Convolution Neural Network (CNN)

CNN = Convolution + Neural Network

Neural Network part we know that

Now we will concentrate on Convolution part

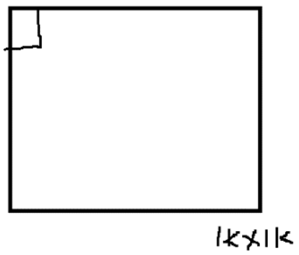
CNN is used for Object detection, Image or Video detection

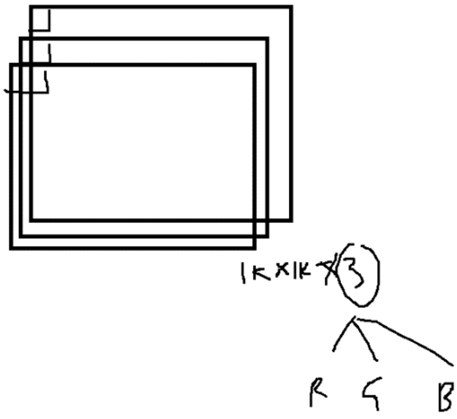
CNN very famous === So many developments are based on CNN only

Why CNN

Imagine we have an image with size 1000 * 1000 (Gray Imgage)

If it is a color image: RGB , the image size is : 1000 * 1000 * 3





For this shape: 1000 * 1000 * 3 , the number of input layer neurons required is 3M

after this if you give one hidden layer with 1000 neurons

How many parameters = 3M * 1000 + 1000≅3B

3B parameters in that starting Input + Hidden layer , to process this we required

so much big compute engines , practically It is not a good idea

why we need pass entire imgae , why can't we pass important features

can't the NN not find the output?

Idea: Convert Bigger image to small image which is having importnat features

this image will pass to the Neural network

Q2) How we get this smaller image from the bigger image

ECE : Fourier series , Fourier Transform

Convolution: Multiplying two Signals ==== > prodce the another signals

we multiply one image with another image ==== > feature map

Original image * filter or kernal = feature map

Q3) How to choose the filter :

In NN filter is nothing but weight matrix

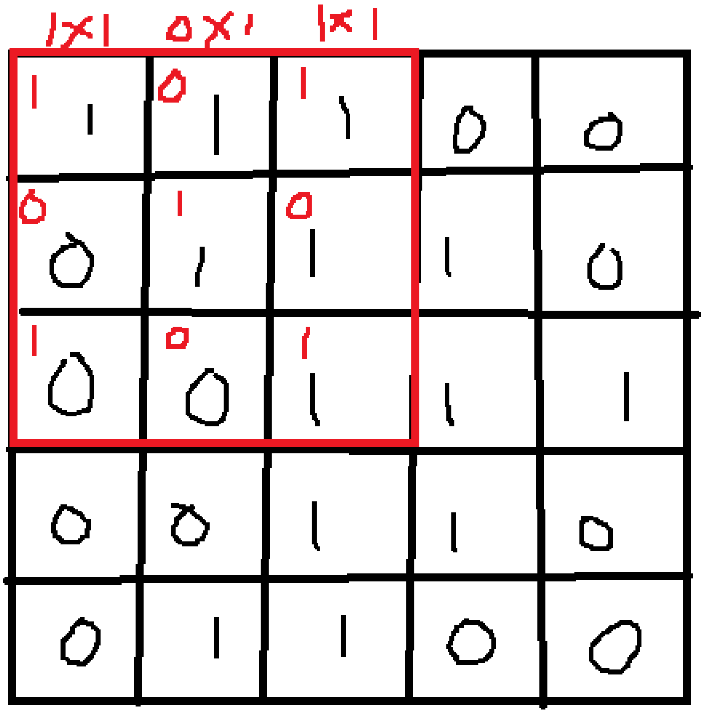
your weight matrix is good, it will multiply with original image === good features

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel



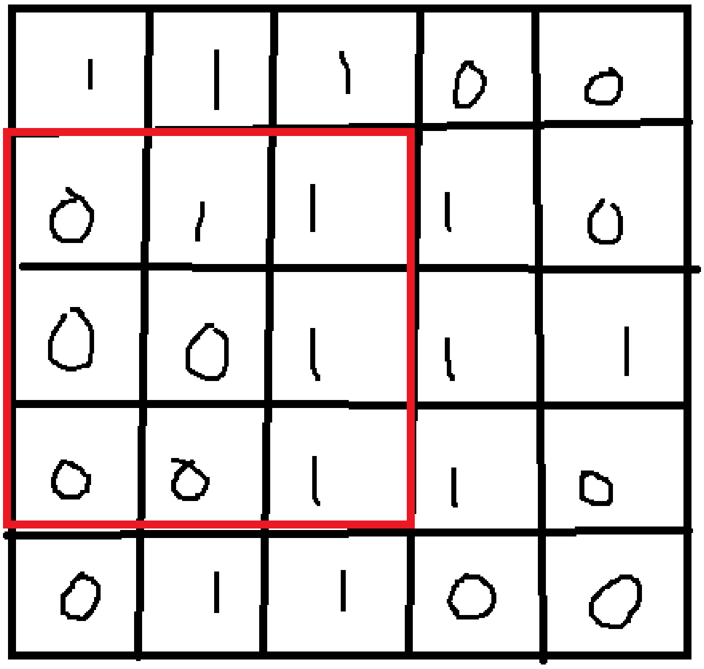
1 1	0 1	1 1	0	0
0 0	1 1	0 1	1	0
1 0	0 0	1 1	1	1
0	0	1	1	0
0	1	1	0	0

$$1 * 1 + 0 * 1 + 1 * 1 + 0 * 0 + 1 * 1 + 0 * 1 + 1 * 0 + 0 * 0 + 1 * 1 = 4$$

1	1 1	0 1	1 0	0
0	0 1	1 1	0 1	0
0	1 0	0 1	1 1	1
0	0	1	1	0
0	1	1	0	0

$$1 * 1 + 0 * 1 + 1 * 0 + 0 * 1 + 1 * 1 + 0 * 1 + 1 * 0 + 0 * 1 + 1 * 1 = 3$$

$$First\ slide\ to\ the\ right\ side\ = (4, 3, 4)$$



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

First slide to the right side $= (4, 3, 4)$

second slide to the right side $= (2, 4, 3)$

Third slide to the right side $= (2, 3, 4)$

We are multiplying Original image with filter , we are getting feature map

4	3	4
2	4	3
2	3	4

Original image size $5 * 5$, *Filter size is* $3 * 3 =$ *Feature map* $3 * 3$

Original image size $6 * 6$, *Filter size is* $3 * 3 =$ *Feature map* $4 * 4$

Original image size $7 * 7$, *Filter size is* $3 * 3 =$ *Feature map* $5 * 5$

Original image size $N * N$, *Filter size is* $F * F =$ *Feature map* $N - F + 1 \ * N - F + 1$

Original image size $5 * 5 * 3$, *Filter size is* $3 * 3 * 3 =$ *Feature map* $3 * 3$

*Original image size 5 * 5, Filter size is 3 * 3 = Feature map 3 * 3*

*Filter size is 3 * 3 = Feature map 3 * 3*

*Filter size is 3 * 3 = Feature map 3 * 3*

Draw back − 1

*Original image size 6 * 6, Filter size is 3 * 3 = FMap: 4 * 4*

*Original image size 6 * 6, Filter size is 4 * 4 = Fmap: 3 * 3*

*Original image size 6 * 6, Filter size is 5 * 5 = Fmap: 2 * 2*

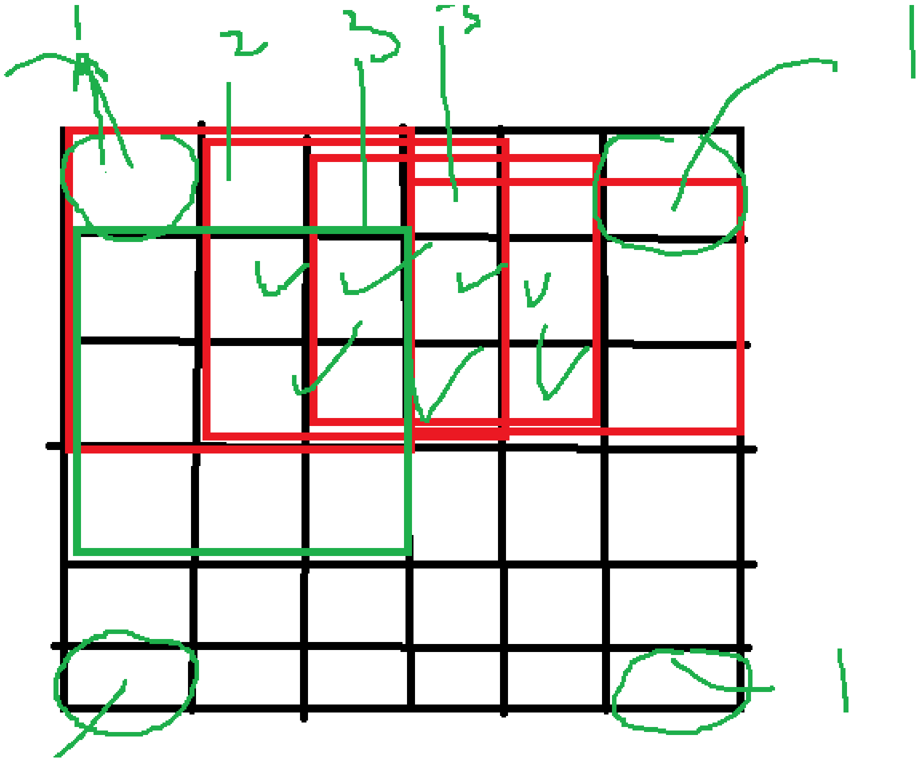
As filter size increases , resultant image (feature map)size is decreasing

which means number of pixels are decreasing , Input becomes less

Less input data is not good

CNN: Instead of sending total image, we will multiply a filter and

we will send a small image which is having importnat features



Draw back − 2: while comvolution edges are calucaulted only one time

with this way we might loss the information

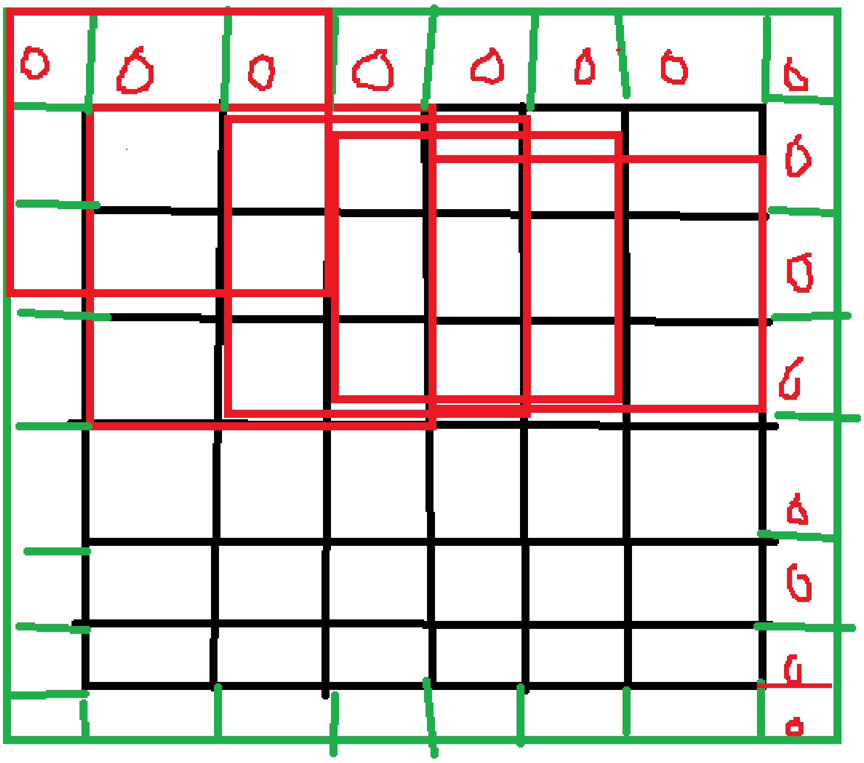
Goal:

we need to avoid image shrink

we need to avoid loss of edges information

Padding: Adding an extra layers

$(6X6) * (3X3) = 4 * 4$



$(6X6) * (3X3) = 4 * 4$

$padding = 1$

$(6X6) * (3X3) \quad Padding = 1 \; : \; 6 * 6$

$6 * 6 \; becomes \; 8 * 8 \; when\, we\, apply\, padding = 1$

$N = 6 * 6 \quad , \quad f = 3 * 3 \quad p = 1 \; :$

$(6X6) * (3X3) \quad Padding = 2:$

$6 * 6 \; becomes \; 10 * 10 \; when\, we\, apply\, padding = 2$

$N + (2p) - F + 1$

Original Image	Filter	Padding	Feature map
N*N	f*f	No	(N-f+1)*(N-f+1)
N*N	f*f	Yes =P	(N+2p-f+1)*(N+2p-f+1)

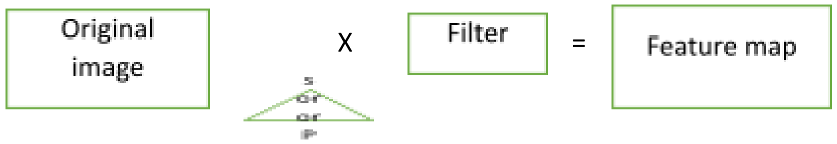
$Stride$:

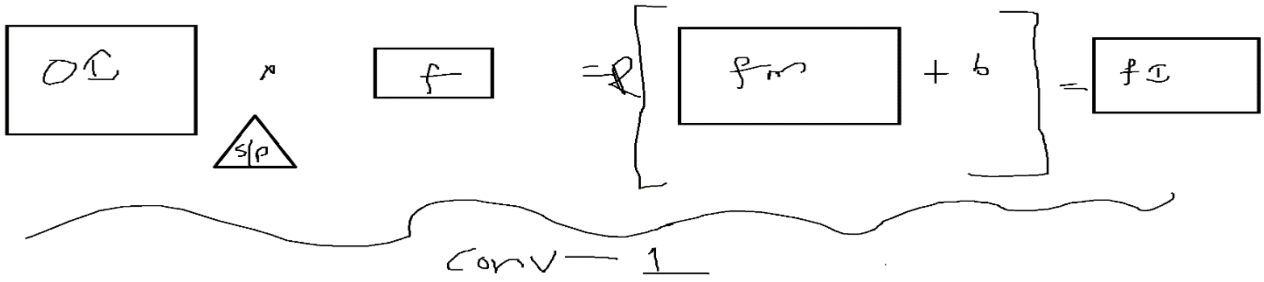
$To\; speed\; up\; the\; computaion\; process\;,\; the\; filter\; comvolve\; with\; original\; image$

$in\; the\; odd\; cases\; which\; means\; sliding\; will\; skip\; one\; step$

$stride\; is\; means\; step\; size$

$by\; default\;\; stride\; (s) = 1$





$$(6\ *\ 6\ )\ (3\ *\ 3) = (4\ *\ 4)$$

$$(6\ *\ 6\ *\ 3)(3\ *\ 3\ *\ 3) = (4\ *\ 4)$$

$Original\ image\ is\ \ 6\ *\ 6\ *\ 3\ ,\ \ Filter\ size\ =\ 3\ *\ 3\ *\ 3\quad Number\ of\ filter\ =\ 2$

$Feature\ map$:

32 * 32   multiply image with 4 * 4 = 29 * 29

32 * 32 * 3   multiply image with 4 * 4 * 3 = 29 * 29

32 * 32   multiply image with 4 * 4 (we are using 3 filters)[4 * 4 * 3] = 29 * 29 * 3

32 * 32 * 3   multiply image with 4 * 4 * 3 (we are using 3 filters)[4 * 4 * 3 * 3] = 29 * 29 * 3

32 * 32   multiply image with 4 * 4 (we are using 3 filters)[4 * 4 * 3] = 29 * 29 * 3

How many parameters are required? = 4 * 4 * 3 + 3 = 51

32 * 32 * 3   multiply image with 4 * 4 * 3 (we are using 3 filters)[4 * 4 * 3 * 3] = 29 * 29 * 3

How many parameters are required? 48 * 3 + 3 = 147

Finally One convolution layer

Original * Filter === > Relu(Fmap + bias) = image

Padding , stride, Number of filters

Pooling layer:

Ones the image we got, Pooling layers

some times we feel Covolution result image also have many pixel values

we want to reduce that, from the all pixels we want identify important pixels

imortant pixels having the more information

- Average pool

For example : 6 * 6 , pooling layer: 2 * 2

- pool

Convolution − layer1 = OI * Filter = Relu(Feamap + bias) = Image

Inuput: Originalimage ==> Convolution − layer1 + Pooling layer = Final image1

Inuput: Final image 1 ===> Convolution − layer2 + Pooling layer = Final Image2

Inuput: Final image − 2 ==> Convolution − layer3 + Pooling layer = Final Image3

+

Final Image3: 28 * 28 = 784 pixels falatten : Fully connected layer

Hidden layer

Output layer

Convolution Neural Netowrk

CNN is biggest development in DL

in ML we saved the model: Model. pkl file

in DL we can save the model: saving the weights

Our main goal: Find the suitable weights in order to reduce the error

weights file or parameter file

weights depends on model architecture

You can develop one good model, you save the weights

and share that weights to your friend , he will do the prediction === Transfer learning

we are not training the model from scratch, we will use already developed model weights

this is called Transfer learning

CNN ==== so many models

1)*LeNet*

2)*AlexNet*: *Geffry Hinton GodFather of AI*

3)*VGG*16 ,*Vgg*19,*Vgg*32

4) *MobileNet*

5) *ResNet* 50

6) *RCN*

7) *fastRcnn*

8) *Mask Rcnn*

9) *Yolo*

CNN python code

MobileNet code

Fine tuning is work

we will use those weights, and we will train the model on our data

we no need to worry about architecture

Directly take that model and pass the input output data

ask the model to learn