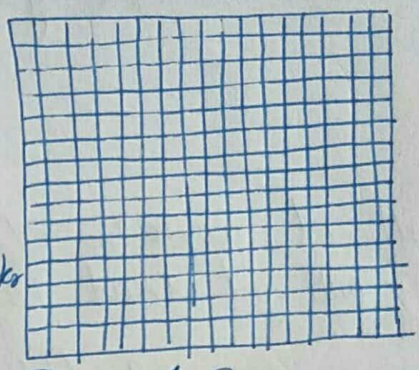
1) What are Images & pixels
2) Inspiration of Convolution neural Networks
3) How Edge detection Works
4) What is convolution
5) padding, strides, pooling
6) Conv. of RGB (color) Images
7) How to build CNN Network for Image classification?

1) Images & pixels

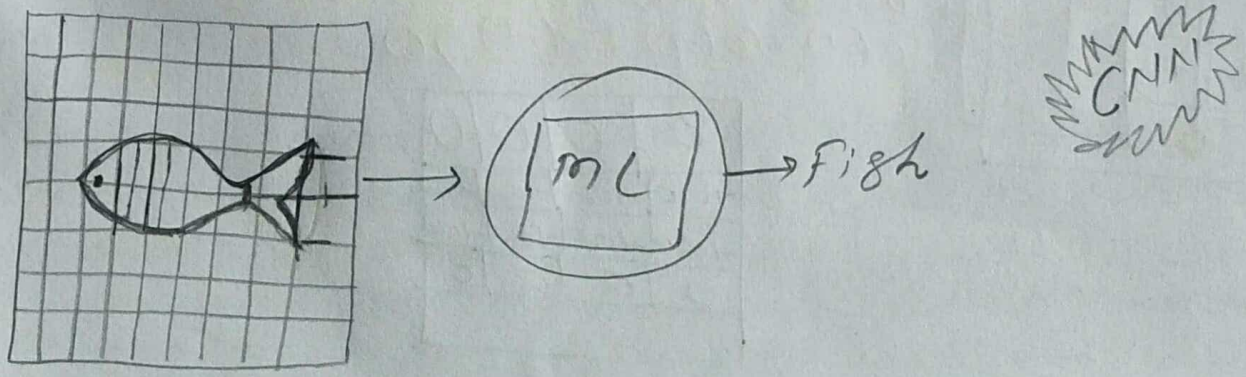
In computer everything is numbers Computer can't understand image as it is, ~~what~~ we have to convert this image in to numbers for computer to understand, Here pixels come in to this picture to help for converting picture to numbers.



Size of Image.

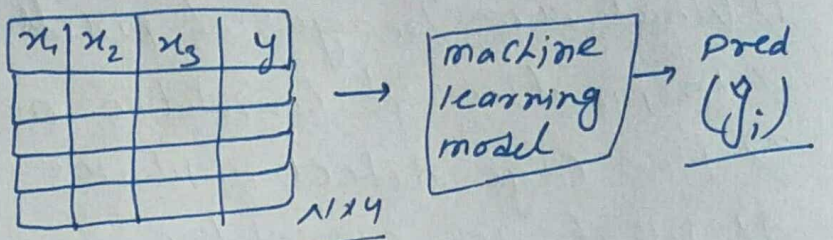
Pixels:- Here, representation of image, a smallest unit of picture is called pixels. Here, every box is a pixels. In computer every box will have a value that is called a pixel value. This value determine color of a particular block.

If a pixel NO's are less then there is a chances of mutiple color will come in to the one box/pixels. So if we increase the no. of pixels that will represent one color in one unit.



Traditional ML :-

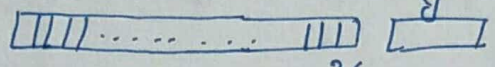
$D(x_i, y_i) \mid x_i \in R^3$, $y_i \in (0, 1)$



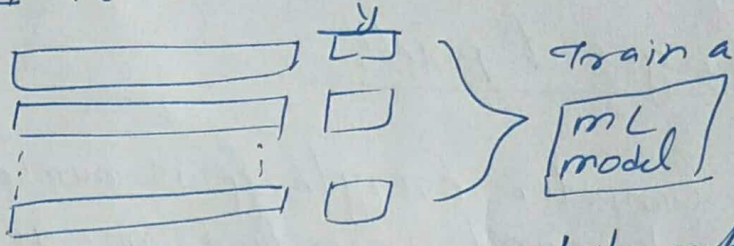
What if my current input will image Not a tabular data

one approach could be :-

1) Convert (6x6) image in to 36 length vector. because 6x6 = 36

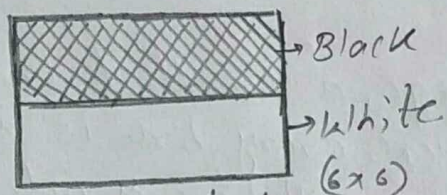


if we have multiple image means we have multiple 36 y →

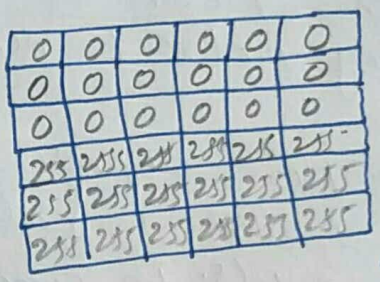


will it work ? → Technically it will work but performance you will get that is very poor.

for improving the performance CNN comes to picture.



Edge Detection

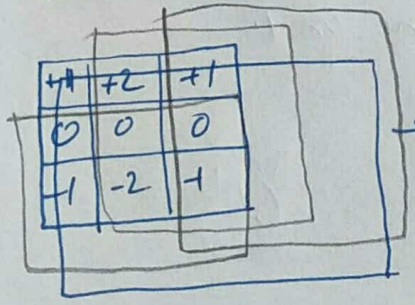


In computer pixel values varies b/w 0 to 255. this is a standard value. Black → 0, White → 255

convolution standard matrix →

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

3x3 ; Sign of convolution (*)



$0\times(+1) + 0(+2) + 0(+1) + (0\times0) + 0(0) + 0(0) + 0(-1) + (0(-2) + 0(-1) \Rightarrow 0$

0	0	0	0
-1020	-1020	-1020	-1020
-1020	-1020	-1020	-1020
0	0	0	0

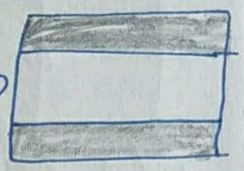
0	0	0	0
-1020	-1020	-1020	-1020
-1020	-1020	-1020	-1020
0	0	0	0

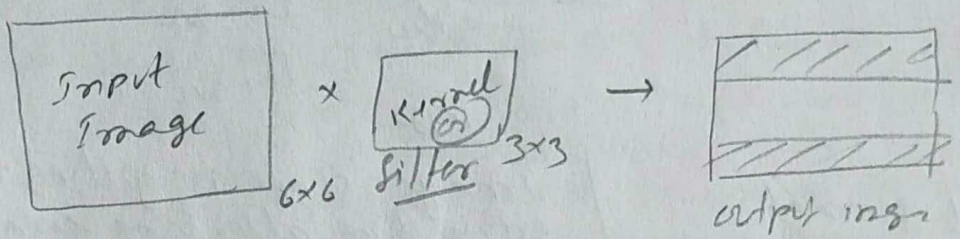
we have range only 0 to 255 but I am getting -1020 that will be a issue. so scientist remove minimum value with 0 and max^m value with 255

Here, min → -1020 → 0
max → 0 → 255

255	255	255	255
0	0	0	0
0	0	0	0
255	255	255	255

4×4





Sobel horizontal Kernal →

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

→ Detect horizontal Edges

Vertical Kernal

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

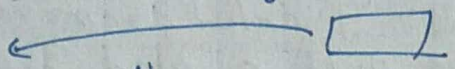
→ For detecting vertical Edges.

is there any other Kernal which can detect slant lines ⧅ depth information, alot of other stuff? → yes; we have lot of Kernals to understand the image. There are diff-diff Kernal with diff-diff shapes. we are using (3×3) only b'coz of industry standards.

→ By using a operation called convolution by using a concept called Kernal, we will be able to detect edges, But edges are not only thing needed to understand an image. we need to understand the color. we need to understand the depth, we need to understand lot of aspect of a given image.

take one example

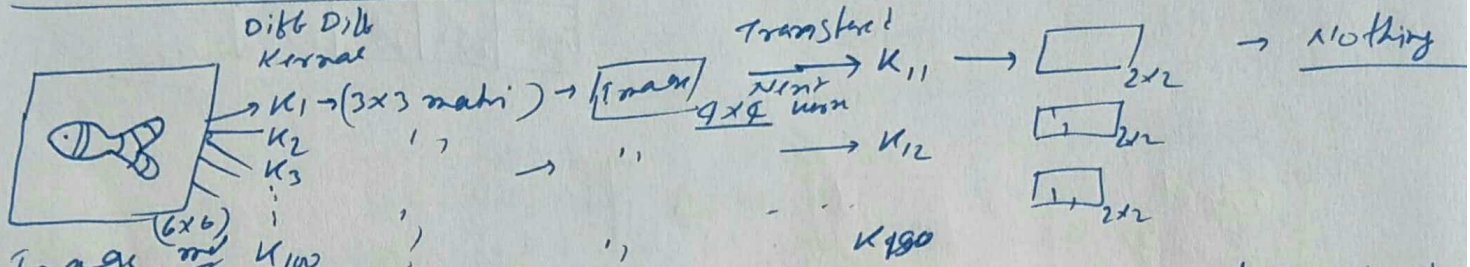
Visual cortex layer in human



v_1
v_2
v_3
v_4
:

let's assume we have an image and our brain see this image using layer V1 first this will detect edge and whatever output it will detect, it will pass to the layer V2 and it will detect depth (may be) next this will pass to the V3 that will detect color. etc at the end passed from multiple layer it can able to understand image

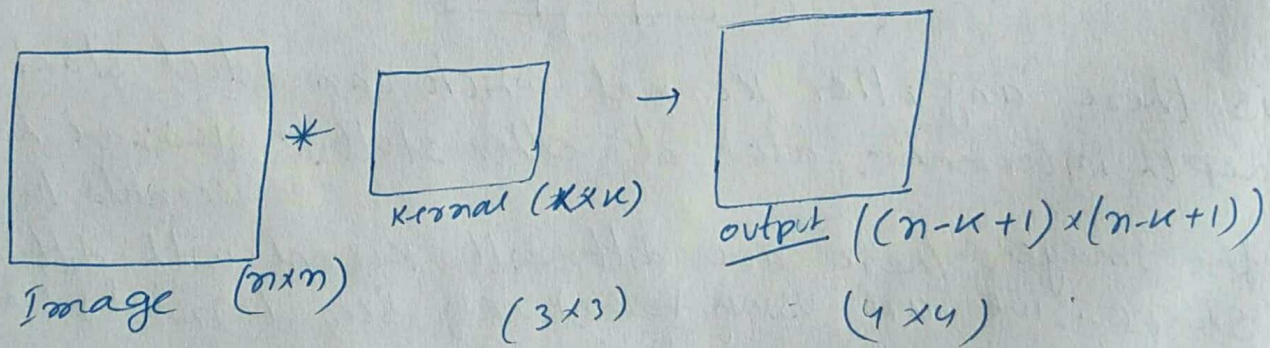
let's take one more example

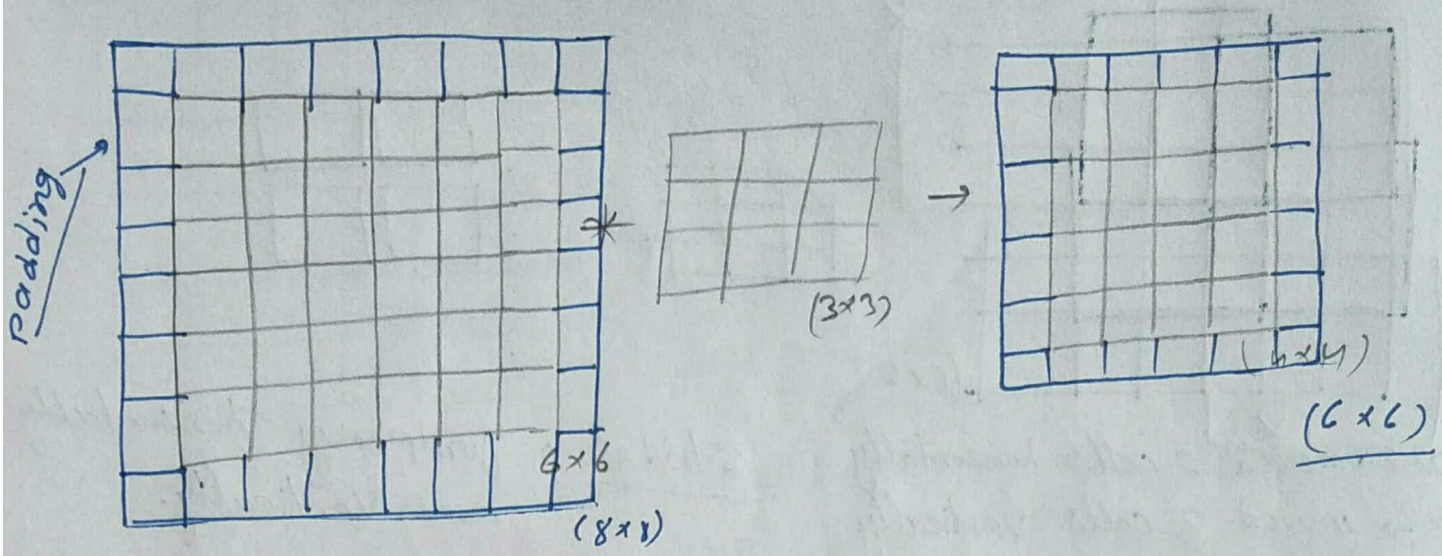


How many no. of kernal don't know, this is a hyperparameter let's tune it.

Image shape is getting reduced every time after using kernal. so at this point, we don't have any info. so here new concepts come padding. to resolve the issue.

So, given a image we pass it to the first layer of convolution the output pass next layer of convolution, and this will learn something new, and we will pass it sequentially but we noticed a problem that the image will get disappeared bcoz of reducing in the shapes



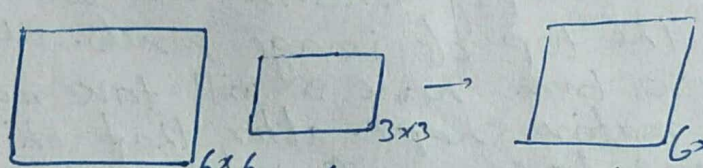


The prob. is what will be the value in the edges in input b'coz we created edges artificially.
if i fill the zero's or one's into the extra edge there is the chances of may be inside value is diff. so it we create diff edges that could be wrong. so wkt one more approach could be fill the nearby value which will not differentiate much in that image. so we are just expanding the image and not creating any artificial edges.

→ I have a 6×6 image, 3×3 Kernal, and I am getting image of 4×4. But my requirement is getting a image 6×6, I don't want to reduce the input size, so if i don't want to reduce my image input size, and I can't change my Kernal b'coz of this comes from research, so what I can do, expand the input size of image. and every given pixel value, we can either fill the nearby values or we can just fill zeroes, based on requirement Now my output will be the same as input image, that concept is called padding

Padding → (1,1) means 1 Row up & 1 Row down and 1 Column left and 1 column right.

P = 1
K = 3
n = 6

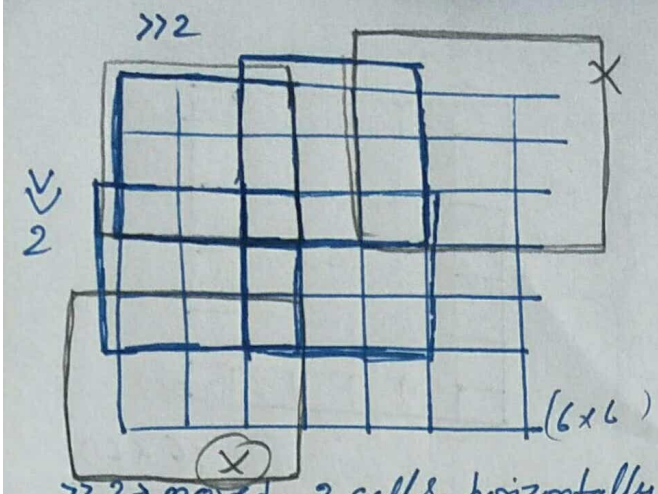


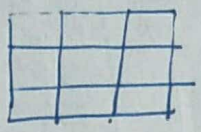
$(n-K+2P+1)$

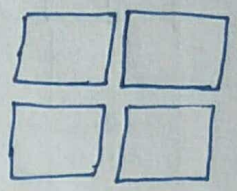
$(n-K+2P+1) \times (n-K+2P+1)$

$(6-3+2(1)+1)\ (6-3+2(1)+1) \rightarrow (6,6)$

Strides :→







>>2 → moved 2 cells horizontally

V 2 → moved 2 cells vertically

⊠ → can't jump b'coz we don't have data there. so we can't do convolution.

Strides → Jumping horizontally and Vertically.

if we are jumping Horizontally 2 blocks & vertically 2 blocks we get an image (2×2)

Here,

$n = 6$
$S = 2$
$K = 3$

So, formula: $\left(\frac{n-K}{S}+1\right) \times \left(\frac{n-K}{S}+1\right)$

Pooling :→

$n = 6$
$K = 3$
$S = (2,2)$
$P = (1,1)$

O/p Shape

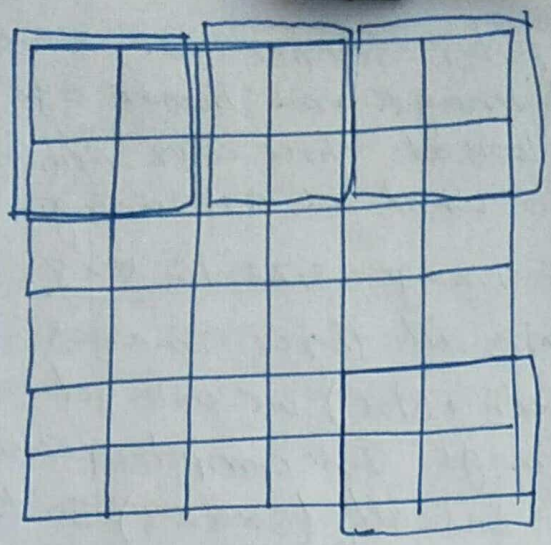
(i) Only Kernel → $(n-K+1)(n-K+1)$

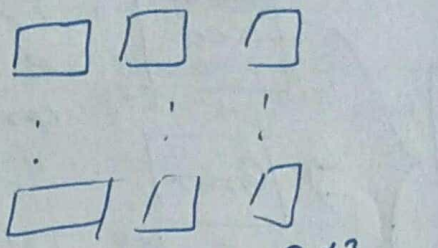
(ii) padding + Kernel → $(n-K+2P+1)(n-K+2P+1)$

(iii) Strides + Kernel → $\left(\frac{n-K}{S}+1\right)\left(\frac{n-K}{S}+1\right)$

(iv) padding + strides + Kernel → $\left(\frac{n-K+2P}{S}+1\right)\left(\frac{n-K+2P}{S}+1\right)$

Pooling :→ let's we have an image, and one pooling matrix with a shape of 2×2, it could be 3×3, 4×4, etc. In the pooling matrix we don't have any value, it's just an empty matrix. we will add pooling matrix upon the top of image pixels. all pixels will have some value, so from here it will take max^m value from the pooling matrix shape. after that shift pooling matrix to next and take max^m value again. This will repeated to all pixels.





3×3

While we are doing pooling we are doing half of the image. if we are using 2×2 pooling.

Why we are doing this?

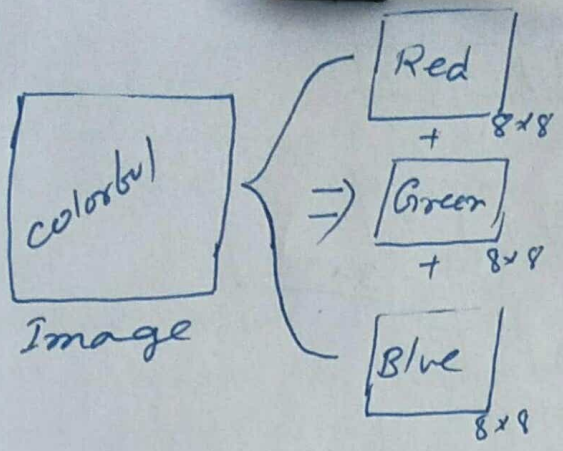
→ aisa ho sakta hai ki uss jagah ke sabhi cell me same color ho jo ki same value ko represent kar raha ho, to hame eski jarurat nahi hai jo ki same information ko multiple time carry kare, to hum aisa karte hai ki uss pooling matrix ke shape ke size me jaake max^m value ko rakh lete hain. ess'se kya hoga ki humlog encapsulate kar rahe hain max^m value/~~info~~ information from the particular matrix

Why only take max^m value? → B'coz humlog max^m information ko retain karna chahte hain from the given image.

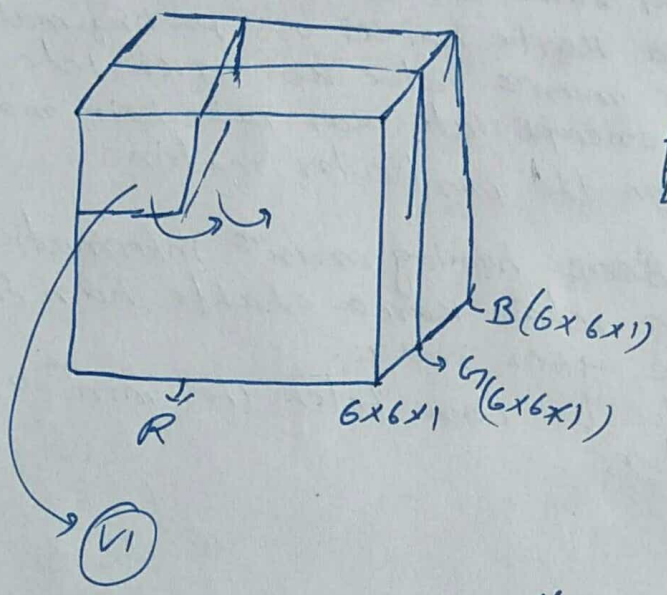
instead of fetching max^m value we can fetch the min^m and avg as well, based upon situation,

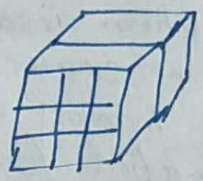
max pooling
Avg pooling

Agar hamare paas 500×500 ka image hain aur hum 3×3 ka convolution matrix use karte hain to hame bahut saare calculation karne parenge. Jisme time bhi bahut lagega. to esiliye hum pooling ka use karte hain, kyuki agar multiple box me same information hai to sabhi me se ek ko lekar aage badhte hain (ess'se calculation bhi jaldi hoga, time bhi kam lagega aur training time bhi reduce ho jayega) eske baad hame 250×250 ka matrix mila ab eske upar 3×3 ka Kernal use karenge.→

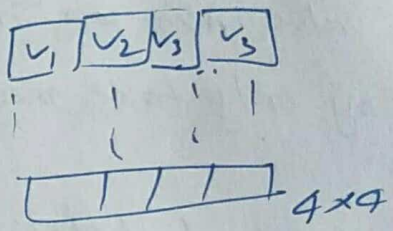


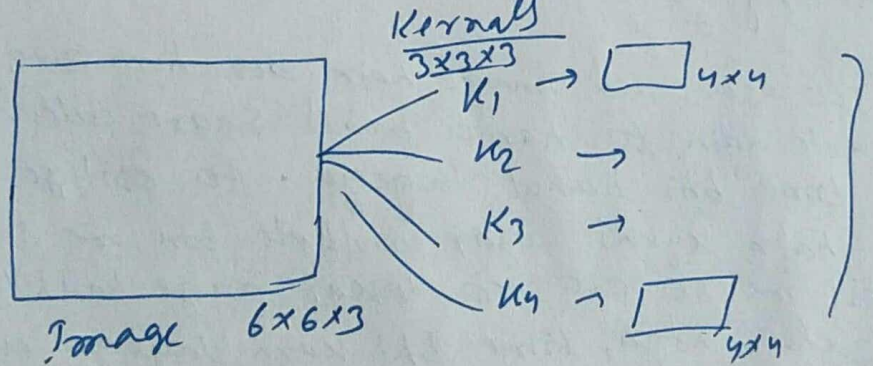
In computer science we have a colorful image and human eye can only look at three color, Red, Green & Blue. So what we are doing. Let's our image size is 8×8 if we mix all three images (Red + Green + Blue) we will get colorful image In computer colorful image is basically (8×8×3) so this 3 is called channels we have one channel for (Red, Green, Blue) if you are reading an image in python your image dimension will be (n×n×3) each of the channel of same input shape

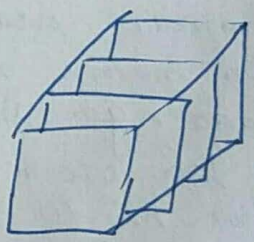


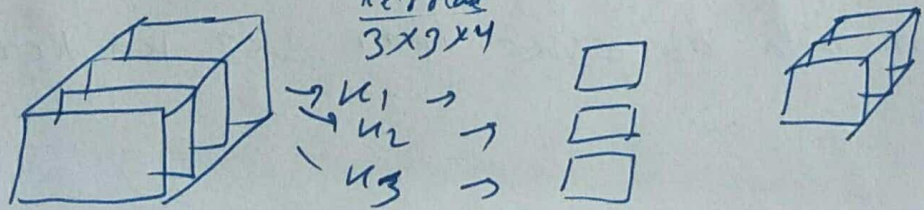


→ Layer 1 + Layer 2 + Layer 3
value = V_1









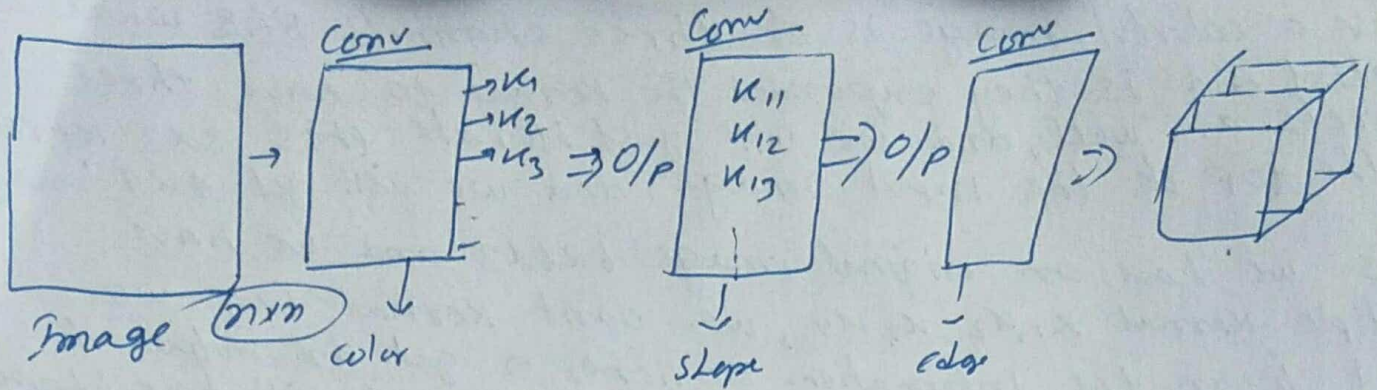
Because a colorful image is of three channels RGB what scientist did is they expanded the kernal to have three channels as well, and we will just iterate this 3×3 kernal on the top of the input image, and we will get 4×4 image

let's we have an original image 6×6×3 and we have multiple kernal K_1, K_2, K_3, K_4,, we want kernal b'coz we want learn the information. so here, 9 get an output of 4×4 after doing kernal and that kernal will have shape 3×3×3 and our output will 4×4, now we will add all the images side by side so we have now 4×4×(4) → no of kernals

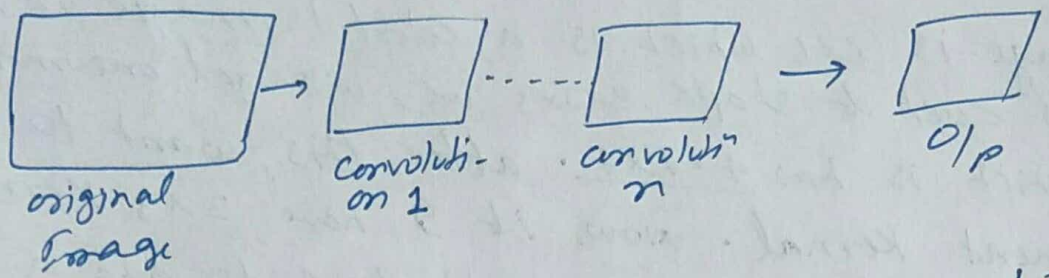
So, if my image is 6×6 which is a colorful image if i do 4 kernals each of shape 3×3×3 we, will get one more 4-D matrix which is has 4×4×4. after this, want to pass to the next kernal. now, if 9 have 3×3 kernal b'coz 9 have 4-channels here 9 would have to add a kernal 3×3×4 then only will be able to do convolution let's assume kernal K1, K2, K3 each one have 3×3×4 all give one output. now these three images we will add side by side here it will become n×n×(3) → no. of kernals. This is how convolution works if you are having a RGB image

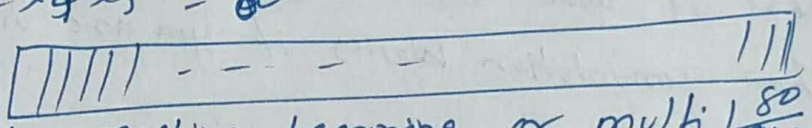
Why do we need all of this? how we are going to detect?

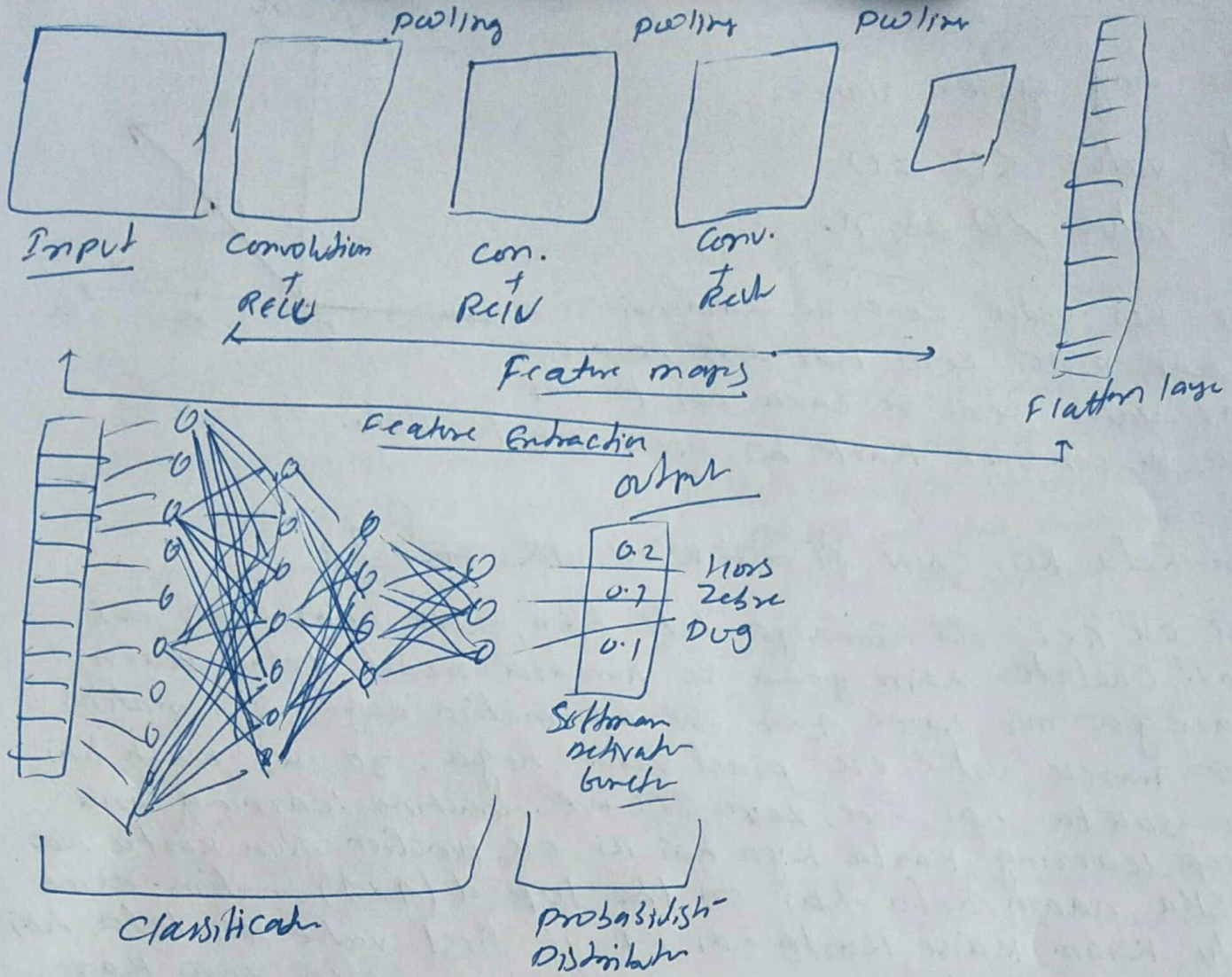
let's we have an image (n×n) we pass it to a convolution layer. convolution layer is basically kernals, bunch of kernals ($K_1, K_2, K_3, \ldots$) it could be anything. whatever the output we get we learned something here, and this output we will pass into the next convolution layer. we have some another kernals so these are diff- kernals. and this kernal shape depend on the output of previous output. on this output we will recon previous operation. at last we will get bunch of images their shape might be $(n_1 \times n_1 \times k_1)$. so after doing this convolution we are having some output as an image.



Now why do we need this, what we are going to do with this output that is where now we will apply ml concepts

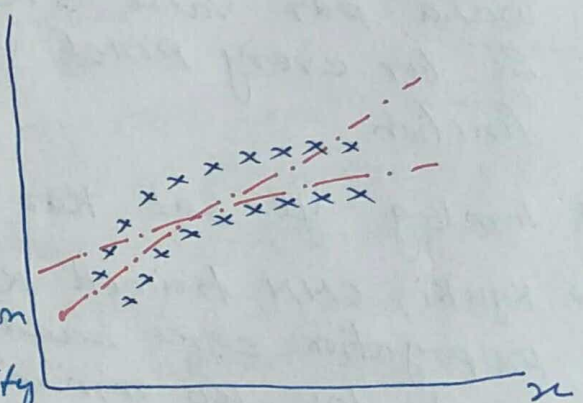


So I have an original image I pass to series of convolutions and each of the convolutions do padding, strides pooling etc. but at the end we will get this output. Now this output is having 4×4×5 shape. Total No. of elements is 4×4×5 = 80 → so if I stretch this in to a 80 vector  Now this I will pass it to machine learning or multi layer perceptrons. to train my model. So this is basicaly capturing the all information. and able to convert in to vector. This is an image embedding. embedding is basically we are encapsulating the information from my original image by passing through a series of convolution, able to get to this vector. So this vector is having all of the information. that is being learned, so this is a actual representation of my original image. Now I will pass this vector to a ml model. I can predict so compare to the first one in which we have directly image in to vector by not doing any convolution, their accuracy and after doing convolutions we do the modelling on the top of this we get an accuracy of is bit better than previous one. and then they moved to DL/MLP that is how you train a ml model



Activation function

Here, 9 cannot classify both the class using ~~straight~~ straight l. line we need a curve so, here, Activation ~~for~~ function comes in to the picture.



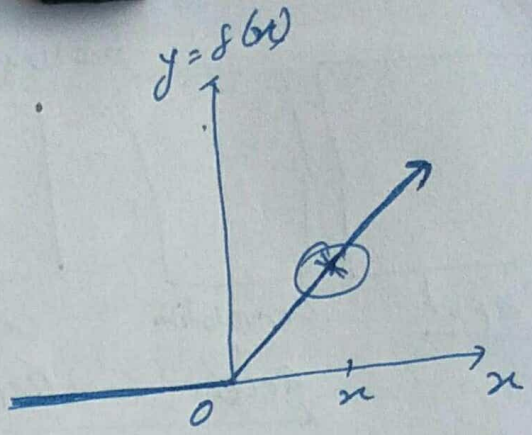
So the main importance of activation function is to introduce Non-linearity becoz in real life most of the data that you see is of Non-linear in nature.

there are some famous activation function are

(i) Sigmoid (ii) Relu (iii) tanh (iv) Leaky Relu

Relu:-

for any given value
if value $<0 = 0$
& value >0 as x



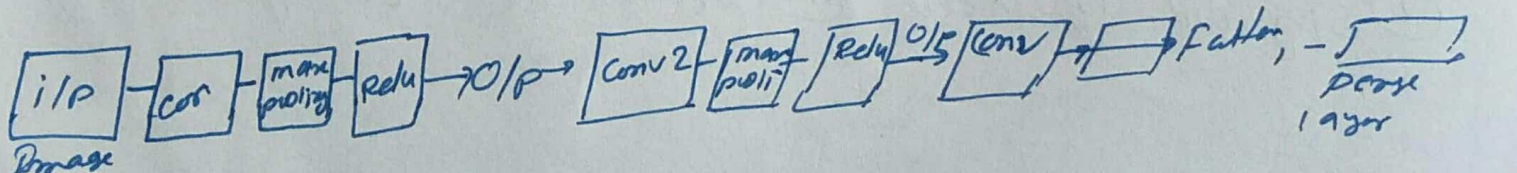
Agar koi value zero se kam hai to aap usko zero kar dete ho aur agar value zero se bara hai to aap usko x consider karte ho according to Relu.

Ham Relu KO CNN में क्यों use कर सकते है?

एक 6×6 का Image lete hain, eske upar 3×3 का conv. chalate hain yaha se hamara model kuchh learn karega aur hame 4×4 का एक matrix dega jis matrix का हरेक value एक pixel value hoga, जो कि kuchh bhi ho sakta hai, -ve, zero (or) +ve. machine learning and deep learning karta kya hai ki ek function Run karta hai jiska naam Relu hai on the top of (4×4) matrix. esme Relu kaam kaise karta hai कि ye first value को leta hai agar eska value zero se kam hai, to esko zero kar dega. phir next value par chala jata hai agar waha par value -ve hoga to usko zero karega aur agar waha par value 5 hai to usko 5 hi chhor dega. So for every pixel value we will apply the Relu function

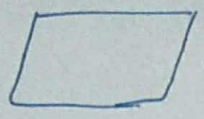
Q hum log ye sab kar kyu rahe hain?

→ Kyuki, CNN trained kaise hota hai, through forward propogation eske baad loss calculate karte hain, aur eske baad loss को minimise karte hain using backward propogation. esi process me hum-log bahut saare differentiation karte hain, to hamare bahut saare mathematical complexity kam ho jata hai, aur process bhi fast ho jata hai



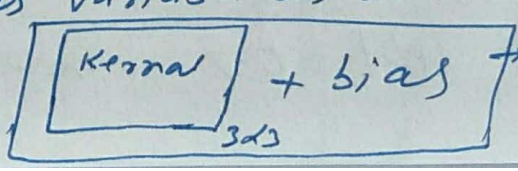
cont

I have an image, I have to classify this as a fish

 → fish.

I have my input image, and i pass pass it to only one convolution layer which is 3×3 with only one kernal and i get output, after getting output i will expand it to flatten, and this i will train a m/l model and I am getting a prediction $\hat{y}_i$. so if this kernal learn something then this flatten vector should be able to differentiate what is a fish. b'coz this output at the end encapsulates the information that is present in the given input image so, let's say in the initial stages we don't know what is the value inside of the kernal, so what you did is you randomly give values $(0,1,2,3,4,5,6,7,8)_{3\times3}$ and you apply convolution on the top of the i/p image and at the end you got the kernal, and at the time of predicting that image as a cat. so what happend, your kernal values are not learning that it is a fish, we are having error. we, will say to model that you made mistake, so what model will do, it will change the kernal values. so in the next iteration, with the same input image you are changing the kernal value after that at the time of classifying the image you classify as a fish. that means this kernal is working for us. on this input image to classify a given image. and how it is learning it by passing this image multiple time to this kernal and calculating this output and output is not what we expect, then it will go back and update this values

Bias in CNN:→ In CNN we have kernal so for each kernal, I will add one variable called bias so this could be any value. and this bias will shift the values of the kernal properly based upon the input values, so wherever you pass a kernal by default it is having bias variable as well. so all of them are trainable parameters.:



OS.listdir('path of dir') → cat folder ka path denge to saare cat ka image ka list load kar lega.
cat
dog

for print 9 image randomly

random.sample(os.listdir('path of folder'), 9)

→ result is not an image that is only file name.

Jb 9 want to read this as an image then 9 have to get the absolute path

what is absolute path → /content/dogs_vs_cats/train/cats/cat.0.jpg

this is the absolute path of certain image.

we need to join

/content/dogs_vs_cats/train/cats

+

random image of dogs/cats that we have

/cat.0.jpg

SO that is where we need to run

image_path = OS.path.join(directory_path, image_file)

Now 9 will do the

Image.open(image_path)

OS.Walk('directory path') → 91's a looping function which will go through all the values of all the files inside of your given directory. ~~so inside that~~

what OS.walk will do is → it will go through each and every directory and it will list down all the content that is available in this directory.

ImageDataGenerator :- load images in batches, batch could be any no.

validation_split = 0.2 → means → Train → 80%.
Validation → 20%.

flow_from_directory

directory → from which directory I want to get the data

target size → while getting the data convert each image in to the shape (256, 256) → so whatever the size of the image that it have it will convert (256, 256)

color_mode = 'rgb'

class_mode → 'categorical', 'binary' → if you have binary classification use binary else categorical for multiclass classification

shuffle = true → while loading the data we will shuffle the image.

batch_size = 32, 100, 200, etc

if our training accuracy is more compare to validation accuracy → overfitting.

if training accuracy is less & validation accuracy is also less → underfitting.

Model checkpoint:→ Epoch 1 validation loss improved and model got saved this is called model checkpoint.

Verbose = 1 → whenever it is saving everytime it is saving it will print, if

Verbose = 2 → For every two time it will print, but it will save in the backend. it will not clutter your output

What are non-trainable parameters:→

Let's say we have max-pooling layer there we have nothing actually, we don't train anything, we just decreasing the value, those are called non-trainable parameters

Data Augmentation :→

Data Augmentation is a technique to increase the no. of images by creating modified version of existing data through methods such as rotation, translation, scaling or flipping with the aim of improving model generalisation & performance.

restore_best_weights = True

↳ Jab bhi hum no. of epoch chalate hain. tab hum harek epoch par model ko save bhi karte jaate hain, wala par hume accuracy aur validation loss bhi milta hai. Jab hamara validation loss me jyada ka improvement hona band ho jata hai. tab hum pichle waale epoch me jaate hain jaha par hamara validation loss aur accuracy best point par tha. aur hum usko save kar lete hain.

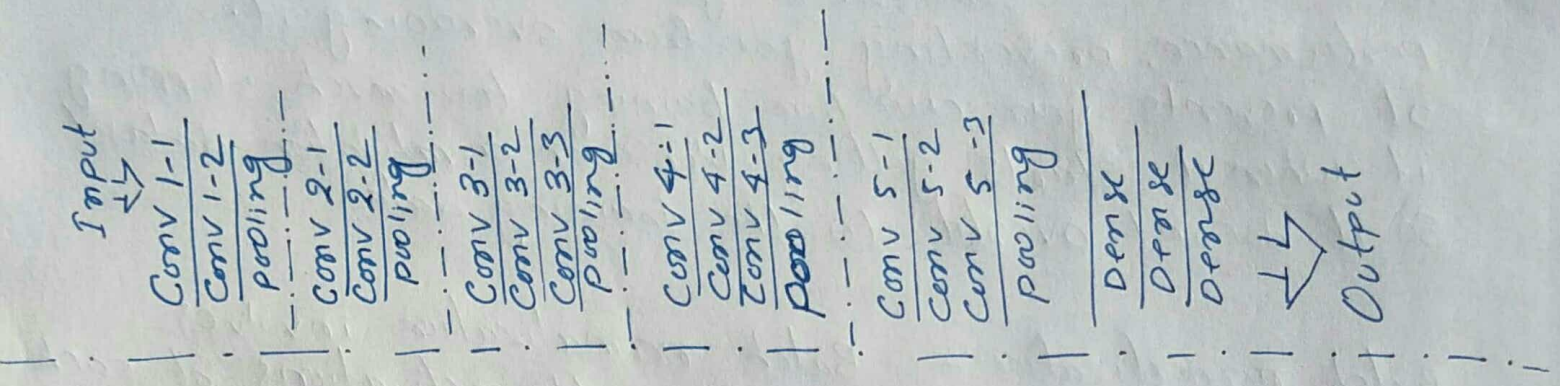
Transfer learning

Transfer learning in CNNs refers to the practice of taking a pretrained model (Convolution Neural Network) which has been trained on a larger data-set for a specific task like image classification, and then adapting it to a different task with limited data by fine tuning its parameter

eg:→ you can take a pretrained model (CNN) trained on imageNet image classification and fine-tune it on a smaller dataset for a specific classification task like identifying different species of flowers. This approach saves time and computational resources while often achieving good performance especially when the new task is related to the original task.

VGG16 Architectures.

VGG16 is object detection and classification algorithm which is able to classify 1000 images with 1000 different categories with 92.7% Accuracy. it is one of the popular algorithms for image classification and is easy to use with transfer learning.



16 trainable layers

Conv Layes - capture Spatial info

FC Layer - Classification Task.

Convolution layers of 3x3 filter with stride 1 and always used the same padding maxpool layer of 2x2 filter of stride 2.

Conv-1 Layer has 64 number of filters.
Conv-2 has 128 filters.
Conv-3 has 256 filters
Conv-4 has 512 filters.

Three Fully connected (FC) layers follow a stack of Convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classificiation and thus contains 1000 channels (one of each class). The final layer is the softmax layer.

Model_checkpoint:→ you are taking snapshots of your CNN's progress during training, it ensures you don't lose your best performing model in case something goes wrong during training, so you can always go back to your best check points.

Early Stopping:- Early stopping is like having a coach who tells you to stop practicing a skill when you've reached your peak performance, preventing you from overdoing it. It prevents your CNN from training too much, stopping it when it starts getting worse on a validation dataset so it doesn't overfit to the training data.

Batch_normalization:→ Batch normalization involves normalizing the activations of each layer in a neural network by adjusting and scaling them to have a mean of zero and a standard deviation of one, based on the mini-batch of data during training.

(i) Stability:→ It helps stabilize the learning process by reducing the chances of activations becoming too large or too small, which can slow down or destabilize training.

(ii) Faster training:→ By normalizing activations, it allows the neural network to converge faster during training, reducing the no of epochs needed to reach a certain level of performance.

(iii) Regularization:- Batch_normalization acts as a form of regularisation, reducing the need for other regularization technique like dropout which can sometimes slow down training.