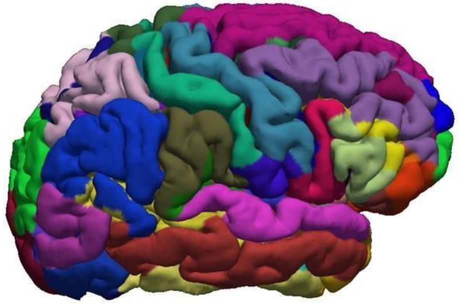
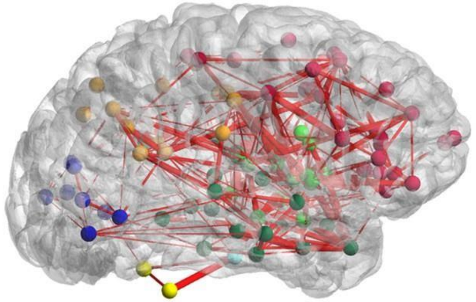
Curriculum Graph Machine Learning: Enhancing Graph Models with Human-like Learning Strategies





SLIIT UNI THE KNOWLEDGE UNIVERSITY

Utrecht University

ICARC

BRAIN LABS

Curriculum Learning: An Efficient Learning Paradigm

2026 ICARC Tutorial Session

Dr. Dharshana Kasthurirathna
Sri Lanka Institute of Information Technology (SLIIT)

Dr. Mahima Weerasinghe
Sri Lanka Institute of Information Technology (SLIIT)

Mr. Menan Velayuthan
Utrecht University

Mr. Sanka Mohottala
Sri Lanka Institute of Information Technology (SLIIT)

Mr. Asiri Gawesha
Sri Lanka Institute of Information Technology (SLIIT)

Mr. Dulara Madhusanka
Sri Lanka Institute of Information Technology (SLIIT)

Ms. Savini Kommalage
Sri Lanka Institute of Information Technology (SLIIT)

SLIIT UNI THE KNOWLEDGE UNIVERSITY

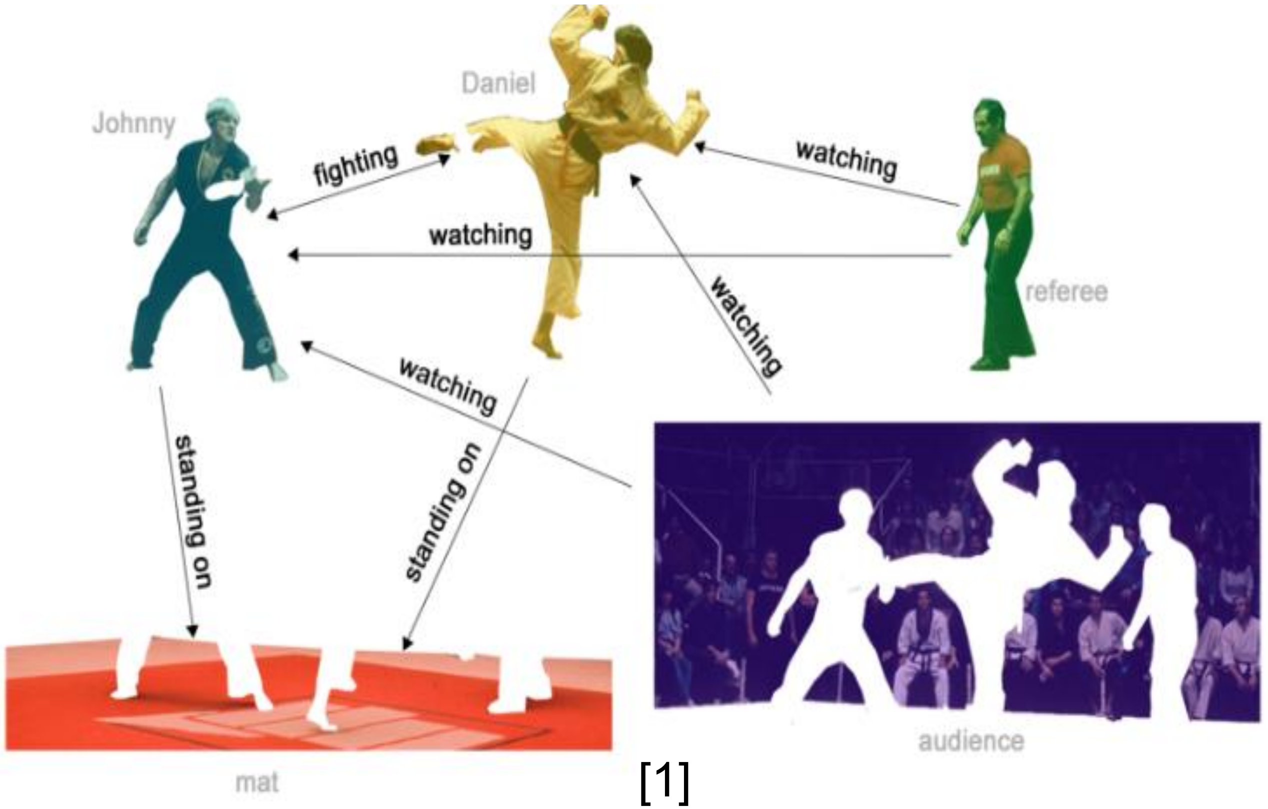
Utrecht University

ICARC

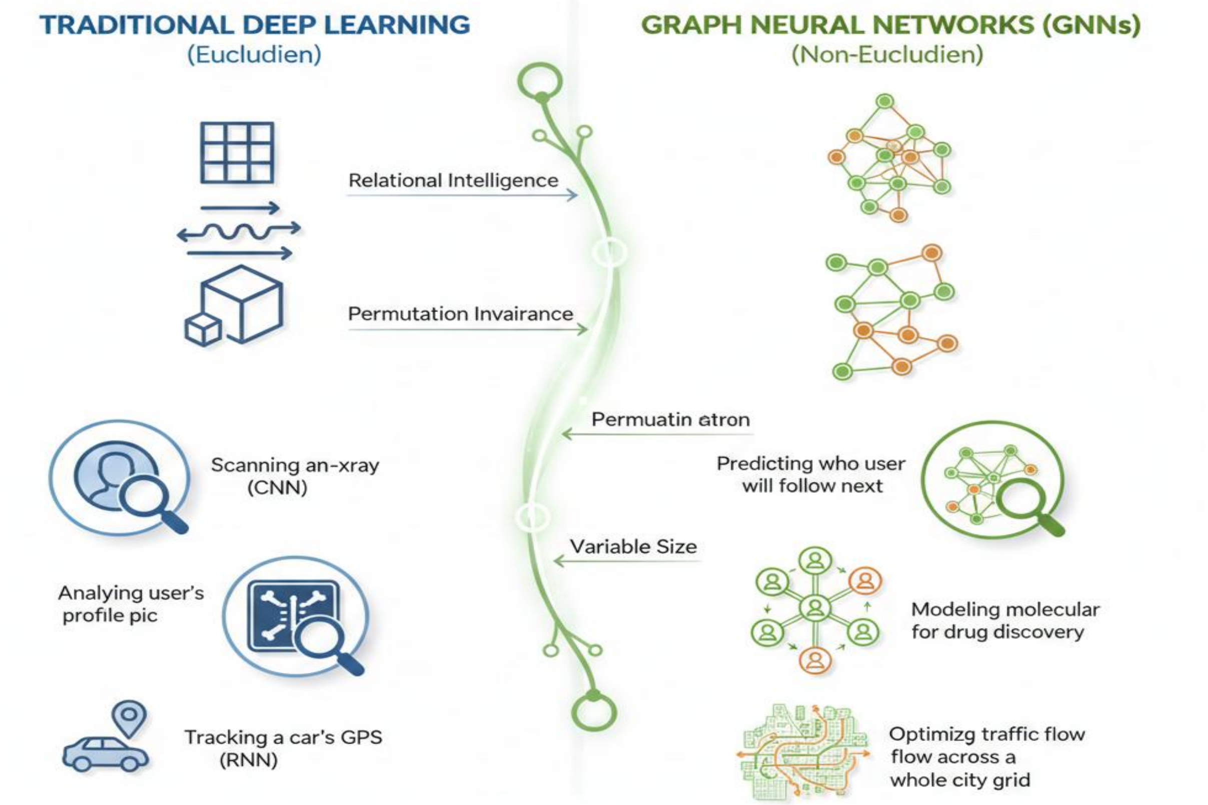
BRAIN LABS

Why GNNs?

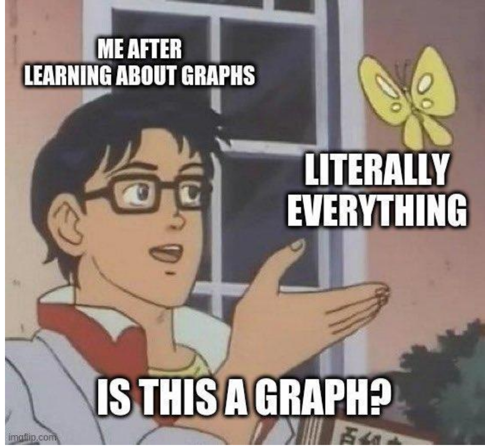
- Traditionally , deep learning deals with Euclidian data such as grids, sequences etc.
- GNNs are designed to capture the complex relationships and dependencies within non-Euclidean data structures, such as graphs and networks.



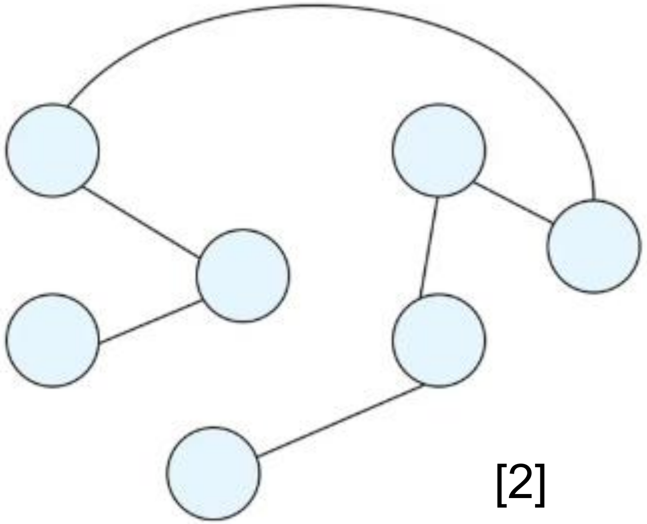
[1]



Definitions



- Graphs are a general language for describing and analyzing entities with relations/interactions.
- Networks are real-world instantiations of graphs with node and/or edge features.

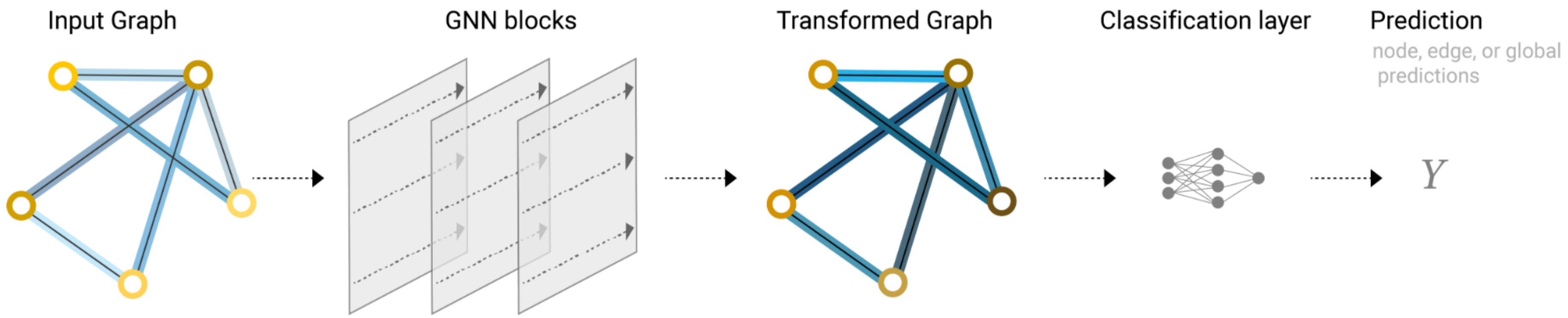


[2]

- **Objects:** nodes, vertices — N
- **Interactions:** links, edges — E
- **System:** network, graph — $G(N,E)$

The Problem with Standard GNN Training

- Most existing Graph Neural Networks (GNNs) train using data samples in random order.
- This ignores that different graph samples have different levels of importance and difficulty, often leading to suboptimal performance.

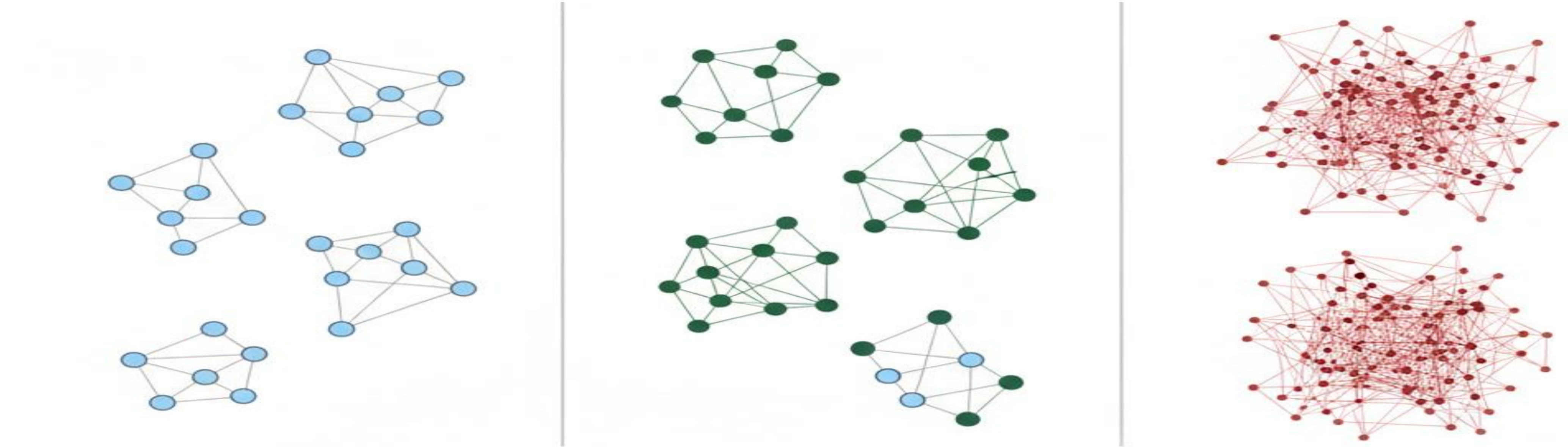


An end-to-end prediction task with a GNN model.

[1]

The Solution: Graph CL

- Integrates Graph Machine Learning with Curriculum Learning (CL) to mimic the human learning process.
- Instead of random training, it organize data in a meaningful order, typically from "easy" to "hard" patterns.



Categorization of Graph CL Methods

Classification by Graph Task Level

Existing methods are primarily categorized by the granularity of the graph task they address:

a) Node Level CL: Focuses on learning presentations for individual nodes (e.g., Node Classification)

b) Link Level CL: Focuses on relations and dependencies between nodes (e.g., Link Prediction)

c) Graph Level CL: Focuses on global properties of the entire graph structure (e.g., Graph Classification)

Classification by Curriculum Type

Within each task level, methods are further divided by how the curriculum is generated:

a) Predefined Graph CL: Uses manually designed, heuristic based policies (e.g., node degree, graph density) to decide training order before training begins.

b) Automatic Graph CL: Relies on computable metrics (e.g., training loss) and model feedback to dynamically design the curriculum during training.

A summary of curriculum graph machine learning methods

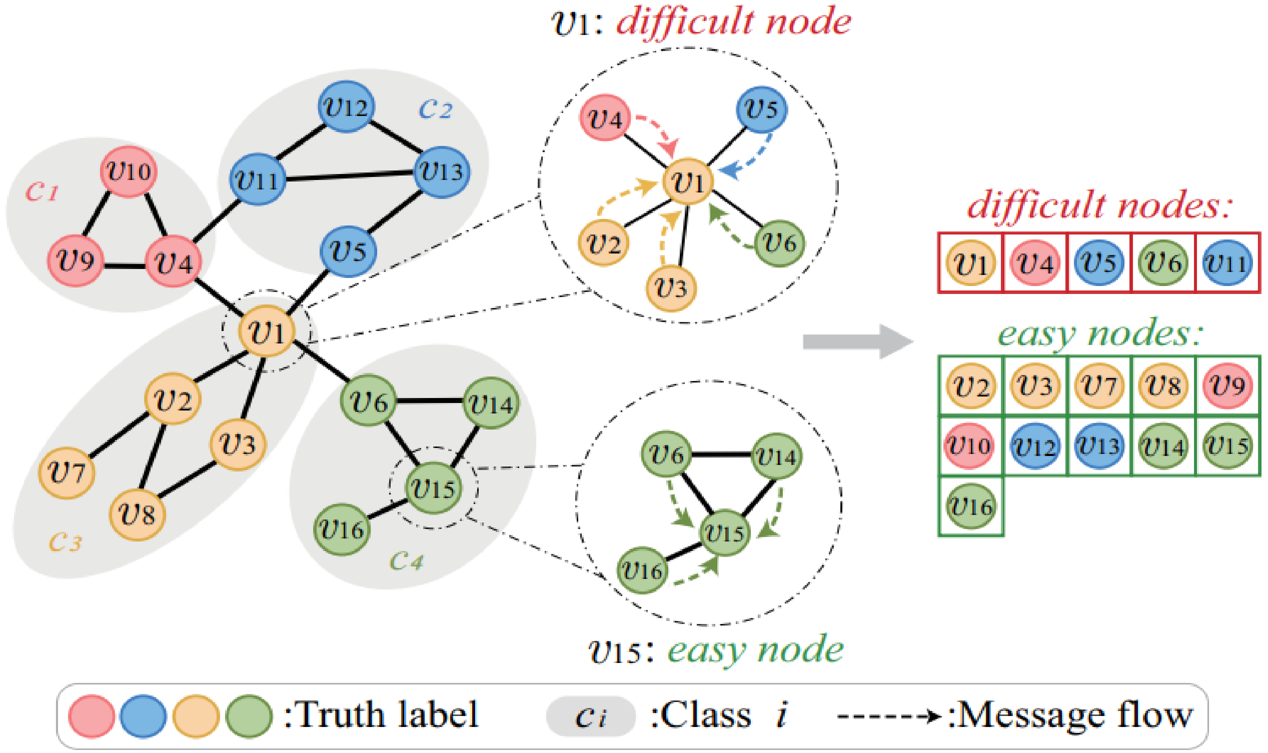
Method	Graph CL Type	Difficulty Measurer	Training Scheduler	Task	Need Label
		Node-level Graph CL			
CLNode [2022]	Predefined	Label Distribution	Continuous	Node Classification	✓
GNN-CL [2022c]	Predefined	Sample Similarity	Continuous	Node Classification	✓
SMMCL [2019]	Predefined	Label Distribution	Discrete	Node Classification	✓
DiGCL [2021]	Predefined	Laplacian Perturbation	Continuous	Node Classification	✗
HSAN [2023]	Predefined	Sample Similarity	Discrete	Node Classification	✗
MTGNN [2020b]	Predefined	Step Length	Discrete	Time Series Forecasting	✓
MentorGNN [2022a]	Automatic	Attention Weight	Discrete	Node Classification	✗
RCL [2023]	Automatic	Self-supervised Loss	Continuous	Node Classification	✓
DRL [2018]	Automatic	Cumulative Reward	Discrete	Node Classification	✓
GAUSS [2022]	Automatic	Sample Loss	Discrete	Node Classification	✓
CGCT [2021]	Automatic	Sample Similarity	Discrete	Image Classification	✓
		Link-level Graph CL			
GCN-WSRS [2018]	Predefined	Sample Similarity	Continuous	Link Prediction	✓
TUNEUP [2022]	Predefined	Node Degree	Discrete	Link Prediction	✓
CHEST [2023]	Predefined	Pretraining Task	Discrete	Link Prediction	✗
GTNN [2022]	Automatic	Sample Loss	Discrete	Relation Extraction	✓
		Graph-level Graph CL			
CurGraph [2021b]	Predefined	Label Distribution	Discrete	Graph Classification	✓
CuCo [2021]	Predefined	Sample Similarity	Continuous	Graph Classification	✗
HACL [2022]	Predefined	Sample Size	Discrete	Graph Classification	✓
Dual-GCN [2021]	Predefined	BLEU Metric	Discrete	Image Captioning	✓
CurrMG [2022]	Automatic	Domain Knowledge	Continuous	Graph classification	✓
HAC-TSP [2022b]	Automatic	Solution Cost	Continuous	Travelling Salesman Problem	✗

[3]

Methods Used in Research Papers

Not All Nodes Are Equal (CLNode)

- Traditional Graph Neural Networks (GNNs) assume all training nodes are of equal quality, but “difficult” nodes (like inter-class nodes at boundaries or mislabeled nodes) can significantly degrade accuracy and robustness.



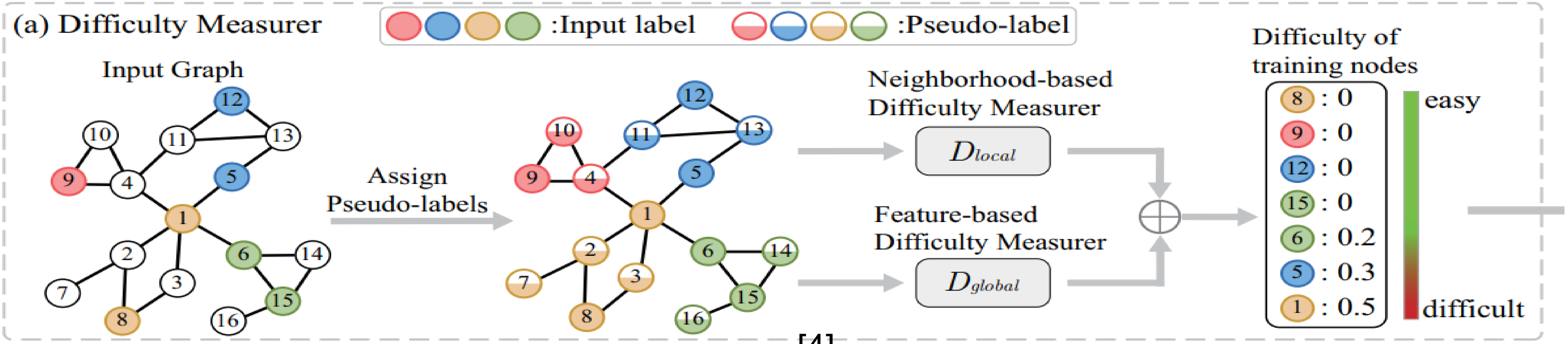
[4]

CLNode Framework

- Multi Perspective Difficulty Measurer: This identifies “difficult” nodes from two angles:

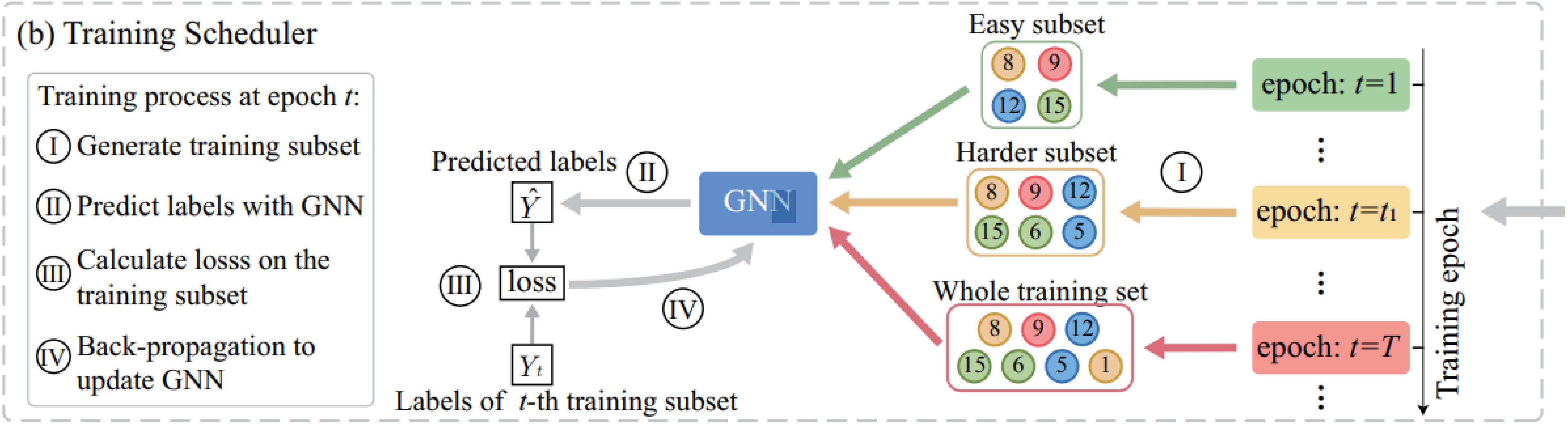
a) **Local Neighborhood-based Measurer**: Measures the “Homophily” of a node. Nodes with many neighbors from different classes are labeled **difficult** (inter-class boundary nodes), while nodes surrounded by the same class are **easy**.

b) **Global Feature-based Measurer**: Calculates the distance between a node’s features and the average feature vector (centroid) of its class. Nodes that are “outliers” or far from their class center are deemed **difficult**.



[4]

- Continuous Training Scheduler: Instead of training on everything at once, CLNode uses a “pacing function” to start training with only the easiest nodes and gradually introduces harder ones as epochs progress.



[4]

Key Results and Contributions

- **Broad Compatibility:** CLNodes is a “plug and play” framework that improves the performance of various GNN architectures (like GCN, GAT, and GraphSAGE) without increasing their time complexity.
- **Performance Boost:** Across five benchmark datasets, adding CLNode consistently improved node classification accuracy (e.g., a 5.7% improvements for GCN on the Amazon Computer dataset) and enhanced robustness against label noise.

Node classification performance on five datasets

	Method	Cora	CiteSeer	PubMed	A-Computers	A-Photo
GCN	Original	73.5±0.8	62.8±2.6	64.3±2.9	79.0±3.7	89.1±0.8
	+CLNode	**77.0±0.7**	**65.5±2.3**	**65.9±1.3**	**84.7±0.5**	**90.8±1.0**
	(Improv.)	3.5%	2.7%	1.6%	5.7%	1.7%
GraphSAGE	Original	70.1±2.3	57.4±3.7	61.3±1.4	71.7±2.4	83.0±2.6
	+CLNode	**72.1±1.4**	**60.3±3.1**	**64.1±3.8**	**77.5±1.6**	**87.5±1.2**
	(Improv.)	2.0%	2.9%	2.8%	5.8%	4.5%
GAT	Original	74.2±1.2	63.7±2.8	64.6±2.5	80.2±0.8	89.4±1.8
	+CLNode	**77.1±1.1**	**65.3±2.6**	**68.2±2.6**	**82.6±1.1**	**90.1±1.1**
	(Improv.)	2.9%	1.6%	3.6%	2.4%	0.7%
SuperGAT	Original	74.4±4.3	**64.8±3.3**	67.4±4.3	81.2±2.0	87.3±2.0
	+CLNode	**75.5±2.7**	63.0±3.2	**72.2±3.0**	**83.4±2.4**	**88.8±1.2**
	(Improv.)	1.1%	-	4.8%	2.2%	1.5%
JK-Net	Original	74.0±1.5	62.1±3.7	**66.0±1.7**	83.2±1.3	89.2±0.7
	+CLNode	**76.8±0.8**	**63.6±1.2**	**71.5±3.2**	**84.4±1.0**	**90.4±0.9**
	(Improv.)	2.8%	1.5%	5.5%	1.2%	1.2%
GCNII	Original	76.2±4.0	64.5±4.3	70.8±6.1	79.8±1.8	87.4±2.1
	+CLNode	**77.8±2.1**	**66.5±2.2**	**71.3±4.6**	**82.2±1.5**	**89.3±2.0**
	(Improv.)	1.6%	2.0%	0.5%	2.4%	1.9%

[4]

PinSage Training Strategy: Curriculum Learning & Hard Negatives

- The authors developed a “curriculum training scheme” that feeds the model increasingly difficult training examples as training progresses. By starting with easy examples and moving to harder ones, they achieved a **12% performance gain.**

Query **Positive Example** **Random Negative** **Hard Negative**

[5]

What is Negative Sampling?

Think of Negative Sampling as creating a “multiple choice question” for the computer to solve.

To teach the model what an image IS, you also have to show it what the image IS NOT.

- The Query (The Question): A picture of a Flower
- The Positive (The Right Answer): Another image of a Flower
- The Random Negatives (Random Wrong Answers): An image of a Hat
- The Hard Negatives (Wrong Answer): An image of a Bird sitting on a Flower stick (which is quite similar to the query)

The Problem with Random Negatives

- With a catalog of 2 billion items, randomly picking 500 negative items is too "easy" for the model. The chances of a random item being similar to the query are tiny.
- This provides a "low resolution" for learning; the model doesn't learn to distinguish between highly relevant items and slightly relevant items.

Introducing "Hard Negatives"

- Hard Negatives are items that are somewhat related to the query item but are not the correct "positive" match.
- They are generated by ranking items using Personalized PageRank scores with respect to the query item. Items ranked between 2000 and 5000 are sampled as hard negatives.
- These items are similar to the query to be confusing, which forces the model to learn finer-grained distinctions between "true" matches and "close" matches.

❑ *Using hard negatives immediately makes training difficult and doubles the number of epochs required for convergence.*

The Curriculum Learning Strategy

- Feed the algorithm “harder-and-harder” examples over time rather than starting with the most difficult ones immediately.
- During the first epoch of training NO hard negative items are used. In subsequent epochs, hard negative are gradually introduced.
- At epoch *n* the system adds *n-1* hard negative items to the set of negative items for each query.

Results and Impact

- Robustness: This strategy prevents the model from getting stuck or converging too slowly, which happens if difficult examples are used too early.
- Performance Gain: The curriculum training scheme resulted in a 12% Performance gain in offline evaluation metrics compared to training without this schedule.
- Outcome: The model successfully learns to distinguish highly related pins from only slightly related ones without sacrificing training efficiency.

[1] https://distill.pub/2021/gnn-intro/

[2] https://www.ibm.com/think/topics/graph-neural-network

[3] H Li, X Wang, W Zhu - arXiv preprint arXiv:2302.02926, 2023 - arxiv.org

[4] X Wei, X Gong, Y Zhan, B Du, Y Luo, W Hu - Proceedings of the sixteenth …, 2023 - dl.acm.org

[5] R Ying, R He, K Chen, P Eksombatchai… - Proceedings of the 24th …, 2018 - dl.acm.org