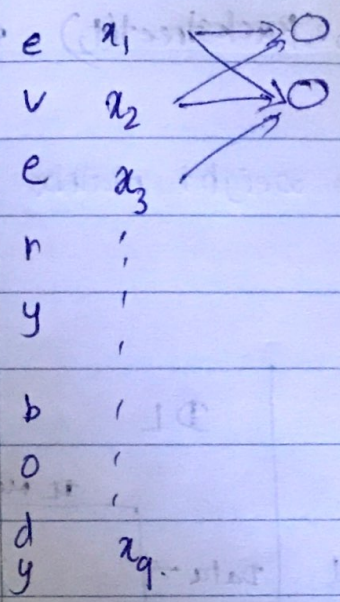
Date: No:

Unit 4 : RNN - Recurrent NN

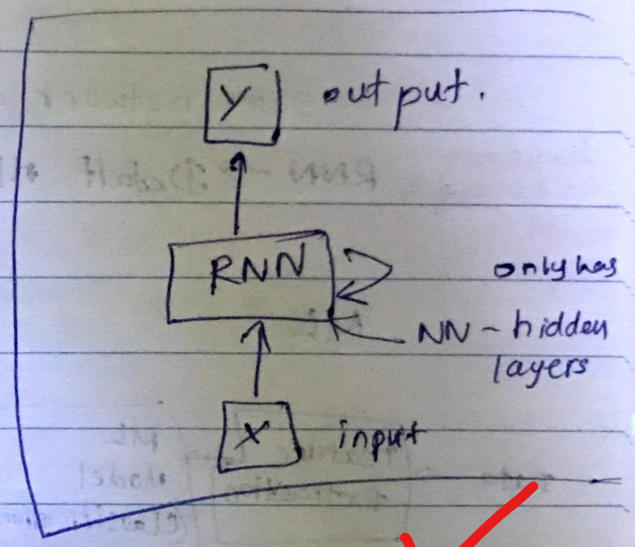
Sequential data vs non-sequential dat.
(length is fixed)

image → S
→ NS.

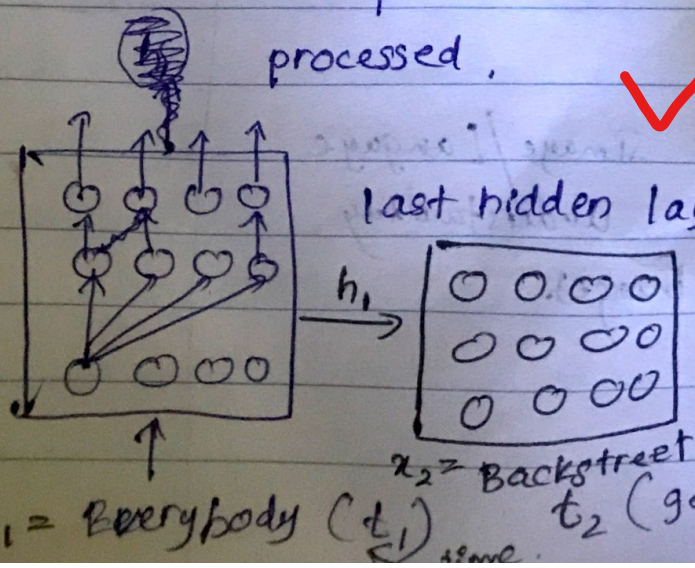
video → S.



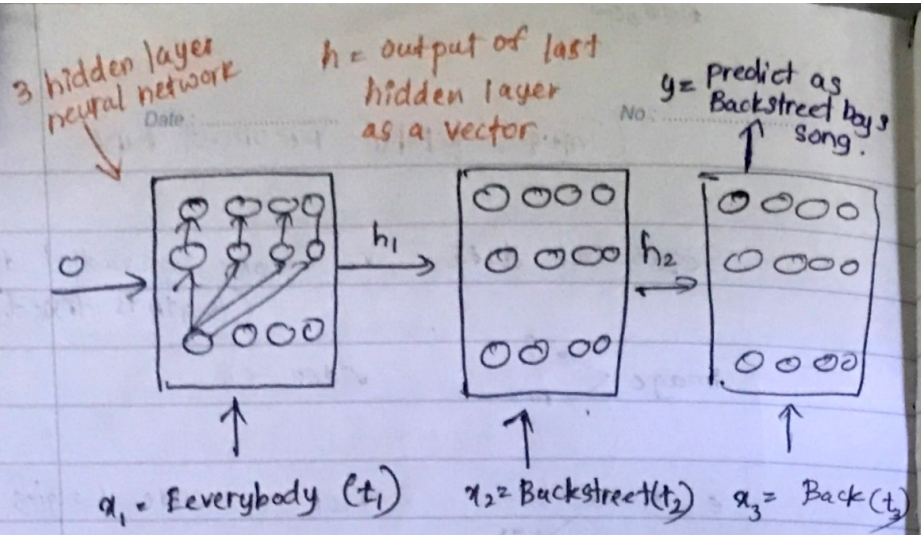
MLP → slide changing need.



Key idea : RNNs have an "Initial state" that is updated as a sequence is processed.



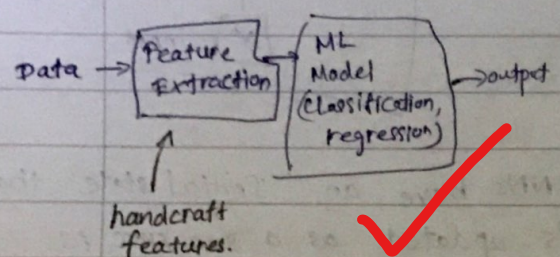
last hidden layer (vector output)
state (h)



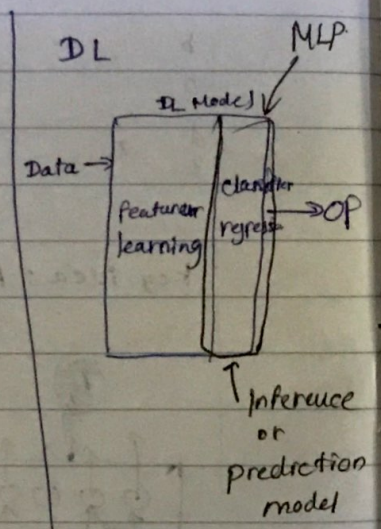
same network = Same weight matrix

RNN → Default MLP

ML

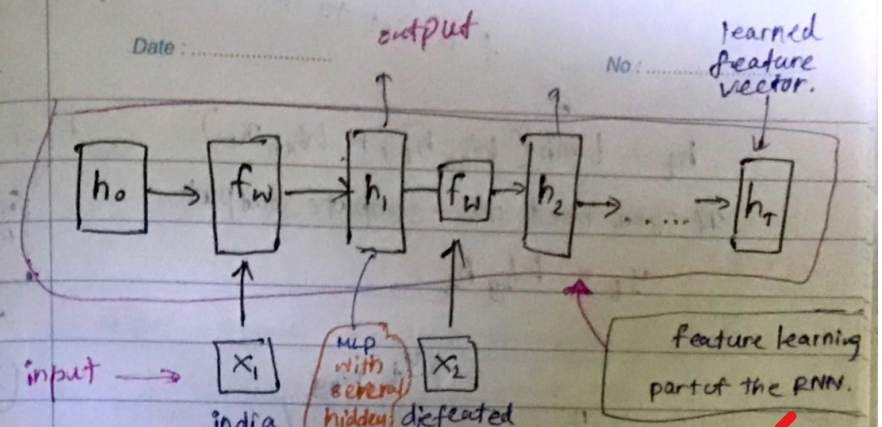


DL

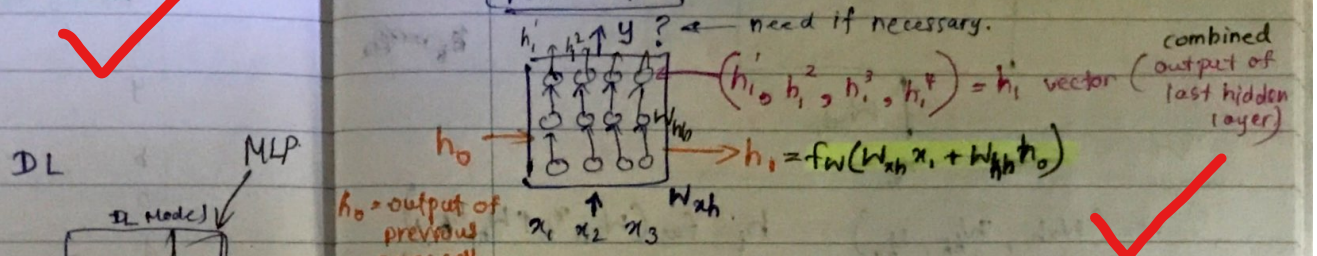


RNN → Image/Language understanding

CNN → Images.



RNN cell



W_{xh} = Weight Connect inputs to the first hidden layer.

W_{hh} = weighs of all hidden layers.

$W = (W_{xh}, W_{hh})$

f_W = activation with weight matrix W

* (Vanilla) RNN cell

$$h_t = f_W(h_{t-1}, x_t)$$

New state — Some function with parameter W — old state — input

Scanned with CamScanner

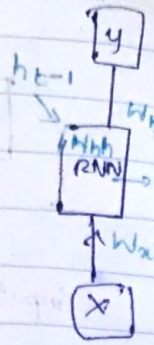
Date : No :

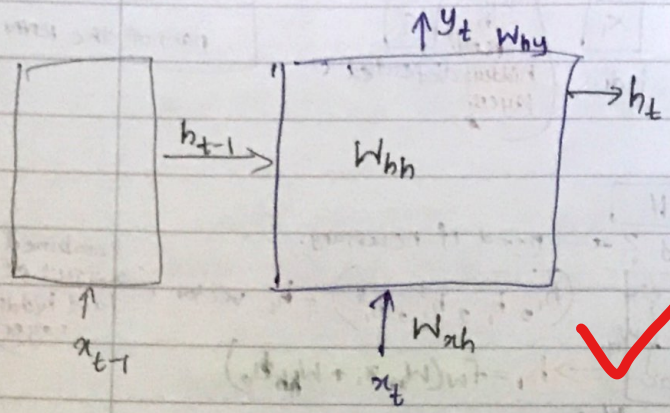
f is set to tanh.

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

If you need to compute output

$$y_t = W_{hy} h_t$$





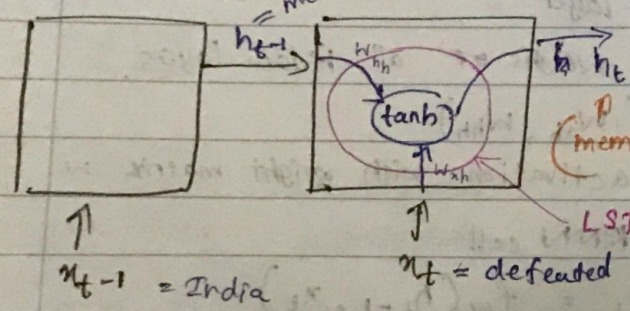
$$W = (W_{xh}, W_{hh}, W_{hy})$$

s u v

$$h_t = f_W(W_{xh} x_t + W_{hh} h_{t-1})$$

$$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1})$$

Vanilla RNN cell



h = state.

$$x = (x_1, x_2, \ldots x_{t-1}, x_t)$$

(memory of "India defeated")

LSTM (Long, short term Memory)

Vanilla RNN cell ⇒ BPA to learn weights

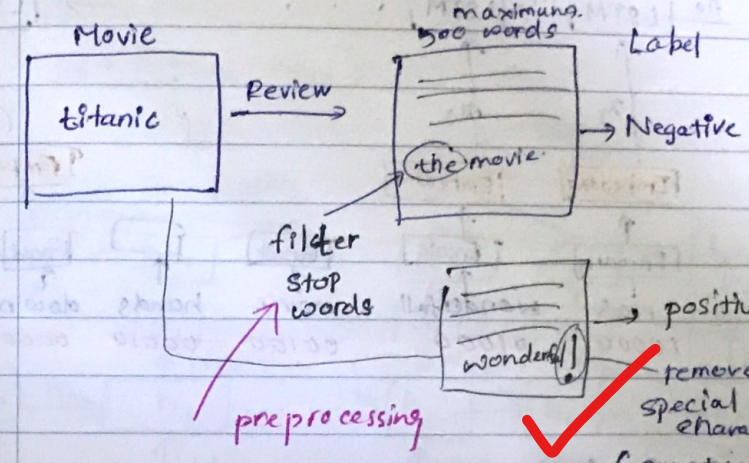
⇒ We might get vanishing gradient problem.

Date : No :

BPTT → BPA with time.

To solve vanishing gradient problem ⇒ Need LSTM

IMDM Movie Review Example



RNN Structure (Architecture)

one-to-many

many-to-one

many-to-many. < Encoder decoder Architecture / Full

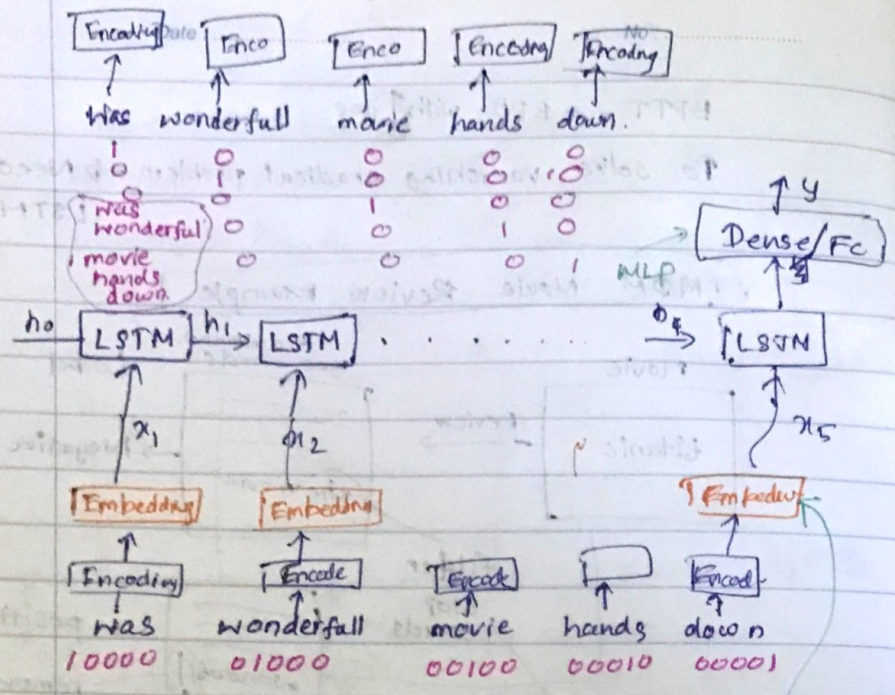
Movie review classification → many-to-one

Many words

one label

Scanned with CamScanner

One-hot-Encoding

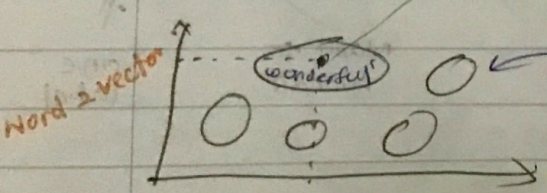


* In text classification
  1) First Encode
  2) Then create vector

Change this
- Word2 vect
- BOW
- TF-IDF

Word Embedding

Wonderfull → 134678 →



word cluster

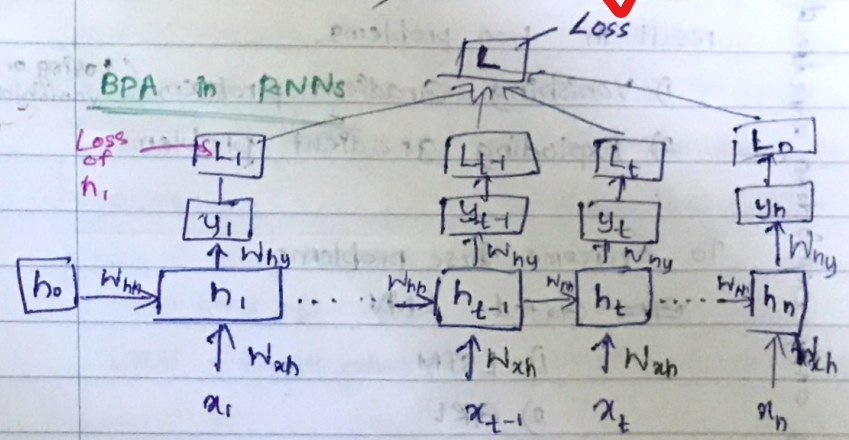
- each word place in a 2D plane
- Each word (point) has location
- This is embedding

normally $\frac{\partial L}{\partial W_y}$

* Training RNNs with Backpropagation with time (BPTT) → use old backpropagation but slite twist

* RNN computational Graph - One to Many Applications.

$W = (W_{xh}, W_{hh})$



$L = \sum_{i=1}^{n} L_i$

we need to update the weight of W_{xh}, W_{hh}, W_{hy}

Updating weights

- Updating each weight of W_{xh}, W_{hh}, W_{hy}

Matrix-Wise (In practice we need to this for each individual weights)

$$\frac{\partial L}{\partial W_{hh}} = \frac{\partial \left(\sum_{i=1}^{n} L_i\right)}{\partial W_{hh}}$$

weight as matrix

→ if we apply chain rule →

Date : ..... No : .....

There is a partial derivative

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}$$ ⟸ Multiplication of small gradient values of a long sequence.

Multiplication

In a vanilla RNN cell

This long gradeint sequences (multiplications) could result in two problems.

1) Vanishing Gradient problem (losing or vanishing memory)
2) Exploding gradient problem

To overcome these problems:

~~Gra~~ Gated RNN

1) LSTM
2) GRU

Solves only exploding Gradient problem

Solution to Vanishing / Exploding problem

1) Gradient clipping → Set a max value for gradient
2) Truncated Backpropagation through time (TBPTT)
   - apply backpropagation for subsequences.
3) Long-short term memory (LSTM)
   - improving the short term memory.
   - fogetting content

Date : ..... No : .....

Long Short term memory (LSTM)

ex: • the clouds, [which partly flow into each other an hang low], are at sky.

vanishing gradient problem

• clouds are at the sky. (long sentence)

ex: • Bob know swimming. [He told me over the phone that he had] served the navy for four long years.

can forget that part (forgetting)

• Bob is nice. Dan, on the other hand, is evil.

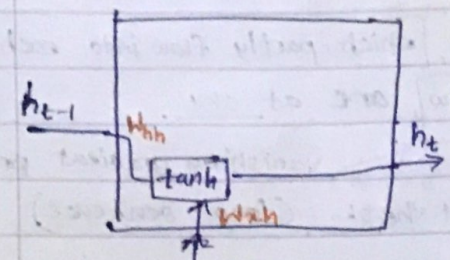
LSTM → do what part to forget and remebering.

1 — remember
0 — forget
} sigmoid function.

forget gate input gate Output gate	⟸ +	RNN cell

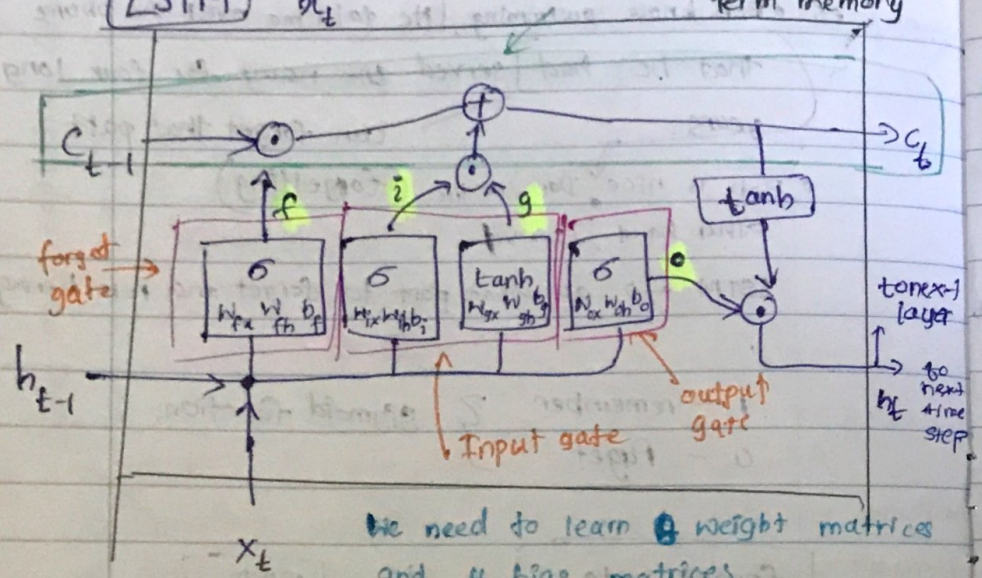
Vanilla RNN cell

$\cancel{h_t = (W_{hh} h_{t-1} + W_{xh} x_t + b)}$



$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b)$

LSTM



We need to learn 8 weight matrices and 4 bias matrices.

⊕ - Element-wise Addition

⊙ - Element-wise multiplication $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \odot \begin{bmatrix} 3 & 4 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 8 \\ 12 & 20 \end{bmatrix}$

- updating h is the final step.

forget gate

$$f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f)$$

input gate.

$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i)$$

decides what to input (Now, we need to add it into the cell state)

$$g_t = \tanh(W_{gx} x_t + W_{gh} h_{t-1} + b_g)$$

this outputs a value between -1 and 1.

- Then we multiply (element-wise) i_t and g_t to do that

$$C_t = (C_{t-1} \odot f_t) \oplus (i_t \odot g_t)$$

decide what need to added to the cell state.

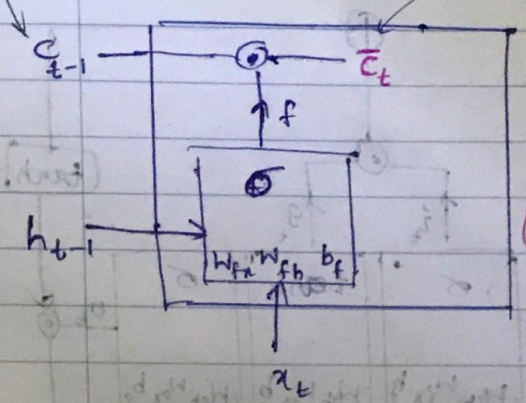
output gate

$$O_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o)$$

$$h_t = O_t \odot \tanh(C_t)$$ ← final state of the cell

cell state

Forget Gate



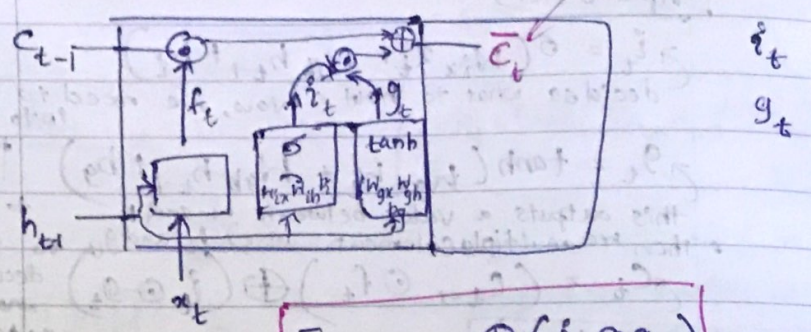
$$f = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f)$$

$$\bar{C}_t = C_{t-1} \odot f$$

Scanned with CamScanner

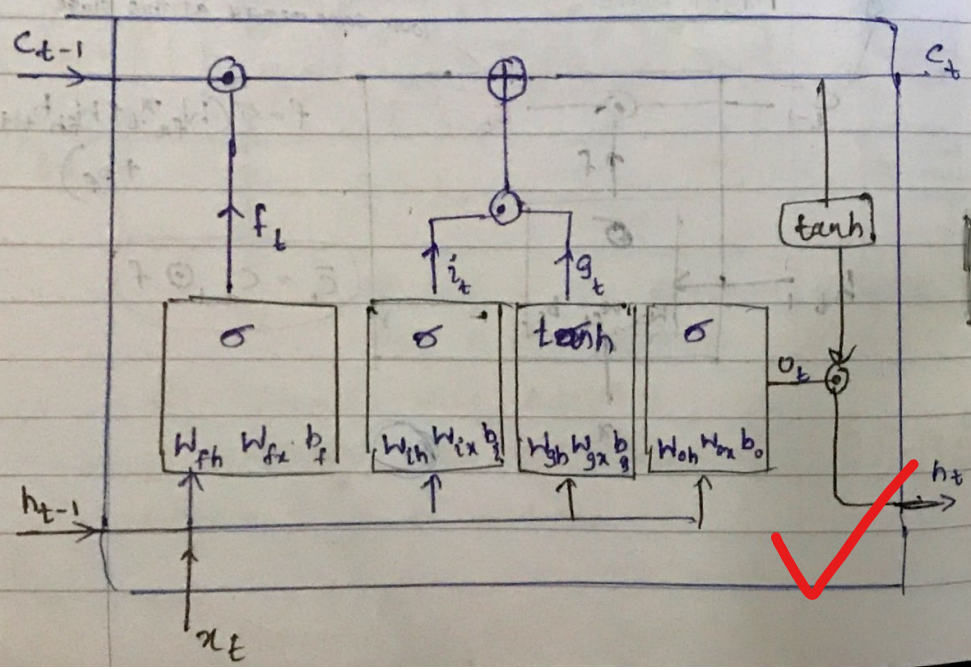
Date : No :

Input Gate



$$\boxed{\bar{c}_t = c_{t-1} \oplus (i_t \odot g_t)}$$

LSTM cell



Date : No :

$$c_t = \underbrace{(c_{t-1} \odot f_t)}_{\text{Certain information removed}} \oplus \underbrace{(i_t \odot g_t)}_{\text{what content to be stored.}}$$

final out put of the LSTM cell (h_t)

$$h_t = o_t \odot \tanh(c_t)$$

Example

punchi kale apni wasse nana kot.

LSTM works only on recurrence.

process the entire sentence at once ⟹ Attention.

Scanned with CamScanner

Date : No :

Unit05: Recurrent Neural Network (2)

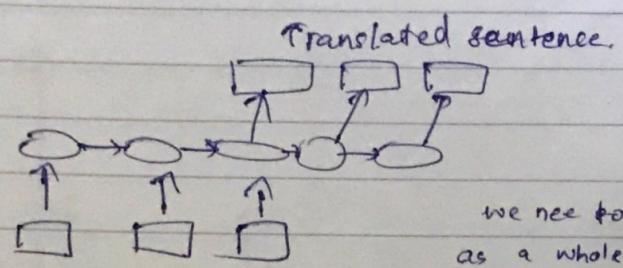
- Sequence Modeling Applications

(input) (output)
- Seq to symbol (class)
  - sentiment analysis
    - Best movie ever → Positive

  } Many to one.

(input) (output)
- Seq to seq.
  - NLP : Name Entity Recognition.
    - Acme Corp → Entities (persons cooperating)
  - Machine Translation

Seq to seq (Machine translation)



we nee to process it as a whole - one input not work by word.

- Limitations of RNNs.
  - Encoding bottleneck.
  - Not long memory.
    - Vanishing gradient problem is not fully address.
  - slow, no parallalization

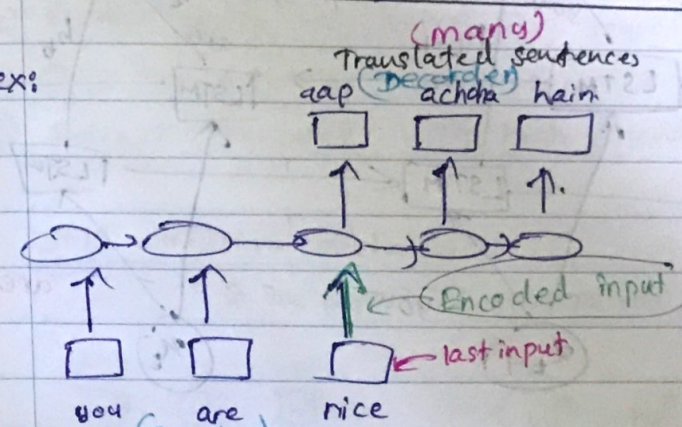
Date : 27.09.2023 No :

RNN(2)

Advanced state of art sequence modeling (processing) models.

Encoder-Decoder seq. to seq models.

ex:

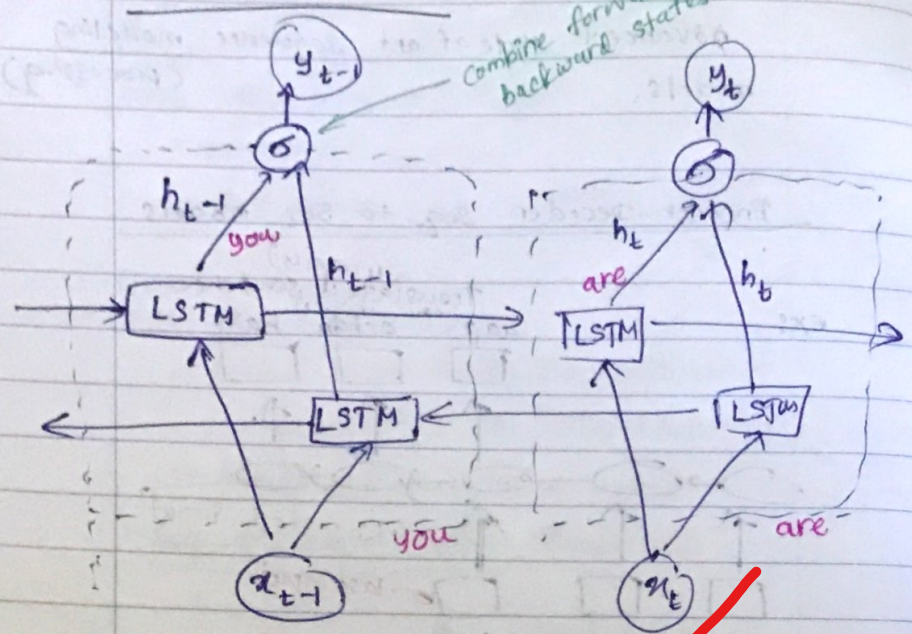


(process the complete sentence)

- First we have to encode complete sentence. Then last encoded input sent.

Scanned with CamScanner

Bi-directional LSTM



In general, Bidirectional RNNs.

* BiLSTM contains 2 LSTM Layers, one for processing input in the ~~and~~ ~~one~~ ~~for~~ forward direction and other for backward direction (sequence)

Desired Compatibilities of Language processing Models

- Continous Stream
- Parallelization (Easy, Easily ~~Gp~~ use GPU)
- Long Memory.

BiLSTM don't have This.

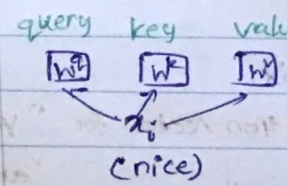
Attention

- identify which part to attend to
- Extract the feature with high attention.

1) Encode position information
2) Extract $\boxed{\text{query, key, value}}$ for search.
3) Compute attention weighting.
4) Extract features with high attention.

Self attention mechanism

in the paper "Attention All you need"

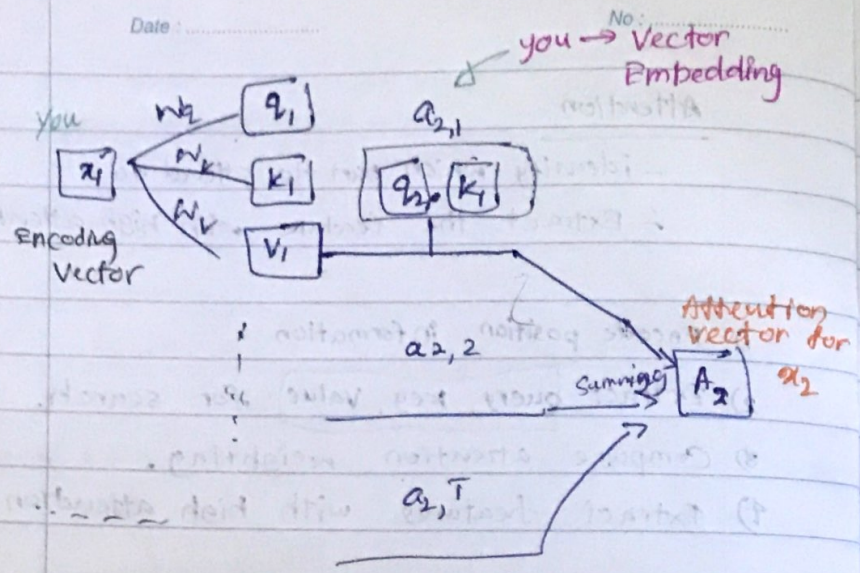


(W^q, W^k, W^v) Trainable weights.

value $\longrightarrow$ Summing.

ex. you are nice - aap achche hain

English Hindi



A_2 → is like how much attention we need to pay on x_2

Attention score matrix.

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}$$

A_1 → Attention vector for "you"
A_2 → " "are"
A_3 → " "nice"

Transformer model → eg: chatgpt.

- pretraining on large unlabled datasets

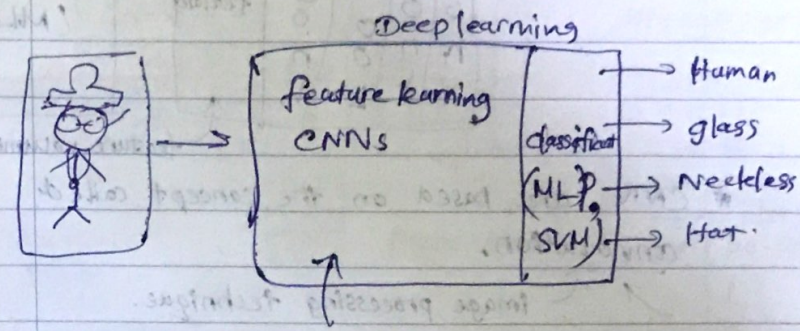
Applications → BERT, GPT-3

Date: (Images) No:

unit 6 : Convolutional Neural Networks (CNNs)

Computer vision task.

1) identify objects
2) Forms description.



Main feature learner in computer vision.

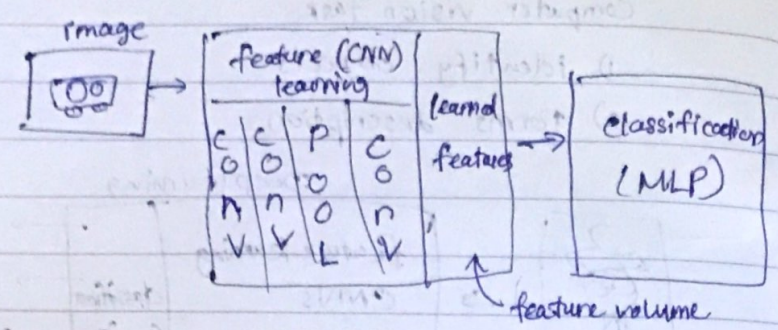
Convolutional → Coming from image processing
→ Used to extract features, transform image to some other form.

Types of layers → convolutional layer. → ResNet, VGG
pooling layer.

Scanned with CamScanner

Date : ..... No : .....

- deep learning technique.

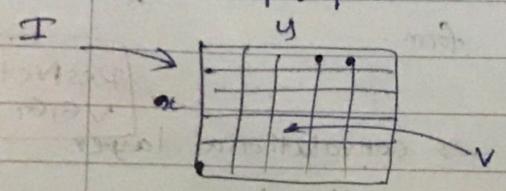


* CNN are based on the concept called convolution.
  - image processing technique.
  - Used to extract features, enhance, etc.

* then we implement convolution using neurons.

Convolution

2D topographic data - data appears on a 2D array or matrix.



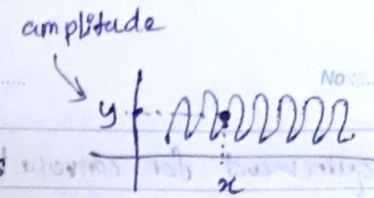
$I(x,y) = v$ ← vector

location — value.

eg:
- in color images $v = (r, g, b)$ ← 3D dimension.
- in gray images v ← 2D

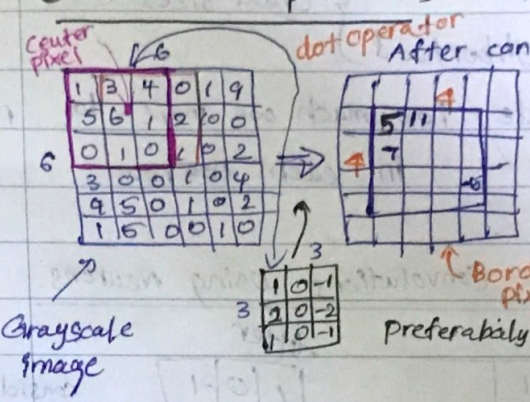
Date : ..... No : .....

- wave profile's



→ can be converted to an image.

Convolution operations



$1\times1 + 3\times0 + 4\times-1 + 2\times5 + 6\times0 + 1\times-1 + 0 + 0 + 0 = 5$

[value of the center pixel will be change]

Filter / Kernel / Mask

1	0	-1
2	0	-2
1	0	-1

filter coefficient

1, 2, 4, 2, 1

$\boxed{1\ 0\ -1}$ ← Difference

1
2
1

← Gaussian distribution

image processing

1) Smoothing → remove the noise (averaging) from the image
2) Sharpening. → highlighting the edges.
   (difference) → (Differentiation)

- this filter first smooth out and then highlight edges.

Scanned with CamScanner

Date: ..... No: .....

Requirement for convolution

1) Image
2) Filter } equal sizes
3) Neighborhood
4) Stride / how much we move to right in each ~~ce~~ row.

Performing Convolutions using Neurons.

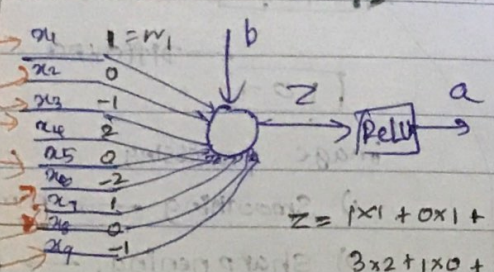
Image

1	0	1
3	1	0
2	2	4

filter

1	0	-1
2	0	-2
1	0	-1

consider filter coefficients as weights



$z = 1\times1 + 0\times1 + -1\times1$
$3\times2 + 1\times0 + 0\times-2$
$2\times1 + 2\times0 + 2\times-1$

$z = 4$

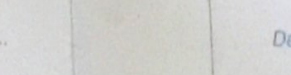
we can add the bias

$a = ReLU(4)$
$= 4$

* this shows that a convolutiono can be implement with neurons.

Date: ..... No: .....

* Therefore, to perform all convolutions of the input image, we can use a grid of neurons. one filter.

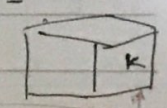


1) size of the grid of neurons ⇒ output feature map size.
2) weights ⇒ weights are shared.

- example in previous need 16 neurons → (4×4)

3) No. of weights = filter size + bias(1)

* We can have multiple filters (k)
  1) size of the grid of neurons = feature map size × k



feature volume of k depth

  2) No. of weights = (filter size × k) + (k biases)

* Input size

Practicaly, input depth can be of any size.

eg:

gray images	RGB	Output volume from the prev. convolution.
⇓	⇓	
depth = 1	depth 3	depth = d

Scanned with CamScanner

Date : ..... No : .....

* Width and height

1) We can fix the width and height through i) padding
   ii) No pooling

2) We can change the width and height through 1) No padding
   2) Pooling } ⟹ change the filter size

⇓

filter depth = input depth

eg: If the input is an RGB image we need to use 3 filters.

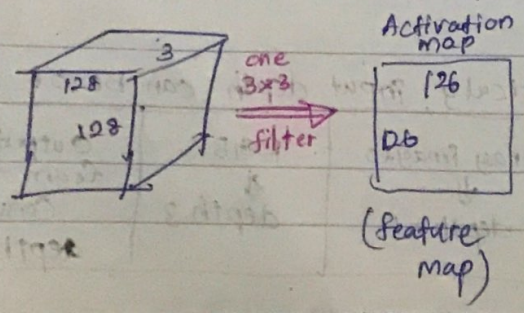
If the filter size is 3×3 ⟹ we need to use ③ 3×3 filters.

How many weights = 27 weights.

(If No bias)

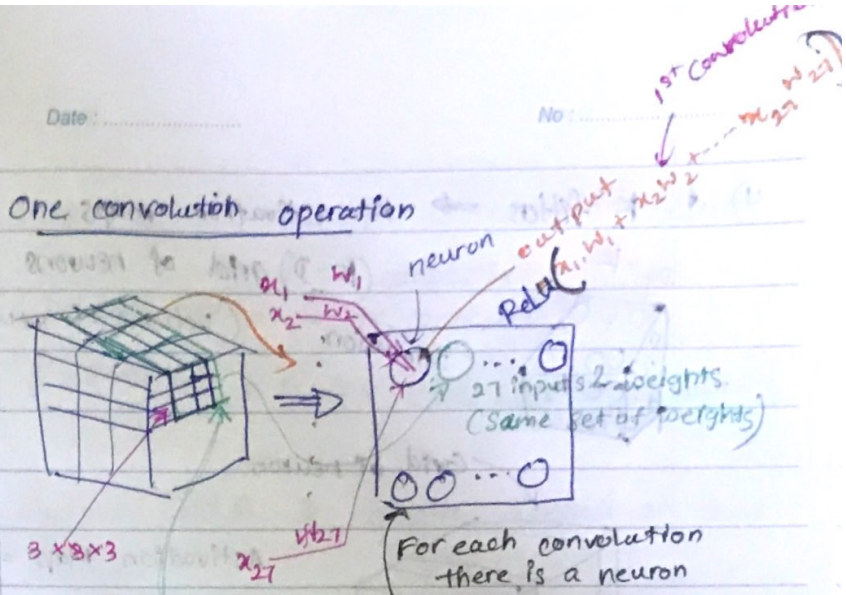
2023/10/11

* In practice we need multiple convolutions



Date : ..... No : .....

One convolution operation



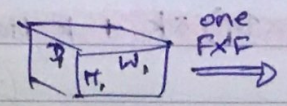
For each convolution there is a neuron

Conclusions

1) * This means activation map is 2D grid. (But input 3D)

2) * One filter ⟹ one set of weights
   eg: 3×3 ⟹ 3×3×3 weights
   with 3D input

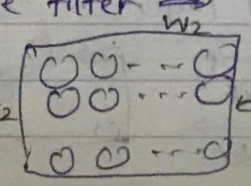
In general,



(one weight matrix)
one set of weight will have F×F×D_1 weights

matrix depth = Input Depth

3) * One filter ⟹ one activation map (Grid of neuron 2D)



each neurons shares the same F×F×D_1 weights.

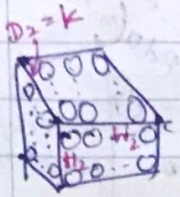
Scanned with CamScanner

4) • K filters ⇒ K activation maps.

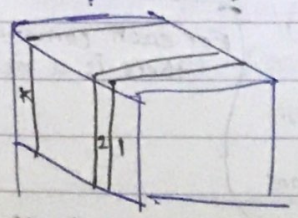
(K-D) grid of neurons.

Dimension

(volume of neurons)



Grid of neuron



Activation Map = AM

- AM 1 neurons shares (W_1) $F \times F \times D_1$ + one b weights.
- AM 1 neurons shares (W_2) $F \times F \times D_1$ + b weights
- ⋮
- AM K neurons shares (W_K) $F \times F \times D_1$ + one bias.

∴ Therefore, total number of weights

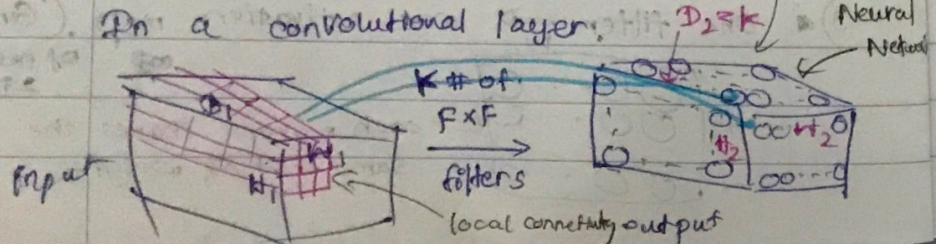
$$= (F \times F \times D_1) \times K + K \text{ bias}$$

from convolutional layer.

(filter coefficient) trainable weights $(F \times F \times D_1)K + K$ bias

In summary,

In a convolutional layer,



only part of the input connected to neural network ⇒ local connectivity.

(gn: "fully connect all are connected").

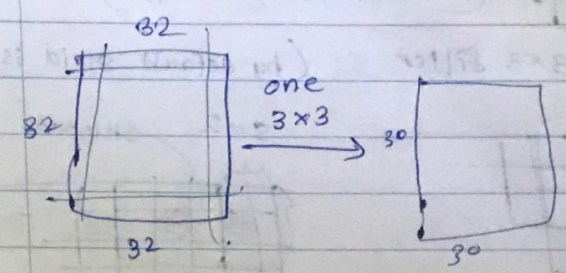
$(F \times F \times D)K + K$ biases ⇒ Parameter sharing.

∴ Output width & height depend on two concepts

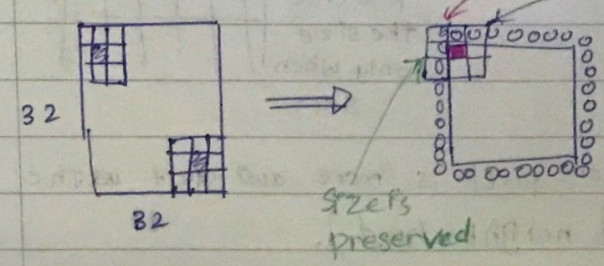
1. Padding
2. Stride.

1) Padding

eg: Gray image

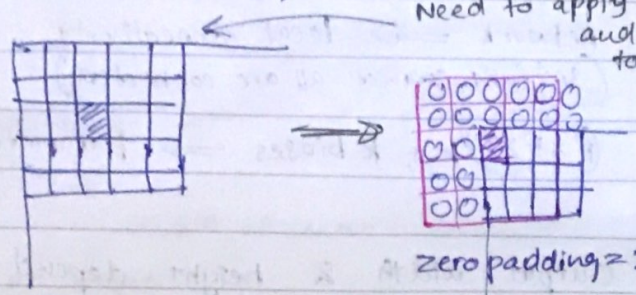


zero padding = 1



If the filter 5×5,

Need to apply 2 rows and columns to padding



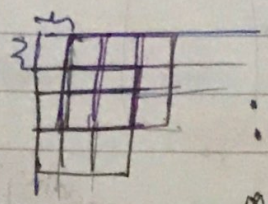
zero padding = 2

Filter size	padding
3×3	1
5×5	2
7×7	3

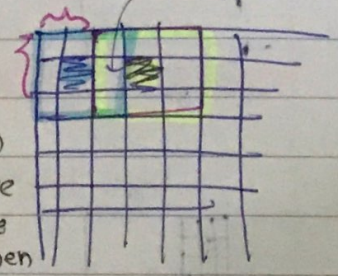
2) Stride ← (How much you move filter)

eg: 3×3 filter (by default strid is 1)

S=1



S=2 skipped

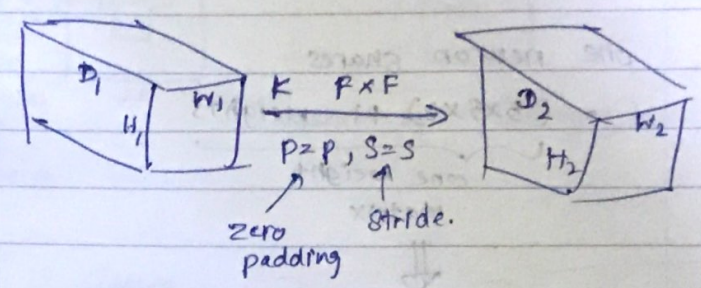


* We can preserve the size only when S=1

(Here one pixel is more and start the second neighborhood).

output size ⇒ $\frac{W}{2} \times \frac{H}{2}$

Output width & height

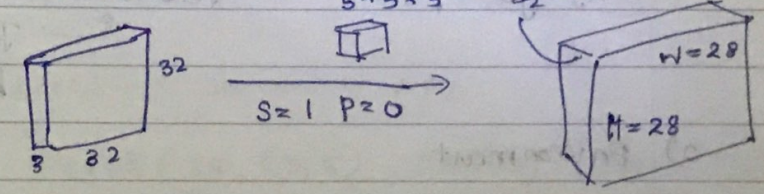


$D_2 = K$

$$W_2 = \left[\frac{W_1 - F + 2P}{S}\right] + 1$$

$$H_2 = \left[\frac{H_1 - F + 2P}{S}\right] + 1$$

eg: 32×32×3



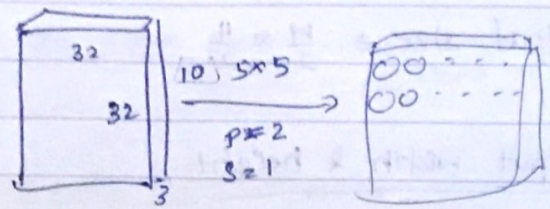
$$H_2 = W_2 = \left[\frac{32 - 5 + 2\times 0}{1}\right] + 1$$

$= 28$

Scanned with CamScanner

Date : ..... No : .....

eg :



one neuron shares

$= \underbrace{(5\times5\times3)}_{\text{one weight matrix}} + 1$ weights

⇓

10 such weight matrices.

Total weights

$= (5\times5\times3)\times10 + 10 = 760$

Implementation of CNN

1) Language – Python → Pytorch / → Tensorflow ← Tensorflow keras API

2) Environment

Anaconda → – own computer * with GPU

– Online collaborative tools

• Google colab.

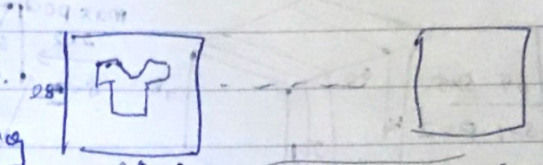
Date : ..... No : .....

3) Problem

- Fashion MNIST (Toy DataSet)

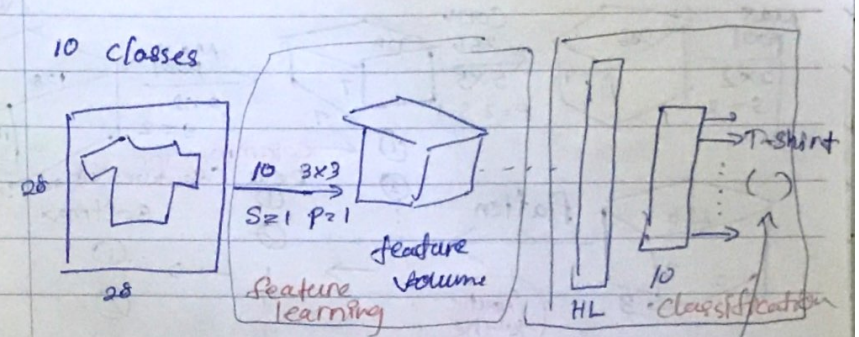
70000 images.

- training 60000
- test 10000



10 fashion items (Images)

10 classes



Highest probability

* fully connected layers $\xrightarrow{\text{called}}$ Dense

keras → sequential API

Conv2D (64, (5, 5), padding = "same")

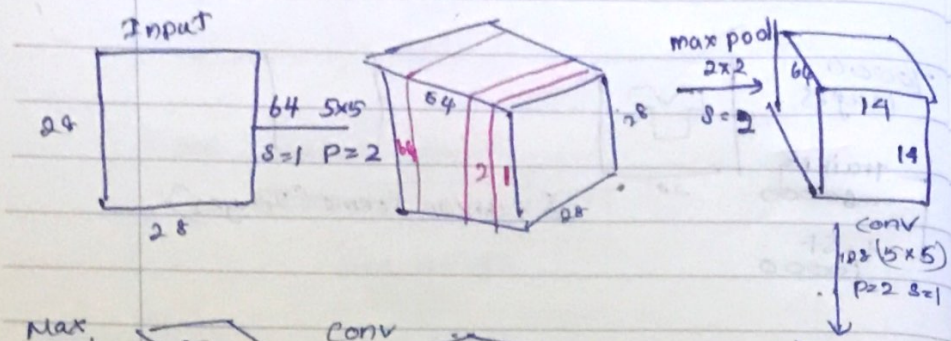
layers filters 2

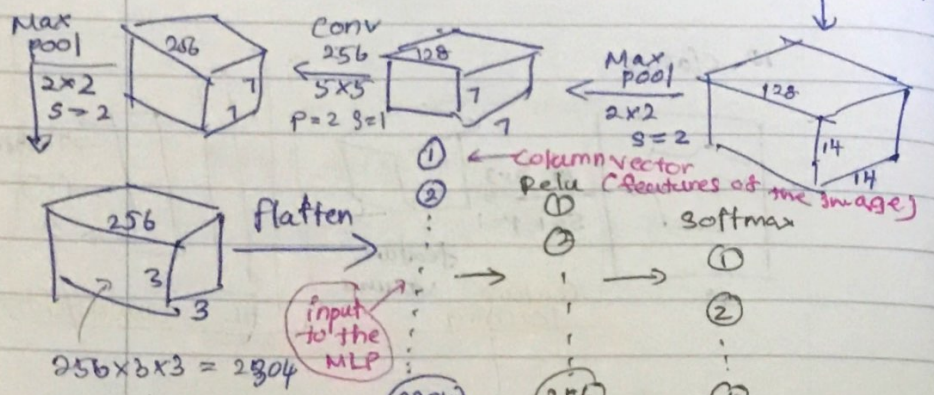
output layer → softmax.

Scanned with CamScanner

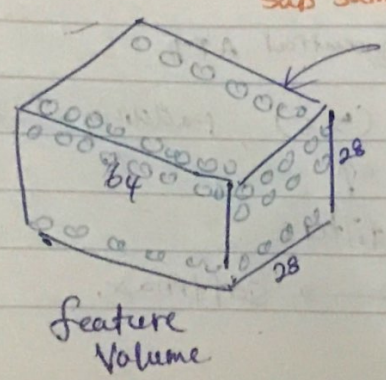
Date : ..... No : .....

CNN Architecture





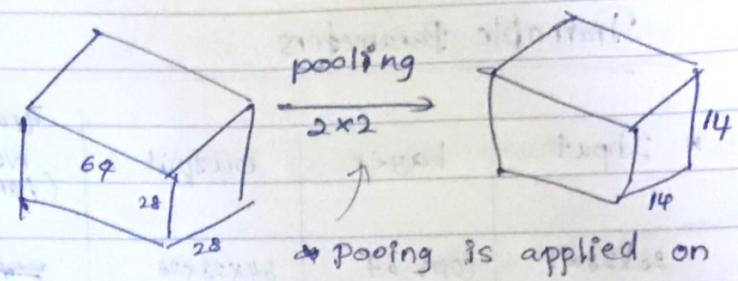
Pooling ← Dimensionality reduction OR Sub sampling technique.



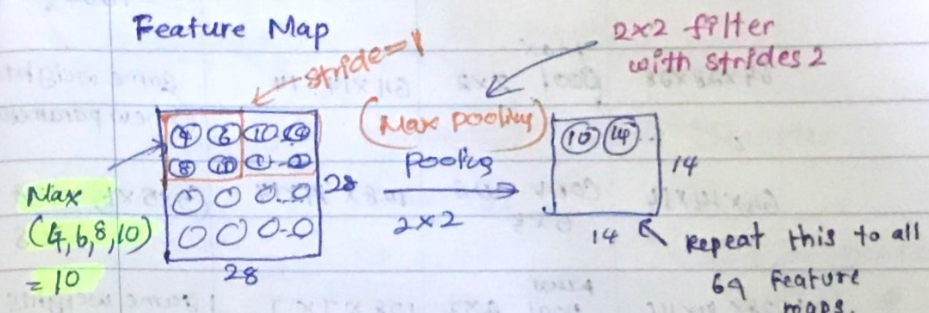
28x28x 64 neurons
= 50176 neurons
problem.

* pooling is used to reduce neurons.

Date : ..... No : .....

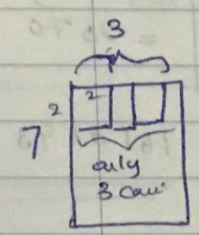


* Pooing is applied on each feature map respectively



Types of pooling

- Max pooling → most common pooling
- Average pooling
- Global max pooling
- Global average pooling



compiling → setting the hyperparameter

$(F \times F \times D) \times K + K$

Date : No :

Trainable Parameters

Input	layer	output	# trainable weights (Parameters)
28x28x1	Conv 64 5x5	64x28x28	~~50176~~ (5x5x1)x64 + 64 = 1664
64x28x28	Max pool 2x2	64x14x14	same weights (0 new parameter)
64x14x14	Conv 128 5x5	128x14x14	(5x5x64)x128 + 128
128x14x14	Max pool 2x2	128x7x7	same weights
128x7x7	Conv 256 5x5	~~128~~ 256x3x3	(5x5x128)x256 + 256
(256x3x3) 2304	Hidden (FC)	256	(2304x256) + 256 = 590080
256	FC	10	(256x10) + 10 = 2570
		Total # parameters	1618698

Date : No :

Popular CNNs & Pre-trained CNN Model

Fashion MNIST image classification

→ CNN:

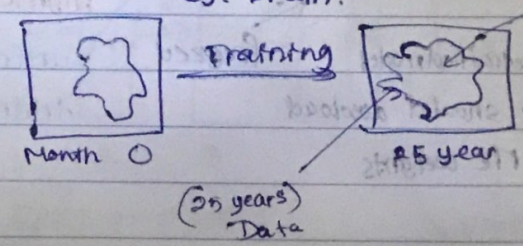
- 3 Conv layers
- 3 pooling layers
- Two FC layers.

our CNN architecture

trained all layers from zero

Approach 2: Pre trained Models (Transfer learning)

eg: Brain.



large Dataset

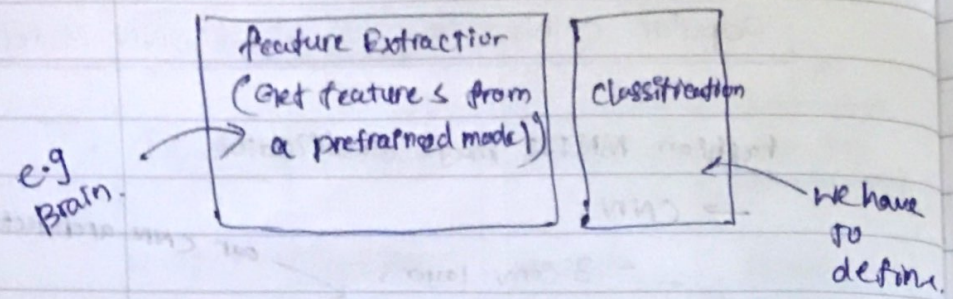
↳ Generalized data

eg: Image data ⇒ 1000 object classes.

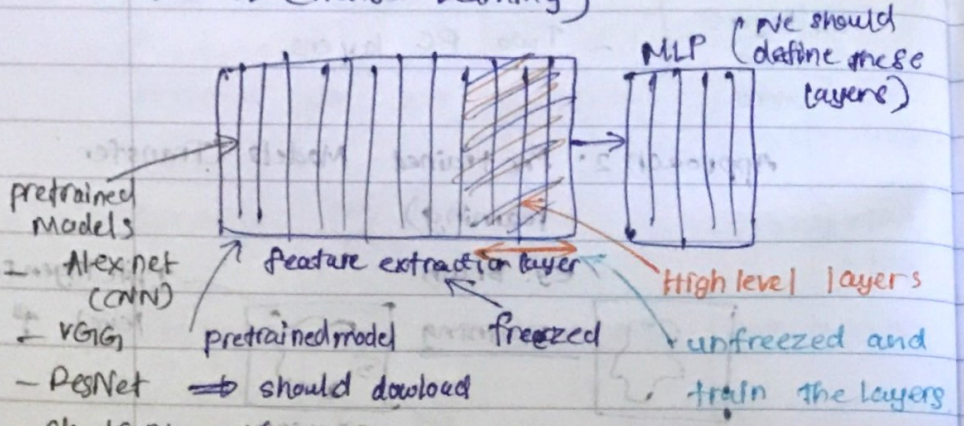
large language model.

Scanned with CamScanner

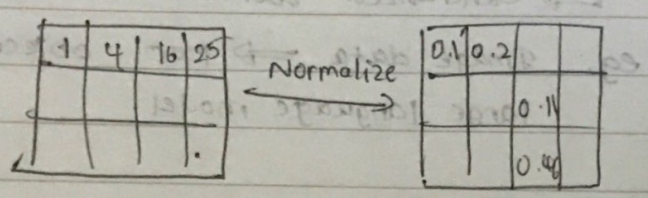
Date : No :



In TL (Tranfer Learning)



Normalize



Flatten 3D → 1D

vanishing problem ~~+ [illegible]~~
— long sequence

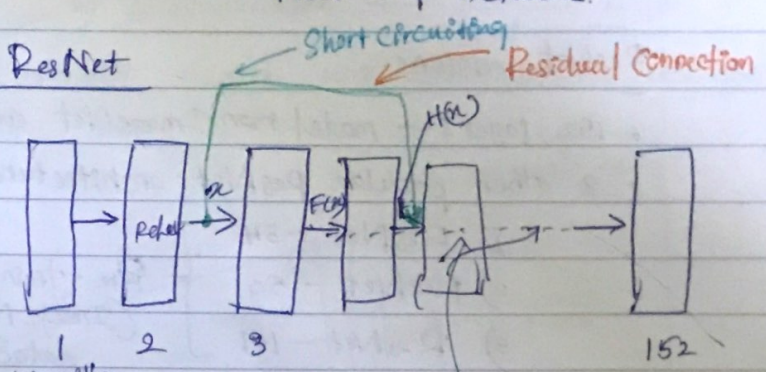
Date : No :

Object Detect → classification + localization

AlexNet
VGG
ResNet
} Trained on ImageNet

(Residual Networking)
152 layers.
Real deep network.

ResNet



Residual Connection ⇐ Solution

$$H(x) = F(x) + x$$

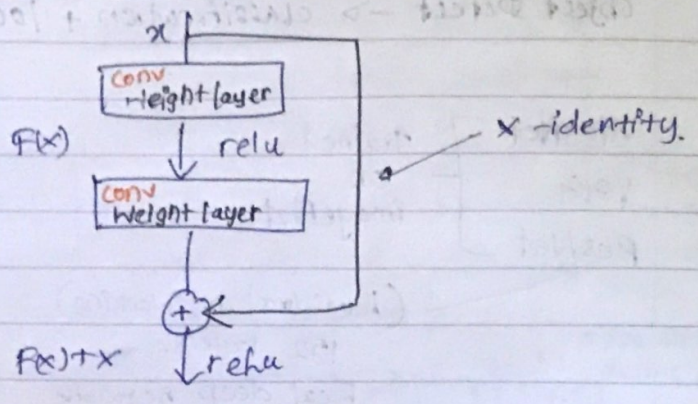
- Forms a residual block.

ResNet ← Stack of residual blocks.

Scanned with CamScanner

Date : No :

Residual Blocks.



ResNet versions

- 152 layers — model won ImageNet competition
- 3 other popular ResNet architectures
  1) ResNet - 34
  2) ResNet - 50
  3) ResNet - 101

  } Pre-trained models (Image Net dataset) available
  - Keras
  - Pytorch.

36 page → image classification
— Backbone Networks

Date : 19.10.2022 No :

GAN

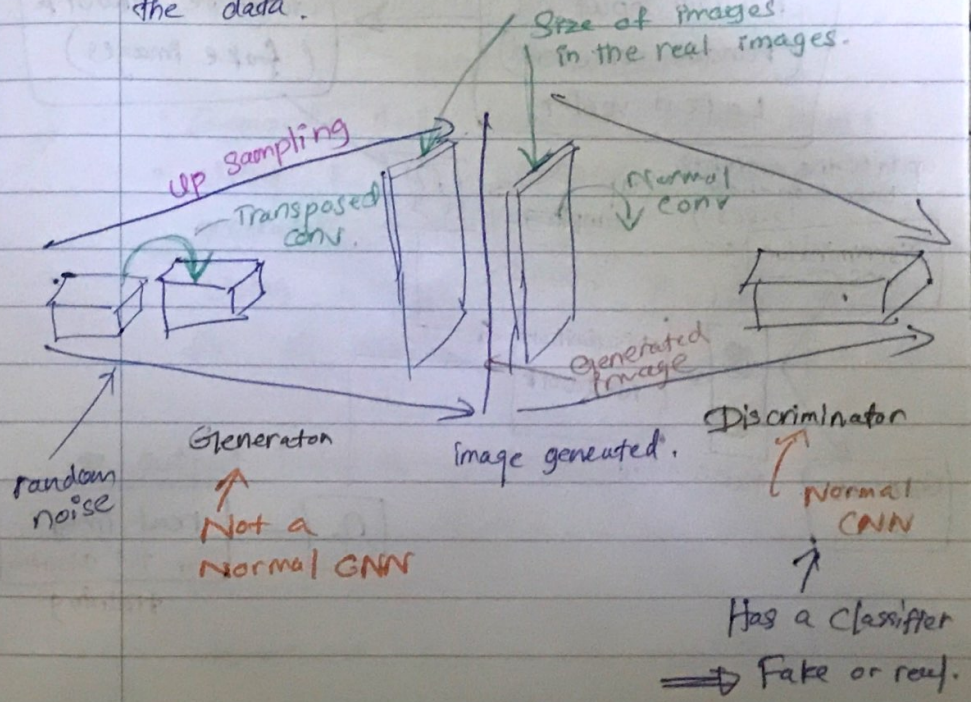
Most popular based GAN → DCGAN (Deep Convolutional Generative Adversarial Network)

using that made many advance GANS

- Any GAN there are 2 networks
  1) Generator
  2) Discriminator ← Recognize wether it is real or not

  Judge (police)

* Generator happen until Discriminator will fool the data.



Scanned with CamScanner

Date : No :

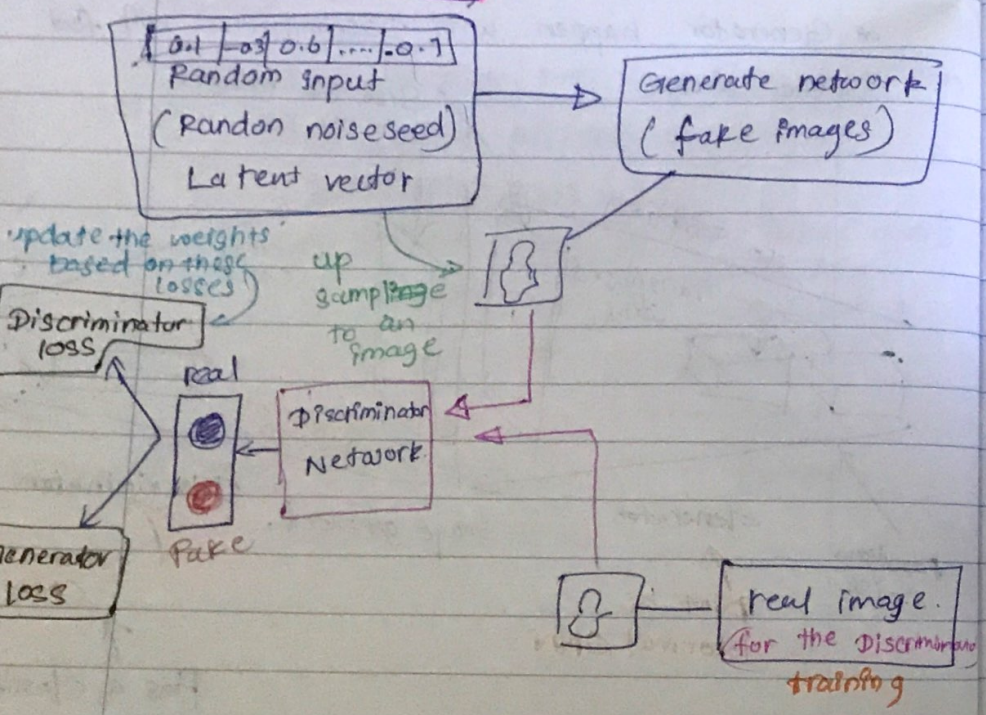
Adversarial

- Generator trying to fake and discriminator trying not to be fooled.

GAN's take game-theoretic approach

- learn to generate from traing...

GAN Architecture



Date : No :

Training GANs

* When discriminator is trained, generator is freezed (No weights updated).

DCGANs characteristics:

- ~~Replacing~~ Fractional-strided conv. (generator)
  transposed conv.
- Using Batchnorm (BN)
  Batch normalization.
- Removing fully connected hidden layers.
- Using ReLU activation function in generator. for all layers
- Using LeakyReLU in the discriminator for all layers.

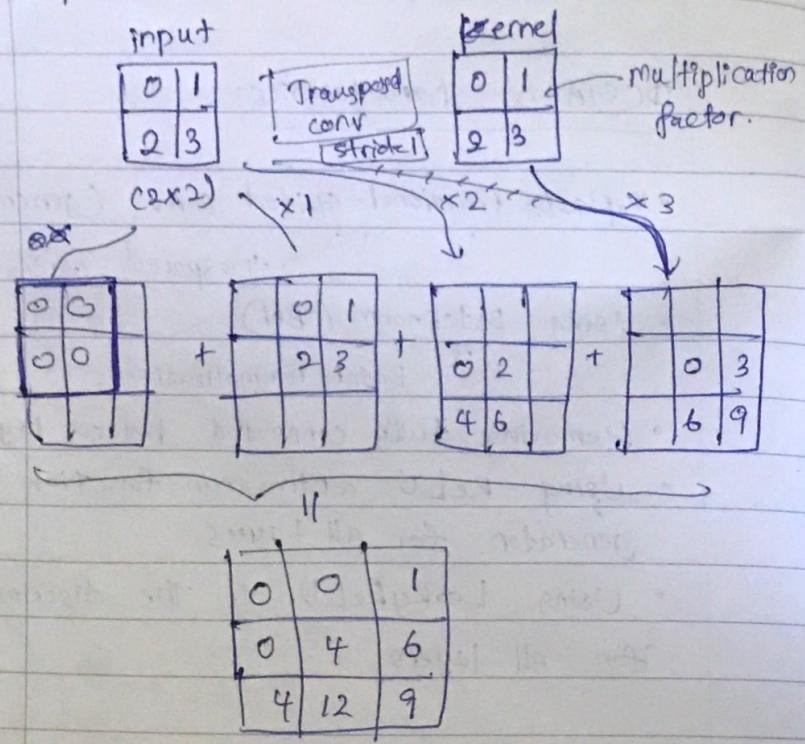
* output layer ← tanh.

Scanned with CamScanner

Date : ..... No : .....

Strided Convolutions (Transposed conv).

- Transposed conv with 2x2 kernel is computed for a 2x2 input tensor.



Date : ..... No : .....

Exam

- What is GAN? using an example.
- Generator, Discriminator.
- DCGAN
- What is meant by transposed conv.
- 7 slide – Architecture. (Slide 9)
- Monalisa example.

$\boxed{7, 9, 10}$ ← slides.

Scanned with CamScanner