# [caffe ssd 在windows上的编译过程](http://blog.csdn.net/buaalei/article/details/54668507)

1.<https://github.com/conner99/caffe>,选择ssd-microsoft 分支

2. 复制配置文件CommonSettings.props.example为CommonSettings.props,按需修改对应的地方，参照caffe-windows的配置

错误解决:

错误1:GLOG\_NO\_ABBREVIATED\_SEVERITIES

解决:在工程加上预编译宏GLOG\_NO\_ABBREVIATED\_SEVERITIES  
C/C++   -->  预处理器   -->   预处理器定义   -->  加上GLOG\_NO\_ABBREVIATED\_SEVERITIES

错误2:opencv 问题

解决:在nuget里把opencv卸载,然后重新搜索opencv2.4.10进行安装

错误3:找不到".\caffe\3rdparty\hungarian.h"文件

解决:在路径.\caffe-master\include\caffe\3rdparty\下添加hungarian.h文件。编译项目，报错：找不到".\src\caffe\3rdparty\hungarian.cpp"文件，在路径.\caffe-master\src\caffe\3rdparty\下添加hungarian.cpp文件

在这里找这个文件,链接：http://pan.baidu.com/s/1mhYuf7y 密码：3jp2

错误4:expected an identifier in caffe.pb.h

解决:修改bbox\_util.cu，注释掉所有带thrust的语句

修改detection\_output\_layer.cu和detection\_output\_layer.cpp文件，注释掉所有regex和rv的语句  
 //boost::regex exp("\"(null|true|false|-?[0-9]+(\\.[0-9]+)?)\"");  
//std::string rv = boost::regex\_replace(ss.str(), exp, "$1");    
//outfile << rv.substr(rv.find("["), rv.rfind("]") - rv.find("["))    
//    << std::endl << "]" << std::endl;   
修改detection\_output\_layer.hpp，注释#include“boost/regex.hpp”  
//#include <boost/regex.hpp>

# 训练

1.准备数据

需要根据PASCAL VOC格式 准备对应的数据，推荐一个好用的[贴标签软件](https://github.com/tzutalin/labelImg)

拿VOC2012为例，VOC2012目录下需要Annotations、ImageSets、JPEGImages文件夹，Annotations存储对应的标签xml信息，JPEGImages存储图片数据，ImageSets 主要使用Main文件夹下的数据，Main文件夹保存了对应的训练、验证数据集的txt文本

然后使用caffe-ssd-microsoft\data\VOC0712目录下create\_data.bat来生成所需要的lmdb文件(create\_data.bat根据实际情况修改内容)

caffe-ssd-microsoft\data\VOC0712目录下get\_image\_size.bat用于生成test.txt中对应图片的大小映射，生成的结果写入test\_name\_size.txt

对于后面的训练需要准备以下文件:

test.txt  用于测试的文件集定义

labelmap\_voc.prototxt   标签定义文件

test\_name\_size.txt 测试图片大小定义文件(可通过get\_image\_size.bat生成)

trainval.txt 训练验证集文件定义(数据排列为乱序，可根据实际情况设定)

train.prototxt  训练网络定义文件  
test.prototxt    测试网络定义文件  
deploy.prototxt    部署定义文件  
solver.prototxt    训练配置文件

2.修改训练文件

caffe-ssd-microsoft\examples\ssd下的ssd\_pascal.bat是用来训练数据用的程序(它是调用ssd\_pascal.py运行的，ssd\_pascal.py用来配置整个的训练环境)，因为windows下的对 路径和linux的不同，所以对应的windows下需要找到ssd\_pascal.py对应的内容进行修改：

train\_data = "{}/data/VOC0712/trainval\_lmdb".format(caffe\_root)  
test\_data = "{}/data/VOC0712/test\_lmdb".format(caffe\_root)  
save\_dir = "{}/models/VGGNet/VOC0712/{}".format(caffe\_root,job\_name)  
snapshot\_dir = "{}/models/VGGNet/VOC0712/{}".format(caffe\_root,job\_name)  
job\_dir = "{}/jobs/VGGNet/VOC0712/{}".format(caffe\_root,job\_name)  
output\_result\_dir = "{}/data/VOC0712/results/{}/Main".format(caffe\_root,job\_name)  
name\_size\_file = "{}/data/VOC0712/test\_name\_size.txt".format(caffe\_root)  
pretrain\_model = "{}/models/VGGNet/VGG\_ILSVRC\_16\_layers\_fc\_reduced.caffemodel".format(caffe\_root)  
label\_map\_file = "{}/data/VOC0712/labelmap\_voc.prototxt".format(caffe\_root)  
f.write('{}\Build\{}\Debug\caffe train ^\n'.format(caffe\_root,'x64'))

[SSD安装及训练自己的数据集](http://blog.csdn.net/zhang_shuai12/article/details/52346878)

最近一直在搞object detection玩，之前用的是faster-rcnn，准确率方面73.2%，效果还不错，但是识别速度有点欠缺，我用的GPU是GTX980ti, 识别速度大概是15fps.最近发现SSD(single shot multibox detector) 这篇论文效果和速度都不错，我自己实验了一下，速度确实比faster-rcnn快不少。下面分两部分来介绍。第一部分介绍SSD的安装，第二部分介绍如何基于SSD训练自己的数据集。

第一部分 SSD安装

系统：ubuntu 14.04   
语言：[**Python**](http://lib.csdn.net/base/11)   
ssd项目主页：<https://github.com/weiliu89/caffe/tree/ssd>   
首先，我们把项目代码clone下来, 然后编译：

git clone https://github.com/weiliu89/caffe.git

cd caffe

git checkout ssd

接下来，我们开始编译caffe,编译caffe非常容易，这里我们滤过，如若感兴趣，可参考我之前的博客：<http://blog.csdn.net/zhang_shuai12/article/details/52289825>，此篇讲的是caffe + cpu + ubuntu14.04的安装， GPU版安装需修改Makefile.config文件， 修改完成后：

make

make py

到这里我们就完成了SSD的安装，接下来我们讲一下如何训练自己的数据集。

第二部分 训练自己的数据集

首先我们不妨先跑一下项目的demo, 需要下载数据集，提前训练好的数据集等。   
下载预训练的模型，链接：<https://gist.github.com/weiliu89/2ed6e13bfd5b57cf81d6>，下载完成后保存在：

caffe/models/VGGNet/

下载VOC2007和VOC2012数据集， 放在/data目录下：

cd data

wget http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval\_11-May-2012.tar

wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval\_06-Nov-2007.tar

wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtest\_06-Nov-2007.tar

tar -xvf VOCtrainval\_11-May-2012.tar

tar -xvf VOCtrainval\_06-Nov-2007.tar

tar -xvf VOCtest\_06-Nov-2007.tar

创建lmdb格式的数据：

cd caffe

./data/VOC0712/create\_list.sh

# It will create lmdb files for trainval and test with encoded original image:

# - $HOME/data/VOCdevkit/VOC0712/lmdb/VOC0712\_trainval\_lmdb

# - $HOME/data/VOCdevkit/VOC0712/lmdb/VOC0712\_test\_lmdb

./data/VOC0712/create\_data.sh

训练和测试：

python examples/ssd/ssd\_pascal.py

论文中，作者已经预训练好模型，下载链接：[http://www.cs.unc.edu/%7Ewliu/projects/SSD/models\_VGGNet\_VOC0712\_SSD\_300x300.tar.gz](http://www.cs.unc.edu/~wliu/projects/SSD/models_VGGNet_VOC0712_SSD_300x300.tar.gz)，我们不必自己再去训练, 下载完成后放入指定的文件夹下。   
测试时，我们使用/example/ssd/目录里的ssd\_detect.ipynb，运行这个文件，需要安装ipython及ipython-notebook, 或者直接把里面的代码拷贝出来，写到一个新的python文件里，比如命名ssd\_detector.py.

OK， 下面修改一系列文件来训练自己的数据集   
两种方案， 第一：保持原来的文件目录结构及文件名不变， 只替换里面的数据。第二：重新新建一个与之前类似的目录结构，改成自己命名的文件夹，第二种方法，有一定的风险性，需要修改程序里涉及数据路径的代码。在之前讲解的faster-rcnnan那篇博客中， 我们采用第一种方案。本次我们采用第二种方案。   
在/data目录下创建一个自己的文件夹：

cd /data

mkdir mydataset

把/data/VOC0712目录下的create\_list.sh 、create\_data.sh、labelmap\_voc.prototxt 这三个文件拷贝到/mydataset下：

cp data/create\* ./mydataset

cp data/label\* ./mydataset

labelmap\_voc.prototxt, 此文件定义label。

在/data/VOCdevkit目录下创建mydataset, 并放入自己的数据集:

cd data/VOCdevkit

mkdir mydataset

cd mydataset

mkdir Annotations

mkdir ImageSets

mkdir JPEGImages

cd ImageSets

mkdir Layout

mkdir Main

mkdir Segmentation

其中Annotations中存放一些列XML文件，包含object的bbox，name等；   
ImageSets中三个子目录下均存放train.txt, val.txt, trainval.txt, test.txt这几个文件，文件内容为图片的文件名（不带后缀）；   
JPEGImages存放所有的图片；

在/examples下创建mydataset文件夹：

mkdir mydataset

文件夹内存放生成的lmdb文件。

上述文件夹创建好后， 开始生成lmdb文件, 在创建之前需要修改相关路径：

./data/mydataset/create\_list.sh

./data/mydataset/create\_data.sh

此时，在examples/mydataset/文件夹下可以看到两个子文件夹, mydataset\_trainval\_lmdb, mydataset\_test\_lmdb；里面均包含data.dmb和lock.dmb;

到此为止,我们的数据集就做好了。接下来就开始训练了。训练程序为/examples/ssd/ssd\_pascal.py，运行之前，我们需要修改相关路径代码：

cd /examples/ssd

vim sd\_pascal.py， 修改如下：

57行： train\_data路径；

59行：test\_data路径；

197-203行：save\_dir、snapshot\_dir、job\_dir、output\_result\_dir路径；

216-220行： name\_size\_file、label\_map\_file路径；

223行：num\_classes 修改为1 + 类别数

315行：num\_test\_image：测试集图片数目

另外， 如果你只有一个GPU, 需要修改285行：   
gpus=”0,1,2,3” ===> 改为”0”   
否则，训练的时候会出错。   
修改完后运行

python ./examples/ssd/ssd\_pascal.py

训练完， 修改ssd\_detector.py中模型路径， 任意找一张图片识别，看看效果怎么样。

如果在这过程中有什么问题， 可随时私信我。

参考：   
【1】《SSD: Single Shot MultiBox Detector》 By Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg.   
【2】<https://github.com/weiliu89/caffe/tree/ssd>