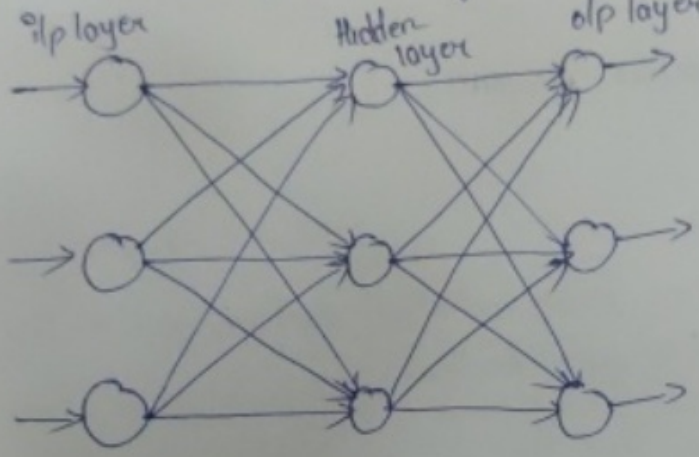
Deep learning

Feed forward neural networks:-

→ FFNN is one of the simplest forms of ANN

→ It is an artificial neural nlw where connect^n's blw the nodes do not form a cycle

→ Because i/p's are processed only in forward direction



→ It could be a single layer perceptron or multilayer perceptron

→ No feedback from o/p to i/p

→ We don't use it where order of seq^n does matter. It predict only for current node, does not depend on previous

→ No memory

→ i/ps are independent to each other

MSE IN FFNN:

to minimize the error for better predict^n

$$E_{total} = \frac{1}{\text{Total o/p}} \sum (\text{actual o/p} - \text{predicted o/p})^2$$

(or)

$$E_{total} = \frac{1}{n} \sum_{m=1}^{n} (o_m - \hat{o}_m)^2$$

Applicat^n

→ Data compress^n
→ pattern Recognit^n
→ Computer vis^n
→ Speech Recognit^n
→ Handwritten characters Recognit^n

Gradient descent

→ It is an iterative first order optimisat^n algorithm, used to find a local minimum/maximum of a given function.

→ This method is commonly used in M.L & D.L "To minimise a cost/ loss fun^n".

Ex:- Linear regress^n

→ starts with initial parameter values & iteratively updates them by taking steps in the direction of steepest descent (or ascent) of the fun^n.

→ Key steps involved are as follows:-

1. Compute the gradient → Compute the objective fun^n w.r.t Parameters.
   → gradient indicates the direct^n of steepest increase in the fun^n value
2. Update the parameters → adjust the parameter values by moving in opposite direct^n of gradient
3. Iterate until convergence-

repeat ↓

This condit^n is typically based on the magnitude of gradient or change in the val

UNIT-II

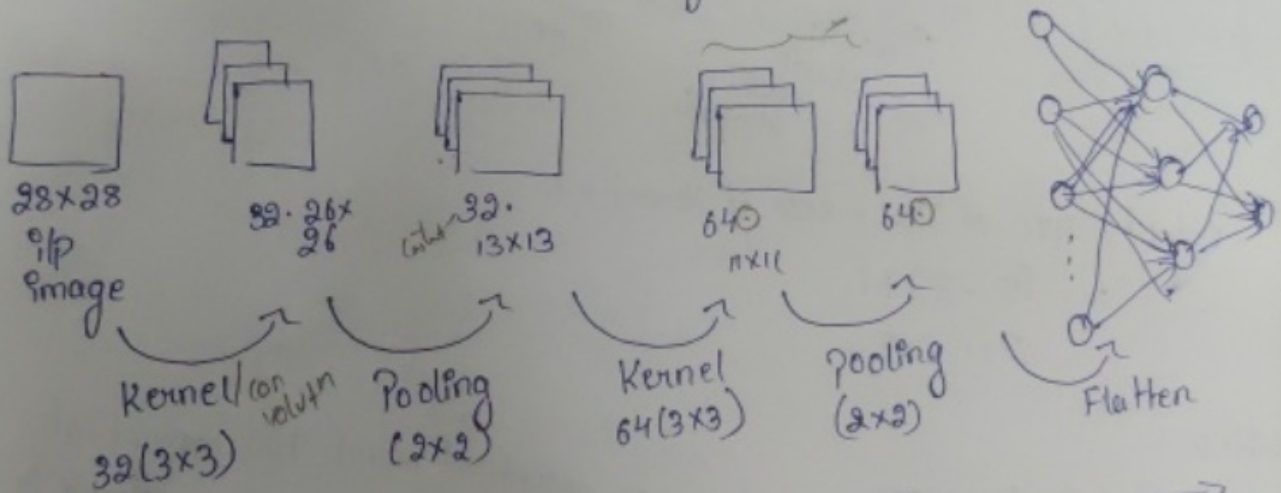
①

CNN

→ Convolutional N.N

→ cnn is a special type of feed forward neural nlw

→ mainly used in images recognitn
images classificatn
objects detectn
Faces Recognitn
audio, video recognition

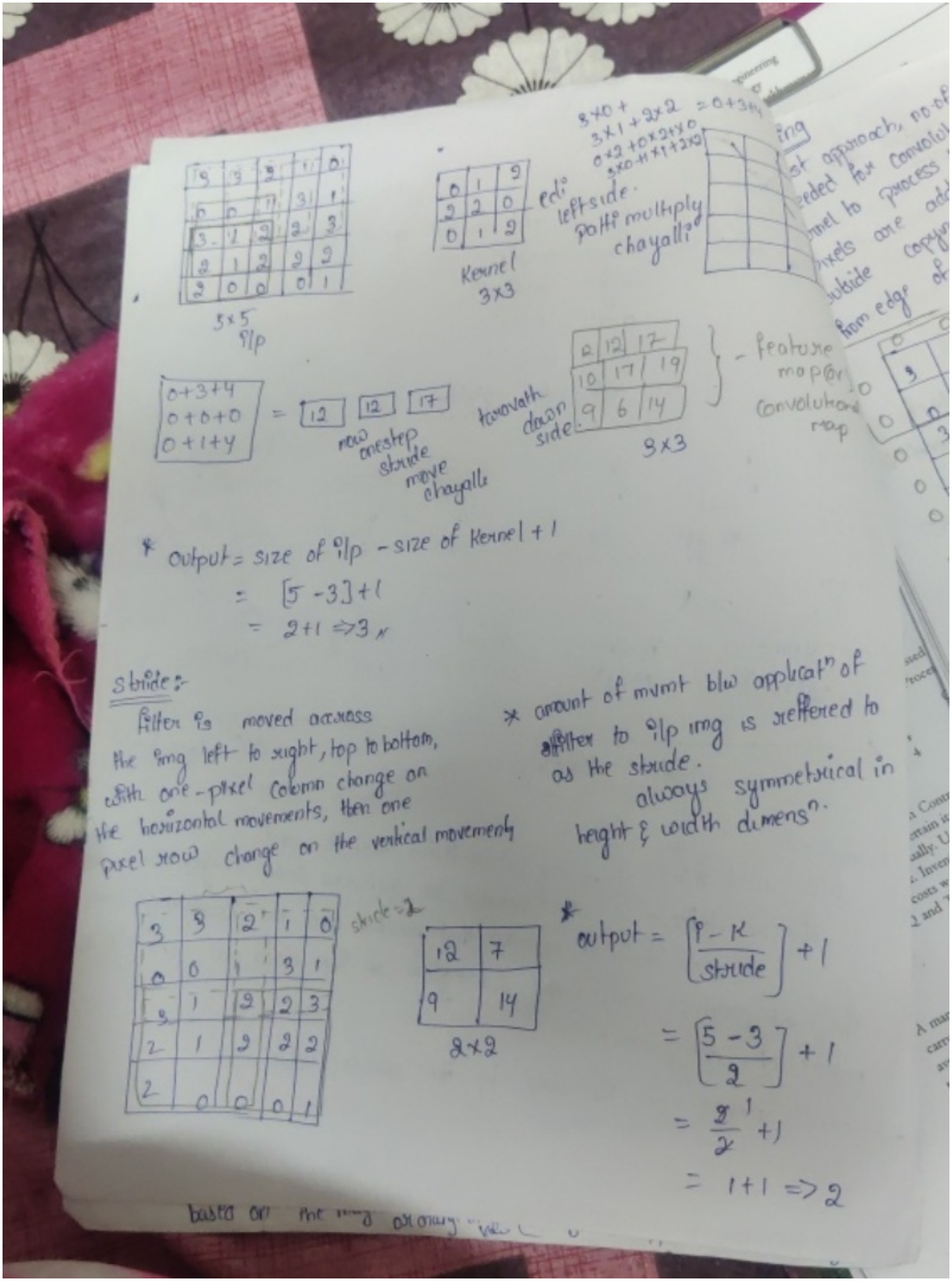


Kernel:-

also called filter
feature detectors
} is ntg but a filter that is used to extract the features from images

→ It is a matrix that moves over the i/p data, performs the dot product with the subregion of i/p data and gets o/p as matrix of dot product

→ Kernel moves on i/p data by the stride value


3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 1 2 2 2
2 0 0 0 1
5×5 i/p
0 1 2
2 2 0
0 1 2
Kernel 3×3
edi
leftside.
patti multiply chayalli
8×0 + 3×1 + 2×2 = 0+3+4
0+2+0×2+0×0
0×0+1×1+2×2
0+3+4
0+0+0
0+1+4
= 12 12 17
row onestep stride move chayalli
towards down side
0 12 17
10 17 19
9 6 14
3×3
- feature map or Convolved map
* output = size of i/p - size of Kernel + 1
= [5 - 3] + 1
= 2 + 1 ⇒ 3
Stride:-
filter is moved across the img left to right, top to bottom, with one-pixel column change on the horizontal movements, then one pixel row change on the vertical movements
* amount of mvmt blw applicat^n of filter to i/p img is referred to as the stride.
always symmetrical in height & width dimens^n.
stride=2
12 7
9 14
2×2
* output = [(i - k)/stride] + 1
= [(5 - 3)/2] + 1
= 2/2 + 1
= 1 + 1 ⇒ 2
basis on the img array

Padding

Best approach, no. of pixels needed for convolutional kernel to process the edge pixels are added onto outside copying the pixels from edge of image.

-> fix the border effect problem with padding

-> preserve same informatⁿ
-> also called border problem

* zeropadding to i/p

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

i/p

0	1	2
2	2	0
0	1	2

add 1 padding to all side to get same information

$$O = \left[\frac{I - K + 2P}{S}\right] + 1$$ (S: moves 1)

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

$$= 5$$

Reduce the dimensⁿ

Pooling:-

-> reqⁿ to down sample the detectⁿ of features in feature maps.

2 pooling methods,
i) avg pooling
ii) max "

* average pooling

$\frac{46}{4}$ =

11.5	14.25
19.5	14.0

* max pooling

2x2 (max = 14, 17)

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

o/p

14	17
11	19

ledu kalulli corudu chayali.

Flattening

involves transforming the entire pooled feature map matrix into a single col which is then fed to neural nlw for processing

Convolutional layers:-

→ which is used to extract the feature from the i/p dataset.

→ it applies a set of learnable filters known as the kernels to the i/p images.

→ filters/ kernels are smaller matrices usually 2x2, 3x3 or 5x5 shape.

i/p image computes with dot product

Visualizing convolutional neural nlw:-

importance of visualizing cnn model:-

1. understanding how the model works
2. Assistance in Hyperparameter tuning
3. finding out the failures of the model
4. Explaining the decisn to a consumer/end-user or a business executive.

Methods:-

preliminary

Activation based

Gradient based.

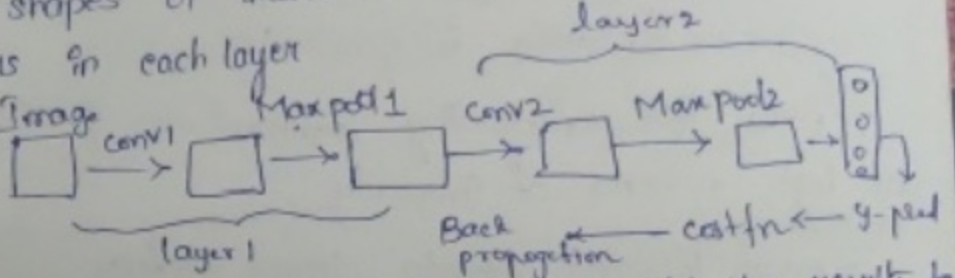
Fnb

Primary methods:-

* Simple methods which show us the overall structure of a trained model

* also print the shapes of individual neural n/w layers as well as the parameters in each layer

→ Keras



C.) Activation Based :-

* We can apply the filters to an i/p image & plot the results to see what our neural n/w is doing

* This enables us to comprehend the types of i/p patterns that trigger a specific filter.

Gradient Based :-

used to manipulate the gradients that are formed form a forward & backward pass while training a model

Regularization Techniques

↓

Used to address overfitting by directly changing the architecture of the model by modifying the model's training process

i) L2 regularization — also known as ridge regression
ii) L1 regularization
iii) Dropout regularization

In this, squared magnitude of the co-efficients (or weights) multiplied with a regularizer term is added to loss (or cost fun^n

Loss

$$\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij} b_j\right)^2 + \lambda \cdot \left(\sum_{j=1}^{p} b_j\right)^2$$

Loss term — Regularizer term

L1

lasso regression^n
absolute value

$$\sum_{i=}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij} b_j^2\right) + \lambda \cdot \sum_{j=1}^{p} |b_j|$$

y_i - labels
x_{ij} - features
b_j - weights
i = no. of training sample
λ = regularizer term

→ will not able to learn complex patterns from i/p

→ weight are same to untai- works best

→ helps to learn complex pattern from data

Dropout

prevents overfitting

Fully Connected:
↓
neurons are trained with entire data sets
→ leads to overfitting

Neural n/w is sparsely connected
→ only some neurons are active during the model training.