arXiv:2302.05007v1 [cs.MA] 10 Feb 2023

多智能体强化学习系统中的可扩展性瓶颈

Kailash Gogineni、Peng Wei、Tian Lan 和 Guru Venkatara mani 乔治·华盛顿大学，美国华盛顿哥伦比亚特区 电子邮件：{ kailashg26、pwei、tlan、guruv}@gwu.edu

摘要—多智能体强化学习（多智能体强化学习）是一个前景广阔的研究领域，可对多个自主决策智能体进行建模和控制。在在线训练期间，多智能体强化学习算法涉及计算密集型操作，例如在多个智能体所拥有的大型观测-动作空间中进行探索和利用。本文旨在分析多类常见多智能体强化学习算法在训练阶段的可扩展性瓶颈。实验结果揭示了多智能体强化学习算法中限制可扩展性的关键模块，并提出了可能有助于解决这些性能问题的策略。

索引词—C.4 系统性能 ＜ C 计算机系统组织；I.2.11.d 多智能体系统 ＜ I.2.11 分布式人工智能 ＜ I.2 人工智能 ＜ I 计算方法

1 引言

强化学习（RL）算法广泛应用于机器人学、航空、自动驾驶、游戏、推荐系统和医疗保健领域。RL 框架通过根据当前观测/状态空间采取动作，使用奖励函数评估状态-动作对的质量，然后转移到新状态 [1]，从而优化人工智能智能体的行为及其与环境的交互。决定动作的函数称为策略。智能体旨在寻找能够最大化总累积（折扣）奖励的最优策略。表示奖励估计值的函数称为价值函数。

多智能体强化学习（MARL [1]）是一个快速发展的研究领域，在训练期间，智能体之间会大量共享观测信息，并且这些智能体的联合动作可能动态地影响环境。训练智能体以实现目标，同时尽量减少彼此之间的干扰（障碍），从而执行竞争性（例如捕食者-猎物）和合作性（例如协作导航）任务 [2]。在合作设定中，所有观测都会被共享，并且训练可以集中进行。一种竞争

竞争性设定区分了智能体群体——也就是说，每个智能体都力求胜过对手。在这种环境中，单个智能体采取的动作会动态影响其他智能体的行为及其奖励。因此，多智能体强化学习训练涉及若干计算密集型任务，这些任务需要应对不断变化的环境。

在本文中，我们旨在理解采用演员-评论家方法实现的、众所周知的无模型多智能体强化学习框架 [2], [3],[4] 中的关键可扩展性瓶颈。这些框架的状态空间通常非常庞大。我们分析多智能体强化学习训练的不同阶段，其中演员网络和评论家网络分别负责策略和价值函数。如图 1 所示，当一组智能体与环境交互时，演员网络输出动作，并且每个智能体都会学习一个独立策略，将其观测映射为最优动作（动作选择），以最大化预期回报。在小批量采样阶段，每个智能体 i 会收集存储在所有其他智能体的

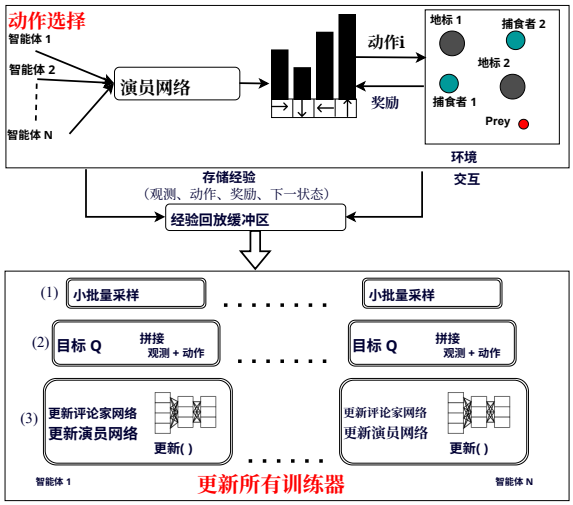
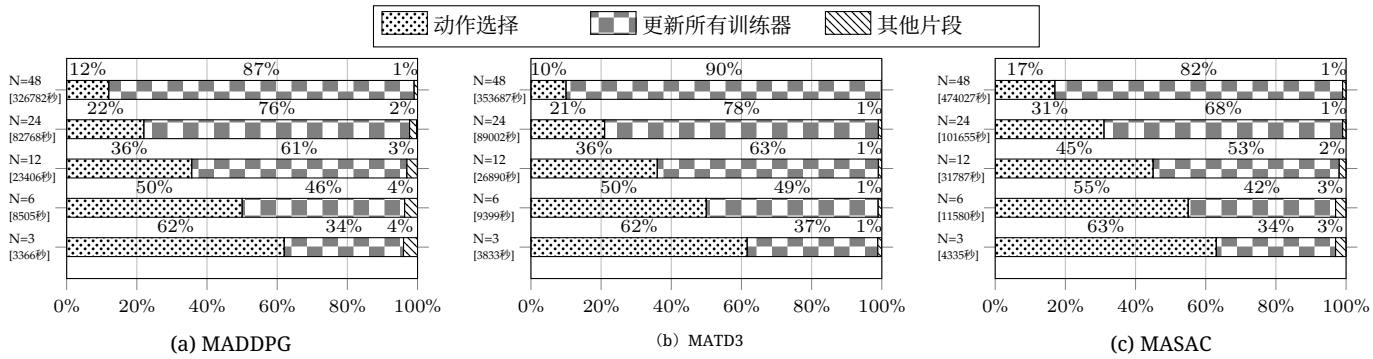


图 1：我们采用的多智能体分散式演员、集中式评论家方法概览（竞争性环境）。

经验回放缓冲区中的观测、动作和新观测。评论家使用从回放缓冲区中存储的过往经验里获取的所有智能体的随机小批量样本来更新价值函数。在更新所有训练器阶段，演员网络使用评论家计算出的 Q 值进行更新 [2]。

2 动机

由于智能体的策略不断演变，且回放缓冲区中的样本会不断刷新，以寻找用于推理的最优策略，因此多智能体强化学习训练对性能要求很高 [2]。图 2 显示，更新所有训练器占 ≈35% 至 ≈90% 的训练时间，且随着多智能体强化学习智能体的数量从 3 个增加到 48 个，这一比例也随之变化。这主要有两个原因：1 在多智能体强化学习中，每个智能体都有自己的演员网络和评论家网络，因为它们可能获得不同的奖励。每个智能体都必须从回放缓冲区中随机抽取一个状态转移小批次，以更新评论家网络和演员网络。这要求每个智能体



(a) MADDPG (b) MATD3 (c) MASAC

图 2：三种多智能体强化学习工作负载的训练时间分解 ，智能体数量为 3 至 48 个。环境为竞争性任务（捕食者 -猎物）。MA 的总训练时间 强化学习算法（单位：秒）显示在纵轴上，单位标注于方括号内。

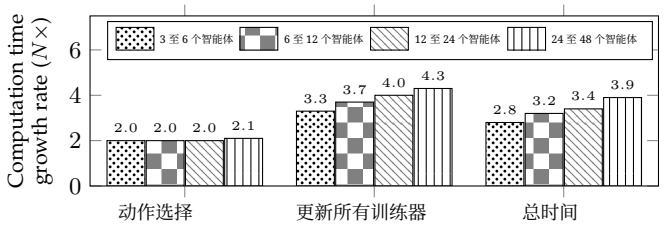


图 3：三个多智能体强化学习框架中，多智能体强化学习模块计算时间的平均增长情况。

从其他每个智能体采样经验。2 由于每个智能体都必须与其他智能体协调，以共享各自的观测和动作，观测空间和动作空间的动态内存需求也会呈二次增长。我们发现，其他多智能体强化学习阶段（例如动作选择）占用的时间较少，且随智能体数量线性增长（图 3）。这是因为，动作选择是基于各个智能体的策略，利用局部观测并与环境交互来执行的。其他环节由经验收集、奖励收集和策略初始化组成，产生的开销可以忽略不计。请注意，智能体在共享环境中交互，而训练时间是所有智能体训练时间的总和。

以 FA3C [5]，为代表的先前研究，重点关注加速多个并行工作线程场景；在这些场景中，每个智能体都在各自的环境中，使用单智能体强化学习算法独立运行。相比之下，我们旨在了解多智能体学习框架，在这类框架中，智能体在同一个共享环境中运行。在此类多智能体强化学习设置中，智能体通常能够高度感知彼此（从而导致空间和动作空间规模庞大）。据我们所知，这是首次对多智能体强化学习的可扩展性瓶颈进行特征研究。

3 背景

通常，包含 N 个智能体的多智能体强化学习设定由一组状态 $S = S_1 \times ... \times S_N$ 和一组动作 $A = A_1 \times ... \times A_N$ 定义。每个智能体使用策略 $\pi_{\theta_i} \quad O_i \times A_i \to :$ $[0, 1]$ $T \quad S \times A_1 \times A_2 \times ... \times A_N$ 选择自己的动作。状态转移（：）函数根据当前状态以及每个智能体的动作，生成下一个状态 S'。每个智能体的奖励 R_i： $S \times A_i \to \mathbb{R}$ 是全局状态和所有其他智能体的动作的函数，目标是最大化自身的期望回报 $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$，其中 γ 表示折扣因子，T 表示时间范围。

为此，我们采用演员-评论家方法，例如 MADDPG [2],MATD3 [3], MASAC [4]。

在 MADDPG [2]，中，每个智能体都会学习一个独立策略，将观测映射为其动作，以最大化由评论家近似计算的期望回报。MADDPG 通过使用 $\mathcal{L}(\theta_i) = \mathrm{IE}_D[(Q_i(S, A_1, ...A_n) - y_i^2]$ 和 $y_i = r_i + \gamma Q_i(S', A_1', ...A_n')_{a_j'=\pi(o_j')}$，以最小化损失和目标 Q 值的方式训练智能体 i 的评论家，其中 S 和 $A_1, ...A_n$ 分别表示联合观测和动作。 D 是经验回放缓冲区，用于存储所有智能体在训练回合结束后获得的观测、动作、奖励和新观测样本。多智能体强化学习框架包含四个网络：演员、评论家、目标演员和目标评论家。 Q_i 和 $\pi(o_j')$ 是目标网络，分别用于实现评论家（Q_i）和演员网络的稳定学习。目标演员根据演员网络输出的状态，利用策略估算下一个动作。目标评论家汇总目标演员的输出，以计算目标 Q 值，进而更新评论家网络，并评估智能体所采取动作的质量。创建目标网络是为了确保训练稳定性。请注意，反向传播阶段的网络更新顺序为：评论家、演员，然后是目标网络。

MATD3 [3] 采用双延迟评论家来解决过估计偏差问题 [3] ，并在从缓冲区采样的动作中加入少量噪声。由于评论家值的变化需要反映到其他智能体的策略中，MATD3会延迟更新目标网络和策略，以便在使用评论家更新演员之前获得准确的评论家。在需要学习制胜策略的领域（例如，捕食者-猎物、协作导航）中，MATD3的表现优于MADDPG [3]。

MASAC [4] 改善了相较于MADDPG和MATD3的收敛特性，并在竞争性和合作环境中获得更高的回报。MASAC采用最大熵强化学习，鼓励智能体在策略中最大限度地进行探索。对于状态-动作值相近的近似最优动作，MASAC赋予它们相同的概率，从而避免反复选择同一个动作。这种学习技巧可以提高稳定性、策略探索能力和样本效率 [4], [6]。

4 评估设置

我们使用多智能体粒子环境（MPE [2]）评估了三种最先进的多智能体强化学习算法：MAD-DPG、MATD3 和 MASAC。我们在表 1 中概述了所选环境的行为。演员和评论家网络均由一个两层 ReLU 多层感知机参数化，每层有 64 个单元

表 1：多智能体粒子环境。

环境	详情
协作 导航	N 个智能体协同移动以到达L 地标和奖励会鼓励智能体 靠近地标。
捕食者- Prey	N 个捕食者协同合作，阻挡 M 个行动迅速的猎物智能体的去路。猎物智能体受到 环境控制，并试图躲避 与捕食者发生碰撞。

采样状态转移时，小批次大小为 1024。在所有实验中，我们使用 Adam 优化器 [7]，学习率为 0.01，最大回合长度为 25（达到终止状态所需的最大步数），并使用 τ = 0.01 更新目标网络。γ是折扣因子，设为 0.95。回放缓冲区的大小为 10^6，MASAC 的熵系数为 0.05。每向回放缓冲区添加 100 个样本后，便更新一次网络参数。

所有工作负载均在 Nvidia GeForce RTX 3090 Ampere 架构上进行训练和性能剖析，该显卡连接至 AMD Ryzen Threadripper PRO 3975WX CPU。该 CPU 拥有 32 个核心、128 MiB 末级缓存、512 GB 主存，时钟频率为 3.5GHz。服务器运行 Ubuntu Linux 20.04.5 LTS 操作系统，配备 CUDA 9.0、cuDNN 7.6.5、PCIe Express® v4.0 和 NCCL v2.8.4 通信库。该机器支持 Python 3.7.15、TensorFlow (v2.11.0)、TensorFlow-GPU (v2.1.0) 和 OpenAI GYM (v0.10.5)。我们使用 Perf [8] 工具和硬件性能计数器进行性能分析。所有工作负载均使用算法推荐的默认超参数训练 60K 个回合。

5 实验评估

为了进行更深入的分析，我们将更新所有训练器划分为多个模块：小批量采样、目标 Q 计算和Q 损失与 P 损失，并在以竞争为主的设定（捕食者-猎物）中展示结果，以了解限制多智能体强化学习可扩展性的关键因素。我们注意到，捕食者智能体以合作设定运作，以最大化其共享回报。因此，我们的测试平台能够评估竞争与合作两种智能体场景。

性能剖析概览。图 4 展示了各模块的耗时占比：小批量采样、目标 Q 计算、Q 损失和 P 损失在不同工作负载下的平均计算时间占比分别为 61%、21%、10% 和 8%。随着智能体数量从 3 个增加到 48 个，我们观察到每周期指令数 (IPC) 稳步下降了 21% 以上（从 1.37 降至 1.08），而全局缓存未命中增加了 54%，这表明随着更多智能体参与，指令吞吐量和缓存性能都会下降。某些关键架构特征（如分支未命中、TLB 加载未命中和全局缓存未命中）的增长率呈超线性，这进一步证实了多智能体强化学习工作负载中存在计算瓶颈（图 5）。

5.1 小批量采样

图 6 中的实验结果显示，在 更新所有训练器 模块中耗时最长的阶段——小批量采样期间，计算时间会随着智能体数量的增加而超线性增长。其他相关性能指标也始终呈现出相同趋势：总指令数-4× 和 末级缓存（*LLC*）加载未命中-3×。猎物智能体与捕食者智能体之间的竞争行为包括：每个捕食者智能体接收相对于所有其他捕食者智能体的速度、位置以及地标作为观测，以捕猎猎物。捕食者智能体之间也存在合作行为，以最大化其共享回报。请注意，各智能体的回放缓冲区彼此独立，以存储各自过去的状态转移。

在此模块中，每个智能体都必须从其他智能体的回放缓冲区中均匀随机地采样一组小批量样本，并更新其评论家网络的参数。每个智能体 i执行 查找-读取-写入操作，其数量随智能体数量增加而增长，N并且这一过程会在全部 N 个智能体上重复。收集转换集的时间复杂度为 $O(N^2K)$，K 表示批次大小。在协作导航（简单扩散 [2]）中，我们观察到类似的可扩展性瓶颈，因为所有智能体都需要共同训练，在彼此避开碰撞的同时抵达地标。

5.2 目标 Q 计算

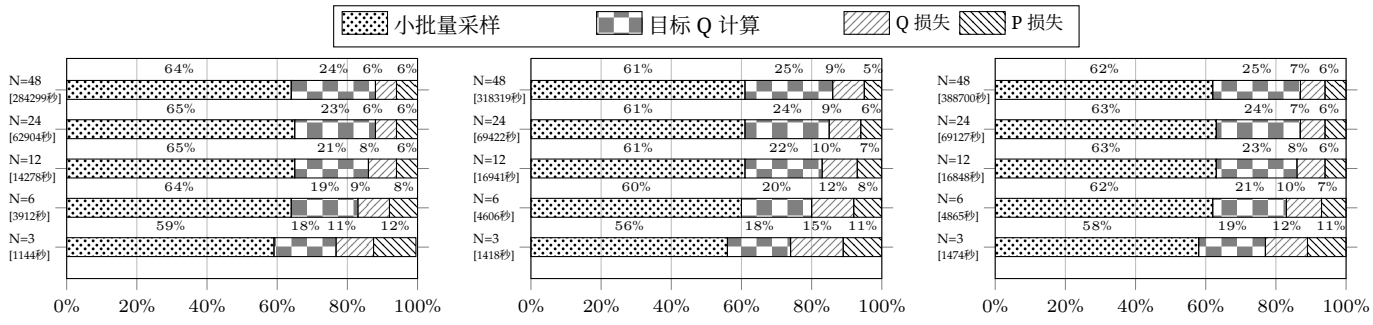
目标 Q 计算阶段是更新所有训练器中耗时第二长的阶段。图 6 显示，该阶段会随智能体数量增加而增长 $4\times$。请注意，在图 4 中，对于目标 Q，其计算时间占更新所有训练器的比例会随智能体数量增加而上升；而Q 损失和 P 损失的执行时间占比则略有下降。这是因为目标 Q的增长速度高于Q 损失（图 6）。每个智能体执行下一动作计算、下一步目标 Q 以及目标 Q 值，其计算量取决于其他所有智能体的联合观测-动作空间。为了计算下一动作，智能体 i 使用其策略网络根据下一状态-*S'*确定下一动作-*a'*。在此阶段，每个智能体的策略网络都涉及与输入权重矩阵的乘法和加法运算，从而影响性能。随后，将得到的 a' 和 S' 数据聚合并拼接为一个向量，以计算协作智能体的下一步目标 Q。Q 函数的输入空间（维度）会随智能体数量呈二次方增长 [9]。每个智能体 i 的目标评论家值都使用目标演员网络的下一步目标 Q值计算得出。值得注意的是，每个智能体都必须读取其他智能体的策略值；因此，对于 N 个智能体，与下一动作-*a'*对应的内存查找操作数量为 $N \times (N-1)$。

5.3 反向传播——Q 损失与 P 损失

反向传播是更新所有训练器中的第三大阶段。该阶段主要是评论家网络的反向传播，用于计算目标评论家网络与评论家网络之间的均方误差损失；同时，演员网络通过最小化 Q 值（评论家网络）来更新。随着智能体数量增加，可训练参数也随之增加，并且会为所有 N 个智能体构建N 个策略网络和N 个评论家网络，因此需要额外花费时间更新每个智能体的权重。每次迭代中，评论家网络的平均计算开销会随着智能体数量每增加 $2\times$ 而最多增长 27%。对于演员网络，平均计算开销会增长 36%，这有助于解释更新各个智能体网络权重时遇到的性能瓶颈。

6 架构指南

实现选择性注意力的架构原语 [6]或许有助于缓解多智能体强化学习中目标 Q 计算和评论家网络的一些可扩展性瓶颈，从而减少网络的输入空间。



(a) MADDPG (b) MATD3 (c) MASAC

图 4：在 Predator-Prey 环境中，针对包含 3 至 48 个智能体的三种不同多智能体强化学习工作负载，更新所有训练器 的训练时间细分。纵轴方括号中显示了更新所有训练器 的总训练时间（秒）。

为了解决与数据移动相关的性能问题，我们注意到，采用存内或近存计算加速器进行处理或许能显著改善相关的架构瓶颈。例如，近期基于 GDDR6 的存内处理（PIM） [10] 可用于加速当前大多数多智能体强化学习框架中四个神经网络的矩阵乘累加和激活操作。未来的 PIM 设计还可加入多智能体强化学习专用的硬件原语（例如高效的小批量采样），以提升硬件效率。多核存内处理神经网络加速器可用于充分利用多智能体强化学习算法中多个神经网络所提供的更高粗粒度硬件并行性。

在小批量采样期间，也可以利用代码优化和数据并行。可使用共享内存多线程来加快采样（例如通过 OpenMP）。这有助于缓解计算瓶颈：每个智能体通过并行线程独立采样自己的状态转移 [11]。我们的硬件性能分析还表明，分支和 TLB行为有改进空间，并且需要通过硬件和软件优化来缓解这些问题。

对于评论家网络的输入，可根据智能体在环境中的重要性，对选定的智能体组采用多级压缩 [12] 技术。此外，小批量采样期间出现的 LLC 缺失表明存在末级缓存竞争，可通过智能的缓存分配策略加以解决。其他模块，例如下一动作计算、环境交互和动作选择阶段，也可能受益于并行化以及对关键模块进行定制加速。

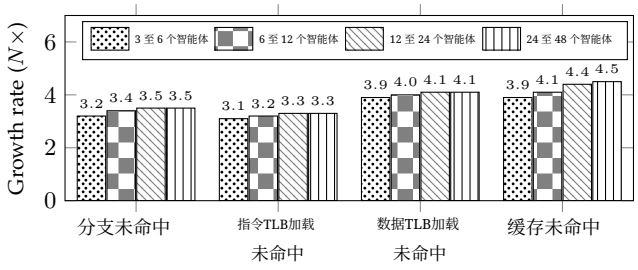


图 5：当智能体数量按 2×扩展时，更新所有训练器在三个多智能体强化学习工作负载上的平均硬件性能分析。性能分析取更新所有训练器三个子函数的平均值。

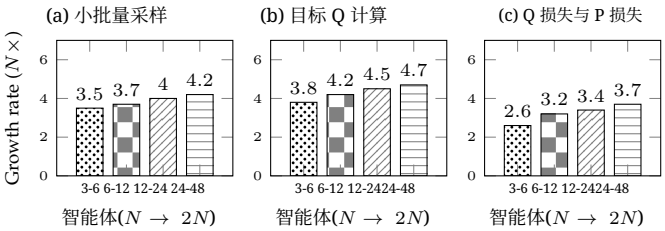


图 6：在三个多智能体强化学习工作负载中，智能体数量按 2× 扩展时，更新所有训练器 内部子函数的计算时间增长率平均值。

7 结论

在本研究中，我们对几种流行的多智能体强化学习算法进行了详细的端到端特征分析，尤其探讨了这些工作负载中的可扩展性瓶颈。实验分析揭示了导致可扩展性瓶颈的关键模块，并概述了克服这些瓶颈的架构指南。

致谢

本研究基于美国国家科学基金会资助的工作，资助编号为 CCF-2114415。

参考文献

[1] R. S. Sutton 和 A. G. Barto，《强化学习：导论》。麻省理工学院出版社，2018年。[2] R. Lowe、Y. I. Wu、A. Tamar、J. Harb、O. Pieter Abbeel 和 I. Mor- datch，“混合合作-竞争环境中的多智能体演员-评论家”，《神经信息处理系统大会》，第 30 卷，2017年。[3] J. Ackermann、V. Gabler、T. Osa 和 M. Sugiyama，“在多智能体领域中使用双重集中式评论家减少过高估计偏差”，《NeurIPS 深度强化学习研讨会》，2019年。[4] T. Haarnoja、A. Zhou、P. Abbeel 和 S. Levine，“软演员-评论家：使用随机演员的离策略最大熵深度强化学习”，载于《国际机器学习大会》。机器学习研究会议论文集，2018年，第 1861–1870 页。[5] H. Cho、P. Oh、J. Park、W. Jung 和 J. Lee，“FA3C: FPGA 加速深度强化学习”，载于《体系结构支持编程语言和操作系统国际会议》，2019年，第 499–513 页。[6] S. Iqbal 和 F. Sha，“面向多智能体强化学习的演员-注意力-评论家”，载于《国际机器学习大会》。机器学习研究会议论文集，2019年，第 2961–2970 页。[7] D. P. Kingma 和 J. Ba，“Adam：一种随机优化方法”，《arXiv 预印本 arXiv:1412.6980》，2014年。[8] V. Ramos，“Python 性能计数器 API”，https://pypi. org/project/performance-features/，2019年5月。[9] H. U. Sheikh 和 L. B¨ol¨oni，“针对兼顾个人与团队奖励的问题的多智能体强化学习”，载于《国际联合神经网络会议》。电气电子工程师学会，2020年，第 1–8 页。

[10] S. Lee、K. Kim、S. Oh、J. Park、G. Hong、D. Ka、K. Hwang、J. Park、K. Kang、J. Kim等，“一款基于 GDDR6 的存内加速器，采用 1ynm 工艺、1.25 V、8Gb、16Gb/s/pin，支持 1TFLOPS 乘加运算，并适用于多种深度学习激活函数”，载于国际固态电路会议，第 65 卷。电气电子工程师学会，2022年，第 1–3 页。[11] T. Kaler、N. Stathas、A. Ouyang、A.-S. Iliopoulos、T. Schardl、C. E. Leiserson 和 J. Chen，“通过快速采样和流水线化加速图神经网络的训练与推理”，《机器学习系统研讨会》，第 4 卷，第 172–189 页，2022年。[12] A. Jain、A. Phanishayee、J. Mars、L. Tang 和 G. Pekhimenko，“Gist：面向深度神经网络训练的高效数据编码”，载于《国际计算机体系结构研讨会》。电气电子工程师学会，2018年，第 776–789 页。