Node4All: Learning Node Representation Beyond Datasets

Dooho Lee [1] **Jaemin Yoo** [2]

Abstract

Node representation learning on graphs has advanced rapidly, yet most methods still require per-dataset training and hyperparameter tuning, limiting their reusability across diverse graph datasets. We introduce Node4All, a node representation learner *applicable to arbitrary graph datasets without dataset-specific optimization*. Node4All is built on two ideas. First, we introduce the Channel Graph Transformer (CGT), which enables a single model to process arbitrary graph datasets. Second, we propose a self-supervised learning framework on synthetic graphs that enables generalization across arbitrary datasets. We evaluate Node4All on node classification across 25 benchmarks against 21 baselines. Despite using a single fixed model without per-dataset tuning, Node4All ranks 5th overall. With an appropriate predictor, it also supports one-shot and in-context learning, outperforming recent graph foundation models. These results show that Node4All is not only reusable across arbitrary graph datasets but also effective in practice. Code and model checkpoints are available in this link.

1. Introduction

A graph is a data structure consisting of a set of nodes V and edges E connecting pairs of nodes, naturally representing many real-world relational systems such as social networks, molecules, knowledge graphs, and biological interactions (Leskovec & Krevl, 2014; Hu et al., 2020a; Wang et al., 2017). However, this flexibility also makes it challenging to design a unified framework that generalizes across graphs, since graphs can have different numbers of nodes $N = |V|$, and their connectivity patterns in E can vary widely.

[1]School of Electrical Engineering, KAIST, Daejeon, Republic of Korea [2]Department of Computer Science and Engineering, Seoul National University, Seoul, Republic of Korea. Correspondence to: Jaemin Yoo <jaeminyoo@snu.ac.kr>.

The Workshop on Graph Foundation Models at the 43rd International Conference on Machine Learning (ICML 2026), Seoul, South Korea. Copyright 2026 by the author(s).

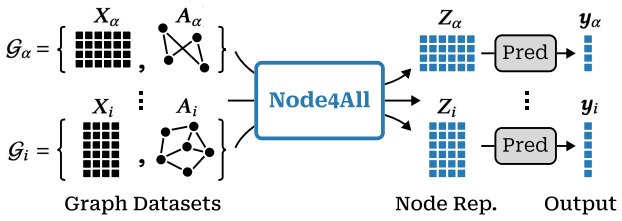

Figure 1. Overview. We propose Node4All, a single reusable node representation learner that can be applied to arbitrary graph datasets without any per-dataset modification.

Graph representation learning (GRL) aims to address this challenge by mapping a graph to a set of compact node representations $\{\boldsymbol{z}_i\}_{i=1}^{N}$ in a shared latent space $\boldsymbol{z}_i \in \mathbb{R}^H$ where H denotes the representation dimensionality (Chen et al., 2020a; Ju et al., 2024; Khoshraftar & An, 2024). These representations provide a standardized interface of graph-structured data and are commonly used as inputs to downstream predictors. As their quality directly influences downstream performance, learning node representations that effectively capture both structural and feature information has remained a central research problem for over a decade.

Early GRL methods learn node embeddings by factorizing structural proximity matrices (Anderson Jr & Morley, 1985; Roweis & Saul, 2000; Ou et al., 2016) or modeling random-walk co-occurrence patterns with skip-gram objectives (Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016). While effective, these methods produce inherently *node-bound* representations, where embeddings are tied to node identities and newly added nodes require full re-optimization. Graph neural networks (GNNs) address this limitation by learning shared message-passing functions over node features and local graph structure rather than fixed node identities (Defferrard et al., 2016; Hamilton et al., 2017; Xu et al., 2018a; Veličković et al., 2018; Wu et al., 2019; Brody et al., 2021). Although GNNs achieve strong performance across diverse benchmarks (Luo et al., 2024; 2025), they are typically optimized for specific supervised tasks, making their representations inherently *label-bound* and often requiring re-optimization for new label sets.

Recently, self-supervised learning (SSL) on graphs has been actively explored. SSL-based methods remove the reliance on labels by learning representations from objectives defined directly on the input graph. Contrastive approaches learn invariant representations by maximizing agreement between

different augmented views (Thakoor et al., 2021; Hassani & Khasahmadi, 2020; Zhang et al., 2021), while generative approaches learns from reconstructing masked or perturbed graph structures and node features (Kipf & Welling, 2016; Hou et al., 2022; Li et al., 2023). As a result, the learned representations become reusable across label spaces and often outperform supervised GNNs.

Despite these advances, most graph representation learning methods, from early random-walk–based approaches to recent SSL methods, remain largely *dataset-bound*. They are trained separately for each target dataset and require dataset-specific architectural choices and hyperparameter tuning to achieve strong performance. Moreover, models trained on one dataset are typically not applicable to graphs with different feature specifications or topologies. As a result, users must repeat training and tuning for every new dataset, a process that is time-consuming, expertise-intensive, and often dependent on labeled data for reliable validation.

In this work, we propose *Node4All*, a node representation learner applicable to arbitrary graph datasets without restrictions on graph topology or feature specification, as illustrated in Figure 1. Node4All eliminates dataset-specific optimization, including training and hyperparameter tuning, requiring only a single inference pass to obtain node representations. Node4All is composed of two key components that address complementary aspects of the problem:

1. **Architecture.** We propose *Channel Graph Transformer (CGT)*, a novel architecture that can process arbitrary graphs under a single fixed parameterization.
2. **Learning.** We propose a self-supervised learning using synthetic graphs that mimic real-world structures, enabling generalization beyond individual datasets.

We evaluate Node4All on node classification across 25 benchmarks spanning diverse feature and structural properties. We compare against 21 supervised and self-supervised baselines each optimized for individual datasets. Despite being applied uniformly across all benchmarks without any modification, Node4All ranks 5th overall among 21 baselines. Moreover, with an appropriate predictor, Node4All supports one-shot and in-context learning, outperforming recent graph foundation models (GFMs) in these settings. These results demonstrate that Node4All not only achieves reusability across arbitrary graph datasets, but also generalizes well and remains effective in practice.

2. Preliminaries

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $N = |\mathcal{V}|$ nodes. Its adjacency matrix is $\boldsymbol{A} \in \{0,1\}^{N\times N}$, where $\boldsymbol{A}_{ij} = 1$ if and only if $(i,j) \in \mathcal{E}$. Node features are given by $\boldsymbol{X} \in \mathbb{R}^{N\times F}$, where each row $\boldsymbol{X}_{n,:} \in \mathbb{R}^F$ represents the feature vector of the n-th node with dimensionality F, and each column $\boldsymbol{X}_{:,f} \in \mathbb{R}^N$ represent the f-th feature channel across all N nodes.

Graph Representation Learning. Given a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ with N nodes, we formalize graph representation learning as learning a function Ψ that maps the graph to H-dimensional node representations $\{\boldsymbol{z}_i\}_{i=1}^N$, collected as a matrix $\boldsymbol{Z} = \Psi(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N\times H}$:

$$\Psi : \mathbb{R}^{N\times F} \times \{0,1\}^{N\times N} \to \mathbb{R}^{N\times H}.$$

Our goal is to learn a representation function Ψ that can process graphs with arbitrary numbers of nodes ($\forall N$) and feature dimensions ($\forall F$) within a single fixed parameterization. The resulting embeddings $\boldsymbol{Z}$ should jointly encode features and graph structure, serving as sufficient node representations for downstream tasks.

Graph Neural Networks. Given a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ with N nodes, a graph neural network (GNN) iteratively updates node representations $\boldsymbol{Z}^{(l)} \in \mathbb{R}^{N\times F_l}$ from the input features $\boldsymbol{Z}^{(0)} = \boldsymbol{X}$, where F_l is the dimensionality at layer l. Each layer follows a message-passing scheme (Gilmer et al., 2017; Xu et al., 2018b; Hu et al., 2020b):

$$\boldsymbol{Z}^{(l)} = \mathrm{Comb}^{(l)}\Big(\mathrm{Agg}^{(l)}(\boldsymbol{Z}^{(l-1)}, \boldsymbol{A}),\ \boldsymbol{Z}^{(l-1)}\Big), \quad (1)$$

where $\mathrm{Agg}^{(l)}$ aggregates the neighboring representations using a matrix $\hat{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$ derived from adjacency matrix $\boldsymbol{A}$, either statically (Kipf & Welling, 2017; Hamilton et al., 2017) or dynamically (Veličković et al., 2018; Chien et al., 2021), and $\mathrm{Comb}^{(l)}$ integrates the aggregated messages with the previous representations, typically via a learnable linear transformation $\boldsymbol{W}^{(l)} \in \mathbb{R}^{F_{l-1}\times F_l}$.

3. Limitations of Existing Works

We first analyze three common assumptions underlying most graph representation learning methods that limit their applicability and generalization beyond individual datasets.

L1. Fixed Feature Dimensionality. Most existing methods instantiate the representation function Ψ using GNNs. These models encode information through feature transformations parameterized by weight matrices $\boldsymbol{W} \in \mathbb{R}^{F\times H}$, where F denotes the input feature dimensionality and H the hidden dimension. This design inherently couples the model parameterization to a fixed input dimensionality F. In practice, however, graph datasets exhibit substantial variation in feature dimensionality, ranging from only a few dimensions (Platonov et al., 2023) to several thousand (Shchur et al., 2018). Consequently, models trained on a specific feature dimensionality cannot be directly transferred to datasets with different dimensionalities $\tilde{F} \neq F$, preventing a single trained model from being applied across datasets.

L2. Optimized on a Specific Dataset. Most graph representation learning methods are trained and optimized on a

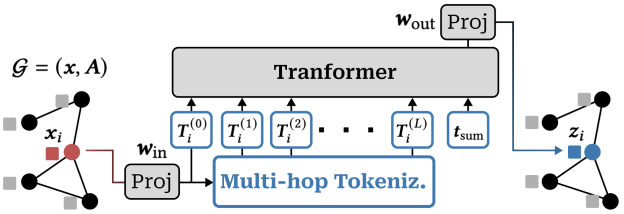

Figure 2. The Channel Graph Transformer (CGT) maps each channel graph to scalar node embeddings via multi-hop tokenization and Transformer-based token readout.

specific target dataset. This dataset-centric optimization ties the learned representations to dataset-specific regularities, including both structural patterns and feature distributions. As a result, such models often struggle to generalize to nodes or graphs that are rare or differ from those observed during training (Yehudai et al., 2021; Mao et al., 2023; Li et al., 2025). Although target-dataset optimization is effective for in-dataset performance, it inherently limits the reuse of learned representations across arbitrary graph datasets.

L3. Specialized to the Target Task. In supervised learning, representations are optimized to separate nodes with different labels and collapse nodes with the same label. When transferred to a task with a new label definition, they may no longer align with the new semantics, leading to degraded performance. Although SSL avoid explicit label supervision, many methods remain task-dependent in practice. Contrastive methods, for example, assume that differently augmented views preserve the same semantic meaning. This assumption can fail depending on the task and dataset, and effective augmentations are often selected using target-task validation performance. Thus, such indirect task guidance ties learned representations to the target task, limiting generalization across tasks and datasets.

Together, we identify three obstacles that limit generalization beyond datasets: what they train **L1**, where they train **L2**, and how they train **L3**. In Sections 4 and 5, we introduce the *Node4All* components that address these challenges.

4. Processing Arbitrary Graphs

In this section, we introduce channel-wise graph processing and its realization, the *Channel Graph Transformer (CGT)*, which handles graphs with arbitrary feature dimensionality under a single parameterization, thereby addressing **L1**.

4.1. Channel-wise Graph Processing

Our key idea is to treat node features as a collection of independent feature channels rather than as a single vector. By processing each channel with a shared model, the architecture naturally supports arbitrary feature dimensionalities under a single parameterization. To formalize this, we define a *channel graph* as a graph induced by a single feature channel with shared node and edge sets.

Definition 4.1 (Channel Graph)**.** Given a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ with node features composed of F scalar channels, $\boldsymbol{X} = \{\boldsymbol{X}_{:,c} = \boldsymbol{x}^{(c)} \in \mathbb{R}^N\}_{c=1}^F$. The *channel graph* corresponding to channel c is defined as

$$\mathcal{G}^{(c)} = (\boldsymbol{x}^{(c)}, \boldsymbol{A}), \quad c = 1, \ldots, F.$$

Thus, a graph $\mathcal{G}$ can be regarded as a collection of channel graphs $\{\mathcal{G}^{(c)}\}_{c=1}^F$ that share the same topology. This viewpoint leads to *channel-wise graph processing*, where the same model is applied uniformly to each channel graph.

Definition 4.2 (Channel-wise Graph Processing)**.** Given a graph $\mathcal{G} = \{\mathcal{G}^{(c)}\}_{c=1}^F$, *channel-wise graph processing* defines a representation function Ψ by applying a shared model f to each channel graph and concatenating the results along feature dimension: Given a graph $\mathcal{G} = \{\mathcal{G}^{(c)}\}_{c=1}^F$,

$$\begin{aligned}\boldsymbol{Z} = \Psi(\boldsymbol{X}, \boldsymbol{A}) &\triangleq \left[f(\mathcal{G}^{(1)}) \parallel \cdots \parallel f(\mathcal{G}^{(F)})\right] \\ &= \left[\boldsymbol{z}^{(1)} \parallel \cdots \parallel \boldsymbol{z}^{(F)}\right] \in \mathbb{R}^{N \times F},\end{aligned}$$

where the shared model $f : \mathbb{R}^N \times \{0,1\}^{N \times N} \to \mathbb{R}^N$ maps each channel graph to a scalar node embedding $\boldsymbol{z}^{(c)} \in \mathbb{R}^N$.

This formulation allows f to process graphs with arbitrary feature dimensionality F, thereby addressing **L1**. We next discuss which architecture should be used to realize f.

4.2. Channel Graph Transformer (CGT)

A standard GNN is suboptimal for processing channel graphs because most of its representational power comes from parametric feature transformations that mix multiple feature channels. This capability is largely lost when the input is a channel graph as each node is associated with only a one-dimensional signal. To overcome this limitation, we propose the Channel Graph Transformer (CGT). As shown in Figure 2, CGT *(i)* encodes each node in the channel graph as a sequence of multi-hop tokens and *(ii)* employs a Transformer (Vaswani et al., 2017) to aggregate these tokens into an updated scalar embedding. This design gives CGT substantially greater expressive capacity for learning on channel graphs than conventional GNNs.

Multi-hop Tokenization A Transformer requires fixed-dimensional tokens that carry meaningful information on their own, such as image patches for computer vision (Dosovitskiy et al., 2021) or word tokens for language (Vaswani et al., 2017). In graphs, however, a node feature alone does not encode structural information, since such information is defined through interactions with its neighborhood. CGT constructs standardized and structurally informative tokens through *coordinate-wise multi-hop aggregation*.

Given a channel graph $\mathcal{G} = (\boldsymbol{x}, \boldsymbol{A})$, we first map the scalar node signal to an initial d-dimensional token

$$\boldsymbol{T}^{(0)} = \boldsymbol{x}\boldsymbol{w}_{\text{in}} \in \mathbb{R}^{N \times d}, \qquad \boldsymbol{w}_{\text{in}} \in \mathbb{R}^{1 \times d}.$$

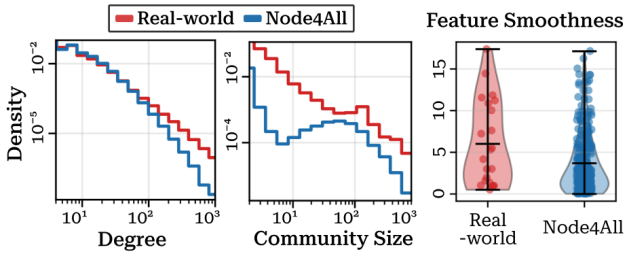

Figure 3. Comparison of 25 real-world graph datasets and 300 Node4All synthetic graphs. Degree and Louvain community-size distributions show similar trends, and average feature smoothness spans a comparable range across datasets.

We then assign each coordinate of the token a distinct receptive-field size. At each CGT layer l, the i-th coordinate is propagated by exactly i hops over the graph, and the resulting representations are combined to form a new d-dimensional token $\boldsymbol{T}^{(l)} \in \mathbb{R}^{N\times d}$:

$$\boldsymbol{T}^{(l)} = g^{(l)}\Big(\big[\hat{\boldsymbol{A}}^0\boldsymbol{T}^{(l-1)}_{:,0} \| \hat{\boldsymbol{A}}^1\boldsymbol{T}^{(l-1)}_{:,1} \| \cdots \| \hat{\boldsymbol{A}}^{d-1}\boldsymbol{T}^{(l-1)}_{:,d-1}\big]\Big),$$

where $\hat{\boldsymbol{A}}$ denotes the symmetrically normalized adjacency matrix. The mixing function $g^{(l)} : \mathbb{R}^{N\times d} \to \mathbb{R}^{N\times d}$ is implemented using a lightweight multi-layer perceptron (MLP) for simplicity. Our tokenization method ensures that each token has a consistent dimensionality d while embedding rich neighborhood information of each node from local to global scales. Repeating this for L layers yields a series of tokens $\boldsymbol{T}^{(0)}, \boldsymbol{T}^{(1)}, \ldots, \boldsymbol{T}^{(L)}$, which serve as the input sequence to the Transformer.

Transformer-based Token Readout CGT then aggregates the token sequence of each node into a single representation using a Transformer. For each node $i = 1, \ldots, N$, we first prepend a learnable summary token $\boldsymbol{t}_{\text{sum}} \in \mathbb{R}^d$:

$$\boldsymbol{S}_i^{(0)} = \big[\boldsymbol{t}_{\text{sum}},\ \boldsymbol{T}_i^{(0)},\ \boldsymbol{T}_i^{(1)},\ \cdots,\ \boldsymbol{T}_i^{(L)}\big] \in \mathbb{R}^{(L+2)\times d}.$$

The sequence is processed by an R-layer Transformer, whose self-attention updates the summary token by attending over entire token set. The final summary-token representation is then projected to a scalar embedding:

$$f(\mathcal{G})_i = \boldsymbol{S}_{i,0}^{(R)\top}\boldsymbol{w}_{\text{out}} \in \mathbb{R}, \qquad \boldsymbol{w}_{\text{out}} \in \mathbb{R}^d.$$

Together, CGT serves as the function f for channel-wise graph processing, with sufficient expressive capacity to encode structural information in the channel graph.

5. Learning for Arbitrary Graphs

While our CGT enables the processing of arbitrary graphs, learning representations that generalize across datasets remains challenging, which relates to limitations **L2** and **L3**. To address these limitations, we propose a new SSL framework based on a series of synthetic graphs.

5.1. Synthetic Graph Generation

Most graph models are trained on only one or a few real-world datasets and therefore tend to fit dataset-specific patterns (**L2**). To improve cross-dataset generalization, we train the model on synthetic graphs instead of limited real-world datasets, exposing it to a wide range of graph distributions. This encourages the model to focus on structural and feature patterns that persist across datasets. Due to space limitations, we briefly describe the generation procedures below and full formulations are provided in Appendix A.

Given that synthetic graph generation admits many possible design choices, we intentionally keep the design simple to avoid introducing unnecessary inductive biases while still covering a broad range of graph distributions. We define a synthetic graph generator as a probability distribution over graphs $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ parameterized by latent variables $\boldsymbol{z}$:

$$p(\mathcal{G}) = \int p(\boldsymbol{z})\, p(\boldsymbol{A} \mid \boldsymbol{z})\, p(\boldsymbol{X} \mid \boldsymbol{A}, \boldsymbol{z})\, d\boldsymbol{z}, \tag{2}$$

A graph $\mathcal{G} \sim p(\mathcal{G})$ is generated by a three-stage process: (i) sampling latent variables $\boldsymbol{z}$, (ii) generating the graph structure $\boldsymbol{A}$ conditioned on $\boldsymbol{z}$, and (iii) generating node features $\boldsymbol{X}$ conditioned on both $\boldsymbol{A}$ and $\boldsymbol{z}$.

Structure generation. We generate graph structures using a Chung–Lu–style random graph model (Chung & Lu, 2002), where latent variables control graph sparsity and degree heterogeneity. By varying these variables, the generator produces diverse structures ranging from nearly regular to heavy-tailed graphs, yielding degree distributions and community sizes similar to those observed in real-world graph datasets (see Figure 3).

Feature generation. Node features are first generated from low-dimensional semantic factors and projected into the observed feature space through random linear mixing. This procedure enables control over intrinsic dimensionality while producing diverse feature patterns. Since independently generated features contain limited graph-aligned information, we further apply stochastic graph propagation to introduce correlations between node features and graph structure. By varying the propagation depth and mixing behavior, the generator produces features with diverse levels of local smoothness, spanning a range comparable to that observed in real-world graph datasets (Figure 3).

5.2. Masked Autoencoder with DropNode

Having specified what to train (CGT) and where to train it (synthetic graphs), we now describe how we train it. We adopt *masked node-feature reconstruction*, following GraphMAE (Hou et al., 2022; 2023), which learns representations by reconstructing masked node features. This self-supervised objective relies on a weak and broadly ap-

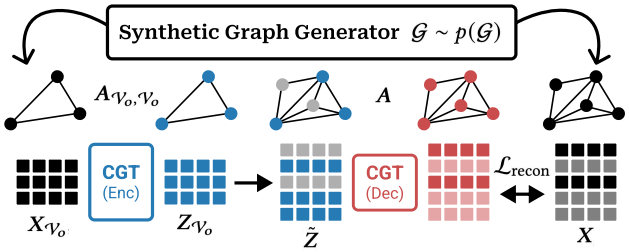

Figure 4. Node4All trains CGT via masked autoencoding on a stream of synthetic graphs to learn node representations that generalize to arbitrary real-world graph datasets.

plicable assumption: node attributes are at least partially predictable from their surrounding context. If this assumption were violated, it would imply that nodes in the graph are largely independent of one another, calling into question whether the graph structure encodes meaningful relational information at all. Such scenarios are uncommon in practical graph datasets, making masked reconstruction a robust objective that mitigates task-specific bias (**L3**).

However, directly applying GraphMAE is not ideal. Unlike prior work that assumes a single dataset with a fixed feature space, our training data consists of multiple synthetic graphs, each with potentially different feature spaces. Thus, replacing masked nodes with a single shared learnable token, e.g., `[MASK]`, can lead to inconsistent semantics across graphs and introduce misleading shortcuts during training.

We therefore adopt a DropNode-style (You et al., 2020; Feng et al., 2020) masked autoencoder, which drops nodes instead of replacing them with mask tokens. Given a synthetic graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ with node set $\mathcal{V}$, we randomly select nodes to drop, $\tilde{\mathcal{V}} \subset \mathcal{V}$, and denote the remaining nodes by $\mathcal{V}_o = \mathcal{V} \setminus \tilde{\mathcal{V}}$. The encoder then operates on the induced subgraph after removing dropped nodes and their incident edges, producing embeddings for the remaining nodes:

$$\boldsymbol{Z}_{\mathcal{V}_o} = f_{\text{enc}}(\boldsymbol{X}_{\mathcal{V}_o},\ \boldsymbol{A}_{\mathcal{V}_o,\mathcal{V}_o}), \qquad \tilde{\boldsymbol{Z}}_i = \begin{cases} \boldsymbol{Z}_i, & i \in \mathcal{V}_o, \\ \boldsymbol{0}, & i \in \tilde{\mathcal{V}}. \end{cases}$$

For reconstruction, we feed the decoder the full graph structure with a node representation matrix $\tilde{\boldsymbol{Z}} \in \mathbb{R}^{N \times D}$, where representations from the encoder are used for observed nodes and zero vectors are assigned to dropped nodes. We optimize the autoencoder f_{enc} and f_{dec} using a cosine reconstruction loss over the dropped nodes:

$$\mathcal{L}_{\text{recon}} = \frac{1}{|\tilde{\mathcal{V}}|} \sum_{i \in \tilde{\mathcal{V}}} \big(1 - \cos\big(f_{\text{dec}}(\tilde{\boldsymbol{Z}}, \boldsymbol{A})_i,\ \boldsymbol{X}_i\big)\big).$$

Node4All. Finally, we introduce *Node4All*, a node representation learner that integrates three components: the Channel Graph Transformer (CGT; Section 4.2), synthetic graph generation (Section 5.1), and masked autoencoding (Section 5.2), as illustrated in Figure 4. Node4All employs CGT as both the encoder f_{enc} and decoder f_{dec}, allowing it to handle graphs with arbitrary feature spaces. During training, a synthetic graph generator produces diverse graphs, and the model learns by reconstructing dropped node features from their surrounding graph context. By repeating this reconstruction across heterogeneous graph structures and feature spaces, Node4All captures generalizable patterns rather than dataset-specific ones, enabling node representation learning to generalize beyond individual datasets.

6. Related Works

Node4All is a node representation learner that shares the goal with graph representation learning methods, which we review in detail in Section 1. Here, we instead focus on another related line of work, graph foundation models (GFMs), which shares the core motivation; generalizing across multiple graph datasets within a single model.

Text-space GFMs The first direction unifies multiple graph datasets by operating in a common text space. These methods leverage pretrained large language models (LLMs) to embed raw text features (Liu et al., 2024; Chen et al., 2025) or to make predictions directly within them (Tang et al., 2024; Ye et al., 2024; Chen et al., 2024). They have shown promising results in handling task heterogeneity, co-training on multiple datasets, transfer learning, and few-shot learning (Liu et al., 2024; Chen et al., 2024; He et al., 2025; Fang et al., 2025). However, many graph datasets contain numerical or categorical attributes, and converting them into text is often infeasible or relies on dataset-specific templates.

Feature-aligned GFMs The second direction extends generalization to graphs with non-textual attributes. To address feature heterogeneity across datasets (**L1**), these approaches project node features into a shared representation space via singular value decomposition with alignment modules (Yu et al., 2025; Zhao et al., 2024) or learnable patching mechanisms (Sun et al., 2025), and have shown promising results in co-training and few-shot adaptation. However, they typically require per-dataset fine-tuning when adapting to new graphs and are less effective when sufficient labels are available (Sun et al., 2025; Yu et al., 2025; Zhao et al., 2024).

Fully Inductive GFMs The third direction aims to develop fully inductive GFMs that can operate on arbitrary graph datasets without training on target data (Zhao et al., 2025). GraphAny achieves full inductiveness by attending over a fixed set of least-mean-square linear predictors derived from features propagated at different scales. TS-GNNs (Finkelshtein et al., 2025) further generalizes this framework by enforcing equivariance to node and feature permutations and invariance to labels. NodePFN (Choi et al., 2026) pursues a related goal by extending tabular foundation models, such as TabPFN (Hollmann et al., 2022; 2025), to

graph-structured inputs. While Node4All also satisfies full inductiveness, it differs in objective: it transforms graphs into node representations, whereas these models perform direct prediction via in-context learning.

Relationship with GFMs Node4All differs from existing GFMs in two key aspects. First, we target full inductive applicability across arbitrary graph datasets. This is difficult for text-space GFMs due to their reliance on textual attributes, and for feature-aligned GFMs due to their dependence on per-dataset adaptation. Second, unlike most GFMs that conduct task-specific predictions, Node4All creates node representations. This enables (i) caching embeddings for reuse across different label spaces and (ii) flexible choices of downstream predictors. In our experiments, we show that both one-shot settings of feature-aligned GFMs and in-context prediction of fully inductive GFMs can be realized, and often improved upon, by selecting an appropriate predictor on top of Node4All representations.

7. Main Experiments

We evaluate the practical effectiveness of Node4All by comparison with existing methods across multiple settings. Specifically, we investigate the following questions:

- **Q1.** Can a single fixed model (Node4All) make competitive performance vs. per-dataset optimized methods?
- **Q2.** How effective are Node4All representations in one-shot learning compared to feature-aligned GFMs?
- **Q3.** How effective are Node4All representations in in-context learning compared to fully inductive GFMs?

7.1. Against Per-Dataset Optimized Methods

Node4All offers clear practical advantages, including *(i)* applicability to arbitrary graph datasets and *(ii)* the elimination of dataset-specific training and hyperparameter tuning. However, these benefits are only meaningful if performance remains competitive. If too much accuracy is sacrificed, users may instead prefer the cost of per-dataset optimization. We therefore evaluate whether a single fixed Node4All model can compete with conventional methods that are trained and optimized separately for each dataset.

Setup. We compare Node4All against 21 baselines, including 12 supervised GNNs, and 6 graph SSL methods spanning both contrastive and generative paradigms, as well as linear and MLPs predictors and node2vec (Grover & Leskovec, 2016). For all representation learning methods, including Node4All, we use the same one-hidden-layer MLP predictor to isolate representation quality. A detailed list of baselines is provided in Appendix C.

We use 25 node classification benchmarks covering diverse domains, feature specifications, and structural properties.

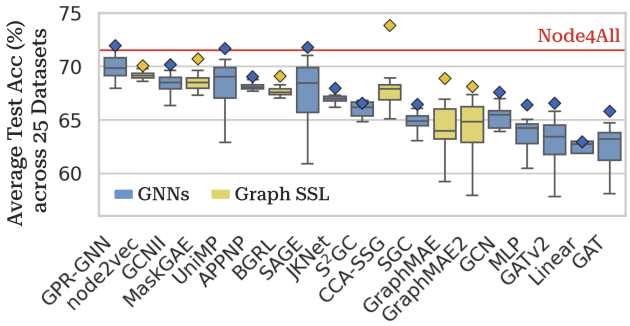

Figure 5. Box plots show the distribution of average test accuracies (%) across hyperparameter configurations for each method. Diamonds denote the performance of each method under Tuned-HP. The red line is Node4All, which exhibits no variation since it requires no dataset-specific tuning.

Dataset details are provided in Appendix G. Public splits are used when available; otherwise, following (Zhao et al., 2025), we sample 20 nodes per class for training and evenly split the remaining nodes into validation and test sets.

Results. Due to the large number of baselines and benchmarks, full results are reported in Appendix F, while we provide a summary in Figure 5. We evaluate baselines under two settings, **Fixed-HP** and **Tuned-HP**, to disentangle the effects of *(i)* per-dataset training and *(ii)* per-dataset hyperparameter tuning. All baselines are trained separately on each target dataset, whereas Node4All is applied to all datasets without *any dataset-specific training or tuning.*

In **Fixed-HP**, results are shown as box plots in Figure 5. Each method uses a single hyperparameter configuration selected to maximize its average validation accuracy across all datasets. This setting isolates the effect of per-dataset training and tests whether a method admits a generally effective configuration that transfers across datasets. Under this setting, Node4All achieves the highest average accuracy among all 21 baselines. These results show that knowledge learned from large collections of synthetic graphs can generalize across diverse datasets and even outperform methods trained separately on each target dataset. Overall, these findings suggest that when extensive per-dataset hyperparameter tuning is not affordable, Node4All provides a strong and reliable alternative to conventional per-dataset training.

In **Tuned-HP**, results are shown as diamond markers in Figure 5. Here, each baseline independently tunes hyperparameters for every dataset. Under this setting, graph SSL methods improve their average accuracy by 1.92%, while supervised GNNs baselines improve by 1.11%. As a result, Node4All's rank shifts from 1st to 5th among 21 baselines, while still remaining competitive. These results highlight that existing methods often require extensive per-dataset tuning to achieve the strongest performance. In contrast, Node4All uses a single fixed model across all datasets, providing a robust and practically attractive alternative.

Table 1. In-context learning performance (%) compared with fully inductive GFMs across 11 datasets. The last column reports the average rank (lower is better), with **best** and second-best results highlighted for each dataset.

Method	Actor	Amazon Photo	Amazon Computer	Cornell	DBLP	Deezer	Mine-sweeper	Pubmed	Tolokers	Texas	Wisconsin	Avg. Rank
GraphAny	$29.51_{\pm 0.55}$	$90.64_{\pm 0.82}$	$83.04_{\pm 1.24}$	$66.49_{\pm 1.48}$	$66.49_{\pm 1.48}$	$52.13_{\pm 3.02}$	$80.46_{\pm 0.15}$	$77.46_{\pm 0.30}$	$78.24_{\pm 0.03}$	$73.52_{\pm 2.96}$	$71.77_{\pm 5.98}$	3.90
TS-GAT	$28.80_{\pm 2.12}$	$89.66_{\pm 1.23}$	$80.23_{\pm 2.44}$	$71.35_{\pm 4.52}$	$64.30_{\pm 3.75}$	$52.88_{\pm 2.86}$	$\mathbf{80.72_{\pm 0.78}}$	$75.46_{\pm 0.86}$	$78.01_{\pm 0.23}$	$74.05_{\pm 4.10}$	$65.88_{\pm 9.86}$	4.27
TS-Mean	$28.09_{\pm 0.93}$	$90.18_{\pm 1.30}$	$81.37_{\pm 1.25}$	$68.65_{\pm 2.42}$	$66.42_{\pm 3.65}$	$52.31_{\pm 2.52}$	$80.68_{\pm 0.38}$	$74.98_{\pm 0.56}$	$78.12_{\pm 0.09}$	$73.51_{\pm 4.01}$	$61.18_{\pm 11.38}$	4.63
NodePFN	$32.99_{\pm 1.09}$	$90.53_{\pm 0.13}$	$81.42_{\pm 0.48}$	$71.89_{\pm 2.76}$	$74.71_{\pm 0.39}$	$53.45_{\pm 0.65}$	$80.66_{\pm 0.25}$	$78.00_{\pm 0.24}$	$78.61_{\pm 0.06}$	$76.22_{\pm 7.53}$	$79.22_{\pm 6.97}$	2.54
TabPFN	$32.93_{\pm 1.71}$	$80.78_{\pm 1.63}$	$67.02_{\pm 2.00}$	$\mathbf{74.05_{\pm 5.57}}$	$53.46_{\pm 2.53}$	$\mathbf{55.10_{\pm 2.48}}$	$80.00_{\pm 0.00}$	$72.20_{\pm 0.00}$	$78.36_{\pm 0.24}$	$\mathbf{82.08_{\pm 5.16}}$	$\mathbf{86.08_{\pm 3.97}}$	3.63
+ Node4All	$\mathbf{33.18_{\pm 1.15}}$	$\mathbf{91.70_{\pm 0.65}}$	$\mathbf{83.59_{\pm 0.73}}$	$66.76_{\pm 7.05}$	$\mathbf{77.67_{\pm 1.54}}$	$55.06_{\pm 1.30}$	$80.36_{\pm 0.36}$	$\mathbf{80.80_{\pm 0.00}}$	$\mathbf{81.15_{\pm 0.61}}$	$76.88_{\pm 5.40}$	$82.35_{\pm 3.62}$	**2.00**

Table 2. One-shot learning performance (%) compared with feature-aligned GFMs across 5 datasets.

Method	Cora	Citeseer	Pubmed	AmzPhoto	AmzComp
GCN	$29.53_{\pm 7.56}$	$26.29_{\pm 6.50}$	$23.32_{\pm 11.56}$	$26.96_{\pm 12.94}$	$24.40_{\pm 5.62}$
GraphPrompt	$28.26_{\pm 12.68}$	$32.51_{\pm 8.73}$	$47.47_{\pm 9.15}$	$48.11_{\pm 9.89}$	$42.82_{\pm 11.67}$
GPF	$32.17_{\pm 6.56}$	$36.79_{\pm 7.70}$	$41.28_{\pm 8.14}$	$47.47_{\pm 8.19}$	$35.75_{\pm 7.12}$
Hassani	$33.35_{\pm 6.93}$	$33.66_{\pm 7.24}$	$39.87_{\pm 8.16}$	$48.48_{\pm 7.07}$	$39.99_{\pm 7.91}$
GCOPE	$35.62_{\pm 11.93}$	$38.33_{\pm 9.28}$	$45.38_{\pm 9.87}$	$52.87_{\pm 9.19}$	$45.65_{\pm 10.69}$
SAMGPT	$47.80_{\pm 11.88}$	$36.38_{\pm 9.10}$	$50.65_{\pm 8.93}$	$58.71_{\pm 8.69}$	$48.22_{\pm 8.17}$
Ridge Reg.	$30.54_{\pm 10.63}$	$39.83_{\pm 8.15}$	$\mathbf{52.42_{\pm 8.09}}$	$52.42_{\pm 8.09}$	$38.58_{\pm 8.87}$
+ Node4All	$\mathbf{51.79_{\pm 12.99}}$	$\mathbf{44.08_{\pm 8.94}}$	$38.76_{\pm 10.51}$	$\mathbf{58.97_{\pm 9.69}}$	$\mathbf{49.72_{\pm 10.64}}$

7.2. One-Shot Learning

One-shot learning is a common evaluation setting for GFMs, where only a single labeled example per class is provided. It is designed to assess whether knowledge pretrained on other datasets can be effectively adapted to a new graph dataset under minimal supervision. We compare Node4All against recent feature-aligned GFMs, which are specifically designed for cross-dataset adaptation in this setting.

Setup. We follow the evaluation protocol of SAMGPT (Yu et al., 2025) and use their reported numbers on five datasets. Baselines include supervised GCN (Kipf & Welling, 2017), feature-aligned GFMs (GCOPE (Zhao et al., 2024), SAMGPT (Yu et al., 2025)), graph cross-domain adaptation model (Hassani (Hassani, 2022)), and graph prompting frameworks (GPF (Fang et al., 2023), GraphPrompt (Liu et al., 2023)). To adapt Node4All representations to the labels with only one-shot supervision, we first reduce their dimensionality to 32 using truncated SVD and then train a ridge regressor with a fixed regularization strength of $\alpha = 10.0$ across all datasets.

Results. We report the accuracy in Table 2. Node4All outperforms the feature-aligned GFMs GCOPE and SAMGPT on 4 out of 5 datasets, as well as other baselines. These results demonstrate that Node4All remains effective even under extreme supervision scarcity. Importantly, while the GFM baselines rely on co-training across multiple real-world datasets, Node4All is trained entirely on synthetic graphs. This supports our claim that training on well-designed synthetic graphs can generalize more effectively than training on a limited collection of real-world datasets.

7.3. In-Context Learning

In-context learning refers to making predictions without updating model parameters, instead using a small set of labeled examples as context. For GFMs, this setting evaluates whether pretrained knowledge from other datasets can be effectively applied to unseen datasets through the provided context. We compare Node4All against fully inductive GFMs designed explicitly for graph in-context learning.

Setup. We compare against three recent fully inductive GFMs: GraphAny (Zhao et al., 2025), TSNet (Finkelshtein et al., 2025), and NodePFN (Choi et al., 2026). We follow the shared evaluation protocol for Node4All, while baseline results are taken directly from the original papers.

Since Node4All does not natively support in-context learning, we pair it with TabPFN (Hollmann et al., 2022; 2025), a tabular foundation model, as an in-context predictor. Node representations from Node4All are provided to TabPFN together with the labeled nodes as context. This Node4All-TabPFN pipeline enables in-context predictions on arbitrary graphs in a single inference pass, without any training or dataset-specific modification, thereby forming a fully inductive GFM. Due to current limitations of TabPFN on feature dimensionality and the number of context samples, we evaluate this setup on 11 datasets that satisfy these constraints.

Results. As shown in Table 1, Node4All achieves the best average rank of 2.0 and the top performance on 6 datasets. These results demonstrate that Node4All representations are highly effective for in-context learning, enabling the structure-agnostic TabPFN to operate as a graph foundation model and outperform recent fully inductive GFMs. Notably, the second-best method, NodePFN, requires per-dataset selection of the PCA dimensionality and propagation depth, whereas Node4All requires no dataset-specific modification. Moreover, Node4All with TabPFN generally outperforms Node4All with the MLP predictor used in the main experiments, highlighting the promise of this combination.

8. Additional Analysis

We present further analyses of Node4All, examining its computational complexity, representational behavior via visualization, and the effects of its core design choices.

8.1. Complexity Analysis

Theoretical. Let d denote the token dimension, $(L+1)$ the token sequence length, and R the number of Trans-

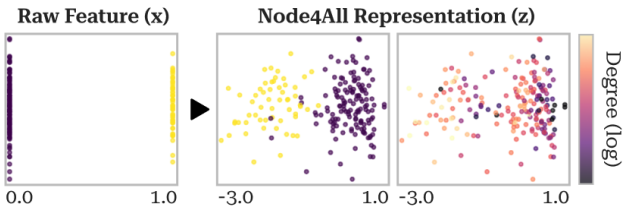

Figure 6. Scatter plots of raw features (left) and Node4All representations (middle, right) for the first feature channel of Amazon Photo, with nodes colored by degree (right).

former layers. The computational cost of CGT decomposes into multi-hop tokenization $\mathcal{O}(L(|E|d^2 + Nd^2))$, Transformer attention $\mathcal{O}(RN(L+1)^2 d)$, and feed-forward layers $\mathcal{O}(RN(L+1)d^2)$, all linear in the number of nodes N and edges $|E|$. Since CGT is applied independently to each channel graph, the overall complexity also scales linearly with the input feature dimensionality F.

Empirical. Training Node4All on an NVIDIA RTX A6000 takes about 10 minutes over 500 synthetic graphs, while inference across all 25 benchmark graphs requires 672 seconds. Since CGT processes all F feature channels independently, runtime scales linearly with feature dimensionality, leading to several-minute inference on datasets with large feature dimensionality ($F > 8000$), such as Coauthor Physics (238 s), FullCora (103 s), and BlogCatalog (90 s). Nevertheless, this remains practical and highly efficient compared to conventional methods, whose hyperparameter tuning often requires several hours or even days.

8.2. Visualization

In this section, we visualize Node4All representations to better understand how it generalizes across arbitrary graphs.

Channel-wise Processing. Node4All processes each feature channel independently, allowing direct inspection of encoding at each channel. In Figure 6, each node appears as a point with its scalar feature value on the horizontal axis, while the vertical axis is added for visualization. For binary features, many nodes initially share identical values (0 or 1; left). After encoding, these nodes map to distinct representations while preserving the original two-group separation (middle), and the resulting ordering further correlates with node degree (right). This shows how Node4All distinguishes nodes sharing the same feature value using structural information. Similar behaviors are observed for categorical and continuous features (Appendix E).

Representation Space. We next visualize the full node representations using t-SNE (Maaten & Hinton, 2008). When colored by the first principal component of the raw features, the learned representations largely preserve the original feature-driven structure. When colored by node degree, clear structural patterns emerge that are absent in the raw features. Together, these observations suggest that the channel-wise structural effects introduced by Node4All accumulate

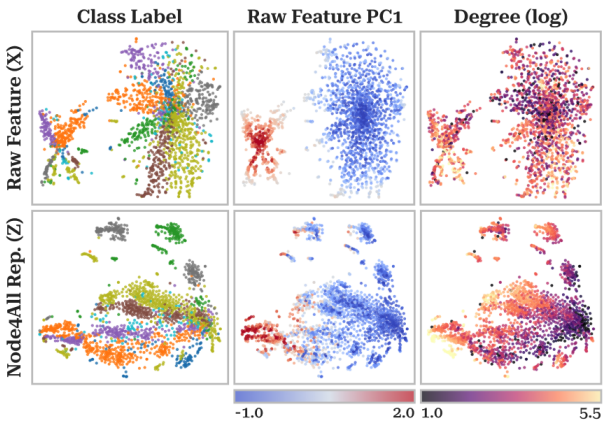

Figure 7. t-SNE visualizations of raw features and Node4All representations on Amazon Photo, colored by class labels (left), raw-feature PC1 (middle), and node degree (right).

coherently in the final representation space. Importantly, this mechanism—distinguishing nodes through graph structure while preserving feature information—relies on generic structural cues rather than dataset-specific semantics, helping explain Node4All's strong cross-dataset generalization.

8.3. Ablation Study

Due to space limitations, detailed ablations are provided in Appendix D. Replacing CGT with a GCN reduces average accuracy from 71.55% to 37.80%, highlighting the necessity of CGT for channel-wise graph processing. Training on real-world datasets instead of synthetic graphs also lowers performance, with OGBN-Arxiv achieving the average accuracy of 69.94% (9th of 21 baselines). These results indicate that both the CGT and synthetic graph pretraining are essential for Node4All's cross-dataset generalization.

9. Conclusion

We introduce Node4All, a node representation learner applicable to arbitrary real-world graph datasets without dataset-specific training or hyperparameter tuning. Node4All uses the Channel Graph Transformer (CGT), a channel-wise architecture that handles heterogeneous feature dimensionalities within a fixed parameterization. Trained once entirely on synthetic graphs, Node4All learns representations that generalize across diverse graph datasets. Empirically, a single Node4All model achieves competitive performance against per-dataset optimized methods and further outperforms recent graph foundation models in one-shot and in-context learning. Overall, Node4All advances node representation learning beyond *dataset-bound*.

Future Works Node4All focuses on node classification, and extending it to edge-level and graph-level tasks remains future work. In addition, the current synthetic graph generator may not fully capture the diversity of real-world graphs, motivating further improvements to graph generation.

Impact Statement

In this paper, we aim to advance graph representation learning beyond dataset-specific settings, primarily for scientific research. Our approach is designed to achieve cross-dataset generalization within a single fixed model, potentially benefiting a wide range of applications. We do not anticipate direct negative societal impacts, but encourage the responsible and ethical use of this work.

References

Anderson Jr, W. N. and Morley, T. D. Eigenvalues of the laplacian of a graph. *Linear and multilinear algebra*, 18 (2):141–145, 1985.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Chen, F., Wang, Y.-C., Wang, B., and Kuo, C.-C. J. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9:e15, 2020a.

Chen, H., Wang, X., Zhang, Z., Li, H., Feng, L., and Zhu, W. Autogfm: Automated graph foundation model with adaptive architecture customization. In *Forty-second International Conference on Machine Learning*, 2025.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020b.

Chen, R., Zhao, T., Jaiswal, A. K., Shah, N., and Wang, Z. Llaga: Large language and graph assistant. In *International Conference on Machine Learning*, pp. 7809–7823. PMLR, 2024.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Choi, J., Kim, J., Kang, W., and Park, N. Learning posterior predictive distributions for node classification from synthetic graph priors. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FmxRzlu0rT.

Chung, F. and Lu, L. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Fang, T., Zhang, Y., Yang, Y., Wang, C., and Chen, L. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36: 52464–52489, 2023.

Fang, Y., Fan, D., Ding, S., Liu, N., and Tan, Q. Uniglm: Training one unified language model for text-attributed graphs embedding. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 973–981, 2025.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., and Tang, J. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33: 22092–22103, 2020.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Finkelshtein, B., İsmail İlkan Ceylan, Bronstein, M., and Levie, R. Equivariance everywhere all at once: A recipe for graph foundation models, 2025.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hassani, K. Cross-domain few-shot graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6856–6864, 2022.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pp. 4116–4126. PMLR, 2020.

He, Y., Sui, Y., He, X., and Hooi, B. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Volume 1*, pp. 448–459, 2025.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 594–604, 2022.

Hou, Z., He, Y., Cen, Y., Liu, X., Dong, Y., Kharlamov, E., and Tang, J. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*, pp. 737–746, 2023.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Ju, W., Fang, Z., Gu, Y., Liu, Z., Long, Q., Qiao, Z., Qin, Y., Shen, J., Sun, F., Xiao, Z., et al. A comprehensive survey on deep graph representation learning. *Neural Networks*, 173:106207, 2024.

Khoshraftar, S. and An, A. A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55, 2024.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Li, H., Wang, X., Zhang, Z., and Zhu, W. Out-of-distribution generalization on graphs: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Li, J., Wu, R., Sun, W., Chen, L., Tian, S., Zhu, L., Meng, C., Zheng, Z., and Wang, W. What's behind the mask: Understanding masked graph modeling for graph autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1268–1279, 2023.

Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4IT2pgc9v6.

Liu, Z., Yu, X., Fang, Y., and Zhang, X. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM web conference 2023*, pp. 417–428, 2023.

Luo, Y., Shi, L., and Wu, X.-M. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems*, 37: 97650–97669, 2024.

Luo, Y., Shi, L., and Wu, X.-M. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. In *Forty-second International Conference on Machine Learning*, 2025.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Mao, H., Chen, Z., Jin, W., Han, H., Ma, Y., Zhao, T., Shah, N., and Tang, J. Demystifying structural disparity in graph neural networks: Can one size fit all? *Advances in neural information processing systems*, 36:37013–37067, 2023.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, 2016.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the*

20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701–710, 2014.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326, 2000.

Roy, O. and Vetterli, M. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pp. 606–610. IEEE, 2007.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 2021.

Sun, Y., Yang, Y., Feng, X., Wang, Z., Zhong, H., Wang, C., and Chen, L. Handling feature heterogeneity with learnable graph patches. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Volume 1*, pp. 1313–1324, 2025.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., and Huang, C. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024.

Thakoor, S., Tallec, C., Azar, M. G., Munos, R., Veličković, P., and Valko, M. Bootstrapped representation learning on graphs. In *ICLR 2021 workshop on geometrical and topological representation learning*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Wang, Q., Mao, Z., Wang, B., and Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE transactions on knowledge and data engineering*, 29(12):2724–2743, 2017.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. Pmlr, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018a.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018b.

Ye, R., Zhang, C., Wang, R., Xu, S., and Zhang, Y. Language is all a graph needs. In *Findings of the association for computational linguistics: EACL 2024*, pp. 1955–1973, 2024.

Yehudai, G., Fetaya, E., Meirom, E., Chechik, G., and Maron, H. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pp. 11975–11986. PMLR, 2021.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

Yu, X., Gong, Z., Zhou, C., Fang, Y., and Zhang, H. Samgpt: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *Proceedings of the ACM on Web Conference 2025*, pp. 1142–1153, 2025.

Zhang, H., Wu, Q., Yan, J., Wipf, D., and Yu, P. S. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.

Zhao, H., Chen, A., Sun, X., Cheng, H., and Li, J. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4443–4454, 2024.

Zhao, J., Zhu, Z., Galkin, M., Mostafa, H., Bronstein, M. M., and Tang, J. Fully-inductive node classification on arbitrary graphs. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhu, H. and Koniusz, P. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=CYO5T-YjWZV.

A. Synthetic Graph Generation Details

In this section, we provide the detailed formulation of the synthetic graph generator used to train Node4All, which was briefly described in Section 5.1.

A.1. Overview

As in Equation (2), we define the synthetic graph generator as a probability distribution over graphs $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ generated in three stages: (i) sampling latent variables $\boldsymbol{z}$, (ii) generating the adjacency matrix $\boldsymbol{A}$, and (iii) generating node features $\boldsymbol{X}$ conditioned on the graph structure. Below, we separately describe the structure and feature generation processes and the sampled variables that control them.

A.2. Structure Generation

We generate graph structures using a Chung–Lu–style random graph model (Chung & Lu, 2002). The generator is controlled by two latent variables: the expected average degree $\bar{d}$ and the degree heterogeneity parameter σ.

Given the number of nodes N, each node $u \in \{1, \dots, N\}$ is assigned an expected degree parameter:

$$\alpha_u \sim \mathcal{N}\left(-\tfrac{1}{2}\sigma^2, \sigma^2\right), \qquad w_u = \bar{d}\exp(\alpha_u).$$

This construction preserves the expected average degree, $\mathbb{E}[w_u] = \bar{d}$, while controlling degree heterogeneity through σ. Small σ produces nearly regular graphs, whereas large σ generates heavy-tailed degree distributions.

Edges are then independently sampled as

$$A_{uv} \sim \text{Bernoulli}\left(\min\left(1, \frac{w_u w_v}{W}\right)\right), \qquad W = \sum_{t=1}^{N} w_t.$$

Under this construction, $\mathbb{E}[d_u \mid \{w\}] \approx w_u$, yielding graphs with controllable sparsity and degree heterogeneity similar to those observed in real-world graph datasets.

A.3. Feature Generation

Our feature generation process is inspired by synthetic prior generation in recent tabular foundation models (Hollmann et al., 2022), while being simplified and adapted to better mimic real-world graph datasets.

Intrinsic dimensionality. To model low-dimensional semantic manifolds commonly observed in real-world datasets, we introduce a rank parameter $\alpha \in (0, 1]$. Given the observed feature dimension F, we define the number of latent semantic factors as

$$\bar{F} = \lceil \alpha F \rceil.$$

We first sample a semantic matrix $\boldsymbol{S} \in \mathbb{R}^{N \times \bar{F}}$, where each latent dimension is independently sampled from a shared distribution across nodes. The semantic matrix is then projected into the observed feature space:

$$\boldsymbol{X} = \boldsymbol{S}\boldsymbol{W}, \qquad \boldsymbol{W} \in \mathbb{R}^{\bar{F} \times F}.$$

The projection matrix $\boldsymbol{W}$ is initialized with orthonormal rows so that the effective rank (Roy & Vetterli, 2007) of $\boldsymbol{X}$ is approximately $\bar{F}$. Overall, this procedure enables explicit control over intrinsic dimensionality while the random projection produces diverse feature patterns.

Graph-aware propagation. Features generated independently of graph structure contain limited graph-aligned variation, while real-world graphs often exhibit correlations between structure and node features. To introduce such dependencies, we apply stochastic graph propagation:

$$\boldsymbol{X} \leftarrow \left((1 + \beta\alpha)\boldsymbol{I} - \alpha\hat{\boldsymbol{A}}\right)^S \boldsymbol{X},$$

where $\hat{\boldsymbol{A}}$ denotes the symmetrically normalized adjacency matrix. The propagation process is controlled by: (i) propagation depth S, (ii) mixing strength α, and (iii) mode selector β. When $\beta = 0$, the operator emphasizes differences between neighboring nodes, while $\beta = 1$ additionally preserves the original node signal.

By varying these latent variables, the generator produces features with diverse levels of local smoothness similar to those observed in real-world graph datasets. We further theoretically analyze the limitations of generating features without this propagation stage in Appendix B.

B. Limitations of Random Node Assignment

Without graph-aware propagation, the generated features remain weakly correlated with the graph structure because they are initially assigned independently of the graph. As a result, they fail to capture edge-aligned variations commonly observed in real-world graphs.

This limitation can be formalized through a spectral analysis of the graph Laplacian. Intuitively, random feature assignment distributes energy uniformly across eigenspaces, making significant concentration in low-dimensional high-frequency subspaces unlikely. These subspaces correspond to localized structural variations between neighboring nodes. The following lemma formalizes this observation.

Lemma B.1. *Let $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ be node features generated independently of the graph structure, and let $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ denote the adjacency matrix. Then, in expectation, at most a fraction k/N of the feature energy lies in any k-dimensional high-frequency eigenspace of the graph Laplacian.*

The proof is provided below. Since $k \ll N$ in practice, this result implies that random node feature assignment

alone is insufficient to generate graph-aligned variations. As a consequence, additional structure-aware processing is required to inject high-frequency components that reflect the underlying graph topology.

Proof. Let $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ and let $\boldsymbol{P}_H$ be an orthogonal projector onto a k-dimensional high-frequency eigenspace of $\boldsymbol{L}$. We generate $\boldsymbol{X} = \boldsymbol{S}\boldsymbol{W}$, where $\boldsymbol{W} \in \mathbb{R}^{\bar{F}\times F}$ has orthonormal rows ($\boldsymbol{W}\boldsymbol{W}^\top = \boldsymbol{I}_{\bar{F}}$), and for each factor $j \in [\bar{F}]$ we sample

$$\mu_j \sim \mathcal{N}(0,1), \qquad z_j \sim \mathcal{N}(0,1), \qquad \sigma_j = |z_j \mu_j|,$$

then draw node-wise entries independently as

$$S_{u,j} \mid (\mu_j, \sigma_j) \sim \mathcal{N}(\mu_j, \sigma_j^2), \qquad u \in [N].$$

Since $\boldsymbol{W}\boldsymbol{W}^\top = \boldsymbol{I}_{\bar{F}}$, we have $\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{S}\boldsymbol{S}^\top$. Moreover, for $u \neq v$,

$$\mathbb{E}[S_{u,j}S_{v,j}] = \mathbb{E}[\mu_j^2], \qquad \mathbb{E}[S_{u,j}^2] = \mathbb{E}[\sigma_j^2] + \mathbb{E}[\mu_j^2],$$

hence

$$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top] = \mathbb{E}[\boldsymbol{S}\boldsymbol{S}^\top] = \bar{F}\,\mathbb{E}[\sigma^2]\,\boldsymbol{I}_N + \bar{F}\,\mathbb{E}[\mu^2]\,\mathbf{1}\mathbf{1}^\top.$$

Therefore,

$$\begin{aligned}\mathbb{E}\big[\|\boldsymbol{P}_H\boldsymbol{X}\|_F^2\big] &= \mathrm{tr}\big(\boldsymbol{P}_H\,\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top]\big)\\ &= \bar{F}\,\mathbb{E}[\sigma^2]\mathrm{tr}(\boldsymbol{P}_H) + \bar{F}\,\mathbb{E}[\mu^2]\mathbf{1}^\top\boldsymbol{P}_H\mathbf{1}.\end{aligned}$$

Using $\mathrm{tr}(\boldsymbol{P}_H) = k$ and $0 \preceq \boldsymbol{P}_H \preceq \boldsymbol{I}$ (so $\mathbf{1}^\top\boldsymbol{P}_H\mathbf{1} \leq \mathbf{1}^\top\boldsymbol{I}\mathbf{1} = N$), we obtain

$$\mathbb{E}\big[\|\boldsymbol{P}_H\boldsymbol{X}\|_F^2\big] \leq \bar{F}\,\mathbb{E}[\sigma^2]\,k + \bar{F}\,\mathbb{E}[\mu^2]\,N.$$

Also,

$$\mathbb{E}\big[\|\boldsymbol{X}\|_F^2\big] = \mathrm{tr}\big(\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top]\big) = \bar{F}\,\mathbb{E}[\sigma^2]\,N + \bar{F}\,\mathbb{E}[\mu^2]\,N^2.$$

Hence for $k \geq 1$,

$$\frac{\mathbb{E}[\|\boldsymbol{P}_H\boldsymbol{X}\|_F^2]}{\mathbb{E}[\|\boldsymbol{X}\|_F^2]} \leq \frac{\bar{F}\,\mathbb{E}[\sigma^2]\,k + \bar{F}\,\mathbb{E}[\mu^2]\,N}{\bar{F}\,\mathbb{E}[\sigma^2]\,N + \bar{F}\,\mathbb{E}[\mu^2]\,N^2} \leq \frac{k}{N},$$

which proves the claim. □

C. Baseline Methods

We compare Node4All against 21 baselines detailed below.

Shallow baselines. We include linear predictors, multi-layer perceptrons (MLPs), as well as node2vec (Grover & Leskovec, 2016). For node2vec representations, we concatenate the learned embeddings with the original node features before passing them to the predictor, ensuring a fair comparison with baselines that utilize node features.

Table 3. Ablation study on the CGT.

Modification	**Total Avg. (25)**
Use GAT instead CGT	$37.80_{\pm 1.29}$
Use GCN instead CGT	$37.61_{\pm 0.92}$
Use GAT instead Multi-hop Tokenization	$68.92_{\pm 1.34}$
Use GCN instead Multi-hop Tokenization	$69.85_{\pm 1.48}$
Use MLP instead Transformer Readout	$68.14_{\pm 2.34}$
Node4All	$\mathbf{71.55_{\pm 1.38}}$

Table 4. Average test accuracy (%) across 25 datasets for CGT trained on real-world datasets with masked autoencoding.

Dataset	**# Nodes**	**Total Avg. (25)**	**Rank @ Baselines**
Wisconsin	251	$60.39_{\pm 3.58}$	20
Cora	2,708	$61.79_{\pm 3.80}$	20
Roman Empire	22,662	$55.32_{\pm 1.78}$	20
Questions	48,921	$69.58_{\pm 1.60}$	9
OGBN-Arxiv	169,343	$69.94_{\pm 1.48}$	9
OGBN-Products	2,449,029	$68.82_{\pm 1.36}$	10
Node4All	-	$\mathbf{71.55_{\pm 1.38}}$	5

Supervised GNN Baselines. We include a diverse set of widely used GNNs covering message passing, attention, propagation, and deep residual architectures: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), APPNP (Gasteiger et al., 2019), JKNet (Xu et al., 2018b), GAT (Veličković et al., 2018), SGC (Wu et al., 2019), GCNII (Chen et al., 2020b), GATv2 (Brody et al., 2021), S^2GC (Zhu & Koniusz, 2021), UniMP (Shi et al., 2021), and GPR-GNN (Chien et al., 2021).

Graph SSL baselines. We include self-supervised graph learning methods spanning both contrastive and generative approaches. The contrastive baselines are BGRL (Thakoor et al., 2021) and CCA-SSG (Zhang et al., 2021), while the generative baselines include GAE (Kipf & Welling, 2016), GraphMAE (Hou et al., 2022), GraphMAE2 (Hou et al., 2023), and MaskGAE (Li et al., 2023).

In the main experiments, we use the same one-hidden-layer MLP predictor for all representation learning methods, including Node4All, to isolate representation quality from the choice of predictor.

D. Ablation Studies

While we provide a summary of the ablation study in Section 8.3, we present detailed results and analyses here.

Channel Graph Transformer (CGT). CGT is designed to process a *channel graph*, where each node is associated with a single scalar feature. As a baseline, we apply standard GNNs architectures (GCN and GAT) by lifting scalar features to a d-dimensional space, performing message passing,

and projecting them back to scalars. As shown in Table 3, this leads to severe performance degradation, with average accuracies of 37.80% and 37.61%, respectively, even below a linear classifier (62.92%). These results suggest that the channel-mixing feature transformations used in standard GNNs are ineffective for channel graphs and highlight the need for a specialized architecture such as CGT.

CGT consists of two components: multi-hop tokenization and Transformer-based token readout (Section 4.2). Since removing either component makes the model ill-defined, we instead evaluate alternative designs. First, we replace dimension-wise tokenization with shared aggregation using GCN or GAT as token generators while preserving the Transformer-based readout. This reduces performance to 69.85% and 68.92%, respectively. Second, replacing the Transformer-based readout with MLP pooling further degrades performance to 68.14%. Overall, these results show that both the tokenization and aggregation designs of CGT are important for achieving strong performance.

Training on Real-world Datasets. While Node4All is pretrained on synthetic graphs, CGT can also be trained on real-world datasets. We evaluate CGT models pretrained on six real-world datasets (Table 4).

All six real-world pretrained variants underperform the synthetic-pretrained Node4All, highlighting the advantage of synthetic graph pretraining for generalization. In detail, CGT fails to learn meaningful representations from small datasets such as Wisconsin, Cora, and Roman Empire, resulting in significantly degraded performance. While it achieves moderate performance when pretrained on larger datasets such as Questions, OGBN-Arxiv, and OGBN-Products, it still remains inferior to the synthetic-pretrained CGT.

Importantly, larger dataset size alone does not guarantee better performance: despite being much larger than OGBN-Arxiv, OGBN-Products does not yield improved results. These observations suggest that effective pretraining depends more on structural and feature diversity than on dataset size alone. Overall, these results show that synthetic graph pretraining provides a controllable and effective way to induce diverse training signals.

E. Additional Visualization

As described in Section 8.2, we visualize how the values of each channel change before and after processing by Node4All. Here, we provide visualizations on datasets with binary (Amazon Photo), categorical (Wiki), and continuous features (OGBN-Arxiv), shown in Figure 8. Across all feature types, Node4All distinguishes nodes with identical or similar feature values while (i) preserving the original feature groups and overall trends, and (ii) separating nodes according to the underlying graph structure.

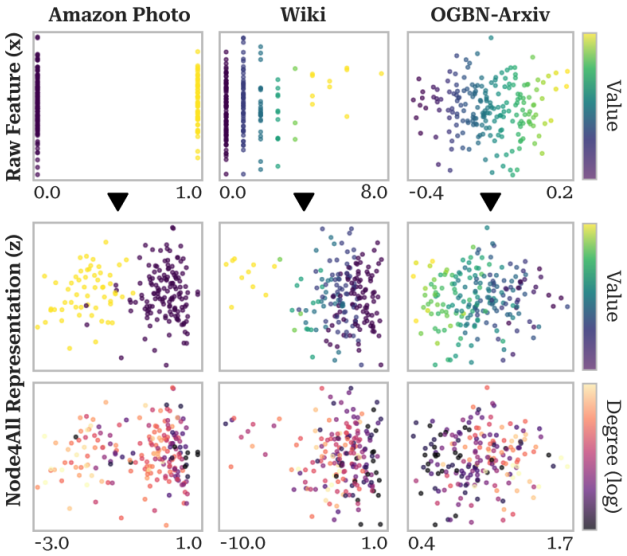

Figure 8. Scatter plots of raw features (top) and Node4All representations (bottom) for the first feature channel on Amazon Photo, Wiki, and OGBN-Arxiv, representing binary, categorical, and continuous features.

F. Full Experiment Results

Full results for the main experiments (Section 7.1) across 21 baselines and 25 benchmarks are reported in Table 5 and Table 6. Table 5 shows results under fixed hyperparameters (**Fixed-HP**), while Table 6 reports results with per-dataset hyperparameter tuning (**Tuned-HP**). Node4All has identical values in both tables since it requires neither per-dataset training nor hyperparameter tuning.

G. Dataset Details

Our dataset selection and splitting strategy largely follows prior fully inductive GFMs, specifically GraphAny (Zhao et al., 2025). Datasets are loaded from PyTorch Geometric (PyG) (Fey & Lenssen, 2019), Deep Graph Library (DGL) (Wang et al., 2019), and the heterophilous graph benchmark released by Yandex Research (Platonov et al., 2023). All graphs are treated as undirected. For the Chameleon and Squirrel datasets, we use the filtered versions provided by the repository[1], since the original datasets were reported to contain issues (Platonov et al., 2023). Detailed dataset statistics are reported in Table 7.

[1] https://github.com/yandex-research/heterophilous-graphs

Table 5. Performance (%) across 25 node classification benchmarks comparing 14 supervised methods and 7 graph representation learning methods. All baselines use *fixed hyperparameter setting* (**Fixed-HP**), where a single hyperparameter configuration is selected based on the best average validation accuracy across all datasets and then applied uniformly to every dataset; models are still trained separately on each dataset. In contrast, Node4All is trained solely on synthetic graphs and applied uniformly across all benchmarks without any per-dataset training or tuning. The **best** and second-best methods in terms of average accuracy are highlighted. Despite never observing target datasets during training, Node4All achieves the best overall performance, demonstrating its effectiveness in generalizing to arbitrary graph datasets within a single model.

Methods	Actor	Amazon Computer	Amazon Photo	Amazon Ratings	Blog Catalog	Chameleon	Citeseer	Coauthor CS	Coauthor Physics	Cora	Cornell	DBLP	Deezer
Fixed Hyperparameter Setting													
Linear	$34.35_{\pm 0.96}$	$69.75_{\pm 1.29}$	$77.77_{\pm 1.17}$	$38.61_{\pm 0.44}$	$68.97_{\pm 1.34}$	$34.10_{\pm 2.53}$	$49.05_{\pm 0.74}$	$84.28_{\pm 0.58}$	$84.24_{\pm 1.78}$	$47.44_{\pm 0.68}$	$69.73_{\pm 7.33}$	$55.28_{\pm 0.76}$	$55.20_{\pm 2.38}$
MLP	$34.79_{\pm 0.98}$	$69.66_{\pm 0.78}$	$78.41_{\pm 1.49}$	$45.77_{\pm 0.38}$	$73.53_{\pm 0.78}$	$39.75_{\pm 4.63}$	$51.60_{\pm 1.39}$	$87.01_{\pm 0.62}$	$88.02_{\pm 1.28}$	$55.79_{\pm 1.14}$	$72.70_{\pm 5.47}$	$55.57_{\pm 0.74}$	$57.16_{\pm 2.10}$
GIN	$24.75_{\pm 4.50}$	$37.05_{\pm 7.61}$	$23.82_{\pm 5.11}$	$44.86_{\pm 3.26}$	$18.21_{\pm 2.49}$	$29.63_{\pm 6.34}$	$53.57_{\pm 4.91}$	$84.50_{\pm 0.58}$	$77.08_{\pm 21.00}$	$70.26_{\pm 1.86}$	$36.49_{\pm 7.67}$	$68.47_{\pm 2.56}$	$55.22_{\pm 2.08}$
GAT	$28.54_{\pm 1.00}$	$83.59_{\pm 1.28}$	$90.44_{\pm 1.38}$	$49.09_{\pm 0.61}$	$48.13_{\pm 9.74}$	$39.97_{\pm 3.25}$	$67.81_{\pm 1.25}$	$89.86_{\pm 0.55}$	$91.81_{\pm 0.90}$	$79.58_{\pm 1.39}$	$41.89_{\pm 5.02}$	$72.84_{\pm 2.40}$	$54.51_{\pm 0.74}$
GAT2	$27.51_{\pm 0.75}$	$84.78_{\pm 0.90}$	$89.86_{\pm 1.21}$	$49.86_{\pm 0.55}$	$49.33_{\pm 12.48}$	$39.63_{\pm 2.82}$	$67.57_{\pm 1.38}$	$89.00_{\pm 0.67}$	$92.03_{\pm 0.82}$	$78.94_{\pm 1.37}$	$40.81_{\pm 4.75}$	$74.76_{\pm 2.65}$	$55.85_{\pm 0.38}$
SGC	$29.27_{\pm 1.15}$	$85.40_{\pm 1.06}$	$91.79_{\pm 0.46}$	$42.19_{\pm 0.50}$	$71.38_{\pm 1.04}$	$41.59_{\pm 4.07}$	$67.18_{\pm 0.40}$	$90.34_{\pm 0.54}$	$92.73_{\pm 0.64}$	$78.22_{\pm 0.47}$	$43.24_{\pm 7.93}$	$67.39_{\pm 3.49}$	$55.05_{\pm 1.51}$
S^2GC	$29.34_{\pm 1.00}$	$85.51_{\pm 1.10}$	$91.95_{\pm 0.41}$	$42.15_{\pm 0.38}$	$78.74_{\pm 1.40}$	$39.46_{\pm 3.42}$	$66.71_{\pm 0.44}$	$90.65_{\pm 0.48}$	$91.74_{\pm 0.90}$	$78.21_{\pm 0.28}$	$40.54_{\pm 5.41}$	$69.98_{\pm 3.33}$	$55.02_{\pm 1.68}$
JKNet	$29.74_{\pm 1.14}$	$85.37_{\pm 0.84}$	$91.86_{\pm 0.55}$	$48.85_{\pm 0.83}$	$71.06_{\pm 1.24}$	$39.38_{\pm 3.36}$	$69.38_{\pm 0.71}$	$90.63_{\pm 0.63}$	$93.12_{\pm 0.43}$	$81.04_{\pm 0.85}$	$43.51_{\pm 5.85}$	$74.11_{\pm 3.57}$	$56.00_{\pm 0.84}$
APPNP	$31.28_{\pm 0.94}$	$85.27_{\pm 1.14}$	$92.03_{\pm 0.42}$	$49.28_{\pm 0.60}$	$86.11_{\pm 2.50}$	$38.88_{\pm 2.92}$	$68.56_{\pm 0.41}$	$90.73_{\pm 0.61}$	$93.64_{\pm 0.43}$	$81.50_{\pm 0.54}$	$38.65_{\pm 4.99}$	$75.32_{\pm 3.14}$	$56.20_{\pm 1.09}$
GCN	$29.41_{\pm 0.75}$	$85.36_{\pm 0.90}$	$91.59_{\pm 0.49}$	$49.53_{\pm 0.37}$	$70.46_{\pm 1.27}$	$39.29_{\pm 2.68}$	$68.67_{\pm 0.69}$	$90.34_{\pm 0.69}$	$93.43_{\pm 0.49}$	$81.07_{\pm 0.36}$	$40.81_{\pm 5.98}$	$70.88_{\pm 3.20}$	$55.50_{\pm 1.76}$
GCN2	$33.76_{\pm 1.06}$	$84.65_{\pm 0.78}$	$91.37_{\pm 0.52}$	$49.39_{\pm 0.78}$	$84.39_{\pm 2.40}$	$40.46_{\pm 3.67}$	$69.00_{\pm 1.35}$	$89.00_{\pm 0.92}$	$92.90_{\pm 0.52}$	$80.55_{\pm 1.33}$	$41.35_{\pm 3.64}$	$73.95_{\pm 3.31}$	$57.02_{\pm 0.98}$
SAGE	$34.31_{\pm 0.72}$	$83.83_{\pm 0.66}$	$91.33_{\pm 0.69}$	$50.48_{\pm 0.57}$	$82.30_{\pm 1.82}$	$41.04_{\pm 3.91}$	$68.42_{\pm 0.65}$	$90.34_{\pm 0.57}$	$93.04_{\pm 0.55}$	$80.14_{\pm 0.76}$	$53.51_{\pm 5.10}$	$71.54_{\pm 2.46}$	$56.35_{\pm 0.94}$
GPR-GNN	$34.06_{\pm 0.69}$	$86.61_{\pm 0.80}$	$92.98_{\pm 0.55}$	$48.75_{\pm 0.65}$	$88.53_{\pm 1.59}$	$38.81_{\pm 2.23}$	$69.83_{\pm 0.91}$	$91.45_{\pm 0.80}$	$93.68_{\pm 0.58}$	$81.25_{\pm 0.78}$	$56.49_{\pm 6.22}$	$72.05_{\pm 2.71}$	$56.40_{\pm 1.10}$
UniMP	$34.86_{\pm 0.83}$	$83.80_{\pm 0.83}$	$91.55_{\pm 0.85}$	$51.16_{\pm 0.80}$	$73.36_{\pm 1.68}$	$42.02_{\pm 3.59}$	$69.09_{\pm 0.54}$	$90.49_{\pm 0.64}$	$92.38_{\pm 0.63}$	$80.00_{\pm 0.73}$	$54.59_{\pm 5.10}$	$72.25_{\pm 2.35}$	$56.29_{\pm 1.05}$
node2vec	$34.14_{\pm 1.52}$	$78.76_{\pm 0.86}$	$87.29_{\pm 1.10}$	$50.98_{\pm 0.31}$	$78.81_{\pm 1.30}$	$40.45_{\pm 2.49}$	$62.06_{\pm 0.56}$	$88.95_{\pm 0.88}$	$88.71_{\pm 1.32}$	$73.49_{\pm 0.70}$	$69.73_{\pm 4.95}$	$68.55_{\pm 1.28}$	$55.32_{\pm 1.85}$
GAE	$29.66_{\pm 1.28}$	$83.85_{\pm 0.75}$	$91.08_{\pm 0.61}$	$49.59_{\pm 0.83}$	$72.76_{\pm 1.14}$	$43.87_{\pm 2.86}$	$59.36_{\pm 0.71}$	$91.25_{\pm 0.40}$	$92.92_{\pm 0.56}$	$79.59_{\pm 1.00}$	$41.62_{\pm 5.69}$	$80.70_{\pm 1.11}$	$54.55_{\pm 1.14}$
BGRL	$29.81_{\pm 1.11}$	$83.28_{\pm 0.80}$	$90.23_{\pm 0.84}$	$50.99_{\pm 0.41}$	$74.74_{\pm 0.82}$	$43.17_{\pm 3.63}$	$66.09_{\pm 1.12}$	$91.21_{\pm 0.47}$	$93.76_{\pm 0.50}$	$82.25_{\pm 0.45}$	$39.19_{\pm 5.30}$	$80.22_{\pm 1.21}$	$54.53_{\pm 1.60}$
CCA-SSG	$29.88_{\pm 0.79}$	$85.12_{\pm 0.50}$	$90.84_{\pm 0.90}$	$49.78_{\pm 0.62}$	$73.92_{\pm 1.18}$	$44.89_{\pm 3.14}$	$68.76_{\pm 1.25}$	$90.67_{\pm 0.50}$	$93.83_{\pm 0.37}$	$83.22_{\pm 0.30}$	$45.14_{\pm 8.55}$	$81.75_{\pm 0.81}$	$56.23_{\pm 1.70}$
GraphMAE	$29.78_{\pm 0.72}$	$83.48_{\pm 0.92}$	$90.74_{\pm 0.63}$	$49.92_{\pm 0.82}$	$62.77_{\pm 2.16}$	$40.17_{\pm 3.39}$	$70.05_{\pm 0.48}$	$91.81_{\pm 0.35}$	$93.75_{\pm 0.45}$	$82.02_{\pm 0.49}$	$41.89_{\pm 5.83}$	$80.36_{\pm 1.06}$	$54.04_{\pm 1.34}$
GraphMAE2	$29.24_{\pm 1.16}$	$84.97_{\pm 0.74}$	$90.59_{\pm 0.79}$	$51.48_{\pm 0.59}$	$68.96_{\pm 2.44}$	$40.81_{\pm 3.37}$	$71.57_{\pm 1.15}$	$91.53_{\pm 0.32}$	$93.62_{\pm 0.54}$	$82.22_{\pm 0.58}$	$47.57_{\pm 6.19}$	$81.45_{\pm 0.76}$	$55.05_{\pm 1.19}$
MaskGAE	$29.94_{\pm 0.61}$	$84.44_{\pm 1.09}$	$90.38_{\pm 0.67}$	$50.13_{\pm 0.71}$	$70.03_{\pm 0.82}$	$43.84_{\pm 3.90}$	$68.66_{\pm 1.40}$	$90.11_{\pm 0.50}$	$92.31_{\pm 0.49}$	$80.67_{\pm 0.78}$	$45.95_{\pm 4.52}$	$81.95_{\pm 0.92}$	$55.23_{\pm 0.99}$
Node4All	$31.93_{\pm 1.79}$	$81.88_{\pm 0.67}$	$91.04_{\pm 0.59}$	$42.09_{\pm 0.79}$	$77.68_{\pm 1.02}$	$42.99_{\pm 4.82}$	$71.45_{\pm 0.58}$	$91.72_{\pm 0.47}$	$93.90_{\pm 0.45}$	$80.57_{\pm 0.59}$	$62.97_{\pm 5.80}$	$76.15_{\pm 0.86}$	$56.42_{\pm 1.87}$

Methods	Mine-sweeper	PubMed	Questions	Roman Empire	Squirrel	Texas	Tolokers	Wiki	Wisconsin	WikiCS	Full-Cora	OGBN Arxiv	**Total Avg.** (25)
Fixed Hyperparameter Setting													
Linear	$80.00_{\pm 0.00}$	$68.57_{\pm 0.31}$	$97.13_{\pm 0.03}$	$63.69_{\pm 0.61}$	$29.72_{\pm 1.79}$	$73.63_{\pm 5.47}$	$78.16_{\pm 0.00}$	$68.23_{\pm 1.97}$	$82.55_{\pm 2.83}$	$72.75_{\pm 0.53}$	$34.15_{\pm 0.28}$	$52.44_{\pm 0.03}$	$62.79_{\pm 1.43}$
MLP	$80.00_{\pm 0.00}$	$70.57_{\pm 0.73}$	$97.14_{\pm 0.13}$	$65.12_{\pm 0.65}$	$37.02_{\pm 2.58}$	$75.52_{\pm 6.52}$	$78.21_{\pm 0.20}$	$75.57_{\pm 0.98}$	$82.94_{\pm 6.01}$	$71.32_{\pm 0.63}$	$43.00_{\pm 0.56}$	$54.60_{\pm 0.19}$	$65.63_{\pm 1.64}$
GIN	$74.10_{\pm 18.04}$	$69.26_{\pm 8.89}$	$87.44_{\pm 28.15}$	$51.24_{\pm 1.29}$	$32.64_{\pm 5.22}$	$60.56_{\pm 5.80}$	$55.63_{\pm 27.59}$	$17.28_{\pm 1.75}$	$49.41_{\pm 11.73}$	$29.35_{\pm 5.55}$	$57.86_{\pm 0.57}$	$60.12_{\pm 4.91}$	$50.75_{\pm 7.58}$
GAT	$81.27_{\pm 0.76}$	$75.73_{\pm 0.92}$	$97.07_{\pm 0.03}$	$43.16_{\pm 0.79}$	$33.74_{\pm 1.29}$	$58.42_{\pm 6.01}$	$80.03_{\pm 0.46}$	$43.67_{\pm 5.12}$	$53.33_{\pm 5.93}$	$78.78_{\pm 0.65}$	$59.80_{\pm 0.61}$	$71.51_{\pm 0.21}$	$64.58_{\pm 2.09}$
GAT2	$81.61_{\pm 0.75}$	$75.12_{\pm 0.99}$	$97.07_{\pm 0.03}$	$45.50_{\pm 1.14}$	$34.02_{\pm 1.86}$	$55.16_{\pm 6.41}$	$81.13_{\pm 0.36}$	$30.97_{\pm 6.23}$	$50.20_{\pm 4.22}$	$79.03_{\pm 0.65}$	$59.03_{\pm 0.80}$	$72.12_{\pm 0.19}$	$64.04_{\pm 2.17}$
SGC	$80.14_{\pm 0.11}$	$76.65_{\pm 0.13}$	$97.11_{\pm 0.02}$	$44.26_{\pm 0.57}$	$38.18_{\pm 1.57}$	$57.60_{\pm 4.18}$	$78.68_{\pm 0.38}$	$71.18_{\pm 1.03}$	$52.16_{\pm 4.31}$	$78.71_{\pm 0.55}$	$59.49_{\pm 0.56}$	$68.97_{\pm 0.05}$	$66.36_{\pm 1.47}$
S^2GC	$80.18_{\pm 0.17}$	$76.54_{\pm 0.11}$	$97.09_{\pm 0.01}$	$46.96_{\pm 0.54}$	$36.58_{\pm 1.11}$	$57.88_{\pm 4.14}$	$78.68_{\pm 0.33}$	$71.97_{\pm 1.27}$	$47.65_{\pm 5.34}$	$79.06_{\pm 0.73}$	$60.17_{\pm 0.53}$	$69.16_{\pm 0.01}$	$66.48_{\pm 1.38}$
JKNet	$80.39_{\pm 0.32}$	$76.56_{\pm 0.42}$	$97.07_{\pm 0.03}$	$53.87_{\pm 0.86}$	$34.23_{\pm 1.15}$	$58.42_{\pm 4.91}$	$80.66_{\pm 0.44}$	$72.35_{\pm 1.02}$	$53.53_{\pm 3.40}$	$78.45_{\pm 0.36}$	$63.05_{\pm 0.50}$	$71.81_{\pm 0.33}$	$67.78_{\pm 1.38}$
APPNP	$80.04_{\pm 0.10}$	$76.90_{\pm 0.16}$	$97.13_{\pm 0.05}$	$56.14_{\pm 0.54}$	$33.67_{\pm 1.30}$	$55.95_{\pm 4.29}$	$78.10_{\pm 0.19}$	$71.19_{\pm 1.40}$	$49.80_{\pm 4.57}$	$78.66_{\pm 0.67}$	$63.56_{\pm 0.38}$	$70.76_{\pm 0.16}$	$67.97_{\pm 1.34}$
GCN	$80.13_{\pm 0.17}$	$76.18_{\pm 0.27}$	$97.14_{\pm 0.03}$	$45.35_{\pm 0.40}$	$33.64_{\pm 0.86}$	$58.96_{\pm 5.23}$	$79.45_{\pm 0.58}$	$70.25_{\pm 0.71}$	$52.35_{\pm 4.30}$	$78.55_{\pm 0.74}$	$62.82_{\pm 0.43}$	$70.88_{\pm 0.28}$	$66.88_{\pm 1.34}$
GCN2	$86.10_{\pm 0.56}$	$75.75_{\pm 1.22}$	$97.20_{\pm 0.03}$	$75.81_{\pm 0.76}$	$34.96_{\pm 1.84}$	$56.79_{\pm 5.79}$	$82.11_{\pm 0.83}$	$75.33_{\pm 1.55}$	$61.37_{\pm 5.26}$	$78.78_{\pm 0.81}$	$60.04_{\pm 0.44}$	$71.89_{\pm 0.31}$	$69.76_{\pm 1.63}$
SAGE	$86.30_{\pm 0.63}$	$75.91_{\pm 0.24}$	$97.23_{\pm 0.06}$	$77.18_{\pm 0.65}$	$37.46_{\pm 1.26}$	$66.82_{\pm 8.62}$	$80.28_{\pm 0.89}$	$73.94_{\pm 1.37}$	$67.45_{\pm 4.57}$	$78.67_{\pm 0.52}$	$61.49_{\pm 0.42}$	$70.88_{\pm 0.12}$	$70.81_{\pm 1.55}$
GPR-GNN	$84.02_{\pm 0.47}$	$76.64_{\pm 0.34}$	$97.27_{\pm 0.04}$	$73.33_{\pm 0.63}$	$34.66_{\pm 1.42}$	$68.75_{\pm 5.95}$	$78.28_{\pm 0.13}$	$72.30_{\pm 1.09}$	$71.18_{\pm 4.73}$	$79.14_{\pm 0.72}$	$63.37_{\pm 0.49}$	$68.39_{\pm 0.20}$	$71.13_{\pm 1.43}$
UniMP	$86.16_{\pm 0.52}$	$76.31_{\pm 0.29}$	$97.23_{\pm 0.04}$	$79.26_{\pm 0.62}$	$37.55_{\pm 1.35}$	$67.93_{\pm 6.97}$	$81.19_{\pm 0.65}$	$74.70_{\pm 1.50}$	$71.57_{\pm 4.04}$	$79.26_{\pm 0.76}$	$60.38_{\pm 0.54}$	$70.83_{\pm 0.23}$	$70.97_{\pm 1.48}$
node2vec	$79.98_{\pm 0.04}$	$65.27_{\pm 0.77}$	$97.15_{\pm 0.05}$	$62.91_{\pm 0.38}$	$38.57_{\pm 2.25}$	$74.73_{\pm 5.29}$	$81.00_{\pm 0.54}$	$79.77_{\pm 0.73}$	$80.98_{\pm 5.48}$	$78.29_{\pm 0.46}$	$56.34_{\pm 0.36}$	$70.10_{\pm 0.07}$	$69.69_{\pm 1.42}$
GAE	$80.11_{\pm 0.28}$	$77.67_{\pm 0.63}$	$97.12_{\pm 0.02}$	$45.86_{\pm 0.41}$	$41.33_{\pm 1.50}$	$53.54_{\pm 6.56}$	$81.06_{\pm 0.55}$	$68.78_{\pm 1.53}$	$50.20_{\pm 2.51}$	$78.79_{\pm 0.88}$	$63.31_{\pm 0.56}$	$72.02_{\pm 0.08}$	$67.22_{\pm 1.34}$
BGRL	$80.24_{\pm 0.30}$	$78.37_{\pm 1.12}$	$97.15_{\pm 0.04}$	$51.45_{\pm 0.46}$	$41.84_{\pm 1.11}$	$53.51_{\pm 3.97}$	$80.95_{\pm 0.80}$	$72.06_{\pm 1.52}$	$52.94_{\pm 4.64}$	$78.24_{\pm 0.62}$	$62.39_{\pm 0.58}$	$71.45_{\pm 0.18}$	$68.00_{\pm 1.34}$
CCA-SSG	$80.13_{\pm 0.17}$	$79.37_{\pm 0.27}$	$97.13_{\pm 0.03}$	$54.80_{\pm 0.50}$	$42.82_{\pm 2.38}$	$56.50_{\pm 6.12}$	$80.81_{\pm 0.50}$	$71.71_{\pm 0.79}$	$50.98_{\pm 3.82}$	$78.45_{\pm 0.76}$	$62.31_{\pm 0.40}$	$71.66_{\pm 0.22}$	$68.83_{\pm 1.46}$
GraphMAE	$80.10_{\pm 0.14}$	$78.63_{\pm 1.19}$	$97.08_{\pm 0.04}$	$46.68_{\pm 0.57}$	$34.80_{\pm 1.72}$	$55.69_{\pm 7.72}$	$78.29_{\pm 0.16}$	$49.41_{\pm 1.96}$	$52.35_{\pm 7.44}$	$78.60_{\pm 0.41}$	$63.76_{\pm 0.47}$	$71.21_{\pm 0.26}$	$66.30_{\pm 1.63}$
GraphMAE2	$80.04_{\pm 0.17}$	$76.18_{\pm 0.82}$	$97.10_{\pm 0.02}$	$48.73_{\pm 0.48}$	$36.61_{\pm 1.52}$	$61.94_{\pm 5.57}$	$78.46_{\pm 0.47}$	$51.47_{\pm 1.75}$	$54.31_{\pm 5.75}$	$79.17_{\pm 0.45}$	$57.96_{\pm 0.50}$	$71.59_{\pm 0.14}$	$67.31_{\pm 1.50}$
MaskGAE	$84.54_{\pm 0.47}$	$80.70_{\pm 0.77}$	$97.15_{\pm 0.06}$	$70.33_{\pm 0.51}$	$43.61_{\pm 2.41}$	$57.01_{\pm 7.70}$	$81.34_{\pm 0.59}$	$69.47_{\pm 0.90}$	$51.18_{\pm 5.00}$	$78.92_{\pm 0.72}$	$62.43_{\pm 0.48}$	$71.78_{\pm 0.28}$	$69.28_{\pm 1.49}$
Node4All	$79.94_{\pm 0.17}$	$78.43_{\pm 0.58}$	$97.06_{\pm 0.07}$	$71.84_{\pm 0.41}$	$42.50_{\pm 1.88}$	$75.52_{\pm 3.81}$	$79.28_{\pm 0.60}$	$78.44_{\pm 0.83}$	$76.27_{\pm 4.51}$	$78.02_{\pm 0.65}$	$62.62_{\pm 0.46}$	$68.14_{\pm 0.20}$	$\mathbf{71.55}_{\pm \mathbf{1.38}}$

Table 6. Performance (%) across 25 node classification benchmarks comparing 14 supervised methods and 7 graph representation learning methods. All baselines use *per-dataset hyperparameter tuning* (**Tuned-HP**), with hyperparameters selected based on validation accuracy for each dataset. The **best** and second-best methods in terms of average accuracy are highlighted. Despite not using any dataset-specific training or hyperparameter optimization, Node4All ranks 5th out of 21 methods, demonstrating competitive performance against extensively tuned, dataset-specific baselines.

Methods	Actor	Amazon Computer	Amazon Photo	Amazon Ratings	Blog Catalog	Chameleon	Citeseer	Coauthor CS	Coauthor Physics	Cora	Cornell	DBLP	Deezer
Per-dataset Hyperparameter Tuning													
Linear	$34.35_{\pm 0.96}$	$69.75_{\pm 1.29}$	$77.77_{\pm 1.17}$	$38.61_{\pm 0.44}$	$68.97_{\pm 1.34}$	$33.52_{\pm 2.73}$	$49.05_{\pm 0.74}$	$84.35_{\pm 0.50}$	$84.24_{\pm 1.78}$	$47.53_{\pm 0.51}$	$69.73_{\pm 7.33}$	$55.28_{\pm 0.76}$	$55.04_{\pm 2.39}$
MLP	$34.82_{\pm 0.95}$	$71.00_{\pm 1.07}$	$78.55_{\pm 1.42}$	$46.14_{\pm 0.47}$	$73.81_{\pm 1.14}$	$39.95_{\pm 3.21}$	$51.40_{\pm 1.52}$	$87.01_{\pm 0.62}$	$88.65_{\pm 0.82}$	$55.73_{\pm 0.80}$	$71.62_{\pm 6.65}$	$55.72_{\pm 0.66}$	$57.23_{\pm 1.42}$
GIN	$26.53_{\pm 1.29}$	$35.26_{\pm 6.71}$	$25.79_{\pm 3.29}$	$44.93_{\pm 3.34}$	$18.21_{\pm 2.49}$	$33.28_{\pm 2.54}$	$53.56_{\pm 4.91}$	$84.97_{\pm 1.02}$	$79.19_{\pm 18.57}$	$54.41_{\pm 20.55}$	$42.43_{\pm 8.38}$	$65.07_{\pm 9.35}$	$55.40_{\pm 0.39}$
GAT	$28.55_{\pm 0.92}$	$85.73_{\pm 0.62}$	$90.99_{\pm 0.62}$	$49.08_{\pm 0.94}$	$48.47_{\pm 9.79}$	$37.32_{\pm 2.39}$	$68.10_{\pm 0.62}$	$89.85_{\pm 0.62}$	$91.89_{\pm 0.82}$	$80.79_{\pm 0.92}$	$41.62_{\pm 5.82}$	$76.74_{\pm 2.47}$	$55.40_{\pm 0.58}$
GAT2	$28.57_{\pm 1.28}$	$85.30_{\pm 0.79}$	$90.34_{\pm 0.66}$	$49.76_{\pm 0.41}$	$57.70_{\pm 3.96}$	$37.55_{\pm 4.65}$	$67.65_{\pm 1.12}$	$89.38_{\pm 1.03}$	$91.59_{\pm 1.12}$	$79.55_{\pm 1.64}$	$41.62_{\pm 7.07}$	$75.38_{\pm 3.28}$	$55.80_{\pm 0.29}$
SGC	$29.06_{\pm 1.09}$	$85.40_{\pm 1.06}$	$91.79_{\pm 0.46}$	$42.41_{\pm 0.64}$	$71.38_{\pm 1.04}$	$41.70_{\pm 2.22}$	$67.29_{\pm 0.46}$	$90.17_{\pm 0.55}$	$92.83_{\pm 0.48}$	$78.46_{\pm 0.25}$	$39.73_{\pm 7.36}$	$73.33_{\pm 3.09}$	$54.90_{\pm 1.61}$
S^2GC	$29.63_{\pm 1.06}$	$85.77_{\pm 0.95}$	$91.86_{\pm 0.65}$	$42.15_{\pm 0.38}$	$78.74_{\pm 1.40}$	$38.79_{\pm 3.01}$	$66.71_{\pm 0.32}$	$90.81_{\pm 0.55}$	$91.74_{\pm 0.90}$	$78.21_{\pm 0.28}$	$38.11_{\pm 8.59}$	$69.98_{\pm 3.33}$	$54.48_{\pm 1.49}$
JKNet	$29.95_{\pm 1.03}$	$85.85_{\pm 1.02}$	$91.66_{\pm 0.56}$	$48.76_{\pm 0.79}$	$71.48_{\pm 0.98}$	$40.88_{\pm 3.81}$	$69.38_{\pm 0.71}$	$90.55_{\pm 0.56}$	$93.33_{\pm 0.49}$	$81.04_{\pm 0.85}$	$45.14_{\pm 5.14}$	$73.89_{\pm 4.01}$	$55.97_{\pm 0.66}$
APPNP	$31.63_{\pm 0.80}$	$85.69_{\pm 1.12}$	$92.35_{\pm 0.38}$	$50.33_{\pm 0.54}$	$85.43_{\pm 1.60}$	$38.35_{\pm 2.38}$	$68.79_{\pm 0.57}$	$91.48_{\pm 0.60}$	$93.30_{\pm 0.53}$	$81.50_{\pm 0.54}$	$49.19_{\pm 5.90}$	$75.32_{\pm 3.14}$	$56.46_{\pm 1.52}$
GCN	$29.76_{\pm 0.73}$	$85.63_{\pm 0.94}$	$91.73_{\pm 0.49}$	$49.49_{\pm 0.46}$	$70.82_{\pm 1.01}$	$39.89_{\pm 2.92}$	$68.67_{\pm 0.69}$	$90.83_{\pm 0.56}$	$93.02_{\pm 0.46}$	$81.58_{\pm 0.67}$	$42.16_{\pm 5.43}$	$75.07_{\pm 3.48}$	$55.46_{\pm 1.60}$
GCN2	$33.83_{\pm 0.76}$	$85.26_{\pm 1.10}$	$91.69_{\pm 0.73}$	$49.95_{\pm 0.54}$	$86.39_{\pm 2.30}$	$39.35_{\pm 3.06}$	$67.95_{\pm 0.77}$	$91.31_{\pm 0.51}$	$92.96_{\pm 0.59}$	$80.50_{\pm 0.53}$	$41.89_{\pm 6.42}$	$75.16_{\pm 2.59}$	$56.57_{\pm 1.22}$
SAGE	$35.66_{\pm 0.91}$	$83.81_{\pm 0.69}$	$91.41_{\pm 0.57}$	$50.54_{\pm 0.55}$	$80.17_{\pm 2.77}$	$40.40_{\pm 3.47}$	$67.40_{\pm 1.36}$	$90.54_{\pm 0.53}$	$92.90_{\pm 0.46}$	$80.14_{\pm 0.76}$	$59.46_{\pm 15.81}$	$72.05_{\pm 2.38}$	$56.49_{\pm 1.25}$
GPR-GNN	$34.91_{\pm 0.54}$	$86.47_{\pm 0.99}$	$93.19_{\pm 0.47}$	$48.81_{\pm 0.64}$	$88.15_{\pm 1.84}$	$38.65_{\pm 3.21}$	$69.96_{\pm 0.97}$	$92.01_{\pm 0.67}$	$93.33_{\pm 0.65}$	$82.22_{\pm 1.01}$	$59.46_{\pm 7.35}$	$76.03_{\pm 2.93}$	$56.40_{\pm 1.10}$
UniMP	$35.52_{\pm 0.93}$	$84.91_{\pm 0.79}$	$92.02_{\pm 0.72}$	$50.37_{\pm 0.48}$	$76.58_{\pm 3.25}$	$39.26_{\pm 4.37}$	$67.00_{\pm 0.42}$	$90.39_{\pm 0.72}$	$92.38_{\pm 0.63}$	$80.11_{\pm 1.15}$	$62.43_{\pm 4.75}$	$71.85_{\pm 2.69}$	$56.75_{\pm 1.28}$
node2vec	$34.31_{\pm 0.87}$	$79.47_{\pm 0.75}$	$87.64_{\pm 1.27}$	$50.98_{\pm 0.31}$	$79.81_{\pm 1.35}$	$40.55_{\pm 2.54}$	$61.76_{\pm 0.55}$	$88.96_{\pm 0.87}$	$89.96_{\pm 1.26}$	$73.73_{\pm 0.86}$	$70.27_{\pm 4.68}$	$70.70_{\pm 0.84}$	$55.60_{\pm 1.97}$
GAE	$30.09_{\pm 0.56}$	$85.63_{\pm 0.83}$	$91.52_{\pm 0.48}$	$50.80_{\pm 0.46}$	$73.63_{\pm 1.28}$	$43.48_{\pm 5.24}$	$65.27_{\pm 1.50}$	$91.42_{\pm 0.46}$	$93.70_{\pm 0.23}$	$81.75_{\pm 0.80}$	$24.05_{\pm 6.22}$	$80.70_{\pm 1.11}$	$55.45_{\pm 1.63}$
BGRL	$30.22_{\pm 1.06}$	$84.90_{\pm 0.58}$	$91.47_{\pm 0.58}$	$51.01_{\pm 0.61}$	$74.51_{\pm 1.16}$	$44.69_{\pm 2.21}$	$64.35_{\pm 0.82}$	$91.65_{\pm 0.39}$	$93.79_{\pm 0.51}$	$81.81_{\pm 0.42}$	$45.95_{\pm 6.04}$	$80.45_{\pm 1.19}$	$55.50_{\pm 1.40}$
CCA-SSG	$36.48_{\pm 0.69}$	$85.99_{\pm 0.85}$	$91.31_{\pm 0.67}$	$51.39_{\pm 0.34}$	$85.05_{\pm 0.67}$	$45.87_{\pm 2.55}$	$68.89_{\pm 0.51}$	$91.65_{\pm 0.23}$	$94.03_{\pm 0.32}$	$82.37_{\pm 0.33}$	$74.59_{\pm 5.16}$	$82.12_{\pm 0.68}$	$59.65_{\pm 1.95}$
GraphMAE	$31.35_{\pm 0.92}$	$86.68_{\pm 0.96}$	$91.73_{\pm 0.46}$	$51.94_{\pm 0.60}$	$69.89_{\pm 1.82}$	$41.87_{\pm 2.86}$	$70.48_{\pm 1.09}$	$92.23_{\pm 0.42}$	$93.75_{\pm 0.43}$	$81.69_{\pm 0.28}$	$45.95_{\pm 6.04}$	$81.65_{\pm 0.39}$	$55.68_{\pm 0.79}$
GraphMAE2	$29.03_{\pm 0.87}$	$86.58_{\pm 0.93}$	$92.03_{\pm 0.57}$	$51.85_{\pm 0.31}$	$69.08_{\pm 1.98}$	$34.16_{\pm 3.76}$	$71.59_{\pm 0.30}$	$92.11_{\pm 0.28}$	$93.90_{\pm 0.48}$	$82.76_{\pm 0.40}$	$47.30_{\pm 5.95}$	$81.96_{\pm 0.57}$	$55.60_{\pm 0.82}$
MaskGAE	$30.10_{\pm 0.81}$	$86.04_{\pm 0.98}$	$91.13_{\pm 0.83}$	$50.65_{\pm 0.57}$	$72.49_{\pm 0.75}$	$43.39_{\pm 2.99}$	$70.76_{\pm 0.72}$	$91.15_{\pm 0.45}$	$93.51_{\pm 0.33}$	$81.29_{\pm 0.57}$	$53.24_{\pm 7.36}$	$82.61_{\pm 0.91}$	$55.54_{\pm 0.78}$
Node4All	$31.93_{\pm 1.79}$	$81.88_{\pm 0.67}$	$91.04_{\pm 0.59}$	$42.09_{\pm 0.79}$	$77.68_{\pm 1.02}$	$42.99_{\pm 4.82}$	$71.45_{\pm 0.58}$	$91.72_{\pm 0.47}$	$93.90_{\pm 0.45}$	$80.57_{\pm 0.59}$	$62.97_{\pm 5.80}$	$76.15_{\pm 0.86}$	$56.42_{\pm 1.87}$

Methods	Mine-sweeper	PubMed	Questions	Roman Empire	Squirrel	Texas	Tolokers	Wiki	Wisconsin	WikiCS	Full-Cora	OGBN Arxiv	**Total Avg.** (25)
Per-dataset Hyperparameter Tuning													
Linear	$80.00_{\pm 0.00}$	$68.57_{\pm 0.31}$	$97.13_{\pm 0.03}$	$63.69_{\pm 0.61}$	$29.72_{\pm 1.79}$	$73.63_{\pm 5.47}$	$78.15_{\pm 0.02}$	$72.06_{\pm 0.73}$	$82.55_{\pm 2.83}$	$72.75_{\pm 0.53}$	$34.15_{\pm 0.28}$	$52.44_{\pm 0.03}$	$62.92_{\pm 1.38}$
MLP	$80.00_{\pm 0.00}$	$70.57_{\pm 0.73}$	$97.27_{\pm 0.05}$	$65.42_{\pm 0.58}$	$38.06_{\pm 1.46}$	$75.52_{\pm 6.52}$	$78.01_{\pm 0.42}$	$75.45_{\pm 0.83}$	$83.14_{\pm 5.96}$	$72.57_{\pm 0.65}$	$46.47_{\pm 0.24}$	$53.80_{\pm 0.14}$	$65.92_{\pm 1.53}$
GIN	$80.15_{\pm 0.22}$	$69.07_{\pm 9.64}$	$97.03_{\pm 0.04}$	$52.43_{\pm 1.19}$	$35.05_{\pm 1.22}$	$60.35_{\pm 5.62}$	$78.16_{\pm 0.00}$	$33.27_{\pm 18.37}$	$49.61_{\pm 11.60}$	$59.28_{\pm 12.64}$	$58.11_{\pm 0.49}$	$65.01_{\pm 1.95}$	$54.26_{\pm 5.83}$
GAT	$81.52_{\pm 1.06}$	$75.75_{\pm 1.28}$	$97.09_{\pm 0.03}$	$43.37_{\pm 0.39}$	$34.90_{\pm 1.68}$	$58.69_{\pm 4.36}$	$81.22_{\pm 0.47}$	$62.90_{\pm 1.87}$	$54.12_{\pm 5.70}$	$78.80_{\pm 0.53}$	$60.50_{\pm 0.83}$	$71.81_{\pm 0.20}$	$65.81_{\pm 1.82}$
GAT2	$82.87_{\pm 0.88}$	$76.83_{\pm 0.36}$	$97.06_{\pm 0.04}$	$60.82_{\pm 0.79}$	$34.83_{\pm 1.94}$	$57.03_{\pm 8.00}$	$81.19_{\pm 0.73}$	$56.36_{\pm 1.94}$	$55.49_{\pm 7.89}$	$79.01_{\pm 0.77}$	$60.84_{\pm 0.47}$	$71.98_{\pm 0.14}$	$66.58_{\pm 2.05}$
SGC	$81.40_{\pm 0.49}$	$77.47_{\pm 0.09}$	$97.11_{\pm 0.02}$	$44.26_{\pm 0.57}$	$38.47_{\pm 2.20}$	$57.60_{\pm 4.18}$	$78.67_{\pm 0.25}$	$71.66_{\pm 1.09}$	$48.43_{\pm 6.74}$	$78.71_{\pm 0.55}$	$60.06_{\pm 0.64}$	$68.69_{\pm 0.02}$	$66.44_{\pm 1.49}$
S^2GC	$81.04_{\pm 0.46}$	$76.54_{\pm 0.11}$	$97.11_{\pm 0.02}$	$48.53_{\pm 0.38}$	$36.54_{\pm 1.24}$	$59.21_{\pm 5.08}$	$78.73_{\pm 0.27}$	$72.61_{\pm 1.14}$	$48.63_{\pm 5.10}$	$78.97_{\pm 0.58}$	$59.86_{\pm 0.45}$	$69.16_{\pm 0.01}$	$66.56_{\pm 1.51}$
JKNet	$81.36_{\pm 0.46}$	$76.25_{\pm 1.29}$	$97.11_{\pm 0.05}$	$54.58_{\pm 3.72}$	$34.66_{\pm 1.33}$	$57.61_{\pm 6.84}$	$80.46_{\pm 0.75}$	$72.26_{\pm 1.08}$	$52.75_{\pm 4.76}$	$79.03_{\pm 0.71}$	$62.91_{\pm 0.39}$	$71.84_{\pm 0.24}$	$67.95_{\pm 1.69}$
APPNP	$80.94_{\pm 0.33}$	$76.79_{\pm 0.16}$	$97.12_{\pm 0.05}$	$60.87_{\pm 0.36}$	$33.92_{\pm 1.37}$	$57.06_{\pm 5.18}$	$78.15_{\pm 0.22}$	$74.72_{\pm 0.56}$	$53.53_{\pm 6.45}$	$78.90_{\pm 0.52}$	$63.80_{\pm 0.45}$	$70.73_{\pm 0.15}$	$69.05_{\pm 1.42}$
GCN	$80.61_{\pm 0.49}$	$77.66_{\pm 0.75}$	$97.11_{\pm 0.06}$	$45.44_{\pm 0.45}$	$36.24_{\pm 1.75}$	$60.04_{\pm 6.31}$	$80.33_{\pm 0.58}$	$70.99_{\pm 1.18}$	$54.12_{\pm 4.31}$	$78.76_{\pm 0.66}$	$63.00_{\pm 0.57}$	$71.22_{\pm 0.22}$	$67.59_{\pm 1.47}$
GCN2	$86.48_{\pm 0.60}$	$76.24_{\pm 0.54}$	$97.20_{\pm 0.06}$	$76.09_{\pm 0.61}$	$34.12_{\pm 0.99}$	$59.23_{\pm 4.89}$	$81.86_{\pm 0.91}$	$76.45_{\pm 1.20}$	$62.35_{\pm 5.39}$	$78.59_{\pm 0.75}$	$61.30_{\pm 0.53}$	$72.04_{\pm 0.13}$	$70.19_{\pm 1.51}$
SAGE	$87.03_{\pm 0.77}$	$76.41_{\pm 0.92}$	$97.24_{\pm 0.06}$	$78.22_{\pm 0.62}$	$38.84_{\pm 1.74}$	$74.15_{\pm 8.66}$	$80.55_{\pm 0.77}$	$73.95_{\pm 1.38}$	$77.06_{\pm 5.89}$	$78.42_{\pm 0.60}$	$60.74_{\pm 0.48}$	$71.06_{\pm 0.17}$	$71.79_{\pm 2.14}$
GPR-GNN	$84.28_{\pm 0.37}$	$77.15_{\pm 0.40}$	$97.27_{\pm 0.05}$	$73.61_{\pm 0.58}$	$35.24_{\pm 1.71}$	$68.76_{\pm 6.63}$	$78.24_{\pm 0.12}$	$75.52_{\pm 0.51}$	$76.08_{\pm 5.93}$	$79.51_{\pm 0.68}$	$64.36_{\pm 0.42}$	$69.34_{\pm 0.28}$	$\underline{71.96_{\pm 1.60}}$
UniMP	$87.90_{\pm 0.43}$	$75.41_{\pm 0.85}$	$97.24_{\pm 0.05}$	$80.54_{\pm 0.77}$	$37.21_{\pm 1.96}$	$69.82_{\pm 8.66}$	$81.85_{\pm 0.60}$	$75.02_{\pm 1.01}$	$77.45_{\pm 2.52}$	$79.13_{\pm 0.52}$	$59.46_{\pm 0.50}$	$71.70_{\pm 0.19}$	$71.69_{\pm 1.61}$
node2vec	$79.91_{\pm 0.16}$	$68.33_{\pm 1.59}$	$97.15_{\pm 0.05}$	$62.60_{\pm 0.49}$	$37.96_{\pm 2.49}$	$73.62_{\pm 4.83}$	$81.05_{\pm 0.65}$	$79.71_{\pm 0.70}$	$82.55_{\pm 3.45}$	$78.17_{\pm 0.65}$	$56.34_{\pm 0.36}$	$70.02_{\pm 0.11}$	$70.05_{\pm 1.35}$
GAE	$80.41_{\pm 0.33}$	$78.83_{\pm 0.86}$	$91.74_{\pm 0.90}$	$49.98_{\pm 0.53}$	$43.87_{\pm 2.10}$	$64.39_{\pm 5.87}$	$80.96_{\pm 0.48}$	$68.63_{\pm 1.35}$	$54.12_{\pm 5.76}$	$79.23_{\pm 0.91}$	$63.96_{\pm 0.39}$	$72.02_{\pm 0.08}$	$67.82_{\pm 1.61}$
BGRL	$80.24_{\pm 0.26}$	$79.64_{\pm 0.37}$	$97.16_{\pm 0.04}$	$51.75_{\pm 0.28}$	$41.37_{\pm 1.49}$	$59.77_{\pm 6.00}$	$81.10_{\pm 0.69}$	$72.05_{\pm 0.90}$	$59.80_{\pm 5.63}$	$79.01_{\pm 0.59}$	$63.45_{\pm 0.31}$	$71.47_{\pm 0.15}$	$69.08_{\pm 1.35}$
CCA-SSG	$80.11_{\pm 0.23}$	$79.42_{\pm 0.40}$	$97.30_{\pm 0.03}$	$65.44_{\pm 0.66}$	$43.08_{\pm 1.75}$	$75.53_{\pm 4.81}$	$81.30_{\pm 0.47}$	$78.16_{\pm 0.87}$	$81.96_{\pm 4.71}$	$79.32_{\pm 0.54}$	$63.73_{\pm 0.46}$	$72.04_{\pm 0.24}$	$\mathbf{73.87_{\pm 1.21}}$
GraphMAE	$80.41_{\pm 0.18}$	$80.41_{\pm 0.43}$	$97.11_{\pm 0.01}$	$49.84_{\pm 0.38}$	$37.19_{\pm 1.65}$	$65.22_{\pm 3.97}$	$78.87_{\pm 0.53}$	$67.10_{\pm 0.80}$	$56.08_{\pm 3.64}$	$78.66_{\pm 0.46}$	$64.48_{\pm 0.44}$	$71.29_{\pm 0.11}$	$68.86_{\pm 1.19}$
GraphMAE2	$80.10_{\pm 0.25}$	$79.01_{\pm 0.58}$	$97.10_{\pm 0.03}$	$50.46_{\pm 0.64}$	$36.88_{\pm 1.84}$	$61.94_{\pm 6.19}$	$78.73_{\pm 0.67}$	$61.25_{\pm 0.85}$	$54.51_{\pm 4.71}$	$79.11_{\pm 0.53}$	$64.57_{\pm 0.34}$	$71.42_{\pm 0.23}$	$68.12_{\pm 1.36}$
MaskGAE	$84.53_{\pm 0.45}$	$81.41_{\pm 0.61}$	$97.17_{\pm 0.07}$	$70.69_{\pm 0.60}$	$44.25_{\pm 1.49}$	$62.73_{\pm 7.81}$	$81.59_{\pm 0.57}$	$72.05_{\pm 0.91}$	$55.88_{\pm 4.41}$	$79.56_{\pm 0.71}$	$64.08_{\pm 0.49}$	$72.45_{\pm 0.25}$	$70.73_{\pm 1.46}$
Node4All	$79.94_{\pm 0.17}$	$78.43_{\pm 0.58}$	$97.06_{\pm 0.07}$	$71.84_{\pm 0.41}$	$42.50_{\pm 1.88}$	$75.52_{\pm 3.81}$	$79.28_{\pm 0.60}$	$78.44_{\pm 0.83}$	$76.27_{\pm 4.51}$	$78.02_{\pm 0.65}$	$62.62_{\pm 0.46}$	$68.14_{\pm 0.20}$	$71.55_{\pm 1.38}$

Table 7. Statistics of the 25 node-classification benchmarks used in the main experiment of this work.

Dataset	#Nodes	#Edges	#Features	#Classes	Feature Type	Train/Val/Test (%)	#Train	#Val	#Test	Homophily	Avg. Degree
Texas	183	741	1703	5	Binary Sparse	47.54/31.69/20.22	87	58	37	0.0609	4.05
Cornell	183	737	1703	5	Binary Sparse	47.54/32.24/20.22	87	59	37	0.1227	4.03
Wisconsin	251	1151	1703	5	Binary Sparse	47.81/31.87/20.32	120	80	51	0.1778	4.59
Chameleon	890	18598	2325	5	Binary Sparse	45.96/32.25/21.80	409	287	194	0.2361	20.9
Squirrel	2223	96219	2089	5	Binary Sparse	47.37/32.30/20.33	1053	718	452	0.2072	43.28
Wiki	2405	25597	4973	17	Unsigned Dense	14.14/42.91/42.95	340	1032	1033	0.6097	10.64
Cora	2708	13264	1433	7	Sparse	5.17/18.46/36.93	140	500	1000	0.8100	4.9
Citeseer	3327	12431	3703	6	Sparse	3.61/15.03/30.06	120	500	1000	0.7355	3.74
BlogCatalog	5196	348682	8189	6	Binary Sparse	2.31/48.85/48.85	120	2538	2538	0.4011	67.11
Actor	7600	60918	932	5	Binary Sparse	48.00/32.00/20.00	3648	2432	1520	0.2167	8.02
AmzPhoto	7650	245812	745	8	Binary Dense	2.09/48.95/48.95	160	3745	3745	0.8272	32.13
Minesweeper	10000	88804	7	2	One-hot	50.00/25.00/25.00	5000	2500	2500	0.6828	8.88
WikiCS	11701	442907	300	10	Signed Dense	4.96/15.12/49.97	580	1769	5847	0.6543	37.85
Tolokers	11758	1049758	10	2	Unsigned Dense	50.00/25.00/25.00	5879	2939	2940	0.5945	89.28
AmzComputer	13752	505474	767	10	Binary Dense	1.45/49.27/49.27	200	6776	6776	0.7772	36.76
DBLP	17716	123450	1639	4	Binary Sparse	0.45/49.77/49.77	80	8818	8818	0.8279	6.97
CoCS	18333	182121	6805	15	Sparse	1.64/49.18/49.18	300	9016	9017	0.8081	9.93
Pubmed	19717	108365	500	3	Binary Dense	0.30/2.54/5.07	60	500	1000	0.8024	5.5
FullCora	19793	146635	8710	70	Signed Dense	7.07/46.46/46.47	1400	9196	9197	0.5670	7.41
Roman Empire	22662	88516	300	18	Signed Dense	50.00/25.00/25.00	11331	5665	5666	0.0469	3.91
Amazon Ratings	24492	210592	300	5	Signed Dense	50.00/25.00/25.00	12246	6123	6123	0.3804	8.6
Deezer	28281	213785	128	2	Unsigned Dense	0.14/49.93/49.93	40	14120	14121	0.5251	7.56
CoPhysics	34493	530417	8415	5	Sparse	0.29/49.85/49.86	100	17196	17197	0.9314	15.38
Questions	48921	356001	301	2	Unsigned Dense	50.00/25.00/25.00	24460	12230	12231	0.8396	7.28
OGBN-Arxiv	169343	2484941	128	40	Signed Dense	53.70/17.60/28.70	90941	29799	48603	0.6542	14.67