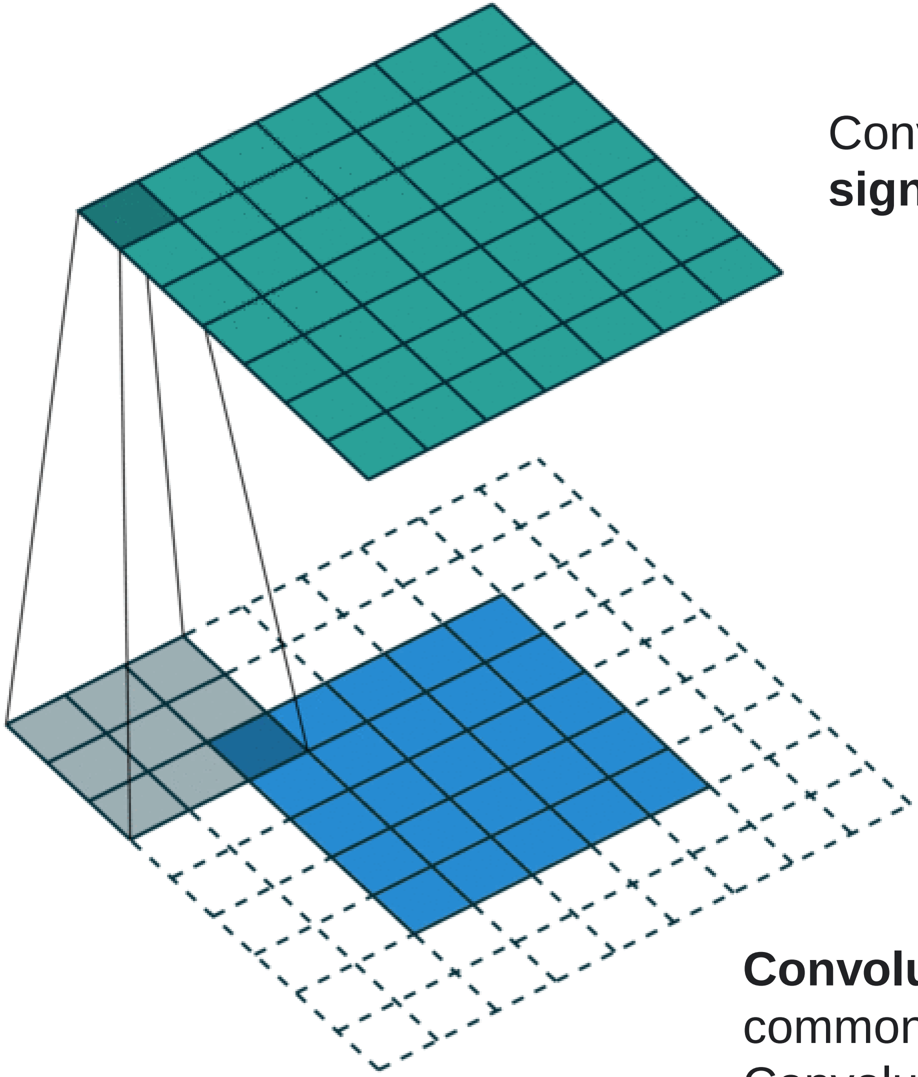
AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

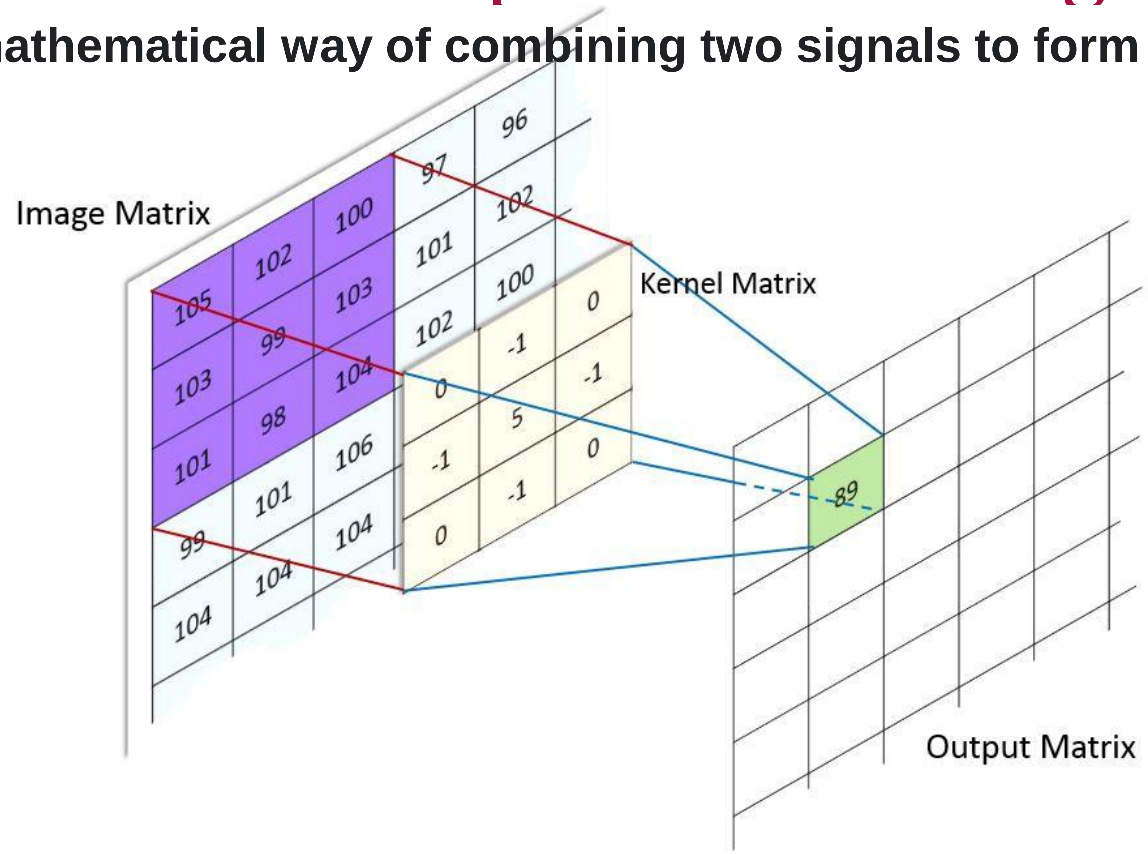
Introduction to Deep Learning

Convolutional Neural Network
(CNN)

Convolution operation in images

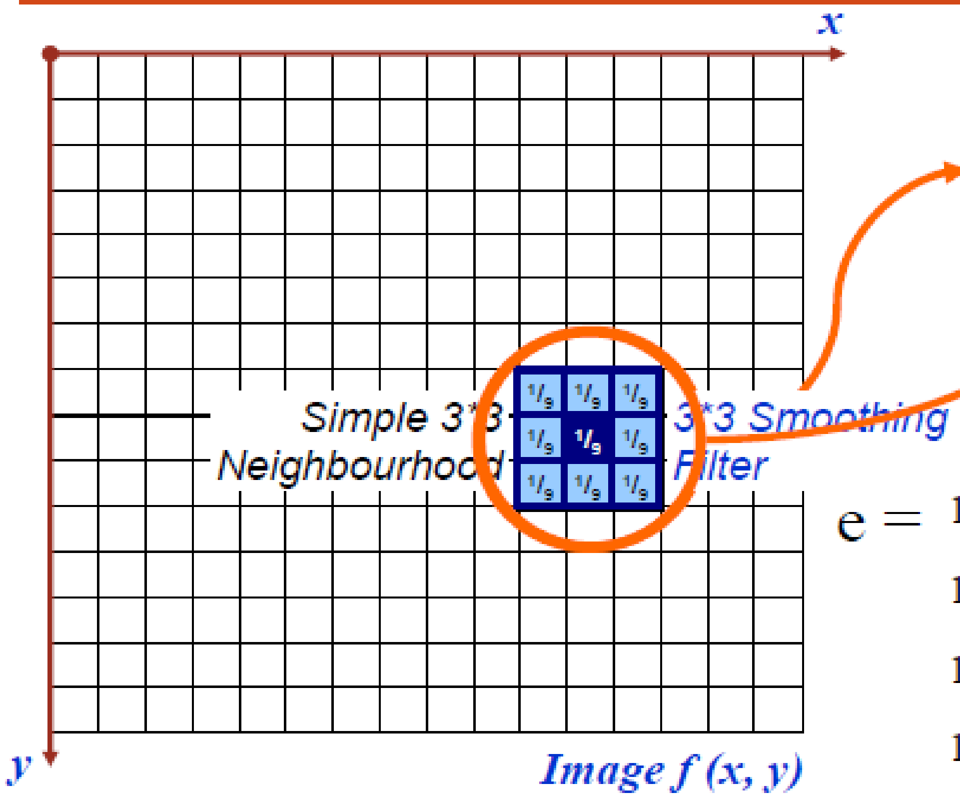
Convolution is **a mathematical way of combining two signals to form a third signal**.





Convolution is a simple mathematical **operation** which is fundamental to many common **image processing operations.**
Convolution is **a mathematical way of combining two signals to form a third signal**. **Convolution** provides a way of `multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality.

Smoothing Spatial Filters



104	100	108
99	106	98
95	90	85

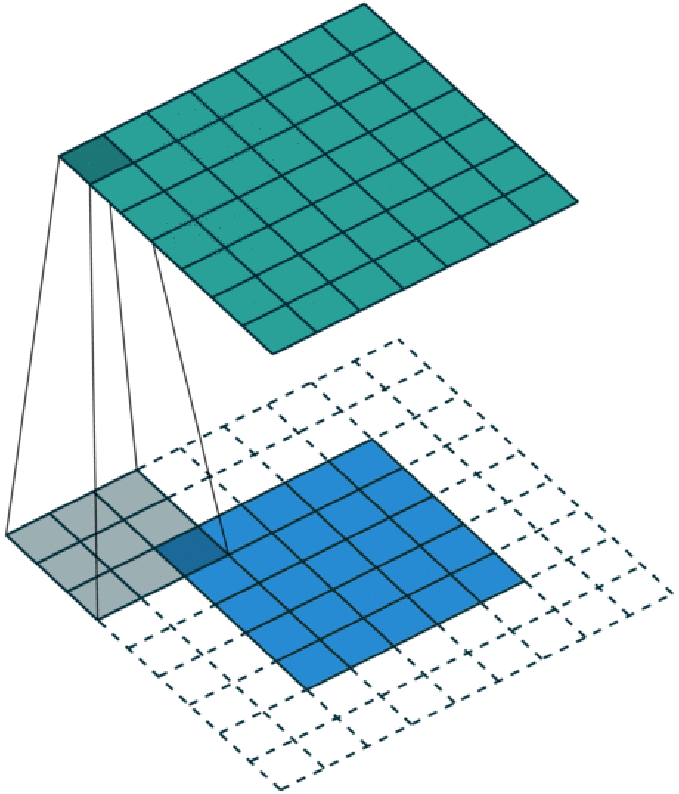
$*$

Original Image Pixels

$$
\begin{aligned}
e = {} & {}^1/_9*106 + \\
& {}^1/_9*104 + {}^1/_9*100 + {}^1/_9*108 + \\
& {}^1/_9*99 + {}^1/_9*98 + \\
& {}^1/_9*95 + {}^1/_9*90 + {}^1/_9*85 \\
= {} & 98.3333
\end{aligned}
$$

- The above is repeated for every pixel in the original image to generate the smoothed image

Convolution operation in images



Image

100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

Kernel/Filter

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

-100
-200
-100
200
400
+200
=400

Sobel Masks

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

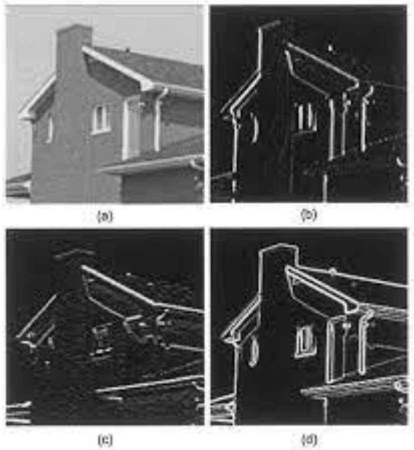
(a)

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

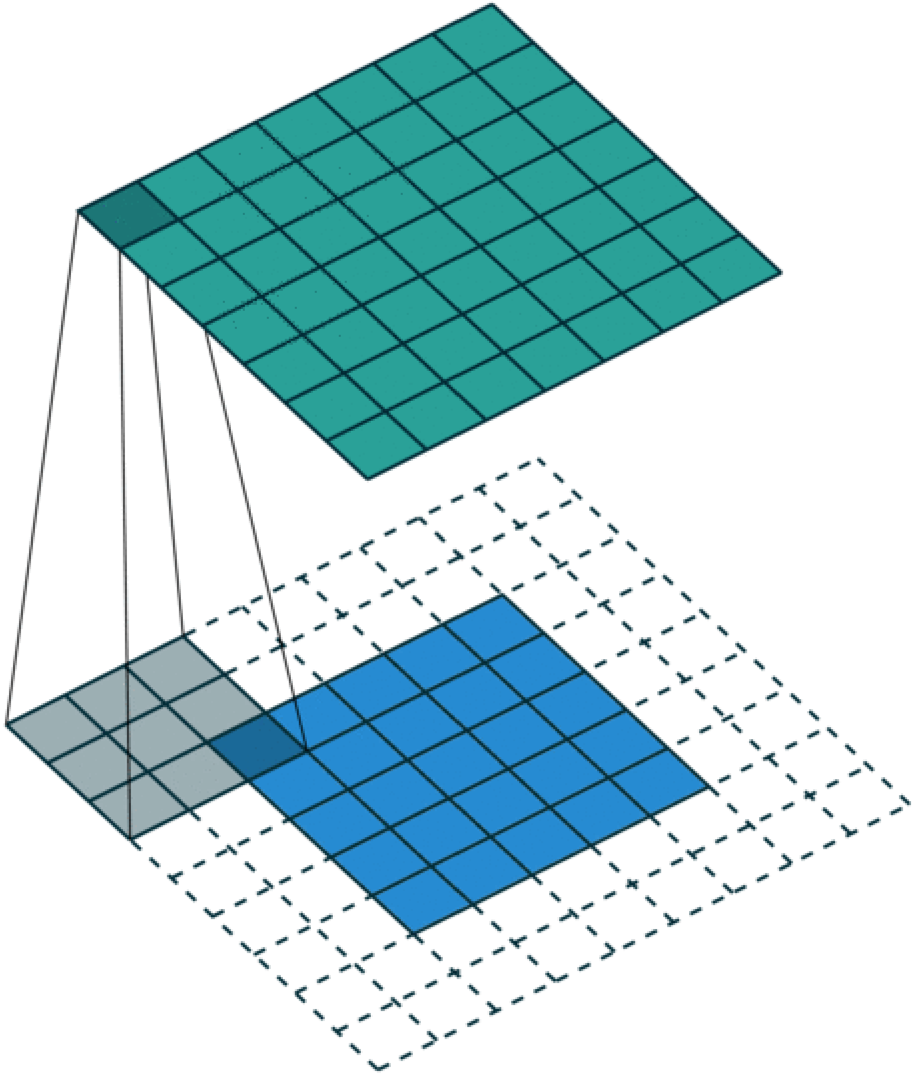
(b)

(a) Horizontal edge Filter (b) Vertical edge Filter

Edge detection using sobel operator



Convolution operation in images



Image

100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

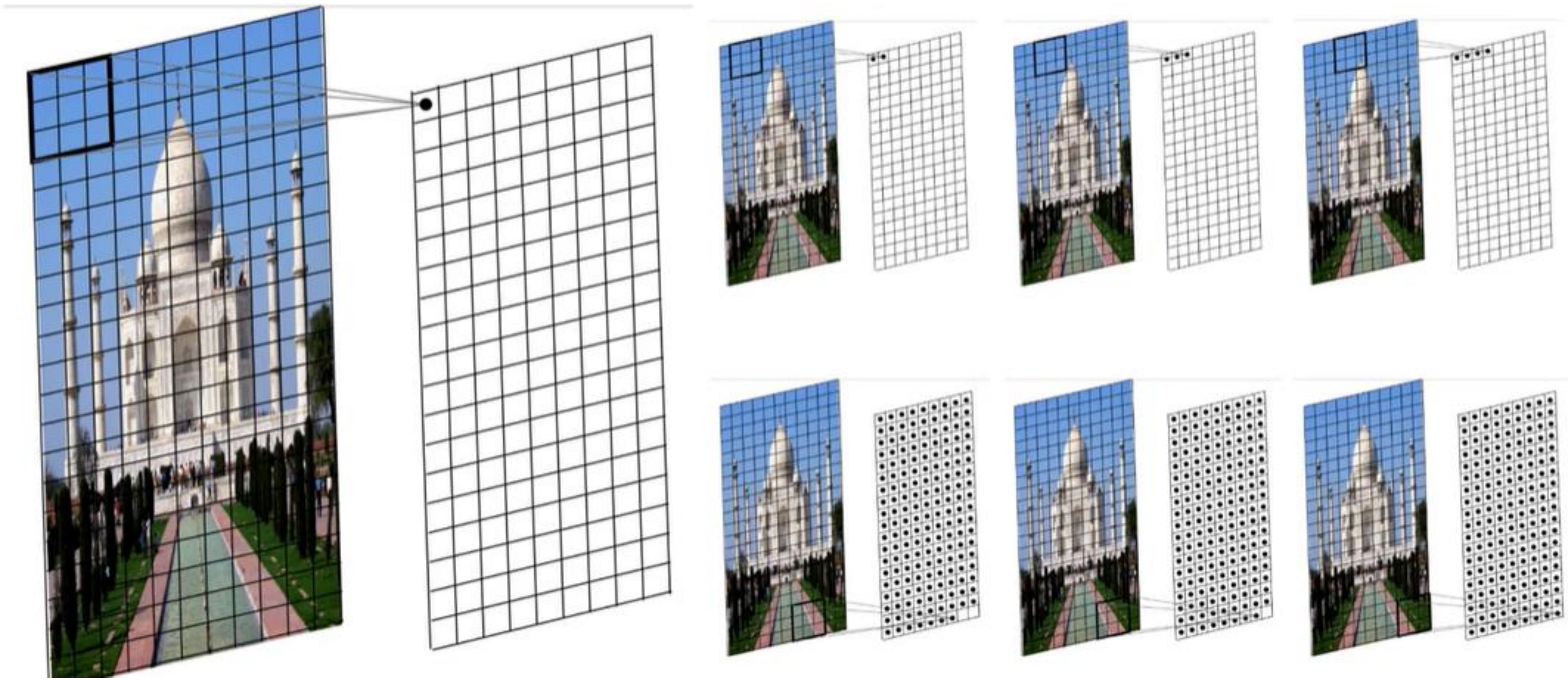
Kernel/Filter

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Average Filter

Blurred image with average filter

2D convolution



Convolution

Convolving an input of 6 X 6 dimension with a 3 X 3 filter results in 4 X 4 output.

•**Input:** n X n
•**Filter size:** f X f
•**Output:** (n-f+1) X (n-f+1)

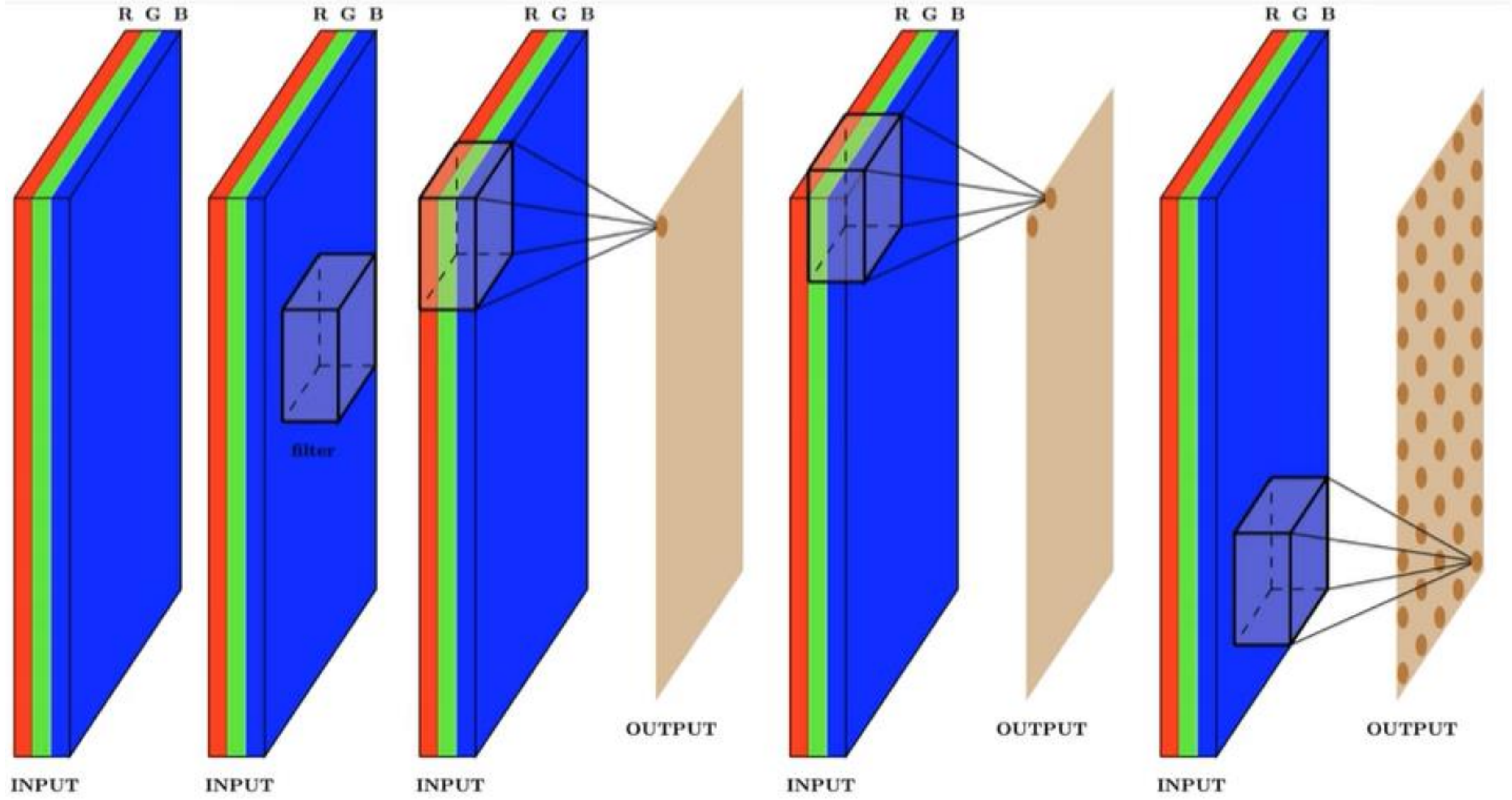
So, convolving a 6 X 6 input with a 3 X 3 filter gave us an output of 4 X 4. Consider one more example:

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

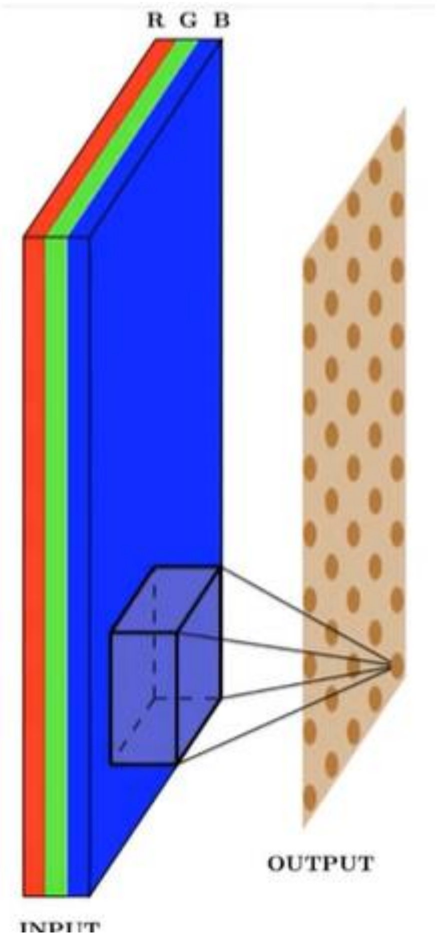
6 X 6 image   3 X 3 filter   4 X 4 matrix

3D Convolution

Input 3D, Filter 3D, Output 2D



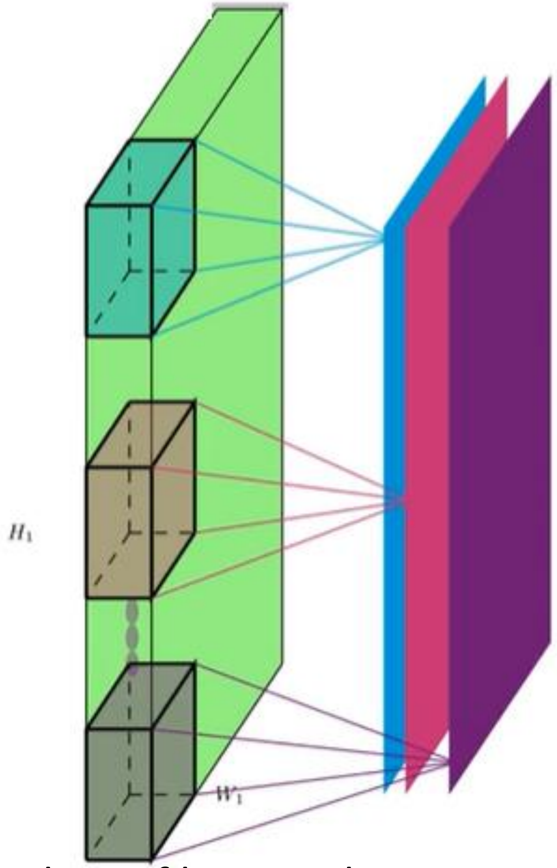
3D Convolution



Important Note

- the input is 3D
- the filter is also 3D
- but the convolution operation that we are performing is 2D
- we are only sliding vertically and horizontally and not along the depth
- this is because the depth of the filter is the same as the depth of the input

3D convolution produce 2D output

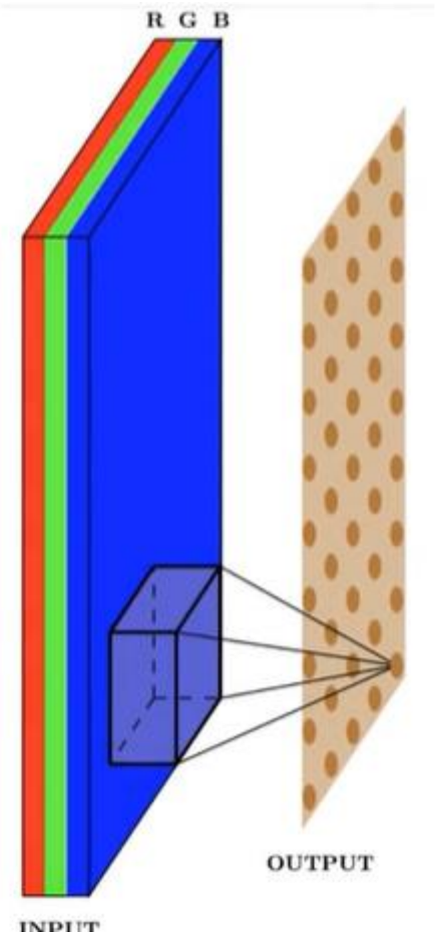


Apply multiple 3D filters to the same image

Important Note

- Each filter applied to a 3D input will give a 2D output
- Combining the output of multiple such filters will result in a 3D output

How to compute Wo,Ho,Do?



Terminology

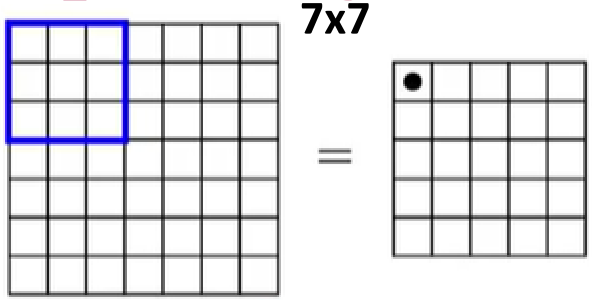
- Input Width (W_I), Height (H_I) and Depth (D_I)
- Output Width (W_O), Height (H_O) and Depth (D_O)
- The spatial extent of a filter (F)
- The number of filters (K)
- Padding (P) and Stride (S)

Question: Given W_I, H_I, D_I, F, K, S and P how do you compute W_O, H_O and D_O ?

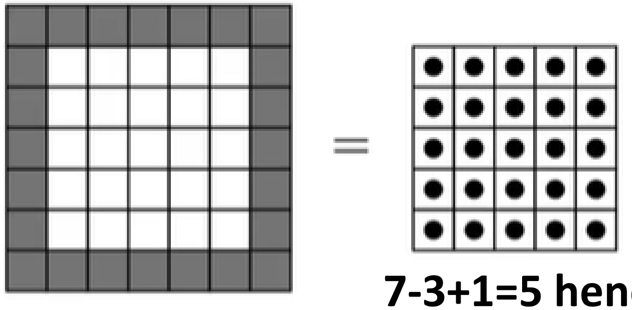
No padding- size of ouput reduce

7x7

5x5

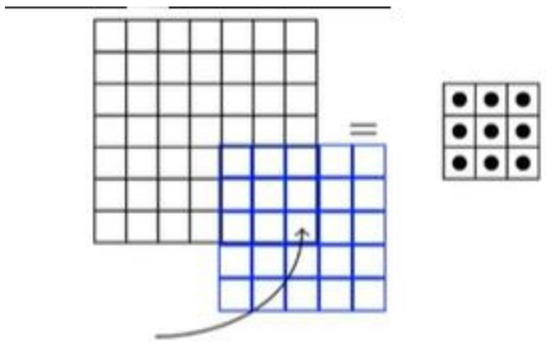


Kernel 3x3



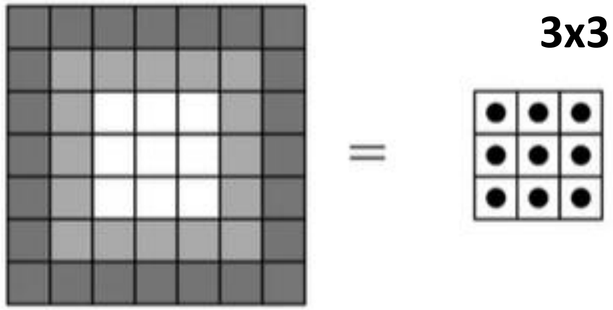
7-3+1=5 hence 5x5

7x7



Kernel 5x5

3x3



7-5+1=3 hence 3x3

- We can't place the kernel at the corners as it will cross the input boundary
- This is true for all the shaded points
- Hence the size of the output will be smaller than that of the input

$$W_O = W_I - F + 1$$

$$H_O = H_I - F + 1$$

- Each filter gives one 2D output
- K filters will give K such 2D outputs
- The depth of the output is the same as the number of filters

Padding

0	0	0	0	0	0	0	0
0	3	3	4	4	7	0	0
0	9	7	6	5	8	2	0
0	6	5	5	6	9	2	0
0	7	1	3	2	7	8	0
0	0	3	7	1	8	3	0
0	4	0	4	3	2	2	0
0	0	0	0	0	0	0	0

$6 \times 6 \rightarrow 8 \times 8$

*

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

3×3

=

-10	-13	1			
-9	3	0			

6×6

- **Without padding**
- **Input:** n X n
- **Filter size:** f X f
- **Output:** (n-f+1) X (n-f+1)

- **With padding**
- **Input:** n X n
- **Padding:** p
- **Filter size:** f X f
- **Output:** (n+2p-f+1) X (n+2p-f+1)

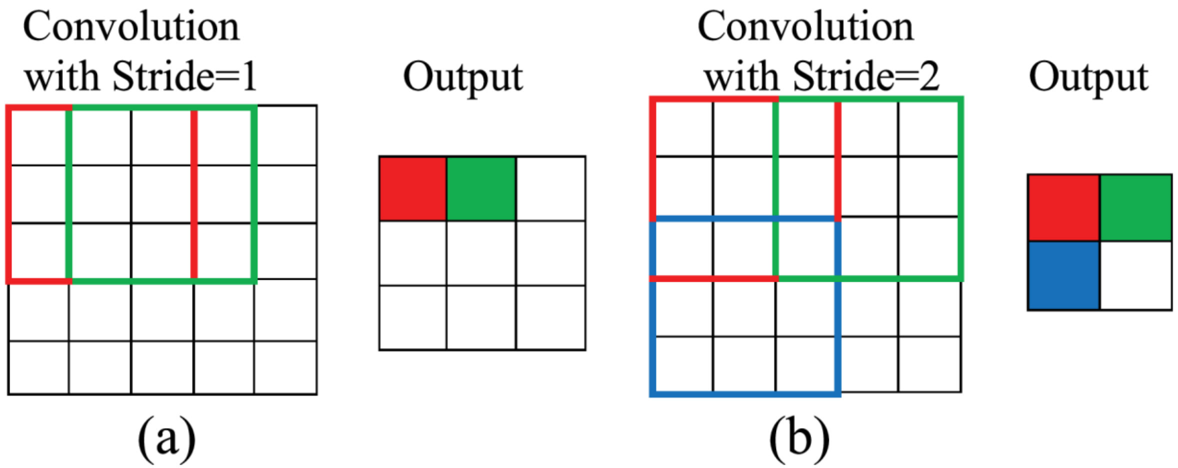
Strided Convolutions

- **Stride** is a parameter of the **neural network's** filter that modifies the amount of movement over the image or video. For example, if a **neural network's stride** is set to 1, the filter will move one pixel, or unit, at a time.

Suppose we choose a stride of 2. So, while convoluting through the image, we will take two steps – both in the horizontal and vertical directions separately. The dimensions for stride *s* will be:

•**Input:** n X n
•**Padding:** p
•**Stride:** s
•**Filter size:** f X f
•**Output:** [(n+2p-f)/s+1] X [(n+2p-f)/s+1]

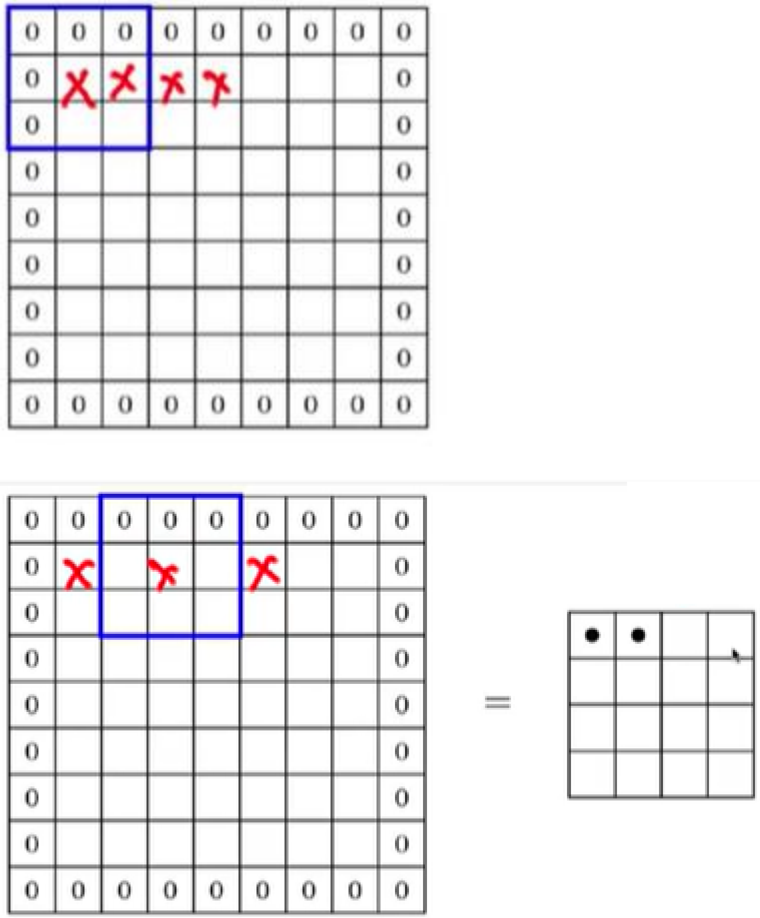
Stride helps to reduce the size of the image, a particularly useful feature.



Stride and padding

defines the interval at which the filter is applied

Higher the Stride ,smaller the size of the output



Wi=9, Hi=9

F=3

P=1

S=2

Wo= $\frac{(9-3+2)}{2}$ **+1= 4**

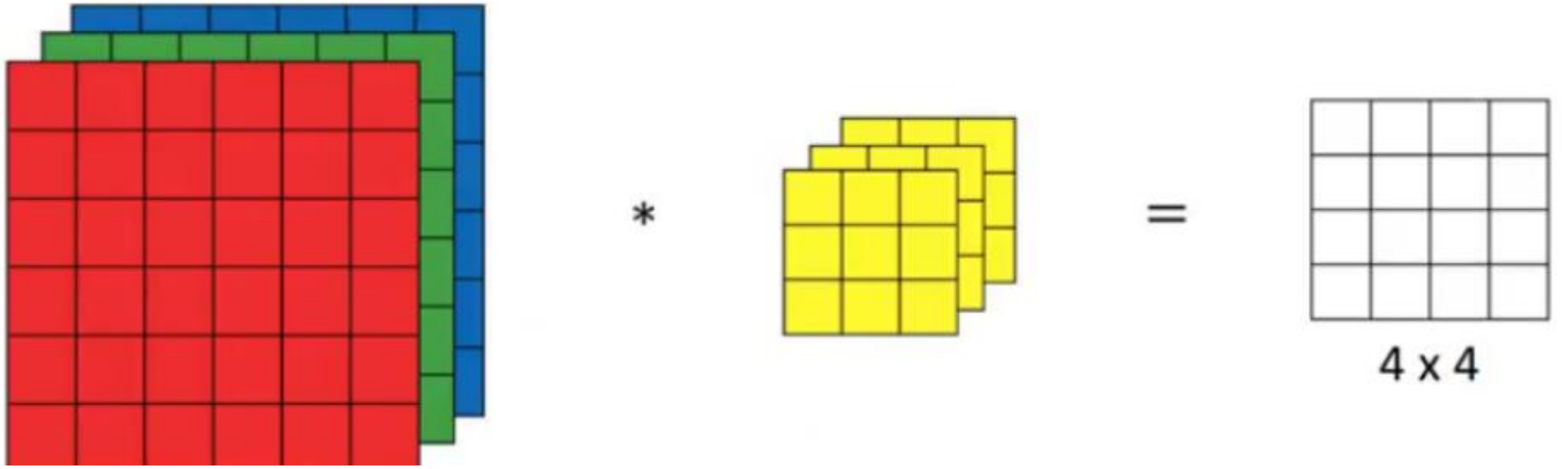
Ho= 4

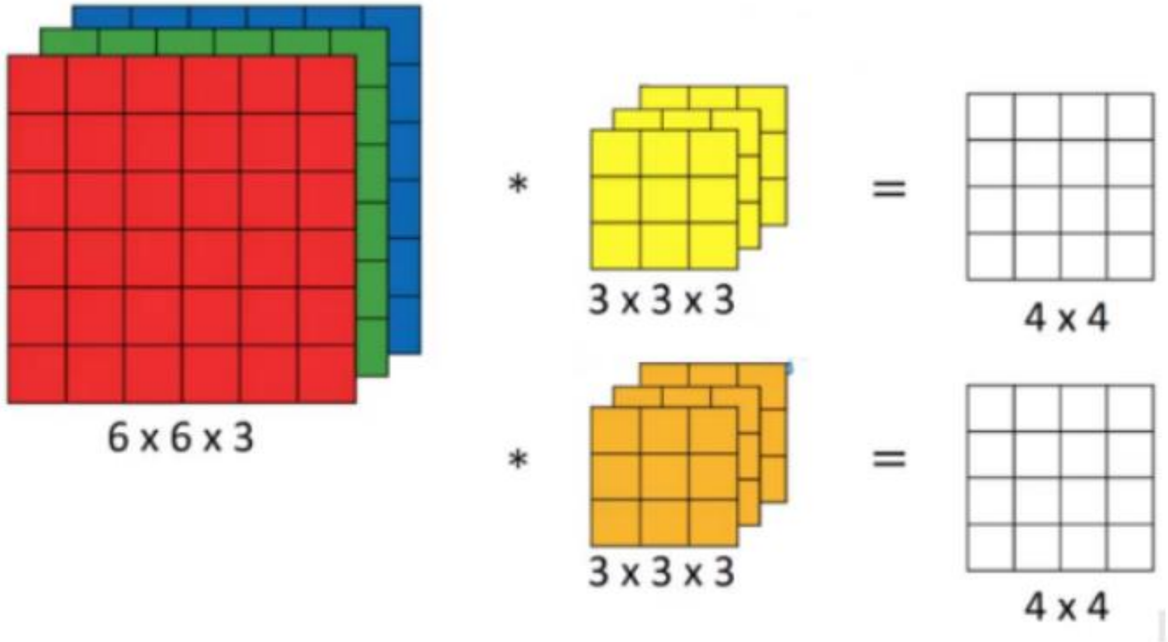
$$W_O = \frac{W_I - F + 2P}{S} + 1$$

$$H_O = \frac{H_I - F + 2P}{S} + 1$$

$$D_O = K$$

Convolutions Over Volume





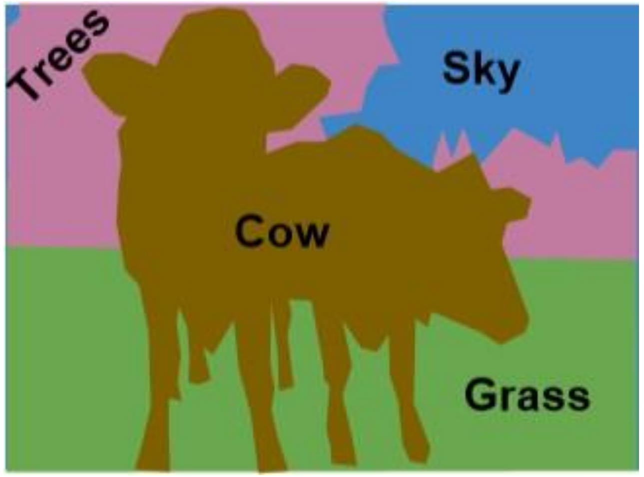
Generalized dimensions can be given as:

- **Input:** n X n X n_c
- **Filter:** f X f X n_c
- **Padding:** p
- **Stride:** s
- **Output:** [(n+2p-f)/s+1] X [(n+2p-f)/s+1] X n_c'
- Here, n_c is the number of channels in the input and filter, while n_c' is the number of filters.

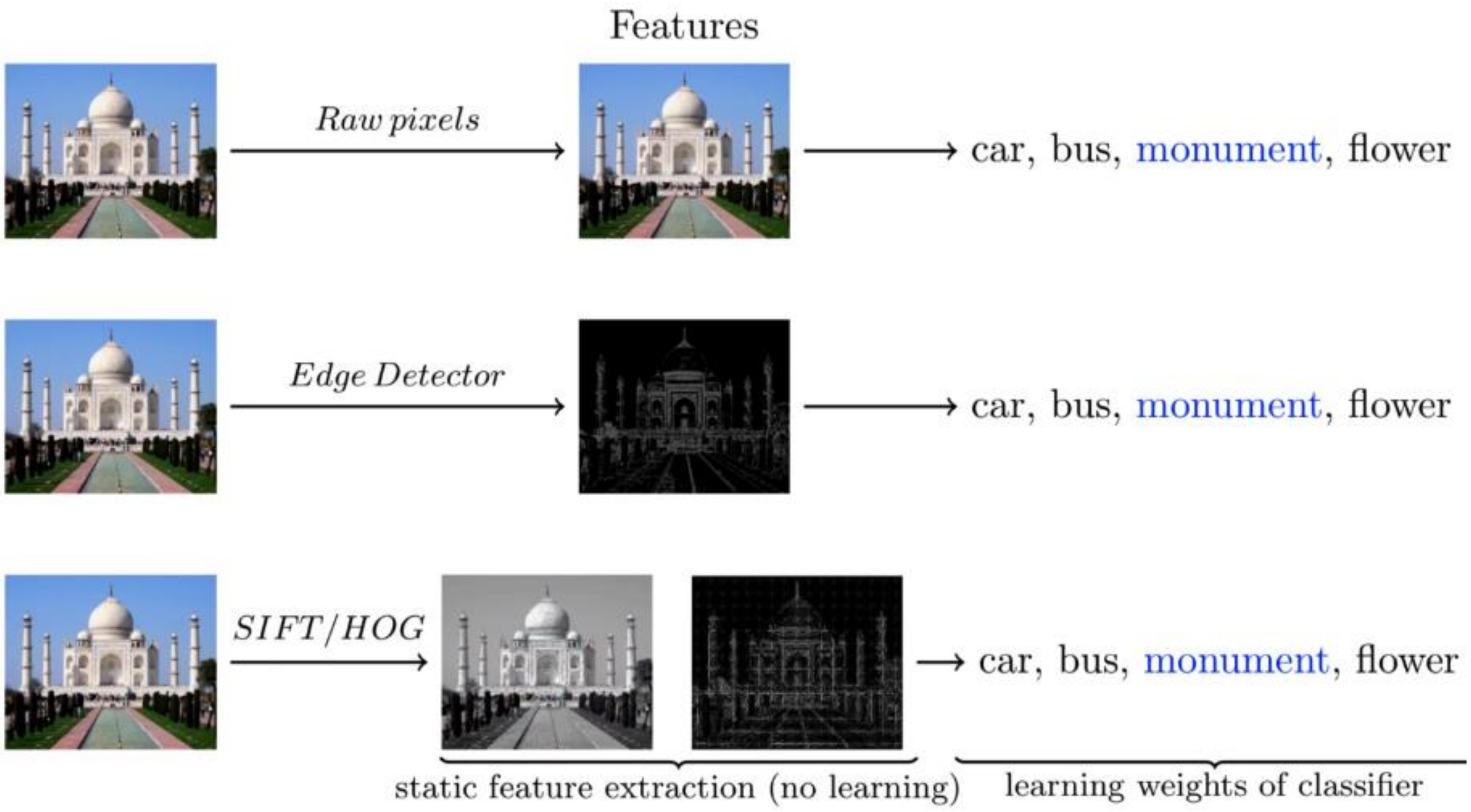
Recollect- Conventional methods

KEY STEPS-IMAGE ANALYSIS

Image Acquisition → Image Enhancement → Image Segmentation → Feature Extraction → Object Recognition/ Machine Learning

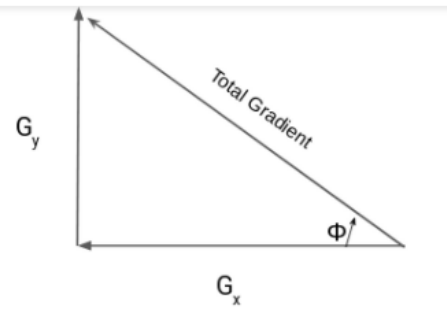


Feature Extraction (Handcrafted features)



Feature Extraction -HOG (Histogram of Gradient)

- The HOG descriptor focuses on the structure or the shape of an object.
- HOG is able to provide the edge direction as well. This is done by extracting the **gradient and orientation** (or you can say magnitude and direction) of the edges
- Additionally, these orientations are calculated in **'localized' portions**. The complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated.
- Finally the HOG would generate a **Histogram** for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'



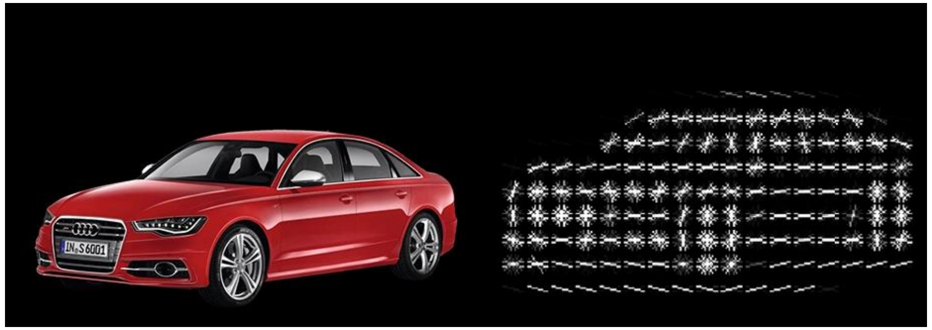
121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

$\tan(\Phi) = Gy / Gx$

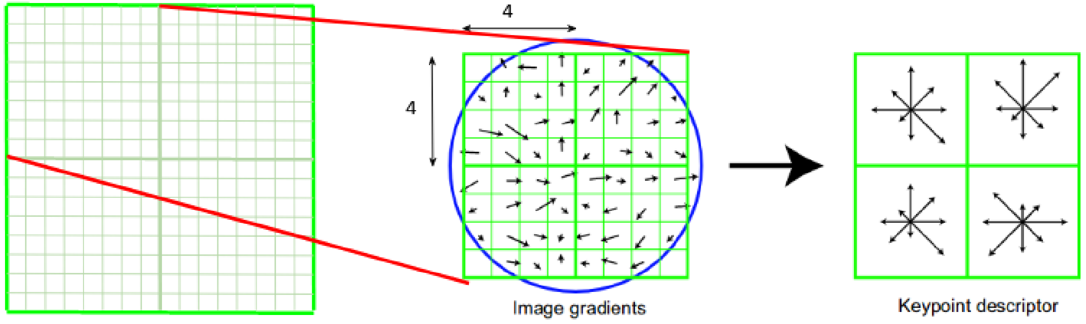
- Change in X direction(G_x) = 89 – 78 = 11
- Change in Y direction(G_y) = 68 – 56 = 8

Total Gradient Magnitude = $\sqrt{[(G_x)^2+(G_y)^2]}$

Total Gradient Magnitude = $\sqrt{[(11)^2+(8)^2]}$ = 13.6

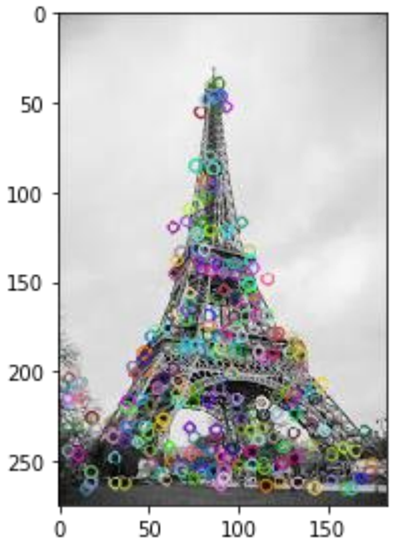


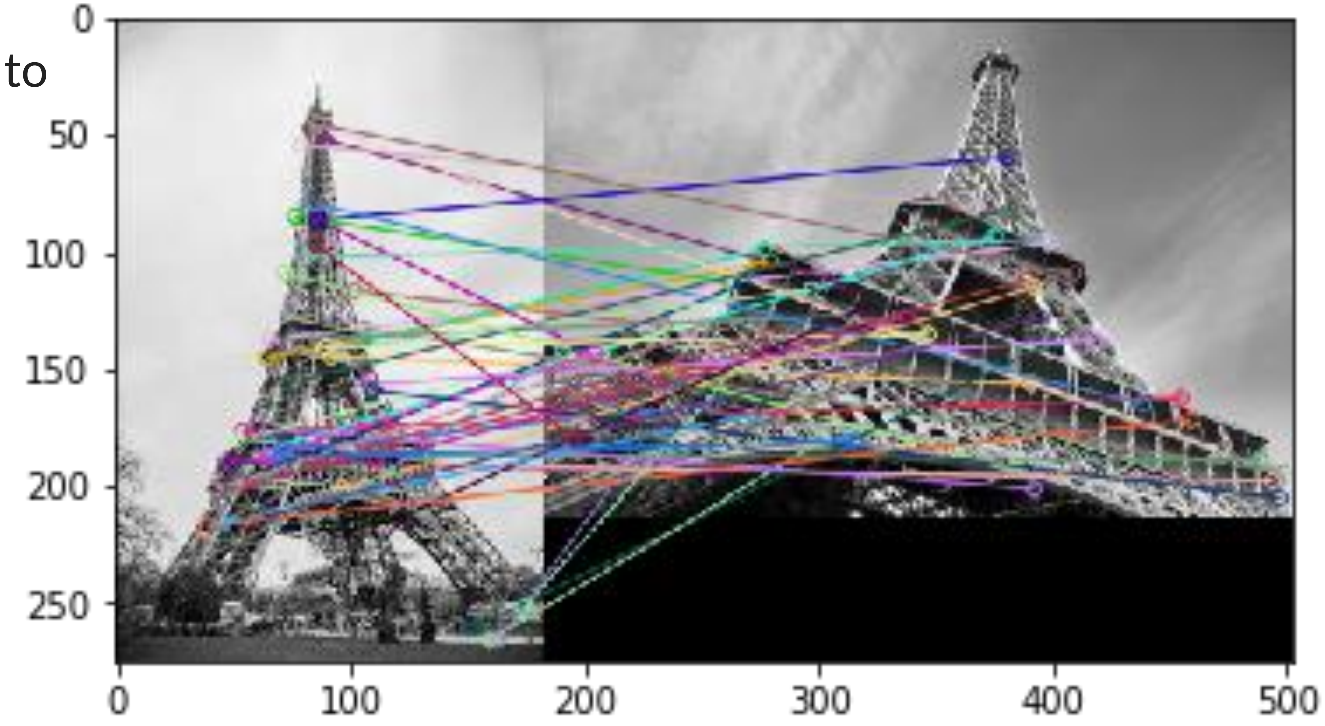
Feature extraction –(SIFT-Scale Invariant Feature Transform)



•**Constructing a Scale Space:** To make sure that features are scale-independent
•**Keypoint Localisation:** Identifying the suitable features or keypoints
•**Orientation Assignment:** Ensure the keypoints are rotation invariant
•**Keypoint Descriptor:** Assign a unique fingerprint to each keypoint

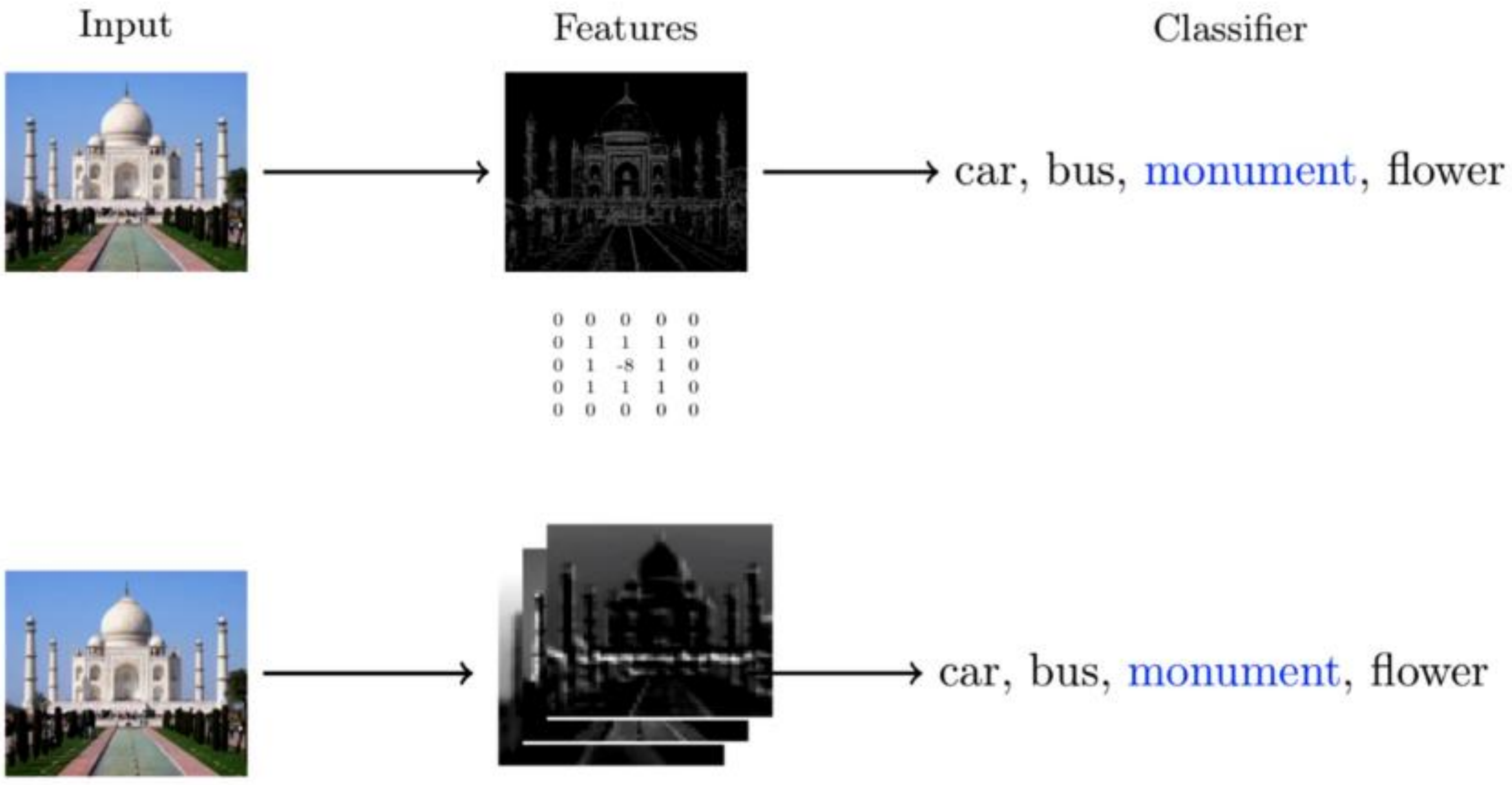
- Rotation Invariant
- Scale Invariant

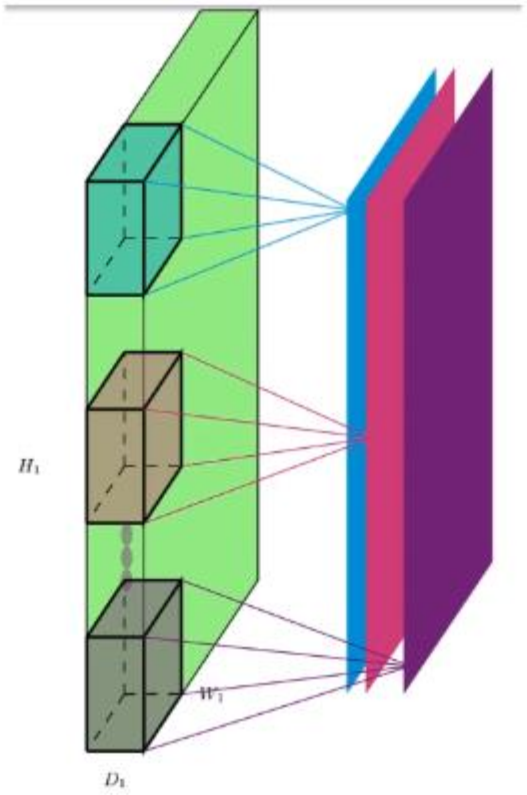




Convolutional Neural Networks

Why not the let the network learn multiple features





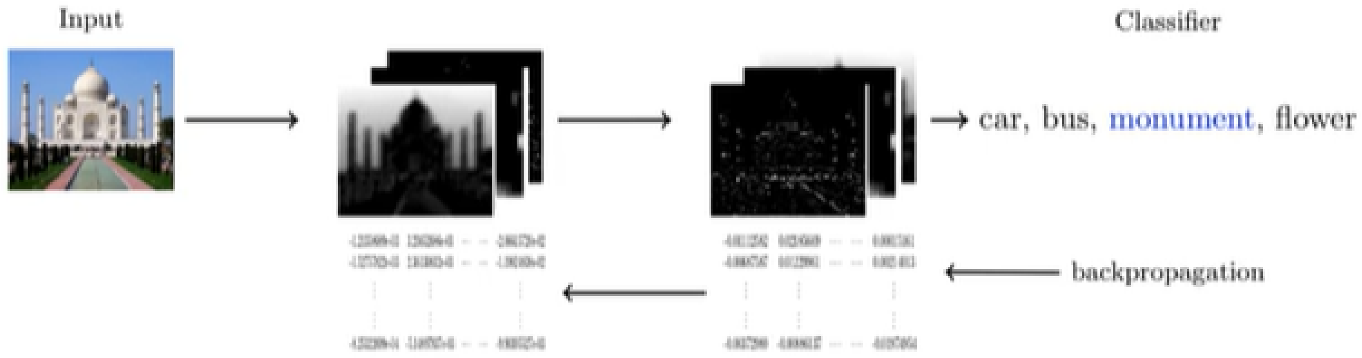
Features are not handcrafted- No explicit feature extraction.
Convolution applied directly on input image !!

Convolutional Neural Networks

Why not the let the network learn multiple layers of feature representations

$$W_O = \frac{W_I - F + 2P}{S} + 1$$

$$H_O = \frac{H_I - F + 2P}{S} + 1$$



Wi x Hi=227x227
F=11 (96 filters of 11x11)
D=3 (Depth of input same as depth of filter)
S=4
P=0

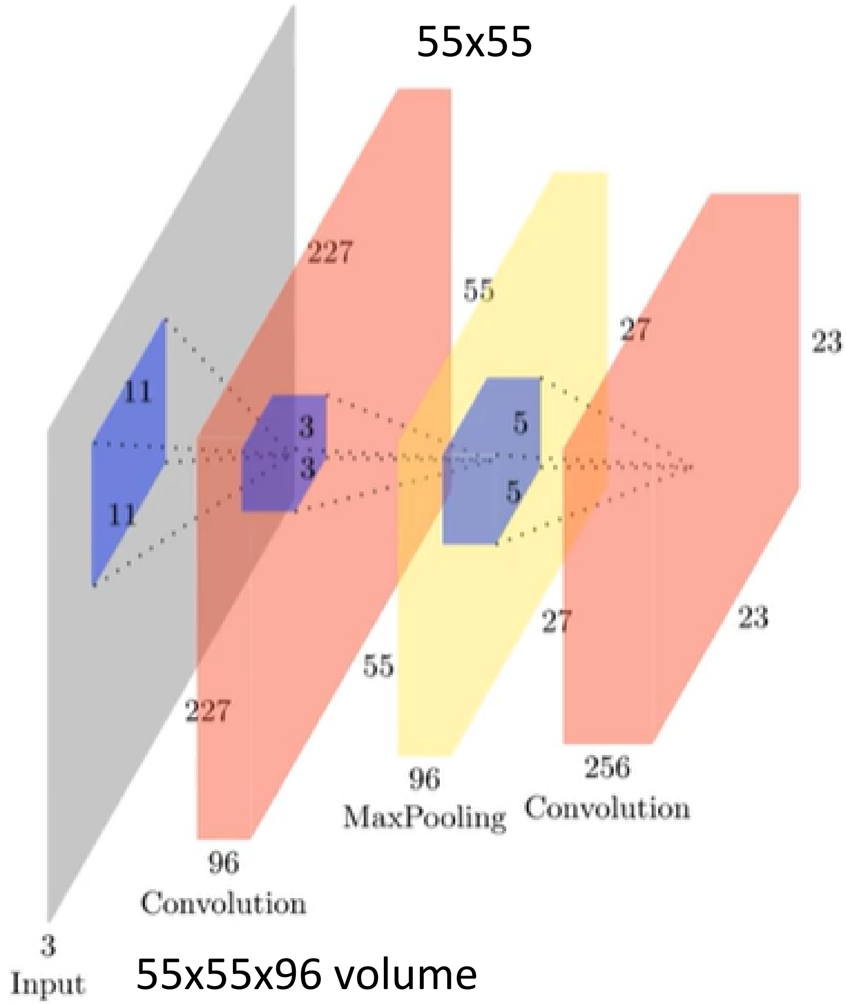
Wo=$\frac{(227-11+0)}{4}$+1= 55
Ho= $\frac{(227-11+0)}{4}$+1= 55
Do = 96 (there are 96 filters)

Depth of filter is always depth of the input

96 filters

227x227

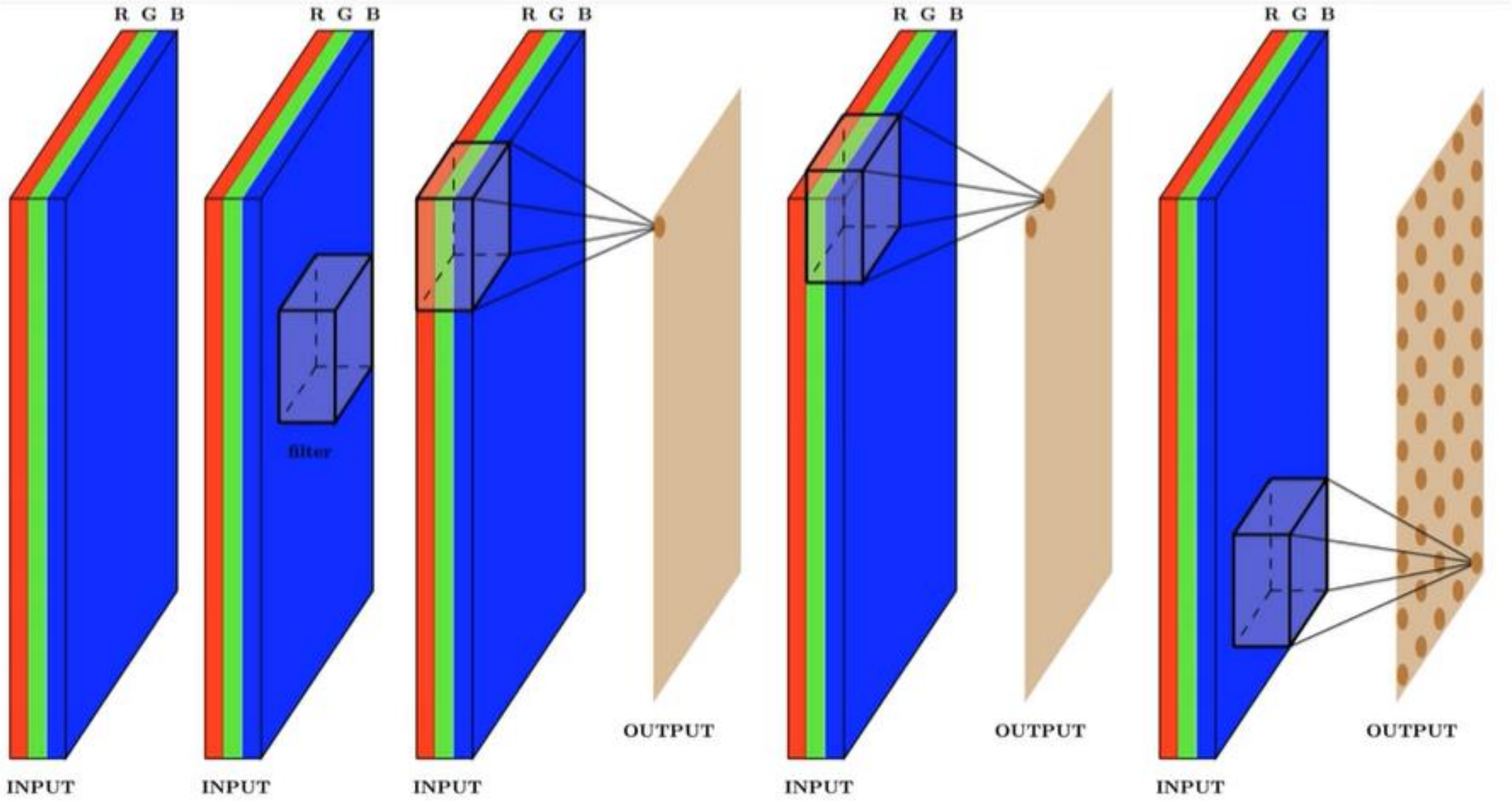
55x55



55x55x96 volume

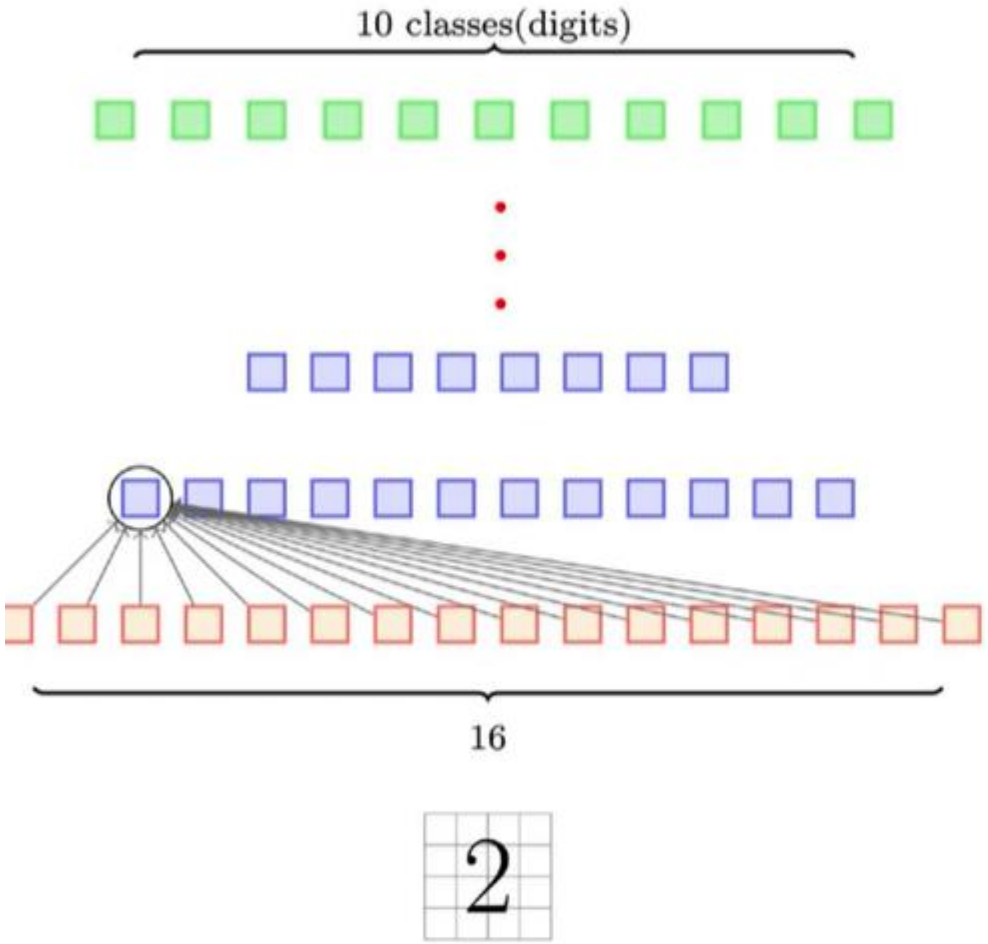
3D Convolution

Input 3D, Filter 3D, Output 2D

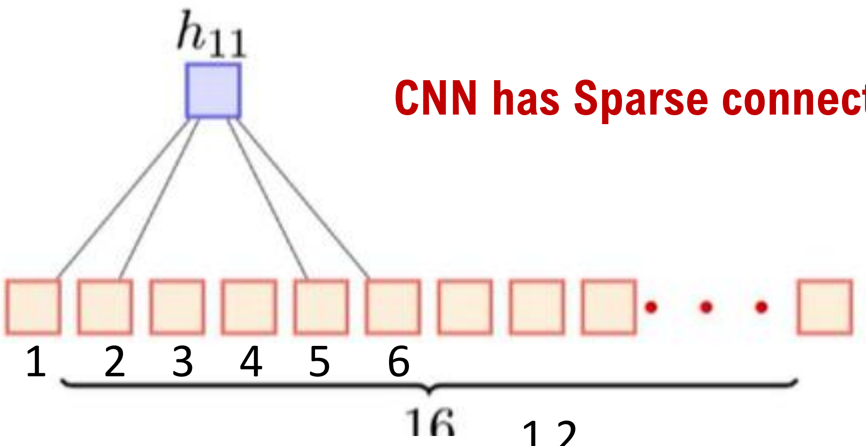


Depth of filter is always depth of the input

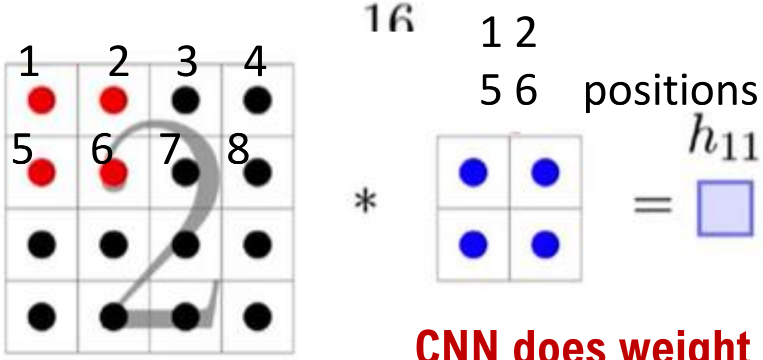
Sparse Connectivity and Weight sharing of CNN



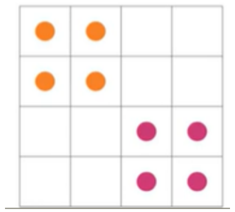
Fully connected- Deep NN



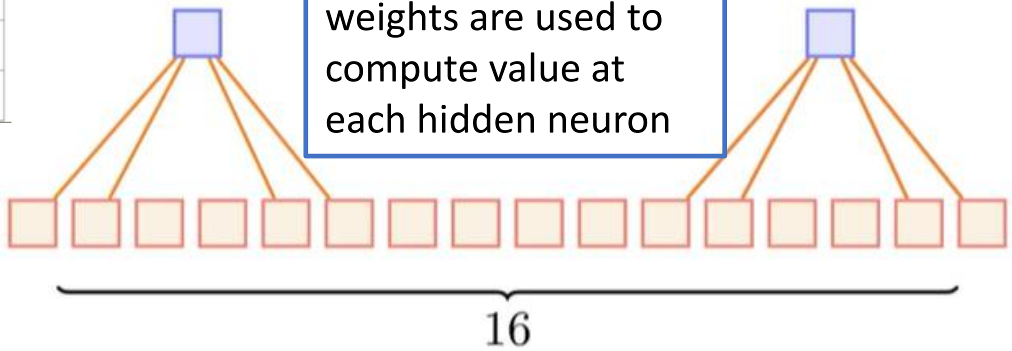
CNN has Sparse connections



CNN does weight sharing



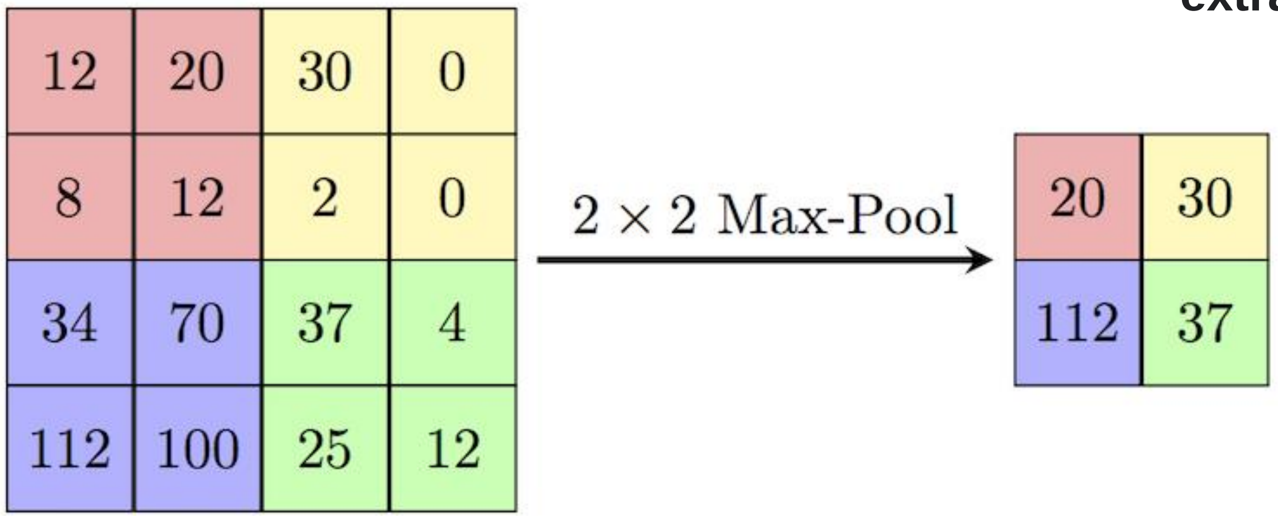
As the kernel slides over input, the same weights are used to compute value at each hidden neuron



Pooling (Max Pooling)

- Pooling layers are generally used to reduce the size of the inputs and hence speed up the computation. Consider a 4 X 4 matrix as shown below:

Max Pooling is **a convolution process where the Kernel extracts the maximum value of the area it convolves**.



Max-pooling **helps in extracting low-level features like edges, points**, etc.

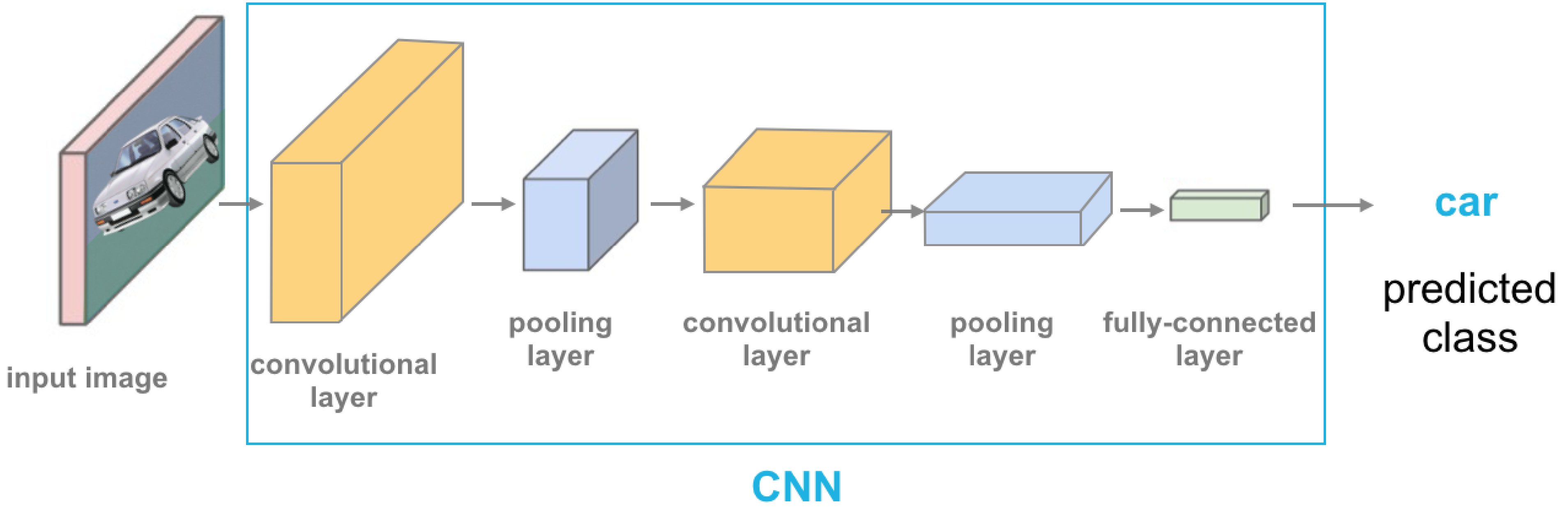
Average Pooling is another pooling operation that calculates the average value for patches of a feature map, and **uses it to create a downsampled (pooled) feature map**. It is used after a convolutional layer.- Avg-pooling goes for smooth features.

Convolutional Neural Network

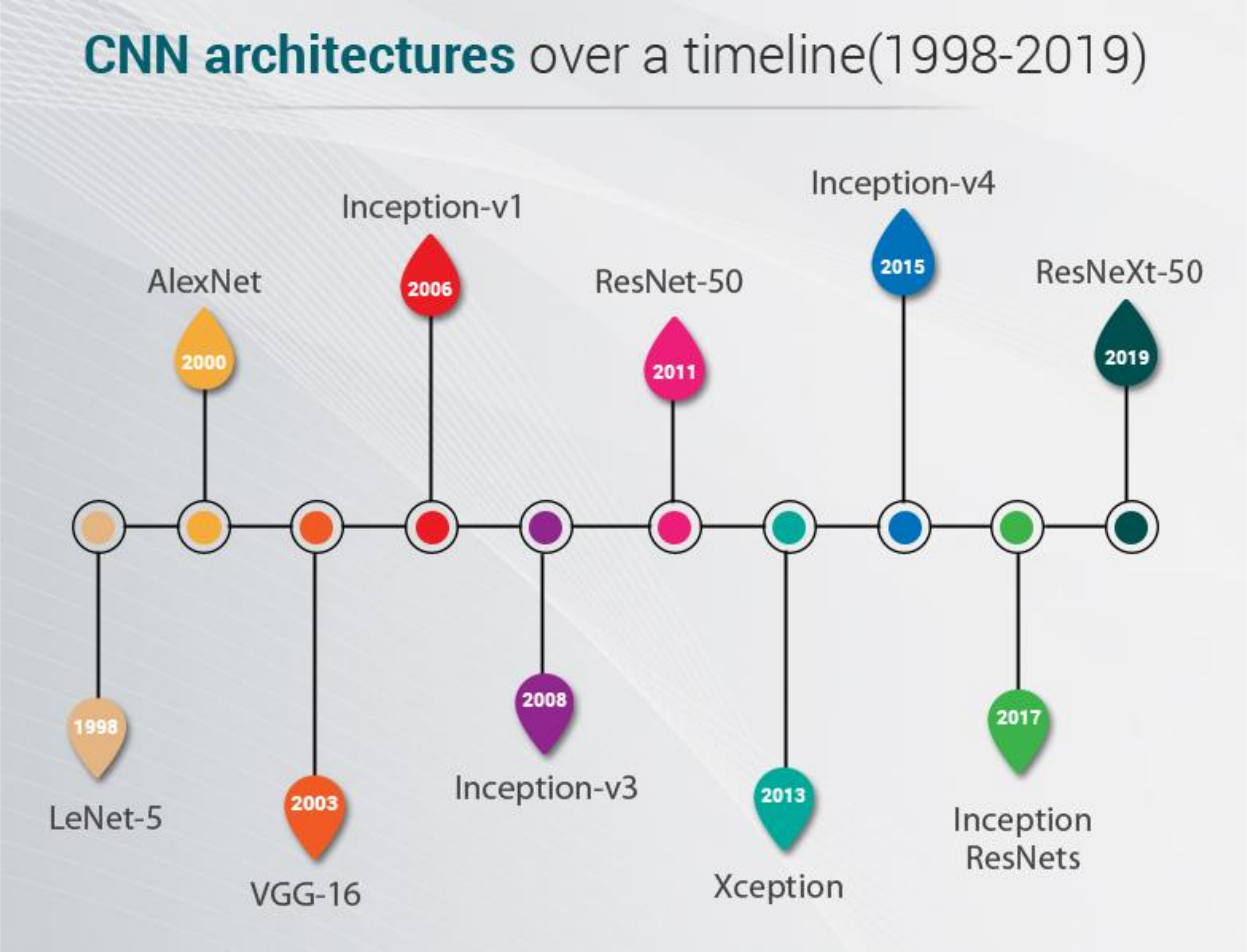
- In a convolutional network (ConvNet), there are basically three types of layers:
    1. Convolution layer
    2. Pooling layer
    3. Fully connected layer

After each convolution there is non-linearity applied using activation functions- Relu/Leaky Relu. H1=g(a1)

.



Different CNN Architectures

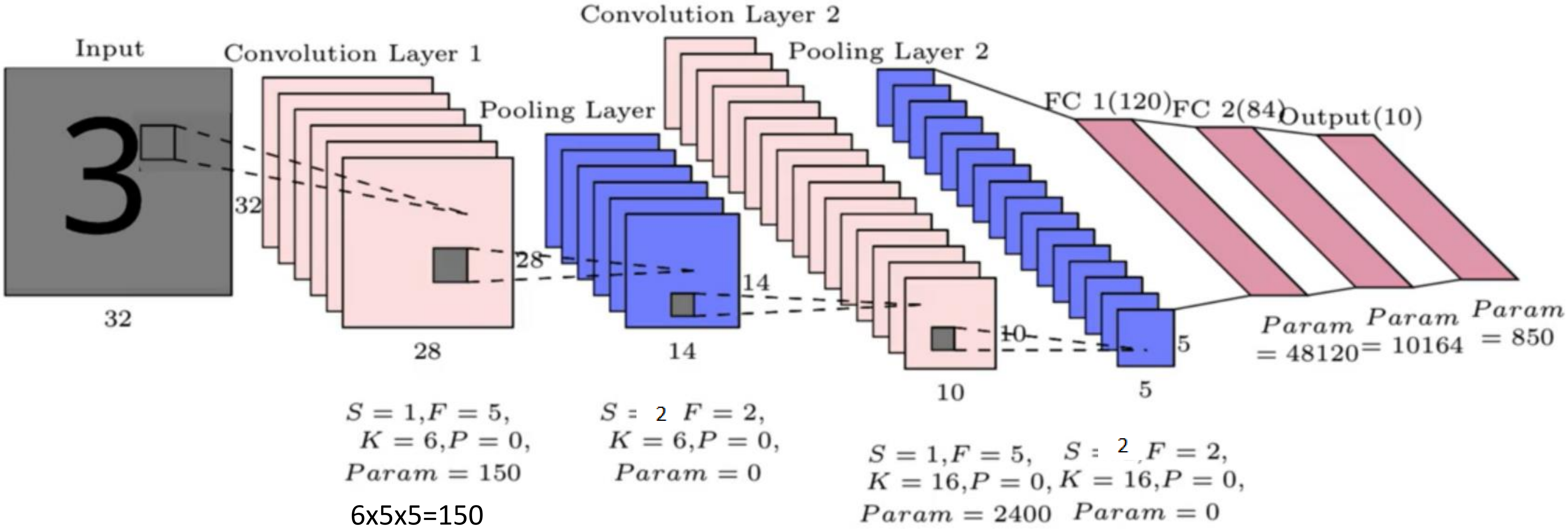


- **No: of Layers** :How many convolutional, max pooling fully connected layers?
- **No: of Filters in each layer**
- **Filter Size**
- **Max pooling:** What arrangement? 2 convolutional and then maxpooling or alternate convolutional and maxpooling?

Use standard tried and tested architectures!

LeNet-5 – First CNN Architecture

Earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.



6x5x5=150

5x5x6x16=2400

Depth of filter is always depth of the input

After each convolution there is non-linearity applied using activation functions- Relu/Leaky Relu. h1=g(a1)

LeNet-5 – First CNN Architecture

Calculation of no of parameters

Conv Layer 1

$$S = 1, F = 5, K = 6, P = 0, Param = 150$$

S=1,F=5,K=6
W0=(32+0-5)/1+1=28,
As no of filters 6 we get 6 outputs of 28x28 (Filter size 5x5)
No: of parameters= 5x5x6=150

[Comparing with fully connected
Flatten 32x32
Flatten 28x28x6
Total parameters =
32x32x28x28x6]

Conv Layer 2

$$S = 1, F = 5, K = 16, P = 0, Param = 2400$$

S=1,F=5,K=16
W0=(14+0-5)/1+1=10,
As no of filters 16 we get 16 outputs of 10x10 from 6 input images(Filter size 5x5)

No: of parameters= 5x5x6x16=2400

[Comparing with fully connected
14x14x6x10x10x16]

FC1

Flatten 5x5x16=400 neurons
Hidden layer size=120
Bias from each node comes to 120
Parameters = 400x120+120=48120

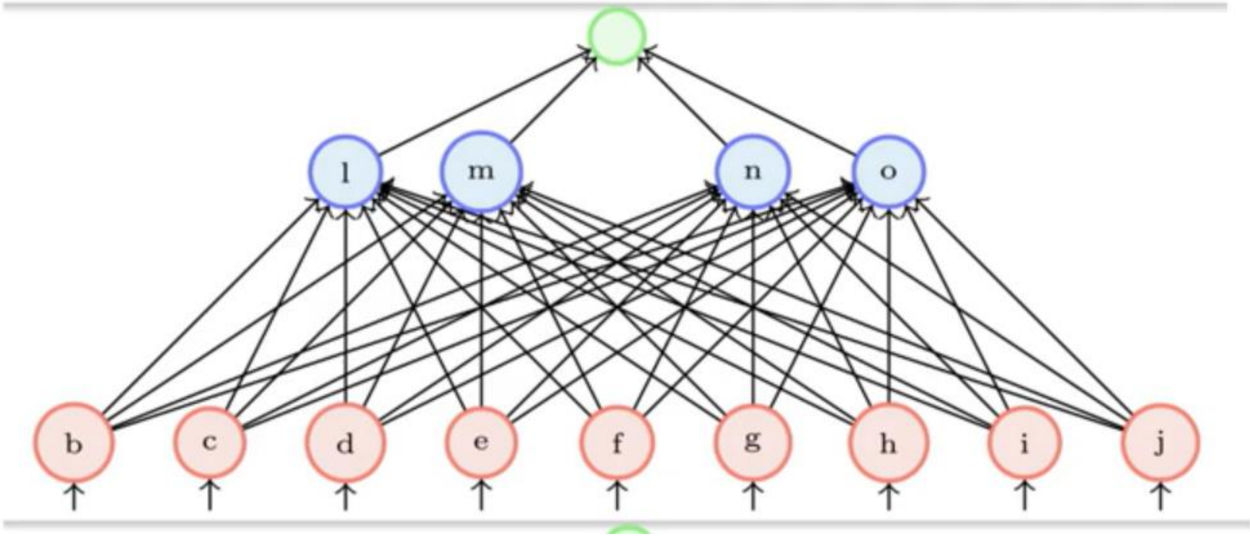
FC2

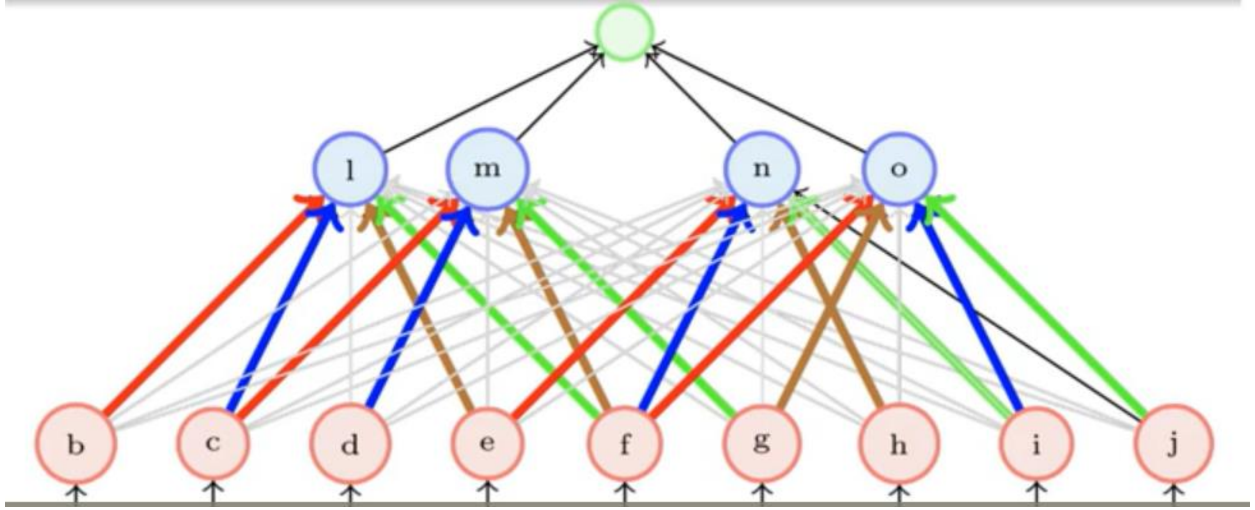
Hidden layer 1 size= 120
Hidden layer2 size =84
Bias from each of 84 hidden layer neuron=84
No of parameters=
120x84+84=10164

FC3

Hidden layer 1 size= 84
Hidden layer2 size =10
Bias from each of 10 hidden layer neuron=10
No of parameters= 84x10=850

How to train a Convolutional Neural Network

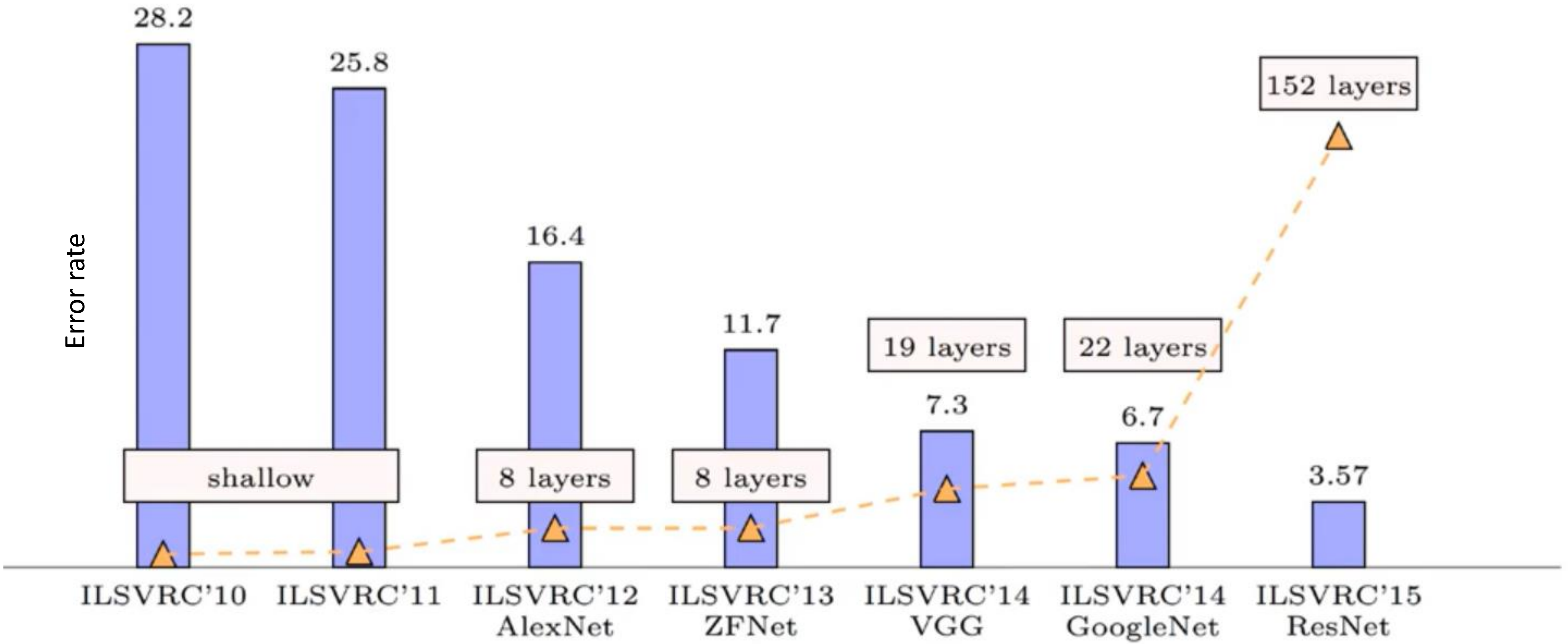




- A CNN can be implemented as a feedforward network
- wherein only a few weights (in color) are active
- the rest of the weights (in gray) are zero

Deep learning frameworks have optimized codes which does not have to do the zero weight calculations or storage

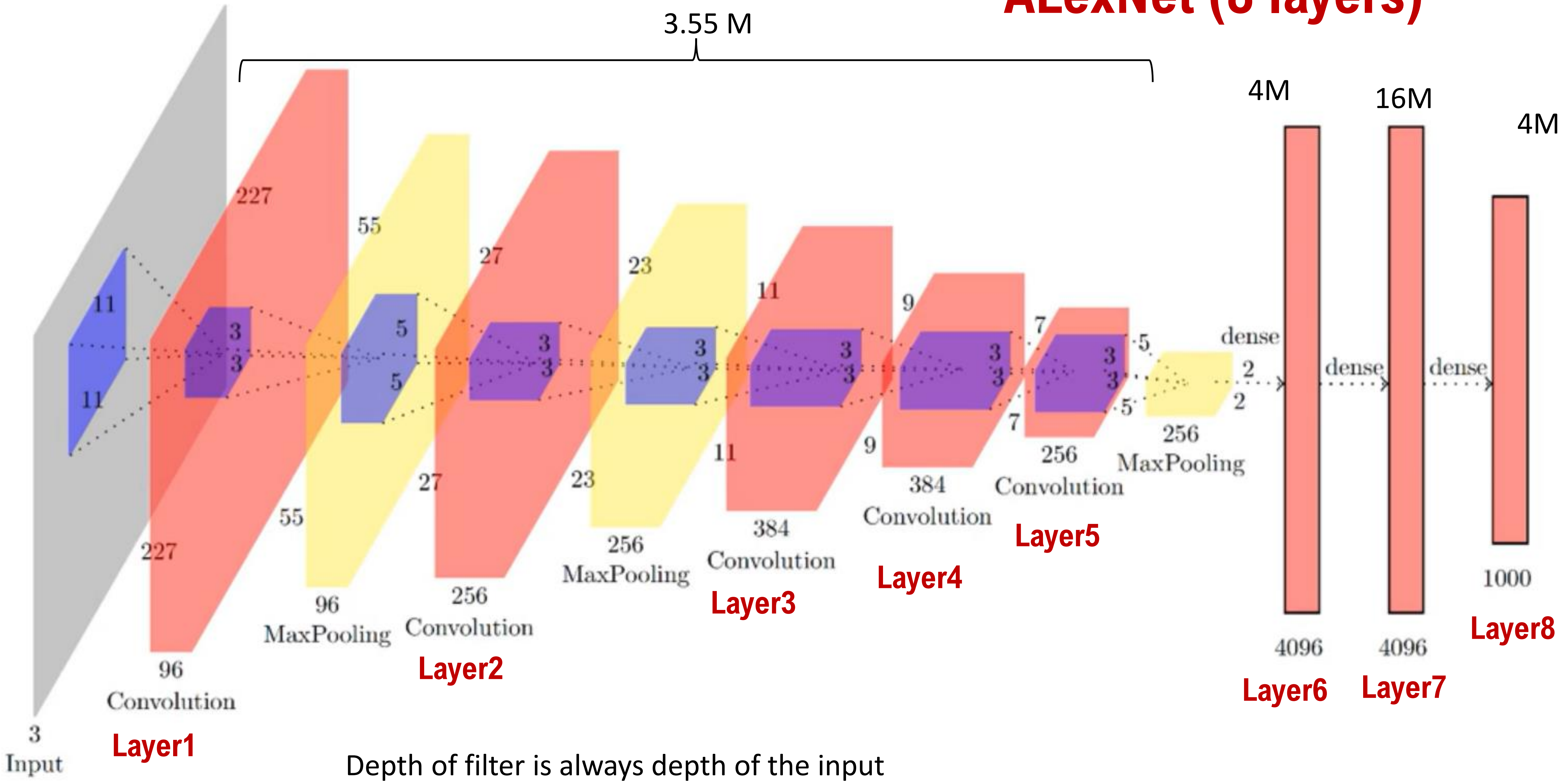
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



https://image-net.org/challenges/LSVRC/index.php

AlexNet is the name of a convolutional neural network (CNN) architecture, designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton, who was Krizhevsky's Ph.D. advisor in the year2000

ALexNet (8 layers)



Depth of filter is always depth of the input

ALexNet (8 layers)

Input: $227 \times 227 \times 3$
Conv1: $K = 96, F = 11$
$S = 4, P = 0$
Output: $W_2 = 55,\ H_2 = 55$
Parameters: $(11 \times 11 \times 3) \times 96 = 34K$

Max Pool Input: $55 \times 55 \times 96$
$F = 3, S = 2$
Output: $W_2 = 27,\ H_2 = 27$
Parameters: 0

Input: $27 \times 27 \times 96$
Conv1: $K = 256, F = 5$
$S = 1, P = 0$
Output: $W_2 = 23,\ H_2 = 23$
Parameters: $(5 \times 5 \times 96) \times 256 = 0.6M$

Max Pool Input: $23 \times 23 \times 256$
$F = 3, S = 2$
Output: $W_2 = 11,\ H_2 = 11$
Parameters: 0

Input: $11 \times 11 \times 256$
Conv1: $K = 384, F = 3$
$S = 1, P = 0$
Output: $W_2 = 9,\ H_2 = 9$
Parameters: $(3 \times 3 \times 256) \times 384 = 0.8M$

Input: $9 \times 9 \times 384$
Conv1: $K = 384, F = 3$
$S = 1, P = 0$
Output: $W_2 = 7,\ H_2 = 7$
Parameters: $(3 \times 3 \times 384) \times 384 = 1.327M$

Input: $7 \times 7 \times 384$
Conv1: $K = 256, F = 3$
$S = 1, P = 0$
Output: $W_2 = 5,\ H_2 = 5$
Parameters: $(3 \times 3 \times 384) \times 256 = 0.8M$

Max Pool Input: $5 \times 5 \times 256$
$F = 3, S = 2$
Output: $W_2 = 2,\ H_2 = 2$
Parameters: 0

FC1
Parameters: $(2 \times 2 \times 256) \times 4096 = 4M$

FC1
Parameters: $4096 \times 4096 = 16M$

FC1
Parameters: $4096 \times 1000 = 4M$

Total Parameters: $27.55M$

Next Class we will see other CNN Architectures

Namah Shivaya