Convolutional Network Neural Network (CNN):

⇒ it is a deep learning algorithm which can take in an i/p image, assign importance (learnable weights & biases) to various aspects/objs in the img and be able to differentiate one from the other.

⇒ the pre-processing required in a convNet is much low as compared to other classification algorithms.

⇒ while in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

⇒ The architecture of a convNet is analogous to that of the connectivity pattern of neutrons in the human brain and was inspired by the organization of the visual cortex.

⇒ Individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. ~~respective~~

⇒ A collection of such fields overlap to cover the entire visual area.

$$\begin{bmatrix} 1 & 1 & 0 \\ 4 & 2 & 1 \\ 0 & 2 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 1 \\ 0 \\ 4 \\ 2 \\ 1 \\ 0 \\ 2 \\ 1 \end{bmatrix}$$

flattening of a 3x3 img matrix into 9x1 vector.

→ an img is nothing but a matrix of pixel values

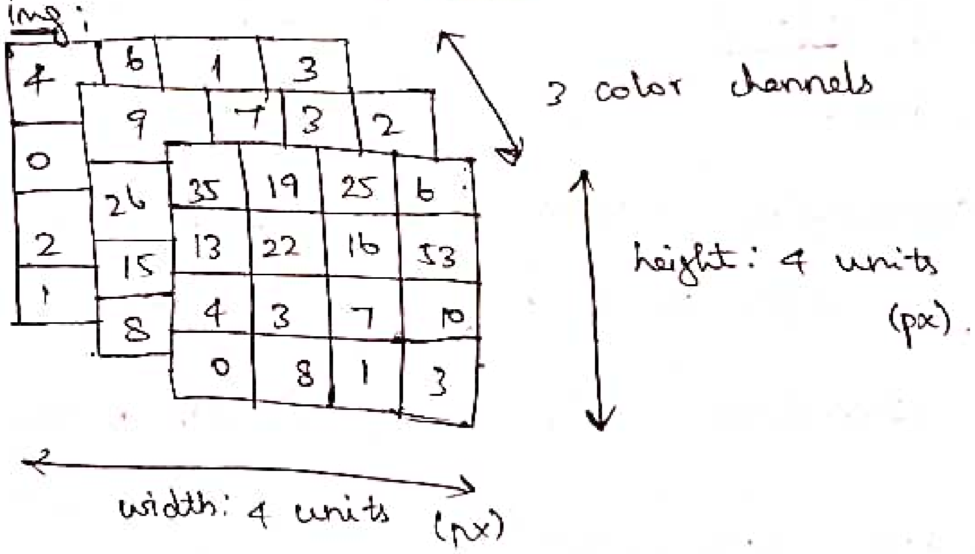
→ In cases of extremely basic binary images, the method might shows an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex imgs having pixel dependencies throughout.

→ A convNet is able to successfully capture the spatial & temporal dependencies in an img through the application of relevant filters.

→ the architecture performs a better fitting to the img dataset due to the reduction in the no. of parameter involved and reusability of weights.

→ the network can be trained to understand the sophistication of the img better.

<u>i/p</u> <u>img</u>:



4X4X3 RGB img.

→ we have an RGB img which has been separated by its three color planes - red, green, blue.

→ there are a no. of such color spaces in which imgs exist - greyscale, Rgb, hsv, CMYK, etc...

→ You can imagine how computationally intensive things would get once the images reach dimensions, (7680 × 4320)

→ The role of the convNet is to reduce the imgs into a form which is easier to process, without losing features which are critical for getting a good prediction.

→ This is imp. when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

Convolution Layer – The kernel

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

image

4	3	4
2	4	3
2	3	4

convolved feature

Convoluting 5×5×1 image with a 3×3×1 Kernal to get a 3×3×1 convolved feature.

→ Image dimensions = 5 (height) × 5 (breadth) × 1 (no. of channels. eg-RGB)

→ the green section resembles our 5×5×1 i/p img, I. The element involved in carrying out the convolutional operation in the first part of a convolutional layer is called the kernal / filter, K, represented in the color yellow.

→ we have selected K as a 3×3×1 matrix.

Kernel / Filter, K =
$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

⇒ the kernel shifts 9 times because of stride length =1 (Non-strided), every time performing a matrix multiplication operation between K and the portion P of the img over which the kernel is hovering.

⇒ the filter moves to the right with a certain stride value till it parses the complete width.

⇒ moving on, it hops down to the beginning (left) of the img with the same stride value and repeats the process until the entire img is traversed.

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	144	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

input channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	154	159	163	168	...
0	155	153	153	155	162	...
...	...	...	...	...	...	...

input channel #2 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	159	152	152	157	167	...
...	...	...	...	...	...	...

Input channel #3 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel channel #2

0	1	1
0	1	0
1	-1	1

Kernel channel #3.

output

-25	466	466	475	...
295	787	798	812	...
				...
				...
...	...	...	...	...

Convolution operation on a $M \times N \times 3$ img matrix with a $3 \times 3 \times 3$ kernel.

⇒ In case of imgs with multiple channels (RGB), the kernel has the same depth as that of the i/p img.

⇒ matrix multiplication is performed between kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel convoluted feature output.

⇒ the obj of the convolution operation is to extract the high level features such as edges, from i/p img.

⇒ ConvNets need not be limited to only one convolutional layer.

→ conventionally, the first convLayer is responsible for capturing the Low-level features such as edges, color, gradient orientation, etc.

→ with added layers, the architecture adapts to the high-level features as well, giving us a network which has the wholesome understanding of img in the dataset, similar to how we would.

→ there are 2 types of results to the operation. one in which the convolved feature is reduced in dimensionality as compared to the i/p, and the other in which the dimensionality is either increased or remains the same.

→ this is done by applying Valid padding in case of the former, or same padding in the case of the latter.

→ when we augment the 5x5x1 img into a 6x6x1 img and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. hence the name - <u>same padding</u>.

→ if we perform the same operation without padding, we are presented with a matrix which has dimensions of the kernel (3x3x1) itself - <u>valid padding</u>.

Pooling Layer:

3.0	3.0	3.0
3.0	3.0	3.0
3.0	3.0	3.0

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

3×3 pooling over 5×5 convolved feature.

→ Similar to the convolutional layer, the pooling layer is responsible for reducing the spatial size of the convolved feature.

→ this is to decrease the computational power required to process the data through dimensionality reduction.

→ furthermore, it is useful for extracting dominant features which are rotational and positional invarient, thus maintaining the process of effectively training of the model.

→ there are two types of pooling :-

*) Max pooling

*) Avg pooling.

Max Pooling:

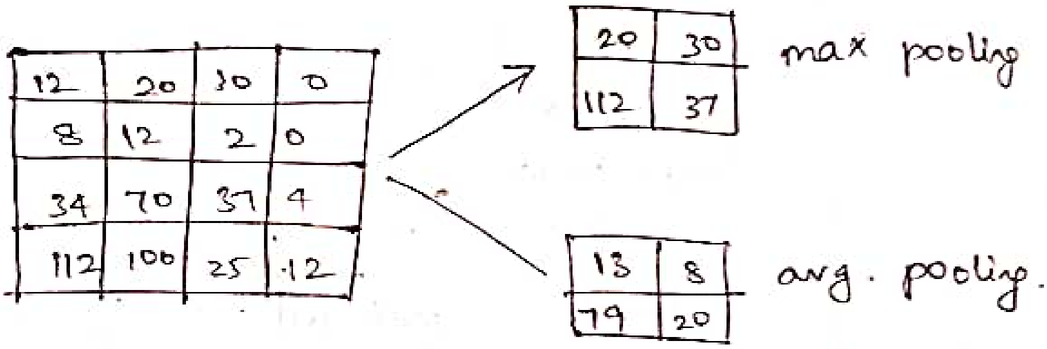
→ Max pooling returns the max. value from the portion of the image covered by the kernal.

Avg pooling:

→ It returns the avg of all the values from the portion of the img covered by the kernel.

→ max pooling also performs as a noise suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

→ Avg. pooling simply performs dimensionality reduction as a noise suppressing mechanism. hence we can say that max pooling performs a lot better than Avg pooling.



Types of pooling.

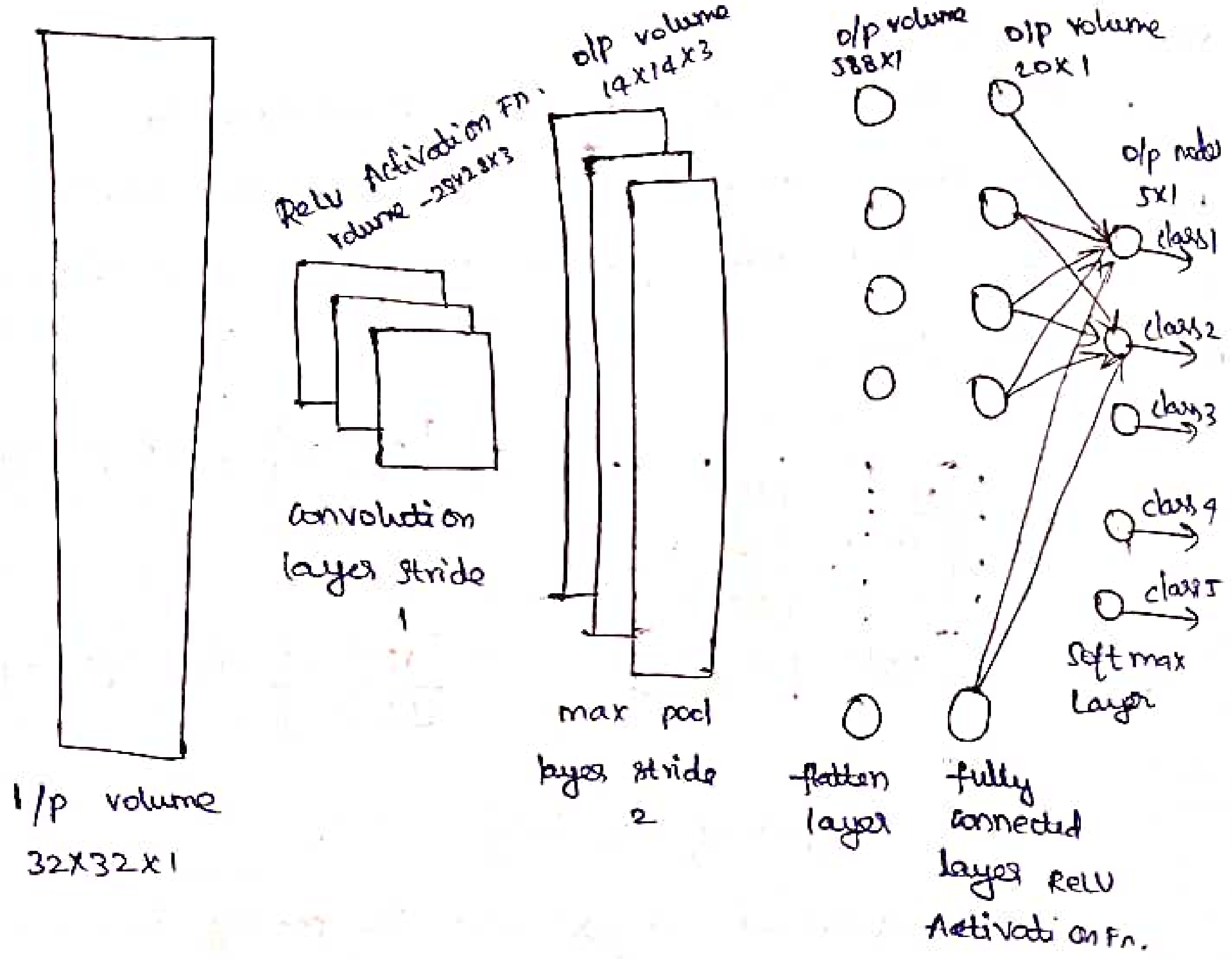
→ the convolutional layer and the pooling layer, together form the i-th layer of a convolutional neural network.

→ depending on the complexities in the img, the no. of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

⇒ after going through the above process, we have successfully enabled the model to understand the features.

⇒ going to flatten the final output & feed it to a regular neural network for classification purposes.

<u>classification</u>:: <u>fully</u> <u>connected</u> <u>layer</u> (FC layers)::



⇒ Adding a fully-connected layer is a cheap way of learning non-linear combinations of the high-level features as represented by the o/p of the convolutional layer.

⇒ fully connected layer is learning a possibly non-linear function in that space.

=> we converted out i/p img into a suitable form for out Multi-level Perceptron, we shall flatten the img into a column vector.

=> the flattened o/p is fed to a feed-forward neural network & backpropagation applied to every iteration of training.

=> over a series of epochs, the model is able to distinguish b/w dominating & certain low level features in img and classify them using the softmax classification technique.