Deep Learning

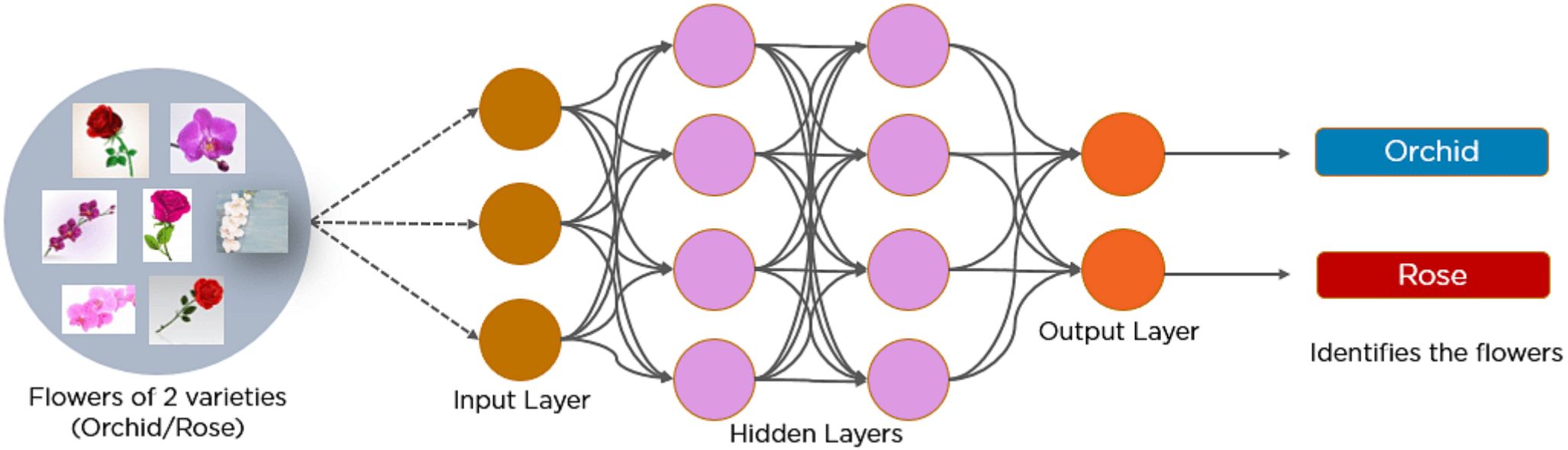
Convolutional Neural Networks : CNN

Convolutional Neural Network (CNN)

- One of the most popular deep neural networks is Convolutional Neural Networks. a class of deep neural networks, most commonly applied to analyze visual imagery.
- Yann LeCun, director of Facebook's AI Research Group, is the pioneer of convolutional neural networks.
- He built the first convolutional neural network called LeNet in 1988. LeNet was used for character recognition tasks like reading zip codes and digits.
- Facial recognition on social media, object detection in self-driving cars, or disease detection using visual imagery is possible due to convolutional neural networks (CNN).

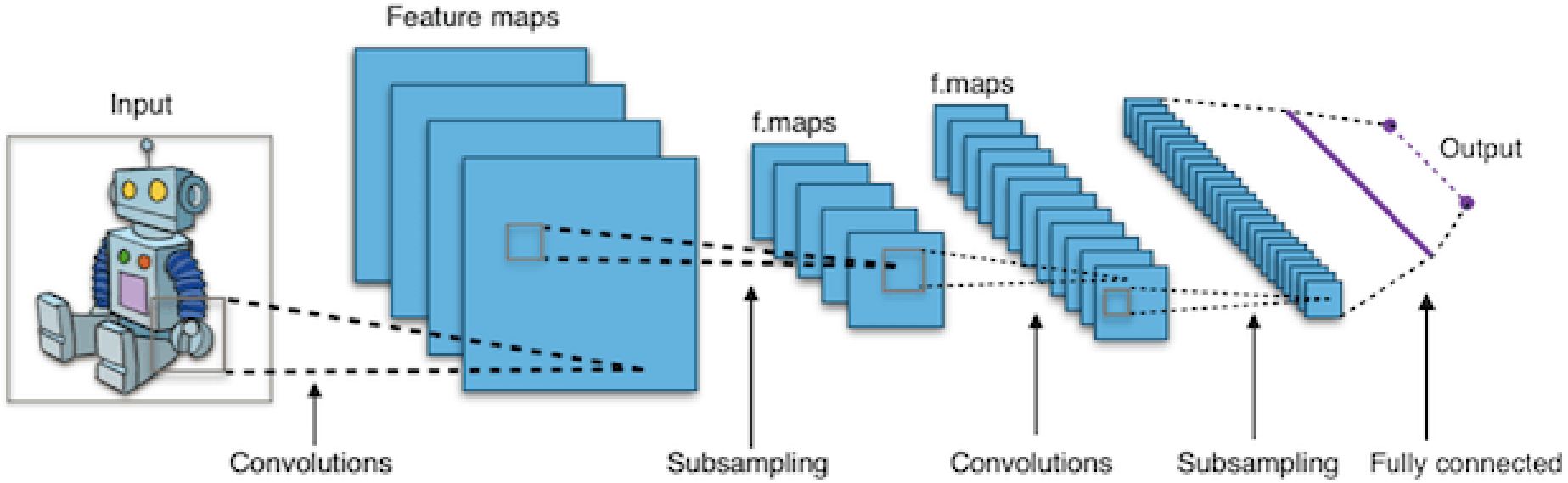
What is Convolutional Neural Network?

- A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image.

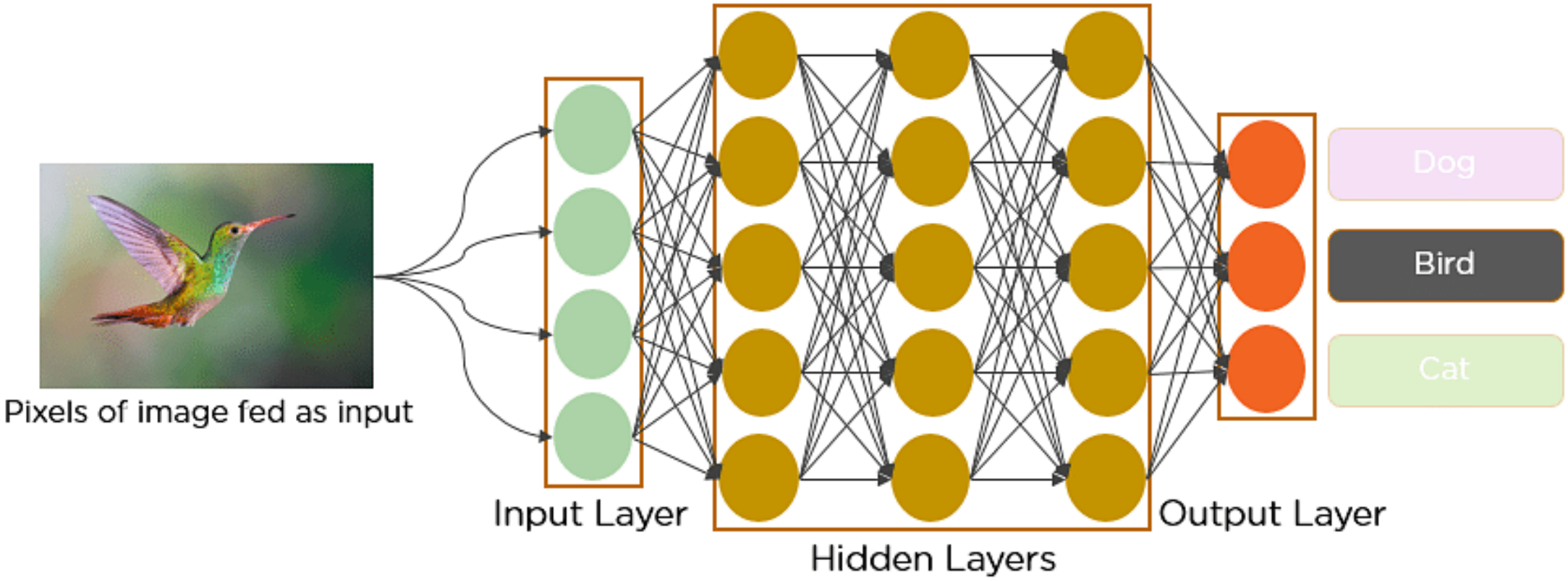


What is Convolutional Neural Network?

- Convolutional Neural Networks (CNN's) are multi-layer neural networks (sometimes up to 17 or more layers) that assume the input data to be images.



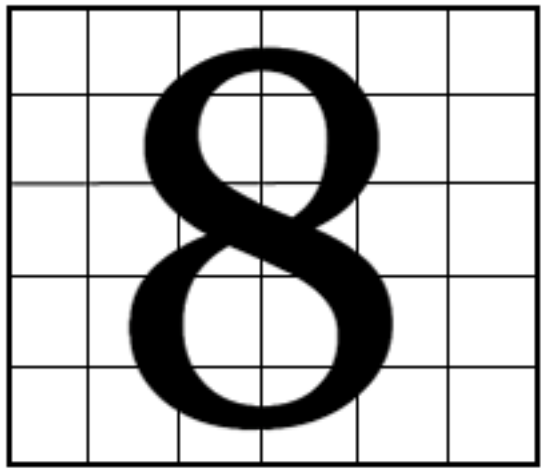
How CNN works?



The hidden layers carry out feature extraction by performing different calculations and manipulations.

How CNN works?

Real Image of the digit 8

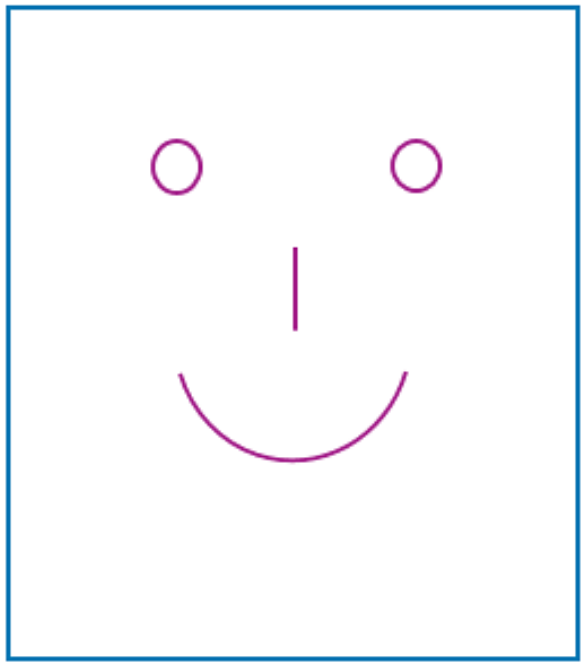


Represented in the form of an array

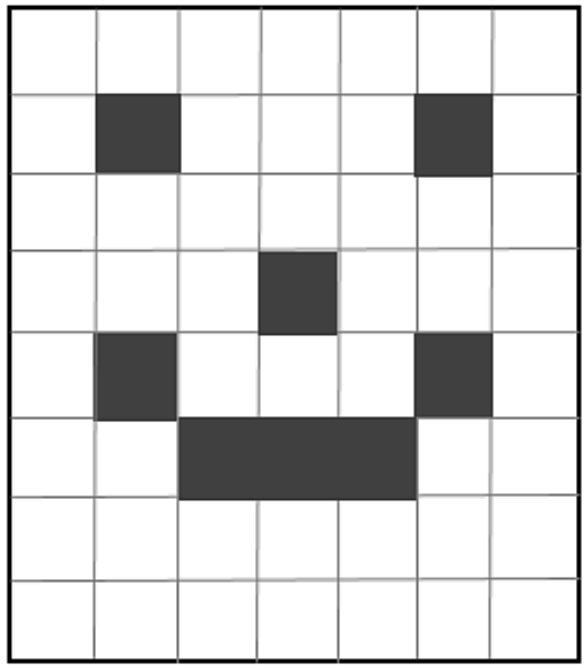
0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0

Digit 8 represented in the form of pixels of 0's and 1's

How CNN recognizes images



Real Image



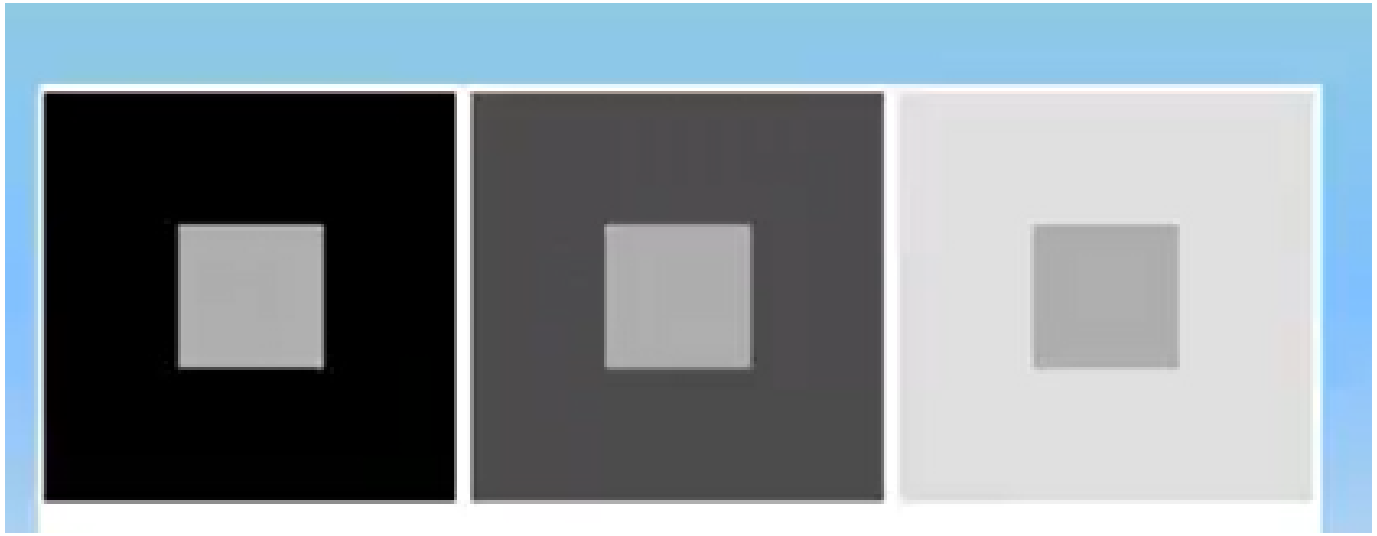
Represented in the form of black and white pixels

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

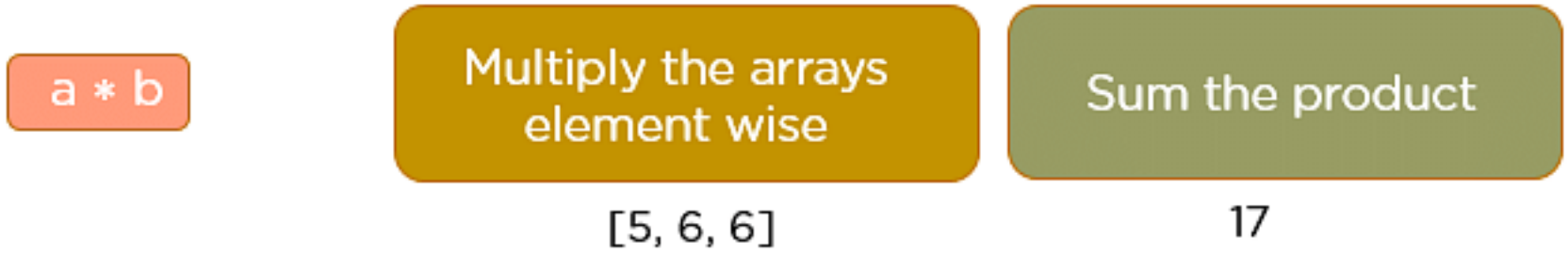
Image represented in the form of a matrix of numbers

Perception of a Pixel

- Perception of a pixel value is not only dependent on the intensity of pixel value but also depends on neighbourhood



How CNN works?



a = [5, 3, 2, 5, 9, 7]

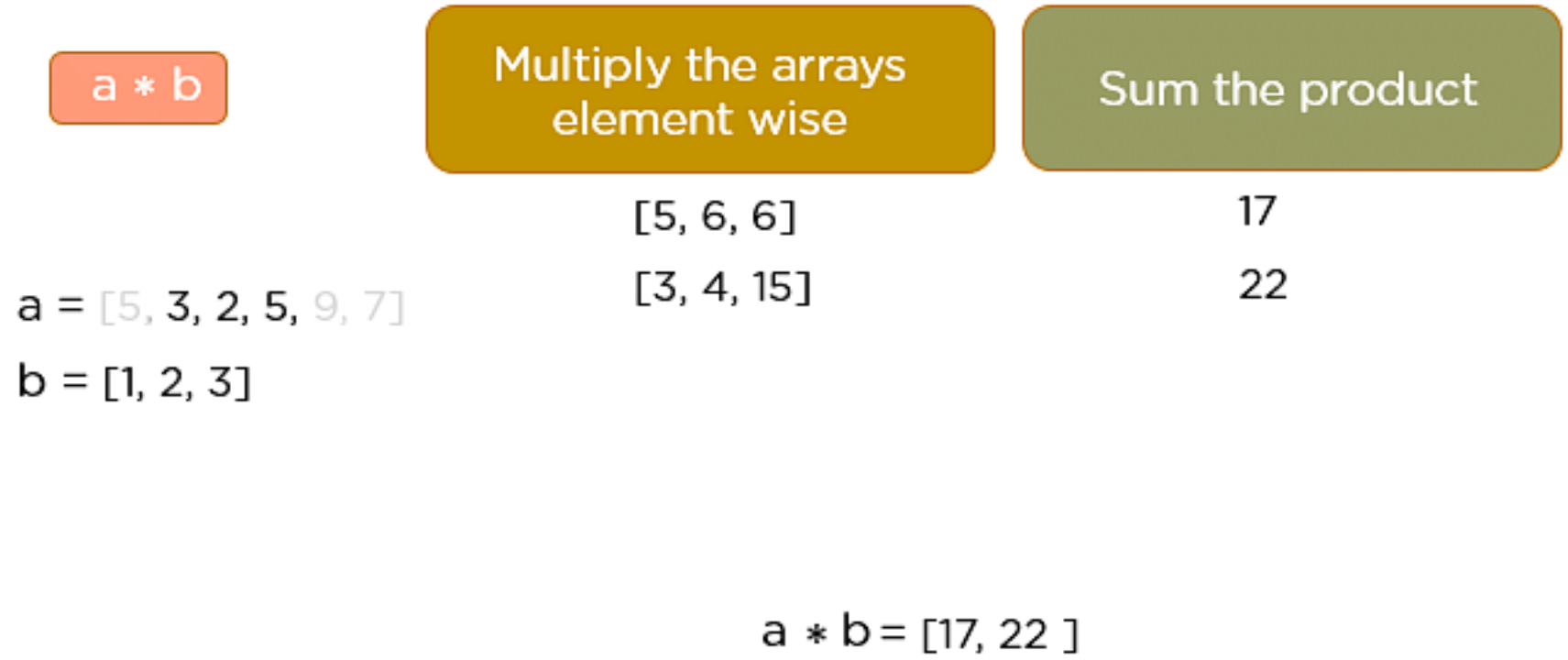
b = [1, 2, 3]

a * b = [17,]

In convolution operation, the arrays are multiplied element-wise, and the product is summed to create a new array, which represents a*b.

How CNN works?

The next three elements from the matrix a are multiplied by the elements in matrix b, and the product is summed up.



This process continues until the convolution operation is complete.

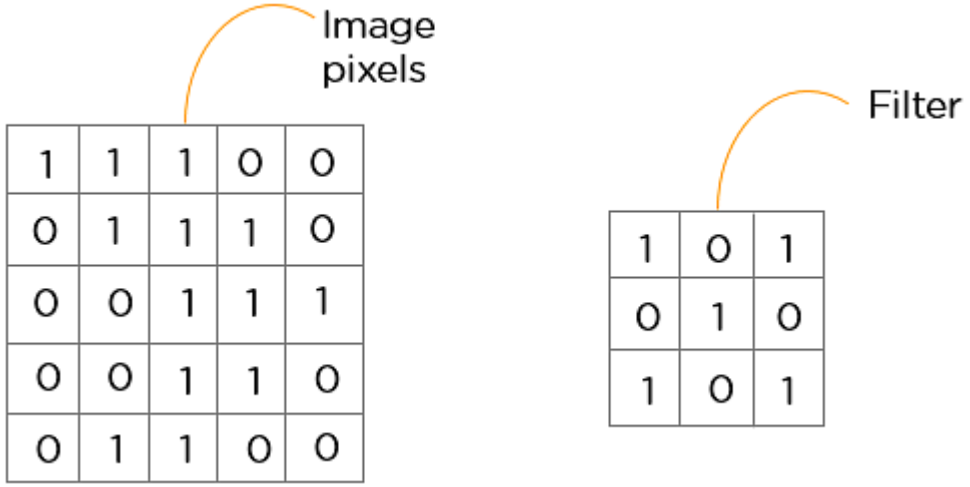
Layers in a Convolutional Neural Network

A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:

1. Convolution layer
2. ReLU layer
3. Pooling layer
4. Fully connected layer

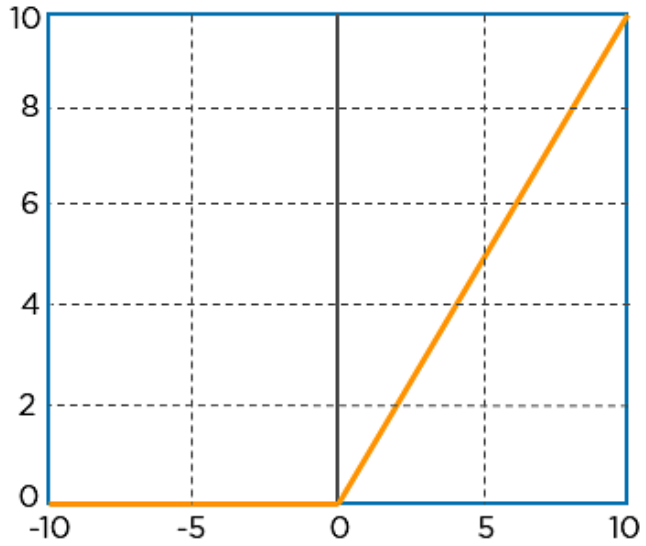
Convolution Layer

- This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.
- Consider the following 5x5 image whose pixel values are either 0 or 1. There's also a filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix.

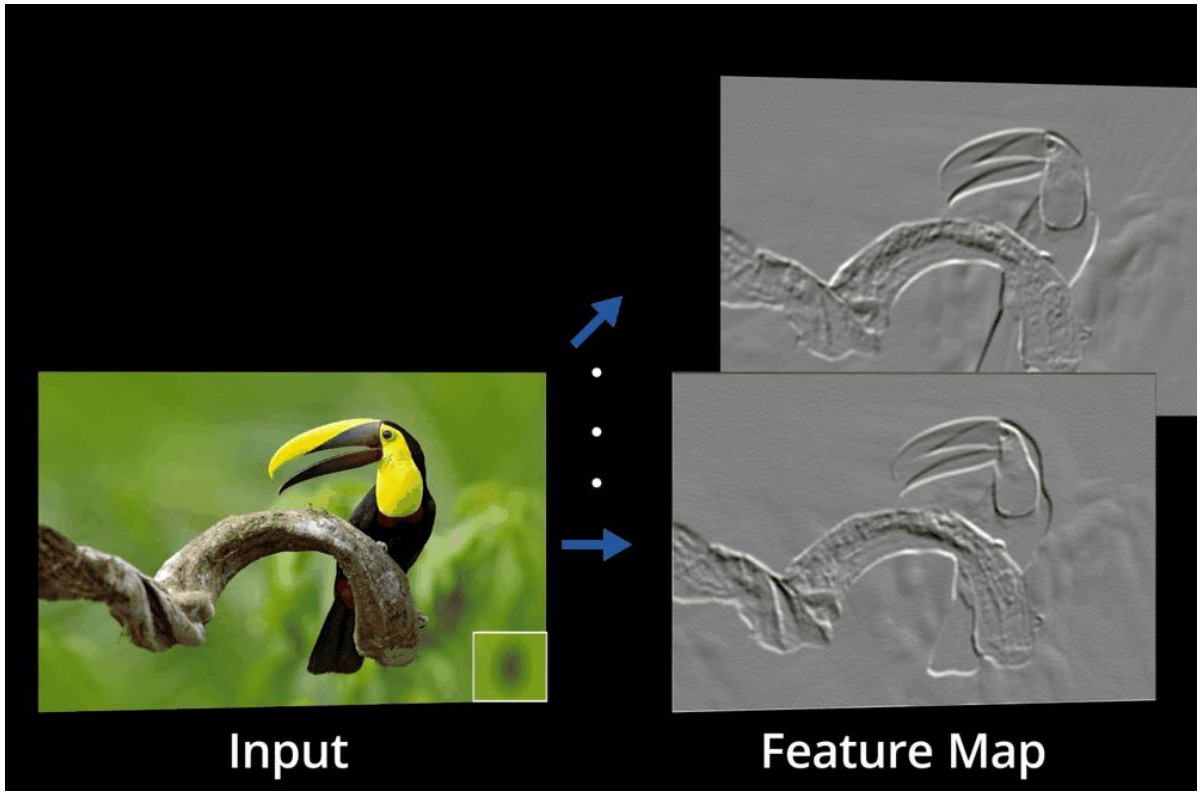


ReLU layer

- ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.
- ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map. Below is the graph of a ReLU function:



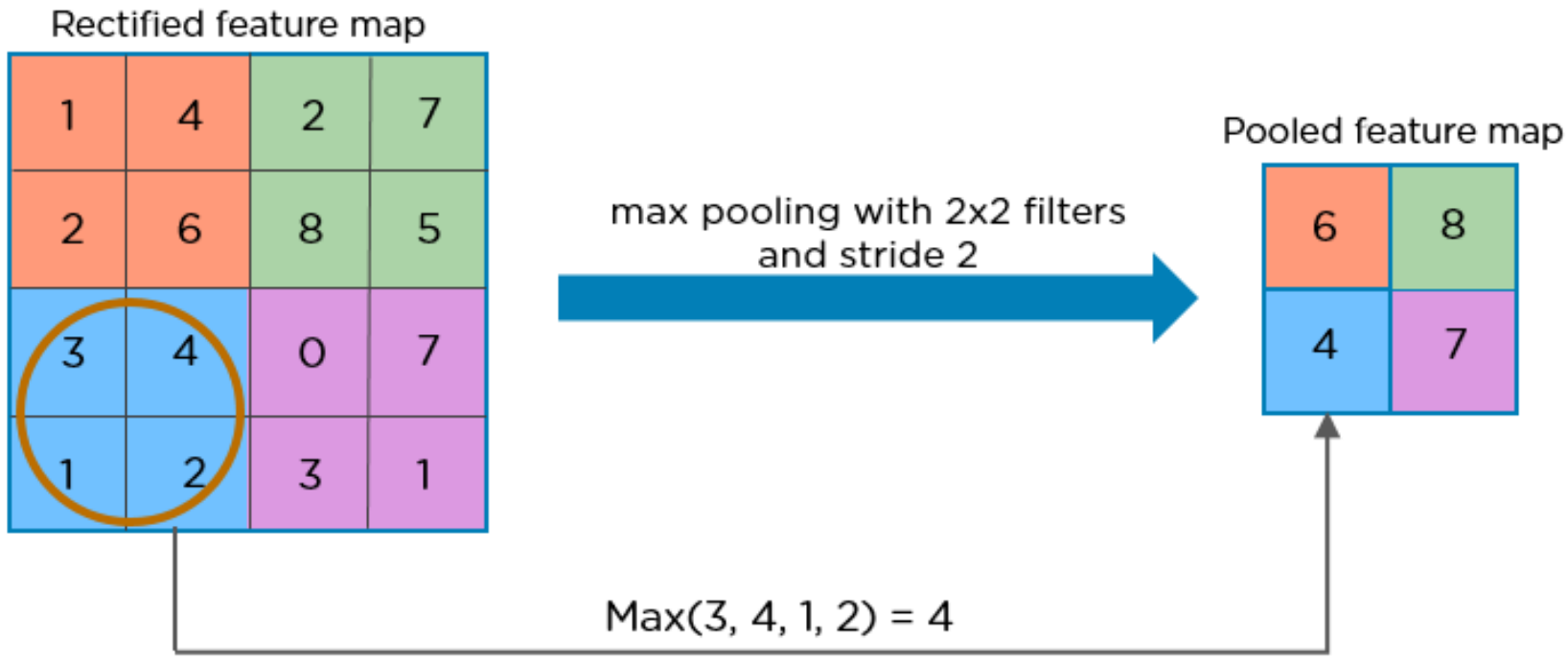
ReLU layer



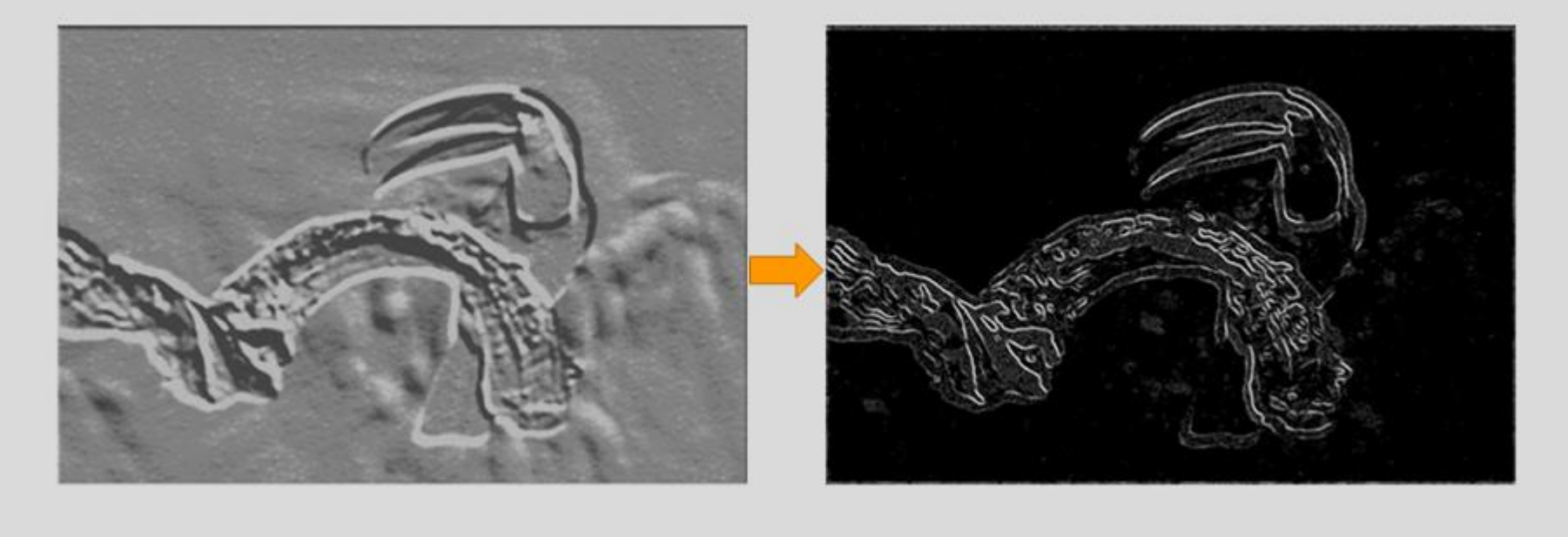
The original image is scanned with multiple convolutions and ReLU layers for locating the features

Pooling Layer

- Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.

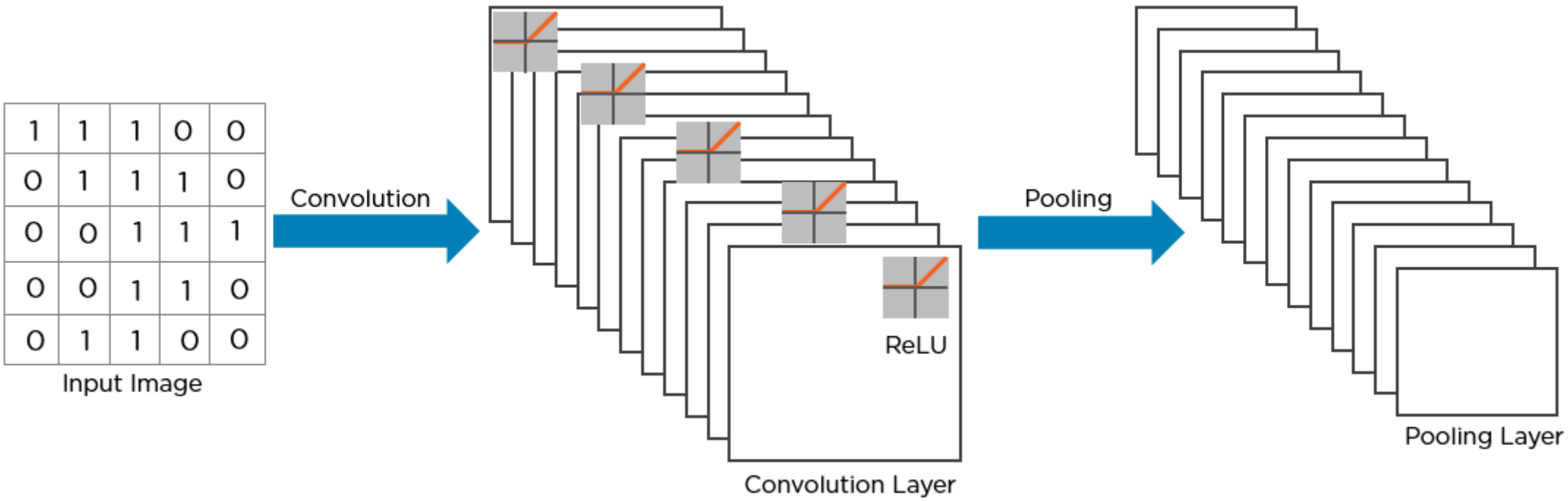


Pooling Layer



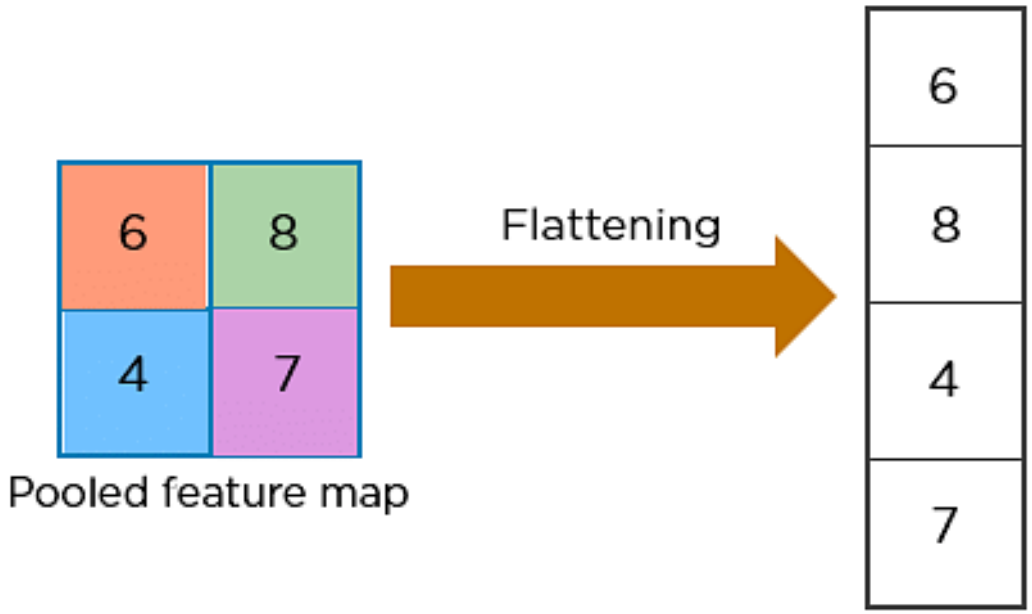
The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.

CNN, ReLU and Pooling Layers

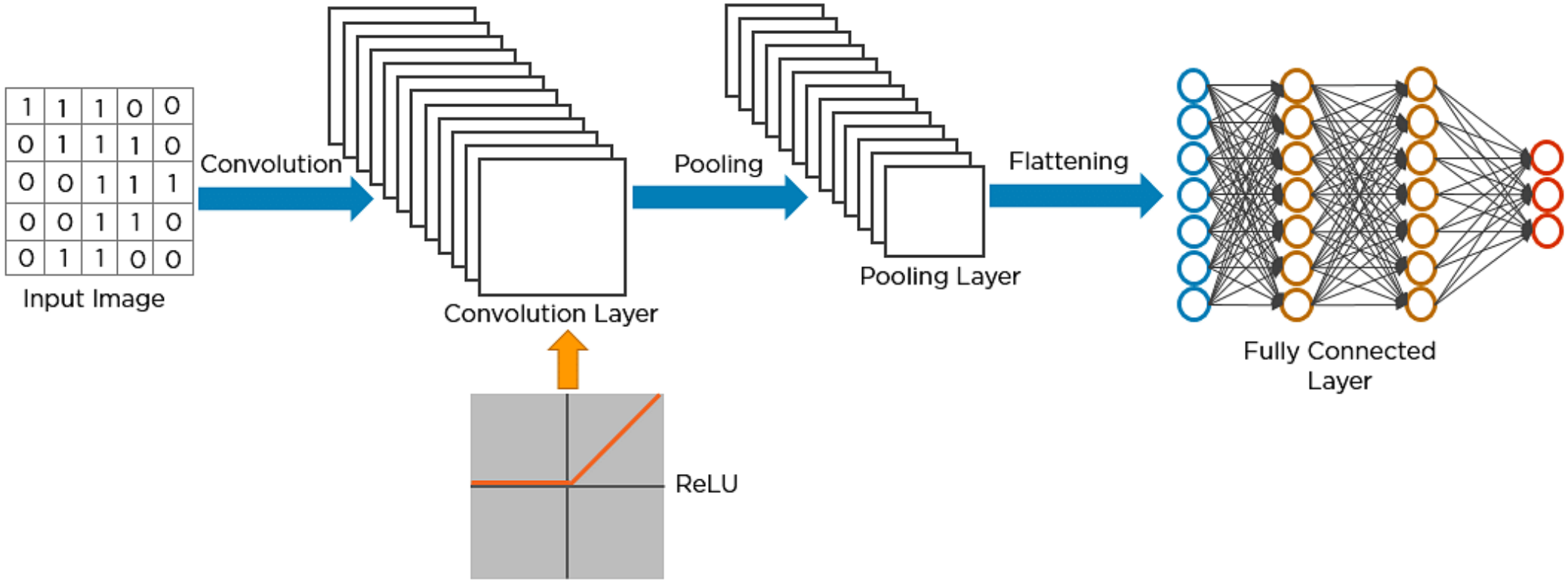


Flattening Process

- The next step in the process is called flattening.
- Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.

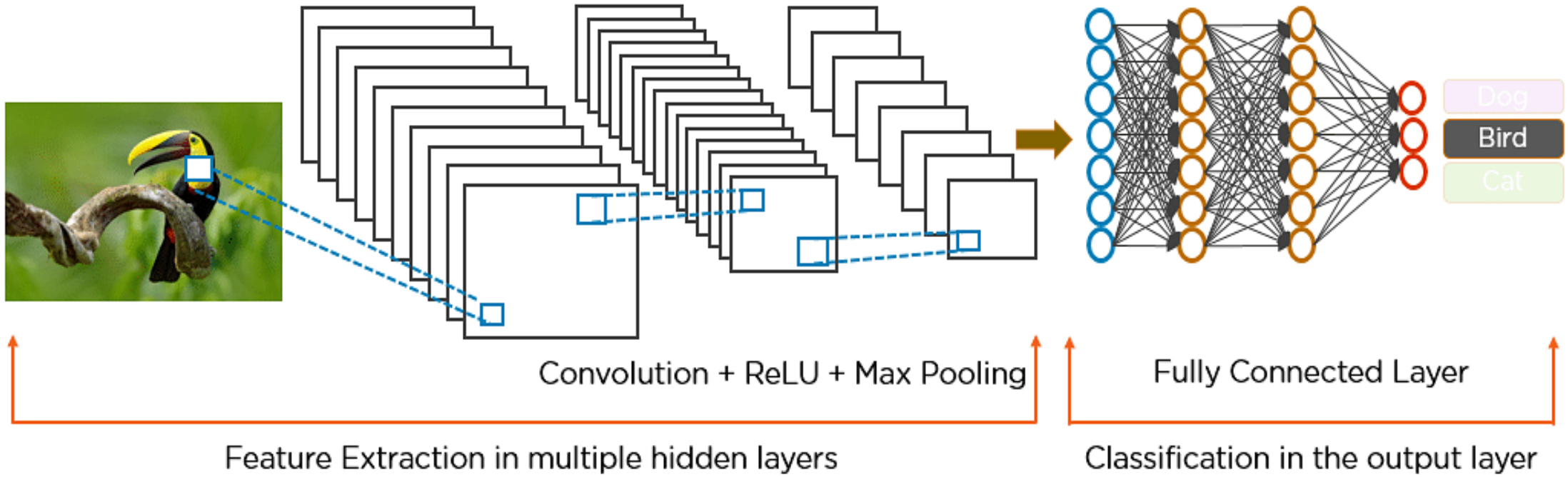


Fully Connected Layer



The flattened matrix is fed as input to the fully connected layer to classify the image.

Fully Connected Layer



How CNN recognizes an image?

- The pixels from the image are fed to the convolutional layer that performs the convolution operation
- It results in a convolved map
- The convolved map is applied to a ReLU function to generate a rectified feature map
- The image is processed with multiple convolutions and ReLU layers for locating the features
- Different pooling layers with various filters are used to identify specific parts of the image
- The pooled feature map is flattened and fed to a fully connected layer to get the final output

Steps for CNN

- Take input image
- Define weight matrix
- Input convolved to extract specific features without losing information and spatial arrangement.
- Reduces the number of parameters from the image.
- So less parameters to train.

How machine looks at an image?

Every image is an arrangement of dots arranged In a special order.
Eg to read an image with number 4 written on it.

The machine will break this image into matrix of pixels

And store the colour code of each pixel at a location.

Here number 1 is lightest and 256 is the darkest shade of Green colour.

Sampling : preserves spatial

25	2	1	44
223	7	6	60
196	8	2	148
249	1	3	40
60	7	1	154
59	1	7	213
214	7	3	163
89	182	219	13
74	146	113	72
89	18	244	85
1	4	8	97
3	4	2	121
2	1	2	131
7	6	8	47
3	5	5	126
7	6	8	121
5	3	1	237

Neural Network : Image Identification

- A fully connected network would take the image as an array by flattening it and considering pixel values as features to predict the number in the image.

- But Spatial arrangement of pixels is lost by flattening.

Preserving Spatial Arrangement

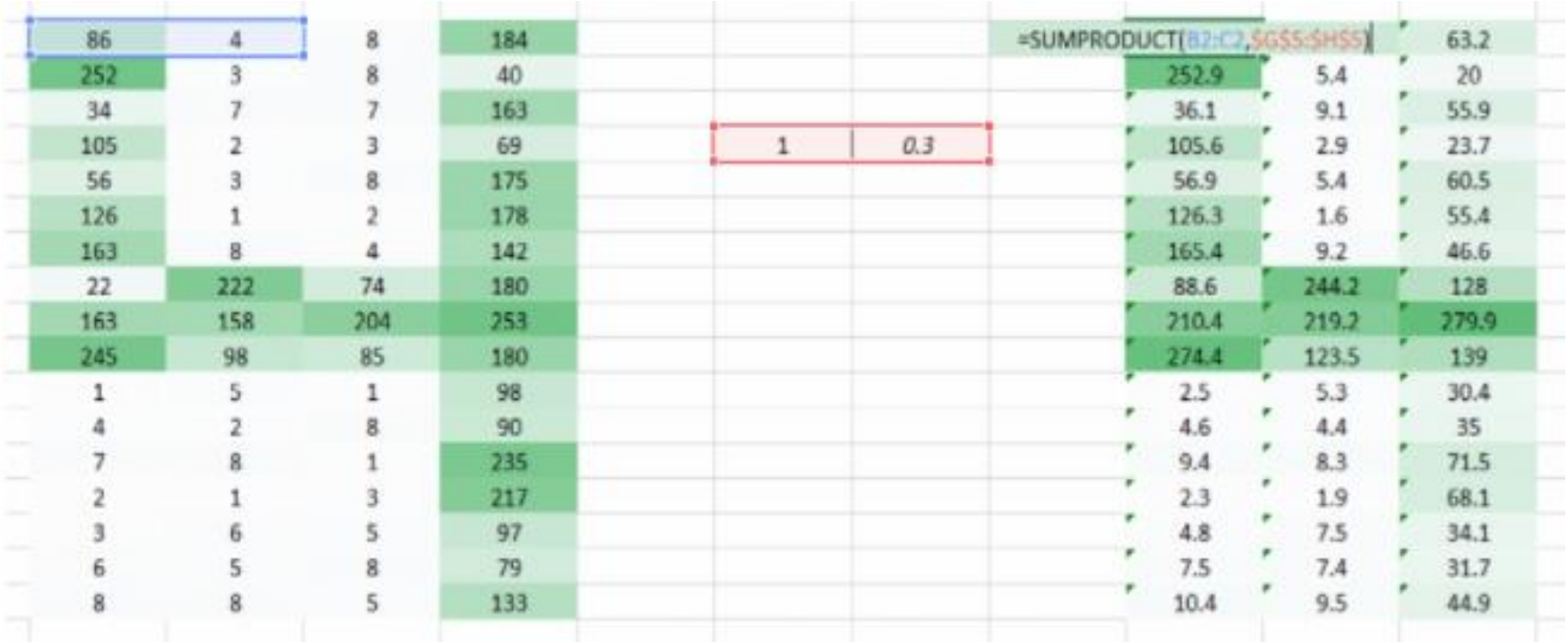
Case 1:

Here we have used a weight to multiply the initial pixel values.

25	2	1	44		=B2*$G$5	6	3	132
223	7	6	60		669	21	18	180
196	8	2	148	Weight	588	24	6	444
249	1	3	40	3	747	3	9	120
60	7	1	154		180	21	3	462
59	1	7	213		177	3	21	639
214	7	3	163		642	21	9	489
89	182	219	13		267	546	657	39
74	146	113	72		222	438	339	216
89	18	244	85		267	54	732	255
1	4	8	97		3	12	24	291
3	4	2	121		9	12	6	363
2	1	2	131		6	3	6	393
7	6	8	47		21	18	24	141
3	5	5	126		9	15	15	378
7	6	8	121		21	18	24	363
5	3	1	237		15	9	3	711

Quantization : taking intensity value to making brighter

Preserving Spatial Arrangement

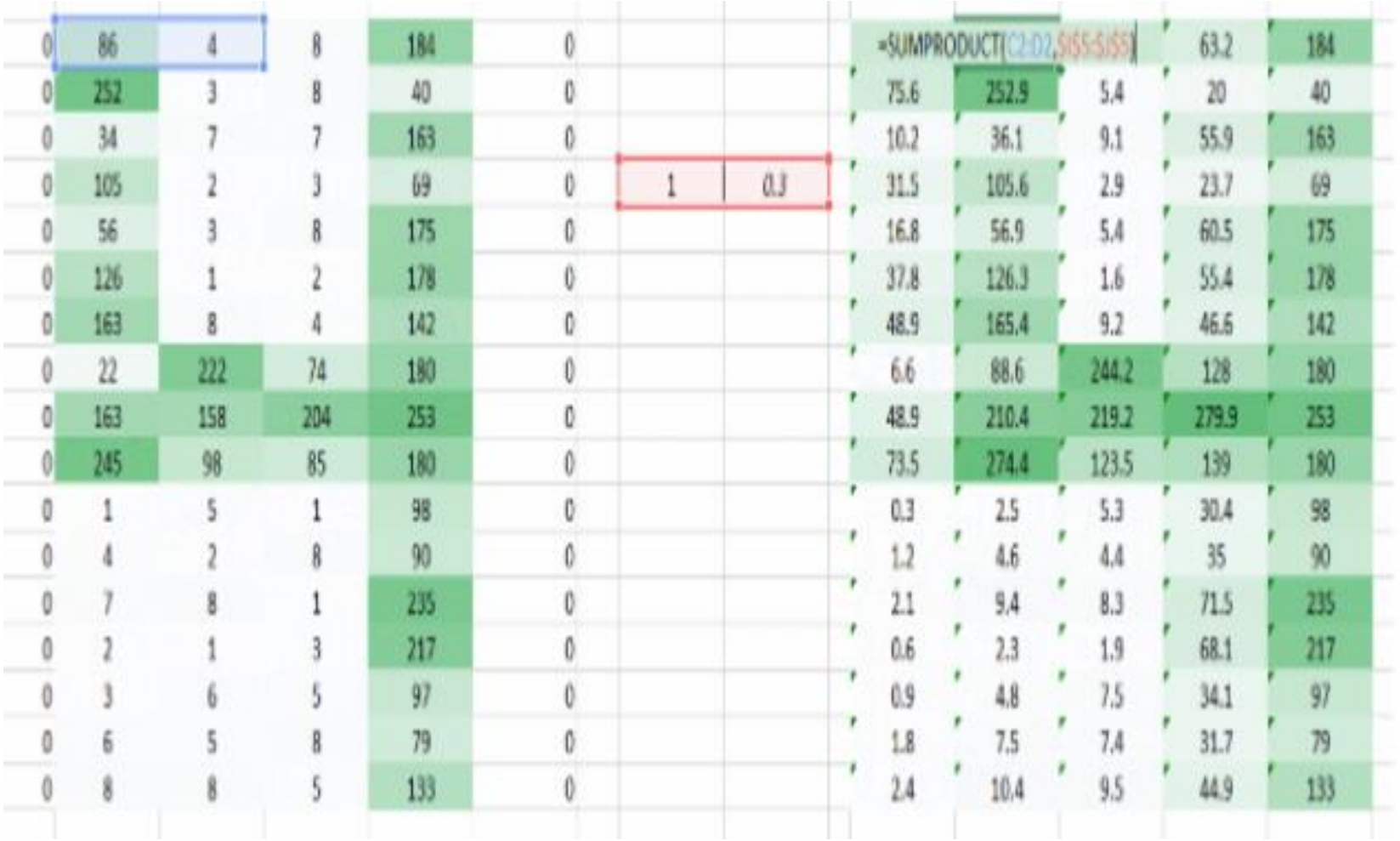


86	4	8	184			=SUMPRODUCT(B2:C2,$G$5:$H$5)		63.2
252	3	8	40			252.9	5.4	20
34	7	7	163			36.1	9.1	55.9
105	2	3	69	1	0.3	105.6	2.9	23.7
56	3	8	175			56.9	5.4	60.5
126	1	2	178			126.3	1.6	55.4
163	8	4	142			165.4	9.2	46.6
22	222	74	180			88.6	244.2	128
163	158	204	253			210.4	219.2	279.9
245	98	85	180			274.4	123.5	139
1	5	1	98			2.5	5.3	30.4
4	2	8	90			4.6	4.4	35
7	8	1	235			9.4	8.3	71.5
2	1	3	217			2.3	1.9	68.1
3	6	5	97			4.8	7.5	34.1
6	5	8	79			7.5	7.4	31.7
8	8	5	133			10.4	9.5	44.9

- 2D/3D arrangement of pixel values needs to be sent therefore two adjacent pixel values and finally two weights also.
- Image becomes a three column arrangement from 4. The image got smaller and by taking two horizontal pixels, feature extraction is being done.
- The right part of the image get slightly lost as the weight value of right is less.
- Gray scale image is a 2d matrix and values are intensity values.
- Size of matrix helps in deciding the spatial parameters

Preserving Spatial Arrangement

Now the network should consider the corners also like other pixels. So put 0's along the side of the weight movement.



0	86	4	8	184	0			=SUMPRODUCT(C2:D2,$I$5:$J$5)			63.2	184
0	252	3	8	40	0			75.6	252.9	5.4	20	40
0	34	7	7	163	0			10.2	36.1	9.1	55.9	163
0	105	2	3	69	0	1	0.3	31.5	105.6	2.9	23.7	69
0	56	3	8	175	0			16.8	56.9	5.4	60.5	175
0	126	1	2	178	0			37.8	126.3	1.6	55.4	178
0	163	8	4	142	0			48.9	165.4	9.2	46.6	142
0	22	222	74	180	0			6.6	88.6	244.2	128	180
0	163	158	204	253	0			48.9	210.4	219.2	279.9	253
0	245	98	85	180	0			73.5	274.4	123.5	139	180
0	1	5	1	98	0			0.3	2.5	5.3	30.4	98
0	4	2	8	90	0			1.2	4.6	4.4	35	90
0	7	8	1	235	0			2.1	9.4	8.3	71.5	235
0	2	1	3	217	0			0.6	2.3	1.9	68.1	217
0	3	6	5	97	0			0.9	4.8	7.5	34.1	97
0	6	5	8	79	0			1.8	7.5	7.4	31.7	79
0	8	8	5	133	0			2.4	10.4	9.5	44.9	133

Preserving Spatial Arrangement

Take multiple weight values in a single turn and put them together

- Weight Value (1,0.3)

87.2	6.4	63.2
252.9	5.4	20
36.1	9.1	55.9
105.6	2.9	23.7
56.9	5.4	60.5
126.3	1.6	55.4
165.4	9.2	46.6
88.6	244.2	128
210.4	219.2	279.9
274.4	123.5	139
2.5	5.3	30.4
4.6	4.4	35
9.4	8.3	71.5
2.3	1.9	68.1
4.8	7.5	34.1
7.5	7.4	31.7
10.4	9.5	44.9

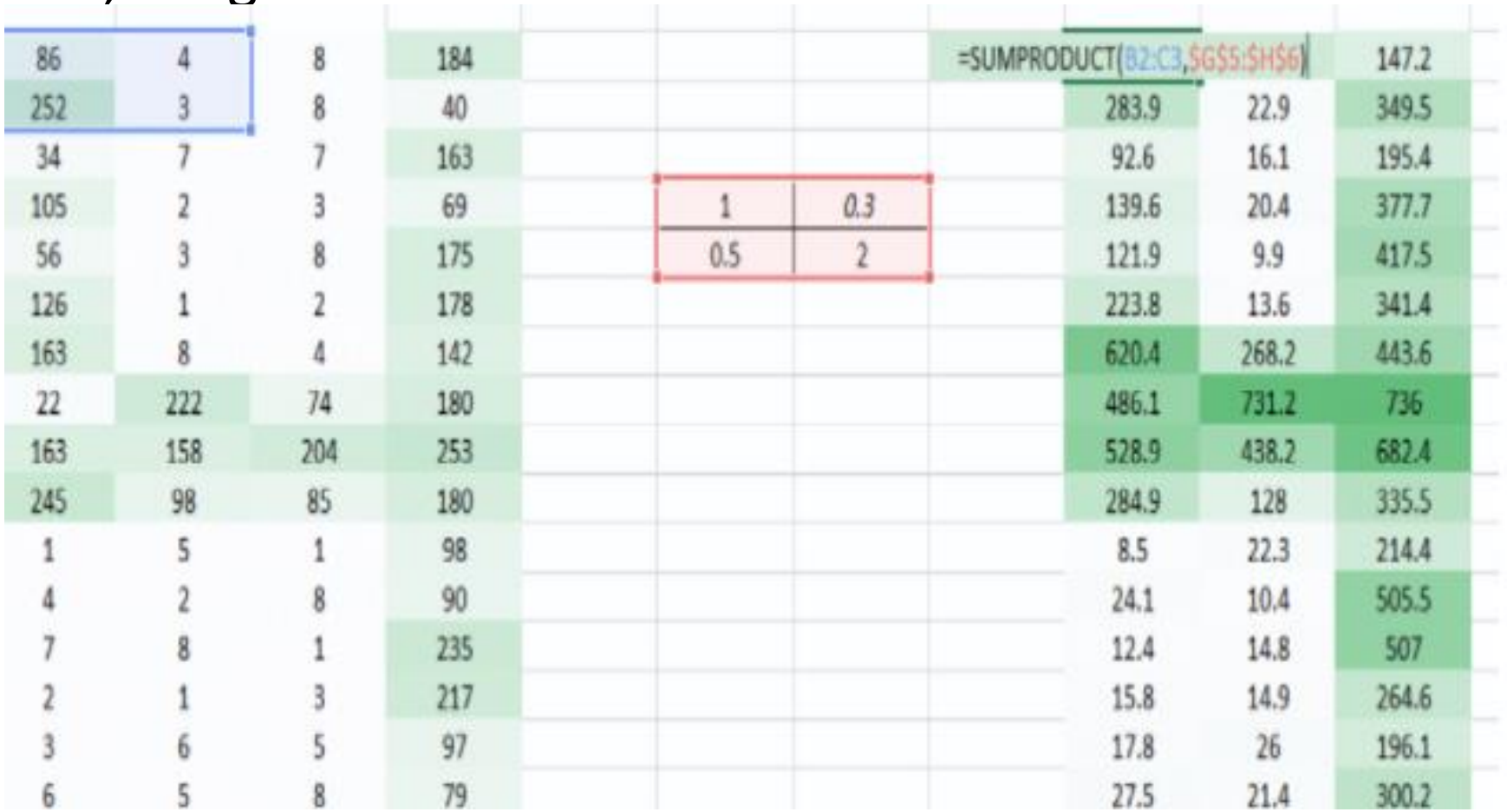
Weight Value (0.1,5)

28.6	40.4	920.8
40.2	40.3	200.8
38.4	35.7	815.7
20.5	15.2	345.3
20.6	40.3	875.8
17.6	10.1	890.2
56.3	20.8	710.4
1112.2	392.2	907.4
806.3	1035.8	1285.4
514.5	434.8	908.5
25.1	5.5	490.1
10.4	40.2	450.8
40.7	5.8	1175.1
5.2	15.1	1085.3
30.3	25.6	485.5
25.6	40.5	395.8
40.8	25.8	665.5

- Final output would be a combined version of the above two images.

Preserving Spatial Arrangement : 2D Matrix

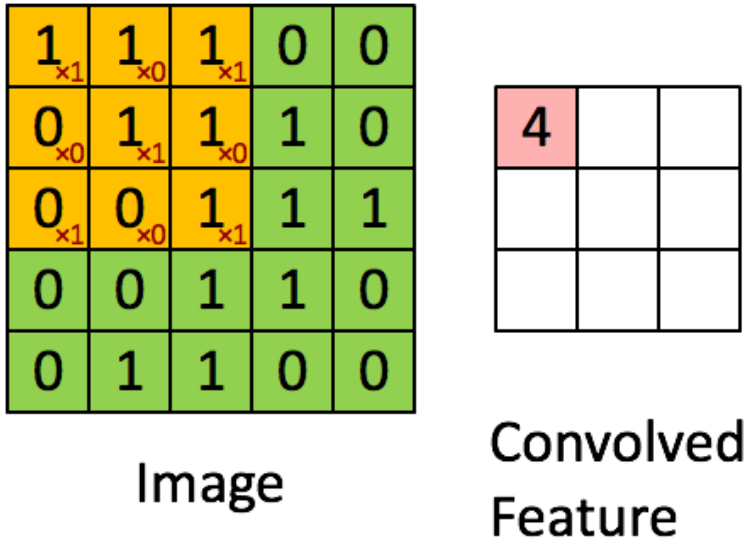
- To preserve the spatial arrangement in both horizontal and vertical direction, weights are taken as 2D matrix



Filter / Kernel

A filter/kernel(3×3 matrix) is applied to the input image to get the convolved feature. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products

Filters detect spatial patterns such as edges in an image by detecting the changes in intensity values of the image This convolved feature is passed on to the next layer.



Filter / Kernel

- The linear combination of the pixels weighted by the convolutional filter extracts some kind of feature from the image. It is done keeping these things in mind:
- Most of the useful features in an image are usually local and it makes sense to take few local pixels at a time to apply convolutions.
- Most of these useful features may be found in more than one place in an image. So, it makes sense to slide a single kernel all over the image in the hope of extracting that feature in different parts of the image using the same kernel.
- The same kernel is used for different set of pixels in an image, therefore the same weights are shared across these pixel sets as we convolve on them.
- And as the number of weights are less than a fully connected layer, we have lesser weights to back-propagate on.
- 3x3 convolution filters work in general, and is often the popular choice!

Dimensions of the filter

- A 2D convolution filter like 3x3 will always have a third dimension in size.
- The third dimension is equal to the number of channels of the input image.
- For example, we apply a 3x3x1 convolution filter on gray-scale images (that has 1 black and white channel) whereas,
- we apply a 3x3x3 convolution filter on a colored image (with 3 channels, red, blue and green).

Setting Filters

- In CNN, the filter values are considered as weights(+bias if set to have). So, as same as normal neural networks, the filters are updated(or optimized) as the weights are optimized during training.
- Before training you can choose the number of filters for each layer, which is the output dimensionality, or the number of output channels.
- If you look up for the operation of CNN, you will understand why the number of channels is the same as the number of filters you set.
- Different filters extract different hidden, latent features of an image.
- Early Convolutional filters in the CNN models used in Image Segmentation are known to extract easilty noticeable features such as lines.
- Later filters detect more abstract features such as textures.
- The number of filters are basically the hyper-parameters that a researcher must set before the training.

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

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

w1[:,:,1]

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

w1[:,:,2]

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

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

toggle movement

Defining a CNN

1.The convolutional layer

2. The Pooling layer (optional)

3. The output layer

The model has two main aspects: the feature extraction front end comprised of convolutional and pooling layers, and the classifier backend that will make a prediction.

1.The convolutional layer

We have initialized the weight as a 3*3 matrix. This weight shall now run across the image such that all the pixels are covered at least once, to give a convolved output. The value 429 above, is obtained by the adding the values obtained by element wise multiplication of the weight matrix and the highlighted 3*3 part of the input image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The 6 x 6 image is converted to 4 x 4.
Parameters are being shared in CNN.

Original Image

Convoluted image

Extracting Features

The weight matrix behaves like a filter in an image extracting particular information from the original image matrix. A weight combination might be extracting edges, while another one might a particular color, while another one might just blur the unwanted noise. The weights are learnt such that the loss function is minimized similar to an MLP. Therefore weights are learnt to extract features from the original image which help the network in correct prediction. When we have multiple convolutional layers, the initial layer extract more generic features, while as the network gets deeper, the features extracted by the weight matrices are more and more complex and more suited to the problem at hand.

Stride and Padding

As we saw above, the filter or the weight matrix, was moving across the entire image moving **one** pixel at a time. We can define it like a hyperparameter, as to how we would want the weight matrix to move across the image. If the weight matrix moves 1 pixel at a time, we call it as a stride of 1. Let's see how a stride of 2 would look like.

INPUT IMAGE

18	54	51	239	244
55	121	75	78	95
35	24	204	113	109
3	154	104	235	25
15	253	225	159	78

WEIGHT

1	0	1
0	1	0
1	0	1

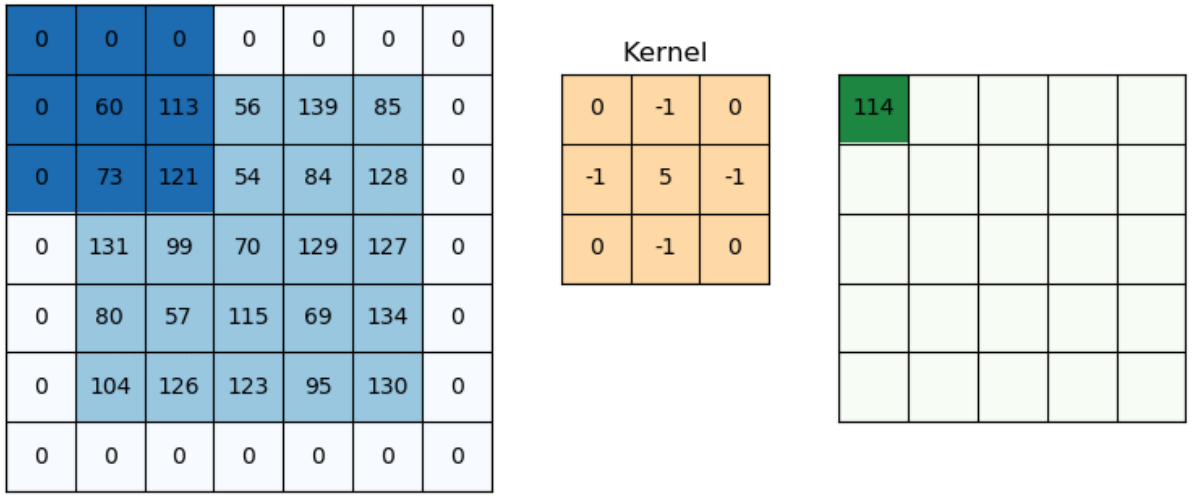
429	686
633	412

As you can see the size of image keeps on reducing as we increase the stride value. Padding the input image with zeros across it solves this problem for us. We can also add more than one layer of zeros around the image in case of higher stride values.

0	0	0	0	0	0	0	0
0	18	54	51	239	244	188	0
0	55	121	75	78	95	88	0
0	35	24	204	113	109	221	0
0	3	154	104	235	25	130	0
0	15	253	225	159	78	233	0
0	68	85	180	214	245	0	0

Padding

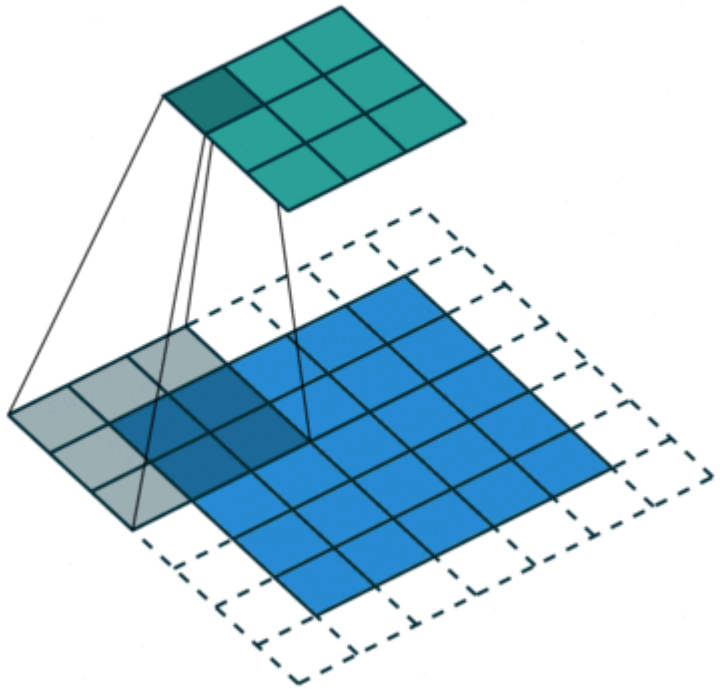
While applying convolutions, the output dimensions are not the same as input, we lose data over borders so we append a border of zeroes and recalculate the convolution covering all the input values.



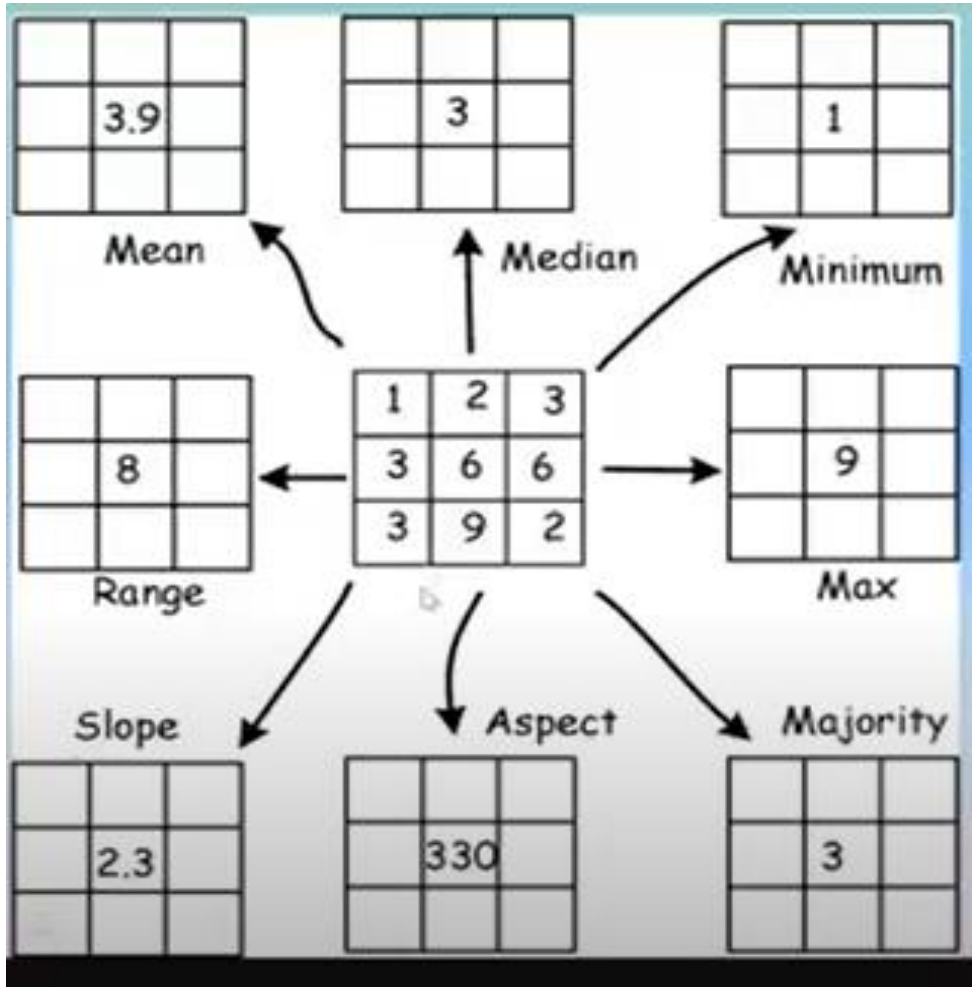
Striding

Sometimes we do not want to capture All the data or information available so we Skip some neighboring cells.

Here the input matrix or image is of dimensions 5×5 with a filter of 3×3 and a stride of 2 so every time we skip two columns and convolute

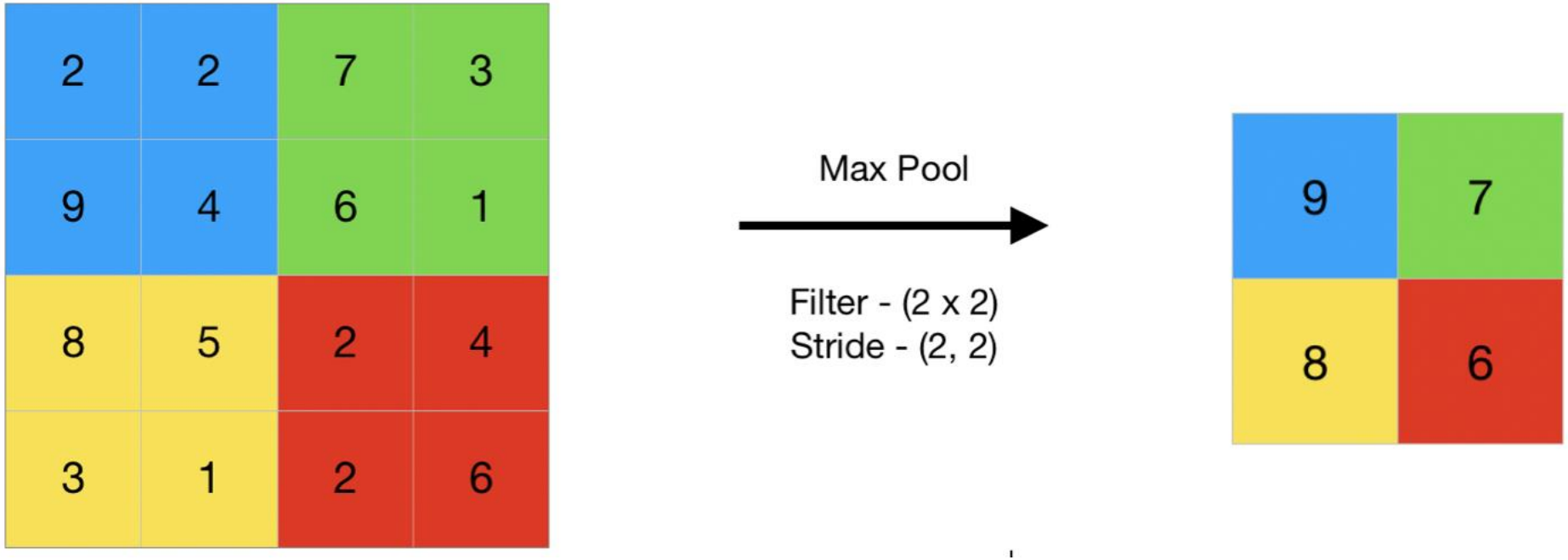


Filters


3.9
Mean
3
Median
1
Minimum
1 2 3
3 6 6
3 9 2
8
Range
9
Max
Slope
Aspect
Majority
2.3
330
3

Pooling

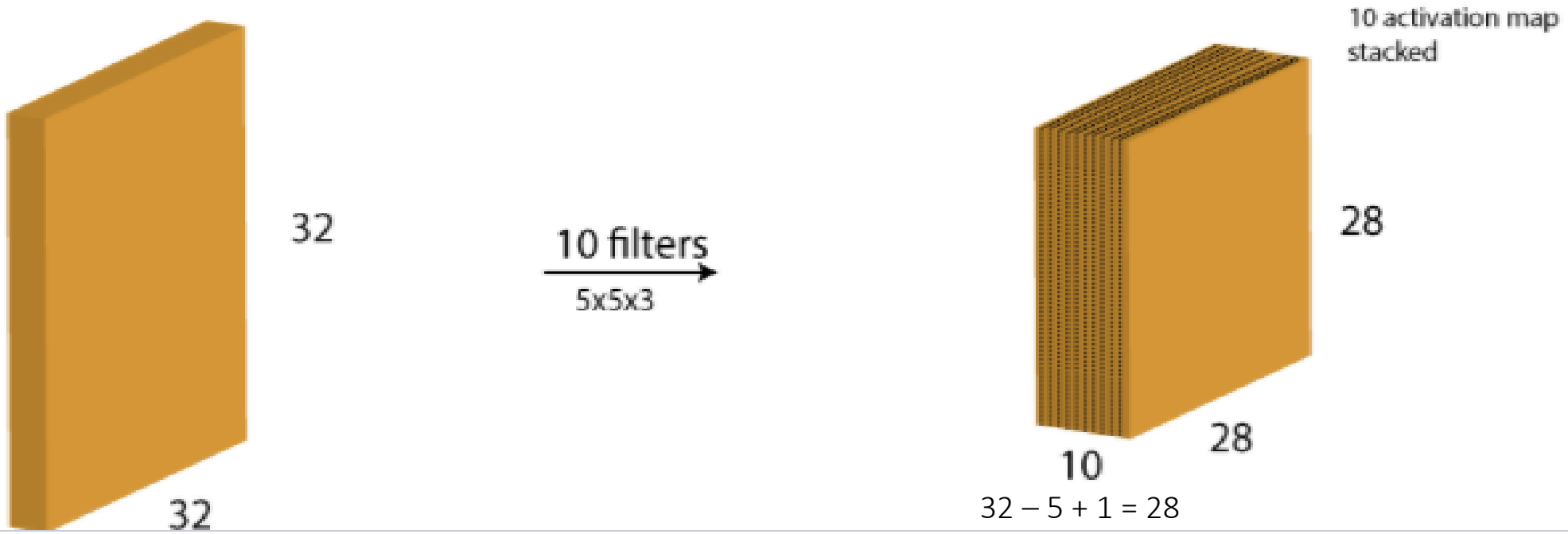
In general terms pooling refers to a small portion, so here we take a small portion of the input and try to take the average value referred to as average pooling or take a maximum value termed as max pooling, so by doing pooling on an image we are not taking out all the values we are taking a summarized value over all the values present !!!



here this is an example of max pooling so here taking a stride of two we are taking the maximum value present in the matrix

Multiple filters and the activation map

One thing to keep in mind is that the depth dimension of the weight would be same as the depth dimension of the input image. The weight extends to the entire depth of the input image. Therefore, convolution with a single weight matrix would result into a convolved output with a single depth dimension. In most cases instead of a single filter(weight matrix), we have multiple filters of the same dimensions applied together. The output from the each filter is stacked together forming the depth dimension of the convolved image. Suppose we have an input image of size 32*32*3. And we apply 10 filters of size 5*5*3 with valid padding. The output would have the dimensions as 28*28*10. You can visualize it as -



32 – 5 + 1 = 28

Output Dimensions

1. **The number of filters** - the depth of the output volume will be equal to the number of filter applied. Remember how we had stacked the output from each filter to form an activation map. The depth of the activation map will be equal to the number of filters.
2. **Stride** - When we have a stride of one we move across and down a single pixel. With higher stride values, we move large number of pixels at a time and hence produce smaller output volumes.
3. **Zero padding** - This helps us to preserve the size of the input image. If a single zero padding is added, a single stride filter movement would retain the size of the original image.

Output Dimensions

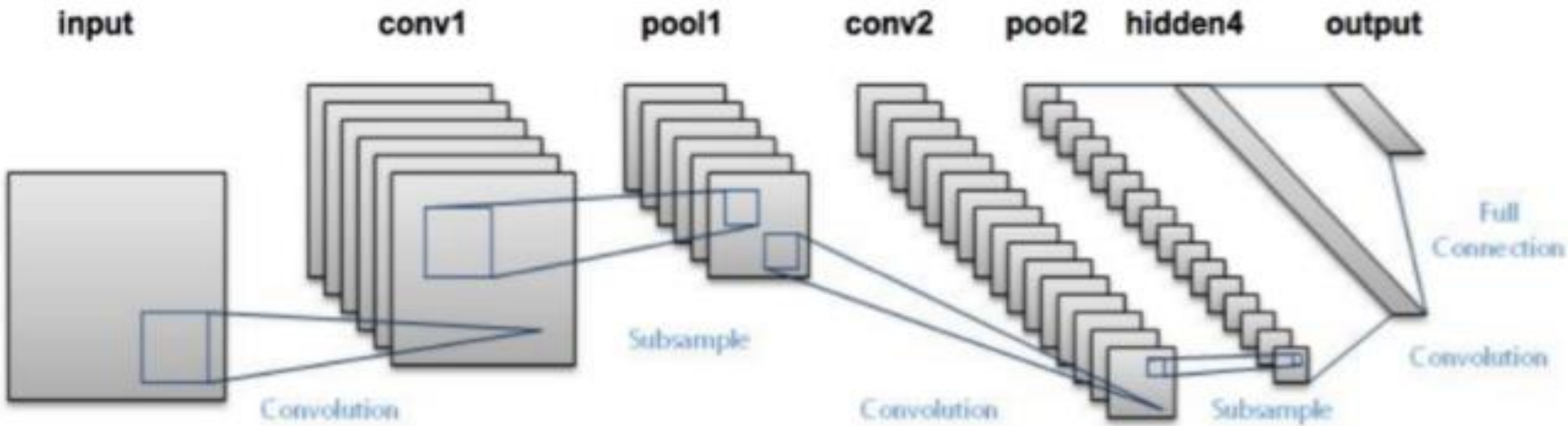
We can apply a simple formula to calculate the output dimensions. The spatial size of the output image can be calculated as([W-F+2P]/S)+1. Here, W is the input volume size, F is the size of the filter, P is the number of padding applied and S is the number of strides. Suppose we have an input image of size 32*32*3, we apply 10 filters of size 3*3*3, with single stride and no zero padding. Here W=32, F=3, P=0 and S=1. The output depth will be equal to the number of filters applied i.e. 10. The size of the output volume will be ([32-3+0]/1)+1 = 30. Therefore the output volume will be 30*30*10.

Output Layer

- The Convolutional and pooling layers would extract features and reduce the number of parameters from original image.
- To get the final output, we need to apply a fully connected layer to generate an output equal to the number of classes we need.
- Conv layers produce 3D activation maps while output required is whether an image belongs to a particular class.
- The output layer has a loss function to compute the error in prediction.
- Updation of biases and weight takes place for error and loss reduction.

Entire CNN Network

CNN as you can now see is composed of various convolutional and pooling layers. Let's see how the network looks like.

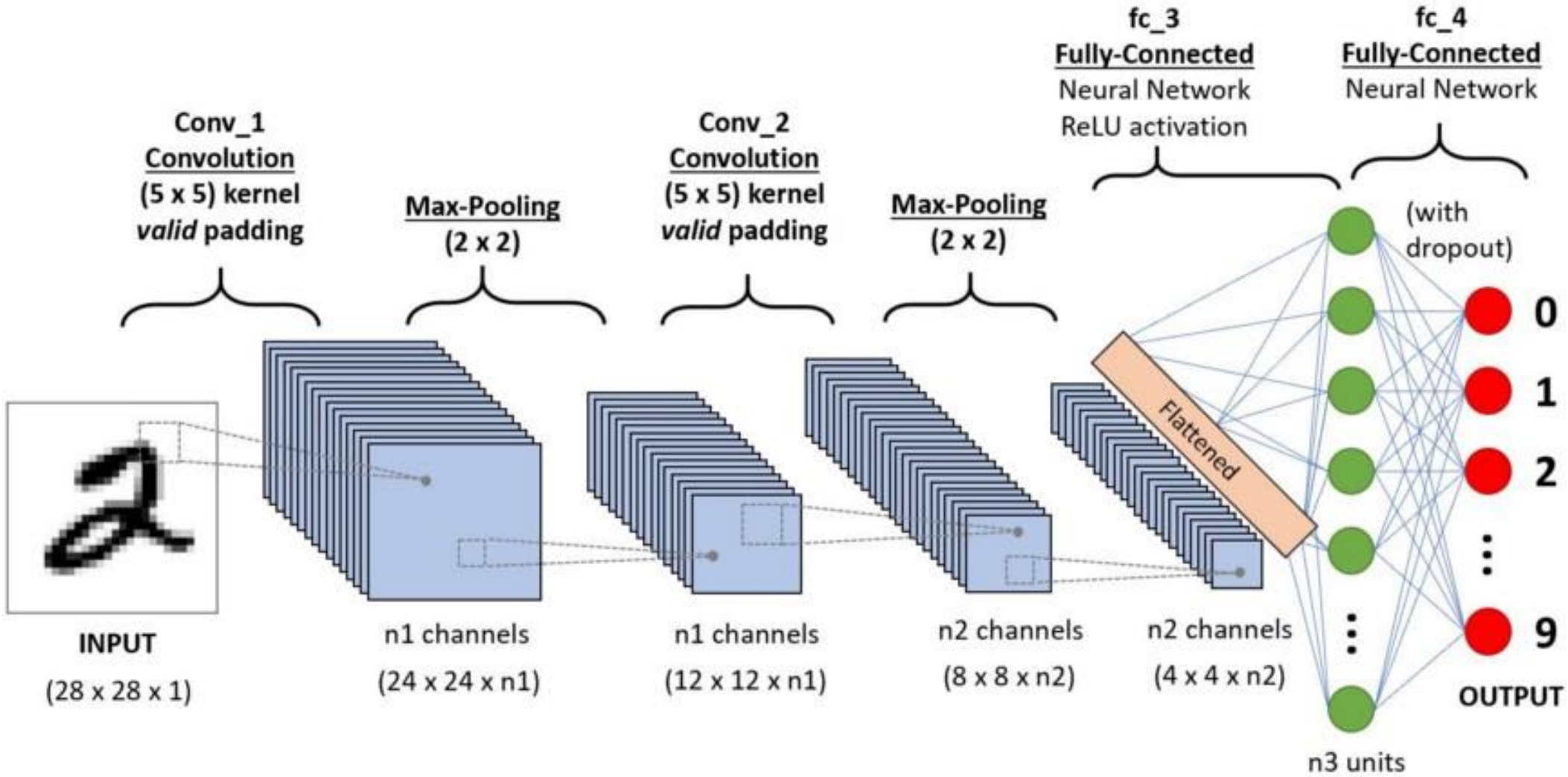


Entire CNN Network

- We pass an input image to the first convolutional layer. The convoluted output is obtained as an activation map. The filters applied in the convolution layer extract relevant features from the input image to pass further.
- Each filter shall give a different feature to aid the correct class prediction. In case we need to retain the size of the image, we use same padding(zero padding), other wise valid padding is used since it helps to reduce the number of features.
- Pooling layers are then added to further reduce the number of parameters
- Several convolution and pooling layers are added before the prediction is made. Convolutional layer help in extracting features. As we go deeper in the network more specific features are extracted as compared to a shallow network where the features extracted are more generic.

Entire CNN Network

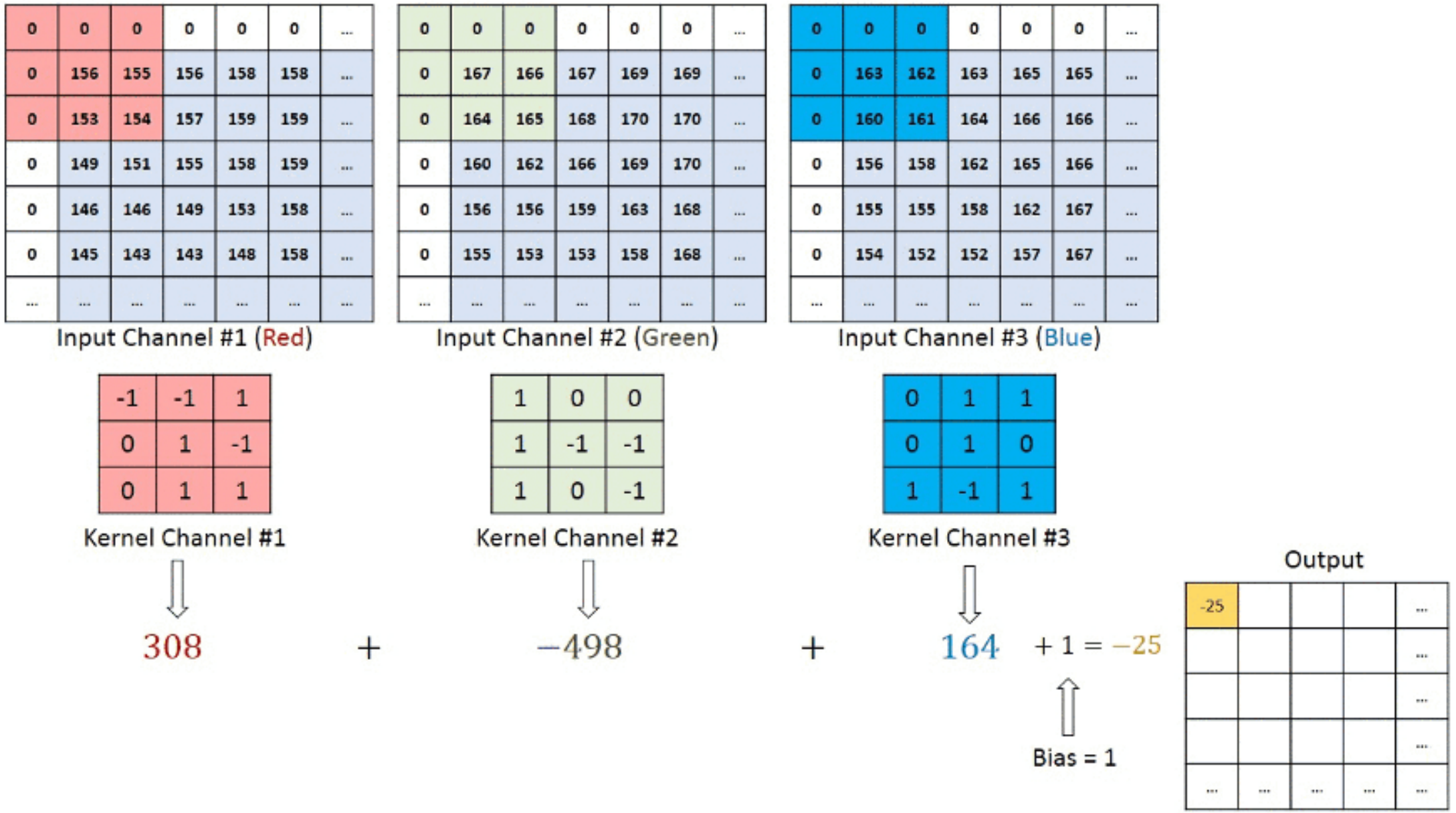
- The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network.
- The output is then generated through the output layer and is compared to the output layer for error generation. A loss function is defined in the fully connected output layer to compute the mean square loss. The gradient of error is then calculated.
- The error is then backpropagated to update the filter(weights) and bias values.
- One training cycle is completed in a single forward and backward pass.



In mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

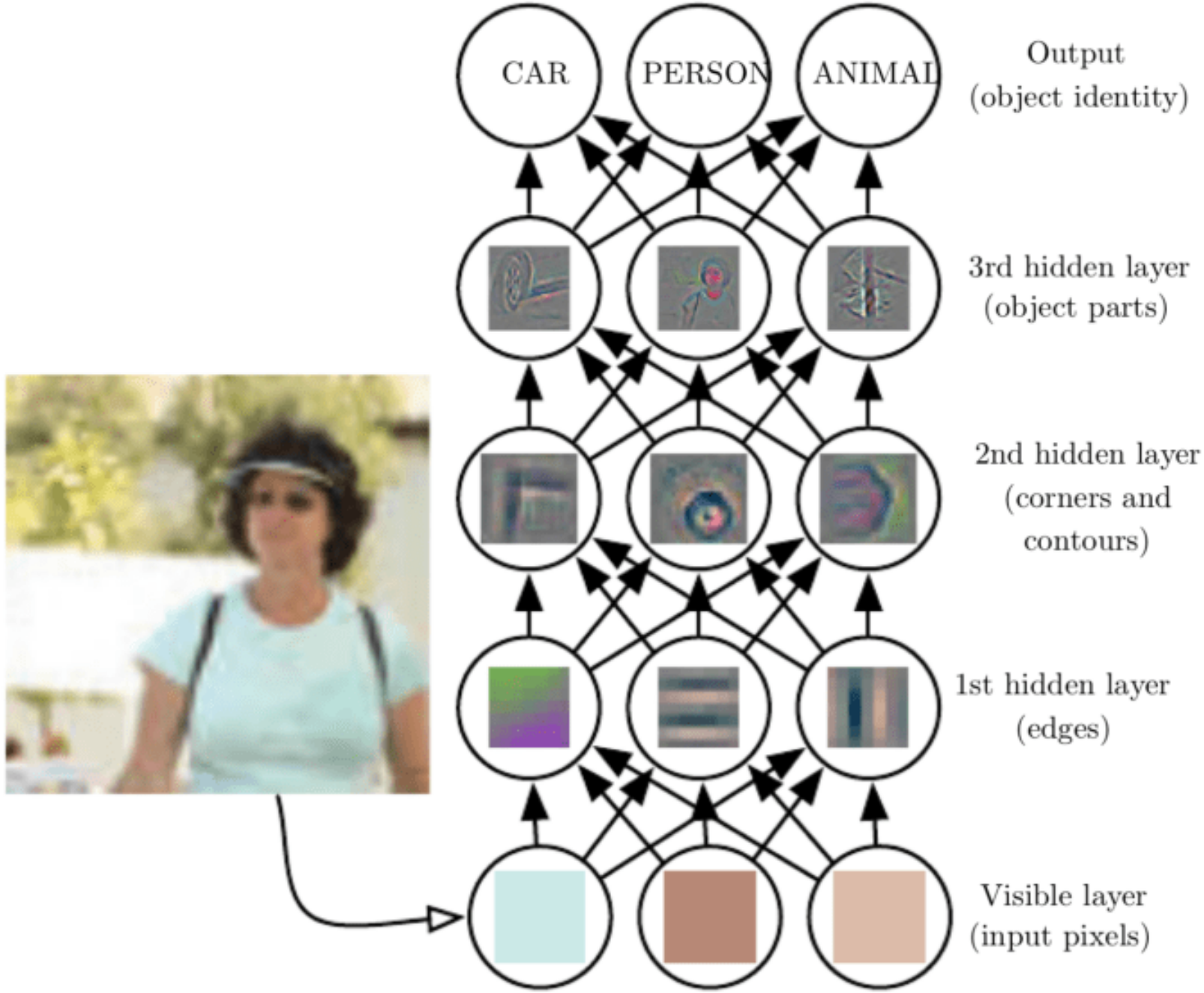
Mainly the role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

RGB color image



- When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

- The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.


CAR
PERSON
ANIMAL
Output
(object identity)
3rd hidden layer
(object parts)
2nd hidden layer
(corners and
contours)
1st hidden layer
(edges)
Visible layer
(input pixels)

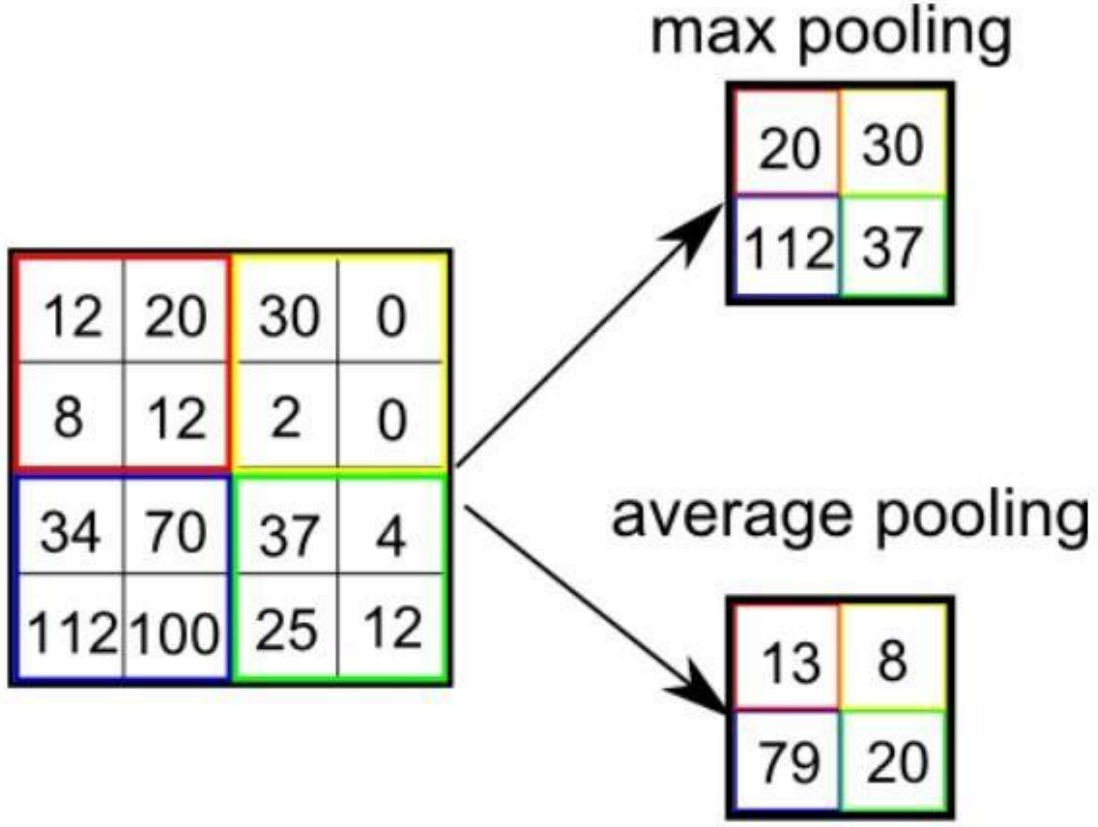
Pooling layer

- 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** by reducing the dimensions. There are two types of pooling average pooling and max pooling.
- So what we do in Max Pooling is we find the maximum value of a pixel from a portion of the image 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.
- On the other hand, **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.

Pooling layer

- Max Pooling is we find the maximum value of a pixel from a portion of the image covered by the kernel. Max Pooling also performs as a **Noise Suppressant**. I

Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism.



- The higher the number of filters, the higher the number of abstractions that your Network is able to extract from image data.
- The reason why the number of filters is generally ascending is that **at the input layer the Network receives raw pixel data**.
- Raw data are always noisy, and this is especially true for image data.

Conclusion

- 1. After every convolution the output is sent to an activation function so as to obtain better features and maintaining positivity eg: ReLu
- 2. Sparse connectivity and weight sharing are the main reason for a convolutional neural network to work
- 3. The concept of choosing a number of filters in between layers and padding and stride and filter dimensions are taken on doing a number of experimentations

Conclusion

- Despite the limits of convolutional neural networks, however, there's no denying that they have caused a revolution in artificial intelligence. Today, CNN's are used in many computer vision applications such as facial recognition, image search, and editing, augmented reality, and more.
- The output from the convolutional layers represents high-level features in the data. While that output could be flattened and connected to the output layer, adding a fully-connected layer is a (usually) cheap way of learning non-linear combinations of these features.
- the main difference between the Convolutional layer and the Dense layer is that Convolutional Layer uses fewer parameters by forcing input values to share the parameters. The Dense Layer uses a linear operation meaning every output is formed by the function based on every input

References and links

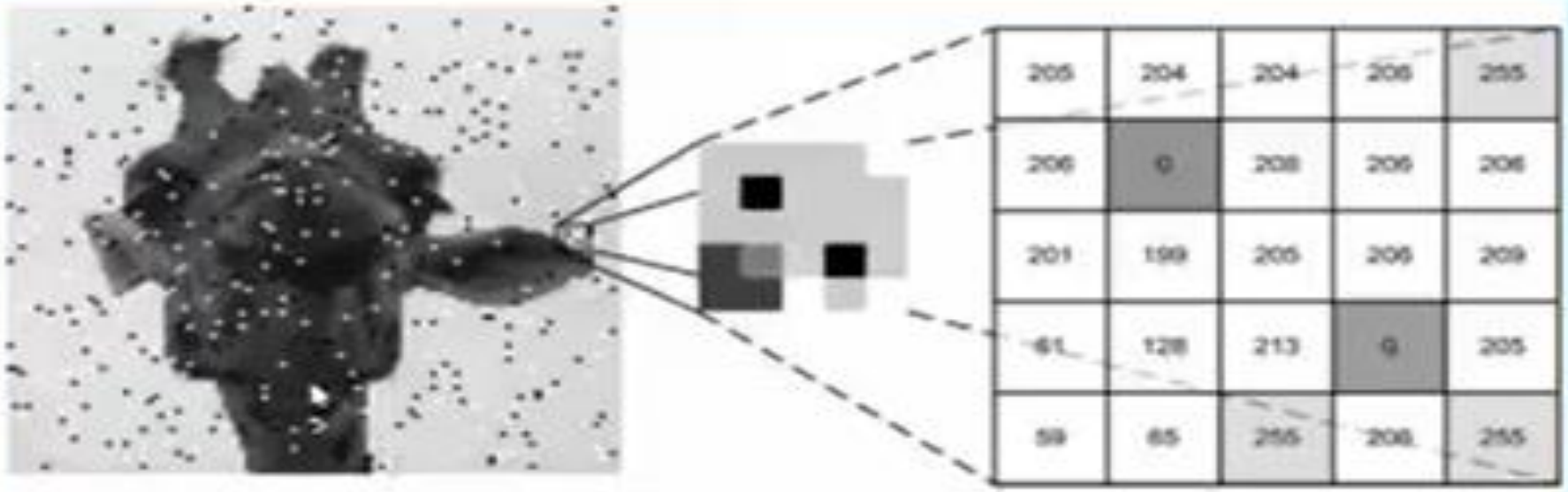
- https://archive.ics.uci.edu/ml/machine-learning-databases/molecular-biology/protein-secondary-structure/
- Medium.com
- Towardsdatascience.com
- Edureka.co
- analyticsvidhya.com
- Medium.com
- Wikipedia

Spatial Filtering: Fundamentals

Input image | 11x11 kernel | 29x29 kernel

Fig. An example of how a mean filter can be used to hide the identity of a person.

Spatial Filtering: Fundamentals



Ordering : [0, 199, 201, 204, *204*, 205, 205, 206, 208]

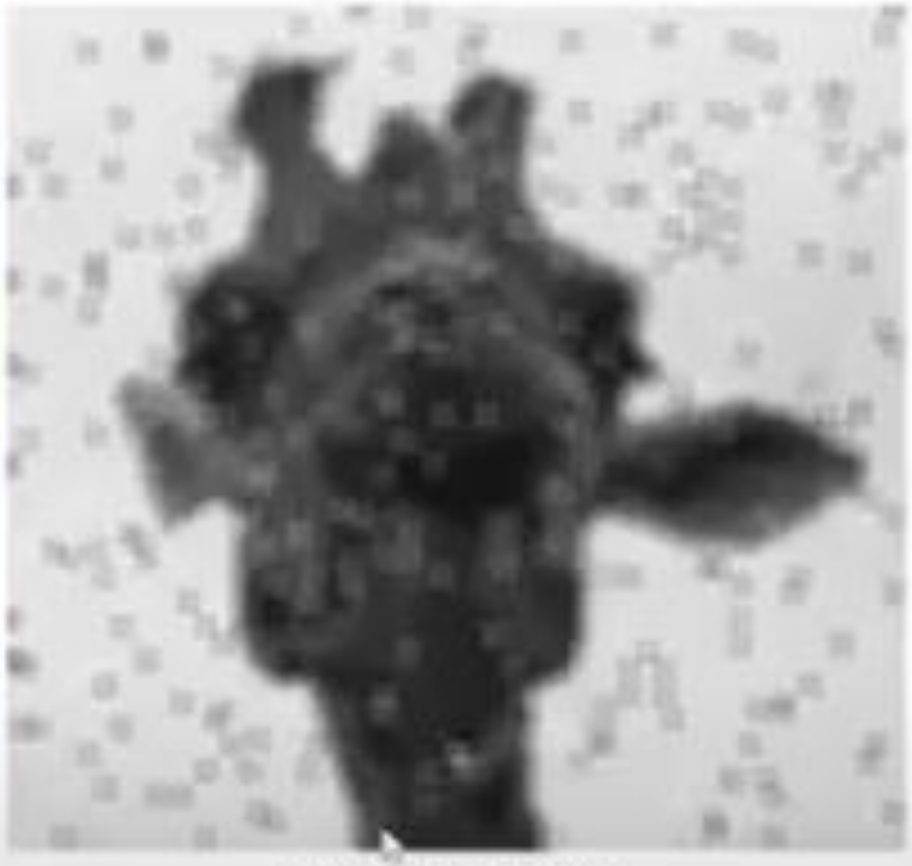
Median = 204

$$\text{Mean value} = \frac{205 + 204 + 204 + 206 + 0 + 208 + 201 + 199 + 205}{9}$$

$$= 181.3 \simeq 181$$

Credits: S. Marsili, EEL484 Lecture notes/slides, IIT Ropar

Spatial Filtering: Fundamentals



Mean filtered

Median filtered