UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Divija L,
Teaching Assistant

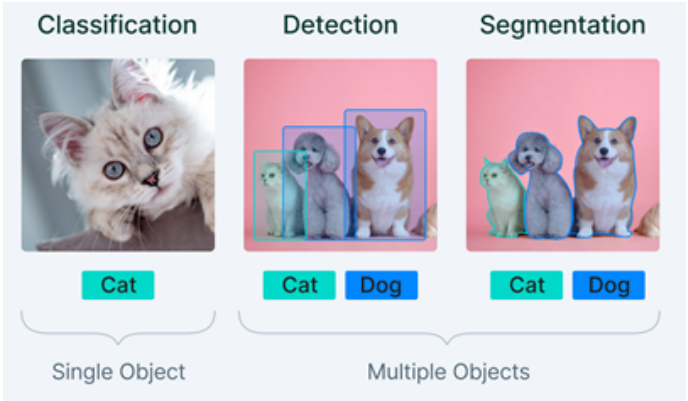
PES UNIVERSITY

Convolutional Neural Network(CNN):

Introduction,Filters, Feature Maps

Introduction To Computer Vision(CV)

- Computer vision is a field of artificial intelligence that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.
- CV is used in various tasks like: **Object Identification, Facial Recognition, Image Segmentation, Image Classification, Neural Style Transfer etc.**
- CV needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognizes images.

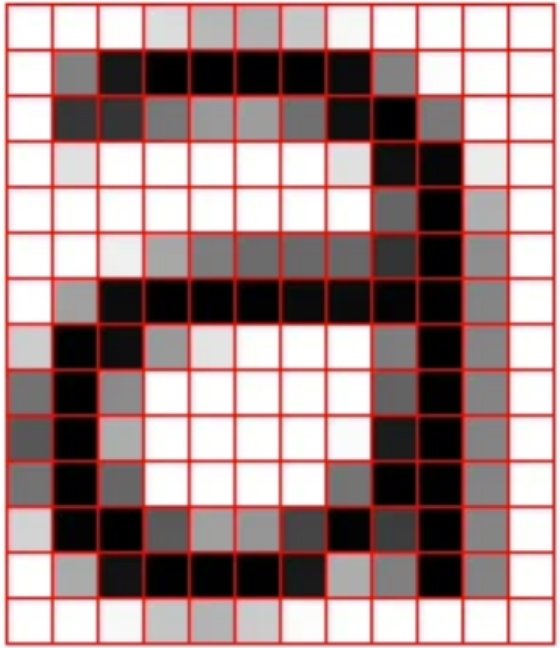


PES UNIVERSITY

A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

How does a computer see an image?

a



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

But how are Image processed?

- Images are processed in the form of an array of pixels, where each pixel has a set of values, representing the presence and intensity of the three primary colors: RGB red, green, and blue.
- All pixels come together to form a digital image.

What's the problem with Fully Connected NN's ?

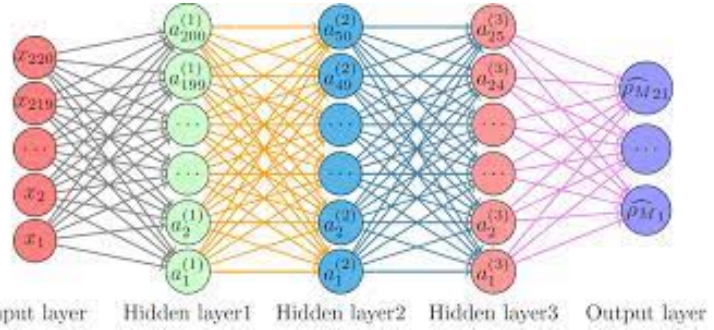
- ***Fully connected Neural Networks:*** All neurons in one layer are connected to all neurons in the next layer. Each neuron in a layer contributes to the activation of every neuron in the subsequent layer, leading to a dense and interconnected structure.

- Imagine in CIFAR Dataset the images are only of size 32 X 32 pixels and have 3 color channels.
  In this case a Single fully connected neuron in a first hidden layer of this neural network would have
  32 x 32 x 32 = 3072 weights (it is still manageable.)

But let's consider a bigger image,
300 x 300 x 3 = 27000 weights
(demands lot of computational power, doesn't leverage the spatial structures and relations of each image)

- A huge neural network like that demands a lot of resources but even then remains prone to overfitting because the large number of parameters enable it to just memorize the dataset.

- They tend to perform less and aren't good for feature extraction.



Why Convolution Neural Networks(ConvNet / CNN) ?

Convolution is a mathematical operation that allows the merging of two sets of information. Convolution is applied to the input data to filter the information and produce a feature map.

Definition 18.1: Convolution

A **convolution** in neural network training is a reduction of the number of weights that the model needs to learn by summarizing the image into fewer pixels.

- A simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. Unlike Fully connected NNs, a neuron in a convolutional layer is not connected to the entire input but just some section of the input data.

- The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved. ConvNet leverage spatial information and are therefore very well suited for classifying images.

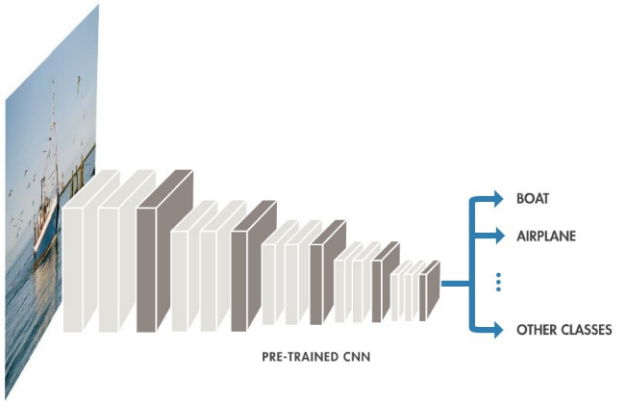
Difference between Fully connected NN vs CNN

Feature	Fully Connected NN (FCN)	Convolutional NN (CNN)
Architecture	Dense, fully interconnected layers	Convolutional layers, pooling layers
Parameter Sharing	No parameter sharing	Shared weights in convolutional filters
Spatial Hierarchies	Do not explicitly capture spatial hierarchies	Capture spatial hierarchies through convolutional and pooling layers
Computational Efficiency	Can be computationally expensive, especially with high-dimensional input	More parameter-efficient, computationally effective, especially for images
Feature Extraction	May struggle with capturing local patterns and spatial relationships	Specialized for feature extraction from local regions, suitable for images

UE22CS645BC2-TDL-Lecture -CNN

Idea of how CNNs Function

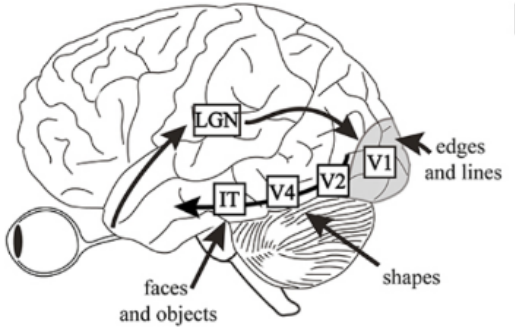
- The CNN is composed of image processing layers that deduce and pass down information from one layer to the next.
- At each layer, information of different abstractions is deduced.
- Generally, in the earlier layers, simpler and more basic ideas will be deduced while later layers will use the gathered information from the previous layers to deduce more complex ideas.
- The following figure lays down an example CNN - in the figure, a boat image is passed from the left end layer to the next until it reaches the right-most layer, where it classifies the class of the image.
- Due to the way the layers learn more complex features the deeper into the network, we can call this a hierarchical learning structure. This is "**Hierarchical Learning Structure**".

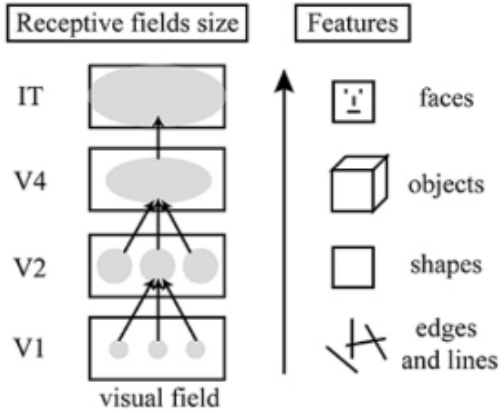


Classifies image as "Boat"

Analogy to Visual Cortex

- Hierarchical layer structure is actually also utilized in the visual cortex ventral pathway.Our vision is based on multiple cortex levels, each one recognizing more and more structured information.
- First, we see single pixels; then from them, we recognize simple geometric forms. And then more sophisticated elements such as objects, faces, human bodies, animals, and so on.
- Individual neurons respond to stimuli only in a restricted region of the visual field known as the **Receptive Field.**
- As we proceed through the visual pathway, the features learned become more complex, just as in the CNN. The receptive visual field size increases as well, as larger receptive field suggests a more holistic and general feature in the image.





Advantages of CNN:

1. Local Receptive Fields:

- In images, there are local patterns and structures that are important for recognizing objects. CNNs use local receptive fields (small regions of the input data) to capture these patterns that ***focuses on specific features*** in the visual data.

2. Hierarchical Feature Learning:

- ***Lower layers capture simple features*** like edges and textures, while ***higher layers learn more complex*** and abstract features, leading to the recognition of objects. This learning makes it highly effective in image-related tasks.

3. Parameter Sharing:

- CNNs use shared weights and biases across different parts of the input data that allows them to learn a set of filters that can be applied across the entire input, making them particularly ***effective for detecting patterns and features in various locations*** of an image. This parameter sharing significantly reduces the number of parameters compared to fully connected networks, making CNN's ***computationally efficient***.

Advantages of CNN:

4. Parameter Efficiency:

- CNNs are more parameter-efficient than fully connected networks, which is crucial when dealing with large images. The use of shared weights and local receptive fields allows CNNs to learn from ***fewer parameters while retaining the ability to capture important features.***

5. Translation Invariance:

- CNNs can recognize patterns regardless of their position in the input space. This is achieved through the use of pooling layers that ***down-sample the spatial dimensions,*** making the network less sensitive to small changes in the position of features.

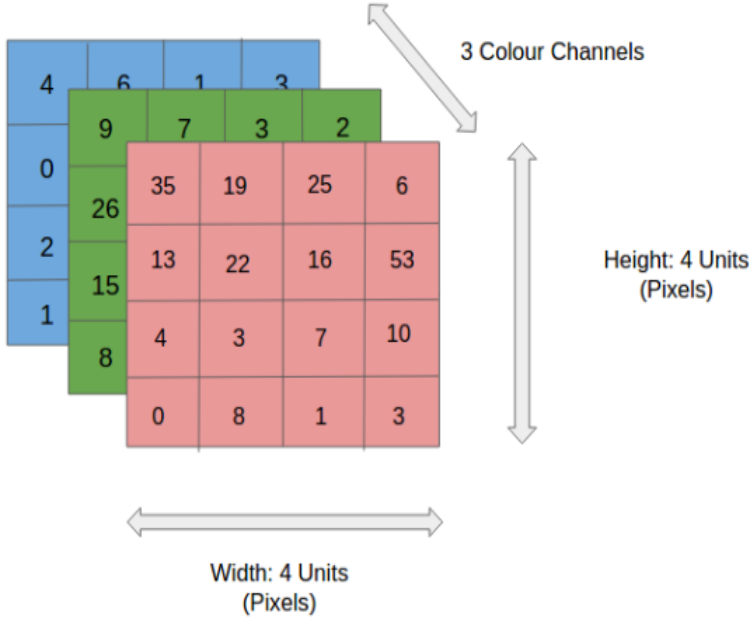
Input Image

- The RGB image in the figure has been separated by its three color planes — Red, Green, and Blue.

- There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

- The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good Prediction.

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



Local Receptive Fields

- To preserve spatial information, each image is represented as a matrix of pixels.

- To encode the local structure a submatrix of adjacent input neurons is connected into one single hidden neuron belonging to the next layer. That single hidden neuron represents one local receptive field. This operation is named convolution.

- Unlike fully-connected layers, where each output unit gathers information from all of the inputs, the activation of a convolution output cell is determined by the inputs in its receptive field.

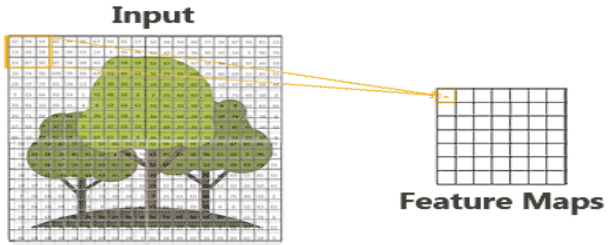
Local Receptive Fields

- This principle works best for hierarchically structured data such as images.
- For eg., suppose that the size of each single submatrix is 5 x 5 and that those submatrices are used with MNIST images of 28 x 28 pixels.
- Then we will be able to generate 23 x 23 local receptive field neurons(feature maps) in the next hidden layer.
- It is possible to slide the submatrices by only 23 positions before touching the borders of the images.
- There can be multiple feature maps that learn independently from each hidden layer.

CNN in a nutshell

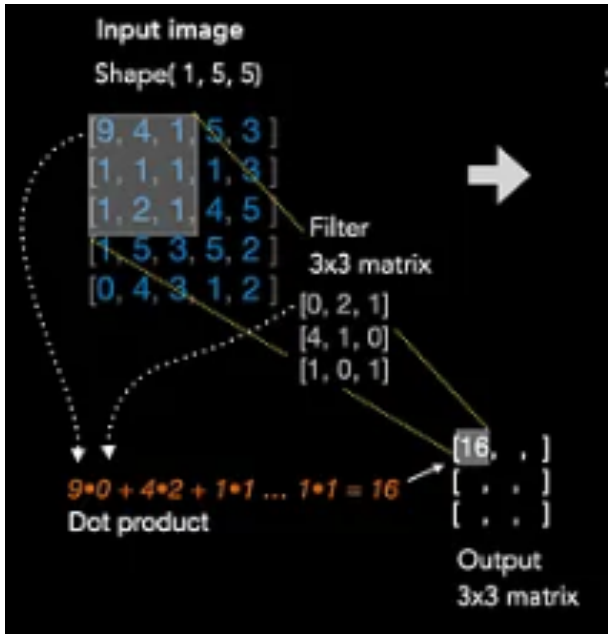
- The first layer is responsible for detecting lines, edges, changes in brightness, and other simple features.
- The information is then passed onto the next layer, which combines simple features to build detectors that can identify simple shapes.
- The process continues in the next layer and the next, becoming more and more abstract with every layer.
- The deeper layer will be able to extract high-order features such as shapes or specific objects.
- The last layers of the network will integrate all of those complex features and produce classification predictions.
- The predicted value will be compared to the correct output, where those that are wrongly classified will cause a large error gap and will cause the learning process to backpropagate to make changes to the parameter in order to give out a more accurate outcome.
- The network goes back and forth, correcting itself until the satisfying output is achieved (where the error is minimised).

CNN in a nutshell



Filters / Kernels

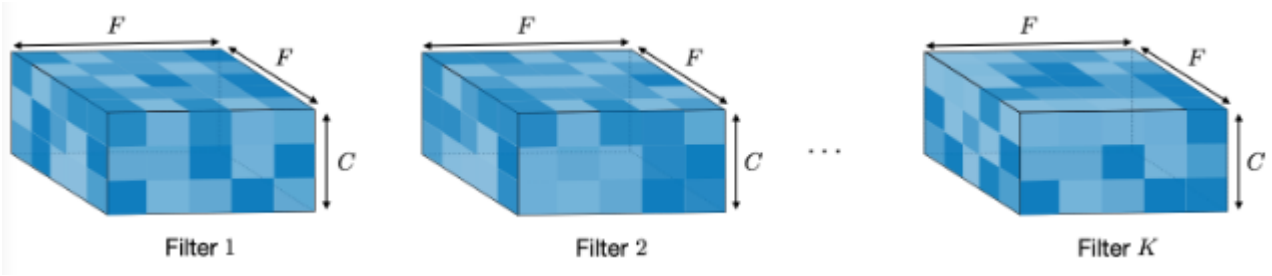
- A kernel in a convolution is an n x n matrix of numbers.
- They are designed to detect and highlight specific patterns or features in the input data.
- The CNN convolution can have multiple filters, highlighting different features, which results in multiple output feature maps (one for each filter).
- It can also gather input from multiple feature maps, for e.g, the output of a previous convolution.
- The combination of feature maps (input or output) is called a volume. In this context, we can also refer to the feature maps as slices.
- For example, in image processing, filters might be designed to detect edges, corners, or textures.
- Each number inside the kernel matrix is multiplied with the value for each pixel it lines up with, and then all the elements of the Kernel are summed to output one single value. Then, the kernel slides over to a new position in the image and the process is repeated.



Filter Hyperparameters

1. DIMENSIONS OF A FILTER:

A filter of size ***F x F*** applied on ***C*** channels is a ***F x F x C*** volume that performs convolutions on a input of size ***I x I x C*** and produces an output feature map/ activation map of size ***O x O x 1***.

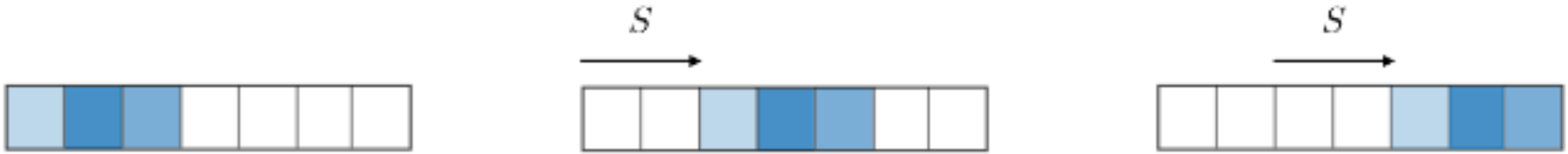


The application of ***K*** filters of size ***F x F*** results in an out feature map of size ***O x O x K.***

Filter Hyperparameters

2. STRIDE:

- For a convolutional or a pooling operation, the stride ***S*** denotes the number of pixels by which the window moves after each operation.
- This shift can be horizontal, vertical, or both, depending on the stride's configuration.



- **Importance of strides** - The choice of stride affects the model in several ways.

Filter Hyperparameters

2. STRIDE:

- **Importance of strides** -

~ Output Size: Larger stride results in a smaller output spatial dimension because filter covers large area of the input image with each step, thus reducing the number of positions it can occupy.

~ Computational Efficiency: Increasing the stride can decrease the computational load.Since the filter moves more pixels per step, it performs fewer operations, which can speed up the training and inference processes.

~ Field of View: Higher stride in the convolutional operation considers a broader area of the input image per step, aiding in capturing global features over finer details.

~ Downsampling: Increasing strides can be used as an alternative to max pooling layers for downsampling the input.

Filter Hyperparameters

3. ZERO-PADDING:

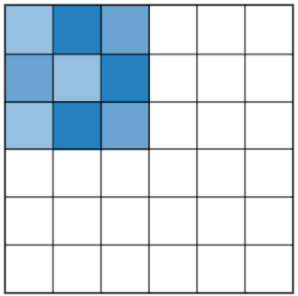
- Zero-padding denotes the process of adding P zeroes to each side of the boundaries of the input. This value can either be manually specified or automatically set.

- It has 3 modes:

(i) VALID MODE:

- ***Where***

  P=0



Purpose:

- No padding
- Drops last convolution if dimensions do not match

Filter Hyperparameters

3. ZERO-PADDING:

(ii) SAME MODE:

- ***Where***

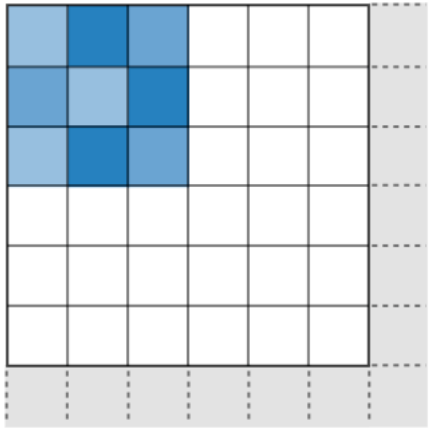
$$P_{\text{start}} = \left\lfloor \frac{S\lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rfloor$$

$$P_{\text{end}} = \left\lceil \frac{S\lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rceil$$

Purpose:

- Padding such that feature map size has size $\left\lceil \frac{I}{S} \right\rceil$
- Output size is mathematically convenient
- Also called ***'half'*** padding

Where
->Filter size is FxF
->Input size is IxI
->Stride is S



Filter Hyperparameters

3. ZERO-PADDING:

(iii) FULL MODE:

- ***Where***

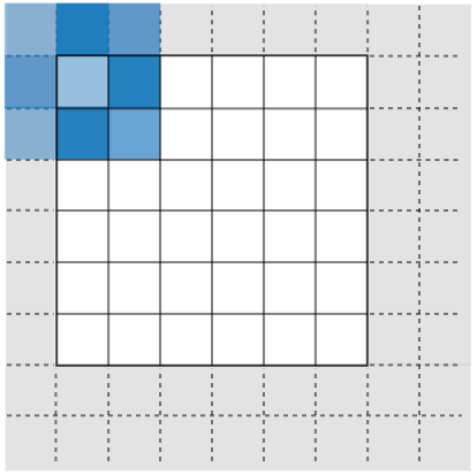
$$P_{\text{start}} \in [\![0, F-1]\!]$$

$$P_{\text{end}} = F - 1$$

Purpose:

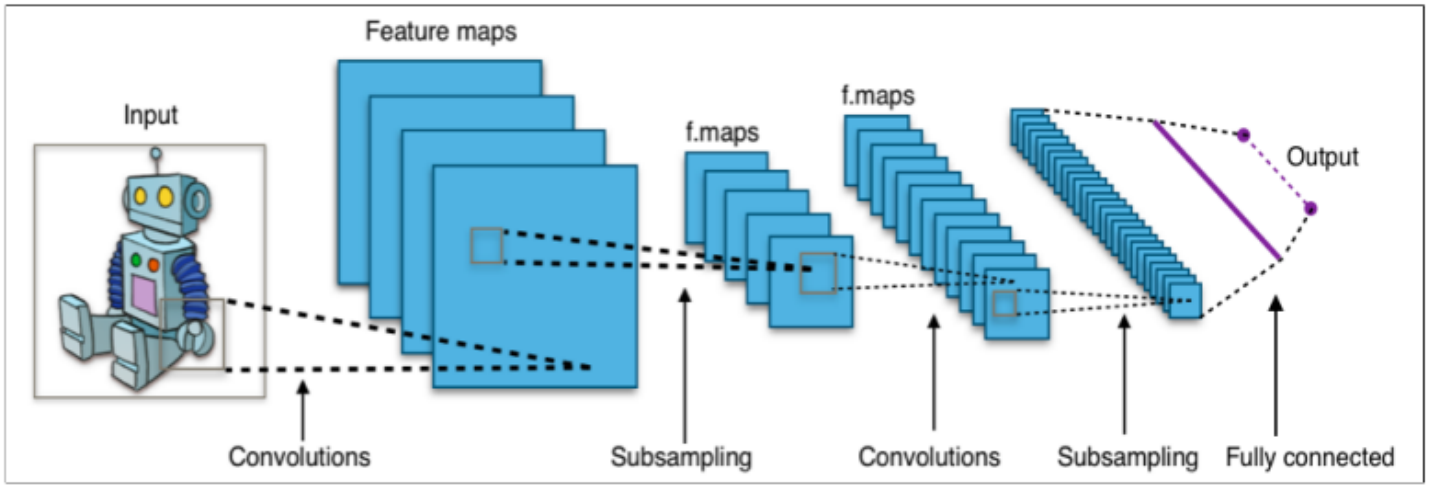
- Maximum padding such that end convolutions are applied on the limits of the input.

- Filter 'sees' the input end-to-end



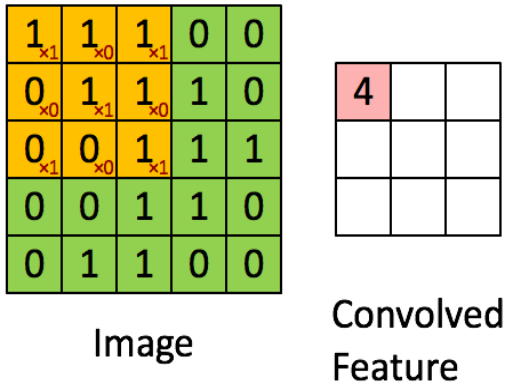
Typical Layers in a ConvNet

- A CNN typically has three layers: ***a convolutional layer, a pooling layer, and fully connected layer.***
- Two different types of layers, **convolutional and pooling**, are typically alternated.
- The depth of each filter increases from left to right in the network.
- The last stage is typically made of one or more **fully connected layers**



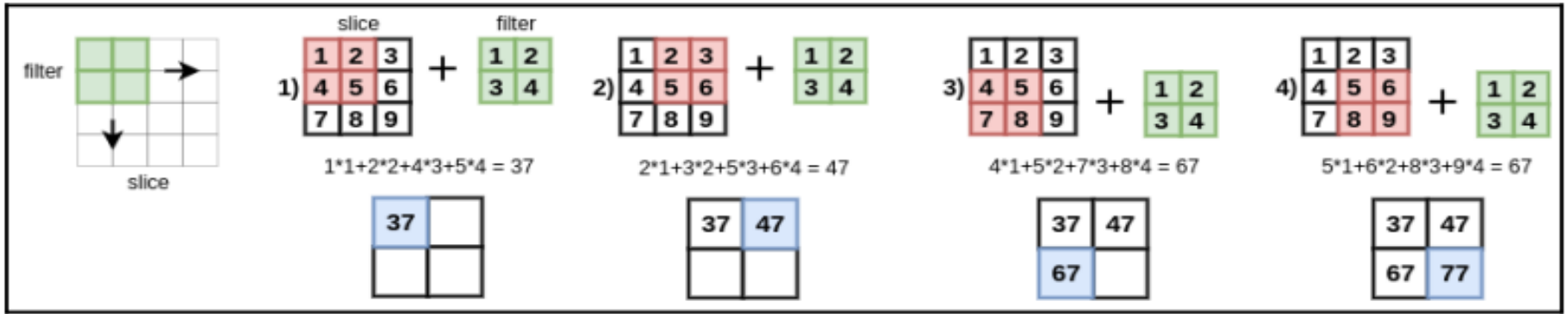
Convolution Operation

- Consider an image of dimensions 5x5x1 [5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)]
- In the illustration, the green section is the **5x5x1 input image, I**.
- The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow.
- The filter **K is a 3x3x1 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 image** over which the kernel is hovering.



Convolution Operation

- Depending on the number of input dimensions, we have 1D, 2D, or 3D convolutions.
- A time series input is a 1D vector, an image input is a 2D matrix, and a 3D point cloud is a 3D tensor. T
- The convolution works as follows:
  1. We slide the filter along all of the dimensions of the input tensor.
  2. At every input position, we multiply each filter weight by its corresponding input tensor cell at the given location.
  3. The input cells, which contribute to a single output cell, are called receptive fields.
  4. We sum all of these values to produce the value of a single output cell.
- 2D convolution with a 2×2 filter applied over a single 3×3 slice



Convolution Operation

$$s_t = \sum_{a=0}^{\infty} x_{t-a} w_{-a} = (x*w)_t$$

input — convolution — filter

- Suppose we are tracking the position of an aeroplane using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy
- To obtain a less noisy estimate we would like to average several measurements
- More recent measurements are more important so we would like to take a weighted average

Convolution Operation

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- The weight array ($\mathbf{w}$) is known as the filter
- We just slide the filter over the input and compute the value of s_t based on a window around x_t

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S					1.80		

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Convolution Operation

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- The weight array ($\mathbf{w}$) is known as the filter
- We just slide the filter over the input and compute the value of s_t based on a window around x_t

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S							1.80	1.96				

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Convolution Operation

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- The weight array ($\mathbf{w}$) is known as the filter
- We just slide the filter over the input and compute the value of s_t based on a window around x_t

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S							1.80	1.96	2.11			

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Convolution Operation

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- The weight array (**w**) is known as the filter
- We just slide the filter over the input and compute the value of s_t based on a window around x_t

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S	1.80	1.96	2.11	2.16		

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Convolution Operation

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- The weight array ($\mathbf{w}$) is known as the filter
- We just slide the filter over the input and compute the value of s_t based on a window around x_t

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S	1.80	1.96	2.11	2.16	2.28	

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Convolution Operation

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- The weight array ($\mathbf{w}$) is known as the filter
- We just slide the filter over the input and compute the value of s_t based on a window around x_t
- Here the input (and the kernel) is one dimensional
- Can we use a convolutional operation on a 2D input also?

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S	1.80	1.96	2.11	2.16	2.28	2.42

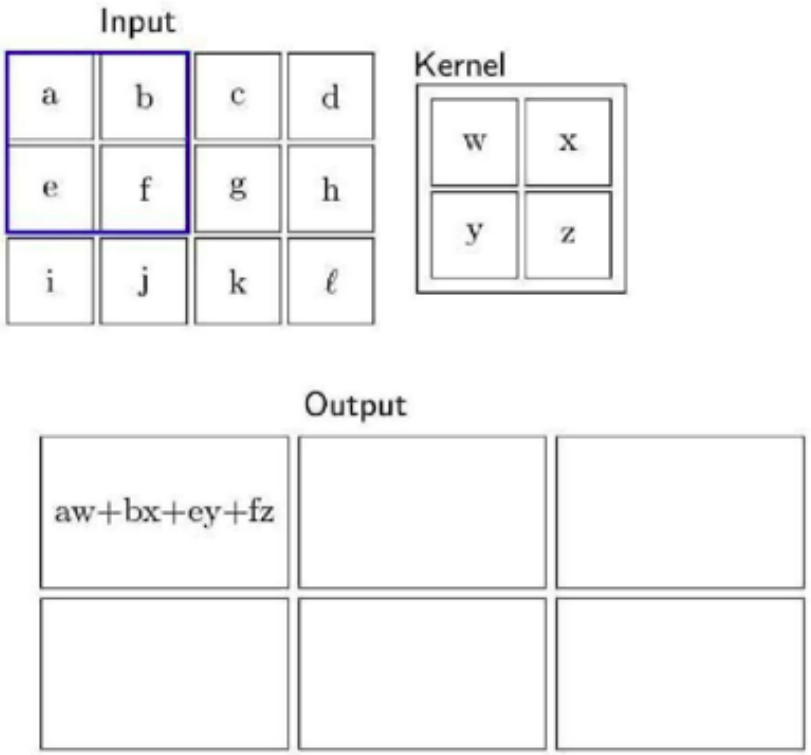
$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Convolution Operation

$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a,j+b} K_{a,b}$$

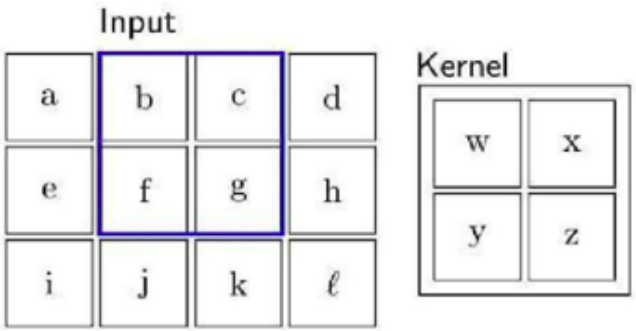
- We can think of images as 2D inputs
- We would now like to use a 2D filter $(m \times n)$
- First let us see what the 2D formula looks like
- This formula looks at all the preceding neighbours $(i-a, j-b)$
- In practice, we use the following formula which looks at the succeeding neighbours

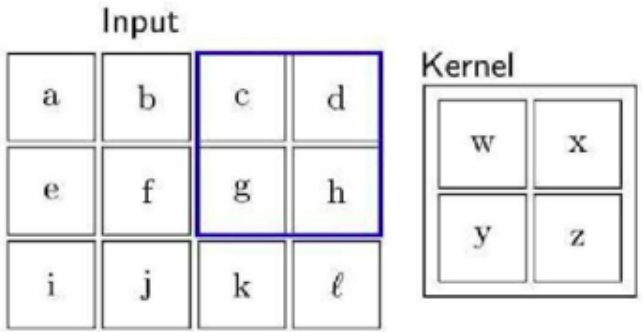
Convolution Operation



- **Let us apply this idea to a toy example and see the results.**

Convolution Operation





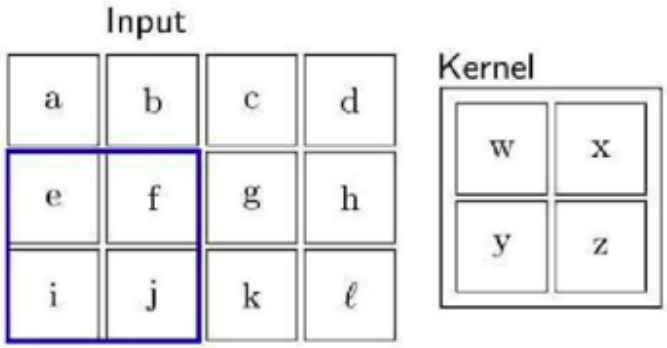
Output

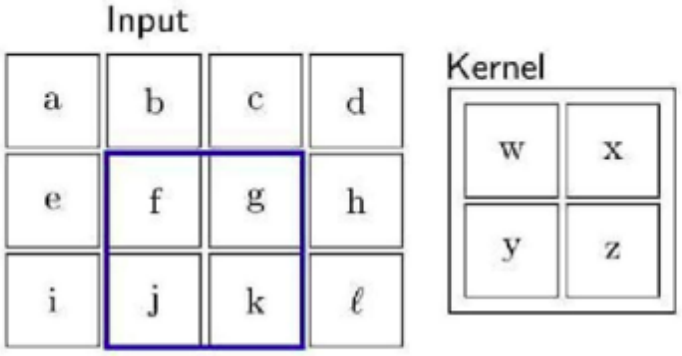
aw+bx+ey+fz	bw+cx+fy+gz	

Output

aw+bx+ey+fz	bw+cx+fy+gz	cw+dx+gy+hz

Convolution Operation





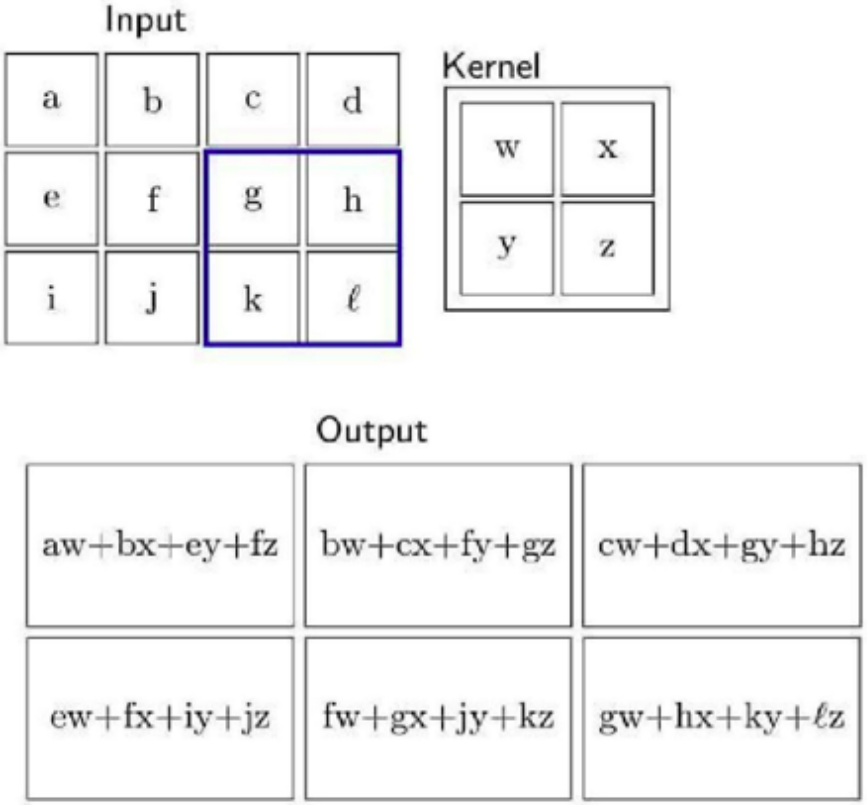
Output

aw+bx+ey+fz	bw+cx+fy+gz	cw+dx+gy+hz
ew+fx+iy+jz		

Output

aw+bx+ey+fz	bw+cx+fy+gz	cw+dx+gy+hz
ew+fx+iy+jz	fw+gx+jy+kz	

Convolution Operation



Finally !!

PES UNIVERSITY

TOPICS IN DEEP LEARNING

Convolution in 1D, 2D, 3D

Effects of Convolutions on Images

Let us see some examples of 2D convolutions applied to images

INPUT ***KERNEL***

OUTPUT

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

blurs the image

INPUT ***KERNEL***

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

sharpens the image

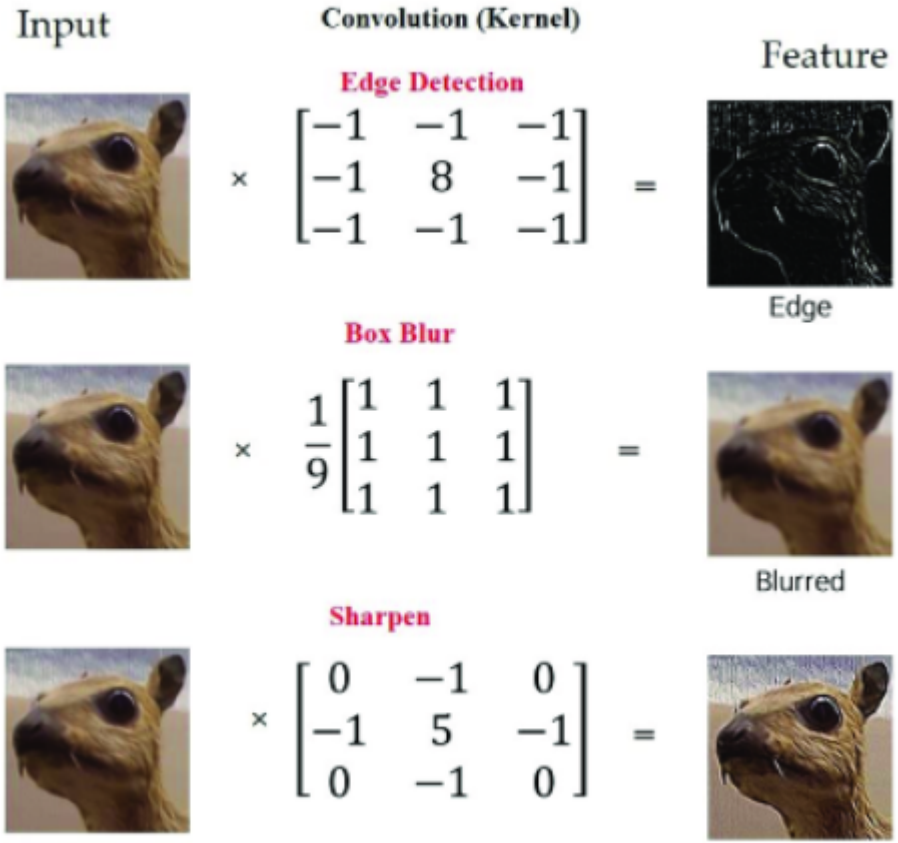
INPUT ***KERNEL***

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

detects the edges

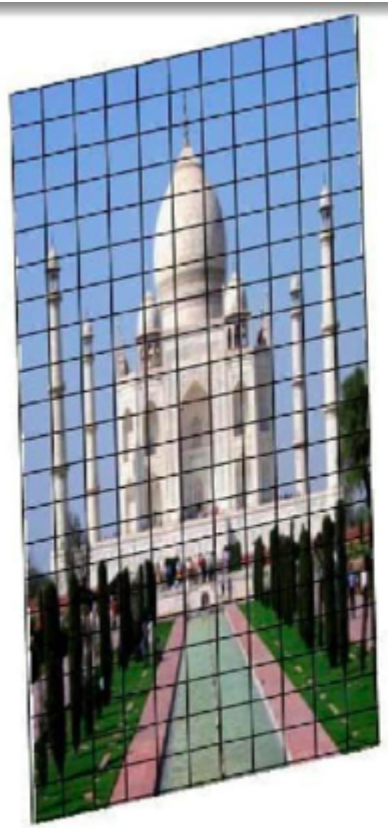
Note how all the edges separating different colours and brightness levels have been identified

Effects of Convolutions on Images - Eg: 2

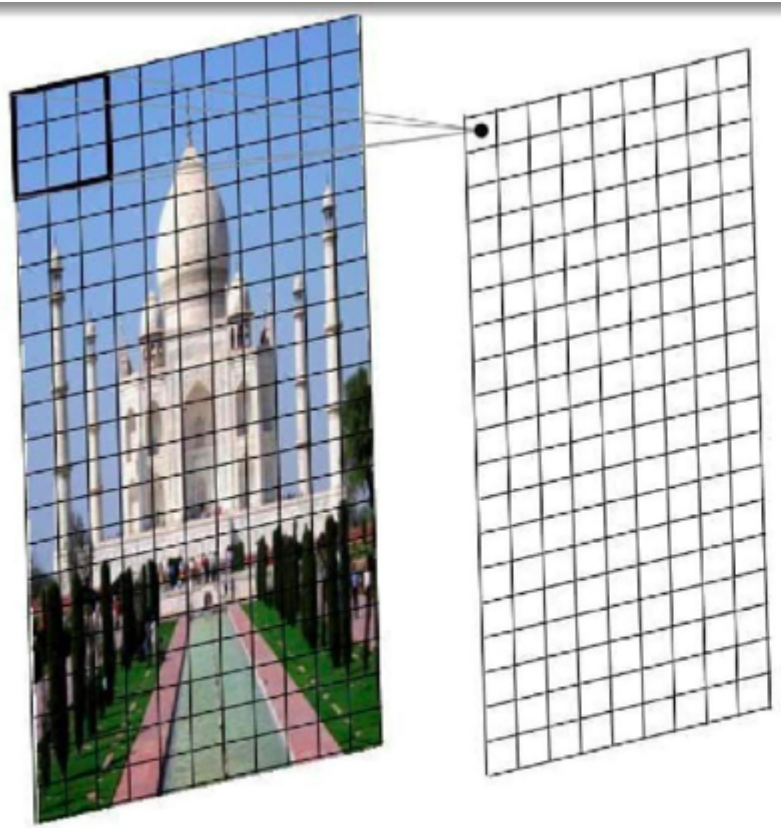


PES UNIVERSITY

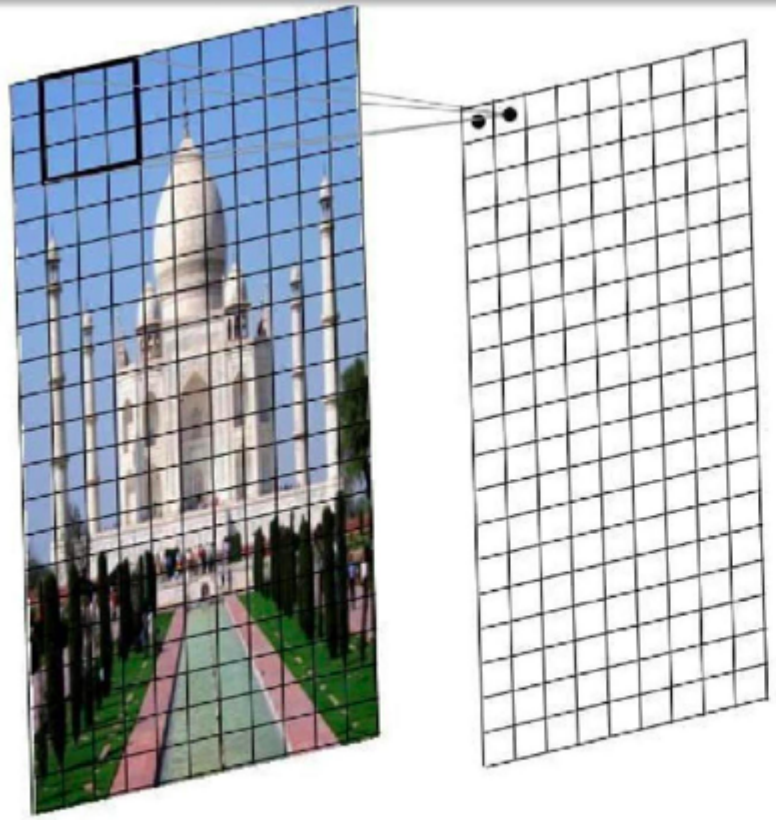
Let’s see the working example of 2D convoultion



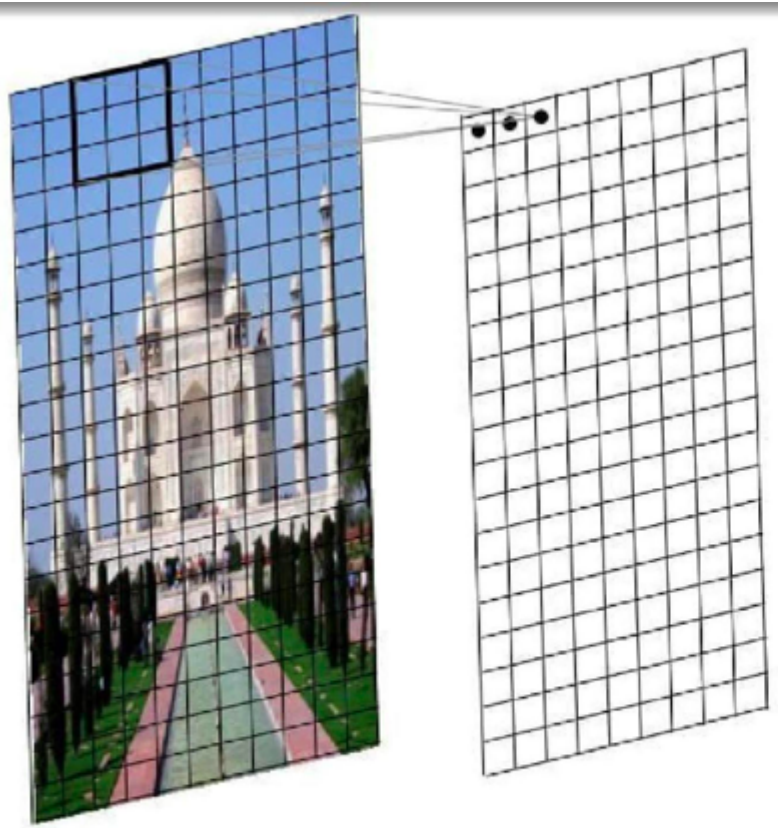
- We just slide the kernel over the input image



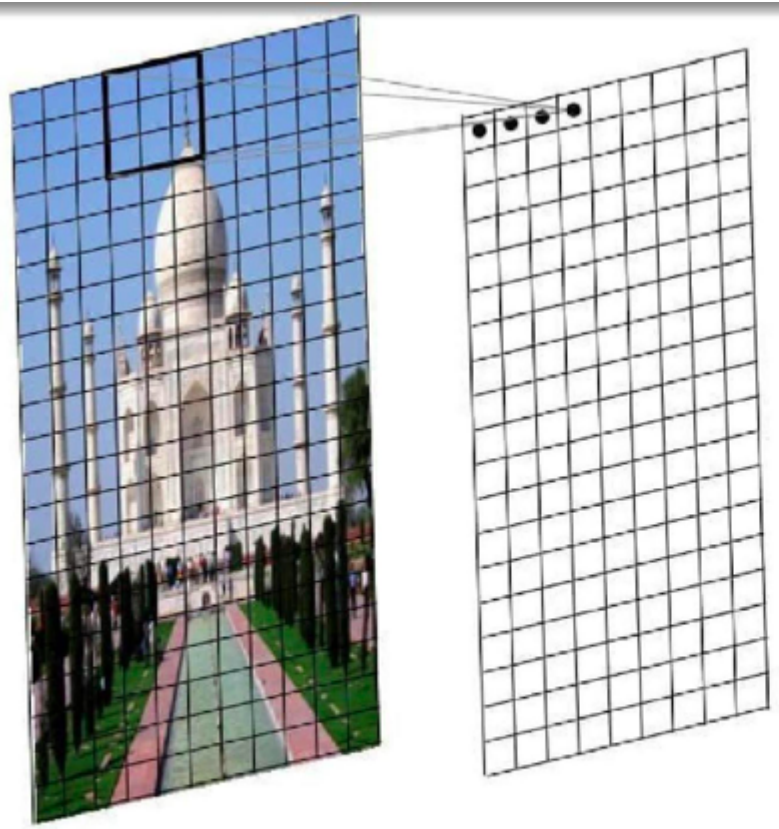
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output



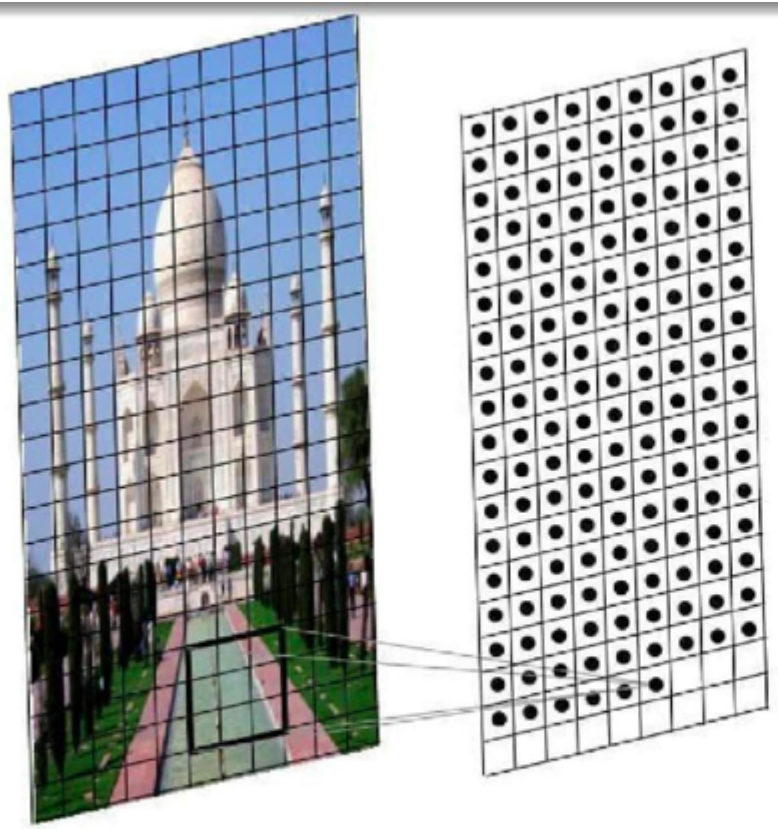
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output



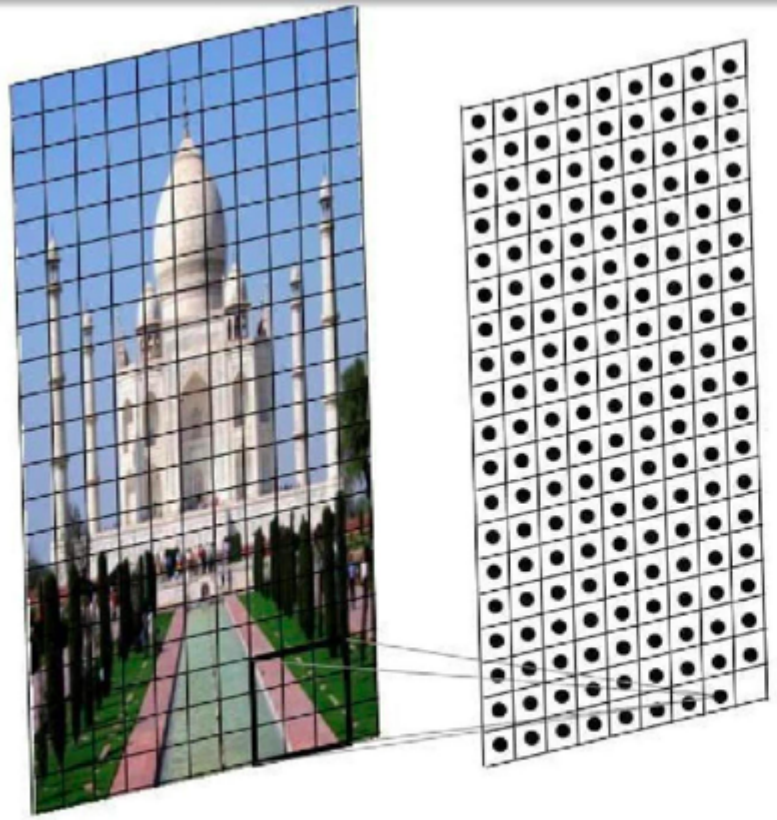
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output



- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

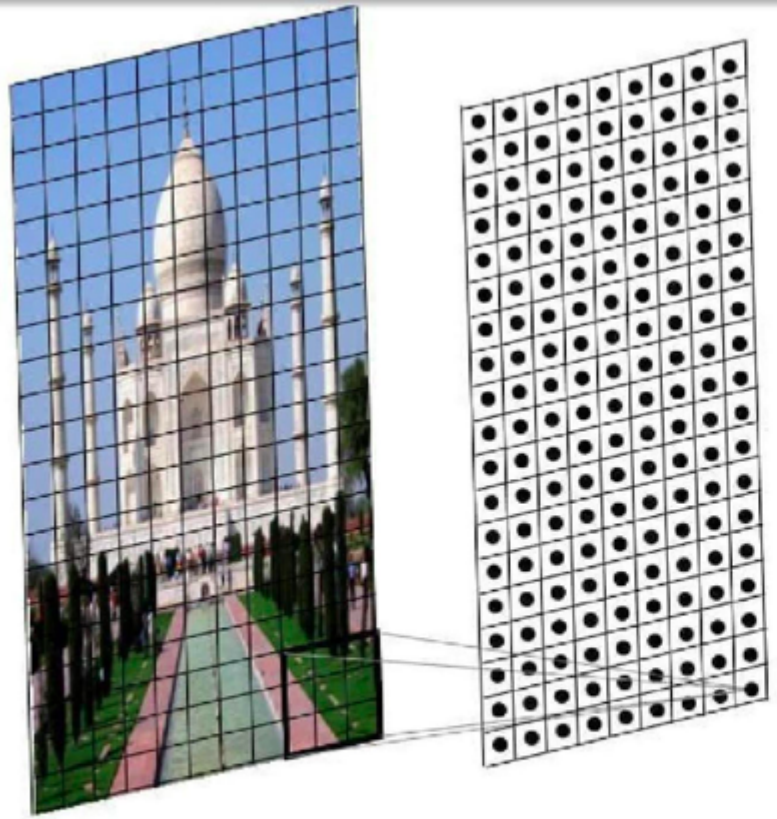


- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output



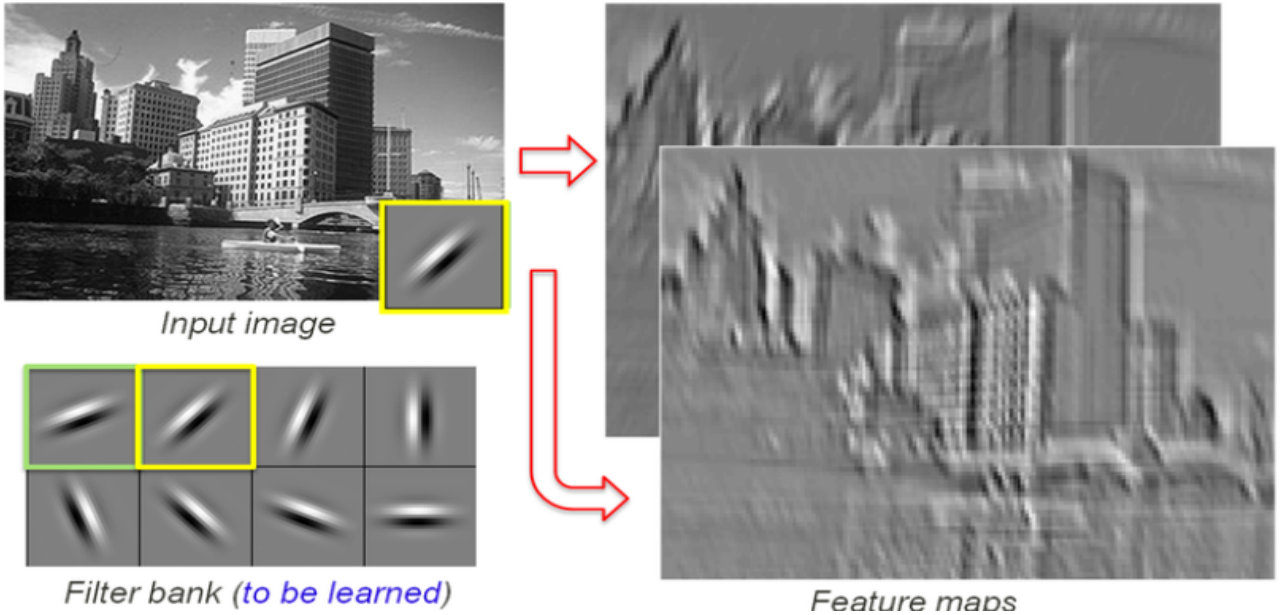
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

Effects of Convolutions on Images

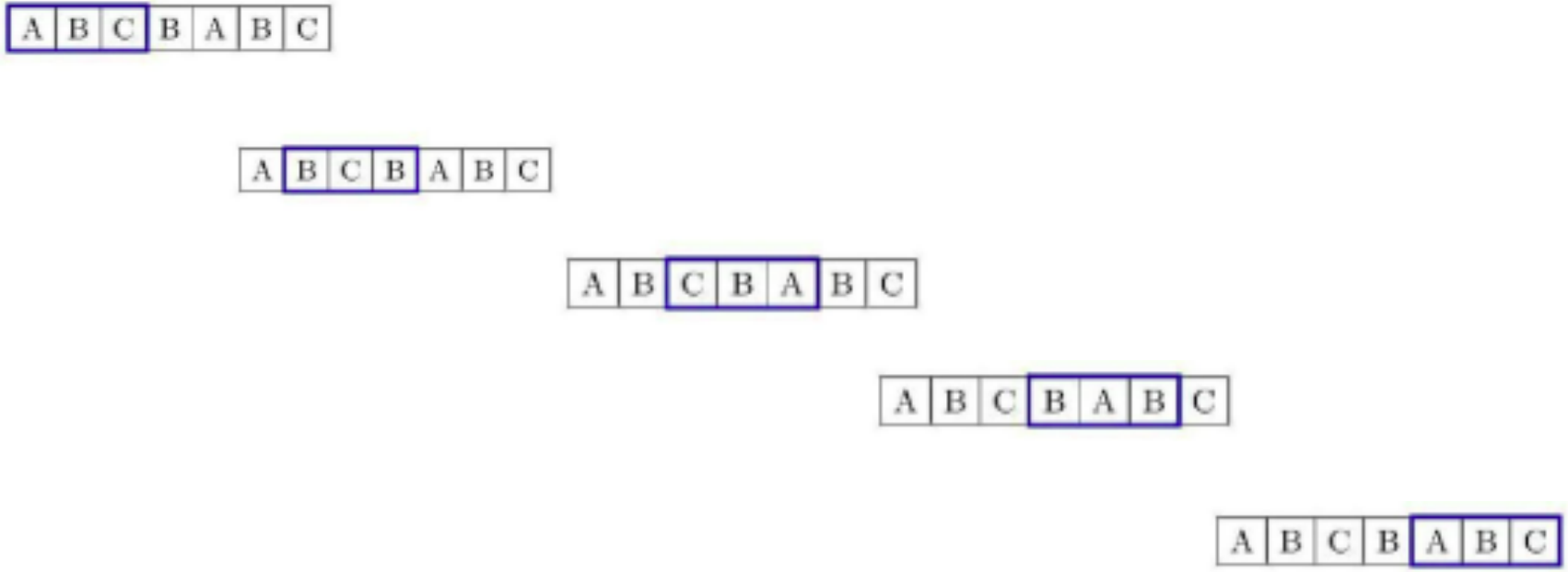


- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- The resulting output is called a feature map.

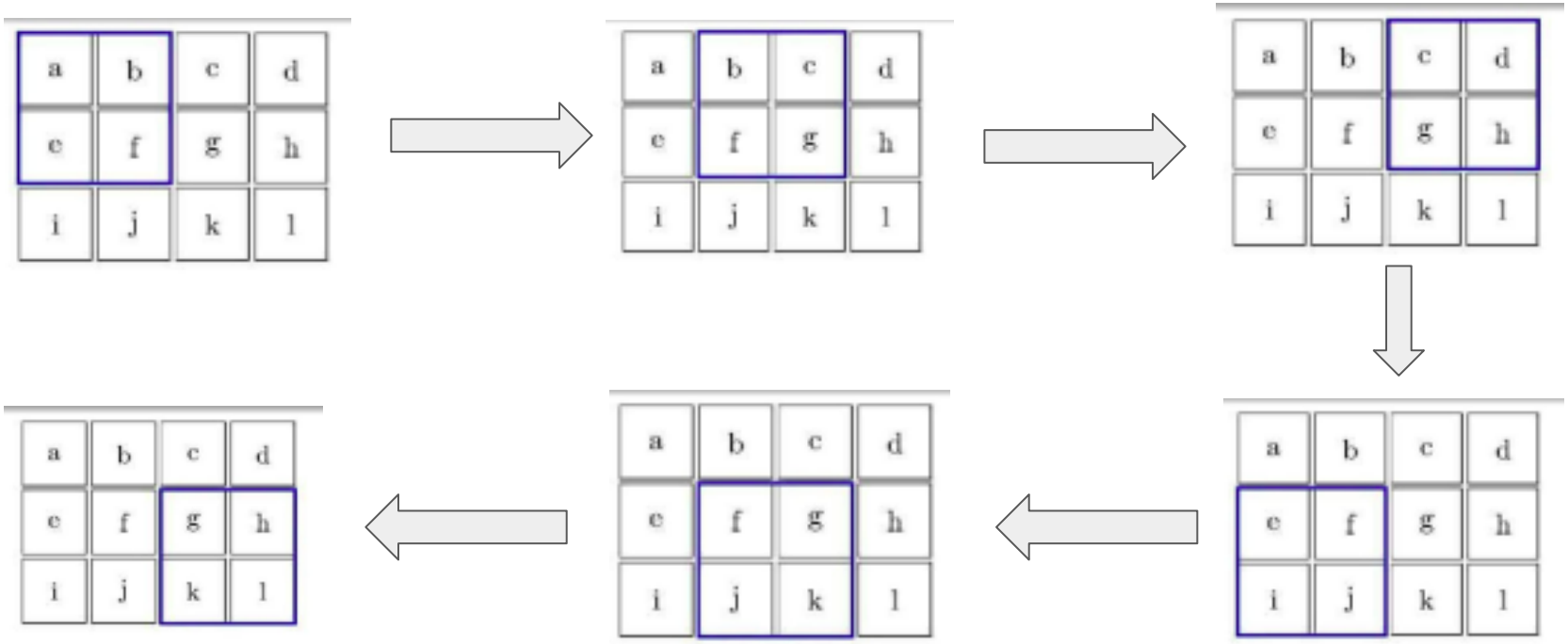
We can use multifilters to get multiple feature maps.



In the 1D case, we slide a one dimensional filter over a one dimensional input

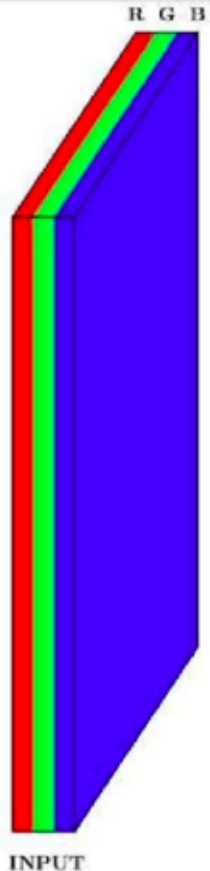


In the 2D case, we slide a two dimensional filter over a two dimensional input

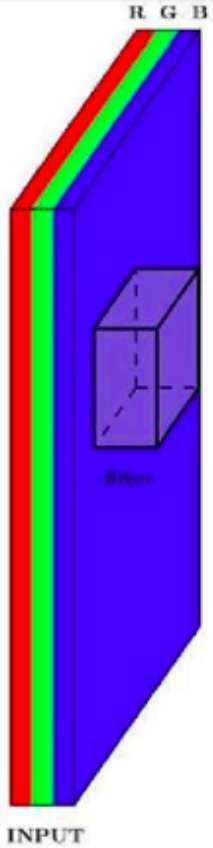


What would happen in case of 3D?

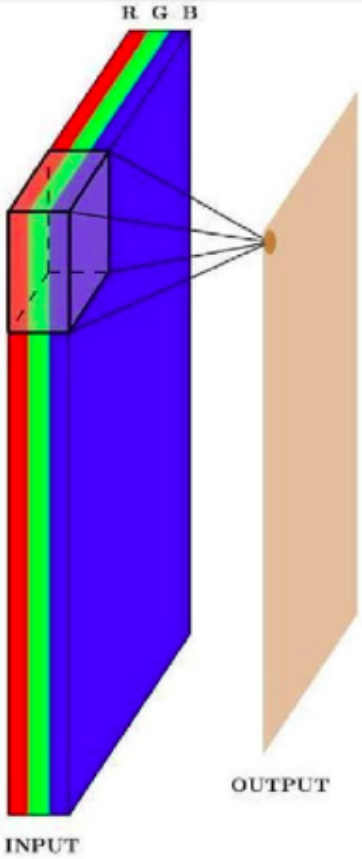
Convolutions in 3D



- What would a 3D filter look like?

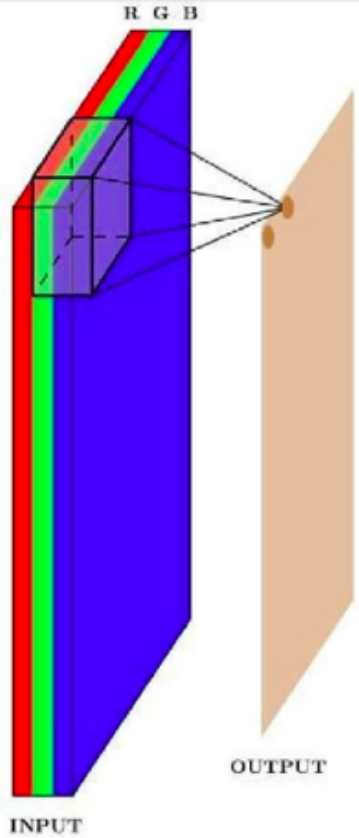


- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume



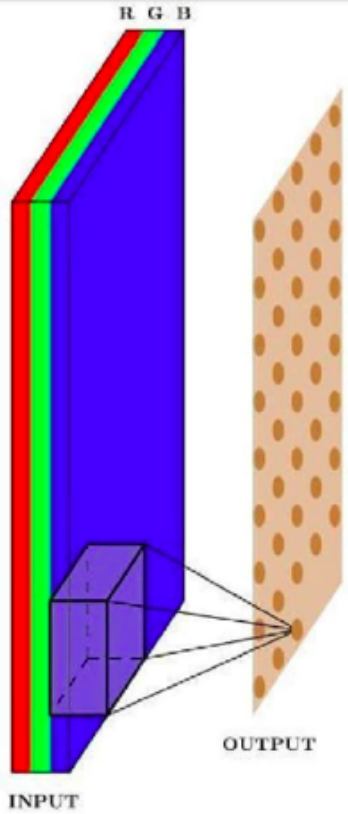
- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Convolutions in 3D



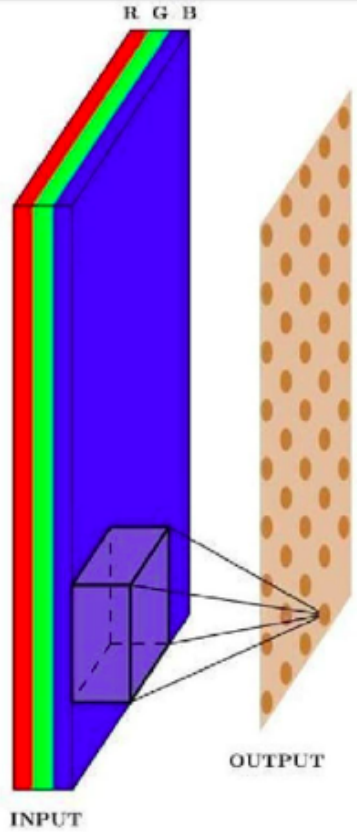
- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Convolutions in 3D



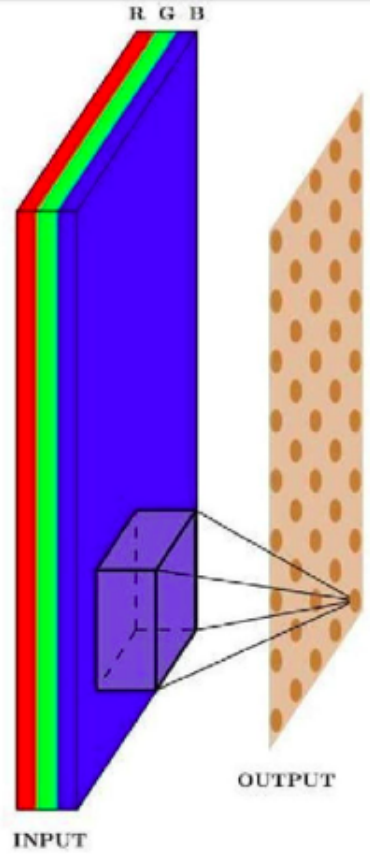
- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Convolutions in 3D

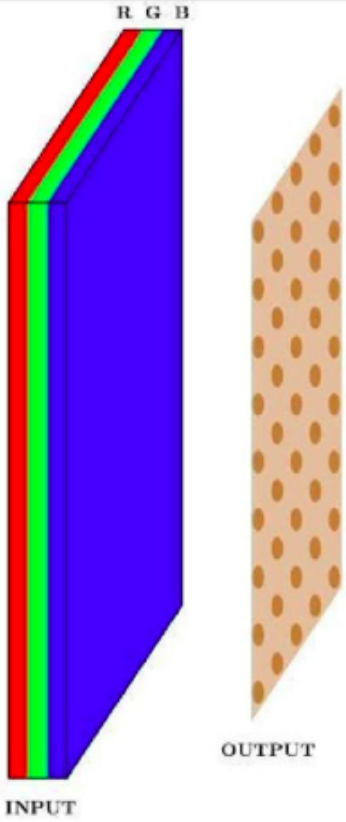


- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

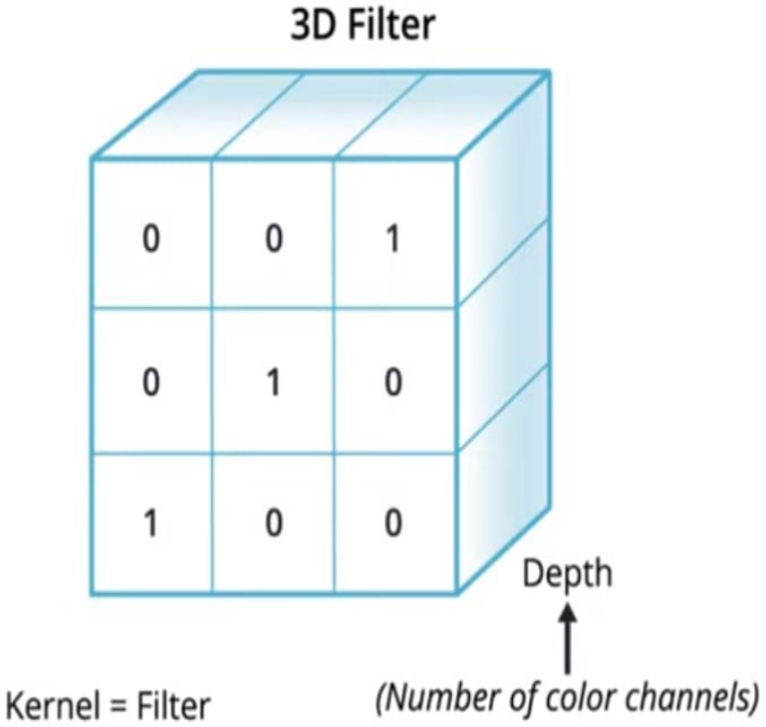
Convolutions in 3D

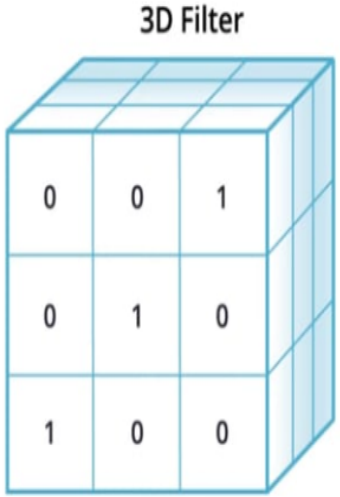


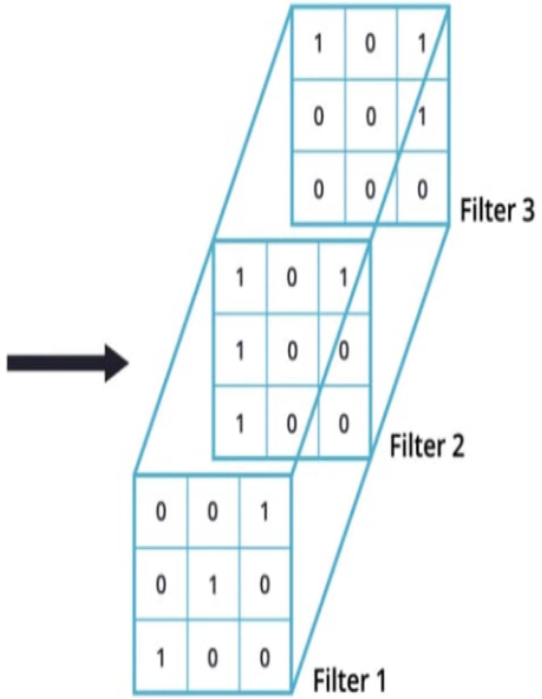
- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation
- Note that in this lecture we will assume that the filter always extends to the depth of the image



- What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation
- Note that in this lecture we will assume that the filter always extends to the depth of the image
- In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth)
- As a result the output will be 2D (only width and height, no depth)
- Once again we can apply multiple filters to get multiple feature maps


3D Filter
0 0 1
0 1 0
1 0 0
Depth
Kernel = Filter
(Number of color channels)


3D Filter
0 0 1
0 1 0
1 0 0


1 0 1
0 0 1
0 0 0
Filter 3
1 0 1
1 0 0
1 0 0
Filter 2
0 0 1
0 1 0
1 0 0
Filter 1

Convolutions in 3D

Suppose we have an RGB image and we want to convolve it with the following 3D filter. As we can see, the depth of our filter consists of three 2D filters. Let's assume that our RGB image is 5 by 5 pixels.

Input Image
5 x 5

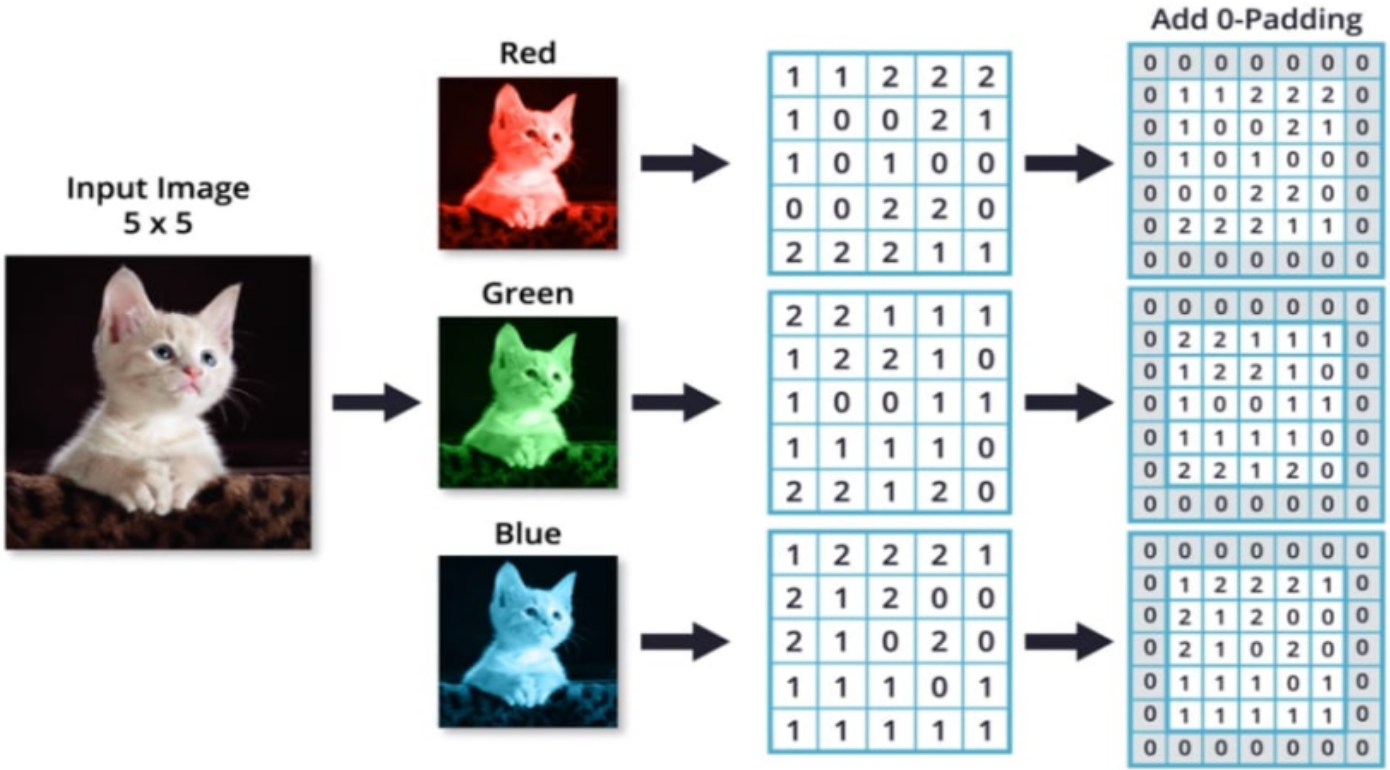
Each color channel corresponds to a two-dimensional array of pixel values

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

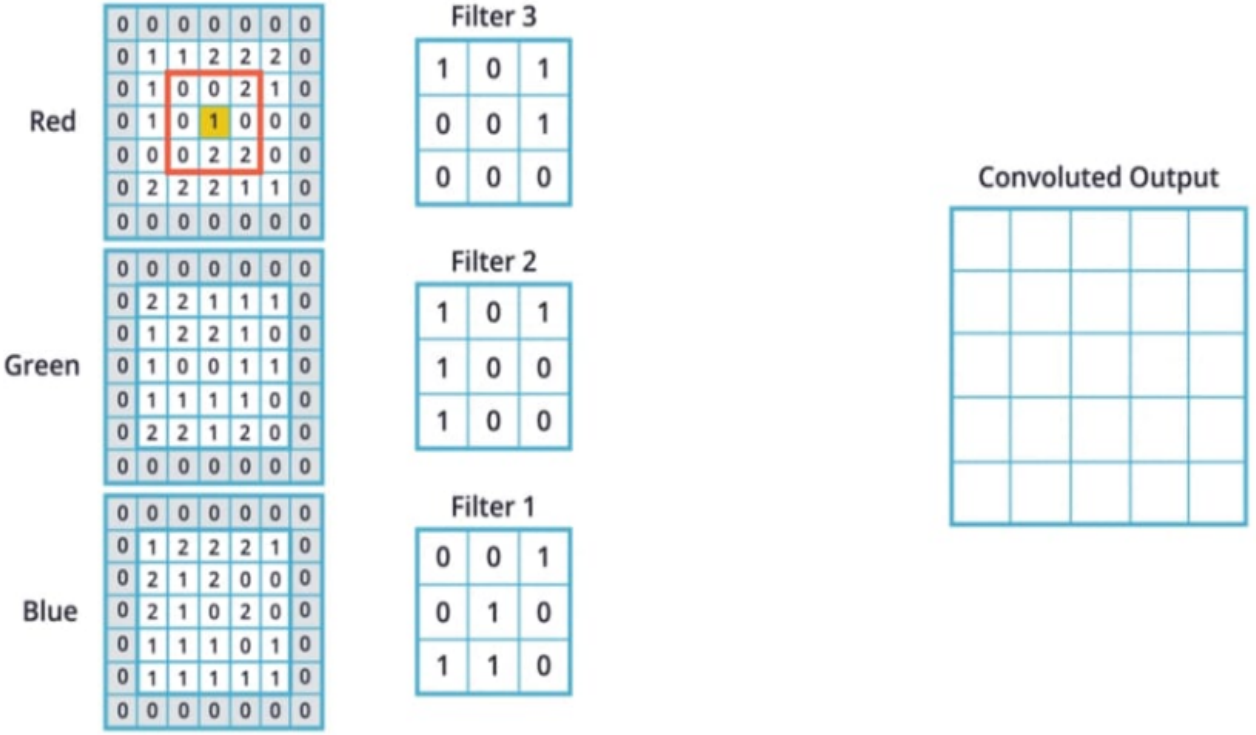
2	2	1	1	1
1	2	2	1	0
1	0	0	1	1
1	1	1	1	0
2	2	1	2	0

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

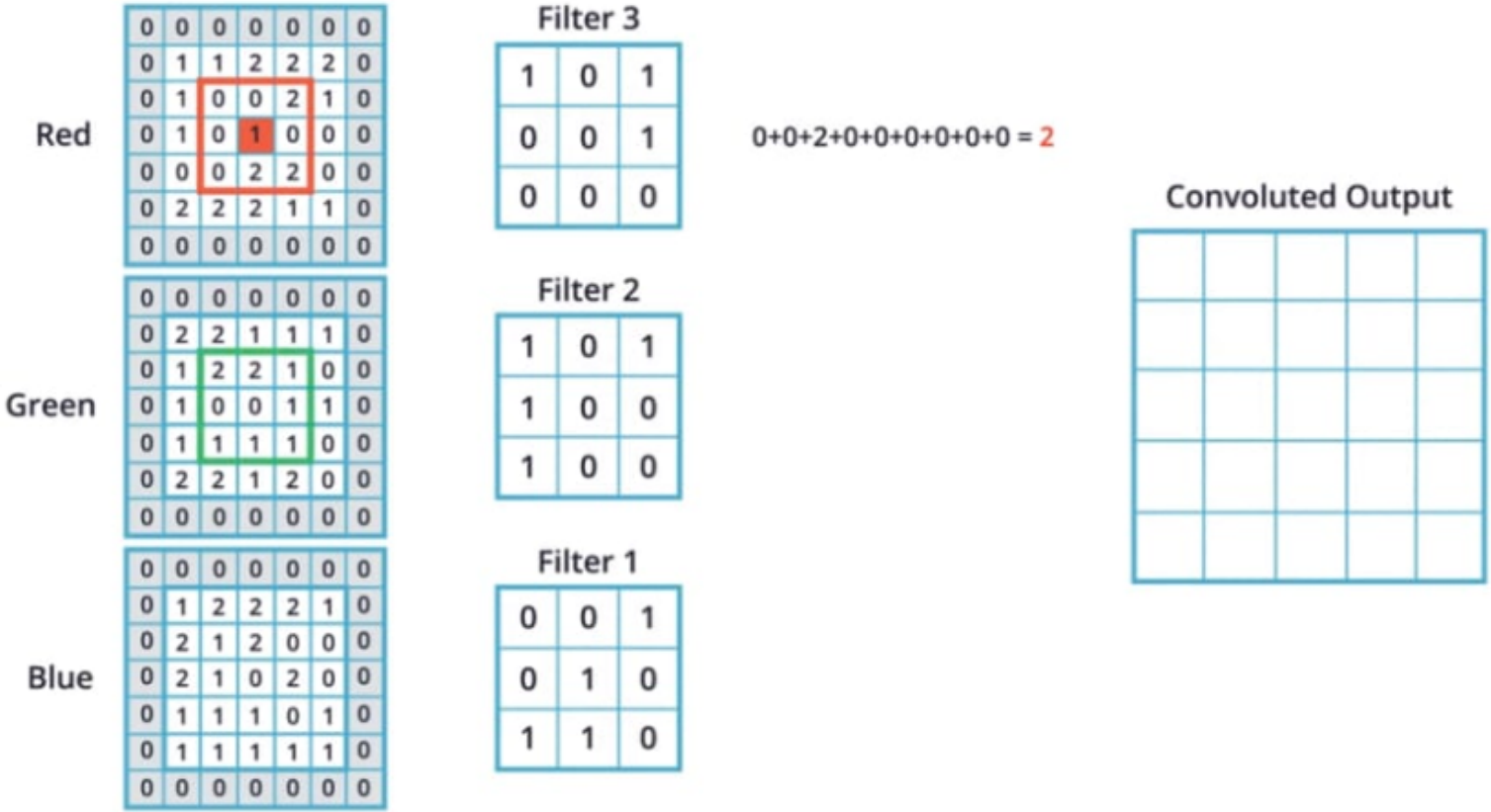
We will add zero padding to each of these arrays in order to avoid losing information when performing the convolution.

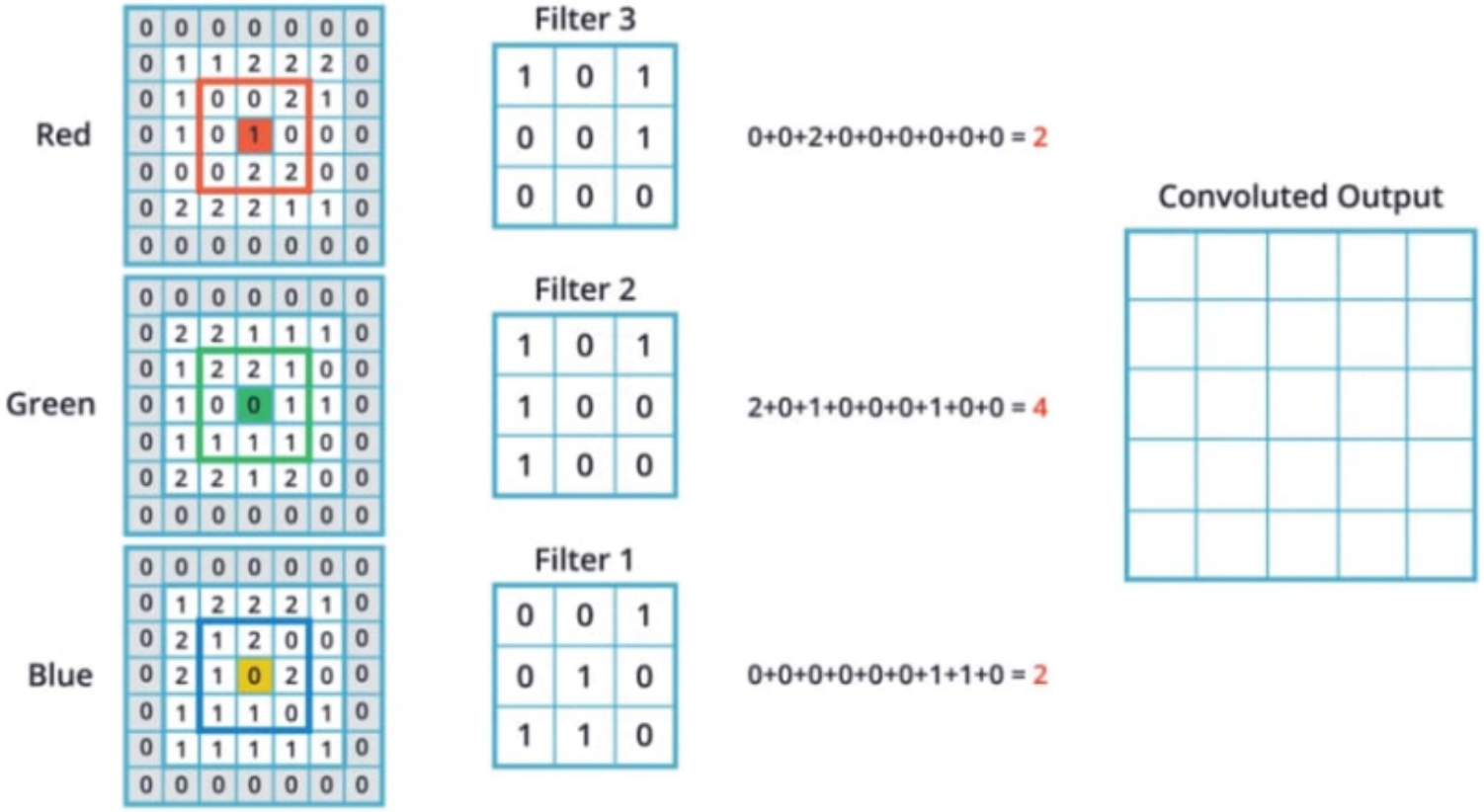


The convolutions will be carried out in exactly the same way as for grayscale images. The only difference is that now we have to perform three convolutions instead of 1

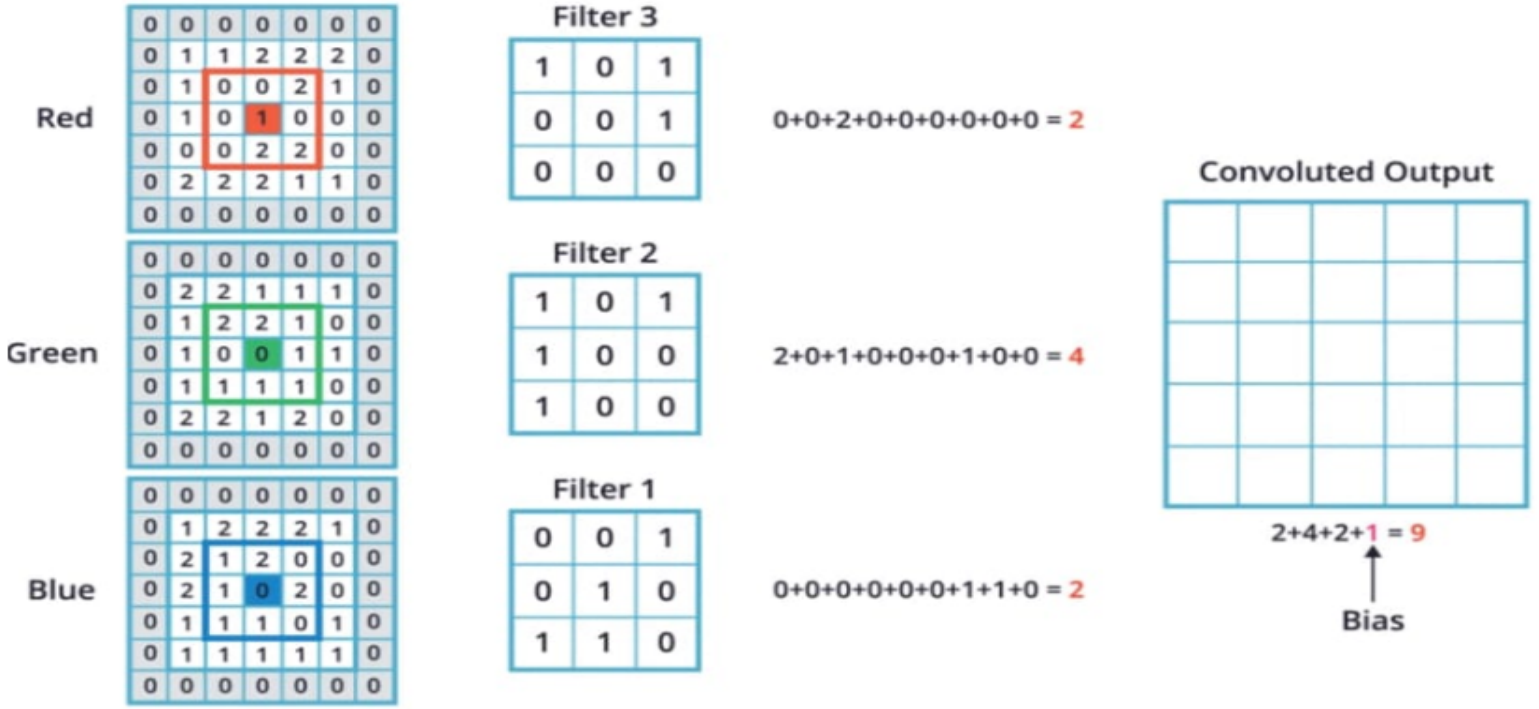


Take each corresponding pixel and filter value, multiply them together, and sum the whole thing up.

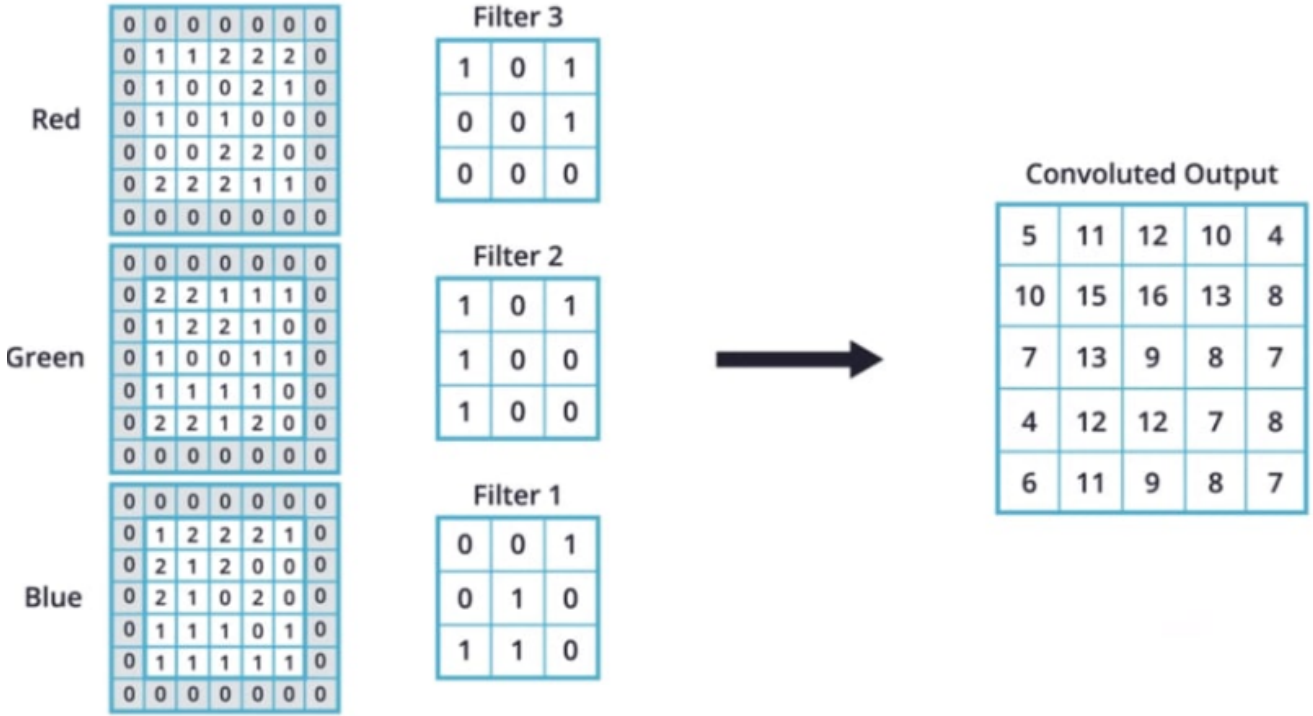



Red
Green
Blue
Filter 3
1 0 1
0 0 1
0 0 0
0+0+2+0+0+0+0+0+0 = 2
Filter 2
1 0 1
1 0 0
1 0 0
2+0+1+0+0+0+1+0+0 = 4
Filter 1
0 0 1
0 1 0
1 1 0
0+0+0+0+0+0+1+1+0 = 2
Convoluted Output

Add a bias value, which usually has a value of 1



To get the full convoluted output, we just perform the same operations for all the other pixels in each of our color channels



Acknowledgements & References

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

https://towardsdatascience.com/cnn-part-i-9ec412a14cb1

https://msail.github.io/previous_material/cnn/

UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Divija L,
Teaching Assistant