**前言**

data目录当中，test目录下是测试集，train目录下是训练集。

data目录中还有yolo模型所需的classes.txt、yolo\_anchors.txt和train\_data.txt。

在pre\_train目录当中，有yolo的预训练模型yolov3.weight，以及用convert.py转换的yolo\_weights.h5。

在model\_data目录中，存储训练结束的模型yolo1.h5、kmeans.py生成的yolo1\_anchors.txt、包含holothurian echinus scallop starfish waterweeds5类的voc1\_class。(yolo\_anchors、voc\_class和yolo.h5是第一次训练的含fish/human\_face的模型）

**数据预处理**

data\_process.py中，对所有图像进行统计，对每一个候选框，进行细致的处理，代码中给出了详细的注释。候选框处理时删除那些面积小于120的过小候选框。之后再靠kmeans.py来生成yolo所需的anchors，最后得到data目录中的train\_data.txt和yolo\_anchors.txt。

**数据增强**

数据增强代码在yolo3目录的utils.py中。

**模型训练**

在train.py中，输入尺寸为416×416，训练集和测试集按照9:1划分。训练过程分为两个阶段来训练，第一阶段冻结预训练所有层，采用RAdam(最小值设为1e-5)， warm\_up策略，batch\_size设为32，训练100轮，同时做Tensorboard记录。在第二阶段打开全部网络层来训练，采用RAdam(最小值设为1e-6)，warm\_up策略，swa算法，cosine-annealing学习率策略（范围1e-2到1e-6），batch\_size设为8（主要原因是显存限制），训练200轮，同时做Tensorboard记录，通过ModelCheckpoint策略对每一轮按照val\_loss来决定是否保存模型，最终选用val\_loss最小的模型来做预测。将最好的模型更名为yolo1.h5。

**预测**

在yolo.py、yolo\_matt.py、yolo\_video.py中，代码中的注释很详细，这里不再阐述。

在参数上，score阈值设为0.1，iou阈值（包括模型阈值和WBF阈值）设为0.3能得到较好的识别结果。yolov3部分的代码是基于qqwweee/keras-yolo3进行更改的。

**运行**

运行yolo\_video.py，在程序运行窗口根据提示输入图片路径，按回车进行对图像的识别。结果如下图所示：

