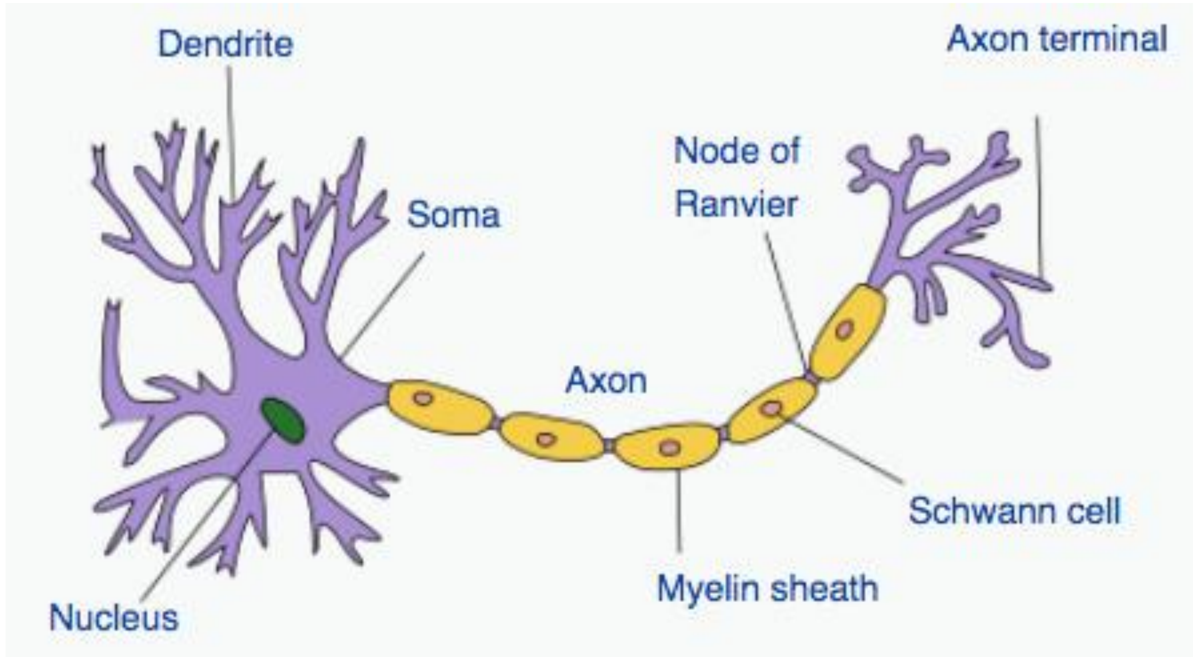
Embedded AI

Dr.Akshita Chanchlani

Topic : Neural Network

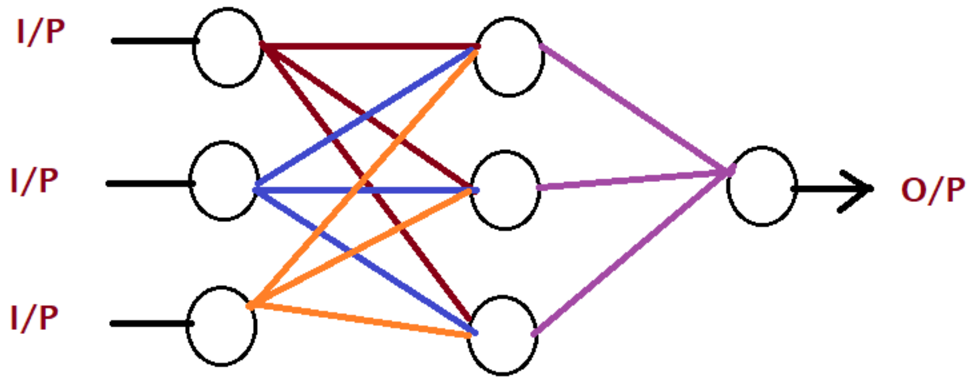
Neuron

- 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

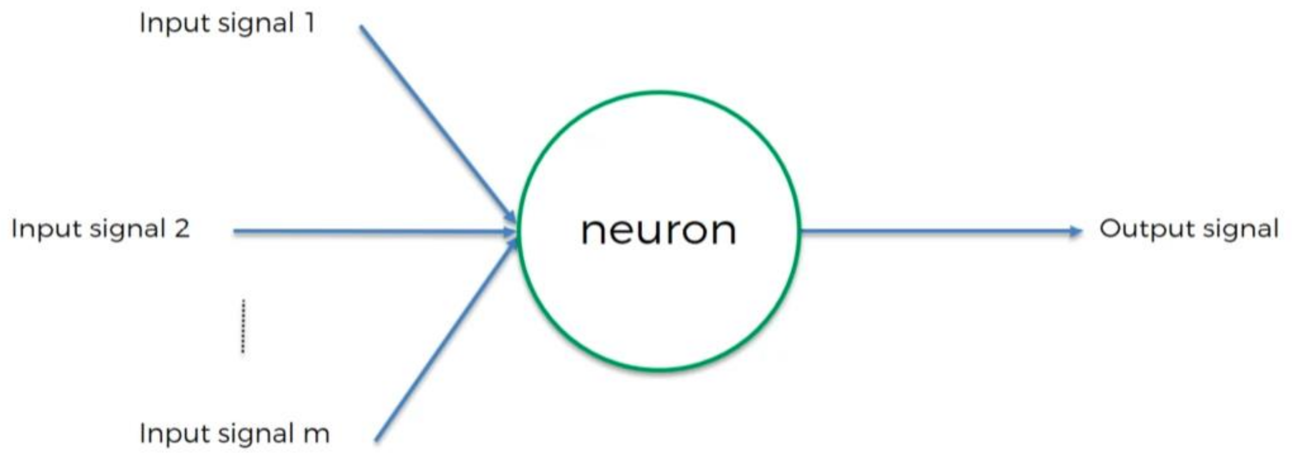


Artificial Neural Network

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

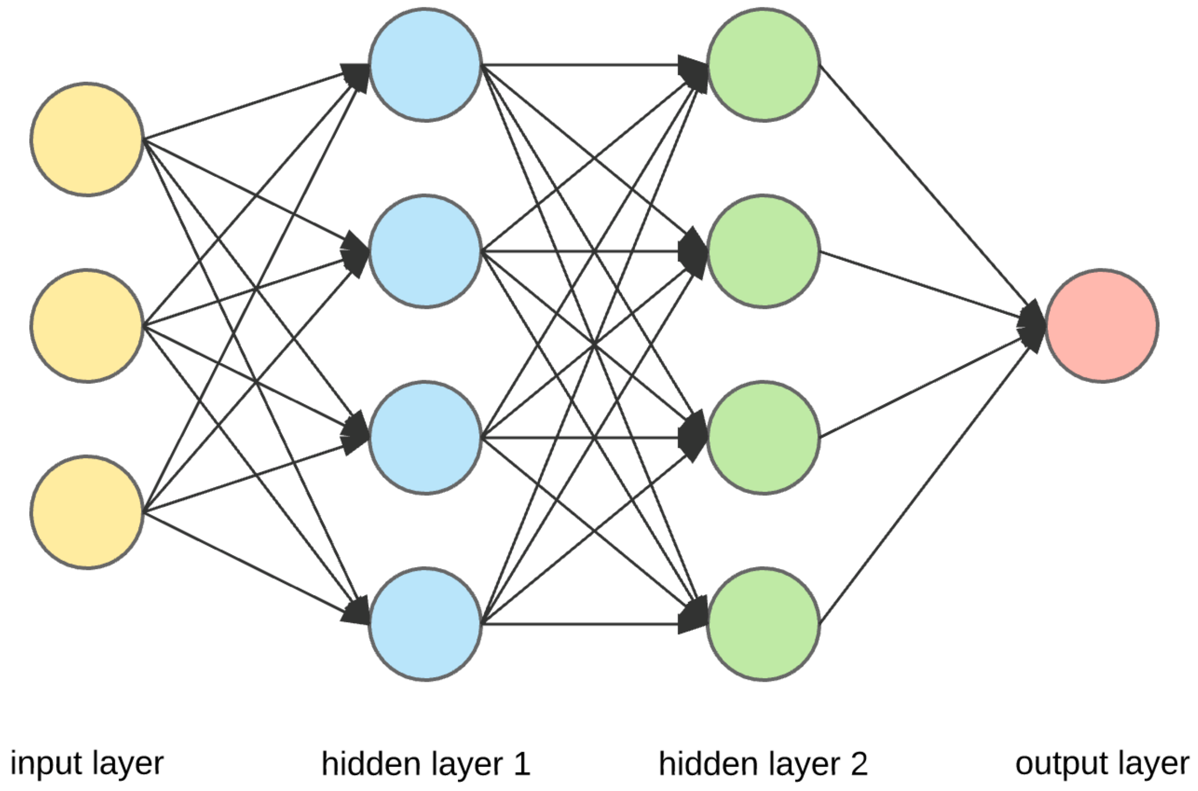


ANN



Artificial Neural Network

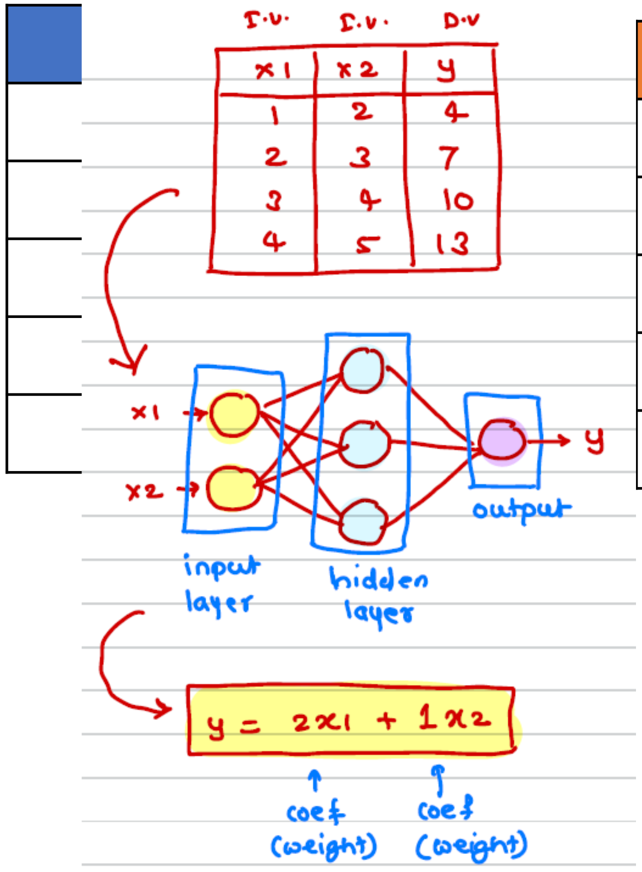
- Artificial Neural Networks (ANN) are **multi-layer** fully-connected neural nets
- They consist of an **input layer**, multiple hidden layers, and an **output layer**
- Every node in one layer is connected to every other node in the next layer
- We can make the network deeper by increasing the number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



Relu sigmoid

Characteristics of Artificial Neural Network

- It is neurally implemented mathematical model
- 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 adjustment of weights.
- The collective behavior of the neurons describes its computational power, and no single neuron carries specific information



I.V.	I.V.	D.V
x1	x2	y
1	2	4
2	3	7
3	4	10
4	5	13

x1
x2
y
output
input layer
hidden layer

$y = 2x_1 + 1x_2$

coef (weight)
coef (weight)

(weight)

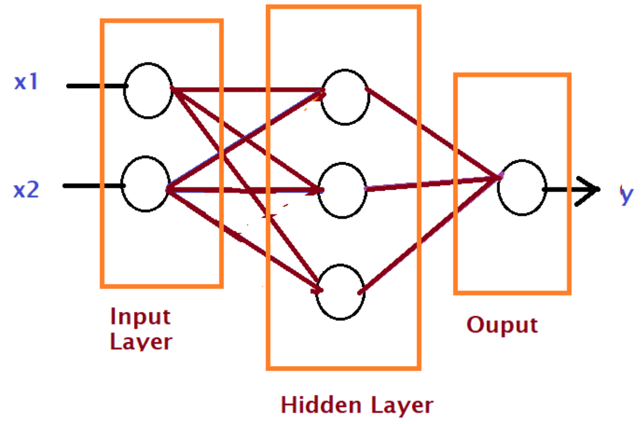
X	Y
1	2
2	4
3	6
4	8
16	?

$Y = 2x$

$Y = 2 * x$

Coefficient (weight)

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



$Y = 2*x1 + 1*x2$

Coefficient (weight)

Coefficient (weight)

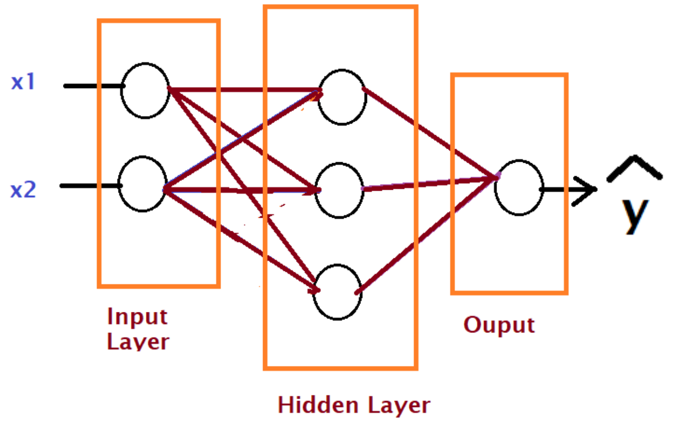
Trial (EPOCH) : 1

Y = 1*x1 + 1 * x2
y = 1+2 = 3
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

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

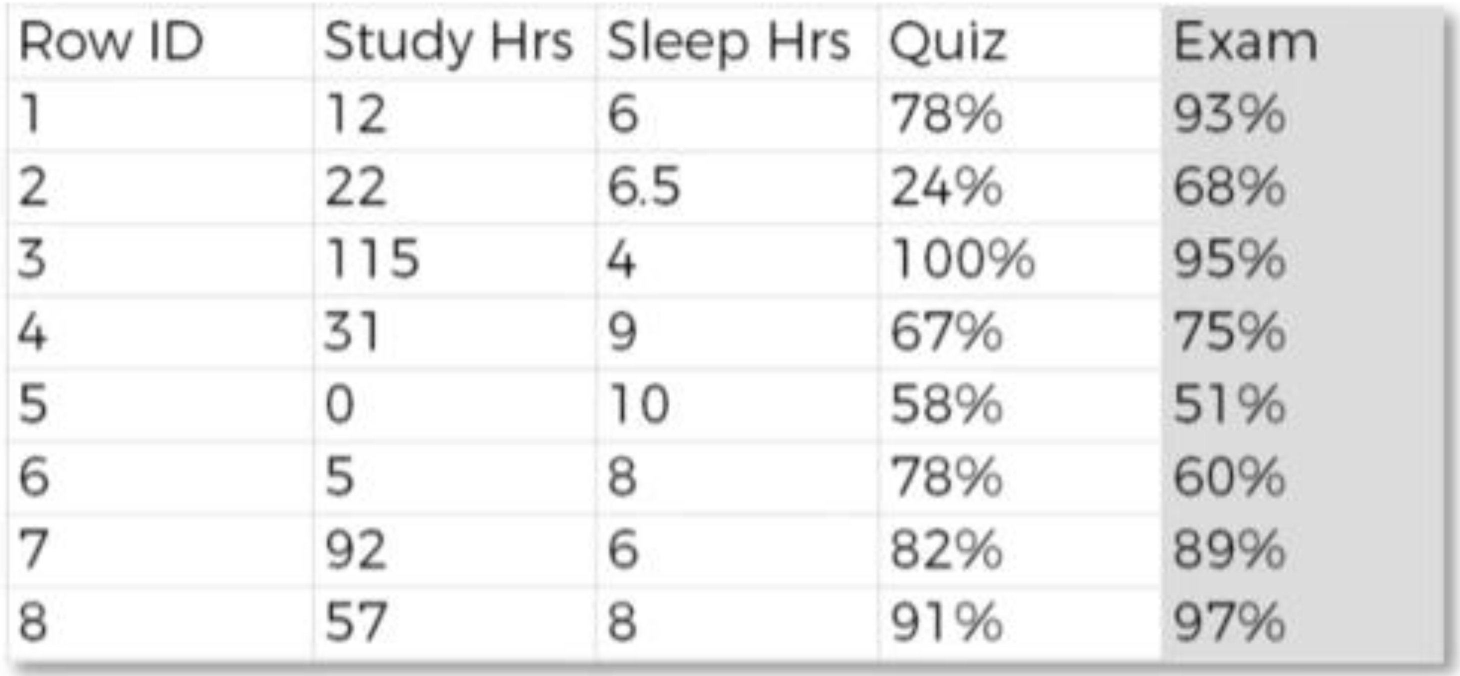
Coefficient (weight)

Coefficient (weight)

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

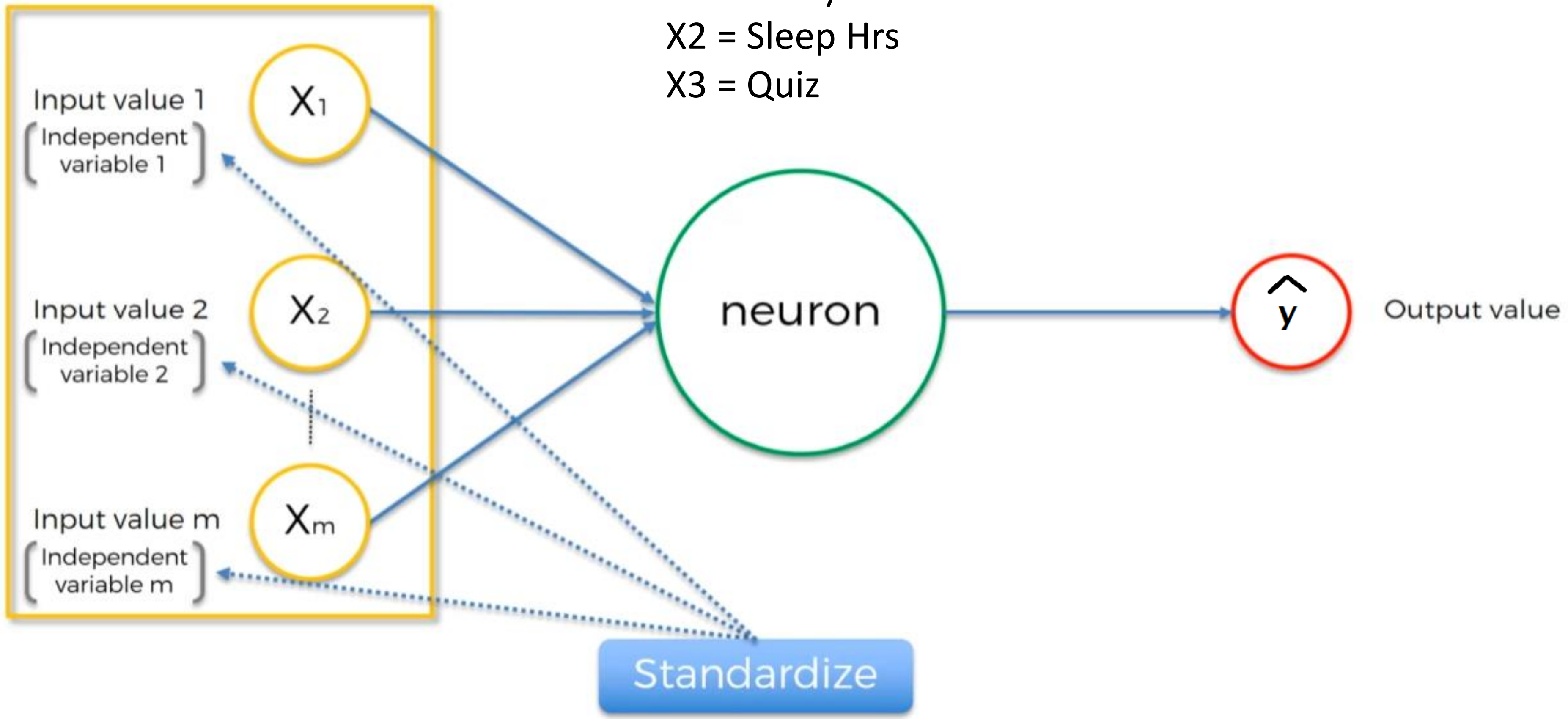
How does it work?



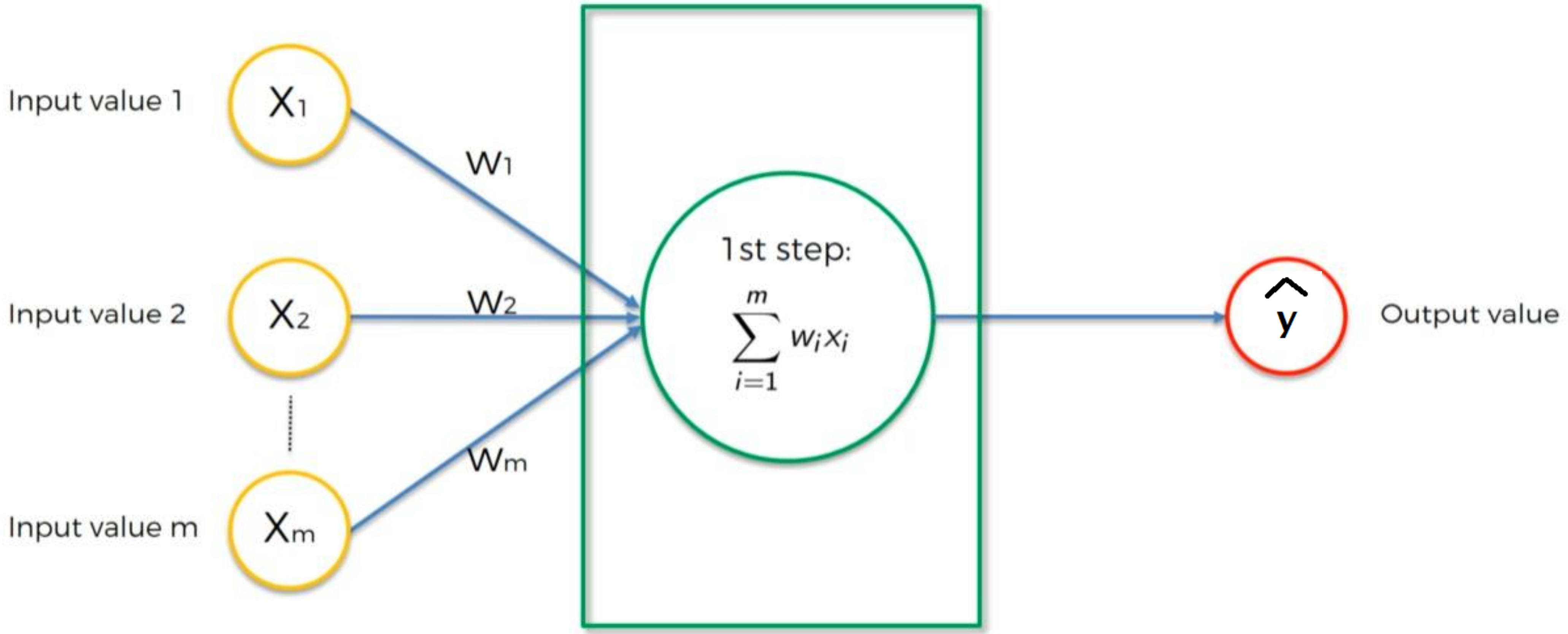
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

Neuron

X1 = Study Hrs
X2 = Sleep Hrs
X3 = Quiz

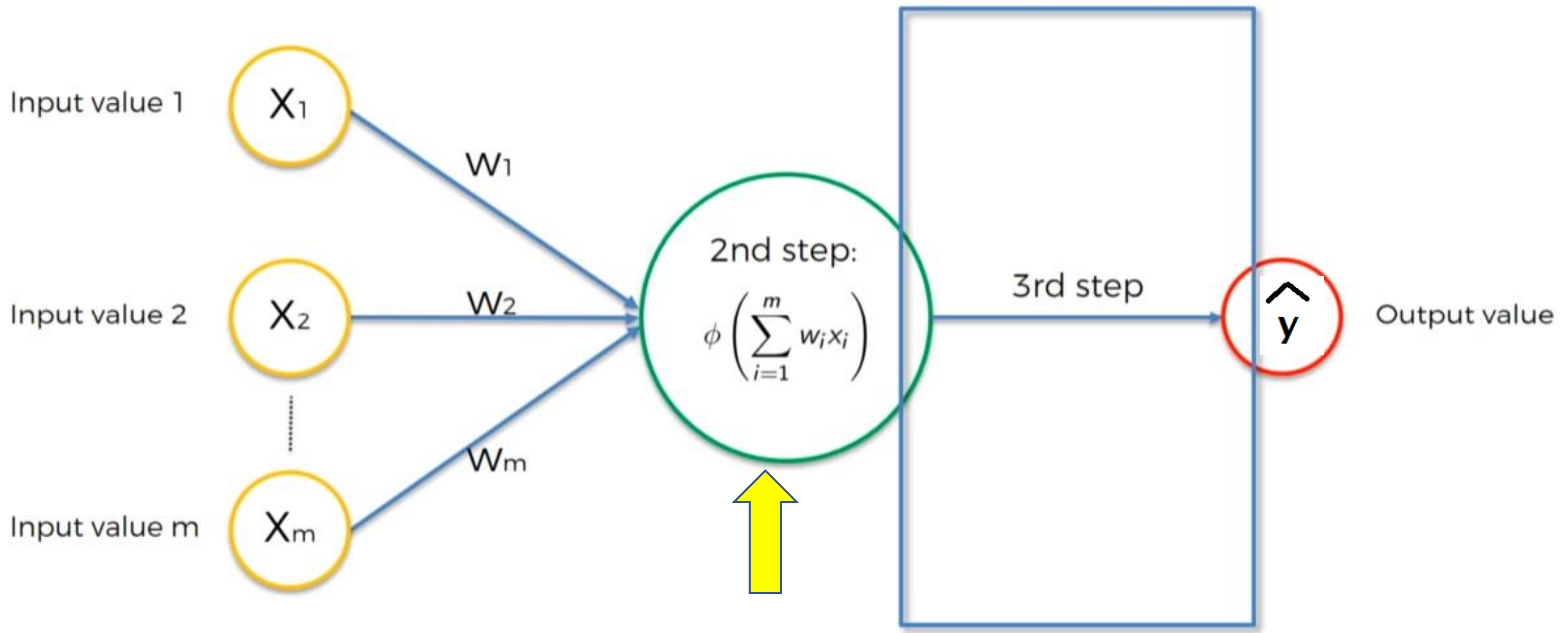


Neuron (Perceptron)



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

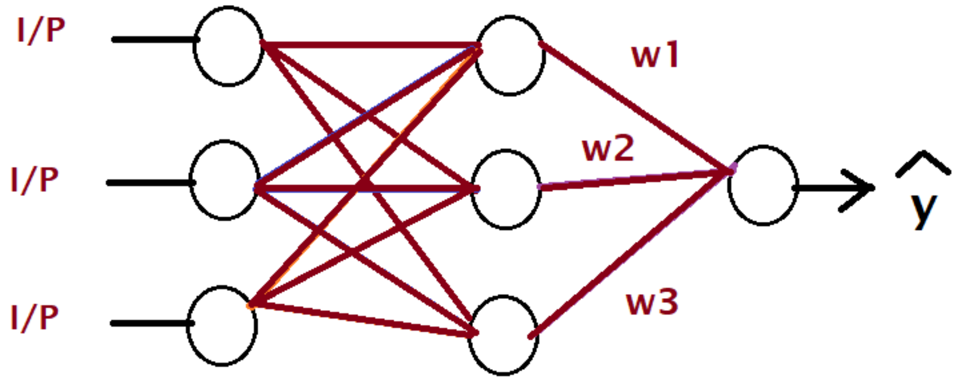
Activation Function



ϕ = **Activation function**

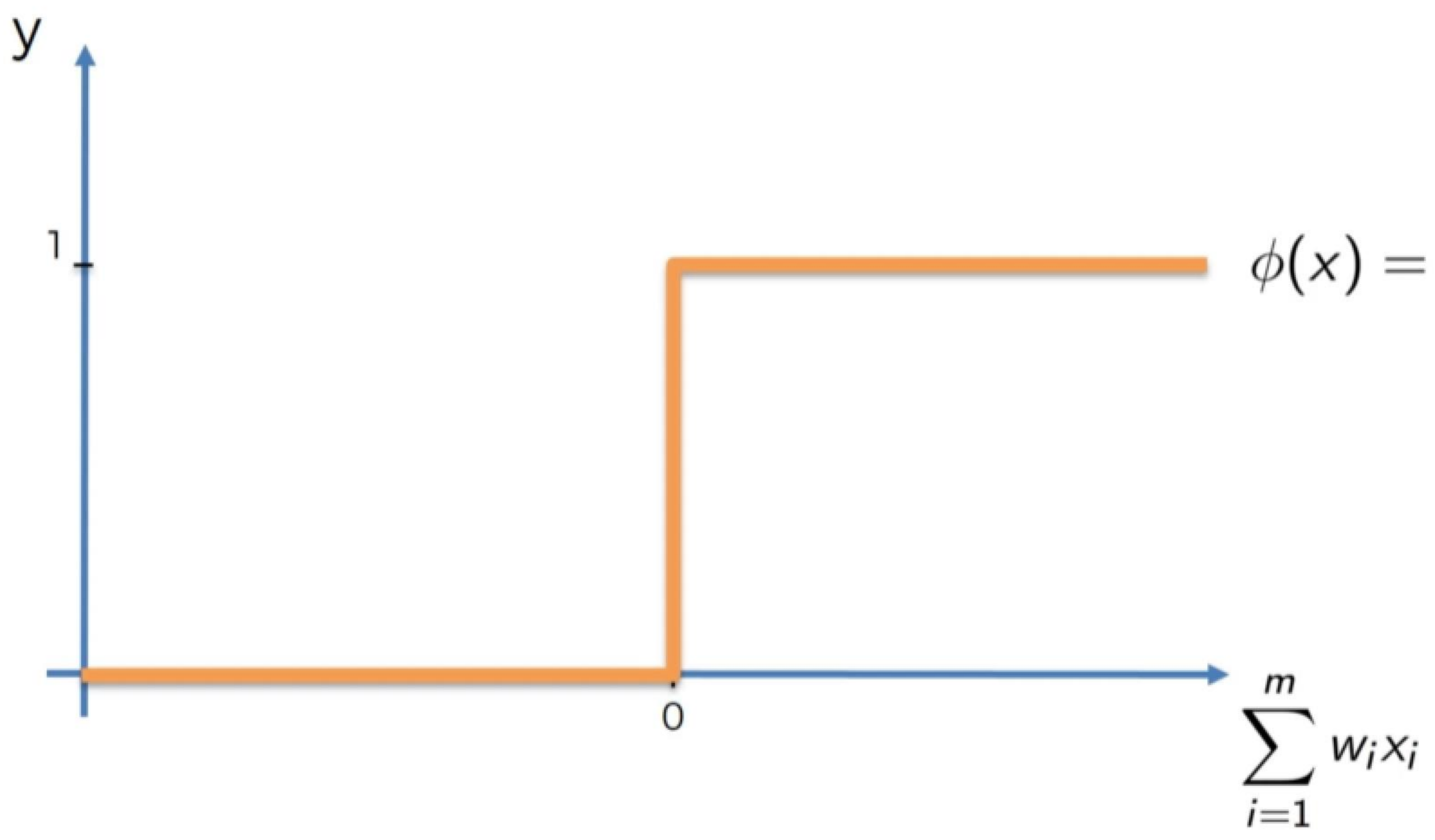
Activation Function

- 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



Activation Function

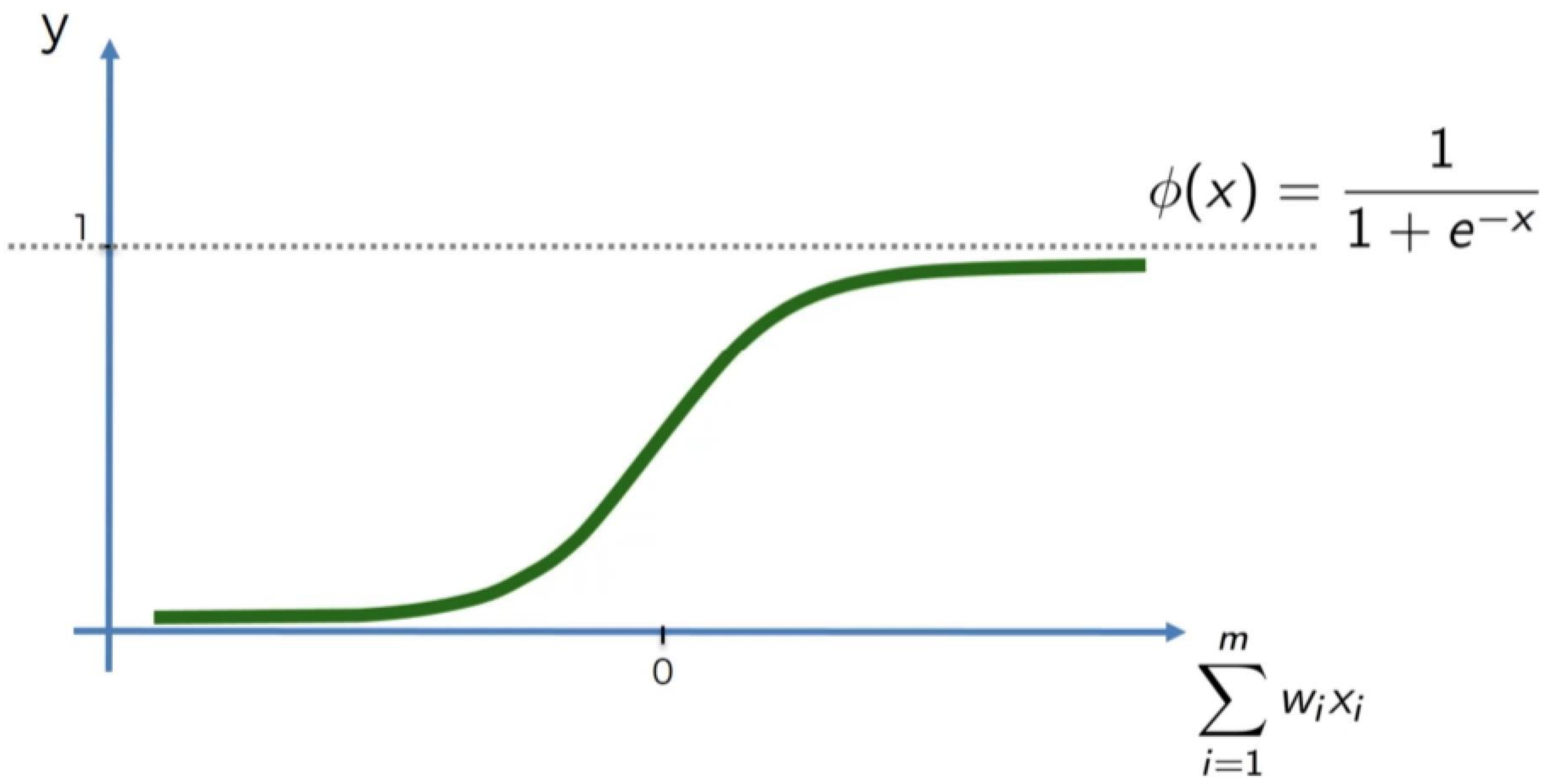
Threshold Function



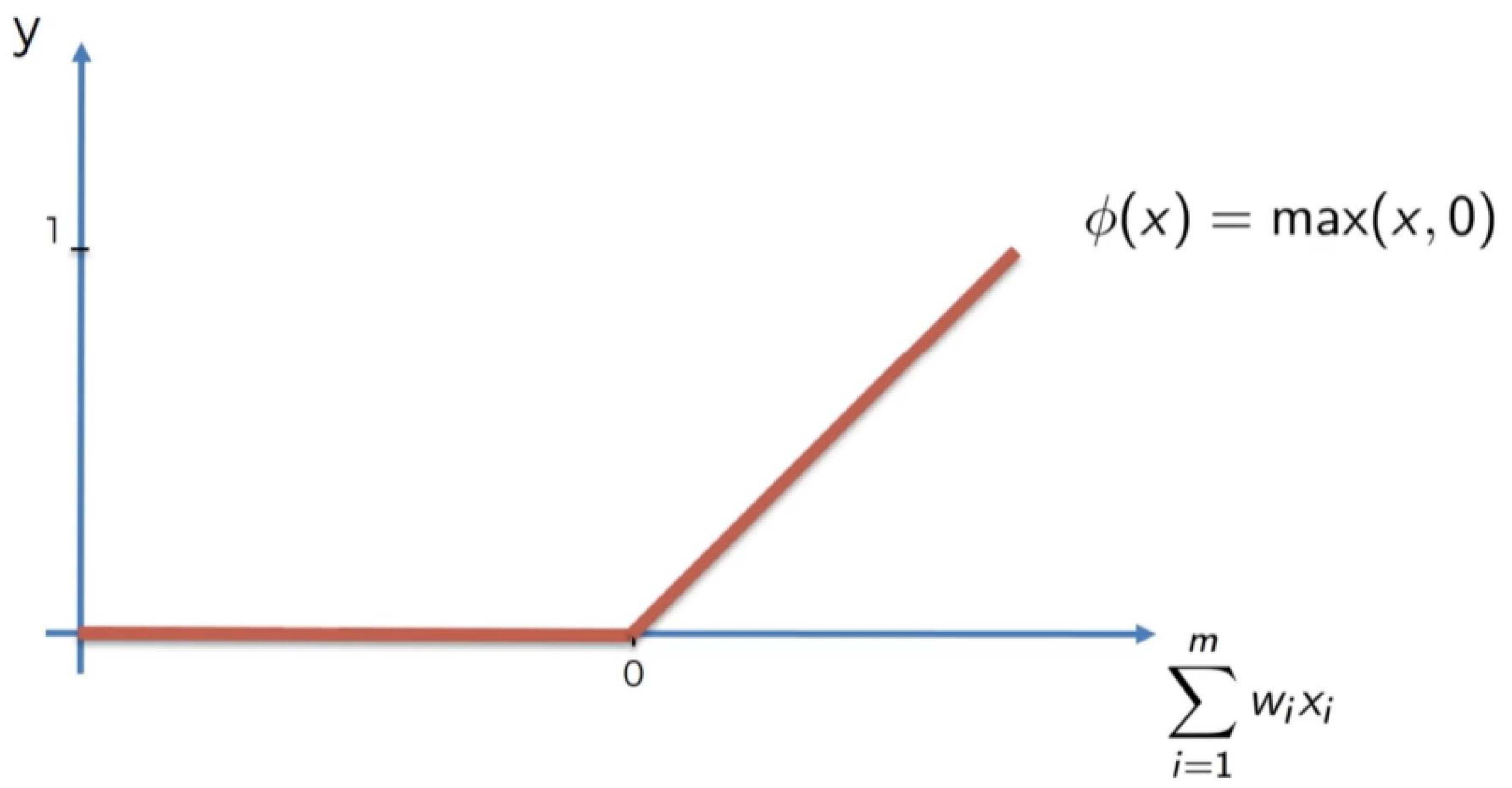
$$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

$$\sum_{i=1}^{m} w_i x_i$$

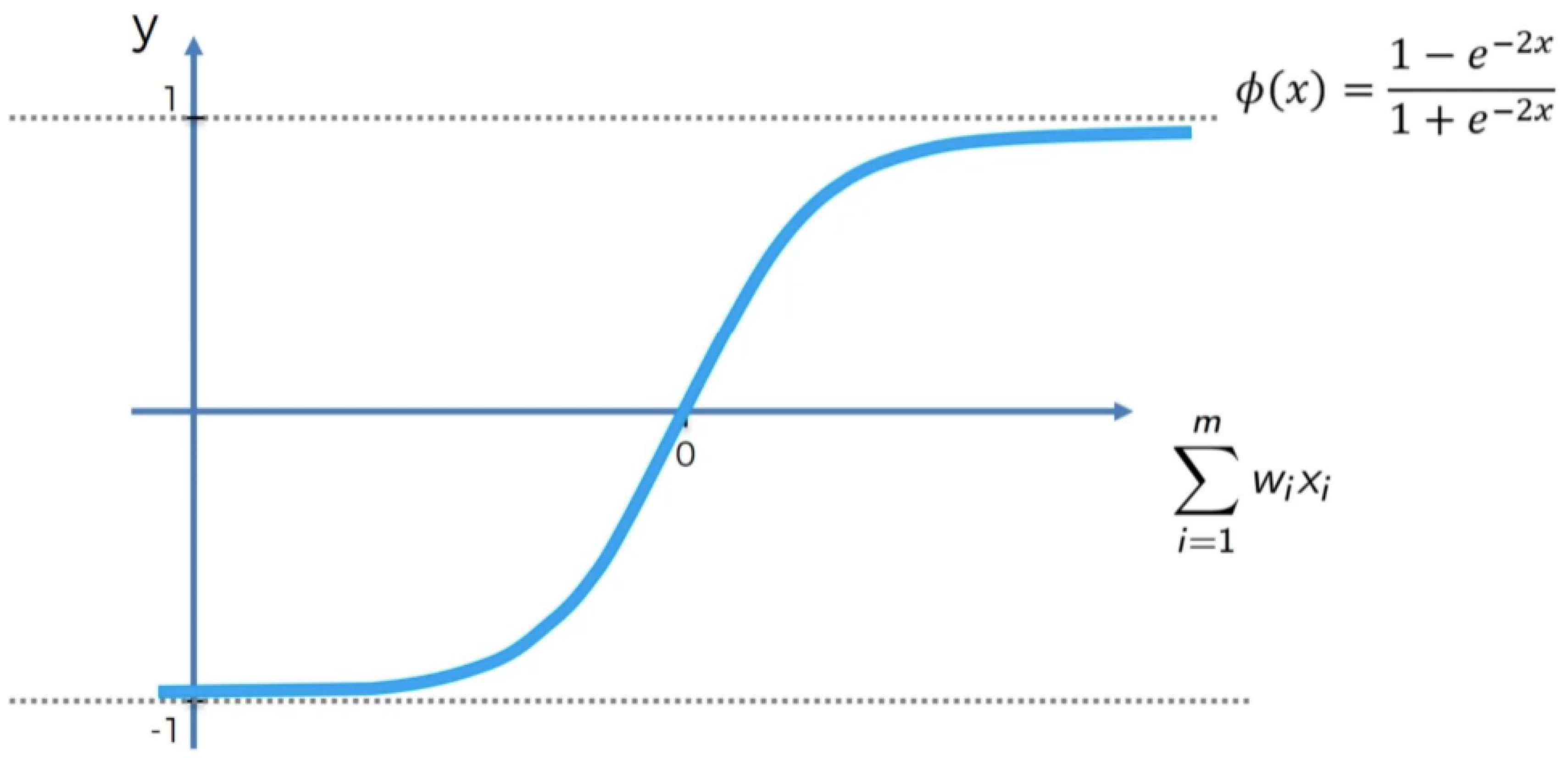
Sigmoid Function



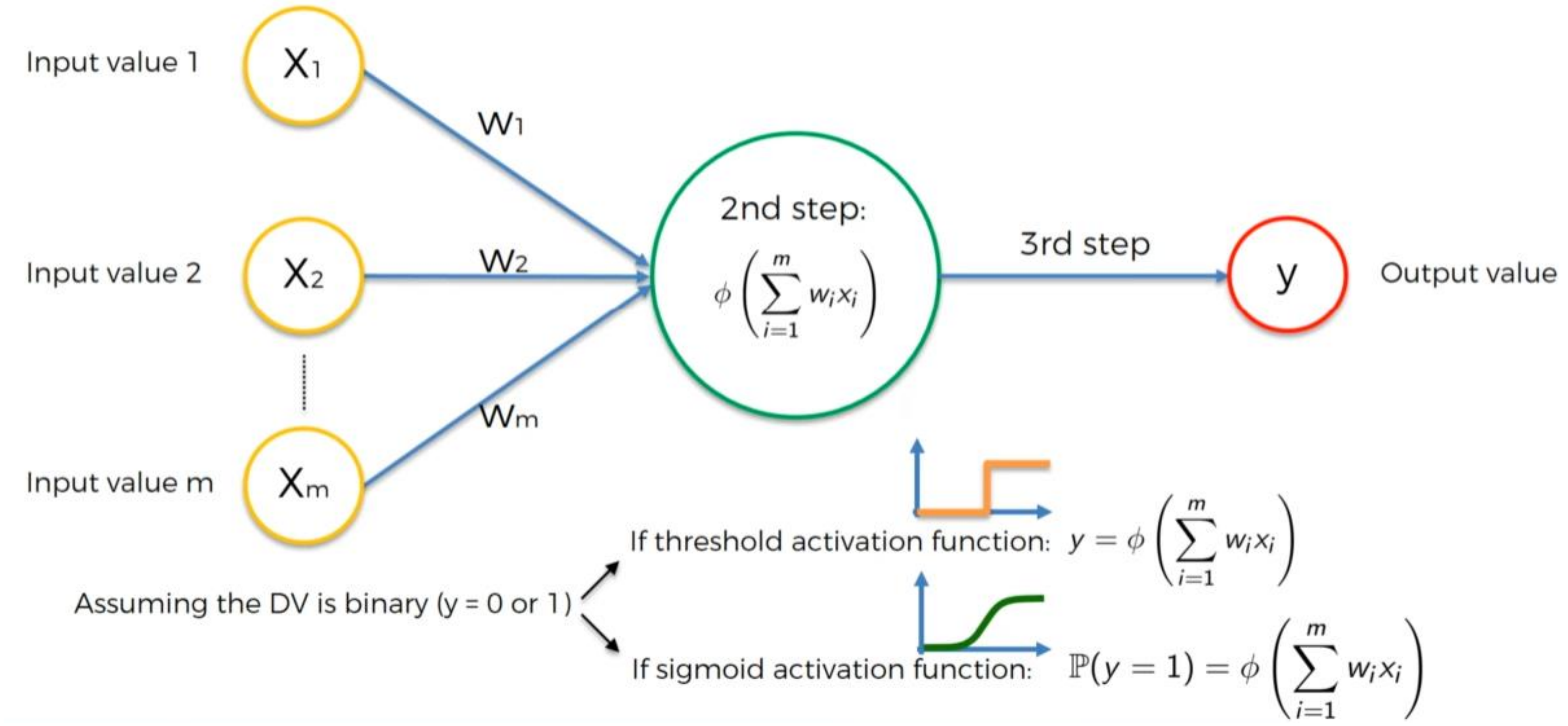
Rectifier Function



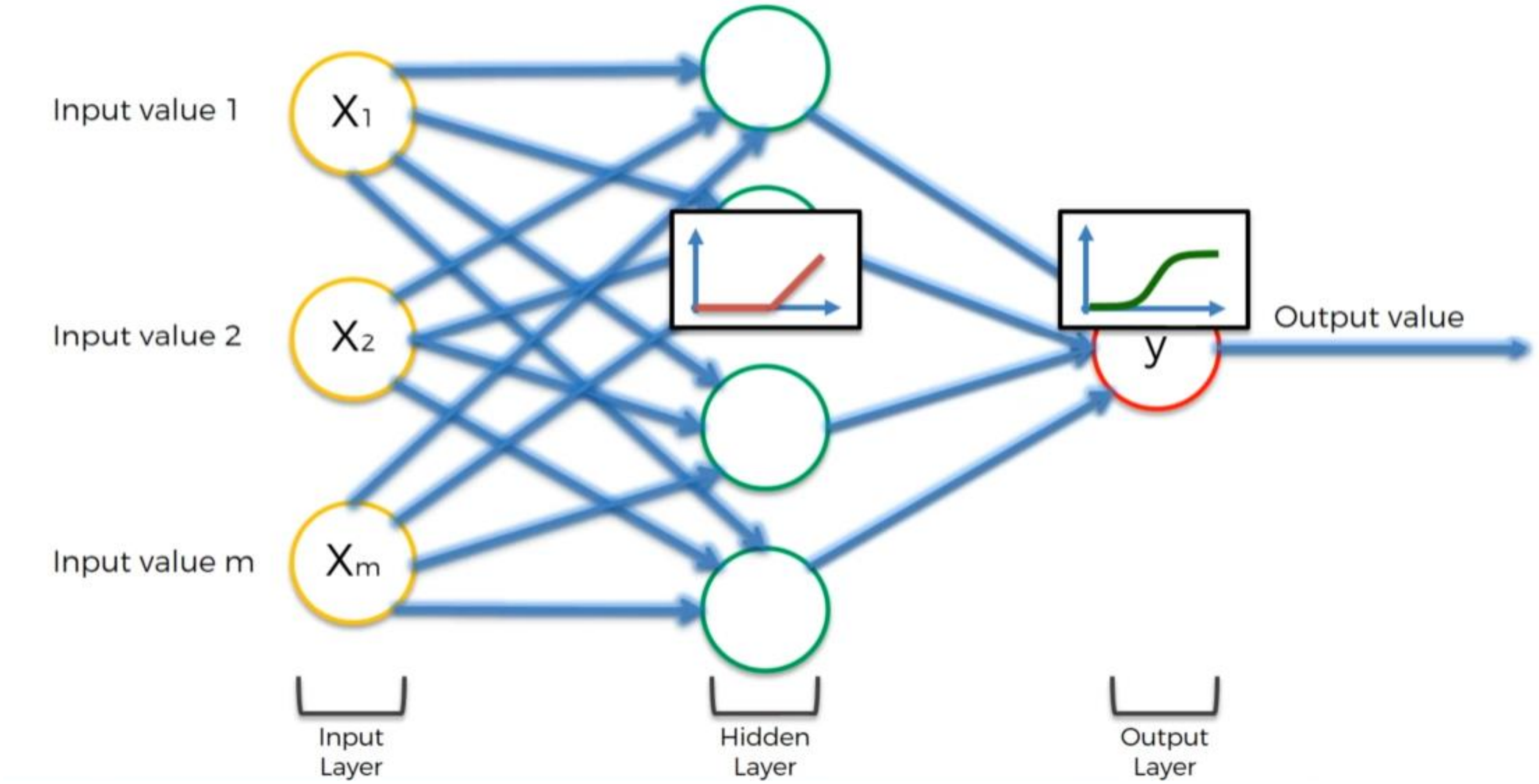
Hyperbolic Tangent Function



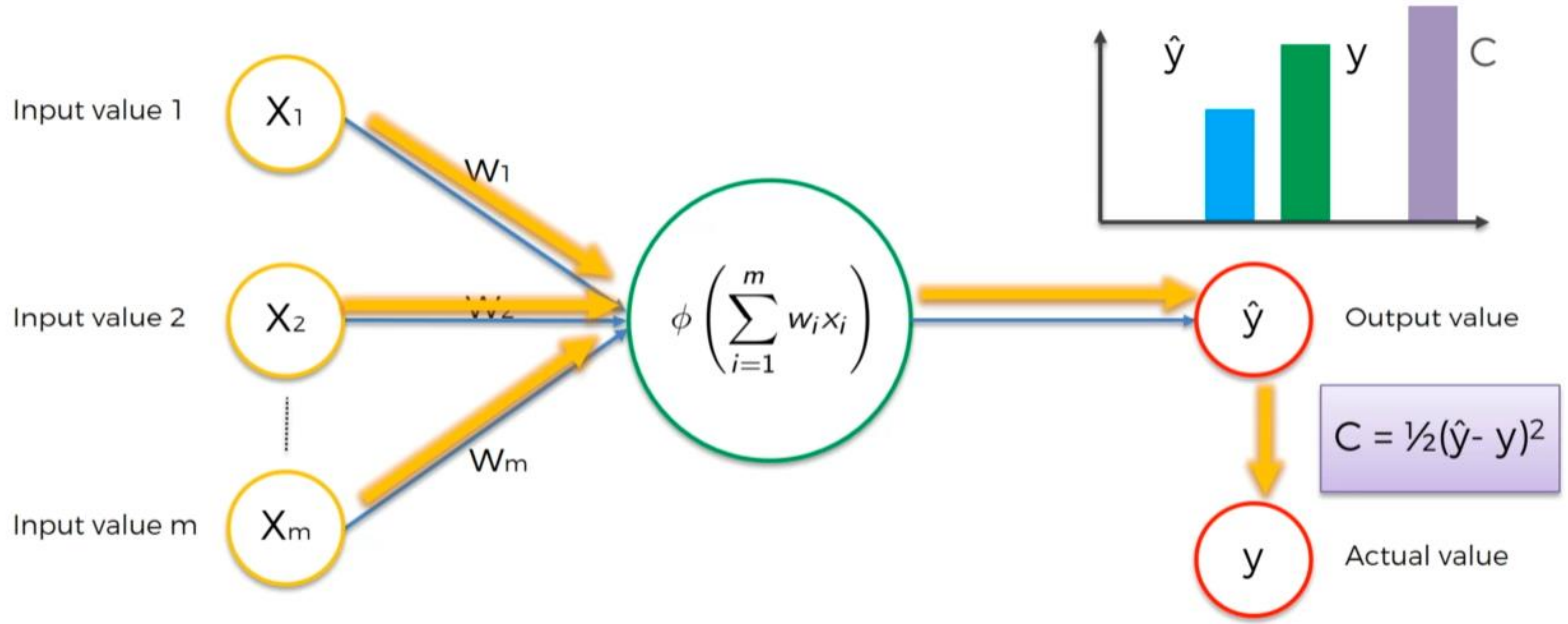
Activation Function


Input value 1
X_1
w_1
Input value 2
X_2
w_2
Input value m
X_m
w_m
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value
If threshold activation function: $y = \phi\left(\sum_{i=1}^{m} w_i x_i\right)$
Assuming the DV is binary (y = 0 or 1)
If sigmoid activation function: $\mathbb{P}(y=1) = \phi\left(\sum_{i=1}^{m} w_i x_i\right)$

Activation Function

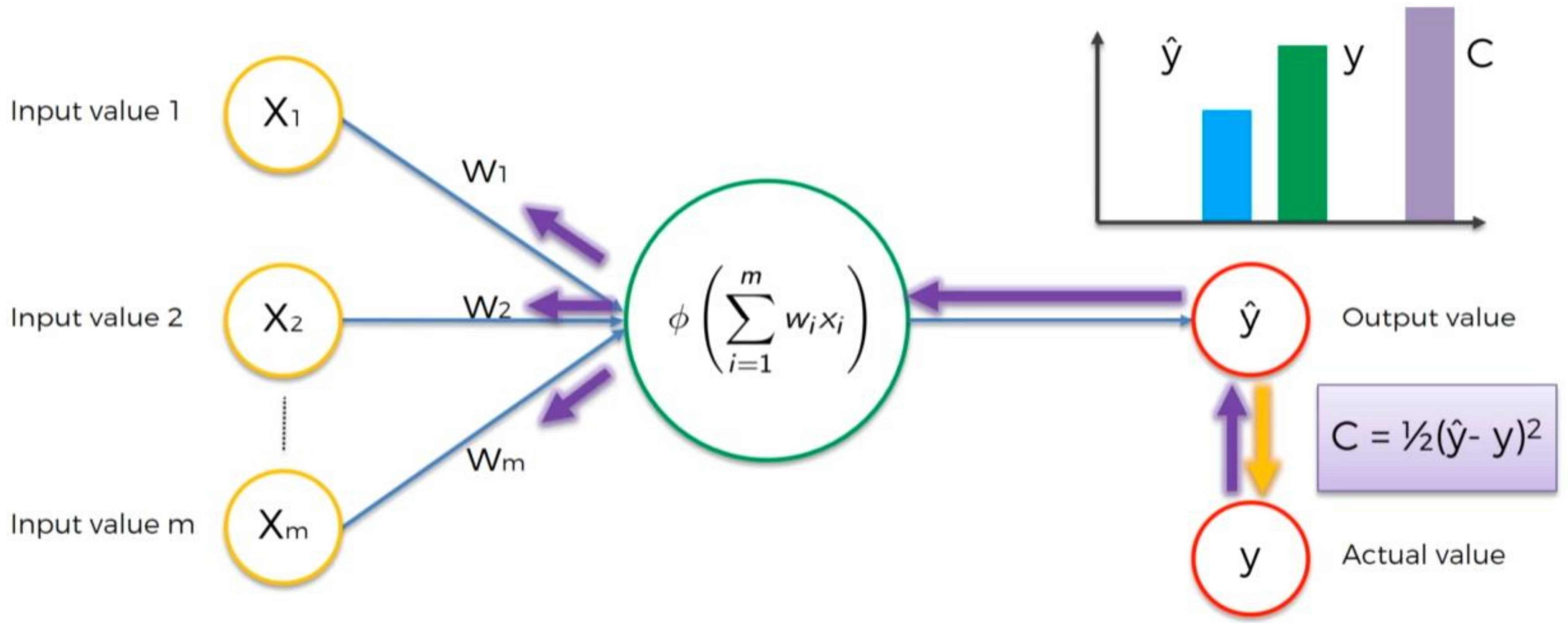


Neural Network - learn



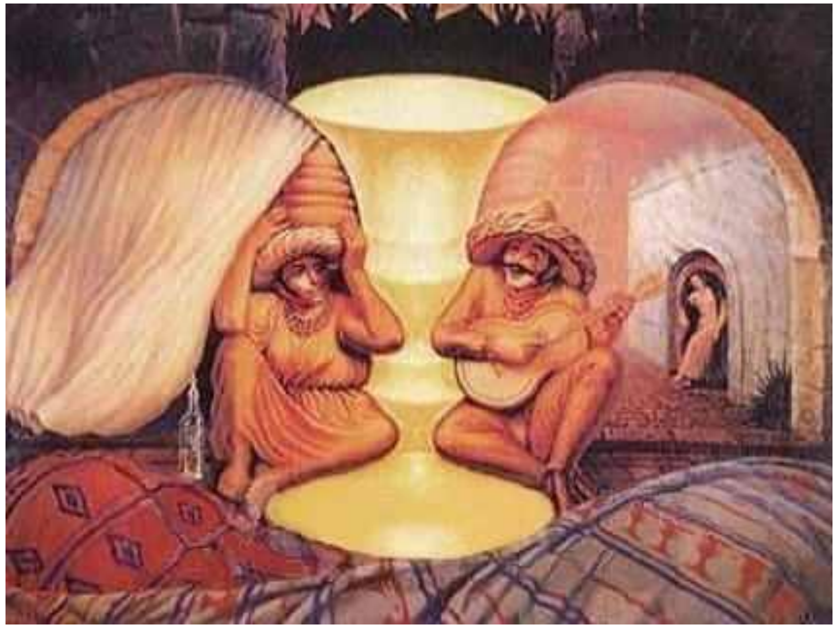
C = Cost Value

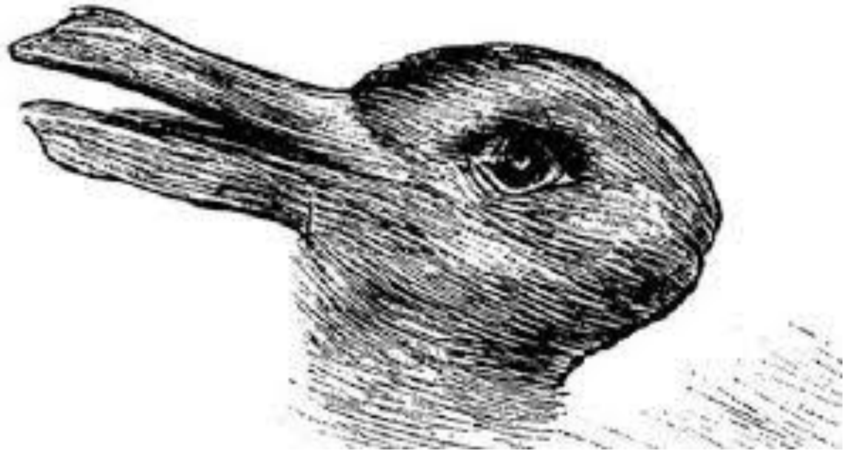
Neural Network - learn



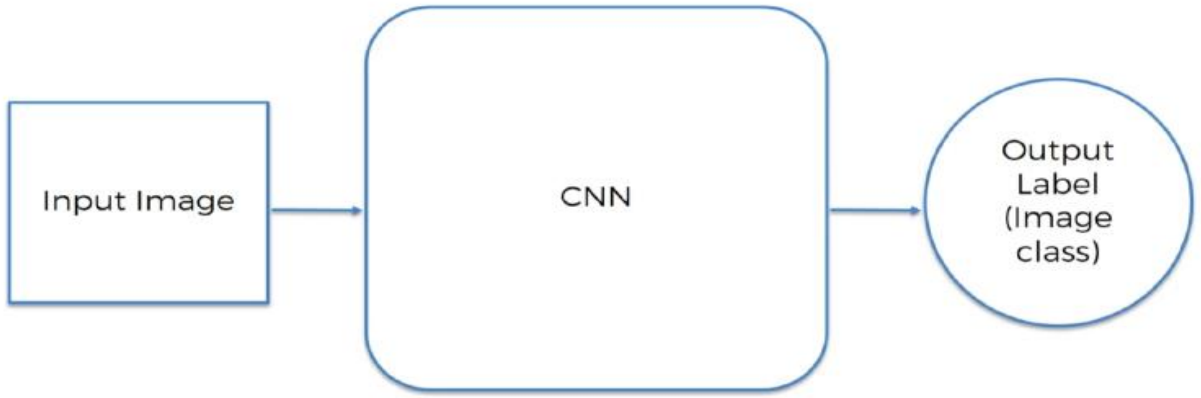
CNN (Convolutional Neural Network)

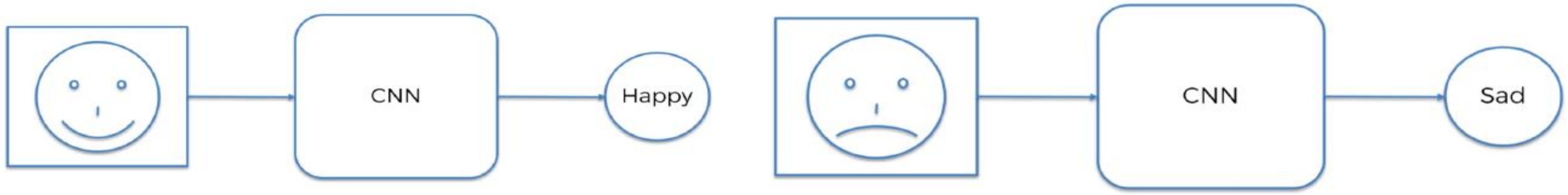
CNN





CNN



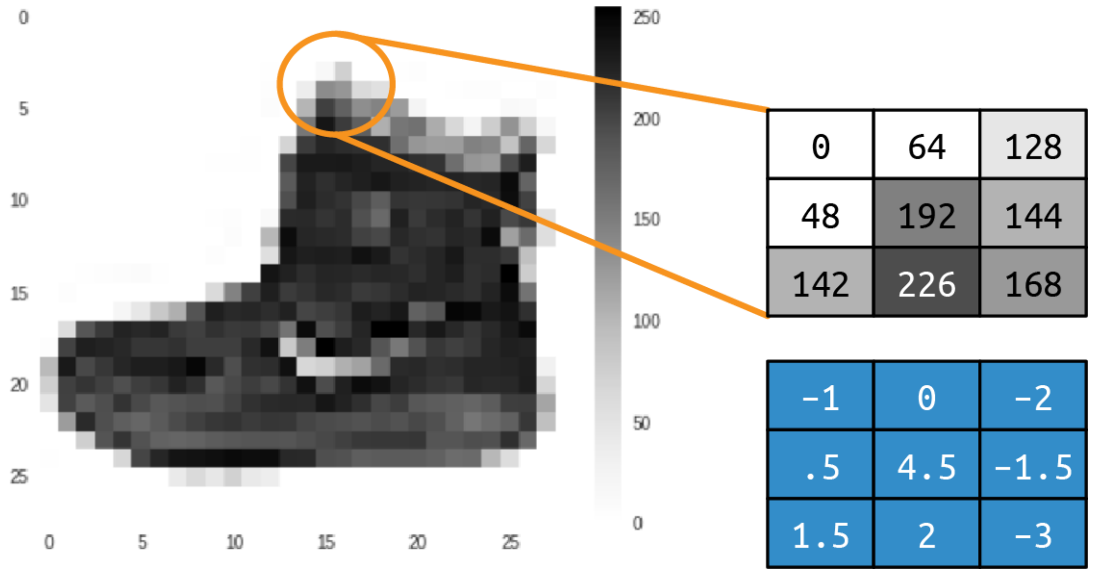


Image

- 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

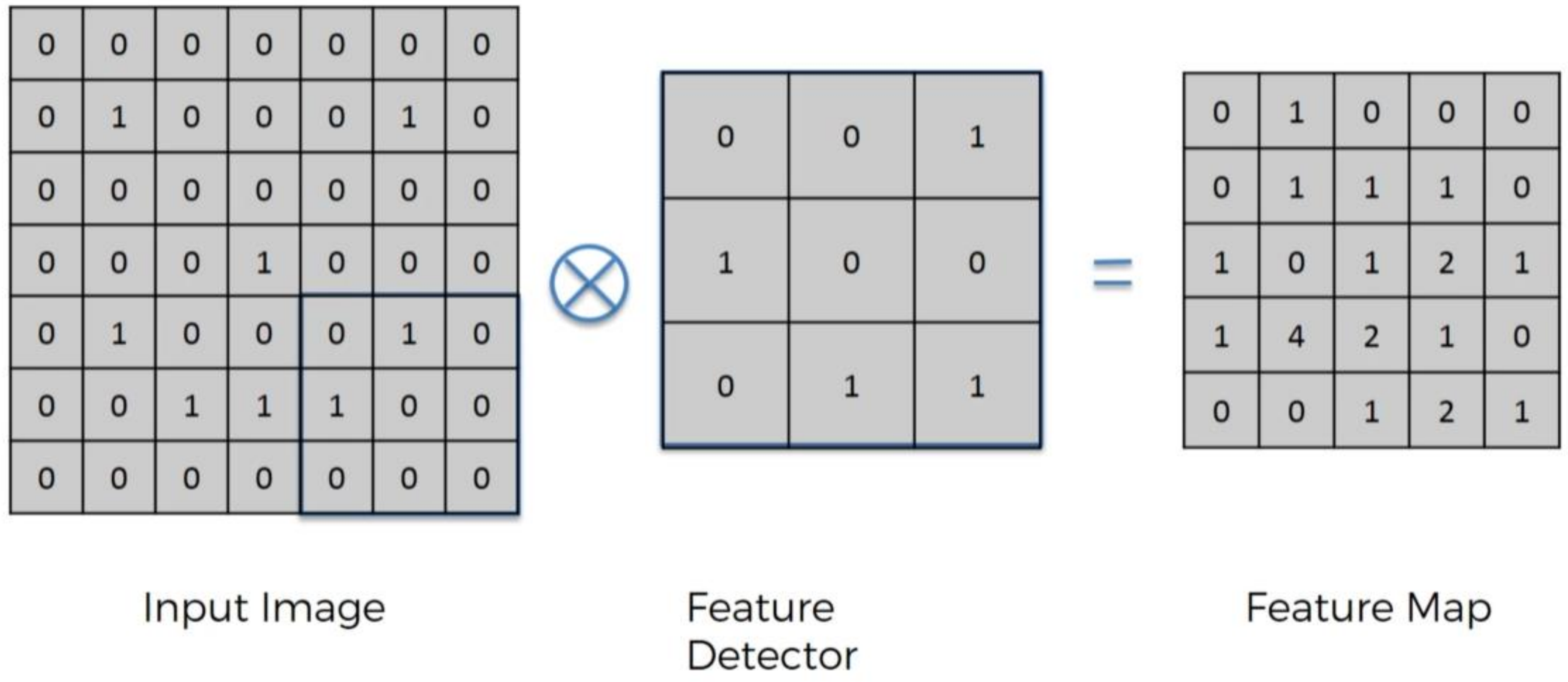
- A convolution is simply a filter of weights that are used to multiply a pixel with its neighbours to get a new value for the pixel
- The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image



0	64	128
48	192	144
142	226	168

-1	0	-2
.5	4.5	-1.5
1.5	2	-3

Convolution



$$\begin{bmatrix} 0&0&0&0&0&0&0\\ 0&1&0&0&0&1&0\\ 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&1&0&0&0&1&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0 \end{bmatrix} \otimes \begin{bmatrix} 0&0&1\\ 1&0&0\\ 0&1&1 \end{bmatrix} = \begin{bmatrix} 0&1&0&0&0\\ 0&1&1&1&0\\ 1&0&1&2&1\\ 1&4&2&1&0\\ 0&0&1&2&1 \end{bmatrix}$$

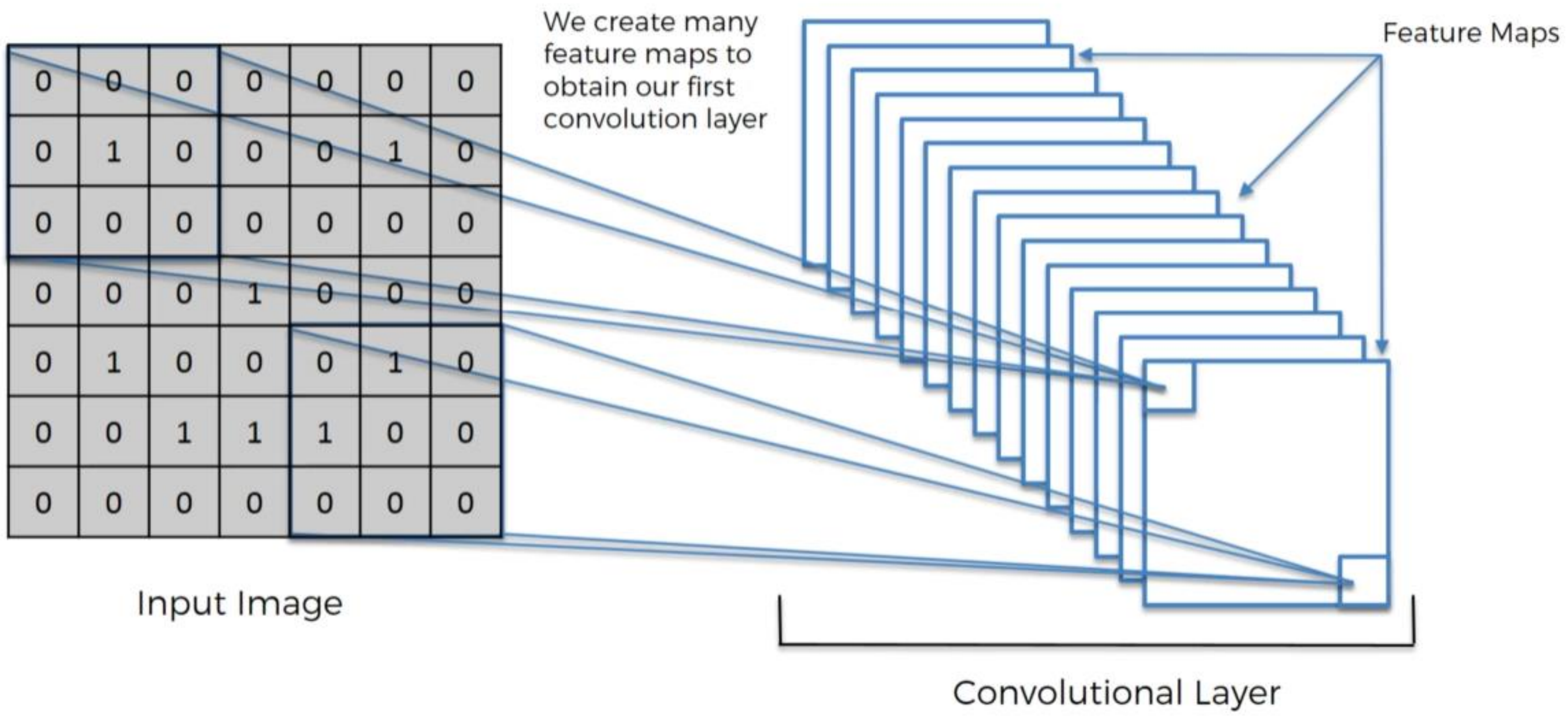
Convolutional Neural Network

- 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

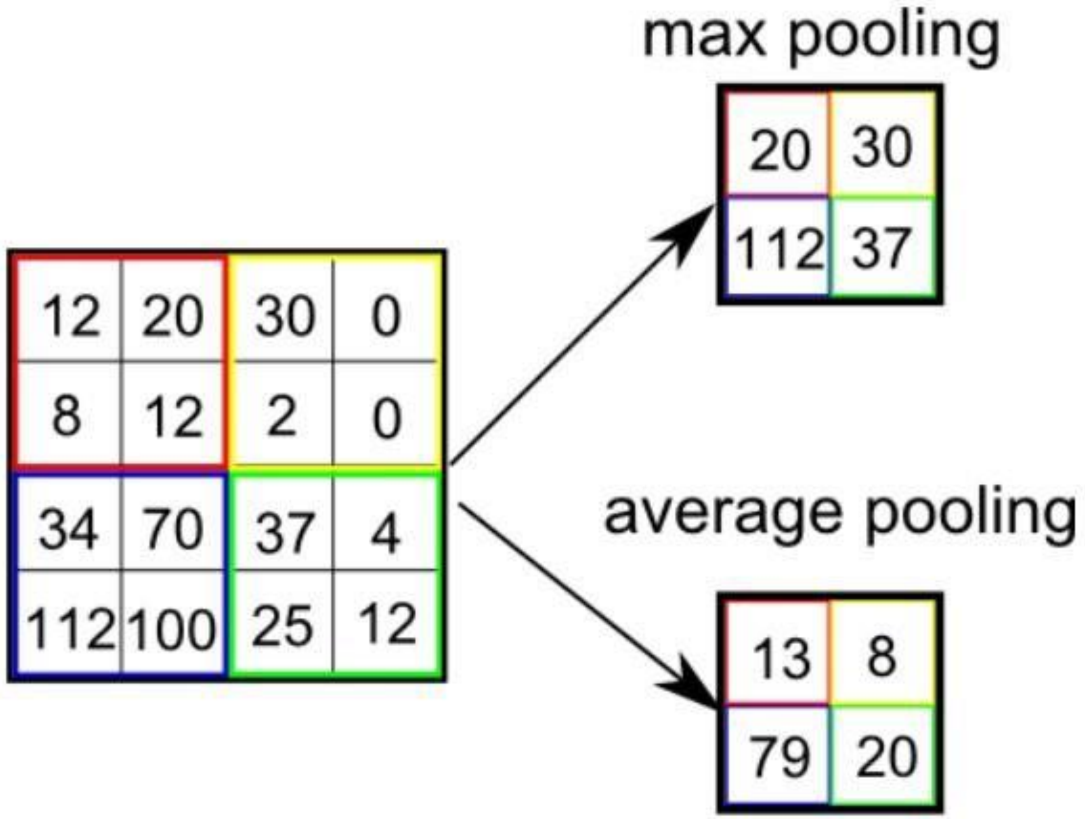
Convolution Layer



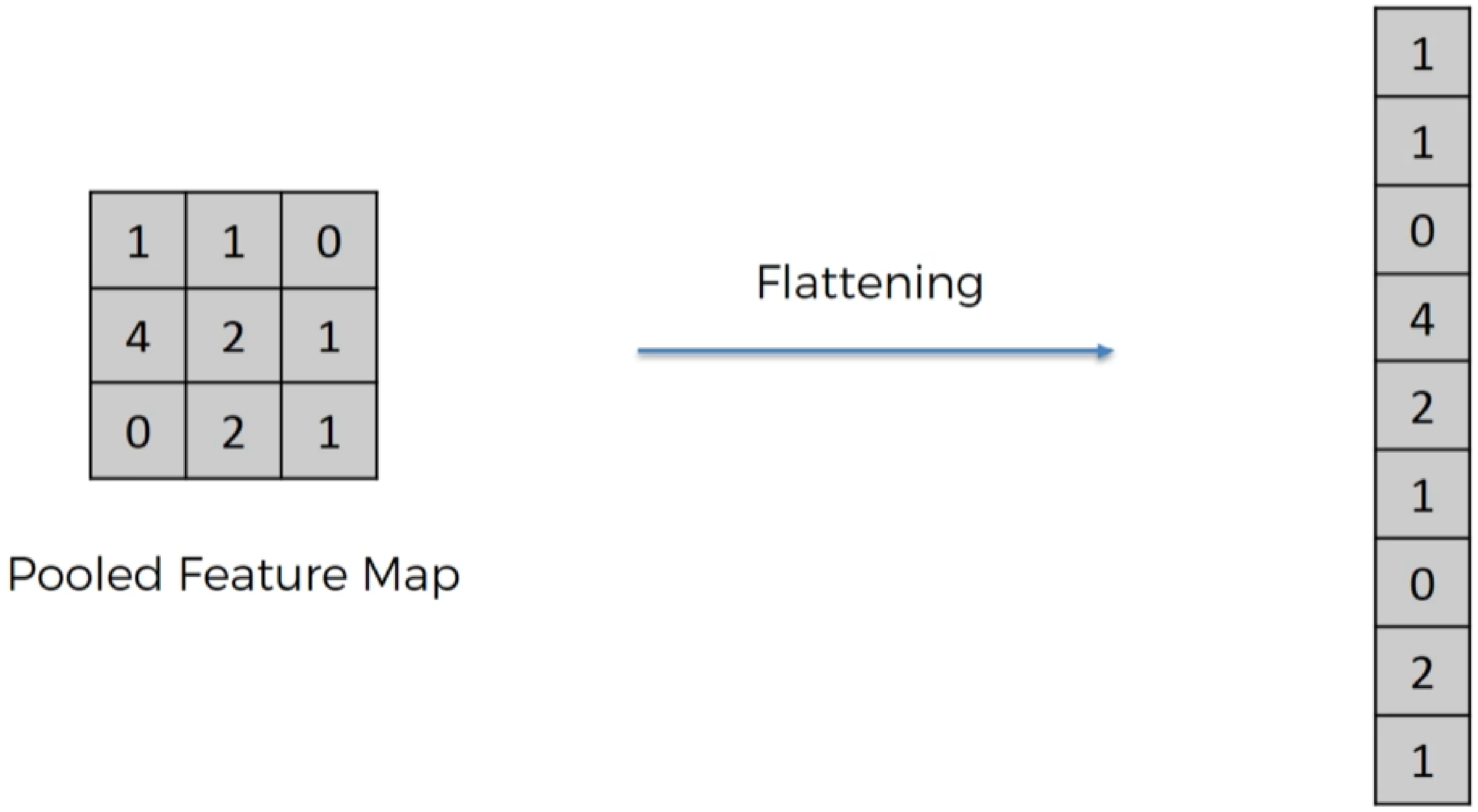
Max Pooling

- 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

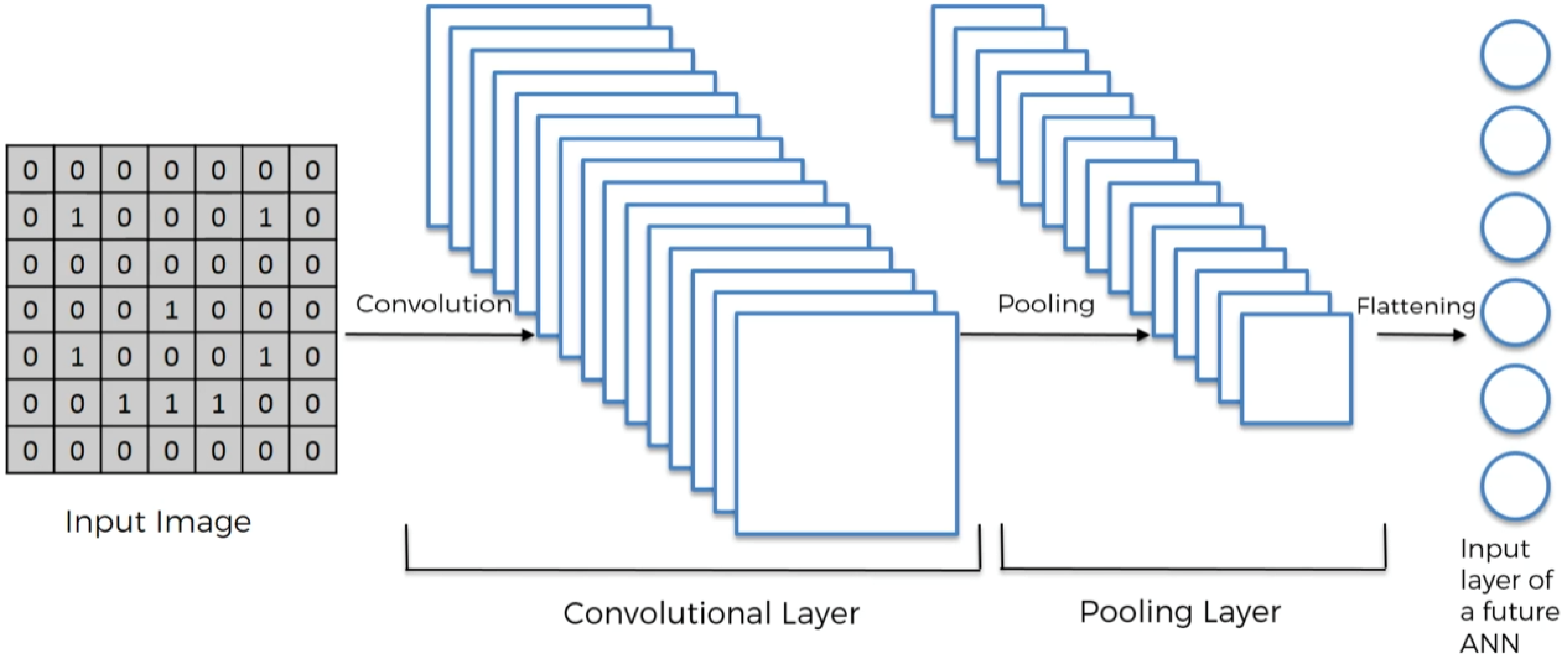
Max Pooling


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

Flattening

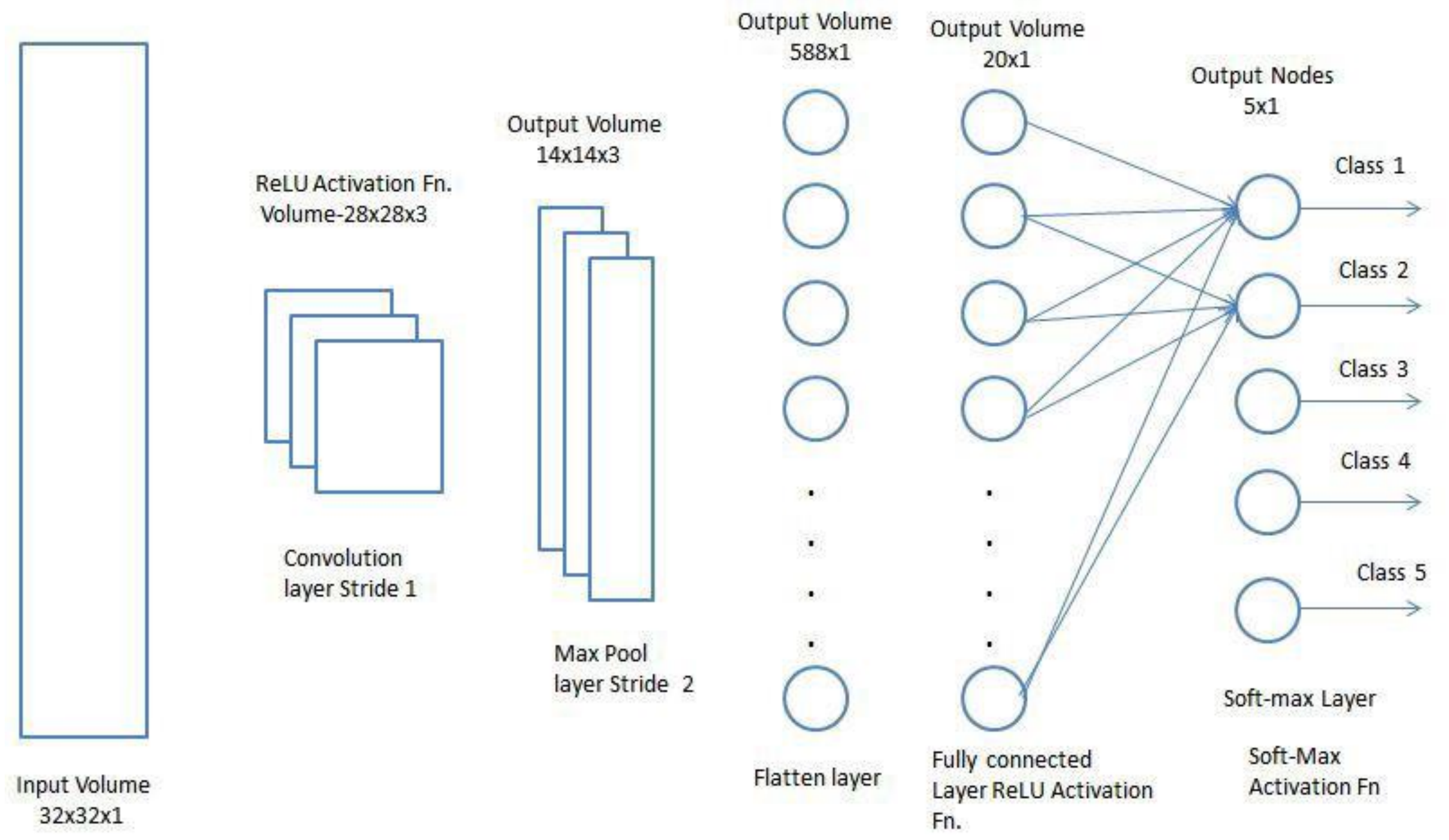


Flattening



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
Convolutional Layer
Pooling
Pooling Layer
Flattening
Input layer of a future ANN

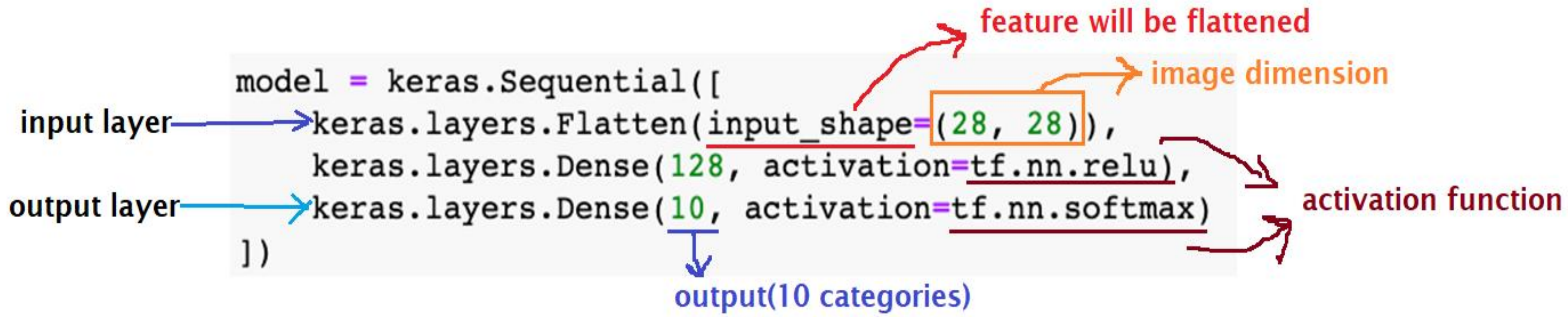
Full Connection



Using TensorFlow

Designing the Neural Network

```
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)
])
```



```
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)
])
```

Activation Function

- The **activation function** of a node defines the output of that node given an input or set of inputs.
- **ReLU (Rectified Linear Unit) :**This is most popular activation function which is used in hidden layer of NN
- **Softmax :** Generally, we use the function at last layer of neural network which calculates the probabilities distribution of the event over ’n’ different events. The main advantage of the function is able to handle multiple classes.
-

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)

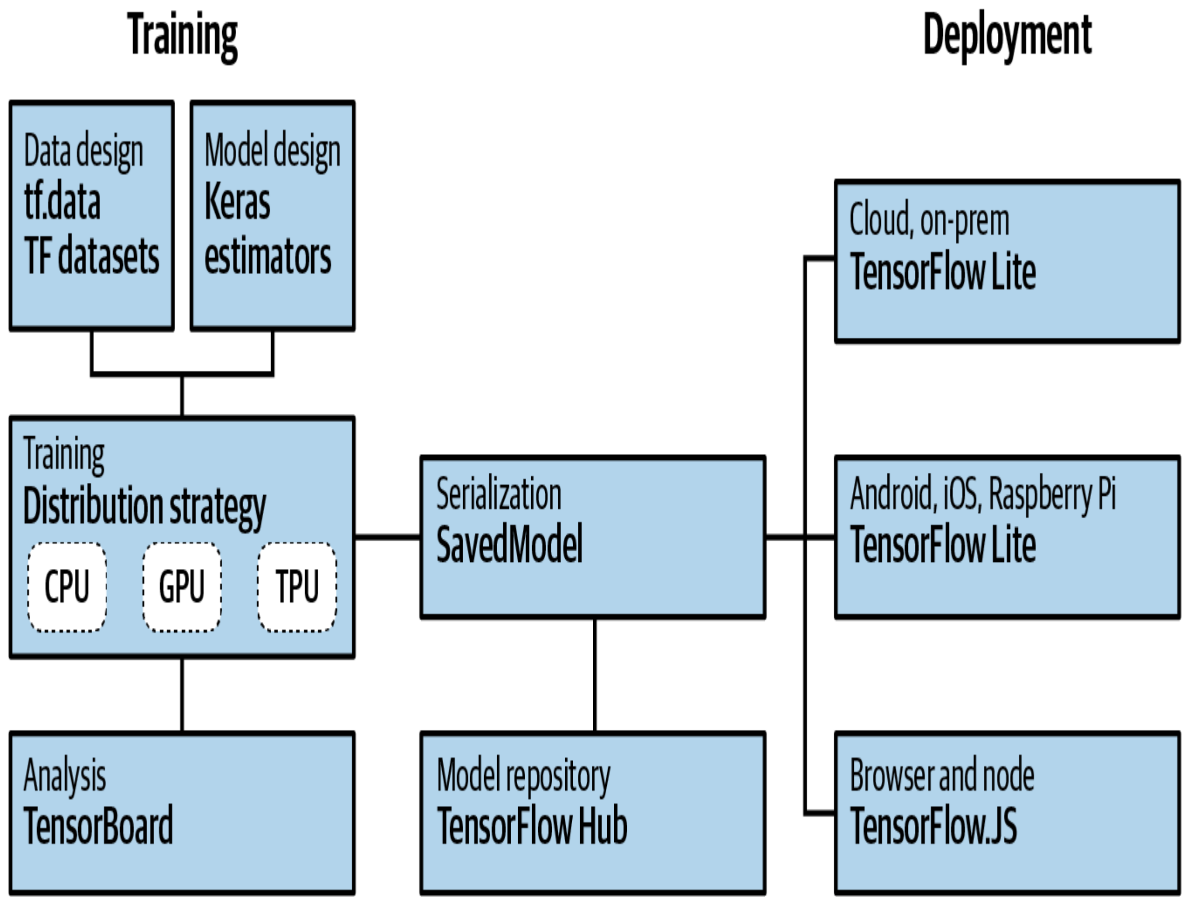
- The next one, Dense (hidden layers), is a layer of neurons, and we're specifying that we want 128 of them
- More neurons means it will run more slowly, as it has to learn more parameters
- It takes some experimentation over time to pick the right values. This process is typically called hyperparameter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

- 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

TensorFlow

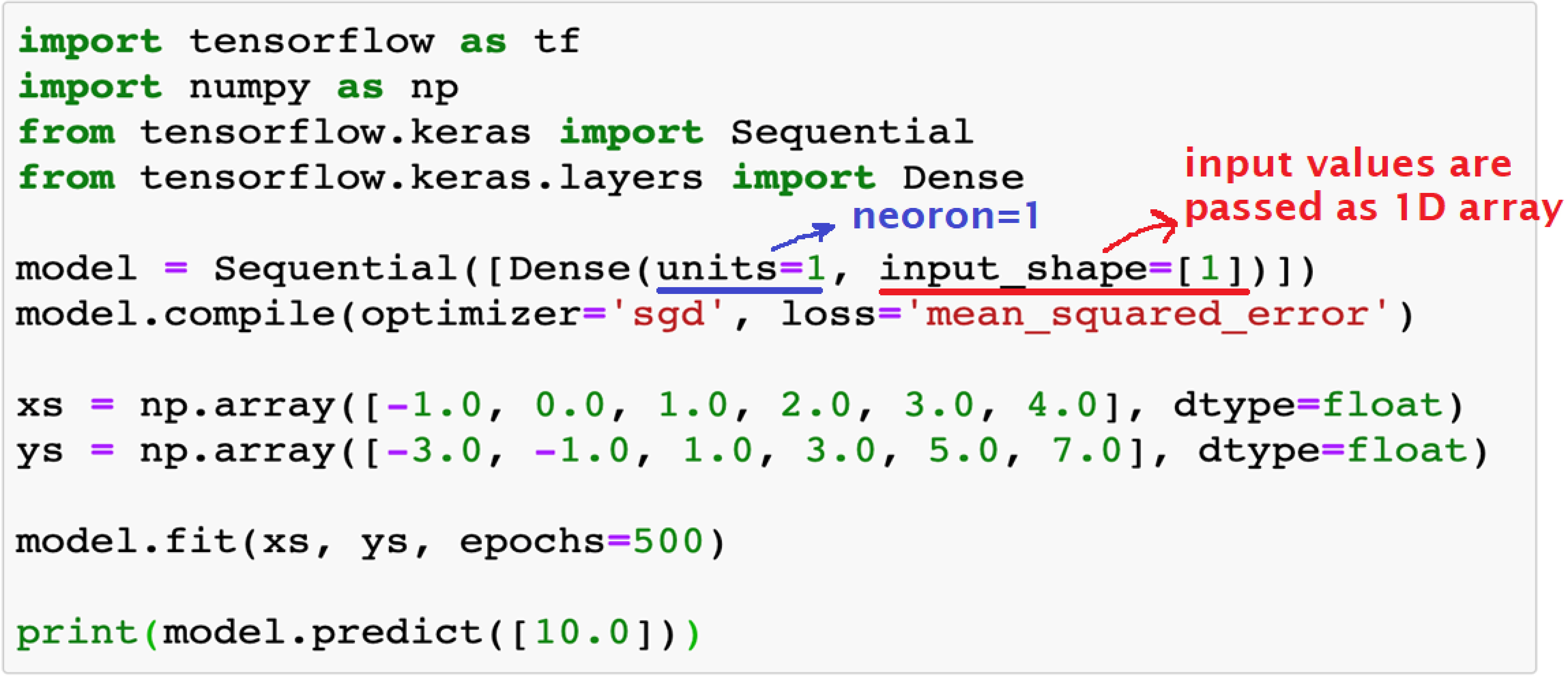
TensorFlow

- 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



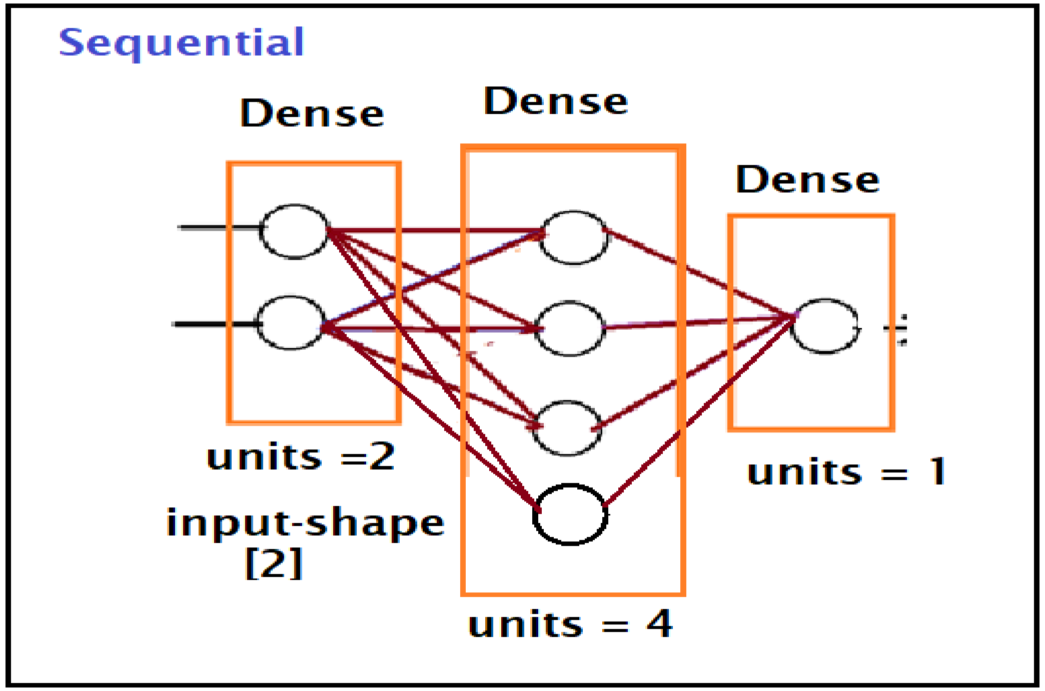
```
import tensorflow as tf
import numpy as np
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

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]))
```



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