edunet
foundation

Unit -3
Computer Vision Basics

edunet foundation

Objective

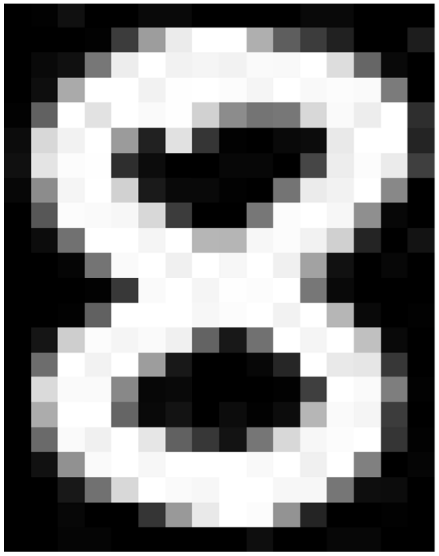
- Introduction to Computer Vision
- Introduction to Image
- Introduction to OpenCV
- Edge Detection
- Introduction to CNN
- Filters
- Padding

What is Computer Vision(CV)?

- Computer vision (CV) is an artificial intelligence (AI) subcategory that focuses on developing and deploying digital systems that process, analyze, and interpret visual input.
- The objective of computer vision is to allow computers to recognize an item or person in a digital image and take appropriate action.
- Convolutional neural networks (CNNs) are used in computer vision to analyze visual input at the pixel level.

What is an Image?

Data in the form of matrix(Rows and Columns) consisting of Pixels



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Image : Image with Pixels

https://mozanunal.com/images/pixel.png

Types of Images

(a)

Color Image

(b)

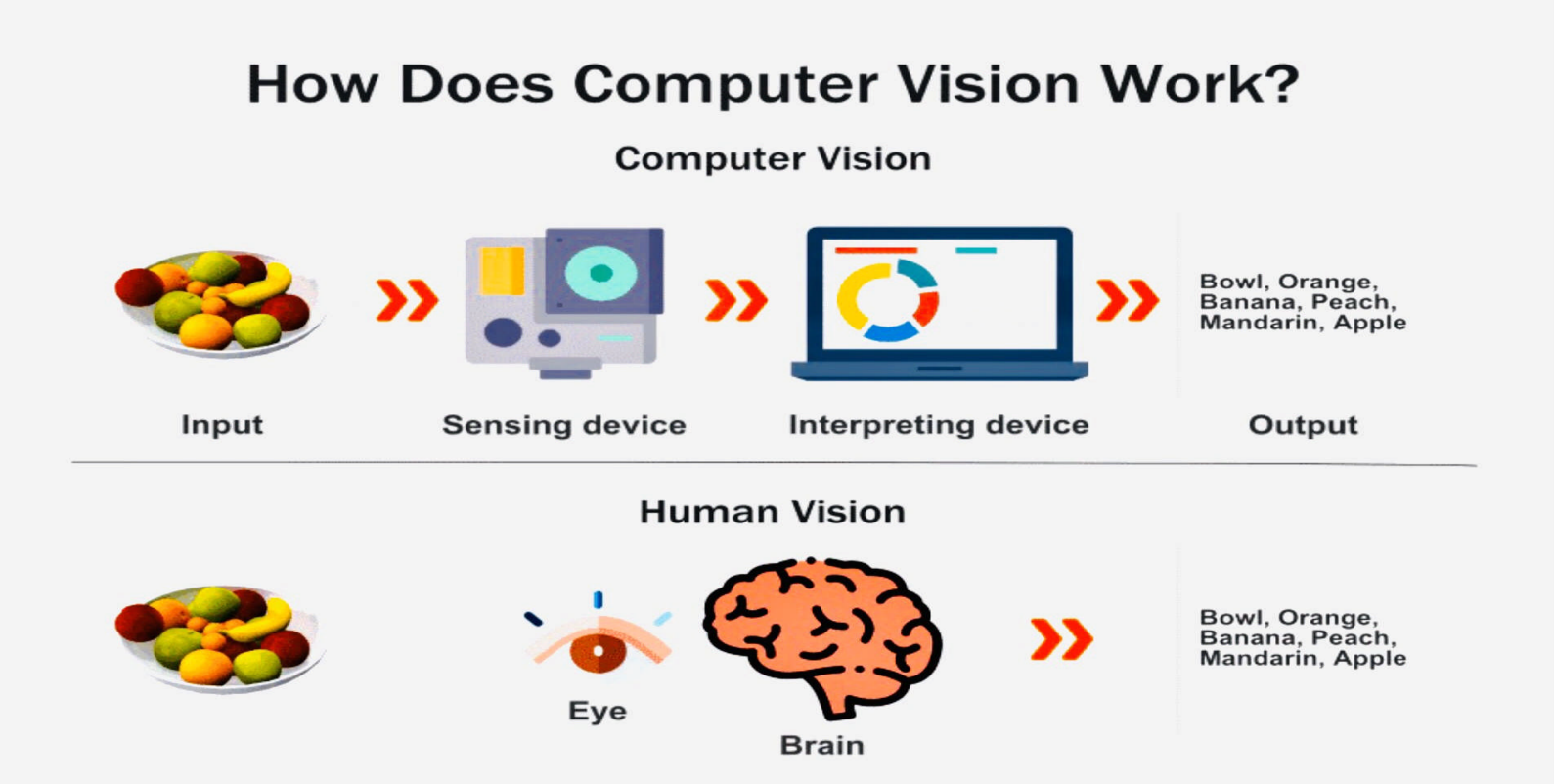
Grayscale Image

(c)

Binary Image

https://www.researchgate.net/profile/Sanskruti-Patel-2/publication/344249310/figure/fig2/AS:935972338425861@1600164603821/Figure2-a-RGB-image-b-Gray-Scale-image-c-Binary-image.jpg

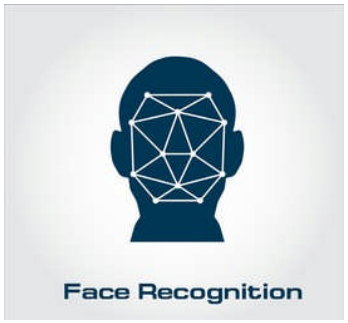
Working of Computer Vision



https://www.woodnots.com/2020/07/computer-vision-basic.html

edunet foundation

Popular Python Libraries for CV



edunet foundation

Basic Operations on Images using CV2

- Modify pixel values by gaining access to them.
- Image attributes may be accessed.
- Choosing an Image Region (ROI)
- Image Splitting and Merging

Functions for accessing image using OpenCV

- Importing OpenCV library: import cv2
- Loading an image : img=cv2.imread("cat.jpg")
- **Splitting and Merging Image Channels**: When necessary, an image's B, G, and R channels can be split into their component planes. The different channels may then be merged to generate a BGR picture once more.

  b,g,r=cv2.split(img)

  img=cv2.merge((b,g,r))

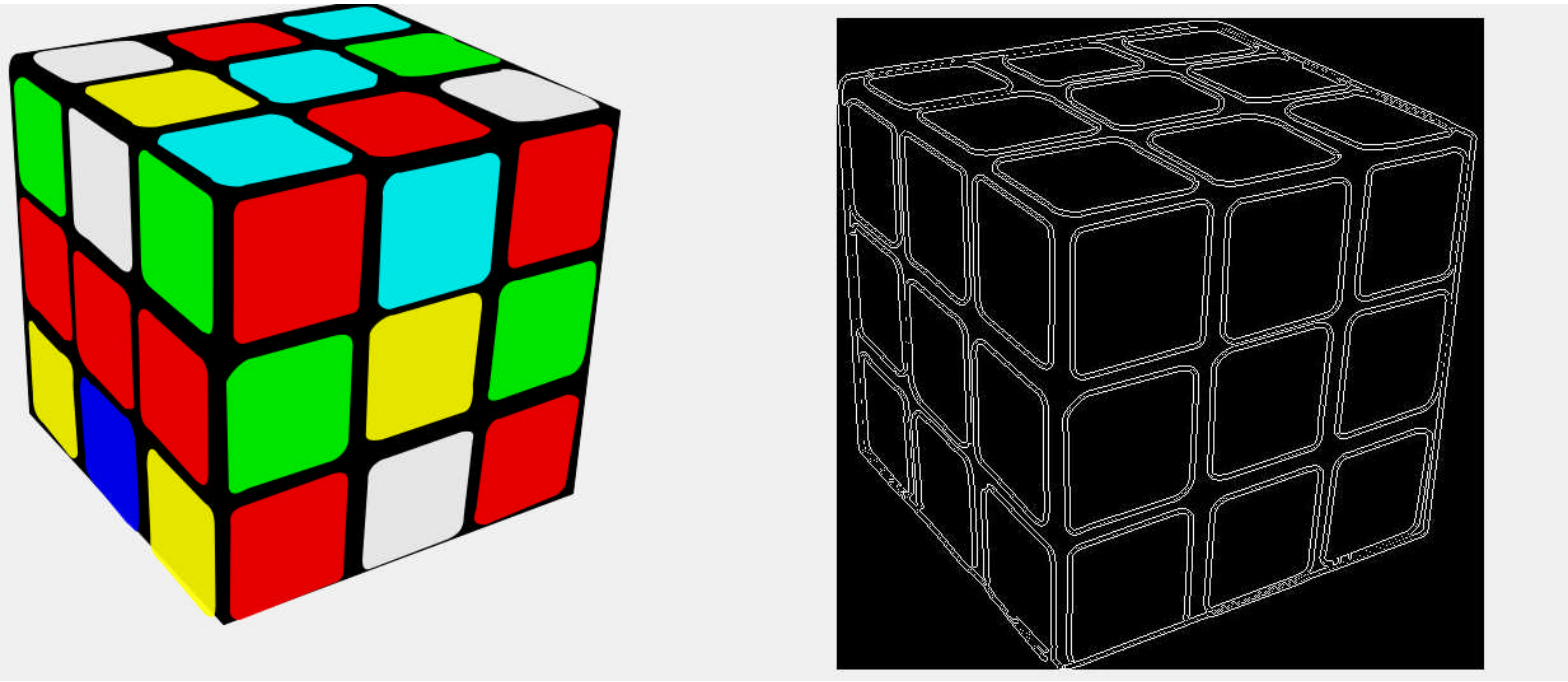
- **Making Borders for Images (Padding):**You may use the cv2.copyMakeBorder() method to build a border around an image, similar to a photo frame. However, it has additional uses for convolution operations, zero padding, and so forth.

Applications of Computer Vision(CV)

Computer vision may be used for a variety of purposes, including:

- CV plays a crucial part in both facial and iris recognition in biometric access management.
- CV lets industrial robots and self-driving automobiles to avoid accidents and travel safely.
- CV may be used in conjunction with other forms of artificial intelligence programming to automate the examination of X-rays and MRIs in digital diagnostics.
- CV allows mixed reality programs to know where a virtual item should be positioned using augmented reality.

Edge Detection



http://4.bp.blogspot.com/-W5XdeoleSss/UY9PpuykF4I/AAAAAAAAAkE/pNR275VKfrg/s1600/q1jjy.png

edunet foundation

Canny Edge Detection

A Canny edge detector is a multi-step algorithm to detect the edges for any input image. It involves the below-mentioned steps to be followed while detecting edges of an image.

- Removal of noise in input image using a Gaussian filter.
- Computing the derivative of Gaussian filter to calculate the gradient of image pixels to obtain magnitude along x and y dimension.
- Considering a group of neighbors for any curve in a direction perpendicular to the given edge, suppress the non-max edge contributor pixel points.
- Lastly, use the Hysteresis Thresholding method to preserve the pixels higher than the gradient magnitude and neglect the ones lower than the low threshold value.

Edge Detection Using OpenCV

```python
import cv2

# Read the original image
img = cv2.imread('test.jpg')
# Display original image
cv2.imshow('Original', img)
cv2.waitKey(0)

# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

# Sobel Edge Detection
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combined X and Y Sobel Edge Detection
# Display Sobel Edge Detection Images
cv2.imshow('Sobel X', sobelx)
cv2.waitKey(0)
cv2.imshow('Sobel Y', sobely)
cv2.waitKey(0)
cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
cv2.waitKey(0)

# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge Detection
# Display Canny Edge Detection Image
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Ref : https://learnopencv.com/edge-detection-using-opencv/

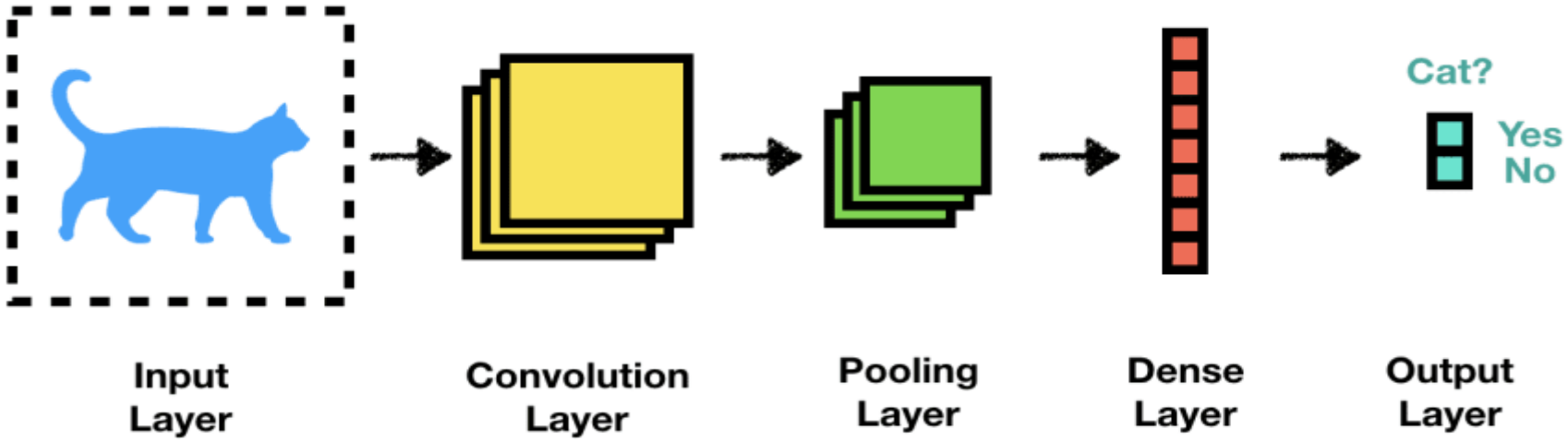
Image Filtering Using Convolution

```python
import cv2
import numpy as np

image = cv2.imread('test.jpg')

# Print error message if image is null
if image is None:
    print('Could not read image')

# Apply identity kernel
kernel1 = np.array([[0, 0, 0],
                    [0, 1, 0],
                    [0, 0, 0]])

identity = cv2.filter2D(src=image, ddepth=-1, kernel=kernel1)

cv2.imshow('Original', image)
cv2.imshow('Identity', identity)

cv2.waitKey()
cv2.imwrite('identity.jpg', identity)
cv2.destroyAllWindows()

# Apply blurring kernel
kernel2 = np.ones((5, 5), np.float32) / 25
img = cv2.filter2D(src=image, ddepth=-1, kernel=kernel2)

cv2.imshow('Original', image)
cv2.imshow('Kernel Blur', img)

cv2.waitKey()
cv2.imwrite('blur_kernel.jpg', img)
cv2.destroyAllWindows()
```

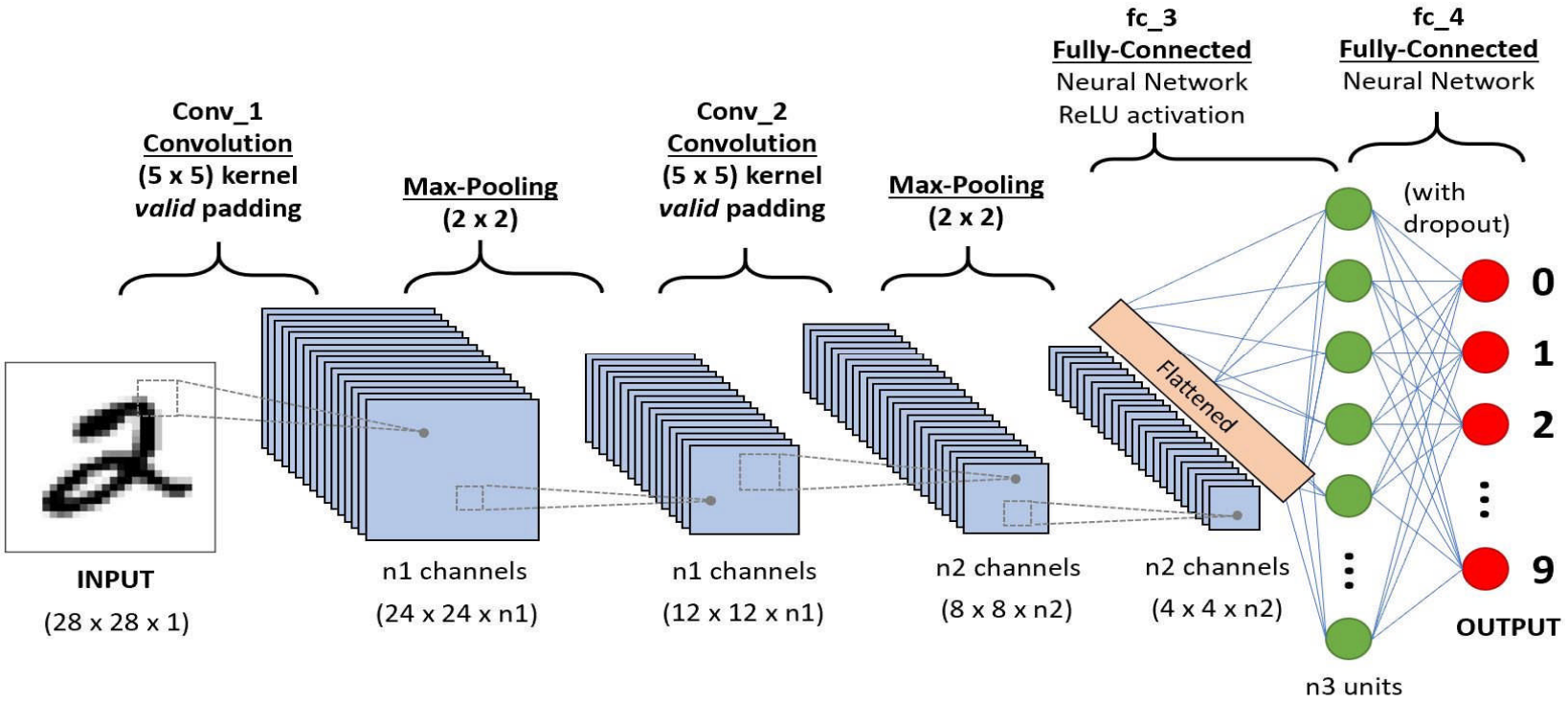
Ref: https://learnopencv.com/image-filtering-using-convolution-in-opencv/

Convolutional Neural Network

- CNNs have stacked layered architecture of several Convolution and Pooling Layers



Convolutional Neural Network



edunet foundation

Understanding CNNs

1. Convolutional Layers.
2. Pooling Layers.
3. Fully-Connected Layers.
4. Padding

Blur Filters

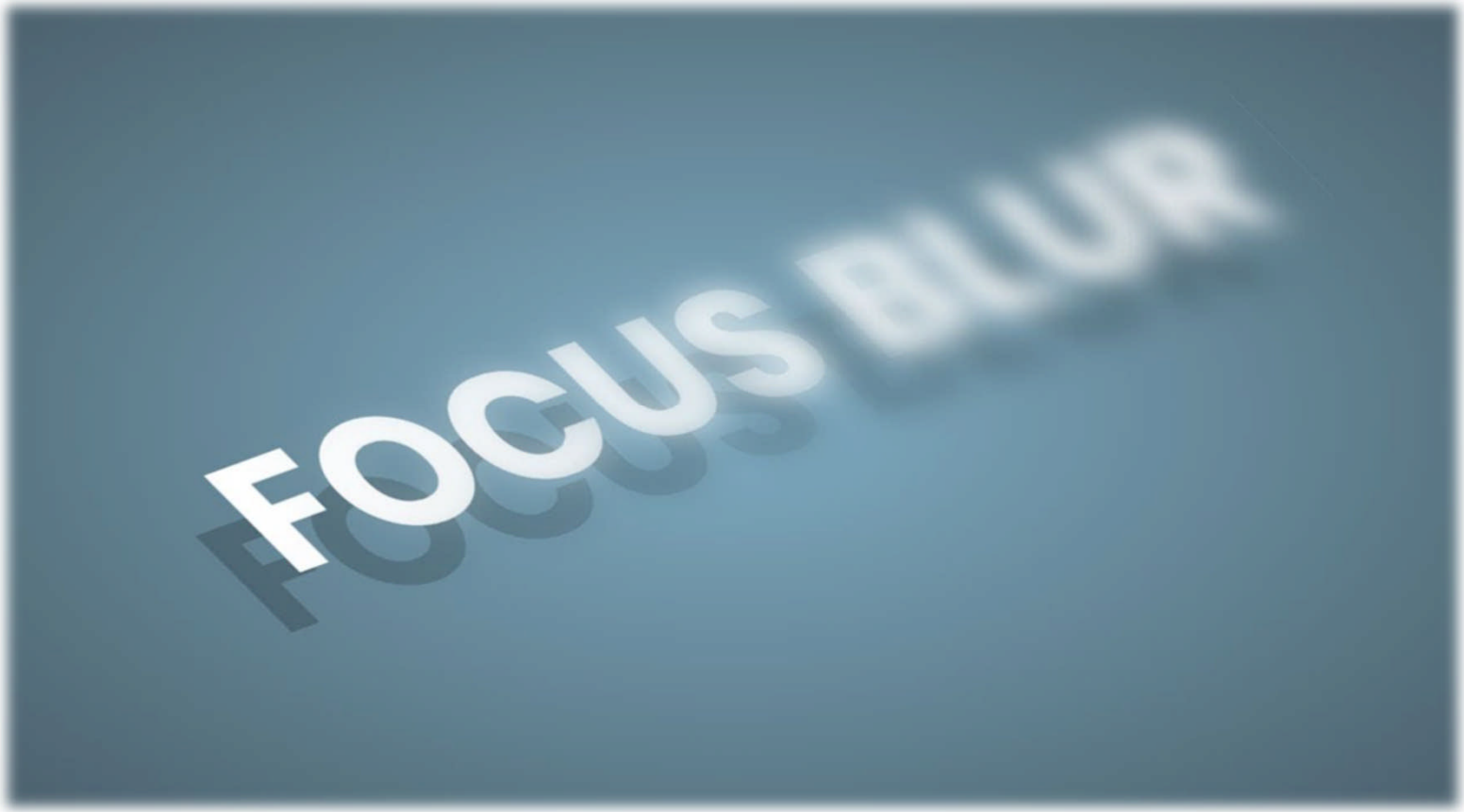


Image Kernel

Small matrix applied to an entire image

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

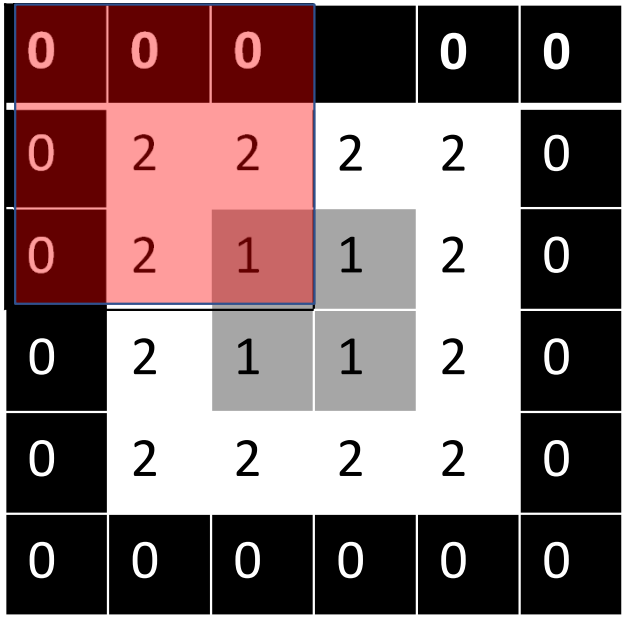
$$\begin{matrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{matrix}$$

Blur Filter

- Filters are referred to as Convolution Kernels.
- The process of passing the Kernels over an image is called Convolution

Note : Padding can be used if you don't want to lose border information

Filter



0	0	0
0	-1	1
0	1	-1

3*3 FILTER

0X0	0X0	0X0
0X0	2X-1	2X1
0X0	2X1	1X-1

Multiply by Filter Weights

0	0	0
0	-2	2
0	2	-1

SUM THE RESULT = 1

Stride Distance = (1,1)

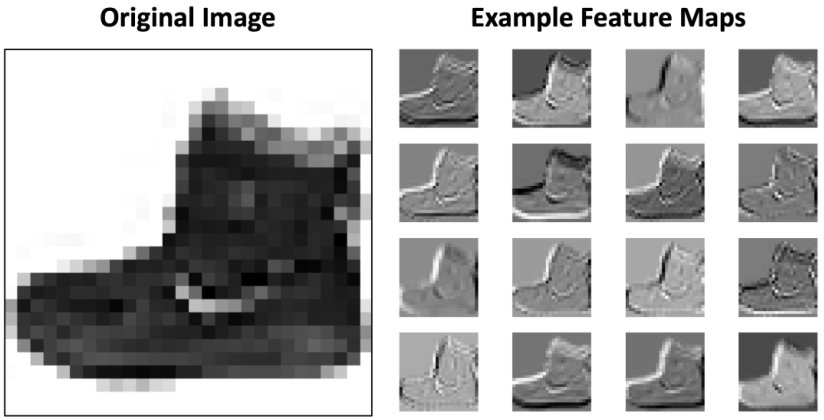
Note: The resolution will decrease because we are taking 9 input values and one output value

Different types of CNNs

- **1D CNN**: With these, the CNN kernel moves in one direction. 1D CNNs are usually used on time-series data.
- **2D CNN**: These kinds of CNN kernels move in two directions. You'll see these used with image labelling and processing.
- **3D CNN**: This kind of CNN has a kernel that moves in three directions. With this type of CNN, researchers use them on 3D images like CT scans and MRIs.

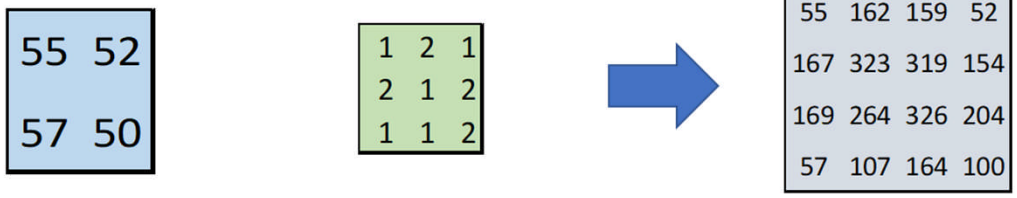
Convolutional Layers

- Perform FEATURE EXTRACTION on **Source Images** by applying **filters**(eg. 16)
- Understand various pattern with regard to Image Structures (Textures, Edges, Corners and Patterns)
- They extract **Feature Maps**

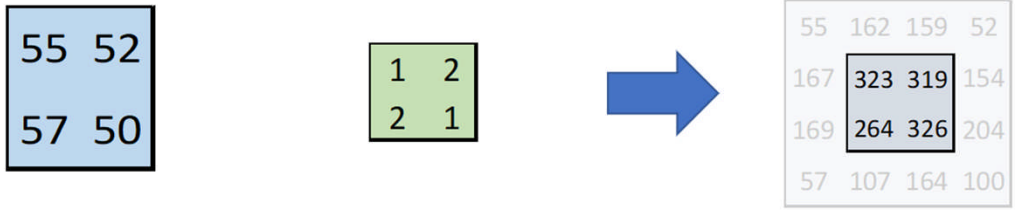


Padding

Padding: “valid”



Padding: “same”



1. valid
2. same

https://miro.medium.com/max/2264/1*XUJwcj95wk-LslLwzJHaAg.png

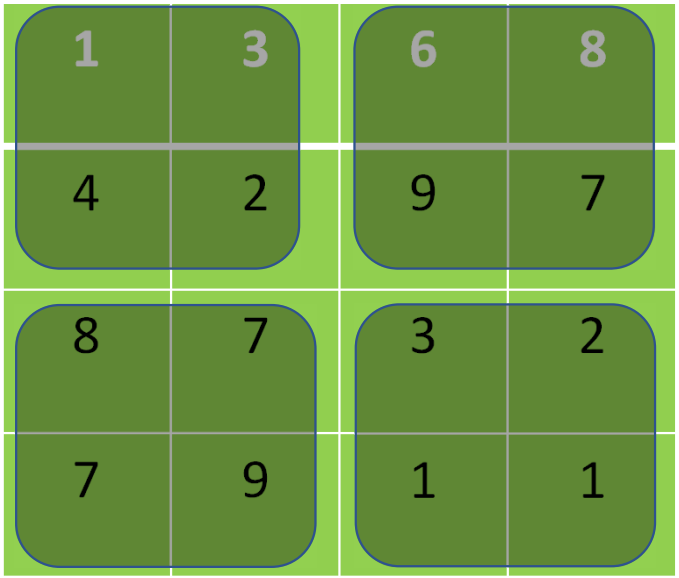
Padding

Padding is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero.

Same Padding - In this type of padding, the padding layers append zero values in the outer frame of the images or data so the filter we are using can cover the edge of the matrix and make the inference with them too.

Valid Padding - This type of padding can be considered as no padding

Max Pooling



Downsamples Feature maps from Conv Layers

4	**9**
9	3

Window 2x2 Stride : 2

Pooling Layers help in reducing Dimensionality after convolutions(compression)

edunet foundation

References

1. https://www.cs.ryerson.ca/~aharley/vis/
2. https://setosa.io/ev/image-kernels/
3. https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967
4. https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm
5. https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9

edunet
foundation

THANK YOU