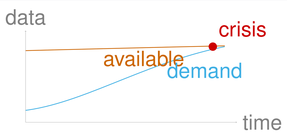
Learning Posterior Predictive Distributions for Node Classification from Synthetic Graph Priors

Jeongwhan Choi*, Jongwoo Kim*, Woosung Kang, Noseong Park
Korea Advanced Institute of Science & Technology (KAIST)

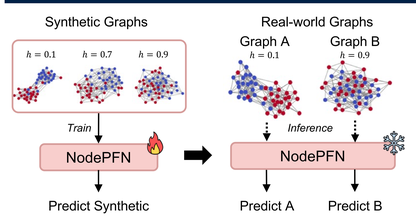
The Data Crisis Motivates Synthetic Priors

- **Healthcare, semiconductors, finance:** data locked by privacy law, trade secrets, cost.
- Public data may be **exhausted this decade** (Krämer, *Nature* 2024); training on AI-generated data triggers **model collapse** (Shumailov et al., *Nature* 2024).



Solution: learn from **mathematical priors** instead of scarce real data.

Why GNNs Cannot Generalize Across Graphs



- **GNNs are graph-specific:** trained on one graph, they can't transfer to another.
- **Graphs are heterogeneous:** homophily and features vary wildly across datasets.
- **LLM-based GFMs** rely on *text-attributed* graphs.
- **Inductive GFMs** (e.g., GraphAny) still require source-graph training — performance varies with the source.

Q: Can a *single* model classify nodes on *any* graph, with *no* real training data?

Background: PFNs Learn PPDs from Priors

Posterior Predictive Distribution — the Bayesian-optimal predictor for a query $\mathbf{x}_{\text{test}}$ given context $\mathcal{D}_{\text{train}}$:

$$p(\mathbf{y}_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}}) = \int p(\mathbf{y}_{\text{test}} \mid \mathbf{x}_{\text{test}}, \phi)\, p(\phi \mid \mathcal{D}_{\text{train}})\, d\phi$$

Prior-data Fitted Network (PFN) (Müller et al., 2022) — replaces the intractable integral with a Transformer f_θ trained on synthetic tasks $\mathcal{D} \sim p(\mathcal{D})$:

$$f_\theta : (\mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}}) \mapsto p(\mathbf{y}_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}})$$

At test time: **in-context learning** on a new dataset in *one forward pass* – no gradient updates.

Our Idea: design *graph* priors $p(\mathcal{G})$ so PFN learns PPDs for ***node classification*** across arbitrary graphs.

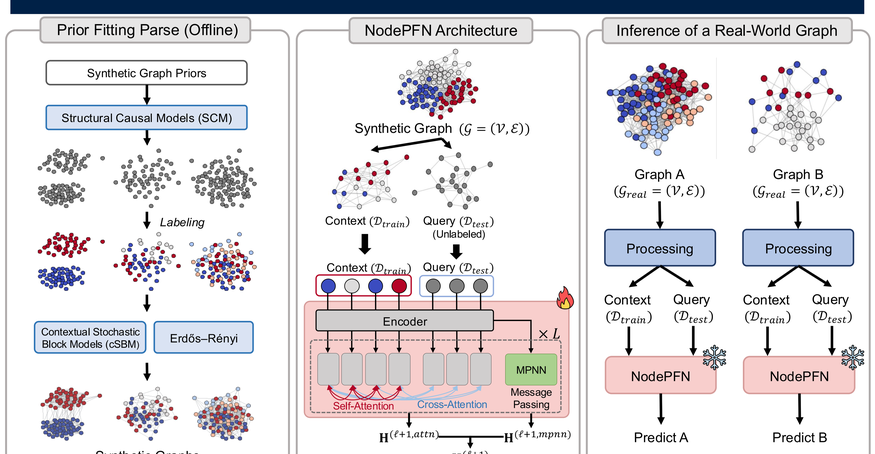
Synthetic Graph Priors from Random Networks

(a) (b) (c) (d)

Figure: Network examples with various graph priors used in NodePFN training. (a) Sparse Erdős–Rényi (ER). (b) Dense ER. (c) Low homophily cSBM. (d) High homophily cSBM. For simplicity, class color-coded nodes are not shown in ER.

One Model, Pretrained on Synthetic Data from Mathematical Priors.

Random graphs generate the pretraining data.
A **single checkpoint** predicts on any real graph.



Open Review .net

NodePFN Wins Without Per-Graph Training

	MLP	GCN	GAT	GraphAny†	NodePFN
Homophily (13)	56.43	73.05	74.39	71.45	**77.39**
Heterophily (10)	58.17	58.84	59.11	61.62	**65.14**
Avg. Acc.	57.30	66.63	67.67	66.24	**71.27**
Avg. Rank	7.41	5.86	5.57	4.38	**1.70**

†Best of 4 GraphAny variants (each trained on a different source graph).

Controlled Synthetic Graphs & Comparison with Training-free Models

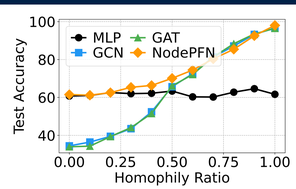


Figure: Synthetic Cora across h.

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

Table: Training-free models vs NodePFN.

Structural Node Classification & Prior Scaling

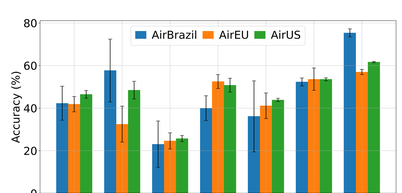


Figure: Airport datasets (one-hot ID features). NodePFN learns **topological roles** without informative features.

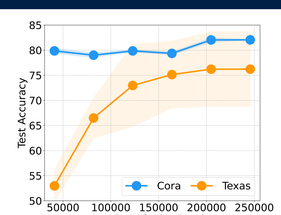


Figure: Heterophily keeps gaining accuracy as priors scale to **250k** synthetic graphs.

Matches LLM-based Graph Foundation Models

Method	Cora	Citeseer	Pubmed	WikiCS
InstructGLM	69.10	51.87	71.26	45.73
GraphText	76.21	59.43	75.11	67.35
LLaGA	74.42	55.73	68.82	73.88
OFA	75.24	**73.04**	**75.61**	77.34
NodePFN	**76.38**	63.08	68.18	**76.29**

Table: Accuracy under the supervised setting of GLBench

- Text-attributed graphs are *not required*: NodePFN works with purely numerical features.
- Competitive with **LLM-based GFMs** — without any text, language model, or pretrained embeddings.

TAKE-HOME MESSAGE

If you can encode your domain as a mathematical prior, this recipe is worth a try.