Deep Learning

* DL is a subset of ML.
* It is a field that is based on learning and improving on its own by examining computer algorithms.

Neural networks:-

* Neural networks learn to perform tasks by being exposed to various datasets and examples without any task specific rules.
* Neural networks are based on computational models for threshold logic.
* components of neural network
    i) neurons
    ii) connections
    iii) weights
    iv) Biases
    v) Propagation function
    vi) Learning rule.
* Types of neural networks
    i) Artificial neural network (ANN)
    ii) Convolutional neural network (CNN)
    iii) Recurrent neural network (RNN)

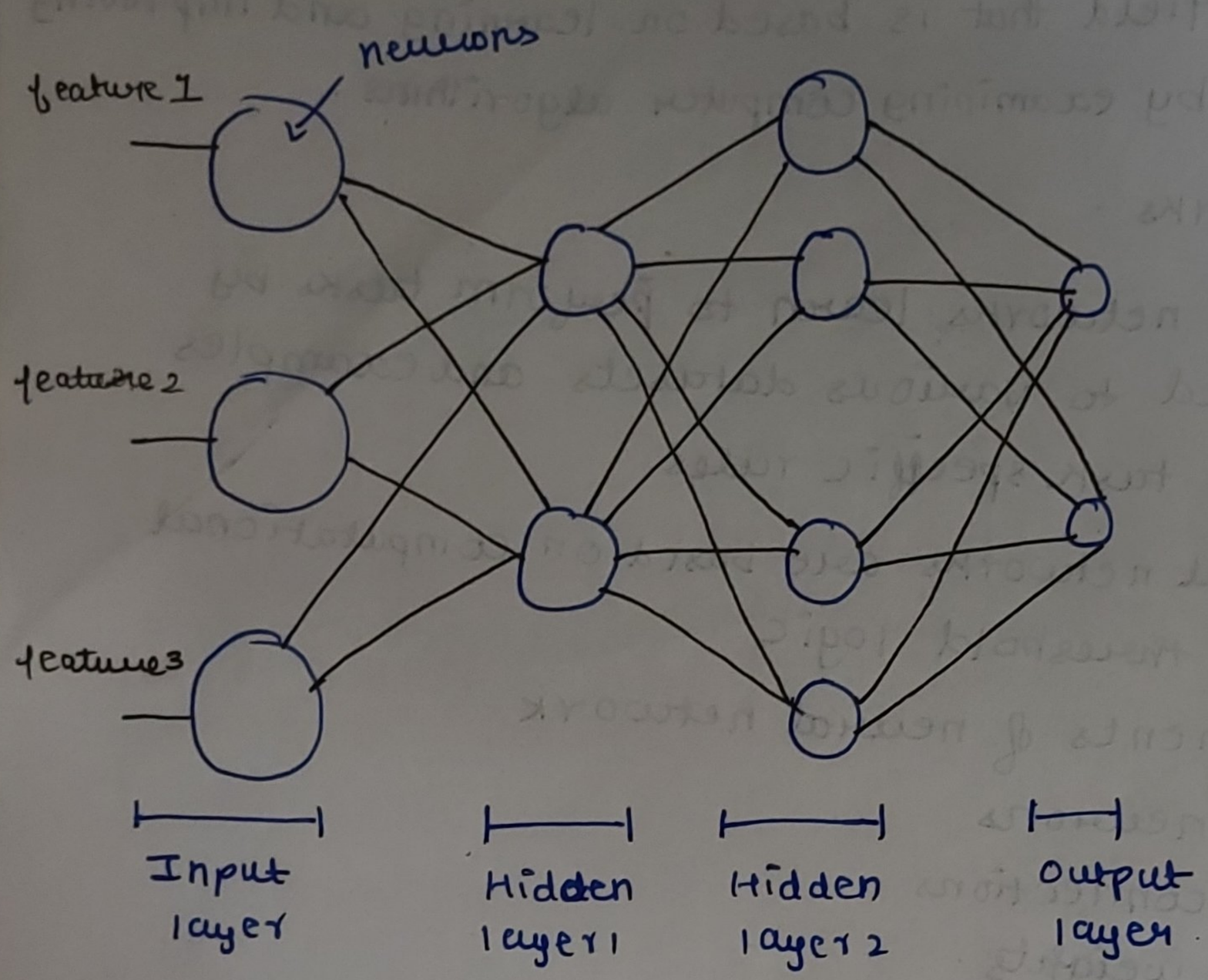
Artificial neural network:- (ANN)

* Similar to human brains that have neurons likewise ANN have neurons that are interconnected to one another in various layers of network.
* These neurons are known as nodes.

Scanned by TapScanner

Architecture of ANN:-

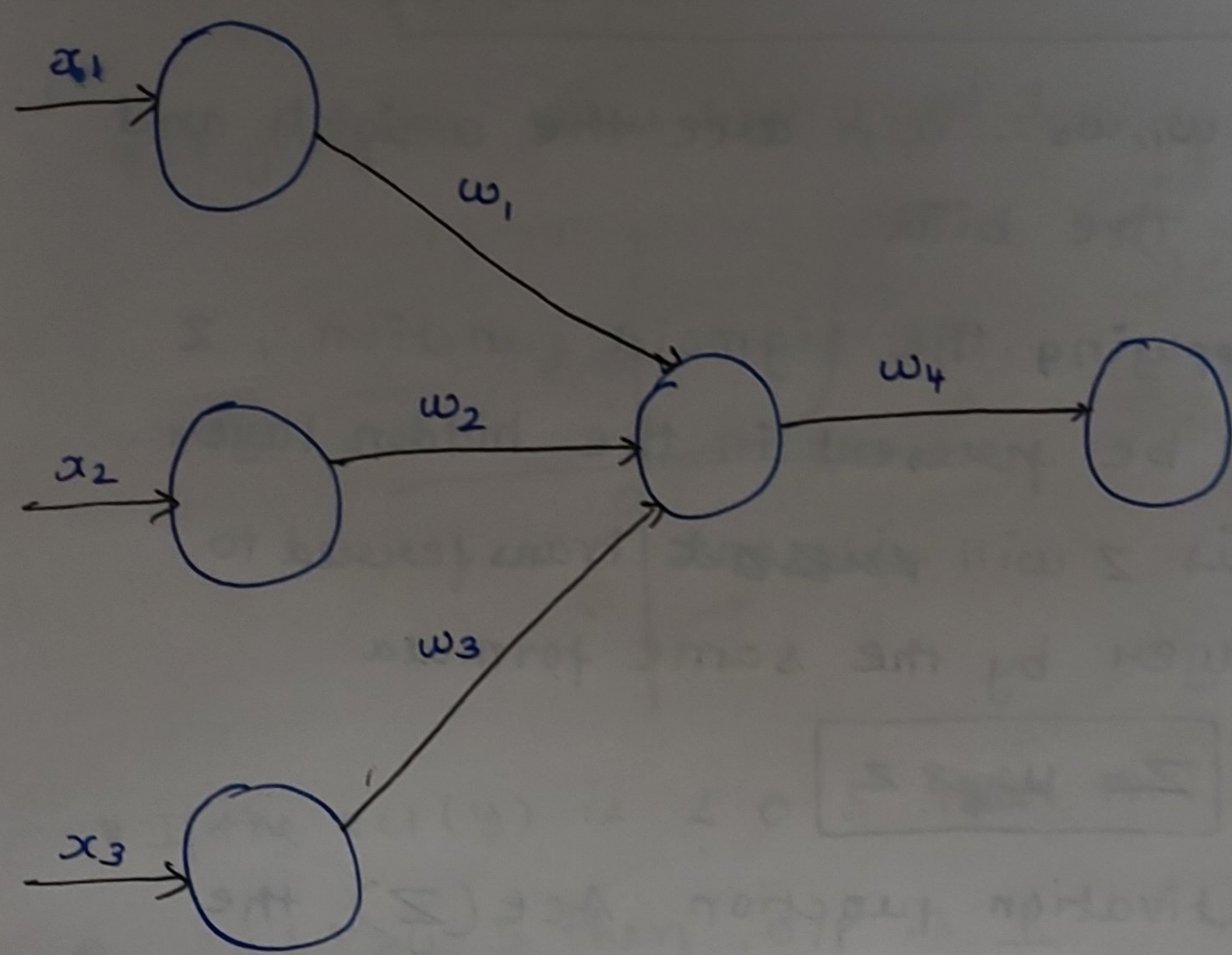
* Neural network consists of a large number of artificial neurons.



* In the Input layer the features will be given as the input to the neurons.
* The main idea behind this ANN is every information should be known by every nodes in the succeding layers, in order to find data patterns.
* ANN can have any number of hidden layers and nodes. the hidden layers performs all the calculations to find hidden features and patterns.
* The input goes through a series of transforma-tions using the hidden layer which finally results in output that is the output layer.

Scanned by TapScanner

working of neural networks:-



* x_1, x_2, x_3 are the features which are the input layer.

* Now the hidden layer want to get the information from ~~the~~ all the input neurons, It will be recieved by the formula

$$\boxed{y = \sum_{i=1}^{n} w_i x_i}$$

* and after calculating this the y will be passed into the sigmoid activation function, that is

$$\boxed{Z = Act(y) = \frac{1}{1+e^{-y}}}$$

Scanned by TapScanner

* according to the above architecture the y will be

$$\boxed{y = w_1x_1 + w_2x_2 + w_3x_3 + b}$$

where $w_1, w_2 \dots w_n$ are the weights and the b is the bias.

* after applying the sigmoid function, Z value will be present in the hidden layer.

now this Z will ~~present~~ transferred to the o/p layer by the same formula

$$\boxed{Z = w_4 \times Z}$$

then activation function Act(Z), the output from this will be the final.

* If the problem statement of above ANN is binary classification, then in the o/p if the value is > 0.5 the value will be 1 else it will be 0.

Activation function:-

* The activation functions adds the non-linear transformation to the i/p making it capable to learn and perform more complex task.

there are 2 types of Activation function

i) Linear

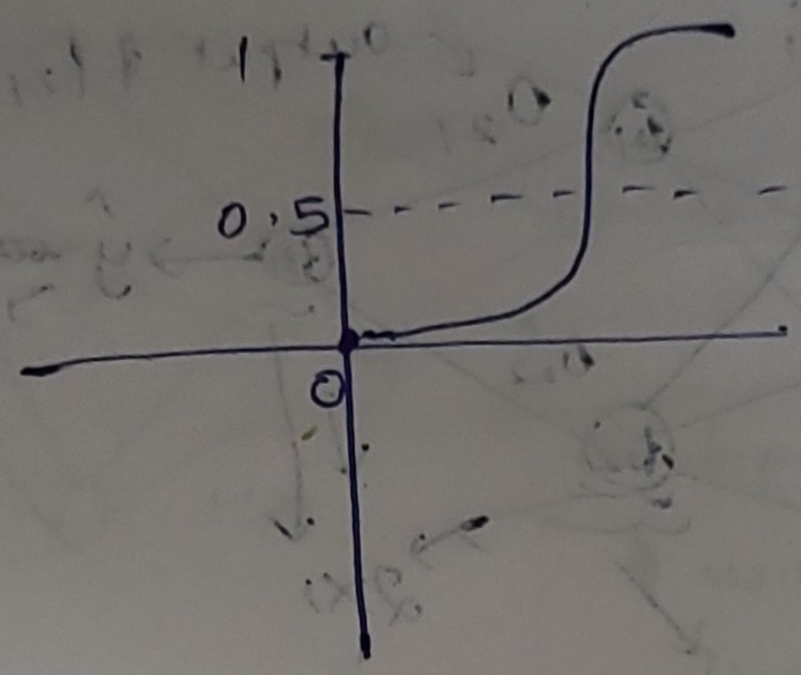
ii) non-linear.

Scanned by TapScanner

Sigmoid activation function:-

$$Act(y) = \frac{1}{1+e^{-y}} \qquad y = \sum_{i=1}^{n} w_i x_i + b_i$$

- It gives the value b/w 0 and 1



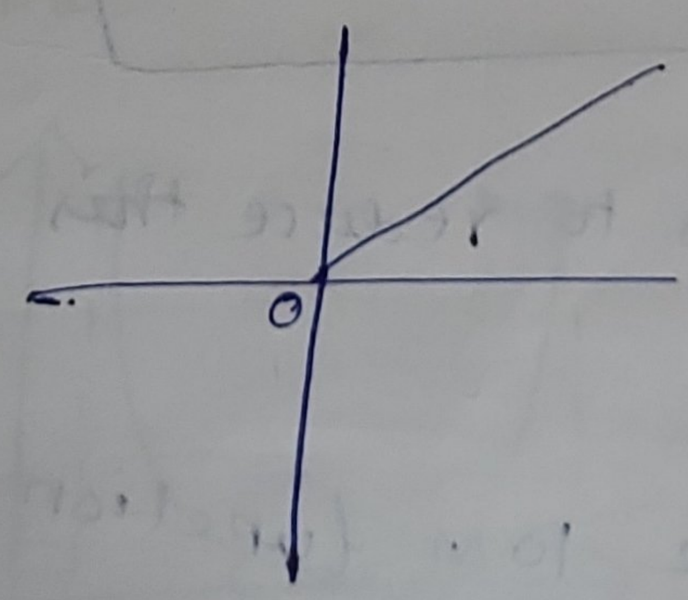
→ classification

- If the act(y) is < 0.5 then the value is 0, if > 0.5 then o/p is 1.

Rectified Linear unit (RELU) act function

- The formula is → Regression

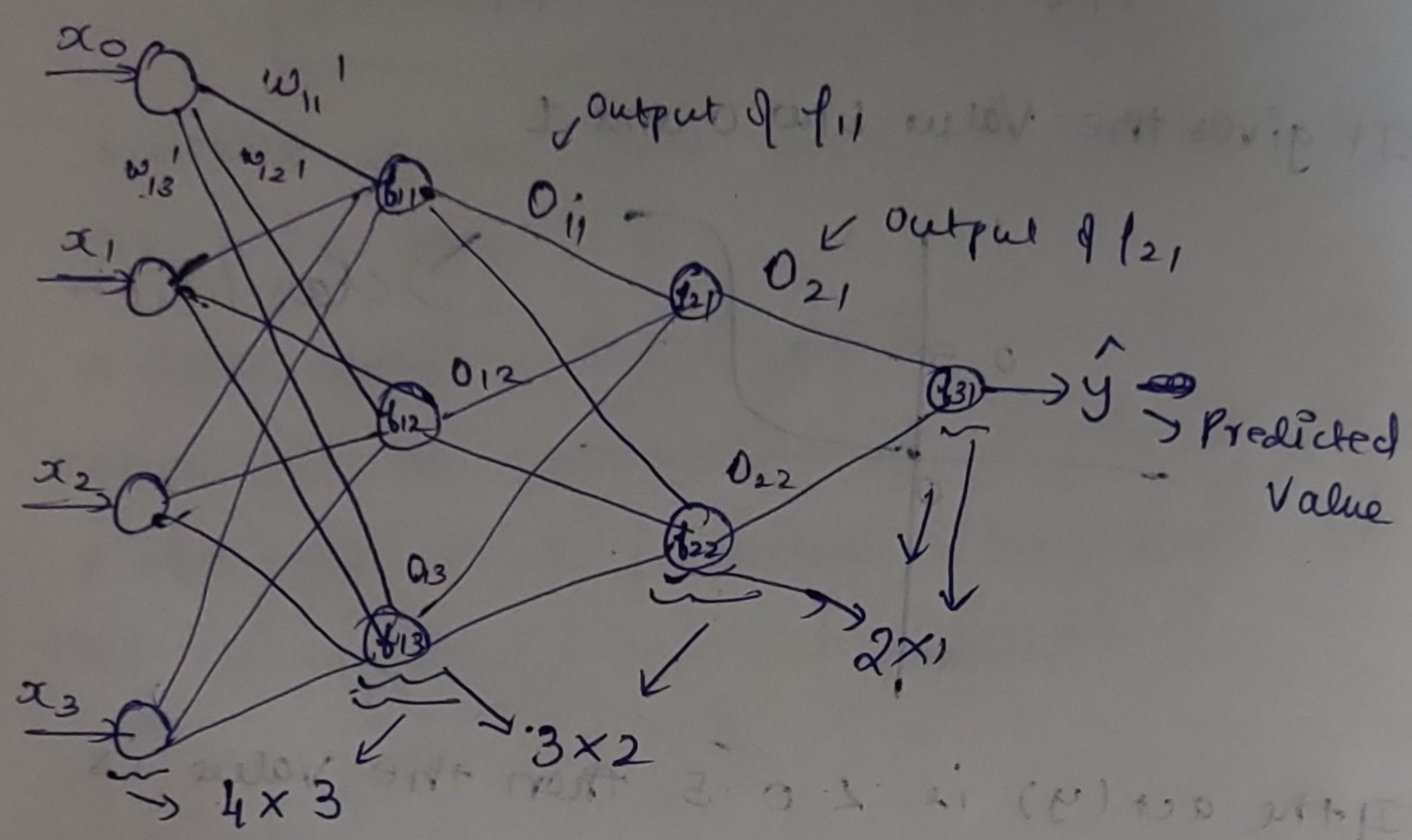
$$Act(y) = max(y, 0) \qquad y = \sum_{i=1}^{n} w_i x_i + b_i$$



In classification in middle hidden layers we can use relu but in o/p layer we should use sigmoid.

Scanned by TapScanner

Training a neural network:-



* we have y that is actual y in training data.

* Then we want to compare y and $\hat{y}$

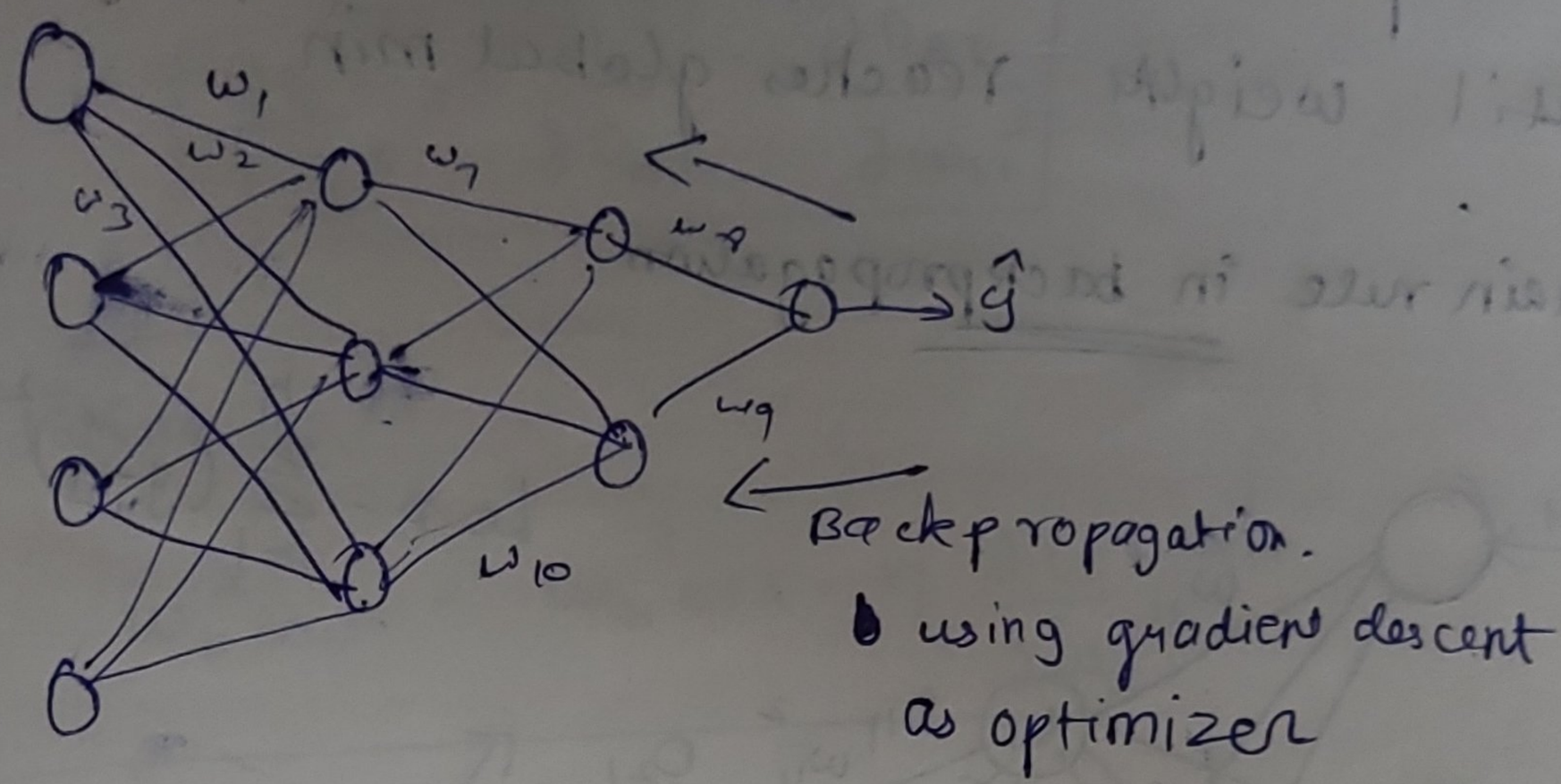
* so we have

$$\boxed{\text{Loss function} = (y - \hat{y})^2}$$

* now our aim is to reduce this loss function.

* So reduce the loss function we need optimizers such as gradient descent etc.

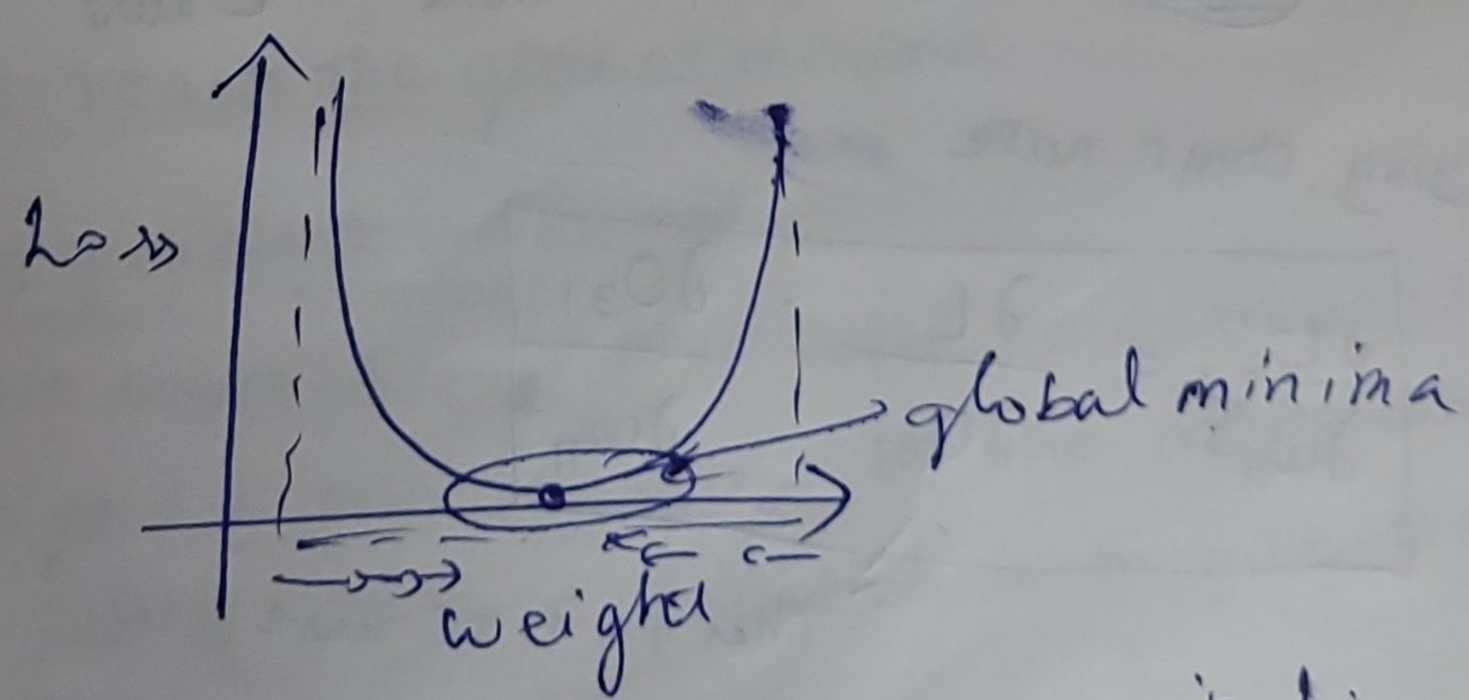
So in order to reduce the loss function we need to update the weights through backpropagation



Aim is to update the weights until the loss function is completely reduced.

$$\boxed{w_{new} = w_{old} - \underset{\text{learning rate}}{\eta} \frac{\partial(\text{Loss})}{\partial w}}$$

the main aim of grad desc is to reach global minima

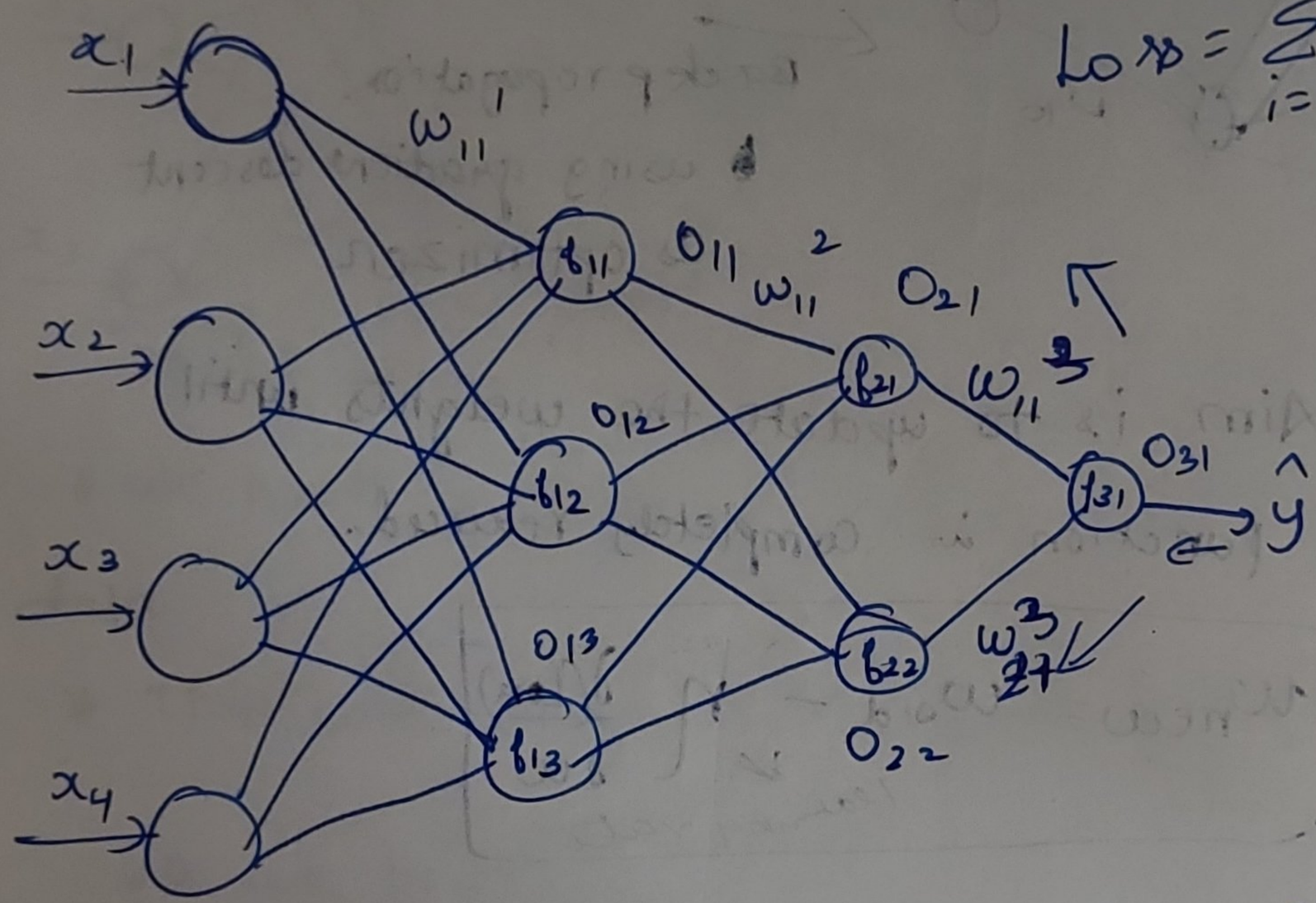


So as we seen the same in Linear regression the same thing no change

This propagation will be done many times in order to get reduced loss function and optimized weights until weight reaches global min.

chain rule in backpropagation

$$Loss = \sum_{i=1}^{n} (y-\hat{y})^2$$



updating w_{11}^3 through backpropagation using optimizers So that Loss function will reduce.

$$w_{11}^3 \text{ new} = w_{11 \text{ old}}^3 - \eta \frac{\partial L}{\partial w_{11}^3}$$

applying chain rule

$$\boxed{\frac{\partial L}{\partial w_{11}^3} = \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w_{11}^3}}$$

This is because w_{11}^3 affects O_{31}

Scanned by TapScanner

updating w_{21}^3

$$w_{21}^3 \text{ new} = w_{21}^3 \text{ old} - \eta \frac{\partial L}{\partial w_{21}^3}$$

$$\boxed{\frac{\partial L}{\partial w_{21}^3} = \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w_{21}^3}}$$

updating w_{11}^2

$$w_{11}^2 \text{ new} = w_{11}^2 \text{ old} - \eta \frac{\partial L}{\partial w_{11}^2}$$

$$\boxed{\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial w_{11}^2}}$$

∴ It is because O_{31} impacts O_{21} and O_{21} impacts w_{11}^2, So this is what chain rule is.

So this step will continue for all the weights and the process will be to and fro until the weights reach the global minima.

Vanishing gradient problem:-

* The sigmoid function will be activated in each and every neuron, so the main aim or the o/p of sigmoid function is to give the o/p b/w 0 to 1.

Scanned by TapScanner

* But during the ~~ch~~ backpropagation we will use gradient descent as the optimizer.

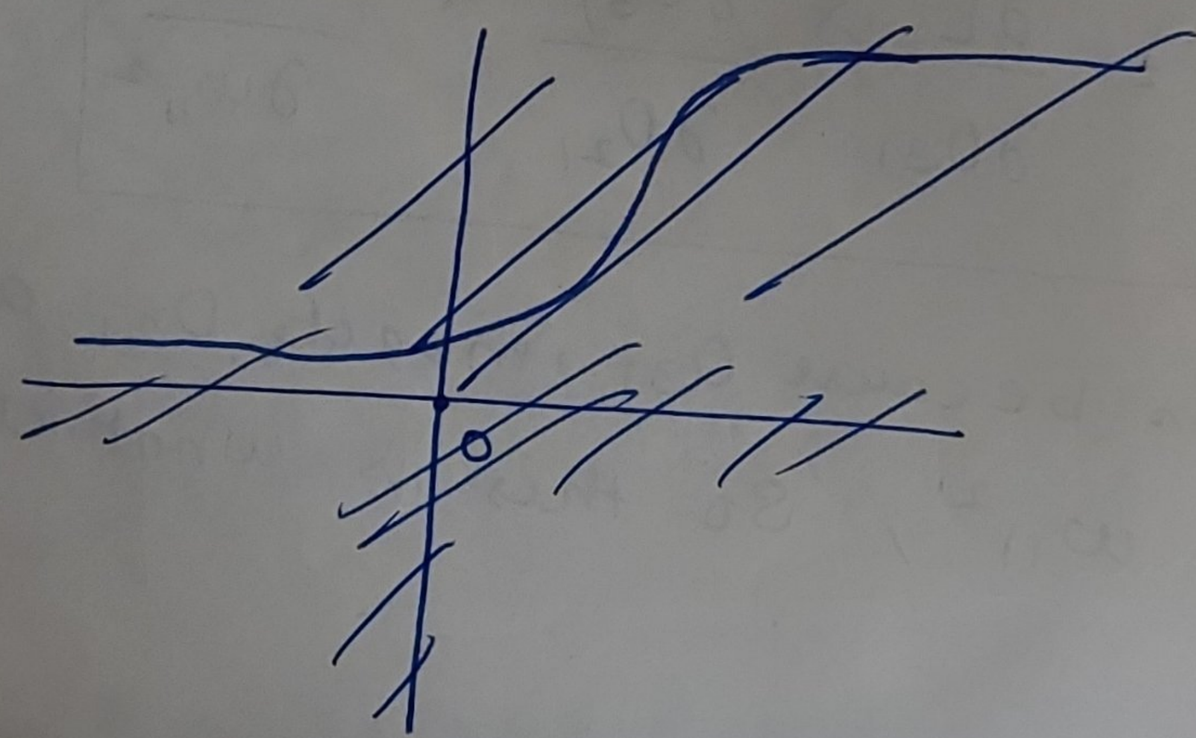
* ~~as applying~~ that is

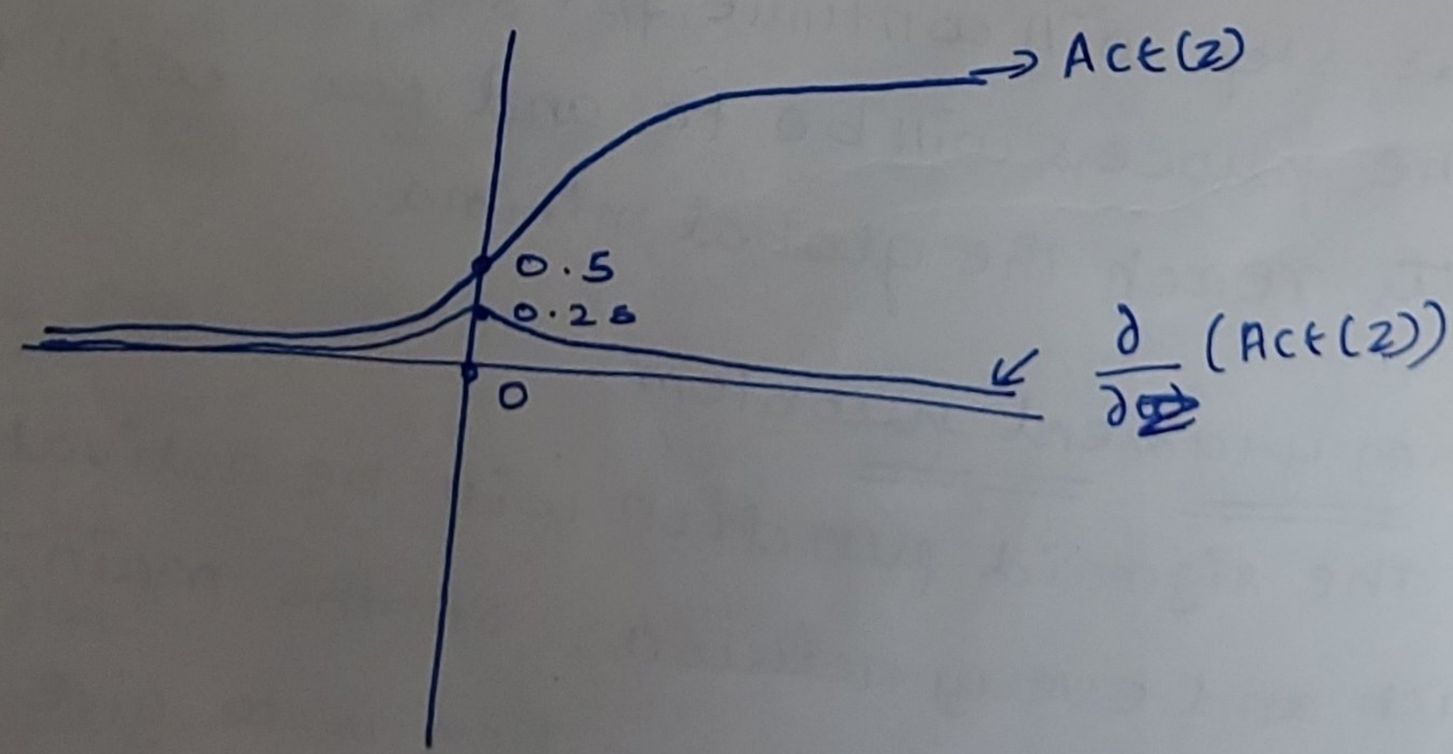
$$w_{11}'^{new} = w_{11}'^{old} - \eta \frac{\partial L}{\partial w_{11}'^{old}}$$

$$\frac{\partial L}{\partial w_{11}'} = \frac{\partial L}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{11}'}$$

* So while applying ~~d~~ grad desc to sigmoid, the derivative of sigmoid will be b/w 0 to 0.25

* So the graph would be





Scanned by TapScanner

~~188 In the chain rule~~

* So if the no. of layers increase ~~the~~, in the chain rule, the each derivative will decrease starting from 0.25 and approaches to 0.

* considering there is 4 layers

$$\frac{\partial L}{\partial w_{11}^{1}} = \frac{\partial L}{\partial O_{41}} \times \frac{\partial O_{41}}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{O_{21}}{\partial w_{11}}$$

$$= 0.20 \times 0.1 \times 0.05 \quad 0.04$$

$$= 10^{-4}$$

$$\boxed{\frac{\partial L}{\partial w_{11}^{1}} = 10^{-4}}$$

now updating the weight

$$w_{11}^{1}\,\text{new} = w_{11}^{1}\,\text{old} - \eta \frac{\partial L}{\partial w_{11}^{1}}$$

assuming $w_{11}^{1}\,\text{old} = 2.5$ & $\eta = 1$

$$w_{11}^{1}\,\text{new} = 2.5 - 1 \times 0.0004$$

$$= 2.5 - 0.0004$$

$$= 2.4996$$, which is approximately equal to ~~old~~ w_{old}

* So there is no meaning in doing this weight updation & backpropagation if the gradient start to vanish.

Scanned by TapScanner

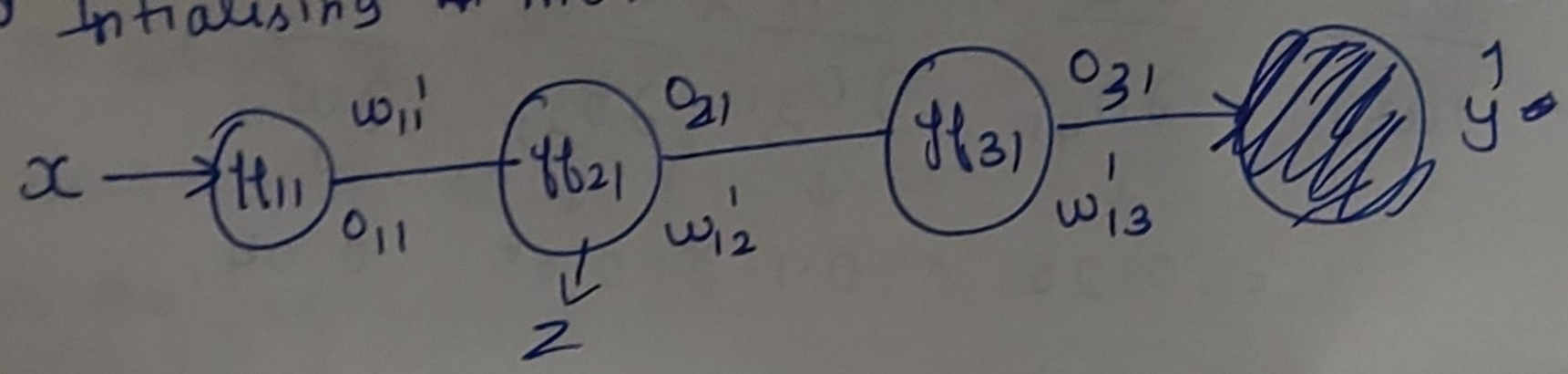
~~This problem~~

* This is the reason why sigmoid is not useful ~~when use~~ in the deep neural network if we use gradient descent as the optimizer.

* This problem is solved using RELU.

Exploding gradient problem:-

* Intialising more weighted weights cause exploding loss



$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial o_{31}} \times \frac{\partial o_{31}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}}$$

$$\boxed{w'_{11}\,new = w'_{11}\,old - \eta \frac{\partial L}{\partial w'_{11}}}$$

$$o_{21} = \phi(z)$$

$$z = w_{21} \cdot o_{11} + b_2$$

$$\frac{\partial o_{21}}{\partial o_{11}} = \frac{\partial \phi(z)}{\partial z} \times \frac{\partial z}{\partial o_{11}}$$

$$= 0 \leq \phi(z) \leq 0.25 * \frac{\partial (w_{21} \cdot o_{11} + b_2)}{\partial o_{11}}$$

Assuming 0.25

$$= 0.25 * w_{21}$$

Scanned by TapScanner

Assuming $w_{21} = 500$

$$\frac{\partial o_{21}}{\partial o_{11}} = 0.25 \times 500$$

$$= 125$$

So Lets take $\frac{\partial L}{\partial w_{111}}$

that is $\frac{\partial L}{\partial o_{31}} \times \frac{\partial o_{31}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{111}}$

$\downarrow$ 100 $\quad \downarrow$ 120 $\quad \downarrow$ 125 $\quad \downarrow$ 100

$= 150000000$ (very huge)

if we apply this in weight updation formula, considering $w_{old} = 250$

$$w_{11}\text{ new}' = 250 - 1 \times 150000000$$

$$= -149999926$$

So the w_{old}' & w_{new}' is right opposite & far away.

So if this is the case then it wont reach global minima.

So initializing higher weights causes this problem.

Properly initializing weights is very important.

Scanned by TapScanner

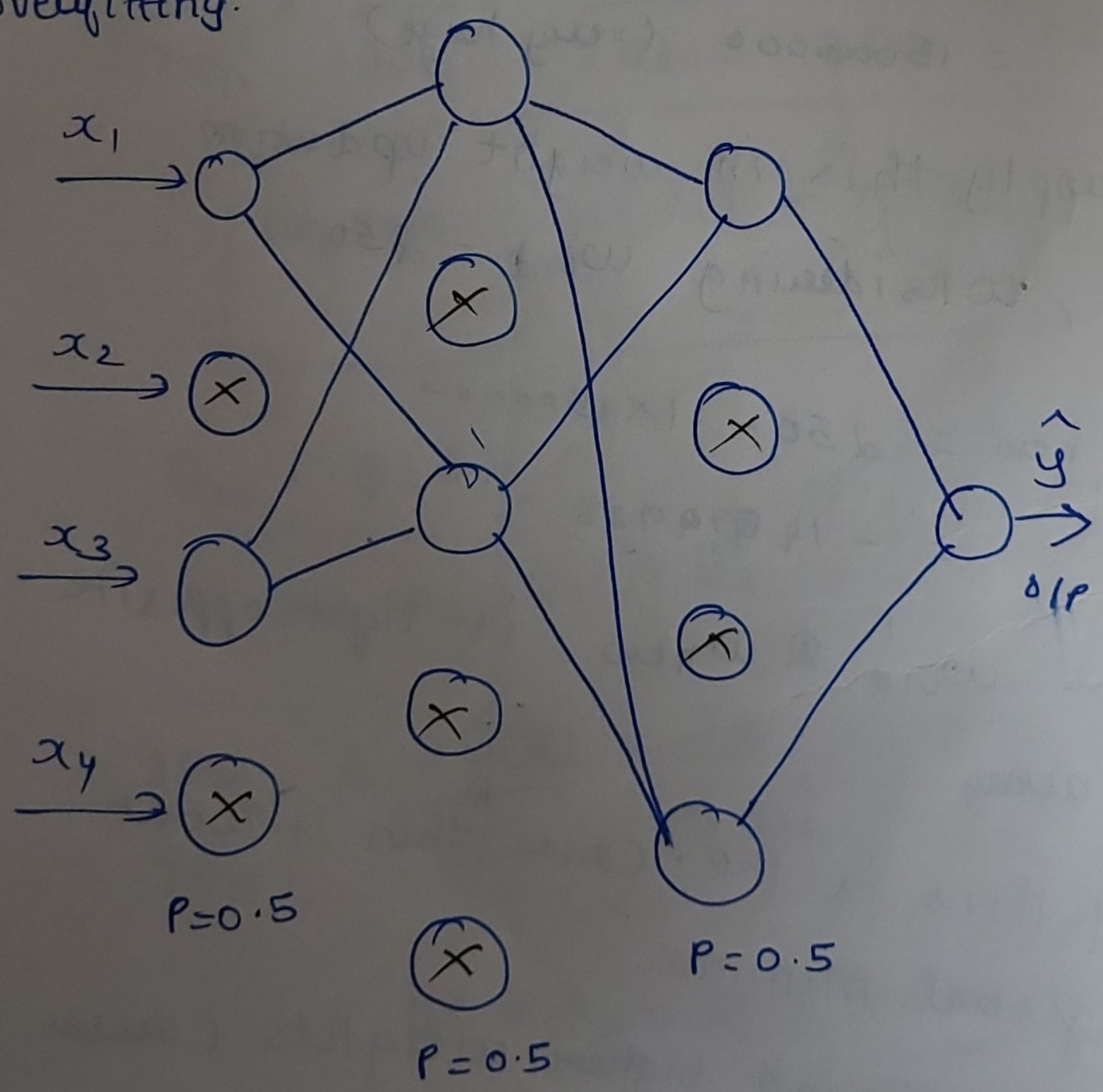
Drop out layers (Regularization)

* If the neural network is very deep then the model will get overfitted

* So reduce the overfitting we need to regularize the model, for that we will use drop out layers.

* The idea behind the dropout is deactivate some of the neurons and activate the remaining neurons for each epochs

* So this reduces the involvement of all the neurons in all epochs which reduces overfitting.



Scanned by TapScanner

* The neurons with ⊗ is the dropped out neuron for the epoch.

* The p is the dropout ratio, this value ranges from $0 \leq P \leq 1$.

* If the neural network is very deep then p value should be slight higher.

* This p value can be selected using hyper parameter optimization.

* This drop out mechnasms of selecting the set of neurons randomly is similar as of random forests randomly selecting the features in each model.

Activation functions:-

① Sigmoid activation function

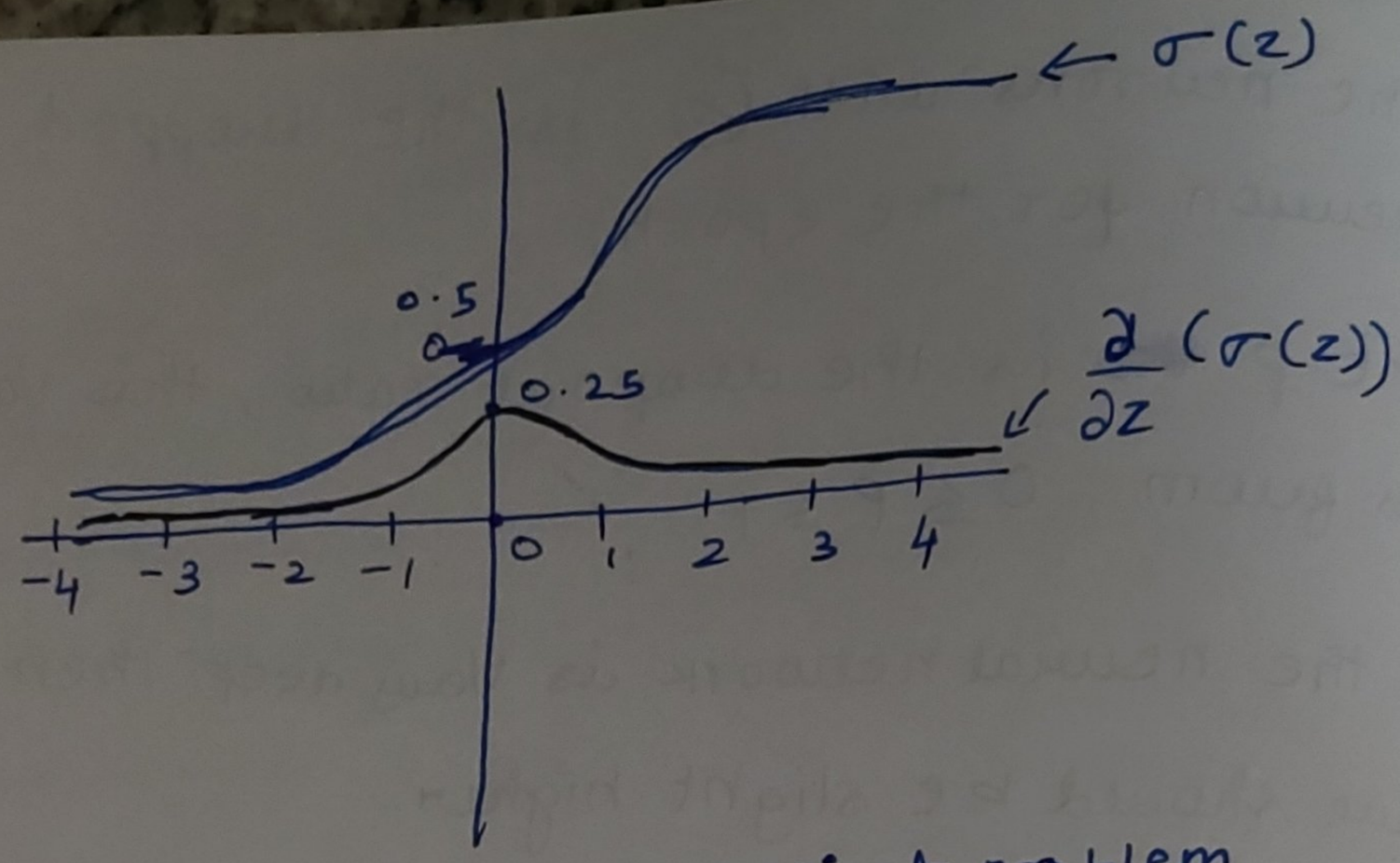
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

* This basically returns the value b/w 0 and 1.

* During backpropagation we will update weights using gradient descent, so in that time we calculate $\frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) \rightarrow$ gradient of $\sigma(z)$

* The gradient of $\sigma(z)$ lies between range of 0 to 0.25.

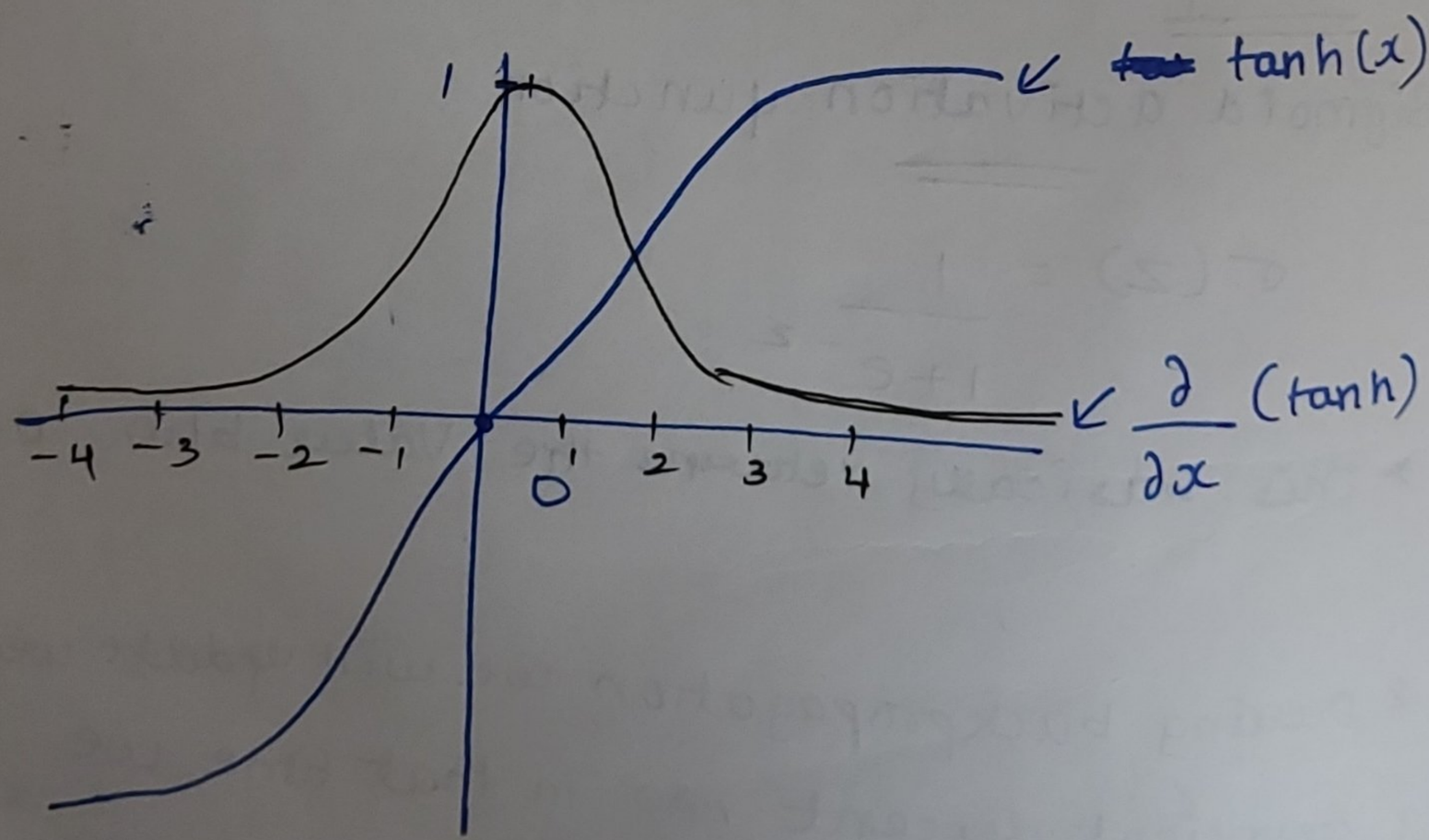
Scanned by TapScanner



* It has vanishing gradient problem.
* not zero centric, means σ(z) does not revolve around zero.

② Threshold activation (tanh)

$$\tanh(x) = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

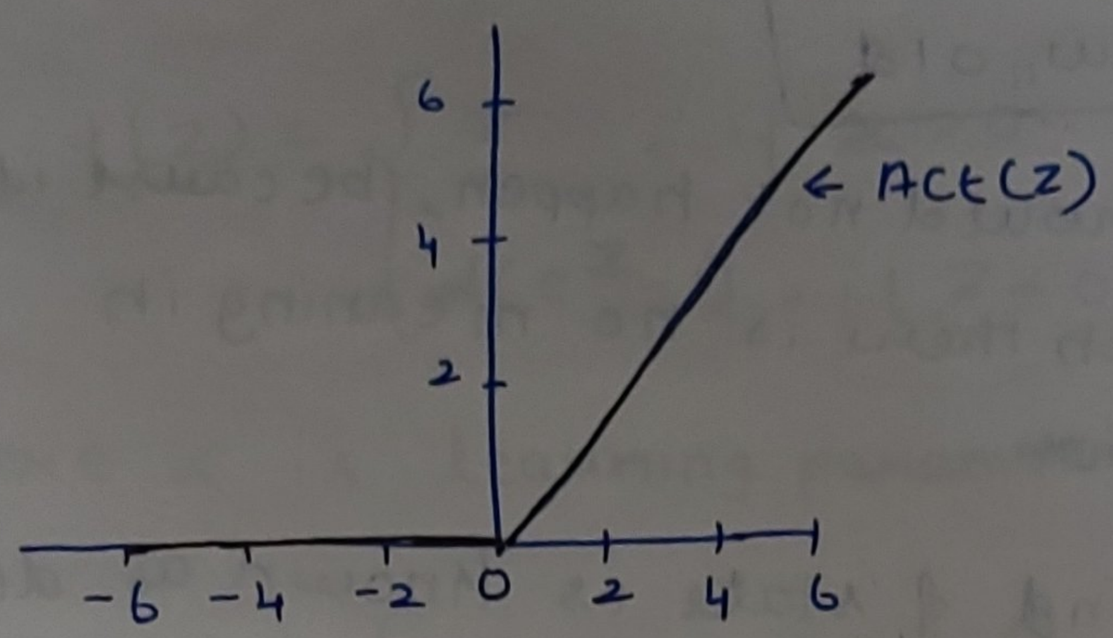


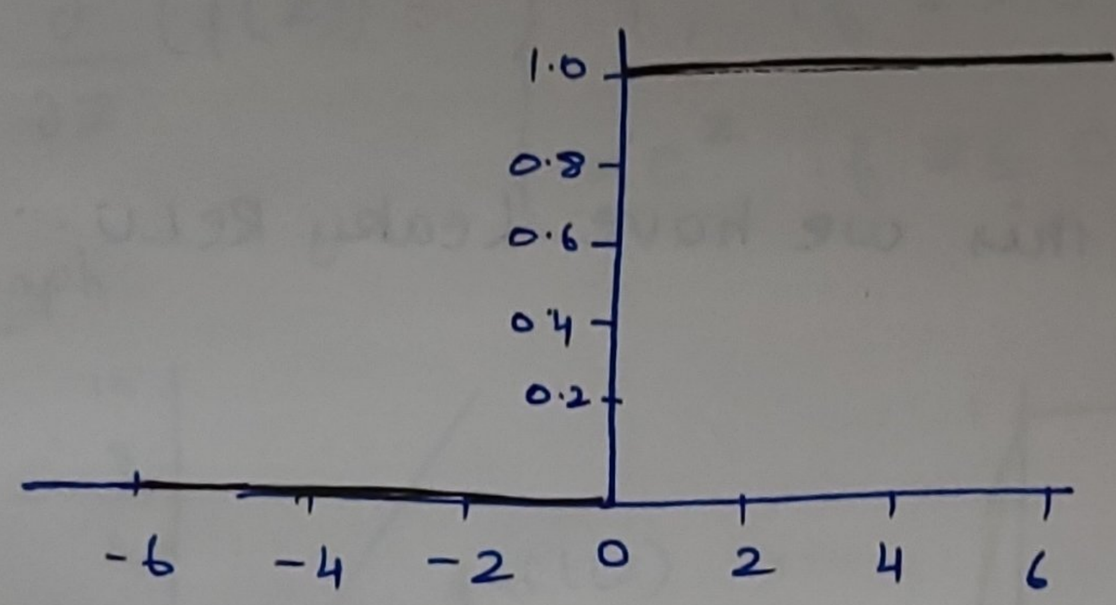
* High computational time
* gradient of tanh(x) ranges b/w 0 and 1
* Zero centric

Scanned by TapScanner

③ Rectified Linear unit (ReLU)

$$Act(z) = max(0, z)$$





← $\frac{\partial}{\partial z}(Act(z))$

(This is step function)

if z is >0 than, z will be the value

if z is <0 than, value will be zero.

$\frac{\partial(z)}{\partial} = 1$

Simply,

$$f(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases} ; \quad \frac{\partial(f(z))}{\partial z} = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$$

Considering chain rule occuring in backpropagation

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}}$$ (applying ReLU)

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\cancel{0.75}\,1 \times 0 \times 0$$

now whole $\frac{\partial L}{\partial w_{11}} = 0$

By applying this in formula

Scanned by TapScanner

$$w_{11\,new} = w_{11\,old} - \eta \frac{\partial L}{\partial w_{11}}$$

$$= w_{11\,old} - 0$$

$$\boxed{w_{11\,new} = w_{11\,old}}$$

∴ This should not happen, because if we got this, then there is no meaning in weight updation

∴ This kind of state is known as dead neuron.

* So to rectify this we have Leaky ReLU.

④ Leaky ReLU

$$act(z) = max(0.01z, z)$$

or

$$f(z) = \begin{cases} z, & \text{if } z > 0 \\ 0.01z, & \text{if } z \leq 0 \end{cases}$$

* The only change from ReLU is for $z \leq 0$ we have $0.01z$, which will avoid being zero, and save neurons from dead ReLU state.

* But due to this 0.01, if we multiple layers, so we need to mult $0.01 \times 0.01 \times 0.01$ many times

* So it will lead to Vanishing gradient problem.

* To rectify this we have ELU.

Scanned by TapScanner

⑤ Exponential Linear Unit (ELU)

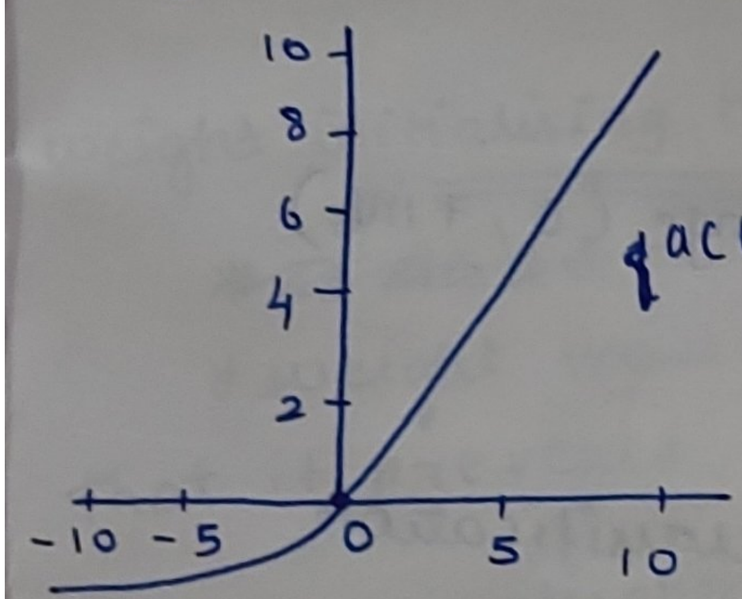
$$act(z) = \max(\alpha(e^{x}-1), x)$$

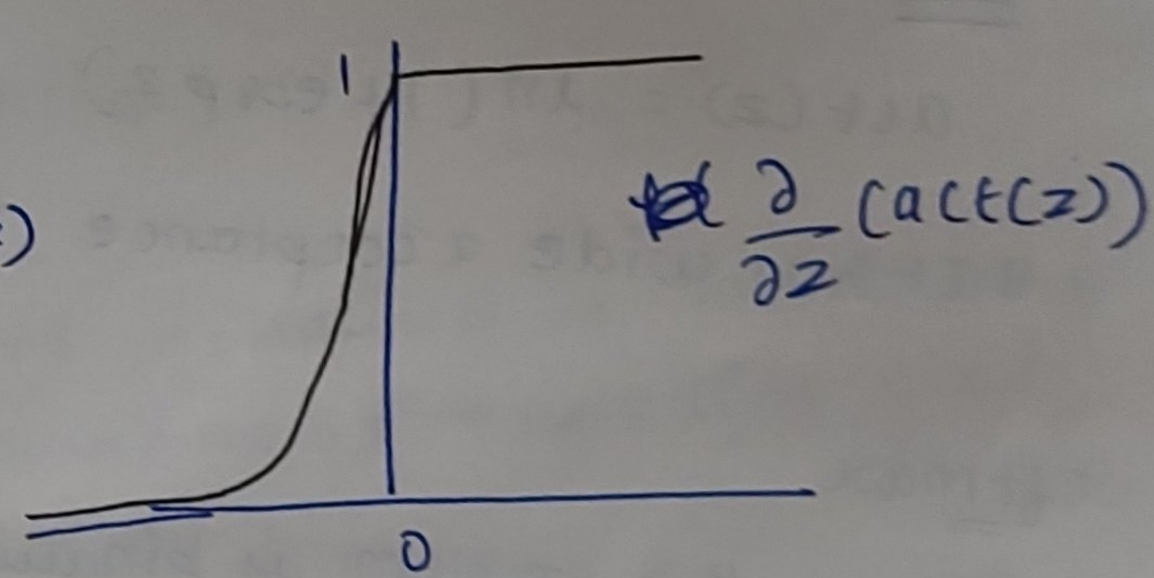
$$f(z) = \begin{cases} z, & \text{if } z > 0 \\ \alpha(e^{z}-1), & \text{if } z \leq 0 \end{cases}$$

where α is learning parameter (hyperparameter)

$$\frac{\partial}{\partial z}(f(z)) = \begin{cases} 1, & \text{if } z > 0 \\ \alpha e^{z} & \text{if } z \leq 0 \end{cases}$$

Graph:-





* Takes high computational time

⑥ PReLU (Parametric ReLU)

$$act(z) = \max(\alpha z, z)$$

if $\alpha = 0.01$

$$act(z) = \max(0.01z, z)$$

∴ It becomes Leaky Relu

if $\alpha = 0$

$$act(z) = \max(0, z)$$

∴ It becomes Relu

Scanned by TapScanner

~~α is~~

* The α value which is Learning hyperparame

useful to avoid negative z value.

* PRelu has a Variants of Relu.

⑦ Swish (Self gated function)

$$act(z) = z \times \sigma(z)$$ ← sigmoid

* This has to be used when there is more than 40 layers.

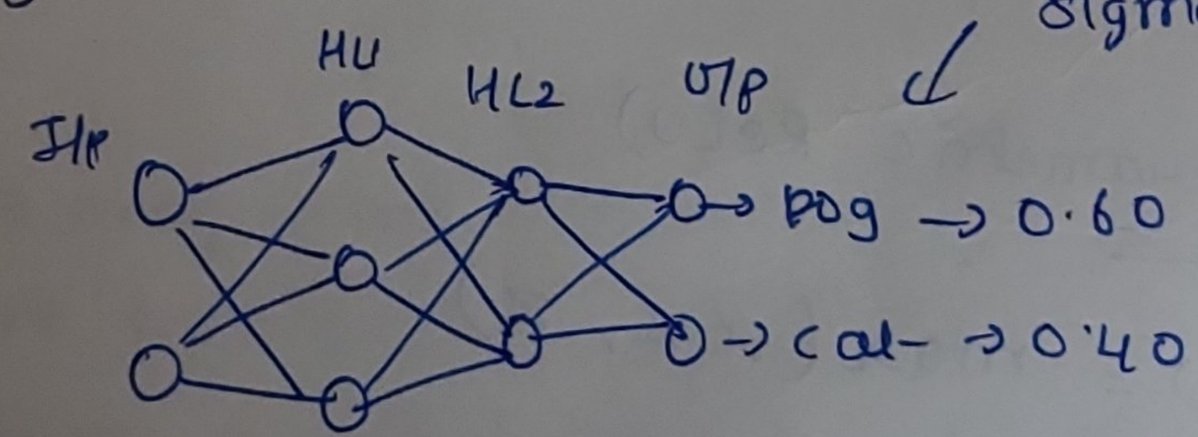
⑧ Softplus

$$act(z) = \ln(1 + \exp z)$$

* It has wide acceptance range (0, +inf)

⑨ Softmax

* Suppose the problem is binary classification b/w dog & cat.



Sigmoid because binary classification

} Probability

$\Rightarrow 0.60 + 0.40 = 1$

* If we have multiclass classification we cannot use sigmoid, because it classify b/w 0 & 1.

* So if we have multiclass classification, we will use softmax.

Scanned by TapScanner

* Suppose we have 4 o/p layers, means our problem has 4 different classes

after softmax

* the o/p will be [0.6, 0.2, 0.1, 0.1] = sum = 1

$$\text{Softmax} = S(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, 2 \ldots K$$

before softmax o/p will be = [40, 30, 10, 5]

after softmax

[0.61, 0.352, 0.117, 0.05]

weight initialising techniques:-

~~* It should~~

* weight should be initialised in such a way that it prevents vanishing or exploding grad.

i) Zero initialisation

* means initialising the weights as 0.

* If this is done, then we can never update the weights.

ii) Random initialisation

* imputing random weights

Best practices

iii) Xavier/Glorot init ~~to~~ uses uniform dist

* used when sigmoid act is used

Scanned by TapScanner

iv) He initialization (normal @ uniform)

& used when Relu is used

Stochastic Gradient descent:-

* There is a problem with gradient descent, that is during weight updation

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w} \quad ; \quad L = \sum_{i=1}^{n} (y_i - \hat{y})^2$$

which is we need to take all the data points in the dataset. The problem with this GD is if dataset has 1million records, it will take more computational time for each epoch, so to avoid this we have SGD & mini-batch SGD

* In SGD instead of taking summation in Loss function we will randomly take only 1 point for computation of each epoch, so the formula would be

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w} \quad ; \quad L = (y - \hat{y})^2$$

& advantage is less computational time, but disadvantage is it will take more time to converge to reach global minima.

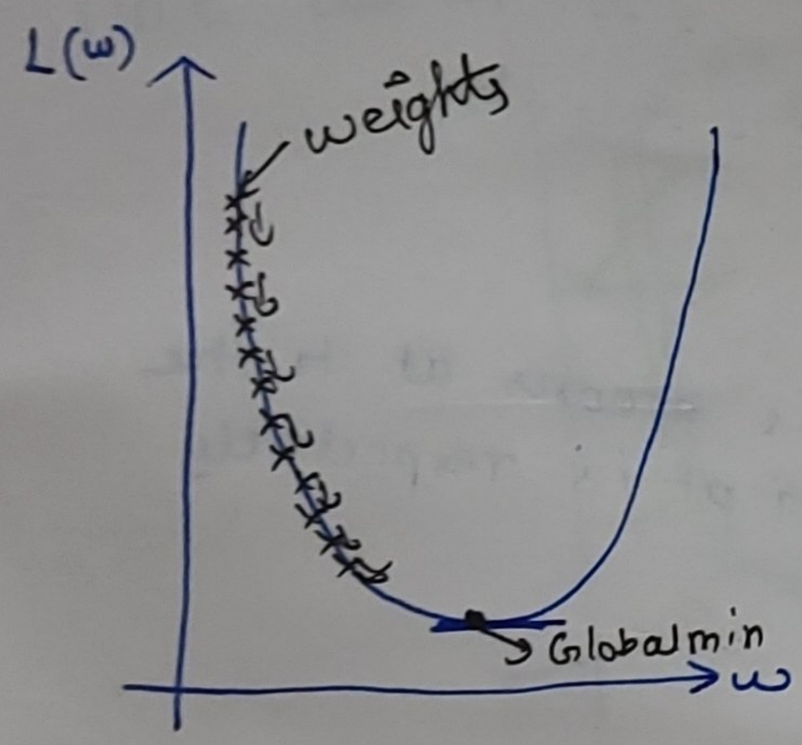
* Minibatch SGD has K data points in Loss functions where as K is always less than n, the formula would be

Scanned by TapScanner

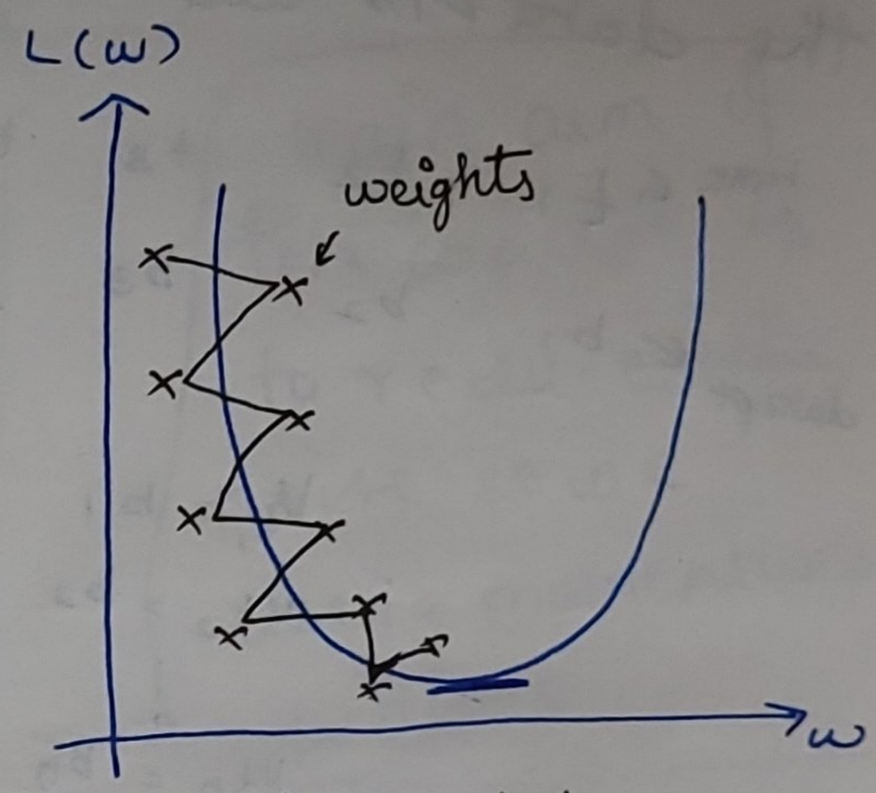
$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w} \; ; \quad L = \sum_{i=1}^{K} (y_i - \hat{y})^2$$

* So instead of taking n datapts that is whole data for computation, we will take as batch of data that is K data pts, which will have optimal computational time.

* The problem with mini-batch is also same, which is it takes more time to converge to global minima.



Gradient descent



mini-batch SGD

Representation

* In mini-batch since we take sample batch we have weights out of the range of usual GD, so to fix this and to reduce the convergence time we have the

technique called exponential moving average. which will smoothen the spread out weights into a smooth line as the linear regression does,

This concept is called the SGD with momentum.

considering in the minibatch diagram, the data pts are in diff time intervals.

time ←	t_1	t_2	t_3	t_4	t_5	t_n
data pt ←	b_1	b_2	b_3	b_4	b_5	b_n

$$\left.\begin{aligned} V_{t_1} &= b_1 \\ V_{t_2} &= b_2 \\ &\vdots \\ V_{t_n} &= b_n \end{aligned}\right\}$$ This means at t_1 the data pt is respectively b_1

now, $V_{t_1} = b_1$

$$V_{t_2} = \gamma \times V_1 + b_2 \quad ; \quad 0 \le \gamma \le 1$$

$V_{t_2} = V_2$

$$= 0.5 \times b_1 + b_2$$

b_2 → more imp

$0.5 \times b_1$ → less importance

$$V_{t_3} = \gamma \times V_2 + b_3$$

$$= \gamma(\gamma \times V_1 + b_2) + b_3$$

$$= \gamma^2 V_1 + \gamma b_2 + b_3$$

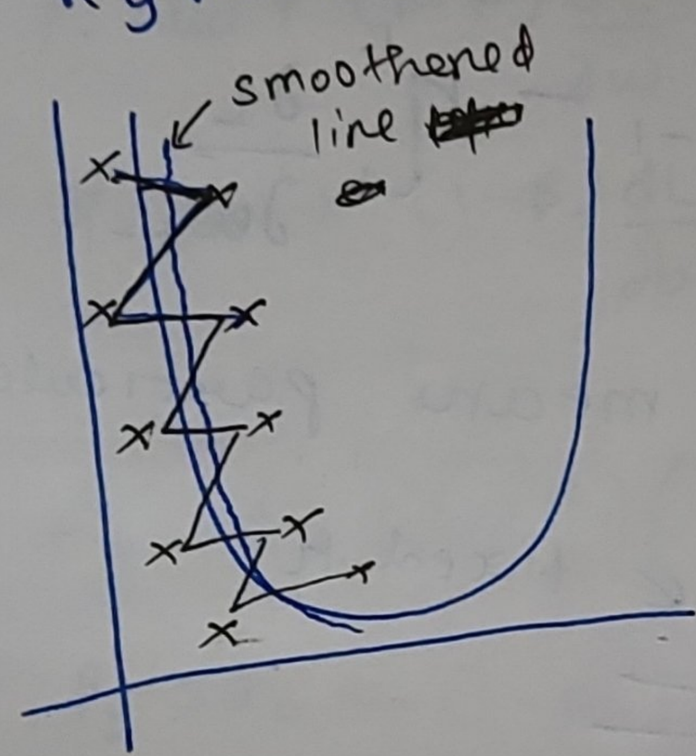
$$= \gamma^2 b_1 + \gamma b_2 + b_3$$

Scanned by TapScanner

$V_3 = 0.25b_1 + 0.5b_2 + b_3$ → more imp

So this how moving avg is calculated between V_1, V_2, V_3.

* If we do this ~~it will~~ for all pts it will remove the noise, means the zig zag movement and produce the smoothened line as regression does



main aim of this concept is to reduce the noise so we using momentum concept

* Final formulas ~~are~~ were if we apply this concept

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

$$= w_{old} - \left[\nu V_{t-1} + \eta \frac{\partial L}{\partial w_{old}} \right]$$

where

$$V_{t-1} = \nu \left[\frac{\partial L}{\partial w_{old}}\right]_t + \nu \left[\frac{\partial L}{\partial w_{old}}\right]_{t-1} + \nu^2 \left[\frac{\partial L}{\partial w_{old}}\right]_{t-2}$$

$$+ \cdots \nu^n \left[\frac{\partial L}{\partial w_{old}}\right]_{t-n}$$

Scanned by TapScanner

Adagrad optimizer:-

~~* In all other optimizers there is η (learning~~

* The optimizers like GD, SGD etc — have η which has the fixed value.

* But in Adagrad the η value will change for each epochs.

* The formula would be

$$w_t = w_{t-1} - \eta'_t \frac{\partial L}{\partial w_{t-1}}$$

where as t means particular time

$$\eta'_t = \frac{\eta}{\sqrt{\alpha_t + \varepsilon}}$$

η ← fixed n

ε → small Positive number, to add if α_t is 0

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial h}{\partial w_i} \right)^2$$

* The η'_t value decreases α_t increase

* The problem here is if the η'_t is keep on reducing for each epoch, convergence will be slow

* If α_t is very high then η'_t can be very less making the term zero and $w_{new} \cong w_{old}$.

Scanned by TapScanner

Adam optimizer :-

* Actually there is no disadv in adadelta RMSProp optimizer, but Adam optimizer is derived with the advantages of SGD momentum (smoothening), & RMSProp (change in η)

* Basically it is the combination of RMSProp & SGD momentum.

$$\left.\begin{aligned} V_{dw} &= \beta_1 V_{dw} + (1-\beta)\frac{dL}{\partial w} \\ V_{db} &= \beta_1 V_{dw} + (1-\beta)\frac{dL}{\partial b} \end{aligned}\right\} \text{momentum (smoothening)}$$

$$\left.\begin{aligned} S_{dw} &= \beta_2 S_{dw} + (1-\beta)\left(\frac{\partial L}{\partial w}\right)^2 \\ S_{db} &= \beta_2 S_{db} + (1-\beta)\left(\frac{\partial L}{\partial w}\right)^2 \end{aligned}\right\} \text{RMSProp}$$

* So final weight updation formula is

$$w_t = w_{t-1} - \frac{\eta * V_{dw}}{\sqrt{S_{dw} + \epsilon}}$$

(η → initial LR.)

$$b_t = b_{t-1} - \frac{\eta * V_{db}}{\sqrt{S_{db} + \epsilon}}$$

Scanned by TapScanner

Convolutional neural network :- (CNN)

* The main objective of CNN is to do the image processing, image classification.

* The process of converting a grayscale (Black & white) or the RGB image into simpler image with minimum scale is called the convolution.

* This process will take place again and again inorder to process the image as step by step.

* The convolutional layers will detect different parts of image such as vertical edge, horizontal edge, using different filters.

Lets consider a 6x6 image (Black & white image)

Grayscale = 0 - 255

0 ⇒ dark

255 ⇒ light

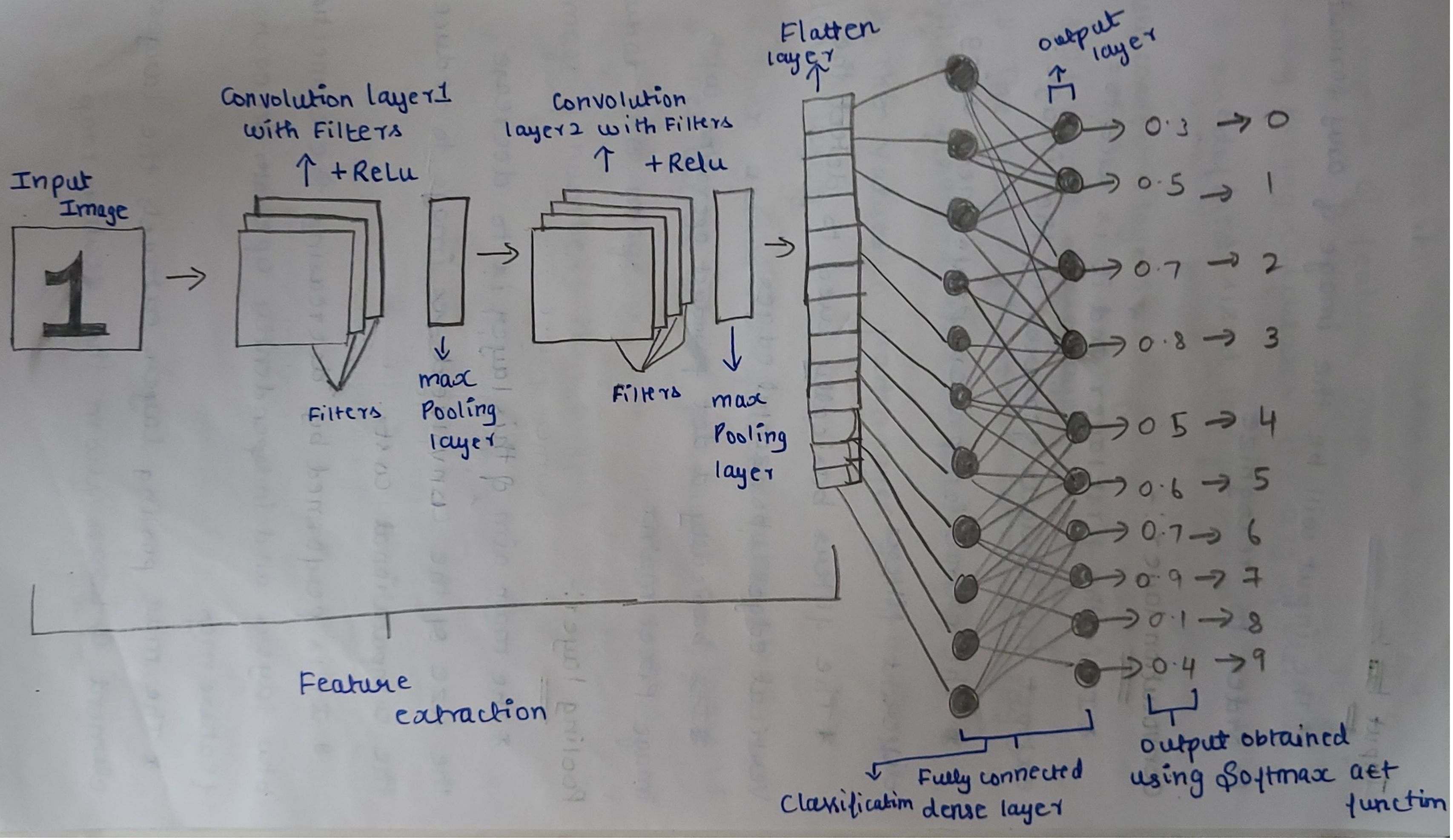
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

6x6

* Now the task is to process this 6x6 image.

* For this we have to do convolution process using filter.

Scanned by TapScanner



Scanned by TapScanner

Input :-

* The input will be the image of any format in order to recognize.

Convolution layer :-

* It is the first layer and it is used to extract the various features of the image.

* The extraction process performed using different filters.

* The filters basically used to detect the vertical edges, horizontal edges

* Its basically a dot product operation with image pixel matrix.

Pooling layer :-

* The main aim of this layer is to decrease the size of the convolved ~~fea~~ image to reduce the computational costs.

* It is performed by decreasing the connection btw layers and independently operates on each feature map.

* The max pooling layer extracts the largest element ~~is taken~~ from the feature map.

Scanned by TapScanner

it literally means, it extracts the maximum information of feature map.

- Avg pooling calculates the avg of elements in a predefined sized image section.

Fully connected layer:-

- It is the dense layer
- The output from the convolution layer will be flattened in the flatten layer.
- Then values in flatten layer given to the dense layer.
- Then actual ANN process takes place.
- Then output will be produced as classified what the image is, Like, if the input is image of number, then output will be number.

Scanned by TapScanner

Recurrent Neural network:- (RNN)

* RNN is a type of ANN which uses sequential data or time series data.

* This RNN is used in natural language processing, speech recognition etc...

* In RNN, the output from previous step are fed as input to the current step.

* In NLP, to predict the next word of the sentence requires the memory of previous word.

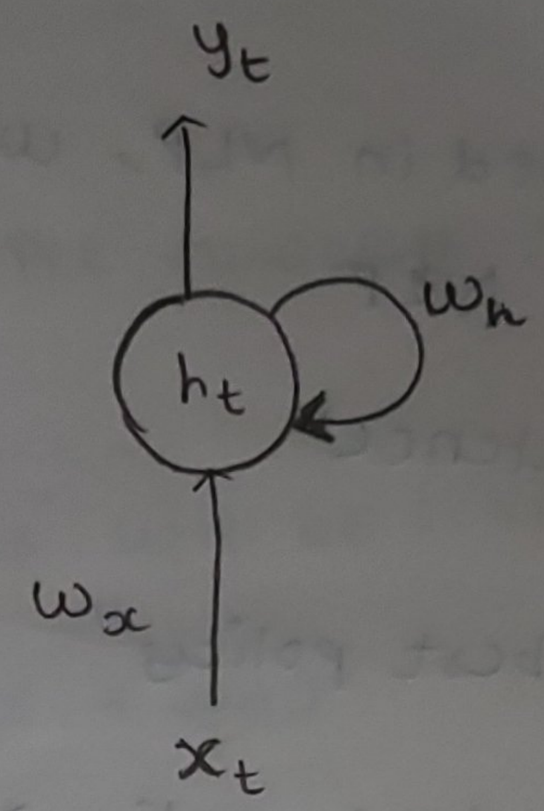
* So the hidden state will have a memory, which remembers all information about what has been calculated.

* In a general neural network, an i/p is fed to an i/p layer and is further processed through no. of hidden layers and a final o/p is produced, with an assumption that 2 successive i/ps are independent of each other or i/p at time step t has no relation with i/p at time step t-1.

Scanned by TapScanner

RNN Architecture:-

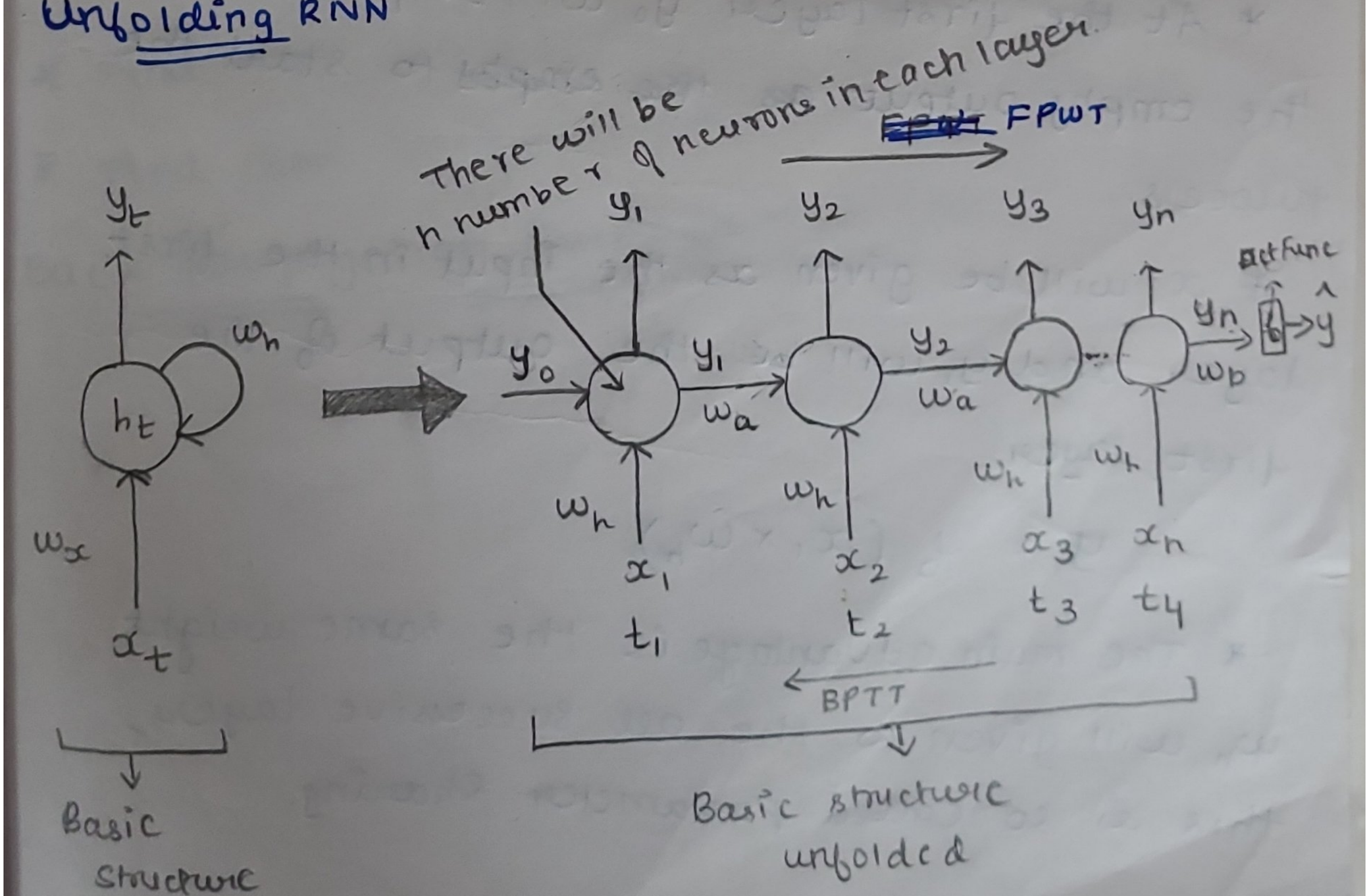
Basic structure:-



* This basic layer will get repeated as the time increases.
* The same weight w_x will be shared for all the repeated layers.

$x_t \Rightarrow$ input at time t ; $y_t \Rightarrow$ output at time t and the output will be given to same network with weight w_h.

Unfolding RNN



Scanned by TapScanner

Forward propogation with time :- (FPWT)

* As per the architecture, the FPWT takes place.

* As the RNN is used in NLP, we can take the examples from NLP.

* Lets take a sentence

"Honesty is best policy"

$x = \langle$ Honesty is best policy $\rangle$

* Then each word in the sentence will be converted to the vectors with any $\mathbb{R}^d$

$$x = \langle x_1, x_2, x_3, x_4 \rangle$$

$$\Downarrow$$

Vectors

* At the first layer y_0 will be ~~passed~~ given as the empty output ~~as the empty~~ to start the process.

* x_1 will be given as the input in the first layer and y_1 will be the output of the first layer.

$$y_1 = f(x_1 \times w_h)$$

* The main advantage is the same weight w_h will given to the all successive layers, this is so called parameter sharing.

Scanned by TapScanner

* then ~~upcom~~ this y, will be given to the same layer but this will occur at time t_2.

* Like wise the process repeats at $t_1 \ldots$ t_n.

* The outputs will be like

$$y_1 = f(x_1 \times w_h)$$

$$y_2 = f(x_2 w_h + y_1 w_a)$$

$$y_3 = f(x_3 w_h + y_2 w_a)$$

$$y_4 = f(x_4 w_h + y_3 w_a)$$

* For the output calculation another weight w_b is used.

* And then loss function will be calculated

* And then back propogation starts to proceed.

Back propogation through time (BPTT):-

Scanned by TapScanner