2021-2022年第一期

《深度学习应用开发TensorFlow实践》

课程期末大作业

题 目 食堂浪费目标检测

姓 名 钱骥晗

单 位 浙大城市学院

专业班级 计算机1901

二○二一年 十二月 月 二十三 日

### 选题背景

粮食问题是世界各国都非常关注的问题,面对当今世界粮食生产和供应的严峻形势,节约粮食、适度消费刻不容缓,然而,现实中，在食紧缺的同时,食品浪费的现象在我国却广泛存在,与我国设资源节约型社会的目标相背离,因此,我们应在风气树立、制度约束、政策鼓励、法律惩处和消费模式等方面进行调整和完善,提高人们的节约意识，减少食品浪费。

因此，我们研究浙大城市学院北秀食堂的食堂浪费情况，希望对节约粮食的号召做出一点微薄之力。

### 开发环境

**SSD框架的开发环境：**

Mindspore-1.5.0版本

Windows 10是x86架构64位操作系统

Python 3.7.5版本

**Yolov5框架的开发环境：**

Pytorch-1.5.0版本

Windows 10是x86架构64位操作系统

Python 3.7.0版本

第三方库：requirements.txt文件

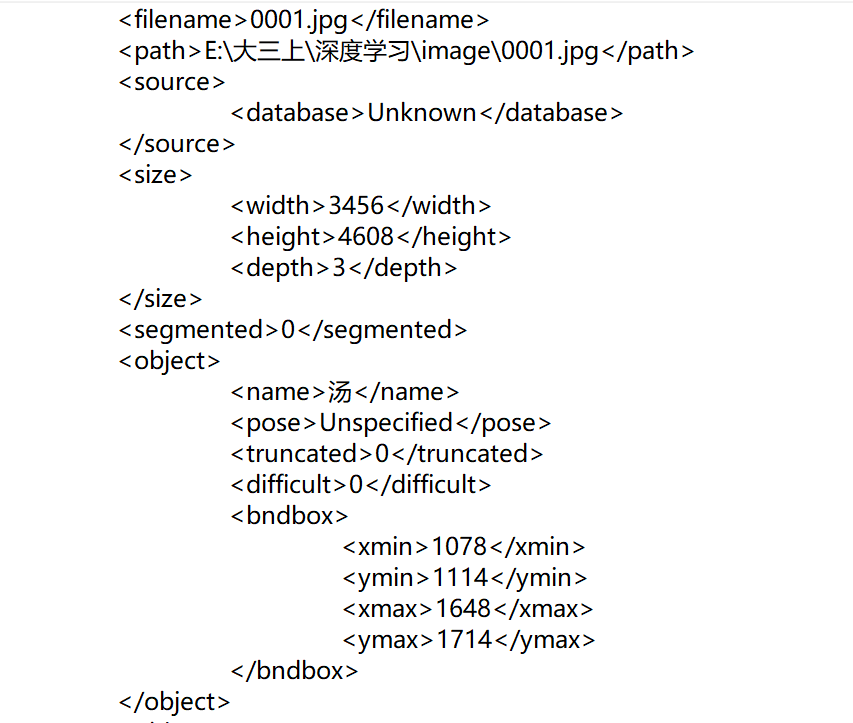
### 数据处理

数据集是浙大城市学院北秀食堂的餐盘图片数据。上面两组是南校一食堂、二食堂的餐盘图片，下面两组是北秀食堂的餐盘图片。经过训练得出，不同食堂因为餐盘餐碗形状不同，不能得出较好的训练效果，很多餐碗甚至无法框选出，所以终于选用北秀食堂作为数据集。

部分餐盘图片参照如下：数据来源于浙大城市学院北秀食堂拍摄的图片，数据集图片为JPG类型，将172张图片作为训练集，20图片作为验证集。



使用LabelImg软件，对图片进行人工标注，标签分为素菜、荤菜、汤、骨头、空、饭六类，其中将骨头、空和汤均认为没有浪费。



### 模型设计

4.1 SSD模型

4.1.1 SSD概述

本项目使用的SSD算法是一种直接预测bounding box的坐标和类别的object detection算法，没有生成proposal的过程。

针对不同大小的物体检测，传统的做法是将图像转换成不同的大小，然后分别处理，最后将结果综合起来，而本项目使用的SSD利用不同卷积层的feature map进行综合也能达到同样的效果。

**算法的主网络结构是VGG16**，将两个全连接层改成卷积层再增加4个卷积层构造网络结构。对其中5个不同的卷积层的输出分别用两个3\*3的卷积核进行卷积，一个输出分类用的confidence，每个default box生成21个confidence（针对VOC数据集包含20个object类别）；一个输出回归用的localization，每个default box生成4个坐标值（x，y，w，h）。

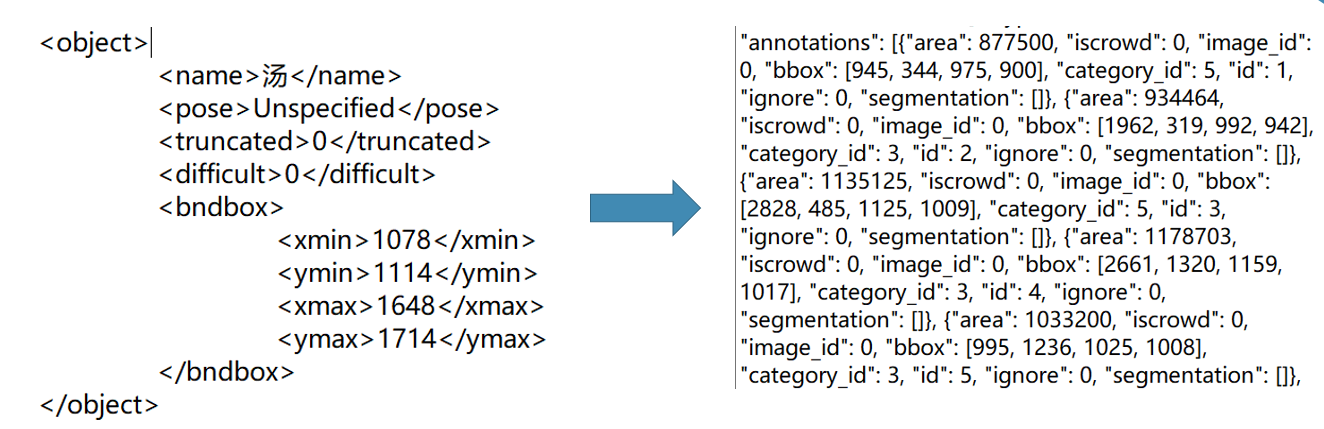
另外这5个卷积层还经过priorBox层生成default box（生成的是坐标）。上面所述的5个卷积层中每一层的default box的数量是给定的。最后将前面三个计算结果分别合并然后传递给loss层。

4.1.2 Mindspore框架的SSD模型

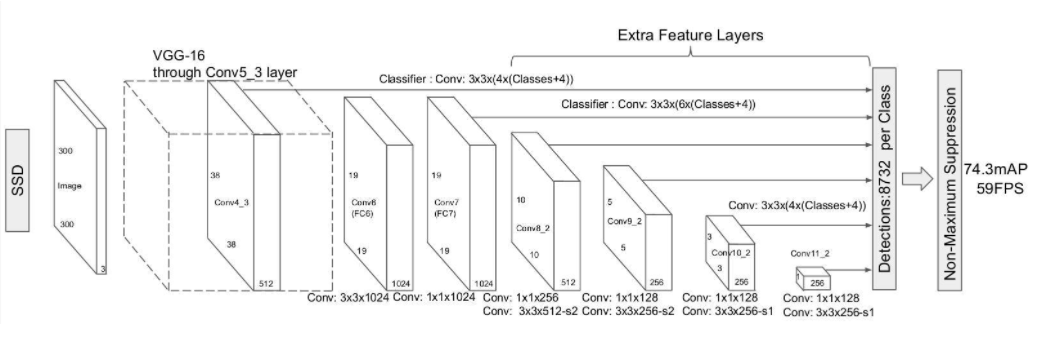
Mindspore的GPU版本只支持Linux系统，在window下只有SSD模型可以用来训练。进行据格式的转换后，可以直接调用。



主要是将标注的数据框信息转换为coco数据集的格式。



4.1.3 SSD网络结构



主要介绍关键步骤annotations字段：category\_id：该注释的类别id；id：当前注释的id号image\_id：该注释所在的图片id号area：区域面积bbox：目标的矩形标注框iscrowd：0或1。0表示标注的单个对象，此时segmentation使用polygon表示；1表示标注的是一组对象，此时segmentation使用RLE格式。

其中Bbox标注框主要分为两部分，第一部分是各个类别的置信度或者评分，值得注意的是SSD将背景也当做了一个特殊的类别，如果检测目标共有 c 个类别，SSD其实需要预测 c + 1 个置信度值，其中第一个置信度指的是不含目标或者属于背景的评分。后面当我们说 c 个类别置信度时，请记住里面包含背景那个特殊的类别，即真实的检测类别只有 c − 1 个。在预测过程中，置信度最高的那个类别就是边界框所属的类别，特别地，当第一个置信度值最高时，表示边界框中并不包含目标。第二部分就是边界框的location，包含4个值 (cx, cy, w, h)，分别表示边界框的中心坐标以及宽高。标注框位置用 d=(d^{cx}, d^{cy}, d^w, d^h)表示，其对应边界框用b=(b^{cx}, b^{cy}, b^w, b^h)表示，那么边界框的预测值 l 其实是 b 相对于d 的转换值：

在这里插入图片描述

在这里插入图片描述

预测时，你需要反向这个过程，即进行解码（decode），从预测值 l  中得到边界框的真实位置 b  ：

在这里插入图片描述

在这里插入图片描述

综上，边界框需要的解码方程式如下：

在这里插入图片描述在这里插入图片描述

4.2 Yolov5与SSD比较

SSD的不足之处：SSD在训练过程中，prior box与GroundTruth的之间IOU达到0.5才会放到网络里面进行训练。大的目标有可能ROI的值会大很多，因此包含的prior box就多，就可以得到充分的训练。相反小目标用于训练的的prior box就会少很多，就得不到充分的训练。所以有些餐盘会识别不到。

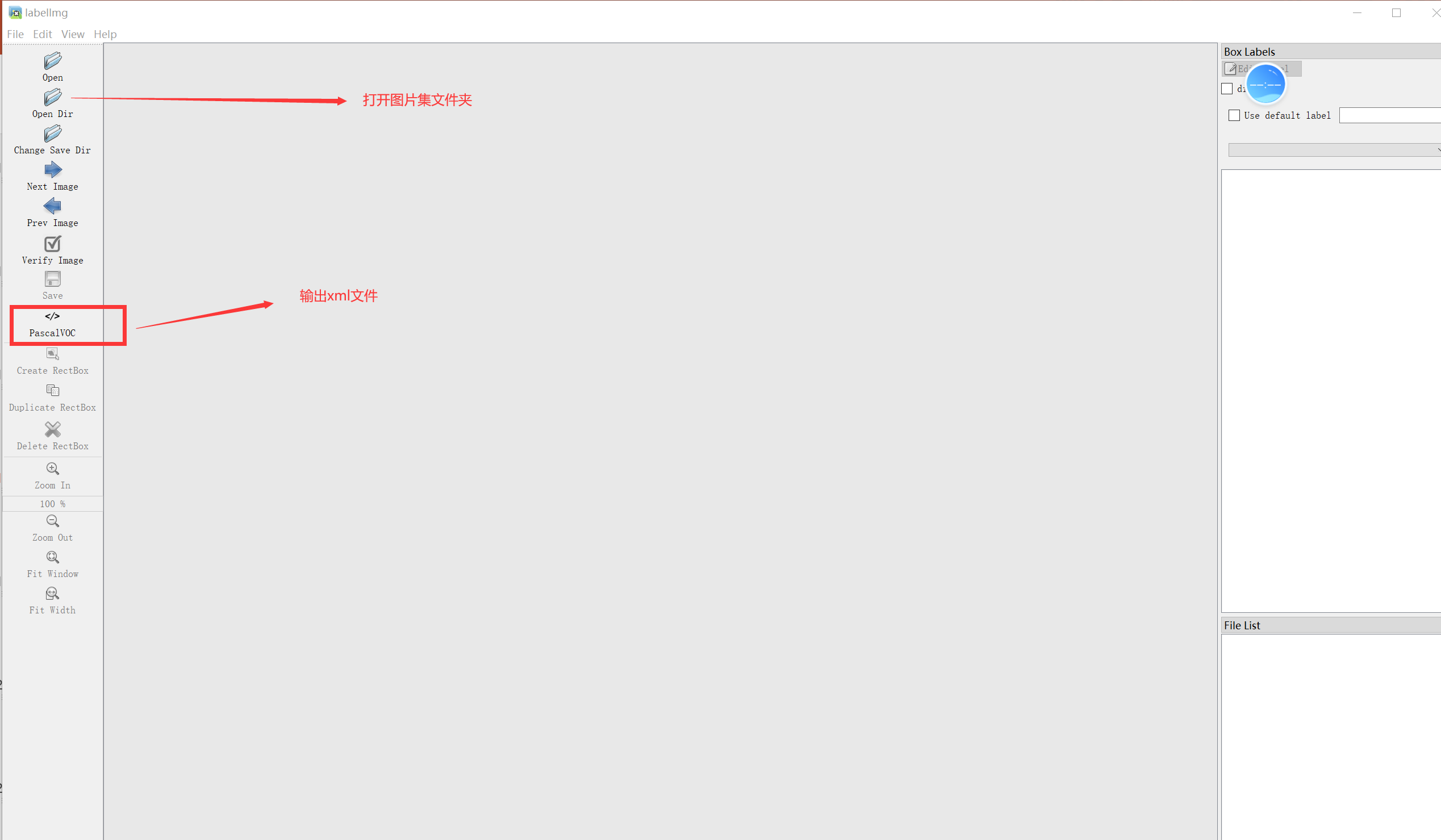
**Y**olov5可取之处：1）Bag of freebies（BoF）：提高精度而不增加推断时间2）解决数据不平衡的问题（背景和目标之间的不平衡；类别不平衡等）3）设计BBox回归的目标函数

4.3 Yolov5模型

4.3.1 Yolov5框架概述与项目预处理

Yolov5框架概述：本项目使用的YOLOv5是一种单阶段目标检测算法，该算法在YOLOv4的基础上添加了一些新的改进思路，使其速度与精度都得到了极大的性能提升。主要的改进思路：输入端：在模型训练阶段，提出了一些改进思路，主要包括Mosaic数据增强、自适应锚框计算、自适应图片缩放；基准网络：融合其它检测算法中的一些新思路，主要包括：Focus结构与CSP结构；Neck网络：目标检测网络在BackBone与最后的Head输出层之间往往会插入一些层，Yolov5中添加了FPN+PAN结构；Head输出层：输出层的锚框机制与YOLOv4相同，主要改进的是训练时的损失函数GIOU\_Loss，以及预测框筛选的DIOU\_nms。

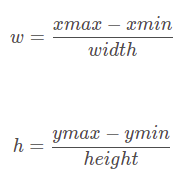
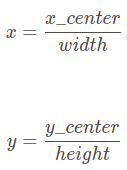
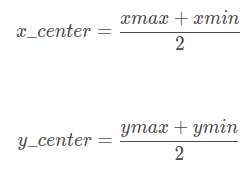
预处理是用LabelImg标注图片，LabelImg是图形图像注释工具。它是用Python编写的，并将Qt用于其图形界面。批注以PASCAL VOC格式（ImageNet使用的格式）另存为XML文件。此外，它还支持YOLO格式在我们的项目中使用方法如下：



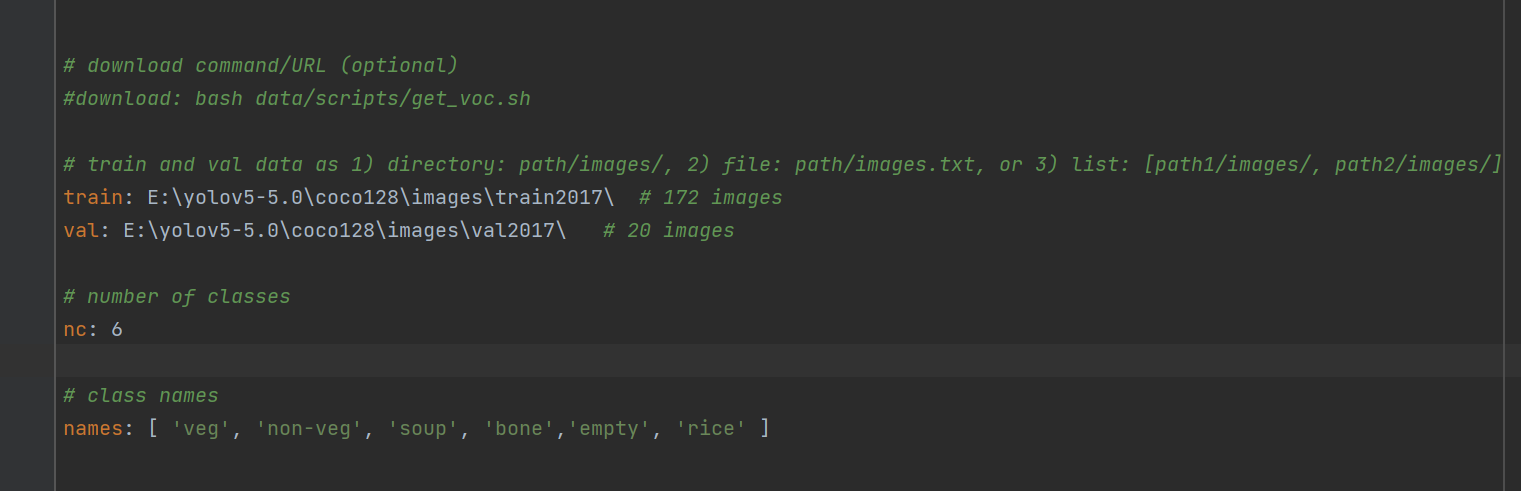
4.3.2 VOC格式xml标签转换为YOLO格式txt标签

* **voc格式标签**：图片的实际宽高，标注框的左上角和右下角坐标；
* **yolo格式标签**：标注框的中心坐标（归一化），标注框的宽和高（归一化）。

转换公式如下：



4.3.3 Python文件中对图片的导入以及标签的处理：

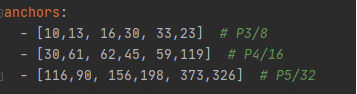


4.3.4 YOLOV5网络结构

（1）输入端：Mosaic数据增强、自适应锚框计算、自适应图片缩放  
（2）Backbone：Focus结构，CSP结构  
（3）Neck：FPN+PAN结构  
（4）输出端：GIOU\_Loss

**1）输入端**

①数据增强：

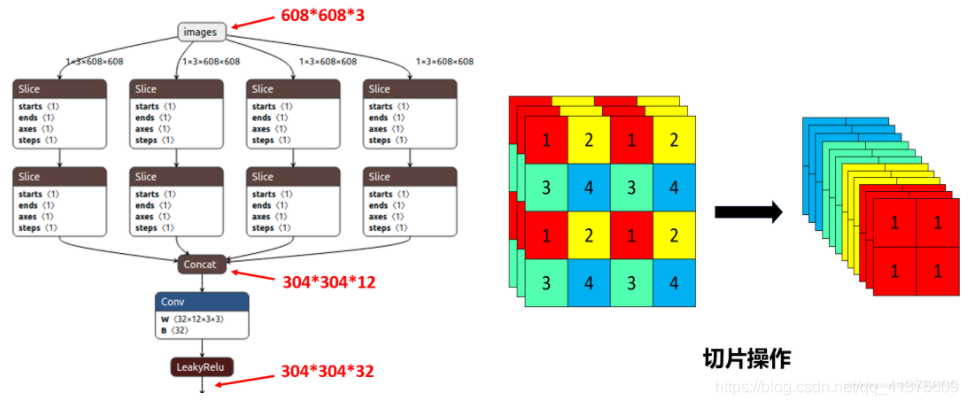
Mosaic数据增强  
②自适应锚定框Auto Learning Bounding Box Anchors  
网络在初始锚框的基础上 基于训练数据 输出预测框，因此初始锚框也是比较重要的一部分。见配置文件\*.yaml， yolov5预设了COCO数据集640×640图像大小的锚定框的尺寸:  
                                
每次训练时，自适应的计算不同训练集中的最佳锚框值。如果觉得计算的锚框效果不是很好，也可以在代码中将自动计算锚框功能关闭。具体操作为train.py中下面一行代码，设置成False  
在这里插入图片描述

③自适应图片缩放  
在常用的目标检测算法中，一般将原始图片统一缩放到一个标准尺寸，再送入检测网络中。Yolo算法中常用416\*416，608\*608等尺寸。因为填充的比较多，会存在信息冗余，所以yolov5对原始图像自适应的添加最少的黑边。

**2）Backbone**

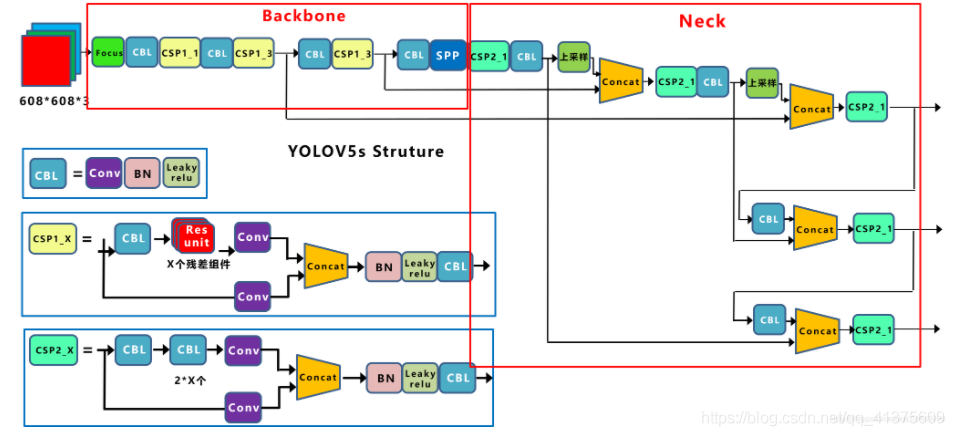
#### ①Focus结构

Yolov5s的结构，原始608\*608\*3的图像输入Focus结构，采用切片操作，先变成304\*304\*12的特征图，再经过一次32个卷积核的卷积操作，最终变成304\*304\*32的特征图。



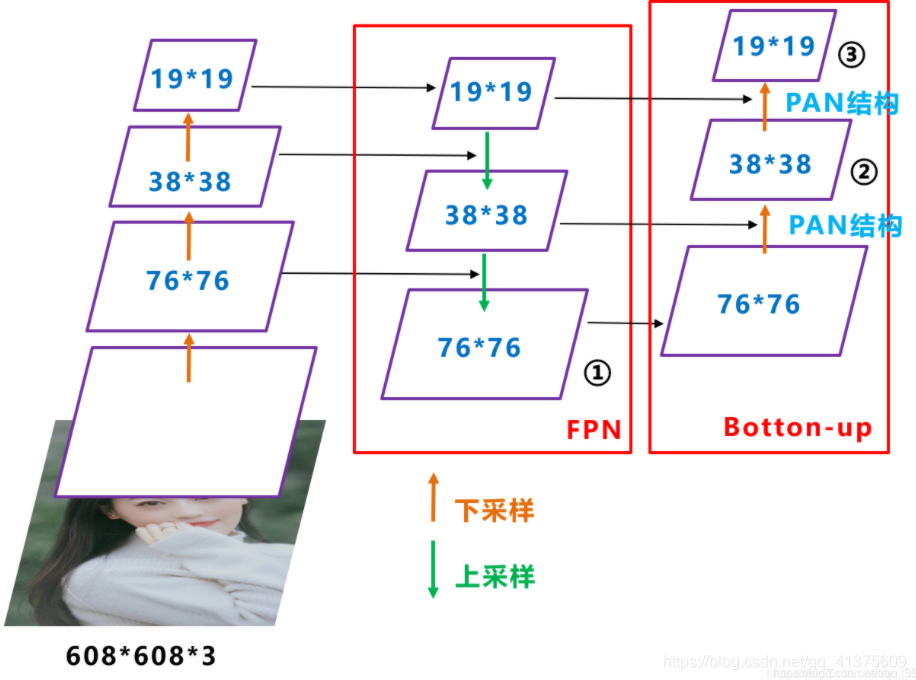
② CSP结构

Yolov5s中设计了两种CSP结构，CSP1\_X结构应用于Backbone主干网络，另一种CSP2\_X结构则应用于Neck中。



3）**Neck**

FPN+PAN，网络特征融合的能力更强



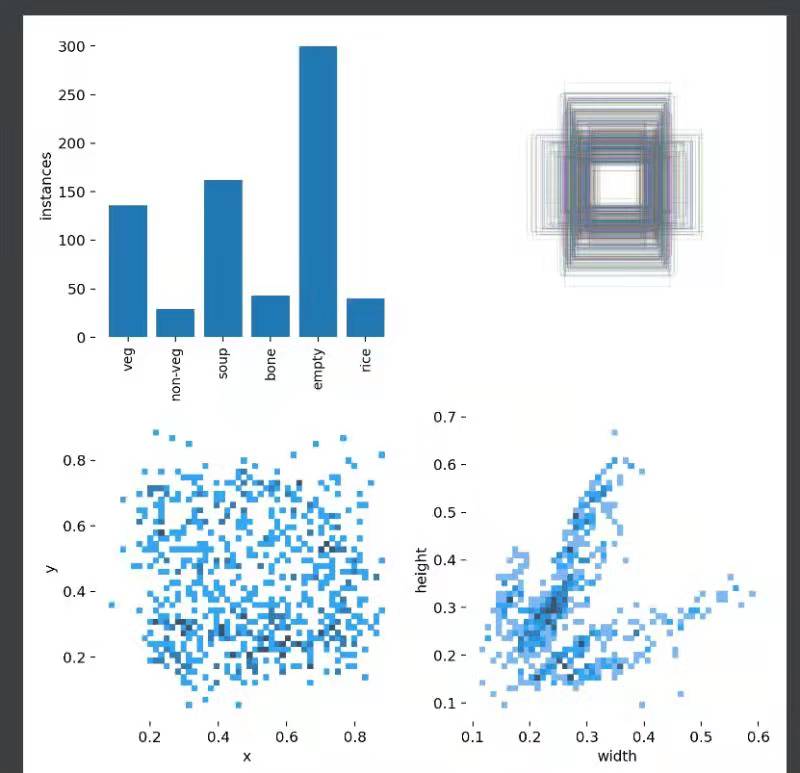
4）**输出端**

①Activation Function  
在 YOLO V5中，中间/隐藏层使用了 Leaky ReLU 激活函数，最后的检测层使用了 Sigmoid 形激活函数  
②nms非极大值抑制  
在同样的参数情况下，将nms中IOU修改成DIOU\_nms。对于一些遮挡重叠的目标，会有一些改进。  
③Optimization Function  
YOLO V5的作者为我们提供了两个优化函数Adam和SGD，并都预设了与之匹配的训练超参数。默认为SGD。  
④Cost Function  
loss = objectness score+class probability score+bounding box regression score  
YOLO V5使用 GIOU Loss作为bounding box的损失。  
YOLO V5使用二进制交叉熵和 Logits 损失函数计算类概率和目标得分的损失。同时我们也可以使用fl\_gamma参数来激活Focal loss计算损失函数。

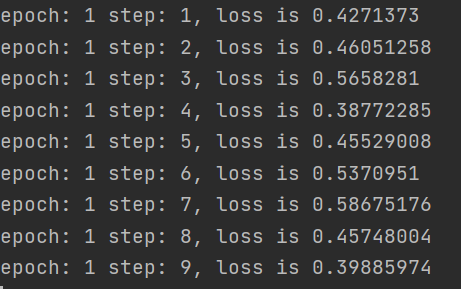
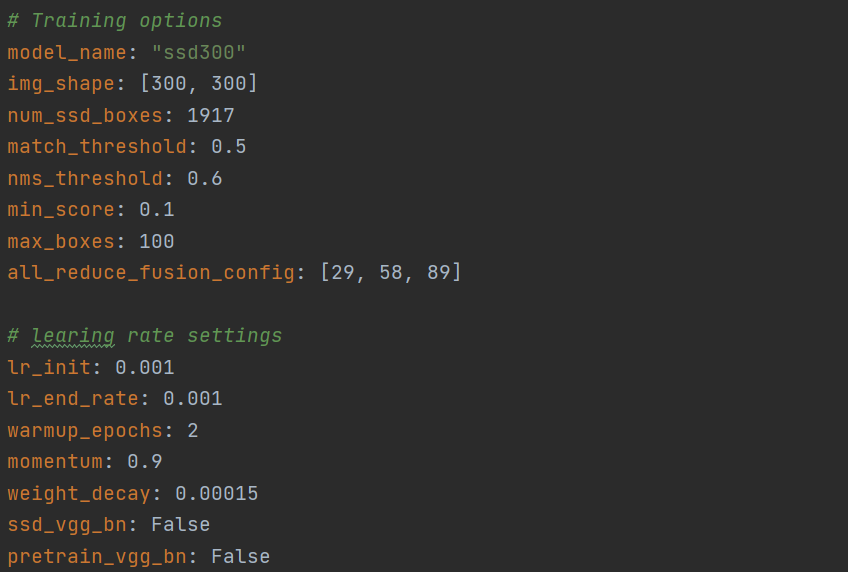
### 模型训练

5.1 SSD模型训练

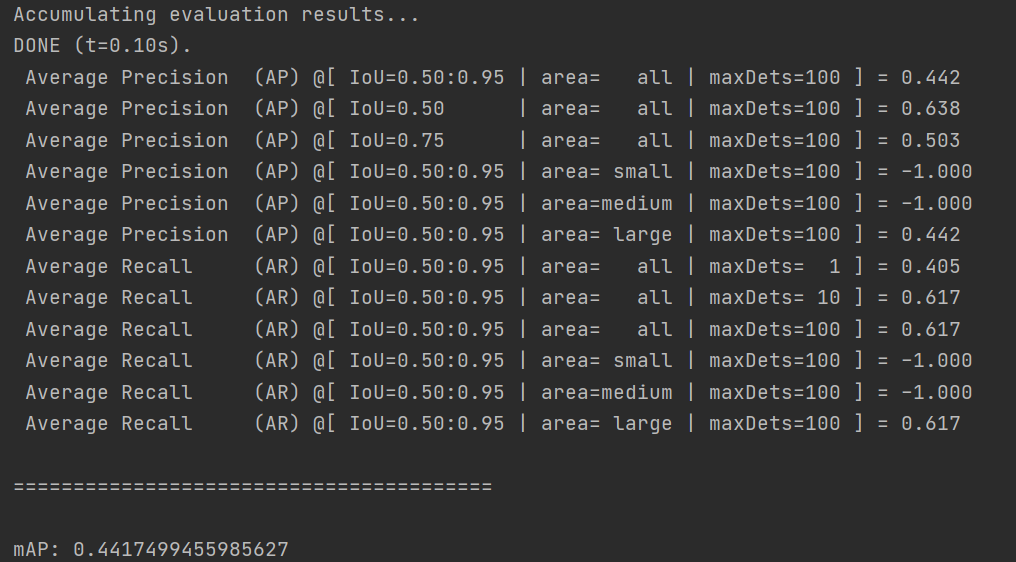
训练集中6种类型的数量主要是空盘居多，其次是蔬菜和汤。边界框的中心点（x,y）呈现随机分布。边界框的宽和高主要分布在width=0.25，height=0.3，说明餐盘大小主要是0.25\*0.3



**训练参数与训练过程如下：**

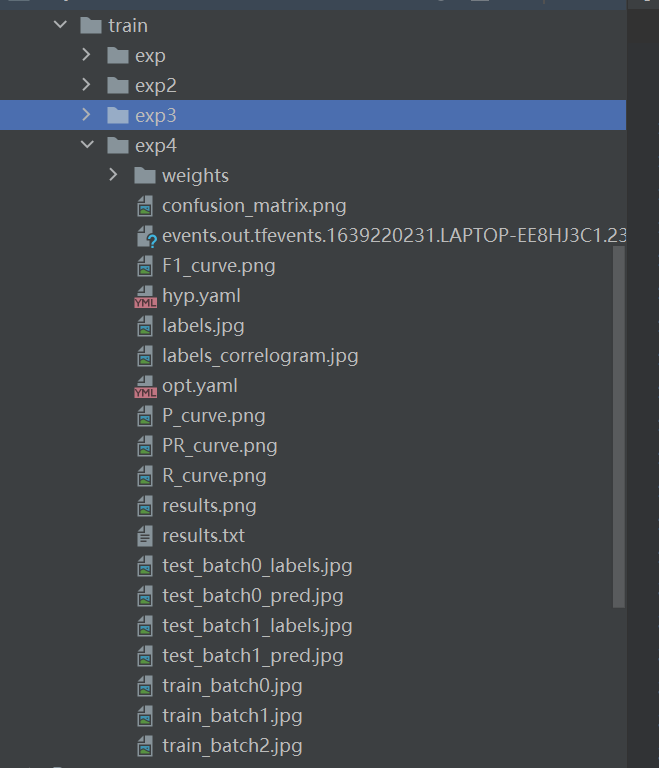


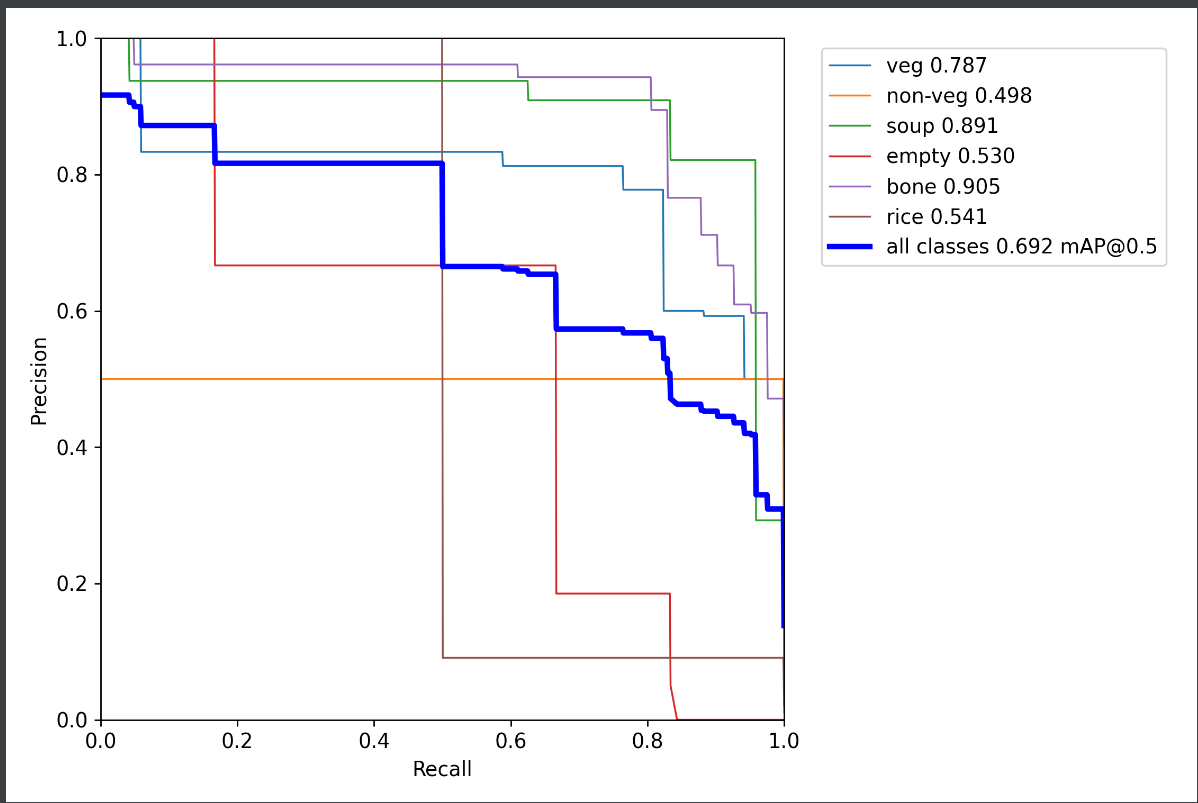
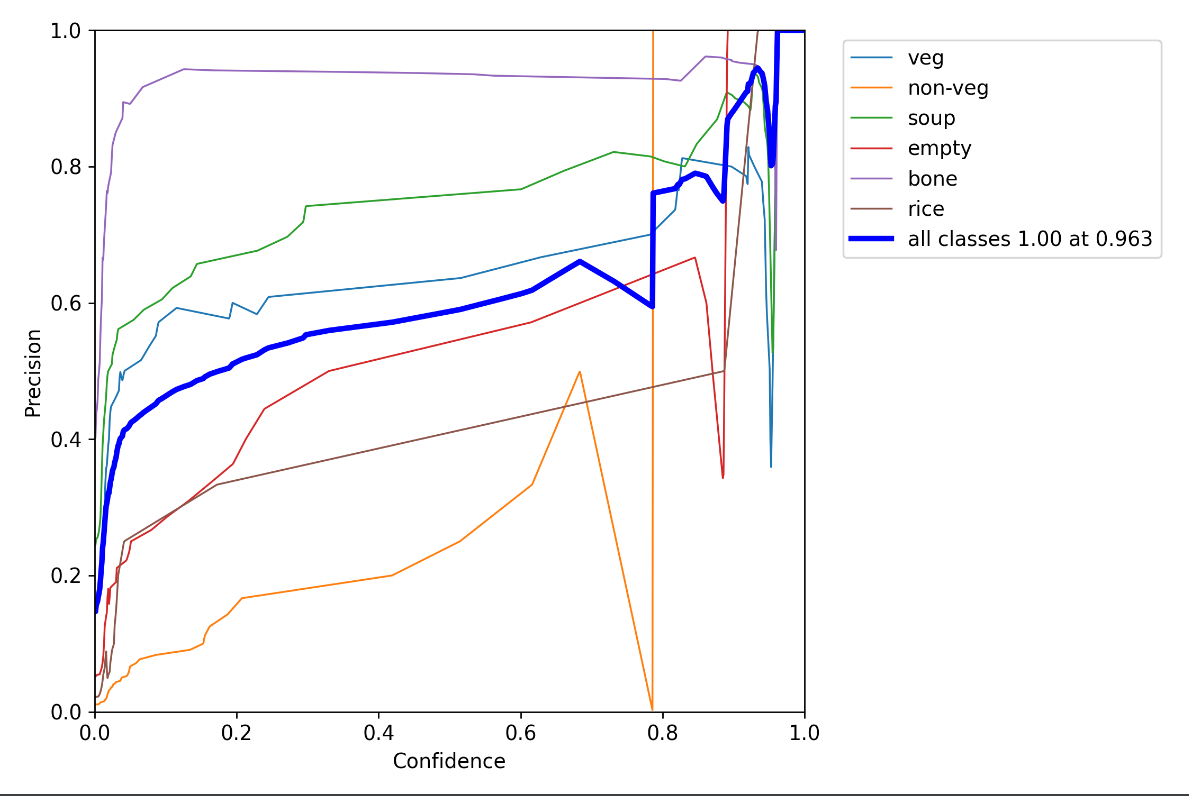
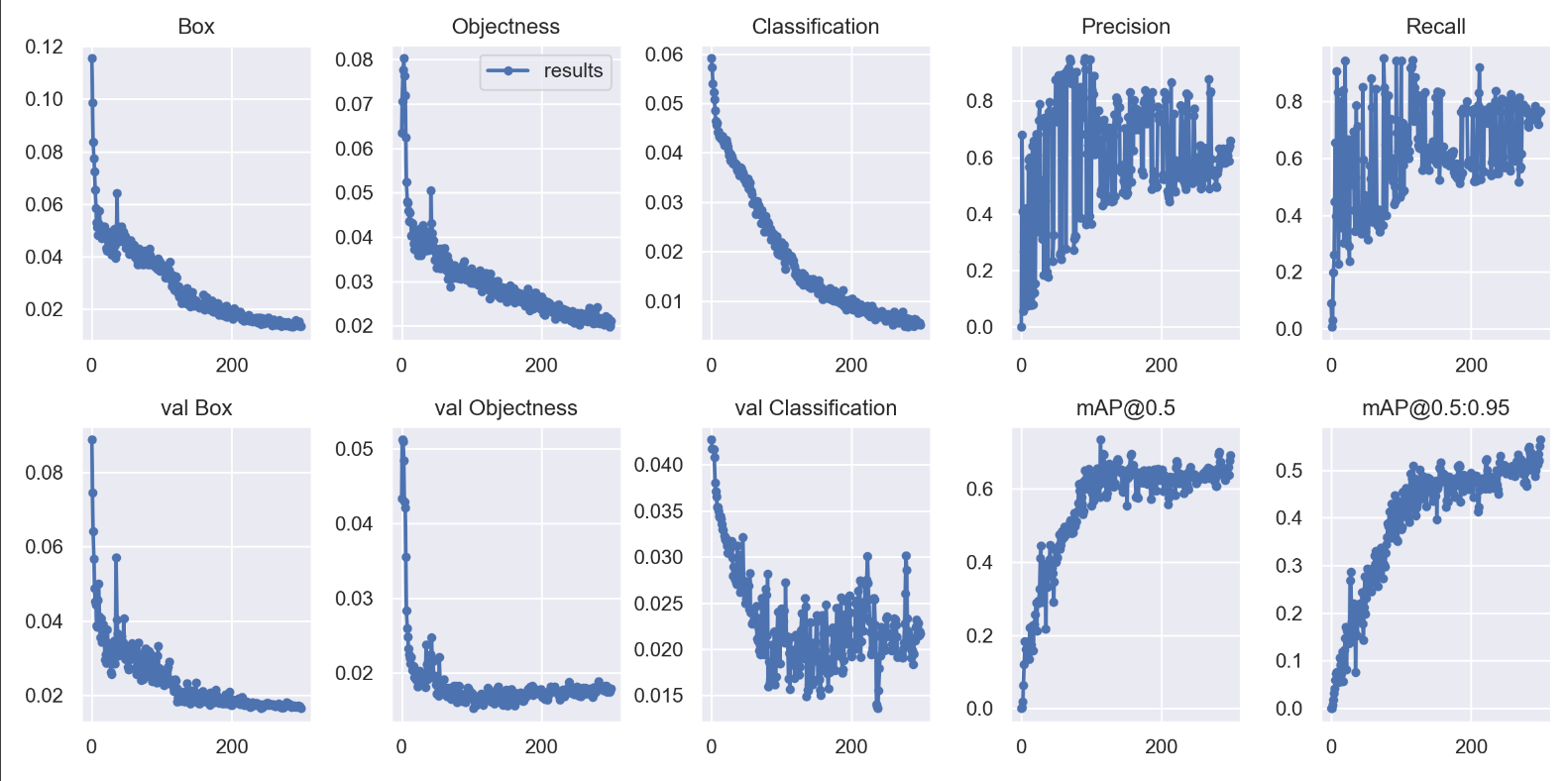
**SSD模型的最终效果与目标检测结果**



5.2 Yolov5模型训练

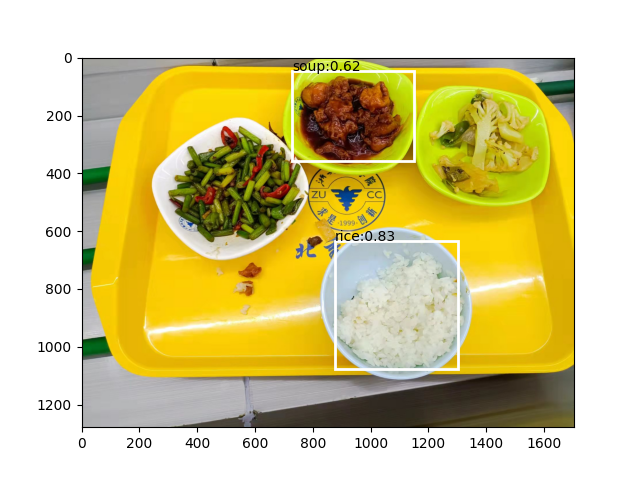
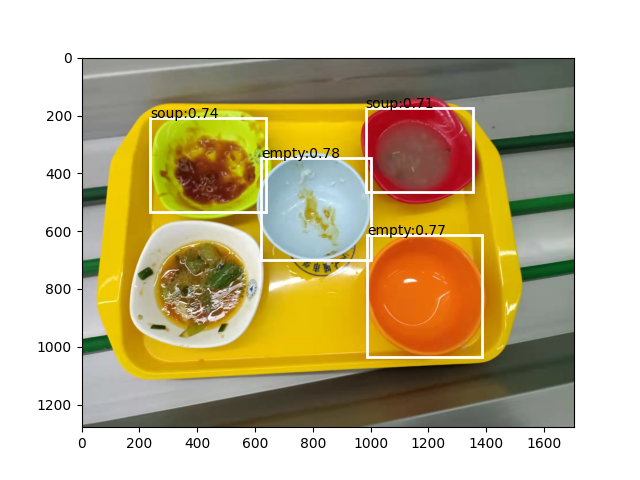
训练输出文件





### 模型评测

6.1 SSD的模型效果



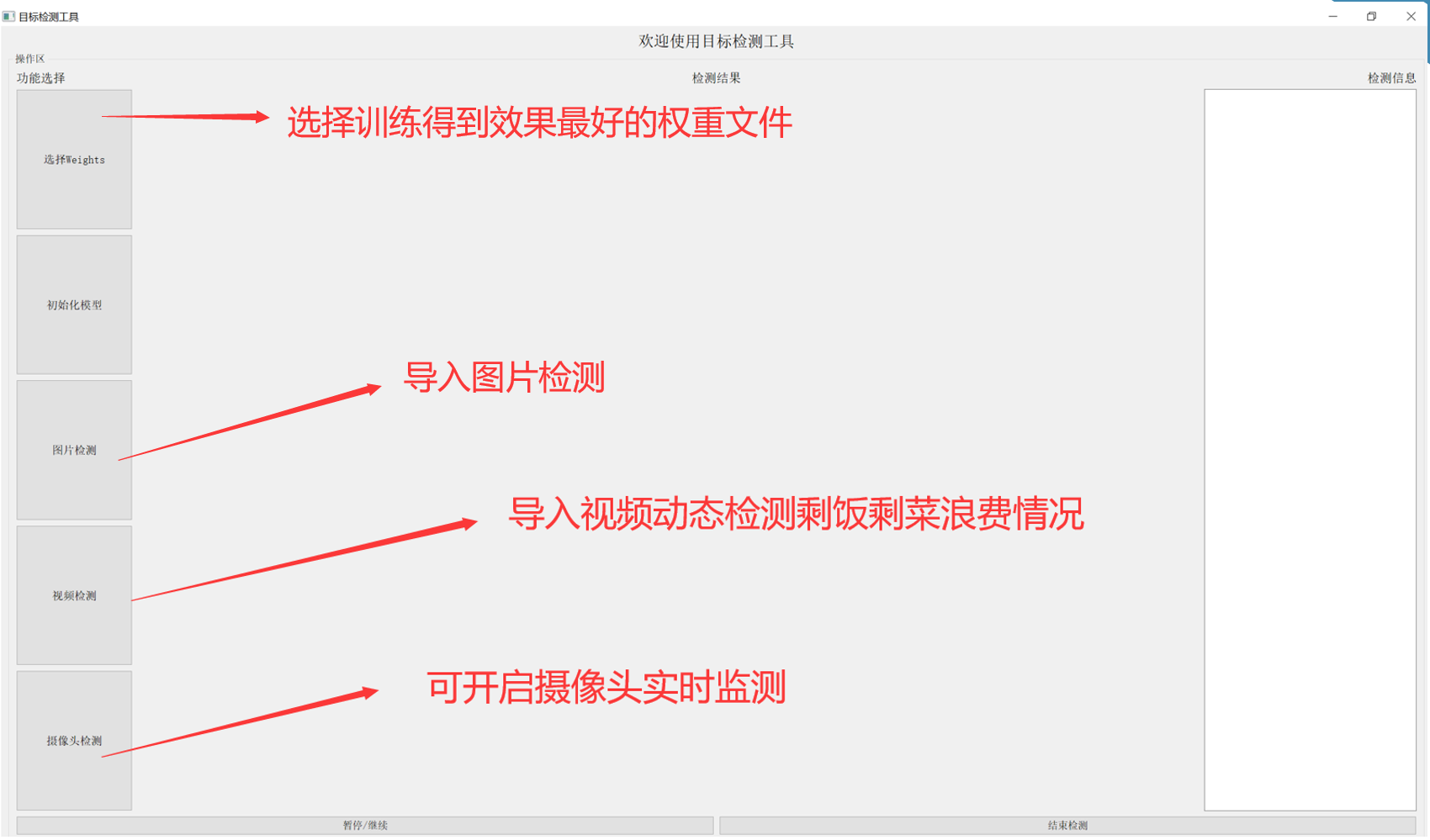
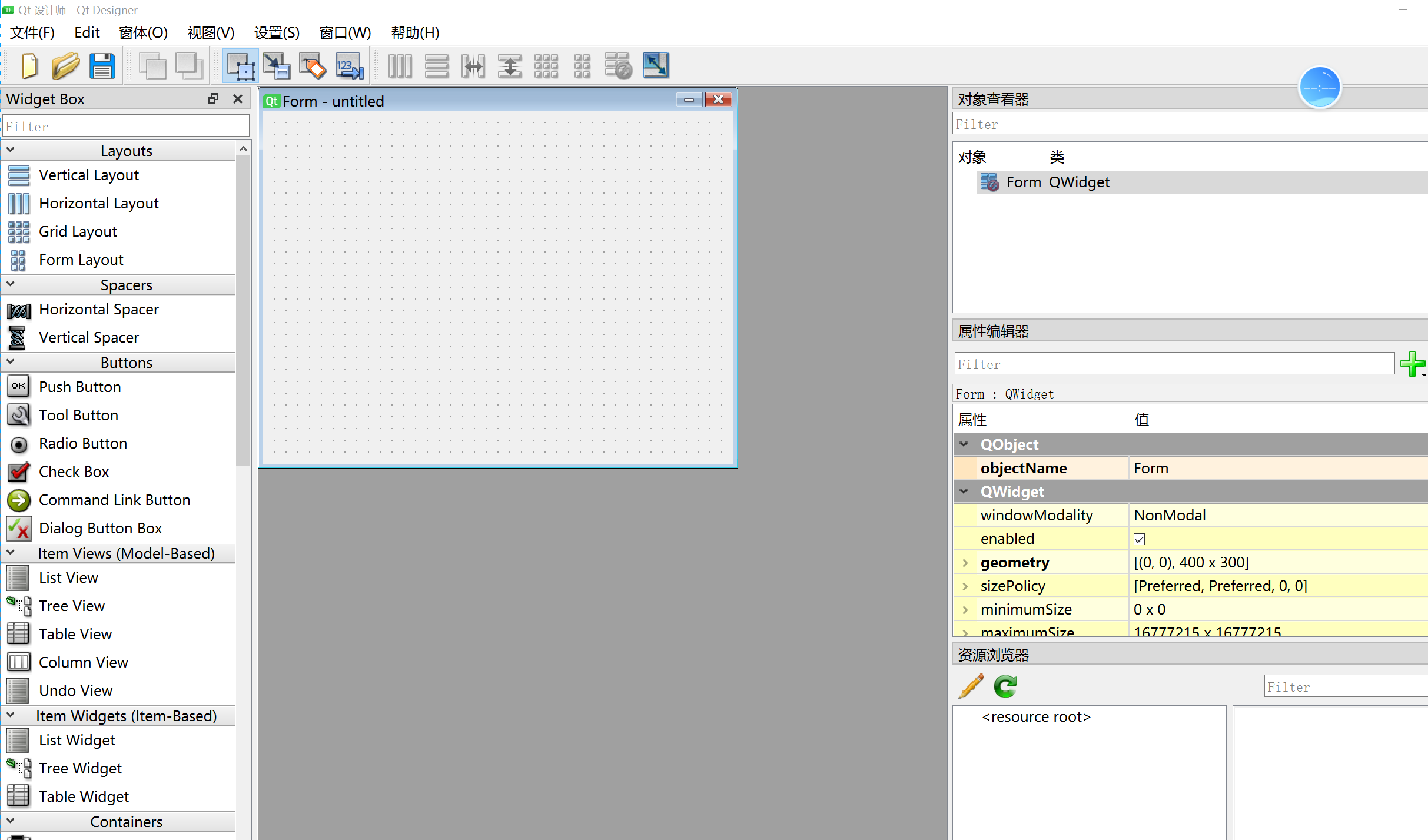
SSD的模型检测效果并不理想，训练也很慢，精确度在0.56~0.78之间，甚至还有一些餐盘无法识别出来，因此我们选用了Yolov5模型重新进行目标检测。

6.2 Yolov5的模型效果



6.3交互界面展示

利用Qt Designer设计界面：



识别效果：



视频检测功能已录制视频展示见文件夹中。

### 大作业提交

* 1. 大作业报告、PPT
  2. 大作业源代码和数据文件

（链接: <https://pan.baidu.com/s/1aikoJo97_e4AISwhhIK-wQ>

提取码：ty31）