

# Yolov3-Test



## 获取程序代码

使用Git下载程序的命令：

```
$ git clone https://github.com/HuangRunHua/Yolov3-Test.git
```

## 安装依赖项

Python 3.8或更高版本与全部 `requirements.txt` 内要求的依赖项，包括 `torch> = 1.7`。要安装所有依赖项，请运行：

```
$ cd yolov3  
$ pip install -r requirements.txt
```

## 训练自己的数据

在开始之前，请确保已经安装所有的依赖项。

## 1.自定义coco.yaml文件

YOLOv3的文件夹内配备预先设计好的 `coco1314.yaml` 文件，它存放在`data`文件夹内。该文件是数据集配置文件，该配置文件定义了

- 用于自动下载的可选下载命令/URL
- 用于训练、测试与验证的图像所在的目录
- 用于训练的类数量
- 类名列表

```
# Train command: python train.py --data coco.yaml
# Default dataset location is next to /yolov3:
#   /parent_folder
#     /coco
#     /yolov3

# download command/URL (optional)

# train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3)
list: [path1/images/, path2/images/]
train: ../coco1314/images/train/  # 1k images
val: ../coco1314/images/val/  # 5000 images
test: ../coco1314/images/test/  # 20288 of 40670 images, submit to
https://competitions.codalab.org/competitions/20794

# number of classes
nc: 6

# 人物的各种类别标签
#names: ['喵内', '日向', '乃爱', '夏音', '小依', '小花']
names: [
    'Hoshino_Miyako',
    'Hoshino_Hinata',
    'Himesaka_Noa',
    'Konomori_Kanon',
    'Tanemura_Koyori',
    'Shirosaki_Hana'
]

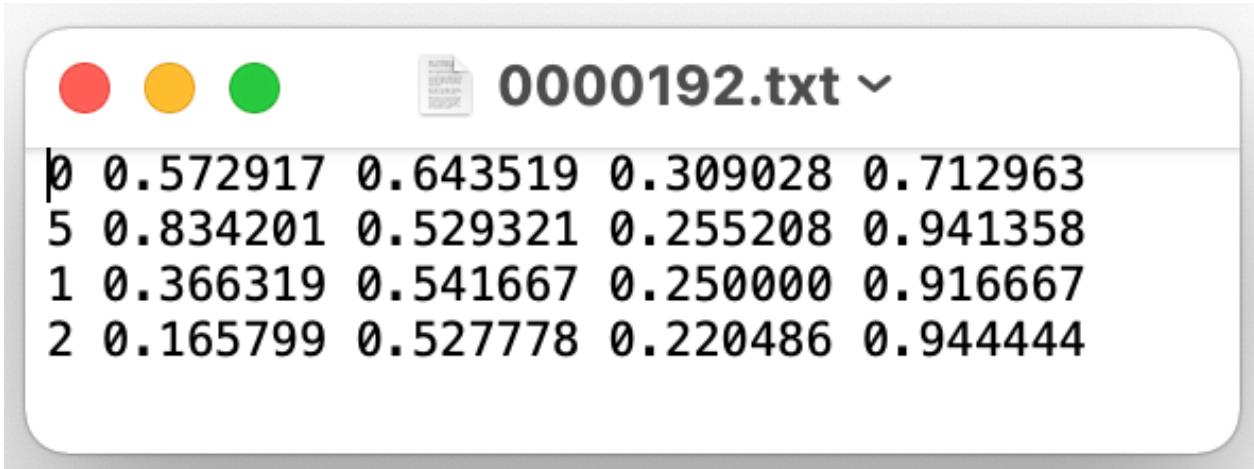
# Print classes
# with open('data/coco.yaml') as f:
#     d = yaml.load(f, Loader=yaml.FullLoader)  # dict
#     for i, x in enumerate(d['names']):
#         print(i, x)
```

⚠ 注意类名列表最好使用英文，如需使得最终的识别标签为中文，请参考[中文显示识别标签](#)一节。

## 2. 创建标签

使用[CVAT](#)、[makesense.ai](#)或[Labelbox](#)等工具标记图像后，将标签导出为**YOLO**格式，每张图像有一个`*.txt`文件（如果没有图像中对象，则不需要`*.txt`文件）。`*.txt`文件规格是：

- 每个对象一行
- 每行都是`class x_center y_center width height`格式。
- 框坐标必须采用规范化的`xywh`格式（从0-1开始）。如果您的框是像素，请按图像宽度除以`x_center`和`width`，将`y_center`和`height`除以图像高度。
- 用于训练的类标签号为零索引（从0开始，如下图所示）



`0000192.txt`文件共包括四个类，分别对应喵内（`0`类）、日向（`1`类）、乃爱（`2`类）和小花（`5`类）。

## 3. 数据集文件存放位置

`/coco1314`文件夹与`/yolov3`文件夹处于同一级别的目录下。**YOLOv3**通过将每个图像路径中的`/images/`的最后一个实例替换为`/labels/`来自动定位每个图像的标签。将标记好的数据集放入`image`文件夹与`labels`文件夹内，并按一定的比例储存到各自的训练集文件夹、测试集文件夹和验证集文件夹内。例如：

```
coco1314/images/train/0000192.jpg # image
coco1314/labels/train/0000192.txt # label
```

Name	Date Modified	Size	Kind
▼ coco1314	Today at 6:35 PM		-- Folder
▼ images	Today at 6:36 PM		-- Folder
> test	Today at 6:36 PM		-- Folder
> train	Today at 6:36 PM		-- Folder
> val	Today at 6:36 PM		-- Folder
> labels	Today at 6:36 PM		-- Folder
LICENSE	Today at 3:46 PM	1 KB	Document
README.md	Today at 6:35 PM	3 KB	Markdo...cument
> ReadMeImages	Today at 6:27 PM		-- Folder
> yolov3-master	🕒 Today at 6:03 PM		-- Folder

## 4.训练模型

通过指定数据集、批处理大小、图像大小和预训练的 `--weights yolov3.pt` 在COCO1314上训练YOLOv3模型，YOLOv3模型将会自动下载。通过指定如下语句开始训练：

```
# Train YOLOv3 on COCO1314 for 1000 epochs
$ python train.py --img 640 --batch 16 --epochs 1000 --data coco1314.yaml --weights
yolov3.pt
```

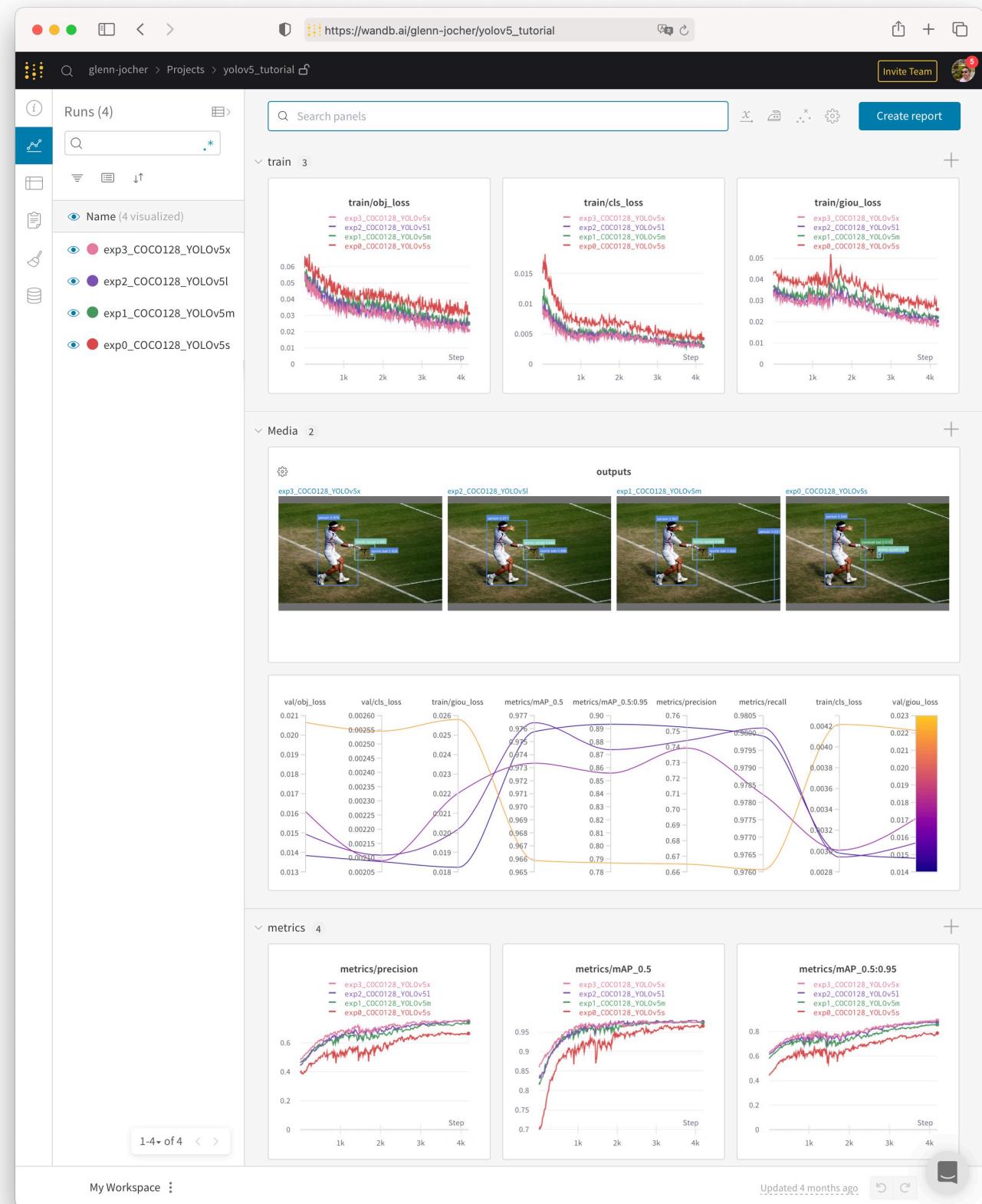
所有训练结果都保存到 `runs/train/` 带有运行目录内，并按照 `runs/train/exp2`、`runs/train/exp3` 等顺序保存所有训练结果。

## 训练过程的可视化

Wandb现已与YOLOv3集成，用于训练运行的实时可视化和云记录。这可以更好地运行比较，并提高团队成员之间的可见度和协作。要启用W&B日志记录，请安装 `wandb`，然后正常训练。采用pip安装 `wandb`：

```
$ pip install wandb
```

在数据训练期间，可以在<https://www.wandb.com/>上看到实时更新，同时可以使用W&B报告工具创建包含结果的详细报告。



## 本地日志记录

默认情况下，所有结果都会记录到 `runs/train`，并为每次新的训练结果如 `\`runs/train/exp2`、`runs/train/exp3` 等创建一个新的实验目录。

本实验的 `train_batch0.jpg` 如下图所示。

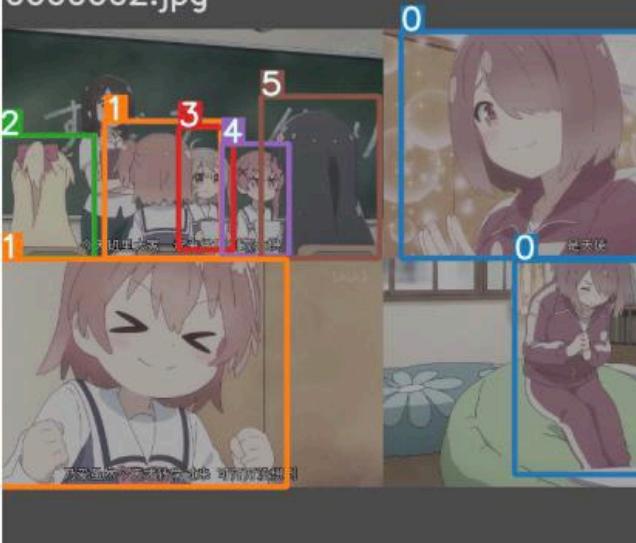
0000001.jpg



0000006.jpg



0000002.jpg



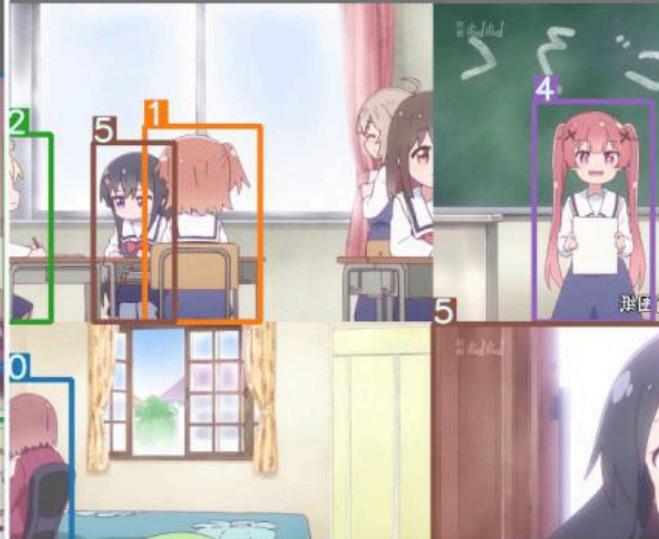
0000007.jpg



0000004.jpg



0000008.jpg



`test_batch2_labels.jpg` 表示部分用于测试的图片合成图，通常如下图所示。

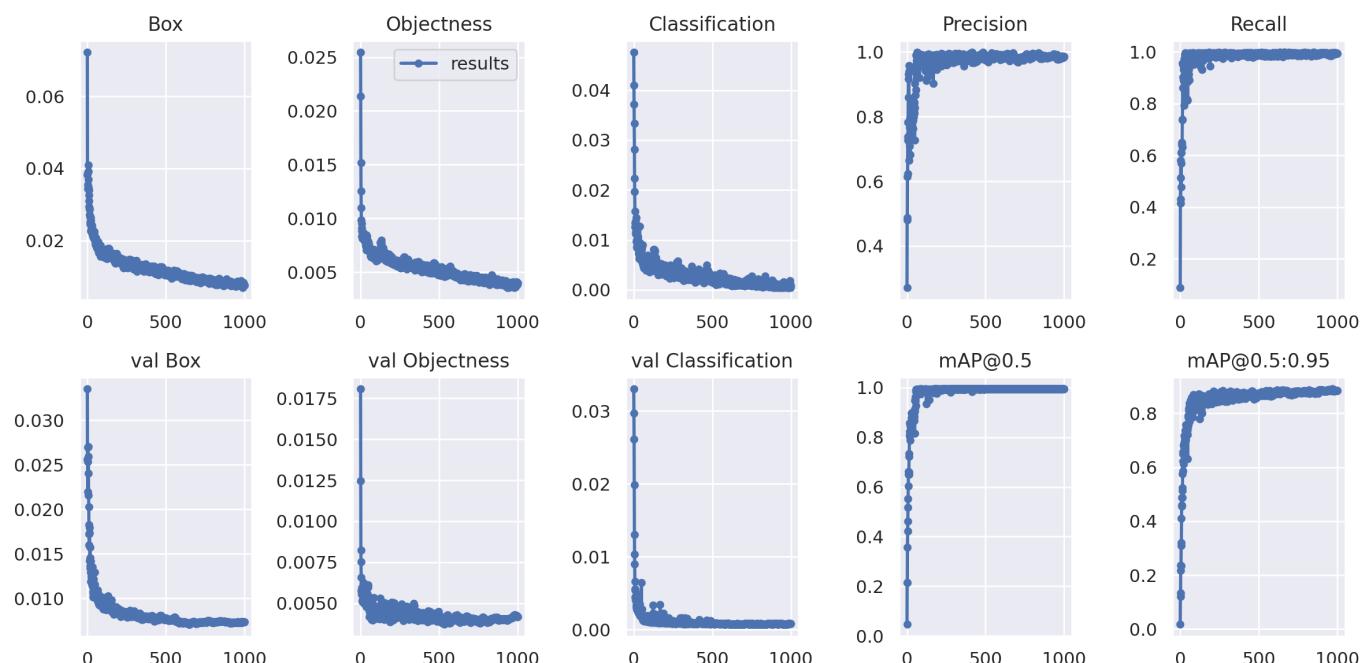


`test_batch2_pred.jpg` 显示了模型识别的结果集合，识别的图片与 `test_batch2_labels.jpg` 给定的图片相同。



**!** 若在 `coco1314.yaml` 文件内采用中文命名训练类别，将会发生中文字符不识别的现象。但不会影响最终的识别结果。

训练损失和性能指标也会记录到自定义 `results.txt` 的日志文件内，该日志文件在训练完成后绘制为 `results.png`（见下图）。



 模型训练完成后，将得到两个模型best.pt与last.pt。根据需要选择对应的模型即可。

## 模型实战

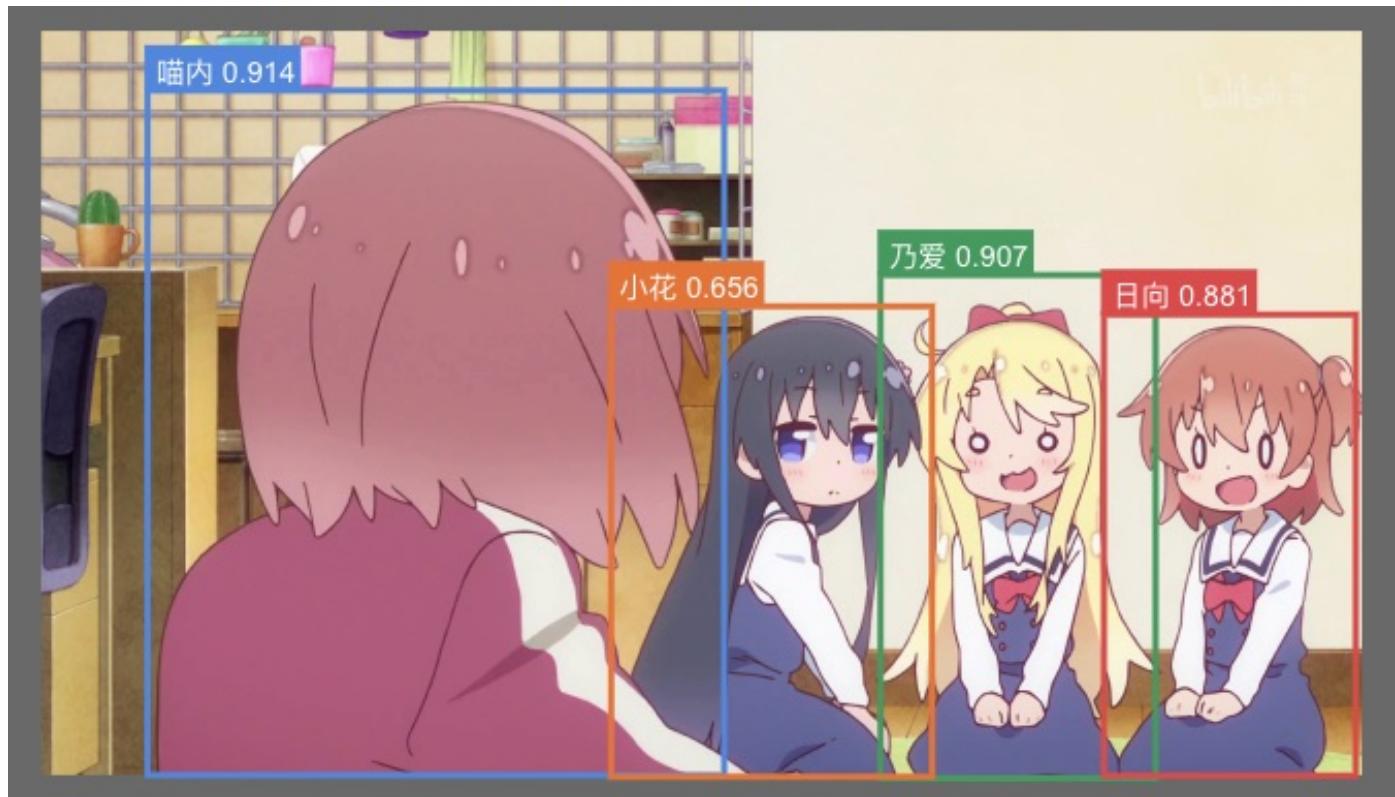
`detect.py` 为训练后的模型提供了施展的机会。

要对某个本地文件（例如 `data/images`）中的示例图像或视频进行推断，可将 `--source` 后的内容改成文件目录索引：

```
$ python detect.py --source data/images --weights best.pt --conf 0.25
```

## 中文显示识别标签

若希望模型实战后将识别的物体标签语言更改为中文，如下图所示：



需要修改 `yolov3-master/utils/plots.py` 内容。

1. 在函数 `plot_one_box_PIL(box, img, color=None, label=None, line_thickness=None)` 内添加中文字体所在的文件路径。

```

def plot_one_box_PIL(box, img, color=None, label=None, line_thickness=None):
    img = Image.fromarray(img)
    draw = ImageDraw.Draw(img)
    line_thickness = line_thickness or max(int(min(img.size) / 200), 2)
    draw.rectangle(box, width=line_thickness, outline=tuple(color)) # plot
    if label:
        fontsize = max(round(max(img.size) / 40), 12)
        font = ImageFont.truetype('font/simsun.ttc', fontsize)
        txt_width, txt_height = font.getsize(label)
        draw.rectangle([box[0], box[1] - txt_height + 4, box[0] + txt_width,
                       box[1]], fill=tuple(color))
        draw.text((box[0], box[1] - txt_height + 1), label, fill=(255, 255, 255),
                  font=font)
    return np.asarray(img)

```

**⚠** 本程序自带宋体文件，储存路径为 `yolov3-master/font/simsun.ttc`。

**⚠** 如需采用其他字体，请事先下载该字体并放入 `yolov3-master/font` 路径下。

2. `yolov3-master/detect.py` 文件内为训练的类别添加一个JSON格式的数据，数据内部的语言可以为中文，如：

```

mylabels = {
    '喵内': '喵内',
    '日向': '日向',
    '乃爱': '乃爱',
    '夏音': '夏音',
    '小依': '小依',
    '小花': '小花'
}

```

在函数 `detect(save_img=False)` 内修改绘图的代码：

```

# Write results
for *xyxy, conf, cls in reversed(det):
    # Write to file
    if save_txt:
        # normalized xywh
        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
        # label format
        line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)
        with open(txt_path + '.txt', 'a') as f:
            f.write(('%g' * len(line)).rstrip() % line + '\n')
    # Add bbox to image
    if save_img or view_img:
        # 若标签的命名为英文格式，则采用以下两行
        # label = f'{names[int(cls)]} {conf:.2f}'

```

```
# plot_one_box(xyxy, im0, label=label, color=colors[int(cls)],  
line_thickness=3)  
  
# 本程序标签的命名使用中文，因此使用以下两行代码  
label = f'{mylabels[names[int(cls)]]} {conf:.2f}'  
im0 = plot_one_box_PIL(xyxy, im0, label=label, color=colors[int(cls)],  
line_thickness=3)
```

## 示例视频

采用迭代1000次后得到的模型进行[天使降临到了我身边](#)视频的识别。点击以查看识别结果。

