

DIGITAL IMAGE PROCESSING (LAB)

Lab Manual

[Spring 2018]

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
| Instructor: *Dr. Noman Islam*  Submitted by: Khalid Anwer (10396) |  |

April 24, 2018

**Lab 1: To setup the environment and familiarize with Python**

1. **Write a small program in Python to print your CV.**

print(**"Name:\n \t\t\t Iqtada Kamal"**)  
print(**"CGPA:\n \t\t\t3.5"**)  
print(**"Semester:\n \t\t\t6th"**)  
print(**"University:\n \t\t\tIqra University Qulshan Campus"**)  
print(**"Skills:\n \t\t\tGood in java,python,"**)  
print(**"Interested In:\n \t\t\tcricket"**)  
print(**"Future Plans:\n \t\t\ app development"**)  
print(**"Experience:\n \t\t\tNone"**)

Name:

Iqtada Kamal

CGPA:

3.5

Semester:

6th

University:

Iqra University Qulshan Campus

Skills:

Good in java,python

Interested In:

cricket

Future Plans:

App development

Experience:

None

1. **Write a program that takes the month (1…12) as input. Print whether the season is summer, winter, spring or autumn depending upon the input month.**

a = int(input(**'Enter the Month\n'**))  
**if**(a==1 **or** a==2 **or** a==3):  
 {print(**"Winter\n"**)  
 }  
**elif**(a==4 **or** a==5 **or** a==6):  
 {

print(**"Spring"**)  
 }  
**elif** (a == 7 **or** a == 8 **or** a == 9):  
 {  
 print(**"Summer"**)  
 }  
**elif** (a == 10 **or** a == 11 **or** a == 12):  
 {  
 print(**"Autumn"**)  
 }  
**else**:print(**"the month does not exist"**)

Enter the Month

6

Spring

1. **To determine whether a year is a leap year, follow these steps:**
   1. **If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.**
   2. **If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.**
   3. **If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.**
   4. **The year is a leap year (it has 366 days).**
   5. **The year is not a leap year (it has 365 days).**

**Write a program to input an year as integer. Using if…else, determines whether the input is a leap year or not.**

x = int(input(**'Enter the Year\n'**))  
**if**(x%4==0):  
 **if**(x%100==0):  
 **if**(x%400==0):  
 {  
 print(**"leap year"**)  
 }  
 **else**:{  
 print(**"Not leap year"**)  
 }  
 **else**:{  
 print(**"leap year"**)  
 }  
**else**:{  
 print(**"Not leap year"**)  
}

Enter the Year

2005

Not leap year

**4. Write a program that takes a line as input and finds the number of letters and digits in the input.**

x = input(**"enter the String\n"**)  
y=z=0  
**for** s **in** x:  
 **if** (s.isdigit()):  
 y=y+1  
 **if** (s.isalpha()):  
 z=z+1  
print(**"letter and digit"**,z,y)

enter the String

iqra123

letter and digit 4 3

**5. Write a program that takes a sentence as input. Compute the frequency of each words and prints them.**

a = input(**'enter the sentence \n'**)

x = len(a.split())  
 print(**"Number of the word are "**,x)

enter the sentence

this is a boy

Number of the word are 4

**Lab 2: To study and implement basic algorithms in Python**

1. **Write a program to generate a dictionary that contains (i,sqrt(i)), where *i* is an integer between 1 and n. *n* is a number input by the user.**

**from** math **import** sqrt  
a = int(input(**"enter a number"**))  
dic = {}  
**for** r **in** range(1,a+1):  
 dic [r] = sqrt(r)  
print(dic)

enter a number3

{1: 1.0, 2: 1.4142135623730951, 3: 1.7320508075688772}

**2. Write a simple calculator program using functions add, sub, mul and div. The program should accepts two numbers and an operator and calls the corresponding function to perform the operation.**

**import** math  
a = int(input(**"enter the 1st num"**))  
b = int(input(**"enter the 2nd num"**))  
op = input(**"enter the operator"**)  
**def** add(a,b):  
 z = a+b  
 **return** z  
**def** sub(a,b):  
 z = a-b  
 **return** z  
**def** mul(a,b):  
 z = a\*b  
 **return** z  
**def** div(a,b):  
 z = a/b  
 **return** z  
**if**(op == **"+"**):  
 z = add(a,b)  
 print(z)  
**if**(op == **"-"**):  
 z = sub(a,b)  
 print(z)  
**if**(op == **"\*"**):  
 z = mul(a,b)  
 print(z)  
**if**(op == **"/"**):  
 z = div(a,b)  
 print(z)

enter the 1st num 3

enter the 2nd num 4

enter the operator \*

12

**3. Write a function that generates a list with values that are square of number between 1 and 20.**

list = []  
**def** sqr(a):  
 p = a\*a  
 **return** p  
**for** r **in** range(1,21):  
 list.append(sqr(r))  
 print(**"the square of "**,r,**" is "**,list[r-1])

The square of 1 is 1

The square of 2 is 4

The square of 3 is 9

The square of 4 is 16

The square of 5 is 25

The square of 6 is 36

The square of 7 is 49

The square of 8 is 64

The square of 9 is 81

The square of 10 is 100

The square of 11 is 121

The square of 12 is 144

The square of 13 is 169

The square of 14 is 196

The square of 15 is 225

The square of 16 is 256

The square of 17 is 289

The square of 18 is 324

The square of 19 is 361

The square of 20 is 400

**4. Define a class named Shape with static method printType. Define methods draw() and area(). Now define two class Rectangle and Triangle. Rectangle has two attributes length and width. The Triangle class has attributes a,b and c. Override the two methods of shape class. Demonstrate the functionality of class by creating its objects.**

**class** Shape():  
 @staticmethod  
 **def** printType():  
 print(**"static method"**)  
 **def** draw(self):  
 print(**"draw"**)  
 **def** area(self):  
 print(**"area"**)  
  
**class** rectangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.width = 23  
 self.length =34  
**class** triangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.a = 2  
 self.b = 3  
 self.c = 4  
 **def** draw(self):  
 print(**"draw again2"**)  
 **def** area(self):  
 print(**"area again2"**)  
s = Shape()  
Shape.printType()  
t = triangle()  
r = rectangle()  
s.area()  
s.draw()  
r.draw()  
t.draw()

static method

area

draw

draw

draw again2

**5. Using recursion, write a program to calculate the reverse of a string.**

str = input(**"Enter the String"**)  
**def** rec(st):  
 **if**(st==**""**):  
 **return ""  
 else**:  
 **return** rec(st[1:])+st[0]  
a = rec(str)  
print(a)

Enter the String

Khan Sahib

bihaS nahK

**Lab 3: To study and understand numpy library**

**a. Import the "numpy" library as "np".**

import numpy as np

**b. Create an array of shape (2, 3, 4) of zeros.**

a = np.zeros((2,3,4))

print(a)

**c. Create an array of shape (2, 3, 4) of ones**

import numpy as np

a = np.ones((2,3,4))

print(a)

**d. Create an array with values 0 to 999 using the "np.arange" function**

import numpy as np

y = np.arange(0,999)

print(y)

**e. Create an array from the list [2, 3.2, 5.5, -6.4, -2.2, 2.4] and assign it to the variable "a"**

a = [2,3.2,5.5,-6.4,-2.2,2.4]

x = np.array(a)

print(x)

**f. Do you know what a[1] will equal? Print it to see**

a = [2,3.2,5.5,-6.4,-2.2,2.4]

x = np.array(a)

print(x[1])

**g. Try printing a[1:4] to see what that equals**

a = [2,3.2,5.5,-6.4,-2.2,2.4]

x = np.array(a)

print(a[1:4])

**h. Create a 2-D array from the following list and assign it to the variable "a": [[2, 3.2, 5.5, -6.4, -2.2, 2.4], [1, 22, 4, 0.1, 5.3, -9], [3, 1, 2.1, 21, 1.1, -2]]**

z = [[2, 3.2, 5.5, -6.4, -2.2, 2.4], [1, 22, 4, 0.1, 5.3, -9], [3, 1, 2.1, 21, 1.1, -2]]

x = np.array(z)

print(x)

**i. Can you guess what the following slices are equal to? Print them to check your understanding. a[:, 3] a[1:4, 0:4] a[1:, 2]**

z = [[2, 3.2, 5.5, -6.4, -2.2, 2.4], [1, 22, 4, 0.1, 5.3, -9], [3, 1, 2.1, 21, 1.1, -2]]

x = np.array(z)

print(x[:, 3])

print(x[1:4, 0:4])

print(x[1:, 2])

**j. Create a 2-D array of shape (2, 4) containing two lists (range(4), range(10, 14)) and assign it to the variable "arr".Print the shape of the array. Print the size of the array. Print the maximum and minimum of the array**

arr = np.array([range(0,4),range(10,14)])

print(arr)

print("shape",arr.shape)

print("size",arr.size)

print("maximum",arr.max())

print("minimum",arr.min())

**k. Continue to use the array "arr" as defined above.Print the array re-shaped to (2, 2, 2).Print the array transposed.Print the array flattened to a single dimension. Print the array converted to floats.**

arr = np.array([range(0,4),range(10,14)])

print(arr)

print("shape",arr.shape)

y = np.reshape(arr, (2, 2,2))

print(y)

**l. Create an an array counting from 1 to 20 inclusive**

a = np.array(range(1,21))

a

**m. The array of multiples of 3 greater than 0 and less than 30**

a = np.array(range(0,30,3))

a

**n. The array of 8 equally spaced floats x where 0 ≤ x ≤ 1**

x=np.linspace(0,1,8)

print(x)

**o. Use np.arange and reshape to create the array A = [[1 2 3 4] [5 6 7 8]]**

A =np.arange(1,9)

A.reshape(2,4)

**p. Use np.array to create the array B = [1 2]**

B = np.array([[1],[2]])

**q. Use broadcasting to add B to A to create the final array A + B**

A+B

​

**Lab 4: To study and implement pandas library**

**1. Create a data series with marks of students : 75, 80, 79, 60**

import pandas as pd

s = pd.Series([75,80,79,60])

print(s)

print(s.max())

print(s.min())

print(s[2])

print(s[2:0:-1])

**2. Create a data frame with name of students, id and marks**

pd.DataFrame({

'patient':['Ali','Fatima','Zain'],

'Age':[25,21,21],

'Disease':['Cough','Obesity','Baldness']

})

**3. Now read the file 'data.csv' in panda**

data = pd.read\_csv('data.csv')

data

**4. What are the columns in the dataframe?**

data.columns

**5. Sort the data based on Marks obtained. Fill all the 'na' cells with 0**

data.sort\_values(by='Mid')

data.Mid

data['Student Name']

data[['Student Name','Mid']]

data.fillna(0)

**6. Display the top 10 rows**

data.head(10)

**7. Display the last 10 rows**

data.tail(10)

**8. Display only the odd rows**

data[0: :3]

**9. Display only those students who got failed in examination**

data[data.Grade=='F']

**10. Find out the basic statistical info about data**

data.describe

**11. How many students got A, B, C, F?**

data.groupby('Grade').count()

**12. What are the mean scores for students who got A, B, C, F?**

data.groupby('Grade').mean()

**Lab 5: To study and implement cv2 library**

**1.**

import cv2

import numpy as np

img = cv2.imread('lena.png',cv2.IMREAD\_GRAYSCALE)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**2.**

import cv2

import numpy as np

img = np.zeros((512,512,3),np.uint8)

img = cv2.line(img,(0,0),(511,511),(255,0,0),5)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**3.**

import cv2

import numpy as np

img = np.ones((512,512,3),np.uint8)\*255

img = cv2.line(img,(0,0),(511,511),(255,0,0),5)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**4.**

import cv2

import numpy as np

img = np.ones((300,350,3))

cv2.circle(img,(100,100),45,color=(255,0,0),thickness=5)

cv2.circle(img,(180,100),45,color=(0,0,0),thickness=5)

cv2.circle(img,(260,100),45,color=(0,0,255),thickness=5)

cv2.circle(img,(140,135),45,color=(0,255,255),thickness=5)

cv2.circle(img,(220,135),45,color=(0,255,0),thickness=5)

cv2.imshow('Olympics',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**5.**

import cv2

import numpy as np

img = cv2.imread('lena.png',cv2.IMREAD\_GRAYSCALE)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**6.**

import cv2

import numpy as np

face =img[20:170,40:190]

img[0:150,0:150]=face

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**7.**

import cv2

import numpy as np

img= cv2.copyMakeBorder(img,20,20,20,20,cv2.BORDER\_CONSTANT)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**8.**

import cv2

import numpy as np

img = cv2.imread('pepsi.jpg')

img = cv2.copyMakeBorder(img,20,20,20,20,cv2.BORDER\_CONSTANT)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Lab 6: To study and implement cv2**

**1. Circle**

import cv2

import numpy as np

img = np.ones((512,512,3),np.uint8)\*255

cv2.circle(img,(256,256),60,(0,0,255))

cv2.circle(img,(256,256),30,(255,3,255),-1)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**2. Pacman**

import cv2

import numpy as np

img = np.ones((512,512,3), np.uint8)\*255

img = cv2.circle(img,(256,256), 63, (128,0,128))

img = cv2.line(img,(256,256),(256,200),(255,0,0),2)

img = cv2.line(img,(256,256),(230,200),(255,0,0),2)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**3. Border**

import cv2

import numpy as np

img = cv2.imread('lena.png',cv2.IMREAD\_GRAYSCALE)

img = cv2.copyMakeBorder(img,20,20,20,20,cv2.BORDER\_CONSTANT)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**4. Rectangle**

import cv2

import numpy as np

img = np.ones((512,512,3),np.uint8)\*255

img = cv2.rectangle(img,(150,250),(510,128),(0,255,0),3)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**5. Hangman**

import numpy as np

import cv2

img = np.ones((512,512,3), np.uint8)\*255

img = cv2.circle(img,(256,256), 63, (128,0,128))

img = cv2.line(img,(256,256),(256,200),(0,0,255),2)

img = cv2.line(img,(256,256),(256,160),(0,0,255),2)

img = cv2.line(img,(256,370),(256,320),(0,0,255),2)

img = cv2.line(img,(284,346),(220,346),(0,0,255),2)

img = cv2.line(img,(257,367),(218,389),(0,0,255),2)

img = cv2.line(img,(257,367),(292,388),(0,0,255),2)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**6. Parallelogram**

import cv2

import numpy as np

img = np.ones((512,512,3),np.uint8)\*255

img = cv2.line(img,(193,246),(155,303),(255,0,0),2)

img = cv2.line(img,(193,246),(310,246),(255,0,0),2)

img = cv2.line(img,(271,307),(310,246),(255,0,0),2)

img = cv2.line(img,(156,306),(272,308),(255,0,0),2)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Lab 7: To study and implement cv2 filters**

**1. ROTATE**

import cv2

import numpy as np

img = cv2.imread('lena.jpg',0)

rows,cols = img.shape

M = cv2.getRotationMatrix2D((cols/2,rows/2),45,1)

dst = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow('image',dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

**2. Thresholding**

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('pic.jpg',0)

ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY\_INV)

ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH\_TRUNC)

ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH\_TOZERO)

ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH\_TOZERO\_INV)

titles = ['Original Image','BINARY','BINARY\_INV','TRUNC','TOZERO','TOZERO\_INV']

images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(6):

plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()

# 3. Adaptive Thresholding

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('pic.jpg',0)

img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,\

cv2.THRESH\_BINARY,11,2)

th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,\

cv2.THRESH\_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',

'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

for i in range(4):

plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()

# 4. Otsu’s Binarization

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('lena2.jpg',0)

ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

blur = cv2.GaussianBlur(img,(5,5),0)

ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

images = [img, 0, th1,

img, 0, th2,

blur, 0, th3]

titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',

'Original Noisy Image','Histogram',"Otsu's Thresholding",

'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

for i in range(3):

plt.subplot(3,3,i\*3+1),plt.imshow(images[i\*3],'gray')

plt.title(titles[i\*3]), plt.xticks([]), plt.yticks([])

plt.subplot(3,3,i\*3+2),plt.hist(images[i\*3].ravel(),256)

plt.title(titles[i\*3+1]), plt.xticks([]), plt.yticks([])

plt.subplot(3,3,i\*3+3),plt.imshow(images[i\*3+2],'gray')

plt.title(titles[i\*3+2]), plt.xticks([]), plt.yticks([])

plt.show()

# 5. Averaging Filter

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('pic2.png')

blur = cv2.blur(img,(5,5))

plt.subplot(121),plt.imshow(img),plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(blur),plt.title('Averaging')

plt.xticks([]), plt.yticks([])

plt.show()

# 6. Gaussian Filtering

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('pic2.png')

blur = cv2.GaussianBlur(img,(5,5),0)

plt.subplot(121),plt.imshow(img),plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(blur),plt.title('Gaussian')

plt.xticks([]), plt.yticks([])

plt.show()

# 7. Median Filtering

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('pic2.png')

median = cv2.medianBlur(img,5)

plt.subplot(121),plt.imshow(img),plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(median),plt.title('Median')

plt.xticks([]), plt.yticks([])

plt.show()

# 8. Erosion

import cv2

import numpy as np

img = cv2.imread('pic3.jpg',0)

kernel = np.ones((5,5),np.uint8)

erosion = cv2.erode(img,kernel,iterations = 1)

cv2.imshow('image',erosion)

cv2.waitKey(0)

cv2.destroyAllWindows()

# 9. Dilation

import cv2

import numpy as np

img = cv2.imread('pic3.jpg',0)

kernel = np.ones((5,5),np.uint8)

dilation = cv2.dilate(img,kernel,iterations = 1)

cv2.imshow('image',dilation)

cv2.waitKey(0)

cv2.destroyAllWindows()

# 10. Opening

import cv2

import numpy as np

img = cv2.imread('pic3.jpg',0)

kernel = np.ones((5,5),np.uint8)

opening = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)

cv2.imshow('image',opening)

cv2.waitKey(0)

cv2.destroyAllWindows()

# 11. Closing

import cv2

import numpy as np

img = cv2.imread('pic3.jpg',0)

kernel = np.ones((5,5),np.uint8)

closing = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)

cv2.imshow('image',closing)

cv2.waitKey(0)

cv2.destroyAllWindows()

# 12. Sobel And Laplacian

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('lena.jpg',0)

laplacian = cv2.Laplacian(img,cv2.CV\_64F)

sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')

plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')

plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()

# 13. Canny Edge Detection

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('lena.jpg',0)

edges = cv2.Canny(img,100,200)

plt.subplot(121),plt.imshow(img,cmap = 'gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(edges,cmap = 'gray')

plt.title('Canny Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()

# 14. Fourier Transform

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('afridi.jpg',0)

f = np.fft.fft2(img)

fshift = np.fft.fftshift(f)

magnitude\_spectrum = 20\*np.log(np.abs(fshift))

plt.subplot(121),plt.imshow(img, cmap = 'gray')

plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(magnitude\_spectrum, cmap = 'gray')

plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])

plt.show()

# 15. Template Matching

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('afridi.jpg',0)

img2 = img.copy()

template = cv2.imread('afridi2.jpg',0)

w, h = template.shape[::-1]

methods = ['cv2.TM\_SQDIFF']

for meth in methods:

img = img2.copy()

method = eval(meth)

res = cv2.matchTemplate(img,template,method)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

if method in [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:

top\_left = min\_loc

else:

top\_left = max\_loc

bottom\_right = (top\_left[0] + w, top\_left[1] + h)

cv2.rectangle(img,top\_left, bottom\_right, 255, 2)

plt.subplot(121),plt.imshow(res,cmap = 'gray')

plt.title('Matching Result'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(img,cmap = 'gray')

plt.title('Detected Point'), plt.xticks([]), plt.yticks([])

plt.suptitle(meth)

plt.show()

# 16. Histogram

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('flower.jpg')

color = ('b','g','r')

for i,col in enumerate(color):

histr = cv2.calcHist([img],[i],None,[256],[0,256])

plt.plot(histr,color = col)

plt.xlim([0,256])

plt.show()

# 17. Histogram Equalization

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('lena.jpg',0)

hist,bins = np.histogram(img.flatten(),256,[0,256])

cdf = hist.cumsum()

cdf\_normalized = cdf \* hist.max()/ cdf.max()

plt.plot(cdf\_normalized, color = 'b')

plt.hist(img.flatten(),256,[0,256], color = 'r')

plt.xlim([0,256])

plt.legend(('cdf','histogram'), loc = 'upper left')

plt.show()

**Lab 8: To study and implement CNN Model**

**1. CNN**

from keras import layers

from keras import models

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(10, activation='softmax'))

from keras.datasets import mnist

from keras.utils import to\_categorical

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

train\_images = train\_images.reshape((60000, 28, 28, 1))

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.reshape((10000, 28, 28, 1))

test\_images = test\_images.astype('float32') / 255

train\_labels = to\_categorical(train\_labels)

test\_labels = to\_categorical(test\_labels)

model.compile(optimizer='rmsprop',

loss='categorical\_crossentropy',

metrics=['accuracy'])

model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

test\_acc

**2. DATA AUGMENTATION**

from keras.preprocessing.image import ImageDataGenerator

import os

datagen = ImageDataGenerator(

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

from keras.preprocessing import image

import matplotlib.pyplot as plt

img = image.load\_img('cat.jpg', target\_size=(150, 150))

x = image.img\_to\_array(img)

x = x.reshape((1,) + x.shape)

i = 0

for batch in datagen.flow(x, batch\_size=1):

plt.figure(i)

imgplot = plt.imshow(image.array\_to\_img(batch[0]))

i += 1

if i % 10 == 0:

break

plt.show()

**Lab 9: To study and implement KNN Model**

**1. KNN**

import numpy as np

import cv2

gray = cv2.imread('digits.png',0)

cells = [np.hsplit(row,100) for row in np.vsplit(gray,50)]

x = np.array(cells)

# Now we prepare train\_data and test\_data.

train = x[:,:50].reshape(-1,400).astype(np.float32) # Size = (2500,400)

test = x[:,50:100].reshape(-1,400).astype(np.float32) # Size = (2500,400)

k = np.arange(10)

train\_labels = np.repeat(k,250)[:,np.newaxis]

test\_labels = train\_labels.copy()

knn = cv2.ml.KNearest\_create()

knn.train(train, cv2.ml.ROW\_SAMPLE, train\_labels)

ret,result,neighbours,dist = knn.findNearest(test,k=5)

matches = result==test\_labels

correct = np.count\_nonzero(matches)

accuracy = correct\*100.0/result.size

print(accuracy)

**2. ENGLISH OCR**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the data, converters convert the letter to a number

data= np.loadtxt('letter-recognition.data.txt', dtype= 'float32', delimiter = ',',

converters= {0: lambda ch: ord(ch)-ord('A')})

# split the data to two, 10000 each for train and test

train, test = np.vsplit(data,2)

# split trainData and testData to features and responses

responses, trainData = np.hsplit(train,[1])

labels, testData = np.hsplit(test,[1])

knn = cv2.ml.KNearest\_create()

knn.train(trainData, cv2.ml.ROW\_SAMPLE,responses)

ret, result, neighbours, dist = knn.findNearest(testData, k=5)

correct = np.count\_nonzero(result == labels)

accuracy = correct\*100.0/10000

print(accuracy)

**3. PRETRAINED**

from keras.applications import VGG16

conv\_base = VGG16(weights='imagenet',

include\_top=False,

input\_shape=(150, 150, 3))

conv\_base.summary()

from keras import models

from keras import layers

model = models.Sequential()

model.add(conv\_base)

model.add(layers.Flatten())

model.add(layers.Dense(256, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

conv\_base.trainable = True

set\_trainable = False

for layer in conv\_base.layers:

if layer.name == 'block5\_conv1':

set\_trainable = True

if set\_trainable:

layer.trainable = True

else:

layer.trainable = False

train\_dir='data'

validation\_dir='data'

from keras.preprocessing.image import ImageDataGenerator

from keras import optimizers

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(150, 150),

batch\_size=20,

class\_mode='binary')

validation\_generator = test\_datagen.flow\_from\_directory(

validation\_dir,

target\_size=(150, 150),

batch\_size=20,

class\_mode='binary')

model.compile(loss='binary\_crossentropy',

optimizer=optimizers.RMSprop(lr=2e-5),

metrics=['acc'])

history = model.fit\_generator(

train\_generator,

steps\_per\_epoch=10,

epochs=2,

validation\_data=validation\_generator,

validation\_steps=10)

**4. DATA AUGMENTATION**

from keras.preprocessing import image

from keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

import scipy.misc

img = image.load\_img('cat.png', target\_size=(150, 150))

x = image.img\_to\_array(img)

x = x.reshape((1,) + x.shape)

i = 0

datagen = ImageDataGenerator(

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

i=0

for batch in datagen.flow(x, batch\_size=1):

i += 1

scipy.misc.imsave('cat/cat'+str(i)+'.png', batch[0])

if i % 100 == 0:

break

img = image.load\_img('dog.png', target\_size=(150, 150))

x = image.img\_to\_array(img)

x = x.reshape((1,) + x.shape)

i = 0

for batch in datagen.flow(x, batch\_size=1):

i += 1

scipy.misc.imsave('dog/dog'+str(i)+'.png', batch[0])

if i % 100 == 0:

break

from keras import layers

from keras import models

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu',

input\_shape=(150, 150, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

from keras import optimizers

model.compile(loss='binary\_crossentropy',

optimizer=optimizers.RMSprop(lr=1e-4),

metrics=['acc'])

train\_dir='data'

validation\_dir='data'

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1./255)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,target\_size=(150, 150),batch\_size=20,

class\_mode='binary')

validation\_generator = test\_datagen.flow\_from\_directory(

validation\_dir,target\_size=(150, 150),

batch\_size=20,

class\_mode='binary')

for data\_batch, labels\_batch in train\_generator:

print('data batch shape:', data\_batch.shape)

print('labels batch shape:', labels\_batch.shape)

break

history = model.fit\_generator(

train\_generator,

steps\_per\_epoch=2,

epochs=5,

validation\_data=validation\_generator,

validation\_steps=50)

import matplotlib.pyplot as plt

acc = history.history['acc']

val\_acc = history.history['val\_acc']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')

plt.plot(epochs, val\_acc, 'b', label='Validation acc')

plt.title('Training and validation accuracy')

plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.legend()

plt.show()