dell AI交通标志识别比赛

一、注意：

·代码已经在<https://github.com/caiyancheng/dell_yoloV5>开源，此处实验记录（./exps）非常全，欢迎下载（注意：其中GAN的finetune的代码进行了部分保密处理，在yolo.py处的接口被删除（接口可以在yolo\_new.py看），但是为了方便理解，我会配图）

·实验已经在<https://wandb.ai/caiyancheng/projects>可视化（此处会比较乱，可以观察实验过程，希望看实验结果可以直接看github中的实验记录）

·跨域模块中我们用了cityscapes和foggy cityscapes数据集进行了有效性验证

二、参赛人员及部分基本信息

1、参赛学校及参赛人员：

（1）参赛学校：复旦大学

（2）参赛人员：蔡彦成、姚志毅、周恩宇

2、环境：

（1）硬件：服务器

（2）操作系统：

Linux version 4.4.0-142-generic (buildd@lgw01-amd64-033) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10) ) #168-Ubuntu SMP Wed Jan 16 21:00:45 UTC 2019

（3）GPU型号：两台NVIDIA GeForce RTX 2080Ti（CUDA10.1）

（4）开发语言：python3.8.8

（5）深度学习框架：pytorch1.7.1 py3.8\_cuda10.1.243\_cudnn7.6.3\_0

（6）具体深度学习库请参见requirements\_all.txt

三、实验步骤+观察到的问题、思考及解决方案

（一）基础yoloV5部分

1、进行数据清洗：

发现图片数据集中存在以下几类图片

（1）完全没有标识符的图片（此类图片在所有实验中均被删除）

569\_cam-image\_array\_.jpg

648\_cam-image\_array\_.jpg

1552\_cam-image\_array\_.jpg

（2）模糊的图片（此类图片被保留，因为考虑到将来实战时我们必将面对这些图片）

（3）清晰的图片（完全保留）

综上，我们一共删除了3张图片

2、将数据分为了两个域：（赛道和毛毡地）

（1）240\_cam~7930\_cam：风格类似如下图片（1855-3=1852张图片）



（2）12960\_cam~16112\_cam：风格类似如下图片（905张图片）



3、rect参数（rectangular training）：

（1）如果加上，输入图片会仍然保持640\*480（w\*h）

优点：速度快

缺点：对于处在方框边缘的图标会稍有不利（因为Anchor不容易在边缘框物体）

（2）如果不加，输入图片会被更改为640\*640（w\*h）,我们考虑了两种方案：双线性插值直接拉伸、直接进行填充灰色

最后我们采取了后者，因为前者会影响空间关系（本身标注的anchor倾向于正方形，拉伸后ground truth就变成了长方形，不利于回归），后者对边缘处的图标会更友好

（3）总结：

最终我们采用了不加rect的方式，在几乎不影响速度的前提下，map得到了较大提升（后面实验进行了证明），之后实验基本都采用该方式

（二）创新（GAN的finetune部分）

我们发现数据集存在两个域，那设计一个跨域的模块就是很有必要的

1、该交通标志数据集的局限性

很容易看出，这个交通标志数据集过于简单，我们进行了简单实验，[map@0.5](mailto:map@0.5)都能达到 99.7，此时可能已经达到了瓶颈（标注可能不够细致导致的），我们不能再在这个数据集上验证跨域模块的有效性）

2、动用cityscape和foggy cityscape数据集

众所周知，这是两个很好的跨域验证集，我们也是基于这两个数据集做了大量实验，最终确定了我们跨域模块的雏形（两个8倍下采样的discriminator）

3、我们在上面两个数据集进行了大量实验，尝试了对yoloV5的backbone（尚未经过detect）的三层输出（80\*80，40\*40，20\*20）进行了各种形式的监督器实验，包括：

（1）对三层同时监督、对前两层（80\*80，40\*40）进行监督、对第一层（80\*80）进行监督

（2）监督器尝试了32倍下采样、8倍下采样、4倍下采样以及不进行下采样，以及32+不进行下采样的联合

（3）采用了BCE loss和LS loss两种方式  
（4）对三层的监督loss采用不同权重

最终我们得到了最好的配置方式：

（1）只对前两层进行监督

（2）采用8倍下采样

（3）采用LS loss

（4）前两层的监督loss权重均取0.1

以上实验都可以在train\_UDA\_single\_low.py中看到

4、跨域模型简述：

**（代码见：train\_UDA\_single\_low.py\domain\_ad.py\discriminator.py）**

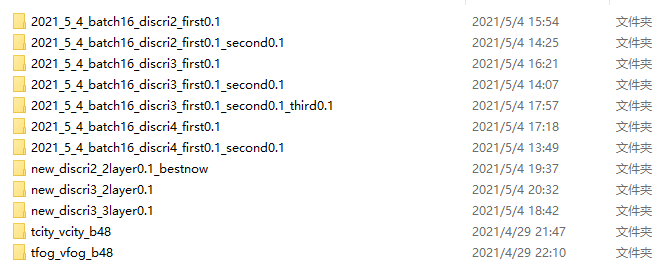
（1）加载下面实验4中训练的最好的模型（基于yoloV5x），在yoloV5x的最后输出的三张特征图（还没有经过detect模块）的前两层（80\*80和40\*40）加入两个discriminator（进行了多次实验，发现8倍下采样的监督器，并只对前两层进行监督，效果最好，无需加入空洞卷积，因为这是检测的特征图，很小）

（2）进行对抗训练（具体可见代码train\_UDA\_single\_low.py）

第一步，固定住两个监督器的权重，载入1域（图片+label）和2域（只有图片），在训练yoloV5的同时，让yoloV5的backbone可以欺骗监督器，使其相信第二个域的图片在特征上类似于第一个域

第二步，固定住yoloV5的backbone,训练监督器，让两个监督器可以区分两个域的特征

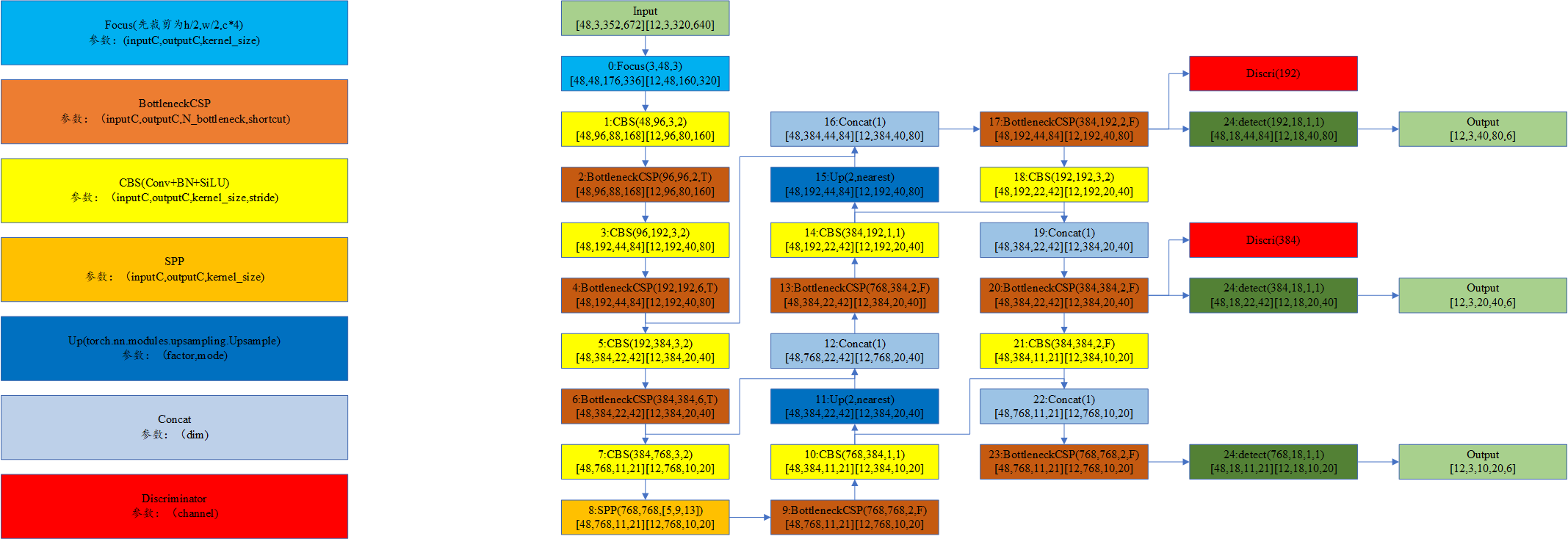
第三步，如此循环训练多个iter，yoloV5的backbone就会渐渐忽略两个域之间的domain gap



这些是在cityscape上的部分训练记录，但是太大无法上传到github

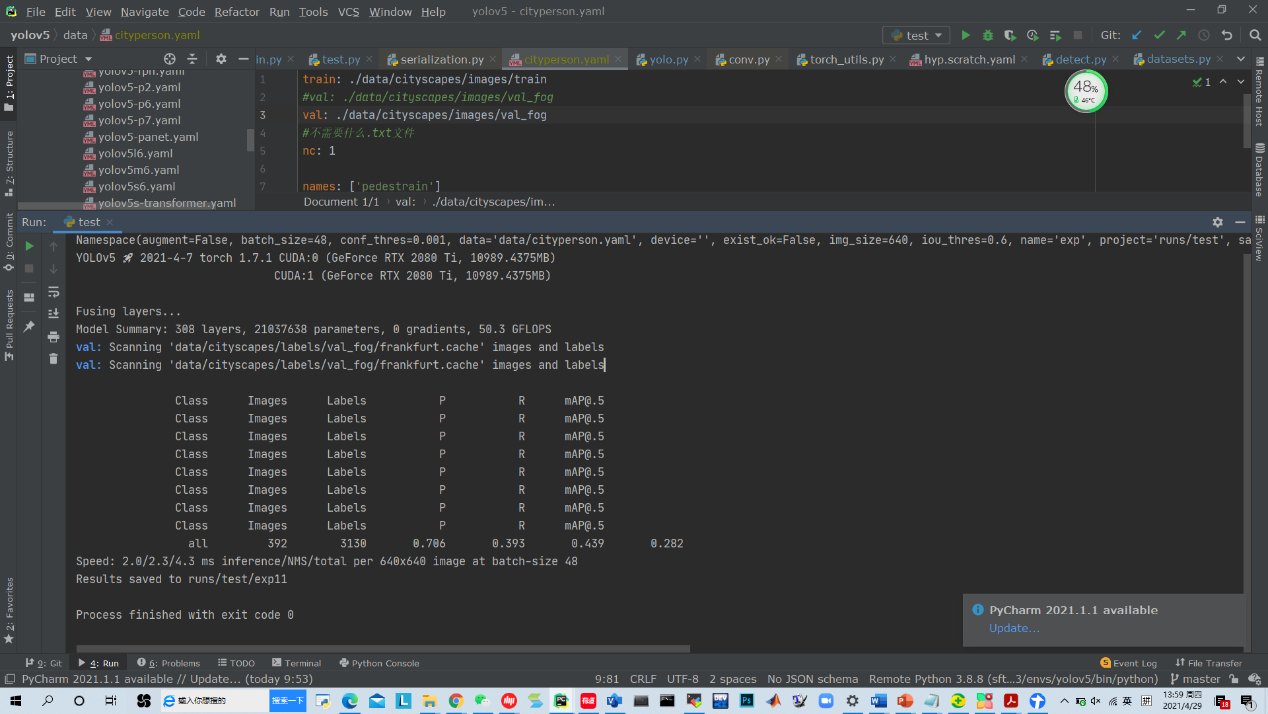
（3）参考图：

**此处使用的是yoloV5m的图，yoloV5x的效果同理**

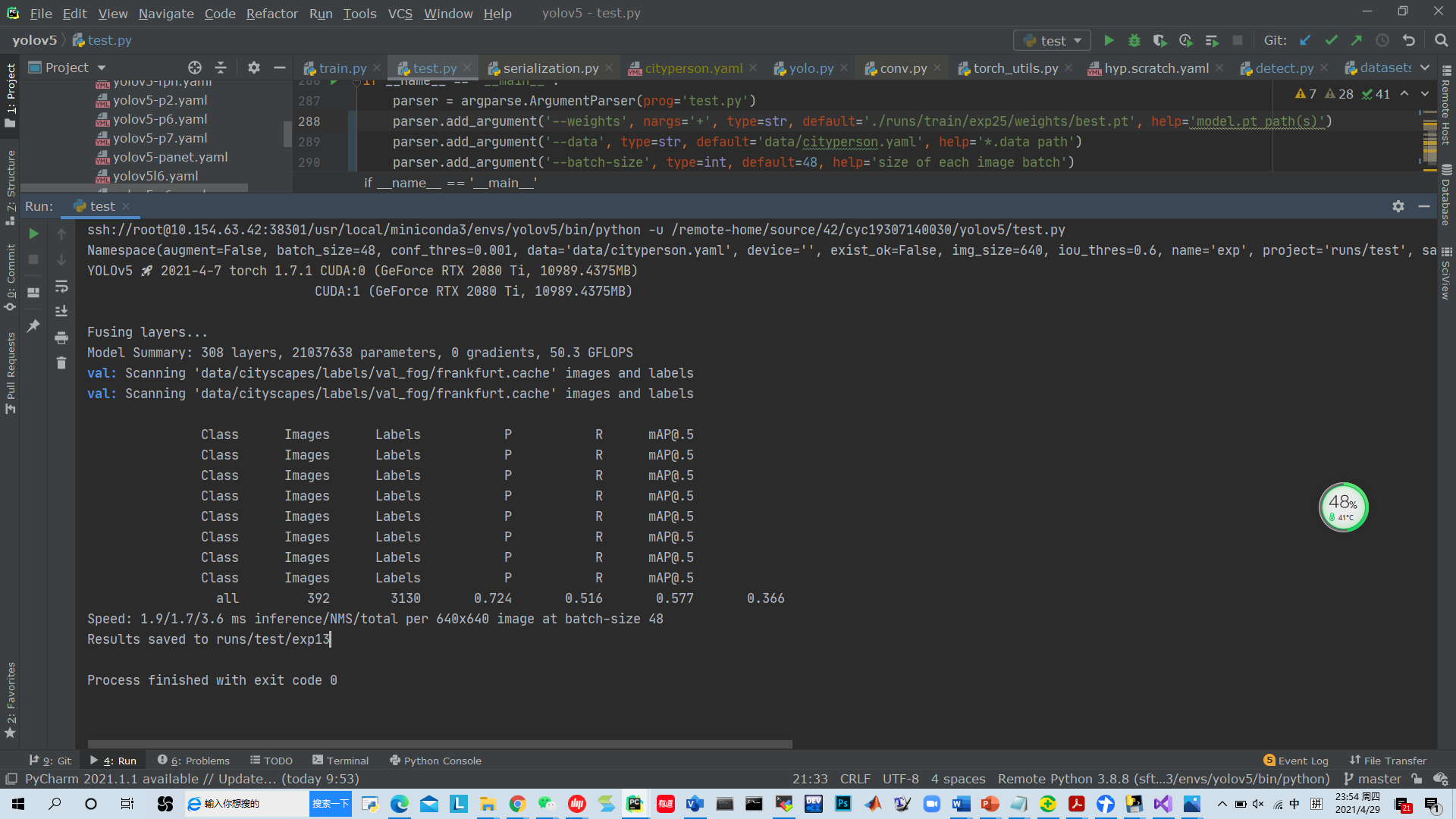


5、在cityscape和foggy cityscape（beta=0.02）上的实验结果（对行人进行检测）

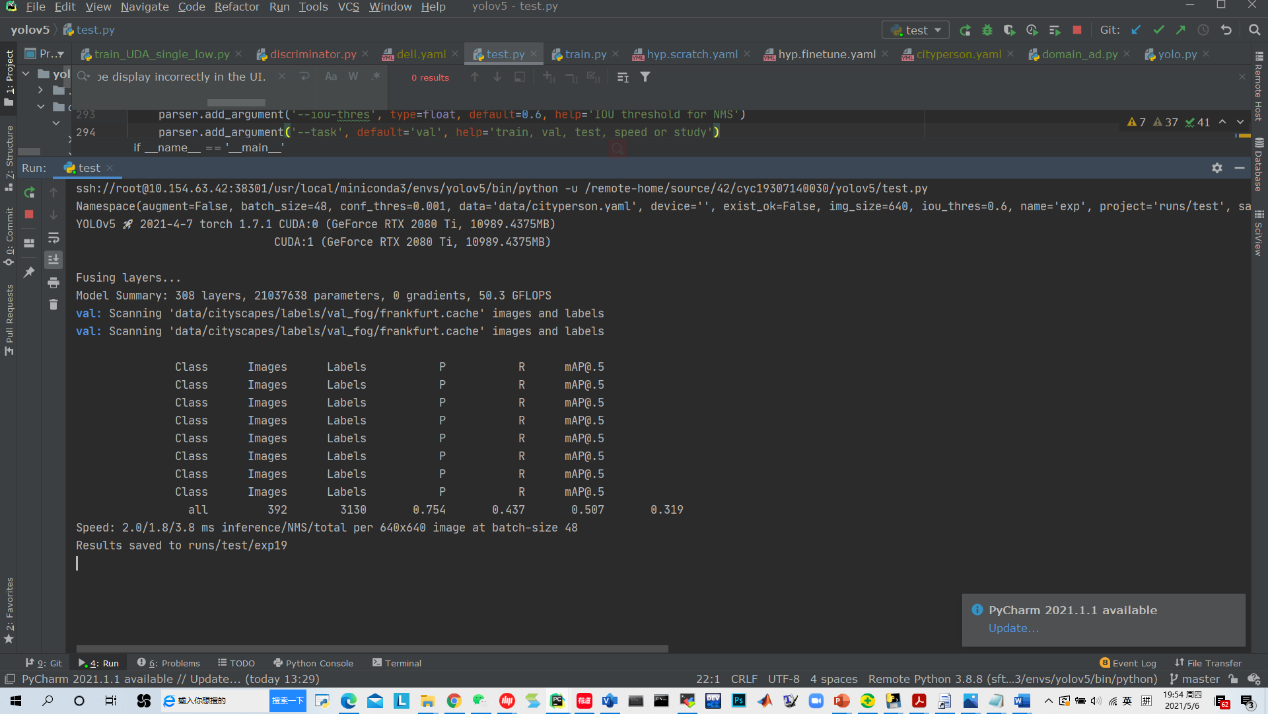
（1）不加跨域的结果：（train——city，test——foggy）



（2）极限：（train——foggy，test——foggy）



（3）加入该跨域模块的结果：（train——city，test——foggy）



显然已经达到了很好的跨域结果，之后可以直接转移到我们现有的交通标志检测的数据集上进行使用

四、部分实验

注意：所有的数据集都可以在github上的./data/dell/里查看

（一）基础yoloV5的实验

1、只在第一个域上进行实验（就是仅使用在赛道上的图片，不使用后半部分在毛毡地上的图片）

**·实验记录：exp\_simple\_norect（./exps下，非常全）**

·模型：yoloV5m

·输入图片大小：640\*640（没有rect）,图片剩余部分使用灰色填充

灰色填充的好处：

·训练集、验证集参数（无test集）：

train\_simple/val\_simple（只处理了第一个域共1852张图片，将其按85：15分成train/val集，随机分配shuffle）

分别：

train\_simple:1575张

val\_simple:277张

·训练出模型的时长：100epochs，0.595hours

·推理出100张图片时长：510ms（如果不算后处理NMS，280ms）

·准确率：

Class Images Labels P R mAP@.5 map@0.5~0.95

all 277 277 0.998 1 0.996 0.858

cancel 277 168 1 1 0.997 0.86

limit 277 109 0.996 1 0.996 0.857

2、只在第一个域上进行实验（就是仅使用在赛道上的图片）

**·实验记录：exp\_simple\_rect（./exps下，非常全）**

·模型：yoloV5m

·输入图片大小：640\*480（进行rect）（输入的tensor是[batch,3,480,640]）

·训练集、验证集参数（无test集）：

train\_simple/val\_simple（只处理了第一个域共1852张图片，将其按85：15分成train/val集，随机分配shuffle）

分别：

train\_simple:1575张

val\_simple:277张

·训练出模型的时长：100epochs，0.550hours

·推理出100张图片时长：460ms（如果不算后处理NMS，280ms）

·准确率：

Class Images Labels P R mAP@.5 map@0.5~0.95

all 277 277 0.981 1 0.996 0.845cancel 277 168 0.971 1 0.997 0.834limit 277 109 0.991 1 0.996 0.855

3、在两个域上进行实验（均分到train/val集中）

**·实验记录：exp\_alldata\_simple\_norect（./exps下，非常全）**

·模型：yoloV5m

·输入图片大小：640\*640（没有rect）

·训练集、验证集参数（无test集）：

train\_alldata\_simple/val\_alldata\_simple(所有图片都被使用，两个域都按照85：15分成train/val集，随机分配shuffle）

分别：

train\_alldata\_simple:1575+770=2345张

val\_alldata\_simple:277+135=412张

·训练出模型的时长：100epochs，###hours #此处训练的直接log文件丢失，无法记录详细时间（大概在0.8h左右）

·推理出100张图片时长：500ms（如果不算后处理NMS，270ms）

·准确率：

Class Images Labels P R mAP@.5 map@0.5~0.95

all 412 412 0.997 0.998 0.997 0.899cancel 412 226 1 0.996 0.997 0.891limit 412 186 0.995 1 0.997 0.908

4、在两个域上进行实验（均分到train/val集中）

**·实验记录：exp\_yolov5x\_alldata\_simple\_norect（./exps下，非常全）**

·模型：**yoloV5x**

·输入图片大小：640\*640（没有rect）

·训练集、验证集参数（无test集）：

train\_alldata\_simple/val\_alldata\_simple(所有图片都被使用，两个域都按照85：15分成train/val集，随机分配shuffle）

分别：

train\_alldata\_simple:1575+770=2345张

val\_alldata\_simple:277+135=412张

·训练出模型的时长：100epochs，2.702hours

·推理出100张图片时长：1070ms（如果不算后处理NMS，840ms）

·准确率：

Class Images Labels P R mAP@.5 map@0.5~0.95

all 412 412 1 1 0.997 0.902cancel 412 226 0.999 1 0.997 0.893limit 412 186 1 1 0.997 0.912

（二）创新（全部是原创）：GAN的finetune

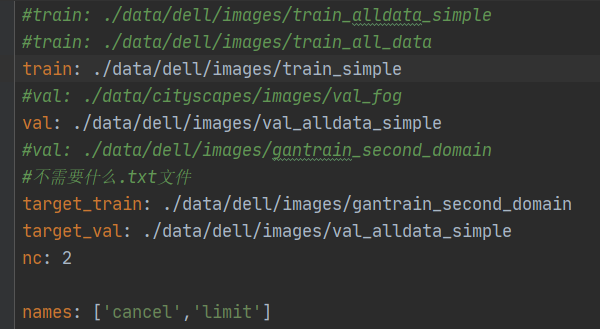
5、在前一个实验的基础上，进行对抗的finetune（因为考虑到第二个域的图片要少一些）

**·实验记录：exp\_gan（./exps下，非常全）**

·模型：**yoloV5\_x**

·输入图片大小：640\*640（没有rect）

·训练集、验证集参数（无test集）：



train\_simple:只包含第一个域的图片

gantrain\_second\_domain(只使用第二个域的图片，在跨域的finetune中被使用，但是没有使用标签，相当于一个无监督的跨域）

·训练出模型的时长：2epochs，10minutese

·推理出100张图片时长：1000ms（如果不算后处理NMS，840ms）

·准确率：

Class Images Labels P R mAP@.5 map@0.5~0.95

all 412 412 1 1 0.997 0.902cancel 412 226 1 1 0.997 0.899limit 412 186 1 1 0.997 0.905

**可以发现cancel的准确率得到了有效提升（提升不大是因为此时已经达到瓶颈，最大的限制应该是标注的问题）**

·模型大小：（实验5）

.pth文件：175.1MB

Model Summary: 607 layers, 87251103 parameters, 87251103 gradients, 217.3 GFLOPS