# **基于[ultralytics](**https://github.com/ultralytics/ultralytics**)的RT-DETR改进项目**

    ------- From 哔哩哔哩 魔鬼面具UP -------

    本项目使用的ultralytics版本为8.0.201,在ultralytics/\_\_init\_\_.py中的\_\_version\_\_有标识.

    其中为了尽量对齐与官方PaddlePaddle-(RT-DETR)中的训练策略,主要对源码作了以下修改:

    1. 在ultralytics/engine/trainer.py中的optimizer\_step函数中的torch.nn.utils.clip\_grad\_norm\_中的参数max\_norm修改为0.1

    2. 在ultralytics/engine/trainer.py中的\_setup\_train函数中将self.args.nbs等于self.batch\_size,这样做的目的是让模型不需要积累梯度再进行更新参数

    3. ultralytics/cfg/default.yaml配置文件的更改<详细请看使用教程视频>

    注意:

    1. 如对以上修改觉得不对的地方,需要提供具体不对的地方和对应官方paddlepaddle或者pytorch版本实现的地方私聊我

    2. 以上修改只是为了<尽量>对齐官方PaddlePaddle的设置,不代表完全对齐,因为官方的设置不代表一定在其他数据集精度高,所以这里只修改了最影响精度部分的设置

    3. 本项目使用方法跟官方一致,具体可参考(https://docs.ultralytics.com/models/rtdetr/)

    4. 此项目只用于训练RT-DETR,如需要训练yolov5,v8模型请移步到官方代码中使用.

# **RT-DETR环境配置**

    1. 执行pip uninstall ultralytics把安装在环境里面的ultralytics库卸载干净.<这里需要注意,如果你也在使用yolov8,最好使用anaconda创建一个虚拟环境供本代码使用,避免环境冲突导致一些奇怪的问题>

    2. 卸载完成后同样再执行一次,如果出现WARNING: Skipping ultralytics as it is not installed.证明已经卸载干净.

    3. 如果需要使用官方的CLI运行方式,需要把ultralytics库安装一下,执行命令:<python setup.py develop>,当然安装后对本代码进行修改依然有效.(develop作用解释具体可看: https://blog.csdn.net/qq\_16568205/article/details/110433714)  注意:不需要使用官方的CLI运行方式,可以选择跳过这步

    4. 额外需要的包安装命令:

        pip install timm thop efficientnet\_pytorch einops grad-cam dill -i https://pypi.tuna.tsinghua.edu.cn/simple

        以下主要是使用dyhead必定需要安装的包,如果安装不成功dyhead没办法正常使用!

        pip install -U openmim

        mim install mmengine

        mim install "mmcv>=2.0.0"

    5. 运行时候如果还缺什么包就请自行安装即可.

# **RT-DETR预训练权重下载**

    请看weights/weight\_download.txt已提供好百度云链接.

# **自带的一些文件说明**

1. train.py

    训练模型的脚本

2. main\_profile.py

    输出模型和模型每一层的参数,计算量的脚本(rtdetr-l和rtdetr-x因为thop库的问题,没办法正常输出每一层的参数和计算量和时间)

3. val.py

    使用训练好的模型计算指标的脚本

4. detect.py

    推理的脚本

5. track.py

    跟踪推理的脚本

6. heatmap.py

    生成热力图的脚本

7. get\_FPS.py

    计算模型储存大小、模型推理时间、FPS的脚本

8. get\_COCO\_metrice.py

    计算COCO指标的脚本

9. plot\_result.py

    绘制曲线对比图的脚本

# **RT-DETR基准模型**

1. ultralytics/cfg/models/rt-detr/rtdetr-r18.yaml(有预训练权重)

    rtdetr-r18 summary: 421 layers, 20184464 parameters, 20184464 gradients, 58.6 GFLOPs

2. ultralytics/cfg/models/rt-detr/rtdetr-r34.yaml

    rtdetr-r34 summary: 501 layers, 30292624 parameters, 30292624 gradients, 88.9 GFLOPs

3. ultralytics/cfg/models/rt-detr/rtdetr-r50-m.yaml

    rtdetr-r50-m summary: 637 layers, 36647020 parameters, 36647020 gradients, 98.3 GFLOPs

4. ultralytics/cfg/models/rt-detr/rtdetr-r50.yaml(有预训练权重)

    rtdetr-r50 summary: 629 layers, 42944620 parameters, 42944620 gradients, 134.8 GFLOPs

5. ultralytics/cfg/models/rt-detr/rtdetr-r101.yaml

    rtdetr-r101 summary: 867 layers, 76661740 parameters, 76661740 gradients, 257.7 GFLOPs

6. ultralytics/cfg/models/rt-detr/rtdetr-l.yaml(有预训练权重)

    rtdetr-l summary: 673 layers, 32970732 parameters, 32970732 gradients, 108.3 GFLOPs

7. ultralytics/cfg/models/rt-detr/rtdetr-x.yaml(有预训练权重)

    rtdetr-x summary: 867 layers, 67468108 parameters, 67468108 gradients, 232.7 GFLOPs

# **RT-DETR改进方案**

#### **目前整合的一些注意力机制 还需要别的注意力机制可从[github](**https://github.com/z1069614715/objectdetection\_script/tree/master/cv-attention**)拉取对应的代码到ultralytics/nn/extra\_modules/attention.py即可. 视频教程可看项目视频中的(如何在yaml配置文件中添加注意力层)**

EMA, SimAM, SpatialGroupEnhance, BiLevelRoutingAttention, BiLevelRoutingAttention\_nchw, TripletAttention, CoordAtt, CBAM, BAMBlock, EfficientAttention(CloFormer中的注意力), LSKBlock, SEAttention, CPCA, deformable\_LKA, EffectiveSEModule, LSKA, SegNext\_Attention, DAttention(Vision Transformer with Deformable Attention CVPR2022), FocusedLinearAttention(ICCV2023), MLCA

### **以RT-DETR-R18为基准模型的改进方案**

1. ultralytics/cfg/models/rt-detr/rt-detr-timm.yaml

    使用[timm](https://github.com/huggingface/pytorch-image-models)库系列的主干替换rtdetr的backbone.(基本支持现有CNN模型)

2. ultralytics/cfg/models/rt-detr/rt-detr-fasternet.yaml

    使用[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)替换rtdetr的backbone.

3. ultralytics/cfg/models/rt-detr/rt-detr-EfficientViT.yaml

    使用[EfficientViT CVPR2023](https://github.com/microsoft/Cream/tree/main/EfficientViT)替换rtdetr的backbone.

4. ultralytics/cfg/models/rt-detr/rtdetr-convnextv2.yaml

    使用[ConvNextV2 2023](https://github.com/facebookresearch/ConvNeXt-V2)替换rtdetr的backbone.

5. ultralytics/cfg/models/rt-detr/rtdetr-EfficientFormerv2.yaml

    使用[EfficientFormerv2 2022](https://github.com/snap-research/EfficientFormer)替换rtdetr的backbone.

6. ultralytics/cfg/models/rt-detr/rtdetr-repvit.yaml

    使用[RepViT ICCV2023](https://github.com/THU-MIG/RepViT)替换rtdetr的backbone.

7. ultralytics/cfg/models/rt-detr/rtdetr-CSwomTramsformer.yaml

    使用[CSwinTramsformer CVPR2022](https://github.com/microsoft/CSWin-Transformer)替换rtdetr的backbone.

8. ultralytics/cfg/models/rt-detr/rtdetr-VanillaNet.yaml

    使用[VanillaNet 2023](https://github.com/huawei-noah/VanillaNet)替换rtdetr的backbone.

9. ultralytics/cfg/models/rt-detr/rtdetr-SwinTransformer.yaml

    使用[SwinTransformer ICCV2021](https://github.com/microsoft/Swin-Transformer)替换rtdetr的backbone.

10. ultralytics/cfg/models/rt-detr/rtdetr-lsknet.yaml

    使用[LSKNet ICCV2023](https://github.com/zcablii/LSKNet)替换rtdetr的backbone.

11. ultralytics/cfg/models/rt-detr/rtdetr-CascadedGroupAttention.yaml

    使用[EfficientViT CVPR2023](https://github.com/microsoft/Cream/tree/main/EfficientViT)中的CascadedGroupAttention改进rtdetr中的AIFI.(详细请看百度云视频-rtdetr-CascadedGroupAttention说明)

12. ultralytics/cfg/models/rt-detr/rtdetr-DWRC3.yaml

    使用[DWRSeg](https://arxiv.org/abs/2212.01173)中的Dilation-wise Residual(DWR)模块构建DWRC3改进rtdetr.

13. ultralytics/cfg/models/rt-detr/rtdetr-AIFI-LPE.yaml

    使用LearnedPositionalEncoding改进AIFI中的位置编码生成.(详细介绍请看百度云视频-20231119更新说明)

14. ultralytics/cfg/models/rt-detr/rtdetr-Ortho.yaml

    使用[OrthoNets](https://github.com/hady1011/OrthoNets/tree/main)中的正交通道注意力改进resnet18-backbone中的BasicBlock.(详细介绍请看百度云视频-20231119更新说明)

15. ultralytics/cfg/models/rt-detr/rtdetr-DCNV2.yaml

    使用可变形卷积DCNV2改进resnet18-backbone中的BasicBlock.

16. ultralytics/cfg/models/rt-detr/rtdetr-DCNV3.yaml

    使用可变形卷积[DCNV3 CVPR2023](https://github.com/OpenGVLab/InternImage)改进resnet18-backbone中的BasicBlock.(安装教程请看百度云视频-20231119更新说明)

17. ultralytics/cfg/models/rt-detr/rtdetr-DCNV2-Dynamic.yaml

    使用自研可变形卷积DCNV2-Dynamic改进resnet18-backbone中的BasicBlock.(详细介绍请看百度云视频-MPCA与DCNV2\_Dynamic的说明)

18. ultralytics/cfg/models/rt-detr/rtdetr-iRMB.yaml

    使用[EMO ICCV2023](https://github.com/zhangzjn/EMO)中的iRMB改进resnet18-backbone中的BasicBlock.(详细介绍请看百度云视频-20231119更新说明)

19. ultralytics/cfg/models/rt-detr/rtdetr-iRMB-Cascaded.yaml

    使用[EfficientViT CVPR2023](https://github.com/microsoft/Cream/tree/main/EfficientViT)中的CascadedGroupAttention对[EMO ICCV2023](https://github.com/zhangzjn/EMO)中的iRMB进行二次创新来改进resnet18-backbone中的BasicBlock.(详细介绍请看百度云视频-20231119更新说明)

20. ultralytics/cfg/models/rt-detr/rtdetr-attention.yaml

    添加注意力模块到resnet18-backbone中的BasicBlock中.(手把手教程请看百度云视频-手把手添加注意力教程)

21. ultralytics/cfg/models/rt-detr/rtdetr-p2.yaml

    添加小目标检测头P2到TransformerDecoderHead中.

22. ultralytics/cfg/models/rt-detr/rtdetr-DySnake.yaml

    添加[DySnakeConv](https://github.com/YaoleiQi/DSCNet)到resnet18-backbone中的BasicBlock中.

23. ultralytics/cfg/models/rt-detr/rtdetr-PConv.yaml

    使用[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的PConv改进resnet18-backbone中的BasicBlock.

24. ultralytics/cfg/models/rt-detr/rtdetr-PConv-Rep.yaml

    使用[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的PConv进行二次创新后改进resnet18-backbone中的BasicBlock.

25. ultralytics/cfg/models/rt-detr/rtdetr-Faster.yaml

    使用[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block改进resnet18-backbone中的BasicBlock.

26. ultralytics/cfg/models/rt-detr/rtdetr-Faster-Rep.yaml

    使用[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block进行二次创新后改进resnet18-backbone中的BasicBlock.

27. ultralytics/cfg/models/rt-detr/rtdetr-Faster-EMA.yaml

    使用[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block进行二次创新后改进resnet18-backbone中的BasicBlock.

28. ultralytics/cfg/models/rt-detr/rtdetr-Faster-Rep-EMA.yaml

    使用[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv和[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block进行二次创新后改进resnet18-backbone中的BasicBlock.

29. ultralytics/cfg/models/rt-detr/rtdetr-AKConv.yaml

    使用[AKConv 2023](https://github.com/CV-ZhangXin/AKConv)改进resnet18-backbone中的BasicBlock.

30. ultralytics/cfg/models/rt-detr/rtdetr-RFAConv.yaml

    使用[RFAConv 2023](https://github.com/Liuchen1997/RFAConv)改进resnet18-backbone中的BasicBlock.

31. ultralytics/cfg/models/rt-detr/rtdetr-RFCAConv.yaml

    使用[RFCAConv 2023](https://github.com/Liuchen1997/RFAConv)改进resnet18-backbone中的BasicBlock.

32. ultralytics/cfg/models/rt-detr/rtdetr-RFCBAMConv.yaml

    使用[RFCBAMConv 2023](https://github.com/Liuchen1997/RFAConv)改进resnet18-backbone中的BasicBlock.

33. ultralytics/cfg/models/rt-detr/rt-detr-unireplknet.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)替换rtdetr的backbone.

34. ultralytics/cfg/models/rt-detr/rtdetr-AIFI-DAttention.yaml

    使用[Vision Transformer with Deformable Attention CVPR2022](https://github.com/LeapLabTHU/DAT)中的DAttention改进AIFI.

35. ultralytics/cfg/models/rt-detr/rtdetr-Conv3XC.yaml

    使用[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的Conv3XC改进resnet18-backbone中的BasicBlock.

36. ultralytics/cfg/models/rt-detr/rtdetr-DRB.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock改进resnet18-backbone中的BasicBlock.

37. ultralytics/cfg/models/rt-detr/rtdetr-DWRC3-DRB.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock对[DWRSeg](https://arxiv.org/abs/2212.01173)中的Dilation-wise Residual(DWR)进行二次创新改进rtdetr.

38. ultralytics/cfg/models/rt-detr/rtdetr-Conv3XCC3.yaml

    使用[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的Conv3XC改进RepC3.

39. ultralytics/cfg/models/rt-detr/rtdetr-DRBC3.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock改进RepC3.

40. ultralytics/cfg/models/rt-detr/rtdetr-DBBC3.yaml

    使用[DiverseBranchBlock CVPR2021](https://github.com/DingXiaoH/DiverseBranchBlock)改进RepC3.

41. ultralytics/cfg/models/rt-detr/rtdetr-DBB.yaml

    使用[DiverseBranchBlock CVPR2021](https://github.com/DingXiaoH/DiverseBranchBlock)改进resnet18-backbone中的BasicBlock.

42. ultralytics/cfg/models/rt-detr/rtdetr-DualConv.yaml

    使用[DualConv](https://github.com/ChipsGuardian/DualConv)改进resnet18-backbone中的BasicBlock.

43. ultralytics/cfg/models/rt-detr/rtdetr-ASF.yaml

    使用[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion来改进rtdetr.

44. ultralytics/cfg/models/rt-detr/rtdetr-ASF-P2.yaml

    在ultralytics/cfg/models/rt-detr/rtdetr-ASF.yaml的基础上进行二次创新，引入P2检测层并对网络结构进行优化.

45. ultralytics/cfg/models/rt-detr/rtdetr-slimneck.yaml

    使用[SlimNeck](https://github.com/AlanLi1997/slim-neck-by-gsconv)中的VoVGSCSP\VoVGSCSPC和GSConv改进rtdetr中的CCFM.

46. ultralytics/cfg/models/rt-detr/rtdetr-slimneck-ASF.yaml

    使用[SlimNeck](https://github.com/AlanLi1997/slim-neck-by-gsconv)中的VoVGSCSP\VoVGSCSPC和GSConv和[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion改进rtdetr中的CCFM.

### **以RT-DETR-R50为基准模型的改进方案**

1. ultralytics/cfg/models/rt-detr/rtdetr-r50-Ortho.yaml

    使用[OrthoNets](https://github.com/hady1011/OrthoNets/tree/main)中的正交通道注意力改进resnet50-backbone中的BottleNeck.(详细介绍请看百度云视频-20231119更新说明)

2. ultralytics/cfg/models/rt-detr/rtdetr-r50-DCNV2.yaml

    使用可变形卷积DCNV2改进resnet50-backbone中的BottleNeck.

3. ultralytics/cfg/models/rt-detr/rtdetr-r50-DCNV3.yaml

    使用可变形卷积[DCNV3 CVPR2023](https://github.com/OpenGVLab/InternImage)改进resnet50-backbone中的BottleNeck.(安装教程请看百度云视频-20231119更新说明)

4. ultralytics/cfg/models/rt-detr/rtdetr-r50-DCNV2-Dynamic.yaml

    使用自研可变形卷积DCNV2-Dynamic改进resnet50-backbone中的BottleNeck.(详细介绍请看百度云视频-MPCA与DCNV2\_Dynamic的说明)

5. ultralytics/cfg/models/rt-detr/rtdetr-r50-iRMB.yaml

    使用[EMO ICCV2023](https://github.com/zhangzjn/EMO)中的iRMB改进resnet50-backbone中的BottleNeck.(详细介绍请看百度云视频-20231119更新说明)

6. ultralytics/cfg/models/rt-detr/rtdetr-r50-iRMB-Cascaded.yaml

    使用[EfficientViT CVPR2023](https://github.com/microsoft/Cream/tree/main/EfficientViT)中的CascadedGroupAttention对[EMO ICCV2023](https://github.com/zhangzjn/EMO)中的iRMB进行二次创新来改进resnet50-backbone中的BottleNeck.(详细介绍请看百度云视频-20231119更新说明)

7. ultralytics/cfg/models/rt-detr/rtdetr-r50-attention.yaml

    添加注意力模块到resnet50-backbone中的BottleNeck.(手把手教程请看百度云视频-手把手添加注意力教程)

8. ultralytics/cfg/models/rt-detr/rtdetr-r50-DySnake.yaml

    添加[DySnakeConv](https://github.com/YaoleiQi/DSCNet)到resnet50-backbone中的BottleNeck中.

9. ultralytics/cfg/models/rt-detr/rtdetr-r50-PConv.yaml

    使用[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的PConv改进resnet50-backbone中的BottleNeck.

10. ultralytics/cfg/models/rt-detr/rtdetr-r50-PConv-Rep.yaml

    使用[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的PConv进行二次创新后改进resnet50-backbone中的BottleNeck.

11. ultralytics/cfg/models/rt-detr/rtdetr-r50-Faster.yaml

    使用[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block改进resnet50-backbone中的BottleNeck.

12. ultralytics/cfg/models/rt-detr/rtdetr-r50-Faster-Rep.yaml

    使用[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block进行二次创新后改进resnet50-backbone中的BottleNeck.

13. ultralytics/cfg/models/rt-detr/rtdetr-r50-Faster-EMA.yaml

    使用[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block进行二次创新后改进resnet50-backbone中的BottleNeck.

14. ultralytics/cfg/models/rt-detr/rtdetr-r50-Faster-Rep-EMA.yaml

    使用[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv和[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)对[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block进行二次创新后改进resnet50-backbone中的BottleNeck.

15. ultralytics/cfg/models/rt-detr/rtdetr-r50-AKConv.yaml

    使用[AKConv 2023](https://github.com/CV-ZhangXin/AKConv)改进resnet50-backbone中的BottleNeck.

16. ultralytics/cfg/models/rt-detr/rtdetr-r50-RFAConv.yaml

    使用[RFAConv 2023](https://github.com/Liuchen1997/RFAConv)改进resnet50-backbone中的BottleNeck.

17. ultralytics/cfg/models/rt-detr/rtdetr-r50-RFCAConv.yaml

    使用[RFCAConv 2023](https://github.com/Liuchen1997/RFAConv)改进resnet50-backbone中的BottleNeck.

18. ultralytics/cfg/models/rt-detr/rtdetr-r50-RFCBAMConv.yaml

    使用[RFCBAMConv 2023](https://github.com/Liuchen1997/RFAConv)改进resnet50-backbone中的BottleNeck.

19. ultralytics/cfg/models/rt-detr/rtdetr-r50-Conv3XC.yaml

    使用[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的Conv3XC改进resnet50-backbone中的BottleNeck.

20. ultralytics/cfg/models/rt-detr/rtdetr-r50-DRB.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock改进resnet50-backbone中的BottleNeck.

21. ultralytics/cfg/models/rt-detr/rtdetr-r50-DBB.yaml

    使用[DiverseBranchBlock CVPR2021](https://github.com/DingXiaoH/DiverseBranchBlock)改进resnet50-backbone中的BottleNeck.

22. ultralytics/cfg/models/rt-detr/rtdetr-r50-DualConv.yaml

    使用[DualConv](https://github.com/ChipsGuardian/DualConv)改进resnet50-backbone中的BottleNeck.

23. ultralytics/cfg/models/rt-detr/rtdetr-r50-ASF.yaml

    使用[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion来改进rtdetr.

### **以RT-DETR-L为基准模型的改进方案**

1. ultralytics/cfg/models/rt-detr/rtdetr-l-GhostHGNetV2.yaml

    使用GhostConv改进HGNetV2.(详细介绍请看百度云视频-20231109更新说明)

2. ultralytics/cfg/models/rt-detr/rtdetr-l-RepHGNetV2.yaml

    使用RepConv改进HGNetV2.(详细介绍请看百度云视频-20231109更新说明)

3. ultralytics/cfg/models/rt-detr/rtdetr-l-attention.yaml

    添加注意力模块到HGBlock中.(手把手教程请看百度云视频-手把手添加注意力教程)

### **以Yolov8为基准模型的改进方案**

1. ultralytics/cfg/models/yolo-detr/yolov8-detr.yaml

    使用RT-DETR中的TransformerDecoderHead改进yolov8.

2. ultralytics/cfg/models/yolo-detr/yolov8-detr-DWR.yaml

    使用RT-DETR中的TransformerDecoderHead和[DWRSeg](https://arxiv.org/abs/2212.01173)中的Dilation-wise Residual(DWR)模块改进yolov8.

3. ultralytics/cfg/models/yolo-detr/yolov8-detr-fasternet.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)改进yolov8.(支持替换其他主干,请看百度云视频-替换主干示例教程)

4. ultralytics/cfg/models/yolo-detr/yolov8-detr-AIFI-LPE.yaml

    使用RT-DETR中的TransformerDecoderHead和LearnedPositionalEncoding改进yolov8.(详细介绍请看百度云视频-20231119更新说明)

5. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-DCNV2.yaml

    使用RT-DETR中的TransformerDecoderHead和可变形卷积DCNV2改进yolov8.

6. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-DCNV3.yaml

    使用RT-DETR中的TransformerDecoderHead和可变形卷积[DCNV3 CVPR2023](https://github.com/OpenGVLab/InternImage)改进yolov8.(安装教程请看百度云视频-20231119更新说明)

7. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-DCNV2-Dynamic.yaml

    使用RT-DETR中的TransformerDecoderHead和自研可变形卷积DCNV2-Dynamic改进yolov8.(详细介绍请看百度云视频-MPCA与DCNV2\_Dynamic的说明)

8. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-Ortho.yaml

    使用RT-DETR中的TransformerDecoderHead和[OrthoNets](https://github.com/hady1011/OrthoNets/tree/main)中的正交通道注意力改进yolov8.(详细介绍请看百度云视频-20231119更新说明)

9. ultralytics/cfg/models/yolo-detr/yolov8-detr-attention.yaml

    添加注意力到基于RTDETR-Head中的yolov8中.(手把手教程请看百度云视频-手把手添加注意力教程)

10. ultralytics/cfg/models/yolo-detr/yolov8-detr-p2.yaml

    添加小目标检测头P2到TransformerDecoderHead中.

11. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-DySnake.yaml

    [DySnakeConv](https://github.com/YaoleiQi/DSCNet)与C2f融合.

12. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-Faster.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block改进yolov8.

13. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-Faster-Rep.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中与[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv二次创新后的Faster-Block-Rep改进yolov8.

14. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-Faster-EMA.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中与[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)二次创新后的Faster-Block-EMA的Faster-Block-EMA改进yolov8.

15. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-Faster-Rep-EMA.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中与[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv、[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)二次创新后的Faster-Block改进yolov8.

16. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-AKConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[AKConv 2023](https://github.com/CV-ZhangXin/AKConv)改进yolov8.

17. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-RFAConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[RFAConv 2023](https://github.com/Liuchen1997/RFAConv)改进yolov8.

18. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-RFAConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[RFCAConv 2023](https://github.com/Liuchen1997/RFAConv)改进yolov8.

19. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-RFAConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[RFCBAMConv 2023](https://github.com/Liuchen1997/RFAConv)改进yolov8.

20. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-Conv3XC.yaml

    使用RT-DETR中的TransformerDecoderHead和[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的Conv3XC改进yolov8.

21. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-SPAB.yaml

    使用RT-DETR中的TransformerDecoderHead和[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的SPAB改进yolov8.

22. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-DRB.yaml

    使用RT-DETR中的TransformerDecoderHead和[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock改进yolov8.

23. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-UniRepLKNetBlock.yaml

    使用RT-DETR中的TransformerDecoderHead和[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的UniRepLKNetBlock改进yolov8.

24. ultralytics/cfg/models/yolo-detr/yolov8-detr-DWR-DRB.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock对[DWRSeg](https://arxiv.org/abs/2212.01173)中的Dilation-wise Residual(DWR)进行二次创新改进yolov8.

25. ultralytics/cfg/models/yolo-detr/yolov8-detr-C2f-DBB.yaml

    使用RT-DETR中的TransformerDecoderHead和[DiverseBranchBlock CVPR2021](https://github.com/DingXiaoH/DiverseBranchBlock)改进yolov8.

26. ultralytics/cfg/models/yolo-detr/yolov8-detr-CSP-EDLAN.yaml

    使用RT-DETR中的TransformerDecoderHead和[DualConv](https://github.com/ChipsGuardian/DualConv)打造CSP Efficient Dual Layer Aggregation Networks改进yolov8.

27. ultralytics/cfg/models/yolo-detr/yolov8-detr-ASF.yaml

    使用RT-DETR中的TransformerDecoderHead和[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion改进yolov8.

28. ultralytics/cfg/models/yolo-detr/yolov8-detr-ASF-P2.yaml

    在ultralytics/cfg/models/yolo-detr/yolov8-detr-ASF.yaml的基础上进行二次创新，引入P2检测层并对网络结构进行优化.

29. ultralytics/cfg/models/yolo-detr/yolov8-detr-slimneck.yaml

    使用RT-DETR中的TransformerDecoderHead和[SlimNeck](https://github.com/AlanLi1997/slim-neck-by-gsconv)中VoVGSCSP\VoVGSCSPC和GSConv改进yolov8的neck.

30. ultralytics/cfg/models/yolo-detr/yolov8-detr-slimneck-asf.yaml

    在ultralytics/cfg/models/yolo-detr/yolov8-detr-slimneck.yaml使用[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion进行二次创新.

### **以Yolov5为基准模型的改进方案**

1. ultralytics/cfg/models/yolo-detr/yolov5-detr.yaml

    使用RT-DETR中的TransformerDecoderHead改进yolov5.

2. ultralytics/cfg/models/yolo-detr/yolov5-detr-DWR.yaml

    使用RT-DETR中的TransformerDecoderHead和[DWRSeg](https://arxiv.org/abs/2212.01173)中的Dilation-wise Residual(DWR)模块改进yolov5.

3. ultralytics/cfg/models/yolo-detr/yolov5-detr-fasternet.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)改进yolov5.(支持替换其他主干,请看百度云视频-替换主干示例教程)

4. ultralytics/cfg/models/yolo-detr/yolov5-detr-AIFI-LPE.yaml

    使用RT-DETR中的TransformerDecoderHead和LearnedPositionalEncoding改进yolov5.(详细介绍请看百度云视频-20231119更新说明)

5. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-DCNV2.yaml

    使用RT-DETR中的TransformerDecoderHead和可变形卷积DCNV2改进yolov5.

6. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-DCNV3.yaml

    使用RT-DETR中的TransformerDecoderHead和可变形卷积[DCNV3 CVPR2023](https://github.com/OpenGVLab/InternImage)改进yolov5.(安装教程请看百度云视频-20231119更新说明)

7. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-DCNV2-Dynamic.yaml

    使用RT-DETR中的TransformerDecoderHead和自研可变形卷积DCNV2-Dynamic改进yolov5.(详细介绍请看百度云视频-MPCA与DCNV2\_Dynamic的说明)

8. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-Ortho.yaml(详细介绍请看百度云视频-20231119更新说明)

    使用RT-DETR中的TransformerDecoderHead和[OrthoNets](https://github.com/hady1011/OrthoNets/tree/main)中的正交通道注意力改进yolov5.

9. ultralytics/cfg/models/yolo-detr/yolov5-detr-attention.yaml

    添加注意力到基于RTDETR-Head中的yolov5中.(手把手教程请看百度云视频-手把手添加注意力教程)

10. ultralytics/cfg/models/yolo-detr/yolov5-detr-p2.yaml

    添加小目标检测头P2到TransformerDecoderHead中.

11. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-DySnake.yaml

    [DySnakeConv](https://github.com/YaoleiQi/DSCNet)与C3融合.

12. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-Faster.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中的Faster-Block改进yolov5.

13. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-Faster-Rep.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中与[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv二次创新后的Faster-Block-Rep改进yolov5.

14. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-Faster-EMA.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中与[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)二次创新后的Faster-Block-EMA的Faster-Block-EMA改进yolov5.

15. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-Faster-Rep-EMA.yaml

    使用RT-DETR中的TransformerDecoderHead和[FasterNet CVPR2023](https://github.com/JierunChen/FasterNet)中与[RepVGG CVPR2021](https://github.com/DingXiaoH/RepVGG)中的RepConv、[EMA ICASSP2023](https://arxiv.org/abs/2305.13563v1)二次创新后的Faster-Block改进yolov5.

16. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-AKConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[AKConv 2023](https://github.com/CV-ZhangXin/AKConv)改进yolov5.

17. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-RFAConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[RFAConv 2023](https://github.com/Liuchen1997/RFAConv)改进yolov5.

18. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-RFAConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[RFCAConv 2023](https://github.com/Liuchen1997/RFAConv)改进yolov5.

19. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-RFAConv.yaml

    使用RT-DETR中的TransformerDecoderHead和[RFCBAMConv 2023](https://github.com/Liuchen1997/RFAConv)改进yolov5.

20. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-Conv3XC.yaml

    使用RT-DETR中的TransformerDecoderHead和[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的Conv3XC改进yolov5.

21. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-SPAB.yaml

    使用RT-DETR中的TransformerDecoderHead和[Swift Parameter-free Attention Network](https://github.com/hongyuanyu/SPAN/tree/main)中的SPAB改进yolov5.

22. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-DRB.yaml

    使用RT-DETR中的TransformerDecoderHead和[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock改进改进yolov5.

23. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-UniRepLKNetBlock.yaml

    使用RT-DETR中的TransformerDecoderHead和[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的UniRepLKNetBlock改进改进yolov5.

24. ultralytics/cfg/models/yolo-detr/yolov5-detr-DWR-DRB.yaml

    使用[UniRepLKNet](https://github.com/AILab-CVC/UniRepLKNet/tree/main)中的DilatedReparamBlock对[DWRSeg](https://arxiv.org/abs/2212.01173)中的Dilation-wise Residual(DWR)进行二次创新改进yolov5.

25. ultralytics/cfg/models/yolo-detr/yolov5-detr-C3-DBB.yaml

    使用RT-DETR中的TransformerDecoderHead和[DiverseBranchBlock CVPR2021](https://github.com/DingXiaoH/DiverseBranchBlock)改进yolov5.

26. ultralytics/cfg/models/yolo-detr/yolov5-detr-CSP-EDLAN.yaml

    使用RT-DETR中的TransformerDecoderHead和[DualConv](https://github.com/ChipsGuardian/DualConv)打造CSP Efficient Dual Layer Aggregation Networks改进yolov5.

27. ultralytics/cfg/models/yolo-detr/yolov5-detr-ASF.yaml

    使用RT-DETR中的TransformerDecoderHead和[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion改进yolov5.

28. ultralytics/cfg/models/yolo-detr/yolov5-detr-ASF-P2.yaml

    在ultralytics/cfg/models/yolo-detr/yolov5-detr-ASF.yaml的基础上进行二次创新，引入P2检测层并对网络结构进行优化.

29. ultralytics/cfg/models/yolo-detr/yolov5-detr-slimneck.yaml

    使用RT-DETR中的TransformerDecoderHead和[SlimNeck](https://github.com/AlanLi1997/slim-neck-by-gsconv)中VoVGSCSP\VoVGSCSPC和GSConv改进yolov5的neck.

30. ultralytics/cfg/models/yolo-detr/yolov5-detr-slimneck-asf.yaml

    在ultralytics/cfg/models/yolo-detr/yolov5-detr-slimneck.yaml使用[ASF-YOLO](https://github.com/mkang315/ASF-YOLO)中的Attentional Scale Sequence Fusion进行二次创新.

### **IoU系列**

更换教程可看(百度云视频-20231125更新说明)

1. IoU,GIoU,DIoU,CIoU,EIoU,SIoU

2. MPDIoU[论文链接](https://arxiv.org/pdf/2307.07662.pdf)

3. Inner-IoU,Inner-GIoU,Inner-DIoU,Inner-CIoU,Inner-EIoU,Inner-SIoU[论文链接](https://arxiv.org/abs/2311.02877)

4. Inner-MPDIoU(利用Inner-Iou与MPDIou进行二次创新)

5. Normalized Gaussian Wasserstein Distance.[论文链接](https://arxiv.org/abs/2110.13389)

# **怎么像yolov5那样输出每一层的参数,计算量？**

参考main\_profile.py,选择自己的配置文件路径即可

# **如何替换主干？**

可以看项目视频-替换主干示例教程.

# **如何绘制曲线对比图?**

在plot\_result.py中的names指定runs/train中的训练结果名字name即可.

比如目前runs/train中有exp,exp1,exp2这三个文件夹,plot\_result.py中names中的值为:['exp', 'exp1', 'exp2'],运行后会自动保存为metrice\_curve.png和loss\_curve.png在当前运行的目录下.

# **如何计算COCO指标?**

可以看项目视频-计算COCO指标教程.

python dataset/yolo2coco.py --image\_path dataset/images/test --label\_path dataset/labels/test

python get\_COCO\_metrice.py --pred\_json runs/val/exp/predictions.json --anno\_json data.json

新旧版的差异就在于 predictions.json的生成方式,新版就是在val.py中把save\_json设置为True即可

# **常见错误和解决方案(如果是跑自带的一些配置文件报错可以先看看第十大点对应的配置文件是否有提示需要修改内容)**

1. RuntimeError: xxxxxxxxxxx does not have a deterministic implementation, but you set 'torch.use\_deterministic\_algorithms(True)'.....

    解决方案：在ultralytics/utils/torch\_utils.py中init\_seeds函数中把torch.use\_deterministic\_algorithms里面的True改为False

2. ModuleNotFoundError：No module named xxx

    解决方案：缺少对应的包，先把YOLOV8环境配置的安装命令进行安装一下，如果还是缺少显示缺少包，安装对应的包即可(xxx就是对应的包).

3. OMP: Error #15: Initializing libiomp5md.dll, but found libiomp5md.dll already initialized.

    解决方案：https://zhuanlan.zhihu.com/p/599835290

<a *id*="a"></a>

4. 多卡训练问题.[参考链接](https://docs.ultralytics.com/yolov5/tutorials/multi\_gpu\_training/#multi-gpu-dataparallel-mode-not-recommended:~:text=just%201%20GPU.-,Multi%2DGPU%20DistributedDataParallel%20Mode%20(%E2%9C%85%20recommended),-You%20will%20have)

    python -m torch.distributed.run --nproc\_per\_node 2 train.py

5. 指定显卡训练.

    1. 使用device参数进行指定.

    2. 参考链接:https://blog.csdn.net/m0\_55097528/article/details/130323125, 简单来说就是用这个来代替device参数.

6. ValueError: Expected more than 1 value per channel when training, got input size torch.Size...

    如果是在训练情况下的验证阶段出现的话,大概率就是最后一个验证的batch为1,这种情况只需要把验证集多一张或者少一张即可,或者变更batch参数.

7. AttributeError: Can't pickle local object 'EMASlideLoss.**\_\_init\_\_**.<locals>.<lambda>'

    可以在ultralytics/utils/loss.py中添加import dill as pickle,然后装一下dill这个包.

    pip install dill -i https://pypi.tuna.tsinghua.edu.cn/simple

8. RuntimeError: Dataset 'xxxxx' error ❌

    将data.yaml中的路径都改为绝对路径.

# **常见疑问**

1. After Fuse指的是什么？

    Fuse是指模型的一些模块进行融合,最常见的就是conv和bn层进行融合,在训练的时候模型是存在conv和bn的,但在推理的过程中,模型在初始化的时候会进行模型fuse,把其中的conv和bn进行融合,通过一些数学转换把bn层融合到conv里面,还有一些例如DBB,RepVGG等等模块支持融合的,这些在fuse阶段都会进行融合,融合后可以一般都可以得到比融合前更快的推理速度,而且基本不影响精度.

2. FPS如何计算？

    在运行val.py后最后会出来Speed: 0.1ms preprocess, 5.4ms inference, 0.0ms loss, 0.4ms postprocess per image这行输出,这行输出就代表了每张图的前处理,推理,loss,后处理的时间,当然在val.py过程中是不需要计算loss的,所以为0,FPS最严谨来说就是1000(1s)/(preprocess+inference+postprocess),没那么严谨的话就是只除以inference的时间,还有一个问题就是batchsize应该设置为多少,其实这行输出就已经是每张图的时间了,但是batchsize还是会对这个时间有所影响,主要是关于并行处理的问题,GPU中可以一次处理多个batch的数据,也可以只处理一个数据,但是处理多batch的数据比处理一个数据的时候整体速度要快,举个例子,比如我有1000张图,我分别设置batchsize为32和batchsize为1,整体运行的时间百分之99都是batchsize为32的快,因此这就导致不同batch输出的时间不同,至于该设置多少来计算FPS,貌似众说纷纭,所以这里我也不好给意见.

    附上yolov5作者对于FPS和Batch的一个实验链接: https://github.com/ultralytics/yolov5/discussions/6649

3. 训练的时候出现两次结构打印是什么情况?

    第一次打印的和第二次打印的主要不同地方就是类别数,第一次打印的是yaml配置文件中的nc参数的类别数的结构,第二次打印的是你实际数据集类别数的结构,其差异就在类别数,实际使用的是第二次打印的结构.