Evaluation of Graph Sampling and Partitioning for Edge Classification and Tracking

Alina Lazar, on behalf of ExaTrkX

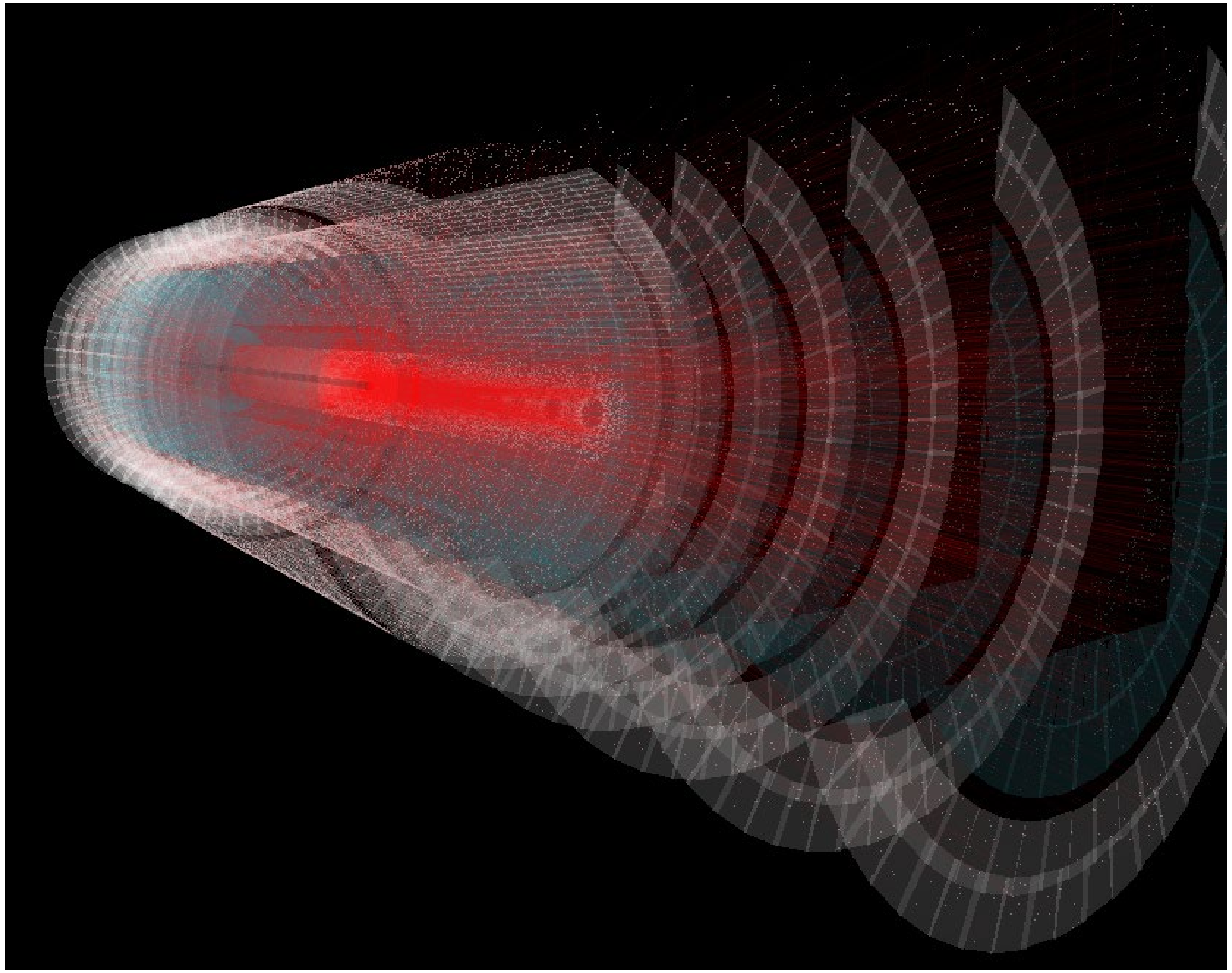
Paolo Calafiura, Xiangyang Ju, Ivan Ladutska, Daniel Murnane, Tuan Minh Pham

TrkX

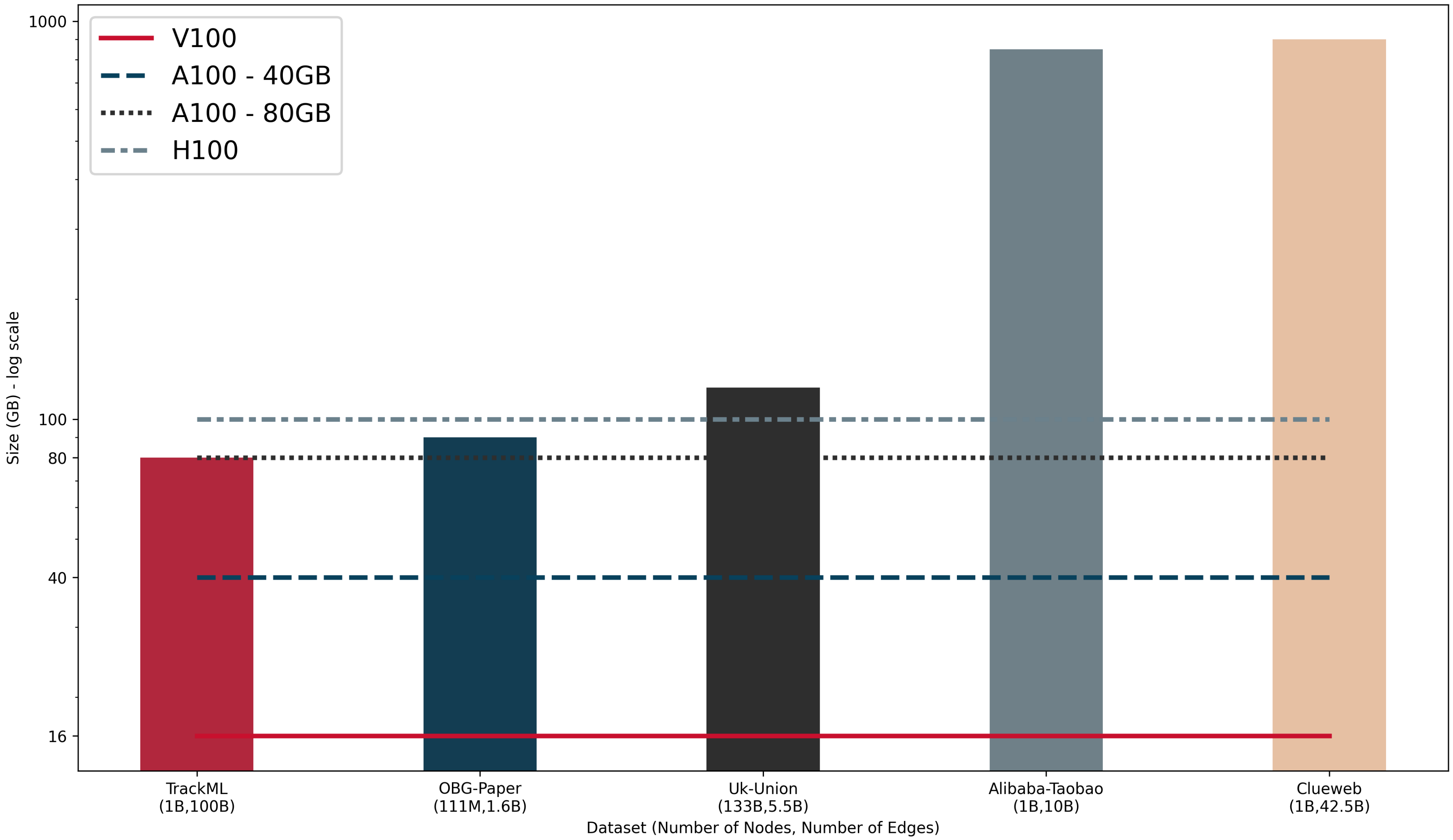
YOUNGSTOWN STATE UNIVERSITY

Graph Neural Networks (GNNs) for Tracking

- GNN-based track pattern reconstruction is becoming the tool for “Connecting the Dots”
- Focus: **Scaling GNN training**
- Training GNNs is challenging due to the irregular nature of graph data
- It takes a long time to train
- Scaling to large graphs that exceed the **memory** capacity of a single device is even more challenging



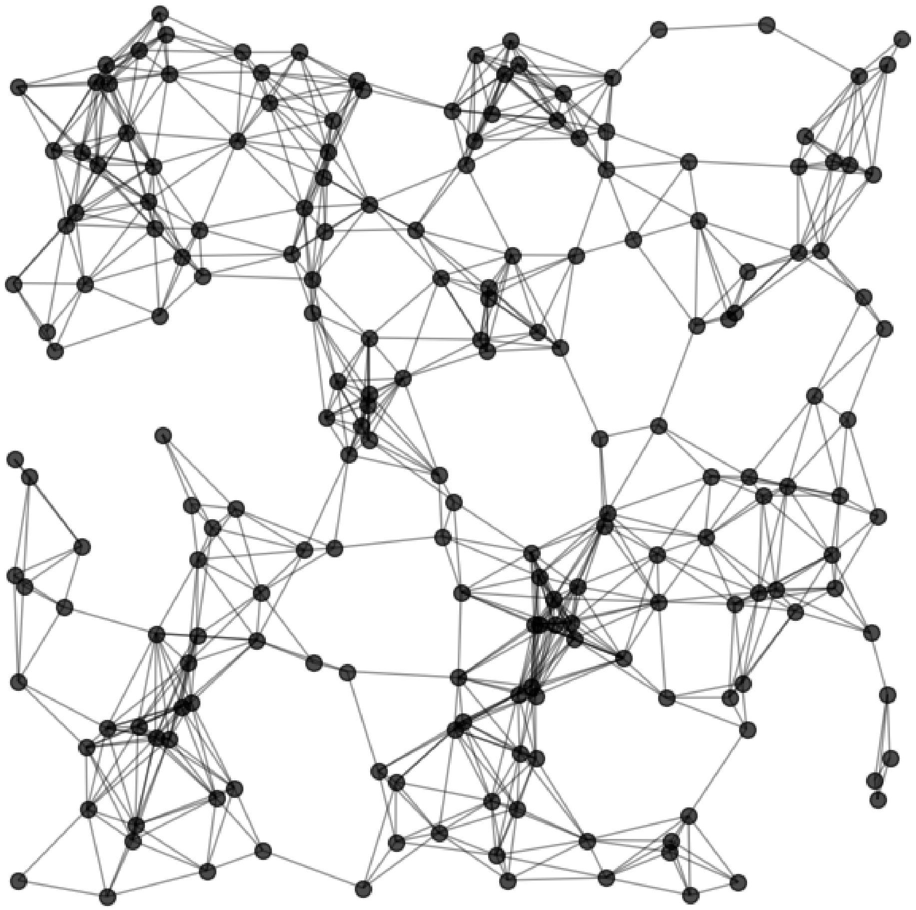
Memory Requirements for Training GNNs on Large Graphs



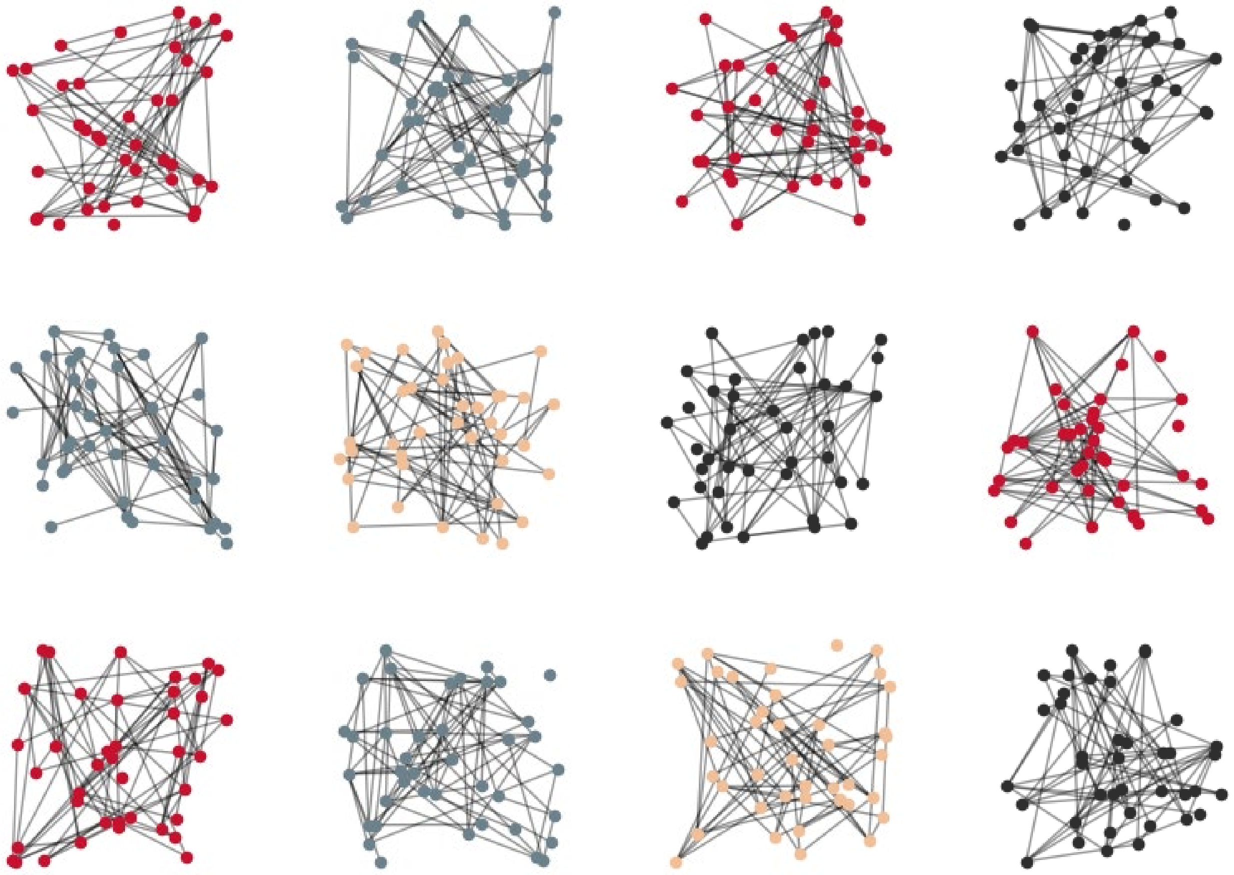
[Legion: Automatically Pushing the Envelope of Multi-GPU System for Billion-Scale GNN Training]

Training GNNs on Large Graphs

ClueWeb (1B nodes, 42.5B edges)

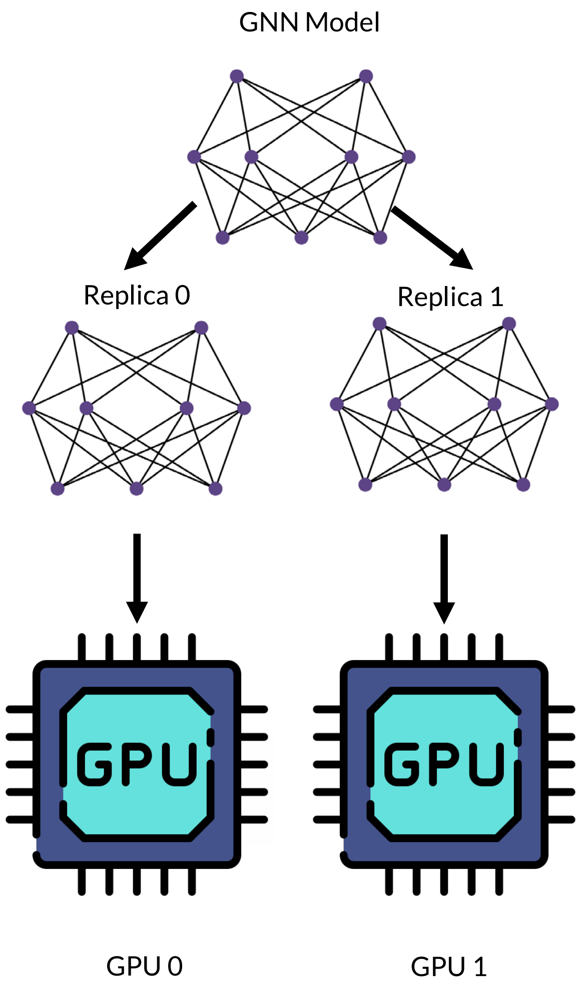


TrackML (1B nodes, 100B edges)
10k events, 100k nodes, 10 million edges

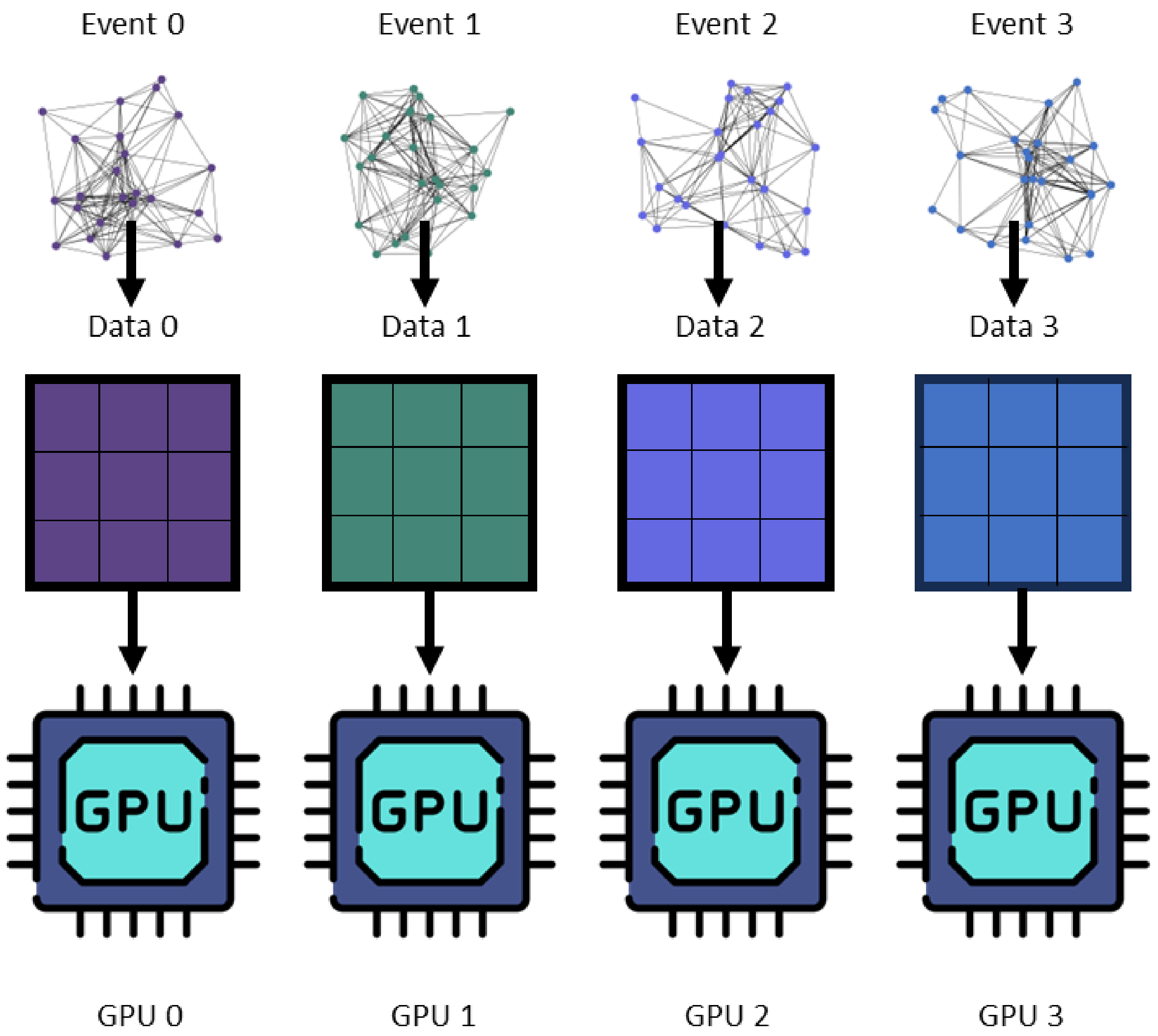


The ClueWeb22 Dataset (lemurproject.org)

Parallelization Schemes – Distributed Data Parallelism (DDP)

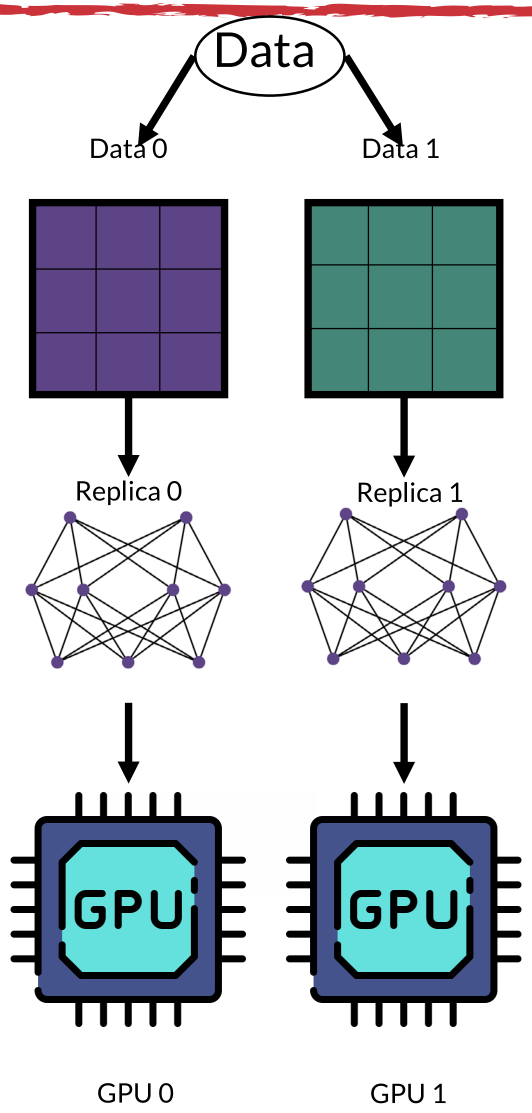


DDP Initialization Model

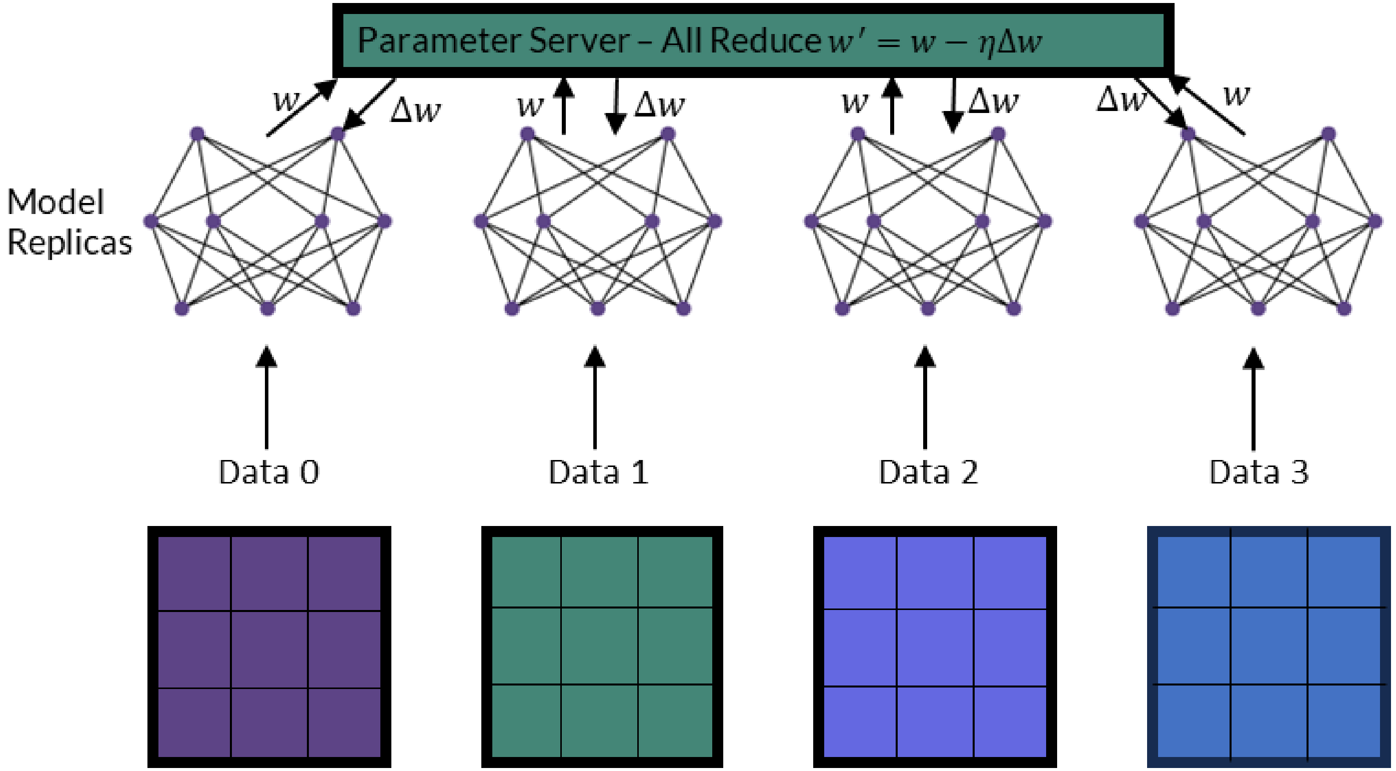


DDP Initialization Data

Parallelization schemes – Distributed Data Parallelism (DDP)



Forward

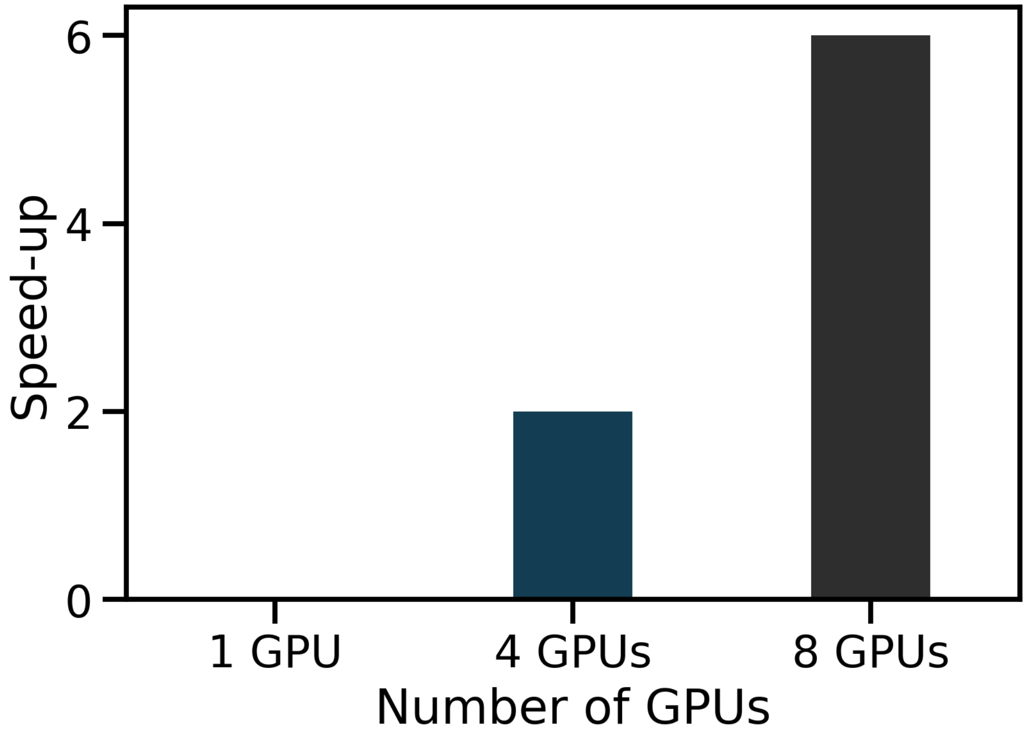


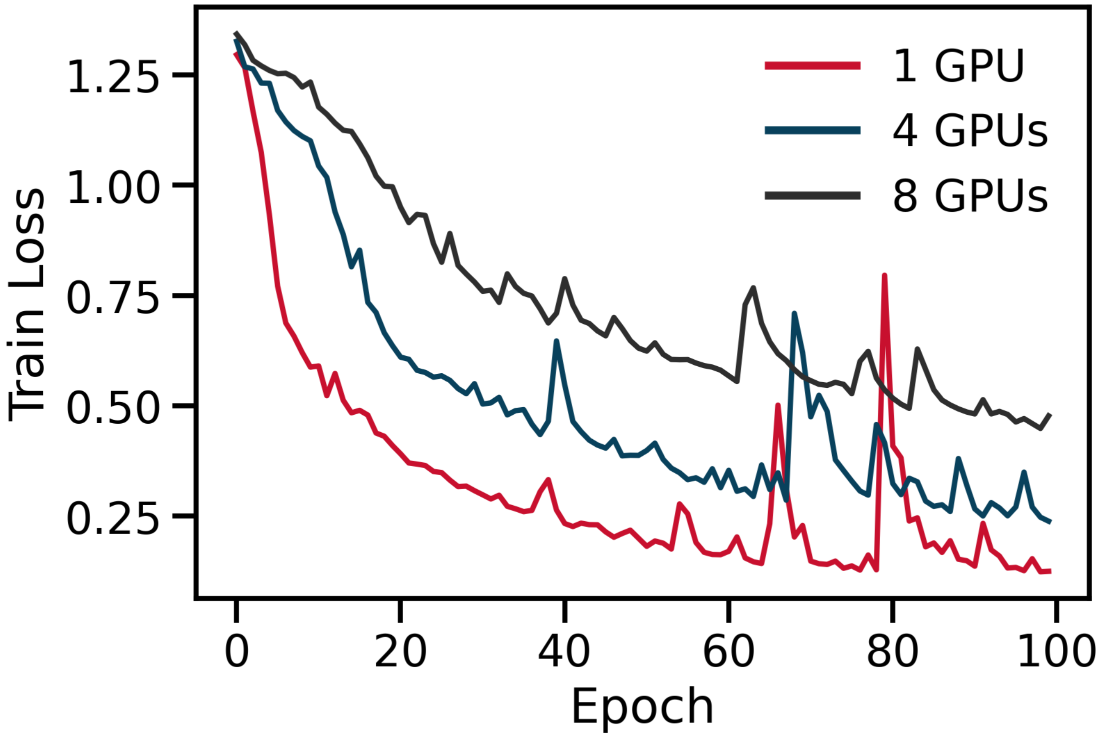
Backward

TrackML Dataset Distributed Data Parallelism Training

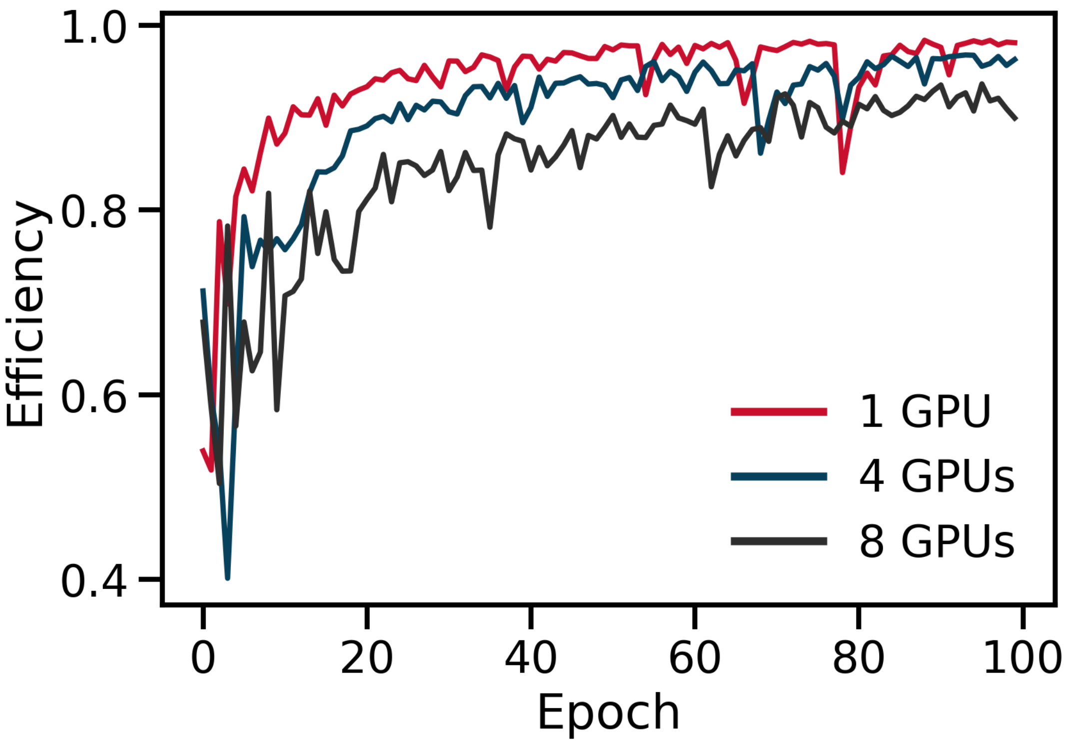
- Experiments were run on A100s nodes with 4 GPUs per node and 80 GB of memory per GPU
- GPU memory utilization of 88.65%

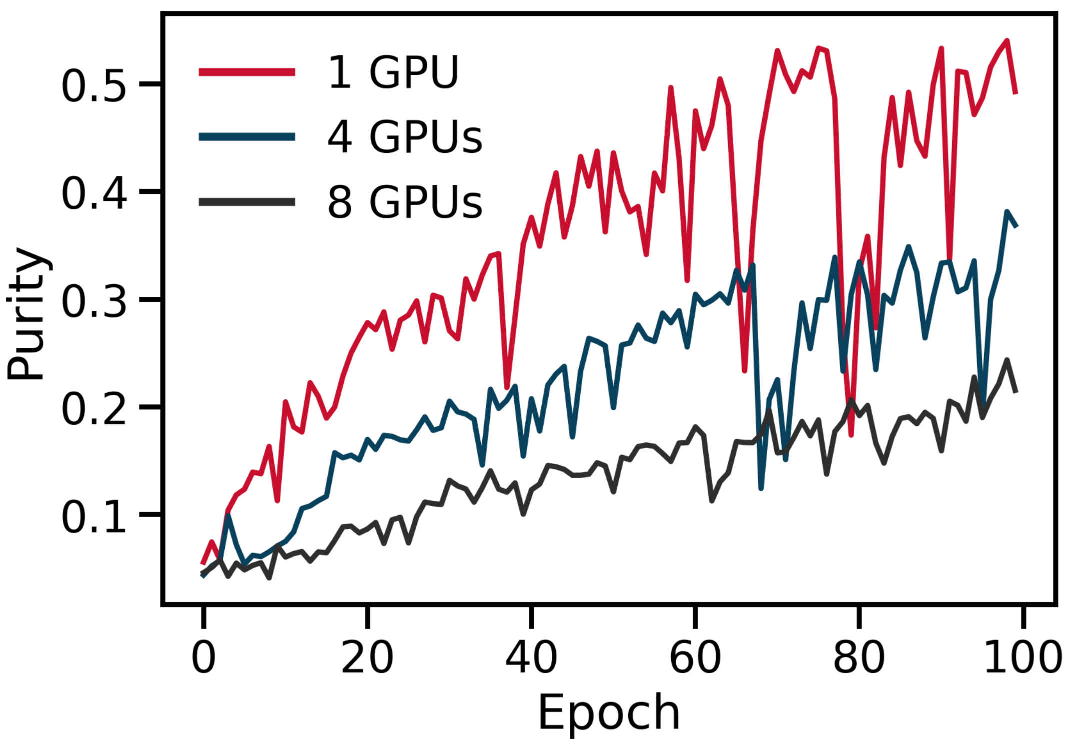
- 80 events for training, 10 for validation and 10 for testing
- Average number of nodes 84k ± 9k
- Average number of edges 2.6m ± 600k





Efficiency and Purity - DDP

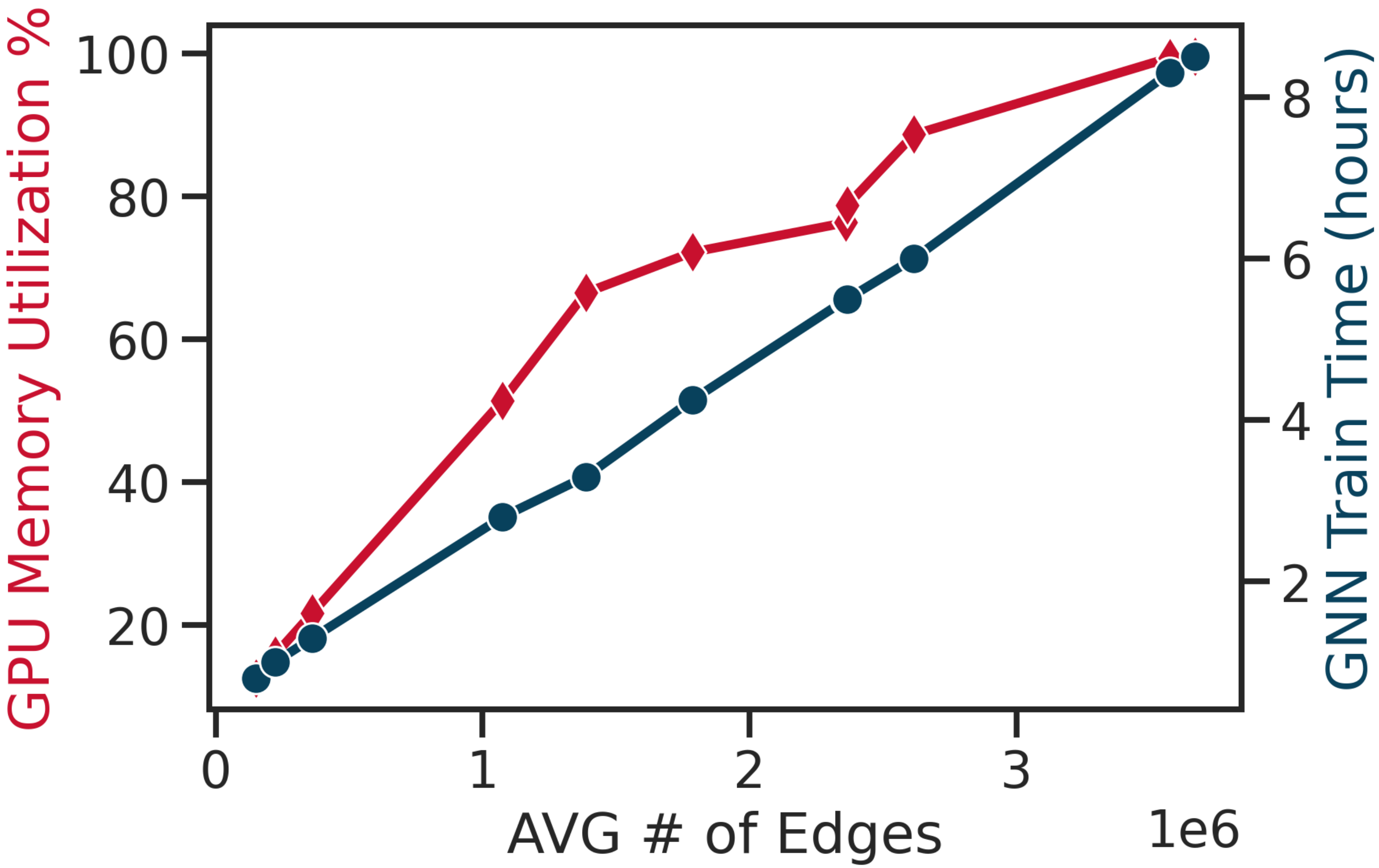




Using the DDP strategy degrades the physics performance in terms of both efficiency and purity.

This talk will explore solutions to address this scaling problem.

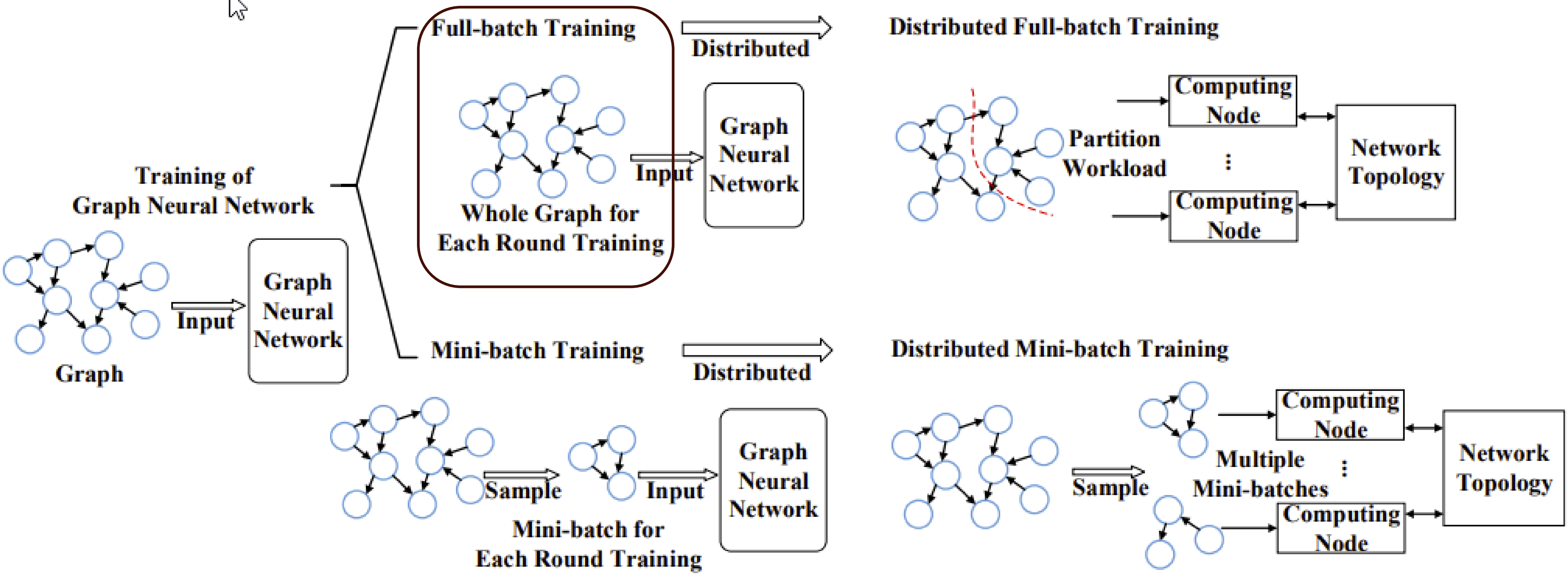
Memory Requirements for Training GNNs on TrackML



Problem: scaling to large event graphs that exceed the memory capacity of single GPUs

Solution: breaking the graphs into smaller subgraphs that can fit in the memory of single GPUs

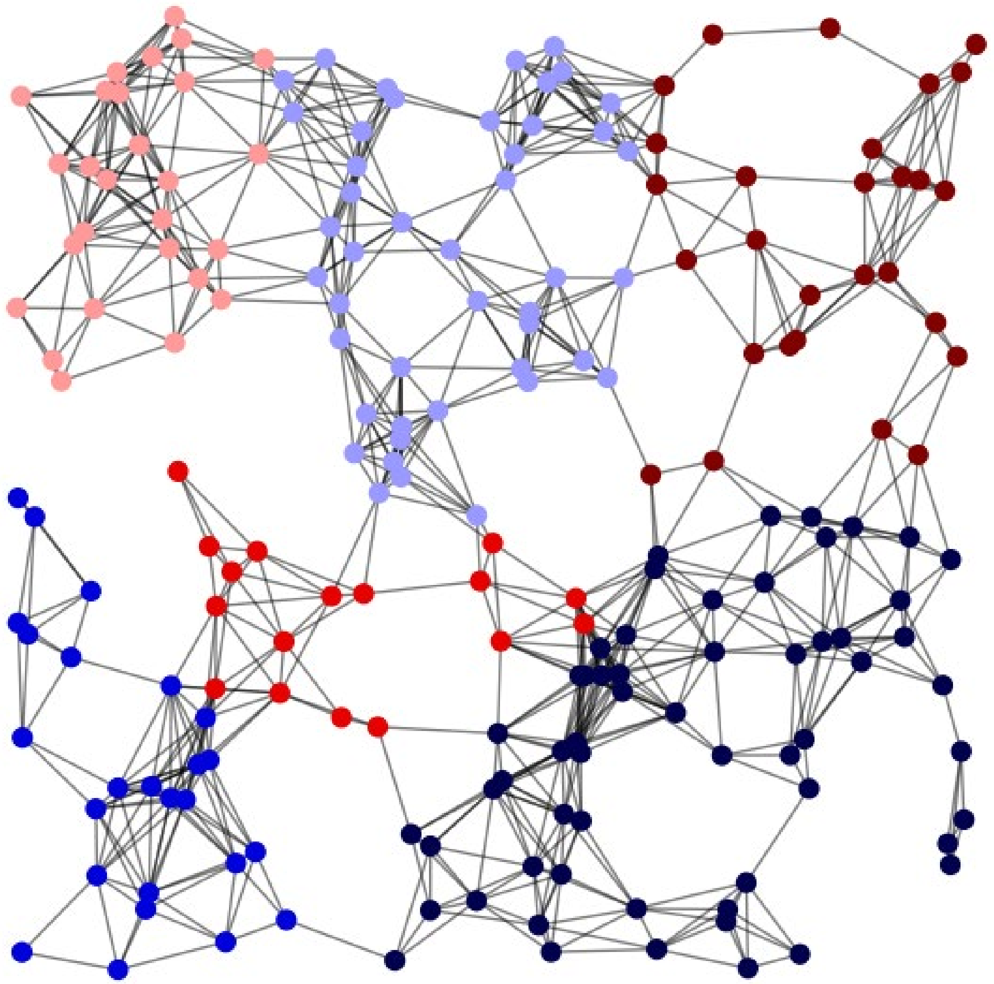
Partitioning versus Mini-Batch Schemes for GNN Training



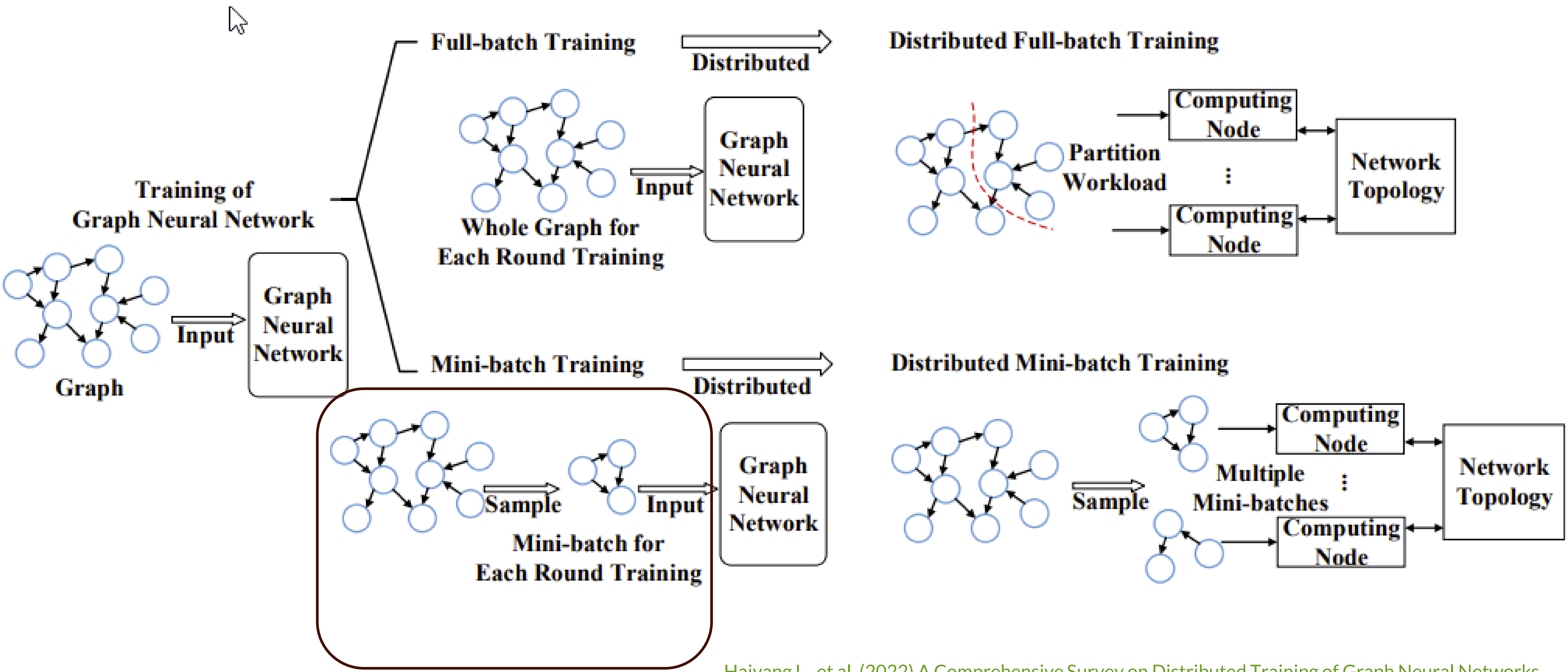
Haiyang L., et al. (2022) A Comprehensive Survey on Distributed Training of Graph Neural Networks

Graph Partitioning GNN Training

- Samples are partitioned across batches because the graph doesn't fit in the device's memory
- Each node and/or edge belongs to one partition
- There is no overlap between partitions
- Colors indicate partition



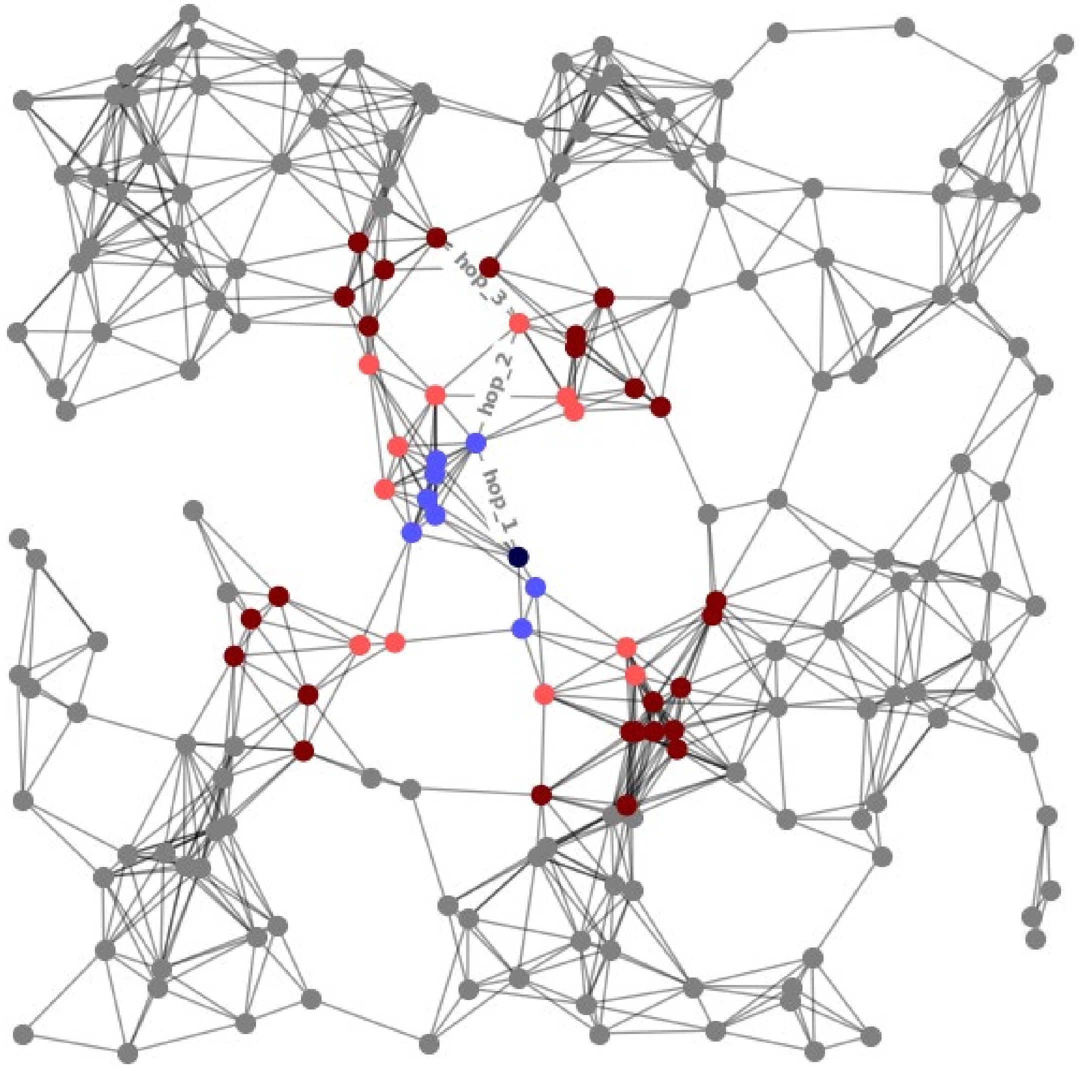
Partitioning versus Mini-Batch Schemes for GNN Training



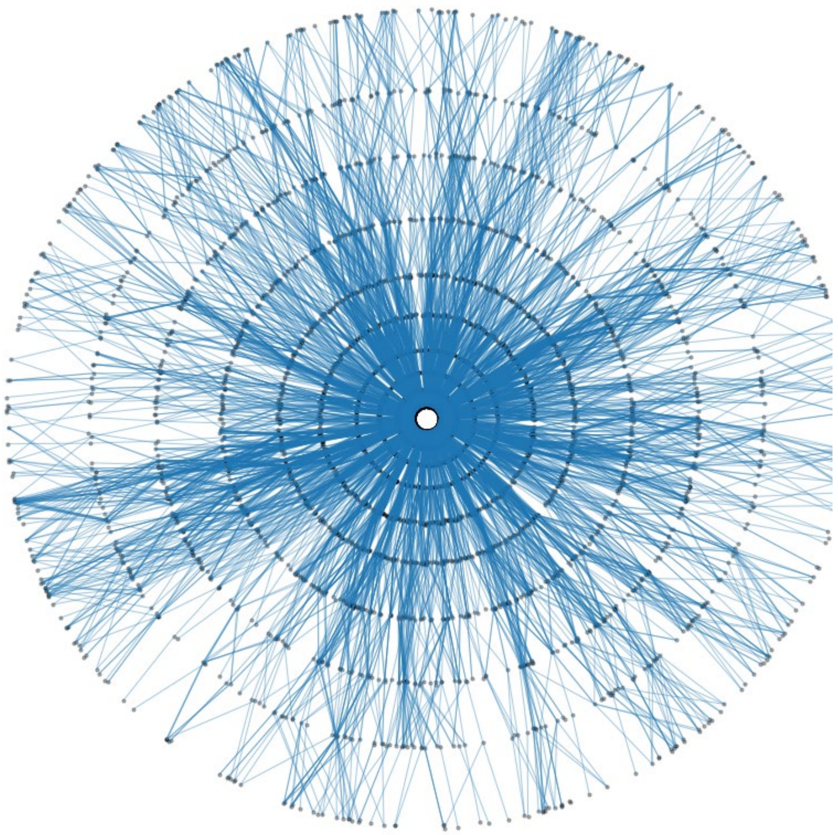
[Haiyang L., et al. (2022) A Comprehensive Survey on Distributed Training of Graph Neural Networks]()

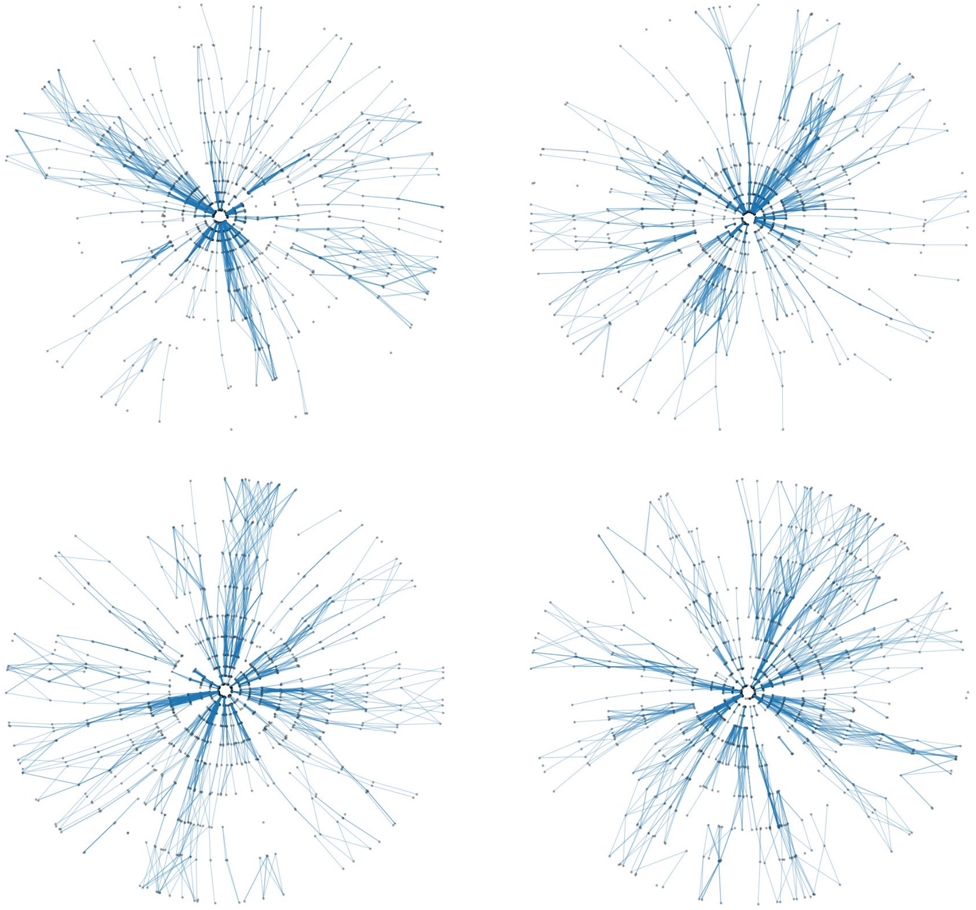
Mini-batch GNN Training

- Sample-based training first samples the graph to build mini-batches
- Sampling starts by selecting random subsets of nodes, edges, or subgraphs to be included in the mini-batch
- In a GNN model with *n* layers, each mini-batch includes the input features of the **n-hop neighborhood** of those target nodes
- There is overlap between the mini-batches
- Once the mini-batches are generated, distributed training can be applied

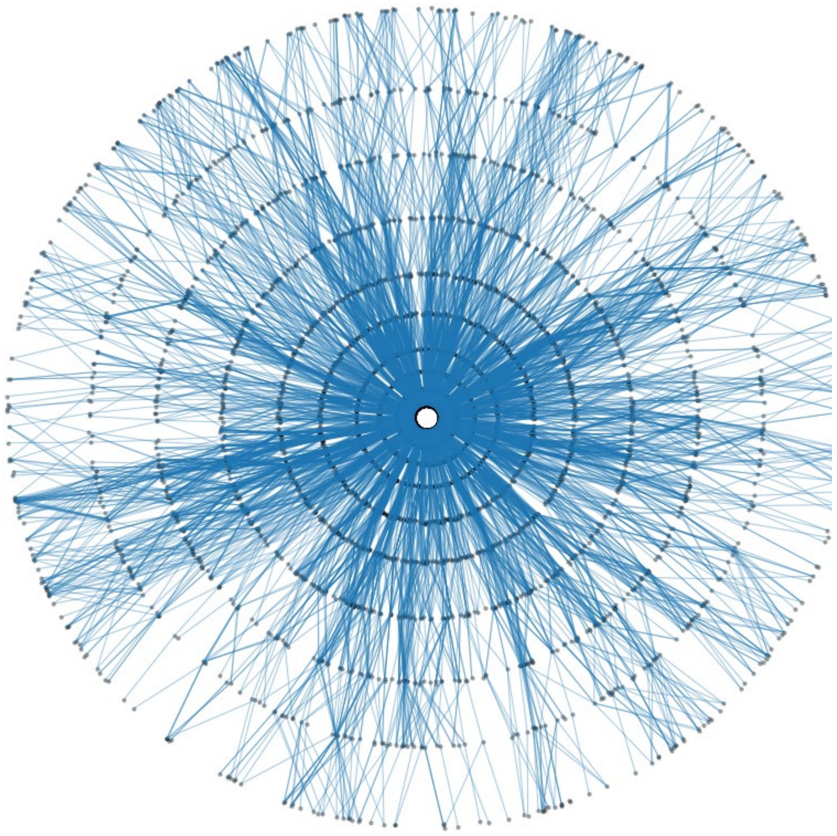


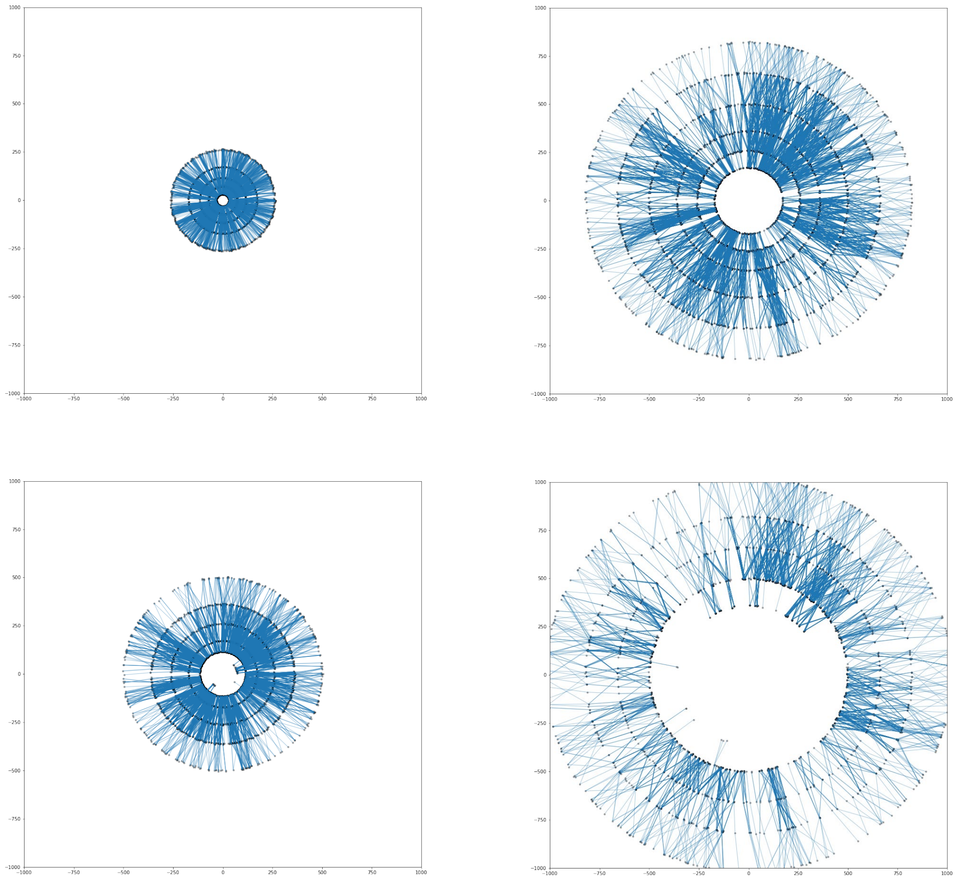
Graph Partitioning





Subgraph Sampling

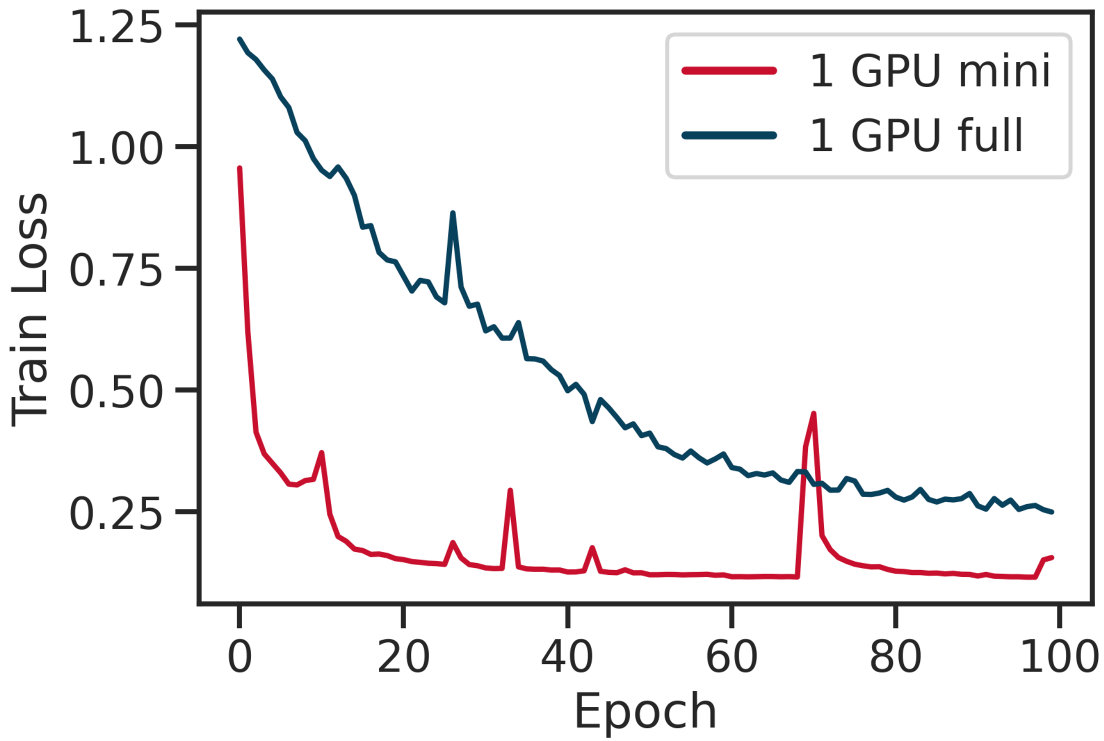


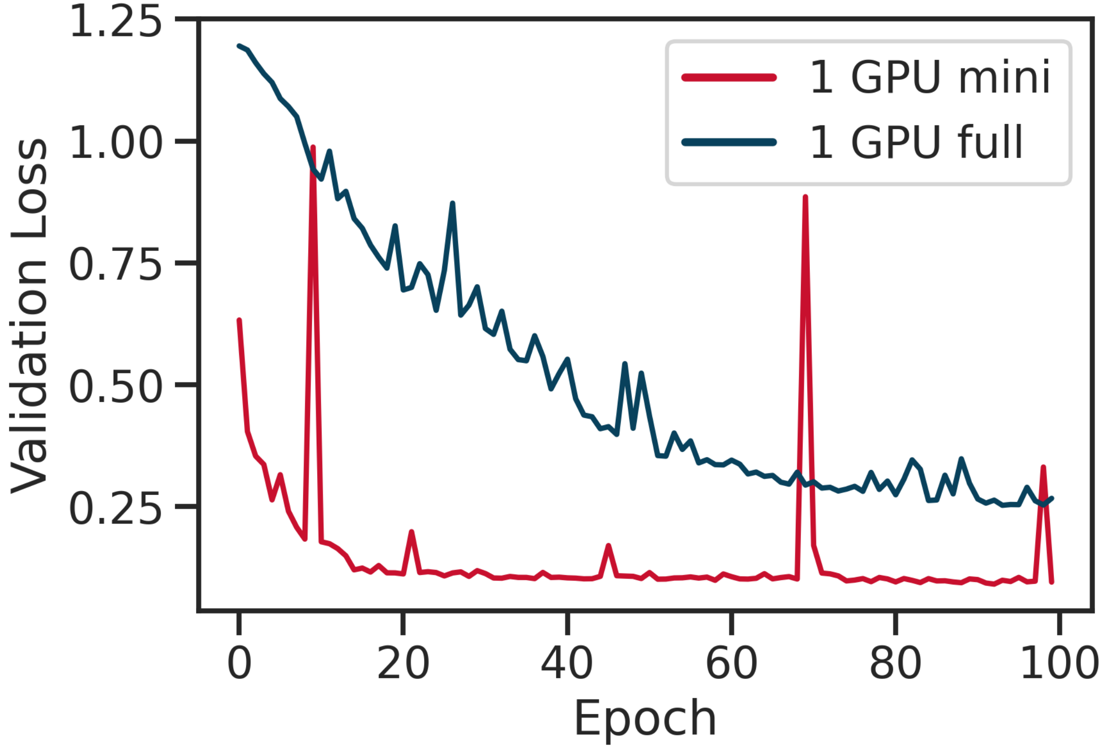


Training and Validation Loss Results for Mini-batch GNN training

- 80 events for training, 10 for validation, and 10 for testing
- Average number of nodes 84k ± 9k
- Average number of edges 150k ± 30k
- Number of nodes in subgraph: 2048

Full-batch – 80 batches
Mini-batch – 3294 batches

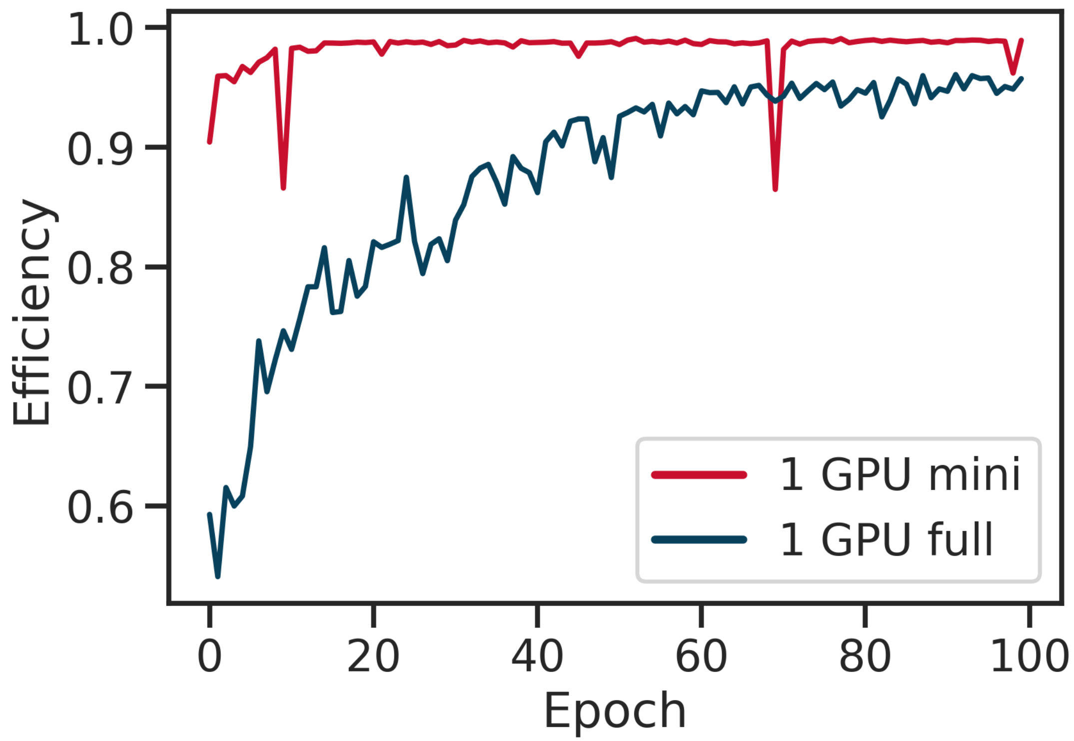




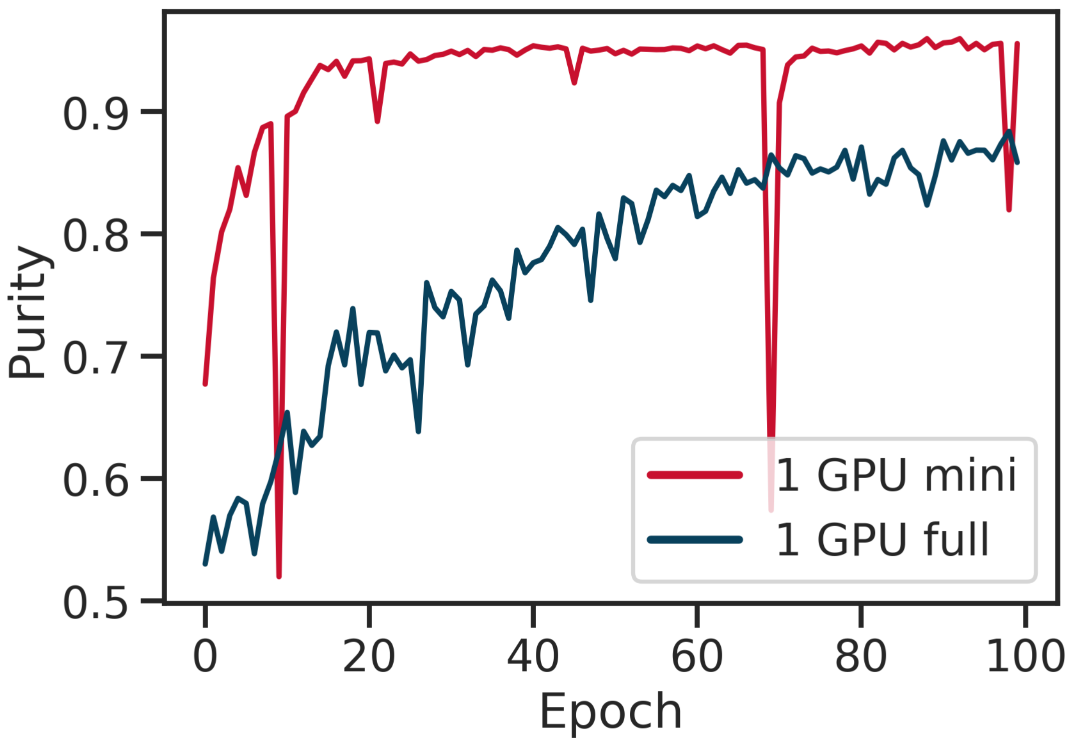
Efficiency and Purity - Full vs Mini-batch GNN training

Mini-batch training produces better models than full-batch training

Full-batch best efficiency: 0.957
Mini-batch best efficiency: 0.989



Full-batch best purity: 0.856
Mini-batch best purity: 0.956

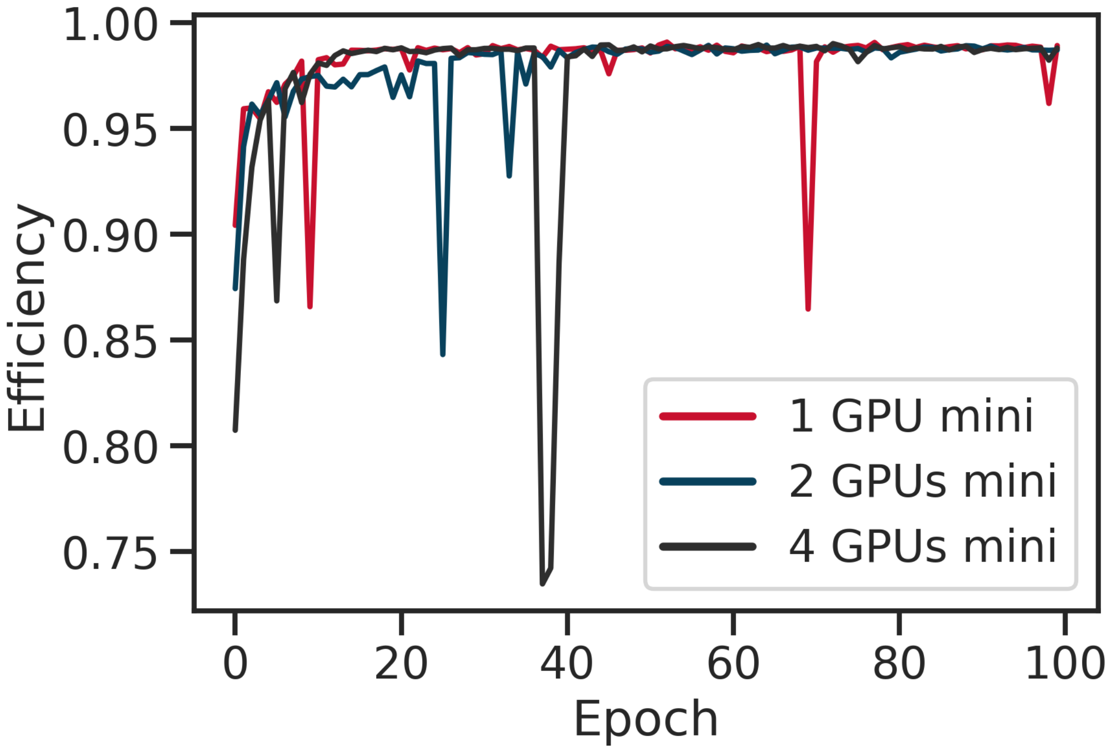


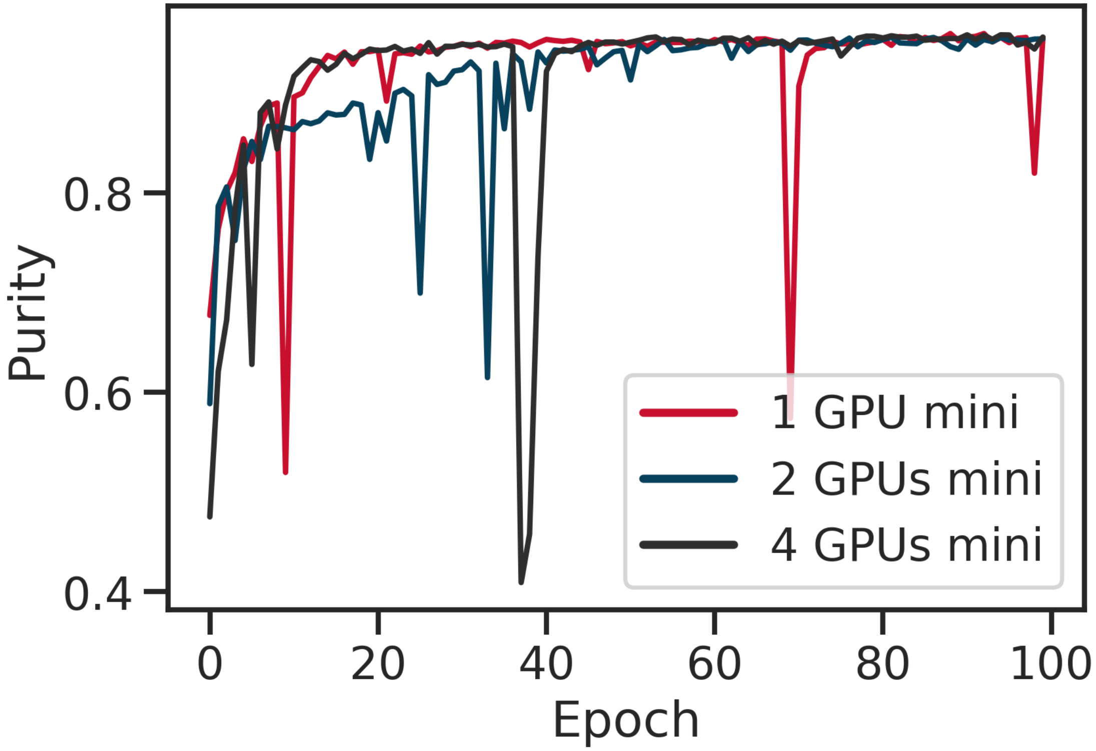
Efficiency and Purity - Mini-batch DDP GNN training

Mini-batch best efficiency:
- 1 GPU – 0.989
- 2 GPUs – 0.987
- 4 GPUs – 0.988

Mini-batch sizes:
- 1 GPU – 3.2k
- 2 GPUs – 1.6k
- 4 GPUs – 0.8k

Mini-batch best purity:
- 1 GPU – 0.956
- 2 GPUs – 0.954
- 4 GPUs – 0.956





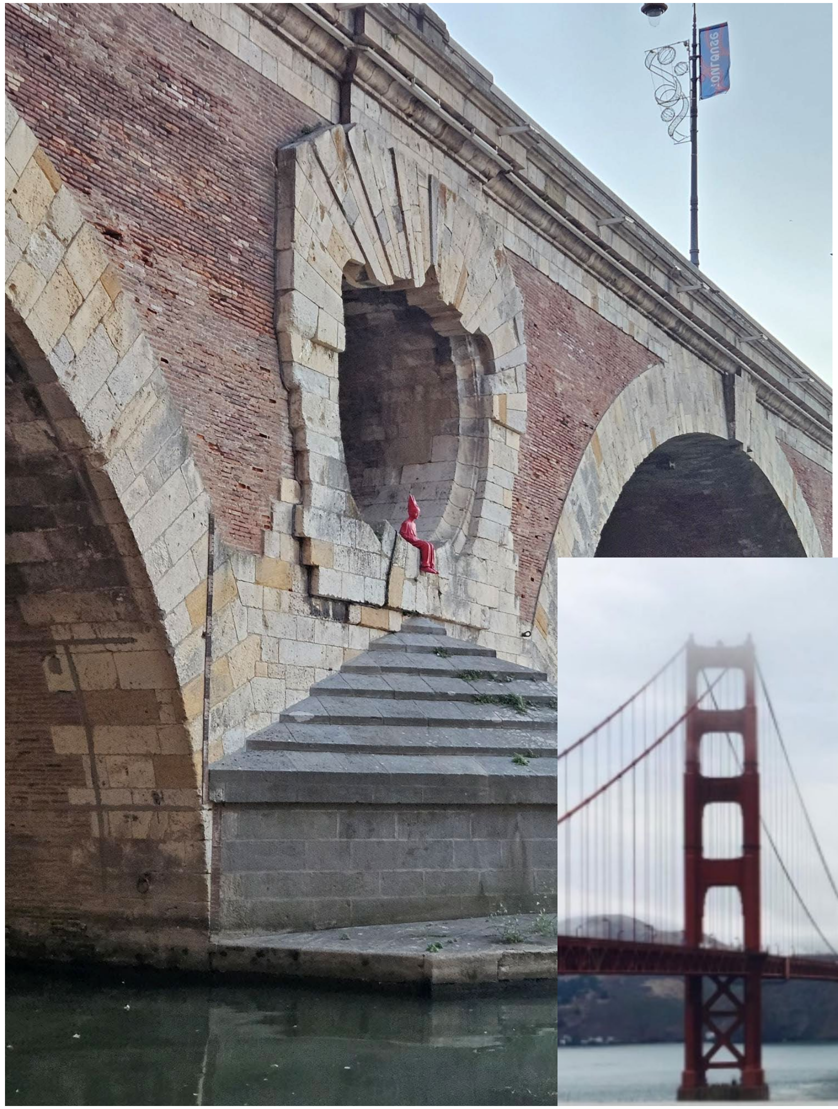
Mini-batch training scales better to multi-GPUs than full-batch training

Conclusions and Future Work

- Graph sampling (mini-batches) improves the performance of GNN training significantly
- Once the mini-batches are generated, distributed training can be applied
- Scaling graph sampling-based training requires:
  - algorithms that can form mini-batches without losing information or generating excessive redundant work
  - systems that can execute these algorithms efficiently

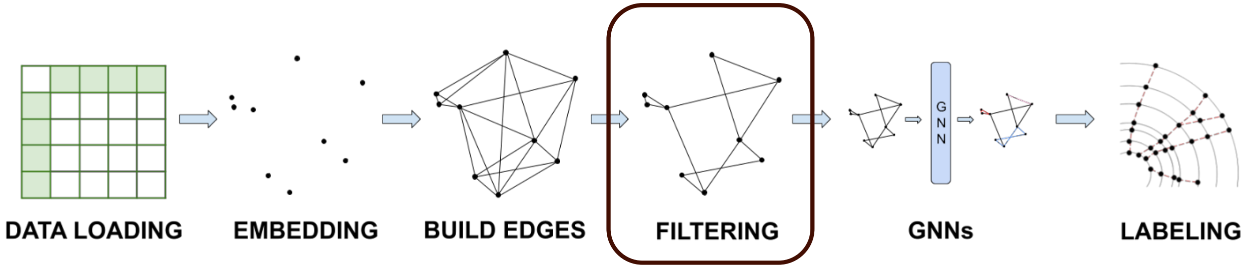
- Further testing and tuning of the sampling and partitioning methods is needed
- Sampling and data loading are expensive
- GPU-based sampling has the potential to significantly reduce end-to-end training time
- How to distribute and store the graph data and how to transfer it in and out of the GPUs to minimize the data transfers?

Thank you!



[Volée en juillet sous le Pont-Neuf, la sculpture de l'enfant au bonnet d'âne va revenir à Toulouse | Actu Toulouse]

Memory Requirements for Training the GNN Pipeline



- Filtering (MLP) is used to reduce the number of edges
- To train the GNN with 10k events takes ~2 weeks
- Increasing the event graph size (the number of edges) increases the GPU's memory utilization

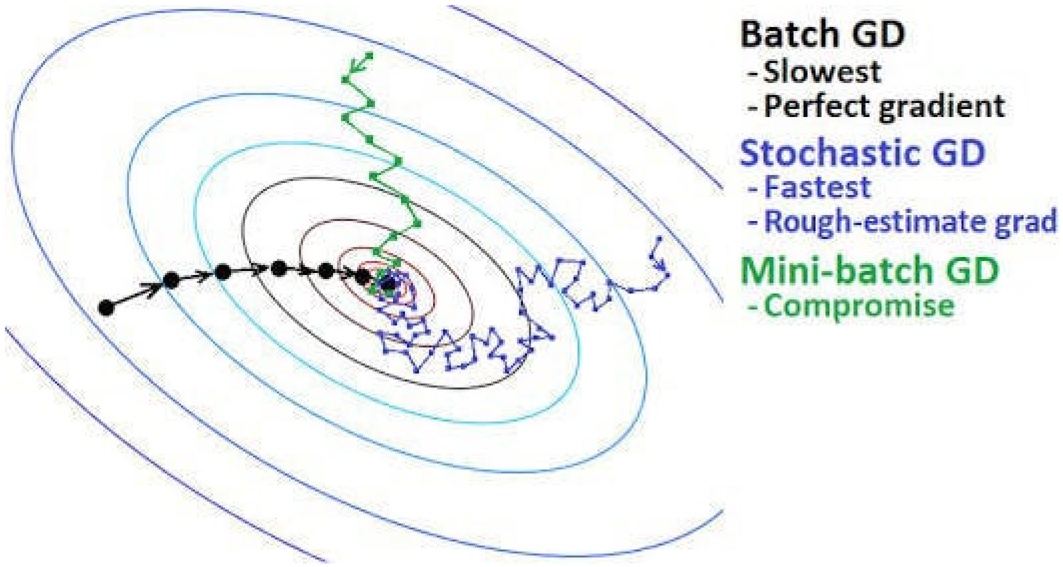
Learning Rate Scaling Rule

Learning Rate Scaling Rule: When the batch size is multiplied by *k*, multiply the learning rate by *k*.

Mini-batch Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \mathcal{B}} \nabla l(x, w_t)$$

η is the learning rate
$\mathcal{B}$ is the mini-batch



Typical practice/suggestion:

- Keep local batch size per worker the same
- Increase the global batch size linearly with the number of devices
- Increase the learning rate proportionally: $lr_{scale} = lr\ * num\_devices$

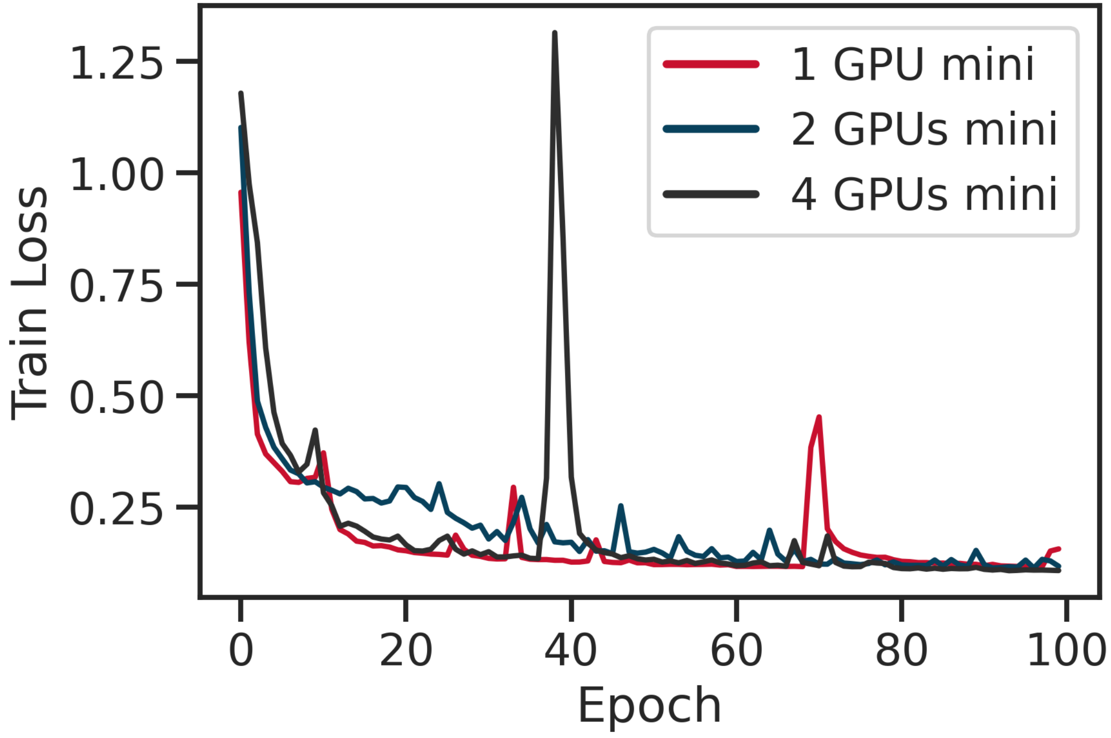
McCandlish, Sam, et al. "An empirical model of large-batch training." *arXiv preprint arXiv:1812.06162* (2018).

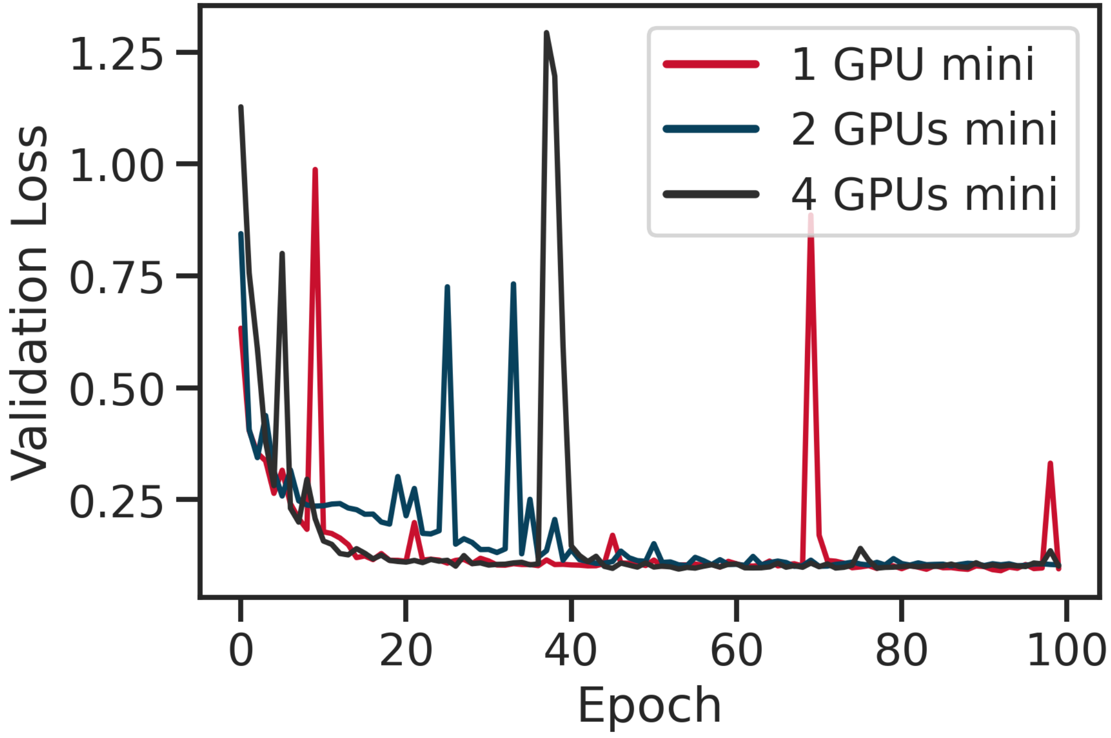
The Relationship between Batch Size and Learning Rate

- Batch size is a **hyperparameter**
- A larger batch size allows computational **speedups** from the parallelism of GPUs
- Too large of a batch size leads to poor generalization
- A batch equal to the entire dataset guarantees convergence to the global optima
- A smaller batch size has been shown to have **faster convergence**
- The downside of using a smaller batch size is that the model is not guaranteed to converge to the global optima

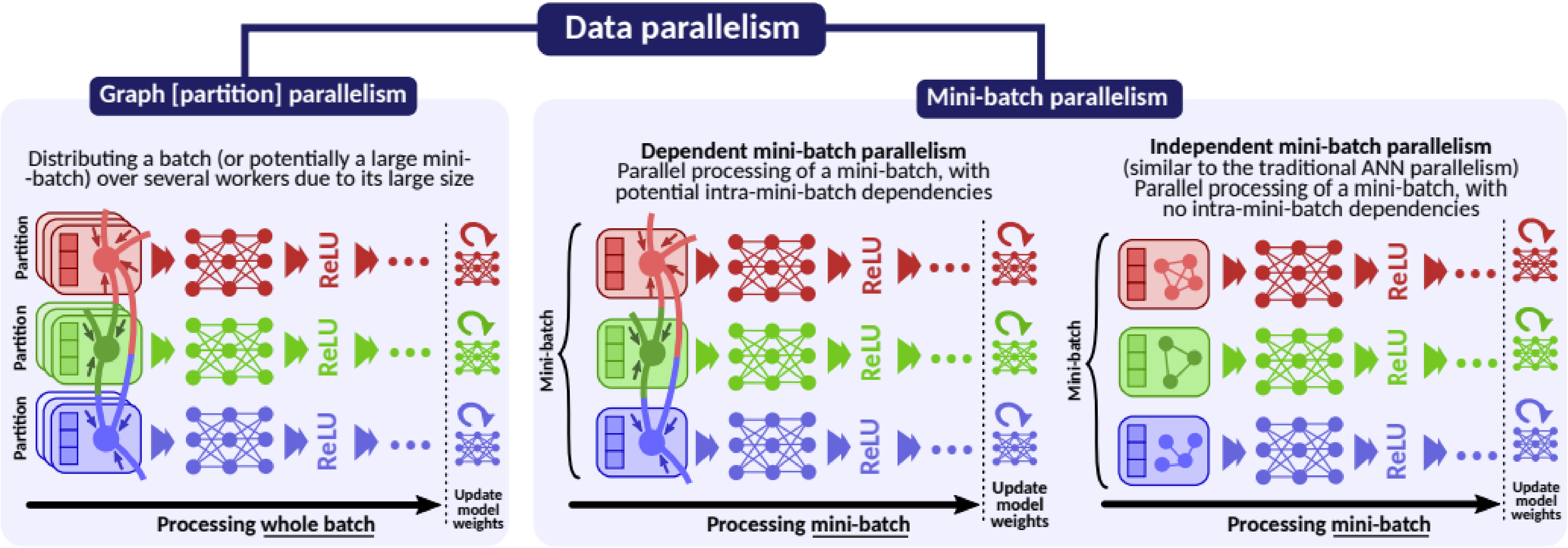
ImageNet Dataset

Training and Validation Loss Results





Overview of Data Parallelism



Besta, M., and Hoefler, T. (2022). Parallel and Distributed Graph Neural Networks: An In-Depth Concurrency Analysis.

Graph vs Mini-batch Parallelism

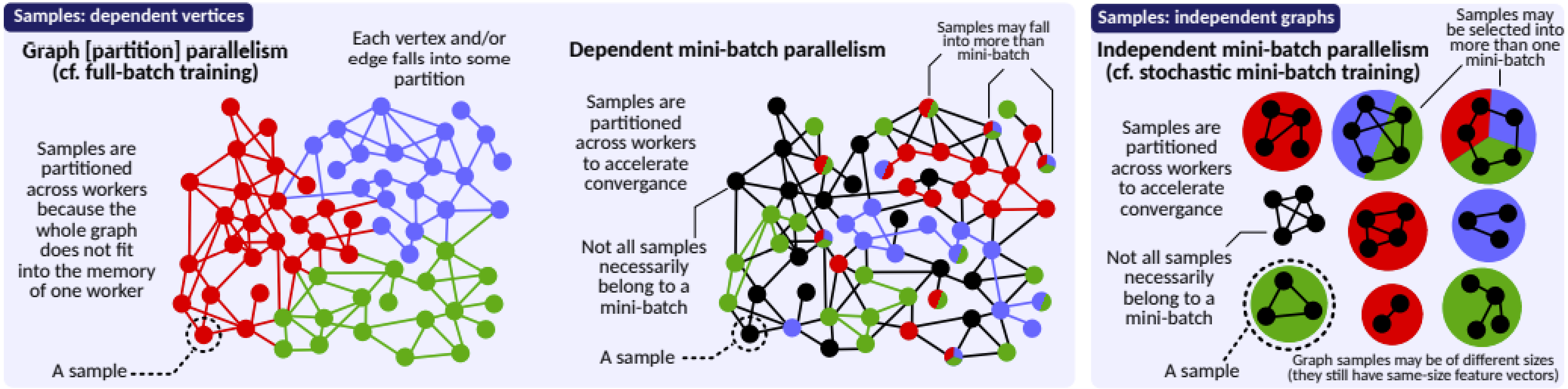
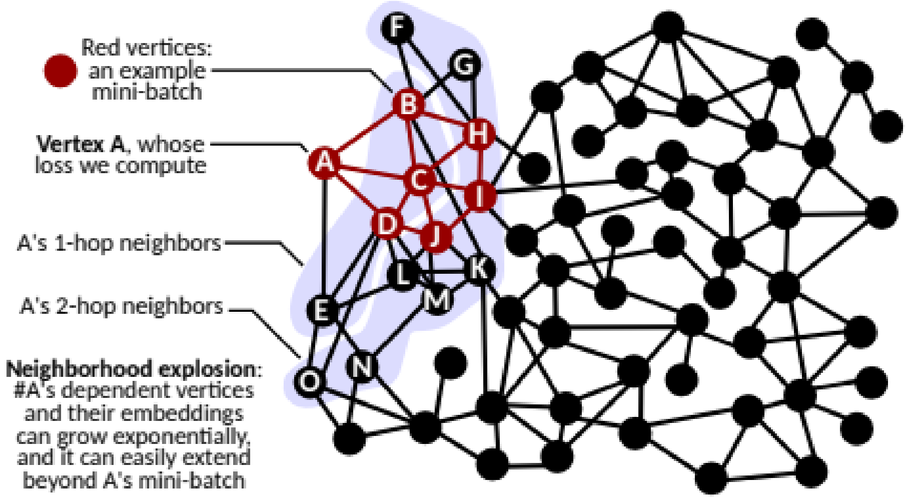
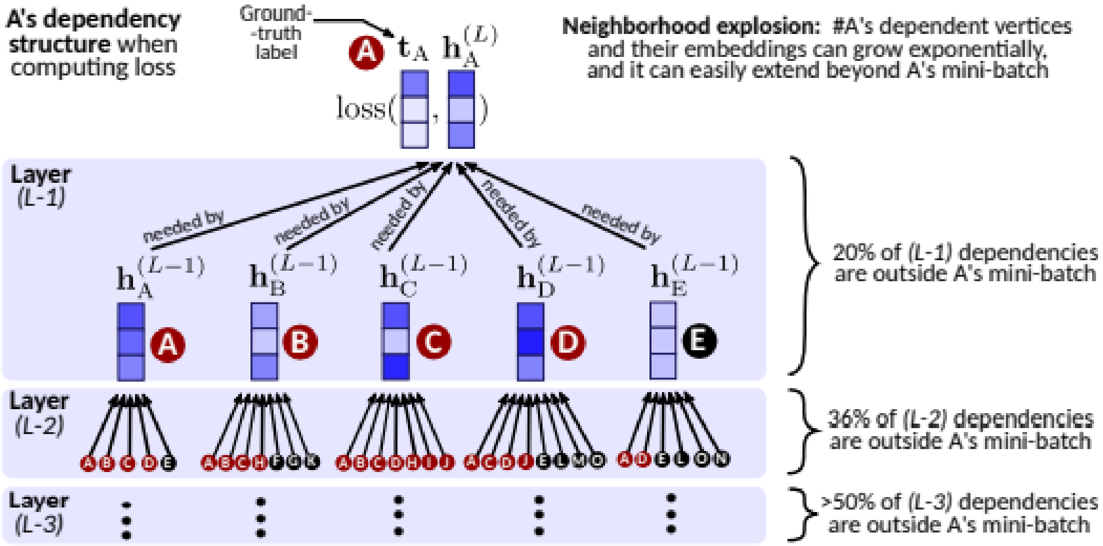


Fig. 8: **(§ 3.1, § 3.2) Graph partition parallelism vs. dependent and independent mini-batch parallelism in GNNs.** Different colors (**red**, **green**, **blue**) indicate different graph partitions or mini-batches, and the associated different workers. Black vertices do not belong to any mini-batch.

[Besta, M., and Hoefler, T. (2022). Parallel and Distributed Graph Neural Networks: An In-Depth Concurrency Analysis.]

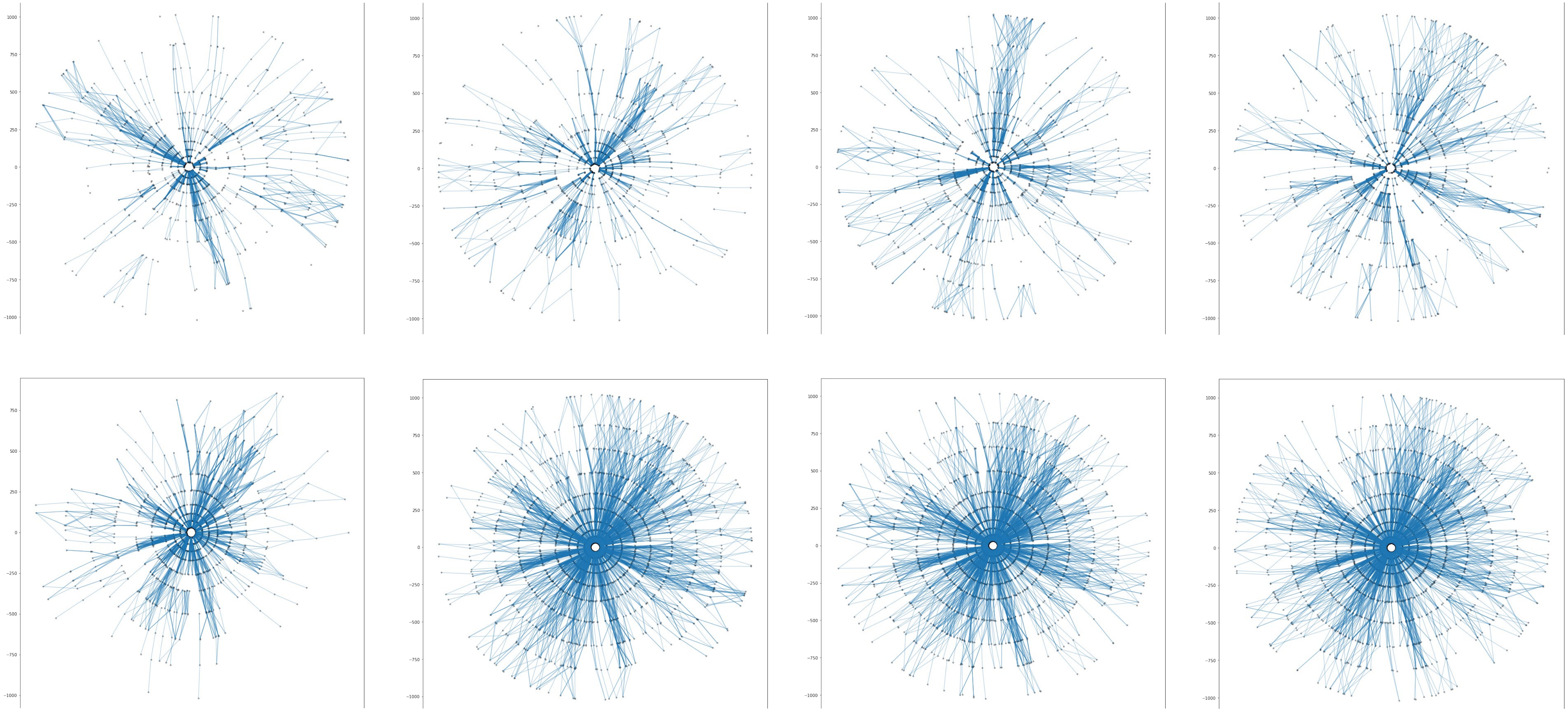
Neighborhood Explosion



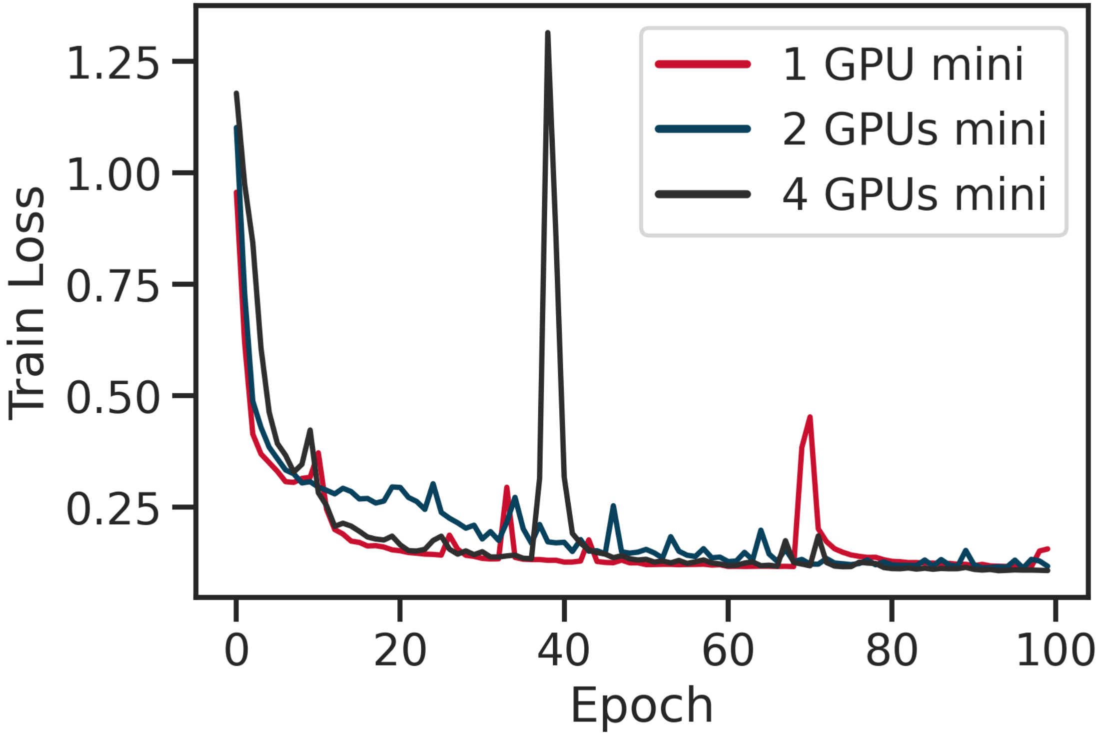


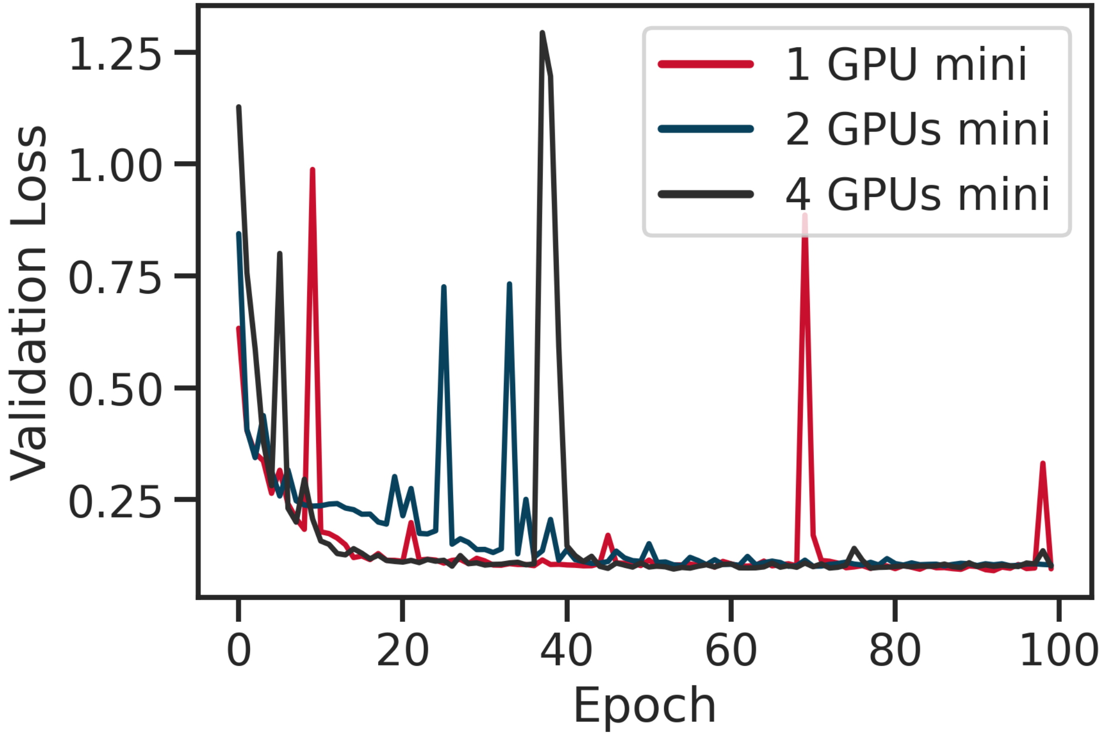
[Besta, M., and Hoefler, T. (2022). Parallel and Distributed Graph Neural Networks: An In-Depth Concurrency Analysis.]()

Graph Partition vs Neighbor Sampling



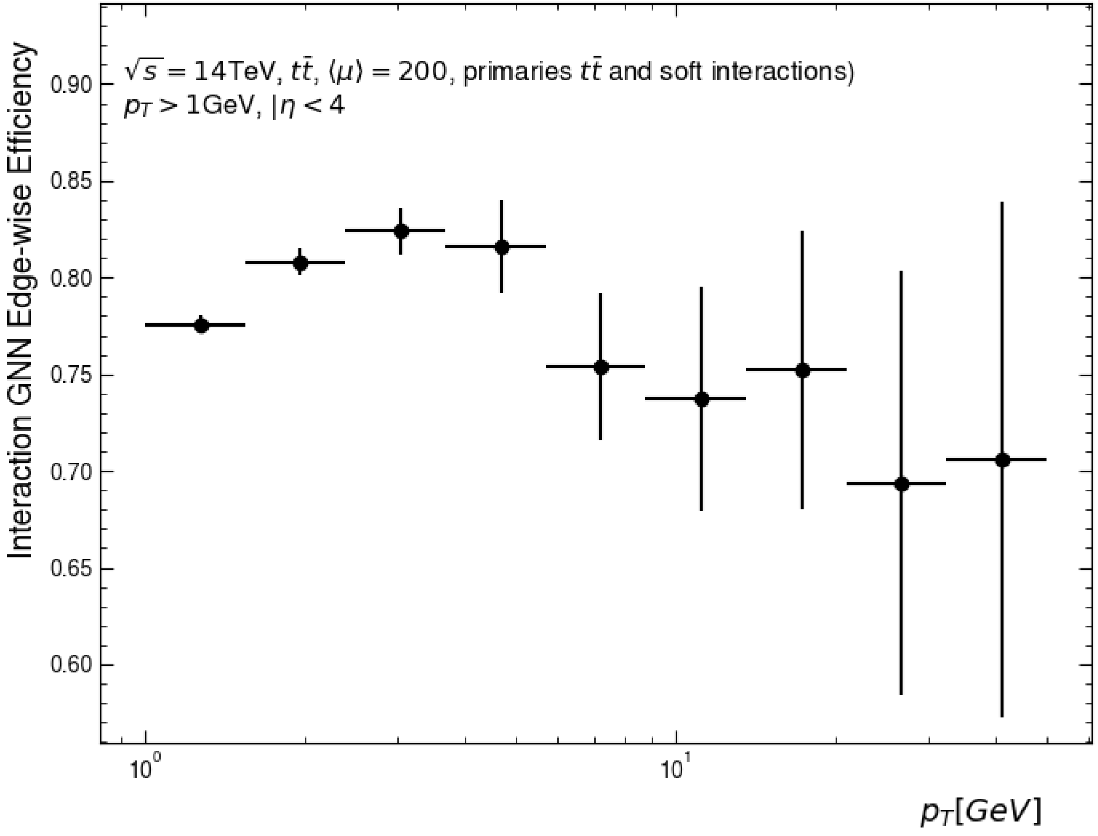
Training and Validation Loss for Mini-batch DDP GNN training





Efficiency - Full vs Mini-batch GNN training

Full-Batch



Mini-Batch