第3次Python课程实验报告

姓名：李金旻 学号：2401210317 日期：2025.1.21

# 一、实验名称

第3次实验：YOLOv3

# 二、实验介绍

本次实验旨在了解使用 yolov3 进行对象检测。该项目使用预测统的 yolov3 模型实现了图像和视频对象检测分类器。yolov3 模型取自 2018 年发布的官方 yolov3 论文。yolov3 实现来自 darknet。此外，该项目还实现了使用网络摄像头实时执行分类的选项。。

# 三、创新点

无

# 四、实验代码

import numpy as np

import argparse

import cv2 as cv

import subprocess

import time

import os

from yolo\_utils import infer\_image, show\_image # 导入自定义的工具函数

FLAGS = [] # 用于存储命令行参数

if \_\_name\_\_ == '\_\_main\_\_':

# 创建参数解析器

parser = argparse.ArgumentParser()

# 添加命令行参数

parser.add\_argument('-m', '--model-path',

type=str,

default='./yolov3-coco/', # 模型文件所在的目录

help='The directory where the model weights and configuration files are.')

parser.add\_argument('-w', '--weights',

type=str,

default='./yolov3-coco/yolov3.weights', # YOLOv3 权重文件路径

help='Path to the file which contains the weights for YOLOv3.')

parser.add\_argument('-cfg', '--config',

type=str,

default='./yolov3-coco/yolov3.cfg', # YOLOv3 配置文件路径

help='Path to the configuration file for the YOLOv3 model.')

parser.add\_argument('-i', '--image-path',

type=str,

help='The path to the image file') # 输入图像路径

parser.add\_argument('-v', '--video-path',

type=str,

help='The path to the video file') # 输入视频路径

parser.add\_argument('-vo', '--video-output-path',

type=str,

default='./output.avi', # 输出视频路径

help='The path of the output video file')

parser.add\_argument('-l', '--labels',

type=str,

default='./yolov3-coco/coco-labels', # COCO 数据集标签文件路径

help='Path to the file having the labels in a new-line separated way.')

parser.add\_argument('-c', '--confidence',

type=float,

default=0.5, # 置信度阈值

help='The model will reject boundaries which has a probability less than the confidence value. Default: 0.5')

parser.add\_argument('-th', '--threshold',

type=float,

default=0.3, # 非极大值抑制（NMS）阈值

help='The threshold to use when applying the Non-Max Suppression. Default: 0.3')

parser.add\_argument('--download-model',

type=bool,

default=False, # 是否下载模型文件

help='Set to True, if the model weights and configurations are not present on your local machine.')

parser.add\_argument('-t', '--show-time',

type=bool,

default=False, # 是否显示推理时间

help='Show the time taken to infer each image.')

# 解析命令行参数

FLAGS, unparsed = parser.parse\_known\_args()

# 如果需要下载模型文件，则调用下载脚本

if FLAGS.download\_model:

subprocess.call(['./yolov3-coco/get\_model.sh'])

# 读取标签文件

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

# 为每个标签生成随机颜色，用于绘制检测框

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

# 加载 YOLOv3 模型

net = cv.dnn.readNetFromDarknet(FLAGS.config, FLAGS.weights)

# 获取输出层的名称

layer\_indices = net.getUnconnectedOutLayers() # 获取未连接的输出层的索引

# 处理返回值类型

if isinstance(layer\_indices, int): # 如果返回值是标量（单个值）

layer\_names = [net.getLayerNames()[layer\_indices - 1]] # 获取对应的层名称

else: # 如果返回值是数组

layer\_names = [net.getLayerNames()[i - 1] for i in layer\_indices.flatten()] # 获取所有输出层的名称

# 如果没有提供图像或视频路径，则使用摄像头进行实时检测

if FLAGS.image\_path is None and FLAGS.video\_path is None:

print('Neither path to an image or path to video provided')

print('Starting Inference on Webcam')

# 如果提供了图像路径，则对图像进行推理

if FLAGS.image\_path:

# 读取图像

try:

img = cv.imread(FLAGS.image\_path) # 读取图像文件

height, width = img.shape[:2] # 获取图像的高度和宽度

except:

raise 'Image cannot be loaded!\nPlease check the path provided!' # 如果图像加载失败，抛出异常

finally:

# 对图像进行推理

img, \_, \_, \_, \_ = infer\_image(net, layer\_names, height, width, img, colors, labels, FLAGS)

show\_image(img) # 显示检测结果

# 如果提供了视频路径，则对视频进行推理

elif FLAGS.video\_path:

# 读取视频

try:

vid = cv.VideoCapture(FLAGS.video\_path) # 打开视频文件

height, width = None, None # 初始化视频的高度和宽度

writer = None # 初始化视频写入器

except:

raise 'Video cannot be loaded!\nPlease check the path provided!' # 如果视频加载失败，抛出异常

finally:

while True:

grabbed, frame = vid.read() # 读取视频帧

# 如果视频读取完毕，则退出循环

if not grabbed:

break

# 如果视频的高度和宽度未初始化，则获取第一帧的尺寸

if width is None or height is None:

height, width = frame.shape[:2]

# 对当前帧进行推理

frame, \_, \_, \_, \_ = infer\_image(net, layer\_names, height, width, frame, colors, labels, FLAGS)

# 如果视频写入器未初始化，则创建写入器

if writer is None:

fourcc = cv.VideoWriter\_fourcc(\*"MJPG") # 定义视频编码格式

writer = cv.VideoWriter(FLAGS.video\_output\_path, fourcc, 30,

(frame.shape[1], frame.shape[0]), True) # 创建视频写入器

# 将检测结果写入视频文件

writer.write(frame)

# 释放资源

print("[INFO] Cleaning up...")

writer.release() # 释放视频写入器

vid.release() # 释放视频读取器

# 如果没有提供图像或视频路径，则使用摄像头进行实时检测

else:

count = 0 # 初始化计数器

vid = cv.VideoCapture(0) # 打开摄像头

while True:

\_, frame = vid.read() # 读取摄像头帧

height, width = frame.shape[:2] # 获取帧的高度和宽度

# 对当前帧进行推理

if count == 0:

frame, boxes, confidences, classids, idxs = infer\_image(net, layer\_names,height, width, frame, colors, labels, FLAGS)

count += 1

else:

frame, boxes, confidences, classids, idxs = infer\_image(net, layer\_names,height, width, frame, colors, labels, FLAGS, boxes, confidences, classids, idxs, infer=False)

count = (count + 1) % 6

# 显示检测结果

cv.imshow('webcam', frame)

# 按下 'e' 键退出

if cv.waitKey(1) & 0xFF == ord('e'):

break

# 释放资源

vid.release() # 释放摄像头

cv.destroyAllWindows() # 关闭所有窗口

# 运行结果

图片一：识别出了显示器和键盘



# 图片二：识别出了手机



# 参考

1. 实验来源：https://github.com/dillipnair/YOLOv3-Object-Detection-with-OpenCV-master
2. 项目介绍：https://pylmagesearch.com/2018/11/12/yolo-object-detection-with-opencv/