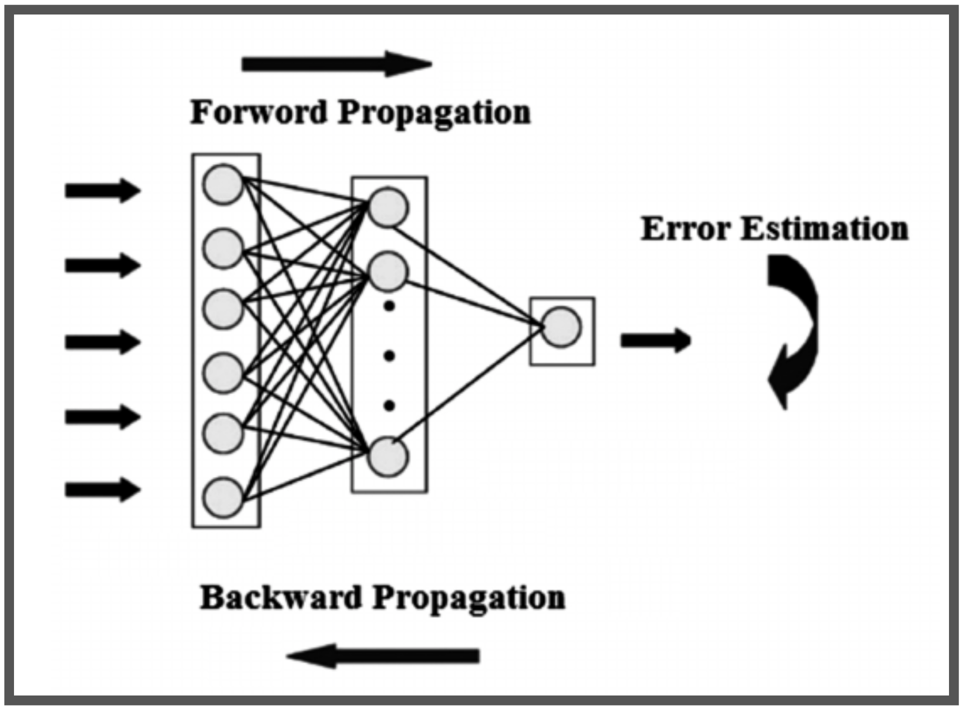
Forward And Backward Propogation

Every Neural Network has 2 main parts:

```
1. Feed Forward Propogation/Forward Propogation.
2. Backward Propogation/Back propogation.
```



Feed Forward Propogation:

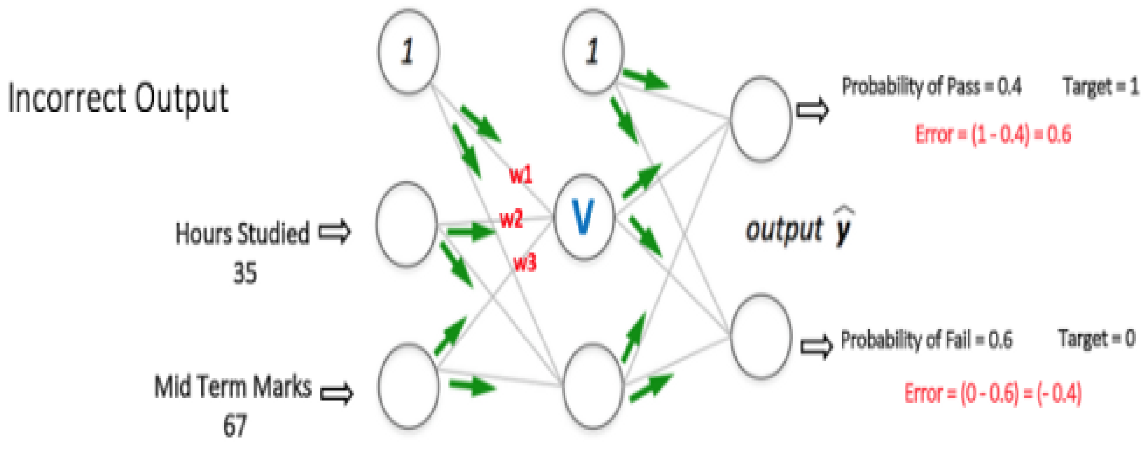
- All weights in the network are randomly assigned. Assume the weights of the connections from the inputs to that node are w1, w2 and w3.
- Here I take an example to better understand of this two concept
- Let take an example of if you study 35 Hour per day then you definitly Pass the exam with 67 Mark. Now we apply this.

    - Input to the network = [35, 67]
    - Desired output from the network (target) = [1, 0] 1 means PASS and 0 means Fail

Then output V from the node in consideration can be calculated as below (f is an activation function such as sigmoid):

$$V = f(1*w1 + 35*w2 + 67*w3)$$

- Similarly, outputs from the other node in the hidden layer is also calculated. The outputs of the two nodes in the hidden layer act as inputs to the two nodes in the output layer. This enables us to calculate output probabilities from the two nodes in output layer.



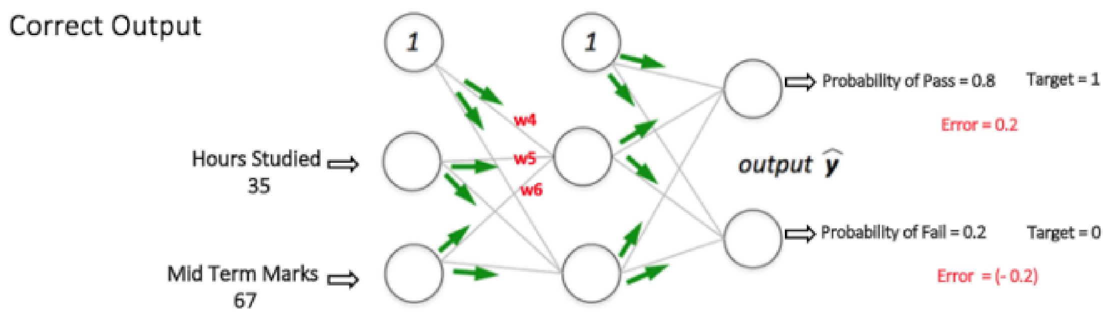
- Suppose the output probabilities from the two nodes in the output layer are 0.4 and 0.6 respectively (since the weights are randomly assigned, outputs will also be random). We can see that the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively), hence the network in above Figure is said to have an ‘Incorrect Output’. As it give output as fail but it not happen that one study 35 hr and secure 67.
- here Weight are randomly assigned so now we do Back Propagation and with Weight Updation.

Back Propagation and Weight Updation:

- Here We calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation to calculate the gradients.
- Then we use an optimization method such as Gradient Descent to ‘adjust’ all weights in the network with an aim of reducing the error at the output layer.

$$\underset{\substack{\uparrow \\ \text{New weight}}}{{}^{*}W_x} = \overset{\substack{\text{Old weight} \\ \downarrow}}{W_x} - \underset{\substack{\uparrow \\ \text{Learning} \\ \text{rate}}}{a} \left(\overset{\substack{\text{Derivative of Error} \\ \text{with respect to weight} \\ \downarrow}}{\frac{\partial Error}{\partial W_x}} \right)$$

- If we now input the same example to the network again, the network should perform better than before since the weights have now been adjusted to minimize the error in prediction.



- As shown in above Figure, the errors at the output nodes now reduce to [0.2, -0.2] as compared to [0.6, -0.4] earlier. This means that our network has learnt to correctly classify our first training example.
- We repeat this process with all other training examples in our dataset. Then, our network is said to have learnt those examples.

```
1. Thanks To https://medium.com/@purnasaigudikandula/a-beginner-intro-to-
neural-networks-
543267bda3c8#:~:text=Neural%20networks%20are%20set%20of,the%20functioning%20of%
20human%20brian.&text=That's%20how%20similar%20the%20Neural,and%20outputs%20wha
t%20it%20is.
2. http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html
```