模型训练

1. 安装环境依赖(https://github.com/AlexeyAB/darknet#requirements-for-windows-linux-and-macos)
2. 训练工具编译

git clone https://github.com/AlexeyAB/darknetcd darknet

mkdir build\_releasecd build\_release

cmake ..

cmake --build . --target install --parallel 8

1. 准备数据集，训练集图片放在train文件夹内，验证集放在val文件夹内。
2. 数据标注

git clone <https://hub.fastgit.org/AlexeyAB/Yolo_mark.git>

cmake .

make

./linux\_mark.sh

使用方法详见Yolo\_mark目录内的readme.md

1. 模型训练

除两个数据集外，启动训练还需要配置几个参数文件。

obj.data

obj.name

train.txt

以上三个文件,会在数据标注时自动生成在Yolo\_mark/x64/Release/data目录下，obj.name文件包含所有目标的类别名，train.txt包含所有训练图片路径，val.txt非必须，可以手动从train文件中分割出30%的图片用于验证。而obj.data文件申明了上述所有文件的路径和类别总数，如果使用自己的数据集，对应参数修改请在标注前完成修改。

yolo.cfg(拓扑)

Yolo.conv（预训练权重）

cfg和conv存在一定对应关系，考虑此处训练的模型最终需部署在openncc上，推荐使用（yolov4-tiny.cfg+[yolov4-tiny.conv.29](https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.conv.29)）或（yolov3-tiny.cfg+[yolov3-tiny.conv.11](https://drive.google.com/file/d/18v36esoXCh-PsOKwyP2GWrpYDptDY8Zf/view?usp=sharing)）的搭配，cfg文件可以直接在darknet/cfg目录下找到。

Cfg文件修改！！

如果目标类别数量不等于80，则必须修改cfg文件。

搜索cfg文件中所有yolo层的位置，若总共有3 类目标，则将[yolo]层classes参数定义为3，再将[yolo]上一层[convolutional]层的filters定义为24.计算方式为filters=（classes+5）\*3。

对yolov4-tiny.cfg来说有两个yolo层，所以一共需要修改4个参数。

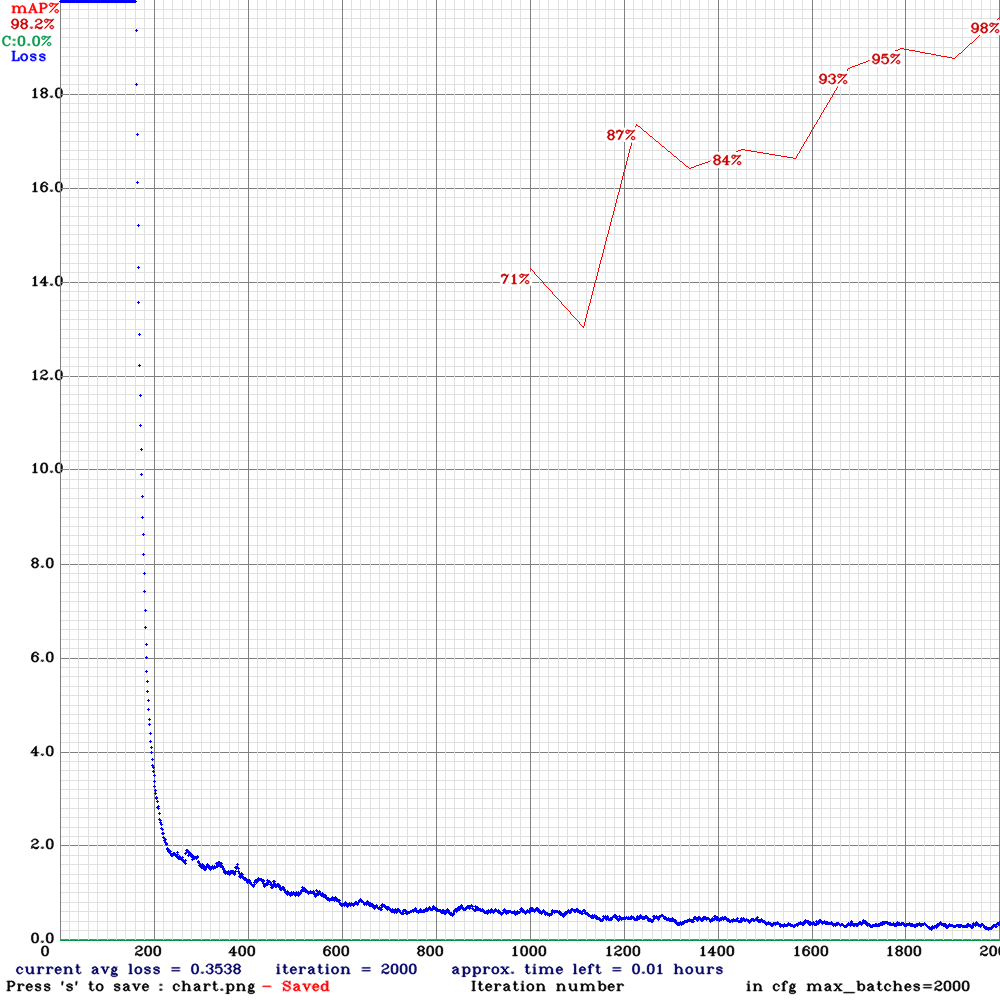
启动训练

第二步编译成功后，会在darknet目录下生成./darknet工具。

输入命令： ./darknet detector train ./obj.data ./yolov4-tiny.cfg ./yolov4-tiny.conv.29 -map

垃圾显卡（低于1080Ti）可能出现内存不够的报错，此时需要在cfg的第一层[net]中将batch参数改到8以下（8，4，2，1）。

若训练进行顺利，可看到如下图的训练日志图表。

训练结束后，可看到一系列.weights文件。这里还是建议制作数据集时设置一个验证集，这样可以直接锁定验证集中map最高的权重yolov4-tiny\_best.weights作为后续使用。

1. 模型转换

Darknet to tensorflow

git clone <https://github.com/RenLuXi/tensorflow-yolov4-tiny.git>

cd tensorflow-yolov4-tiny

python convert\_weights\_pb.py --class\_names obj.names --weights\_file yolov4-tiny\_best.weights --tiny

这里需要用到第五步中obj.names和yolov4-tiny\_best.weights.

Tensorflow to IR(openvino推理格式)

修改json配置文件

打开tensorflow-yolov4-tiny目录下的yolo\_v4\_tiny.json，将其中的classes值修改为你自己的类别数，openvino进行tensorflow转换需要用到这个文件。

然后替换json配置文件

cp ./yolo\_v4\_tiny.json /opt/intel/openvino/deployment\_tools/model\_optimizer/extensions/front/tf

进入openvino模型转换工具目录

cd /opt/intel/openvino/deployment\_tools/model\_optimizer

转换命令

python mo.py --input\_model yolov4-tiny.pb --transformations\_config ./extensions/front/tf/yolo\_v4\_tiny.json --batch 1 --data\_type FP32 --reverse\_input\_channels

IR to blob

老办法，先初始化openvino环境，然后把上一步生成的xml和bin文件丢过去转换

source /opt/intel/openvino\_2020.3.194/bin/setupvars.sh

cd /opt/intel/openvino\_2020.3.194/deployment\_tools/inference\_engine/lib/intel64

cp /opt/intel/openvino/deployment\_tools/model\_optimizer/yolov4-tiny.xml ./

cp /opt/intel/openvino/deployment\_tools/model\_optimizer/yolov4-tiny.bin ./

/opt/intel/openvino\_2020.3.194/deployment\_tools/inference\_engine/lib/intel64/myriad\_compile -m yolov4-tiny.xml -o yolov4-tiny.blob -VPU\_MYRIAD\_PLATFORM VPU\_MYRIAD\_2480 -VPU\_NUMBER\_OF\_SHAVES 6 -VPU\_NUMBER\_OF\_CMX\_SLICES 6

1. 使用模型

把xml，bin，blob三个文件放入OpenNCC-yolo中使用，具体方法可参考OpenNCC yolo用户使用手册。