基于 YOLOv5 模型的图像目标检测

实验指导书

一、实验目的

本实验旨在使用 YOLOv5 目标检测模型进行图像中的目标检测。YOLOv5 是一种流行的目标检测模型，采用了单阶段（one-stage）检测方法，具有极高的检测速度和良好的检测性能。

（1）理解 YOLOv5 模型的基本原理和工作机制。

（2）掌握 YOLOv5 模型的部署和使用方法。

（3）进行基于 YOLOv5 模型的目标检测实验，包括模型训练、推理和评估。

二、实验要求

（1）基于 Python 语言和任意一种深度学习框架（实验指导书中使用 PyTorch 框架进行介绍），从零开始完成数据读取、网络构建、模型训练和模型测试等过程，最终实现一个可以完成基于 YOLOv5 模型的图像目标检测的程序。

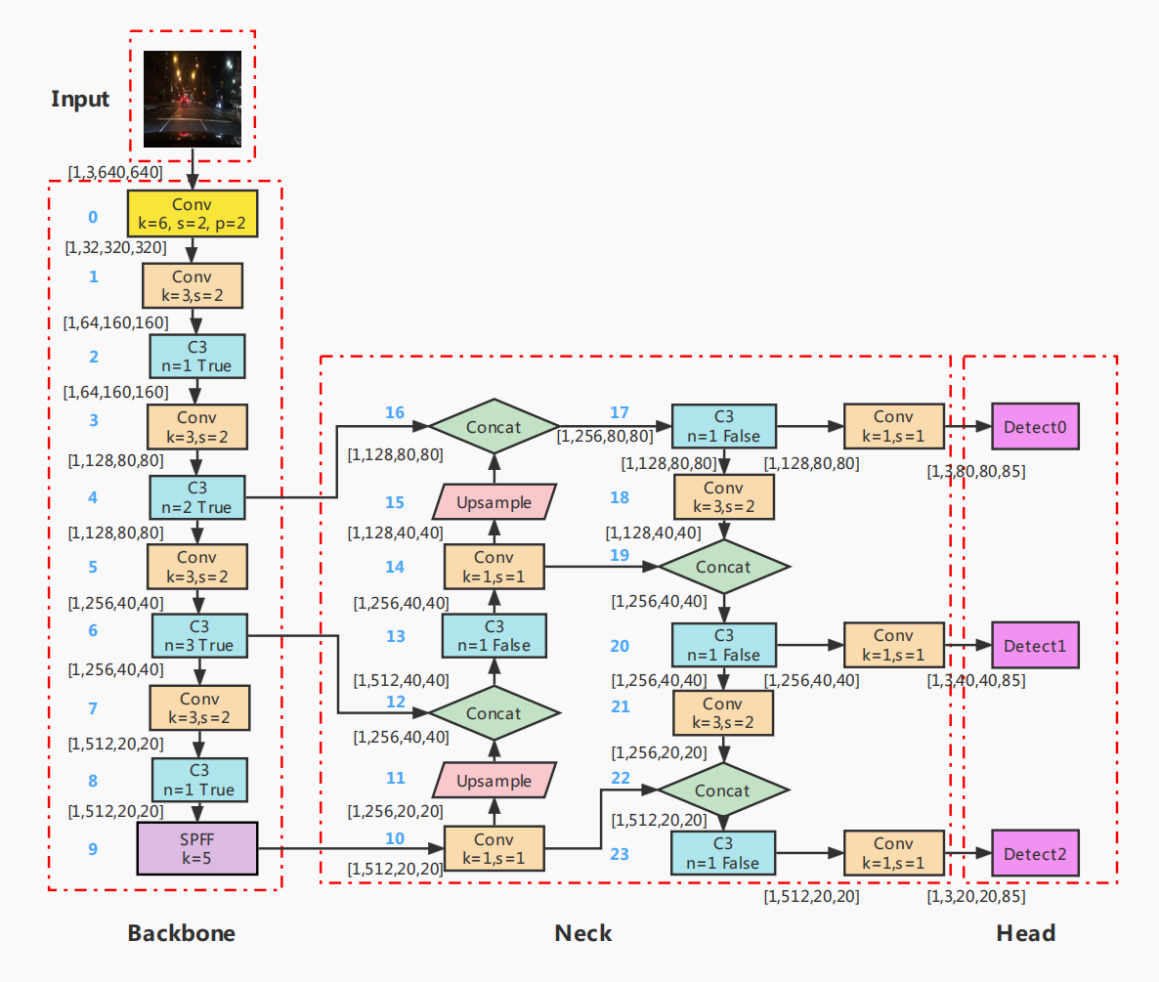
（2）在自定义数据集上进行训练和评估，实现测试集检测精度达到 90%以上。

（3）在规定时间内提交实验报告、代码和 PPT。

三、YOLOv5 模型原理

YOLOv5 主要分为输入端，backbone、Neck 和 head(prediction)。backbone 是 New CSP-Darknet53。Neck 层为 SPFF 和 New CSP-PAN。Head 层为 YOLOv3 head。

YOLOv5 6.0 版本的主要架构如下图所示：



四、实验所需工具

在开始实验之前，请确保已经完成以下准备工作：

（1）安装 Python 虚拟环境（建议使用 Anaconda）。

（2）安装 PyTorch 和 torchvision 库。

（3）下载 YOLOv5 源代码包。

五、实验步骤

（以下步骤为 YOLOv5 安装配置及训练推理的详细步骤）。

注意：在运行 YOLOv5 前，先要安装好 Anaconda 环境，具体操作可参考下述 Anaconda3 的安装配置及使用教程（详细过程）：

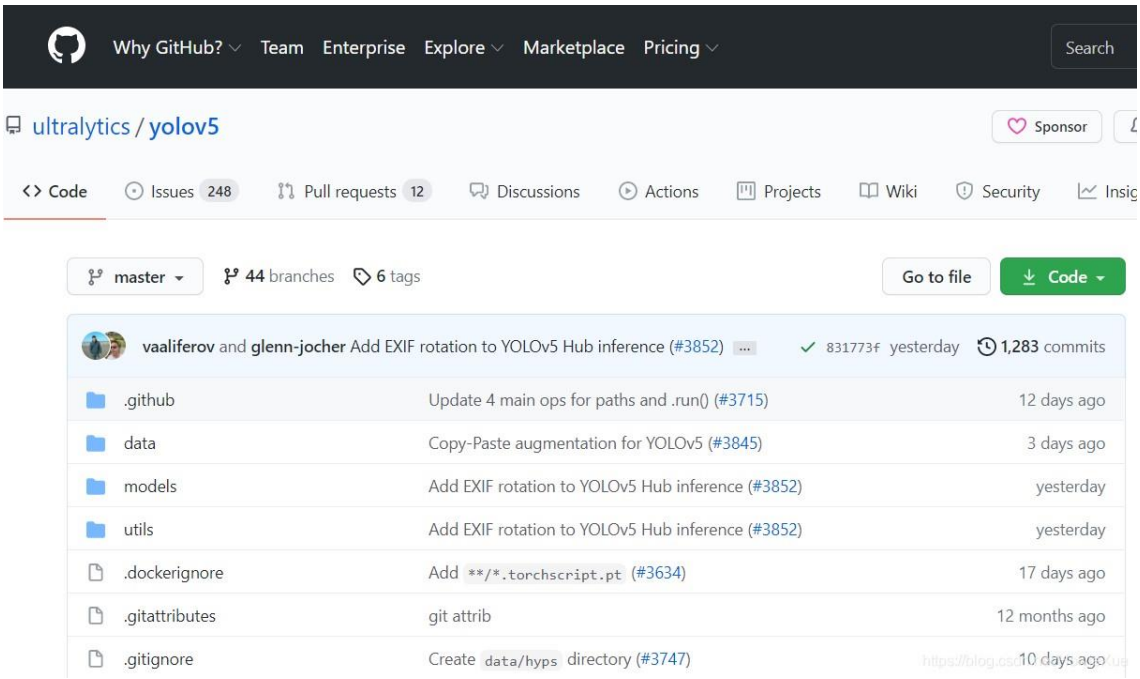
https://howiexue.blog.csdn.net/article/details/118442904。

YOLOv5 官方指南：https://docs.ultralytics.com/quick-start/。

5.1. 下载 YOLOv5

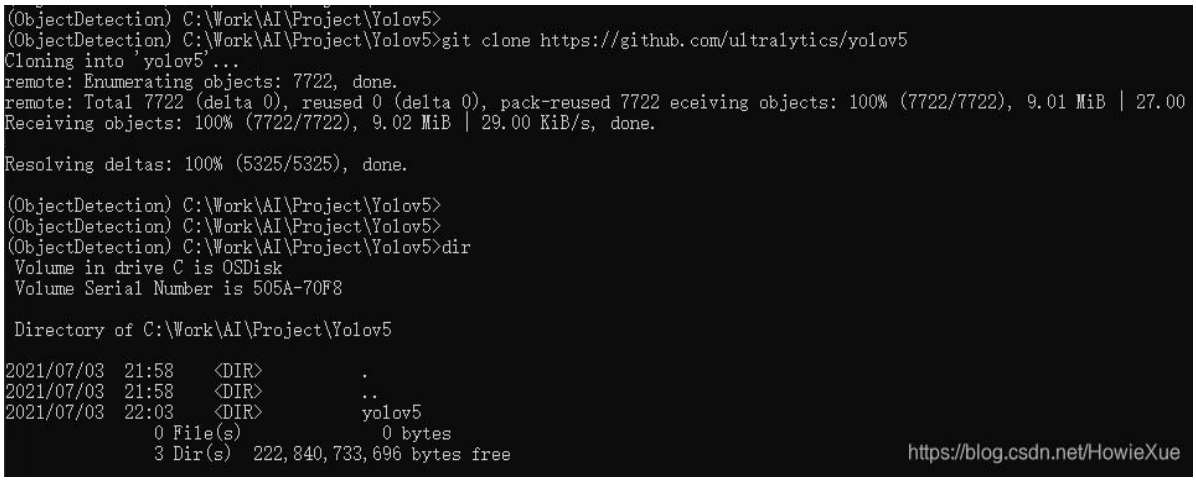
5.1.1 下载 YOLOv5 源码

Github 地址：https://github.com/ultralytics/YOLOv5

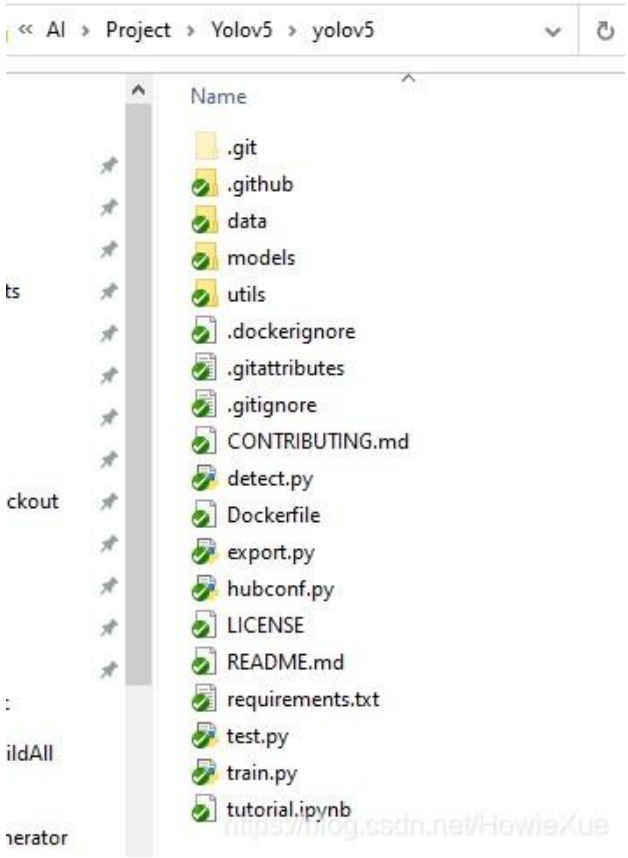


命令行 git clone 到本地工作目录，等待下载完成：

git clone https://github.com/ultralytics/YOLOv5

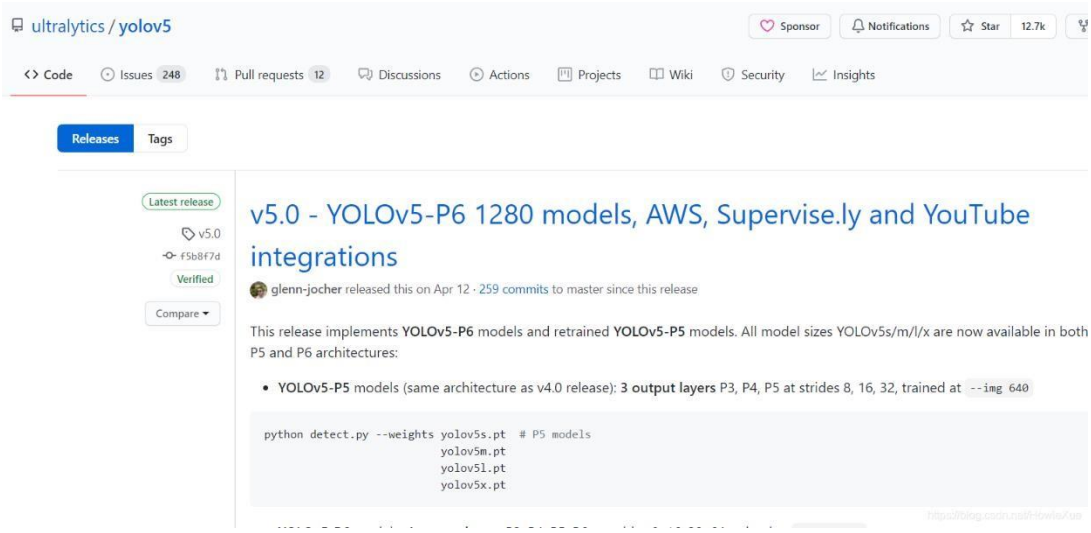


YOLOv5 代码目录架构：



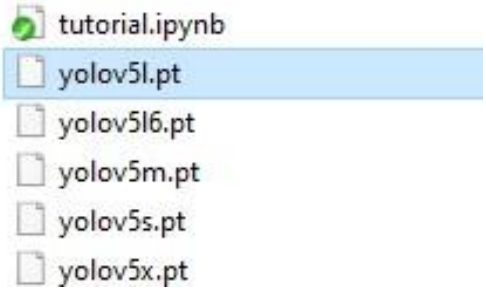
5.1.2 下载 YOLOv5 预训练模型

可以直接采用已经初始训练好的权重，不需要在本机再做训练。下载地址：https://github.com/ultralytics/YOLOv5/releases。

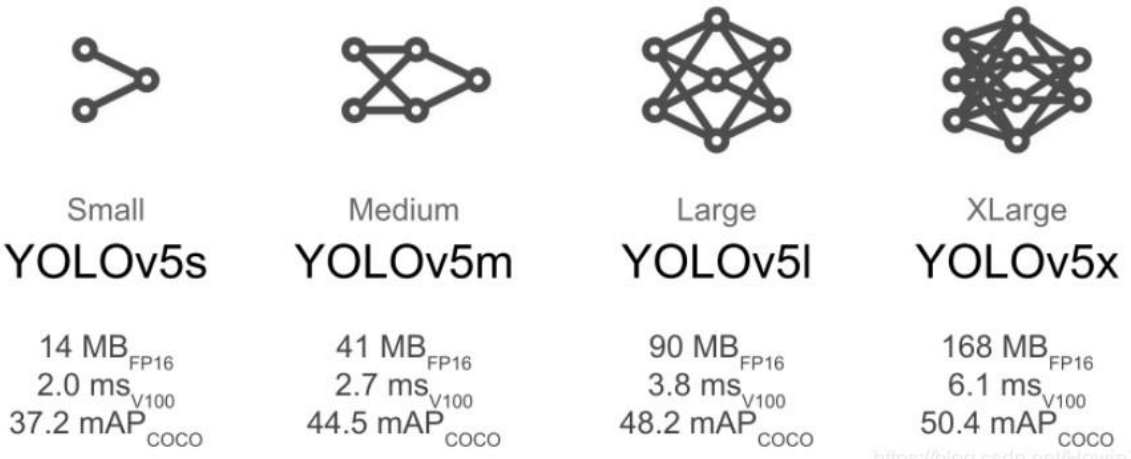


找到最新的 release，点开下面的 Assets 下载（.pt 文件就是模型的权重文件）。

下载后放到 YOLOv5 源码根目录，或新建一个 weights/目录中使用：



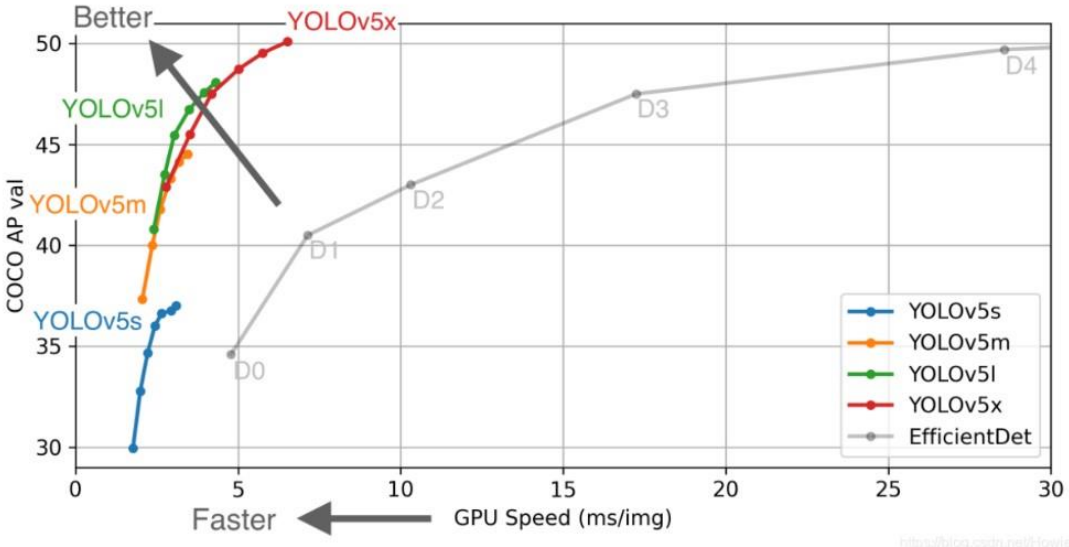
YOLOv5 共有四种预训练模型权重：YOLOv5s、YOLOv5m、YOLOv5l 和 YOLOv5x。使用不同的预训练模型，效果和精度不一样，如下图所示：



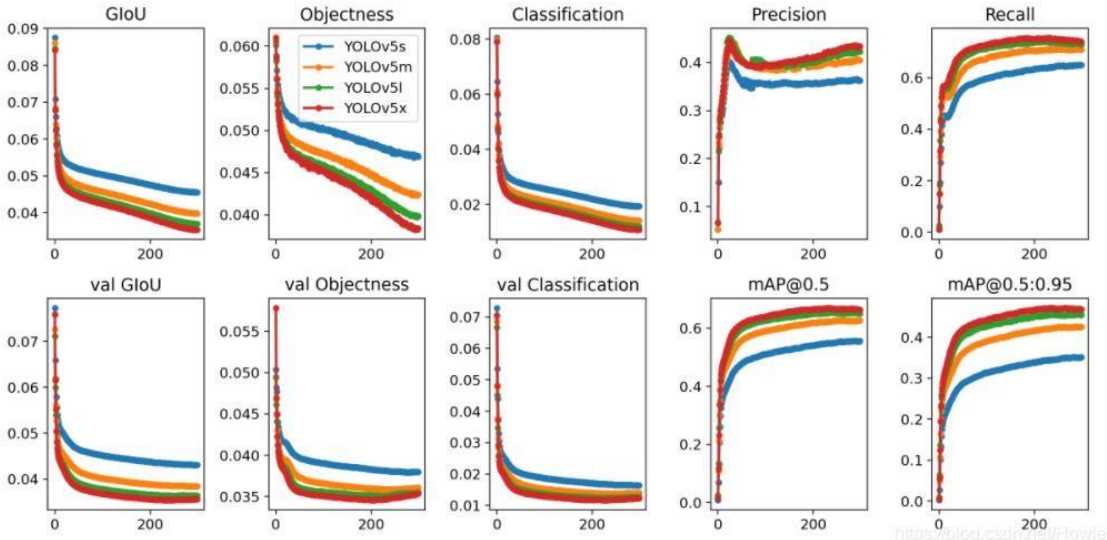
其中，YOLOv5s 目标检测速度最快，因为其网络参数最少，但相应的，检测效果相比是最差的。而 YOLOv5x 是检测效果最好的，参数最多，但是运行时间上最慢。可以根据实际需要，例如目标检测场景比较看重速度，就用 YOLOv5s.pt。

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{test} 0.5:0.95	mAP^{val} 0.5	Speed V100 (ms)	params (M)	FLOPs 640 (B)
YOLOv5s	640	36.7	36.7	55.4	**2.0**	7.3	17.0
YOLOv5m	640	44.5	44.5	63.1	2.7	21.4	51.3
YOLOv5l	640	48.2	48.2	66.9	3.8	47.0	115.4
YOLOv5x	640	**50.4**	**50.4**	**68.8**	6.1	87.7	218.8
YOLOv5s6	1280	43.3	43.3	61.9	**4.3**	12.7	17.4
YOLOv5m6	1280	50.5	50.5	68.7	8.4	35.9	52.4
YOLOv5l6	1280	53.4	53.4	71.1	12.3	77.2	117.7
YOLOv5x6	1280	**54.4**	**54.4**	**72.0**	22.4	141.8	222.9
YOLOv5x6 TTA	1280	**55.0**	**55.0**	**72.0**	70.8	-	-

YOLOv5 模型在 GPU 下的效率测试数据图：

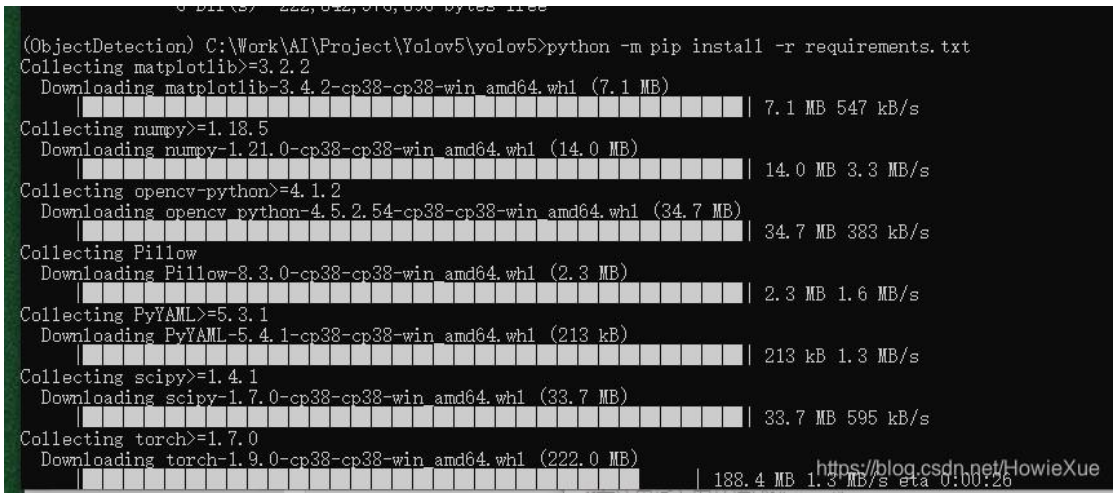


YOLOv5 模型在 Coco 数据集的训练参数图：



5.2 安装部署 YOLOv5 模型

源码下载完后，开始安装 YOLOv5 所需模块。命令行：pip install -r requirements.txt，等待安装完成即可。



如果没有 cuda 默认安装的 pytorch-cpu 版，如果有 gpu 可以安装GPU版：https://pytorch.org/get-started/locally/，可以提升模型运行速度。

也可以打开看下 requirements.txt 的依赖包：

```
# pip install -r requirements.txt

# base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow
PyYAML>=5.3.1
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0

# logging -------------------------------------
tensorboard>=2.4.1
# wandb

# plotting ------------------------------------
seaborn>=0.11.0
pandas

# export --------------------------------------
# coremltools>=4.1
# onnx>=1.9.0
# scikit-learn==0.19.2  # for coreml quantization

# extras --------------------------------------
# Cython  # for pycocotools https://github.com/cocodataset/cocoapi/issues/172
# pycocotools>=2.0  # COCO mAP
thop  # FLOPs computation
```

5.3 测试 YOLOv5

使用源码中的 Detect.py 即可。

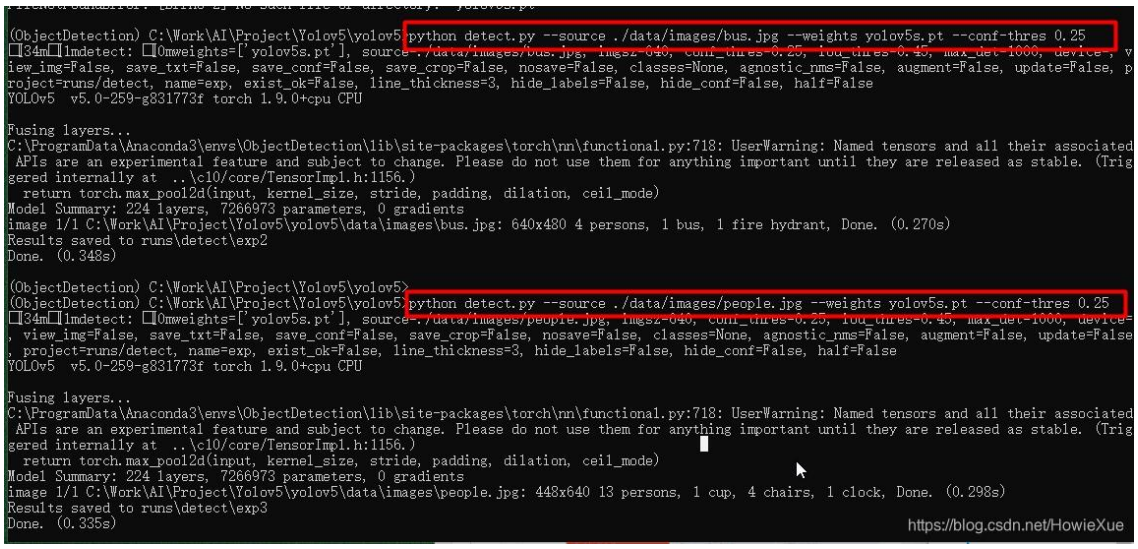
```
python detect.py --source 0  # webcam
                           file.jpg  # image
                           file.mp4  # video
                           path/  # directory
                           path/*.jpg  # glob
                           'https://youtu.be/NUsoVlDFqZg'  # YouTube video
                           'rtsp://example.com/media.mp4'  # RTSP, RTMP,
```

（最新版还支持了直接通过 Youtube 的 url 进行目标检测。）

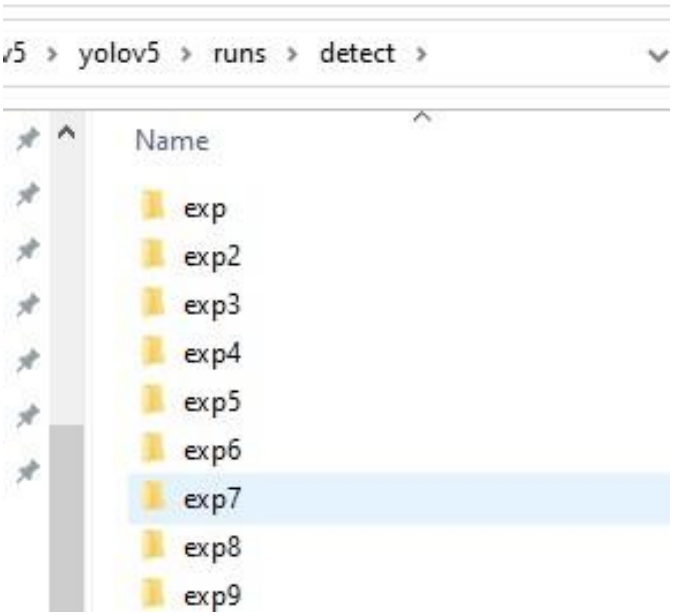
5.3.1 输入图像测试通过 detect.py 对图像进行目标检测

python detect.py --source ./data/image/bus.jpg

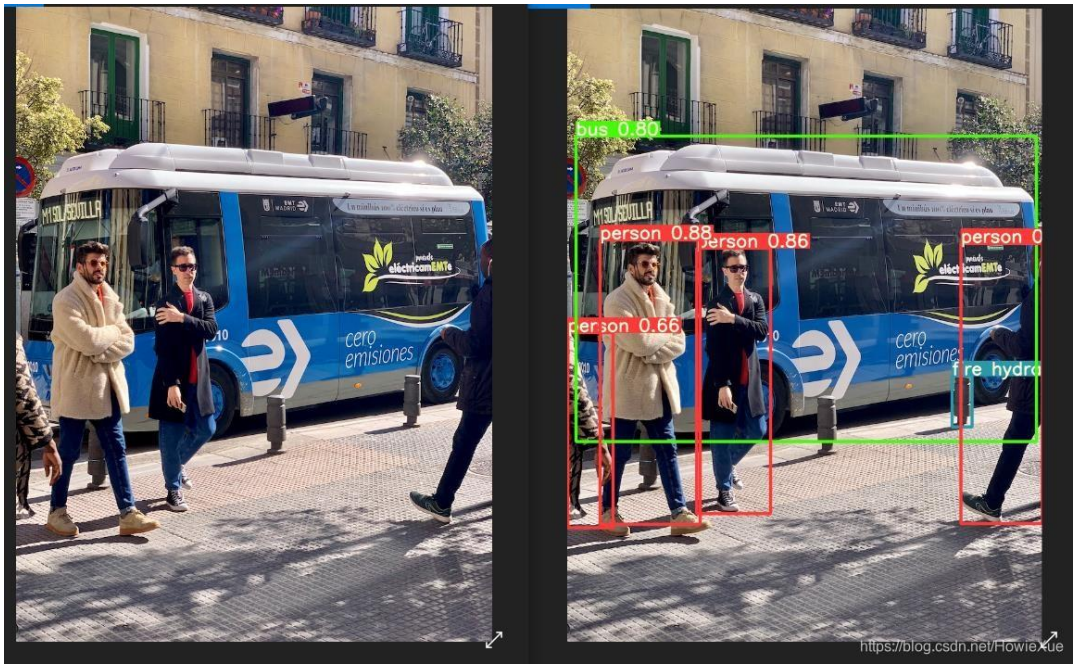
（说明：detect.py 默认使用同目录下的 YOLOv5s.pt 模型，如果想用其他的权重，可以用 -weights 进行指定。）



然后，可以在 YOLOv5/runs/detect 目录下找到模型输出结果的文件夹：



接下来，查看一下源码自带的 bus.jpg 的识别效果：

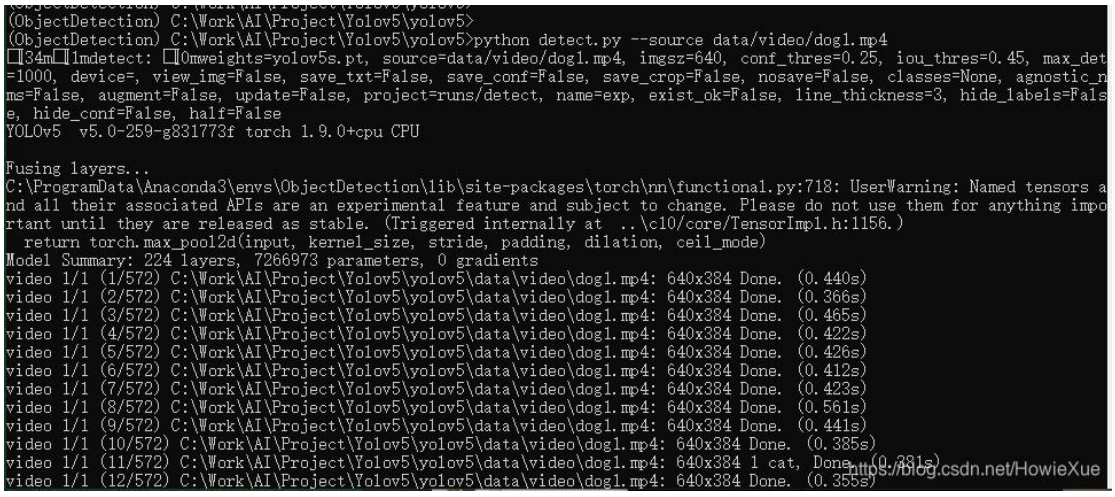


5.3.2 Video 视频测试

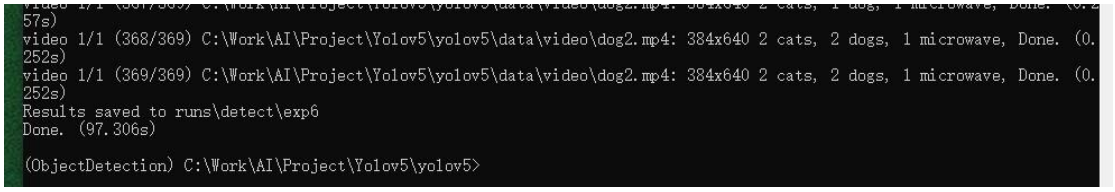
官方源码自带没有视频 demo，我们将待测视频放到 data 目录中即可：

然后，通过 detect.py 对视频进行目标检测：

```
python detect.py --source data/video/dog1.mp4
```



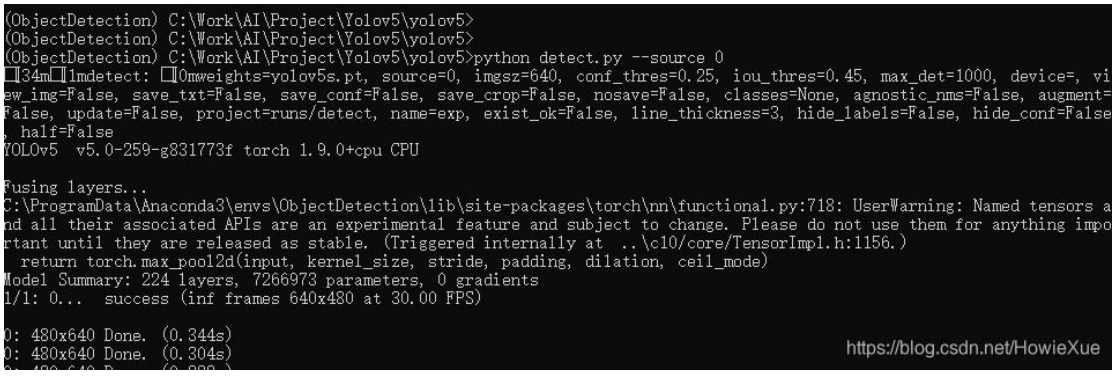
视频 Detect 运行速度有些略慢，是因为视频需要先转换为图片，再放到模型里处理。



5.3.3 摄像头测试

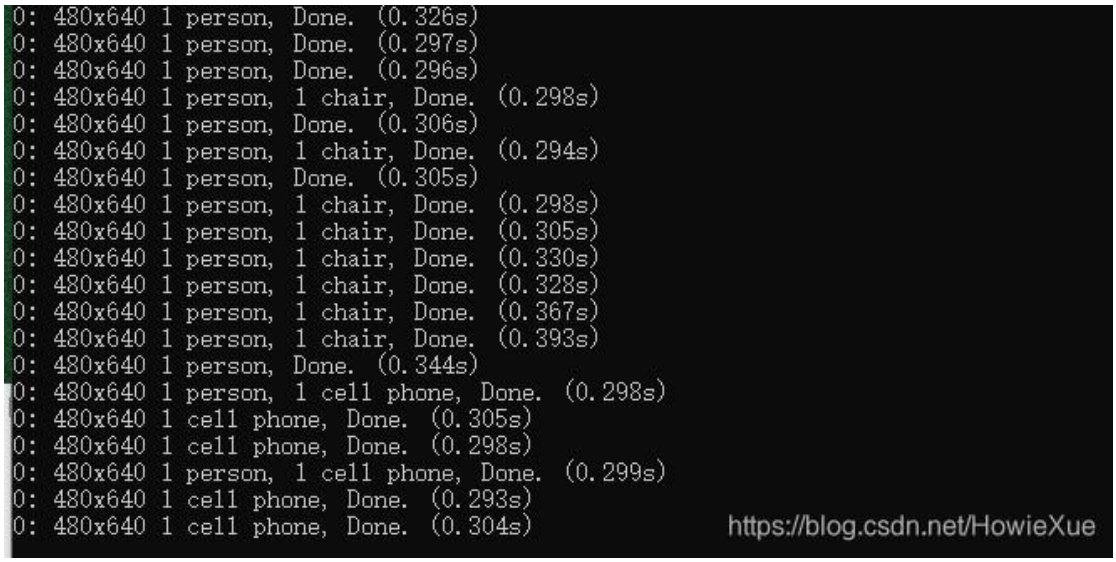
使用本机摄像头测试 YOLOv5 实时检测：

python detect.py --source 0



之后就会弹出摄像头页面，识别到的物体会用不同颜色的方框进行标识，同时上面会显示名称和概率（置信度）。

如果通过摄像头检测出识别的物体会打印出来：比如下面的 person、cell、phone 等。



至此，YOLOv5 的安装使用和测试流程都已经成功实现。YOLOv5 配置简单，并且识别效率和速度都是很棒的。

六、在自定义数据集上运行 YOLOv5

在自定义数据集上运行 YOLOv5 的具体细节，请参考博客：https://howiexue.blog.csdn.net/article/details/118463534?ydreferer=aHR0cHM6Ly9ob3dpZXh1ZS5ibG9nLmNzZG4ubmV0L2FydGljbGUvZGV0YWlscy8xMTg0NDU3NjY%2Fc3BtPTEwMDEuMjAxNC4zMDAxLjU1MDY%3D。

实际上，Github 官网的预训练模型的识别率等也有一定缺陷，并不能很好的直接应用在实际工程中，在大多数应用中都需要重新调整，并基于相应数据集训练适用于自己项目的模型。参数调优过程一般要反复多次进行微调-训练-测试，最终得出符合需求较优的 HyperPara，应用在项目中。调整超参数在 data/hyps/hyp.finetune.yaml：

最终主要是调整 lr0(学习率)为 0.0030，同时修改了 epoch、batch-size，部分超参数含义：

- lr0: 0.0032 #学习率
- lrf: 0.12 #余弦退火超参数 (Cosine Annealing)
- momentum: 0.843 #学习率动量
- weight_decay: 0.00036 #权重衰减系数
- warmup_epochs: 2.0 #预热学习 epoch
- warmup_momentum: 0.5 #预热学习率动量
- warmup_bias_lr: 0.05 #预热学习率
- box: 0.0296 #giou 损失的系数
- cls: 0.243 #分类损失的系数
- cls_pw: 0.631 #分类 BCELoss 中正样本的权重
- obj: 0.301 #有无物体损失的系数
- obj_pw: 0.911 #有无物体 BCELoss 中正样本的权重
- iou_t: 0.2 #标签与 anchors 的 iou 阈值（iou-training-threshold）

当然，针对视频图像识别中，移动的物体识别率本身就比静止图像差，同时运动检测越剧烈，偏差越大。

同时还要考虑到，在测试过程中，手工拍照的数据量比较小，也可能会导致模型欠拟合，造成模型识别效果不良。实际图像数据量，每个类别最好大于 1500 张，并且做一些图像增强、变换等数据预处理技术来增加模型的鲁棒性，防止过拟合。针对模型调优的技巧，可参考官网

的 Tips ： https://docs.ultralytics.com/tutorials/training-tips-bestresults/。

在进行模型测试时，无论是加载模型的速度还是对测试图片的推理速度，都能明显感觉到 YOLOv5 速度更快，尤其是加载模型的速度，因为同样的数据集训练出来的模型 YOLOv5 更加轻量级，模型大小减小为 YOLOv3 的将近四分之一。

至此，YOLOv5 自定义数据训练、测试和模型调优都已完成。

七、实验总结

本实验旨在介绍和部署 YOLOv5 模型进行目标检测任务，并评估其在自定义数据集上的性能表现。通过实验完成以下主要任务：

7.1 数据准备：

(1)使用 LableImg 标注工具对自定义数据集进行标记。

(2)将标记的数据集拆分为训练集和验证集。

7.2 模型训练：

(1)使用 YOLOv5 模型在训练集上进行训练。

(2)优化训练参数，包括学习率、批量大小等。

(3)监控训练过程，记录损失和指标的变化。

7.3 模型评估：

(1)在验证集上评估训练后的 YOLOv5 模型。

(2)计算目标检测的准确率、精确率、召回率和 F1 分数。

(3)可视化模型的检测结果，分析模型的性能和缺陷。

7.4 结果分析：

(1)分析模型在不同类别目标上的表现差异。

(2)探讨模型在不同尺度、角度和遮挡情况下的鲁棒性。

总的来说，本实验为学生提供了一个全面理解和应用 YOLOv5 目标检测模型的机会，并为进一步研究和实践计算机视觉领域提供了基础和启发。