VIT BHOPAL
www.vitbhopal.ac.in

Introduction of Neural Network and Back-Propagation

Dr. Virendra Singh Kushwah

Assistant Professor Grade-II

School of Computing Science and Engineering

Virendra.Kushwah@vitbhopal.ac.in

7415869616

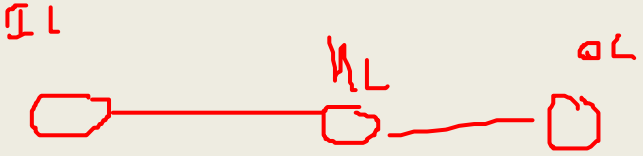
What is a Neural Network?

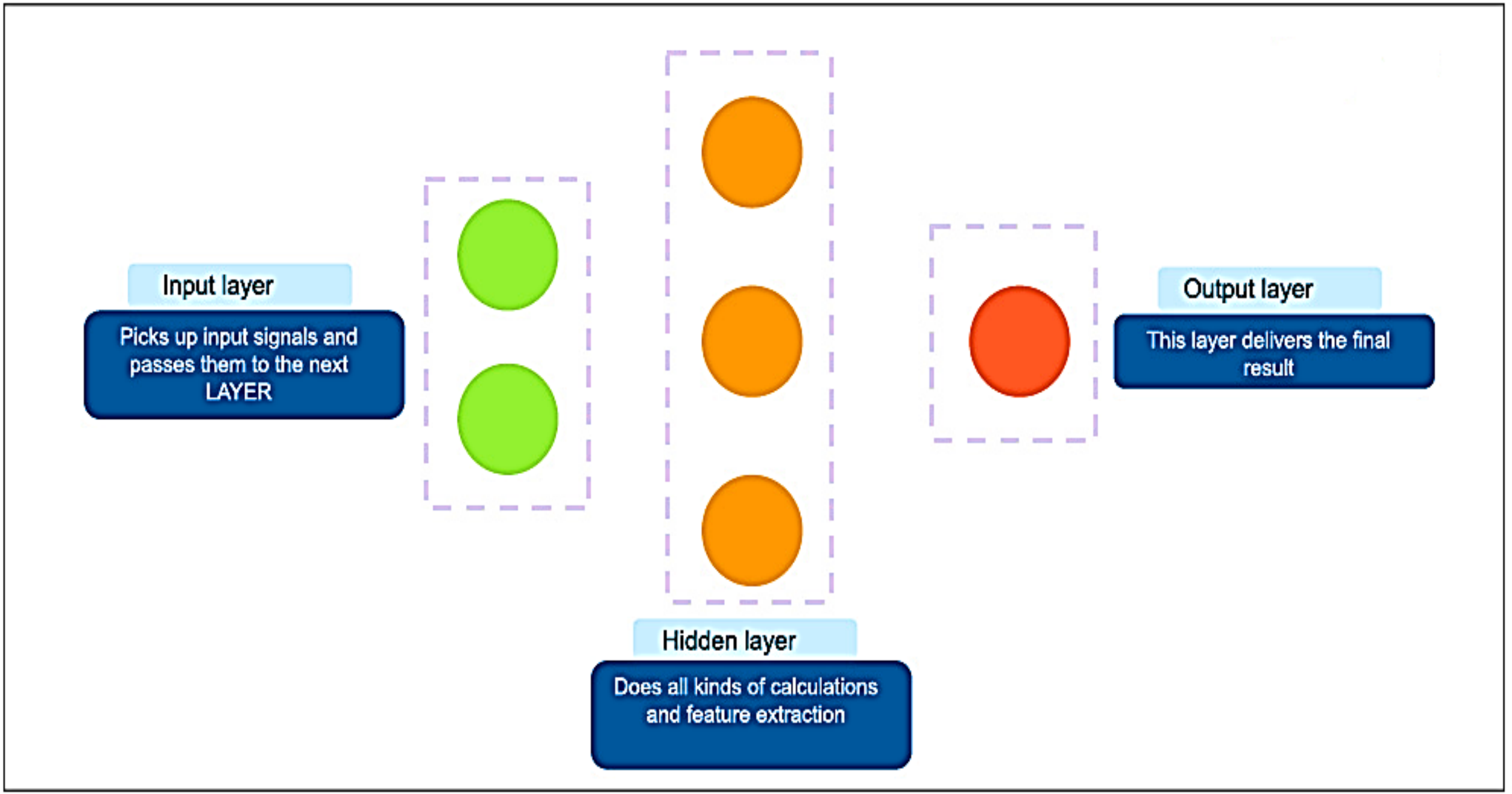
- You've probably already been using neural networks on a daily basis. When you ask your mobile assistant to perform a search for you—say, Google or Siri or Amazon Web—or use a self-driving car, these are all neural network-driven. Computer games also use neural networks on the back end, as part of the game system and how it adjusts to the players, and so do map applications, in processing map images and helping you find the quickest way to get to your destination.

- A neural network is a system or hardware that is designed to operate like a human brain.

Working of Neural Network

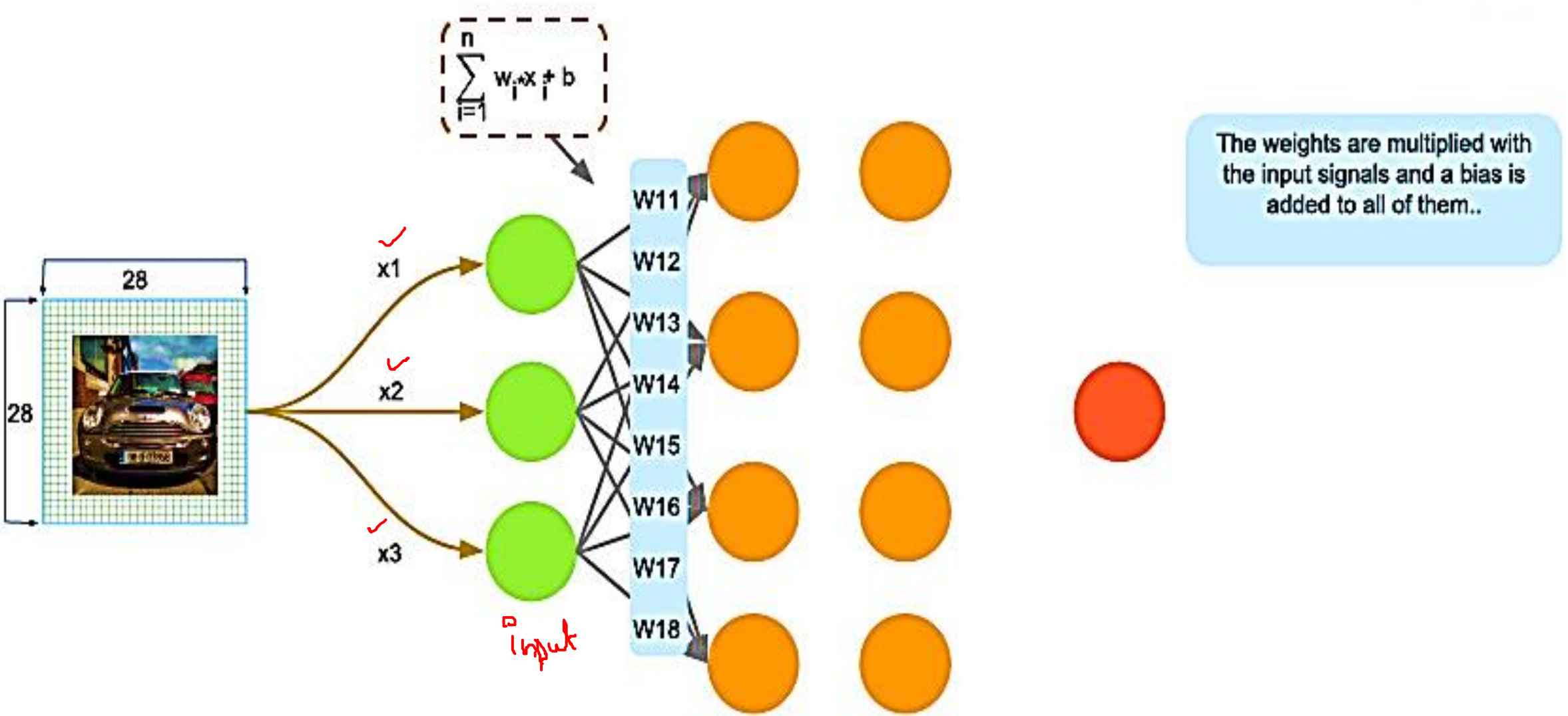
- A neural network is usually described as having different layers. The first layer is the input layer, it picks up the input signals and passes them to the next layer. The next layer does all kinds of calculations and feature extractions—it's called the hidden layer. Often, there will be more than one hidden layer. And finally, there's an output layer, which delivers the final result.

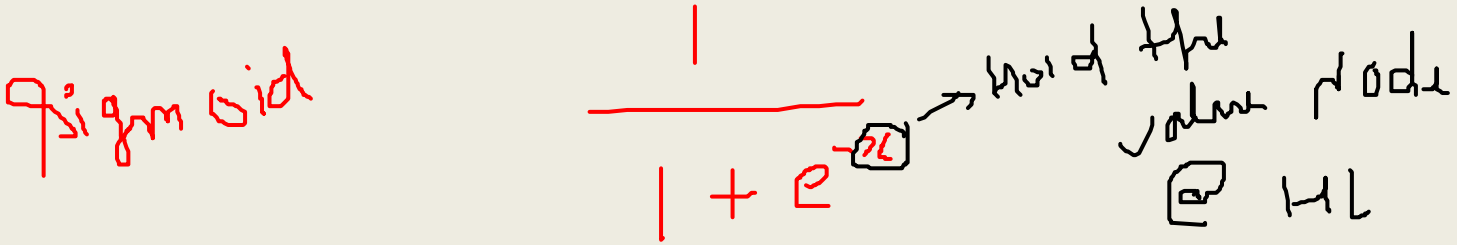



Input layer
Picks up input signals and passes them to the next LAYER
Hidden layer
Does all kinds of calculations and feature extraction
Output layer
This layer delivers the final result

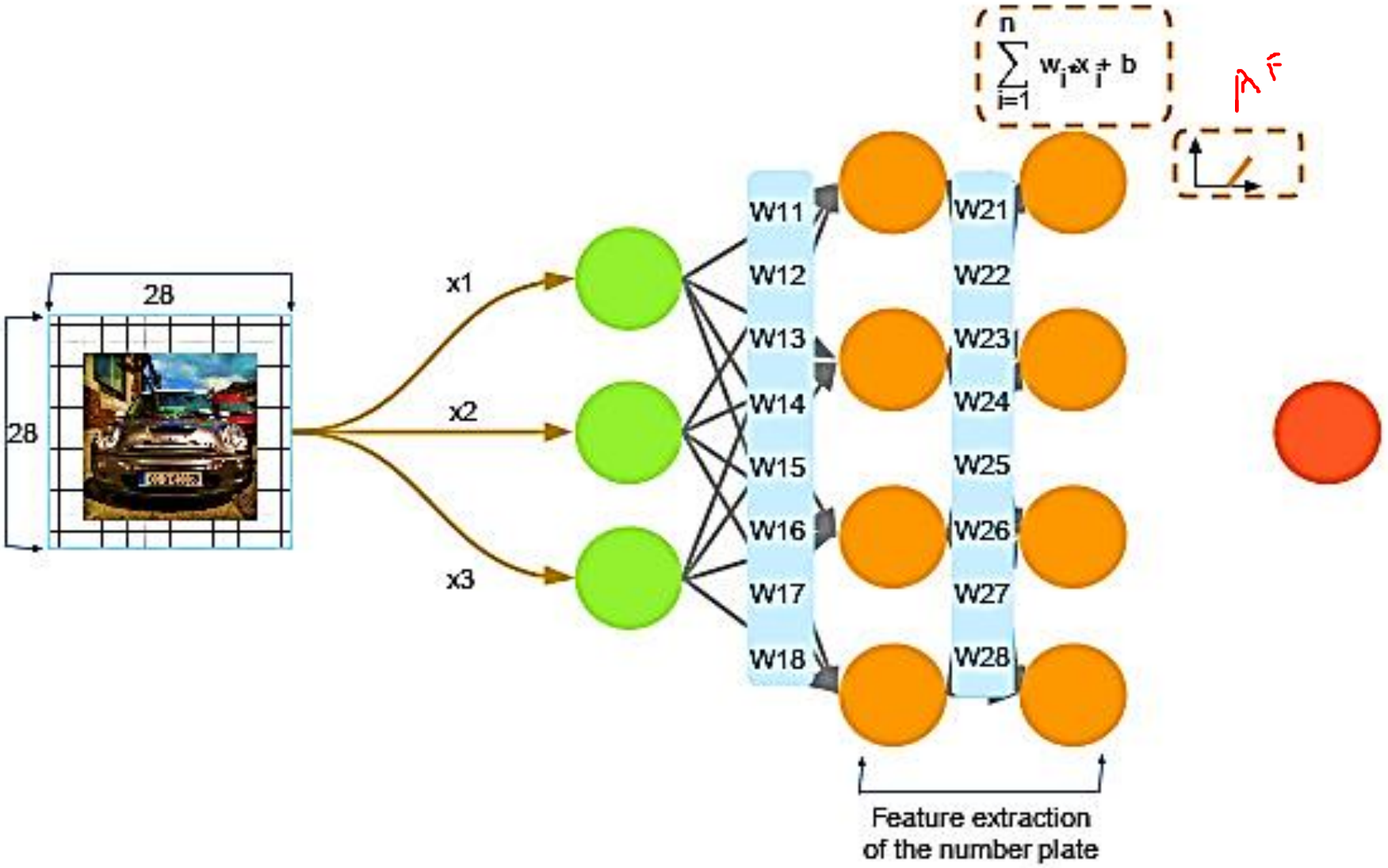
- Let’s take the real-life example of how traffic cameras identify license plates and speeding vehicles on the road. The picture itself is 28 by 28 pixels, and the image is fed as an input to identify the license plate. Each neuron has a number, called activation, which represents the grayscale value of the corresponding pixel, ranging from 0 to 1—it’s 1 for a white pixel and 0 for a black pixel. Each neuron is lit up when its activation is close to 1.

- Pixels in the form of arrays are fed into the input layer. If your image is bigger than 28 by 28 pixels, you must shrink it down, because you can't change the size of the input layer. In our example, we'll name the inputs as X1, X2, and X3. Each of those represents one of the pixels coming in. The input layer then passes the input to the hidden layer. The interconnections are assigned weights at random. The weights are multiplied with the input signal, and a bias is added to all of them.


$\sum_{i=1}^{n} w_i x_i + b$
28
28
x1
x2
x3
W11
W12
W13
W14
W15
W16
W17
W18
Input
The weights are multiplied with the input signals and a bias is added to all of them..



- The weighted sum of the inputs is fed as input to the activation function, to decide which nodes to fire for feature extraction. As the signal flows within the hidden layers, the weighted sum of inputs is calculated and is fed to the activation function in each layer to decide which nodes to fire.

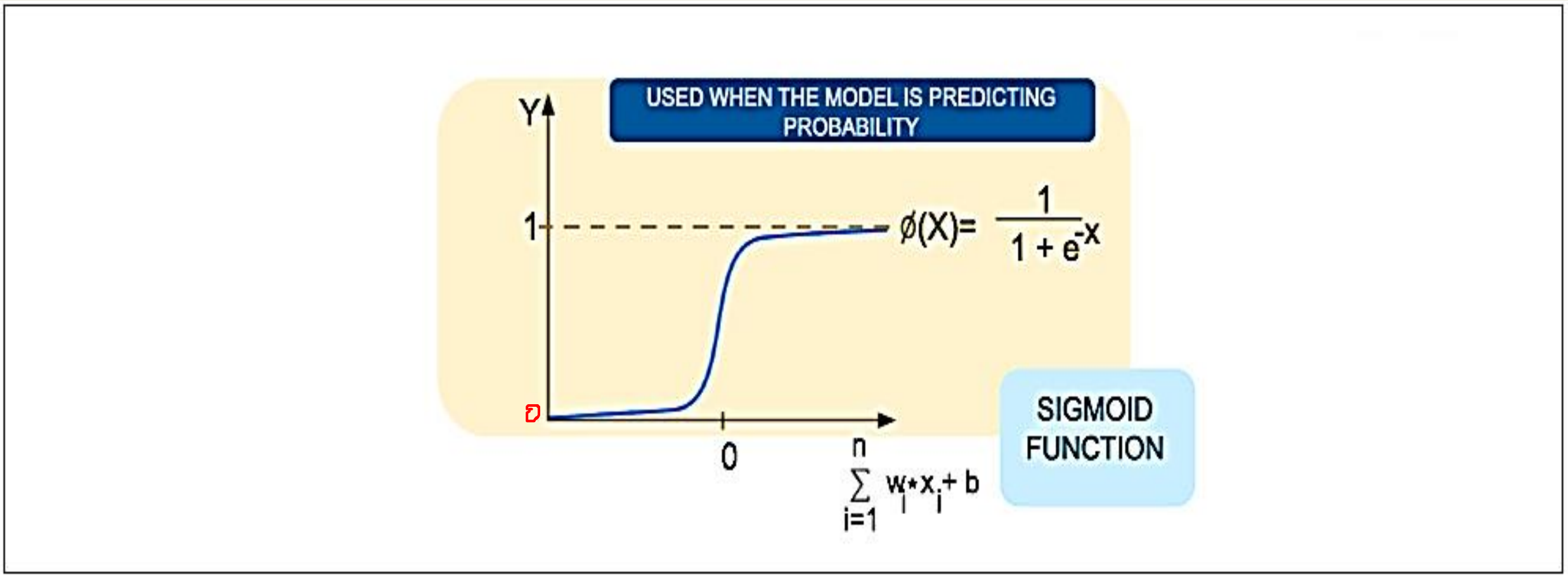


As the signal flows within the hidden layers, the weighted sum of inputs is calculated and is fed to the activation function in each layer to decide which nodes to fire

- Here we'll take a detour to examine the neural network activation function. *There are different types of activation functions.*

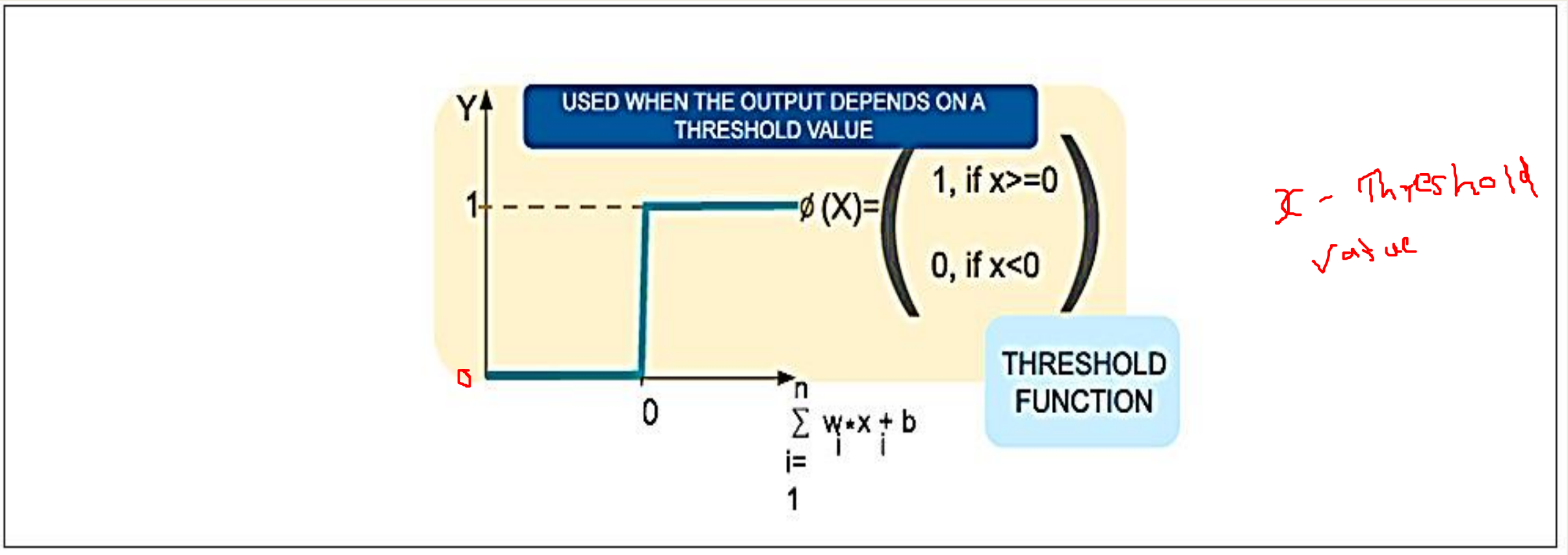
Sigmoid Function

- The sigmoid function is used when the model is predicting probability.



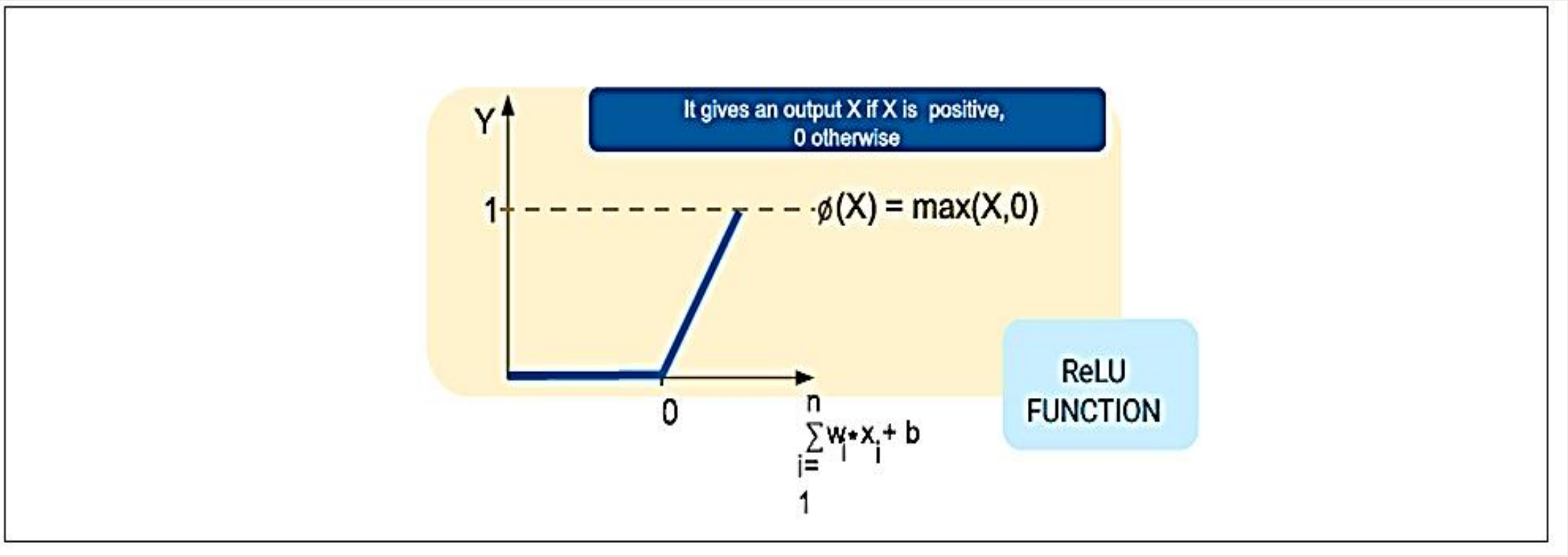
Threshold Function

- The threshold function is used when you don't want to worry about the uncertainty in the middle.



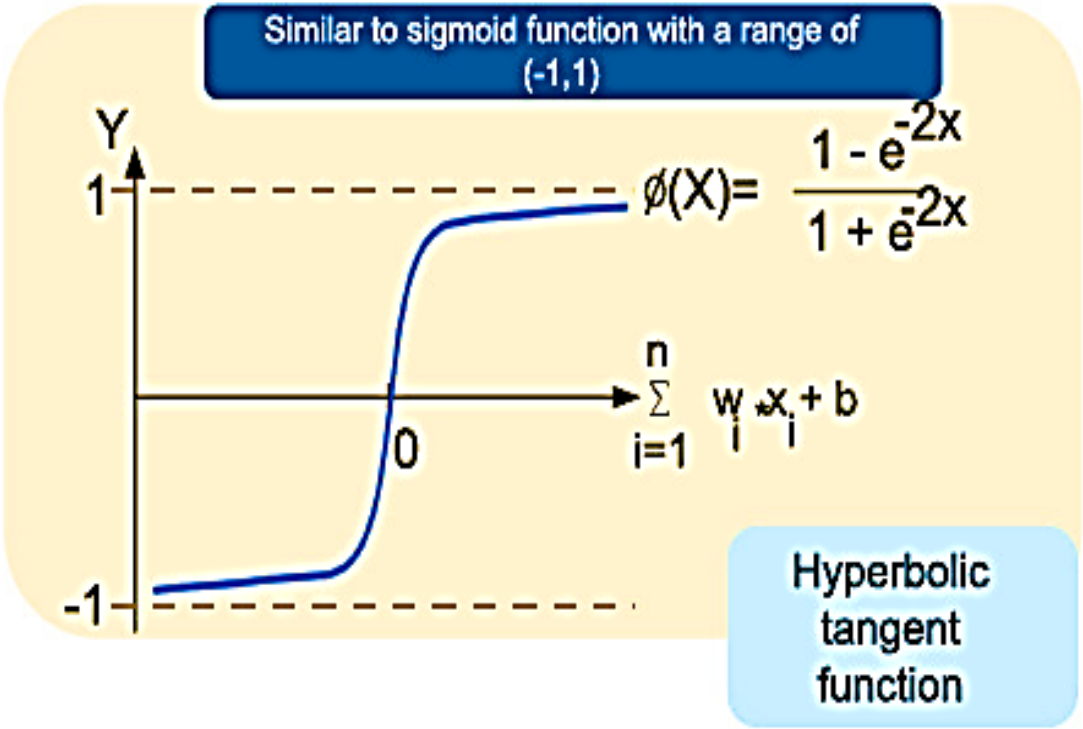
ReLU (rectified linear unit) Function

- The ReLU (rectified linear unit) function gives the value but says if it's over 1, then it will just be 1, and if it's less than 0, it will just be 0. The ReLU function is most commonly used these days.



Hyperbolic Tangent Function

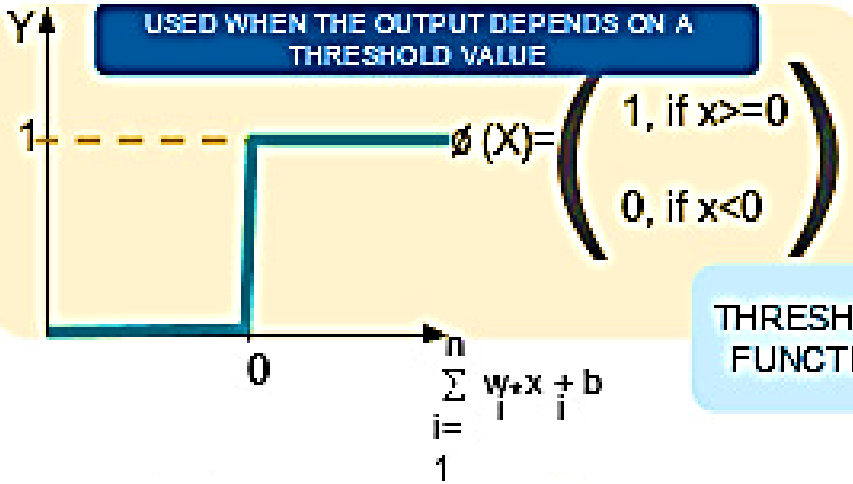
- The hyperbolic tangent function is similar to the sigmoid function but has a range of -1 to 1.



- Now that you know what an activation function is, let's get back to the neural network. Finally, the model will predict the outcome, applying a suitable application function to the output layer. In our example with the car image, optical character recognition (OCR) is used to convert it into the text to identify what's written on the license plate. In our neural network example, we show only three dots coming in, eight hidden layer nodes and one output, but there's really a huge amount of input and output.

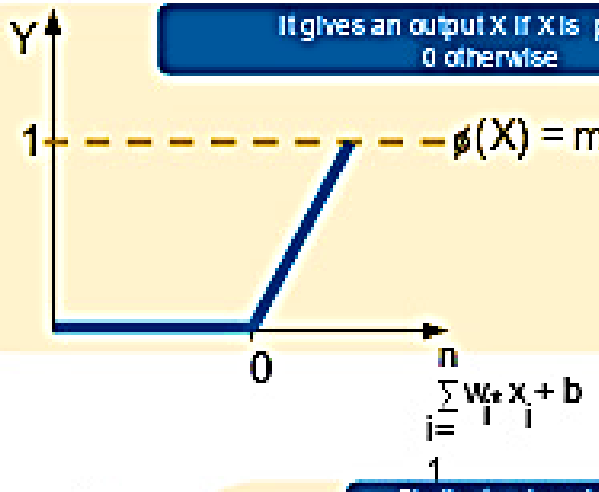
DL → CNN →

USED WHEN THE OUTPUT DEPENDS ON A THRESHOLD VALUE



THRESHOLD FUNCTION

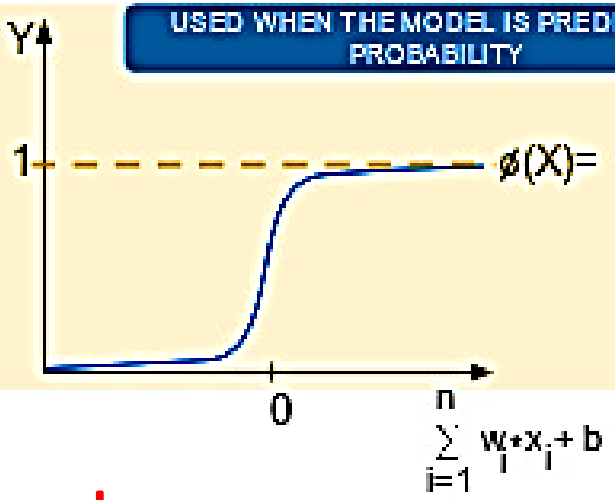
It gives an output X if X is positive, 0 otherwise



relu FUNCTION

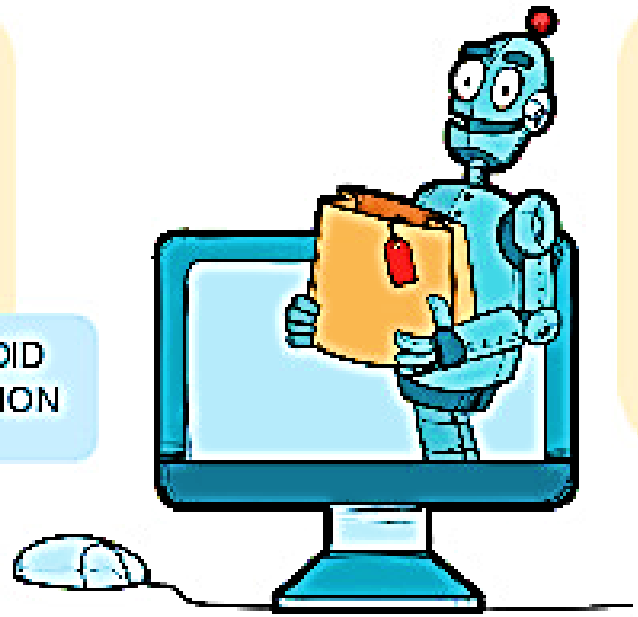
0 to ∞

USED WHEN THE MODEL IS PREDICTING PROBABILITY

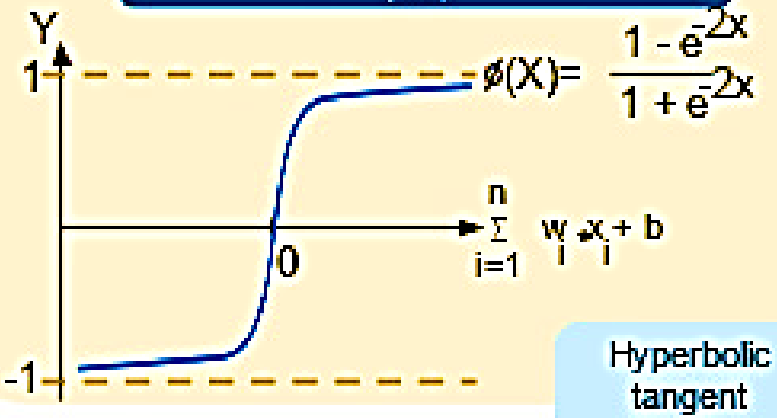


SIGMOID FUNCTION

Range 0 to 1

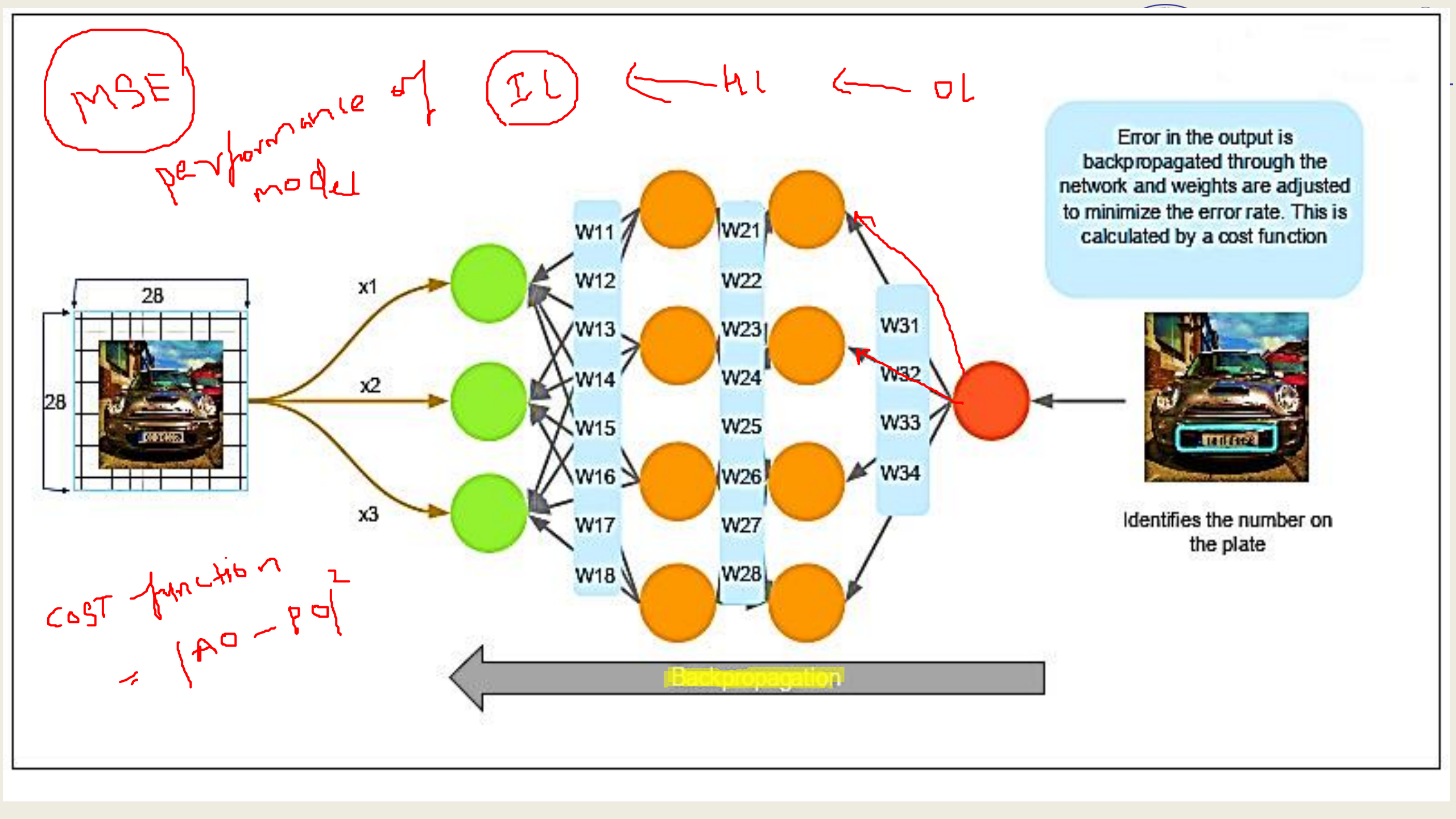


Similar to sigmoid function with a range of (-1,1)



Hyperbolic tangent function

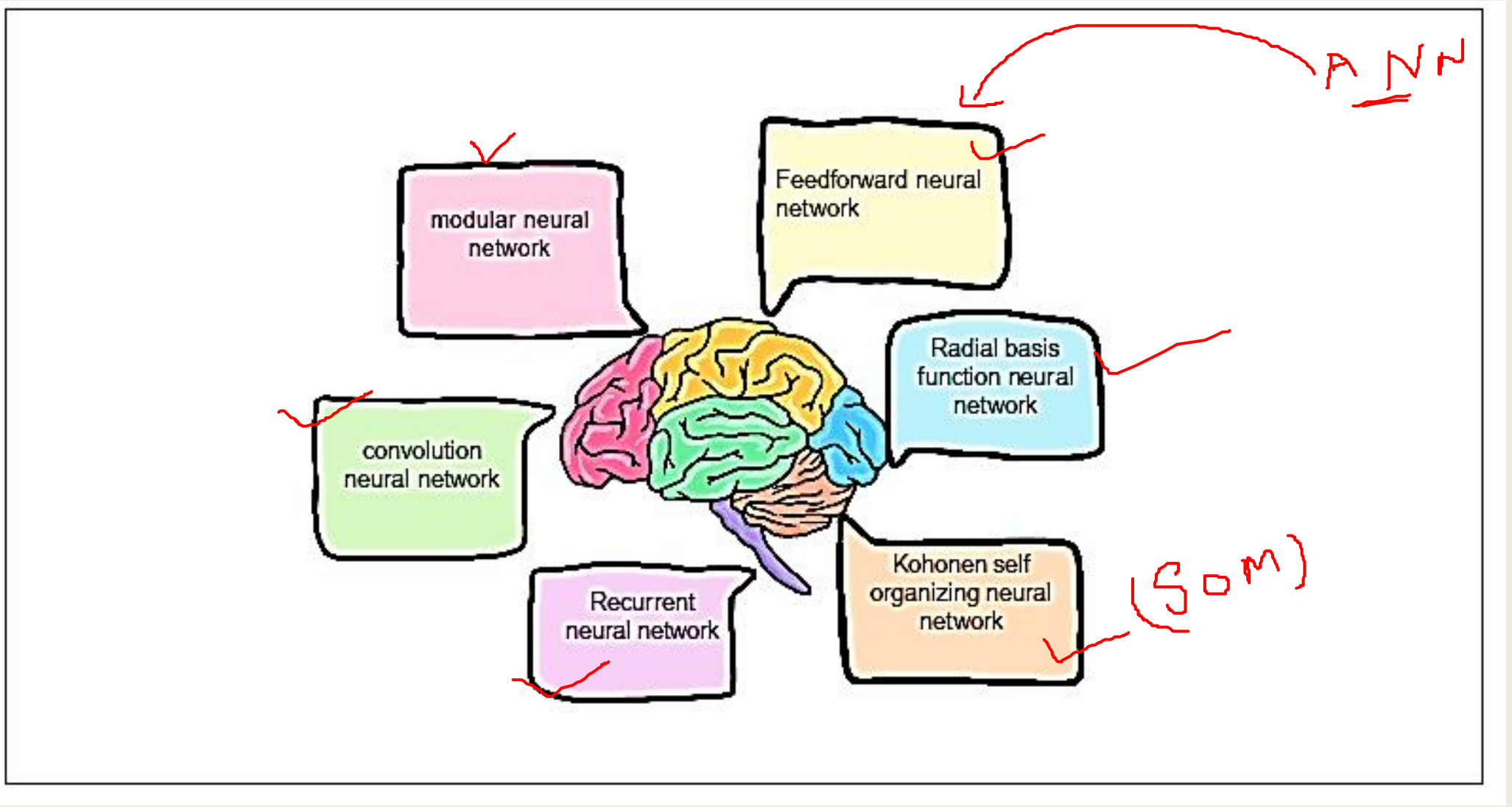
- Error in the output is back-propagated through the network and weights are adjusted to minimize the error rate.
- This is calculated by a cost function.
- You keep adjusting the weights until they fit all the different training models you put in.


MSE
performance of model
IL
HL
OL
Error in the output is backpropagated through the network and weights are adjusted to minimize the error rate. This is calculated by a cost function
28
28
x1
x2
x3
W11
W12
W13
W14
W15
W16
W17
W18
W21
W22
W23
W24
W25
W26
W27
W28
W31
W32
W33
W34
Identifies the number on the plate
Backpropagation
cost function
= (A0 - p0)^2

- The output is then compared with the original result, and multiple iterations are done for maximum accuracy.

- With every iteration, the weight at every interconnection is adjusted based on the error.

Types of Neural Networks



- **Feed-forward Neural Network**
- This is the simplest form of ANN (artificial neural network); data travels only in one direction (input to output). This is the example we just looked at. When you actually use it, it's fast; when you're training it, it takes a while. Almost all vision and speech recognition applications use some form of this type of neural network.

- **Radial Basis Functions Neural Network**
- This model classifies the data point based on its distance from a center point. If you don't have training data, for example, you'll want to group things and create a center point. The network looks for data points that are similar to each other and groups them. One of the applications for this is power restoration systems.

- **Kohonen Self-organizing Neural Network or Self-Organizing Map (SOM)**
- Vectors of random input are input to a discrete map comprised of neurons. Vectors are also called dimensions or planes. Applications include using it to recognize patterns in data like a medical analysis.

- **Recurrent Neural Network**

(LSTM)

- In this type, the hidden layer saves its output to be used for future prediction. The output becomes part of its new input. Applications include text-to-speech conversion.

- **Convolution Neural Network**

(Deep learning)

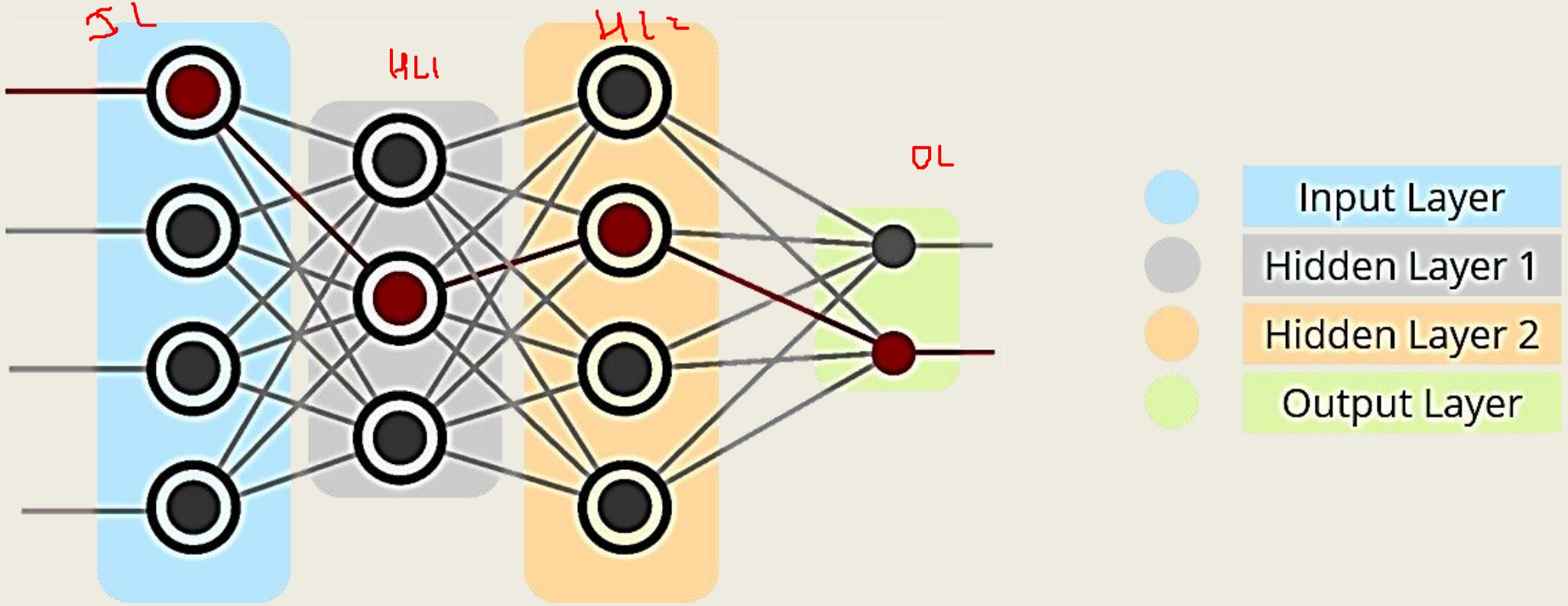
- In this type, the input features are taken in batches—as if they pass through a filter. This allows the network to remember an image in parts. Applications include signal and image processing, such as facial recognition.

- **Modular Neural Network**
- This is composed of a collection of different neural networks working together to get the output. This is cutting-edge and is still in the research phase.

Backpropagation

multiple HLs

- Backpropagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks).



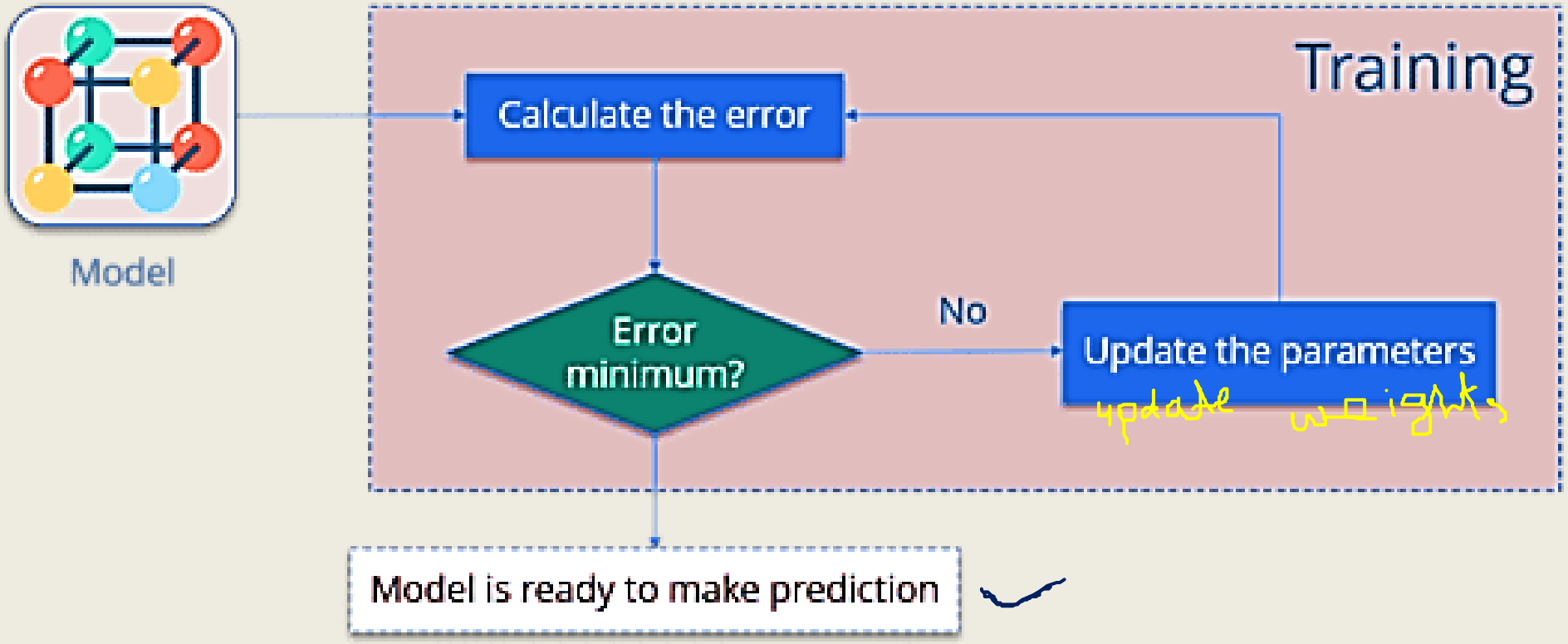
Why We Need Backpropagation?

- While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact.

- Now obviously, we are not superhuman. So, it's not necessary that whatever weight values we have selected will be correct, or it fits our model the best.

- Okay, fine, we have selected some weight values in the beginning, but our model output is way different than our actual output i.e. the error value is huge.

- **Now, how will you reduce the error?**
- Basically, what we need to do, we need to somehow explain the model to change the parameters (weights), such that error becomes minimum.
- Let's put it in an another way, we need to train our model.
- One way to train our model is called as Backpropagation. Consider the diagram below:



- **Let me summarize the steps for you:**

1. Calculate the error – How far is your model output from the actual output.
2. Minimum Error – Check whether the error is minimized or not.
3. Update the parameters – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
4. Model is ready to make a prediction – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

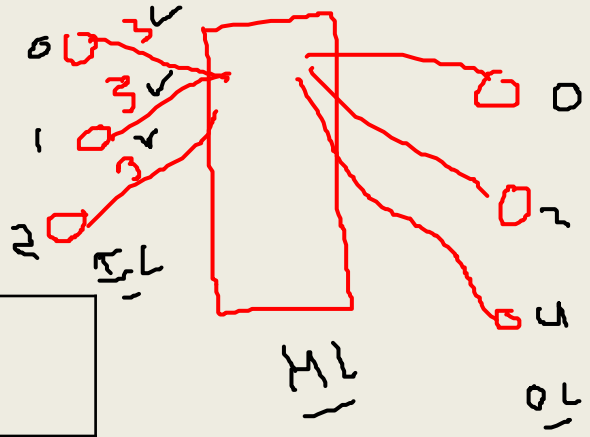
What is Gradient Descent

- The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the ***delta rule or gradient descent***.

- The weights that minimize the error function is then considered to be a solution to the learning problem.

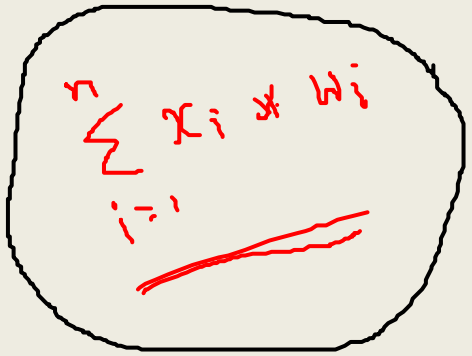
Let's understand how it works with an example:

- Consider the below table:



Input	Desired Output
0	0
1	2
2	4

- Now the output of your model when ‘W” value is 3:



used to get Calculation

Input	Desired Output	Model output (W=3)
0	0	0 (0×3)
1	2	3 (1×3)
2	4	6 (2×3)

- Notice the difference between the actual output and the desired output:

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error
0	0	0	0	0
1	2	3	1	1
2	4	6	2	4

- Let's change the value of 'W'. Notice the error when 'W' = '4'

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error	Model output (W=4)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	4	4
2	4	6	2	4	8	16

- Now if you notice, when we increase the value of 'W' the error has increased. So, obviously there is no point in increasing the value of 'W' further. But, what happens if I decrease the value of 'W'?
- Consider the table below:

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error	Model output (W=2)	Square Error
0	0	0	0	0	0	0
1	2	3	2	4	3	0
2	4	6	2	4	4	0

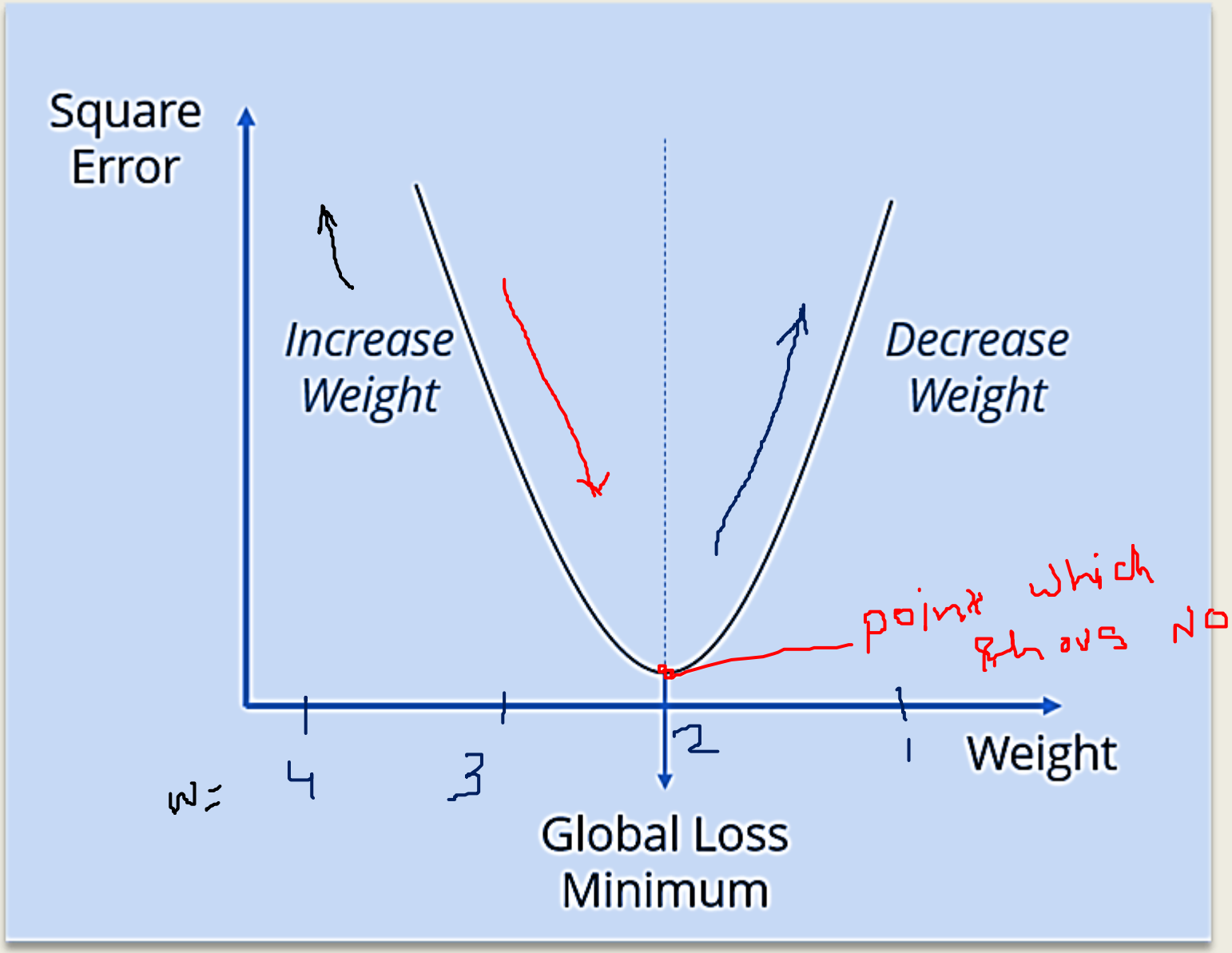
- Now, what we did here:

- We first initialized some random value to ‘W’ and propagated forward.
- Then, we noticed that there is some error. To reduce that error, we propagated backwards and increased the value of ‘W’.
- After that, also we noticed that the error has increased. We came to know that, we can’t increase the ‘W’ value.
- So, we again propagated backwards and we decreased ‘W’ value.
- Now, we noticed that the error has reduced.

- So, we are trying to get the value of weight such that the error becomes minimum. Basically, we need to figure out whether we need to increase or decrease the weight value.

- Once we know that, we keep on updating the weight value in that direction until error becomes minimum. You might reach a point, where if you further update the weight, the error will increase. At that time you need to stop, and that is your final weight value.

Consider the graph below:



We need to reach the 'Global Loss Minimum'.

This is nothing but Backpropagation.

VIT®

BHOPAL

www.vitbhopal.ac.in