

Learning Posterior Predictive Distributions for Node Classification from Synthetic Graph Priors

Jeongwan Choi¹, Jongwoo Kim¹, Woosung Kang¹, Noseong Park¹

¹KAIST

ICLR 2026 · KCC 2026 Top Conference Session

Outline

- 1 Introduction
- 2 Background: A Bayesian View
- 3 Method: NodePFN
- 4 Experiments
- 5 Conclusion

The Problem: A New GNN for Every Graph

- GNNs need **per-graph training**: a model trained on one graph does not transfer to another.
- Real graphs are **heterogeneous** — homophily, community structure, and features vary widely.
- We want LLM-style **in-context** generalization: one pre-trained model that adapts with *no* updates.

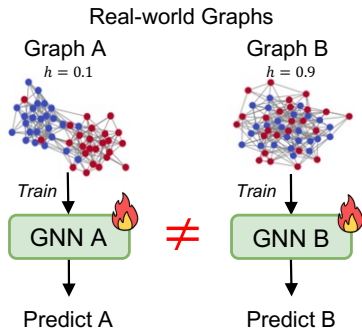
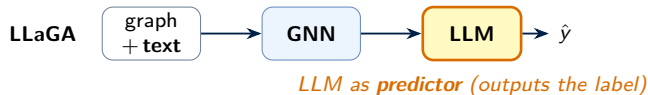
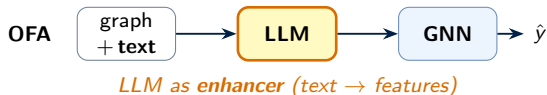


Figure: A new model for every graph

Question: Can a *single* model classify nodes on *any* graph, with *no* graph-specific training?

The Gap: Existing Approaches Fall Short

- LLM-based graph methods^{[1][2]} need **text-attributed** graphs and an **LLM** in the loop — but the LLM is used in *different roles*, and either way it is a hard dependency.



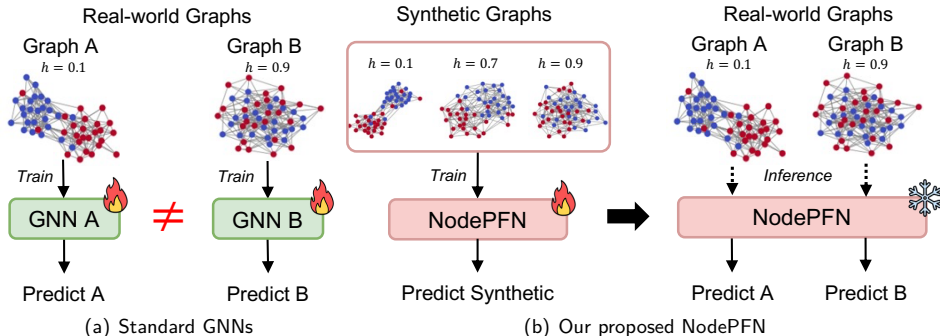
Enhancer or predictor — both *depend on an LLM and text*. **NodePFN** needs neither —
so where does it learn from?

[1] Hao Liu et al. "One For All: Towards Training One Graph Model For All Classification Tasks". In: *ICLR*. 2024.

[2] Runjin Chen et al. "Llaga: Large language and graph assistant". In: *arXiv preprint arXiv:2402.08170* (2024).

Our Idea: Learn from Synthetic Graphs

- NodePFN **never trains on real graphs**. We pre-train *once* on **synthetic** graphs.
- At test time it reads the labeled nodes as **context** and predicts the rest.



Why does training on *synthetic* graphs generalize? It learns a **posterior predictive distribution**.

Outline

- 1 Introduction
- 2 Background: A Bayesian View
- 3 Method: NodePFN
- 4 Experiments
- 5 Conclusion

The Posterior Predictive Distribution (PPD)

- **Task.** Predict a **test** node's label given the labeled **train** nodes as context, $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}$.
- **PPD.** Rather than choose a single hypothesis, marginalize over *all* hypotheses ϕ , each weighted by its posterior $p(\phi \mid \mathcal{D}_{\text{train}})$:

$$\underbrace{p(y_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}})}_{\text{posterior predictive (PPD)}} = \int \underbrace{p(y_{\text{test}} \mid \mathbf{x}_{\text{test}}, \phi)}_{\text{prediction under } \phi} \underbrace{p(\phi \mid \mathcal{D}_{\text{train}})}_{\text{posterior over } \phi} d\phi$$

ϕ : a hypothesis mapping features $\rightarrow$ labels $\int \cdots d\phi$: marginalize over *all* hypotheses

- **Bayes-optimal:** the PPD minimizes expected loss and **adapts to each graph** through the posterior.
- The integral is **intractable**; we therefore treat the PPD as a **learning target**.

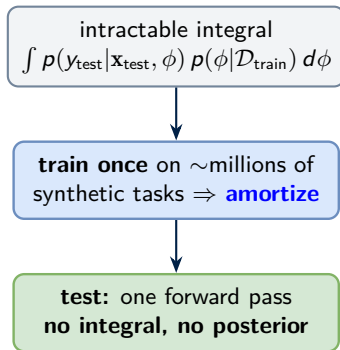
Prior-Fitted Networks: Learn to Imitate the PPD

- The prior generates *unlimited* synthetic datasets with known answers.
- Train f_θ to predict held-out labels from the context:

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}_{\mathcal{D} \sim p(\mathcal{D})}}_{\text{average over millions of synthetic datasets}} \left[\underbrace{-\log f_\theta(y_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}})}_{\text{penalty if it under-rates the true label}} \right]$$

Theorem^[3]: Loss minimization $\Rightarrow$ PPD

Minimizing this loss = getting closer to the *true* PPD; with enough capacity, f_θ *becomes* the PPD.



Amortize: solve a million practice problems *once*; at test time just recall the pattern.

Outline

- 1 Introduction
- 2 Background: A Bayesian View
- 3 Method: NodePFN**
- 4 Experiments
- 5 Conclusion

Method Overview: Three Phases

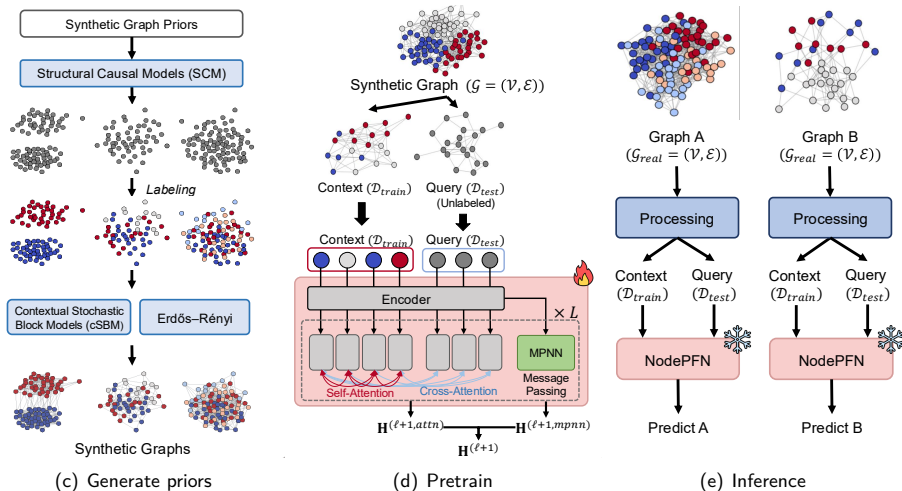
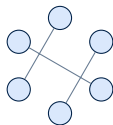


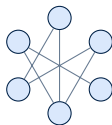
Figure: (a) Sample diverse synthetic graphs $\rightarrow$ (b) pre-train a graph-aware PFN *once* $\rightarrow$ (c) deploy on any real graph instantly

A Principled Prior: Inheriting Decades of Network Science

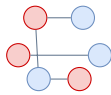
- We don't hand-design a prior — we **inherit** the canonical **random graph models** of network science as **inductive bias**.



Sparse ER



Dense ER



cSBM, low h



cSBM, high h



BA, hub

- Random (ER): pure random edges. (Erdős–Rényi, 1959)^[4]
- Contextual stochastic block model (cSBM): planted **community** blocks with node features. (Holland 1983^[5]; Deshpande 2018^[6])
- Scale-free (BA): a few high-degree **hubs**. (Barabási–Albert, 1999)^[7]

[4] Paul Erdős and Alfréd Rényi. “On random graphs I”. In: *Publ. math. debrecen* 6.290–297 (1959), p. 18.

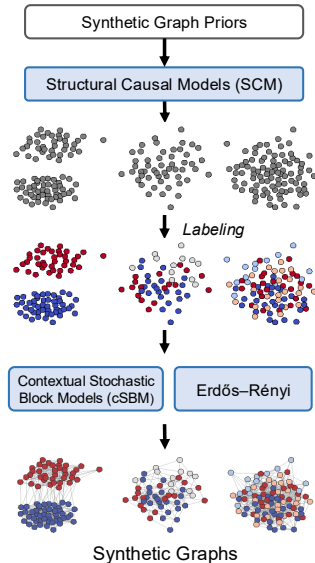
[5] Paul W Holland et al. “Stochastic blockmodels: First steps”. In: *Social networks* 5.2 (1983), pp. 109–137.

[6] Yash Deshpande et al. “Contextual stochastic block models”. In: *Advances in neural information processing systems* 31 (2018).

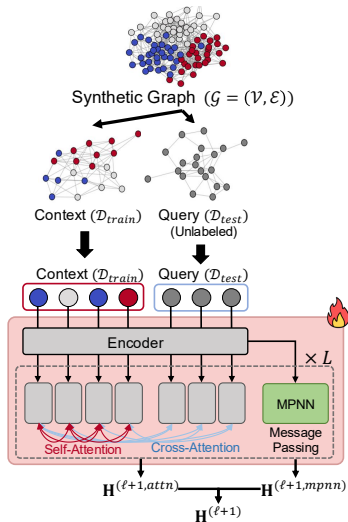
[7] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.

Synthetic Graph Priors from Random Networks

- **Design goal:** a prior whose **coverage** spans real-graph regimes:
 - **Features & labels:** a structural causal model (SCM), used by Hollmann et al.^[8], draws features and labels.
 - **Structure on top:** sample edges as **ER + cSBM** (50/50).
 - **Homophily sweep:** $h = \frac{p_{in}}{p_{in} + p_{out}} \in [0.1, 0.9]$.



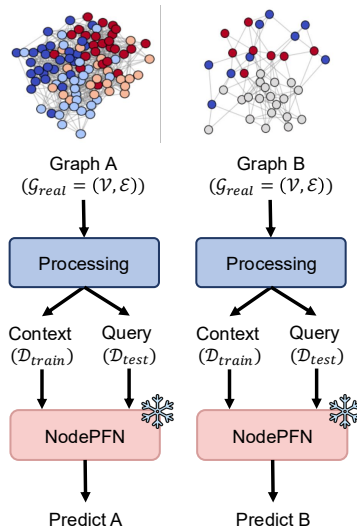
Dual-Branch Architecture: The In-Context Mechanism



- **Attention:** train nodes self-attend; test nodes cross-attend.
- Test nodes don't attend each other $\Rightarrow$ predictions *conditionally independent* given context (matches the PPD).
- **MPNN branch:** adds structural bias that attention can't capture.

Training & Inference

- Training — one-time, offline
 - Synthetic graphs only — never sees real data.
 - Random train/test split per synthetic graph.
 - Minimize expected cross-entropy.
- Inference — per real graph, one forward pass
 - Features: processing & smoothing, then zero-pad to a fixed width — or **dim-reduction** when the dimension exceeds the max.
 - Classes: using the first k of the max softmax outputs for a k -class graph.
 - No gradient updates, no fine-tuning.



Outline

- 1 Introduction
- 2 Background: A Bayesian View
- 3 Method: NodePFN
- 4 Experiments**
- 5 Conclusion

Research Questions & Experimental Setup

- Research questions:
 - RQ1 — **Robustness**: stable across controlled homophily?
 - RQ2 — **Generalization**: accurate on 23 real benchmarks?
 - RQ3 — **Comparison**: vs. training-free & LLM-based methods?
 - RQ4 — **Structure only**: feature-free classification?
 - RQ5 — **Ablation**: what does each component contribute?
- Setup:
 - **23** real benchmarks — 13 homophily + 10 heterophily.
 - **One** pre-trained NodePFN for *all* datasets — no per-graph tuning.

RQ1: Stable Across the Homophily Spectrum

- Synthetic Cora, sweep homophily h from 0 to 1.
- MLP: flat (ignores structure).
- GCN/GAT: **collapse at low h** (heterophily).
- NodePFN: high everywhere — no sudden drop.

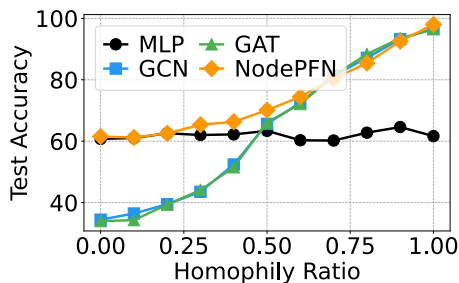


Figure: Accuracy vs. homophily h (synthetic Cora)

RQ2: One Model Wins on 23 Benchmarks

- Best avg. rank **1.70** (next best 4.38); biggest gains on **heterophily**, where GNNs struggle.

Table: Average over 23 node-classification benchmarks (a single pre-trained NodePFN).

Method	Homophily (13)	Heterophily (10)	Avg. Acc. $\uparrow$	Avg. Rank $\downarrow$
MLP	56.43	58.17	57.30	7.41
GCN	73.05	58.84	66.63	5.86
GAT	74.39	59.11	67.67	5.57
GraphAny ^[9] (best)	71.45	61.62	66.24	4.38
NodePFN	77.39	65.14	71.27	1.70

[9] Jianan Zhao et al. "Fully-inductive Node Classification on Arbitrary Graphs". In: *ICLR*. 2025.

RQ3: Comparison with Training-free Models and LLM-based GFMs

Table: Training-free models vs NodePFN.

Method	Cora	Pubmed	Wisconsin	Texas
Linear	52.80	59.30	80.00	32.35
SGC ^[10]	78.20	72.98	57.64	46.03
HGC	22.50	46.32	64.32	57.54
LabelProp ^[11]	60.30	63.44	16.08	23.53
TFGNN ^[12]	60.03	40.04	14.51	19.91
NodePFN	82.06	78.00	81.18	76.22

Table: Accuracy under the setting of GLBench^[13]

Method	Cora	Citeseer	Pubmed	WikiCS
InstructGLM ^[14]	69.10	51.87	71.26	45.73
GraphText ^[15]	<u>76.21</u>	59.43	<u>75.11</u>	67.35
LLaGA ^[16]	74.42	55.73	68.82	73.88
OFA ^[17]	75.24	73.04	75.61	<u>77.34</u>
NodePFN	76.38	<u>63.08</u>	68.18	76.29

[10] Felix Wu et al. "Simplifying graph convolutional networks". In: *ICML*. 2019.

[11] Xiaojin Zhu and Zoubin Ghahramani. "Learning from labeled and unlabeled data with label propagation". In: *Technical Report CMU-CALD-02-107*. 2002.

[12] Ryoma Sato. *Training-free Graph Neural Networks and the Power of Labels as Features*. 2024.

[13] Yuhan Li et al. "Glbench: A comprehensive benchmark for graph with large language models". In: *NeurIPS* (2024).

[14] Ruosong Ye et al. "Language is all a graph needs". In: *arXiv preprint arXiv:2308.07134* (2023).

[15] Jianan Zhao et al. "Graphtext: Graph reasoning in text space". In: *arXiv preprint arXiv:2310.01089* (2023).

[16] Runjin Chen et al. "Llaga: Large language and graph assistant". In: *arXiv preprint arXiv:2402.08170* (2024).

[17] Hao Liu et al. "One For All: Towards Training One Graph Model For All Classification Tasks". In: *ICLR*. 2024.

RQ4: Classification from Structure Alone

- NodePFN learns **topological roles** without informative features — outperforming Node2Vec^[18], LINE^[19], and GNN baselines.

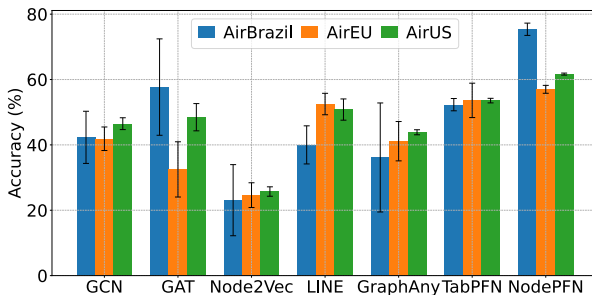


Figure: Structural node classification results

[18] Aditya Grover and Jure Leskovec. "Node2vec: Scalable Feature Learning for Networks". In: *KDD*. 2016, pp. 855–864.

[19] Jian Tang et al. "LINE: Large-Scale Information Network Embedding". In: *TheWebConf (former WWW)*. 2015, pp. 1067–1077.

RQ5: Why It Works — ablations

Table: Ablations (RQ5).

	Cora	Wisc.	Tolok.
TabPFN ^[20]	53.1	72.9	78.2
NodePFN w/o MPNN	75.5	70.1	78.1
NodePFN-Seq	80.6	78.8	77.9
NodePFN (Full)	82.1	81.2	78.6

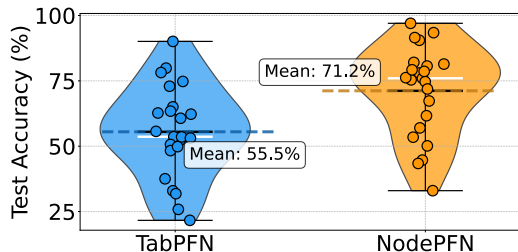


Figure: TabPFN vs. NodePFN

Cheaper Total Time-to-Deployment & Prior Data Scaling

- GNNs: retrain **per dataset**
- NodePFN: one-time pre-train (**6 GPU-hours**), then **amortized** over unlimited datasets.

Over 23 datasets	GCN	NodePFN
Avg. accuracy	66.63	71.27
Avg. ranking	5.86	1.70
Total train (s)	188	47.78
Total predict (s)	12.35	

- Heterophily keeps gaining accuracy as priors scale to **~250k** synthetic graphs.

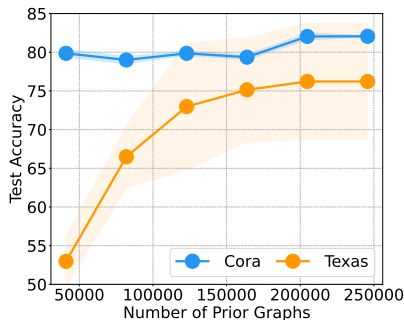


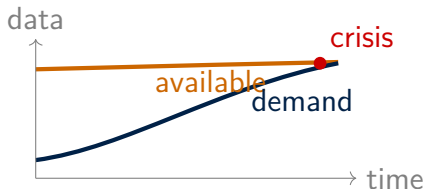
Figure: Accuracy vs. prior scale

Outline

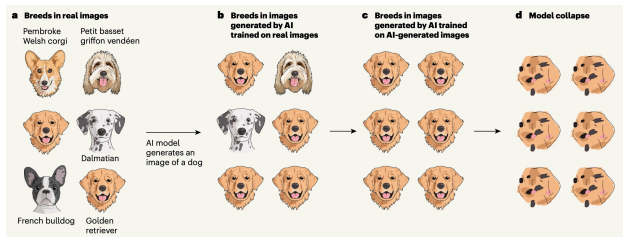
- 1 Introduction
- 2 Background: A Bayesian View
- 3 Method: NodePFN
- 4 Experiments
- 5 Conclusion**

Discussion & Future Work

- Limitations
 - fixed maximum #classes and feature dimension;
 - attention is $\mathcal{O}(N^2)$ $\rightarrow$ hard to scale to very large graphs.
- Future Work
 - **efficient / linear attention** for massive graphs.
 - **data-scarce domains:** learning from synthetic priors (not real data) suggests a path for settings where data is locked or being exhausted^[21].
 - **overcoming model collapse:** AI-generated data may cause model collapse^[22].



Running out of data



[21] [Katrina Krämer](#). “AI & robotics briefing: The year machines will run out of data.”. In: *Nature* (2024).

[22] [Iliia Shumailov et al.](#) “AI models collapse when trained on recursively generated data”. In: *Nature* 631.8022 (2024), pp. 755–759.

One Sequential Recommendation Model Pretrained from Synthetic Priors Predicts Multiple Datasets

Woosung Kang

wskang@kaist.ac.kr

Korea Advanced Institute of Science
& Technology

Daejeon, Republic of Korea

Jiwon Jeong

zzioni@kaist.ac.kr

Korea Advanced Institute of Science
& Technology

Daejeon, Republic of Korea

Jonghyeok Shin

jonghyeok@kaist.ac.kr

Korea Advanced Institute of Science
& Technology

Daejeon, Republic of Korea

Jeongwhan Choi

jeongwhan.choi@kaist.ac.kr

Korea Advanced Institute of Science
& Technology

Daejeon, Republic of Korea

Noseong Park*

noseong@kaist.ac.kr

Korea Advanced Institute of Science
& Technology

Daejeon, Republic of Korea

Reference I

- [1] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.
- [2] Runjin Chen et al. “Llaga: Large language and graph assistant”. In: *arXiv preprint arXiv:2402.08170* (2024).
- [3] Yash Deshpande et al. “Contextual stochastic block models”. In: *Advances in neural information processing systems* 31 (2018).
- [4] Paul Erds and Alfréd Rényi. “On random graphs I”. In: *Publ. math. debrecen* 6.290-297 (1959), p. 18.
- [5] Aditya Grover and Jure Leskovec. “Node2vec: Scalable Feature Learning for Networks”. In: *KDD*. 2016, pp. 855–864.
- [6] Paul W Holland et al. “Stochastic blockmodels: First steps”. In: *Social networks* 5.2 (1983), pp. 109–137.

Reference II

- [7] Noah Hollmann et al. “TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second”. In: *ICLR*. 2023.
- [8] Katrina Krämer. “AI & robotics briefing: The year machines will run out of data.”. In: *Nature* (2024).
- [9] Yuhan Li et al. “Glbench: A comprehensive benchmark for graph with large language models”. In: *NeurIPS* (2024).
- [10] Hao Liu et al. “One For All: Towards Training One Graph Model For All Classification Tasks”. In: *ICLR*. 2024.
- [11] Samuel Müller et al. “Transformers Can Do Bayesian Inference”. In: *ICLR*. 2022.
- [12] Ryoma Sato. *Training-free Graph Neural Networks and the Power of Labels as Features*. 2024.

Reference III

- [13] Ilia Shumailov et al. “AI models collapse when trained on recursively generated data”. In: *Nature* 631.8022 (2024), pp. 755–759.
- [14] Jian Tang et al. “LINE: Large-Scale Information Network Embedding”. In: *TheWebConf (former WWW)*. 2015, pp. 1067–1077.
- [15] Felix Wu et al. “Simplifying graph convolutional networks”. In: *ICML*. 2019.
- [16] Ruosong Ye et al. “Language is all a graph needs”. In: *arXiv preprint arXiv:2308.07134* (2023).
- [17] Jianan Zhao et al. “Fully-inductive Node Classification on Arbitrary Graphs”. In: *ICLR*. 2025.
- [18] Jianan Zhao et al. “Graphtext: Graph reasoning in text space”. In: *arXiv preprint arXiv:2310.01089* (2023).
- [19] Xiaojin Zhu and Zoubin Ghahramani. “Learning from labeled and unlabeled data with label propagation”. In: *Technical Report CMU-CALD-02-107*. 2002.

Thank You!

Questions?

Homepage: jeongwhanchoi.com

Email: jeongwhan.choi@kaist.ac.kr



Homepage