

【RT-DETR有效改进】利用MobileNetV3替换Backbone (轻量化网络结构, 提点)

发布时间: 2025-10-18 12:07:22

前言

大家好, 这里是RT-DETR有效涨点专栏。

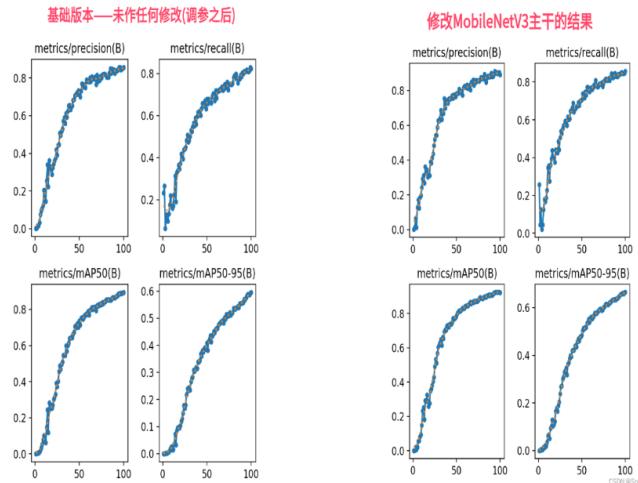
本专栏的内容根据ultralytics版本的RT-DETR进行改进, 内容持续更新, 每周更新文章数量3-10篇。

专栏以ResNet18、ResNet50为基础修改版本, 同时修改内容也支持ResNet32、ResNet101和PPHGNet版本, 其中ResNet为RT-DETR官方版本1: 1移植过来的, 参数量基本保持一致(误差很小很小), 不同于ultralytics仓库版本的ResNet官方版本, 同时ultralytics仓库的一些参数是和RT-DETR相冲的所以我也是会教大家调好一些参数和代码, 真正意义上的跟ultralytics的和RT-DETR官方版本的无区别。

✿ 欢迎大家订阅本专栏, 一起学习RT-DETR ✿

一、本文介绍

本文给大家带来的改进机制是**MobileNetV3**, 其主要改进思想集中在结合硬件感知的网络架构搜索 (NAS) 和 NetAdapt算法, 以优化移动设备CPU上的性能。它采用了新颖的架构设计, 包括反转残差结构和线性瓶颈层, 以及新的高效分割解码器Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP), 以提升在移动分类、检测和分割任务上的表现。实验表明, MobileNets在资源和准确性的权衡方面表现出色, 并在多种应用 (如对象检测、细粒度分类、面部属性识别和大规模地理定位) 中展现了其有效性。可以看出其mAP增加了大概三四个点, 同时参数量却下降了百分之四十以上!



专栏链接: RT-DETR剑指论文专栏, 持续复现各种顶会内容——论文收割机RT-DETR

目录

- 一、本文介绍
- 二、MobileNetV3的框架原理
 - 2.1 NAS和NetAdapt算法
 - 2.2 反转残差结构和线性瓶颈层
- 三、MobileNetV3的核心代码
- 四、手把手教你添加MobileNetV3网络结构
 - 4.1 修改一
 - 4.2 修改二

最新文章

所有文章

Redis缓存: 热点数据查询的数据库
减压策略

《超级马里奥派对 空前盛会》NS2
版本预告片发布 新功能演示!

2

古代数与现代方法的计算差别对比

日常运维: 11G RAC环境GRID目录
及文件权限被篡改的修复

看行情的网站有哪些 简单明了易操作

2025-07-03: 使字符频率相等的最
少操作次数. 用go语言, 给定一个
字符串 s, 如果某个字符串 t 中所
有字符的出现次数相

2025年路沿石厂家 TOP 企业品牌
排行榜, 五莲花 / 五莲红 / 五
莲灰 / 芝麻灰 / 芝麻白 / 芝麻黑 /
黄锈石 / 黄金麻 / 湛海沙 / 白麻路
沿石公司推荐!

Mac玩《枪火重生》攻略, 教你如
何在苹果电脑上运行《枪火重
生》!

ABoVE/ASCENDS: 合并大气二氧
化碳、甲烷和气象数据, 2017

推荐阅读

c语言读写操作-C语言中的文件读写
操作及其在实际项目中的应用

环境变量设置文件在软件开发中的
应用与实践

仿真出激活码-利用深度学习技术实
现高效仿真出激活码的方法

手机模拟鼠标-在手机上模拟鼠标操
作的实用技巧与最佳实践

amd驱动程序-探索AMD驱动程
序: 优化性能与稳定性的关键

c和c 字符串-C与C++面向对象编程
的对比与融合实践

阿里巴巴CRM系统: 提高客户关系
管理效率的关键工具

电子邮件的格式-电子邮件格式与规
范

4.3 修改三
4.4 修改四
4.5 修改五
4.6 修改六
4.7 修改七
4.8 修改八
4.9 RT-DETR不能打印计算量问题的解决
4.10 可选修改

五、MobileNetV3的yaml文件

5.1 yaml文件
5.2 运行文件
5.3 成功训练截图

六、全文总结

sql语言是具有 功能-SQL语言在IT领域的重要功能与应用

html before after 使用JavaScript
模块化提升应用程序性能与可维护性

二、MobileNetV3的框架原理

Searching for MobileNetV3

Andrew Howard¹ Mark Sandler¹ Grace Chu¹ Liang-Chieh Chen¹ Bo Chen¹ Mingxing Tan²
Weijun Wang¹ Yukun Zhu¹ Ruoming Pang² Vijay Vasudevan² Quoc V. Le² Hartwig Adam¹

¹Google AI, ²Google Brain

{howarda, sandler, cxy, lcchen, bochen, tammingxing, weijunw, yukun, rpanq, vrv, qvl, hadam}@google.com

CODN(F9n7T)

官方论文地址：官方论文地址点击即可跳转

官方代码地址：官方代码地址

摘要：我们提出了下一代 MobileNets，它基于一系列互补的搜索技术以及新颖的架构设计。MobileNetV3 通过结合硬件感知的网络架构搜索 (NAS) 以及 NetAdapt 算法进行优化，适应移动电话 CPU，并通过新颖的架构进步进行了改进。本文探索了自动化搜索算法和网络设计如何共同利用互补方法来提升整体技术水平。通过这一过程，我们发布了两个新的 MobileNet 模型：MobileNetV3-Large 和 MobileNetV3-Small，分别针对高资源和低资源使用场景。然后将这些模型适配并应用于目标检测和语义分割任务。对于语义分割（或任何密集像素预测）任务，我们提出了一种新的高效分割解码器 Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP)。我们在移动分类、检测和分割方面实现了新的最佳成绩，与 MobileNetV2 相比，MobileNetV3-Large 在 ImageNet 分类的准确度上提高了 3.2%，同时减少了 20% 的延迟。MobileNetV3-Small 与具有相似延迟的 MobileNetV2 模型相比，准确度提高了 6.6%。MobileNetV3-Large 检测在 COCO 检测上的速度比 MobileNetV2 快 25% 以上，准确度大致相同。在 Cityscapes 分割任务中，MobileNetV3-Large LR-ASPP 比 MobileNetV2 R-ASPP 快 34%，准确度相似。

CODN(F9n7T)

MobileNetV3的主要改进思想集中在结合硬件感知的网络架构搜索 (NAS) 和NetAdapt算法，以优化移动设备CPU上的性能。它采用了新颖的架构设计，包括反转残差结构和线性瓶颈层，以及新的高效分割解码器Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP)，以提升在移动分类、检测和分割任务上的表现。这些改进通过精心设计的轻量级架构，实现了更高的准确度、更低的延迟，并在不同的资源使用场景中实现了更好的性能。

MobileNetV3的主要创新点包括：

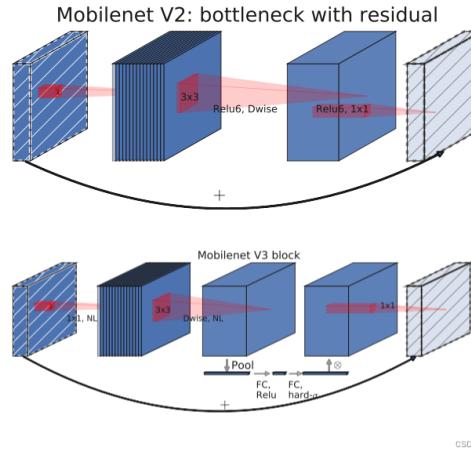
1. 结合了硬件感知的网络架构搜索 (NAS) 和NetAdapt算法，针对移动设备CPU进行优化。
2. 引入了新颖的架构设计，包括反转残差结构和线性瓶颈层。
3. 提出了高效的Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP) 作为新的分割解码器。

2.1 NAS和NetAdapt算法

MobileNetV3采用了硬件感知的网络架构搜索 (NAS) 和NetAdapt算法，这两种技术相互补充，可以结合起来有效地为特定硬件平台找到优化的模型。特别是，它采用了平台感知NAS进行块级搜索，类似于之前的MnasNet-A1方法，使用相同的基于RNN的控制器和相同的分解层次搜索空间，以便为大型移动模型找到全局网络结构，目标是大约80ms的延迟。然后在此基础上应用NetAdapt算法和其他优化措施。这种方法允许在顺序方式中对单个层进行微调，而不是尝试推断粗略但全局的架构。NetAdapt的第一个技术是用于层级搜索，它更适用于小型移动模型，因为对于小型模型来说，准确性随着延迟的变化更加显著，因此需要一个较小的权重因子 $w = -0.15$ 来补偿不同延迟下的较大概准确性。通过这个新的权重因子，我们从头开始一个新的架构搜索，以找到初始种子模型，然后应用NetAdapt和其他

2.2 反转残差结构和线性瓶颈层

MobileNetV3在架构上进行了一些修改，以降低某些较慢层的延迟，同时保持准确性。这些修改超出了当前搜索空间的范围。第一项修改重新设计了网络的最后几层是如何相互作用以更有效地生成最终特征的。基于MobileNetV2的反转瓶颈结构的当前模型在最终层使用 1×1 卷积以扩展到更高维的特征空间。这一层对于预测中拥有丰富的特征至关重要。然而，这也增加了额外的延迟。为了减少延迟并保留高维特征，我们将这一层移到最后的平均池化之后。



CSDN @Snu77

上图展示了MobileNetV2和MobileNetV3的网络结构层。

上侧 (MobileNetV2层): 展示了反转残差和线性瓶颈结构。每个块由狭窄的输入和输出层组成，这些层没有非线性操作，后面跟着扩展到更高维空间并投影到输出的操作。残差连接连接了瓶颈层，而不是扩展层。

下侧 (MobileNetV2 + Squeeze-and-Excite): 展示了与Squeeze-and-Excite层一起使用的MobileNetV3。与先前不同，在残差层中应用了挤压和激励操作。

三、MobileNetV3的核心代码

下面的代码是整个MobileNetV1的核心代码，大家如果想学习可以和上面的框架原理对比着看一看估计会有一些收获，使用方式看章节四。

```
"""A from-scratch implementation of MobileNetV3 paper ( for educational purposes ).

Paper
    Searching for MobileNetV3 - https://arxiv.org/abs/1905.02244v5

author : shubham.aiengineer@gmail.com
"""

import torch
from torch import nn
from torchsummary import summary

class SqueezeExcitationBlock(nn.Module):
    def __init__(self, in_channels: int):
        """Constructor for SqueezeExcitationBlock.

        Args:
            in_channels (int): Number of input channels.
        """
        super().__init__()

        self.pool1 = nn.AdaptiveAvgPool2d(1)
        self.linear1 = nn.Linear(in_channels // 4
                               # divide by 4 as mentioned in the paper, 5.3. Large squeeze-and-excite
                               )
        self.act1 = nn.ReLU()
        self.linear2 = nn.Linear(in_channels // 4, in_channels)
        self.act2 = nn.Hardsigmoid()

    def forward(self, x):
        """Forward pass for SqueezeExcitationBlock."""
        identity = x

        x = self.pool1(x)
        x = torch.flatten(x, 1)
        x = self.linear1(x)
        x = self.act1(x)
        x = self.linear2(x)
        x = self.act2(x)

        x = identity * x[:, :, None, None]
        return x

class ConvNormActivationBlock(nn.Module):
    def __init__(_

```

```

    self,
    in_channels: int,
    out_channels: int,
    kernel_size: list,
    stride: int = 1,
    padding: int = 0,
    groups: int = 1,
    bias: bool = False,
    activation: torch.nn = nn.Hardswish,
):
    """Constructs a block containing a convolution, batch normalization and activation layer

Args:
    in_channels (int): number of input channels
    out_channels (int): number of output channels
    kernel_size (list): size of the convolutional kernel
    stride (int, optional): stride of the convolutional kernel. Defaults to 1.
    padding (int, optional): padding of the convolutional kernel. Defaults to 0.
    groups (int, optional): number of groups for depthwise separable convolution. Defaults to 1.
    bias (bool, optional): whether to use bias. Defaults to False.
    activation (torch.nn, optional): activation function. Defaults to nn.Hardswish.

"""
super().__init__()

self.conv = nn.Conv2d(
    in_channels,
    out_channels,
    kernel_size,
    stride=stride,
    padding=padding,
    groups=groups,
    bias=bias,
)
self.norm = nn.BatchNorm2d(out_channels)
self.activation = activation()

def forward(self, x):
    """Perform forward pass."""
    x = self.conv(x)
    x = self.norm(x)
    x = self.activation(x)

    return x

class InverseResidualBlock(nn.Module):
    def __init__(_
        self,
        in_channels: int,
        out_channels: int,
        kernel_size: int,
        expansion_size: int = 6,
        stride: int = 1,
        squeeze_exitation: bool = True,
        activation: nn.Module = nn.Hardswish,
):
        """Constructs a inverse residual block

Args:
    in_channels (int): number of input channels
    out_channels (int): number of output channels
    kernel_size (int): size of the convolutional kernel
    expansion_size (int, optional): size of the expansion factor. Defaults to 6.
    stride (int, optional): stride of the convolutional kernel. Defaults to 1.
    squeeze_exitation (bool, optional): whether to add squeeze and exitation block or not. Defaults to True.
    activation (nn.Module, optional): activation function. Defaults to nn.Hardswish.

"""
        super().__init__()

        self.residual = in_channels == out_channels and stride == 1
        self.squeeze_exitation = squeeze_exitation

        self.conv1 = (
            ConvNormActivationBlock(
                in_channels, expansion_size, (1, 1), activation=activation
            )
            if in_channels != expansion_size
            else nn.Identity()
        ) # If it's not the first layer, then we need to add a 1x1 convolutional layer to expand the number of channels
        self.se = SqueezeExcitationBlock(expansion_size, expansion_size, (kernel_size, kernel_size), stride=stride, padding=kernel_size // 2, groups=expansion_size // 2, activation=activation, )
        if self.squeeze_exitation:
            self.se = SqueezeExcitationBlock(expansion_size)

        self.conv2 = nn.Conv2d(
            expansion_size, out_channels, (1, 1), bias=False
        ) # bias is false because we are using batch normalization, which already has bias
        self.norm = nn.BatchNorm2d(out_channels)

    def forward(self, x):
        """Perform forward pass."""
        identity = x

        x = self.conv1(x)
        x = self.depthwise_conv(x)

        if self.squeeze_exitation:
            x = self.se(x)

        x = self.conv2(x)
        x = self.norm(x)

        if self.residual:
            x = x + identity

        return x

class MobileNetV3(nn.Module):
    def __init__(_
        self,
        num_classes: int = 1000,
        input_channel: int = 3,
        config: str = "large",
        dropout: float = 0.8,
):
        """Constructs MobileNetV3 architecture

Args:
    num_classes: An integer count of output neuron in last layer. default 1000
    input_channel: An integer value input channels in first conv layer, default is 3.
    config: A string value indicating the configuration of MobileNetV3, either 'large' or 'small', default is 'large'.
    dropout: [0, 1] : A float parameter for dropout in last layer, between 0 and 1, default is 0.8.

"""
        super().__init__()

        # The configuration of MobileNet3.

```

```

# input channels, kernel size, expansion size, output channels, squeeze excitation, activation, stride
RE = nn.ReLU
HS = nn.Hardswish
config_dict = {
    "small": [
        (16, 3, 16, 16, True, RE, 2),
        (16, 3, 72, 24, False, RE, 2),
        (24, 3, 128, 48, True, HS, 2),
        (24, 5, 96, 48, True, HS, 2),
        (40, 5, 240, 48, True, HS, 1),
        (48, 5, 288, 96, True, HS, 2),
        (96, 5, 576, 96, True, HS, 1),
        (96, 5, 576, 96, True, HS, 1),
    ],
    "large": [
        (16, 3, 16, 16, False, RE, 1),
        (16, 3, 64, 24, False, RE, 2),
        (24, 3, 72, 24, False, RE, 1),
        (24, 3, 128, 48, True, HS, 2),
        (40, 5, 120, 48, True, RE, 1),
        (40, 5, 120, 48, True, RE, 1),
        (48, 3, 240, 80, False, HS, 2),
        (80, 3, 184, 80, False, HS, 1),
        (80, 3, 184, 80, False, HS, 1),
        (80, 3, 184, 80, False, HS, 1),
        (80, 3, 480, 112, True, HS, 1),
        (112, 3, 672, 112, True, HS, 1),
        (112, 3, 672, 112, True, HS, 1),
        (160, 5, 960, 160, True, HS, 1),
        (160, 5, 960, 160, True, HS, 1),
    ],
},
self.model = nn.Sequential(
    ConvNormActivationBlock(
        input_channel=16, (3, 3), stride=2, padding=1, activation=nn.Hardswish
    ),
)
for (
    in_channels,
    kernel_size,
    expansion_size,
    out_channels,
    squeeze_excitation,
    activation,
    stride
) in config_dict[config]:
    self.model.append(
        InvertedResidualBlock(
            in_channels=in_channels,
            out_channels=out_channels,
            kernel_size=kernel_size,
            expansion_size=expansion_size,
            stride=stride,
            squeeze_excitation=squeeze_excitation,
            activation=activation,
        )
    )

hidden_channels = 576 if config == "small" else 960
_out_channel = 1024 if config == "small" else 1280

self.model.append(
    ConvNormActivationBlock(
        out_channels,
        hidden_channels,
        (1, 1),
    )
)
if x.size(1) in self.index:
    position = self.index.index(x.size(1)) # Find the position in the index list
    results[position] = x

return results

if __name__ == "__main__":
    # Generating sample image
    image_size = (1, 3, 224, 224)
    image = torch.randn(*image_size)

    # Model
    mobilenet_v3 = MobileNetV3(config="small")

    # summary
    mobilenet_v3
    # input_size=3,
    # input_data=image,
    # col_names=[“input_size”, “output_size”, “num_params”],
    # device=“cpu”,
    # depth=2,
    # )

    out = mobilenet_v3(image)
    print("Output shape : ", out.shape)

```

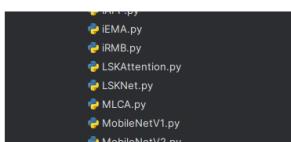
四、手把手教你添加MobileNetV3网络结构

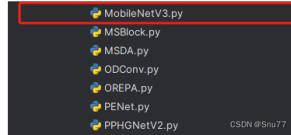
下面教大家如何修改该网络结构，主干网络结构的修改步骤比较复杂，我也会将task.py文件上传到CSDN的文件中，大家如果自己修改不正确，可以尝试用我的task.py文件替换你的，然后只需要修改其中的第1、2、3、5步即可。

修改过程中大家一定要仔细

4.1 修改一

首先我们找到如下“ultralytics/nn”的目录，我们在这个目录下创建一个新的目录，名字为‘Addmodules’（此文件之后就用于存放我们的所有改进机制），之后我们在创建的目录内创建一个新的py文件复制粘贴进去，可以根据文章改进机制起来。这里大家根据自己的习惯命名即可。





4.2 修改二

第二步我们在我们创建的目录内创建一个新的py文件名字为'__init__.py'（只需要创建一个即可），然后在其内部导入我们本文的改进机制即可。

```
> hub
> models
> nn
> Addmodules
  + __init__.py
  + ACMix.py
  + AFPNHead3.py
  + AFPNHead4.py
  + AKConv.py
  + ASFFHead.py
  + ASFFYOLO.py
  + Biformer.py
  + BiFFN.py
  + CARAFE.py
  + CGAttention.py

  1 from .SwinNeck import *
  2 from .SlimNeck import *
  3 from .CARAFE import *
  4 from .RCGSSA import *
  5 from .FocalModulation import *
  6 from .SPOConv import *
  7 from .RFACConv import *
  8 from .ShuffleConv import *
  9 from .ShuffleNetV1 import *
 10 from .ShuffleNetV2 import *
 11 from .MobileNetV1 import *
 12 from .MobileNetV2 import *
 13 from .MobileNetV3 import * -----
 14 from .LSKNet import *
 15 from .SwinTransformer import *
 16 from .SENetV1 import *
 17 from .SENetV2 import *
 18 from .VanillaLeNet import *

CSDN @Smu
```

4.3 修改三

第三步我们找到如下文件“ultralytics/nn/tasks.py”然后在开头导入我们的所有改进机制（如果你用了我多个改进机制，这一步只需要修改一次即可）。

```
  RevCoV1.py          Q: Segment_DySnakeConv      Cc W :  0 results ↑ ↓ ⌂
  RFACconv.py
  SACconv.py
  ScConv.py
  SENetV1.py
  SENetV2.py
  ShuffleNetV1.py
  ShuffleNetV2.py
  SlimNeck.py
  SPOConv.py
  SwinTransformer.py
  TransferXt.py
  TripleAttention.py
  UniRePNet.py
  VanillaNet.py

  ↴ modules
    __init__.py
    block.py
    conv.py

  1   # Ultraalytics YOLOv8, AGPL-3.0 license
  2   from .Addmodules import *
  3
  4   import contextlib
  5   from copy import deepcopy
  6   from pathlib import Path
  7
  8   import torch
  9   import torch.nn as nn
 10  from ultralytics.nn.modules import (AIFF, C1, C2, C3, CSTR, SPPF, Bottleneck, Box
 11                                         Classify, Concat, Conv, Conv2d, ConvTranspose, Det
 12                                         Focus, GhostBottleneck, GhostConv, HGBlock, HGSt
 13                                         ResnetLayer, TDETRDecoder, Segment)
 14
 15  from ultralytics.utils import DEFAULT_CFG_DICT, DEFAULT_CFG_NEVS, LOGGER, colorstr, e
 16  from ultralytics.utils.checks import check_file, check_requirements, check_sufix, check_yaml
 17  from ultralytics.utils.loss import v8ClassificationLoss, v8DetectionLoss, v8PoseLoss,
 18
 19  from ultralytics.utils.plotting import feature_visualization
 20
 21  from ultralytics.utils.torch_utils import (fuse_common_and.bn, fuse_deconv_and.bn, init
 22                                              make_divisible, model_info, scale_img, tim
 23                                              CSN@9s, CSN@18s, CSN@36s, CSN@72s, CSN@144s, CSN@288s, CSN@576s, CSN@1152s, CSN@2304s, CSN@4608s, CSN@9216s)
```

4.4 修改四

添加如下两行代码！！

```
699     ch = [ch]
700     layers, save, c2 = [], [], ch[-1] # layers, savelist, ch out
701
702     backbone = False
703
704
705     for i, (f, n, m, args) in enumerate(d['backbone']) + d['head']:
706         # from number module args
707         t = m
708
709         m = getattr(torch.nn, m[3:]) if 'nn.' in m else globals()[m] # get module
710
711         for j, a in enumerate(args):
712             if isinstance(a, str):
713                 with contextlib.suppress(ValueError):
714                     a = eval(a)
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1905
1906
1907
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1994
1995
1996
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2004
2005
2006
2006
2007
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2014
2015
2016
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2024
2025
2026
2026
2027
2027
2028
2028
2029
2029
2030
2031
2032
2033
2034
2035
2035
2036
2037
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2055
2056
2057
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2065
2066
2067
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2085
2086
2087
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2104
2105
2106
2106
2107
2107
2108
2108
2109
2109
2110
2111
2112
2113
2114
2115
2115
2116
2117
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2125
2126
2127
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2135
2136
2137
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2145
2146
2147
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2155
2156
2157
2157
2158
2158
2159
2159
2160
2161
2162
2163
2164
2165
2165
2166
2167
2167
2168
2168
2169
2169
2170
2171
2172
2173
2174
2175
2175
2176
2177
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2185
2186
2187
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2204
2205
2206
2206
2207
2207
2208
2208
2209
2209
2210
2211
2212
2213
2214
2215
2215
2216
2217
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2225
2226
2227
2227
2228
2228
2229
2229
2230
2231
2232
2233
2234
2235
2235
2236
2237
2237
2238
2238
2239
2239
2240
2241
2242
2243
2244
2245
2245
2246
2247
2247
2248
2248
2249
2249
2250
2251
2252
2253
2254
2255
2255
2256
2257
2257
2258
2258
2259
2259
2260
2261
2262
2263
2264
2265
2265
2266
2267
2267
2268
2268
2269
2269
2270
2271
2272
2273
2274
2275
2275
2276
2277
2277
2278
2278
2279
2279
2280
2281
2282
2283
2284
2285
2285
2286
2287
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2295
2296
2297
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2304
2305
2306
2306
2307
2307
2308
2308
2309
2309
2310
2311
2312
2313
2314
2315
2315
2316
2317
2317
2318
2318
2319
2319
2320
2321
2322
2323
2324
2325
2325
2326
2327
2327
2328
2328
2329
2329
2330
2331
2332
2333
2334
2335
2335
2336
2337
2337
2338
2338
2339
2339
2340
2341
2342
2343
2344
2345
2345
2346
2347
2347
2348
2348
2349
2349
2350
2351
2352
2353
2354
2355
2355
2356
2357
2357
2358
2358
2359
2359
2360
2361
2362
2363
2364
2365
2365
2366
2367
2367
2368
2368
2369
2369
2370
2371
2372
2373
2374
2375
2375
2376
2377
2377
2378
2378
2379
2379
2380
2381
2382
2383
2384
2385
2385
2386
2387
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2395
2396
2397
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2404
2405
2406
2406
2407
2407
2408
2408
2409
2409
2410
2411
2412
2413
2414
2415
2415
2416
2417
2417
2418
2418
2419
2419
2420
2421
2422
2423
2424
2425
2425
2426
2427
2427
2428
2428
2429
2429
2430
2431
2432
2433
2434
2435
2435
2436
2437
2437
2438
2438
2439
2439
2440
2441
2442
2443
2444
2445
2445
2446
2447
2447
2448
2448
2449
2449
2450
2451
2452
2453
2454
2455
2455
2456
2457
2457
2458
2458
2459
2459
2460
2461
2462
2463
2464
2465
2465
2466
2467
2467
2468
2468
2469
2469
2470
2471
2472
2473
2474
2475
2475
2476
2477
2477
2478
2478
2479
2479
2480
2481
2482
2483
2484
2485
2485
2486
2487
2487
2488
2488
2489
2489
2490
2491
2492
2493
2494
2495
2495
2496
2497
2497
2498
2498
2499
2499
2500
2501
2502
2503
2504
2504
2505
2506
2506
2507
2507
2508
2508
2509
2509
2510
2511
2512
2513
2514
2515
2515
2516
2517
2517
2518
2518
2519
2519
2520
2521
2522
2523
2524
2525
2525
2526
2527
2527
2528
2528
2529
2529
2530
2531
2532
2533
2534
2535
2535
2536
2537
2537
2538
2538
2539
2539
2540
2541
2542
2543
2544
2545
2545
2546
2547
2547
2548
2548
2549
2549
2550
2551
2552
2553
2554
2555
2555
2556
2557
2557
2558
2558
2559
2559
2560
2561
2562
2563
2564
2565
2565
2566
2567
2567
2568
2568
2569
2569
2570
2571
2572
2573
2574
2575
2575
2576
2577
2577
2578
2578
2579
2579
2580
2581
2582
2583
2584
2585
2585
2586
2587
2587
2588
2588
2589
2589
2590
2591
2592
2593
2594
2595
2595
2596
2597
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2604
2605
2606
2606
2607
2607
2608
2608
2609
2609
2610
2611
2612
2613
2614
2615
2615
2616
2617
2617
2618
2618
2619
2619
2620
2621
2622
2623
2624
2625
2625
2626
2627
2627
2628
2628
2629
2629
2630
2631
2632
2633
2634
2635
2635
2636
2637
2637
2638
2638
2639
2639
2640
2641
2642
2643
2644
2645
2645
2646
2647
2647
2648
2648
2649
2649
2650
2651
2652
2653
2654
2655
2655
2656
2657
2657
2658
2658
2659
2659
2660
2661
2662
2663
2664
2665
2665
2666
2667
2667
2668
2668
2669
2669
2670
2671
2672
2673
2674
2675
2675
2676
2677
2677
2678
2678
2679
2679
2680
2681
2682
2683
2684
2685
2685
2686
2687
2687
2688
2688
2689
2689
2690
2691
2692
2693
2694
2695
2695
2696
2697
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2704
2705
2706
2706
2707
2707
2708
2708
2709
2709
2710
2711
2712
2713
2714
2715
2715
2716
2717
2717
2718
2718
2719
2719
2720
2721
2722
2723
2724
2725
2725
2726
2727
2727
2728
2728
2729
2729
2730
2731
2732
2733
2734
2735
2735
2736
2737
2737
2738
2738
2739
2739
2740
2741
2742
2743
2744
2745
2745
2746
2747
2747
2748
2748
2749
2749
2750
2751
2752
2753
2754
2755
2755
2756
2757
2757
2758
2758
2759
2759
2760
2761
2762
2763
2764
2765
2765
2766
2767
2767
2768
2768
2769
2769
2770
2771
2772
2773
2774
2775
2775
2776
2777
2777
2778
2778
2779
2779
2780
2781
2782
2783
2784
2785
2785
2786
2787
2787
2788
2788
2789
2789
2790
2791
2792
2793
2794
2795
2795
2796
2797
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804

```

4.5 修改五

找到上百多行大概把吕体看图片，按照图片来修改就行，添加红框内的部分，注意没有()只是函数名。

```
726     if m in (Classify, Conv, ConvTranspose, GhostConv, Bottleneck, GhostBottleneck, SPP, sppF, DWConv, Focus,
727     BottleneckSPP, C1, C2, C2f, C3, C3TR, C3GHost, nn.ConvTranspose2d, DWConvTranspose2d, C3x, RepC3,
728     Blocks, ConvNormLayer):
```

```

711     c1, c2 = ch[f], args[0]
712     if c2 != nc: # if c2 not equal to number of classes (i.e. for Classify() output)
713         c2 = make_divisible(min(c2, max_channels) * width, divisor=8)
714     if m is Blocks:
715         if args[1] in {'BottleNeck'}:
716             c1, c2 = ch[f], args[0] * 4
717             args = [c1, args[0], *args[1:]]
718         else:
719             args = [c1, c2, *args[1:]]
720     else:
721         args = [c1, c2, *args[1:]]
722     if m in (BottleneckCSP, C1, C2, C2f, C3, C3TR, C3Ghost, C3x, RepC3, Blocks):
723         args.insert(_index_2, n) # number of repeats
724         n = 1
725
726     elif m in {vanillanet_5, vanillanet_6, vanillanet_7, vanillanet_8, vanillanet_9, vanillanet_10,
727                 repvit_10_0, repvit_10_9, repvit_11_0, repvit_11_9, repvit_11_5, repvit_11_3, LSKNet, LSKNET_tiny,
728                 LSKNET_base, SwinTransformer, MobileNetV1, MobileNetV2, MobileNetV3, shufflenet_v1_x0_5,
729                 shufflenet_v1_x1_0, shufflenet_v1_x1_5, shufflenet_v1_x2_0, shufflenetv2, revcol_small, revcol_tiny,
730                 revcol_base, revcol_xlarge, revcol_large, efficient, efficientnet_v2, FasterNet,
731                 CSWin_64_12221_tiny_224, CSWin_64_24322_small_224, CSWin_96_24322_base_224, CSWin_144_24322_large_224,
732                 convnext2_atto, convnext2_tiny, convnext2_small, convnext2_tiny,
733                 transnext_micro, transnext_tiny, transnext_small, transnext_base,
734                 unireplknet_a, unireplknet_f, unireplknet_p, unireplknet_o, unireplknet_t, unireplknet_s,
735                 unireplknet_b, unireplknet_l, unireplknet_xl, efficientvit_backbone_b0, efficientvit_backbone_b1,
736                 efficientvit_backbone_b2, efficientvit_backbone_b3,
737                 EfficientViT_M0, EfficientViT_M1, EfficientViT_M2, EfficientViT_M3, EfficientViT_M4, EfficientViT_M5,
738                 GhostnetV1, GhostnetV2};
739
740     m = m(*args)
741     c2 = m.width_list # 返回通道列表
742     backbone = True
743
744 elif m is AIFI:
745     args = [ch[f], *args]
746
747 elif m in (Detector, Segment, Pose, OBB):
748     args.append([cn[x] for x in f])
749     if m is Segment:
750         args[0] = make_divisible(min(args[0], max_channels) * width, divisor=8)
751     elif m is RTDETRDecoder: # special case, channels arg must be passed in index 1
752         args.insert(1, [cn[x] for x in f])
753     else:
754         c2 = ch[f]
755
756     m_ = nn.Sequential(*m(*args)) for i in range(n)) if n > 1 else m(*args) # module
757     t = str(i)[8:-2].replace('__main__', '') # module type
758     m_np = sum(x.numel() for x in m_.parameters()) # number params
759     m_i, n_f, n_type = i, f, t # attach index, 'from' index, type
760     if verbose:
761         LOGGER.info(f'{i}>>{str(f)}>{n}>>{m_np:10.0f} {t:<45}{str(args):<30}') # print
762         save.extend(x % for x in ([f] if isinstance(f, int) else f) if x != -1) # append to savelist
763     layers.append(m_)
764     if i == 0:
765         ch = []
766         ch.append(c2)
767     return nn.Sequential(*layers), sorted(save)
768
769

```

4.6 修改六

用下面的代码替换红框内的内容。

```

711     elif m in (Detector, Segment, Pose, OBB):
712         args.append([cn[x] for x in f])
713         if m is Segment:
714             args[0] = make_divisible(min(args[0], max_channels) * width, divisor=8)
715         elif m is RTDETRDecoder: # special case, channels arg must be passed in index 1
716             args.insert(1, [cn[x] for x in f])
717         else:
718             c2 = ch[f]
719
720         m_ = nn.Sequential(*m(*args)) for i in range(n)) if n > 1 else m(*args) # module
721         t = str(i)[8:-2].replace('__main__', '') # module type
722         m_np = sum(x.numel() for x in m_.parameters()) # number params
723         m_i, n_f, n_type = i, f, t # attach index, 'from' index, type
724         if verbose:
725             LOGGER.info(f'{i}>>{str(f)}>{n}>>{m_np:10.0f} {t:<45}{str(args):<30}') # print
726             save.extend(x % for x in ([f] if isinstance(f, int) else f) if x != -1) # append to savelist
727         layers.append(m_)
728         if i == 0:
729             ch = []
730             ch.append(c2)
731     return nn.Sequential(*layers), sorted(save)
732
733

```

```

if isinstance(c2, list):
    m_ = m
    m_.backbone = True
else:
    m_ = nn.Sequential(*m(*args) for _ in range(n)) if n > 1 else m(*args) # module
    t = str(m)[8:-2].replace('__main__', '') # module type
    m_np = sum(x.numel() for x in m_.parameters()) # number params
    m_i, n_f, n_type = i + 4 if backbone else i, f, t # attach index, 'from' index, type
    if verbose:
        LOGGER.info(f'{i}>>{str(f)}>{n}>>{m_np:10.0f} {t:<45}{str(args):<30}') # print
    save.extend(x % for x in ([f] if isinstance(f, int) else f) if x != -1) # append to savelist
    layers.append(m_)
if i == 0:
    ch = []
if isinstance(c2, list):
    ch.append(c2)
    if len(c2) != 5:
        ch.insert(0, None)
else:
    ch.append(c2)

```

4.7 修改七

修改七我们需要来到文件的开头，然后将下面红框内的部分用我给的代码进行替换即可，同时大家主意好不要替换错了！！

```
58     if augment:
59         return self._predict_augment(x)
60     return self._predict_once(x, profile=False, visualize=False, embed=None)
61
62     """
63     def _predict_once(self, x, profile=False, visualize=False, embed=None):
64         """
65         Perform a forward pass through the network.
66
67         Args:
68             x (torch.Tensor): The input tensor to the model.
69             profile (bool): Print the computation time of each layer if True, defaults to False.
70             visualize (bool): Save the feature maps of the model if True, defaults to False.
71             embed (list, optional): A list of feature vectors/embeddings to return.
72
73         Returns:
74             (torch.Tensor): The last output of the model.
75             """
76         y, dt, embeddings = [], [], [] # outputs
77         for m in self.model:
78             if m.f != -1: # if not from previous layer
79                 x = y[m.f] if isinstance(m.f, int) else [x if j == -1 else y[j] for j in m.f] # from earlier layers
80             if profile:
81                 self._profile_one_layer(m, x, dt)
82             x = m(x) # run
83             y.append(x if m.i in self.save else None) # save output
84             if visualize:
85                 feature_visualization(x, m.type, m.i, save_dir=visualize)
86             if embed and m.i in embed:
87                 embeddings.append(nn.functional.adaptive_avg_pool2d(x, output_size=(1, 1)).squeeze(-1).squeeze(-1)) # flatten
88             if m.i == max(embed):
89                 return torch.unbind(torch.cat(embeddings, dim=1), dim=0)
90
91     """
92     def _predict_augment(self, x):
93         """
94         Perform augmentations on input image x and return augmented inference. ***
95         """
96         LOGGER.warning(f"WARNING: {self.__class__.__name__} does not support augmented inference yet !")
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
825
826
827
827
828
829
829
830
831
832
832
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
164
```

```

241     if nc and nc != self.yaml['nc']:
242         LOGGER.info(f'Overriding model.yaml nc={self.yaml["nc"]} with nc={nc}')
243         self.yaml['nc'] = nc # override YAML value
244         self.model, self.save = parse_model(deepcopy(self.yaml), ch=ch, verbose=verbose) # model, savelist
245         self.names = {f'{i}': f'i' for i in range(self.yaml['nc'])} # default names dict
246         self.inplace = self.yaml.get('inplace', True)
247
248         # Build strides
249         m = self.model[-1] # Detect()
250         if isinstance(m, (Detect_Segment, Pose, OBB)):
251             s = 256 # fix min stride
252             n.inplace = self.inplace
253             forward = lambda x: self.forward(x)[0] if isinstance(m, (Segment, Pose, OBB)) else self.forward(x)
254             n.stride = torch.tensor([s / x.shape[-2] for x in forward(torch.zeros(1, ch, s, s))]) # forward
255             self.stride = n.stride
256             n.bias_init() # only run once
257         else:
258             self.stride = torch.Tensor([32]) # default stride for i.e. RTDETR
259
260         # Init weights, biases
261         initialize_weights(self)
262         if verbose:
263             self.info()
264             LOGGER.info('')
265
266         # Load weights
267         self.load_state_dict(torch.load(weights, map_location='cpu'))

```

CSN@Su7

4.9 RT-DETR不能打印计算量问题的解决

我们找到如下文件ultralytics/utils/torch_utils.py按照如下的图片进行修改，来解决计算量打印不出来的问题。

```

1 import ...
2 def get_flops(model, imgsz=320):
3     """Return a YOLO model's FLOPs."""
4     try:
5         model = de_parallel(model)
6         p = next(model.parameters())
7         stride = 640 # max stride
8         in_ = torch.empty((1, p.shape[1], stride, stride), device=p.device) # input image in BCHW format
9         flops = thop.profile(deepcopy(model), inputs=[in_], verbose=False)[0] / 100 * 2 if thop else 0 # stride GFLOPs
10        imgsz = imgsz if isinstance(imgsz, list) else [imgsz, imgsz] # expand if int/float
11        return flops * imgsz[0] / stride * imgsz[1] / stride # 640x640 GFLOPs
12    except Exception:
13        return 0
14

```

CSN@Su7

4.10 可选修改

有些读者的数据集部分图片比较特殊，在验证的时候会导致形状不匹配的报错，如果大家在验证的时候报错形状不匹配的错误可以固定验证集的图片尺寸，方法如下 ->

找到下面这个文件ultralytics/models/yolo/detect/train.py然后其中有一个类是DetectionTrainer class中的build_dataset函数中的一个参数rect_mode == 'val'改为rect=False

```

1 usage
2 def build_dataset(self, img_path, mode='train', batch=None):
3     """
4         Build YOLO Dataset.
5
6     Args:
7         img_path (str): Path to the folder containing images.
8         mode (str): train mode or val mode, users are able to customize different augmentations for each mode.
9         batch (int, optional): Size of batches, this is for 'rect'. Defaults to None.
10    """
11    gs = max(int(de_parallel(self.model).stride.max()) if self.model else 0), 32
12    return build_yolo_dataset(self.args, img_path, batch, self.data, mode=mode, rect=False, stride=gs)
13
14
15    def get_dataloader(self, dataset_path, batch_size=10, rank=0, mode='train'):
16        """
17            Construct and return dataloader.
18        """
19        assert mode in ['train', 'val']
20        with torch.distributed_zero_first(rank): # init dataset + cache only once if DDP
21            dataset = self.build_dataset(dataset_path, mode, batch_size)
22            shuffle = mode == 'train'
23            if getattr(dataset, 'rect', False) and shuffle:
24                LOGGER.warning('WARNING ▲ rect=True is incompatible with DataLoader shuffle=True, setting shuffle=False')
25

```

CSN@Su7

五、MobileNetV3的yaml文件

5.1 yaml文件

大家复制下面的yaml文件，然后通过我给大家的运行代码运行即可，RT-DETR的调参部分需要后面的文章给大家讲，现在目前免费给大家看这一部分不开放。

```

# Ultralytics YOLO 🚀, AGPL-3.0 license
# RT-DETR-1 object detection model with P3-P5 outputs. For details see https://docs.ultralytics.com/models/rtdetr

# Parameters
nc: 80 # number of classes
scale: 80 # model config scaling constants, i.e. 'model=yolov8n-cls.yaml' will call yolov8-cls.yaml with scale 'n'
        # [depth, width, max_channels]
        # [1.00, 1.00, 1024]

backbone:
    # [from, repeats, module, args]
    [-1, 1, MobileNetV3, []] # 4

head:
    # [4, 4, Conv, [256, 1, 1, None, 1, 1, False]] # 5 input_proj.2
    [-1, 4, AIFI, [1024, 8]] # 6
    [-1, 4, Conv, [256, 1, 1, 1, 1, False]] # 7, VS, lateral_convs.0

    [-1, 4, nn.Upsample, [None, 2, 'nearest']] # 8
    [-1, 4, Conv, [256, 1, 1, None, 1, 1, False]] # 9 input_proj.1
    [[-2, -1], 1, Concat, [1]] # 10
    [-1, 3, RepC3, [256, 0.5]] # 11, fpn_blocks.0
    [-1, 3, Conv, [256, 1, 1]] # 12, V4, lateral_convs.1

    [-1, 1, nn.Upsample, [None, 2, 'nearest']] # 13
    [-1, 1, Conv, [256, 1, 1, None, 1, 1, False]] # 14 input_proj.0
    [[-2, -1], 1, Concat, [1]] # 15 cat backbone_P4
    [-1, 3, RepC3, [256, 0.5]] # X3 (16), fpn_blocks.1

    [-1, 1, Conv, [256, 3, 2]] # 17, downsample_convs.0
    [[-1, 12], 1, Concat, [1]] # 18 cat V4
    [-1, 3, RepC3, [256, 0.5]] # F4 (19), pan_blocks.0

    [-1, 1, Conv, [256, 3, 2]] # 20, downsample_convs.1
    [[-1, 7], 1, Concat, [1]] # 21 cat VS
    [-1, 3, RepC3, [256, 0.5]] # F5 (22), pan_blocks.1

    [[16, 19, 22], 1, RTDETRDecoder, [nc, 256, 300, 4, 8, 3]] # Detect(P3, P4, P5)

```

5.2 运行文件

大家可以创建一个train.py文件将下面的代码粘贴进去然后替换你的文件运行即可开始训练。

```

import warnings
from ultralytics import RTDETR
warnings.filterwarnings('ignore')

if __name__ == '__main__':
    model = RTDETR('替换你想要运行的yaml文件')
    # model.load('') # 可以加载你的断点和训练权重
    model.train(dataloader='替换你的数据集地址即可',
                cache=False,
                imgsz=72,
                epochs=72,
                batch_size=4,
                workers=0,
                device='cpu',
                project='runs/RT-DETR-train',
                name='exp',
                # amp=True
                )

```

5.3 成功训练截图

下面是成功运行的截图（确保我的改进机制是可用的），已经完成了有1个epochs的训练，图片太大截不全第2个epochs了。

```

0      -1 1 2970118 MobileNetV3          []
1      -1 1 246272 ultralytics.nm.modules.conv.Conv [960, 256, 1, 1, None, 1, 1, False]
2      -1 1 789760 ultralytics.nm.modules.transformer.AIFI [256, 1024, 8]
3      -1 1 60848 ultralytics.nm.modules.conv.Conv [256, 256, 1, 1]
4      -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
5      3 1 29184 ultralytics.nm.modules.conv.Conv [112, 256, 1, 1, None, 1, 1, False]
6      [-2, -1] 1 0 ultralytics.nm.modules.conv.Concat [1]
7      -1 3 657920 ultralytics.nm.modules.block.RepC3 [512, 256, 3, 0.5]
8      -1 1 60848 ultralytics.nm.modules.conv.Conv [256, 256, 1, 1]
9      -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
10     2 1 10752 ultralytics.nm.modules.conv.Conv [40, 256, 1, 1, None, 1, 1, False]
11     [-2, -1] 1 0 ultralytics.nm.modules.conv.Concat [1]
12     -1 3 657920 ultralytics.nm.modules.block.RepC3 [512, 256, 3, 0.5]
13     -1 1 590336 ultralytics.nm.modules.conv.Conv [256, 256, 3, 2]
14     [-1, 12] 1 0 ultralytics.nm.modules.conv.Concat [1]
15     -1 3 657920 ultralytics.nm.modules.block.RepC3 [512, 256, 3, 0.5]
16     -1 1 590336 ultralytics.nm.modules.conv.Conv [256, 256, 3, 2]
17     [-1, 7] 1 0 ultralytics.nm.modules.conv.Concat [1]
18     -1 3 657920 ultralytics.nm.modules.block.RepC3 [512, 256, 3, 0.5]
19     [16, 19, 22] 1 397684 ultralytics.nm.modules.head.RTDETRDecoder [1, [256, 256, 256], 256, 300, 4, 8, 3]
rtdetr-ShuffleNetV1 summary: 581 layers, 11988218 parameters, 11988218 gradients, 27.8 GFLOPs

TensorBoard: Start with 'tensorboard --logdir runs\train\exp10', view at http://localhost:6006/
train: Scanning C:\Users\Administrator\Desktop\RT-DETR\smoking detection.v1-smoking-detection-v1.yolov8\train\labels.cache...
train: WARNING ▲ C:\Users\Administrator\Desktop\RT-DETR\smoking detection.v1-smoking-detection-v1.yolov8\train\images\fPckcjragA\WARNING ▲ Box and segment counts should be equal, but got len(segments) = 691, len(boxes) = 24489. To resolve this only boxes will val: WARNING ▲ C:\Users\Administrator\Desktop\RT-DETR\smoking detection.v1-smoking-detection-v1.yolov8\valid\images\EszouinTXAE00\WARNING ▲ Box and segment counts should be equal, but got len(segments) = 30, len(boxes) = 913. To resolve this only boxes will val: Scanning C:\Users\Administrator\Desktop\RT-DETR\smoking detection.v1-smoking-detection-v1.yolov8\valid\labels.cache... 507 images
Plotting labels to runs\train\exp10\labels.jpg...
optimizer: AdamW(lr=0.0001, momentum=0.937) with parameter groups 104 weight(decay=0.0), 150 weight(decay=0.0001), 107 bias(decay=0.0)
WARNING ▲ TensorBoard graph visualization failure 'NoneType' object is not subscriptable
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\train\exp10
Starting training for 72 epochs...

```

Epoch	GPU_mem	giou_loss	cls_loss	ll_loss	Instances	Size	AP	P@1
1/73	3.47G	0.49	7.921	1.195	??	468*	77.69	1.0000 Snu77

六、全文总结

从今天开始正式开始更新RT-DETR剑指论文专栏，本专栏的内容会迅速铺开，在短期大量更新，价格也会阶梯性上涨，所以想要和我一起学习RT-DETR改进，可以在前期直接关注，本文专栏旨在打造全网最好的RT-DETR专栏为想要发论文的家进行服务。

专栏链接：RT-DETR剑指论文专栏，持续复现各种顶会内容——论文收割机RT-DETR



出处: <http://www.hzhcontrols.cn/>

原文: <http://www.hzhcontrols.cn/new-2008627.html>

本文版权归原作者所有

欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利

相关: 漏洞sql注入 linux怎么看本机数据库 linux怎么查看数据库端口 龙管家数据库被清除怎么恢复 linux创建db2数据库

上一篇: 统计学-R语言-6.3

下一篇: 免费开源OCR 软件Umi-OCR

联系方式

QQ:623128629
QQ群:568015492
需要定制控件可以联系我们

产品

开源版
Pro版
Core版

关于

这是一个全面的控件库，其中包括了开源版本的.net framework控件库，欢迎您的加入。

友情链接: [Dotnet9](#) [1024todo](#)编程工具 [WTM框架](#) [技术宅](#) [天剑博客](#) [申请友链](#)
Copyright © 2019 HZHControls All rights 粤ICP备19038928号-11