南京農業大學

计算机视觉与图像处理

课程设计

基于 yolov11 手写汉字的识别

姓　　名: 叶俊泽

班　　级: 人工智能 221

专　　业: 人工智能

学　　号: 11522105

2025 年 4 月

目录

1. 项目整体思路

本项目旨在实现基于深度神经网络的手写汉字识别功能，采用 YOLOv11 目标检测模型作为基础框架，构建完整的识别系统。项目的整体流程如下：

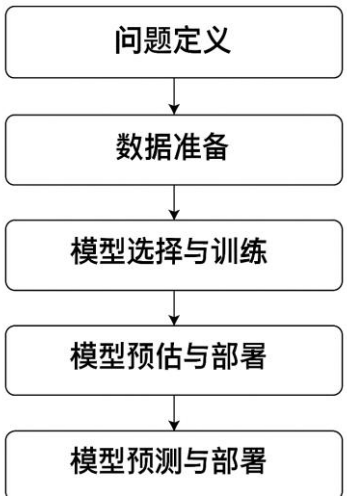
首先，我们将任务形式定义为多类别目标检测问题：每张图像中包含一个手写汉字,模型需要识别出其具体类别。项目使用课程提供的汉字手写图像数据集，共包含 501 汉字，每类约 50 张图像，并使用 char_dict.json 文件建立类别索引。

在数据处理阶段，我们使用自定义脚本 get_dataset.py 对原始数据进行清洗、预处理，并自动生成 YOLO 格式的数据集，完成训练集、验证集和测试集的划分，同时构建了数据配置文件 dataset.yaml。

模型训练部分基于轻量级的 YOLOv11 框架，在已有预训练模型的基础上进行微调。通过 train.py 脚本，我们设置了合理的超参数，进行多轮训练，并保留最优性能模型 best_model.pt。

训练完成后，使用 test.py 脚本对模型性能进行评估，主要指标包括 mAP@0.5、mAP@0.5:0.95，并生成学习曲线、混淆矩阵等图表，直观展示模型识别能力。

最后，我们利用 predict.py 脚本对任意输入图像进行预测，实现模型的部署验证。模型能够准确识别图像中的手写汉字，并输出检测框、类别标签及置信度信息，完成从数据处理到模型预测的全流程闭环。



2. 数据集处理方式

2.1 数据集构建与划分

本项目所使用的数据集由课程提供，包含 500 个文件夹，每个文件夹中约有 50 张手写体汉字图片。每个文件夹的名称是该汉字对应的 Unicode 编码，例如“00699”表示汉字“的”。此外，配套的 char_dict.json 文件用于将这些 Unicode 编码映射为实际汉字字符，形成类别标签索引。

为了适配 YOLOv11 目标检测模型的训练需求,本项目编写了 get_dataset.py 脚本对原始数据进行结构化处理。主要流程如下：

映射构建与清洗：

首先加载 char_dict.json 文件，并筛选出与实际图像文件夹匹配的有效 Unicode 编码。剔除空字符、无效类别后，构建连续编号的类别 ID(class_id)，确保 YOLO 模型的标签是从 0 开始的整数类型。

图像扫描与标签提取：

脚本支持常见图像格式（如.jpg、.png 等），遍历每个文件夹下的图片，并根据其上级文件夹名确定所属类别。每张图像被标注为一个完整覆盖图像区域的边界框（观察汉字在图像中居中且唯一），以满足 YOLO 所需的标注格式：

```
<class_id> 0.5 0.5 1.0 1.0
```

即类别编号 + 归一化的中心点 + 全图范围的宽高。

数据集划分：

使用 sklearn.model_selection.train_test_split 对图像及其标签进行两次分层抽样划分，确保各类别在训练集、验证集和测试集中分布均匀：
测试集比例：15%；验证集比例：15%；训练集比例：70%

```
yolo_hanzi_dataset/
├── images/
│   ├── train/
│   ├── val/
│   └── test/
└── labels/
    ├── train/
    ├── val/
    └── test/
```

同时为每张图像生成对应的 .txt 标注文件，文件名由其原始路径中的父文件夹名和图像名组合而成，确保唯一性，避免重名冲突。最后生成 dataset.yaml 配置文件，记录数据集根路径、图像目录、类别数量和类别名称等信息，供 YOLOv11 模型加载使用。为保证训练过程对中文字符的兼容性，该文件采用 UTF-8 编码保存，类名列表也通过 json.dumps(..., ensure_ascii=False) 写入，确保汉字不被转义。

3.模型设计与实现

3.1 模型选择与描述

在本项目中，我们选择使用 YOLOv11n 作为手写汉字识别的核心模型。YOLOv11 是目标检测领域的新一代深度神经网络模型，具有速度快、精度高、部署轻量化等优点，广泛应用于各种实际场景。本项目所使用的 yolov11n 为其中的最小版本（Nano），在保持基本检测能力的前提下，显著降低了参数量与计算开销，非常适用于小目标检测任务与实时推理需求。

YOLOv11 相较于前代版本（如 YOLOv8）在架构上进行了多项关键改进，旨在在保持高精度的同时进一步提升模型的计算效率和部署灵活性。

C3k2 模块：该模块是对传统 CSP（Cross Stage Partial）结构的优化，采用两个较小的卷积核替代大型卷积操作，从而减少计算开销并提升特征提取效率 。

C2PSA（Channel-wise and Point-wise Spatial Attention）模块：引入空间注意力机制，提升模型对关键区域的关注度，从而提高检测精度，尤其在小目标和遮挡场景中具有优势 。

LSKA（Large Kernel Separable Convolution Attention）模块：通过将二维卷积核分解为水平和垂直的一维卷积，降低计算复杂度的同时增强模型对目标形状的学习能力 。

此外，YOLOv11 保留了 YOLO 系列一贯的单阶段检测范式，支持目标检测、实

例分割、图像分类等多种计算机视觉任务，具备良好的通用性和扩展性。通过引入多项架构创新，在保持模型精度的同时显著提升了计算效率和部署灵活性，为本项目的手写汉字识别任务提供了有力的技术支持。

4.模型训练

4.1 模型超参数设置

考虑到本项目所采用的数据集具有“类别数多、每类样本少”的典型特点，我们在 YOLOv11n 模型训练过程中对超参数配置做了针对性调整，以尽量避免过拟合并提升模型在小样本场景下的泛化能力。训练过程采用基于 AdamW 优化器的自适应权重更新策略，并搭配了合理的学习率预热、马赛克增强控制与正则化配置，使得模型在较短训练周期内实现了稳定收敛。

训练图像的输入分辨率设置为 128×128，兼顾模型计算效率与手写字符的可辨性，批大小设为 64，利用多线程数据加载——32 线程——与全缓存加速，确保训练过程流畅稳定。为缓解早期模型震荡并控制训练节奏，设置了 5 轮预热（warmup）与 patience=8 的早停机制。此外，为提升模型对手写字体微变形的鲁棒性，在图像增强方面仅保留了轻度旋转、平移与缩放，避免使用易破坏笔画结构的翻转与混合增强策略——如 MixUp、Flip 等，以免误导学习目标。

主要的模型训练超参数设置如下表所示：

参数类别	参数名称	设置值	说明
训练规模	epochs	100	总训练轮数，结合 patience=8 控制早停
	batch	64	每轮训练图像数量
	imgsz	128	输入图像尺寸
优化器	optimizer	AdamW	自适应权重优化器，适合小样本任务
	lr0, lrf	0.001/0.01	初始/最终学习率
	momentum	0.937	动量参数
	weight_decay	0.00075	权重衰减项，防止过拟合

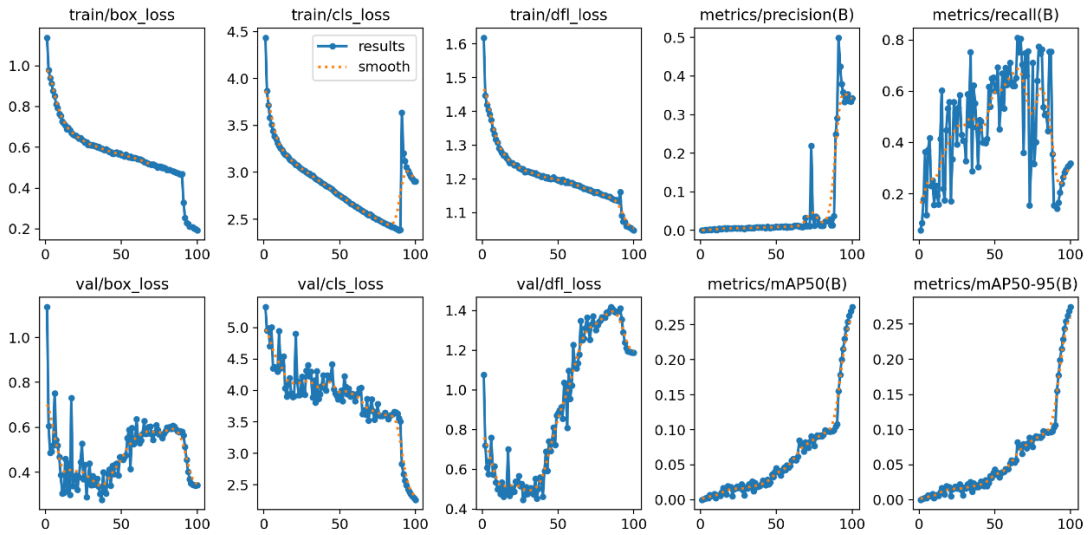
正则化	dropout	0.5	分类任务正则化
	degrees	5	随机旋转角度
	translate	0.1	平移增强
	scale	0.25	缩放增强
	mosaic	0.24	启用马赛克拼接的概率
	mixup,flipud/lr	0	禁用混合增强与翻转
稳定性	warmup_epochs	5	训练预热轮数，防止初期梯度不稳定
	deterministic	TRUE	启用确定性训练，保证实验可复现
	seed	0	固定随机种子
加速与缓存	cache, amp	TRUE	开启数据缓存与混合精度训练

根据理论分析以及后期的测试结果，上述训练参数的配置在稳定性、效率与泛化能力之间取得了良好平衡。尤其在小样本、大类别的目标检测任务中，通过谨慎控制数据增强的强度与优化器学习节奏，有效抑制了模型过拟合的趋势，提升了其对真实手写图像的识别能力。

4.2 模型优化过程

尽管模型最终取得了较为理想的检测效果，但整个训练与优化过程并非一帆风顺。最大的挑战出现在数据增强策略的选择上。起初，为了最大程度缓解过拟合问题，我在训练配置中启用了大量图像增强参数，包括大幅度的旋转、剪切、翻转等操作。然而对于手写汉字识别这一任务而言，字符结构的细微差异往往是区分类别的关键,而这些强度过高的增强方式在无形中破坏了原始字形的空间结构，导致模型难以提取有效特征。

在持续训练 100 个 epoch 后，模型在验证集上的性能表现极差，预测准确率和召回率均低于 0.5，说明其已经严重偏离了学习目标。这一阶段的失败经验促使我重新审视增强策略的适用性，并逐步调低增强幅度，最终形成更贴合任务特性的轻量级增强配置，从而使模型恢复了稳定的学习能力并显著提升了性能。



从上述训练过程的指标曲线可以明显观察到模型在早期阶段存在严重的过拟合问题，尤其在验证集上的各项损失震荡剧烈，精度（Precision）与召回率（Recall）始终维持在较低水平，且波动幅度较大，反映出模型在泛化能力上的严重不足。

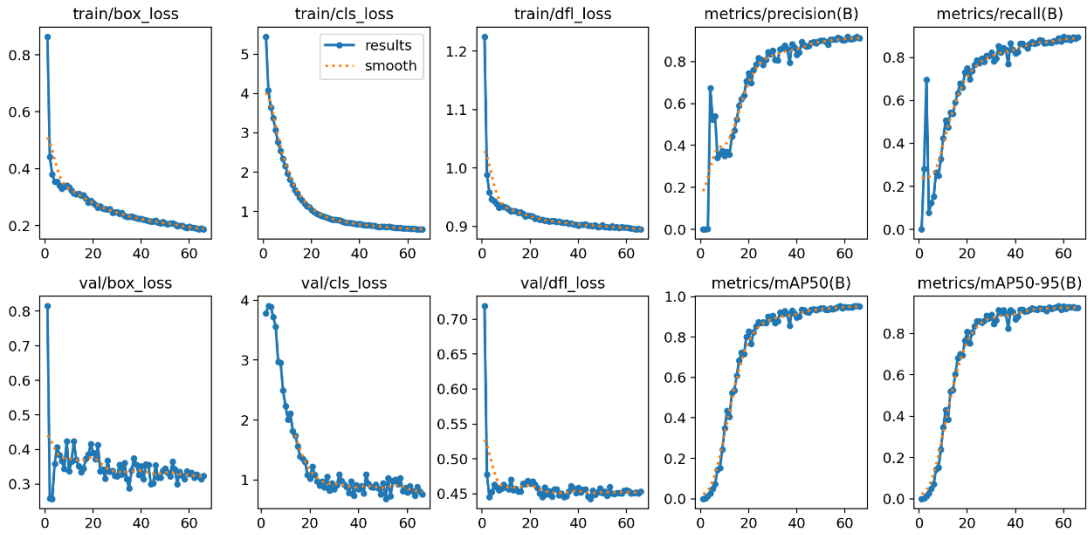
然而，一个值得注意的现象是：由于我在训练配置中设置了“取消马赛克增强”策略的触发点为最后 10 个训练周期，模型在这一阶段的性能出现了明显反弹。具体表现为精度与召回率均呈显著上升趋势，同时训练与验证损失同步下降，mAP 等评估指标也出现陡升拐点。这一变化表明，模型本身具备较强的特征提取与目标识别能力，而训练早期性能不佳，很可能是由于数据增强策略与超参数设置不当，特别是过于激进的增强方式破坏了图像的结构信息，干扰了模型的学习过程。

这一阶段的观察为后续的训练策略优化提供了有益启示：适度控制数据增强的强度，并根据模型学习阶段动态调整增强策略；如逐步减少干扰性较强的增强操作，可以有效提升模型收敛速度与最终性能。

因此，在后续的训练中，我基于前一轮训练过程中观察到的失败原因，对训练策略进行了系统性的调整。首先在数据增强部分，我显著降低了图像扰动的强度：如限制旋转角度在±10°以内，取消左右翻转操作，调整剪切与缩放的比例区间，确保手写汉字的笔画结构不被破坏，从而保留字符的关键信息。同时，保

留 Mosaic 增强的前提下，引入了 CosineAnnealing 学习率调度器，使得模型在后期训练阶段以更平滑的方式收敛，有效降低了训练震荡。

改进之后的结果如下：

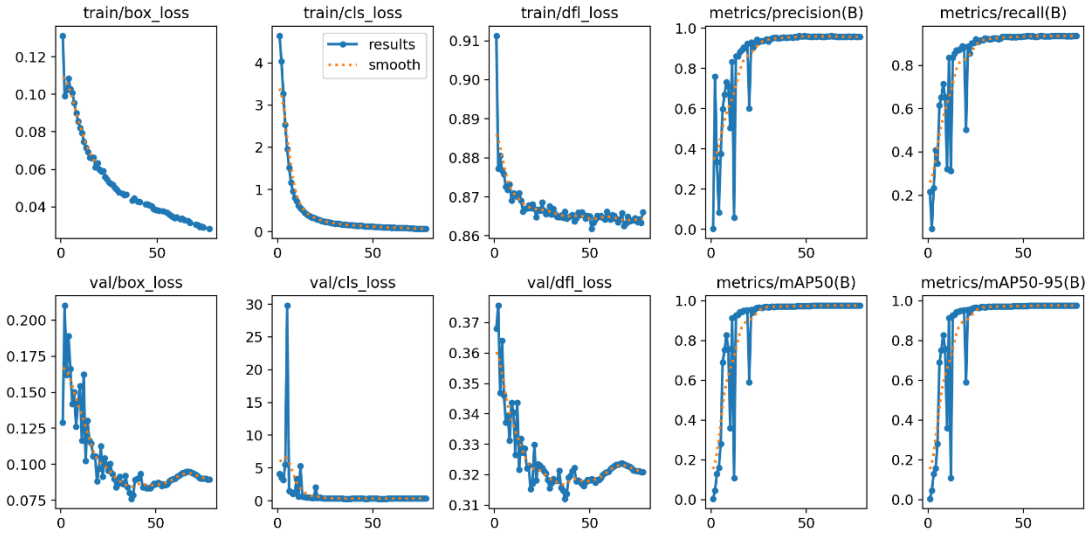


虽然这一轮训练结果相比前一次取得了显著进展，各项指标均表现良好，尤其是验证集上的 mAP50 和 mAP50-95 几乎达到饱和状态，但仍存在值得关注的问题：模型的验证精确度始终徘徊在 0.9 左右，而训练集的精确度已突破 0.93。这种“训练优于验证”的差距，表明模型仍存在一定程度的过拟合，且可能陷入了优化过程中的“鞍点”——即 loss 曲面上的平坦区域，梯度较小，难以跳出局部最优。

深入分析超参数配置后，我发现，为了加快训练过程中的验证速度，我在上一轮中设置了较大的 batch size。这虽然在硬件资源允许的前提下提升了吞吐率，但也降低了模型训练过程的扰动性（stochasticity），使得模型更容易陷入局部最优。为此，我在下一轮训练中作出如下调整：将 batch size 调整为 64，以重新引入一定程度的梯度扰动；同时将图像输入尺寸设置为 128，使模型能够更充分地捕捉细节特征；此外，还将数据加载线程数增至 32，以提升训练效率并充分利用多核 CPU 资源。

上述调整不仅有效缓解了过拟合问题，也加速了模型的优化收敛过程，最终在保持模型稳定性的同时，取得了更高的泛化性能。训练与验证精度的差距进一

步缩小，验证集上 mAP50 达到理想状态——精确度 0.97——说明模型已较好地掌握了手写汉字的判别性特征，优化过程也趋于全局最优。此轮训练的成功，充分印证了 batch size、图像尺度、数据加载策略等训练细节对模型最终性能的关键影响。



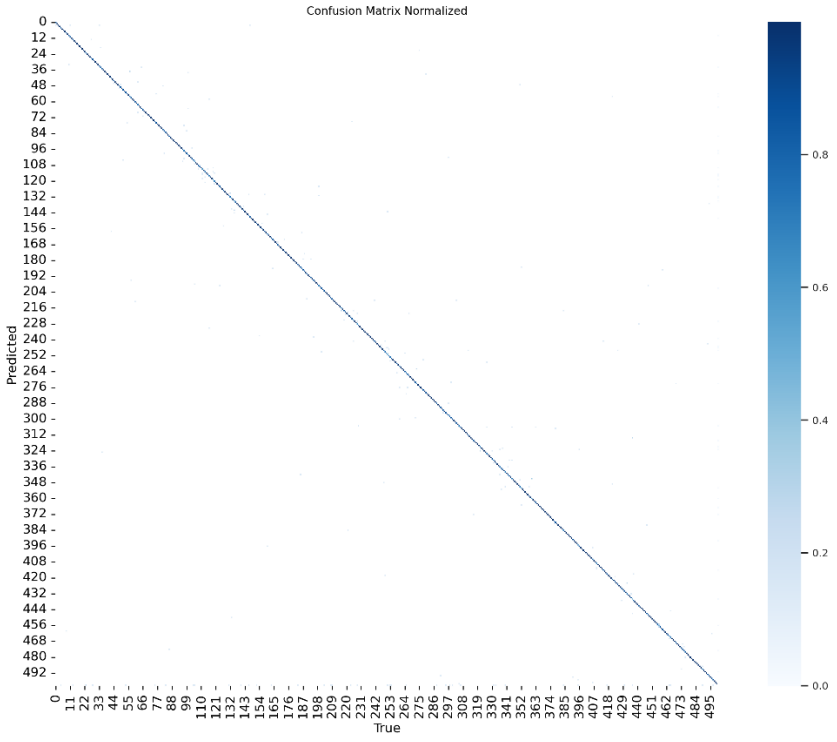
5. 结果展示与评估

为了全面、系统地评估本模型在目标检测任务中的实际性能表现，我从**训练过程监控、分类精度验证、以及多维度评估曲线**三个层面展开分析，并辅以关键图表进行直观展示，以确保结果解读既有数据支撑，也具备实际意义。

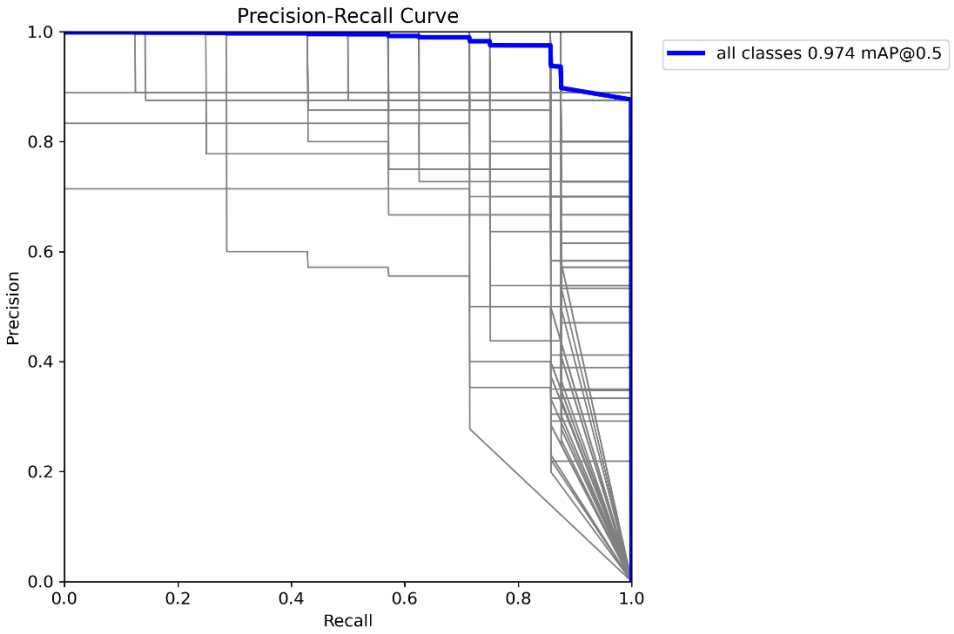
从训练过程中的指标曲线（上文已经给出）可以看出，三项核心损失——定位损失（box_loss）、分类损失（cls_loss）与分布焦点损失（dfl_loss）在训练初期迅速下降，并在第 30 轮之后趋于平稳，未出现明显震荡，验证集上的损失亦同步下降且无反弹迹象。这表明模型不仅在训练集上达成了良好的拟合效果，同时具备了相当的泛化能力。与此同时，几项核心性能指标（如精确度 Precision、召回率 Recall 以及平均精度均值 mAP@0.5 和 mAP@0.5:0.95）也同步提升并趋于饱和，表现尤为亮眼。例如，验证集的 mAP@0.5 稳定保持在 0.97 以上，而更为严格的 mAP@0.5:0.95 指标也接近 0.97，说明模型在不同 IoU 阈值下

均有稳定、可靠的表现。虽然训练集与验证集之间的指标仍存在轻微差距，但这类轻度过拟合现象已通过早期对超参数（包括 batch size、图像输入尺寸、数据加载线程数等）的细致调优得到有效控制，避免了模型性能陷入瓶颈区。

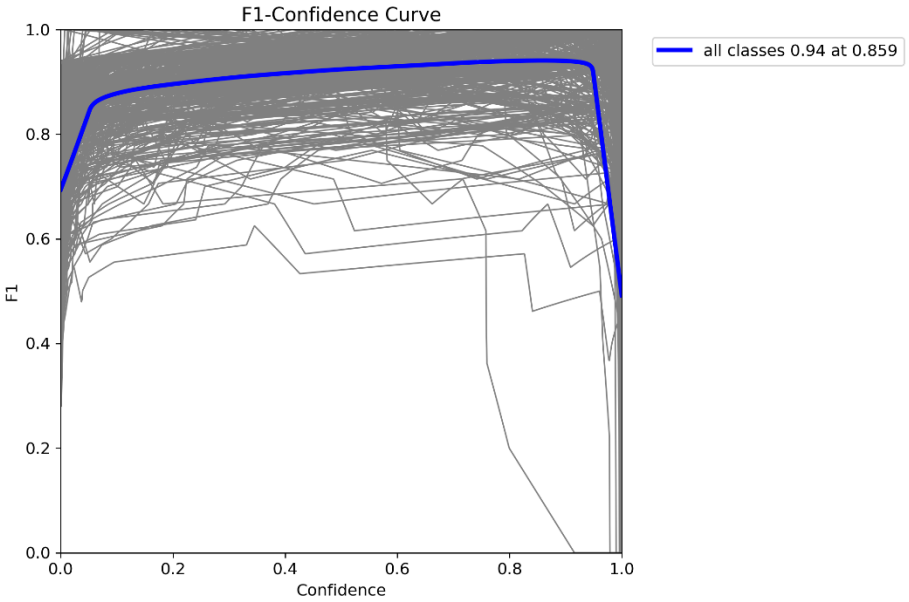
进入更高层次的评估环节，归一化混淆矩阵为我们展示了模型在近 500 个类别上的分类能力。图中对角线部分颜色深邃清晰，非对角区域几乎为空白，充分说明模型在多类别识别任务中表现出极强的判别力，类别间几乎无明显混淆，整体分类鲁棒性极高。



进一步地，我引入了多种置信度敏感的性能曲线，深入探讨模型在不同阈值下的行为表现。首先，Precision-Recall 曲线展示了在 Recall 从 0 到 0.95 的广泛范围内，Precision 始终保持在接近 1.0 的高水平，仅在 Recall 极大时才出现略微下降，印证了高达 0.974 的 mAP@0.5 得分。这说明模型既能准确识别目标，也能充分检出目标，达成了高查准率与高查全率的理想统一。

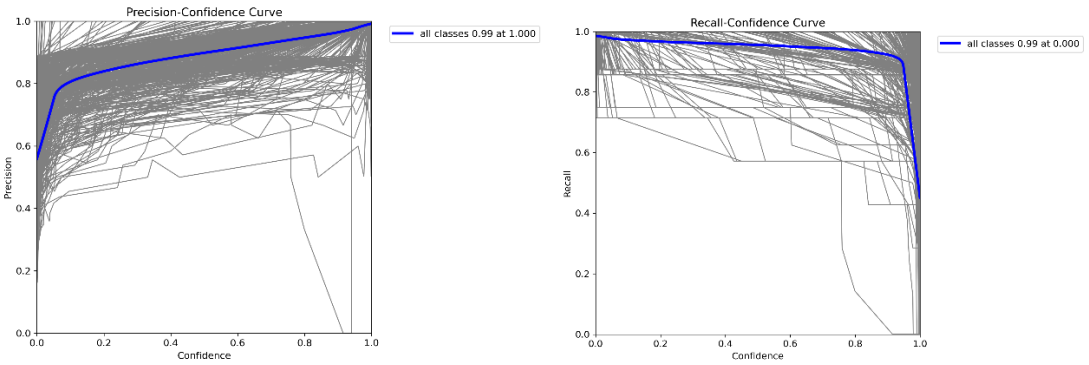


与此同时，F1-Confidence 曲线进一步揭示了精确率与召回率之间的最优平衡点。图中蓝色曲线表明，当置信度阈值设置在约 0.859 时，F1 分数达到峰值 0.94，说明此时模型在 Precision 与 Recall 两端取得最佳折中效果，为实际部署提供了阈值设定的重要参考依据。

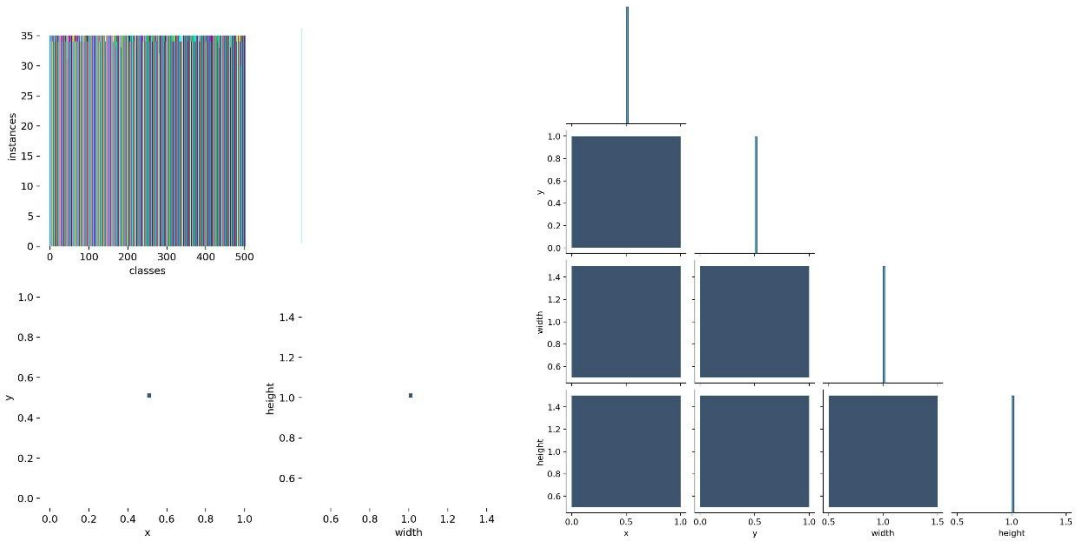


此外，Precision-Confidence 与 Recall-Confidence 两条曲线则分别反映了精确率与召回率对置信度变化的敏感性。前者随置信度提升而单调递增，最高值

约为 0.99；而后者则单调递减，在极低阈值下几乎可检出全部目标。这种互补关系清晰地展现了在不同业务场景下应如何灵活权衡精度与覆盖率，为后续策略优化提供了图形化支持。



值得一提的是，模型的优异表现并非偶然，也深受数据集结构影响。通过分析 labels 与 labels_correlogram，可以发现本数据集具备如下特性：类别分布较为均衡、目标位置高度集中于图像中心（x ≈ 0.5，y ≈ 0.5）、目标尺寸较大且近似满幅（width ≈ 1.0，height ≈ 1.0）。这种分布的规律性极大降低了任务复杂度，为模型提供了稳定的学习基础，也是模型能够达到高性能的重要先验保障。



综上，通过训练指标的动态追踪、混淆矩阵的全类别验证，以及 PR、F1、Precision/Recall-Confidence 等多视角指标曲线的系统分析，不仅充分证实了

当前模型具备极高的准确性、稳定性与泛化能力，还进一步结合数据集结构对模型结果做出了合理解读。这些结果为后续的模型精调或推理部署提供了坚实的数据基础与决策支持。

6. 项目思考

在多次的总结以及试错中，最终得到了令人满意的结果。**在可控更长的时间内，我们似乎愿意接受更接近全局最优解的模型**。这本身就是一次关于“收敛稳定性”与“泛化能力”的博弈。在模型能力以及很强的情况下，对数据集的洞悉，其实才是模型成功的一半。标签分布图揭示了一个不容忽视的“先验偏好”——目标几乎都处于图像中心，尺寸也接近全图，类别相对均衡。这种分布上的“人为简化”让模型在训练中得以高效收敛。但也同时提示我们：当前模型表现的好，有多少是模型能力本身。

真正优秀的目标检测系统，不在于追求某一个指标的极致，而是在于构建一个多维约束下的平衡体——**它能够在复杂世界中快速、稳定、准确地做出决策，同时还能对不确定性保持足够的泛化能力。**

如何让目标检测系统具备自我调优、甚至自我解释的能力？这或许就是通往下一代视觉智能的必经之路。**倘若未来的目标检测器能融合视觉语言、上下文感知乃至对环境结构的建模，那么它将不只是看到目标，而是真正理解场景了。**

7.附录

项目代码文件所示：具体使用说明请仔细阅读 README.md 文件。

```
.
├── get_dataset.py       # 数据集准备脚本
├── load_model.py        # 模型加载工具，包括直接加载yolo模型或者选择加载底层的torch.nn.Module模型
├── train.py             # 模型训练脚本
├── test.py              # 模型测试脚本
├── predict.py           # 预测推理脚本
├── yolo11n.pt           # 预训练模型
├── best_model.pt        # 训练得到的最佳模型
├── runs/                # 训练结果和预测输出
│   ├── detect/
│   │   ├── train*/      # 训练、测试集测试日志和权重
│   │   └── predict/     # 预测结果
└── ultralytics/
    └── cfg/
        └── default.yaml # 训练配置文件
```