**CODING:**

**FOR DETECTING OBJECTS IN VIDEO**

import numpy as np

import cv2

confidenceThreshold = 0.5

NMSThreshold = 0.3

modelConfiguration = 'cfg/rcnn.cfg'

modelWeights = 'rcnn.weights'

labelsPath = 'coco.names'

labels = open(labelsPath).read().strip().split('\n')

np.random.seed(10)

COLORS = np.random.randint(0, 255, size=(len(labels), 3), dtype="uint8")

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)

outputLayer = net.getLayerNames()

outputLayer = [outputLayer[i[0] - 1] for i in net.getUnconnectedOutLayers()]

video = cv2.VideoCapture('video.mp4')

writer = None

(W, H) = (None, None)

try:

prop = cv2.CAP\_PROP\_FRAME\_COUNT

total = int(video.get(prop))

print("[INFO] {} total frames in video".format(total))

except:

printf("Could not determine no. of frames in video")

count = 0

while True:

(ret, frame) = video.read()

if not ret:

break

if W is None or H is None:

(H,W) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB = True, crop = False)

net.setInput(blob)

layersOutputs = net.forward(outputLayer)

boxes = []

confidences = []

classIDs = []

for output in layersOutputs:

for detection in output:

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

if confidence > confidenceThreshold:

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype('int')

x = int(centerX - (width/2))

y = int(centerY - (height/2))

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

#Apply Non Maxima Suppression

detectionNMS = cv2.dnn.NMSBoxes(boxes, confidences, confidenceThreshold, NMSThreshold)

if(len(detectionNMS) > 0):

for i in detectionNMS.flatten():

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

text = '{}: {:.4f}'.format(labels[classIDs[i]], confidences[i])

cv2.putText(frame, text, (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

if writer is None:

fourcc = cv2.VideoWriter\_fourcc(\*'MJPG')

writer = cv2.VideoWriter('video\_output.avi', fourcc, 30, (frame.shape[1], frame.shape[0]), True)

if writer is not None:

writer.write(frame)

print("Writing frame" , count+1)

count = count + 1

writer.release()

video.release()

**FOR DETECTING OBJECTS AN IMAGE**

import numpy as np

import cv2

confidenceThreshold = 0.5

NMSThreshold = 0.3

modelConfiguration = 'cfg/rcnn.cfg'

modelWeights = 'rcnn.weights'

labelsPath = 'coco.names'

labels = open(labelsPath).read().strip().split('\n')

np.random.seed(10)

COLORS = np.random.randint(0, 255, size=(len(labels), 3), dtype="uint8")

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)

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

(H, W) = image.shape[:2]

#Determine output layer names

layerName = net.getLayerNames()

layerName = [layerName[i[0] - 1] for i in net.getUnconnectedOutLayers()]

blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB = True, crop = False)

net.setInput(blob)

layersOutputs = net.forward(layerName)

boxes = []

confidences = []

classIDs = []

for output in layersOutputs:

for detection in output:

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

if confidence > confidenceThreshold:

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype('int')

x = int(centerX - (width/2))

y = int(centerY - (height/2))

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

#Apply Non Maxima Suppression

detectionNMS = cv2.dnn.NMSBoxes(boxes, confidences, confidenceThreshold, NMSThreshold)

if(len(detectionNMS) > 0):

for i in detectionNMS.flatten():

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

color = [int(c) for c in COLORS[classIDs[i]]]

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

text = '{}: {:.4f}'.format(labels[classIDs[i]], confidences[i])

cv2.putText(image, text, (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

cv2.imshow('Image', image)

cv2.waitKey(0)

**FOR DETECTING OBJECTS IN WEBCAM**

import numpy as np

import cv2

confidenceThreshold = 0.5

NMSThreshold = 0.3

modelConfiguration = 'cfg/rcnn.cfg'

modelWeights = 'rcnn.weights'

labelsPath = 'coco.names'

labels = open(labelsPath).read().strip().split('\n')

np.random.seed(10)

COLORS = np.random.randint(0, 255, size=(len(labels), 3), dtype="uint8")

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)

outputLayer = net.getLayerNames()

outputLayer = [outputLayer[i[0] - 1] for i in net.getUnconnectedOutLayers()]

video\_capture = cv2.VideoCapture(0)

(W, H) = (None, None)

while True:

ret, frame = video\_capture.read()

frame = cv2.flip(frame, 1)

if W is None or H is None:

(H,W) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB = True, crop = False)

net.setInput(blob)

layersOutputs = net.forward(outputLayer)

boxes = []

confidences = []

classIDs = []

for output in layersOutputs:

for detection in output:

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

if confidence > confidenceThreshold:

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype('int')

x = int(centerX - (width/2))

y = int(centerY - (height/2))

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

#Apply Non Maxima Suppression

detectionNMS = cv2.dnn.NMSBoxes(boxes, confidences, confidenceThreshold, NMSThreshold)

if(len(detectionNMS) > 0):

for i in detectionNMS.flatten():

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

text = '{}: {:.4f}'.format(labels[classIDs[i]], confidences[i])

cv2.putText(frame, text, (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

cv2.imshow('Output', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

#Finally when video capture is over, release the video capture and destroyAllWindows

video\_capture.release()

cv2.destroyAllWindows()