2018年秋-第13周

上周已经阅读了一部分 ssn 的源码，然后作者的源码给的也比较完整，还是决定先把论文复现完再回过头看 faster rcnn。然后本科课程安排里面有个课设要验收了，相当于要复现一篇 find tiny face 的文章，所以这周也花了些时间在这个上面。

一、ssn

帧和光流的提取用到 dense_flow。

作者一开始用的一个 bash 命令，其中.sh 文件中比较重要的是：

```
python tools/build_of.py ${SRC_FOLDER} ${OUT_FOLDER} --num_worker ${NUM_WORKER} --new_width 340 --new_height 256
```

然后相当于运行 build_of.py，在这个里面：ext 的默认值是 avi，然后 activitynet 的数据是 MP4，而作者提供的那个.sh 文件中忽略了对--ext 取值进行说明，所以一开始报错

```
    parser.add_argument("--ext", type=str, default='avi', choices=['avi','mp4'], help='video file extensions')
```

之后用 gpu 对帧进行提取：

```
    pool = Pool(num_worker)
    if flow_type == 'tvl1':
        pool.map(run_optical_flow, zip(vid_list, xrange(len(vid_list))))
    elif flow_type == 'warp_tvl1':
        pool.map(run_warp_optical_flow, zip(vid_list, xrange(len(vid_list))))
```

将 mp4 文件路径和对应序号并行送到 run_optical_flow 函数，这个里面大部分是路径设置，有一行值得提一下：

```
    dev_id = (int(current._identity[0]) - 1) % NUM_GPU
```

这个代码可以获得当前任务使用的 gpu 的序号。

整个 run_optical_flow 比较核心的：

```
cmd = os.path.join(df_path + 'build/extract_gpu')+' -f {} -x {} -y {} -i {} -b 20 -t 1 -d {} -s 1 -o {} -w {
    quote(vid_path), quote(flow_x_path), quote(flow_y_path), quote(image_path), dev_id, out_format, new_size

os.system(cmd)
```

相当于一个命令行命令，具体的是调用编译后的 dense_flow。一开始没有编译的概念，所以找不到 build/文件夹，并且 readme 也没有提示，所以报错

```
Open ▾  +          errors.log [Read-Only]          Save  -  +  ×
                          trash:
File  Edit  View  Search  Tools  Documents  Help
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not foundsh: 1:
./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
sh: 1: ./lib/dense_flow/build/extract_gpu: not found
                 Plain Text ▾  Tab Width: 8 ▾     Ln 1, Col 1   ▾   INS
```

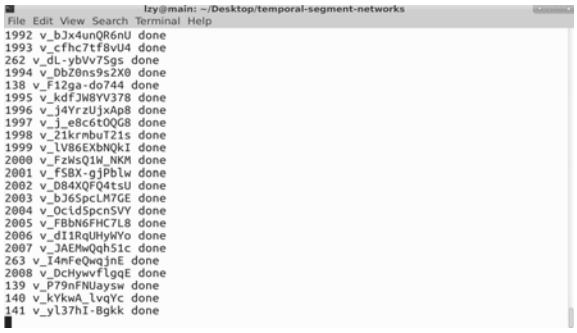
后来单独在 GitHub 上找 dense_flow：

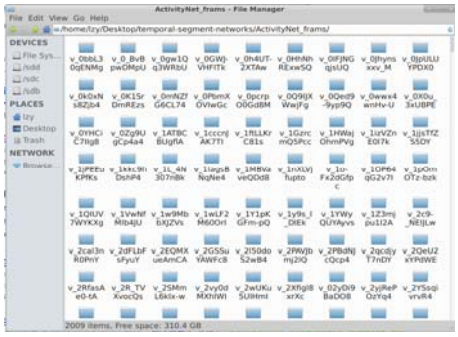
```
git clone --recursive http://github.com/yjxiong/dense_flow
mkdir build && cd build
cmake .. && make -j
```

cmake 编译 c 文件，然后 make –j 用来充分利用本机计算资源，总之编译之后就有了 extract_gpu

然后就利用编译好的文件开始提取帧和光流了：

```
cmd = os.path.join(df_path + 'build/extract_gpu')+' -f {} -x {} -y {} -i {} -b 20 -t 1 -d {} -s 1 -o {} -w {
    quote(vid_path), quote(flow_x_path), quote(flow_y_path), quote(image_path), dev_id, out_format, new_size
```





因为不同机器上的解码器可以输出不同数量的帧。所以要开始训练和测试，需要使提议列表适应为每个视频提取的实际帧数。即：

```
python gen_proposal_list.py DATASET FRAMES_PATH
```

其中核心代码：

```
frame_dict = parse_directory(args.frame_path, key_func=key_func)

process_proposal_list(norm_list_tmpl.format(configs['train_list']),
                      out_list_tmpl.format(configs['train_list']), frame_dict)
process_proposal_list(norm_list_tmpl.format(configs['test_list']),
                      out_list_tmpl.format(configs['test_list']), frame_dict)

print("proposal lists for dataset {} are ready for training.".format(args.dataset))
```

总结一下这个函数的输入输出：

parse_directory：输入帧文件夹地址；输出帧文件夹名、帧数、光流图像数

process_proposal_list：输入数据集给定的 Proposal、待输出匹配的 Proposal 路径、parse_directory 的输出；（输出为）直接在函数内部按照路径生成匹配的 Proposal

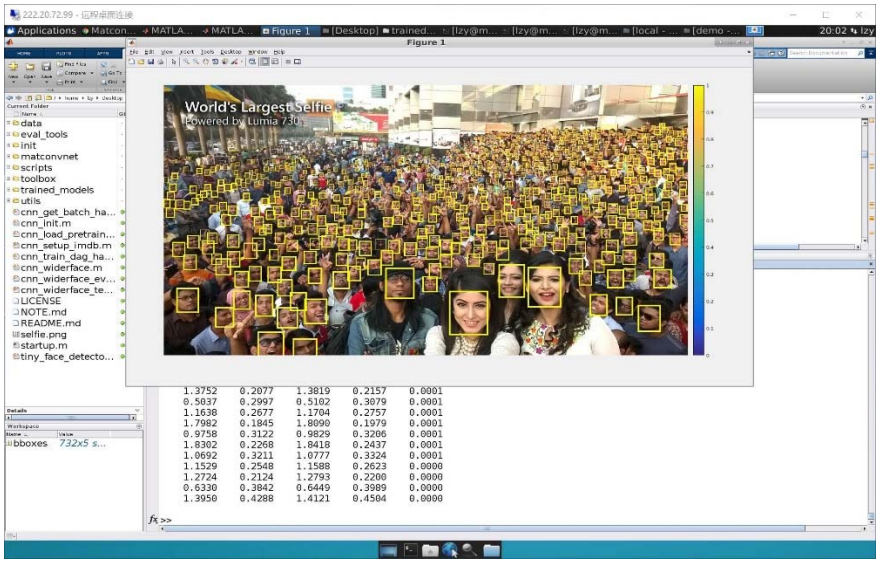
二、find tiny face

这一部分主要是在 cuda 和 cudnn 的配置上花了很多时间，而且现在还有一些关于

cudnn 的 bug 没有解决，但还是利用 cpu 顺利跑了作者提供的 demo。

因为服务器用的 cuda9.0 版本，所以有个 gpu_20 的问题查了很多资料才解决，但是 cudnn 好像还是不匹配。

Demo 效果如下图：



猜测原因是 matconvnet 没有安装好的问题，所以下一周除了继续复现 ssn 之外可能在这个部分也得花些时间。