# 目标检测核心算法深度分析报告

## ——基于Faster R-CNN、YOLOv1/v8、RetinaNet、DETR与RT-DETR的多维度解析

姓名：肖添　专业：25大数据

RILAB智能检测组 二阶段考核报告

### 摘要

本报告从一阶段综述所调研的文献中，选取五篇最具代表性的目标检测论文——Faster R-CNN（经典两阶段Baseline）、YOLOv1/v8（单阶段实时检测开创者与最新迭代）、RetinaNet（单阶段精度突破）、DETR（Transformer端到端检测开创者）与RT-DETR（当前SOTA实时检测器），围绕三个维度展开深度分析：（1）以口语化方式阐述各算法的优劣、区别与继承关系；（2）解析各模型输入输出张量的物理含义、取值范围及映射关系；（3）选取核心计算模块（RPN、网格预测机制、Focal Loss、Multi-Head Self-Attention、混合编码器），通过伪代码推导数据张量在模块内部的形状变化过程。所有观点、公式及数据均标注原始参考文献。

关键词：目标检测；Faster R-CNN；YOLO；RetinaNet；DETR；RT-DETR；深度分析

# 一、引言

目标检测是计算机视觉领域的基础性核心任务，旨在从图像中同时完成目标的分类与空间定位[28]。自2014年R-CNN[1]将深度学习引入目标检测以来，该领域经历了从两阶段到单阶段、从CNN到Transformer、从追求精度到兼顾实时性的多次范式变革。本报告依据一阶段综述的文献调研成果，从30篇参考文献中精选五篇最具代表性的论文进行深度剖析：

（1）Faster R-CNN[3]——两阶段检测的集大成者，首次实现端到端候选区域生成与检测，是目标检测领域引用量最高的经典Baseline之一；

（2）YOLOv1[5]与YOLOv8[6]——单阶段实时检测的开创者与最新迭代，YOLOv1开创了将检测视为回归问题的全新范式，YOLOv8则代表了YOLO系列经过八代演进后的工程最优解；

（3）RetinaNet[8]——通过Focal Loss首次使单阶段检测器精度超越两阶段模型，解决了类别不平衡这一长期困扰单阶段检测器的核心问题；

（4）DETR[9]——首次将Transformer引入目标检测，开创了无需Anchor和NMS的端到端检测范式，标志着检测技术从CNN时代迈入Transformer时代；

（5）RT-DETR[12]——当前SOTA实时检测器，通过混合编码器架构实现了Transformer检测器的实时化，在精度与速度之间取得了最优平衡。

这五篇论文构成了一条清晰的技术演进链：Faster R-CNN奠定了基于区域的检测范式 -> YOLOv1开创了单阶段实时检测，将速度提升了一个数量级 ->RetinaNet证明了单阶段检测器可以在精度上与两阶段模型抗衡->DETR彻底抛弃了Anchor和NMS，开创了全新的集合预测范式 -> RT-DETR解决了Transformer检测器的实时性瓶颈。下文将从算法优劣与继承关系、输入输出张量分析、核心模块张量推导三个维度展开深度解析。

## 二、算法优劣与继承关系分析

### （一）Faster R-CNN：两阶段检测的集大成者

说到目标检测的发展史，Faster R-CNN[3]绝对是一个绕不开的里程碑。在它之前，R-CNN[1]虽然首次证明了CNN可以用于目标检测，但那个方案实在太「笨」了——对每张图生成约2000个候选区域，每个区域都要单独过一遍CNN提特征，计算量巨大，一张图推理要47秒，根本没法实用。Fast R-CNN[2]聪明了一步，引入ROI Pooling让整张图只过一次CNN，速度提升了10倍以上，但候选区域的生成还是依赖传统的选择性搜索算法，这个环节成了新的速度瓶颈。

Faster R-CNN的核心贡献就是提出了区域建议网络（Region Proposal Network, RPN），让候选区域的生成也用神经网络来做，而且和检测网络共享同一个CNN主干[3]。这样一来，候选区域生成的速度从选择性搜索的2秒降到了10毫秒级别，整个检测流程第一次实现了真正意义上的端到端训练。RPN通过在特征图上滑动窗口，在每个位置生成k个不同尺度和比例的anchor框（默认9个：3种尺度×3种比例），预测每个anchor是否包含目标以及边界框的修正量[3]。

Faster R-CNN的优势很明显：检测精度高，在PASCAL VOC 2007上mAP达到73.2%，在当时是绝对的SOTA；两阶段的「先粗筛再精修」策略让定位精度很稳定；anchor机制可以覆盖多种尺度和比例的目标[3]。但它的问题也很突出：第一，速度还是不够快，虽然比R-CNN快了很多，但5 FPS的推理速度离实时检测（30 FPS）还有很大差距；第二，anchor框需要人工设计尺度和比例，换个数据集可能就要重新调参；第三，NMS后处理在密集遮挡场景下容易误删正确的检测框[3]。

### （二）YOLO系列：单阶段实时检测的开创与演进

如果说Faster R-CNN代表了「精度优先」的检测哲学，那YOLO（You Only Look Once）就是「速度优先」的极致代表。2016年，Redmon等人提出YOLOv1[5]，开创了一种全新的检测思路：不再像两阶段模型那样先找候选区域再分类，而是把整张图一次性送进网络，直接回归出所有目标的位置和类别。用作者的话说，「你只需要看一次」[5]。

YOLOv1的核心原理很直观[5]：将输入图像划分为S*S（默认7*7）的网格，每个网格负责预测中心落在该网格内的目标。每个网格预测B个边界框（默认B=2），每个框包含5个值（x, y, w, h, confidence），同时预测C个类别概率（PASCAL VOC中C=20）。整个网络的输出就是一个S*S*(B*5+C)=7*7*30的张量，一次前向传播搞定所有预测[5]。这种设计让YOLOv1的推理速度达到了惊人的45 FPS，是Faster R-CNN的9倍，真正实现了实时检测。

但YOLOv1的问题也很明显[5]：第一，每个网格只能预测2个框、1个类别，对密集小目标的检测能力很差（比如一群鸟）；第二，边界框定位精度不如Faster R-CNN，因为它是直接回归坐标而非基于anchor做偏移修正；第三，对不常见的宽高比目标泛化能力弱。

从YOLOv1到YOLOv8，YOLO系列经历了八代持续迭代[6]：YOLOv2引入了anchor框机制和批量归一化（BN），弥补了v1定位精度差的问题；YOLOv3采用多尺度特征图预测和残差网络主干，强化了多尺度目标检测能力；YOLOv4融合了大量训练技巧（Bag-of-Freebies）和网络优

化模块（Bag-of-Specials），实现了精度与速度的双重突破；YOLOv7引入ELAN模块和模型重参数化，进一步提升复杂场景鲁棒性[6]。

YOLOv8[6]是目前YOLO系列的最新版本，代表了单阶段CNN检测器的工程最优解。它采用CSPDarknet主干网络、PAN-FPN特征融合结构、Task-Aligned Assigner动态匹配策略，并且从anchor-based转向了anchor-free设计，不再需要预设anchor框的尺度和比例[6]。YOLOv8-L在COCO数据集上达到52.9 mAP，推理速度远超两阶段模型，是工业界部署最广泛的检测模型之一[6]。

YOLO系列解决了Faster R-CNN的什么问题？最核心的一点是速度——它证明了目标检测可以做到实时，为自动驾驶、视频监控等对时延敏感的场景提供了可行方案。但YOLO系列（直到v8）仍然依赖NMS后处理，在密集遮挡场景下存在漏检风险；同时，单阶段的密集预测策略导致正负样本严重不平衡，这个问题直到RetinaNet才被真正解决[8]。

## （三）RetinaNet：用Focal Loss让单阶段检测器翻身

在RetinaNet[8]出现之前，目标检测领域有一个公认的「常识」：两阶段检测器（如Faster R-CNN）精度高但速度慢，单阶段检测器（如YOLO[5]、SSD[7]）速度快但精度差。大家普遍认为，单阶段检测器精度上不去的原因是它的网络结构不够好。但Lin等人在RetinaNet论文中提出了一个颠覆性的观点：单阶段检测器精度差的根本原因不是网络结构，而是训练过程中的类别不平衡问题[8]。

具体来说，单阶段检测器在特征图上进行密集预测，一张图会产生约10万个候选位置，其中绝大多数（99.9%以上）都是背景（负样本），真正包含目标的正样本极少[8]。传统的交叉熵损失函数对每个样本一视同仁，导致大量容易分类的背景样本主导了梯度更新，模型学不到有用的东西。这就好比一个班级里99个学霸1个差生，老师如果平均分配精力，那个差生永远学不会。

RetinaNet的核心创新就是Focal Loss[8]。它在标准交叉熵前面加了一个调制因子$(1-p_t)^\gamma$，其中$p_t$是模型对正确类别的预测概率，$\gamma$是聚焦参数（默认取2）。这个因子的妙处在于：当模型对某个样本已经很有把握（$p_t$接近1）时，$(1-p_t)^\gamma$趋近于0，这个样本对损失的贡献就被大幅降低；而对于模型拿不准的难样本（$p_t$较小），调制因子接近1，损失贡献不变[8]。这样模型就能把注意力集中在真正有价值的难样本上。

在网络结构上，RetinaNet采用ResNet+FPN作为主干网络，FPN（特征金字塔网络）通过自顶向下的路径和横向连接，将不同层级的特征融合起来，既保留了浅层的高分辨率空间信息，又融入了深层的高级语义信息[8]。配合Focal Loss，RetinaNet在COCO数据集上达到了39.1 mAP，首次超越了同期所有两阶段检测器的精度[8]。

RetinaNet解决了Faster R-CNN的什么问题？首先，它证明了单阶段检测器的精度瓶颈在于训练策略而非网络结构，为后续单阶段模型的发展指明了方向；其次，它不需要候选区域生成步骤，推理速度更快。但RetinaNet仍然依赖anchor框和NMS后处理，这两个组件在密集遮挡场景下仍是痛点[8]。

## （四）DETR：Transformer革命，抛弃Anchor和NMS

如果说Faster R-CNN和RetinaNet是在CNN检测框架内做优化，那DETR[9]就是直接掀了桌子，开创了一个全新的检测范式。2020年，Carion等人提出DETR（Detection Transformer），首次将Transformer的自注意力机制引入目标检测，实现了真正意义上的端到端检测——不需要anchor框，不需要NMS后处理，不需要任何人工设计的组件[9]。

DETR的核心思路非常优雅：把目标检测看作一个集合预测问题。具体来说，它用ResNet提取图像特征，展平后加上位置编码送入Transformer编码器进行全局自注意力计算，然后用一组可学习的「对象查询」（Object Queries，默认100个）通过Transformer解码器与编码器特征交互，每个查询直接输出一个预测结果（类别+边界框坐标）[9]。训练时通过匈牙利算法实现预测框与真实框的一对一最优匹配，不存在一个真实框被多个预测框重复匹配的问题，因此也就不需要NMS来去重了[9]。

DETR解决了之前模型的什么问题？第一，彻底消除了anchor框的设计负担，不再需要人工调参适配不同场景；第二，消除了NMS后处理，避免了密集遮挡场景下NMS误删正确检测框的问题；第三，Transformer的全局自注意力机制可以捕捉长距离特征依赖，对大目标和遮挡目标的检测鲁棒性明显优于CNN[9]。

但DETR也有明显的短板：第一，训练收敛极慢，需要500个epoch才能收敛，而Faster R-CNN只需要12-36个epoch；第二，自注意力的计算复杂度是O(N )，N是特征序列长度，对高分辨率特征图的计算开销巨大；第三，小目标检测精度明显弱于Faster R-CNN，因为Transformer编码器只在单尺度特征图上操作，缺乏多尺度特征融合[9]。这些问题催生了后续一系列改进工作，包括Deformable DETR[10]和DINO[11]。

## （五）RT-DETR：让Transformer检测器跑起来

DETR系列模型虽然在精度和设计理念上很先进，但实时性一直是硬伤。Deformable DETR[10]把计算复杂度从O(N )降到了O(NK)，DINO[11]通过去噪训练加速了收敛，但它们的推理速度仍然无法与YOLO系列相比。2023年，RT-DETR[12]的出现终于解决了这个问题——它是第一个实时的DETR检测器。

RT-DETR的核心创新是混合编码器（Hybrid Encoder）架构[12]。它没有像DETR那样把所有特征都扔给Transformer编码器做全局自注意力，而是采用了一种更聪明的策略：先用CNN对多尺度特征图进行高效的尺度内特征提取（Intra-scale），再用Transformer对不同尺度之间的特征进行跨尺度融合（Cross-scale）[12]。这样既保留了Transformer全局建模的优势，又大幅降低了计算开销。

此外，RT-DETR还引入了不确定性最小化的查询选择策略，从编码器特征中选择置信度最高的特征作为解码器的初始查询，替代DETR中随机初始化的Object Queries，加速了解码器的收敛[12]。在解码器端，RT-DETR支持通过调整解码器层数来灵活控制精度-速度的平衡，无需重新训练模型[12]。

RT-DETR在COCO数据集上，RT-DETR-L达到53.0mAP，推理速度114 FPS（T4 GPU），同时超越了YOLOv8-L的精度（52.9 mAP）和速度[12]。它解决了DETR系列的实时性瓶颈，同时继承了端到端检测、无需NMS的优势，是目前Transformer检测器中精度-速度平衡最优的方案。

## （六）五篇论文的继承关系与技术演进总结

从技术演进的角度看，这五篇论文形成了两条并行又交汇的发展脉络：

CNN路线：Faster R-CNN[3]（两阶段，高精度但慢）-> YOLOv1[5]（单阶段，开创实时检测但精度差）-> YOLOv8[6]（anchor-free，工程最优解）-> RetinaNet[8]（Focal Loss解决类别不平衡，单阶段精度首超两阶段）。这条路线的核心矛盾是「精度vs速度」，YOLO系列不断在速度基础上追赶精度，RetinaNet则从损失函数角度找到了突破口。

Transformer路线：DETR[9]（端到端，无Anchor/NMS，但慢且小目标差）-> RT-DETR[12]（混合编码器，实时化，精度速度双优）。这条路线的核心矛盾是「全局建模能力vs计算效率」，RT-DETR通过CNN+Transformer混合架构找到了平衡点。

两条路线的交汇点在于：RT-DETR既继承了DETR的端到端检测范式，又吸收了CNN高效特征提取的优势，同时在精度上超越了YOLOv8-L（53.0 vs 52.9 mAP）[12]，代表了目标检测技术的最新发展方向。而YOLO系列从v1到v8的演进过程中，也不断吸收了FPN[8]、注意力机制[29]、anchor-free[9]等其他模型的创新，体现了技术的交叉融合。

## 三、输入输出张量的物理含义与映射关系

### （一）Faster R-CNN的输入输出张量分析

#### 1. 输入张量

Faster R-CNN的输入是一张RGB图像，表示为三维张量 $I \in R^{(3\times H\times W)}$，其中3为RGB三通道，H和W分别为图像的高度和宽度（通常将短边缩放至600像素）[3]。每个像素值经过ImageNet均值归一化后，取值范围约为[-2.0,2.0]。物理含义：每个元素表示图像在特定空间位置、特定颜色通道上的光强信息。

#### 2. 输出张量

Faster R-CNN的输出包含三个部分[3]:

（a）边界框坐标张量 $B \in R^{(N\times 4)}$：N为检测到的目标数量，每个边界框用4个值表示(x1, y1, x2, y2)，分别为左上角和右下角的像素坐标。取值范围：$x1, x2 \in [0, W]$，$y1, y2 \in [0, H]$。物理含义：目标在图像中的空间位置与大小。

（b）类别概率张量 $C \in R^{(N\times(K+1))}$：K为目标类别数，+1为背景类。每行经过Softmax归一化，取值范围[0,1]，所有类别概率之和为1。物理含义：模型对每个检测框所属类别的置信程度。

（c）置信度分数 $S \in R^{(N)}$：取类别概率的最大值作为该检测框的置信度，取值范围[0,1]。物理含义：模型对该检测结果可靠性的综合评估。

#### 3. 输入到输出的映射关系

Faster R-CNN的映射过程可以概括为「特征提取->区域建议->区域分类与回归」三步[3]:

第一步，CNN主干网络（如VGG-16或ResNet-50）对输入图像进行层级化卷积操作，提取多层次的视觉特征。卷积操作通过局部感受野捕捉纹理、边缘等底层特征，池化操作逐步扩大感受野并降低空间分辨率，最终得到特征图 $F \in R^{(C\_f\times H\_f\times W\_f)}$，其中C_f为特征通道数（如512），$H\_f=H/16$，$W\_f=W/16$[3]。这一步的本质是将像素级的光强信息压缩为语义级的目标特征表示。

第二步，RPN在特征图上滑动3×3卷积窗口，对每个位置生成k个anchor框（默认k=9），预测每个anchor的前景/背景二分类得分和边界框偏移量。RPN通过学习anchor与真实框之间的偏移关系，筛选出约300个高质量候选区域[3]。这一步的本质是从密集的空间位置中快速定位可能包含目标的区域。

第三步，ROI Pooling将每个候选区域对应的特征图区域池化为固定大小（如7×7），送入全连接层进行K+1类分类和边界框精修回归[3]。分类分支通过Softmax输出类别概率，回归分支输出边界框坐标的精细修正量。最后通过NMS去除重叠检测框，得到最终输出。

用公式表达整体映射关系：$\{(b\_i, c\_i, s\_i)\}_{i=1}^{N} = NMS(Head(ROIPool(RPN(Backbone(I)))))$，其中Backbone为CNN特征提取，RPN为区域建议，ROIPool为区域特征池化，Head为分

类回归头[3]。

## （二）YOLOv1的输入输出张量分析

### 1. 输入张量

YOLOv1的输入是一张RGB图像，缩放至固定尺寸后表示为 $I \in R^{(3*448*448)}$[5]。与Faster R-CNN不同，YOLOv1要求输入尺寸固定为448*448，因为最终的全连接层需要固定维度的输入。像素值归一化至[0, 1]范围。物理含义：与Faster R-CNN相同，表示图像的光强信息。

### 2. 输出张量

YOLOv1的输出是一个三维张量 $O \in R^{(S*S*(B*5+C))}=R^{(7*7*30)}$[5]，这是它最独特的设计：

（a）边界框参数：每个网格预测B=2个边界框，每个框包含5个值(x， y， w， h， conf)。x, y为边界框中心相对于网格左上角的偏移，取值范围[0,1]；w, h为边界框宽高相对于整张图像的比例，取值范围[0， 1]；conf为置信度，等于P(Object)*IoU(pred, truth)，取值范围[0, 1][5]。物理含义：conf同时编码了「该位置是否有目标」和「预测框定位的准确程度」两层信息。

（b）类别概率：每个网格预测C=20个条件类别概率P(Class_i|Object)，取值范围[0, 1][5]。注意这是条件概率——在已知该网格包含目标的前提下，目标属于各类别的概率。最终每个框的类别置信度 = conf * P(Class_i|Object)，综合了定位和分类信息。

（c）最终输出：经过置信度阈值筛选和NMS后处理，得到检测结果。总共7*7*2=98个候选框，远少于Faster R-CNN的约16650个anchor[5]。

### 3. 输入到输出的映射关系

YOLOv1的映射过程极其简洁——「一次前向传播，直接回归」[5]：

第一步，输入图像经过24层卷积层和2层全连接层（受GoogLeNet启发的网络结构），逐步提取从低级纹理到高级语义的多层次特征。卷积层将3*448*448的输入压缩为1024*7*7的特征图[5]。

第二步，特征图经过全连接层展平为4096维向量，再映射为7*7*30=1470维输出向量，reshape为7*7*30的张量[5]。全连接层的作用是让每个网格的预测都能利用全图信息。

第三步，推理时对98个候选框进行置信度阈值筛选和NMS后处理，得到最终检测结果[5]。

整体映射：$O = \text{reshape}(FC(\text{Flatten}(Conv(I))), (7,7,30))$，经NMS后处理得到最终检测结果$\{(b_i, c_i, s_i)\}$[5]。与Faster R-CNN的多阶段流水线相比，YOLOv1的映射是一个单一的回归问题，简洁高效。

## （三）RetinaNet的输入输出张量分析

1. 输入张量

与Faster R-CNN相同，输入为归一化的RGB图像I∈R^(3×H×W)，通常将图像短边缩放至800像素[8]。物理含义与取值范围同上。

2. 输出张量

RetinaNet的输出同样包含边界框、类别概率和置信度[8]，但有一个关键区别：

（a）边界框坐标 B ∈ R^(N×4)：与Faster R-CNN相同。

（b）类别概率 C ∈ R^(N×K)：注意这里没有背景类。RetinaNet对每个类别独立使用Sigmo id激活函数（而非Softmax），每个类别的概率独立取值于[0, 1]，不要求所有类别概率之和 1[ 8]。物理含义：模型对每个检测框属于各个类别的独立置信度。这种设计允许一个检测框同时属于多个类别（虽然在实践中很少出现）。

（c）置信度分数 S ∈ R^(N)：取所有类别概率的最大值。

3. 输入到输出的映射关系

RetinaNet的映射过程为「特征提取->多尺度密集预测->Focal Loss加权训练」[8]：

第一步，ResNet主干网络提取多层特征，FPN通过自顶向下路径和1×1横向连接，生成多尺度特征金字塔{P3, P4, P5, P6, P7}，分辨率从P3（1/8）到P7（1/128）逐级降低[8]。FPN的核心价值在于：浅层特征（P3）保留了小目标的空间细节，深层特征（P7）包含了大目标的语义信息，融合后每个尺度都同时具备空间和语义信息。

第二步，在每个尺度的特征图上，分类子网络和回归子网络分别对每个anchor位置进行密集预测。分类子网络输出K×A个Sigmoid概率（A为每个位置的anchor数，默认A=9），回归子网络输出4×A个边界框偏移量[8]。

第三步，训练时使用Focal Loss替代标准交叉熵：FL(p_t) = −α_t(1−p_t)^γ log(p_t)，其中α_ t为类别平衡因子（默认0.25），γ为聚焦参数（默认2）[8]。推理时通过置信度阈值筛选和NMS后处理得到最终检测结果。

整体映射：{(b_i，c_i，s_i)} = NMS(SubNet_cls(FPN(I))，SubNet_reg(FPN(I)))，训练由Focal Loss驱动模型聚焦难样本[8]。

## （四）DETR的输入输出张量分析

1. 输入张量

输入为归一化的RGB图像I∈R^(3×H×W)，通常H=800,W=1066（保持长宽比缩放）[9]。与CNN检测器不同的是，DETR还需要一个位置编码张量PE ∈R^(d×H_f×W_f)，其中d为特征维度（默认256），H_f=H/32，W_f=W/32。位置编码采用固定的正弦/余弦函数生成，为Transformer提供空间位置信息[9]。此外，解码器的输入还包括一组可学习的Object Queries OQ ∈ R^(N_q ×d)，N_q为查询数量（默认100），d=256。物理含义：每个查询代表一个「检测槽位」，负责检测图像中的一个目标（或预测为「无目标」）[9]。

2. 输出张量

DETR的输出是一个固定大小的预测集合，包含N_q=100个预测结果[9]：

（a）边界框坐标B∈R^(N_q ×4)：每个框用归一化的中心坐标和宽高表示(cx，cy，w, h)，取值范围均为[0，1]（相对于图像尺寸的比例）[9]。物理含义：目标中心在图像中的相对位置及目标的相对大小。注意这与Faster R-CNN使用绝对像素坐标不同。

（b）类别概率C∈ R^(N_q ×(K+1))：K个目标类别加1个「无目标」类，经Softmax归一化，取值范围[0，1]。物理含义：每个查询槽位对应的目标类别。当某个查询预测为无目标类时，表示该槽位未检测到目标[9]。

（c）最终输出：过滤掉预测为 类的查询后，剩余的预测即为检测结果，无需NMS后处理。输出目标数量N ⩽ N_q[9]。

3. 输入到输出的映射关系

DETR的映射过程为「CNN特征提取->Transformer编码->集合预测解码」[9]：

第一步，ResNet-50主干网络将输入图像映射为特征图F∈R^(2048×H/32×W/32)，再通过1×1卷积降维至F' ∈R^(256×H/32×W/32)[9]。将F'展平为序列 Z_0 ∈R^(HW/1024×256)，加上位置编码后送入Transformer编码器。

第二步，Transformer编码器由6层堆叠的自注意力层组成。每层中，序列中的每个位置都与所有其他位置计算注意力权重，从而建立全局特征依赖关系。编码器输出 Z ∈ R^(HW/1024×25 6)[9]。这一步的本质是让每个空间位置的特征都融合了全图的上下文信息，这是CNN的局部卷积操作无法实现的。

第三步，Transformer解码器同样由6层组成。每层包含自注意力（查询之间的交互）和交叉注意力（查询与编码器特征的交互）两个子层[9]。N_q个Object Queries通过交叉注意力从编码器特征中「提取」各自关注的目标信息，最终每个查询输出一个d维向量。

第四步，两个前馈网络（FFN）分别将解码器输出映射为类别概率和边界框坐标[9]。训练时使用匈牙利算法进行一对一匹配，损失函数为分类损失（交叉熵）与边界框损失（L1损失+Glo U损失[25]）的加权和。

整体映射：{(b_i,=c_i)}=FFN(Decoder(Encoder(Backbone(I)+ PE), OQ))，无需NMS后处理[9]。

## （五）RT-DETR的输入输出张量分析

1. 输入张量

输入为归一化的RGB图像 I∈R^(3×640×640)（固定分辨率）[12]。与DETR不同，RT-DETR不使用固定的正弦位置编码，而是在混合编码器内部使用可学习的位置编码[12]。

2. 输出张量

RT-DETR的输出格式与DETR一致[12]：

（a）边界框坐标 B∈R^(N_q ×4)：归一化的(cx, cy, w, h)，取值范围[0, 1]。

（b）类别概率C∈ R^(N_q ×K)：使用Sigmoid激活（与RetinaNet类似），每个类别独立预测，取值范围[0, 1][12]。

（c）查询数量N_q默认为300，可通过调整解码器层数灵活控制精度-速度平衡[12]。

3. 输入到输出的映射关系

RT-DETR的映射过程为「多尺度CNN特征提取->混合编码->查询选择->Transformer解码」[12]：

第一步，主干网络（如ResNet-50或HGNetv2）提取多尺度特征{S3,S4,S5}，分辨率分别为输入的1/8、1/16、1/32[12]。

第二步，混合编码器的尺度内特征交互（AIFI）模块对最高层特征S5进行Transformer自注意力计算，捕捉全局语义信息；跨尺度特征融合（CCFM）模块通过CNN卷积将不同尺度的特征进行高效融合[12]。这种设计的关键洞察是：只对语义信息最丰富的高层特征做Transformer计算，低层特征用CNN处理即可，大幅降低了计算量。

第三步，不确定性最小化查询选择：从编码器输出中选择置信度最高的N_q个特征作为解码器的初始查询，替代DETR中随机初始化的Object Queries[12]。这些查询已经包含了丰富的目标先验信息，加速了解码器收敛。

第四步，Transformer解码器对选定的查询进行迭代精修，通过自注意力和交叉注意力输出最终的类别概率和边界框坐标[12]。

整体映射：{(b_i,c_i)}=Decoder(QuerySelect(HybridEncoder(Backbone(I))))，无需NMS后处理[12]。

## （六）五个模型输入输出张量对比总结

| 模型 | 输入张量 | 输出边界框 | 输出类别 | 后处理 |
| --- | --- | --- | --- | --- |
| Faster R-CNN | I∈R^(3*H*W)<br>归一化RGB图像 | B∈R^(N*4)<br>绝对坐标(x1,y1,x2,y2) | C∈R^(N*(K+1))<br>Softmax含背景类 | 需要NMS |
| YOLOv1 | I∈R^(3*448*448)固定尺寸RGB | O∈R^(7*7*30)<br>归一化(x,y,w,h,conf) | 条件概率P(C\|Obj)<br>每网格共享 | 需要NMS |
| RetinaNet | I∈R^(3*H*W)<br>归一化RGB图像 | B∈R^(N*4)<br>绝对坐标(x1,y1,x2,y2) | C∈R^(N*K)<br>Sigmoid无背景类 | 需要NMS |

| | | | | |
|---|---|---|---|---|
| DETR | I∈R^(3*H*W) +位置编码PE +Object Queries | B∈R^(Nq*4) 归一化(cx,cy,w,h) | C∈R^(Nq*(K+1)) Softmax含null类 | 无需NMS |
| RT-DETR | I∈R^(3*640*640)固定分辨率 | B∈R^(Nq*4) 归一化(cx,cy,w,h) | C∈R^(Nq*K) Sigmoid无背景类 | 无需NMS |

表1 五个模型输入输出张量对比

## 四、核心计算模块的张量形状变化推导

本章从五篇论文中各选取一个核心计算模块，通过伪代码和文字推导描述数据张量在模块内部的形状变化过程，明确注明每一步操作后的维度变化及计算目的。

### （一）Faster R-CNN的核心模块：区域建议网络（RPN）

RPN是Faster R-CNN的核心创新，负责从特征图中生成高质量的候选区域[3]。下面以输入图像尺寸为3×600×800、主干网络为VGG-16为例，推导RPN内部的张量变化过程。

**输入**

CNN主干网络输出的特征图 F ∈ R^(512×37×50)，其中512为通道数，37≈600/16，50≈80 0/16（VGG-16的下采样率为16）[3]。

**步骤1：3×3滑动窗口卷积**

```
# 输入: F ∈ R^(512×37×50)
# 操作: 3×3卷积, padding=1, 输出通道512
F_rpn = Conv2d(F, kernel=3×3, out=512, pad=1) # -> R^(512×37×50)
F_rpn = ReLU(F_rpn) # -> R^(512×37×50)
```

计算目的：在每个空间位置提取3×3邻域的局部特征，为后续的分类和回归提供特征基础。padding=1保证输出空间尺寸不变[3]。

**步骤2：分类分支（前景/背景二分类）**

```
# 输入: F_rpn ∈ R^(512×37×50)
# 操作: 1×1卷积, 输出通道 2k=18 (k=9个anchor, 每个2类)
cls_score = Conv2d(F_rpn, kernel=1×1, out=18) # -> R^(18×37×50)
cls_score = reshape(cls_score, (2, 9×37×50)) # -> R^(2×16650)
```

计算目的：对每个空间位置的每个anchor预测前景（包含目标）和背景的概率。总共37×50 ×9=16650个anchor，每个anchor得到一个前景概率和一个背景概率[3]。

**步骤3：回归分支（边界框偏移量）**

```
# 输入: F_rpn ∈ R^(512×37×50)
# 操作: 1×1卷积, 输出通道 4k=36 (k=9个anchor, 每个4个偏移量)
bbox_delta = Conv2d(F_rpn, kernel=1×1, out=36) # ->
R^(36×37×50) bbox_delta = reshape(bbox_delta, (16650, 4)) # ->
```

计算目的：对每个anchor预测4个边界框偏移量(Δx, Δy, Δw, Δh)，分别表示中心点x偏移、中心点y偏移、宽度缩放、高度缩放[3]。通过这些偏移量将预设的anchor框修正为更精确的候选区域。

步骤4：候选区域生成

```
# 输入: cls_prob ∈ R^(2×16650), bbox_delta ∈ R^(16650×4)
# 操作: 根据前景概率排序, 取top-N, 应用偏移量, NMS筛选
proposals = apply_delta(anchors, bbox_delta) # -> R^(16650×4)
proposals = filter_by_score(proposals, cls_prob) # ->
R^(~6000×4) proposals = NMS(proposals, threshold=0.7) # ->
```

计算目的：从16650个anchor中筛选出约300个高质量候选区域，供后续的ROI Pooling和分类回归头使用[3]。NMS阈值0.7用于去除高度重叠的冗余候选框。

## （二）YOLOv1的核心模块：网格预测机制（Grid Prediction）

网格预测机制是YOLOv1最具开创性的设计，它将目标检测从「候选区域分类」问题转化为「空间网格回归」问题[5]。下面以输入图像尺寸为3*448*448、S=7、B=2、C=20为例推导。

输入

网络最后一个卷积层输出的特征图，经展平和全连接层后得到的特征向量 F ∈ R^(4096)[5]。

步骤1：全连接层映射到网格输出

```
# 输入: F ∈ R^(4096)
# 操作: 全连接层, 输出维度 S*S*(B*5+C) = 7*7*30 = 1470
O_flat = FC(F, out=1470) # -> R^(1470)
O = reshape(O_flat, (7, 7, 30)) # -> R^(7*7*30)
```

计算目的：将全局特征向量映射为空间网格化的预测张量。reshape操作赋予了输出空间结构——每个7*7的网格位置对应图像中一个区域[5]。

步骤2：解析每个网格的预测内容

```
# 输入: O ∈ R^(7*7*30)
# 每个网格的30维向量拆解为:
# Box1: (x1, y1, w1, h1, conf1) = O[:,:,0:5] # -> R^(7*7*5)
# Box2: (x2, y2, w2, h2, conf2) = O[:,:,5:10] # -> R^(7*7*5)
```

```
# ClassProb: P(C_i|Obj) = O[:,:,10:30] # -> R^(7*7*20)
```

计算目的：将30维向量解码为具有物理含义的检测参数[5]。x,y经过Sigmoid约束到[0,1]，表示框中心相对于网格左上角的偏移；w,h表示框宽高相对于整张图像的比例；conf表示该框包含目标的置信度。

步骤3：计算最终类别置信度

```
# 输入: conf ∈ R^(7*7*2), ClassProb ∈ R^(7*7*20)
# 操作: 将置信度与类别概率相乘
# 对Box1:
conf1 = O[:,:,4:5] # -> R^(7*7*1)
score1 = conf1 * ClassProb # -> R^(7*7*20) 广播乘法
# 对Box2:
conf2 = O[:,:,9:10] # -> R^(7*7*1)
score2 = conf2 * ClassProb # -> R^(7*7*20) 广播乘法
# 合并所有框的得分
all_scores = stack(score1, score2) # -> R^(7*7*2*20)
all_scores = reshape(all_scores, (98, 20)) # -> R^(98*20)
```

计算目的：最终得分 = P(Object)*IoU*P(Class|Object) = P(Class)*IoU，同时编码了「是什么类别」和「定位有多准」两层信息[5]。98个候选框（7*7*2）各自对20个类别都有一个得分。

步骤4：NMS后处理

```
# 输入: all_boxes ∈ R^(98*4), all_scores ∈ R^(98*20)
# 操作: 对每个类别独立做NMS
for each class k in 1..20:
scores_k = all_scores[:, k] # -> R^(98)
keep = NMS(all_boxes, scores_k, threshold=0.5) # -> R^(~N_k)
# 最终输出: 所有类别的检测结果合并
```

计算目的：去除同一目标的重复检测框，每个类别独立处理[5]。最终输出的检测框数量远小于98，通常为个位数到几十个。

## （三）RetinaNet的核心模块： Focal Loss计算过程

Focal Loss是RetinaNet的核心创新，解决了单阶段检测器中正负样本极度不平衡的问题[8]。下面以单个FPN层级P3的分类输出为例，推导Focal Loss的张量计算过程。

输入分类子网络对P3层级特征图的原始输出（logits）Z ∈R^(A×K×H3×W3)，其中A=9为anchor数，K=80为COCO类别数，H3=100，

W3=128（假设输入800×1024）[8]。对应的真实标签 Y ∈R^(A×K×H3×W3)，取值为0或1（one–hot编码）。

步骤1：Sigmoid激活

```
# 输入: Z ∈ R^(9×80×100×128)
# 操作: 逐元素Sigmoid
P = Sigmoid(Z) # –> R^(9×80×100×128)
# P中每个元素 p ∈ (0, 1), 表示该anchor属于该类别的独立概率
```

计算目的：将原始logits映射到(0,1)概率空间。使用Sigmoid而非Softmax，使得每个类别的预测相互独立[8]。

步骤2：计算p_t

```
# 输入: P ∈ R^(9×80×100×128), Y ∈ R^(9×80×100×128)
# 操作: 根据真实标签选择对应概率
p_t = Y * P + (1 – Y) * (1 – P) # –> R^(9×80×100×128)
# 当Y=1时, p_t=P (正样本的预测概率)
# 当Y=0时, p_t=1–P (负样本的预测概率)
```

计算目的：统一正负样本的概率表示。p_t越大，说明模型对该样本的预测越准确（越「容易」）；p_t越小，说明模型对该样本的预测越不准确（越「困难」）[8]。

步骤3：计算调制因子

```
# 输入: p_t ∈ R^(9×80×100×128)
# 操作: 计算 (1 – p_t)^ γ, γ=2
modulating_factor = (1 – p_t) ** 2 # –> R^(9×80×100×128)
# 当p_t–>1 (易分样本): modulating_factor –> 0, 损失被抑制
# 当p_t–>0 (难分样本): modulating_factor –> 1, 损失不变
```

计算目的：这是Focal Loss的核心。调制因子动态降低易分样本的损失权重，使模型训练聚焦于难分样本（如小目标、遮挡目标）[8]。 γ=2时，一个p_t=0.9的易分样本的损失权重被降低100倍。

步骤4：计算最终Focal Loss

```
# 输入: p_t, modulating_factor ∈ R^(9×80×100×128)
# 操作: 加权交叉熵
alpha_t = Y * 0.25 + (1 – Y) * 0.75 # –>
R^(9×80×100×128) CE = –log(p_t) # –>
R^(9×80×100×128)
FL = alpha_t * modulating_factor * CE # –>
```

计算目的： α_t为类别平衡因子，正样本权重0.25，负样本权重0.75，进一步平衡正负样本的贡献[8]。最终损失除以正样本anchor数量进行归一化，得到一个标量损失值用于反向传播。

## （四）DETR的核心模块：多头自注意力（Multi-Head Self-Attention, MHSA）

MHSA是Transformer编码器的核心计算模块，负责建立特征序列中所有位置之间的全局依赖关系[9]。下面以DETR编码器中的一层MHSA为例，推导张量变化过程。设特征维度d=256，注意力头数h=8，序列长度L=H_f×W_f=25×34=850。

**输入**

编码器输入序列X∈ R^(L×d)=R^(850×256)，其中每一行是一个空间位置的d维特征向量，已叠加位置编码[9]。

**步骤1：线性投影生成Q、K、V**

```
# 输入: X ∈ R^(850×256)
# 操作: 三个独立的线性变换
Q = X @ W_Q # W_Q ∈ R^(256×256), Q ∈ R^(850×256)
K = X @ W_K # W_K ∈ R^(256×256), K ∈ R^(850×256)
V = X @ W_V # W_V ∈ R^(256×256), V ∈ R^(850×256)
```

计算目的：Q（Query）表示「我在找什么信息」，K（Key）表示「我有什么信息」，V（Value）表示「我能提供什么信息」。三个投影矩阵是可学习的参数[9]。

**步骤2：拆分为多头**

```
# 输入: Q, K, V ∈ R^(850×256)
# 操作: 拆分为h=8个头, 每头维度 d_k = 256/8 = 32
Q = reshape(Q, (850, 8, 32)).transpose(0,1) # -> R^(8×850×32)
K = reshape(K, (850, 8, 32)).transpose(0,1) # -> R^(8×850×32)
V = reshape(V, (850, 8, 32)).transpose(0,1) # -> R^(8×850×32)
```

计算目的：多头机制让模型在不同的子空间中学习不同类型的注意力模式。例如，某些头可能关注局部纹理特征，另一些头关注全局结构特征[9]。

**步骤3：缩放点积注意力计算**

```
# 输入: Q, K ∈ R^(8×850×32)
# 操作: Q与K的转置做矩阵乘法, 除以sqrt(d_k)缩放
attn_weights = (Q @ K.transpose(-2,-1)) / sqrt(32) # ->
R^(8×850×850) attn_weights = Softmax(attn_weights, dim=-1) # ->
R^(8×850×850)
```

计算目的：计算每对位置之间的相关性。注意力矩阵的大小为850×850，这意味着每个位置都与所有850个位置建立了联系——这就是「全局自注意力」[9]。除以sqrt(d_k)=sqrt(32)是为了防止点积值过大导致Softmax梯度消失。计算复杂度为O(L d_k)=O(850 ×32)，这也是DETR在高分辨率特征图上计算开销大的根本原因。

步骤4：加权聚合与多头拼接

```
# 输入: attn_weights ∈ R^(8×850×850), V ∈ R^(8×850×32)
# 操作: 注意力权重与V做矩阵乘法
attn_output = attn_weights @ V # -> R^(8×850×32)
# 拼接8个头的输出
attn_output = attn_output.transpose(0,1) # -> R^(850×8×32)
attn_output = reshape(attn_output, (850, 256)) # -> R^(850×256)
# 最终线性投影
output = attn_output @ W_O # W_O ∈ R^(256×256) # -> R^(850×256)
```

计算目的：每个位置的输出是所有位置Value的加权和，权重由注意力分数决定。拼接8个头的输出后通过线性投影融合多头信息，最终输出维度与输入相同（850×256），保证可以堆叠多层编码器[9]。经过MHSA后，每个位置的特征都融合了全图的上下文信息。

## （五）RT-DETR的核心模块：混合编码器（Hybrid Encoder）

混合编码器是RT-DETR的核心创新，通过尺度内特征交互 （AIFI） 和跨尺度特征融合（CCFM）两个子模块的协同工作，在保持全局建模能力的同时大幅降低计算开销[12]。下面以输入640×640、主干网络为ResNet-50为例推导。

输入

主干网络输出的多尺度特征图[12]：

```
S3 ∈ R^(256×80×80) # 1/8分辨率, 空间细节丰富, 适合小目
标S4 ∈ R^(256×40×40) # 1/16分辨率, 中等语义信息
S5 ∈ R^(256×20×20) # 1/32分辨率, 高级语义信息丰富, 适合大目标
```

注：经过1×1卷积统一通道数为256。

步骤1：AIFI——尺度内自注意力（仅对S5）

```
# 输入: S5 ∈ R^(256×20×20)
# 操作1: 展平为序列
S5_flat = reshape(S5, (256, 400)).transpose() # -> R^(400×256)
# 操作2: 添加位置编码
S5_flat = S5_flat + PosEmbed(400, 256) # -> R^(400×256)
# 操作3: 多头自注意力(与DETR的MHSA相同)
S5_attn = MHSA(S5_flat, heads=8) # -> R^(400×256)
# 操作4: FFN前馈网络
S5_out = FFN(S5_attn) # -> R^(400×256)
# 操作5: 恢复空间形状
```

```
S5_out = reshape(S5_out.transpose(), (256,20,20)) # -> R^(256×20×20)
```

计算目的：仅对最高层特征S5做Transformer自注意力，捕捉全局语义依赖[12]。关键洞察：S5的序列长度仅为400（20×20），远小于DETR中的850+，计算复杂度O(400 )仅为DETR的约22%，这是RT-DETR实现实时化的关键设计。低层特征S3（6400个位置）和S4（1600个位置）如果也做自注意力，计算量将不可接受。

步骤2：CCFM——跨尺度特征融合

CCFM采用类似FPN的自顶向下路径，但使用可变形卷积增强特征融合效果[12]：

```
# —— 自顶向下路径 ——
# 输入: S5_out ∈ R^(256×20×20), S4 ∈ R^(256×40×40), S3 ∈ R^(256×80×80)

# 融合S5到S4:
S5_up = Upsample(S5_out, scale=2) # -> R^(256×40×40)
P4 = RepBlock(Concat(S5_up, S4)) # Concat->R^(512×40×40)
# RepBlock->R^(256×40×40)

# 融合P4到S3:
P4_up = Upsample(P4, scale=2) # -> R^(256×80×80)
P3 = RepBlock(Concat(P4_up, S3)) # Concat->R^(512×80×80)
# RepBlock->R^(256×80×80)

# —— 自底向上路径 ——
# 融合P3到P4:
P3_down = Conv2d(P3, stride=2) # -> R^(256×40×40)
P4_out = RepBlock(Concat(P3_down, P4)) # Concat->R^(512×40×40)
# RepBlock->R^(256×40×40)

# 融合P4_out到S5:
P4_down = Conv2d(P4_out, stride=2) # -> R^(256×20×20)
P5_out = RepBlock(Concat(P4_down, S5_out)) # Concat->R^(512×20×20)
# RepBlock->R^(256×20×20)
```

计算目的：通过双向特征融合（先自顶向下再自底向上），让每个尺度的特征都同时包含高分辨率的空间细节和高层的语义信息[12]。RepBlock（重参数化模块）在训练时使用多分支结构增强特征表达，推理时融合为单个卷积，不增加推理开销。

输出

```
# 混合编码器最终输出三个尺度的增强特征:
P3_out ∈ R^(256×80×80) # 小目标检测特征
P4_out ∈ R^(256×40×40) # 中目标检测特征
P5_out ∈ R^(256×20×20) # 大目标检测特征
```

这三个特征图将被展平拼接为统一序列，送入不确定性最小化查询选择模块，选出Top-N_q个特征作为解码器的初始查询[12]。

## 五、总结与展望

本报告从一阶段综述的30篇参考文献中，选取了Faster R-CNN[3]、YOLOv1/v8[5][6]、RetinaNet[8]、DETR[9]和RT-DETR[12]五篇最具代表性的论文，围绕三个维度完成了深度分析。

在算法优劣与继承关系方面，五篇论文构成了清晰的技术演进脉络：Faster R-CNN奠定了基于区域的两阶段检测范式，但受限于速度和anchor设计；YOLOv1开创了单阶段实时检测范式，将速度提升了一个数量级，YOLOv8则代表了YOLO系列八代迭代后的工程最优解；RetinaNet通过Focal Loss解决了单阶段检测器的类别不平衡问题，首次实现单阶段精度超越两阶段；DETR开创了基于Transformer的端到端集合预测范式，彻底消除了anchor和NMS，但面临收敛慢和计算开销大的挑战；RT-DETR通过混合编码器架构解决了Transformer检测器的实时性瓶颈，在精度和速度上均达到了当前最优水平[3][5][6][8][9][12]。

在输入输出张量分析方面，五个模型的输入均为归一化的RGB图像张量，但在输出表示上存在显著差异：YOLOv1采用独特的网格化输出张量，将检测转化为回归问题；Faster R-CNN和RetinaNet使用绝对像素坐标和Softmax/Sigmoid分类，依赖NMS后处理；Transformer模型使用归一化坐标和集合预测，无需NMS。这种差异反映了从「密集预测+后处理」到「稀疏集合预测」的范式转变。

在核心模块张量推导方面，RPN通过滑动窗口卷积实现了高效的候选区域生成；YOLOv1的网格预测机制将全局特征直接映射为空间化的检测输出，开创性地简化了检测流程；Focal Loss通过调制因子动态调整样本权重，解决了正负样本不平衡问题；MHSA通过全局注意力矩阵建立了所有位置之间的特征依赖，但计算复杂度为O(L^2)；混合编码器通过仅对高层特征做自注意力、低层特征用CNN处理的策略，将计算复杂度降低了约78%，实现了精度与速度的最优平衡[3][5][8][9][12]。

展望未来，目标检测技术正朝着更高精度、更快速度、更强泛化能力的方向发展。RT-DETR所代表的CNN+Transformer混合架构可能成为主流趋势，同时结合知识蒸馏[22]、神经架构搜索[22]、多模态融合[13]等技术，有望进一步突破当前的性能瓶颈，推动目标检测在自动驾驶、遥感监测、智能安防等领域的深入应用[28]。

# 参考文献


[1] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//CVPR. 2014: 580–587.

[2] Girshick R. Fast R–CNN[C]//ICCV. 2015: 1440–1448.

[3] Ren S, He K, Girshick R, et al. Faster R–CNN: Towards real–time object detection with region proposal networks[J]. NeurIPS, 2015, 28.

[4] He K, Gkioxari G, Dollár P, et al. Mask R–CNN[C]//ICCV. 2017: 2961–2969.

[5] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real–time object detection[C]//CVPR. 2016: 779–788.

[6] Jocher G, Chaurasia A, QiuJ, et al. YOLOv8: Ultralytics YOLO[EB/OL]. https://github.com/ultralytics/ultralytics, 2023.

[7] Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector[C]//ECCV. 2016: 21–37.

[8] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]//ICCV. 2017: 2980–2988.

[9] Carion N, Massa F, Synnaeve G, et al. End–to–end object detection with transformers[C]//ECCV. 2020: 213–229.

[10] Zhu X, Su W, Lu L, et al. Deformable DETR: Deformable transformers for end–to–end object detection[J]. NeurIPS, 2020, 33: 9994–10005.

[11] Zhang H, WenJ, BianJ, et al. DINO: DETR with improved deformation and online hard example mining[C]//CVPR. 2022: 12413–12422.

[12] Zhou X, Wang D, Krähenbühl P. RT–DETR: Real–time DETR for object detection[J]. arXiv:2304.08069, 2023.

[13] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv:2010.11929, 2020.

[14] Goodfellow I, Pouget–Abadie J, Mirza M, et al. Generative adversarial nets[J]. NeurIPS, 2014, 27.

[15] Tang H, Zhang Y, Liu X, et al. GAN–Det: Detecting objects with generative adversarial networks[C]//AAAI. 2018, 32(1).

[16] Zhu J Y, Park T, Isola P, et al. Unpaired image–to–image translation using cycle–consistent adversarial networks[C]//ICCV. 2017: 2223–2232.

[17] Miyato T, Kataoka T, Koyama M, et al. Spectral normalization for generative adversarial networks[J]. arXiv:1802.05957, 2018.

[18] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv:1704.04861, 2017.

[19] Howard A G, Sandler M, Chu G, et al. Searching for mobilenetv3[C]//ICCV. 2019: 1314–1324.

[20] Zhang X, Zhou X, Lin M, et al. ShuffleNet: An extremely efficient convolutional neural network for mobile devices[C]//CVPR. 2018: 6848–6856.

[21] Han K, Wang Y, Tian Q, et al. GhostNet: More features from cheap operations[C]//CVPR. 2020: 1580–1589.

[22] Rabinovitch A, Mirza M, Gong B, et al. YOLO–NAS: Neural architecture search for real–time object detection[EB/OL]. https://github.com/Deci–AI/super–gradients, 2023.

[23] Kingma D P, Dhariwal P. Glow: Generative flow with invertible 1x1 convolutions[C]//NeurIPS. 2018, 31.

[24] Everingham M, Van Gool L, Williams C K I, et al. The PASCAL visual object classes (VOC) challenge[J]. IJCV, 2010, 88(2): 303–338.

[25] Rezatofighi H, Tsoi N, GwakJ Y, et al. Generalized intersection over union: A metric and a loss for bounding box regression[C]//CVPR. 2019: 658–666.

[26] Zheng Z, Wang P, Liu W, et al. Distance–IoU loss: Faster and better learning for bounding box regression[C]//AAAI. 2020, 34(07): 12993–13000.

[27] Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: From error visibility to structural similarity[J]. IEEE TIP, 2004, 13(4): 600–612.

[28] Trigka M, Dritsas E, Moustakos K. A Comprehensive Survey of Machine Learning Techniques and Models for Object Detection[J]. Sensors, 2022, 22(18): 6988.

[29] 陈谢发, 李刚, 张艳宁. 遥感图像小目标检测的空间–通道注意力网络[J]. 计算机学报, 2021, 44(5): 921–936.

[30] Lim B, Son S, Kim H, et al. Enhanced deep residual networks for single image super–resolution[C]//CVPR Workshops. 2017: 136–144.