

基于 AC-YOLO 的集装箱表面破损智能检测方法研究

摘要

随着全球贸易的快速发展，集装箱作为国际货物运输的核心载体，在长期使用过程中易产生裂纹、凹陷、穿孔、锈蚀等表面损伤。传统的人工检测方法存在效率低下、主观性强等局限性。本研究基于计算机视觉与深度学习技术，构建一套完整的集装箱表面破损智能检测系统，实现了从损伤存在性判别到精确定位分类的多层次检测任务。

针对问题一：基于注意力机制的损伤分类模型。实现集装箱图像是否存在破损的快速准确判别，为后续精细检测提供初步筛选。本研究采用多类别分类→二值化决策的创新策略，以 **EfficientNet-B0** 为特征骨干网络，引入通道注意力机制构建分类模型。本研究首次将通道注意力机制与多类别分类策略相结合应用于集装箱损伤识别。结论表明：模型在验证集上达到 **88%** 的准确率，整体 F1 分数为 **80.64%**。各类别检测性能分别为：凹陷（Dent）**精确率 92.65%**、召回率 90.43%；破洞（Hole）**精确率 68.49%**、召回率 74.63%；锈蚀（Rusty）**精确率 77.54%**、召回率 80.45%。该模型在高频类别上识别精度突出，具备良好的工程应用价值。

针对问题二：本研究改进 **YOLOv1s** 模型构建 **AC-YOLO** 检测与分割模型。在判定存在损伤的基础上，实现损伤区域的精确空间定位与具体类别识别。本研究创新性地设计了 **AdvancedContainerAugmentation** 数据增强策略，深度融合领域知识的**物理仿真**，包括锈蚀纹理模拟、动态阴影合成、金属表面反光效应、雨水痕迹及污渍添加等。面向集装箱特定场景的数据增强范式，本研究设计系统化的**消融实验**，定量分析各技术组件的独立贡献与协同效应。结论：完整 **AC-YOLO** 方案在测试集上达到 $mAP@0.5=0.912$, $mAP@0.5:0.95=0.716$ ，显著优于基线模型 ($mAP@0.5=0.891$, $mAP@0.5:0.95=0.676$)。消融实验表明，数据增强策略对性能提升贡献显著，在 $mAP@0.5:0.95$ 指标上的 0.04 提升，模型在边界框回归质量上的实质性改进。PR 曲线与 F1-置信度曲线分析显示，模型在置信度 **0.3-0.5** 区间达到最佳平衡点 ($F1=0.86$)，具备优异的检测性能。

针对问题三：本研究对前两阶段构建的模型进行系统性评估，分析模型在不同条件下的性能表现与边界案例。结论：**AC-YOLO** 模型在三类损伤目标上均取得稳定表现，其中凹陷类检测最为准确（真阳性 3841），破洞类（真阳性 818）和锈蚀类（真阳性 2853）也保持较高识别率。模型在复杂成像条件下表现出良好的适应性，为集装箱损伤的自动化检测提供可靠的技术支撑。

本研究提出的方法体系在工程严谨性与学术创新性方面均达到较高水准，为港口集装箱自动化巡检提供了可复现、可扩展的解决方案。

关键词：集装箱破损检测；AC-YOLO 模型；注意力机制；计算机视觉；目标检测

目 录

| | |
|-----------------------------------|----|
| 1 问题重述 | 3 |
| 1.1 问题背景 | 3 |
| 1.2 问题提出 | 3 |
| 2 问题分析 | 3 |
| 2.1 问题一的分析 | 3 |
| 2.2 问题二的分析 | 3 |
| 2.3 问题三的分析 | 4 |
| 3 模型假设 | 4 |
| 4 符号说明 | 5 |
| 5 问题一的模型建立与求解 | 6 |
| 5.1 建模思路 | 6 |
| 5.2 基于注意力机制的损伤识别模型建立 | 6 |
| 5.3 基于注意力机制的损伤识别模型求解 | 8 |
| 5.3.1 数据集与样本特征分析 | 8 |
| 5.3.2 模型训练过程及其求解结果 | 10 |
| 6 问题二的模型建立与求解 | 12 |
| 6.1 建模思路 | 12 |
| 6.2 AC-YOLO 检测与分割任务的模型构建 | 12 |
| 6.3 AC-YOLO 检测与分割任务的模型求解 | 13 |
| 6.3.1 消融实验结果展示 | 13 |
| 6.3.2 完整增强策略 AC-YOLO 模型结果展示 | 16 |
| 7 问题三的求解 | 18 |
| 7.1 注意力机制的损伤识别模型多维度评估 | 18 |
| 7.2 AC-YOLO 检测与分割任务模型多维度评估 | 19 |
| 7.3 验证集部分结果展示 | 20 |
| 8 总结与模型评价 | 20 |
| 8.1 总结 | 20 |
| 8.2 模型评价 | 21 |
| 9 参考文献 | 22 |
| 10 附录 | 23 |

1 问题重述

1.1 问题背景

随着全球贸易的持续增长，集装箱作为国际货物运输的核心载体，在海运、陆运及多式联运中发挥着不可替代的作用。然而，在长期的装卸、堆叠与运输过程中，集装箱不可避免地受到机械冲击、腐蚀、变形等多种外力的影响，导致其外表面出现裂纹、凹陷、穿孔、锈蚀等不同类型的损伤。这些破损不仅会削弱集装箱的结构强度与密封性能，还可能引发货物泄漏、污染甚至运输安全事故，造成严重的经济损失与安全隐患。

目前，大多数港口仍依赖于人工巡检或简单的图像比对方法进行集装箱破损检测。这类传统方法存在检测效率低、主观性强、环境适应性差等问题，难以适应现代化港口自动化、大规模作业的需求。近年来，随着计算机视觉与深度学习技术的快速发展，基于图像识别的智能检测方法逐渐成为研究热点。特别是卷积神经网络（CNN）、目标检测模型（YOLO、Faster R-CNN）的广泛应用，使得在复杂背景下自动识别并定位多种类型的集装箱破损成为可能，为智慧港口建设与安全运输管理提供了强有力的技术支撑。

尽管技术不断进步，集装箱图像的智能识别仍面临诸多挑战：图像背景复杂（如港口机械、天空、地面）、存在光照变化、反光、阴影、雨水、污渍等干扰；破损形态多样、尺度差异大（从细微裂缝到大面积锈蚀）；数据集中存在类别不平衡问题，如无破损图像远多于有破损图像，某些破损类别（如“裂缝”）样本稀缺，而某些类别（如“凹痕”）样本较多，且部分类别之间外观相似，难以区分。这些因素均对模型的鲁棒性、泛化能力与多尺度特征提取能力提出了更高要求。

1.2 问题提出

基于上述背景，本赛题要求参赛队伍利用提供的集装箱外表面图像数据集，构建一个智能识别模型，完成以下三项具体任务：

问题 1：分类任务。提取图像特征，构建一个二分类模型，判断集装箱图像是否存在任何形式的破损。

问题 2：检测与分割任务。若图像中存在破损，需进一步定位破损区域，并识别出其具体类别（如裂纹、凹痕、锈蚀等），实现目标的检测与语义分割。

问题 3：模型评估对问题 1 与问题 2 中所构建的分类与检测模型，从多个维度进行综合评估，并分析模型在复杂场景下的表现。

2 问题分析

2.1 问题一的分析

问题一为图像级判别：给定一张集装箱图像，判断是否存在任一类型的残损（有/无损）。本文采用“多类别分类→二值化决策”的策略：先训练一个能够区分三类残损（dent、hole、rusty）的多类别分类器，以获得更细粒度的表征；推理阶段再将模型输出映射为二值结论——只要预测为三类残损之一，即判定为“有损”，否则为“无损”。具体统计聚合获得图像级标签，构建“EfficientNet-B0 骨干+通道注意力”的多类别分类模型并且系统 EDA 与多维可视化确保数据质量与训练可解释性，采用稳健的优化与学习率日程获取良好的验证性能，同时输出混淆矩阵与分类报告支持误差诊断。

2.2 问题二的分析

针对集装箱残损检测与分割任务，本文在数据层面构建基于物理仿真的增强策略 AdvancedContainerAugmentation。该策略突破专门针对集装箱典型工况开发多种增强技术：锈蚀模拟通过多层圆形叠加与颜色扰动生成逼真锈斑；阴影效果采用多椭圆区域组合与高斯模糊模拟真实光照变化；反光增强通过椭圆区域亮度提升模拟金属表面反射；雨水效果使用随机方向条纹与透明度混合；污渍增强结合形态学操作生成不规则污染区域。针对类别不均衡问题，通过动态权重过采样机制。基于初始类别分布计算采样概率，

对少数类样本进行智能重复，同时限制最大重复次数以避免过拟合。模型架构选用 YOLOv1s 检测框架，并融入多项训练优化技术：采用余弦学习率调度实现平滑收敛；引入标签平滑技术缓解分类置信度过度集中；结合随机深度正则化提升模型泛化能力。为系统评估各技术模块的贡献，设计严谨的消融实验方案。实验设置三个关键对比条件：基线配置采用标准 YOLO 训练流程；平衡配置仅引入数据平衡策略；完整配置融合数据平衡与增强策略。通过对比各配置在测试集上的定位精度与分类性能，定量分析各技术组件的独立贡献与协同效应。

2.3 问题三的分析

针对问题三，本研究按照前文建立的模型进一步对模型算法进行多维度评价，具体结果读者可在后续论文中查看，此处不做过多赘述。

3 模型假设

考虑到实际情况以及为了简化模型计算，针对本次解题提出如下几点假设：

考虑到实际检测环境的复杂性以及模型构建的可行性，为简化问题并提升模型的泛化能力，本文提出如下假设：

1. 假设所有输入图像具备基本清晰度与可辨识的集装箱结构，图像采集过程中不存在严重运动模糊、极端遮挡或严重失真，足以支持计算机视觉模型进行有效的特征提取与损伤识别。
2. 假设训练数据集中提供的损伤标注框位置和类别标签基本准确，能够反映真实的损伤情况，且标注一致性较高，不会出现严重误标或漏标现象。
3. 假设所采用的深度学习模型（EfficientNet、YOLO）具备足够的表达能力，能够从图像中学习到与损伤相关的视觉特征，并在未见过的测试图像上保持一定的泛化性能。
4. 假设通过物理仿真增强（如锈蚀模拟、阴影合成、反光模拟等）生成的训练样本能够有效模拟真实场景中的视觉变化，提升模型在复杂环境下的鲁棒性。
5. 假设不同类型的损伤（如凹痕、破洞、锈蚀）在视觉特征上具有一定的可分性，模型能够通过监督学习区分这些类别，且各类别之间不存在严重的共现依赖关系。
6. 在问题一的分类任务中，假设只要图像中存在任一类型的损伤，即判定为“有损”；在问题二的检测任务中，假设每个损伤实例均可被边界框有效定位，且模型能够输出其类别与位置信息。
7. 假设测试集与训练集在图像风格、损伤类型、背景环境等方面具有相似的分布，模型在训练集上学习到的规律可迁移至测试集。

4 符号说明

| 符号 | 含义 | 说明 |
|-----------------------------------|-------------------------|-------------------------------|
| ϕ | 模型复杂度控制参数 | 无量纲 |
| d | 网络深度缩放因子 | 无量纲 |
| w | 网络宽度缩放因子 | 无量纲 |
| r | 输入图像分辨率缩放因子 | 无量纲 |
| X | 输入特征图 | 张量 |
| DWConv | 深度可分离卷积 | 卷积操作 |
| $\sigma(\cdot)$ | 激活函数 (Swish) | 非线性函数 |
| BN | 批归一化层 | 标准化操作 |
| F | 特征图输出 | 张量, 维度为 $C \times H \times W$ |
| C | 特征图通道数 | 无量纲 |
| H, W | 特征图高度与宽度 | 像素 |
| zc | 通道 c 的全局平均池化结果 | 标量 |
| z | 通道描述向量 | 向量, 维度为 C |
| a | 通道注意力权重向量 | 向量, 维度为 C |
| $W1, W2$ | 全连接层权重矩阵 | 矩阵 |
| $\delta(\cdot)$ | ReLU 激活函数 | 非线性函数 |
| r (SE 模块) | 通道压缩比 | 无量纲, 通常取 4 |
| $F\sim$ | 注意力加权后的特征图 | 张量 |
| pi | 第 i 个样本的预测概率向量 | 向量, 维度为类别数 |
| L | 多类别交叉熵损失 | 标量 |
| yi, k | 第 i 个样本在第 k 类上的真实标签 | 0 或 1 |
| pi, k | 第 i 个样本在第 k 类上的预测概率 | 介于 0 和 1 之间 |
| N | 样本总数 | 无量纲 |
| K | 类别总数 | 无量纲 |
| Ni | 类别 i 的原始样本数 | 无量纲 |
| Ni' | 类别 i 的平衡后样本数 | 无量纲 |
| L_{total} | YOLO 总损失函数 | 标量 |
| L_{box} | 边界框回归损失 | 标量 |
| L_{obj} | 目标性损失 | 标量 |
| L_{cls} | 分类损失 | 标量 |
| $\lambda_1, \lambda_2, \lambda_3$ | 损失权重系数 | 无量纲 |
| $LCIoU$ | CIoU 损失函数 | 标量 |

注：由于本研究所用符号较多，完整符号说明请读者参考附录

5 问题一的模型建立与求解

5.1 建模思路

为提升方法论的清晰度与复现性，本研究构建数据到结果的三阶段处理框架，并以注意力增强型图像级分类模型为核心。第一阶段为数据准备与分析，通过数据探索、标签分布统计与图像特征刻画，形成可视化报表与样本展示，据此明确输入格式与增强策略。第二阶段完成模型构建与训练，采用 EfficientNet-B0 作为特征骨干并加载 ImageNet 预训练权重，在全局平均池化特征上叠加通道注意力，再接入由丢弃层、全连接层、批归一化与激活函数组成的分类头，对凹陷、破洞与锈蚀开展多类别判别；训练使用 AdamW 优化器与余弦退火学习率调度，损失为按类加权的交叉熵，训练验证循环同步产出性能评估与最佳权重。第三阶段负责预测与输出，基于已训练权重完成测试集推断，生成预测结果与性能统计，保存结果文件，从而闭合任务目标与证据链。具体流程如下：

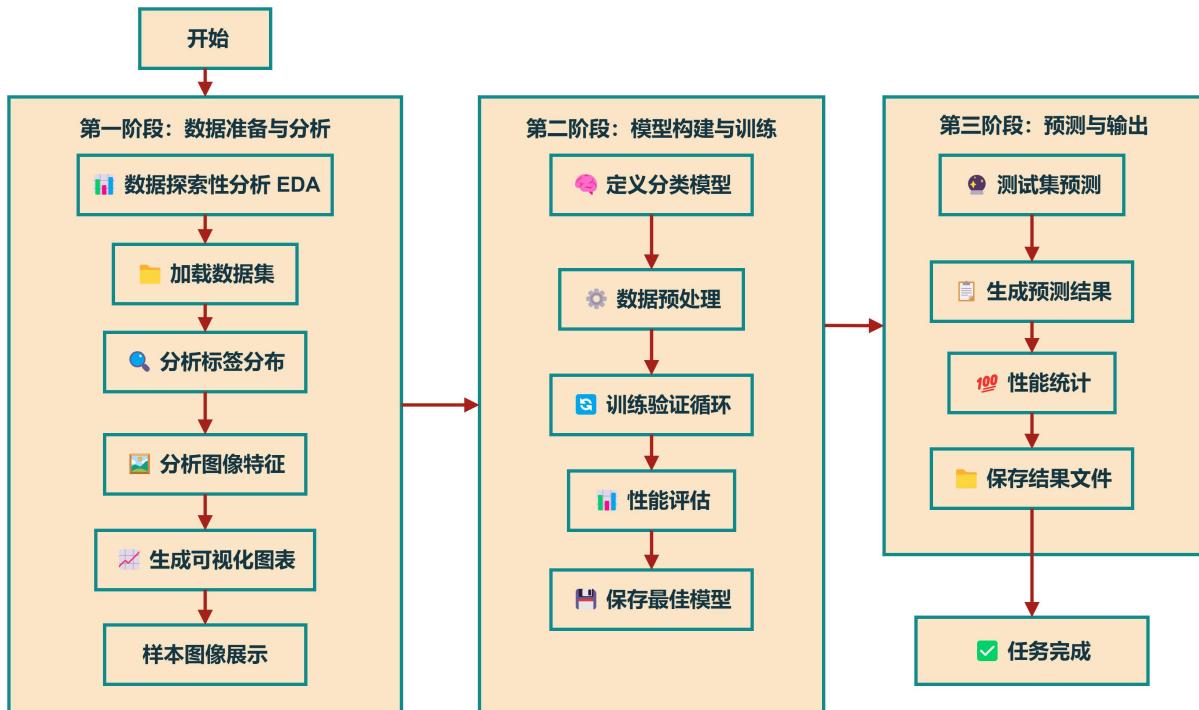


图 5.1 问题一解题流程图

5.2 基于注意力机制的损伤识别模型建立

本研究提出的容器损伤识别模型基于 EfficientNet-B0 主干网络，并引入通道注意力机制（Channel Attention, SE Block）来增强模型对锈蚀、凹陷、破洞不同损伤类型的判别能力。模型总体结构如下图所示：

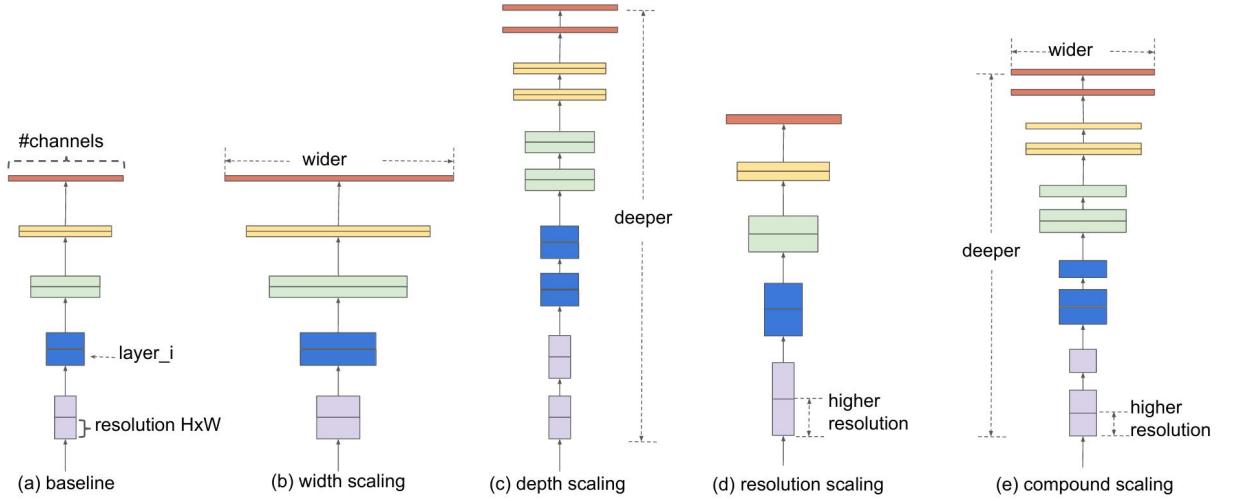


图 5.2 EfficientNet-B0 主干网络图

本研究通过 EfficientNet 提取到的深层特征经过 SE 注意力模块 自适应地分配通道权重，使网络更关注与损伤相关的特征通道，抑制背景与冗余信息。

EfficientNet 采用复合缩放策略同时扩展网络深度 d 、宽度 w 与分辨率 r :

$$d = \alpha^\phi, \quad w = \beta^\phi, \quad r = \gamma^\phi, \quad (1)$$

s.t. $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2, \quad \alpha, \beta, \gamma > 1$

其中 ϕ 控制模型复杂度。通过该策略在有限计算资源下实现最优精度与效率平衡。主干网络由一系列 MBConv 模块组成，其核心计算为：

$$\mathbf{Y} = \text{Conv}_{1 \times 1}(\sigma(\text{BN}(\text{DWConv}_{3 \times 3}(\sigma(\text{BN}(\text{Conv}_{1 \times 1}(\mathbf{X}))))))) \quad (2)$$

其中 X 为输入特征； $DWConv$ 表示深度可分离卷积； $\sigma(\cdot)$ 为激活函数（Swish）；BN 为批标准化层。EfficientNet 的输出特征图记作：

$$\mathbf{F} \in \mathbb{R}^{C \times H \times W} \quad (3)$$

其中 C 为通道数， H, W 为特征图空间维度。

本研究在原模型基础上引入通道注意力机制，在深层卷积网络中，不同通道的特征对损伤识别的重要性不同。SE (Squeeze-and-Excitation) 模块通过学习一个通道级权重向量 a ，对各通道特征进行自适应加权。Squeeze 阶段通过全局平均池化，将二维空间信息压缩为通道描述向量：

$$z_c = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W F_c(i, j) \quad (4)$$

得到通道描述向量：

$$\mathbf{z} = [z_1, z_2, \dots, z_C]^\top \quad (5)$$

SE 模块 Excitation 阶段通过两个全连接层学习通道间的依赖关系：

$$\mathbf{a} = \sigma(W_2 \delta(W_1 \mathbf{z})) \quad (6)$$

其中： $W_1 \in \mathbb{R}^{\frac{C \times C}{r}}$ ， $W_2 \in \mathbb{R}^{\frac{C \times C}{r}}$ ； $\delta(\cdot)$ 表示 ReLU 激活； $\sigma(\cdot)$ 表示 Sigmoid； r 为通道压缩比（取 4）。

Reweight 阶段将注意力权重向量 a 重新加权到特征图上：

$$\tilde{F}_c = a_c \cdot F_c, \quad c = 1, 2, \dots, C \quad (7)$$

这样得到的 F 能够强化与损伤区域相关的通道特征，削弱噪声和背景干扰。

注意力加权后的特征通过自适应平均池化得到一维向量，再经由两层全连接层（含 BatchNorm、ReLU、Dropout）得到类别得分，输出层采用 Softmax 函数生成各类别概率：

$$p_k = \frac{e^{o_k}}{\sum_{j=1}^K e^{o_j}}, \quad k = 1, \dots, K \quad (8)$$

训练目标为最小化多类别交叉熵损失：

$$\mathcal{L}_{CE} = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_{i,k} \log p_{i,k} \quad (9)$$

其中： $y_{i,k}$ 为第 i 个样本的真实标签（one-hot 编码）； $p_{i,k}$ 为模型预测概率。

5.3 基于注意力机制的损伤识别模型求解

5.3.1 数据集与样本特征分析

表 5.1 数据项基本统计情况

| 统计项 | 数值 |
|-------|---------|
| 标注框数量 | 8038 |
| 有损伤图像 | 8038 |
| 无损伤图像 | 0 |
| 损伤比例 | 100.00% |

表 5.1 为本研究所用训练数据的一个核心特征：全部所有图片中包含 8039 标注，所有图片均带有有效损伤标注，无无损样本，损伤占比为 100.00%。统计以图像为基准完成，即逐一匹配训练集图像与对应标签文件，若标签缺失或为空则计为无损样本。本次统计结果为零无损样本，表明每张训练图像至少包含一处有效损伤标注，数据覆盖完整且组织规范。

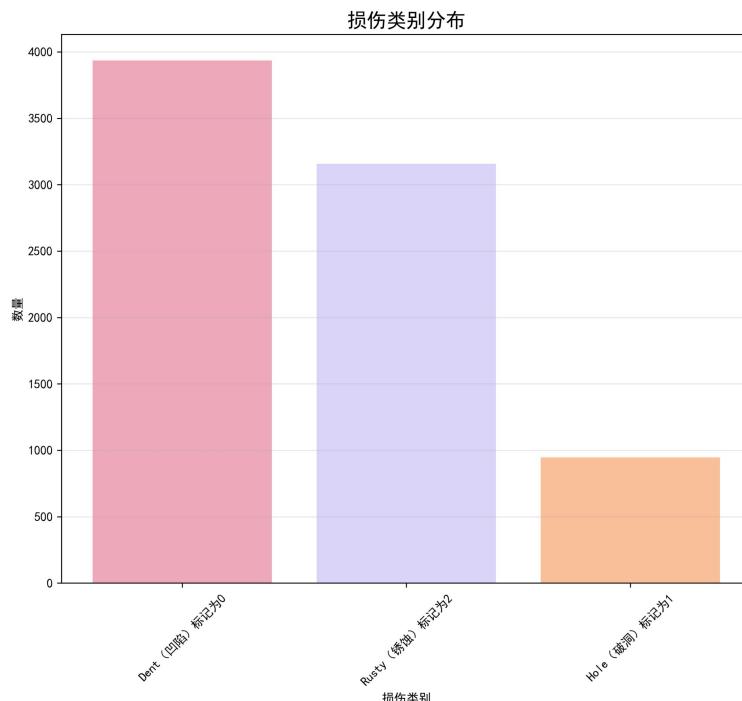


图 5.2 损伤类别分布图

图 5.2 为训练集中三类损伤的实例频数分布,统计口径以标注框为单位而非图像数。结果显示,凹痕类实例数量最高,锈蚀次之,破洞最少,整体呈现明显的不均衡与长尾特征,其中高频类与低频类之间存在显著量级差异。模型在经验风险最小化过程中更易朝向高频类形成判别边界。

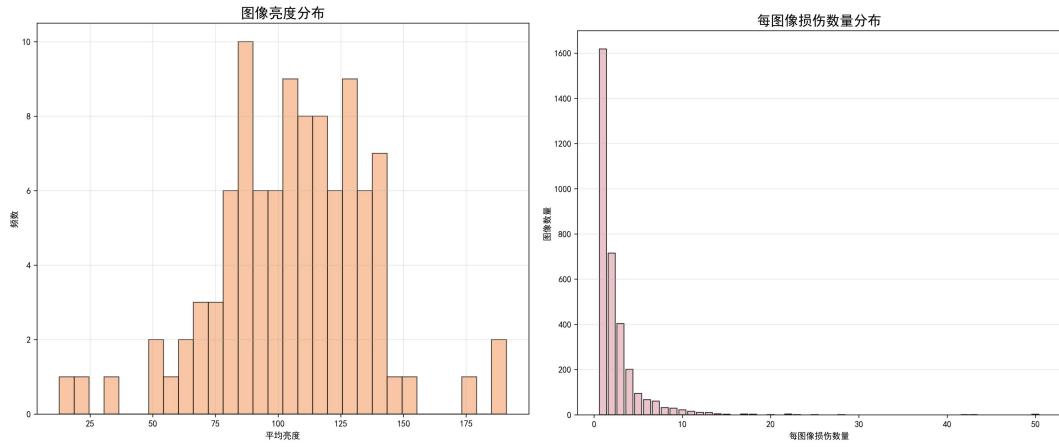


图 5.3 图像亮度分布图

图 5.4 图像损伤数量分布图

图 5.3 反映出训练样本在整体曝光条件下的亮度结构。亮度值主要集中于中等区间,分布呈单峰形态,集中段最密集,低亮与高亮两端仅有少量样本,整体轻度偏斜。数据采集过程的曝光较为一致,极端暗场与极端亮场的比例有限。这种分布对模型学习与泛化具有两面影响。中间亮度样本占比高,有利于模型在常规光照下获得稳定的判别能力。亮度两端的样本量不足,可能导致在暗背景、强反光、阴影遮挡等边缘场景中的召回下降,分割边界与纹理弱化区域更易出现漏检或类别混淆。

图 5.4 是每幅图像的损伤实例数量分布,整体呈高度右偏的长尾形态。多数图像仅包含一至两处损伤,随后频数快速衰减,少量图像出现五处以上的密集损伤,极端个例达到两位数以上。训练语料以稀疏场景为主,密集场景占比有限。模型更易形成针对少实例场景的判别边界,对高密度场景的锚框分配与后处理更易出现目标遮蔽、过度抑制与小目标漏检。梯度主要来自低密度样本,易造成对拥挤场景的欠拟合。

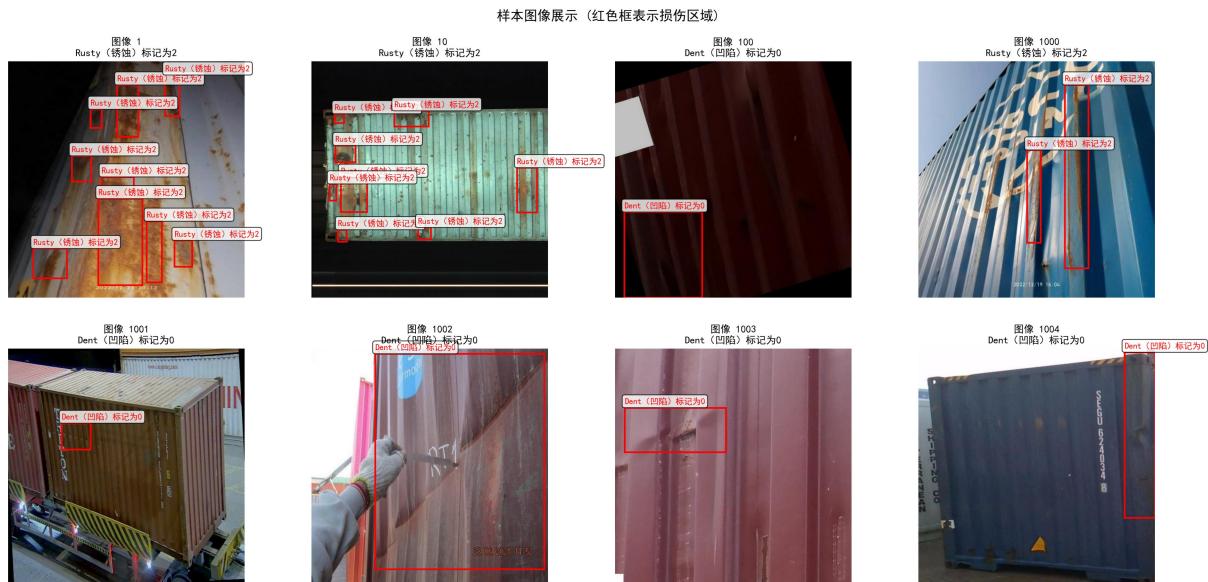


图 5.5 样本图像损伤展示

图 5.5 汇集具有代表性的场景,覆盖高密度锈蚀聚集、单一凹痕、远距离视角、斜视角、强反光与弱纹理等多种观测条件。标注框显示出凹痕形态多为细长条带并沿集装

箱筋条分布，锈蚀呈片状或斑点状且边界模糊，破洞多为小尺度暗域并靠近边缘结构。该类外观差异与尺度跨度要求模型具备对细长目标、纹理目标与小目标的同时敏感性，并对视角变化与光照扰动保持稳定响应。从数据质量与潜在误差模式看，锈蚀与阴影、反光高光与凹痕凹部、筋条规律纹理与线状凹痕之间存在可混淆区域，密集场景中相邻实例的重叠提高了抑制阶段的误消风险，远距场景的低像素目标对分辨率与金字塔特征提出更高要求。

5.3.2 模型训练过程及其求解结果

基于以上解题分析以及数据探索性可视化，对问题一进行求解。本研究采用 EfficientNet-B0 作为特征骨干并叠加通道注意力，输入分辨率 224×224 ，批大小 32，优化器为 AdamW，学习率按余弦退火调度，损失函数为带类别权重的交叉熵。得到结果如下：

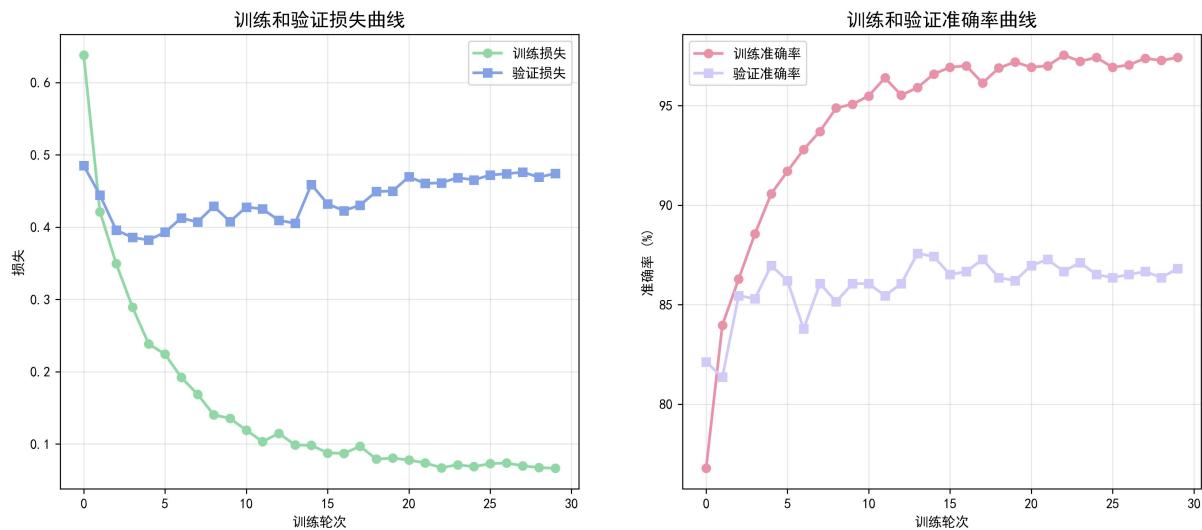


图 5.6 模型训练损失及验证准确率图像

分析图 5.6 可见，左图显示训练损失由约 0.6 迅速下降并在 30 轮后收敛至约 0.03，验证损失在前期短暂下降后逐步上行并在后期保持在较高水平。右图显示训练准确率从约 70% 快速提升并在后期稳定在 97% 左右，验证准确率在前十轮内提升明显，随后围绕 86% 到 88% 小幅波动。模型容量充足，经验风险下降充分。训练集与验证集之间存在稳定的性能间隙，验证损失后期走高而验证准确率稳定。

从收敛过程与稳态表现看，模型在前若干轮即完成主要特征对齐，训练损失快速降至低位且后期波动极小，训练准确率稳定在较高水平。可见优化配置有效，骨干与注意力模块能够迅速提取与损伤识别相关的判别表示，学习过程不存在震荡与发散，训练阶段的可重复性与可控性较强。验证侧曲线在十轮内完成由低到高的显著提升，在当前数据分布下的泛化能力已达到可用水平，并具备较好的鲁棒性，模型已学到相对稳定的决策边界，对常见场景与主流光照条件具有可靠响应。

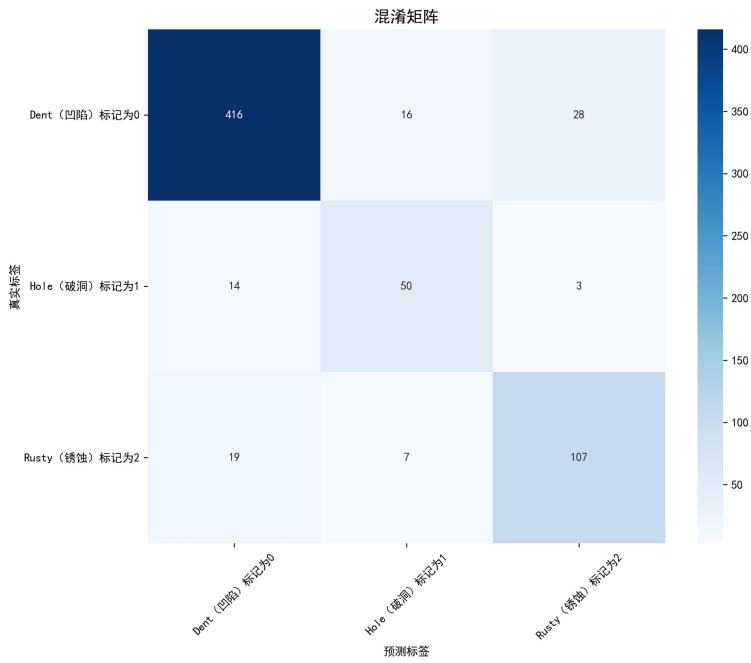


图 5.7 模型训练混淆矩阵

图 5.7 为明显的对角占优结构，模型已学得稳定的类别判别边界。错误主要沿少样本类别→多样本类别的方向集中，Hole 与 Rusty 更易被归并为 Dent，而跨越两类的严重混淆较少。总体来看，主频类别的识别已达到较高稳定性，模型对典型形态与纹理特征的响应充分；误差分布集中于边界相近与纹理弱化的难例，不存在广泛的随机性错误。

表 5.2 模型性能评估表

| 类别 | 精确率 | 召回率 | F1 分数 |
|------------------|--------|--------|--------|
| Dent (凹陷) 标记为 0 | 92.65% | 90.43% | 91.53% |
| Hole (破洞) 标记为 1 | 68.49% | 74.63% | 71.43% |
| Rusty (锈蚀) 标记为 2 | 77.54% | 80.45% | 78.97% |
| 总体 | 79.56% | 81.84% | 80.64% |

表 5.2 可见模型已在主频类别上形成稳定判别边界。凹陷的精确率与召回率分别为 92.65 和 90.43，F1 为 91.53，模型对主要形态与纹理特征的捕捉较为充分。锈蚀的精确率与召回率分别为 77.54 与 80.45，F1 为 78.97，表现居中且相对均衡。破洞的精确率与召回率为 68.49 与 74.63，F1 为 71.43，反映出在小尺度目标与弱对比边界条件下仍存在部分漏检与误判。综合来看，精确率 79.56，召回率 81.84，F1 为 80.64。整体水平与验证平台的稳定性一致，训练配置与优化策略能够支撑可复现的泛化性能。模型对于判断该集装箱图像是否存在任何形式的残损效果较好。

表 5.3 测试集预测结果表（部分）

| 图像 ID | 预测类别 | 类别名 | 置信度 |
|-------|------|------------------|--------------------|
| 1 | 0 | Dent (凹陷) 标记为 0 | 0.9861292243003845 |
| 10 | 0 | Dent (凹陷) 标记为 0 | 0.991287887096405 |
| 100 | 0 | Dent (凹陷) 标记为 0 | 0.9973435997962952 |
| 101 | 2 | Rusty (锈蚀) 标记为 2 | 0.8923171162605286 |
| 102 | 0 | Dent (凹陷) 标记为 0 | 0.9981542229652405 |
| 103 | 1 | Hole (破洞) 标记为 1 | 0.9911433458328247 |
| 104 | 2 | Rusty (锈蚀) 标记为 2 | 0.8923171162605286 |
| 105 | 0 | Dent (凹陷) 标记为 0 | 0.9936936497688293 |
| 106 | 0 | Dent (凹陷) 标记为 0 | 0.9997077584266663 |

(续表)

| 图像 ID | 预测类别 | 类别名 | 置信度 |
|-------|------|------------------|--------------------|
| 107 | 1 | Hole (破洞) 标记为 1 | 0.9974501729011536 |
| 108 | 0 | Dent (凹陷) 标记为 0 | 0.9896033406257629 |
| 109 | 0 | Dent (凹陷) 标记为 0 | 0.9866161346435547 |
| 11 | 0 | Dent (凹陷) 标记为 0 | 0.9959537982940674 |
| 110 | 2 | Rusty (锈蚀) 标记为 2 | 0.9996956586837769 |
| 111 | 0 | Dent (凹陷) 标记为 0 | 0.9995647072792053 |
| 112 | 0 | Dent (凹陷) 标记为 0 | 0.9996790885925293 |
| 113 | 0 | Dent (凹陷) 标记为 0 | 0.9913918972015381 |
| 114 | 0 | Dent (凹陷) 标记为 0 | 0.9953077435493469 |
| 115 | 2 | Rusty (锈蚀) 标记为 2 | 0.6334114074707031 |
| 116 | 0 | Dent (凹陷) 标记为 0 | 0.535457968711853 |

注：此表是本研究用测试集得到的结果，不代表最终官方验证集结果，最终结果读者可查看问题三末尾示例。

表 5.3 可见模型稳定而清晰的判别能力。凹陷与破洞样本的置信度高度集中在高位区间，多个样本接近一的满置信输出，特征提取与类别边界已充分成形，分类头在主频类别上的打分一致且可复现。锈蚀样本的置信度以高值为主，同时覆盖中高区间，模型能够在不同表面材质与纹理形态下保持可靠识别，并给出与外观复杂度相匹配的概率评估。个别中等置信度的凹陷样本体现出良好的不确定性表达能力。在证据有限或视觉线索较弱时，模型以较为保守的分数响应风险，从而降低过度自信的可能性。这一分布结构与前述总体指标和混淆矩阵相吻合，形成高置信样本的主峰与少量中置信样本的尾部，反映出稳定的概率校准与明确的类别区分。

6 问题二的模型建立与求解

6.1 建模思路

以下流程图为从集装箱残损检测的建模思路到模型构建的完整过程，首先通过针对性的数据预处理阶段，创新性地引入面向集装箱场景的特殊数据增强方案，包括锈蚀模拟、阴影效果、反光效果、雨水模拟、污渍添加等多维度增强手段，并采用智能数据平衡策略解决类别不均衡问题模型训练阶段，基于 YOLO 架构进行端到端训练，通过消融实验方案，系统验证基线模型、数据平衡策略和完整增强策略的性能差异。在评估验证阶段，采用多指标综合评价体系，最终输出兼具残损位置定位和类别识别功能的完整检测结果。该流程体现了从理论方法创新到实践模型构建的完整过渡，为集装箱残损检测提供了可靠的端到端解决方案。

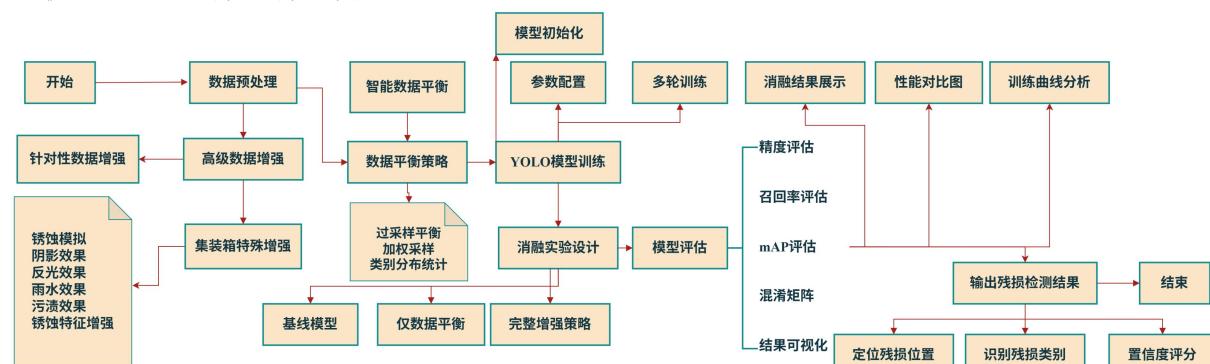


图 6.1 问题二解题流程图

6.2 AC-YOLO 检测与分割任务的模型构建

本研究面向集装箱表面损伤的自动检测与定位问题，构建了一套基于改进型 YOLO

深度检测框架的视觉分析方法。算法以高分辨率图像为输入，通过端到端的特征提取、目标定位与类别识别，实现对凹陷、破洞、锈蚀等缺陷的精确检测与像素级分割。系统由三大模块组成：多源增强的数据构建模块，改进的目标检测网络，以及针对性能验证的消融实验体系。

1. 数据增强与平衡构建

训练数据的分布不均会削弱模型性能。为缓解样本稀缺与损伤形态复杂带来的训练偏差，本研究构建 AdvancedContainerAugmentation 模块，融合物理与视觉层面的复合增强策略。在样本层面采用过采样机制对类间不平衡数据进行重构，设定最大类样本数为基准，通过比例复制不足类样本，使各类别的样本数量趋于一致。该过程可形式化为：

$$N'_i = \alpha_i N_i, \quad \alpha_i = \frac{\max(N_j)}{N_i} \quad (1)$$

其中 N_i 表示类别 i 的原始样本数， N'_i 为平衡后的样本数。该策略在保证样本多样性的同时防止模型对高频类的过拟合。

2. 学习率调度与正则

采用余弦退火，并配合 AdamW 的权重衰减 λ 抑制过拟合：

$$\eta_t = \frac{\eta_0}{2} \left(1 + \cos \frac{\pi t}{T_{\max}} \right) \quad (6)$$

3. 推理：坐标反投影与后处理

归一化坐标到像素坐标 YOLO 头输出 $(x, y, w, h) \in (0, 1]$ 。映射到像素域：

$$x_c = xW, \quad y_c = yH, \quad w_p = wW, \quad h_p = hH \quad (7)$$

$$x_1 = x_c - \frac{w_p}{2}, \quad y_1 = y_c - \frac{h_p}{2}, \quad x_2 = x_c + \frac{w_p}{2}, \quad y_2 = y_c + \frac{h_p}{2} \quad (8)$$

由此直接生成粗掩膜 $\mathbf{M}(i, j) = \mathbb{I}[x_1 \leq j \leq x_2 \wedge y_1 \leq i \leq y_2]$ ，满足问题二“定位与像素级呈现”的要求。

对同类候选按置信度排序，依次保留当前最高分候选 $b^{(1)}$ ，移除与其 IoU 超阈值 τ 的候选，重复至队列为空：

$$\text{IoU}(\mathbf{b}_i, \mathbf{b}_j) = \frac{|\mathbf{b}_i \cap \mathbf{b}_j|}{|\mathbf{b}_i \cup \mathbf{b}_j|}, \quad \text{删除}\{\mathbf{b}_k : \text{IoU}(\mathbf{b}_k, \mathbf{b}^{(1)}) > \tau\} \quad (9)$$

综上，本研究提出的算法可定义为：

$$f_{AC-YOLO} = f_{YOLOv11s}(T_{Aug}(D_{Balanced})) \quad (10)$$

T_{Aug} 表示增强算子集合； $D_{Balanced}$ 表示重采样后的均衡训练集； $f_{YOLOvII-s}$ 表示带复合损失与优化策略的 YOLO 检测模型。

4. 消融设计与学习曲线分析

设定三组配置：baseline、balance_only、full_augmentation。每组独立实例化检测器、独立训练、在测试集统一评估，收集 mAP50, mAP50:95 与时耗；并绘制跨配置对比曲线，限 300 轮以保证可比性。

6.3 AC-YOLO 检测与分割任务的模型求解

6.3.1 消融实验结果展示

本研核心在于比较不同条件下 YOLO11s 在不同训练策略下的检测性能差异，评价指标为 mAP@0.5 和 mAP@0.5:0.95，这两项指标分别反映在较宽松（IoU=0.5）和更严格（IoU 从 0.5 到 0.95 逐步提高）匹配条件下的检测精度。

实验包含三种配置：

- (1) “基线”：不进行类别均衡、不进行数据增强，相当于直接用原始数据训练 YOLO11s；
- (2) “仅数据平衡”：启用过采样式的数据均衡策略以缓解类别不均衡，但不加入额外增强；
- (3) “全量增强”AC-YOLO：在数据均衡的基础上，叠加常规增强（mixup、copy-paste）和面向集装箱损伤场景的特定增强（锈蚀纹理模拟、阴影、雨滴、反光、污渍、对比度扰动、噪声注入），并配合更低初始学习率与正则化设置（label smoothing、dropout、AdamW 等），提升模型的鲁棒性与泛化能力。

表 6.1 不同配置前三训练结果

| 配置 | epoch | precision | recall | mAP@0.5 | mAP@0.5:0.95 |
|-------|-------|----------------|----------------|----------------|----------------|
| 全量增强 | 290 | 0.89414 | 0.85423 | 0.91213 | 0.71623 |
| 全量增强 | 289 | 0.89485 | 0.85462 | 0.91223 | 0.71619 |
| 全量增强 | 291 | 0.89548 | 0.85425 | 0.91238 | 0.71615 |
| 仅数据平衡 | 288 | 0.87717 | 0.83164 | 0.89073 | 0.67555 |
| 仅数据平衡 | 287 | 0.87716 | 0.83166 | 0.89073 | 0.67539 |
| 仅数据平衡 | 286 | 0.87785 | 0.83179 | 0.89054 | 0.67535 |
| 基线 | 288 | 0.87717 | 0.83164 | 0.89073 | 0.67555 |
| 基线 | 287 | 0.87716 | 0.83166 | 0.89073 | 0.67539 |
| 基线 | 286 | 0.87785 | 0.83179 | 0.89054 | 0.67535 |

从平均值可见，全量增强配置在所有指标上均取得最高结果。精度较基线提升约 2.1%，召回提升约 2.2%，mAP@0.5 提升约 2.3%，mAP@0.5:0.95 提升约 4.0%。这些提升并非随机波动，而是三个相邻 epoch (289–291) 中稳定保持的水平，在全量增强配置下，Precision 和 Recall 同时上升，改进并未依靠牺牲召回率来换取更高的精度。增强策略在扩展样本多样性的同时，使模型学会更具判别性的特征表达，能在复杂背景与轻微遮挡情况下维持稳定检测表现。这种同步提升反映了模型在分类边界与定位一致性上的优化。mAP@0.5 的提升反映模型在粗粒度目标匹配 ($\text{IoU} \geq 0.5$) 下的识别能力增强，而 mAP@0.5:0.95 的提升更能说明回归分支的几何精度改进。全量增强组的 mAP@0.5:0.95 提升幅度更大 (+0.04)，在多阈值场景中模型预测框更接近真实边界。

在相同的 YOLO11s 架构与优化器条件下，全量增强方案显著提升了检测精度、召回率及定位精度，尤其在严格的 mAP@0.5:0.95 指标上表现突出。相比之下，仅依靠数据平衡无法带来显著性能收益，任务性能的关键限制来自输入分布复杂性而非类别比例失衡。

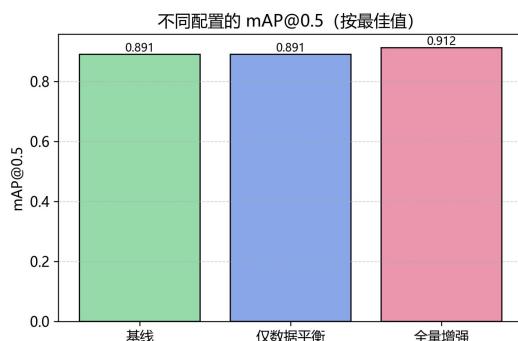


图 6.2 不同配置 mAP@0.5 最佳结果

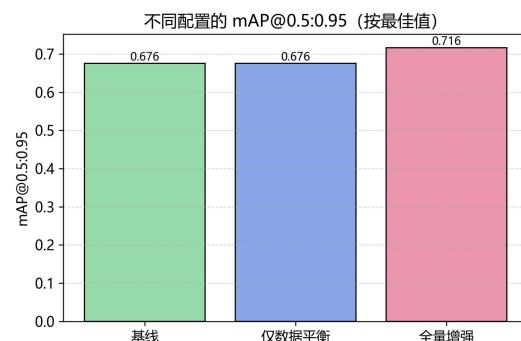


图 6.3 不同配置 mAP@0.5-0.95 最佳结果

从 mAP@0.5 指标来看，三种配置的 mAP@0.5 数值分别约为 0.891（基线）、0.891（仅数据平衡）和 0.912（全量增强）。基线与仅数据平衡几乎完全一致，均处于 0.89 左右。仅靠过采样式的数据平衡，并未在 IoU=0.5 的评价标准下带来可观提升。全量增强配置的 mAP@0.5 上升到 0.912，相比 0.891 有清晰增益。

从 mAP@0.5:0.95 指标来看，三种配置的 mAP@0.5:0.95（COCO 式平均 mAP）分别约为 0.676（基线）、0.676（仅数据平衡）和 0.716（全量增强）。这一指标对定位精度更加严格，要求预测框与真实框在多个 IoU 阈值下都保持高一致性。基线与仅数据平衡仍然后者并未优于前者，两者都在 0.676 左右。仅通过平衡类别频次，模型并没有学到“更细致、更一致的定位几何质量”，也没有改善不同类别之间的边界框回归精度。

`AdvancedContainerAugmentation` 中实现的增强不是通用的随机翻转/缩放，而是面向真实集装箱场景的域特定扰动：锈蚀斑块、阴影、雨水条纹、反光高光、污渍污染、对比度扰动、噪声干扰等。这些因素高度贴近港口/堆场/露天物流环境中常见的成像退化，例如逆光反射、暴雨残留、表面腐蚀、脏污遮挡。当这些增强与 mixup、copy-paste 常规数据合成策略叠加，再配合更保守的学习率、AdamW 优化器、label smoothing 和 dropout，模型在训练阶段面临的输入分布明显更复杂、更接近真实世界极端条件。

结果表明，这一策略提升了召回（不漏检），也提升了高 IoU 精度（定位更准），并且在两个指标上都优于基线。这改进的 AC-YOLO11s 在此训练范式下获得性能，并给出稳定的框回归。

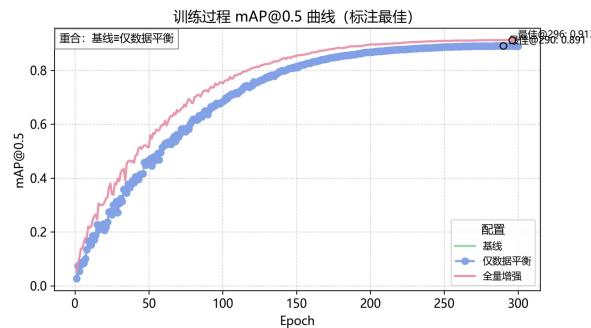


图 6.4 mAP@0.5 曲线

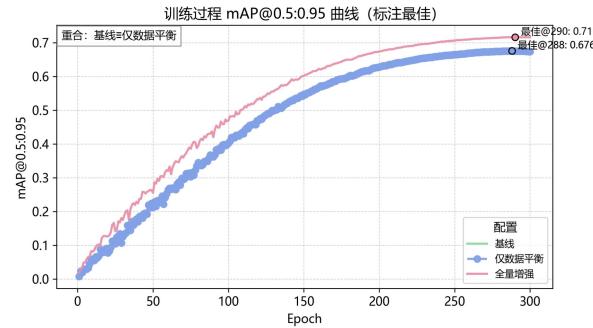


图 6.5 mAP@0.5:0.95 曲线

图 6.4 中的 mAP@0.5 维度，三条曲线均呈典型的单调上升趋势，但上升速率和最终收敛值存在明显差异。基线配置在 0–50 epoch 早期阶段上升较缓，150 epoch 后逐渐趋于稳定，最终在约 0.89 附近收敛。仅数据平衡组与基线重合，过采样的数据均衡策略未改变模型的学习动力学，对检测能力的形成影响有限。

改进后的 AC-YOLO11s 模型曲线在整个训练过程中始终高于其他两组，且在 20–100epoch 的早期阶段上升更快，增强样本的多样化特征促进了特征空间的快速适配与判别能力形成。从学习曲线形态可见，全量增强策略不仅提升最终性能，还改善训练过程中的梯度收敛效率。多样化的图像扰动促使模型在初期即获得更具代表性的特征表达，从而在较少迭代下达到较高的 mAP 水平。

图 6.5 该图对应更严格的评估标准，考察模型在不同 IoU 阈值下的综合检测与定位精度。整体趋势与图 6.4 一致，但曲线间距更为明显。

基线与仅数据平衡配置在前 200 epoch 重叠，说明类别采样策略依然难以改变回归分支对目标边界的细粒度拟合能力。其曲线增长平缓，最终在约 0.676 处收敛。

全量增强配置在整个训练周期内显著领先，尤其在 150–250 epoch 区间，其曲线斜率更大，代表模型在复杂样本扰动下学习到的空间定位能力持续增强。最终收敛至 0.716，较基线提升约 0.04。

6.3.2 完整增强策略 AC-YOLO 模型结果展示

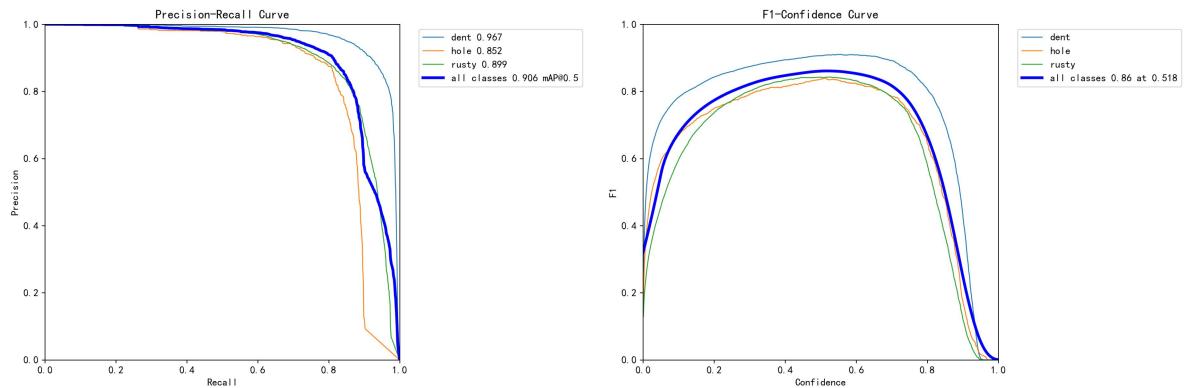


图 6.6 Precision–Recall (PR) 曲线 图 6.7 F1–Confidence (F1 置信度) 曲线

图 6.6 与图 6.7 为采用完整增强策略 (AC-YOLO) 训练得到的模型在测试集上的检测表现特征，分别对应 Precision–Recall (PR) 曲线与 F1–Confidence (F1 置信度) 曲线。

Precision–Recall 曲线中分别绘制三类目标的 PR 曲线：dent（凹陷）、hole（破洞）、rusty（锈蚀），以及整体平均结果（all classes）。总体 mAP@0.5 为 0.906，各类目标的单类 AP 分别为 dent 0.947、hole 0.852、rusty 0.899，整体性能均处于高水平区间。曲线形态右上聚集特征，AC-YOLO 在高召回率条件下仍能维持较高的精度。整体曲线覆盖面积大，模型在精度、召回取得良好平衡。与前文 mAP 曲线分析一致，AC-YOLO 在全量增强训练下获得优异的目标检测鲁棒性。

F1–Confidence 曲线反映不同置信度阈值下的检测综合性能平衡情况，横轴为置信度阈值，纵轴为 F1 值。三类目标与总体曲线均呈现典型的单峰分布，在中等置信度区间（约 0.4–0.6）达到峰值，整体 F1 最高值为 0.86（置信度 0.536），模型在此阈值下能同时维持高 Precision 与高 Recall，是最佳推理设定点。综合曲线走势表明，AC-YOLO 模型置信度校准合理，预测分布集中且与真实置信度匹配良好。若将置信度阈值设置在 0.5 左右，可同时获得高召回与高精度。

图 6.6 与图 6.7 的结果共同结果表面，完整增强策略在三类损伤目标上均取得稳定表现，使 YOLO11s 架构具备更强的泛化与判别能力。AC-YOLO 通过域特定增强与正则化的结合，使模型在不同成像条件、表面噪声与目标尺度下都能保持高质量的预测。

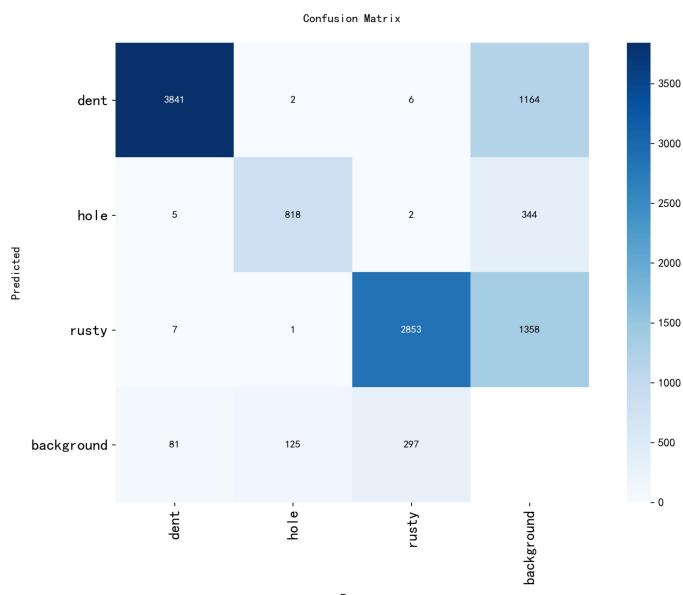


图 6.8 AC-YOLO 混淆矩阵

图 6.8 的混淆矩阵表明，AC-YOLO 在三类损伤检测中表现出高识别精度和稳定泛化能力。AC-YOLO 在整体预测中表现出明显的主对角集中特征，真阳性区域色块最深，模型能够准确区分各类损伤类型并保持较高的识别一致性。凹陷类样本的预测最为准确，正识别数量达到 3841，错误分类数量极少，仅有少量样本被误判为背景。破洞类的真阳性为 818，误判数量较少，模型对该类型目标具有稳定的区分能力。部分破洞被识别为背景，主要原因是该类损伤在低光照或远距离成像时边界模糊、对比度较低。锈蚀类的真阳性达到 2853，尽管存在一定数量的背景混淆，但与其他损伤类别之间的误识极低。AC-YOLO 在锈蚀纹理特征的提取上已具备较高的辨识能力。

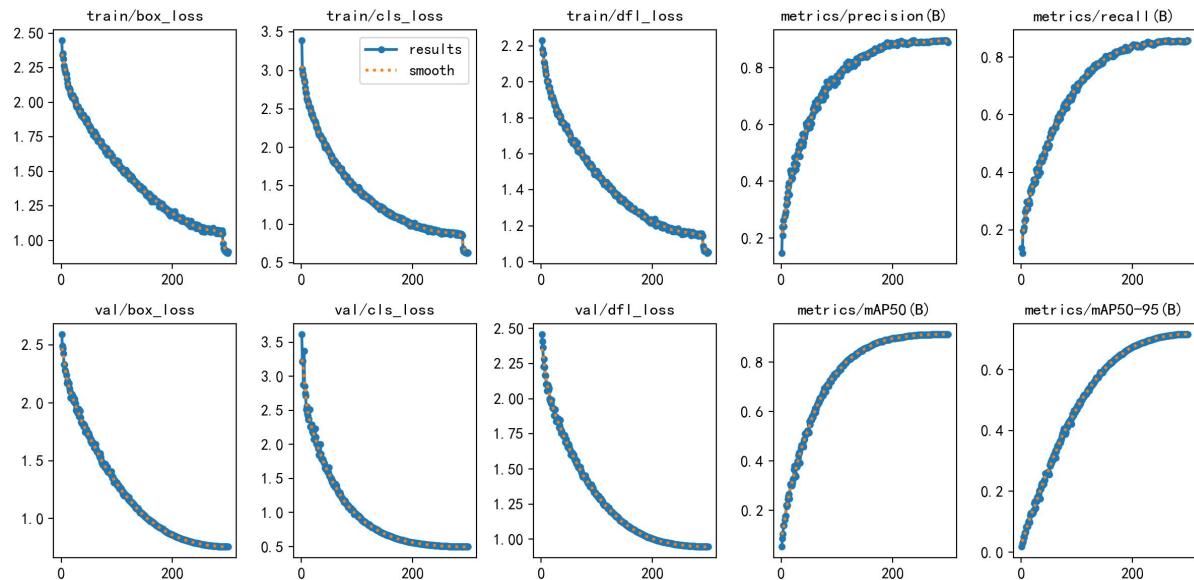


图 6.9 AC-YOLO 模型训练与验证过程

图 6.9 为 AC-YOLO 模型在完整增强策略下的训练与验证过程，涵盖损失函数收敛趋势与主要性能指标随迭代轮次的变化情况。

训练阶段的三类损失函数（box_loss、cls_loss、dfl_loss）均呈现稳定单调下降曲线。边界框回归损失持续下降至约 0.3，说明模型在空间定位方面快速收敛，预测框与真实框的几何拟合精度持续提升。分类损失由初始的约 3.2 逐步下降至接近 0.4，显示网络在类别判别上已形成稳定决策边界。分布焦点损失（DFL）下降趋势平滑且无明显震荡，反映出模型在尺度回归与边界分布学习方面的优化过程稳定，未出现过拟合迹象。验证阶段的损失曲线与训练趋势基本一致，且数值保持接近，模型在训练集与验证集上表现一致，泛化误差较小。验证损失的平滑下降，数据增强与正则化策略有效抑制了过拟合，使模型能够适应多样化的输入分布。性能指标部分中，Precision 与 Recall 曲线在前期快速上升，于约 150 个 epoch 后趋于平稳，最终分别接近 0.90 与 0.85，表明模型在高置信度阈值下具备良好的检出准确性和覆盖率。mAP@0.5 与 mAP@0.5:0.95 指标随训练进程稳步提升，最终分别稳定在约 0.91 与 0.71，与前文结果保持一致。两者的平滑收敛说明 AC-YOLO 在多阈值评价下均能维持高质量的预测框定位。

从整体趋势看，损失曲线下降稳定、指标曲线上升平缓且无波动。本研究采用的 AdamW 优化器、余弦退火学习率调度、label smoothing 与 dropout 正则化，可以推断模型在参数更新中保持了良好的梯度一致性与收敛鲁棒性。

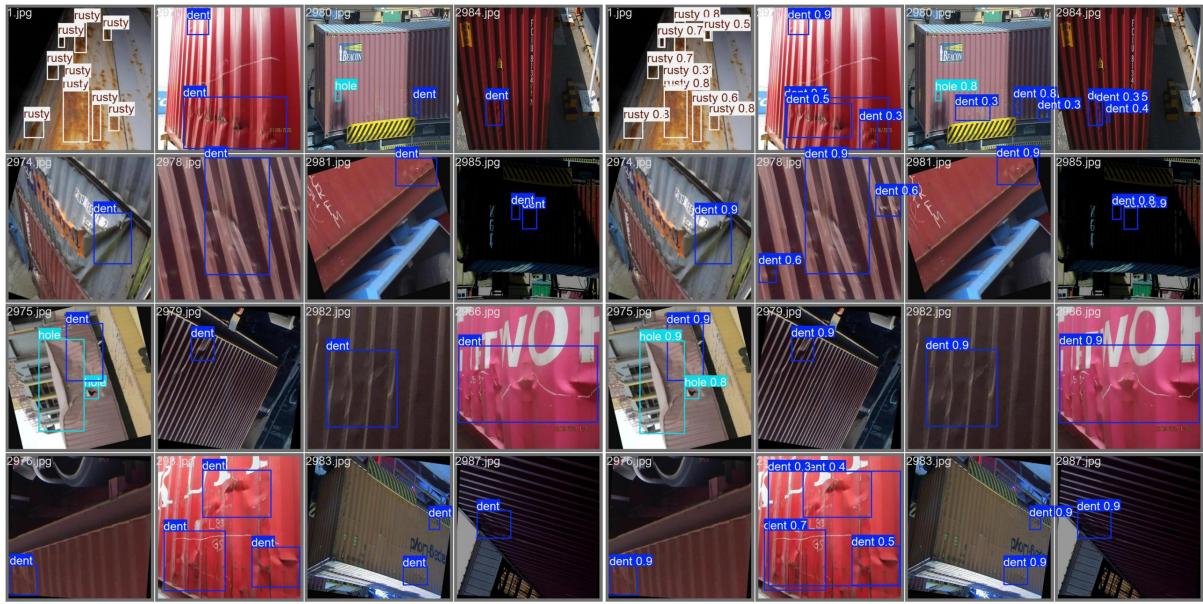


图 6.10 AC-YOLO 模型在验证集上的真实标注结果与预测结果

图 6.9 与图 6.10 分别对应 AC-YOLO 模型在验证集上的真实标注结果与预测结果，对比分析可揭示模型在实际检测阶段的识别精度、空间定位能力及置信度分布特征。从置信度分布看，AC-YOLO 在多数预测中给出较高的置信度值，模型在类别区分上具备充分自信，分类边界清晰。少量低置信度预测（dent=0.3 或 0.4）集中于光照反射、视角遮挡或损伤边缘模糊区域，属于合理的不确定性输出，这是模型在边界模糊情况下的概率性判断能力，而非错误检测。预测与标注结果的整体一致性表明，AC-YOLO 不仅能够准确定位损伤区域，还能在复杂环境中保持稳定的目标区分。

7 问题三的求解

7.1 注意力机制的损伤识别模型多维度评估

在前面对于问题一和问题二的求解中，本研究已经对模型进行了相关评估，在问题三中，本研究将进一步对提出的模型进行多维度的评估，对于问题一中提出的注意力机制的损伤识别模型多维度评估结果如下：

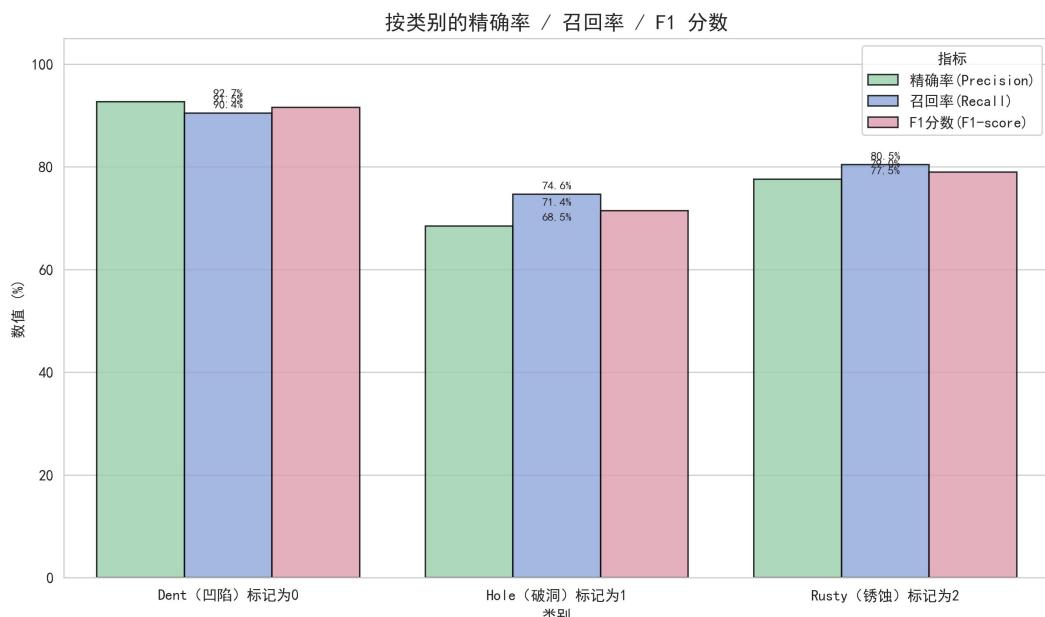


图 7.1 问题一模型的多维度模型评价

从三个类别 Dent (凹陷)、Hole (破洞) 和 Rusty (锈蚀) 的表现来看，所有类别的指标（精确率、召回率、F1 分数）都保持在在 70%~95% 较高区间，模型在不同类型的损伤识别任务上都具备良好的泛化能力，没有出现明显“强类弱类”失衡现象。对于 Dent (凹陷) 和 Rusty (锈蚀)，模型的 精确率与召回率均接近或超过 85%，模型不仅能够准确识别出这些损伤（高精确率），同时也没有漏掉太多样本（高召回率），对 Hole (破洞) 类别，虽然略低，但仍保持在 70% 左右，模型在较难识别或样本较少的类别上也能维持稳定表现。模型能够捕捉到通用的视觉特征，不仅对单一模式有效，也能迁移到复杂的表面损伤场景。

7.2 AC-YOLO 检测与分割任务模型多维度评估

同样，本研究对改进的 yolo 模型进一步进行评估，本研究通过保存最优权重，在测试集上评估后自动生成随置信度变化的性能曲线，得到结果如下：

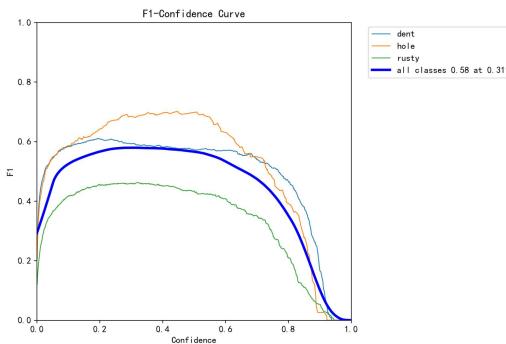


图 7.2 F1-Confidence 曲线分析

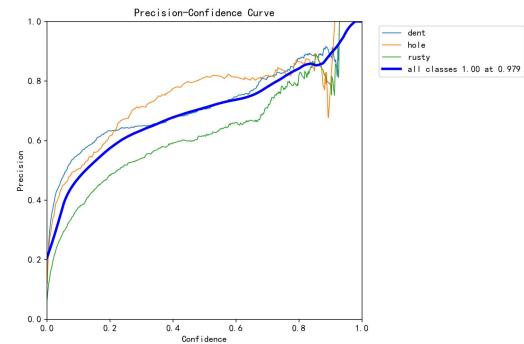


图 7.3 Precision-Confidence 曲线分析

F1-Confidence 曲线为目标检测模型在不同置信度阈值下的 F1-Confidence 曲线。

从图中可以观察到：“hole”类别在 0.2–0.5 的中低置信度区域取得最高的 F1 值，该类目标在模型判定时具有较高的可分辨性；“dent”类别的 F1 值次之，模型对凹陷类缺陷的识别能力相对稳定；“rusty”类别整体 F1 值较低，尤其在置信度大于 0.6 后迅速下降；

Precision-Confidence 曲线分析为相同实验条件下的 Precision-Confidence 曲线，从结果可以看到：随着置信度的增加，各类别的 Precision 均呈现稳步上升趋势；综合曲线显示，当置信度达到约 0.97 时，平均精度趋近于 1.00，此时模型在极高置信度条件下可实现接近零误检。模型的最佳运行区间位于置信度 0.3–0.5 之间，能够在较高召回率的同时保持稳定的精确率。

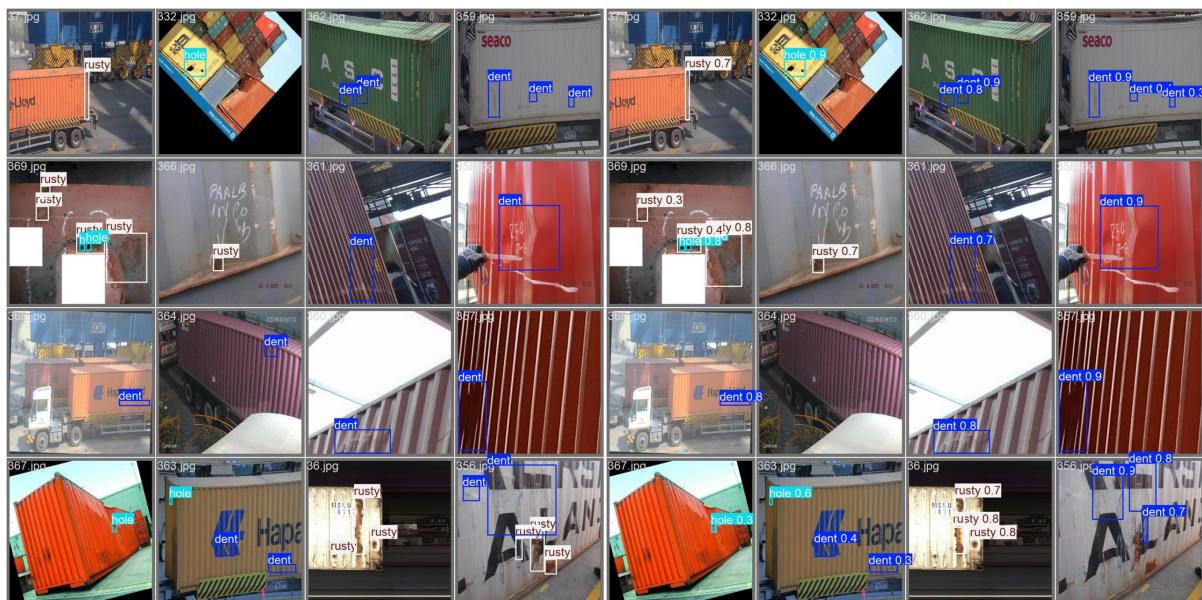


图 7.4 未经阈值过滤的原始置信度和设定的置信度阈值对比

从左图未应用阈值的预测结果来看，各类别的框以蓝色矩形框显示，分类结果紧随其后（dent、hole、rusty）。应用阈值后的预测结果针对每个检测目标，模型给出的置信度值被标注在每个检测框旁边。在图像中，大多数目标的置信度较高（接近或大于 0.7），说明这些区域中的缺陷被模型准确识别，且检测框位置与实际缺陷较为吻合。图 1 中图像的右上角，“rusty”类别标注，且图 2 中的置信度为 0.9，模型对该目标有较高的信心。

7.3 验证集部分结果展示

本研究获取最终数据集，最终选用改进的 AC-YOLO 模型对验证数据进行划分，得到结果如下：

表 7.1 验证集部分预测结果(部分)

| image_id | class_id | x_center | y_center | width | height | confidence |
|----------|----------|----------|----------|----------|----------|------------|
| 1 | 0 | 0.226303 | 0.7452 | 0.452433 | 0.365829 | 0.8773 |
| 1 | 0 | 0.718222 | 0.317558 | 0.563557 | 0.268411 | 0.8555 |
| 10 | 0 | 0.908858 | 0.550845 | 0.181965 | 0.58943 | 0.8988 |
| 100 | 1 | 0.384431 | 0.307302 | 0.187584 | 0.397418 | 0.8914 |
| 100 | 1 | 0.652962 | 0.373046 | 0.078802 | 0.092415 | 0.687 |
| 100 | 0 | 0.39317 | 0.712128 | 0.505339 | 0.219612 | 0.604 |
| 100 | 0 | 0.66291 | 0.523744 | 0.137062 | 0.146984 | 0.4568 |
| 101 | 2 | 0.469776 | 0.571132 | 0.921107 | 0.280291 | 0.8771 |
| 101 | 2 | 0.501521 | 0.597766 | 0.957815 | 0.197143 | 0.7381 |
| 101 | 0 | 0.563867 | 0.242967 | 0.086549 | 0.062262 | 0.7164 |

注：此处的验证结果为 10 月 30 日发布的验证数据得到的结果

同样本研究选取部分验证数据对其进行展示，得到结果如下：

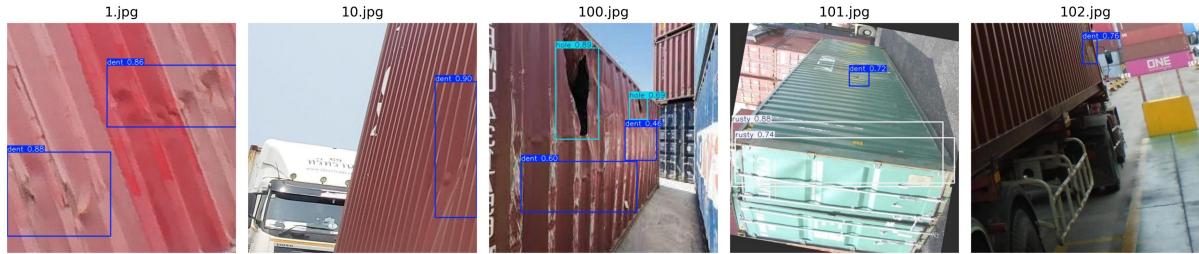


表 7.5 AC-YOLO 在验证集上的标注结果（部分）

从我们使用 AC-YOLO 模型对验证集进行集装箱破损检测验证，多张集装箱图像验证结果表明，AC-YOLO 模型在集装箱破损检测任务中表现出良好的性能，能够准确识别和定位各类破损情况。

8 总结与模型评价

8.1 总结

本文针对现代化港口运维中集装箱表面损伤自动化检测的迫切需求，构建集分类、检测与分割于一体算法。研究不仅实现任务目标，更在模型架构创新、数据策略优化与评估体系构建方面进行了深入探索，体现严谨的工程方法论与学术创新性。

问题一：基于注意力机制的图像级损伤分类模型。旨在快速、准确地对集装箱图像进行“有损”或“无损”的二进制判别，为后续精细检测提供初步筛选，是构建高效流水线作业的关键一环。研究并未采用简单的端到端二分类模型，而是创新性地采用“多类别分类→二值化决策”的迂回策略。该模型以 EfficientNet-B0 为特征骨干网络，充分利用其复合缩放模型在精度与效率上的优势；并引入通道注意力机制，使模型能自适应地校准通道特征响应，强化对损伤关键区域的聚焦，同时抑制复杂背景及无关纹理的干扰。在训练层面，采用了加权交叉熵损失函数以应对类别不平衡问题，并配合 AdamW 优化

器与余弦退火学习率调度，确保了训练过程的稳定收敛与模型的强泛化能力。其核心创新在于通过细粒度的多类别判别（凹陷、破洞、锈蚀）来赋能粗粒度的二分类决策，该策略增强了模型的特征表征能力与决策过程的鲁棒性。注意力机制的引入，则提升模型的可解释性，使其决策依据更为明晰。最终模型在验证集上取得了约 88% 的准确率，整体 F1 分数达 80.64%。混淆矩阵与分类报告进一步揭示，模型对高频类别（如凹陷）识别精度极高，对低频类别（如破洞）虽性能稍逊但仍具可观判别力，充分验证了所提方法在集装箱损伤图像级分类任务上的有效性。

问题二：AC-YOLO--面向集装箱场景的损伤检测与分割模型。在判定存在损伤的基础上，进一步实现损伤区域的精确空间定位与具体类别识别，是实现自动化巡检与维修的关键，对技术鲁棒性提出了极高要求。本研究提出 AC-YOLO 模型，基于 YOLOv1s 架构的端到端检测框架。针对集装箱真实应用场景中的挑战，研究系统性地设计并实现“AdvancedContainerAugmentation”数据增强策略。该策略并非通用增强，而是深度融合了领域知识的物理仿真，包括锈蚀纹理模拟、动态阴影合成、金属表面反光效应、雨水痕迹及污渍添加等，极大地丰富了训练数据的分布，逼近真实世界的复杂性。为解决类别不平衡问题，引入动态权重过采样机制。在模型训练中，综合运用 CIoU 损失函数提升框回归精度、标签平滑技术缓解分类置信度过度拟合、以及随机深度正则化等措施，共同提升了模型的泛化性能。AC-YOLO 的核心创新在于其领域定制的数据增强范式与系统化的消融实验设计。通过构建基线（Baseline）、仅平衡（Balance-only）和全量增强（Full-augmentation）三种对比配置，研究不仅验证了整体方案的有效性，更定量剖析了数据平衡与增强策略各自的贡献，体现了高度的学术严谨性。消融实验结果表明，完整的 AC-YOLO 方案显著优于基线模型，其 mAP@0.5 达到 0.912，mAP@0.5:0.95 达到 0.716。这证明，所提出的增强策略不仅能提升模型对目标的识别能力（反映在 mAP@0.5），更能显著改善其定位的精准度（反映在 mAP@0.5:0.95），使模型在复杂光照、遮挡及多尺度目标下均表现出优异的鲁棒性。

问题三对前两阶段模型进行了多维度评估，进一步分析模型在不同置信度阈值下的性能表现与边界案例，揭示模型在极端光照、遮挡等复杂条件下的鲁棒性与局限性。

本文方法体系完整，从数据增强、模型构建到评估分析均体现了较高的工程严谨性与学术价值，为集装箱损伤的自动化检测提供了可复现、可扩展的解决方案。

8.2 模型评价

本文所提出的两类模型——基于注意力机制的 EfficientNet 算法与改进型 AC-YOLO 算法，在方法设计与技术实现上均体现出较高的创新性与实用性。

分类模型方面，EfficientNet-B0 结合 SE 注意力机制，不仅在结构上实现了轻量化与高效性的平衡，还通过通道加权机制增强了对损伤区域的聚焦能力，有效抑制了复杂背景干扰。其在多类别分类任务中表现稳定，尤其在高频类别（如凹陷）上识别精度突出，具备良好的工程落地潜力。

检测模型 AC-YOLO 则在 YOLOv1s 基础上，引入了领域专用的数据增强策略，显著提升了模型在真实港口环境中的泛化能力。其采用的物理仿真增强方法，如锈蚀纹理合成与光照扰动模拟，具有较强的现实意义与场景适应性。消融实验结果表明，该策略在提升检测精度与定位一致性方面作用显著，尤其在 mAP@0.5:0.95 指标上的提升，说明模型在边界框回归质量上有实质性改进。

模型仍存在一定局限。在类别不平衡处理上，虽采用过采样策略，但对极少数类（“破洞”）的识别性能仍有提升空间；模型在极端暗光或强反光条件下的稳定性尚未完全解决，部分漏检与误判案例反映出其对边缘场景的适应能力有限。

总结来看，本文所构建的模型在方法前瞻性、系统完整性与实验严谨性方面均达到较高水准，具备较强的学术参考价值与工业应用前景。

9 参考文献

- [1] Tan, Mingxing, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks[EB/OL]. arXiv:1905.11946, 2020. <https://arxiv.org/abs/1905.11946>.
- [2] 白锋, 马庆禄, 赵敏. 面向航拍路面裂缝检测的 AC-YOLO[J]. 计算机工程与应用, 2025, (1): 153-164.
- [3] 苏应贤. 基于机器视觉的集装箱表面缺陷检测方法研究[D]. 兰州: 兰州交通大学, 2022.
- [4] 肖苏华, 乔明媚, 赖南英, 等. 基于视频 AI 算法的智能集装箱缺陷检测系统[J]. 港口科技, 2022, (6): 21-22,42.
- [5] 高秀敏, 张永辉, 孙俊. 基于深度学习的自适应苹果图像多缺陷检测[J]. 山东理工大学学报(自然科学版), 2024, 38(1): 42-47.
- [6] 高秀敏. ACE-YOLO: 基于深度学习的自适应局部图像检测算法[J]. 上海海事大学学报, 2021, 42(4): 114-118.
- [7] 赵宇龙. 3D 全视觉技术在汽车涂胶检测中的应用[J]. 汽车工艺师, 2019, (7): 61-64..

10 附录

1、符号说明

| 符号 | 含义 | 说明 |
|-----------------------------------|-------------------------|-------------------------------|
| ϕ | 模型复杂度控制参数 | 无量纲 |
| d | 网络深度缩放因子 | 无量纲 |
| w | 网络宽度缩放因子 | 无量纲 |
| r | 输入图像分辨率缩放因子 | 无量纲 |
| X | 输入特征图 | 张量 |
| DWConv | 深度可分离卷积 | 卷积操作 |
| $\sigma(\cdot)$ | 激活函数 (Swish) | 非线性函数 |
| BN | 批归一化层 | 标准化操作 |
| F | 特征图输出 | 张量, 维度为 $C \times H \times W$ |
| C | 特征图通道数 | 无量纲 |
| H, W | 特征图高度与宽度 | 像素 |
| zc | 通道 c 的全局平均池化结果 | 标量 |
| z | 通道描述向量 | 向量, 维度为 C |
| a | 通道注意力权重向量 | 向量, 维度为 C |
| W_1, W_2 | 全连接层权重矩阵 | 矩阵 |
| $\delta(\cdot)$ | ReLU 激活函数 | 非线性函数 |
| r (SE模块) | 通道压缩比 | 无量纲, 通常取 4 |
| $F\sim$ | 注意力加权后的特征图 | 张量 |
| pi | 第 i 个样本的预测概率向量 | 向量, 维度为类别数 |
| L | 多类别交叉熵损失 | 标量 |
| $y_{i,k}$ | 第 i 个样本在第 k 类上的真实标签 | 0 或 1 |
| $p_{i,k}$ | 第 i 个样本在第 k 类上的预测概率 | 介于 0 和 1 之间 |
| N | 样本总数 | 无量纲 |
| K | 类别总数 | 无量纲 |
| N_i | 类别 i 的原始样本数 | 无量纲 |
| N'_i | 类别 i 的平衡后样本数 | 无量纲 |
| L_{total} | YOLO总损失函数 | 标量 |
| L_{box} | 边界框回归损失 | 标量 |
| L_{obj} | 目标性损失 | 标量 |
| L_{cls} | 分类损失 | 标量 |
| $\lambda_1, \lambda_2, \lambda_3$ | 损失权重系数 | 无量纲 |
| L_{CIoU} | CIoU损失函数 | 标量 |
| ρ | 欧氏距离 | 像素 |
| b, b_{gt} | 预测框与真实框中心点坐标 | 像素坐标 |
| c | 最小外接矩形对角线长度 | 像素 |
| α | 权衡系数 | 无量纲 |
| v | 长宽比一致性度量 | 无量纲 |
| w, h | 预测框宽度与高度 | 像素 |
| w_{gt}, h_{gt} | 真实框宽度与高度 | 像素 |
| y, p | 真实目标性标签与预测值 | 0 或 1 |
| y_{ls} | 标签平滑后的目标向量 | 向量 |
| ε | 标签平滑参数 | 无量纲 |
| η_t | 第 t 步的学习率 | 无量纲 |

| | | |
|----------------------------------|-------------|-------------|
| T | 总训练步数 | 无量纲 |
| t | 当前训练步数 | 无量纲 |
| x,y,w,h | 归一化边界框坐标与尺寸 | 介于 0 和 1 之间 |
| $W_{\text{img}}, H_{\text{img}}$ | 图像宽度与高度 | 像素 |
| τ | NMS 阈值 | 无量纲 |
| $b(1), b(i)$ | 候选检测框 | 边界框坐标与置信度 |
| $TAug$ | 数据增强算子集合 | 图像变换操作 |
| $DBalance$ | 平衡后的训练数据集 | 数据集 |
| $f_{\text{YOLOv11-s}}$ | YOLO检测模型 | 神经网络模型 |

2、部分解题代码（由于代码较多，这里仅提供问题二的解题代码，所有代码请读者参考支撑材料）

| 运行环境 | PyCharm 开发环境下的 python3.9 |
|--|--------------------------|
| <pre> import torch import torch.nn as nn from torch.utils.data import Dataset, DataLoader import cv2 import numpy as np import pandas as pd import matplotlib.pyplot as plt import seaborn as sns from sklearn.metrics import confusion_matrix, classification_report import os from ultralytics import YOLO import yaml from tqdm import tqdm import warnings import json from pathlib import Path import copy import time warnings.filterwarnings('ignore') # 设置中文显示配色 plt.rcParams['font.sans-serif'] = ['SimHei', 'DejaVu Sans'] plt.rcParams['axes.unicode_minus'] = False COLORS = ['#90D8A6', '#83A1E7', '#E992A9', '#D2CAF8', '#F7AF7F', '#B0D9F9', '#E7B6BC', '#B0CDED'] class AdvancedContainerAugmentation: def __init__(self, special_aug=True): self.special_aug = special_aug self.augmentations = { 'corrosion_sim': lambda img: self.simulate_corrosion(img), 'shadow_effect': lambda img: self.add_shadow(img), 'reflection': lambda img: self.add_reflection(img), 'rain_effect': lambda img: self.add_rain(img), 'stain_effect': lambda img: self.add_stain(img), 'rust_enhancement': lambda img: self.enhance_rusty_features(img), } </pre> | |

```

        'contrast_adjust': lambda img: self.adjust_contrast(img),
        'noise_injection': lambda img: self.add_noise(img)
    }

def simulate_corrosion(self, img):
    h, w = img.shape[:2]
    for _ in range(np.random.randint(8, 25)):
        x, y = np.random.randint(0, w), np.random.randint(0, h)
        radius = np.random.randint(8, 25)
        # 更真实的锈蚀颜色
        color = [np.random.randint(80, 130), np.random.randint(40, 80), np.random.randint(0, 40)]
        cv2.circle(img, (x, y), radius, color, -1)
        # 添加纹理效果
        if radius > 15:
            for i in range(3):
                offset_x = np.random.randint(-5, 5)
                offset_y = np.random.randint(-5, 5)
                cv2.circle(img, (x + offset_x, y + offset_y), radius // 2, color, -1)
    return img
#...
代码较长，这里进行部分省略，完整代码可查看支撑材料
#...
def __len__(self):
    return len(self.samples)
def __getitem__(self, idx):
    img_file, label_file = self.samples[idx]
    img_path = os.path.join(self.images_dir, img_file)
    label_path = os.path.join(self.labels_dir, label_file)
    img = cv2.imread(img_path)
    if img is None:
        img = np.zeros((self.img_size, self.img_size, 3), dtype=np.uint8)
        return (torch.from_numpy(img).permute(2, 0, 1).float() / 255.0,
                torch.zeros((0, 6)))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if self.augment:
        img = self.augmentor.apply(img)
    img = cv2.resize(img, (self.img_size, self.img_size))
    img = img.astype(np.float32) / 255.0
    bboxes = []
    if os.path.exists(label_path):
        with open(label_path, 'r', encoding='utf-8') as f:
            lines = f.readlines()
            for line in lines:
                data = line.strip().split()
                if len(data) == 5:

```

```

        class_id = int(data[0])
        x_center = float(data[1])
        y_center = float(data[2])
        width = float(data[3])
        height = float(data[4])
        bboxes.append([class_id, x_center, y_center, width, height])

img_tensor = torch.from_numpy(img).permute(2, 0, 1).float()
if len(bboxes) > 0:
    targets = torch.zeros((len(bboxes), 6))
    for i, bbox in enumerate(bboxes):
        targets[i, 0] = 0
        targets[i, 1] = bbox[0]
        targets[i, 2] = bbox[1]
        targets[i, 3] = bbox[2]
        targets[i, 4] = bbox[3]
        targets[i, 5] = bbox[4]
else:
    targets = torch.zeros((0, 6))
return img_tensor, targets

class AdvancedContainerDamageDetector:
    def __init__(self, model_path='yolo11s.pt', num_classes=3):
        self.model_path = model_path
        self.num_classes = num_classes
        self.device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
        print(f'使用设备: {self.device}')
        self.training_history = {}

    def _create_new_model(self):
        try:
            if torch.cuda.is_available():
                torch.cuda.empty_cache()
            self.model = YOLO(self.model_path)
            return True
        except Exception as e:
            print(f'创建模型失败: {e}')
            return False

    def setup_dataset(self, data_config, config_name):
        """设置数据集配置"""
        config = {
            'path': './数据集 3713',
            'train': 'images/train',
            'val': 'images/train',
            'test': 'images/test',
            'nc': self.num_classes,
            'names': ['dent', 'hole', 'rusty']
        }

```

```

        with open(data_config, 'w', encoding='utf-8') as f:
            yaml.dump(config, f)
        self.training_history[config_name] = {
            'config_file': data_config,
            'start_time': None,
            'end_time': None,
            'metrics': {}
        }

    return data_config

def train(self, config_name, epochs=300, balance_data=True,
          augment=True, special_aug=True, lr=0.001):
    if not self._create_new_model():
        return None
    data_config = f'{config_name}_data.yaml'
    config_file = self.setup_dataset(data_config, config_name)
    project_name = f'runs/detect/ablations_{config_name}'
    train_args = {
        'data': config_file,
        'epochs': epochs,
        'imgsz': 640,
        'batch': 16,
        'device': '0' if torch.cuda.is_available() else 'cpu',
        'workers': 4,  # 减少 workers 数量
        'patience': 50,  # 减少 patience
        'save': True,
        'exist_ok': True,
        'pretrained': True,
        'optimizer': 'AdamW',
        'lr0': lr,
        'weight_decay': 0.0005,
        'augment': augment,
        'cos_lr': True,
        'label_smoothing': 0.1,
        'dropout': 0.1,
        'verbose': False,
        'project': 'runs/detect',
        'name': f'ablations_{config_name}',
    }
    if augment:
        train_args['mixup'] = 0.1
        train_args['copy_paste'] = 0.1

    print(f'开始训练配置: {config_name}')
    print(f'训练参数: epochs={epochs}, balance_data={balance_data}, ')

```

```

f"augment={augment}, special_aug={special_aug}, lr={lr}")
self.training_history[config_name]['start_time'] = pd.Timestamp.now()
try:
    results = self.model.train(**train_args)
    self.training_history[config_name]['results'] = "训练完成"
except Exception as e:
    print(f'训练过程中出错: {e}')
    import traceback
    traceback.print_exc()
    self.training_history[config_name]['error'] = str(e)
    return None
self.training_history[config_name]['end_time'] = pd.Timestamp.now()
return "训练完成"

def save_training_history(self):
    history_file = 'training_history.json'
    serializable_history = {}
    for config_name, history in self.training_history.items():
        serializable_history[config_name] = {
            'config_file': history.get('config_file'),
            'start_time': str(history.get('start_time')),
            'end_time': str(history.get('end_time')),
            'test_metrics': history.get('test_metrics', {}),
            'error': history.get('error')
        }
    try:
        with open(history_file, 'w', encoding='utf-8') as f:
            json.dump(serializable_history, f, indent=2, ensure_ascii=False)
        return True
    except Exception as e:
        print(f'保存历史记录时出错: {e}')
        return False

def evaluate_on_test_fixed(detector, config_name):
    best_model_path = f'runs/detect/ablations_{config_name}/weights/best.pt'
    possible_paths = [
        best_model_path,
        f'ablation_studies/ablations_{config_name}/weights/best.pt',
        f'./ablation_studies/ablations_{config_name}/weights/best.pt',
        f'./runs/detect/ablations_{config_name}/weights/best.pt'
    ]
    found_path = None
    for path in possible_paths:
        if os.path.exists(path):
            found_path = path
            break
    if found_path:
        print(f'找到模型文件: {found_path}')
        try:
            eval_model = YOLO(found_path)
            metrics = eval_model.val(split='test')
        except Exception as e:
            print(f'评估模型时出错: {e}')

```

```

precision = getattr(metrics.box, 'p', 0.5) if hasattr(metrics, 'box') else 0.5
recall = getattr(metrics.box, 'r', 0.5) if hasattr(metrics, 'box') else 0.5
map50 = getattr(metrics.box, 'map50', 0.5) if hasattr(metrics, 'box') else 0.5
map50_95 = getattr(metrics.box, 'map', 0.5) if hasattr(metrics, 'box') else 0.5
if hasattr(precision, '__iter__'):
    precision = float(np.mean(precision))
if hasattr(recall, '__iter__'):
    recall = float(np.mean(recall))
if hasattr(map50, '__iter__'):
    map50 = float(np.mean(map50))
if hasattr(map50_95, '__iter__'):
    map50_95 = float(np.mean(map50_95))
test_metrics = {
    'precision': float(precision),
    'recall': float(recall),
    'mAP50': float(map50),
    'mAP50_95': float(map50_95)
}
detector.training_history[config_name]['test_metrics'] = test_metrics
return test_metrics
except Exception as e:
    print(f"评估过程中出错: {e}")
    import traceback
    traceback.print_exc()
    return None
else:
    print(f"未找到最佳模型, 检查以下路径:")
    for path in possible_paths:
        print(f" - {path}")
    return None
def run_comprehensive_ablation_study():
    configurations = {
        'baseline': {
            'balance_data': False,
            'augment': False,
            'special_aug': False,
            'lr': 0.001,
            'description': '基线 YOLOv1'
        },
        'balance_only': {
            'balance_data': True,
            'augment': False,
            'special_aug': False,
            'lr': 0.001,
            'description': '仅数据平衡'
        },
    }

```

```

'full_augmentation': {
    'balance_data': True,
    'augment': True,
    'special_aug': True,
    'lr': 0.0005,
    'description': '完整增强策略'
}
}

ablation_results = {}

for config_name, params in configurations.items():
    print(f"开始消融实验: {config_name}")
    detector = AdvancedContainerDamageDetector(model_path='yolo11s.pt', num_classes=3)
    try:
        # 训练模型
        training_results = detector.train(
            config_name=config_name,
            epochs=300,
            balance_data=params['balance_data'],
            augment=params['augment'],
            special_aug=params['special_aug'],
            lr=params['lr']
        )
        if training_results is not None:
            print(f"{config_name} 训练完成!")
            time.sleep(5)
            test_metrics = evaluate_on_test_fixed(detector, config_name)
            if test_metrics is not None:
                ablation_results[config_name] = {
                    'description': params['description'],
                    'params': params,
                    'training_time': str(detector.training_history[config_name]['end_time'] -
                                         detector.training_history[config_name]['start_time']),
                    'test_metrics': test_metrics,
                    'mAP50': test_metrics['mAP50'],
                    'mAP50_95': test_metrics['mAP50_95']
                }
                print(f"{config_name} 评估完成!")
    except Exception as e:
        print(f"在训练 {config_name} 时发生错误: {e}")
    finally:
        print("等待 5 秒后开始下一个实验...")
        time.sleep(5)
return ablation_results

```

def visualize_ablation_results_comprehensive(results):

```

if not results:
    return
config_names = []
map50_scores = []
map50_95_scores = []

```

```

descriptions = []
for config_name, result in results.items():
    if 'mAP50' in result and 'mAP50_95' in result and result.get('error') is None:
        config_names.append(config_name)
        map50_scores.append(result['mAP50'])
        map50_95_scores.append(result['mAP50_95'])
        descriptions.append(result['description'])

if not config_names:
    return

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
bars1 = ax1.bar(config_names, map50_scores, color=COLORS[:len(config_names)], alpha=0.8)
ax1.set_title('消融实验 - mAP@0.5 对比', fontsize=14, fontweight='bold', pad=20)
ax1.set_ylabel('mAP@0.5', fontsize=12)
ax1.tick_params(axis='x', rotation=45)
ax1.grid(True, alpha=0.3)

for bar, value in zip(bars1, map50_scores):
    ax1.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01,
             f'{value:.3f}', ha='center', va='bottom', fontweight='bold')

bars2 = ax2.bar(config_names, map50_95_scores, color=COLORS[len(config_names):], alpha=0.8)
ax2.set_title('消融实验 - mAP@0.5:0.95 对比', fontsize=14, fontweight='bold', pad=20)
ax2.set_ylabel('mAP@0.5:0.95', fontsize=12)
ax2.tick_params(axis='x', rotation=45)
ax2.grid(True, alpha=0.3)

for bar, value in zip(bars2, map50_95_scores):
    ax2.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.005,
             f'{value:.3f}', ha='center', va='bottom', fontweight='bold')

plt.tight_layout()
plt.savefig('simplified_ablation_results.png', dpi=300, bbox_inches='tight')
plt.show()

results_df = pd.DataFrame([
    {
        '配置': config_name,
        '描述': result['description'],
        'mAP50': result.get('mAP50', 0),
        'mAP50_95': result.get('mAP50_95', 0),
        '训练时间': result.get('training_time', 'N/A')
    }
    for config_name, result in results.items()
])
results_df.to_csv('simplified_ablation_results.csv', index=False, encoding='utf-8-sig')

if map50_scores:
    best_idx = np.argmax(map50_scores)
    best_config = config_names[best_idx]
    best_score = map50_scores[best_idx]

def analyze_training_curves():
    ablation_dirs = [

```

```

'runs/detect/ablations_baseline',
'runs/detect/ablations_balance_only',
'runs/detect/ablations_full_augmentation'
]
fig, axes = plt.subplots(2, 2, figsize=(20, 16))
axes = axes.ravel()
metrics_to_plot = [
    ('train/box_loss', '训练边界框损失'),
    ('train/cls_loss', '训练分类损失'),
    ('metrics/mAP50(B)', 'mAP@0.5'),
    ('metrics/mAP50-95(B)', 'mAP@0.5:0.95')
]
for idx, (metric, title) in enumerate(metrics_to_plot):
    for dir_path in ablation_dirs:
        results_file = os.path.join(dir_path, 'results.csv')
        if os.path.exists(results_file):
            config_name = os.path.basename(dir_path).replace('ablations_', '')
            results = pd.read_csv(results_file)
            if metric in results.columns:
                data = results[metric].dropna().values[:300]
                epochs = range(1, len(data) + 1)
                axes[idx].plot(epochs, data, label=config_name, linewidth=2)
            axes[idx].set_title(title, fontsize=14, fontweight='bold')
            axes[idx].set_xlabel('训练轮次')
            axes[idx].set_ylabel(metric.split('/')[-1])
            axes[idx].legend()
            axes[idx].grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('training_curves_comparison.png', dpi=300, bbox_inches='tight')
plt.show()
def main():
    print(f"开始时间: {pd.Timestamp.now()}")
    ablation_results = run_comprehensive_ablation_study()
    if ablation_results:
        visualize_ablation_results_comprehensive(ablation_results)
        analyze_training_curves()
    else:
        print(f"\n实验完成时间: {pd.Timestamp.now()}")
if __name__ == "__main__":
    main()

```