分类号 _____ 编号_____

U D C_____ 密级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：　改善滞后性的图神经网络训练框架

姓　　名：　喻子洋

学　　号：　11910419

系　　别：　数学系

专　　业：　数学与应用数学

指导教师：　张振 副教授

2023 年 4 月 15 日

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名:

_____ 年____ 月____ 日

改善滞后性的图神经网络训练框架

喻子洋

（数学系　指导教师：张振）

[摘要]：尽管最近 GNN 取得了成功，但在具有数百万节点和数十亿条边的大型图上训练 GNN 仍然具有挑战性，这在许多现实世界的应用中是普遍存在的。在分布式 GNN 训练中，” 基于分区” 的方法享有较低的通信成本，但由于掉落的边而遭受信息损失，而” 基于传播” 的方法避免了信息损失，但由于邻居爆炸而遭受令人望而却步的通信开销。这两者之间的一个自然的” 中间点” 是通过缓存历史信息（例如，节点嵌入），这可以实现恒定的通信成本，同时保留不同分区之间的相互依赖。然而，这样的好处是以涉及过时的训练信息为代价的，从而导致呆板性、不精确性和收敛问题。为了克服这些挑战，本文提出了 SAT（滞后性缓解训练），这是一个新颖的、可扩展的分布式 GNN 训练框架，可以自适应地减少嵌入的滞后性。具体来说，我们提出将嵌入预测建模为学习由历史嵌入引起的时间图，并建立一个子图不变的弱监督辅助模块来预测未来的嵌入，该模块以数据驱动的方式减少呆滞性，并具有良好的可扩展性。预测的嵌入作为一个桥梁，以在线的方式交替训练分布式 GNN 和辅助模块。最后，我们提供了广泛的理论和经验分析来证明所提出的方法可以在性能和收敛速度上大大改善现有的框架，而只需最小的额外费用。

[关键词]：在线学习；　图神经网络；　分布式训练

[**ABSTRACT**]: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[**Key words**]: LaTeX, Interface

目录

1. 简介

图神经网络在分析非欧几里得图数据方面取得了令人印象深刻的成功，并在各种应用中取得了可喜的成果，包括社交网络、推荐系统和知识图等[1-3]。尽管 GNN 有很大的前景，但在应用于现实世界中常见的大型图时，GNN 遇到了巨大的挑战–大型图的节点数量可以达到数百万甚至数十亿。在大型图上训练 GNN，由于缺乏内在的并行性，共同面临着挑战。在大型图上训练 GNN，由于反向传播优化中缺乏固有的并行性，以及图节点之间严重的相互依赖性，共同构成了挑战。为了应对这种独特的挑战，分布式 GNN 训练是一个很有前景的开放领域，近年来吸引了越来越多的关注[4-7]，并已成为在大型图上快速准确训练的标准。

现有的分布式 GNN 训练方法可以分为两类，即" *基于分割*" 和" *基于传播*"，根据它们如何解决*计算/通信成本*和*信息损失*之间的权衡。" 基于分割" 方法[5, 8, 9] 通过丢弃跨子图的边，将图分割成不同的子图，因此大图上的 GNN 训练被分解成多个较小的训练任务，每个训练任务在一个孤立的子图中并行，减少子图间的通信。然而，这将导致严重的信息损失，因为对子图之间节点的依赖关系一无所知，并导致性能下降。为了减轻信息损失，" 基于传播" 的方法[10-13] 不忽略不同子图之间的边，子图之间的邻居通信，以满足 GNN 的邻居聚合。然而，随着 GNN 的深入，参与邻居聚合的邻居数量呈指数级增长（即，*邻居爆炸*[14]），因此不可避免地遭受巨大的通信开销和困扰的效率。打破基于分区和传播的方法的悖论的一个自然方法是通过增加第三个维度，形成一个 *抵消三角*。具体来说，通过利用离线存储器来存储子图外邻居的*历史*嵌入，并在下一个阶段需要时将其拉到 GPU 上，可以实现相对于节点数量的恒定通信成本，同时保留子图之间的相互依赖关系。这种技术[6, 7, 15, 16] 已被广泛用于分布式 GNN 训练，并取得了最先进的性能。

2. 背景

图形神经网络。GNN 的目的是在一个具有节点表征 $X \in R^{|V| \times d}$ 的图上学习信号/特征的函数，其中 d 表示节点特征维度。对于典型的半监督节点分类任务[17]，其中每个节点都与一个标签相关、一个层的 GNN 参数化为被训练来学习节点表征，这样可以被准确预测。GNN 的训练过程实际上可以描述为基于*消息传递机制*[18] 的节点表示学习。从分析上看，给定一个图和一个节点，GNN 的第三层被定义为

$$\begin{aligned} \mathbf{h}_v^{(\ell+1)} &= \boldsymbol{f}_{\boldsymbol{\theta}}^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \{\!\{\mathbf{h}_u^{(\ell)}\}\!\}_{u \in \mathcal{N}(v)}\right) \\ &= \Psi_{\boldsymbol{\theta}}^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \Phi_{\boldsymbol{\theta}}^{(\ell+1)}\left(\{\!\{\mathbf{h}_u^{(\ell)}\}\!\}_{u \in \mathcal{N}(v)}\right)\right), \end{aligned} \tag{1}$$

其中，$h_v^{(l)}$ 表示节点 v 在 ℓ 第三层的表示，$\mathbf{h}_v^{(0)}$ 被初始化为 $\mathbf{x}_v$($\mathbf{X}$ 中第 v 行)，$mathcal N(v)$ 表示节点 v 的邻居集合。GNN 的每一层，即 $\boldsymbol{f}_{\boldsymbol{\theta}}^{(\ell)}$，可以进一步分解为两个部分：1) 聚合函数 $\Phi_{\boldsymbol{\theta}}^{(\ell)}$，它将节点 v 的邻居的节点表示作为输入，输出聚合的邻居表示。2) 更新函数 $\Psi_{\boldsymbol{\theta}}^{(\ell)}$，它结合 v 的表示和聚集的邻域表示，为下一层更新节点 v 的表示。$\Phi_{\boldsymbol{\theta}}^{(\ell)}$ 和 $\Psi_{\boldsymbol{\theta}}^{(\ell)}$ 都可

以选择在不同类型的 GNN 中使用各种函数。为了在一台机器上训练 GNN，我们可以在训练数据的整个图上使经验损失 $\mathcal{L}(\boldsymbol{\theta})$ 最小化，即 $\mathcal{L}(\boldsymbol{\theta}) = |\mathcal{V}|^{-1}\sum_{v\in\mathcal{V}} Loss(\mathbf{h}_v^{(L)}, \mathbf{y}_v)$，其中 $Loss(\cdot,\cdot)$ 表示损失函数（如交叉熵损失），$\mathbf{h}_v^{(L)}$ 表示来自 GNN 最后一层的节点 v 的表示。

GNN 的分布式训练。分布式 GNN 训练首先将原始图划分为多个没有重叠的子图，这也可以被认为是小批。然后在不同的设备上并行地训练不同的小批。这里，Eq. 1可以进一步重新表述为

$$\mathbf{h}_v^{(\ell+1)} = \boldsymbol{f}_{\boldsymbol{\theta}}^{(\ell+1)}\Big(\mathbf{h}_v^{(\ell)}, \underbrace{\{\!\{\mathbf{h}_u^{(\ell)}\}\!\}_{u\in\mathcal{N}(v)\cap\mathcal{S}(v)}}_{\text{In-subgraph nodes}} \cup \underbrace{\{\!\{\mathbf{h}_u^{(\ell)}\}\!\}_{u\in\mathcal{N}(v)\setminus\mathcal{S}(v)}}_{\text{Out-of-subgraph nodes}}\Big), \tag{2}$$

其中 $\mathcal{S}(v)$ 表示节点 v 所属的子图。在本文中，我们考虑用多台本地机器和一台全球服务器对 GNN 进行分布式训练。原始输入图 $\mathcal{G}$ 首先被划分为 M 子图，其中每个 $\mathcal{G}_m(\mathcal{V}_m, \mathcal{E}_m)$ 代表第 m 子图。我们的目标是通过最小化每个局部损失，以分布式方式找到最佳参数集 $\boldsymbol{\theta}$，即

$$\begin{aligned} &\min_{\boldsymbol{\theta}_m} \mathcal{L}_m^{\text{Local}}(\boldsymbol{\theta}_m) = \frac{1}{|\mathcal{V}_m|}\sum_{v\in\mathcal{V}_m} Loss(\mathbf{h}_v^{(L)}, \mathbf{y}_v), \\ &\text{s.t.} \quad \forall\ m = 1, 2, \cdots, M \ \ \textit{in parallel}, \end{aligned} \tag{3}$$

其中 $\{\boldsymbol{\theta}_m\}_{m=1}^M$ 是每个本地 GNN 模型的参数，$\mathbf{h}_v^{(L)}$ 遵循 Eq. 2递归。在每一轮通信中，每个局部参数都被汇总以更新全局参数，即 $\boldsymbol{\theta} = AGG(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_M)$。

现有的分布式 GNN 框架一般都试图在通信成本和信息损失之间找到一个最佳的平衡点。" 基于分区" 的方法将经典分布式训练中现有的数据并行技术在 *i.i.d* 数据上推广到图数据上，并享有最小的通信成本。具体来说，当前子图之外的邻居的嵌入（如 Eq. 2）被放弃。" 基于传播" 的方法考虑使用每个子图的邻居节点的通信来满足 GNN 的邻居聚合。如公式 2所示，当前子图之外的邻居节点的表示在不同子图之间交换。然而，随着 GNN 模型的深入，参与邻居聚合过程的邻居数量呈指数级增长，这就是所谓的邻居爆炸问题 *neighborhood explosion*。

然而，上述想法的一个潜在问题是嵌入的呆滞性，这意味着在前向传递过程中节点嵌入的延迟，反过来导致后向传递过程中梯度的延迟。这两种延迟都会损害模型的性能和收敛性。

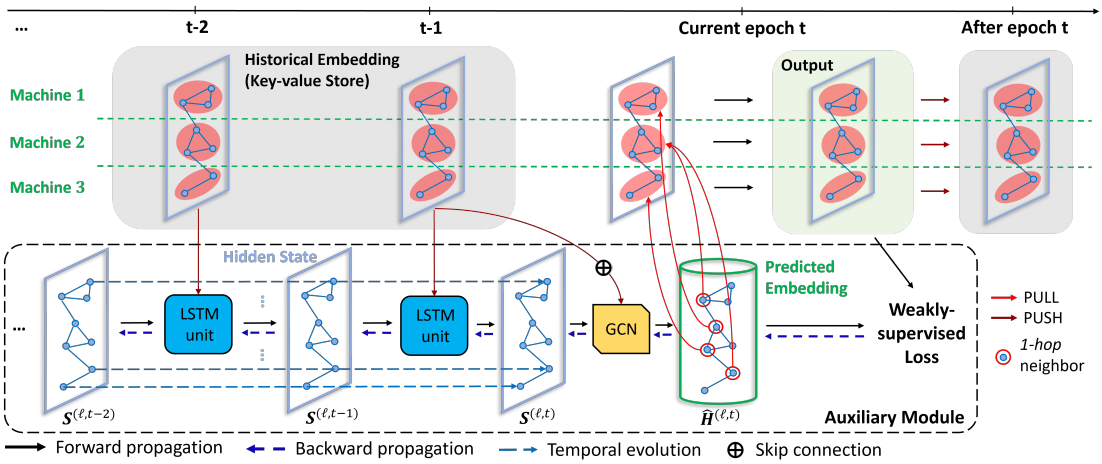

图 1 A high level overview of our framework (per-layer view). Best viewed in color.

3. 问题描述

分布式 GNN 训练下的时态图预测. 我们遵循 Eq. 2和 Eq. 3中定义的 GNN 的分区并行分布式训练。考虑到任何分区 $\mathcal{G}_m$，我们将 Eq. 2以矩阵形式重新表述为

$$\mathbf{H}_{in}^{(\ell+1,m)} = \boldsymbol{f}_{\boldsymbol{\theta}}^{(\ell+1)}\left(\mathbf{H}_{in}^{(\ell,m)}, \mathbf{H}_{out}^{(\ell,m)}\right), \tag{4}$$

其中 $\mathbf{H}_{in}^{(\ell,m)}$ 和 $\mathbf{H}_{out}^{(\ell,m)}$ 分别表示分区 $\mathcal{G}_m$ 上第 ℓ 层的子图内和子图外节点嵌入矩阵。如前所述，直接在每个子图之间交换 $\mathbf{H}_{out}^{(\ell,m)}$ 将导致指数级的通信成本。现有的方法通过存储在离线存储器中的历史值对 $\mathbf{H}_{out}^{(\ell,m)}$ 进行近似，即 $\mathbf{H}_{out}^{(\ell,m)} \approx \tilde{\mathbf{H}}_{out}^{(\ell,m)}$，这导致了 $\delta\mathbf{H}^{(\ell,m)} = \mathbf{H}_{out}^{(\ell,m)} - \tilde{\mathbf{H}}_{out}^{(\ell,m)}$。相反，我们考虑以数据驱动的方式预测 $\mathbf{H}_{out}^{(\ell,m)}$，使预测的嵌入 $\hat{\mathbf{H}}_{out}^{(\ell,m)}$ 具有较小的滞后性。

在这项工作中，我们创新性地将分布式 GNN 训练下的嵌入预测任务表述为对时态图演化模式的建模。

具体来说，考虑 $\bar{\mathcal{G}}_m = (\bar{\mathcal{V}}_m, \bar{\mathcal{E}}_m)$ 的子图，该子图由 $\mathcal{G}_m^{(t)}$ 及其 1-*hop* 邻居诱导。由于 $\mathbf{H}_{in}^{(m)}$ 和 $\mathbf{H}_{out}^{(m)}$ 在训练过程中不断变化（即、在历时中），图的序列 $\bar{\mathcal{G}}_m^{(t)} = (\bar{\mathcal{V}}_m, \bar{\mathcal{E}}_m, \mathbf{H}_{in}^{(t,m)}, \mathbf{H}_{out}^{(t,m)})$，$t = 1, 2, \cdots, T$ 成为一个具有固定拓扑结构的*时空图*[19]，其中 T 表示历时的总数。

我们的目标是建立一个模型，积极主动地捕捉不断变化的模式，并以在线方式预测嵌入。详细来说，给定时间图 $\{\bar{\mathcal{G}}^{(s)} : t - \tau \leq s \leq t\}$ 到历时 t，其中 τ 表示滑动窗口长度，我们学习一个映射函数 M_{ω}，参数为 $\boldsymbol{\omega}$，这样它可以预测下一个历时的子图外嵌入 $\hat{\mathbf{H}}_{out}^{(t+1,m)}$。对 M_{ω} 的训练和推理在每个历时 $t = 1, 2, \cdots, T-1$ 进行一次。

尽管有此必要，但由于现有的几个挑战，如何处理上述问题是一个开放的研究领域：1). 难以设计出一种高效的、可扩展的减少呆滞性的算法。2). 嵌入的未知和复杂演变。3). 由于嵌入的未知动态性，在理论上很难获得对具有预测嵌入的分布式 GNN 框架的性能和收敛性的保证。

训练目标。为了有效而廉价地获得辅助模型的监督，我们提出了一种弱监督的误差回归损失来训练辅助模型，它带来的额外计算成本几乎为零。具体来说，对于任何

子图外的节点 $u \in \mathcal{N}(v) \setminus \mathcal{G}_m$，一定存在另一个子图 $\mathcal{G}_n$，使得 $u \in \mathcal{G}_n$（每个节点必须属于某些图的分区）。在子图 $\mathcal{G}_n$ 上用 u 替换节点 v，在 Eq. 6中可以得到 $\mathbf{h}_u^{(\ell)}$，其中的计算在 GNN 模型的前向传播中是/emphhidden。用 $\mathbf{h}_u^{(\ell)}$ 减去 $\tilde{\mathbf{h}}_u^{(\ell)}$ 的结果是地真相嵌入误差 $\Delta\mathbf{h}_u^{(\ell)} = \mathbf{h}_u^{(\ell)} - \tilde{\mathbf{h}}_u^{(\ell)}$、而辅助模型 $M_\ell(\cdot)$ 的训练损失可以定义为 $M_\ell(\cdot)$,

$$\min \sum\nolimits_{u \in \mathcal{G}} \|\Delta\hat{\mathbf{h}}_u^{(\ell)} - \Delta\mathbf{h}_u^{(\ell)}\|_2^2, \tag{5}$$

其中最小化是在 M_ℓ 的参数上。我们的上述目标函数可以通过从 $\mathcal{G}$ 中取样的小批次轻松地扩展到随机训练，这就允许了*可控*的训练样本量，这取决于辅助模型的效率和性能之间的权衡。

备注 1).。如公式所示 7，辅助模型借用了 skip-connection[20] 的思想，辅助模型的预测可以解释为一个错误*修正*项，它纠正了前一时代嵌入的呆滞性。**2）.**。我们在 Eq. 6中的表述是相当普遍的，而且是与模型无关的。现有的具有历史嵌入的 GNN 训练框架[6, 15, 16] 可以享受我们的预测嵌入来进一步提升他们的性能。

4. 解决方法

在这一节中，我们介绍了我们提出的呆滞性-减弱分布式 GNN 训练框架，以解决上述挑战。对于第一个挑战，我们建立了一个子图不变的弱监督辅助模块，用于未来的嵌入预测，该模块以数据驱动的方式减少呆滞性，并具有良好的可扩展性。对于第二个挑战，我们建议将 GNN 与递归结构结合起来，这样前者可以捕获节点连接中的信息，后者可以捕获时间演变的信息。同时，我们的在线设计允许辅助模块和分布式 GNN 交替训练，并使之相互受益。我们通过引入下采样预训练和自适应微调频率，进一步提高辅助模块的性能和效率。最后，为了应对最后的挑战，我们探讨了所引入的辅助模块在分布式环境下如何影响模型性能和收敛的理论保证。

4.1 框架

在本地机器上训练的 GNN 的每个副本将利用所有可用的图信息，即在前向和后向传播中都放弃了边缘。从分析上看，计算每个本地梯度 $\nabla\mathcal{L}_m^{\text{Local}}$，如公式 3中定义的，将涉及子图外的邻居信息。对于子图外的邻居节点，我们考虑使用由辅助模型 (表示为 $\hat{\mathbf{h}}_v^{(\ell)}$) 预测的 *predicted embedding* 代替真实嵌入。形式上，给定 $v \in \mathcal{G}_m(\mathcal{V}_m, \mathcal{E}_m)$ 的节点来自第 m 个子图，通过修改 Eq. 2实现前向传播。

$$\mathbf{h}_v^{(\ell+1)} = \boldsymbol{f}_{\boldsymbol{\theta}}^{(\ell+1)}\Big(\mathbf{h}_v^{(\ell)}, \{\!\{\mathbf{h}_u^{(\ell)}\}\!\}_{u \in \mathcal{N}(v) \cap \mathcal{G}_m} \cup \underbrace{\{\!\{\hat{\mathbf{h}}_u^{(\ell)}\}\!\}_{u \in \mathcal{N}(v) \setminus \mathcal{G}_m}}_{\text{predicted embedding}}\Big), \tag{6}$$

其中 $\forall\, u \in \mathcal{N}(v) \setminus \mathcal{G}_m$ 满足

$$\hat{\mathbf{h}}_u^{(\ell)} = \tilde{\mathbf{h}}_u^{(\ell)} + \Delta\hat{\mathbf{h}}_u^{(\ell)}, \quad \Delta\hat{\mathbf{h}}_u^{(\ell)} = M_\ell(\tilde{\mathbf{h}}_u^{(\ell)}, \xi). \tag{7}$$

这里，$\tilde{\mathbf{h}}_u^{(\ell)}$ 表示前一个历时的嵌入，$M_\ell(\cdot)$ 表示 ℓ 第三层的辅助模型。ξ 表示任何其他历史信息作为 $M_\ell(\cdot)$ 的额外输入。如公式 7所示，预测的嵌入等同于前一纪元的嵌入加上一个误差项，这个误差项是由辅助模型仅根据历史信息预测的。

请注意，在 Eq 7中，我们的辅助模型是在不同的子图中共享的（$M_\ell(\cdot)$ 不含 m）。原因有 3 个方面：**1).**。从理论上讲，在分割之前，所有的子图都来自同一个全局输入图，因此共享相同的嵌入分布。我们通过在不同的子图/分区中共享我们的辅助模块来编码这种归纳偏见，这使得辅助模块能够捕捉重要的或重复的模式，而不考虑子图的差异。**2).**。实际上，由于过度平滑的问题，GNN 模型的层次相对较浅[21]。另一方面，现代图可以轻易地超过数百万个节点。为了处理这样巨大的图，人们必须将其划分为大量的子图，以便将每个子图装入一个 GPU。因此，在分布式 GNN 问题中，我们通常有 $M \gg L$，所以一个更有效的内存方式是在不同的子图上共享辅助模型。**3).**。更重要的是，这种共享策略还可以增加辅助模型的训练样本量，从而提高辅助模型的*generalizability*。如公式 5所示，M_ℓ 享有整个输入图 $\mathcal{G}$ 的训练样本量（即 $|\mathcal{V}(\mathcal{G})|$），而针对分区的辅助模型，即 $M_{\ell,m}$ 只能有最多 $|\mathcal{V}(\mathcal{N}(\mathcal{G}_m) \setminus \mathcal{G}_m)|$ 样本来学习，这是子图 $\mathcal{G}_m$ 的 1-emphhop 邻居数量。

参考文献

[1] DAI H, DAI B, SONG L. Discriminative embeddings of latent variable models for structured data[C]//International conference on machine learning. [S.l. : s.n.], 2016: 2702-2711.

[2] YING R, HE R, CHEN K, et al. Graph convolutional neural networks for web-scale recommender systems[C]//Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. [S.l. : s.n.], 2018: 974-983.

[3] LEI K, QIN M, BAI B, et al. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks[C]//IEEE INFOCOM 2019-IEEE Conference on Computer Communications. [S.l. : s.n.], 2019: 388-396.

[4] THORPE J, QIAO Y, EYOLFSON J, et al. Dorylus: Affordable, Scalable, and Accurate GNN Training with Distributed CPU Servers and Serverless Threads[C/OL]// 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21). [S.l.]: USENIX Association, 2021: 495-514. https://www.usenix.org/conference/osdi21/presentation/thorpe.

[5] RAMEZANI M, CONG W, MAHDAVI M, et al. Learn Locally, Correct Globally: A Distributed Algorithm for Training Graph Neural Networks[J]. ArXiv preprint arXiv:2111.08202, 2021.

[6] WAN C, LI Y, WOLFE C R, et al. PipeGCN: Efficient full-graph training of graph convolutional networks with pipelined feature communication[J]. ArXiv preprint arXiv:2203.10428, 2022.

[7] CHAI Z, BAI G, ZHAO L, et al. Distributed Graph Neural Network Training with Periodic Historical Embedding Synchronization[J]. ArXiv preprint arXiv:2206.00057, 2022.

[8] ANGERD A, BALASUBRAMANIAN K, ANNAVARAM M. Distributed training of graph convolutional networks using subgraph approximation[J]. ArXiv preprint arXiv:2012.04930, 2020.

[9] JIA Z, LIN S, GAO M, et al. Improving the accuracy, scalability, and performance of graph neural networks with roc[J]. Proceedings of Machine Learning and Systems, 2020, 2: 187-198.

[10] MA L, YANG Z, MIAO Y, et al. {NeuGraph}: Parallel Deep Neural Network Computation on Large Graphs[C]//2019 USENIX Annual Technical Conference (USENIX ATC 19). [S.l. : s.n.], 2019: 443-458.

[11] ZHU R, ZHAO K, YANG H, et al. Aligraph: a comprehensive graph neural network platform[J]. ArXiv preprint arXiv:1902.08730, 2019.

[12] ZHENG D, MA C, WANG M, et al. Distdgl: distributed graph neural network training for billion-scale graphs[C]//2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3). [S.l. : s.n.], 2020: 36-44.

[13] TRIPATHY A, YELICK K, BULUÇ A. Reducing communication in graph neural network training[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. [S.l. : s.n.], 2020: 1-14.

[14] HAMILTON W, YING Z, LESKOVEC J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.

[15] CHEN J, ZHU J, SONG L. Stochastic Training of Graph Convolutional Networks with Variance Reduction[C]//International Conference on Machine Learning. [S.l. : s.n.], 2018: 942-950.

[16] FEY M, LENSSEN J E, WEICHERT F, et al. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings[C]//International Conference on Machine Learning. [S.l. : s.n.], 2021: 3294-3304.

[17] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks[J]. ArXiv preprint arXiv:1609.02907, 2016.

[18] GILMER J, SCHOENHOLZ S S, RILEY P F, et al. Neural message passing for quantum chemistry[C]//International conference on machine learning. [S.l. : s.n.], 2017: 1263-1272.

[19] WU L, CUI P, PEI J, et al. Graph neural networks[G]//Graph Neural Networks: Foundations, Frontiers, and Applications. [S.l.]: Springer, 2022: 27-37.

[20] HE K, ZHANG X, REN S, et al. Identity mappings in deep residual networks[C]// European conference on computer vision. [S.l. : s.n.], 2016: 630-645.

[21] CHEN D, LIN Y, LI W, et al. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view[C]//Proceedings of the AAAI Conference on Artificial Intelligence:vol. 3404. [S.l. : s.n.], 2020: 3438-3445.

附录

在这一节中，我们描述了详细的实验设置、额外的实验结果和完整的证明。我们重新使用了从 GNNAutoscale[16] 中采用的部分代码；本文的代码可在：https://github.com/Ziyang-Yu/SAT。请注意，为了提高可读性，对代码进行了重新组织。

实验设置的细节

所有的实验都是在 AWS 上的 EC2 **g4dn.metal** 虚拟机（VM）实例上进行的，该实例拥有 8 美元的 NVIDIA T4 GPU，96 美元的 vCPUs，384 美元 GB 的主内存。其他重要信息，包括操作系统版本、Linux 内核版本和 CUDA 版本，在表 reftab: 版本中进行了总结。为了公平比较，我们对所有 10 个框架，PipeGCN、PipeGCN$^+$、GNNAutoscale、GNNAutoscale$^+$、DIGEST、DIGEST$^+$、VRGCN、VRGCN$^+$、DIGEST-A 和 DIGEST-A$^+$ 使用相同的优化器（亚当）、学习率和图划分算法。对于 PipeGCN、GNNAutoscale、DIGEST、VRGCN、DIGEST-A 独有的参数，如每个节点从每层采样的邻居数和层数，我们为 PipeGCN$^+$、GNNAutoscale$^+$、DIGEST$^+$、VRGCN$^+$ 和 DIGEST-A$^+$ 选择相同值。

这十个框架中的每一个都有一套专门针对该框架的参数；对于这些专属参数，我们对其进行调整，以达到最佳性能。请参考 **small_benchmark/conf** 下的配置文件，了解所有模型和数据集的详细配置设置。

致谢

SUSTechThesis 目前版本为 1.3.3, LaTeX 毕业论文模板项目从提出到现在已有两年了。感谢为本项目贡献代码的开发人员们：

- 梁钰栋（南方科技大学，本科 17 级）；
- 张志炅（南方科技大学，本科 17 级）。

以及使用本项目，并提出诸多宝贵的修改意见的使用人员们：

- 李未晏（南方科技大学，本科 15 级）；
- 张尔聪（南方科技大学，本科 15 级）。

此外，目前的维护者并非计算机系，可能存在对协议等的错误使用，如果你在本模板中发现任何问题，请在 GitHub 中提出 Issues，同时也非常欢迎对代码的贡献！