

**图像处理与计算机视觉**

**课程设计**

**非法挖山作业车识别方法**

**姓名：朱泠皞 班级：人工智能1班 学号：U2020xxxxx**

**姓名：张凯迪 班级：人工智能1班 学号：U2020xxxxx**

[1.背景描述与问题简介 1](#_Toc_5)

[1.1背景说明 1](#_Toc_7)

[1.2任务描述 1](#_Toc_9)

[1.3数据集说明 2](#_Toc_11)

[1.4评价指标 2](#_Toc_13)

[2.问题分析 4](#_Toc_15)

[2.1数据增强 4](#_Toc_17)

[2.1.1同类数据增强 4](#_Toc_3)

[2.1.2混合数据增强 5](#_Toc_35)

[2.2目标检测模型 7](#_Toc_19)

[2.2.1两阶段目标检测 7](#_Toc_37)

[2.2.2单阶段目标检测 7](#_Toc_39)

[2.3迁移学习 8](#_Toc_21)

[3.实验过程和结果分析 9](#_Toc_23)

[3.1数据处理方法 9](#_Toc_25)

[3.1.1验证集切分 9](#_Toc_41)

[3.1.2标注格式转换 9](#_Toc_43)

[3.1.3数据增强流程 11](#_Toc_45)

[3.2实验条件 13](#_Toc_1)

[3.3模型训练与结果分析 13](#_Toc_29)

[3.3.1 RetinaNet 14](#_Toc_47)

[3.3.2 YOLOv5 16](#_Toc_49)

[3.3.3 YOLOv8 18](#_Toc_51)

[3.4后处理与可视化 20](#_Toc_31)

[4.总结 27](#_Toc_33)

# 1.背景描述与问题简介

## 1.1背景说明

非法挖山作业是一种严重的违法行为，不仅破坏了自然环境和生态平衡，还对人类生产生活造成了极大的危害。为了有效遏制非法挖山作业行为，提高监管效率，保护生态环境，监管部门使用无人机对山体周边进行飞行巡视，并对潜在的挖山作业车进行识别与检测。

为了实现这一目标，一种快速、精准的针对非法作业车的目标检测算法是不可或缺的。在本次计算机视觉课程设计中，我们尝试基于深度学习的方式，设计一种有效的目标检测算法流程，实现上述的任务目标。

## 1.2任务描述

设计一个基于深度学习的目标识别模型，对无人机获取的图像进行检测，识别出作业车，并给出其类型和检测框，识别结果样例如下图所示。

识别目标：

（1）渣土车slagcar

（2）挖掘机excavator

（3）推土机bulldozer

（4）压路机soilcompactor

任务难点：

（1）无人机飞行高度、镜头焦距等参数变化，获取的图像目标尺寸变化大；

（2）作业车目标数量不均匀：渣土车目标多，其他目标少；

（3）作业车类型多样，颜色，规格等因素使得同一类型的目标存在多种变式；

## 1.3数据集说明

作业车训练数据集是一个小型目标检测数据集。整个数据集包括291张图像、标签和类别信息，图像均为无人机拍摄获取。其中渣土车（slagcar）目标最多，压路机（soilcompactor）目标最少。

数据集结构如下：

Dataset

| - images

| | - 6p\_622.jpg

| | - ...

| - labels

| | - 6p\_622.txt

| | - ...

| - classes.txt

/images文件夹中，包括全部291张图像的jpg文件，每张图片的分辨率均为。

/labels文件夹中，包括全部291张图像含有各种目标类别及其回归框坐标的txt文件。对应的jpg图像和txt标注具有相同的文件名。txt文件中的标注为yolo格式。

classes.txt文件记录了所有目标的类别。在本任务中，共有4类。

## 1.4评价指标

**主要的评价指标：识别精度（Precision）和召回率（Recall）**

精确度（Precision）又称查准率，表示预测结果为正例的样本中，实际为正样本的比例。计算公式为：

召回率（Recall）又称查全率，表示预测结果中为正样本中，实际为正的样本数量占全体正样本的比例。计算公式为：

在计算Precision和Recall的过程中，需要使用混淆矩阵。对某一特定类别的分类结果，模型的预测根据正确与否可以抽象为一个二分类机器学习问题。利用二分类问题的混淆矩阵可以得到上述计算需要的变量，以slagcar为例：

TP（True Positive）表示预测结果为正，实际值也为正的样本数量。本问题中表示预测类别为slagcar，实际类别也是slagcar。

TN（True Negative）表示预测结果为负，实际值也为负的样本数量。本问题中表示预测类别为非slagcar，实际类别也是非slagcar。

FP（False Positive）表示预测结果为正，但实际值为负的样本数量。本问题中表示预测类别为slagcar，但实际类别为非slagcar。

FN（False Negative）表示预测结果为负，但实际值为正的样本数量。本问题中表示预测类别为非slagcar，但实际类别为slagcar。

按照上述评价方式，可以对每一类目标的检测性能进行评估。但是这一类评价指标收到手工设定的置信度阈值以及IOU阈值的影响。若希望去除这类超参数的影响，或对模型的整体性能进行综合评价，需要使用更加复杂的评估指标。

**综合评价指标：AP（Average Precision）与mAP（mean Average Precision）**

AP用于评估算法在图像中识别和定位某一特定类别对象的能力。AP的计算需要使用Precision-Recall曲线，该曲线用于衡量不同置信度阈值下Precision和Recall之间的关系。AP的值等于此曲线上不同召回率下的精度值的平均值，等价于计算PR曲线下方的面积。

本次实验采用COCO评价指标计算AP。对IOU阈值在0.5至0.95的区间上，每隔0.05生成一条PR曲线，对每条PR曲线采样100个Recall值对应的Precision，将得到的1000个Precision进行算数平均，即得到单一类别对应的AP。

当需要综合评估多类别下的模型性能时，使用mAP指标。mAP等于所有类别AP值的算数平均。

# 2.问题分析

## 2.1数据增强

考虑到训练数据集规模较小，为了获得更好的泛化性能，避免过拟合现象，采用合适的数据增强是十分必要的。下面总结了计算机视觉任务中常用的数据增强操作，在实验中，我们将根据下面的基本操作，构造一个适用于本次任务的数据增强流程。

#### 2.1.1同类数据增强

**几何空间增强**

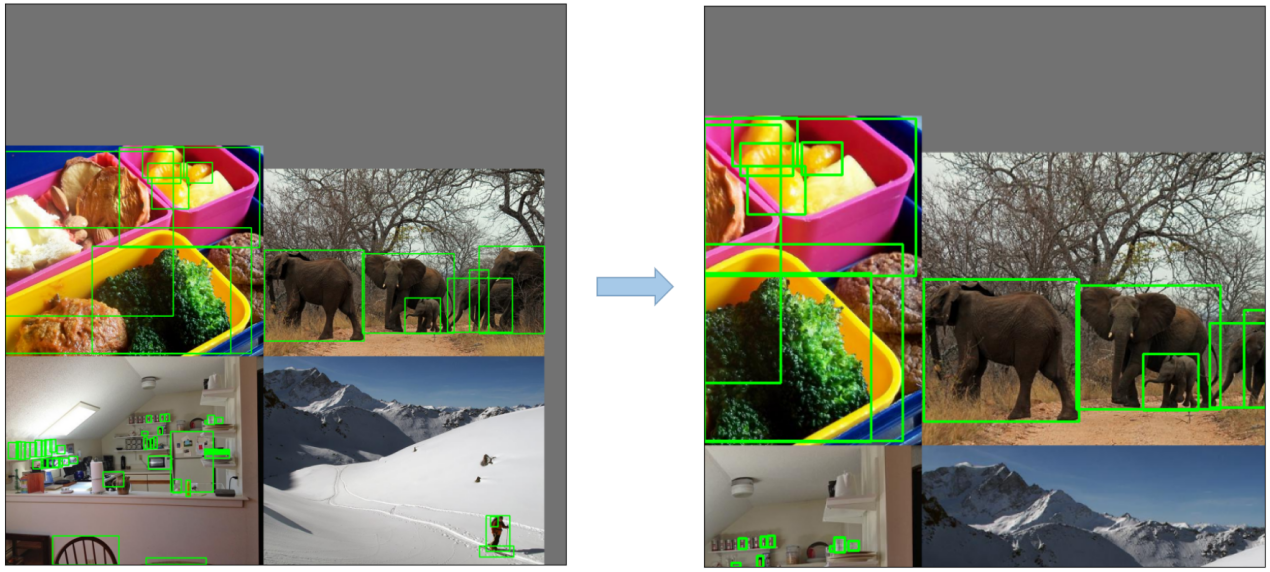
1.随机翻转（RandomFlip）

以一个给定的概率随机对输入图像进行水平或垂直翻转。由于现实生活中水平对称的场景出现概率大于上下翻转对称的概率，因此一般使用水平方向翻转。



2.随机裁剪（RandomCrop）

在图片的随机位置，按照指定的大小进行裁剪。这种数据增强的方式能够在保留图像比例的基础上，移动图片上各区域在图片上的位置。



3.旋转（Rotate）

以一个给定的概率随机对输入图像进行旋转，旋转角度为之间的随机值。

4.平移（Translate）

以一个给定的概率随机对输入图像进行平移，平移的相对值为之间的随机值。

**颜色空间增强**

1.色彩抖动（ColorJitter）

在一定范围内，对图像的亮度（Brightness）、对比度（Contrast）、饱和度（Saturation）和色相（Hue）进行随机变换，从而模拟真实拍摄中不同灯光环境等条件的变化。

2.随机灰度化（RandomGrayscale）

按照一定概率，将图片转变为灰度图。这种增强方法消除了颜色的影响，在特定场景有所应用。

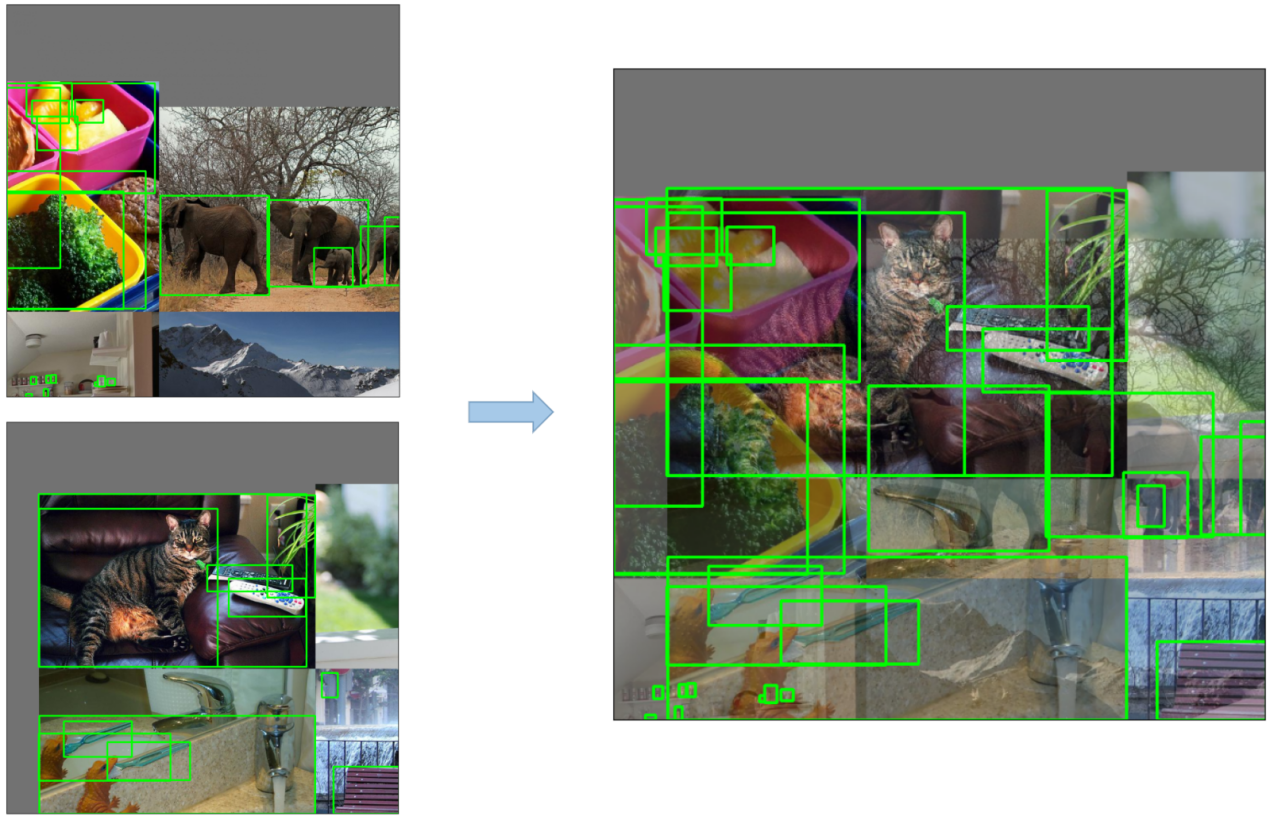
3.随机光照变换（Lighting）

首先在训练数据集中对所有图像的像素进行 PCA（主成分分析），从而获得 RGB 空间中的特征值和特征向量，代表了光照强度对图片像素的影响。之后沿着此特征向量的方向对每个像素的颜色值进行随机幅度的加减。

#### 2.1.2混合数据增强

1.MixUp

将两张图片直接进行线性组合，对应的标签也进行线性组合。其中叠加在背景图片上的图片强度按照分布进行随机采样，为超参数。



2.CutMix

随机将图像的一部分区域裁切掉，然后将这部分区域随机填充训练集中的其他图像的区域像素值，随机填充区域大小的比例按照分布进行随机采样，为超参数。

3.Mosaic

在训练数据集中随机采样4张图片，将四张图片按比例resize后拼接成一张新的图像。此方法在极大丰富了检测物体的背景的同时，等价于增大了训练的batch size。



## 2.2目标检测模型

#### 2.2.1两阶段目标检测

两阶段目标检测是常见的目标检测网络结构，这种结构将目标检测任务分为两个阶段：“区域生成”和“区域验证”。

在第一阶段，模型首先通过滑动窗口或卷积神经网络（CNN）等手段生成一系列候选区域（Region proposals），这些区域可能包含图像中的目标对象。生成的候选区域数量通常非常庞大，可能达到数十万甚至数百万个。

在第二阶段，模型会对这些生成的候选区域进行验证，以确定其中哪些区域确实包含目标对象。这一阶段通常采用一种名为 RPN（Reigion Proposal Net）的机制，对每个候选区域进行分类和定位精度的评估。如果一个候选区域被判定为目标对象，则将其保留下来；否则，将其剔除。保留下来的候选区域会分别经过Classification Head和Regression Head，得到最终的预测类别以及预测框坐标。

两阶段目标检测模型的优点在于，它能够有效地减少误检和漏检的情况。通过第一阶段的区域生成步骤，可以大大缩小候选区域的数量，从而降低第二阶段验证的复杂性。同时，两阶段目标检测模型还可以采用更复杂的分类器和定位器来提高检测精度。

然而两阶段目标检测模型的缺点也是显著的。首先，它需要两个阶段的计算过程，导致较高的模型复杂度。其次，区域生成阶段的网络训练是困难的。通常需要大量的人工标注数据进行训练，这不仅增加了数据采集和标注的成本，而且也限制了模型的泛化能力。此外，数量庞大的候选区域全部需要通过区域验证阶段的计算，这使得模型的计算量较大，推理速度较慢。

#### 2.2.2单阶段目标检测

单阶段目标检测模型是一种直接在图像上预测目标对象位置和类别的目标检测方法。与两阶段目标检测模型不同，单阶段目标检测模型不需要生成候选区域，而是直接对整个图像进行分类和定位。

单阶段目标检测模型的优点在于，它只有一个阶段的计算过程，相对于两阶段目标检测模型，时间复杂度更低。此外，由于不需要生成候选区域，单阶段目标检测模型可以避免人工标注数据的需求，从而降低了数据采集和标注的成本，同时也提高了模型的泛化能力。

然而，单阶段目标检测模型也存在一些缺点。由于没有对候选区域进行预先筛选，单阶段目标检测模型需要面对大量的背景区域，这会降低模型的检测精度。此外，单阶段目标检测模型通常需要采用更复杂的网络结构和训练方法来提高检测性能，这也会增加模型的设计和训练成本。

综合实际应用情况，以及两种模型的优劣，我们决定在本次实验中基于单阶段目标检测模型进行实验。

实验中使用到的单阶段目标检测模型有：Retinanet，YOLOv5，YOLOv8。

## 2.3迁移学习

迁移学习致力于将某个领域或任务上学习到的知识或模式应用到不同但相关的领域或问题中，从相关领域中迁移标注数据或者知识结构、完成或改进目标领域或任务的学习效果。具体而言，迁移学习是一种将已训练好的模型（预训练模型）参数迁移到新的模型来帮助新模型训练的方法。

迁移学习的优势在于：（1）在微调之前，模型的初始性能更高。（2）在大多数情况下，模型不需要大量数据就能得到更好的性能。这是由于模型已经事先加载了在主干任务上性能优异的预训练权重，这一套权重已经使得模型处于一个较好的初始状态。在接下来的模型微调的过程中，只需要输入少量目标问题的数据，模型就能够在综合主干问题的信息的基础上，迁移到新的目标问题上。相较于从头训练一个新模型，迁移学习不仅能在更少数据的基础上获得更优的性能，同时还能大大节省训练资源和训练时间。

在本实验中，考虑到可用训练数据量规模较小（仅有291张训练图像，数百个目标实例）从头训练一个新的模型必然陷入过拟合中，因此我们采用迁移学习的方式，向目标模型加载COCO数据集预训练权重，在其基础上以迁移学习的思想训练新的目标检测模型。

# 3.实验过程和结果分析

## 3.1数据处理方法

#### 3.1.1验证集切分

考虑到数据集并未给出测试集，我们必须从训练集中切分出一定量的数据作为验证集，在训练过程中对模型性能进行评估。我们规定训练集包括250张图像，验证集包括41张图像，训练集与验证集的比例为。

验证集的41张图像使用Python random.choice()函数随机选取，以尽量保证训练集和验证集的数据分布一致性。

对验证集进行数据统计，得到验证集中每一类目标的数量如下：

|  |  |  |
| --- | --- | --- |
| Class | Images | Instances |
| All | 41 | 197 |
| slagcar | 41 | 167 |
| excavator | 41 | 26 |
| bulldozer | 41 | 3 |
| soilcompactor | 41 | 1 |

#### 3.1.2标注格式转换

标注的默认格式为YOLO格式，其基本格式如下：

1. 1 0.201823 0.380093 0.143229 0.154630
2. 1 0.347656 0.412037 0.143229 0.100000
3. 1 0.394271 0.309259 0.143750 0.101852
4. 1 0.498177 0.223611 0.146354 0.125000
5. 1 0.559375 0.308796 0.140625 0.112037
6. 1 0.497135 0.422685 0.102604 0.147222
7. 1 0.237760 0.496759 0.142187 0.106481

txt文件中的每一行表示一个Instance，其中第一个元素代表Instance的类别标签，后续4个元素表示目标框的坐标。坐标格式为中心点x-y坐标——目标框宽-高格式，且均相对图像的宽、高归一化。

为了使用YOLO之外的模型，必须将数据集标注格式转化为COCO格式。COCO格式标注使用json文件，以字典的形式存储图像、标注以及目标类别的相关信息。全部的标注保存在单个文件中。

COCO标注的关键字众多，其中一部分是非必须的。必要的COCO标注关键字如下：

1. {
2. "images": [image],
3. "annotations": [annotation],
4. "categories": [category]
5. }
7. image = {
8. "id": int,
9. "width": int,
10. "height": int,
11. "file\_name": str,
12. }
14. annotation = {
15. "id": int,
16. "image\_id": int,
17. "category\_id": int,
18. "segmentation": RLE **or** [polygon],
19. "area": float,
20. "bbox": [x,y,width,height], # (x, y) 为 bbox 左上角的坐标
21. "iscrowd": 0 **or** 1,
22. }
24. categories = [{
25. "id": int,
26. "name": str,
27. "supercategory": str,
28. }]

要实现YOLO格式到COCO格式的转换，只需要改变回归框的坐标格式，记录在annotation.bbox字段，同时计算出回归框的绝对面积，记录在annotation. area字段即可。

首先，使用opencv.imread()函数读取图像，得到绝对宽高和。由此得到cxcywh格式的回归框绝对坐标：

按照几何关系可以计算得到回归框左上角坐标：

相应地，计算出回归框的面积：

如果需要使用分割-检测一体模型（如Mask-RCNN），还可以相应的填写annotation.segmentation字段。由于实例分割不属于此次实验的范围，因此不再讨论如何从回归框计算segmentation annotation。

#### 3.1.3数据增强流程

由于数据量较少，样本量极不平衡，因此对输入图像进行一定程度上的数据增强是十分有必要的，我们参考了Soft Teacher，RTMDet等经典模型使用的数据增强流程，提出了下面的数据增强pipeline：

step1：Mosaic，随机抽取训练集中的几张图片拼接成一张大型图片，每张用于拼接的图片保留原始长宽比例，缩放至（每张图片的长和宽不超过1280，下同）。

step2：RandomResize，将拼接后的图像按原比例缩放至，保持一定的对小目标的分辨率。

step3：RandomCrop，从图像中按比例随机裁切大小的图像

step4：RandomHSVAug，在HSV颜色空间内以0.5的概率对图像进行随机扰动，实现颜色空间上的数据扩增

step5：RandomFlip，按照0.5的概率对图像进行随机水平翻转（因为上下翻转的图像在实际情况中几乎不会出现）。

step6：Pad，将比例不一的图像使用特定像素补全成的正方形图像

原始图像顺序经过上述6个步骤后，既完成了数据增强，同时获得了长宽比例与分辨率一致的图像，可以直接输入模型进行训练。我们相信使用上述的数据增强pipeline，可以使模型在原有数据的基础上，观察到更多增强情况下的图像实例，在保证与训练集同分布数据上的性能的同时获得较强的泛化性能。



经过yolov5后第一个batch数据增强得到的效果

我们首先使用结构较为简单的单阶段检测器RetinaNet验证上述自定义增强方案的有效性。此后，按照实验环境的要求，在性能和实时性有优势的YOLOv5、YOLOv8模型上验证方案的泛化性能。我们使用YOLOv8模型作为最终的检测器。

## 3.2实验条件

实验环境：

（1）**MMDetection for RetinaNet**

Ubuntu18.04

CUDA version11.7，Python3.8，Pytorch1.9.1，cudatoolkit11.1

Nvidia Titan XP 12G×2

（2）**YOLO Environment for v5 and v8**

Ubuntu22.04LTS

CUDA version:12.2，Python3.8，Pytorch1.9.1，cudatoolkit11.1

Nvidia Titan X

依赖包：

gitpython

matplotlib

numpy

opencv-python

Pillow

PyYAML

scipy

tqdm

ultralytics

pandas

setuptools

若希望使用MMdetection，还需额外安装MIM、MMEngine、MMCV，然后通过https://github.com/open-mmlab/mmdetection.git使用源码安装MMdetection

## 3.3模型训练与结果分析

#### 3.3.1 RetinaNet

**模型简介**

RetinaNet是2018年提出的单阶段目标检测模型，其主要关注于单阶段目标检测中正负样本极度不平衡问题。这个问题在目标检测中广泛存在，但对单阶段目标检测模型的影响更为突出。

两阶段检测器之所以能在一定程度上减轻类别不平衡问题，是因为模型的两阶段级联结构以及启发式采样。在区域建议阶段，RPN能够快速将候选对象的数量缩小至1~2k的较小数目，此过程天然的过滤掉了大部分的背景样本。在区域验证部分，启发式采样、固定的前景/背景比例、在线难样本挖掘等操作都保持了前景和背景的相对平衡。

相比之下，单阶段检测器必须处理在图像中定期采样的一组数量庞大的候选对象位置，这些位置密集地覆盖空间位置、尺度、纵横比等。这是单阶段检测器的精度不及两阶段检测器的主要原因。

RetinaNet提出新的Focal loss函数用于解决类别不平衡问题。Focal loss通过对减少简单样本的权重来平衡样本对模型的影响。这意味着即使简单样本的数量十分庞大，它们对总的损失函数的共享并不会很显著。这使得模型侧重于训练稀疏的难样本。

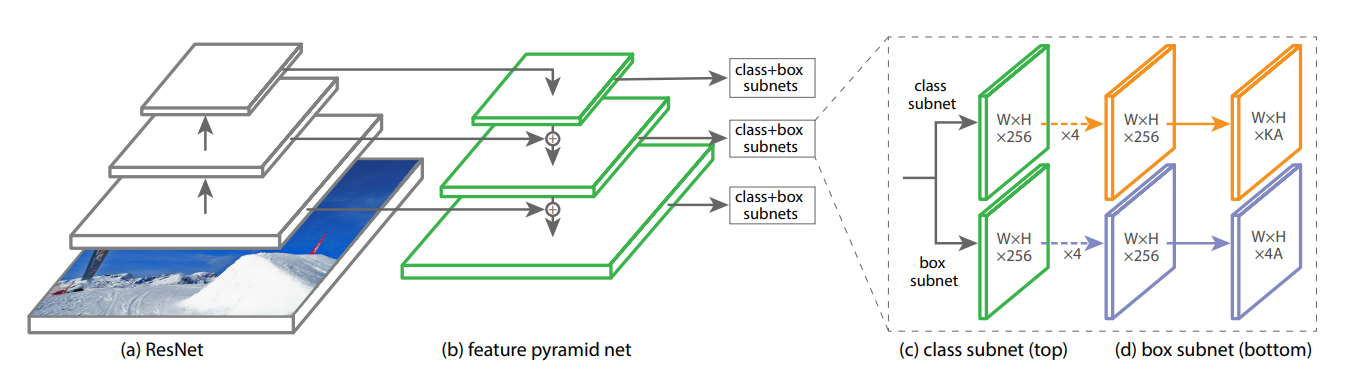
Focal loss改进自Cross Entropy loss。以二分类为例，Cross Entropy loss表示为：

Focal loss在其基础上增加了一个调整因子（modulation factor），用于降低高置信度样本的损失，表达式为：

记，则的值越大，说明样本越容易被分类正确

Focal loss具有两个性质：（1）当的值较大时，调整因子接近于0，而当的值较小时，调质因子接近1。这意味着容易分类样本的损失被减少了权重，等价于将分类不准确样本的权重增加了。（2）参数平滑的调整的权重的下降率，当时，Focal loss等价于Cross Entropy。

RetinaNet的网络结构的设计尽可能简单，在保证运算效率的同时使用Focal loss提升网络的性能。模型由ResNet backbone，FPN neck，以及参数不共享的cls head和reghead组成。



**基本性能测试**

RetinaNet的实验基于MMDetection框架展开。首先使用默认的模型微调配置，仅加载COCO预训练权重，不进行任何超参数改动，训练一组参考性能结果。

实验使用2张Nvidia Titan XP显卡，batch size=4，max\_epoch=24；

优化器选择Momentum SGD，momentum=0.9，learning\_rate=0.01，weight\_decay=1e-4，前500 iteration使用线性学习率预热，训练中使用阶梯学习率更新策略，在16、22epoch分别对学习率进行缩减，缩减比率gamma=0.1；

图像采用轻度数据增强，首先将图像resize至分辨率，然后以0.5的概率随机进行左右翻转。

基础实验在验证集上的性能表现如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| category | mAP | mAP50 | mAP75 | mAPs | mAPm | mAPl |
| all | 0.636 | 0.889 | 0.749 | 0.0 | 0.435 | 0.769 |
| slagcar | 0.541 | 0.845 | 0.622 | 0.0 | 0.394 | 0.747 |
| excavator | 0.42 | 0.711 | 0.376 | - | 0.229 | 0.66 |
| bulldozer | 0.683 | 1.0 | 1.0 | - | 0.683 | - |
| soilcompactor | 0.9 | 1.0 | 1.0 | - | - | 0.9 |

**自定义流程实验**

自定义流程采用上述的数据增强方案，相应地需要对训练超参数进行适当改动。实验同样加载COCO预训练权重。

实验使用2张Nvidia Titan XP显卡，batch size=4，max\_epoch=70；

优化器选择Momentum SGD，momentum=0.9，learning\_rate=0.01，weight\_decay=1e-4，前10 epoch使用线性学习率预热，初始值0.001，训练中使用阶梯学习率更新策略，在20、40、60epoch分别对学习率进行缩减，缩减比率gamma=0.5；

图像使用3.1.3节中的数据增强流水线。

自定义实验的性能表现如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| category | mAP | mAP50 | mAP75 | mAPs | mAPm | mAPl |
| all | 0.677 | 0.952 | 0.781 | 0.0 | 0.511 | 0.776 |
| slagcar | 0.608 | 0.932 | 0.734 | 0.0 | 0.517 | 0.733 |
| excavator | 0.485 | 0.875 | 0.39 | - | 0.299 | 0.696 |
| bulldozer | 0.717 | 1.0 | 1.0 | - | 0.717 | - |
| soilcompactor | 0.9 | 1.0 | 1.0 | - | - | 0.9 |

可知，使用自定义数据增强流程后，RetinaNet在验证集上的性能有明显的提升，在所有类别上的mAP性能都一定程度的改善。

为了验证数据增强流程的泛化性能，我们在实时性更强，理论性能更优的YOLOv5模型和YOLOv8模型上继续实验。

#### 3.3.2 YOLOv5

**模型简介**

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。主要的改进思路有以下几点：

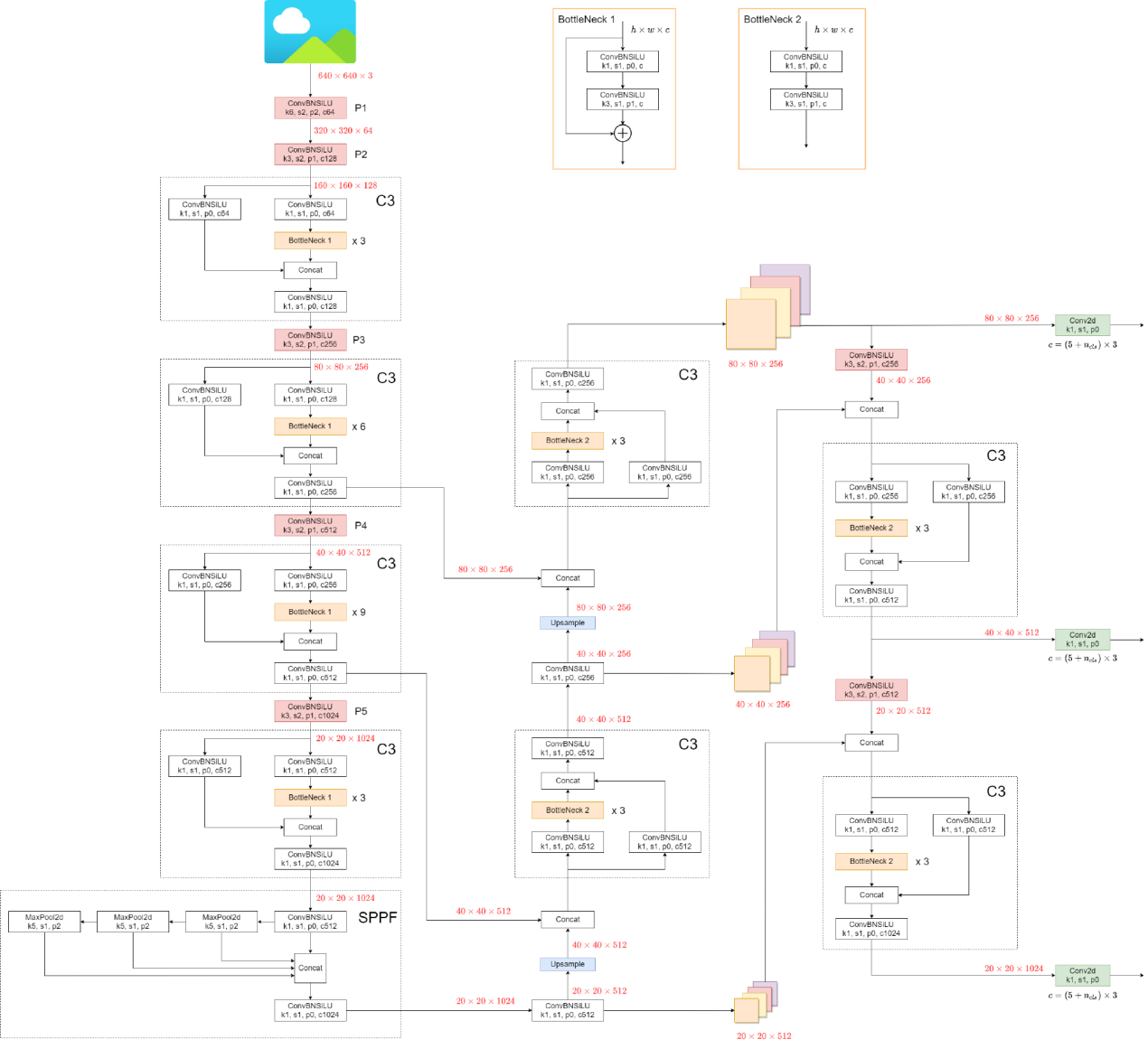
1.输入端：在模型训练阶段，提出了一些改进思路，主要包括Mosaic数据增强、自适应锚框计算、自适应图片缩放；

2.基准网络：融合其他检测算法中的一些新思路，主要包括：Focus结构与CSP结构；

3.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；

4.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。

Yolov5的网络架构如下图所示：



Yolov5l框架

对于一个目标检测算法而言，我们通常可以将其划分为4个通用的模块，具体包括：输入端、基准网络、Neck网络与Head输出端，对应于上图中的4个红 色模块。 YOLOv5算法具有4个版本，具体包括：YOLOv5s、YOLOv5m、YOLOv5l、YOLOv5x四种，以下以Yolov5l作为例子进行讲解。

**输入端**：输入端表示输入的图片。该网络的输入图像大小为640\*640，该阶段通常包含一个图像预处理阶段，即将输入图像缩放到网络的输入大小，并进行归一化等操作。在网络训练阶段， YOLOv5使用Mosaic等数据增强操作提升模型的训练速度和网络的精度；并提出了一种自适应锚框计算与自适应图片缩放方法。

**基准网络**：基准网络通常是一些性能优异的分类器种的网络，该模块用来提取一些通用的特征表示。 YOLOv5中不仅使用了CSPDarknet53结构，而且使用了Focus结构作为基准网络。

**Neck网络**：Neck网络通常位于基准网络和头网络的中间位置，利用它可以进一步提升特征的多样性及鲁棒性。虽然YOLOv5同样用到了SPP模块、FPN+PAN模块，但是实现的细节有些不同。

**Head输出层**：Head用来完成目标检测结果的输出。针对不同的检测算法，输出端的分支个数不尽相同，通常包含一个分类分支和一个回归分支。YOLOv4利用GIOU\_Loss来代替Smooth L1 Loss函数，从而进一步提升算法的检测精度。

**实验**

YOLOv5是基于Ultralytics框架的一个模型，Ultralytics被构建为用于训练对象检测、实例分割和图像分类模型的统一框架。本文主要测试了不同参数条件下的训练效果。

本实验使用一张Titan X进行训练，选择预训练模型YOLOv5m6.pt，采用YOLOv5m6的框架。epochs=100，bach-size=8，imgsz=1280即将图像resize到1280\*1280。

优化器选择Momentum SGD，momentum=0.937，learning\_rate=0.01，weight\_decay=0.0005，前3 epoch使用线性学习率预热，初始值0.1，最后一轮学习率为0.001。

图像使用3.1.3节中的数据增强流水线。

自定义实验的性能表现如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| category | Images | Instances | Precision | Recall | mAP50 | mAP50-95 |
| all | 41 | 197 | 0.973 | 0.433 | 0.749 | 0.566 |
| slagcar | 41 | 167 | 0.937 | 0.888 | 0.961 | 0.676 |
| excavator | 41 | 26 | 0.956 | 0.845 | 0.892 | 0.487 |
| bulldozer | 41 | 3 | 1 | 0 | 0.149 | 0.107 |
| soilcompactor | 41 | 1 | 1 | 0 | 0.995 | 0.995 |

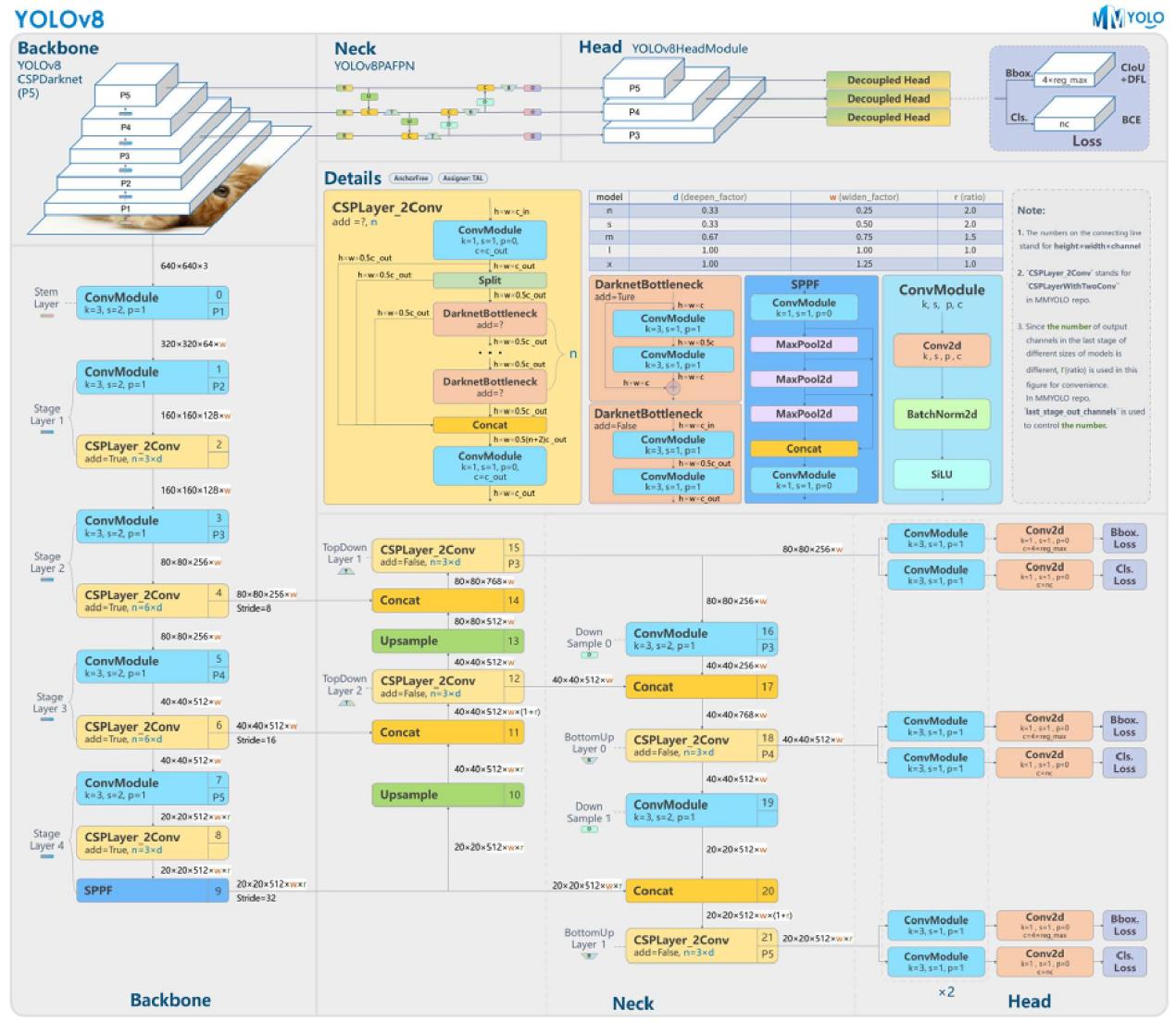
#### 3.3.3 YOLOv8

**模型简介**

YOLOv8 是来自 Ultralytics 的最新的基于 YOLO 的对象检测模型系列，提供最先进的性能。利用以前的 YOLO 版本， YOLOv8 模型更快、更准确 ，同时为训练模型提供统一框架，可以执行物体检测、实例分割以及图像分类三种任务。

Yolov8模型中有5个模型用于检测，分割和分类，分别为Yolov8n，Yolov8s，Yolov8m，Yolov8l和Yolov8x。

Yolov8的网络结构如下所示：



Yolov8模型框架

Yolov8算法的核心特征和改动可以归结如下：

1.提供了一个全新的 SOTA 模型，包括 P5 640 和 P6 1280 分辨率的目标检测网络和基于 YOLACT 的实例分割模型。和 YOLOv5 一样，基于缩放系数也提供了 N/S/M/L/X 尺度的不同大小模型，用于满足不同场景需求

2.Backbone：骨干网络和 Neck 部分可能参考了 YOLOv7 ELAN 设计思想，将 YOLOv5 的 C3 结构换成了梯度流更丰富的 C2f 结构，并对不同尺度模型调整了不同的通道数。

3.Head：Head部分较yolov5而言有两大改进：1）换成了目前主流的解耦头结构(Decoupled-Head)，将分类和检测头分离 2）同时也从 Anchor-Based 换成了 Anchor-Free

4.Loss：

1) YOLOv8抛弃了以往的IOU匹配或者单边比例的分配方式，而是使用了Task-Aligned Assigner正负样本匹配方式。

2）并引入了 Distribution Focal Loss(DFL)

5.Train：训练的数据增强部分引入了 YOLOX 中的最后 10 epoch 关闭 Mosiac 增强的操作，可以有效地提升精度

从上面可以看出，YOLOv8 主要参考了最近提出的诸如 YOLOX、YOLOv6、YOLOv7 和 PPYOLOE 等算法的相关设计，本身的创新点不多，偏向工程实践，主推的还是 ultralytics 这个框架本身。

**实验**

本实验使用一张Titan X进行训练，选择预训练模型YOLOv8m.pt，采用YOLOv8m的框架。epochs=100，bach-size=16，imgsz=640即将图像resize到640\*640。

优化器选择Momentum SGD，momentum=0.937，learning\_rate=0.01，weight\_decay=0.0005，前3 epoch使用线性学习率预热，初始值0.1，最后一轮学习率为0.0001。

图像使用3.1.3节中的数据增强流水线。

自定义实验的性能表现如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| category | Images | Instances | Precision | Recall | mAP50 | mAP50-95 |
| all | 41 | 197 | 0.893 | 0.862 | 0.928 | 0.681 |
| slagcar | 41 | 167 | 0.95 | 0.801 | 0.926 | 0.622 |
| excavator | 41 | 26 | 1 | 0.648 | 0.794 | 0.454 |
| bulldozer | 41 | 3 | 0.862 | 1 | 0.995 | 0.753 |
| soilcompactor | 41 | 1 | 0.76 | 1 | 0.995 | 0.895 |

## 3.4后处理与可视化

对于YOLOv5模型，我们可视化了其在数据增强后第一个batch的groundtruth和预测效果如下图所示。



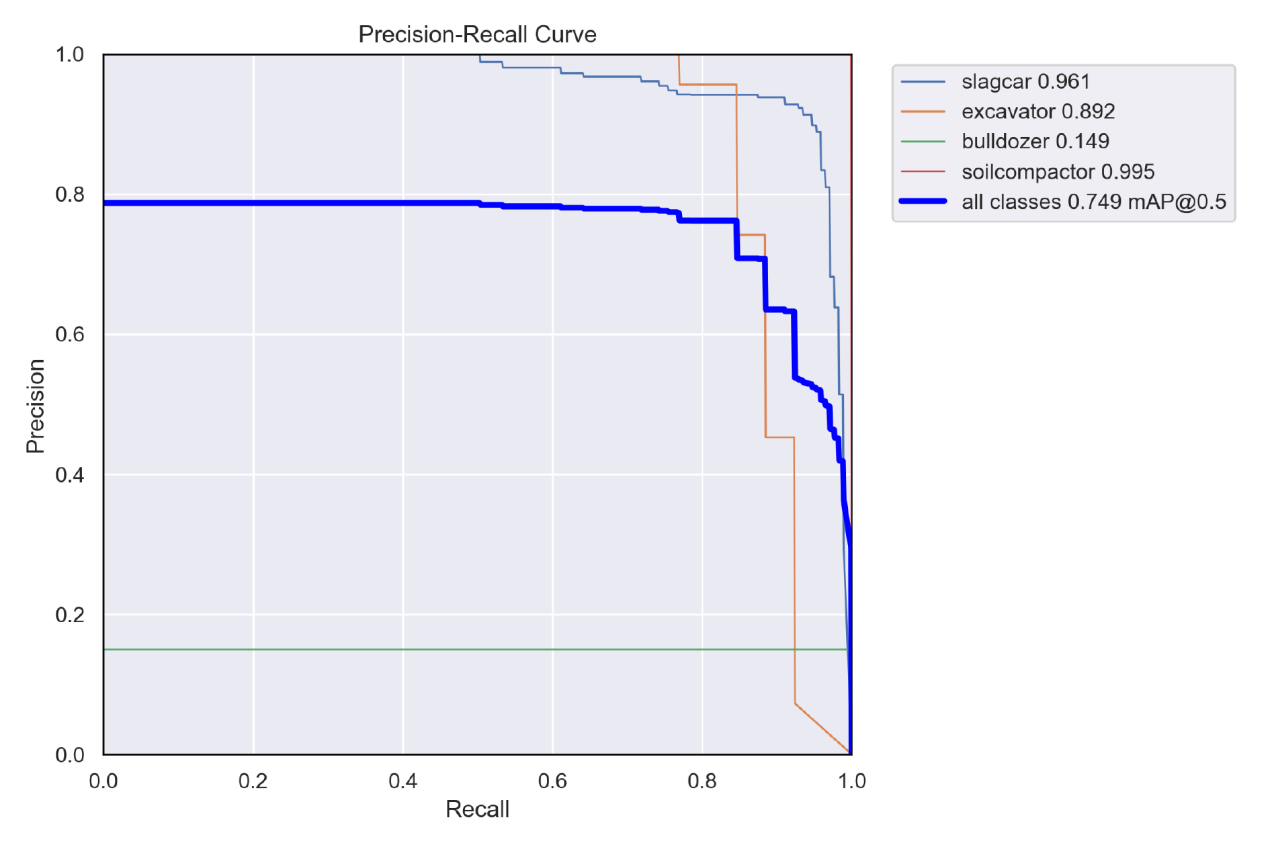
batch0的groundtruth



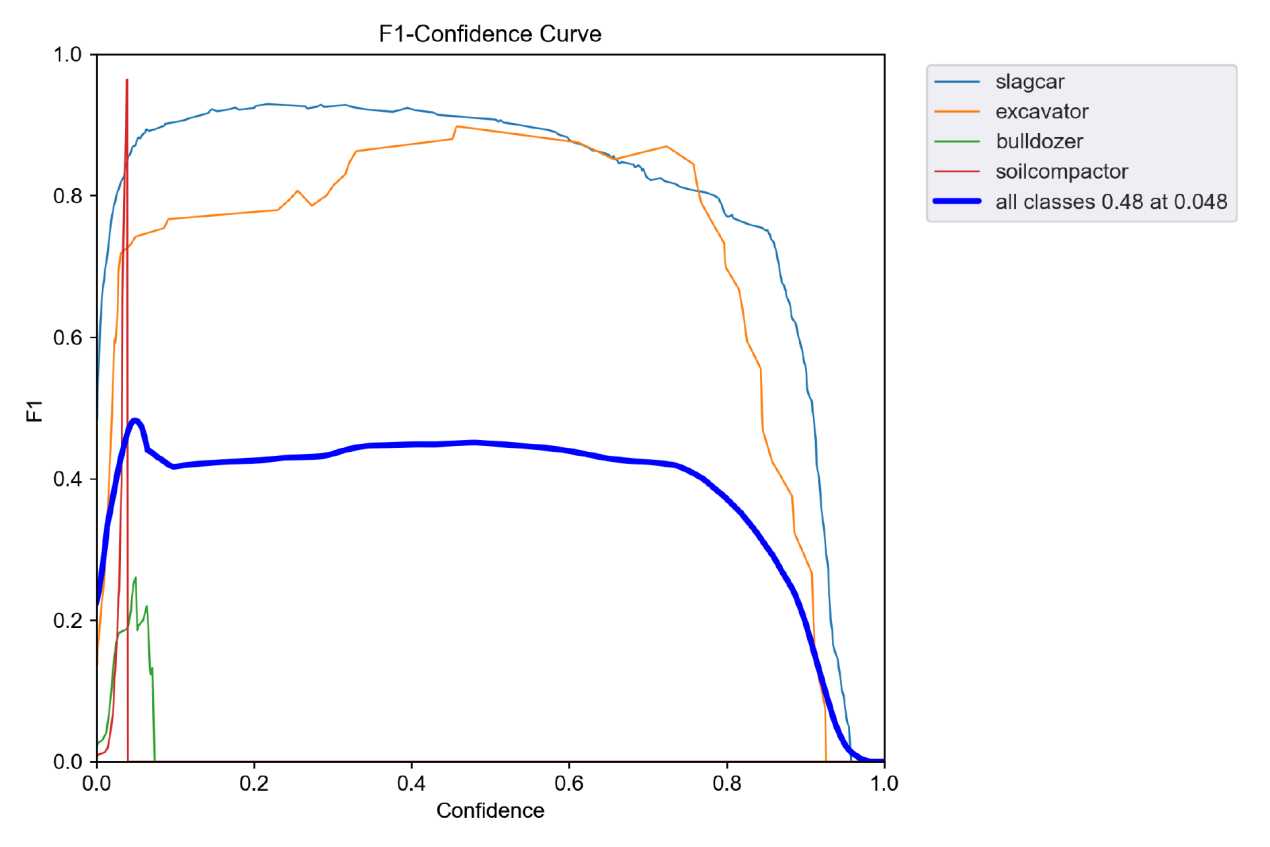
batch0的预测

可以看出在truch1\_19990.jpg这张图中，yolov5模型甚至识别到了不在真实标签中的渣土车，然而在truck2\_5910.jpg这张图中，却十分遗憾并未识别出来。但总体上识别的精度准确率都非常高。

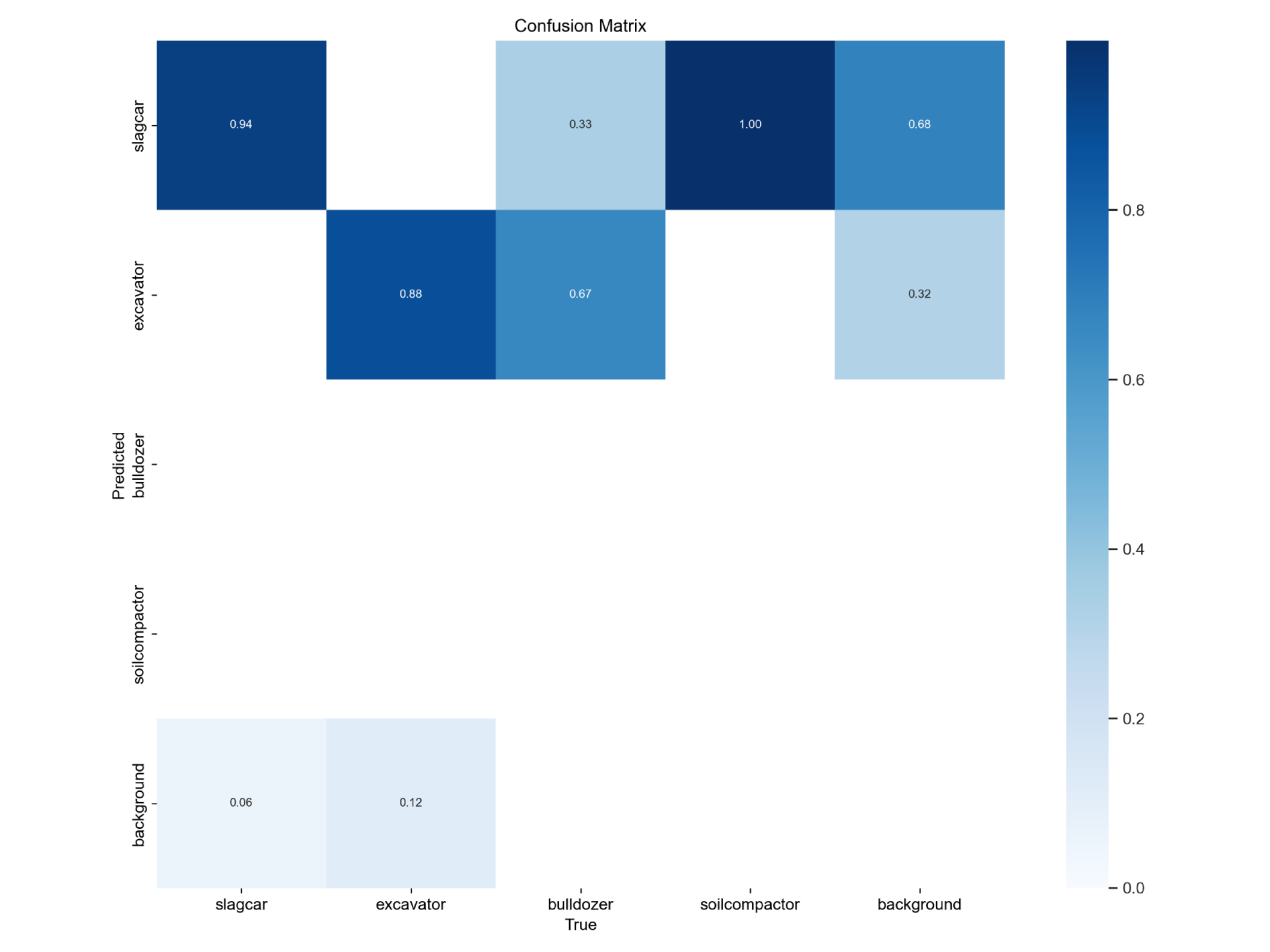
随后，我又绘制YOLOv5模型在我们测试集上的Precision-Recall Curve, F1-Confidence Curve, Confusion Matrix, Precision-Confidence Curve和Recall-Confidence Curve，如下图所示。



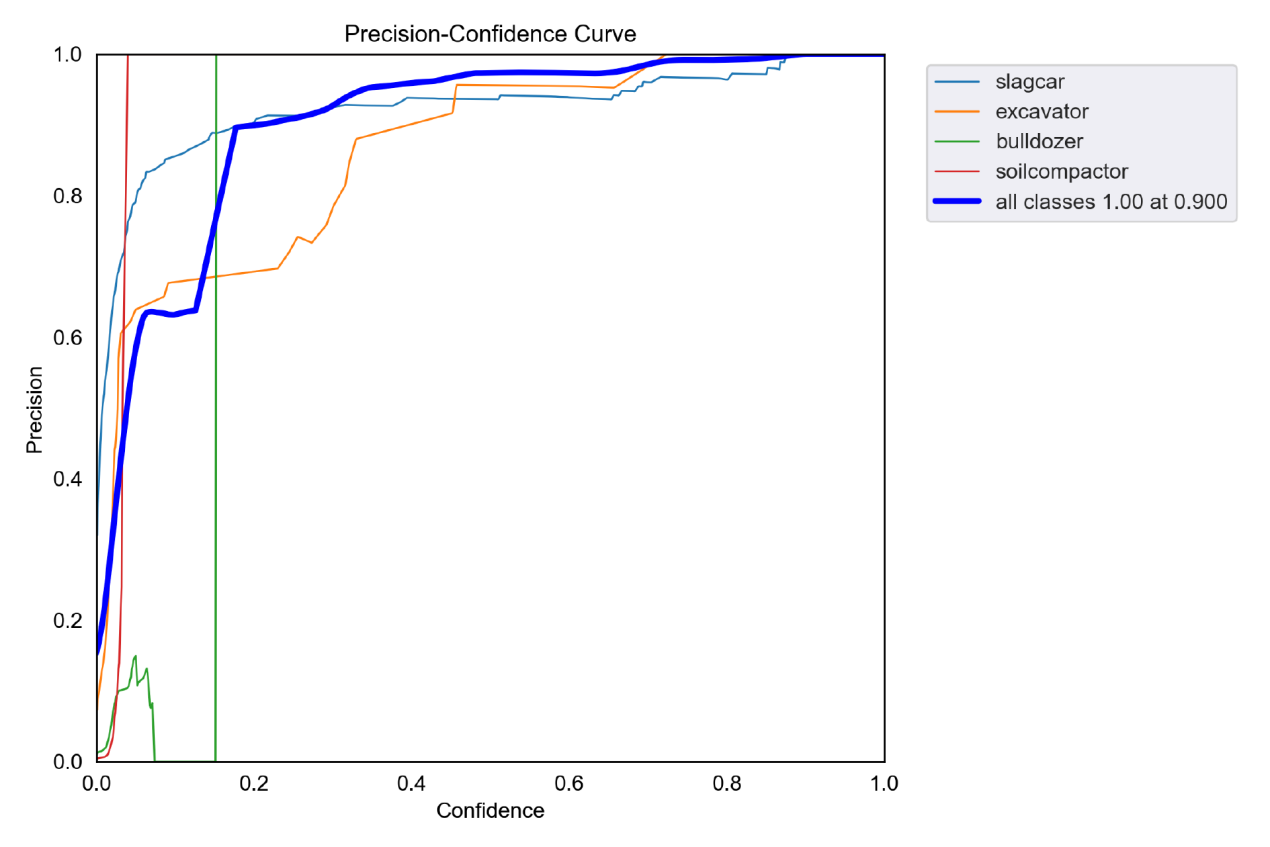
YOLOv5：Precision-Recall Curve



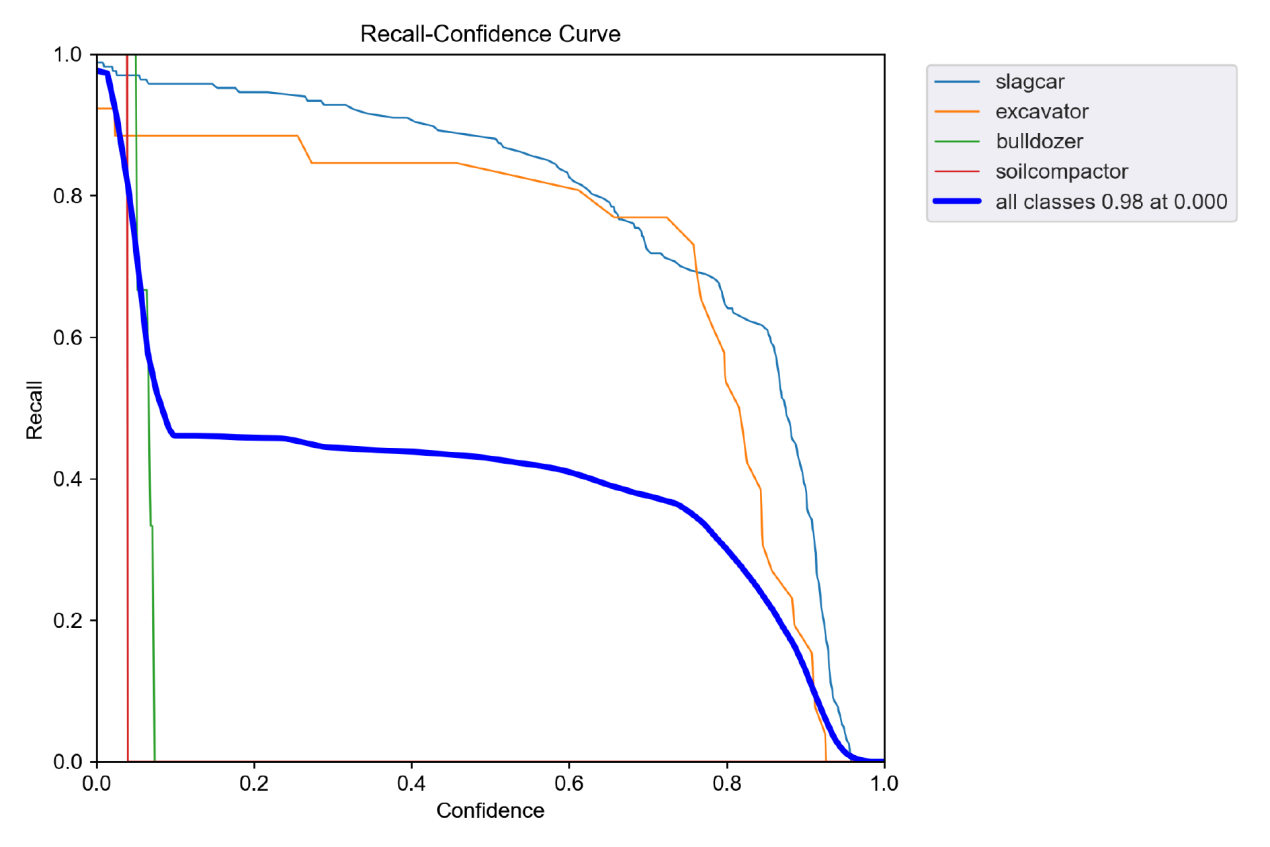
YOLOv5：F1-Confidence Curve



YOLOv5：confusion matrix



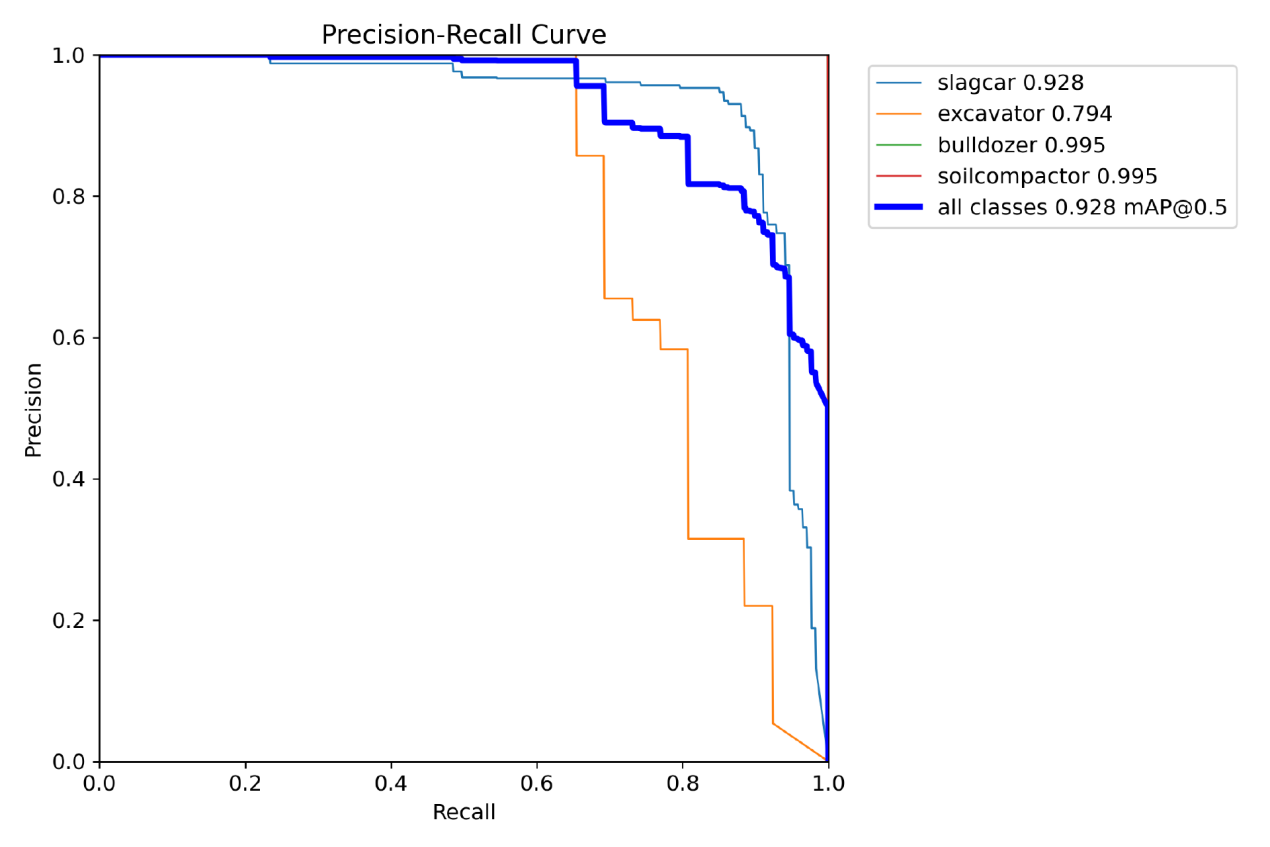
YOLOv5：Precision-Confidence Curve



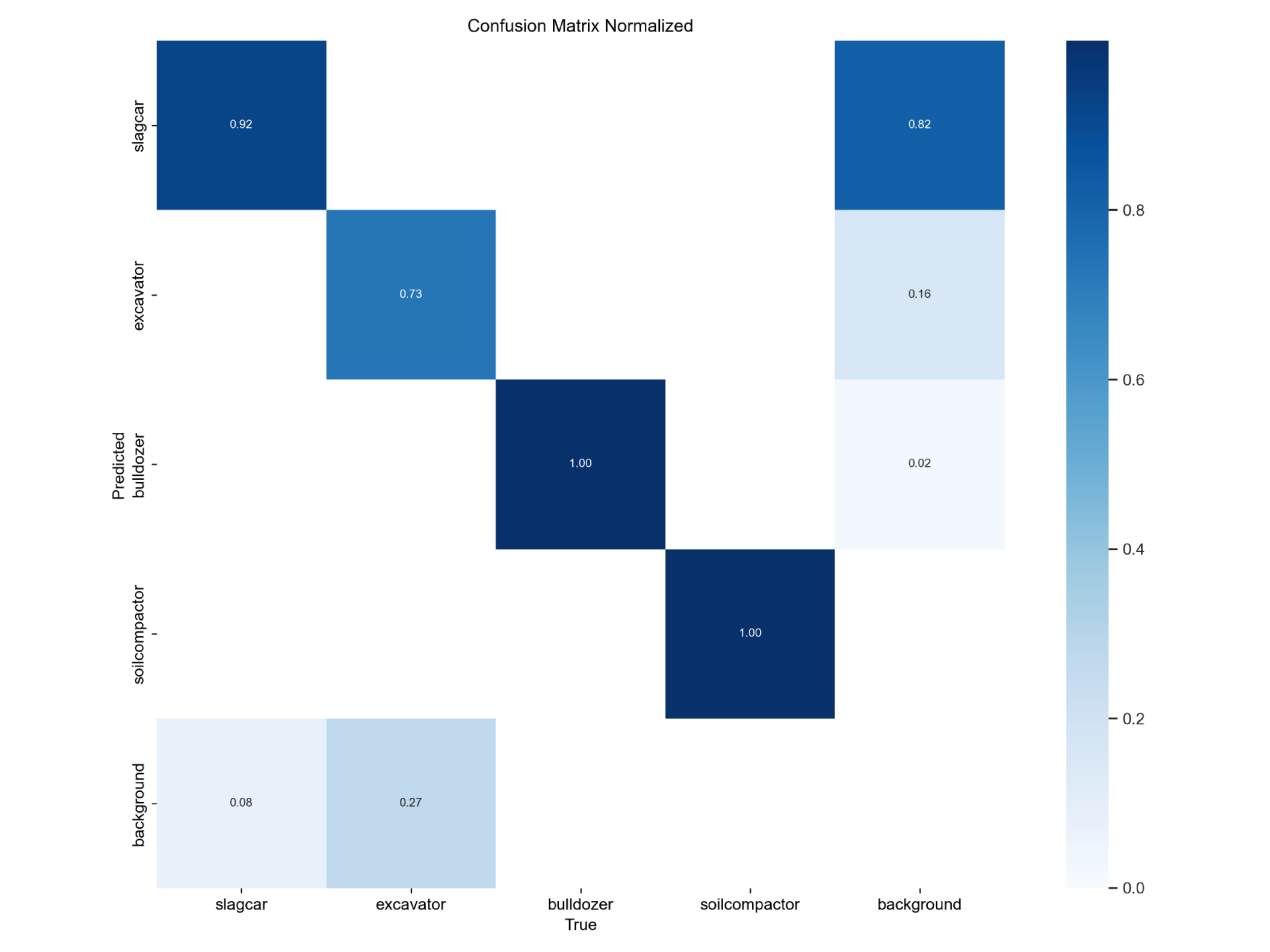
YOLOv5：Recall-Confidence Curve

综合上图可以看出对于训练集较多的slagcar和excavator来说，YOLOv5的识别准确率很高，然而对于样本量很少的bulldozoer和soilcompactor来说，则很有可能会误判为slagcar和excavator，且对于PR曲线来说，限制总体识别准确率的重要因素还是在于bulldozer的识别。

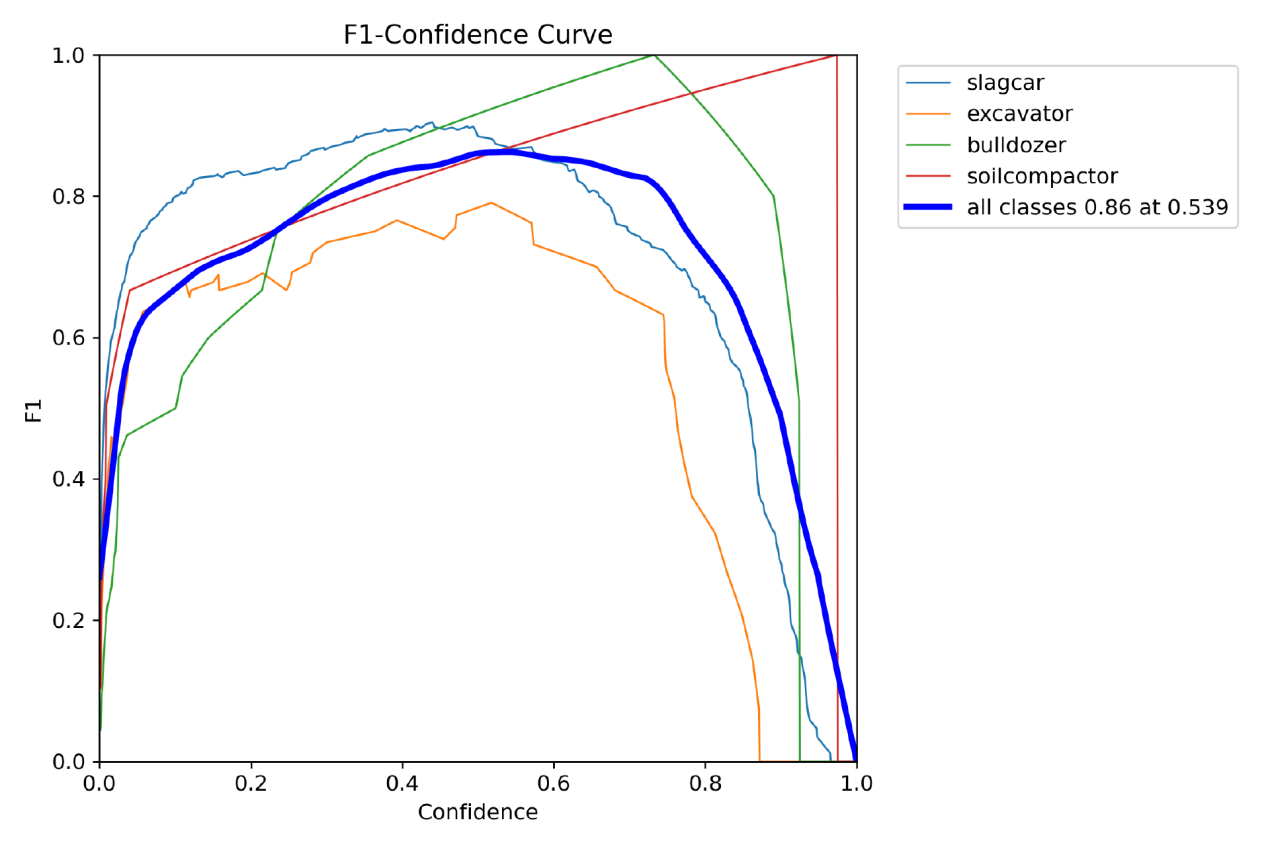
然而，采用YOLOv8模型后，样本整体的识别均有所提升，YOLOv8在我们测试集上的Precision-Recall Curve, Confusion Matrix, F1-Confidence Curve, Precision-Confidence Curve和Recall-Confidence Curve如下图所示。



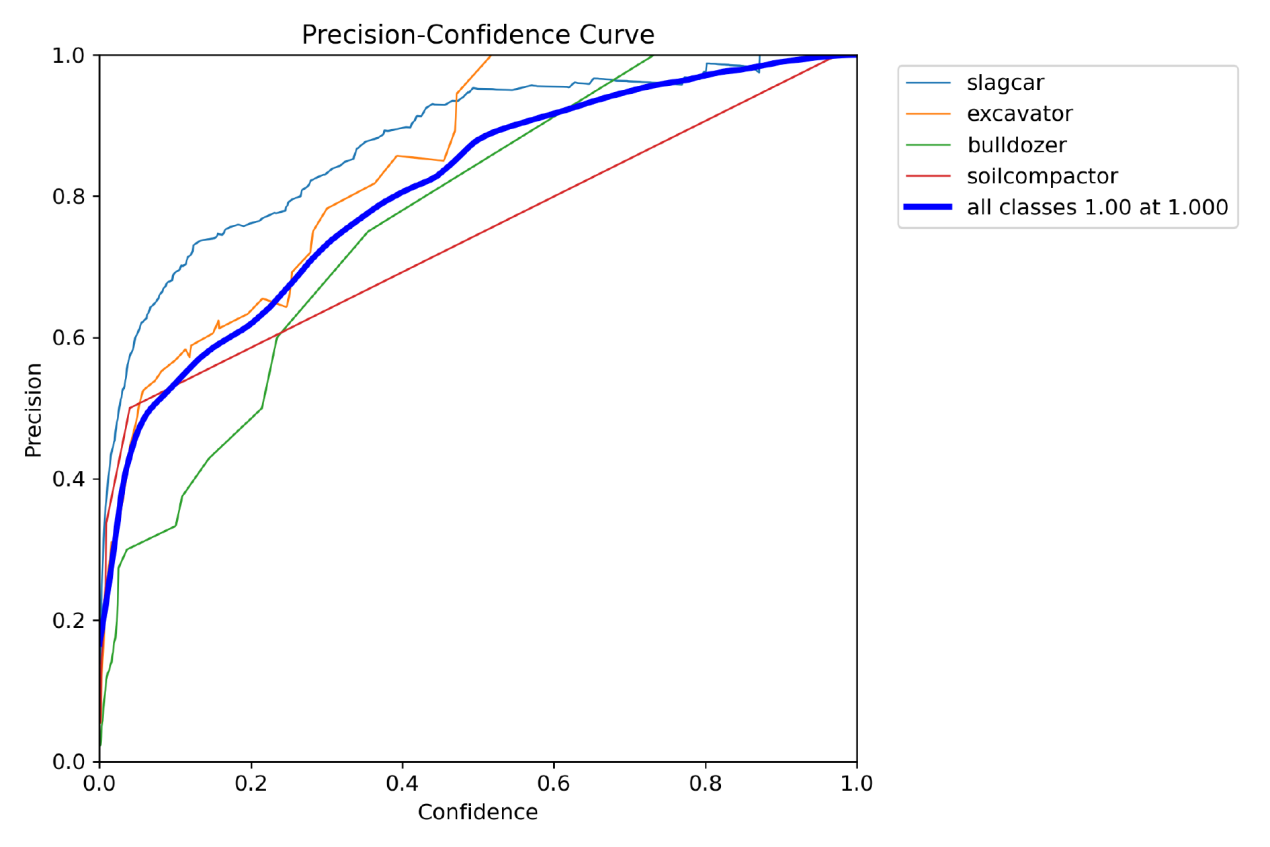
YOLOv8：Precision-Recall Curve



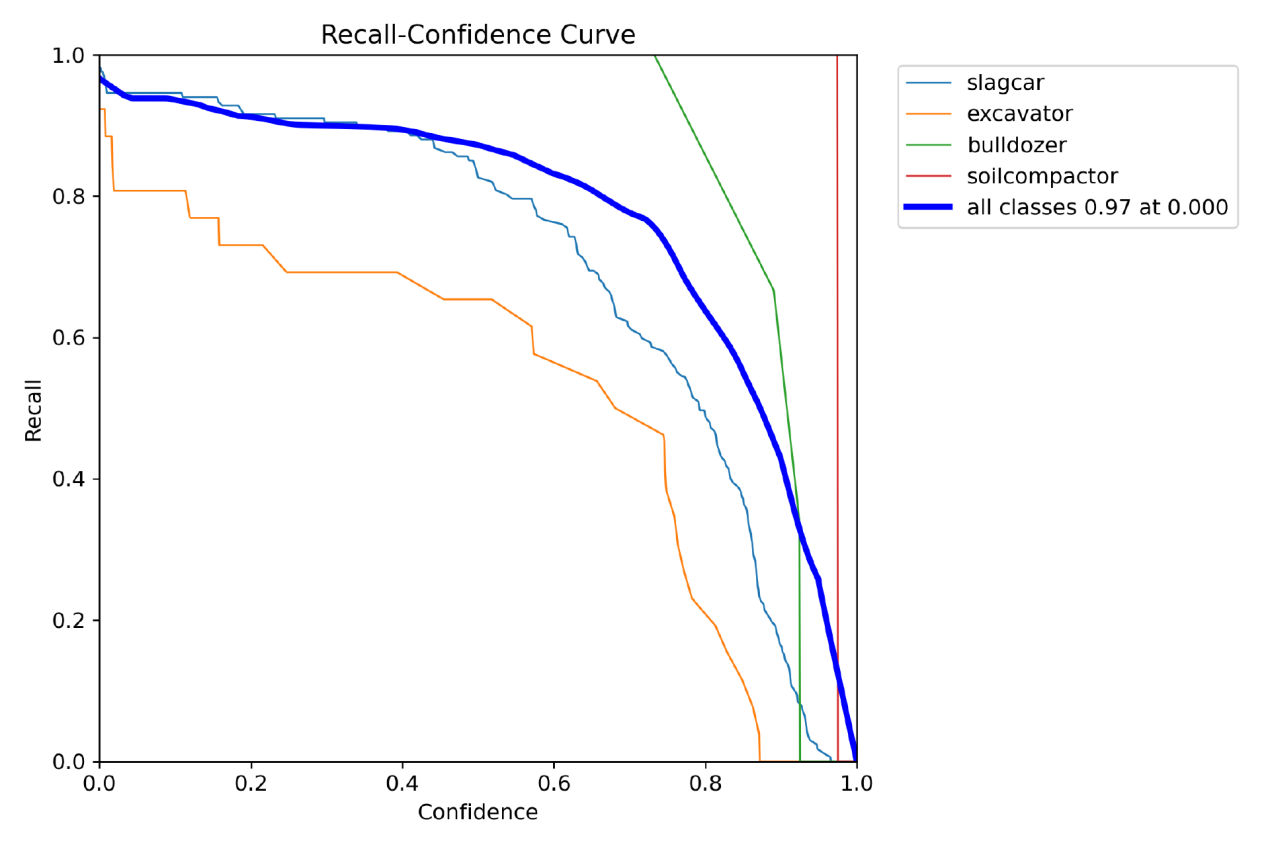
YOLOv8：confusion\_matrix



YOLOv8：F1-Confidence Curve



YOLOv8：Precision-Confidence Curve



YOLOv8：Recall-Confidence Curve

可以看出，相对于YOLOv5来说，YOLOv8在识别的准确性上更加卓越，误判的情况更少，且在mAP为0.5时达到了0.928的准确率。因此这也是我们使用YOLOv8作为最终验收的原因。

# 4.总结

**朱泠皞**：

本次计算机视觉课设的题目非常有挑战性。在普通的目标检测问题上，增加了类别不平衡问题，以及小样本问题。如何处理好这两个问题对我们是一个挑战，同时也是一个宝贵的学习的机会。在此之前，我对目标检测问题的理解还比较粗浅，经过这次课程设计，我系统的学习了目标检测的相关算法，包括两阶段模型，单阶段模型，以及IOU，AP，AR，mAP等概念。此外，为了解决类别不平衡问题和小样本问题，我还深入的研究了当前计算机视觉以及目标检测领域常用的数据增强方案，学习到了很多先进的思想和算法。

不仅如此，我还深入的学习了模型微调的基本知识。模型微调技术在本次课程设计中起到了非常大的作用，不仅大大缩短了训练时间，同时让我们获得了非常好的效果。在微调RetinaNet的时候，我还观察到了过拟合的现象，在尝试数据增强的时候，我也学习到了数据增强的基本思想和规则，并不是随意增强都能够起到预期的作用，而是要task motivated，只有提出针对性的增强方案，才能够得到预期中的效果。

在本次课程设计中，我还广泛的尝试了各种目标检测算法，我还较为系统的学习了当前目标检测方向的热门框架MMdetection，深入的了解了RetinaNet，YOLOv5/v8，RTMDet，DETR等目标检测模型的结构和思想。这对我未来的学习研究有非常大的帮助。

**张凯迪**：

经过本次实验，我了解了一些目标检测的算法，从单阶段模型，两阶段模型到基于Transformer架构的目标检测框架。然而，由于我们目标检测样本量过于稀疏，则对两阶段模型以及基于transformer架构的目标检测模型不太适用。因此我一开始就瞄准的是目标检测方面最为闻名的YOLO模型。

由于曾经使用过YOLOv5模型做实验，因此，这次我轻车熟路直接用上了YOLOv5模型在我们的课设上。对于数据集的划分，一开始没有考虑太多，直接将后面46张图片作为验证集，剩下的图片作为训练集进行训练。谁知不试不知道一试吓一跳，第一次我的模型mAP50在四个标签上平均值可以跑到95的分数，心想这课设也还挺简单。后来和队友讨论了发现，后46张图片只有slagcar一种类型的样本，所以才会有这么高的准确率。因此我决定重新划分数据集进行试验。重新划分数据集后发现，我们的模型在反复调参后最终只能跑到0.68的成绩，并不是非常理想。

后来，看到ultrarysitic的官网上面出了YOLOv8模型，在coco数据集上产生了很不错的效果。于是我采用了YOLOv8m的模型，测试后YOLOv8m能够跑到0.928的分数，实属不易。因此我们一直决定采用YOLOv8m作为最终的验收模型。

综合以上过程，我在跑模型的过程中学会了一些调参方法，同时也学会了一些基本的数据增强方法，目标检测模型的一些基本评价指标，对之后的科研之路有了更加丰富的知识储备。