西北工業大學

本科毕业设计（论文）

题 目 基于区块链智能合约的加密数据冗余删除技术研究

专业名称 计算机科学与技术专业

学生姓名 张博

指导教师 崔禾磊

完成时间 2023 年 6 月

摘　要

西北工业大学（简称西工大）坐落于陕西西安，是我国唯一一所以同时发展航空、航天、航海（三航）工程教育和科学研究为特色的多科性、研究型、开放式大学，现隶属于工业和信息化部。新中国成立以来，西工大一直是国家重点建设的高校，1960 年被国务院确定为全国重点大学，“七五”、“八五”均被国务院列为重点建设的全国 15 所大学之一，是全国首批设立研究生院的 22 所高校之一，1995 年首批进入“211 工程”，2001 年进入“985 工程”，是“卓越大学联盟”成员高校，先后获得“全国文明单位”、“全国创先争优先进基层党组织”和“全国毕业生就业典型高校”等荣誉称号和表彰奖励。学校秉承“公诚勇毅”校训，弘扬“三实一新”（基础扎实、工作踏实、作风朴实、开拓创新）校风，扎根西部、献身国防，历史上书写了新中国多个“第一”，今天在创建一流大学和一流学科上续写新的辉煌。

西北工业大学（简称西工大）坐落于陕西西安，是我国唯一一所以同时发展航空、航天、航海（三航）工程教育和科学研究为特色的多科性、研究型、开放式大学，现隶属于工业和信息化部。新中国成立以来，西工大一直是国家重点建设的高校，1960 年被国务院确定为全国重点大学，“七五”、“八五”均被国务院列为重点建设的全国 15 所大学之一，是全国首批设立研究生院的 22 所高校之一，1995 年首批进入“211 工程”，2001 年进入“985 工程”，是“卓越大学联盟”成员高校，先后获得“全国文明单位”、“全国创先争优先进基层党组织”和“全国毕业生就业典型高校”等荣誉称号和表彰奖励。

综上, 本文主要做的工作有

1. 分析 A
2. 分析 B
3. 提出 C

关键词:　学位论文，模板, LaTeX

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

To sum up, this paper works on those

1. Balabala 1
2. Balabala 12
3. Balabala 123
4. Balabala 1234
5. Balabala 12345

KEY WORDS: thesis, template, LaTeX

目 录

第一章　绪论

1.1 研究背景及意义

(1) 选题背景

预训练模型是已经用广泛的样本训练过的模型。它已经在一个大型数据集上针对特定任务进行了训练。这个数据集可以是多种形式的，例如图像、文本或音频等。预训练模型的训练过程会产生一种通用的特征表示, 这些特征可以被用来执行类似任务的新数据。从头开始训练一个深度学习模型可能需要花费数周或数月的时间, 特别是在缺乏大量数据集的情况下（如 BERT [1], GPT-3 [2]）。在像 ImageNet 这样的大型数据集上训练神经网络，该数据集包含 1000 个类别的 1400 多万张图片，在这样的数据集上重新训练这样的模型是一种很大的开销。使用预训练模型可以作为模型的起点, 这意味着，当为一个新任务微调预训练的模型时，我们不必从随机权重开始。而是可以使用预训练的权重作为初始化，然后只训练后面的特定于新任务的层。这需要更少的数据和训练时间。这可以节省大量的时间和精力。此外预训练模型有其他方面的优势。它们可以将知识从一般领域转移到特定领域。神经网络的浅层倾向于学习一般的特征，如边缘和形状，而后深则学习更具体的特征。因此，在一般图像上训练的预训练模型可以提供一般的低层次特征，然后你只需要为你的具体任务重新训练后面的层。这就是所谓的迁移学习。近年来, 学术界和工业界都对开发精度更高, 参数量更大的预训练模型感兴趣, 因为采用较大的模型会带来更高的准确性。如图1-1所示, 近年机器学习模型参数量近几年呈现爆炸式增长。远远超出了常用 GPU 的显存限制（通常只有 40-80GB，如 Nvidia A100），这也对我们如何快速，高效的训练模型提出了全新的挑战。

研究人员表明，较大的模型会带来更高的准确性。从过去几年深度神经网络 (DNN) 驱动的机器学习技术的快速发展来看，研究者们发现加入更多 DNN 模型参数是最直接但不太复杂的方法之一提高 ML 算法的性能 [3]。然而，DNN 模型容量通常受到计算资源和能源成本的限制 [4]。根本原因是 DNN 的密集架构，其中计算成本通常与参数数量成线性比例。为了解决这种问题，混合专家 (MoE) [5] 在 DNN 中被广泛使用，它通过使用多个并行子模型 (混合专家) 引入了稀疏架构，其中每个输入经过门网络，动态选择并转发给少数专

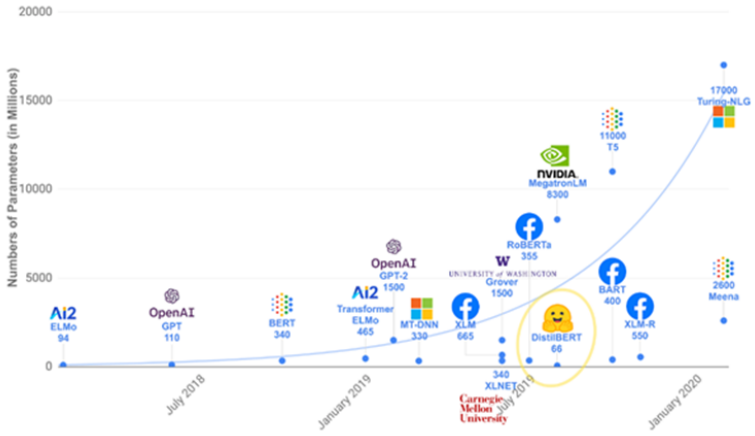

图 1-1　深度学习模型参数显著增长

家处理。专家混合似乎有望将模型扩展到极端尺寸。如图1-2 所示，与直接将小模型缩放为大密集模型不同，MoE[5] 模型由许多小模型组成，即专家。训练样本被送入不同的专家，由轻量级可训练门网络动态选择。在 MoE 中，由于稀疏激活专家，节省了大量额外的计算量，与传统的密集型 DNN 相比，可以显着增加同一时间段内训练的样本数，提高模型精度。MoE 技术是如今将 DNN 扩展到万亿参数的流行方法之一。

MoE 混合专家系统是一种稀疏模型，因此其训练过程不同于传统的密集型 DNN 模型。主要有以下三点挑战：

- **动态激活特性：** MoE 模型的稀疏激活特性使得它在 GPU 集群分布式训练时与现有的静态并行策略不匹配。因为 MoE 模型的每个样本只会被激活一个专家（也就是一个子模型），而其他的专家则不会被激活。这导致静态并行策略无法充分利用计算资源，因为只有部分计算节点被激活，而其他节点则处于空闲状态。

- **额外通信开销：** MoE 模型引入了 GPU 集群节点间额外的 All-to-All 通信，这种通信方式需要在所有计算节点之间进行数据传输和同步，由于 All-to-All 通信是一种同步通信方式，因此它会严重影响训练速度和效率。但是这种通信方式在 MoE 模型中是必需的，因为它需要将每个计算节点计算得到的结果进行汇总和组合，以得到最终的预测结果。

- **节点负载不均衡：** 由于 MoE 模型中的 Gating 在训练过程中不断变化，因此数据可能被分配到不同的专家上，导致负载分配不平衡的问题。如

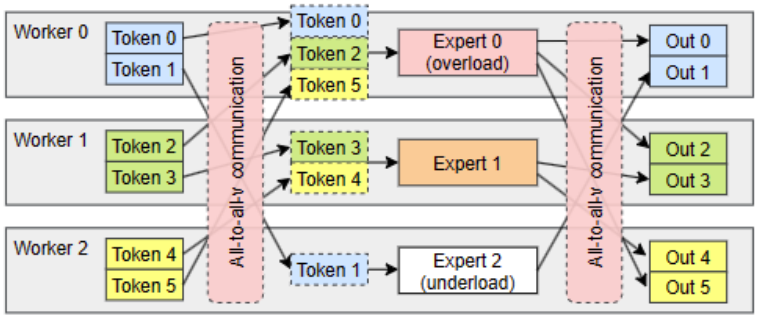

图 1-2　分布式 MoE 模型训练过程

果某些专家的负载过高，而其他专家的负载过低，就会导致训练时间的延长和模型性能的下降。因此，需要根据每个专家的激活情况和计算负载进行动态的数据分配和负载均衡，以确保每个专家的计算负载均衡，并最大限度地利用计算资源。

现有的分布式训练框架[6, 7]对于其稀疏性的结构没有很好的支持，因此本次毕业设置拟设计一种更加高效的 MoE 训练框架，加速其分布式训练的过程, 从而降低训练大规模的 MoE 模型架构的成本。

(2) 选题依据

随着人工智能技术的不断发展，MoE 模型在各个领域都具有广泛的应用前景。它可以将多个不同的模型集成起来，以提高模型的性能和泛化能力。目前 MoE 模型在语音识别、自然语言处理、计算机视觉等领域都取得了很好的效果。因此，研究 MoE 模型的分布式训练和优化策略，可以进一步提高模型的训练效率和性能，适应更复杂和庞大的深度学习任务和数据集的需求。不仅能够为深度学习领域的研究提供新的思路和方法，还可以为各个领域的应用提供更加高效和精确的解决方案。

1.2 研究内容

研究目标。

本课题针对 MoE 模型带来以上挑战，拟分析现有的 MoE 模型分布式训练系统存在的缺陷，并设计一套全新的 MoE 模型训练系统。如图1-3所示，我们提出了一种全新的 MoE 训练系统，通过在算法层面（gating policy）和系统层面（Expert placement scheduler）提出创新的训练解决方案，旨在克服 MoE 模型训练过程中的瓶颈，并更好地适应复杂而庞大的深度学习任务需求。

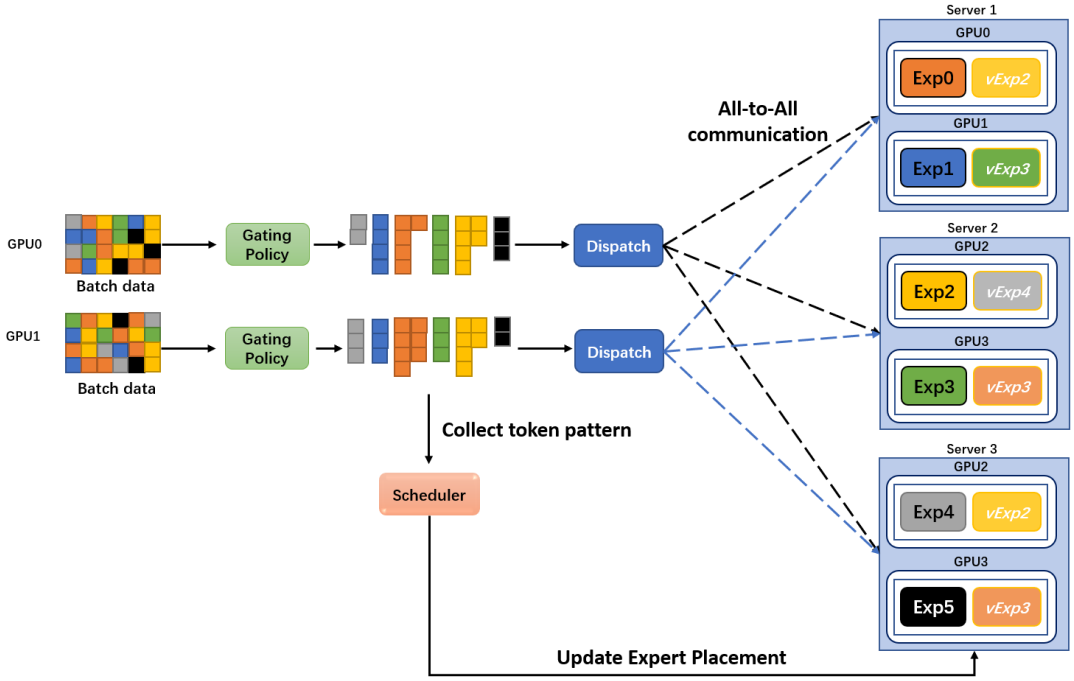

图 1-3　传统的 Top1 和 Top2 Gating 策略

在算法层面，我们针对 MoE 模型的关键部分，即 gating 策略，进行了改进。我们通过动态调整数据分派策略，使其获得较快的收敛速度，同时也有合适的全局通信开销。

同时，在系统层面，我们提出了一种专家分配调度器（Expert placement scheduler），它能够根据网络拓扑以及任务负载，找到最合适的专家并行的策略。这样的设计考虑任务的特性、数据的分布以及专家的能力，我们能够动态地优化专家的分配策略，使得每个专家都能够发挥最大的潜力，并在整个系统中平衡负载和资源利用率。

这种全新的训练解决方案使得 MoE 模型能够更好地应对复杂和庞大的深度学习任务需求。通过优化算法和系统设计，我们能够充分发挥 MoE 模型的潜力，提高模型的准确性和泛化能力，为解决现实世界中的复杂问题提供了有力的工具。我们的研究对于推动 MoE 技术的发展和应用具有重要意义，并在深度学习模型训练的领域取得了显著的突破。

1.2.1 基于动态路由的数据分派策略

如图1-4所示在传统的 MoE 模型中，采用固定的 Top1 和 Top2 Gating 策略，即将数据发送到分数最高（次高）的 expert 进行处理。然而，采用 Top1

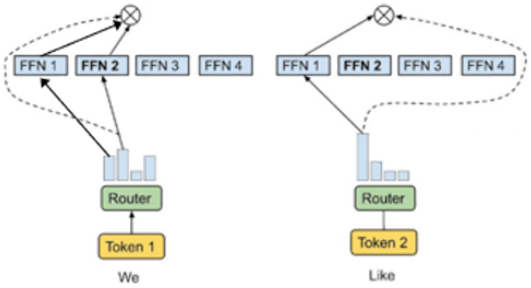

图 1-4　传统的 Top1 和 Top2 Gating 策略

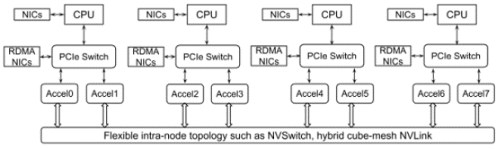

图 1-5　GPU 集群内网络拓扑连接图

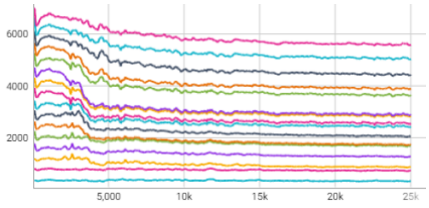

图 1-6　8 层 128 专家的 Transformer-XL[8] 的 MoE 模型, 第 1 层 16 个专家负载变化情况

Gating 时，由于只选择一个最高分数的 expert，可能会错过其他有价值的信息，导致模型收敛速度较慢；而采用 Top2 Gating 时，虽然可以选择两个最高分数的 expert，但每轮训练时间较长，因为需要进行两次 All-to-All 通信。因此我们能否将 Top1 和 Top2 Gating 策略结合起来，以一种动态的方式选择合适的数据分派策略, 从而实现较快的收敛速度和较短的训练时间。一种简单的方法是，将每个数据按照 Top1 和 Top2 的分数进行排序，然后将它们分别分配给 Top1 和 Top2 的 expert 进行处理。这种方法可以利用 Top1 和 Top2 的优点，避免错过其他有价值的信息，同时减少通信开销，提高训练效率。

1.2.2 基于网络拓扑的自动负载均衡策略

虽然基于 MoE 的算法开辟了一个巨大扩展模型参数量机会，但它也对训练系统带来了新的挑战，而这些挑战在之前的密集型 DNN 训练算法和系统中从未见过。根本原因是动态专家选择和灵活的 MoE 结构。具体来说，每个 MoE 层由一定数量的并行专家组成，这些专家分布在加速器（本工作中的 GPU）上，其中每个 GPU 根据智能门函数将每个输入数据分配给几个最适合的专家并取回相应的输出以将它们组合起来。这意味着每个专家的工作量基本上是不确定的，取决于输入数据和门函数。在图1-6中，我们使用了一个 8

层的 Transformer-xl MoE 模型，分析了第 1 层 16 个专家负载变化情况。

我们发现，在 MoE 模型中，不同专家的负载在每次迭代中都会发生变化。这种现象是由于 MoE 的稀疏性动态数据分配所导致的，导致每个专家获取的数据不均匀，从而产生了负载不均衡的问题。这会导致一些节点或 expert 的负载过重，从而影响整个模型的训练速度和效果。因此，需要采取适当的负载均衡策略来缓解这个问题。此外在 MoE 模型中所有专家都需要从其他 GPU 上那里获得输入，这引入了 GPU 集群所有节点间额外的 All-to-All 通信，且 All-to-All 通信与后续计算是完全同步关系，并行较差。而 GPU 集群内部节点之间的网络带宽并不相同，节点通信效率存在差异。因而 All-to-All 通信也成为了大规模 MoE 训练中最耗时的操作之一。它通常实现为具有可变消息大小的同步 All-to-All 操作。考虑到动态特性的数据分派会导致计算和通信的严重不平衡，这样的方法会导致严重的开销。

因此，为了解决负载不均衡问题，需要采取一些适当的负载均衡策略，我们需要结合 GPU 集群内部节点之间的网络带宽不相同的情况，选择最佳的负载均衡策略，以提高整个模型的训练效率和性能。

1.3 国内外研究现状

1.3.1 MoE 训练系统

随着 MoE 训练范式的普及，许多科研机构和企业都开源了 MoE 训练框架和系统。DeepSpeed-MoE 利用多种分布式并行方法结合 MoE 并行性，包括数据并行、张量切片 [9]、Zero 内存优化 [10] 来训练更大的模型。至于 MoE 的推理，DeepSpeed[11] 设计了一种名为 PR-MoE 的新型稀疏激活模型和模型压缩技术来减小 MoE 模型的大小，以及一种有效的通信方法来优化延迟。FastMoE[12] 是一个分布式 MoE 训练系统，它提供了一个分层接口和简单的机构，说明如何基于数据并行性和张量切片并行性使用 Megatron-LM[9] 和 Transformer-XL[8]。与 DeepSpeed 的实施不同，FastMoE 使用复杂的优化方法来减少网络流量。Fairseq-MoE[13] 是一个序列建模框架，用于训练用于摘要、翻译和语言建模的自定义模型。而 Tutel[14] 在通信和计算方面进一步优化了 Fairseq 系统，其性能提升了约 40%。Tutel 中的优化已集成到 DeepSpeed 中，以促进 MoE 模型训练。

1.3.2 MoE 数据分派策略

MoE 的核心问题之一是如何设计 gating 策略，即如何根据输入分配不同的专家网络。不同的 gating 策略会影响模型的性能、稀疏性、均衡性和公平性。

- Softmax gating[12]：这是最简单的一种 gating 策略，它使用一个 softmax 层来为每个输入分配一个概率分布，表示每个专家网络的权重。这种方法可以看作是多个专家网络合作来产生输出，但是也会导致所有的专家网络都被激活，从而增加计算量和内存消耗。

- Top-k gating[14]：这种 gating 策略只选择概率最高的 k 个专家网络来处理输入，其他的专家网络则被忽略。这种方法可以实现稀疏性，即只有少数的专家网络被激活，从而节省计算量和内存消耗。但是这种方法也会带来一些问题，比如如何确定 k 的值，以及如何保证每个专家网络都能被充分利用。

- Noisy softmax gating[15]：这种 gating 策略在 softmax gating 的基础上增加了一个可学习的噪声权重，用来提高不同专家网络的 gating 均衡性。这种方法可以防止某些专家网络被过度使用或者被忽略，从而提高模型的公平性和泛化能力。

- Hierarchical softmax gating[15]：这种 gating 策略将多个专家网络组织成一个层次结构，每一层都有一个 softmax gating 来决定下一层的激活。这种方法可以减少 softmax gating 的计算复杂度，从而提高模型的效率。

- Hash layer[16]：这种 gating 策略使用哈希函数来为每个输入分配一个或多个专家网络，而不需要学习任何参数或者使用额外的损失函数。这种方法可以实现极高的稀疏性和效率，同时保持或者提升模型的性能。

- Topology-aware gating[17]: TA-MoE 中提出了一种拓扑感知的路由策略，它能够根据网络拓扑的变化动态地调整 MoE 的数据分派的调度策略。通过基于通信建模的方法，TA-MoE 将调度问题抽象为一个优化目标，并得到了适用于不同拓扑结构的近似调度模式。他们设计了一种拓扑感知辅助损失函数，它可以自适应地根据底层拓扑调整数据分派策略，而不会牺牲模型的准确性。

1.3.3 分布式深度学习训练系统概述

分布式深度学习训练是一种将一个任务划分为较小的子任务并在多个处理器或设备上同时运行的技术 [18]。这种方法可以加快任务的执行速度，特别是当子任务可以在不同的处理器或设备上并行执行时。在分布式深度学习训练中，我们通过将训练数据和模型参数分布在多个处理器或设备上，然后在多个 GPU 上同时执行子任务来提高训练速度。这使我们能够突破单个 GPU 的内存限制，训练出比单个处理器或设备上所能训练的更大的深度学习模型。通过将训练过程在多个 GPU 上并行化，我们有效地增加了可用于训练模型的计算能力。每个 GPU 在训练数据的一个子集和模型参数的一部分上工作。然后通过汇总各个 GPU 的结果来建立整体模型。随着深度学习模型的规模和复杂性不断增加，分布式训练的好处变得更加明显。更大的神经网络需要更多的数据和计算来优化大量的权重和参数。通过利用多个 GPU，我们可以扩大可用资源的规模，以满足这些大规模模型的需求。

目前，分布式深度学习训练中使用的并行性主要有三种类型：

- **数据并行性 (Data parallel)[18]：**在数据并行要求整个模型能够装入每个处理器或设备的内存中，这种方案将大规模的数据集分成多个小批次，分别在不同的处理器或设备上并行计算，以提高深度学习模型的训练速度和效率。在每个训练迭代或历时结束时，模型参数在所有处理器或设备上同步。更具体地说，每个处理器或设备在其训练数据部分的一批训练样本上工作。它在这个本地批次上执行前向和后向传播，以计算梯度并更新其模型副本中的权重和偏差。计算完成之后需要执行同步步骤，来自每个处理器或设备的模型参数被平均到一起（太频繁的同步会因为通信开销而降低训练速度，而太不频繁的同步则会导致独立的模型副本之间出现分歧）。通过数据并行和模型参数的定期同步，训练数据和计算可以有效地分布在多个处理器或设备上，以加快深度学习模型的训练。总的来说，数据并行难度相对较低，只需要并行化已有代码，但是其额外的通信开销随着 GPU 数量增加而增加，大规模训练性能较差。
- **模型并行性 (Model parallel)[19]：**模型并行可用于训练太大而无法放入单个处理器或设备的内存中的模型，但是需要处理器或设备之间进行更多通信模型的参数分割到多个 GPU 卡或服务器上训练。这种方法将复杂的深度学习模型分割成独立的子模型，并分配到多张 GPU 卡或服务

器上，从而减少单个设备的计算负担。在每张 GPU 计算完成对应部分之后，需要将中间的激活值通过网络传输（NCCL[20]）的方式发送给其他 GPU 参与后续阶段参与计算。使用模型并行需要对模型进行重构以分割参数，因此相应的并行度比较高。这种方式可以利用更多的计算资源来加速非常大模型的训练, 但是参数分割和同步亦需要投入额外工作并存在较高的通信开销。总的来说, 模型并行适合那些模块化清晰且参数易于分割的深度学习模型，并且可以更好地随处理器数量线性缩放。

- **流水线并行 (Pipeline parallel)[21]：**流水线并行是一种通过将深度学习模型划分为多个独立阶段, 并行执行不同阶段来加速训练模型的技术。具体操作包括: 首先将模型分割成多个连续的阶段, 如 Embedding 层、卷积层、池化层等并将一个 Batch 的数据划分为多个 Macro-Batch 分配给不同的阶段进行计算。当一个 Macro-Batch 通过一个阶段后, 将结果传递给下个阶段, 直至所有的 Macro-batch 通过流水线。不同阶段的计算可以同时执行, 形成 computaional pipeline。整个 Batch 全部通过后, 损失函数和梯度计算在最后一个阶段完成。流水线并行适用于很大的模型, 超出单个 GPU 容量。它可以利用多个 GPU 来加速训练。流水线并行的优点是可以扩展到多个 GPU 来利用更多计算资源, 利用计算资源更有效率。但是它主要挑战是需要更严格的同步不同阶段的计算, 部分阶段可能存在空转, 需要通过调整 Macro-Batch 大小来缓解。通常模型并行与流水线并行可能一起使用，以更好的平衡流水线并行中各个阶段的计算负载与数据通信开销。

并行训练模型时，我们需要通过合理的任务划分和调度来优化深度学习训练的效率和可扩展性，并根据深度学习模型本身的特点和可用的硬件资源选择最佳的并行方案。从而在保证模型精度的同时，实现训练速度提升，支持更大规模的训练，以及提高可靠性和容错性。这种方式可以更充分地利用计算资源，提高深度学习模型的训练效率和可扩展性，从而适应越来越复杂和庞大的深度学习任务和数据集的需求。

1.4 本文组织结构

本文的组织结构如下:

本文的第二章是 MoE 训练过程的概述。该章主要介绍了 MoE 混合专家模型在分布式深度学习训练中的主要过程, 分析了其正向反向传播的计算过

程, 并解释了为什么 MoE 模型与现有传统集中式训练系统的不匹配。

本文的第三章是基于动态路由的数据分派策略的设计方式。该章首先分析了现有的数据分派策略的主要方式, 并总结了现有设计的不足之处。进而更进一步，给出了我们设计的基于动态路由的数据分派策略具体的方案架构, 包括方案的详细流程和性能分析。

本文的第四章为基于网络拓扑的自动负载均衡策略方案设计。该章首先分析了现有 GPU 数据中心网络拓扑。并结合每个专家在训练过程的负载变化现象, 建立模型寻找最佳的负载均衡方案设计。

本文的第五章为系统实现以及实验结果展示。该章给出了整体的设计方案以及实验结果展示。

本文的第六章为总结与展望。该章总结了本文的主要贡献，以及对未来工作的展望。

1.5 本章小结

本章首先对论文选题的背景和研究意义进行讨论，提出了稀疏混合专家 MoE 模型对于现有深度学习训练系统的挑战，然后分别介绍了本文的研究点。随后对国内外研究现状进行了简要的介绍。

第二章　MoE 模型训练过程概述

MoE 模型在原本 Transformer block 的 FFN 中通过添加门控函数以及一系列相互并列的专家组成。在训练过程中门控函数主要将输入数据分配给不同的专家模型参与前向/反向计算，这一过程主要通过 All-to-All 通信实现。待每个专家计算完成相对应的数据之后，我们需要通过一次额外的 All-to-All 通信，将计算完成的激活值返回原本对应的 GPU 上，并参与后续的计算过程。

2.1 MoE 模型训练

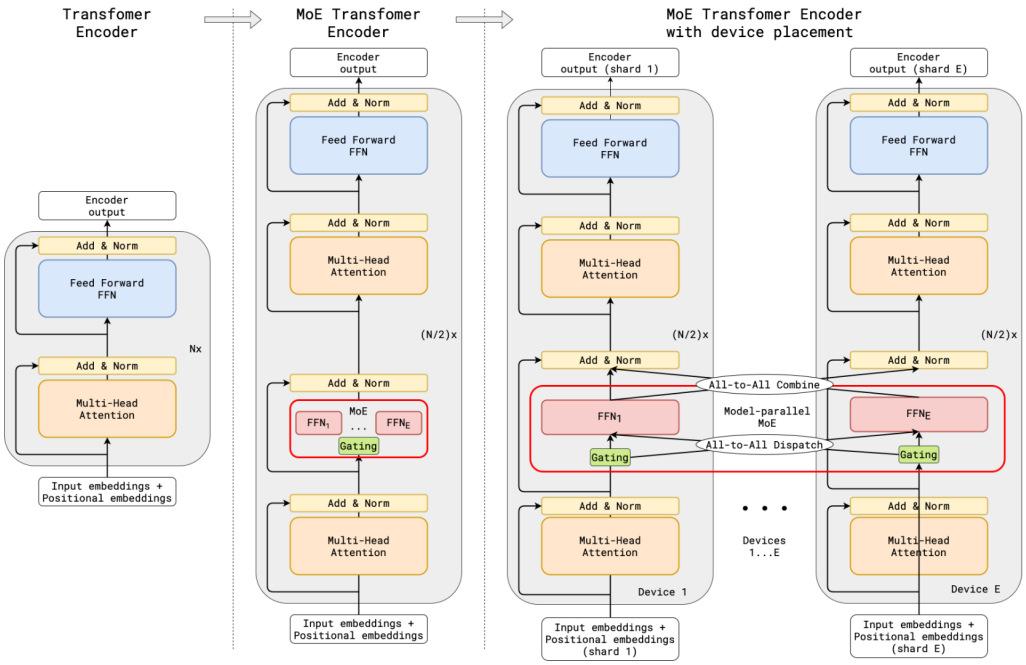

图 2-1　MoE 模型训练过程

图2-1展示了传统的集中式的 Transformer 模型训练过程转变到分布式的 Transformer-MoE 训练过程的示意图。绿色的部分是增加的门控函数，用于决定将数据分派至哪一个 expert 参与计算，红色部分的 $\mathbf{FFN_1}$ – $\mathbf{FFN_E}$ 表示该层中具有 E 个互相独立的专家个数。由于 GPU 的显存有限，不能将所有的专家的参数都在一张 GPU 中保存下。因此 GShard[5] 的研究者们提出了专家并行的概念，将每个专家的参数单独放置在 GPU 上，其他部分采用数据并行的架构，从而实现了一种混合的分布式训练过程。在训练过程结束之后，通常需

要额外的 All-reduce 用于同步数据并行模块的梯度，最后根据每个可训练参数的梯度，调用优化器更新参数。

整体的训练流程总结如下:

- **1. 数据划分**

  每个 GPU 将训练数据随机筛选，确保每一轮训练中每个 GPU 上包含的数据子集是互不重复的。每个子集分配到不同的 GPU 或计算节点上进行训练。

- **2. 数据并行 & 专家并行**

  在 MoE 模型训练中，数据并行和专家并行通常结合使用，以实现更高效的分布式训练。具体而言，可以将训练数据划分为多个子集，并将每个子集分配到不同的 GPU 或计算节点上。在每个 GPU 或计算节点上，可以使用专家并行的方式对每个专家模型进行训练，同时使用数据并行的方式对整个 MoE 模型进行训练。这样可以将计算负载和数据负载都分散到多个 GPU 或计算节点上，从而加速整个 MoE 模型的训练过程。

- **3. 前向传播**

  每一层的前向传播主要在以下两个步骤区别与传统的密集型 Transformer 计算。(数据分派和专家计算)。

  在数据分派阶段，MoE 模型使用 Gating 门函数对每个输入数据进行加权打分，以决定每个专家模型的贡献。这个 Gating 函数由一层 MLP 和 softmax 组成。具体而言，MoE 模型将输入 x 输入到门控模型中，得到一个 E 维的得分，$f = [f_1, f_2, \cdots, f_E]$，$f_i$ 表示第 i 个专家模型的权重得分。门控向量的每个元素都是非负的，并且它们的和等于 1，即 $\sum_{i=1}^{K} g_i = 1$。最后通过全局所有 GPU 的 All-to-All 通信，将对应数据发送给相应的 GPU，参与后续的计算。

  专家计算阶段，每个 GPU 上接收到其他 GPU 传输来的数据后（假设有 d 维），将其输入相应的专家模型参与前向计算。得到 d 维的输出向量 r。之后通过反向的 All-to-All 通信，将激活值返回对应的 GPU 上，得到每个 GPU 上原本数据的输出向量 $z = [z_1, z_2, \cdots, z_E]$。然后，MoE 模型将每个输出向量 z_i 与对应的门控向量 g_i 进行按元素乘法，得到一个

加权输出向量 w，其中 $w_i = g_i z_i$。最后，MoE 模型将所有加权输出向量 w_i 进行累加，得到最终的输出向量 y，其中 $y = \sum_{i=1}^{E} w_i$。

- **4. 反向传播**

  是用于计算 MoE 模型的梯度，其过程类似于传统的神经网络模型。在完成前向传播计算之后，使用 PyTorch 的自动求导机制（autograd）建立计算图，并自动计算每个参数的梯度。具体而言，可以使用 loss 函数对模型的输出进行评估，并计算输出和目标值之间的误差。然后，使用误差及其对模型参数的导数，计算每个参数的梯度。反向传播过程可以使用链式法则（chain rule）实现，即将误差从输出层向输入层传播，并依次计算每个参数的梯度。

- **5. 梯度计算**

  是将每个模型的梯度进行同步，以便在参数更新时使用。由于 MoE 模型的分布式训练过程涉及多个 GPU 或计算节点的并行计算，因此需要使用 all-reduce 等方法将所有模型的梯度进行同步。

- **6. 参数更新**

  使用优化器对模型参数进行更新，以最小化损失函数。在 MoE 模型训练中，可以使用常见的优化器，如 Adam、SGD 等，对每个模型的参数进行更新。通常，参数更新的速率会受到学习率（learning rate）等超参数的控制，以平衡模型的收敛速度和稳定性。

- **7. 重复迭代**

  将以上步骤重复多次，直到模型收敛为止。在每次迭代中，可以使用不同的训练数据子集，防止模型陷入局部最优解。

2.2 本章小结

本章我们主要分析了现有的 MoE 模型训练过程，MoE 模型的训练过程通常比传统的神经网络模型更为复杂和耗时，需要充分利用分布式计算和并行计算等技术，以提高训练速度和效率。同时，MoE 模型的设计和调整也需要考虑多个因素，如门控模型的设计、专家模型的选择和训练方式等，以提高模型的性能和泛化能力。

第三章　基于动态路由的数据分派策略

在 MoE 模型的每一层前向/反向传播中，都有一个计算各个专家权重的门网络，他的作用主要是根据每个输入样本的特征来预测每个专家的权重或者分配的系数。但是门网络的数据分派方式又会影响到全局通信量，因此如何设计合适的数据分派方式，在保证模型收敛速度的同时，不会带来巨大的系统总通信量，使值得深入研究的问题。

3.1 MoE 数据分派方式概述

门网络的架构通常是 MLP+Softmax，通过学习可调节的 MLP 权重参数，预测各个计算专家的权重，从而实现对输入样本的有效处理。

3.1.1 数据分派流程

在 MoE 模型的训练过程中，他的过程如下：

- **专家容量设置**：在 MoE 模型训练开始之前，人为地为每个专家设定一个容量的限制。这个容量限制可以根据每个专家的计算资源、存储能力或其他约束条件来确定。当数据量超过专家的容量限制时，系统会对数据进行强制截断，以确保专家在其容量范围内进行计算。通过设置专家容量限制，可以控制每个专家参与计算的数据量，从而平衡计算资源的使用和模型的性能。

- **输入特征**：：输入样本的特征被提供给门网络作为输入。通过学习可调节的权重参数，门网络预测每个专家的权重。这些权重参数在训练过程中通过优化算法进行调整，以最佳地预测专家权重。

- **权重计算**: 具体而言，MoE 模型将输入 x 输入到门网络的 MLP 层，之后通过 Softmax 函数规范化得到一个 E 维的得分，$f = [f_1, f_2, \cdots, f_E]$，其中 f_i 表示第 i 个专家模型的权重得分。且权重之和等于 1，即 $\sum_{i=1}^{K} f_i = 1$。

- **Top-k 分派**: 根据门网络的输出，通常采用 Top-k gating 的方式将数据分派给具体的专家进行计算。一种常见的方式是使用 Top-1/Top-2 Gating，

根据专家权重得分从高到低对专家进行排序，并将数据发送到选择的前 k 个专家中参与后续计算。

- **系统开销**: 由于采用 Top-k Gating，整个系统的通信数据量是 Top-1 All-to-All 通信量的 k 倍。因为 All-to-All 通信是全局的通信操作，需要权衡模型的收敛速度和系统的通信量，选择适当的 gating 策略。

这些步骤组成了 MoE 混合专家系统中数据分派和权重计算的基本过程，使得不同专家能够根据其权重参与输入样本的处理和计算。

3.2 动态路由的数据分派系统设计

动态路由是一种用于数据分派的系统设计和算法，它可以根据数据特征和专家模型的状态动态地分配数据到合适的专家进行计算。

3.2.1 算法设计

主要设计包含以下步骤:

1. 初始阶段：类似于传统的 gating 策略，系统在计算开始之前需要进行一些初始设置。这包括设定专家容量，确定输入特征，并通过 MLP 和 Softmax 层计算权重。这些权重代表每个专家在当前数据上的重要程度或贡献度。

2. 动态数据分派：与传统的固定分派模式不同，动态路由的数据分派是根据门网络的打分结果进行动态决策。在分派数据时，根据权重得分 $f = [f_1, f_2, \cdots, f_E]$ 从高到低排列，选择得分最高的两个专家，记为 f_{k1} 和 f_{k2}。如果 $f_{k1} - f_{k2}$ 小于预先设定的阈值 Threshold，系统将采用 Top-2 Gating 的方式将数据发送给这两个专家参与计算。否则，系统按照 Top-1 Gating 的形式选择得分最高的专家，将数据发送给该专家进行计算。即系统动态地选择使用 Top-k（k=1 或 2）的专家参与计算，根据权重得分的差异性来确定采用哪种分派模式。

3.2.2 系统实现

对于系统实现而言，采用动态路由的方式确实会带来一些困难。在传统的 Top-k Gating 中，每个 GPU 在 All-to-All 通信时发送的数据量是均等的，因

此在系统实现时比较容易，可以直接调用 PyTorch（NCCL）的 All-to-All 通信实现来完成通信过程。

然而，在采用动态路由后，每个 GPU 上发送的数据量是不均等的。如果仍然按照传统的 Top-2 Gating 的方式发送数据，实际上并未减少通信量。为了解决这个问题，我们采用了 All-to-All 通信的 unequal 模式，该模式不会发送额外的用于填充的 0，从而达到减少数据量的目的。

(1) Unequal All-to-All **通信实现**

通过实现 unequal 模式的 All-to-All 通信，我们可以根据动态路由的结果，灵活地分派数据并减少通信量。这在系统实现上可能会带来一些挑战，在实现过程中，需要对通信模块进行适当的调整，以支持 unequal 模式的数据传输。

通过在算法上实现动态路由的数据分派模式，在系统上实现 unequal All-to-All 通信模式。我们可以提高系统的通信效率和计算性能。这样，系统能够根据实际的数据分派需求，灵活地选择数据发送的方式，并减少不必要的通信开销。

3.3 本章小结

第四章　面向加密数据去重的动态用户更变方案设计

4.1 动态用户更变方案概述

4.1.1 系统构成

4.1.2 威胁模型及假设

4.1.3 方案设计要求

4.2 动态用户更变方案架构设计

4.2.1 符号定义及概念

4.2.2 用户更变详细流程

4.2.3 安全性分析

4.3 本章小结

第五章　基于通信延迟的地理位置验证方案设计

5.1 地理位置验证方案概述

5.1.1 节点距离估计

5.1.2 基本测量流程

5.2 地理位置验证方案架构设计

5.2.1 验证方案框架详细流程

5.2.2 结果置信度计算

5.3 本章小节

第六章　系统测试与分析

6.1 系统测试环境

6.2 加密数据去重方案

6.3 动态用户更变方案

6.4 地理位置检验方案

6.5 系统界面设计

6.6 本章小节

第七章　总结与展望

7.1 本文工作总结

7.2 未来工作展望

参考文献

[1] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[2] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33:1877-1901.

[3] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models[J]. arXiv preprint arXiv:2001.08361, 2020.

[4] Sharir O, Peleg B, Shoham Y. The cost of training nlp models: A concise overview[J]. arXiv preprint arXiv:2004.08900, 2020.

[5] Lepikhin D, Lee H, Xu Y, et al. Gshard: Scaling giant models with conditional computation and automatic sharding[J]. arXiv preprint arXiv:2006.16668, 2020.

[6] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.

[7] Deepspeed[EB/OL]. https://www.deepspeed.ai/.

[8] Dai Z, Yang Z, Yang Y, et al. Transformer-xl: Attentive language models beyond a fixed-length context[J]. arXiv preprint arXiv:1901.02860, 2019.

[9] Shoeybi M, Patwary M, Puri R, et al. Megatron-lm: Training multi-billion parameter language models using model parallelism[J]. arXiv preprint arXiv:1909.08053, 2019.

[10] Rajbhandari S, Rasley J, Ruwase O, et al. Zero: Memory optimizations toward training trillion parameter models[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-16.

[11] Rajbhandari S, Li C, Yao Z, et al. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale[C]//International Conference on Machine Learning. PMLR, 2022: 18332-18346.

[12] He J, Qiu J, Zeng A, et al. Fastmoe: A fast mixture-of-expert training system[J]. arXiv preprint arXiv:2103.13262, 2021.

[13] Ott M, Edunov S, Baevski A, et al. fairseq: A fast, extensible toolkit for sequence modeling [J]. arXiv preprint arXiv:1904.01038, 2019.

[14] Hwang C, Cui W, Xiong Y, et al. Tutel: Adaptive mixture-of-experts at scale[J]. arXiv preprint arXiv:2206.03382, 2022.

[15] Xu C, McAuley J. A survey on dynamic neural networks for natural language processing[J]. arXiv preprint arXiv:2202.07101, 2022.

[16] Roller S, Sukhbaatar S, Weston J, et al. Hash layers for large sparse models[J]. Advances in Neural Information Processing Systems, 2021, 34:17555-17566.

[17] Chen C, Li M, Wu Z, et al. Ta-moe: Topology-aware large scale mixture-of-expert training[J]. Advances in Neural Information Processing Systems, 2022, 35:22173-22186.

[18] Ben-Nun T, Hoefler T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis[J]. ACM Computing Surveys (CSUR), 2019, 52(4):1-43.

[19] Shazeer N, Cheng Y, Parmar N, et al. Mesh-tensorflow: Deep learning for supercomputers[J]. Advances in neural information processing systems, 2018, 31.

[20] Nccl[EB/OL]. https://github.com/NVIDIA/nccl.

[21] Huang Y, Cheng Y, Bapna A, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism[J]. Advances in neural information processing systems, 2019, 32.

致　谢

感谢 XXX...

毕业设计小结

毕业论文是大学四年的最后一份大作业...

本科期间研究成果产出

以第一作者身份发表论文

- **NSS 2023**:

1

1

1

参与科研项目

- **重点研发**:

附　录

这是一份附录，请放置一些独立的证明、源代码、或其他辅助资料。