Module 2

CNN — ConvNet

Deep learning

Date / /
Page

- Convolutional Network — class of feed forward NN
- Deep feed forward network
- used in image recognition, audio & video recognition
- Made up of multiple layers:
  - Convolutional layers
  - pooling layers
  - fully connected layers

a) Convolutional layers

- Key component of CNN
- filters are applied to I/p images to extract features
- features: edges, shapes, textures etc.

b) Pooling layers

: O/p of Convolutional layer is passed to pooling layers

- used to down-sample the feature Maps [Matrix]
- Reduce Spatial Dimensions while retaining important information

c) Fully connected layers →

O/p of pooling layers (reduced dimensions based images) pass to one or more fully connected layers.

Date / /
Page

- These layers are used to make a prediction or classify the image.

⇒ Uses of CNN

- used to Recognize patterns & features from large dataset of labeled images
- used to classify new images or extract features
  - Object Detection / Recognition
  - Image segmentation
  - Object classification
  - widely used in computer vision, image processing & other related fields

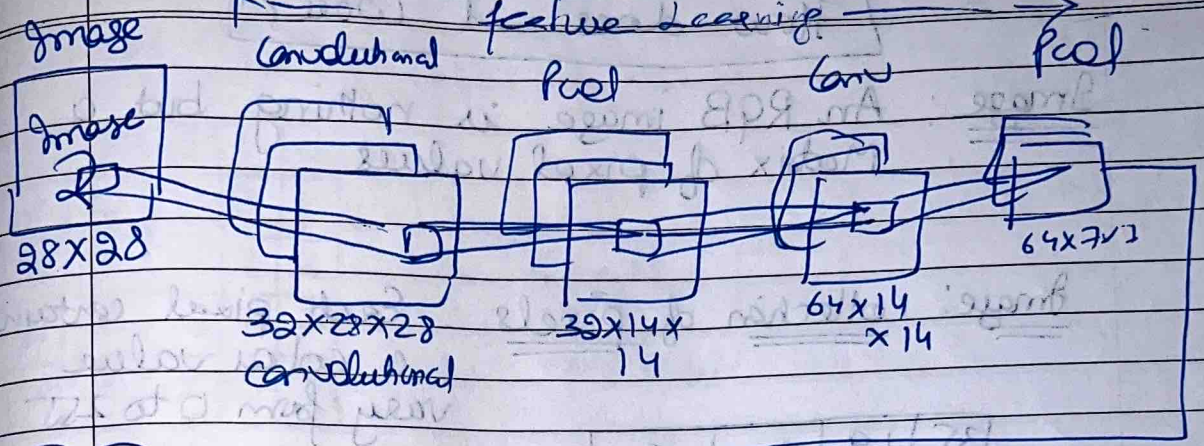
eg. Self driving cars, Medical imaging & security systems

Building Blocks

CNN :- Analyze visual Imagery

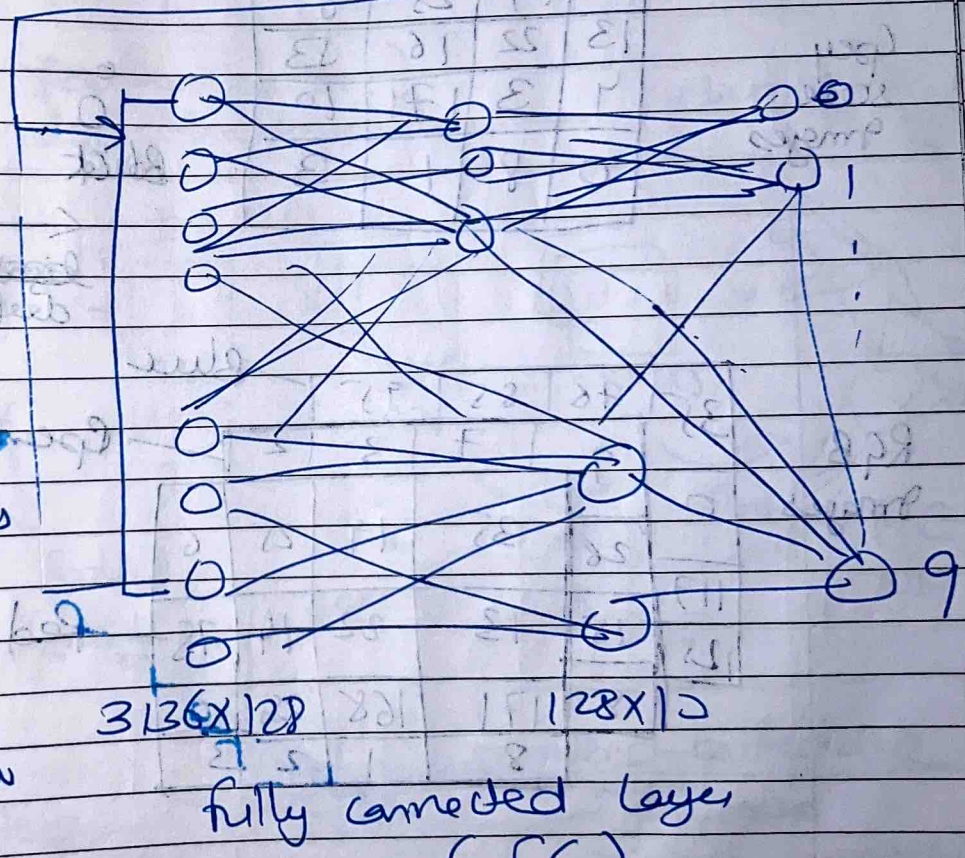
Date ... / ... / ...

Page ...



Role of the ConvNets is

To Reduce the images into a form that is easier to process, without losing features that are critical for getting a GOOD PREDICTION



fully connected layer (FC)

Extract features (detect object)

final O/P

How does it work?

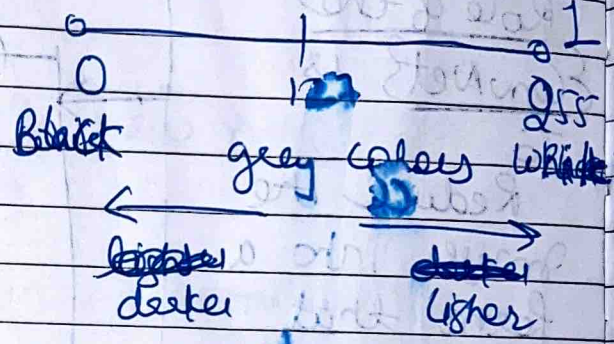
Date...../...../.....
Page.....

Image: An RGB image is nothing but a Matrix of pixel values

Image: collection of pixels: Each pixel contains a color value vary from 0 to 255

Grey scale images

35	19	25	6
13	22	16	13
4	3	171	10
0	8	1	3



RGB Images

35	76	85	95	Blue
	9	7	3	2
0	26	35	19	25	6
117	15	13	22	14	78
125	8	171	168	256	0
		8	1	2	3

Green

Red

3 Color Channels

eg. Grey-scale Image

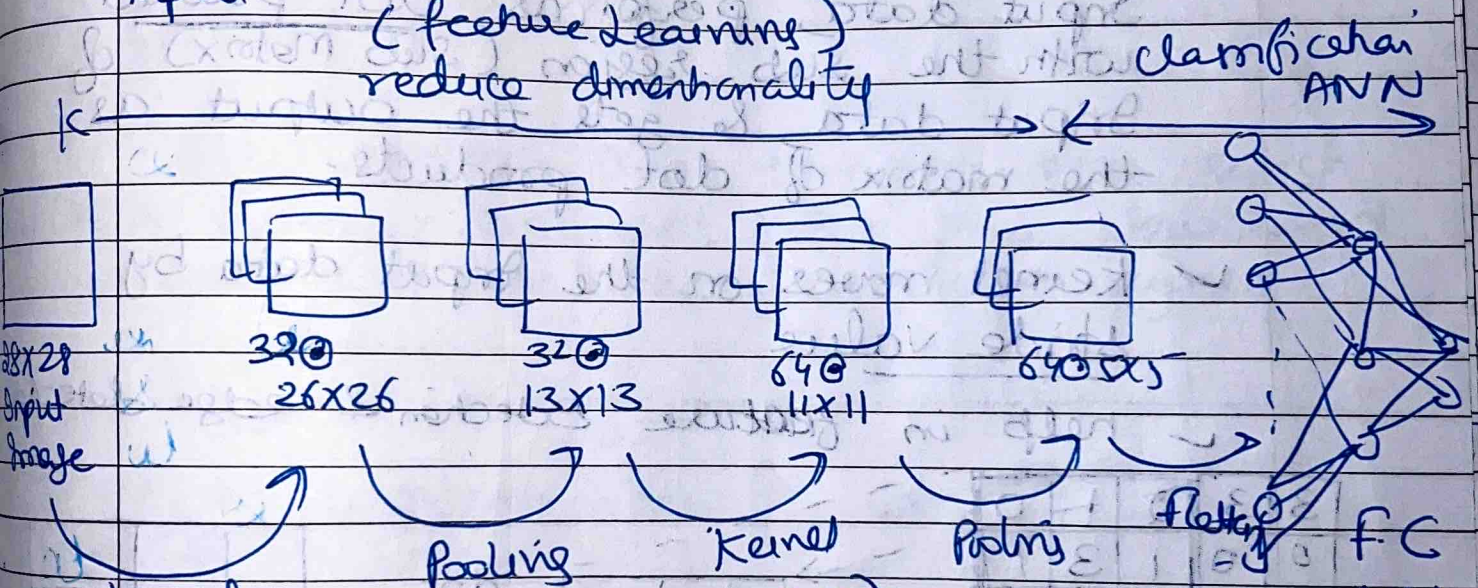
Date...../...../.....

Page.....

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

5x5 input

→ Extract feature using CNN



① Kernel
- ↳ filter
- ↳ feature detector

② Stride

③ Padding

④ Pooling

⑤ Flatten

Need to Reduce

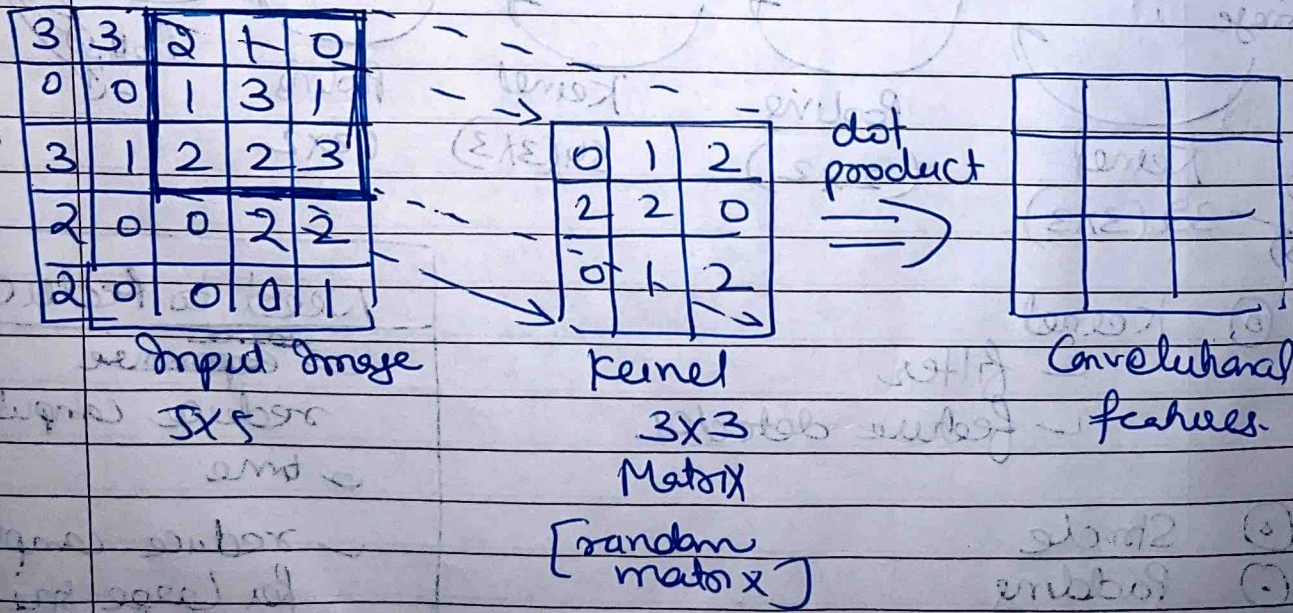
- ↳ ~~reduce~~ ~~expensive~~
- reduce computation time
- ↳ reduce complexity for large size image

Date ..../..../....
Page ....

To reduce the dimentionality of original image (in order to reduce complexity), kernels & pooling layers used.

What is Kernel? ~ Heart of CNN

- Kernal is a filter that is used to extract the features from the images
- It is a Matrix that moves over the Input data, performs dot product with the sub region (sub Matrix) of Input data & gets the output as the matrix of dot products.
- Kernel moves on the Input data by stride value
  - help in feature detection & edge detection



Input Image 5x5

Kernel 3x3 Matrix [random matrix]

Convolutional features

Dot product eg:

Date: / /
Page:

$$= \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 2 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

$$= \left[(2*0+1*1+2*0) + (1*3+3*2+1*0) + (2*0+2*1+3*2) \right]$$

$$= 0+1+0+3+6+0+0+1+6$$

$$= (16) \text{ feature pair}$$

Now move to next one, select sub matrix again perform dot matrix mean downward. to generate a feature map of convolution layers

e.g CNN working (How it works)

Let

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

Input image 5x5

Move one by one:

0	1	2
2	2	0
0	1	2

kernel 3x3 (filter)

Step 1: subMatrix:

$$\begin{bmatrix} 3 & 3 & 2 \\ 0 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

kernel

$$= [3*0 + 3*1 + 2*2] + [0*2 + 0*2 + 1*0] + [3*0 + 1*1 + 2*2]$$

$$= [3+4] + [0] + [1+4]$$

$$= [12]$$

Step 2 → ~~[illegible]~~ / move to next ~~col~~ & again select subMatrix (new)

→ Move

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

kernel

0	1	2
2	2	0
0	1	2

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

$$= \begin{bmatrix} (3*2 + 2*1 + 1*2) + \\ [0*2 + 1*2 + 3*0] + \\ [1*0 + 2*1 + 2*2] \end{bmatrix} = \begin{bmatrix} 4 + \\ 2 + \\ 6 \end{bmatrix}$$

$$= [12]$$

Date ... / ... / ...

Page ...

Step3 – Stride to next col. one by one

Move →

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

·

0	1	2
2	2	0
0	1	2

$$= \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 2 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$ dot Matrix

$$= 17$$

Step4 → Now ~~Stride~~ Move to downward (one step down.

Move ↓

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

0	1	2
2	2	0
0	1	2

Kernel

$$= (10)$$

repeat till all subMatrix not selected to extract feature & so

Design feature Map

Date...../...../.....

Page.....

12	12	17
10	17	19
9	6	14

reduce dimensionality

3X3

Convolutional layers.

To find Size of feature Map →

Output = [Size of Input − Size of Kernel] + 1

eg: = [5−3] + 1

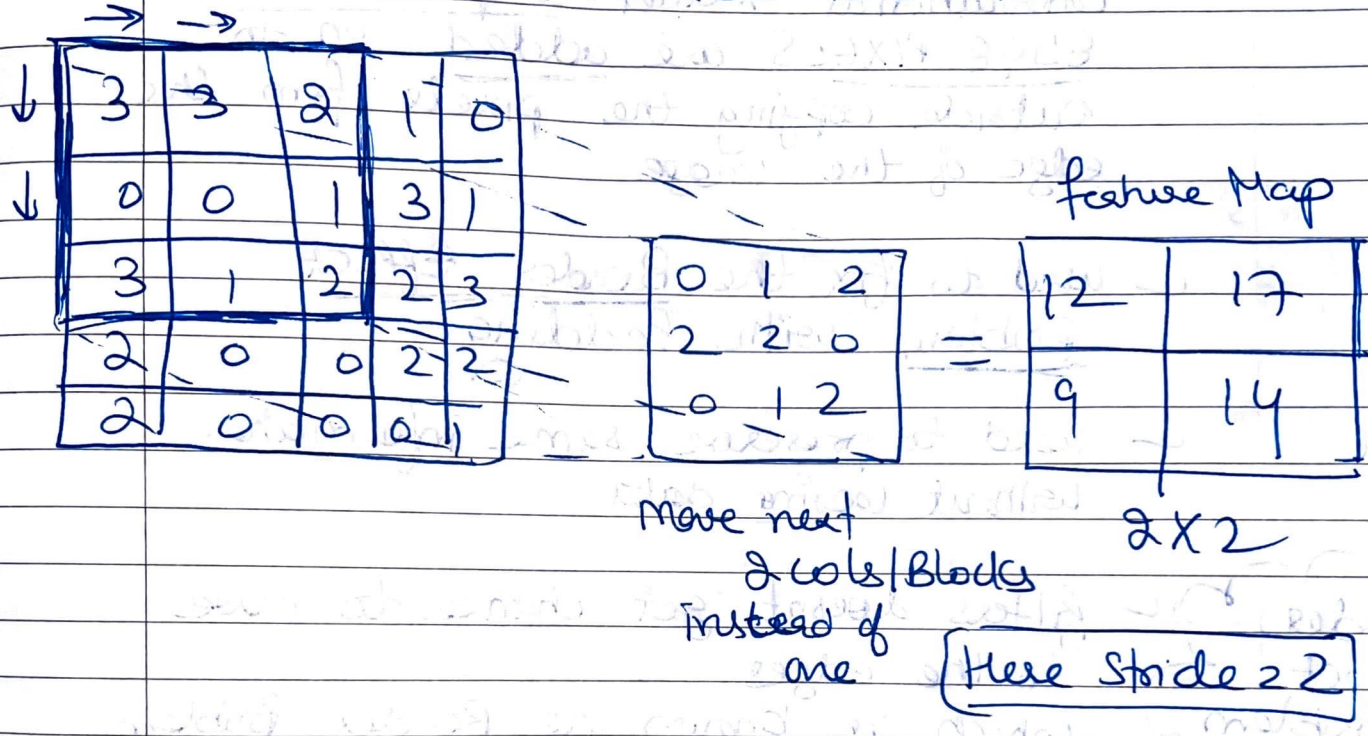
= 2+1

= (3) 3X3 Matrix

STRIDE – Move to next 2 cols/rows instead of 1

Than o/p will be of feature Matrix 2X2

→ used to reduce the dimensions more



Move next 2 cols/Blocks instead of one

Here Stride = 2

⇒ To find Size of feature Matrix after stride:

$$O/P = \left[\frac{\text{Size of Input} - \text{Size of Kernel}}{\text{Stride}}\right] + 1$$

$$= \left[\frac{7-3}{2}\right] + 1$$

$$= \left[\frac{5-3}{2}\right] + 1 = 2 \text{ i.e } 2\text{X}2 \text{ Matrix}$$

[redacted] — used to ↑ the dimensions (H x W)

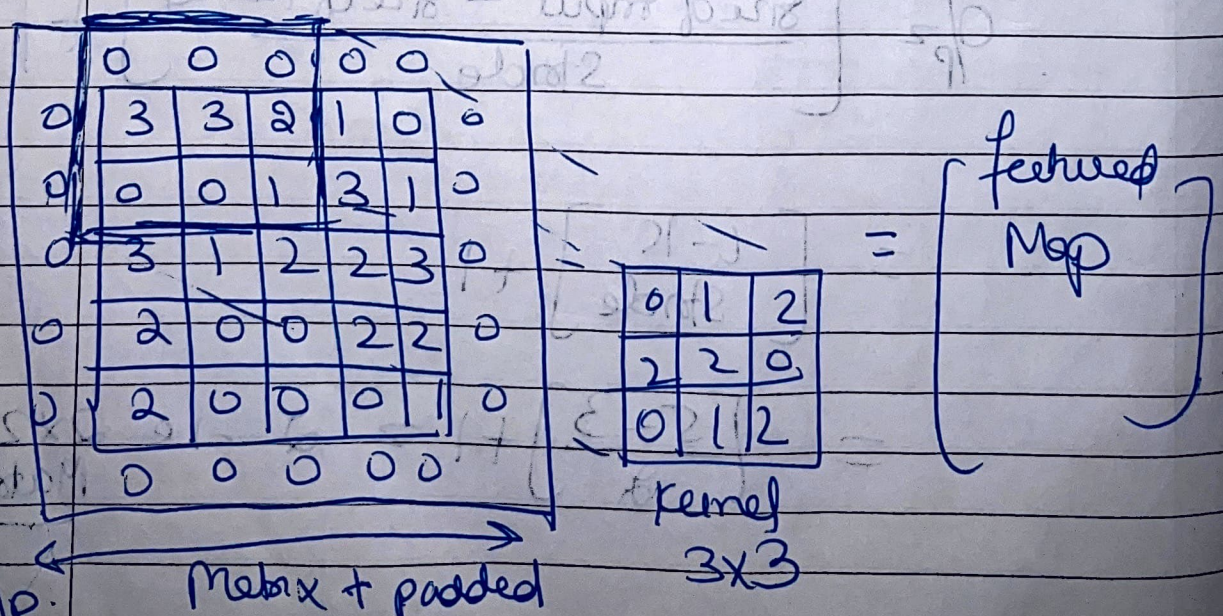
Padding → (Border Problem) solver

Padding is the best approach, where the number of pixels needed for the convolutional kernel to process the EDGE PIXELS are added on to outside copying the pixels from the edge of the image

- used to fix the Border Effect Problem with Padding
- used to preserve same information without losing data

Border Effect Problem

- filters doesn't get chance to move to the edges.... which is known as Border Problem
- To solve this problem we add 0 padding all over the Input edges.



Now kernel will move by including padding values also within Matrix. (5x5 Matrix)

(*) Total no. of size of feature Map Matrix:

$$O = \left[\frac{\text{Input} - \text{kernel} + 2\text{padding}}{S}\right] + 1$$

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

Padding 1 add on all edges

Stride = 1 one move

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

= 5 X 5 Matrix (preserve same information)

$$\boxed{I/P = O/P}$$

~~Feature Map Matrix~~

Pooling [Detect features & reduce dimensions]

Max Pooling

~~Avg~~ Pooling

What is Pooling? The output of conv layer passed to pooling i.e Featured Matrix

- Pooling is required to down sample the detection of features in feature Maps
- Pooling layers provide an approach to down sampling feature Maps by summarising the presence of features in patches of the feature Map.
- Two common pooling methods are →

(a) Max Pooling : Summarize most activated presence of a feature

~~(b) Average Pooling~~

- (b) Average Pooling — Summarize the average presence of a feature
- It is used to reduce the dimensions w/o loosing important data/features

Date...../...../.....

Page.....

I Max Pooling (let 2x2 Matrix)

Let feature Map Matrix : 2x2 Matrix Sampling from feature Map & find Max. of it to get 2x2 max pooling matrix

2x2 ←

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

5x5 feature Map

→ MAX Pooling (2x2) ?

Step 1

$$\text{Max}\begin{bmatrix} 6 & 14 \\ 14 & 12 \end{bmatrix} = (14)$$

Step 2

$$\text{Max}\begin{bmatrix} 17 & 11 \\ 12 & 17 \end{bmatrix} = (17)$$

Step 3 Move down and

$$= \text{Max}\begin{bmatrix} 8 & 10 \\ 11 & 9 \end{bmatrix}$$

$$= (11)$$

Step 4:

$$= \text{Max}\begin{bmatrix} 17 & 19 \\ 6 & 14 \end{bmatrix}$$

$$= (19)$$

$$\text{Max Pooling}_{2x2} = \begin{bmatrix} 14 & 17 \\ 11 & 19 \end{bmatrix}$$

II Average Pooling

Date ... / ... / ...
Page ...

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

→ Average Matrix

Step) 2 $\begin{bmatrix} 6 & 14 \\ 14 & 12 \end{bmatrix}$

Avg $\begin{bmatrix} 6 & 14 \\ 14 & 12 \end{bmatrix}$

$= \frac{\text{Sum of elements}}{\text{total elements}}$

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

$= \frac{46}{4} = 11.5$

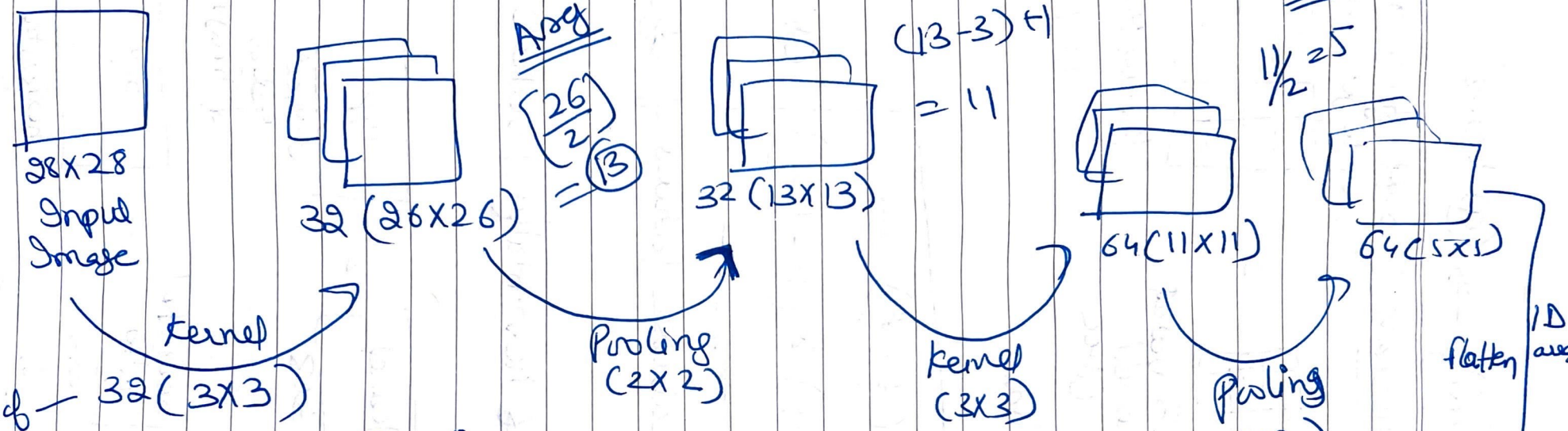
Ily for others

Average Pooling :

11.5	4.25
9.5	14.0

Convolutional Neural Network

CNN



reduce dimensions (filter features) size

Here

$$O = (i - k) + 1$$
$$= (28 - 3) + 1$$
$$= 26$$
$$= 26 \times 26$$

Kernel

Average

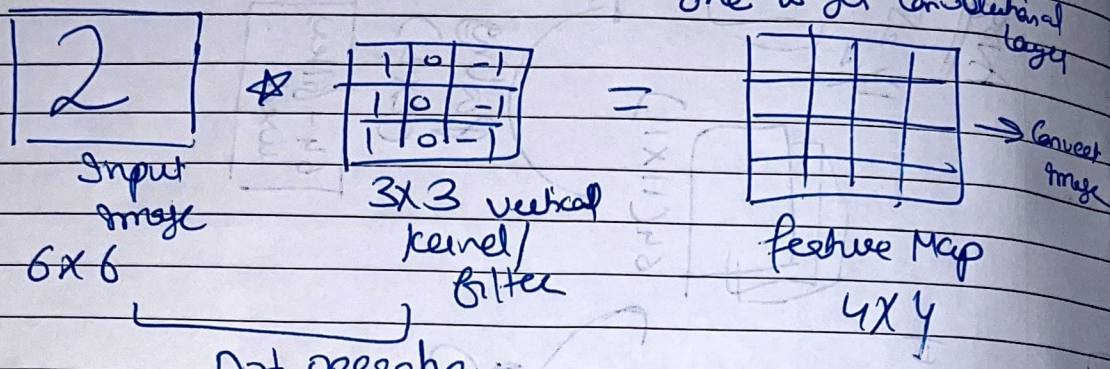
$$= \frac{\text{size of Matrix}}{2}$$

Pooling

Convolution operation → ① kernel

Date ... / ... / ...
Page ...

(a) Dot Matrix operation : Shift filter one by one to get Convolutional layer



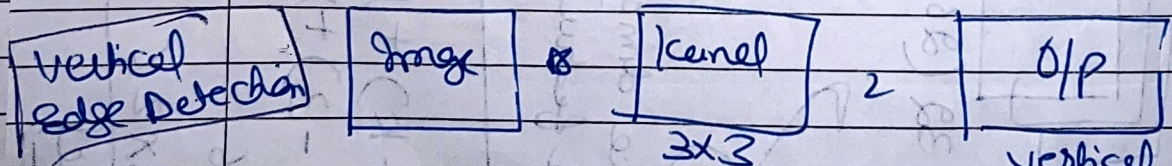
formula : $(n \times n) * (f \times f) = (n-f+1) * (n-f+1)$ (size)

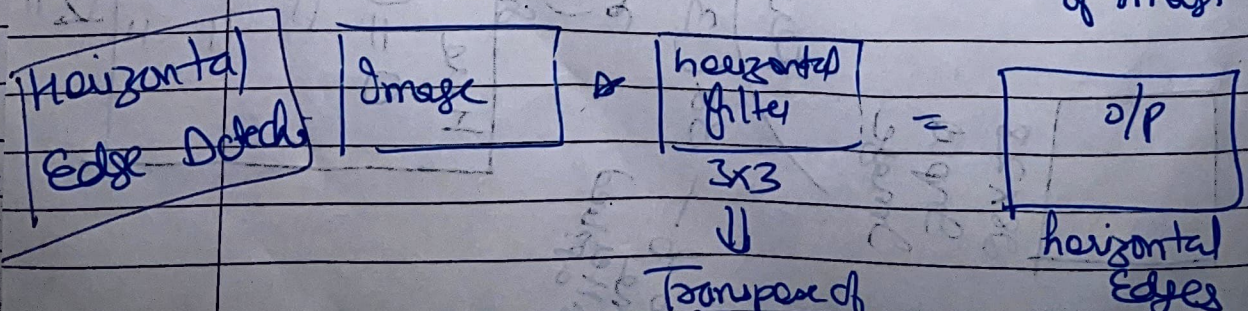
Size of feature Map

→ used to identify valuable features of object

Uses

- feature Detection (horizontally / vertically)
- Edges Detection





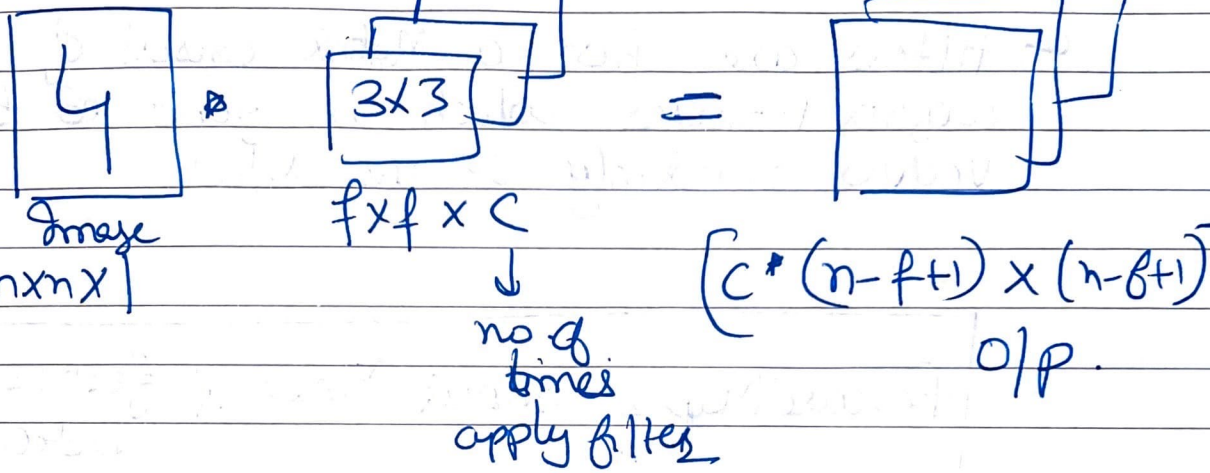
Date...../...../.....

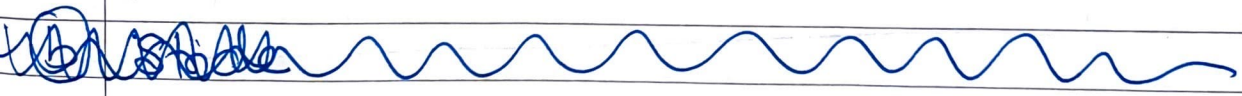
Page.....

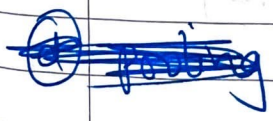
Transpose = $\begin{bmatrix} +1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

- Can apply single filter multiple time on particular image

eg.







Convolution Basic function

Date ... / ... / ...
Page ...

Convolution is the process of combining a function to produce the output of the other function.

- The Input Image is convoluted with the filters in CNN (kernel) & resulting in a feature Map

- filters are just a Matrix consist of weights + biases which is generated by vectors randomly in the N/W

> Feature Map = Input Image X feature Detector (kernel/filter)

Activation functions used in CNN

- ReLU
- Soft-Max

Relu: Rectified linear Activation function
- It is a piecewise linear function

> if I/P: +ve, O/P: I/P
> else O/P = 0

fully connected layer + flatten

Date...../...../.....
Page.....

At the end of CNN, FC layer of neurons, used to ~~train~~ classify images into distinct categories.

- Every activation unit in the next layer is coupled to all I/p's from this layer.
- If overfitting problem occured, than use dropout strategies to reduce it.
- It comprises the weights & biases together with the neurons & used to connect neurons b/w 2 separate layers

~~Convolution Activation~~

Limitations of CNN

① prone to overfitting, if not enough data.
② need more time to Train a CNN if have large dataset
③ Require specialized hardware (GPU) to perform training of dataset
- Complexity high
④ Require large amount labelled data

Q: Difference between feed-forward NN & RNN.