环境：tf1.13

>=Python3.5（编程注意点：python编辑均采用空格，不可出现tab。）

>=opencv3.2

文件部署：

Layers

Faster-ssd.Py

Config

data\_prepare

Graphview

Root

saved\_networks

loss.Py

main.py

hyperparameter.py

Specific\_net

Utils

Config.py

Root为根文件夹目录。

layers存放一些封装好的层，比如faster的proposal层，以及一些具有前沿的层（例如capsule）。Specific\_net用于一些常用的网络整合，例如vgg，其中包围我们自身积累的一些裁剪网络。

Faster-ssd.py主要用于检测任务的训练与测试。

Utils内包含一些常用的函数，例如大量的iou计算（包括GPU与cpu）。

Config内部包含算法策略细节的参数调整，不包括学习率等统一的超参数（学习率不属于某一算法独有的，检测与识别或者分割都有）。其内容有config.py读取。

data\_prepare是数据层，需要包含乒乓操作机制，高效的数据读取。目前数据流采用faster的数据流，以读取txt目录为主，实时加载jpg图像。

loss.Py计算loss的函数，内部不包含loss的形式，例如smoothL1loss，margin等，这个属于特定的层，在layers内，loss.py加载layers内的loss计算层，但具体的bp规则是在loss.py内实现，例如采用sgd还是adam，是否cut梯度等。

hyperparameter.py是设置学习率等统一超参数。

main.py运行程序作用，可以输入一些与任务参数无关的内容（例如是运行检测任务还是分类任务）。

Graphview为保存运行数据，tf画图使用。saved\_networks用于保存网络参数。

第一期实现内容：检测，识别二合一

具体：检测：faster，ssd

识别: 各种loss

第二期实现内容：各种前沿层，加入量化机制，各种specific net加入，一键tf转TRT工具

需要用到的caffe的faster层：

frcnn\_anchor\_target\_layer：用于采样anchor的正负样本，用于rpn的label

frcnn\_proposal\_layer：rpn输出

frcnn\_proposal\_target\_layer：对rpn输出结果进行采样，用于生成rcnn的正负样本

roi\_pooling\_layer：roipooling层

roi\_mask\_pooling\_layer：roialign层

frcnn\_roi\_data\_layer：数据层

smooth\_L1\_loss\_layer

**第一期：**

需要编写的层：

Rpn\_anchor\_target\_layer.py:对应frcnn\_anchor\_target\_layer，多个策略需要加入

Rpn\_proposal\_layer.py:对应frcnn\_proposal\_layer，需要注意，NMS有很多变种，需要加入，比如softNMS，myNMS等，用户可以自行选择NMS实验，排序也有多种选择，根据耗时自行选择。

Rcnn\_proposal\_target\_layer.py:对应frcnn\_proposal\_target\_layer。

Roi\_pooling\_layer.py:对应roi\_pooling\_layer与roi\_mask\_pooling\_layer，两者结合一起，怎么用可以用户选择。

[Detection](file:///C:\Users\Administrator\AppData\Local\youdao\dict\Application\7.3.0.0807\resultui\dict\?keyword=detection)\_data\_layer.py:对应frcnn\_roi\_data\_layer，需要有乒乓操作，且有各种扰动机制，除了frcnn\_roi\_data\_layer出现的hsv，gamma，lab，scale，还需要加入特征点数据层的各种扰动，还需要注意label的选择，rpn与stage2的label是不一样的，但可以有选择机制。

Loss层的loss包含smoothL1loss，softmax，MSE，各种margin，SVN，每一个均为一个py文件。带weigh与不带weigh都需要（weigh在smoothL1loss分内外weigh），可以用户选择，详细参考smooth\_L1\_loss\_layer。

所有的layers内的层均采用class的类型方式。

**第二期：**

Specific\_net中加入VGG net，各种inception，resnet等，加入我们自身的一些前置网络。

加入capsule层，doublerpn层等

加入量化机制

加入转TRT的便捷工具

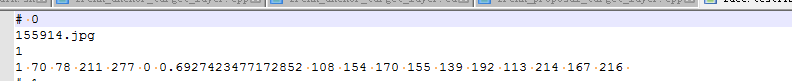
加入分布式

工程采用github整合。

数据流形式：

Data层输入：图片（BGR）+txt中的label

Label的输入顺序为：



# 序号

图片名

Label的个数

Label的类别（bg为0） bbx diff 质量判断 五点坐标

输出图片数据：（batch，w，h，n）

输出label：字典

字典的key为batch\_index

字典的内容：{label，图片名}

列表内存放了该batch\_index下的label

格式如下：

{‘0’:

{ ‘img’:’155914.jpg’,

’label’:[{‘cls’:1,

’bbx’:[70,78,211,277],

’diff’:0,

’jq’:0.692..,

’landmark’:[108,…]

},

{…},

{…}

…]

},

‘1’:{…},

…

}