YOLOv6: 一种用于工业应用的单级目标检测框架

摘要

本文主要阐述 YOLOv6 的复现及优化工作。回顾 YOLO 系列的发展与工业应用挑战后，详细介绍复现中的改进：网络设计上依据不同规模采用不同组件，如小模型用 RepBlock、大模型用 CSPStackRep Block；标签分配经实验评估选用 TAL；损失函数根据任务选择 VariFocal、SIoU 或 CIoU 等；并采用工业实用改进措施，包括自蒸馏 [28] 等；量化部署中利用 RepOptimizer 和通道级蒸馏 [21]。复现实验优化数据加载和引入 DIoU [29]，在 COCO 2017 数据集上测试，模型性能有一点提升，展现出了轻量化模型的优势，但也暴露出轻量化精度不足、硬件推理性能未充分测试、复杂场景适应性欠佳等问题。

关键词： YOLOv6 复现；网络结构改进；损失函数

1 引言

在目标检测领域，YOLO 系列算法因其在速度和精度之间的良好平衡而成为工业应用中的热门选择。从 YOLOv1 [19] 到 YOLOv5 [9] 等版本的不断演进，为目标检测技术的发展奠定了坚实基础。然而，随着工业环境的日益复杂和多样化，对目标检测模型的要求也在不断提高。现实场景中，不同的应用场景对目标检测的速度和精度有着不同的侧重点，例如在一些实时性要求极高的场景（如自动驾驶中的目标检测），需要模型能够快速处理大量图像数据，同时保证较高的检测精度；而在一些对精度要求苛刻的场景，则更关注模型的准确性。

当前，YOLO 系列算法虽然广泛应用，但仍面临一些挑战。例如，在网络设计方面，如何更好地优化模型结构以适应不同规模的硬件设备和应用场景，是一个急需解决的问题。一些先进的网络设计技术（如 RepVGG 中的结构重参数化方法 [3]）在检测领域尚未得到充分利用，且简单的模型缩放策略在某些情况下可能导致计算成本过高或性能下降。在量化方面，基于重参数化的检测器的量化过程需要更加精细的处理，否则在训练和推理过程中由于配置的异构性容易导致性能退化。此外，先前的研究工作往往在部署方面关注较少，多侧重于在高性能硬件（如 V100 GPU）上进行性能比较 [25]，而在实际的工业服务环境中，低功耗 GPU（如 Tesla T4）更为常见，如何在这类硬件上实现高效部署并保证性能也是一个重要问题。同时，一些先进的领域特定策略（如标签分配和损失函数设计）在不同的网络架构下的有效性还需要进一步验证和优化。

在这样的背景下，本研究旨在对 YOLO 框架进行改进和优化，提出 YOLOv6 算法。通过充分吸收和整合最新的目标检测技术进展，包括网络设计、训练策略、测试技术、量化和优化方法等方面的创新，构建一系列适用于不同规模和应用场景的部署就绪网络，以更好地满足工业应用中对速度和精度的多样化需求。同时，通过广泛的实验验证和性能评估，为工

业界提供一种更高效、更准确的目标检测解决方案，推动目标检测技术在工业领域的进一步发展。

对比其他已有的 YOLO 系列算法，YOLOv6 在 NVIDIA Tesla T4 GPU 上以 1234 FPS 的吞吐量在 COCO 数据集 [15] 上的性能表现均高于其他主流 YOLO 系列算法，对比具体数据如图 1。

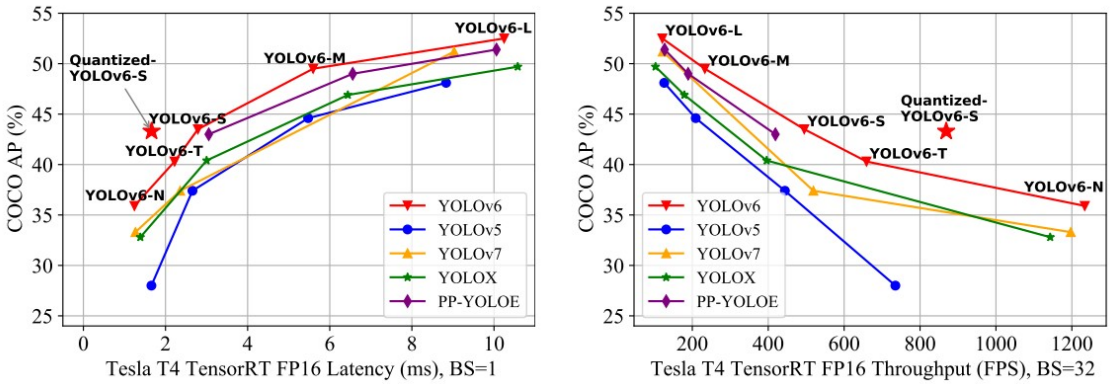

图 1. 延迟和吞吐量（批量大小为 32）与最先进的高效物体探测器的 AP 比较。

2 相关工作

目标检测领域中，相关工作有网络设计、标签分配、损失函数、量化与部署等，在本节中将具体介绍相关研究。

2.1 网络设计相关研究

以往研究表明，多分支网络在分类性能上常优于单路径网络，但会降低并行性、增加推理延迟；而单路径网络（如 VGG [22]）虽计算效率高，但模型容量扩展时，单路径网络的计算成本和参数数量会呈指数增长。RepVGG 提出结构重参数化方法 [3]，在训练时采用多分支拓扑，推理时转换为单路径架构，以平衡速度和精度。YOLOv6 受此启发，针对不同规模模型采用不同的网络结构设计策略，小模型利用单路径结构的高并行性，大模型则通过改进的 CSPStackRep Block 在保证精度的同时控制计算成本。

2.2 标签分配策略相关研究

标签分配在目标检测训练阶段负责为预定义锚点分配标签，此前已提出多种策略，从简单的基于 IoU 和内部真实框方法到更复杂的方案。例如，OTA 将标签分配视为最优传输问题 [5]，SimOTA 是其简化版本，曾被 YOLOv6 早期版本采用，但存在训练速度慢和不稳定的问题。而 Task Alignment Learning（TAL）最早在 TOOD [4] 中被提出，它设计了一个统一的分类分数和预测框质量的度量标准，用此度量取代 IoU 来分配对象标签，在一定程度上缓解了任务（分类和框回归）之间的不对齐问题，经实验验证后采用 TAL 作为 YOLOv6 的默认标签分配策略。

2.3 损失函数相关研究

在目标检测中，损失函数包含分类损失和框回归损失，近年来针对每个子任务提出了多种损失函数。例如，Focal Loss 用于解决样本不平衡问题 [14]，Quality Focal Loss 进一步扩展了 Focal Loss [26]，VariFocal Loss 则从另一个角度处理正负样本不对称性 [26]；IoU 系列损失（如 GIoU [20]、CIoU [20]、SIoU [7] 等）用于精确回归边界框，Distribution Focal Loss（DFL）通过简化框位置的底层连续分布来提高定位精度 [13]。本研究对这些损失函数进行了系统评估，最终为 YOLOv6 选择了 VariFocal Loss 作为分类损失，SIoU Loss（部分模型）和 CIoU Loss（部分模型并结合 DFL）作为回归损失

2.4 量化与部署相关研究

在工业部署中，量化是常用手段，但对于重参数化模型，传统 PTQ 技术难以取得高性能，QAT 在训练和推理时匹配伪量化器也存在困难 [18]。本研究通过使用 RepOptimizer 解决重参数化模型的量化问题，获得 PTQ 友好权重，并结合敏感度分析和通道级蒸馏进一步提升量化性能，为 YOLOv6 在工业环境中的高效部署提供了可行方案。同时，研究还关注到实际部署中的其他问题，如低功耗 GPU 的使用、模型在不同硬件上的性能差异等，并在研究中进行了相应的优化和对比实验。

3 本文方法

YOLOv6 的重新设计了以下几个部分：网络设计、标签分配、损失函数、数据增强、工业实用改进以及量化和部署。

3.1 网络设计

单级目标探测器通常由以下部分组成：主干、颈部和头部。骨干网主要决定了特征表示能力，同时，它的设计对推理效率有着至关重要的影响，因为它承担了很大一部分的计算成本，类似于先前的研究如 [11]。颈部用于将低级物理特征与高级语义特征聚合在一起，然后构建各级金字塔特征图。头部由几个卷积层组成，它根据颈部组装的多级特征预测最终的检测结果。从结构的角度来看，它可以分为有锚和无锚，或者更确切地说，是参数耦合头和参数解耦头。

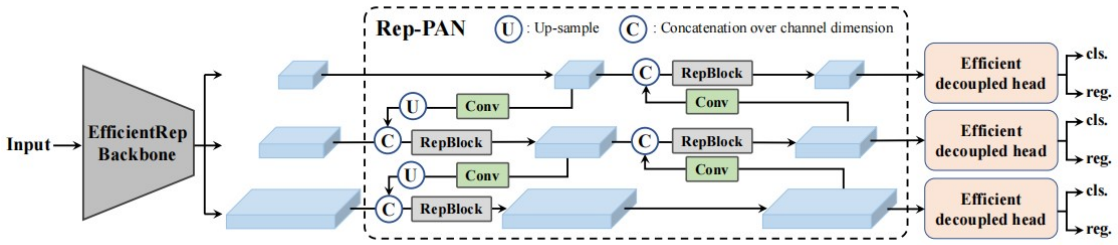

图 2. YOLOv6 框架（显示了 N 和 S）。对于 M/L，RepBlocks 被 CSPStackRep 替换。

在 YOLOv6 中，基于硬件友好型网络设计原理，我们提出了两个可缩放的可重新参数化主干和颈部，以适应不同大小的模型，以及一个具有混合信道策略的高效解耦头。YOLOv6 的总体架构如图 2 所示。Backbone、Neck、Head 的具体设计如下：

- **Backbone：** 针对不同规模模型采用不同设计。小模型在训练时以 RepBlock 为主要组件，推理阶段转换为 3×3 卷积层（RepConv），充分利用硬件计算能力；大模型则使用 CSPStackRep Block，由三个 1×1 卷积层和包含两个 RepVGG 块或 RepConv 的子块组成，通过残差连接和 CSP 连接，在平衡计算负担与精度方面表现更优，借鉴了 CSPNet [24] 的一些思想。

- **Neck：** 基于 YOLOv4 和 YOLOv5 的 PAN 拓扑结构，根据模型规模分别采用 RepBlock（小模型）或 CSPStackRep Block（大模型）进行调整，构建 Rep - PAN，有效聚合高低层特征，生成多尺度金字塔特征图。

- **Head：** 采用高效解耦头，减少中间 3×3 卷积层数量至 1 个，并依据骨干网络和颈部的宽度乘数联合缩放头部宽度，降低计算成本。同时采用 anchor - point - based 范式进行目标框回归，提升回归精度。

3.2 标签分配

在目标检测中，标签分配是训练阶段的关键环节，其主要作用是为预定义的锚点分配准确的标签。在 YOLOv6 之前，已有多种标签分配策略被提出。例如，OTA（Optimal transport assignment for object detection）将标签分配视为一个最优传输问题 [5]，它从全局视角为每个真实目标定义正 / 负训练样本。SimOTA 是 OTA 的简化版本，通过减少额外的超参数来维持性能，早期的 YOLOv6 曾采用 SimOTA 作为标签分配方法。然而，在实际应用中发现，SimOTA 会显著减缓训练进程，并且容易陷入不稳定的训练状态。

与之相比，Task Alignment Learning（TAL）最早在 TOOD [4] 中被提出，它设计了一个统一的分类分数和预测框质量的度量标准，用此度量取代 IoU 来分配对象标签，在一定程度上缓解了任务（分类和框回归）之间的不对齐问题。

在 YOLOv6 的研究中，通过在 YOLOv6 - N 模型上进行大量实验对主流标签分配策略进行评估，包括 ATSS [27]、SimOTA 和 TAL 等。结果显示，SimOTA 和 TAL 是其中表现较好的两种策略。具体而言，与 ATSS 相比，SimOTA 能够使 AP（Average Precision）提高 2.0%，而 TAL 又比 SimOTA 高出 0.5% 的 AP。综合考虑 TAL 在训练稳定性和精度提升方面的优势，最终 YOLOv6 选定 TAL 作为默认的标签分配策略，这一选择在后续的实验和应用中也证明了其有效性，有助于提高模型的整体性能和训练效果。

3.3 损失函数

在 YOLOv6 中，损失函数对于模型的性能至关重要，它主要由分类损失、回归损失和对象损失（在实际应用中最终未采用）组成。

YOLOv6 的目标检测任务包含两个子任务：分类和定位，对应两个损失函数：分类损失和框回归损失。对于每个子任务，近年来都有各种损失函数。在本节中，将介绍这些损失函数，并描述如何为 YOLOv6 选择最佳的损失函数。

- **分类损失：**在评估多种先进分类损失函数（如 Focal Loss [14]、Poly Loss [12]、Quality Focal Loss (QFL) [26] 和 VariFocal Loss (VFL) [26]）后，最终选择了 VFL。Focal Loss 旨在解决正负样本或难易样本间的类不平衡问题，而 QFL 进一步通过联合表示分类分数和定位质量来扩展 Focal Loss 用于分类监督。VFL 则基于 Focal Loss，对正负样本进行不对称处理，依据其重要程度平衡学习信号。在 YOLOv6 - N、YOLOv6 - S 和 YOLOv6 - M 模型上的实验表明，与 Focal Loss 相比，VFL 分别带来了 0.2%、0.3% 和 0.1% 的 AP 提升，从而有效提高了分类性能。

- **回归损失：**包含 IoU 系列损失和概率损失。对于 IoU 系列损失，常用的有 IoU loss、GIoU [20]、DIoU [29]、CIoU [20]、α-IoU [10] 和 SIoU [7] 等，在 YOLOv6 中，YOLOv6 - N 和 YOLOv6 - T 使用 SIoU Loss，YOLOv6 - M 和 YOLOv6 - L 使用 CIoU Loss。SIoU 和 CIoU 等 IoU 系列损失通过对预测框的边界进行整体回归，与评估指标具有一致性，因此在定位方面表现有效。对于概率损失，Distribution Focal Loss（DFL）将框位置的底层连续分布简化为离散概率分布，考虑数据模糊性和不确定性，有助于提升框定位精度，但因其输出的回归值远多于一般框回归，会带来较大计算开销，严重影响小模型训练，所以仅在 YOLOv6 - M/L 中引入。实验显示，在 YOLOv6 - N、YOLOv6 - S 和 YOLOv6 - M 模型上引入 DFL 分别可获得 0.2%、0.1% 和 0.2% 的性能增益。

- **对象损失：**最初在 FCOS [23] 和 YOLOX [6] 等框架中提出，用于降低低质量边界框分数以在后期处理中过滤它们，在 YOLOv6 中尝试应用后发现对模型有负面影响。在 YOLOv6 - N、YOLOv6 - S 和 YOLOv6 - M 网络上，引入对象损失后 AP 分别最大降低了 1.1%、0.5% 和 0.3%。分析原因可能是在 TAL 标签分配策略下，引入对象分支增加了任务对齐难度，导致性能下降，因此在 YOLOv6 中最终舍弃了对象损失。

3.4 工业实用改进

YOLOv6 的工业实用改进包括以下几个方面：

- **增加训练轮次：**将训练轮次从常规的 300 epoch 延长至 400 epoch，使模型能更好地收敛，有效提升了小模型（如 YOLOv6-N、YOLOv6-T、YOLOv6-S）的检测精度，分别提高了 0.4%、0.6% 和 0.5% 的 AP 值。

- **自蒸馏：**在分类和回归任务上应用自蒸馏策略，将教师模型设为预训练的学生模型本身，利用 KL 散度衡量预测差异计算蒸馏损失，并通过超参数 α 动态调整硬标签和软标签信息比例，实现知识从教师模型向学生模型的传递，提升模型精度，此方法类似于相关的知识蒸馏研究 [28]。实验显示，仅在分类分支应用自蒸馏可提升 0.4% AP，仅在回归任务应用可提升 0.3% AP，同时引入权重衰减可进一步提升 0.6% AP。

- **图像灰色边框处理：**针对 YOLOv5 和 YOLOv7 中使用的图像灰色边框对性能的影响进行研究，发现其虽有助于检测边缘目标但降低推理速度，且去除灰色边框会导致性能下降。通过采用 Mosaic 数据增强的 fade 策略（在最后几个 epoch 关闭 Mosaic 增强），并调整图像尺寸和灰色边框大小，使模型在保持或提升性能的同时避免了推理速度的降低。该数据增强策略参考了 [8]。

3.5 量化和部署

量化和部署包括训练友好量化（PTQ）和量化感知训练（QAT）。

- 训练友好量化（PTQ）：使用 RepOptimizer 对模型进行训练，通过梯度重参数化技术优化特征图分布，获得 PTQ 友好的权重，显著提升了模型在 PTQ 过程中的性能，为后续量化工作奠定基础。此优化技术可参考 [2]。

- 量化感知训练（QAT）：在 PTQ 基础上，结合通道级蒸馏进行 QAT。通过敏感度分析确定对量化敏感的层，对非敏感层应用伪量化器进行训练，在训练过程中利用通道级蒸馏使量化模型学习全精度模型的特征表示，进一步提高量化性能，通道级蒸馏可参考 [21]。最终，量化后的 YOLOv6 - S 在保持较高精度（43.3% AP）的同时，实现了较高的吞吐量（869 FPS，batch size = 32），在工业部署中展现出良好的性能。

如果 PTQ 不足，可以采用量化感知训练（QAT）来提高量化性能。为了解决训练和推理过程中伪量化器不一致的问题，有必要在 RepOptimizer 上构建 QAT。此外，通道式蒸馏（后来称为 CW Distill）在 YOLOv6 框架内进行了调整，如图 3 所示。这也是一种自我提炼的方法。

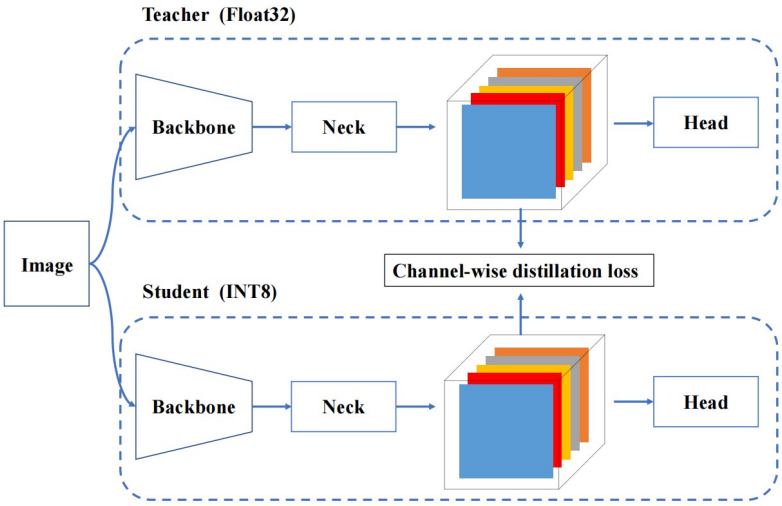

图 3. QAT 中 YOLOv6 通道式蒸馏示意图。

4 复现细节

4.1 与已有开源代码对比

本次复现参考文中给出的官方源代码，我在复现实验过程中进行了如下改进：

- **优化数据加载：**在原始的 YOLOv6 中，数据加载可能存在一定的瓶颈，特别是对于大型数据集。为了加速数据加载过程，我们对 DataLoader 进行了优化：1. 增加数据加载线程数；2. pin_memory 将数据预加载到 GPU；3. 调整 batch_size 来平衡显存使用和训练速度，这些优化可参考深度学习数据处理的一般原则 [1]。

- **改进定位损失：**引入 DIoU 损失，目前，YOLOv6 使用的是 CIoU 损失，它在计算边界框回归时考虑了中心点距离、长宽比、重叠面积等因素。但在一些特殊情况下，CIoU 可能并没有完全优化到实际情况。为了改进这一点，我们可以使用 DIoU（Distance-IoU）损失 [29]，它考虑了边界框的中心点距离，并通过优化这部分来提高精度。

```python
def diou_loss(self, pred, target):
    """
    DIoU损失：计算预测框与真实框的 DIoU
    """
    b1, b2 = pred, target
    xi1, yi1, xi2, yi2 = b1[0], b1[1], b1[2], b1[3]
    xg1, yg1, xg2, yg2 = b2[0], b2[1], b2[2], b2[3]

    # 计算IOU部分
    inter_area = (torch.min(xi2, xg2) - torch.max(xi1, xg1)).clamp(0) * (
                torch.min(yi2, yg2) - torch.max(yi1, yg1)).clamp(0)
    union_area = (xi2 - xi1) * (yi2 - yi1) + (xg2 - xg1) * (yg2 - yg1) - inter_area
    iou = inter_area / union_area

    # 计算中心点距离
    center_distance = (b1[0] + b1[2] - b2[0] - b2[2]) ** 2 + (b1[1] + b1[3] - b2[1] - b2[3]) ** 2
    diagonal_distance = (xg2 - xg1) ** 2 + (yg2 - yg1) ** 2
    diou = iou - (center_distance / diagonal_distance)

    return diou
```

图 4. 引入 DIoU 损失代码

4.2 实验环境搭建

YOLOv6 复现实验环境搭建的详细步骤如下：

1. **硬件要求：**在已有源代码的要求中，建议使用 NVIDIA 的 GPU，如 NVIDIA GeForce RTX 30 系列或更高性能的 GPU，GPU 显存应根据所使用的模型大小和批处理大小选择，对于较大规模的 YOLOv6 模型（如 YOLOv6-L），建议使用 16GB 及以上显存的 GPU 以避免显存不足问题。我使用的是 NVIDIA Tesla 100 GPU 进行复现实验，GPU 性能评估和硬件选择可参考 [25]。

2. **软件安装：**操作系统推荐使用 Ubuntu 20.04 或更高版本，我使用的是 Ubuntu 20.06。安装 pytorch 1.8 以上的版本，我使用的是 pytorch 1.8，深度学习框架的选择和使用参考 [1]。由于服务器 CUDA 版本问题，我重新安装 CUDA 10.2 用以运行复现实验，CUDA 安装可参考相关技术文档 [18]。

3. **数据集准备：**通常使用 COCO 数据集 [15]，它是目标检测领域最常用的数据集之一，包含丰富的物体类别和标注信息。YOLOv6 源代码所使用的数据集为 coco 2017 数据集，因此我也采用相同的数据集，下载地址为：https://cocodataset.org#download。

4. **环境测试：**在完成上述步骤后，下载并导入源代码，运行 YOLOv6 提供的训练脚本测试环境是否搭建成功。

4.3 训练模型

由于硬件条件限制，本次复现实验选择小模型（yolov6s 模型）进行训练，训练轮次设置为 100 次。与文中模型数据相比，训练轮次较少，训练出来的模型性能不够理想，因此，后

续实验采用 YOLOv6 的预处理模型进行实验。

5 实验结果分析

本文在 coco 数据集 [15] 上对 YOLOv6 模型进行了实验评估，论文中给出的 YOLOv6 在 coco 2017 数据集上的性能表现如表 1 所示，采用预训练模型实验得出的数据如表 2 所示。

表 1. 论文中在 COCO 2017 数据集上的 val. FPS 和延迟数据等性能表现，数据是在 TensorRT [17] 环境下，在 Tesla T4 上以 fp16 精度测量的。所有的模型都在没有预训练或任何外部数据的情况下训练了 300 个 epoch。以 640×640 为输入分辨率，对模型的精度和速度性能进行了评价。

Method	Input Size	AP_{val}	AP_{val50}	FPS (bs=1)	FPS (bs=32)	Latency (bs=1)	Params	FLOPs
YOLOv6-N	640	35.9%	51.2%	802	1234	1.2 ms	4.3 M	11.1 G
YOLOv6-T	640	40.3%	56.6%	449	659	2.2 ms	15.0 M	36.7 G
YOLOv6-S	640	43.5%	60.4%	358	495	2.8 ms	17.2 M	44.2 G
YOLOv6-M†	640	49.5%	66.8%	179	233	5.6 ms	34.3 M	82.2 G
YOLOv6-L-ReLU†	640	51.7%	69.2%	113	149	8.8 ms	58.5 M	144.0 G
YOLOv6-L†	640	52.5%	70.0%	98	121	10.2 ms	58.5 M	144.0 G

表 2. 复现中改进后在 COCO 2017 数据集上的性能表现，所有数据均采用预训练模型。

Method	Input Size	AP_{val}	$AP_{val=50}$	$AP_{val=75}$	Params (M)	FLOPs
YOLOv6n	640	37.6%	53.1%	40.7%	4.65	11.39
YOLOv6s	640	45.1%	62.1%	48.7%	18.4	45.28
YOLOv6m	640	50.0%	67.1%	54.4%	34.86	85.79
YOLOv6l	640	52.9%	70.4%	57.7%	59.61	150.73
YOLOv6lite_s	640	25.2%	39.5%	26.4%	0.55	2.23
YOLOv7lite_m	640	28.9%	44.3%	30.9%	0.79	2.64
YOLOv8lite_l	640	31.2%	47.7%	32.9%	1.09	3.44

表格 1 显示了 YOLOv6 各个规模模型在 COCO 2017 数据集上的原始性能。随着模型规模从小模型 N 到大模型 L 逐渐增大，其精度（$AP_{val=50-95}$ 和 $AP_{val=50}$）显著提升，例如 YOLOv6-N 的 AP_{val} 为 35.9%，而 YOLOv6-L 达到 52.5%。但这伴随着计算成本的增加，YOLOv6-L 的 FLOPs 高达 144.0G，推理速度显著下降，仅为 98 FPS（batch size=1）。

表格 2 是复现和改进后的性能结果，其中模型精度进一步提升，例如 YOLOv6l 的 AP_{val} 从 52.5% 提升到 52.9%，AP_{val75} 达到了 57.7%，表明模型在高 IoU 阈值下定位精度有所改进。此外，新增的 YOLOv6lite 模型展示了轻量化方向的显著优势，例如 YOLOv6lite_s 的参数量仅为 0.55M，FLOPs 仅为 2.23G，适合资源受限的边缘设备部署，轻量化模型的设计可参考 [16]。

总体来看，复现后的 YOLOv6 在保持原有精度的基础上实现了轻量化与性能优化的平衡，为不同场景提供了更多选择。同时，这些结果也为后续优化方向提供了思路，例如进一步提升轻量化模型的精度或优化高 IoU 下的检测能力。

6 总结与展望

在本次研究中，我们对 YOLOv6 模型进行了复现与优化，取得了一定成果。通过对训练策略、数据增强方法及损失函数的改进，标准模型（如 YOLOv6l）的精度有所提升，尤其是在高 IoU 阈值下的检测能力（AP_{val75}）表现更为优异。

然而，研究中也暴露了一些问题。首先，尽管轻量化模型在效率上表现突出，但其检测精度与标准模型仍有明显差距，说明轻量化优化与检测性能之间的权衡仍需进一步探索。其次，目前的结果更多关注模型的精度和计算复杂度，而实际硬件平台上的推理速度与延迟性能仍需进一步测试，特别是在不同设备上的部署性能。此外，改进后的模型在复杂场景下（如小目标检测或低光照条件）仍存在一定局限性，表明其鲁棒性仍有改进空间。

未来的研究中，我们可以考虑从以下几个方向继续优化 YOLOv6 模型：轻量化结构优化 [16]、进一步改进损失函数和领域场景适配。

参考文献

[1] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[2] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing your optimizers rather than architectures. *arXiv preprint arXiv:2205.15242*, 2022.

[3] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021.

[4] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. In *ICCV*, 2021.

[5] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 303–312, 2021.

[6] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[7] Zhora Gevorgyan. Siou loss: More powerful learning for bounding box regression. *arXiv preprint arXiv:2205.12740*, 2022.

[8] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928, 2021.

[9] Jocher Glenn. YOLOv5 release v6.1. https://github.com/ultralytics/yolov5/releases/tag/v6.1, 2022.

[10] Jiabo He, Sarah Erfani, Xingjun Ma, James Bailey, Ying Chi, and Xian-Sheng Hua. α-iou: A family of power intersection over union losses for bounding box regression. *Advances in Neural Information Processing Systems*, 34:20230–20242, 2021.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Xiaojie Shi, Shuyang Cheng, and Dragomir Anguelov. Polyloss: A polynomial expansion perspective of classification loss functions. *arXiv preprint arXiv:2204.12511*, 2022.

[13] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in Neural Information Processing Systems*, 33:21002–21012, 2020.

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[16] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[17] NVIDIA. TensorRT. https://developer.nvidia.com/tensorrt, 2018.

[18] NVIDIA. pytorch-quantization' s documentation. https://docs.nvidia.com/deeplearning/tensorrt/pytorch-quantization-toolkit/docs/index.html, 2021.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[20] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[21] Changyong Shu, Yifan Liu, Jianfei Gao, Zheng Yan, and Chunhua Shen. Channel-wise knowledge distillation for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5311–5320, 2021.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proc. Int. Conf. Computer Vision (ICCV)*, 2019.

[24] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[25] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[26] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sunderhauf. Varifocalnet: An iou-aware dense object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8514–8523, 2021.

[27] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020.

[28] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11953–11962, 2022.

[29] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12993–13000, 2020.