大模型训练技术综述

A Survey on Large Model Training Technologies

田海东/TIAN Haidong，张明政/ZHANG Mingzheng，
常锐/CHANG Rui，童贤慧/TONG Xianhui

（中兴通讯股份有限公司，中国 深圳 518057）
(ZTE Corporation, Shenzhen 518057, China)

DOI：10.12142/ZTETJ.202402004
网络出版地址：http://kns.cnki.net/kcms/detail/34.1228.TN.20240419.0912.002.html
网络出版日期：2024-04-20
收稿日期：2024-03-02

摘要：实现高效训练已成为影响大模型应用普及的关键要素之一。按照数据准备、数据加载、模型初始化及评估、训练并行、模型状态保存的一般训练流程，对大模型高效训练的主要技术进行分析和论述。面对大模型规模的持续增长、数据处理类型的扩展，现有大模型训练技术仍存在较大的优化空间。认为未来大模型训练重点研究方向包括以数据为中心、数据加载智能化和异构加速、网络通信领域定制、训练并行及自动化。

关键词：大模型；数据准备；数据加载；模型初始化；模型评估；训练并行；训练网络；检查点

Abstract: Achieving efficient training has become one of the key factors affecting the popularization of large model applications. The main technologies of efficient training of large models are analyzed and discussed according to the general training process of data preparation, dataloader, model initialization and evaluation, training parallelism, and model state preservation. In the face of the continuous growth of large model scale and the expansion of data processing types, there is still a large room for optimization of existing large model training technologies. In the future, the key research directions of large model training include data-centric, intelligent dataloader and heterogeneous acceleration, customization in the field of network communication, training parallelism and automation.

Keywords: large model; data preparation; dataloader; model initialization; model evaluation; training parallelism; training network; checkpoint

引用格式：田海东, 张明政, 常锐, 等. 大模型训练技术综述 [J]. 中兴通讯技术, 2024, 30(2): 21-28. DOI: 10.12142/ZTETJ.202402004
Citation: TIAN H D, ZHANG M Z, CHANG R, et al. A survey on large model training technologies [J]. ZTE technology journal, 2024, 30(2): 21-28. DOI: 10.12142/ZTETJ.202402004

近年来，深度学习[1-5]领域取得了重大进展，特别是以ChatGPT为代表的大语言模型（下文统称大模型），在人机问答、内容生成领域展示出了人工智能的强大威力，让人们看到了通用人工智能（AGI）的曙光。未来大模型有望成为一项引发新一代工业革命和促进社会发展的变革性技术。然而，大模型的训练对数据准备与预处理、并行训练过程计算访存效率、过程状态保存与故障恢复等方面都有着苛刻的要求。如何实现大模型的高效训练，已成为学术界和工业界共同的研究热点。本文中，我们将按照模型训练的一般流程，并聚焦主要训练过程，对大模型高效训练的主要技术进行分析和论述。

1 数据准备

按照所能处理的数据类型，大模型可分为两类：语言类大模型[6-8]和多模态类大模型。语言类大模型的文本数据来源不仅包括网页、对话文本、书籍等通用数据，还包含多种语言的语料库、科技论文、代码等专用数据。多模态类大模型的数据一般来源于网页和专用数据集，常见形式是图片文本对。语言类大模型训练数据的准备流程一般包括收集、过滤、去重、隐私去除、分词。在转化为向量数据后，相关数据被加载到图形处理器（GPU）中进行训练[9]。而多模态类大模型则需要对图片、视频、语音等非文本的数据对象，先进行适当的解码、缩放、裁剪、归一化处理，再通过特征提取将其转化为向量数据。数据准备的主要方法如下：

1）去重和过滤

由于来源数据良莠不齐，大模型中会不可避免的引入噪声、冗余、无关甚至有害的数据，有些数据还会涉及个人隐私。文献[10]研究发现，反复重复一小部分数据可能会对系统性能造成巨大危害。这是因为重复数据的训练会导致系统从零开始训练和微调性能变差。所以删除重复的数据，可使

系统使用更少的训练步骤来实现相同或更好的准确性。如何定量和定性地理解数据集本身就是一个研究挑战。文献[11]提出了后缀数组子串处理和相似度匹配算法两种可扩展的技术，以检测和删除重复的训练数据。

RefindedWeb[12]对数据集的处理方式更为激进，采用了Bloomfilter和Simhash近似去重，这导致删除率远高于其他方法。RefindedWeb同时证明，只用网页数据并通过严格的过滤、去重和脚本处理等手段，同样可以获取大模型所需的训练语料。

文献[13]则引入数据年龄、质量及毒性危害程度、领域组成等更多维度的量化评价指标，分析对大模型训练的影响，为模型训练数据准备提供了指导方法。

2）建立数据生产体系

从零开始预训练大模型需要高昂的成本。开源的大模型通常不附带开源的数据集。因此，如何高效体系化地收集处理数据显得尤为重要。Ziya2[14]构建了完整的数据生产体系，包括预处理数据、自动评分、基于规则的过滤、消除重复内容和评估数据等子任务。RefinedWeb[12]则提出宏数据优化（MDR）处理数据的思想，侧重于去重和过滤。

3）隐私和安全

一般来说，模型泄露数据的主要原因是过拟合。此时模型会记住数据集中的数据。在大规模数据上进行训练也会存留“记忆”。文献[15]基于此在GPT2上进行攻击验证，认为在数据准备、增加噪声、微调各阶段中都需要有防止数据泄露的措施。删除数据集敏感数据是其中一种方法。知识遗忘作为一种替代方法，也可以用来降低语言模型的隐私风险[16]。文献[15]在执行遗忘时不仅能提供更强的隐私保障，还能保障模型性能不下降，并发现一次性忘记许多样本会导致显著的模型性能下降，而顺序遗忘数据可以缓解这种情况。

2 数据加载

大模型训练需要大量数据、资源和时间，它涉及服务器中所有资源的综合利用，如图1所示。数据加载是指，从存储器中读取数据，根据不同的模型训练要求，对读取的数据设置不同的预处理规则。存储介质、缓存大小、用于获取和预处理数据的CPU线程数量等因素，都会影响到数据加载的性能。

理想情况下，数据加载部分需要稳定地将预处理后的数据发送给GPU，以便GPU能够持续进行数据计算，充分发挥计算能力。但在实际工作中，主流的训练框架如PyTorch、TensorFlow等，在进行数据加载和数据预处理时都存在性能瓶颈。我们将这些瓶颈统称为数据停滞。

目前有多种解决方案都在研究如何解决模型训练过程中的数据停滞问题，主要包括以下几种：

1） 缓存策略

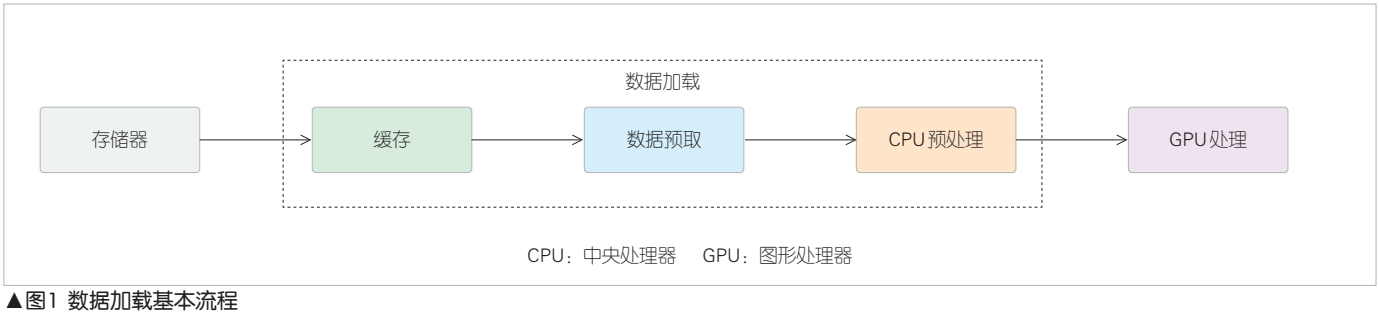
很多训练平台（如PyTorch等）提供了DataLoader，支持提前将数据从存储器读取到内存中，并且通常依赖操作系统的页面缓存技术来缓存重复访问的原始训练数据。通过实验数据分析，数据集的访问模式与操作系统缓存替换策略并不一致。操作系统的页面缓存机制在提升训练效率方面并不明显。

分析大模型训练有其独特的数据访问模式。每个训练周期内系统会以随机方式遍历一次数据集中的所有数据样本。而基于样本重要性、动态打包和多任务处理机制等高速缓存置换算法[17]的研究，则能够有效减少数据停滞时间，提升训练速度。

2） 分布式数据加载

分布式技术可以将训练任务分配到多个计算节点上进行并行计算，以提高训练效率。分布式训练将数据分成多个批次，然后在不同的存储节点上并行加载数据，以减少数据加载等待时间。对于分布式训练过程中存在的同一份数据在多个节点上重复缓存的情况[18]，如果缓存之间缺乏协调，分布式训练就容易受到存储接口的限制。此时，可通过设置数据流策略将存储接口操作与计算操作并行化处理，来进一步提高训练效率。

另外，当参数量很大时，数据通信量也可能会成为模型



▲图1 数据加载基本流程

训练的瓶颈，此时需要通过通信和计算重叠等方式降低通信等待时间[19]。

3） 数据预处理卸载

将数据预取和数据预处理的过程卸载到GPU等设备上，构建高效的数据流水线，可以减少数据加载的等待时间，让训练过程更加流畅。DALI[20]是专门为GPU优化的数据加载库，通过在GPU上运行数据处理管道从而加速数据预处理过程。DALI的性能优于传统的数据流水线，但代价是占用了GPU有限的计算和内存资源。发掘训练模式和数据特征，进行数据压缩，有助于实现GPU内存优化[21]。目前PyTorch等主流深度学习框架都已经支持DALI的使用。

此外，数据流分析工具如DS-Analyzer[22]，可以针对具体的训练场景，精确发现数据流中的性能问题。对数据停滞进行预测和分析，能够为提高训练效率指明具体的改进方向，例如：CoorDL[22]验证了在特定情况下，模型训练能够获得比DALI更高的资源利用率和更好的性能。

3 模型初始化及评估

这一章节中我们主要探讨两个问题：如何做好预训练大模型的初始化准备，以及如何在训练过程中确定更好的模型评估指标，具体包括模型规模的选择、模型超参数的初始化设置、模型的评估等方面。

3.1 模型规模选择

在进行预训练之前，了解大模型的扩展法则可以帮助我们很好地平衡模型大小、数据的规模以及计算量之间的关系。这里我们给出两种大模型的扩展法则，具体说明如下：

1） KM扩展法则

Decoder-only的模型算力有如下关系：

$$C \sim \tau T = 6PD, \tag{1}$$

其中，C表示模型的总计算量，τ表示吞吐量，T表示训练时间，P表示模型的参数量，D表示token数。

2020年OpenAI[23]团队首次通过实验给出了模型性能和模型参数量、数据规模、模型计算量之间的幂律关系。由幂律关系发现，在相同算力下模型的参数量更重要。

2） Chinchilla扩展法则

Google的DeepMind团队提出了Chinchilla扩展法则[24]。他们在一个更大范围（7 000万到160亿参数）的模型和更大范围（50亿到5 000亿tokens）的数据条件下探讨上述KM扩展法则，得出一个系数不同的幂律关系，即Chinchilla扩展法则。Chinchilla扩展法则认为模型大小和数据大小应该以同等的比例增加。

3.2 模型初始化

大模型训练是一个高度实验性的过程，需要承担较高的试错成本。过程中会涉及大量的超参数设置，包括权重初始化、归一化方法、激活函数、位置嵌入、学习率、优化器等。这里我们介绍一些常见的初始化设置。

1） 权重初始化

合理的初始化权重可以帮助模型更快地收敛，使模型拥有更好的性能。最常见的就是高斯噪声初始化。为解决深层Transformer收敛困难的问题，2019年XU Q.[25]认为收敛困难的原因是层归一化（LN）和残差连接相互影响导致梯度消失，提出了利普希茨约束参数初始化（LRI）的参数归一化方式。与此同时，ZHANG B.[26]提出了一种参数初始化方式DS-Init。该方法通过在初始化阶段减少模型参数的方差，来减少残差连接输出的方差，从而缓解反向传播过程中数据通过正则化层时的梯度问题。2020年，HUANG X. S.[27]提出了一种参数初始化策略T-fixup。该策略可以使模型参数在没有预热（WarmUp）和层归一化的情况下仍能够更新收敛。

2） 归一化

训练不稳定是大模型面临的一个难题。LN[28]被广泛应用到Transfromer架构中。LN的位置对于大模型的性能至关重要。2020年，XIONG R.等提出了Pre-LN[29]。相对于一般Tranformer中的LN，研究人员将LN这一阶段提前，解决了学习率WarmUp阶段超参敏感问题，同时优化了收敛过程速度慢等问题，但这也带来了一定的模型性能损失。Chinchilla[24]则采用RMS Norm[30]的方式，取消了传统LN上的均值计算，在训练速度和性能方面都具有优势。

3） 激活函数

为了获得良好性能，前馈网络需要设置合适的激活函数。现有大模型中，GeLU[31]被广泛使用。此外，最新的大模型如PaLM和LaMDA，使用了GeLU的变体SwiGLU[32]和GeGLU[33]，取得了更好的性能。

3.3 模型评估

在模型训练过程中（或对于那些训练好的模型），我们需要评估模型的能力。通常会有很多基准数据集可用于评估模型的逻辑推理、翻译、自然语言推理、问答等方面的能力。这里我们对常用评估方法及指标做一个介绍。

1） 文本对比

一些常规的指标，比如BLEU、ROUGE、METEOR等，

可用来衡量文本的重叠度。PYRAMID[34]可以衡量语义重叠度。对于代码生成模型，pass@k[35]是一个重要衡量指标。对于多输出的复杂模型，KoLA[36]将大模型的评价和认知层面联系起来。KoLA的评测任务由认知层级决定。认知层级包括知识记忆（KM）、知识理解（KU）、知识应用（KA）、知识创造（KC）。

当然，我们也可以从其他方面来评估模型，比如：鲁棒性（NL-Augmenter[37]的语义不变扰动）、基于计数的性别和种族bias[38]、不确定性及公平性等。

2） 自动评估和人类评估

自动评估是指利用一些小模型对大模型的结果进行评估，例如毒性评估相关（ML-based Perspective[39]）、对话系统等。

此外，我们也可以使用单纯的人类评估。人类评估是自然语言处理领域中衡量模型或算法性能的关键方法。然而，人类评估存在不稳定、可重复性低等问题，这可能导致评估结果不够准确。HUSE[40]尝试模仿人类评估的方法，并将人类评估和统计评估相结合，实现多样性评估。

4 训练并行及网络

随着深度学习模型参数和数据规模的增长，传统的单机单卡模式已无法满足大模型的训练要求。因此，我们需要基于单机多卡、多机多卡进行大模型的分布式训练。而利用计算集群，从大量数据中高效地训练出性能优良的大模型是分布式训练的首要目标。为了实现该目标，需要根据硬件资源与数据模型规模的匹配情况，来对计算任务、训练数据和模型进行划分，从而实现分布式训练并行。

4.1 常见的并行策略

这里我们介绍几种常见的并行策略，即数据并行、张量并行和流水线并行。在大模型训练的过程中，通常会将上述并行策略进行组合使用，即混合并行。

1） 数据并行

数据并行是提高训练吞吐量的基本方法之一。它将模型参数和优化器状态复制到多个GPU上，然后将整个训练语料库分配到这些GPU上。这样，每个GPU只需要处理分配给自己的数据，并执行前向和反向传播以获取梯度。在不同GPU上计算出的梯度将进一步聚合以获得整个批量的梯度，然后更新所有GPU上的模型。由于不同GPU上的梯度计算是独立进行的，数据并行机制具有高度可扩展性，因此可以通过增加GPU数量来提高训练吞吐量。以PyTorch为例，数据并行方式有：数据并行（DP）、分布式数据并行（DDP）、完全分片数据并行（FSDP）等。

2） 张量并行

张量并行专注于分解大模型的张量，将一个张量沿着特定维度分成N块。每个GPU保持整个张量的$1/N$，同时不影响整个计算图的正确性。张量并行可以通过跨GPU通信将多个GPU的输出结果聚合成最终结果。最早的张量并行方案由Megatron-LM[41]提出，它是一种高效的一维张量并行实现方法。为了平均分配计算和内存负荷，Colossal-AI[42]把张量沿着两个维度进行切分，这就是二维张量并行。除此之外，Colossal-AI还可以支持更高维度的张量并行。

3） 流水线并行

流水线并行是指将大模型的不同层分配到不同的GPU上，以降低单个GPU的显存消耗。然而，传统流水线并行方式的GPU空泡率较高。针对该问题，Gpipe[43]提出了micro-batch的方式以减少空泡率。PipeDream[44]在Gpipe的基础上使用1F1B的方式优化流水线并行策略，以减少流水的空闲时间和显存。在后续的一些研究中，PipeDream-2BW和PipeDream-Flush[45]等基于原始的PipeDream做了进一步的改进。

4） 序列并行

序列并行是指将序列这个维度划分到不同的GPU上进行并行计算。例如LI S.[46]为解决大模型输入序列长度的限制，将输入序列分割到多个GPU上，并提出环自注意力，将环状通信与自注意力相结合，可以处理超过1.14×10^5的长度序列。

同样地，Megatron-LM[41]在进行张量并行的时候，将LayerNorm和Dropout的输入按长度进行了划分，使得各GPU只需要完成一部分Dropout和LayerNorm操作，并使用选择性激活重计算以减少激活显存。

4.2 其他并行优化策略

1） 基于显存的优化技术ZeRO

ZeRO[47]是由DeepSpeed提出的在数据并行过程中降低显存的一种技术。该技术可以使得单个GPU的显存占用随着GPU的数量增加而线性下降。ZeRO一共提供了3种解决方案：优化器分区、梯度分区和参数分区。前2种方案不会带来新的通信开销，第3种方案会增加50%的通信开销。与此同时，ZeRO提供ZeRO-R在数据并行节点间划分激活，来减少激活的显存开销。

2） 基于模型结构的并行

混合专家（MoE）模型是一种基于模型结构的并行架构。它将大模型拆分成多个小模型（专家模型），在每一轮

迭代中根据样本决定一部分专家用于计算，并引入可训练的门机制保证计算能力的优化。Gshard[48]首次将MoE应用到Transformer上，将间隔的前馈神经网络（FFN）替换成MoE。Switch Transformers[49]是在T5模型上应用MoE设计的，它简化了MoE的路由算法，具有门机制，即每次只推选一个专家。

在后续的研究中，微软的Deepspeed-MoE[50]提出了一种端到端的MoE训练和推理解决方案，有效减少了MoE模型的大小。Faster-MoE[51]提升了MoE模型分布式训练效率，与大模型优化策略（ZeRO、Gshard等）相比获得了显著的性能加速。

3） 自动并行

自动并行的目标是：用户给定一个模型和所使用的机器资源后，系统能够自动地帮用户选择一个较好或者最优的并行策略来实现高效执行。可以说，自动并行是分布式并行的终极目标，它能够减少或避免工程师手动设置分布式并行策略。自动并行分为半自动和全自动两种模式。其中，半自动模式是指用户自己指定张量的切分方式，如Gshard[48]；全自动模型是指由框架自适应选择切分方式，如Flexflow[52]等提到的全自动并行切分方案。

目前许多训练框架如PyTorch、TensorFlow等实现了自动并行。Alpa[53]通过自动搜索intra-op的调度和inter-op的切分方式，几乎兼顾了所有的并行策略，是自动并行的集大成者。

4.3 训练网络

大模型训练集群的各个计算节点之间通过网络进行互联。网络性能直接决定节点间的通信效率，进而影响整个训练集群的吞吐和性能。随着模型规模的持续增大，大模型训练网络面临着多种挑战：大规模扩展、高通量和低延迟等，除了需要带宽增强等基础技术的支撑[54]，还需从互联协议、网络拓扑、在网计算等多个方面进行优化。

1） 互联协议

训练网络互联技术通常分为两类：总线互联协议（包括：NVLink、CXL等）和网络互联协议（包括：RoCE、Infiniband等）。其中，前者用于计算芯片之间短距离、小规模的互联，而后者则用于计算节点之间长距离、大规模的数据通信。随着总线和网络技术的发展，这两类技术已出现逐渐融合的趋势。例如，英伟达的NVLink 4.0已经可以支持256个GPU的互联，CXL在其最新的规范中也明确将支持机架间的互联。

2） 网络拓扑

大模型训练对网络拓扑的扩展性、可靠性和成本等都提出了更高要求。在高性能计算的发展中，Torus无疑占据了重要的位置。相比于Torus结构，胖树网络路由算法更容易实现，网络性能相对更出色。但是胖树网络在扩展至更大规模时需增加网络层数，从而导致链路数随之指数增长，这会大大增加网络成本。Dragonfly由J. JIM等在2008年提出[55]。它的特点是网络直径小、成本较低，在高性能计算方面有着显著优势。然而，面对整体网络节点的增多，Dragonfly等网络结构依然面临网络连线复杂、网络设计成本高、所需全局光纤数多等挑战。

除了上述拓扑结构，MIT和META的rail-only[56]等还提出了定制化拓扑结构。这些拓扑结构专门针对大模型的通信需求进行设计，目的是在提升性能的同时显著降低成本。

3） 在网计算

在网计算通过网络交换侧和端侧设备的协同，利用网络内部的硬件计算引擎，在网络通信过程中实现复杂操作的卸载。基于树状聚合的机制，在网计算可以同时支持多个集合操作。以典型的AllReduce算子为例，传统的通信交互复杂度为$O(\log N)$（N代表网络节点规模），启动在网计算功能后，其交互复杂度变为$O(C)$（C代表网络层级），大大简化了计算节点间的通信交互过程，提升了计算效率。

在训练网络中最知名的在网计算技术是英伟达的可扩展分层聚合和归约协议（SHARP）[57]。Intel提出的switchML[58]系统在其Tofino专用芯片的可编程交换机上，实现了All-Reduce操作，充分利用了交换机的编程能力。

5 训练状态保存

随着参数规模达到千亿级，大模型训练时长会达到数十天，训练过程也可能因各种软硬件故障而中断。这就需要定期保存模型训练的中间状态，包括GPU内存中的模型参数和优化器状态。发生故障时，将最近的检查点载入到GPU内存中，可以实现快速的故障恢复，系统此时只会丢失很短时间内的计算结果。然而，检查点操作过程中序列化、压缩、文件IO的低效所引起的检查点停滞问题，也会阻塞训练任务，浪费GPU计算资源。因此，我们需要对检查点操作进行优化。主要优化方法如下：

1） 异步处理

GPU与CPU处理进行异步设计时，GPU主要完成前向传播、后向传播，CPU完成参数更新和检查点。微软Fiddle团队在CheckFreq[59]中使用动态建模分析，将CPU处理的检查点快照和持久化进行后台异步处理。DeepFreeze[60]设计了VELOC框架用于实现序列化和压缩异步。Gemini[61]给出的交

错流水也是异步的思想。

2） 轻量级、细粒度任务调度

将接口访问任务拆解为轻量级、细粒度子任务可以实现局部并行。例如：DeepFreeze[60]通过建立有向无环图实现分片和序列任务的重新组织调度，进而可以实现分层并行；Gemini[61]使用交错流水的方式进行接口访问任务调度。文献[62]建立分层模型，并使用模拟的方法改进接口传输调度算法。

3） 检查点计划及存储策略

关于检查点的生成频度，Mimose[63]等研究出一种GPU内存在线估算器，可以预测给定激活张量的内存使用率输入，并生成一个检查点计划，有效避免GPU内存溢出问题。

模型训练状态的存储策略也会影响检查点效率。在分布式训练中，可通过副本布局策略化来提升检查点的保存和读取效率。Gemini[61]采用多副本的方式，在本地和远程机器的CPU内存中维护检查点，并通过环状拓扑算法提高本地读取副本的恢复时间。

6 总结与展望

本文按照大模型训练的一般流程，回顾和总结了大模型训练主要阶段的相关技术背景及要点。随着大模型参数规模的不断增大、多模态数据处理类型的扩展，大模型训练的各个阶段都存在较大的优化空间。为进一步提升训练效率，我们认为后续还需重点展开以下几个方面的研究：

1）以数据为中心。数据的质量和数量对大模型的训练结果非常重要，这已经成为学术界和产业界的共识。很多研究人员开始转向以数据为中心的研究，其主要目的是设法提升数据质量，增加数据数量，而不是过多地考虑模型结构。这种转变在大模型领域尤其明显。

2）数据加载智能化和异构加速。根据模型需求动态调整数据加载策略，并结合事务感知或应用感知的缓存预取策略，有助于加快数据加载过程。此外，对于图像、视频、音频等非文本数据，如何结合专用集成电路（ASIC）或现场可编程门阵列（FPGA）等异构加速技术来有效提升数据预处理的效率，也是后续重要的研究方向。

3）网络通信领域定制。针对大模型训练场景特征，融合CXL等低延迟总线技术的发展，网络通信还需在新型网络拓扑、流量工程优化、互联总线协议领域定制等方面进行针对性优化，以更好地适配大模型训练网络的特点。

4）训练并行及自动化。大模型的算法结构和规模正在快速迭代，如何充分利用有限的计算存储网络资源，通过多维度细粒度的并行拆分策略和卸载技术，实现高效的训练并行，是一个需要持续研究的主题。未来，在用户给定模型和机器资源后，能够有效组合多种并行技术，自动帮助用户制定最优的并行策略，是分布式训练并行的终极目标。

致谢

感谢中兴通讯股份有限公司熊先奎、张景涛、姚海东和朱炫鹏对本研究的帮助！

参考文献

[1] AWADA U, ZHANG J K, CHEN S, et al. Machine learning driven latency optimization for Internet of things applications in edge computing [J]. ZTE Communications, 2022, 21(2): 40–52. DOI: 10.12142/ZTECOM.202302007

[2] CAI W B, YANG S L, SUN G, et al. Adaptive load balancing for parameter servers in distributed machine learning over heterogeneous networks [J]. ZTE Communications, 2023, 21(1): 72–80. DOI: 10.12142/ZTECOM.202301009

[3] ZHAO Z P, ZHAO Y L, YAN B Y, et al. Auxiliary fault location on commercial equipment based on supervised machine learning [J]. ZTE Communications, 2022, 20(S1): 7–15. DOI: 10.12142/ZTECOM.2022S1002

[4] 韩炳涛, 刘涛, 唐波. 深度学习的10年回顾与展望 [J]. 中兴通讯技术, 2022, 28(6): 75–84. DOI: 10.12142/ZTETJ.202206013

[5] 张振国, 杨倩倩, 贺诗波. 基于深度学习的图像语义通信系统 [J]. 中兴通讯技术, 2023, 29(2): 54–61. DOI: 10.12142/ZTETJ.202302011

[6] 潘囿丞, 侯永帅, 杨卿, 等. 大规模语言模型的跨云联合训练关键技术 [J]. 中兴通讯技术, 2023, 29(4): 49–56. DOI: 10.12142/ZTETJ.202304010

[7] 曾炜, 苏腾, 王晖, 等. 鹏程·盘古：大规模自回归中文预训练语言模型及应用 [J]. 中兴通讯技术, 2022, 28(2): 33–43. DOI: 10.12142/ZTETJ.202202006

[8] 韩旭, 张正彦, 刘知远. 知识指导的预训练语言模型 [J]. 中兴通讯技术, 2022, 28(2): 10–15. DOI: 10.12142/ZTETJ.202202003

[9] ZHAO W Z, ZHOU K, LI J Y, et al. A survey of large language models [EB/OL]. (2023–03–31) [2024–02–25]. https://arxiv. org/abs/2303.18223

[10] HERNANDEZ D, BROWN T, CONERLY T, et al. Scaling laws and interpretability of learning from repeated data [EB/OL]. (2022–05–21)[2024–02–25]. https://arxiv.org/abs/2205.10487

[11] LEE K, IPPOLITO D, NYSTROM A, et al. Deduplicating training data makes language models better [EB/OL]. [2024–02–25]. http://arxiv.org/abs/2107.06499

[12] PENEDO G, MALARTIC Q, HESSLOW D, et al. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only [EB/OL]. (2023–07–01) [2024–02–25]. https://arxiv.org/abs/2306.01116

[13] LONGPRE S, YAUNEY G, REIF E, et al. A pretrainer's guide to training data: measuring the effects of data age, domain coverage, quality, &toxicity [EB/OL]. (2023–05–22) [2024–02–25]. https://arxiv.org/abs/2305.13169

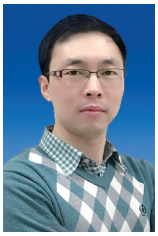
[14] GAN R, WU Z, SUN R, et al. Ziya2: data–centric learning is all LLMs need [EB/OL]. (2023–05–22) [2024–02–25]. https://arxiv.org/abs/2311.03301

[15] CARLINI N, TRAMER F, WALLACE E, et al. Extracting training

data from large language models [EB/OL]. (2020-12-14)[2024-02-25]. https://arxiv.org/abs/2012.07805
[16] JANG J, YOON D, YANG S, et al. Knowledge unlearning for mitigating privacy risks in language models [EB/OL]. (2022-10-04)[2024-02-25]. https://arxiv.org/abs/2210.01504
[17] CHEN W J, HE S B, XU Y W, et al. iCache: an importance-sampling-informed cache for accelerating I/O-bound DNN model training [C]// /Proceedings of 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2023. DOI: 10.1109/HPCA56546.2023.10070964
[18] CHILIMBI T, SUZUE Y, APACIBLE J, et al. Project Adam: building an efficient and scalable deep learning training system [C]//Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation. ACM, 2014: 571-582. DOI: 10.5555/2685048.2685094
[19] HASHEMI S H, JYOTHI S A, CAMPBELL R H. icTac: accelerating distributed deep learning with communication scheduling [EB/OL]. (2018-05-08) [2024-02-25]. https://arxiv. org/abs/1803.03288
[20] NVIDIA. 2020. Fast AI data preprocessing with NVIDIA DALI [EB/OL]. [2024-02-25]. https://devblogs. nvidia. com/fast-ai-data-preprocessing-with-nvidia-dali/
[21] CHEN T Q, XU B, ZHANG C Y, et al. Training deep nets with sublinear memory cost [EB/OL]. [2024-02-25]. http://arxiv. org/abs/1604.06174
[22] MOHAN J, PHANISHAYEE A, RANIWALA A, et al. Analyzing and mitigating data stalls in DNN training [J]. Proceedings of the VLDB endowment, 2021, 14(5): 771-784. DOI: 10.14778/3446095.3446100
[23] KAPLAN J, MCCANDLISH S, HENIGHAN T, et al. Scaling laws for neural language models [EB/OL]. [2024-02-25]. http://arxiv. org/abs/2001.08361
[24] HOFFMANN J, BORGEAUD S, MENSCH A, at al. Training compute-optimal large language models [EB/OL]. (2022-05-29) [2024-02-25]. https://arxiv.org/abs/2203.15556
[25] XU Q, LI C, GONG C, et al. An efficient 2D method for training super-large deep learning models [EB/OL]. (2021-04-12)[2024-02-25]. https://arxiv.org/abs/2104.05343
[26] ZHANG B, TITOV I, SENNRICH R. Improving deep transformer with depth-scaled initialization and merged attention [C]// Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics. DOI: 10.18653/v1/d19-1083
[27] HUANG X S, PÉREZ F, BA J, et al. Improving transformer optimization through better initialization [C]//Proceedings of the 37th International Conference on Machine Learning. ACM, 2020: 4475-4483. DOI: 10.5555/3524938.3525354
[28] BA J L, KIROS J R, HINTON G E. Layer normalization [EB/OL]. (2016-07-21)[2024-02-25]. https://arxiv.org/abs/1607.06450
[29] XIONG R, YANG Y, HE D, et al. On layer normalization in the transformer architecture [EB/OL]. (2020-02-12) [2024-02-25]. https://arxiv.org/abs/2002.04745
[30] ZHANG B, SENNRICH R. Root mean square layernormalization [EB/OL]. (2019-10-16) [2024-02-25]. https://arxiv. org/abs/1910.07467
[31] HENDRYCKS D, GIMPEL K. Gaussian error linear units (GELUs) [EB/OL]. [2024-02-25]. http://arxiv.org/abs/1606.08415
[32] SHAZEER N. GLU variants improve transformer [EB/OL]. (2020-02-12)[2024-02-25]. http://arxiv.org/abs/2002.05202
[33] DAUPHIN Y N, FAN A, AULI M, et al. Language modeling with gated convolutional networks [EB/OL]. (2016-12-23)[2024-02-25]. https://arxiv.org/abs/1612.08083
[34] NENKOVA A, PASSONNEAU R, MCKEOWN K. The pyramid method: incorporating human content selection variation in summarization evaluation [J]. ACM transactions on speech and language processing, 4(2): 4-es. DOI: 10.1145/1233912.1233913
[35] ZAN D G, CHEN B, ZHANG F J, et al. Large language models meet NL2Code: a survey [EB/OL].[2024-02-25]. http://arxiv.org/abs/2212.09420
[36] YU J F, WANG X Z, TU S Q, et al. KoLA: carefully benchmarking world knowledge of large language models [EB/OL]. [2024-02-25]. http://arxiv.org/abs/2306.09296
[37] DHOLE K D, GANGAL V, GEHRMANN S, et al. NL-augmenter: a framework for task-sensitive natural language augmentation [EB/OL]. (2021-12-06) [2024-02-25]. http://arxiv. org/abs/2112.02721
[38] BORDIA S, BOWMAN S R. Identifying and reducing gender bias in word-level language models [EB/OL]. (2019-04-05) [2024-02-25]. http://arxiv.org/abs/1904.03035
[39] LEES A, TRAN V Q, TAY Y, et al. A new generation of perspective API: efficient multilingual character-level transformers [C]//Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 2022: 3197 - 3207. DOI: 10.1145/3534678.3539147
[40] HASHIMOTO T, ZHANG H, LIANG P. Unifying human and statistical evaluation for natural language generation [C]// Proceedings of the 2019 Conference of the North. Association for Computational Linguistics, 2019. DOI: 10.18653/v1/n19-1169
[41] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-LM: training multi-billion parameter language models using GPU model parallelism [EB/OL]. (2019-09-17) [2024-02-25]. http://arxiv.org/abs/1909.08053
[42] LI S G, LIU H X, BIAN Z D. Colossal-AI: a unified deep learning system for large-scale parallel training [EB/OL]. (2021-10-28) [2024-02-25]. https://arxiv.org/abs/2110.14883
[43] HUANG Y, CHENG Y, BAPNA A, et al. Gpipe: efficient training of giant neu-ral networks using pipeline parallelism [EB/OL]. (2019-09-17)[2024-02-25]. https://arxiv.org/abs/1811.06965
[44] HARLAP A, NARAYANAN D, PHANISHAYEE A, et al. Pipedream: fast and efficient pipeline parallel DNN training [EB/OL]. (2018-06-08) [2024-02-25]. https://arxiv. org/abs/1811.06965
[45] NARAYANAN D, PHANISHAYEE A, SHI K, et al. Memory-efficient pipeline-parallel DNN training [EB/OL]. (2019-09-17) [2024-02-25]. https://arxiv.org/abs/2006.09503
[46] LI S G, XUE F Z, BARANWAL C, et al. Sequence parallelism: long sequence training from system perspective [EB/OL]. (2021-05-26)[2024-02-25]. http://arxiv.org/abs/2105.13120
[47] RASLEY J, RAJBHANDARI S, RUWASE O, et al. DeepSpeed: system optimizations enable training deep learning models with over 100 billion parameters [C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2020. DOI: 10.1145/3394486.3406703
[48] LEPIKHIN D, LEE H, XU Y Z, et al. GShard: scaling giant models with conditional computation and automatic sharding [EB/OL]. (2020-07-30)[2024-02-25]. http://arxiv.org/abs/2006.16668
[49] WILLIAM F, ZOPH B, SHAZEER M. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity [EB/OL]. (2021-01-11) [2024-02-25]. https://arxiv. org/abs/2101.03961

[50] RAJBHANDARI S, LI C L, YAO Z W, et al. DeepSpeed-MoE: advancing mixture-of-experts inference and training to power next-generation AI scale [EB/OL]. [2024-02-25]. http://arxiv.org/abs/2201.05596
[51] HE J A, ZHAI J D, ANTUNES T, et al. FasterMoE: modeling and optimizing training of large-scale dynamic pre-trained models [C]//Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. ACM, 2022. DOI: 10.1145/3503221.3508418
[52] JIA Z H, ZAHARIA M, AIKEN A. Beyond data and model parallelism for deep neural networks [EB/OL]. [2024-02-25]. http://arxiv.org/abs/1807.05358
[53] ZHENG L M, LI Z H, ZHANG H, et al. Alpa: automating inter-and intra-operator parallelism for distributed deep learning [EB/OL]. (2022-01-28)[2024-02-25]. https://arxiv.org/abs/2201.12023
[54] 王卫斌, 周建锋, 黄兵. ODICT融合的网络2030 [J]. 中兴通讯技术, 2022, 28(1): 47-56. DOI: 10.12142/ZTETJ.202201011
[55] KIM J, DALLY W J, SCOTT S, et al. Technology-driven, highly-scalable dragonfly topology [EB/OL]. [2024-02-25]. https://pages.cs.wisc.edu/~markhill/restricted/isca08_dragonfly.pdf
[56] WANG W Y, GHOBADI M, SHAKERI K, et al. How to build low-cost networks for large language models (without sacrificing performance)? [EB/OL]. (2023-07-22)[2024-02-25]. https://doi.org/10.48550/arxiv.2307.12169
[57] GRAHAM R L, LEVI L, BURREDY D, et al. Scalable hierarchical aggregation and reduction protocol (SHARP)TM streaming-aggregation hardware design and evaluation[C]//International Conference on High Performance Computing. Springer, 2020: 41-59. DOI: 10.1007/978-3-030-50743-5_3
[58] SAPIO A, CANINI M, HO C Y, et al. Scaling distributed machine learning with In-network aggregation [EB/OL]. [2024-02-25]. http://arxiv.org/abs/1903.06701
[59] MOHAN J, PHANISHAYEE A, CHIDAMBARAM V. CheckFreq: fre-quent, fine-grained DNN checkpointing [EB/OL]. [2024-02-25]. https://www. microsoft. com/en-us/research/uploads/prod/2020/12/checkfreq-fast21.pdf
[60] NICOLAE B, LI J L, WOZNIAK J M, et al. DeepFreeze: towards scalable asynchronous checkpointing of deep learning models [C]//Proceedings of 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID). IEEE, 2020: 172-181. DOI: 10.1109/CCGrid49817.2020.00-76
[61] WANG Z, JIA Z, ZHENG S, et al. GEMINI: fast failure recovery in distributed training with In-memory checkpoints [C]// Proceedings of the 29th Symposium on Operating Systems Principles. ACM, 2023: 364-381. DOI: 10.1145/3600006.3613145
[62] MAURYA A, NICOLAE B, RAFIQUE M M, et al. Towards efficient I/O scheduling for collaborative multi-level checkpointing [C]//Proceedings of 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2021: 1-8. DOI: 10.1109/MASCOTS53633.2021.9614284
[63] LIAO J J, LI M Z, SUN Q X, et al. Mimose: an input-aware checkpointing planner for efficient training on GPU [EB/OL]. (2022-09-06)[2024-02-25]. https://arxiv.org/abs/2209.02478

作 者 简 介



田海东，中兴通讯股份有限公司先进计算存储架构师、项目经理；主要从事计算存储系统架构演进、近数据处理、存储体系端侧优化等方向的研究。

张明政，中兴通讯股份有限公司先进计算存储算法工程师；主要从事人工智能深度学习、感算融合等方向的算法研究。

常锐，中兴通讯股份有限公司先进计算存储架构师；主要从事大模型训推系统、数据流支撑平台、安全可信计算等方向的研究。

童贤慧，中兴通讯股份有限公司先进计算存储系统工程师；主要从事数据预处理、安全可信计算、对等系统等方向的研究。