中山大学硕士学位论文

**面向分布式深度学习的梯度压缩机制的研究**

**Research on Piece Based Distributed Transaction Execution by Dynamic Chopping**

学位申请人： 匡迪

指导教师： 吴维刚 教授

专业名称： 软件工程

答辩委员会主席（签名）：

答辩委员会委员（签名）：

二零一八年五月八日

# 论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日 期：

# 学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

学位论文作者签名： 导师签名：

日期： 年 月 日 日期： 年 月

论文题目：面向分布式深度学习的梯度压缩机制的研究

专 业：软件工程

硕 士 生：匡迪

指导教师：吴维刚 教授

# 摘要

**关键词：**分布式事务，分片，可串行性，冲突分析，并发控制

Title: Research on Piece Based Distributed Transaction Execution by

Dynamic Chopping

Major: Computer Science and Technology

Name: Di Kuang

Supervisor: Prof. Weigang Wu

# Abstract

**Keywords:** distributed transaction; piece; serializability; conflict analysis; concurrency control

# 目录

[摘要 I](#_Toc514252392)

[Abstract III](#_Toc514252393)

[目录 V](#_Toc514252394)

[第 1 章 绪论 1](#_Toc514252395)

[1.1 研究背景 1](#_Toc514252396)

[1.2 研究意义 3](#_Toc514252397)

[1.3 研究内容与主要创新点 4](#_Toc514252398)

[1.4 论文结构安排 5](#_Toc514252399)

[第 2 章 相关工作综述 7](#_Toc514252400)

[2.1 分布式系统的数据访问 7](#_Toc514252401)

[2.2 分布式事务 11](#_Toc514252402)

[2.3 本章小结 15](#_Toc514252403)

[第 3 章 基于动态分片的分布式事务执行系统 17](#_Toc514252404)

[3.1 系统概述 17](#_Toc514252405)

[3.2 事务执行流程 21](#_Toc514252406)

[3.3 系统假设与拓展 26](#_Toc514252407)

[3.4 正确性 28](#_Toc514252408)

[3.5 本章小结 29](#_Toc514252409)

[第 4 章 事务片段的冲突分析和并发控制 31](#_Toc514252410)

[4.1 可串行性的分析与保障 31](#_Toc514252411)

[4.2 事务片段的冲突分析 36](#_Toc514252412)

[4.3 冲突片段的并发控制 44](#_Toc514252413)

[4.4 本章小结 51](#_Toc514252414)

[第 5 章 实验与结果分析 53](#_Toc514252415)

[5.1 实验环境 53](#_Toc514252416)

[5.2 实验设置 56](#_Toc514252417)

[5.3 冲突分析实验 57](#_Toc514252418)

[5.4 整体性能实验 61](#_Toc514252419)

[5.5 可扩展性实验 67](#_Toc514252420)

[5.6 本章小结 69](#_Toc514252421)

[第 6 章 总结和展望 71](#_Toc514252422)

[6.1 总结 71](#_Toc514252423)

[6.2 展望 71](#_Toc514252424)

[参考文献 73](#_Toc514252425)

[硕士学位期间与硕士学位论文相关的研究成果清单 79](#_Toc514252426)

[致谢 81](#_Toc514252427)

# 绪论

## 研究背景

## 研究意义

## 论文结构安排

# 相关工作综述

## 分布式系统的数据访问

**2.1.1 性能优先**

**2.1.2 一致性优先**

**2.1.3 灵活性优先**

## 分布式事务

**2.2.1 事务的基础特性**

## 本章小结

# 前期工作和问题描述

从前文的描述中，我们了解到通信常常成为分布式深度学习的瓶颈。为了验证这一结论，我们进行了一些前期工作。在前期工作中，我们针对不同的深度学习框架进行了一系列的实验，并且用实验结果更好的说明了通信对分布式深度学习的影响。同时根据当前对梯度压缩领域的研究，我们发现如何在对梯度进行压缩的同时保持精度和加快收敛速度是一个非常具有挑战性的课题。

本章前半部分介绍了前期工作中的实验以及实验结果分析，后半部分详细描述了本文设计的机制要解决的问题，为后面介绍我们提出的机制奠定了基础。

## 前期工作

从第一章和第二章的很多文献中，我们知道了通信是影响分布式深度学习的主要因素。为了更好的了解在工作节点增加的时候，通信对分布式深度学习的影响程度，我们针对不同的深度学习框架进行了一定规模的测评实验，其中包括英特尔公司开发的BigDL框架和由C++以及CUDA实现的轻量级工具Caffe。但是由于原生的Caffe并不支持分布式训练，所以我们选择了Caffe的分布式扩展Caffe-oMPI。

本次测评实验主要是在跨节点分布式并行场景下对不同的深度学习框架进行训练。在实验中，我们使用了3种不同类型的神经网络模型（AutoEncoder自动编码器，LeNet和ResNet）和2个数据集（MNIST和Cifar-10）。AutoEncoder、LeNet和ResNet是三个比较经典的网络模型，而且它们的网络层数依次增加，分别为3、7和101层，我们认为它们可以较好的体现出不同深度的网络模型的情况。同时，MNIST和Cifar-10数据集是业界公认的范例应用，很多的工作都是使用它们来作为评判性能的数据集。所以我们决定用这三个模型和两个数据集来搭配进行实验。另外，在本次的测评实验中我们使用的分布式深度学习工具BigDL和Caffe-oMPI均采用了第二章中所描述的数据并行的方法。

**3.1.1 Caffe-oMPI实验结果分析**

对Caffe-oMPI框架进行实验时，我们设计了3个评价指标，分别是准确率，加速比和通信时间占比。在实验中，我们细粒度地收集到很多和评价指标相关的数据，并且将它们绘制成了图表。目前，我们主要是研究通信对分布式深度学习的影响，所以下面主要介绍通信时间占比的实验结果分析。

通信时间占比主要是在一次迭代训练中通信时间占整个训练时间的比例。图3-1是我们所绘制的通信时间占比的实验柱状图。

从图3-1(a)中可以看出，在模型AutoEncoder的训练过程中，随着工作节点个数的增加，通信时间占总训练时间的比例并没有呈现出规律的增加，但是总的来说还是增加的。



(a) AutoEncoder (b) LeNet (c) ResNet

图3-1 Caffe-oMPI通信时间占比结果图

从图3-1(b)和图3-1(c)中，我们了解到在模型LeNet和ResNet的训练过程中，工作节点数增加时，通信时间的占比会呈现近似指数的增长，造成这些现象的原因主要是这两个模型在训练的时候，模型的计算时间会大于模型的通信时间，但是当工作节点的个数增加的时候，计算时间减少的量会远大于通信时间增加的量，所以通信时间占比是呈现近似指数增长的。

通过对图3-1的分析，我们知道了在Caffe-oMPI框架中对网络层数不同的神经网络模型进行大规模的并行实验时，工作节点个数的增加会导致通信时间在整个训练时间中的比例显著上升，这样会导致通信时间的增加远高于计算时间的减少。

**3.1.2 BigDL实验结果分析**

在对BigDL深度学习框架进行实验时，我们也设计了很多的评价指标，其中包括平均单次迭代时间、模型加速比、平均通信时间、模型准确度等等。同时，我们根据这些评价指标收集到了很多数据。在这里，主要是研究通信对分布式深度学习的影响，所以主要介绍平均通信时间和通信时间占比的实验结果分析。

平均通信时间是指进行一次迭代训练，各个工作节点之间进行参数传输所需要的平均时间。图3-2是根据实验数据所绘制的平均通信时间结果图。

数据并行度和平均通信时间数据并行度和平均通信时间

(a) AutoEncoder (b) LeNet (c) ResNet

图3-2 BigDL平均通信时间结果图

从图3-2中可以很明显的看出，在3个模型的训练中，平均通信时间都是随着节点数的增加有着非常明显的增加。在模型AutoEncoder和ResNet的训练过程中，当工作节点从2个增加到32个的时候，我们的平均通信时间从45ms增加到了350ms。在模型LeNet的训练过程中，当工作节点从2个增加到16个时，我们的平均通信时间也从25ms增长到175ms左右了。从三个模型的训练的平均通信时间结果图中，可以得出如下结论：当增加工作节点的个数时，我们的平均通信时间的增长速度已经接近指数增长。那么我们在对不同的网络层数的网络模型进行训练的时候，增加工作节点的个数，这些网络模型的平均通信时间都会呈指数的增长趋势。

图3-3是通信时间占比结果图。



(a) AutoEncoder (b) LeNet (c) ResNet

图3-2 BigDL通信时间占比结果图

从图3-2(a)和图3-2(b)中可以看出，在模型AutoEncoder和LeNet的训练过程中，节点数增加时通信时间占比会缓慢增加。

从图3-2(c)中可以看出，在ResNet模型训练过程中，节点数增加，通信时间占比会呈指数增长。造成三个模型通信时间占比随节点数增加出现不同的增长趋势的主要原因是三个模型的规模不同。对于ResNet模型，网络层数很深，训练过程中计算时间会远大于通信时间，那么在节点数增加时，计算时间减少的量会远大于通信时间增加的量，就造成了通信时间占比的指数增长。

总的来说，在BigDL框架下，当节点数增加时，三个模型的通信时间和通信时间占比都会显著增加。

从对Caffe-oMPI和BigDL框架的测评结果和分析情况来看，无论模型结构是否复杂，节点数增加的时候，通信时间和通信时间占比会显著增加。同时，在一些模型中，通信时间的增加远远大于计算时间的减少，这样通信时间就成为了分布式深度学习的一个瓶颈，会导致泛化能力变差。

## 问题描述

# 基于All-Reduce 的梯度压缩机制

根据本文第三章的描述，梯度压缩是解决通信瓶颈问题的一种重要的方法。在前期介绍的梯度压缩方法中，怎样减少通信的数据量，以及怎样加快模型的收敛速度和保持模型的收敛精度，是需要认真思考的问题。本章针对上面的这些问题，设计了基于All-Reduce 的梯度压缩机制，其中包括基于熵的自适应梯度稀疏化算法、动态学习率调节算法以及修正动量的SGD 算法。

## 梯度压缩机制概述

在分布式深度学习中，大多数采用数据并行的方法来对训练任务进行加速。本文第二章中说明了数据并行是将多个模型的实例部署在很多个不同的工作节点之上，相互间独立的进行训练，最后对各个工作节点计算得到的梯度信息汇总来完成参数更新。在参数更新的过程中，各个工作节点之间发送和接收数据的时间以及相互之间等待的时间是分布式深度学习中大部分的通信时间。通过设计基于All-Reduce 的梯度压缩机制，可以在数据并行和同步通信下极大的减少通信的数据量，同时加快收敛性和确保收敛精度。本节将对基于All-Reduce 的梯度压缩机制进行一般性阐述，表明本机制的创新点。

本文提出的基于All-Reduce 的梯度压缩机制，在All-Reduce 通信拓扑结构上实现了基于熵的自适应梯度稀疏化算法、动态学习率调节算法和动量修正SGD 算法。在通信频率上，我们改变了之前的固定阈值或百分比的模式，利用网络中每一层梯度的熵值来动态确定阈值，设计出了一种自适应梯度稀疏化算法。但是，由于梯度稀疏化减少通信的数据量，这样会对收敛速度和收敛精度造成影响。所以，在收敛速度上，我们设计了一种基于当前训练梯度的最大值来自动调节学习率的算法，能很好的弥补通信量减少带来的影响。另外，我们在动量SGD 的基础上对每一次迭代的动量进行修正，减少动量累积和通信量减少对收敛精度造成的影响，以此来提高我们的收敛精度。

在通信拓扑结构方面，我们选择了All-Reduce 模式。因为我们使用同步并行的参数更新模式，而All-Reduce 模式不仅简单方便，而且对使用同步并行的并行训练非常有利。同时，在All-Reduce 拓扑结构中，每一个工作节点需要和其他所有的工作节点进行通信，以此来同步梯度。All-Reduce 拓扑结构以及增加工作节点后的结构如下：

从图4-1中可以看出当工作节点大量增加的时候，计算时间减少，但是通信时间大量的增加。在图4-2中由于我们对工作节点之间所传输的梯度进行了压缩，减少了通信的数据量，通信时间明显减少了，同时我们也可以了解到梯度压缩在All-Reduce 通信拓扑结构上是可行的。

接下来我们将详细介绍我们的自适应梯度稀疏化算法、动态学习率调节算法和动量修正SGD 算法。

## 基于熵的自适应梯度稀疏化算法

根据第二章对梯度压缩领域现状的描述，在进行梯度稀疏化时，我们需要将对本次参数更新不重要的梯度进行延迟发送，以达到减少通信数据量的目的。

目前，在梯度稀疏化方面，主流的方法有两种，一种是固定阈值，在训练开始前设置好阈值，然后每个工作节点都会将大于阈值的梯度元素发送给其他工作节点，而小于阈值的梯度元素则会在本地进行累积，直到大于阈值之后再进行发送。另一种是固定发送梯度的比例，在进行训练前设置好需要发送梯度的比例，比如1% ，然后在每一次迭代训练中每个工作节点都会将前1%的梯度元素发送给其他工作节点，而剩下的梯度元素则累积起来在之后的训练中再发送。在这些方法中，我们发现只能根据经验来选择一个固定的阈值或者比例，而一个固定的阈值或比例可能会导致错误反馈积累的退化情况，导致传输的数据量很大，这样就不能达到减少通信数据量的目的了。针对这个问题，我们提出了一种基于熵的自适应梯度稀疏化算法。

本节将从设计思路和算法流程两个方面介绍我们提出的基于熵的自适应梯度稀疏化算法。其中，设计思路主要描述了算法的主要思想，算法流程则主要介绍算法的实现过程。

**4.1.1 设计思路**

本节的算法主要关注怎样动态选择阈值，让它比固定阈值或比例更好的适应模型的训练过程。在训练模型的过程中，每一次迭代中每个工作节点都需要将所有的梯度全部传给其他的工作节点。当增加工作节点个数时，每一次迭代传递梯度的时间会大大增加，通信时间也会大大增加。同时，在每次迭代中所产生的梯度元素的大小并不是一直不变的。梯度元素会随着训练时间和网络层数的不同发生动态的变化，其中所包含的信息也会不同，对后面训练过程产生不同的影响。那么，我们主要着眼于梯度元素及其中信息的变化设计了基于熵的自适应梯度稀疏化算法。

信息是指人类社会传播的一切内容，指音讯、消息、通信系统传输和处理的对象。在每一次的迭代训练过程中，梯度元素被在各个节点之间传输，和数字、文字和自然语言一样，是信息的载体。但是，信息是一个有些抽象的概念，怎样才能度量信息的多少。在1928年，信息定量化的初步设想被提出来：信息量是信息多少的量度。如果一个事件发生的概率越低，则这个事件包含的信息量就越大，公式如下：

其中表示事件*x* 发生的概率。信息定量化只是对信息的初步度量。直到1948年，香农指出“信息是用来消除随机不确定性的东西”，并提出了信息熵的概念，才解决了信息的度量问题。

在信息论中，信息熵是接收的每一条消息中所包含的信息的平均值，用于描述平均而言事件的信息量的大小，单位是比特，符号是。在数学上，信息熵是事件中包含信息量的期望值。对于任意一个事件，它的熵的定义如下：

其中表示事件，表示事件中每种情况可能出现的概率。从公式中可以看出，事件的不确定性越大，计算出的信息熵也就越大，相应的信息量也就越大。

在神经网络的训练中，主要包括前向传播和反向传播。前向传播可以简单的描述为：数据从输入层经过隐藏层，最后由输出层产生结果的过程。在后向传播中，需要将前向传播产生的结果与正确的结果比较，计算得误差值，再根据误差值计算出相应的梯度，将梯度从输出层向前一层一层的传递，并进行相应权重的修订。由于神经网络中每一层的网络结构都是不同的，而且梯度是一层一层向前传递的，所以神经网络中的每一层产生的梯度的尺寸和大小都是不同的。根据上面对信息熵的描述，神经网络中每一层的梯度的信息熵的值也是不同的，而信息熵在一定程度上表示了信息量的大小。于是，我们将每一层梯度的信息量作为一个重要的标准，信息量越大代表所包含的信息越多，那一层梯度的重要性也就越高，所需要发送的梯度也就越多。

另外，我们设置了一个固定的比例 ，表示在处理每个minibatch 后要发送的梯度的比例。由于计算出的信息熵的单位是比特，我们将取值为的形式，比如，，等，然后通过实验来选择出表现最好的值。在我们得到了每一层梯度的信息熵和固定比例，就可以计算出要发送梯度的百分比，然后在每一层的梯度中利用选择算法选出我们需要的阈值。在计算出阈值之后，我们仅仅只需要发送绝对值大于阈值的梯度元素，大大减少了工作节点之间通信的数据量。

在算法中，虽然我们只是简单的设置了固定比例，但是对每一层的梯度都使用同样的固定比例，这样可以充分的利用每一层梯度的熵在不同训练时间和网络层中的变化动态的确定需要发送梯度的阈值。这样选择出来的阈值可以更好的适应神经网络中不同时间的训练状况，能更合理的发送出最重要的梯度值。

**4.1.2 算法流程**

在设计思路中，我们详细介绍了设计整个算法的设计思想，下面我们将详细说明算法的执行流程。

在算法中，首先需要根据每一层的梯度来计算出相应的信息熵，我们将这个计算过程分为3步。第一步是将每一层所有的梯度分为*N*个不同的区间，第二步是统计出现在每个区间中梯度元素的个数和利用梯度元素的个数计算每个区间的概率，最后一步是根据信息熵的计算公式和每个区间的概率来计算这一层梯度的熵值。对于第一步中*N* 值，我们同样是根据实验来选择表现最好的*N* 值。整个算法的具体步骤如下：

1. 输入神经网络中每一个网络层的梯度（其中*l* 表示层数）以及和*N* 的初始值。
2. 根据每一个网络层的梯度计算出这一层中的最大梯度和最小梯度。
3. 利用最大梯度和最小梯度求出*N* 个区间中每一个区间最大值和最小值的间隔。
4. 统计在每一个区间中的梯度元素的个数。
5. 根据每一个区间中梯度的个数和这一层梯度的总数计算出每一个区间的概率。
6. 利用熵的计算公式和每个区间的概率计算出这一层梯度的熵，并且计算出这一层梯度需要发送的百分比。
7. 通过快速选择算法在这一层梯度中选择出我们需要的阈值，并且将绝对值大于阈值的梯度发送给其他节点。

整个算法的伪代码如算法所示。

**Algorithm 1** Entropy-based Threshold Selection Algorithm

**1: Input:** Gradients , *l* = *1..L* for each of the L layers in the net, *K*, *N*

**2: for** layer *l = 1..L* **do**

**3:** Find

**4:** Find

**5:** Compute the interval: *H =*

**6: for** bin*i* = *1..N* **do**

**7:** num = 0

**8: for** *k = 1,2,……* **do**

**9: if**  >= and < **then**

**10:**  num = num + 1

**end if**

**end for**

**7:** Compute Number *h(i)* = num

**8: end for**

**9: for** *k = 1..N* **do**

**10:** Compute Probability *p(i) =*

**11: end for**

**12:** Entropy *H* = -

**13:**  The ratio *t* =

**14:** Threshold **= Quickselect(**, *t***)**

**15: end for**

**16: Output: ,** *l = 1..L*

## 动态学习率调节算法

在上一节中，我们详细的介绍了基于熵的自适应梯度稀疏化算法。它会减少工作节点之间通信的数据量，但是这会对模型训练过程中的收敛速度造成影响。为了解决收敛速度的问题，我们需要给学习率设计一个良好的调节算法。因为学习率是我们训练模型的过程中的一个非常重要的参数，一个好的学习率调节算法可以加快模型的收敛速度。在平常的深度学习模型训练中，通常会采用常数或者幂指数作为学习率，形式较固定，并且依赖自己的经验，不能很好的适应模型的运行。所以我们提出了一种自动学习率调节算法，在模型训练过程中调节学习率提高收敛速度。

本节将从设计思路和算法流程两个方面介绍我们提出的自动学习率调节算法。设计思路主要描述算法的思想，算法流程主要介绍算法的实现过程。

**4.3.1 设计思路**

在本算法中，我们主要关注怎么根据模型训练中的运行状态来自动的调节学习率。首先，我们需要制定一个测试指标来确定调节学习率的时机。在深度学习模型的训练中，数据集通常被分为训练集、验证集和测试集。训练集用来拟合模型或者调试神经网络。验证集是主要用来调整模型的参数从而选择最优的模型。测试集则是检验最终选择模型的性能，以及验证模型的泛化性能。在训练的过程中，每个epoch 结束之后会对验证集进行训练得到相应的验证集损失值，而且验证集损失值会随着训练的进行不断的发生变化，所以我们将验证集损失值作为测试指标。

另外，我们了解到学习率的相关规律。当学习率太小时，模型的收敛速度太慢，容易陷进局部最优解。当学习率太大的时候，收敛速度很快，容易发生损失值爆炸和振荡，一般使用在训练开始时。所以我们在训练的开始阶段，选择较大的学习率初始值，来避免局部最优解，保证模型的快速收敛。在之后的训练中对学习率进行调节时，我们考虑到当前梯度能很好的反应模型的运行状况，所以我们在上一个epoch 学习率的基础上利用当前梯度的最大值来调节学习率。具体的公式如下：

其中表示上一个epoch 中的学习率；是指上一个epoch中的验证集损失值；则是梯度的最大值。

在训练中，每个epoch 训练结束时得到相应的验证集损失值，并和上一个epoch 训练结束后的验证集损失值进行比较。如果现在的损失值小于上一次的损失值，说明模型收敛平稳，学习率的下降速度应该平缓，分母就使用了平方根来控制变化的速度。如果现在的损失值大于上一次的损失值，则说明模型没有很好的收敛，需要将学习率加速下降，所以相对于上面式子的分母进行了平方。

**4.3.2 算法流程**

## 动量修正SGD 算法

**4.3.2 示例与分析**

**4.3.3 正确性**

## 本章小结

# 实验与结果分析

## ·实验环境

**5.1.1 系统实现**

# 总结和展望

## 总结

## 展望

# 参考文献

致谢

首先，我要由衷地感谢导师吴维刚教授对本人的精心指导，他将我带入了研究的大门，并为我指明了研究的方向。吴维刚教授对待科研工作一丝不苟，他敏锐的洞察力和开阔的视野在潜移默化中感染着我，也督促着我更好地完成科研工作。在完成硕士学位的这几年，吴老师在每周例会和不定期讨论中给予的指导都让我获益良多，同时为我创造了许多历练机会。在完成学位论文的过程中，从相关研究工作的调研，到具体研究内容的提出和改进，以及实验的实施细节，吴老师都提供了热心的帮助。这些都使我的科研能力较入学前有了很大的提升。

另外，我要感谢同研究方向的吴立慧师姐、周萌和容英耀师弟。在完成本人研究工作的过程中，与几位进行的深度讨论给我提供了重要的帮助，使得一些难点问题得以解决，让我的研究能够进一步地开展，非常感谢几位的帮助。同时，我还要感谢实验室的所有成员，他们对我的科研工作和生活都给予了很多建议和帮助。

此外，感谢所有对本人研究内容提供评审意见的专家，他们的专业意见使本人的研究工作得以不断完善。

最后，特别的感谢要献给我的家人，他们的关爱、支持和鼓励给了我奋斗的动力与勇气。