训练步骤：

1. 准备标签

数据集目录结构如下：

dataset

train\_data

Annotations

\*.xml

JPEGImages

\*.jpg

labels

\*.txt

validation\_data

Annotations

\*.xml

JPEGImages

\*.jpg

labels

\*.txt

1.1 图片标注：

使用labelimg标注生成Annotations目录下xml文件

1.2 生成网络输入的标签文件：

编写脚本face\_label.py通过Annotations，JPEGImages生成yolo网络结构需要的labels目录下的txt文件。

执行命令：python face\_label.py

txt文件内容结构如下：  
<object-class> <x> <y> <width> <height>

<object-class> 类别索引：0 – (classes -1)

<x> <y> 中心点坐标

<width> <height> 宽，高

1. 根据自己的数据集生成anchors

yolo-tiny: 6组, yolo: 9组

2.1 使用python脚本：

gen\_anchors.py

2.2 darknet自带的命令：

./darknet detector calc\_anchors cfg/voc.data -num\_of\_clusters 9 -width 416 –height 416

1. 配置文件

3.1 cfg目录:

3.1.1 yolov3-face.cfg

batch=64 #一批训练样本的数量，每batch个样本更新一次参数)

subdivisions=4 #batch/subdivisions作为一次性送入训练器的样本数量)

max\_batches=50000 #训练批次总数：建议至少classes\*2000

steps=40000, 45000

width=416 (32的倍数)

height=416

[convolutional]

filters=18

[yolo]

anchors= xx,xx, xx,xx, … #根据自己的数据集生成的anchors

classes=1

num=6

random=1 #设置为0，表示关闭多尺度训练

注：

每次训练传入网络模型图片个数=batch/subdivisions, 然后把结果合起来也就是一个batch更新模型权重一次；

filters的计算公式为：（classes + 5）\* 3

3.1.2 face.data:

class=1 (类别个数)

train=/home/xx/train.txt (训练标签文件路径)

valid=/home/xx/Validation.txt (验证标签文件路径)

names=data/face.names (分类类别文件)

backup=backup (权重文件存放目录)

3.2 data目录:

3.2.1 face.names (列出了分类的所有类别)

face

1. 多尺度训练

Darknet框架特殊的数据增强方式：随机化输入网络的图片尺寸，提升网络的泛化能力。

1. 训练
   1. 修改Makefile文件

启用GPU：

GPU=1

CUDNN=1

NVCC=/usr/local/cuda-11.3/bin/nvcc

COMMON+= -DGPU /usr/local/cuda-11.3/include/

LDFLAGS+= -L/usr/local/cuda-11.3/lib64

编译命令：

make clean

make

编译成功，会生成两个文件：

darknet

libdarknet.so

5.2 训练命令：

./darknet detector train cfg/face.data cfg/yolov3-face.cfg weights/darknet53.conv.74 >> log.txt

* 1. 命令参数

-clear 从头训练

-thresh

-gpus 0,1,2,3

-map 在训练过程中查看mAP和Loss曲线

1. 验证

./darknet detector test cfg/face.data cfg/yolov3-face.cfg weights/yolov3-tiny-face\_final.weights

1. 评估模型

7.1 计算map

./darknet detector map cfg/face.data cfg/yolov3-face.cfg weights/yolov3-tiny-face\_final.weights

7.2 标准版的yolov3, 第一版本数据集，416 x 416

yolov3-tiny:

34.7M

94.27%

yolov3:

246.3M

94.16%

7.3 mobilenetv2-yolov3+-lite:

优化点：

1. 去掉SPP
2. 使用mobilenetv2作为backbone: groups分组卷积
3. 去掉AB版的特殊参数
4. 去掉Mish激活函数，使用relu激活函数

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 模型名称 | 图像尺寸 | 模型大小 | 训练批次 | Yolo Head | backbone | 多尺度训练 | map | 优化点 |
| Yolov3 | 416x416 | 246.3M | 50200 | 3 | Darknet53 | 128~416 | 94.16% | 原始打标数据集 |
| Yolov3-tiny | 416x416 | 34.7M | 50200 | 2 | Darknet53 | 128~416 | 94.27% | 原始打标数据集 |
| Yolov3-tiny-迭代版 | 416x416 | 34.7M | 50200 | 2 | Darknet53 | 128~416 | 94.04% | 1. 数据集: 去掉远距离小框 (第一次修正); |
| Mobilenetv2-yolov3+-lite | 320x320 | 6.8M | 50200 | 2 | Mobilenetv2 | 128~416 | 93.89% | 1. 网络结构优化: 去掉spp, mobilenetv2作为backbone, relu激活函数;  2. 数据集: 去掉远距离小框 (第一次修正); |
| Mobilenetv2-yolov3+-lite-迭代版1 | 416x416 | 6.8M | 50200 | 2 | Mobilenetv2 | 288~480 | 94.05% | 1. 数据集: 去掉远距离小框 (第一次修正);  2. 生成修正后数据集对应的anchors  3. 多尺度训练图像随机尺寸为288～480 |
| Mobilenetv2-yolov3+-lite-迭代版2 | 416x416 | 6.8M | 50200 | 2 | Mobilenetv2 | 288~480 | 96.23% | 1. 数据集: 去掉远距离小框和修正不规范标签(第二次修正);  2. 生成修正后数据集对应的anchors  3. 多尺度训练图像随机尺寸为288～480 |

1. 工具

9.1 python web服务器

python -m http.server 8081

9.2 模型可视化工具

在线可视化模型工具:

<https://netron.app/>