

赞同 2563



分享

深入浅出Yolo系列之Yolov5核心基础知识完整讲解



江大白

关注他

2,563 人赞同了该文章

大白在之前写过[《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》](#)

对**Yolov4**的相关**基础知识**做了比较系统的梳理，但**Yolov4**后不久，又出现了**Yolov5**，虽然作者没有放上和**Yolov4**的直接测试对比，但在COCO数据集的测试效果还是很可观的。

很多人考虑到**Yolov5**的**创新性不足**，对算法是否能够**进化**，称得上**Yolov5**而议论纷纷。

但既然称之为**Yolov5**，也有很多非常不错地方值得我们学习。不过因为**Yolov5**的网络结构和**Yolov3**、**Yolov4**相比，不好可视化，导致很多同学看**Yolov5**看的云里雾里。

因此本文，大白主要对Yolov5四种网络结构的各个细节做一个深入浅出的分析总结，和大家一些探讨学习。

当然，随着旷视科技**Yolox**的发布，大白也深入研究，制作了[《深入浅出Yolo系列之Yolox核心基础完整讲解》](#)，可以直接[点击查看](#)。

版权申明：本文包含图片，都为大白使用**PPT**所绘制的，如需**网络结构高清图**和**模型权重**，可[点击查看下载](#)。

更新提醒（2022.4.2）：[《AI未来星球》](#)开放加入，超值福利，[点击加入](#)。



其中也包含了大白耗尽洪荒之力，准备了包括Yolo目标检测在内的**31节视频课程**，[可以点击查看](#)。

求职跳槽福利：为了便于大家求职、跳槽的准备，大白将**45家大厂**的**3500篇面经**，按照知识框架，整理成700多页的《**人工智能算法岗江湖武林秘籍**》，限时开放下载，[点击查看下载](#)。

本文目录

1 Yolov5 四种网络模型

1.1 Yolov5网络结构图

1.2 网络结构可视化

1.2.1 Yolov5s网络结构

1.2.2 Yolov5m网络结构

1.2.3 Yolov5l网络结构

1.2.4 Yolov5x网络结构

2 核心基础内容

2.1 Yolov3&Yolov4网络结构图

2.2 Yolov5核心基础内容

2.2.1 输入端

2.2.2 Backbone

2.2.3 Neck

2.2.4 输出端

2.3 Yolov5四种网络结构的不同点

2.3.1 四种结构的参数

2.3.2 Yolov5网络结构

2.3.3 Yolov5四种网络的深度

2.3.4 Yolov5四种网络的宽度

3 Yolov5相关论文及代码

4 小目标分割检测

5 Yolov5之自有数据集训练超详细教程

6 Yolox核心基础完整讲解

7 Yolov5之自有数据集训练超详细教程

8 相关数据集下载

9 后语

1 Yolov5四种网络模型

Yolov5官方代码中，给出的目标检测网络中一共有4个版本，分别是**Yolov5s**、**Yolov5m**、**Yolov5l**、**Yolov5x**四个模型。

学习一个新的算法，最好在脑海中对**算法网络的整体架构**有一个清晰的理解。



赞同 2563



分享



但比较尴尬的是，**Yolov5代码**中给出的网络文件是**yaml格式**，和原本Yolov3、Yolov4中的**cfg**不同。

因此无法用**netron工具**直接可视化的查看网络结构，造成有的同学不知道如何去学习这样的网络。

比如下载了**Yolov5**的**四个pt格式**的权重模型：



大白在《深入浅出Yolo系列之Yolov3&Yolov4核心基础完整讲解》中讲到，可以使用**netron工具**打开网络模型。

但因为netron对pt格式的文件兼容性并不好，直接使用netron工具打开，会发现，根本无法显示全部网络。

因此可以采用**pt->onnx->netron**的折中方式，先使用Yolov5代码中**models/export.py**脚本将pt文件转换为onnx格式，再用netron工具打开，这样就可以看全网络的整体架构了。



如果有同学对netron工具还不是很熟悉，这里还是放上安装netron工具的详解，如果需要安装，可以移步大白的另一篇文章：《网络可视化工具netron详细安装流程》

如需下载**Yolov5**整体的4个网络pt文件及onnx文件，也可[点击链接查看下载](#)，便于直观的学习。

1.1 Yolov5网络结构图

安装好netron工具，就可以可视化的打开Yolov5的网络结构。

这里大白也和之前讲解Yolov3&Yolov4同样的方式，绘制了Yolov5s整体的网络结构图。配合netron的可视化网络结构查看，脑海中的架构会更加清晰。

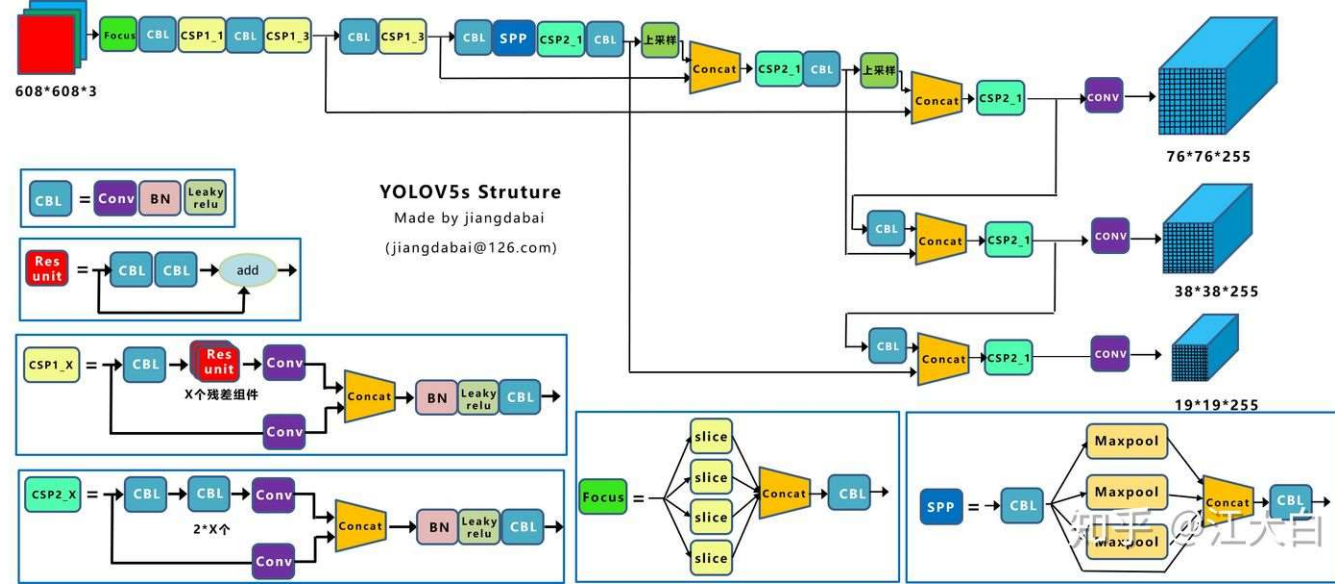


赞同 2563



分享





赞同 2563

分享

本文也会以Yolov5s的网络结构为主线，讲解与其他三个模型（**Yolov5m**、**Yolov5l**、**Yolov5x**）的不同点，让大家对于Yolov5有一个深入浅出的了解。

1.2 网络结构可视化

将四种模型pt文件的转换成对应的onnx文件后，即可使用[netron工具](#)查看。
但是，有些同学可能不方便，使用脚本转换查看。
因此，大白也上传了每个网络结构图的图片，也可以直接[点击查看](#)。
虽然没有netron工具更直观，但是也可以学习了解。

1.2.1 Yolov5s网络结构

Yolov5s网络是Yolov5系列中**深度最小**，特征图的**宽度最小**的网络。后面的3种都是在此基础上不断加深，不断加宽。

上图绘制出的网络结构图也是**Yolov5s**的结构，大家也可[直接点击查看](#)，Yolov5s的网络结构可视化的图片。

1.2.2 Yolov5m网络结构

此处也放上netron打开的**Yolov5m**网络结构可视图，[点击即可查看](#)，后面第二版块会详细说明不同模型的不同点。

1.2.3 Yolov5l网络结构

此处也放上netronx打开的**Yolov5l**网络结构可视图，[点击即可查看](#)。

1.2.4 Yolov5x网络结构

此处也放上netronx打开的**Yolov5x**网络结构可视图，[点击即可查看](#)。



2 核心基础内容

2.1 Yolov3&Yolov4网络结构图

2.1.1 Yolov3网络结构图

Yolov3的网络结构是比较经典的**one-stage结构**，分为**输入端、Backbone、Neck和Prediction**四个部分。

大白在之前的[《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》](#)中讲了很多，这里不多说，还是放上绘制的**Yolov3的网络结构图**。



赞同 2563



分享

2.1.2 Yolov4网络结构图

Yolov4在Yolov3的基础上进行了很多的创新。
比如**输入端**采用mosaic数据增强，
Backbone上采用了CSPDarknet53、Mish激活函数、Dropblock等方式，
Neck中采用了SPP、FPN+PAN的结构，
输出端则采用CIOU_Loss、DIOU_nms操作。

因此Yolov4对Yolov3的各个部分都进行了很多的整合创新，关于Yolov4详细的讲解还是可以参照大白之前写的[《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》](#)，写的比较详细。





赞同 2563



分享

2.2 YOLOv5核心基础内容

YOLOv5的结构和YOLOv4**很相似**，但也有一些不同，大白还是按照从整体到细节的方式，对每个板块进行讲解。

上图即**YOLOv5**的网络结构图，可以看出，还是分为**输入端、Backbone、Neck、Prediction**四个部分。

大家可能对**YOLOv3**比较熟悉，因此大白列举它和YOLOv3的一些主要的不同点，并和YOLOv4进行比较。

- (1) **输入端**：Mosaic数据增强、自适应锚框计算、自适应图片缩放
- (2) **Backbone**：Focus结构，CSP结构
- (3) **Neck**：FPN+PAN结构
- (4) **Prediction**：GIOU_Loss

下面丢上YOLOv5作者的算法性能测试图：





赞同 2563



分享

Yolov5作者也是在COCO数据集上进行的测试，大白在之前的文章讲过，COCO数据集的小目标占比，因此最终的四种网络结构，性能上来说各有千秋。

Yolov5s网络最小，速度最少，AP精度也最低。但如果检测的以大目标为主，追求速度，倒也是个不错的选择。

其他的三种网络，在此基础上，不断加深加宽网络，AP精度也不断提升，但速度的消耗也在不断增加。

2.2.1 输入端

(1) Mosaic数据增强

Yolov5的输入端采用了和Yolov4一样的Mosaic数据增强的方式。

Mosaic数据增强提出的作者也是来自Yolov5团队的成员，不过，**随机缩放、随机裁剪、随机排布**的方式进行拼接，对于小目标的检测效果还是很不错的。



Mosaic数据增强的内容在之前[《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》](#)文章中写的很详细，详情可以查看之前的内容。

（2）自适应锚框计算

在Yolo算法中，针对不同的数据集，都会有**初始设定长宽的锚框**。

在网络训练中，网络在初始锚框的基础上输出预测框，进而和**真实框groundtruth**进行比对，计算两者差距，再反向更新，**迭代网络参数**。

因此初始锚框也是比较重要的一部分，比如Yolov5在Coco数据集上初始设定的锚框：

在Yolov3、Yolov4中，训练不同的数据集时，计算初始锚框的值是通过单独的程序运行的。

但Yolov5中将此功能嵌入到代码中，每次训练时，自适应的计算不同训练集中的最佳锚框值。

当然，如果觉得计算的锚框效果不是很好，也可以在代码中将自动计算锚框功能**关闭**。

控制的代码即**train.py**中上面一行代码，设置成**False**，每次训练时，不会自动计算。

（3）自适应图片缩放

在常用的目标检测算法中，不同的图片长宽都不相同，因此常用的方式是将原始图片统一缩放到一个标准尺寸，再送入检测网络中。

比如Yolo算法中常用**416*416**，**608*608**等尺寸，比如对下面**800*600**的图像进行缩放。

但**Yolov5代码**中对此进行了改进，也是**Yolov5推理速度**能够很快的一个不错的trick。



赞同 2563



分享



作者认为，在项目实际使用时，很多图片的长宽比不同，因此缩放填充后，两端的黑边大小都不同，而如果填充的比较多，则存在信息冗余，影响推理速度。

因此在Yolov5的代码中datasets.py的letterbox函数中进行了修改，对原始图像**自适应的添加最少的黑边**。



赞同 2563



分享

图像高度上两端的黑边变少了，在推理时，计算量也会减少，即目标检测速度会得到提升。

这种方式在之前github上Yolov3中也进行了讨论：
<https://github.com/ultralytics/yolov3/issues/232>

在讨论中，通过这种简单的改进，推理速度得到了37%的提升，可以说效果很明显。

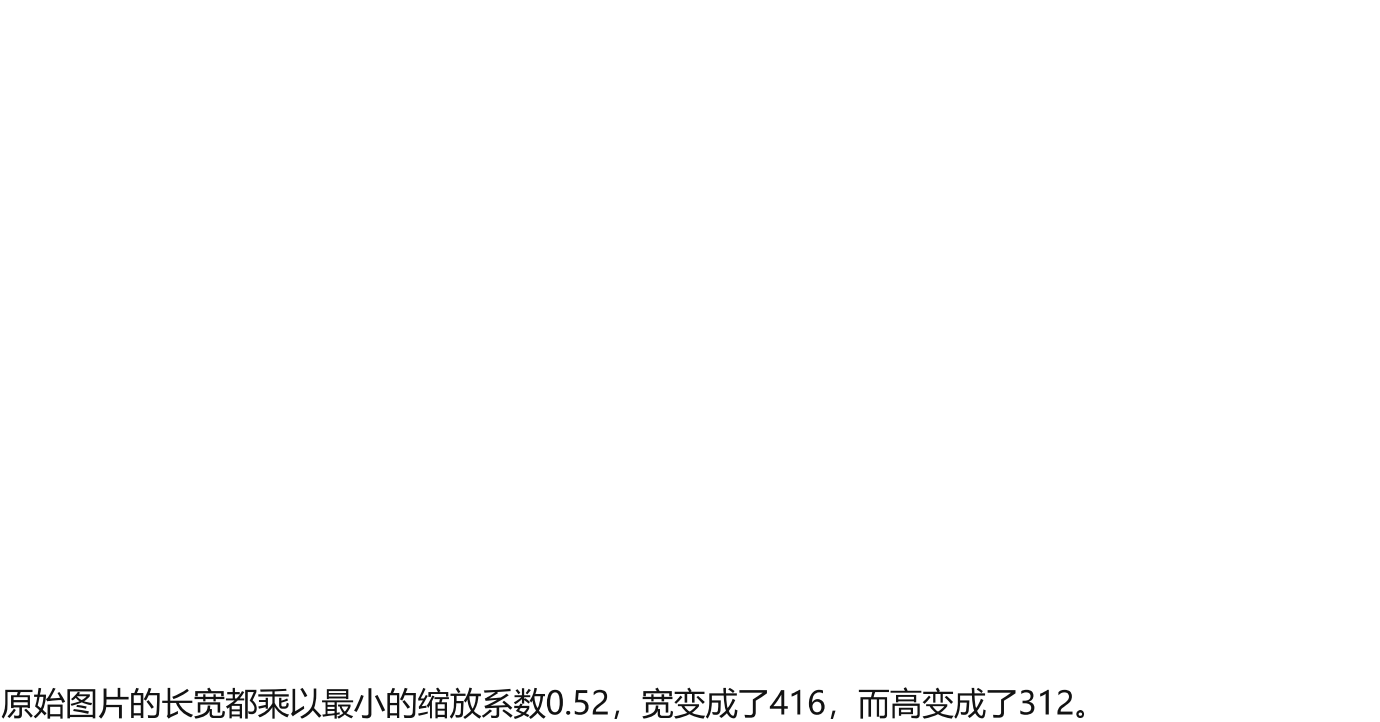
但是有的同学可能会有**大大的问号？**如何进行计算的呢？大白按照Yolov5中的思路详细的讲解一下，在**datasets.py的letterbox函数中**也有详细的代码。

第一步：计算缩放比例

原始缩放尺寸是416*416，都除以原始图像的尺寸后，可以得到0.52，和0.69两个缩放系数，选择小的缩放系数。



第二步：计算缩放后的尺寸



赞同 2563



分享

第三步：计算黑边填充数值



将 $416 - 312 = 104$ ，得到原本需要填充的高度。再采用numpy中`np.mod`取余数的方式，得到8个像素，再除以2，即得到图片高度两端需要填充的数值。

此外，需要注意的是：

a.这里大白填充的是黑色，即 **(0, 0, 0)**，而Yolov5中填充的是灰色，即 **(114,114,114)**，都是一样的效果。

b.训练时没有采用缩减黑边的方式，还是采用传统填充的方式，即缩放到416*416大小。只是在测试，使用模型推理时，才采用缩减黑边的方式，提高目标检测，推理的速度。

c.为什么`np.mod`函数的后面用**32**？因为Yolov5的网络经过5次下采样，而2的5次方，等于**32**。所以至少要去掉32的倍数，再进行取余。



2.2.2 Backbone

(1) Focus结构



赞同 2563



分享

Focus结构，在Yolov3&Yolov4中并没有这个结构，其中比较关键切片操作。

比如右图的切片示意图， $4 \times 4 \times 3$ 的图像切片后变成 $2 \times 2 \times 12$ 的特征图。

以Yolov5s的结构为例，原始 $608 \times 608 \times 3$ 的图像输入Focus结构，采用切片操作，先变成 $304 \times 304 \times 12$ 的特征图，再经过一次32个卷积核的卷积操作，最终变成 $304 \times 304 \times 32$ 的特征图。

需要注意的是：Yolov5s的Focus结构最后使用了32个卷积核，而其他三种结构，使用的数量有所增加，先注意下，后面会讲解到四种结构的不同点。

(2) CSP结构

Yolov4网络结构中，借鉴了CSPNet的设计思路，在主干网络中设计了CSP结构。

Yolov5与Yolov4不同点在于，Yolov4中只有主干网络使用了CSP结构。

而Yolov5中设计了两种CSP结构，以**Yolov5s网络**为例，**CSP1_X结构**应用于**Backbone主干网络**，另一种**CSP2_X结构**则应用于**Neck**中。





赞同 2563



分享

这里关于CSPNet的内容，也可以查看大白之前的[《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》](#)。

2.2.3 Neck

Yolov5现在的Neck和Yolov4中一样，都采用FPN+PAN的结构，但在Yolov5刚出来时，只使用了FPN结构，后面才增加了PAN结构，此外网络中其他部分也进行了调整。

因此，大白在Yolov5刚提出时，画的很多结构图，又都重新进行了调整。

这里关于FPN+PAN的结构，大白在[《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》](#)中，讲的很多，大家应该都有理解。

但如上面CSPNet结构中讲到，Yolov5和Yolov4的不同点在于，



Yolov4的Neck结构中，采用的都是普通的卷积操作。而Yolov5的Neck结构中，采用借鉴CSPnet设计的CSP2结构，加强网络特征融合的能力。



赞同 2563



分享

2.2.4 输出端

(1) Bounding box损失函数

在《深入浅出Yolo系列之Yolov3&Yolov4核心基础知识完整讲解》中，大白详细的讲解了IOU_Loss，以及进化版的GIOU_Loss，DIOU_Loss，以及CIOU_Loss。

Yolov5中采用其中的CIOU_Loss做Bounding box的损失函数。

Yolov4中也采用CIOU_Loss作为目标Bounding box的损失。

(2) nms非极大值抑制

在目标检测的后处理过程中，针对很多目标框的筛选，通常需要nms操作。



因为CIOU_Loss中包含影响因子v，涉及groudtruth的信息，而测试推理时，是没有groundtruth的。

所以Yolov4在DIOU_Loss的基础上采用DIOU_nms的方式，而Yolov5中采用加权nms的方式。

可以看出，采用DIOU_nms，下方中间箭头的黄色部分，原本被遮挡的摩托车也可以检出。



赞同 2563



分享

大白在项目中，也采用了DIOU_nms的方式，在同样的参数情况下，将nms中IOU修改成DIOU_nms。对于一些**遮挡重叠的目标**，确实会有一些改进。

比如下面**黄色箭头部分**，原本两个人重叠的部分，在参数和普通的IOU_nms一致的情况下，修改成DIOU_nms，可以将两个目标检出。

虽然大多数状态下效果差不多，但在不增加计算成本的情况下，有**稍微的改进**也是好的。

2.3 Yolov5四种网络结构的不同点

Yolov5代码中的四种网络，和之前的Yolov3，Yolov4中的**cfg文件**不同，都是以**yaml**的形式来呈现。



而且四个文件的内容基本上都是一样的，只有最上方的**depth_multiple**和**width_multiple**两个参数不同，很多同学看的一脸懵逼，不知道只通过两个参数是如何控制四种结构的？

2.3.1 四种结构的参数

大白先取出Yolov5代码中，每个网络结构的两个参数：

(1) Yolov5s.yaml

(2) Yolov5m.yaml

(3) Yolov5l.yaml

(4) Yolov5x.yaml

四种结构就是通过上面的两个参数，来进行控制网络的**深度**和**宽度**。其中**depth_multiple**控制网络的**深度**，**width_multiple**控制网络的**宽度**。

2.3.2 Yolov5网络结构

四种结构的yaml文件中，下方的网络架构代码都是一样的。

为了便于讲解，大白将其中的Backbone部分提取出来，讲解如何控制网络的宽度和深度，yaml文件中的Head部分也是同样的原理。

在对网络结构进行解析时，yolo.py中下方的这一行代码将四种结构的**depth_multiple**，**width_multiple**提取出，赋值给**gd**，**gw**。后面主要对这**gd**，**gw**这两个参数进行讲解。



赞同 2563



分享



下面再细致的剖析下，看是如何控制每种结构，深度和宽度的。

2.3.3 YOLOv5四种网络的深度



赞同 2563



分享

(1) 不同网络的深度

在上图中，大白画了两种CSP结构，CSP1和CSP2，其中CSP1结构主要应用于Backbone中，CSP2结构主要应用于Neck中。

需要注意的是，四种网络结构中每个CSP结构的深度都是不同的。

a.以YOLOv5s为例，第一个CSP1中，使用了1个残差组件，因此是**CSP1_1**。而在YOLOv5m中，则增加了网络的深度，在第一个CSP1中，使用了2个残差组件，因此是**CSP1_2**。

而YOLOv5l中，同样的位置，则使用了**3个残差组件**，YOLOv5x中，使用了**4个残差组件**。

其余的第二个CSP1和第三个CSP1也是同样的原理。

b.在第二种CSP2结构中也是同样的方式，以第一个CSP2结构为例，YOLOv5s组件中使用了 $2 \times X = 2 \times 1 = 2$ 个卷积，因为 $X = 1$ ，所以使用了1组卷积，因此是**CSP2_1**。

而YOLOv5m中使用了**2组**，YOLOv5l中使用了**3组**，YOLOv5x中使用了**4组**。

其他的四个CSP2结构，也是同理。

YOLOv5中，网络的不断加深，也在不断增加**网络特征提取**和**特征融合**的能力。

(2) 控制深度的代码

控制四种网络结构的核心代码是**yolo.py**中下面的代码，存在两个变量，**n**和**gd**。

我们再将**n**和**gd**带入计算，看每种网络的变化结果。



(3) 验证控制深度的有效性

我们选择最小的yolov5s.yaml和中间的yolov5l.yaml两个网络结构，将gd(depth_multiple)系数带入，看是否正确。



赞同 2563



分享

a. yolov5s.yaml

其中depth_multiple=0.33，即gd=0.33，而n则由上面红色框中的信息获得。

以上面网络框图中的第一个CSP1为例，即上面的第一个红色框。n等于第二个数值3。

而gd=0.33，带入（2）中的计算代码，结果n=1。因此第一个CSP1结构内只有1个残差组件，即CSP1_1。

第二个CSP1结构中，n等于第二个数值9，而gd=0.33，带入（2）中计算，结果n=3，因此第二个CSP1结构中有3个残差组件，即CSP1_3。

第三个CSP1结构也是同理，这里不多说。

b. yolov5l.xml

其中depth_multiple=1，即gd=1

和上面的计算方式相同，第一个CSP1结构中，n=3，带入代码中，结果n=3，因此为CSP1_3。

下面第二个CSP1和第三个CSP1结构都是同样的原理。

2.3.4 Yolov5四种网络的宽度





赞同 2563



分享

(1) 不同网络的宽度:

如上图表格中所示，四种yolov5结构在不同阶段的卷积核的数量都是不一样的，因此也直接影响卷积后特征图的第三维度，即**厚度**，大白这里表示为网络的**宽度**。

a.以Yolov5s结构为例，第一个Focus结构中，最后卷积操作时，卷积核的数量是32个，因此经过**Focus结构**，特征图的大小变成**304*304*32**。

而yolov5m的**Focus结构**中的卷积操作使用了48个卷积核，因此**Focus结构**后的特征图变成**304*304*48**。yolov5l，yolov5x也是同样的原理。

b. 第二个卷积操作时，yolov5s使用了64个卷积核，因此得到的特征图是**152*152*64**。而yolov5m使用96个特征图，因此得到的特征图是**152*152*96**。yolov5l，yolov5x也是同理。

c. 后面三个卷积下采样操作也是同样的原理，这样大白不过多讲解。

四种不同结构的卷积核的数量不同，这也直接影响网络中，比如**CSP1**，**CSP2等结构**，以及各个普通卷积，卷积操作时的卷积核数量也同步在调整，影响整体网络的计算量。

大家最好可以将结构图和[前面第一部分四个网络的特征图链接](#)，对应查看，思路会更加清晰。

当然卷积核的数量越多，特征图的厚度，即**宽度越宽**，网络提取特征的**学习能力也越强**。

(2) 控制宽度的代码

在yolov5的代码中，控制宽度的核心代码是**yolo.py**文件里面的这一行：

它所调用的子函数**make_divisible**的功能是：

(3) 验证控制宽度的有效性



我们还是选择**最小的yolov5s**和**中间的yolov5l**两个网络结构，将**width_multiple**系数带入，看是否正确。



赞同 2563



分享

a. yolov5s.yaml

其中**width_multiple=0.5**，即**gw=0.5**。

以第一个卷积下采样为例，即Focus结构中下面的卷积操作。

按照上面Backbone的信息，我们知道Focus中，标准的c2=64，而**gw=0.5**，代入（2）中的计算公式，最后的结果=32。即Yolov5s的Focus结构中，卷积下采样操作的卷积核数量为**32个**。

再计算后面的第二个卷积下采样操作，标准c2的值=128，**gw=0.5**，代入（2）中公式，最后的结果=64，也是正确的。

b. yolov5l.yaml

其中**width_multiple=1**，即**gw=1**，而标准的**c2=64**，代入上面（2）的计算公式中，可以得到Yolov5l的Focus结构中，卷积下采样操作的卷积核的数量为64个，而第二个卷积下采样的卷积核



数量是128个。

另外的三个卷积下采样操作，以及**yolov5m**，**yolov5x结构**也是同样的计算方式，大白这里不过多解释。

3 Yolov5相关论文及代码

3.1 代码

Yolov5的作者并没有发表论文，因此只能从代码角度进行分析。

Yolov5代码：[github.com/ultralytics/...](https://github.com/ultralytics/yolov5)

大家可以根据网页的说明，下载训练，及测试，流程还是比较简单的。

3.2 相关论文

另外一篇论文，**PP-Yolo**，在Yolov3的原理上，采用了很多的**tricks调参方式**，也挺有意思。

感兴趣的话可以参照另一个博主的文章：[点击查看](#)

4 小目标分割检测

目标检测发展很快，但对于**小目标的检测**还是有一定的瓶颈，特别是**大分辨率图像小目标检测**。比如**7920*2160**，甚至**16000*16000**的图像。



赞同 2563



分享





赞同 2563



分享

图像的分辨率很大，但又有很多小的目标需要检测。但是如果直接输入检测网络，比如yolov3，检出效果并不好。

主要原因是：

(1) 小目标尺寸

以网络的输入608*608为例，yolov3、yolov4，yolov5中下采样都使用了5次，因此最后的特征图大小是**19*19，38*38，76*76**。

三个特征图中，最大的76*76负责检测小目标，而对应到608*608上，每格特征图的感受野是 **$608/76=8*8$** 大小。

再将608*608对应到7680*2160上，以最长边7680为例， $7680/608*8=101$ 。



即如果原始图像中目标的宽或高小于101像素，网络很难学习到目标的特征信息。

(PS：这里忽略多尺度训练的因素及增加网络检测分支的情况)

(2) 高分辨率

而在很多遥感图像中，长宽比的分辨率比7680*2160更大，比如上面的16000*16000，如果采用直接输入原图的方式，很多小目标都无法检测出。

(3) 显卡爆炸

很多图像分辨率很大，如果简单的进行下采样，下采样的倍数太大，容易丢失数据信息。

但是倍数太小，网络前向传播需要在内存中保存大量的特征图，极大耗尽GPU资源,很容易发生**显存爆炸**，无法正常的训练及推理。

因此可以借鉴2018年YOLT算法的方式，改变一下思维，对大分辨率图片先进行分割，变成一张张小图，再进行检测。

需要注意的是：

为了避免两张小图之间，一些目标正好被分割截断，所以两个小图之间设置**overlap重叠区域**，比如分割的小图是**960*960**像素大小，则overlap可以设置为**960*20%=192**像素。

每个小图检测完成后，再将所有的框放到大图上，对大图整体做一次**nms操作**，将重叠区域的很多重复框去除。

这样操作，可以将很多小目标检出，比如**16000*16000**像素的遥感图像。



赞同 2563



分享



注意：这里关于小图检测后，放到大图上的方法，发现评论中，很多的同学可能想的过于复杂了，采用的方式，其实按照在大图上裁剪的位置，直接回归到大图即可。：

此外，国内还有一个10亿像素图像目标检测的比赛，也是用的这样的方式，大白将其中一个讲解不错的视频，也放到这个，大家可以[点击查看](#)。



赞同 2563



分享

无人机视角下，也有很多小的目标。大白也进行了测试，效果还是不错的。

比如下图是将**原始大图**->**416*416**大小，直接使用目标检测网络输出的效果：

可以看到**中间黄色框区域**，很多汽车检测漏掉。



再使用分割的方式，将大图**先分割成小图，再对每个小图检测**，可以看出中间区域很多的汽车都被检测出来：



赞同 2563



分享

不过这样的方式有优点也有缺点：

优点：

(1) 准确性

分割后的小图，再输入目标检测网络中，对于**最小目标像素的下限**会大大降低。

比如分割成608*608大小，送入输入图像大小608*608的网络中，按照上面的计算方式，原始图片上，长宽**大于8个像素**的小目标都可以学习到特征。

(2) 检测方式

在大分辨率图像，比如遥感图像，或者无人机图像，如果无需考虑实时性的检测，且对**小目标检测**也有需求的项目，可以尝试此种方式。

缺点：

(1) 增加计算量

比如原本7680*2160的图像，如果使用直接大图检测的方式，一次即可检测完。

但采用分割的方式，切分成N张608*608大小的图像，再进行N次检测，会大大增加检测时间。

借鉴Yolov5的四种网络方式，我们可以采用**尽量轻**的网络，比如Yolov5s网络结构或者更轻的网络。



当然Yolov4和Yolov5的网络各有优势，我们也可以借鉴Yolov5的设计方式，对Yolov4进行**轻量化改造**，或者进行**剪枝**。

5 Yolov5之自有数据集训练超详细教程

大白使用人头数据集，对于Yolov5进行训练，并记录成超级详细的训练教程。



赞同 2563



分享

6 YoloX核心基础完整讲解

旷视科技发布的YoloX有很多干货，可以查看大白的另一篇YoloX文章。

7 YoloX之自有数据集训练超详细教程

对于YoloX，大白也使用人头数据集，对于YoloX进行训练，并记录成超级详细的训练教程。

8 相关数据集下载

人工智能行业，除了算法，数据是非常重要的金矿。

比如在目标检测项目中，需要检测人体，那我们就需要人体数据集。当需要检测车辆，就需要车辆数据集。

为了便于大家查找相关数据集，大白将梳理的所有的数据集进行汇总，便于大家对应查看。

(1) 260+ 各类公开数据集





赞同 2563



分享

可以根据不同的需求，筛选出自己需要的数据集，在官网中进行下载。

数据集链接：[点击查看](#)

(2) 数十个可下载数据集





赞同 2563



分享

大白将一些收集的数据集，以及购买的部分数据集，都放在百度网盘中，可以进行下载。

下载链接：[点击查看](#)

(3) 花费过万购买的数据集





赞同 2563



分享

在从事AI行业的这三年中，大白零零散散，花费了一万多，购买了各类项目中所需要的数据集。

而且为了便于大家了解不同的数据集，算法功能、类别标签等详细信息，大白还制作了数据集管理文档，甚至在每行最后贴上了数据集图片，可以直接点击打开，查看是否自己需要的数据。

数据集管理文档清单链接：[点击查看](#)

数据集下载方式：[点击查看](#)

以上三种方式，如果大家都有的话，应该基本上可以解决学习中或者工作中，很多的训练需求了。

9 后语

综合而言，在实际测试中，Yolov4的准确性有不错的优势，但Yolov5的多种网络结构使用起来更加灵活，我们可以根据不同的项目需求，**取长补短**，发挥不同检测网络的优势。

此外，Yolox的改进trick中，也有很多值得借鉴的地方。

希望在人工智能的道路上，**和大家共同进步**。

编辑于 2022-04-01 20:17



赞赏

4 人已赞赏



目标检测 yoloV4 yoloV3

赞同 2563



分享

文章被以下专栏收录



人工智能入门
30天入门人工智能



机器学习算法工程师
微信公众号，内容更精彩！



机器学习算法工程师
欢迎关注同名微信公众号



深度学习目标检测的那点儿东西
找出图像中的目标（物体），确定它们的位置和类别。



深度学习深度学习
深度学习系列相关，幸会。

推荐阅读

深入浅出Yolo系列之Yolov5核心基础知识完整讲解

对 Yolov4的相关基础知识做了比较系统的梳理，但Yolov4后不久，又出现了Yolov5，虽然作者没有放上和Yolov4的直接测试对比，但在COCO数据集的测试效果还是很可观的。很多人考虑到 Yolov5的创...

新智元 发表于新智元

YOLO系列梳理（三）YOLOv5

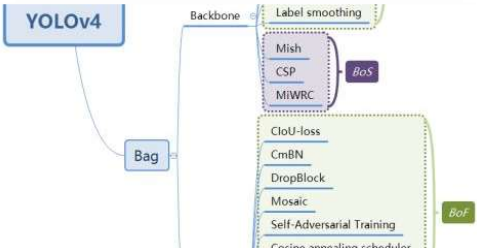
前言YOLOv5 是在 YOLOv4 出来之后没多久就横空出世了。今天笔者介绍一下 YOLOv5 的相关知识。目前 YOLOv5 发布了新的版本，6.0 版本。在这里，YOLOv5 也在5.0基础上集成了更多特性，同时也对...

CV技术指... 发表于YOLO系...



YOLOV5简单介绍以及YOLO系列算法比较

AI高级人... 发表于深度人脸识...



YOLOv4 介绍及其模型优化方法

贰浪先生



写下你的评论...

VIP

+

寒阶听燕语

2020-08-11

yolov5中对bbox的检测公式也进行了修改：[github.com/ultralytics/...](https://github.com/ultralytics/yolov5)

16

江大白

(作者)

回复

寒阶听燕语

2020-08-12

赞,感谢分享!

赞

cyf

回复

寒阶听燕语

2021-02-23

我还是看不到检测公式的原理，您能帮忙解释下吗

赞

展开其他 3 条回复

TimZhang

2021-09-20

感觉yolov5是针对yolov4在工程应用上做了一些列工程优化，使得大家用起来有更多的选择和更加的简单

6

snow

2021-05-23

yolov4的二作(cspnet的一作)与之进行了激烈的探讨~：[About reproduced results · Issue #6 · ultralytics/yolov5](#)

emm, 看完涨了不少姿势🤔

5

pony马

回复

snow

2021-10-30

确实火药味挺浓的，csp的一作咄咄逼人，不过还是在认真讨论

赞

搬砖的

2020-08-11

大佬强，先占个坑，正在做yolov5->tensorrt的跨平台部署，[github.com/enazoe/yolo-...](https://github.com/enazoe/yolo-tensorrt)



5

江大白

(作者)

回复

搬砖的

2020-08-11

赞

▲

赞同 2563

🔗

分享




搬砖的 回复 江大白 (作者)

2020-08-28

已经完成了, yolov5 tensorrt




6




snow

2021-05-29

v5现在的CSP结构都统一用C3了, 和v4的CSP结构很像, 少一个卷积。这是不是说明两条路径先bnleakyrelu再concat的结果要好于原来的先concat在bn_leakyrelu




4




取不到名字 回复 snow

03-24

从4.0开始将CSP换为C3, 作者解释的三个原因: 1) 使模型更为简单2) 参数量更少3) 便于推理过程的fuse操作。
作者解释连接: github.com/ultralytics/...



1




海风

01-10

yolov5激活函数不是leakyrelu, 是silu




3



影醉闲轩窗 回复 海风

02-16

你接触的太迟了, 大白写知乎的时候是v5早期版本



8



张德帅 回复 海风


04-18

现在已经有6个版本的YOLOv5了



1

展开其他 3 条回复




凛冬啊


2021-07-04

“三个特征图中, 最大的76*76负责检测小目标, 而对应到608*608上, 每格特征图的感受野是608/76=8*8大小。”

感受野的大小不是这么计算的吧, 不是下采样倍数啊




3




江大白 (作者) 回复 凛冬啊

2021-07-07

可以参考下这个链接, 关于感受野的定义[卷积神经网络中感受野的详细介绍_Microstrong-CSDN博客_感受野](#)



赞



凛冬啊 回复 江大白 (作者)

2021-07-07

首先不参考也知道, 感受野和下采样倍数是两回事。
感受野的定义: 在卷积神经网络CNN中, 决定某一层输出结果中, 一个元素所对应的输入



赞同 2563



分享



感受野的定义：在卷积神经网络CNN中，决定某一层输出结果中一个元素所对应的输入层的区域大小，被称作感受野。也就是76*76 feature map上的一个点和原图上的多少像素点有关，这么多卷积下来怎么可能只有8×8. 其次，感受野根本和输入的分辨率无关。其实从定义也看的出来最后，感受野应该区分理论感受野和实际感受野，所以你说

的是哪个感受野呢？
网上资料很多，我就不贴了

👍 6

展开其他 3 条回复

 知乎用户x1WH9Z2021-03-18

对于第四部分-小目标检测部分，是作者自己总结的吗

👍 3

 江大白👑(作者) 回复 知乎用户x1WH9Z2021-03-18

是的

👍 2

 小包总2021-05-15

训练的时候不是640px，这里为什么是608px，谢谢！

👍 2

 江大白👑(作者) 回复 小包总2021-05-19

网络训练的尺寸其实不是固定的，因为网络下采样5次，2的五次方等于32，因此网络的输入尺寸只需要是32的倍数，能保证下采样缩小5次即可。

👍 4

 这江 回复 江大白👑(作者)2021-05-30

那最后的输出是否也会因为尺寸的不同而不同🤔

👍 赞

展开其他 2 条回复

 dingyufeng2021-11-26

楼主，您好，想请教一个问题，相同平台、gpu配置下，理论上yolov5s是不是要比yolov4—tiny训练的速度更快？因为我在相同条件下训练了这两个模型，yolov4-tiny训300epoch用时4个半小时，而yolov5s训了50个epoch就用了5个半小时，这个情况是不是有问题？

👍 1

 仰面望天 回复 dingyufeng02-22

是yolov5可以修改batch-size的原因吗？

👍 赞

 普朗克2021-11-25

赞同 2563



分享



	楼主你好，请问针对yolov5有哪些可以继续改进的点	
	1	
	江大白  (作者) 回复 普朗克	2021-11-25
	可以私聊下我，之前刚给一个小伙伴整套的建议	
	1	
	青城沐雨 回复 普朗克	05-30
	请问有哪些改进的建议啊	
	赞	
展开其他 1 条回复		
	阿尼亚啃大瓜	03-03
	感谢楼主！学到了很多👍	
	1	
	佳人留影	2020-10-20
	您好，楼主，大分辨率小目标检测具体要怎么操作，有具体的开源代码吗？	
	9	
	kgg	2020-09-05
	感觉写很好，只是可惜最重要的anchor assign部分被跳过了	
	8	
	CV波比 回复 kgg	05-18
	兄弟 有这部分的博客吗	
	赞	
	江大白  (作者) 回复 CV波比	05-20
	你好，指的CSDN吗？	
	赞	
查看全部 8 条回复		
	嘻嘻	2021-05-24
	v5的focus和passthrough有啥区别吗，都是切片	
	2	
	林甜球 回复 嘻嘻	2021-12-21
	切片顺序不同 focus列优先 passthrough行优先 可以看看源码	
	1	
	林甜球 回复 嘻嘻	2021-12-21


赞同 2563

分享



 林甜球 回复 嘻嘻	2021-12-21
不过文章里的focus又是行优先，那可能就理解成两个是完全相同的结构，叫法不同	
 赞	
 塔努努	2020-08-13
楼主的文章，真是金玉良言	
 1	
 江大白  (作者) 回复 塔努努	2020-08-13
很高兴对你有帮助	
 赞	
 qsc	2020-08-11
想问一下v4的dropblock是加在了哪儿 看alexeyab的代码 是定义了dropout layer（即dropblock）但是没看见他加在什么地方	
 1	
 哈喽你好  回复 qsc	2020-10-25
请问您知道加在哪儿了吗	
 赞	
 qsc 回复 哈喽你好 	2020-10-25
not really 玄学 你可以看看pp-yolo加在哪儿了 yolov4应该没加	
 赞	
 qsc	2020-08-11
yolo层如何计算xywh其实也有点不一样	
 1	
 江大白  (作者) 回复 qsc	2020-08-12
是的,楼上的兄台也分享了相关的问题链接	
 赞	
 liguiyuan	2021-11-24
楼主，您好，请教一个关于nms的问题，小目标检测时，目标刚好被分割成两张图片，这两个小图都检测到了该目标，把所有框放到大图后这个目标就会有两个框，但这两个框的iou值小于0.45（或某个阈值），就无法过滤去掉其中一个框，请问你怎么操作	
 赞	
 冷雨	2021-11-21
"得到8个像素，再除以2，即得到图片高度两端需要填充的数值" 所以到底是填充104还是8？为什么是8不是104？为什么5次下采样就填充8？	
 赞	



赞



江大白 (作者) 回复 冷雨

2021-11-23

这里填充8，是将312拉到320，填充最少的像素，达到可以被32整除的效果。此外5次下采样，得到网络输入需要被32整除的信息，而8是填充的最小数值。

赞

