1月12日

临近寒假之际，指导老师李瑞轩召集小组成员开会，给我们在寒假做了一些安排，查看文献，熟悉框架，同时还给每个成员安排了实验室师兄师姐指导，叮嘱我们积极和师兄师姐沟通，争取在开学前完成文献翻译和开题报告的初稿。

和实验室师兄李成杰做了一些关于课题的交流。师兄简明扼要的告诉我课设开展的流程和准备工作，精准的发给了一些实用参考文献，和编程所需要的指导教程，并帮我从中挑选了一篇英文文献做翻译文献，这篇文献对这个课题有很大的启发作用，帮我在开展课设时劈开了不少的坑。

。

1月14日

和实验室师兄李成杰做了一些关于课题的交流。师兄简明扼要的告诉我课设开展的流程和准备工作，精准的发给了一些实用参考文献，和编程所需要的指导教程，并帮我从中挑选了一篇英文文献做翻译文献，这篇文献对这个课题有很大的启发作用，帮我在开展课设时避开了不少的坑。

1月20日

经过几天在网上收集资料，对深度学习有了初步的了解。先是了解深度学习中的神经网络，其次和课题紧密相关的卷积神经网络（CNN），了解到CNN结构形式为卷积层，池化层，全连接层。接着对CNN模型参数优化的算法，主要了解基于梯度下降思想的各种优化算法。其中，理解了使用梯度下降思想寻找多元函数的最优解，感到这种思想精妙绝伦，和大一学的微积分中牛顿法求方程的解有相似之处。

1月16日

师兄推荐的一篇英文文献Deep learning with Elastic Averaging SGD\_Full，主要讲的是深度学习的一种优化算法。之前，我对深度学习是一无所知的，所以一开始上来阅读英文文献，结合翻译工具加上自己的英文底子，文章能翻译过来，但很多地方看不懂，不过还是勉强把整篇文章读了三遍，最后还是选择先去互联网上了解深度学习的基础知识。

1月23日

了解到卷积神经网络的优化算法SGD后，我再一次读了英文文献，这一次有很多之前不明白的点豁然开朗了，所谓EASGD，是在SGD的基础上加入了一个移动平均值的参数，旨在减少与参数服务器的通信开销。今天开始对文献的翻译工作。

1月22日

经过前几天的学习，对深度学习的优化算法随机梯度下降，有了新的理解，以一个二次函数为例，所谓梯度即是导数，梯度下降的过程则是沿着函数曲线切线渐进的过程，所谓步长或者学习率，其实是导数定义中的分母，这样也就很好理解为什么学习率过大会导致震荡，因为学习率过大，切点就会在极值处两侧来回。

1月24日

前几日我了解了单机CNN模型的训练流程和原理，今天主要了解分布式的原理。分布式中的核心是如何构建合理的分布式集群中节点的通信架构，目前来说有两种成熟的通信架构parameter-serer和Ring All-Reduce。Parameter-server中需要中心聚合点，故通信容易成为瓶颈，Ring All-reduce 没有中心聚合点，所有节点地位同等，但扩展性受到限制。

1月24日

结合自己的英语底子和翻译工具，对文献进行翻译工作。有了对相关知识的了解，翻译的也比较准确，再这个过程也加强了一下自己的英语学习。翻译的过程又让我加强了对EASGD的理解，弹性参数很小时，模型可以进一步探索参数空间。这次文献翻译工作给了一个经验，对论文的翻译要先广泛了解其基本知识，因为论文中省略了很多中间过程。

1月28日

前些天完成了文献翻译，今天准备开始撰写开题报告。先在网上寻找国内外现状的一些资料，补充了分布式深度学习当前流行的通信架构和优化算法相关知识，比如adam,基于动量的SGD。由于这一次对报告格式要求特别严格，花费了很多时间在报告的格式上比如公式插入及交叉引用，图片插入题注及交叉引用，样式的使用。

1月25日

经过这是十几天的学习，基本明白了整个课题的工作内容，今天特意从总体上回顾了一下。首先使用vmvare搭建分布式环境，采用Parameter-server通信架构，采用CNN架构在mnist和cifar10上进行训练，使用同步SGD，异步SGD，限制异步频率的部分同步SGD，梯度延迟补偿的异步SGD等优化算法，对模型的训练时间和准确率作出评估。

1月30日

今天天继续写开题报告，写到研究方案时，对照样板，实在不知道怎么写，就从网上参考，写了文献参考法和实验法，总之写的很概括不够具体，主要是对课题还是准备不足，但可能这段时间写累了，就这样草草了事。

2月16日

过年这段时间基本没有关注过课题，过完年后把之前的开题报告发给指导老师斧正。老师首先指出文章的一些格式问题，其次指出在分布式框架中内容不够充分，研究方案太够抽象，不够具体。果然当初偷的懒现还得重新弥补。在分布式框架中对节点间通信进行扩充，在研究方案中加入了细化的工作流程。

2月18日

开题报告定稿后，开始制作开题答辩的PPT。在网上找了一个精美的PPT模板，将开题报告的关键信息填充到模板，同时也学习了PPT制作的一些小技巧。制作完PPT，对着PPT做了两次预演，至此，开题报告部分完成。

3月9日

今天开始答辩，看着同学们一个个降讲的流畅自然，眉飞色舞，我不禁有些慌张，因为之前制作完PPT后预演了两编之后就在也没有预演了，今天只是在脑子里大致回顾了一下流程。果不其然，我为我的轻视付出了代价，在演讲过程中，流程有些混乱，一些地方解释不清，最后“惨烈”收场。

3月15日

今天先入门了一下pytorch框架，了解了pytorch中的tensor,并大致读懂了使用Pytorch训练cifar10的代码。以前一直不懂什么叫做框架，就像大一刚学系的时候不懂什么叫做集成开发环境（IDE），Pytorch框架就是python中的一个用于深度学习的功能模块，凭着C/C++，Java等语言的底子，配合一些网络查询，代码很容易就弄懂了，但对python语言还一无所知。

3月13日

今天安装了pytorch框架。这个安装过程并不轻松，按照官网给出的安装命令安装，经常出现httperror，换了国外代理，也并没有改善。通过在网上查找解决方案，也是一波三折。我是通过conda来安装的，网上说更换源并设置默认源，使用官网命令依然没有解决问题，后来又说官网命令中要去掉最后两个字，那代表的是使用官方源，这样才把pytorch安装成功。

3月16日

今天开始接触pythony语言。先是了解其基本数据结构，和其他语言差别不大，然后了解几种逻辑结构，和其他语言形式略有不同，if 条件判断语句不能加括号，for 循环形式也有点差异，让我感受最深的是没有显示的变量定义，变量没有类型，对象才有类型，变量只是对象的引用，和java中引用有点类似。总之，一边看教程，一边在安装好的pytroch上做着测试，了解了其基本知识。

3月18日

今天继续学习python的模块导入部分。这个部分和java打包导入机制是类似的，了解到import好from \* import 两种导入机制的区别，import 导入的包有自己的命名空间，from \* import \* 将导入的符号加入了被导入包的命名空间。所以我在使用global（）时因为没有很好理解这一点，对我造成了很大的困扰，始终找不到所定义的全局类。

3月20日

今天学习了python中各种格式化输出print，形式上和其他语言差别较大。同时继续理解python中全局变量和局部变量的定义，也有一个类似于C语言中的main函数。总而言之，通过这几天的学习，我发现python的其内部原理机制更多的类似于java，完全忽视了内存管理，这让我略微不习惯。

3月25日

休闲几天，今天在网上找了好几份深度学习的源码，准备了解一下深度学习训练的编程过程，可是在第一部分就卡住了。深度学习第一步，准备数据集，pytorch中提供了下载数据集和预处理的函数，但问题是这个函数下载数据集老是发生httperror，换代理依然不管用。网上一堆解决方案，但是靠谱额没几个，只能按挨个挨个试，从其他地方把数据集下载下来放在相应目录，这种方法不管用，因为pytorch中对其进行了验证和处理，并不一致。于是严格按照其目录中的文件设置，才通过了。下载数据集这个过程大概花了我两天时间。不得不感慨网上信息实在是鱼龙混杂。

3月28日

今天使用在windows10 上使用VMWARE创建了几台虚拟机并安装pytorch。前几天在windows10上安装好了pytorch并下载好了数据集，把坑都踩完了，然后依次在几台虚拟机重复windows10上的操作。后来知道了虚拟机可以克隆，所以只需要在一台虚拟机上配置好，然后就克隆配置好的虚拟机就行了。虚拟机的硬件设置关机后可以随时改变，方便我们布置不同的分布式环境。

4月1日

依然做一些前置准备工作，今天主要设置参数服务节点为静态IP，方便程序运行和调式。在虚拟机中使用NAT模式和主机共享IP，原本虚拟机是DHCP动态分配IP，在ubutnu18.04修改、etc/netplan/\*.yal文件，设置静态IP地址。修改过程中多次出现格式问题，和python一样也是靠缩进，其中注意到tab键和空格键的使用区别，平时使用其他IDE时，编辑器自动帮我们转换了。修改后运行sudo netplan apply 命令，静态IP修改成功。

4月3日

这一次课设编程特意去学习了一次版本控制，下载了windows版的github desktop。其实在大四上学期做虚拟机实验已经初步介绍的在ubuntu上git命令行的使用，版本控制的大致工作是熟悉的。Windows上使用图形界面形式github desktop，无需教程就能轻松操作，绑定了自己的github账号。为了确保万一,在github desktop上做了一些实验。

4月2日

由于虚拟机中编程实在太卡，我是在windows 10上编好的代码，能够运行，在迁移到虚拟机上去。在windows10上使用vscode进行编程，发现vscode的终端无法使用conda 命令进行切换到激活pytroch的python，仔细阅读官网，做出多番尝试，原来conda在安装过程在windows上只对cmd进行了初始化设置，没有对powershell初始化，而vscode默认终端识poewshell，官网中也支持对其他shell，按照其方法操作并没有成功，查找多方资料也没有成功，最后只能将vscode的默认终端改为cmd。

4月4日

前面几天基本都是做准备工作，没有涉及到课题的难点，从今天开始算是正式进入编程流程。首先从网上找到一份深度学习训练的源码，尝试读懂并实现。

前几天已经准备好了数据集，使用torchvison.dataset可以直接加载验证已准备好的数据集，使用torch.nn.module构建卷积神经网络，并且再网上了解了一下神经网络的参数设置技巧，对于Mnist数据集，其太过简单，随便设置参数都能取得很好的训练效果，cifar10数据集则暂时参照网上的设置。

4月5日

今天主要学习了深度学习过程的损失函数，损失函数是用来评估模型和真实数据之间的差异。在图像分类中主要使用交叉熵函数(CrossEntropyLoss)。交叉熵函数是对CNN的输出进行softmax，得到每个样本的概率分布向量，在对其取lg对数，对每个样本的概率向量选取其中真实类别位置的概率对数的绝对值相加再取其平均值，一开始的loss值应当在2.31左右。

4月7日

今天进一步了解了pytorch中CNN的参数设置。使用torch.nn.sequential构建卷积层池化，只需要设置好输入通道，输出通道，卷积核大小，卷积核滑动步长，零填充长度这个5个参数，就能构建一个卷积层，选择relu作激活函数，maxpool作池化，使用dropout防止过拟合。最后使用torch.nn.linear构建全连接层，使用该函数之前需要用view方法将其一维化。

4月8日

今天继续实现深度学习教程的训练部分。选择合适参数优化器，使用pytorch再点基于动量的SGD，对数据集中的每一个mini-batch先使用model进行前向传播，计算出输出值，再讲输出值和样本值传给损失函数，进行后向传播，计算出梯度,使用优化器step，进行一次参数更新，就这样循环多次。最终训练10epochs，训练效果就达到了94.2%。虽然没有达到网上说的随便就能达到98%的准确度，但至少流程没什么问题，也就没有细细探究其中的问题。

4月11日

今天继续学习分布式训练的通信。使用pytorch中TorchProcess创建一个进程，设置参数服务节点IP地址和端口号，使用torch.distributed.init\_process\_group初始化分布式进程组，采用’gloo’通信协议，在计算机节点和参数服务节点都调用这个方法，只不过传rank参数时注意传自己的rank。

4月10日

之前初步学习了深度学习的训练过程，今天学习分布式训练方法。首先我们使用DistributedSampler设置分割策略，Dataloader加载划分到该节点的数据集子集。这次完成分布式训练第一步数据集的分布式划分，这种数据集分布策略避免了数据集在各节点的传输，节省了训练时间。

4月13日

昨天发现pytorch分布式不支持windows，把代码移植到在虚拟机ubuntu上运行，结果还是报拒绝连接错误。网上查找原因，先是尝试需要先运行参数服务节点，可依然不行，又有文章提到需要添加环境变量，使用gloo协议，则需要添加环境变量GLOO\_IFPATH-

\_NAME=ens33 (网卡名称)，这样才解决问题。不过这也是我第一次碰到这种添加环境变量的情况。

4月12日

昨日完成了分布式通信部分的编程。今天在windows上运行时，报错提示没有init\_process\_group方法。我依次排除方法名打错了，没有导入相应的包的错误，最后在网上查找原因，有人提到是pytorch的分布式不支持windows，在官网下载的的时候选择的是windows版本，这让我误以为pytorch支持windows，所以一直没有想这个原因，再上官网仔细看这个包的说明，发现的确不支持windows。

4月15日

到目前为止，我已经学会了分布式通信和深度学习训练的编程实现，只需要将两者结合起来，而这也没有难度。先是在cdmodel.py中定义了在mnist和cifar10数据集上CNN模型，为了弄懂定义的每一个方法的精准含义（之前是靠猜的），上官方文档仔细阅读，前面几次出现的问题都证明了官方文档有提到过，网上的教程也会出现纰漏，同时加深对CNN的理解。

4月14日

今天在测试分布式通信的代码时，init\_process\_group又突然出现拒绝连接的错误。上网也没查找到原因，正好万事不决就重启。重启后解决了问题，但之后又出现了了同样的问题，而且也不能每次都靠重启来解决，虚拟机重启太慢了。这一天不断的上网查找原因实验都没解决问题，至到晚上脑子灵光一闪会不会是进程号别占用，使用ps aux|grep python查看最近进程，并结束掉，使用lsof -i:port查看使用的固定的端口号的进程别结束掉，果然把问题解决了。

4月17日

今天继续计算节点的编程。Main方法部分完成了，torchprocess开启一个计算节点进程，在这之前定义了两个能够父子进程间通信的列表，用来存储迭代次数和平均损失值。在计算节点中，一开始使用scatter完成计算节点和参数服务节点模型初始参数的随机一致，每一次循环中都要完成所分到的数据子集的遍历，前向传播计算输出值，后向传播计算梯度值并使用torch.distribuetd.send向参数服务节点发送梯度，使用receive接收更新后的模型参数，计算每一个mini-batch的loss值累加求平均，并添加到存之前定义的存储列表中，计算进程结束后打印损失函数曲线。

4月16日

今天开始编程实现计算节点。现在python的main方法中注册任务安排网络队列，设定了另一个没用的端口号，这个队列在注册的网络进程组是共享的，这样能方便参数服务节点能够调度任务。在这里也出现了错误，这个对象是不可能lamda调用的，源码实现也是参考网上实现的，我估摸者需要自定义一个方法，把方法作为参数，而不是网络队列这个对象的引用，这样才把问题解决了。

4月18日

计算节点的编程完成后，也同样查看官方文档，弄懂每一个方法的精准定义。了解了enumerate是遍历一个列表，元组，可迭代对象，返回元素数据和索引；也不太清楚，为什么vscode中总队torch.ones\_like总是报没有该方法的错误，运行的时候也没问题，估计是vscode中的配置不到位吧，网上也没查找到原因，也弄清楚了model.parameter里包含了网络模型参数和对应的梯度。损失函数的后向传播就能自动更新parameter和梯度。

4月19日

为了便于代码的可扩展性，使用的globals，它返回当前命名空间一个全局变量的字典。也是借此机会搞懂了import和from \* impot \*的区别在于命名空间的不同，import导入的模块的命名空间和被导入文件的命名空间是独立的，所以使用的时候要加上包名；而from \* import \*方式导入的模块中所有符号加入到当前文件命名空间。同时也搞懂了全局变量和局部变量的区别。在方法内部使用全局变量需要加上global关键词，除了在main方法不需要。

4月20日

今天开始编写参数服务节点代码。先是注册任务网络队列，开启参数服务节点进程，注册分布式进程组，使用scatter分发随机的初始模型参数。开始进入循环，先是使用receive接收计算节点传递过的梯度，接着按照不同通信模式下进行更新模型参数。训练完成后，聚合收集到的准确率求出平均准确率并记录训练时间。

4月21日

在参数服务节点编程中我们使用了4种方式来更新参数。今天主要实现了同步SGD，同步SGD在每轮的mini-batch训练中，会等待所有的计算节点完成本地局部梯度计算，然后将每个工作节点上计算的本地局部随机梯度收集，平均得到全局梯度。同步SGD的收敛性等价于单机SGD的收敛性，但由于同步SGD要等待所有的计算节点完成梯度计算，因此同步SGD的训练效率会受到计算能力最差的节点的制约。

4月23日

今天完成了限制异步频率的部分同步。其更新公式和异步SGD更新公式)相同，只不过此时的延迟步数有取值限制。如果将延迟步数理解成异步频率，延迟步数=0，LAF-SGD退化成了同步SGD；延迟步数=∞或者延迟步数=计算节点数（在异步SGD模型参数更新中，随机梯度延迟的平均数是计算节点数），LAF-SGD则进化成了异步SGD。也就是说，LAF-SGD并没有解决甚至改善梯度延迟的问题，只是限制了梯度延迟的迭代次数，所以使用该算法优化的模型的准确率和训练时间是介于同步SGD和异步SGD之间的。

4月22日

今天完成了异步SGD。步SGD是节点完成一个mini-batch训练后直接更新模型参数，不需要等待其他节点，因此，异步SGD的训练效率要比同步SGD快。然而每一次更新，不同节点将会获取最新的模型参数，但是节点获取的时间是有延迟的，使用的梯度时间节点是落后于模型参数的时间节点的，所以得到的模型参数也可能不一致。因此更新模型参数的梯度是延迟的，会降低算法的稳定性的模型的准确度。

4月25日

今天基本上完成了所有的编程工作，进行调试。先在两台虚拟机上测试，参数节点程序必须先运行，之后再运行计算节点程序。由于之前各个模块都单独测试过，基本没出现什么毛病，最多出现类似变量名打错了的小错误。程序运行成功后，先测试mnist数据集，结果发现mnist数据集的准确率只能到达92%，和网上说的随便一个模型就能98%左右，于是就怀疑是不是学习考率等参数设置错误，进行了一天的调式，也没有找出原因。

4月24日

今天完成了补偿梯度延时的异步SGD（DC-ASGD）。DC-ASGD是对异步SGD中的梯度延迟进行补偿，改善因梯度延迟导致的模型准确率下降。其使用梯度展开泰勒公式的一阶近似，其中梯度的梯度又使用Hessian矩阵的对角矩阵做近似，在几乎没有增加计算量和存储代价的同时，提高了梯度的及时性，最终达到拥有异步SGD的训练时间的同时还能提高模型准确率。

4月26日

昨天在mnist训练模型的时候发现准确率老是提不上去,今天终于找到了一种原因，是数据集的没有预处理正则化，在mnist上期望值设置为0.1307,标准差设为0.3081,准确率推动阿了97%左右，此时我以为已经没什么错误了。但是在CIFAR10数据集上准确率只有30%,不管怎么修改模型，都无法大幅度提升准确率最后发现是检测部分的代码写错了。

4月27日

前些日子已经调试好程序了，从今天开始训练。今天主要是在mnist上训练模型。设置了4种计算机的分布式环境，是由节点计算能力是否均衡和节点通信条件是否均衡相组合的。由于mnist的CNN模型比较简单，每次的结果都以训练10次取平均值作为结果，每一种算法都训练，基本上一天就训练完了。

4月28日

今天开始在cifar10数据集上训练，初始设置的CNN模型比mnist复杂多了，训练一次花费1个多小时，准确率能够达到80%,为了提高准确率，使用了其他现成的高准确率模型，但是机器内存不足，无法使用，所以最终只使用了该模型。这个模型训练太慢，我一边训练开始准备写毕业论文。

4月29日

今天继续在cifar10上训练，同时整理资料写毕业论文。重新学习了word插图、插表、插入公式的技巧。今天勉强写了论文中的国内外现状和面临的挑战和趋势。

5月1日

这几天一直在cifar10上训练，毕业论文也完成了理论背景部分。本课题主要使用的parameter-server通信架构，使用了同步SGD，异步SGD，限制异步频率的部分同步SGD，DC-ASGD。

5月5日

到今天才完成了在cifar10数据集上的全部训练，同时也根据实验数据完成了论文中的实验过程和实验结果与分析部分，得出结论：无论计算节点计算能力是否均衡、通信条件是否均衡下，SSGD拥有最高的准确率，训练时间最长；ASGD准确率最低，训练时间最短；其改进版DC-ASGD准确率接近SSGD，训练时间却和ASGD保持一致；LAF-SGD的准确率和训练时间介于SSGD和ASGD之间，而且准确率变化不大，但是训练时间差异较大。