

注意：Deeplab于2019年11月在github上进行了更新，文档中部分地方区分了旧版（2019年11月以前）和新版（更新后的github版本）。

学员可使用百度网盘旧版deeplab目录下载旧版models.tar.gz或者使用从github上克隆新版deeplab进行项目实践。

1 安装tensorflow

1.1 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
```

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。

1.2 添加Aanaconda国内镜像配置

清华TUNA提供了 Anaconda 仓库的镜像，运行以下命令：

```
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
```

1.3 安装tensorflow

注意：需要安装tensorflow 1.10以上的版本 创建tensorflow环境，环境名字可自己确定，这里本人使用tfgpu作为环境名：

```
conda create -n tfgpu python=3.7
```

安装成功后激活tfgpu环境：

```
source activate tfgpu
```

在所创建的tfgpu环境下安装tensorflow的gpu版本, 执行命令:

```
conda install tensorflow-gpu==1.15.0
```

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

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

保存退出后执行: source ~/.bashrc

该命令将自动回到base环境, 再执行 `source activate tfgpu` 到tfgpu环境。

注意: 已安装tensorflow的学员可使用升级命令:

```
sudo pip install --upgrade --ignore-installed tensorflow-gpu==1.15.0 -i  
https://pypi.tuna.tsinghua.edu.cn/simple
```

或使用Anaconda环境下安装命令:

```
conda install tensorflow-gpu==1.15.0
```

2 Deeplab项目安装及测试

2.1 克隆deeplab项目

官方代码github地址: <https://github.com/tensorflow/models/tree/master/research/deeplab>

```
git clone https://github.com/tensorflow/models.git
```

2.2 添加项目依赖路径

编辑文件~/.bashrc

```
sudo gedit ~/.bashrc
```

加上一句:

```
export  
PYTHONPATH=/home/bai/models/research/slim:/home/bai/models/research:$PYTHONPATH
```

执行

```
source ~/.bashrc
```

2.3 测试deeplab

```
cd /home/bai/models/research/deeplab
```

执行:

```
python model_test.py
```

最后输出应没有错误信息

如果有错误信息: ModuleNotFoundError: No module named 'tf_slim'

执行下面三个命令:

```
cd /home/bai/models/research/slim
```

```
python setup.py build
```

```
python setup.py install
```

3 数据集处理

数据集处理分成三步:

- 标注数据 (对CamVid省略)
- 制作指引文件
- 将数据转换成TFRecord

3.1 CamVid数据集下载

下载地址: 百度网盘链接

链接: <https://pan.baidu.com/s/1YDwnIpyh-x4TY31teKMDfg>
提取码: nhrp

数据集的文件夹结构为:

|— test |— testannot |— train |— trainannot |— val |— valannot

由上到下分别是测试集、测试集标签、训练集、训练集标签、验证集、验证集标签。

对于CamVid, 其中训练集、测试集、验证集中的图片数目分别为train 367, test 233, val 101。

3.2 制作指引文件

TF 提供了一种统一输入数据的格式—— TFRecord 它有两个优点:

1. 可以将一个样本的所有信息统一起来存储, 这些信息可以是不同的数据类型。其内部使用“Protocol Buffer”二进制数据编码方案。
2. 利用文件队列的多线程操作, 使得数据的读取和批量处理更加方便快捷。

在制作TFRecord之前, 需要有文件指引将数据集分类成训练集、测试集、验证集, 故需要创建指引文件。

将所有图片和mask分在两个文件夹下, 设置如下:

/home/bai/dataset/CamVid/image: 存放所有的输入图片, 共有701张, 这其中包括训练集、测试集、验证集的图片。

/home/bai/dataset/CamVid/mask: 存放所有的标签图片, 共有701张, 和image文件夹下的图片是一一对应的。

对于CamVid数据集, 创建了一个目录/home/bai/dataset/CamVid/index, 该目录下包含三个.txt文件:

- train.txt: 所有训练集的文件名称
- trainval.txt: 所有验证集的文件名称
- val.txt: 所有测试集的文件名称

3.3 将数据转换成TFRecord

创建文件夹tfrecord:

```
mkdir tfrecord
```

将上述制作的数据集打包成TFRecord, 使用的是build_voc2012_data.py

在目录/home/bai/models/research/deeplab/datasets下执行

```
python build_voc2012_data.py \  
  --image_folder="/home/bai/dataset/CamVid/image" \  
  --semantic_segmentation_folder="/home/bai/dataset/CamVid/mask" \  
  --list_folder="/home/bai/dataset/CamVid/index" \  
  --image_format="png" \  
  --output_dir="/home/bai/dataset/CamVid/tfrecord"
```

- image_folder : 数据集中原输入数据的文件目录地址
- semantic_segmentation_folder: 数据集中标签的文件目录地址
- list_folder : 将数据集分类成训练集、验证集等的指引目录文件目录
- image_format : 输入图片数据的格式, CamVid的是png格式
- output_dir: 制作的TFRecord存放的目录地址(自己创建)

4 网络训练

4.1 修改训练脚本

在DeepLabv3+模型的基础上, 主要需要修改以下两个文件

- data_generator.py
- train_utils.py

添加数据集描述

在datasets/data_generator.py文件中, 添加camvid数据集描述:

```
_CAMVID_INFORMATION = DatasetDescriptor(  
    splits_to_sizes={  
        'train': 367, # num of samples in images/training  
        'val': 101, # num of samples in images/validation  
    },  
    num_classes=12,  
    ignore_label=255,  
)
```

因为CamVid共有11个classes, 所以加上 ignore_label 共12个物体类别。

注册数据集

同时在datasets/data_generator.py文件, 添加对应数据集的名称:

```
_DATASETS_INFORMATION = {
    'cityscapes': _CITYSCAPES_INFORMATION,
    'pascal_voc_seg': _PASCAL_VOC_SEG_INFORMATION,
    'ade20k': _ADE20K_INFORMATION,
    'camvid': _CAMVID_INFORMATION, #camvid示例
}
```

修改train_utils.py

对应的utils/train_utils.py中，将关于 `exclude_list` 的设置修改（新版第210行；旧版第159行），作用是在使用预训练权重时候，不加载该 logit 层：

```
exclude_list = ['global_step', 'logits']
if not initialize_last_layer:
    exclude_list.extend(last_layers)
```

修改train.py

如果想在DeepLab的基础上fine-tune其它数据集，可在deeplab/train.py中修改输入参数。

其中有一些选项：

- 使用预训练的所有权重，设置`initialize_last_layer=True`
- 只使用网络的backbone，设置`initialize_last_layer=False`和`last_layers_contain_logits_only=False`
- 使用所有的预训练权重，除了logits以外。因为如果是自己的数据集，对应的classes不同(这个我们前面已经设置不加载logits)，可设置`initialize_last_layer=False`和`last_layers_contain_logits_only=True`

使用的设置是：

`initialize_last_layer=False` #157行 (新版)

`last_layers_contain_logits_only=True` #160行 (新版)

`initialize_last_layer=False` #133行 (旧版)

`last_layers_contain_logits_only=True` #136行 (旧版)

4.2 下载预训练模型

在model_zoo上下载预训练模型：

下载地址：https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md

下载的预训练权重为`xception_cityscapes_trainfine`

文件大小为439M

下载到deeplab目录下，然后解压：

```
tar -zxvf deeplabv3_cityscapes_train_2018_02_06.tar.gz
```

需要注意对应的解压文件目录为：

```
/home/bai/models/research/deeplab/deeplabv3_cityscapes_train
```

注意：在训练CamVid时，没有考虑类别之间的平衡问题，所以没有做imblance的修正。如果你是二分类或者存在严重的类别不平衡情况，可以参考后面roadscene分割项目案例中的类别不平衡修正方法。

注意如下几个参数：

- tf_initial_checkpoint: 预训练的权重，因为CamVid和自己的RoadScene数据集都和CityScapes类似，所以使用的是CityScapes的预训练权重
- train_logdir: 训练产生的文件存放位置
- dataset_dir: 数据集的TFRecord文件
- dataset: 设置为在data_generator.py文件设置的数据集名称

4.3 CamVid上的训练指令

在目录 /home/bai/models/research/deeplab下执行

```
python train.py \
  --logtostderr \
  --training_number_of_steps=300 \
  --train_split="train" \
  --model_variant="xception_65" \
  --atrous_rates=6 \
  --atrous_rates=12 \
  --atrous_rates=18 \
  --output_stride=16 \
  --decoder_output_stride=4 \
  --train_crop_size=321,321 \
  --train_batch_size=4 \
  --dataset="camvid" \
  --
tf_initial_checkpoint='/home/bai/models/research/deeplab/deeplabv3_cityscapes_train/model.ckpt' \
  --train_logdir='/home/bai/models/research/deeplab/exp/camvid_train/train' \
  --dataset_dir='/home/bai/dataset/CamVid/tfrecord'
```

Q: 如何设置train_crop_size的值?

A: $\text{output_stride} * k + 1$, where k is an integer. For example, we have 321x321, 513x513。

5 网络测试

5.1 测试结果可视化

在目录 ~/models/research/deeplab下执行

```
python vis.py \
  --logtostderr \
  --vis_split="val" \
  --model_variant="xception_65" \
  --atrous_rates=6 \
  --atrous_rates=12 \
  --atrous_rates=18 \
  --output_stride=16 \
  --decoder_output_stride=4 \
```

```
--vis_crop_size=360,480 \
--dataset="camvid" \
--colormap_type="pascal" \
--checkpoint_dir='/home/bai/models/research/deeplab/exp/camvid_train/train' \
--vis_logdir='/home/bai/models/research/deeplab/exp/camvid_train/vis' \
--dataset_dir='/home/bai/dataset/CamVid/tfrecord'
```

- vis_split: 设置为测试集
- vis_crop_size: 设置360,480为图片的大小
- dataset: 设置为我们在data_generator.py文件设置的数据集名称
- dataset_dir: 设置为创建的TFRecord
- colormap_type: 可视化标注的颜色

可到目录deeplab/exp/camvid_train/vis下查看可视化结果

5.2 性能评估

在目录 /home/bai/models/research/deeplab下执行

```
python eval.py \
--logtostderr \
--eval_split="val" \
--model_variant="xception_65" \
--atrous_rates=6 \
--atrous_rates=12 \
--atrous_rates=18 \
--output_stride=16 \
--decoder_output_stride=4 \
--eval_crop_size=360,480 \
--dataset="camvid" \
--checkpoint_dir='/home/bai/models/research/deeplab/exp/camvid_train/train' \
--eval_logdir='/home/bai/models/research/deeplab/exp/camvid_train/eval' \
--dataset_dir='/home/bai/dataset/CamVid/tfrecord' \
--max_number_of_evaluations=1
```

- eval_split: 设置为测试集
- crop_size: 同样设置为360和480
- dataset: 设置为camvid
- dataset_dir: 设置为我们创建的数据集

查看mIoU值:

tensorboard --logdir /home/bai/models/research/deeplab/exp/camvid_train/eval

查看训练过程的loss:

tensorboard --logdir /home/bai/models/research/deeplab/exp/camvid_train/train

