Input layer:

- Number of Neurons = Number of input features
- Input neurons does not have any weights and Bias
- It simply pass the inputs to the next layers

Hidden layer

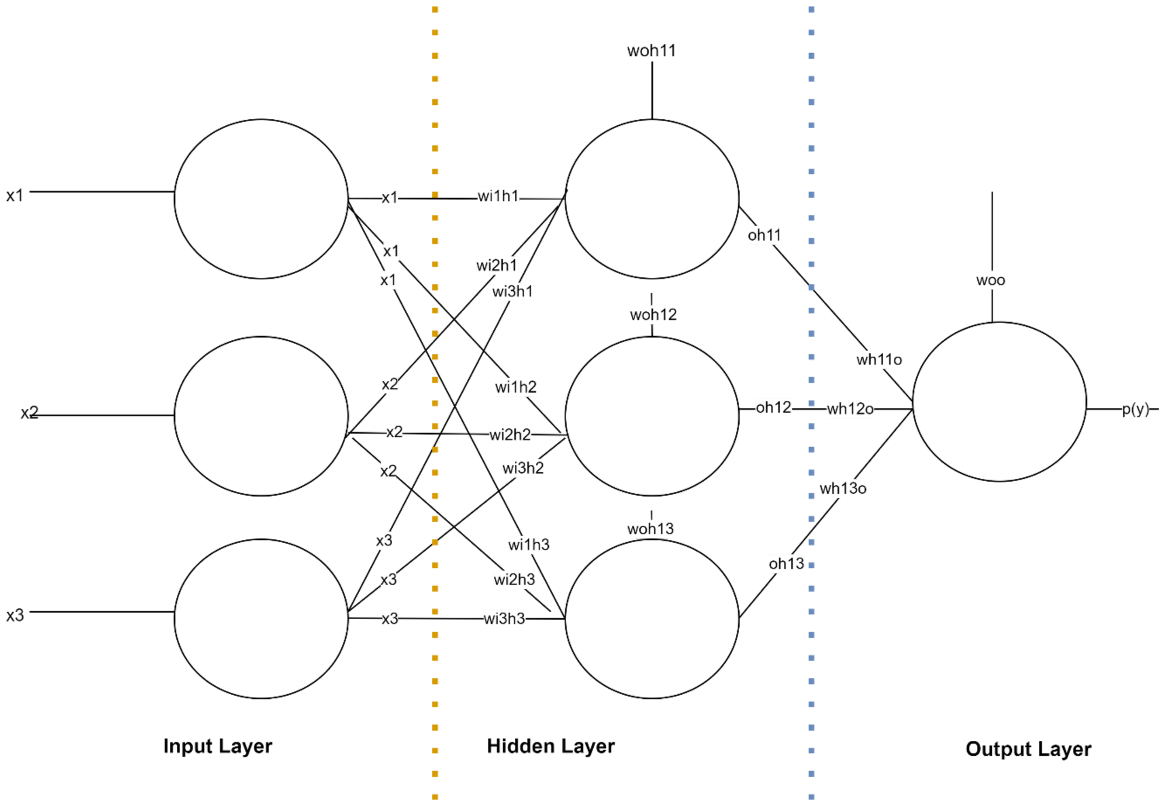
- we have two questions
    - How many hidden layers you want to choose
    - How many neurons in each hidden layer
- The more hidden layer === Analogy === Model complexity is more
- The more hidden layer === Analogy === the more depth in Decision tree
- The more hidden layers : Model might becomes overfit
- The less hidden layers: Model might become underfit
- The hidden layer ===== Analogy ===== Bias − Variance tradeoff
- Hidden layer has weights and bias
- Input for hidden layer: $W * X + W_o$
- output for hidden layer: $Activation(W * X + W_o)$

Output layer:

- how many neurons need to choose
- Bi clasification problem: Number of neurons = One neuron
- Multi classification: Number of Neurons = Number of lables
- You want identify covide zone: RED GREEN YELLOW : 3 Lables = 3neurons
- Output layer has bias and also weights

ANN: Artifical Neural Network: Perceptron (I − H − O)

DNN: Deep Neural Network: Multi Layer perceptron (I − H1 − H2 − H3 − O)



$w_{i1h1} = Input\ layer\ Neuron\ -\ 1\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 1$

$w_{i1h2} = Input\ layer\ Neuron\ -\ 1\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 2$

$w_{i1h3} = Input\ layer\ Neuron\ -\ 1\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 3$

$w_{i2h1} = Input\ layer\ Neuron\ -\ 2\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 1$

$w_{i2h2} = Input\ layer\ Neuron\ -\ 2\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 2$

$w_{i2h3} = Input\ layer\ Neuron\ -\ 2\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 3$

$w_{i3h1} = Input\ layer\ Neuron\ -\ 3\ to\ \ Hidden\ layer\ of\ Neuron\ -\ 1$

$w_{i3h2} = Input\ layer\ Neuron\ -\ 3\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 2$

$w_{i3h3} = Input\ layer\ Neuron\ -\ 3\ \ to\ \ Hidden\ layer\ of\ Neuron\ -\ 3$

$$w_{oh11} = Bias\ of\ Hidden\ layer - 1\ of\ Neuron - 1$$

$$w_{oh12} = Bias\ of\ Hidden\ layer - 1\ of\ Neuron - 2$$

$$w_{oh13} = Bias\ of\ Hidden\ layer - 1\ of\ Neuron - 3$$

$$o_{h11} = output\ of\ Hiddenlayer - 1\ of\ Nueron - 1$$

$$o_{h12} = output\ of\ Hiddenlayer - 1\ of\ Nueron - 2$$

$$o_{h13} = output\ of\ Hiddenlayer - 1\ of\ Nueron - 3$$

$$o_{h11} = Activation(w_{oh11} + w_{i1h1} * x_1 + w_{i2h1} * x_2 + w_{i3h1} * x_3)$$

$$o_{h12} = Activation(w_{oh12} + w_{i1h2} * x_1 + w_{i2h2} * x_2 + w_{i3h2} * x_3)$$

$$o_{h13} = Activation(w_{oh13} + w_{i1h3} * x_1 + w_{i2h3} * x_2 + w_{i3h3} * x_3)$$

$$w_{oh11} + w_{i1h1} * x_1 + w_{i2h1} * x_2 + w_{i3h1} * x_3$$

$$w_{oh12} + w_{i1h2} * x_1 + w_{i2h2} * x_2 + w_{i3h2} * x_3$$

$$w_{oh13} + w_{i1h3} * x_1 + w_{i2h3} * x_2 + w_{i3h3} * x_3$$

$$A\left[\underset{3X3}{\begin{bmatrix} W_{i_1h_1} & W_{i_2h_1} & W_{i_3h_1} \\ W_{i_1h_2} & W_{i_2h_2} & W_{i_3h_2} \\ W_{i_1h_3} & W_{i_2h_3} & W_{i_3h_3} \end{bmatrix}} X \underset{3X1}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} + \begin{bmatrix} W_{0h_{11}} \\ W_{0h_{12}} \\ W_{0h_{13}} \end{bmatrix}\right] = \begin{bmatrix} O_{h_{11}} \\ O_{h_{12}} \\ O_{h_{13}} \end{bmatrix}$$

$$W * X + W_o = W^T . X + W_o$$

$w_{h110} = Hidden\ layer - 1\ of\ Neuron - 1\ \ to\ \ Output$

$w_{h120} = Hidden\ layer - 1\ of\ Neuron - 2\ \ to\ \ Output$

$w_{h130} = Hidden\ layer - 1\ of\ Neuron - 3\ \ to\ \ Output$

$$w_{o_{out}} = Output\ bias$$

$$Final\ output = \ Activation(w_{o_{out}} + O_{h11} * w_{h110} + O_{h12} * w_{h120} + O_{h13} * w_{h130})$$

$Final\ output\ \ range\ depends\ on\ Activation\ Function$

$Assume\ that\ Activation\ function\ is\ \ Sigmoid: \ \ 0\ to\ 1$

$That\ we\ called\ Logits,\ assume\ that\ the\ logit\ value > 0.5\ \ :\ Yes$

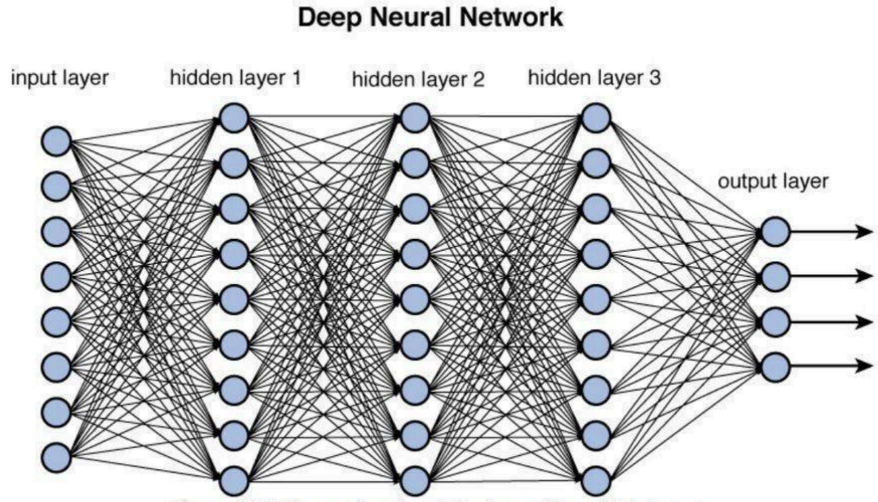


Figure 12.2 Deep network architecture with multiple layers.

$Input\ layer:\ \ 10\ inputs\ ======\ \ >\ 10N$

$Hidden\ layer-1:\ \ 10\ Neurons$

$Output\ layer: 10\ classes\ ======\ \ >\ 10N$

Layer	Params
Input layers : 10N	0
Hidden layer : 10N	10*10+10=110
Outpu layer: 10N	10*10+10=110
Total	220

$Input\ layer:\ \ 28\ inputs\ ======\ \ >\ 28N$

$Hidden\ layer-1:\ \ 128\ Neurons$

$Output\ layer: 10\ classes\ ======\ \ >\ 10N$

Layer	Params
Input layers : 28N	0
Hidden layer : 128N	28*128+128=3712
Outpu layer: 10N	128*10+10=1290
Total	5002

Layer	Params
Input layers : 8N	0
Hidden layer1 : 9N	8*9+9=81
Hidden layer2 : 9N	9*9+9=90
Hidden layer3 : 9N	9*9+9=90
Outpu layer: 4N	9*4+4=40
Total	301

$Input\ layer:\ \ 28\ X\ 28\ ======\ \ >\ 784N$

$Hidden\ layer-1:\ \ 128\ Neurons$

$Output\ layer: 10\ classes\ ======\ \ >\ 10N$

Layer	Params

Input layers : 784N	0
Hidden layer : 128N	784*128+128=100480
Outpu layer: 10N	128*10+10=1290
Total	101770

1) How to choose the weights
2) How to choose the activation Functions
3) How to choose hidden layers
   The more hidden layers : Model becomes overfit
   How to avoid the overfit
4) If we choose more neurons in hidden layers
   More params to be trained
   Complexity
   How to speed up the process

If you take any Deep learning or any NN problem what is the Goal?

Find the suitable coefficients, to minimise the cost functions

At starting we intialise weights randomly

Then will pass the inputs , that inputs multiply with weights(Intilaised randomly)

Finally we will get output ===== > Forward Prpogation

You will calculate the error , error will be huge

Who is the responsible (which layer weights are responsible)

Credit Assignment: Every layer every weight is responsible

Then will go back and update the weights ===== > Back prpogation

Input will go through weights and will give output

will calculate error, to reduce the error will go back and update the weight

Forward Prpogation + Back prpogation = Epoch

In Linear regreesion How to find the coeff: *OLS*

In Deep learning: *Gradient Descent algorithm*