

DIGITAL IMAGE PROCESSING

Lab Manual

[Spring 2018]

|  |  |
| --- | --- |
| Student Name: Nazeer Bin Zafar |  |
| Student Id: 10413 |  |
|  |  |

|  |  |
| --- | --- |
| Prepared By: *Dr. Noman Islam* |  |
| Instructor: *Dr. Noman Islam* |  |

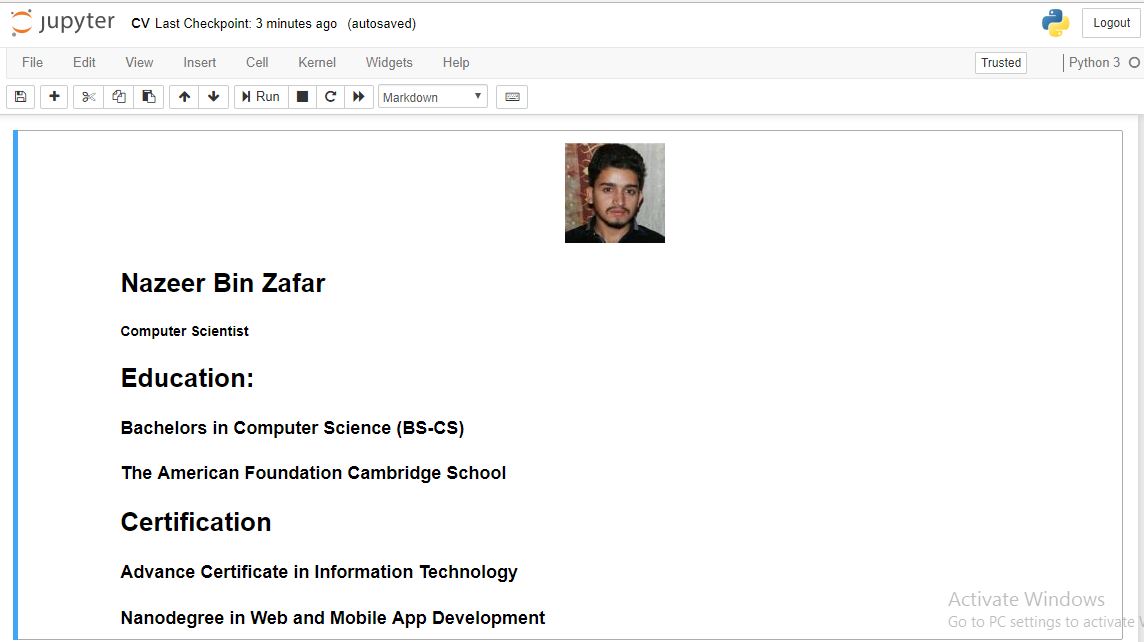
**LIST OF EXPERIMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Date** | **Experiment** |  |
| **1** | \_\_/\_\_/\_\_ | To setup the environment and familiarize with Python |  |
| **2** | \_\_/\_\_/\_\_ | To study and implement algorithms in Python |  |
| **3** | \_\_/\_\_/\_\_ | To study and understand numpy library |  |
| **4** | \_\_/\_\_/\_\_ | To study and implement pandas library |  |
| **5** | \_\_/\_\_/\_\_ | To install OpenCV and study basics of Open CV |  |
| **6** | \_\_/\_\_/\_\_ | To study and implement basic image processing operations in OpenCV |  |
| **7** | \_\_/\_\_/\_\_ | To study and implement advanced image processing operations in OpenCV |  |
| **8** | \_\_/\_\_/\_\_ | To study and implement machine learning algorithm using OpenCV |  |
| **9** | \_\_/\_\_/\_\_ | To study and implement convolutional neural network using OpenCV |  |
|  |  |  |

. **Lab Tasks:**

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

1. Write a small program in jupyter-notebook to print your CV.



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.

month = int(input());

if(month >= 3 and month <= 5 ):

print('spring');

elif(month >=6 and month <= 8):

print('Summer');

elif(month >= 9 and month <= 11):

print('Autumn');

elif(month == 12 or month == 1 or month == 2):

print('Winter');

else:

print('Incorrect input')

1. To determine whether a year is a leap year using if-else statements, follow these steps:

year = int(input());

if (year % 4 == 0 and year % 100 == 0 and year % 400 == 0):

print(str(year) + ' is a leap year');

elif(year % 4 == 0 and year % 100 != 0):

print(str(year) + ' is a leap year');

else:

print(str(year) + ' is not a leap year');

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

sentance = input();

digitCount = 0;

letterCount = 0;

for x in range(len(sentance)):

if(sentance[x].isdigit()):

digitCount = digitCount + 1;

else:

letterCount = letterCount + 1

print('No. of Digits: ' + str(digitCount) + '\nNo. of letter: '+ str(letterCount));

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

sentance = input().split(' ');

dic = {};

for x in sentance:

if x in dic:

dic[x] = dic[x] + 1

else:

dic[x] = 1

print(dic);

**Lab Tasks:**

**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.

import math;

n = int(input());

dic = {};

for i in range(1,n+1):

dic[i] = float(format(math.sqrt(i), '.2f'));

print(dic);

1. 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.

num1 = int(input('type first number\n'));

num2 = int(input('type second number\n'));

opr = input('type operator\n');

def add():

return num1 + num2;

def sub():

return num1 - num2;

def mul():

return num1 \* num2;

def div():

return num1 / num2;

def default():

return 'invalid input';

switch = {

'+': add(),

'-': sub(),

'\*': mul(),

'/': div()

}

print(switch.get(opr, default()));

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

squares = [];

for num in range(1,21):

squares.append(num\*\*2)

print(squares)

1. Demonstrate the functionality of class by creating its objects.

class Shape():

@staticmethod

def printType():

print('static method called');

def draw(self):

print('draw method called of Shape class');

def area(self):

print('area method called of Shape class');

class Rectangle(Shape):

width = 0;

length = 0;

def draw(self):

print('draw method called of Rectangle class');

def area(self):

print('area method called of Rectangle class');

class Triangle(Shape):

a = 0;

b = 0;

c = 0;

def draw(self):

print('draw method called of Triangle class');

def area(self):

print('area method called of Triangle class');

obj = Rectangle();

obj.draw()

obj = Triangle();

obj.printType()

obj = Shape();

obj.draw()

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

sentance = list(input());

reverseString = [];

counter = 10;

def reversing():

global counter;

reverseString.append(sentance[counter]);

if counter == 0:

return print(''.join(reverseString));

counter -= 1;

return reversing();

reversing();

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

**Lab Task:**

Open the Python Notebook provided with this lab and perform the tasks.

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

In [ ]:

import numpy as np

​

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

In [ ]:

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

​

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

In [ ]:

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

​

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

In [ ]:

ary = np.arange(1000)

​

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"

In [ ]:

a = np.array([2, 3.2, 5.5, -6.4, -2.2, 2.4])

​

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

In [ ]:

print(a[1])

​

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

In [ ]:

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]]

In [ ]:

a = np.array([[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]])

​

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]

In [ ]:

print(a[:4,:]); print(a[1:4,0:4]); print(a[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

In [ ]:

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

print(arr.shape)

print(arr.size)

print(arr.max())

print(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.

In [ ]:

print(arr.reshape((2,2,2)))

print(arr.T)

print(arr.reshape(8))

print(arr.astype(float))

​

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

In [ ]:

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

​

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

In [ ]:

arr = np.array([x for x in range(1,30) if x % 3 == 0])

​

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

In [ ]:

arr = np.linspace(0.0, 1.0, num=8, endpoint=False)

​

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

In [ ]:

A = np.array(np.arange(1,9))

​

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

In [ ]:

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

​

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

In [ ]:

A = np.array(np.arange(1,9))

A = A.reshape(4,2)

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

**Lab Task:**

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

Open the Python Notebook provided with this lab and perform the tasks.

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

In [ ]:

import pandas as pd

import numpy as np

data = np.array([ 75, 80, 79, 60])

result = pd.Series(data)

​

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

In [ ]:

data = pd.DataFrame({

'name': ['Nazeer','Wahab'],

'id': [10421,12013],

'marks': [84, 93],

})

data[data.marks > 90]

​

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

In [ ]:

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

​

4. What are the columns in the dataframe?

In [ ]:

data.columns

​

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

In [ ]:

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

data.fillna(value = 0)

​

6. Display the top 10 rows

In [ ]:

data.head(10)

​

7. Display the last 10 rows

In [ ]:

data.tail(10)

​

8. Display only the odd rows

In [ ]:

data[1:-1:2]

​

9. Display only those students who got failed in examination

In [ ]:

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

data[data.Grade=="F"]

​

10. Find out the basic statistical info about data

In [ ]:

data.describe()

​

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

In [ ]:

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

​

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

In [ ]:

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

​

**Lab 5: To install OpenCV and study basics of Open CV**

**Lab Tasks:**

1. Consider the following image of model Lena. Load the image using Open CV and show on screen

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

cv2.imshow('Lena', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Create a border around the image

img = cv2.imread('lena.png',1)

constant = cv2.copyMakeBorder(img, 10,10,10,10, cv2.BORDER\_CONSTANT, value=[255,8,8])

cv2.imshow('Lena', constant)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Create a copy of the face and paste it to the top right position

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

face = img[80:167, 70:160]

img[0:87, 0:90] = face

cv2.imshow('Lena', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Consider the following Pepsi logo. Blend it over the Lena’s image

lena = cv2.imread('lena.png')

pepsi = cv2.imread('pepsi.png')

resizedPepsi = cv2.resize(pepsi, (220,220))

dst = cv2.addWeighted(lena, 0.7, resizedPepsi, 0.3, 0)

cv2.imshow('Lena', dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Using bitwise AND, OR and NOT operators, paste the image of Pepsi on Lena’s image. The background of Pepsi logo should not be pasted over, but only ROI will be pasted.

img1 = cv2.imread('lena.png')

img2 = cv2.imread('pepsi.png')

rows,cols,channels = img2.shape

roi = img1[0:rows, 0:cols ]

img2gray = cv2.cvtColor(img2,cv2.COLOR\_BGR2GRAY)

ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH\_BINARY)

mask\_inv = cv2.bitwise\_not(mask)

img1\_bg = cv2.bitwise\_and(roi,roi,mask = mask\_inv)

img2\_fg = cv2.bitwise\_and(img2,img2,mask = mask)

dst = cv2.add(img1\_bg,img2\_fg)

img1[0:rows, 0:cols ] = dst

cv2.imshow('res',img1)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Draw the following Olympic circles using Open CV.

import numpy as np

import cv2

img = np.full((512,512,3), 255, np.uint8)

blue = cv2.circle(img,(140,123), 43, (255,8,8), 3)

black = cv2.circle(img,(240,123), 43, (0,0,0), 3)

red = cv2.circle(img,(340,123), 43, (2,0,253), 3)

yellow = cv2.circle(img,(187,163), 43, (24,252,242), 3)

green = cv2.circle(img,(290,163), 43, (9,109,61), 3)

cv2.imshow('olympic',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Lab Tasks:**

**Lab 6: To study and implement basic image processing operations in OpenCV**

* Consider the image provided in previous lab. Transform the image as follows: resize to twice of the original size, translated 30 pixels horizontally and 50 pixels vertically, rotated by 45o clockwise

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

resizedTwice = cv2.resize(img, (img.shape[1]\*2, img.shape[0]\*2))

rows1,cols1 = resizedTwice.shape

mask = np.float32([[1,0,30],[0,1,50]])

translatedImg = cv2.warpAffine(resizedTwice,mask,(cols1,rows1))

rows2, cols2 = translatedImg.shape

M = cv2.getRotationMatrix2D((cols2/2,rows2/2),45,1)

dst = cv2.warpAffine(translatedImg,M,(cols2,rows2))

cv2.imshow('Lena', dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Perform the following thresholding on the image: cv2.THRESH\_BINARY, cv2.THRESH\_BINARY\_INV, cv2.THRESH\_TRUNC, cv2.THRESH\_TOZERO, cv2.THRESH\_TOZERO\_INV. Provide your narration on the behavior of various types of thresholding

img = cv2.imread('abstracted.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)

cv2.imshow('threshold1', thresh1)

cv2.imshow('threshold2', thresh2)

cv2.imshow('threshold3', thresh3)

cv2.imshow('threshold4', thresh4)

cv2.imshow('threshold5', thresh5)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Now apply adaptive thresholding and Otsu binarization. Do you see any improvement in the result? Explain your answer with proper reason.

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

img = cv2.medianBlur(img,5)

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)

cv2.imshow('threshold2', th2)

cv2.imshow('threshold3', th3)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Apply various filters such as averaging, Gaussian and median. Which one gives better result? Explain your answer with proper reason

#Averaging

img = cv2.imread('noisyLena.png', 0)

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

cv2.imshow('Averaging', avg)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Gaussian Blurring

img = cv2.imread('noisyLena.png', 0)

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

cv2.imshow('Gaussian', gaussian)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Median Blurring

img = cv2.imread('noisyLena.png', 0)

median = cv2.medianBlur(img,5)

cv2.imshow('Median', median)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Bilateral Filtering

img = cv2.imread('noisyLena.png', 0)

bilateral = cv2.bilateralFilter(img,9,75,75)

cv2.imshow('Bilateral', bilateral)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Applying erosion, dilation, opening and closing on the image. Explain the behavior of the operators

#Erosian

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

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

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

cv2.imshow('Morphological Transformation', erosion)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Dilation

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

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

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

cv2.imshow('Morphological Transformation', dilation)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Opening

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

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

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

cv2.imshow('Morphological Transformation', opening)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Closing

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

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

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

cv2.imshow('Morphological Transformation', closing)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Apply different derivative operators such as Sobel, Laplacian and Canny edge detection on the image.

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

#Laplacian

laplacian = cv2.Laplacian(img,cv2.CV\_8U)

cv2.imshow('laplacian', laplacian)

cv2.waitKey(0)

cv2.destroyAllWindows()

#SobelX

sobelx = cv2.Sobel(img,cv2.CV\_8U,1,0,ksize=3)

cv2.imshow('SobelX', sobelx)

cv2.waitKey(0)

cv2.destroyAllWindows()

#SobelY

sobely = cv2.Sobel(img,cv2.CV\_8U,0,1,ksize=3)

cv2.imshow('SobelY', sobely)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Canny Detection

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

cv2.imshow('Canny', canny)

cv2.waitKey(0)

cv2.destroyAllWindows()

Lab Tasks:

**Lab 7: To study and implement advanced image processing operations in OpenCV**

* Calculate the Fourier transform of the image and plot

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

dft = cv2.dft(np.float32(img),flags = cv2.DFT\_COMPLEX\_OUTPUT)

dft\_shift = np.fft.fftshift(dft)

magnitude\_spectrum = 20\*np.log(cv2.magnitude(dft\_shift[:,:,0],dft\_shift[:,:,1]))

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

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

plt.show()

* Now apply Laplacian, Gaussian and Sobel operator on the image. Now calculate the Fourier transform of the processed image and plot it. What behavior do you see?

#Fourier on Laplacian

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

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

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

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

plt.show()

dft = cv2.dft(np.float32(laplacian),flags = cv2.DFT\_COMPLEX\_OUTPUT)

dft\_shift = np.fft.fftshift(dft)

magnitude\_spectrum = 20\*np.log(cv2.magnitude(dft\_shift[:,:,0],dft\_shift[:,:,1]))

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

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

plt.show()

#Fourier on SobelX

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

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

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

plt.show()

dft = cv2.dft(np.float32(sobelx),flags = cv2.DFT\_COMPLEX\_OUTPUT)

dft\_shift = np.fft.fftshift(dft)

magnitude\_spectrum = 20\*np.log(cv2.magnitude(dft\_shift[:,:,0],dft\_shift[:,:,1]))

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

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

plt.show()

#Fourier on SobelY

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

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

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

plt.show()

dft = cv2.dft(np.float32(sobely),flags = cv2.DFT\_COMPLEX\_OUTPUT)

dft\_shift = np.fft.fftshift(dft)

magnitude\_spectrum = 20\*np.log(cv2.magnitude(dft\_shift[:,:,0],dft\_shift[:,:,1]))

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

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

plt.show()

* Find the position of head of the player in the image using the template (Template Matching):

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

template = cv2.imread('face.png',0)

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

# Apply template Matching

res = cv2.matchTemplate(img,template,cv2.TM\_SQDIFF)

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

# If the method is TM\_SQDIFF or TM\_SQDIFF\_NORMED, take minimum

top\_left = min\_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.show()

* Find the histogram of the image

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

plt.hist(img.ravel(),256,[0,256])

plt.show()

* Apply histogram equalization on the input image. Explain the behavior of the equalization operation.

#equ = cv2.equalizeHist(img)

#cv2.imshow('Equalization',equ)

#cv2.waitKey(0)

#cv2.destroyAllWindows()

**Lab Tasks:**

**Lab 8: To study and implement machine learning algorithm using OpenCV**

**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)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)

**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)

**Lab Tasks:**

**Lab 9: To study and implement convolutional neural network using OpenCV**

import numpy as np

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D

from keras.utils import np\_utils

import matplotlib.pyplot as plt

from keras import backend as K

K.set\_image\_dim\_ordering('th')

# fix random seed for reproducibility

np.random.seed(7)

(x\_train, y\_train), (x\_test,y\_test) = mnist.load\_data()

#plt.imshow(x\_train[5])

#print(y\_train[5])

# just to make large dataset to small

x\_train = x\_train[:10000]

y\_train = y\_train[:10000]

x\_test = x\_test[:2000]

y\_test = y\_test[:2000]

x\_train = x\_train.reshape(x\_train.shape[0],1,28,28).astype('float32')

x\_test = x\_test.reshape(x\_test.shape[0],1,28,28).astype('float32')

# normalize inputs from 0-255 to 0-1

x\_train = x\_train / 255

x\_test = x\_test / 255

#one hot encoded

y\_train = np\_utils.to\_categorical(y\_train) # np.unique

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

def baseline\_model():

model = Sequential()

model.add(Conv2D(30, (5,5), input\_shape=(1,28,28), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2))) # picks max value in 2x2 region of feature map

model.add(Conv2D(15, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2)) # randomly exclude 20% of neurons in the layer in order to reduce overfitting.

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(50, activation='relu'))

model.add(Dense(num\_classes, activation='softmax')) # it will give multiple classes as an output

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

model = baseline\_model()

model.fit(x\_train, y\_train, validation\_split=0.22, epochs=10, batch\_size=200)

scores = model.evaluate(x\_test, y\_test)

print("acc: %.2f%%" % (scores[1]\*100))

import numpy as np

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D

from keras.utils import np\_utils

from keras.preprocessing.image import ImageDataGenerator

from keras.preprocessing import image

import matplotlib.pyplot as plt

from keras import backend as K

K.set\_image\_dim\_ordering('th')

#Data augmentation takes the approach of generating more training data

#from existing training samples, by augmenting the samples via a number of random

#transformations. This helps model to generalize better

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')

img = image.load\_img('cat.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):

plt.figure(i)

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

i += 1

if i % 4 == 0:

break

plt.show()