中南大學
CENTRAL SOUTH UNIVERSITY

工业互联网导论
（设计报告）

题　　目：基于云边端架构的钢材表面缺陷检测系统设计

学生姓名：白泽阳

指导老师：黄科科 教授

学　　院：自动化学院

专业班级：人工智能 2102 班

2023 年 6 月

基于云边端架构的钢材表面缺陷检测系统设计

摘要

钢材表面缺陷检测的困难性因复杂的工业环境及高误漏检率而增加，导致低效的检测过程。因此，运用深度学习快速精确地识别钢材表面缺陷具有重大研究价值。本研究以广泛应用的钢材为对象，利用基于MSFT-YOLO的深度学习目标检测算法进行表面缺陷检测研究。

首先，我们对NEU-DET数据集进行了数据增强以解决样本少且训练效果不佳的问题，提高了网络模型的鲁棒性，防止过拟合。针对工业场景中的图像背景干扰、缺陷类别混淆、缺陷尺度变化及小缺陷检测效果不佳等问题，我们通过在模型主干和检测头中添加基于Transformer设计的TRANS模块，使特征与全局信息相结合，通过多尺度特征融合结构增强了对不同尺度目标的动态调整能力。

为了进一步提升MSFT-YOLO的性能，我们引入了策略如数据扩充和多步训练等。测试结果显示，MSPF-YOLO可以实现实时检测，而MSFT-YOLO的平均检测准确率达到69.3%，比基线模型(YOLOv5)提高约7%，表现出优势和启示性。

针对当前工业环境对钢材表面缺陷检测的需求，我们利用物联网和大数据技术，采用工业摄像头拍摄钢材图片，通过五连死亡传输协议传输至边缘设备进行预处理，然后上传至云端。云端部署的缺陷检测模型对图片进行检测，并返回训练结果。

关键词：钢材表面缺陷检测　YOLOv5　MSFT-YOLO　深度学习　云边协同

目录

第 1 章　绪论

1.1　课题研究背景及意义

随着科技的持续发展和生产力的不断提升， 我国在工业智能化及其相关领域取得了巨大的进步， 已然成为了新的“世界工厂”。市场对工业产品的外观及质量也提出了更高的要求， 这要求传统的生产模式要向智能化生产模式进行转变。从宏观角度来看， 制造业的智能化、自动化、无人化将是必然的趋势。世界各国都对智能化工业生产趋势做出了积极的响应， 而我国提出“中国制造 2025”战略以目前工业制造技术为基础， 利用互联网， 人工智能等技术优化工业生产结构、提高产量， 加强新兴科学技术与工业制造之间的联系，以实现我国制造业的腾飞发展。

中国钢铁行业正在经历规模迅速扩张、产量飞速提升的一段时期， 该时期着力于解决国民经济发展和用钢问题。但在钢材的生产和加工过程中， 容易受到环境中不良因素的影响， 从而使表面产生多种类型的缺陷。划痕、开裂、斑块等典型的钢材表面缺陷会导致钢材的性能和使用年限降低， 甚至直接造成安全事故的发生。因此， 表面缺陷检测在钢材生产过程中是一个十分重要的环节， 各大企业也将其视为提高产品质量的关键技术， 相关学者也在进行钢材表面缺陷检测方法的研究。最初钢材表面缺陷检测主要靠

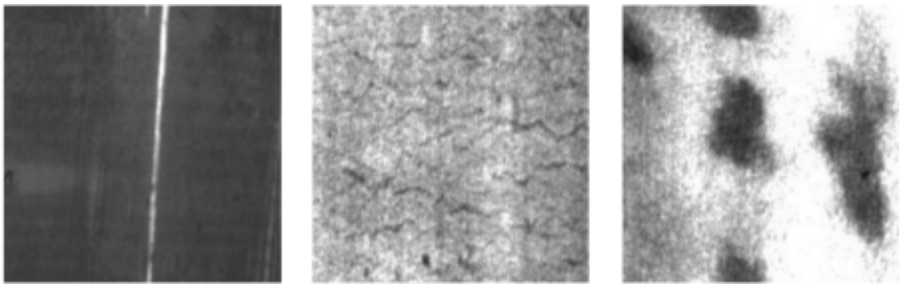

图1 钢材表面缺陷

人工目检， 这种方法不仅成本高、检测效率低， 更重要的是会出现大量误检和漏检， 在实际的工业环境中并不适用。之后出现了一些特定产品缺陷的检测方法来替代人工检测， 例如涡流无损检测、微波检测、线结构光扫描检测等， 但这些检测方法仍然存在材料限制、无法进行缺陷分类等问题， 不能很好的满足精确高效的检测要求。 HALCON、OpenCV、VisionPro 等机器视觉软件在应用中同样存在一些局限， 如版权限制、功能单一等。

目前计算机视觉的检测方法已经在许多工业场景中落地应用， 运用深度学习目标检测方法的大致流程为： 首先使用成像设备对金属生产过程中的表面图像或视频进行拍摄； 然后将其输入到训练好的网络模型中； 最后， 网络模型就可以自动推理出缺陷钢材表面缺陷区域并进行分类。相比于传统方法， 与深度学习相关的方法不再局限于某种固定程序， 而是根据卷积神经网络（Convolutional Neural Network，CNN）对缺陷样本进

行自主学习， 从而克服了一个传统算法只能检测一种缺陷的问题。还可以通过改进网络结构， 参数调整等方式优化网络模型， 使之可以检测出面积微小、特征模糊的缺陷。因此基于深度学习的钢材表面缺陷检测方法已成为目前工业生产和学术研究的热点。

1.2 国内外研究现状

目标检测是计算机视觉领域最关键的下游任务之一， 对于目标在图像中的定位和类别信息进行预测是其主要目的， 在工业场景中具有重要的应用价值。由于钢材表面缺陷检测主要将缺陷的位置和特征信息作为依据来对缺陷进行预测， 所以深度学习目标检测算法同样可以应用在钢材表面缺陷检测中。因此， 国内外研究现状将从深度学习目标检测算法和钢材表面缺陷检测算法这两个方面来进行的介绍。

1.2.1 深度学习目标检测算法研究现状

在 1991 年被提出的人脸检测算法是目标检测方向最早的研究。与深度学习目标检测方法不同， 机器视觉方法主要由以下阶段组成： 区域选择(Sliding window)、特征提取(SIFT 、HOG 等) 以及分类器(SVM 、Adaboost) 三个阶段组成， 具有计算复杂度高、复杂工业场景下适应能力弱等问题。研究者必须研究多种算法以弥补手工设计特征在表达能力上的不足同时还要设法减少模型的计算量。

2012 年被学术界称为深度学习的元年， 这一年 Hinton 等人提出了利用卷积神经网络提取输入图像中的隐藏特征的 AlexNet 网络， 并在 ImageNet 竞赛中获得冠军。卷积神经网络可以极大地提升了网络对于目标特征提取地能力， 自此越来越多的学者展开了深度学习方面的研究。与传统目标检测算法不同的是， 速度快、精度高、可移植、鲁棒性强是深度学习目标检测算法的特点。

按照不同的阶段数量， 目标检测算法可以分为两阶段目标检测(Two-stage) 和单阶段目标检测(One-stage) 。两阶段目标检测算法在某种意义上是传统目标检测算法的研究方式上的延伸， 其主要由两个阶段组成： 第一阶段通过区域建议网络(Region Proposal Network ， RPN) 生成候选区域(Region Proposal) 并使用 CNN 对目标进行隐藏特征提取； 第二阶段使用分类器分类并修正位置坐标。为了满足工业和生活场景的需要， 获得更加轻便地网络模型以提升算法的实时性， 相关学者提出了单阶段目标检测算法， 其将目标检测视为回归任务， 去掉了区域建议网络， 直接在图像中回归出目标的位置和类别。不同的目标检测算法对不同的应用场景有不同的侧重点， 每个算法也都有其最佳的适用范围。

(1) 两阶段目标检测算法

2014 年， Ross Girshick 等人提出 R-CNN 目标检测算法， 目标检测的精确率被极大地提高了， 同时 R-CNN 是在全部的候选区域上提取特征信息， 会产生重复多余的数据， 使推理速度下降。所以运行 R-CNN 非常消耗算力， 也不满足实际场景中实时性的要求。

2015 年， Kaiming He 等人为改善 R-CNN 重复计算的问题提出 SPP-Net 目标检测算法， 在 CNN 最后一个卷积层添加了 SPP Pooling，使用不同尺寸的特征图输入， 然

后输出固定尺寸的图片。但同 R-CNN 一样具有消耗过多的内存和算力的缺点。

2015 年， Ross Girshick 改进了 SPP-Net 并提出 Fast R-CNN 目标检测算法， 提高了计算速度，同时在一定程度上降低了内存和算力的占用。

2015 年， Kaiming He 等人提出 Faster R-CNN 目标检测算法， 提出使用 RPN 进行边框训练来解决选择性搜索(Selective Search) 效率慢的问题。但沿用的感兴趣区域(Region of Interest，ROI) 候选网络， 进行过多的下采样模糊了目标的细节信息， 使得其在小目标检测上的效果还有待提升。同时受限于两阶段目标检测算法的理念， Faster R-CNN 算法的推理速度仍然很慢。

2016 年，Jifeng Dai 等人提出 R-FCN 目标检测算法， 为实现更高效的目标检测， 该算法运用了基于区域的全卷积神经网络。与 R-CNN 系列应用了数百个计算代价高昂的区域子网络相比， 全卷积网络几乎所有计算都在图像上。其主干使用残差网络(Residual Network，ResNet)在数据集相同情况下速度和精度均超过了 Faster R-CNN。

(2)单阶段目标检测算法

2016 年， Joseph Redmon 等人提出 YOLO 目标检测算法， 由两阶段目标检测算法发展而来， 摒弃了提取候选区域的过程， 只用一阶段就完成了识别任务， 大大提升了目标检测速度。

2016 年， Wei Liu 等人提出 SSD 目标检测算法， SSD 是一种用于目标检测的单一卷积神经网络结构， 其主要运用了多尺度多长宽比的密集锚点和特征金字塔网络， 在增加了检测速度的同时几乎不损失检测精度。

2017 年， Joseph Redmon 等人在 YOLO 的基础上提出了 YOLOv2 目标检测算法，其进行了联合训练策略的实验并且解决了过拟合以及预设先验框不合理的问题， 同时提高了检测速度和检测精度。

2018 年，Joseph Redmon 等人将 YOLOv2 进行了全面的升级， 提出了 YOLOv3[24]在保持高检测速度的前提下， 重点增强了对小目标的检测效果。同时对每个目标候选框增加了概率性预测，可一定程度地优化前景和背景类别不均衡的问题。

2018 年， Tsung-Yi Lin 等人提出了 RetinaNet，不需要过多改变网络结构， 只需要将损失函数替换为聚焦损失函数 Focal Loss 就可以进一步解决前景和背景样本不均衡的问题，且有效地解决了密集目标检测中的焦距损失。

2020 年，Alexey Bochkovskiy 等人以 YOLOv3 为基础提出 YOLOv4 在 CSPNet 中增加了空间金字塔池化(Spatial Pyramid Pooling，SPP) 和 PANet 结构， 同时还融合了了多种数据增强策略实现了检测速度和检测精度的进一步提升。

2020 年， Mingxing Tan 等人以 EfficientNet 为骨干网络提出了 EfficientDet，它内置了一系列可供自主扩展的目标检测网络， 其发展了 EfficientNet 中复合和放缩的理念，可使用户根据实际任务进行自定义网络模型结构，平衡了检测的效果与速度。

2020 年， Glenn Jocher 等人在 GitHub 社区开放了 YOLOv5 的源码， 作者在改进网络结构的同时使用了一些训练技巧， 进一步在提升精了检测确度和速度。与 YOLOv4 不同的是， 其同时推出了不同网络模型大小的 4 个版本， 并且使用 PyTorch 平台， 利

于二次开发和部署，这使得 YOLOv5 成为当前业界最优秀的目标检测算法之一。

1.2.2 钢材表面缺陷检测算法研究现状

传统的机器视觉方法通常使用固定的程序或算法来进行缺陷检测， 算法的原理包含数学、物理和计算机等相关学科， 设计流程较为复杂。光源、工业相机、算法程序和控制结构等部分一起构成了工业中机器视觉的检测环境。首先将目标通过相机拍摄得到图像信息， 然后将图像传输至程序算法并进行检测， 最后依靠检测结果或相应要求进行下一步操作。基于传统机器视觉的钢材表面缺陷检测方法大多为基于图像分割、特征提取和光谱计算的算法， 主要针对特定的缺陷对象进行检测， 并不具有普适性。基于深度学习的钢材表面缺陷检测任务由缺陷定位和分类两个子任务组成。由于无法给出缺陷定位的相关信息， 分类任务适用仅需要分辨出缺陷类别的场景。为同时得到分类和定位信息， 两阶段目标检测算法的做法是先得到缺陷的候选区域， 再对缺陷的候选区域而不是整个图像进行分类。单阶段目标检测算法做法是将缺陷分类和定位视为同一个任务进行回归， 虽然损失了一些精度， 但使得检测速度极大地提高。基于深度学习的钢材表面缺陷检测方法，对不同的缺陷数据和样本均适用，在工业中的应用更为广泛。

（1）基于传统机器视觉的钢材表面缺陷检测算法

2008 年， Jong Pil Yun 等人提出了一种基于非抽样小波变换的钢材缺陷检测方法。该方法的数据来源为带钢原材料表面图像， 可以将缺陷尺寸差异的影响降到最低， 实验结果表明该方法对于钢材表面的线形缺陷具有良好的检测效果。

2009 年， Jong PilYun 等人为解决钢材表面缺陷灰度水平与背景相似度高的问题，提出了一种基于 Gabor 滤波器的缺陷检测算法。其使用 uDEAS 算法计算能量分离目标函数的极小值。最后在实际钢坯表面图像上进行的实验结果验证了该算法的有效性。

2010 年， Luiz A.O. Martins 等人提出了一个基于图像分析技术的钢材缺陷自动分类系统， 对 3 种复杂几何形状的钢材缺陷进行分类。其运用了主成分分析和自组织图的方法进行特征提取，系统经过验证可以有效地对 3 种缺陷进行分类。

2012 年， Chang Hyun Park 等人针对钢板黑色树脂层表面缺陷检测问题， 提出了一种基于图像的二阶统计的检测方法。并使用支持向量机（SVM）将检测到的缺陷对象进行类别区分，研究结果表明该方法可以检测到大多数的缺陷。

2013 年， 管声启等人针对带钢表面缺陷的特点， 提出一种基于图像零均值化的检测方法。其利用零均值化、维纳滤波和 Sobel 锐化等方法弱化环境噪声干扰， 并使用最大类间方差法对带钢表面缺陷进行了识别。

2016 年， 陈海永等人提出一种基于谱残差视觉注意模型的带钢表面缺陷在线检测算法。该算法检测速度快， 解决了带钢缺陷检测实时性的问题并且降低了环境光源对特征信息提取的影响。实验结果表明该在线算法对带钢常见缺陷类型有着较低的漏检率和误检率。

（2）基于深度学习的钢材表面缺陷检测算法

2017 年， 邢健夫构建了一种基于 AlexNet 的卷积神经网络模型对钢材表面缺陷进行分类。该研究将多种带钢表面缺陷进行数据扩充， 并制作成带钢表面缺陷数据集。在

该数据集中以原始网络与改进的网络进行对比试验， 实验结果表明改进方法增强了网络对于带钢表明缺陷的分类能力。

2018 年， Jiangyun Li 等人改进了 YOLO 网络， 将其变为全卷积的形式。用含有 6 种带钢表面缺陷的自建数据集进行训练， 得到的模型可以预测缺陷的位置和尺寸信息，对于实时的钢带表面缺陷检测提供了方法论支持，对带钢质量评价也具有重要意义。

2019 年，陈建强等人以 SSD 网络作为基础进行改进，利用交叉特征融合的方法增强低层特征图的语义信息， 从而提高带钢表面缺陷的识别率。钢带表面小尺寸缺陷检测的效果有明显提升，且满足实时检测的需求。

2019 年， Yu He 等人提出了基于改进 Faster R-CNN 的带钢表面缺陷检测网络， 将骨干网络中不同层级的特征图结合为一个多尺度特征图， 更好地利用了全局的语义信息， 减少特征丢失。其使骨干网络使用 ResNet 进行特征提取并在 NEU-DET 数据集上取得了较高的精度值。

2020 年， 徐镪等人基于 YOLOv3 网络进行改进， 将特征提取网络替换成轻量级 MobileNet，并使用 Inception 结构来减少模型参数量， 以提检测的升实时性。利用空洞卷积增强提取小目标特征的能力，从而提高带钢表面缺陷整体的检测精度。

2021 年， 罗晖等人提出一种基于图像增强与改进 Cascade R-CNN 的钢轨表面缺陷检测方法。应用交并比(IoU) 平衡采样、感兴趣区域对齐和完全交并比(CIoU) 损失来解决特征图不匹配和预测边框回归不准确的问题。该方法的平均精度相对于未改进的 Cascade R-CNN 高，且缩短了检测时间。

1.2.3 钢材表面缺陷检测关键问题

在钢材表面缺陷检测任务中， 算法中网络模型优化和部署是极其重要的环节， 每种算法都有其最佳的适用范围和场景， 目前各类研究旨在提高检测的准确性和实时性。根据国内外的研究现状可以看出， 现有一些的检测方法已初具成效， 不过依然存在几个关键性的问题：

(1) 缺陷特征提取问题

由于工业相机、环境光源以及其他不可控因素会对钢材表面缺陷图像的采集造成影响， 会导致采集到的图像随机性较强， 图像中的特征信息也不够丰富。一些基础网络结构容易受到其影响而无法充分地学习到缺陷的特征， 致使现有检测方法存在检测精度低、漏检率和误检率高的问题。因此现阶段的一个重要问题就是如何稳定且充分地提取缺陷的特征， 以适应光源和环境对图像成像的干扰， 能够高效地学习到缺陷的特征也是网络可以适应更多数据样本和场景变化的前提。

(2) 小目标缺陷检测问题

对于小目标的判定学术界有两种标准， 一种是在 COCO 数据集中提出的绝对尺度定义， 即当目标像素尺寸小于 32×32 时会被认定为小目标； 第二种是相对尺度定义， 即当目标区域尺寸占比小于原图的 10% 时会被认定为小目标。根据相关调查和研究发现，钢材表面会存在许多尺寸较小、像素占比少并且分布密集的小目标缺陷。其自身存在着语义信息少、覆盖面积小、边缘信息模糊等先天不足等问题。特征提取网络在进行下采

样操作时会对连续将特征图缩小后， 可能会损失很多小目标缺陷的位置和语义信息， 导致其检测效果并不理想，甚至出现漏检的情况。

(3) 缺陷定位偏差问题

基于深度学习的钢材表面缺陷检测任务最终的输出包含两个方面： 一方面是包含有缺陷目标位置信息的矩形预测框， 另一方面是与每个预测框相对应的缺陷目标分类信息。由于全监督的深度学习目标检测方法对于人工标注数据集的准确度要求苛刻， 在图像上手工标记的目标矩形框会存在一定误差， 这些误差会影响到网络模型对缺陷目标位置的学习和预测。最终可能导致预测框与真实值发生较大偏移或多个预测框重叠的现象发生，影响检测的整体精度。因此，对于目标框策略的改进也是十分必要的。

(4) 多类型缺陷分类问题

由于在工艺和理论方面还无法完全解释缺陷形状与产生原因的内在联系， 缺陷类型的划分也没有相对统一的标准， 此前的研究多为基于某一特定类型的缺陷， 如划痕和微小缺陷等。此前的研究多为基于某一特定类型的缺陷。相关的缺陷检测方法具无法适用于所有类型缺陷的检测， 且经常出现错误分类的情况。因此， 本研究中同样存在的一个难点是多种类型钢材表面缺陷的误检问题。

(5) 检测实时性问题

在实际的工业检测线上， 由于计算资源有限， 设备对于图像的响应和处理速度往往无法达到大规模量产的需要。生产者为了控制成本， 计算设备的性能会有瓶颈， 算法中一些冗杂的部分计算会影响整体的检测效率。因此相关的钢材表面缺陷检测算法也应面向工业生产， 换用更轻量高效的算法网络或者计算方式， 在保证检测精度的同时减少计算量， 以满足工业中对于缺陷检测实时性的要求。综上所述钢材表面缺陷检测是集理论研究意义与实际应用价值并存的一项任务， 且存在很多的技术难点， 因此对于钢材表面缺陷检测方法的研究和创新同样也充满了挑战。

1.3 主要研究和章节安排

本研究采用东北大学发布的带钢表面缺陷检测数据集 NEU-DET 为模拟实验研究对象， 充分收集国内外相关的调查和研究资料， 并以此为基础开展基于 MSFT-YOLO 模型的钢材表面缺陷检测的研究。针对钢材表面缺陷图像检测中存在的关键问题， 以提升精度、降低漏检率和误检率为主要目的。

首先对 NEU-DET 数据集进行了深度分析， 根据数据集自身的特点将锚框(Anchor)选择方法进行了改进； 针对图像背景干扰大、缺陷类别易混淆、缺陷尺度变化大、小缺陷检测效果差的工业场景， 提出 MSFT-YOLO 模型。通过在主干和检测头中加入基于 Transformer 设计的 TRANS 模块， 使特征与全局信息相结合。通过组合多尺度特征融合结构对不同尺度的特征进行融合， 增强了检测器对不同尺度目标的动态调整。为了进一步提高 MSFT-YOLO 的性能， 我们还引入了大量有效的策略， 如数据扩充和多步训练方法， 然后通过增加注意力机制来优化特征提取网络， 并在个人服务器上进行部署。在 NEU-DET 数据集上的测试结果表明， MSFT-YOLO 能够实现实时检测， MSFT-YOLO

的平均检测准确率为 69.3%，较基线模型（YOLOv 5）提高约 7 个百分点， 具有一定的优势和启发性。

为适应工业互联网发展趋势和国家工业需要， 我们拟利用网络传输协议搭建云-边-端结构。本次模拟实验设计中， 将上述检测模型搭载在个人服务器上作为云端， 利用个人电脑作为边端， 钢材工厂检测设备作为端。当钢材工厂检测设备通过传感器将图片拍摄记录后， 通过物联网传输协议将原始数据传输到边端。在边端中， 通过数字图像处理技术， 如卷积， 傅里叶变换法、 Gabor 滤波法、小波变换法等具体方法实现图像清理和图像增强， 通过网络传输协议将预处理的图像发送到服务器云端， 在云端实现具体检测后， 将结果返回至边端， 方便进一步管理。鉴于模拟实验的局限性， 我们继续利用东北大学发布的带钢表面缺陷检测数据集 NEU-DET 为模拟实验研究对象， 仅从边端预处理过程开始过程执行。

本文共分六个章节，每章的具体介绍如下：

第一章：绪论。

本章首先深刻地阐述了当前智能制造背景下研究钢材表面缺陷检测方法的意义， 然后详尽地分析了深度学习目标检测算法与钢材表面缺陷检测算法之间的关联和国内外研究现状， 进而总结出钢材表面缺陷检测任务的关键性问题， 最后给出本文的主要研究内容，设计方法思路以及章节安排。

第二章： YOLOv5 网络模型改进。

本章首先介绍了目标检测算法的两种形式， 通过两种形式的比较阐述选择 YOLO 系列为研究基础的原因， 通过目标检测算法常用性能度量的介绍， 讲述了度量性能的依据和各指标评价的关系， 搭建出完整的评价体系， 最后， 通过对 YOLOv1 到 v5 系列模型的特点和缺点分析， 阐述了研究基础 YOLOv5 模型的优势和特点， 最后结合之前对模型的研究， 找出了对钢铁表面缺陷检测存在的一些无法解决的问题， 并用数据事实证明了对于钢铁表面缺陷检测， 改造 YOLOv5 模型的必要性， 最后， 简要引出了 MSFT-YOLO 模型的基本结构，改进点和具体方法。

第三章：基于 MSFT-YOLO 的钢材表面缺陷检测。

本章首先详细介绍了由东北大学发布的钢材表面缺陷检测 NEU-DET 数据集和钢铁表面缺陷的详细信息， 接着讲述了在边端实现的数据增强方案。然后介绍了模型训练的环境， 将改进后的 MSFT-YOLO 网络模型进行训练， 得到精度最优的网络模型， 并与改进前后网络模型性能进行对比分析。

第四章：云边端架构搭建及网络传输协议。

本章根据实际工业中的检测需求围绕轻量化， 模型部署进行了一系列研究。针对大型工业互联网平台开发的大环境需要， 整体架构采用工业互联网的设计思路， 即采用云计算、物联网、大数据等技术相结合的方案， 实现数据的采集、传输、存储和分析。在我们的模拟设想中， 钢铁工厂设备通过传感器将图片数据通过物联网传输协议传输给在另一个地方的钢铁公司管理部门， 公司通过内部电脑， 或是公司服务器汇总数据， 并通过数字图像处理技术实现图像去噪， 增强， 将预处理后的图片上传到云端。通过网络传

输协议， 我们可以实现图片的传输， 并且可以从云端自动获取返回的训练结果， 通过数据处理技术， 可以将处理结果存入公司数据库系统中， 便于进一步管理和分析， 实现工厂， 管理， 算力的远程连接。因此， 我们首先介绍了云边端架构的目标和组成部分， 并简要结合上述设想， 介绍了组成部分在我们系统中的实例， 然后通过 http 协议， MIME 类型和 Base64 编码的介绍， 我们阐述了如何实现云边端架构间的数据通信， 最后， 通过传输图片实例， 服务器反应实例， 和管理员 API 实例， 具体展示实现过程， 自此完整地展示了基于云边端架构的钢铁表面缺陷检测系统的设计方案。

第五章： 总结与展望。主要对全文的研究工作进行梳理和总结， 反思当前研究中存在的不足。结合钢材表面缺陷检测技术的发展情况， 给出自己的判断和思考， 为下一阶段的工作指明方向

第 2 章　YOLOv5 网络模型改进

钢铁缺陷检测问题主要采用目标检测算法进行解决， 而该算法根据是否存在特定的候选区域分为两阶段和单阶段目标检测算法， 以 R-CNN 系列为代表的两阶段的深度学习目标检测算法的思路是： 先使用设计好的区域建议网络生成感兴趣区域， 然后将目标检测任务转变为对生成的感兴趣区域中图像进行分类的任务。由于两阶段目标检测算法只在生成的建议区域内进行类别分类与位置回归， 这样会忽略目标在整个图像上的空间信息， 使得漏检的几率上升。为优化这一问题， 单阶段目标检测算法随着研究的深入被也提出， 它将分类和定位视为同一任务， 加强了空间信息的利用。 YOLO 系列是单阶段目标检测中最优秀的算法。

2.1　性能度量

评价指标可以将深度学习网络训练结果进行量化， 从而反映算法和模型在实际检测过程中的性能。为反映评估模型的性能的优劣， 选取适应问题的、合适的评价指标必不可少。

2.1.1　精确率与召回率

目标检测评价指标中的精确率（Precision Rate）和召回率（Recall Rate）可以通过以下公式进行计算：

$$R_P = \frac{TP}{TP + FP} \tag{2.1}$$

$$R_h = \frac{TP}{TP + FN} \tag{2.2}$$

其中， R_P 为精确率, R_h 为召回率， TP（True Positive）为正样本被正确预测为正的数量， FP（False Positive）为负样本被错误预测为正样本的数量， FN（False Negative）为正样本被错误预测为负的数量。

而对于钢铁缺陷检测———— 目标检测任务而言， 在仅当预测框与真实框的 IOU(交并比) 大于设定阈值， 且分类正确时， 才认为一个正样本被正确预测。交并比阈值的大小会影响预测框筛选的严格程度， 相关研究通常将交并比阈值设置为 0.5。在目标检测任务中一般认为较高的精确率和召回率对应着更低的误检和漏检率。

2.1.2　交并比

目标检测任务中， $b_g t$ 和 $b_p red$ 分别表示物体的实际边界的真实框和检测出的预测框。 $b_g t$ 和 $b_p red$ 的交并比（Intersection over Union，IoU）被用来评价预测框的准确程度。如下图 2.1 所示， 红色矩形框为预测框， 绿色矩形框为真实框， IoU 越大预测框的尺寸和位置就越贴合真实框， 公式 2.3 代表两个边界框重叠的部分与两个边界框的集合部分的比值。

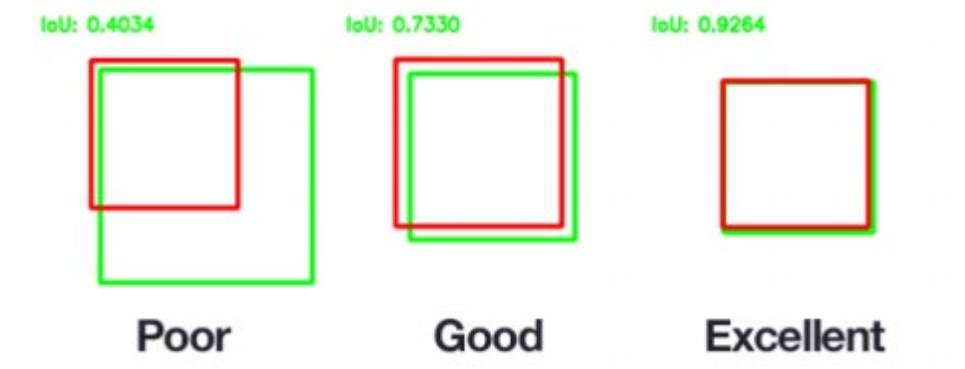

图2 交并比示意图

$$IoU = \frac{Area\ (b_{gt} \cap b_{pred})}{Area\ (b_{gt} \cup b_{pred})} \tag{2.3}$$

2.1.3 平均精度与平均精度均值

平均精度(Average Precision，AP)是目标检测中最基础的评价指标，其公式为：

$$AP = \int_0^1 R_p(t)dt \tag{2.4}$$

其中 $R_p(t)$ 表示交并比阈值取 t 时的精确率。在多类别的目标检测任务中待检测物体有多种分类， 因此通常用平均精度均值(MeanAverage Precision，map) 作为最重要的评价指标，用来反映网络模型对于所有类别目标的检测精度，其计算如下：

$$mAP = \frac{\sum_{n=1}^{N} AP_n}{N} \tag{2.5}$$

本文将以上评价指标统称为精度， 评价指标对目标分类和定位的准确程度进行了量化，数据化的结果也有助于进一步的分析和研究。

2.2 YOLO 系列模型的主要特点

（1） YOLOv1

将目标检测任务视为回归问题是 YOLO 在内的单阶段目标检测算法的基本思想。YOLOv1 以 CNN 作为骨干网络， 使用卷积和池化层对输入图像进行下采样并提取隐藏特征， 再将预测结果从全连接层中输出， 在 VOC2007 和 VOC2012 测试数据集上 YOLOv1 分别得到了 63.4% 和 57.9% 的 mAP。。同时 YOLO 没有基于局部而是基于整张图像的回归， 可以有效地利用空间信息， 大大降低了背景误检率。然而 YOLOv1 仍旧存在一些不足， 相较于等同时期的两阶段目标检测算法， 精度较差， 对于密集小目标或具有异常尺寸的目标检测效果较差。具体如下：

（1.1）由于输出层为全连接层， 因此在检测时， YOLO 训练模型只支持与训练图像相同的输入分辨率。

（1.2）虽然每个格子可以预测 B 个 bounding box，但是最终只选择只选择 IOU 最高的 bounding box 作为物体检测输出， 即每个格子最多只预测出一个物体。当物体占画面比例较小， 如图像中包含畜群或鸟群时， 每个格子包含多个物体， 但却只能检测出其中一个。这是 YOLO 方法的一个缺陷。

（1.3）YOLO loss 函数中， 大物体 IOU 误差和小物体 IOU 误差对网络训练中 loss 贡献值接近(虽然采用求平方根方式， 但没有根本解决问题)。因此， 对于小物体， 小的 IOU 误差也会对网络优化过程造成很大的影响，从而降低了物体检测的定位准确性

（2） YOLOv2

YOLO v2 为提升网络模型的检测性能， 将批量正则化(batch norm)、高分辨率分类器(hi-res classifier)、锚框机制(anchor boxes)、新的骨干网络和尺度先验(dimension priors）等策略加入到算法中， 对比 YOLOv1，其在 VOC2007 和 VOC2012 数据集上的 mAP 值均有明显的增加， 分别达到了 78.6% 和 73.4% 。YOLOv2 中对精度提升帮助较大的策略有尺度先验与位置预测的组合(dimension priors and location prediction） 以及高分辨率分类器(hi-res detector)。

YOLOv2 模型在目标检测任务上取得了很好的性能， 但也存在一些不足之处， 包括：

(2.1）定位精度相对较低： YOLOv2 使用单个边界框来预测每个目标的位置和大小。这种设计限制了其对小目标和密集目标的精确定位能力， 容易出现边界框漏检或者误检的情况。

(2.2）多尺度处理的困难： YOLOv2 在图像上使用了多尺度的特征提取来检测不同大小的目标。虽然这种设计可以处理尺度变化的目标， 但也导致了模型难以捕捉到细节信息。尺度较小的目标往往被模糊掉，而尺度较大的目标可能会被分割成多个部分。

(2.3）类别不平衡问题： YOLOv2 采用交叉熵损失函数进行目标分类， 但在一些场景中， 不同类别的目标数量可能存在较大的不平衡性。这可能导致模型在少数类别上表现较差，因为模型倾向于优化常见类别的预测。

(2.4）对重叠目标的处理有限： 当目标之间有重叠或者部分遮挡时， YOLOv2 很难准确地将它们分割开来。模型通常只能选择一个边界框来预测目标， 这可能导致漏检或者误检。

(2.5）训练样本标注困难： YOLOv2 需要为每个目标提供边界框的位置和类别标签。标注准确的边界框需要耗费大量的时间和人力， 尤其是当目标较小、密集或者有遮挡时。

（3） YOLOv3

YOLOv3 将 YOLOv2 的特征提取网络由 Darknet-19 改为 Darknet-53，增加卷积层的数量能够在不过拟合的情况下更好地抽象出目标特征， 进而增强算法的检测性能。YOLOv3 清楚简洁的结构为此后的单阶段目标检测算法打开了新的思路， 即使对小目标检测的性能方面不如两阶段算法， 但值得肯定的是在增加卷积层的情况下， 还可以保持较高的检测速度。 YOLOv3 还可以方便地部署在移动设备， 因此成为了工业上目标检

测任务的主流算法之一。

尽管 YOLOv3 在 YOLO 系列中取得了进一步的改进和性能提升， 但仍存在以下一些不足之处：

（3.1）目标尺度方面的限制： 尽管 YOLOv3 引入了三个不同尺度的特征图来处理不同大小的目标， 但对于极小目标和极大目标仍然存在一定的限制。较小的目标可能会因为特征图分辨率较低而难以准确检测，而较大的目标可能会被分割成多个边界框。

（3.2）模型复杂度和计算成本： YOLOv3 相对于 YOLOv2 来说更加复杂， 具有更多的卷积层和参数。这导致 YOLOv3 的模型大小更大， 需要更多的计算资源进行训练和推理。因此，它可能不适用于计算资源有限的嵌入式设备或需要实时性能的应用。

（3.3）定位精度的提升有限： 尽管 YOLOv3 在定位精度方面相对于 YOLOv2 有所改进， 但与其他一些目标检测模型相比， 仍然存在一定的不足。特别是对于小目标和高密度目标的定位， YOLOv3 可能存在一些困难，并且可能会产生较大的定位误差。

（3.4）目标分类的困难： YOLOv3 使用了交叉熵损失函数进行目标分类， 但在存在类别不平衡或者类别间难以区分的情况下， 分类性能可能受到影响。模型可能会倾向于优化常见类别的预测，而对于罕见类别的识别可能会较差。

（3.5）复杂背景下的目标检测： YOLOv3 在复杂背景下的目标检测能力相对较弱。当目标与背景色相近或者存在复杂纹理时，模型可能会出现误检测或漏检测的情况。

（4） YOLOv4

YOLOv4 相对于 YOLOv3 在输入端引入了 Mosaic 数据增强 [26]，在骨干网络中使用不包含全连接层的 CSPDarknet53 模块结构， 并使用特征金字塔（Feature Pyramid Networks ， FPN） 和路径聚合（Path Aggregation Network ， PAN） [52] 结构进行信息传递与融合。训练方面， 损失函数由 YOLOv3 的二分交叉熵损失函数变为 CIoU_Loss，对损失函数进行优化有助于网络模型在规定的周期内进行局部寻优。激活函数方面， 将 YOLOv3 使用的 Leaky ReLU 激活函数改为 Mish 激活函数。检测方面， 预测框筛选策略由传统的 NMS 变为 DIoU_NMS，不仅考虑了 IoU 的数值， 还考虑到了预测框中心点之间的距离关系。

尽管 YOLOv4 在 YOLO 系列中引入了一些重要的改进和性能提升， 但仍存在以下一些不足之处：

（4.1）高计算资源需求： YOLOv4 相较于之前的版本更复杂、更深层的网络结构，需要更多的计算资源进行训练和推理。这导致 YOLOv4 在计算资源有限的情况下可能不太适用，尤其是对于嵌入式设备或需要实时性能的应用。

（4.2）定位精度的提升有限： 虽然 YOLOv4 在定位精度方面相对于之前的版本有所改进， 但与一些其他目标检测模型相比， 仍然存在一定的不足。特别是对于小目标和高密度目标的定位， YOLOv4 可能仍然存在一些困难，并可能产生较大的定位误差。

（4.3）训练和调优的复杂性： YOLOv4 的网络结构和超参数较多， 对于训练和调优过程的要求较高。合理地设置学习率、数据增强方法、正则化等参数对于获得良好的性能是至关重要的。因此，使用 YOLOv4 进行训练和优化可能需要更多的经验和时间。

（4.4）对类别不平衡的敏感性： YOLOv4 在目标分类时仍然使用交叉熵损失函数，对于存在类别不平衡或类别间难以区分的情况， 分类性能可能受到影响。模型可能会倾向于优化常见类别的预测，而对于罕见类别的识别可能会较差。

（4.5）鲁棒性和复杂场景下的性能： YOLOv4 在复杂背景、遮挡和高噪声环境下的目标检测能力相对较弱。当目标与背景颜色相近或存在复杂纹理时， 模型可能会出现误检测或漏检测的情况。

（5） YOLOv5

YOLOv5 包含 4 个模型， 为别 YOLOv5s 、YOLOv5m 、YOLOv5l 和 YOLOv5x，其网络结构相同， 但要根据任务对检测精度和检测时间的需求更改网络的深度系数 depth_multiple 和宽度系数 width_multiple 来进行调整。如下图 2.2 所示， YOLOv5 首先会对送入网络的待检图像进行自适应图片缩放， 得到输入像素大小为 608×608 的图像， 然后骨干网络使用跨阶段局部网络结构(Cross Stage Partial Network ，CSP）进行特征提取， 颈部网络同样使用 FPN+PAN 的网络结构来结合提取到的特征， 最后由检测头预测出目标的位置和类别信息。

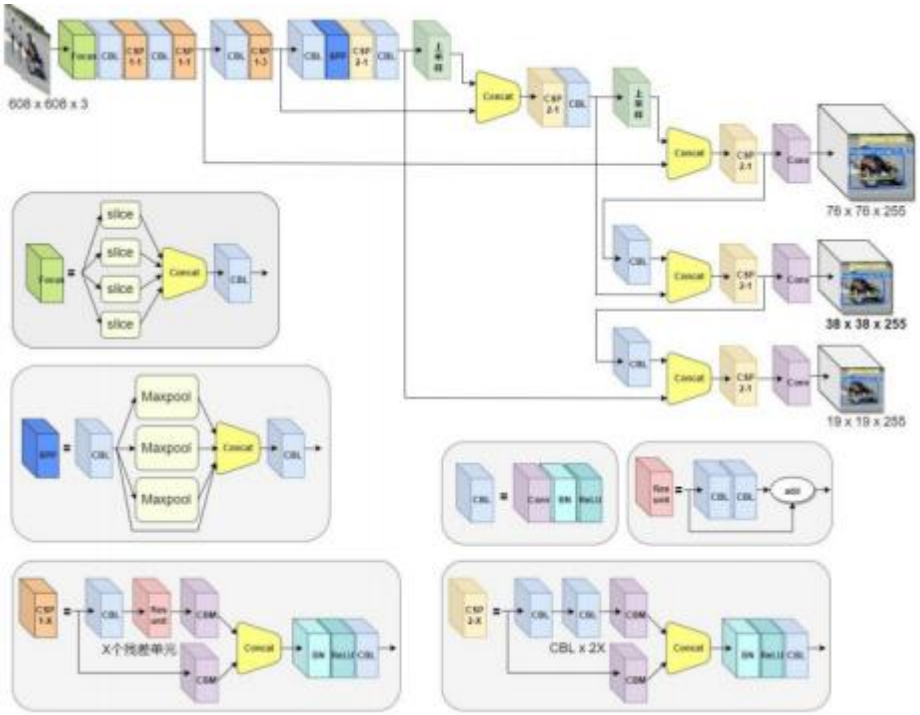

图3 YOLOv5s 网络结构

其中， Focus 结构由 4 个切片模块构成。 CBL （Convolution Batch Normalization Leaky ReLU unit）包含卷积层、批量归一化层(Batch Normalization ，BN） 和 Leaky ReLU 激活函数。残差单元(Res unit）通过 2 个 CBL 进行残差操作构建而成。 CSP1-X 结构用于骨干网路， 由 1 个 CBL、X 个残差单元和 2 个卷积层构成。 CSP2-X 结构用于颈部网络， 由 2X 个 CBL 和 2 个卷积层构成。空间金字塔池化层(Spatial Pyramid Pooling ，SPP）由 1 个 CBL 和 3 个最大池化层构成。 Concat 负责组合深层和浅层的

通道信息。

YOLOv5 网络还对输入的图片同样采取了 Mosaic 数据增强， 这种方法不仅扩充了数据集， 也变相提高了批量大小(Batch Size)，能够有效提升训练效果。图片的降采样在骨干网络中完成， 并在 3 个不同尺度的特征图层中检测目标， 其中 3 个不同尺度的特征图层由最终的 3 次降采样得到， 像素尺寸大的特征图提供目标的位置信息， 像素尺寸小的特征图则提供深层的语义信息。 YOLOv5 对前人的工作做了总结和整理并进行了许多创新， 获得了当前单阶段目标检测算法的最高精度。其使用 PyTorch 框架也利于二次开发，因此选取 YOLOv5 作为本文后续模型改造的研究基础。

2.3 钢材表面缺陷检测中存在的问题

通过分析以上 YOLO 系列算法在 COCO 数据集中的对比结果和模型缺点可以发现， 随着算法研究的进一步发展， YOLO 系列目标检测算法在减少检测用时的同时， 检测精度也得到了提升。由于简化了网络结构， 网络训练也更加节省时间和计算成本。但是对于特定情景，如进行钢材表面缺陷检测任务时会存在一些问题。

疯裂， 夹杂， 斑块， 点蚀表面， 轧入鳞片， 划痕是钢铁表面的主要缺陷， 相关介绍将在第三章 NEU-DET 数据集介绍时具体阐述。在使用 YOLOv5 对 NEU-DET 数据集进行检测时候，存在问题如下：

(1)某些类别缺陷目标的 AP 值过低

下图 2.3 是使用原始 YOLOv5 在 NEU-DET 数据集上进行训练后得到的 PR 曲线图， 从中可以看到裂纹(crazing) 类型缺陷的 AP 值明显低于其他 5 种类型缺陷， 导致 mAP 值被拉低， 影响了网络模型整体的检测效果。原因是裂纹类型的缺陷同类缺陷间差距大，不同类缺陷间差距小，网络容易流失该类型缺陷的关键特征信息。

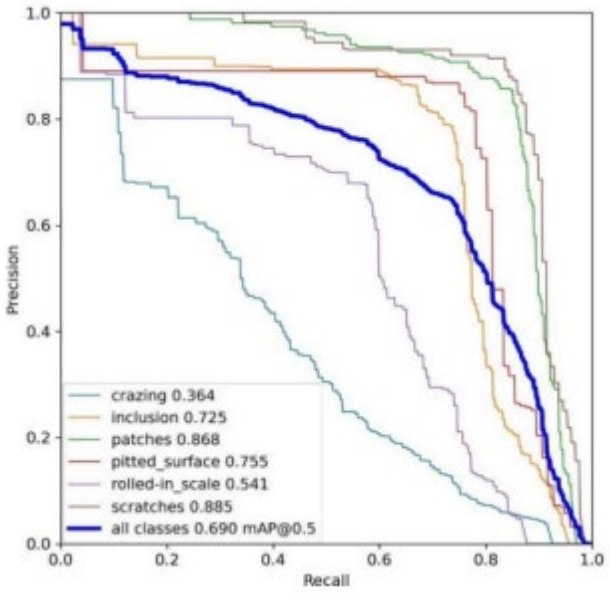

图4 原始 YOLOv5 在 NEU-DET 数据集上的 PR 曲线

（2）在复杂的情况下容易出现漏检和误检的现象钢材表面上通常包含了许多小目标、遮挡目标和密集目标的缺陷， 在实际工业条件下环境则会更为复杂。如下图 2.4 所示， 在使用原始 YOLOv5s 训练得到的模型对钢材表面缺陷进行检测时发现， 对于识别难度较大的一些目标， 容易出现漏检和误检的现象。下图 2.4（a）中红色的标记的是未被识别出的缺陷， 2.4 (b) 中粉色预测框是被误检的缺陷。由此可见， 原 YOLOv5 模型在处理复杂情况下的缺陷时性能有所欠缺， 这是因为缺陷图像多为灰度图像， 本身特征提取就不够充分， 加之工业中环境中拍摄的数据集噪点过多， 对于网络学习到关键的信息也有一定程度的干扰。

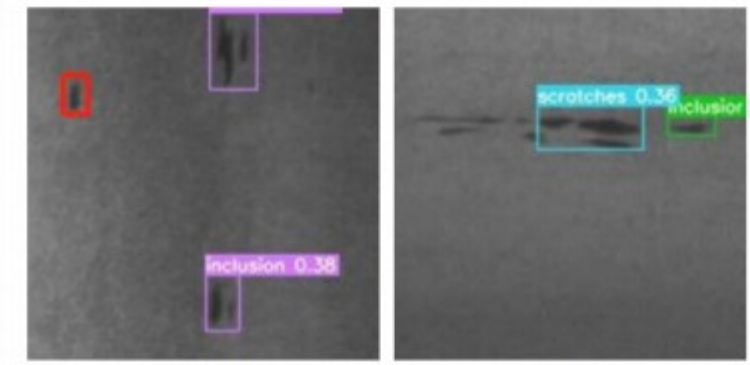

图5 漏检和误检现象

2.4 MSFT-YOLO 模型的提出

通过换用层数更深的卷积神经网络通常可以提高模型对于目标特征提取的能力， 但对于钢材表面缺陷检测任务来说并非如此， 过于复杂的模型可能会导致本就不明显的缺陷特征丢失，同时也会增加运算成本。

针对图像背景干扰大、缺陷类别易混淆、缺陷尺度变化大、小缺陷检测效果差的工业场景需求，我们使用了 MSFT-YOLO 模型。通过在主干和检测头中加入基于 Transformer 设计的 TRANS 模块，使特征与全局信息相结合。通过组合多尺度特征融合结构对不同尺度的特征进行融合， 增强了检测器对不同尺度目标的动态调整。为了进一步提高 MSFT-YOLO 的性能， 我们还引入了大量有效的策略， 如数据扩充和多步训练方法， 在边端对图片进行预处理去噪和增强。在 NEU-DET 数据集上的测试结果表明， MSPF-YOLO 能够实现实时检测， MSFT-YOLO 的平均检测准确率为 69.3，较基线模型（YOLOv 5）提高约 7%，较 Faster R-CNN 提高约 18%，具有一定的优势和启发性。

MSFT-YOLO 的总体原理图如下图 2.5 所示， 主要包括三个部分： 骨干网部分、特征增强部分和预测部分。在第一部分主干中， 我们没有使用 YOLOv5 原有的卷积层，而是主要使用了 TRANS 结构， 通过将其组装到 CSPDarknet 中来扩展卷积的接收域。TRANS 为检测提供了具有全局信息的多层次特征， 增强了 MSFT-YOLO 对钢铁表面背景特征的识别能力。在网络的颈部， 用简单有效的 BiFPN 结构代替 PANet 对骨干网

的多层次特征组合进行加权， 并将 TRANS 模块集成到预测头中， 替代原有的预测头，挖掘了 YOLOv5 自注意的预测潜力， 能够在高密度场景中准确定位目标， 并能处理目标的大尺度变化，非常适合钢铁缺陷的检测。

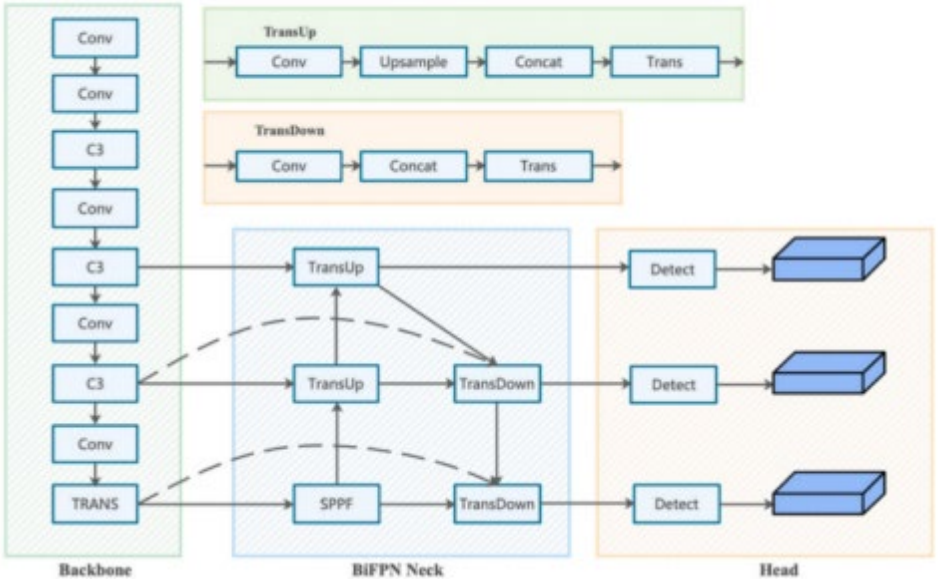

图6 MSFT-YOLO 原理图

TRANS 结构的实现可以基于以下步骤：

特征分组： 将输入的特征分成多个特征向量， 每个特征向量对应一个空间位置或通道。

位置编码： 为每个特征向量引入位置编码， 以表示其在整个特征图中的位置信息。位置编码可以使用固定的方式(如正弦/余弦函数)或通过学习得到。

自注意力计算： 对于每个特征向量， 使用自注意力机制计算其与其他特征向量之间的相关性。自注意力机制可以通过计算特征向量之间的相似度(如点积注意力) 并归一化得到注意力权重。这样每个特征向量都会获得与其他特征向量的关联权重。

特征融合： 使用注意力权重对特征向量进行加权融合， 得到融合后的特征表示。融合后的特征表示综合考虑了全局范围内的相关信息。

构特征图：将融合后的特征表示重新组合成与原始特征图相同的形状。

MSFT-YOLO 模型中， TRANS 结构被应用于主干网部分， 以扩展卷积的接收域。通过将 TRANS 结构嵌入到 CSPDarknet 等卷积网络中， 可以增加特征之间的交互和全局上下文的建模能力。这有助于提高钢铁缺陷检测器对背景特征的识别能力， 从而增强整个检测系统的性能。

在 MSFT-YOLO 模型中，采用了 BiFPN (Bi-Directional Feature Pyramid Network) 结构来替代 PANet (Path Aggregation Network)，以对骨干网的多层次特征进行加权和融合。

传统的目标检测算法中， 通常使用特征金字塔结构(例如 FPN) 来融合多个尺度

的特征图， 以便在不同尺度上进行目标检测。 PANet 是一种常见的特征金字塔结构， 用于特征的上下文聚合。然而， PANet 在处理特征信息时存在信息传递不完整的问题。

为了克服 PANet 的局限性， MSFT-YOLO 采用了 BiFPN 结构来代替 PANet。BiFPN 是一种双向的特征金字塔网络， 它通过从不同尺度上进行自上而下和自下而上的信息传递来增强特征的表示能力。

BiFPN 结构由以下步骤组成：

自上而下的特征传递： 从最高分辨率的特征开始， 通过上采样操作将特征图的分辨率逐渐减小，同时通过连接和融合操作与低分辨率的特征图进行信息传递。

自下而上的特征传递： 从最低分辨率的特征开始， 通过下采样操作将特征图的分辨率逐渐增加，同时通过连接和融合操作与高分辨率的特征图进行信息传递。

特征融合： 在自上而下和自下而上的信息传递过程中， 使用注意力机制或卷积操作对特征进行加权融合，以综合利用不同尺度特征的表征能力。

通过使用 BiFPN 结构， MSFT-YOLO 能够更好地捕捉不同尺度特征的语义信息，并进行加权融合， 以提高目标检测的准确性和鲁棒性。这种特征金字塔网络的设计有助于 MSFT-YOLO 在处理具有不同尺度的目标和复杂背景的工业场景中表现更好。

第 3 章　基于 MSFT-YOLO 的钢材表面缺陷检测

在确定了 YOLOv5 的改进方案后， 我们通过代码实现了 MSFT-YOLO 的模型搭建， 部署在个人服务器上， 并在 NEU-DET 数据集上进行验证， 试图得到在钢材表面缺陷检测任务中性能表现最优的网络模型。下面我们将通过更详尽的实验和分析来对第 2 章的结论进行补充。下面，我们首先将对数据集、实验环境和整体实验流程进行介绍。

3.1　钢材表面缺陷数据集

3.1.1　数据集介绍

NEU-DET（Northeastern University-Detect）数据集是由东北大学发布的钢材表面缺陷数据库中子数据集， 收集了钢带的 6 种典型表面缺陷， 即轧制氧化皮(RS)，斑块(Pa)，开裂(Cr)，点蚀表面(PS)，内含物(In) 和划痕(Sc) 对于缺陷检测任务， 数据集提供了注释， 标明了每个图像中缺陷的类别和位置。该数据集包括 1800 张灰度图像， 每种缺陷 300 张图像， 每种缺陷的典型样本如图 3.1.1 所示。在 NEU-DET 数据集中， 缺陷尺寸大小和分布是不均匀的， 这就导致在训练过程中对于一些小目标缺陷的特征提取总是不太充分。在网络训练之前会将输入图像缩放到指定尺寸， 经过放缩后的缺陷尺寸也可能会接近小目标大小。每种缺陷的详细介绍如表 3.1.1 所示。

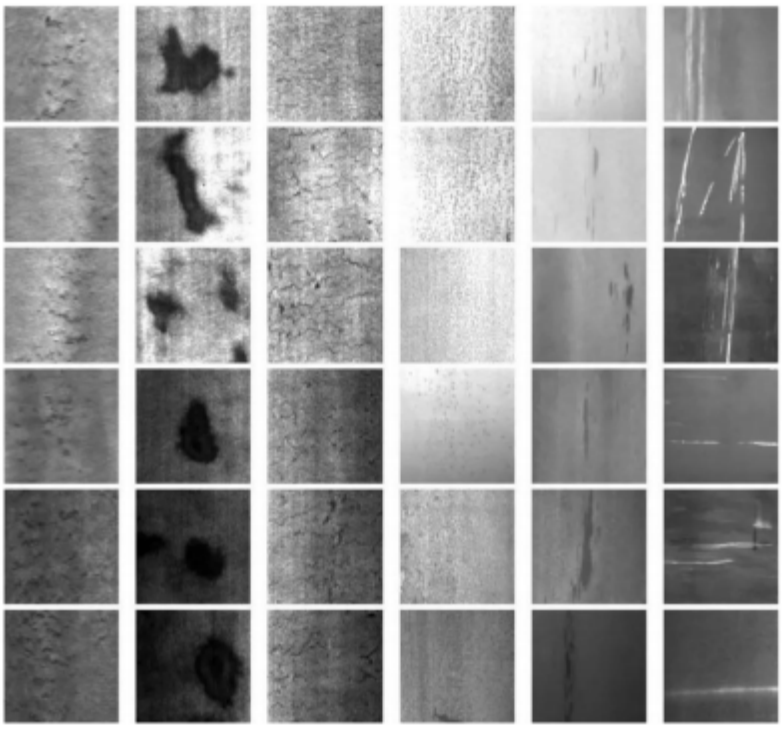

图7 NEU—DET 数据集经典缺陷典型样本

将数据集中的图像用 LabelImg 工具软件标注并以 YOLOv5 数据的格式分类存储。数据集中共 1800 张钢材表面缺陷图像， 将其中 1260 张作为训练集， 360 张作为验证集，

缺陷类型	外观特征	产生原因	工业标准
rolled-in scale 轧制氧化皮	一般存在于在带钢的上下表面呈珪点、鱼鳞、条或块状, 并且疏松易脱落。	钢材本身产生氧化铁皮或轧制时带入到带钢表面所形成。	表面通常不允许存在氧化铁皮或允许存在薄层氧化铁皮。
patches 斑块	存在于钢材边缘或中部的锈蚀和乳化液斑, 呈不规则的点状、块状和条片状的不规则斑块, 颜色通常为黄色或黑色。	钢材与大气中的物质反应生成黄色的氯化亚铁呈现。退火阶段含有铁元素的化合物被还原成黑色铁粉。	根据钢板产品洁净度要求, 控制表面斑迹的数量。
crazing 开裂	带钢表面存在的不连续的裂纹。呈树枝状向外发散形成圆形或椭圆形。	当晶界被低熔点相削弱, 如热脆时进行变形会出现开裂。	钢带表面不允许存在开裂缺陷。
pitted- surface 点蚀表面	不均匀地分布于带钢的上下表面, 呈连续或局部的橘皮状粗糙表面。	氧化铁皮附着在高温磨损的车辊上时形成点蚀。众多点蚀形成点蚀表面。	不同用途的标准和使用要求不同。
inclusion 杂质	存在于带钢破裂区域的点状、块状或长条状的不规则杂质, 颜色没有统一标准。	车制过程中残留或锻造加热阶段非金属物质脱落。	不同用途的标准和使用要求不同
Scratches 划痕	区域大小、深度不规则的机械性缺陷, 通常为条形。呈黑色、灰白色或金属光泽。	由于钢板与机械部件相对运动出现摩擦而产生。	不同用途的标准和使用要求不同。

表 3.1.1 数据集每种缺陷详细说明

剩余 180 张作为测试集， 训练集、验证集、测试集的比例大致为 7:2:1，以这样的比例分配数据集可以保证更好的训练效果。

3.1.2 数据增强方案

在将数据集送入云端模型进行训练之前， 我们在边端利用 PC 电脑对工业摄像头原始 NEU-DET 数据图像进行了预处理。经过对比了多种图像预处理方法后选取了效果较好的线性对比度增强方法来对 NEU-DET 数据集进行预处理， 该方法适用于图像的灰度值范围较小，导致部分输入图像整体较暗或较亮的情况。

（1）去噪：

中值滤波：使用中值滤波器对图像进行平滑处理，去除椒盐噪声和斑点噪声。

双边滤波： 结合了空间距离和灰度相似性的滤波方法， 能够保留边缘信息的同时去除噪声。增强：

（2）增强：

对比度增强： 使用对比度拉伸或直方图拉伸等方法增强图像的对比度， 突出缺陷区域。

直方图均衡化： 通过重新分布图像的像素值来增强图像的对比度， 使细节更加清晰。

这套方案的主要目标是去除图像中的噪声并增强缺陷的对比度， 以便更好地检测和分析钢材表面的缺陷。中值滤波和双边滤波是常用的去噪方法， 选择合适的滤波器。对比度增强和直方图均衡化可以突出缺陷区域， 使其更加明显， 便于后续的检测算法处理。

下面是代码示例

```
import cv2
import numpy as np

#读取原始图像
image = cv2.imread("original_image.jpg", cv2.IMREAD_GRAYSCALE)

#去噪处理
denoised_image = cv2.medianBlur(image, 5)

#对比度增强
enhanced_image = cv2.equalizeHist(denoised_image)

#图像修复
#可以根据具体需求使用图像分割、形态学操作等方法进行空洞填充

#显示处理结果
cv2.imshow("Original Image", image)
```

```
cv2 .imshow ( " Den ois e d Image " , denoised_image)
cv2 .imshow ( " Enhanced Image " , enhanced_image)
cv2 .waitKey (0)
cv2 .destroyAllWindows ()
```

在这个示例中， 我们使用了 Python 的 OpenCV 库进行图像处理操作。首先， 我们使用 cv2.imread 函数读取原始图像， 将其转换为灰度图像。然后， 使用 cv2.medianBlur 函数对图像进行中值滤波处理， 去除噪声。接下来， 使用 cv2.equalizeHist 函数对图像进行直方图均衡化， 增强图像的对比度。若工业设备摄像功能不好， 你甚至还可以在图像修复部分使用适当的图像分割或形态学操作方法来填补缺陷区域， 或者结合其他 Python 图像处理库和算法，如 NumPy 、PIL 等，来进一步优化和扩展图像处理方案。

3.2 训练环境

基于 MSFT-YOLOv5 的钢材表面缺陷检测研究使用的框架为 PyTorch。具体训练环境配置见表 3.2.1.

配置环境	名称
深度学习框架	Pytorch2.0.1
编译器	python3.9.0
GPU	NVIDIA GeForce RTX 2060
加速模块	cuda11.8

表 3.2.1 训练环境配置

3.3 改进后网络训练

为了更快速完成训练同时保证训练效果， YOLOv5 在网络模型的训练过程中采取了一些训练技巧。如训练学习率预热、带重启的随机梯度下降、余弦退火学习率调整和自动混合精度训练等。

(1) 训练学习率预热在训练开始阶段， 网络模型的权重由随机初始化得到， 训练情况会因为学习率设置偏大而不稳定。学习率预热策略将初始几个周期的学习率设置较小， 在此情况下网络可以沿着稳定的方向对模型进行训练， 待训练稳定再调整至预设的学习率。经过实验验证，该策略能使训练平稳进行并且加速收敛，提升训练效果。

(2) 带重启的随机梯度下降在网络模型训练过程中观察发现， 目标损失函数在规定的训练周期内存在多个局部最小值， 带重启的随机梯度下降方法可以调整学习率使 Loss

值沿训练周期内的全局最小值下降，避免将局部最小值视为最优解。

（3）余弦退火学习率调整带重启的随机梯度下降方法为了避免 Loss 的局部最小值而提高了学习率， 然后则会通过余弦退火的方法来减小学习率， 以保证卷积神经网络训练的可控性，稳定运行的训练同样有助于网络模型收敛并且提升训练效果。

（4）混合精度训练混合精度代表数据类型中有不止一种精度的张量， 在 PyTorch 深度学习框架的自动混合精度训练模块里有两种张量， 后者占用空间少、响应迅速， 便于 CUDA 中的核心模块对其进行优化加速。在优化 GPU 内存的同时， 降低了训练时长。但其同样存在一些缺点， 例如部分 16 位精度以下的分辨率梯度信息会因为过窄的取值范围以及过大的允许误差而损失。

3.4 训练结果

在 NEU-DET 数据集上的训练结果如表 3.4.1、表 3.4.2 所示.

	Precision	Recall	mAP50
All	0.628	0.672	0.692
Crazing	0.472	0.087	0.318
Inclusion	0.555	0.841	0.722
Patches	0.735	0.949	0.919
Pitted_ surface	0.788	0.724	0.754
Rolled- in_ scale	0.600	0.598	0.593
scratches	0.615	0.833	0.848

表 3.4.1 改进前 YOLOv5 模型训练结果

从表中可以看到改进后的模型在测试集上的表现各有提升， 原模型的平均检测准确率为 62.8%，改进后的模型平均检测准确率为 69.3%，提升 6.5%。这说明采用的改进策略是有用的。

	Precision	Recall	mAP50
All	0.693	0.934	0.702
Crazing	0.831	0.159	0.389
Inclusion	0.614	0.855	0.811
Patches	0.745	0.917	0.913
Pitted_ surface	0.655	0.665	0.688
Rolled- in_ scale	0.552	0.551	0.567
scratches	0.762	0.754	0.844

表 3.4.2 改进后 MSFT-YOLOv5 模型训练结果

第 4 章　云边端架构搭建及网络传输协议

在上一章节中， 重点讨论了基于 MSFT-YOLO 模型， 实现了对钢铁表面缺陷的检测的相关实验， 本章则根据实际工业中的检测需求围绕轻量化， 模型部署进行了一系列研究。针对大型工业互联网平台开发的大环境需要， 整体架构采用工业互联网的设计思路， 即采用云计算、物联网、大数据等技术相结合的方案， 实现数据的采集、传输、存储和分析。在我们的模拟设想中， 钢铁工厂设备通过传感器将图片数据通过物联网传输协议传输给在另一个地方的钢铁公司管理部门， 公司通过内部电脑， 或是公司服务器汇总数据， 并通过数字图像处理技术实现图像去噪， 增强， 将预处理后的图片上传到云端。通过网络传输协议， 我们可以实现图片的传输， 并且可以从云端自动获取返回的训练结果， 通过数据处理技术， 可以将处理结果存入公司数据库系统中， 便于进一步管理和分析，实现工厂，管理，算力的远程连接。

4.1　云边端架构

云边端架构是一种分布式计算架构， 将计算任务和资源在云端和边缘设备之间进行协同处理。该架构的目标是在云端和边缘设备之间实现计算和数据的有效协同， 以提供更高效的计算能力和低延迟的服务。下图 4.1 为云边端架构示例：

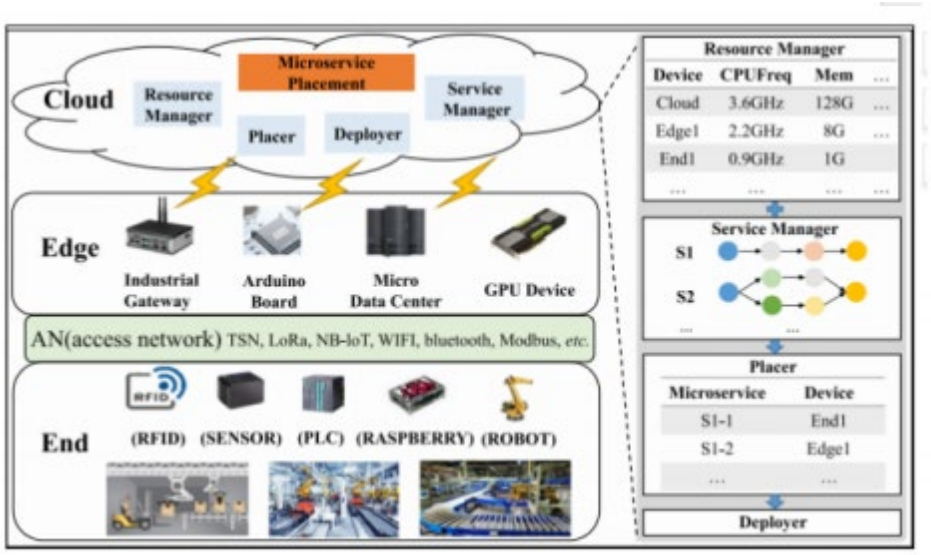

图8 云边端架构示意图

在云边端架构中，通常包含以下几个组件：

(1) 云端服务器：

云端服务器是具备高性能计算和存储能力的中心节点， 负责处理大规模的数据和复杂的计算任务。它可以提供强大的算力和存储能力， 并且能够扩展和管理边缘设备， 在本次模拟实验中，负责 MSFT—YOLO 模型的搭载。

(2)边缘设备：

边缘设备是位于用户或物联网终端附近的智能设备， 例如智能手机、传感器、摄像头等。边缘设备通常具备一定的计算和存储能力， 能够进行部分数据处理和简单的决策。它们负责采集、预处理和传输数据到云端服务器， 并接收来自云端的指令和结果。在本次实验中， 主要是位于公司的服务器或是电脑承担预处理和传输数据到云端服务器， 并接收来自云端的指令和结果，位于工厂的工业设备负责采集原始数据。

(3)网络连接：

云边端架构依赖于可靠的网络连接， 以保证云端和边缘设备之间的通信和数据传输。这可以包括有线网络(如 Ethernet)、无线网络(如 Wi-Fi、蜂窝网络) 或专用的边缘网络(如物联网专用网络)等。

(4)数据传输和处理：

边缘设备负责采集和传输数据到云端服务器， 同时云端服务器也可以将指令和计算结果传输回边缘设备。数据传输可以通过边缘网关或直接连接到云端进行， 具体取决于网络架构和应用需求。数据在云端和边缘设备之间进行处理和分析， 可以利用云端的强大计算资源和边缘设备的实时响应能力。

云边端架构的优势在于将计算和数据处理分布到更接近终端用户或设备的边缘， 减少了数据传输的延迟和网络带宽压力， 并提供更好的实时响应能力。同时， 云边端架构也能够充分利用云端的高性能计算和存储资源， 处理更复杂的任务和大规模数据分析。这种架构在物联网、智能城市、智能制造等领域具有广泛的应用前景。

4.2 云边端的通信——互联网传输协议

4.2.1 HTTP 传输协议

HTTP (HyperText Transfer Protocol，超文本传输协议) 是用于传输超媒体文档(如 HTML) 的应用层协议。它被设计用来通过网络传输数据， 包括文本、图像、音频、视频等。由于其简单且强大的特性，

HTTP 协议广泛应用于各种场景，包括工业互联网。

在工业互联网(Industrial Internet of Things, IIoT) 中， HTTP 协议常常用来在设备和云服务器之间进行数据传输。比如， 设备可以使用 HTTP 协议向服务器发送设备状态、传感器数据， 或接收来自服务器的指令。服务器可以使用 HTTP 协议接收设备的数据，或向设备发送指令。

HTTP 支持多种方法，包括 GET 、POST 、PUT 、DELETE 等，它们分别对应于不同的操作，例如获取数据、发送数据、更新数据和删除数据。

HTTP 请求和响应都由三部分组成： 起始行、头部和主体。起始行包含了方法、 URL 和 HTTP 版本； 头部包含了一些元数据， 如内容类型、内容长度等； 主体则包含了实际的数据。

4.2.2 MIME 类型

MIME （Multipurpose Internet Mail Extensions，多用途互联网邮件扩展）是一种用来描述和标记数据内容类型的互联网标准。在 HTTP 协议中， MIME 类型常常用在“Content-Type”和“Accept”头部，用来表示发送或接收的数据的类型。

MIME 类型是由两部分组成的： 大类和小类， 用“/”分隔。例如， 文本文件的 MIME 类型是”text/plain”，JPEG 图像的 MIME 类型是”image/jpeg”，JSON 数据的 MIME 类型是”application/json”。而在钢铁表面缺陷检测中，图片信息的传输必不可少。

在工业互联网中， MIME 类型可以帮助设备和服务器正确地处理数据。例如， 如果设备向服务器发送一个 JPEG 图像， 那么它可以在 HTTP 请求的“Content-Type”头部中使用”image/jpeg”，这样， 服务器就知道它需要以图像的方式来处理接收到的数据。

同时， 设备也可以在 HTTP 请求的“Accept”头部中使用特定的 MIME 类型， 告诉服务器它希望接收哪种类型的数据。例如， 如果设备只能处理 JSON 数据， 那么它可以发送一个带有”Accept: application/json” 头部的 HTTP 请求， 这样， 服务器就知道它需要返回 JSON 数据。利用这样的机制， 我们可以决定返回的数据类型， 方便边端设备接收相应的数据，存入公司相应的数据库系统中，进行进一步管理和分析。

4.2.3 Base64 编码在工业互联网中的应用

在工业互联网（Industrial Internet of Things，IIoT）的环境中， 传感器和设备经常需要将各种类型的数据发送到云服务器。这些数据可能包括文本、数字、二进制数据，以及图片、音频、视频等。然而， 网络传输协议通常设计为最好处理文本数据， 因此对于非文本的二进制数据， 如图片、音频和视频等， 直接传输可能会出现问题。这时候，Base64 编码就能派上用场。 Base64 是一种基于 64 个可打印字符来表示二进制数据的表示方法。

通过 Base64 编码， 可以将二进制数据转换为由 A-Z、a-z、0-9、+ 和/这 64 个字符组成的字符串，这样，就可以在只支持文本数据的网络协议中传输二进制数据。

例如， 一个工业相机可能需要将拍摄的图片发送到云服务器。由于图片是二进制数据， 所以相机可以首先将图片数据进行 Base64 编码， 然后再通过 HTTP 或 MQTT 等协议发送到服务器。

4.2.4 Base64 编码的基本原理

Base64 编码的基本原理很简单。首先， 将每三个字节的二进制数据分成四组， 每组六位。然后， 将这四组数据分别转换为 0-63 的十进制数。最后， 将这四个十进制数映射到 Base64 的 64 个字符中，得到四个字符的编码结果。

如果原始数据的字节数不是 3 的倍数， 那么在编码时可以使用填充字符（通常是”=”）来补齐。例如， 如果原始数据只有两个字节， 那么编码结果将有三个字符和一个填充字符；如果原始数据只有一个字节，那么编码结果将有两个字符和两个填充字符。

4.3 云边端通信示例

4.3.1 图片上传示例

工业设备(相机）拍摄照片， 经过自动预处理之后， 通过 http 请求将数据发送到云端服务器， 服务器时刻监听服务， 当收到数据后， 使用模型处理接收到的图片， 并把数据更新到数据库中。同时，服务器可以接收监控请求信息，监控请求返回统计数据。

下面是一段传输示例：

```
POST /api/uploadImage HTTP/1 .1
Host : center-server .com/image
Content-Type : multipart/form-data ; boundary=----
WebKitFormBoundary7MA4YWxkTrZu0gW
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " timestamp "
2023-06-04T08 :00:00Z
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " deviceName "
Factory-Zone1-Device-Sensor
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " deviceUUID "
1f346544 -027f-11ee-be56-0242ac120002
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " imageType "
jpg
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " imageFile " ; filename= "
   image . jpg "
Content-Type : image/jpeg
/9j/4AAQSkZJRgABAQEAYABgAAD/2
   wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRof
Hh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2
   wBDAQkJCQwLDBgNDRgyIRwh
MjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjI

yMjL/wAAR
CAABAAEDASIAAhEBAxEB/8
   QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAA
AgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fA
```

```
kM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaG

lqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDx

MXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8
   QAHwEAAwEBAQEB
AQEBAQAAAAAAAAECAwQFBgcICQoL/8
   QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxE
EBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKj

U2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKk

pOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk

5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwD3+iigD//2Q==
------WebKitFormBoundary7MA4YWxkTrZu0gW --
```

上述示例中：

POST /api/uploadImage HTTP/1.1 开始一个新的 POST 请求， 向服务器发送图片。

Host: cloud-server.com 指定了服务器的域名。

Content- Type: multipart/ form- data; boundary= — -

WebKitFormBoundary7 MA4 YWxkTrZu0 gW

指定了内容类型为 multipart/form-data ，并设置了分隔符。

每个 ——WebKitFormBoundary7MA4YWxkTrZu0gW 都是一个边界， 表示一个新的数据部分的开始。

Content-Disposition: form-data; name=”timestamp” 等行定义了表单数据的名字，例如时间戳、设备名、设备编号、图片类型等。

Base64-encoded image data 是 Base64 编码的图片数据。

请求结束的标志是 ——WebKitFormBoundary7MA4YWxkTrZu0gW– ，注意这个边界和其他的边界有所不同，它的后面多了两个破折号(–)。

4.3.2 服务器处理结果 response

向服务器发出传输后，需要云端调用部署的检测系统，

```
[Information] -14653- 2023-06-06  Request method : POST ,
  Request URI : /api/uploadImage , Content-Type : multipart/form-
```

```
  data ;
[Information] -14653- 2023-06-06  Device info received :
  Factory-Zone1-Device-Sensor , UUID : 1f346544 -027f-11ee-be56
  -0242ac120002
[Information] -14653- 2023-06-06  Image Type : jpg , File
  received : image .jpg
[Information] -14653- 2023-06-06  Model processing started
[Information] -14653- 2023-06-06  Model processing completed
  successfully
[Information] -14653- 2023-06-06  Result image generated :
  result .jpg
[Information] -14653- 2023-06-06  Database update started
[Information] -14663- 2023-06-06  Database updated with new
  image and processing results successfully
[Information] -14663- 2023-06-06  Response method : POST ,
  Response URI : /api/uploadImage , Content-Type : multipart/form
  -data ;
[Information] -14663- 2023-06-06  Response message : Image
  processed successfully
[Information] -14663- 2023-06-06  End of request processing
```

下面展示服务器处理结果：

```
HTTP/1 .1 200 OK
Server : IndustrialCloudServer/1 .0
Date : Mon , 04 Jun 2023 09:01:15 GMT
Content-Type : multipart/form-data ; boundary=----
WebKitFormBoundary7MA4YWxkTrZu0gW
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " t im es t amp "
2023-06-04T09 :01:15Z
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " deviceName "
Factory-Zone1-Device-Sensor
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " deviceUUID "
1f346544 -027f-11ee-be56-0242ac120002
------WebKitFormBoundary7MA4YWxkTrZu0gW
```

```
Content-Disposition : form-data ; name= " imageType "
jpg
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition : form-data ; name= " resultImage" ; filename= "
   result.jpg"
Content-Type : image/jpeg
/9j/4AAQSkZJRgABAQEAYABgAAD/4QBARXhpZgAASUkqAAgAAAABAGmHBAABA
AAAGgAAAAAAAACAAKgCQABAAAAAQAAAAOgCQABAAAAAQAAAAAAAD/2wBDAA
MCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYIDAoMDAsKCwsNDhIQDQ4RDgsLEBY
QERMUFRUVDA8XGBYUGBIUFRT/2wBDAQMEBAUEBQkFBQkUDQsNFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBT/wAARC
AABAAEDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQ
oL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaE
II0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVW
V1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipq
rKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+P
n6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQ
DBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLR
ChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpa
nN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6ws
PExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxE
APwD8qqKKAP/2Q==------WebKitFormBoundary7MA4YWxkTrZu0gWConte
nt-Disposition : form-data ; name= " result"
Image processed successfully
------WebKitFormBoundary7MA4YWxkTrZu0gW --
```

其中：

HTTP/1.1 200 OK：这是 HTTP 响应的起始行， 表示 HTTP 版本为 1.1，响应的状态码为 200，状态码的文本解释为 OK。

Server: IndustrialCloudServer/1.0 ：此行表示服务器的名称和版本。

Date: Mon, 04 Jun 2023 09:01:15 GMT ：此行表示此响应消息的时间戳。

Content- Type: multipart/ form- data; boundary= — -

WebKitFormBoundary7MA4YWxkTrZu0gW ：此行定义了 HTTP 响应的内容类型是

multipart/form data，并且设定了数据分隔符。

后续的 ——WebKitFormBoundary7MA4YWxkTrZu0gW 表示每个数据部分的起始。

Content-Disposition: form-data; name=”timestamp” 等行定义了每个部分的名字，

如时间戳、设备名称、设备编号、图片类型和处理后的图片。

‘Base64-encoded image data 是处理后的图片数据，该数据使用 Base64 编码。

Content-Disposition: form-data; name=”result” 定义了处理结果的部分名。

Image processed successfully 表示处理结果的具体信息。

——WebKitFormBoundary7MA4YWxkTrZu0gW– 表示响应内容的结束， 注意它与其他边界的不同之处在于它后面多了两个破折号(–)。

4.3.3 管理者 API

```
GET /api/queryDeviceResults HTTP/1 .1
Host : cloud-server .com
Authority : Bearer
eyJhbGciOiJIUzI1NiJ9 .
   eyJleHAiOjE2ODU4NTA3MTguMzY0MzksImlhdCI6MTY4NTg0NzExOC4zNjQ
zODYsImlzcyI6ImNoYXJTZXJ2ZXJMYW1iZXJ0IiwibmJmIjoxNjg1ODQ3MTE4L
jM2NDM4OCwidXNlcl9
pZCI6MCwidXNlcm5hbWUiOiJBIn0 .
   rxLLNm2vbUXWEyE3_0aYpVujADCF0qjM_p_RFCOm8ZM
```

返回数据

```
HTTP/1 .1 200 OK
Server : IndustrialCloudServer/1 .0
Date : Mon , 04 Jun 2023 09:01:15 GMT
Content-Type : application/json
{
" code" : 0,
" data" : [
{
" timestamp" : " 2023-06-04T08:00:00Z" ,
" deviceName " : " Factory-Zone1-Device-Sensor" ,
" deviceUUID " : " 1f346544 -027f-11ee-be56 -0242ac120002" ,
" hasDefect" : false ,
" imageURL " : " http://cloud-server .com/images/processed/1.jpg"
},
{
" timestamp" : " 2023-06-04T08:05:00Z" ,
" deviceName" : " Factory-Zone2-Device-Sensor" ,
" deviceUUID " : " 1f346544 -027f-11ee-be56 -0242ac120003" ,
```

```
"hasDefect": true,
"imageURL": "http://cloud-server.com/images/processed/2.jpg"
}

]
}
```


HTTP/1.1 200 OK：这是 HTTP 响应的起始行， 它表示使用的 HTTP 版本为 1.1，响应状态码为 200，状态码的文本解释为 OK。

Server: IndustrialCloudServer/1.0：此行表示响应的服务器名称及其版本。

Date: Mon, 04 Jun 2023 09:01:15 GMT：此行表示响应生成的时间。

Content-Type: application/json：此行表明了响应数据的媒体类型是 JSON 格式。内的内容是响应的主体，包含的是 JSON 格式的数据。

”code”: 0, 表示响应的状态码， 0 通常表示请求处理成功。

”data”: [...] 这部分包含了请求的实际数据， 是一个数组。数组中每个元素都是一个对象， 每个对象包含了时间戳（timestamp）、设备名（deviceName）、设备编号（deviceUUID）、是否有缺陷（hasDefect）以及图片链接地址（imageURL）等信息。

第 5 章　总结与展望

5.1　总结

本文以 YOLOv5 目标检测算法为基础对钢材表面缺陷检测问题进行了相应研究。对原始 YOLOv5 进行了改进， 提高了算法检测的精度， 将实验得到的最优模型应用于钢材缺陷检测任务中。并将改进后的模型部署在云服务器上， 为钢材表面缺陷检测在工业中的应用提供了一种高效可行的云边端协同缺陷检测系统方案。本文主要工作如下：

（1）本章首先介绍了选择 YOLO 进行缺陷检测的原因， 并通过常见的目标检测评估指标构建了本文的评价体系。通过增加 Trans 模块对 YOLOv5 进行改进， 得到了 MSFT—YOLO 模型。

（2）基于改进 YOLOv5 的钢材表面缺陷检测实验。首先通过数据增强方法优化初始数据集， 提升训练效果并减少过拟合的风险， 增强算法模型的适应能力。然后搭建了钢材表面缺陷检测模型训练的实验环境。对于钢材表面缺陷检测任务， 对比了改进前后模型性能。 MSFT-YOLO 模型相较于 YOLOv5 平均检测准确率提高了 6.5%，提升了算法对于钢材表面缺陷检测的整体性能。

（3）针对当前工业环境中对于钢材表面缺陷检测的要求我们利用网络传输协议将工业摄像头拍摄的钢材表面图片传输到边缘设备， 对图片进行预处理之后上传到云端， 利用云端模型对图片进行检测，再返回检测结果。

5.2　展望

深度学习目标检测任务作为当前的研究热点， 许多优秀的方法和策略可供选择， 但由于研究时间和实验条件有限， 没能将更多的方法应用在钢材表面缺陷检测的研究与云边端架构缺陷检测系统中。本文研究中还可以完善的地方有：

（1）虽然本文提出的改进 YOLOv5 算法在检测精度有较大提升， 但对于钢材表面缺陷中复杂目标， 小目标和体遮挡目标的检测精度相较于两阶段目标检测算法网络仍然存在一定的提升空间。

（2）本文中网络模型均采用了全监督的学习方式， 后续可以尝试根据深度学习前沿的研究领域提出的半监督和无监督方式对网络模型进行训练。

（3）在云边端的架构过程中， 可以考虑使用多种传感器进行数据传输， 结合多模态技术，进一步提高缺陷检测的准确率。

（4）可以考虑建立云数据库系统， 对数据进行管理， 并实时反馈， 对工厂运行状况进行调整。

致谢

首先，我要向我们的指导教师表示最深的感激。他的专业知识和深思熟虑的指导使我们有机会在工业互联网领域进行深入研究。我记得每次在实验室，当我对工业互联网的设计和改进感到困惑时，他总是耐心地回答我的问题，分享他的独特视角，并指导我采用更严谨的研究态度。他的贡献对我的研究进展至关重要。

我也要感谢我的同学们，他们的热情讨论和深入探讨在我的研究过程中起到了关键的作用。我仍然记得那些深夜的讨论，我们一起针对数据集扩充策略、模型调整和工业应用场景进行思考，他们的思维火花给我带来了许多新的灵感和解决方案。

论文只是一个开始，学术研究的道路任重而道远。在未来的日子里，我将继续努力，希望做出更多有价值的研究，回馈所有给我帮助和支持的人。

再次感谢所有帮助过我、支持过我、鼓励过我的人。

附录 A 附录代码

```python
class TransformerLayer(nn.Module):
  # Transformer layer https://arxiv.org/abs/2010.11929 (LayerNorm layers removed for better performance)
  def __init__(self, c, num_heads):
    super().__init__()
    self.q = nn.Linear(c, c, bias=False)
    self.k = nn.Linear(c, c, bias=False)
    self.v = nn.Linear(c, c, bias=False)
    self.ma = nn.MultiheadAttention(embed_dim=c, num_heads=num_heads)
    self.fc1 = nn.Linear(c, c, bias=False)
    self.fc2 = nn.Linear(c, c, bias=False)

  def forward(self, x):
    x = self.ma(self.q(x), self.k(x), self.v(x))[0] + x
    x = self.fc2(self.fc1(x)) + x
    return x

```

基于Transformer的一个层，包括一个多头自注意力机制（MultiheadAttention）和两个全连接层（Linear）。输入通过自注意力机制和全连接层后，进行残差连接和层归一化。

```python
class TransformerBlock(nn.Module):
  # Vision Transformer https://arxiv.org/abs/2010.11929
  def __init__(self, c1, c2, num_heads, num_layers):
    super().__init__()
    self.conv = None
    if c1 != c2:
      self.conv = Conv(c1, c2)
    self.linear = nn.Linear(c2, c2)  # learnable position embedding
    self.tr = nn.Sequential(*[TransformerLayer(c2, num_heads) for _ in range(num_layers)])
    self.c2 = c2

  def forward(self, x):
```

```
        if self.conv is not None:
            x = self.conv(x)
        b, _, w, h = x.shape
        p = x.flatten(2).unsqueeze(0).transpose(0, 3).squeeze(3)
        return self.tr(p + self.linear(p)).unsqueeze(3).transpose(0, 3).reshape(b, self.c2, w, h)
```

```

一个包含多个`TransformerLayer`的模块。如果输入和输出的通道数不同，会先通过一个卷积层（Conv）调整通道数。然后，将输入展平并通过一个线性层（Linear）进行位置编码，最后通过`TransformerLayer`进行处理。

```python
class C3TR(C3):
    # C3 module with TransformerBlock()
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5):
        super().__init__(c1, c2, n, shortcut, g, e)
        c_ = int(c2 * e)
        self.m = TransformerBlock(c_, c_, 4, n)
```

新定义的TRANS模块：继承自`C3`模块，将`Bottleneck`模块替换为`TransformerBlock`模块。

```python
# parameters
nc: 80  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
```
```

```
  [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
  [-1, 3, C3, [128]],
  [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
  [-1, 9, C3, [256]],
  [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
  [-1, 9, C3, [512]],
  [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
  [-1, 1, SPP, [1024, [5, 9, 13]]],
  [-1, 3, C3TR, [1024, False]],  # 9  <-------- C3TR() Transformer module
 ]

# YOLOv5 head
head:
 [[-1, 1, Conv, [512, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 6], 1, Concat, [1]],  # cat backbone P4
  [-1, 3, C3, [512, False]],  # 13

  [-1, 1, Conv, [256, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 4], 1, Concat, [1]],  # cat backbone P3
  [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

  [-1, 1, Conv, [256, 3, 2]],
  [[-1, 14], 1, Concat, [1]],  # cat head P4
  [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

  [-1, 1, Conv, [512, 3, 2]],
  [[-1, 10], 1, Concat, [1]],  # cat head P5
  [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

  [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
 ]
```

自定义的模型配置文件：在backbone部分的最后添加了一个`C3TR`模块，这个模块是之前定义的，包含了Transformer模块。这样，模型在最后一层的特征提取阶段就会使用Transformer模块，这会帮助模型更好地理解图像的全局信息。