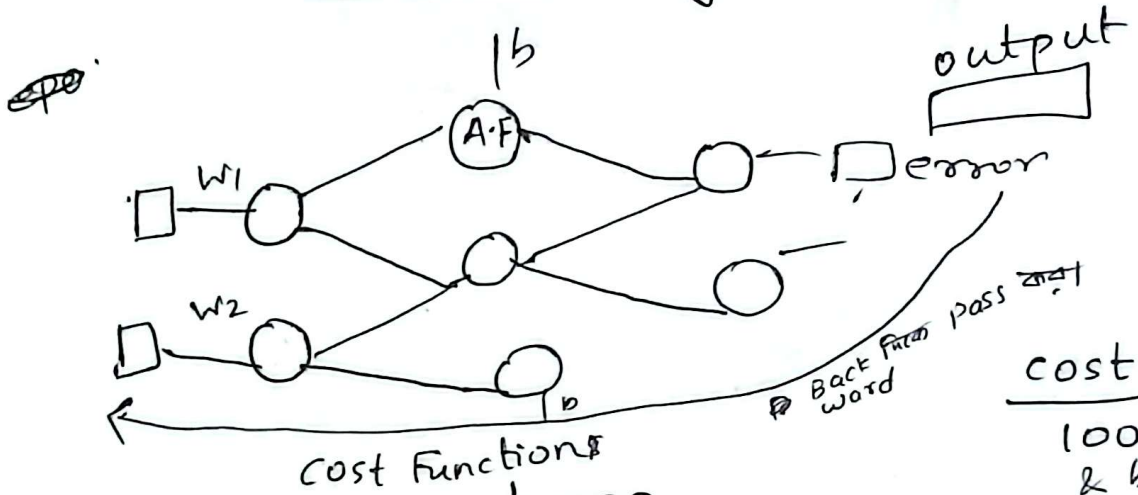
Backward Propagation



Weights & Bias change

Backward ~~func~~ pass করে, weight & bias update

epoch – Forward & backward এর ~~propagation~~ Combination

Back propagation is the famous algorithm for training

$$e_j = d_i - y_i$$

$$\varepsilon_{av} = \frac{1}{2N} \sum e_j^2$$

19 Aug 2025 ~~Backward~~

1) Introduction
2) Perceptron
3) AND gate, OR gate using single
   XOR can't use single per
   . Hidden Neuron
   Activation Function and its Type
4) Softmax
   Biological cell
   Universal Approximation Theory
   line Bend, video
   Cost Function
   Binary classification
   Multi class
   MSE, MAE
   Cost function, error find
   Optimization- parameter optimize
   Gradient Descent - ভুল বের করে, local
   Mini-batch, momentum, SGD, RMSprop, Adam
   Performance Metrics
   Recall, Precision কোথা, কখন use করা হয়
   F1-score

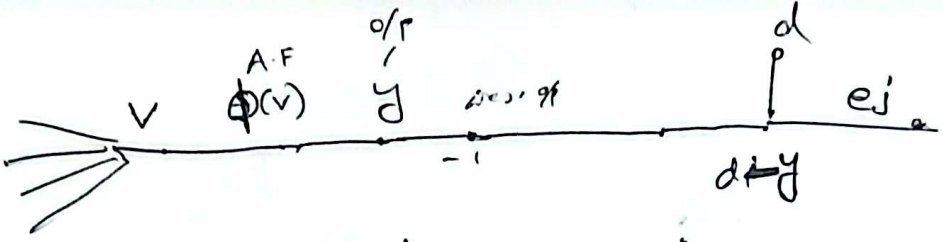
5) Adam (Adaptive Moment Estimation):-

(1) It is used for training machine learning models.
(2) It is an optimization algorithm.
(3) Adam adapts the learning rate for each parameter individually based on the historical gradients of those parameter.

Swarm Intelligence Algorithms

1) Ant colony Optimization –

(1) Ant colony optimization is inspired by the behavior of ants.
(2) Ants find the shortest path between their colony and food sources by laying down pheromones which guide other ants to the path.



If any update that means Δw

$$\Delta w_{ij} = \eta \times \text{local gradient} \times \text{input signal } (y_i)$$

η (eta) — Learning rate

We should calculate local gradient

Hidden layer and output layer both different

local gradient

- output layer
- Hidden layer

* Example

output unit

$$\delta_5 = y(1-y)(y_{target} - y)$$

$$= 0.69(1-0.69) \times -$$

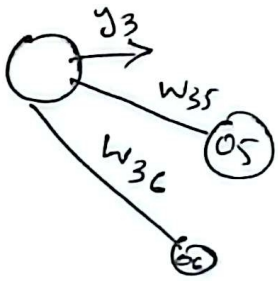
$$O \xrightarrow{y_6} y \qquad y_6(1-y)(y_{target} - y)$$

$$\delta_5 = -0.0406$$

because it it 6th

Hidden unit

$$\delta_3 = y_3(1-y_3)\, w_{35} * \delta_5$$



$$y_3(1-y_3) \Sigma (w_{35} \times \delta_5 + w_{36} \times \delta_6)$$

if 4 then sum 4

$$\delta_4 = y_4(1-y_4) w_{45} \times \delta_5$$

if find w_{45}

$$\Delta w_{45} = \eta \delta_5 y_4$$

$$= 1 \times -0.0406 \times 0.66$$

$$= -0.0269$$

$$w_{45}(new) = w_{45}(old) + \Delta w_{45}$$

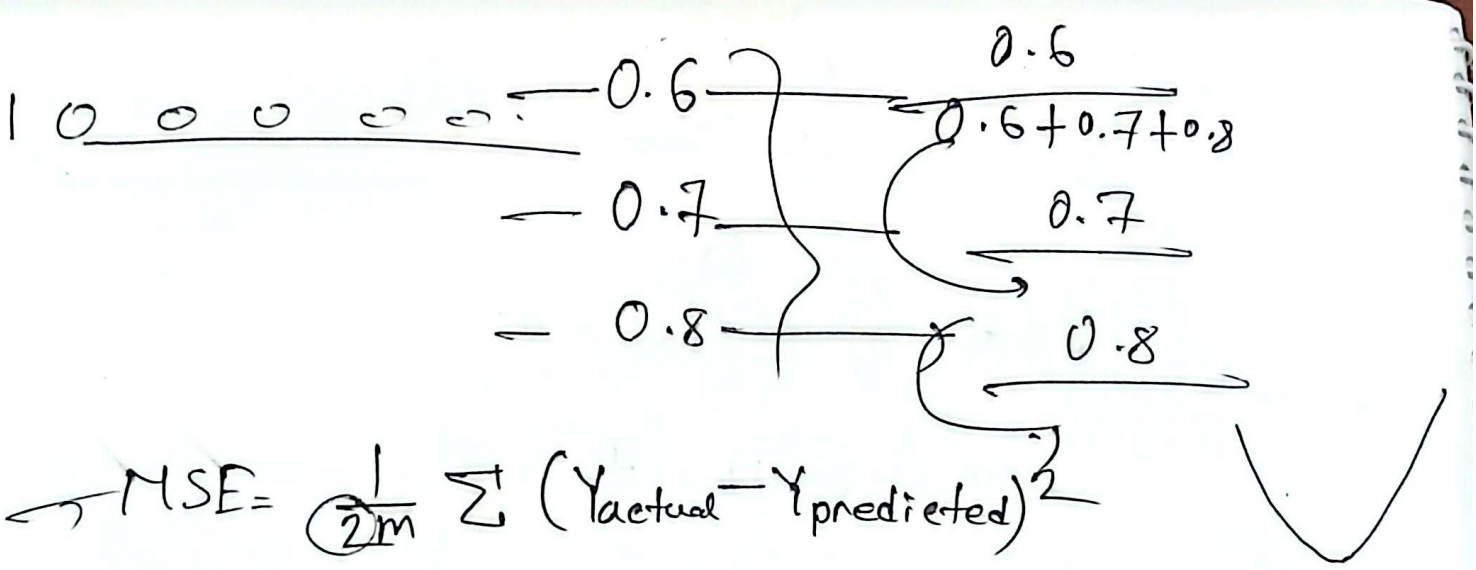
Machine Learning

Biological Neuron

~~to~~ Linear Regress — Cost & gradient

Logistic Regress → cost Function

Loss and cost Function



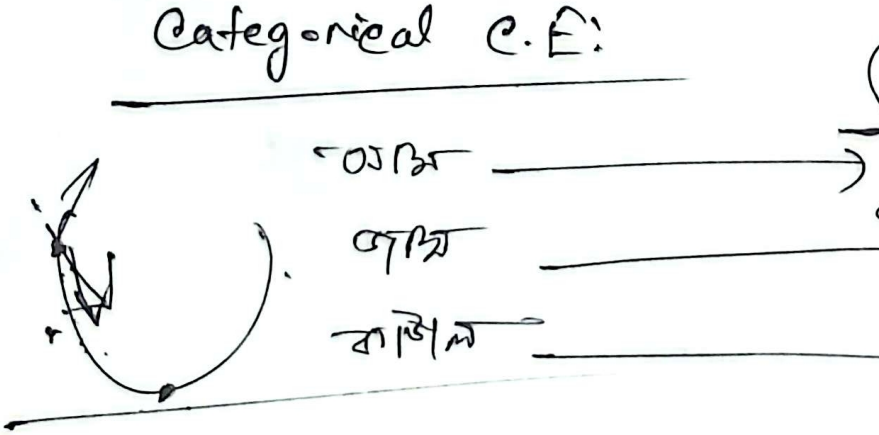
→ MSE = $\frac{1}{2m} \sum (Y_{actual} - Y_{predicted})^2$

→ MAE = $\frac{1}{m} \sum |Y_{actual} - Y_{predicted}|$

Classifica

1 0

Categorical C.E:



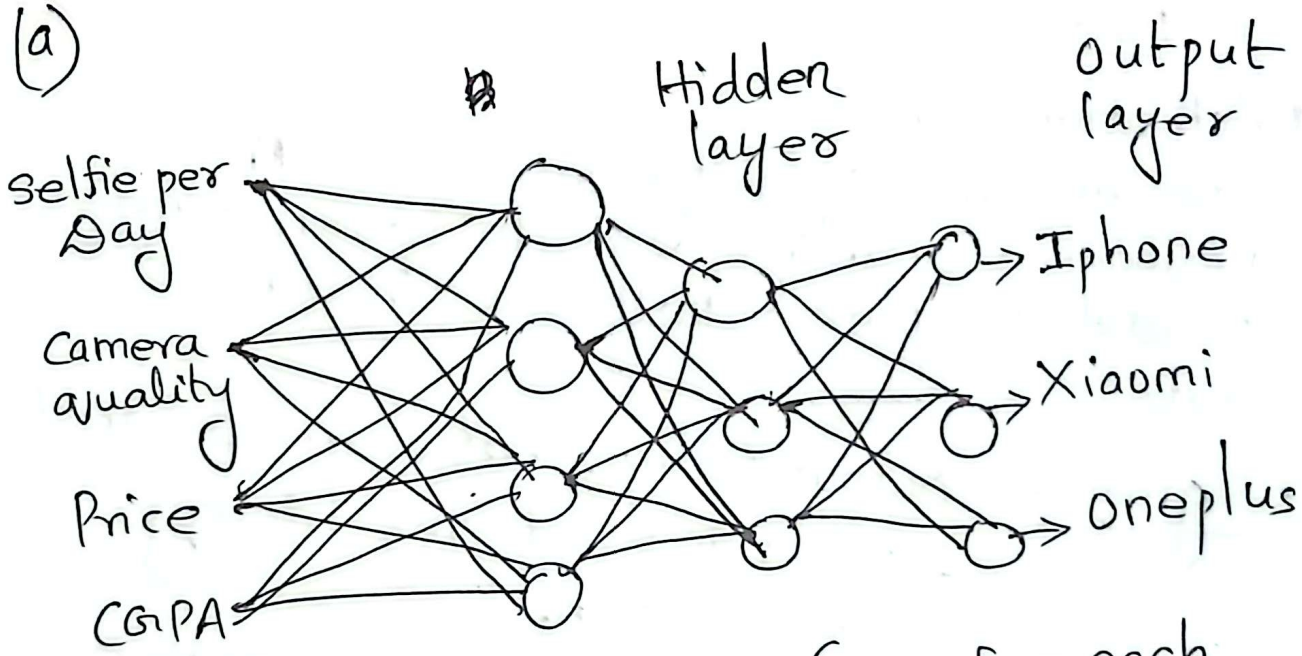
one-hot encoding

Discretization

0.6	0.2	0.8
0	0	1

$$\theta_{new} = \theta_{old} - \eta \frac{d(\text{loss})}{d\theta_{old}}$$

3) (a)



Input layer: 4 neurons (one for each feature)

hidden layer: 2 hidden layer with 4 and 2 neurons respectively using ReLU activation function for non-linearity.

Output layer: 3 neurons (one for each phone model using softmax function to handle multiclass classification.

Loss function: categorical cross-entropy which is ~~suitable~~ appropriate for multi-class problems.

3) b) Analysing datasets, appropriate cost function binary cross entropy

$$J = \frac{-1}{n} \sum \left[y \log(\hat{y}) + (1-y) \log(1-\hat{y}) \right]$$

Here $n = $ class $= 3$

For data 1

$$1 \log(0.85) + (1-1) \log(1-0.85)$$

$$= 0 - 0.0705$$

For data 2

$$0 \log(0.12) + (1-0) \log(1-0.12)$$

$$= -0.0555$$

For data 3

$$1 \log(0.92) + (1-1) \log(1-0.92)$$

$$= -0.0362$$

$$\text{Average cost} = \frac{-1}{3} \left[(-0.0705) + (-0.0555) + (-0.0362) \right]$$

$$= 0.054$$

⇒ Dense layer / Fully-connected layer

When neuron is connected to every other neuron in the next layer.

⇒ Sparse layer – when not connected

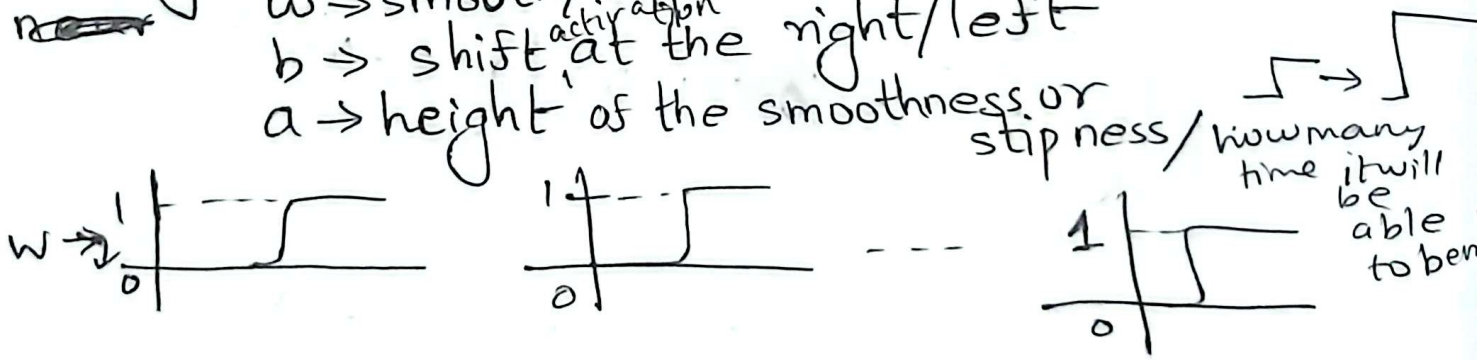
Video = Why neural networks can learn any function:

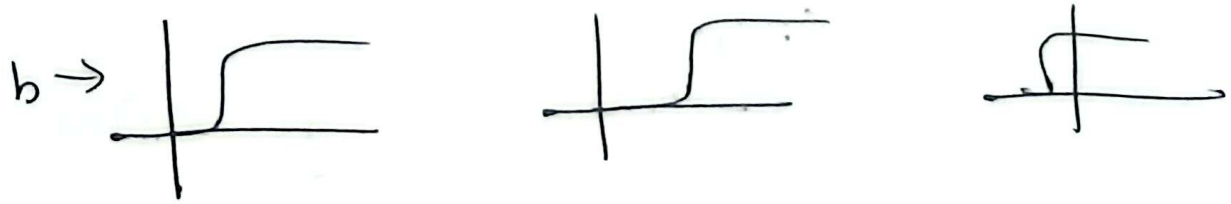
adding more neurons can add more steps

w → smooth/stip

b → shift at the right/left

a → height of the smoothness or stipness / how many time it will be able to ben





- two neuron in hidden layer

Role

(1) Activation function – Activation function introduce non-linearity into the neural network, enabling it to learn and model complex patterns in the data.

- Sigmoid – output values between 0 and 1. Often used for binary classification.
- ReLU – Outputs the input if positive otherwise zero.
- tanh – Outputs value between 1 and -1 centered the data; often used in hidden layer.
- Softmax – extends logistic regression to multiple classes.

↳ o/p zero for negatives, linear for positives.

2) b) Binary cross ~~entry~~ entropy is appropriate.

$$J = \frac{-1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i) \right]$$

Sample, $n = 3$

y_i = can I go on a trip

$\hat{y}_i$ = predicted value

For sample 1:

$$[0 \log(0.15) + (1-0) \log(1-0.15)]$$
$$= -0.070$$

For sample 2:

$$[1 \log(0.40) + (1-1) \log(1-0.40)]$$
$$= -0.397$$

For sample 3:

$$[1 \log(0.92) + (1-1) \log(1-0.92)]$$
$$= -0.0362$$

Now,

$$J = \frac{-1}{3} [(-0.070) + (-0.397) + (-0.0362)]$$
$$= 0.16773$$

1) a) Linear Regression hypothesis:

- $h_\theta(x) = \theta_0 + \theta_1 x$

Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \{h_\theta(x) - y_i\}^2$$

Gradient descent update Rules.

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1)$$

$\theta_0 = 5$, $\theta_1 = 2$, $\alpha = 0.1$

$h_1\theta(x) = 5 + 2 \cdot 250 = 505$

$h_2\theta(x) = 5 + 2 \cdot 220 = 445$

$h_3\theta(x) = 5 + 2 \cdot 280 = 565$

$h_4\theta(x) = 5 + 2 \cdot 235 = 475$

Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \{h_\theta(x) - y_i\}^2$$

$$= \frac{1}{2 \cdot 4} \left| (505 - 18)^2 + (445 - 15)^2 + (565 - 22)^2 + (475 - 17)^2 \right|$$

$$= 115835.25$$

Gradient Descent

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} (\theta_0, \theta_1)$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{4}(487 + 430 + 543 + 458)$$

$$= 479.5$$

$$\frac{\partial}{\partial \theta_1} J = \frac{1}{4} \sum |h_\theta(x) - y_i| x_i$$

$$= \frac{1}{4}(487 \cdot 250 + 430 \cdot 220 + 543 \cdot 280 + 458 \cdot 235)$$

$$= 119005$$

$$\theta_0 = 5 - 0.1(479.5) = -42.95$$

$$\theta_1 = 2 - 0.1(119005) = -11898.5$$

$\therefore$ cost $J = 115835.25$

update $\theta_0 = -42.95$

$\theta_1 = -11898.5$

1) Supervised learning- Learning with labeled data training.
Eg:- Email spam classification
Input: Email content, Output: spam/Not spam (labeled)

~~Unsupervised learning~~

Supervised learning	Unsupervised Learning	Reinforcement learning
(1) learning from labeled data.	(1) Identifies patterns of unlabeled data.	(1) Learning from interaction with the environment.
(2) Types of problem: Classification, Regression	(2) Types of problem: Clustering, Association	(2) Types of Problem: Sequential Decision making
(3) Algorithm: SVM, Decision Tree, Neural networks.	(3) Algorithm: k-means, PCA, Auto encodees	(3) Algorithm: Q-learning
(4) Applications:- Medical diagnosis, Fraud detection.	(4) Applications:- Cyber security, e-commerce.	(4) Autonomous driving, Robotics gaming.
(5) Eg:- Email spam classification	(5) Eg:- Customer segmentation I/p- Customer purchase data, o/p, Cust groups	(5) Eg:- Gameplaying (Chess, go) Agent: game player Environment - Game board Reward: win/loss/draw

2) b) Binary cross entropy is appropriate

$$J = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log \hat{y_i} + (1 - y_i) \log (1 - \hat{y_i}) \right]$$

For sample 1:

$$\left[0 \log (0.15) + (1 - 0) \log (1 - 0.15) \right]$$

Already done once

$$= -0.07058$$

3) b) Can we use the same linear regression cost function for logistic regression problem? Justify your answer.

⇒ No, we cannot use the linear regression (MSE) cost function for logistic regression.

- With sigmoid hypothesis, MSE makes the cost function non-convex, leading to multiple local minima and slow convergence or inefficient learning.
- Instead, logistic regression uses the log loss (cross-entropy) cost function which is convex and ensures efficient training and correct optimization.

Loss function: Binary cross entropy

$$J = \frac{-1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y_i}) + (1-y_i) \log(1-\hat{y_i}) \right]$$

Justification:-

ReLU : outputs the input if positive, otherwise zero. Prevents vanishing gradients, computationally efficient.

Sigmoid output : output value between range 0 and 1, often used for binary classification.

Binary Cross-Entropy :- Optimal for binary classes such as two classes. Penalizes confident wrong directions.

3) b) Binary cross entropy is the appropriate cost function.

$$J = \frac{-1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y_i}) + (1-y_i) \log(1-\hat{y_i}) \right]$$

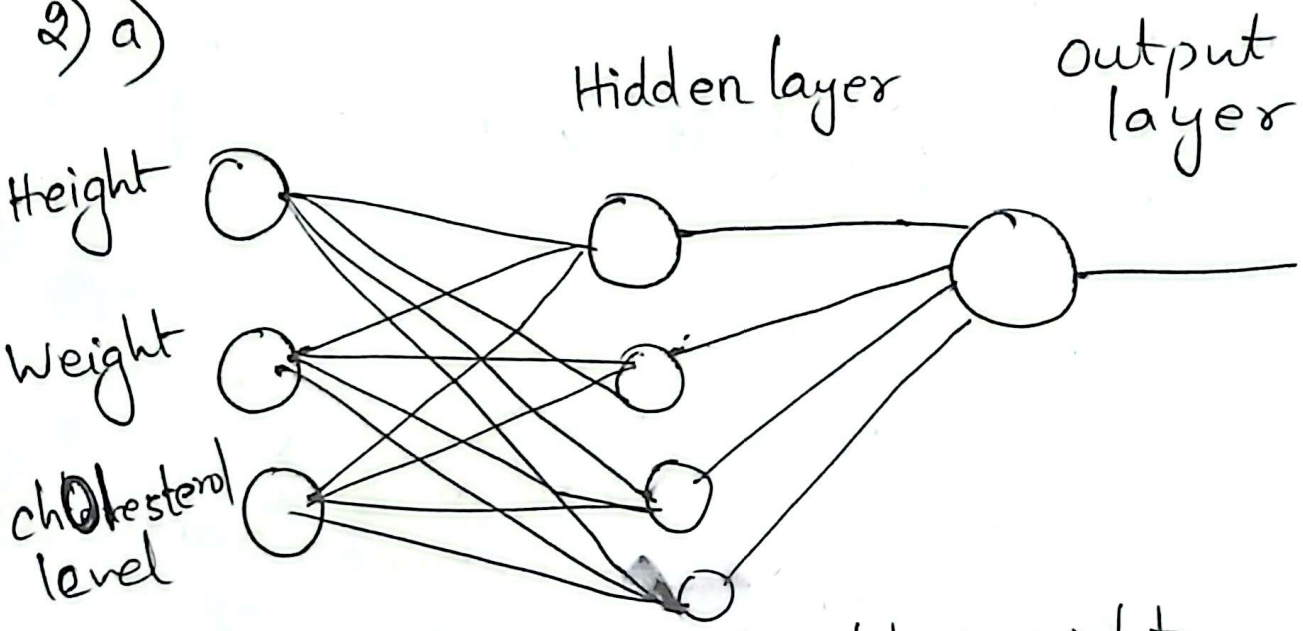
sample; $n = 3$

for sample 1

$$1 \log(0.85) + (1-1) \log(1-0.85)$$

$$= -0.070$$

2) a)



Input layer: 3 neurons (Height, Weight, Cholesterol) one for each feature.

~~Output~~ Hidden layer: 4 neurons ~~with~~ using ReLU activation function to introduce non-linearity.

Output layer = 1 neuron using ~~softmax activation function to handle multiclass~~ Binary cross entropy for binary classification.

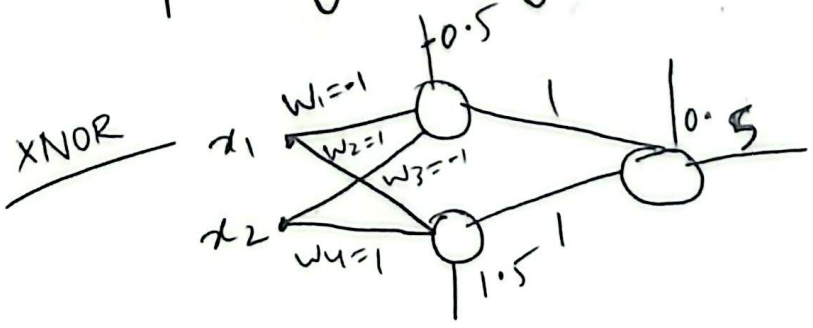
Activation functions:-

Hidden layer: Sigmoid or ReLU

$ReLU(x) = \max(0, x)$, $sigmoid(x) = \frac{1}{1+e^{-x}}$

Output layer: Sigmoid (for probability output)

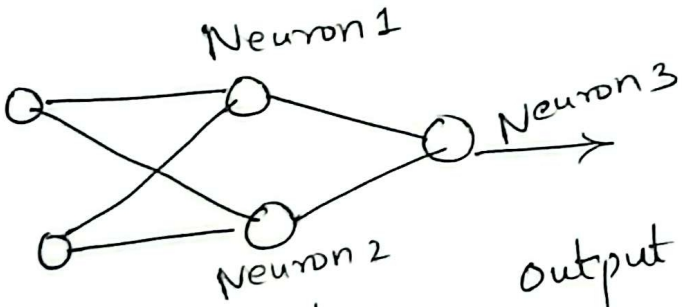
XNOR

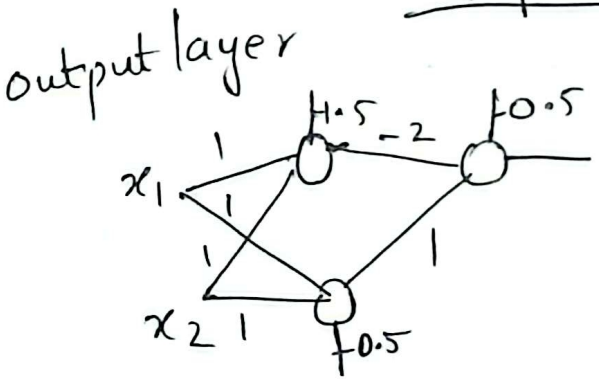


Logic operation	W1	W2	bias
OR	+1	+1	-0.5
AND	1	1	-1.5
NOR	-1	-1	0.5
NAND	-1	-1	1.5

1) b) Determine the parameters required to design an XOR gate using the perceptron rule with threshold value of 1

x_1	x_2	XOR output
0	0	0
0	1	1
1	0	1
1	1	0





When input

$(0,0) = (0\times1)+(0\times1)-1.5 = -1.5 < 0 = 0$

$(0\times1)+(0\times1)-0.5 = -0.5 < 0 = 0$

$(0\times-2)+(0\times1)-0.5 = -0.5 < 0 = 0$

$(0,1) = (0\times1)+(1\times1)-1.5 = -0.5 < 0 = 0$

$(0\times1)+(1\times1)-0.5 = 0.5 > 0 = 1$

$(0\times-2)+(1\times1)-0.5 = 0.5 > 0 = 1$

$(1,0) = (1\times1)+(0\times1)-1.5 = -0.5 < 0 = 0$

$(1\times1)+(0\times1)-0.5 = 0.5 > 0 \rightarrow 1$

$(0\times-2)+(1\times1)-0.5 = 0.5 > 0 \rightarrow 1$

$(1,1) = (1\times1)+(1\times1)-0.5 = 0.5 > 0 \rightarrow 1$

$(1\times1)+(1\times1)-0.5 = 1.5 > 0 \rightarrow 1$

$(1\times-2)+(1\times1)-0.5 = -1.5 < 0 \rightarrow 0$

For sample 2

$0 \log(0.12) + (1-0) \log(1-0.12)$

$= -0.055$

For sample 3

$1 \log(0.92) + (1-1) \log(1-0.92)$

$= -0.0362$

Total cost:

$J = \frac{-1}{3} \left[(-0.070) + (-0.055) + (-0.0362) \right]$

$= 0.0537$

Q) Explain the problems that may arise while training with gradient descent.

⇒ (1) slow convergence, if learning rate is small

(2) Divergence or oscillation if learning rate is too high.

(3) Local minima or saddle points may trap the algorithm.

(4) Vanishing or exploding gradients make learning unstable.

(5) High computational cost for large datasets when using batch gradient descent.

Q) What is the difference between linear regression and classification models?

⇒ Linear regression: Predicts a continuous numeric value (eg:- price, Temperature)

Classification: Predicts categorical outcomes (eg:- spam/not spam, disease/no disease)

Regression uses metrics like MSE (Mean Squared Error), while classification uses metrics like accuracy, precision and recall.

Q) Compare Biological neuron Vs Artificial Neuron.

Aspect	Biological Neurons	Artificial neurons
1) Structure	cell body, dendrites, soma, axon, synapses	Input nodes, weights, activation function, output
2) Processing	Electrochemical signal	Mathematical computations.
3) Speed	slow (milliseconds)	Fast (nanoseconds)
4) Learning	synaptic plasticity	Weight adjustment algorithms.
5) Connectivity	~10,000 connections per neuron	Variable, typically fewer.
6) Fault tolerance	High redundancy, self-repair	Limited fault tolerance

Biological neurons use ~~electro~~chemical neurotransmitters, artificial use numerical weights, Biological ~~or~~ systems are massively parallel; artificial are often sequential - Biological learning is continuous; artificial learning is batch-based.

Q) To calculate the cost function for linear regression, we use MSE (Mean squared Error).

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(hx^{(i)} - y^{(i)}\right)^2$$

$hx^{(i)}$ → Predicted price

$y^{(i)}$ → Actual price

m = no. of training examples

House	Actual price ($y^{(i)}$)	Predicted price ($hx^{(i)}$)
1	1000	1350
2	1500	1590
3	2000	1800

(1) $(1350 - 1000)^2 = 350^2 = 122500$

(2) $(1590 - 1500)^2 = 90^2 = 8100$

(3) $(1800 - 2000)^2 = (-200)^2 = 40000$

Total cost function

$$J(\theta) = \frac{1}{2 \times 3} \left[122500 + 8100 + 40000\right]$$

$$= \frac{1}{6} \times 170600 = 28433.33$$

~~For first~~

Q) Loss function: Binary Cross-Entropy loss (For binary classification. Also known as log-loss).

$$L = \frac{-1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y_i}) + (1-y_i) \log(1-\hat{y_i}) \right]$$

(natural log - ln)

For first student :-

$$-[1 \times \log(0.85) + (1-1) \log(0.85)]$$

$$= 0.1625$$

For second student :-

$$-[0 \times \log(0.12) + (1-0) \times \log(0.88)]$$

$$= 0.1278$$

For third student:

$$-[1 \times \log(0.92) + (1-1) \times \log(0.08)]$$

$$= 0.0834$$

$$\text{Average cost} = \frac{0.1625 + 0.1278 + 0.0834}{3}$$

$$= 0.1246$$

Q) Loss function used:

Binary cross entropy

$Y_1 = 0, \quad \hat{Y}_1 = 0.15$

$Y_2 = 1, \quad \hat{Y}_2 = 0.40$

$Y_3 = 1, \quad \hat{Y}_3 = 0.92$

$$\text{Loss} = \frac{-1}{n} \sum_{i=1}^{n} \left[y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i) \right]$$

$$= \frac{-1}{3} \quad L_1 = 0 \log(0.15) + (1-0) \log(1-0.15)$$

$$= -0.0705$$

$$L_2 = 1 \log(0.40) + (1-1) \log(1-0.40)$$

$$= -0.398$$

$$L_3 = 1 \log(0.92) + (1-1) \log(1-0.92)$$

$$= -0.0362$$

$$\text{Loss} = \frac{-1}{3} \left[(-0.0705) + (-0.398) + (-0.0362) \right]$$

$$= 0.168$$

1) One-hot Encoding – A method to represent categorical data as binary vector.

2) Label-encoded – A method of converting categorical data into numeric labels (integers).

3) Discretization (Binning):–

A · The process of converting continuous numerical data into categorical bins (intervals)

eg:- suppose Age = [5, 12, 20, 35, 60]

child (0-12)
Teen (13-19)
Adult (20-59)
senior (60+)

Age	Bin
5	child
12	child
20	Adult
35	Adult
60	senior

color	Encoded
Red	0
Green	1
Blue	2

color	R	G	B
R	1	0	0
G	0	1	0
B	0	0	1

Multiclass classification:

Loss function → Categorical Cross Entropy

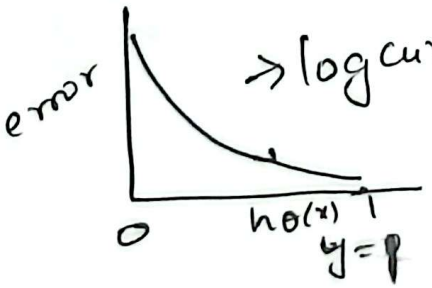
$$\frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y_{ij}})$$

y_{ij} ↓ True label, $\hat{y_{ij}}$ ↓ Predicted label

C = No. of classes
N = No. of samples
For class j

$h_\theta x = \theta_0 + \theta_1 x$ · h → hypothesis

$$cost(h_\theta(x)) = \begin{cases} -\log(h_\theta(x)), & \text{if } y=1 \\ -\log(1-h_\theta(x)), & \text{if } y=0 \end{cases}$$



y = 0

$$Loss = -\sum_{j=1}^{k} y_j \log(1-\hat{y_j})$$

$$cost = \frac{-1}{M}\sum_{i=1}^{M}\sum_{j=1}^{K} y_{ij}\log(1-\hat{y_{ij}})$$

m = no. of training example

31 Aug 2025

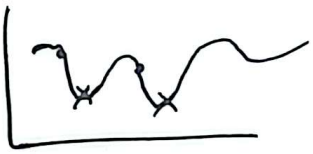
1) last layer node [illegible] কয়টি।
linear, o/p

Fatigue
Positive cause miss করা যাবে না।

		Fatigue	NF
predic	F	Tp	FP
	NF		TN

FN কমাতে হবে
Recall

Possible, [illegible]
GD



Initialization change - better result

2) a) No, not change
Network- Linear Activation Function।
Use proper non-liner A·F

b) Learning rate বেশি হয়েছে
Overshoot

c)

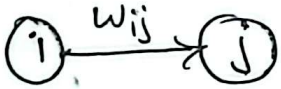
3) a) Data set balance না
Recall

b) Bias

c) change করে করে দেখতে হবে।

31 Aug 2025

Backpropagation

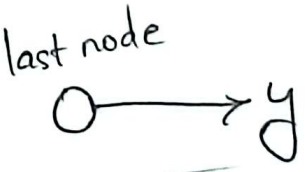


Local gradient - destination

$$\Delta w_{ij} = \eta * \delta_{ij} * input$$

~~If~~ If node

output layer
Hidden layer



$$\Delta w_{ij} = \eta * \delta_{ij} \times input$$

Output unit

$$\delta = y(1-y)(y_{pre} - y_{act})$$

For hidden unit:

$$\delta_3 = y(1-y) \Sigma(0.3x - 0.0406)$$

$$\Delta w_{14} = \eta \times \delta_4 \times 0.35$$

$$w_{14}(new) = w_{14}(old) + \Delta w_{14}$$

8) Attention Mechanism in Neural Networks

Attention mechanism in deep learning enhancing model performance

Understand complex image or translate a sentence from one language to another.

The selective focus is what we refer to as attention

Types :-

1) self-Attention Mechanism
2) Scaled Dot-Product Attention
3) Multi-Head Attention
4) Location-Based attention

deeplearning - Types of neural network

3) Recurrent Neural Network
Repeatedly চলতে থাকে।

use for tasks involving sequences like text, speech or time series.

It handles sequential data in which the current o/p is a result of previous i/p by looping over themselves .to hold internal state (memory)

4) Long short-Term Memory Networks (LSTMs)
Variant of RNNs.
used when need to model long-term ~~sequences~~ dependencies in sequences

I live in Bangladesh

~~where~~.

5) Gated Recurrent Units (GRUs)

6) Radial Basis Function Networks
Good choice for function approximal-

9) Generative Adversarial Networks
generator and discriminator
↓
Combine Real Data samples & Generated fake samples.

Generated ↓ fake sample

Till local gradient all calculations

$\Delta b_1 = \eta \times \delta_4 \times 1$

$b_1 = 0.4$

Htu

bias input always 1

From google

Metrics format

$\begin{bmatrix} \Delta b_1 \\ \vdots \end{bmatrix}$

ai by hand propagation can you calculate back

$\begin{bmatrix} - & - \\ - & - \\ - & - \end{bmatrix}$

$X \cdot W^T$

In Exam it will be given in Matrics form

also can be given in Network system

2 Sept 2025

GeeksforGeeks

* Types of neural networks

1) Feedforward Neural Networks

Training - Backpropagation

When the data is static and has no sequential dependencies.

2) Convolutional Neural Networks (CNNs)

I am

normal mathematical operation

Input layer → convolution layer → Max pooling layer → Dense layer → Outp layer

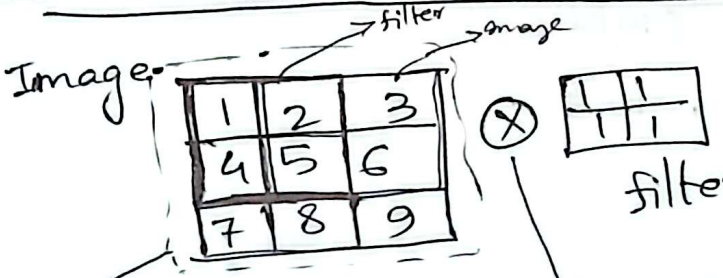
Used When working with image, video or grid-structured data

computer pixel val

2	3	7
5	6	7
8	9	1

একটা matrix এর convolutional.

Convolutional Neural Networks (CNNs) –



filter << image

Pixel to pixel multiply then সব pixel add

$1\times1+2\times1+4\times1+5\times1 = 12$

convolution operation

ছোট হয়ে গেছে because padding করি নাই

shifting is called

Pixel কমে গেছে কমে না যাওয়ার জন্য image বাড়াতে হয়

12	16
11	16

In Image if we use sharpening filter then we get sharp image.

12	16
11	16

* Padding

$5\times5 \rightarrow 7\times7$

$\downarrow$

5×5

Blur

H-filter

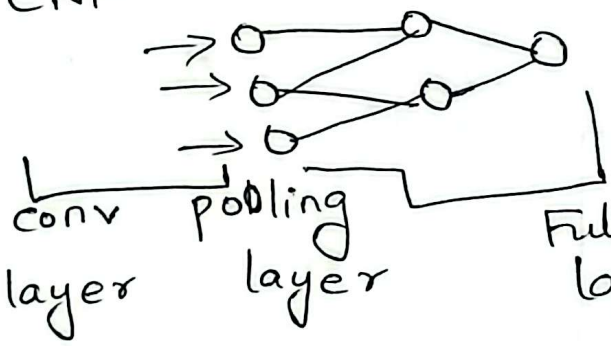
Next class Read Convolution operation and come.

7 September 2025

Deep learning with CNN

* How to choose a Deep Net
  - CNN
  - RNN

* CNN

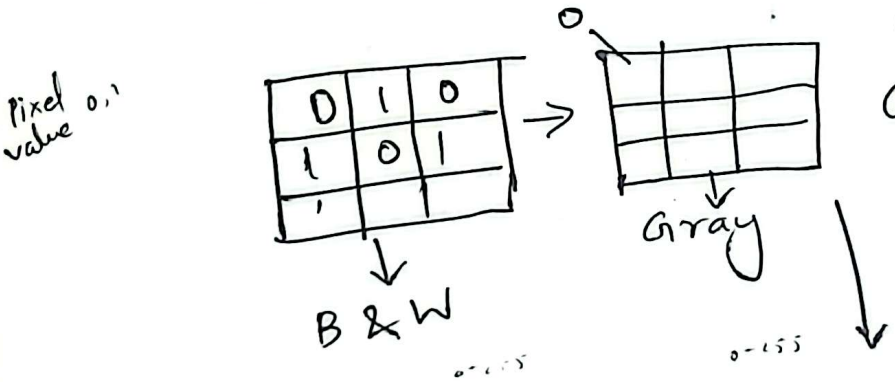


Fully connected layer

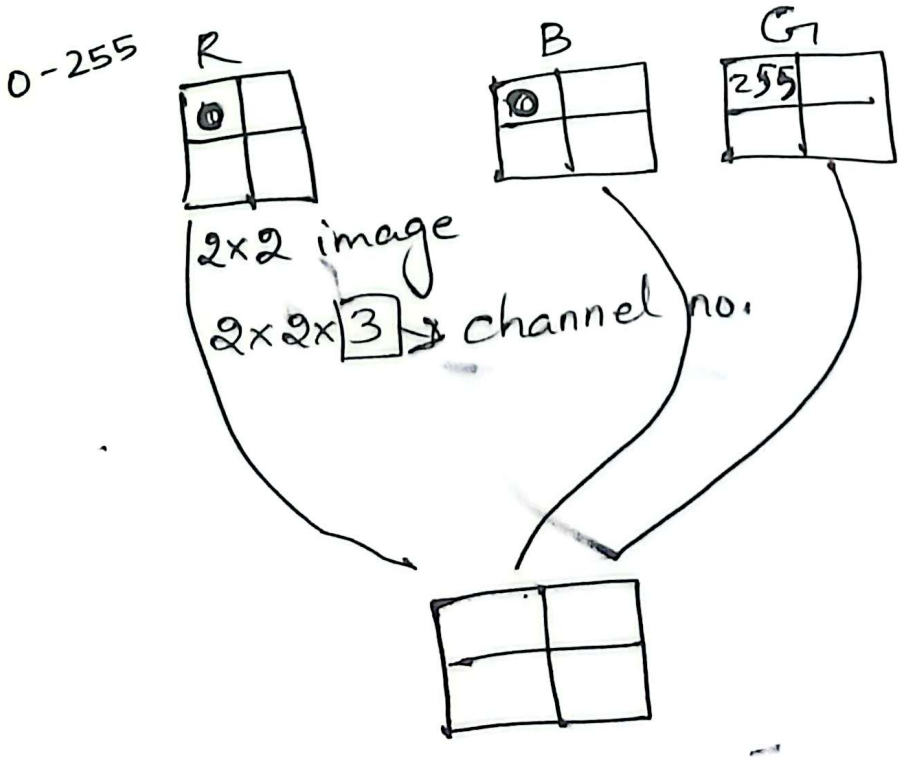
Pixel value 0,1 Binary

Grayscale image 0-255

Middle - Black - White



RGB (Primary color) R, G, B colors known as



Pixel would be green if G - 255

Binary image - 4 bit pixel image

Grayscale - 4 bit pixel

RGB - 12 bit

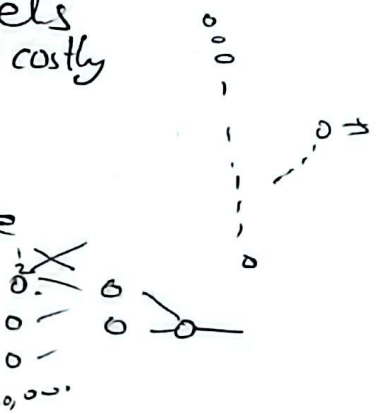
Image of size 200 × 200 × 3

↓ wide ↓ high ↓ color channels

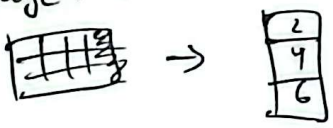
color image 1,29000 inputs আসবে computationally costly

Model এর parameter বেড়ে গেলে - overfit

Conv, pooling - picture টা down sample



layer 1o



Matrix টা কে flatten layer এ convert

Pg 17 * MLP vs ConvNet

Pg 18 * X or O classify করবে

pixel image

Image - binary White = 1 Black = -1

* What about trickier cases?

Pg 20 * How ConNet

Pixel to pixel compare

Not same as actual pixel

feature

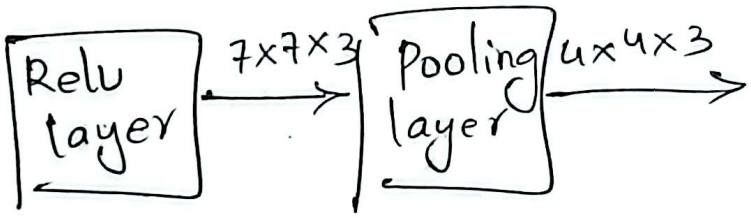
broadview তে compare

Pooling layer

- down sampling process

Pooling Filter size = 2×2 , stride = 2

- max pooling
- Average pooling



shape is changing

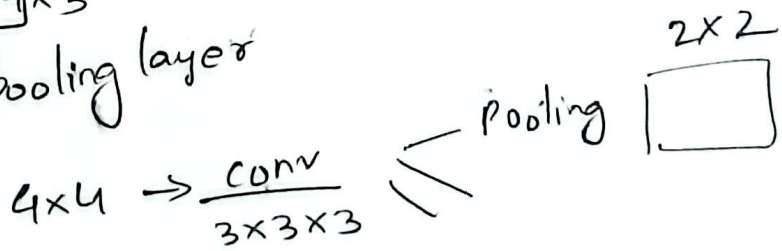
7×7×3

↓

ei value টি chang - convolution layer এর দ্বারে

4×4×3

↓ pooling layer



feature map বাড়তেছে

Flatten - 2D or 1D কে flat করে

* Convolution Layer - Filters

3 টা filter দিয়ে কাজ করবো। strides কতো এক ঘরে move করবো - spatial position

- feature map

3×3×64

3×3×1 → filter কয়টা আছে

H W

Padding - 0 or 1 → করছি

করিনাই

Mean value টা filter centre এ বসালে

- filter use করে যে নতুন image তৈরি হয়

তা হলো - that is feature map

Input size (w) : 9 means 9×9

Feature Map size = $1 + (W - F + 2P)/S$

Black color is -ve value

9×9 conv laye 3 → 7×7×3

3টা filter featuremap = 3

9×9 [conv layer-1 3×3×3] → 7×7×3 → [Relu layer-1] → 7×7×3

Relu to remove -ve value . and becomes 0

-ve

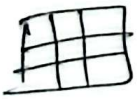
* Conv layers
  - dot product
  - Use ReLU layer
    Pixel value negative হয়ে যায়
    By using ReLU negative we can remove
  - Fully-connected

Pg 16
* Feature matching
Convolution region to region match করতে সাহায্য করে।

9 sept 2025

200 × 200 × 3
= 120000
100

point to point fit
model overfit



Conv layer
↓
image এর উপর convolution operation চলে
We can use RELU ~~activ~~ layer
~~layer~~

Pooling layer
Image ছোট হয়ে যায়
Pooling 1D layer এ convert হয়

* Hyperparameter (knobs)

- Translearning

Vgg16 architecture diagram

CNN - input image size (9x9) fixed

64-filter

4096 4096 4096

fc fc layer

2560

google CNN - model architecture - layer wise

cnn visualization - CNN explorerer

14 September 2025

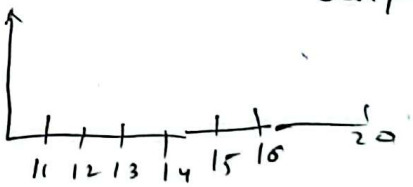
I am ____ ____

↓

next word prediction

sequential problem

depend on another output



Google - Understanding Recurrent Neural Network

I/p → Hidden layer → O/p

14 I/p → Hidden → output

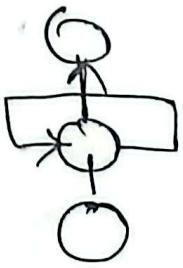
layer WH মাধ্যমে

output হিসেবে যায়

15 I/p → Wx → Hidden layer → O/p

↓

pass to next HL



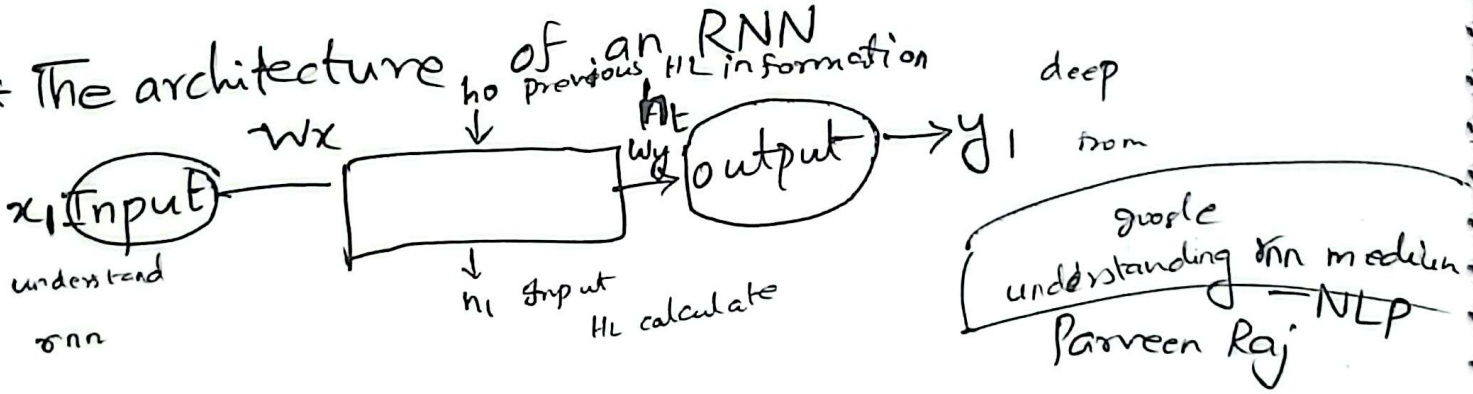
Normal Feed forward Neural Network

Can't memorize previous input

list of common applications —

- Speech Recognition
- Sentiment Classification
- Machine Translation (i.e chinese to English)
- Video Activity Recognition
- Name Entity Recognition (i.e Identifying names in sentences)

* The architecture of an RNN



value

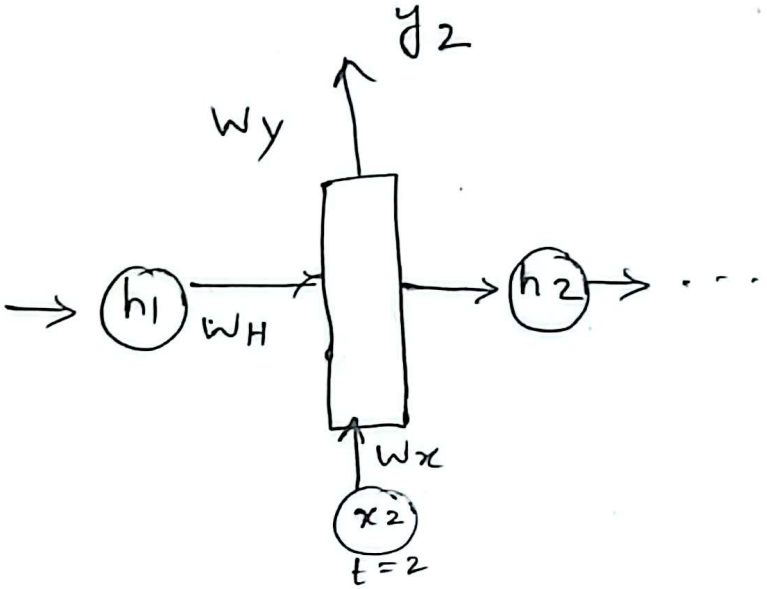
1) Categorical value → string type
2) numerical value

categorical को numerical value
convert को encoding करना।

$$d=\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \quad o=\begin{bmatrix}0\\1\\0\\0\end{bmatrix} \quad g=\begin{bmatrix}0\\0\\1\\0\end{bmatrix} \quad s=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Encoding of the word "dogs"

$X = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}_{4\times1}$



$h_1 W_H$

~~$x \cdot W_x + h_1 W_H$~~

$a_t = W_H h_{t-1} + W_x x_t$

$h_t = \tanh(a_t)$

3×1

$y_t = \text{Softmax} = (h_t \cdot W_y)$

4×1

W_x

0 — W_1
0 — W_2
0
0

$a_1 = W_H h_0 \quad W_x x_1$

( ) = 1 0 0 0

send to o/p layer

softmax / sigmoid

$3\times3 \quad 3\times1 \quad 4\times1$

$a_t = W_H h_{t-1} + W_x x_t$

3×4

classification problem

~~Update~~

16 September 2025

Backward Propagation

* BP for Multilayer (with HN)

$$\Delta w_{ij} = \eta \delta_j \times input$$

↓ gradient = 0 weight update করতে হবে না।

1) Vanishing gradient problem
2) Exploding gradient problem

Understanding recurrent Neural networks এটা দেখ

Google
Backward propagation

W_x, W_H, W_y

* Introduction to Long Short-Term memory (LSTM)

~~No memory cell~~

RNN কোন memory cell থাকে না
LSTM এ dedicated memory cell থাকে।

In LSTM 3 special gates
↓ To control

1) Input gate - যে info আসে এটা imp or not
memory তে রাখতে হবে কি হবেনা decide করে।

* Types of RNN architecture —

1) One-to-one — one I/p one o/p — classification
2) One to many — Image caption — 1 I/p many o/p
3) many to one — Many to one — (sentiment analysis)
4) Many to ~~Many one~~ many — sequence
5) ~~Man~~ — POS

Advantages & Disadvantages of RNN

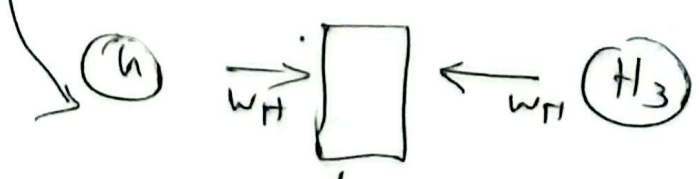
Adv
RNN are even used with convolutions to extend the effective pixel neighborhood.

Disadv
It cannot process very long sequence if using Tanh or Relu as an Activation function

* Applications of RNN
Machine Translation, Speech Recognition

* Variation of RNN
1) Bidirectional Neural Network
2) LSTM

CT
RNN Transformer model



Bidirectional NN

For multiple - categorical cross entropy
Classification

* RNN **Back propagation through time** Weight matrix 3টা
Wx, Wy, Wh

1) Initialize weights
2)
3) Compute the loss
4) Update weights based on gradients
5)

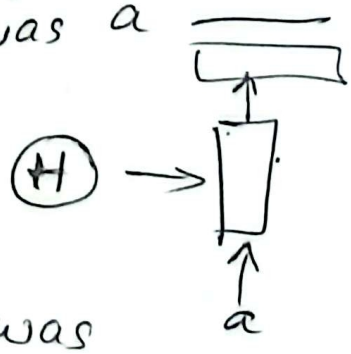
Previous loss + current loss
weight update

$Wx =$ [ ] 3×4

RNN problem

One major problem: Vanishing gradients

model কে আগের layer
predict করে
like a memory based



The brown and black dog, which was playing with the cat, was a german shepherd.

- Vanishing gradients make it difficult for the model to learn long-term dependencies.

2) Forget gate – decide memory থেকে info. আছে oita delete করতে হবে কি হবে না,

3) Output gate – কোন info. memory থেকে যা আছে current output/current prediction আসবে কি আসবে না।

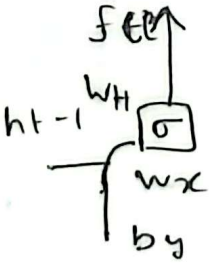
* What is the Vanishing gradient problem?

1) Cell state (Memory cell)
kind of as a conveyer belt,

C_{t-1} ⊗ ⊕ C_t ——→ C_{t+1}

2) Forget gate
decides what info. should be forgotten
It does a dot product h(t-1) and x(t)
sigmoid layer o/p a number b/w 0 and 1

0 → forget
1 → full



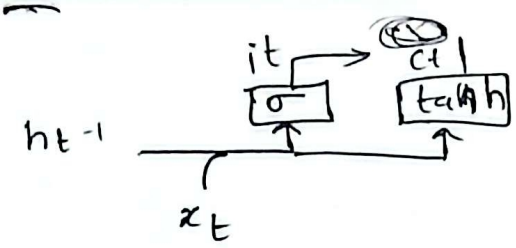
$f(t) = \sigma(W_x \cdot x_t + W_H \cdot h_{t-1}) + by$

3) Input gat – decides if that new info. is going to be stored in the cellstate.

3 parts –

(1) A Sigmoid layer decides the values to be updated. (কোথায় বসাবো)

(2) A tanh activation function layer
(কোন candidate বসাবে)



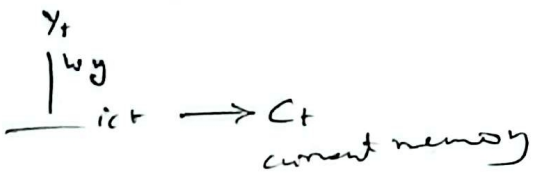
current I/p
hidden state info.
memory update

$$C_f = C_{t-1} \times f_t + i_t \times C_t$$

(4) Output gate

sigmoid layer

tanh layer - Memory থেকে value আছে।



LSTM হলো RNN এর variation. long dependance handle করে LSTM.

GRU

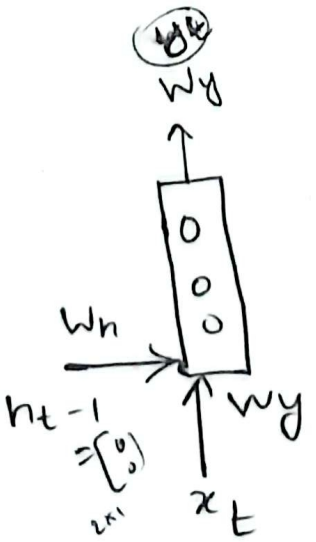
21 September 2025

$h_{t-1} = [\ ]$

$W_H = [\ ]$

$W_x = \begin{bmatrix} & \end{bmatrix}$

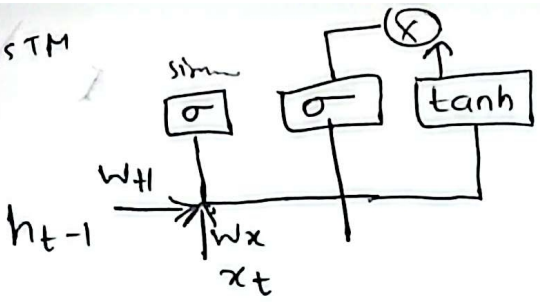
$x = \begin{bmatrix} & \end{bmatrix}$ $y_t =$

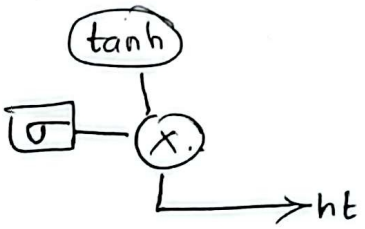


$h_1 = \tanh(W_x \times x_t + W_H h_{t-1})$

SimpleRNN (–)

LSTM





$\sigma(x_t + W_h$

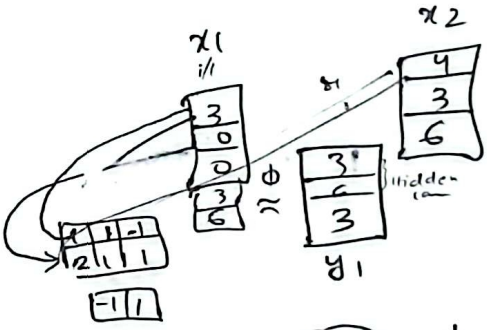
- RNN } i/p থেকে o/p বের করতে হবে (matrix multiplication)
  LSTM }

8.

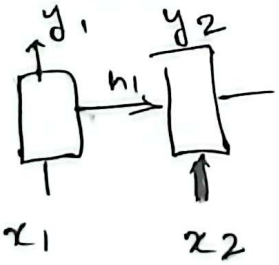
Google rnn by hand

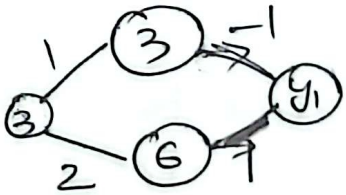
$x =$ | 3 | 4 | 5 | 6 |

$$\begin{bmatrix} 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

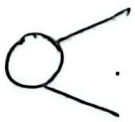


Relu





$-3+6$
$= 3$



$[3\ 6]\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

$$\begin{bmatrix} 4 \\ 8 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = 4+8 \quad -4+8$$

$$\begin{bmatrix} 4 & -8 \\ 4 & 8 \end{bmatrix}$$

32-5

$a_1 = W_x \cdot x_t + W_H h_{t-} + b_f$

$h = AF(a_1)$

$y_1 = AF(h_1 \cdot W_y)$

Input node

$d = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ 0 0 0 0

bias Hidden node & jetota bias othota

RNN

Node একটা নোডের

Input sequence x	x_1	x_2	x_3	x_4
3	4	5	6

(3)

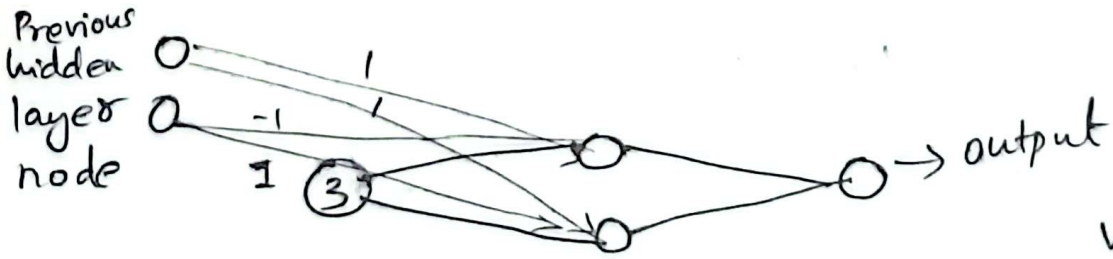
Parameter A $\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ B $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ C $[-1 \mid 1]$

A·F Φ : ReLU

Hidden states h_0 $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Output sequence Y | | | | |

W_H



$W_H = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

$W_x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $W_y = [-1 \mid 1]$

Current I/p
Previous

23 Sept 2025

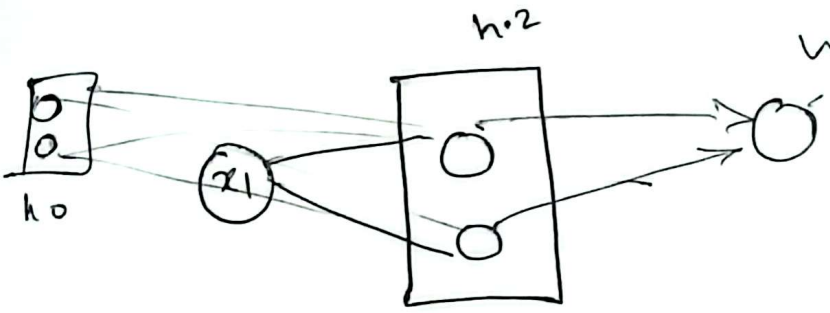
intro.

Google ~~introducti~~ medium

Understanding recurrent neural networks রকম n-i/p

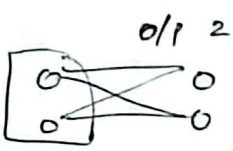
node কয়টা থাকে বলে দিতে হয়

h_0 = previous state এর hidden layer o/p



$W_y = [\ ]_{1\times 2}$

যদি 3 টা তাহলে $[\ ]_{1\times 3}$



$W_y = \begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix}$

$W_x = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$

$W_H = [\square\ \square]_{2\times 2}$

$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}_{2\times 1}$

Depends on hidden unit node

$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$

$W_x = [\ ]_{2\times 2}$

1×3 $\quad$ 3×3 $\quad$ 1×3

$x_t \cdot W_x + W_H \cdot h_{t-1}$

2
-1
0
1

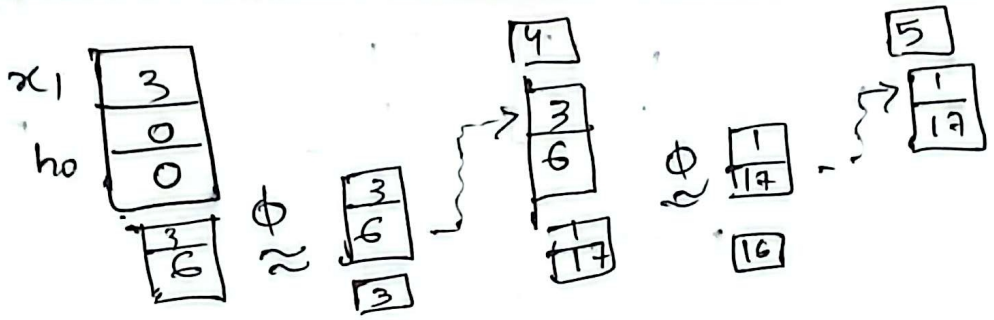
1	0
0	2
1	2

$$\begin{bmatrix} 2 \\ -1 \\ 0 \\ 1 \end{bmatrix}_{1\times4} \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}_{4\times3} \cdot x_1 \begin{bmatrix} 2 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}_{4\times3}$$

X $\quad$ W_t

Google medium LSTM

* Introduction to Long Short Term Memory (LSTM)
* Understanding Recurrent Neural network (RNN)



$$a_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} 3 + \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

$$h_1 = \underset{\text{Relu}}{\begin{bmatrix} 3 \\ 6 \end{bmatrix}} \times \overset{w_y}{\downarrow} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 6 \end{bmatrix} =$$

$$a_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} 4 + \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 4 \\ 8 \end{bmatrix} + \begin{bmatrix} -3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1 \\ 17 \end{bmatrix}$$

$$h_1 = \begin{bmatrix} -1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 17 \end{bmatrix}$$

$$= 16$$

$$a_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \times 5 + \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 17 \end{bmatrix}$$

$$= \begin{bmatrix} 5 \\ 10 \end{bmatrix} + \begin{bmatrix} 1 & -17 \\ 1 & 17 \end{bmatrix}$$

$$= \begin{bmatrix} 5 \\ 10 \end{bmatrix} + \begin{bmatrix} -16 \\ 18 \end{bmatrix}$$

$$= \begin{bmatrix} -11 \\ 28 \end{bmatrix}$$

using relu A.F.

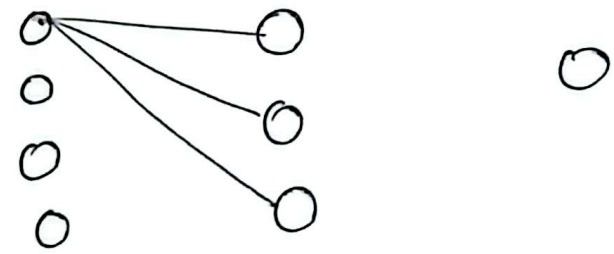
$$h_1 = \begin{bmatrix} -11 \\ 28 \end{bmatrix} = \begin{bmatrix} 0 \\ 28 \end{bmatrix}$$

$$y_1 = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 28 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 28 \end{bmatrix}$$

$$= 28$$

xcell



$W_X = [3 \times 4]$

$W_H = [3 \times 3]$

$W_Y = [1 \times 3]$

$$a_1 = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

tanh A·F
-1 to 1

Q) W_X, W_Y, W_H value দিবে

~~y_3~~ y_3 value find করতে হবে

$W_Y = \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix}$

Transpose করে multiply করতে হবে।

gate
5 টা layer
।

Q) input source — column সংখ্যা
" Destination — ROW সংখ্যা

$W_Y = \text{shape} \begin{bmatrix} w_{y1} \\ w_{y2} \\ w_{y3} \end{bmatrix}$ · এখানে 3 টা row

transpose করতে হবে $W_Y = [w_{y1} \quad w_{y2} \quad w_{y3}]$

Transformer model – RNN এর variation
(Combination of two transforma)

LSTM- অতিরিক্ত / অনেক বেশি sequence মনে রাখতে পারে না।
এর সমাধান একটা model আছে which is transformer model.

Sequence to sequence model (seq 2 seq)

The bank near the street and the bank of the river.
It is not mentioned like which bank
সবাইকে same ভাবে priority দেয়।

Attention concept – concept came from human.
We hear others people's voice less than our own friend's voice

Model attention use করলে ভালো হবে।

Jay Alammar

Eg:- Neural Machine Translation

seq 2 seq models with attention

seq 2 seq → Two types i/p sequence encoder হিসেবে ca calculation
Encoder → Process করে context vector
Decoder ← generate করে
Output sequence generate করে

→ Encoder & Decoder both are RNN in translation model.

RNN bas

RNN weight matrix

Wx, Wy, Wh

বিভিন্ন gate

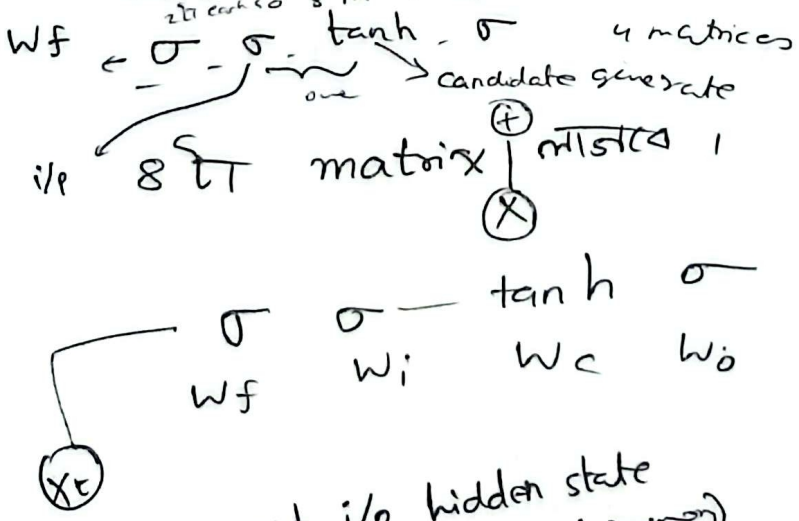
Wf forget gate matrix
Wc input gate candid mat
Wo o/p matrix
Yellow Wx Green Wh
Total matrix = 2

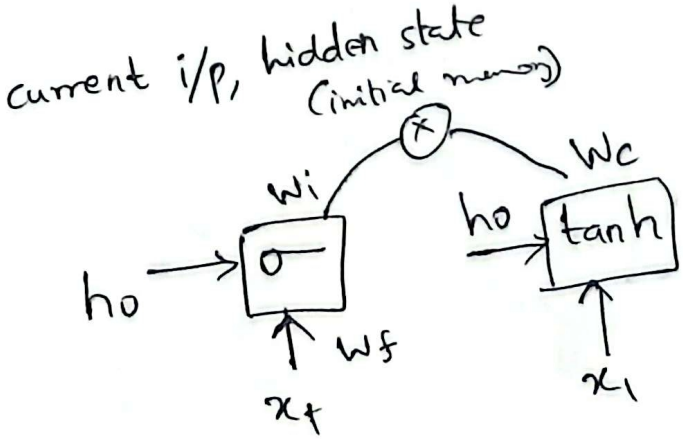
RNN math বুঝে আসবে

7 October 2025

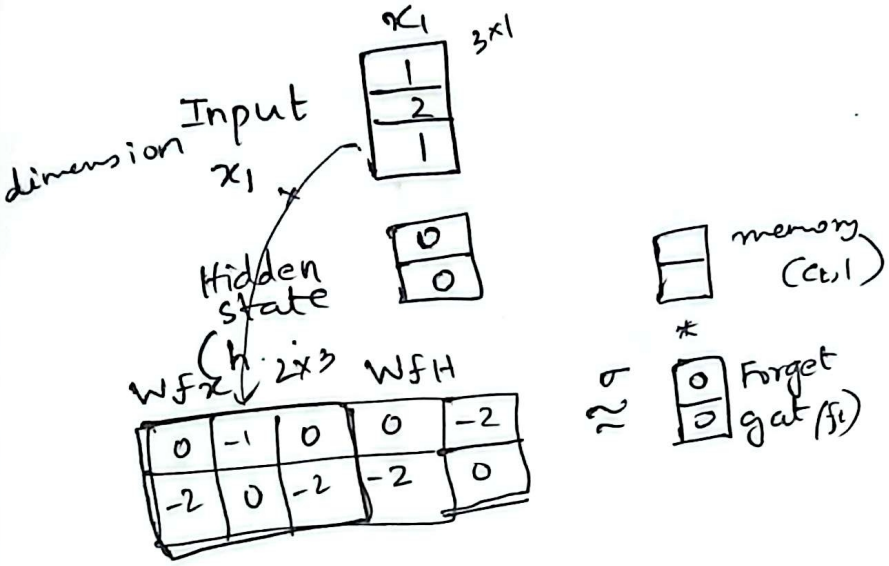
LSTM

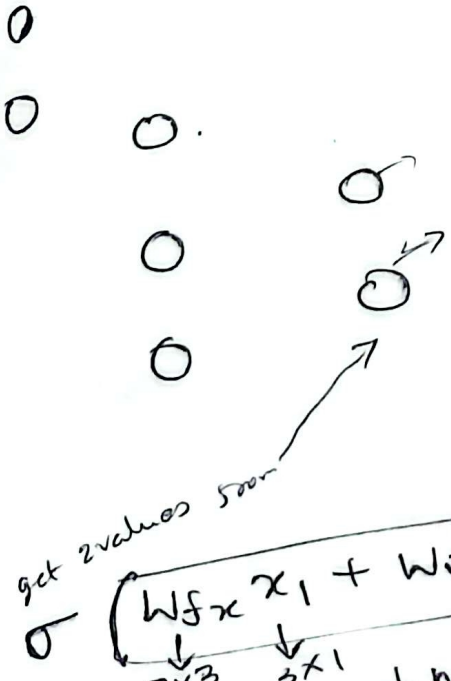
understanding rnn medium
can you calculate rnn by hand
Introduction to lstm medium





LSTM





$(2\times3)\,(3\times1)$
$= (2,1)$

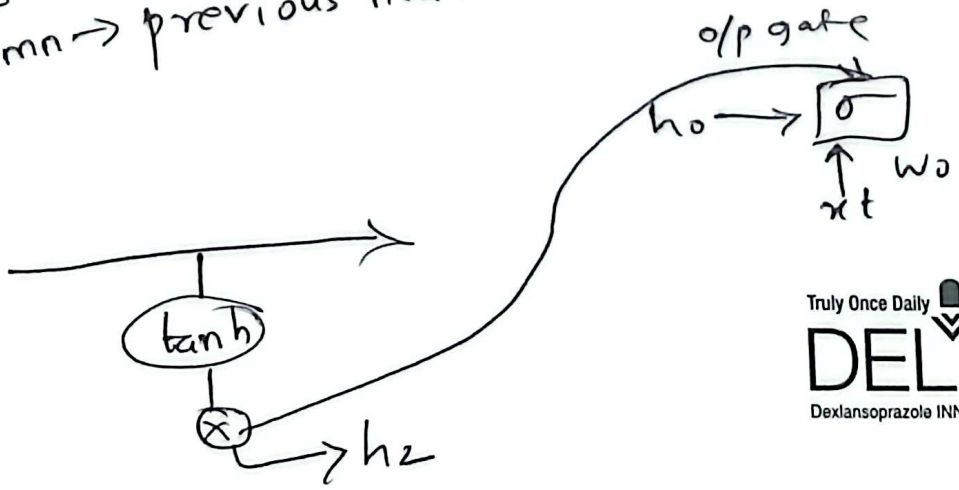
get 2 values soon

$\sigma\left(W_{fx}\,x_1 + W_{fh}\,h_0\right)$

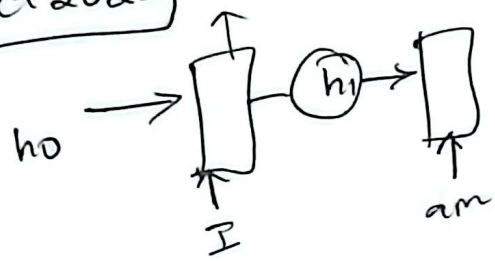
2×3 $\quad$ 3×1

row → next node

column → previous node



8 Oct 2025



sequence to sequence model

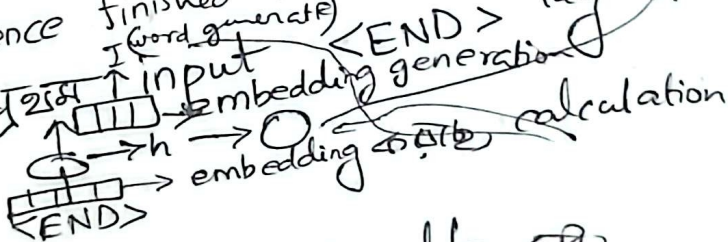
Jay Alammar

Visualizing a Neural Machine Translation Model
(Mechanics of seq2seq Models with Attention)

To transform word into vector we turn to the class of methods called word embedding.

After the sentence finished there will be end tag.

Decoder



+ ho

* Attention module
weight multiply handle
Extra attention calculate - extra attention

Attention Mechanism

12 Oct 2025

~~Self-Calculat~~

Self-Attention -

~~Decoder~~ Encoder-Decoder Attention

Encoder থেকে ~~যে~~ context vector টা আসে

~~Fuzzy logic~~ Neural Networks

Human brain, perceptron, Mac Culloch's pit, Single neuron, Hidden layer with proper A.F, Universal ~~activation~~ Approximation Theorem, A.F (Types), Forward Propagation, Evaluation Matrix, Precision, Recall, Backpropagation (Gradient descent, Cost Function, Classification (cost), Regression, MSE, MAE), Early stopping, underfitting, overfitting, local minima, CNN (Full connected layer, Pooling layer, Relu, filter, Convolutional operation, padding, pooling method, strides, Filter size, feature map) RNN (model, Time by Time change, Hidden info. one to another layer pass, RNN math), LSTM (dedicated memory line), forget gate, input gate o/p, memory ~~গেট~~, output gate long dependency problem solve, seq2seq model, Transformer model Basics)

Fuzzy logic

Introduction to Fuzzy logic systems

Human মতো বানানো.

Binary logic

Thinking capability and behavior like human.

Uncertainty, partial truth

* Introduction

Our model ..... vague information.

Fuzzy logic value – 0 থেকে 1 any value নেওয়া যাবে।

* Crisp sets and logic

} → crisp set / binary set

Tall = {

Medium =

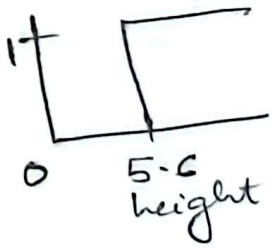
Short = {

}

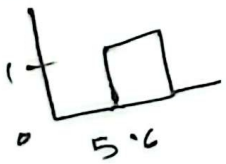
Threshold set 5·6

5·59

$$I_A(x) = \begin{cases} 1 & ; 4 \le x \le 1 \\ 0 & ; \text{otherwise} \end{cases}$$



5·6 – 5·9



short graph

* Fuzzy sets

Member ship function

Tall = { 1 0· }

Medium = { 0 1 }

short = { 0 0 }

Tall = { 0 0

Medium = 0·8 0·2

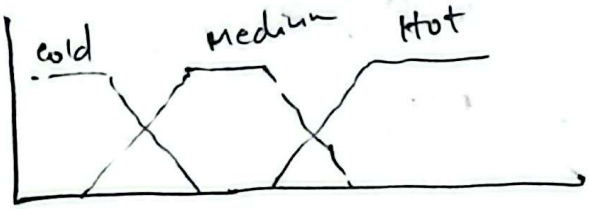
Short = { 0·2 0·8

* Basic configuration

1) Fuzzification
2) Fuzzy Inference Engine/system
3)
4)

* Design of Fuzzy system

AI integrated fan-speed



A total of nine rules are used to describe the knowledge necessary to operate our fan:

1) Fuzzification – input হিসেবে যে crisp set এ আসে Fuzzy membership এ convert

2) Fuzzy inference Engine

With this input ২ rules are fired with a degree higher than zero

কোন rule গুলো fire করে

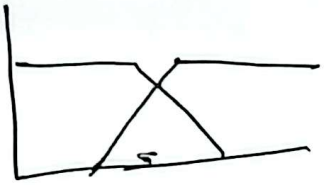
Fan speed graph

* Mamdani Algorithm → ~~Fuzzy হিসেবে crisp value~~

Mamdani and sugeno ৪ inference systems

gives min. value

Triangle Membership function – Increasing and decreasing of tempt.



Trapezoidal Membership function

* Fuzzy operation

Union এ max value নিবো

Fuzzy set $A = \{ 0/0, 0/1, 0/2, 0.5/3, 1/4, 0.5/5, \ldots \}$

$B = \{ \ldots 0/4, 0.5/5, 1/6, \ldots \}$

From A & B (Max value)

$A \cup B = \{ 0.5/3, 1/4, \ldots \}$

min value — $A \cap B = \{ 0/3, 0.5/5, \ldots \}$

Complement 2লে 1 থেকে বাদ দিতে হয়।

slide

* Union – Max value নিতে হয়।
* Intersection
* Complement

$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

google * Fuzzy Inference Systems

If, then logic কাজ করে

nducted by Lotfi Zadeh (1921-2017) in 1965.
like crisp set ...
he membership function ..... degree of belonging.

Oct 2025

A very brief introduction to Fuzzy logic

Intro to fuzzy logic & fuzzy systems

Membership value – 0 or 1

Crisp set and logic

Uncertainty express

0 থেকে 12 A∪B
4 থেকে 12 intersect

$12 \leq x \leq 4$

4থেকে 12 2টুকু 1 বাকি 0

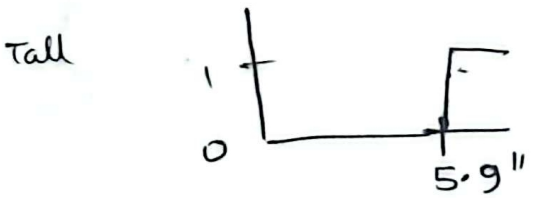
5.6 ft

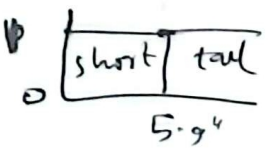
* Fuzzy sets

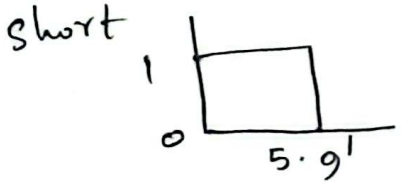
value from 0 to 1

tall = { 0.8

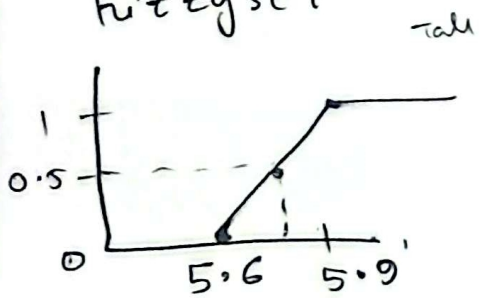
short = {0.2

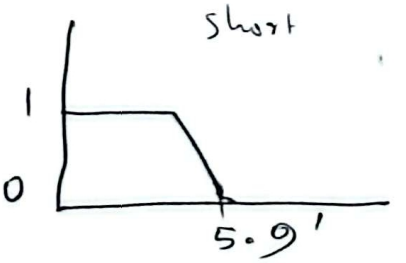


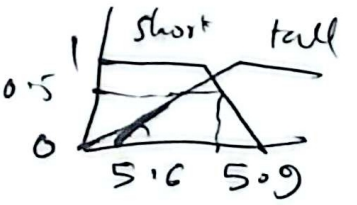




Fuzzy set

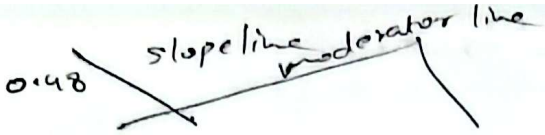






tall = { 0.5 }

short = {0.5 }



Mamdani inference system এ min 0.29

slow = 0.48
Moderate = 0.29
↓
crisp এ convert

Centre of gravity – আমাদের centre of gravity point কতো
Defuzzification – Fuzzy থেকে crisp conversion.

slide * Properties of Fuzzy sets

1) Report submit both online & offline – 10
   Hardcopy print & submit to sir
   Plagarism less than 30%
   AI less than 20%
   ~~or~~ Latex format ~~for~~ format given
   min 20 pg
   * No code in methodology

Github
- ipyb – python এর notebook file
model h5 [□] model এর ভেতরে o/p দিবে

2) Github link – code, Model tester video – 10

27 oct submission dat = 11.59

PUC CGIP projects — Youtube
PUC IP projects
OpenGL C++
3) Youtube video submission + viva
10

Padding:

Padding means adding extra rows and columns (usually filled with zeros) around the edges of an input image before applying the convolution operation.

1) Validation data
Used to tune hyperparameters and check performance during training.
Model evaluation during training.
2) Validation data helps to avoid overfitting (when the model memorizes training data).

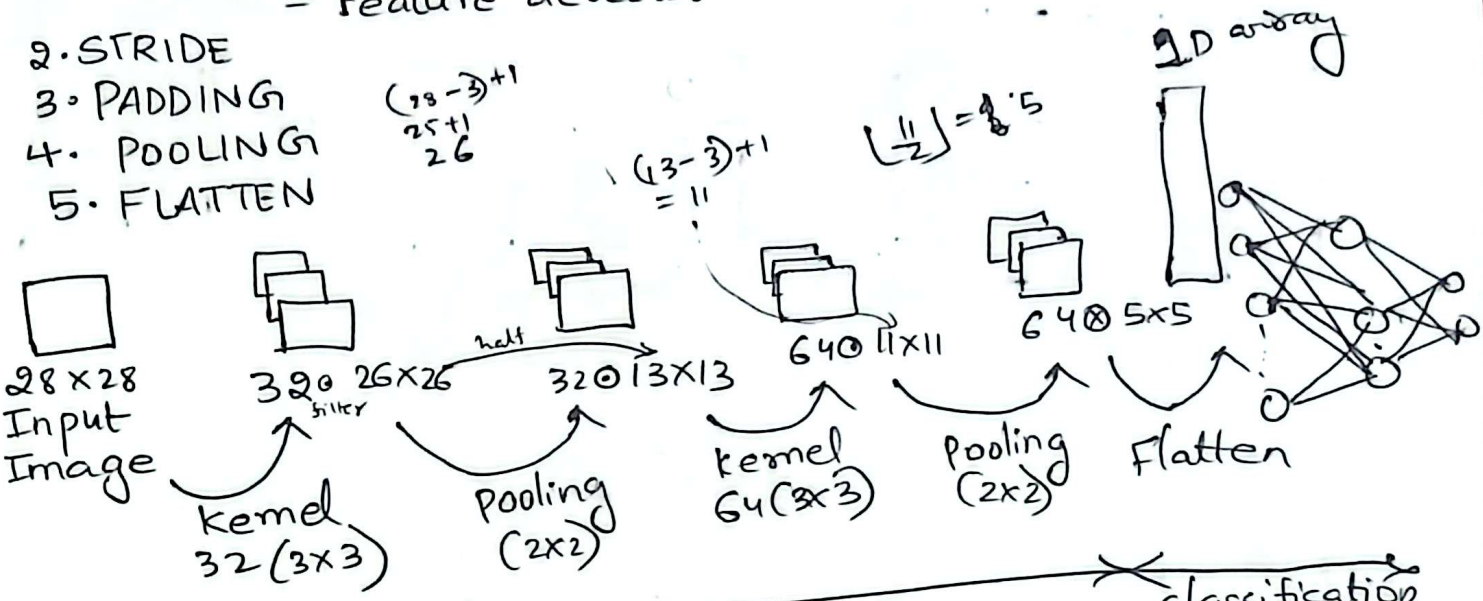
2) Training data — helps the model learn.
Model training phase.

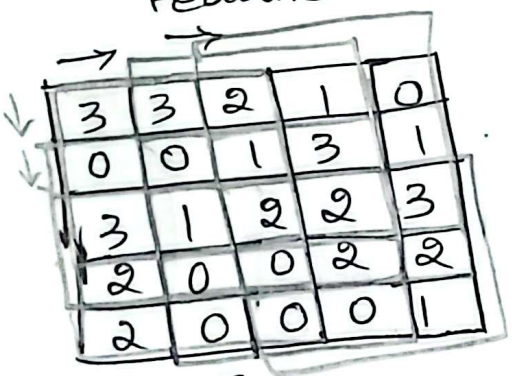
3) Testing data —
Measures generalization ability — how well the model works on new, unseen data.
verbose = 0, shows loss of every epoch

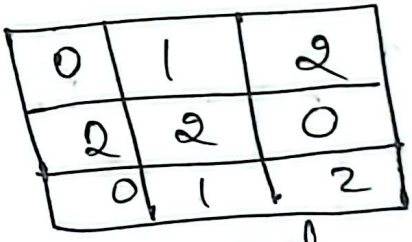
Convolutional works well on grid-type data.

1. KERNEL = Filters
= Feature detector

2. STRIDE
3. PADDING
4. POOLING
5. FLATTEN

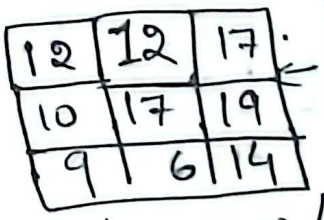






$3\times0+3\times1+2\times2 + 0\times2+0\times2+1\times0 + 3\times0+1\times1+2\times2$

$0+3+4 +0+0+0 +0+1+4$



Output = [i − k] + 1
= [5 − 3] + 1 = 3

Output Dimension = $\left\lfloor \frac{\text{Input Dimension} - \text{kernel size} + 2 \times \text{Padding}}{\text{stride}} \right\rfloor + 1$

Dimensions after each operation:

1) After 3x3 convolution : 10x10x40
2) After ReLU Activation : 10x10x40
3) After 3x3 Max Pooling : 10x10x40
4) After 3x3 convolution : 10x10x20
5) After ReLU Activation : 10x10x20
6) After 2x2 Max Pooling : 6x6x20

1) 3x3 convolution (40 channels) with stride 1 and padding 1

kernel size = 3x3 stride: 1 Padding: 1

Input dimensions: 10x10x10

Output Width = $\left\lfloor \frac{10-3+2\times1}{1} \right\rfloor + 1 = 10$

Output Height = $\left\lfloor \frac{10-3+2\times1}{1} \right\rfloor + 1 = 10$

Output channels = 40

Dimension = 10x10x40

2) ReLU Activation

- Does not change dimensions
- Output Dimensions : 10x10x40

3) 3x3 Max Pooling with stride 1 and padding 1

kernel size = 3x3
stride = 1
Padding = 1

Output width = $\left\lfloor \frac{10-3+2\times1}{1} \right\rfloor + 1 = 10$

Output height = $\left\lfloor \frac{10-3+2\times1}{1} \right\rfloor + 1 = 10$

output channels = 40 (channel dimension remains unchanged in pooling)

Dimension = 10x10x40

Pooling – Max Pooling (2×2)

Reduce dimension

6	14	17	11	3
14	12	12	17	11
8	10	17	19	13
11	9	6	14	12
6	4	4	6	4

5×5

14	17
11	19

$6+14+14+12 = \frac{46}{4}$

Avg pooling

11.5	14.25
9.5	14.0

$\frac{5}{2} = \lfloor 2.5 \rfloor = 2$

CNN calculating Parameters (Output Dimensions)

Q) Consider the input tensor of dimensions 10×10×10 where the last dimension is the channel dimension. The following operations are applied:

1. 3×3 convolution (40 channels) with stride 1 and padding 1 for each dimension.
2. ReLU activation
3. 3×3 max pooling with stride 1 and padding 1 for each dimension.
4. 3×3 convolution (20 channels) with stride 1 and padding 1 for each dimension.
5. ReLU activation.
6. 2×2 max pooling with stride 2 and padding 1, for each dimension.

Stride – move two blocks

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

5 × 5
Input

0	1	2
2	2	0
0	1	2

kernel
3×3

Feature map

12	12	7
10	17	9
9	6	14

$0+3+4+0+0+0+0+1+4$
$0+1+0+2+6+0+0+2+6$

12	7
9	14

2×2

$$0 = \left[\frac{i-k}{s}\right] + 1$$

$$= \left[\frac{5-3}{2}\right] + 1 = 2$$

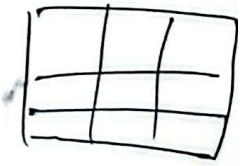
Padding (Border problem solver)

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

stride

0	1	2
2	2	0
0	1	2

3 × 3



3×3
5×5

$$0 = \left[\frac{i-k+2p}{s}\right] + 1 = \left[\frac{5-3+2\times1}{1}\right] + 1 = \left[\frac{5-3+2}{1}\right] + 1$$

$$= 5 \times 5$$

6	14	17	11	3
14	12	12	17	11
8	10	17	19	13
11	9	6	14	12
6	4	4	6	4

5×5

4) 3×3 Convolution (20 channels) with stride 1 and padding 1

Kernel size = 3×3 Padding = 1
stride = 1 Input dimension = 10×10×40

Output width = $\left\lfloor \frac{10-3+2\times1}{1} \right\rfloor + 1 = 10$

Output height = $\left\lfloor \frac{10-3+2\times1}{1} \right\rfloor + 1 = 10$

Output channels = 20

Dimension = 10×10×20

5) ReLU Activation
- Does not change dimension
- Output Dimension = 10×10×20

6) 2×2 Max pooling with stride 2 and padding 1

Kernel size = 2×2 Padding = 1
stride = 2

Output width = $\left\lfloor \frac{10-2+2\times1}{2} \right\rfloor + 1 = 6$

Output height = $\left\lfloor \frac{10-2+2\times1}{2} \right\rfloor + 1 = 6$

output channels = 20 (channel dimension remain unchanged in pooling)

Dimension = 6×6×20

Parameter Count

1) Number of Parameters = (kernel height × kernel width × Input channels + 1) × Output channels

First 3×3 convolution (40 channels):

kernel height = 3 Input channels: 10 (initial input channels)

kernel width = 3 Output channels = 40

Parameters = $(3 \times 3 \times 10 + 1) \times 40 = (90+1) \times 40 = 3640$

4) second 3×3 convolution (20 channels):

The input to this layer is the output of the first max pooling layer, which maintains 40 channels.

Output channels = 20

Parameter = $(3 \times 3 \times 40 + 1) \times 20 = (360+1) \times 20$

$= 7220$

Total Parameters

Total number of parameters in the CNN:

Total parameters = 3640 (First conv) + 7220 (2nd conv)

$= 10860$.

Thus, the CNN has a total of 10,860 parameters, solely from the convolutional layers as pooling layers and ReLU activations do not add any learnable parameters.

CNN

Convolution layer → Pooling layer → Fully connected layer

↳ down sampling (to extract feature)

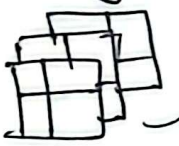
↳ Pixel Value

→ Flatten layer (convert matrix to vector)

→ Binary image (0 or 1)

→ Grey scale image – (0-255)
↳ Pixel value

→ RGB image / color image
(3 channel)



R-G-B
↳ 3 channel (2×2×3)

Feature mapping → Region wise matching

⇒ Will be computational costly if convolutional layer, pooling layer are not used.

eg: 120,000 —down sampling→ 100 pixels

→ ReLU (introduce non-linearity by converting negative values to 0).

Convolutional layer → Pooling layer → Flatten layer (convert 2D matrix to 1D vector)

$$\frac{32-6+2\times0}{1}+1 = 28\times28\times6$$

$$\frac{28-2+2\times0}{2}+1 = 14\times14\times6$$

$$\frac{14-5+2\times0}{1}+1 = 10\times10\times16$$

$$\frac{10-2+2\times0}{2}+1 = 5\times5\times16$$

$$\frac{5-5+2\times0}{2}+1 = 1\times1\times120$$

21 October 2025

* Membership function
* various type of Membership function
* Fuzzy set represention
* Fuzzy Representation
* Empty Fuzzy set
* Fuzzy operations
* Union - max value
* Intersection - min value
* Complement $1-\mu_A(x) = \mu_{\neg A}(x)$
* Multiplying a Fuzzy set by a crisp number
* properties of Fuzzy sets (Just theory)
* Fuzzy operations example (exam like this Q)

$\mu_a(x) = \frac{x}{x+1}$ $\quad x=[0,5]$

$\mu_b(x) = 4-(x+1)$

$0 \cdot \frac{0}{0+1} = 0$

$\frac{1}{1+1} =$

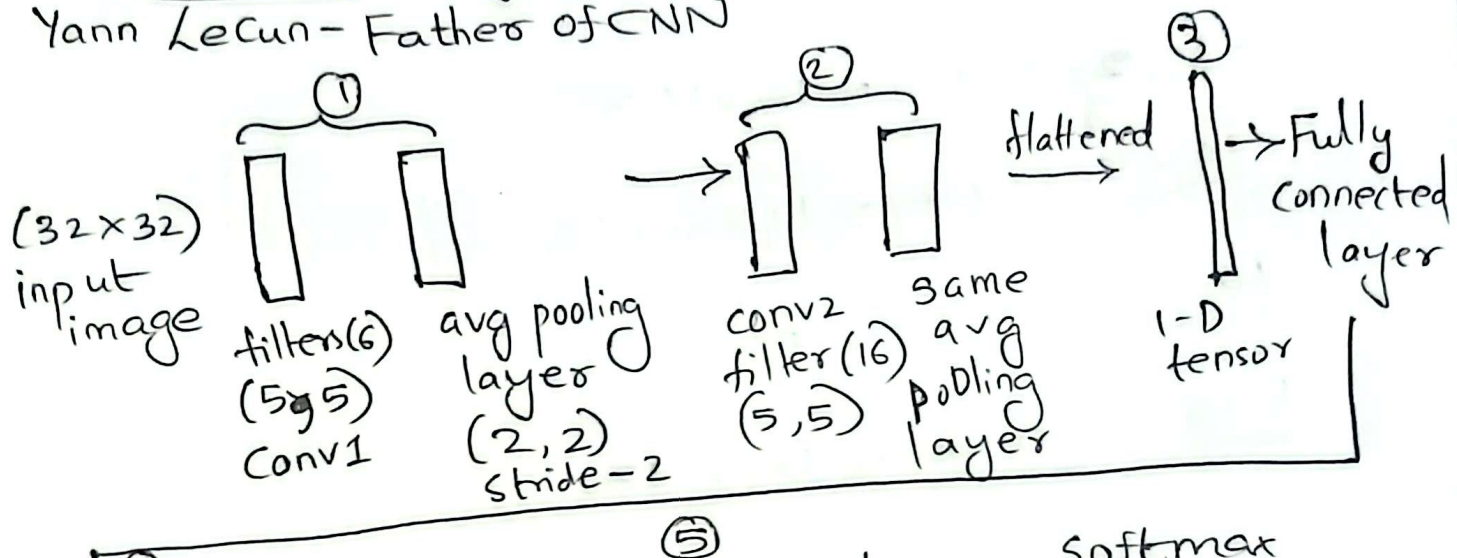
$B = \{3/0, 2/1,$

$= 4-0+1$

$A = \{0/0, 0.5/1, 0.67/2 \ldots\}$

LeNet: (LeNet-5 layers)

Yann LeCun - Father of CNN



$(32\times32) \rightarrow (28,28,6) - (14,14,6) \rightarrow (10,10,16) - (5,5,16)$

(32×32) → Pixel image

→ (25×16) 400 flattenlayer → (400×120) (120×84) (84×10) → 10 digits

When CNN increases, filter size increases

LeNet – activation function (tanh)

$\frac{W-F+2P}{S}+1$

$\frac{5-6}{1}+1$

Parameter calculation:

conv 1: $(Kw \times kh \times \text{Input chan} + 1) \times \text{output channel}$

$(5\times5\times3+1)\times6 = 456$

conv 2: $(5\times5\times3+1)\times16 = 2416$

$(5\times5\times16+1)\times120 = 48120$

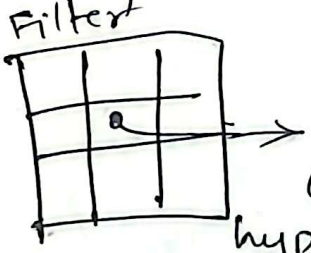
$(120\times84+84) = 10,164$

$(84\times10+10) = 850$

Strides- strides defines how far the filter moves across the image during convolution.

Feature map -

Feature map represents the activation produced by applying a specific filter to input data.



3×3×1 → filter value is 1

center by convertion, hyperparameter can be changed.

⊗ sign for convolution operation

Feature map size = $\frac{W-F+2P}{S}+1$

F = Filter size
P = padding
W → Picture size
S → stride

= Relu layer

- Pooling layer → Pools out important pixels (also works as filter) → Filters does not have any value

↳ Max-pooling
↳ Average pooling } Reduces size

CNN - Uses transfer learning

If I want to use transfer learning, image size has to be same (disadvantage)

- approach
→ LeNet
→ AlexNet
→ ZF Net
→ VGG
→ Google Net
→ ResNet