dell AI交通标志识别比赛

注意：

·代码已经在<https://github.com/caiyancheng/dell_yoloV5>开源，此处实验记录非常全，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，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

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

3、模型：

yoloV5m\yoloV5x

总共进行了以下几组实验

（1）只在第一个域上进行实验（就是仅使用在赛道上的图片）

**·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

·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）

·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

·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

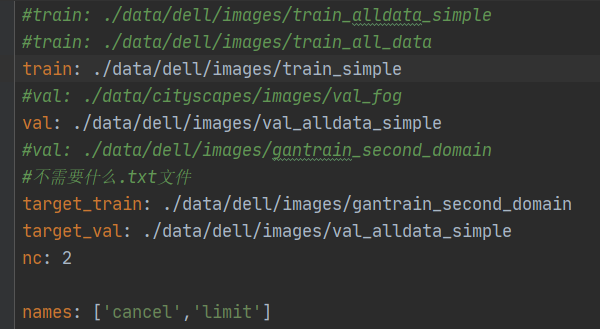
（5）在前一个实验的基础上，进行对抗的finetune（因为考虑到第二个域的图片要少一些）

**·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fintune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，10minutes

·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）**

跨域模型简述：

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

（1）加载（4）中训练的最好的模型，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）

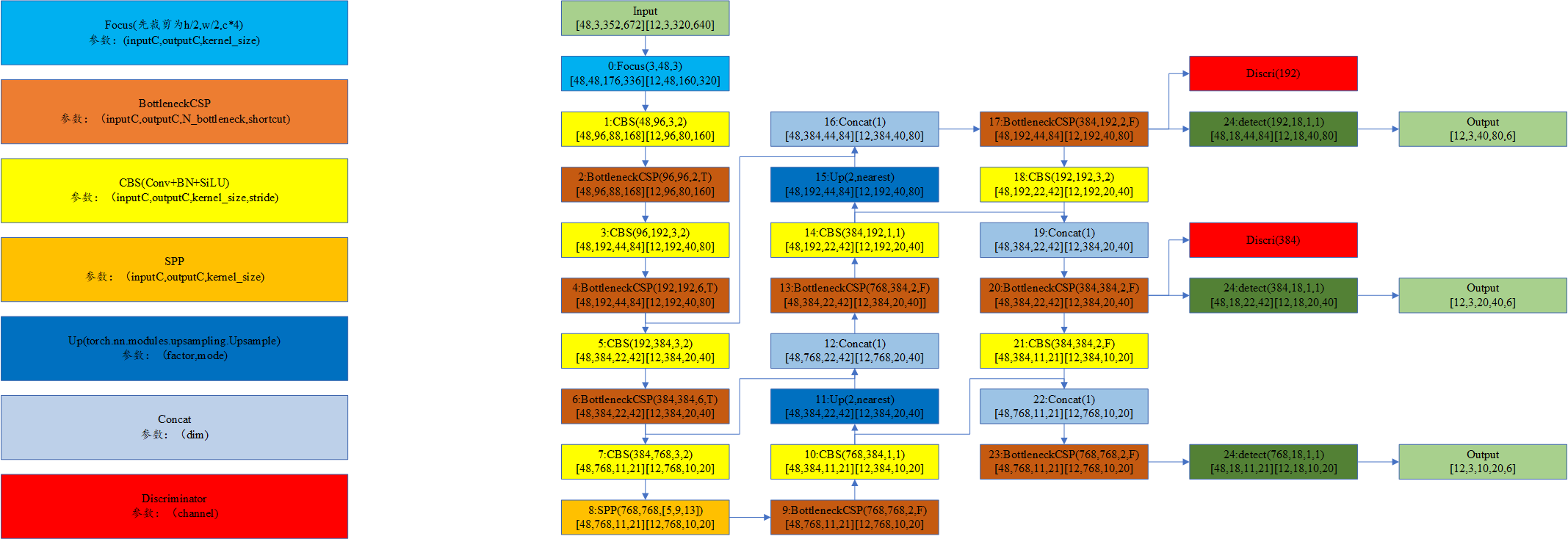
第一步，固定住两个监督器，让yoloV5的backbone可以欺骗监督器，使其相信第二个域的图片在特征上类似于第一个域

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

第三步，如此循环，yoloV5的backbone就会渐渐忽略domain gap

（3）参加图：

**此处使用的是yoloV5m的图，yoloV5x的效果同理**



问题：

1、不清楚的图片？

2、inference?

3、