NJU_NLP_SummerCamp_2019_report_week1

> 2019.07.01~2019.07.07
> @author Eric ZHU

本周学习内容

1. Tensorflow的基础使用方法（因课程安排，暂时使用Tensorflow；下周考虑开始学习Pytorch）
2. 基础的多层感知机网络与深度学习
3. 以CNN为主的图像识别系统（LeNet-5 手动实现 + AlexNet, InceptionV3调用实现）
4. 课程内容：图像增强，基础的迁移学习（基于预训练的VGG和ResNet）与 Adversarial Attack原理 + 实现

部分重点笔记

1. 深度学习：一类通过多层非线性变换对高复杂性数据进行建模算法的集合
2. 非线性激活函数的意义：使神经网络去线性化，拥有更强的表达能力
3. 正则化的意义：减小模型的复杂度，避免过拟合
4. 超参数对模型的影响
    1. 学习率：过高过低的学习率都会因为无法优化而降低模型有效容限；学习率最优点在训练的不同时间点都可能变化，因此需要有效的学习率衰减策略。
    2. 损失函数超参数（如正则化系数）：多个损失函数之间基本保持数量级相近，不建议对其进行大幅度修改。
    3. 批样本容量：增大Batch_size有助于加快模型收敛，提高模型的精度，但会增大内存负担，使得网络的训练过程需要经过更多个epoches.
    4. Dropout比例：增加dropout可以显著避免过拟合，一般添加在输入层或输出层，常用值为0.2到0.5
    5. 模型深度：同条件下，增加深度意味着模型具有更多参数，更强的拟合能力，但对时间和计算力的要求也就越高。此外，过深的网络往往难以训练，还更容易受到 Adversarial Examples 的攻击。
    6. 卷积核尺寸：增加卷积核尺寸可以增大过滤器的感知域，提高模型对特征的抽取能力，但会显著增大计算量；在实践中常常用多个小尺寸卷积层的串联代替大尺寸卷积层（如 Inception 中用1x1卷积层进行升降维操作；用两个3x3卷积层的串联替代5x5卷积层，减少了约45%的参数量）。
5. 在数据量不足的情况下，合理使用预训练网络进行微调（fine-tune，对瓶颈层进行再训练）可以大幅提升模型的效果与表现。
6. Tencent AI Lab 对 Image Caption 问题的解决策略：LSTM + 强化学习（训练一个对 Caption 结果的打分器）

附录

1. 7.6-7.7_CIFAR-10_CNN 实验记录（见下页）
2. 7.4-7.5_MNIST_CNN （见下页）
3. 7.3_MNIST 实验记录 （见下页）

基于CNN的CIFAR-10数据集分类任务

> 时间：2019-07-06
> 以下代码参考了Tensorflow的Advanced CNN教程与示例代码

图像输入部分

1. 数据读取：用CPU进行所有的图像读取与预处理工作，用GPU进行模型的训练工作，提高模型训练的效率。在训练开始时，预处理20000张处理过的CIFAR图像填充到随机化处理队列中，避免图像I/O过程影响模型的训练速度。
2. 图像增强（训练过程中）：对原始图像进行随机切割，翻转，调整（随机失真），增大训练样本的数据量。
    1. 切割：略小于原始图像，增加训练数据量并减小计算量
    2. 翻转：对图像随机进行左右翻转
    3. 亮度调整：对图像随机进行亮度调整（在一定范围内）
    4. 对比度调整：对比度增大阈值大于减小阈值（高对比度常常有助于识别）
3. 图像增强（测试过程中）：
    1. 切割：从原图像中心进行切割，防止影响图像主体
    2. 标准化：对原图像的RGB值进行线性标准化，使模型对图像的动态范围变化不敏感。

模型预测部分

该网络在AlexNet的基础上进行了一定修改，其模型结构如下。

层名称	说明
conv1	采用5x5卷积核，步长为1，全0填充，过滤器深度为64，激活函数为ReLU
pool1	采用3x3最大池，步长为2*
norm1	LRN层，对同一层响应较小的神经元进行抑制
conv2	采用5x5卷积核，步长为1，全0填充，过滤器深度为64，激活函数为ReLU
norm2	LRN层，对同一层响应较小的神经元进行抑制
pool2	采用3x3最大池，步长为2
local3	含有384个节点的全连接层，激活函数为ReLU
local4	含有192个节点的全连接层，激活函数为ReLU
softmax_linear	生成最终结果的softmax层

*：重叠池化（Overlapping Pooling），步长小于池化范围，可以抽取更强的特征表达，但增大了计算量。

模型训练部分

模型的目标函数是求交叉熵损失（Cross_entropy）和所有权重衰减项（L2）的和。使用标准的梯度下降法对参数进行优化，采用滑动平均值调整参数，用指数衰减法调整学习率。

模型调整

> 训练机配置如下：
>
> CPU : I7-8700 (6C12T)
> GPU: RTX2060 (6G)
> Tensorflow-gpu = 1.14.0
> CUDA = v10.0
> cuDNN = v7.3.1
>
> 原始参数如下：
>
> ```
> Batch_size = 128
> Steps = 20000
> Moving_Average_Decay = 0.9999
> Num_Epochs_per_decay = 350
> Learning_Rate_Decay_Factor = 0.1
> nitial_Learning_Rate = 0.1
> ```

改动描述	训练速度	测试集正确率
原始参数	9600 ex./s, 0.013 sec/batch	89.9%
Batch_size = 256 (增大一倍)	11000 ex./s, 0.023 sec/batch	92.7%
不使用滑动平均	9800 ex./s, 0.013sec/batch	87.6%
删去两个LRN层	13000 ex./s, 0.010sec/batch	90.0%
在local4与softmax_linear间加入84神经元的全连接层	9200 ex./s, 0.014sec/batch	89.4%
Batch_size = 256, 训练次数= 50k	11000 ex./s, 0.023sec/batch	96.1%

调整结果

1. 增大Batch_size有助于加快模型收敛，提高模型的精度，但会增大内存负担，使得网络的训练过程需要经过更多个epoches.
2. 滑动平均模型在一定程度上优化了模型的表现；学习率的指数衰减使得较多的训练次数产生效果。
3. LRN层果然好像没有什么用...

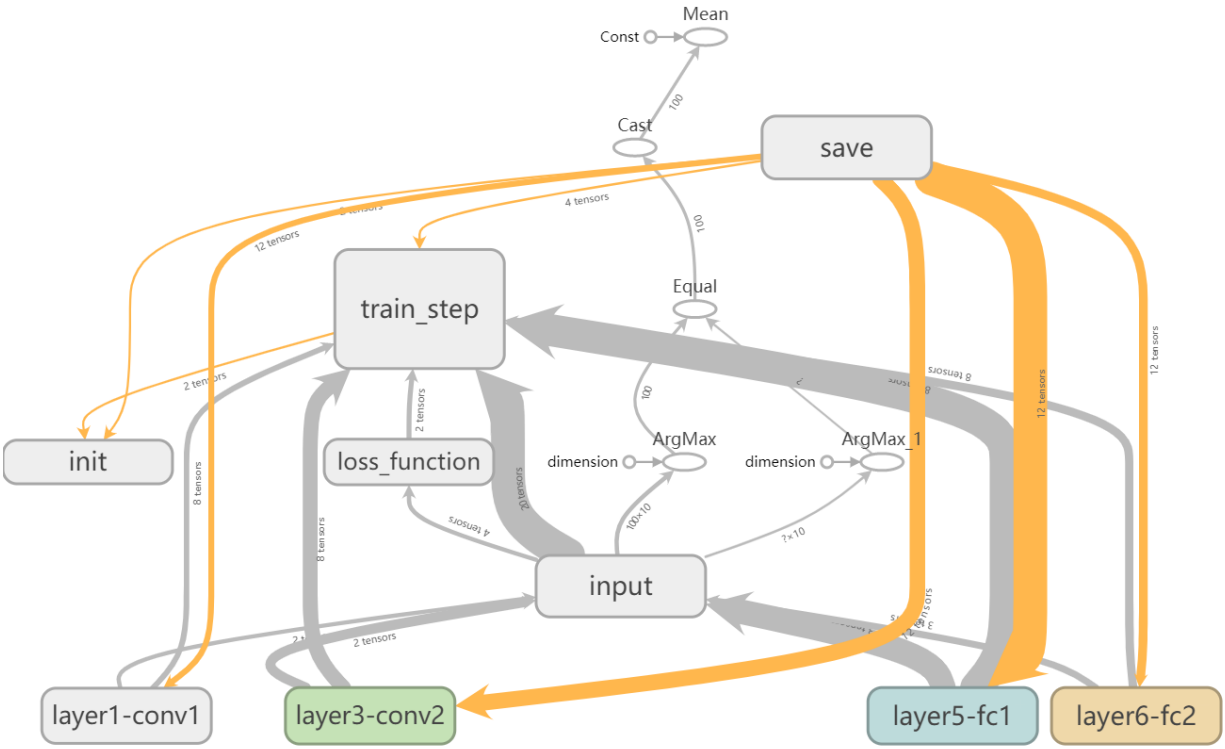
基于卷积神经网络(CNN)与MNIST数据集的分类任务

> 时间：2019-07-05

文件结构

名称	描述
MNIST_data\	数据集本身
log\	训练过程中的日志文件
model\	保存的训练完成的模型文件
data\	关于记录编写的其他文件
mnist_inference.py	模型的预测部分（前向传播）
mnist_train.py	模型的训练部分（反向传播）
mnist_eval.py	模型的评估器

模型结构细节



a. 预测部分

1. 输入层：784个节点（等于输入图片像素）
2. 隐藏层：共两层
    1. 卷积层，采用5*5卷积核，深度为32，不采用全0填充，步长为1，激活函数为ReLU
    2. 池化层，采用2*2最大池，长宽步长均为2
    3. 卷积层，采用5*5卷积核，深度为64，不采用全0填充，步长为1，激活函数为ReLU
    4. 池化层，采用2*2最大池，长宽步长均为2
    5. 全连接层，512个节点，L2正则，激活函数为ReLU，训练过程中dropout为50%
3. 输出层：10个节点 （等于输出类别数目），L2正则

b.训练部分

1. 采用AdamOptimizer进行优化
2. 采用交叉熵平均值与正则化损失的和为损失函数
3. 采用指数衰减法调整学习率
4. 对训练完成的模型进行保存，便于后续评估调用

c. 评估部分

1. 调用前阶段保存的模型，基于MNIST的测试集进行评估。

测试结果与反思

测试结果

对于本项目的网络模型，训练次数为1k，在MNIST数据集上达到了99.16%的正确率。

```
After 1001 training steps, validation accuracy = 0.9916
```

结果反思

以下纪录在模型训练与调整中遇到的问题

1. 对于较高的学习率初值，在训练初期可能出现无法收敛的情况；为此应调整学习率初值。（一般1e-2到1e-5）
2. 训练过程中，全连接层采用dropout可以显著地避免过拟合，提高模型的泛化能力。
3. 复杂的网络模型将会大幅影响模型训练的速度。（例如：在池化层1与卷积层2间插入一个Inception-v3模块，利用云计算资源进行训练，在3k次训练时可以达到99.26%的正确率，但模型训练的时间增加了约6倍）

```
After 3001 training steps, validation accuracy = 0.9926
```

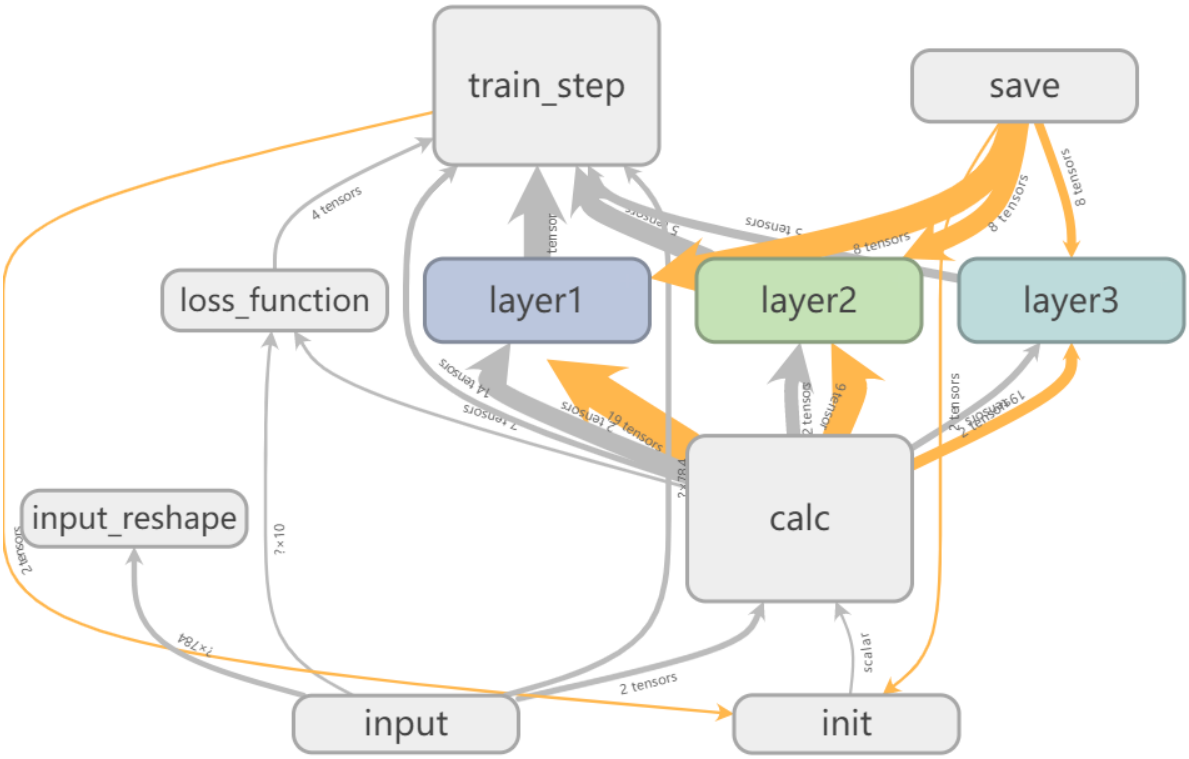
基于神经网络与MNIST数据集的分类任务

> 时间：2019-07-04

文件结构

名称	描述
MNIST_data\	数据集本身
log\	训练过程中的日志文件
model\	保存的训练完成的模型文件
data\	关于记录编写的其他文件
mnist_inference.py	模型的预测部分（前向传播）
mnist_train.py	模型的训练部分（反向传播）
mnist_eval.py	模型的评估器

模型结构细节



a. 预测部分

1. 输入层：784个节点（等于输入图片像素）
2. 隐藏层：共两层
    1. 该层共250个节点，采用RELU作为激活函数，L2正则避免过拟合，滑动平均模型优化。
    2. 该层共250个节点，采用RELU作为激活函数，L2正则避免过拟合，滑动平均模型优化。
3. 输出层：10个节点 （等于输出类别数目），使用L2正则避免过拟合，滑动平均模型优化。

b.训练部分

1. 采用批量梯度下降法进行优化
2. 采用交叉熵平均值与正则化损失的和为损失函数
3. 采用指数衰减法调整学习率
4. 对训练完成的模型进行保存，便于后续评估调用

c. 评估部分

1. 调用前阶段保存的模型，基于MNIST的测试集进行评估。

测试结果与反思

测试结果

对于4层神经网络（一层输入，两层隐含，一层输出），取Batch大小为100，训练次数为10k，该模型在测试集上可以达到最高98.34%的正确率。

```
Use standard file APIs to check for files with this prefix.
I0704 16:55:57.963792 32348 saver.py:1280] Restoring parameters from c:\Users\zhuxi\tf\path\to\model\model.ckpt-10001
After 10001 training steps, validation accuracy = 0.9834
```

结果反思

以下纪录在模型训练与调整中遇到的问题

1. 引入激活函数RELU后，对模型的正确率有了极大的提升（92%-->97%），引入L2正则化，滑动平均模型与指数衰减法后，进一步提高了模型的正确率（97%-->98%）。
2. 网络结构与神经元个数对正确率的影响：
    1. 对于3层神经网络（一层输入，一层隐含，一层输出），将隐含层节点个数由500提升至800后，正确率获得了显著的提升（97.2%-->98.0%）
    2. 对于4层神经网络（784+250+250+10），正确率略高于3层结构（784+800+10），深层神经网络可以更好地提取数据中的特征。
3. 在训练过程中，出现了损失函数突然增大的现象，需要解决 //TODO