Person

bicycle

car

motorcycle

airplane

bus

train

truck

boat

traffic light

fire hydrant

stop sign

parking meter

bench

bird

cat

dog

horse

sheep

cow

elephant

bear

zebra

giraffe

backpack

umbrella

handbag

tie

suitcase

frisbee

skis

snowboard

sports ball

kite

baseball bat

baseball glove

skateboard

surfboard

tennis racket

bottle

wine glass

cup

fork

knife

spoon

bowl

banana

apple

sandwich

orange

broccoli

carrot

hot dog

pizza

donut

cake

chair

couch

potted plant

bed

dining table

toilet

tv

laptop

mouse

remote

keyboard

cell phone

microwave

oven

toaster

sink

refrigerator

book

clock

vase

scissors

teddy bear

hair drier

toothbrush

import cv2

import argparse

import numpy as np

ap = argparse.ArgumentParser()

ap.add\_argument('-i', '--image', required=True,

help = 'path to input image')

ap.add\_argument('-c', '--config', required=True,

help = 'path to yolo config file')

ap.add\_argument('-w', '--weights', required=True,

help = 'path to yolo pre-trained weights')

ap.add\_argument('-cl', '--classes', required=True,

help = 'path to text file containing class names')

args = ap.parse\_args()

def get\_output\_layers(net):

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

return output\_layers

def draw\_prediction(img, class\_id, confidence, x, y, x\_plus\_w, y\_plus\_h):

label = str(classes[class\_id])

color = COLORS[class\_id]

cv2.rectangle(img, (x,y), (x\_plus\_w,y\_plus\_h), color, 2)

cv2.putText(img, label, (x-10,y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

image = cv2.imread(args.image)

Width = image.shape[1]

Height = image.shape[0]

scale = 0.00392

classes = None

with open(args.classes, 'r') as f:

classes = [line.strip() for line in f.readlines()]

COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

net = cv2.dnn.readNet(args.weights, args.config)

blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)

net.setInput(blob)

outs = net.forward(get\_output\_layers(net))

class\_ids = []

confidences = []

boxes = []

conf\_threshold = 0.5

nms\_threshold = 0.4

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.5:

center\_x = int(detection[0] \* Width)

center\_y = int(detection[1] \* Height)

w = int(detection[2] \* Width)

h = int(detection[3] \* Height)

x = center\_x - w / 2

y = center\_y - h / 2

class\_ids.append(class\_id)

confidences.append(float(confidence))

boxes.append([x, y, w, h])

indices = cv2.dnn.NMSBoxes(boxes, confidences, conf\_threshold, nms\_threshold)

for i in indices:

i = i[0]

box = boxes[i]

x = box[0]

y = box[1]

w = box[2]

h = box[3]

draw\_prediction(image, class\_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))

cv2.imshow("object detection", image)

cv2.waitKey()

cv2.imwrite("object-detection.jpg", image)

cv2.destroyAllWindows()

Open Terminal in the directory and follow the command below:-

python yolo\_opencv.py --image PHOTO.jpg --config yolov3.cfg --weights yolov3.weights --classes yolov3.txt

To save the output image press ctrl+s and save the output image.

[net]

# Testing

# batch=1

# subdivisions=1

# Training

batch=64

subdivisions=16

width=608

height=608

channels=3

momentum=0.9

decay=0.0005

angle=0

saturation = 1.5

exposure = 1.5

hue=.1

learning\_rate=0.001

burn\_in=1000

max\_batches = 500200

policy=steps

steps=400000,450000

scales=.1,.1

[convolutional]

batch\_normalize=1

filters=32

size=3

stride=1

pad=1

activation=leaky

# Downsample

[convolutional]

batch\_normalize=1

filters=64

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=32

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=64

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=64

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=64

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

######################

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=255

activation=linear

[yolo]

mask = 6,7,8

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=80

num=9

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[upsample]

stride=2

[route]

layers = -1, 61

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=255

activation=linear

[yolo]

mask = 3,4,5

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=80

num=9

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[upsample]

stride=2

[route]

layers = -1, 36

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=256

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=256

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=256

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=255

activation=linear

[yolo]

mask = 0,1,2

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=80

num=9

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

