

YOLOv5目标检测实战-TensorRT加速

课程演示环境: Ubuntu 16.04, cuda 10.2, cudnn 7.6.5

1 安装pytorch

1) 安装Anaconda

Anaconda 是一个用于科学计算的 Python 发行版, 支持 Linux, Mac, Windows, 包含了众多流行的科学计算、数据分析的 Python 包。

1. 先去官方地址下载好对应的安装包 下载地址:<https://www.anaconda.com/download/#linux>
2. 然后安装anaconda

```
bash ~/Downloads/Anaconda3-2020.07-Linux-x86_64.sh
```

anaconda会自动将环境变量添加到PATH里面, 如果后面你发现输出conda提示没有该命令, 那么你需要执行命令 `source ~/.bashrc` 更新环境变量, 就可以正常使用了。如果发现这样还是没用, 那么需要添加环境变量。编辑`~/.bashrc` 文件, 在最后面加上

```
export PATH=/home/bai/anaconda3/bin:$PATH
```

注意: 路径应改为自己机器上的路径

保存退出后执行: `source ~/.bashrc` 再次输入 `conda list` 测试看看, 应该没有问题。

添加Anaconda国内镜像配置

清华TUNA提供了 Anaconda 仓库的镜像, 运行以下三个命令:

```
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
conda config --set show_channel_urls yes
```

2) 安装pytorch

注意: 如果使用yolov5版本v4.0以上的代码, 要使用pytorch1.7

首先为pytorch创建一个anaconda虚拟环境, 环境名字可自己确定, 这里本人使用python3.7作为环境名:

```
conda create -n python3.7 python=3.7
```

安装成功后激活pyhton3.7环境:

```
conda activate python3.7
```

在所创建的pytorch环境下安装pytorch的1.7版本, 执行命令:

```
conda install pytorch torchvision cudatoolkit=10.2 -c pytorch
```

注意：10.2处应为cuda的安装版本号

编辑~/.bashrc 文件，设置使用pytorch1.7环境下的python3.8

```
alias python='/home/bai/anaconda3/envs/python3.7/bin/python3.7'
```

注意：python路径应改为自己机器上的路径

保存退出后执行：`source ~/.bashrc`

该命令将自动回到base环境，再执 `conda activate python3.7` 到pytorch环境。

2 yolov5项目克隆和安装

1) 克隆yolov5项目

克隆项目到本地

```
git clone https://github.com/ultralytics/yolov5.git
```

或者直接下载V4.0代码

2) 安装所需库

使用清华镜像源：

在yolov5路径下执行：

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple -r requirements.txt
```

注意：simple 不能少，是 https 而不是 http

3) 下载预训练权重文件

下载yolov5s.pt, yolov5m.pt, yolov5l.pt, yolov5x.pt权重文件，并放置在weights文件夹下

百度网盘下载链接：

链接：<https://pan.baidu.com/s/14O704m9olHx8KK38Pf3RuQ> 提取码：y47n

注意：如果使用yolov5版本v4.0以上的代码，下载相应的权重

4) 安装测试

测试图片：

在yolov5路径下执行

```
python detect.py --source ./data/images/ --weights weights/yolov5s.pt --conf 0.4
```

3 安装TensorRT

参考[官网安装教程](#)

<https://docs.nvidia.com/deeplearning/tensorrt/install-guide/index.html>

1) 下载安装包:

1. Go to: <https://developer.nvidia.com/tensorrt>.
2. 点击 **立即下载(Download Now)**
3. 选择合适的TensorRT版本
4. Select the check-box to agree to the license terms.
5. Click the package you want to install. Your download begins.

下载得到文件TensorRT-7.0.0.11.Ubuntu-16.04.x86_64-gnu.cuda-10.2.cudnn7.6.tar.gz

2) 安装TensorRT:

1. 解压TensorRT-7.0.0.11.Ubuntu-16.04.x86_64-gnu.cuda-10.2.cudnn7.6.tar.gz
2. 将下面环境变量写入环境变量文件~/.bashrc并保存

```
export LD_LIBRARY_PATH=TensorRT解压路径/lib:$LD_LIBRARY_PATH
```

例如:

```
export LD_LIBRARY_PATH=/home/bai/TensorRT-7.0.0.11/lib:$LD_LIBRARY_PATH
```

#当cuda环境没有指定时, 也需要指定

```
export LD_LIBRARY_PATH=/usr/local/cuda-10.2/lib64:$LD_LIBRARY_PATH
export CUDA_INSTALL_DIR=/usr/local/cuda-10.2
export CUDNN_INSTALL_DIR=/usr/local/cuda-10.2
```

使刚刚修改的环境变量文件生效

```
source ~/.bashrc
```

3. 复制TensorRT路径下/lib、/include文件夹到对应系统文件夹

(非必须, 如果需要使用TensorRT进行编译, 这一步是必须的)

```
sudo cp -r ./lib/* /usr/lib          # 在TensorRT-7.0.0.11路径下执行
sudo cp -r ./include/* /usr/include  # 在TensorRT-7.0.0.11路径下执行
```

3) 安装pycuda

如果要使用python接口的tensorrt, 则需要安装pycuda

```
pip install pycuda
```

测试TensorRT

```
$ python
>>>import tensorrt
>>>tensorrt.__version__
```

4 测试TensorRT示例代码

1) 下载pgm文件

去tensorrt目录下的data文件夹找到对应数据集mnist的download_pgms.py，然后执行就可以了，

```
python download_pgms.py
```

等看到文件夹下有了x.pgm文件就说明下载好了

备注：PGM 是便携式灰度图像格式(portable graymap file format)，在黑白超声图像系统中经常使用PGM格式的图像。

2) 编译后可执行得到测试结果

编译：

```
cd TensorRT-7.0.0.11/samples/sampleMNIST
make clean
make
```

转到bin目录下面，make后的可执行文件在此目录下

```
cd /TensorRT-7.0.0.11/bin
```

执行

```
./sample_mnist
```

5 克隆tensorrtx

```
git clone https://github.com/wang-xinyu/tensorrtx.git
```

6 安装opencv

注意：OpenCV版本<=4.0（不要用最新版本）

本人安装的是opencv 3.4.4。首先到opencv官网下载opencv-3.4.4.tar.gz。执行以下命令

```
tar xvf opencv-3.4.4.tar.gz
```

```
cd opencv-3.4.4/
```

```
cmake .
```

```
make
```

```
sudo make install
```

在执行上述的cmake时可根据自己的电脑配置和安装的opencv版本情况设置命令参数：

(注意cuda的版本设置)

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \  
-D CMAKE_INSTALL_PREFIX=/usr/local \  
-D INSTALL_PYTHON_EXAMPLES=ON \  
-D INSTALL_C_EXAMPLES=OFF \  
-D OPENCV_EXTRA_MODULES_PATH=~/.opencv_contrib-3.4.4/modules \  
-D PYTHON_EXECUTABLE=/usr/bin/python \  
-D WITH_CUDA=ON \      # 使用CUDA  
-D WITH_CUBLAS=ON \  
-D DCUDA_NVCC_FLAGS="-D_FORCE_INLINES" \  
-D CUDA_ARCH_BIN="10.1" \      # 需要去官网确认使用的GPU所对应的版本  
-D CUDA_ARCH_PTX="" \  
-D CUDA_FAST_MATH=ON \  
-D WITH_TBB=ON \  
-D WITH_V4L=ON \  
-D WITH_QT=ON \      # 如果qt未安装可以删去此行;若因为未正确安装qt导致的Qt5Gui报错,  
-D WITH_GTK=ON \  
-D WITH_OPENGL=ON \  
-D BUILD_EXAMPLES=ON ..
```

本人使用的cmake命令如下:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D  
INSTALL_PYTHON_EXAMPLES=ON -D INSTALL_C_EXAMPLES=OFF -D  
OPENCV_EXTRA_MODULES_PATH=~/.opencv_contrib-3.4.4/modules -D  
CUDA_GENERATION=Auto -D PYTHON_EXECUTABLE=/usr/bin/python -D WITH_TBB=ON -D  
WITH_V4L=ON -D WITH_GTK=ON -D WITH_OPENGL=ON -D BUILD_EXAMPLES=ON ~/.opencv-  
3.4.4
```

`sudo make install` 执行完毕后OpenCV编译过程就结束了, 接下来就需要配置一些OpenCV的编译环境首先将OpenCV的库添加到路径, 从而可以让系统找到。

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
```

执行此命令后打开的可能是一个空白的文件, 不用管, 只需要在文件末尾添加 `/usr/local/lib`

执行如下命令使得刚才的配置路径生效

```
sudo ldconfig
```

配置bash

```
sudo gedit /etc/bash.bashrc
```

在最末尾添加 `PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig`
`export PKG_CONFIG_PATH`

保存, 执行如下命令使得配置生效

```
source /etc/bash.bashrc
```

更新

```
sudo updatedb
```

7 生成yolov5s.wts文件

// 下载权重文件yolov5s.pt // 将文件tensorrtx/yolov5/gen_wts.py 复制到ultralytics/yolov5 // ensure the file name is yolov5s.pt and yolov5s.wts in gen_wts.py // go to ultralytics/yolov5 执行

```
python gen_wts.py
```

// a file 'yolov5s.wts' will be generated.

// copy文件'yolov5s.wts' 文件到tensorrtx/yolov5/build目录下

8 编译tensorrtx/yolov5

// go to tensorrtx/yolov5 // ensure the macro NET in yolov5.cpp is s // update CLASS_NUM in yololayer.h if your model is trained on custom dataset

执行

```
mkdir build
cd build
cmake ..
make
```

如出现错误

1.FATAL:In-source builds are not allowed.

You should create separate directory for build files.

解决方法：1) 先删除刚才在当前目录下创建的CMakeCache.txt文件和CMakeFiles目录；

2) 再新建目录，比如build目录，在build目录执行cmake.

9 执行TensorRT加速后的yolov5命令

在tensorrtx/yolov5/build下执行

```
sudo ./yolov5 -s
```

// serialize model to plan file i.e. 'yolov5s.engine'

```
sudo ./yolov5 -d ../samples
```

// deserialize plan file and run inference, the images in samples will be processed.

或者：

在tensorrtx/yolov5下执行python脚本load and run the tensorrt model in python

// install python-tensorrt, pycuda, etc. // ensure the yolov5s.engine and libmyplugins.so have been built

```
python yolov5_trt.py
```

10 INT8量化加速

1. 准备校准图片（calibration images），可以从你的训练集随机选择 1000张图片。对于coco, 可以从百度网盘下载校准图片集 `coco_calib.zip`
2. unzip it in tensorrtx/yolov5/build
3. set the macro `USE_INT8` in yolov5.cpp and make

```
cd build  
make clean  
cmake ..  
make
```

4. serialize the model and test

```
sudo ./yolov5 -s
```

```
sudo ./yolov5 -d ../samples
```