

2025.11.8

YOLO环境配置

YOLO推理入门

2025.11.9

YOLO训练入门-上

1. 用 coco8数据集 跑通基本的训练
2. 理解什么是数据集

2025.11.11

YOLO训练入门-上

3. 安装并使用 labeling

YOLO训练入门-下

搞搞YOLO数据集

1. 准备数据集
2. 训练自己的模型

YOLO深入验证

2025.11.13

STFT

SAHI

2025.11.16

2025.11.17

SAHI (Slicing Aided Hyper Inference)

1. 将YOLO11和SAHI结合使用于切片推理

2025.11.8

YOLO环境配置

下载ultralytics源码

下载预训练模型 (.pt文件)

安装 Anaconda3-2024.06-1-Windows

Select Installation Type 强烈建议选择 Just me

Advanced Installation Options 保持默认即可

Anaconda 配置清华镜像源 (可选) :

1. conda 镜像源

1. 打开 Anaconda Prompt (注意: 一般情况下打开 Anaconda Prompt 是不用以管理员身份打开的)
2. 在 base 环境下输入 notepad .condarc
3. 打开 <https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/>
4. 将:

```
1 channels:  
2   - defaults  
3 show_channel_urls: true  
4 default_channels:  
5   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main  
6   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r  
7   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2  
8 custom_channels:  
9   conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud  
10  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

复制到刚刚打开的记事本

5. 在 base 环境下输入 type .condarc, 检查输出

2. pip 镜像源

1. 打开 <https://mirrors-i.tuna.tsinghua.edu.cn/help/pypi/>
2. 将 `pip config set global.index-url`
`https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple` 复制到 prompt 中直接运行

Anaconda 常用命令：

命令	作用
conda env list	列举所有环境及保存路径
conda create -n <name> python=3.xx	创建一个名为<name>的环境，python版本为3.xx
conda remove -n <name> --all	删除名为<name>的环境
conda activate <name>	激活（进入）名为<name>的环境
conda deactivate	退出当前环境
python --version	查看当前环境的Python版本
pip list	列举当前环境安装的所有包
pip install <package>==x.xx	为当前环境安装名为<package>的包，包的版本为x.xx
pip show <package>	查看当前环境中<package>这个包的信息
pip uninstall <package>	为当前环境卸载名为<package>的包

创建虚拟环境：

1. 打开
2. 输入 `conda create -n hwh_ultralytics_win python=3.11`

安装 pytorch 包：

1. 打开 torch.org
2. 选择对应版本：

PyTorch Build	Stable (2.9.0)		Preview (Nightly)		
Your OS	Linux	Mac	Windows		
Package	Pip	LibTorch	Source		
Language	Python		C++ / Java		
Compute Platform	CUDA 12.6	CUDA 12.8	CUDA 13.0	ROCm 6.4	CPU
Run this Command:	<pre>pip3 install torch torchvision --index-url https://download.pytorch.org/whl/cu128</pre>				

3. 将命令复制到 anaconda prompt
4. 安装成功后输入 python 进入交互环境，输入 import torchvision 和 import torch
5. 输入 torch.cuda.is_available()，返回 True 代表可以使用 gpu\
6. 输入 torch.randn(1).cuda()，注意返回的是 cuda:0 代表使用的显卡的编号是0
7. ctrl+z 回车退出python

安装其它包：

1. 找到源码文件夹路径：
2. 打开 anaconda prompt, 输入 `pushd <源码文件夹路径>`, 进入源码文件夹中
3. 输入 `pip install -e .` 意思是：
 1. 把当前源码文件夹作为一个可编辑的 python 包安装到当前的虚拟环境中（-e 的意思是可编辑模式，别的项目 import ultralytics 时，不会将源码复制一份过去，而是链接到该文件夹下）
 2. 同时自动安装当前源码在 pyproject.toml 中声明的依赖包
4. 尝试 `yolo detect predict`，结束后在源码文件夹下 `runs\detect\predict` 中查看结果

安装 Pycharm（专业版 为了使用ssh，现在社区版和专业版合并了）：

1. 运行安装程序
2. 切换中文
3. 添加解释器
4. 尝试运行简单推理代码：

```
1 from ultralytics import YOLO
2
3 model = YOLO(r"yolo11n.pt")
4 model.predict(
5     source=r'ultralytics\assets',
6     save=True,
7     show=False
8 )
```

5.

YOLO推理入门

推理 官网文档：docs.ultralytics.com/zh/modes/predict/

加载模型：

```
1 model = YOLO(r"yolo11n.pt")
```

```
1 print(model.task) # 输出此模型的任务（YOLO 支持的任务类型：目标检测 / 旋转目标检测 /
姿态估计 / 实例分割 / 图像分类）
2 print(model.names) # 输出此预训练权重中可以识别的种类的名字
```

有关 `model.names` 的研究：

（训练前加载预训练权重并打印 `model.names`，训练后加载训练得到的权重并打印 `model.names`）

```
1 | print(sum(p.numel() for p in model.parameters())) # 有无更简单的查看参数量的方法?
```

预训练模型及其任务:

	Detect 目标检测	OBB 旋转目标检测	Segment 实例分割	Pose 姿态估计	Classify 图像分类
YOLOv5	yolov5nu.pt yolov5su.pt yolov5mu.pt yolov5lu.pt yolov5xu.pt				
YOLOv8	yolov8n.pt yolov8s.pt yolov8m.pt yolov8l.pt yolov8x.pt	yolov8n-obb.pt yolov8s-obb.pt yolov8m-obb.pt yolov8l-obb.pt yolov8x-obb.pt	yolov8n-seg.pt yolov8s-seg.pt yolov8m-seg.pt yolov8l-seg.pt yolov8x-seg.pt	yolov8n-pose.pt yolov8s-pose.pt yolov8m-pose.pt yolov8l-pose.pt yolov8x-pose.pt	yolov8n-cls.pt yolov8s-cls.pt yolov8m-cls.pt yolov8l-cls.pt yolov8x-cls.pt
YOLOv9	yolov9t.pt yolov9s.pt yolov9m.pt yolov9c.pt yolov9e.pt		yolov9c-seg.pt yolov9e-seg.pt		
YOLOv10	yolov10n.pt yolov10s.pt yolov10m.pt yolov10b.pt yolov10l.pt yolov10x.pt				
YOLOv11	yolo11n.pt yolo11s.pt yolo11m.pt yolo11l.pt yolo11x.pt	yolo11n-obb.pt yolo11s-obb.pt yolo11m-obb.pt yolo11l-obb.pt yolo11x-obb.pt	yolo11n-seg.pt yolo11s-seg.pt yolo11m-seg.pt yolo11l-seg.pt yolo11x-seg.pt	yolo11n-pose.pt yolo11s-pose.pt yolo11m-pose.pt yolo11l-pose.pt yolo11x-pose.pt	yolo11n-cls.pt yolo11s-cls.pt yolo11m-cls.pt yolo11l-cls.pt yolo11x-cls.pt
YOLOv12	yolo12n.pt yolo12s.pt yolo12m.pt yolo12l.pt yolo12x.pt	一到目标检测这个任务 yolov11和yolov12模型命名比较特殊 少了一个"v"@林亿饼			

2025.11.9

YOLO训练入门-上

1. 用 coco8数据集 跑通基本的训练

排除潜在报错:

1. 打开 %appdata% 文件夹
2. 找到 Ultralytics 文件夹
3. 删除其中的 settings.json 文件

1. Pycharm 打开 Ultralytics 项目

2. python文件:

```

1  from ultralytics import YOLO
2
3  if __name__ == '__main__':
4      model = YOLO(r"yolo11n.pt")
5      model.train(
6          data=r'coco8.yaml',
7          epochs=10,
8          imgsz=640,
9          batch=2,
10         cache=False,
11         workers=0,
12     )

```

3. 运行

若因下载错误报错，可以手动下载，并将其放在报错信息中提示的路径下

4. 查看最后返回的信息以找到训练结果的路径

coco8.yaml 存放的路径在 ultralytics/cfg/datasets 中，也就是说，训练的data选项传入的路径会自动加上这个前缀

2. 理解什么是数据集

训练集:验证集:测试集 = 8:1:1

数据集配置文件格式为 yaml，存放路径为 ultralytics/cfg/datasets

内容为：

```

1  path:
2  train:
3  val:
4  test:
5
6  names:

```

数据集格式：

```

1  datasets\mydataset
2  - images
3    - train
4    - val
5  - labels
6    - train
7    - val

```

images文件夹中存放数据，格式为 .jpg

labels文件夹中存放标签，格式为 .txt

标签格式：（这种格式的标签称为YOLO格式标签）

1. 每行代表一个框
2. 每行共五个数字：
 1. 第一个整数代表类型编号（yaml文件中names字段的配置）
 2. 后四个小数代表 经过长宽归一化后的 $(cx, cy, w, h) = (center_x, center_y, width, height)$

（注意，图片的左上角是坐标原点，向右是x轴正方向，向下是y轴正方向）

2025.11.11

YOLO训练入门-上

3. 安装并使用 labelimg

安装labelimg:

1. 创建一个 python=3.8 的虚拟环境
2. `pip install labelimg` （使用系统代理好像会无法下载，可以尝试使用虚拟网卡）

使用labelimg:

1. 准备：
 1. 图片文件夹路径:
 2. 标签文件夹路径:
 3. classes.txt 路径: （注意：classes.txt 必须放在标签文件夹中）

1 | `classes.txt` 也就是将数据集的yaml配置文件的names字段复制过去，并将每行前面的编号、冒号、空格删除

2. 打开 anaconda prompt, 进入 labelimg 虚拟环境, 输入:

`labelimg <图片文件夹路径> <classes.txt文件的路径> <标签文件夹路径>`

3. 左上角 view, 勾选 Display Labels

YOLO训练入门-下

通过任务管理器查看硬件资源的利用率

win11任务管理器gpu窗口如何看cuda占用率:

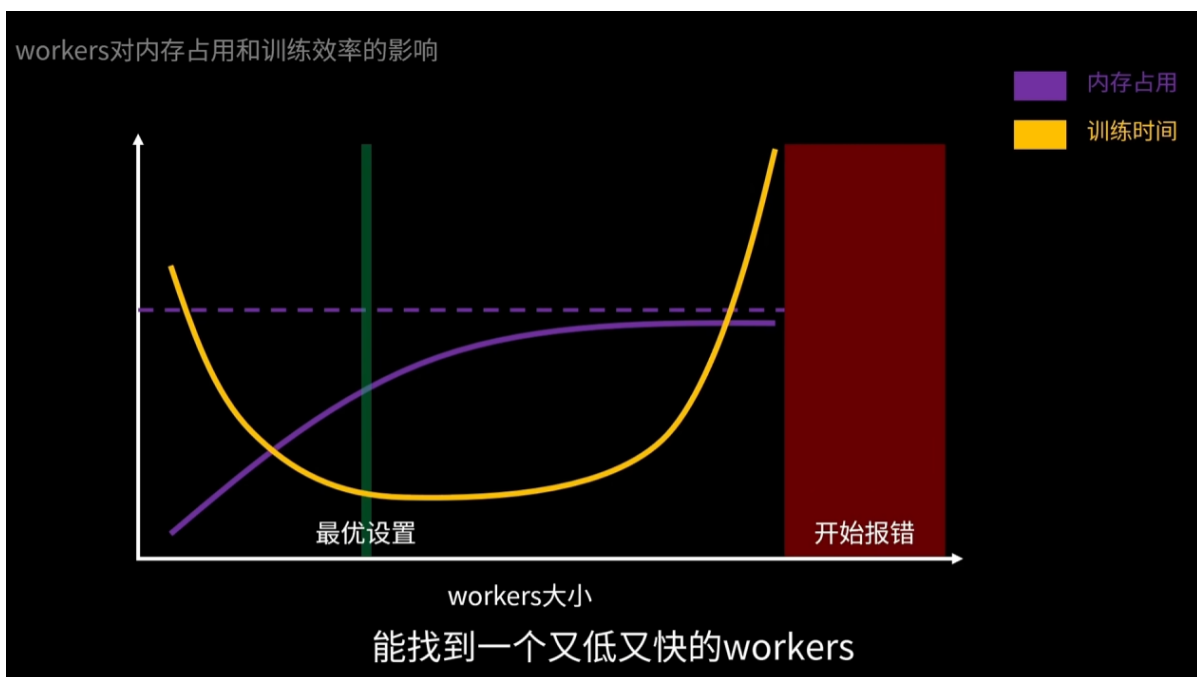
1. 打开 设置-系统-屏幕-显示卡-高级图形设置
2. 关闭 硬件加速GPU计划
3. 重启电脑

高效训练的特征：

1. Cuda 占用又高又稳
2. 硬件利用率都不到 100%

训练选项：

1. imgsz：将输入的图片保持长宽比缩放到能够放进 640*640 的正方形中（imgsz必须是32的倍数）（默认640）
2. batch：batch size
3. cache：使用缓存（可以选择 'ram'）
4. workers：（workers增大，内存消耗增大）





📌 Note

做 数据增强（图片随机预处理）的消融实验

搞搞YOLO数据集

1. 准备数据集

首先查有没有现成的：

两个网站：

1. roboflow universe
2. kaggle

如果没有现成的数据集，则自行制作：

1. 获取图片：

1. 网络搜索图片
2. 自行采集图片：
 1. 随即拍摄几段视频
 2. 从视频中抽取帧（用ai生成python代码）

2. 获取标签：（经过第一步以后，已经得到了images文件夹-train/val）

1. 新建labels文件夹
2. 在laebIs文件夹下创建classes.txt，并写入类别名称
3. 使用labelimg打开
4. 确认左侧显示的是YOLO
5. 开始拉框（A和D快速切换上一个/下一个图片，W快速新建一个框）
6. （View - Auto Save Mode）

半自动标注：原数据集图片数量较大，都没有标注

1. 将原数据集中的图片分出一小部分作子集
2. 分为train和val，并手动标注
3. 按标准流程训练一个模型
4. 在推理脚本中，载入best.pt模型，数据源选择原数据集图片，参数 `save_txt` 设置为True，推理后的predict结果文件夹中

2. 训练自己的模型

首先可以尝试基于预训练的权重进行微调：

然后可以再尝试选择某一模型但是从头开始训练：

Tips:

1. 注意，不是版本越高效果越好，需要多尝试。
2. 代码实现中可以使用列表来自动切换模型
3. 在每次训练脚本投入使用前，都先用一个epoch=3进行测试

YOLO深入验证

验证 即 对验证集通过模型后的预测结果进行打分，用来判断当前模型权重效果好不好，是否需要保存为best.pt

$IoU = \text{交集面积} / \text{并集面积} ()$

自行设置IoU阈值，以区分预测框正确与否

TP = True Positive = 正确的预测框（即真实框）

FP = False Positive = 错误的预测框

FN = False Negative = 漏掉的真实框

$P = \text{Precision} = \text{精确率} = \text{正确的预测框} / \text{所有的预测框} = TP / (TP+FP)$

$R = \text{Recall} = \text{召回率} = \text{正确的预测框} / \text{所有的真实框} = TP / (TP+FN)$

$F1 \text{ Score} = 2PR / (P+R)$

1. 预处理
2. 模型预测
3. 后处理：

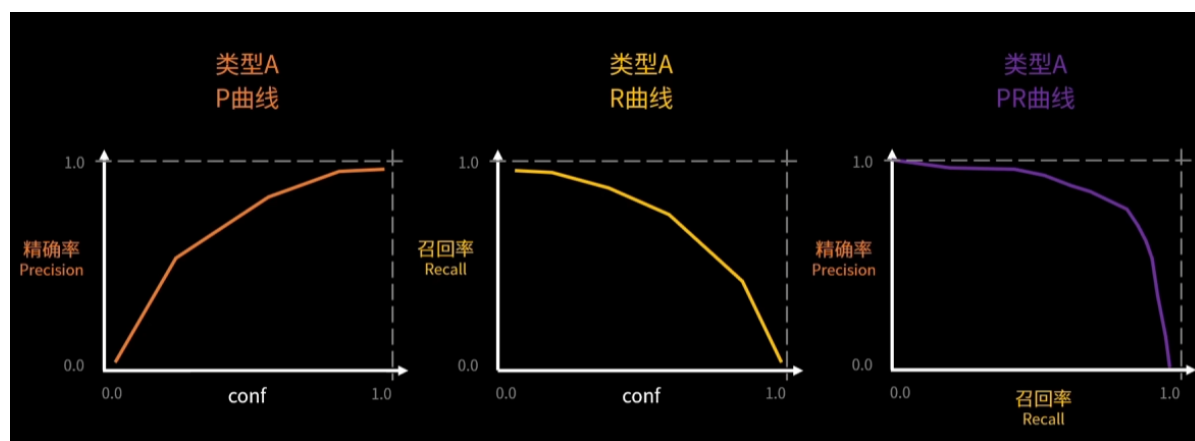
1. 排除置信度 `conf` 低于阈值的框

2. 通过非极大值抑制 NMS 排除重叠框
3. 通过置信度限制预测框总数

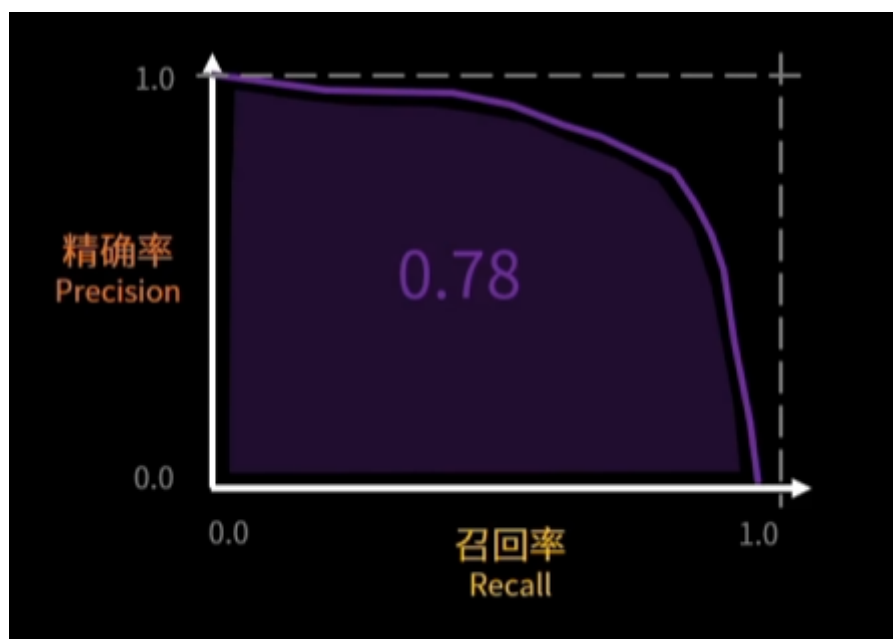
得到所有预测结果后，通过变化置信度阈值 threshold_conf 可以得出一组组的预测框（比如 $\text{threshold_conf}=0.2$ 时，有一组预测框符合要求； $\text{threshold_conf}=0.4$ 时，又是另一些框符合要求），将每组预测框与真实框进行对比，（固定 threshold_iou 的情况下）可以得到 threshold_conf 和 $P / R / F1$ 的函数关系，从而得到一个使 $F1$ 取到最大值的 最优 threshold_conf



在 $\text{threshold_conf} - P$ 和 $\text{threshold_conf} - R$ 这两个函数中，相当于一个 threshold_conf 对应一个 P 和一个 R ，也就是说 这一个 P 和 这一个 R 是对应的，那么就可以导出 $R - P$ 的函数（横轴为 R ，纵轴为 P ），即为 PR 曲线



PR 曲线与 x 轴 和 y 轴 围成的面积即为 $AP = \text{Average Precision} = \text{平均精度}$



要想AP大，则需要PR曲线右上角的点尽可能靠近(1,1)，主要取决于 threshold-P 和 threshold-R 曲线的交点的y值尽可能大，即有某一个conf可以同时使得P和R都很多大

以上所说的都是对于某一个类别的预测框，也就是说 每个类别的预测框都可以得到一个AP值，对所有类别平均后可以得到 $mAP = \text{mean Average Precision} = \text{平均平均精度}$

以上所说的都是对于固定的 threshold_iou（决定预测框和真实框重合多少时才算这个预测框正确），随着IoU变化：从0.5到0.95，步进0.05，共10个threshold_iou等级，在每个等级上可以分别得到一个mAP值，做平均可以得到 $mAP@50:95$

（最常用：P / R / $mAP@50$ / $mAP@50:95$ ）

将这些指标加权求和，得到最终指标 fitness，作为该轮权重的最终分数

2025.11.13

STFT

参数选择

SAHI

https://www.bilibili.com/video/BV1njV3zEE6k/?spm_id_from=333.337.search-card.all.click&vd_source=dd2ca5ebb544df909fc692b8ba5d5efa

https://blog.csdn.net/qg_42589613/article/details/142738864

2025.11.16

跑通基线

2025.11.17

SAHI (Slicing Aided Hyper Inference)

1. 将YOLO11和SAHI结合使用于切片推理

参考文档: <https://docs.ultralytics.com/zh/guides/sahi-tiled-inference/>