**学习周报**

1. **学习内容**
2. 学习pytorch框架的基础知识

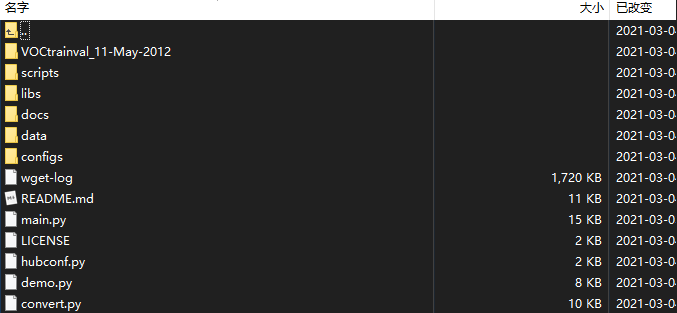
主要学习了安装gpu版本pytorch库、构造Tensor的方法、常用计算函数、autograd等梯度问题、梯度优化器torch.optim、数据加载器dataLoader、torchvision预处理图片以及加载自带数据集、保存以及加载模型、model.train()和model.eval()的区别、数据和模型转移到显卡上运行的方法、 多GPU并行计算等

1. Deeplab算法理解和代码阅读

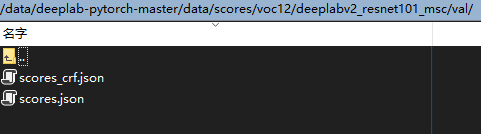
（1）借助微信公众号的整理https://mp.weixin.qq.com/s/rvP8-Y-CRuq4HFzR0qJWcg，已理解Deeplab算法的核心思想；与此同时，跑通了pytorch版本的deeplab代码，tensorflow版的比较繁杂就没有继续研究了。

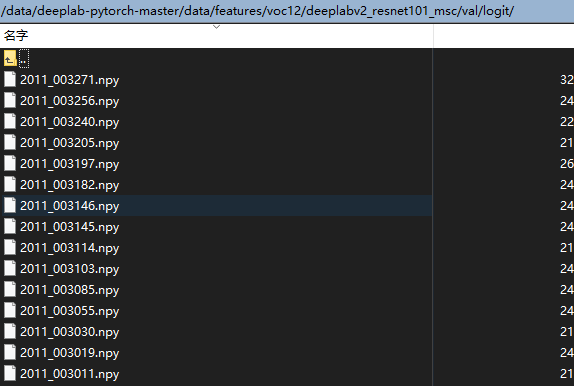
（2）关于实验

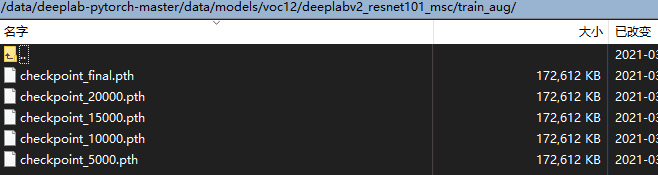
【代码目录】：

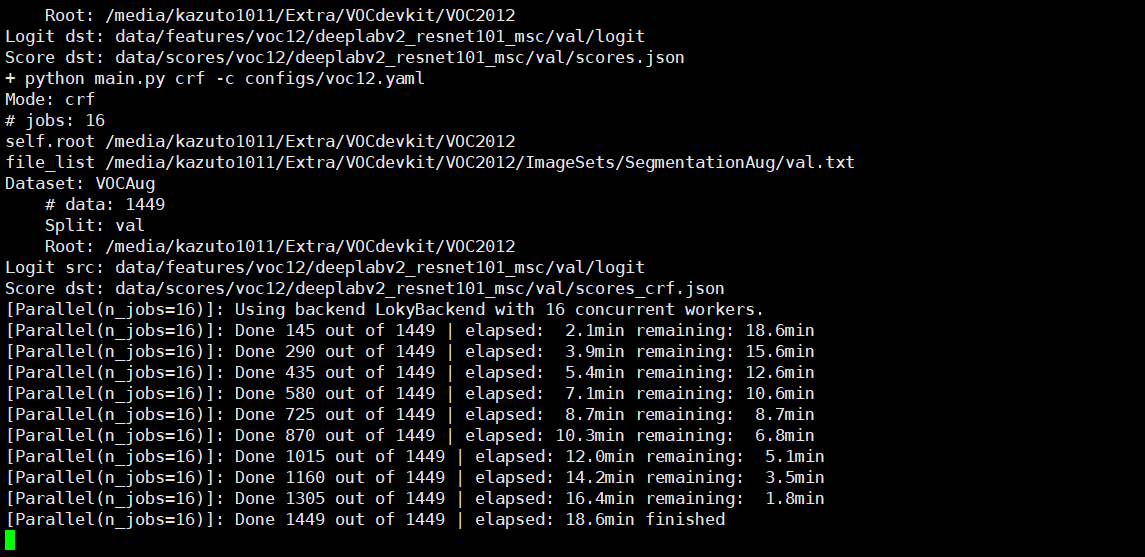


【实验代码结果】：









【实验代码梳理与分析】：

* 实验运行过程：

使用configs/ conda\_env.yaml来创建满足要求的conda环境；使用Scripts/setup\_voc12.sh脚本下载并解压数据集至voc.yaml文件中的ROOT位置（/media/kazuto1011/Extra/VOCdevkit），其中voc12数据集下载完成后还需要下载SegmentationAug并放置在Imagesets目录下，即数据增强版；利用Scripts/setup\_caffemodels.sh来下载预训练好的caffe模型文件，然后使用”python convert.py”进行转化；最后使用Scripts/train\_and\_eval.sh来运行模型训练、测试以及加上crf后处理的过程。

* 代码结构分析：

Libs文件是代码的核心部分:其中的models子文件下是模型deeplab v1、v2、v3、msc的实现文件; utils包含了自定义的学习率调整、crf后处理、语义分割评价方法的实现; datasets子文件夹下定义了数据集父类base.py来抽象数据集从文件名加载文件、进行数据增强的功能，然后voc.py等具体数据集文件定义了具体数据集的子类并结合scripts中yaml配置文件进行使用；

data文件夹是模型训练结果保存的位置，其下的logs子文件夹是日志，features文件夹保存了模型在数据集上预测结果的npy矩阵文件，models子文件下保存了训练过程中的模型参数pth文件，scores是模型在测试集上的评价指标结果json文件；

Main.py是模型训练和测试的主文件，主要过程就是获得训练和测试数据集的dataLoader、构造DeepLabV2\_ResNet101\_MSC模型、加载了caffe预训练模型的参数进行参数初始化、定义损失函数、进行模型训练和测试、保存模型参数并用tensorboard的SummaryWriter记录一些统计数据。

1. DenseNet图像分类算法理解和代码阅读

（1）算法理解

【概念】：Densenet与Resnet有相似之处，但它采用的是密集连接方式，即每一层都要接受其前面所有层的feature map作为输入，输入时所有的feature map在channel维度上进行连接,且每层的输出都是固定channel数的feature map。

【结构】：其具有DenseBlock + Transition的网络结构：DenseBlock由多个DenseLayer层组成，每个层的特征图大小相同，层与层之间采用密集连接方式；Transition用于连接任意两个相邻的DenseBlock块，并且通过Pooling使得特征图的大小减小。由于DenseLayer层之间的密集连接方式，一个DenseBlock的最后部分必然导致特征图的channel数过高，故提出了与Resnet中类似的BottleNeck来解决此问题。BottleNeck由“BN-RL-1\*1Conv-BN-RL-Conv”组成，其引入了1\*1的conv将特征图的channel先降低一个维度（通常是4\*growth\_rate)，然后利用3\*3Conv输出growth rate个channel特征图。

【优点】Densenet带来了更强的梯度流动，一方面提升了梯度的反向传播使得网络更加容易训练，另一方面由于每一层都直连input和loss所以能减轻梯度消失；减少了参数数量；

保存了低维度的特征，倾向于给出更平滑的决策边界。

【缺点】Densenet多次连接操作需要数据的多次复制，显存容易增加得快，需要一定的显存优化技术；densenet是一种更特殊的网络，不如resnet泛化性更强。

1. 代码 DenseBlock + Transition + DenseLayer + BottleNeck 模型类

依据参数不同有densenet121、densenet161、densenet169、densedensenet201



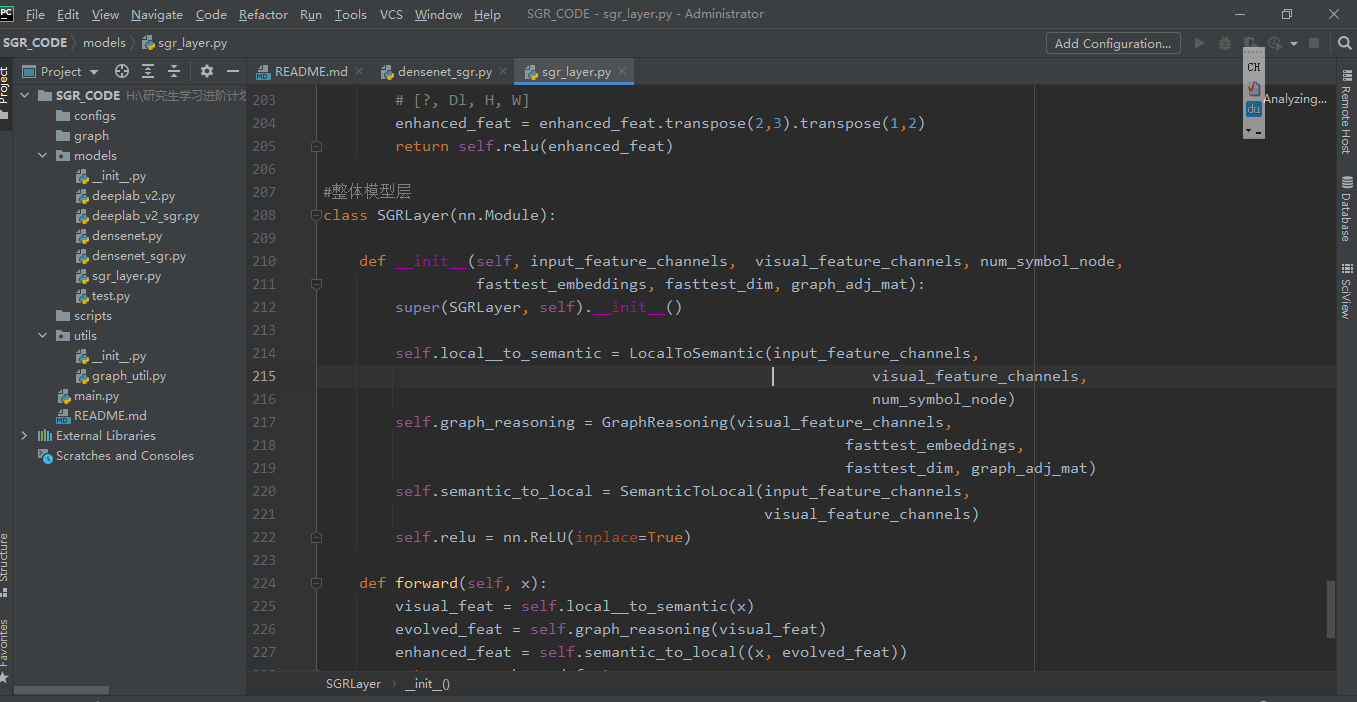
1. 编写NIPS的代码

原先论文代码是tensorflow版，已将其重写成pytorch定义的模型类；

分类任务Densenet和densenet\_sgr已经完成；

图像语义分割任务上的deeplab代码可以复用；

模型训练测试以及metric都还没写。



1. **学习问题**
2. 知识图的数据未搞定，原先只有coco这一数据集的adj\_mat，但是也没有原始图
3. Pytorch相比于tensorflow版代码要更易懂、模块化更清楚，但是有些重要功能没有提供，导致操作不方便：pytorch中是[？，Channel，H，W]这种图片处理方式，论文中的频繁的矩阵维度切换不如tensorflow中[？，H，W，Channel]方便；pytorch没有tf.map\_fn

功能，无法方便批量处理图片；pytorch中[?,M,N] 与 [?, N,P]的三维矩阵相乘不支持

1. **学习计划和安排**
2. 重点解决知识图数据的生成问题
3. 编写cifar-100上的图像分类问题上的模型训练与测试代码