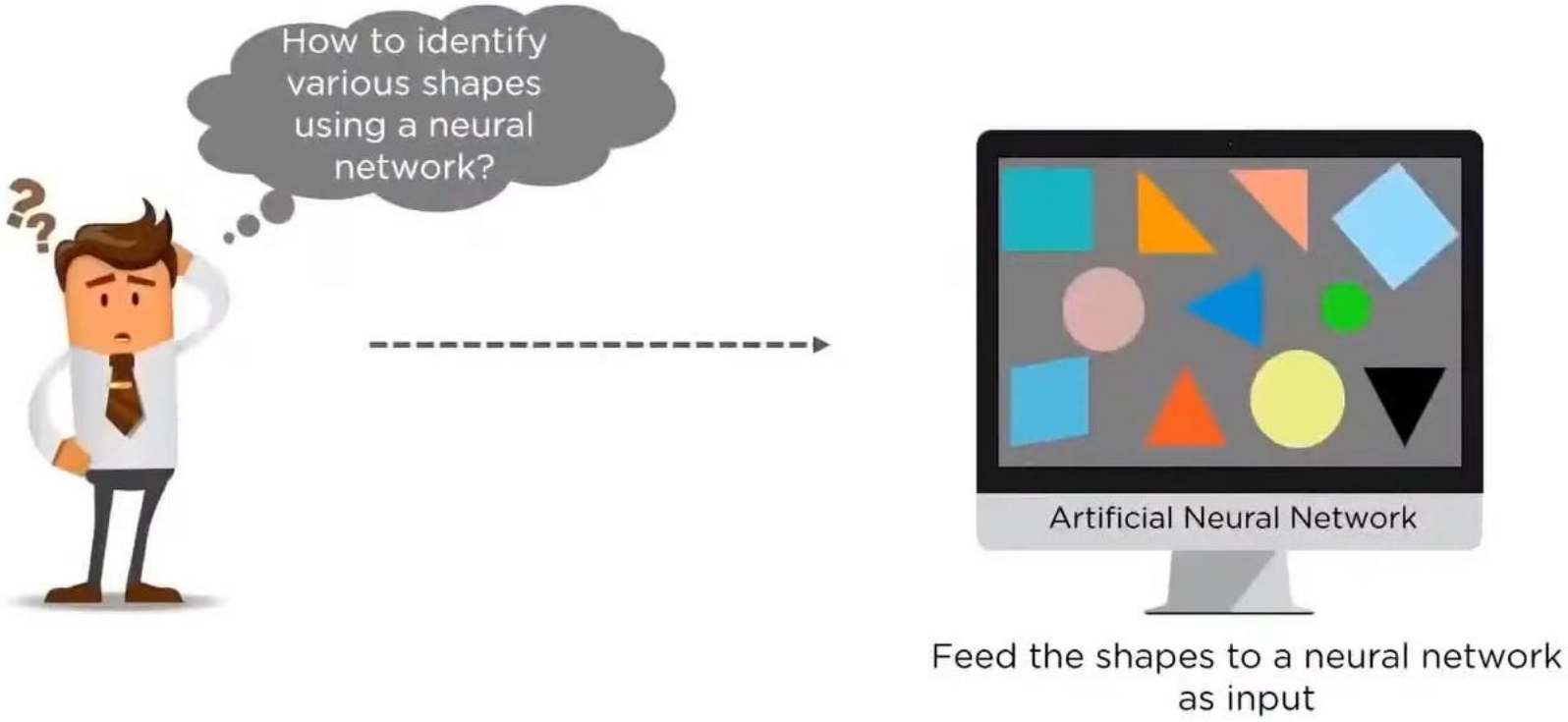
- The activation function is important for two reasons: **first, it allows you to fire the Neuron.**

- **Second this model captures the presence of non-linear relationships** between the inputs.

- It contributes to the conversion of the input into a more usable output.

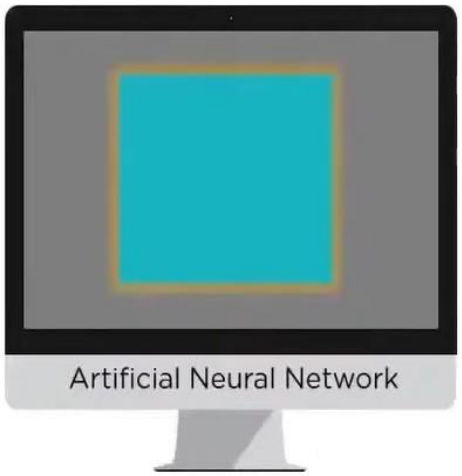
Working of Artificial Neural Network

- Let us find out how Artificial Neural Network can be used to identify different shapes like Square, Circles, and Triangles.

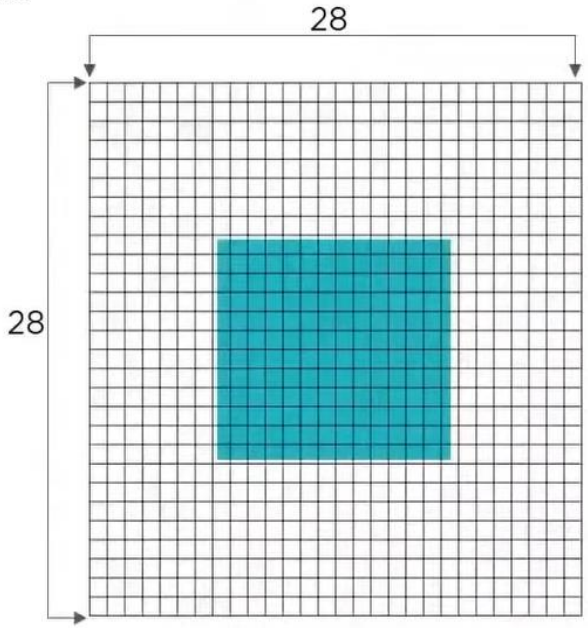


Working of Artificial Neural Network

Lets consider the shape of a square



Shape of a Square

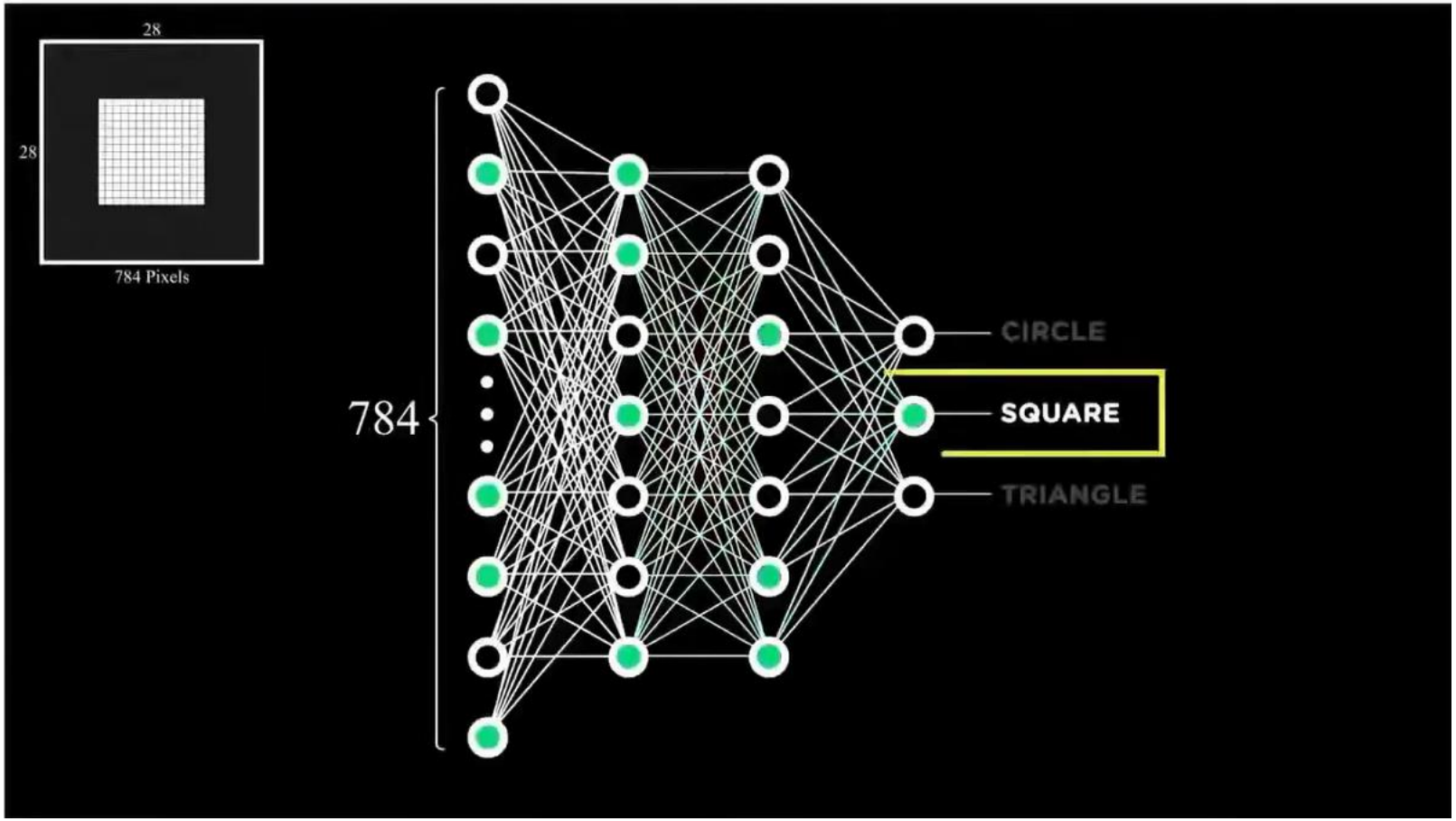


28*28 = 784

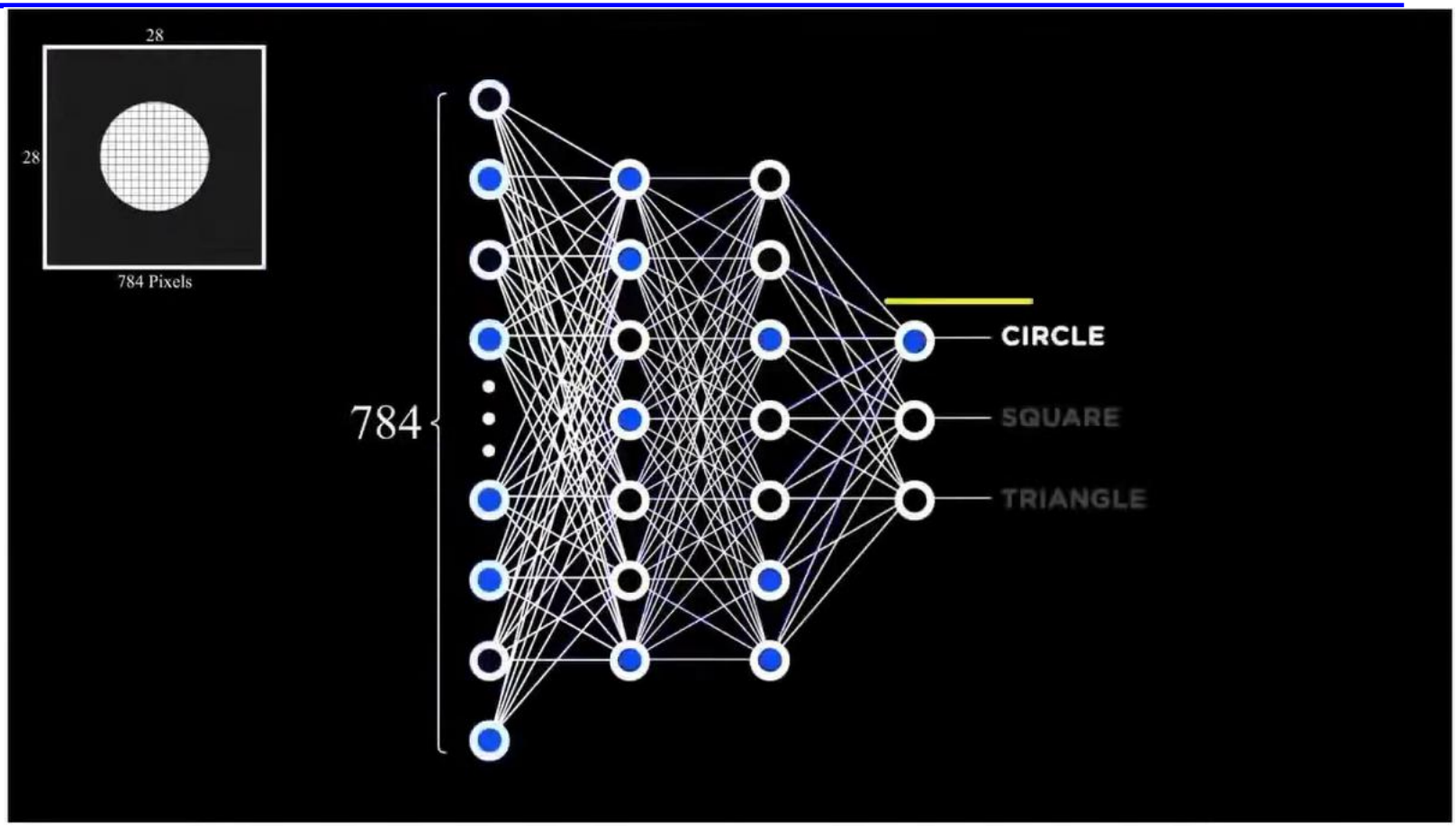
Strengths

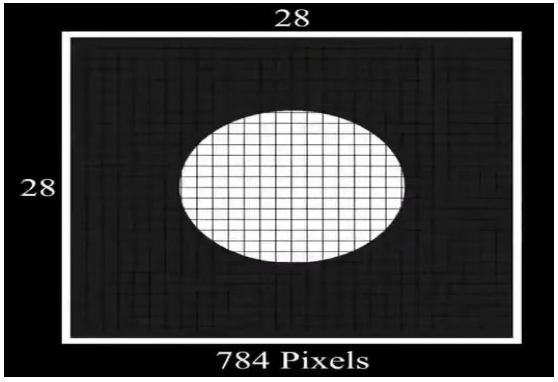
- 28*28 pixels of the input image is taken as input i.e. 784 neurons

Working of Artificial Neural Network

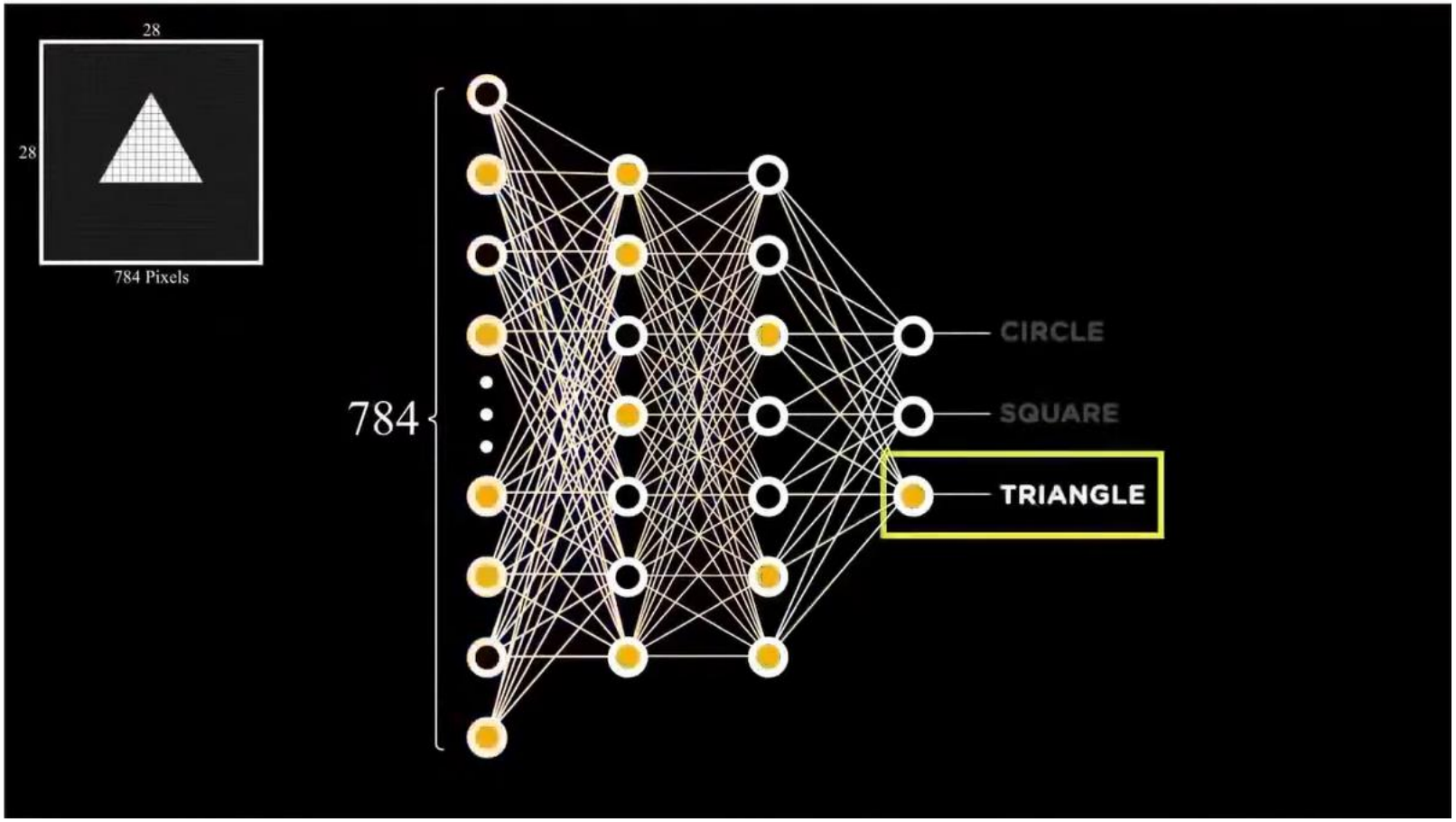


Working of Artificial Neural Network



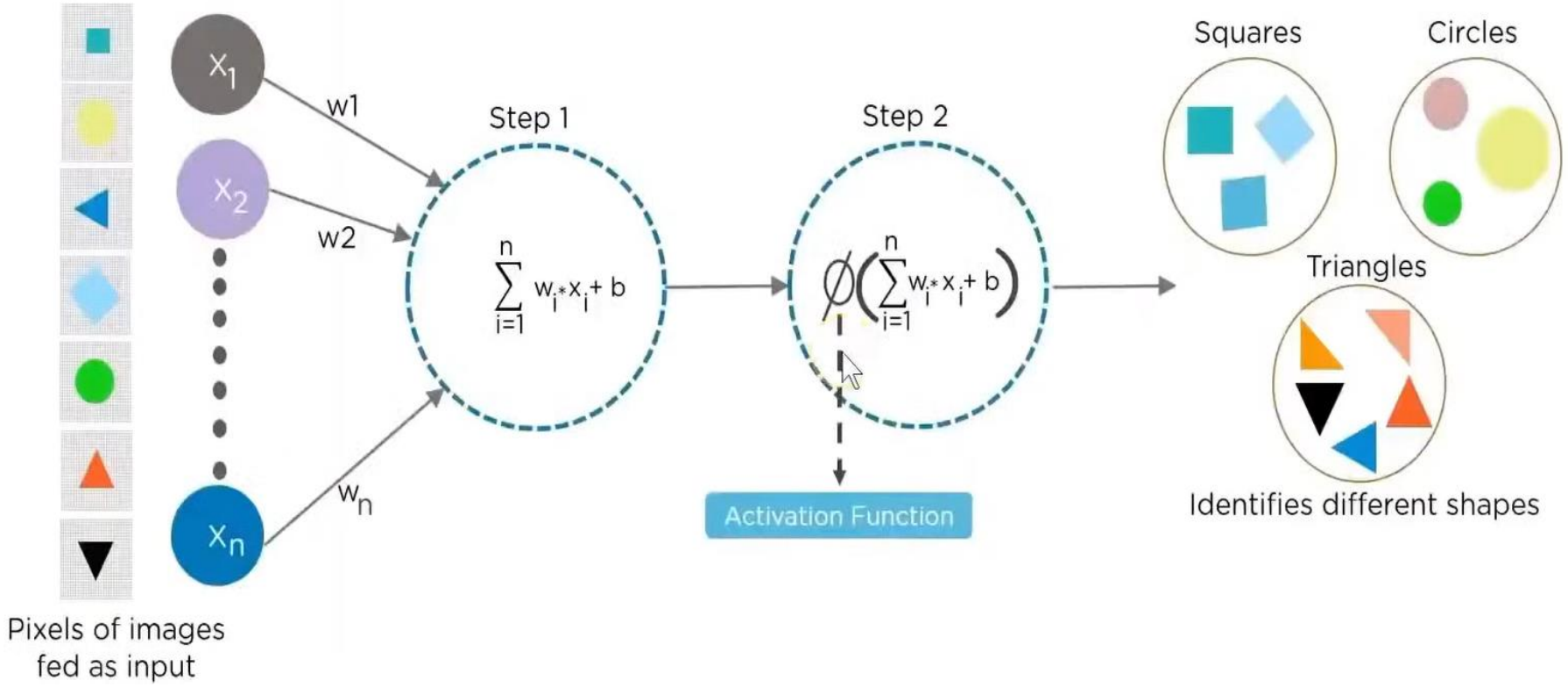


Working of Artificial Neural Network



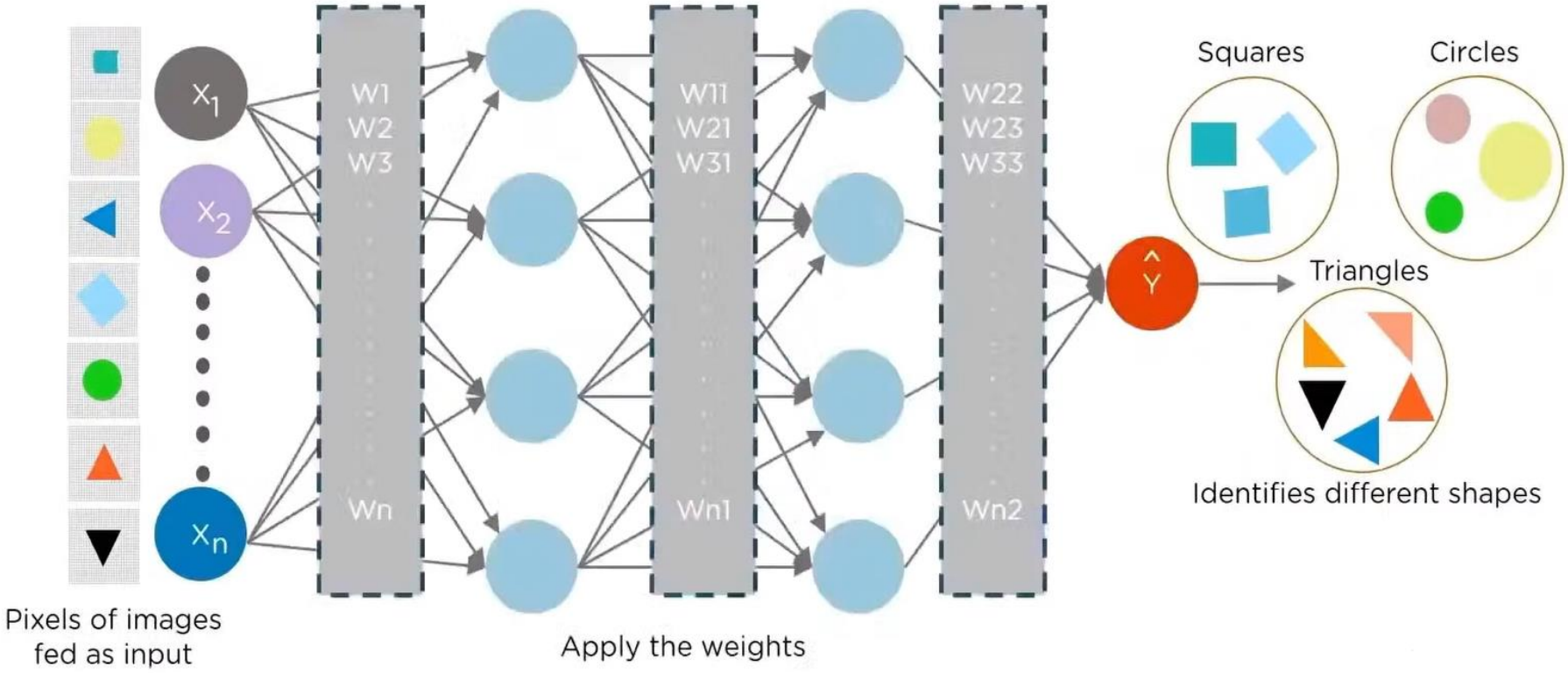
Working of Artificial Neural Network

Lets find out how an Artificial Neural Network can be used to identify different shapes

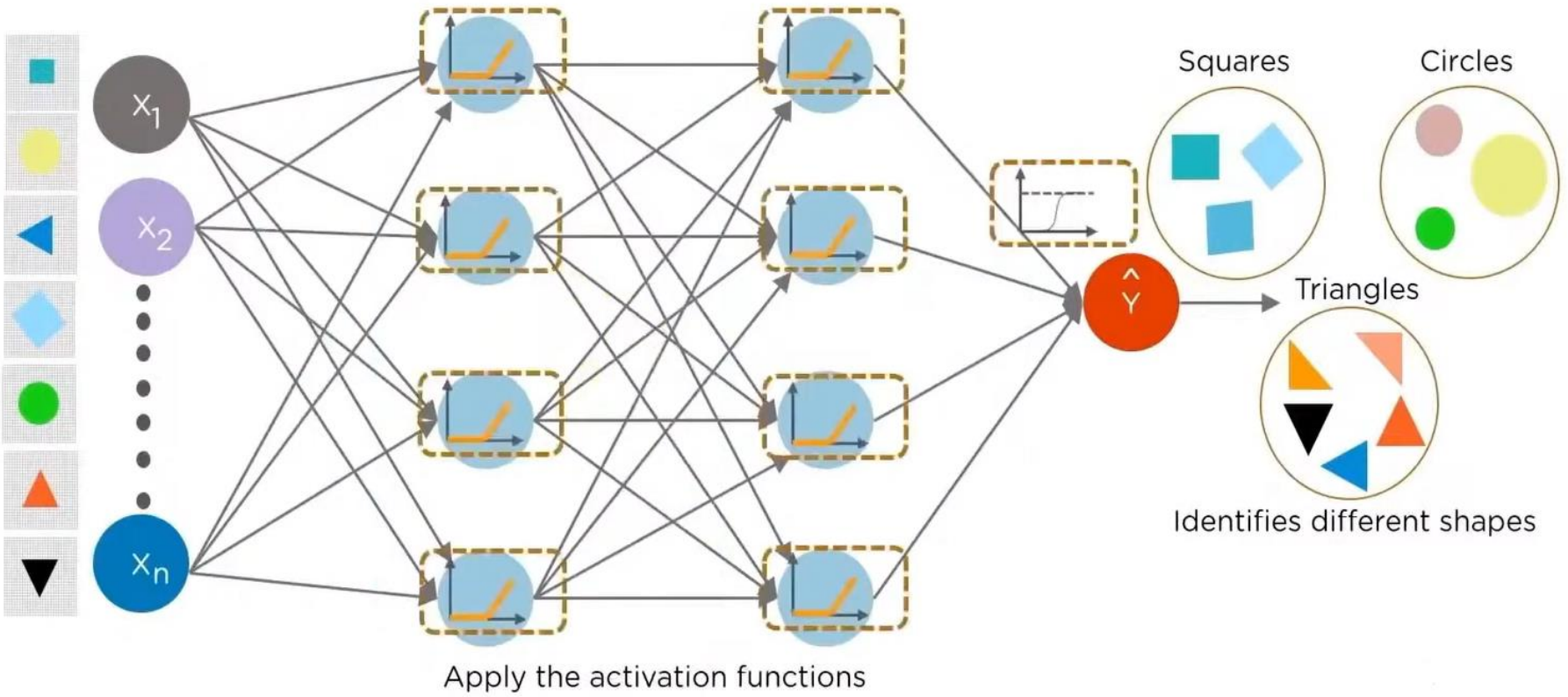


Working of Artificial Neural Network

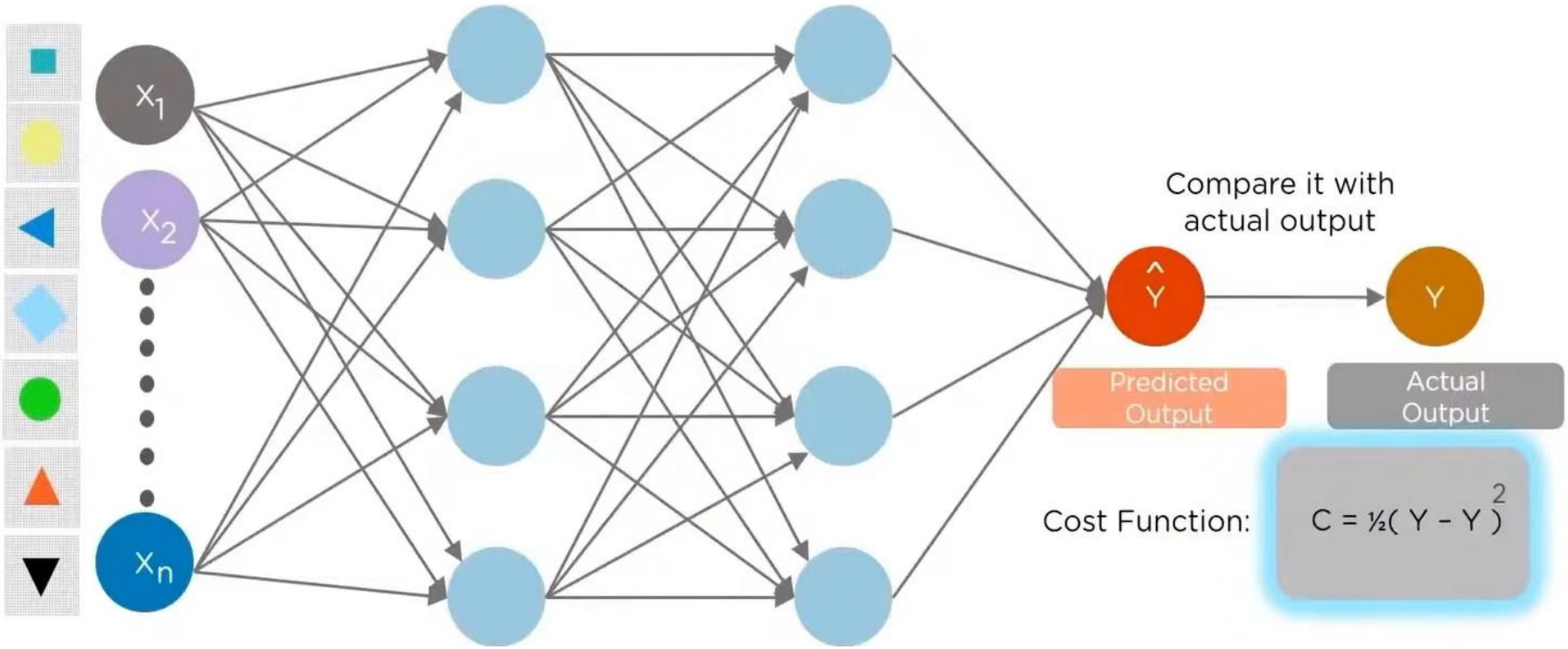
Lets find out how an Artificial Neural Network can be used to identify different shapes



Working of Artificial Neural Network



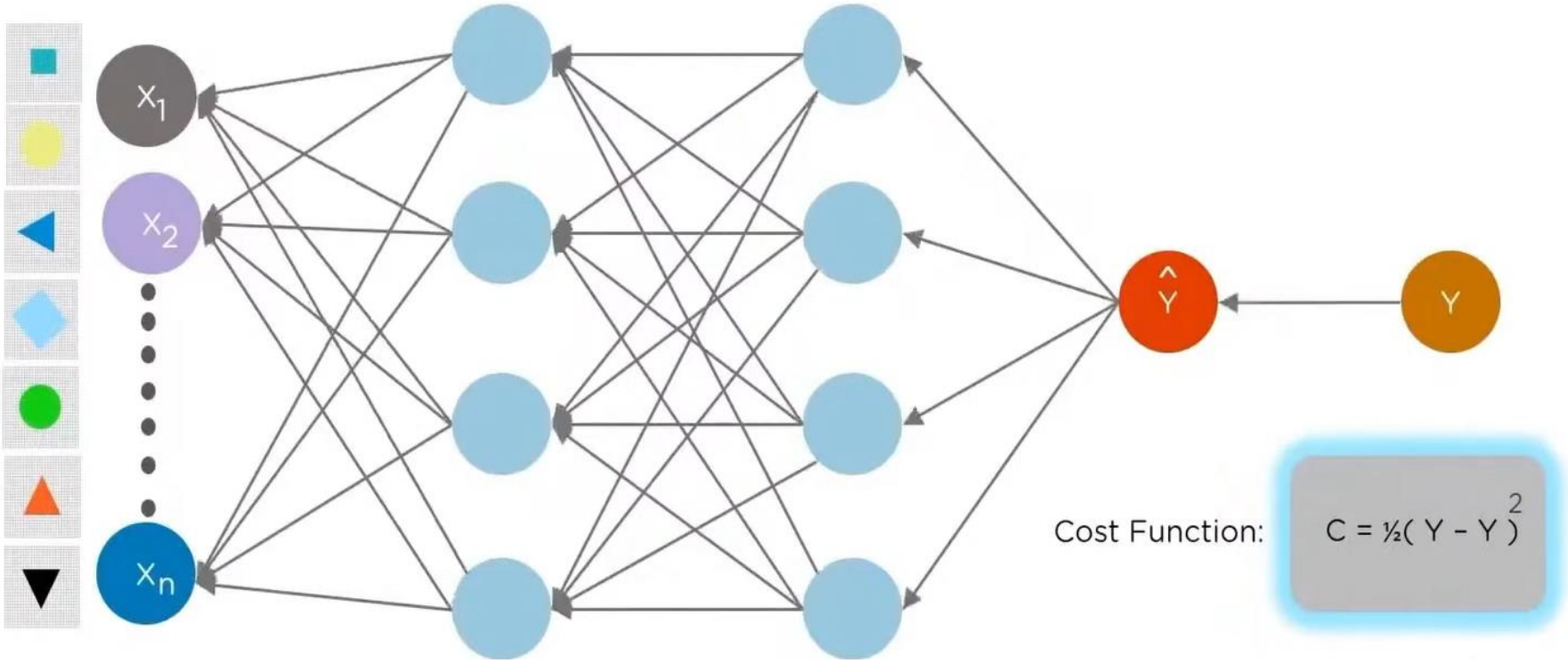
Working of Artificial Neural Network



Working of Artificial Neural Network

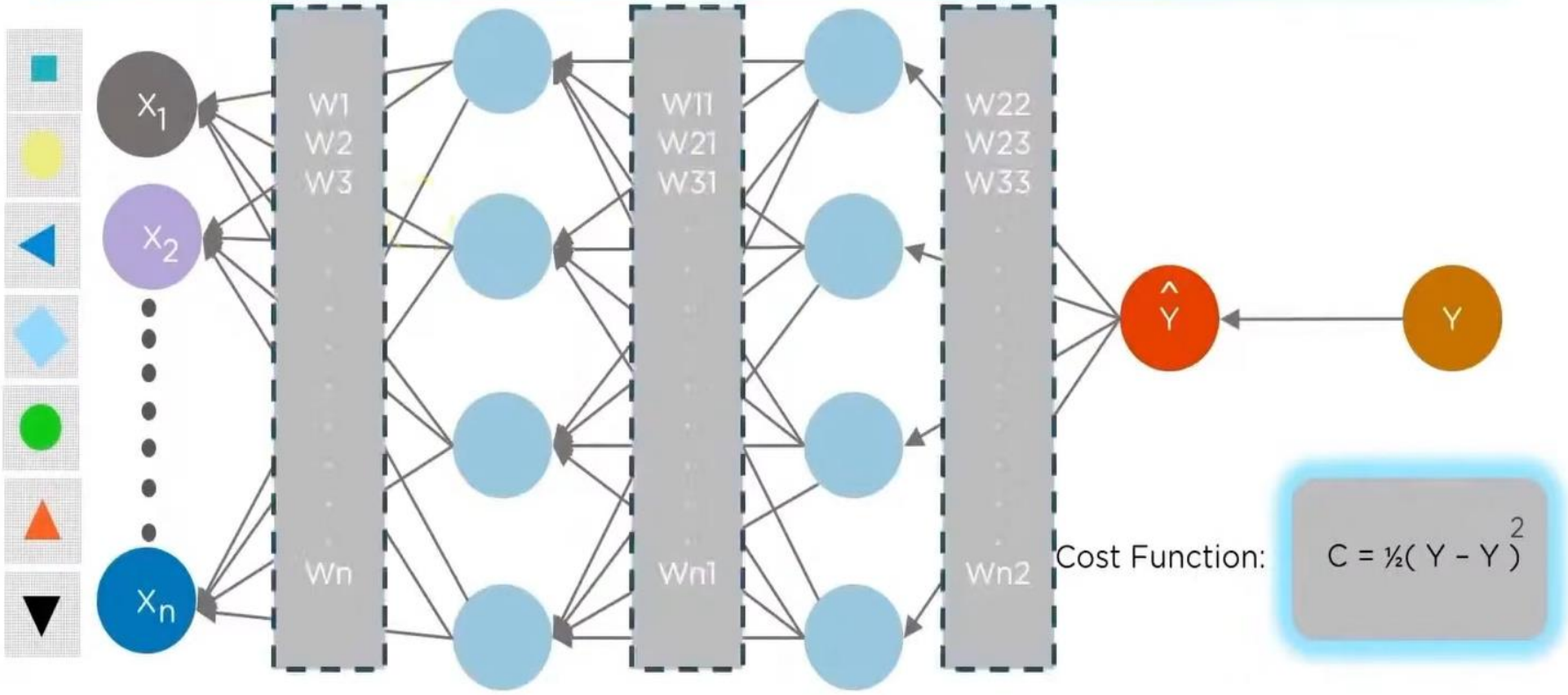
Neural Network use back propagation method along with the performance of the Neural Net.

A Cost function is used to reduce the error rate between predicted and actual output.

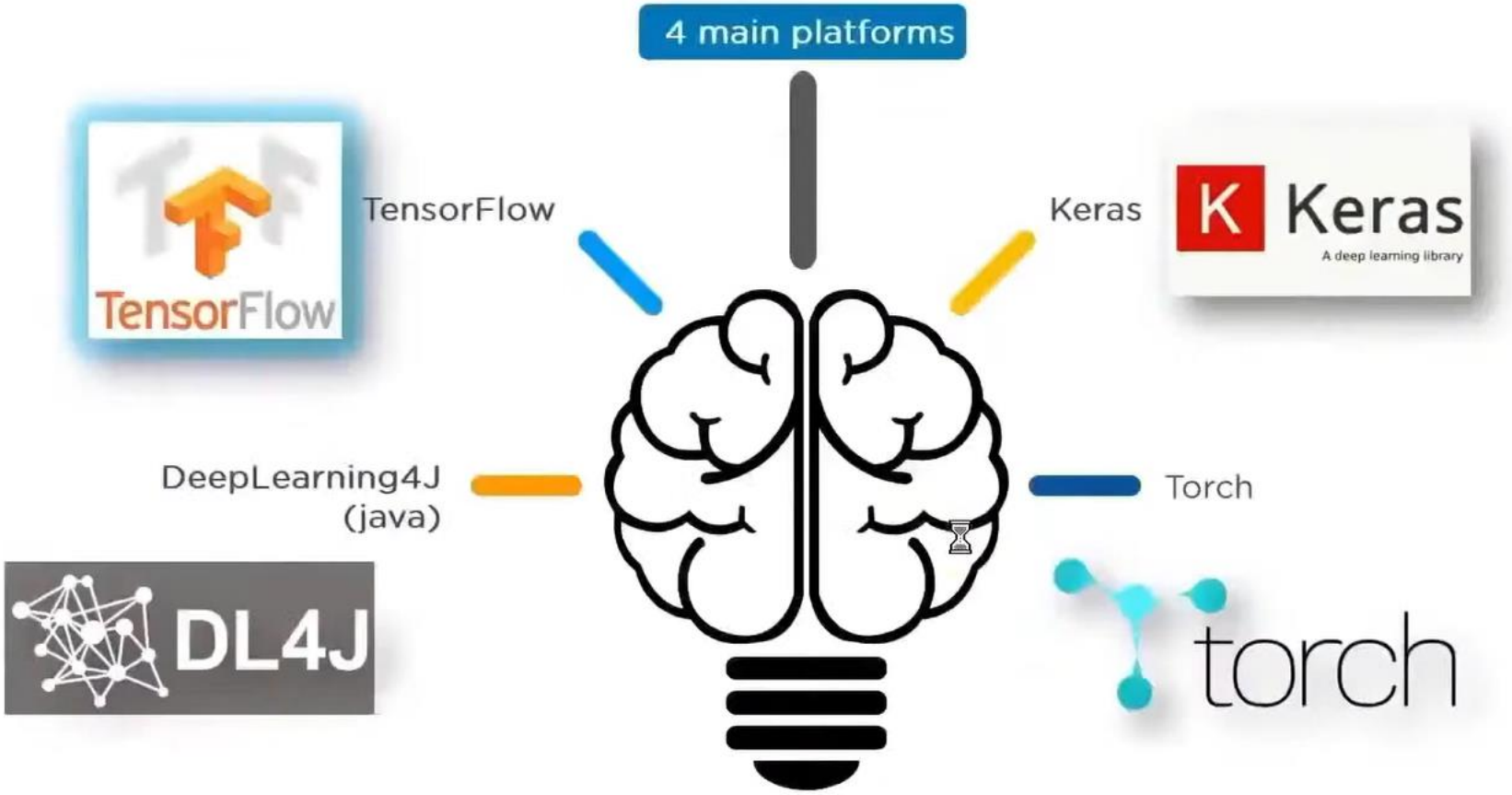


Working of Artificial Neural Network

The *Cost* value is the difference between the neural nets predicted output and the actual output from a set of labelled training data. The least cost value is obtained by making adjustments to the weights and biases iteratively throughout the training process.



Deep Learning Platforms

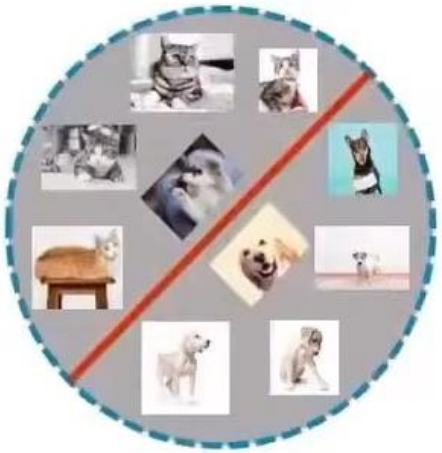


Use Case-Problem Statement

Use Case-Problem Statement

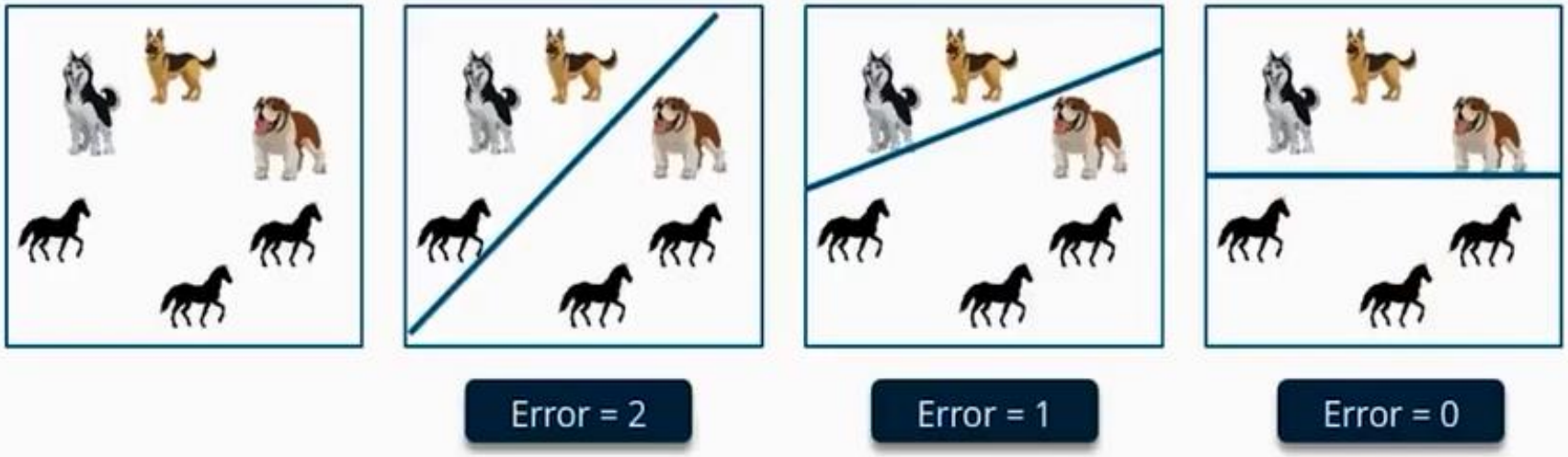
PROBLEM STATEMENT

CLASSIFY PHOTOS OF CATS AND DOGS USING NEURAL NETWORK



Perceptron Learning Algorithm

It is used to classify any linearly separable set of inputs.



Benefits of Artificial Neural Networks

- ANNs offers many key benefits that make them particularly well-suited to specific issues and situations:

1. **ANNs can learn and model non-linear and complicated interactions**, which is critical since many of the relationships between inputs and outputs in real life are non-linear and complex.

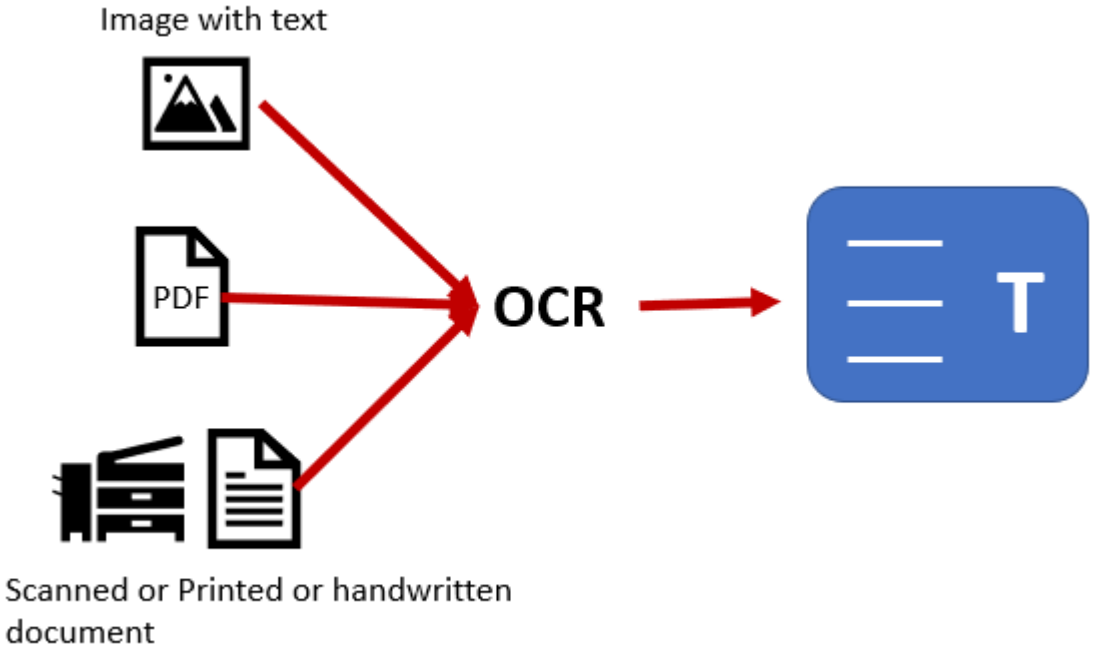
2. ANNs can generalize – After learning from the original inputs and their associations, **the model may infer unknown relationships from anonymous data, allowing it to generalize and predict unknown data.**

3. **ANN does not impose any constraints on the input variables**, unlike many other prediction approaches (like how they should be distributed).

Application of Artificial Neural Networks

1. Image Processing and Character recognition:

- **ANNs play a significant part in picture and character recognition** because of **their capacity to take in many inputs, process them, and infer hidden and complicated, non-linear correlations.**
- Character recognition, such as handwriting recognition, has many applications in fraud detection (for example, bank fraud) and even national security assessments.



Application of Artificial Neural Networks

2. Forecasting

Forecasting is widely used in everyday company decisions (sales, the financial allocation between goods, and capacity utilization), economic and monetary policy, finance, and the stock market.

Forecasting issues are frequently complex; For example, predicting stock prices is complicated with many underlying variables (some known, some unseen).

Disadvantages of Artificial Neural Networks

1. Hardware Dependence:

2. Understanding the network's operation:

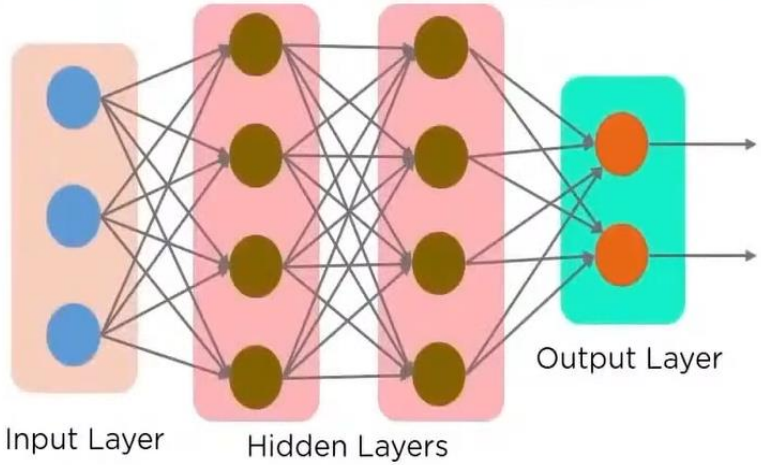
3. Assured network structure:

4. Difficulty in presenting the issue to the network:

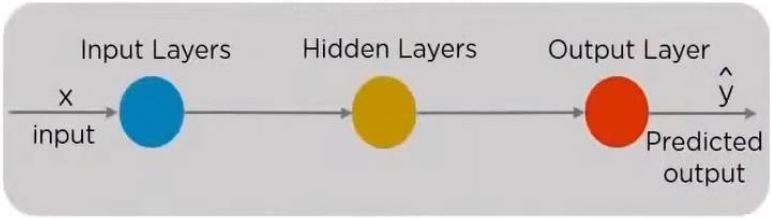
5. The network's lifetime is unknown:

Feed Forward Neural Networks

In a Feed-Forward Network , information flows only in forward direction, from the input nodes, through the hidden layers (if any) and to the output nodes. There are no cycles or loops in the network.



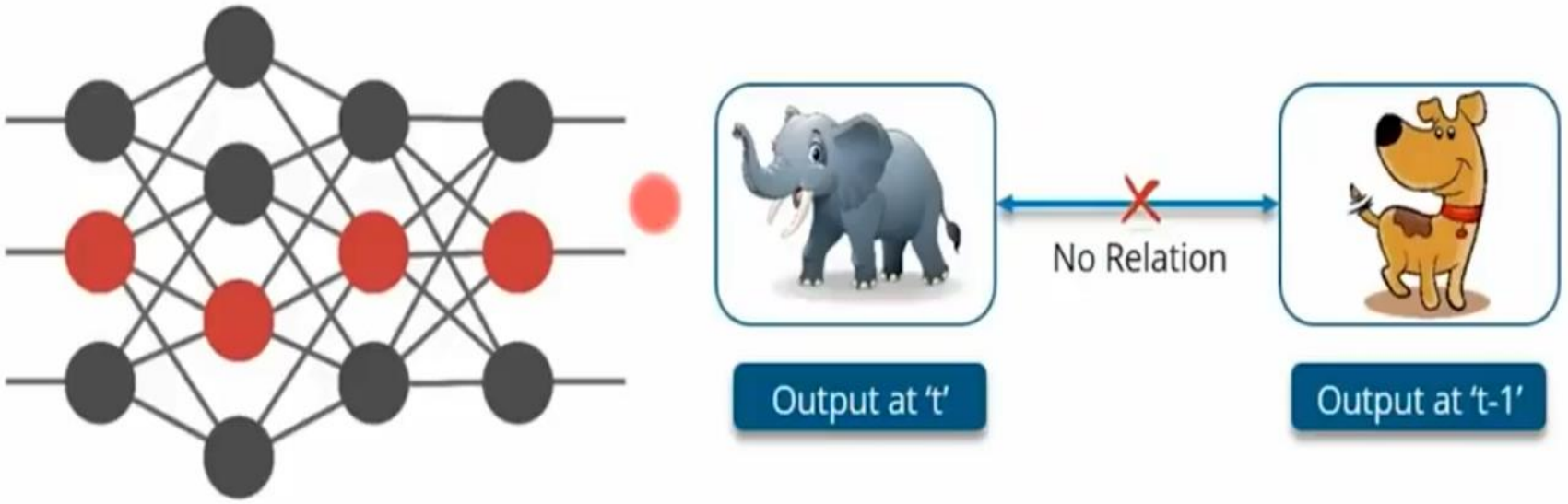
Simplified presentation



- Decisions are based on current input
- No memory about the past
- No future scope

Feedforward Neural Network Limitation

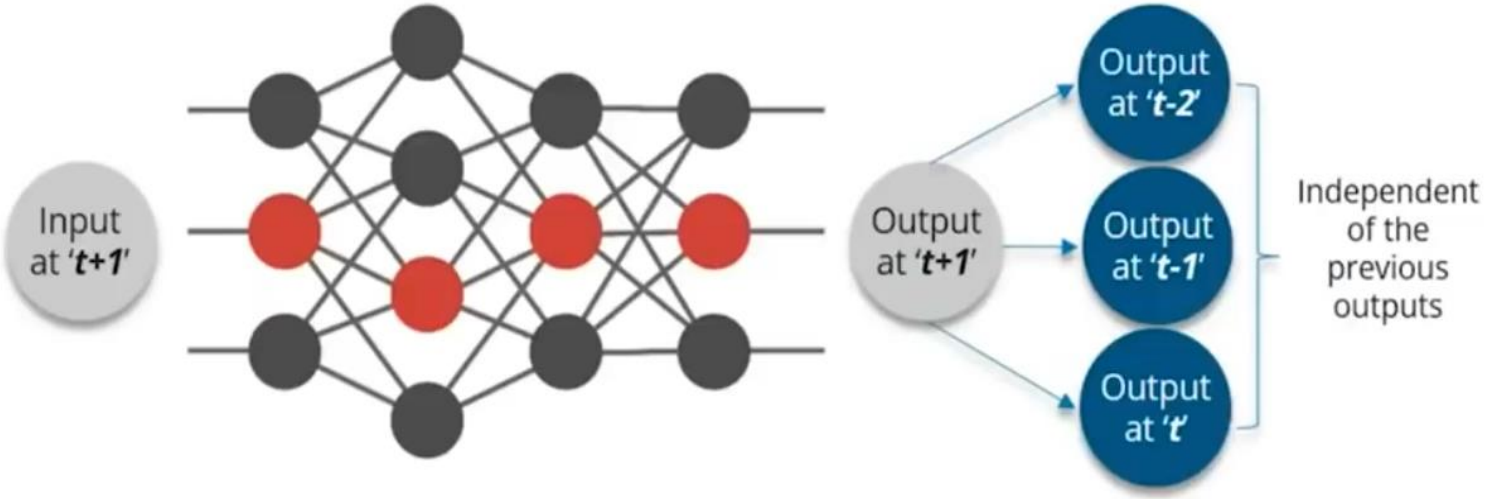
A trained feedforward network can be exposed to any random collection of photographs, and the first photograph it is exposed to will not necessarily alter how it classifies the second



Seeing photograph of a dog will not lead the net to perceive an elephant next

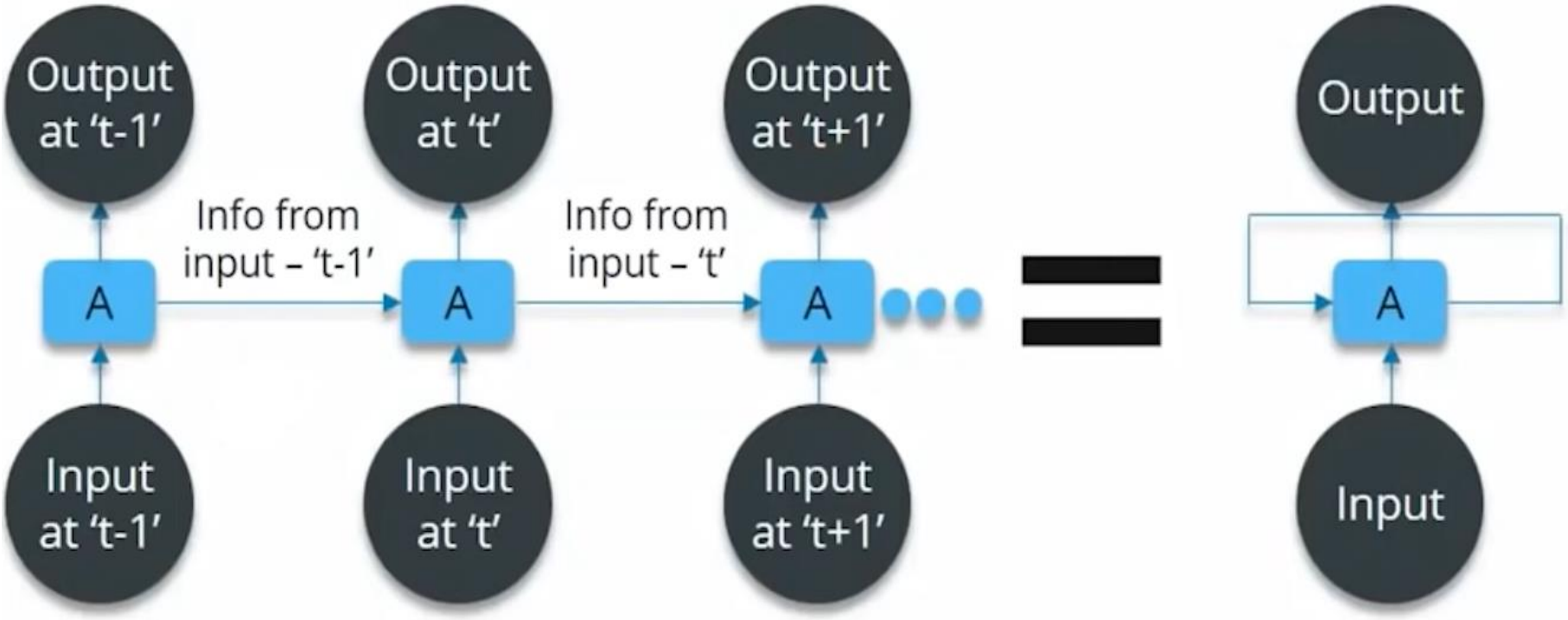
Feedforward Neural Network Limitation

When you read a book, you understand it based on your understanding of previous words



I cannot predict the next word in a sentence if I use feedforward nets

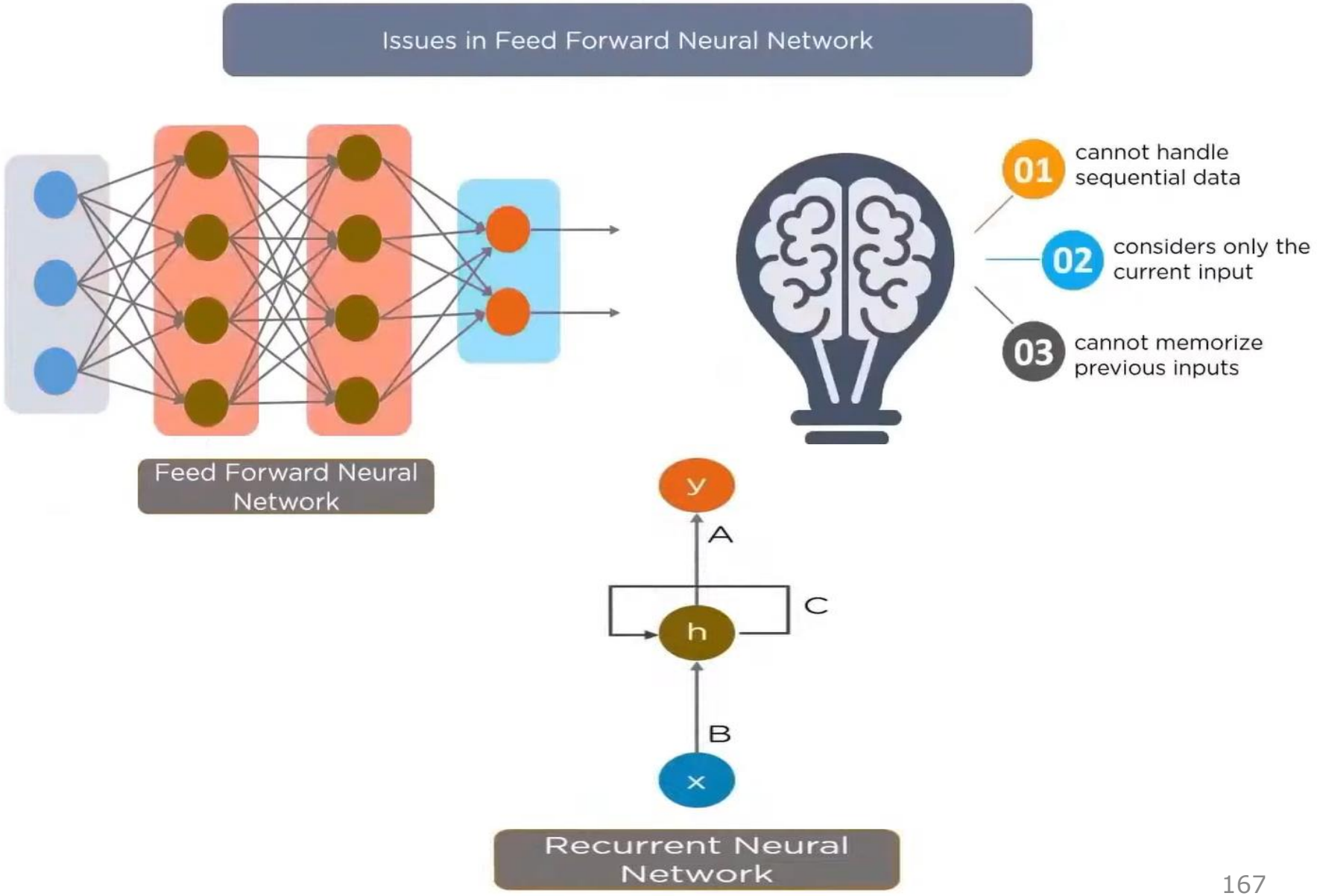
How to Overcome This Challenge



Why Recurrent?

- The **central loophole in neural networks** is that **it does not have memory.**
- Since no memory is associated, it becomes **very difficult to work on sequential data like text corpora** where we have sentences associated with each other, and even time-series where data is entirely sequential and dynamic.
- Here, **Recurrent Neural Networks** comes to play.
- **RNN addresses the memory issue by giving a feedback mechanism** that looks back to the previous output and serves as a kind of memory.
- Since **the previous outputs gained during training leaves a footprint,** it is very easy for the model to predict the future tokens (outputs) with the help of previous ones.

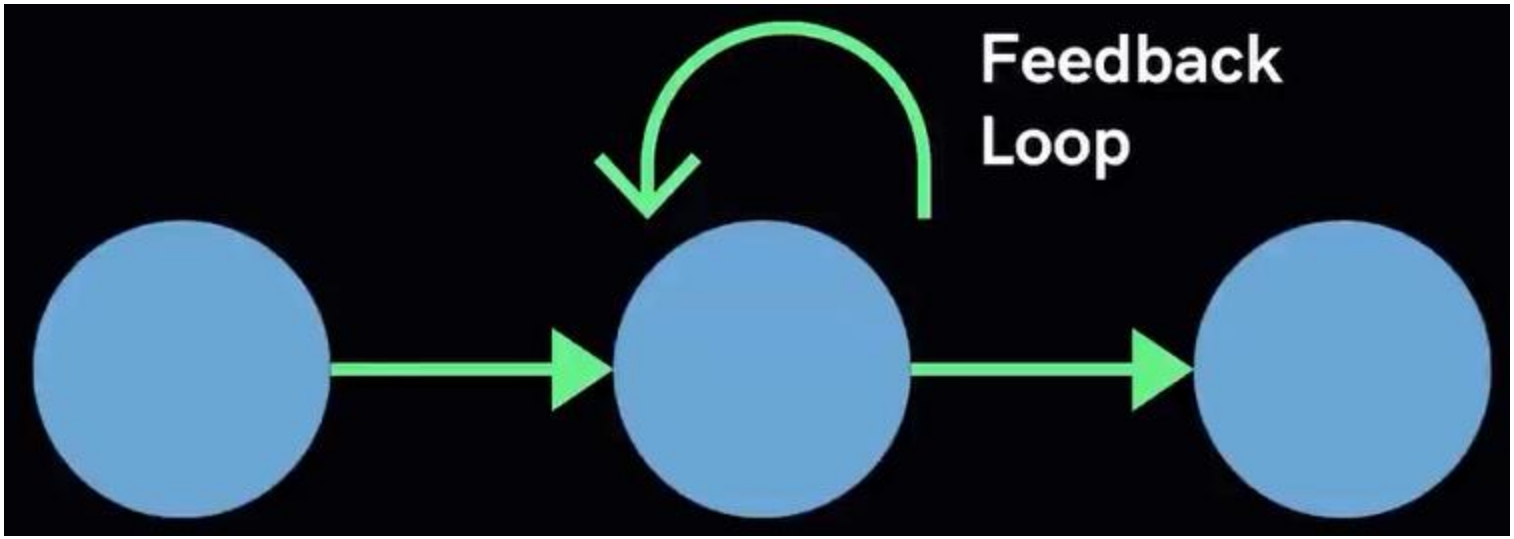
Why Recurrent Neural Networks



Recurrent Neural Networks

Recurrent Neural Networks

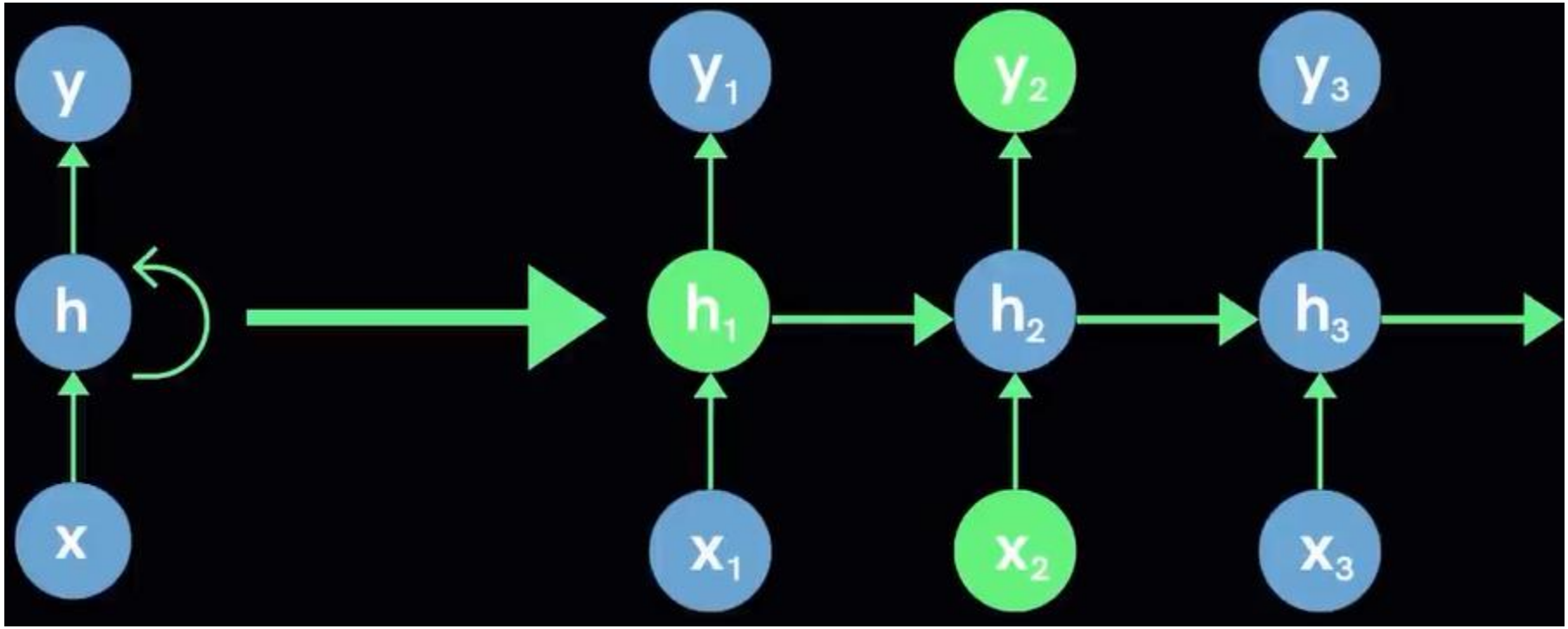
Uses a feedback loop in the hidden layers



Recurrent Neural Networks

Recurrent Neural Networks

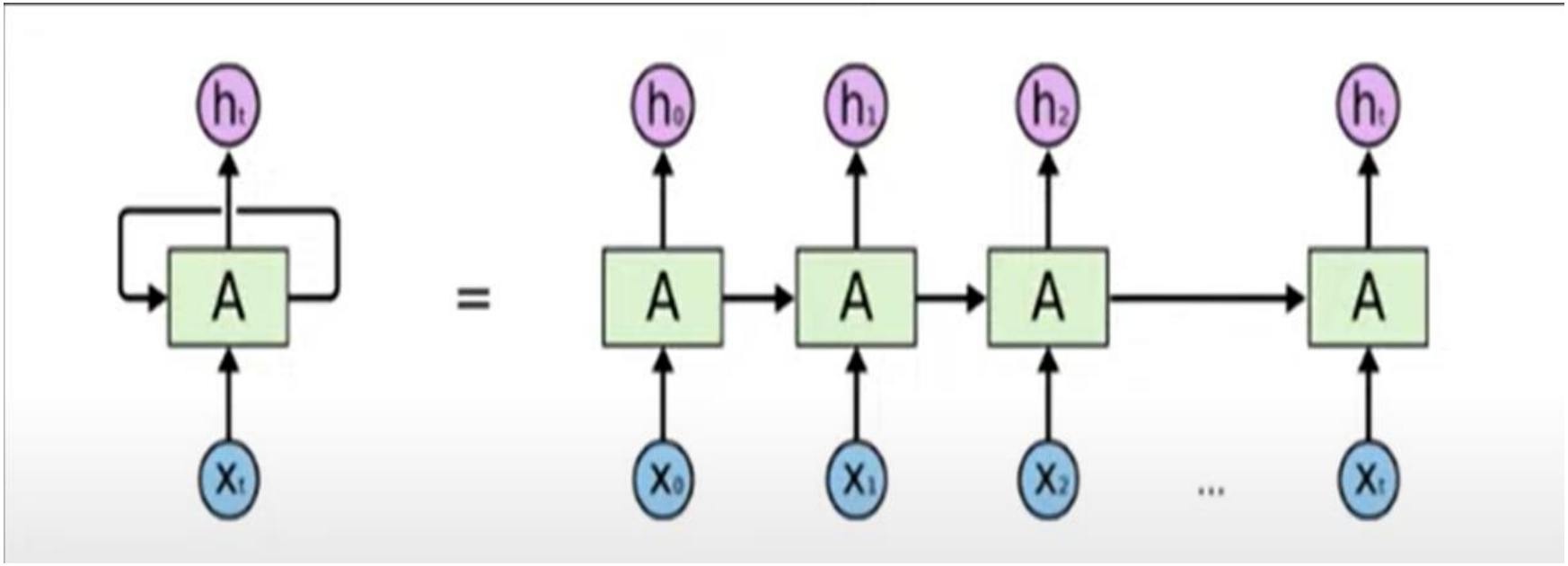
Uses a feedback loop in the hidden layers



Recurrent Neural Networks

A Recurrent Neural Network is a type of Neural Network commonly used in speech recognition and natural language processing.

RNNs are designed **to recognize a data's sequential characteristics** and **use pattern to predict the next likely scenario.**



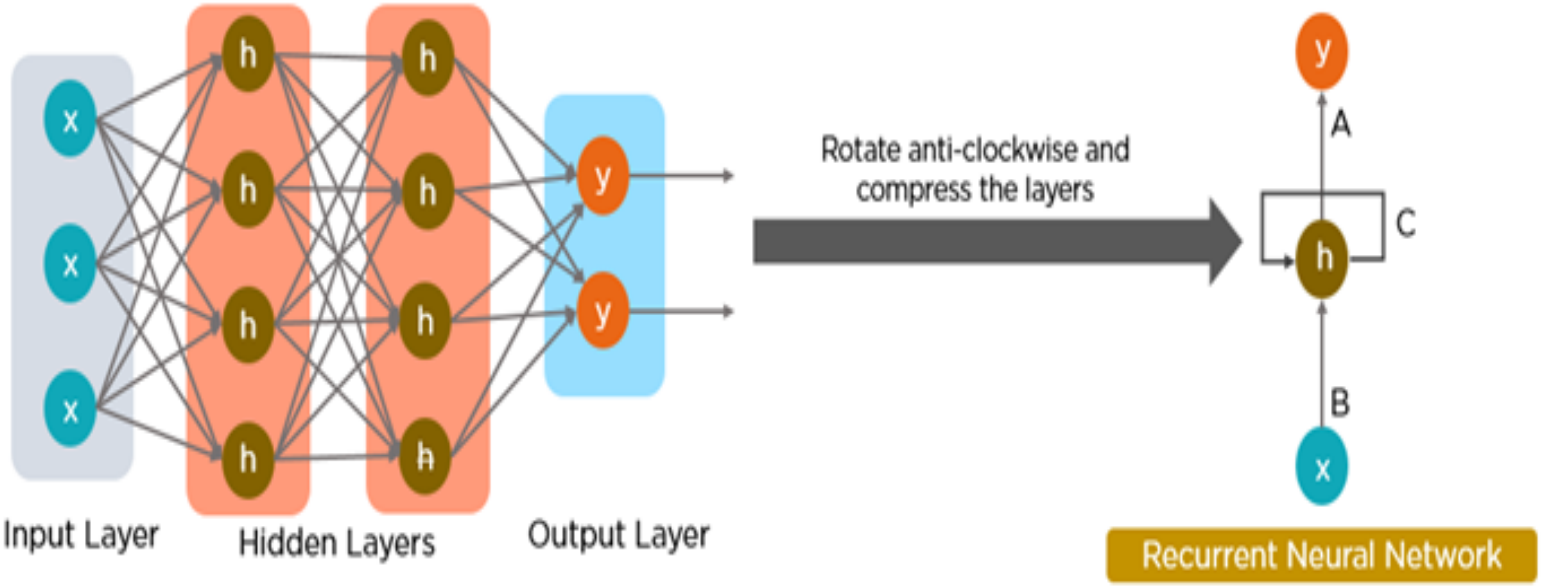
What is Recurrent Neural Network?

- Recurrent Networks are a type of artificial neural network designed **to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data coming from sensors, stock markets and government agencies.**

What is a Recurrent Neural Network (RNN)?

RNNs are a type of neural network that can be used to model sequence data. RNNs, which are formed from feedforward networks, are similar to human brains in their behaviour.

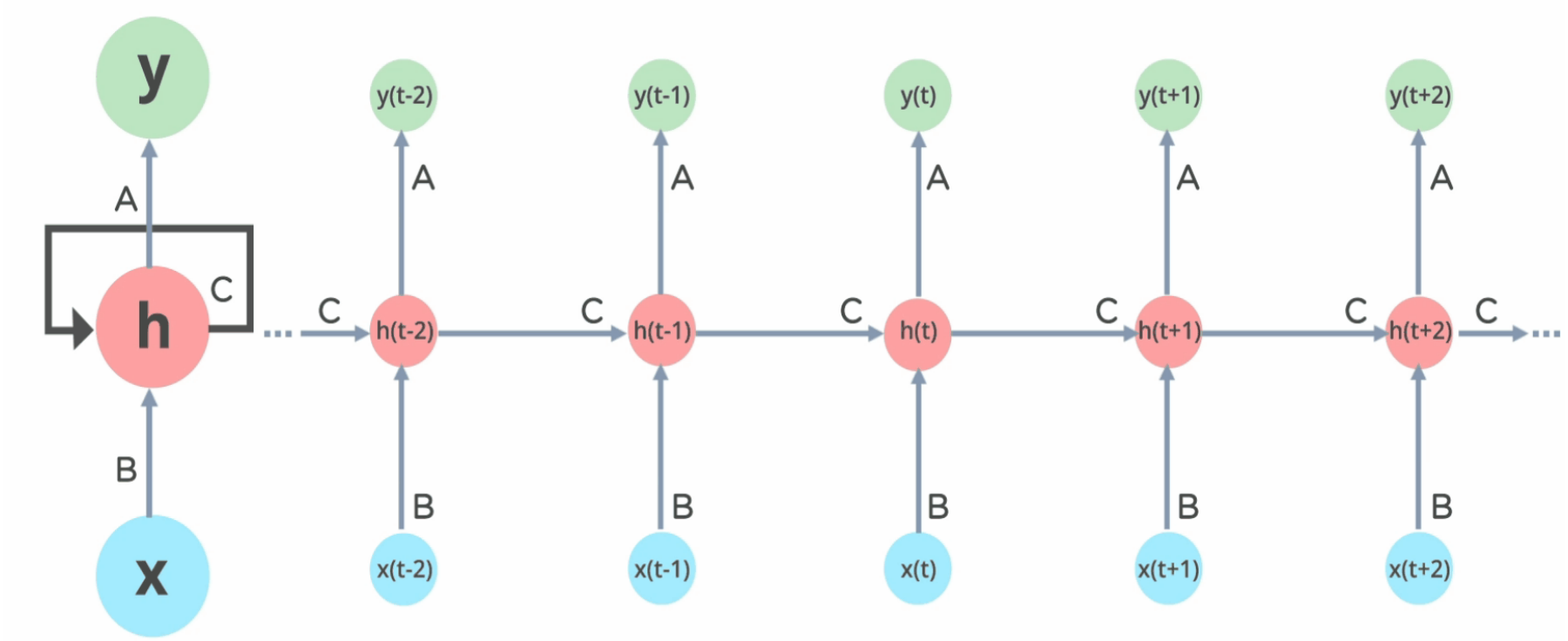
Simply said, recurrent neural networks can anticipate sequential data in a way that other algorithms can't.



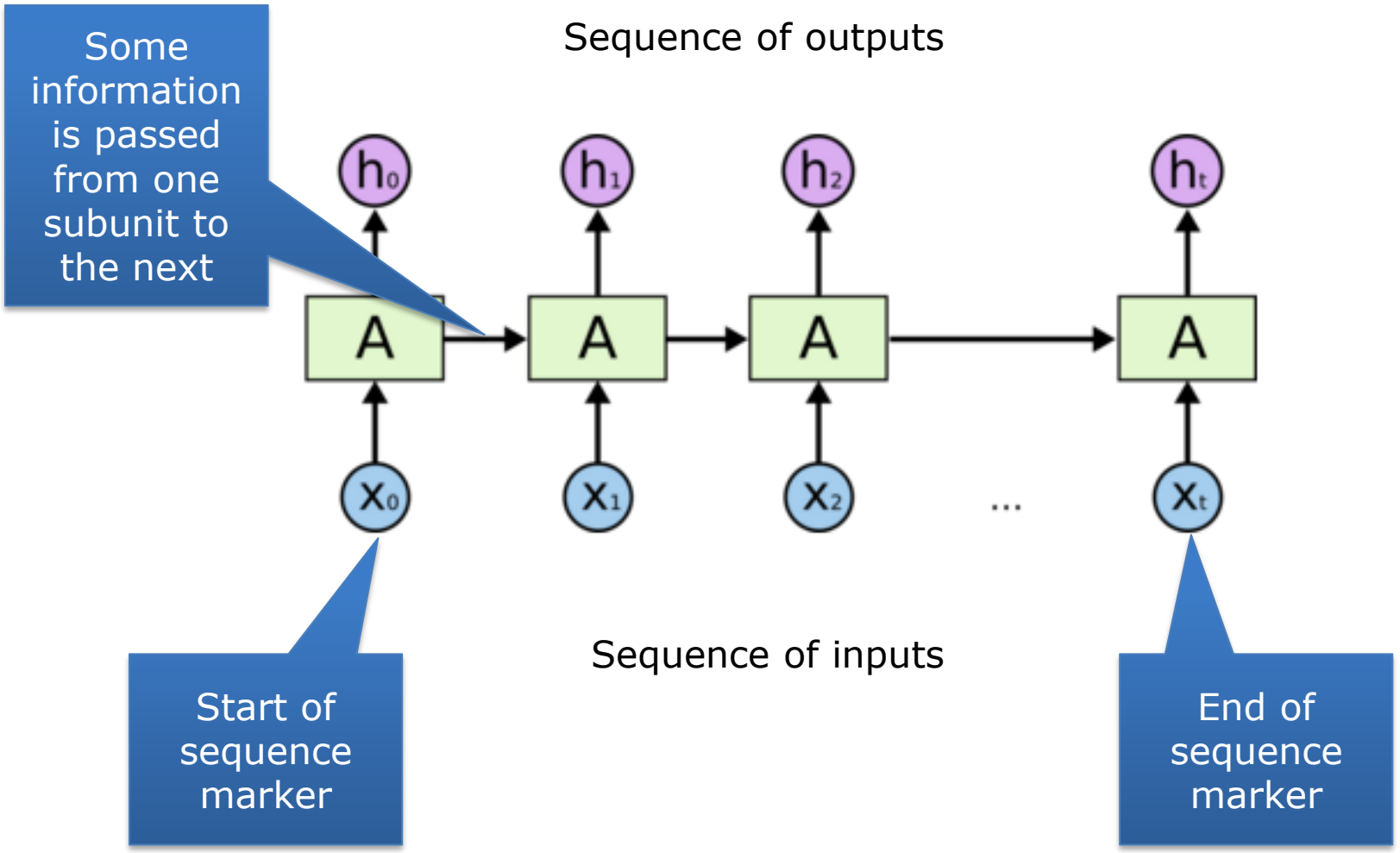
RNNs have a Memory that stores all information about the calculations. It employs the same settings for each input since it produces the same outcome by performing the same task on all inputs or hidden layers.

How does Recurrent Neural Networks work?

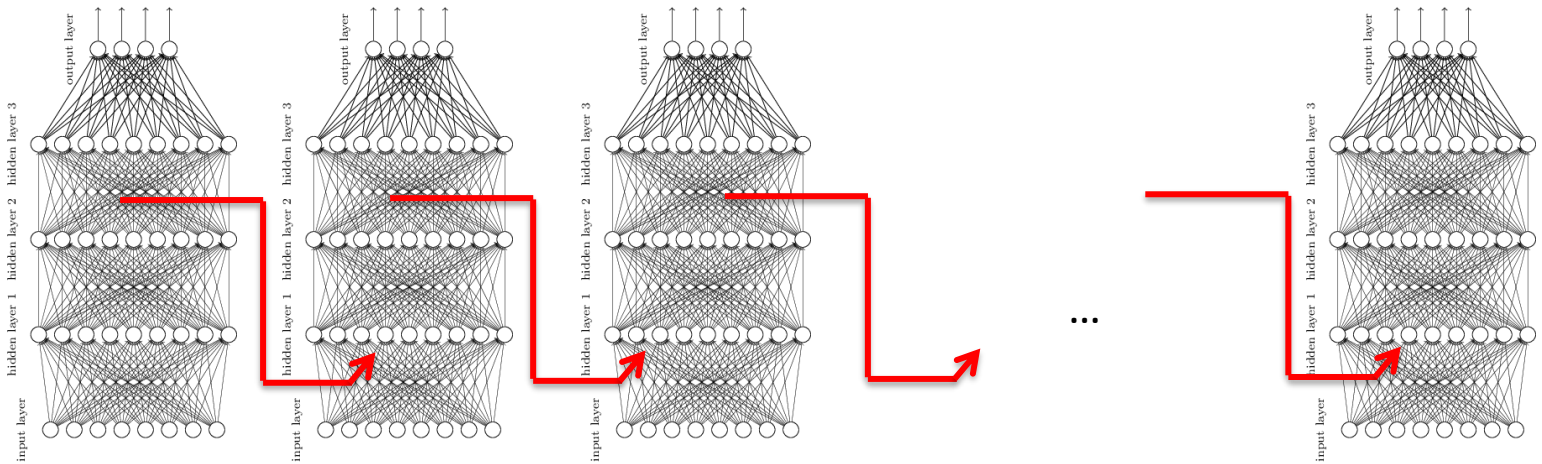
The information in recurrent neural networks cycles through a loop to the middle-hidden layer.



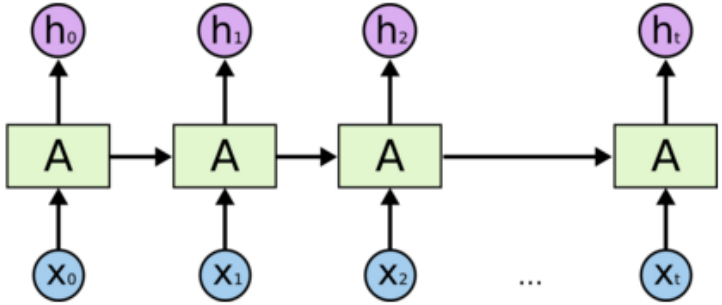
Architecture for an RNN



Architecture for an 1980's RNN

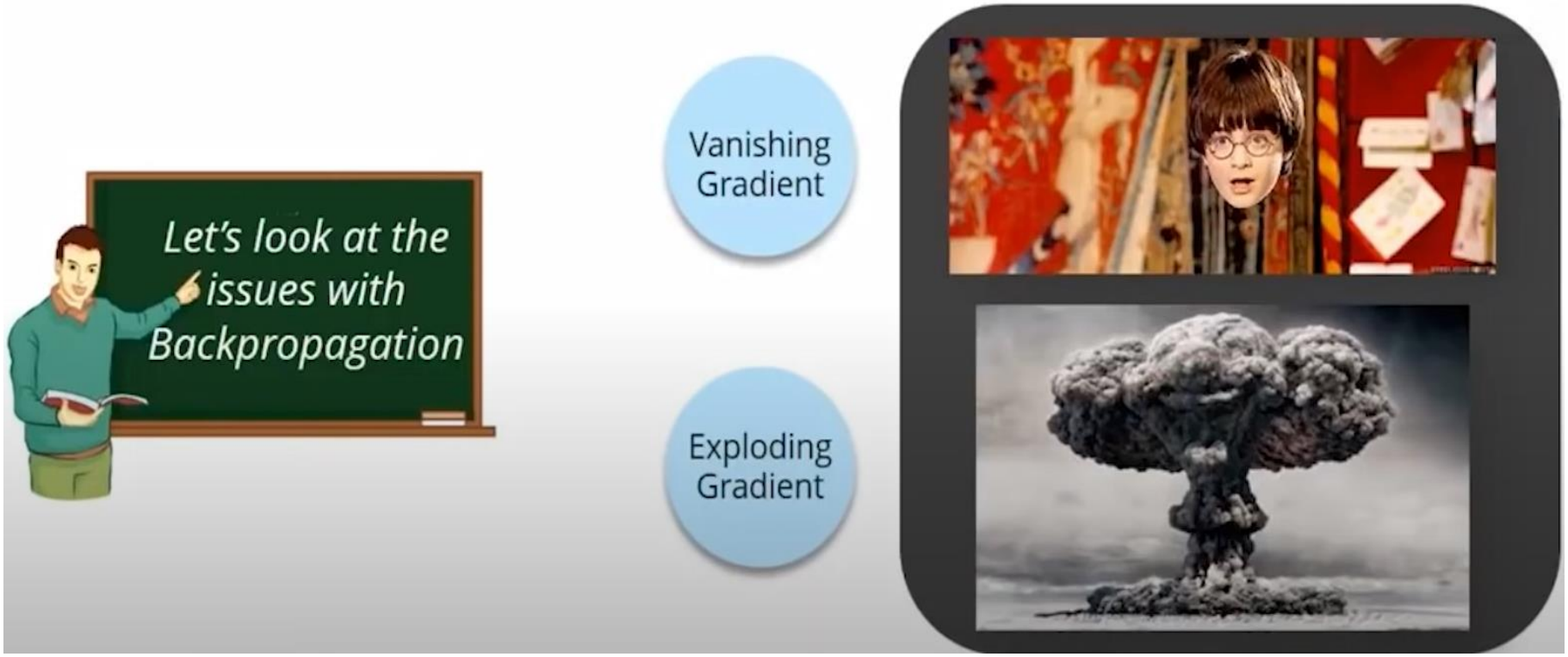


Problem with this: it's extremely deep and very hard to train



Training a Recurrent Neural Network

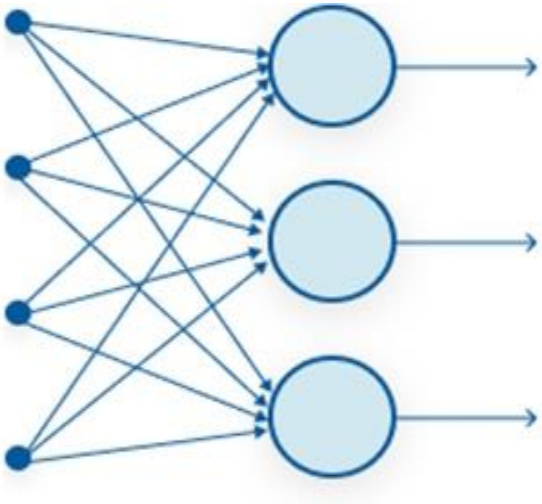
Recurrent Neural Nets uses Backpropagation Algorithm, but it is applied for every time stamp. It is commonly known as Backpropagation Through Time (BTT).



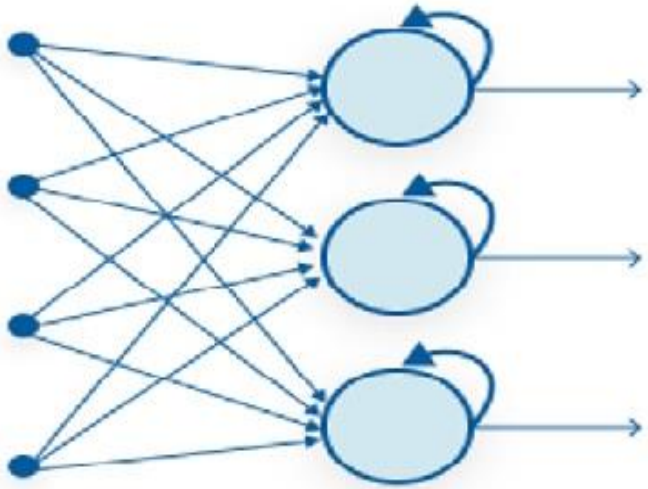
Recurrent Vs Feedforward Neural Network

A feed-forward neural network has only one route of information flow: from the input layer to the output layer, passing through the hidden layers. The data flows across the network in a straight route, never going through the same node twice.

The information flow between an RNN and a feed-forward neural network is depicted in the two figures below.



Feed-Forward Neural Network



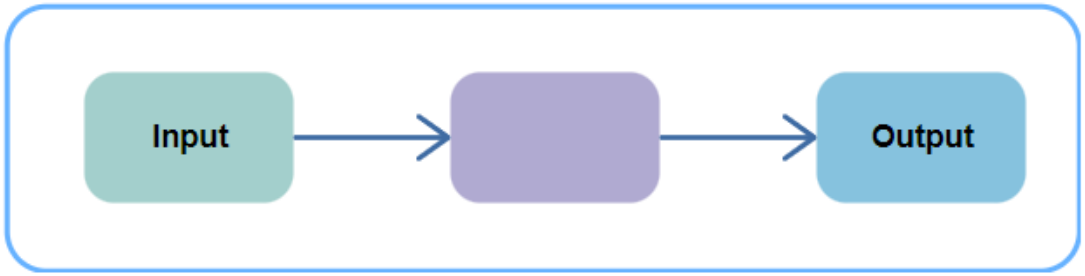
Recurrent Neural Network

Types of Recurrent Neural Network

One-to-One RNN

The simplest type of RNN is One-to-One, which allows a single input and a single output.

It has fixed input and output sizes and acts as a traditional neural network.



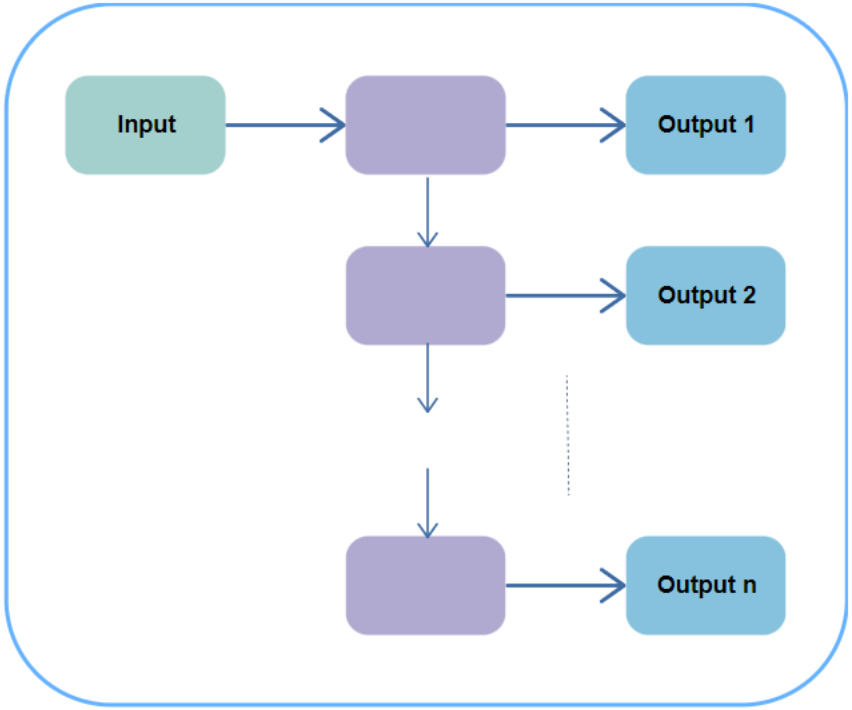
The One-to-One application can be found in Image Classification.

Types of Recurrent Neural Network

One-to-Many RNN

One-to-Many is a type of RNN that gives multiple outputs when given a single input.

It takes a fixed input size and gives a sequence of data outputs.



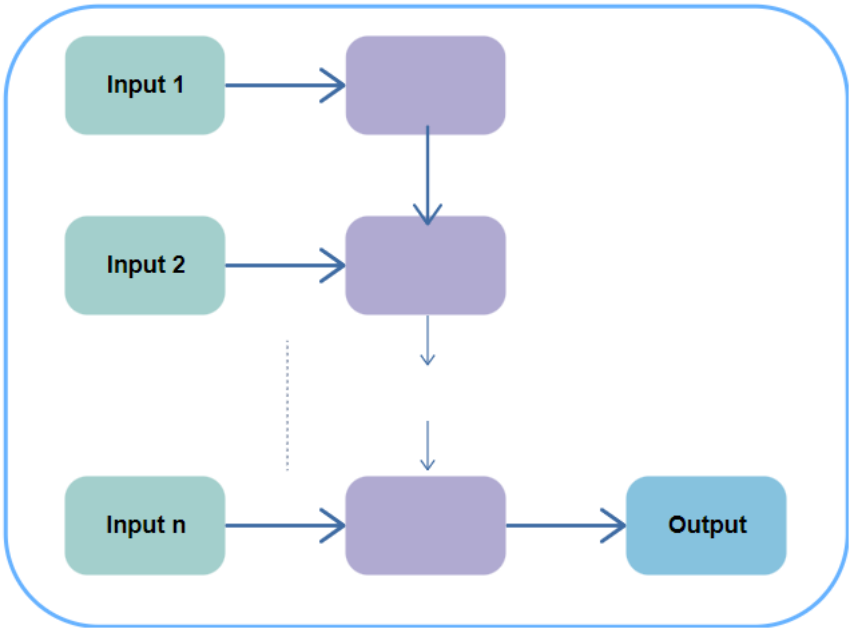
Its applications can be found in Music Generation and Image Captioning

Types of Recurrent Neural Network

Many-to-One RNN

Many-to-One is used when a single output is required from multiple input units or a sequence of them.

It takes a sequence of inputs to display a fixed output.



Sentiment Analysis is a common example of this type of Recurrent Neural Network.

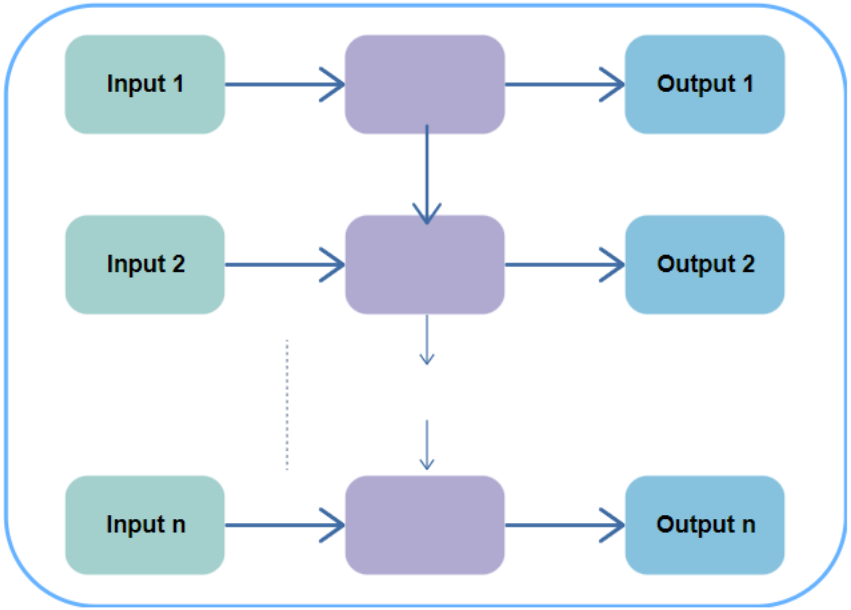
Types of Recurrent Neural Network

Many-to-Many RNN

Many-to-Many is used to generate a sequence of output data from a sequence of input units.

This type of RNN is further divided into the following two subcategories:

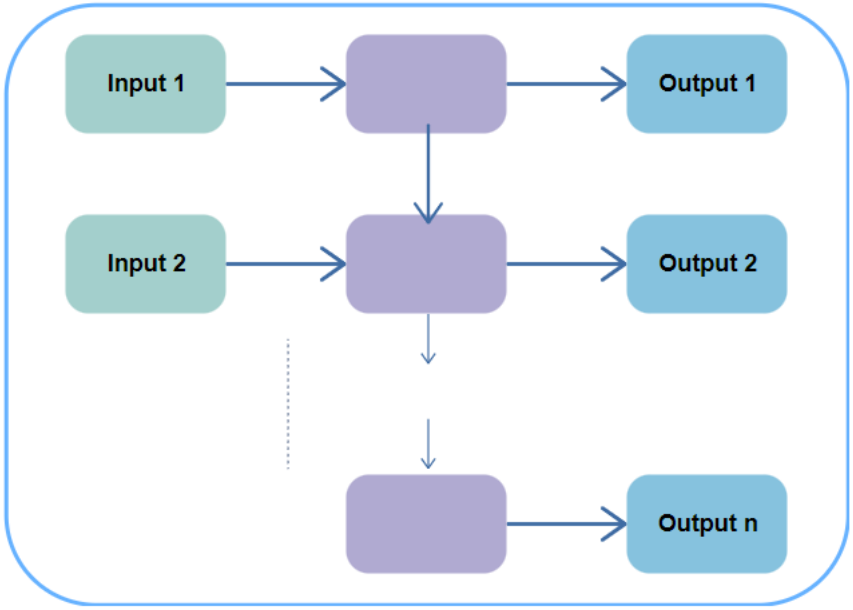
1. Equal Unit Size: In this case, the number of both the input and output units is the same. A common application can be found in Name-Entity Recognition.



Types of Recurrent Neural Network

Many-to-Many RNN

2. Unequal Unit Size: In this case, inputs and outputs have different numbers of units. Its application can be found in *Machine Translation*.

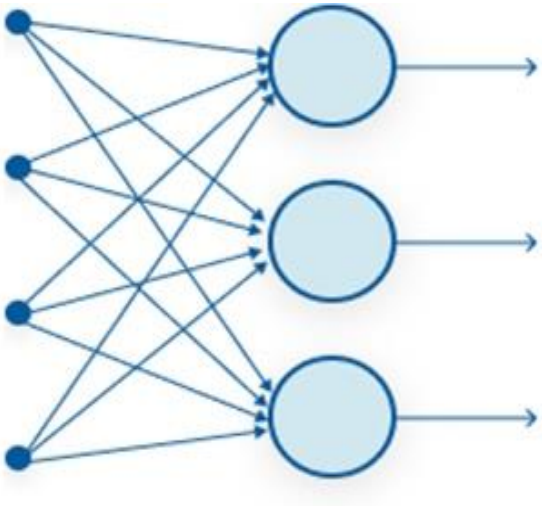


Language Translation, Video Caption recognition etc.

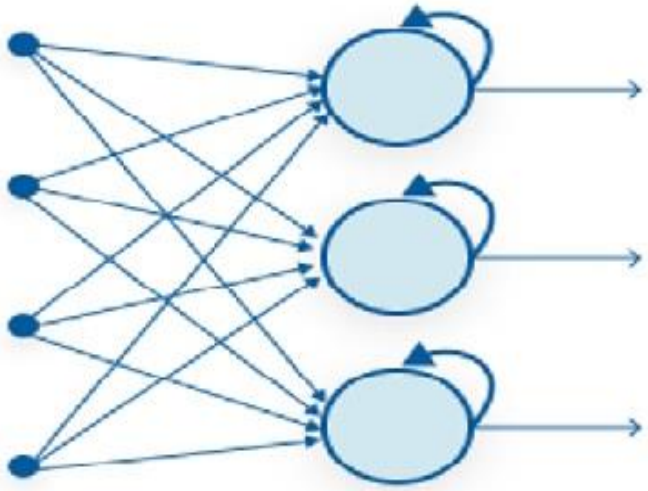
Recurrent Vs Feedforward Neural Network

A feed-forward neural network has only one route of information flow: from the input layer to the output layer, passing through the hidden layers. The data flows across the network in a straight route, never going through the same node twice.

The information flow between an RNN and a feed-forward neural network is depicted in the two figures below.



Feed-Forward Neural Network



Recurrent Neural Network

Two issues of Standard RNNs

A gradient is a partial derivative. A gradient quantifies how much the output of a function varies when the inputs are changed slightly.

A function's slope is also known as its gradient.

The steeper the slope, the faster a model can learn, the higher the gradient.

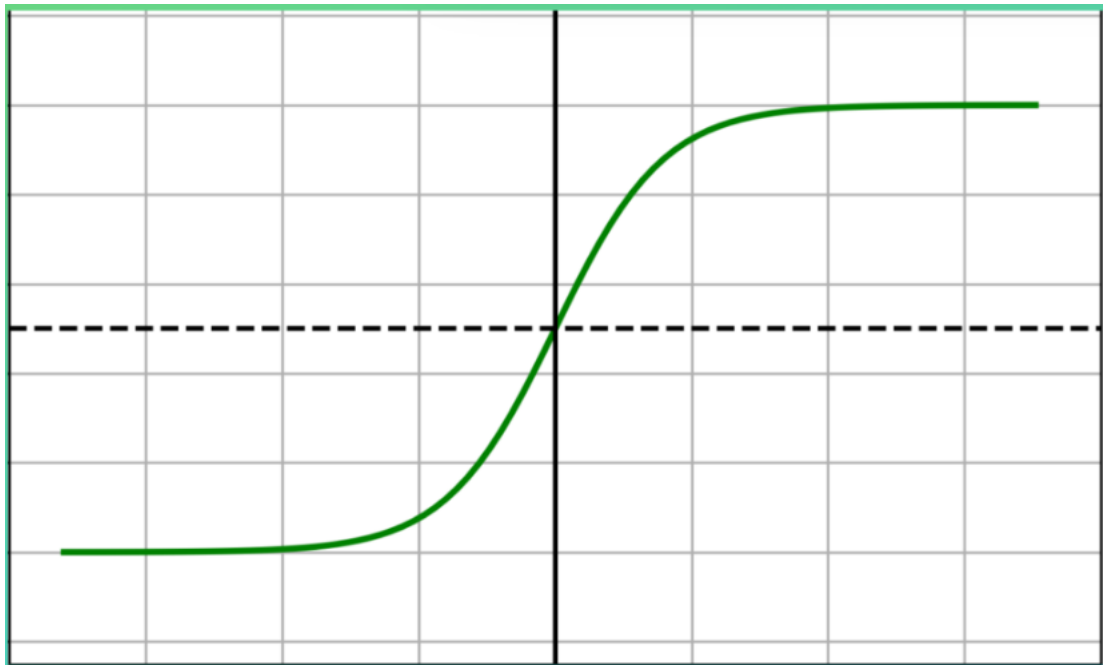
The model, on the other hand, will stop learning if the slope is zero.

A gradient is used to measure the change in all weights in relation to the change in error.

Two issues of Standard RNNs

Vanishing Gradients: Vanishing gradients occur when the gradient values are too small, causing the model to stop learning or take far too long. This was a big issue in the 1990s, and it was far more difficult to address than the exploding gradients. Fortunately, Sepp Hochreiter and Juergen Schmidhuber's LSTM concept solved the problem.

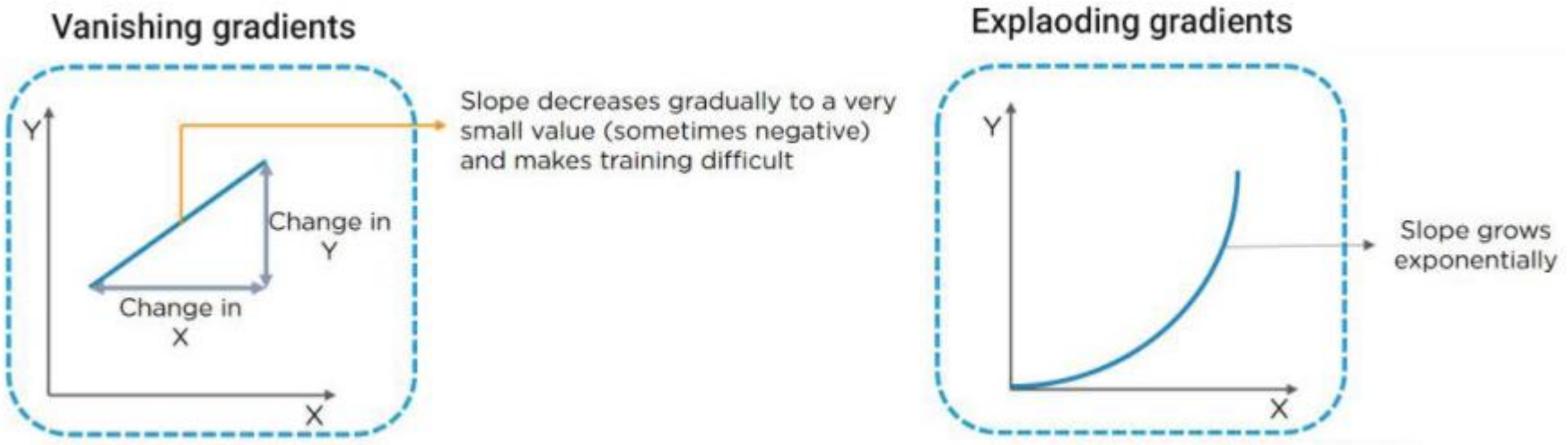
The vanishing gradient problem describes a situation encountered in the training of neural networks where the gradients used to update the weights shrink exponentially.
As a consequence, the weights are not updated anymore, and learning stalls



Two issues of Standard RNNs

Exploding Gradients: Exploding gradients occur when the algorithm gives the weights an absurdly high priority for no apparent reason. Fortunately, truncating or squashing the gradients is a simple solution to this problem.

Exploding gradients can cause problems in the training of artificial neural networks. When there are exploding gradients, **an unstable network can result and the learning cannot be completed. The values of the weights can also become so large as to overflow and result in something called NaN values**



Why Vanishing Gradient Problem Occurs

- Recurrent Neural Networks uses a hyperbolic tangent function, what we call the tanh function.
- The range of this activation function lies between [-1,1], with its derivative ranging from [0,1].
- Now we know that RNNs are a deep sequential neural network. Hence, due to its depth, the matrix multiplications continually increase in the network as the input sequence keeps on increasing.
- Hence, while we use the **chain rule of differentiation during calculating backpropagation**, **the network keeps on multiplying the numbers with small numbers.**
- **When you keep on multiplying a number with negative values with itself? It becomes exponentially smaller, squeezing the final gradient to almost 0, hence weights are no more updated, and model training halts.**
- It leads to poor learning, which we say **as "cannot handle long term dependencies"** when we speak about RNNs.

Why Exploding Gradient Problem Occurs

- Similar concept to the vanishing gradient problem, but just the opposite of the process.
- Suppose in this case our **gradient value is greater than 1 and multiplying a large number to itself makes it exponentially larger leading to the explosion of the gradient.**

Recurrent Neural Networks Applications

Recurrent Neural Networks are used to tackle a variety of problems involving sequence data.

There are many different types of sequence data, but the following are the most common: Audio, Text, Video, Biological sequences.

Using RNN models and sequence datasets, you may tackle a variety of problems, including :

- ❖ Speech recognition
- ❖ Generation of music
- ❖ Automated Translations
- ❖ Analysis of video action
- ❖ Sequence study of the genome and DNA

A Brief Overview of Recurrent Neural Networks (RNN) - Analytics Vidhya

Recurrent Neural Networks Applications

"A Dog catching a ball in mid air"

Image captioning

RNN is used to caption an image by analyzing the activities present in it

Recurrent Neural Networks Applications

Time series prediction

Any time series problem like predicting the prices of stocks in a particular month can be solved using RNN

Recurrent Neural Networks Applications

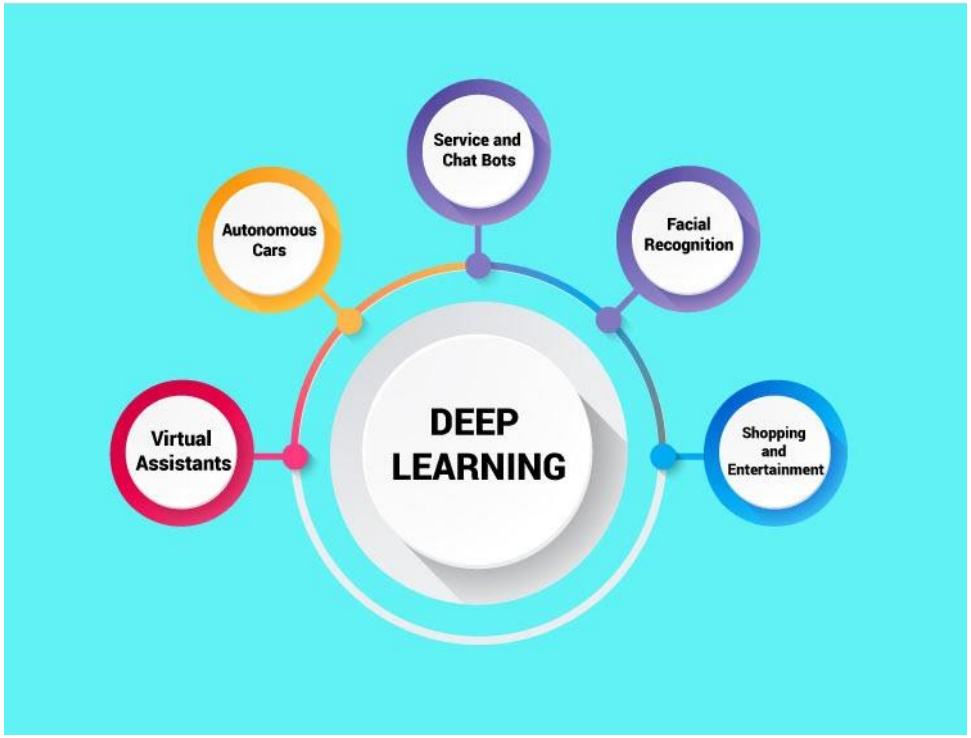
When it rains, look for rainbows.
When it's dark, look for stars.

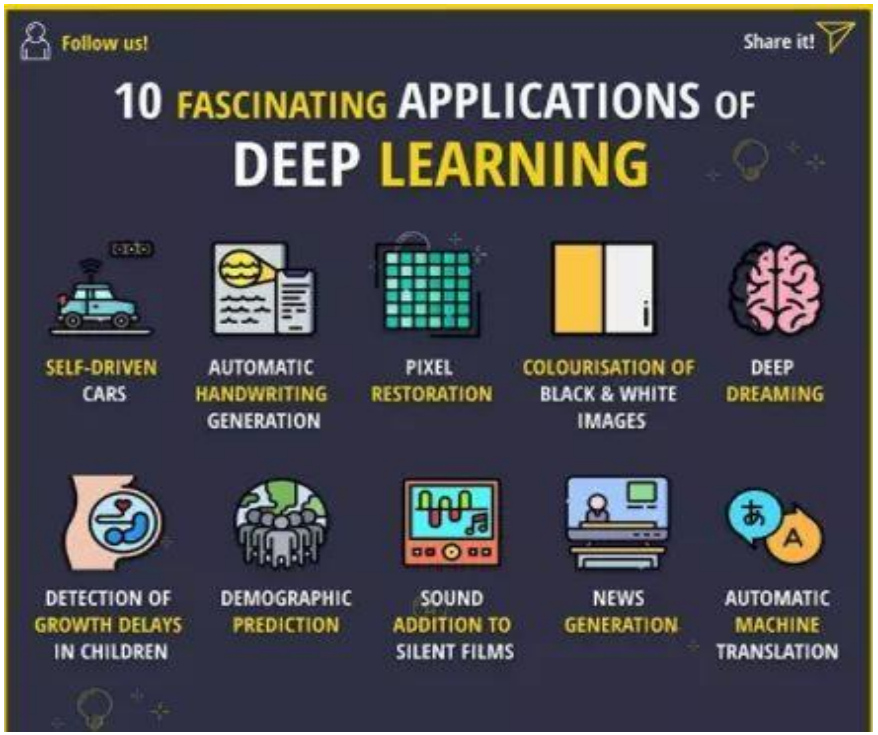
Positive Sentiment

Natural Language Processing

Text mining and Sentiment analysis can be carried out using RNN for Natural Language Processing

Applications of Deep Learning





Applications of Deep Learning

Autonomous Driving Cars

Distinguishes different types of objects, people, road signs and drives without human intervention

Applications of Deep Learning

Building Robots

Deep Learning is used to train robots to perform human tasks

Applications of Deep Learning

Medical Diagnosis

Deep Neural Nets are used to identify suspicious lesions and nodules in lung cancer patients

ANN, CNN & RNN Comparison

	ANN	CNN	RNN
Basics	One of the simplest types of neural networks.	One of the most popular types of neural networks.	The most advanced and complex neural network.
Structural Layout	Its simplicity comes from its feed forward nature — information flows in one direction only.	Its structure is based on multiple layers of nodes including one or more convolutional layers.	Information flows in different directions, which gives it its memory and self-learning features.
Data Type	Fed on tabular and text data.	Relies on image data.	Trained with sequence data.
Complexity	Simple in contrast with the other two models.	Considered more powerful than the other two.	Fewer features than CNN but powerful due to its self-learning & memory potential.
Commendable Feature	Ability to work with incomplete knowledge and high fault tolerance.	Accuracy in recognizing images.	Memory and self-learning.
Feature type: spatial recognition	No	Yes	No
Feature type: Recurrent connections	No	No	Yes
Main Drawback	Hardware dependence.	Large training data required.	Slow and complex training and gradient concerns.
Uses	Complex problem solving such as predictive analysis.	Computer vision including image recognition	Natural language processing including sentiment analysis and speed recognition.

Top 8 Frameworks


ThankYou
Merci
Salamat
Kop
Tack
Hvala
Xie
Gra
kun
Diolch
Kiitos
Sheun
umesc
Kasih
Mamnoon
Todah
Shnorhakalutiun
Shokriya
Dziękuję
Shokrun
Spaas
Mul
Ačí
Gamsahapnida
Teşekkür
Dekuju/Dekujeme
Ngiyabonga
Dank
Takk
Cam
or
al
Dhanyavad
Grazie
Faleminderit
Dakujem
Waad
Daw
Dhanyavaadalu
Dankie
Go
Kruthagnathalu
Khopjai
Arigatou
krap
Dhonnobaad
Or
ederim
Asante
Hain
Dhan
daa
Grazzi
raibh
Gracias
Nandree
Blagodariya
Gomapsupnida
Fyrir
Terima
Enkosi
Danke
dank
Euxaristo
Kun
Shukriya