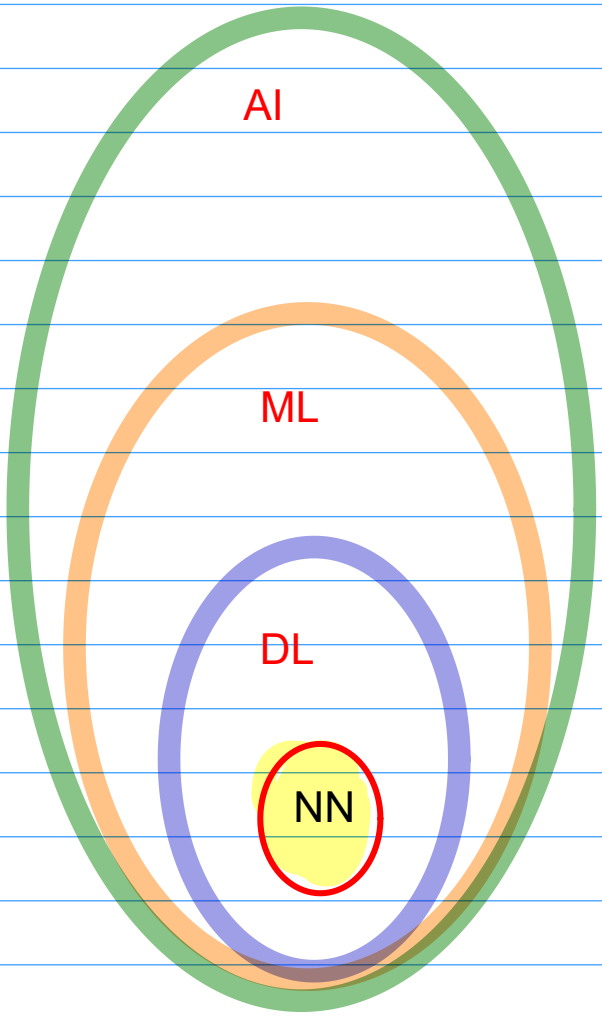
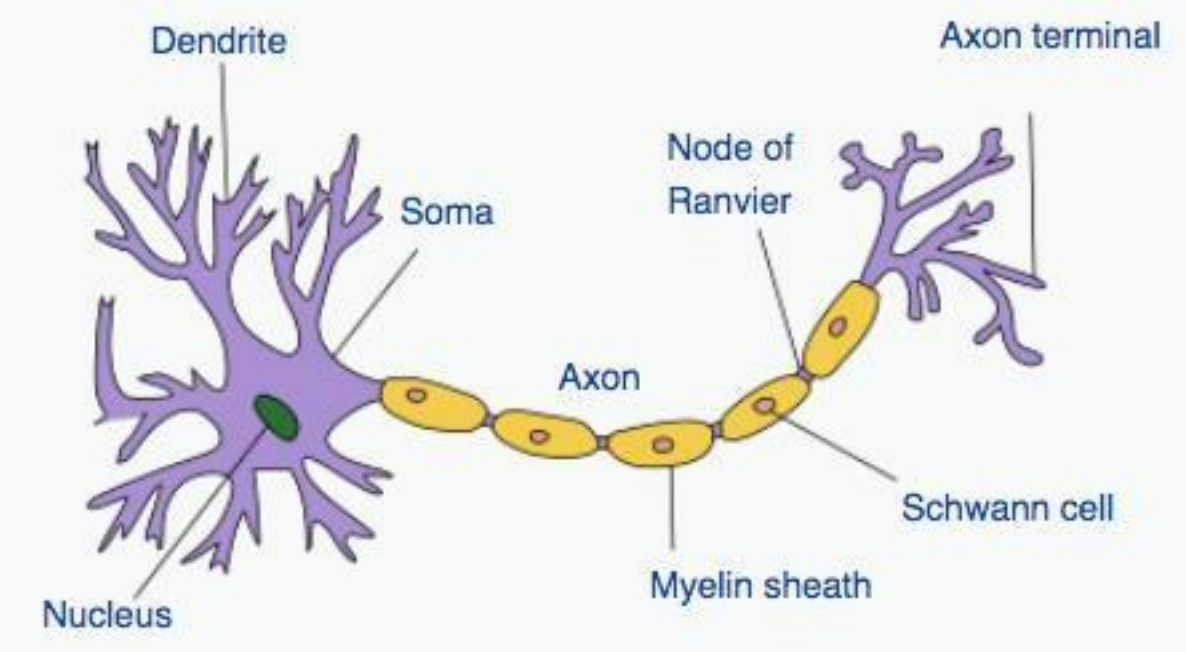
Neutral Network Introduction

Trainer : Sujata Mohite
Email: sujata.mohite@sunbeaminfo.com


AI
ML
DL
NN

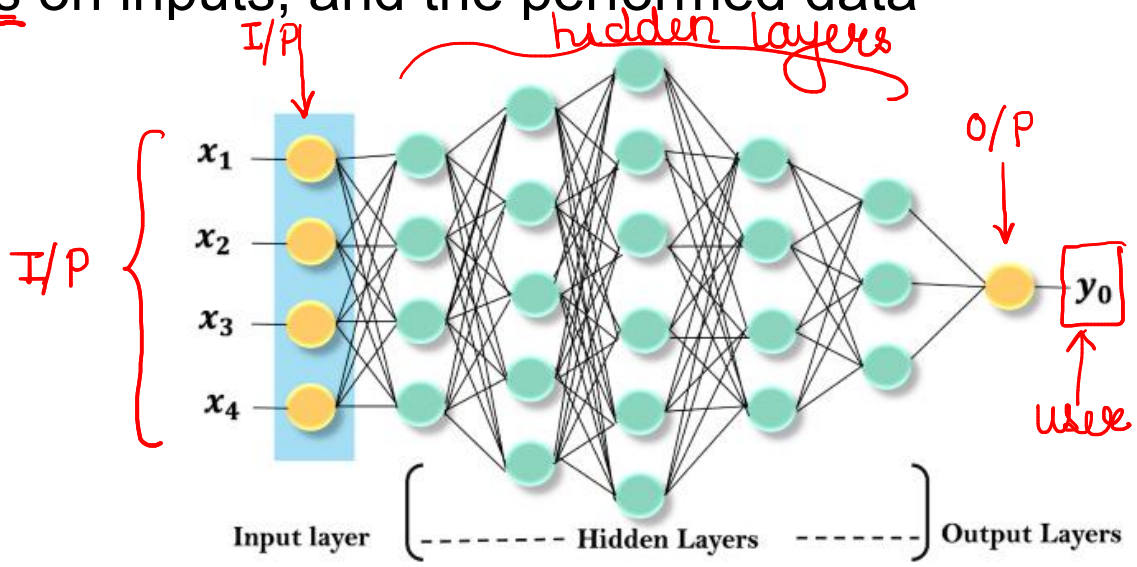
Neuron

- A neuron or nerve cell, is an electrically excitable cell that communicates with other cells via specialized connections called synapses
- A typical neuron consists of a cell body (soma), dendrites, and a single axon
- The soma is the body of the neuron
- The dendrites of a neuron are cellular extensions with many branches
- The axon primarily carries nerve signals away from the soma, and carries some types of information back to it.



Neuron

- The first layer is called an Input layer, the last layer is called an output layer, and all layers between these two layers are called hidden layers.
- In the deep neural network, there are multiple hidden layers, and each layer is composed of neurons. These neurons are connected in each layer.
- The input layer receives input data, and the neurons propagate the input signal to its above layers.
- The hidden layers perform mathematical operations on inputs, and the performed data forwarded to the output layer.
- The output layer returns the output to the user.



Types of Neural Network

- Artificial Neural Networks (ANN) ← huge & complex data / tables
- Convolution Neural Networks (CNN) ← images / videos
- Recurrent Neural Networks (RNN) ← text data / audio data / Time Series data

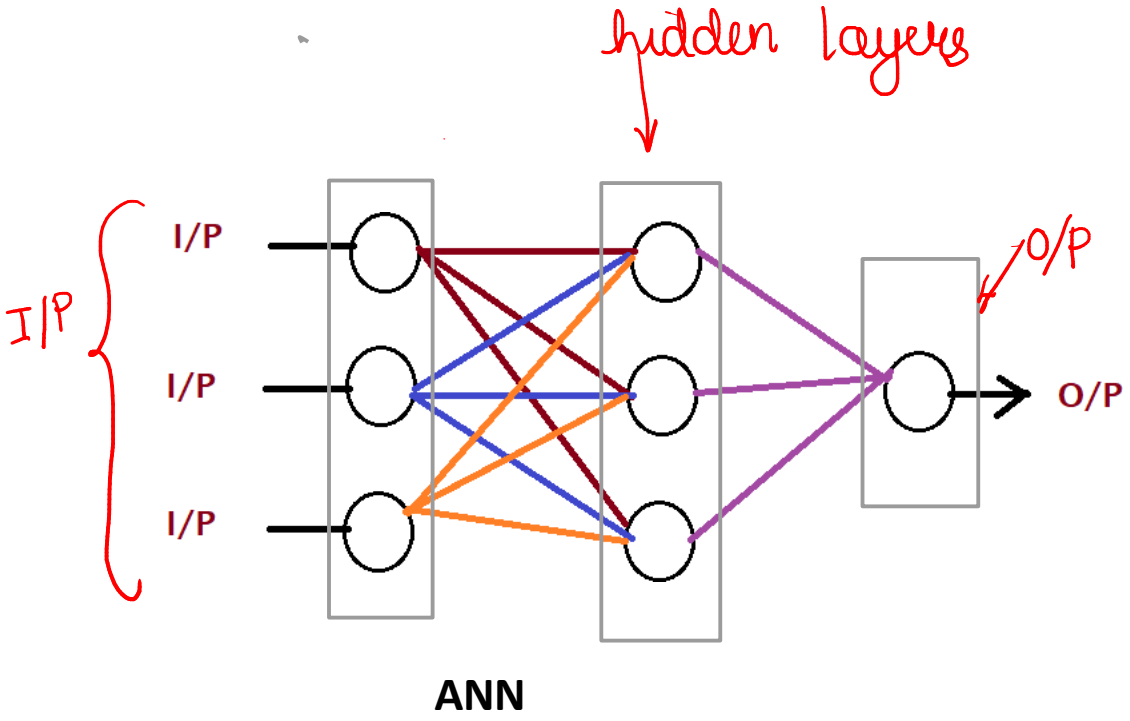
Data is small, we can go for diff. ML algos.

so, huge data & unstructured data we will go for NN.

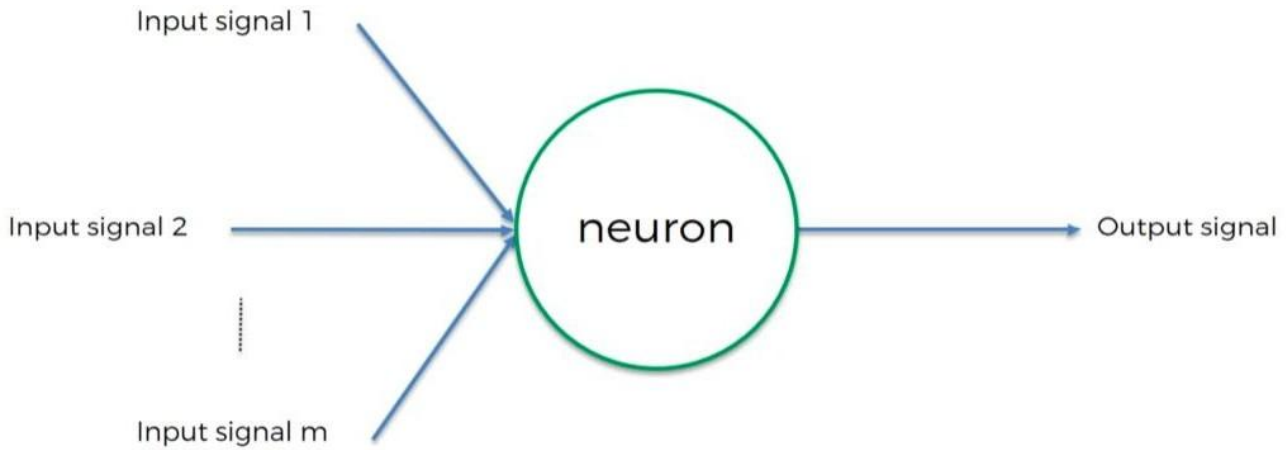
images/videos

Artificial Neural Network

- Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neutral networks that constitute animal brain
- The basic building block is a neuron

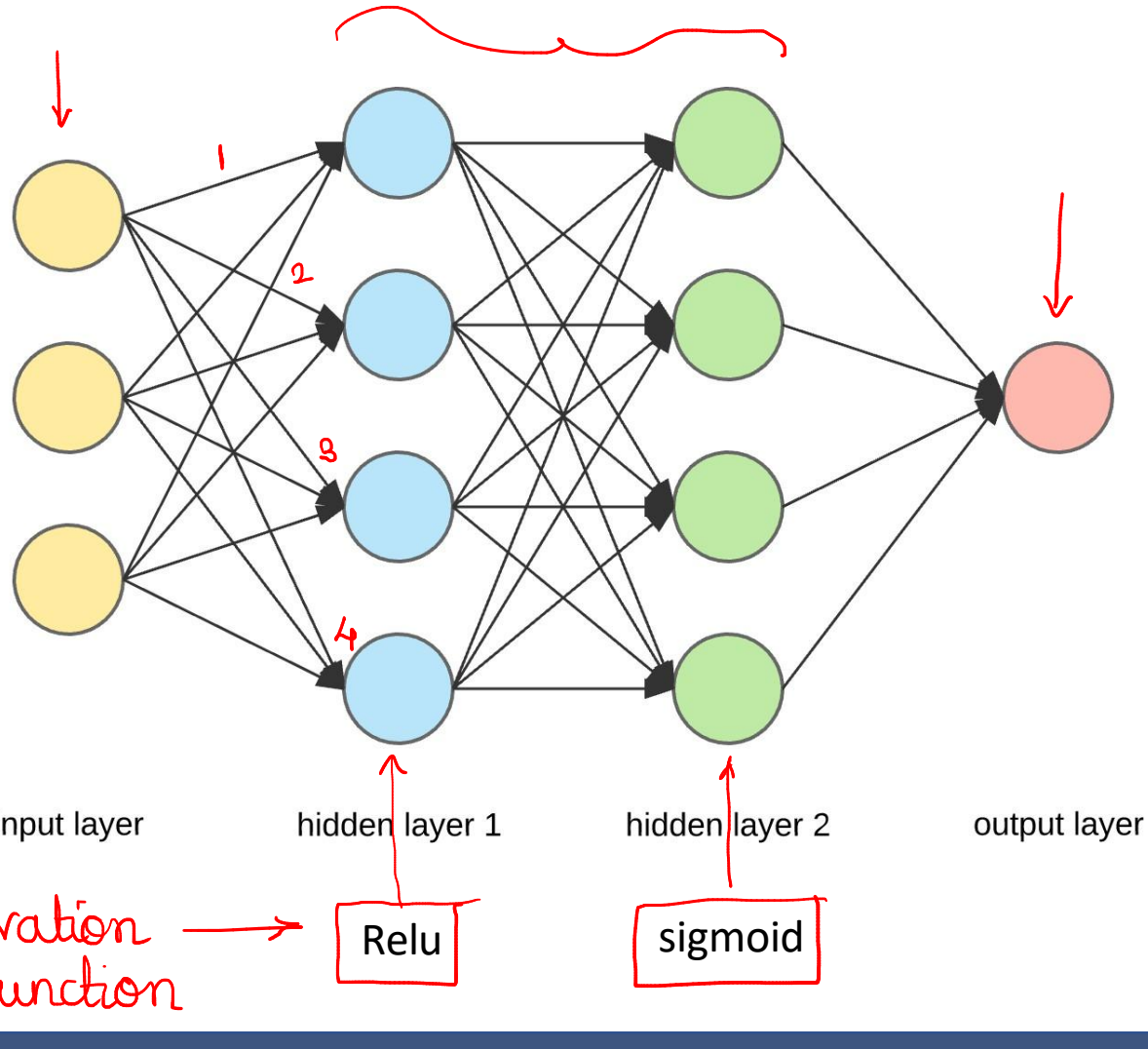


ANN



Artificial Neural Network

- Artificial Neural Networks (ANN) are **multi-layer** fully-connected neural nets
- They consist of an **input layer**, multiple hidden layers, and an **output layer**
- Every node in one layer is connected to every other node in the next layer
- We can make the network deeper by increasing the number of hidden layers.

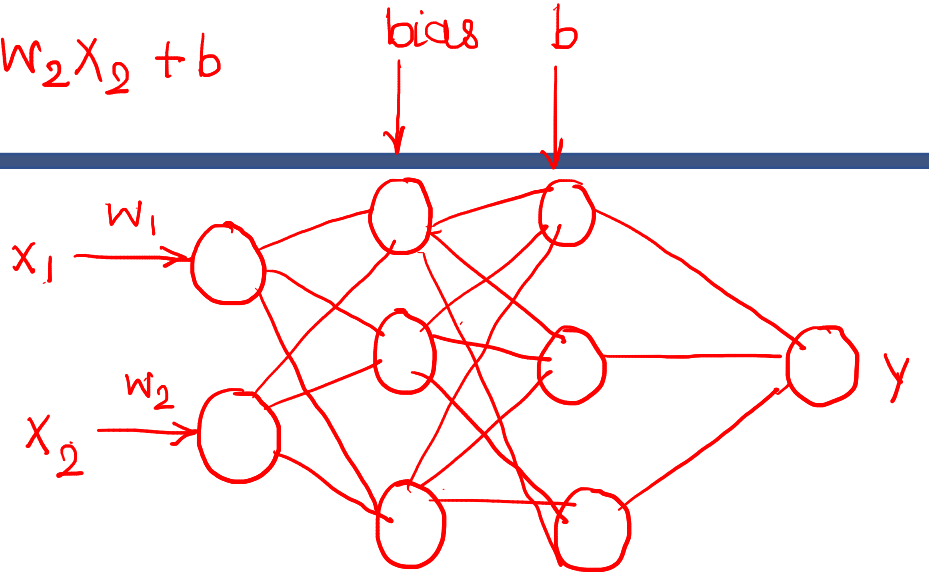


Characteristics of Artificial Neural Network

- It is neurally implemented mathematical model (weights & bias)
- It contains huge number of interconnected processing elements called **neurons** to do all operations
- Information stored in the neurons are basically the weighted linkage of neurons
- The input signals arrive at the processing elements through connections and connecting weights.
- It has the ability to learn , recall and generalize from the given data by suitable assignment and adjustments of weights
- The collective behavior of the neurons describes its computational power, and no single neuron

Components of Perceptron

$\sum_{i=1}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + b$



A **Perceptron** is an **Artificial Neuron**.

$\sum wixi + bias = w1x1 + w2x2 + w3x3 + bias$

output = f(x) = 1 if $\sum w1x1 + b >= 0$; 0 if $\sum w1x1 + b < 0$

b = Bias: The bias term helps the perceptron make adjustments independent of the input, improving its flexibility in learning.

w = Weights: Each input feature is assigned a weight that determines its influence on the output. These weights are adjusted during training to find the optimal values. at the o/p

o/p →

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

Neural Networks

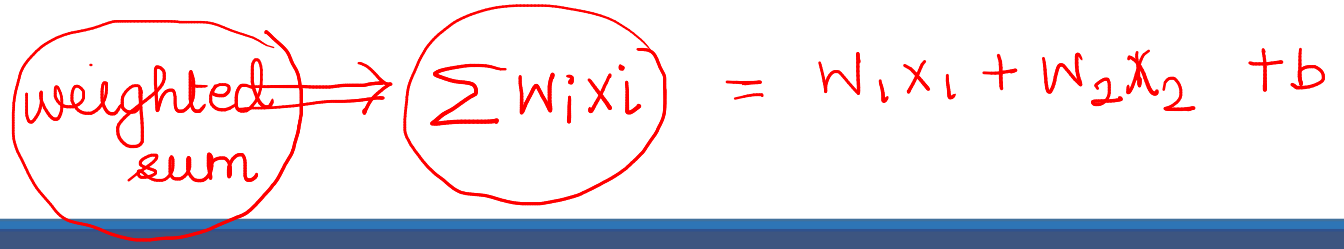
b = Bias: The bias term helps the perceptron make adjustments independent of the input, improving its flexibility in learning.

w = Weights: Each input feature is assigned a weight that determines its influence on the output. These weights are adjusted during training to find the optimal values.

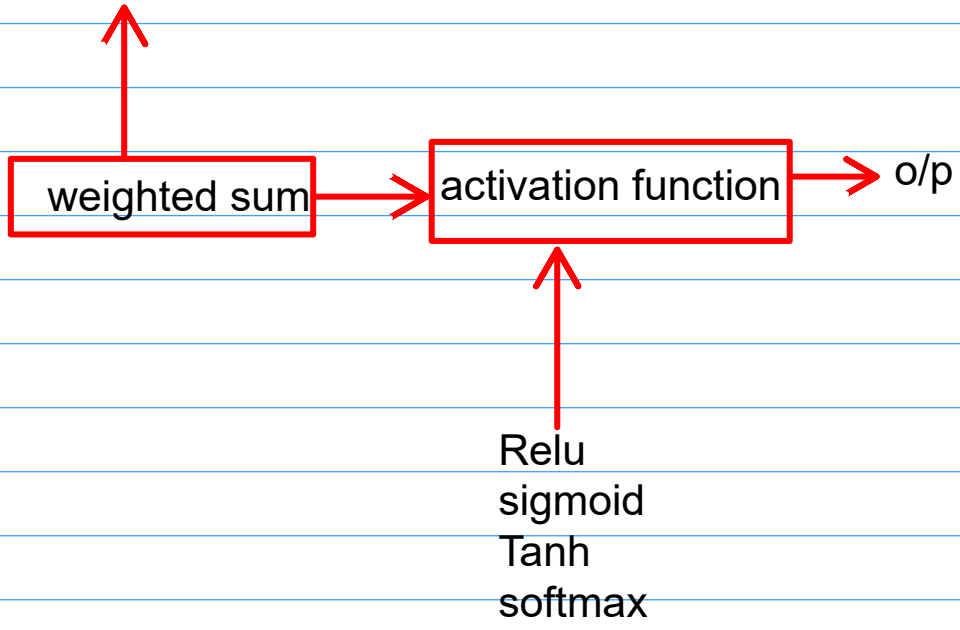
Summation Function: The perceptron calculates the weighted sum of its inputs using the summation function. The summation function combines the inputs with their respective weights to produce a weighted sum.

Activation Function: The weighted sum is then passed through an activation function, which take the summed values as input and compare with the threshold and provide the output as 0 or 1.

Output: The final output of the perceptron, is determined by the activation function's result. For example, in binary classification problems, the output might represent a predicted class (0 or 1).



$$w_i^T x_i = w_1 x_1 + w_2 x_2 + b$$



Neural Networks

In a neural network, the activation function's role is -

- to introduce non-linearity,
- enabling the network to learn complex patterns and relationships in data.

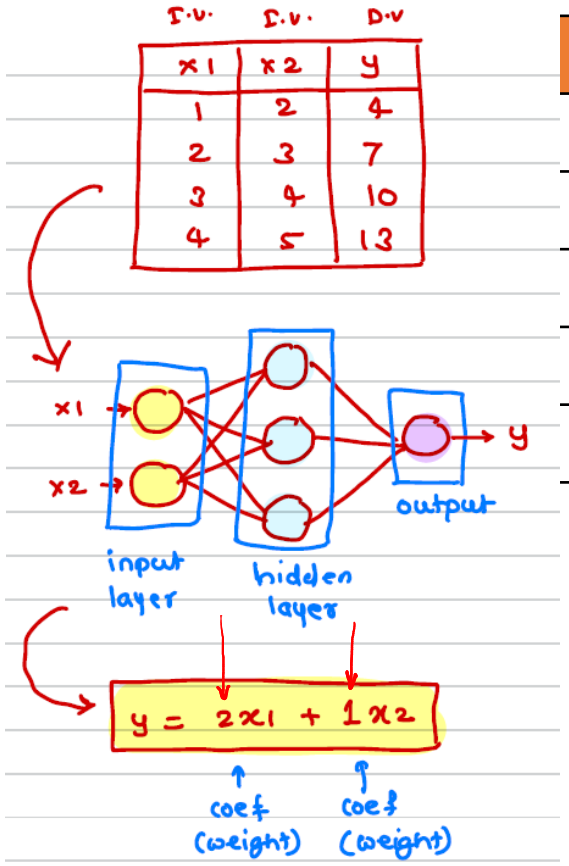
It transforms the weighted sum of inputs into an output signal for a neural network layer, determining whether a neuron should be activated and influencing the network's overall behavior.

Weights

$y = 2x_1 + 1x_2$

$2 \times 1 + 1 \times 2 = 4$

$2 \times 2 + 1 \times 3 = 7$



(weight)

X	Y
1 x2	2
2 x2	4
3 x2	6
4 x2	8
16 x2	? (32)

Y = 2x

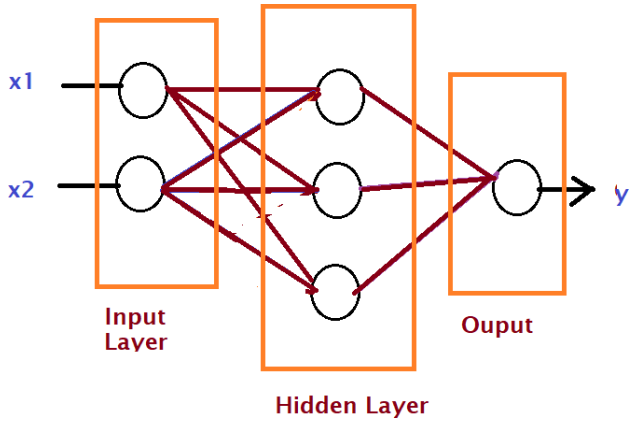
value before input (x) is called as weight

Y = 2 * x

Coefficient (weight)

X1 (input variable)	X2 (input variable)	Y (dependent variable)
1	2	4
2	3	7
3	4	10
4	5	13

formula is generated on there own by the NN



Y = 2*x1 + 1*x2

Coefficient (weight)

Coefficient (weight)

An epoch refers to one complete pass through the entire training dataset once by the neural network

Trial (EPOCH) : 1

1 x 2 + 1 x 3 = 5 X

Y = 1*x1 + 1 * x2

y = 1+2 = 3 X

y = 3 + 7 = 10

Trial (EPOCH) :2

Y = 2 * x1 + 1 * x2

y = 2 * 1 + 1*2 = 4

y = 2 * 2 + 1* 3 = 7

It proves the formula

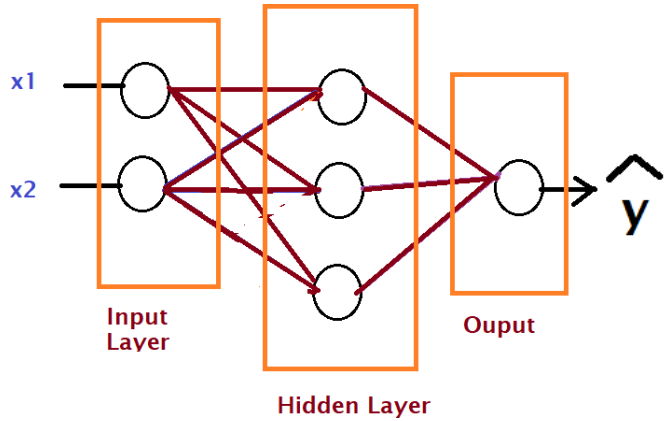
Y = 2*x1 + 1*x2 is giving correct prediction

(2 x 1 + 1 x 2) = 4 ✓

2 x 2 + 1 x 3 = 7 ✓

Output / predicted value is represented as $\hat{y}$

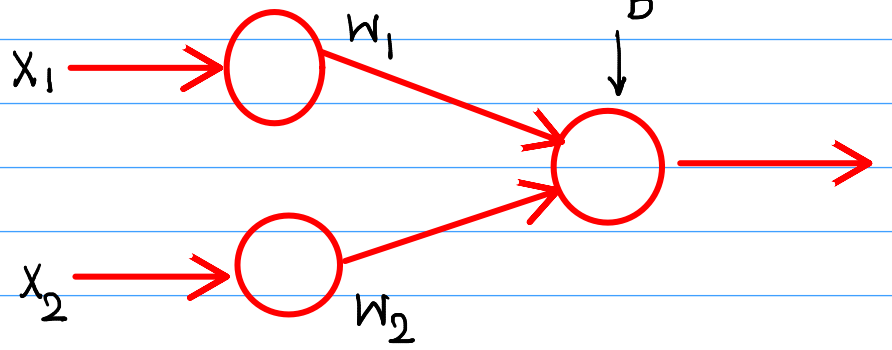
X1 (input variable)	X2 (input variable)	Y (dependent variable)
1	2	4 ✓
2	3	7
3	4	10
4	5	13



Y = 2*x1 + 1*x2

Coefficient (weight)

Coefficient (weight)



predicted value

$y - \hat{y}$ = error

cost function

Application of Neural Network

- Neural network is suitable for the research on *Animal behavior, predator/prey relationships etc.*
- It would be easier to do *proper valuation* of property, buildings, automobiles, machinery etc. with the help of neural network.
- Neural Network can be used in betting on horse races, sporting events and most importantly in stock market .
- It can be used to predict the correct judgement for any crime by using a large data of crime details as input and the resulting sentences as output.
- By analyzing data and determining which of the data has any fault (files diverging from peers) called as *Data mining, cleaning and validation* can be achieved through neural network.
- Neural Network can be used to predict targets with the help of echo patterns we get from sonar, radar, seismic and magnetic instruments .
- It can be used efficiently in *Employee hiring* so that any company can hire right employee depending upon
  the skills the employee has and what should be it's productivity in future .
- It has a large application in *Medical Research* .
- It can be used to for *Fraud Detection* regarding credit cards , insurance or taxes by analyzing the past records

How does it work?

x_1 = study hrs

x_2 = sleep hrs

x_3 = Quiz

Input

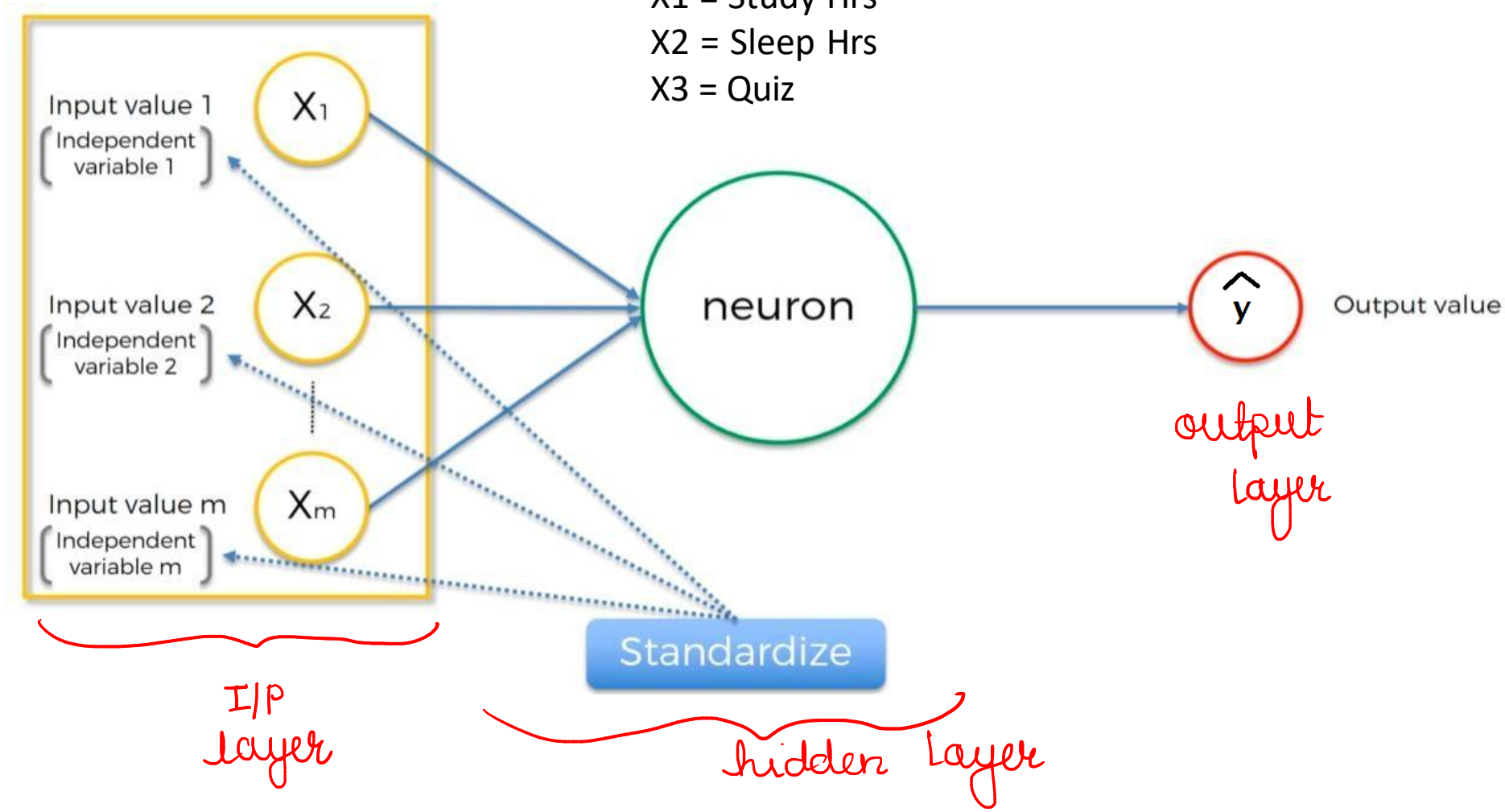
O/P

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

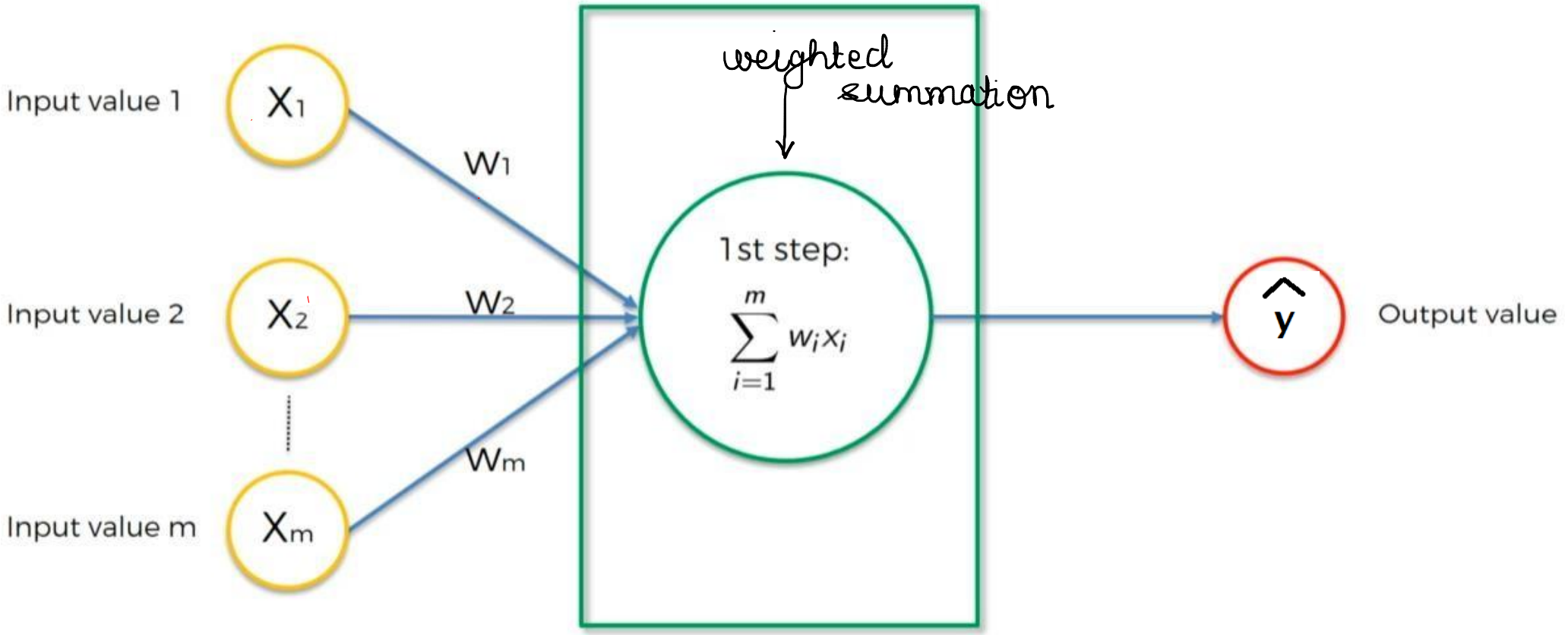
Exam ← y

Neuron

X1 = Study Hrs
X2 = Sleep Hrs
X3 = Quiz



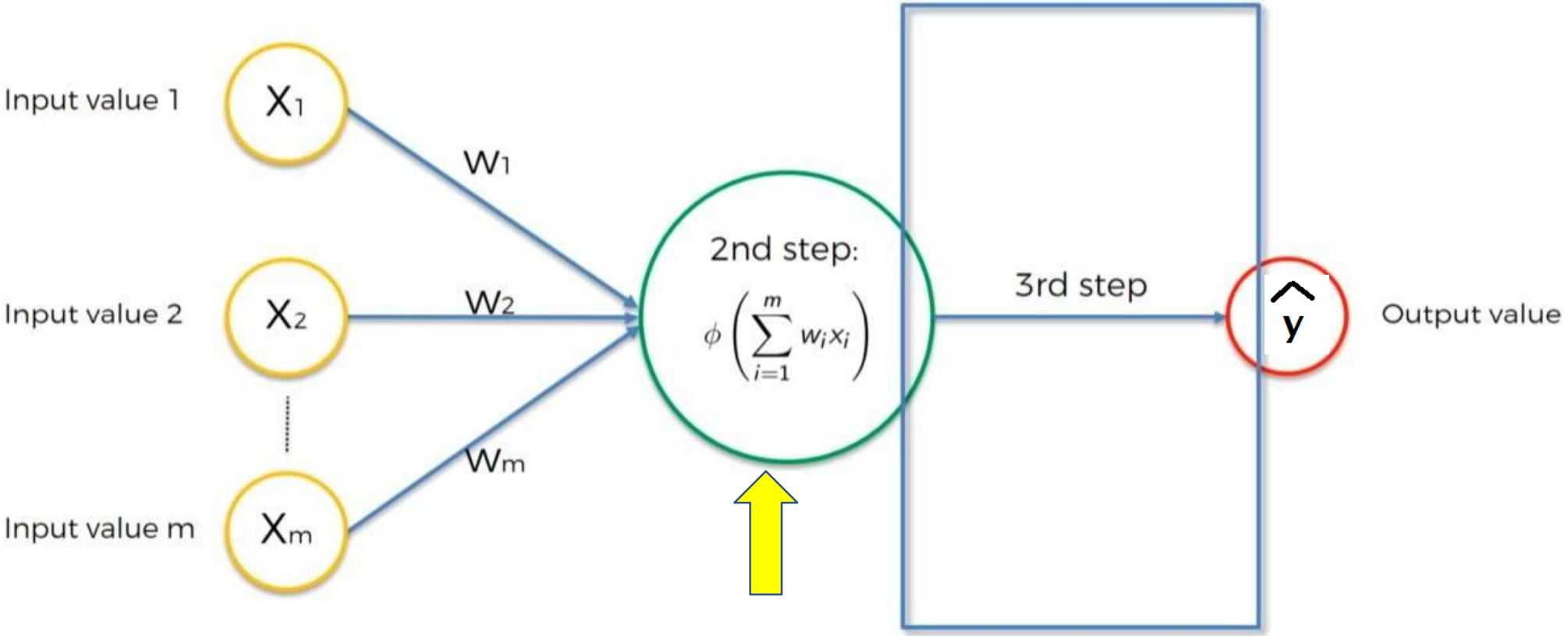
Neuron (Perceptron)



Total = x1 * w1 + x2 * w2 + x3 * w3 + b

Activation Function

← generates a f^n/model based on the calculatn of weights.

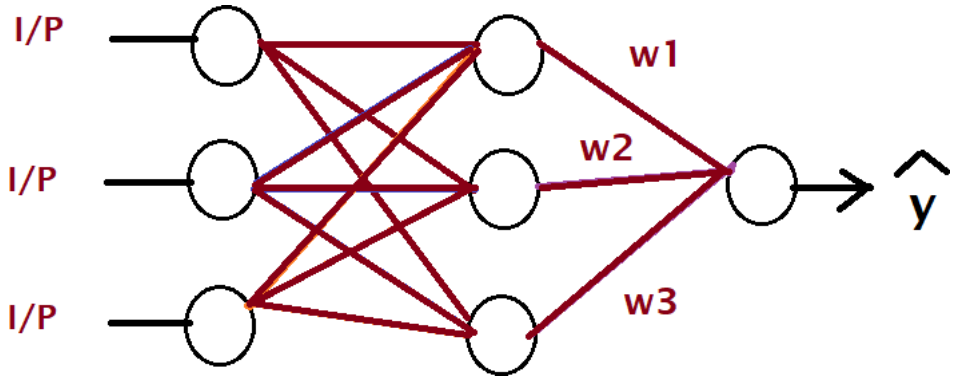


ϕ = **Activation function**

Activation Function

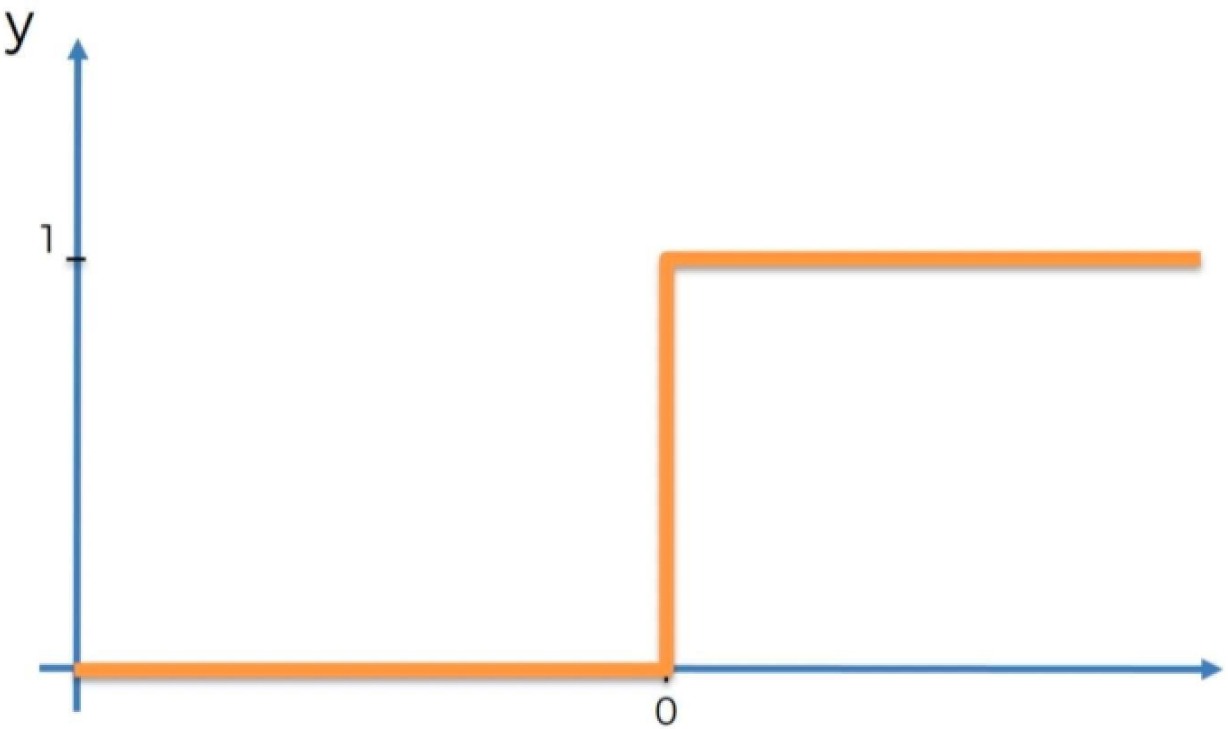
- Their main purpose is to convert a input signal of a node in a A-NN to an output signal
- That output signal can be used as a input in the next layer in the stack
- Specifically in A-NN
  - we do the sum of products of inputs(**X**) and their corresponding Weights(**W**)
  - apply a Activation function **f(x)** to it to get the output of that layer
  - feed it as an input to the next layer

$x_1 w_1 + x_2 w_2 - - -$



Activation Function (tanh / softmax / Relu / sigmoid)

Threshold Function



$$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

Threshold Function

Threshold Function (Binary Step Function) :-

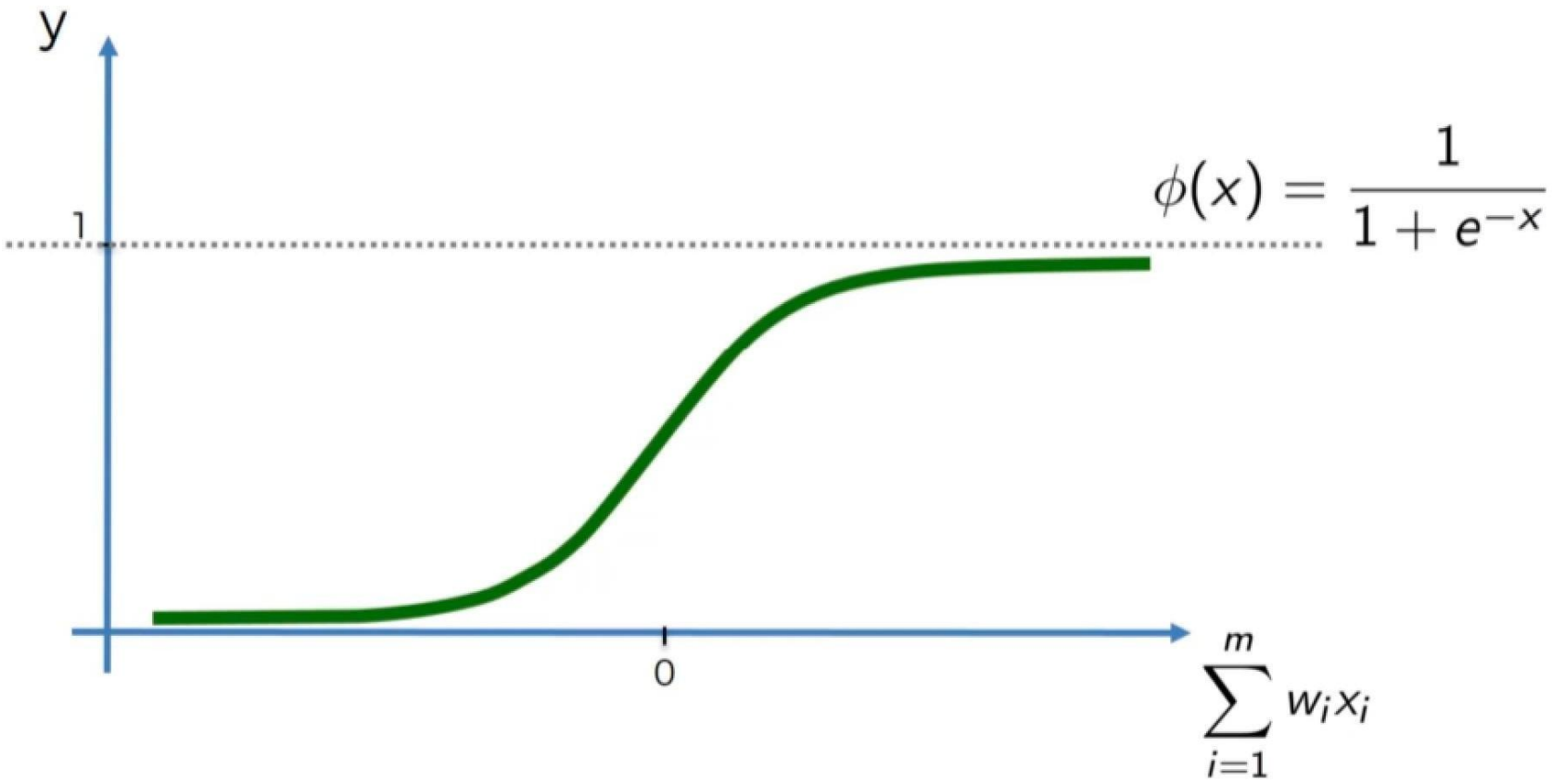
The Threshold activation function is one of the **simplest activation functions**.

It produces a **binary output** based on a fixed threshold value.

The output ranges from **0 to 1**.

The Threshold function is mainly used in **perceptrons and decision-making systems**, but it is **not used in training deep neural networks** due to non-differentiability.

Sigmoid Function



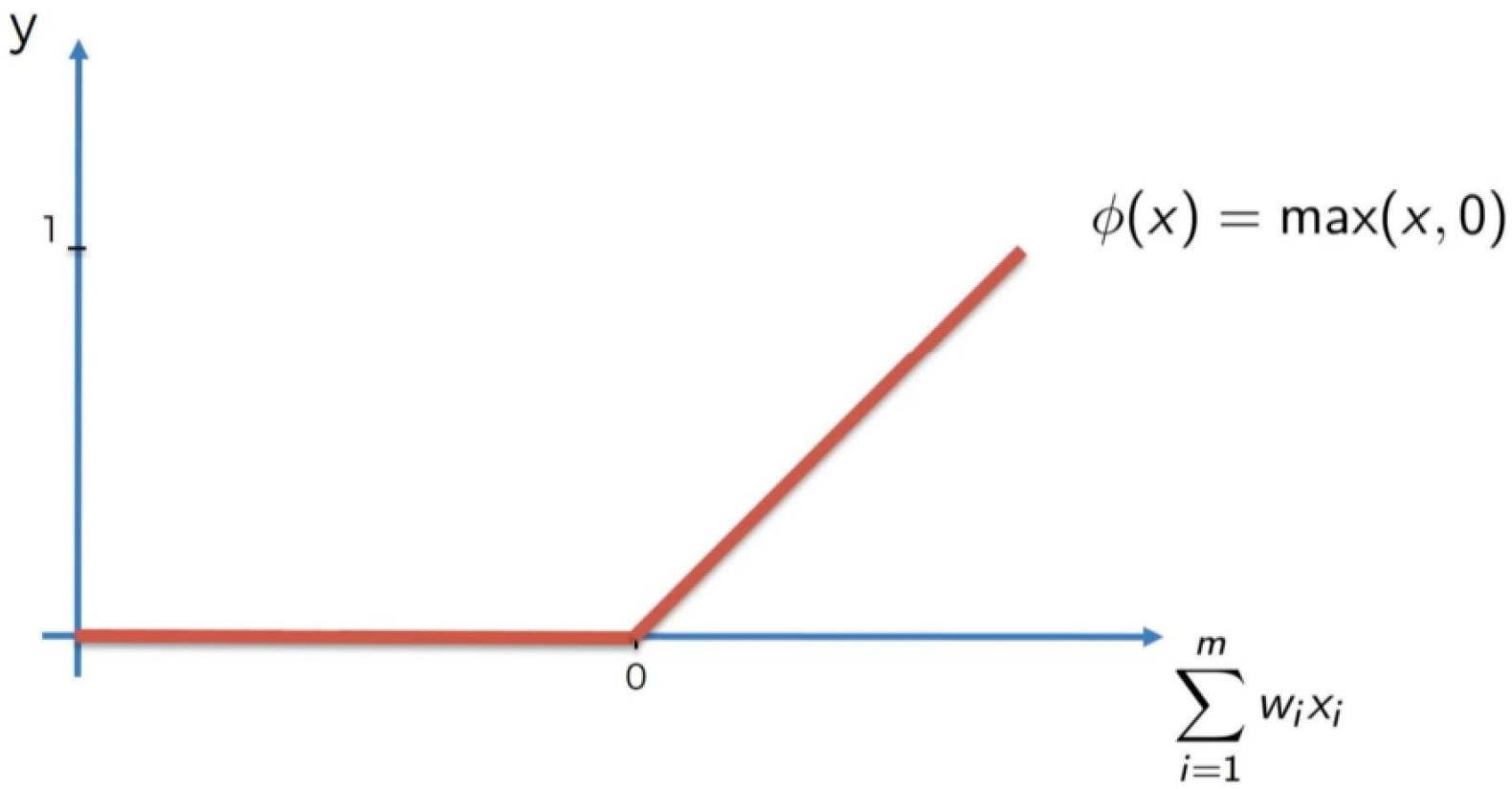
Sigmoid Function

- The Sigmoid activation function is a **non-linear activation function**.
- It maps input values into **probability-like outputs**.
- The output ranges from **0 to 1**.
- Sigmoid is commonly used in the **output layer of binary classification neural networks**.

▪ The sigmoid activation function is a mathematical function that maps any real-valued number into a value between 0 and 1.

▪ It's commonly used in binary classification tasks.

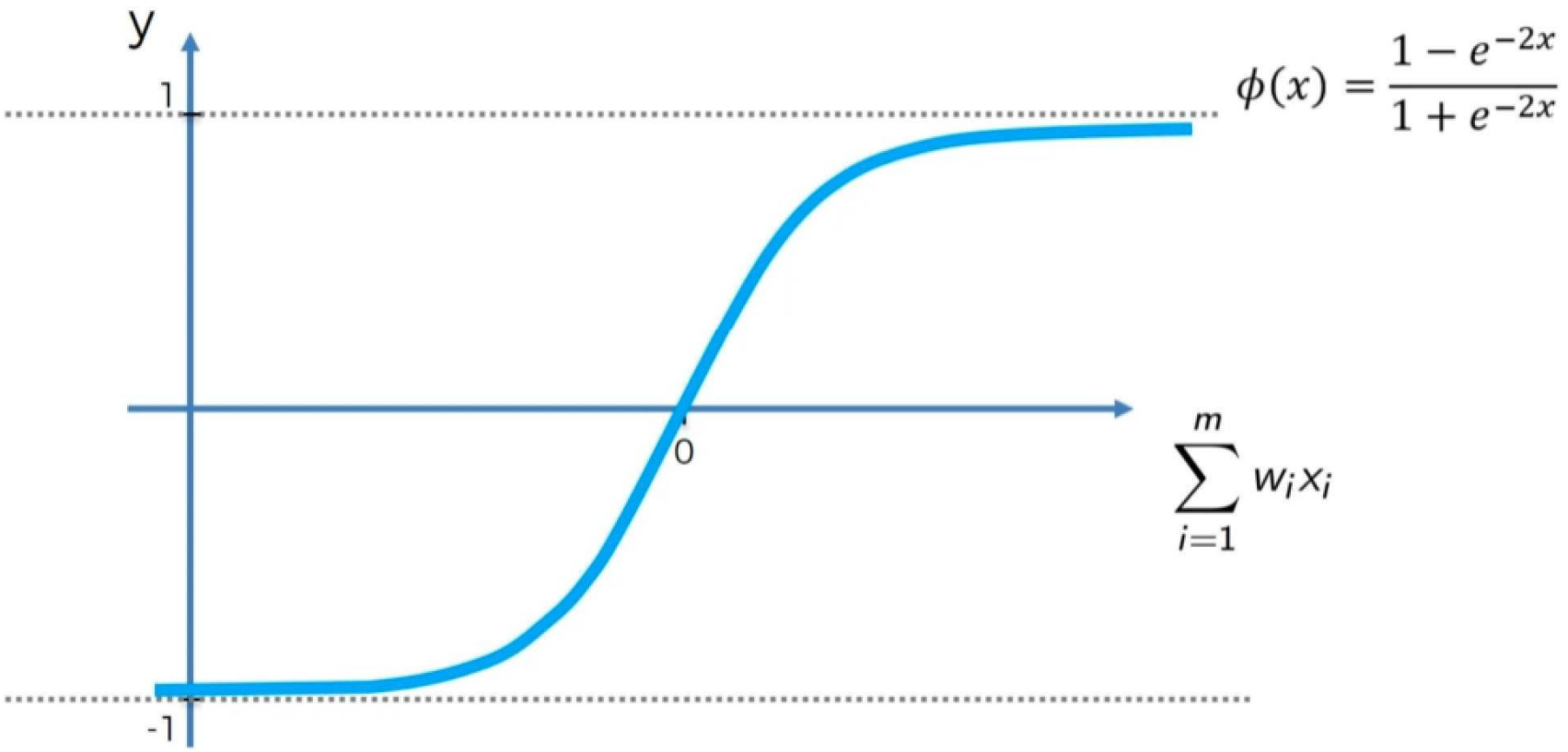
Rectifier Function



Rectifier Function

- The **Rectifier Function**, commonly known as **ReLU**, is one of the **most widely used activation functions** in deep learning.
- The ReLU activation function is one of the most widely used activation function.
- The output ranges from **0 to infinity**.
- It is commonly used in the **hidden layers of deep neural networks**.

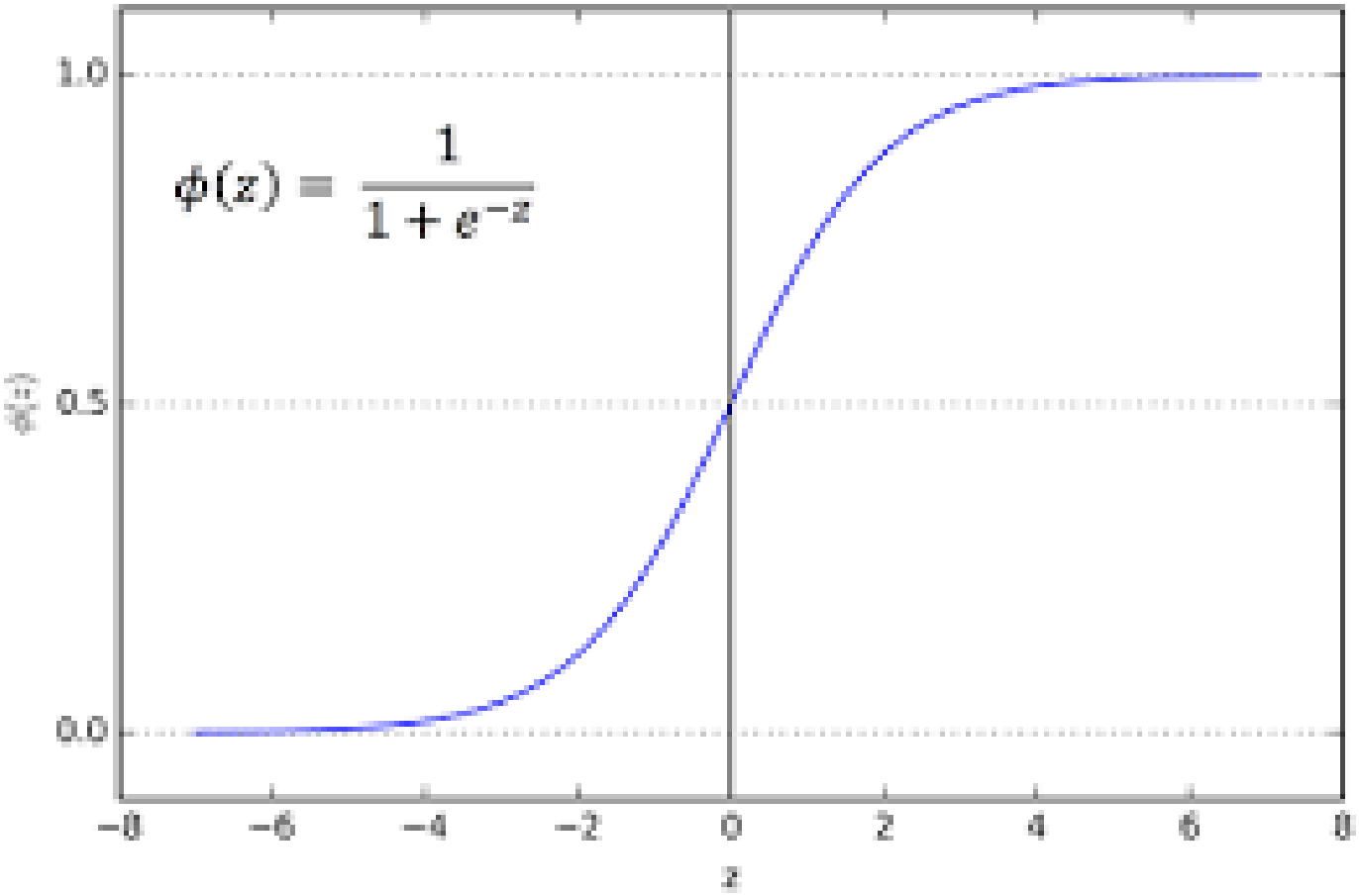
Hyperbolic Tangent Function



Hyperbolic Tangent Function

- The Tanh activation function is a **non-linear activation function**.
- It is a scaled version of the sigmoid function.
- The output ranges from **−1 to +1**.
- Tanh is commonly used in **hidden layers of neural networks**, especially when zero-centered outputs are preferred.

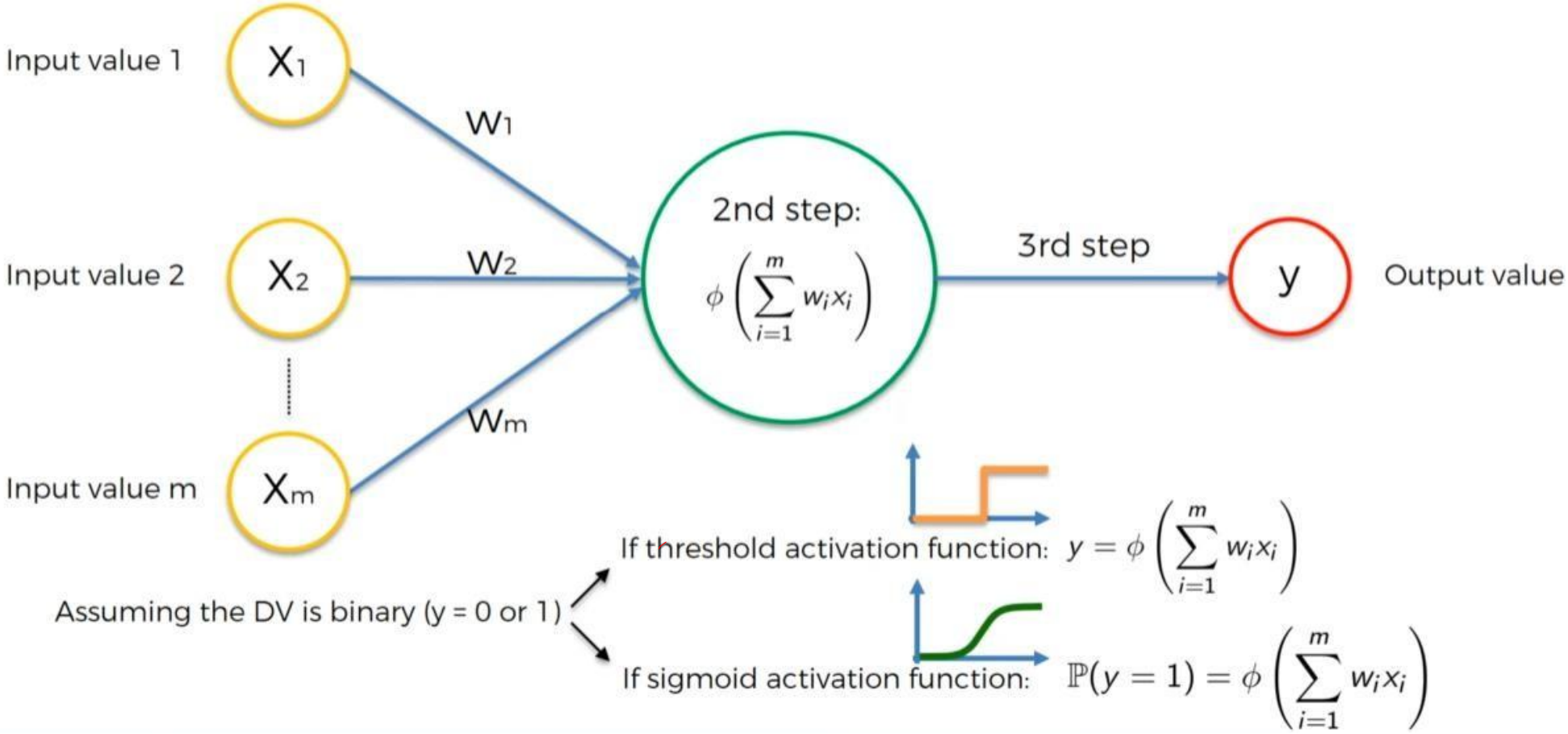
Softmax Function



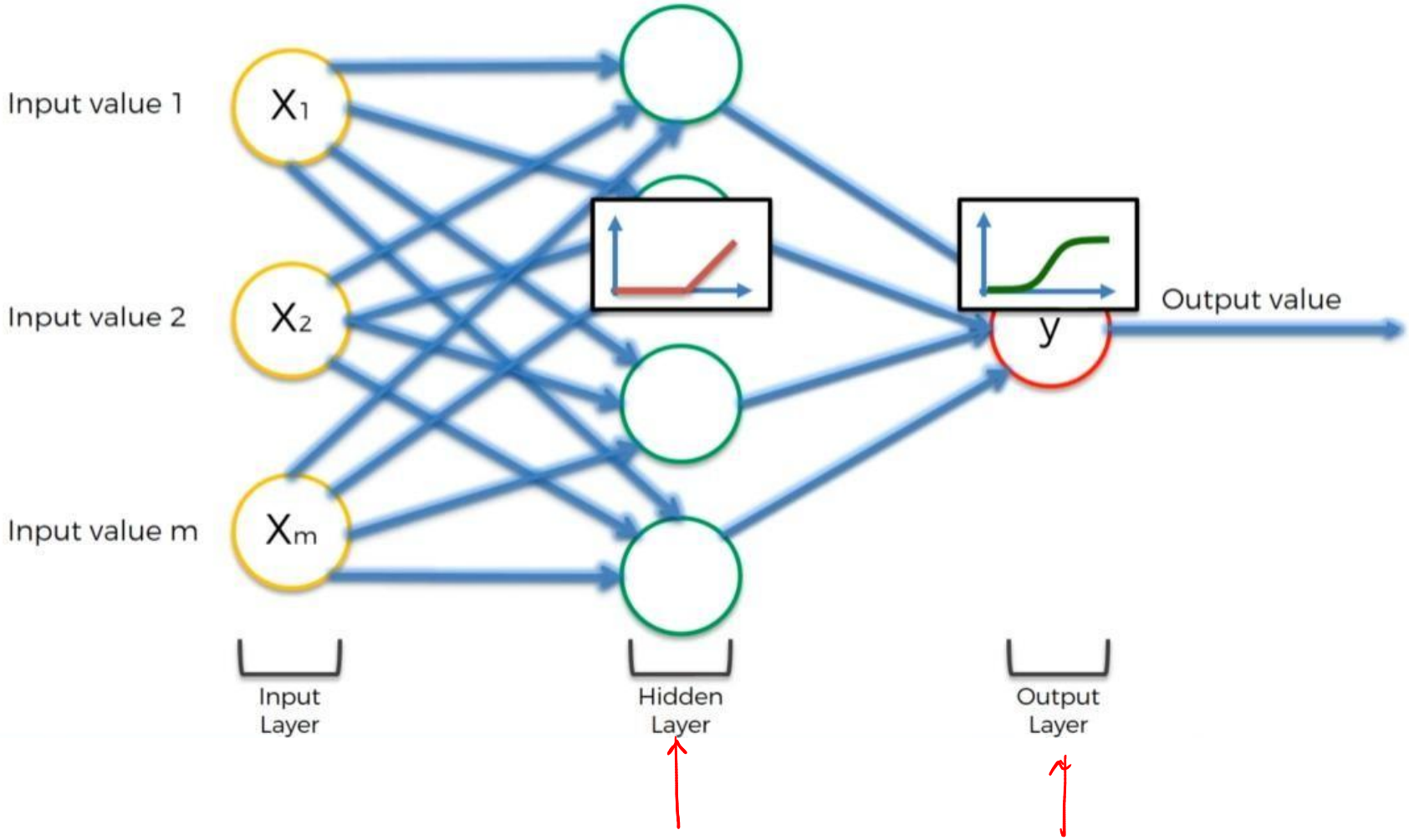
Softmax Function

- The Softmax activation function is used for **multi-class classification problems**.
- It converts raw scores into **probability values**.
- The output ranges from **0 to 1**, and the **sum of all outputs equals 1**.
- Softmax is commonly used in the **output layer of neural networks** for multi-class classification.

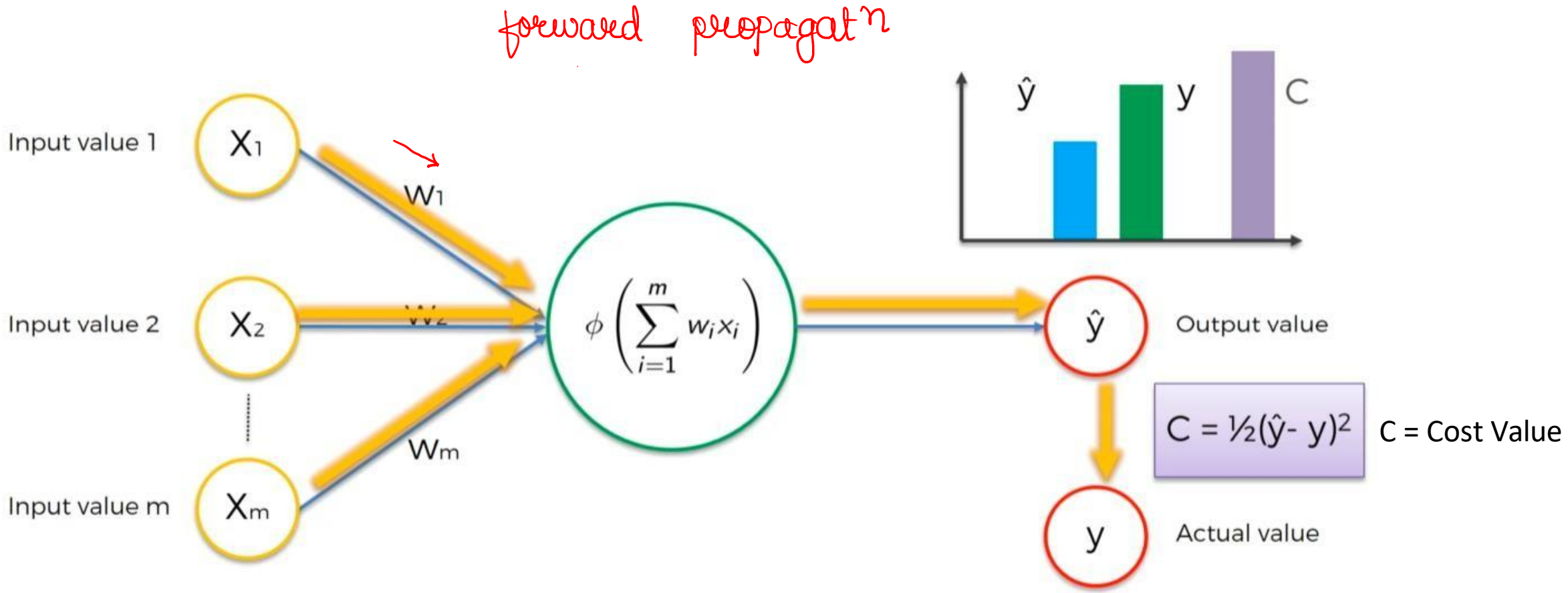
Activation Function



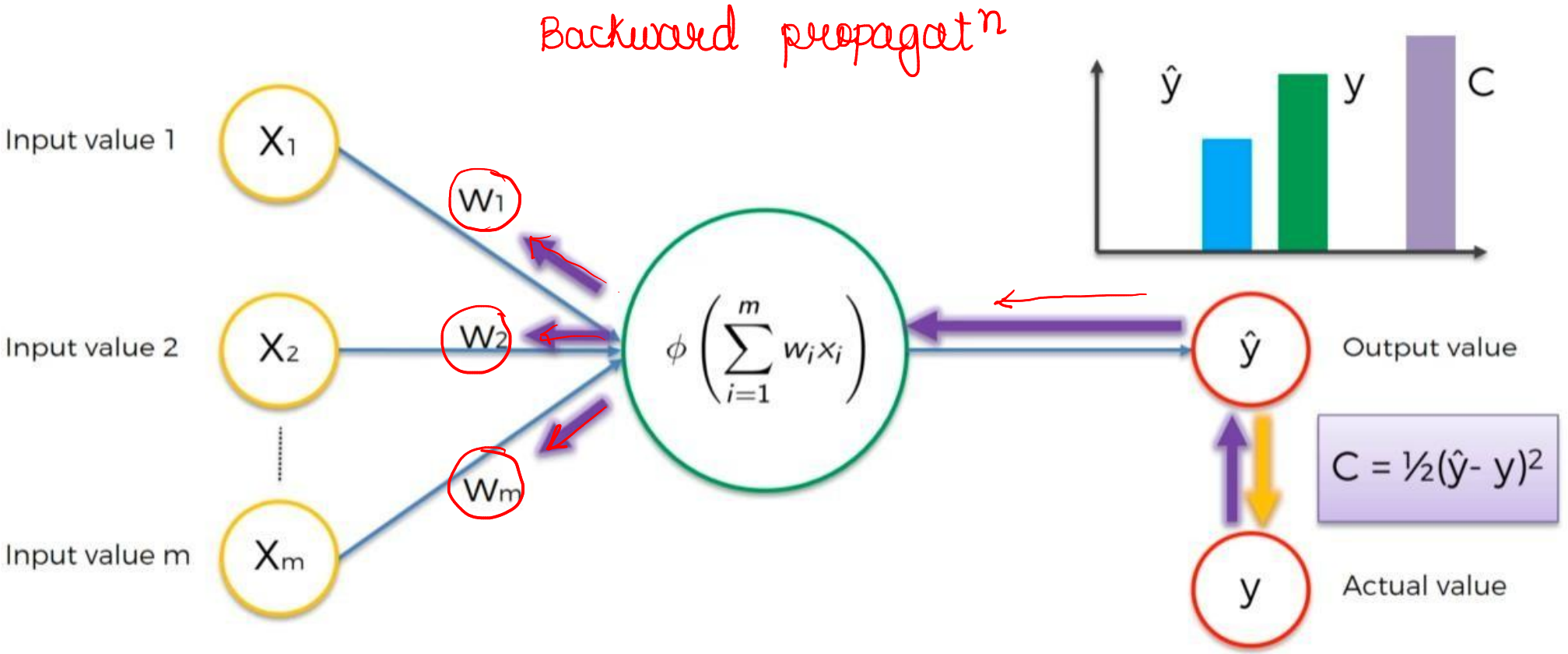
Activation Function



Neural Network - learn

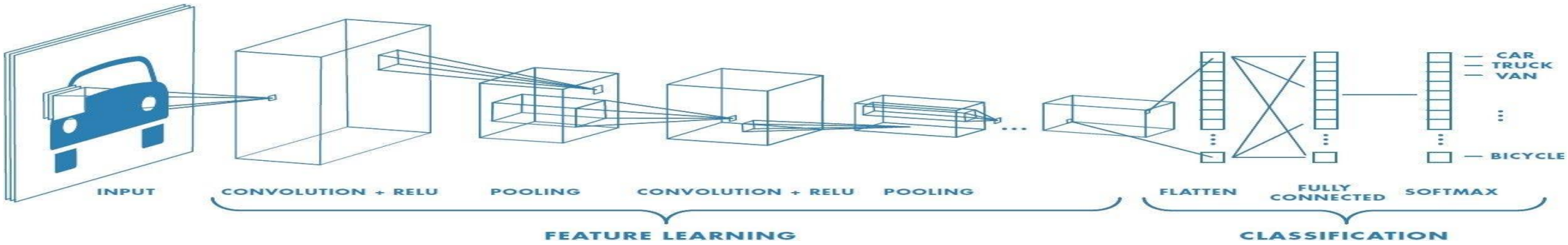


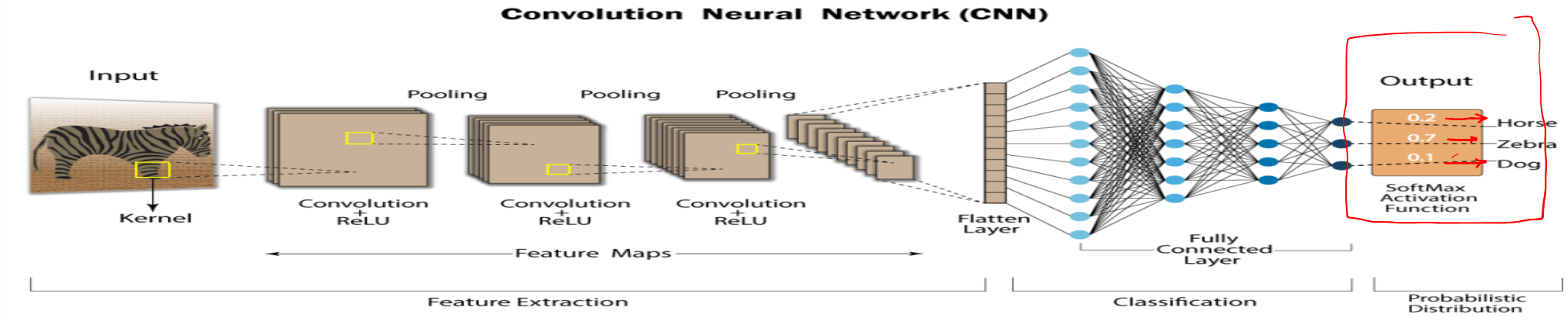
Neural Network - learn



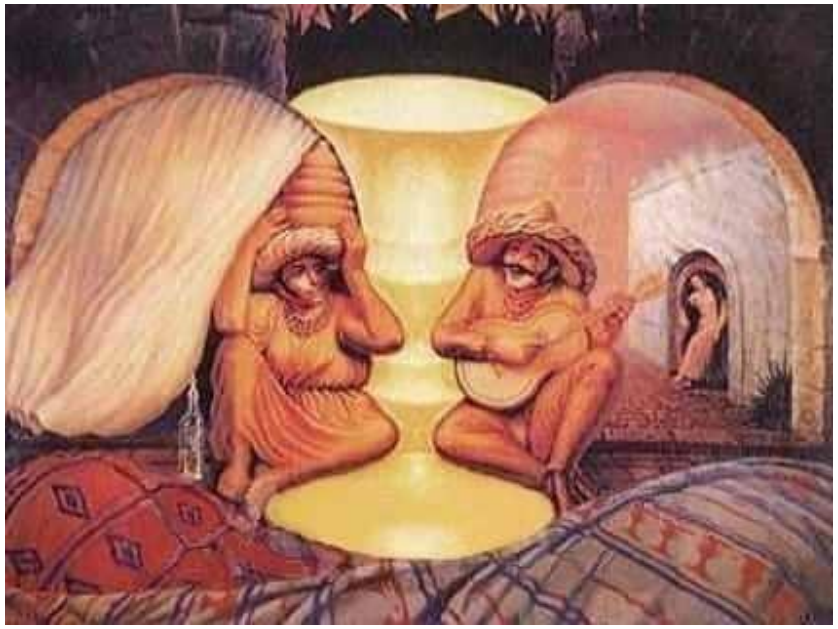
Convolution Neural Network (CNN)

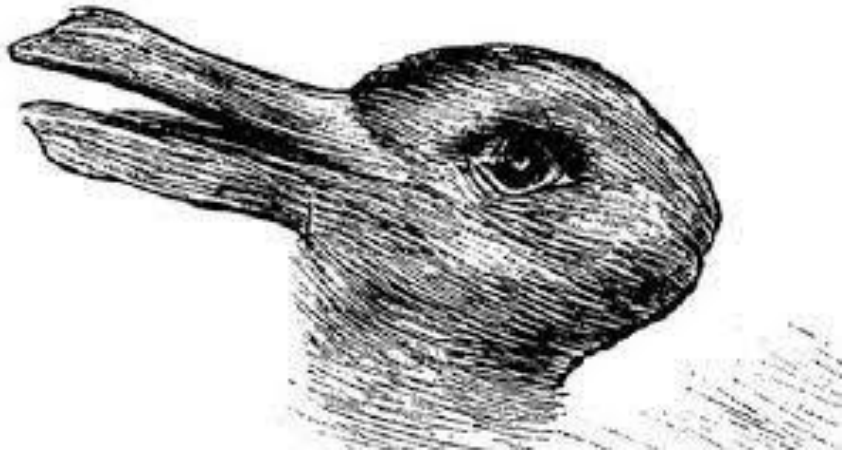
- These CNN models are being used across different applications and domains, and they're especially used in image and video processing projects.





CNN





These images are classic visual illusions. In the context of CNNs (Convolutional Neural Networks),
they're commonly used to illustrate ambiguity, feature interpretation, and hierarchical perception.

Same pixels → multiple interpretations

Rubin's vase (faces vs. vase)
Duck–rabbit
Three-eyed face illusion

In CNN terms:
The same input image can support multiple valid high-level interpretations depending on:

- which features are emphasized,
- the scale of observation,
- and how features are combined in deeper layers.

CNNs must commit to one interpretation, unlike humans who can switch.

CNNs work bottom-up:

Early layers → edges, corners, textures

Middle layers → parts (eyes, beaks, faces)

Deep layers → objects or classes

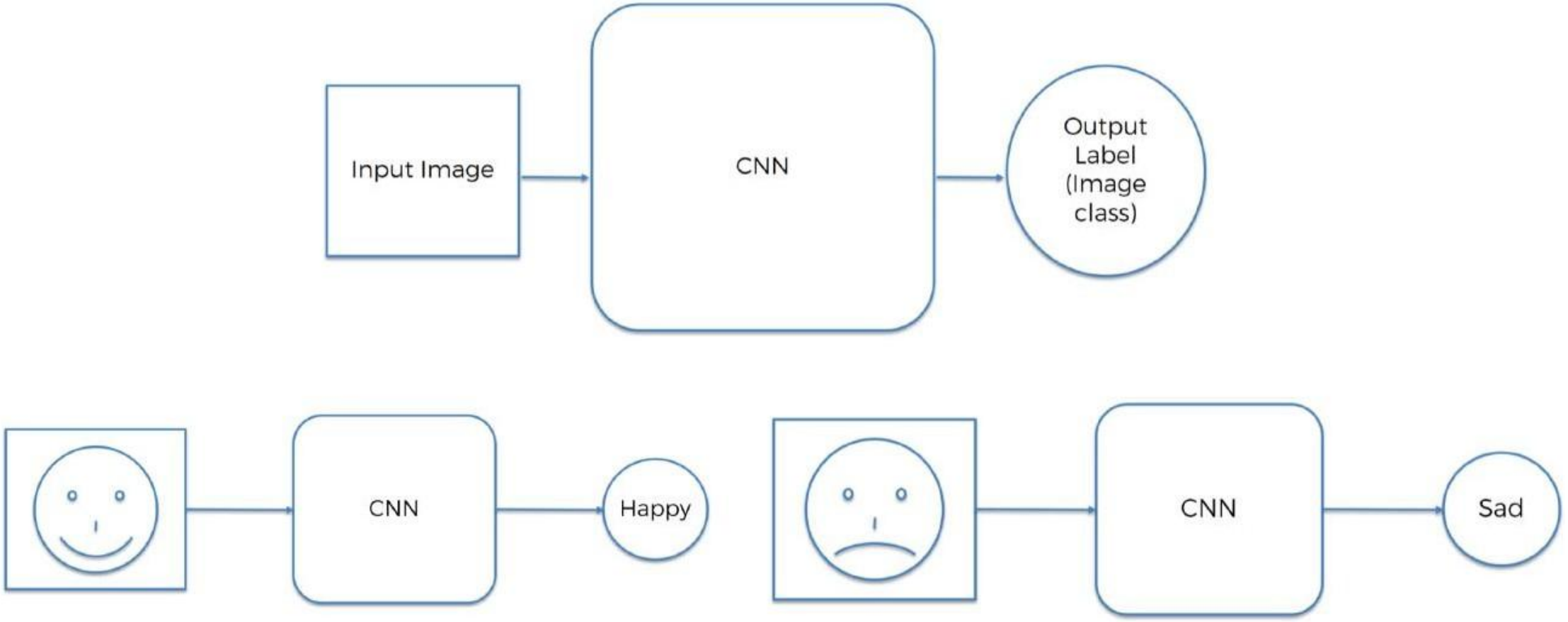
Local features are compatible with more than one global object

The duck-rabbit has edges that fit both a beak and ears.

A CNN might confidently classify only one, depending on training bias.

In CNNs, these images demonstrate that perception is a learned, biased mapping from features to labels—not an inherent understanding of visual reality.

CNN

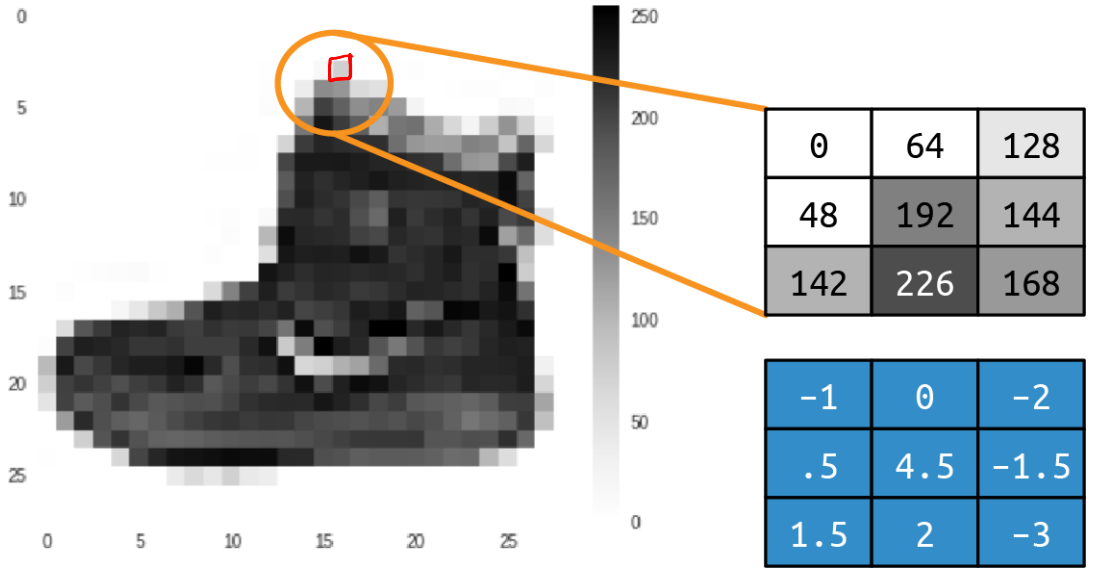


Image

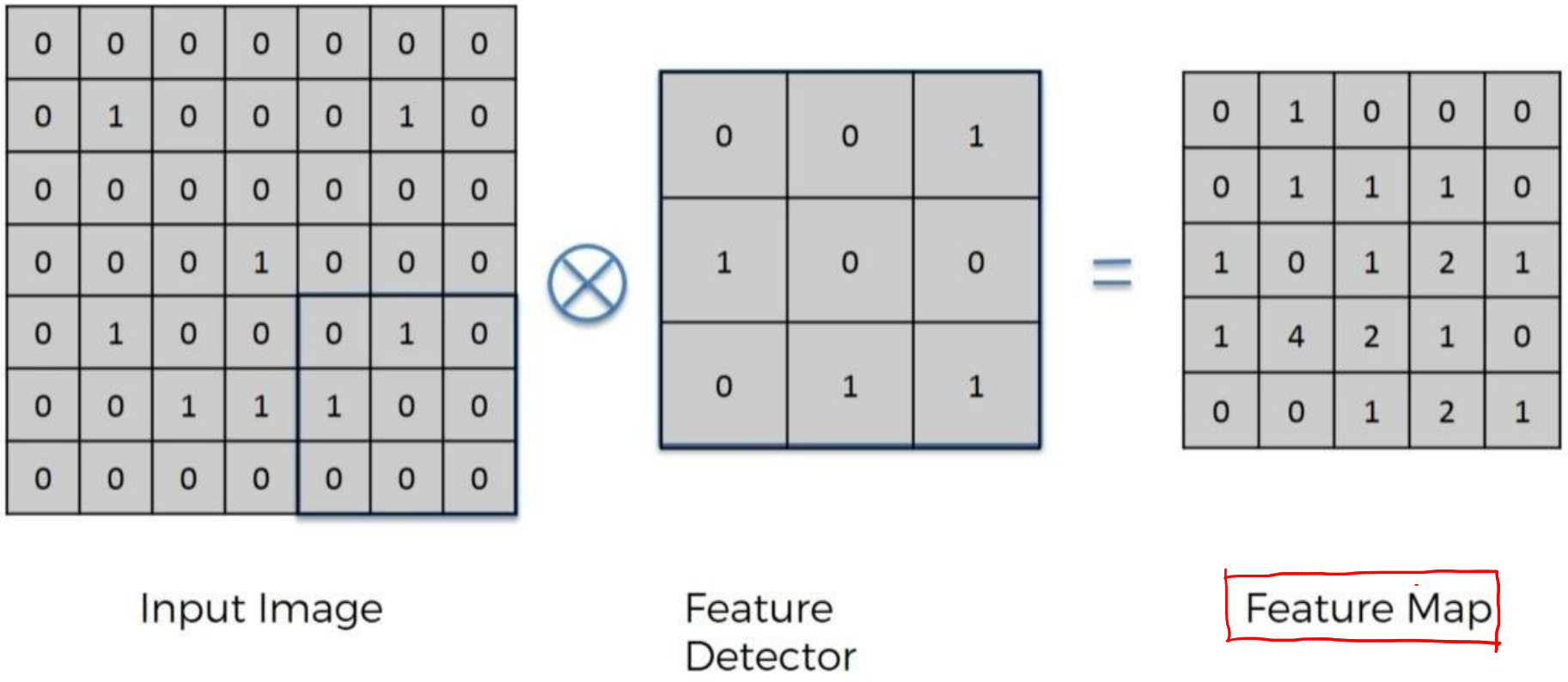
- An image is an artifact that depicts visual perception, such as a photograph or other two-dimensional picture, that resembles a subject—usually a physical object—and thus provides a depiction of it
- It can be seen as a two-dimensional (2D) view of a 3D world
- A digital image is a numeric representation of a 2D image
- It is a finite set of digital values, which are called pixels
- Pixel = picture + element
- SD = 640 * 480
- HD = 1280 * 720 / 1920 *1080
- UHD = 3840 * 2160
- …..

Convolution

- A convolution is simply a filter of weights that are used to multiply a pixel with its neighbours to get a new value for the pixel
- The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image



Convolution



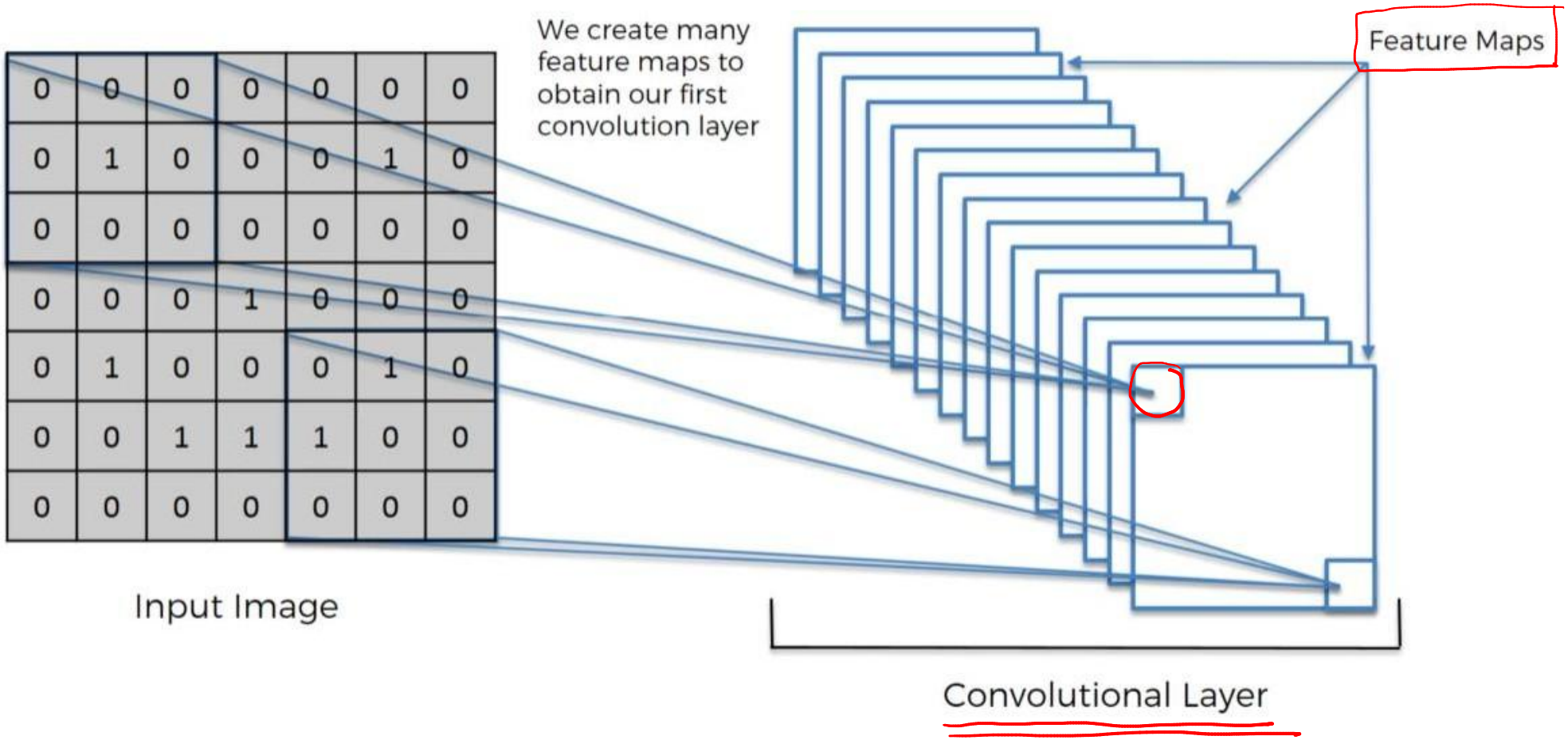
Convolutional Neural Network

- A Convolutional Neural Network (ConvNet / CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other
- The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics
- The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the
  Human Brain and was inspired by the organization of the Visual Cortex
- Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field
- A collection of such fields overlap to cover the entire visual area

Convolutional Neural Network

- There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet

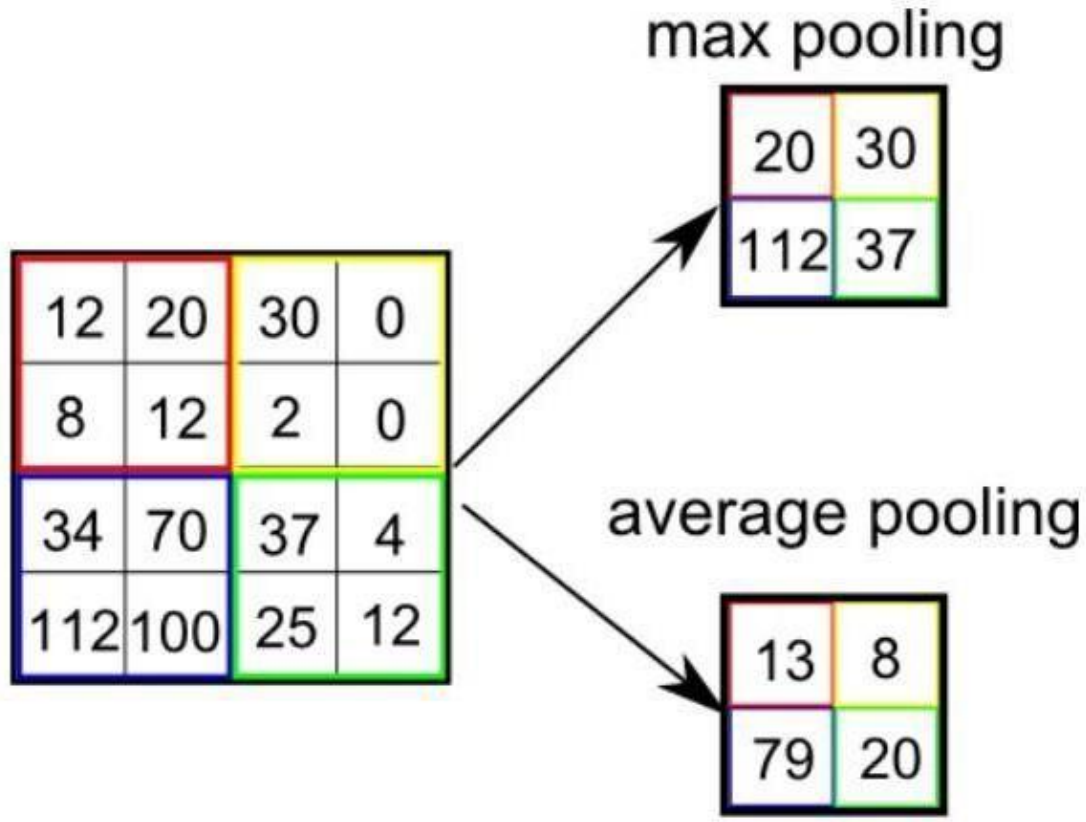
Convolution Layer



Max Pooling

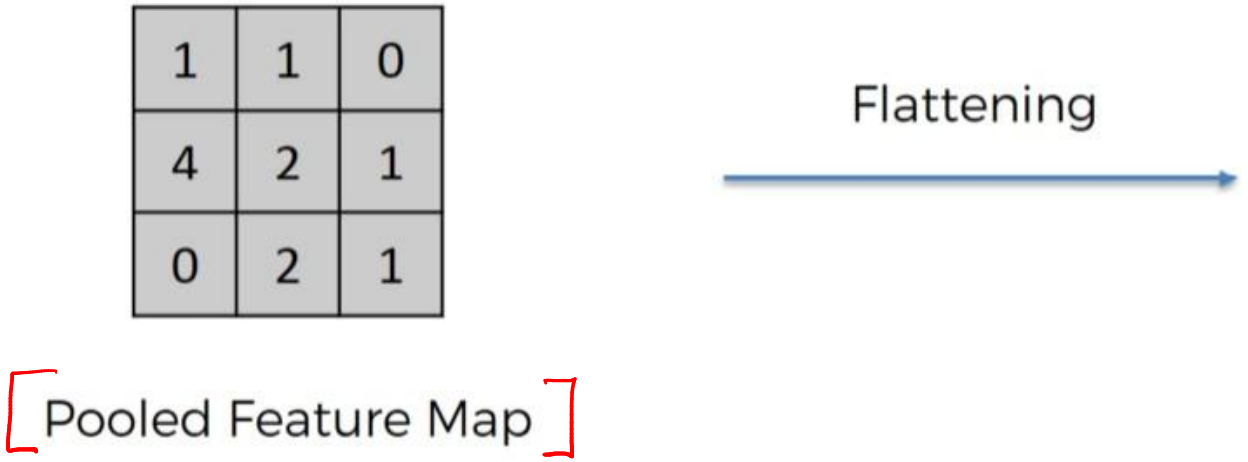
- Pooling layer is responsible for reducing the spatial size of the Convolved Feature
- Dimension = number of columns
- This is to decrease the computational power required to process the data through dimensionality reduction
- Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model
- There are two types of Pooling
  - Max Pooling
    - Max Pooling returns the maximum value from the portion of the image covered by the Kernel
    - It also performs as a Noise Suppressant
    - It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction
  - Average Pooling
    - Average Pooling returns the average of all the values from the portion of the image covered by the Kernel
    - It performs dimensionality reduction as a noise suppressing mechanism

Max Pooling


max pooling
20 30
112 37
12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
average pooling
13 8
79 20

A feature map is divided into small regions (usually 2×2 windows), and a pooling operation is applied to each region to reduce the size of the feature map.

Flattening



The pooled feature map (2D matrix) is converted into a 1D vector

This process is called Flattening

Before flattening (2D pooled feature map):

1	1	0
4	2	1
0	2	1

After flattening (1D vector):

1
1
0
4
2
1
0
2
1

Fully Connected (Dense) layers accept only 1D input

Flattening prepares the data for:

classification
decision making

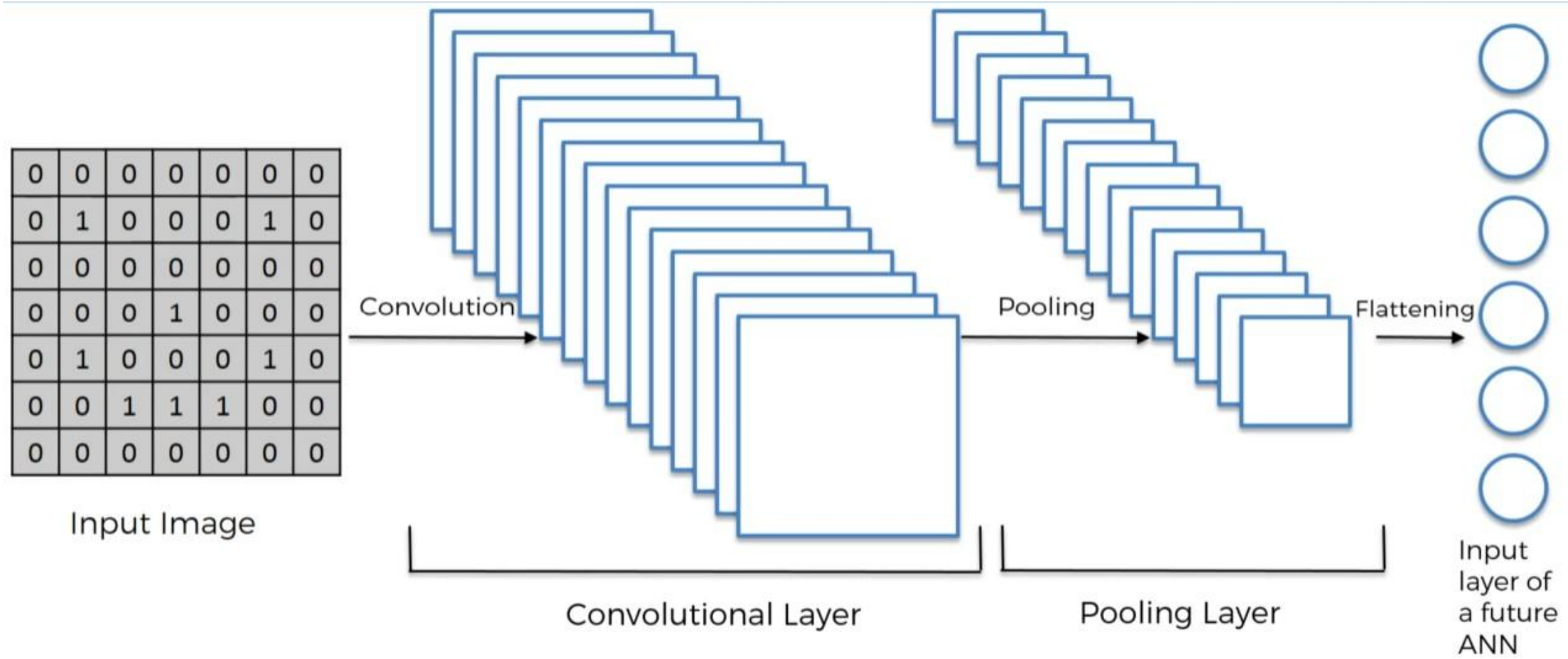
Convolution → Pooling → Flattening → Fully Connected Layer → Output

Flattening does not change values

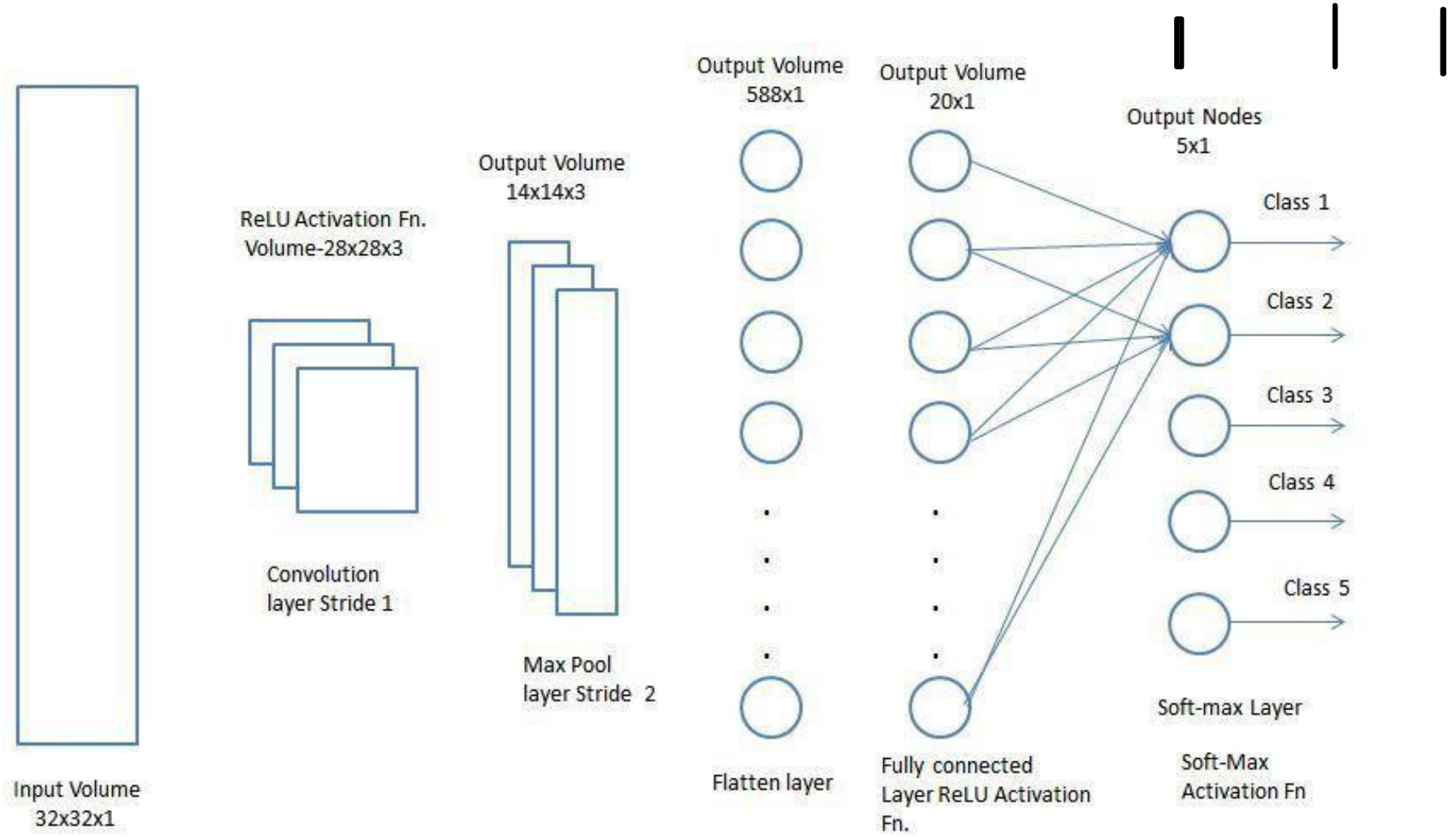
It only changes the shape

No learning or parameters involved

Flattening

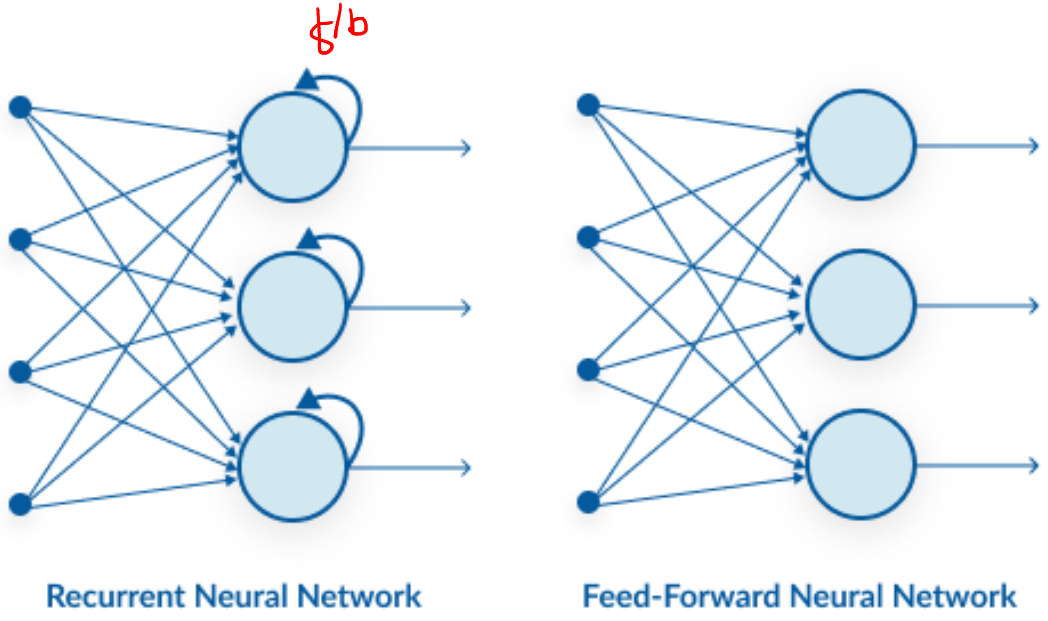

0 0 0 0 0 0 0
0 1 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 1 0
0 0 1 1 1 0 0
0 0 0 0 0 0 0
Input Image
Convolution
Pooling
Flattening
Convolutional Layer
Pooling Layer
Input layer of a future ANN

Full Connection



Recurrent Neural Networks(RNN)

- A looping constraint on the hidden layer of ANN turns to RNN.
- We can use **Recurrent Neural Networks** to solve the problems related to:
  - Time Series data
  - Text data
  - Audio data



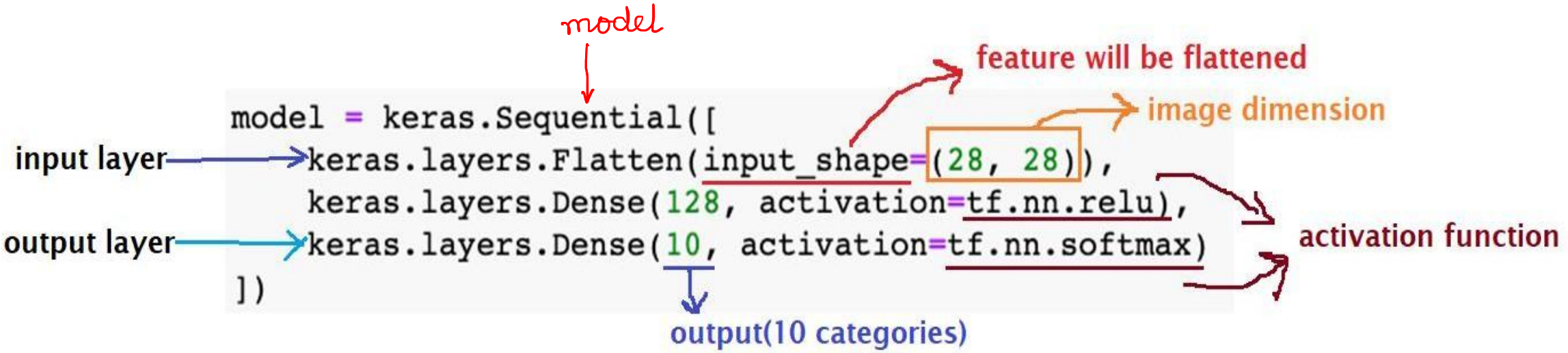
pre trained model or train on it own

Using TensorFlow

Designing the Neural Network

API on top of tensorflow

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```



Designing the Neural Network

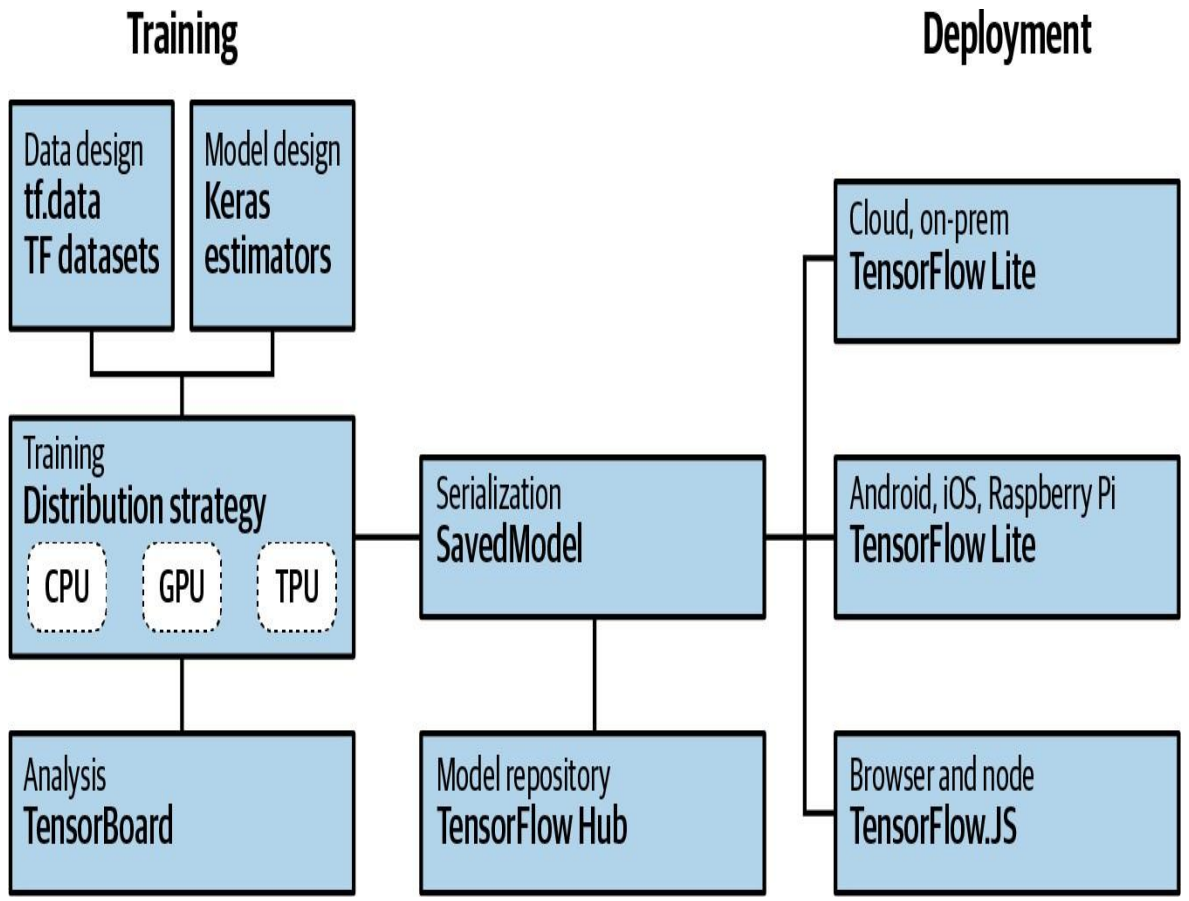
- The first, Flatten, isn't a layer of neurons, but an input layer specification
- Our inputs are 28 × 28 images, but we want them to be treated as a series of numeric values
- Flatten takes the image (a 2D array) and turns it into a line (a 1D array)

- Finally, there's another Dense layer, which is the output layer which has 10 neurons, because we have 10 classes
- Each of these neurons will end up with a probability that the input pixels match that class, so our job is to determine which one has the highest value

- The next one, Dense (hidden layers), is a layer of neurons, and we're specifying that we want 128 of them
- More neurons means it will run more slowly, as it has to learn more parameters
- It takes some experimentation over time to pick the right values. This process is typically called hyperparameter tuning

TensorFlow

- TensorFlow is an open source platform for creating and using machine learning models backed by Google.
- It implements many of the common algorithms and patterns needed for machine learning, saving you from needing to learn all the underlying math and logic and enabling you to just to focus on your scenario
- It's aimed at everyone from hobbyists, to professional developers, to researchers pushing the boundaries of artificial intelligence
- Importantly, it also supports deployment of models to the web, cloud, mobile, and embedded systems



Saved model format is h5)

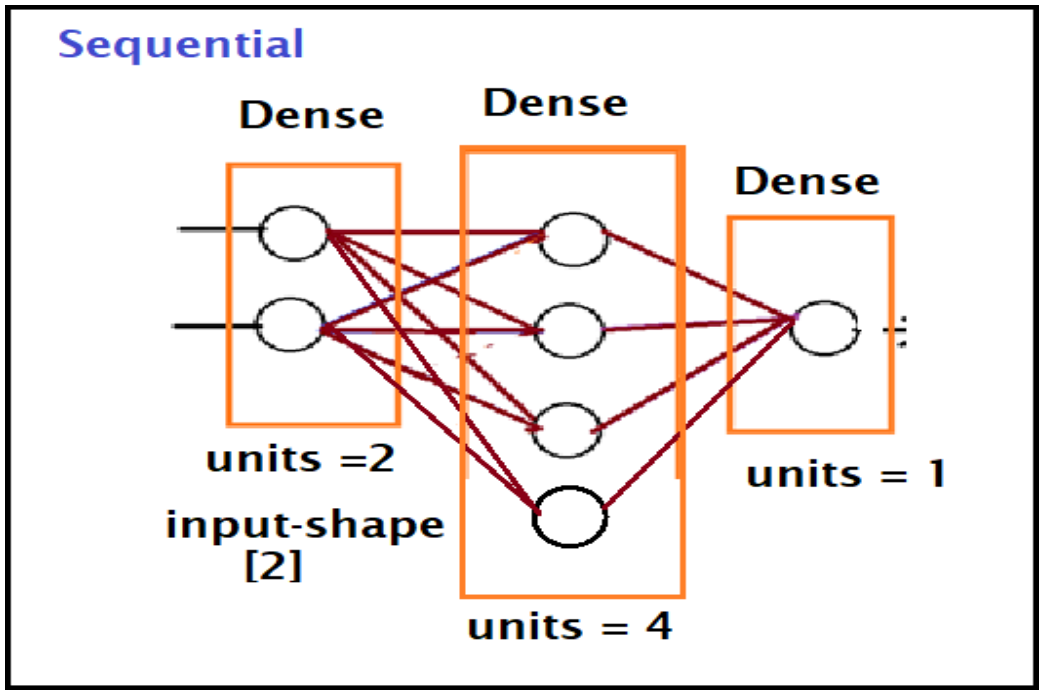
First Program using Tensorflow

```
import tensorflow as tf
import numpy as np
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

neoron=1

input values are passed as 1D array

MSE



Sequential = Model

Dense = Layer

Units = neurons

X	-1	0	1	2	.....	10
Y	-3	-1	1	3	.....	?

weights

Y = 2*x -1

y = 2*10 -1

y = 19

First Program Explained

- Sequential (Container , ANN)
  - When using TensorFlow, you define your layers using Sequential
  - Inside the Sequential, you then specify what each layer looks like
- Layer
  - A layer is represented as Dense in TensorFlow
  - “Dense” means a set of fully (or densely) connected neurons where where every neuron is connected to every neuron in the next layer
- Compiling model
  - The computer starts guessing to create the model (formula) and comparing with the observed value
  - The optimizer used in this stage helps the machine to optimize the guess
    - SGD: stochastic gradient descent, a complex mathematical function that, when given the values, the previous guess, and the results of calculating the errors (or loss) on that guess, can then generate another one
    - ADAM:the optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments

Thank You!!