**FACE DETECTION**

**Import Libraries**

import cv2

import matplotlib.pyplot as plt

**Read the Image and display shape**

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people1.jpg')

display (image.shape)

**Display the image with Matplotlib**

plt.imshow(image)

plt.show()

**Print Image Values – Pixel Values**

print(image)

**Display the image with CV2**

The image will be opened in a new window

The image will open and close quickly, it is necessary to add wait key

Once the wait is added the image will be opened till you press a key

cv2.imshow('', image)

cv2.waitKey(0)

**Resize the image**

**The image pixel size is 1280 X 1920. Since this is a**

**big image use resizes to reshape the image 800X600**

image = cv2.resize(image, (800, 600))

image.shape

**Display the image – small size image**

cv2.imshow('',image)

cv2.waitKey(0)

**Gray scale the image – Convert to Back and White**

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cv2.imshow('',image\_gray)

cv2.waitKey(0)

**Display the Size of the Grayscale image (Back and White)**

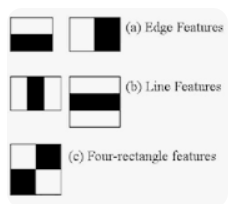
**Once dimension has reduced – No colour dimension**

display (image\_gray.shape)

Create the Model CascadeClassifier

**cascade classifier** is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Cascading classifiers are **trained with several hundred "positive" sample views of a particular object and arbitrary "negative" images of the same size**. After the classifier is trained it can be applied to a region of an image and detect the object in question.



**haarcascade\_frontalface\_default. xml** : **Detects faces**.

haarcascade\_eye. xml Detects the left and right eyes on the face.

**Load Face Detector .xml file**

face\_detector = cv2.CascadeClassifier('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Cascades\\haarcascade\_frontalface\_default.xml')

face\_detector

**Display the Face detection co-ordinates**

Each row for each face detected -The values in each row have the co-ordinates such as x,y, width, height.

In this example, the algorithm detected 6 faces, that is 6 rows in the result.

detections = face\_detector.detectMultiScale(image\_gray)

display (detections)

**Display the length- Number of faces detected**

display (len(detections))

**Display each detected face with rectangle**

x=390 # X - Co ordinates

y=323 # Y- Co ordinates

w=56 # Face Width

h=56 # Face Height

# Draw the rectangle with co-ordinate specified above

# (0,255,255) - This is RBG colour for rectangle

# 1 is the thickness of rectangle

cv2.rectangle(image\_gray, (x, y), (x + w, y + h), (0,255,255), 1)

cv2.imshow('', image\_gray)

cv2.waitKey(0)

**Display second detected face with rectangle**

x=115 # X - Co ordinates

y=124 # Y- Co ordinates

w=52 # Face Width

h=52 # Face Height

# Thickness is 5 , more thicker rectangle

cv2.rectangle(image\_gray, (x, y), (x + w, y + h), (0,255,255), 5)

cv2.imshow('', image\_gray)

cv2.waitKey(0)

**Display third detected face with rectangle**

x=475 # X - Co ordinates

y=123 # Y- Co ordinates

w=59 # Face Width

h=59 # Face Height

cv2.rectangle(image\_gray, (x, y), (x + w, y + h), (0,255,255), 5)

cv2.imshow('', image\_gray)

cv2.waitKey(0)

**Display all detected face with rectangle**

for (x, y, w, h) in detections:

#print (x, y, w, h)

cv2.rectangle(image\_gray, (x, y), (x + w, y + h), (0,255,255), 5)

cv2.imshow('', image\_gray)

cv2.waitKey(0)

plt.show()

**Face Detection by Colour image**

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people1.jpg')

display (image.shape)

image = cv2.resize(image, (800, 600)) # Resize image

display (image.shape)

detections = face\_detector.detectMultiScale(image)

display (detections)

**Display Image**

for (x, y, w, h) in detections:

#print (x, y, w, h)

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 5)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Parameter -Scale Factor**

The argument scaleFactor **determines the factor by which the detection window of the classifier is scaled down per detection pass**. A factor of 1.1 corresponds to an increase of 10%. Hence, increasing the scale factor increases performance, as the number of detection passes is reduced

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people1.jpg')

image = cv2.resize(image, (800, 600)) # Resize image

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = face\_detector.detectMultiScale(image\_gray, scaleFactor = 1.09)

display (detections)

for (x, y, w, h) in detections:

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 5)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Parameter –** **minNeighbors**

Load the image and display it

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people2.jpg')

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

Face Detections in the normal way – It has many False Positive

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people2.jpg')

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = face\_detector.detectMultiScale(image\_gray, scaleFactor = 1.09)

display (detections)

for (x, y, w, h) in detections:

print(w, h)

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Use Parameter –** **minNeighbors – It reduces False Positive**

minNeighbors – **Parameter specifying how many neighbors each candidate rectangle should have to retain it**. In other words, this parameter will affect the quality of the detected faces. Higher value results in less detections but with higher quality

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people2.jpg')

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = face\_detector.detectMultiScale(image\_gray, scaleFactor = 1.2,minNeighbors=7)

display (detections)

for (x, y, w, h) in detections:

print(w, h)

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Use Parameter –** **minSize and  maxSize**

1. **minSize** : Minimum possible object size. Objects smaller than that are ignored. That is the rectangle size.
2. **maxSize** : Maximum possible object size. Objects larger than that are ignored.

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people2.jpg')

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = face\_detector.detectMultiScale(image\_gray, scaleFactor = 1.2,

minNeighbors=7,minSize=(20,20), maxSize=(100,100))

display (detections)

for (x, y, w, h) in detections:

print(w, h)

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Eye Detector**

**Load eye detector .xml file**

eye\_detector= cv2.CascadeClassifier('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Cascades\\haarcascade\_eye.xml')

display (eye\_detector)

**Use both eye detection and Face detection**

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people1.jpg')

display (image.shape)

image = cv2.resize(image, (1600,1000)) # Resize image

print(image.shape)

face\_detections = face\_detector.detectMultiScale(image, scaleFactor = 1.3, minSize = (30,30))

for (x, y, w, h) in face\_detections:

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

eye\_detections = eye\_detector.detectMultiScale(image, scaleFactor = 1.1, minNeighbors=10, maxSize=(60,60))

for (x, y, w, h) in eye\_detections:

print(w, h)

cv2.rectangle(image, (x, y), (x + w, y + h), (0,0,255), 2)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Car Detector**

**Load .xml File**

car\_detector = cv2.CascadeClassifier('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Cascades\\cars.xml')

display (car\_detector)

**Load the image and display the shape**

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\car.jpg')

display (image.shape)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Car Detector**

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = car\_detector.detectMultiScale(image\_gray, scaleFactor = 1.03, minNeighbors=8)

for (x, y, w, h) in detections:

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Clock Detector**

**Load .xml File**

clock\_detector = cv2.CascadeClassifier('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Cascades\\clocks.xml')

display (clock\_detector)

**Load image**

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\clock.jpg')

display (image.shape)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Clock Detector**

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = clock\_detector.detectMultiScale(image\_gray, scaleFactor = 1.03, minNeighbors=1)

for (x, y, w, h) in detections:

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Full Body Detector**

**Load .xml File**

full\_detector = cv2.CascadeClassifier('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Cascades\\fullbody.xml')

display (full\_detector)

**Load image**

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people3.jpg')

display (image.shape)

cv2.imshow('',image)

cv2.waitKey(0)

plt.show()

**Body Detector**

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

detections = full\_detector.detectMultiScale(image\_gray, scaleFactor = 1.05, minNeighbors=5,

minSize = (50,50))

for (x, y, w, h) in detections:

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

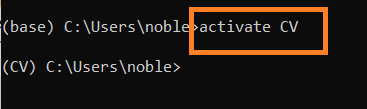
plt.show()

Face detection with HOG and Dlib

Install dlib

Activate new environment – CV

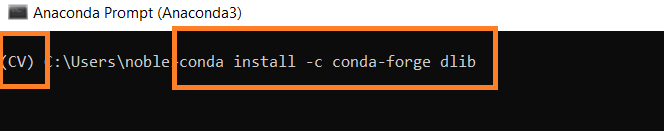
activate CV



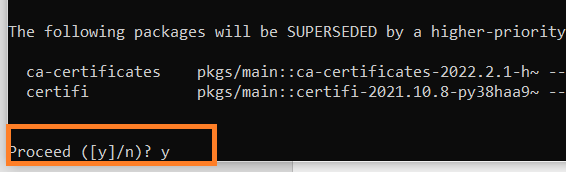
Install DLIB

Type in the command prompt of CV

conda install -c conda-forge dlib



Proceed Y/N - > Type Y



After install import DLIB in Jupyter Notebook



Import Library

import dlib

Load Image

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people2.jpg')

cv2.imshow('',image)

cv2.waitKey(0)

Create the Model

face\_detector\_hog = dlib.get\_frontal\_face\_detector()

detections = face\_detector\_hog(image, 1) # 1 – is the scale factor

Prediction / Detections Create the Model

display (len(detections))

display (detections)

Mark the face in one image

print(detections[0])

print(detections[0].left())

print(detections[0].top())

print(detections[0].right())

print(detections[0].bottom())

cv2.rectangle(image, (detections[0].left(), detections[0].top()), (detections[0].right(), detections[0].bottom()), (0, 255, 255), 2)

cv2.imshow('',image)

cv2.waitKey(0)

For loop to mark face in all images

for face in detections:

l, t, r, b = face.left(), face.top(), face.right(), face.bottom()

cv2.rectangle(image, (l, t), (r, b), (0, 255, 255), 2)

cv2.imshow('',image)

cv2.waitKey(0)

Face detection with CNN and Dlib

Load Image

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people2.jpg')

cv2.imshow('',image)

cv2.waitKey(0)

Open the Detector

cnn\_detector = dlib.cnn\_face\_detection\_model\_v1('C:\\Noble\\Training\\Course Content\\Computer Vision\\Weights\\mmod\_human\_face\_detector.dat')

cnn\_detector

Create the model and Mark the image

# It may take more time to display the result

detections = cnn\_detector(image, 1)

for face in detections:

l, t, r, b, c = face.rect.left(), face.rect.top(), face.rect.right(), face.rect.bottom(), face.confidence

print(c)

cv2.rectangle(image, (l, t), (r, b), (255, 255, 0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

Example with Haarcascade , HOG ,CNN

Note: CNN code is not mentioned below, due to performance challenges

Haarcascade

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people3.jpg')

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

haarcascade\_detector = cv2.CascadeClassifier('C:\\Noble\\Training\\Course Content\\Computer Vision\\Cascades\\haarcascade\_frontalface\_default.xml')

detections = haarcascade\_detector.detectMultiScale(image\_gray, scaleFactor = 1.001, minNeighbors=5, minSize = (5,5))

for (x, y, w, h) in detections:

cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

cv2.imshow('',image)

cv2.waitKey(0)

HOG

image = cv2.imread('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Face Detection\\Data Set\\people3.jpg')

face\_detector\_hog = dlib.get\_frontal\_face\_detector()

detections = face\_detector\_hog(image, 4)

for face in detections:

l, t, r, b = (face.left(), face.top(), face.right(), face.bottom())

cv2.rectangle(image, (l, t), (r, b), (0, 255, 255), 2)

cv2.imshow('',image)

cv2.waitKey(0)