Deep Learning

what is Deep Learning

→ It is process of teaching machine, how human brain works. (mimicing the human brain)

→ This is achieved by "Multi layer NN"

Three main neural networks

① ANN – Artificial Neural network

→ with this NN we can solve Clissification and Regression problems

② CNN – Convolution Neural Network

→ we use it to solve image and video frame type data input

→ Advanced CNN

eg: RCNN, Masked RCNN, Detectron, yoloV3, V8

③ RNN – Recurent Neural Network

→ Solve Text data, Time series Data

Advanced RNN

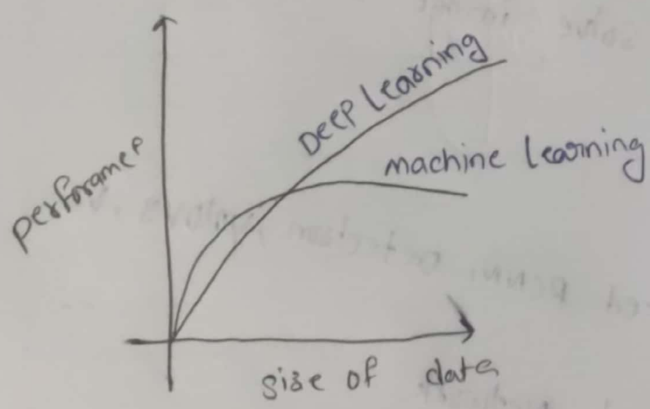
LSTM, RNN GRU, Bidirectional LSTM, RNN, Encoder, decoder, transforms, BERT, Attention model

Note :- To solve above ⓐ Solution we use Mostly TensorFlow, PyTorch libraies.

② Why Deep Learning ~~becoing~~ becoming popular

In 2005 major social media platforms like Faceb , orkut was facing major issue with storing data like "Image, Text, Document". So in 2011-2016 companies like Cloudera, Home work come up with solution of Hadoop to store unstruced and struced data. In 2015 companies start thinking to utilize stored data to improve theries products like more recomendation, friends recomendation.

Graph of Data vs performace in ML vs DL



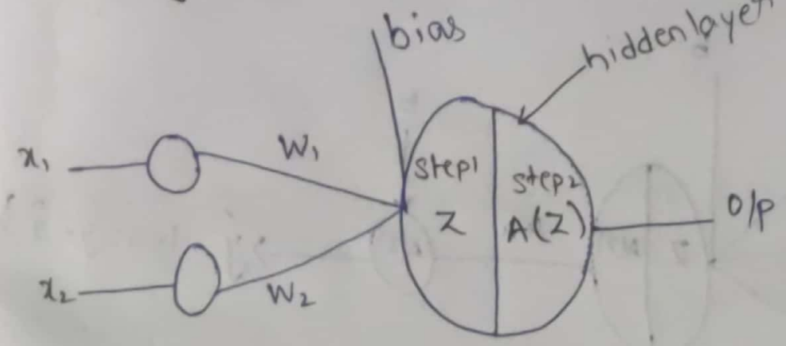
Domains
(i) Medical → xrays, cancer detection
(ii) Ecomerce
(iii) logestics

③ perceptron [ANN]

parts of of NN

(i) I/p layer
(ii) Hidden layer
(iii) weights
(iv) Activation function
(v) Bias (weights are zero neuron not active so we use bias)
(vi) o/p layear

Single layear NN



Feed Forward NN (Forward propogation)

Backword proopogation

Step-1

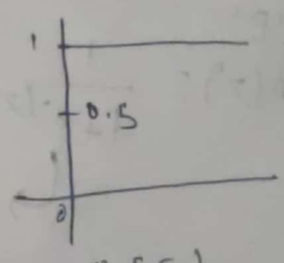
$Z = x_1 w_1 + x_2 w_2 + b$

$Z = \sum_{i=1}^{N} x_i w_i + b \approx \boxed{y = mx + c}$

→ If NN predict Correct, o/p keep the Same weights
→ If NN predict wrong o/p update the weights in back propagation

Step2
Activation Function

eg Step Function

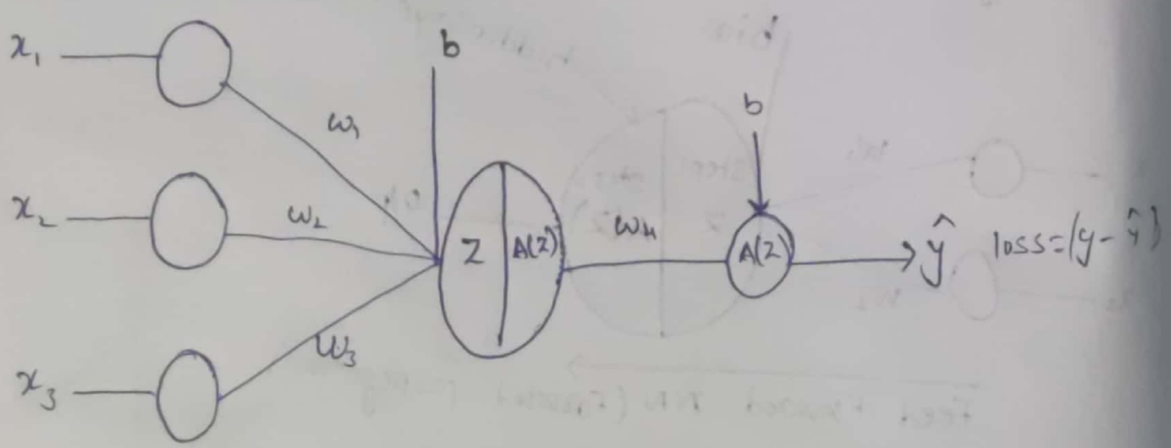


$> 0.5 = 1$
$< 0.5 = 0$

2-layered neural network

Let consider sample data

x_1 IQ	x_2 Study hrs	x_3 play hrs	y o/p
95	4	4	1
100	5	2	1
95	2	7	0



Step-1

Intilize ramdom weight i.e w_1 w_2 w_3 = 0.01 0.02 0.03

$Z = 95 \times 0.01 + 4 \times 0.02 + 4 \times 0.03 + 0.01$

$= 1.151$

Step-2

$A(z) = \frac{1}{1+e^{-(z)}} = \frac{1}{1+e^{-1.151}} = 0.758$

↳ Sigmoid Activation fu^n

HL 2:-

$z = 0.759 \times 0.02 + 1 \times 0.03$

$= 0.04518$

Step 2

$O_2 = \frac{1}{1+e^{-0.04511}} = 0.51129$

Backward propogation and weight updation

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

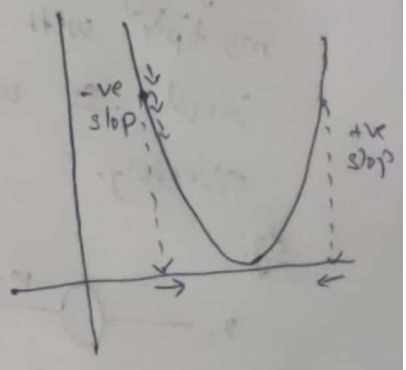
$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b_{old}}$

Optimizers :- To reduse the Loss values

$w_{new} = w_{old} - \eta(-ve)$ $\qquad w_{new} = w_{old} - \eta(+ve)$

$w_{new} = w_{old} + (+ve)$ $\qquad = w_{old} - (-ve)$

$w_{new} >> w_{old}$ $\qquad w_{new} << w_{old}$



Chain rule of derivation

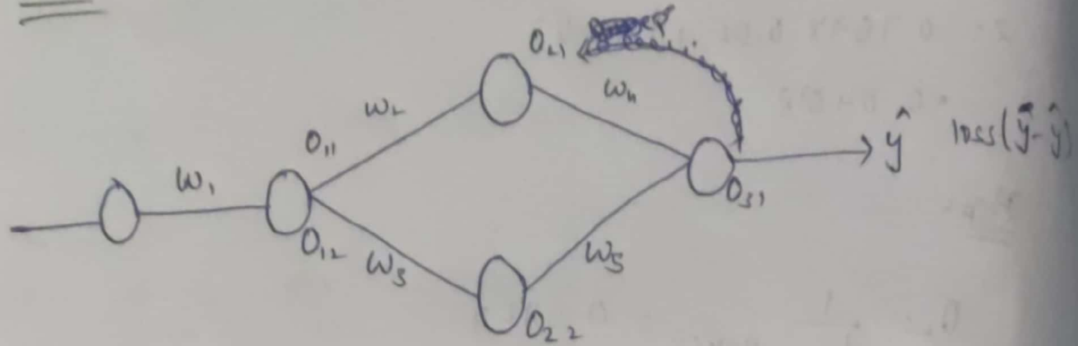
$w_{new} = w_{old} - \eta \boxed{\frac{\partial L}{\partial w_{old}}} \rightarrow$ Slope

$w_4 = \frac{\partial L}{\partial w_{old}} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial w_{4old}}$ $\qquad \boxed{O_2 = A(z)}$

$w_1 = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$

$\rightarrow \frac{\partial L}{\partial w_{1old}} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_{1old}}$

Let's take



To update w_1 weight, we have two paths

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

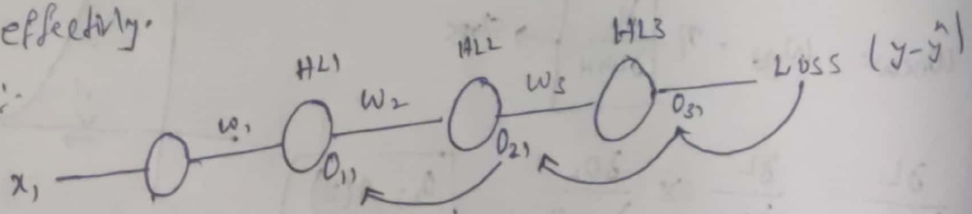
$$+$$

$$\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial O_{12}} \times \frac{\partial O_{21}}{\partial w_{1old}}$$

Vanishing Gradient problem and Activation function

→ In Deep NN sigmoid activation function will not work. because derivative of sigmoid activation function always gives b/w (0 - 0.25) when this value multiply with learning rate(η) value become very small. so weight updation will not happen very effectivly.

Eg:-



$$\frac{\partial L}{\partial W_{1old}} = \frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial W_{11}}$$

Let's take

$$\boxed{z = w_3 \times O_{12} + b_3}$$

$$\frac{\partial O_{31}}{\partial O_{21}} = \frac{\partial \sigma(z)}{\partial O_{21}} = \frac{\partial \sigma(z)}{\partial z} \times \frac{\partial z}{\partial O_{21}} \quad \frac{\partial (w_3 \times O_{21} + \overset{\text{Const}}{b_3})}{\partial O_{21}}$$

$$= (0 - 0.25) \times w_3$$

$$\frac{\partial O_{31}}{\partial O_{21}} = [0.25] \times w_3 \Rightarrow \text{Small value}$$

$$\because W_{new} = W_{old} - \underbrace{\eta \underbrace{\boxed{\frac{\partial L}{\partial W_{old}}}}_{\text{Small value}}}_{\text{Small Value}}$$

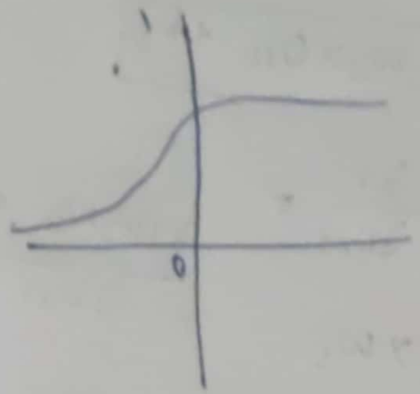
$$W_{new} \approx W_{old}$$

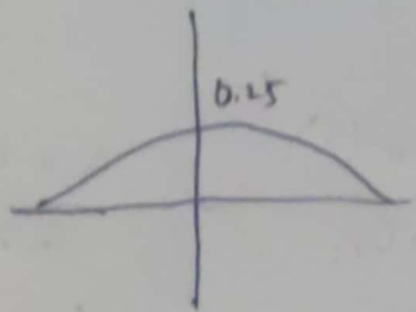
→ To fix this issue we use different Activation function

Activation Function

1. Sigmoid Activation Function
2. Tanh
3. Relu
4. parseint
5. ELU
6. Swiss

① Sigmoid Activation function





Advantage

① clear prediction
1 or 0

Disadvantage

① prone to vanishing gradient problem

②. Function output is not zero centroid.
↳ efficient weight updation

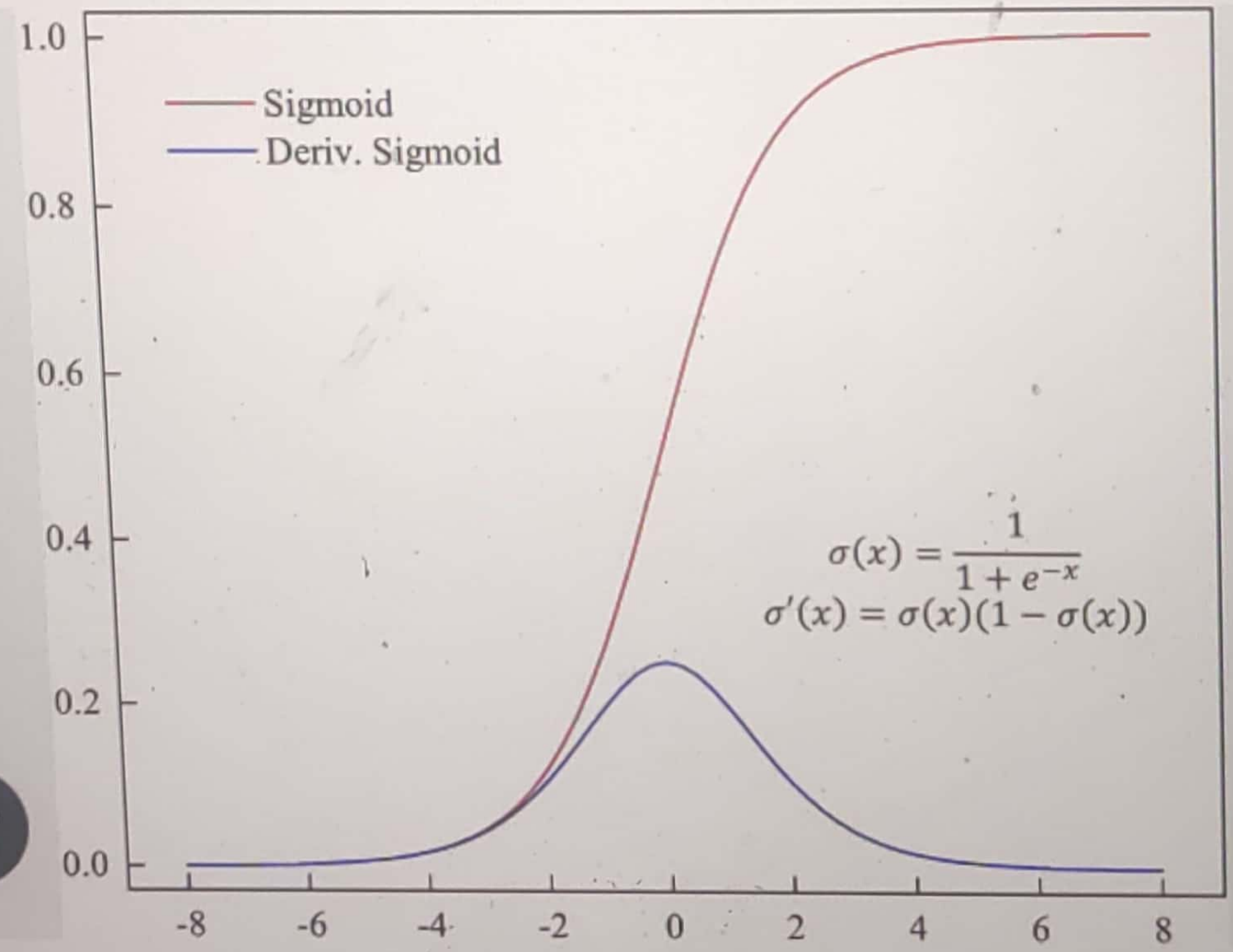
Zero Centroid:- To move the blue point to black point place we use standardization

→ standardization is technique to make point close to zero center.

→ zero center make efficient in weight updation

Scanned with CamScanner

ResearchGate



② Tanh

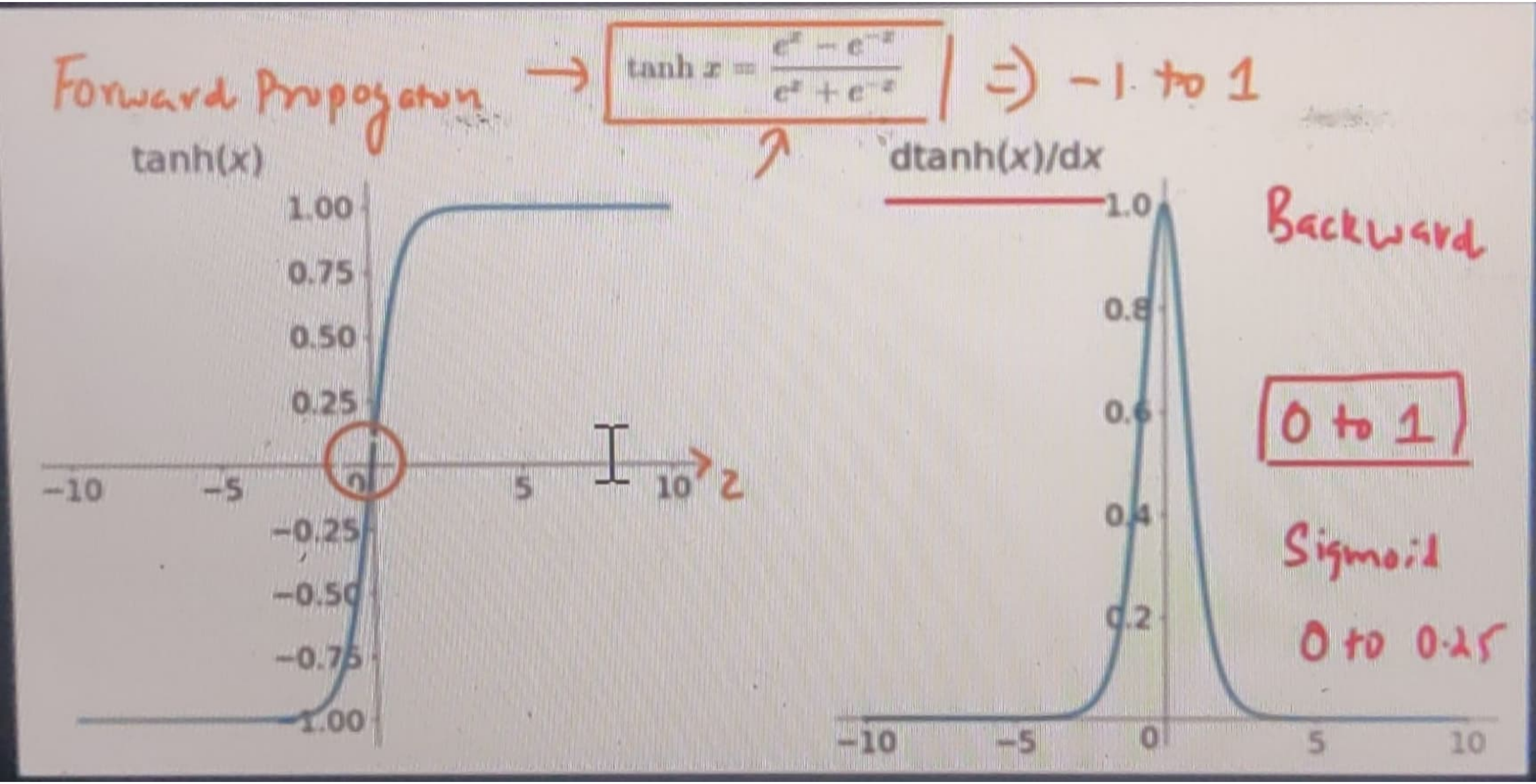
Advantage

① Zero Centriod

Dis - Adv

① Time Complexity

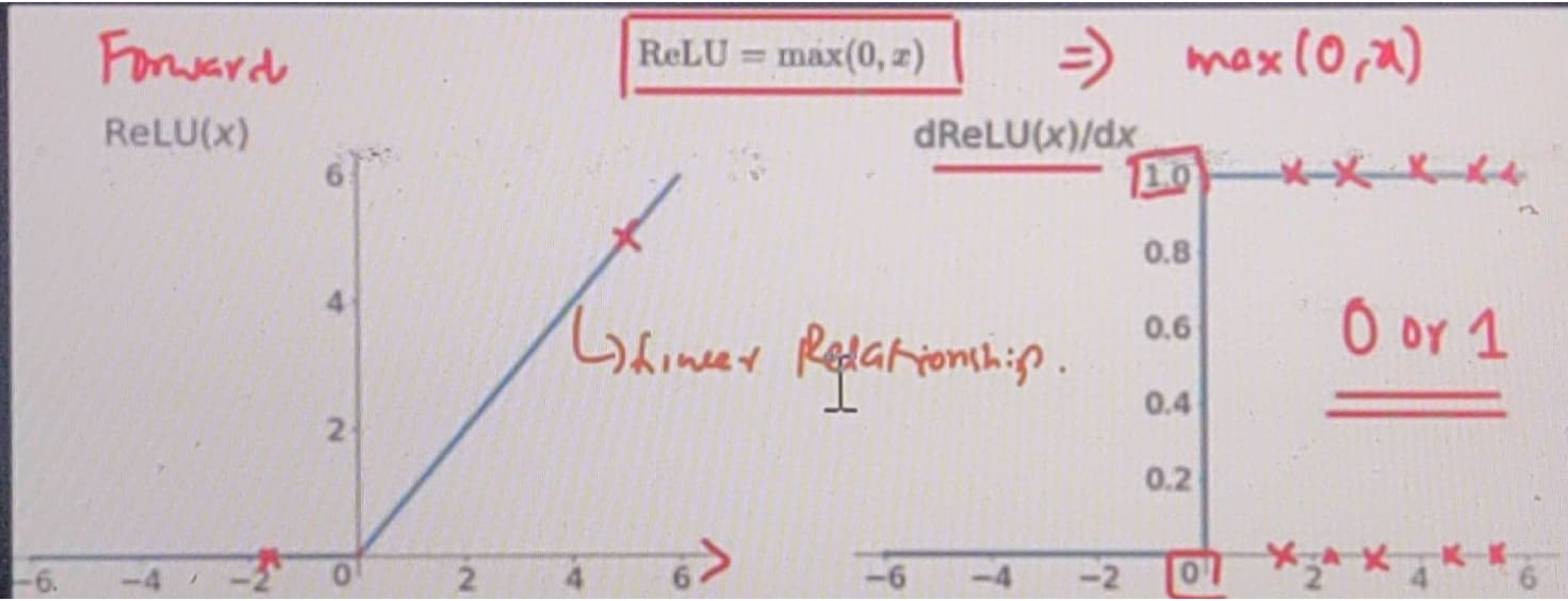
② Vanishing gradient problem still consists.


Forward Propogation
tanh x = (e^x − e^−x)/(e^x + e^−x)
⇒ −1 to 1
tanh(x)
dtanh(x)/dx
z
Backward
0 to 1
Sigmoid
0 to 0.25

③ ReLU Activation Function

Dis-Adv

→ max=0, It will make neuron dead

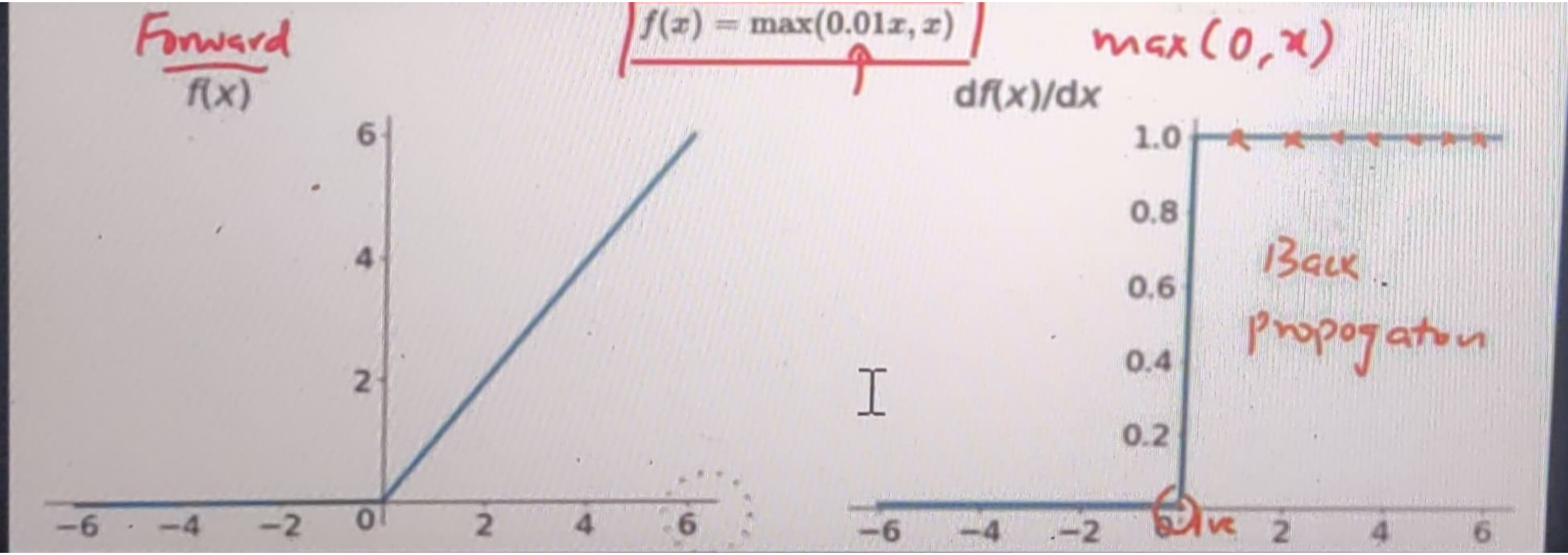

Forward
ReLU = max(0, x)
=> max(0, x)
ReLU(x)
dReLU(x)/dx
Linear Relationship.
0 or 1

④ Leaky Relu or parametric Relu

$\max(\alpha x, x)$ (α = Hyper parameter)

Adv

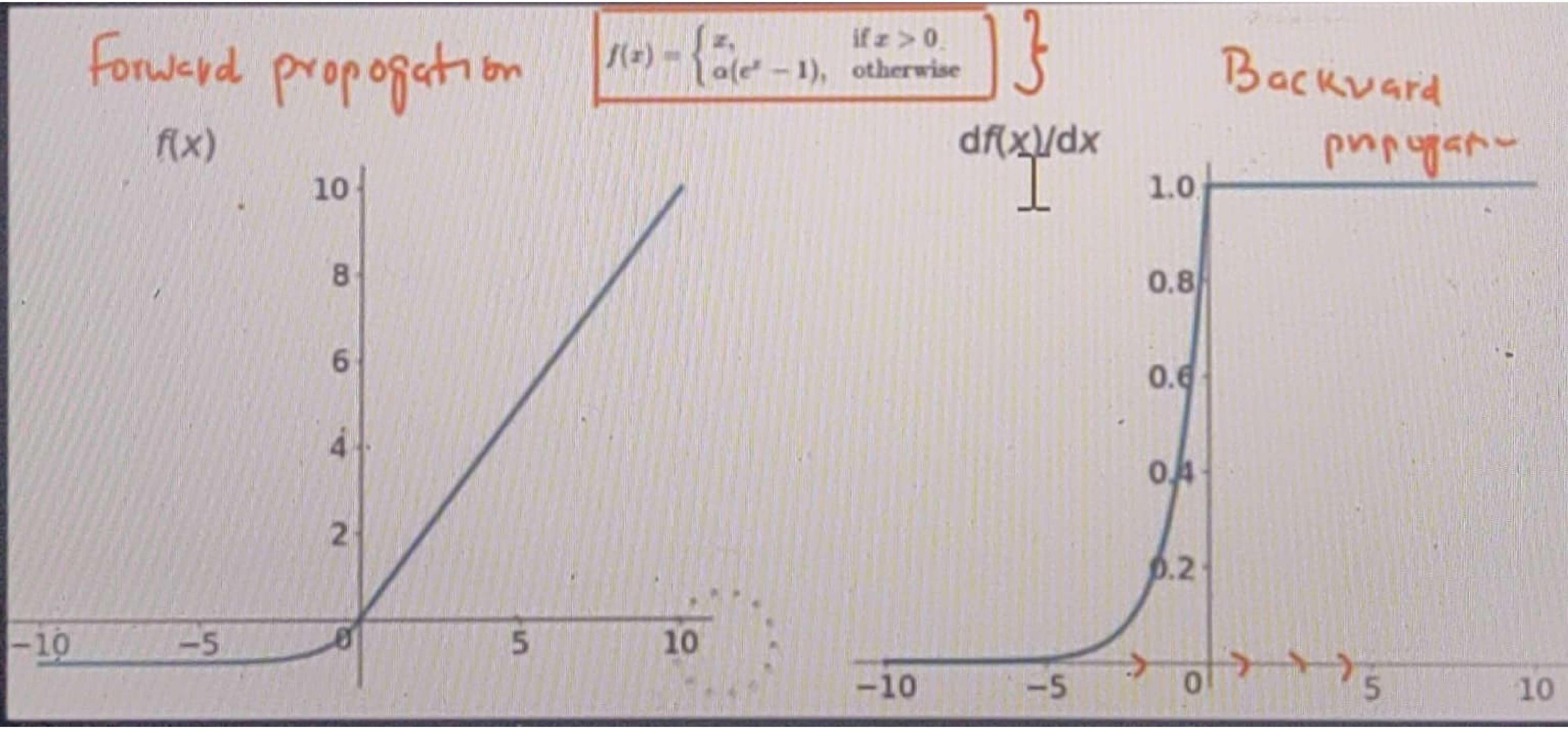
→ prevent dead neuron


Forward
f(x)
f(x) = max(0.01x, x)
max(0,x)
df(x)/dx
Back
Propogation
-6 -4 -2 0 2 4 6
0.2 0.4 0.6 0.8 1.0

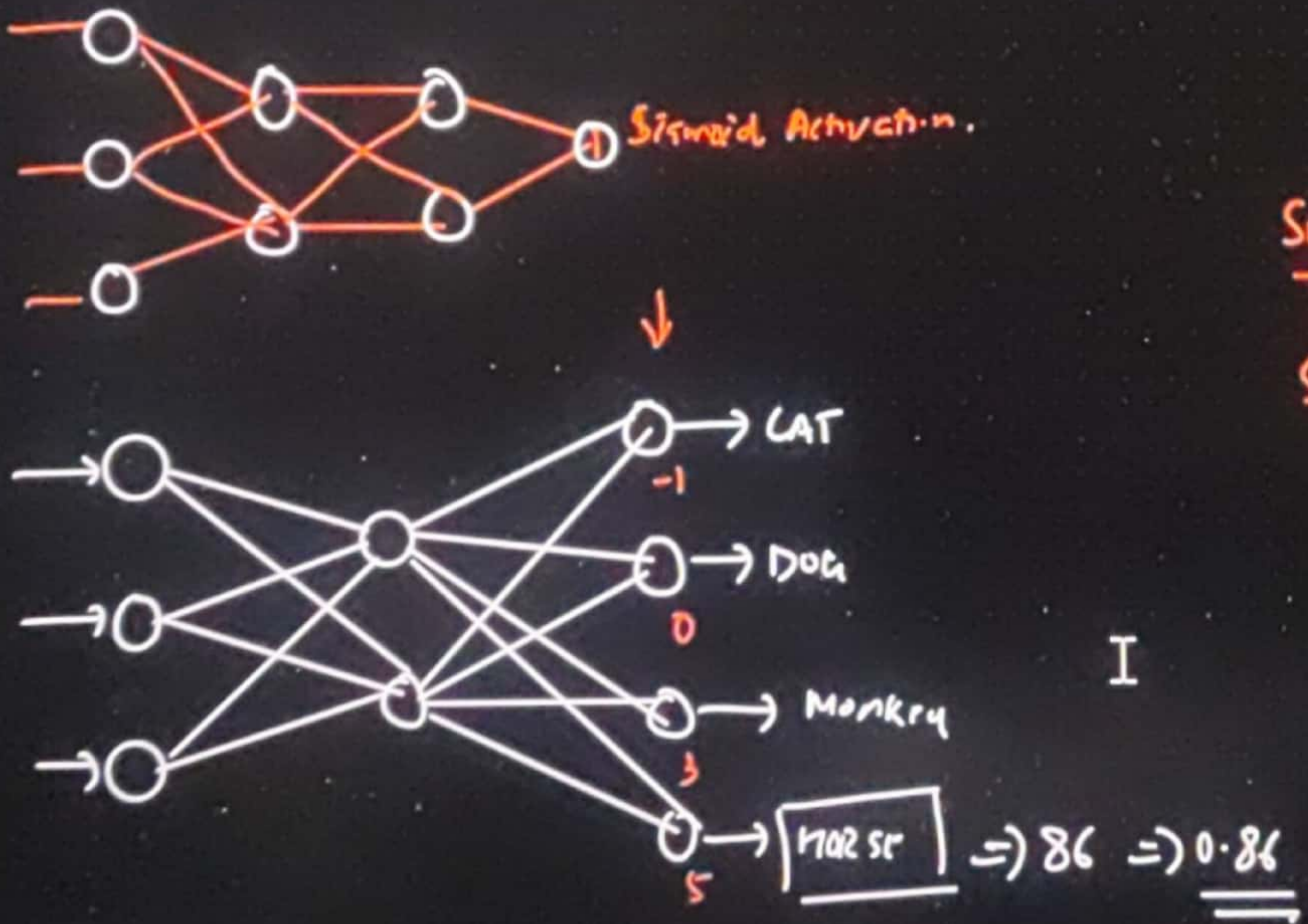
5. ELU (Exponential Linear units)

<u>Dis-Adv</u>

→ Time Complexity.


Forward propogation
$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$
Backvard
propogar-
f(x)
df(x)/dx
10
8
6
4
2
−10
−5
0
5
10
1.0
0.8
0.6
0.4
0.2
−10
−5
0
5
10

⑥ Softmax Activation Function



Softmax Activation

$$\text{Softmax} = \frac{e^{y_i}}{\sum_{k=0}^{n} e^{y_k}}$$

Softmax Activatn. $\Leftarrow$

$$\text{Car} = \frac{e^{-1}}{e^{-1+0+3+5}} = 0.00033$$

$$\text{Dog} = \frac{e^{0}}{e^{-1+0+3+5}} = 0.0024$$

$$\text{Monkey} = \frac{e^{3}}{e^{-1+0+3+5}} = 0.0183$$

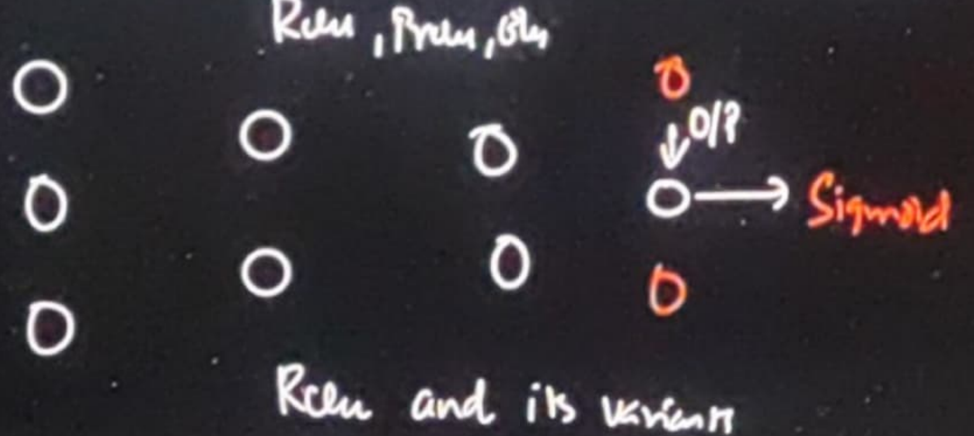
$$\text{Horse} = \frac{e^{5}}{e^{-1+0+3+5}} = 0.1353$$

$$\Pr(\text{Horse}) = \frac{0.1353}{0.00033 + 0.0024 + 0.0183 + 0.1353}$$

$$\approx \underline{\underline{86\%}}$$

Regression $\Rightarrow$ Linear

7 Which Activation function To use When



Regression => Linear

Sigmoid → Binary Classif

Softmax => Multiclass Classification