**İMAGE.PY**

import cv2

import numpy as np

from matplotlib import pyplot as plt

import glob, os

model = cv2.dnn.readNetFromDarknet("yolov4-tiny-obj.cfg","yolov4-tiny-obj\_last.weights")

layers = model.getLayerNames()

unconnect = model.getUnconnectedOutLayers()

unconnect = unconnect-1

output\_layers = []

for i in unconnect:

output\_layers.append(layers[int(i)])

classFile = 'obj.names'

classNames=[]

with open(classFile,'rt') as f:

classNames = f.read().rstrip('\n').split('\n')

print(classNames)

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

img\_width = img.shape[1]

img\_height = img.shape[0]

img\_blob = cv2.dnn.blobFromImage(img,1/255,(416,416),swapRB=True)

model.setInput(img\_blob)

detection\_layers = model.forward(output\_layers)

ids\_list = []

boxes\_list = []

confidences\_list = []

for detection\_layer in detection\_layers:

for object\_detection in detection\_layer:

scores = object\_detection[5:]

predicted\_id = np.argmax(scores)

confidence =scores[predicted\_id]

if confidence > 0.10:

label = classNames[predicted\_id]

bounding\_box = object\_detection[0:4] \* np.array([img\_width,img\_height,img\_width,img\_height])

(box\_center\_x, box\_center\_y ,box\_width ,box\_height) = bounding\_box.astype("int")

start\_x = int(box\_center\_x- (box\_width/2))

start\_y = int(box\_center\_y - (box\_height/2))

ids\_list.append(predicted\_id)

confidences\_list.append(float(confidence))

boxes\_list.append([start\_x,start\_y,int(box\_width),int(box\_height)])

max\_ids = cv2.dnn.NMSBoxes(boxes\_list,confidences\_list,0.5,0.4)

for max\_id in max\_ids:

max\_class\_id=max\_id

box = boxes\_list[max\_class\_id]

start\_x = box[0]

start\_y = box[1]

box\_width = box[2]

box\_height= box[3]

predicted\_id = ids\_list[max\_class\_id]

label = classNames[predicted\_id]

print(classNames[predicted\_id])

confidence=confidences\_list[max\_class\_id]

end\_x = start\_x + box\_width

end\_y = start\_y+box\_height

cv2.rectangle(img,(start\_x,start\_y),(end\_x,end\_y),(255, 0, 0),2)

cv2.putText(img,label,(start\_x,start\_y-20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255, 0, 0),1,1)

cv2.imshow("img",img)

cv2.waitKey(0)

**REAL-TİME.PY**

import cv2

import numpy as np

from matplotlib import pyplot as plt

import glob, os

model = cv2.dnn.readNetFromDarknet("yolov4-tiny-obj.cfg","yolov4-tiny-obj\_last.weights")

layers = model.getLayerNames()

unconnect = model.getUnconnectedOutLayers()

unconnect = unconnect-1

output\_layers = []

for i in unconnect:

output\_layers.append(layers[int(i)])

classFile = 'obj.names'

classNames=[]

with open(classFile,'rt') as f:

classNames = f.read().rstrip('\n').split('\n')

print(classNames)

vid = cv2.VideoCapture(0)

while(True):

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

break

ret, frame = vid.read()

img\_width = frame.shape[1]

img\_height = frame.shape[0]

img\_blob = cv2.dnn.blobFromImage(frame,1/255,(416,416),swapRB=True)

model.setInput(img\_blob)

detection\_layers = model.forward(output\_layers)

ids\_list = []

boxes\_list = []

confidences\_list = []

for detection\_layer in detection\_layers:

for object\_detection in detection\_layer:

scores = object\_detection[5:]

predicted\_id = np.argmax(scores)

confidence =scores[predicted\_id]

if confidence > 0.10:

label = classNames[predicted\_id]

bounding\_box = object\_detection[0:4] \* np.array([img\_width,img\_height,img\_width,img\_height])

(box\_center\_x, box\_center\_y ,box\_width ,box\_height) = bounding\_box.astype("int")

start\_x = int(box\_center\_x- (box\_width/2))

start\_y = int(box\_center\_y - (box\_height/2))

ids\_list.append(predicted\_id)

confidences\_list.append(float(confidence))

boxes\_list.append([start\_x,start\_y,int(box\_width),int(box\_height)])

max\_ids = cv2.dnn.NMSBoxes(boxes\_list,confidences\_list,0.5,0.4)

for max\_id in max\_ids:

max\_class\_id=max\_id

box = boxes\_list[max\_class\_id]

start\_x = box[0]

start\_y = box[1]

box\_width = box[2]

box\_height= box[3]

predicted\_id = ids\_list[max\_class\_id]

label = classNames[predicted\_id]

print(classNames[predicted\_id])

confidence=confidences\_list[max\_class\_id]

end\_x = start\_x + box\_width

end\_y = start\_y+box\_height

cv2.rectangle(frame,(start\_x,start\_y),(end\_x,end\_y),(255, 0, 0),2)

cv2.putText(frame,label,(start\_x,start\_y-20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255, 0, 0),1,1)

cv2.imshow('frame', frame)