# GNNs Beyond HBM: Compressed, Bit-Exact Feature Streaming for Out-of-Core Training

Rajni Parshuram Pawar, Luke Logan, Xian-He Sun, and Anthony Kougkas
Illinois Institute of Technology, Chicago, IL, USA
rpawar4@hawk.illinoistech.edu, llogan@illinoistech.edu, sun@illinoistech.edu, akougkas@illinoistech.edu

***Abstract*—Graph neural networks (GNNs) increasingly train on node-feature matrices that exceed GPU high-bandwidth memory (HBM): `ogbn-papers100M` is $\sim$57 GB (52.9 GiB), beyond commodity 8–40 GB accelerators. Out-of-core feature streaming already trains such graphs but streams features *uncompressed* over PCIe. GnnPress is a compressed feature-store abstraction: it exposes an ordinary array of floats to the GNN while storing features compressed and materializing only a small resident window, so the working set is bounded by the window, not the matrix. On an NVIDIA A100-SXM4-40GB, GnnPress trains `ogbn-papers100M`, which out-of-memories an in-core layout. Lossless `zstd` is bit-identical to in-core every epoch ($\max|\Delta|=0$); GPU-native lossy cuSZp reaches $3.13\times$ while perturbing loss by at most $10^{-6}$. Mini-batch SGD reaches $61\%$ accuracy at a measured $1.7$ GiB peak GPU ($31\times$ below in-core).**

## I. Introduction

GNNs increasingly train on graphs whose node-feature matrices dwarf GPU HBM. The `ogbn-papers100M` benchmark [6] stores a 111-million-node, 128-dimensional matrix of roughly 57 GB (52.9 GiB), beyond the 8–40 GB capacity of commodity accelerators: an in-core layout out-of-memories the 40 GB A100 we use. Out-of-core feature streaming—features in host memory, only touched vertices copied into HBM—already trains such graphs; systems such as GIDS [1], Ginex [2], MariusGNN [3], GraphBolt [4], and HMGraph [5] take this route, but all stream features *uncompressed* across PCIe. Error-bounded scientific compressors (SZ [9], ZFP [10]) and GPU-initiated storage (BaM [12]) are routine in HPC yet unused on the GNN feature path. GnnPress asks: what if the streamed feature store were compressed—bit-exactly (`zstd` [8]) or with an error-bounded GPU codec (cuSZp [11])—at no accuracy cost?

## II. Method

GnnPress exposes an ordinary array of floats to the GNN while, underneath, storing features compressed and materializing only a small resident window. It is a two-level HBM cache: an uncompressed binary cache, where aggregation reads features, sits over a compressed cache, and level movements are decompression and compression events rather than plain copies. A double-buffered streaming window prefetches and evicts pages so decompression overlaps computation, and the compressed matrix spills to host DRAM when it exceeds HBM; the resident working set is proportional to the window, not the matrix, and a PageRank predictor keeps hot pages resident. We evaluate on an NVIDIA A100-SXM4-40GB with a SIGN-style [7] one-hop model that precomputes mean aggregation once, then trains a two-layer MLP. Two datasets anchor the study: `ogbn-arxiv`, small enough to fit in HBM as a bit-exact reference, and `ogbn-papers100M`, which does not. Storage uses `zstd` (lossless) or cuSZp (GPU-native lossy).

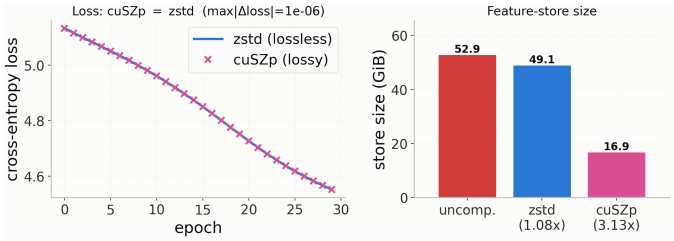



Fig. 1. Full-batch GD, loss-only comparison. Left: lossy cuSZp gives the *same* training loss as lossless `zstd` ($\max|\Delta\text{loss}|=10^{-6}$). Right: the 52.9 GiB matrix stores at 49.1 GiB (`zstd`, $1.08\times$) vs. 16.9 GiB (cuSZp, $3.13\times$).

## III. Results

**Correctness.** With lossless `zstd`, the streamed run is bit-identical to the in-core run at every epoch: the maximum absolute difference in training loss, training accuracy, and validation accuracy is exactly zero. This holds under mini-batch SGD across four configurations, which shrinks the resident window from 32 MB to 2 MB on arxiv. Because reads return bit-identical bytes and gradients accumulate in a fixed order, the model output is provably unchanged by streaming.

**Compression is the novelty.** On the 52.9 GiB papers100M matrix, lossless `zstd` is only modestly effective on dense floats, reducing the store to 49.1 GiB ($1.08\times$). GPU-native cuSZp reaches 16.9 GiB ($3.13\times$) while leaving training essentially unperturbed: the lossy loss trajectory tracks the lossless one to within $10^{-6}$ over thirty epochs (Fig. 1). At equal host budget the $3.13\times$ codec holds three times the graph and moves a third of the bytes over PCIe.

**It learns, at bounded GPU memory.** Mini-batch SGD trains papers100M to $61.1\%/61.2\%$ train/validation accuracy over 172 classes, $\sim 9.8\times$ the majority-class floor of full-batch descent, at a measured peak GPU footprint of 1.7 GiB

TABLE I
MEASURED RESULTS ON `ogbn-papers100M` (52.9 GiB, A100-40GB).

| Metric | `zstd` (lossless) | cuSZp (lossy) |
|---|---|---|
| Store ratio | 1.08× | 3.13× |
| Peak GPU | 1.7 GiB (1734 MiB) | 2.8 GiB (2858 MiB) |
| vs. in-core (52.9 GiB) | 31× | 19× |
| Loss $\Delta$ vs. `zstd` | — | $\leq 10^{-6}$ |
| Bit-exact (arxiv) | yes ($\max\|\Delta\|=0$) | — |

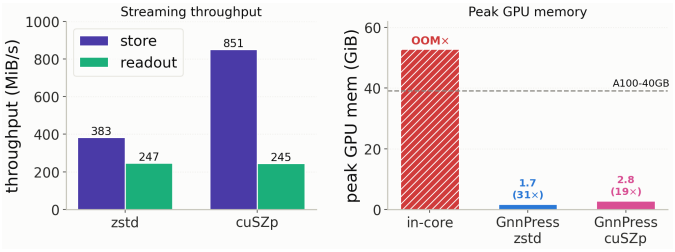



Fig. 2. Left: streaming throughput—readout pinned at ∼246 MiB/s (fetch-path bound, *not* codec-bound). Right: *measured* peak GPU—in-core needs the whole 52.9 GiB resident (OOMs the 40 GB A100); GnnPress trains in 1.7 GiB (`zstd`) / 2.8 GiB (cuSZp).

(lossless; 2.8 GiB with cuSZp, whose GPU compressor costs ∼1.1 GiB more)—31× below the 52.9 GiB an in-core layout needs, which overflows the 40 GB card (Fig. 2, Table I). Peaks come from `cudaMemGetInfo`: 1734 and 2858 MiB on papers100M, 822 MiB on arxiv, 1652 MiB on products. This peak is dominated not by the bounded 64 MiB window but by an $O(N)$ labels array reaching 847 MiB.

**What the results do not show.** The validation split selects every tenth node by index, not OGB's temporal split, so accuracies are not leaderboard-comparable. The lossless ratio on dense features is small, so `zstd`'s benefit is streaming reach, not compression. Readout is pinned near 246 MiB/s, ∼100× below PCIe (fetch-path bound), so we make no speedup claim and lack a zero-copy baseline for fair throughput comparison.

## IV. CONCLUSION

GnnPress shows that the feature stream underlying out-of-core GNN training can be compressed without disturbing the computation—bit-exact under a lossless codec, accuracy-neutral under a 3.13× lossy codec—while training beyond HBM at a bounded, measured GPU footprint on a real A100. Next steps are a zero-copy baseline to quantify throughput fairly and a compressed label store to remove the $O(N)$ term dominating peak memory, turning a correctness-and-capacity result into an end-to-end performance story.

## ACKNOWLEDGMENT

Anthropic Claude assisted with language editing; the authors are responsible for all content.

## REFERENCES

[1] J. B. Park *et al.*, "Accelerating sampling and aggregation operations in GNN frameworks with GPU-initiated direct storage accesses," *Proc. VLDB Endow.*, 2024.
[2] Y. Park *et al.*, "Ginex: SSD-enabled billion-scale GNN training on a single machine via provably optimal in-memory caching," *Proc. VLDB Endow.*, 2022.
[3] R. Waleffe *et al.*, "MariusGNN: Resource-efficient out-of-core training of graph neural networks," in *EuroSys*, 2023.
[4] Deep Graph Library Team, "GraphBolt: A scalable data loading and graph sampling framework for GNN training," DGL 2.0, 2024.
[5] "HMGraph: Boosting GNN training on hierarchical memory via coordinated cache," in *Int. Conf. Parallel Processing (ICPP)*, 2025.
[6] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," in *NeurIPS*, 2020.
[7] F. Frasca *et al.*, "SIGN: Scalable inception graph neural networks," arXiv:2004.11198, 2020.
[8] Y. Collet and M. Kucherawy, "Zstandard compression and the application/zstd media type," RFC 8878, 2021.
[9] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *IEEE IPDPS*, 2016.
[10] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graph.*, 2014.
[11] Y. Huang *et al.*, "cuSZp: An ultra-fast GPU error-bounded lossy compression framework," in *SC*, 2023.
[12] Z. Qureshi *et al.*, "GPU-initiated on-demand high-throughput storage access in the BaM system architecture," in *ASPLOS*, 2023.