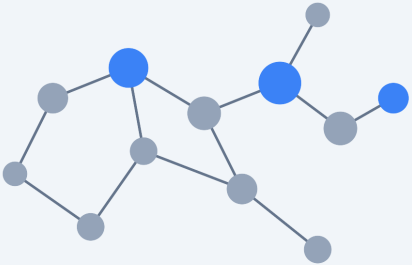
UNIVERSITY OF LUXEMBOURG

GraphSAGE

Inductive Representation Learning on Large Graphs

Thomas Gantz
Alberto Finardi
Tommaso Crippa
Jan Marxen

December 8, 2025



UNIVERSITY OF LUXEMBOURG

Introduction

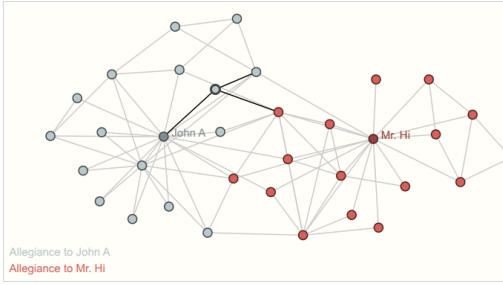
UNIVERSITY OF LUXEMBOURG

Graph Prediction Tasks

Introduction

Primary focus: Node-level prediction — predict properties of individual nodes (classification, regression).

- **Node-level (focus):** Predict node attributes or labels using features and neighbor information.
- **Edge-level:** Predict relationships between node pairs (link prediction, edge classification).
- **Graph-level:** Predict properties of whole graphs (e.g., molecule properties).

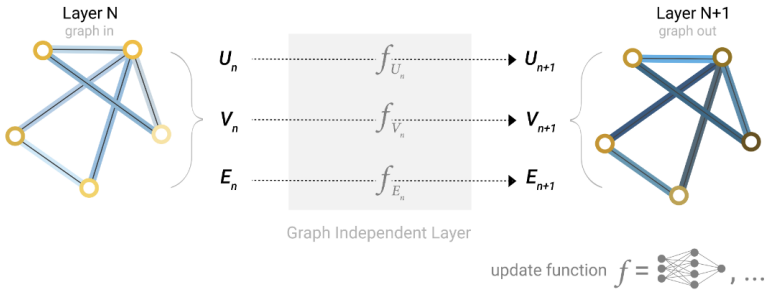


What is a Graph Neural Network?

Introduction

A learnable transformation on graph attributes that:

- Updates node/edge/graph features using **neural networks**
- **Respects graph structure** by aggregating information from neighbors
- Is **permutation invariant** (order of nodes doesn't matter)

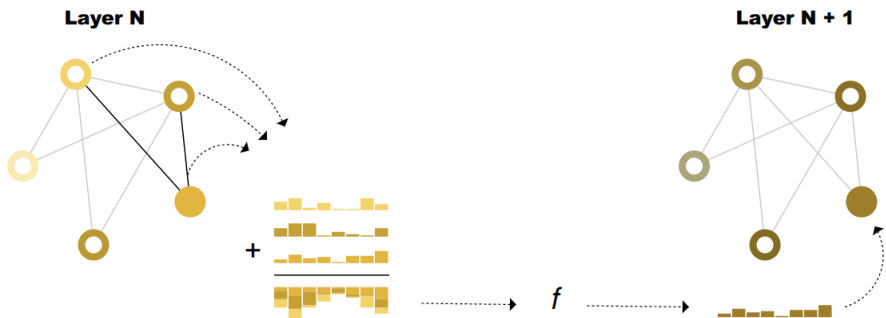


Message Passing: The Core Idea

Introduction

Three steps repeated at each layer:

1. **Gather:** Collect embeddings from neighboring nodes
2. **Aggregate:** Combine neighbors' info (sum / mean / max)
3. **Update:** Apply learned transform using the aggregated vector



UNIVERSITY OF LUXEMBOURG

Message Passing: Notation

Introduction

Message Passing Equation

$$h_v^{(k)} = \sigma\left(W \cdot \left[h_v^{(k-1)}, \text{AGG}\left(\{h_u : u \in N(v)\}\right)\right]\right)$$

- $h_v^{(k)}$ — Node v repr. at layer k
- h_u — Neighbor node u repr.
- $N(v)$ — Neighbors of v
- k — Layer index (hops)
- $\text{AGG}(\cdot)$ — Aggregator (mean, sum, max)
- W — Learnable weight matrix
- σ — Activation (ReLU, tanh)

UNIVERSITY OF LUXEMBOURG

Before GraphSAGE: The Problem

Introduction

Let's audit the limitations of existing approaches...

- **DeepWalk / node2vec:** Transductive: embed every node; new nodes need full retraining.
  - ▷ Not GNNs — random-walk based embeddings, no message passing
- **GCNs:** Often require access to the full graph during training/inference; can be costly to scale.
  - ▷ GNNs — but transductive: fixed node set at training
- **Result:** No compact parametric function to generate embeddings for unseen nodes.

This motivates GraphSAGE's key insight...

The Key Insight

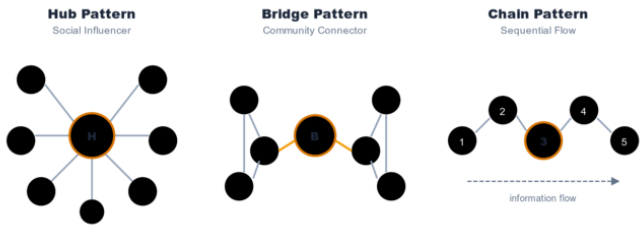
The Key Insight

The Key Insight

Don't learn embeddings for each node...

Learn a FUNCTION that generates embeddings

By sampling & aggregating neighborhood features



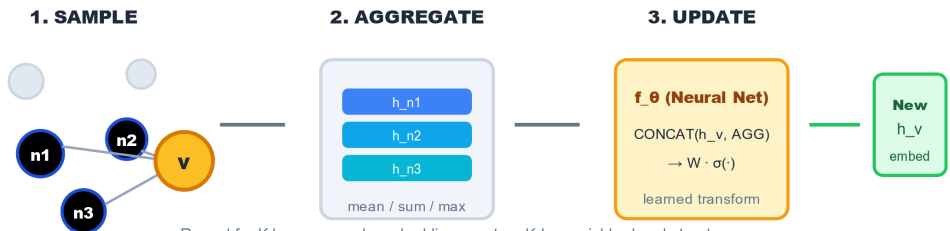
GraphSAGE Framework

GraphSAGE: Inductive Framework

GraphSAGE Framework

Core Principle: Sample + Aggregate

- Learn **aggregator functions** (not node embeddings)
- For any node v: **sample neighbors, aggregate their features**
- Pass through learned neural networks
- **Inference:** Apply same function to unseen nodes



uni.lu
UNIVERSITY OF
LUXEMBOURG

Implementation

UNIVERSITY OF LUXEMBOURG

System Architecture Overview

Implementation

Dataset: ogbn-products (Amazon co-purchase network)

- 2.4M nodes (products), 61M edges, 47 classes
- 8% train / 2% val / 90% test split

Training Environment:

- **HPC Cluster:** MeluXina (Luxembourg National Supercomputer)
- **GPUs:** Up to 4x NVIDIA A100 (40GB) per node
- **Framework:** PyTorch 2.1.2 + PyTorch Geometric
- **Containerization:** Apptainer/Singularity for reproducibility

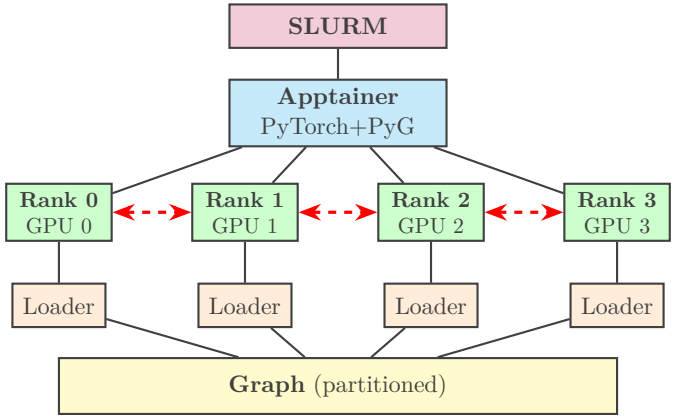
Model Architecture:

- 5-layer GraphSAGE (SAGEConv + LayerNorm + ReLU)
- Hidden dimension: 256, Dropout: 0.5

UNIVERSITY OF LUXEMBOURG

System Architecture Diagram

Implementation



Multi-GPU DDP: data partitioned across 4 GPUs, gradients synced via NCCL

UNIVERSITY OF LUXEMBOURG

Execution Model: Distributed Data Parallel

Implementation

Process Initialization:

- SLURM launches 4 processes (1 per GPU) on single node
- Each process: independent Python interpreter + CUDA context
- NCCL backend for GPU-to-GPU communication (InfiniBand)

Data Distribution:

- Training set (250k nodes) partitioned: 62.5k per rank
- No overlap between ranks → each processes unique subset
- Each rank has independent NeighborLoader for k-hop sampling

Gradient Synchronization:

- DDP automatically wraps model: synchronizes gradients after backward()
- All-reduce operation: averages gradients across all GPUs
- Learning rate scaled linearly: $\text{lr}_{\text{eff}} = \text{lr}_{\text{base}} \times N_{\text{GPUs}}$

UNIVERSITY OF LUXEMBOURG

Execution Model: Training Workflow

Implementation

1. **Data Loading (Parallel)**:
   - Each rank: 4-8 worker processes prefetch batches
   - NeighborLoader samples k-hop subgraphs
   - Pinned memory → GPU transfer (non-blocking)
2. **Forward + Backward (Parallel)**:
   - Each GPU: processes batch independently
   - 5-layer message passing: aggregate → transform → activate
   - Compute loss (cross-entropy), backpropagate gradients
3. **Gradient Synchronization**:
   - DDP all-reduce: average gradients across 4 GPUs (NCCL)
   - Optimizer step with synchronized gradients
4. **Evaluation (Rank 0 Only)**:
   - Every 5 epochs: validation accuracy on full validation set
   - Checkpoint best model, early stopping (patience = 10)

UNIVERSITY OF LUXEMBOURG

Code Structure & Containerization

Implementation

Main Python Scripts:

- `train_graphsage.py` — Single-GPU training baseline
- `train_graphsage_ddp.py` — Multi-GPU DDP training (enhanced)
- `plot_batch.py` — Batch size benchmarking analysis
- `plot_neighbor.py` — Neighbor sampling strategy analysis

Apptainer Container:

- **Base:** `pytorch/pytorch:2.1.2-cuda12.1-cudnn8-runtime`
- **Dependencies:** PyTorch Geometric, pyg-lib, OGB, torch-scatter/sparse
- **Why containerize?**
  - — Complex dependency graph (CUDA-compiled extensions)
  - — Reproducibility across HPC environments
  - — Avoid version conflicts on shared cluster

UNIVERSITY OF LUXEMBOURG

Why HPC and Parallelism?

Implementation

1. **Scale of the Problem:**
   - Dataset cannot fit in single GPU
   - Neighbor sampling: Each batch node samples 5-hop neighborhoods
   - Fanout [15,10,10,10,10] → exponential growth: ∼150k neighbors per seed node
   - Batch 128 nodes × 150k expansion = 19.2M nodes per batch

2. **Memory Bottleneck:**
   - Model parameters: only ∼8 MB
   - Sampled subgraphs + intermediate activations: **30-40 GB** per batch
   - Single GPU (40 GB) cannot handle large batches → slow training

3. **Solution: Multi-GPU Parallelism**
   - Split data across 4 GPUs → 4x memory capacity
   - Effective batch size: 2048 nodes (faster convergence)
   - Training time: **hours instead of days**

Hyperparameters

Hyperparameters

Hyperparameters

Hyperparameter tuning details coming soon...

UNIVERSITY OF LUXEMBOURG

Results

Results

Results

Experimental results coming soon...

UNIVERSITY OF LUXEMBOURG

Any Question?

UNIVERSITY OF LUXEMBOURG

References

References

[1] Dominique Brodbeck et al. “Domesticating Bead: Adapting an Information Visualization System to a Financial Institution”. In: *Proceedings of the IEEE Symposium on Information Visualization (InfoVis ’97)*. Los Alamitos, CA: IEEE Computer Society, 1997, pp. 73–80.

[2] Aditya Grover and Jure Leskovec. “node2vec: Scalable Feature Learning for Networks”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 855–864.

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 1024–1034. URL: https://arxiv.org/abs/1706.02216.

[4] Susan Havre et al. “ThemeRiver: Visualizing Thematic Changes in Large Document Collections”. In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 9–20. DOI: 10.1109/2945.981848.

[5] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “DeepWalk: Online Learning of Social Representations”. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 701–710.

[7] James A. Wise et al. “Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents”. In: *Readings in Information Visualization: Using Vision to Think*. Ed. by Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. San Francisco: Morgan Kaufmann, 1999, pp. 442–445.