2018-09-第二周

`CASIA`

1. 工作总结与安排

上周总结

1. 本周主要对机器学习所需要的Nvidia显卡支持进行了环境配置，分别在Windows10以及Ubuntu 18.04下进行，并调通；
2. Tensorflow-gpu这个库在win10上的支持并不好，会出现很多不知明的问题，慎用；Pycharm IDE会有很多bug，比如Debug不正常，或者页面无法关闭等等，请谨慎使用；
3. 在Ubuntun下安装cuda前要先安装合适的Nvidia驱动，这个过程会遇到黑屏，也就是系统无法识别到N卡，以下详述；
4. 本周通过对孟老师对深度学习和点云知识的讲解，掌握了深度学习的一些基本概念、基本工作过程以及三维点云的基本知识；学习了Logitic回归过程，使用numpy进行向量化；
5. 分析了SequeezeSeg的项目工程，熟悉了深度学习工程的结构，项目构建的基本过程，以下详述；
    - 总结
      本周在环境上浪费了不少时间，在神经网络这块有了一定的基本了解，开始慢慢熟悉起来，大体能看懂代码中各个部分的作用；下周集中精力对输入转换进行编写，深度学习知识及时补充；

下周安排

1. 修改SequeezeSeg代码，将竞赛数据转换成SequeezeSeg可用的输入数据，使用给出的数据进行模型训练；
2. 分析SequeezeSeg代码，掌握深度学习项目组织方式及简单应用的写法；
3. 熟悉tensorflow，掌握基本用法；
4. 深度学习基础持续学习中。。。

2. 学习记录

论文SequeezeSeg

1. 使用python脚本查看了工程用npy格式数据的格式，如下：

```
1.
2.  #!/usr/bin/python3
3.
4.
5.  import numpy as np
6.  import argparse
7.
8.  def load(path):
9.      npy = np.load(path)
10.     print(np.shape(npy))
11.     print(npy)
12.
13. if __name__ == '__main__':
14.     parser = argparse.ArgumentParser(description = "script")
15.     parser.add_argument("--path", type = str, default = None)
16.
17.     args = parser.parse_args()
18.     print("parser path: ")
19.     print(args)
20.
21.     load(args.path)
```

如下：

```
Namespace(path='./2011_09_26_0093_0000000432.npy')
(64, 512, 6)
[[[ 20.4279995   20.36599922   1.18499994   0.40000001  28.87007713   0.        ]
  [ 20.47699928  20.2859993    1.18400002   0.20999999  28.84841537   0.        ]
  [ 20.97400093  20.64999962   1.204        0.28999999  29.45812035   0.        ]
  ...,
  [ 24.03800011 -23.65500069   1.35000002   0.52999997  33.75214005   0.        ]
  [ 23.91900063 -23.68700027   1.347        0.15000001  33.68986511   0.        ]
  [ 22.90200043 -22.89599991   1.30400002   0.14        32.41032028   0.        ]]

 [[ 20.62299919  20.54999924   1.00899994   0.31        29.13123131   0.        ]
  [ 20.64900017  20.4470005    1.00699997   0.46000001  29.07705498   0.        ]
  [ 21.12199974  20.7840004    1.023        0.17        29.65063286   0.        ]
  ...,
  [ 24.04400063 -23.68400002   1.13600004   0.5         33.76886749   0.        ]
  [ 23.9470005  -23.66300011   1.13399994   0.18000001  33.68504715   0.        ]
  [  0.           0.           0.           0.          0.           0.        ]]

 [[ 20.46100044  20.44199944   0.86699998   0.34999999  28.93578339   0.        ]
  [ 20.54899979  20.40099907   0.86799997   0.27000001  28.9692173    0.        ]
  [ 20.82999992  20.48699951   0.87400001   0.25999999  29.22960663   0.        ]
  ...,
  [ 23.89900017 -23.54199982   0.972        0.13        33.56088257   0.        ]
  [ 23.54400063 -23.3390007    0.963        0.16        33.16558838   0.        ]
  [ 23.10899925 -23.0529995    0.95200002   0.13        32.65536499   0.        ]]

 ...,
 [[  0.           0.           0.           0.          0.           0.        ]
  [  0.           0.           0.           0.          0.           0.        ]
  [  0.           0.           0.           0.          0.           0.        ]]
```

其中，最后一维的6个分量，分别为

$x, y, z, intensity, r = \sqrt{x^2 + y^2 + z^2}, category$，共32768个点，即一个npy文件包含了64*512个点的数据；训练数据共10848个，即共有10848个npy文件；

[文件]

2. 在imdb.py的read_batch里，有数据完整的组装过程，即imdb通过读入.npy格式数据，组装成模型需要的输入；

   通过read_batch将npy组织成**lidar_per_batch**（batch * height * width * 5），**lidar_mask_per_batch**（batch * height * width *

1），**label_per_batch**（batch * height * width），这个部分还在理；

深度学习基础知识点

- 噪点
  图像噪声，是一种亮度或颜色信息的随机变化现象，由传感器和电器等产生；
- 坏点
  图像出现在相同位置处的杂点，而噪点的位置是随机的；
- 图像卷积
- 卷积核（滤波器）：可做为神经元
- 全卷积（FCN）
  CNN->卷积->pooling->上采样(反卷积)->复原
- 上采样
- 池化（下采样）
- 1X1 卷积

Python知识点拾遗

- 本周学习参考地址：
  Python3
- 本周练习存放地址：
  1. python类及高级用法
  2. vectorization向量化用法
- PS：
  1. python3.0中整除用//，其他所有的除法都是精确除法；
  2. __xxx__，如例，以__ 开始并以之结束为特殊变量，而以__开始的为private变量；
  3. 如果要获得一个对象的所有属性和方法，可以使用dir()函数；
  4. pip list 查看当前安装的库， pip uninstall 删除某个库；
  5. inf 无穷，nan：not a number；
  6. Tensorflow tf.placeholder 占位符用法：

```
1.
```

```
#!/usr/bin/env python


# _*_ coding: utf-8 _*_


import tensorflow as tf
import numpy as np


# 定义placeholder

input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)


# 定义乘法运算

output = tf.multiply(input1, input2)


# 通过session执行乘法运行

with tf.Session() as sess:

# 执行时要传入placeholder的值

    print sess.run(output, feed_dict = {input1:[7.],input2: [2.]}
)


#以上结果 [ 14.]
```

7. easydict

> 可以方便地应用 . 来访问dict的值。

例如，普通的dictionary，访问值只能用下面的方式：

```
In [9]: d = {'foo':3, 'bar':{'x':1, 'y':2}}

In [10]: d['foo']
```

```
Out[10]: 3


In [11]: d.foo
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-15fca6602818> in <module>()
----> 1 d.foo

AttributeError: 'dict' object has no attribute 'foo'
```

使用edict 包装后，就不一样了：

```
In [12]: from easydict import EasyDict as edict

In [13]: easy = edict(d)

In [14]: easy.foo
Out[14]: 3
```

8.

```
class Chain(object):
    def __init__(self, path=''):
        self._path = path
    def __getattr__(self, path):
        return Chain('%s/%s', %(self._path, path))
    def __str__(self):
        return self._path

    __repr_ = __str__
```

Ubuntu18.04下安装cuda

1. 分盘策略
   efi 分区（efi格式）： 在UEFI模式下需要有该分区（必须）
   /boot （ext4）启动分区，在第一次安装时分配了200M，过程中提示了错误，虽然系统进去了，但是后期会发现少了很多库，所以第二次重装的时候分配了1024M，需知悉；

swap（swap格式）交换分区，一般与系统的内存大小相等就可以了；
/（ext4）用户根目录，系统大小，可多分配；
/home（ext4）用户工作目录，可多分配；

2. gcc与g++的安装
   这个是ubuntu16.04下跟18.04下安装cuda的不同的一点，默认情况18.04下安装的是gcc与g++ 7.0的版本，而安装cuda需要的是小于6的版本，所以我们要对gcc、g++进行降级；

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 50   #
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 50   #
```

3. 驱动安装
   安装了几次系统，黑屏的情况全部倒在这一步，后来解决的方式的是让系统自己安装所有合适的驱动，而不只是安装它推荐的版本，如下：

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt update
```

   查看可用的驱动：

```
ubuntu-drivers devices
```

   画重点：

```
sudo ubuntu-drivers autoinstall
```

   后续：

```
reboot
sudo apt install nvidia-cuda-toolkit
nvcc --version
```

4. cuda安装
   这一步唯一要注意的是已经安装过驱动，安装cuda的过程中不要再重复安装；

```
sudo ./cuda_linux.run -toolkit -samples -silent -override
```

会检查少哪些库，以下安装少的库：

```
sudo apt-get install freeglut3-dev build-essential libx11-dev
libxmu-dev libxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev
```

5. 环境变量设置
   设置环境变量：

```
export PATH=/usr/local/cuda-9.1/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-9.1/lib64\                    $
{LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}
```

6. 验证安装
   验证cuda安装：

```
cd NVIDIA_CUDA-9.0_Samples/5_Simulations/fluidsGL
make clean && make
./fluidsGL
```

PS：每隔几秒监视显卡运行状态

```
watch -n 10 nvidia-smi
```

其他

- CloudCompare
  图形化点云软件，ubuntu下安装：

```
# cloudcompare安装（ubuntu）
$ snap install cloudcompare
```