arXiv:2303.01277v1 [cs.DC] 2 Mar 2023

通过异步单比特通信加速分布式全图 GNN 训练

Meng Zhang [1] **Qinghao Hu** [1 2] **Peng Sun** [2 3] **Yonggang Wen** [1] **Tianwei Zhang** [1]

摘要

由于高内存需求与有限的 GPU 内存之间存在矛盾，在大型图上训练图神经网络（GNN）颇具挑战。近年来，分布式全图 GNN 训练已被广泛用于解决这一问题。然而，大量的 GPU 间通信开销会导致吞吐量严重下降。现有的通信压缩技术主要针对传统的深度神经网络训练，其瓶颈在于梯度和参数的同步。我们发现，这些技术在分布式 GNN 训练中效果不佳，因为其瓶颈在于前向传播期间逐层传输特征，以及反向传播期间逐层传输特征梯度。为此，我们提出了一种高效的分布式 GNN 训练框架**Sylvie**，该框架在 GNN 中采用单比特量化技术，并进一步将缩减后的通信与计算进行流水线化，从而大幅降低开销，同时保持模型质量。具体而言，Sylvie 提供了一个轻量级的低比特模块，用于量化发送的数据，并在每一层将接收的数据反量化为全精度数值。此外，我们提出了一种有界陈旧性适配器，用于控制引入的陈旧性，以进一步提升性能。我们对多种模型和数据集进行了理论收敛性分析和大量实验，结果表明，Sylvie 可将训练吞吐量最高提升 28.1×。

1 引言

近年来，图神经网络已广受欢迎，并在图等结构化数据的学习任务中展现出最先进方法（SOTA）的性能。图神经网络通过消息传递操作捕捉给定图的内在依赖关系（Abadal 等人，2021）。为了更新节点，图神经网络首先聚合节点邻居的特征向量，然后将它们组合起来。尽管图神经网络在图相关任务中表现出色，但在包含数百万至数十亿个节点的大规模图上训练图神经网络仍是一个长期存在的问题，因为加载和计算输入图需要大量内存资源（Kipf 和 Welling，2016；Liu 等人，2021；Zhang 和 Chen，2018），这阻碍了更复杂的图神经网络模型在实践中的发展。

针对这一问题，现有解决方案可分为两种不 。首先，一些工作（Chen 等人，2018；Chiang 等人，2019；Huang 等人，2018；Zeng 等人，2020；Zou 等人，2019）提出了基于采样的方法，在每次迭代中仅选择部分节点和边进行训练。然而，大多数此类方法都需要集中式数据存储，这将导致高昂的数据传输成本。更重要的是，

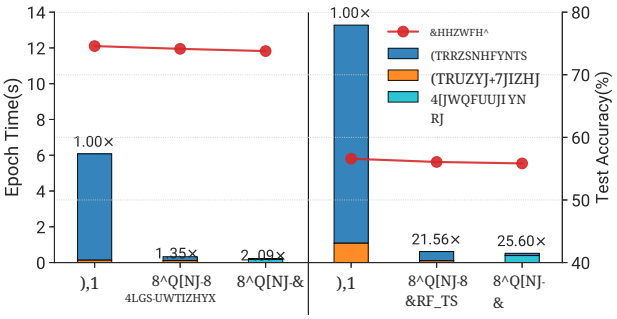


图 *1*。使用 DGL、Sylvie-S 和 Sylvie-A 在两台服务器上对两个数据集进行图卷积网络训练的轮次时间和测试准确率。Sylvie 的加速比最高可达 28.1×

这些方法会造成模型准确率损失（Hamilton 等人，2017；Jia 等人，2020；Wan 等人，2022a）。表1展示了在 Ogbn-products 数据集（Hu 等人，2020）上分别采用基于采样的模式和全图训练训练 GraphSAGE 模型时的测试准确率。基于采样的模式的准确率始终低于全图训练，尤其是在样本规模减小时。

分布式图神经网络训练（Thorpe 等人，2021；Zheng 等人，2021；Zhu 等人，2019）是另一个很有前景的方向，它在多个图形处理器或节点上进行全图训练，以减少每个图形处理器上的计算时间和内存需求。该方法首先将整个图划分为若干子图，使每个子图都能装入单个图形处理器，然后在各图形处理器上本地训练子

[1]南洋理工大学 [2]上海人工智能实验室 [3]商汤研究院。通讯作者：Tianwei Zhang <tian- wei.zhang@ntu.edu.sg>。

预印本。

表 *1*。在 Ogbn-products 上训练 GraphSAGE 时，不同层采用不同邻居采样数量的全图训练和基于采样的训练的测试准确率。

	基于采样			全图
样本规模	5	10	15	
准确率（%）	73.55	74.87	76.84	**79.19**

图。当前的图神经网络框架（如 DGL（Wang 等人，2020）和 PyG（Fey & Lenssen，2019））存在可扩展性限制，且缺乏对分布式全图训练的系统支持。此外，与经典的分布式 DNN 训练（例如使用 AlexNet 进行图像分类（Wen 等人，2017））不同，后者的训练样本彼此独立；由于子图之间存在节点依赖，因此很难将数据并行应用于图神经网络，从而不可避免地产生数据通信开销。为解决这一问题，近期的研究（Jia 等人，2020；Mostafa，2022；Tripathy 等人，2020；Wan 等人，2022a；b）提出了多种实现高效分布式全图训练的方法。然而，这些方法也存在若干缺点。以往的方法（Jia 等人，2020；Mostafa，2022；Tripathy 等人，2020）仍面临严重的通信开销，导致训练时间大幅增加。一些研究（Mostafa，2022；Wan 等人，2022a）以可能损失准确率为代价训练大规模图。此外，某些方法（Tripathy 等人，2020；Wan 等人，2022a；b）难以推广到各种图神经网络模型或数据集。

为应对上述限制，我们提出了新框架 Sylvie，在保持模型准确率的同时有效加速分布式全图 GNN 训练。Sylvie 由五个主要模块组成：(1) 图引擎 首先将原始图分区到各个图形处理器，并选择需要通信的节点。(2) 量化器 对通信特征和梯度进行压缩。(3) 通信器 在各分区之间传输数据。(4) 反量化器以误差补偿方式对接收的数据进行解压缩。(5) 训练器 最后利用通信数据执行模型的前向传播和反向传播计算。我们还提出了 Sylvie 的两个变体：(1) Sylvie-S 按顺序执行上述模块。(2) Sylvie-A 集成异步流水线技术，以进一步提升性能。我们还为 Sylvie-A 设计了有界陈旧性适配器，用于在陈旧特征和梯度影响收敛的情况下增强模型收敛性。图1 展示了在两台机器上使用 Sylvie 训练时的部分结果。Sylvie 在准确率损失较小的情况下显著缩短了训练时间。大量的通信开销（蓝色柱）被削减至极小部分，从而显著缩短了轮次时间并带来明显的加速比。在 Sylvie-S 的基础上，Sylvie-A 进一步将缩减后的通信过程与其他操作重叠，最高可实现 28。1× 倍加速。除具备多服务器设置下的可扩展性外，Sylvie 在单机上也能实现出色的性能：例如，相较于 DGL，最高可提升 9.3×。

总而言之，我们的贡献如下：

• 据我们所知，Sylvie 是首个 实现一比特宽通信量化以加速多 GPU 图神经网络训练的框架，相较于现有图神经网络框架，最高可实现 28.1× 倍加速比。

• 基于大量实验和理论分析，Sylvie 能够在准确率损失可忽略不计的情况下，良好地保持模型收敛。

• 除了空间 大小缩减外，我们还通过集成异步流水线技术（Sylvie-A），考虑了正交的时间 维度。此外，为了更好地实现模型收敛，我们设计了有界陈旧性适配器，以减少陈旧数据造成的误差。

2 背景与相关工作

2.1 图神经网络

图神经网络是一类机器学习算法，通过学习图的连通性来建模节点之间的关系。通常，每一层的迭代学习过程包含两个步骤：特征聚合和更新。考虑一个图 $\mathcal{G}=(V,\ E)$，其节点为 $V=\{v_1,\cdots,v_{|V|}\}$，边为 $E=\{e_1,\cdots,e_{|E|}\}$，节点特征矩阵为 $\boldsymbol{X}\in\mathbb{R}^{|V|\times d}$。对于任意一层 $l\in[1,L]$，聚合和更新步骤可表示为：

$$\boldsymbol{z}_v^{(l)}=\rho^{(l)}\left(\left\{\boldsymbol{h}_u^{(l-1)}\mid u\in\mathcal{N}(v)\right\}\right) \tag{1}$$

$$\boldsymbol{h}_v^{(l)}=\phi^{(l)}\left(\boldsymbol{z}_v^{(l)},\boldsymbol{h}_v^{(l-1)}\right) \tag{2}$$

其中 $\mathcal{N}(v)$ 表示节点 v的邻居节点。聚合函数 $\rho^{(l)}$ 获取邻居节点的嵌入 $\boldsymbol{h}_u^{(l-1)}$，得到中间聚合结果 $\boldsymbol{z}_v^{(l)}$，随后将其与节点 v 自身的特征向量 $\boldsymbol{h}_v^{(l-1)}$ 一同作为更新函数 $\phi^{(l)}$ 的输入，从而得到第 l $\boldsymbol{h}_v^{(l)}$层学习到的嵌入。不同的图神经网络在聚合函数和更新函数上各有不同。本文主要关注三种常用的 GNN 模型：GraphSAGE（Hamilton 等人，2017）、GCN（Kipf 和 Welling，2016）和 GAT（Veli˘ckovi´c 等人，2018），但我们的框架也很容易扩展到其他 GNN 模型。

2.2 分布式 GNN 训练

为了在完整图上进行分布式 GNN 训练，首先在主机端对整个输入图进行分区，使其能够装入单个图形处理器。由于每个节点及其特征只会分配给一个图形处理器，因此存在一些节点，它们与本地分区中的节点相连，却不属于该分区，这些节点称为**HALO 节点**。如图3所示，对于 GPU-1 上的分区，节点 4 每层都需要从 GPU-0 获取节点 7 的额外特征，并从 GPU-2 获取节点 1 的额外特征

，以更新其嵌入。在反向传播过程中，本地计算得到的特征梯度也不完整。由于每层中节点只交换其一跳邻居，因此从 2 跳到 L 跳的数据流位于其他分区上。因此，HALO 节点的特征梯度也需要在每层进行广播。由于 HALO 节点的数量可能非常庞大，这种通信开销不可忽视。此外，随着分区数量和层大小增大，这种开销也会进一步加剧。对于 GPU-1 上的节点 4，必须在第 1 层开始之前完成节点 7 和节点 1 的通信。第 1 层结束后，其余各层也严格按照顺序重复这一过程，如图4(a) 所示。在这种情况下，过多的通信会占用训练时间，并阻碍后续计算。

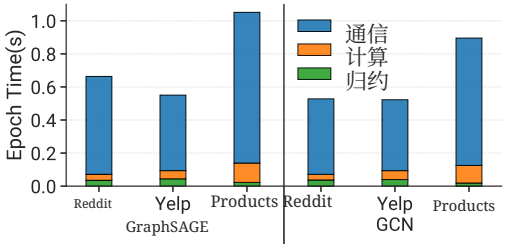


图 2。在单台服务器（8 个图形处理器）上使用 DGL 进行原始分布式图神经网络训练时，每个训练轮次的训练时间及其细分情况。

为了更直观地展示巨大的通信成本，我们对多个数据集和模型的轮次时间及其组成进行了分析，如图2所示。可以清楚地看到，在三个数据集上，无论是 GraphSAGE 还是图卷积网络，通信时间几乎占据了整个训练过程的主导地位（最高达 89.23%），而计算和梯度同步操作（全归约）仅占很小一部分。因此，过高的通信开销严重限制了分布式 GNN 训练的可扩展性和效率。

以往工作提出了新的框架来加速分布式 GNN 训练，例如 AliGraph （ Zhu 等人，2019 ）和 NeuGraph （ Ma 等人，2019）。然而，这些方法都将分区存储在中央处理器中，不可避免地导致频繁的 CPU-GPU 交换，并大幅削弱分布式训练的优势。DistDGL（Zheng 等人，2021）给出了扩展性结果，但仅针对基于采样的方法。LLCG （Ramezani 等人，2022）则完全舍弃分区之间的依赖信息，并增加一个全局校正服务器来弥补误差，由此带来冗余开销。此外，这些工作仅支持对图进行小批量训练，而不支持全图训练。

与上述基于采样的方法不同，ROC（Jia 等人，2020）加速了分布式全图训练，但它也将分区存储在中央处理器中，并面临高昂的 CPU-GPU 数据传输成本。SAR（Mostafa，2022

）为大规模全图训练节省了内存，但由于重物化，计算负担更重。BNS-GCN（Wan 等人，2022a）对边界节点采用随机采样，显著加速了训练，但丢弃节点连接可能会降低模型性能，而且其性能高度依赖图结构。CAGNET（Tripathy 等人，2020）提出了不同的维度划分方法，将节点特征切分为子向量以加速训练，但代价是增加通信和同步开销。PipeGCN（Wan 等人，2022b巧妙地隐藏了通信开销，通过对计算和通信进行流水线处理，同时保留所有边界节点信息。然而，当通信时间远大于计算部分时，其效果会大幅下降，如 §4.1 所示。

2.3 神经网络量化

模型量化通常用于加速传统的深度神经网络推理（ Jacob 等人，2018；Krishnamoorthi，2018），而我们的目标是提升训练速度。压缩也常用于减少通信开销并加速分布式 DNN 训练（Alistarh 等人，2017；Wen 等人，2017）。然而，分布式 DNN 训练的开销源于对所有梯度和参数进行同步。因此，这些方法无法直接应用于我们的场景，因为如图2所示，在图神经网络训练中，诸如全归约的梯度同步开销可以忽略不计。

近年来，一些研究（Feng 等人，2020；Zhao 等人，2020）通过模拟应用图神经网络模型量化，以缩减内存，但底层计算仍采用 32 位全精度。近期的一项工作 EXACT（Liu 等人，2021）旨在降低内存需求，但代价是增加训练时间开销，严重损害训练效率。其他研究（Tailor 等人，2021；Wang 等人，2022）则对图神经网络模型进行量化，以实现高效推理。与我们的工作相比，这些方法的目标各不相同，且仅考虑小规模数据集。

不同于上述量化工作，我们是**首个**探索在图神经网络中对通信进行量化的研究，这能够在几乎不造成准确率损失的情况下大幅降低开销。与以往工作不同，我们不对激活值或权重进行量化（Jacob 等人，2018；Krishnamoorthi，2018），原因在于：（1）计算仅占很小一部分，而嵌入和特征梯度的通信占据了训练开销的主要部分（图2）；（2）图神经网络中特有的稀疏计算，例如 cuSPARSE 中的稀疏矩阵乘法（cuS，2022），缺乏对低精度计算的支持；相比之下，DNN 中的稠密运算（例如通用矩阵乘法）支持快速低精度计算。

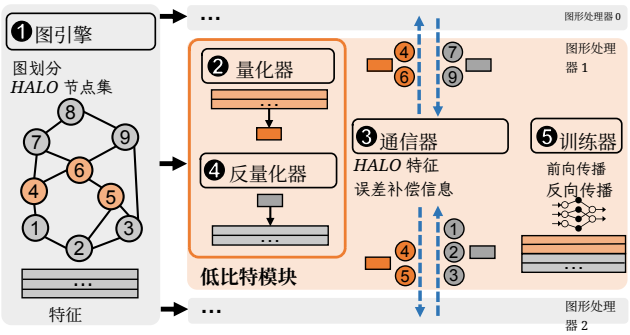


图3. Sylvie框架及其工作流程概览。我们重点关注 GPU-1 上的子图。橙色节点和矩形分别表示分配给 GPU-1 的节点及其对应特征。灰色部分则表示其他图形处理器上的节点/特征。

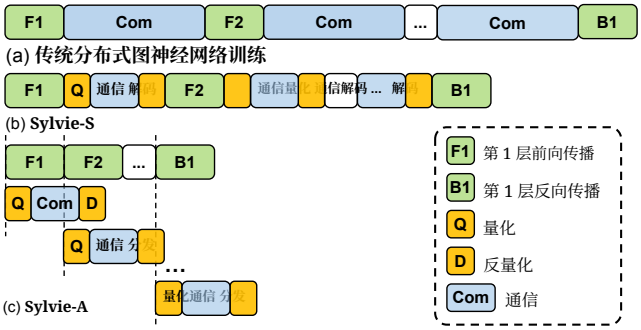


图 4. 传统分布式训练、Sylvie-S 和 Sylvie-A 的工作流程对比。Sylvie 借助低比特模块显著降低通信开销。Sylvie-A 则通过流水线进一步缩短训练时间。

3 提出的框架

3.1 概述

图3 展示了 Sylvie 的整体工作流程，共包含五个步骤。❶ 图引擎首先在主机端将输入图划分为多个子图，并为每个分区构建 HALO 节点集，以准备后续通信。随后，每个分区（包括图结构和特征）都会被分配到一块独立的图形处理器上。❷接下来，每块图形处理器中的量化器会将 HALO 节点数据量化为1位整数。例如，GPU-1 中节点 4、5 和 6 的数据会被量化，因为 GPU-0 和 GPU-1 中的其他分区需要这些数据。❸ 之后，这些量化数据以及相应的误差补偿信息（用于帮助恢复数据）会通过网络通信在各分区之间进行广播。❹ 从其他分区接收到量化数据后，反量化器会在每块图形处理器中利用误差补偿信息，将压缩数据恢复为全精度值。量化器和反量化器共同构成低比特模块，负责数据转换。❺ 这些恢复后的数据与原始数据一起，用于在前向和反向传播中训练模型。

我们介绍基于 Sylvie 的两个变体（图4）。
(1)**Sylvie-S**：所有这些模块均按顺序同步执行。
(2)**Sylvie-A**：我们在低比特模块、通信和计算之间采用异步流水线技术，以进一步提升训练性能。然而，由于陈旧特征和梯度，这种异步流水线可能会损害某些数据集或模型上的收敛性。为此，我们进一步设计了有界陈旧性适配器，每隔固定数量的训练轮次定期同步所有分区的最新数据。

3.2 Sylvie框架

图引擎。为进行分布式 GNN 训练，图引擎 首先根据可用图形处理器的数量，在主机端对图结构和特征矩阵进行分区，并构建每个分区的 HALO 节点集。随后，将每个分区加载到单独的图形处理器上，并且每个图形处理器都保存一份完整模型的副本。

量化器。不同于现有工作对所有激活值或权重进行量化，Sylvie 是首个对分布式 GNN 训练中的通信数据进行量化的方法。我们以图3中 GPU-1 上子图在前向传播过程中的示例来说明量化机制。对于模型的 l层，量化器首先根据 HALO 节点集选择需要通信的节点特征（例如节点 4、5 和 6 的特征） $\boldsymbol{S}^{(l-1)} \subseteq \boldsymbol{H}^{(l-1)}$，然后将其量化为低精度的 1位整数$\hat{\boldsymbol{S}}_{1bit}^{(l-1)}$。对于 $\boldsymbol{h}^{(l-1)}$ 中的每个节点特征 $\boldsymbol{S}^{(l-1)}$，若要将其量化为 1 位整数，我们使用以下公式：

$$\bar{\boldsymbol{h}}^{(l-1)} = \frac{\boldsymbol{h}^{(l-1)} - \min(\boldsymbol{h}^{(l-1)})}{\max(\boldsymbol{h}^{(l-1)}) - \min(\boldsymbol{h}^{(l-1)})} B \tag{3}$$

$$\hat{\boldsymbol{h}}_{1bit}^{(l-1)} = \begin{cases} \left\lceil \bar{\boldsymbol{h}}^{(l-1)} \right\rceil & \text{w.prob.} \quad \bar{\boldsymbol{h}}^{(l-1)} - \left\lfloor \bar{\boldsymbol{h}}^{(l-1)} \right\rfloor \\ \left\lfloor \bar{\boldsymbol{h}}^{(l-1)} \right\rfloor & \text{otherwise} \end{cases} \tag{4}$$

其中 $B = 2^b - 1$ 表示将数据量化为 b 位整数时的量化区间数。在我们的工作中，由于采用 1 位量化，b的默认值为 1。公式4 采用随机舍入（Courbariaux等人，2015）。

通信器。量化后，$\hat{\boldsymbol{S}}_{1bit}^{(l-1)}$（包含所有量化后的 $\hat{\boldsymbol{h}}_{1bit}^{(l-1)}$）会通过通信器广播给相应的图形处理器。此外，通信器还会随量化数据发送误差补偿信息，例如缩放因子（$\max(\boldsymbol{h}^{(l-1)}) - \min(\boldsymbol{h}^{(l-1)})$ 与 B之间的比值）和零点 $\min(\boldsymbol{h}^{(l-1)})$，以便更准确地无偏恢复数据。如表 3所示，与传输的嵌入或梯度的主数据相比，误差补偿数据的大小要小得多，因此产生的通信开销可以忽略不计。

反量化器。由通信器管理，每个分区本身都会接收压缩特征 $\hat{\boldsymbol{R}}_{1bit}^{(l-1)}$ 以及来自其他分区的误差补偿信息。然后

反量化器 借助误差补偿信息，将它们反量化为 32 位全精度浮点值 $\tilde{\boldsymbol{R}}^{(l-1)}$，以尽可能确保恢复后的特征与原始特征保持一致。$\hat{\boldsymbol{R}}_{1bit}^{(l-1)}$ 中每个嵌入 $\hat{\boldsymbol{h}}_{1bit}^{(l-1)}$ 的反量化过程表示为：

$$\tilde{\boldsymbol{h}}^{(l-1)} = \frac{(\max(\boldsymbol{h}^{(l-1)}) - \min(\boldsymbol{h}^{(l-1)}))\hat{\boldsymbol{h}}_{1bit}^{(l-1)}}{B} + \min(\boldsymbol{h}^{(l-1)}) \tag{5}$$

其中，恢复后的 $\tilde{\boldsymbol{R}}^{(l-1)}$ 包含所有反量化后的 $\tilde{\boldsymbol{h}}^{(l-1)}$。

训练器。使用 $\tilde{\boldsymbol{R}}^{(l-1)}$ 与 $\boldsymbol{H}^{(l-1)}$（其已存储在此分区中），节点嵌入 $\boldsymbol{H}^{(l)}$ 的 l层根据方程1 和 2 由训练器更新。

反向传播的训练过程与上述过程类似，不同之处在于通信的数据是特征梯度，并且 训练器 在接收到的数据反量化后逐步计算梯度。图4(b) 展示了上述 Sylvie-S 在各分区中的工作流程。与图4(a) 所示的传统分布式训练相比，该方法大幅降低了通信开销。对于 Sylvie-S，每一层的计算都在通信步骤之后进行，其他分区也采用相同的流程。

3.3 异步通信与计算重叠

为进一步缩短每轮训练时间并降低量化和反量化操作带来的开销，我们受到 Wan 等人 (2022b) 的启发，提出了 Sylvie-A，通过将 低比特模块 和通信与 GNN 计算进行流水线化，构建在 Sylvie 之上。图4 (c) 展示了 Sylvie-A 的详细工作流程。与原始分布式训练和 Sylvie-S 不同，Sylvie-A 直接利用此分区中已有的信息开始每一层的计算。同时，低比特模块并行量化数据并通信压缩数据。因此，Sylvie-A 打破了计算与通信之间的顺序关系，实现了异步分布式 GNN 训练。请注意，在 Sylvie-A 中，当前重叠进行的通信将在下一训练轮次中使用，从而确保计算开始时的数据完整性。

有界陈旧性适配器。异步化可能引入陈旧嵌入和特征梯度，从而影响模型收敛。尽管如 §4.2所示，对于大多数数据集或模型，模型收敛对陈旧性的影响具有鲁棒性，但在其他情况下，减轻其对收敛的不利影响仍然很重要。Sylvie-A 实现了一种简单而有效的有界陈旧性适配器，定期强制同步嵌入或梯度，以控制陈旧性带来的误差。尽管有界陈旧性已在传统分布式机器学习工作中已有相关研究（Xian 等人，2021），旨在改善随机梯度下降收敛，但针对分布式 GNN 训练中陈旧性的限制措施仍然鲜见。具体而言，该模块每隔固定的训练轮次间隔 ϵ_s 执行一次顺序训练，以同步最新的嵌入或梯度，其中 ϵ_s 可由用户设定，以权衡训练吞吐量与收敛速度。在其余训练轮次中，训练仍采用流水线模式。启用有界陈旧性适配器的 Sylvie-A 记为 Sylvie-Aϵ_s。图8 中的实验结果表明，有界陈旧性适配器能够显著提升异步流水线技术下的收敛速度。

3.4 使用 Sylvie 进行 GNN 训练

我们详细介绍 Sylvie-S 如何在前向和反向传播过程中应用于图神经网络。

前向传播。每个分区中 Sylvie-S 的前向传播过程如算法所示。1。初始节点嵌入设为特征向量（第 2 行）。首先，我们获取 HALO 节点集，以确保后续能够正确分发节点。在 T 个训练轮次中，每个 GNN 层内，顶点使用上一层的一跳邻居嵌入来更新自身的嵌入，其中可能包括其嵌入存储在不同分区中的 HALO 节点。这些嵌入由各自分区中的量化器 进行量化（第 10 行），通过网络通信进行分发（第 11 行），然后由当前分区中的反量化器 进行反量化（第 12 行）。恢复后的嵌入与最初存储在此图形处理器中的嵌入拼接起来，形成 $\tilde{\boldsymbol{H}}_n^{(l-1)}$以更新嵌入。算法1 中的第 15 行是公式1和2的矩阵形式，其中 $\boldsymbol{A}_n = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}_n + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$是归一化邻接矩阵， $\boldsymbol{D}$ 是度矩阵，而 $\boldsymbol{I}$ 是单位矩阵。

反向传播。算法2 展示了 Sylvie-S 的反向传播过程。在前向阶段，用于更新的特征嵌入包含已存储的嵌入以及从其他分区通信而来的嵌入。因此，在反向传播过程中，本地计算出的特征梯度并不完整，因为本地无法获得超过 1 跳的数据流。因此，从第 L 层到第 1 层，特征梯度 $\boldsymbol{J}_n^{(l)}$也需要在分区之间进行通信。与前向阶段类似，量化器和反量化器负责对特征梯度进行量化和反量化（第 10 行和第 12 行）。整合梯度后（第 13 行），即可正确计算权重（第 17 行）。

3.5 理论分析

在本节中，我们研究 Sylvie 的收敛性。我们从理论上分析低比特模块 的无偏特性以及其引入的有限噪声，因此模型的准确率

算法 1 Sylvie-S 的前向阶段

输入 分区标识 n、子图 $\mathcal{G}_n$、特征矩阵 $\boldsymbol{X}_n$、标签 Y_n，邻接矩阵 $\boldsymbol{A}_n$、训练轮数 T、层大小 L、局部节点集 V_n、权重 $\boldsymbol{W}^{(l-1)}$

分区 $n = 1, 2, ..., N$ 并行执行:
  $\boldsymbol{H}_n^{(0)} = \boldsymbol{X}_n$ ▷初始化节点嵌入
  $V_{\text{HALO},\ n} = \{$节点 $v \in \mathcal{G}_n :\ v \notin V_n\}$
  分发 $V_{\text{HALO},\ n}$ 并接收 $[V_{\text{HALO},1}, ..., V_{\text{HALO},N}]$
  分发 V_n 并接收 $[V_1, ..., V_N]$
  $[S_1, ..., S_N] = [V_n \cap V_{\text{HALO},1}, ..., V_n \cap V_{\text{HALO},N}]$: :
  $[R_1, ..., R_N] = [V_{\text{HALO},n} \cap V_1, ..., V_{\text{HALO},n} \cap V_N]$
  对于从 1 **到** T **的** t，**执行**
    对于从 1 **到** L **的** l，**执行**
      $\hat{\boldsymbol{S}}_{1bit}^{(l-1)} =$量化($[\boldsymbol{H}_n^{(l-1)}(S_1)$，$...\ \boldsymbol{H}_n^{(l-1)}(S_N)]$)
      将$\hat{\boldsymbol{S}}_{1bit}^{(l-1)}$发送至分区 1，$...\ N$。接收 $\boldsymbol{R}_{1bit}^{(l-1)}$ 来自分区 1，$.....\ N$
      $\tilde{\boldsymbol{R}}^{(l-1)} =$反量化($\hat{\boldsymbol{R}}_{1bit}^{(l-1)}$)
      $\tilde{\boldsymbol{H}}_n^{(l-1)} =$拼接($\boldsymbol{H}_n^{(l-1)}$，$\tilde{\boldsymbol{R}}^{(l-1)}$)
      如果 $l \neq L$ **则**
        $\boldsymbol{H}_n^{(l)} = \sigma\left(\boldsymbol{A}_n^\top \tilde{\boldsymbol{H}}_n^{(l-1)} \boldsymbol{W}_{t-1}^{(l)}\right)$ ▷更新嵌入
      else
        $\boldsymbol{H}_n^{(l)} = \boldsymbol{A}_n^\top \tilde{\boldsymbol{H}}_n^{(l-1)} \boldsymbol{W}_{t-1}^{(l)}$
      结束 if
    结束 for
    $\mathcal{L} =$损失(softmax($\boldsymbol{H}_n^{(L)}$) Y_n)
  结束 for

算法 2 Sylvie-S 的反向阶段

输入 分区标识 n、标签 Y_n、邻接矩阵 $\boldsymbol{A}_n$、训练轮次数量-量 T、层大小 L、局部节点集 V_n、权重 $\boldsymbol{W}^{(l-1)}$

分区 $n = 1, 2, ..., N$ 并行执行:
  对于 t **从** 1 **到** T **执行**
    对于 l **从** L **到** 1 **执行**
      如果 $l = L$ **则**
        $\boldsymbol{J}_n^{(L)} = \nabla_{\boldsymbol{H}_n^{(L)}} \mathcal{L}$
      结束如果
      $\boldsymbol{G}_n^{(l)} = \left[\boldsymbol{A}_n \tilde{\boldsymbol{H}}_n^{(l-1)}\right]^\top \left(\boldsymbol{J}_n^{(l)} \circ \sigma'\left(\boldsymbol{A}_n \tilde{\boldsymbol{H}}_n^{(l-1)} \boldsymbol{W}_{t-1}^{(l)}\right)\right)$ ▷计算权重梯度
      如果 $l > 1$ **则**
        $\boldsymbol{J}_n^{(l-1)} = \boldsymbol{A}_n^\top \left(\boldsymbol{J}_n^{(l)} \circ \sigma'\left(\boldsymbol{A}_n \tilde{\boldsymbol{H}}_n^{(l-1)} \boldsymbol{W}_{t-1}^{(l)}\right)\right)\left[\boldsymbol{W}_{t-1}^{(l)}\right]^\top$ ▷计算特征梯度
        $\hat{\boldsymbol{S}}_{1bit}^{(l-1)} =$量化($[\boldsymbol{J}_n^{(l-1)}(R_1)$ $, ..., \boldsymbol{J}_n^{(l-1)}(R_N)]$)
        发送 $\hat{\boldsymbol{S}}_{1bit}^{(l-1)}$ 至分区 $1, ..., N$。 接收 $\hat{\boldsymbol{R}}_{1bit}^{(l-1)}$ 来自分区 $1, ..., N$
        $\tilde{\boldsymbol{R}}^{(l-1)} =$反量化($\hat{\boldsymbol{R}}_{1bit}^{(l-1)}$)
        $\boldsymbol{J}_n^{(l-1)} = \boldsymbol{J}_n^{(l-1)} + \tilde{\boldsymbol{R}}^{(l-1)}$ ▷整合特征梯度
      结束 if
    结束 for
    $\boldsymbol{G} =$全归约($\boldsymbol{G}_n$) ▷梯度同步
    $\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \eta \boldsymbol{G}$ ▷更新模型
  结束循环

Sylvie 上的准确性和收敛性可以得到良好保持。

无偏低比特模块。以下定理刻画了量化的无偏特性，参考自 ActNN（Chen 等人，2021）。

定理 1（无偏嵌入）假设 $\bar{\boldsymbol{h}}^{(l)} - \lfloor \bar{\boldsymbol{h}}^{(l)} \rfloor \sim \mathcal{U}(0,1)$，$D$ 是图神经网络层的隐藏维度，量化嵌入和反量化嵌入是无偏的。

$$\mathbb{E}\left[\tilde{\boldsymbol{h}}^{(l)}\right] = \mathbb{E}[\text{Dequantize}(\text{Quantize}(\boldsymbol{h}^{(l)}))] = \boldsymbol{h}^{(l)} \tag{6}$$

$$\text{Var}(\tilde{\boldsymbol{h}}^{(l)}) = \frac{D\left[\max(\boldsymbol{h}^{(l)}) - \min(\boldsymbol{h}^{(l)})\right]^2}{6B^2} \tag{7}$$

定理 2（无偏梯度）存在随机量化策略$\mathbf{C}$，可生成无偏权重梯度。

$$\mathbb{E}\left[\hat{\boldsymbol{G}}^{(l)}\right] = \boldsymbol{G}^{(l)} \tag{8}$$

$$\text{Var}[\hat{\boldsymbol{G}}^{(l)}] = \text{Var}\left[\boldsymbol{G}^{(l)}\right] + \sum_{m=l}^{L} \mathbb{E}\left[\text{Var}\left[\mathbf{B}^{(l\sim m)}\left(\hat{\boldsymbol{J}}^{(m)}, \hat{\mathbf{C}}^{(m)}\right) \mid \hat{\boldsymbol{J}}^{(m)}\right]\right] \tag{9}$$

其中 $\mathbf{B}^{(l)}$ 是第 l层的反向传播函数。

这两个定理证明，量化和反量化操作是无偏的，因此随后计算出的权重和梯度。此外，Sylvie 中这两种操作引入的噪声有限，因此可以很好地保持模型质量。公式7 和9 表明，量化会给训练数据带来一些额外噪声，且噪声与比特数成反比。尽管噪声会逐层累积，导致典型卷积神经网络的准确率大幅下降（通常为 >5%）（Chen 等人，2021），但我们发现它对图神经网络的影响可以忽略不计。图神经网络的层数（通常为 2 到 4 层）远少于卷积神经网络，因此图神经网络比卷积神经网络更能容忍噪声（Liu 等人，2021）。因此，量化适用于分布式 GNN 训练。表2 还表明，实际应用中，引入的噪声对准确率的影响可以忽略不计。

模型收敛。给定无偏梯度，我们可以证明 Sylvie 的收敛性。假设我们采用形式为 $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \boldsymbol{G}$ 的常见随机梯度下降，从初始权重 $\boldsymbol{W}_1$开始。我们作出以下假设:

- The loss $\mathcal{L}$ is continuous differentiable and $\nabla \mathcal{L}(\boldsymbol{W})$ is β-Lipschitz continuous (Lip, 2022).
- $\mathcal{L}$ is bounded below by $\mathcal{L}_{inf}$.
- There exists $\sigma^2 > 0$, such that $\forall \boldsymbol{W}, \text{Var}\left[\hat{\boldsymbol{G}}\right] \leq \sigma^2$, where for any vector $\mathbf{x}, \text{Var}[\mathbf{x}] := \mathbb{E}\|\mathbf{x}\|^2 - \|\mathbb{E}[\mathbf{x}]\|^2$.

那么，我们可以得到以下收敛定理，该定理引自 Bottou 等人(2018)。

定理 3（收敛）如果 $0 < \eta \leq \frac{1}{\beta}$，对于迭代 t 在

$\{1, ..., T\}$，其中 T 是最大迭代次数，我们有

$$\mathbb{E}\left\|\nabla \mathcal{L}\left(\boldsymbol{W}_t\right)\right\|^2 \leq \frac{2\left(\mathcal{L}\left(\boldsymbol{W}_1\right)-\mathcal{L}_{inf}\right)}{\eta T}+\eta \beta \sigma^2 \quad (10)$$

公式10 的第一项会随着迭代次数 T 趋于无穷而收敛至零。因此，该算法会收敛到驻点的邻域内，其半径由梯度方差控制。

此外，与大多数压缩方法（Chen 等人，2021；Liu 等人，2021）不同，Sylvie 在每个训练轮次中仅对部分嵌入和梯度（即通信所需的部分）进行量化，从而使无偏梯度能够通过权重传播。这控制了噪声量，因此可以采用极端的 1 位压缩，同时将准确率损失控制在较低水平。现有研究中也有类似技术（Dong 等人，2017；Stock 等人，2020；Tailor 等人，2021），这些研究采用了子集量化。

4 实验

我们首先在多节点和单节点设置下，将 Sylvie 与其他分布式全图 GNN 训练方法进行比较（§4.1）。然后，我们展示 Sylvie 在不同数据集和模型上的收敛情况（§4.2）。为探究量化对性能的影响，我们使用不同位宽进行量化，并评估准确率和训练时间（§4.3）。最后，我们分析 Sylvie 的开销（§4.4）。

数据集和模型。我们在四个真实世界的大规模图基准数据集上评估 Sylvie：Reddit（Hamilton 等人，2017）、Yelp（Zeng 等人，2020）、Ogbn-products（Hu 等人，2020）和 Amazon（He 和 McAuley，2016）。我们选择了三种常用的 GNN 模型，这些模型通常用于评估 GNN训练：原生 GCN （ Kipf 和 Welling，2016 ）、 GraphSAGE （ Hamilton 等人，2017）和 GAT（Veli˘ckovi´c 等人，2018）。

基线方法。对于基线方法，我们将 Sylvie 与四种最先进的分布式全图训练方法进行比较：（1）DGL（Wang 等人，2020）：基于最新 DGL 0.9 的标准分布式 GNN训练；（2）SAR（Mostafa，2022）；（3）PipeGCN（Wan 等人，2022b）；（4）BNS-GCN（Wan 等人，2022a）：p 其值设为 0。1，如论文所建议。$p = 0$ 由于其测试准确率最低、收敛速度最慢且存在严重过拟合，因此并不实用。基线方法在分布式 GNN 系统设计方面彼此正交，因此我们可以进行公平比较。

测试平台。我们在流行的 GNN 训练库 DGL 0.9（Wang 等人，2020）和 PyTorch 1.10（Paszke 等人，2019）的最新稳定版本上实现了 Sylvie。所有实验均使用配备 8 个图形处理器（英伟达 RTX 3090，每个配有 24GB GDDR6X 内存）、一颗 32 线程中央处理器（AMD Threadripper PRO 3955WX）和 192GB DDR4 内存的机器。服务器内通信（中央处理器与图形处理器之间，以及图形处理器之间）通过 PCIe 4.0 通道实现。

4.1 性能评估

多服务器上的性能。如今，在多台机器上训练模型已逐渐成为常规需求。图5 展示了 Sylvie 与最先进的基线方法在两台机器上、三个模型和四个数据集上的吞吐量对比。这里的吞吐量定义为每秒运行的训练轮次数，并将各方法的吞吐量相对于 DGL 进行归一化。在每项训练任务中，我们将前 10 个训练轮次视为预热阶段，仅记录此后的统计数据。可以清楚地看到，在每个数据集和模型上，Sylvie 的表现都大幅优于其他方法。具体而言，Sylvie-S 的吞吐量提升高达 12。2∼21。6×，超过 DGL，并远胜 SAR 和 PipeGCN。在基线方法中，SAR 的吞吐量最低，因为它没有解决通信开销问题，而且由于重新物化，其计算负担甚至有所增加。Sylvie-S 的吞吐量也比 BNS-GCN 高 1。4∼2。3× 。

Sylvie-A 进一步提升了训练性能，其加速比分别达到 DGL 的 15.2∼28.1× 倍和 BNS-GCN 的 1.8∼3.3× 倍。此外，我们注意到，PipeGCN 的性能与 DGL 相近，因为在多服务器训练中，通信成本远高于计算成本。在这种情况下，通信几乎无法被隐藏，因此性能提升微乎其微。

我们在表 2 中展示了详细的归一化训练吞吐量和测试准确率。在多服务器设置下，Sylvie 的训练吞吐量始终远高于其他方法，体现了其在大规模分布式训练中的有效性。此外，Sylvie 的测试准确率仅有轻微下降。相比之下，BNS-GCN 在图注意力网络上相较于 DGL 的准确率损失最高达 4.9%，表明其对其他模型的通用性有限。由于传输的嵌入和梯度存在陈旧问题，Sylvie-A 的模型准确率在 $-1.24\%\sim+1.25\%$ 之间适度波动。

此外，图6分别展示了 Sylvie 在双服务器和三服务器系统中的训练吞吐量。我们可以观察到，随着服务器数量增加，Sylvie 仍能保持出色的性能，甚至实现更高的吞吐量加速比。在这两种设置下，与其他方法相比，Sylvie 都提供了最佳的训练加速，而 SAR 和 PipeGCN 在大规模训练中的性能则非常有限。总而言之，Sylvie 能够为更大规模的训练场景提供理想的性能。

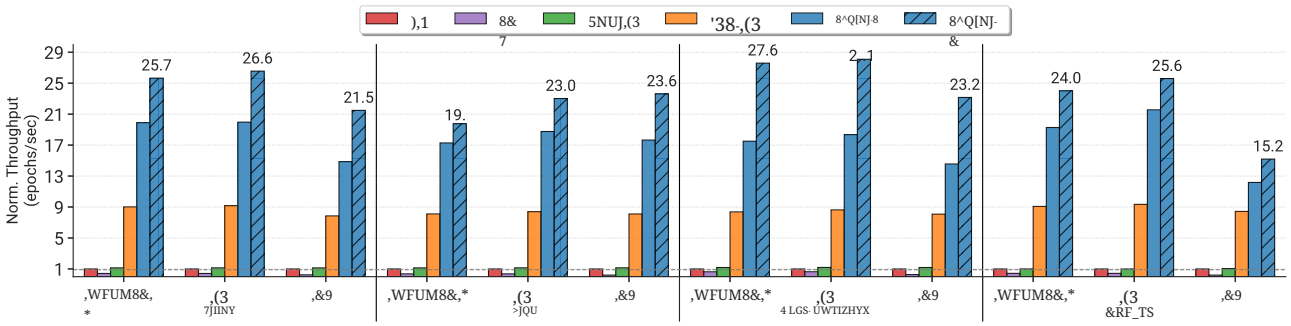


图 *5.* 在两台服务器上使用四个数据集训练三个具有代表性的模型时，不同方法的训练吞吐量（相对于 DGL 归一化，虚线所示）。Sylvie 的表现最高可比 DGL 提升 28.1×。

表 *2.* 在两台机器上训练时，Sylvie 与其他基线方法的训练吞吐量和测试准确率详细对比，其中最佳性能以粗体标出。在所有模型和数据集上，Sylvie 的吞吐量始终优于其他方法。

模型	方法	Reddit		Yelp		Ogbn-products		Amazon	
		Thr.	测试准确率（%）	Thr.	微平均 F1(%)	Thr.	测试准确率（%）	Thr.	测试准确率（%）
GraphSAGE	DGL	1.00×	97.10±0.01	1.00×	65.07±0.19	1.00×	79.19±0.15	1.00×	**81.29**±0.02
	SAR	0.42×	96.02±0.12	0.37×	60.51±0.09	0.64×	74.42±0.07	0.43×	78。85±0。07
	PipeGCN	1.15×	97.02±0.11	1.15×	65.14±0.08	1.19×	**79.29**±0.05	1.05×	81。27±0.08
	BNS-GCN	9.02×	97.14±0.01	8.11×	**65.22**±0.23	8.38×	79.11±0.11	9.08×	80。90±0.05
	Sylvie-S	19.90×	**97.15**±0.11	17.29×	65.07±0.23	17.51×	78.86±0.17	19.27×	81.22±0.07
	Sylvie-A	**25.66**×	96.87±0.03	**19.76**×	64.92±0.38	**27.59**×	78.85±0.56	**24.01**×	81.24±0.11
GCN	DGL	1.00×	94.84±0.58	1.00×	47.50±0.07	1.00×	**74.58**±0.20	1.00×	**56.59**±0.11
	SAR	0.42×	95.34±0.17	0.38×	47.00±0.12	0.65×	70.13±0.10	0.43×	53.08±0.07
	PipeGCN	1.15×	94.69±0.56	1.16×	47.16±0.01	1.20×	74.04±0.23	1.01×	56.56±0.34
	BNS-GCN	9.18×	95.00±0.33	8.40×	47.27±0.37	8.64×	73.54±0.42	9.34×	56.47±0.60
	Sylvie-S	19.97×	**95.49**±0.04	18.76×	**48.77**±0.14	18.35×	74.14±0.49	21.56×	56.07±0.07
	Sylvie-A	**26.56**×	95。31±0.01	**23.02**×	47.62±0.30	**28.09**×	73.78±0.19	**25.60**×	55.84±0.21
GAT	DGL	1.00×	93.97±0.60	1.00×	44.39±0.16	1.00×	78。14±0.12	1.00×	42。84±0.96
	SAR	0.25×	91.47±0.08	0.21×	44.30±0.11	0.27×	76.40±0.06	0.21×	42.48±0.07
	PipeGCN	1.14×	94.49±0.64	1.15×	43.75±0.23	1.19×	77.03±0.11	1.04×	42.37±0.07
	BNS-GCN	7.86×	89。08±0.63	8.11×	43.66±0.24	8.08×	74.07±0.92	8.43×	40.67±0.79
	Sylvie-S	14.86×	**94.55**±0.77	17.66×	**44.44**±0.62	14.57×	78.00±0.01	12.18×	**42.95**±0.16
	Sylvie-A	**21.49**×	93。40±0。62	**23.62**×	43。15±0.63	**23.16**×	**78.38**±0.18	**15.20**×	41。83±0.25

单服务器上的性能。我们还在一台配备 8 个图形处理器的机器上测试了 Sylvie 的性能。图7由于篇幅限制，展示了不同方法吞吐量的部分结果。Sylvie 在训练吞吐量方面仍优于其他方法，最高达到 7.0

× 使用 GraphSAGE 训练时相较于 DGL 在 Ogbn-products 上的加速比。与涉及更多分区、通信开销更大的多服务器设置相比，此处的加速效果相对不那么显著。这表明，训练规模越大，Sylvie 的效果越好。

通信量与时间。为证明训练加速源于通信量减少，我们记录了每个训练轮次的实际通信量以及训练时间分解，如表 3所示。我们观察到，Sylvie 大幅减少了通信量。例如，Amazon数据集原本每个训练轮次的通信量为 5632.6 MB。部署 Sylvie 后，

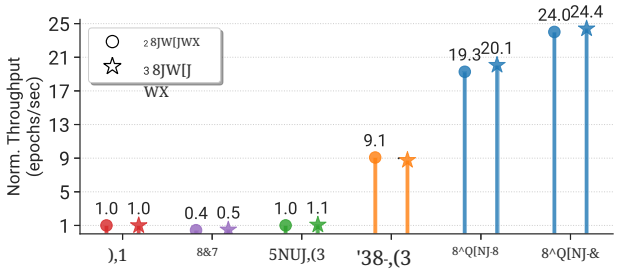


图*6*。 Amazon 上 GraphSAGE 在两台服务器和三台服务器上的归一化训练吞吐量。

传输的嵌入和梯度仅为 176.1 MB，通信量减少了近 32× 。因此，通信时间大幅缩短。时间分解结果显示，通信在每轮训练时间中占据很大比重，因此训练吞吐量也有所提升。请注意，我们还传输了部分误差补偿信息，但其占比小得多（例如，

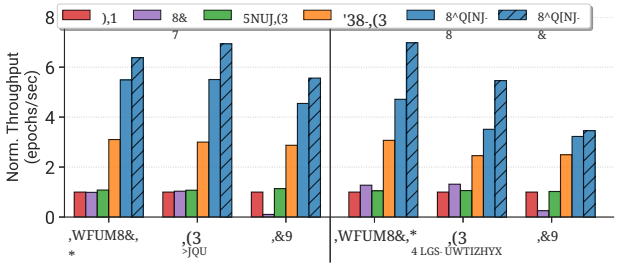


图7。在一台配备 8 个图形处理器的机器上训练时的部分吞吐量结果。Sylvie 仍然优于其他方法。

表3.在两台服务器上使用四个数据集训练 GraphSAGE 时的每轮通信量及时间分解。Sylvie 的通信量减少了近 32×。

	方法	通信量（MB）		每轮时间（秒）	
		主数据	误差补偿	总计	通信
Reddit	DGL	2791.7	0	7.28	6.62
	Sylvie-S	87.3	10.7	0.37	0.29
Yelp	DGL	2348.1	0	4.98	4.70
	Sylvie-S	73.4	4.5	0.29	0.21
Ogbn-products	DGL	3420.6	0	6.03	5.87
	Sylvie-S	106.9	26.6	0.34	0.22
Amazon	DGL	5632.6	0	13.33	11.47
	Sylvie-S	176.1	11	0.69	0.57

Amazon 为 11.0 MB），且仅产生可忽略的开销。

4.2 对收敛的影响

我们在图8中考察了 Sylvie 的收敛曲线。可以看到，在这种情况下，Sylvie-S 的曲线与 DGL 的几乎完全相同。然而，在部分数据集或模型的早期阶段，Sylvie-A 的收敛速度较慢。为减轻陈旧嵌入和特征梯度造成的误差，我们在 Reddit 上使用有界陈旧性适配器训练 Sylvie-A，其中 $\epsilon_s = \{2, 5\}$。我们观察到，通过定期同步最新数据，Sylvie-A 能够显著提升收敛速度；同步越频繁（ϵ_s 越小），其收敛曲线就越接近原生训练。

4.3 量化效果分析

位宽的影响。为探究量化位宽 b 对训练吞吐量提升的贡献，我们在图9中展示了 Sylvie-S 上 GraphSAGE 在不同量化位宽下的训练轮次时间及其分解情况。随着位宽降低，通信开销近乎线性减少，从而缩短轮次时间。原始方法使用 FP32 数据类型进行通信，通信几乎占据了整个轮次时间（1.21 秒中有 0.99 秒）。切换到 FP16 数据类型后，通信成本几乎减半。与原始方法相比，使用 1 位量化可将通信开销减少近 89.8%，每轮训练时间减少 84.2%。

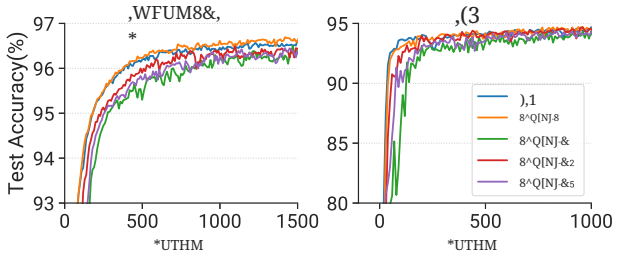


图 8。使用 DGL、Sylvie-S、Sylvie-A 以及配备有界陈旧性适配器（$\epsilon_s = \{2, 5\}$）的 Sylvie-A 在 Reddit 上训练期间的测试准确率收敛情况。

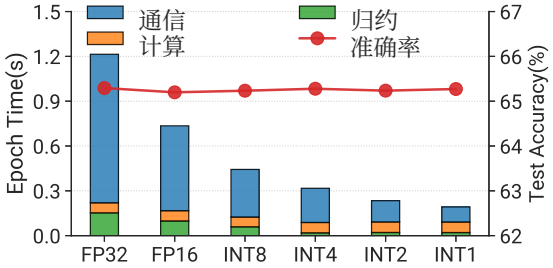


图 9。在单服务器上使用不同量化位宽训练 Yelp 数据集上的 GraphSAGE 时，每个训练轮次的训练时间、时间分解及相应的测试准确率（N=8）。

量化比例的影响。为进一步证明我们的方法对模型性能的有效性，我们通过将**所有**嵌入和特征梯度量化为 1 位来进行训练。表4 展示了 Sylvie-S 与对所有数据进行量化后的测试准确率对比。在所有数据集和模型上，对所有数据进行量化都会导致严重的准确率损失，例如，Sylvie-S 的准确率为 97.2%，而对所有数据进行量化时仅为 70.6%。准确率下降是由于绝对量化对数据造成了过度失真，损害了训练中的前向传播和反向传播。

4.4 开销分析

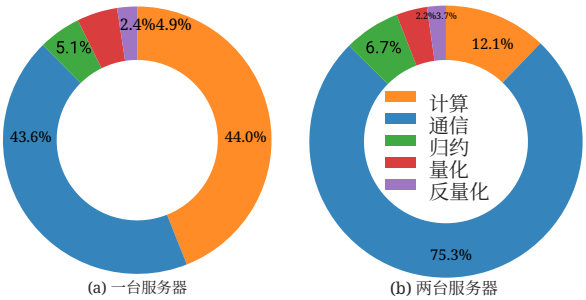
量化开销。为探究低比特模块所占用的开销以及寻找潜在的优化机会，我们在图10中记录了使用 Sylvie-S 在单服务器和两台服务器上训练时，一个训练轮次中各部分所耗费的时间。两种情况下，该模块消耗的时间占比均为最小，表明我们的方法带来的开销可以忽略不计。具体而言，在双服务器设置下，它们的总占比（5.9%）甚至小于全归约的占比（6.7%）。此外，在多服务器设置下，通信开销所占比重更大。

5 未来工作

误差补偿。不可避免地，量化和反量化过程会在训练过程中引入误差。这些嵌入和嵌入梯度的误差甚至会在后续层中累积，严重影

表 *4*。使用 Sylvie 训练并量化所有嵌入和特征梯度时的测试准确率对比。

	Reddit		Yelp		Ogbn-products	
	Sylvie-S	量化全部	Sylvie-S	量化全部	Sylvie-S	量化全部
GraphSAGE	97.15	70.63	65.07	57.14	78.86	58.25
GCN	95.49	92.60	48.77	42.24	74.14	61.25
GAT	94.55	91.72	44.44	25.76	78.00	69.95



(a) 一台服务器　(b) 两台服务器

图*10*。在单服务器和两台服务器上使用 Sylvie-S 对 Reddit 进行 GraphSAGE 训练时，训练轮次时间中各组件所占的比例。低比特模块的开销可以忽略不计。

响模型质量，尤其是在 GNN 模型变深时。在未来的工作中，我们可以利用以下两点观察来调整准确率与训练吞吐量之间的权衡。第一点观察是，训练初期的训练轮次可以采用较低的位宽量化，而后期轮次则采用较高位宽，且不会影响收敛。这样，我们便可以根据训练过程的状态自动调整位宽。另一点观察来自节点的分布特性。在 GNN 层中，聚合阶段是产生大量误差的来源，尤其是在度较高的节点处。随着节点度的增加，聚合值的方差也会增加。因此，未来可以考虑根据节点的重要性，对不同节点采用不同的位宽量化。

内存占用。除了显著提升训练吞吐量外，探索通过量化减少内存占用的可能性也很有价值。使用低精度值有望减少分配的最大内存。例如，在 Ogbn-products 数据集上训练 GraphSAGE 时，采用全精度 FP32 通信的最大内存占用约为 6882 MB，而采用半精度 FP16 通信时约为 6642 MB。然而，由于我们使用图形处理器进行 1位量化和反量化，发现训练过程中会产生一些额外的内存开销。Sylvie-S 的最大内存占用达到 6918 MB，略高于原生训练。尽管如此，利用量化减少内存占用仍是一个有前景的方向，我们将其留作未来可能的研究工作。

6 结论

本文提出了 Sylvie，这是一种高效的分布式图神经网络训练框架，它通过将通信数据量化为低位宽值，在尽可能保持模型质量的同时，大幅降低通信成本。我们还将 Sylvie 与异步流水线技术和有界陈旧性适配器相结合，进一步提升训练性能。大量实验表明，Sylvie 可将训练吞吐量提升至多 28.1×%。

参考文献

利普希茨连续性。 https://en.wikipedia.org/wiki/Lipschitz_连续性，2022 年。

英伟达 CUDA 稀疏矩阵库。 https://docs.nvidia.com/cuda/cusparse/index.html#abstract，2022 年。

Abadal, S.、Jain, A.、Guirado, R.、L´opez-Alonso, J. 和 Alarc´on, E. 图神经网络计算：从算法到加速器的综述。*CoRR*，abs/2010.00130，2021 年。

Alistarh, D.、Grubic, D.、Li, J.、Tomioka, R. 和 Vojnovic, M. QSGD：通过梯度量化和编码实现高效通信的随机梯度下降。载于神经信息处理系统进展，NeurIPS '17，2017 年。

Bottou, L.、Curtis, F. E. 和 Nocedal, J. 大规模机器学习的优化方法。*CoRR*，abs/1606.04838，2018 年。

Chen, J., Ma, T., and Xiao, C. Fastgcn：通过重要性采样实现图卷积网络的快速学习。载于国际学习表征会议，ICLR '18，2018。

Chen, J., Zheng, L., Yao, Z., Wang, D., Stoica, I., Mahoney, M., and Gonzalez, J. Actnn：通过2位激活压缩训练减少训练内存占用。载于第*38*届国际机器学习会议论文集，ICML '21，2021。

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-gcn：一种用于训练深层大型图卷积网络的高效算法。载于第*25*届*ACM SIGKDD*国际知识发现与数据挖掘会议，KDD '19，2019。

Courbariaux, M., Bengio, Y., 和 David, J.-P. Binaryconnect：在传播过程中使用二值权重训练深度神经网络。载于神经信息处理系统进展，NeurIPS '15，2015。

Dong, Y., Ni, R., Li, J., Chen, Y., Zhu, J., 和 Su, H. 利用随机量化训练精确的低比特深度神经网络。*CoRR*, abs/1708.01001，2017。

Feng, B., Wang, Y., Li, X., Yang, S., Peng, X., and Ding, Y. Sgquant：通过专用量化挤出图神经网络的最后一比特。载于*2020 IEEE* 第*32*届人工智能工具国际会议 *(ICTAI)*，2020。

Fey, M. and Lenssen, J. E. 使用 PyTorch Geometric 进行快速图表示学习。*CoRR*，abs/1903.02428，2019。

Hamilton, W., Ying, Z., and Leskovec, J. 大型图上的归纳式表示学习。载于神经信息处理系统进展，NeurIPS ’17，2017。

He, R. and McAuley, J. 起起落落：通过单类协同过滤对时尚趋势的视觉演变进行建模。载于第*25*届万维网国际会议论文集，WWW ’16，2016。Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. 开放图基准：图上的机器学习数据集。载于神经信息处理系统进展，NeurIPS ’20，2020。Huang, W., Zhang, T., Rong, Y., and Huang, J. 面向快速图表示学习的自适应采样。载于第*32*届神经信息处理系统国际会议论文集，NIPS’18，2018。Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. 通过量化与训练实现神经网络的高效纯整数推理。载于*IEEE*计算机视觉与模式识别会议论文集，CVPR ’18，2018。Jia, Z., Lin, S., Gao, M., Zaharia, M., and Aiken, A. 通过 ROC 提升图神经网络的准确率、可扩展性和性能。载于机器学习与系统会议论文集，MLSys ’20，2020。Kipf, T. N. and Welling, M. 基于图卷积网络的半监督分类。载于国际学习表征会议，ICLR ’16，2016。Krishnamoorthi, R. 面向高效推理的深度卷积网络量化：白皮书。*CoRR*，abs/1806.08342，2018。Liu, Z., Zhou, K., Yang, F., Li, L., Chen, R., and Hu, X. Ex-act：通过极限激活压缩实现可扩展的图神经网络训练。载于国际学习表征会议，ICLR ’21，2021。

Ma, L., Yang, Z., Miao, Y., Xue, J., Wu, M., Zhou, L., and Dai, Y. NeuGraph：在大型图上并行执行深度神经网络计算。发表于*2019*年*USENIX*年度技术会议，USENIX ATC ’19，2019。Mostafa, H. 顺序聚合与重物化：在大型图上分布式全批次训练图神经网络。发表于机器学习与系统会议论文集，MLSys ’22，2022。Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., a nd Chintala, S. PyTorch：一种命令式风格的高性能深度学习库。发表于神经信息处理系统进展，NeurIPS ’19，2019。Ramezani, M., Cong, W., Mahdavi, M., Kandemir, M., and Sivasubramaniam, A. 局部学习，全局纠正：一种用于训练图神经网络的分布式算法。发表于国际学习表征会议，ICLR ’22，2022。Stock, P., Fan, A., Graham, B., Grave, E., Gribonval, R., Jegou, H., and Joulin, A. 通过量化噪声训练实现极致模型压缩。发表于国际学习表征会议，ICLR ’20，2020。Tailor, S. A., Fernandez-Marques, J., and Lane, N. D. Degree-quant：图神经网络的量化感知训练。*CoRR*，abs/2008.05000，2021。Thorpe, J., Qiao, Y., Eyolfson, J., Teng, S., Hu, G., Jia, Z., Wei, J., Vora, K., Netravali, R., Kim, M., and Xu, G. H. Dorylus：借助分布式CPU服务器和无服务器线程，实现经济、可扩展且准确的GNN训练。发表于第*15*届*USENIX*操作系统设计与实现研讨会，OSDI ’21，2021。Tripathy, A., Yelick, K., and Buluc, A. 减少图神经网络训练中的通信。发表于*SC20*：高性能计算、网络、存储与分析国际会议，2020。Veli ˇckovi ´c, P., Cucurull, G., Casanova, A., Romero, A., Li `o, P., and Bengio, Y. 图注意力网络。发表于国际学习表征会议，ICLR ’18，2018。Wan, C., Li, Y., Li, A., Kim, N. S., and Lin, Y. BNS-GCN：利用分区并行和随机边界节点采样高效进行图卷积网络全图训练。发表于机器学习与系统会议论文集，MLSys ’22，2022a。

Wan, C., Li, Y., Wolfe, C. R., Kyrillidis, A., Kim, N. S., and Lin, Y. PipeGCN：通过流水线式特征通信高效进行图卷积网络全图训练。发表于国际学习表征会议，ICLR ’22，2022b。

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library：一种以图为中心、高性能的图神经网络软件包。*CoRR*, abs/1909.01315, 2020. Wang, Y., Feng, B., and Ding, Y. Qgtc：通过 GPU 张量核心加速量化图神经网络。载于第 *27* 届 *ACM SIGPLAN* 并行编程原理与实践研讨会论文集，PPoPP ’22, 2022. Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad：通过三值梯度减少分布式深度学习中的通信。载于神经信息处理系统进展，NeurIPS ’17, 2017. Xian, L., Li, B., Liu, J., Guo, Z., and Du, D. H. C. H-ps：一种用于分布式神经网络训练的异构感知参数服务器。*IEEEAccess*, 9:44049–44058, 2021. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint：一种基于图采样的归纳式学习方法。载于国际学习表征会议，ICLR ’20, 2020. Zhang, M. and Chen, Y. 基于图神经网络的链接预测。载于神经信息处理系统进展，NeurIPS ’18, 2018. Zhao, Y., Wang, D., Bates, D., Mullins, R., Jamnik, M., and Lio, P. 学习得到的低精度图神经网络。*CoRR*, abs/2009.09232, 2020. Zheng, D., Ma, C., Wang, M., Zhou, J., Su, Q., Song, X., Gan, Q., Zhang, Z., and Karypis, G. Distdgl：面向十亿规模图的分布式图神经网络训练。*CoRR*, abs/2010.05337, 2021. Zhu, R., Zhao, K., Yang, H., Lin, W., Zhou, C., Ai, B., Li, Y., and Zhou, J. Aligraph：一个综合性的图神经网络平台。*VLDB Endowment*, 12:2094–2105, 2019. Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., and Gu, Q. 用于训练深层大型图卷积网络的层依赖重要性采样。载于第 *33* 届神经信息处理系统进展国际会议论文集，NIPS’19, 2019.