



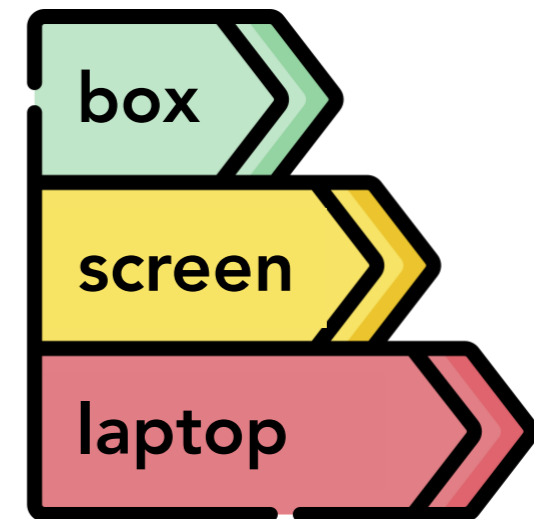
BNS-GCN: Efficient Full-Graph Training of Graph Convolutional Networks with Partition-Parallelism and Random Boundary Node Sampling

Cheng Wan*, **Youjie Li***, Ang Li, Nam Sung Kim, Yingyan Lin

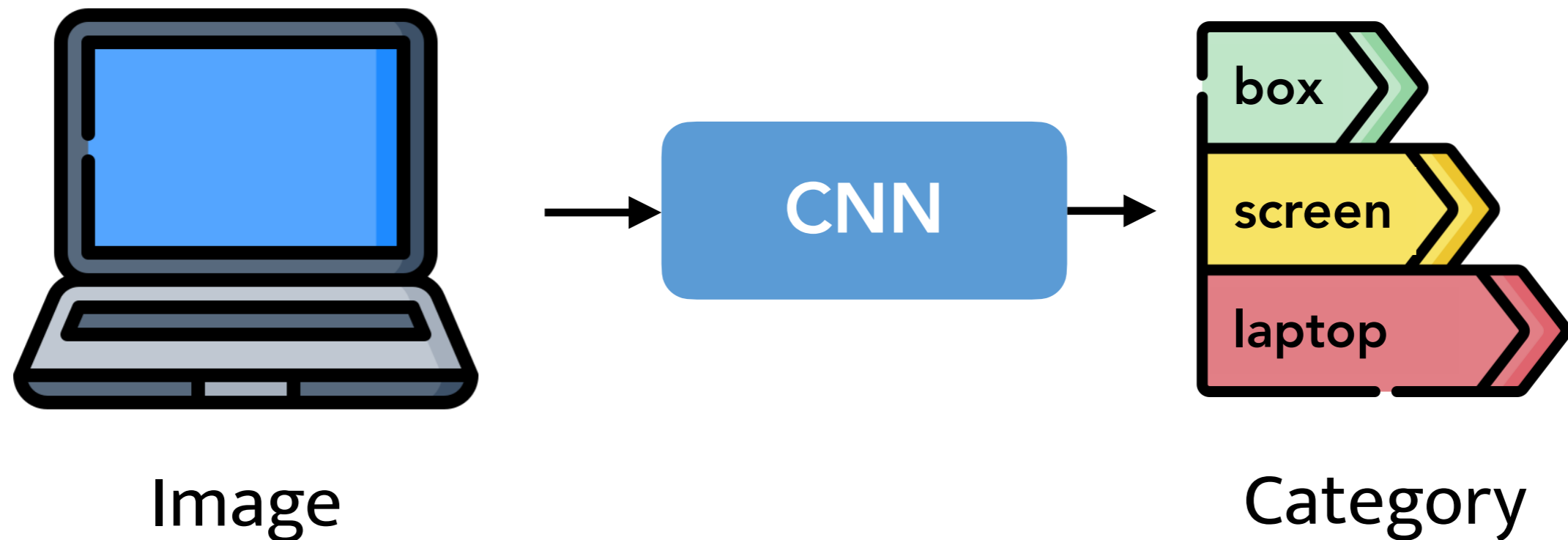
MLSys 2022



Image

**Classify**

Category



Convolutional Neural Networks (CNNs)
are powerful in image classification

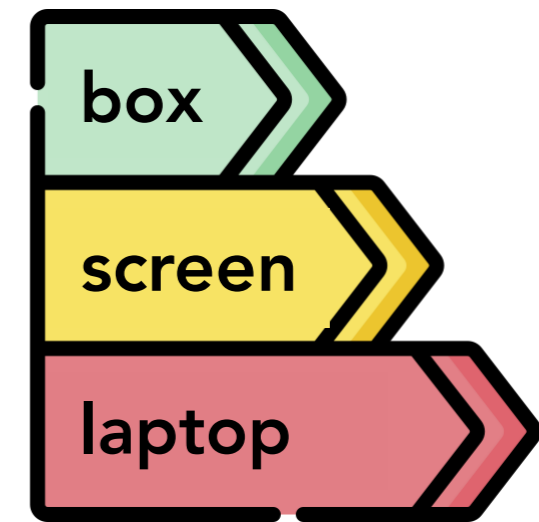
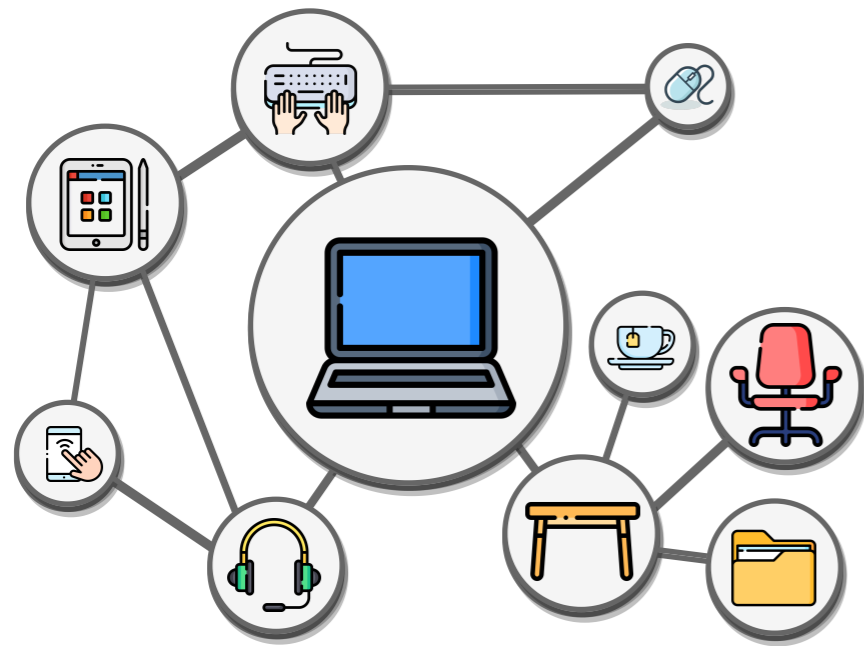


Image + **Relationship**
(e.g., co-purchase)

Category

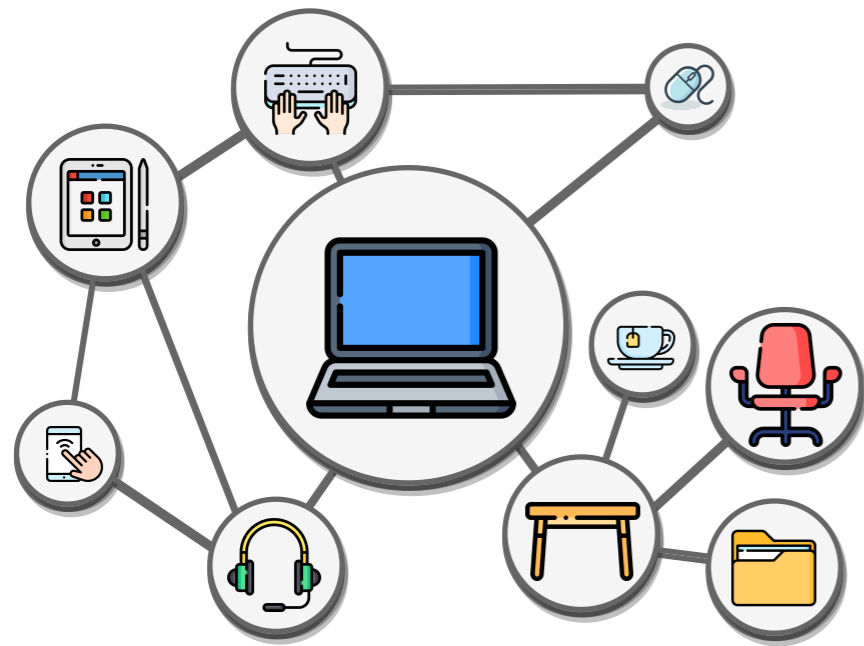
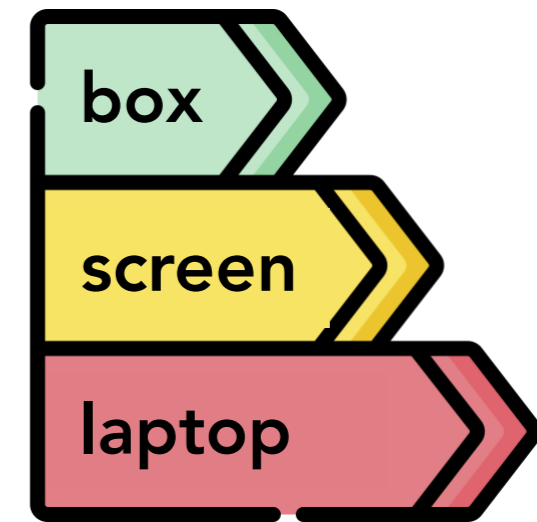


Image + **Relationship**
(e.g., co-purchase)



Category

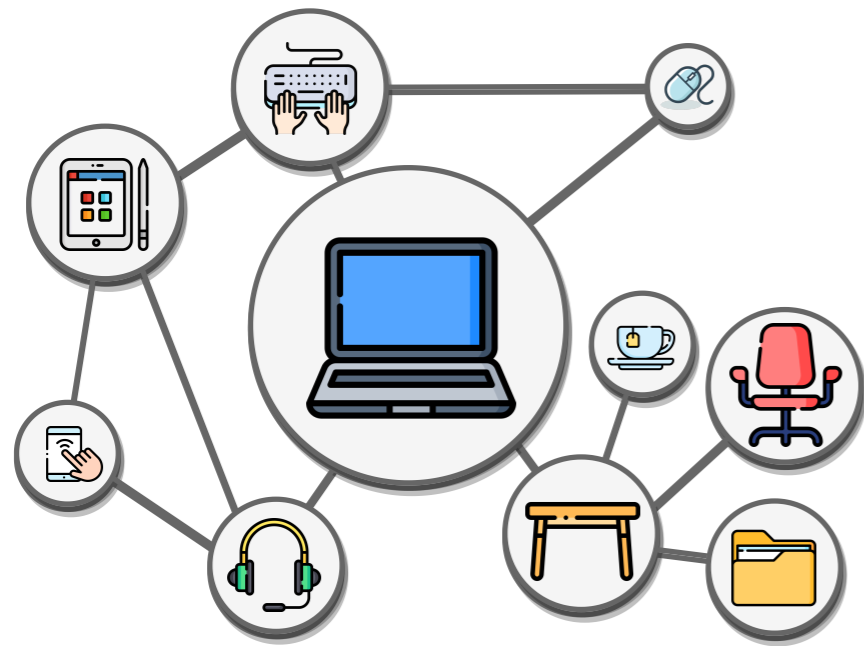
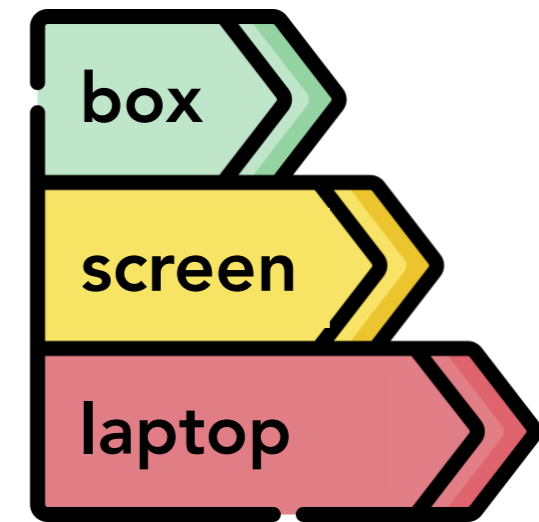


Image + **Relationship**
(e.g., co-purchase)



Category

How to incorporate **relationship**
among data samples?

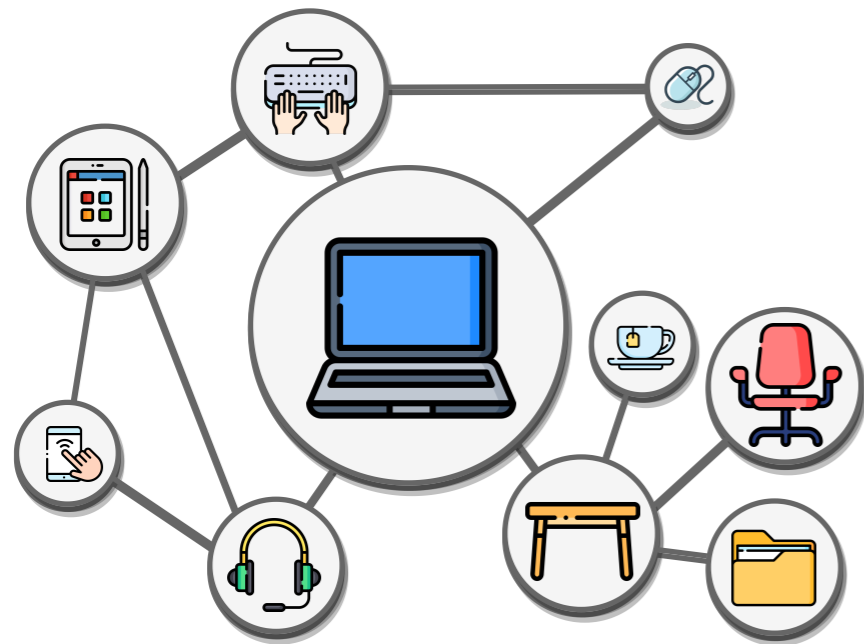
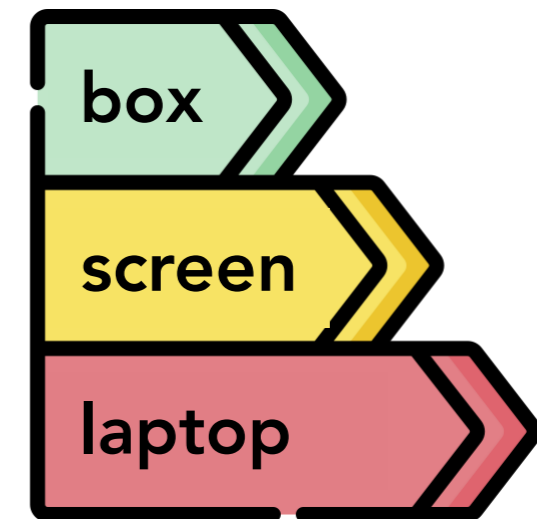


Image + **Relationship**
(e.g., co-purchase)

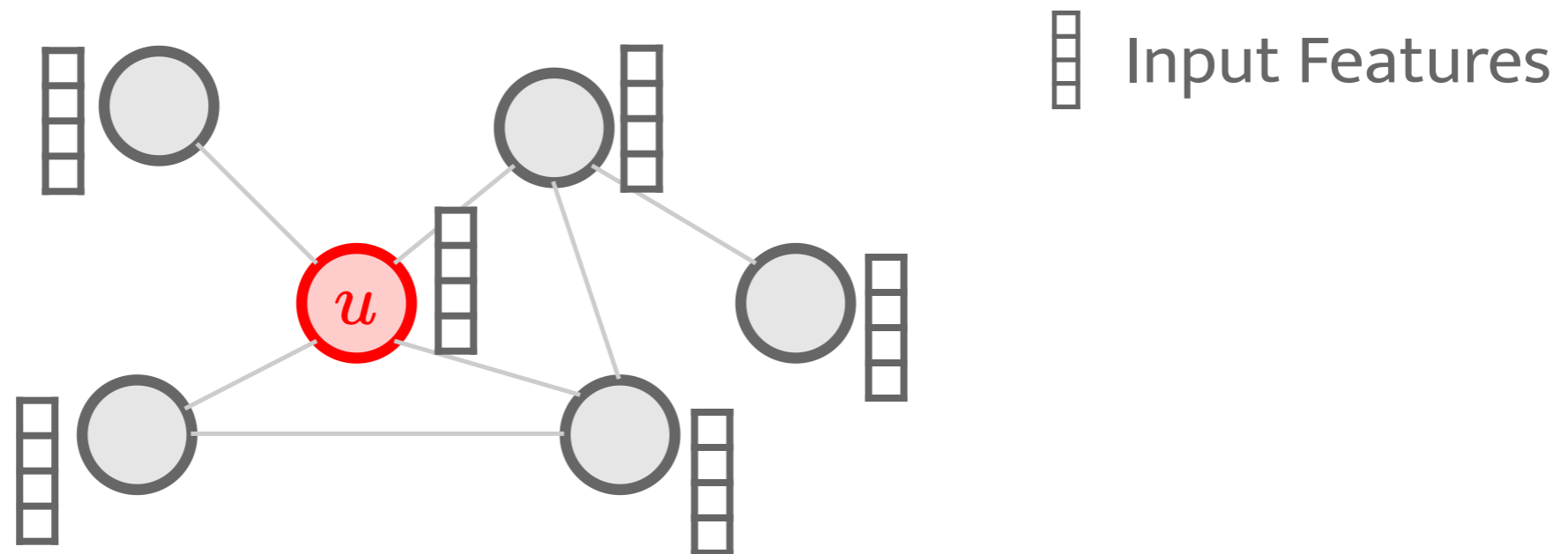


Category

Graph Convolutional Network (GCN)

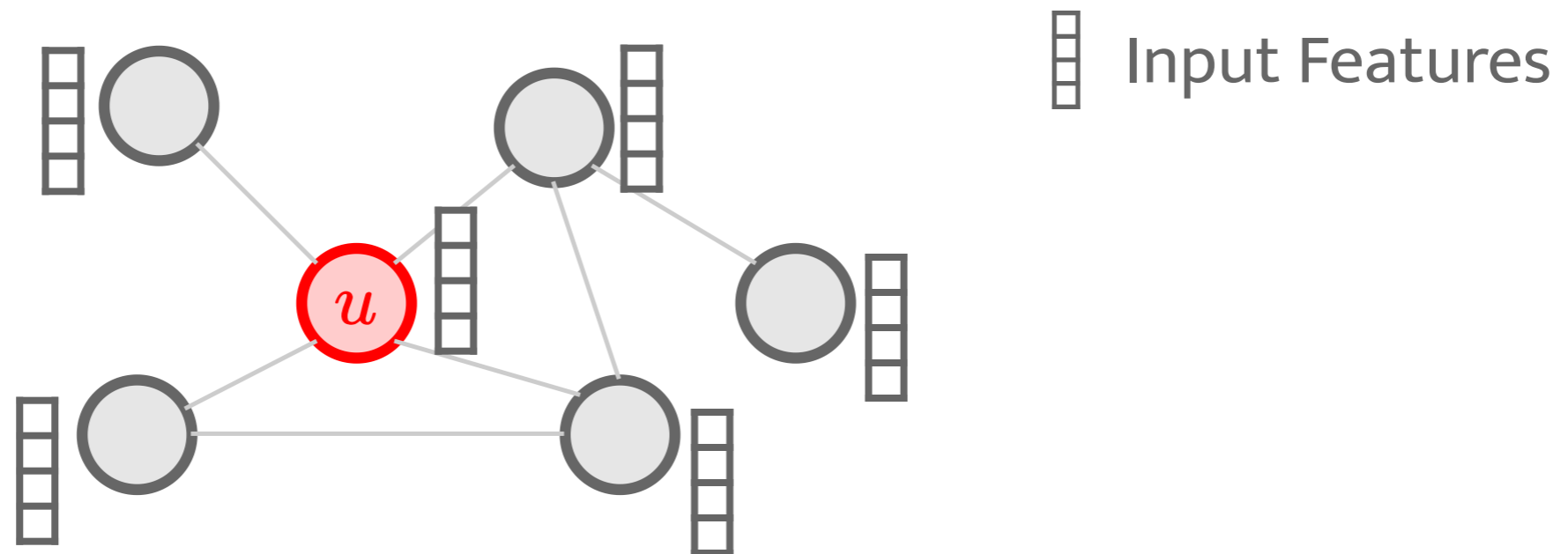
the SOTA model for capturing **relationship**

Graph Convolutional Network (GCN)



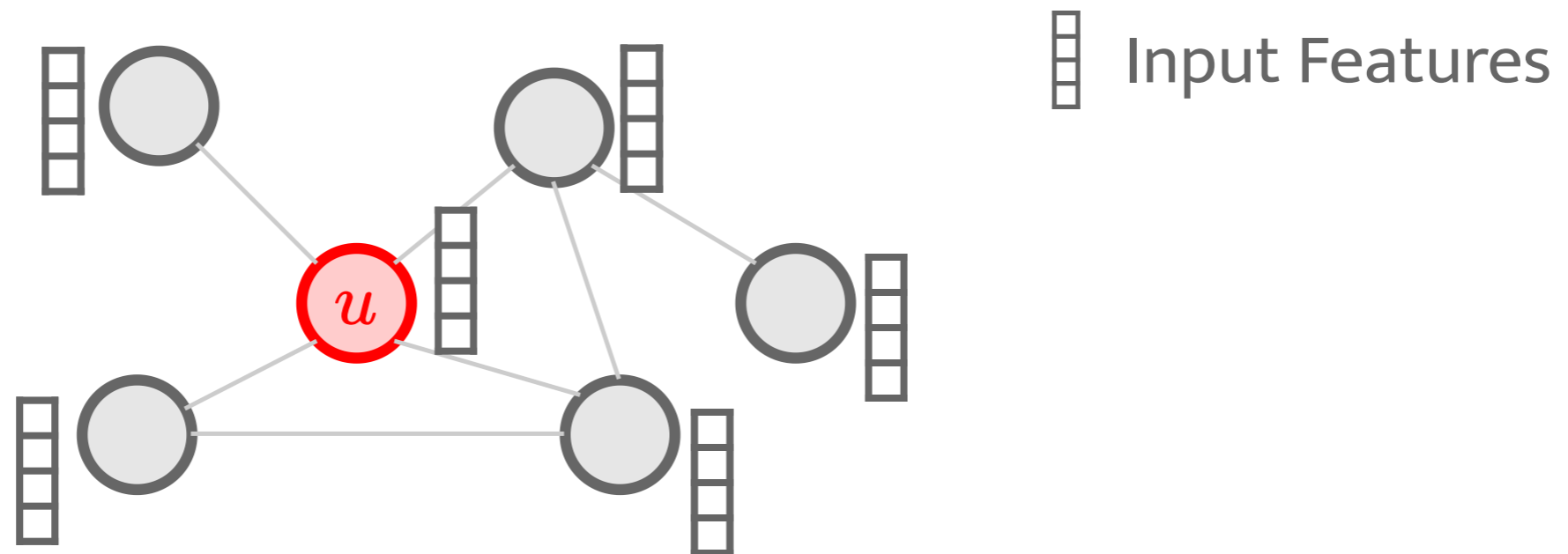
Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?



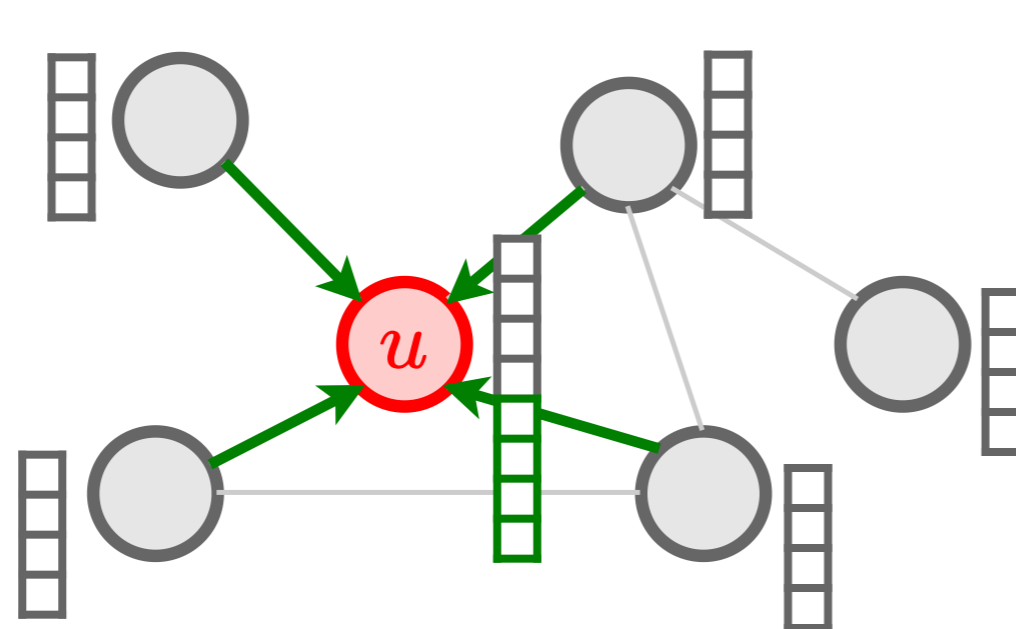
Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?



Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?

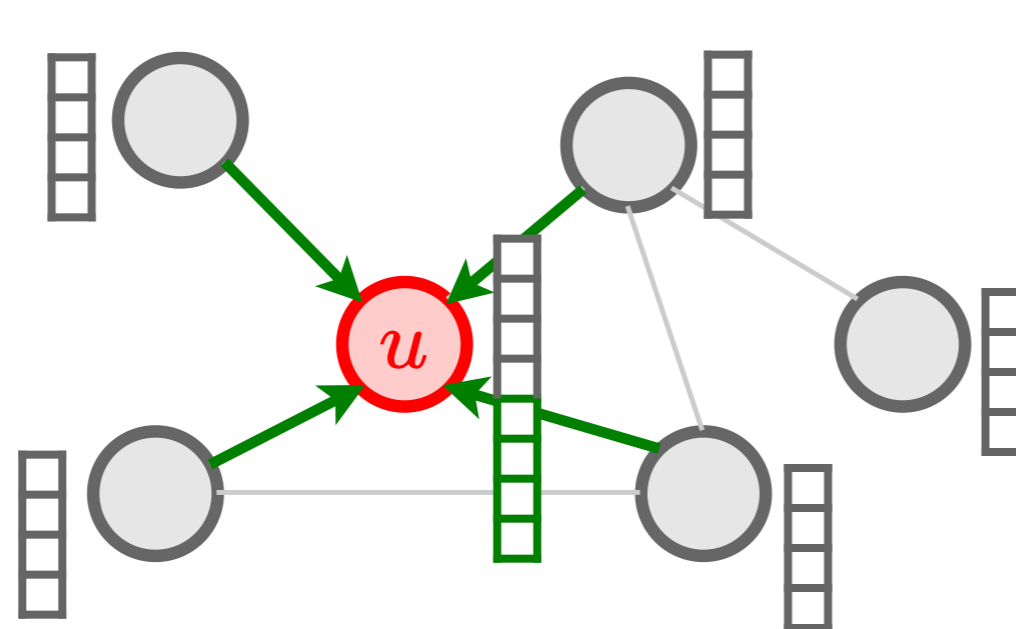


 Input Features

① **Neighbor Aggregation**
(e.g., average pooling)

Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?

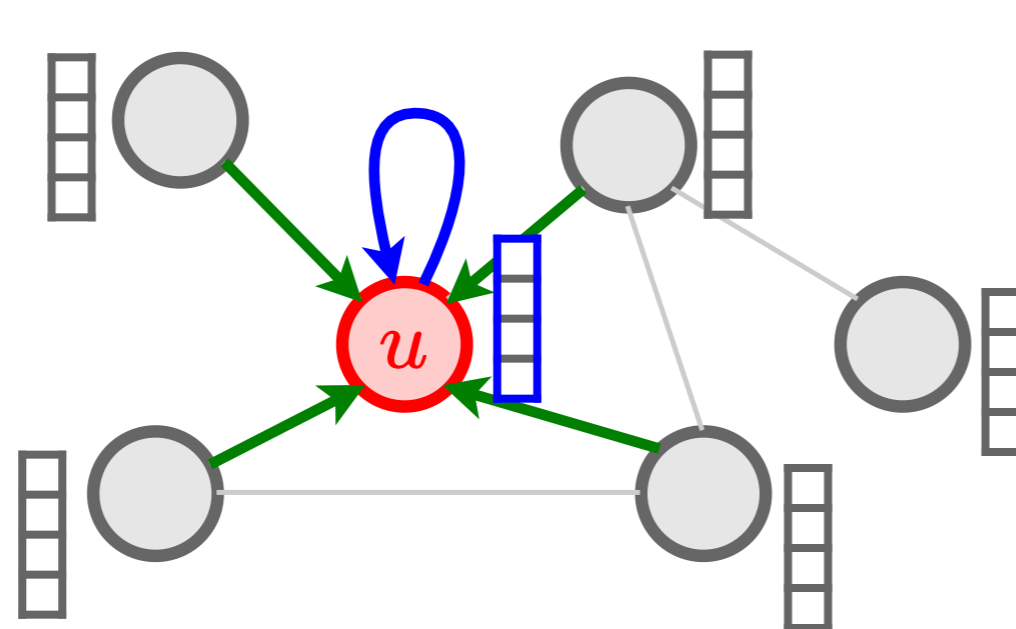


 Input Features

① Neighbor Aggregation
(e.g., average pooling)

Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?



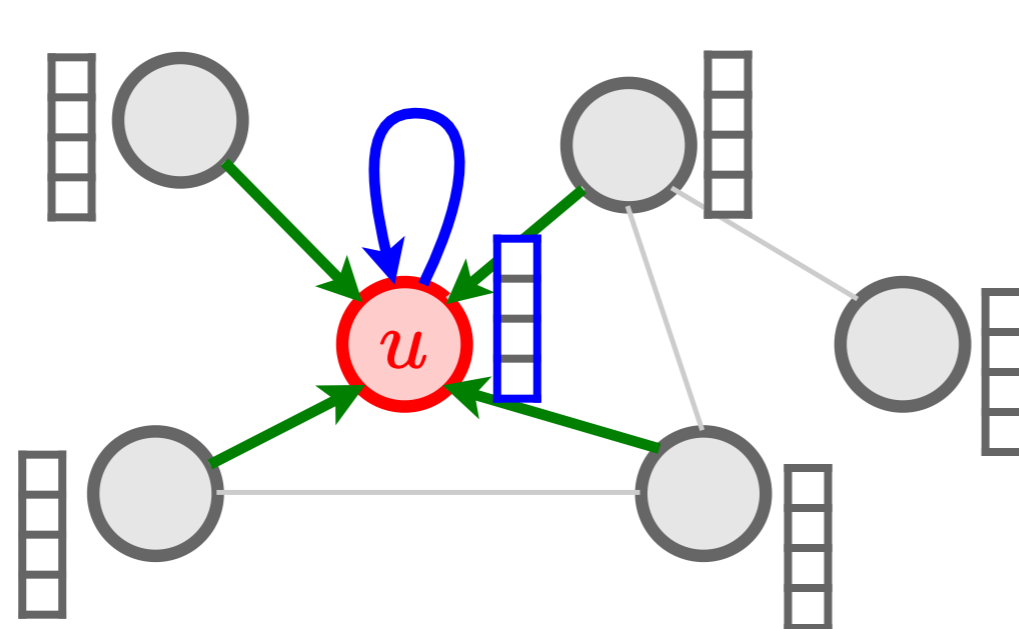
 Input Features

① Neighbor Aggregation
(e.g., average pooling)

② Feature Update
(e.g., MLP)

Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?



 Input Features

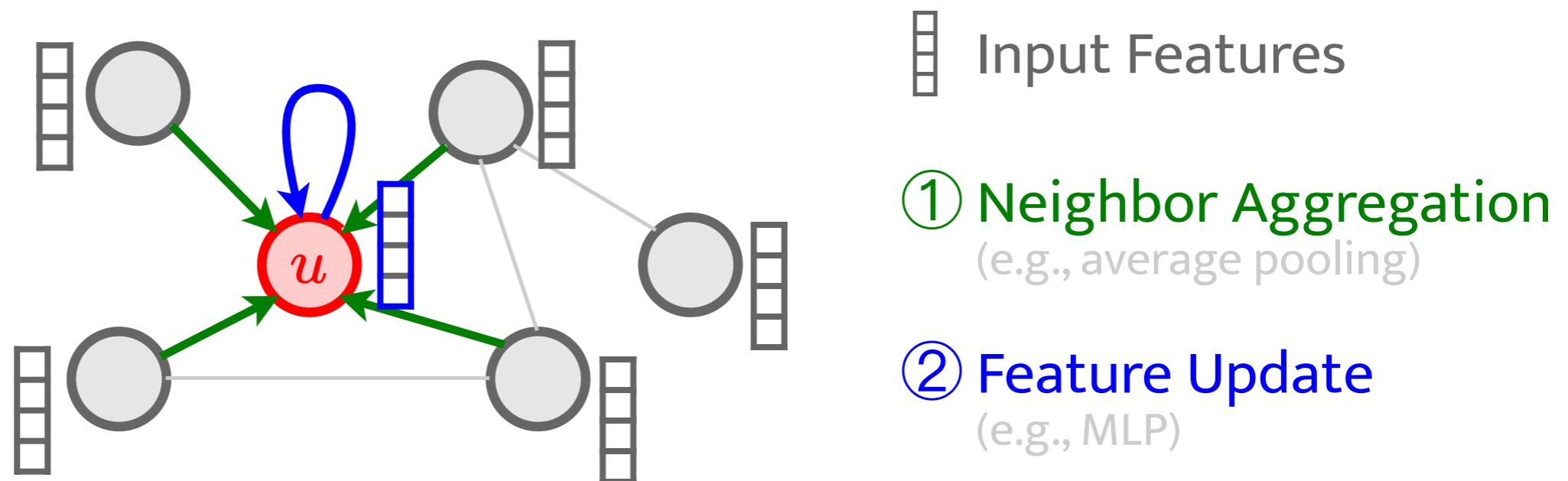
① Neighbor Aggregation
(e.g., average pooling)


② Feature Update
(e.g., MLP)

Output embedding  can be fed to the next GCN layer

Graph Convolutional Network (GCN)

How to compute the embedding of **node u** ?



Output embedding  can be fed to the next GCN layer
or be used to downstream tasks
(e.g., node classification)

Challenge: **Giant** Graphs for GCNs



Amazon Co-Purchase Dataset [1,2]

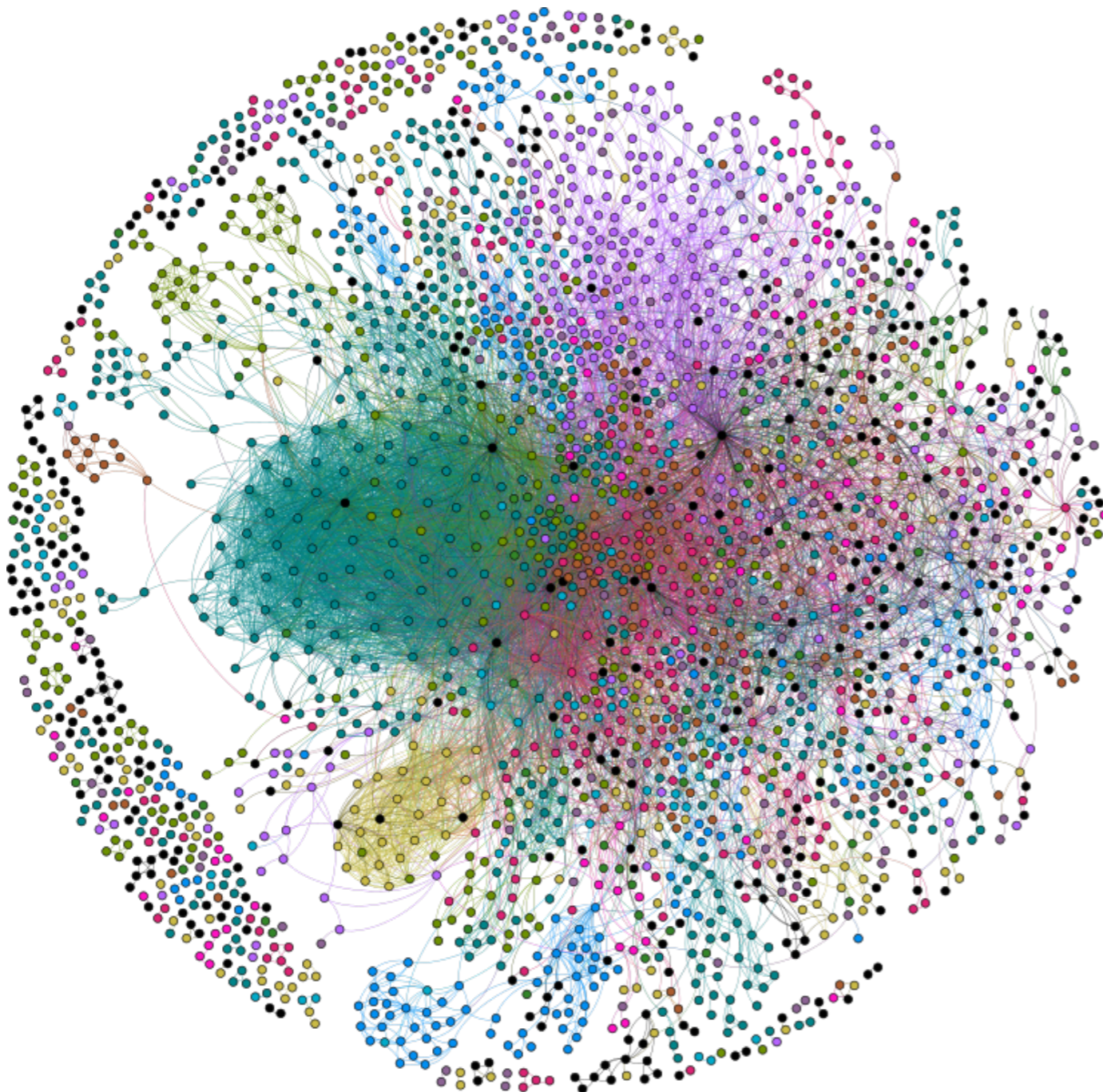
9.4M nodes

231M edges

[1] Backman, Evans and Girke. Large-scale Bioactivity Analysis of the Small-molecule Assayed Proteome. *PLoS ONE*'17.

[2] McAuley et al. Image-based Recommendations on Styles and Substitutes. *SIGIR*'15.

Challenge: **Giant** Graphs for GCNs



Amazon Co-Purchase Dataset [1,2]

9.4M nodes

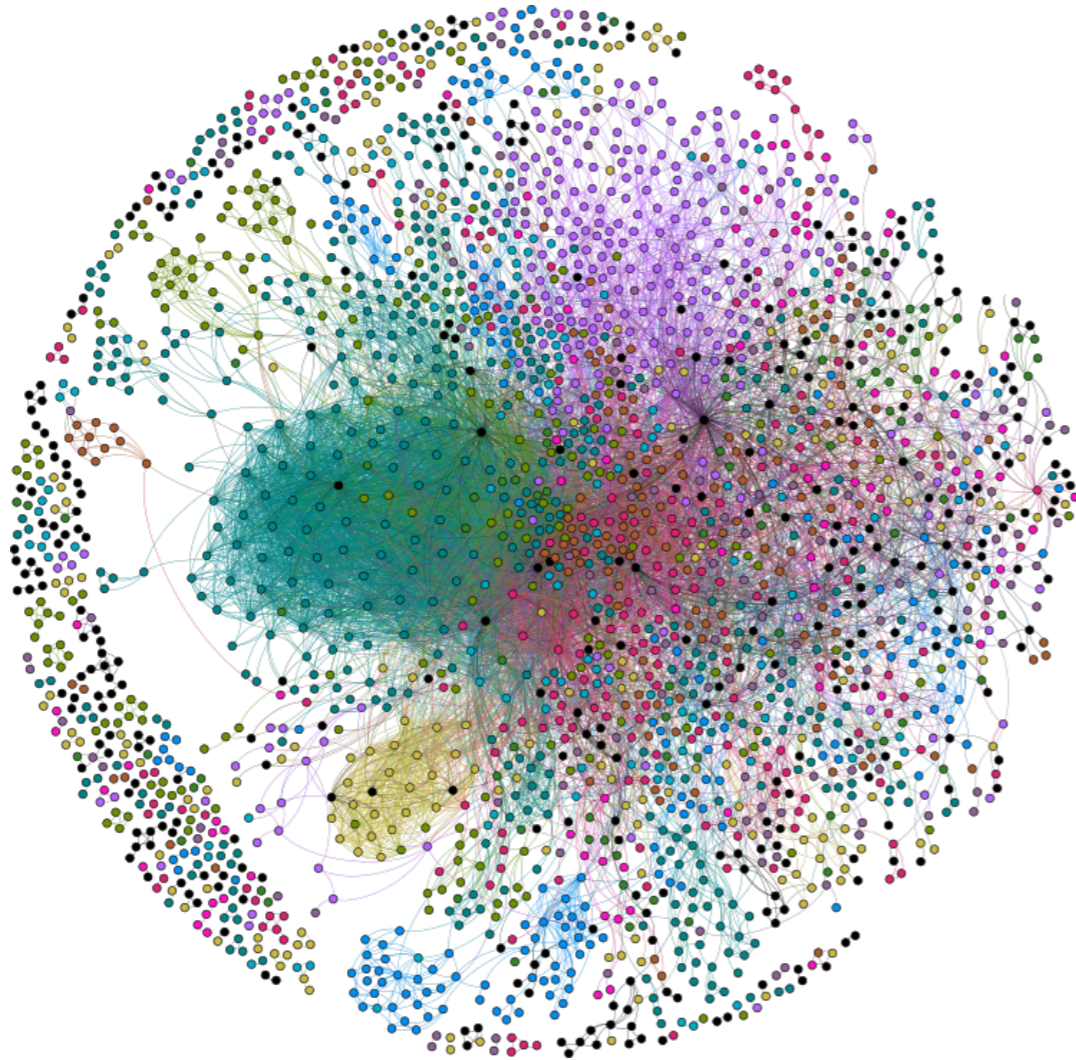
231M edges

>100GB memory for a 3-layer GCN

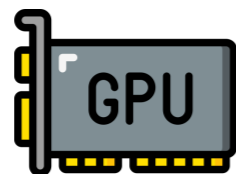
[1] Backman, Evans and Girke. Large-scale Bioactivity Analysis of the Small-molecule Assayed Proteome. *PLoS ONE*'17.

[2] McAuley et al. Image-based Recommendations on Styles and Substitutes. *SIGIR*'15.

Challenge: **Giant** Graphs for GCNs

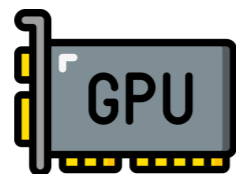
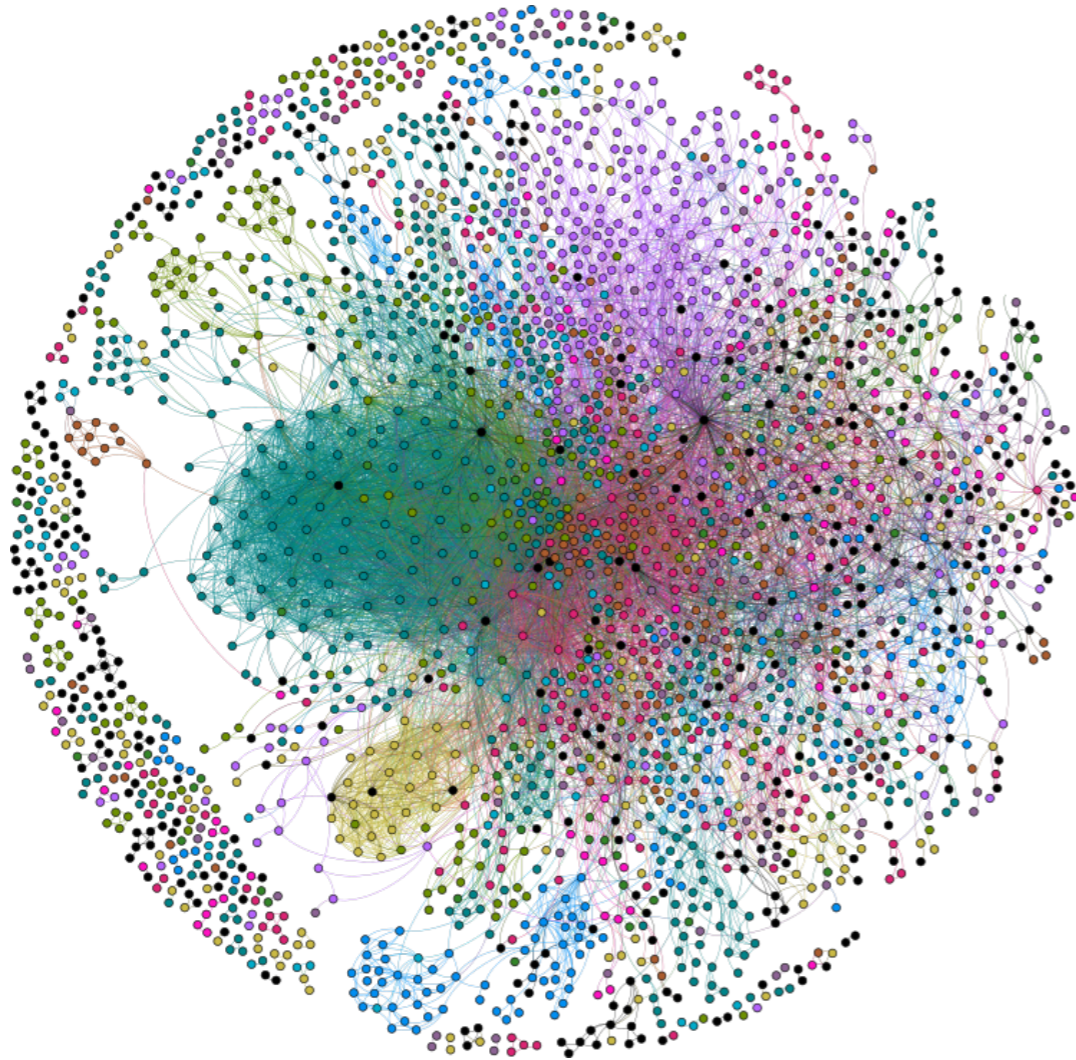


>100GB training



16GB V100

Challenge: **Giant** Graphs for GCNs



>100GB training

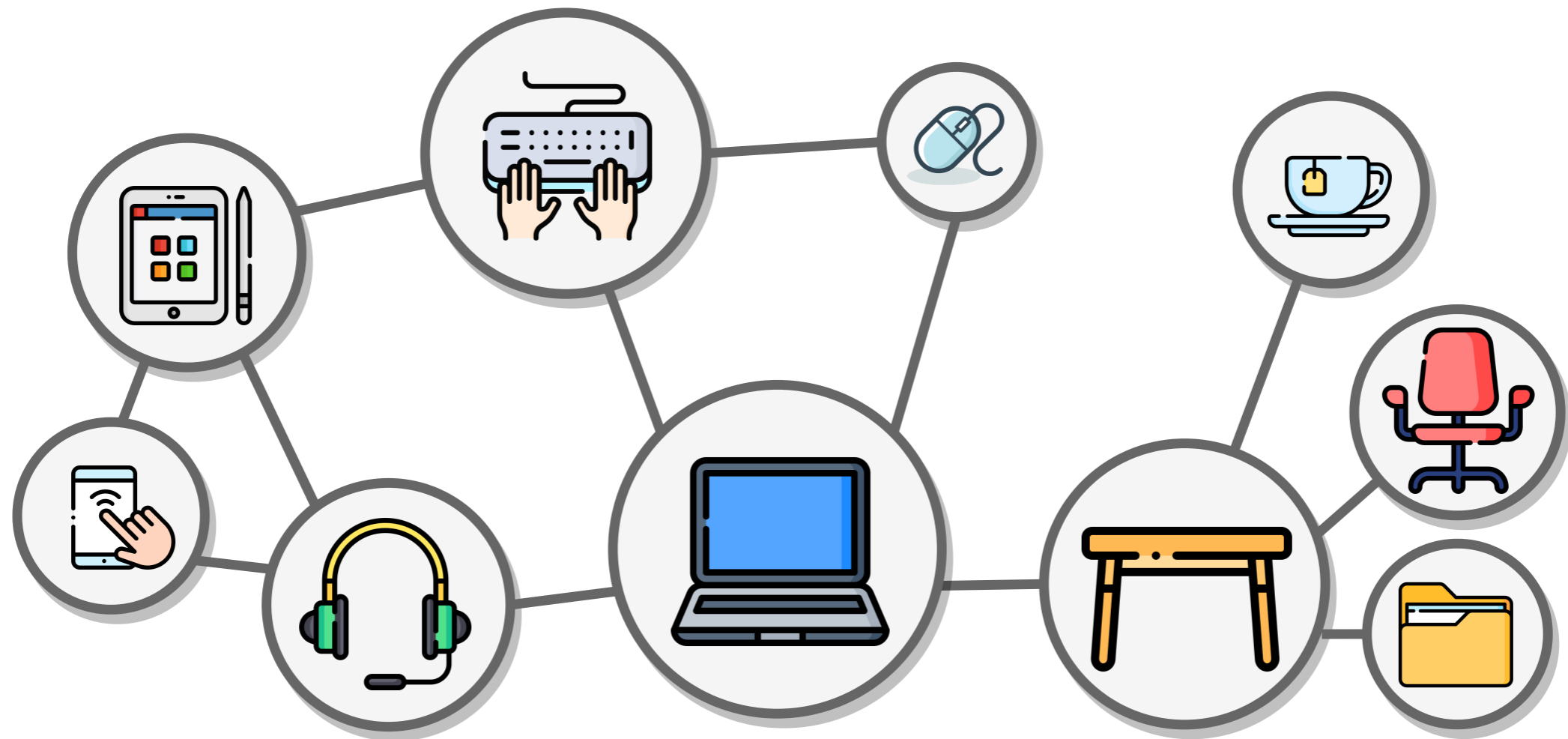
NOT fit

16GB V100

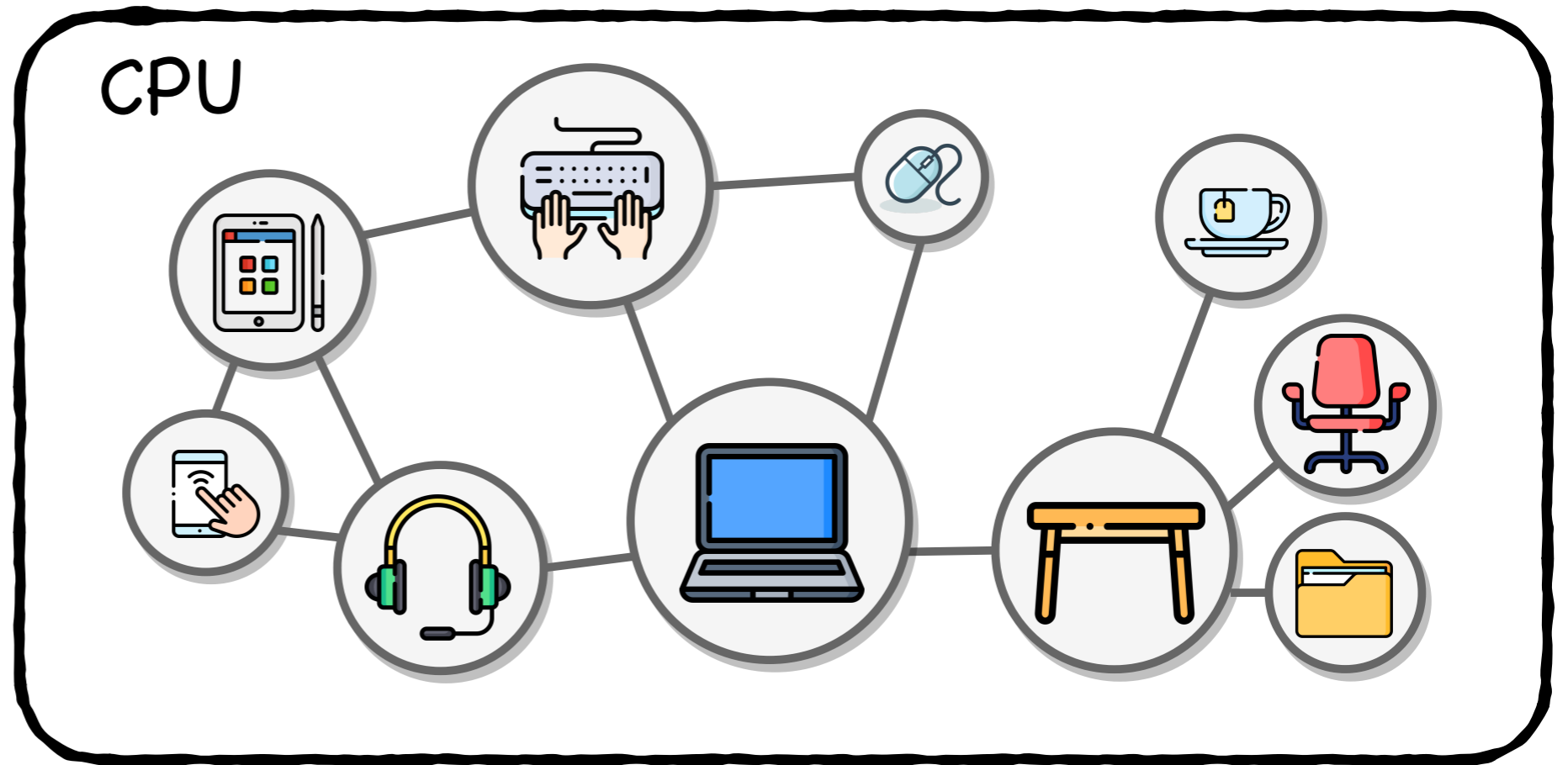
How to train a GCN at scale? **Efficiently?**



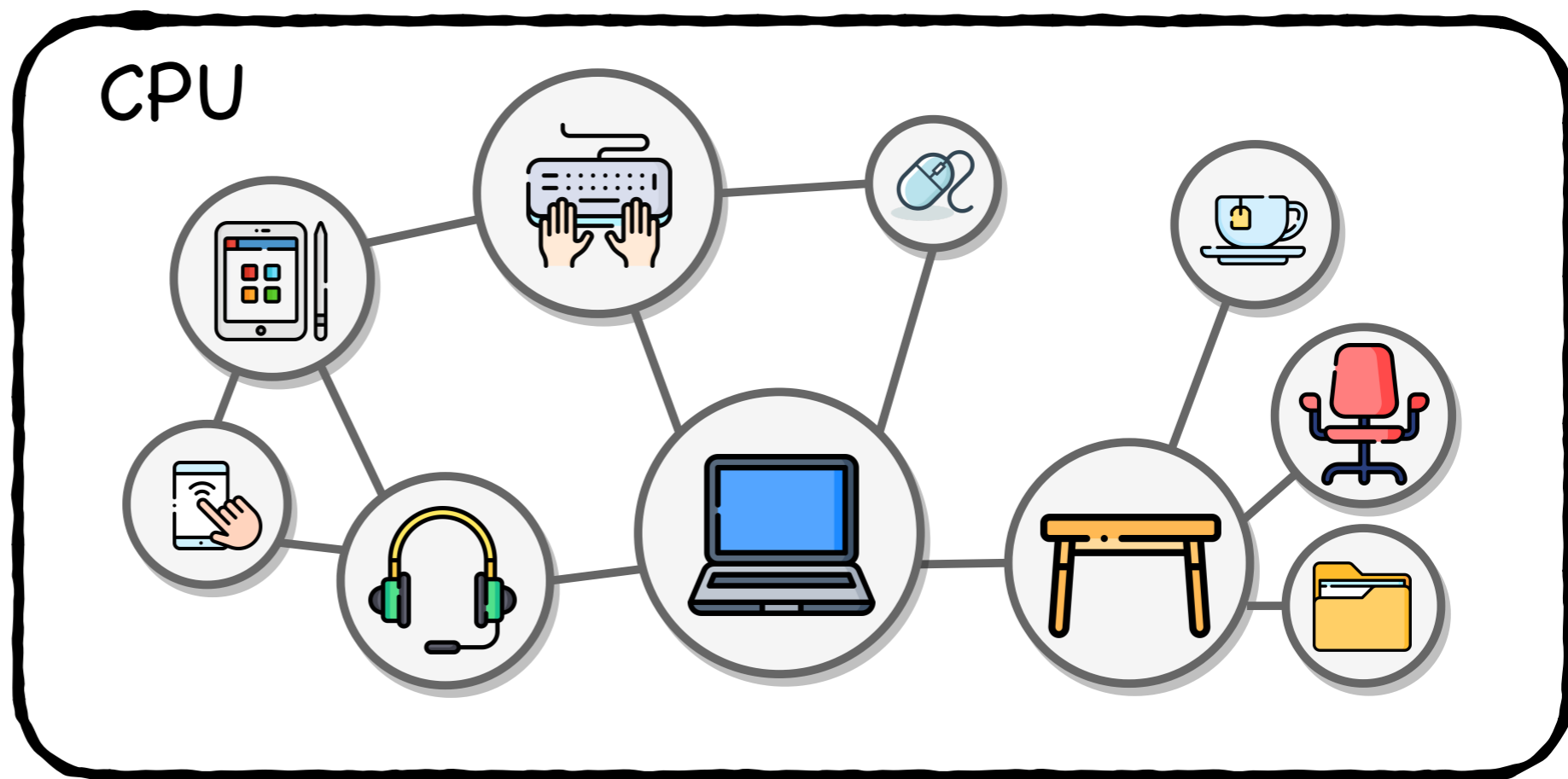
Category I



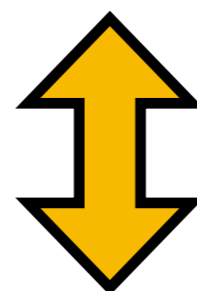
Storage in CPU



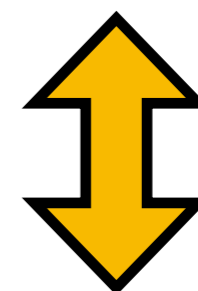
Storage in CPU



Swap

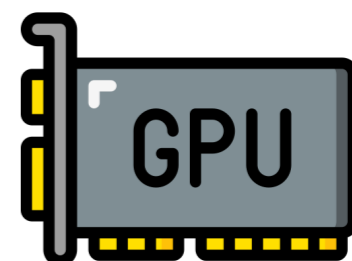
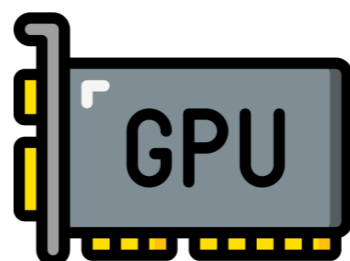
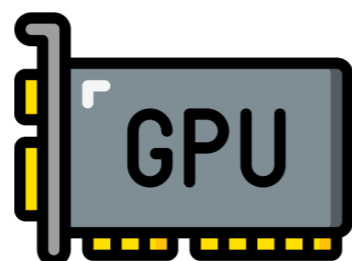


Swap

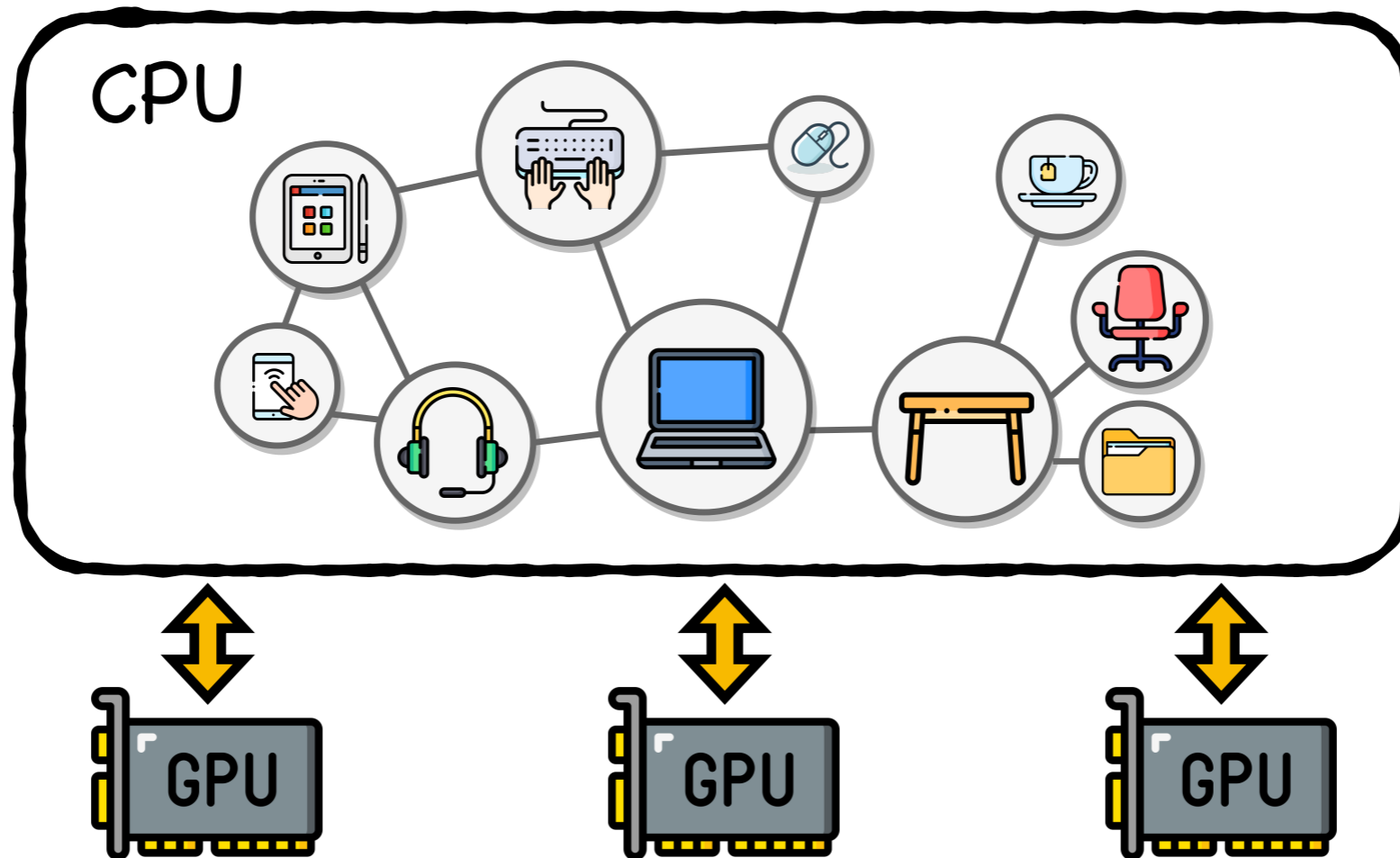


Swap

Training in GPU



Category I: Swap-Based Methods

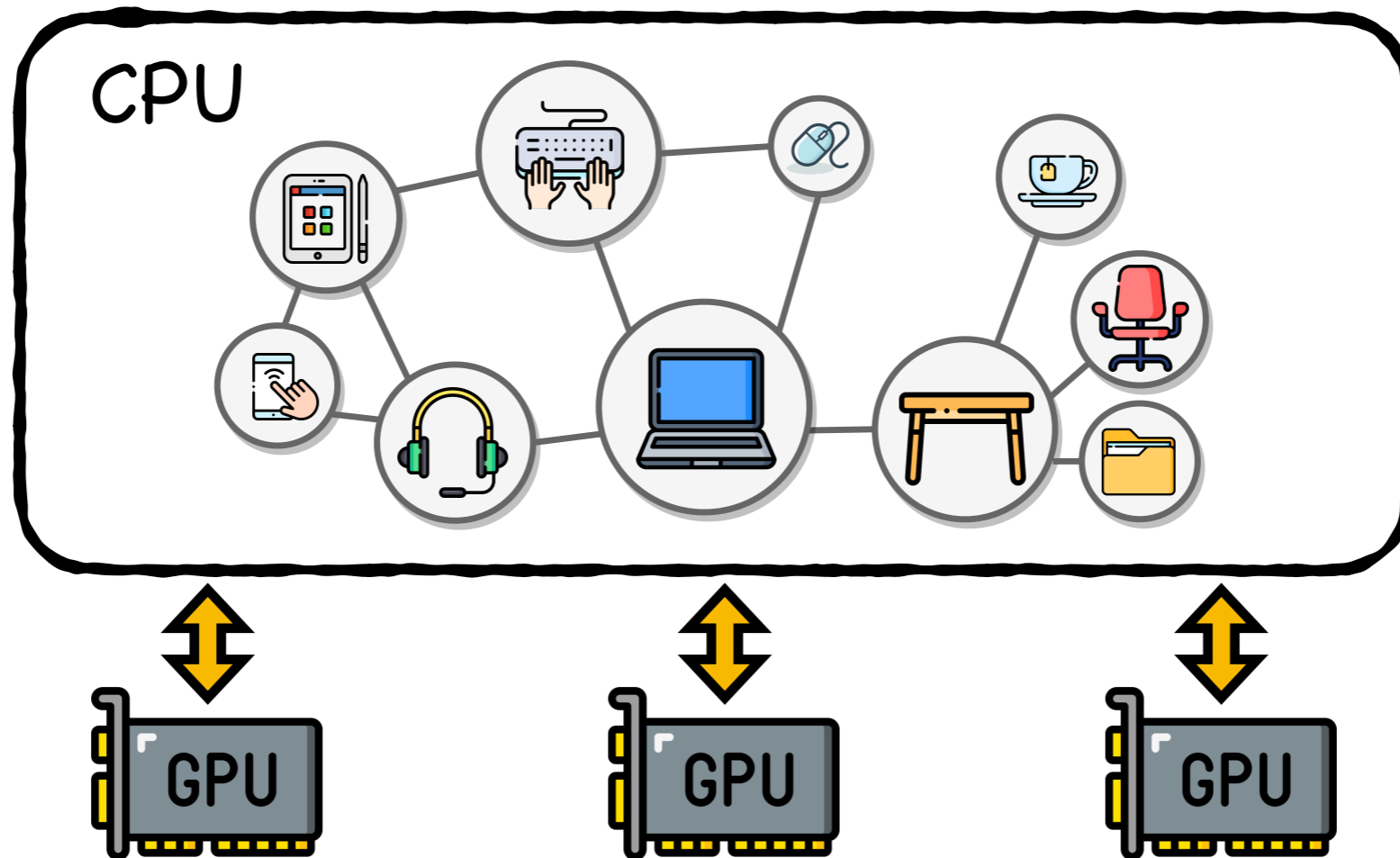


[1] Ma et al. NeuGraph: Parallel Deep Neural Network Computation on Large Graphs. *USENIX ATC'19*

[2] Jia et al. Improving the Accuracy, Scalability, and Performance of Graph Neural Networks with Roc. *MLSys'20*

[3] Fey et al. GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings. *ICML'21*

Category I: Swap-Based Methods



Pro: Scalability of Graph

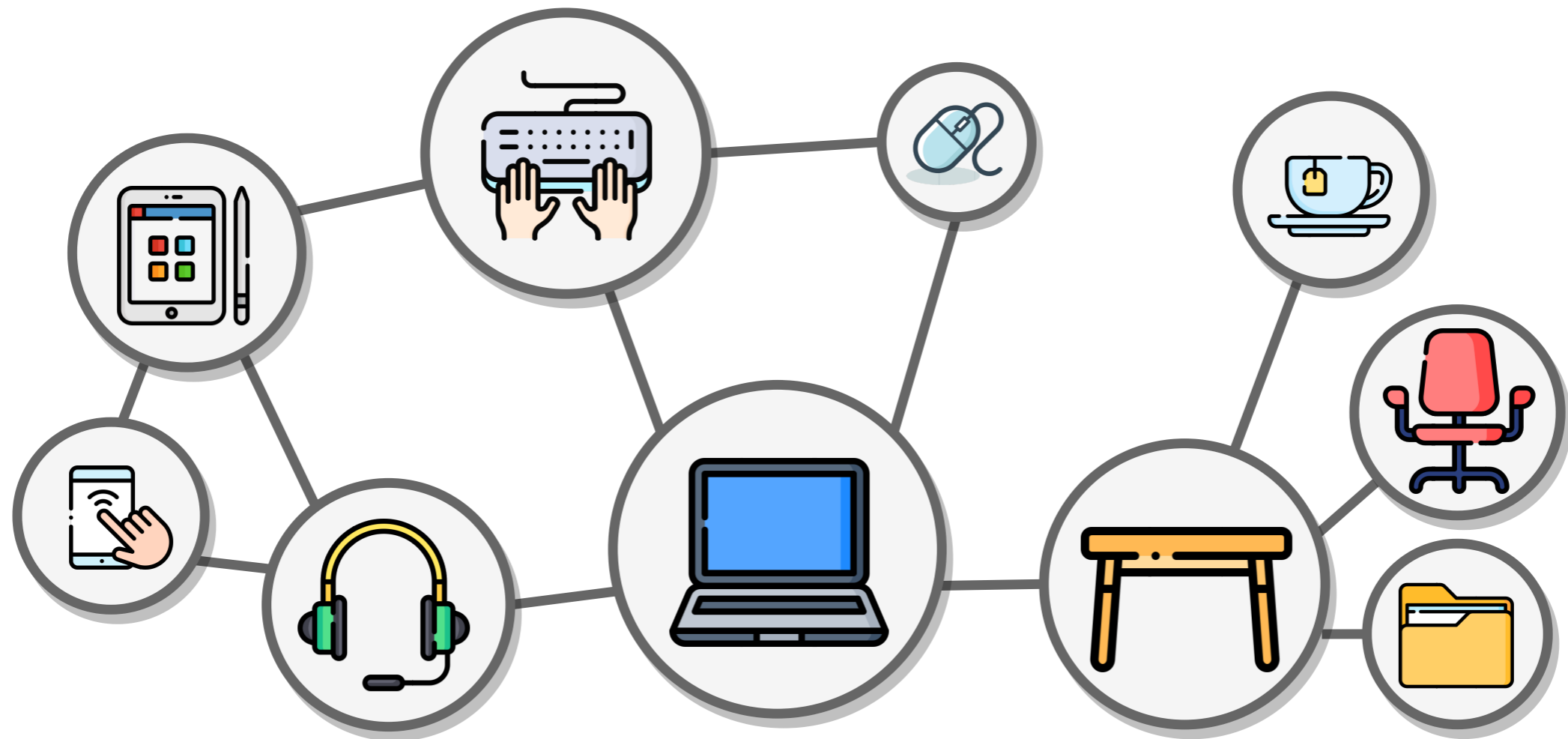
Con: Expensive CPU-GPU Swap

[1] Ma et al. NeuGraph: Parallel Deep Neural Network Computation on Large Graphs. *USENIX ATC'19*

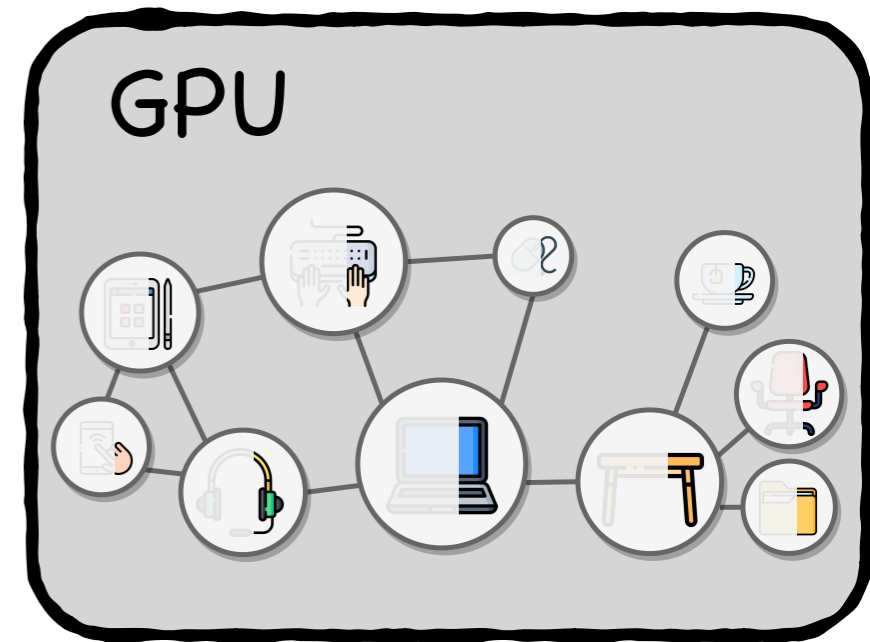
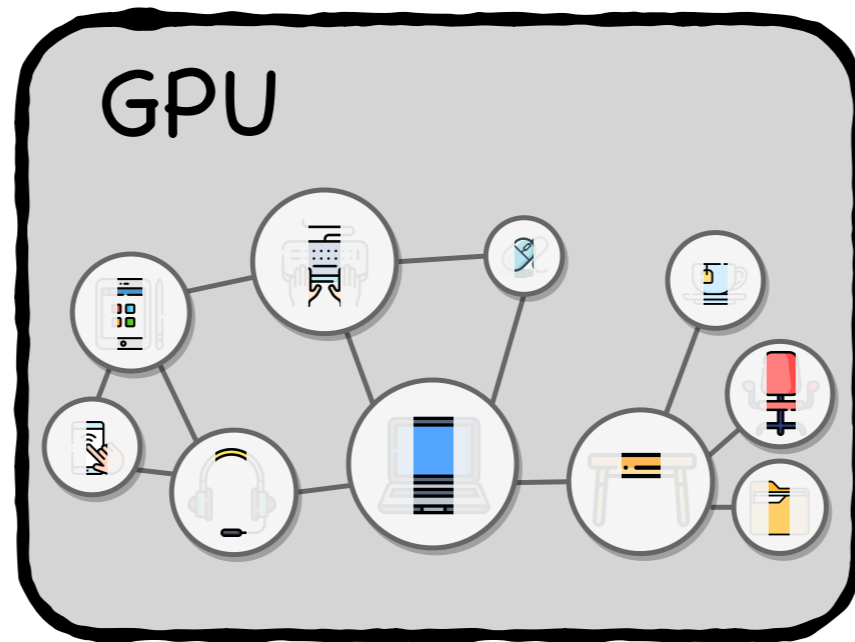
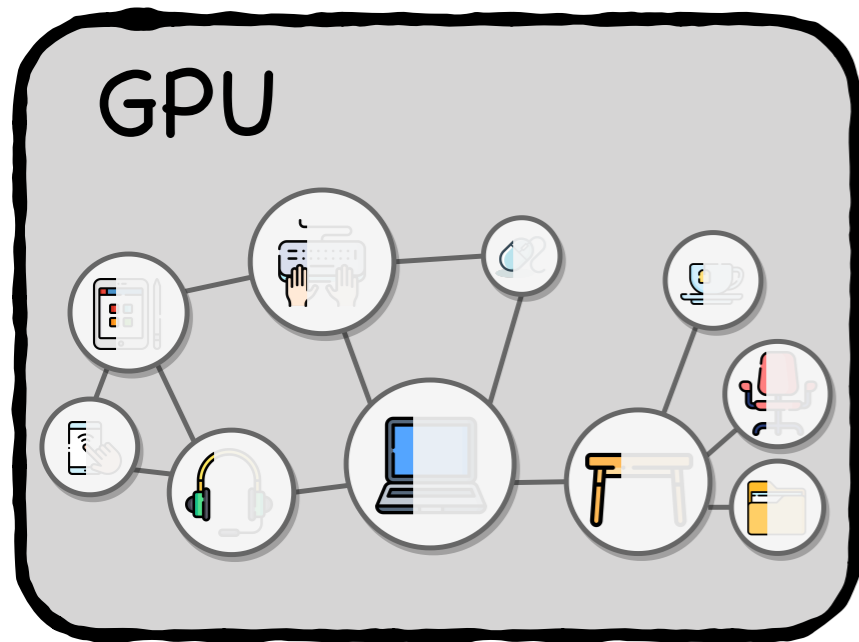
[2] Jia et al. Improving the Accuracy, Scalability, and Performance of Graph Neural Networks with Roc. *MLSys'20*

[3] Fey et al. GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings. *ICML'21*

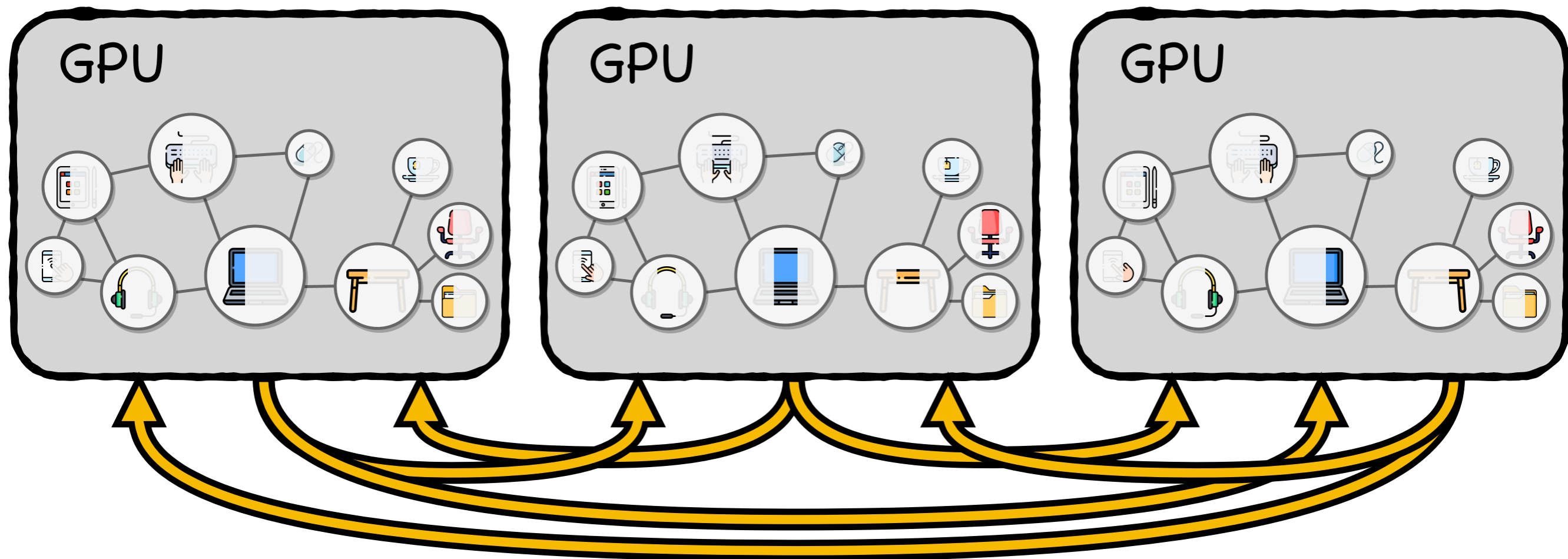
Category II



Slicing features across GPUs

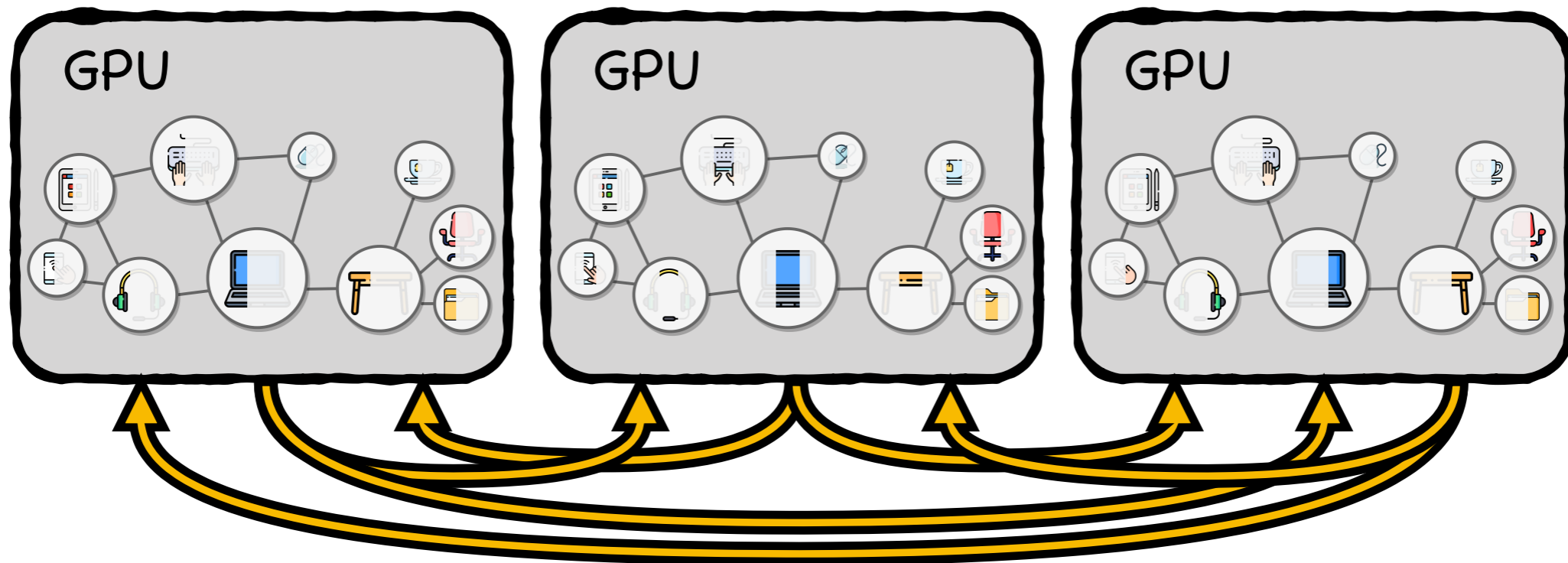


Slicing features across GPUs



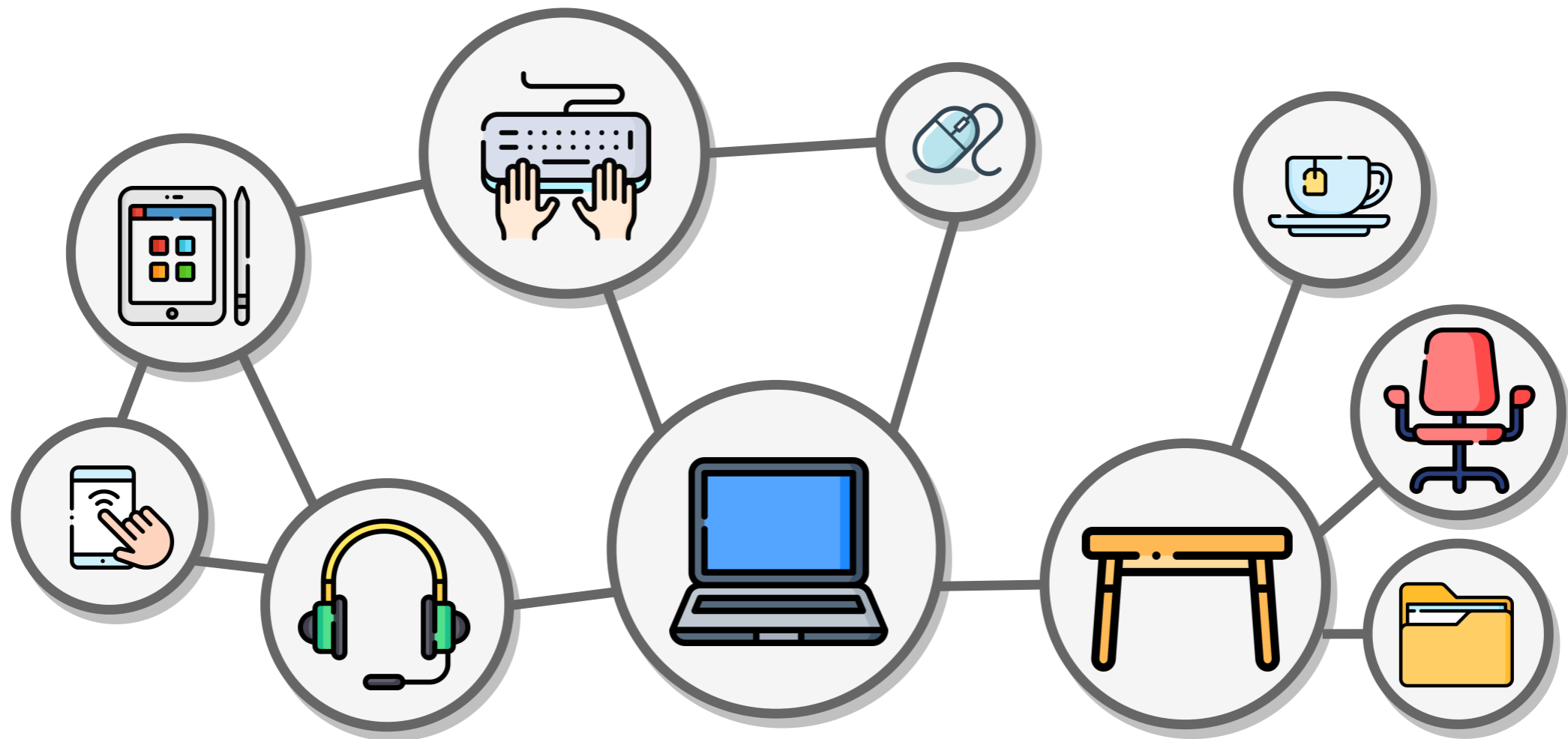
Broadcast

Category II: Slice-Based Methods

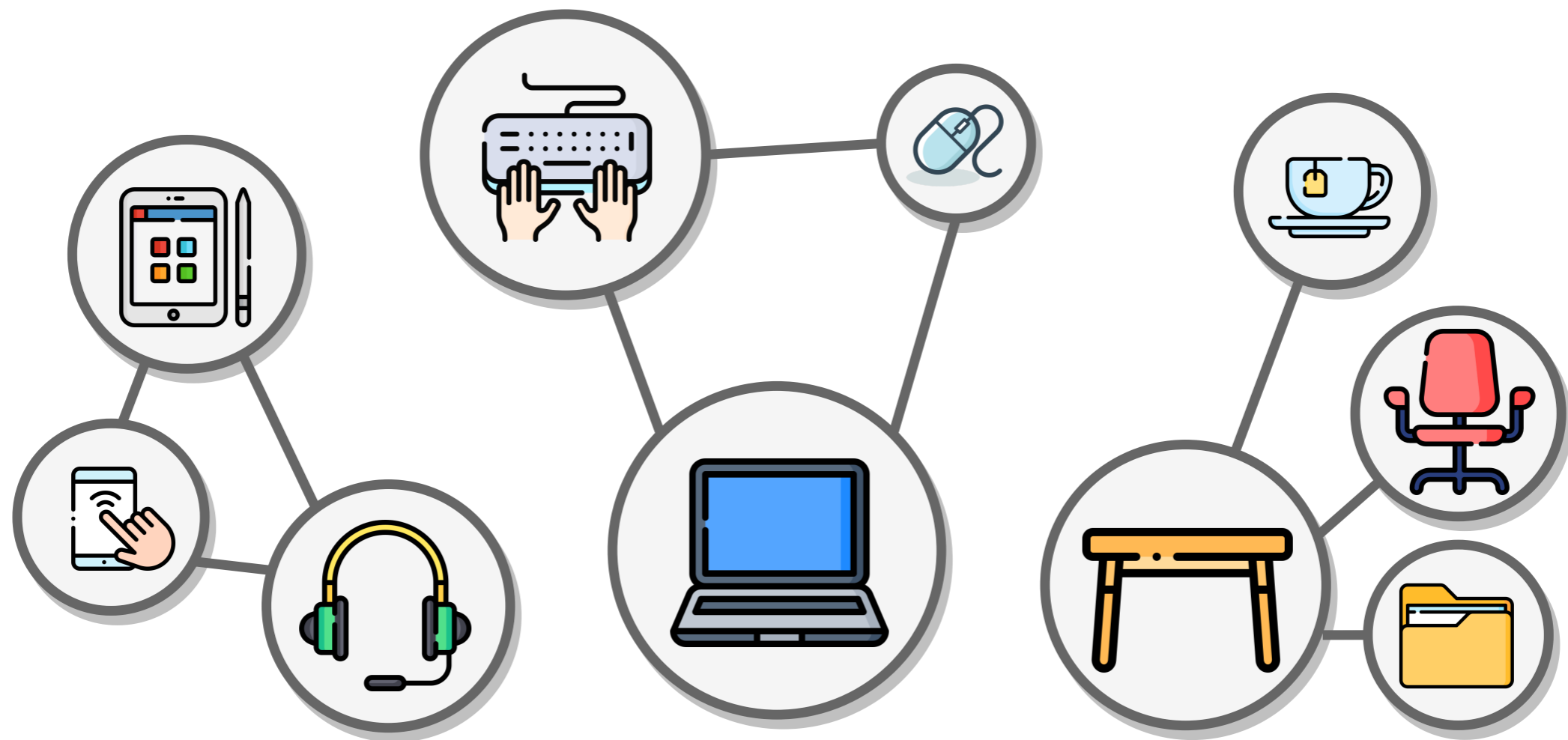


Pro: Balanced Workload
Con: Expensive Broadcast

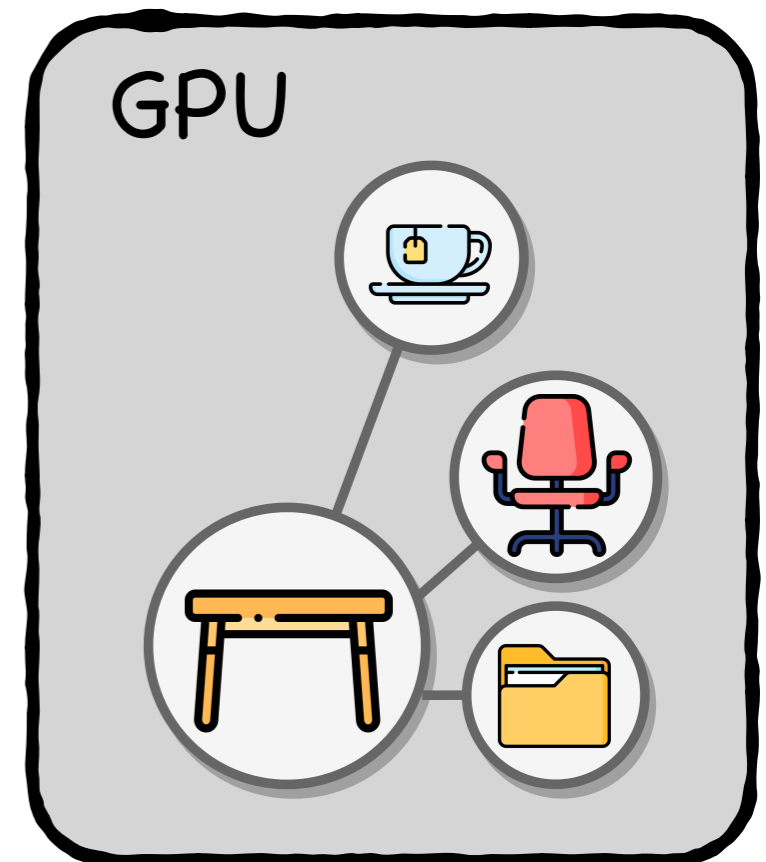
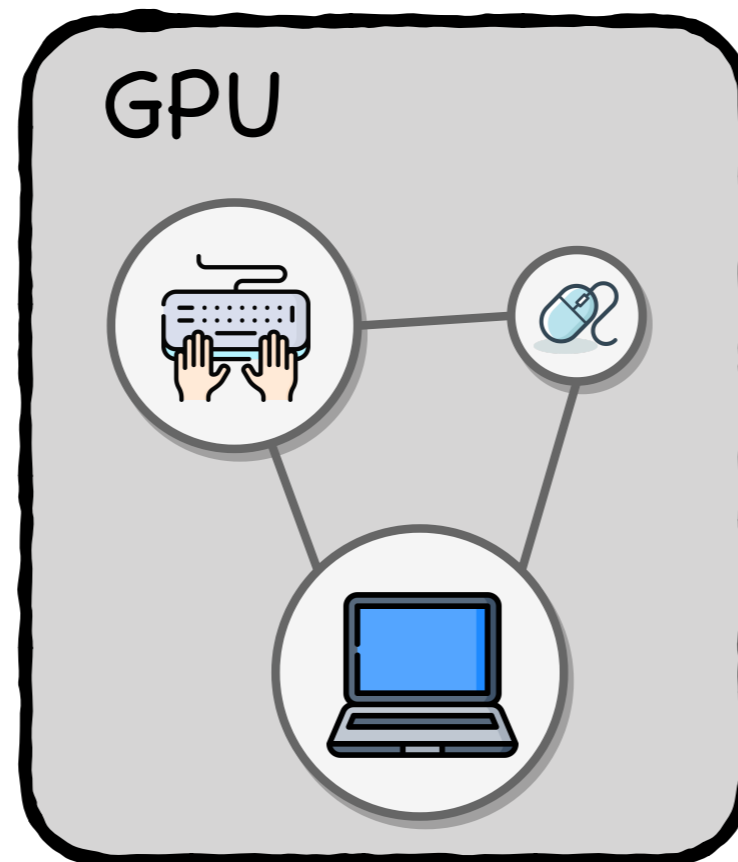
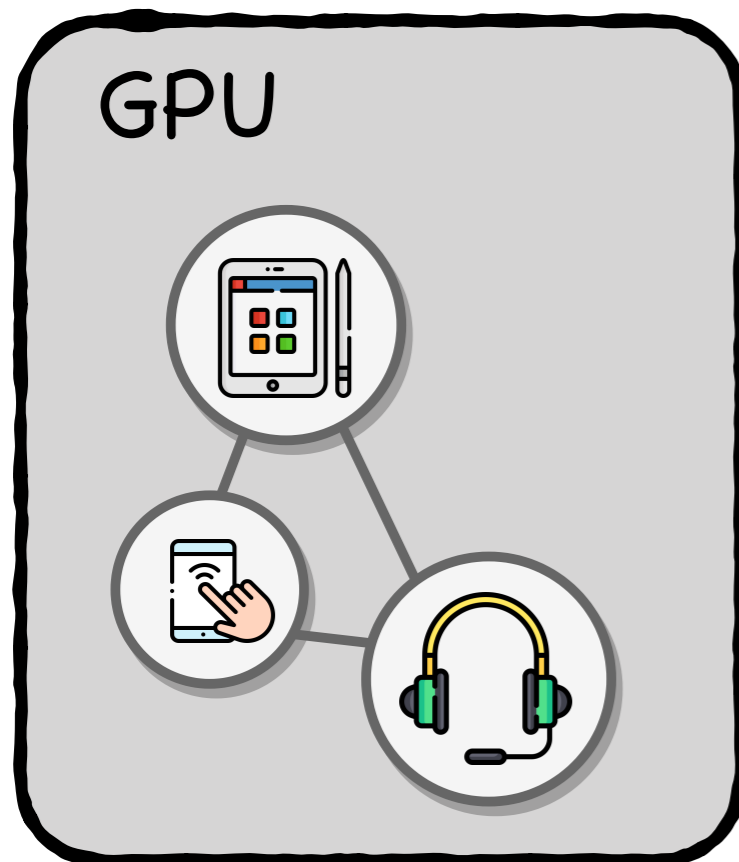
Category III



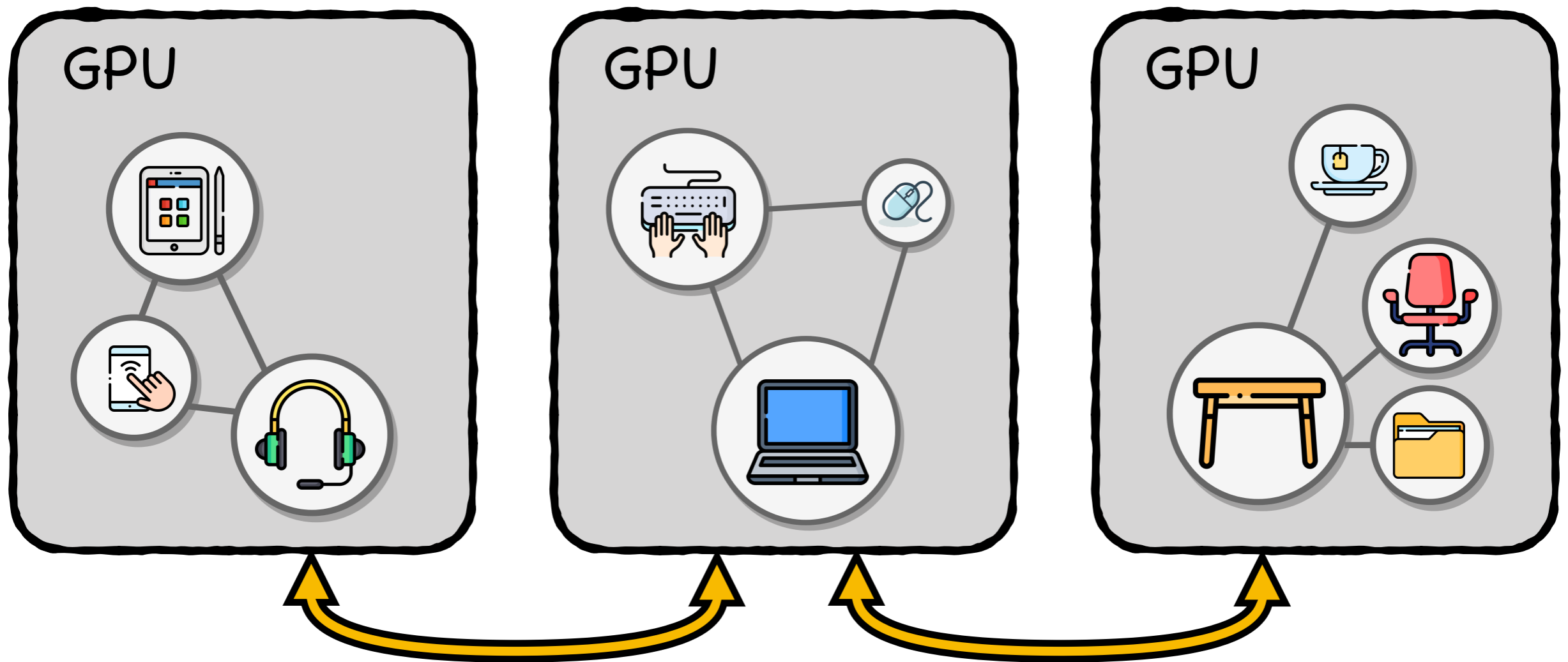
Category III



Assigning one partition to one GPU

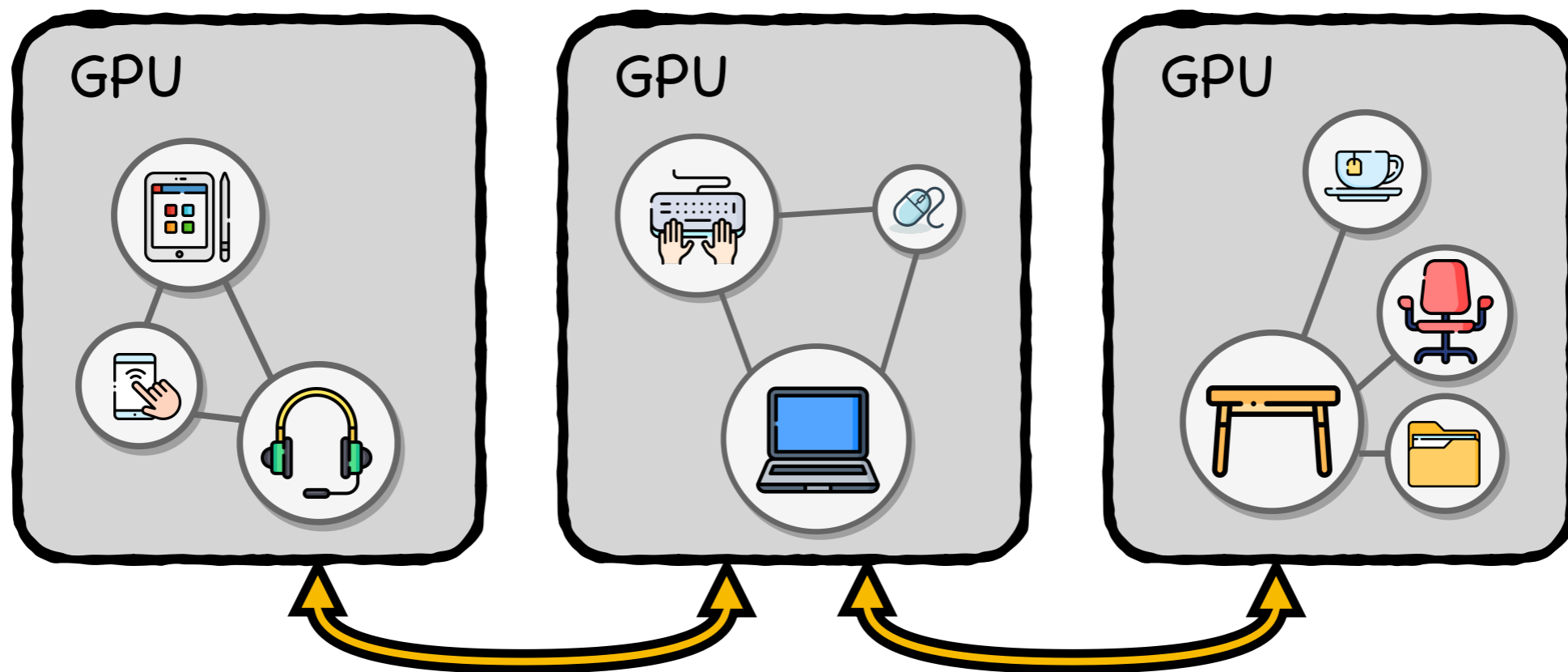


Assigning one partition to one GPU



Point-to-point

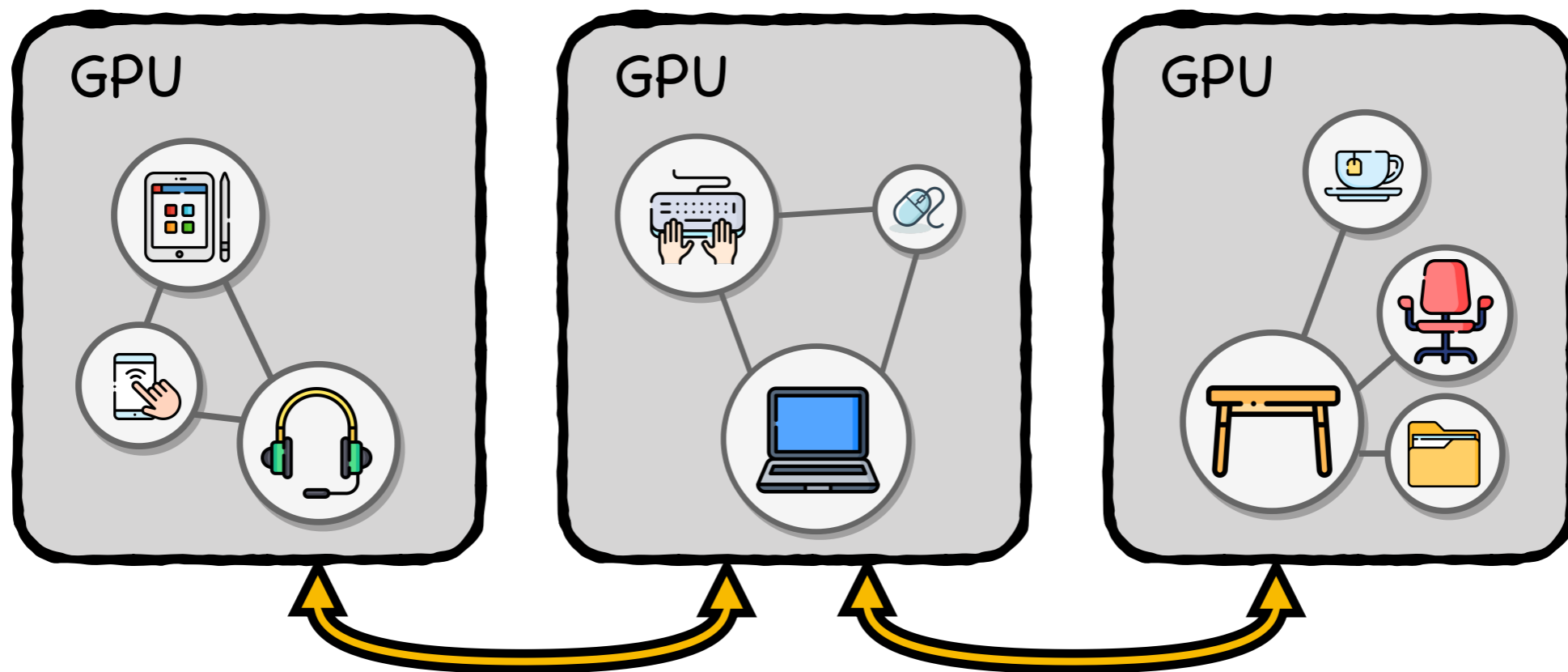
Category III: Partition-Based Methods



[1] Thorpe et al. Dorylus: Affordable, Scalable, and Accurate GNN Training with Distributed CPU Servers and Serverless Threads. *OSDI'21*

[2] Wan et al. PipeGCN: Efficient Full-Graph Training of Graph Convolutional Networks with Pipelined Feature Communication. *ICLR'22*

Category III: Partition-Based Methods



Pro: Reduced Communication
The drawback is not well studied

[1] Thorpe et al. Dorylus: Affordable, Scalable, and Accurate GNN Training with Distributed CPU Servers and Serverless Threads. *OSDI'21*

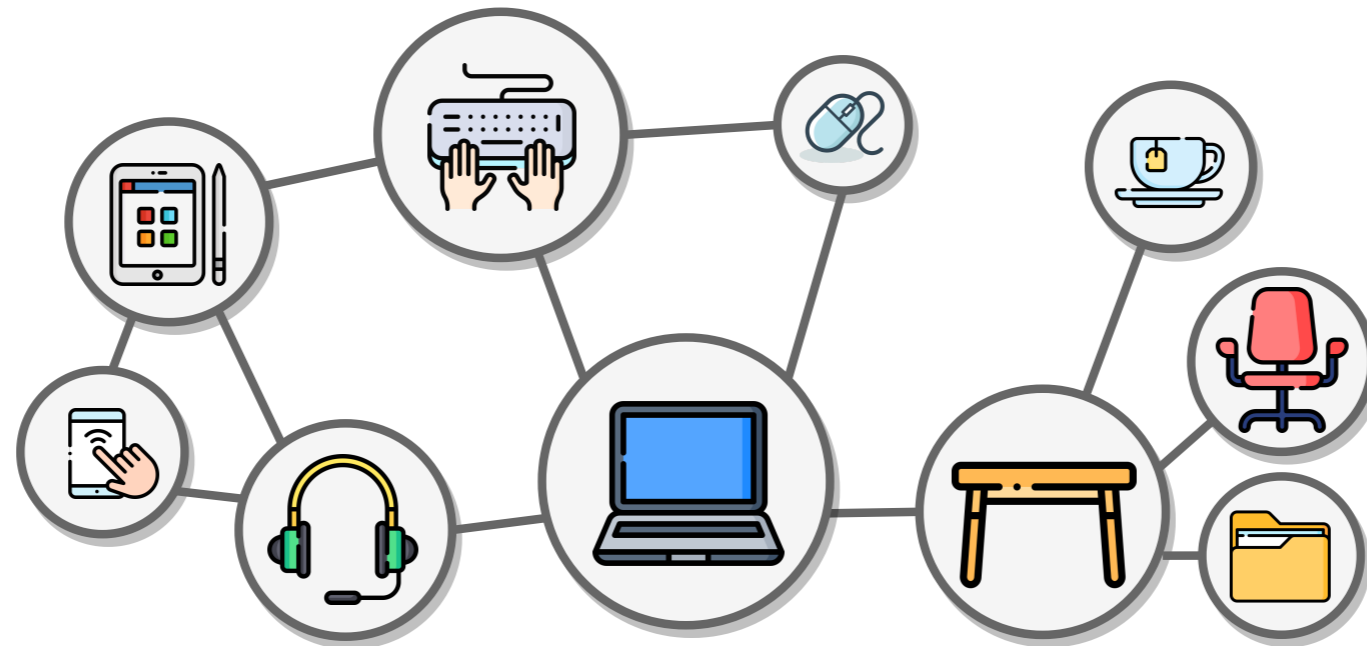
[2] Wan et al. PipeGCN: Efficient Full-Graph Training of Graph Convolutional Networks with Pipelined Feature Communication. *ICLR'22*

BNS-GCN

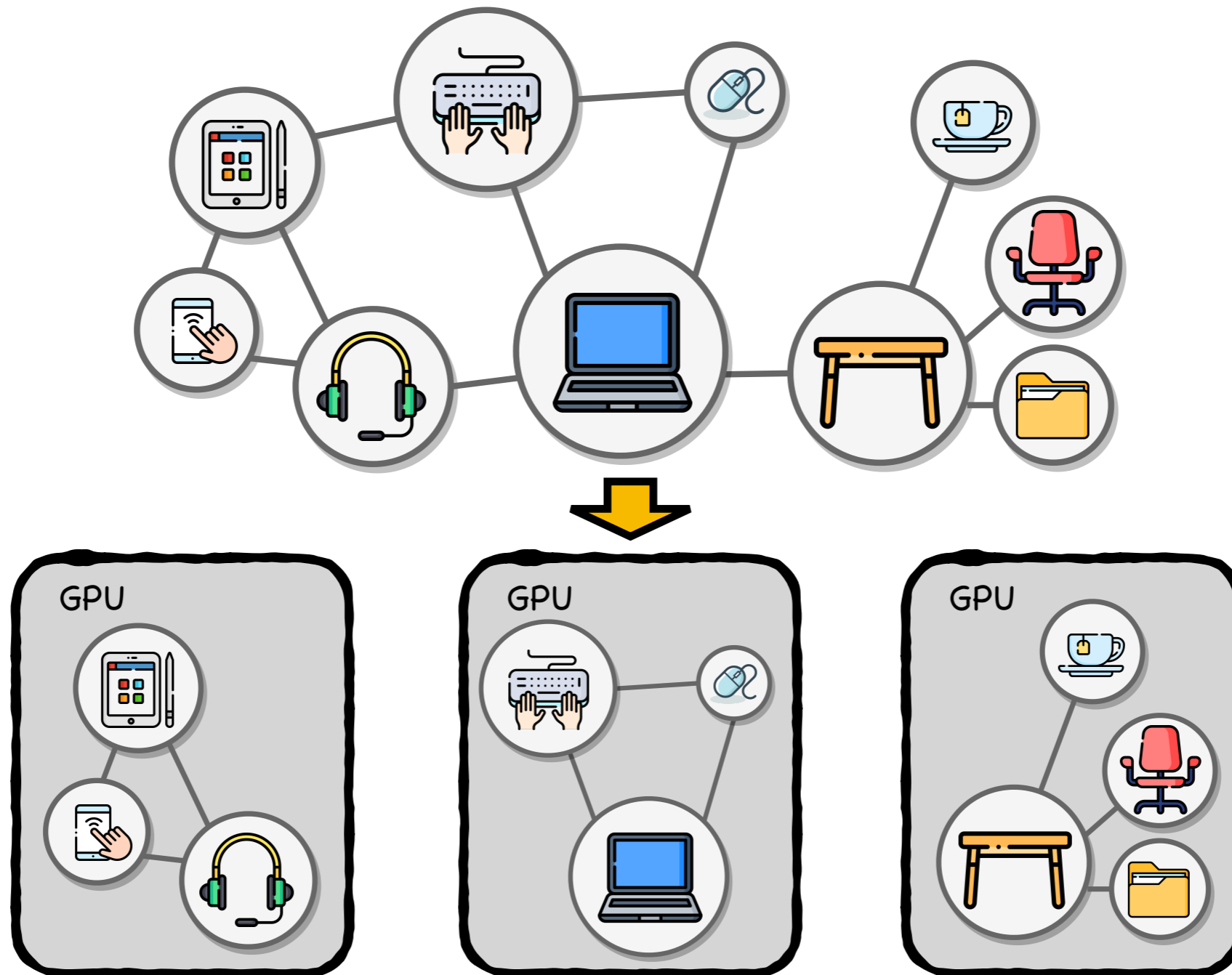
BNS-GCN

- ◆ Identifying **drawbacks** of partition-based training
- ◆ Proposing a simple-yet-effective **solution**
- ◆ Providing theoretical and empirical **validation**

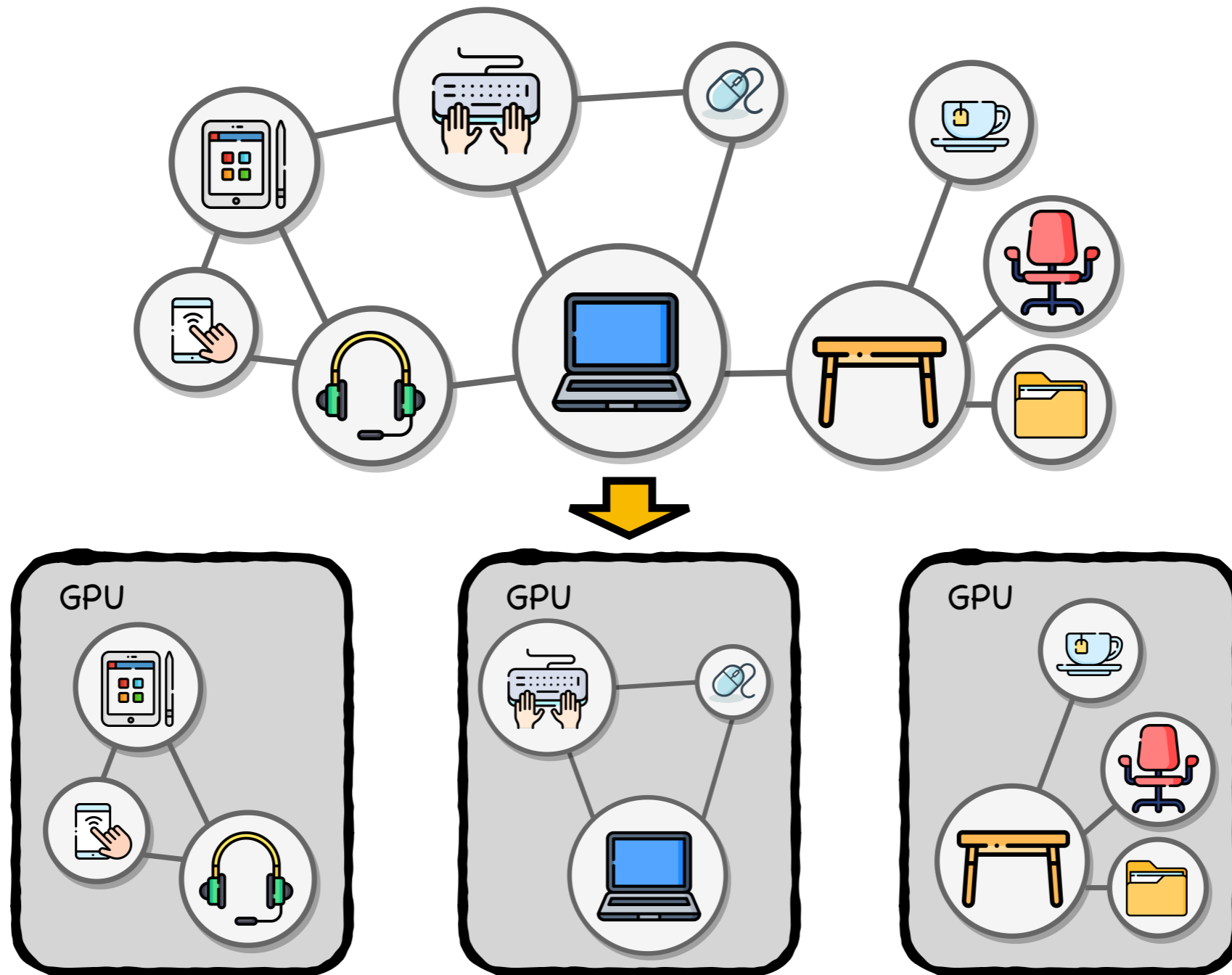
Understanding Partition-Based Methods



Understanding Partition-Based Methods

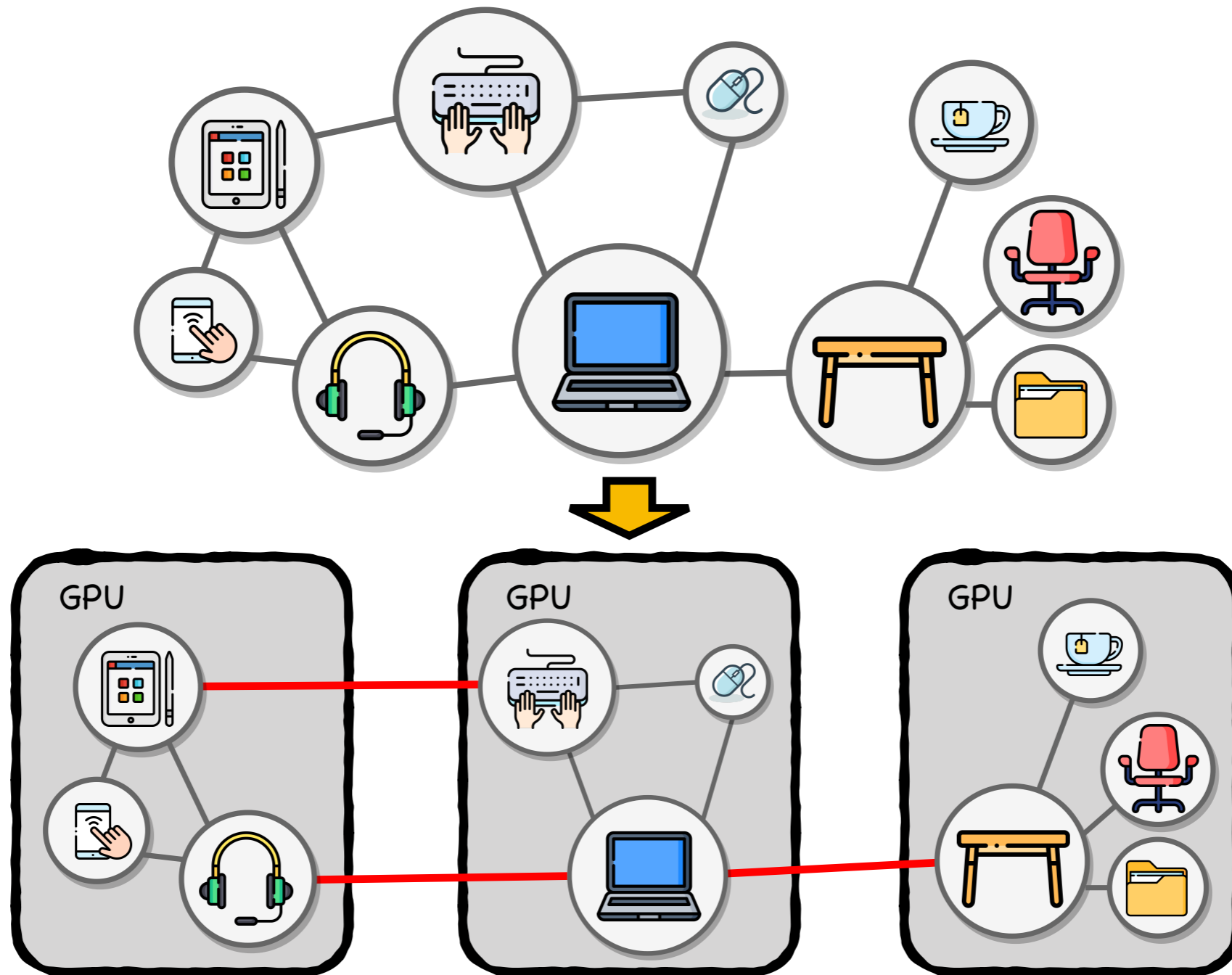


Understanding Partition-Based Methods



Similar to Data Parallelism

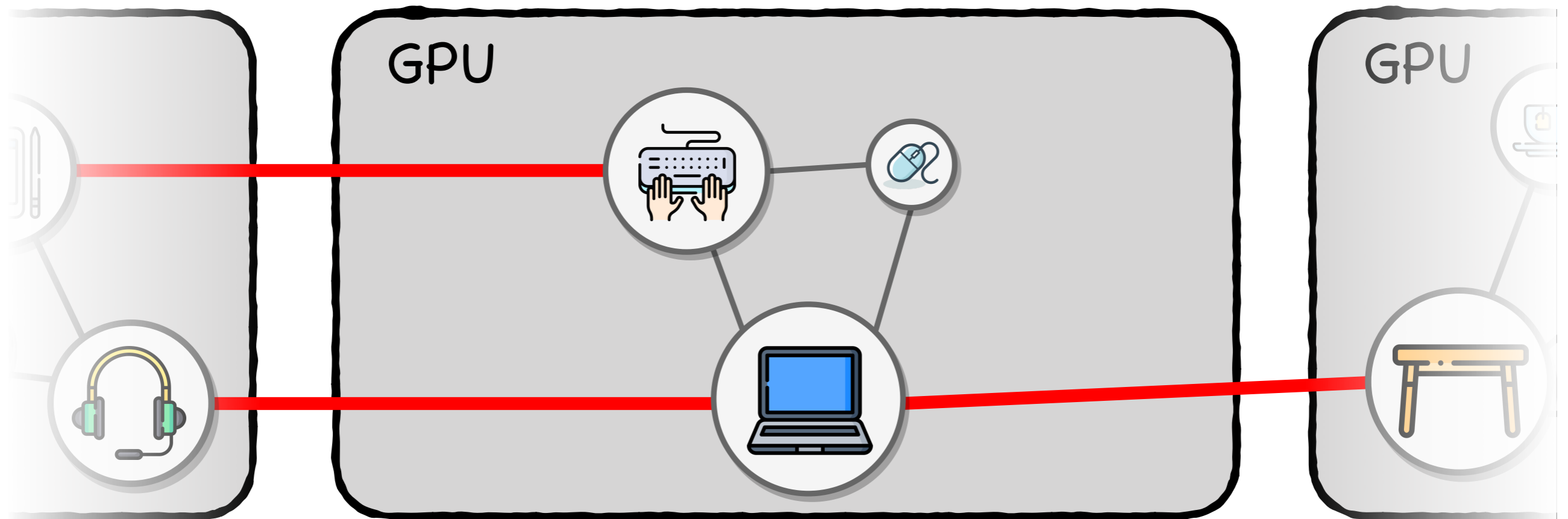
Understanding Partition-Based Methods



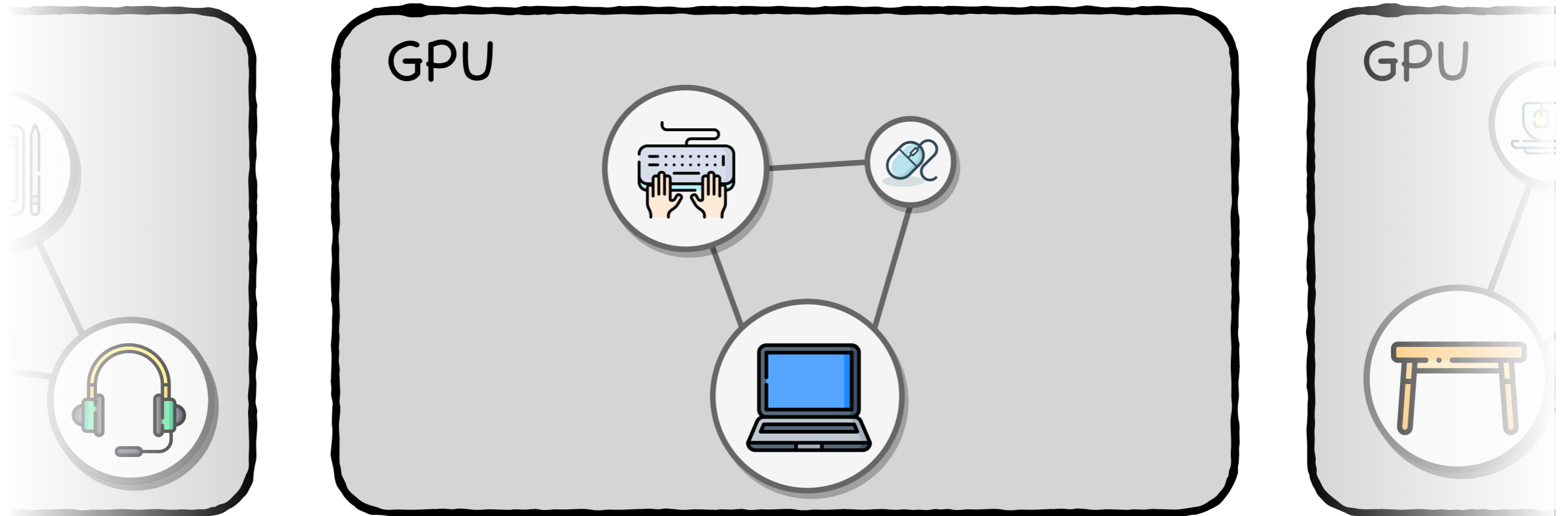
Similar to Data Parallelism

Difference: **Dependency** among Data

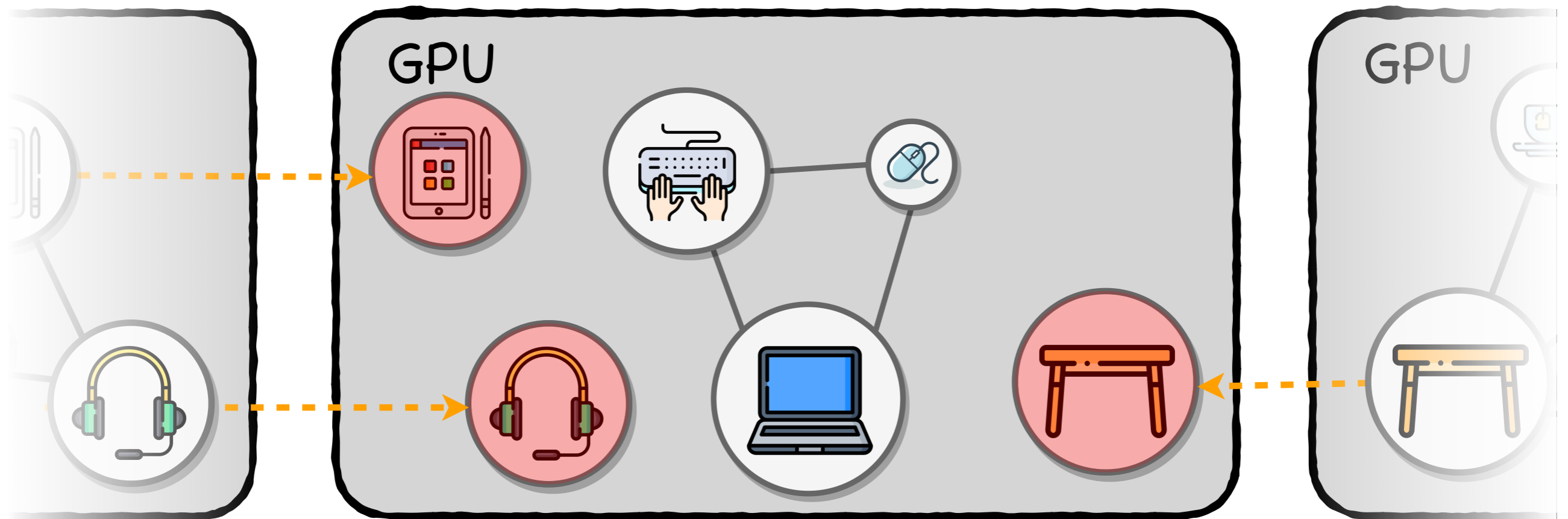
Understanding Partition-Based Methods



Understanding Partition-Based Methods

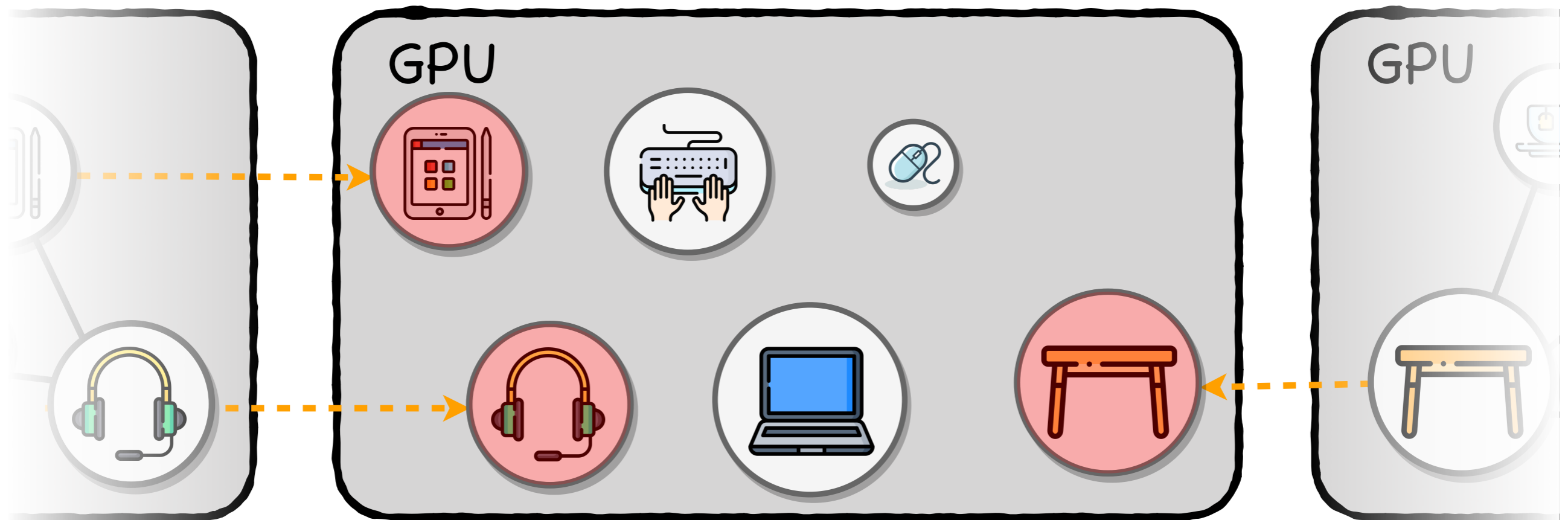


Understanding Partition-Based Methods

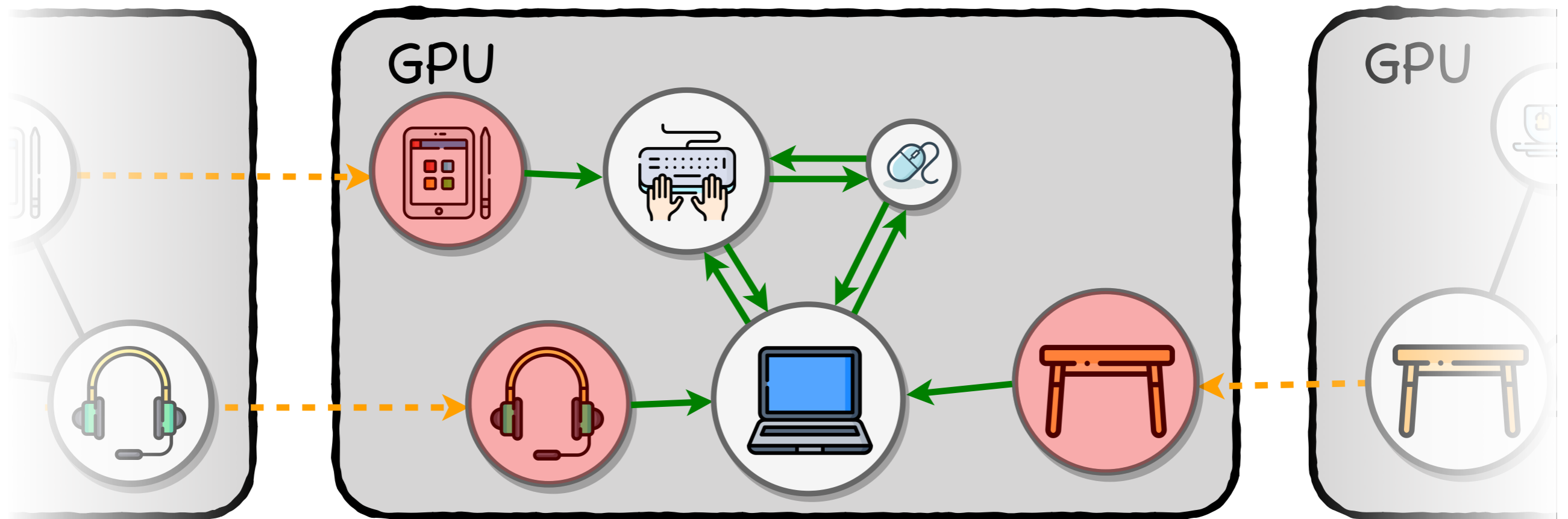


Communicating remote features

Understanding Partition-Based Methods

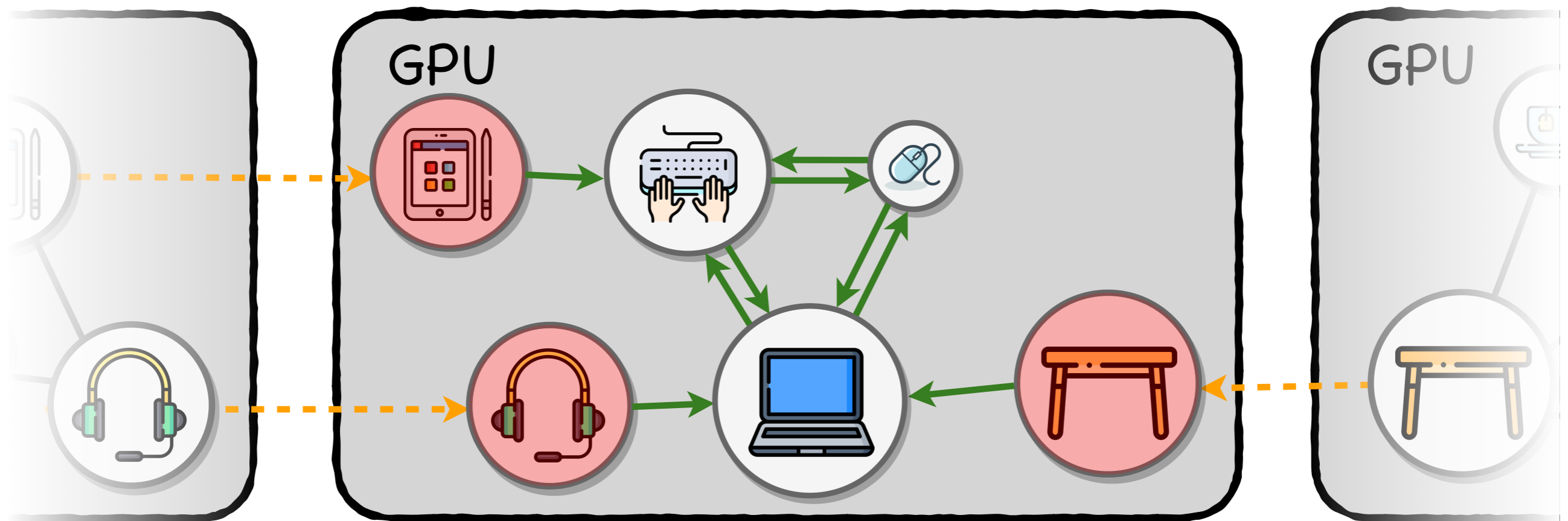


Understanding Partition-Based Methods



Computing local features

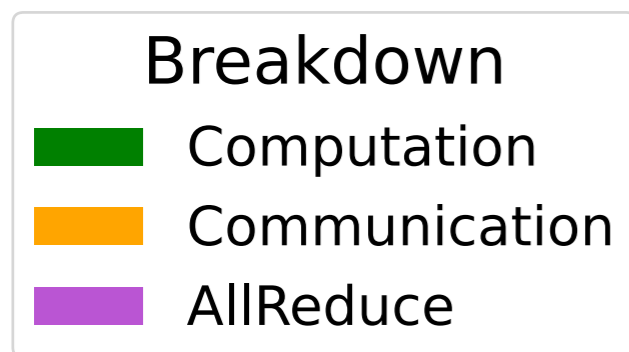
Understanding Partition-Based Methods



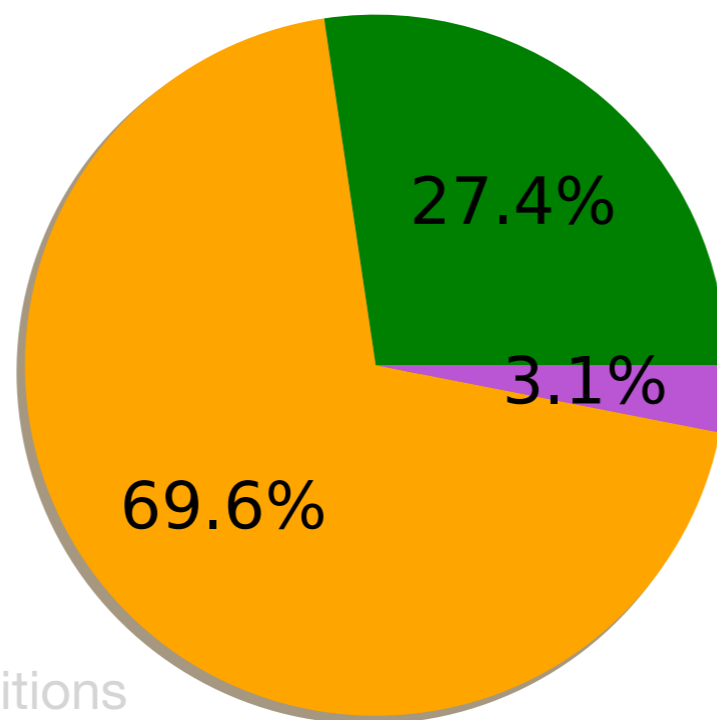
○ Inner Node ● Boundary Node

Identifying Drawbacks

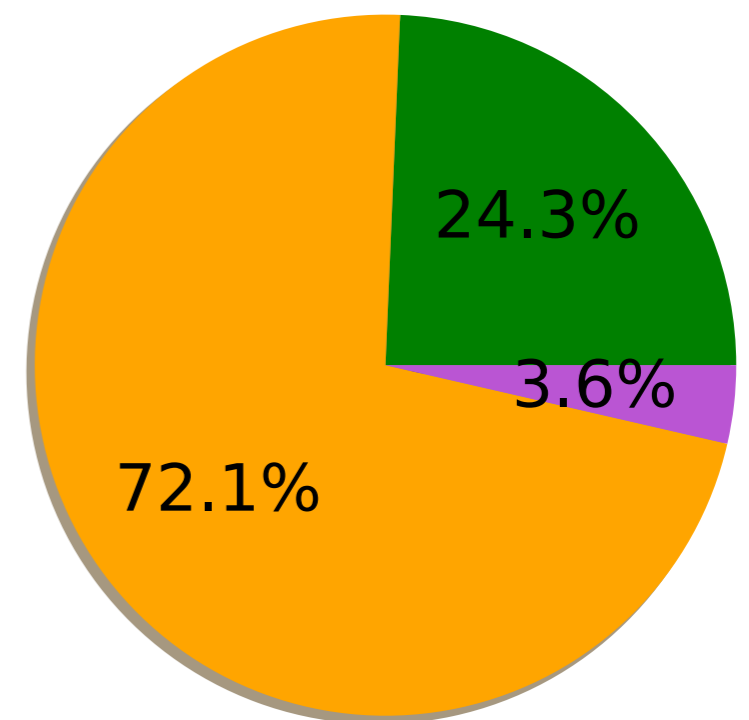
Training Time Breakdown



Reddit

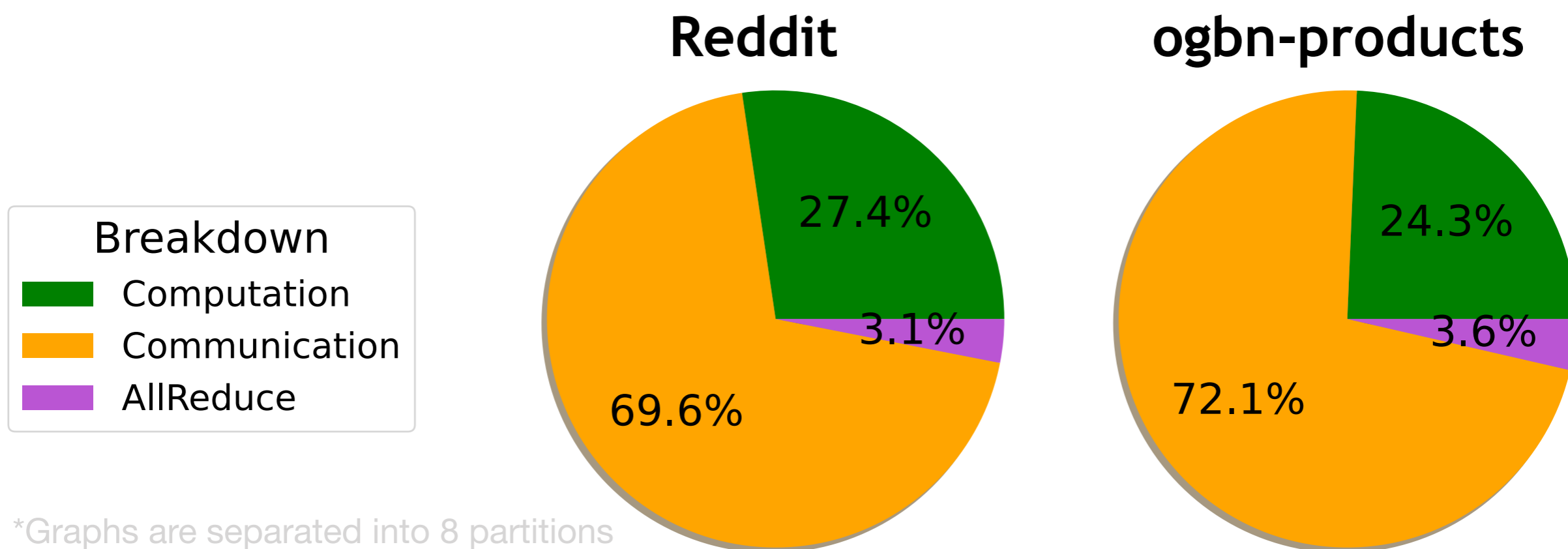


ogbn-products



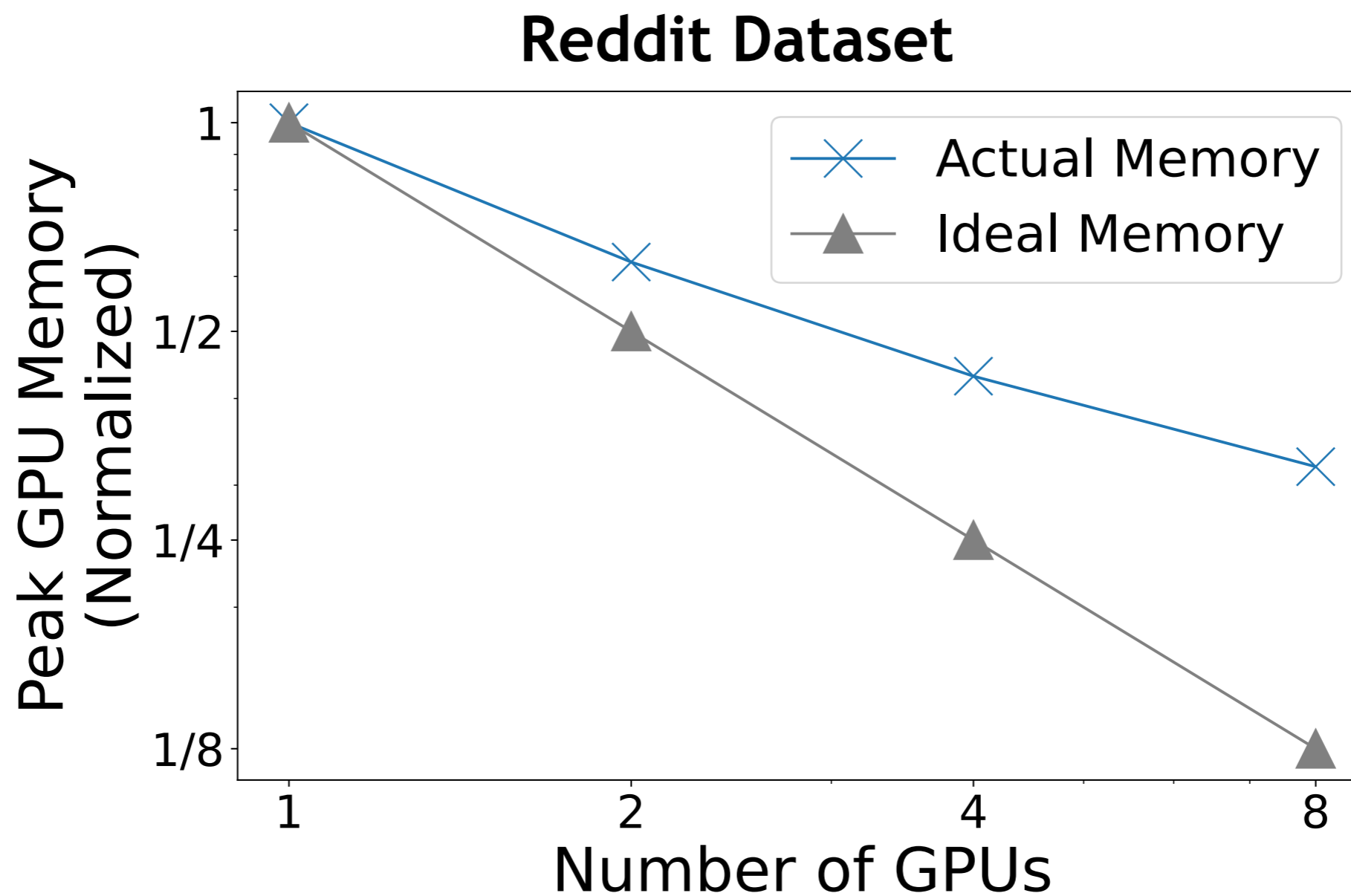
*Graphs are separated into 8 partitions

Training Time Breakdown

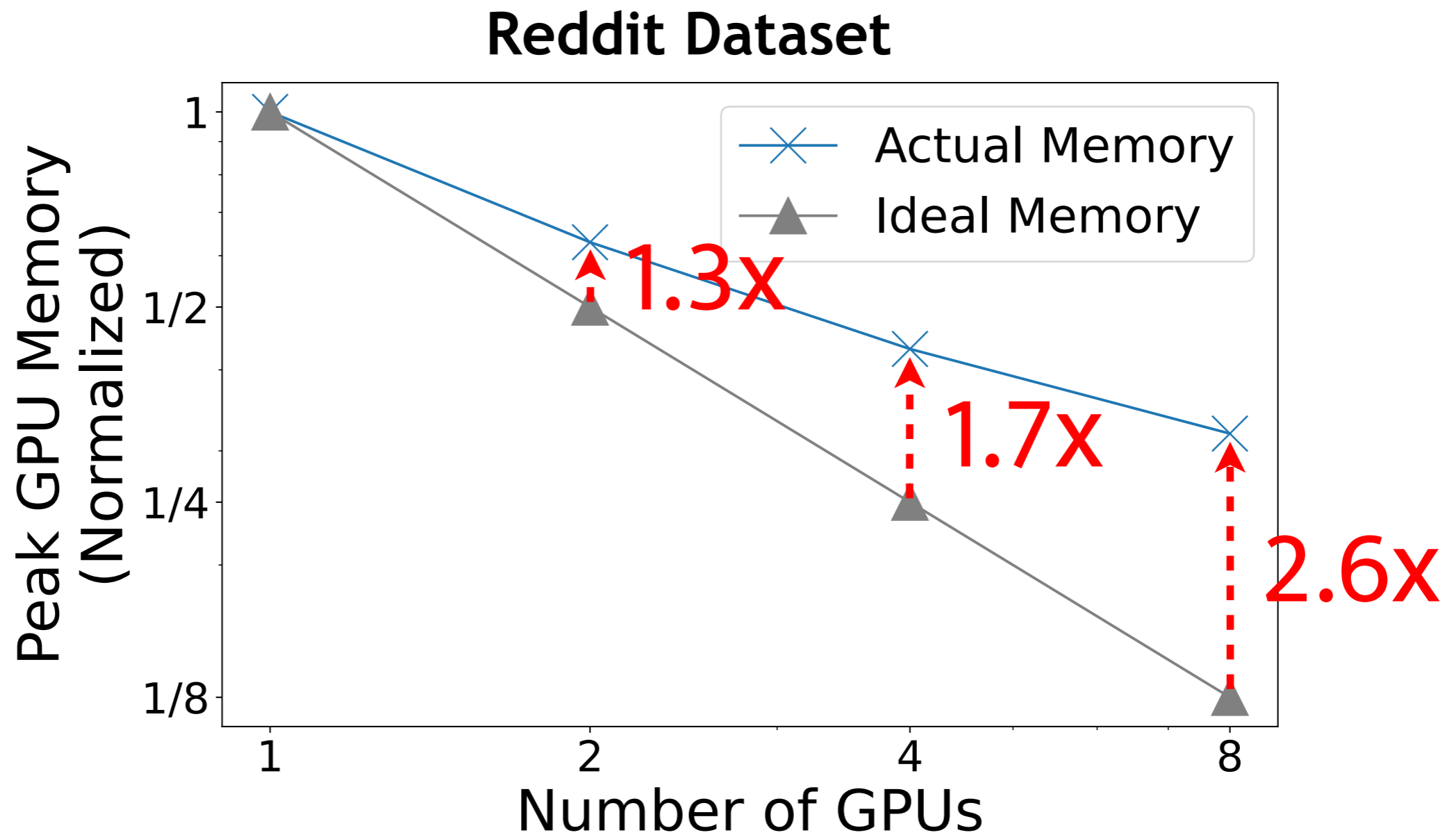


Drawback I: Significant Communication Overhead

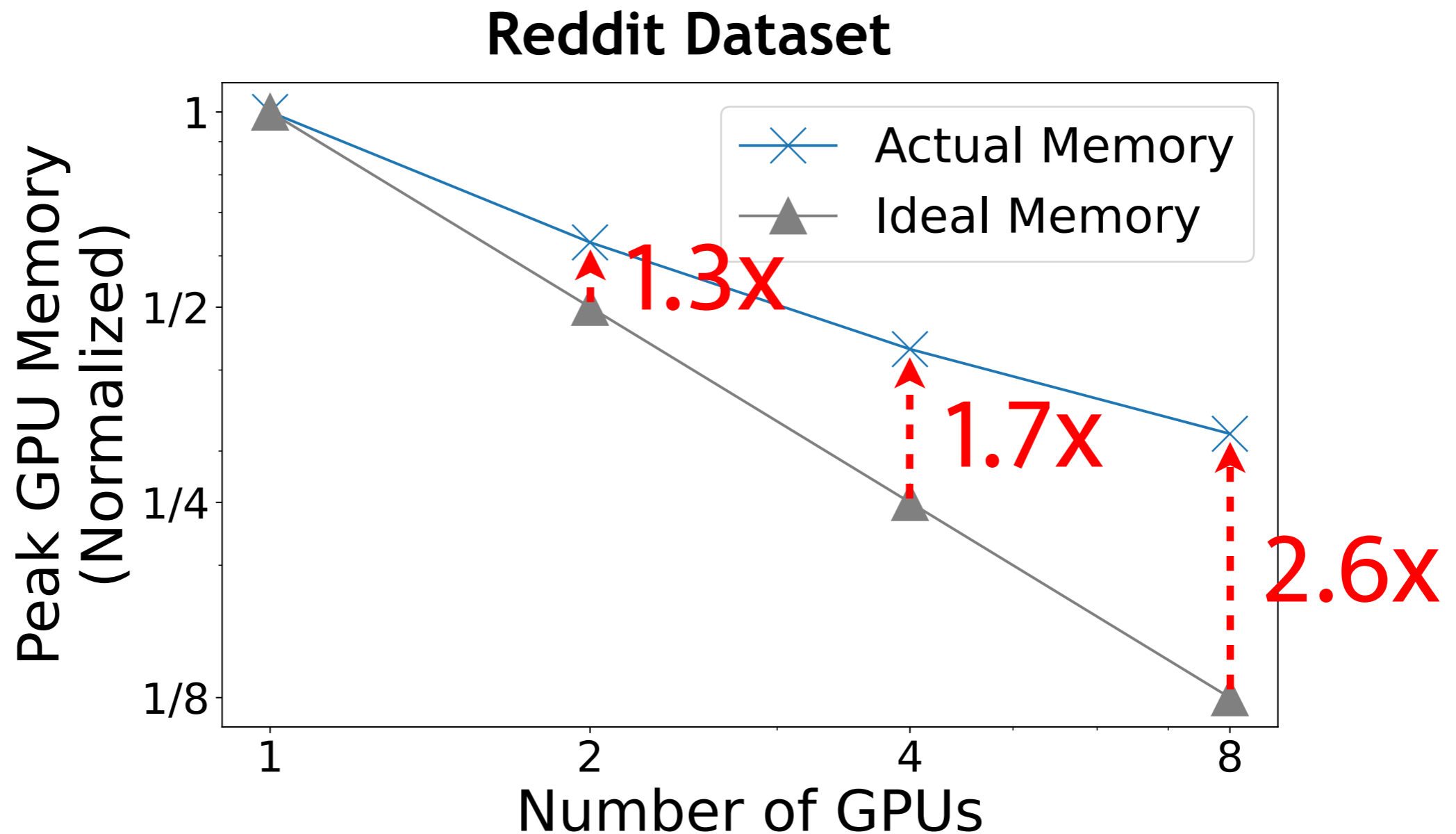
Training Memory Requirement



Training Memory Requirement



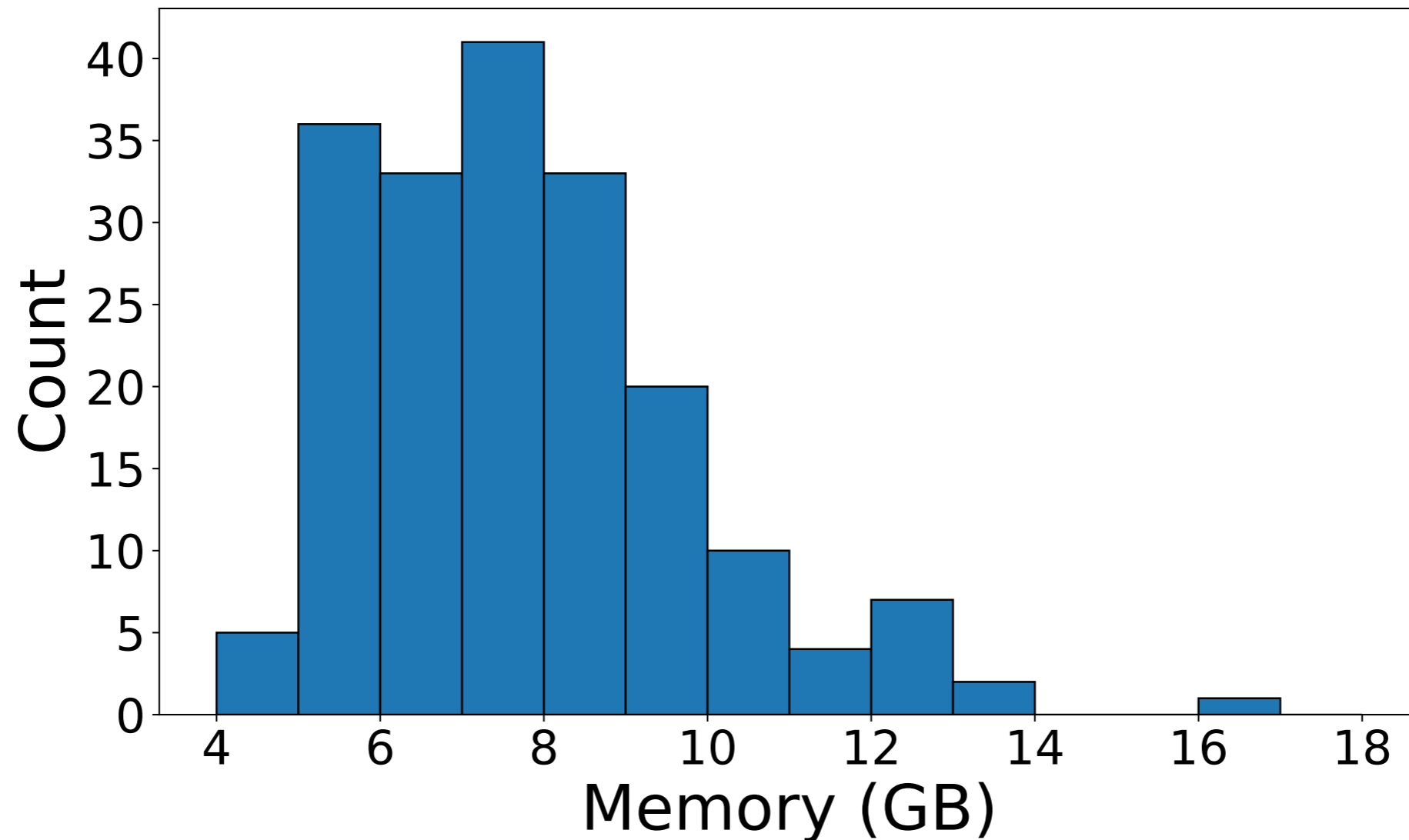
Training Memory Requirement



Drawback II: Unscalable Memory Requirement

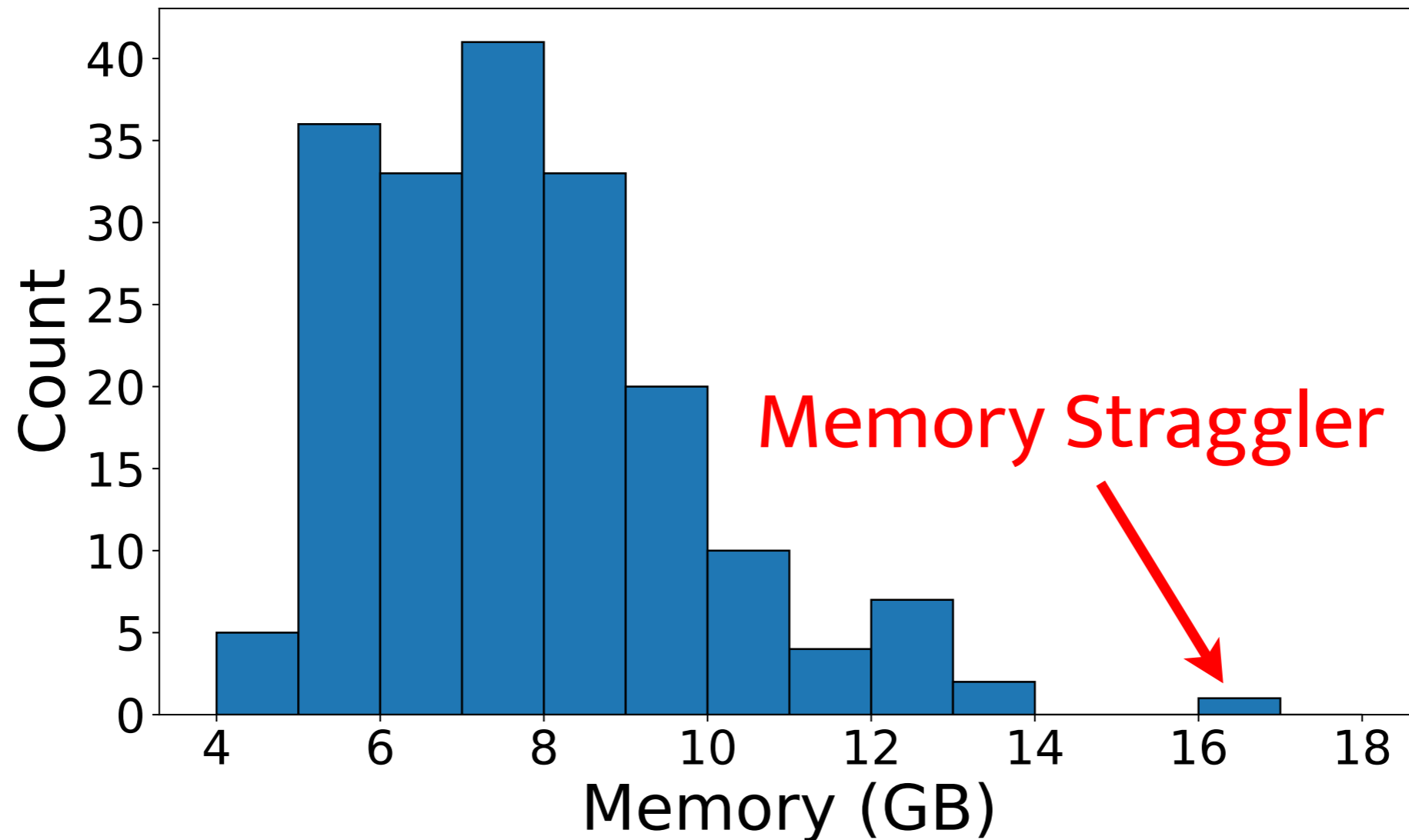
Per-GPU Memory Distribution

ogbn-papers100M (192 partitions)



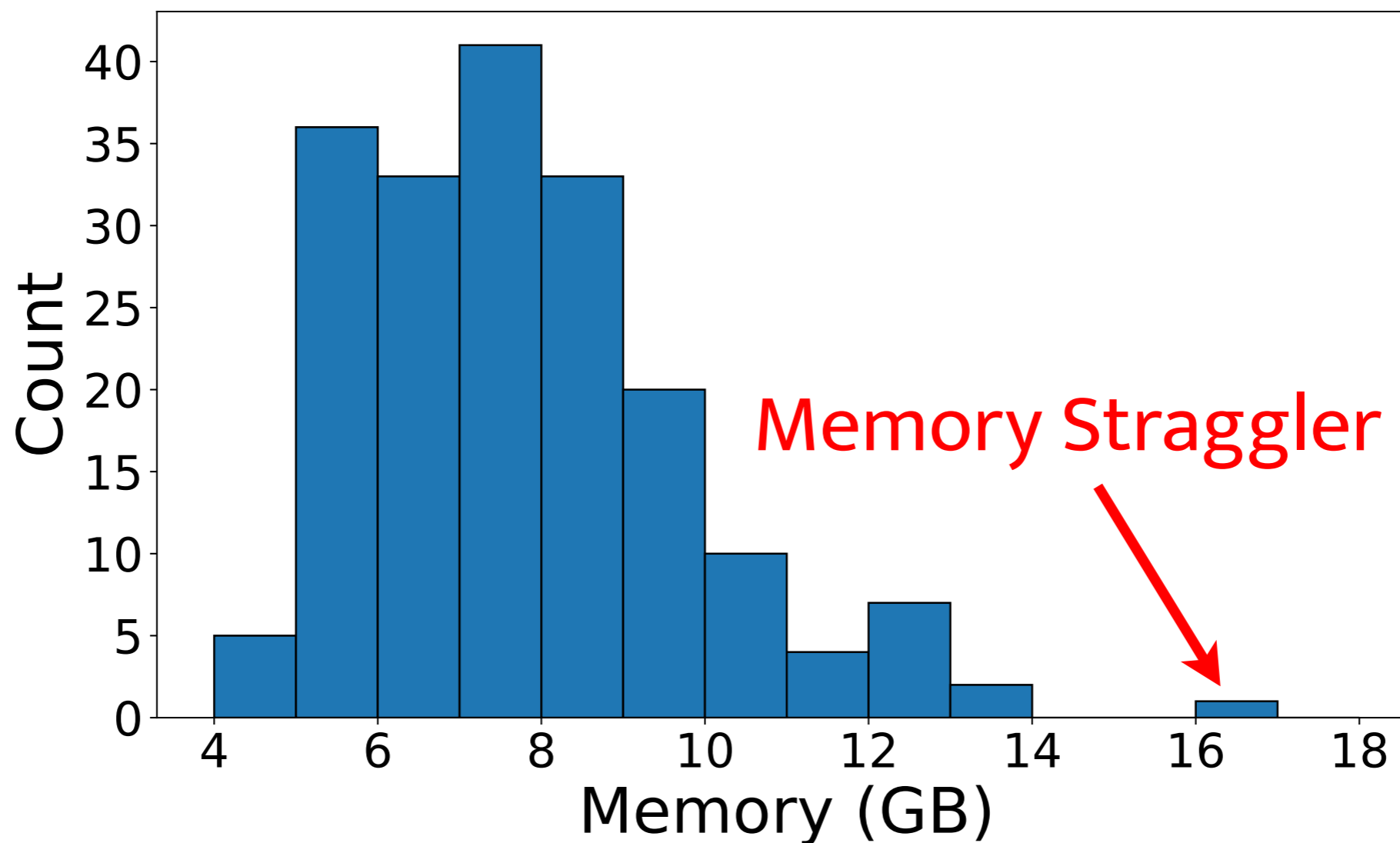
Per-GPU Memory Distribution

ogbn-papers100M (192 partitions)



Per-GPU Memory Distribution

ogbn-papers100M (192 partitions)



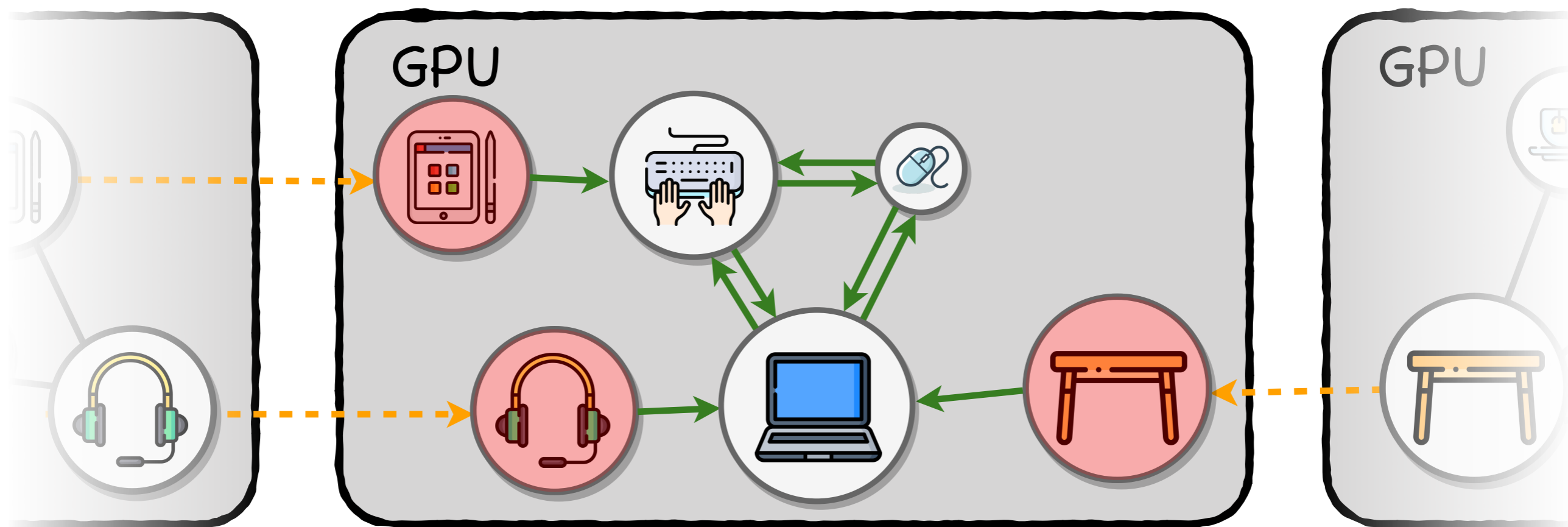
Drawback III: Imbalanced Memory across GPUs

What's the underlying **cause**?



Understanding Communication Volume

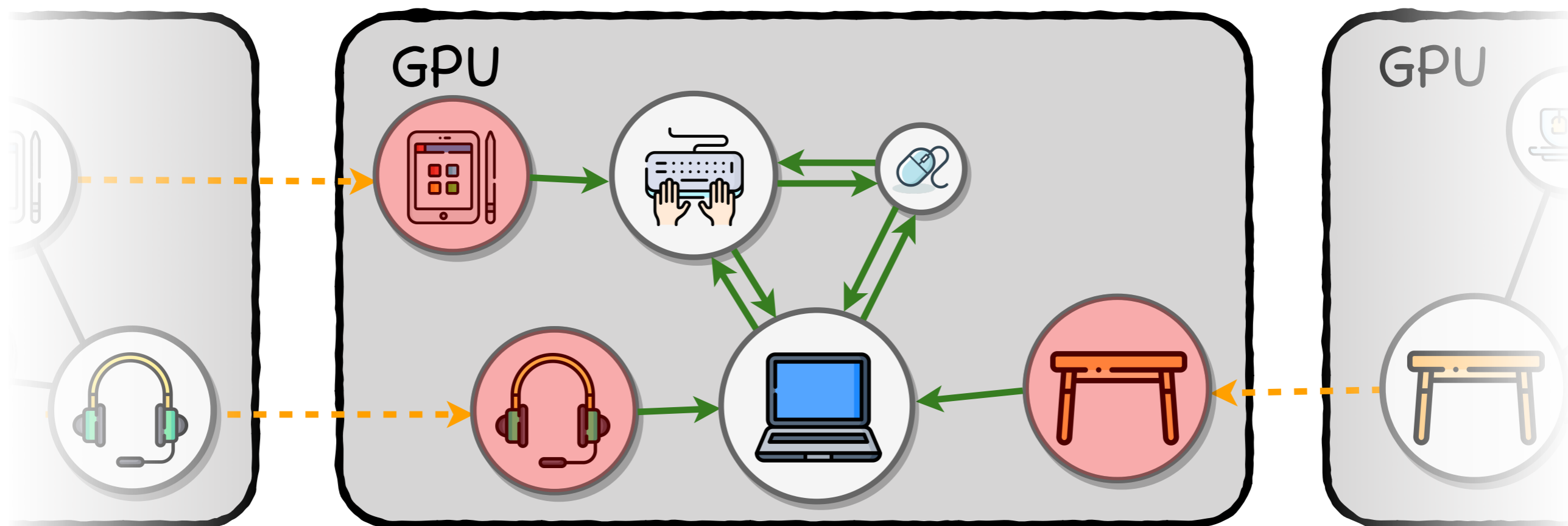
Understanding Communication Volume



○ Inner Node

● Boundary Node

Understanding Communication Volume

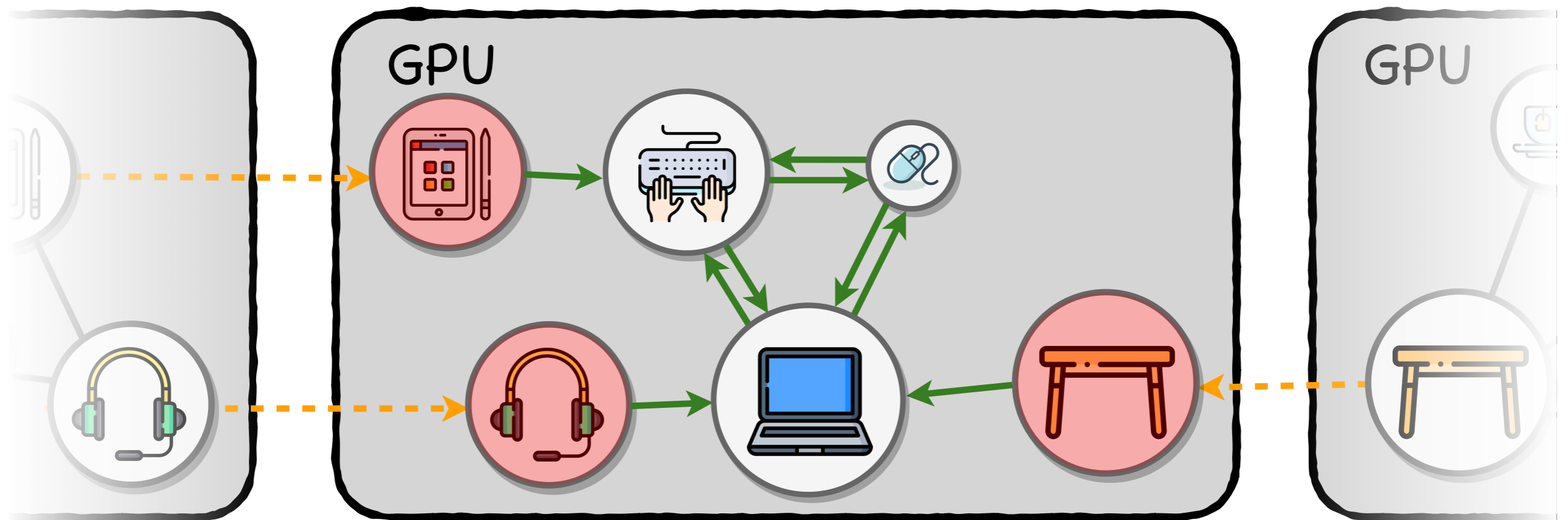


○ Inner Node

● Boundary Node

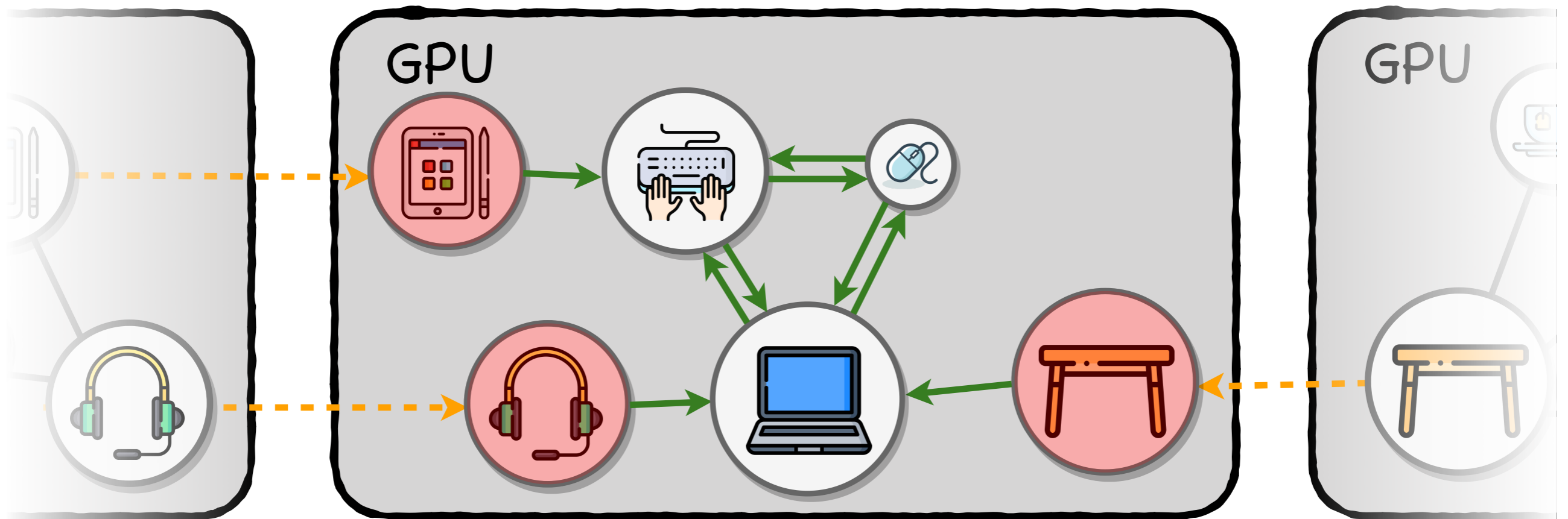
The i -th partition has $n_{in}^{(i)}$ inner nodes and $n_{bd}^{(i)}$ boundary nodes

Understanding Communication Volume



$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

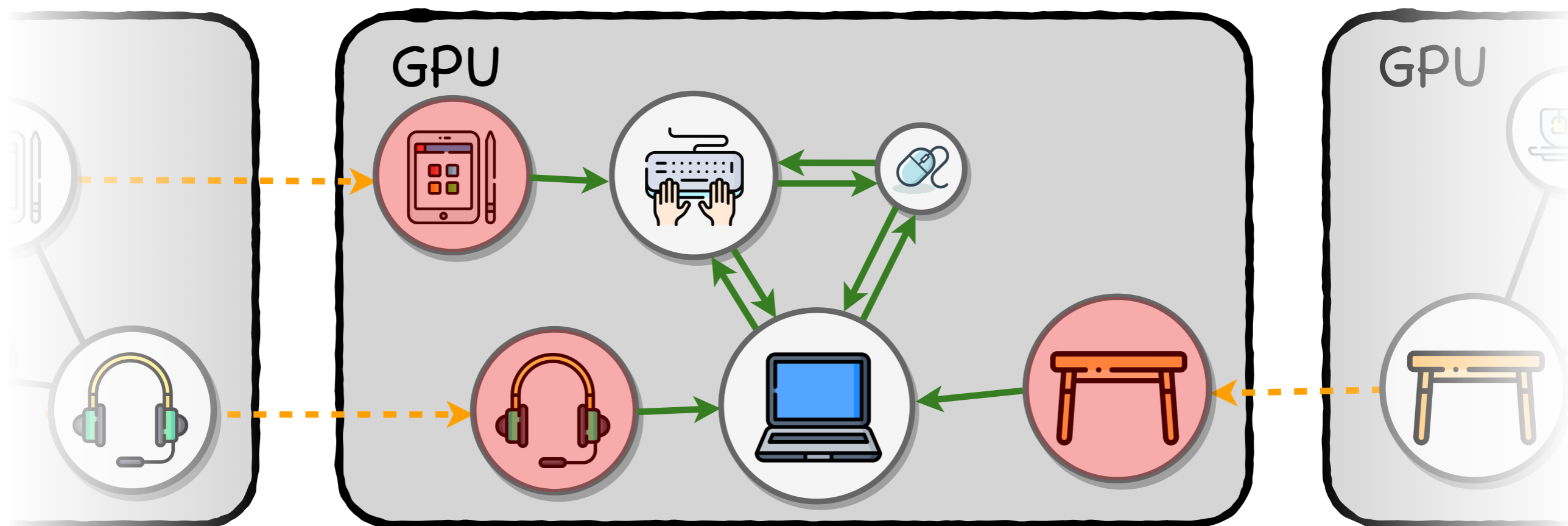
Understanding Communication Volume



$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

Comm. Volume \propto # Boundary Nodes

Understanding Communication Volume



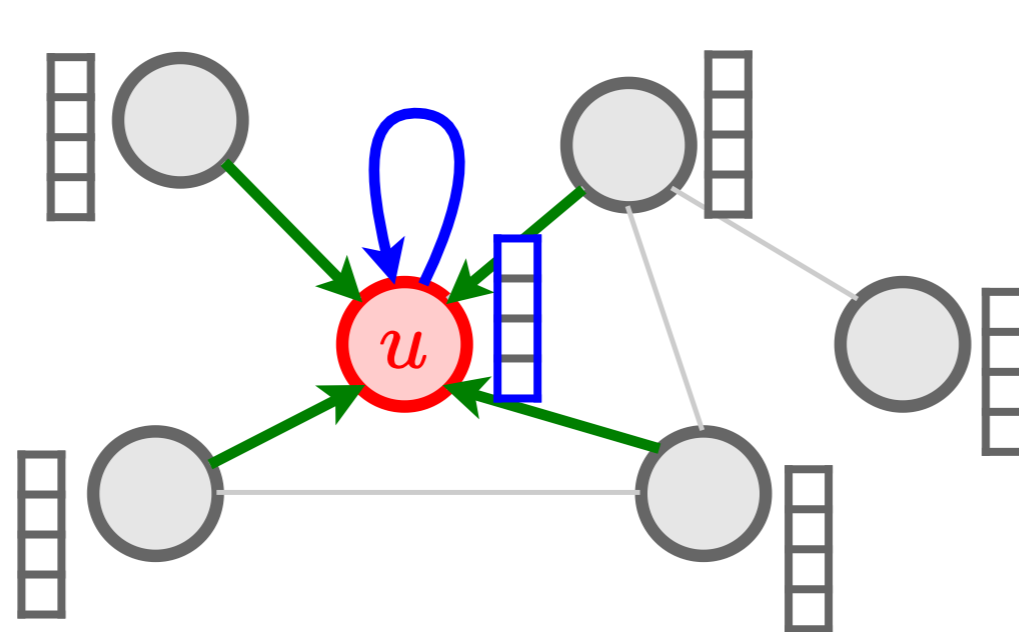
$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

Comm. Volume \propto # Boundary Nodes

! Min-Cut is not Optimal

Understanding Memory Requirement

Understanding Memory Requirement

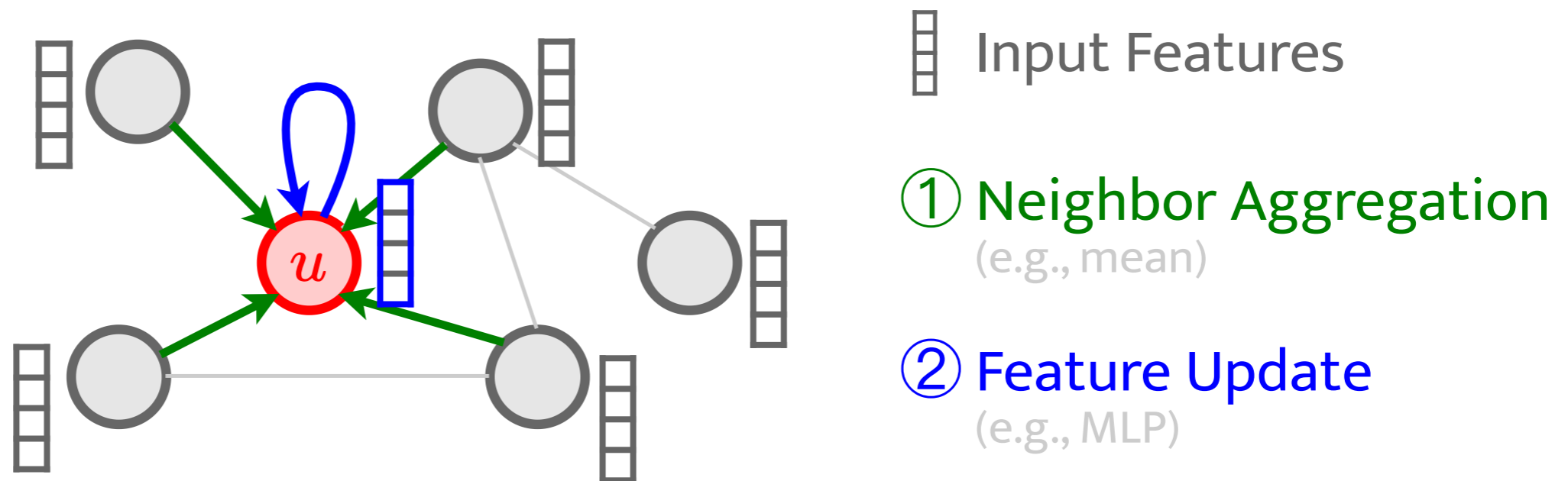


 Input Features

① Neighbor Aggregation
(e.g., mean)

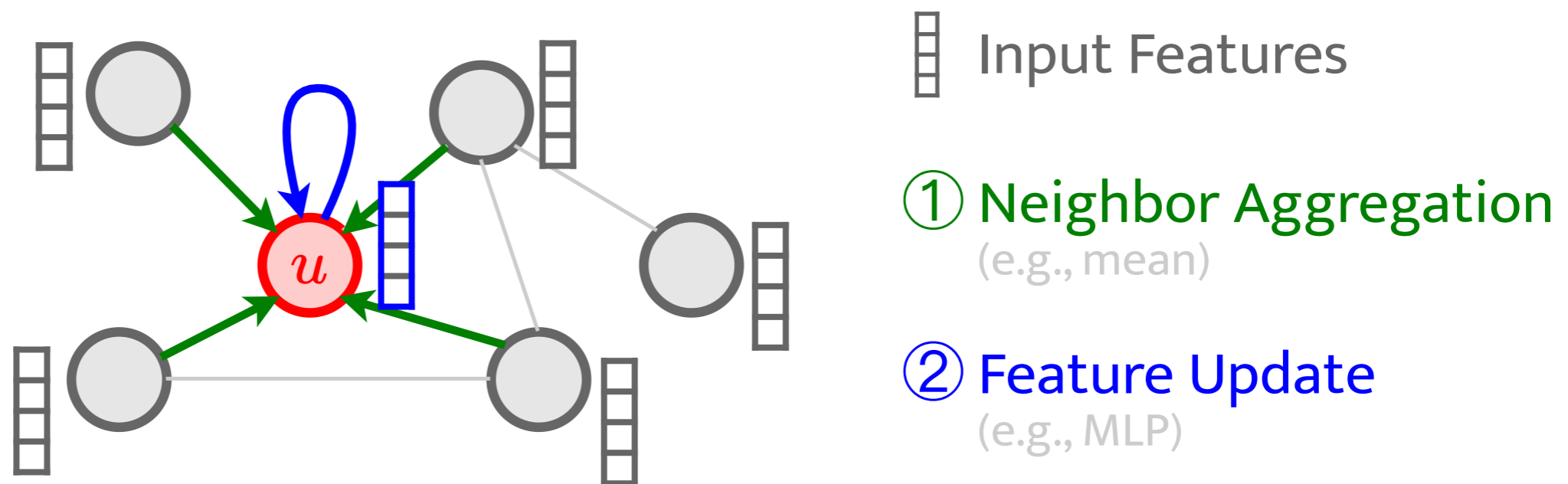
② Feature Update
(e.g., MLP)

Understanding Memory Requirement



$$\text{Mem}(\mathcal{G}_i) \propto 3n_{in}^{(i)} + n_{bd}^{(i)}$$

Understanding Memory Requirement



$$\text{Mem}(\mathcal{G}_i) \propto 3n_{in}^{(i)} + n_{bd}^{(i)}$$

Aggregation: $n_{in}^{(i)} + n_{bd}^{(i)}$

Linear + Activation: $2n_{in}^{(i)}$

Contribution I: Identify the Underlying Cause

- I Significant Communication Overhead
- II Unscalable Memory Requirement
- III Imbalanced Memory across GPUs

What's the underlying cause?

Contribution I: Identify the Underlying Cause

- I Significant Communication Overhead —
- II Unscalable Memory Requirement
- III Imbalanced Memory across GPUs

$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

$$\text{Mem}(\mathcal{G}_i) \propto 3n_{in}^{(i)} + n_{bd}^{(i)}$$

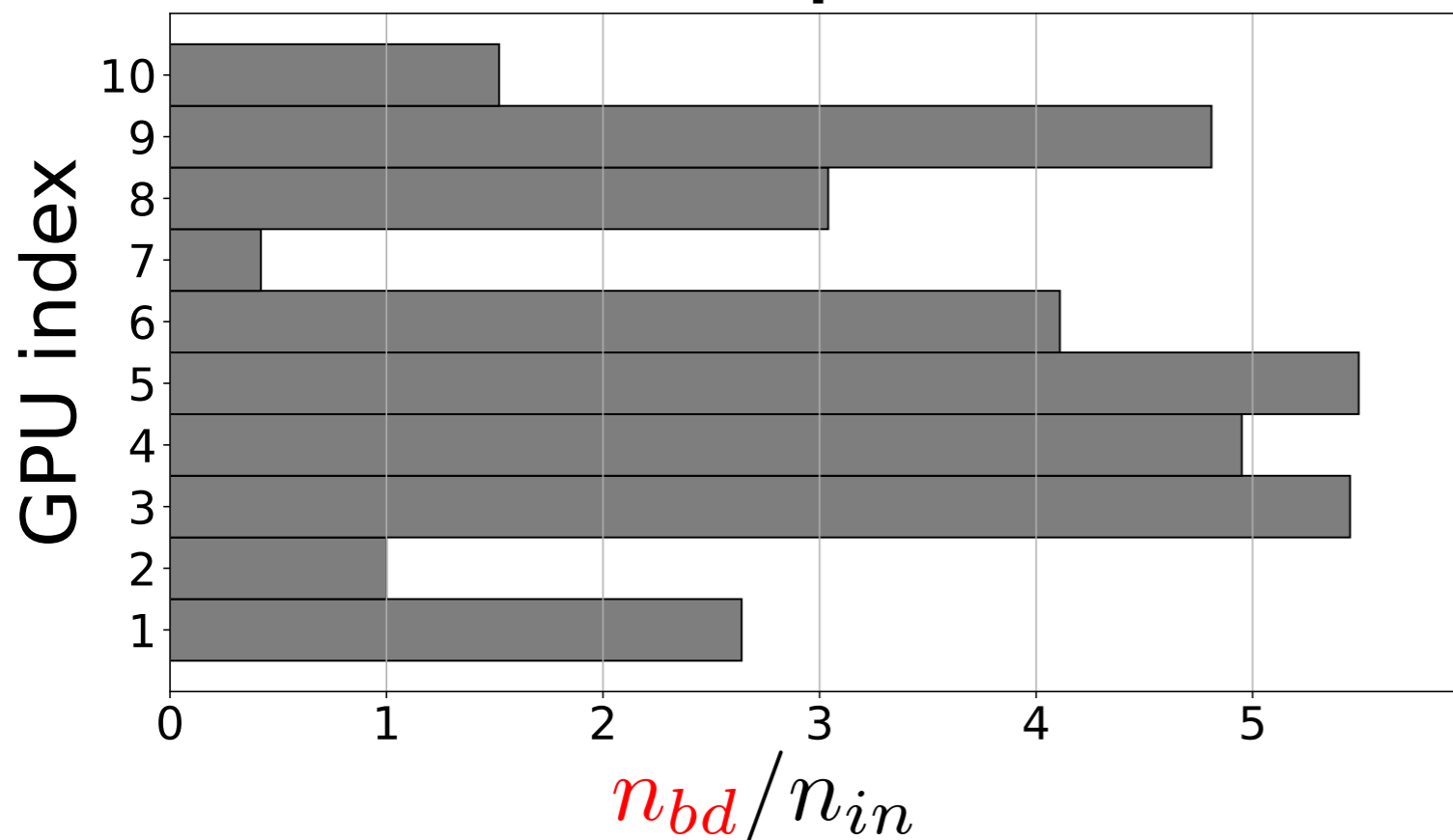
Contribution I: Identify the Underlying Cause

- I Significant Communication Overhead —
- II Unscalable Memory Requirement
- III Imbalanced Memory across GPUs

$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

$$\text{Mem}(\mathcal{G}_i) \propto 3n_{in}^{(i)} + n_{bd}^{(i)}$$

Reddit (10 partitions)



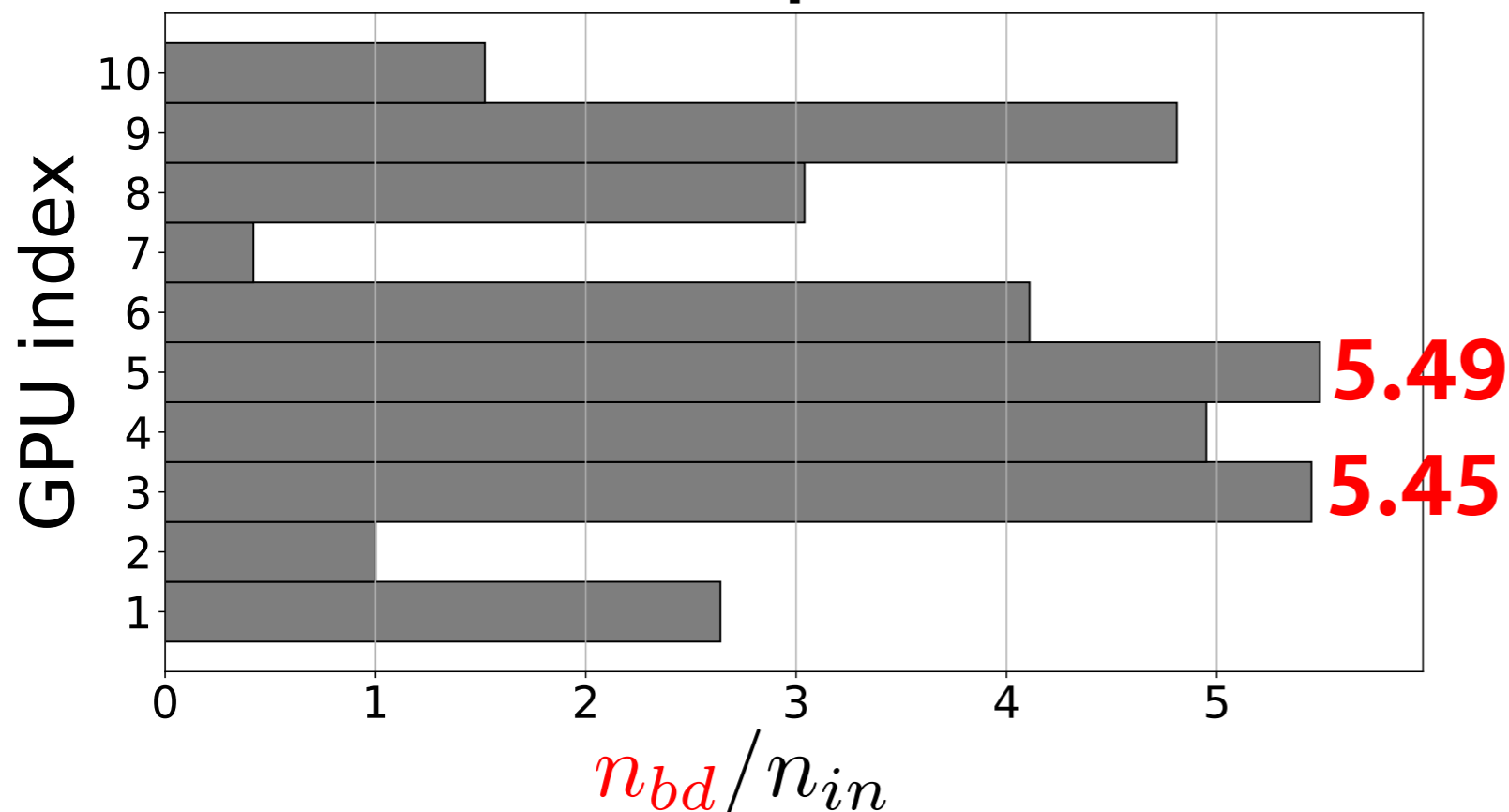
Contribution I: Identify the Underlying Cause

- I Significant Communication Overhead —
- II Unscalable Memory Requirement
- III Imbalanced Memory across GPUs

$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

$$\text{Mem}(\mathcal{G}_i) \propto 3n_{in}^{(i)} + n_{bd}^{(i)}$$

Reddit (10 partitions)



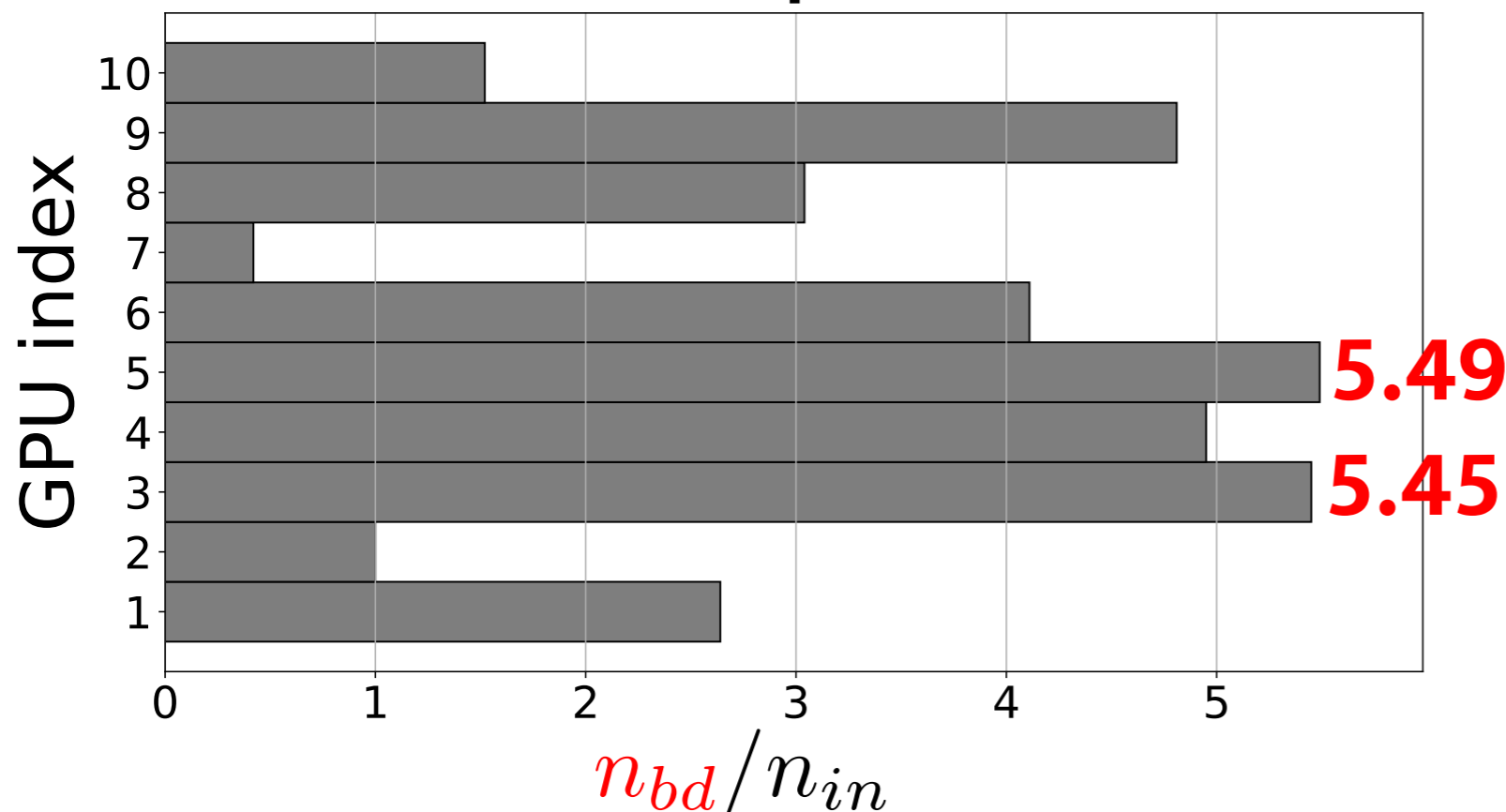
Contribution I: Identify the Underlying Cause

- I Significant Communication Overhead —
- II Unscalable Memory Requirement
- III Imbalanced Memory across GPUs

$$\text{Vol}_{\text{total}} = \sum_i \text{Vol}(\mathcal{G}_i) = \sum_i n_{bd}^{(i)}$$

$$\text{Mem}(\mathcal{G}_i) \propto 3n_{in}^{(i)} + n_{bd}^{(i)}$$

Reddit (10 partitions)

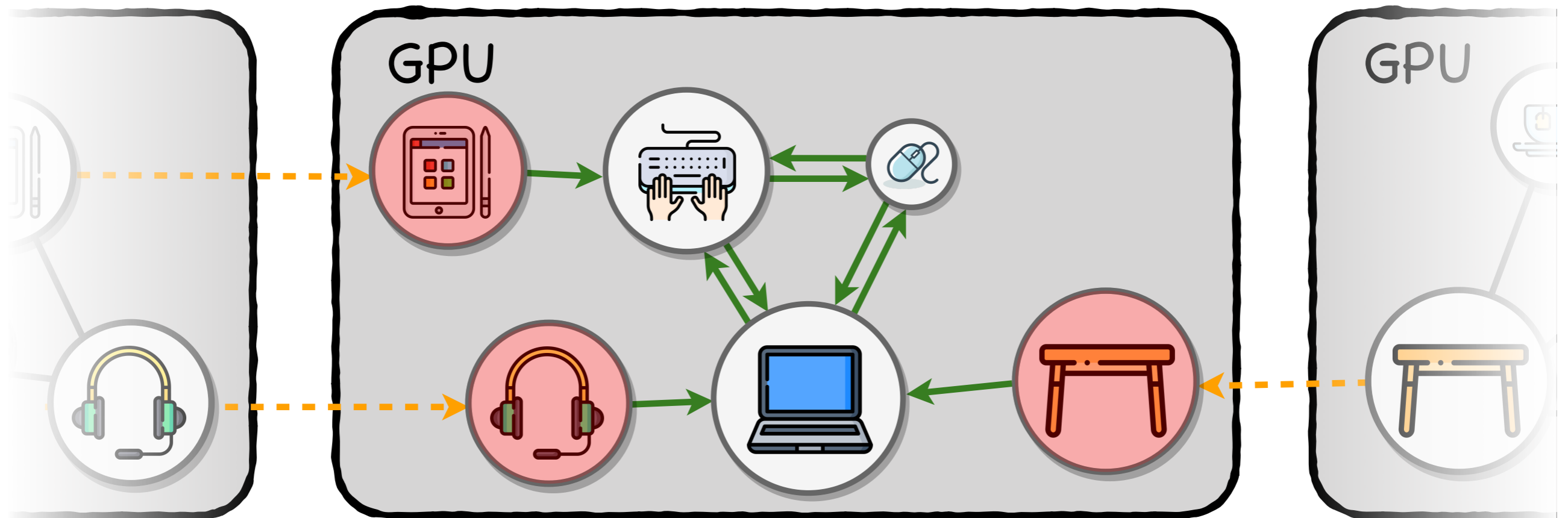


Boundary nodes
are the cause 😞

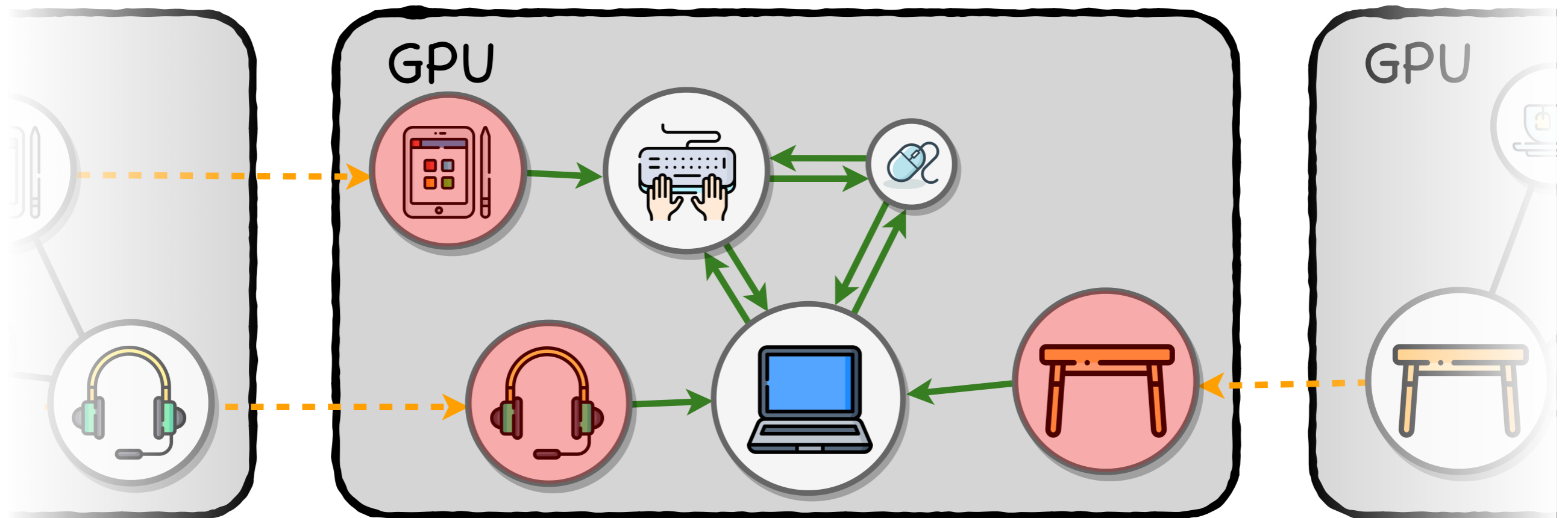
How to solve them? **One stone** three birds?



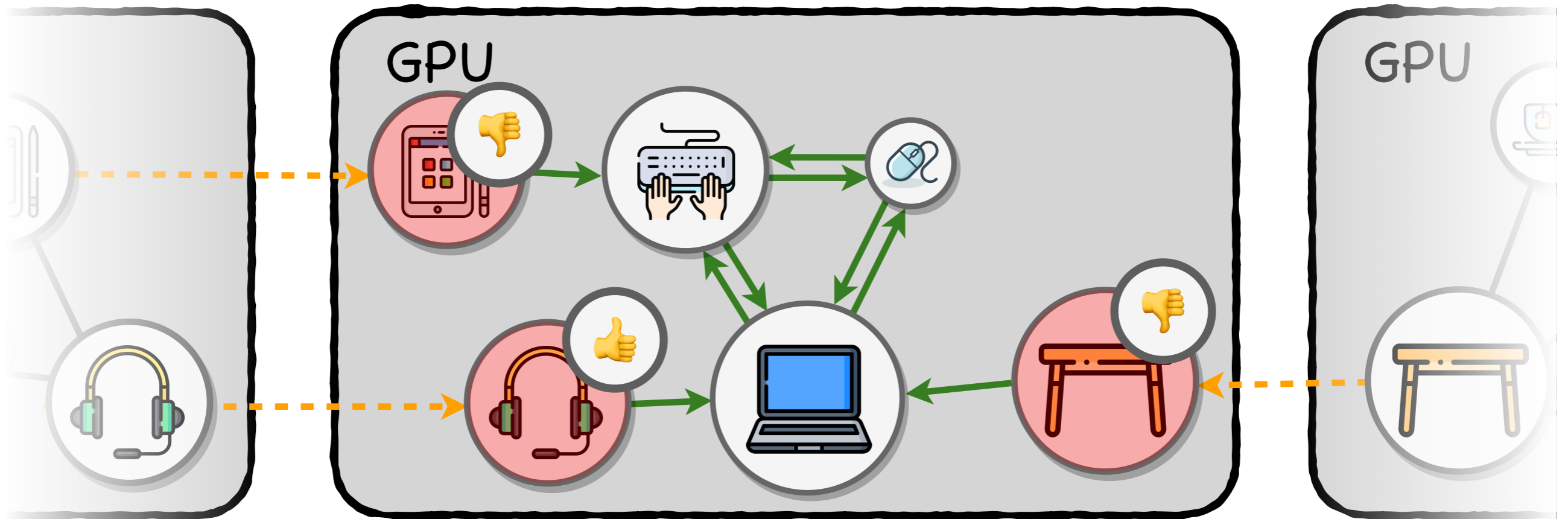
Contribution II: Propose Boundary Node Sampling



Contribution II: Propose Boundary Node Sampling

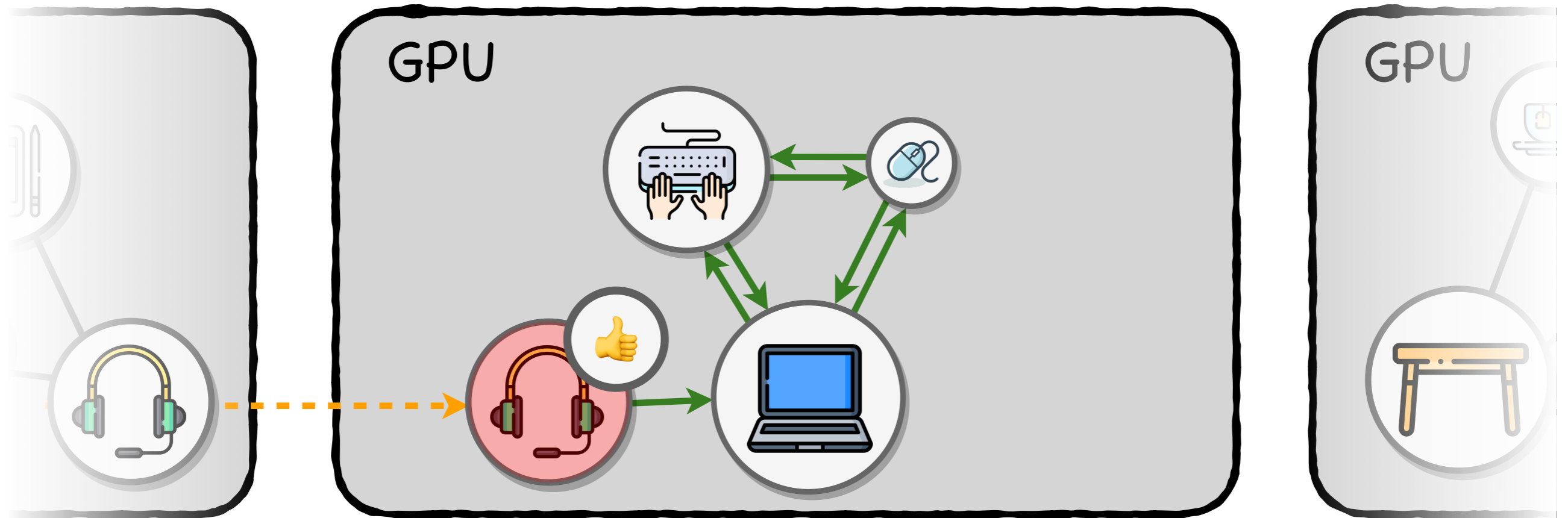


Contribution II: Propose Boundary Node Sampling



Step 1: Sampling each boundary node with probability p

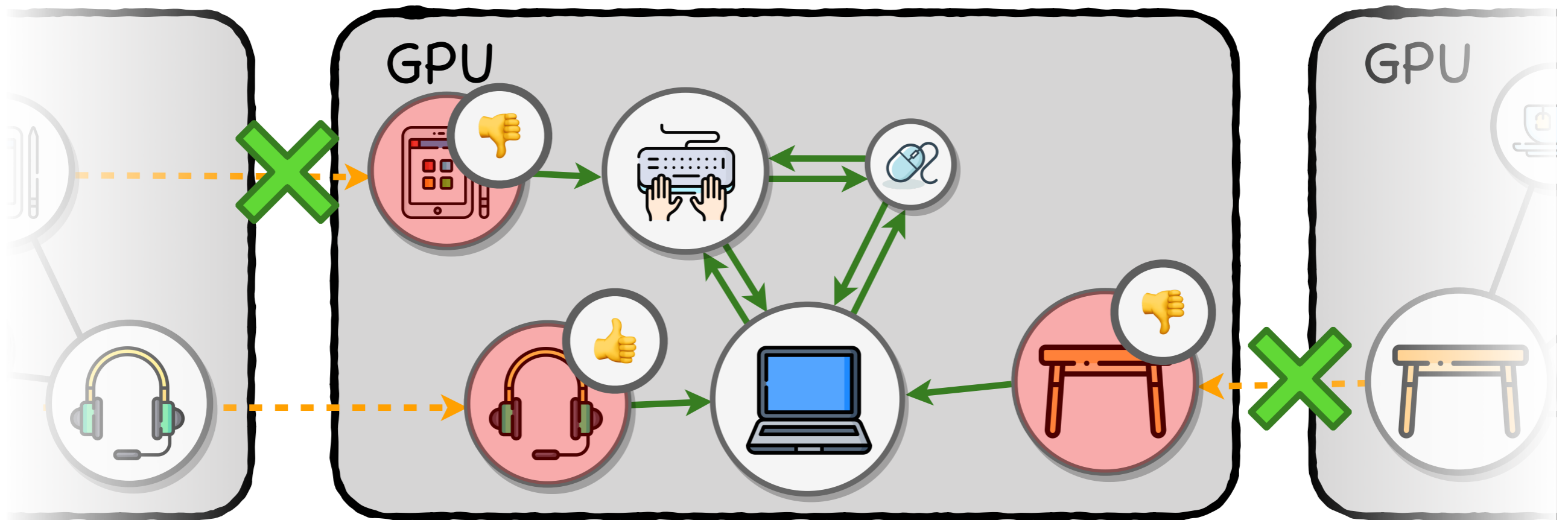
Contribution II: Propose Boundary Node Sampling



Step 1: Sampling each boundary node with probability p

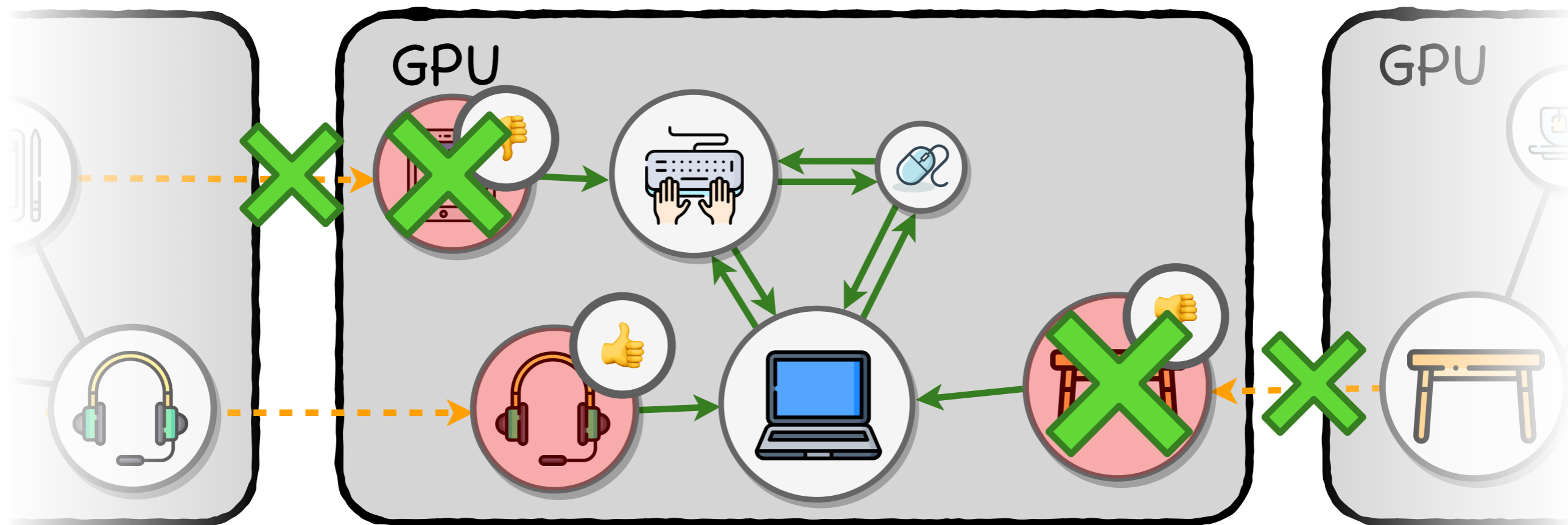
Step 2: Removing unsampled nodes

Contribution II: Propose Boundary Node Sampling



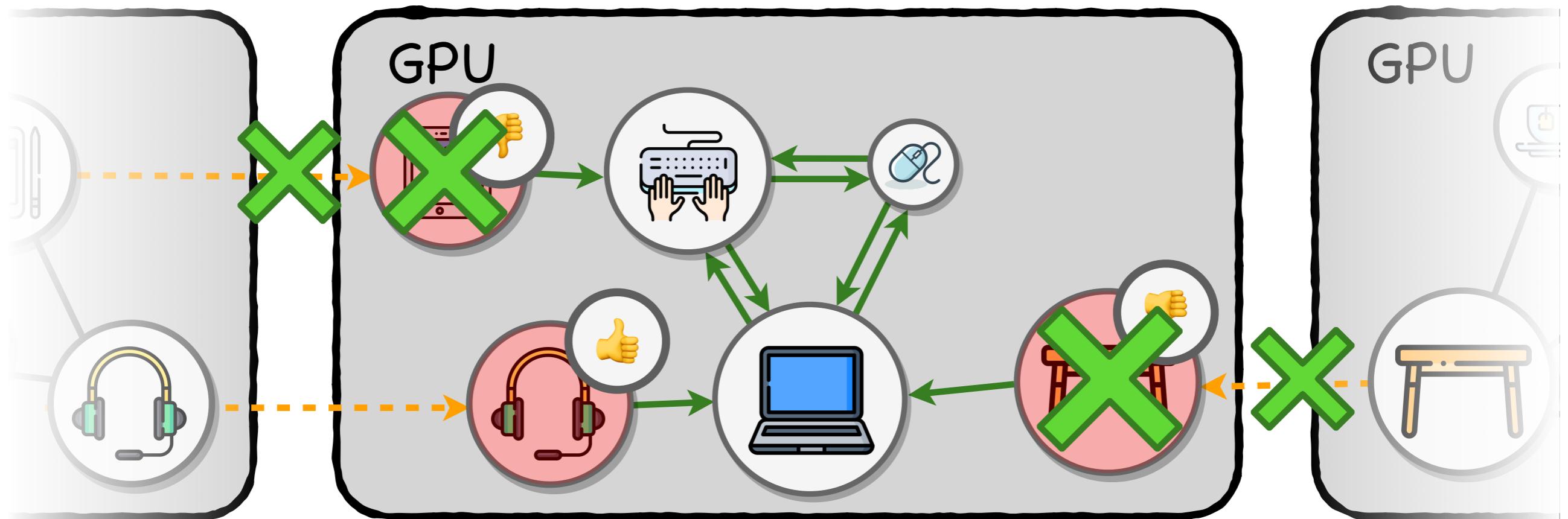
Reducing communication volume

Contribution II: Propose Boundary Node Sampling



Reducing communication volume
Reducing memory requirement

Contribution II: Propose Boundary Node Sampling



Reducing communication volume
 Reducing memory requirement
 Balancing memory across GPUs




$(n_{bd}^{(i)}$ becomes negligible)

Contribution III: Validate BNS-GCN in **Theory**

Method	Variance
---------------	-----------------




We compare the **variance** of feature approximation
(lower is better)

Contribution III: Validate BNS-GCN in **Theory**

Method	Variance
BNS-GCN	$\mathcal{O}(\mathcal{B})$  boundary neighbor set
LADIES [NeurIPS'19]	$\mathcal{O}(\mathcal{N})$  neighbor set
FastGCN [ICLR'18]	$\mathcal{O}(\mathcal{V})$  global node set

*We fix output nodes and sampling size




Contribution III: Validate BNS-GCN in **Theory**

Method	Variance
BNS-GCN	$\mathcal{O}(\mathcal{B})$  boundary neighbor set
LADIES [NeurIPS'19]	$\mathcal{O}(\mathcal{N})$  neighbor set
FastGCN [ICLR'18]	$\mathcal{O}(\mathcal{V})$  global node set

*We fix output nodes and sampling size

$$\underline{\mathcal{B}} \subseteq \underline{\mathcal{N}} \subseteq \underline{\mathcal{V}}$$

Contribution III: Validate BNS-GCN in **Theory**

Method	Variance
BNS-GCN	$\mathcal{O}(\mathcal{B})$  boundary neighbor set
LADIES [NeurIPS'19]	$\mathcal{O}(\mathcal{N})$  neighbor set
FastGCN [ICLR'18]	$\mathcal{O}(\mathcal{V})$  global node set

*We fix output nodes and sampling size

$\mathcal{B} \subseteq \mathcal{N} \subseteq \mathcal{V}$  **BNS-GCN has the best feature approximation**

Contribution III: Validate BNS-GCN in **Theory**

Method	Variance
BNS-GCN	$\mathcal{O}(\mathcal{B} \gamma^2)$
LADIES [NeurIPS'19]	$\mathcal{O}(\mathcal{N} \gamma^2)$
FastGCN [ICLR'18]	$\mathcal{O}(\mathcal{V} \gamma^2)$
VR-GCN [ICML'18]	$\mathcal{O}(D\Delta\gamma^2)$
GraphSAGE [NIPS'17]	$\mathcal{O}(D\gamma^2)$

*We fix output nodes and sampling size

More analysis is in our paper

Experiments

Experiment Setup

Considered Datasets

Reddit, ogbn-products, Yelp and ogbn-papers100M

Dataset Description

Name	# Nodes	# Edges	Environment
Reddit	233K	114M	10 RTX-2080Ti (11GB)
ogbn-products	2.4M	62M	
Yelp	716K	7.0M	
ogbn-papers100M	111M	1.6B	32 x (6 Tesla V100 (16GB))

Experiment Setup

Considered Datasets

Reddit, ogbn-products, Yelp and ogbn-papers100M

Benchmarked Baselines

ROC [MLSys'20] (swap-based) and CAGNET [SC'20] (slice-based)

Experiment Setup

Considered Datasets

Reddit, ogbn-products, Yelp and ogbn-papers100M

Benchmarked Baselines

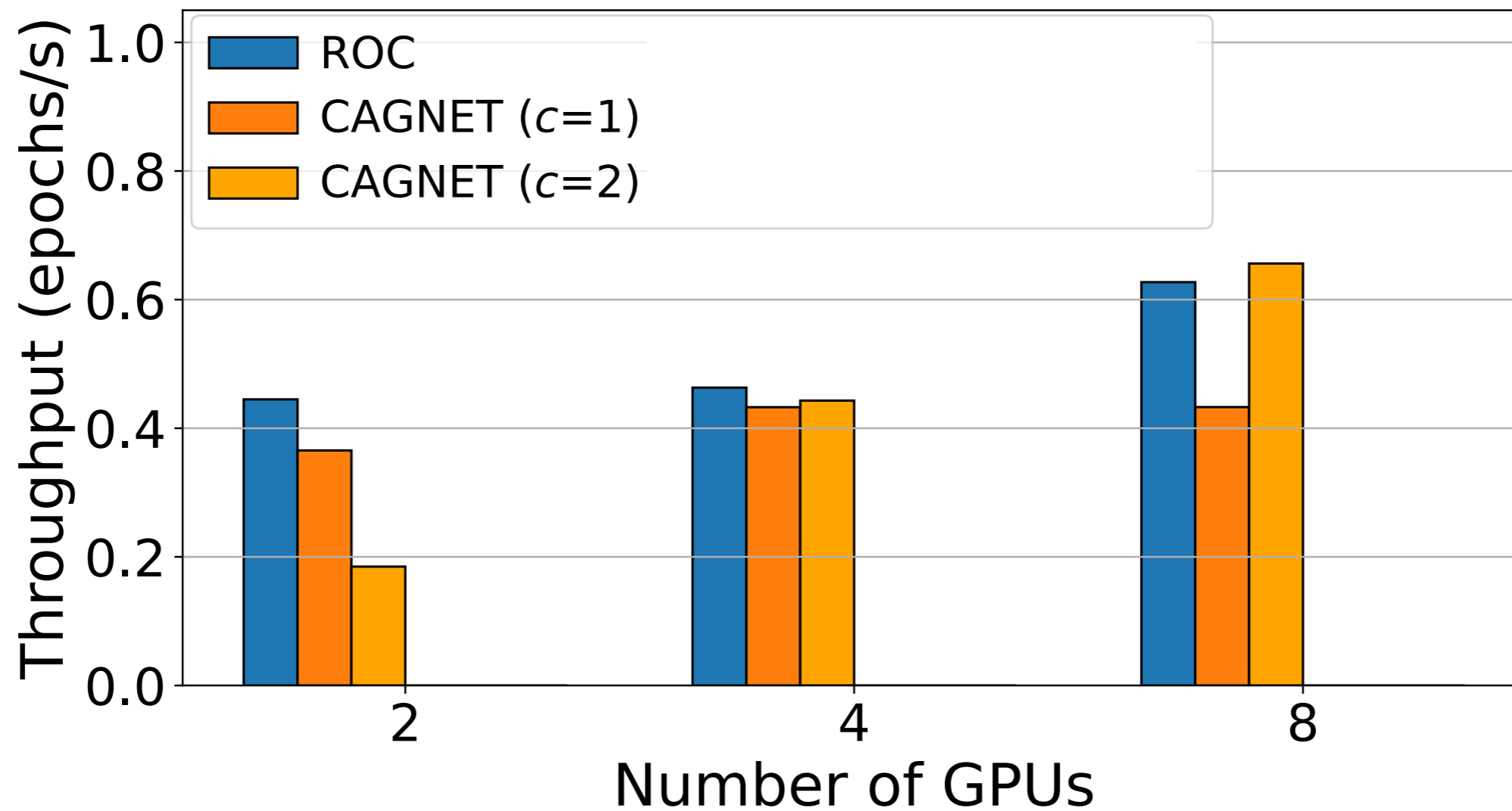
ROC [MLSys'20] (swap-based) and CAGNET [SC'20] (slice-based)

Adopted Toolkits

DGL 0.7.0 and PyTorch 1.9.1

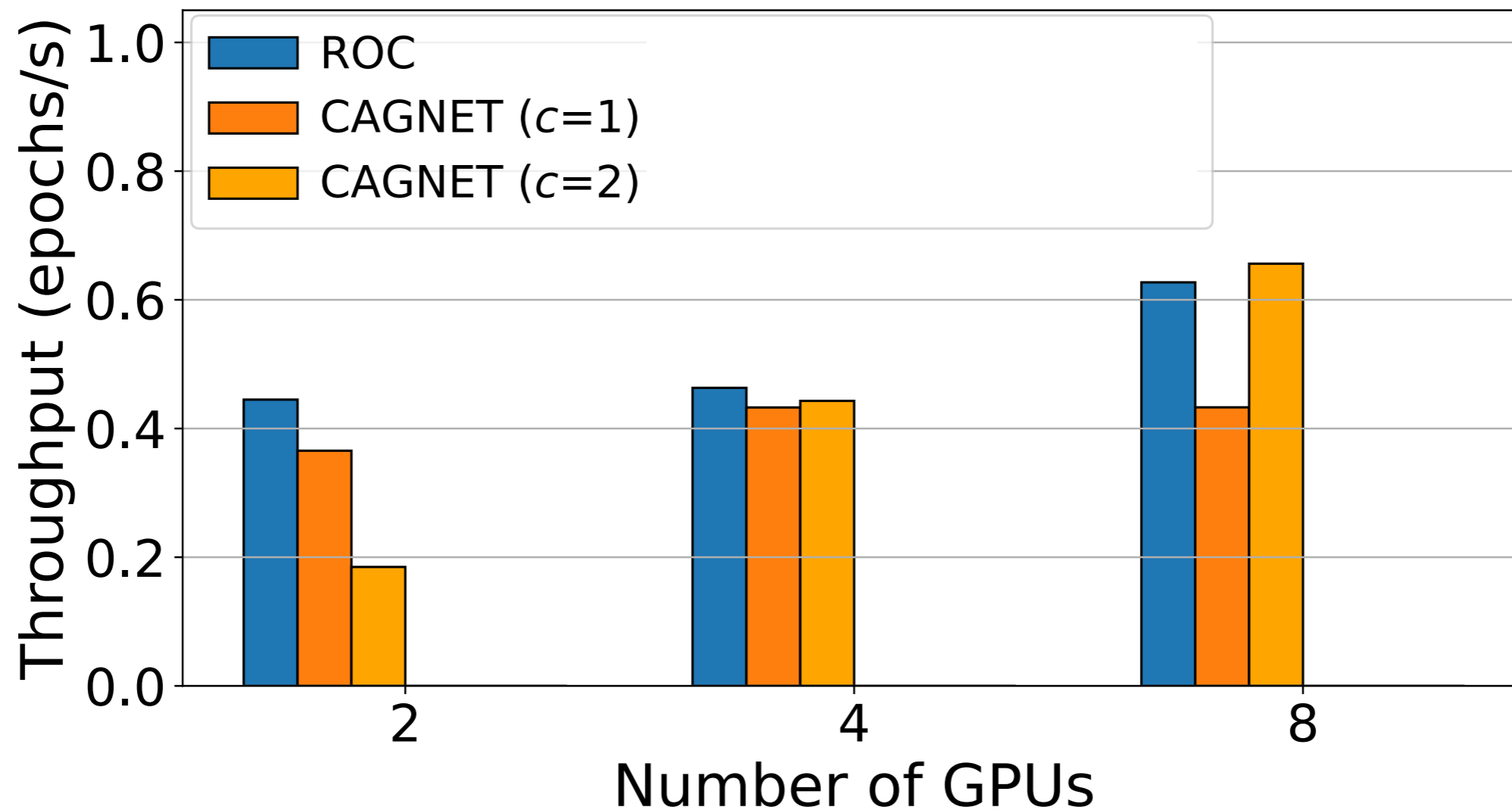
Training Throughput Comparison

Reddit Dataset



Training Throughput Comparison

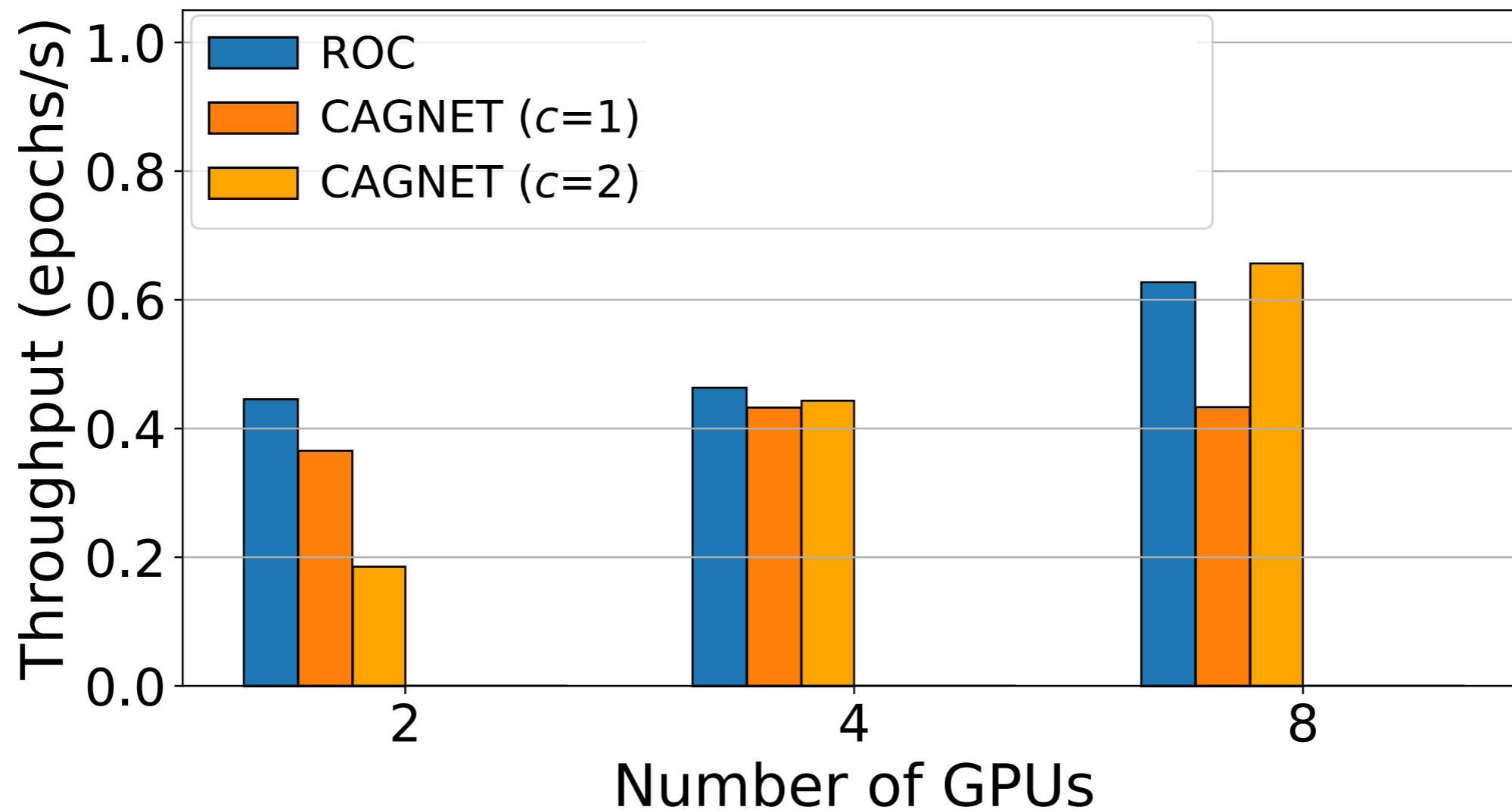
Reddit Dataset



Baselines: throughput <0.7 epochs/s

Training Throughput Comparison

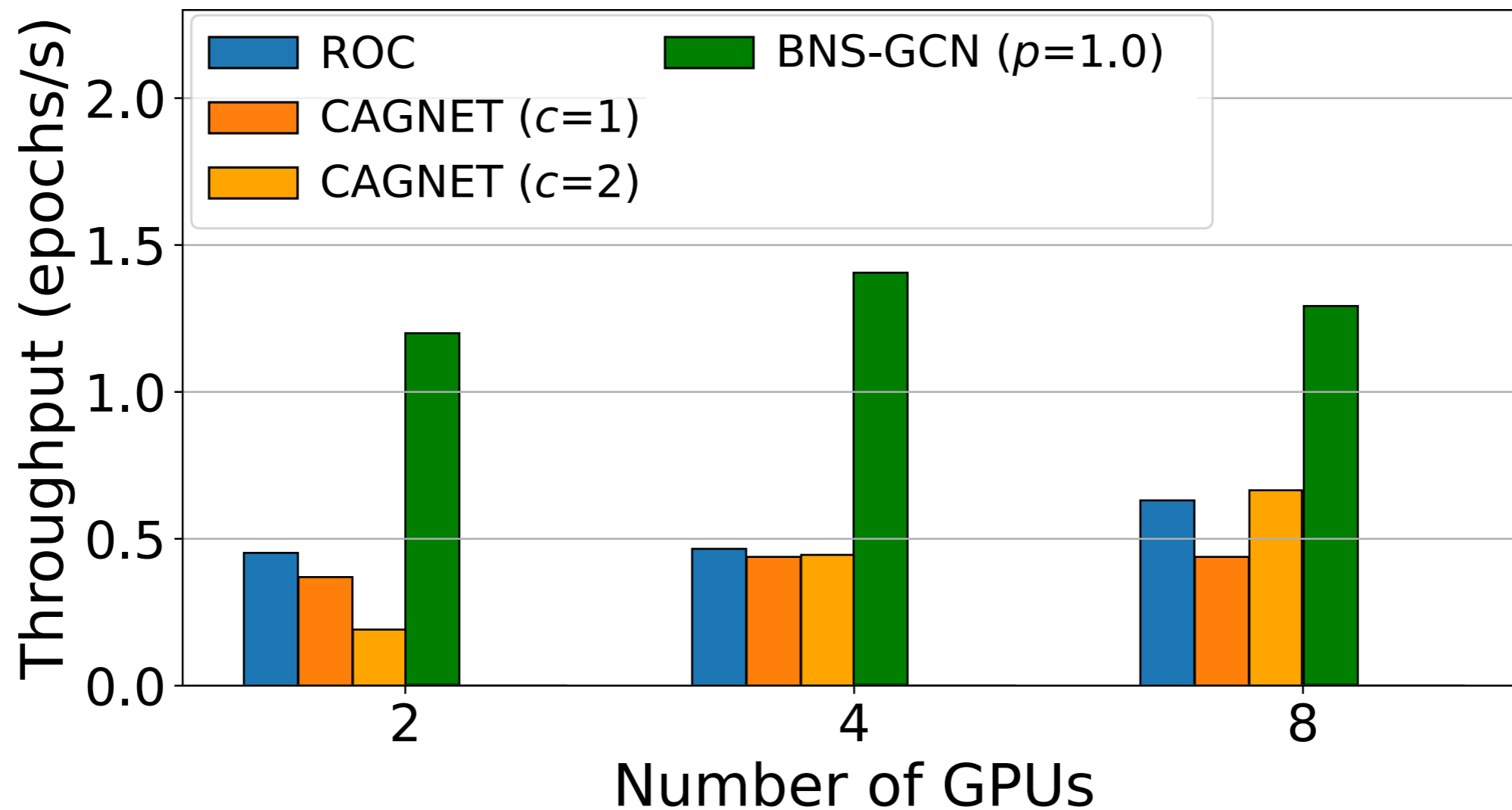
Reddit Dataset



Baselines: throughput < 0.7 epochs/s

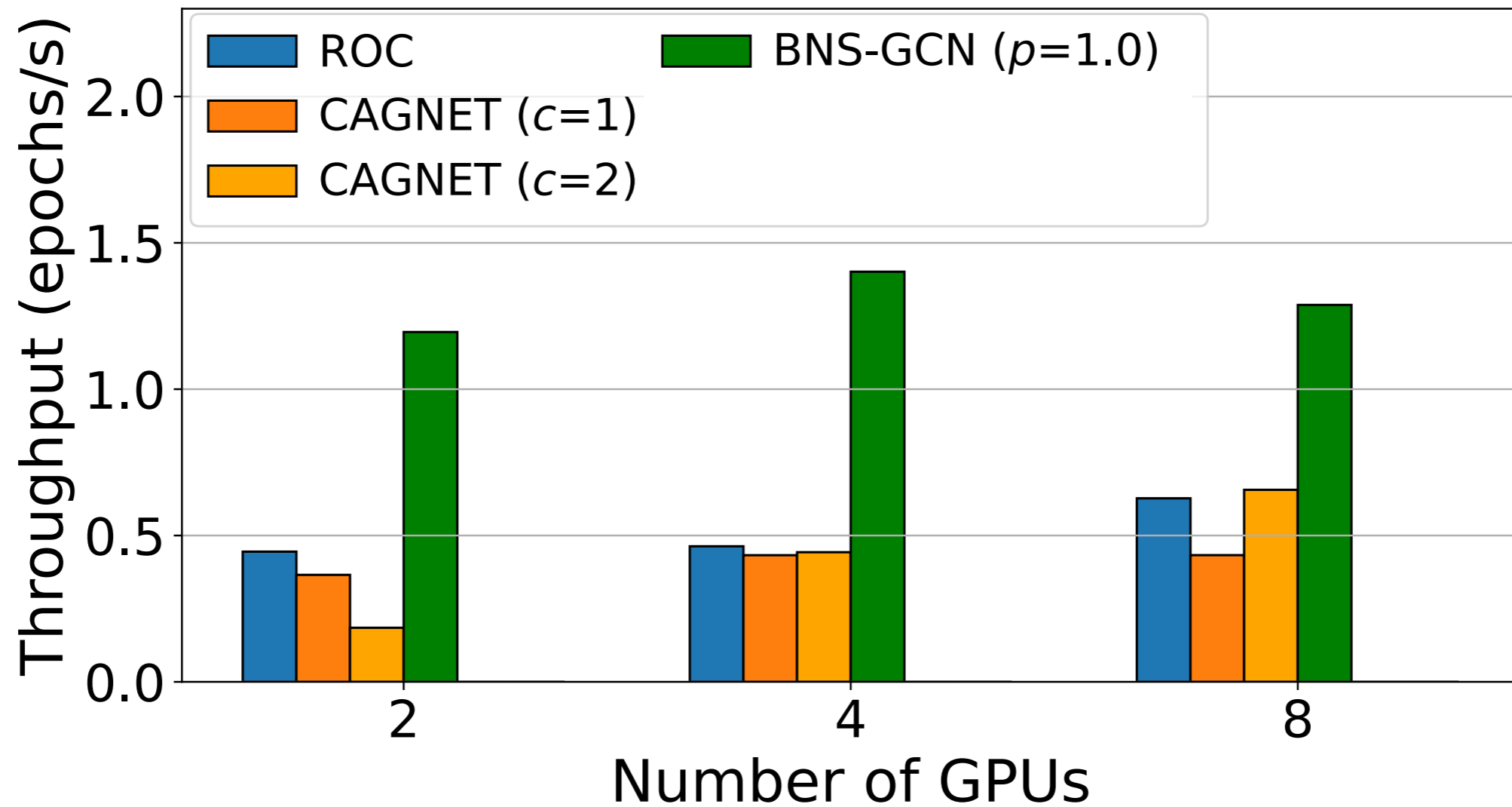
Training Throughput Comparison

Reddit Dataset



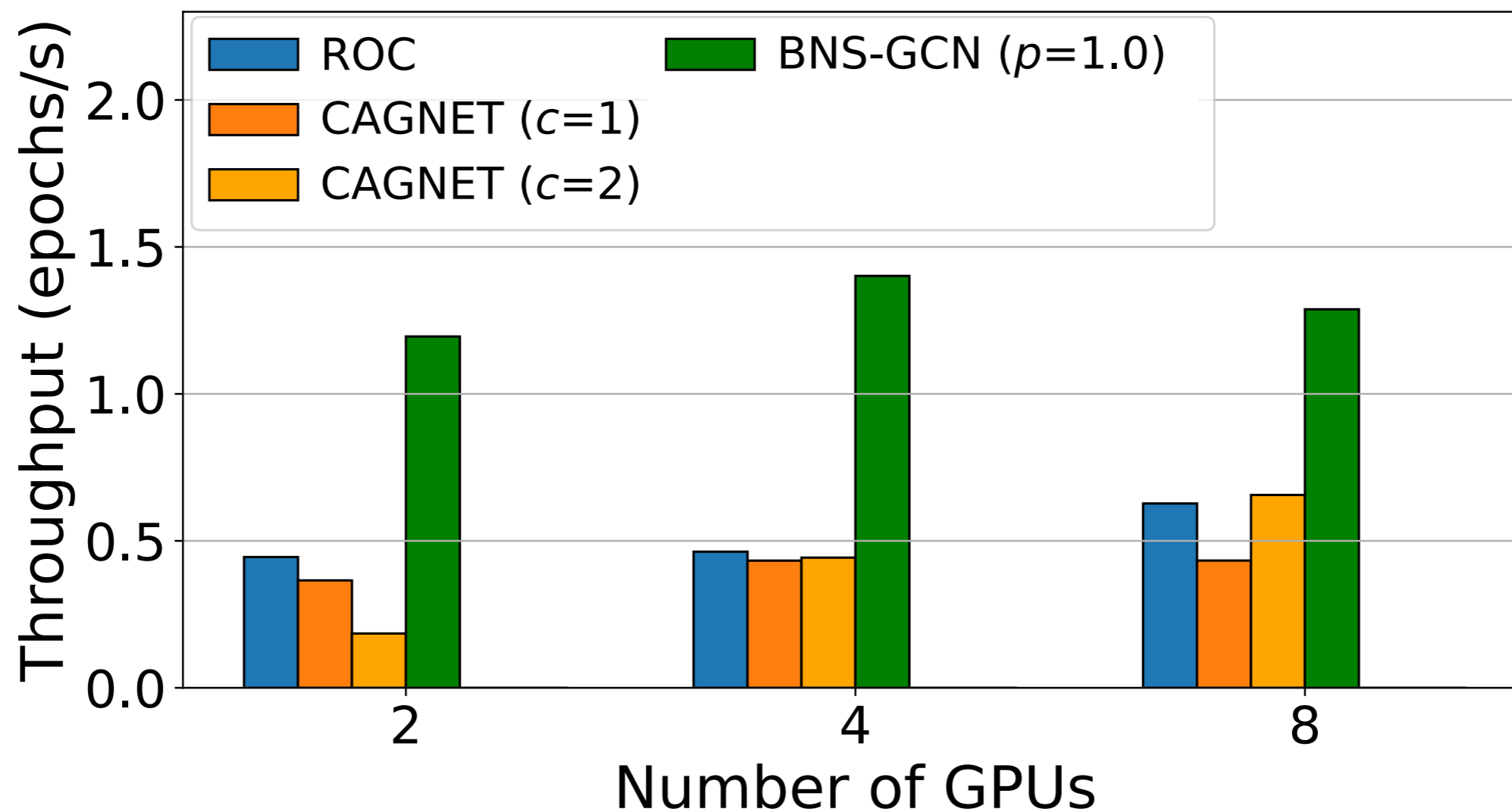
Training Throughput Comparison

Reddit Dataset



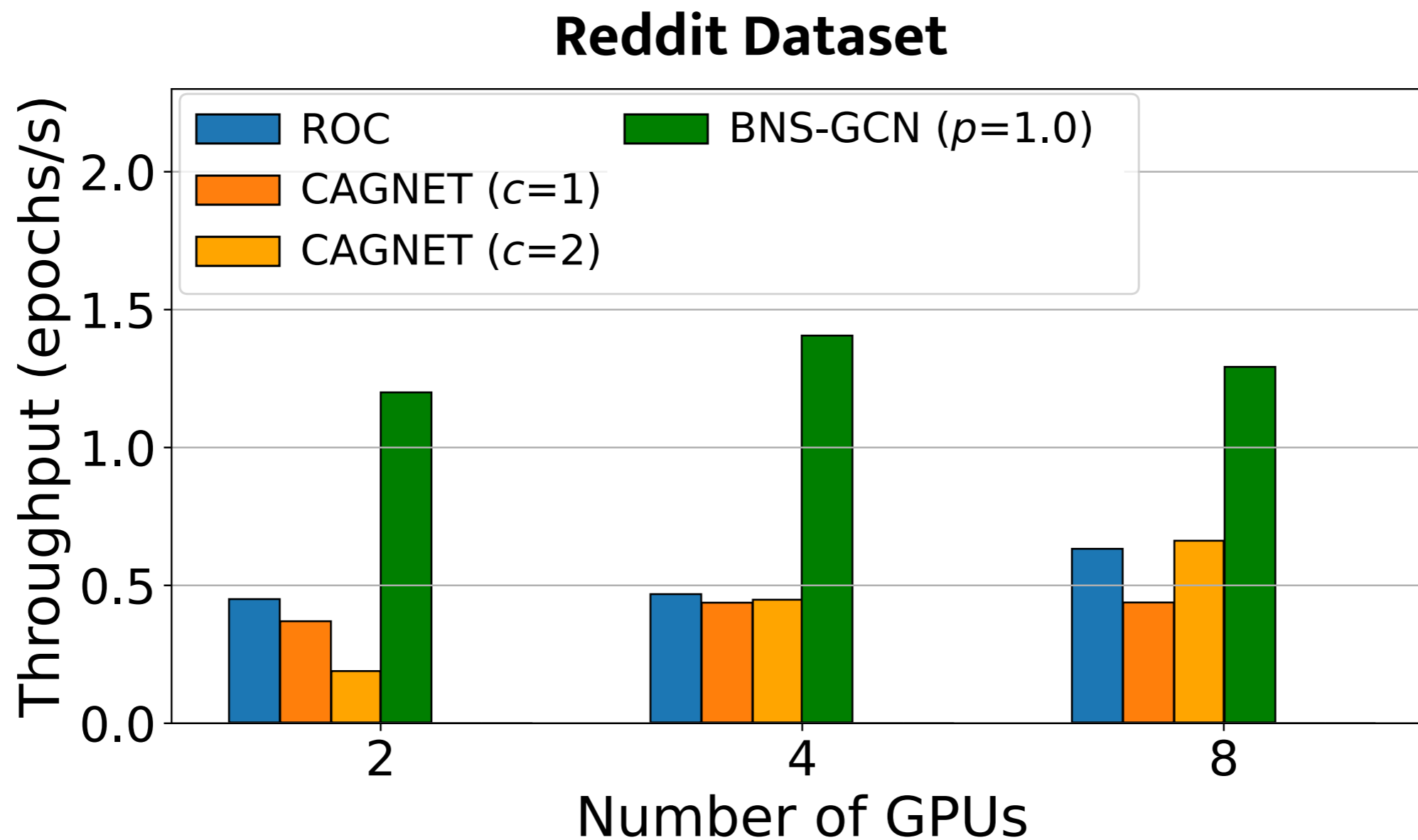
Training Throughput Comparison

Reddit Dataset



Partition-based training: throughput >1.2 epochs/s

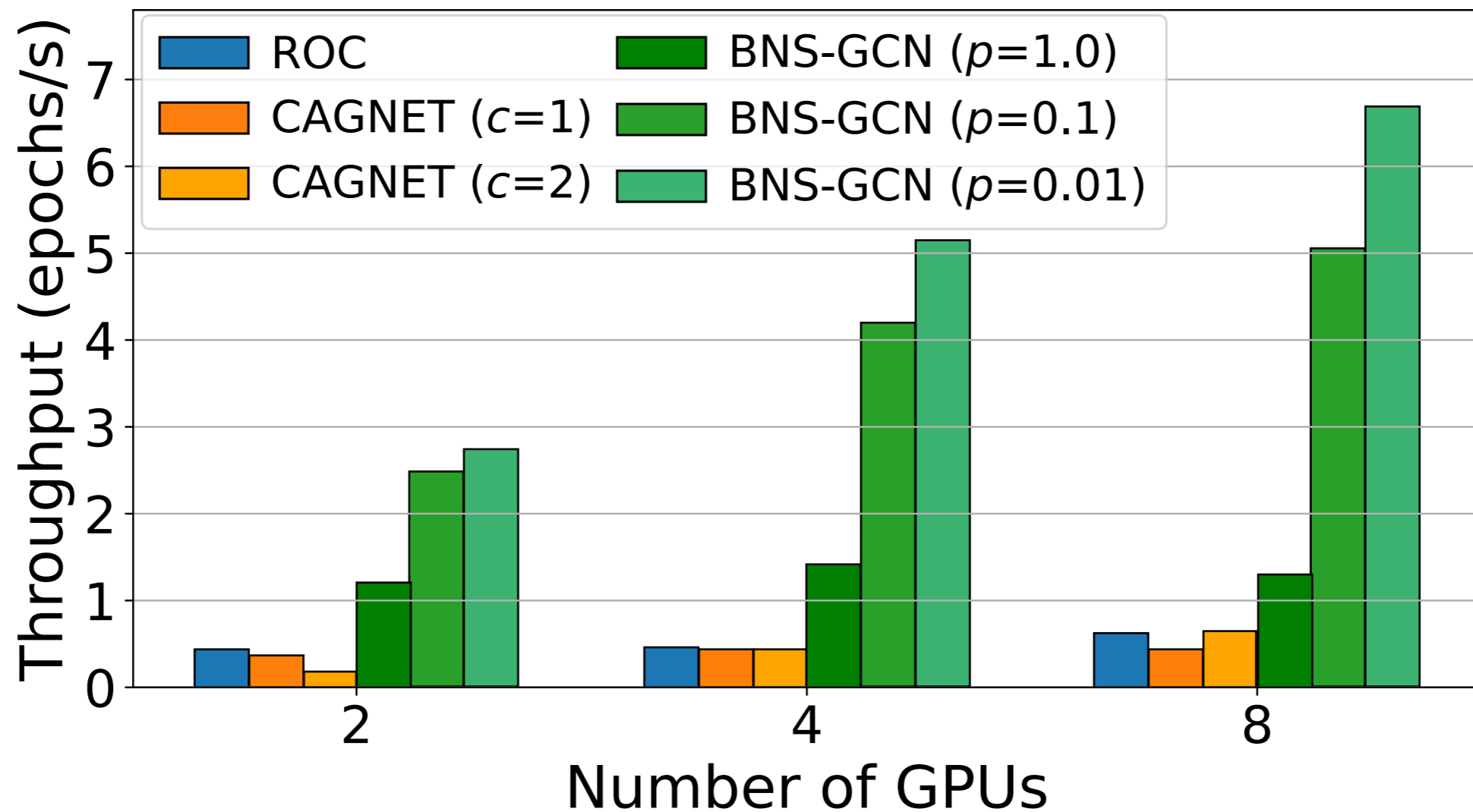
Training Throughput Comparison



Partition-based training: throughput >1.2 epochs/s

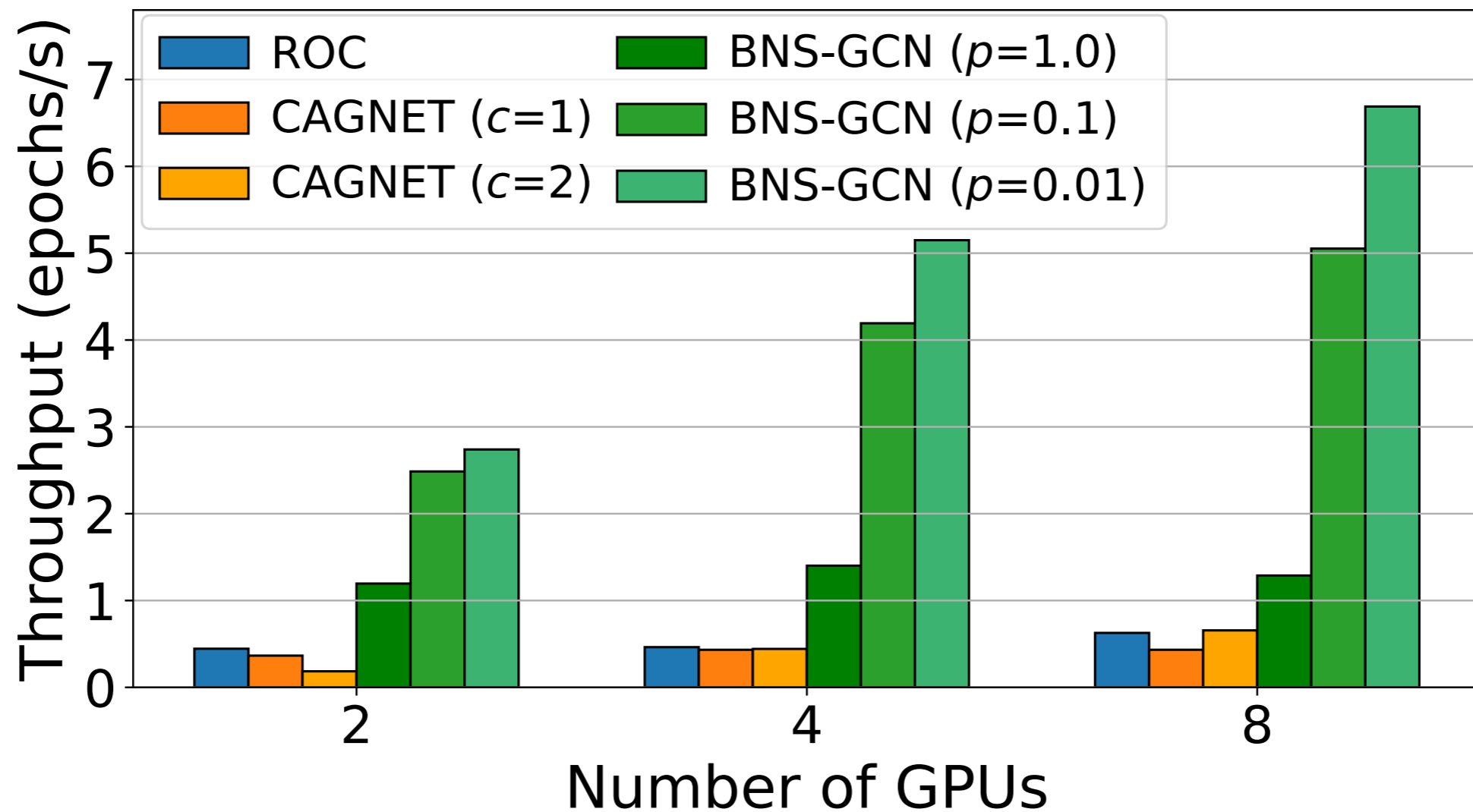
Training Throughput Comparison

Reddit Dataset



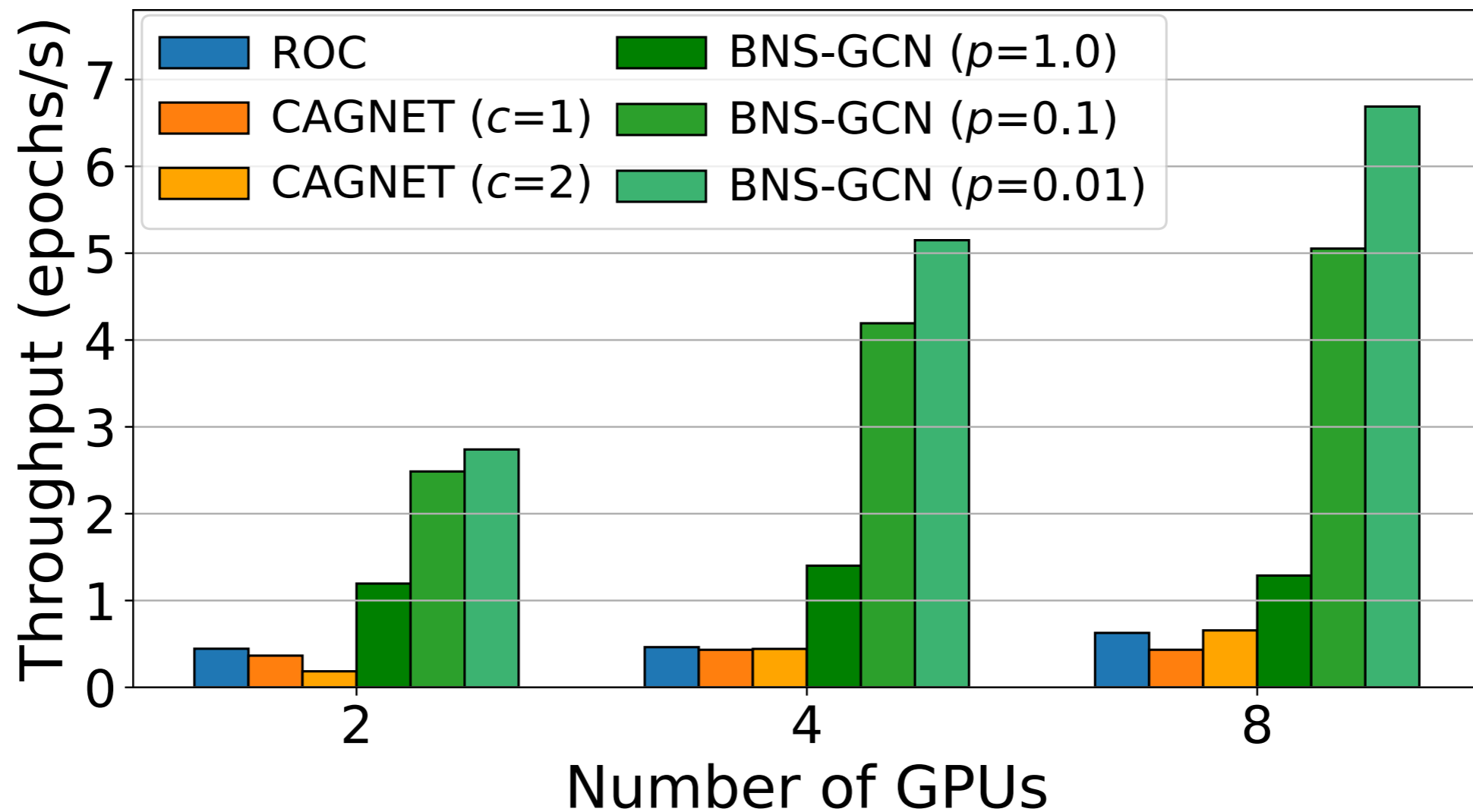
Training Throughput Comparison

Reddit Dataset



Training Throughput Comparison

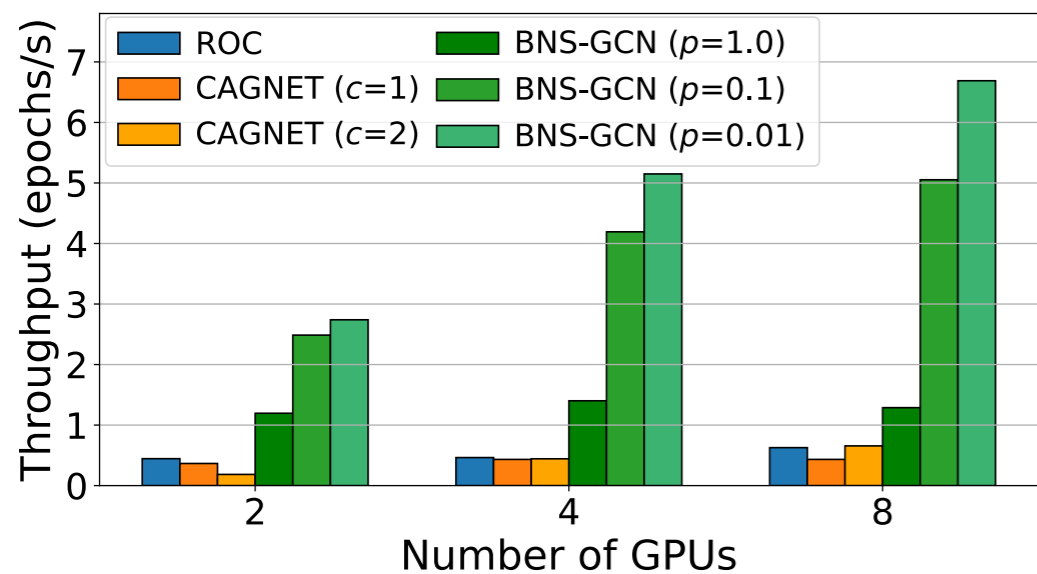
Reddit Dataset



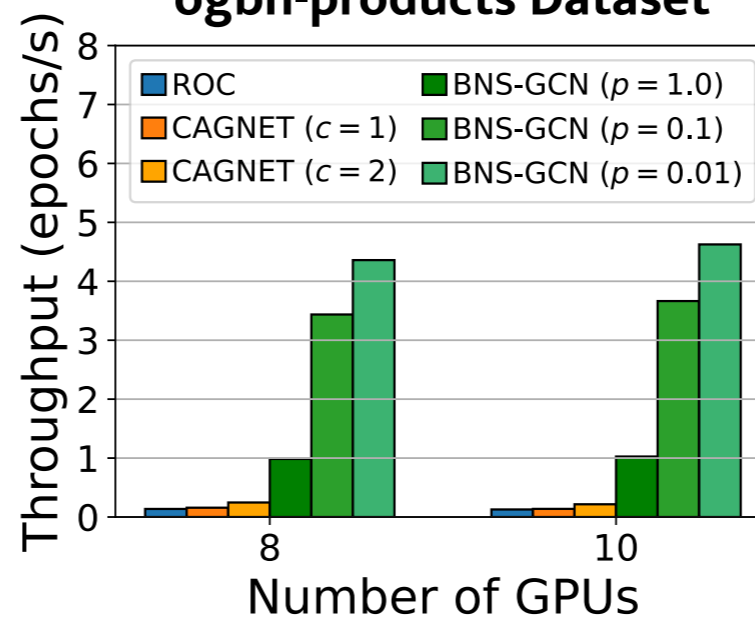
BNS-GCN: 8.9x~16.2x throughput improvement

Training Throughput Comparison

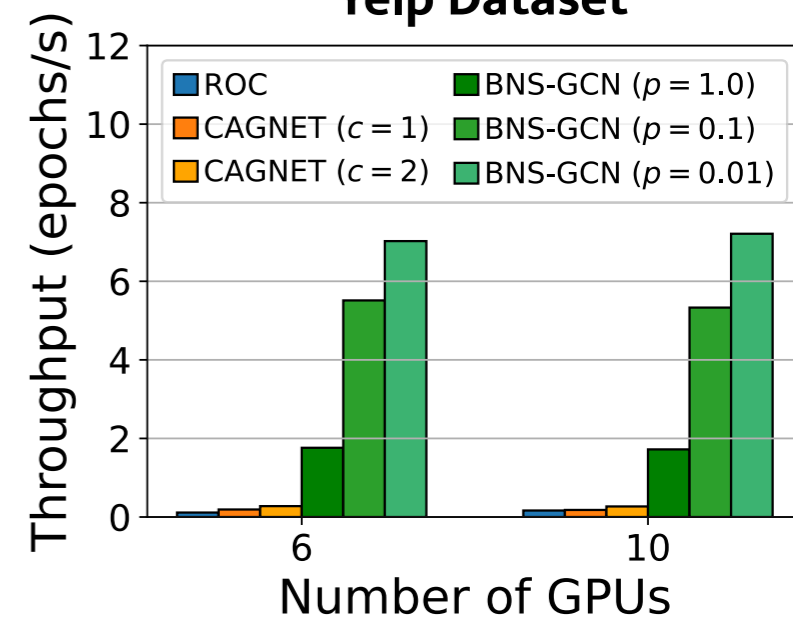
Reddit Dataset



ogbn-products Dataset

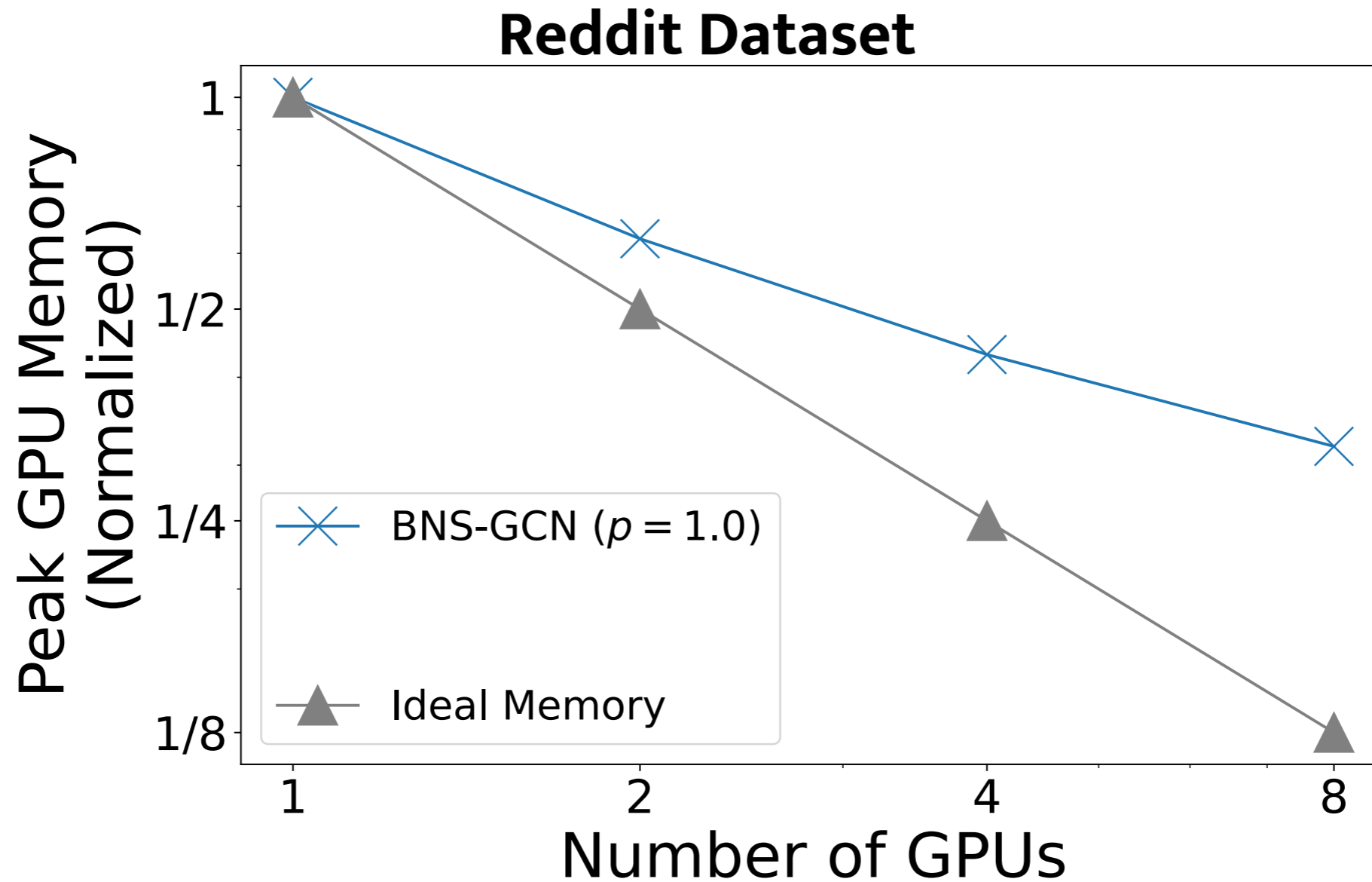


Yelp Dataset

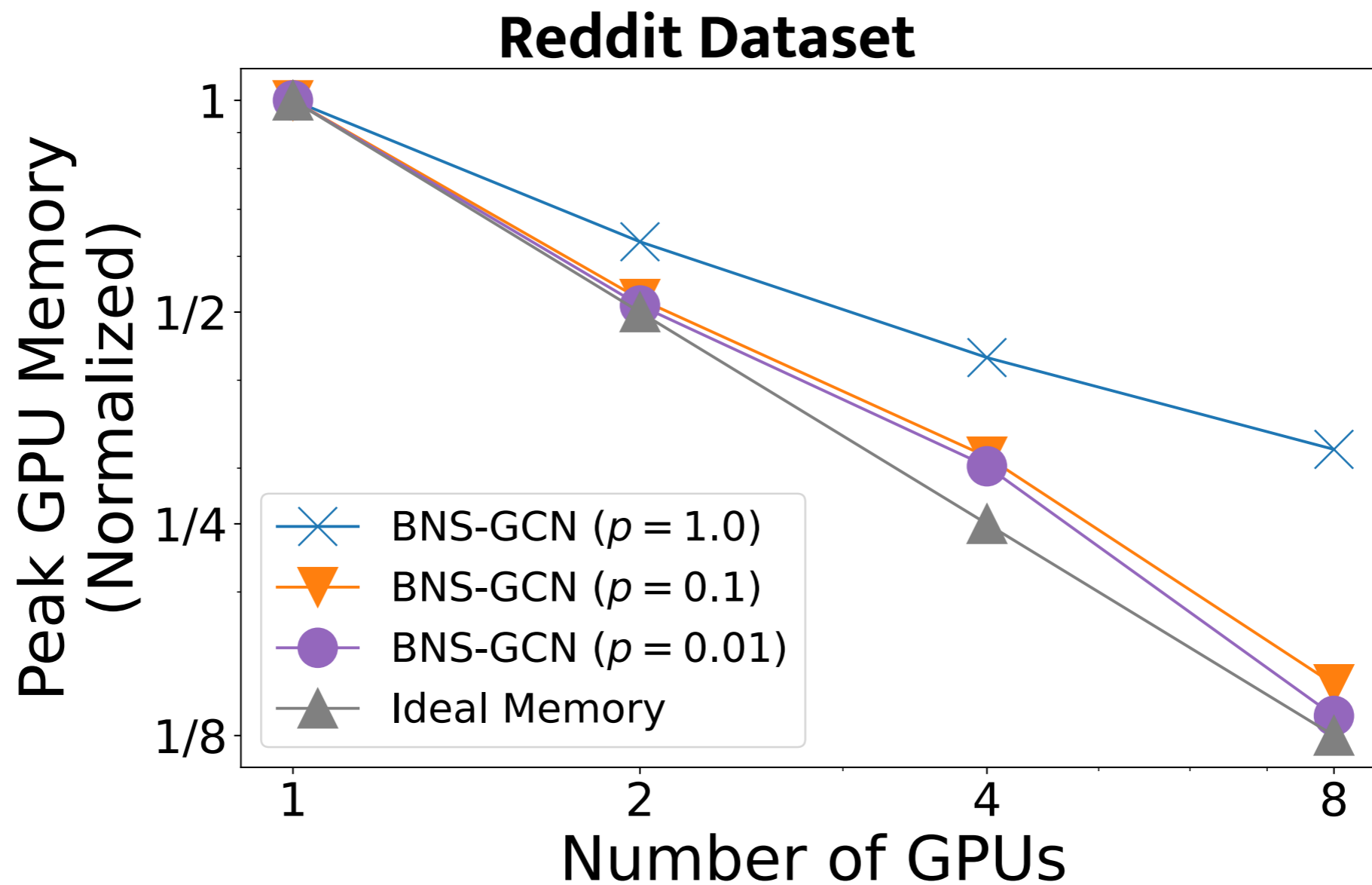


BNS-GCN is consistently faster

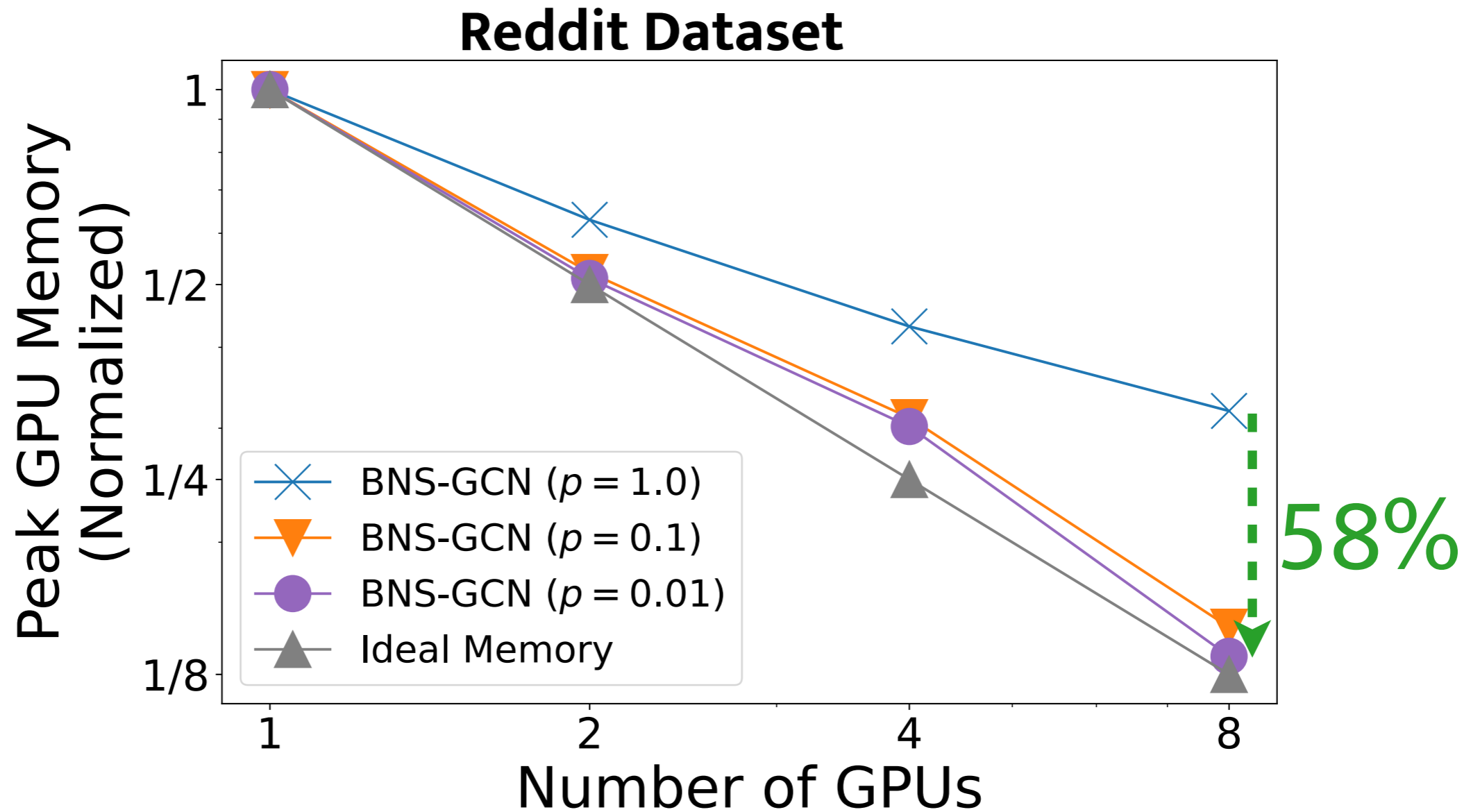
Memory Saving



Memory Saving

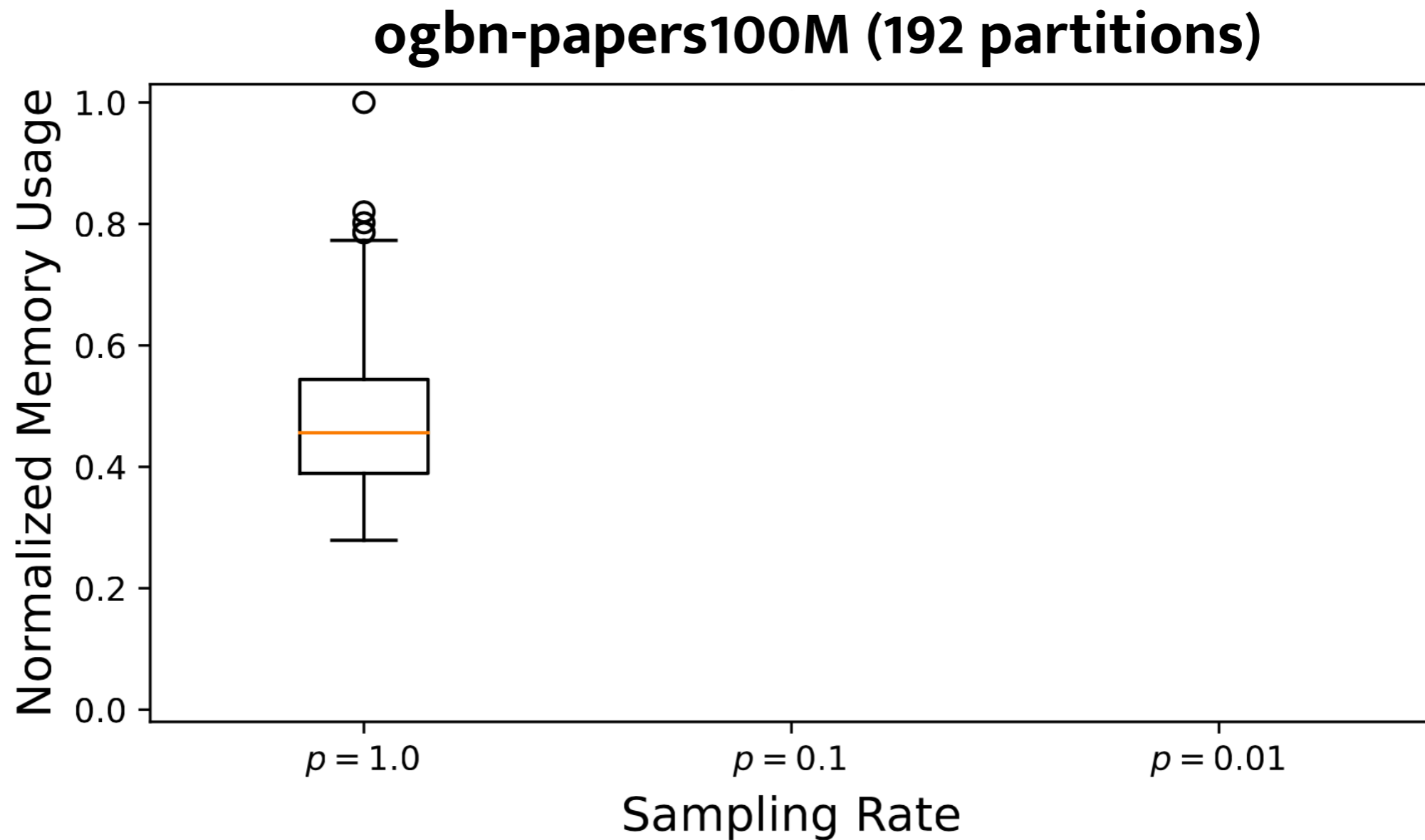


Memory Saving

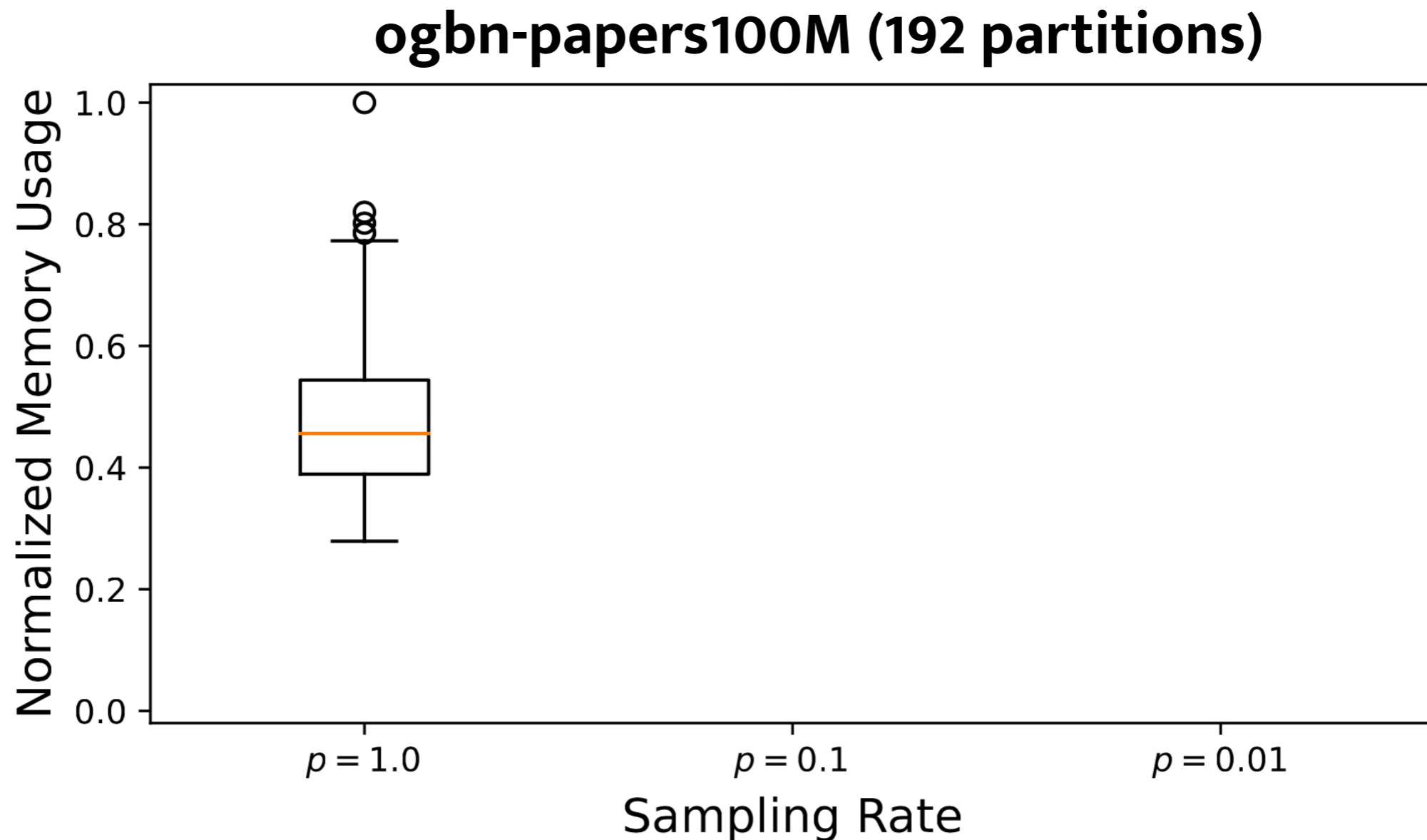


BNS-GCN saves the memory by up to 58%

Balancing Memory Requirement

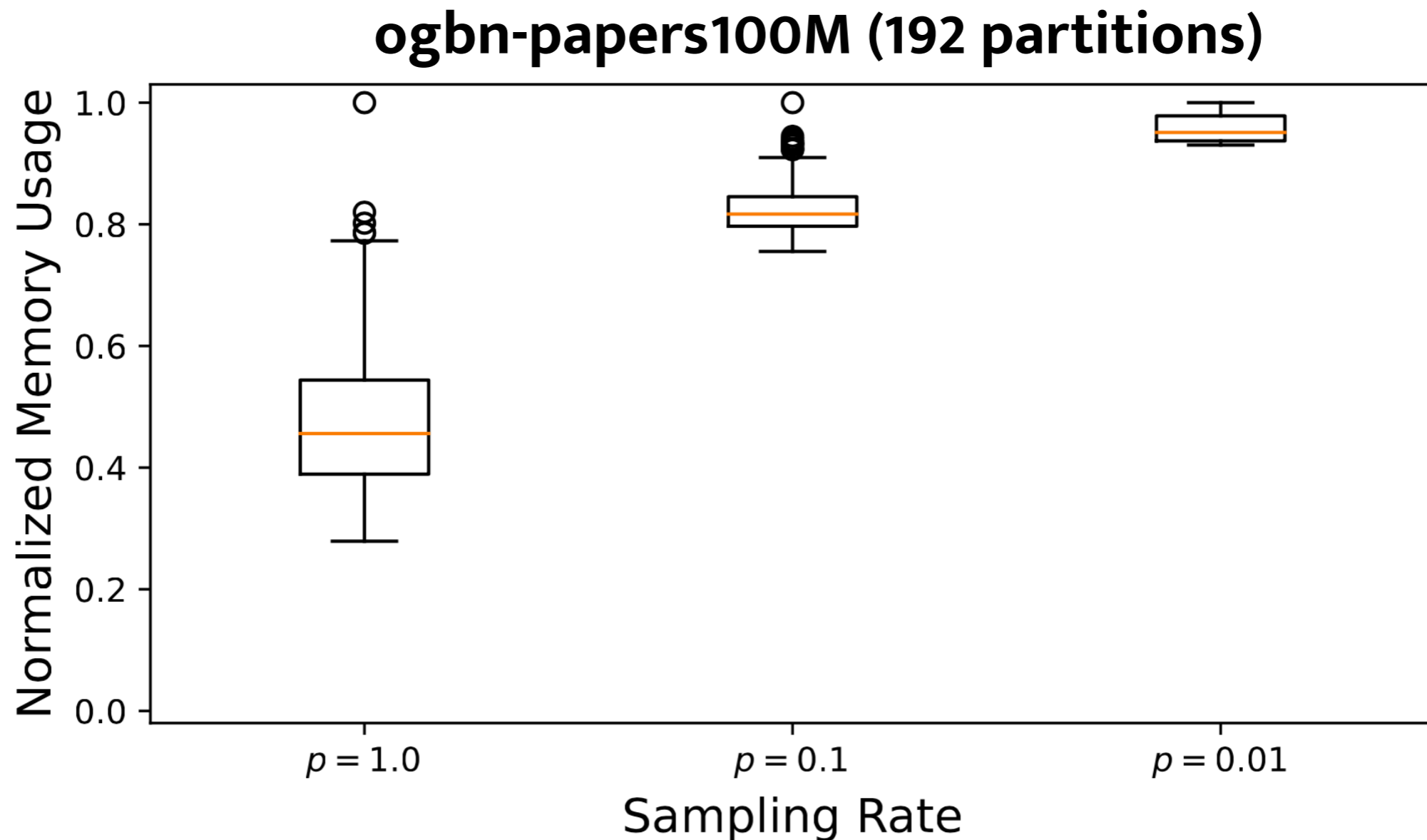


Balancing Memory Requirement



Without BNS: >75% partitions utilize <60% memory

Balancing Memory Requirement



Without BNS: >75% partitions utilize <60% memory

With BNS: nearly all partitions utilize >80% memory

Training Accuracy Comparison

Dataset	Reddit			ogbn-products			Yelp		
# Partitions	2	4	8	5	8	10	3	6	10

Training Accuracy Comparison

Dataset	Reddit			ogbn-products			Yelp		
# Partitions	2	4	8	5	8	10	3	6	10
BNS-GCN (p=1.0)	97.11	97.11	97.11	79.14	79.14	79.14	65.26	65.26	65.26

BNS-GCN (p=1.0) is equivalent to vanilla training

Training Accuracy Comparison

Dataset	Reddit			ogbn-products			Yelp		
# Partitions	2	4	8	5	8	10	3	6	10
BNS-GCN (p=1.0)	97.11	97.11	97.11	79.14	79.14	79.14	65.26	65.26	65.26
BNS-GCN (p=0.1)	97.15	97.14	97.18	79.36	79.48	79.30	65.32	65.26	65.34
BNS-GCN (p=0.01)	97.09	97.03	96.91	79.43	79.28	79.21	65.27	65.31	65.29

BNS-GCN (p=1.0) is equivalent to vanilla training
 Sampling boundary nodes **maintains** the accuracy

Training Accuracy Comparison

Dataset	Reddit			ogbn-products			Yelp		
# Partitions	2	4	8	5	8	10	3	6	10
BNS-GCN (p=1.0)	97.11	97.11	97.11	79.14	79.14	79.14	65.26	65.26	65.26
BNS-GCN (p=0.1)	97.15	97.14	97.18	79.36	79.48	79.30	65.32	65.26	65.34
BNS-GCN (p=0.01)	97.09	97.03	96.91	79.43	79.28	79.21	65.27	65.31	65.29
BNS-GCN (p=0.0)	97.03	96.87	96.81	78.65	78.83	78.79	65.28	65.27	65.23

BNS-GCN (p=1.0) is equivalent to vanilla training

Sampling boundary nodes **maintains** the accuracy

Dropping boundary nodes **decreases** the accuracy

Training Accuracy Comparison

Dataset	Reddit			ogbn-products			Yelp		
# Partitions	2	4	8	5	8	10	3	6	10
BNS-GCN (p=1.0)	97.11	97.11	97.11	79.14	79.14	79.14	65.26	65.26	65.26
BNS-GCN (p=0.1)	97.15	97.14	97.18	79.36	79.48	79.30	65.32	65.26	65.34
BNS-GCN (p=0.01)	97.09	97.03	96.91	79.43	79.28	79.21	65.27	65.31	65.29
BNS-GCN (p=0.0)	97.03	96.87	96.81	78.65	78.83	78.79	65.28	65.27	65.23
FastGCN [ICLR'18]		93.7			60.42			26.5	
GraphSAGE [NIPS'17]		95.4			78.70			63.4	
AS-GCN [NIPS'18]		96.3			OOM			OOM	
LADIES [NeurIPS'19]		94.3			77.46			60.2	
VR-GCN [ICML'18]		96.3			OOM			64.0	
ClusterGCN [KDD'19]		96.6			78.97			60.9	
GraphSAINT [ICLR'20]		96.6			79.08			65.3	

Full-graph training reaches **higher** accuracy than sampling-based methods

Conclusion

Conclusion

- ◆ Identified **three key drawbacks** in partition-based GCN training
 - Underlying cause: **boundary nodes**

Conclusion

- ◆ Identified **three key drawbacks** in partition-based GCN training
 - Underlying cause: **boundary nodes**
- ◆ Proposed **Boundary Node Sampling (BNS-GCN)** to tackle the three drawbacks

Conclusion

- ◆ Identified **three key drawbacks** in partition-based GCN training
 - Underlying cause: **boundary nodes**
- ◆ Proposed **Boundary Node Sampling (BNS-GCN)** to tackle the three drawbacks
- ◆ Validated BNS-GCN in both **theory** and **experiments**

Conclusion

- ◆ Identified **three key drawbacks** in partition-based GCN training
 - Underlying cause: **boundary nodes**
- ◆ Proposed **Boundary Node Sampling (BNS-GCN)** to tackle the three drawbacks
- ◆ Validated BNS-GCN in both **theory** and **experiments**

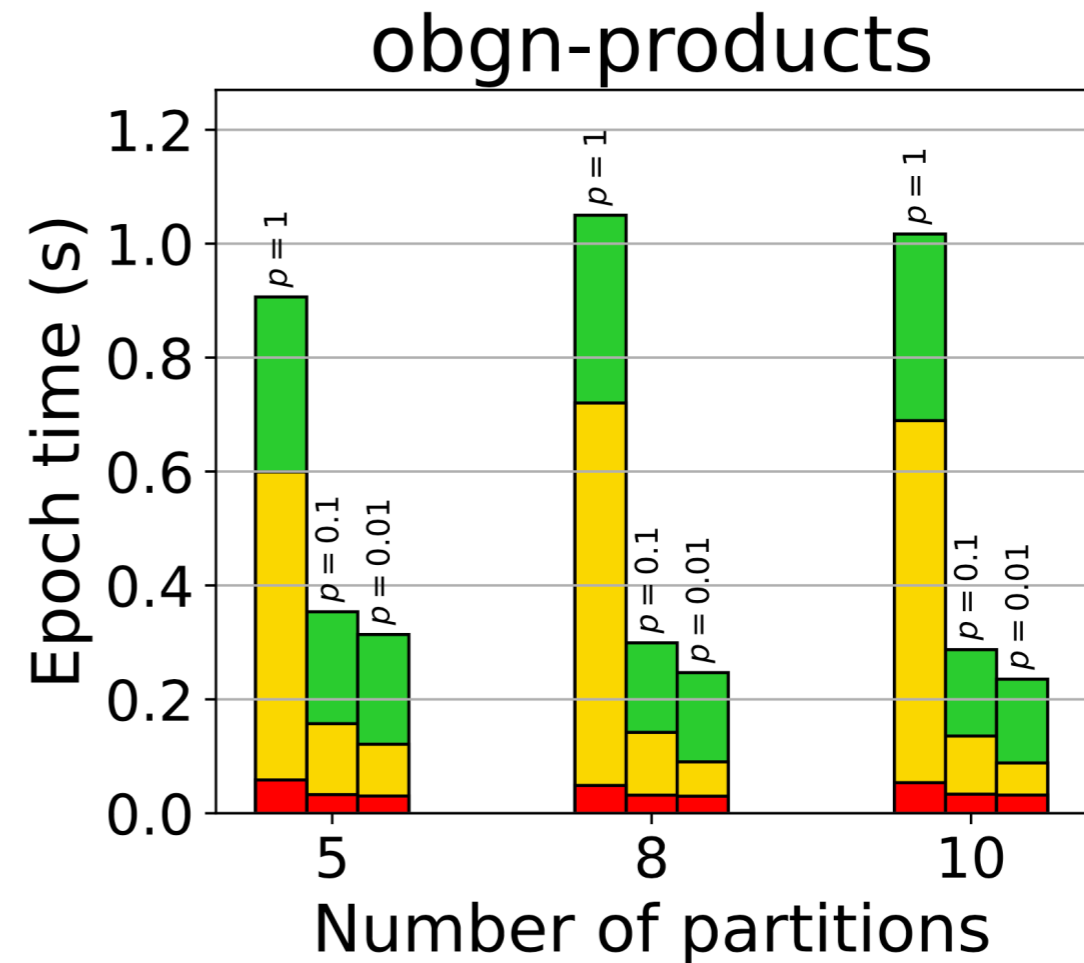
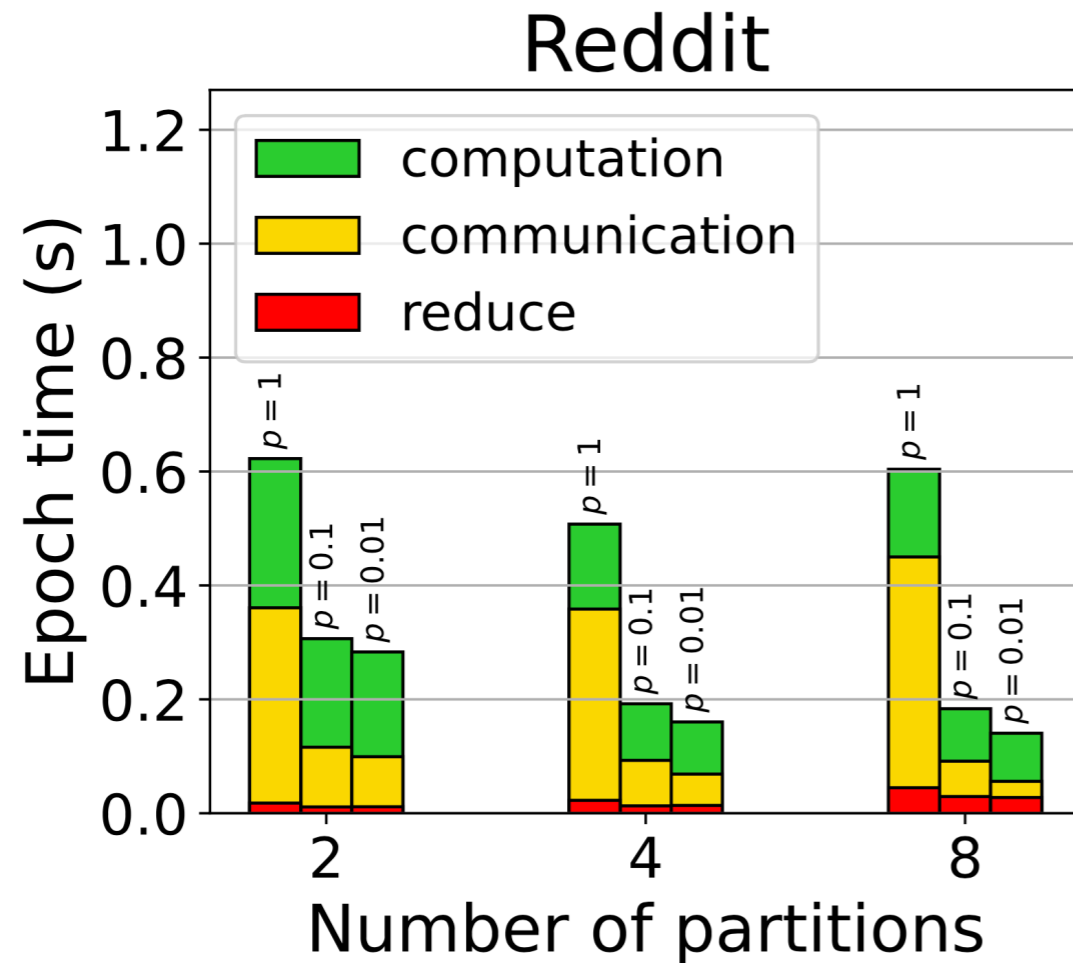


<https://github.com/RICE-EIC/BNS-GCN>

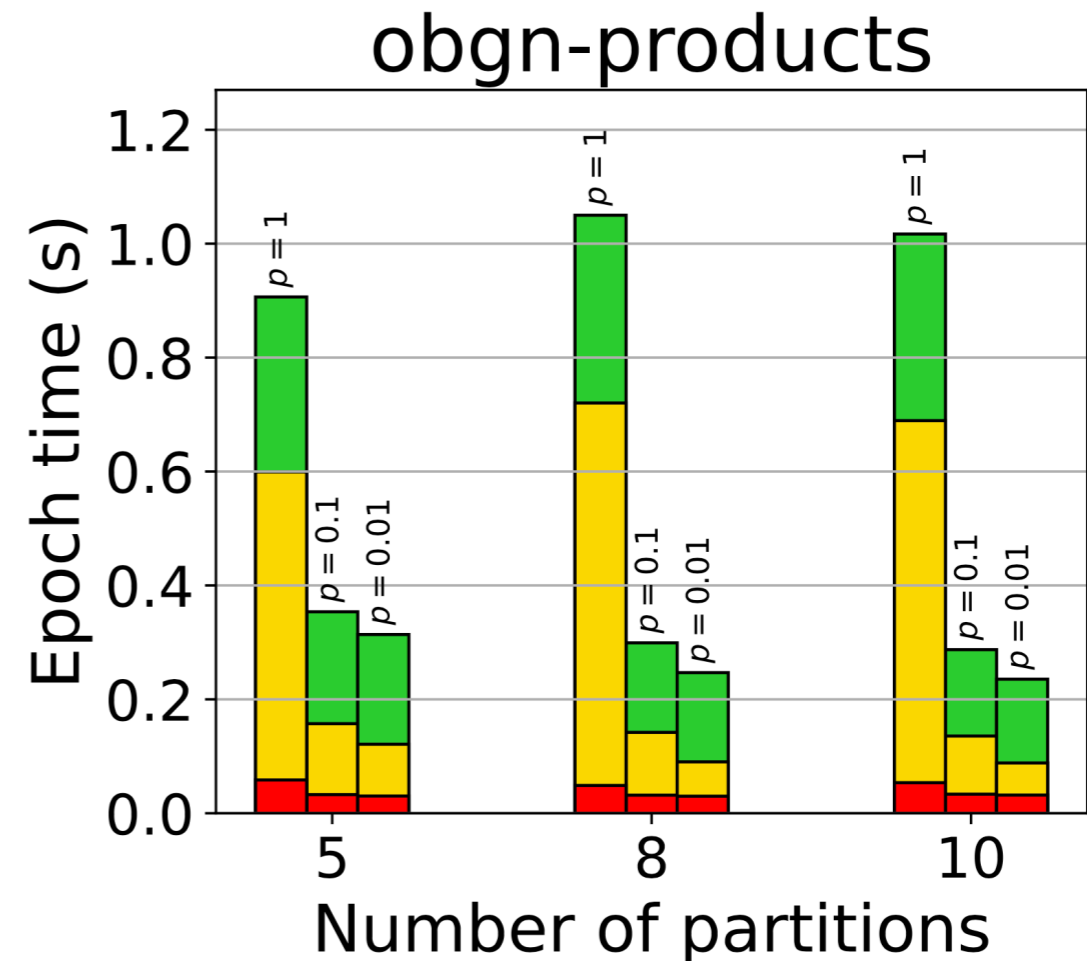
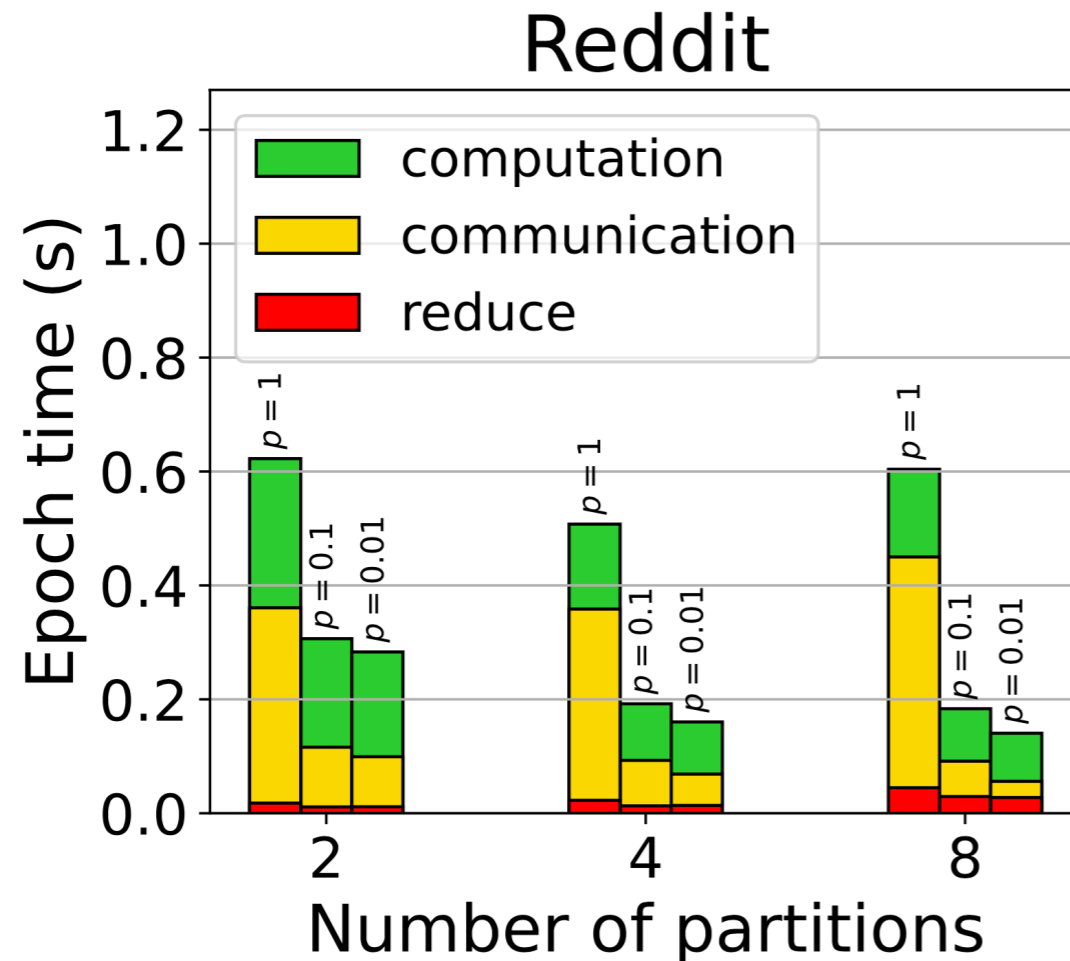


Backup Slides

Time Breakdown

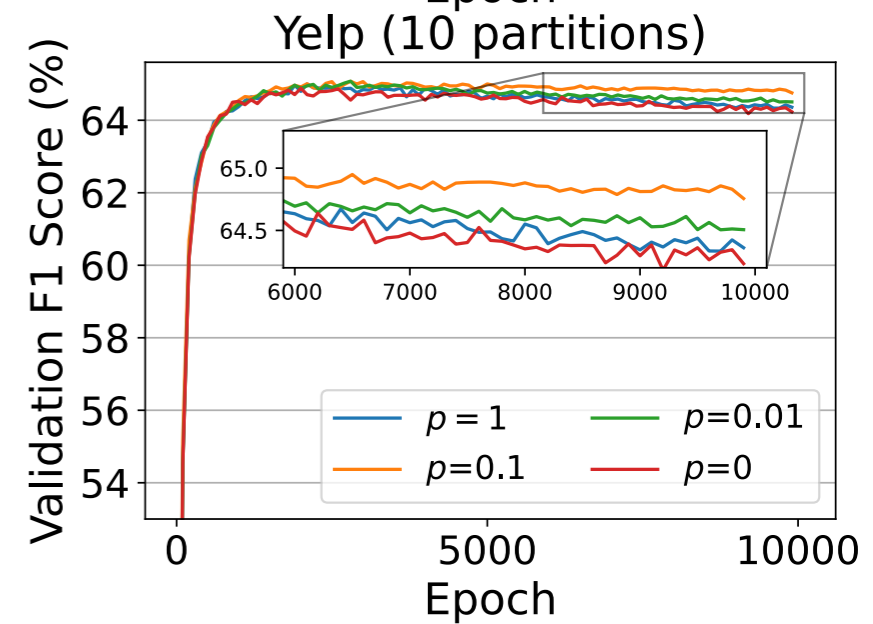
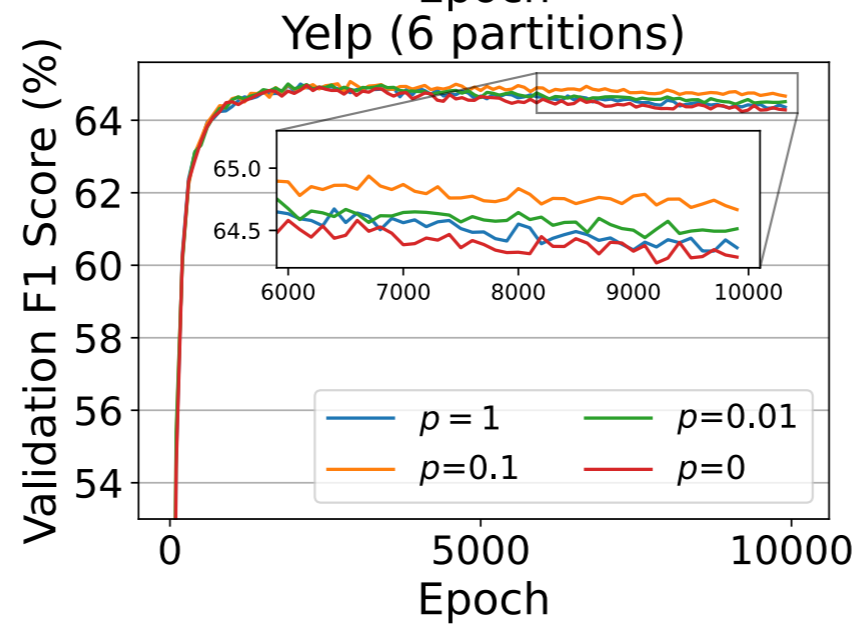
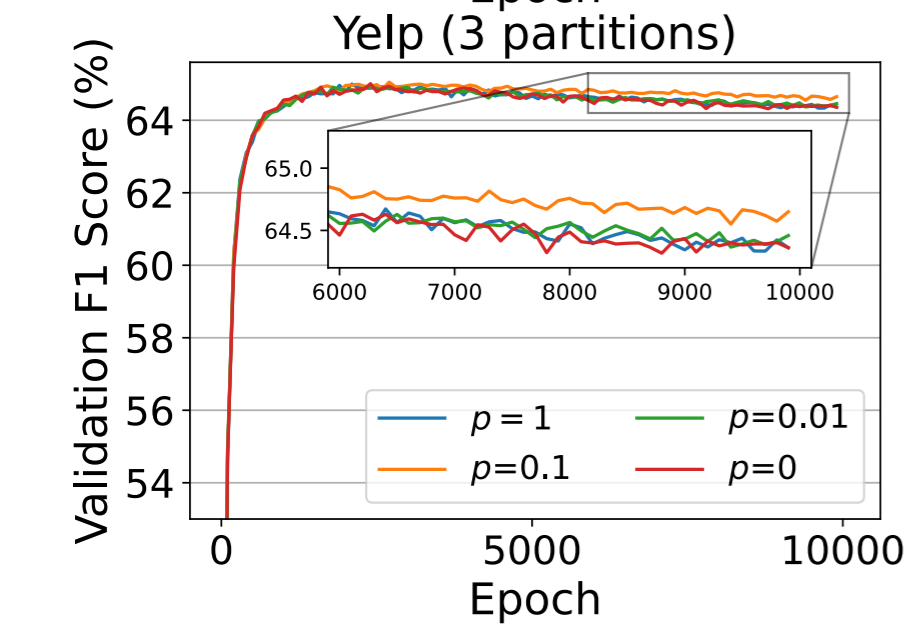
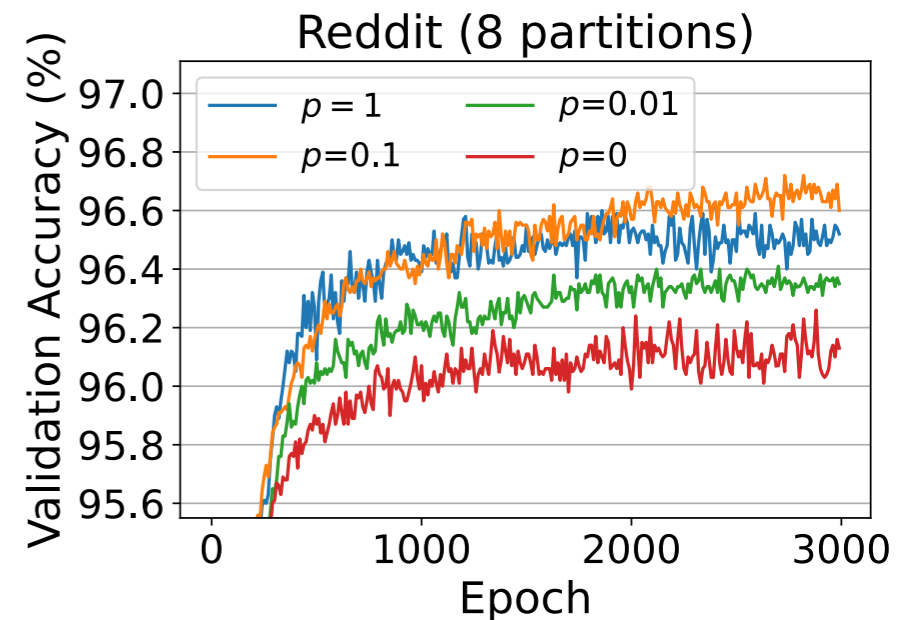
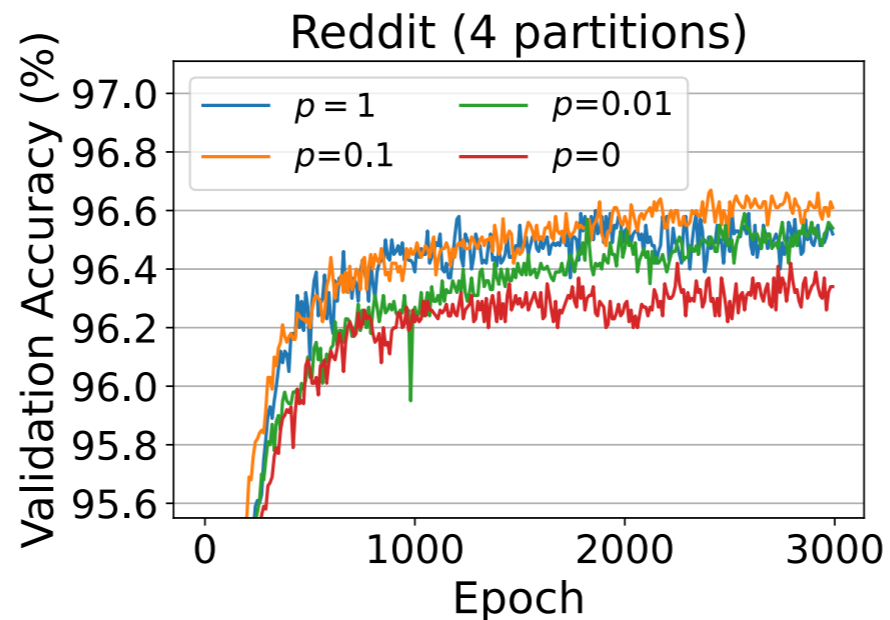
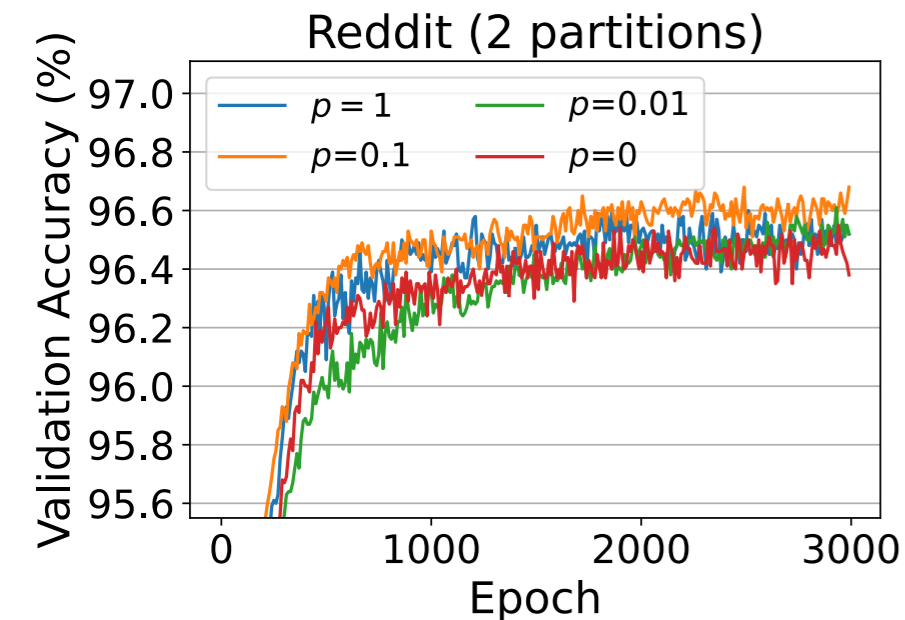


Time Breakdown

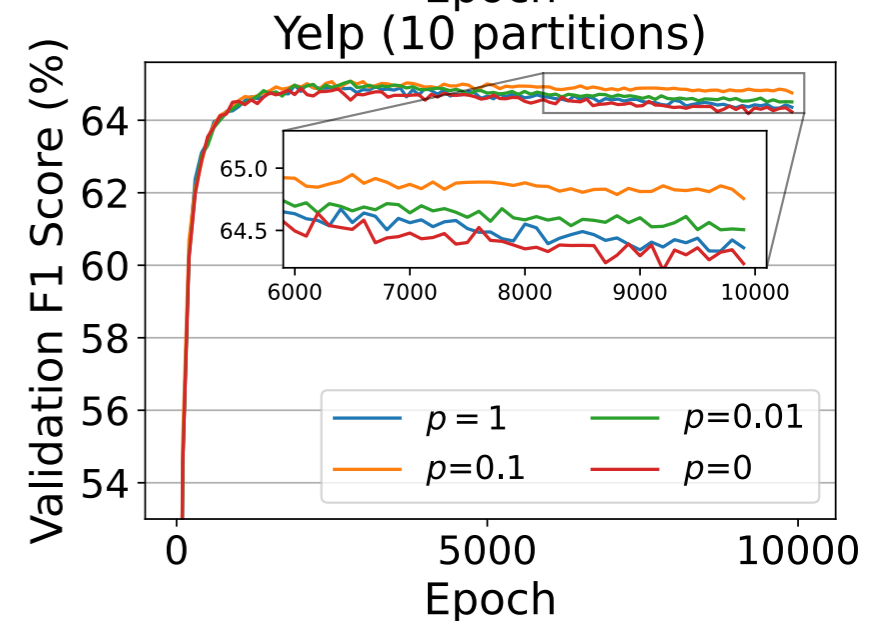
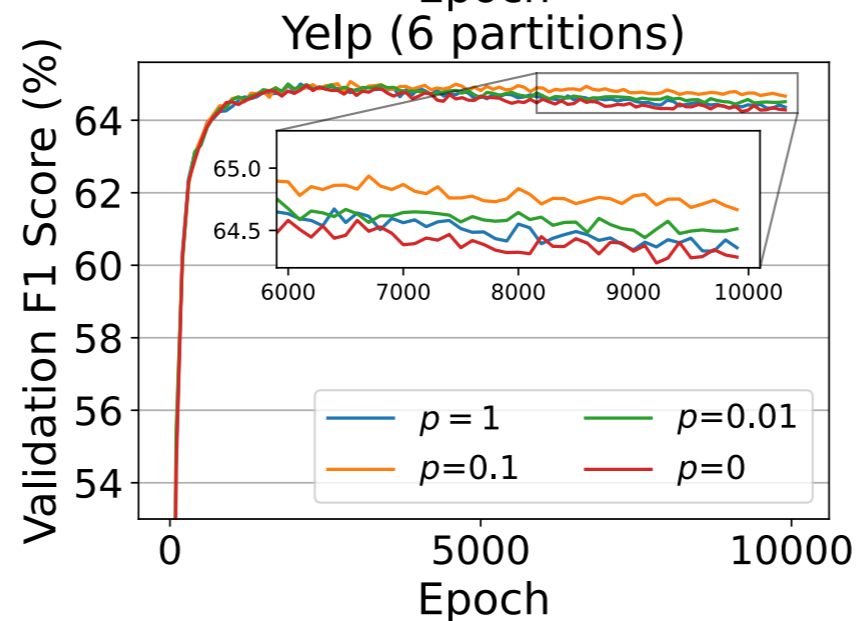
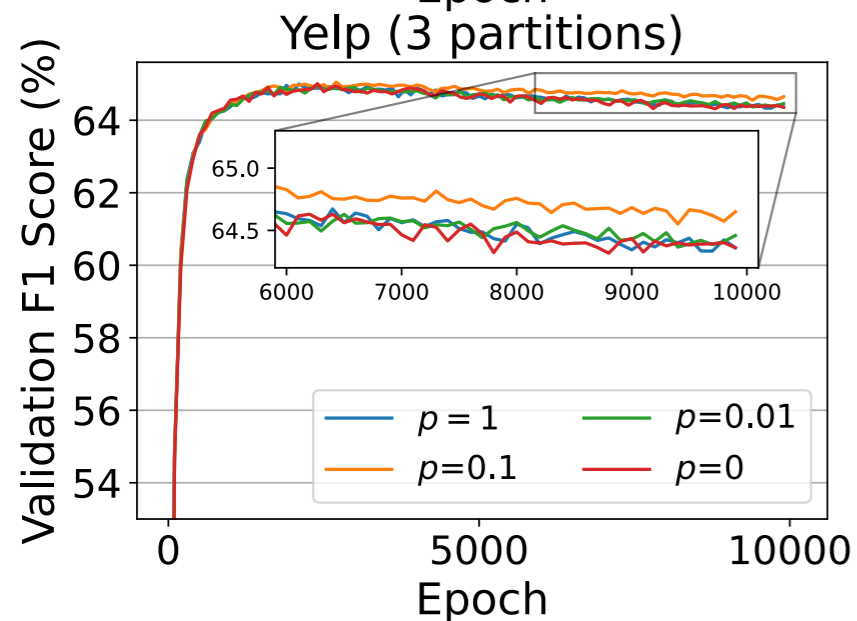
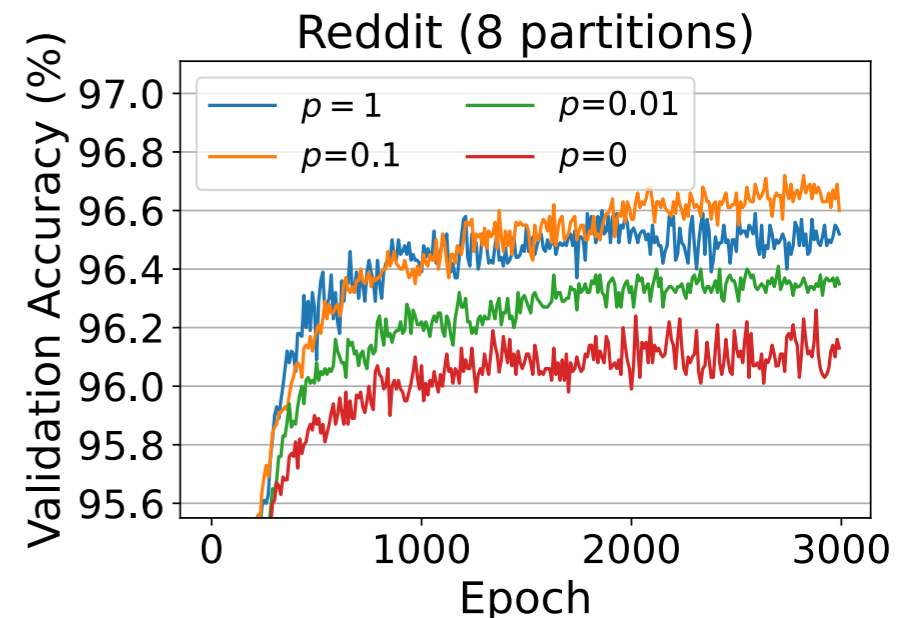
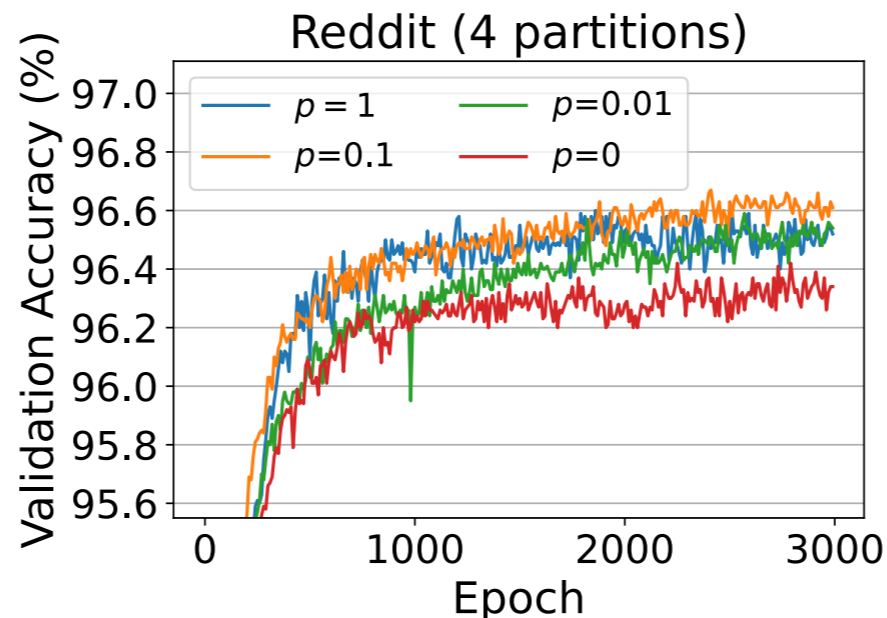
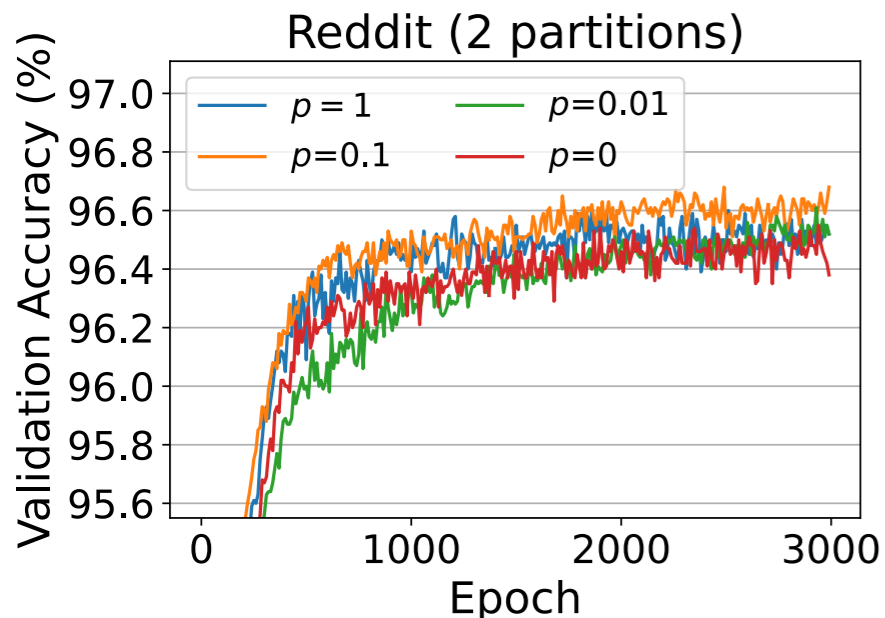


Communication overhead is reduced by **74%~93%**

Training Convergence



Training Convergence



On Reddit: $p=0$ has the worst convergence

On Yelp: $p=0/1$ suffers from overfitting

BNS-GCN with Random Partition

Table 7: Test score (%) of BNS-GCN on top of random partition, where +/- shows the accuracy difference from BNS-GCN on top of METIS in Table 4.

Method	Reddit (8 partitions)		ogbn-products (10 partitions)		Yelp (10 partitions)	
Random+BNS ($p = 1.0$)	97.11	+0.00	79.14	+0.00	65.26	+0.00
Random+BNS ($p = 0.1$)	96.95	-0.20	79.57	+0.27	65.18	-0.16
Random+BNS ($p = 0.0$)	93.37	-3.47	75.39	-3.40	64.92	-0.31

Table 8: Training efficiency improvement of BNS-GCN ($p = 0.1$) on top of different partition methods.

Dataset	Throughput		Memory		# Boundary Nodes	
	METIS	Random	METIS	Random	METIS	Random
Reddit (8 partitions)	3.1×	5.0×	0.47×	0.36×	460k	1,016k
ogbn-products (10 partitions)	3.4×	7.3×	0.75×	0.31×	1,848k	16,797k
Yelp (10 partitions)	3.1×	5.1×	0.83×	0.49×	649k	2,026k

BNS-GCN vs DropEdge vs BES

Table 9: Comparison between BNS-GCN and edge sampling methods, DropEdge and Boundary Edge Sampling (BES).

Dataset	Method	Epoch Comm (MB)	Epoch Time (sec)	Test Score (%)
Reddit (2 partitions)	DropEdge	301.3	0.613	97.12
	BES	207.9	0.484	97.16
	BNS-GCN	30.4	0.319	97.17
ogbn-products (5 partitions)	DropEdge	1364.0	0.938	79.38
	BES	521.1	0.551	79.31
	BNS-GCN	138.7	0.388	79.36
Yelp (3 partitions)	DropEdge	718.7	0.606	65.30
	BES	195.3	0.328	65.30
	BNS-GCN	75.7	0.270	65.32