

# 学生实验报告

学号	1120192933	学院	计算机学院
姓名	李桐	专业	人工智能

## 商品识别模型集成

### 1 实验目的

- (1) 理解 shell 脚本的作用。
- (2) 理解常用的 shell 命令。
- (3) 掌握 shell 脚本串行化一系列操作的能力。

### 2 实验原理

shell 命令

### 3 实验条件与环境

要求	名称	版本要求	备注
编程语言	Shell	无	
开发环境	dsw	无要求	
其他工具	无	无要求	
硬件环境	台式机、笔记本均可	无要求	

## 4 实验步骤及操作

步骤序号	1
步骤名称	串行化数据预处理
步骤描述	将数据预处理模块写成 shell 脚本
代码及讲解	<pre>def parse_args():     parser=argparse.ArgumentParser(description='input')     parser.add_argument('--name',default='')     parser.add_argument('--root',default='./data3')     parser.add_argument('--data_rpath',default='./data3')     parser.add_argument('--I_path',default='./data3/I/')     parser.add_argument('--V_path',default='./data3/V/')     parser.add_argument('--anns_spath',default='./data3/')      args=parser.parse_args()      return args  args=parse_args() name=args.name root=args.root data_rpath=args.data_rpath img_spath=args.I_path video_img_spath=args.V_path  timer_start=`date "+%Y-%m-%d %H:%M:%S"` export PYTHONIOENCODING=utf-8  python exp1_data.py  timer_end=`date "+%Y-%m-%d %H:%M:%S"` duration=`echo eval \$(( \$(date +%s -d "\${timer_end}") - \$(date +%s -d "\${ti echo "数据处理耗时: \$duration"</pre> <p>这一部分是进行数据的预处理，是实验 1 的内容，所有的参数都可以自己挑选输入，这里我把默认的参数都设置为了我的代码里面的参数。</p>

步骤序号	2
步骤名称	串行化检测模型的训练
步骤描述	将模型训练写成 shell 脚本
代码及讲解	<pre>import argparse def parse_args():     parser=argparse.ArgumentParser(description='input')     parser.add_argument('--ann_file_train',default='./data3/i_train.json')     parser.add_argument('--ann_file_test',default='./data3/v_train.json')     parser.add_argument('--I_path',default='./data3/I/')     parser.add_argument('--V_path',default='./data3/V/')     parser.add_argument('--work_dir',default='./expresult/det/')     parser.add_argument('--check',default='./data/cascade_rcnn_r50_fpn_20e      args=parser.parse_args()      return args</pre>

```
timer_start=`date +%Y-%m-%d %H:%M:%S`

python ./mmdetection/tools/train.py exp3.py

duration=`echo eval $(( $(date +%s -d "${timer_end}") - $(date +%s -d "${timer_start}") ))`
echo "检测模型耗时: $duration"
```

这一部分是进行检测模型的训练，是实验 3 的内容，我原来写成了所有的参数都可以自己挑选输入，这里我把默认的参数都设置为了我的代码里面的参数。就像上面第一个图这样，但是会报错(这一点在下面实验的讨论中有提到)。后来又改成了正常的，不接受外部的参数传入：

```
# parser.add_argument('--path', default='./data3/V/')
# parser.add_argument('--work_dir', default='./expresult/det/')
# parser.add_argument('--check', default='./data/cascade_rcnn_r50_fpn_20e_20181123-db483a09.pth')

# args=parser.parse_args()
# return args

# args=parse_args()
# ann_file_train=args.ann_file_train
# ann_file_test=args.ann_file_test
# img_spath=args.I_path
# video_img_spath=args.V_path

ann_file_train='./data3/i_train.json'
ann_file_test='./data3/v_train.json'
img_spath='./data3/I/'
video_img_spath='./data3/V/'
```

```
python mmdetection/tools/test.py exp3.py ./expresult/det/lastest.pth --format-only --options=jsonfile_prefix=./expresult/v_fs_cascade
python exp5-2_coco.py
python exp5-3_filter.py
```

训练好检测模型之后需要进行评估，并且将候选框进行初步筛选，即实验 5 的内容。但是这里参数可以输入，我已经将默认参数设定为我的参数。因为没有其他的文件的干扰了，之前不行的原因是 train、test 文件还需要传参，这里是直接运行这一个文件，只有这一个需要传参，所以可以。

```
def parse_args():
    parser=argparse.ArgumentParser(description='input')
    parser.add_argument('--original_json_I',default='./data3/i_train.json')
    parser.add_argument('--original_json_V',default='./data3/v_train.json')
    parser.add_argument('--now_json_I',default='./expresult/i_fs_cascade.bbox.json')
    parser.add_argument('--now_json_V',default='./expresult/v_fs_cascade.bbox.json')
    parser.add_argument('--will_json_I',default='./expresult/i_fs_cascade.json')
    parser.add_argument('--will_json_V',default='./expresult/v_fs_cascade.json')

    args=parser.parse_args()

    return args
```

```
args=parse_args()
original_json_I=args.original_json_I
original_json_V=args.original_json_V
now_json_I=args.now_json_I
now_json_V=args.now_json_V
```

```
def parse_args():
    parser=argparse.ArgumentParser(description='input')
    parser.add_argument('--now_json_I',default='./expresult/i_fs_cascade.json')
    parser.add_argument('--now_json_V',default='./expresult/v_fs_cascade.json')
    parser.add_argument('--will_json_I',default='./expresult/i_fs_cascade_filter.json')
    parser.add_argument('--will_json_V',default='./expresult/v_fs_cascade_filter.json')

    args=parser.parse_args()

    return args
```

```
args=parse_args()
now_json_I=args.now_json_I
now_json_V=args.now_json_V
will_json_I=args.will_json_I
will_json_V=args.will_json_V
```

步骤序号	3
步骤名称	串行化识别模型的训练
步骤描述	识别模型的训练写成 shell 脚本
代码及讲解	<pre>def parse_args():     parser=argparse.ArgumentParser(description='input')     parser.add_argument('--ann_file_train',default='./data3/i_train.json')     parser.add_argument('--ann_file_test',default='./data3/v_train.json')     parser.add_argument('--I_path',default='./data3/I/')     parser.add_argument('--V_path',default='./data3/V/')     parser.add_argument('--outpath',default='./expresult/rec')     args=parser.parse_args()      return args</pre> <p>这一部分是进行识别模型的训练，是实验 6 的内容，所有的参数都可以自己挑选输入，这里我把默认的参数都设置为了我的代码里面的参数。同样会报错，也是同样的改法。</p> <pre>timer_start=date "+%Y-%m-%d %H:%M:%S" python exp6.py  duration="echo eval `((\${date +%s -d "\${timer_end}"} - \${date +%s -d "\${timer_start}"}))`   awk '{t=split("60 s 60 m 24 h 999 d",a);for(i=1;i&lt;length(a);i++)a[i]=a[i]*a[0];print a[0]}'" echo "识别模型训练耗时: \$duration"</pre>

步骤序号	4
步骤名称	串行化商品识别模型预测
步骤描述	将商品识别模型预测写入 shell 脚本
代码及讲解	<pre> timer_start=\$(date +%Y-%m-%d %H:%M:%S) python exp7.py  duration=\$(echo eval `((\${date +%s -d "\${timer_end}") - \${date +%s -d "\${timer_start}"})`   awk '{t=split("60 s 60 m 24 h 999 d",a);for(i=1;i&lt;length(a);i++)t=t+a[i];print t}'` echo "搜索和匹配耗时: \$duration" </pre> <p>加载模型、构建数据集，将传入进来的图片、视频帧提取特征，然后进行搜索和匹配，如果传入多个模型，那就使用多个模型的结果进行融合。</p> <pre> def parse_args():     parser=argparse.ArgumentParser(description='input checkpoint')     parser.add_argument('--checkponit',default='./expresult/rec.pth')     #python try.py --checkponit='./results2/1-optimizer-Adam.pth', './resu     parser.add_argument('--I_json_path',default='./expresult/i_fs_cascade_     parser.add_argument('--V_json_path',default='./expresult/v_fs_cascade_     parser.add_argument('--I_path',default='./data3/I/')     parser.add_argument('--V_path',default='./data3/V/')     parser.add_argument('--output_path',default='./expresult/topk.txt')      args=parser.parse_args()      return args </pre>



#### (4) 最终结果的解读与讨论

我原来写得是这样的

```
timer_start=`date +%Y-%m-%d %H:%M:%S`
python exp6.py

duration=`echo eval `((${date +%s -d "${timer_end}") - ${date +%s -d "${timer_start}"})`) | awk '{t=split("60 s 60 m 24 h 999 d",a);for`
echo "识别模型训练耗时: $duration"
```

```
def parse_args():
    parser=argparse.ArgumentParser(description='input')
    parser.add_argument('--ann_file_train',default='./data3/i_train.json')
    parser.add_argument('--ann_file_test',default='./data3/v_train.json')
    parser.add_argument('--I_path',default='./data3/I/')
    parser.add_argument('--V_path',default='./data3/V/')
    parser.add_argument('--outpath',default='./expresult/rec')
    args=parser.parse_args()

    return args
```

我以为会直接按照默认的跑，结果给我报了错：

```
video: 0 0
数据处理耗时: 0s
usage: train.py [-h] [--ann_file_train ANN_FILE_TRAIN]
               [--ann_file_test ANN_FILE_TEST] [--I path I_PATH]
               [--V_path V_PATH] [--work_dir WORK_DIR] [--check CHECK]
train.py: error: unrecognized arguments: exp3.py
Exception ignored in: <function TemporaryFileCloser.__del__ at 0x7fc387326950>
Traceback (most recent call last):
```

后来发现这样不行，只能又改成了之前的样子，也就是 `config` 内部的参数不依靠外界获得：

```
# parser.add_argument('--v_path',default='./data3/v/')
# parser.add_argument('--work_dir',default='./expresult/det/')
# parser.add_argument('--check',default='./data/cascade_rcnn_r50_fpn_20e_20181123-db483a09.pth')

# args=parser.parse_args()

# return args

# args=parse_args()
# ann_file_train=args.ann_file_train
# ann_file_test=args.ann_file_test
# img_spath=args.I_path
# video_img_spath=args.V_path

ann_file_train='./data3/i_train.json'
ann_file_test='./data3/v_train.json'
img_spath='./data3/I/'
video_img_spath='./data3/V/'
```

不行的原因我认为是 `train`、`test` 文件还需要传参，如果是直接运行这一个文件，或者只有一个文件需要传参，那么就可以。

```
(PyTorch-1.8) [ma-user work]$timer_start= date +%Y-%m-%d %H:%M:%S"
(PyTorch-1.8) [ma-user work]$timer_end= date +%Y-%m-%d %H:%M:%S"
(PyTorch-1.8) [ma-user work]$duration= echo eval "$((date +%s -d "${timer_end}") - $(date +%s -d "${timer_start}"))" | awk '{t=split("60 s 60 m 24
h 999 d",a);for(n=1;n<t;n=2){if($1==0)break;s=$1a[n]a[n+1];s=int($1/a[n])}print s}'
(PyTorch-1.8) [ma-user work]$echo "耗时: $duration"
耗时: 0s
(PyTorch-1.8) [ma-user work]$
```

还有一个时间上的疑问，就是不论我中间隔了多久，这样去计算时间的时候，都会给出 0s，很怪，不知道为什么，可能是用的 smart art 平台的原因？

## 6 收获与体会

理解了 shell 脚本的作用、理解了常用的 shell 命令、掌握了 shell 脚本串行化一系列操作的能力。

```
(PyTorch-1.8) [ma-user work]$sh environment_solve.sh
sh: environment_solve.sh: No such file or directory
(PyTorch-1.8) [ma-user work]$sh environment_solve.sh
Looking in indexes: http://repo.myhuaweicloud.com/repository/pypi/simple
Looking in links: https://download.openmmlab.com/mmcv/dist/cu102/torch1.8.0/index.html
Collecting mmcv-full
  Downloading https://download.openmmlab.com/mmcv/dist/cu102/torch1.8.0/mmcv_full-1.5.1-cp37-cp37m-manylinux1_x86_64.whl (42.4 MB)
    | 42.4 MB 3.2 MB/s
Requirement already satisfied: numpy in /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages (from mmcv-full) (1.21.5)
Requirement already satisfied: yapf in /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages (from mmcv-full) (0.32.0)
Requirement already satisfied: opencv-python<3 in /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages (from mmcv-full) (4.1.2.30)
Requirement already satisfied: pyyaml in /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages (from mmcv-full) (6.0)
Requirement already satisfied: addict in /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages (from mmcv-full) (2.4.0)
```

使用 smart art，每次进入环境都需要安装，所以写了一个脚本，其实就已经接触到了 sh 文件。顺便吐槽一下，环境不能一次开太长时间，要不花钱很多。但是来回换 cpu、gpu 还要用好久时间安装环境。好不容易舍得开 gpu 了，结果安装了半个小时环境。

一直在用 smart art 写，刚才从 smart art 上面把代码下载下来了，我发现时间居然不是最后一次修改的时候，而是下载的时间。

## 7 备注及其他

无。