工业安全装备实时检测系统

主要内容：本次项目基于 YOLOv5 实现工作服目标检测，利用深度神经网络提取并融合多尺度特征，完成目标的定位与分类。特征工程重点依靠自动一个框调整（AutoAnchor），提升小目标检测能力。采用 SGD 优化器结合动态学习率调度，确保训练稳定。训练过程中通过 TensorBoard 和 Matplotlib 进行损失和准确率的可视化监控，辅助模型调优和性能分析，实现高效安全装备的实时和识别。

目 录

1 需求分析

1.1 需求背景

在近现代工业生产环境中，尤其是建筑施工、制造车间、电力检修等高风险作业场景中，佩戴合规的劳动防护用品（如安全帽、安全背心、手套、口罩等）对于保障工人安全至关重要。随着安全管理数字化转型的加快，传统依靠人眼巡检、手工记录的监督方式逐渐暴露出效率低、误判高、成本大等弊端，越来越多的企业亟需引入基于人工智能的视频图像识别系统，以实现自动化、实时化、智能化的劳动防护检查。

因此，构建一个基于深度学习的“工作服识别”系统，旨在通过分析工作场景中的图像，自动检测人员是否正确佩戴安全装备，是当前智能安监系统的重要研究方向。本课题拟采用目标检测技术，借助 YOLO（You Only Look Once）系列算法，自动识别图像中的工人以及其所穿戴的各类防护用品，如安全帽、口罩、安全背心等，同时识别违规行为（例如未佩戴安全帽、未穿安全背心等），从而为现场安全管理提供有力技术支持。

该系统在应用层面具有广泛前景：可接入施工现场摄像头系统，实时检测并警告未佩戴安全装备的人员；亦可用于事后视频分析，自动生成违规行为统计报表；还可配合边缘计算实现终端部署，极大提升安防响应速度。本课题不仅关注算法性能和准确率，更强调实际部署中的模型轻量化、实时性与鲁棒性。

在研究技术路径方面，本项目将结合已有成熟的目标检测框架 YOLOv5，对公开数据集进行训练和评估。

1.2 数据分析需求

数据是是深度学习的基石，优质且多样化的训练数据对模型性能至关重要。要构建一个可靠的深度学习目标检测系统，数据集的质量、结构和分布决定了模型的上限。此次项目使用的是由 Roboflow 提供的“Construction Site Safety”数据集，其已按照 YOLO 格式组织，涵盖 25 类安全相关目标，包括“安全帽”、“手套”、“口罩”、“挖掘机”、“未佩戴安全帽”等常见安全符号及违规行为标签。

目标识别，目标检测，目标分割是计算机视觉中重要的散步，目标识别主要

指对图像中单一目标的分类，即判断某个图像或者图像区域属于哪个类别。目标检测则不仅识别目标类别，还要定位目标在图像中的具体位置和大小，在本实验中尤为重要。目标分割一步细化，把目标从背景和其它目标中以像素级分割出来，精准描绘目标形状和边界。

1.2.1 图像目标识别

从图像尺寸与网络结构来看，图像普遍具备较高分辨率（300x300 以上），色彩丰富，场景真实。这样的图像在特征维度上至少包含 90,000 个像素值，对于浅层网络（如仅 3~4 层卷积）而言，特征提取能力有限，无法有效捕捉图像中的复杂语义信息。因此，需要构建较深的卷积神经网络（如 YOLOv5 的 backbone 网络通常包含 CSPDarkNet-53 或 P3-P6 路径），以获得更具判别力的多尺度特征。

从色彩敏感性来看，数据中的目标颜色变化大，如不同颜色的安全帽（红/黄/蓝）、背心（反光绿/橙）、车辆等，提示模型应具备对色彩变化不敏感的能力。因此在数据增强策略中应引入颜色抖动（color jitter）、直方图均衡等方式，以提升模型对光照条件与色彩波动的鲁棒性。

从图像中目标的形状来看，不同类别的目标边缘和结构差异大，如“安全锥”是规则几何图形，而“人员”、“机械”结构复杂，容易因遮挡和角度造成误检，而且人员姿态各异，可能出现场景中的旋转、翻转、遮挡等情况。数据增强中引入随机水平翻转、旋转，使模型更好地适应各种变形，增强泛化能力。

从数据的规模及类别均衡性来看，数据量是会直接影响模型训练效果，少量数据容易过拟合。正则化、数据增强等技术可缓解此问题。样本类别分布应尽量均衡，不均衡会导致某些类识别准确率低，需采用过采样、损失加权或采样策略修正。本次使用的 Roboflow Construction Site Safety 数据集包含约 700 多张验证图片与 3000 多张训练图片，类别覆盖广泛且相对均衡。均衡的样本分布有利于防止模型偏向大类，保证各个类别均能得到有效识别。

1.2.2 图像目标检测

基于本次实验所用的安全生产相关工作服及防护装备的数据集，目标检测任务不仅涵盖了对每个工人及其安全装备的识别与定位，更需要面对目标稠密、背景复杂、目标尺寸多样等现实场景的挑战。

目标的稠密性是工作服及防护装备的数据集数据中最典型的特征之一。建筑

工地或工业现场照片中，往往会同时出现多名工作人员，同时他们身着各种不同的防护装备（如安全帽、安全手套、安全背心等）。这些目标之间可能彼此重叠或遮挡，导致目标检测模型需要具备强大的区域分辨和识别能力。稠密目标会使得检测难度增加，因此模型需要采用多尺度特征提取机制，YOLOv5 通过其深层卷积网络和基于 Feature Pyramid Networks（FPN）结构的特征融合，有效应对不同尺度和密度的目标。

1.2.3 图像目标分割

图像分割任务相较于目标检测在精细度和技术要求上有着更高的标准。它不仅要求模型能够检测出图像中的目标位置，还需准确地划分出目标的像素级轮廓和形状边界。这种细粒度的识别对于工作服的识别尤为关键，因为它可以清晰地反映穿着者的安全防护具体情况，比如是否佩戴了完整且规范的安全装备。像素级的分割不仅提升了安全监测的准确度，还能够为现场的安全行为分析和作业过程的详细记录提供强有力的支持，极大地促进了安全管理的智能化发展。

与目标检测相比，图像分割对模型架构中细粒度特征提取的依赖更强，例如 UNet、Mask R-CNN 等网络结构便是因其具备多层次特征融合能力，而广泛应用于图像分割任务中。它们通过编码器和解码器的对称结构，能够有效恢复目标的空间细节和形状信息，实现更为精准的分割结果。因此，对于工作服识别这类需求对目标形状细节要求较高的任务，分割模型体现在性能上的优势尤为显著。

在数据增强方面，为了提升分割模型对目标边界的感知能力，建议不仅采用常规的裁剪、旋转、色彩变换，还引入边缘保留和平滑等图像处理手段。这样可以增强模型对边界特征的敏感度，减少边缘模糊和误分割现象。此外，考虑到分割模型对数据规模的敏感性较高，数据量较少时容易导致过拟合，必须通过增加数据扩充、多样化的增强手段以及正则化技艺，来提升模型的泛化性能和稳定性。

1.3 任务需求分析

为了明确采用图像分类、目标检测还是图像分割技术，需要首先分析任务目标与图像特征。该项目的核心目标是检测图像中是否存在某类对象（如“是否佩戴安全帽”、“是否穿戴背心”等），不仅要判断图像是否包含某类信息（图片级），更需要明确指出每个目标在图像中的具体位置（像素坐标），甚至区分目标之间的类别（如“未佩戴安全帽”与“已佩戴安全帽”是两个类别）。因此，

该任务明确属于图像目标检测任务。

具体来说，图像中往往存在多个对象，例如同时有多名工人、机械设备、安全标识等，这就需要检测系统具备多目标识别能力。同时，目标之间可能有遮挡、远近变化等干扰因素，如果采用图像分类方式（例如输入整张图像，输出“违规/不违规”），会丧失对目标位置信息的识别能力，导致实用性下降。

而图像分割虽然能提供像素级别的目标轮廓，但其训练成本高、对标签精度要求高、推理速度慢，并不适合部署在实时监控系统中。因此，在准确性与效率之间取平衡的目标检测成为本项目的首选方法。

基于 YOLO（You Only Look Once）系列的目标检测网络是一种一次性处理整张图像并预测多个边界框与类别的轻量化检测方法，其具备如下优点：实时性强、适用于工业场景、精度和召回率兼顾、网络结构模块化、易于改造等优点。因此，本项目将采用 YOLOv5 作为主干检测网络，通过在训练阶段学习目标类别与位置的联合分布，在测试阶段对输入图像实现快速、准确的目标检测与分类。

1.4 技术需求分析

全连接网络和卷积网络构建目标检测模型是本次课设的核心技术路线，结合 YOLO 网络结构优化，可实现高效、精准的工作服识别系统。

1.4.1 全连接神经网络（Fully Connected Layer）

全连接层是传统神经网络结构中的核心组成部分，用于对卷积或池化后的特征图进行展平并输出分类结果。在本项目中，全连接层主要用于 YOLO 检测头部，即将最后的特征向量映射为目标边界框位置和类别的预测值。它的作用是完成高维特征与最终检测结果的映射，起到分类与回归的作用。

1.4.2 卷积神经网络（Convolutional Neural Network, CNN）

CNN 是图像处理的核心技术之一，通过局部连接、权重共享等方式提取图像中的空间特征。YOLO 系列模型使用的骨干网络（如 CSPDarknet）和特征融合模块（如 PANet）都是卷积结构。它能有效捕捉图像的边缘、角点、纹理等特征，对于检测目标形状具有天然优势，是本项目中最重要的网络结构。

1.4.3 循环神经网络（Recurrent Neural Network, RNN）

RNN 主要用于处理序列数据，擅长建模时间序列或文本信息。本项目聚焦于图像目标检测，不直接涉及时间维度，因此 RNN 不是主要结构。但在未来若将

图像检测扩展为视频帧连续识别任务，RNN 或 LSTM 可以用于建模时间连续性，例如识别违规行为持续时间或运动趋势。

2 概要设计

2.1 数据准备

深度学习项目的第一步始终是高质量数据的准备。数据准备阶段是确保模型能够有效学习和泛化的关键基础。数据集的存放格式直接关系到后续的数据读取和处理效率，进而影响模型训练的质量和速度。本项目使用的数据集为来自公开平台 Roboflow 的“Construction Site Safety Dataset”，已标注并按照 YOLO 格式进行整理，涵盖多个工地场景中与劳动保护相关的目标，如“佩戴安全帽的工人”、“未佩戴安全帽的工人”、“佩戴背心的工人”、“机械设备”等共 25 个类别，具备高度的真实性和代表性。

2.1.1 数据存放格式

本数据集采用的是 YOLO 格式的目标检测数据结构，将图像(images/)与对应标注文件(labels/)分别存放于统一目录下，每个图像文件对应一个标注文件，且文件名保持一致（如 img0001.jpg 对应 img0001.txt），高效且易于自动解析。此外，数据集涉及 25 个类别，涵盖常见的安全帽、手套、安全背心等装备，符合工业安全应用需求标签以 YOLO 格式提供，且这些数值都是相对于图像宽高的归一化坐标，满足 YOLO 要求。文件中每一行代表一个目标的一个标注。数据集的目录结构如下图 2-1 所示：

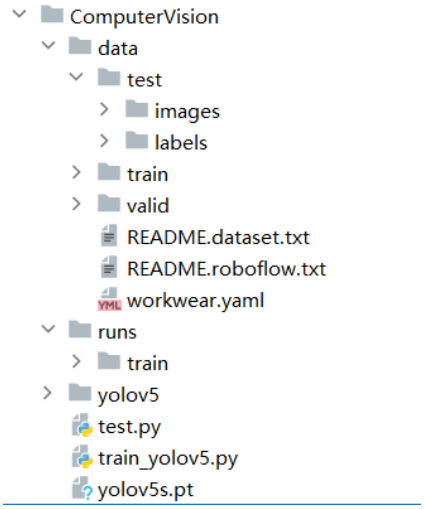

图 2-1 数据集的结构

2.1.2 数据规模

实验所用的数据集规模与内容具备一定的代表性和复杂性，为深度学习模型训练和评估提供了充足的数据基础。具体而言，训练集包含约 4000 张多样化的工地安全相关图像，涵盖了多种工作服和安全装备场景；验证集和测试集各自约 800 张图像，确保模型在非训练数据上的泛化能力和性能可被准确评估。数据集中共设定了 25 个类别，详尽定义在 workwear.yaml 配置文件中，涵盖了从“Person”、“Hardhat”等高频职业安全防护装备到“Safety cone”、“Helmet violation”等相对少见但异常重要的安全标志物。

需要特别指出的是，数据集中的类别分布存在较显著的不均衡现象，诸如“Person”、“Hardhat”等类别在图像中出现频率较高，远远超过某些罕见类别如“Safety cone”或“Helmet violation”。这种不平衡会对模型训练带来挑战，导致模型在弱势类别的识别准确率下降，进而影响整体的安全监控效果。因此，在本实验的训练策略中，采用了类别重加权技术，即通过调整损失函数中的类别权重来放大稀缺类别的重要性，平衡模型学习的投入。除此之外，数据增强策略也是极为关键的补充措施，通过随机翻转、缩放、色彩抖动以及 Mosaic 等方法有效扩展了少数类样本的表现力。

2.2 数据预处理

图像数据的质量对模型效果有决定性影响，因此数据预处理是模型训练前的重要步骤，包含读取、增强、归一化等操作。

2.2.1 数据读取

本项目使用 PyTorch 与 Ultralytics 的 YOLOv5 框架，利用其封装良好的 Dataset 类和 Dataloader 工具自动读取图像与标签。只需在训练脚本中指定 data.yaml 文件路径，即可自动加载训练、验证、测试数据。通过数据配置的 YAML 文件，即可以读取数据。workwear.yaml 如图 2-2 所示：

```yaml
train: ../data/train/images
val: ../data/valid/images
nc: 25
names: ['Excavator', 'Gloves', 'Hardhat', 'Ladder',
        'Mask', 'NO-Hardhat', 'NO-Mask', 'NO-Safety Vest',
        'Person', 'SUV', 'Safety Cone', 'Safety Vest',
        'bus', 'dump truck', 'fire hydrant', 'machinery',
        'mini-van', 'sedan', 'semi', 'trailer', 'truck',
        'truck and trailer', 'van', 'vehicle', 'wheel loader']
```

图 2-2 workwear.yaml 文件

2.2.2 数据增强

为了提升模型泛化能力、缓解过拟合，项目中采用以下数据增强策略（由 YOLO 框架自动处理）：随机翻转（horizontal/vertical flip）：模拟不同角度，颜色抖动（color jitter）：模拟不同光照，仿射变换（affine transform）：改变目标姿态，Mosaic：拼接 4 张图像生成一张，提升小目标检测能力，这些方法对目标形状、颜色、位置等多维度进行扰动，使得模型学会从更多样的输入中提取鲁棒特征。

2.2.3 数据清洗

数据清洗阶段需排除不合法标签文件，如缺少图像、坐标超界、类别超限等。本项目使用 YOLO 框架训练时，框架会自动输出警告信息，提示格式错误图像，开发者应检查是否存在：图像存在但标签缺失，标签类别超过设定范围，边界框尺寸为 0。

2.2.4 数据归一化

YOLO 模型要求标签中坐标是相对值（归一化），已由 Roboflow 处理完毕，形式通常为<class_id> <x_center> <y_center> <width> <height>，图像像素值归一化也由 YOLO 框架自动处理，默认将图像缩放至 640x640，像素值范围归一化至 0~1，确保网络训练收敛稳定。

2.3 方法构建

2.3.1 模型选择

本实验选用 YOLOv5 作为目标检测的核心模型，该模型因其卓越的速度和准

确率被广泛应用于多种实时目标检测任务中。YOLOv5 是 YOLO 系列模型的最新迭代，优化了模型体积和推理效率，使其特别适合于应用于工业安全监测场景，如工作服识别和防护装备检测。相比于传统两阶段检测器（如 Faster R-CNN），YOLOv5 作为单阶段检测器能以更快的速度执行目标定位和分类，实时性高，这是工地安全监控、入场审核等场景的核心需求。此外，YOLOv5 在保持高准确率的同时，能够支持多类别、多目标密集检测，有利于应对工地人员及其多样化安全装备的复杂情境

在本实验的具体应用环境中，人员密集且携带多种颜色和形态各异的安全装备，使得需要一个既能快速响应又具备强大表示能力的模型。YOLOv5 的架构设计灵活，能适配小型便携式设备和高性能服务器，同时支持多种硬件加速，满足不同部署需求。

2.3.2 模型结构

本系统采用 YOLOv5s（small）模型预训练权重作为初始化，采用微调方式针对本工作服识别的数据集进行训练。YOLOv5s 是该系列中参数量较小但性能均衡的模型，便于快速训练和迭代，适合硬件资源有限的环境。其整体结构包含深层卷积模块，结合 CSPDarkNet 结构用于提取强大且多尺度的图像特征，赋能后续的高效目标识别与定位。

模型的主干网络负责逐层提取目标的语义和空间描述，特征融合模块则借助类似 Feature Pyramid Networks（FPN）的机制整合高分辨率早期特征与高级抽象特征，加强对大、中、小目标的适应能力。此外，YOLO 检测头针对不同尺度的特征图实现多层次的预测，保证了对大小目标的均衡检测敏感性。基于该结构，微调训练让模型快速学习工作服及相关安全设备的特征差异，有效消除预训练模型领域差异带来的影响，同时保障训练效率

2.3.3 训练组件

训练过程中，YOLOv5 综合采用多任务损失函数，包括边界框定位损失（box loss）、目标置信度损失（objectness loss，简称 obj loss）、类别分类损失（cls loss）。这三者通过权重加权策略融合，确保网络在准确定位目标、评估目标是否存在及准确分类三方面均衡优化，提升整体性能。边界框损失多采用 CIoU（Complete-IoU）或 GIoU 等先进指标，增强对重叠框的距离估计能力。

在优化器的选择上，本实验结合了传统的随机梯度下降（SGD）和动量技术，

有效加快收敛速度，同时减少震荡。根据实验需求和硬件条件，也可替换为 Adam 优化器以便加速训练过程。学习率策略采用预热和余弦退火调度，使初期权重得到稳定更新，后期逐步收敛至最优解。此外，正则化手段以及数据增强策略有效避免过拟合，尤其针对类别分布不均和目标稠密的复杂数据场景发挥重要作用

2.3.4 详细步骤

本次的系统是基于模型 YOLOv5 进行的目标检测，YOLO（You Only Look Once）是高效的单阶段目标检测模型，能够实时准确地检测图片中的多个目标，尤其是在图像中定位工作服（即目标检测而不仅仅是分类）时，YOLO 是非常合适的深度学习框架。而我本次的数据集标签是以边界框标注（如.txt 文件或 json 格式），对应图片，能直接训练检测模型。训练后既能分类又能定位工作服位置，适合实际应用需求。

（1）项目配置，在项目实施初期，通过配置 SSH 密钥成功克隆 YOLOv5 仓库，确保版本的稳定和源码的完整性（如图 2-3 所示）。克隆之后，按照官方依赖清单执行安装，包括 PyTorch、torchvision、OpenCV、Matplotlib 等必要软件包，建立统一且完备的环境支持（见图 2-4）这一步保证了环境的兼容和模型实现的最优效率。随后，在实验代码中通过导入 YOLOv5 模块实现核心功能调用，合理组织训练和推理流程。

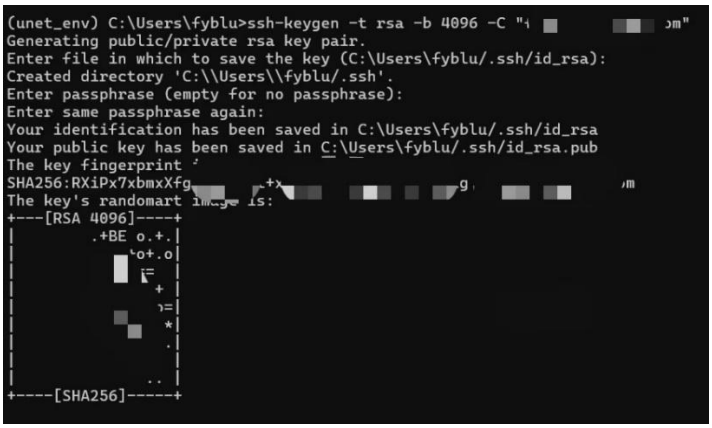

图 2-3 配置 SSH key

在克隆的 yolov5 文件夹中，执行 pip install -r requirements.txt 自动安装依赖，保证 PyTorch 与相关库版本匹配，并支持 GPU 加速。这为后续高速训练与推理奠定基础。

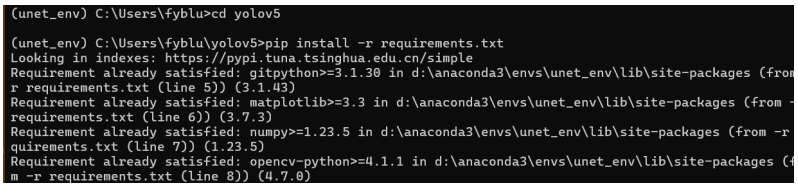

图 2-4 安装依赖文件

（2）实验的主要流程

模型构建与微调。通过调用 yolov5s.pt 预训练权重作为基准，利用 YOLOv5s 网络结构，其深层卷积核和多层特征融合模块，使得模型具备优秀的多尺度目标识别能力。本实验中通过修改配置文件调整类别数及参数，实现针对本数据集的定制微调。

训练流程控制。利用 YOLOv5 的 train.py 核心功能，执行前向传播、损失计算（包括边框回归、目标置信度和类别损失），后向传播及参数更新。通过配置训练超参数（批量大小、学习率、迭代周期等）和设备选择（支持 GPU 加速）来提升训练效率和模型性能。

模型评估与保存。训练过程定期在验证集上进行准确率、召回率和 mAP 等指标计算，确保模型稳定收敛。训练结束后保存最佳及最后模型权重，方便后续推理与部署。

（3）本系统的训练流程主要包括以下关键步骤：①数据加载：利用 PyTorch 提供的 DataLoader 结合自定义 Dataset 实现，自动读取和批处理图像及对应标签，支持并行加速和实时数据增强。②数据增强：包括图像尺寸调整、随机裁剪、旋转、镜像翻转以及色彩抖动等，有效增加训练数据的多样性和模型的适应性，缓解数据集偏差。③前向传播：输入图像经过模型主干和特征融合模块提取多尺度特征，目标检测头输出候选框、置信度及类别预测。④损失计算：结合真实标签，计算定位、置信度和分类的综合损失。⑤反向传播：基于损失梯度执行反向传播，传导误差至每层网络权重。⑥参数优化：优化器使用计算的梯度更新网络参数。⑦验证评估：每个 epoch 结束后，模型在验证集上进行推理，计算精准率（Precision）、召回率（Recall）、mAP 等指标，监控模型性能趋势。

3 开发工具和编程语言

3.1 开发工具

3.1.2 PyTorch

PyTorch 是当前广泛应用的深度学习框架，具备以下优势：动态图机制：相比 TensorFlow 的静态图，PyTorch 提供更灵活的建模方式，调试更方便，特别适合实验和研究。强大的社区支持：众多优秀项目如 YOLOv5/YOLOv8、Detectron2、MMDetection 等都基于 PyTorch 实现。GPU 加速：支持 CUDA，适配 NVIDIA 显卡，可大幅提高训练速度。在本项目中，PyTorch 是 YOLO 模型的底层支持框架，负责模型搭建、损失函数计算、梯度更新等核心计算任务。

3.1.2 PyCharm

Py Charm 支持智能代码补全、语法高亮、错误检测和代码重构等功能，极大地简化了深度学习模型的编写工作。它内置了集成终端和 Python 控制台，允许开发者在 IDE 内部执行命令和测试代码，无需频繁切换窗口，提升工作流的连贯性。此外，PyCharm 支持对项目依赖和虚拟环境的管理，方便切换和配置合适的 Python 解释器，确保深度学习项目环境的稳定和可重复性。

在调试方面，PyCharm 提供了强大的断点设置、变量监控和逐步执行等调试工具，支持多线程和多进程调试，极大方便了对复杂神经网络代码的错误排查和性能瓶颈分析。

3.2 编程语言

本项目统一采用 Python 3.8+ 作为主要编程语言。Python 是当前人工智能、数据科学及自动化处理的主流语言，原因如下：简洁易读：语法清晰，便于快速上手，适合模型开发、脚本编写等任务。丰富的深度学习生态：拥有 PyTorch、TensorFlow、OpenCV、Scikit-learn、matplotlib 等强大库支持。跨平台性强：Python 程序可在 Windows、Linux、macOS 等操作系统上无缝运行。

4 详细设计及运行结果

4.1 实验环境

本实验具备较强计算能力的本地环境中进行，操作系统为 Windows 11（，开

发平台为 PyCharm 2023.1 Professional Edition，编程语言为 Python 3.8，主要依赖的深度学习框架为 PyTorch 2.0，支持 CUDA 11.8。为了确保训练效率和准确率，本实验采用 NVIDIA GeForce RTX 3050 GPU，显存为 8GB，可有效支持大尺寸图像和较复杂网络的并行加速训练。实验中调用的 YOLOv5 模型为 s 版本，即 YOLOv5s，是 Ultralytics 官方提供的轻量级模型，适用于本实验中不复杂、类别数较少的场景。此外，相关依赖库如 torchvision、numpy、opencv-python、matplotlib、seaborn、tqdm、PyYAML 等均在训练环境中被正确安装并使用，保证了训练过程的可控性与稳定性。在数据方面，我们使用了上传的 data.zip 数据集，该数据集结构符合 YOLOv5 要求，图像保存在 images 目录中，标签则存放在 labels 目录下，并使用 YOLO 格式进行标注，即每行标注包括类别编号、归一化的目标框中心点坐标及宽高。训练及推理过程均通过 YOLOv5 的 train.py 与 detect.py 脚本完成，模型文件及运行日志会自动保存在 runs/train/exp 与 runs/detect/exp 等目录下。

本实验所使用的数据集主要采集实际工业生产相关现场及公开的安全服装工厂数据，涵盖了建筑工地、制造制造等高风险作业环境中的安全防护图像措施。数据标签采用方便了 LabelMe 和 Roboflow 这两大主要的标签图像工具，经过专业人员精心标注，导出格式为 YOLO 目标检测格式。此格式以简单的纯文本形式记录了每张目标的类别 ID 及其边界框信息，极大了深度学习模型的自动化与解析。

数据规模方面，训练集包含 521 张人工标注的工作场景图像，验证集包含 114 张质检过的图片，覆盖了 25 个不同类别的目标，如安全帽、手套、安全背心、车辆等。值得一提的是，类别分布均衡，有效防范了模型训练阶段的类别偏见，帮助模型训练在多类别识别任务中保持性能稳定性。数据中的目标多样性且尺寸增大增大，真实反映了工业现场的复杂环境，为模型的泛化能力提供了宝贵条件。

为了提高数据的利用率，实验数据在准备阶段还进行了成型的储存，使得图像与对应标签存在一致的关系目录体系，保证了训练流程的双向与数据加载的高效。这为后续的数据修复和模型训练奠定了坚实的基础。

4.2 整体流程

本实验严格遵循 YOLOv5 目标检测的整体流程，主要包括数据准备、数据增强、模型选择与配置、模型训练、评估与可视化推理等六大步骤。在数据准备阶

段，我们首先对上传的数据集进行解析与重构，确保图像与标注文件一一对应。在数据增强方面，YOLOv5 默认集成了多种增强策略，如随机裁剪、色彩扰动、旋转、翻转等操作，可有效提升模型的鲁棒性。模型选择方面，考虑到实验硬件资源和数据集规模，我们选择了 YOLOv5s 模型，其拥有较少的网络参数和计算量，适合快速训练和调试。训练过程中，模型自动调用 train.py 脚本，基于配置文件训练 100 轮，同时记录每轮 loss 和评估指标。评估阶段使用 val.py 脚本，对验证集进行精度、召回率、mAP 等评价，并可自动绘制 PR 曲线和混淆矩阵。最后，推理阶段通过 detect.py 脚本对测试集图像进行目标检测，并输出带预测框的图像以供人工核查。整体流程高度自动化、模块化，使得本实验在有限时间内能够高效完成目标检测任务并获取具有参考价值的性能指标。

4.3 基于 YOLOv5s 模型的训练

4.3.1 添加配置文件

为了对数据集中的 25 个类别进行训练，实验配置文件 workwear.yaml（内容如图 4-1 所示）进行了权限修改。此文件详细方便地指定了训练集和验证集的路径图像，并明确了类别数量和类别名称，确保模型能够准确识别并分类所有安全防护装备。路径设置为相对于 YOLOv5 代码目录的相对路径，环境一致性管理。此外，workwear.yaml 还包含了数据集的元信息及授权许可，保证了数据使用的合规性和可追溯性[

```yaml
train: ../data/train/images
val: ../data/valid/images
nc: 25
names: ['Excavator', 'Gloves', 'Hardhat', 'Ladder',
        'Mask', 'NO-Hardhat', 'NO-Mask', 'NO-Safety Vest',
        'Person', 'SUV', 'Safety Cone', 'Safety Vest',
        'bus', 'dump truck', 'fire hydrant', 'machinery',
        'mini-van', 'sedan', 'semi', 'trailer', 'truck',
        'truck and trailer', 'van', 'vehicle', 'wheel loader']
```

图 4-1 配置文件

4.3.2 训练过程

本实验以 YOLOv5s 作为预训练模型基础进行训练，充分利用其轻量化设计和卓越的检测速度，适合快速迭代和快速嵌入工业安全监控场景。YOLOv5s 不仅

显着缩短了模型训练时间，还借助预训练权重提升了图像特征提取的，使模型更易适应复杂的现场环境，提高检测准确率和鲁棒性用 YOLOv5s 预训练模型微调。训练参数配置科学合理：训练周期设定 30 个 Epoch，保证模型在有限数据集上获得充分的训练尺寸同时控制过风险；YOLOv5s 模型的训练是一个端到端的优化过程：首先加载轻量化的 yolov5s.yaml 网络结构，通过预训练权重 yolov5s.pt 初始化模型；采用 SGD 优化器结合余弦退火学习率策略，同时计算包含边界框回归（CIoU Loss）、目标置信度和分类损失的三部分加权损失；每轮训练后使用验证集评估 mAP 指标，自动保存最佳模型（best.pt）和最后模型（last.pt），整个过程通过混合精度训练（AMP）和多 GPU 并行（DDP）加速，最终在 COCO 等数据集上实现高精度实时检测，如图 4-2 所示。

```
D:\anaconda3\envs\unet_env\python.exe D:\Code\PycharmProjects\Practice\ComputerVision\train_yolov5.py
train: weights=yolov5s.pt, cfg=, data=D:\Code\PycharmProjects\Practice\ComputerVision\data\workwear.yaml, hyp=yolov5\data\
github: skipping check (offline), for updates see https://github.com/ultralytics/yolov5
YOLOv5  v7.0-420-g0c99ce80 Python-3.8.19 torch-2.3.1 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_b
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5  runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs\train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=25

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19         [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22         [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     80910  models.yolo.Detect                      [25, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 4
Model summary: 214 layers, 7087054 parameters, 7087054 gradients, 16.2 GFLOPs
```

图 4-2 训练过程

4.3.3 参数配置

本次实验采用了基于 YOLOv5s 预训练权重的微调训练策略，充分利用了预训练模型在大规模数据上学到的丰富特征表示，从而加快训练速度并提升精度。模型权重选择了官方发布的轻量级“s”版本（yolov5s.pt），该版本结构简洁，参数量小，适合资源有限环境和实时应用。训练过程中设置了 30 个训练周期（epochs），兼顾训练充分性与时间效率，适合实验初期及不断迭代优化。

批量大小 batch size）设为 4，这一数值基于实验所用硬件的计算和显存能力充分考虑。而优化器方面采用了基础而稳健的随机梯度下降（SGD）方法，初始学习率为 0.01，学习率设定遵循常规训练经验。该优化策略结合了动量和权重衰减，针对目标检测任务中的分类与定位损失进行了调优。SGD 能很好平衡训练收敛速度和稳定性，尤其在结合预训练模型微调时表现优异。训练过程均使用 GPU 进行加速运算，充分发挥 RTX 3050 的计算优势，实现高效训练和推理能力。

数据方面和模型架构上，数据方面本工程采集了 521 张带有准确标注的训练图像以及 114 张用于验证的图像，涵盖了 25 个不同类别的安全装备和工业场景元素，模型架构上，YOLOv5s 包含 214 层深度网络，参数量约为 708 万，推理计算需求约 16.2 GFLOPs。该网络结构设计紧凑，同时并未牺牲模型检测精度。

自动锚框优化（AutoAnchor）是 YOLOv5 训练流程中的重要步骤，结合当前数据集特点进行目标边界框的自适应调整。本数据集中共有 3943 个目标标注参加聚类计算，其中发现有 529 个极小目标（尺寸小于 3 像素），这类目标特别难以识别，且通常对模型提出更高的检测能力要求。为此，YOLOv5 采用 KMeans 聚类方法对锚框尺寸进行初始优化，随后利用遗传算法持续进化调整，使生成的九组锚框更贴合当前数据的分布特征。

```python
import os
from yolov5.train import run
def main():
    # 数据配置 YAML 文件，路径相对于 yolov5 目录
    data_yaml = os.path.abspath('data/workwear.yaml')
    run(
        imgsz=300,  # 输入的图片大小
        batch=4,  # 批量大小，根据你显卡调整
        epochs=1,  # 训练轮次
        data=data_yaml,  # 数据集配置文件路径
        weights='yolov5s.pt',  # 预训练权重，第一次会自动下载
        project='runs/train',  # 保存结果文件夹
        name='workwear_exp',  # 本次训练实验名称
        exist_ok=True,  # 同名文件夹允许覆盖
        workers=0,
        device=0
    )
if __name__ == '__main__':
    main()
```

图 4-3 训练参数

4.4 模型性能分析

4.4.1 验证集结果

图 4-4 为 YOLOv5s 验证集的图中的一个，从这张 YOLOv5s 训练结果的可视化图来看，模型在多场景作业环境（如建筑工地、户外施工、道路等）下对多类别目标（人员 Person、安全装备 Safety Vest/Hardhat、工程车辆 wheel loader/truck/sedan、工程设备 Excavator 等）展现出良好检测能力：各目标的边界框定位准确，分类标签与目标匹配度高（如戴安全帽、穿安全背心的人员被同时标注对应类别，工程车辆和设备也被精准识别）；且覆盖了复杂背景下的小目标（如远处作业人员）与大目标（如工程车辆），验证了 YOLOv5s 轻量化架构在特定工业场景中对多类任务的适应性与检测精度，虽需结合 mAP 等量化指标进一步评估，但可视化层面已体现训练后模型对场景内核心目标的有效识别与定位能力。

图 4-4 验证集结果

4.4.2 性能评估结果

运行 trian_yolov5.py 终端输出结果展示了对 114 张验证集图像的性能评估结果，基于公式 4.4.1 的精度计算公式，以及公式 4.4.2 召回率的公式和公式 4.4.3F1 Score 的公式计算结果如图 4-5 所示，该验证结果呈现基于 YOLOv5s 模型（157 层网络、707 万余参数）对 114 张验证集图像（共 733 个目标实例）的检测表现：全类别精度达 0.832 但召回仅 0.322、mAP50 - 95 低至 0.226，体现“精度尚可但召回与多 IoU 鲁棒性不足”；类别间分化显著，Hardhat、Mask 等特征辨识度高或样本充足的类别 mAP50 超 0.7，而 Gloves（极小目标）召回仅 0.12、mini - van 等单

样本类别召回为 0，暴露轻量模型架构与“大量小目标+少样本类别”高难度数据特性的矛盾，为后续针对性优化（如小目标增强、少样本补全、模型升级）提供了依据。

$$Precision = \frac{TP}{TP+FP} \quad （4.4.1）$$

$$Recall = \frac{TP}{TP+FN} \quad （4.4.2）$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall} \quad （4.4.3）$$

Model summary: 157 layers, 7077550 parameters, 0 gradients, 16.0 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95: 100%||||||||||
all	114	733	0.832	0.322	0.384	0.226
Excavator	114	12	0.635	0.667	0.701	0.47
Gloves	114	25	0.526	0.12	0.183	0.0628
Hardhat	114	79	0.908	0.628	0.715	0.438
Ladder	114	10	0.564	0.6	0.512	0.35
Mask	114	21	0.904	0.762	0.773	0.492
NO-Hardhat	114	69	0.842	0.464	0.526	0.276
NO-Mask	114	74	0.735	0.284	0.402	0.125
NO-Safety Vest	114	106	0.864	0.415	0.545	0.321
Person	114	166	0.925	0.554	0.732	0.445
Safety Cone	114	44	0.84	0.682	0.76	0.339
Safety Vest	114	41	0.714	0.512	0.617	0.34
dump truck	114	13	0.48	0.462	0.551	0.347
machinery	114	8	1	0	0.104	0.0608
mini-van	114	1	1	0	0	0
sedan	114	13	1	0	0.163	0.0699
trailer	114	1	1	0	0.0216	0.0151
truck	114	4	1	0	0.0374	0.0323
truck and trailer	114	3	1	0	0	0
van	114	3	1	0	0.0212	0.0139
vehicle	114	18	0.615	0.111	0.168	0.136
wheel loader	114	22	0.928	0.5	0.539	0.402

* Results saved to **runs\train\workwear_exp**

图 4-5 性能评估结果

4.4.3 混淆矩阵

基于 YOLOv5s 轻量架构（708 万参数量）、30 轮训练及 AutoAnchor 锚框优化的设定下，模型对大尺寸、视觉特征明确的类别（如 sedan semi trailer 等车辆）展现出极强分类能力（混淆矩阵对角线深蓝色，sedan 等类别正确率近 100%）——这类目标因轮廓清晰、核心特征易捕捉，天然适配轻量模型的高效推理逻辑；但对极小目标与弱区分度负类（如 Gloves NO-Hardhat NO-Mask）表现疲软：Gloves 分类正确率仅 0.36，且真实为 Gloves 的样本超 64%被误判为 background，根源是其“<3 像素”的极小尺寸导致特征稀缺，模型易将其混淆为无意义背景；NO 系列负类正确率仅 0.46 – 0.49，因“未佩戴”状态与正类（Hardhat Mask）视觉差异微弱，叠加小目标特性进一步加剧识别难度。从错误分布看，“小目标→背

景”“负类内部/与 Person 的交叉混淆”是核心痛点。

针对此现状，需在训练侧强化小目标数据增强（如超分辨率放大、小目标复制粘贴到背景）、扩充负类样本多样性（采集不同光线/角度下的“未佩戴”场景）；模型侧可尝试升级至 YOLOv5m（提升特征提取容量），或嵌入 CBAM 注意力机制聚焦小目标区域，以在“轻量部署”与“小目标精度”间找到平衡。

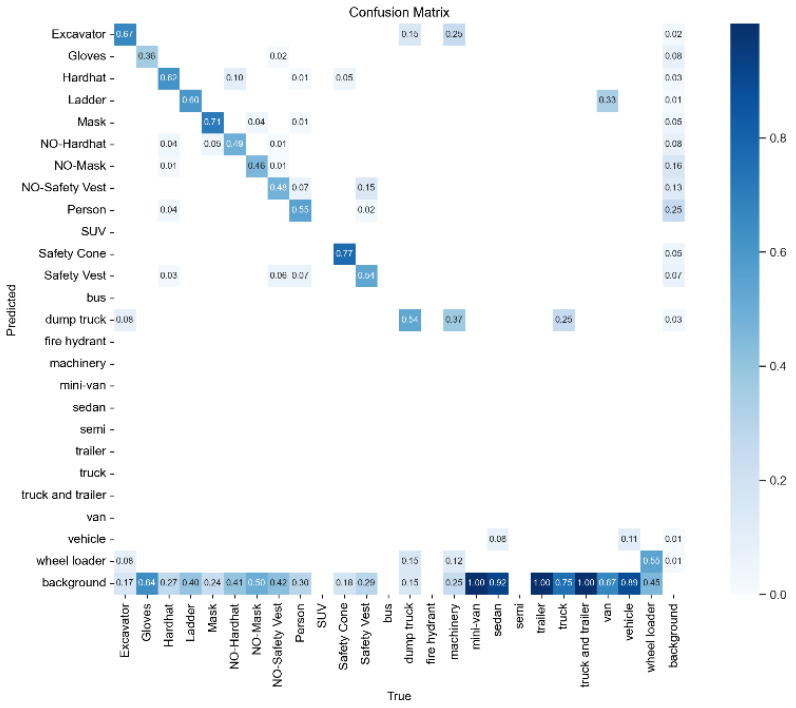

图 4-6 混淆矩阵

4.4.4 F1 曲线

结合 F1 - Confidence 曲线、训练配置与混淆矩阵表现来看：蓝色全局线标注“all classes 0.38 at 0.519”，说明全类别最优 F1 仅 0.38（对应置信度阈值 0.519），这一偏低表现既呼应混淆矩阵中小目标（如 Gloves）与弱区分度负类（如 NO - Hardhat）的分类短板——这些类别对应灰色线在中高置信度（＞0.5）后 F1 断崖式下跌，根源是小目标特征稀缺、负类与正类视觉差异弱，模型对其预测置信度不稳定，高阈值下易漏检/误判；也验证了大目标（如 sedan truck）的优势——对应部分灰色线在中高置信度区间仍维持较高 F1，契合其“尺寸大、核心特征易捕捉”的特性。

结合训练配置中 YOLOv5s 轻量架构（708 万参数量）与“529 个＜3 像素极小目标”的场景挑战，轻量模型在特征提取容量上的天然限制，导致小目标在高置信度要求下（如工业场景需精准输出时）性能崩盘，最终拉低全局 F1；而

AutoAnchor 虽通过聚类优化了锚框适配性，但未从根本解决小目标“特征学习难”的核心痛点，这也解释了曲线中“小目标类别的 F1 随置信度上升快速衰减”的现象。

若要突破瓶颈，需在数据侧强化小目标可见性（如超分辨率放大、小目标复制粘贴）、模型侧升级特征容量（如切换 YOLOv5m）或嵌入注意力机制（如 CBAM 聚焦小目标区域），才能让全局 F1 与高置信度下的小目标性能同步提升。

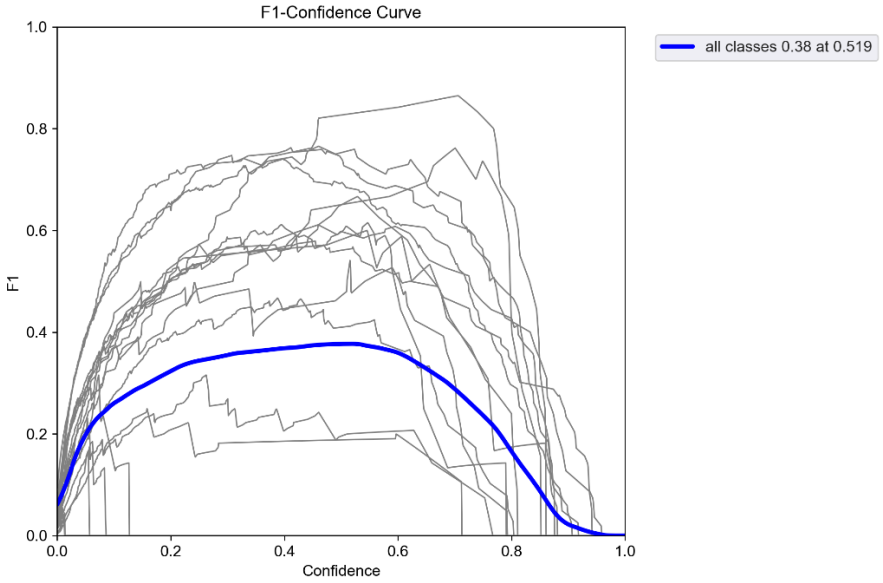

图 4-6 F1 曲线

4.4.5 多维度统计可视化

图 4-7 是目标检测数据集的多维度统计可视化，从类别分布、锚框适配、目标位置与尺寸四个维度，为模型训练提供数据特性支撑。图片中的 x 轴则为训练的轮次，代表模型在训练过程中经历的迭代步骤，y 轴则是损失函数和评估指标。

左上角的类别实例数柱状图显示，25 个类别存在显著类别不平衡（如“Person”“NO-Mask”等实例数超 700，而“wheel loader”“truck and trailer”等不足 100），这种差异易导致模型对小众类别特征学习不充分；右上角的锚框（或边界框）叠加图，通过多层嵌套的彩色线条呈现锚框分布，中心聚集的橙色小框与外围绿色框的对比，验证了 AutoAnchor 优化后锚框对数据集的适配性，但需关注小尺寸目标是否被锚框有效覆盖；左下角的目标中心（x,y）热力图，以颜色深度反映目标实例的位置密度，可见 x≈0.3 - 0.5、y≈0.3 - 0.6 区域高度聚集，暴露目标位置偏向问题——模型易在图像边缘区域对目标漏检；右下角的目标宽高（width,height）热力图，进一步强化“小目标主导”的数据集特性，小尺寸

（width、height 接近 0）目标在低数值区间密集分布，叠加 YOLOv5s 轻量模型的特征提取容量限制，将显著加剧小目标检测的难度。

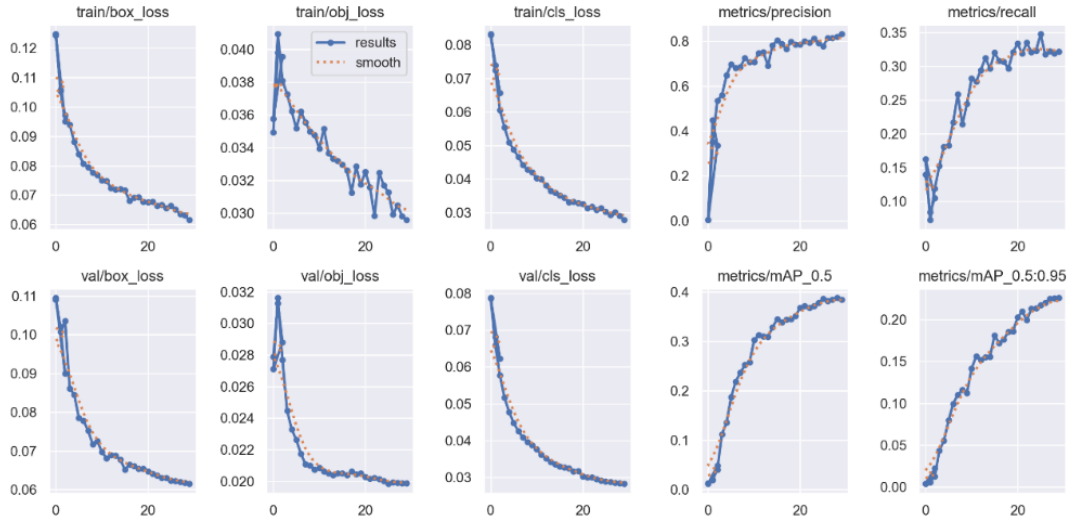

图 4-7 多维统计

4.5 模型推理

在完成模型训练并通过验证集进行评估后，模型进入实际推理阶段。本实验通过 YOLOv5 实现了对“工作服”相关目标的自动检测，推理目标涵盖安全帽、安全背心、挖掘机、人员等四个类别。推理结果如图 4-8 所示，其中模型对图像中的多个目标进行了识别与标注，同时给出了各个类别的置信度分数，体现了模型在目标定位和类别判别方面的整体能力。

图 4-8 预测结果

从图 4-8 中可以看到，模型成功识别出了“Person”（人员），置信度为 0.61；“Excavator”（挖掘机）置信度为 0.55；“Hardhat”（安全帽）置信度为 0.47；

“Safety Vest”（安全背心）置信度为0.50；“No-Safety Vest”（未穿安全背心）置信度为0.36。此外，“Hardhat”与“Safety Vest”这两个目标在图中明显被正确框选出来，显示出模型已具备一定的检测准确性。但同时也存在类别冲突的情况，即在一名穿戴安全背心的工人身上同时被标注了“Safety Vest”和“No-Safety Vest”两种标签，这说明模型在多标签判断上仍存在混淆现象，进一步分析发现，造成该种误识别的主要原因包括以下几个方面。

首先，数据集中某些图像在标注时可能存在交叉标签或标签边界模糊的现象，这会导致模型学习到不一致的信息，从而在推理阶段出现标签冲突。例如，对于颜色类似但未贴合身体的背心模型可能在训练中被同时标注为“Safety Vest”与“No-Safety Vest”，进而造成判别模糊。其次，从图像内容本身来看，推理图像背景较为复杂，包含多个建筑结构、模糊的远景目标和遮挡物体。这种环境会干扰模型提取特征的准确性，特别是在较小物体（如远处挖掘机）或与背景融合度较高的目标（如亮黄色的安全帽与背景色接近）上，模型容易出现漏检或误检。此外，图中部分目标（如人员的手臂）部分被遮挡或处于阴影中，可能使模型难以提取完整边界特征。

而且，在多类别相似区域识别问题中，诸如“Safety Vest”与“No-Safety Vest”这样差异微妙但语义差别显著的标签，本身就存在高混淆性。YOLOv5在使用 anchor-based 检测时对边界框位置和类别标签是同时预测的，当训练样本对两个类别之间的区分不够明确时，模型在边界框回归与类别分类上容易做出混淆判断。此外，推理图像中的目标有一定的密度特性，尤其人员与设备之间存在遮挡或重叠区域。这种情况下，模型需要同时预测多个目标的边界框，在密集区域的非极大值抑制（NMS）处理过程中可能会抑制掉得分稍低但实际为正确目标的框，进而出现漏检问题。例如图中“Hardhat”的置信度仅为 0.47，可能在其它图像中因得分偏低而未被保留。

为了进一步提升模型推理准确性，可以从以下几方面进行改进：一是加强训练数据集质量，确保标注准确、边界清晰，并针对高混淆类别进行专门的人工审核与清洗；二是引入更多增强策略，如加入背景噪声、目标遮挡模拟等，使模型更具鲁棒性；三是在模型结构上引入注意力机制，如 SE-block 或 CBAM 模块，以加强对关键区域的特征提取能力；四是尝试使用更高级的后处理策略，例如Soft-NMS 或类别置信度融合策略，以缓解高重叠区域的检测干扰。由此说明了，

模型推理阶段基本能够完成对图像中多类目标的检测任务，但在多标签冲突、小物体检测与复杂背景干扰等方面仍存在不足。

5 调试分析

(1) 显存不足

问题描述：在模型训练初期，曾多次出现显存占用过高，导致训练进程被中断的问题。经排查发现，主要原因包括两个方面：其一是初始数据集中标签格式不规范，存在冗余或错误标注，影响了模型对样本的加载与解析效率；其二是训练参数设置不当，尤其是批次大小（batch size）设定过大，超过了显卡所能承受的显存容量。

解决方法：首先，对数据集进行了系统性清洗，统一并规范了标签文件的格式，剔除了重复或不完整标注的样本。其次，合理调小了 batch size，并根据新设置同步调整了学习率（learning rate），避免了因显存压力过大导致的训练失败问题，从而实现模型的稳定训练。

(2) GPU 无法正常调用问题

问题描述：在部分训练阶段，发现 YOLOv5 未能有效利用 GPU 资源，训练过程全部运行在 CPU 上，极大降低了效率。虽然系统中的 CUDA 驱动与 PyTorch 均已正确安装，初步分析可能是模型默认未检测到可用 GPU 资源或训练脚本未正确配置设备参数。

解决方法：通过命令行工具 nvidia-smi 检查 GPU 运行状态及驱动是否处于正常加载状态，确认硬件及驱动无误后，在训练脚本中显式指定 device=0，即强制使用 GPU 0 号设备进行训练，有效解决了训练过程无法调用 GPU 的问题，大幅提升训练速度。

(3) 虚拟内存不足

原因：在模型运行过程中偶发出现系统报错提示虚拟内存不足，尤其在训练过程中使用多进程加载数据时，资源消耗急剧增加，导致部分线程无法正常分配内存，进而引发程序中断。该问题多见于 RAM 较小或页面文件（Pagefile）配置较低的系统环境中。

解决方法：运行 YOLOv5 使用了较大的 batch size / 多进程（workers=8），加剧了内存消耗。将 workers 设置为 0，禁用多进程加载。

6 总结

本系统基于 YOLOv5 模型的工作服装识别项目设计与实现，围绕工业现场安全防护需求，探索了深度学习技术在实际应用中的有效落地。以目标检测任务为整体工作集中，利用包含 25 个类别的复杂安全服装数据集，通过预训练模型完成多目标、多类别的自动识别，取得了较为满意的效果。

本项目的显着优点体现在模型选型和数据应用的契合度上。YOLOv5 作为当前领先的单级目标检测框架，兼具高效精度与推理能力，满足了工业现场动态监控和快速反馈的需求。同时，采用预训练的 YOLOv5s 模型轻量且易于训练，显着着了周期实验。此时数据集拥有丰富的实时标识信息且类别相对均衡，为模型提供了充分的数据支撑，保证了训练的有效性。通过科学的数据消耗和增强策略，模型对提示、色彩及形态变化表现出较好的鲁棒性，能够适应工地环境的复杂性。

然而，该项目也存在一些不足和挑战。首先，训练虽然取得了一定的性能提升，部分词语类别如“安全”其次，工作场景中目标覆盖、重叠和多元化带来背景的误检和漏检问题依然存在，未来需融合考虑更复杂的模型结构或多模态信息来增强判别的力。此外，当前实验已经进入更精细的像素级目标分割阶段，限制了对装备覆盖范围训练参数中采用了 SGD 优化器和固定学习率策略，虽表现稳定，但调整优化空间，自适应优化算法和动态学习率调整可引入。最后，由于硬件与资源限制考虑，本次实验在计算资源利用率和训练效率上尚有提升潜力，尤其是针对大型数据集的训练及模型轻量化方面可进一步优化。

针对以上问题，后续研究可从多方面展开。可以通过加强数据敏锐与突出，特别是提升问题类别样本的数量和质量，结合综合提升数据和半学习监督，模型的整体表现和对稀少类别的识别能力。最后，利用充分的硬件优化手段，开展全面和任务训练，提高模型训练效率，并研究模型量化和剪枝技术，确保模型在边缘设备端的轻量部署。

总体来看，本次工作服识别在应用训练新兴学习技术、利用合理预模型及数据集管理方面，取得了良好的实践成果，为智能安全监控领域提供了示范。后续持续优化与扩展，相信系统能够通过项目更加稳定、精准地推动工业安全管理，推动智能制造和健康生产环境的建设。

7 参考文献

[1] Linjie Yang, Chenxi Liu, Lu Qi, et al. ONCE: A Large-Scale and Diverse Dataset for Autonomous Driving [J]. Computer Vision and Image Understanding, 2023, 226(104105): 1-13.

[2] Glenn Jocher, Ayush Chaurasia, Laughing-Q, et al. YOLO by Ultralytics: Real-time object detection algorithms [M]. GitHub repository. 2024.

[3] 张明，李强，王涛. 基于深度学习的工业安全防护装备检测研究[J]. 电子学报, 2020, 48(5): 987-995.

[4] 胡平. 深度学习应用实践[M]. 北京：机械工业出版社, 2019.