1. 正常训练

任务号：yolo22

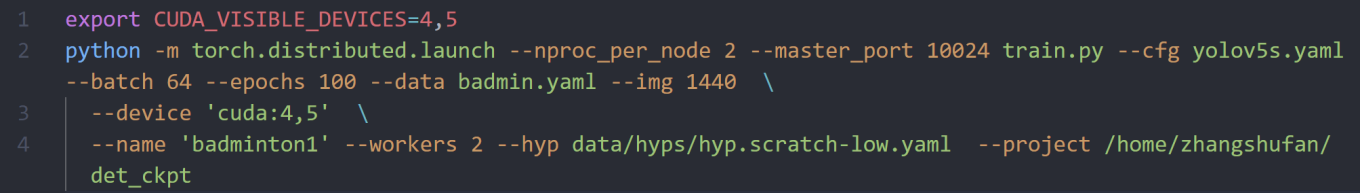
模型：yolov5s

数据：hyp.scratch-low.yaml

GFLOPs：16.5

数据集：训练集大约24640张图片，验证集大约1440张图片

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

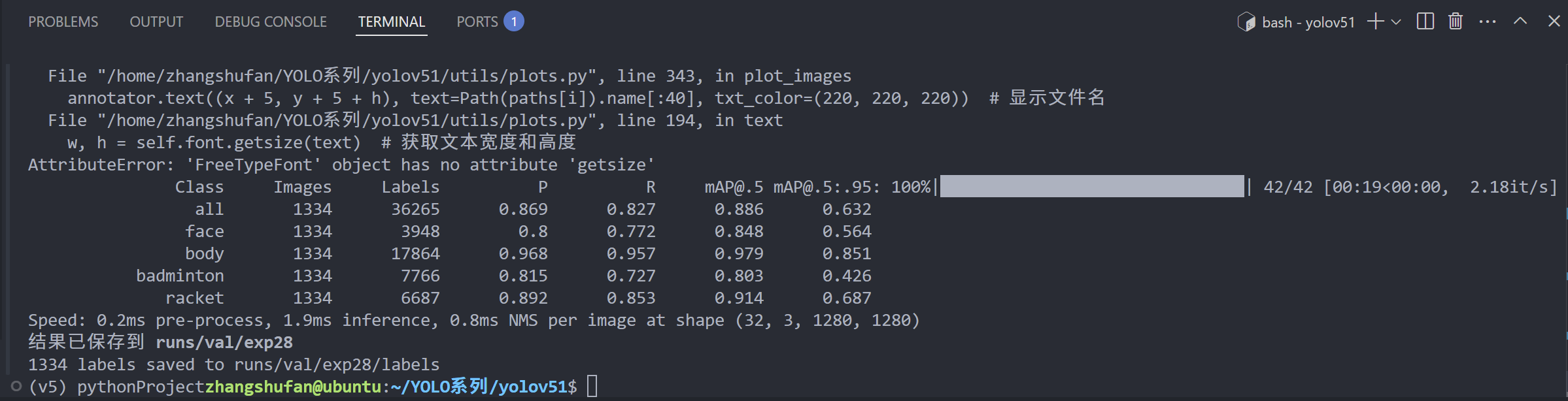
推理：

输入图像1280

conf-thres：0.25

iou-thres：0.45

最终结果（保存在badminton12）：



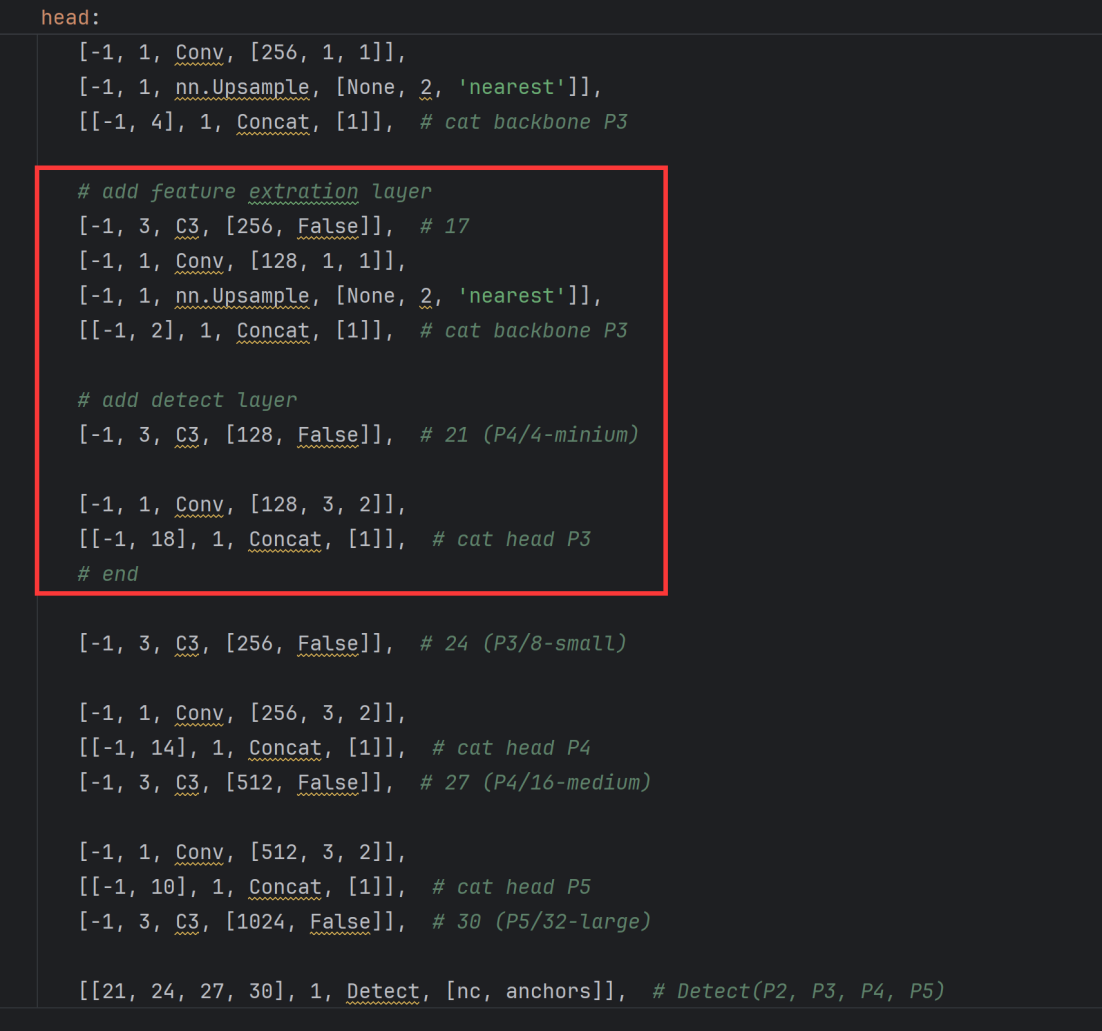
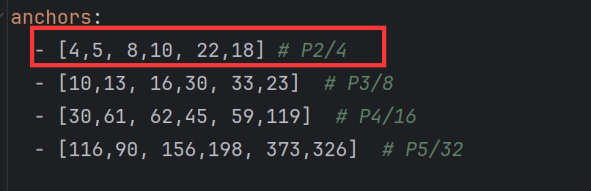
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.869 | 0.827 | 0.886 | 0.632 |
| face | 0.800 | 0.772 | 0.848 | 0.564 |
| body | 0.968 | 0.957 | 0.979 | 0.851 |
| badminton | 0.815 | 0.727 | 0.803 | 0.426 |
| racket | 0.892 | 0.853 | 0.914 | 0.687 |

以上结果显示，yolov5s初始模型对于大物体的识别效果较好，而对于小物体（badminton、face）识别效果较差，羽毛球的mAP是最低的，并且对于所有的类别来说，精度都高于召回率。

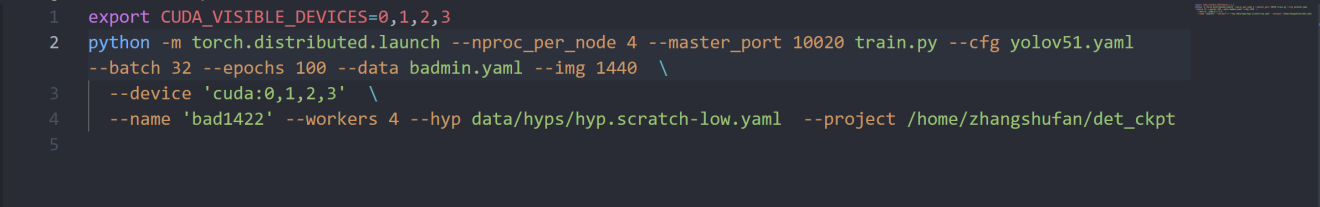
1. 使用yolov51进行训练（添加了一个P2层，并且添加了一组anchor）

任务号：yolo11

如图：

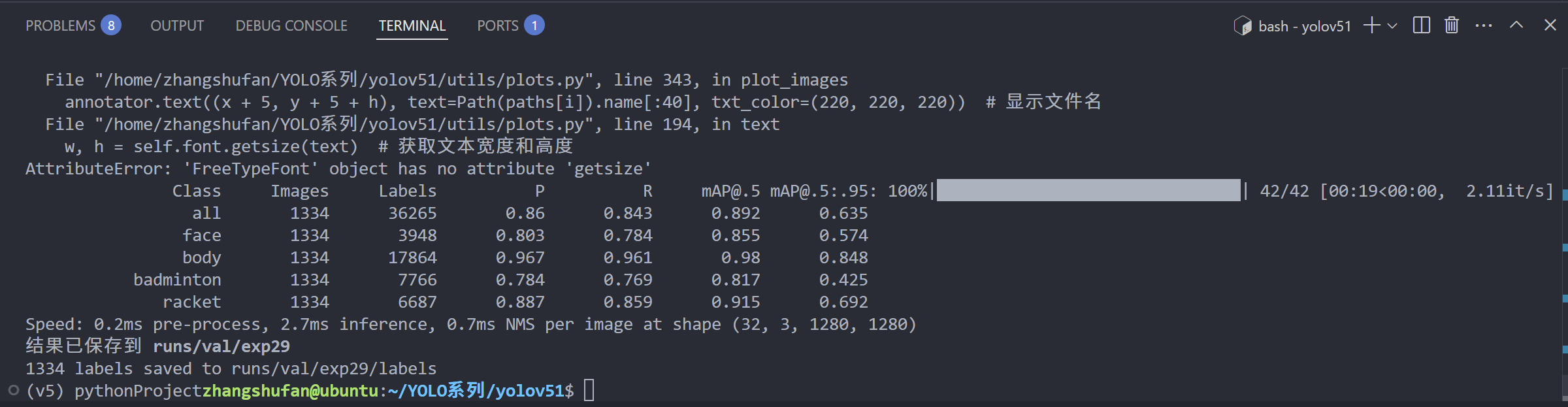


训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton0）：

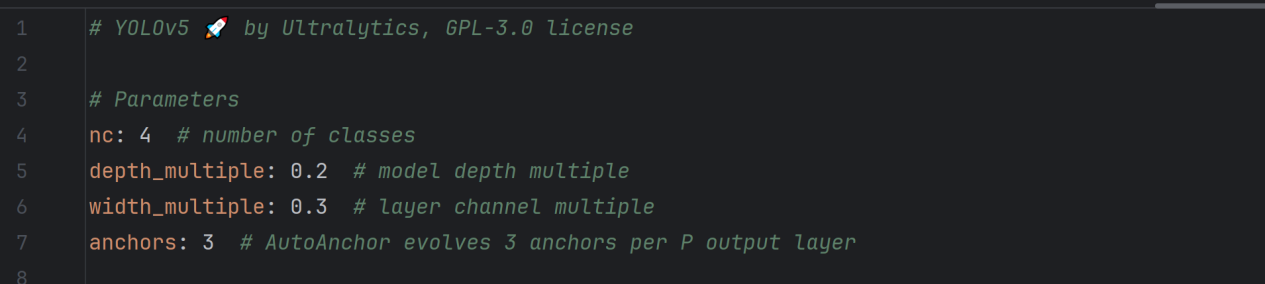


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.860 | 0.843 | 0.892 | 0.635 |
| face | 0.803 | 0.784 | 0.855 | 0.574 |
| body | 0.967 | 0.961 | 0.980 | 0.848 |
| badminton | 0.784 | 0.769 | 0.817 | 0.425 |
| racket | 0.887 | 0.859 | 0.915 | 0.692 |

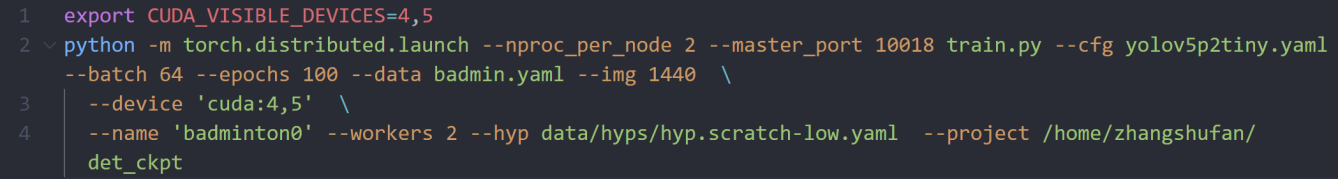
我们可以发现，加上这个P2层之后，我们模型对于小目标（badminton，face）的的召回率有了大幅度的提升，总体召回率提升了1.6%，mAP也有明显的提升，不过精度略有下降，下降了0.9%。badminton的精度下降了3.1%，召回率上升了4.2%，mAP0.5上升了1.4%。总的来说，添加一个P2层对于小目标的提升尤为明显，模型也更容易实现对目标的精确估计。

1. 使用yolov5p2tiny进行训练（将网络的深度和宽度改为0.2和0.3，移除固定anchor，添加了一个P2层）

如图（P2层和上面的一样）：

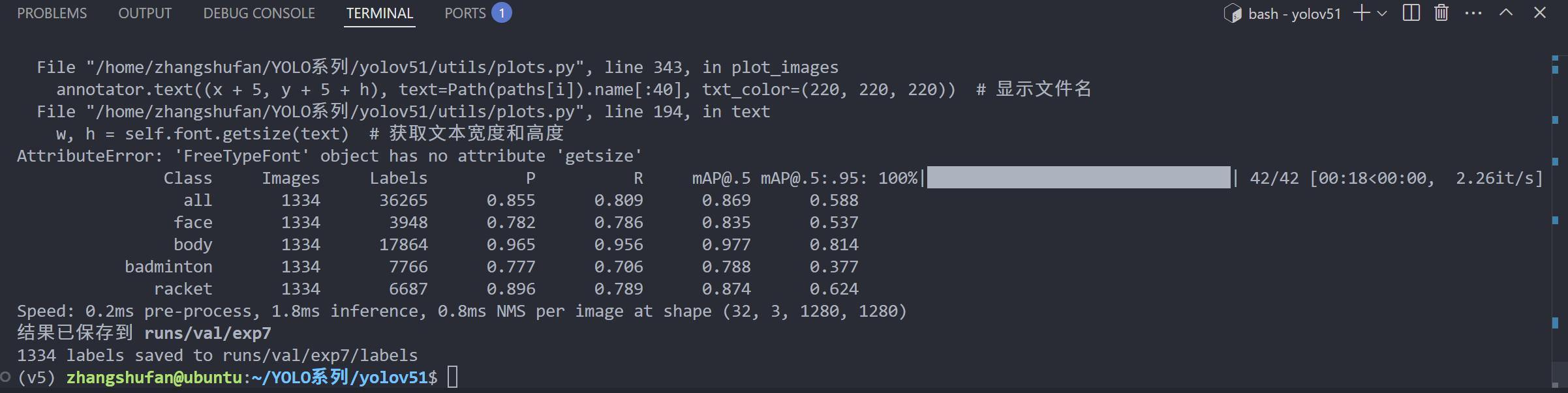


训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton04）：

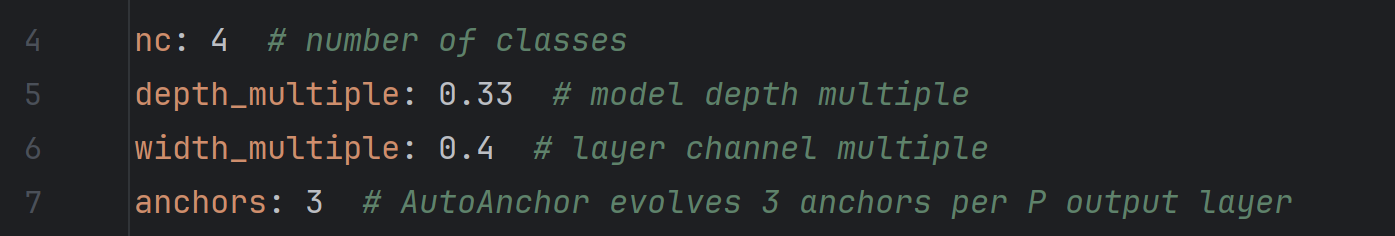


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.855 | 0.809 | 0.869 | 0.588 |
| face | 0.782 | 0.786 | 0.835 | 0.537 |
| body | 0.965 | 0.956 | 0.977 | 0.814 |
| badminton | 0.777 | 0.706 | 0.788 | 0.377 |
| racket | 0.896 | 0.789 | 0.874 | 0.624 |

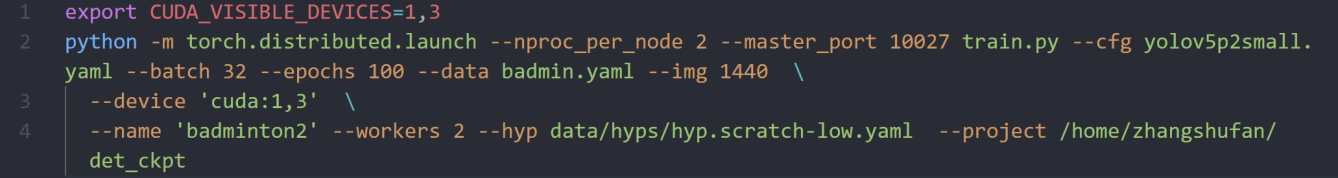
网络改小之后，总体精度和召回率都有下降，不过对于badminton的精度和召回率都有明显下降，mAP0.5:0.95也有明显下降。

1. 使用yolov5p2small进行训练（将网络的深度和宽度改为0.33和0.4，移除固定anchor，添加了一个P2层）

如图（P2层和上面的一样）：

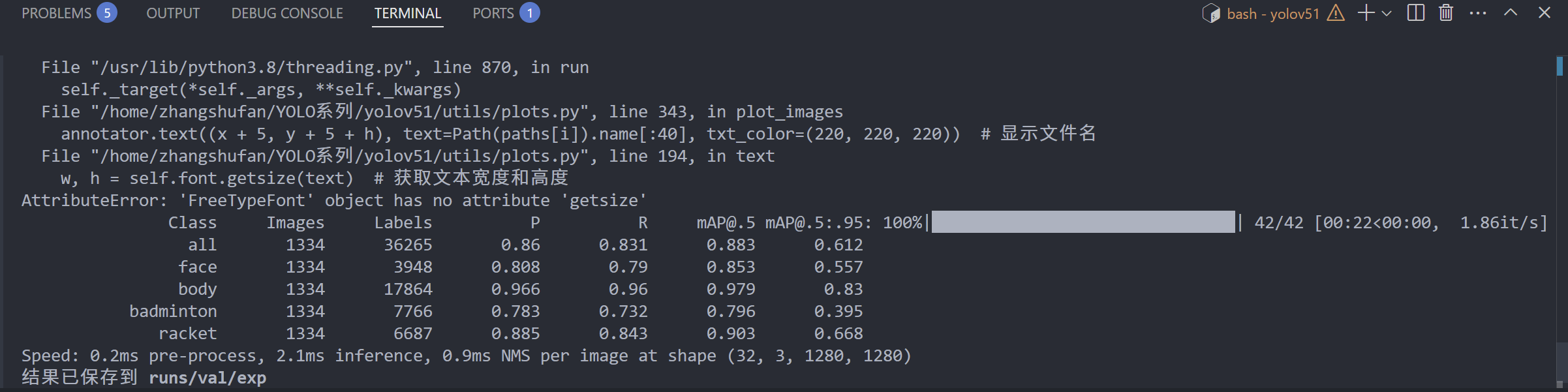


训练参数：



Batchsize为32，输入图像大小1440，训练轮数100

最终结果（保存在badminton27）：



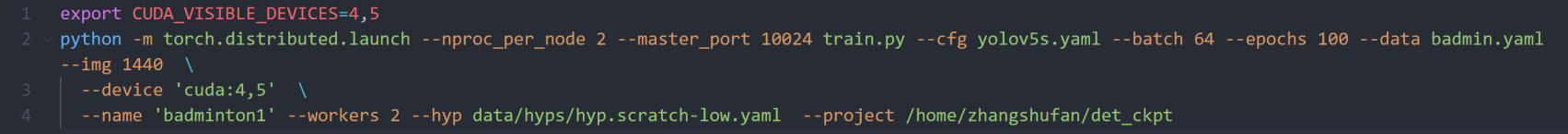
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.860 | 0.831 | 0.883 | 0.612 |
| face | 0.808 | 0.790 | 0.853 | 0.557 |
| body | 0.966 | 0.960 | 0.979 | 0.830 |
| badminton | 0.783 | 0.732 | 0.796 | 0.395 |
| racket | 0.885 | 0.843 | 0.903 | 0.668 |

这个训练结果，对比之前加了P2的正常的s模型网络，我们发现，精度和那个几乎一样，但是召回率下降了一点，总体召回率下降了1.2%，badminton召回率下降了3.7%，但是face的精度和召回率都有一点上升，或许网络改小一点能有利于face的提升，但是对于badminton一定是下降。

1. 加上Focalloss

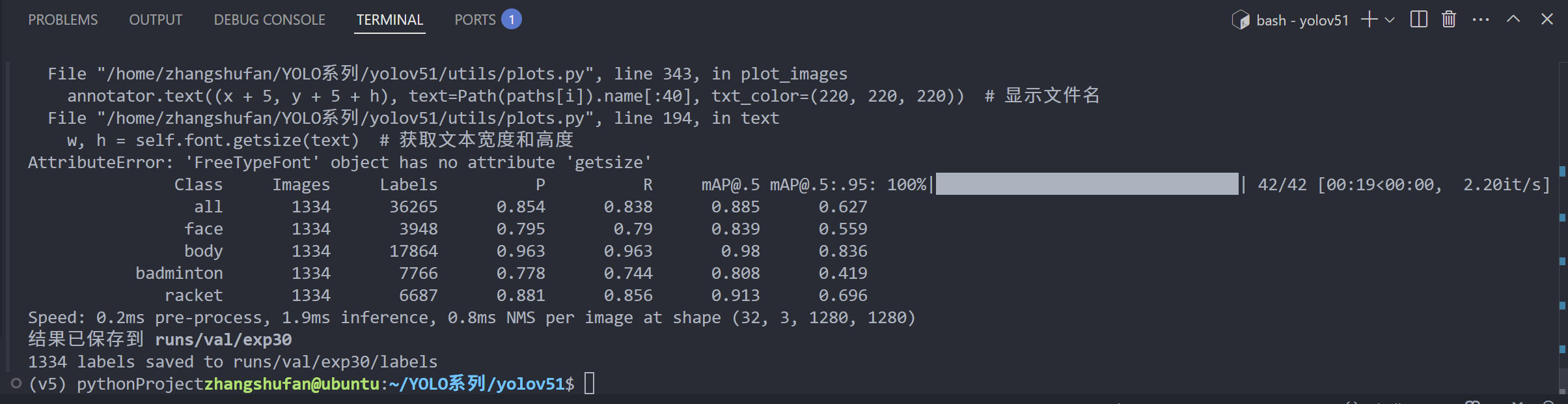
将超参数fl\_gamma改为1.5即可

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton13）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.854 | 0.838 | 0.885 | 0.627 |
| face | 0.795 | 0.790 | 0.839 | 0.559 |
| body | 0.963 | 0.985 | 0.980 | 0.836 |
| badminton | 0.778 | 0.744 | 0.808 | 0.419 |
| racket | 0.881 | 0.856 | 0.913 | 0.696 |

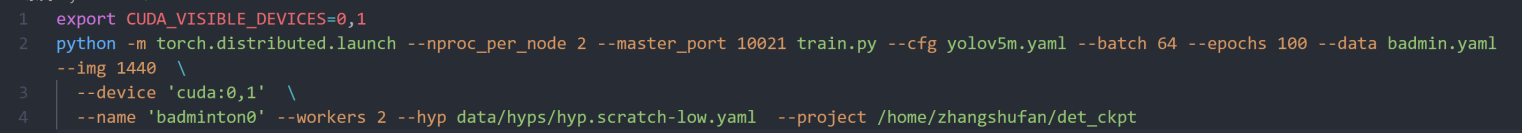
加上Focal loss之后，模型精度出现了下降，召回率出现了上升，总体精度下降了1.5%，召回率上升了1.1%。对于badminton来说精度下降的有点多，下降了3.7% ，召回率上升了1.7%。（数据集中好像并不涉及到类别不平衡的问题）。加上Focal loss对本实验并不能产生任何提升。

不过Focal loss有助于减小训练时的波动。

1. 使用yolov5m模型

GFLOPs：49.0

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton02）：  


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.852 | 0.847 | 0.891 | 0.643 |
| face | 0.779 | 0.792 | 0.849 | 0.577 |
| body | 0.964 | 0.960 | 0.980 | 0.853 |
| badminton | 0.791 | 0.755 | 0.809 | 0.429 |
| racket | 0.874 | 0.883 | 0.924 | 0.710 |

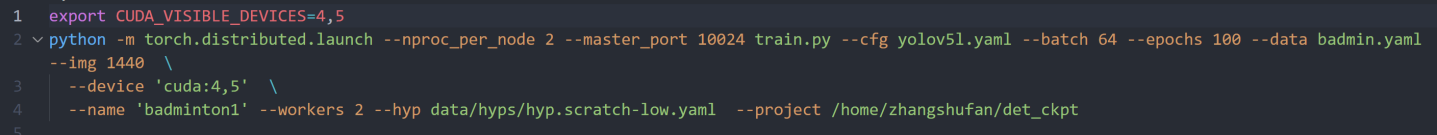
从结果我们可以明显看出，yolov5m相比yolov5s有了在召回率方面有了明显提升，提升了2.0%，但是精度也出现了明显下降，下降了1.6%。mAP0.5有了小幅度提升，mAP0.5:0.95大幅度提升，有1.1%，对于复杂细节的目标提升较大，不过训练时间也延长了。

并且我发现大一点的模型会存在过拟合的现象。

1. 使用yolov5l模型

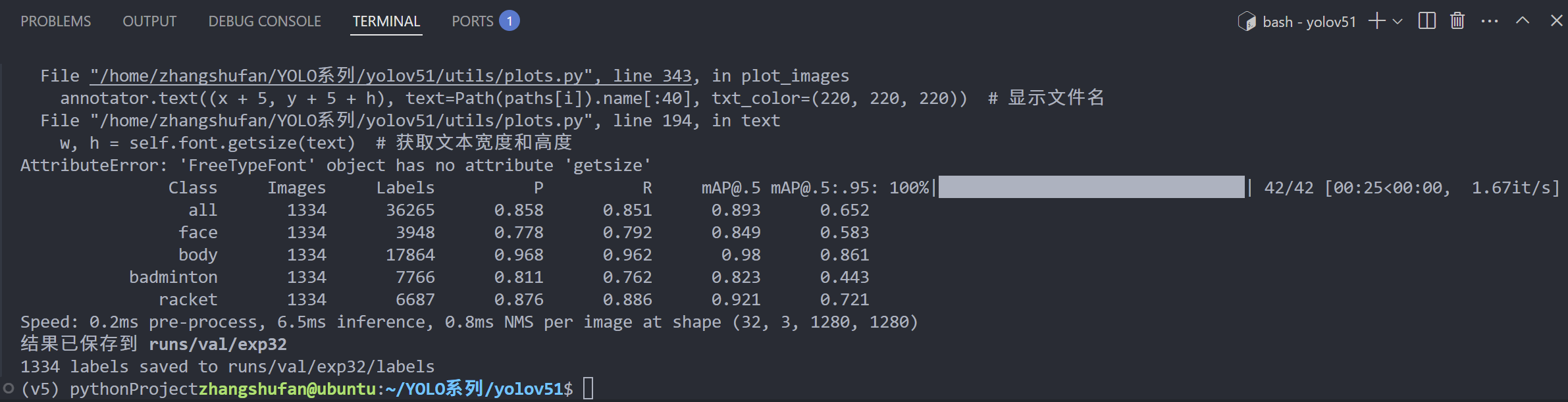
GFLOPs：109.1

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton15）：



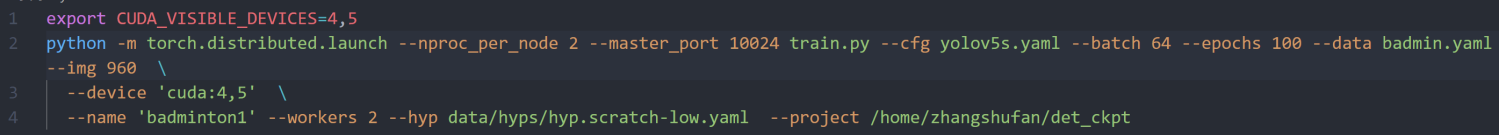
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.858 | 0.851 | 0.893 | 0.652 |
| face | 0.778 | 0.792 | 0.849 | 0.583 |
| body | 0.968 | 0.962 | 0.980 | 0.861 |
| badminton | 0.811 | 0.762 | 0.823 | 0.443 |
| racket | 0.876 | 0.886 | 0.921 | 0.721 |

从结果来看，yolov5l相比yolov5s，精度有了微小下降，但是召回率出现了大幅提升。mAP0.5和mAP0.5:0.95都出现了提升，尤其是mAP0.5:0.95。但是这个模型太大，可能不符合我们任务的需求。

更大更复杂的模型还有个坏处就是，容易发生过拟合。并且训练的时候波动相对较大。

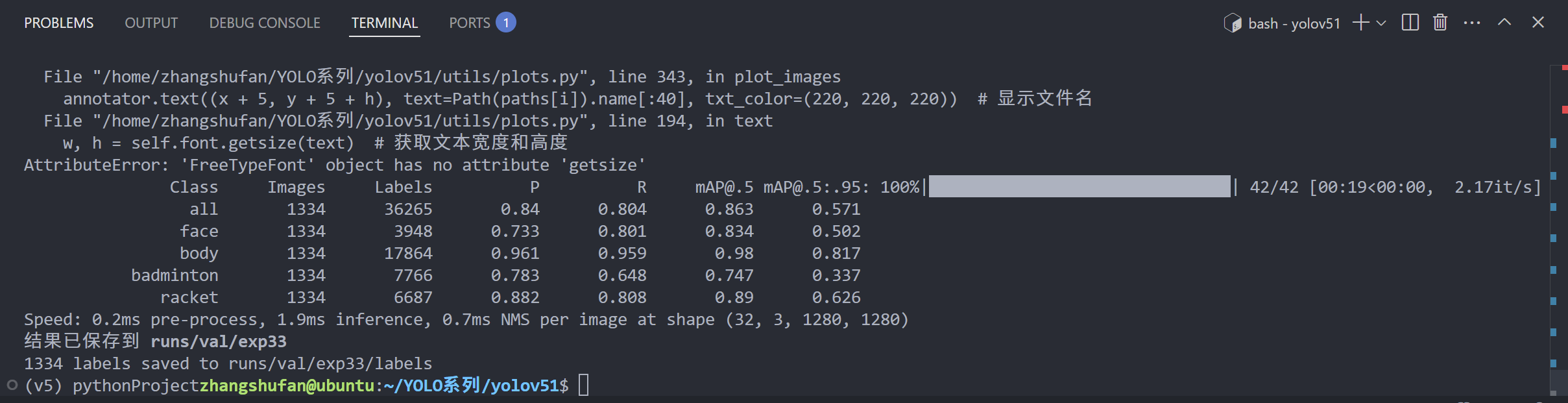
1. 分辨率调整至960

训练参数：



Batchsize为64，输入图像大小960，训练轮数100

最终结果（保存在badminton16）：

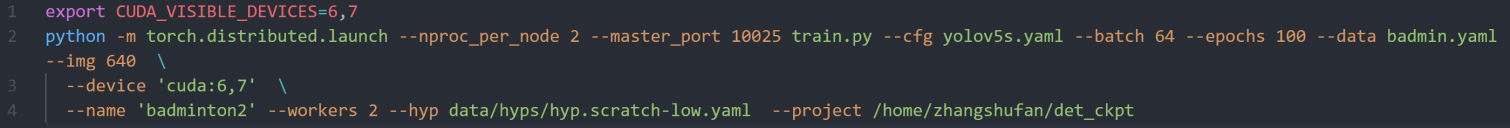


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.840 | 0.804 | 0.863 | 0.571 |
| face | 0.733 | 0.801 | 0.834 | 0.502 |
| body | 0.961 | 0.959 | 0.980 | 0.817 |
| badminton | 0.783 | 0.648 | 0.747 | 0.337 |
| racket | 0.882 | 0.808 | 0.890 | 0.626 |

从结果上看，分辨率下降到960之后，总体精度和召回率都下降相当大，mAP0.5也有大幅度下降，但是下降幅度最大的还是mAP0.5:0.95。对于小目标，尤其是badminton，下降幅度也特别大。分辨率降低导致细节丢失，对于精细检测（mAP0.5:0.95）的性能下降很多。

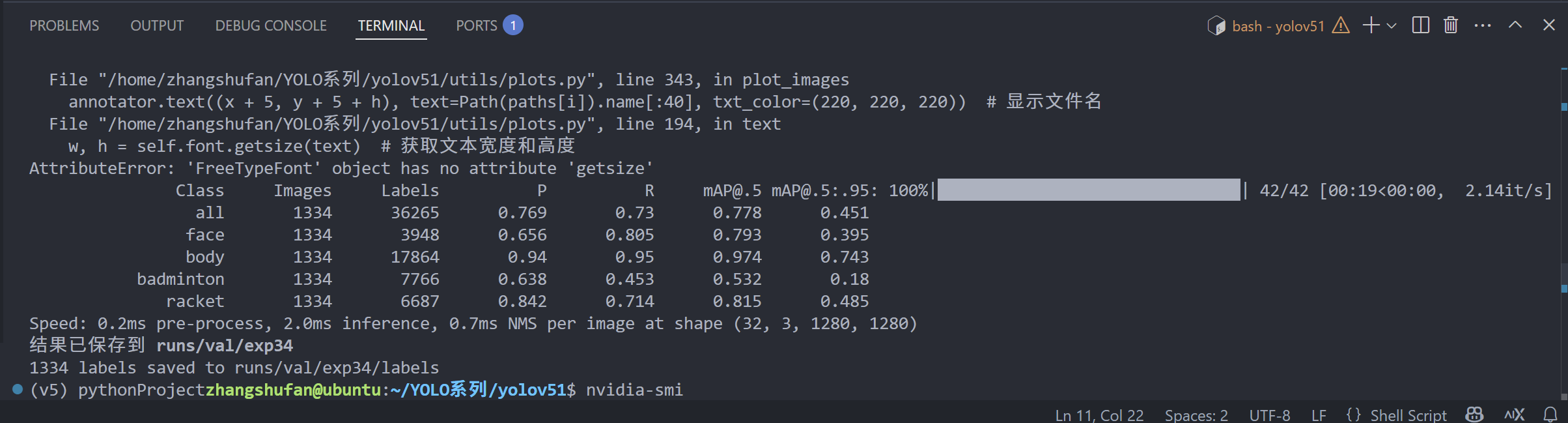
1. 分辨率调整至640

训练参数：



Batchsize为64，输入图像大小640，训练轮数100

最终结果（保存在badminton2）：

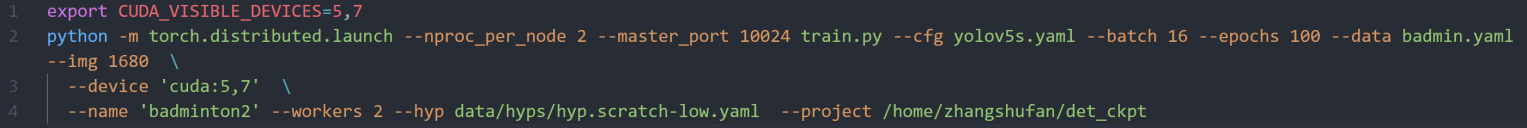


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.769 | 0.730 | 0.778 | 0.451 |
| face | 0.656 | 0.805 | 0.793 | 0.395 |
| body | 0.940 | 0.950 | 0.974 | 0.743 |
| badminton | 0.638 | 0.453 | 0.532 | 0.180 |
| racket | 0.842 | 0.714 | 0.815 | 0.485 |

除了body检测性能基本保持不变，其他都超大幅度下降。

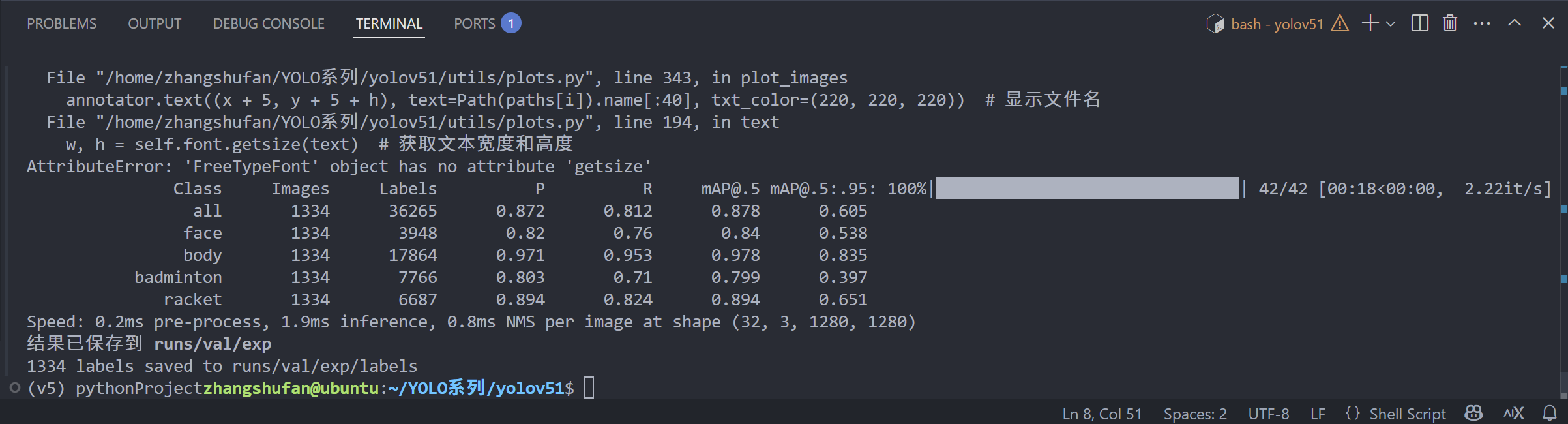
1. 分辨率调整至1680

训练参数：



Batchsize为64，输入图像大小1680，训练轮数100

最终结果（保存在badminton24）：

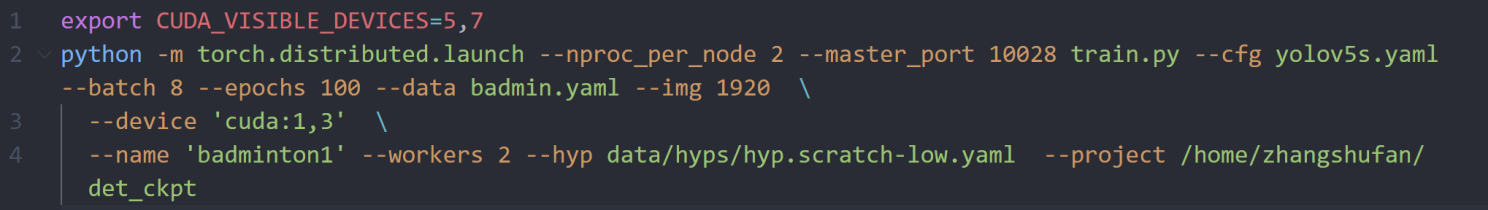


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.872 | 0.812 | 0.878 | 0.605 |
| face | 0.820 | 0.760 | 0.840 | 0.538 |
| body | 0.971 | 0.953 | 0.978 | 0.835 |
| badminton | 0.803 | 0.710 | 0.799 | 0.397 |
| racket | 0.894 | 0.824 | 0.894 | 0.651 |

在分辨率1680下训练，但是用1280推理的时候，我发现精度几乎保持不变，但是召回率下降了比较多，总体召回率下降了1.5%。对于badminton来说，精度和召回率都出现了一定幅度的下降，而且mAP0.5:0.95也出现了一定幅度的下降，说明模型精细预测能力出现了下降。

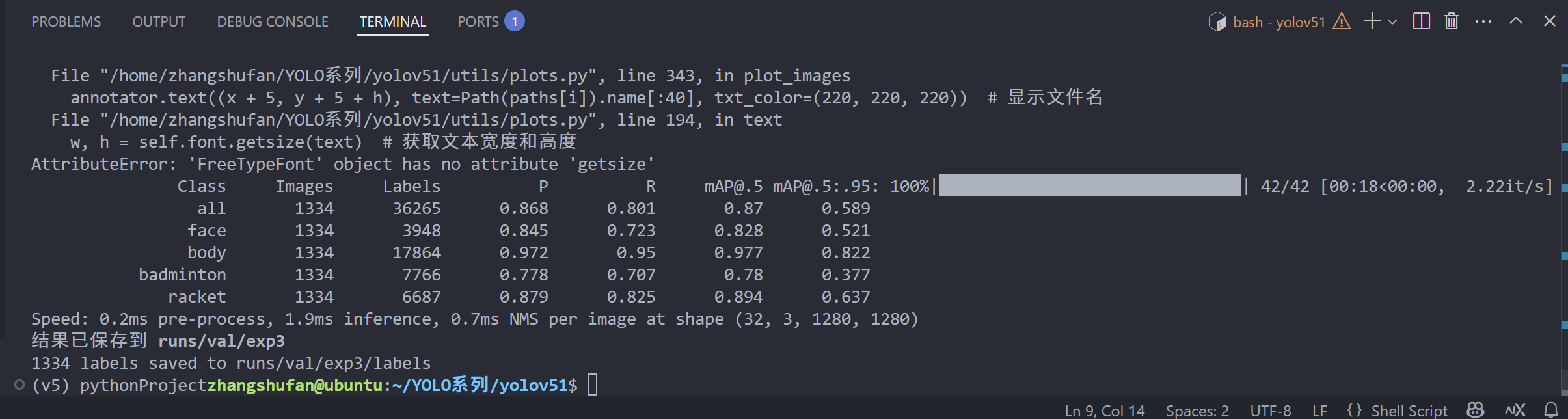
1. 分辨率调整至1920

训练参数：



Batchsize为64，输入图像大小1920，训练轮数100

最终结果（保存在badminton114）：

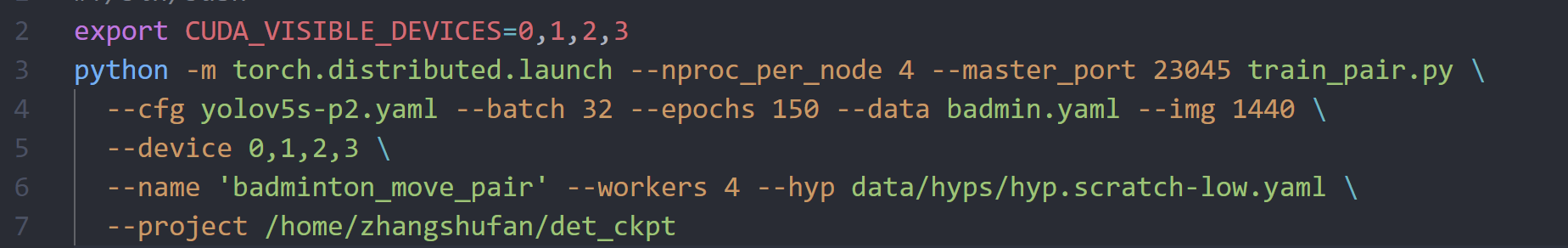


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.868 | 0.801 | 0.870 | 0.589 |
| face | 0.845 | 0.723 | 0.828 | 0.521 |
| body | 0.972 | 0.950 | 0.977 | 0.822 |
| badminton | 0.778 | 0.707 | 0.780 | 0.377 |
| racket | 0.879 | 0.825 | 0.894 | 0.637 |

在分辨率1920下训练，并且在1280上推理，从结果能看出召回率下降了非常多，总体召回率下降了2.6%，并且mAP0.5:0.95也下降的非常厉害，下降了4.3%。这么看来，模型在1920上的训练结果还不如1440上的训练结果。

1. 双图检测（5个类别）

训练参数：



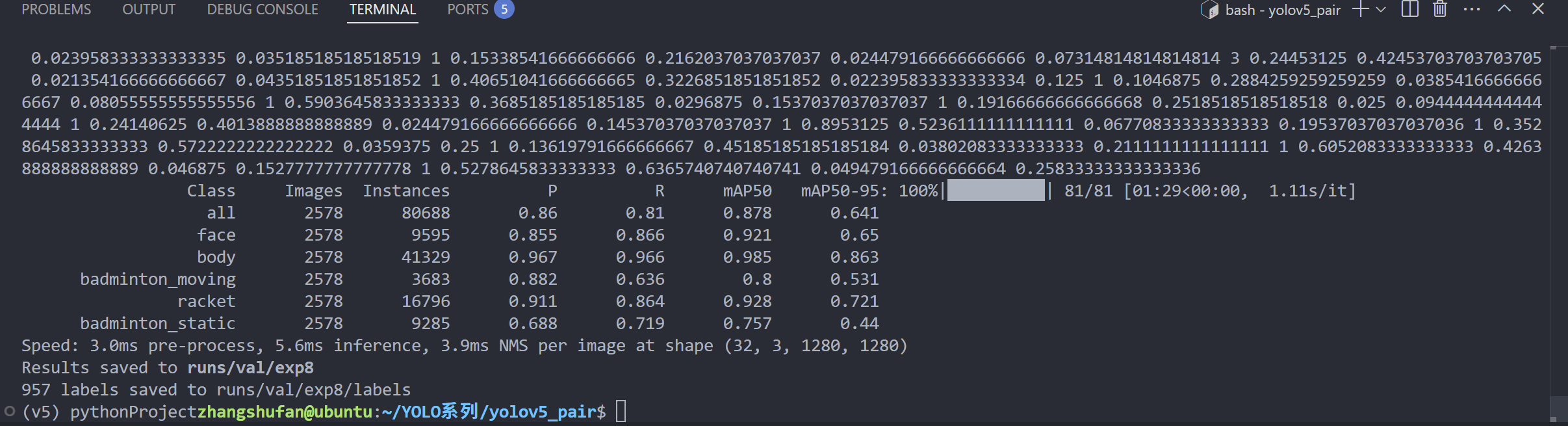
Batchsize为32，输入图像大小1440，训练轮数150

网络模型：采用yolov5-p2模型

数据集：训练集大约77460张图片，验证集大约2578张图片

推理参数不变，放在1280上进行推理

最终结果（保存在badminton\_move\_pair7）：



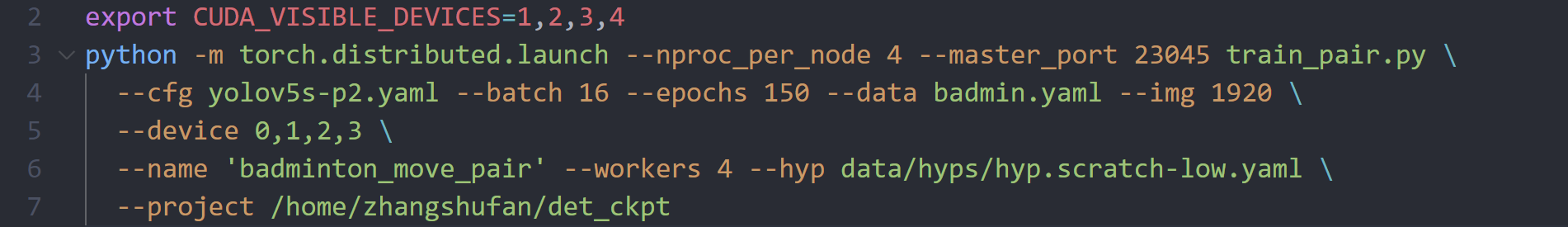
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.860 | 0.810 | 0.878 | 0.641 |
| face | 0.855 | 0.866 | 0.921 | 0.650 |
| body | 0.967 | 0.966 | 0.985 | 0.863 |
| badminton\_moving | 0.882 | 0.636 | 0.800 | 0.531 |
| racket | 0.879 | 0.825 | 0.894 | 0.637 |
| badminton\_static | 0.688 | 0.719 | 0.757 | 0.440 |

从结果来看，face相比之前的训练结果有了明显提升（这也许是因为数据集扩充的结果），模型区分badminton\_moving和badminton\_static的准确性还是不够高，特别是对于badminton\_static，准确性还有待提高。

不过从结果来看，双图检测还是有明显的效果，如果是单图检测的话，是几乎无法区分moving和static的羽毛球的。

1. 双图检测，分辨率调整至1920

训练参数：



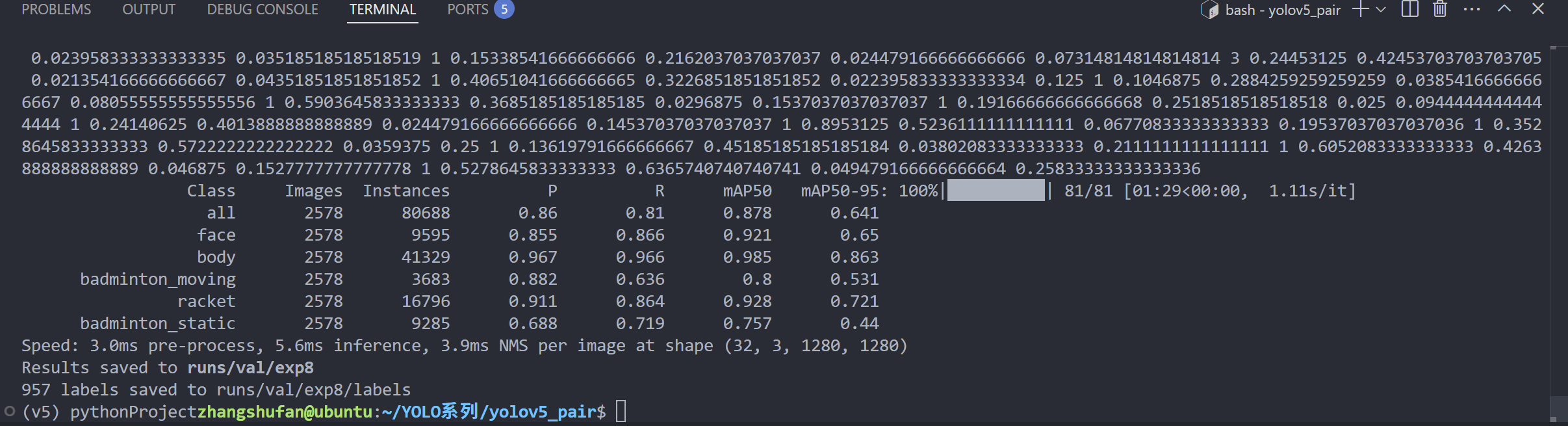
Batchsize为16，输入图像大小1920，训练轮数150

网络模型：采用yolov5-p2模型

数据集：训练集大约77460张图片，验证集大约2578张图片

推理参数不变，放在1280上进行推理

最终结果（保存在badminton\_move\_pair8）：



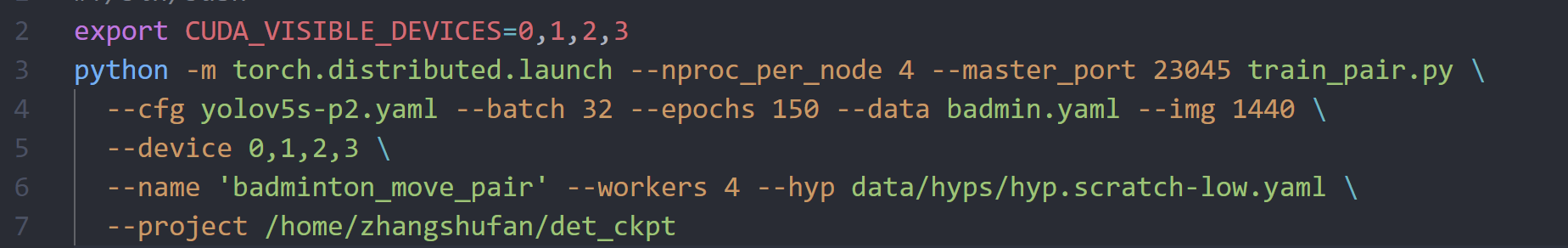
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.860 | 0.810 | 0.878 | 0.641 |
| face | 0.855 | 0.866 | 0.921 | 0.650 |
| body | 0.967 | 0.966 | 0.985 | 0.863 |
| badminton\_moving | 0.882 | 0.636 | 0.800 | 0.531 |
| racket | 0.879 | 0.825 | 0.894 | 0.637 |
| badminton\_static | 0.688 | 0.719 | 0.757 | 0.440 |

从结果来看，face相比之前的训练结果有了明显提升（这也许是因为数据集扩充的结果），模型区分badminton\_moving和badminton\_static的准确性还是不够高，特别是对于badminton\_static，准确性还有待提高。

不过从结果来看，双图检测还是有明显的效果，如果是单图检测的话，是几乎无法区分moving和static的羽毛球的。

1. 三图检测（5个类别）

训练参数：

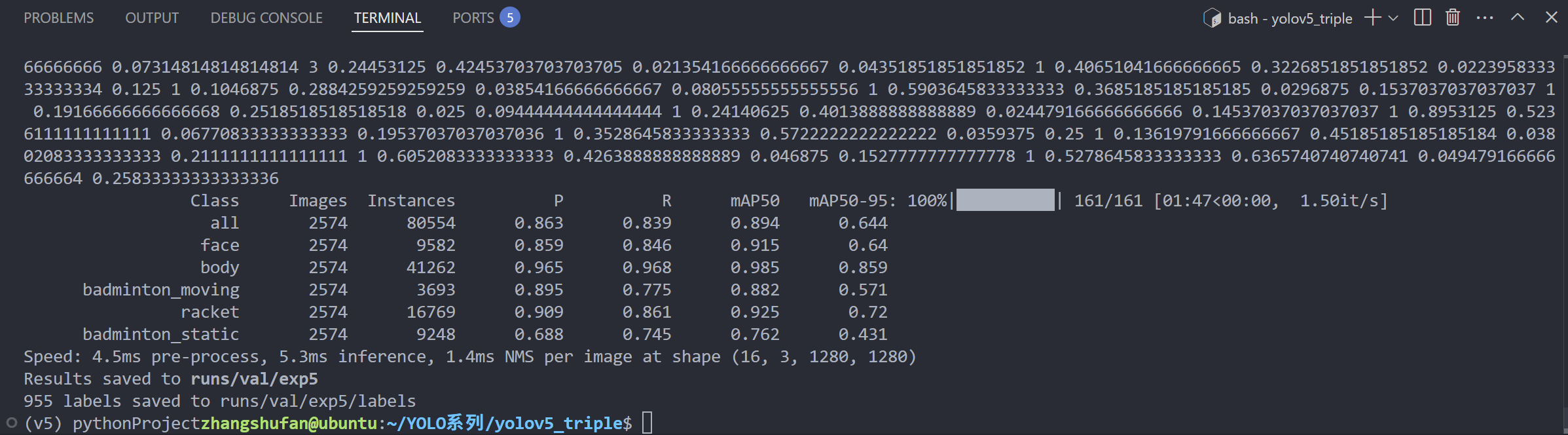


Batchsize为32，输入图像大小1440，训练轮数150

网络模型：采用yolov5-p2模型

数据集：训练集大约77460张图片，验证集大约2578张图片

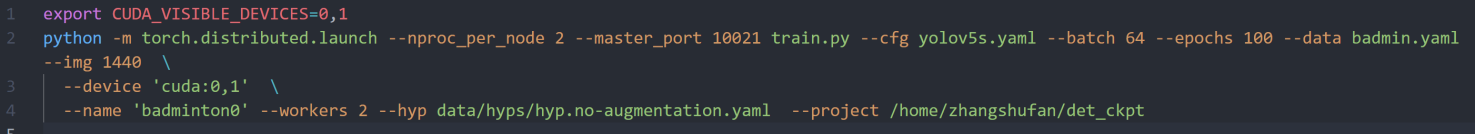
最终结果（保存在badminton\_move\_triple5）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.863 | 0.839 | 0.894 | 0.644 |
| face | 0.859 | 0.846 | 0.915 | 0.640 |
| body | 0.967 | 0.966 | 0.985 | 0.859 |
| badminton\_moving | 0.895 | 0.775 | 0.882 | 0.571 |
| racket | 0.909 | 0.861 | 0.925 | 0.720 |
| badminton\_static | 0.688 | 0.745 | 0.762 | 0.431 |

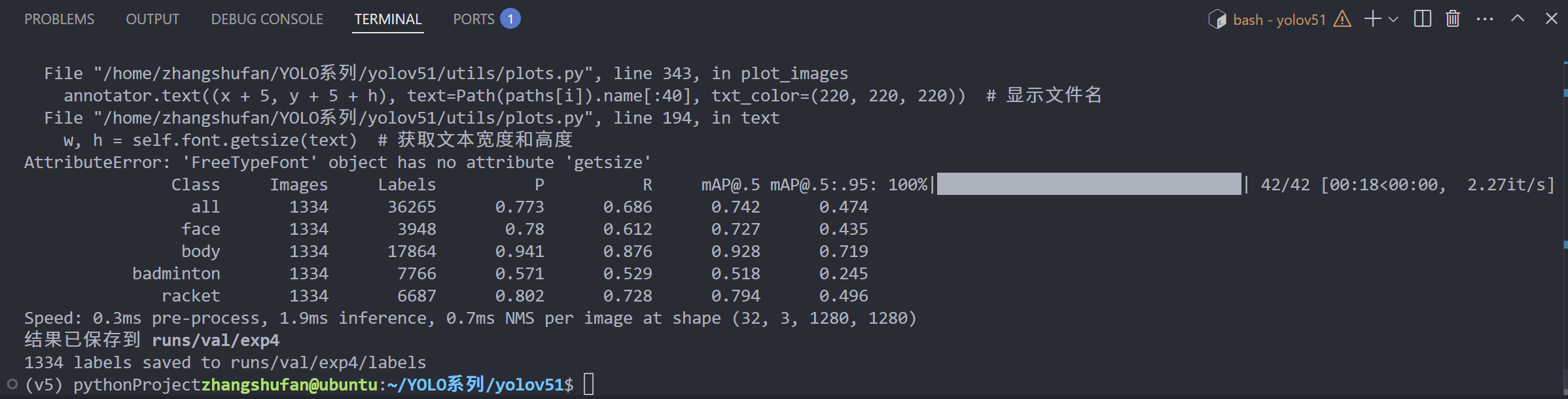
1. 数据增强，使用hyp.no-augmentation.yaml（无数据增强）

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton03）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.773 | 0.686 | 0.742 | 0.474 |
| face | 0.780 | 0.612 | 0.727 | 0.435 |
| body | 0.941 | 0.876 | 0.928 | 0.719 |
| badminton | 0.571 | 0.529 | 0.518 | 0.245 |
| racket | 0.802 | 0.728 | 0.794 | 0.496 |

不使用数据增强的话，各方面性能都下降的厉害，就连body的检测性能都下降的很厉害。

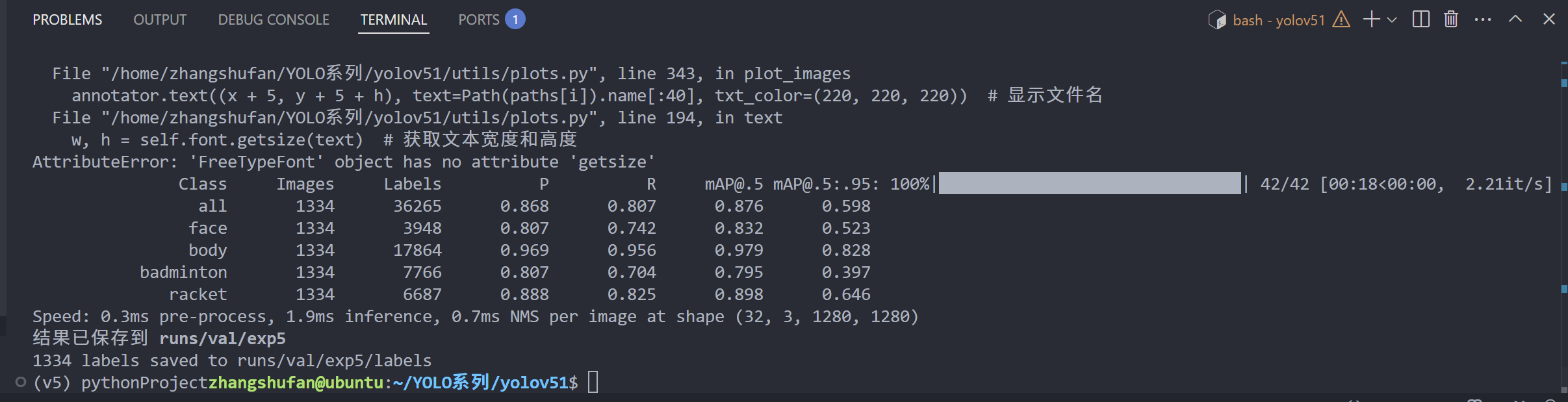
1. 数据增强，使用hyp.scratch-med.yaml（中等数据增强）

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton17）：



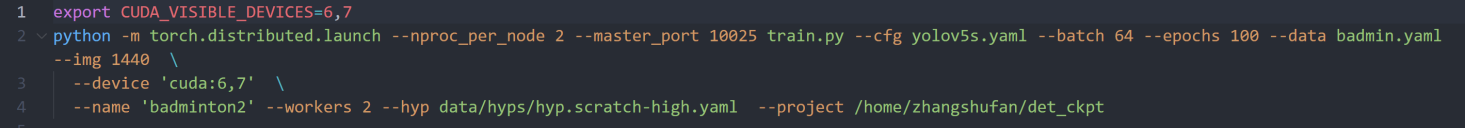
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.868 | 0.807 | 0.876 | 0.598 |
| face | 0.807 | 0.742 | 0.832 | 0.523 |
| body | 0.969 | 0.956 | 0.979 | 0.828 |
| badminton | 0.807 | 0.704 | 0.795 | 0.397 |
| racket | 0.888 | 0.825 | 0.898 | 0.646 |

中等数据增强，总体精度基本不变，召回率相比原始结果还是下降比较大，对于mAP0.5的变化不大，但是对于mAP0.5:0.95有较大的下降幅度。中等数据增强反正会降低模型对于精细检测的性能。

使用更高等级的数据增强的好处就是可以防止过拟合，使训练曲线更加平滑。

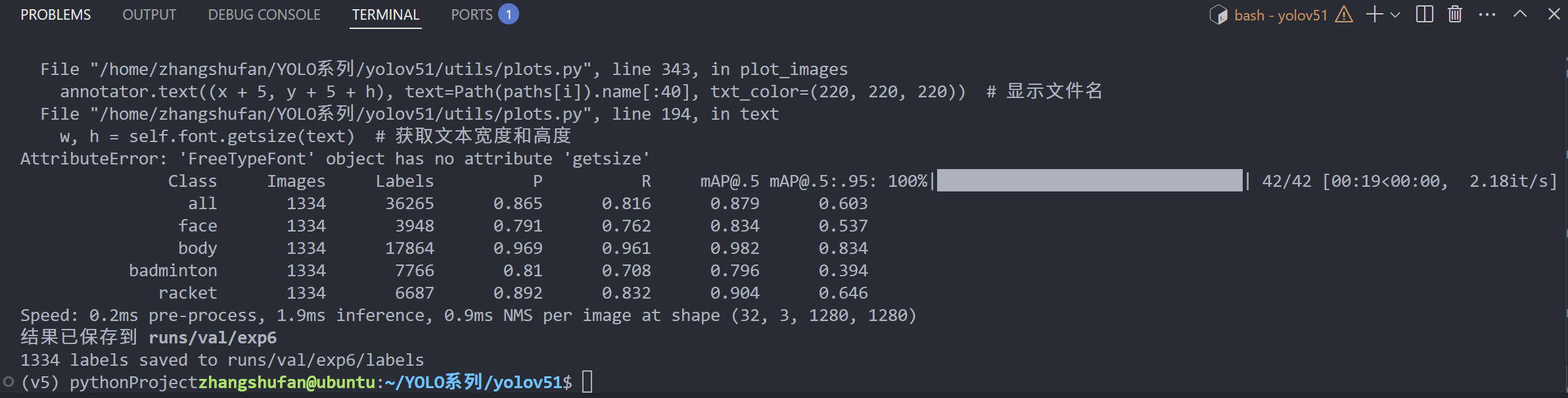
1. 数据增强，使用hyp.scratch-high.yaml（高数据增强）

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton22）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.865 | 0.816 | 0.879 | 0.603 |
| face | 0.791 | 0.762 | 0.834 | 0.537 |
| body | 0.969 | 0.961 | 0.982 | 0.834 |
| badminton | 0.810 | 0.708 | 0.796 | 0.394 |
| racket | 0.892 | 0.832 | 0.904 | 0.646 |

使用高级数据增强之后，精度几乎保持不变，召回率相比中等数据增强稍有上升，但是还是和原始结果有差距，特别是badminton。高数据增强的好处是可以减小训练时的波动，防止过拟合的发生。

1. 不同代版本，使用yolov8n进行训练

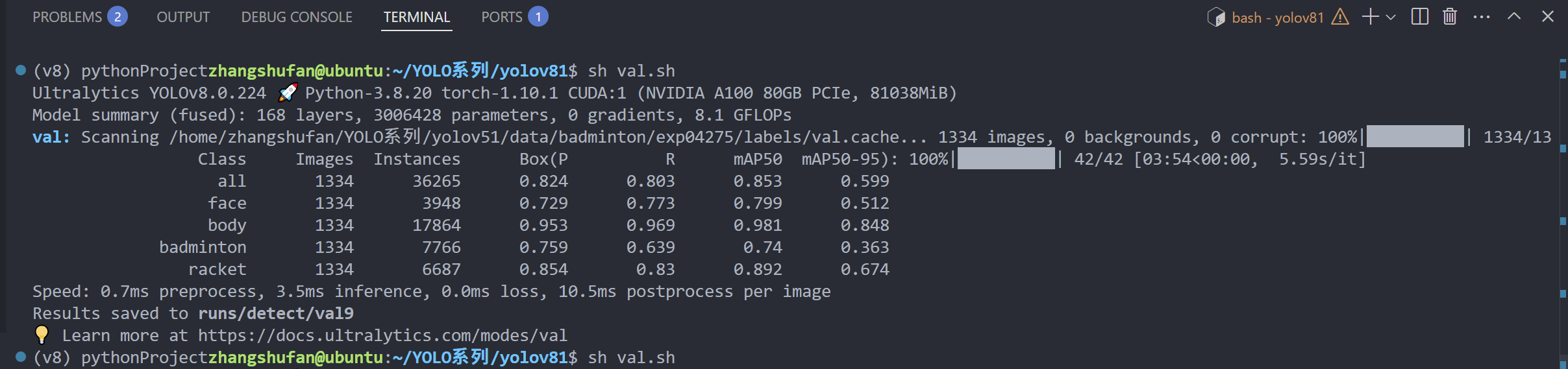
GFLOPs：8.9

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton34）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.824 | 0.803 | 0.853 | 0.599 |
| face | 0.729 | 0.773 | 0.799 | 0.512 |
| body | 0.953 | 0.969 | 0.981 | 0.848 |
| badminton | 0.759 | 0.639 | 0.740 | 0.363 |
| racket | 0.854 | 0.830 | 0.892 | 0.674 |

根据训练结果，我们可以发现，使用yolov8n训练出来的结果并不理想，所有指标相比yolov5s训练结果都下降了非常多，尤其是badminton类别下降的很厉害，这说明了yolov8n模型并不适合我们这个实验。

1. 不同代版本，使用yolov8s进行训练

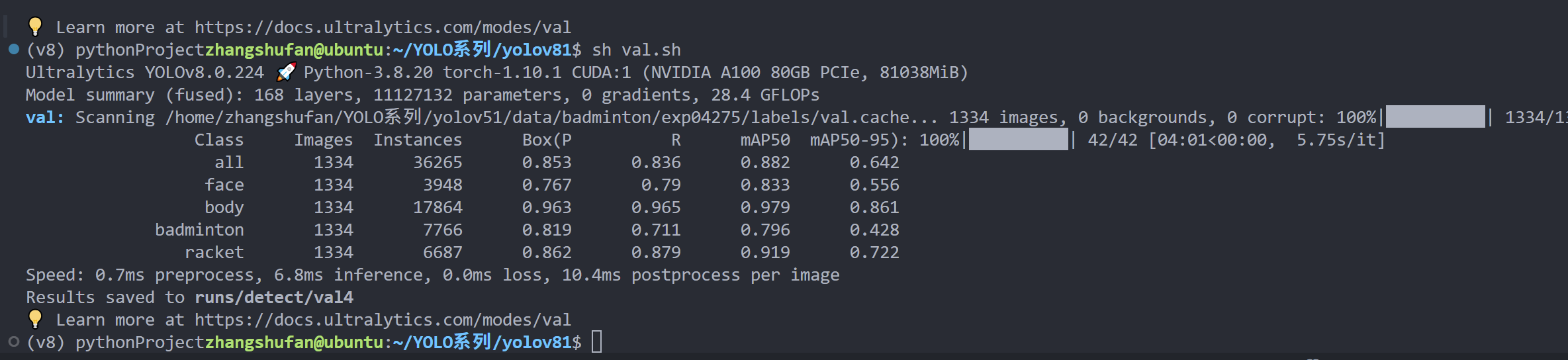
GFLOPs：28.8

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton42）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.853 | 0.836 | 0.882 | 0.642 |
| face | 0.767 | 0.790 | 0.833 | 0.556 |
| body | 0.963 | 0.965 | 0.979 | 0.861 |
| badminton | 0.819 | 0.711 | 0.796 | 0.428 |
| racket | 0.862 | 0.879 | 0.919 | 0.722 |

根据训练结果对比，总体精度比yolov5s训练结果稍微下降（1.6%），召回率稍微上升（0.9%），mAP0.5几乎不变而mAP0.5:0.95稍微上升（1.0%）。但是，对于badminton类别，精度几乎没有提升，而召回率下降了1.6%，其他指标也未提升。对于face来说，精度出现了大幅下降（3.3%），召回率出现了一定上升（1.8%），效果也不理想。这或许是因为yolov8提出的背景并不是解决小目标的问题。

1. yolov8s添加一个P2层

如图：

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

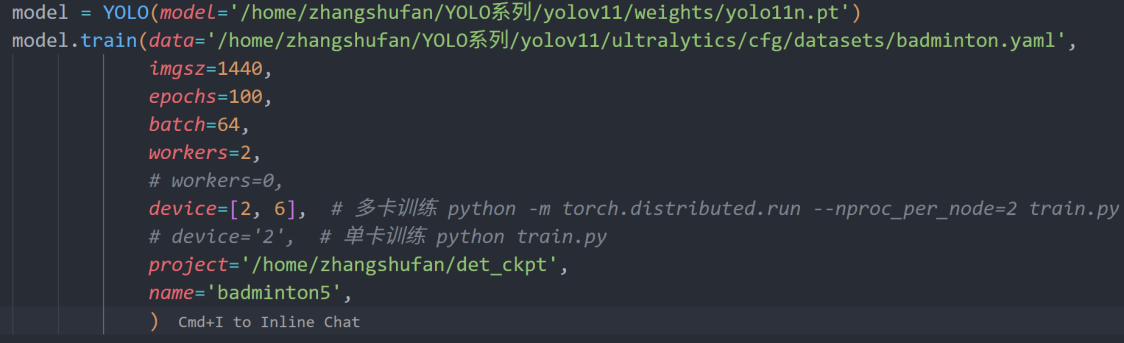
最终结果（保存在badminton）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.853 | 0.836 | 0.882 | 0.642 |
| face | 0.767 | 0.790 | 0.833 | 0.556 |
| body | 0.963 | 0.965 | 0.979 | 0.861 |
| badminton | 0.819 | 0.711 | 0.796 | 0.428 |
| racket | 0.862 | 0.879 | 0.919 | 0.722 |

1. 不同代版本，使用yolov11n进行训练

GFLOPs：6.5

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton58）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.829 | 0.815 | 0.853 | 0.609 |
| face | 0.748 | 0.767 | 0.798 | 0.525 |
| body | 0.954 | 0.966 | 0.979 | 0.851 |
| badminton | 0.756 | 0.688 | 0.749 | 0.386 |
| racket | 0.853 | 0.838 | 0.886 | 0.676 |

1. 不同代版本，使用yolov11s进行训练

GFLOPs：21.5

训练参数：



Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton64）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.854 | 0.834 | 0.878 | 0.643 |
| face | 0.764 | 0.791 | 0.831 | 0.562 |
| body | 0.967 | 0.961 | 0.977 | 0.861 |
| badminton | 0.814 | 0.710 | 0.788 | 0.432 |
| racket | 0.870 | 0.875 | 0.917 | 0.717 |

根据训练结果对比，yolov11s训练结果总体精度比yolov5s训练结果稍微下降（1.5%），召回率稍微上升（0.7%），mAP0.5小幅下降（0.8%）而mAP0.5:0.95稍微上升（1.1%）。但是，对于badminton类别，精度几乎没有变化，而召回率下降了1.7%，其他指标也未提升。对于face来说，精度出现了大幅下降（3.6%），召回率出现了一定上升（1.9%），效果也不理想。这或许也是因为yolov11提出的背景并不是解决小目标的问题。

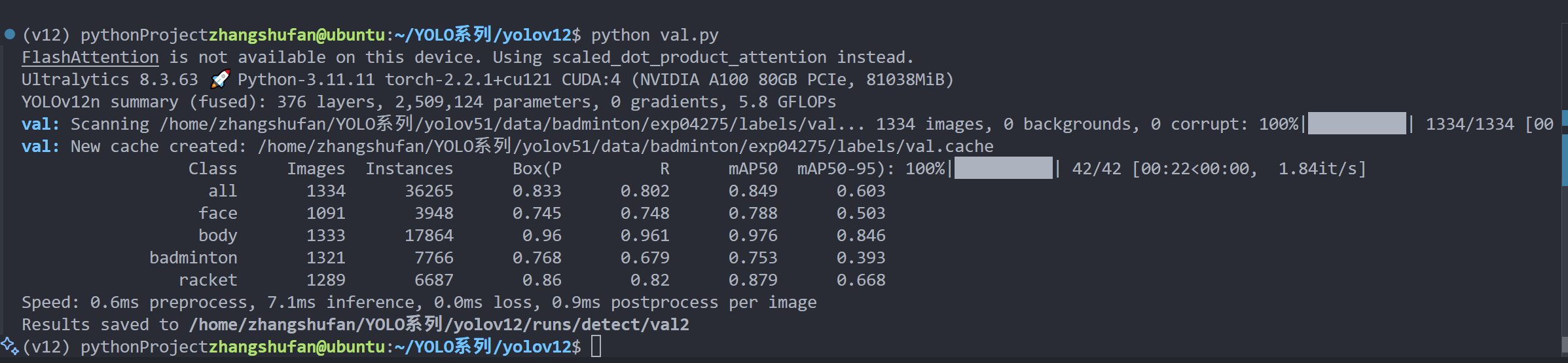
1. 不同代版本，使用yolov12n进行训练

GFLOPs：6.0

训练参数：

Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton73）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.833 | 0.802 | 0.849 | 0.603 |
| face | 0.745 | 0.748 | 0.788 | 0.503 |
| body | 0.960 | 0.961 | 0.976 | 0.846 |
| badminton | 0.768 | 0.679 | 0.753 | 0.393 |
| racket | 0.860 | 0.820 | 0.879 | 0.668 |

1. 不同代版本，使用yolov12s进行训练

GFLOPs：19.4

训练参数：

Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton83）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.836 | 0.845 | 0.876 | 0.640 |
| face | 0.729 | 0.797 | 0.819 | 0.549 |
| body | 0.961 | 0.969 | 0.980 | 0.861 |
| badminton | 0.800 | 0.738 | 0.796 | 0.438 |
| racket | 0.855 | 0.875 | 0.910 | 0.714 |

根据训练结果对比，yolov12s训练结果的精度比较低，但是召回率很高，并且召回率超过了精度（之前的模型都是精度>召回率），总体精度比yolov5s训练结果稍微下降3.3%，召回率上升1.8%，mAP0.5小幅下降（1.0%）而mAP0.5:0.95稍微上升（0.8%）。对于badminton类别，精度下降了1.5%，而召回率上升了1.1%，mAP0.5:0.95上升了1.2%。对于face来说，精度出现了大幅下降（7.1%），召回率出现了一定上升（2.5%）。yolov12s的总体表现仍然不是很理想

1. 不同代版本，使用yolov10n进行训练

GFLOPs：6.7

训练参数：

Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton92）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.826 | 0.790 | 0.847 | 0.575 |
| face | 0.758 | 0.741 | 0.809 | 0.492 |
| body | 0.949 | 0.957 | 0.984 | 0.829 |
| badminton | 0.742 | 0.645 | 0.700 | 0.336 |
| racket | 0.854 | 0.817 | 0.896 | 0.642 |

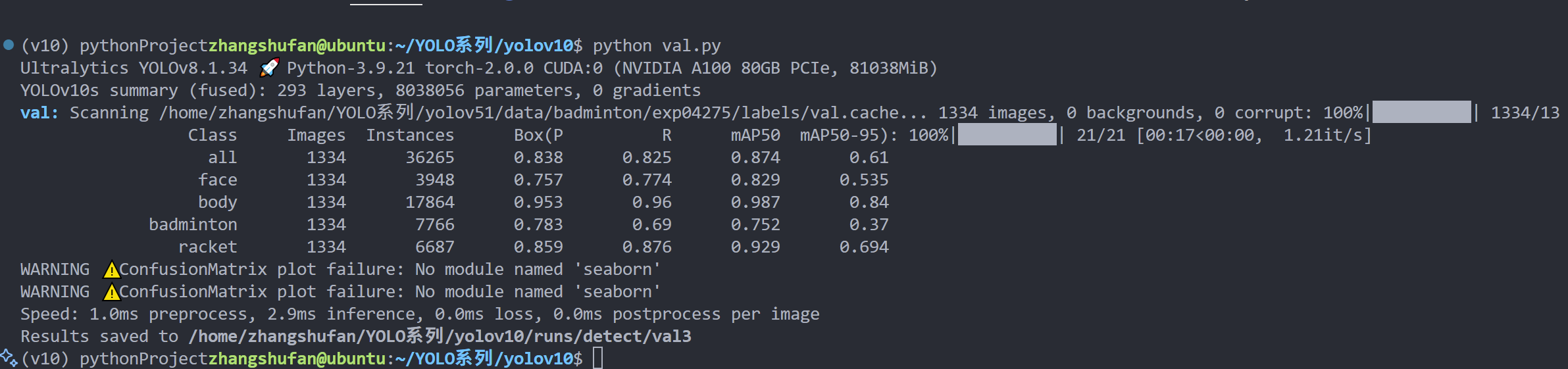
1. 不同代版本，使用yolov10s进行训练

GFLOPs：21.6

训练参数：

Batchsize为64，输入图像大小1440，训练轮数100

最终结果（保存在badminton93）：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P | R | mAP0.5 | mAP0.5：0.95 |
| all | 0.838 | 0.825 | 0.874 | 0.610 |
| face | 0.757 | 0.774 | 0.829 | 0.535 |
| body | 0.953 | 0.960 | 0.987 | 0.840 |
| badminton | 0.783 | 0.690 | 0.752 | 0.370 |
| racket | 0.859 | 0.876 | 0.929 | 0.694 |