"Activation Func"

determines if a neuron would be activated or not.

can be thought as a gate

Imp!- → if AF not used, non-linear thing won't work.

Ideal AF

→ non linear.

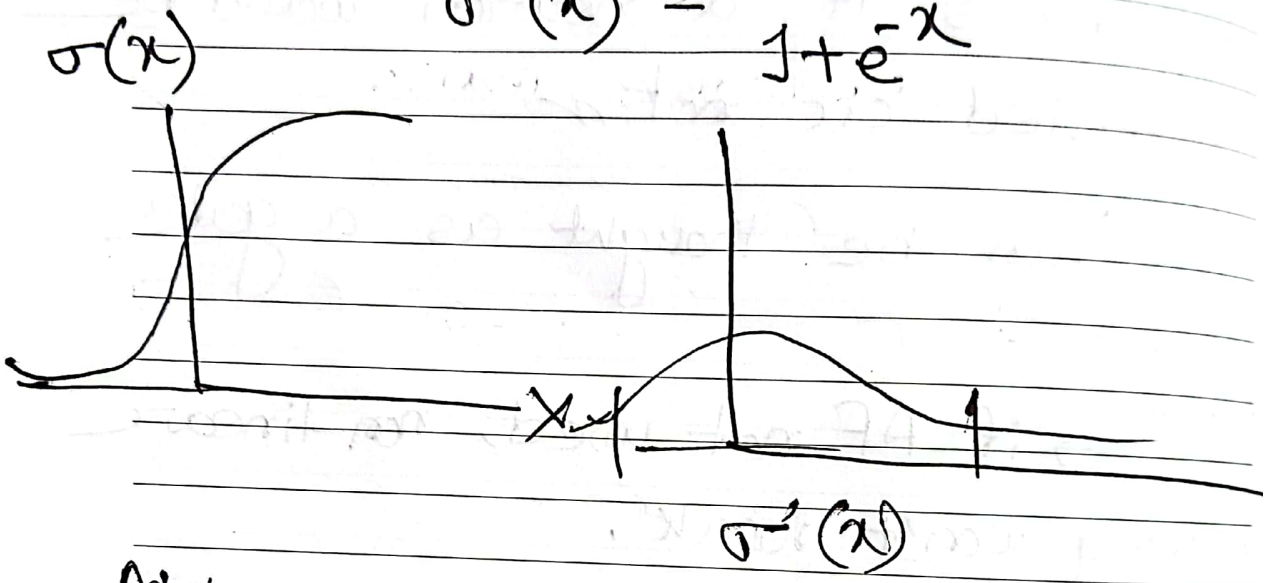
→ Differential

→

Note

Sigmold :-

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



Adv:-

→ output range [0, 1]
↳ helps to acheive non-linearity of data.

→ Differenciable → backp → $\frac{\partial L}{\partial w}$
helps the NN to learn during BP.

Note

Disad:-

1) saturating function $[-\infty, \infty] \to [0,1]$

↳ vanishing gradient problem

↓

backp → update

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

0

Here $w_n = w_0$

training error,

* Yen onek bono value nile 0 hote pare

mane no update will take place.

Note

ii) this func is non-zero centered.

means, normalize hoi na

means, mean eder 0 na.

either sobh (+) or (−),

Problem → training slow hoye jai.

↓

so convergence problem

see, ekbathe either (+) hobe or (−)

like if i want to minimize loss & adjust

then eta to ekta restriction.

Note

iii) As formula is exponential so computation power problage.

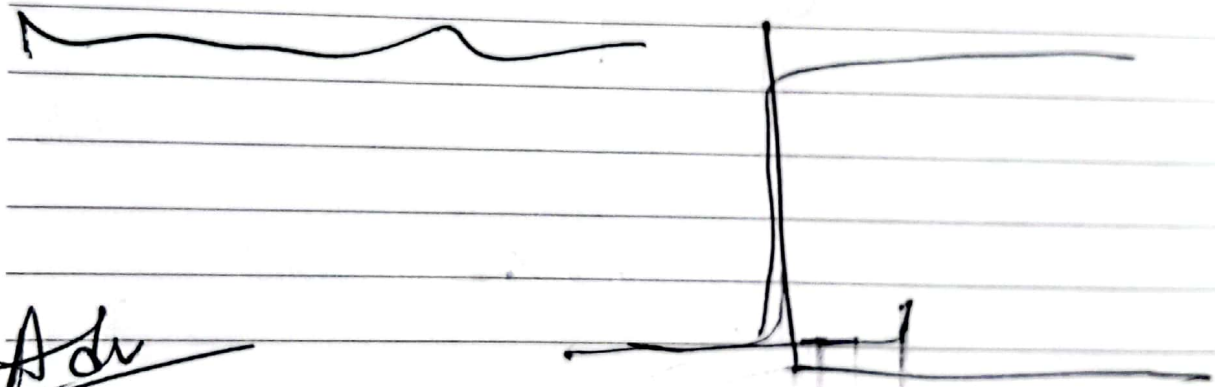
mostly used in output layer.

Note

Tanh :-

output rang → [-1, 1]

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



Adv

i) zero centric
+ve & -ve dutai
pawa jabe,

ii) Differentiable

iii) Non linear.

Note

Disadv

→ saturating func, mane squish kore. → vanishing gradient problem

→ Computational power ↑.

But modified ver version of sigmoid

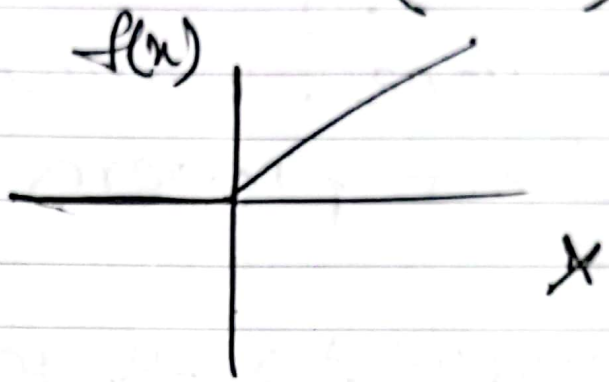
Note

Relu :-

$$f(x) = \max(0, x)$$

neg value $= 0$

pos value $= x$.



adv :-

i) Non linear.

ii) Not saturate in +ve region
↓
positive dike joto khushi jete pare.

iii) computationally inexpensive

iv) main adv :- it activates only one neuron at a time.

Note

CamScanner

Disad :-

i) Not zero centrice.

ii) not differenciable.

iii) for (-) input do nothing.

iv) dead neuron for (-) dataset.

Note

leaky relu:-

adv: i) overcome dead neuron

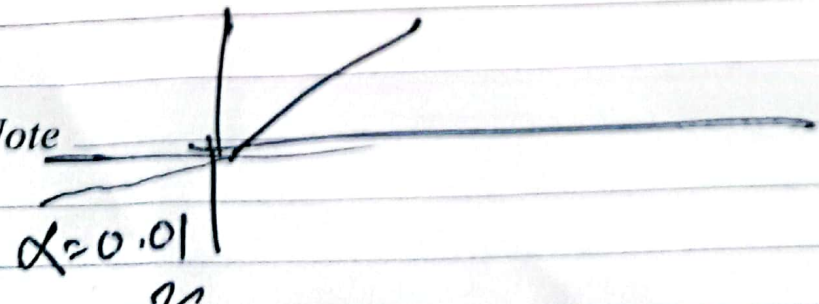
ii) less comp power.

dis:-

does not provide much learning for (-) dataset.

neg hole $\alpha = 0.01$ constant value neya hoi.

Note



Binary classification → sigmoid

Multiclass classifi → Softmax

Softmax :-

$$f(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

comb of mul sig func.

all sigmoids ← $\sum_j e^{x_j}$

output would be in 0 to 1.
can be thought as probability.

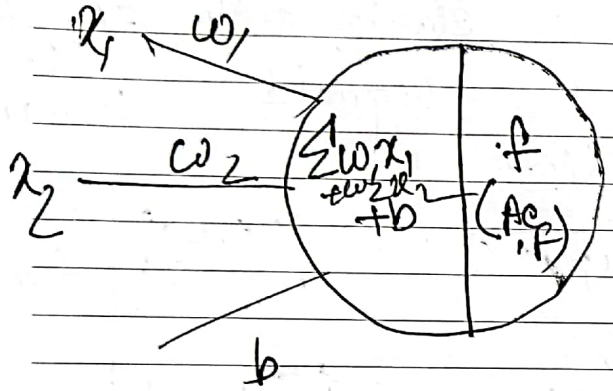
sum of output = 1.

adv :- multi class, probalistic output.

dis :-

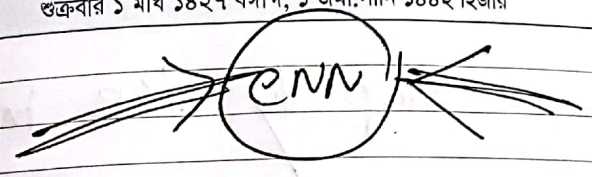
Vanishing gradient descent problem.

Note



	HL	out
Regression –	Relu, ~~Leaky~~ Relu, tanh	linear
Classif →	~~Sigmoid~~. n	sigmoid, softmax,

Note



convo vs pooling.

→ keep track of object

Convo: → convo layers analyze the objects by looking at patterns.

Key property: Translation equivarence: means if an output/object moves by, the convo output also moves by that amount.

Basically scanning the image with a magnifying class to determine where what is what

Output: – similar to original image just detecting the patterns.

Note

CamScanner

pooling :- simplify the image by picking the most imp information.

key property : translation invarience.

ie if image change a lil bit doesn't affect.

pooling reduces the image size & make the model focus on key features rather than exact positions.

Note

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \end{bmatrix} \qquad \begin{bmatrix} -1/2 & 1 & -1/2 \\ -1/2 & 1 & -1/2 \\ -1/2 & 1 & -1/2 \end{bmatrix}$$

$0 + 1 + 0 + 0 + 1 + 0 - 1/2 + 1 - 1/2$

$= 1$

$\rightarrow$ $-1/2 \; -1/2 \; -1/2 \; -1/2 \; -1/2 \; -1/2$

$0 + 1 - 0 - 1/2 + 1 - 1/2 - 2$

$$\begin{bmatrix} 2 & -2 \\ -1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1/2 \\ -1/2 & 4 \end{bmatrix}$$

Note

CamScanner

AlexNet is a CNN network
groundbreaking model to improve
image classification.

Dataset →
large number of image
for training AI models
1 million image labeled with
1000 different categories.
→ crowdsource.

Note

what did it solved?
→ goal was to classify image
into one of the 1000 categories.

Why imp?
→ how effective deep learning
could be for image classification
→ reduced classification errors
significantly.
→ 60M parameters
→

Note

CamScanner

key features:-

→ layer - 8 (5 conv + 3 fully convo)

→ Ac. f → Relu.

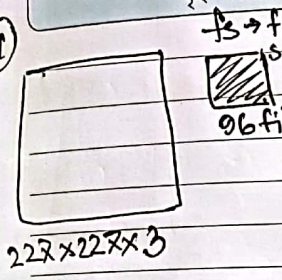
→ to prevent overfitting, dropout layer were used

→ max Pooling,

convo → Relu → Pooling
↓
local response normalization

Note

(i)



$n=227$
$f=11$ $p=0$
$s=4$

$\rightarrow \left[\frac{n-f+2p}{s}+1\right] \times \left[\frac{n-2p+f}{s}+1\right]$

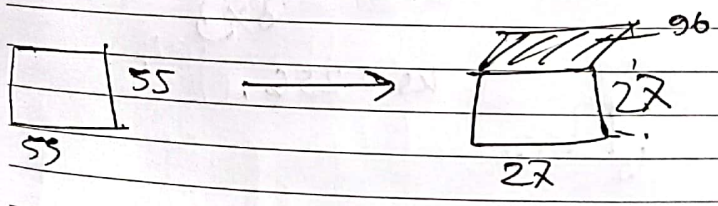
$= 55 \times 55$

so $n=55$ $d=96$ → no of filter depth
$w=55$

1st convo layer.

(ii) Overlapping max Pool

Same formula,



Note

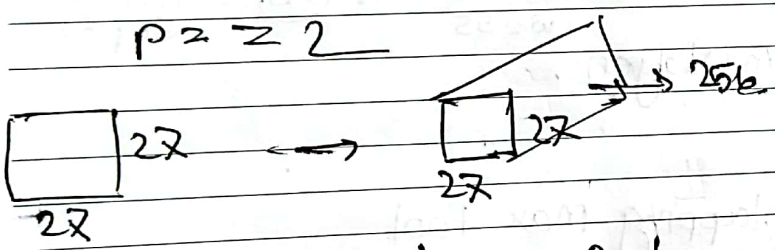
2nd convo layer →

5x5 filter size

256 kernels

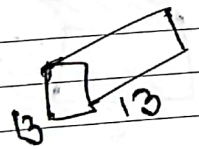
stride = 1

p = 2



↓ max Pool, s = 2

3x3



ks = 256,

Note

Residual connections are a clever way to make deep neural networks, particularly CNN easier to train.

introduce a skip connection where input to a layer can skip to some layers.

<u>why we need?</u>

→ vanishing gradient problem, during backpropagation, hard to update weight.

→ Underfitting:- above 56 layers noise created.

Note

skip some computations.

residual layer output

$(H)^{l}(x) = F(x) + x$

$F(x)$ → changes needs to be learned

x → input

With small datasets, there are some challanges like overfitting as there ain't enough value.

Note

Transfer learning can help here.

it allows you to re-use pretrained models.

like when cnn's are trained on large datasets like image net (1000 categories) it learns the basics and general patterns,

→ These learned patterns can be reused for different tasks

Note if new dataset is much smaller