Deep Learning Architecture –20CS721

(Elective Course – 7th Semester CS&E)

Dr.Srinath.S, Associate Professor & HoD

Department of Computer Science and Engineering

SJCE, JSS S&TU, Mysuru- 570006

Pre - Requisite

- Linear Algebra
- Elementary Probability and Statistics
- Machine Learning / Pattern Recognition.
- Programming skills – Python preferred

Course Outcomes

- After completing this course, students should be able to:

Course Outcomes:

CO1	Discuss the deep learning algorithms which are more appropriate for various types of learning tasks in various domains.
CO2	Implement Deep Neural Network.
CO3	Design the Convolutional Neural Network.
CO4	Design the Recurrent Neural Networks.
CO5	Analyse the concept of Autoencoders and Reinforcement Learning.

Syllabus and Text Book Details

Unit No.	Course Content	No. of Hours
1	**Introduction to ANN:** Biological to Artificial neuron, Training an MLP, training a DNN with TensorFlow, Fine tuning NN Hyper Parameters Up and Running with TensorFlow	7
2	**Deep Neural network:** Introduction, Vanishing Gradient problems, Reusing pretrained layers, Faster optimizers, Avoiding over fitting through regularization.	8
3	**Distributing Tensor flow across devices and servers:** Multiple devices on a single machine, multiple servers, parallelizing NN on a Tensor Flow cluster Convolution Neural Network: Architecture of the visual cortex, Convolutional layer, Pooling layer, CNN architecture.	8
4	**Recurrent Neural Network:** Recurrent neurons, Basic RNN in Tensor Flow, Training RNN, Deep RNNs, LSTM Cell, GRU Cell, NLP.	8
5	**Autoencoders:** Efficient data representation, Performing PCA, stacked autoencoders, Unsupervised pretraining using SA, Denoising, Sparse auto encoders, variational and other autoencoders. Reinforcement Learning: Learning to optimize rewards.	8

Text Book:

Sl. No.	Author/s	Title	Publisher Details
1	Aurelien Geron	Hands on Machine Learning with Scikit-Learn & Tensor Flow	O'Reilly, 2019

Reference Books:

Sl. No.	Author/s	Title	Publisher Details
1	Lan Good fellow and Yoshua Bengio and Aaron Courville	Deep Learning	MIT Press2016
2	Charu C. Aggarwal	Neural Networks and Deep Learning	Springer International Publishing, 2018
3	Andrew W. Trask	Grokking Deep Learning	Manning Publications
4	Sudharsan Ravichandran	Hands-On Learning Algorithms with Python	

Sl. No.	Web Link
1	https://onlinecourses.nptel.ac.in/noc20_cs62/preview
2	https://nptel.ac.in/courses/106/105/106105215/

Course Outcomes:

CO1	Discuss the deep learning algorithms which are more appropriate for various types of learning tasks in various domains.
CO2	Implement Deep Neural Network.
CO3	Design the Convolutional Neural Network.
CO4	Design the Recurrent Neural Networks.
CO5	Analyse the concept of Autoencoders and Reinforcement Learning.

Mapping Course Outcomes with Program Outcomes & Program Specific Outcomes:

Course Outcomes	Program Outcomes												PSO's			
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	P012	PSO1	PSO2	PSO3	PSO4
CO1	3	3	-	-	-	-	-	-	-	-	-	-	3	-	-	-
CO2	3	3	3	3	-	-	-	-	-	-	-	-	3	-	3	-
CO3	3	3	3	3	-	-	-	-	-	-	-	-	3	-	3	3
CO4	3	3	3	3	-	-	-	-	-	-	-	-	3	-	3	3
CO5	3	3	3	3	-	-	-	-	-	-	-	-	3	-	3	-

1 - Low association, 2 - Moderate association, 3 - High association

Assessment Weightage in Marks

Class Test –I	10
Quiz/Mini Projects/ Assignment/ seminars	10
Class Test – II	10
Quiz/Mini Projects/ Assignment/ seminars	10
Class Test – III	10
Total	50

Question Paper Pattern

- **Semester End Examination (SEE)**
- **Semester End Examination (SEE) is a written examination of three hours duration of 100 marks with 50% weightage.**

- **Note:**
- **• The question paper consists of TWO parts PART- A and PART- B.**

- **• PART- A consists of Question Number 1-5 are compulsory (ONE question from each unit)**

- **• PART-B consists of Question Number 6-15 will have internal choice. (TWO question from each unit)**

- **• Each Question carries 10 marks and may consist of sub-questions.**
- **• Answer 10 full questions of 10 marks each**

Source:

- Material is based on *Hands-On Machine Learning with Scikit_Learn and TensorFlow: Concepts, Tools and Techniques* (by Aurelien Geron), Wikipedia, and other sources.

UNIT – 1 Introduction to ANN:

- **Introduction to ANN:** Biological to Artificial neuron, Training an MLP, training a DNN with TensorFlow, Fine tuning NN Hyper Parameters Up and Running with TensorFlow

Quick look into ML

MACHINE LEARNING

Dr.Shrinath.S
Associate Professor, Department of CS&E
Sri Jayachamarajendra College of Engineering,
JSS Science and Technology University, Mysuru- 570006

Introduction

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Deep Learning (DL)
- Data Science

Artificial Intelligence

- Artificial intelligence is intelligence demonstrated by machines, as opposed to natural intelligence displayed by animals including humans.

Machine Learning

- Machine Learning – Statistical Tool to explore the data.

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

If you are searching some item in amazon… next time… without your request… your choice will be listed.

Variants of Machine Learning:

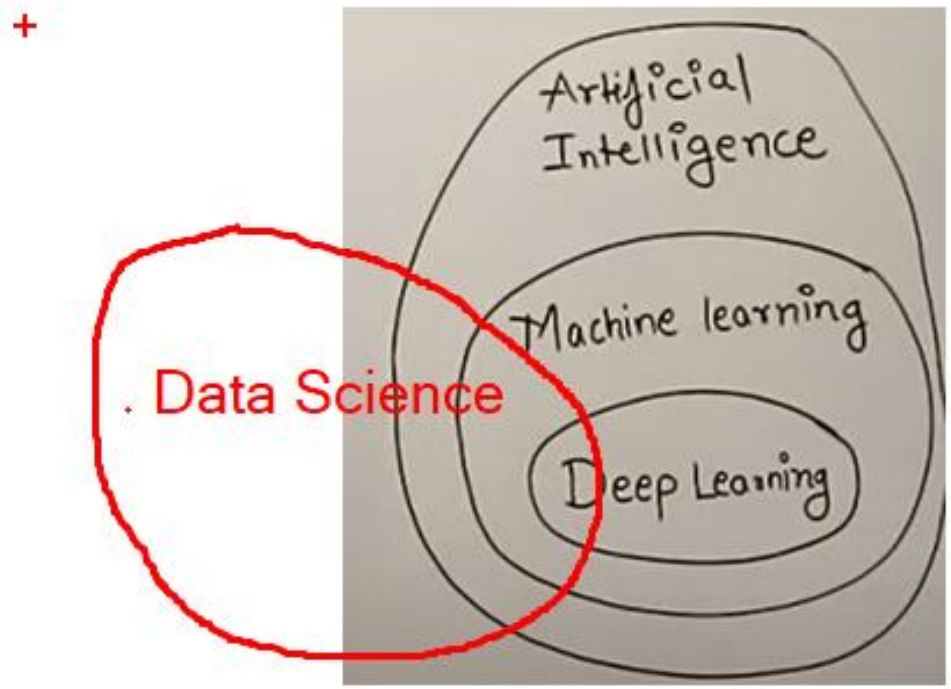
- Supervised
- Unsupervised
- Semi supervised
- Reinforcement Learning

Deep Learning

- It is the subset of ML, which mimic human brain.

- Three popular Deep Learning Techniques are:
  - ANN – Artificial Neural Network
  - CNN- Convolution Neural Network
  - RNN- Recurrent Neural Network

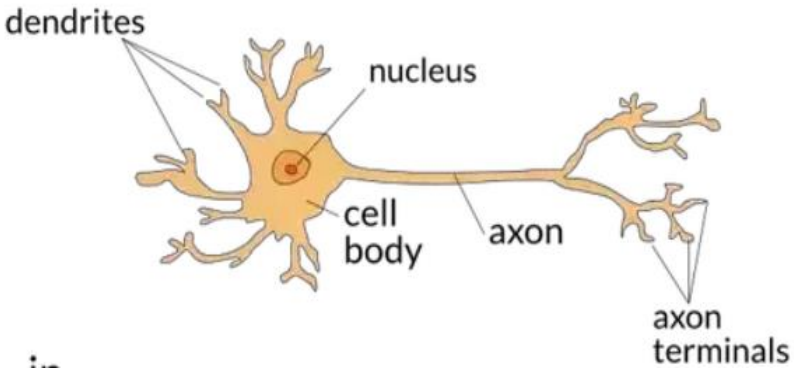
Summary:

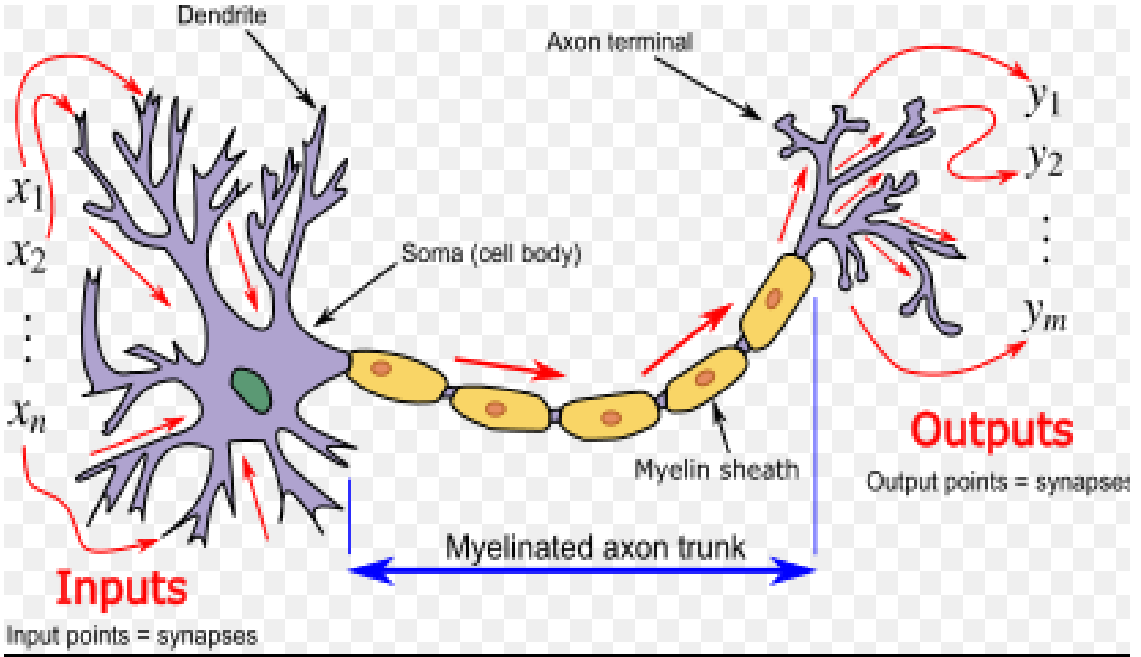

Artificial Intelligence
Machine learning
Deep Learning
Data Science

Introduction to ANN

Biological Neural Network to ANN

Biological Neural Network (BNN)

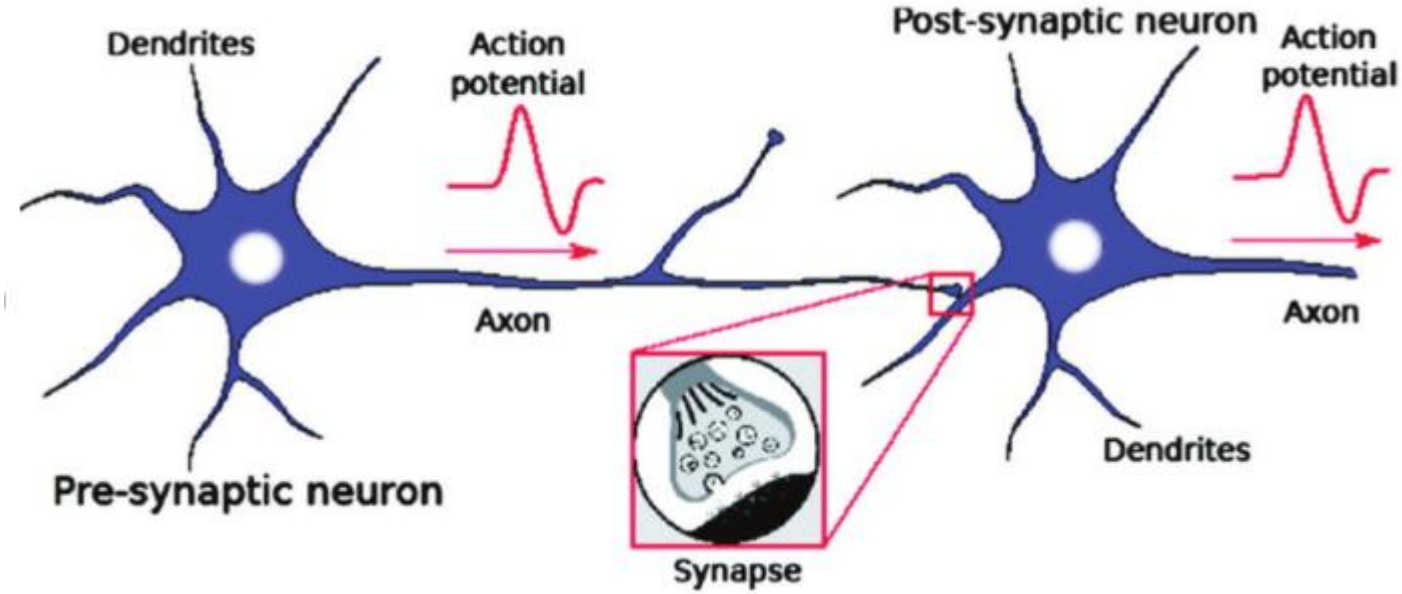




BNN parts

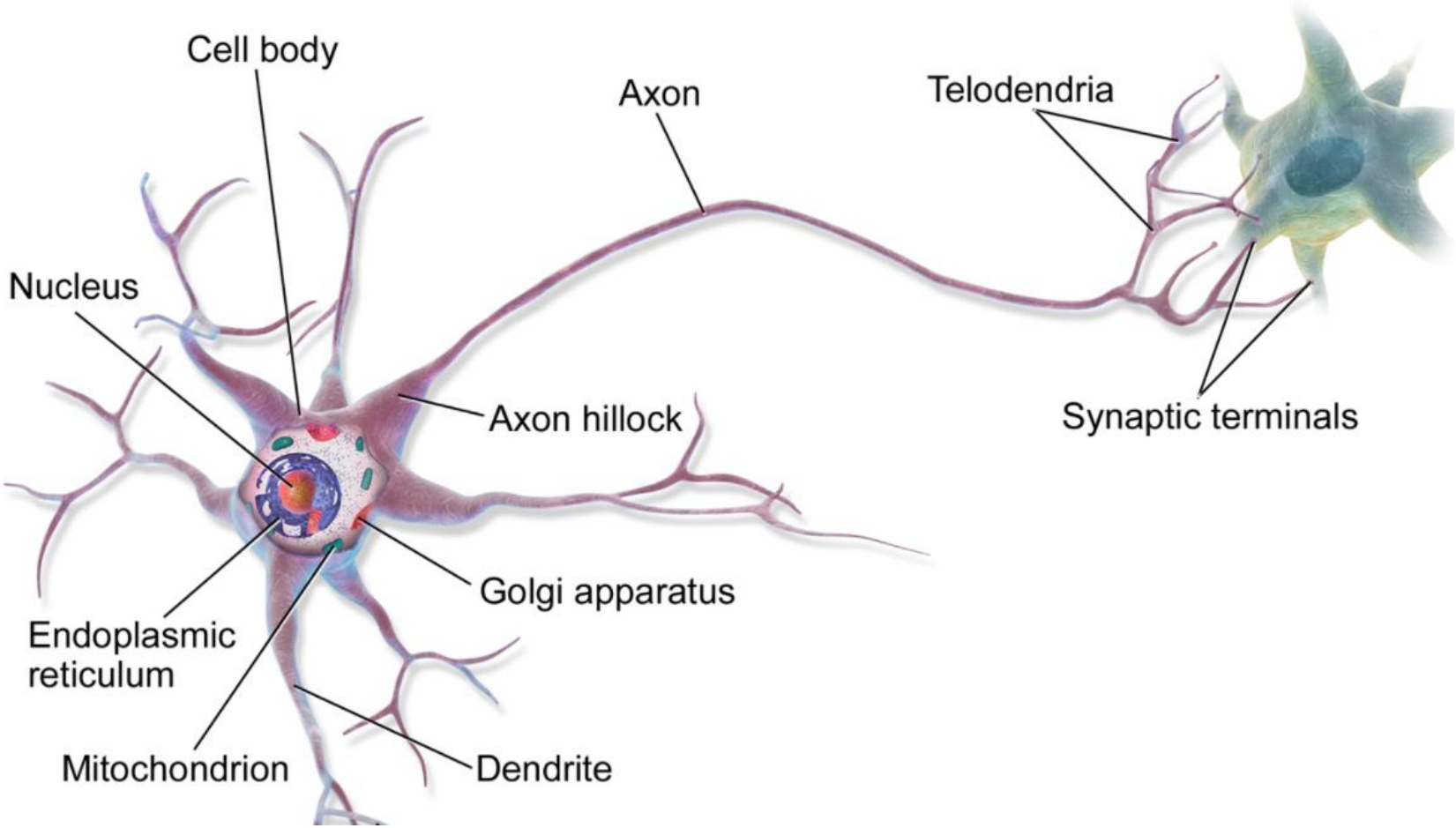
- BNN is composed of a cell body and many branching extensions called **dendrites** and one long extension called the **axon**.
- Primarily the parts of BNN are:
  - Cell body
  - Dendrites – Input part
  - Axon - output
- BNN is an interconnection of several biological neurons.
- Interconnection between two neurons is as shown in the next slide

Two neurons interconnected



- At the end AXON splits off into many branches called **telodendrion** and the tip of these branches are called synaptic terminals or simply **synapses.**
- The synapses of one neurons are connected to the dendrites of other neurons.
- Electric impulses called signals are passed from one neuron to another.
- BNN is a collection of billions of neurons, and each neurons are typically connected to thousands of other neurons.

Another view of BNN interconnection



Multiple layers in a biological network

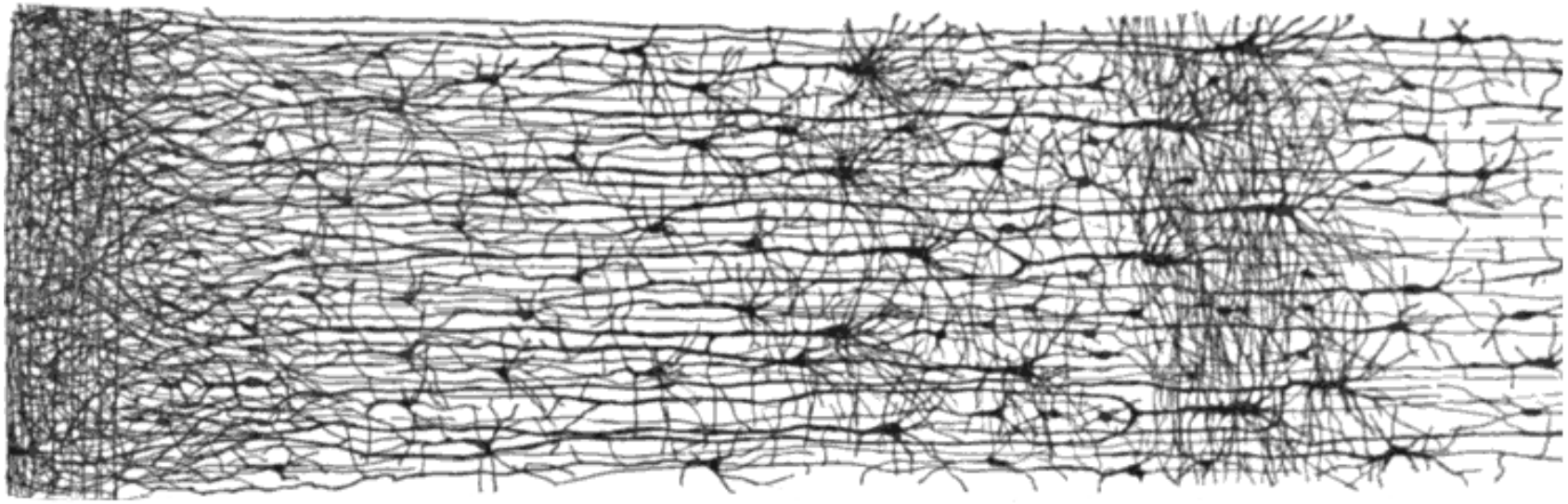


Figure 1-2. Multiple layers in a biological neural network (human cortex)[5]

Artificial Neural Network (ANN)

Logical Computations with Neurons

- The artificial neuron simply activates its output when more than a certain number of its inputs are active.
- Let us see some of the ANNs performing simple logical computations.

ANNs performing simple logical computations

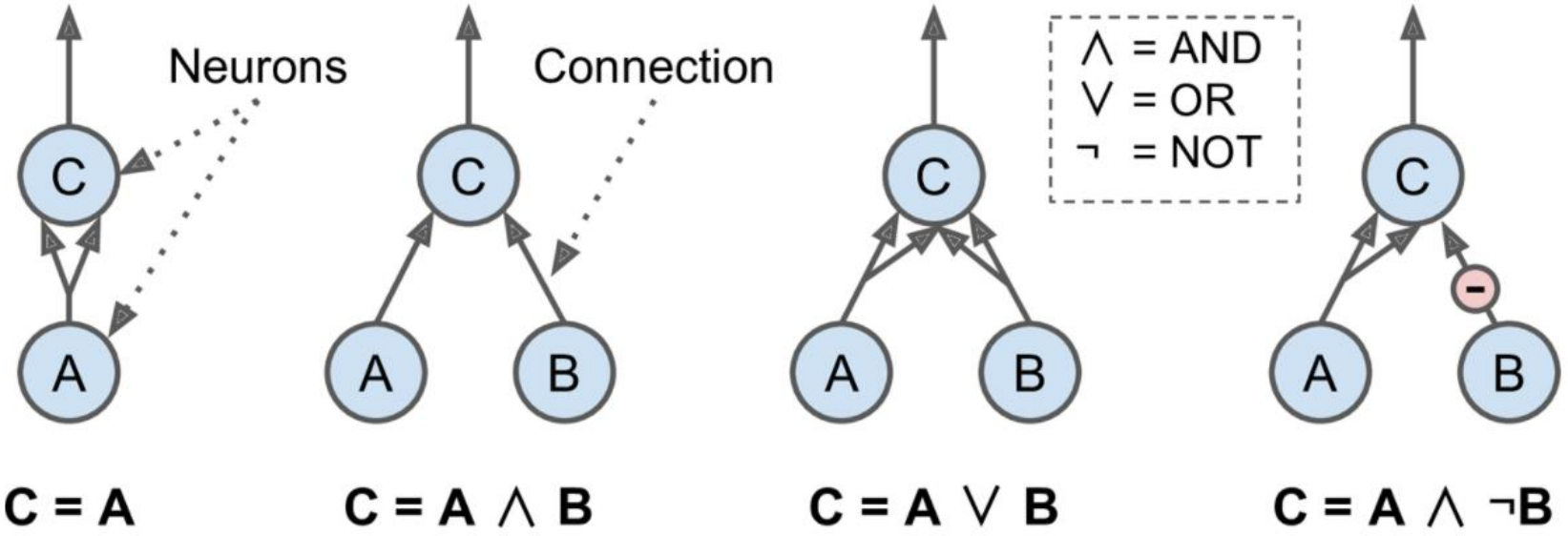


Figure 1-3. ANNs performing simple logical computations

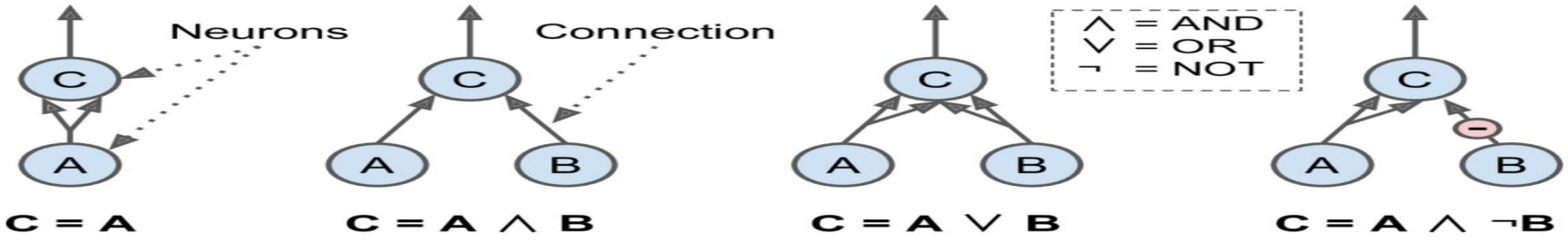


Figure 1-3. ANNs performing simple logical computations

- The first network on the left is simply the identity function: if neuron A is activated, then neuron C gets activated as well (since it receives two input signals from neuron A), but if neuron A is off, then neuron C is off as well.

- The second network performs a logical AND: neuron C is activated only when both neurons A and B are activated (a single input signal is not enough to activate neuron C).

- The third network performs a logical OR: neuron C gets activated if either neuron A or neuron B is activated (or both).

- Finally the fourth network computes a slightly more complex logical proposition: neuron C is activated only if neuron A is active and if neuron B is off. If neuron A is active all the time, then you get a logical NOT: neuron C is active when neuron B is off, and vice versa.

Perceptron

- ***Perceptron is a single layer neural network*** *or simply a neuron.*
- *So perceptron is a ANN with single layer neural network without having hidden layers.*

Perceptron consists of 4 parts

- input values
- weights and a Constant/Bias
- a weighted sum, and
- Step function / Activation function

Linear threshold unit (LTU)

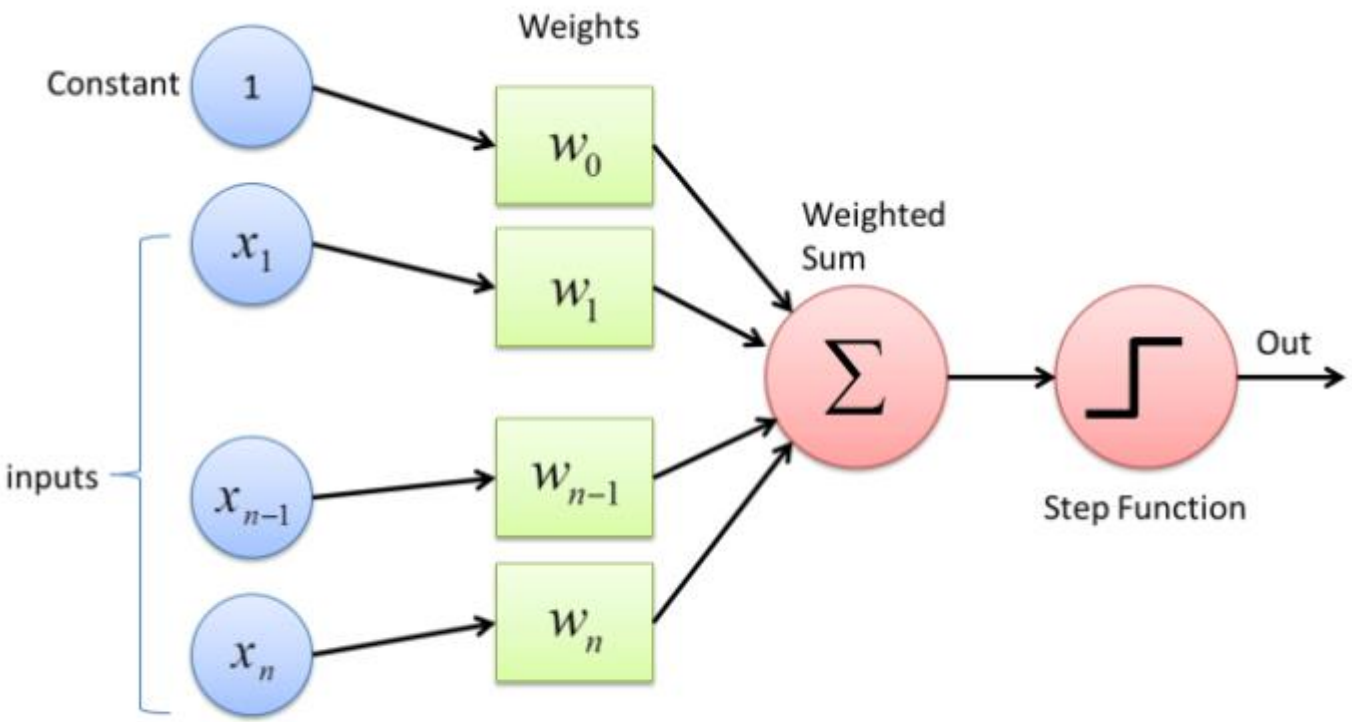


Fig : Perceptron

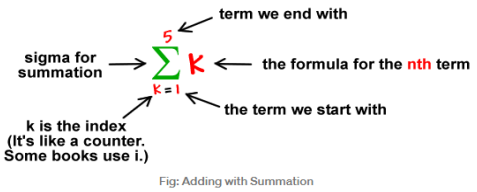


Fig: Adding with Summation

- A perceptron can have multiple input and single output as shown in the previous diagram (single LTU).
- Perceptron is simply composed of a single layer of LTUs.
- For example a 2 input and 3 output perceptron is as shown in the next slide.
- However a single layer perceptron will not have hidden layer.

A Perceptron with two inputs and three outputs is represented in Figure 10-5. This Perceptron can classify instances simultaneously into three different binary classes, which makes it a multioutput classifier.

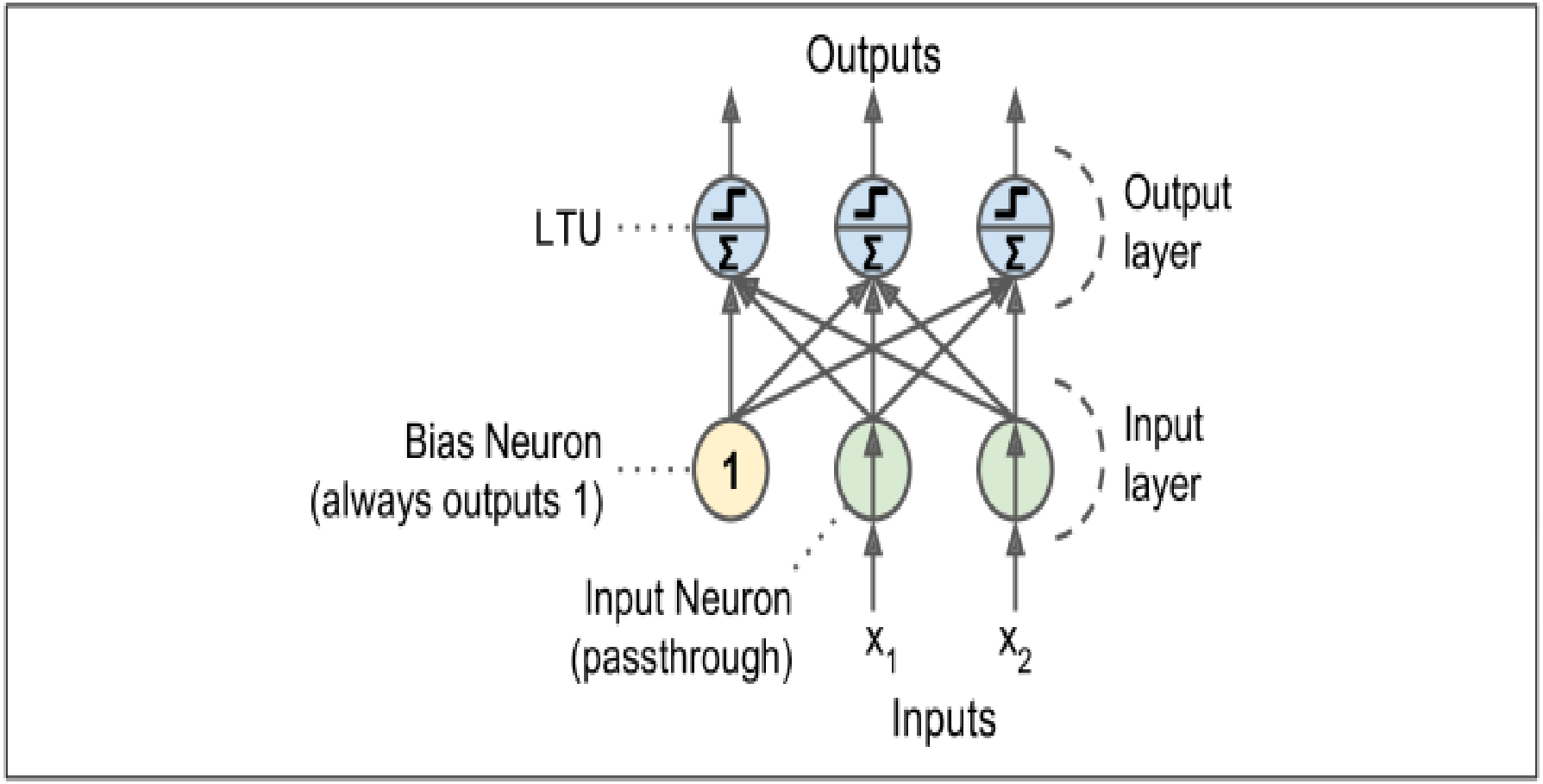


Figure 10-5. Perceptron diagram

Working of Perceptron

- The perceptron works on these simple steps:

- a. All the inputs ***x*** are multiplied with their weights ***w***. Let's call it ***k.***
- ***Add*** all the multiplied values and call them ***Weighted Sum.***
- ***Finally Apply*** that weighted sum to the correct ***Activation Function***.

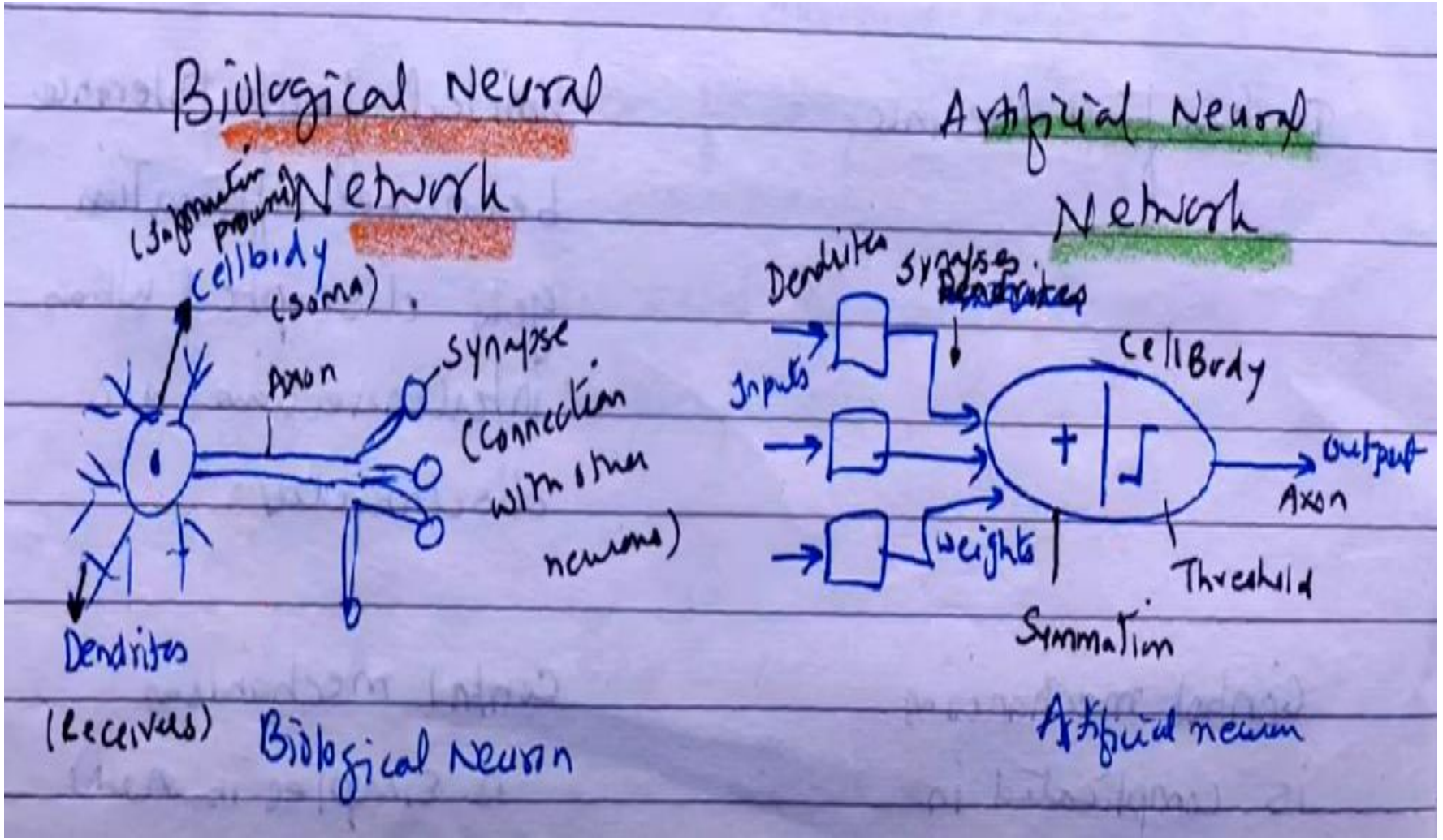
  For Example: Heaviside step function.

Step activation function

Equation 10-1. Common step functions used in Perceptrons

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

Comparison between BNN and ANN



Equation for the perceptron learning rule

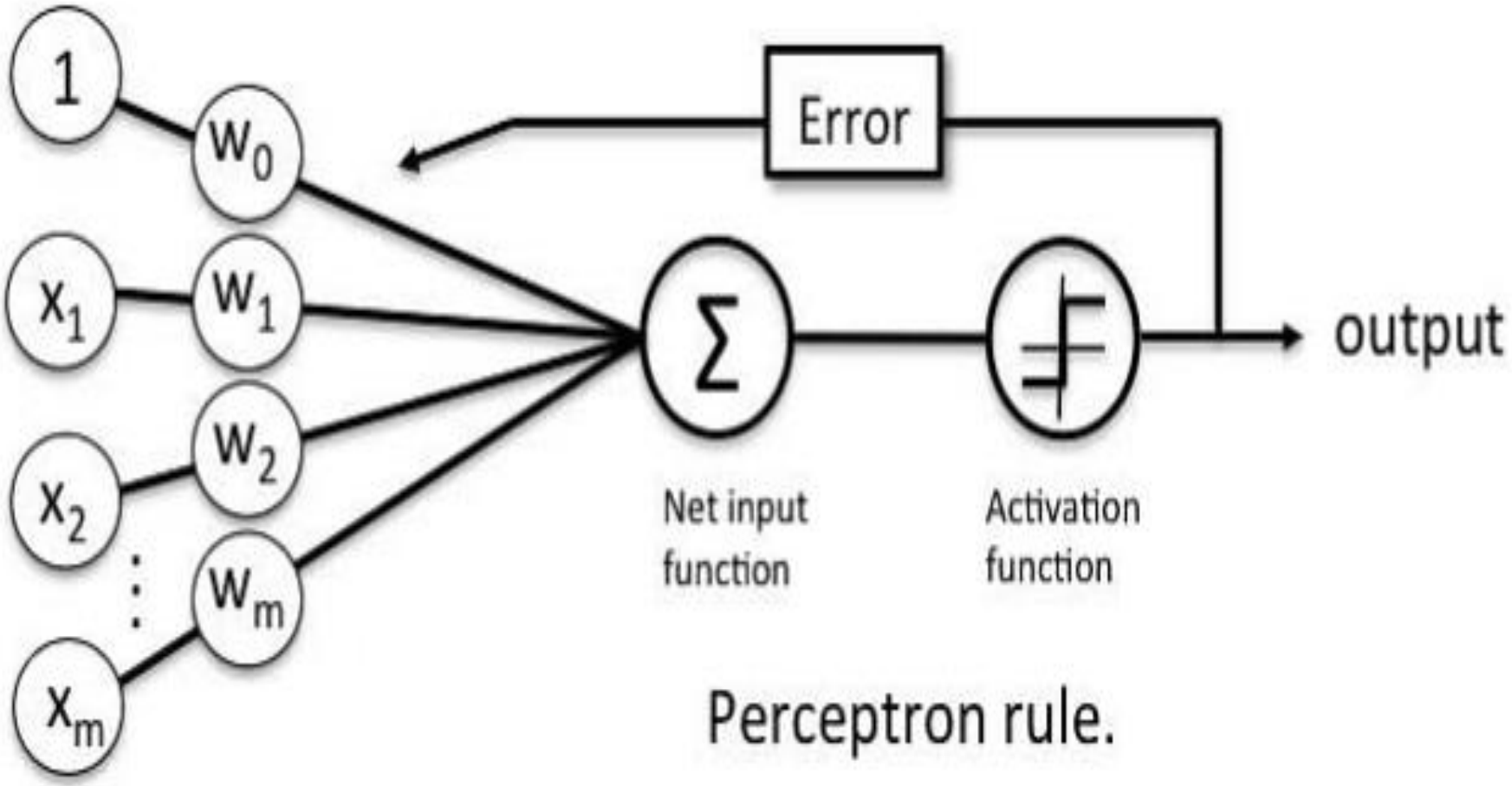
- Perceptrons are trained considering the error made by the network.
- For every output neuron that produced a wrong prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction.
- Equation is given in the next slide

- Perceptron learning rule (weight update)

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

- • wi, j is the connection weight between the ith input neuron and the jth output neuron.
- • xi is the ith input value of the current training instance.
- • ^yj is the output of the jth output neuron for the current training instance.
- • yj is the target output of the jth output neuron for the current training instance.
- • η is the learning rate.
- This process is repeated till the error rate is close to zero

Perceptron Learning Rule



- A perceptron is simply composed of a single layer of LTUs, with each neuron connected to all the inputs.
- some of the limitations of Perceptrons can be eliminated by stacking multiple Perceptrons.
- The resulting ANN is called a Multi-Layer Perceptron (MLP).
- MLP will have one or more hidden layers.

Multi Layer Perceptron (MLP)

An MLP is composed of one (passthrough) input layer, one or more layers of LTUs, called *hidden layers*, and one final layer of LTUs called the *output layer* (see Figure 10-7). Every layer except the output layer includes a bias neuron and is fully connected to the next layer. When an ANN has two or more hidden layers, it is called a *deep neural network* (DNN).

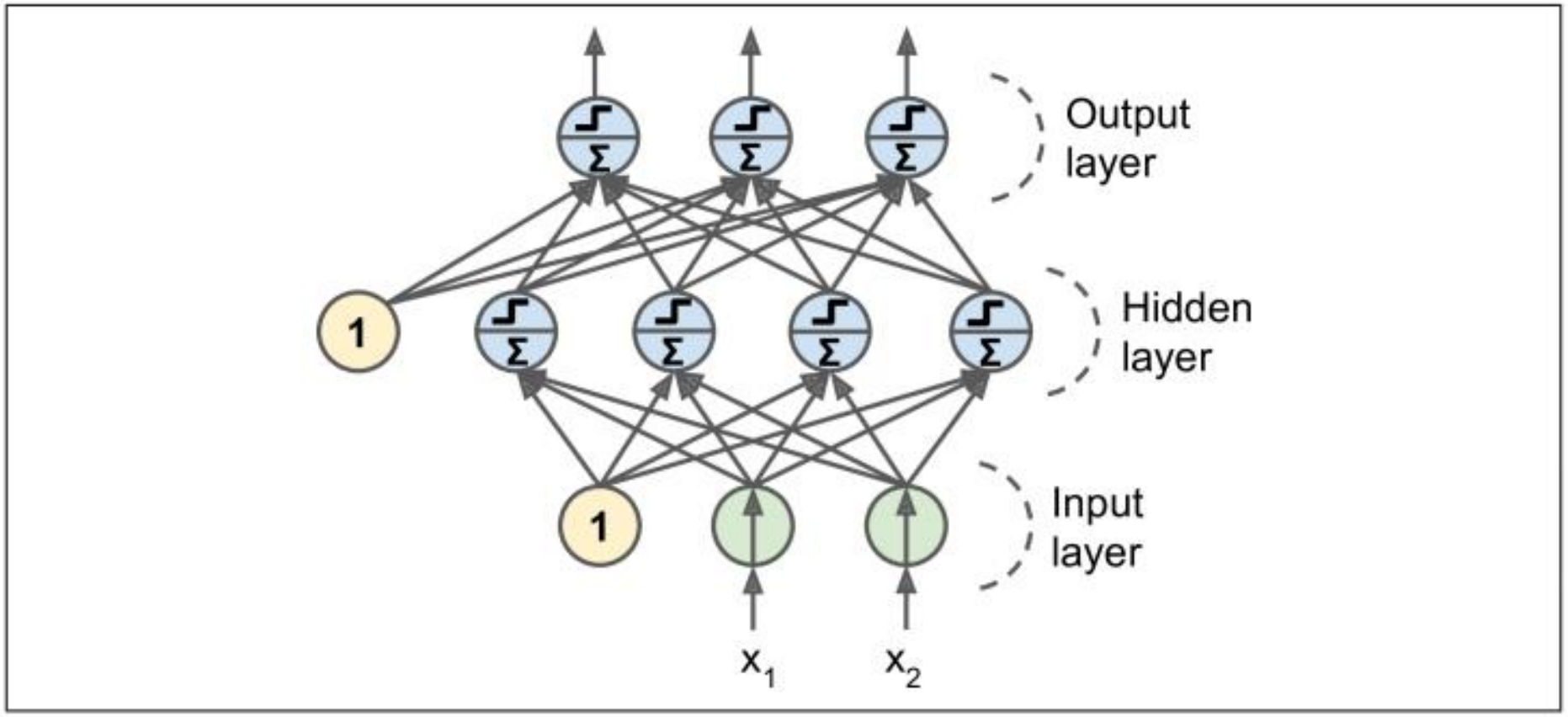
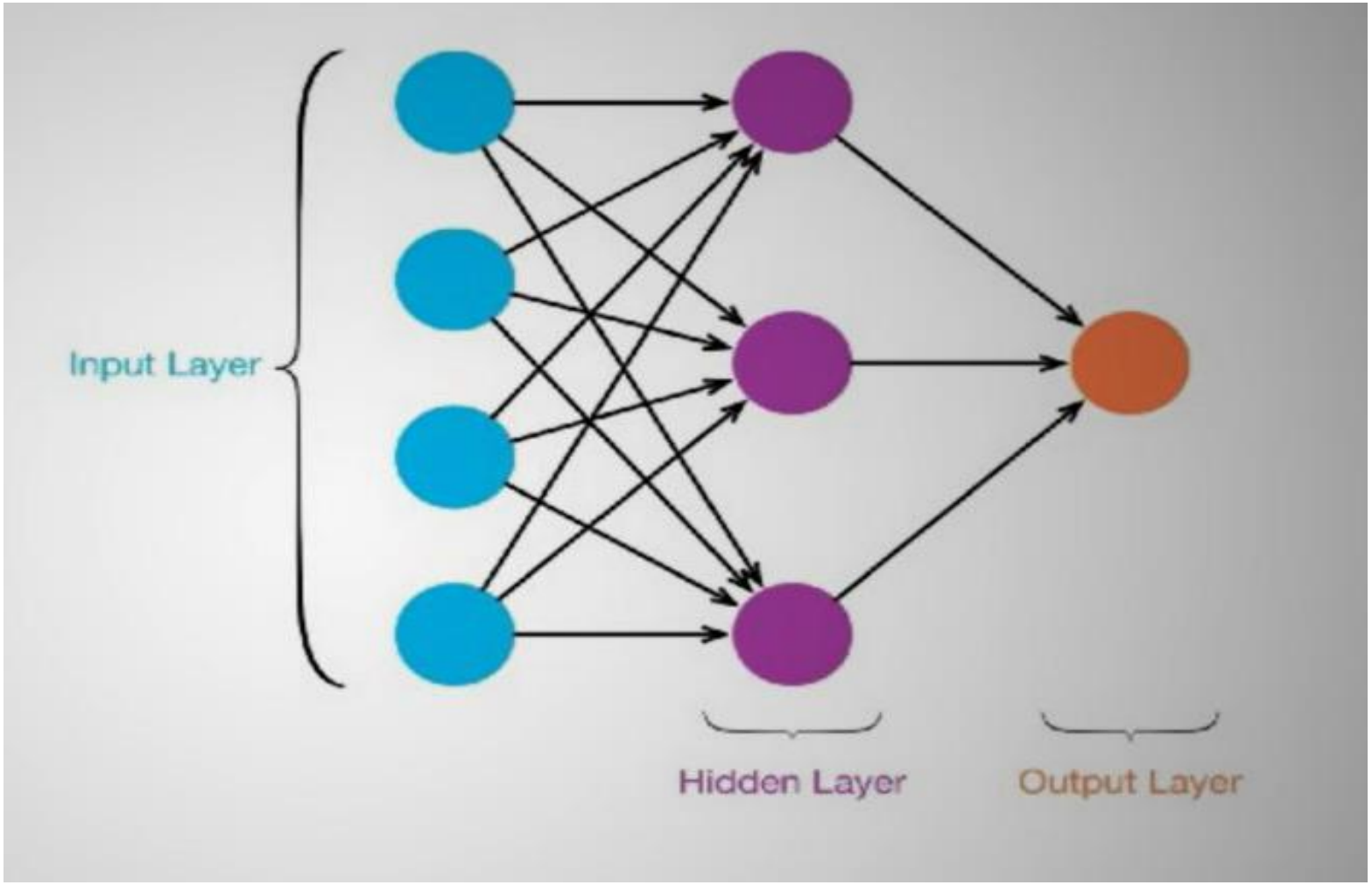
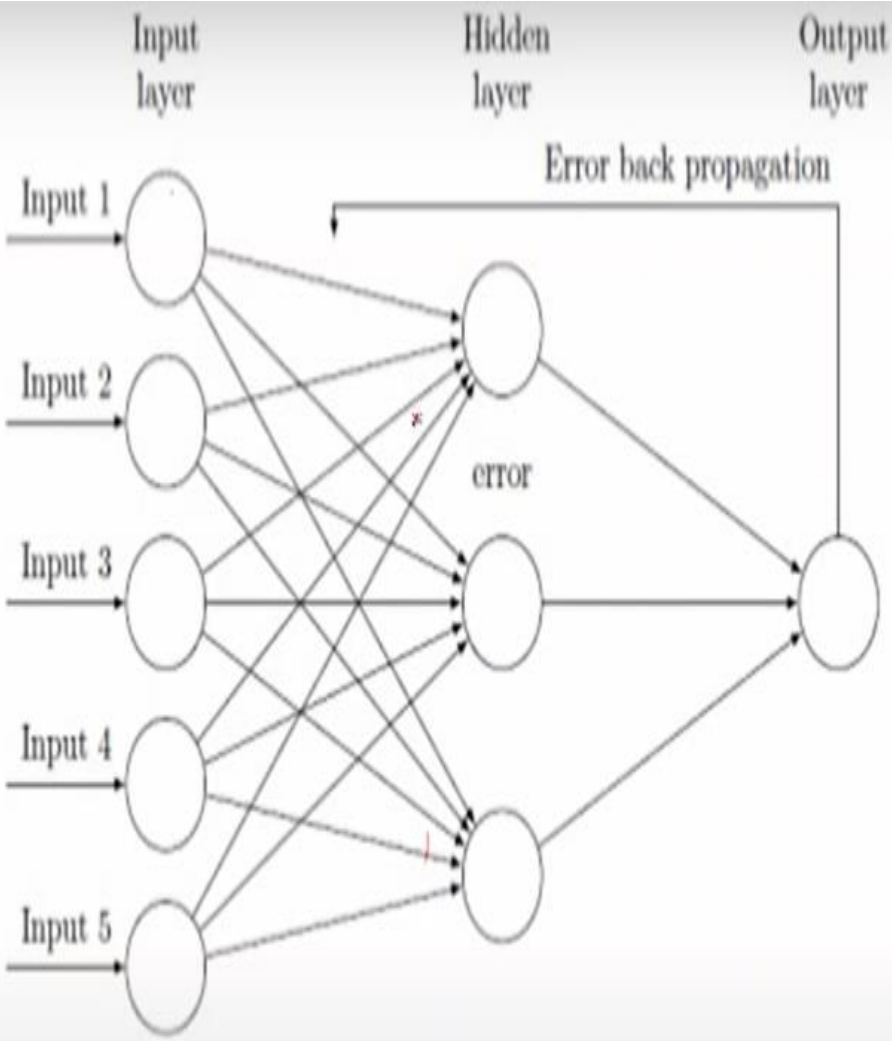


Figure 10-7. Multi-Layer Perceptron

MLP : Simplified view



Example for ANN



Banana?

Apple?

Mango?

Shallow or Deep ANN

- MLP can be either *shallow* or *deep*.
- They are called shallow when they have only one hidden layer (i.e. one layer between input and output).
- They are called deep when hidden layers are more than one. (Two or more)
- This is where the expression **DNN** (Deep Neural Network) comes.
- So DNN is a variant of ANN having 2 or more hidden layer.

Summary

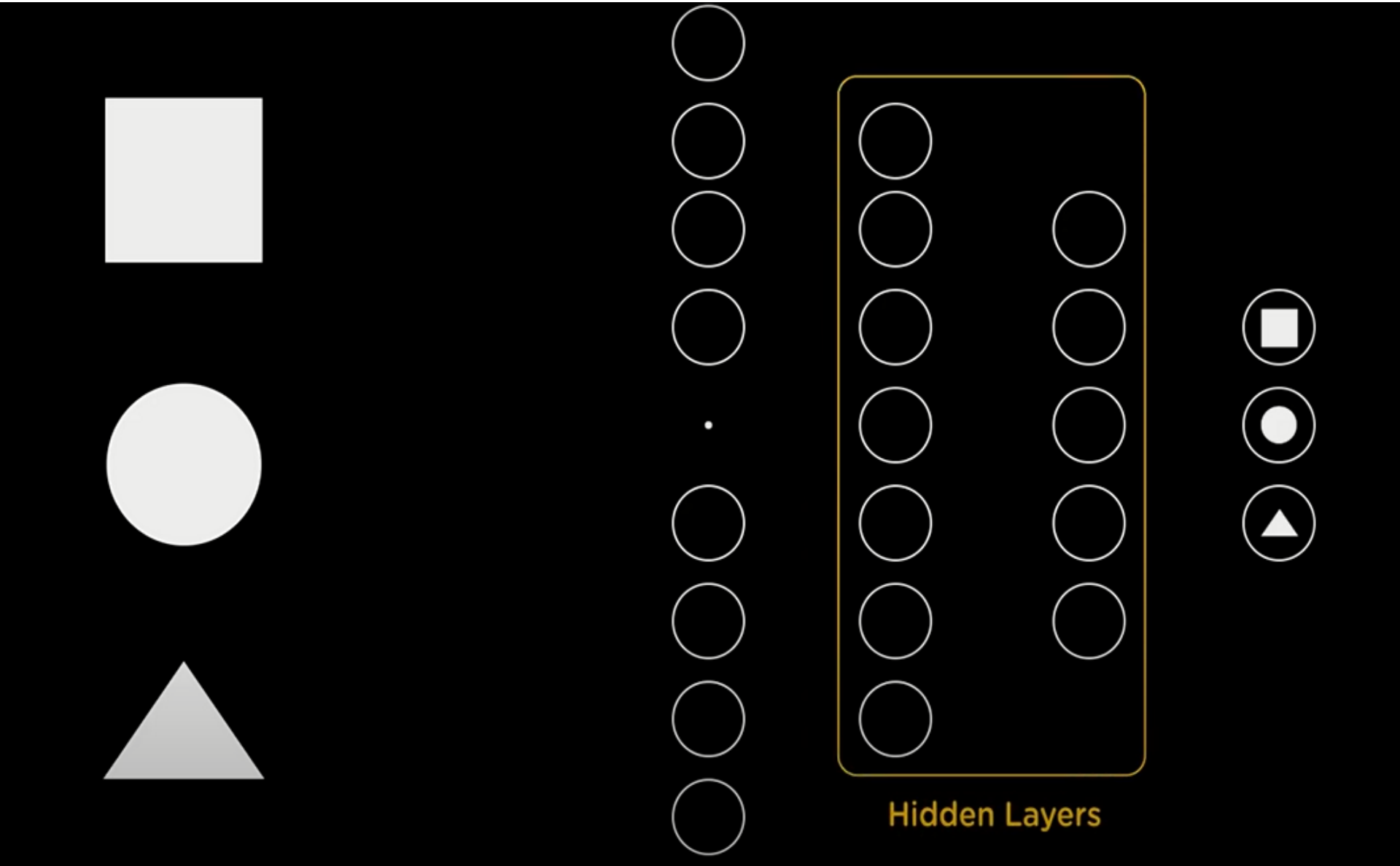
- **Perceptron:** It is a ANN with single layer neural network without having hidden layers. It will have only input and output layer.
- MLP – ANN with 2 or more layers called MLP
- MLP with only one hidden layer is called shallow ANN
- MLP with two or more hidden layers is called deep ANN, which is popularly known as Deep Neural Network.
- Perceptron, Shallow ANN, Deep ANN are all variants of ANN.

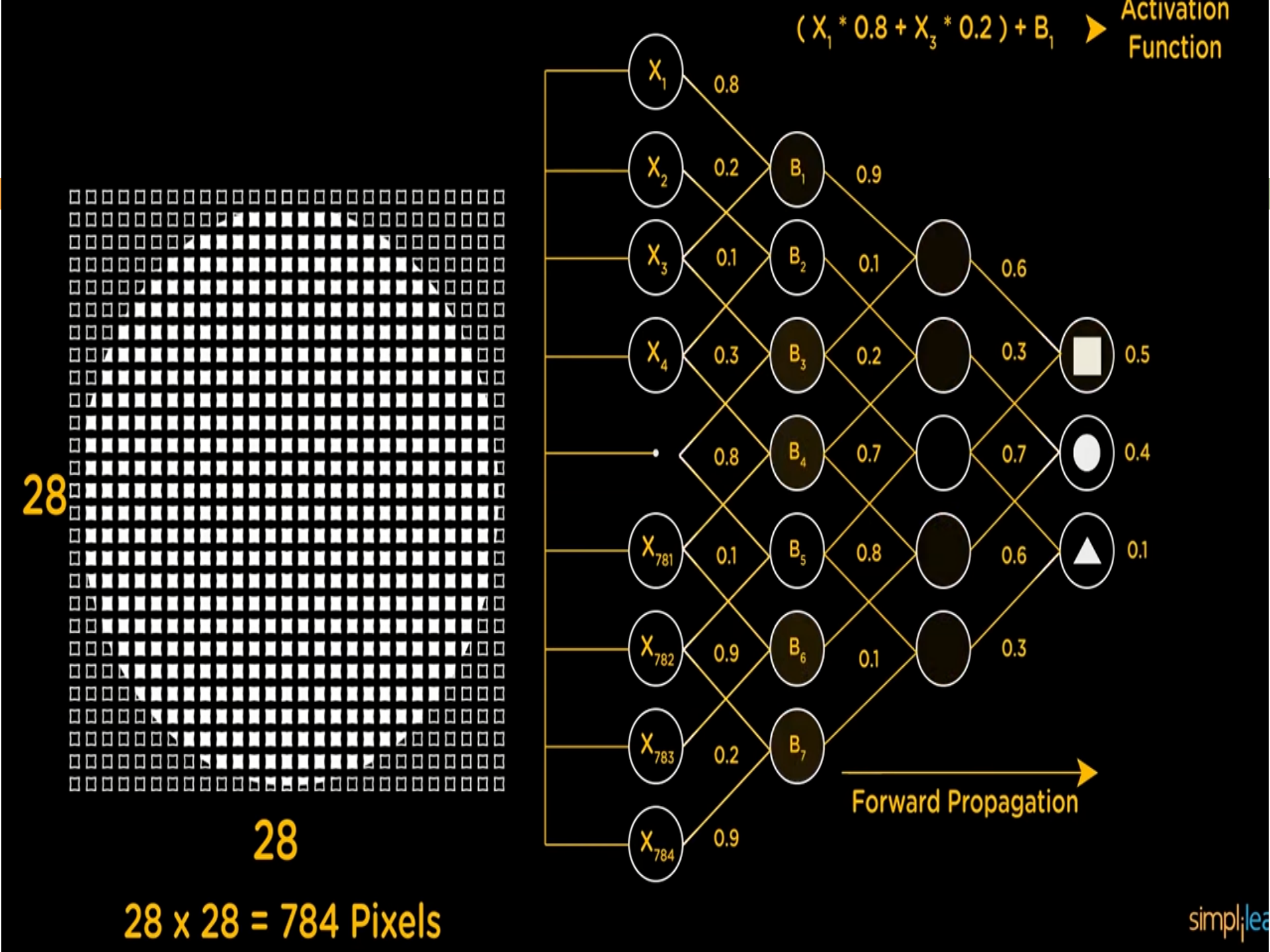
How many hidden layers?

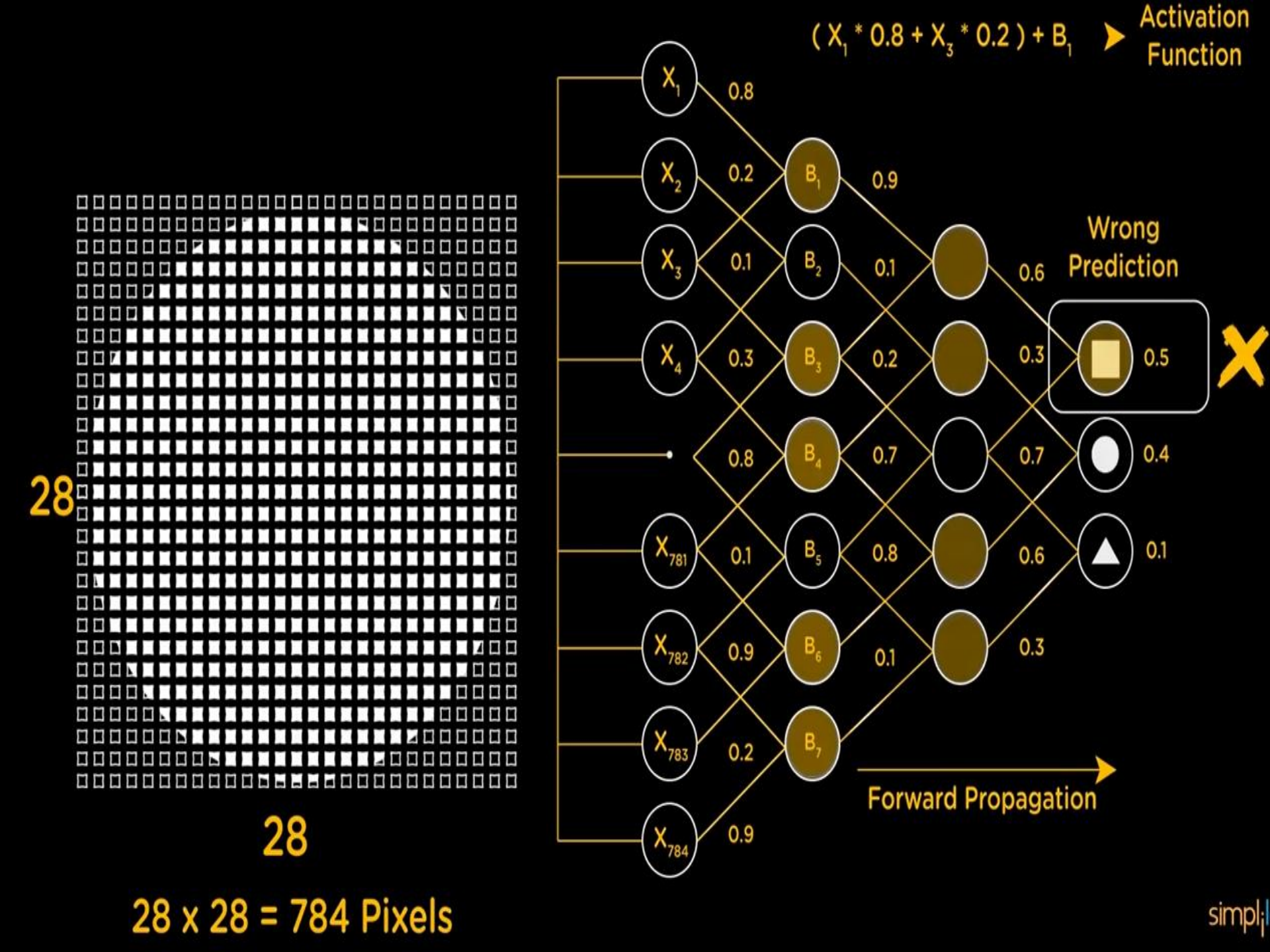
- For any application, number of hidden layers and number of nodes in each hidden layer is not fixed.

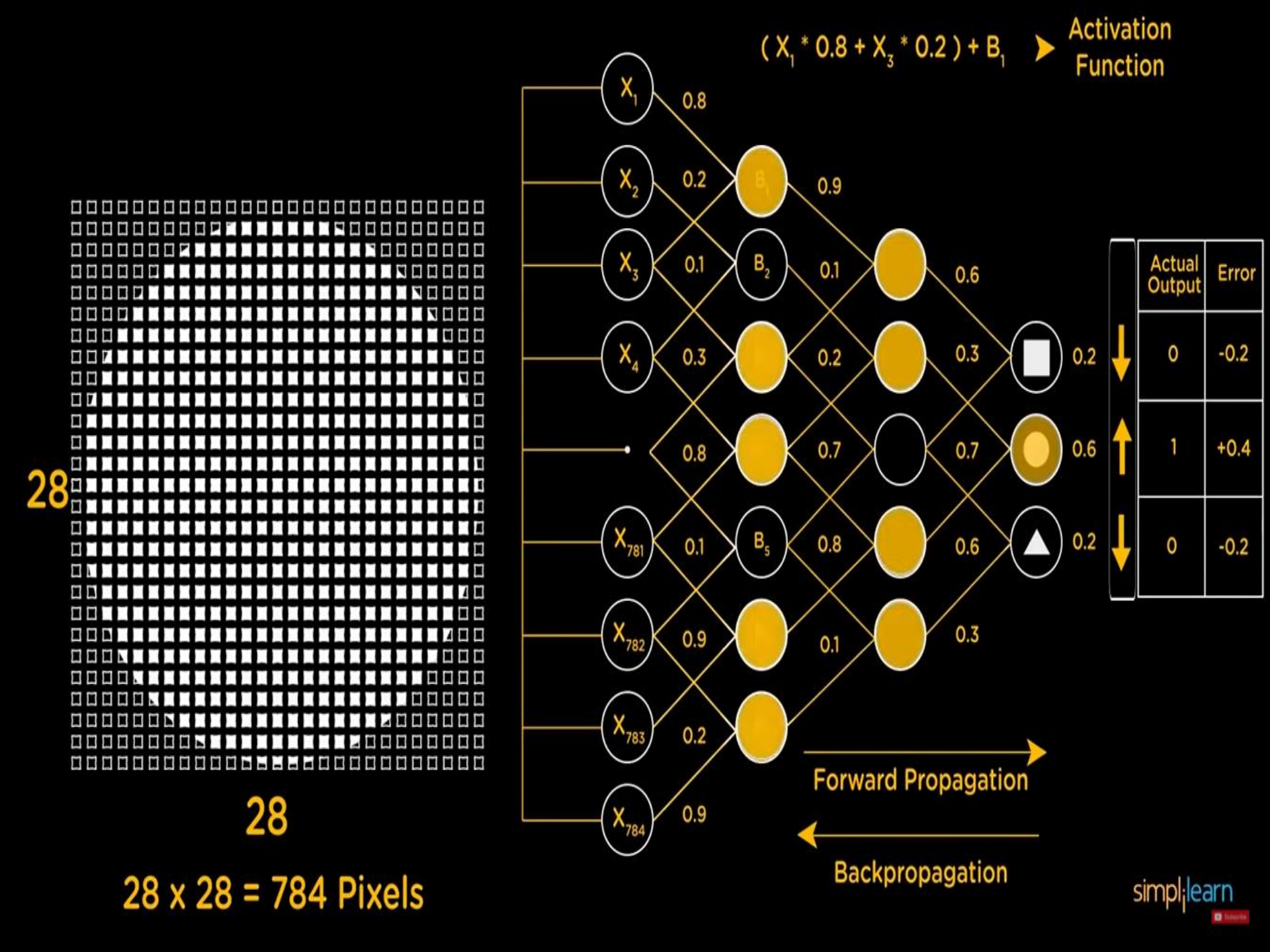
- It will be varied till the output moves towards zero error or till we get a satisfactory output.

Example: Neural Network to find whether the given input is Square or circle or triangle




(X1 * 0.8 + X3 * 0.2) + B1
Activation Function
X1
X2
X3
X4
X781
X782
X783
X784
0.8
0.2
0.1
0.3
0.8
0.1
0.9
0.2
B1
B2
B3
B4
B5
B6
B7
0.9
0.1
0.2
0.7
0.8
0.1
0.6
0.3
0.7
0.6
0.3
0.5
0.4
0.1
Forward Propagation
28
28
28 x 28 = 784 Pixels


(X1 * 0.8 + X3 * 0.2) + B1
Activation Function
X1
X2
X3
X4
X781
X782
X783
X784
0.8
0.2
0.1
0.3
0.8
0.1
0.9
0.2
B1
B2
B3
B4
B5
B6
B7
0.9
0.1
0.2
0.7
0.8
0.1
0.6
0.3
0.7
0.6
0.3
Wrong Prediction
0.5
0.4
0.1
Forward Propagation
28
28
28 x 28 = 784 Pixels
simpli


(X1 * 0.8 + X3 * 0.2) + B1
Activation Function
X1
X2
X3
X4
X781
X782
X783
X784
0.8
0.2
0.1
0.3
0.8
0.1
0.9
0.2
B1
B2
B5
0.9
0.1
0.2
0.7
0.8
0.1
0.6
0.3
0.7
0.6
0.3
0.2
0.6
0.2
Actual Output
Error
0
-0.2
1
+0.4
0
-0.2
28
28
28 x 28 = 784 Pixels
Forward Propagation
Backpropagation
simplilearn

CNN

CNN (Convolutional Neural Network):

They are designed specifically for **computer vision** (they are sometimes applied elsewhere though).

Their name come from **convolutional layers.**

They have been invented to receive and process pixel data.

RNN

RNN (Recurrent Neural Network):

They are the "time series version" of ANNs.

They are meant to process *sequences* of data.

They are at the basis of forecast models and language models.

The most common kind of recurrent layers are called **LSTM** (Long Short Term Memory) and

GRU (Gated Recurrent Units): their cells contain small, in-scale ANNs that choose how much past information they want to let flow through the model. That's how they modeled "memory".

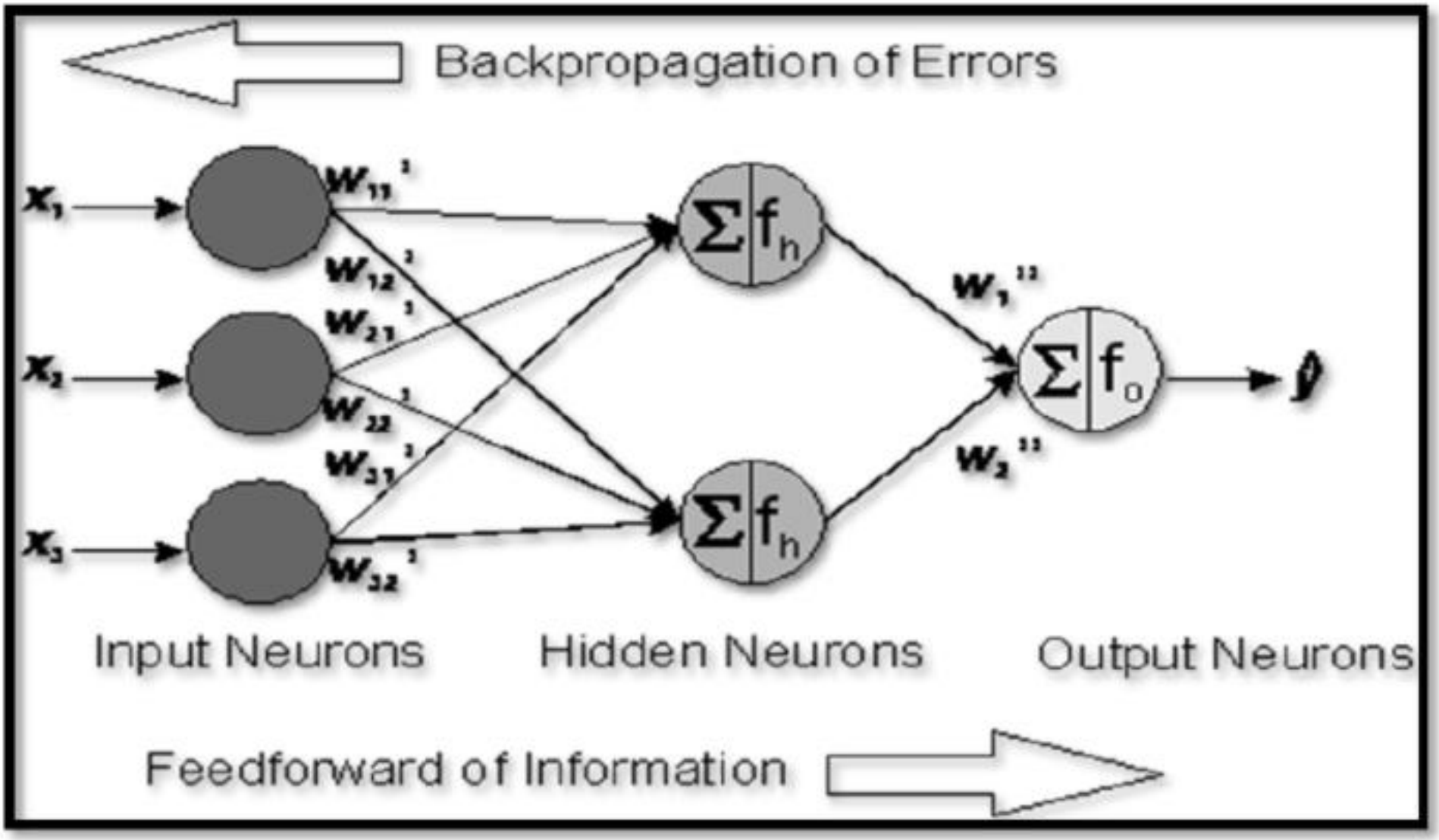
Forward and Backward Propagation

- Forward Propagation is the way to move from the Input layer (left) to the Output layer (right) in the neural network. It is also called as Feed forward.

- The process of moving from the right to left i.e backward from the Output to the Input layer is called the Backward Propagation.

- Backward propagation is required to correct the error or generally it is said to make the system to learn.

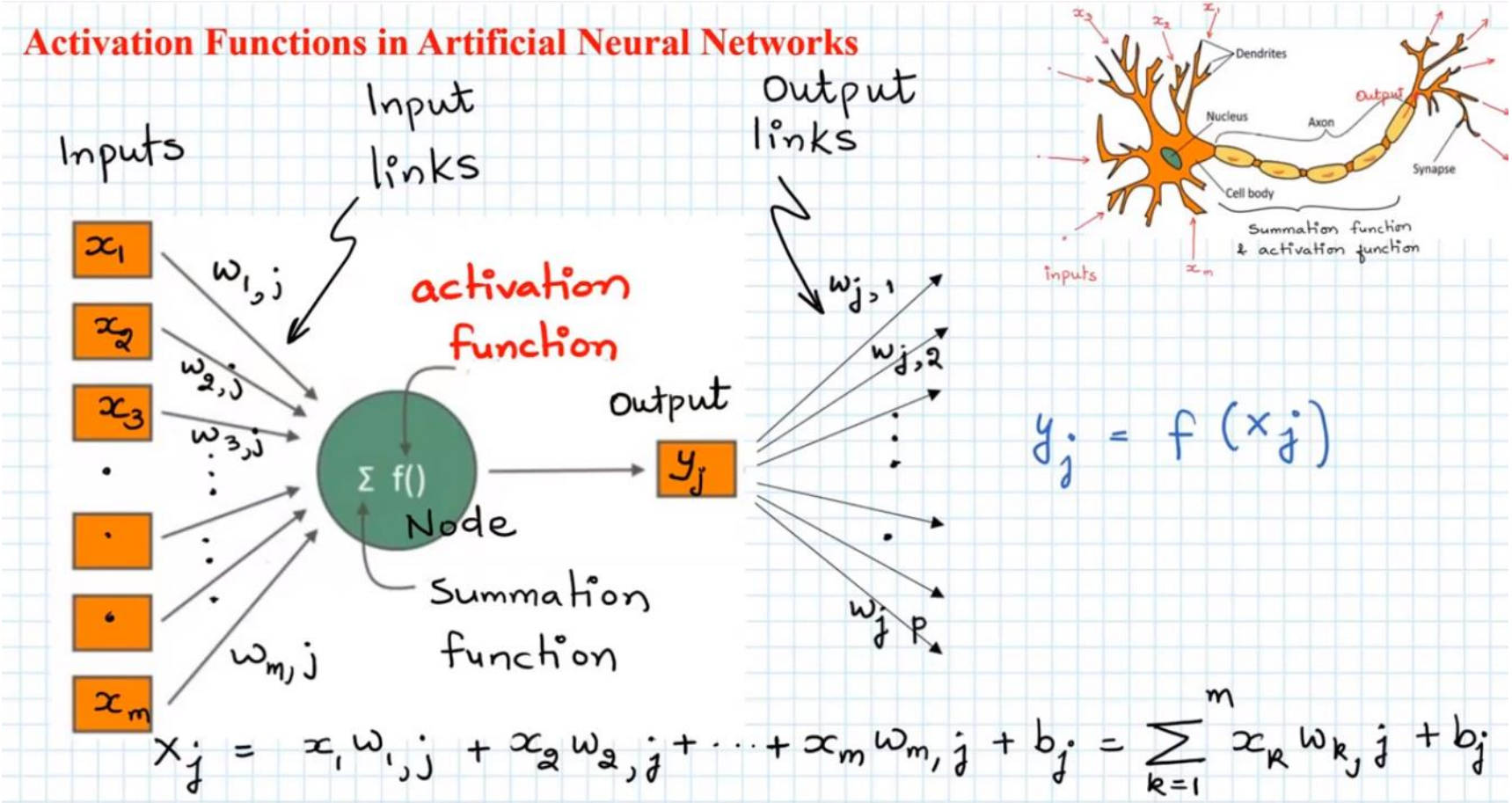
Feed forward and Backward propagation



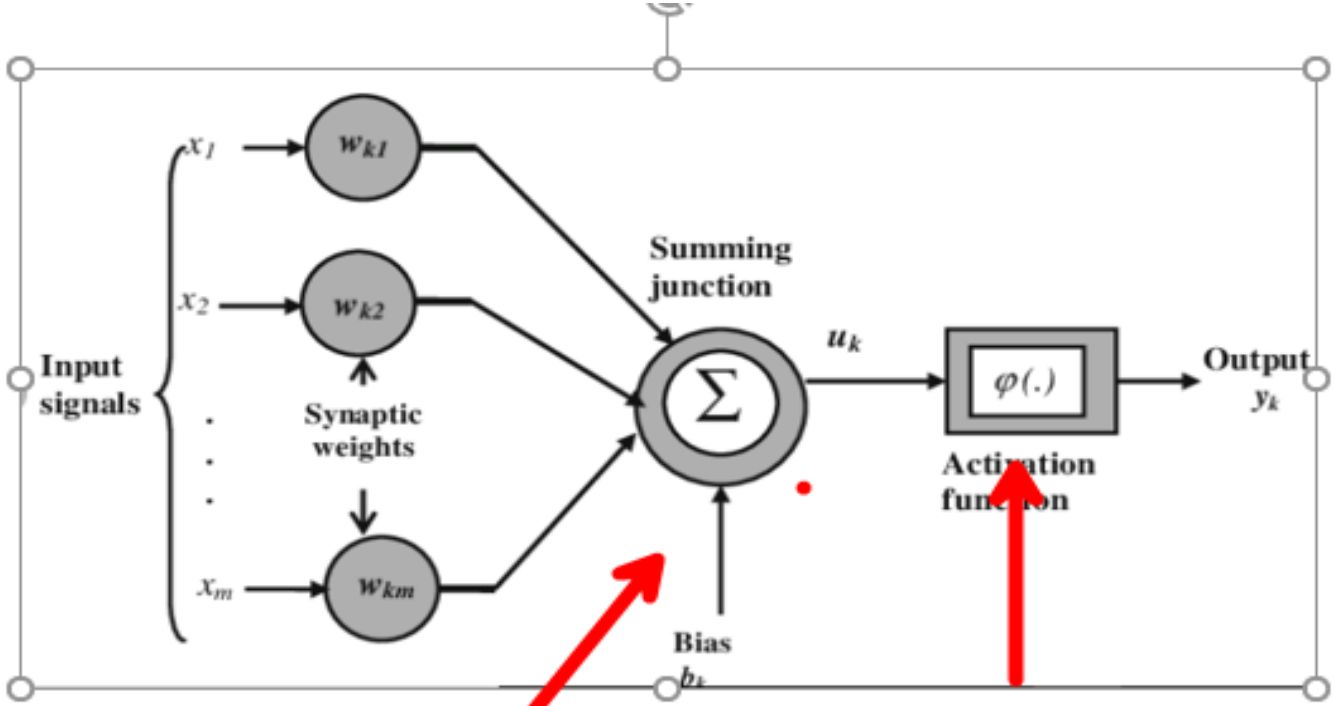
Backward propagation

- Measure the network's output error (difference between actual and obtained)
- Tweak the waits to correct or to reduce the error.
- Move from output layer to input layer one step at a time.
- Compute how much each neuron in the last hidden layer contributed to each output neuron's error.
- Later it moves to the next hidden layer in the reverse direction till the input layer and keeps updating the waits.
- Tweaking the weights to reduce the error is called gradient descent step

Summary… and moving toward activation functions



Linear and Non-linear part of neuron



Need of activation function:

- They are used in the hidden and output layers.
- Activation function is a function that is added into an artificial neural network in order **to help the network learn complex patterns in the data**.
- The activation function will decide what is to be fired to the next neuron

Can ANN work without an activation function?

- Then it becomes linear.
- Basically, the cell body has two parts, one is linear and another one is non-linear. Summation part will do linear activity, and activation function will perform non-linear activity.
- If activation function is not used then, every neuron will only be performing a linear transformation on the inputs using the weights and biases.
- Although linear transformations make the neural network simpler, but this network would be less powerful and will not be able to learn the complex patterns from the data. Hence the need for activation function.

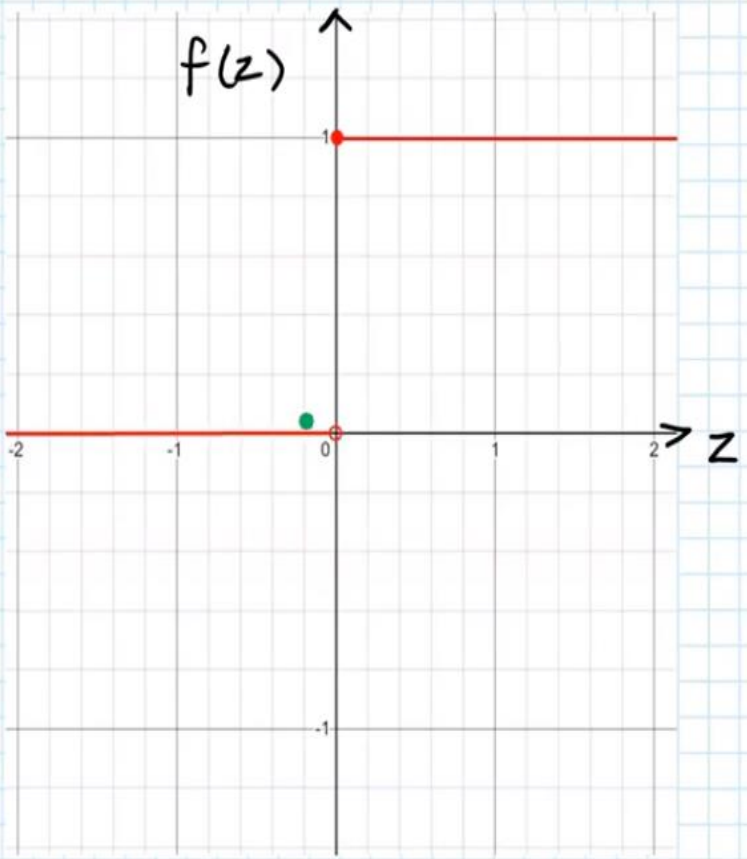
Popular Activation Functions

1. Popular types of activation functions are:
    1. Step function
    2. Sign function
    3. Linear function
    4. ReLU (Rectified Linear Unit): no –ve value
    5. Leaky ReLU
    6. Tanh
    7. Sigmoid
    8. softmax

1. Step Function

1. Step function

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



Range = Possible Output

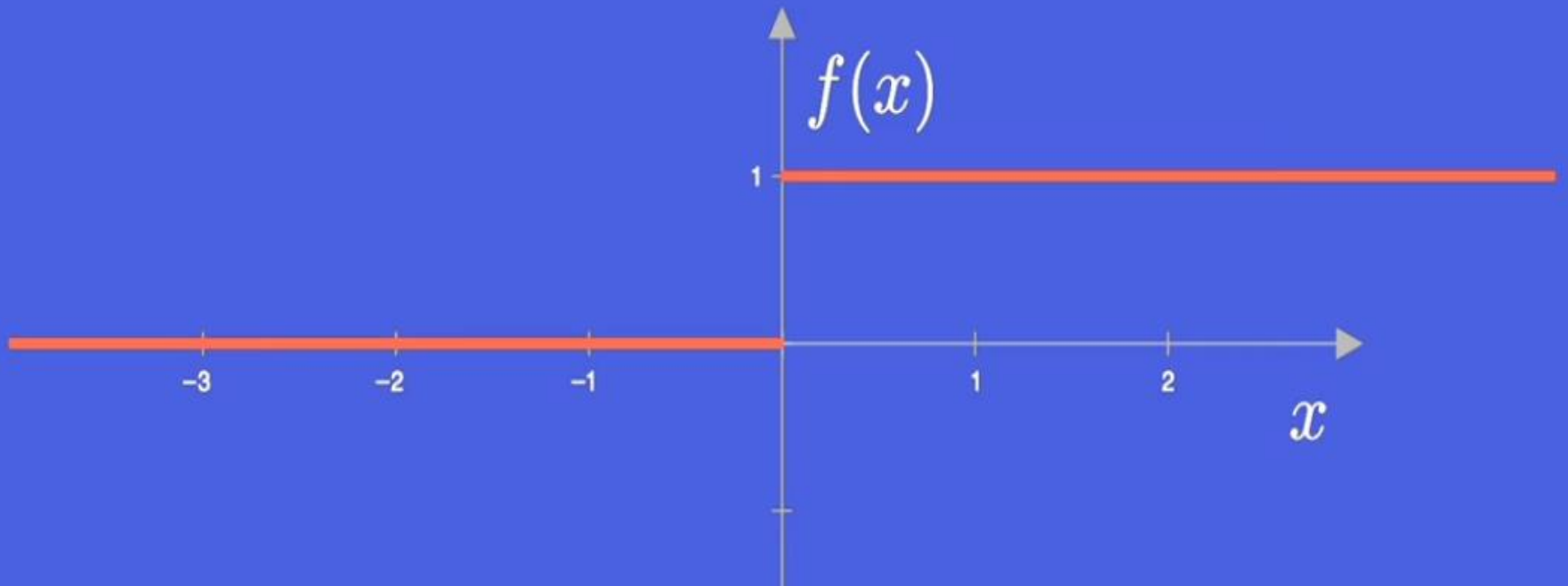
$\{0, 1\}$

Used in: Hidden Layer, Output Layer for Classification

It is simple, but not used regularly

Step Function

$$f(x) = \begin{cases} 1 \ if \ \ x \geq \theta \\ 0 \ otherwise \end{cases}$$



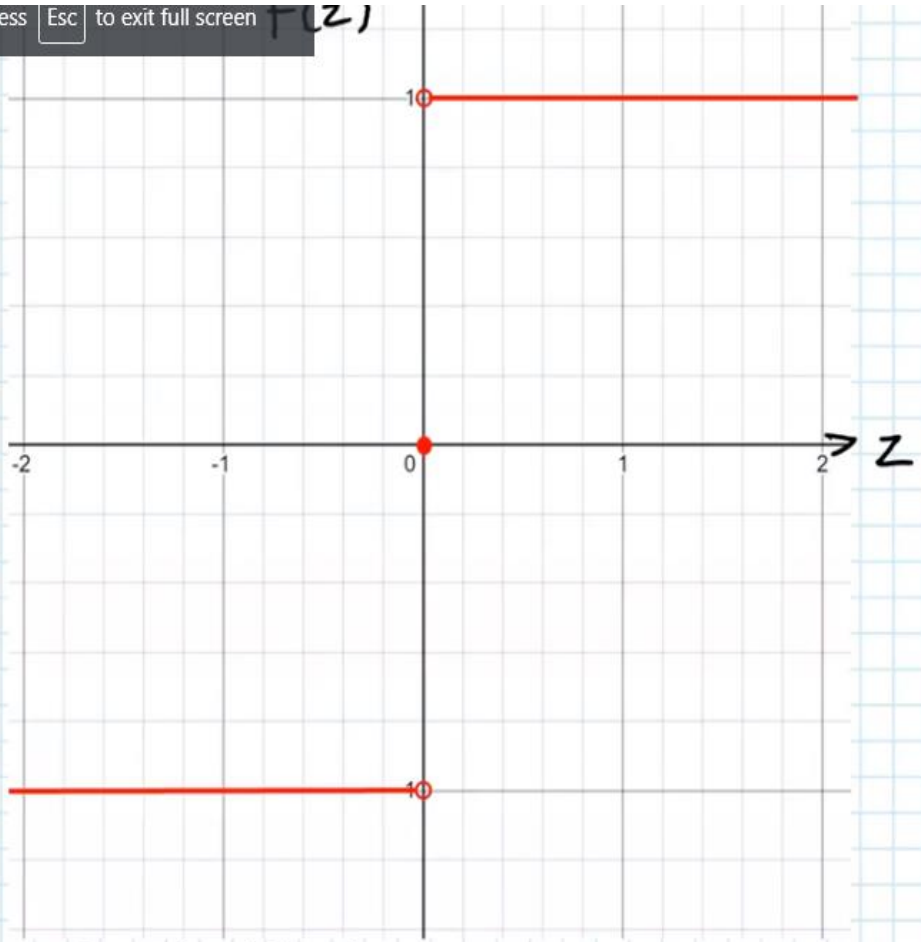
2. Sign function

2. Signum (sgn or sign) function

$$f(z) = \begin{cases} -1 & z < 0 \\ 1 & z > 0 \end{cases}$$

Range = Possible Outputs

$\{-1, 1\}$



Used in: Hidden Layer, Output Layer for Classification

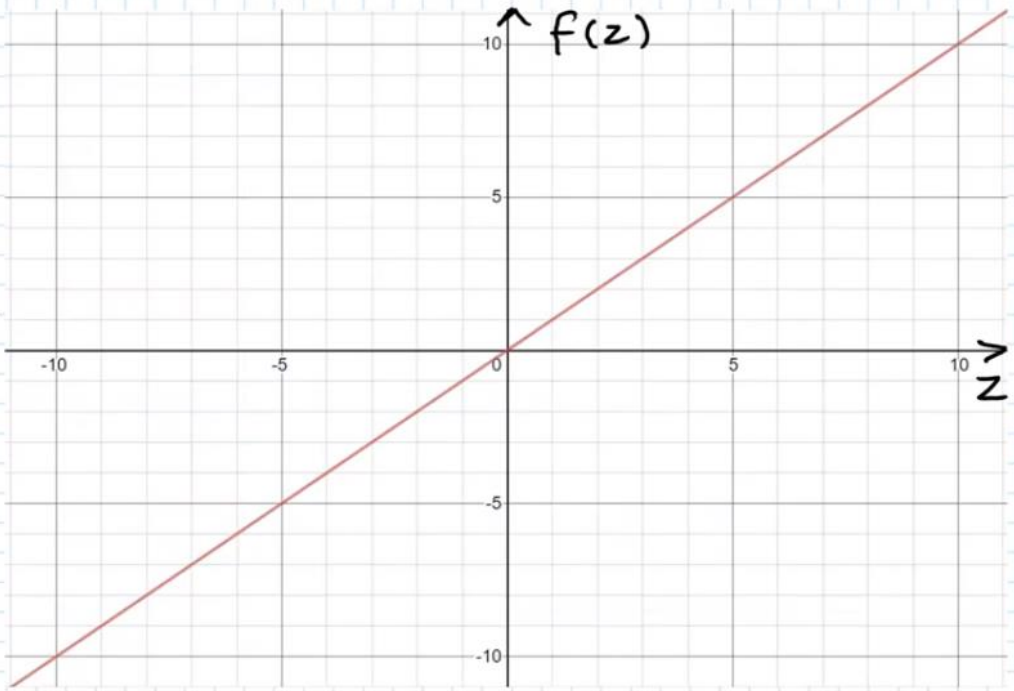
3. Linear function

3. Linear Function

$f(z) = z$

Range = Possible Outputs

$(-\infty, \infty)$



Used in: Hidden Layer, Output Layer for Regression

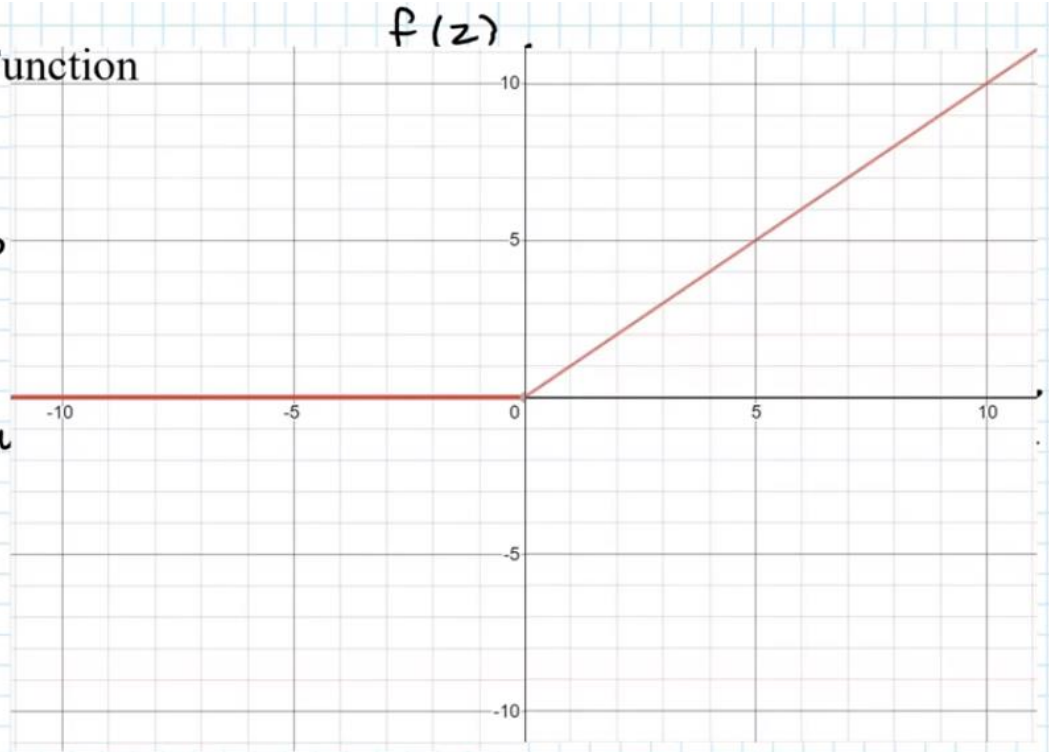
4. ReLU function

4a. Rectified Linear Unit (ReLU) Function

$$f(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$

Range = Possible Outpu

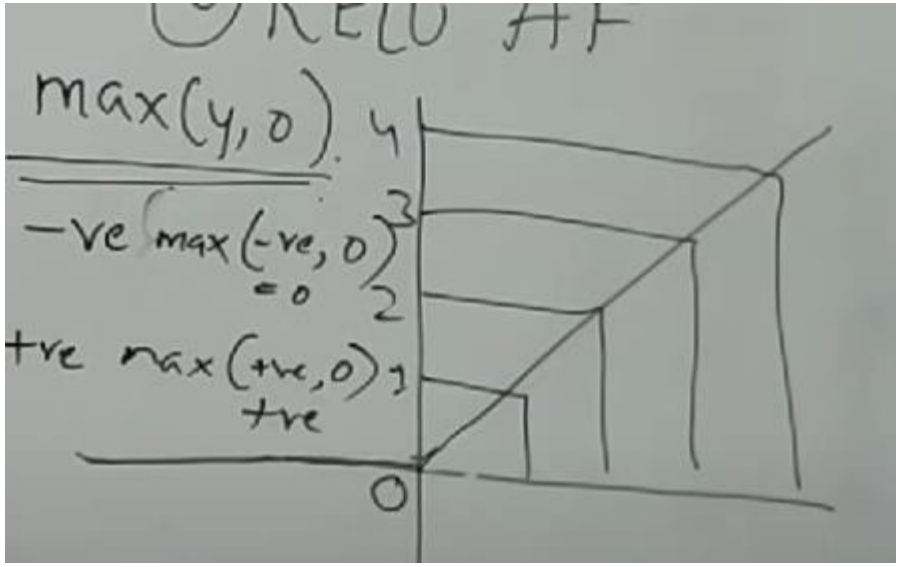
$[0, \infty)$



Used in: Hidden Layer, Output Layer for Regression (only positive output)

ReLU (Rectified Linear Unit)

- It will produce the same output for +ve value, and 0 for all –ve values.



5. Leaky Rectified Linear Unit

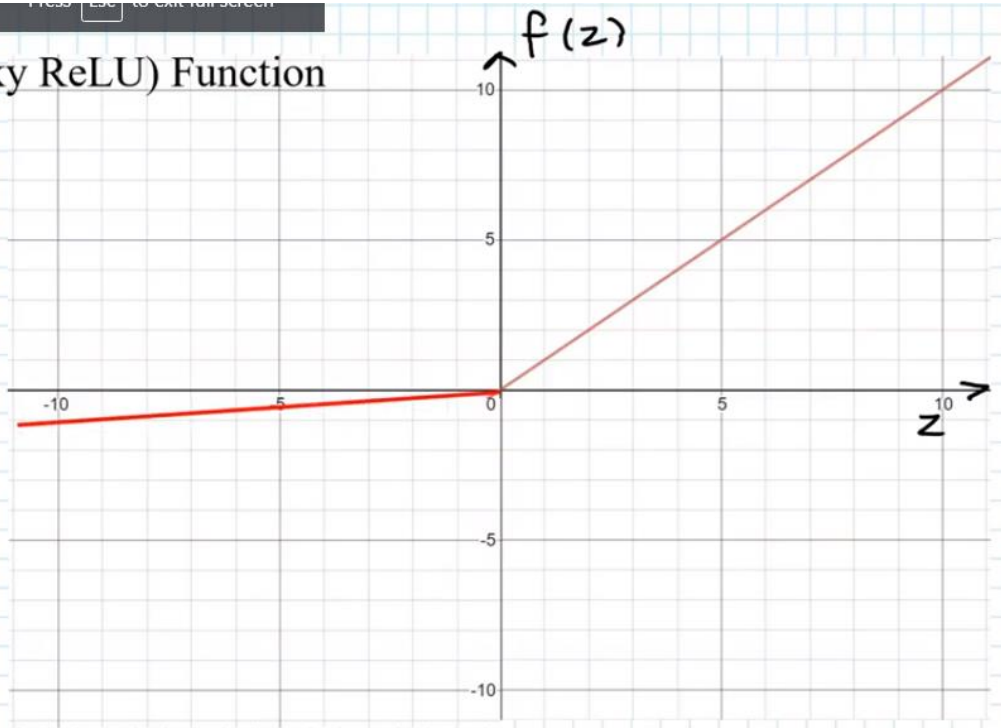
4b. Leaky Rectified Linear Unit (Leaky ReLU) Function

$$f(z) = \begin{cases} az & z < 0 \\ z & z \geq 0 \end{cases}$$

a is a small + number

Range = Possible Outputs

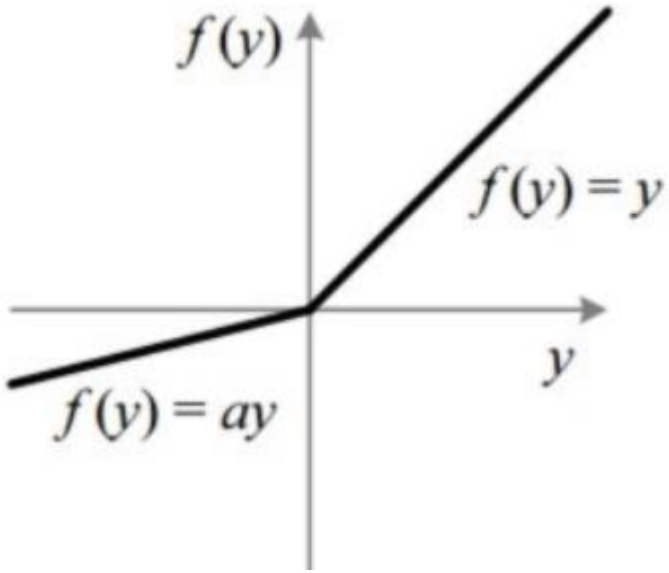
$(-\infty, \infty)$



Used in: Hidden Layer

Leaky ReLU

- Leaky Rectified Linear Unit, or Leaky ReLU, is **a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope**.



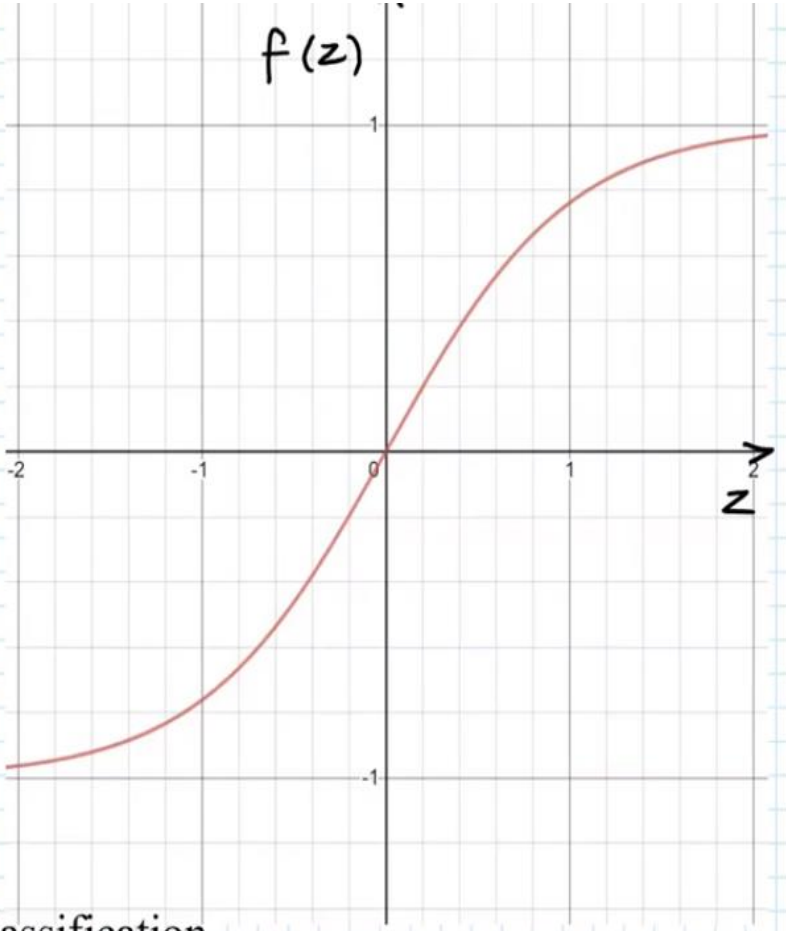
6. Tanh: Hyperbolic Tangent : any value between -1 to +1.

5. Hyperbolic tangent; tanh(z)

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

Range = Possible Outputs

(-1, 1)



Used in: Hidden Layer, Output Layer for Classification

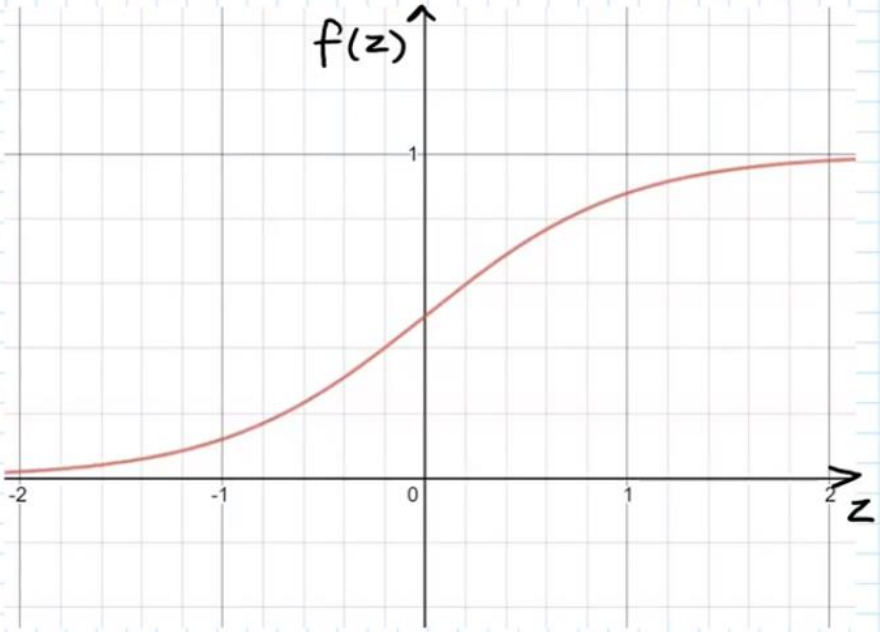
7 Sigmoid Function

6a. Sigmoid Function

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

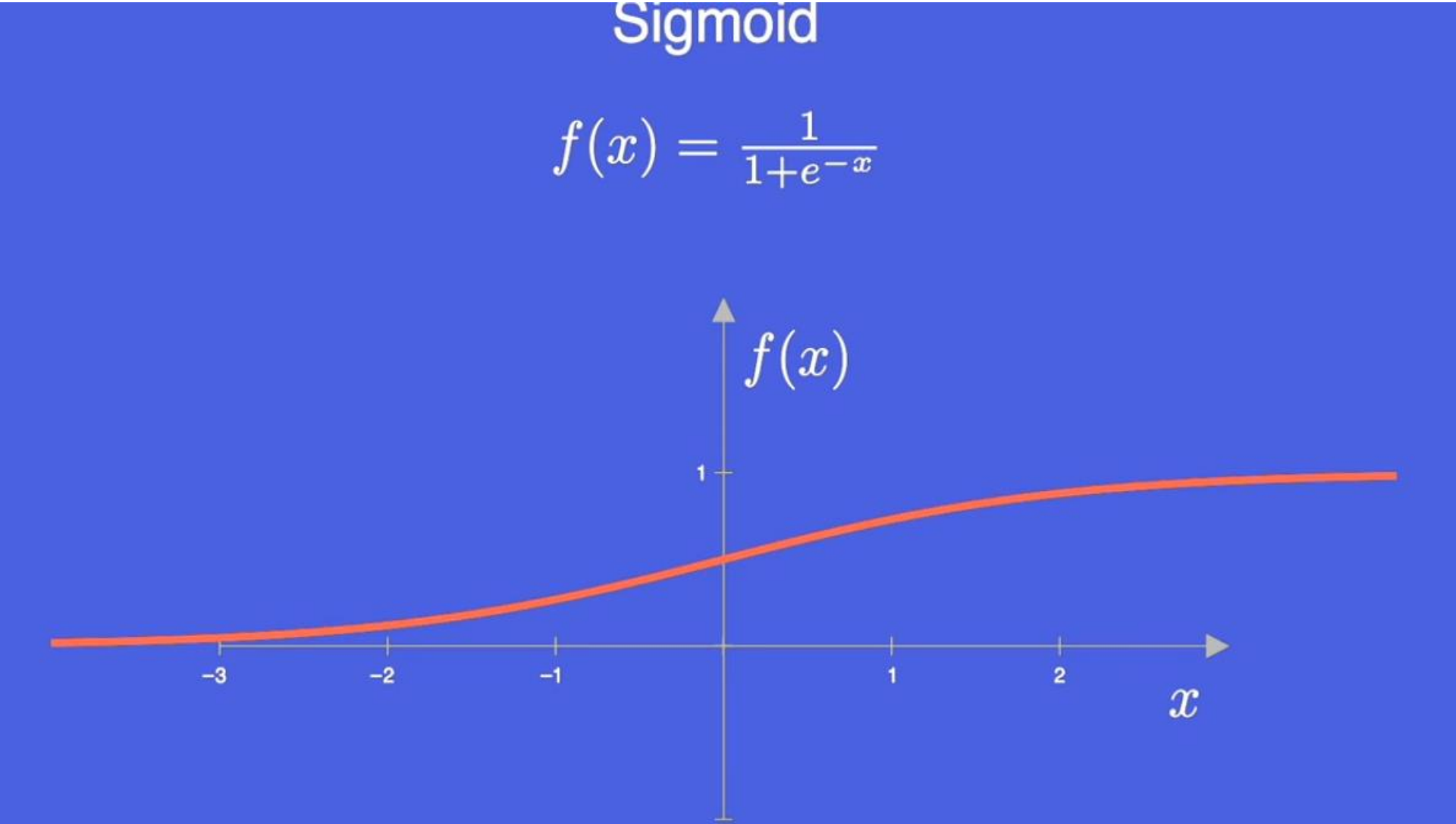
Range = Possible Outputs

(0, 1)



Used in: Hidden Layer, Output Layer for Classification

Popularly used, as x increase e^x becomes very small, and hence tends towards 0



Sigmoid: It is used for classification

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad e = 2.71$$

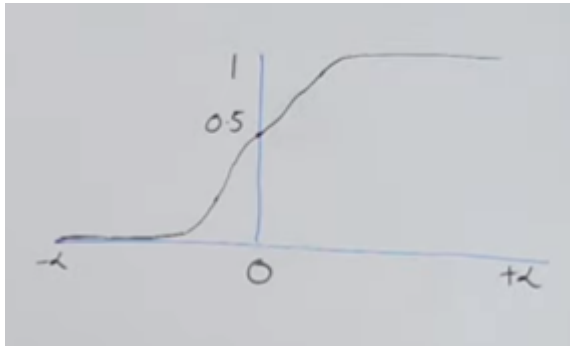
$x = 0 \quad \frac{1}{1+\frac{1}{e^0}} = \frac{1}{2} = 0.5$

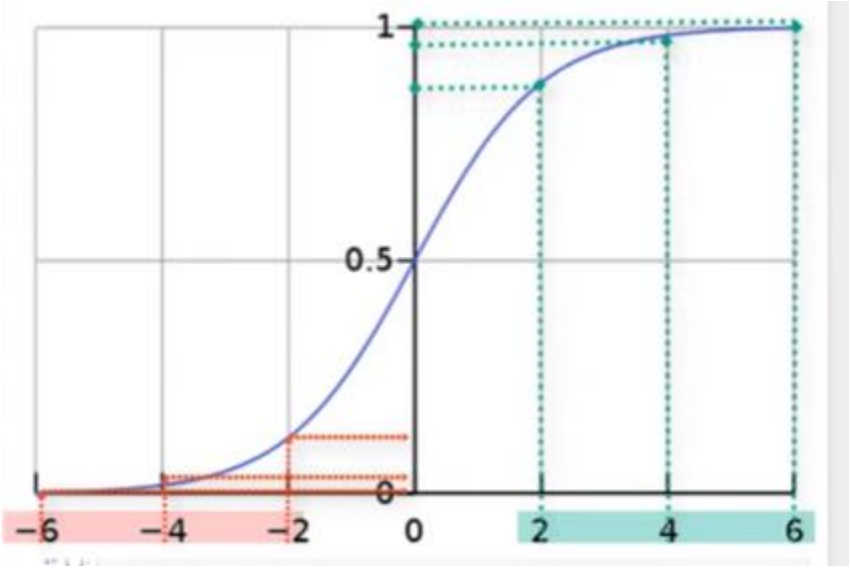
$x = 1 \quad \frac{1}{1+\frac{1}{e^1}} = 0.73$

$x = +\infty \quad \frac{1}{1+\frac{1}{\infty}} = 1$

$x = -1 \quad \frac{1}{1+2.71} = 0.26$

$x = -\infty \quad \frac{1}{1+\infty} = 0$





8. SoftMax function : It is the variant of sigmoid function with multi class classification

6b. Softmax Function

$$z_1, z_2, \cdots, z_n$$

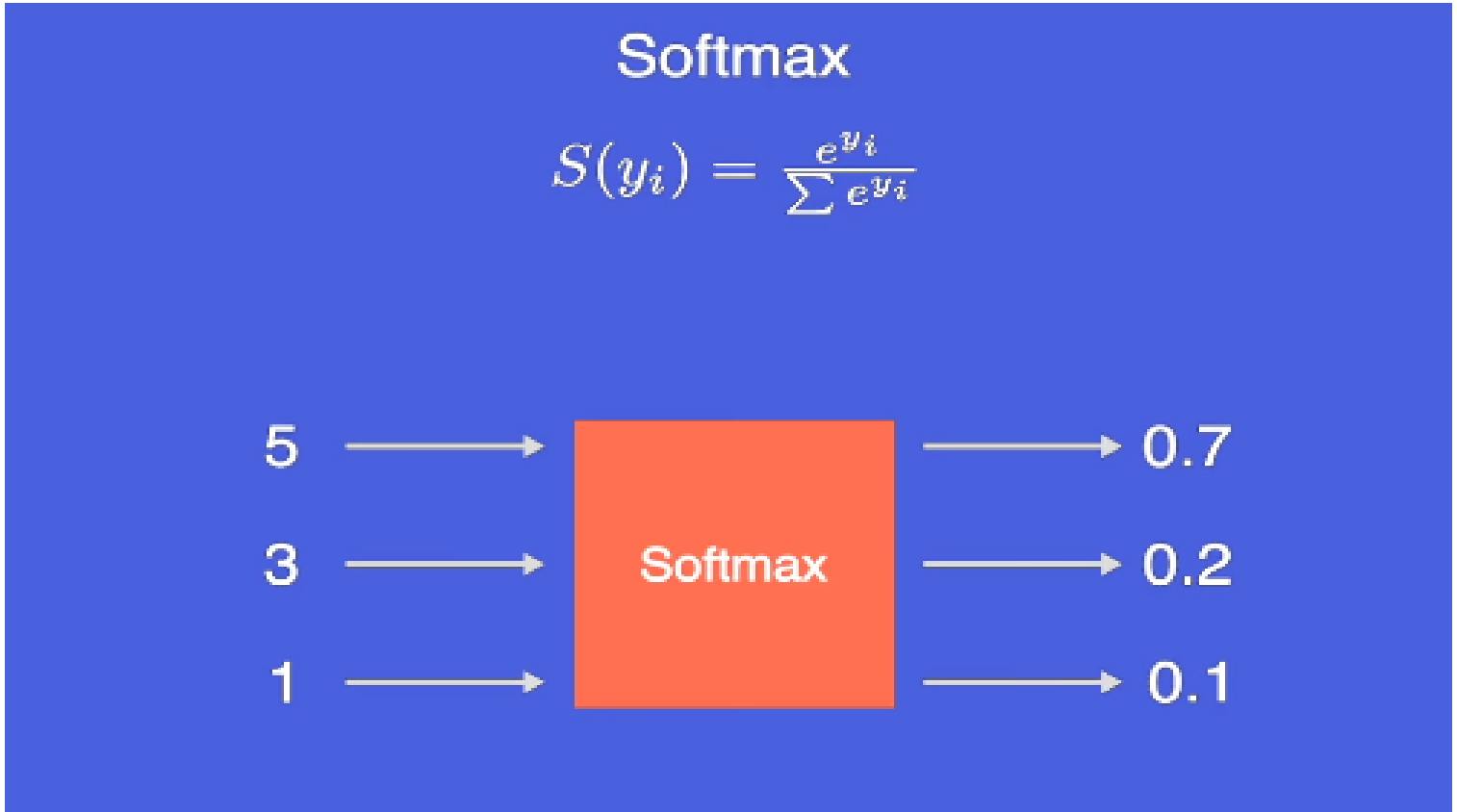
$$f(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_n}}$$

$$f(z_i) = \frac{e^{z_i}}{e^{z_1} + e^{z_2} + \cdots + e^{z_n}} = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

Used in: Output Layer for Multiclass Classification

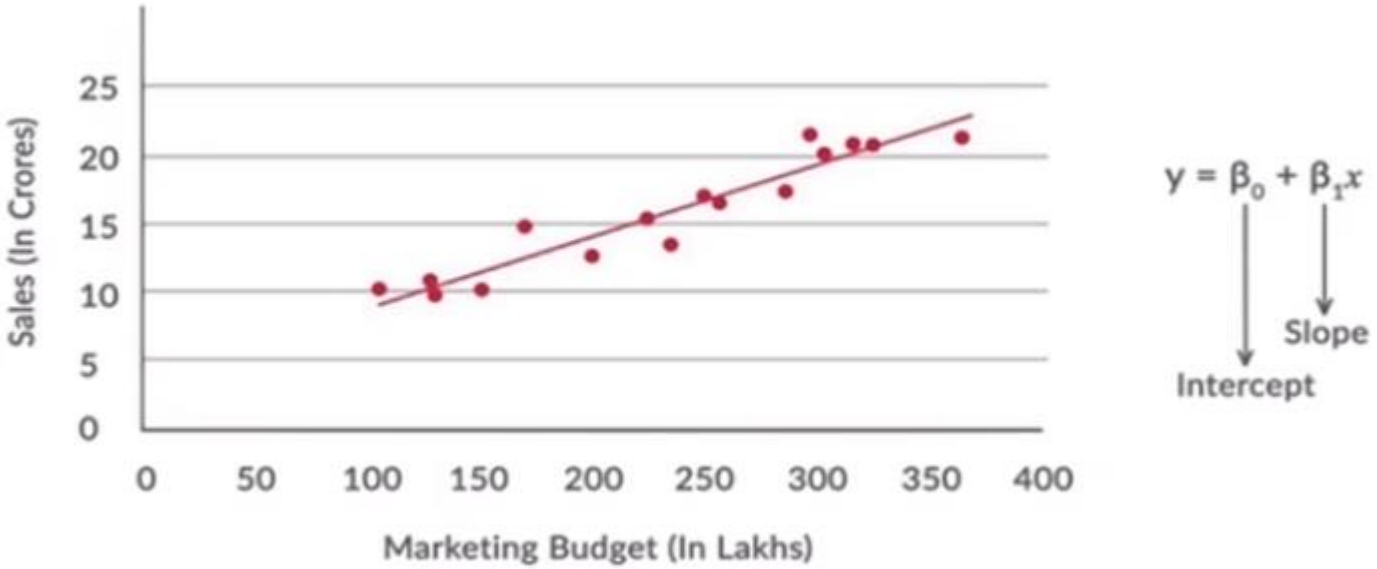
Soft-max function is used popularly in output layer

- Output will be given to soft-max function, which converts the raw input to a value between (0,1).



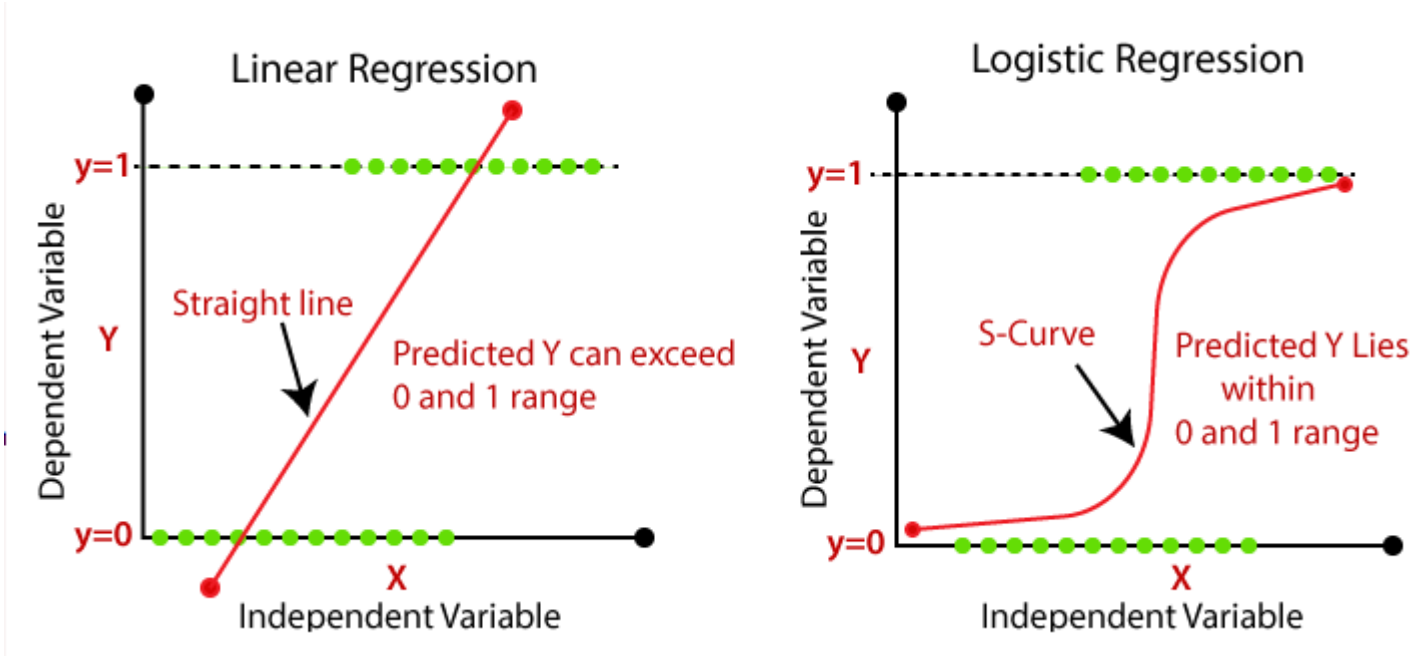
Logistic Regression

- **Linear Regression is used to handle regression problems whereas Logistic regression is used to handle the classification problems**.
- Linear regression provides a continuous output but Logistic regression provides discreet output



- **Y = mx + c** is the equation for a linear line where m is the slope and c is the intercept.

Compare linear vs Logistic regression



Linear Regression

- Linear Regression is one of the most simple Machine learning algorithm that comes under Supervised Learning technique and used for solving regression problems.
- It is used for predicting the continuous dependent variable with the help of independent variables.
- The goal of the Linear regression is to find the best fit line that can accurately predict the output for the continuous dependent variable

Logistic Regression

- Logistic regression is one of the most popular Machine learning algorithm that comes under Supervised Learning techniques.
- It can be used for Classification as well as for Regression problems, but mainly used for Classification problems.
- Logistic regression is used to predict the categorical dependent variable with the help of independent variables.

Training an MLP with TensorFlow's

- Training (60%)
- Validation (20%)
- Testing (20%)

Tensorflow and Scikit-learn (SK learn)

- Scikit-learn (SK Learn) is a general-purpose machine learning library is better for traditional Machine Learning,
- While TensorFlow (tf) is positioned as a deep learning library is better for Deep Learning.
- The obvious and main difference is that TensorFlow does not provide the methods for a powerful feature engineering as sklearn such as dimensional compression, feature selection, etc.

How to work with DL algorithms?

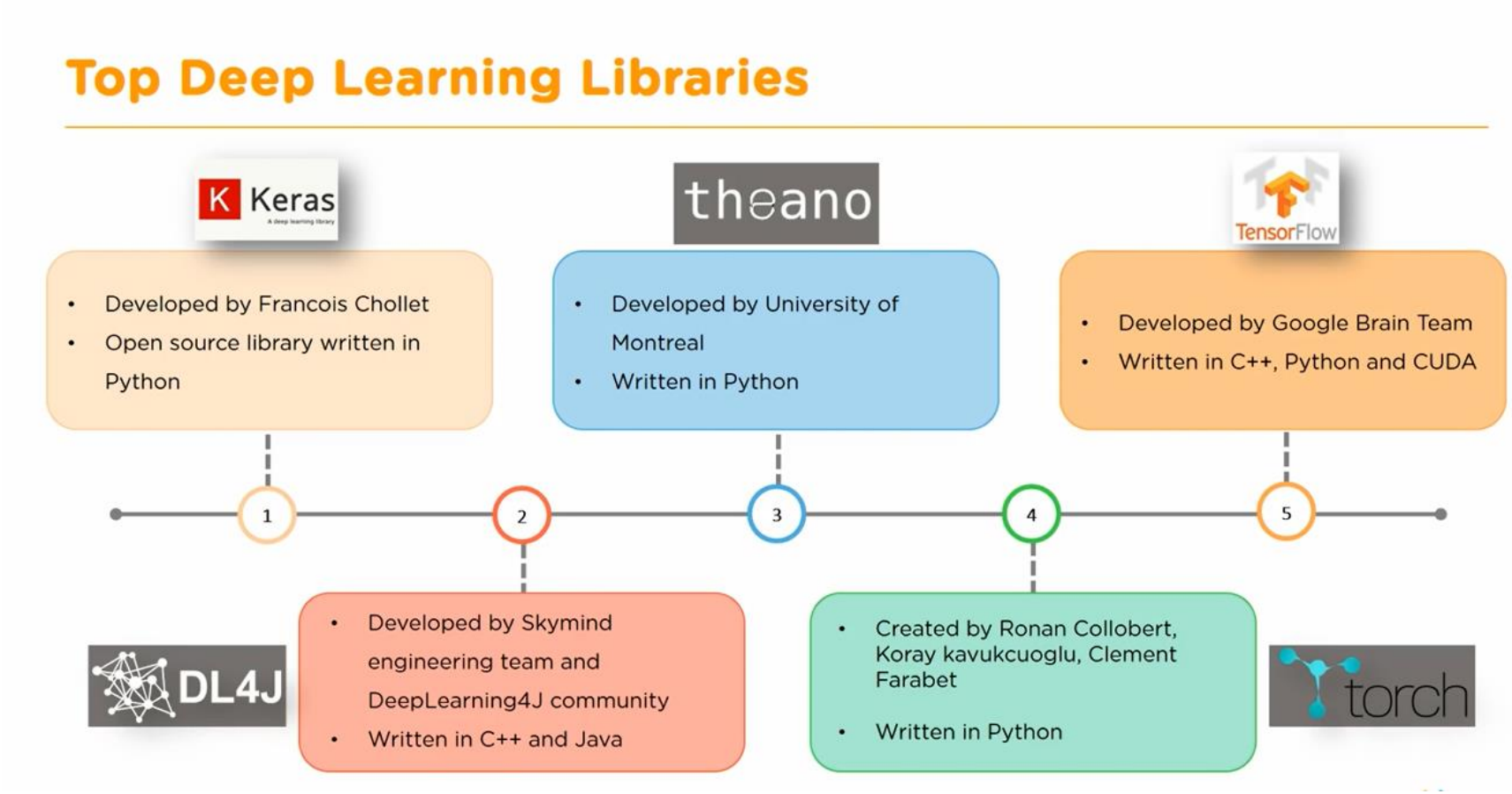
- You need a programming language, preferred is Python.
- Lot of libraries available including Tensorflow.
- Others are Keras, Theano, torch, DL4J
- Tensorflow is from google and Keras is now embedded into Tensorflow.
- Tensorflow also supports traditional ML algorithms also.

Example


```
import tensorflow as tf


model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128),
  tf.keras.layers.ReLU(),
  tf.keras.layers.Dense(10),
  tf.keras.layers.Softmax()
])
```

What is Tensor Flow?

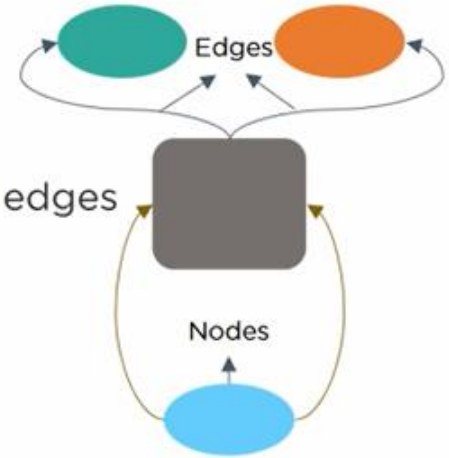


What is Tensorflow?

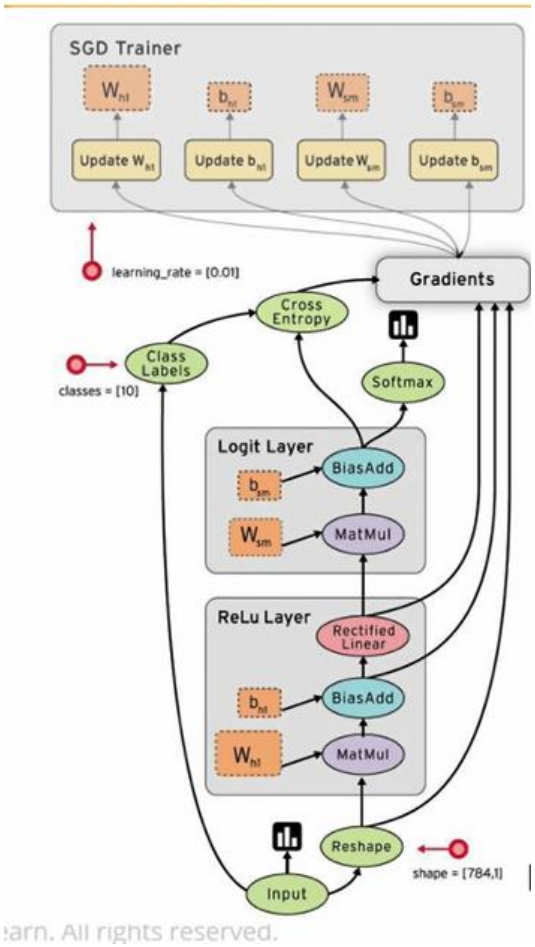
- It is from google
- It was originally developed for large numerical computations.
- Later it was introduced with ML and DL algorithms
- It accepts data in multidimensional array called "Tensor"

Tensorflow works on the basis of Dataflow graphs

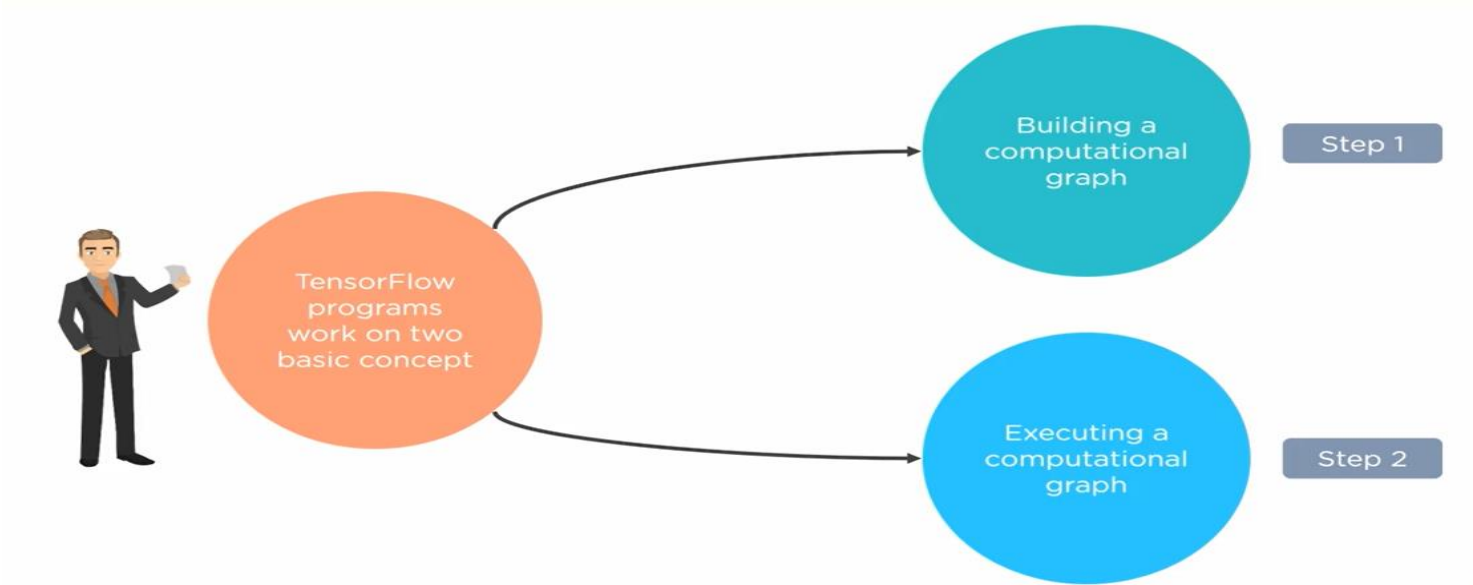
Works on the basis of Data Flow graphs that have nodes and edges



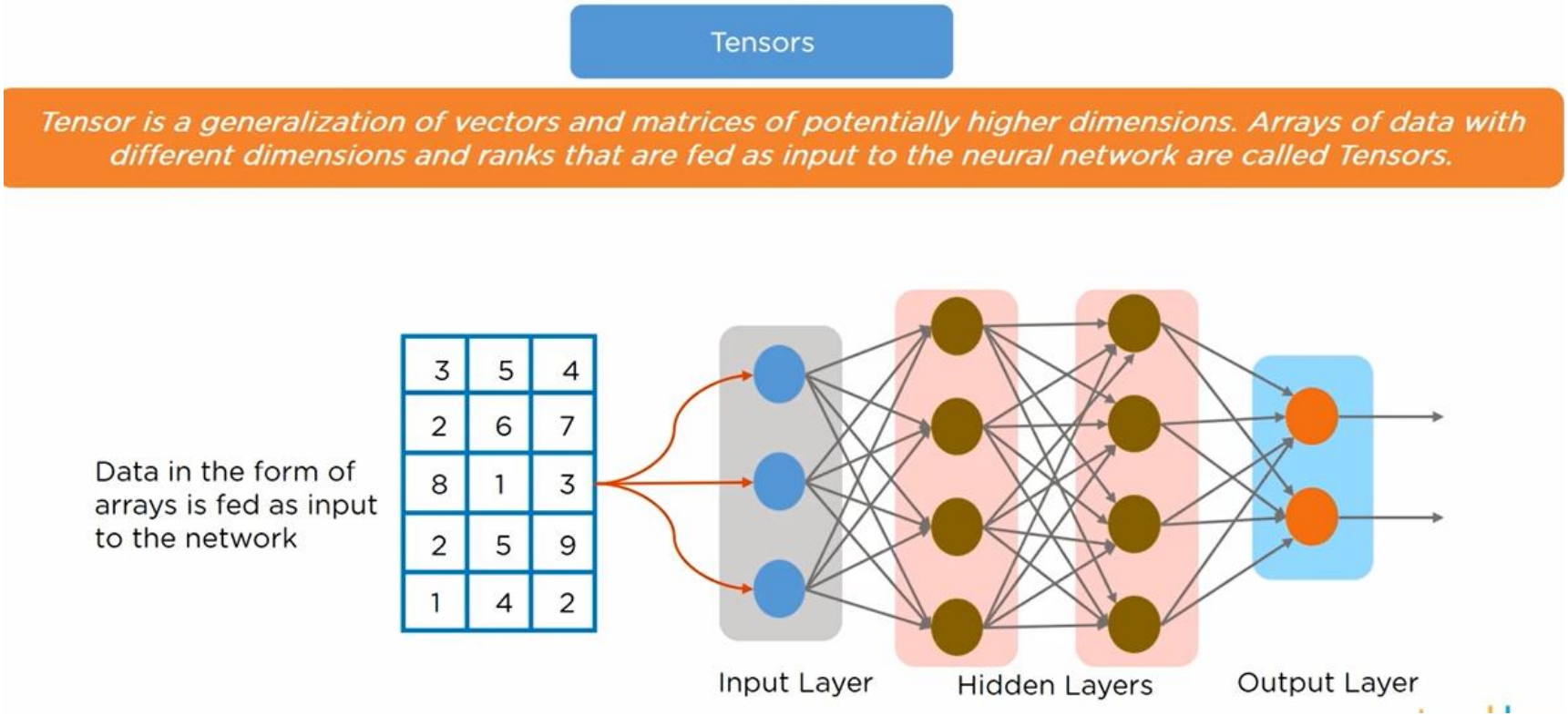
In tensor flow graphs are created and are executed by creating sessions



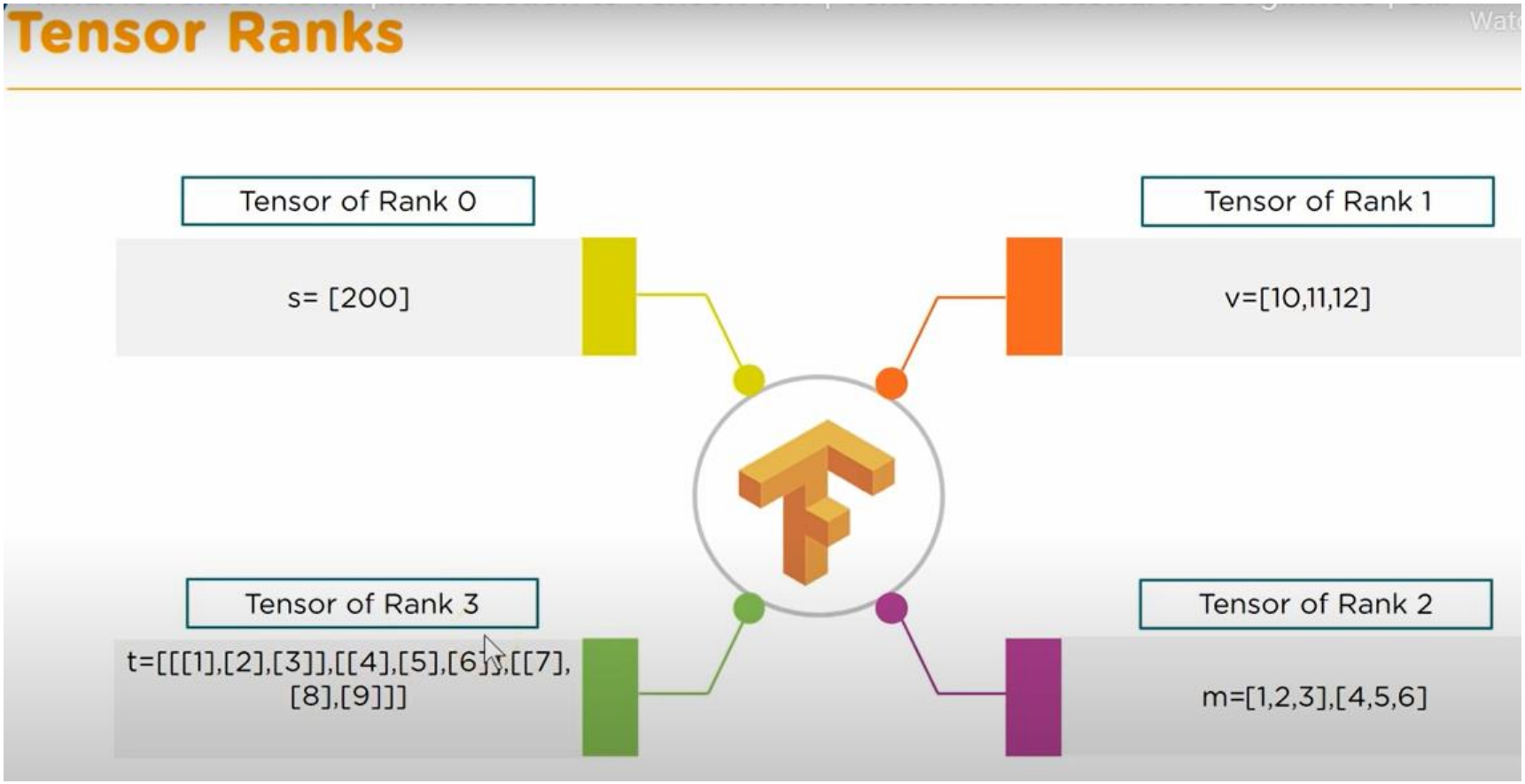
- All the external data at fed into what is know as placeholder, constants and variables.
- To summarize, Tensorflow starts building a computational graph and in the next step it executing the computational graph.



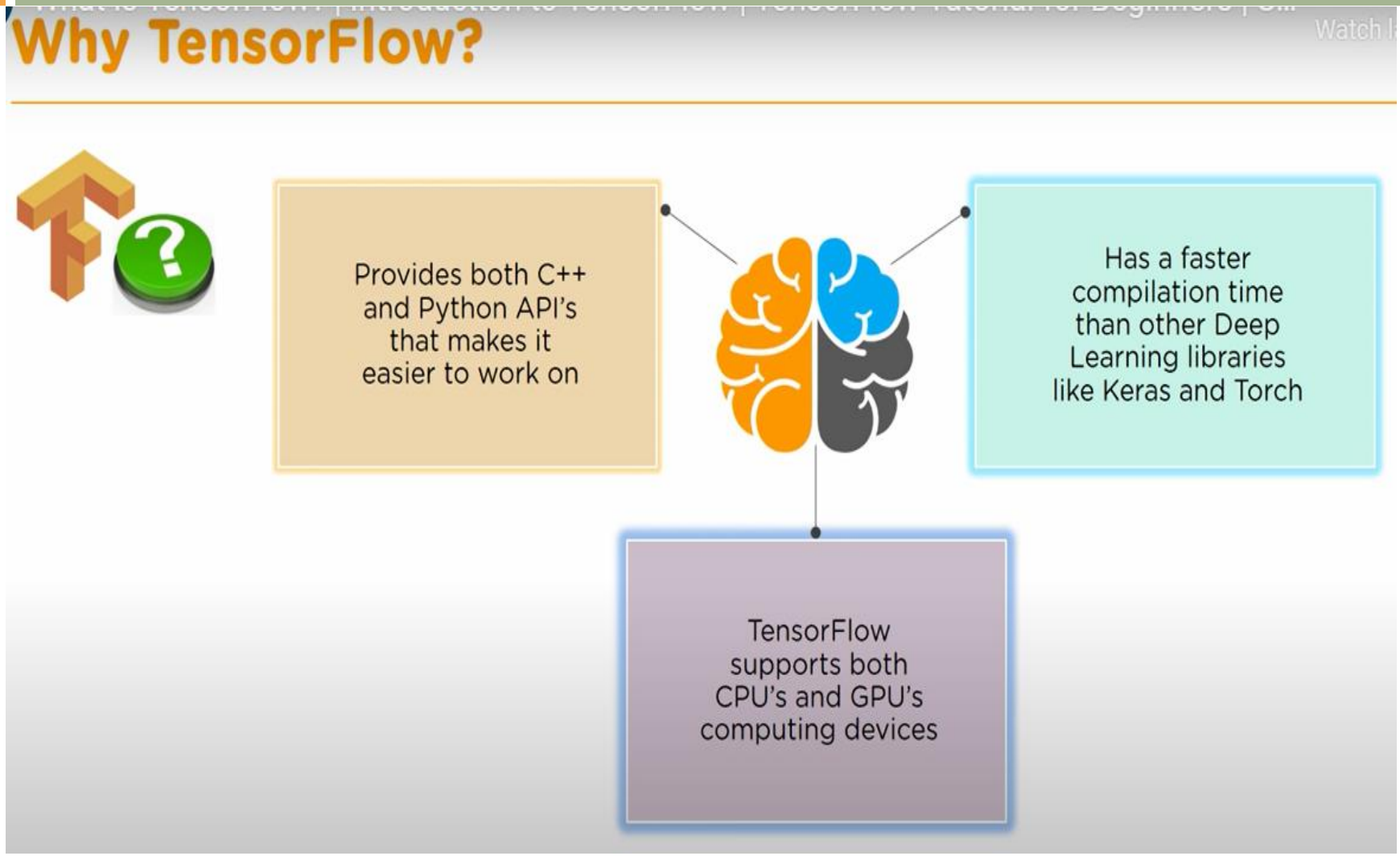
Tensors


Tensors
Tensor is a generalization of vectors and matrices of potentially higher dimensions. Arrays of data with different dimensions and ranks that are fed as input to the neural network are called Tensors.
Data in the form of arrays is fed as input to the network
3 5 4
2 6 7
8 1 3
2 5 9
1 4 2
Input Layer
Hidden Layers
Output Layer

Ranks (dimensions) of tensor



Why to use TensorFlow?



Components of Tensorflow: Constants

- Programming using Tensorflow is bit different from programming on SK Learn and also on python.
- In Tensorflow the storage consists of
  - Constants
  - Variables
  - Placeholders

Constants are parameters whose value does not change. To define a constant we use *tf.constant()* command.

Example:

```
a = tf.constant(2.0, tf.float32)
b = tf.constant(3.0)
Print(a, b)
```

Variables

- In variables, V must be in capital letters.
- Value of the variable can be changed.. But not of constant.

Variables allow us to add new trainable parameters to graph. To define a variable we use *tf.Variable()* command and initialize them before running the graph in a session.

Example:

```
W = tf.Variable([.3],dtype=tf.float32)
b = tf.Variable([-.3],dtype=tf.float32)
x = tf.placeholder(tf.float32)
linear_model = W*x+b
```

Placeholder

- They are used to feed the data from outside.
- Say from a file, from image file, CSV file and so on.
- Feed_dict id popularly used to feed the data to the placeholder.

Placeholders allow us to feed data to a tensorflow model from outside a model. It permits a value to be assigned later. To define a placeholder we use *tf.placeholder()* command.

Example:

```
a = tf.placeholder(tf.float32)
b = a*2
with tf.Session() as sess:
result = sess.run(b,feed_dict={a:3.0})

print result
```

- Constants, Variable and placeholder…
- Create Graph using the above, then you will have session and session object and run it.
- Every computation you perform is a node in a graph.
- Initially tf object is created, which is the default graph, which will not have any constant, variable …

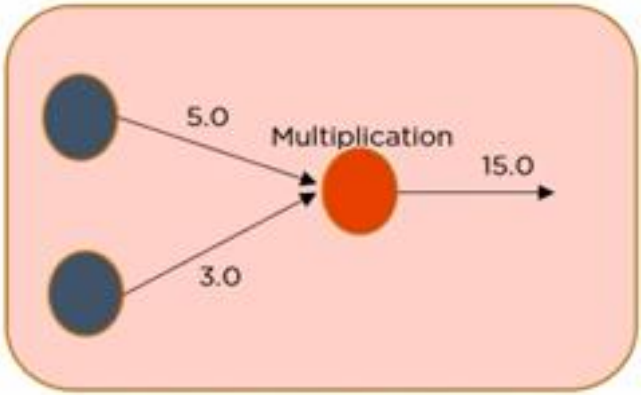
Running a session in Tensorflow

- Multiplication of ‘a’ and ‘b’ are done while running the session (last statement)

A *session* is run to evaluate the nodes. This is called as the *TensorFlow runtime*.

Example:

```
a = tf.constant(5.0)
b = tf.constant(3.0)
c = a*b
# Launch Session
sess = tf.Session()
# Evaluate the tensor c
print(sess.run(c))
```



Running a Computation Graph

Tensor flow – where to execute?

- **TensorFlow is already pre-installed**
- When you create a new notebook on colab.research.google.com, TensorFlow is already pre-installed and optimized for the hardware being used. Just import tensorflow as tf , and start coding.

Tensor flow can also be executed in Jupyter Notebook

- Inside the notebook, you can **import TensorFlow in Jupyter Notebook with the tf alias**. Click to run.

Training an MLP with TensorFlow's

- The simplest way to train an MLP with TensorFlow is to use the high-level API TF.Learn.
- The DNNClassifier class makes it trivial to train a deep neural network with any number of hidden layers, and a softmax output layer to output estimated class probabilities.
- For example, the following code trains a DNN for classification with two hidden layers (one with 300 neurons, and the other with 100 neurons) and a softmax output layer with 10 neurons.

Piece of code of for training MLP


```
import tensorflow as tf

feature_columns = tf.contrib.learn.infer_real_valued_columns_from_input(X_train)
dnn_clf = tf.contrib.learn.DNNClassifier(hidden_units=[300, 100], n_classes=10,
                                         feature_columns=feature_columns)
dnn_clf.fit(x=X_train, y=y_train, batch_size=50, steps=40000)
```

tf is tensorflow

Code creates set of real valued columns from the training set.
Then create the DNNClassifier, with two hidden layers of 300 and 100 neurons and with output layer of 10 neurons.
Finally program is run for 40,000 epochs in a batch of 50.

Fine tuning NN Hyper Parameters - Up and Running with TensorFlow

- In a simple MLP you can change the number of layers, number of neurons per layer, the type of activation function and also the weight initialization logic.
- The above are the Hyper Parameters to be fine tuned in a Neural Network.

Number of Hidden Layers

- For many problems, you can just begin with a single hidden layer and you will get reasonable results.
- It has actually been shown that an MLP with just one hidden layer can model even the most complex functions provided it has enough neurons.
- For a long time, these facts convinced researchers that there was no need to investigate any deeper neural networks.
- But they overlooked the fact that deep networks have a much higher parameter efficiency than shallow ones.
- They can model complex functions using exponentially fewer neurons than shallow nets, making them much faster to train

Number of hidden layers..cont

- Very complex tasks, such as large image classification or speech recognition, typically require networks with dozens of layers (or even hundreds) and they need a huge amount of training data.
- However, you will rarely have to train such networks from scratch: it is much more common to reuse parts of a pretrained state-of-the-art network that performs a similar task. Training will be a lot faster and require much less data

Number of Neurons per Hidden Layers

- Usually the number of neurons in the input and output layers is determined by the type of input and output your task required.
- For the hidden layer the common practice is to size them to form a funnel, with fewer and fewer neurons at each layer.
- For example a typical neural network may have two hidden layers, the first with 300 neurons and the second with 100.
- However, this practice is not as common now, and you may simply use the same size for all hidden layers; for example all hidden layers with 150 neurons.
- Neurons can be gradually increased until the network starts overfitting.

Activation Functions

- In most of the cases you can use the ReLU activation function in the hidden layers. It is a bit faster to compute than other activation functions.
- For the output layer, the softmax activation function is generally a good choice for classification tasks.

End of Unit - 1