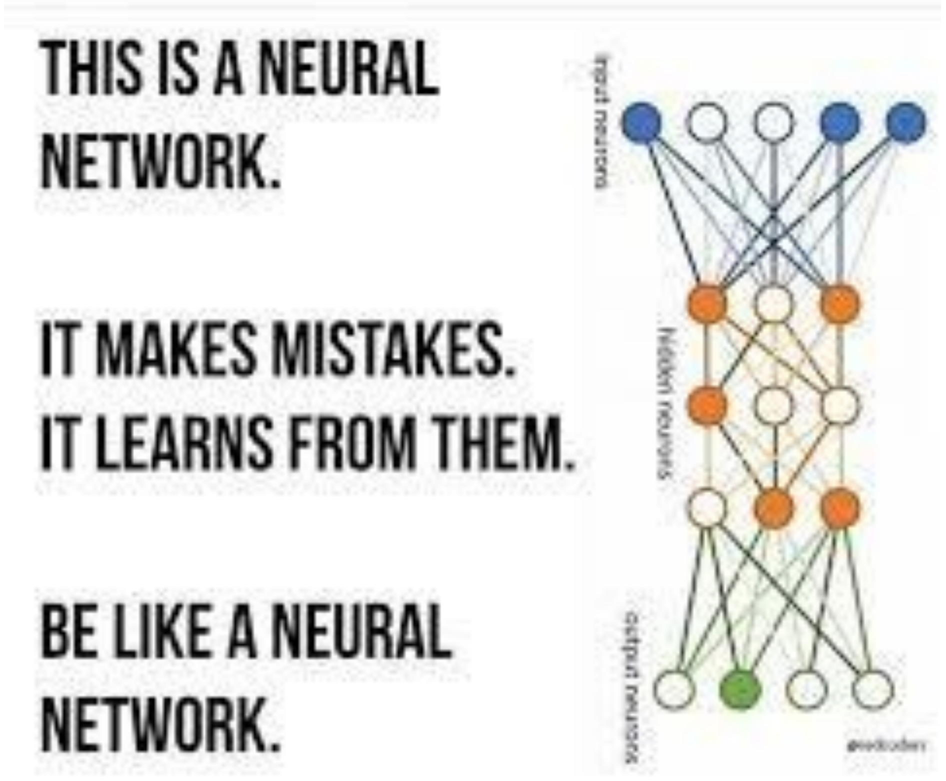
# Introduction to Neural Networks



# Whether you should go surfing (Yes: 1, No: 0)?

- Marine Life. Sharks just have to come top of the list. ...
- Drowning. There is a very real risk of drowning while surfing. ...
- Waves. Waves may look nice from the beach but can be incredibly powerful. ...
- Locals. ...
- Riptides. ...
- Surfboards. ...
- Leash Tangles. ...
- The Sea Bed.

## Example 1: whether you should go surfing (Yes: 1, No: 0).

- The decision to go or not to go is our predicted outcome, or y-hat.
- Let’s assume that there are three factors influencing your decision-making:
- Are the waves good? (Yes: 1, No: 0)
- Is the line-up empty? (Yes: 1, No: 0)
- Has there been a recent shark attack? (Yes: 0, No: 1)
- X1 = 1, since the waves are pumping
- X2 = 0, since the crowds are out
- X3 = 1, since there hasn’t been a recent shark attack

Assign some weights to determine importance. Larger weights signify that particular variables are of greater importance to the decision or outcome.

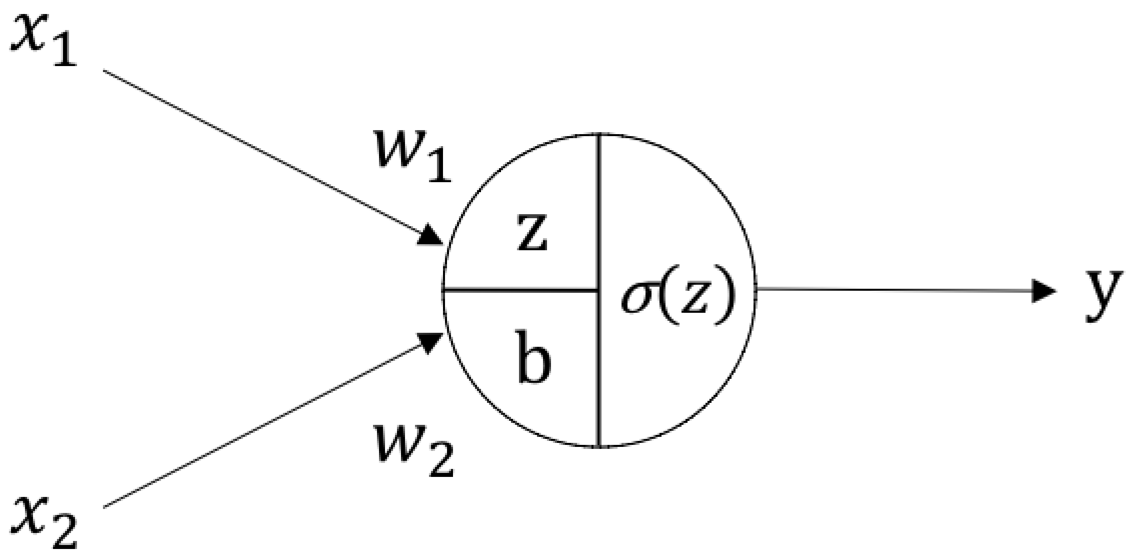
- W1 = 5, since large swells don’t come around often
- W2 = 2, since you’re used to the crowds
- W3 = 4, since you have a fear of sharks
- Finally, we’ll also assume a threshold value of 3, which would translate to a bias value of 3.
- With all the various inputs, we can start to plug in values into the formula to get the desired output.
- Y-hat = (1*5) + (0*2) + (1*4) + (-3) = 6   6 > 1

$$\text{output} = f(x) = \begin{cases} 1 \text{ if } \sum w_1 x_1 + b \geq 0 \\ 0 \text{ if } \sum w_1 x_1 + b < 0 \end{cases}$$

## Summary

- our brain is fully aware of the context of the data we use in making predictions
- In  the Neural Network nothing other than apply the formula to the bunch of data to produce a probability/guess that we term as prediction.
- While our brain reasons with the data it comes in contact with,
- the Neural Network makes no sense out of the data, it rather takes in input, applies a formula and produces an output –

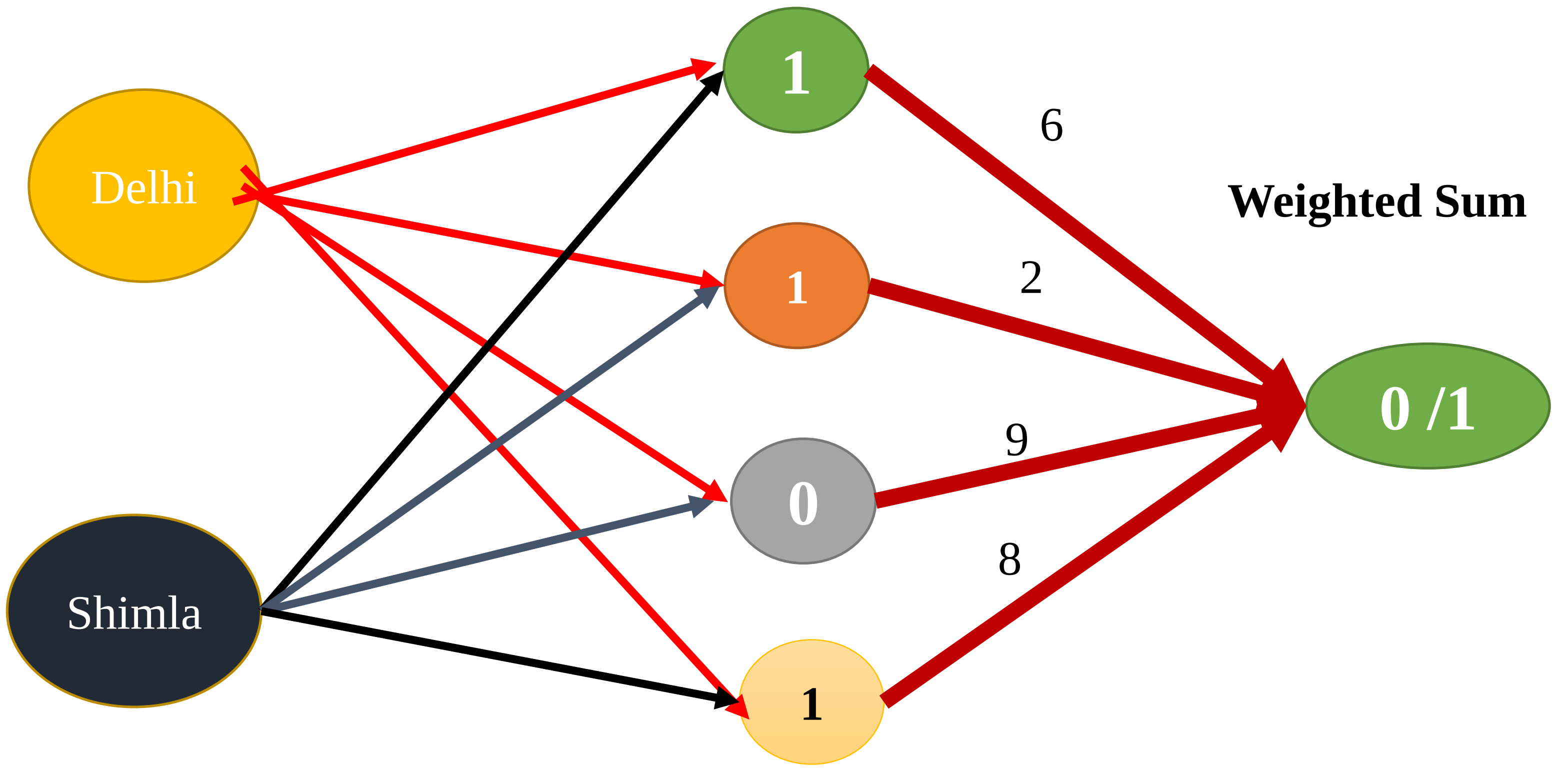
# Example 2:



$$z = x_1 w_1 + x_2 w_2 + b$$
$$y = \sigma(z) = \frac{1}{1+\exp(-z)}$$

| x1 | x2 | w1 | w2 | weighted input z | sigmoid | y |
|---|---|---|---|---|---|---|
| 0.1 | 0.3 | 0.6 | 0.2 | $z = 0.1 * 0.6 + 0.3 * 0.2 + 0$ | $\sigma(z) = \frac{1}{1+\exp(-0.12)}$ =0.530 | 0.530 |
| 0.7 | 0.2 | 0.8 | 0.1 | $z = 0.7 * 0.8 + 0.2 * 0.1 + 0$ | $\sigma(z) = \frac{1}{1+\exp(-0.58)}$ =0.641 | 0.641 |
| 0.2 | 0.4 | 0.9 | 0.5 | $z = 0.2 * 0.9 + 0.4 * 0.5 + 0$ | $\sigma(z) = \frac{1}{1+\exp(-0.38)}$ =0.594 | 0.594 |

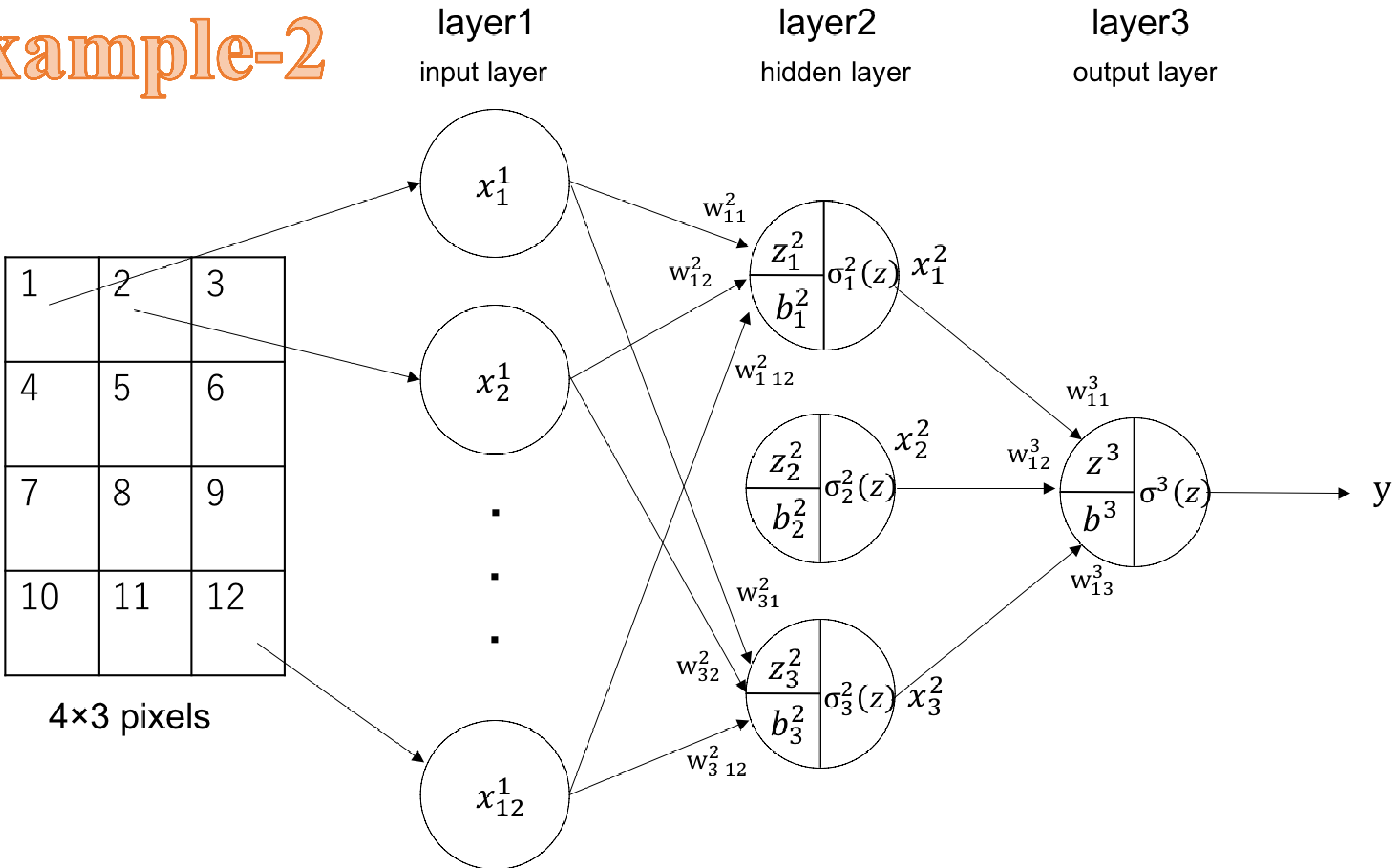
# Forward Propagation

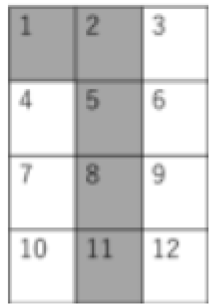


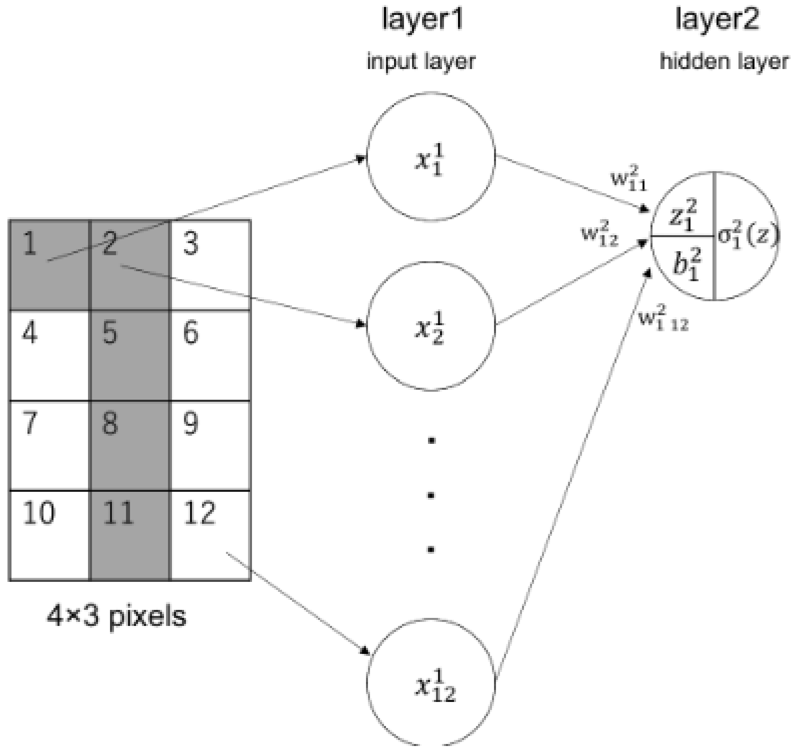
## A neural network is composed of 3 types of layers

- **Input layer** — It is used to pass in our input(an image, text or any suitable type of data for NN).
- **Hidden Layer** — These are the layers in between the input and output layers. These layers are responsible for learning the mapping between input and output.
- **Output Layer** — This layer is responsible for giving us the output of the NN given our inputs.

# Example-2


layer1
input layer
layer2
hidden layer
layer3
output layer
1
2
3
4
5
6
7
8
9
10
11
12
4×3 pixels
$x_1^1$
$x_2^1$
$x_{12}^1$
$w_{11}^2$
$w_{12}^2$
$w_{1\ 12}^2$
$w_{31}^2$
$w_{32}^2$
$w_{3\ 12}^2$
$z_1^2$
$b_1^2$
$\sigma_1^2(z)$
$x_1^2$
$z_2^2$
$b_2^2$
$\sigma_2^2(z)$
$x_2^2$
$z_3^2$
$b_3^2$
$\sigma_3^2(z)$
$x_3^2$
$w_{11}^3$
$w_{12}^3$
$w_{13}^3$
$z^3$
$b^3$
$\sigma^3(z)$
y





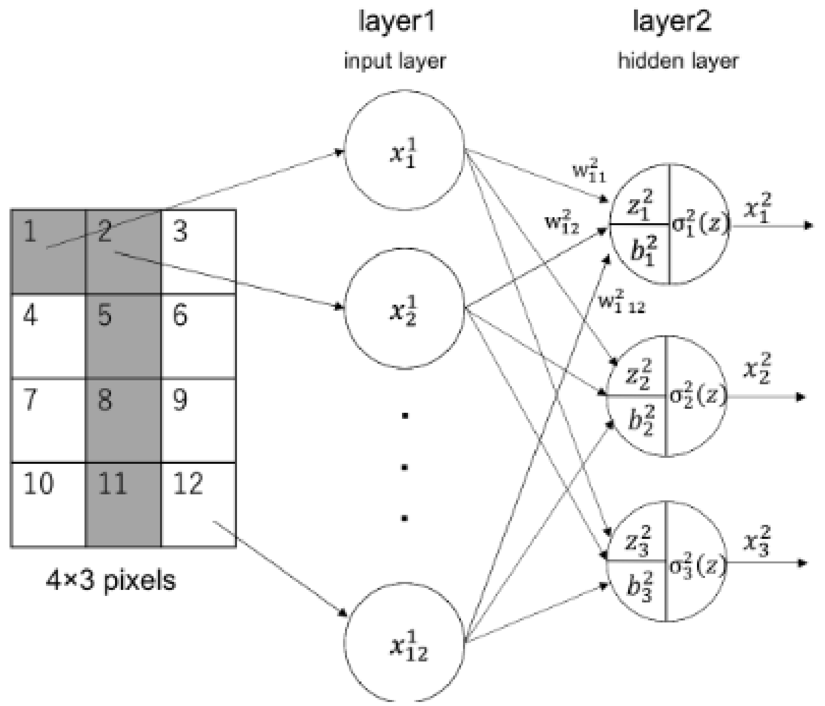
$$x_1^1 = 1,\quad x_2^1 = 1,\quad x_3^1 = 0$$
$$x_4^1 = 0,\quad x_5^1 = 1,\quad x_6^1 = 0$$
$$x_7^1 = 0,\quad x_8^1 = 1,\quad x_9^1 = 0$$
$$x_{10}^1 = 0,\quad x_{11}^1 = 1,\quad x_{12}^1 = 0$$

$$w_{11}^2 = 0.48,\ w_{12}^2 = 0.29,\ w_{13}^2 = 0.58,\ w_{14}^2 = 0.61,\ w_{15}^2 = 0.47,\ w_{16}^2 = 0.18,$$
$$w_{17}^2 = 0.02,\ w_{18}^2 = 0.49,\ w_{19}^2 = 0.67,\ w_{1\,10}^2 = 0.62,\ w_{1\,11}^2 = 0.017,\ w_{1\,12}^2 = 0.28$$
$$b_1^2 = 0$$

be as follows:

$$\begin{aligned}
z_1^2 &= x_1^1 w_{11}^2 + x_2^1 w_{12}^2 + x_3^1 w_{13}^2 + x_4^1 w_{14}^2 + x_5^1 w_{15}^2 + x_6^1 w_{16}^2 + x_7^1 w_{17}^2 \\
&\quad + x_8^1 w_{18}^2 + x_9^1 w_{19}^2 + x_{10}^1 w_{1\,10}^2 + x_{11}^1 w_{1\,11}^2 + x_{12}^1 w_{1\,12}^2 + b_1^2 \\
&= 1 \times w_{11}^2 + 1 \times w_{12}^2 + 0 \times w_{13}^2 + 0 \times w_{14}^2 + 1 \times w_{15}^2 + 0 \times w_{16}^2 + 0 \times w_{17}^2 \\
&\quad + 1 \times w_{18}^2 + 0 \times w_{19}^2 + 0 \times w_{1\,10}^2 + 1 \times w_{1\,11}^2 + 0 \times w_{1\,12}^2 + b_1^2 \\
&= 1 \times 0.48 + 1 \times 0.29 + 0 \times 0.58 + 0 \times 0.61 + 1 \times 0.47 + 0 \times 0.18 + 0 \times 0.02 \\
&\quad + 1 \times 0.49 + 0 \times 0.67 + 0 \times 0.62 + 1 \times 0.017 + 0 \times 0.28 + 0 \\
&= 1 \times 0.48 + 1 \times 0.29 + 1 \times 0.47 + 1 \times 0.49 + 1 \times 0.017 + 0 \\
&= 1.277
\end{aligned}$$

$$\begin{aligned}
x_1^2 = \sigma_1^2(z_1^2) &= \tfrac{1}{1+\exp(-z_1^2)} \\
&= \tfrac{1}{1+\exp(-(1.277))} \\
&= 0.782
\end{aligned}$$



$w^2_{11} = 0.48,\ w^2_{12} = 0.29,\ w^2_{13} = 0.58,\ w^2_{14} = 0.61,\ w^2_{15} = 0.47,\ w^2_{16} = 0.18,$

$w^2_{17} = 0.02,\ w^2_{18} = 0.49,\ w^2_{19} = 0.67,\ w^2_{1\,10} = 0.62,\ w^2_{1\,11} = 0.017,\ w^2_{1\,12} = 0.28$

$w^2_{21} = 0.52,\ w^2_{22} = 0.036,\ w^2_{23} = 0.90,\ w^2_{24} = 0.099,\ w^2_{25} = 0.46,\ w^2_{26} = 0.87,$

$w^2_{27} = 0.99,\ w^2_{28} = 0.83,\ w^2_{29} = 0.15,\ w^2_{2\,10} = 0.14,\ w^2_{2\,11} = 0.64,\ w^2_{2\,12} = 0.88$

$w^2_{31} = 0.48,\ w^2_{32} = 0.055,\ w^2_{33} = 0.56,\ w^2_{34} = 0.16,\ w^2_{35} = 0.86,\ w^2_{36} = 0.34,$

$w^2_{37} = 0.40,\ w^2_{38} = 0.06,\ w^2_{39} = 0.66,\ w^2_{3\,10} = 0.72,\ w^2_{3\,11} = 0.077,\ w^2_{3\,12} = 0.29$

$b^2_1 = 0$

$b^2_2 = 0$

$b^2_3 = 0$

The equations below are the 3 weighted inputs of the hidden layer.

$$\begin{cases} z_1^2 &= x_1^1 w_{11}^2 + x_2^1 w_{12}^2 + x_3^1 w_{13}^2 + x_4^1 w_{14}^2 + x_5^1 w_{15}^2 + x_6^1 w_{16}^2 + x_7^1 w_{17}^2 \\ & \quad + x_8^1 w_{18}^2 + x_9^1 w_{19}^2 + x_{10}^1 w_{1\,10}^2 + x_{11}^1 w_{1\,11}^2 + x_{12}^1 w_{1\,12}^2 + b_1^2 \\ z_2^2 &= x_1^1 w_{21}^2 + x_2^1 w_{22}^2 + x_3^1 w_{23}^2 + x_4^1 w_{24}^2 + x_5^1 w_{25}^2 + x_6^1 w_{26}^2 + x_7^1 w_{27}^2 \\ & \quad + x_8^1 w_{28}^2 + x_9^1 w_{29}^2 + x_{10}^1 w_{2\,10}^2 + x_{11}^1 w_{2\,11}^2 + x_{12}^1 w_{2\,12}^2 + b_2^2 \\ z_3^2 &= x_1^1 w_{31}^2 + x_2^1 w_{32}^2 + x_3^1 w_{33}^2 + x_4^1 w_{34}^2 + x_5^1 w_{35}^2 + x_6^1 w_{36}^2 + x_7^1 w_{37}^2 \\ & \quad + x_8^1 w_{38}^2 + x_9^1 w_{39}^2 + x_{10}^1 w_{3\,10}^2 + x_{11}^1 w_{3\,11}^2 + x_{12}^1 w_{3\,12}^2 + b_3^2 \end{cases}$$

We can perceive each z consisting the same inputs x, but different weights w and bias b.
Let's insert the values to the variables.

$$\begin{cases} z_1^2 &= 1 \times w_{11}^2 + 1 \times w_{12}^2 + 0 \times w_{13}^2 + 0 \times w_{14}^2 + 1 \times w_{15}^2 + 0 \times w_{16}^2 + 0 \times w_{17}^2 \\ & \quad + 1 \times w_{18}^2 + 0 \times w_{19}^2 + 0 \times w_{1\,10}^2 + 1 \times w_{1\,11}^2 + 0 \times w_{1\,12}^2 + b_1^2 \\ z_2^2 &= 1 \times w_{21}^2 + 1 \times w_{22}^2 + 0 \times w_{23}^2 + 0 \times w_{24}^2 + 1 \times w_{25}^2 + 0 \times w_{26}^2 + 0 \times w_{27}^2 \\ & \quad + 1 \times w_{28}^2 + 0 \times w_{29}^2 + 0 \times w_{2\,10}^2 + 1 \times w_{2\,11}^2 + 0 \times w_{2\,12}^2 + b_2^2 \\ z_3^2 &= 1 \times w_{31}^2 + 1 \times w_{32}^2 + 0 \times w_{33}^2 + 0 \times w_{34}^2 + 1 \times w_{35}^2 + 0 \times w_{36}^2 + 0 \times w_{37}^2 \\ & \quad + 1 \times w_{38}^2 + 0 \times w_{39}^2 + 0 \times w_{3\,10}^2 + 1 \times w_{3\,11}^2 + 0 \times w_{3\,12}^2 + b_3^2 \end{cases}$$

$$\Longleftrightarrow \begin{cases} z_1^2 &= 1 \times 0.48 + 1 \times 0.29 + 0 \times 0.58 + 0 \times 0.61 + 1 \times 0.47 + 0 \times 0.18 + 0 \times 0.02 \\ & \quad + 1 \times 0.49 + 0 \times 0.67 + 0 \times 0.62 + 1 \times 0.017 + 0 \times 0.28 + 0 \\ z_2^2 &= 1 \times 0.52 + 1 \times 0.039 + 0 \times 0.90 + 0 \times 0.099 + 1 \times 0.46 + 0 \times 0.87 + 0 \times 0.99 \\ & \quad + 1 \times 0.83 + 0 \times 0.15 + 0 \times 0.14 + 1 \times 0.64 + 0 \times 0.88 + 0 \\ z_3^2 &= 1 \times 0.48 + 1 \times 0.055 + 0 \times 0.56 + 0 \times 0.16 + 1 \times 0.86 + 0 \times 0.34 + 0 \times 0.40 \end{cases}$$

$$\Longleftrightarrow \begin{cases} z_1^2 &= 1 \times 0.48 + 1 \times 0.29 + 1 \times 0.47 + 1 \times 0.49 + 1 \times 0.017 + 0 \\ z_2^2 &= 1 \times 0.52 + 1 \times 0.039 + 1 \times 0.46 + 1 \times 0.83 + 1 \times 0.64 + 0 \\ z_3^2 &= 1 \times 0.48 + 1 \times 0.055 + 1 \times 0.86 + 1 \times 0.06 + 1 \times 0.077 + 0 \end{cases}$$

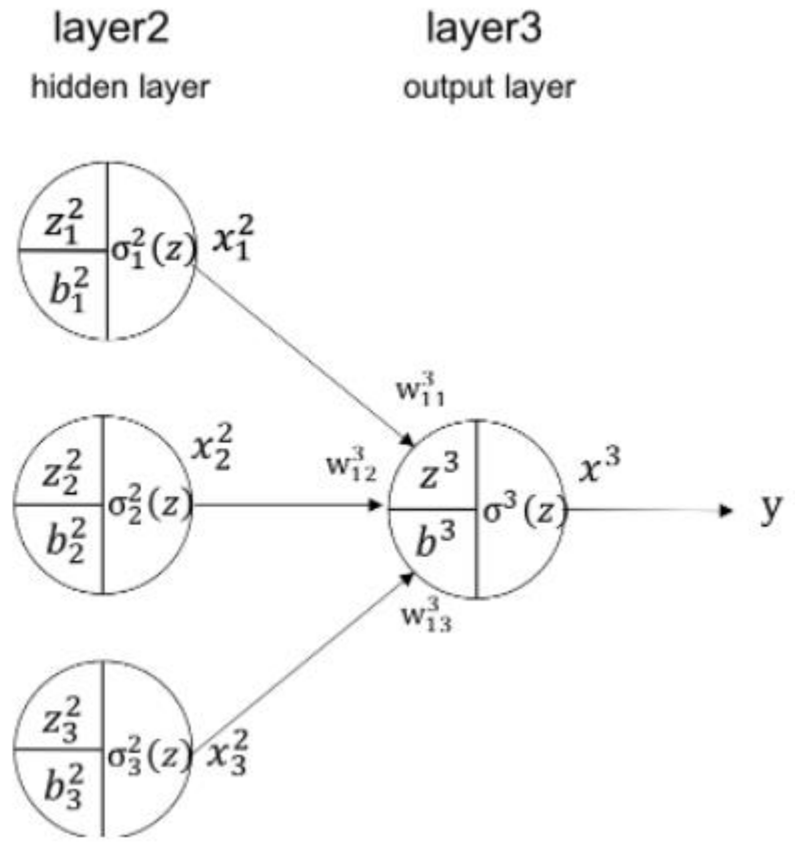
$$\Longleftrightarrow \begin{cases} z_1^2 &= 1.277 \\ z_2^2 &= 2.489 \\ z_3^2 &= 1.532 \end{cases}$$

$$\begin{cases} x_1^2 = \sigma_1^2(z_1^2) = \frac{1}{1+\exp(-z_1^2)} = \frac{1}{1+\exp(-(1.277))} = 0.782 \\ x_2^2 = \sigma_2^2(z_2^2) = \frac{1}{1+\exp(-z_2^2)} = \frac{1}{1+\exp(-(2.489))} = 0.923 \\ x_3^2 = \sigma_3^2(z_3^2) = \frac{1}{1+\exp(-z_3^2)} = \frac{1}{1+\exp(-(1.532))} = 0.822 \end{cases}$$

$$w_1^3 = 0.33,\ w_2^3 = 0.21,\ w_3^3 = 0.96$$

$$b^3 = 0$$

$$\begin{cases} x_1^2 = 0.782 \\ x_2^2 = 0.923 \\ x_3^2 = 0.822 \end{cases}$$



The weighted input in the output layer will be,

$$\begin{aligned} z^3 &= x_1^2 w_{11}^3 + x_2^2 w_{12}^3 + x_3^2 w_{13}^3 + b^3 \\ &= 0.782 \times w_{11}^3 + 0.923 \times w_{12}^3 + 0.822 \times w_{13}^3 + b^3 \\ &= 0.690 \times 0.33 + 0.989 \times 0.21 + 0.550 \times 0.96 + 0 \\ &= 0.963 \end{aligned}$$

The output of the unit will become as follows:

$$x^3 = \sigma^3(z^3) = \frac{1}{1+\exp(-z^3)} = \frac{1}{1+\exp(-(0.963))} = 0.724$$

Sigmoid function has the following properties:

$$\begin{aligned} &1.\ 0 < \sigma(z) < 1 \quad (\lim_{z\to-\infty} \sigma(z) = 0,\ \lim_{z\to\infty} \sigma(z) = 1) \\ &2.\ \sigma(0) = 0.5 \end{aligned}$$

$$\begin{aligned} &When\ \ y = x^3 = \sigma^3(z^3) \\ &\begin{cases} y\ is\ close\ to\ 1 \quad (\ y \geq 0.5\ ) \rightarrow The\ number\ on\ the\ picture\ is\ 1 \\ y\ is\ close\ to\ 0 \quad (\ y < 0.5\ ) \rightarrow The\ number\ on\ the\ picture\ is\ not\ 1. \end{cases} \end{aligned}$$

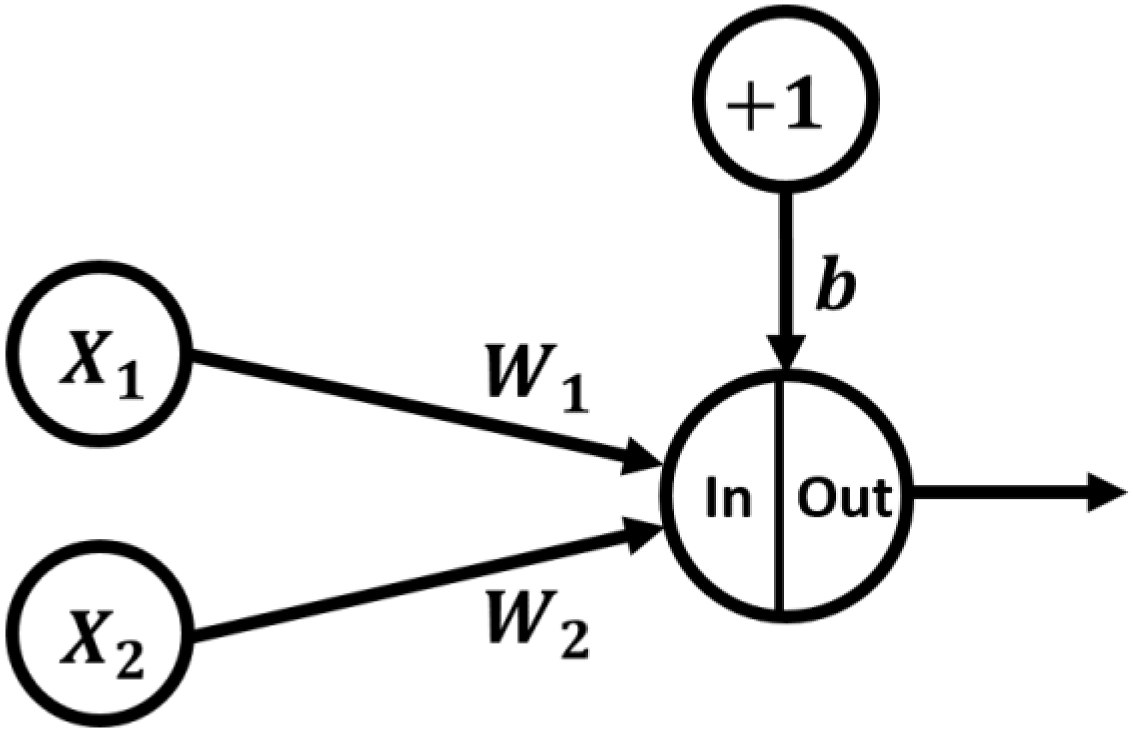
With this in mind, let's consider the calculation result.

$$y = x^3 = \sigma^3(z^3) = 0.724$$

It is $y \geq 0.5$ , thus $y\ is\ close\ to\ 1$. This means $The\ number\ on\ the\ picture\ is\ 1$.

- **Example 3:**

- 1 input layer with 2 inputs ($X_1$ and $X_2$),
- 1 output layer with 1 output. There are no hidden layers.
- The weights of the inputs are $W_1$ and $W_2$, respectively.
- The bias is treated as a new input neuron to the output neuron which has a fixed value +1 and a weight b. Both the weights and biases could be referred to as **parameters**.



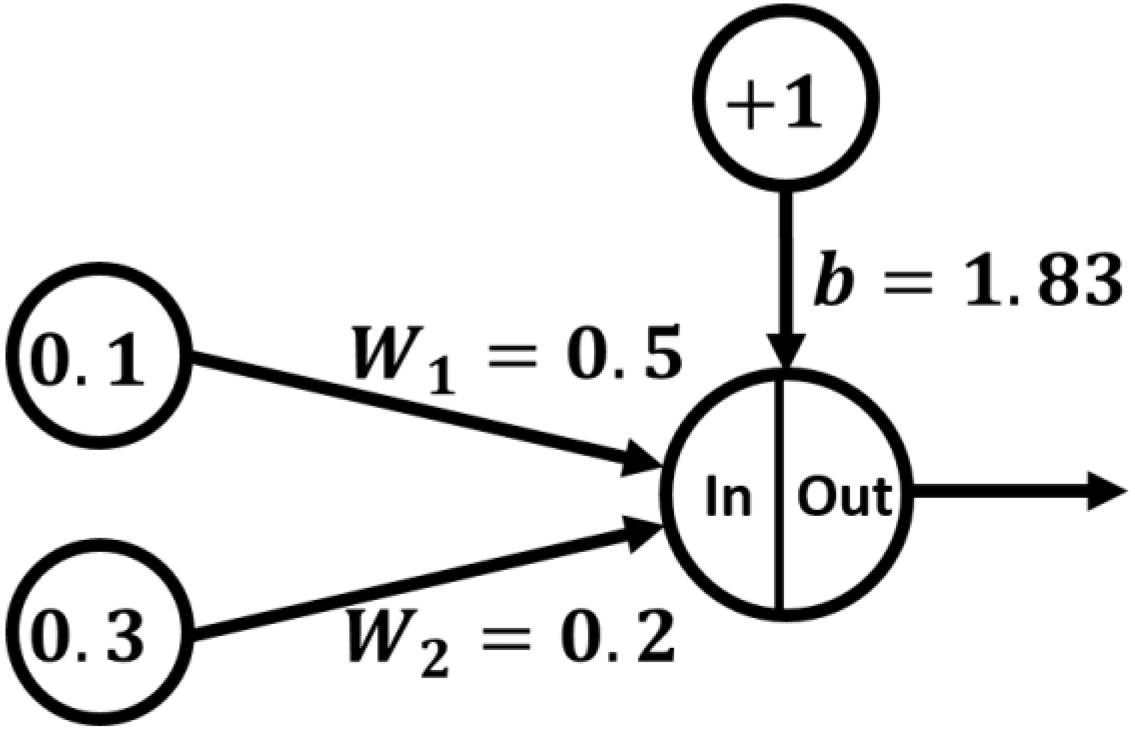
$$f(s) = \frac{1}{1+e^{-s}}$$

- s is the sum of products (SOP) between each input and its corresponding weight:
- $s=X_1 * W_1 + X_2 * W_2 + b$

| X1 | X2 | Desired Output |
|---|---|---|
| 0.1 | 0.3 | 0.03 |

Assume that the initial values for both weights and bias are like in the next table.

| W1 | W2 | b |
|---|---|---|
| 0.5 | 0.2 | 1.83 |



## Forward pass

- $s=X_1 * W_1 + X_2 * W_2 + b$
- $s=0.1 * 0.5 + 0.3 * 0.2 + 1.83$
- $s=1.94$
- The value 1.94 is then applied to the activation function (sigmoid), which results in the value 0.874352143.

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

$$f(s) = \frac{1}{1+e^{-1.94}}$$

$$f(s) = \frac{1}{1+0.143703949}$$

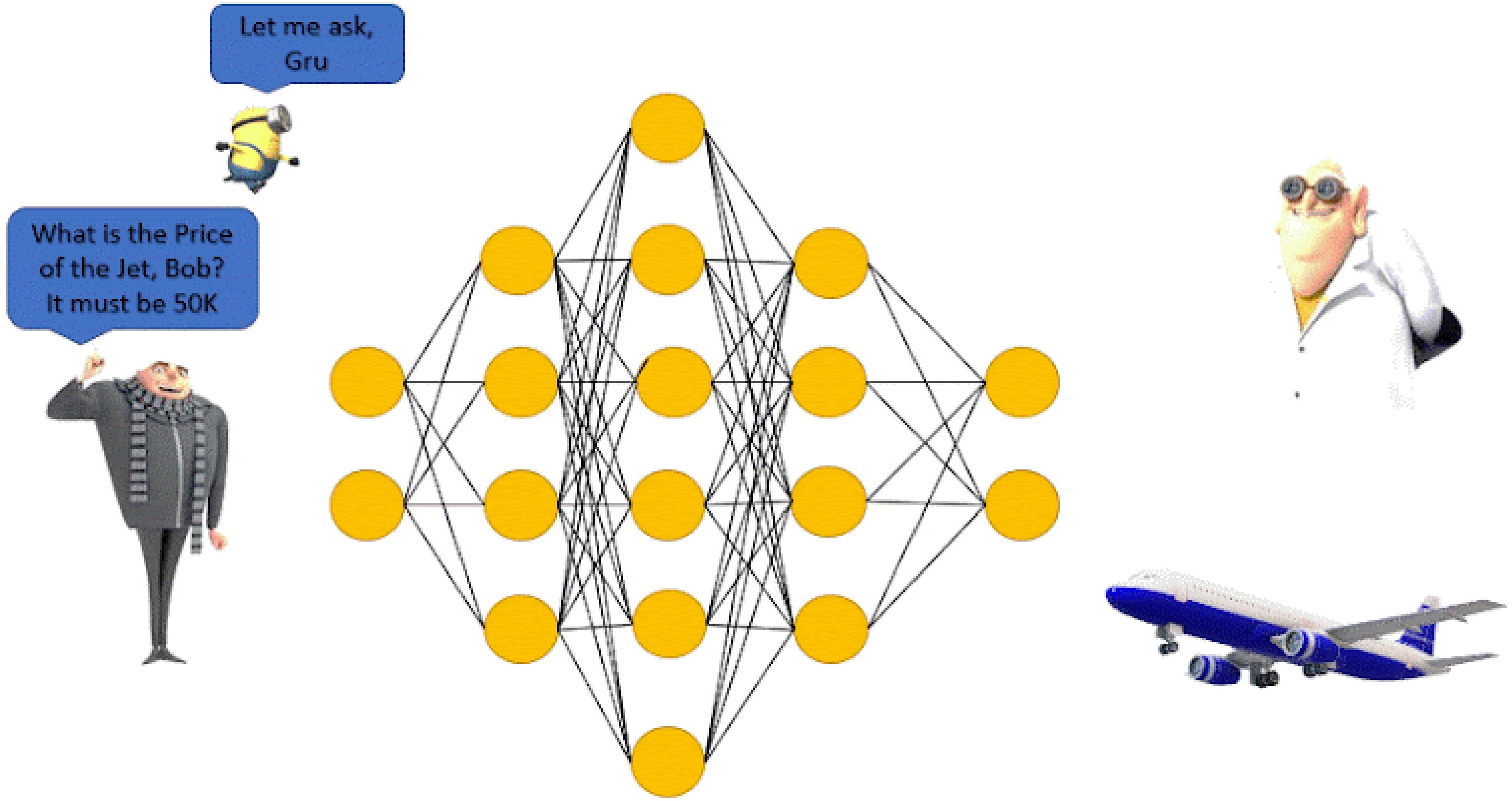
$$f(s) = \frac{1}{1.143703949}$$

$$f(s) = 0.874352143$$

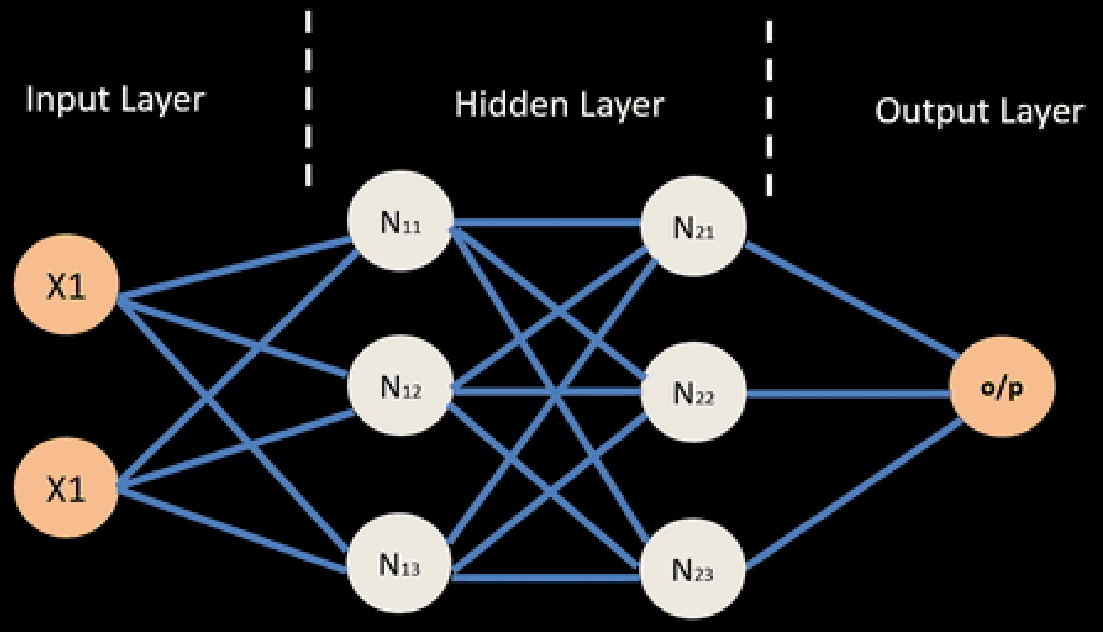
- Actual output : 0.03
- Predicted output : 0.874352143
- It’s obvious that there’s a difference between the desired and expected output. But why?

## Why backpropagation????

- The backpropagation algorithm is one of the algorithms responsible for updating network weights with the objective of reducing the network error. It’s quite important.
- If the current error is high, the network didn’t learn properly from the data.
- What does this mean? It means that the current set of weights isn’t accurate enough to reduce the network error and make accurate predictions.
- As a result, we should update network weights to reduce the network error.


Let me ask, Gru
What is the Price of the Jet, Bob? It must be 50K

# Neural Network – Backpropagation

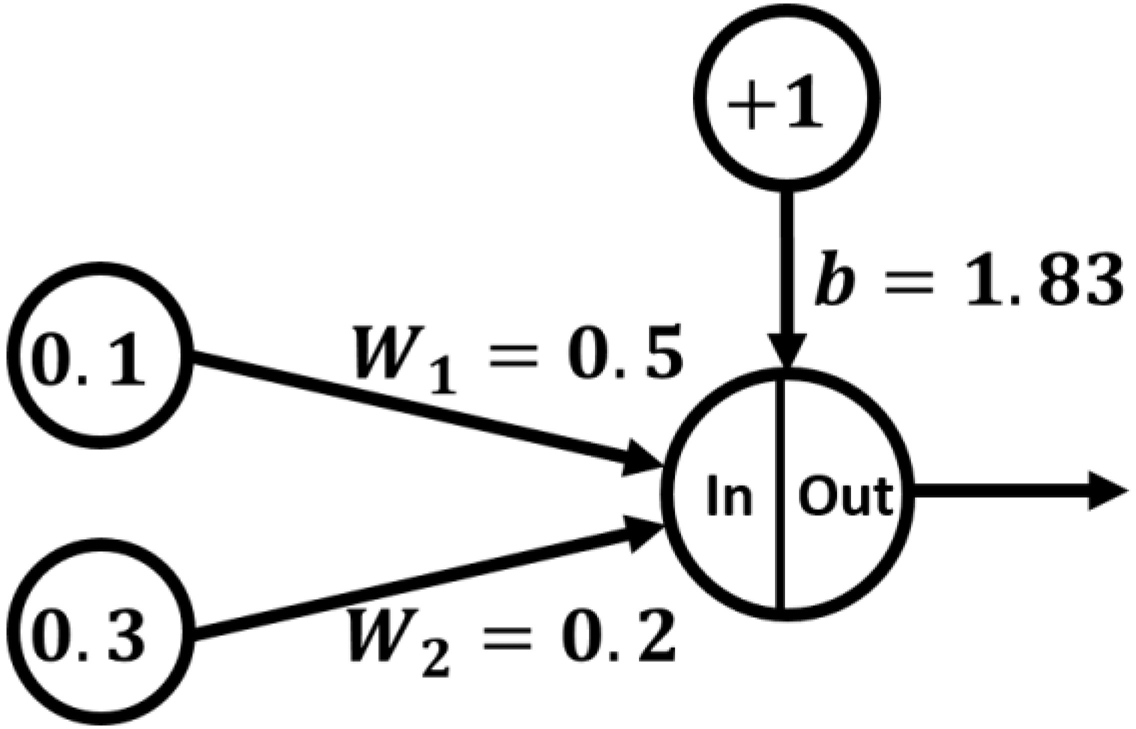


- s is the sum of products (SOP) between each input and its corresponding weight:
- $s=X_1 * W_1 + X_2 * W_2 + b$

| X1 | X2 | Desired Output |
|---|---|---|
| 0.1 | 0.3 | 0.03 |

Assume that the initial values for both weights and bias are like in the next table.

| W1 | W2 | b |
|---|---|---|
| 0.5 | 0.2 | 1.83 |



## Forward pass

- $s=X_1 * W_1 + X_2 * W_2 + b$
- $s=0.1* 0.5+ 0.3*0.2+1.83$
- $s=1.94$
- The value 1.94 is then applied to the activation function (sigmoid), which results in the value 0.874352143.

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

$$f(s) = \frac{1}{1+e^{-1.94}}$$

$$f(s) = \frac{1}{1+0.143703949}$$

$$f(s) = \frac{1}{1.143703949}$$

$$f(s) = 0.874352143$$

## Error of our network based on an error function

- The error functions tell how close the predicted output(s) are from the desired output(s).
- The optimal value for error is **zero**, meaning there’s no error at all, and both desired and predicted results are identical.
- One of the error functions is the **squared error function**

$$E = \frac{1}{2}(desired - predicted)^2$$

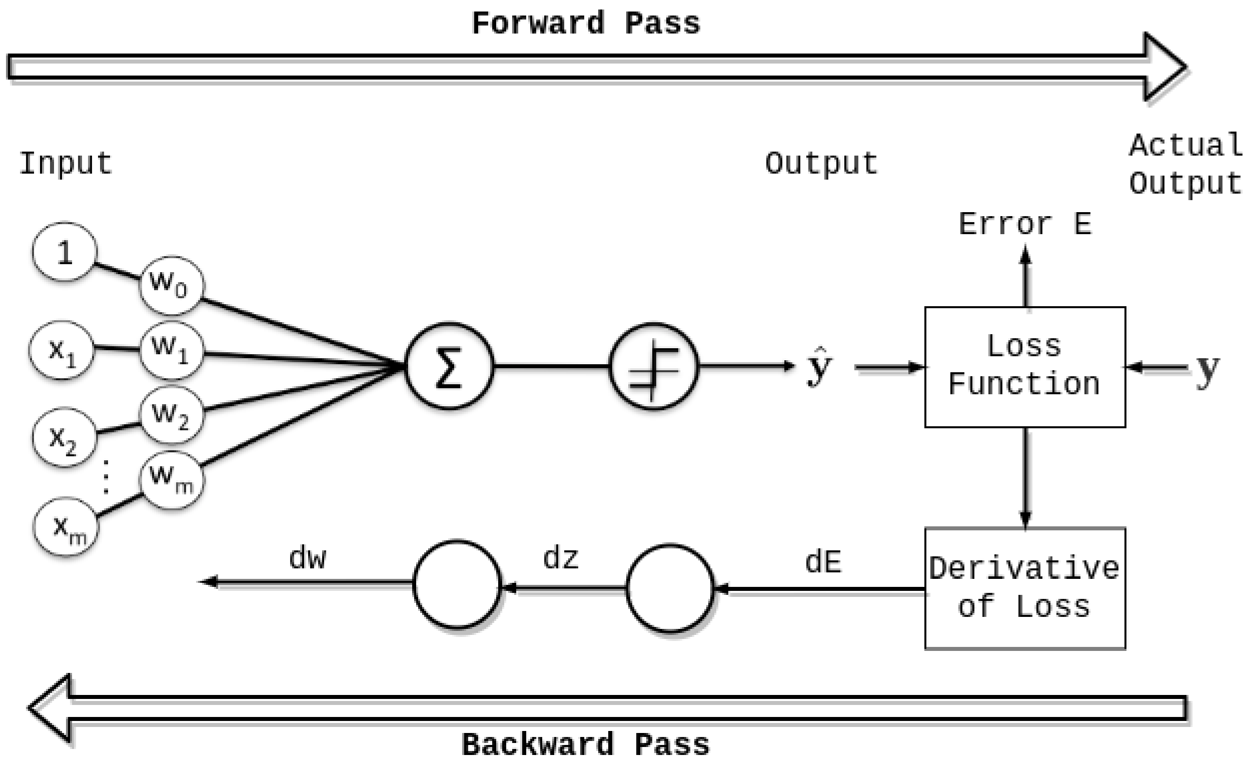
$$E = \frac{1}{2}(0.03 - 0.874352143)^2$$

$$E = \frac{1}{2}(-0.844352143)^2$$

$$E = \frac{1}{2}(0.712930542)$$

$$E = 0.356465271$$

- Knowing that there's an error, what should we do? We should minimize it.
- To minimize network error, we must change something in the network.
- Remember that the only parameters we can change are the weights and biases.
- We can try different weights and biases, and then test our network.
- We calculate the error, then the forward pass ends, and we should start the **backward pass** to calculate the derivatives and update the parameters.



- The parameters-update equation just depends on the **learning rate** to update the parameters. It changes all the parameters in a direction opposite to the error.
- But, using the backpropagation algorithm, we can know how each single weight correlates with the error. This tells us the effect of each weight on the prediction error. That is, which parameters do we increase, and which ones do we decrease to get the smallest prediction error?
- For example, the backpropagation algorithm could tell us useful information, like that increasing the current value of W1 by 1.0 increases the network error by 0.07. This shows us that a smaller value for W1 is better to minimize the error.

- One important operation used in the backward pass is to calculate derivatives
- Y=x^2Z+H

$$\frac{\partial Y}{\partial X} = \frac{\partial}{\partial X}(X^2 Z + H)$$

$$\frac{\partial Y}{\partial X} = 2XZ + 0$$

$$\frac{\partial Y}{\partial X} = 2XZ$$

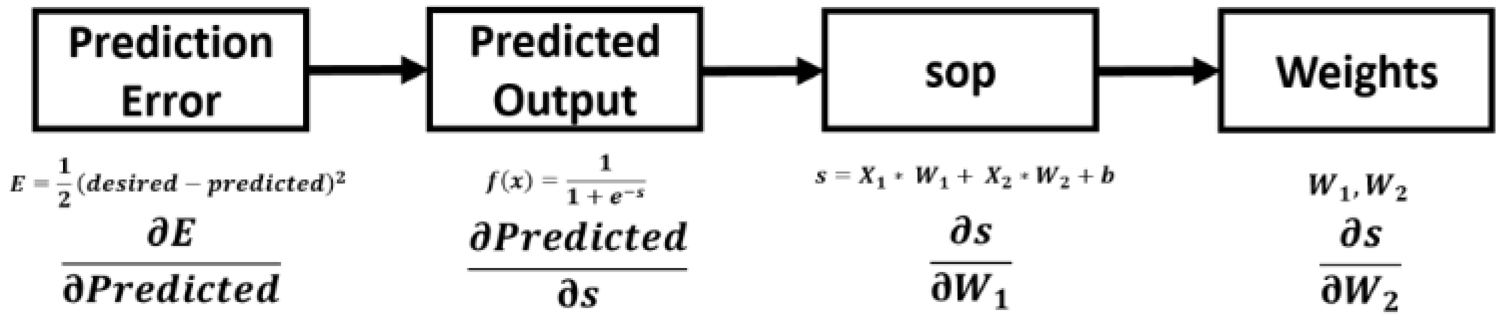
- our target is to calculate $\partial E/W_1$ and $\partial E/W_2$ as we have just two weights $W_1$ and $W_2$. Let’s calculate them.

- The prediction error is calculated based on this equation:

$$E = \frac{1}{2}(desired - predicted)^2$$

- The **desired** term in the previous equation is a constant, so there’s no chance for reaching parameters through it.
- The **predicted** term is calculated based on the sigmoid function, like in the next equation: $f(s) = \frac{1}{1+e^{-s}}$

- Again, the equation for calculating the predicted output doesn’t have any parameter. But there’s still variable s (SOP) that already depends on parameters for its calculation, according to this equation:
- $s=X_1* W_1+ X_2*W_2+b$


Prediction Error
Predicted Output
sop
Weights
$E = \frac{1}{2}(desired - predicted)^2$
$f(x) = \frac{1}{1+e^{-s}}$
$s = X_1 * W_1 + X_2 * W_2 + b$
$W_1, W_2$
$\frac{\partial E}{\partial Predicted}$
$\frac{\partial Predicted}{\partial s}$
$\frac{\partial s}{\partial W_1}$
$\frac{\partial s}{\partial W_2}$

1. Network error W.R.T the predicted output.
2. Predicted output W.R.T the SOP.
3. SOP W.R.T each of the 3 parameters.

- As a total, there are four intermediate partial derivatives:
- $\partial E/\partial Predicted$, $\partial Predicted/\partial_s$, $\partial_s/W_1$ **and** $\partial_s/W_2$
- To calculate the derivative of the error W.R.T the weights, simply multiply all the derivatives in the chain from the error to each weight, as in the next 2 equations:
- $\partial E/W_1 = \partial E/\partial Predicted * \partial Predicted/\partial_s * \partial_s/W_1$
- $\partial EW_2 = \partial E/\partial Predicted * \partial Predicted/\partial s * \partial_s/W_2$

- For the derivative of the error W.R.T the predicted output:
- $\partial E/\partial Predicted=\partial/\partial Predicted(1/2(desired\text{-}predicted)^2)$
- $=2*1/2(desired\text{-}predicted)^{2-1}*(0-1)$
- $=(desired\text{-}predicted)*(-1)$
- $=predicted\text{-}desired$
- By substituting by the values:
- $\partial E/\partial Predicted=predicted\text{-}desired=0.874352143-0.03$
- $\partial E/\partial Predicted=0.844352143$

## Remember: the quotient rule can be used to find the derivative of the sigmoid function as follows:

$$Sigmoid = 1/(1 + \mathrm{e}^{-x})$$

$$\frac{\partial(1/(1 + \mathrm{e}^{-x}))}{\partial x} = 1/(1 + \mathrm{e}^{-x}) \times (1 - 1/(1 + \mathrm{e}^{-x}))$$

$$\frac{\partial Sigmoid}{\partial x} = Sigmoid \times (1 - Sigmoid)$$

- *0.874352143(1-0.874352143)*
- *=0.874352143(0.125647857)*
- *∂Predicted/∂s=0.109860473*

For the derivative of SOP W.R.T W1:

$\partial_s/W_1=\partial/\partial W_1(X_1* W_1+ X_2*W_2+b)$

$=1*X_1*(W_1)^{(1-1)}+ 0+0$

$=X_1*(W_1)^{(0)}$

$=X_1(1)$

$\partial_s/W_1=X_1$

By substituting by the values:

$\partial_s$/W1=$X_1$=0.1

For the derivative of SOP W.R.T W2:

$\partial_s/W_2=\partial/\partial W_2(X_1* W_1+ X_2*W_2+b)$

$=0+1*X_2*(W_2)^{(1-1)}+0$

$=X_2*(W_2)^{(0)}$

$=X_2(1)$

$\partial_s/W_2=X_2$

By substituting by the values:

$\partial_s/W_2=X_2=0.3$

- For the derivative of the error W.R.T W1:
- $\partial E/W_1=\partial E/\partial Predicted* \partial Predicted/\partial_s* \partial_s/W_1$
- $\partial E/W_1=0.844352143*0.109860473*0.1$
- $\partial E/W_1=0.009276093$
- For the derivative of the error W.R.T W2:
- $\partial EW_2=\partial E/\partial Predicted* \partial Predicted/\partial s* \partial_s/W_2$
- $\partial E/W_2=0.844352143*0.109860473*0.3$
- $\partial E/W_2=0.027828278$
- Finally, there are two values reflecting how the prediction error changes with respect to the weights:
- *0.009276093 for* $W_1$
- *0.027828278 for* $W_2$
- What do these values mean? These results need interpretation.

## Conclusions based on derivatives

- Derivative sign
- Derivative magnitude (DM)
- If the derivative sign is **positive**, that means increasing the weight increases the error. In other words, decreasing the weight decreases the error.
- If the derivative sign is **negative**, increasing the weight decreases the error. In other words, if it’s negative then decreasing the weight increases the error.
- But by how much does the error increase or decrease? The derivative magnitude answers this question.
- For positive derivatives, increasing the weight by p increases the error by DM*p.
- For negative derivatives, increasing the weight by p decreases the error by DM*p.

- Because the result of the $\partial E/W_1$ derivative is positive, this means if W1 increases by 1, then the total error increases by 0.009276093.
- Because the result of the $\partial E/W_2$ derivative is positive, this means that if W2 increases by 1 then the total error increases by 0.027828278.

## Updating weights

For $W_1$:

$W_{1new}=W_1-\eta*\partial E/W_1$

$=0.5-0.01*0.009276093$

$W_{1new}=0.49990723907$

For $W_2$:

$W_{2new}=W_2-\eta*\partial E/W_2$

$=0.2-0.01*0.027828278$

$W_{2new}= 0.1997217172$

Note that the derivative is subtracted (not added) from the old value of the weight, because the derivative is positive.

The new values for the weights are:

$W_1=0.49990723907$

$W_2= 0.1997217172$

**Parameters update equation**

$W_{(n+1)}=W(n)+\eta[d(n)-Y(n)]X(n)$

$=[1.83, 0.5, 0.2]+0.01[0.03-0.874352143][+1, 0.1, 0.3]$

$=[1.83, 0.5, 0.2]+0.01[-0.844352143][+1, 0.1, 0.3]$

$=[1.83, 0.5, 0.2]+-0.00844352143[+1, 0.1, 0.3]$

$=[1.83, 0.5, 0.2]+[-0.008443521, -0.000844352, -0.002533056]$

$=[1.821556479, 0.499155648, 0.197466943]$

| W1new | W2new | bnew |
|---|---|---|
| 0.197466943 | 0.499155648 | 1.821556479 |

new forward pass calculations:

$s=X_1* W_1+ X_2*W_2+b$

$s=0.1*0.49990723907+ 0.3*0.1997217172+1.821556479$

$s=1.931463718067$

$f(s)=1/(1+e^{-s})$

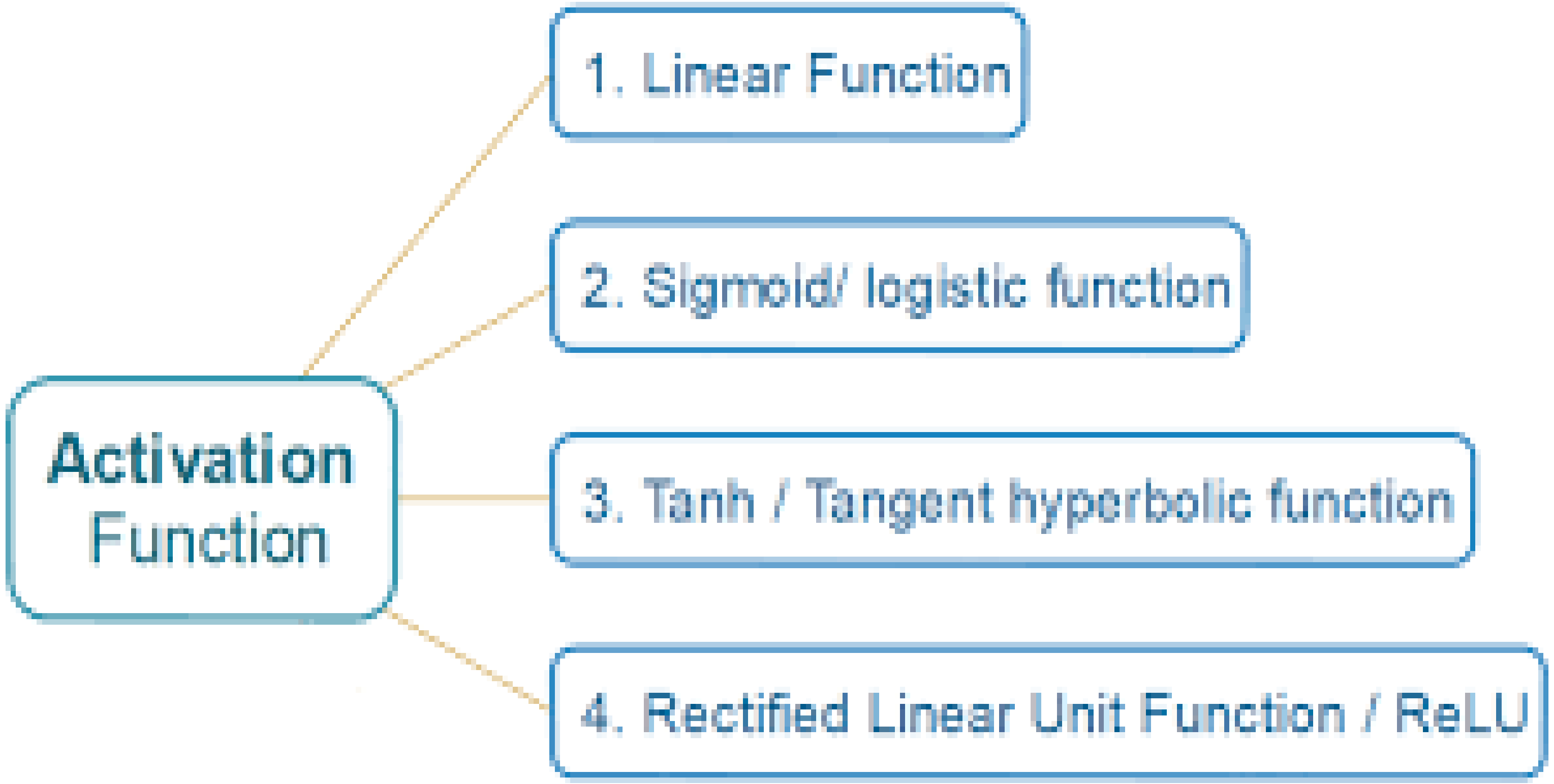
$f(s)=1/(1+e^{-1.931463718067})$

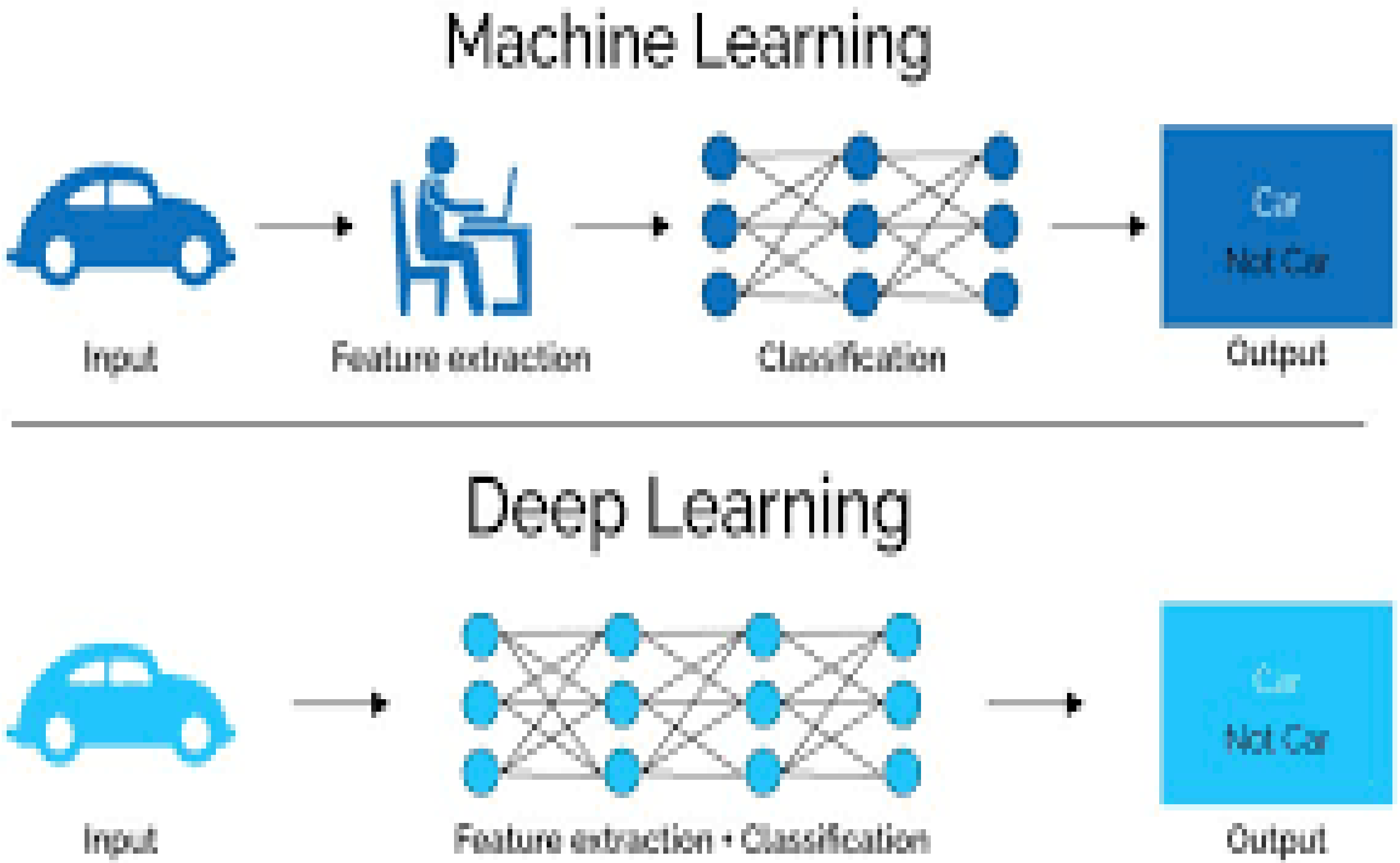
$f(s)=0.873411342830056$

$E=1/2(0.03-0.873411342830056)^2$

$E=0.3556713466071990 7$

- new error (0.35567134660719907)
- old error (0.356465271)
- reduction of 0.00079392439280009043.
- As long as there’s a reduction, we’re moving in the right direction.
- The error reduction is small because we’re using a small learning rate (0.01).
- The forward and backward passes should be repeated until the error is 0 or for a number of epochs (i.e. iterations).


Activation Function
1. Linear Function
2. Sigmoid/ logistic function
3. Tanh / Tangent hyperbolic function
4. Rectified Linear Unit Function / ReLU


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output