TO LEARN IS TO WANDER: LEARNING ACROSS GRAPHS AND TASKS WITH RANDOM WALKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph foundation models aim to transfer across graphs, feature spaces, relational schemas, and prediction tasks, yet existing approaches typically generalize only within particular graph modalities or tasks. We propose WANDER, a graph foundation model designed to operate across these settings within *a single pretrained checkpoint*. Following the prior-predictive perspective, we formulate graph learning as completion of a partially observed graph. We realize this task-general view through a common interface based on random walks, allowing the same model to operate across homogeneous and multi-relational graphs with varying features, labels, and relational schemas. WANDER can increase its structural context at inference time without changing its learned parameters and, under suitable assumptions, universally approximates the corresponding Bayes-optimal predictor on bounded connected graphs. Empirically, a single pretrained checkpoint achieves state-of-the-art or highly competitive results across node classification, homogeneous link prediction, and knowledge-graph link prediction. Moreover, joint pretraining across graph modalities and tasks preserves performance in specialized settings while enabling positive transfer and the composition of separately learned capabilities at inference time.

1 INTRODUCTION

Graph-structured data arises across a wide range of domains, from molecules and citation networks to social systems and knowledge graphs (Pei et al., 2020; Liben-Nowell & Kleinberg, 2003; Huang et al., 2023b). These domains differ in the information associated with their graphs and in the prediction problems of interest, and graph-learning models have traditionally been trained end-to-end for a specific task setting. Graph foundation models aim to make learning on such data more general: rather than training a separate model for every graph and task, the goal is to learn models that transfer across graphs, feature spaces, relation types, and prediction problems.

Existing approaches, however, only partially realize this goal and the emerging landscape of GFMs remains fragmented, as summarized in Table 1: some generalize node prediction across attributed homogeneous graphs with arbitrary feature and label spaces (Zhao et al., 2025; Xia & Huang, 2026; Finkelshtein et al., 2026; Hayler et al., 2025; Choi et al., 2026; Eremeev et al., 2026), some others generalize link prediction across knowledge graphs with arbitrary relational vocabularies (Galkin et al., 2024; Zhang et al., 2024; Huang et al., 2025; Kim et al., 2026), and some focus on link prediction on homogeneous graphs (Dong et al., 2024; Liao et al., 2026). What is still missing is a unified formulation that spans these different graph-learning settings. This leaves two closely related questions: *what predictive function should such a general graph foundation model learn? How can a single architecture realize this function across previously unseen graphs, feature spaces, relations, and tasks?*

Table 1: Existing GFMs generalize across different subsets of graph modalities and prediction tasks. WANDER uses a single model across both node- and edge-level tasks on homogeneous and multi-relational graphs.

Graph Foundation Model Families	Input Graphs: node features	Input Graphs: edge types	Prediction Task: node property	Prediction Task: link
ULTRA (2024), FLOCK (2025)	✗	✓	✗	✓
OPENGRAPH, ANYGRAPH (2024)	✓	✗	✓	✓
TS-NET (2026), GRAPHPFN (2025)	✓	✗	✓	✗
UNILP (2024), TFMLINKER (2026)	✗	✗	✗	✓
WANDER (ours)	✓	✓	✓	✓

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This work makes three contributions toward answering these questions:

(1) A task-general predictive objective. We answer the first question through a probabilistic view of graph learning. Following the prior-predictive perspective of TABPFN (Hollmann et al., 2023), we view an observed graph as a partially specified realization of an underlying complete graph drawn from a graph distribution. Node- and edge-level prediction problems can then be expressed as completion tasks on this partially observed graph: completing node labels corresponds to node classification, while completing edges corresponds to link prediction. More generally, each task defines a query about an unobserved component of the graph, with the Bayes-optimal prediction given by the posterior expectation of the query-dependent target conditioned on the observation. This gives one predictive function across graph-learning tasks.

(2) A common computational architecture. Realizing this task-general predictor requires a common interface that can operate across varying graph structures, feature spaces, relations, and label spaces. We propose WANDER, which uses random walks as this interface. Random walks have a long history in graph learning (Perozzi et al., 2014; Grover & Leskovec, 2016; Tönshoff et al., 2021; Tan et al., 2023; Kim et al., 2025), and have recently re-emerged as a computational primitive in knowledge graph foundation models (Kim et al., 2026). WANDER extends this interface across graph settings and prediction tasks by representing features, labels, and structure through separate channels and processing their information along sampled walks, enabling a single architecture to operate on homogeneous and multi-relational graphs, accommodate varying feature and label spaces, and answer both node- and edge-level queries. It has three properties particularly useful for a foundation model.

- Unlike message-passing GFMs (Finkelshtein et al., 2026; Eremeev et al., 2026), whose structural context is fixed by network depth, WANDER can flexibly adjust its structural context at inference time by varying its random-walk sampling strategy, without modifying its learned parameters.
- Random-walks can capture structural properties beyond standard message passing architectures. We formalize this by showing that, under suitable capacity assumptions, WANDER is a universal approximator of the Bayes-optimal graph predictor on bounded connected graphs (Theorem 3).
- WANDER is permutation equivariant in distribution, allowing stochastic symmetry breaking while respecting graph symmetries at the distributional level to facilitate generalization (Proposition 4).

(3) A generalist model with compositional and adaptable capabilities. Empirically, we evaluate a *single* pretrained WANDER checkpoint across node classification, homogeneous link prediction, and knowledge-graph link prediction, obtaining state-of-the-art or highly competitive results across all three settings. Beyond this breadth, joint pretraining yields *positive transfer* across graph-learning settings. Further to this, WANDER can *combine separately learned capabilities* to solve new task settings. Finally, its random-walk interface supports *adaptation* without parameter updates: expanding or targeting the sampled context improves long-range reasoning on synthetic and real-world graphs. Together, these findings show benefits of unification beyond task coverage alone.

2 RELATED WORK

Graph foundation models for node property prediction. Node property prediction has traditionally relied on message-passing neural networks such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019), which are typically trained end-to-end on a single graph. Early graph foundation models aim for architectures that transfer across feature and label spaces. GraphAny (Zhao et al., 2025) combines predictions from multiple linear graph filters through learned attention that transfer across graphs. TS-GNN (Finkelshtein et al., 2026) formalizes the symmetry requirements of node-level graph foundation models under node, feature, and label permutations, and proposes a corresponding symmetry-preserving architecture. Another line of work builds on *tabular* foundation models (Hollmann et al., 2023; 2025; Qu et al., 2025), which perform *in-context learning* from labeled rows at inference time. G2T-FM (Eremeev et al., 2025) and TAG (Hayler et al., 2025) encode graphs as node-level tables for in-context prediction with pretrained tabular foundation models. NodePFN (Choi et al., 2026) extends TabPFN (Hollmann et al., 2023) to graphs through pretraining on synthetic graphs, retaining a single representation per node and a fixed maximum feature dimension. GraphPFN (Eremeev et al., 2026) builds on LimiX (Zhang et al., 2025) with graph-aware aggregation between feature-tokenized blocks, more naturally accommodating varying feature dimensions across graphs. WANDER instead uses random walks as its structural interface for in-context learning, accommodating varying feature spaces and edge types.

Graph foundation models for link prediction. Classical approaches for link prediction over *attributed graphs*, where nodes carry features and edges are typically of a single type, range from GNN-based encoders (Kipf & Welling, 2017) to link-specific architectures such as SEAL (Zhang & Chen, 2018) and BUDDY (Chamberlain et al., 2023). More recent foundation models aim to replace graph-specific training with transferable predictors that generalize across unseen graphs, including Prodigy (Huang et al., 2023a), OFA (Liu et al., 2024), UniLP (Dong et al., 2024), OpenGraph (Xia et al., 2024), and AnyGraph (Xia & Huang, 2026). TFMLinker (Liao et al., 2026) instead tabularizes link-centric graph information for in-context prediction with a pretrained tabular foundation model. In parallel, *knowledge graphs* link prediction has developed a separate family of architectures, from R-GCN (Schlichtkrull et al., 2018) and CompGCN (Vashishth et al., 2020) to query-conditioned models such as NBFNet (Zhu et al., 2021) and C-MPNNs (Huang et al., 2023b). Knowledge graph foundation models (KGFMs) further extend this setting to fully inductive transfer, where both entities and relation types can be unseen at test time. ULTRA (Galkin et al., 2024), MOTIF (Huang et al., 2025), and TRIX (Zhang et al., 2024) achieve this primarily through relation- and entity-level message passing, while FLOCK (Kim et al., 2026) instead uses random walks as the interface for structural reasoning. These models provide broad transfer within either homogeneous attributed graphs or relational graphs, but not a common model across both regimes and prediction tasks.

Random walks for graph learning. Random walks have long been used to capture graph structure, with DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) learning node embeddings from sampled walk sequences. Later methods use walks more directly as graph representations: CRaWL (Tönshoff et al., 2021) processes collections of walks with neural networks, while WalkLM (Tan et al., 2023) incorporates node attributes into walk sequences for language-model-based processing. RWNN (Kim et al., 2025) further treats anonymized random walks as a general computational interface for graph learning. FLOCK (Kim et al., 2026) extends this paradigm to knowledge graphs for zero-shot link prediction. WANDER builds on this line by using the same walk-based graph computation across graph settings and tasks, while its in-context architecture accommodates node attributes, labels, and relational edge types within a single model.

3 PRELIMINARIES

Attributed edge-typed graphs. An *attributed edge-typed graph* is a tuple $G = (V, E, R, X, Y)$, where V is the set of nodes, R is the set of relation types, $E \subseteq V \times R \times V$ is the set of typed edges, $X \in \{\emptyset\} \cup \mathbb{R}^{V \times d_X}$ is the matrix of d_X-dimensional node features, and $Y \in \{\emptyset\} \cup ([d_Y] \cup \{\perp\})^V$ contains the node labels, with $d_Y \in \mathbb{N}$ denoting the number of classes. For any $k \in \mathbb{N}$, we write $[k] := \{1, \ldots, k\}$. We call a node $v \in V$ *unlabeled* if $Y_v = \perp$. Features and labels may both be absent. We define $\mathbb{G}_{n,m,b}$ as the set of attributed edge-typed graphs with at most n nodes, m relation types, and b feature and label dimensions and $\mathbb{G}^\star_{n,m,b} \subset \mathbb{G}_{n,m,b}$ as the set of *fully specified graphs* without unlabeled nodes. In the following, we refer to attributed edge-typed graphs simply as *graphs*.

Partial observations and graph isomorphism. We write $\pi = (\pi_V, \pi_R, \pi_X, \pi_Y)$ for bijective relabelings of nodes, relation types, feature dimensions, and class labels, respectively. We extend π_Y by $\pi_Y(\perp) = \perp$ and denote the relabeled graph by $\pi(G)$. A graph G is a *partial observation* of G' (written $G \preceq G'$) if, up to such a relabeling, it can be obtained from G' by deleting a subset of its edges and masking a subset of its node labels with $\perp$. Two graphs are isomorphic (written $G \cong G'$) if $G \preceq G'$ and $G' \preceq G$.

Modeling posterior graph distributions. In this work we are interested in predicting missing labels or edges in an (incomplete) observed graph $G_{\text{obs}} \in \mathbb{G}_{n,m,b}$ conditioned on a prior distribution over graphs. Formally, let p be a probability distribution over the isomorphism classes of $\mathbb{G}^\star_{n,m,b}$. In the following, we slightly abuse notation by identifying an isomorphism class with any of its representatives and write G interchangeably for a graph and its isomorphism class. We denote by $p(\cdot \mid G_{\text{obs}})$ the conditional distribution over graph completions consistent with G_{obs}.

However, rather than explicitly representing the full posterior distribution, we aim to model posterior expectations of query-dependent target random variables. Formally, we define a *link prediction query* $q = (h_q, r_q, ?)$ consisting of a query head $h_q \in V_{\text{obs}}$ and query relation $r_q \in R_{\text{obs}}$, and a *label prediction query* $q = V_q \subset V_{\text{obs}}$ consisting of a set of unlabeled nodes. For $G^\star \sim p(\cdot \mid G_{obs})$ we then define the *query target* random variable $T_q(G^\star)$, where

$$T_q(G^\star) = \begin{cases} (\mathbf{1}_{(h_q, r_q, t) \in E^\star})_{t \in V} & \text{if } q = (h_q, r_q, ?) \quad \textit{Link Prediction} \\ ((\mathbf{1}_{Y^\star_v = c})_{c \in [d_Y]})_{v \in V_q} & \text{if } q = V_q \quad \textit{Node Classification} \end{cases}$$

The Bayes-optimal predictor for query q is therefore $M_p^\star : (G_{\text{obs}}, q) \mapsto \mathbb{E}_{G^\star \sim p(\cdot|G_{\text{obs}})}[T_q(G^\star)]$.

Random walks. Given a graph G, we define a walk η of length S in G as a sequence $\eta = (v_0, r_1^{\alpha_1}, v_1, \ldots, r_S^{\alpha_S}, v_S)$, where $v_j \in V$, $r_j \in R$, and $(v_{j-1}, r_j, v_j) \in E$ if $\alpha_j = 1$ and $(v_j, r_j, v_{j-1}) \in E$ if $\alpha_j = -1$. We denote the set of such walks of arbitrary length by $\mathcal{W}(G)$. A random-walk algorithm A associates with every graph-query pair (G, q) a random variable $A(G, q)$ taking values in $\mathcal{W}(G)$.

4 Learning the Bayes-Optimal Graph Predictor

This section characterizes the task-general predictor introduced above. We establish Bayes-level compatibility across graph tasks, characterize when expanding the graph prior can improve target performance, and formalize the predictor's permutation symmetries. Proofs are in Appendix C.

Learning across tasks. Any graph foundation model can be trained to approximate $M_p^\star$ using samples $G^\star \sim p$, incomplete observations $G_{\text{obs}} \preceq G^\star$, and queries q, by minimizing

$$\mathcal{L}_p(\theta) = \mathbb{E}_p[\ell_q(M_\theta(G_{\text{obs}}, q), T_q(G^\star))] \,.$$

Here, the expectation includes observation and query sampling, consistent with the posterior defined above. Categorical cross-entropy for node classification and binary cross-entropy for link prediction both have $M_p^\star$ as their population minimizer. Moreover, restricting this objective to one task leaves its optimal predictions unchanged: tasks merely query different components of the same posterior over graph completions. Thus, assuming sufficient capacity, *joint training has no Bayes-level disadvantage over single-task training*.

Learning across diverse graphs. Broadening the pretraining prior can improve coverage of target graphs, but may also introduce graph families whose predictions are unsuitable for some targets. Theorem 1 shows that expansion cannot hurt if the observation rules out the added family wherever its predictions would be worse. More generally, it guarantees improvement when the family's posterior-weighted predictive benefits outweigh its harm.

Consider two graph distributions p and r and define p_λ by sampling from p with probability $(1-\lambda)$ and from r otherwise. Further, define $\beta_{\lambda,r}(G_{\text{obs}})$ as the posterior weight of r, i.e., the probability that the complete graph stems from r given observation G_{obs}. Then we can write

$$M_{p_\lambda}^\star(G_{\text{obs}}, q) = \big(1 - \beta_{\lambda,r}(G_{\text{obs}})\big) M_p^\star(G_{\text{obs}}, q) + \beta_{\lambda,r}(G_{\text{obs}}) M_r^\star(G_{\text{obs}}, q).$$

Thus, the influence of an added family depends on its posterior relevance to each target observation. Let now ν be a target distribution over complete graphs, observations, and queries, and define the risk of a classifier M as $R_\nu(M) := \mathbb{E}_\nu[\ell_q(M(G_{\text{obs}}, q), T_q(G^\star))]$.

Theorem 1. *Assume that ℓ_q is convex in its prediction argument, that $p(\cdot \mid G_{\text{obs}})$ is defined ν-almost surely, and that the risks below are finite. Define the added family's conditional risk difference relative to the existing predictor as*

$$\Delta_\nu(G_{\text{obs}}, q) := \mathbb{E}_\nu\Big[\ell_q(M_r^\star(G_{\text{obs}}, q), T_q(G^\star)) - \ell_q\big(M_p^\star(G_{\text{obs}}, q), T_q(G^\star)\big) \mid G_{\text{obs}}, q\Big].$$

On observations with zero marginal probability density under r, set $M_r^\star = M_p^\star$. Then

$$R_\nu(M_{p_\lambda}^\star) - R_\nu(M_p^\star) \le \mathbb{E}_\nu[\beta_{\lambda,r}(G_{\text{obs}}) \cdot \Delta_\nu(G_{\text{obs}}, q)] \,.$$

Thus, expanding the prior helps when the added graph family receives posterior mass where its predictions are beneficial and cannot hurt where it is ruled out by the observation.

Permutation equivariance. Lastly, we formalize the symmetries of the Bayes-optimal predictor under a relabeling π as introduced in Section 3. We write $\pi(q)$ for the correspondingly relabeled query and π_{out} for the induced relabeling of output indices: node indices transform according to π_V and, for node classification, class indices additionally transform according to π_Y.

Proposition 2. *For any graph prior p over isomorphism classes, the Bayes-optimal predictor is permutation equivariant: $M_p^\star(\pi(G), \pi(q)) = \pi_{\text{out}}\big(M_p^\star(G, q)\big)$.*

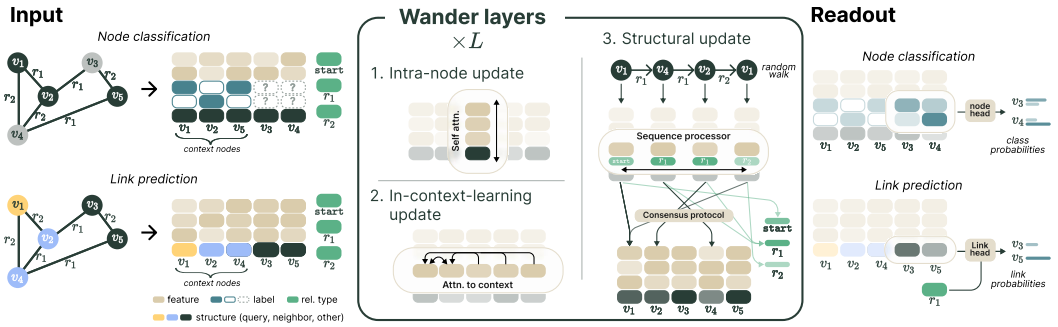


Figure 1: **Overview of WANDER.** Each layer combines (1) intra-node attention across feature, label, and structure tokens; (2) in-context attention from each node to context nodes within the same channel; and (3) a stochastic structural update that processes sampled walks and aggregates proposals through a consensus protocol. After L layers, task-specific readouts map the query node's label tokens to class probabilities or each candidate node's structure token to a link probability.

5 METHODOLOGY

In this section, we present the WANDER framework. The implementation details for the experiments presented in Section 7 are provided in Appendix A. An overview is provided in Figure 1. We additionally include a comparison to existing foundation models in Appendix B.

Input and internal representation. Let $G = (V, E, R, X, Y)$ and query q be the input to WANDER. Depending on the prediction task, we define the set of *context nodes* as

$$V_{\text{context}} = \begin{cases} \{v \in V \mid v = h_q \text{ or } (h_q, r_q, v) \in E\}, & \text{if } q = (h_q, r_q, ?), \\ \{v \in V \mid Y_v \neq \perp\}, & \text{if } q = V_q. \end{cases}$$

Thus, context nodes correspond to observed positive examples for link prediction and observed labeled examples for node prediction.

Internally, WANDER maintains node states $\mathbf{v} : V \times [d_X + d_Y + 1] \to \mathbb{R}^d$ and relation states $\mathbf{r} : R \to \mathbb{R}^d$. Each node has one token for every feature dimension and label class, together with a single *structure token*; we refer to these indices collectively as *channels*. At the beginning of a forward pass, feature and label tokens are initialized from their observed values, structure tokens encode the node's role relative to the query, and relation types all receive the same learnable initial state. We denote the resulting states by $\mathbf{v}^{(0)}$ and $\mathbf{r}^{(0)}$.

Single layer. Each layer $l \in \{0, \ldots, L-1\}$ updates these states through three operations.

Intra-node update. For each node $v \in V$, an intra-node processor μ_θ is applied to the set of tokens $\mathbf{v}^{(l)}(v, \cdot)$ corresponding to v, yielding intermediate updates $\mathbf{v}^{(l)'}(v, \cdot)$. This allows feature, label, and structural information associated with the same node to interact.

In-context-learning update. For each channel $i \in [d_X + d_Y + 1]$, each token $\mathbf{v}^{(l)'}(v, i)$ attends to the context tokens $\{\mathbf{v}^{(l)'}(u, i) \mid u \in V_{\text{context}}\}$, yielding intermediate states $\mathbf{v}^{(l)''}(v, i)$. This provides the in-context learning mechanism through which observed examples can inform the query representations.

Stochastic structural update. Finally, graph structure is incorporated through random walks. This update consists of four steps: First, a *random walk algorithm* A produces a set of walks $\eta_1, \ldots, \eta_K$ on G. Next, a *recording protocol* $w : \eta_j \mapsto \{Z_{j,i} \mid i \in [d_X + d_Y + 1]\}$ broadcasts each walk to one walk recording per channel i. Then, a *sequence processor* $\nu_\theta : Z_{j,i} \mapsto Z'_{j,i}$ is applied to each sequence independently. Since the same state may be encountered by several walks, a permutation-invariant *consensus protocol* aggregates its proposals, yielding updated states $\mathbf{v}^{(l+1)}(\cdot, \cdot), \mathbf{r}^{(l+1)}(\cdot)$.

We require the recording protocol w to be equivariant to permutations of node and relation identities. All channel-wise modules are shared across channels of the same type (feature, label and structure).

Readout. After L layers, the same representation supports the different prediction tasks. For a node-prediction query $q = V_q$, the label tokens of $v \in V_q$ are passed to a node head, producing

class probabilities $M_\theta(G, q) \in [0, 1]^{V_q \times d_Y}$. For a link-prediction query $q = (h_q, r_q, ?)$, the structure token of each candidate node, together with the state of the query relation r_q, is passed to a link head, producing $M_\theta(G, q) \in [0, 1]^V$.

6 THEORETICAL ANALYSIS OF WANDER

This section establishes the model's main theoretical guarantees, namely universality and permutation equivariance. All proofs can be found in Appendix C.

Universality. We first formalize universality with respect to a graph prior. We here restrict attention to *admissible* graph priors, under which the continuous graph attributes admit a continuous mapping within each discrete isomorphism class. A formal definition is given in Appendix C. We call a model architecture M a *universal graph prior approximator* if for any admissible prior p over isomorphism classes of $\mathbb{G}^\star_{n,m,b}$ and any $\epsilon, \delta > 0$, there exists a choice of network parameters θ and a finite sampling budget $K \in \mathbb{N}$ such that

$$\Pr\left[\|M_{\theta,K}(G, q) - M^\star_p(G, q)\| < \epsilon\right] > 1 - \delta$$

for all connected observations $G \in \mathcal{G}_{n,m,b}$ in the support of p and all valid queries q.

Theorem 3. *Fix $n, m, b \in \mathbb{N}$. With at least two layers, a walk length of at least n, a sufficiently uniform random walk sampler, and sufficient hidden dimension and capacity of all modules*[1]*,* WANDER *is a universal graph prior approximator for graph priors over $\mathbb{G}^\star_{n,m,b}$.*

The result concerns the general WANDER framework under the capacity assumptions of Appendix C, rather than the finite implementation used in our experiments. For graphs with at most n nodes, a walk length of n suffices. The number of sampled walks, however, can be increased arbitrarily at inference time without changing the learned parameters, allowing the sampling budget required by the approximation guarantee. Unlike the results of Kim et al. (2025; 2026), the required walk length need not scale with the graph's cover time.

Permutation equivariance. A graph foundation model intended to approximate the Bayes-optimal predictor $M^\star_p$ should respect the same symmetries (see Proposition 2) by design. For the stochastic computation of WANDER, we require *equivariance in distribution*. We call a random-walk algorithm A permutation invariant if, for every G, q, η, and π,

$$\Pr[A(G, q) = \eta] = \Pr[A(\pi(G), \pi(q)) = \pi(\eta)],$$

and a stochastic model M_θ permutation equivariant in distribution if

$$M_\theta(\pi(G), \pi(q)) \overset{d}{=} \pi_{\text{out}}(M_\theta(G, q)).$$

Proposition 4. *Suppose that the random-walk algorithm A is permutation invariant and that the intra-node processor μ_θ is equivariant in distribution with respect to permutations of the feature and label channels. Then every* WANDER *model is permutation equivariant in distribution with respect to node, relation, feature, and label permutations.*

7 EXPERIMENTS

We organize our experiments around four main questions. (**Q1**) Can a single WANDER model remain competitive with foundation models specialized to individual graph-learning settings? (**Q2**) Does joint pretraining across these settings cause interference or enable positive transfer? (**Q3**) Can WANDER combine separately learned capabilities during inference? (**Q4**) Can WANDER adapt its structural computation at inference time to different graph and task requirements? Additional experiments on runtime, structural expressivity, finetuning, pretraining priors, and architecture ablations are provided in Appendices D.10–D.14.

7.1 GENERALIST PERFORMANCE ACROSS GRAPH LEARNING TASKS (Q1)

To answer **Q1**, we train WANDER across a deliberately diverse graph prior, spanning multiple tasks and evaluate on three standard graph-learning regimes represented in our benchmark suite: knowledge-graph link prediction, node classification on simple graphs, and homogeneous link prediction.

[1] See Assumptions (A1)-(A5) in Appendix C.

Table 2: Knowledge graph link prediction (MRR and Hits@10).

	Inductive (e,r)		Inductive (e)		Transductive		Mean		Mean rank	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
# graphs	23		18		13					
ULTRA	0.345	0.513	0.431	0.566	0.312	0.458	0.366	0.518	3.250	3.417
TRIX	0.368	0.540	0.455	0.592	0.339	0.500	0.390	0.548	2.657	2.750
FLOCK	0.369	0.554	0.456	0.604	0.340	0.509	0.391	0.560	2.343	2.222
WANDER	**0.378**	**0.558**	**0.460**	**0.609**	**0.353**	**0.516**	**0.399**	**0.565**	**1.750**	**1.611**

Pretraining prior. Following the prior-data fitted network paradigm (Müller et al., 2022; Hollmann et al., 2023), we construct most of WANDER's pretraining distribution synthetically. For attributed graphs, we combine two approaches inspired by NODEPFN (Choi et al., 2026) and GRAPHPFN (Eremeev et al., 2026): NODEPFN generates attributes through an SCM before sampling graph structure using stochastic block or Erdős–Rényi models (Holland et al., 1983; Erdős & Rényi, 1959), while GRAPHPFN first samples graph structure and subsequently generates attributes with a graph-aware SCM. We also diversify the structural prior using Watts–Strogatz (Watts & Strogatz, 1998), random geometric (Gilbert, 1961), and kernel-based graph models (Lovász & Szegedy, 2006). These synthetic graphs provide both node-classification and homogeneous link-prediction episodes. Since realistic synthetic generation of multi-relational graphs is less straightforward, we additionally include FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), and CoDEx Medium (Safavi & Koutra, 2020) for multi-relational link-prediction pretraining. Details and discussion of the pretraining prior are provided in Appendix D.1.

Knowledge graph link prediction. We evaluate whether WANDER can match foundation models specialized to knowledge graph reasoning by comparing against ULTRA (Galkin et al., 2024), TRIX (Zhang et al., 2024), and FLOCK (Kim et al., 2026) in the zero-shot setting. We consider entity prediction, where, given a head entity h and relation r, the model ranks candidate tails for the query $(h, r, ?)$. All models are pretrained on the same three knowledge graphs included in WANDER's pretraining. We evaluate on 54 knowledge graphs spanning diverse domains and three settings: inductive on entities and relations (inductive e, r), inductive on entities (inductive e), and transductive. Note that in our experiment, all three settings are in principle indifferent, as all models encounter unseen relations in the inference graph and align with previous work (Galkin et al., 2024; Kim et al., 2026). We use the filtered ranking protocol (Bordes et al., 2013) and report mean reciprocal rank (MRR) and Hits@10; per-dataset results are given in Table 12. As shown in Table 2, WANDER achieves the strongest overall results, with a mean MRR of 0.399 and Hits@10 of 0.565, compared with 0.391 and 0.560 for the strongest specialized baseline FLOCK.

Homogeneous link prediction. Next, we evaluate WANDER on homogeneous link prediction, where, given a source node, the model ranks all other nodes as candidate link targets. We consider ten datasets from diverse domains (see Appendix D.4) and report Recall@20. We compare against three classes of baselines: the classical heuristics Common Neighbors (CN) (Liben-Nowell & Kleinberg, 2003) and Resource Allocation (RA) (Zhou et al., 2009); the end-to-end link-prediction models BUDDY (Chamberlain et al., 2023) and NBFNET (Zhu et al., 2021), tuned per dataset; and the graph foundation model ANYGRAPH (Xia & Huang, 2026). Two recent foundation models, UNILP (Dong et al., 2024) and TFMLINKER (Liao et al., 2026), also target homogeneous link prediction but omit node features and score node pairs individually. In contrast, WANDER conditions on a source node and jointly scores all candidate targets. Because pairwise scoring makes full-candidate Recall@20 impractical at this scale, we compare with UNILP separately in Appendix D.4. Table 3 summarizes the results. WANDER achieves the highest mean Recall@20 of 43.39%, compared with 38.82% for the strongest end-to-end baseline, NBFNET, and 27.85% for ANYGRAPH. Thus, WANDER remains competitive with dataset-specific models while substantially outperforming the existing foundation-model baseline.

Node classification on simple graphs. Finally, to evaluate whether WANDER can match models specialized to node classification, we evaluate it in the zero-shot setting on a test set of 26 attributed simple graphs from diverse domains (see Appendix D.5 for details). We consider a strongly tuned GCN (Kipf & Welling, 2017) as a supervised baseline, performing a per-dataset grid search over learning rate, layer normalization, dropout, and skip connections. We further compare against the graph foundation models GRAPHANY (Zhao et al., 2025), NODEPFN (Choi et al., 2026), and

Table 3: Homogeneous link prediction (Recall@20 %).

	CiteSeer	Cora	PubMed	CS	DDI	Products Home	P2P Gnutella	Email Enron	Proteins Spec1	SOC Epinions	Mean	Mean rank
CN	29.83	33.30	11.02	49.09	3.45	56.94	1.46	52.88	15.65	15.73	26.94	5.00
RA	30.26	34.99	11.12	55.63	4.85	63.21	1.29	63.07	20.67	16.37	30.15	4.10
BUDDY	44.32	40.03	25.72	48.28	9.42	58.66	7.10	50.18	13.67	10.78	30.82	4.00
NBFNET	34.21	45.98	22.88	62.17	**16.34**	68.19	**9.29**	69.56	**39.85**	19.70	38.82	1.90
ANYGRAPH	32.35	44.98	15.59	47.87	7.67	55.04	5.80	40.75	15.36	13.06	27.85	4.70
WANDER	**65.88**	**56.78**	**31.73**	**64.62**	12.98	**71.12**	9.06	**69.78**	31.40	**20.59**	**43.39**	**1.30**

Table 4: Node classification (test accuracy %, averaged over all available splits).

	Web-KB	Air-ports	Wiki filt.	Actor	Plane-toid	Ama-zon	Wiki CS	Cit. Full	Roman Emp.	Amz. Rat.	Coau-thor	HM Cat.	Pokec 100k	City Netw.	Mean	Mean rank
# graphs	3	3	2	1	3	2	1	2	1	1	2	1	1	3		
mean # nodes	206	573	1.6k	7.6k	8.6k	10.7k	11.7k	18.8k	22.7k	24.5k	26.4k	46.6k	100k	180k		
GCN	51.3	46.9	37.4	33.1	70.8	87.9	**79.4**	66.2	**79.6**	**51.2**	92.0	**66.4**	78.4	38.2	60.64	3.12
TABPFNV3	73.3	32.0	44.0	**39.0**	67.2	83.1	73.8	59.2	65.7	49.1	90.8	42.1	43.2	47.0	58.66	3.69
GRAPHANY	63.0	40.2	29.2	29.1	74.7	86.9	75.4	64.2	64.2	42.5	91.7	37.1	44.0	18.9	54.85	3.96
NODEPFN*	74.5	59.1	44.3	32.7	**76.2**	85.7	76.0	—	48.8	44.2	91.7	—	78.1	44.6	—	—
GRAPHPFN	73.6	**65.2**	**49.5**	38.7	71.9	88.2	79.1	66.0	67.8	48.6	91.1	55.2	83.8	45.1	66.55	2.42
WANDER	**78.5**	64.0	45.9	36.4	75.4	**88.3**	79.3	**69.9**	65.4	49.0	**92.5**	48.7	**86.1**	**53.8**	**68.20**	**1.73**

*NodePFN is omitted from columns containing datasets that exceed its maximum number of supported classes.

GRAPHPFN (Eremeev et al., 2026), as well as the tabular foundation model TABPFNv3 (Grinsztajn et al., 2026). While GRAPHANY was pretrained on the ogbn-arxiv graph from Hu et al. (2020), all other foundation models were pretrained on synthetic priors. Table 4 summarizes the results, with datasets from the same family grouped into a single column. Overall, WANDER closely matches the performance of the strongest node-classification foundation model, GRAPHPFN, achieving a mean accuracy of 68.20%, surpassing 66.55%.

7.2 TRANSFER THROUGH JOINT TRAINING (Q2)

We next investigate whether joint pretraining across tasks causes interference or enables positive transfer. For each task, we train a single-task variant for exactly the number of steps allocated to that task during joint pretraining. This keeps the number of task-specific updates fixed, while the jointly trained model additionally receives updates from the other tasks. As shown in Table 5, knowledge-graph link prediction as well as node classification are essentially unaffected by joint training. For homogeneous link prediction, however, joint training improves Recall@20 from 40.37% to 43.39%. This suggests that joint pretraining does not necessarily trade off task specialization and can instead yield positive transfer across graph-learning settings. The effect is particularly pronounced for homogeneous link prediction, which shares the link-prediction objective with knowledge graphs while operating on the homogeneous attributed graphs also encountered in node classification.

Table 5: Joint vs. single-task training.

	KG LP (MRR)	Hom. LP (R@20)	NC (Acc.)
WANDER	0.399	43.39	68.20
WANDER *one task*	0.398	40.37	67.69

7.3 COMPOSING SEPARATELY LEARNED CAPABILITIES (Q3)

Table 6: KG entity prediction (MRR) with vs. without node features and edge types.

edge types	features	ConceptNet100k	WN-v1-WN-v4	WN-18RR
✗	✗	0.135	0.162	0.116
✗	✓	0.202	0.273	0.163
✓	✗	0.258	0.626	0.550
✓	✓	**0.334**	**0.637**	**0.564**

During pretraining, WANDER encounters node features on simple graphs and edge types on knowledge graphs, but never both simultaneously. To test whether it can combine these separately learned capabilities, we augment several knowledge graphs with node features obtained by embedding entity names using MiniLM (Wang et al., 2020). As shown in Table 6, both sources of information improve entity prediction individually, while using them jointly performs best on all three benchmarks. This indicates that WANDER can compositionally generalize to attributed, edge-typed graphs, even though this type of graph was never encountered during training.

7.4 Inference-Time Adaptation (Q4)

A distinctive property of WANDER is that its structural computation remains adjustable after training: we can vary the number and length of sampled walks, as well as the sampling mechanism, without changing learned parameters. We investigate whether this flexibility enables adaptation to different structural requirements at inference time (**Q4**).

Adapting the reasoning range. We first consider GRIDS, a synthetic node-classification task designed to require long-range propagation. It consists of five disconnected 25×25 grids, one per class, with only two randomly labeled nodes per grid; test nodes are on average 11 hops from their nearest labeled node. Figure 2 shows the predicted true-class probability when evaluating the same pretrained model with walk lengths 4, 16, and 64. Increasing the walk length progressively propagates information from the labeled context nodes across the graph, raising accuracy from 56.6% at length 4 to 96.6% at length 64. At this setting, WANDER also substantially outperforms all GFM baselines as shown in Table 7.

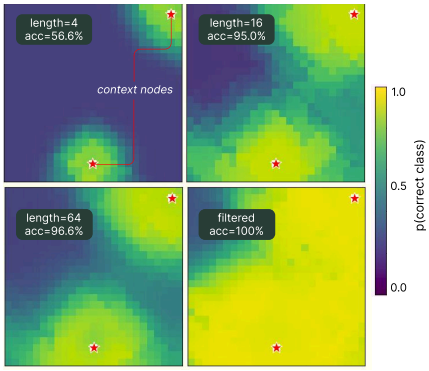


Figure 2: Predicted true-class probability on single grid taken from GRIDS for increasing walk lengths and context-aware walk filtering.

Inference-time adaptation is not limited to changing the walk budget. As a simple example, we additionally modify the sampling mechanism by retaining only length-64 walks that visit at least one labeled context node. This targeted sampling further increases accuracy on GRIDS to 100%, illustrating that WANDER can adapt not only *how much* but also *which* information is collected.

Table 7: Accuracy on GRIDS and four real-world graphs without node features.

	GRIDS	Real-world
GRAPHANY	20.0	34.0
NODEPFN	26.6	41.2
GRAPHPFN	23.3	60.1
WANDER	96.6	76.5
WANDER (filt.)	**100.0**	**78.3**

Long-range supervision in real-world graphs. We next test whether the ability to exploit distant supervision extends beyond the synthetic setting. We consider PubMed, Full DBLP, Coauthor CS, and Coauthor Physics, whose training splits provide comparatively sparse label supervision. We remove all node features, leaving graph structure and observed training labels as the only predictive signals. As shown in Table 7, WANDER achieves the best performance across the four datasets. This suggests that the long-range reasoning capability highlighted by GRIDS is also useful on real-world graphs with sparse supervision.

8 Conclusion

We introduced WANDER, a graph foundation model that uses random walks as a shared interface across graph modalities and prediction tasks. By framing graph learning as completion of a partially observed graph, a single pretrained model supports node- and edge-level prediction on homogeneous and multi-relational graphs. Under suitable assumptions, WANDER universally approximates the corresponding Bayes-optimal predictor. A single checkpoint achieves state-of-the-art or highly competitive performance across node classification, homogeneous link prediction, and knowledge-graph link prediction. More broadly, our results suggest that graph foundation models need not be organized around a fixed modality or task: a shared interface can support capabilities learned in one setting and reused in another, opening a path toward composing them across settings.

The current pretraining prior remains limited: most attributed graphs are synthetic, and multi-relational pretraining uses only three real-world knowledge graphs. Inference can also be costly, particularly with large random-walk budgets and repeated sampling. Reducing these budgets speeds up prediction without retraining, but may reduce predictive performance. Future work should broaden the prior, particularly toward graphs combining node features and edge types, extend the framework to node regression and graph classification, and develop more efficient, adaptive sampling strategies.

AI USE STATEMENT

We used generative AI tools for editing parts of the manuscript, implementation and coding support, and internal review of the paper. AI-assisted literature suggestions were assessed by the authors, and AI-assisted code was reviewed and tested. All AI-assisted text and review suggestions were critically evaluated and revised by the authors.

REPRODUCIBILITY STATEMENT

We provide an anonymous code repository at `https://anonymous.4open.science/r/wander-6BEC`. Appendix A specifies the model architecture, random-walk sampler, recording protocol, and training losses. Appendix D documents the pretraining distribution, hyperparameters, datasets, evaluation protocols, and additional experiments; complete proofs of the theoretical results are given in Appendix C.

REFERENCES

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. Graphland: Evaluating graph machine learning models on diverse industrial data. *Advances in Neural Information Processing Systems*, 38, 2026.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *International Conference on Learning Representations*, 2023.

Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

Jeongwhan Choi, Jongwoo Kim, Woosung Kang, and Noseong Park. Learning posterior predictive distributions for node classification from synthetic graph priors. In *International Conference on Learning Representations*, 2026.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 110–121, 2018.

Kaiwen Dong, Haitao Mao, Zhichun Guo, and Nitesh V Chawla. Universal link predictor by in-context learning on graphs. *arXiv preprint arXiv:2402.07738*, 2024.

Paul Erdős and Alfréd Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6: 290–297, 1959.

Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models. *arXiv preprint arXiv:2508.20906*, 2025.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. GraphPFN: A prior-data fitted graph foundation model. In *International Conference on Machine Learning*, 2026.

Ben Finkelshtein, Ismail Ilkan Ceylan, Michael Bronstein, and Ron Levie. Equivariance everywhere all at once: A recipe for graph foundation models. *Advances in Neural Information Processing Systems*, 38:30377–30434, 2026.

Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L. Hamilton. NodePiece: Compositional and parameter-efficient representations of large knowledge graphs. In *International Conference on Learning Representations*, 2022.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *ICLR*, 2024.

Edward N Gilbert. Random plane networks. *Journal of the society for industrial and applied mathematics*, 9(4):533–543, 1961.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. Tabpfn-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, 2016.

Adrian Hayler, Xingyue Huang, İsmail İlkan Ceylan, Michael Bronstein, and Ben Finkelshtein. Bringing graphs to the table: Zero-shot node classification via tabular foundation models. *arXiv preprint arXiv:2509.07143*, 2025. doi: 10.48550/arXiv.2509.07143.

Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *elife*, 6:e26726, 2017.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

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

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. PRODIGY: Enabling in-context learning over graphs. In *NeurIPS*, 2023a.

Xingyue Huang, Miguel Romero, Ismail Ceylan, and Pablo Barceló. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. *Advances in Neural Information Processing Systems*, 36:19714–19748, 2023b.

Xingyue Huang, Pablo Barceló, Michael M. Bronstein, İsmail İlkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? In *ICML*, 2025.

Jinwoo Kim, Olga Zaghen, Ayhan Suleymanzade, Youngmin Ryou, and Seunghoon Hong. Revisiting random walks for learning on graphs. In *International Conference on Learning Representations*, volume 2025, pp. 82497–82547, 2025.

Jinwoo Kim, Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and İsmail İlkan Ceylan. Flock: A knowledge graph foundation model via learning on random walks. In *International Conference on Learning Representations*, 2026.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. Ingram: Inductive knowledge graph embedding via relation graphs. In *International conference on machine learning*, pp. 18796–18809. PMLR, 2023.

Huidong Liang, Haitz Sáez de Ocáriz Borde, Baskaran Sripathmanathan, Michael Bronstein, and Xiaowen Dong. Towards quantifying long-range interactions in graph machine learning: a large graph dataset and a measurement. In *International Conference on Learning Representations*, volume 2026, pp. 34655–34688, 2026.

Tianyin Liao, Chunyu Hu, Yicheng Sui, Xingxuan Zhang, Peng Cui, Jianxin Li, and Ziwei Zhang. Tfmlinker: Universal link predictor by graph in-context learning with tabular foundation models. *arXiv preprint arXiv:2602.08592*, 2026.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *International conference on learning representations*, volume 2024, pp. 20188–20210, 2024.

Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021.

László Lovász and Balázs Szegedy. Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B*, 96(6):933–957, 2006.

Xin Lv, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Wei Zhang, Yichi Zhang, Hao Kong, and Suhui Wu. Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 5694–5703, 2020.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 2925–2933, 2020.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Mikhail Mironov and Liudmila Prokhorenkova. Revisiting graph homophily measures. In Guy Wolf and Smita Krishnaswamy (eds.), *Proceedings of the Third Learning on Graphs Conference*, volume 269 of *Proceedings of Machine Learning Research*, pp. 1:1–1:22. PMLR, 26–29 Nov 2025.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, 2014.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 385–394, 2017.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Tara Safavi and Danai Koutra. Codex: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8328–8350, 2020.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: a uniform language model fine-tuning framework for attributed graph embedding. In *NeurIPS*, 2023.

Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International conference on machine learning*, pp. 9448–9457. PMLR, 2020.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *TMLR*, 2021.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pp. 57–66, 2015.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Lianghao Xia and Chao Huang. AnyGraph: Graph foundation model in the wild. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 882–896. Association for Computational Linguistics, 2026.

Lianghao Xia, Ben Kao, and Chao Huang. Opengraph: Towards open graph foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2365–2379, 2024.

Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 564–573, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, et al. Limix: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Yucheng Zhang, Beatrice Bevilacqua, Mikhail Galkin, and Bruno Ribeiro. TRIX: A more expressive model for zero-shot domain transfer in knowledge graphs. In *LoG*, 2024.

Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael M. Bronstein, and Jian Tang. Fully-inductive node classification on arbitrary graphs. In *International Conference on Learning Representations*, 2025.

Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. A multi-task perspective for link prediction with new relation types and nodes. *arXiv preprint arXiv:2307.06046*, 2023.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS*, 2021.

A METHODOLOGY DETAILS

We here provide further details on the implementation of WANDER used for our experiments. Note that the statements in Section 6 refer to the more general framework described in Section 5.

Initialization. At the beginning of each forward pass, all hidden states are initialized independently of the input graph identities. All relation types, including a *start* relation used at the beginning of random walks, receive the same learnable initial embedding. Structure tokens are initialized with one of three learnable embeddings according to the node's structural relation to the link-prediction query head, when present: identical, neighboring, or neither. For node classification, where no query head is defined, all nodes receive the *neither* embedding. Feature tokens are initialized by applying a two-layer MLP to their scalar feature value. Finally, label tokens are initialized with one of three learnable embeddings, corresponding to an observed positive label, an observed negative label, or an unknown label.

Intra-node update. The intra-node module exchanges information between the channels of each node independently. For a node $v \in V$, let

$$\mathbf{V}_v^{(l)} := \big(\mathbf{v}^{(l)}(v,1), \ldots, \mathbf{v}^{(l)}(v, d_X + d_Y + 1)\big) \in \mathbb{R}^{(d_X+d_Y+1)\times d}$$

denote the sequence of its feature, label, and structure tokens. We update this sequence using a pre-normalized residual multi-head attention layer,

$$\mathbf{V}_v^{(l)\prime} = \mathbf{V}_v^{(l)} + \mathrm{MHA}_{\mathrm{RoPE}}\Big(\mathrm{RMSNorm}(\mathbf{V}_v^{(l)})\Big),$$

where the same module is applied independently to every $v \in V$. We use rotary positional embeddings (RoPE) (Su et al., 2024) to associate the individual channels with distinct positions and enable the model to distinguish two channels with distributionally similar values. Since fixed positional encodings would break equivariance with respect to permutations of feature and label channels, we uniformly randomize the ordering of feature and label channels in each forward pass and assign RoPE positions only after this permutation. The resulting model therefore remains equivariant in distribution with respect to feature and label permutations. We use RMSNorm (Zhang & Sennrich, 2019).

ICL-update. The attention-to-context module instead exchanges information between nodes while keeping channels separate. For each channel $i \in [d_X + d_Y + 1]$, let

$$\mathbf{V}_i^{(l)\prime} := \big(\mathbf{v}^{(l)\prime}(v,i)\big)_{v\in V} \in \mathbb{R}^{|V|\times d}.$$

We independently update each channel using a pre-normalized Transformer block,

$$\widetilde{\mathbf{V}}_i^{(l)} = \mathbf{V}_i^{(l)\prime} + \mathrm{MHA}_{|V_{\mathrm{context}}}\Big(\mathrm{RMSNorm}(\mathbf{V}_i^{(l)\prime})\Big),$$

followed by

$$\mathbf{V}_i^{(l)\prime\prime} = \widetilde{\mathbf{V}}_i^{(l)} + \mathrm{FFN}\Big(\mathrm{RMSNorm}(\widetilde{\mathbf{V}}_i^{(l)})\Big).$$

Here, the attention is masked such that every node may attend only to keys and values corresponding to nodes in V_{context}, while queries are computed for all nodes. We use separate attention modules for the three channel types (structural, feature, and label) while sharing the respective module across all channels of the same type. The feed-forward network (FFN) is implemented as a SwiGLU (Shazeer, 2020) block.

We next describe the random walk algorithm and then how we sample walks during training. Sampling during inference is explained in Appendix D.

Random walk algorithm. Let $v_0 \in V$ be a starting node given as input to the random walk algorithm. The algorithm constructs a walk $\eta = (v_0, r_1^{\alpha_1}, v_1, \ldots, r_S^{\alpha_S}, v_S)$ iteratively. At step $j \geq 1$, given the current node v_{j-1}, we define its set of neighbors as

$$N(v_{j-1}) := \{u \in V \mid \exists r \in R : (v_{j-1}, r, u) \in E \ \vee \ (u, r, v_{j-1}) \in E\}.$$

Note that we include incoming neighbors in this definition. To avoid immediate backtracking whenever possible, we define the set of admissible next nodes as

$$N_j := \begin{cases} N(v_0), & \text{if } j = 1, \\ N(v_{j-1}) \setminus \{v_{j-2}\}, & \text{if } j > 1 \text{ and } N(v_{j-1}) \setminus \{v_{j-2}\} \neq \emptyset, \\ N(v_{j-1}), & \text{if } j > 1 \text{ and } N(v_{j-1}) = \{v_{j-2}\}. \end{cases} \tag{1}$$

We first sample the next node uniformly, $v_j \sim \mathrm{Unif}(N_j)$. Given v_j, let

$$R(v_{j-1}, v_j) := \{(r, 1) \mid (v_{j-1}, r, v_j) \in E\} \cup \{(r, -1) \mid (v_j, r, v_{j-1}) \in E\}$$

denote the set of relation–direction pairs connecting v_{j-1} to v_j. We then sample $(r_j, \alpha_j) \sim \mathrm{Unif}\big(R(v_{j-1}, v_j)\big)$.

Thus, neighboring nodes are sampled independently of the number of relation types connecting them, and immediate backtracking occurs only if the previously visited node is the only available neighbor.

Choice of random walks. Let K_{base} denote the base walk budget, which may depend on the input graph (see Appendix D.2). For each structural update, we sample $3K_{\text{base}}$ walks, with their starting locations chosen according to the prediction task.

For a link-prediction query $q = (h_q, r_q, ?)$, we follow Kim et al. (2026) and divide the walks into three equally sized groups. The first K_{base} walks start at the query head h_q, providing structural context local to the query. For the next K_{base} walks, we first sample a relation $s \sim \mathrm{Unif}(R)$ and then an edge (v, s, u) uniformly among the observed edges of type s; this edge forms the first step of the walk, after which the usual random-walk transition rule is applied. Sampling the relation before the edge prevents frequent relation types from dominating the walk budget and ensures coverage of broad relations. The remaining K_{base} walks start from nodes sampled uniformly from V, providing broader, query-independent coverage of the graph.

For node prediction query V_q, we likewise sample $3K_{\text{base}}$ walks. We start $\lceil 3K_{\text{base}}/2 \rceil$ walks from nodes sampled uniformly from V_q and the remaining $\lfloor 3K_{\text{base}}/2 \rfloor$ from nodes sampled uniformly from V_{context}.

Recording protocol. Let $\eta = (v_0, r_1^{\alpha_1}, v_1, \ldots, r_S^{\alpha_S}, v_S)$ be a sampled walk, and let $V(\eta) \subseteq V$ and $R(\eta) \subseteq R$ denote the sets of nodes and relation types occurring in η, respectively. We define bijections

$$\mathrm{anon}_{\eta,V} : V(\eta) \to [|V(\eta)|], \qquad \mathrm{anon}_{\eta,R} : R(\eta) \to [|R(\eta)|],$$

which assign identifiers according to the order of first appearance in η. For each walk position, we first construct a shared walk embedding that is independent of the channel. At layer l, the shared embedding associated with node v, incoming relation $r \in R \cup \{\texttt{start}\}$, direction α and query q is

$$\begin{aligned} \mathrm{step}_\eta^{(l)}(v, r^\alpha, q) = {} & e^{(l)}_{\text{node}}(\mathrm{anon}_{\eta,V}(v)) + e^{(l)}_{\text{rel}}(\mathrm{anon}_{\eta,R}(r)) + e^{(l)}_{\text{dir}}(\alpha) \\ & + e^{(l)}_{\text{queryhead}}\big(\mathbf{1}_{h_q=v}\big) + e^{(l)}_{\text{queryrel}}\big(\mathbf{1}_{r_q=r}\big) + W^{(l)}_{\text{rel}}\, \mathbf{r}^{(l)}(r), \end{aligned} \tag{2}$$

where the query indicators are set to zero whenever the corresponding query component is not defined (i.e., for a label prediction query $V_q \subset V$). For the relation `start` we set direction $\alpha = 1$. The embedding tables $e_*^{(l)}$ and projection $W_{\text{rel}}^{(l)}$ are layer-specific.

For each channel $i \in [d_X + d_Y + 1]$, we combine this shared walk embedding with the current hidden state of the corresponding node–channel pair:

$$\mathbf{z}^{(l)}_{\eta,v,r,\alpha,i} = W^{(l)}_{\text{in},\tau(i)} \mathbf{v}^{(l)\prime\prime}(v, i) + g^{(l)}_{\tau(i)}\, \mathrm{step}^{(l)}_\eta(v, r^\alpha, q),$$

where $\tau(i) \in \{\text{feature}, \text{label}, \text{structure}\}$ denotes the channel type. Both $W^{(l)}_{\text{in},\tau(i)}$ and the learnable scalar gate $g^{(l)}_{\tau(i)}$ are layer- and channel-type-specific, but shared across all channels of the same type.

For each walk η and channel i, the position embeddings are stacked in walk order to obtain

$$Z^{(l)}_{\eta,i} = \left(\mathbf{z}^{(l)}_{\eta,v_0,\text{start},i}, \mathbf{z}^{(l)}_{\eta,v_1,r_1,\alpha_1,i}, \ldots, \mathbf{z}^{(l)}_{\eta,v_S,r_S,\alpha_S,i}\right).$$

Nodes and relation types may occur multiple times in this sequence whenever they are revisited by the walk.

Sequence processor. Each sequence is then processed by a bidirectional GRU (Cho et al., 2014) followed by a SwiGLU feed-forward network, both with pre-normalization and residual connections:

$$\widetilde{Z}^{(l)}_{\eta,i} = Z^{(l)}_{\eta,i} + W^{(l)}_{\mathrm{GRU}}\,\mathrm{BiGRU}^{(l)}\left(\mathrm{RMSNorm}(Z^{(l)}_{\eta,i})\right),$$
$$Z^{(l)\prime}_{\eta,i} = \widetilde{Z}^{(l)}_{\eta,i} + \mathrm{FFN}^{(l)}\left(\mathrm{RMSNorm}(\widetilde{Z}^{(l)}_{\eta,i})\right).$$

We use separate modules for the three channel types (structural, feature, and label) while sharing the respective module across all channels of the same type. Finally, linear output projections are applied independently at every sequence position. Let $\mathbf{z}' \in \mathbb{R}^d$ denote the processed representation at a position associated with node v and channel i. For every channel type, we produce a multi-head node–channel update proposal and confidence vector,

$$\Delta_{v,i} = \mathrm{reshape}_{d_H \times H}\left(W^{(l)}_{\Delta,\tau(i)}\mathbf{z}'\right), \qquad a_{v,i} = W^{(l)}_{a,\tau(i)}\mathbf{z}',$$

where $d_H \cdot H = d$, such that $\Delta_{v,i} \in \mathbb{R}^{d_H \times H}$ and $a_{v,i} \in \mathbb{R}^H$. The output projections are layer- and channel-type-specific and are shared across channels of the same type.

For the structure channel $i = d_X + d_Y + 1$, the same processed position additionally proposes an update to its incoming relation $r \in R \cup \{\mathrm{start}\}$:

$$\Delta_r = \mathrm{reshape}_{d_H \times H}\left(W^{(l)}_{\Delta,\mathrm{rel}}\mathbf{z}'\right), \qquad a_r = W^{(l)}_{a,\mathrm{rel}}\mathbf{z}'. \tag{3}$$

Thus, every position proposes an update to its corresponding node–channel state, while structure-channel positions additionally propose an update to the hidden state of their incoming relation type. Since nodes and relation types may occur repeatedly within and across sampled walks, each hidden state can receive multiple proposals.

Consensus protocol. The sequence processor yields for all pairs $(v, i) \in V \times [d_X + d_Y + 1]$ as well as relation types $r \in R \cup \{\texttt{start}\}$ a (possibly empty) set $\mathcal{P} = \{(\Delta^{(m)}, a^{(m)})\}_{m=1}^{M}$, with $\Delta^{(m)} = (\Delta^{(m,1)}, \ldots, \Delta^{(m,H)})$ and $a^{(m)} = (a^{(m,1)}, \ldots, a^{(m,H)})$, of update proposals and confidence scores. The consensus protocol then forms a multi-head softmax-normalized weighted average over each set of update proposals to decide on the final update:

$$c(\mathcal{P}) := \Big\|_{h=1}^{H} \sum_{m'=1}^{M} \frac{\exp(a^{(m',h)})}{\sum \exp(a^{(m,h)})}\Delta^{(m',h)},$$

where $\|$ denotes concatenation. For $\mathcal{P} = \emptyset$, we set $c(\mathcal{P}) = \mathbf{0}$. Finally, hidden states are updated in a residual way, i.e., $\mathbf{v}^{(l+1)}(v,i) = \mathbf{v}^{(l)\prime\prime}(v,i) + c(\mathcal{P}_{v,i})$ and $\mathbf{r}^{(l+1)}(r) = \mathbf{r}^{(l)}(r) + c(\mathcal{P}_r)$ for all $v \in V$, $i \in [d_X + d_Y + 1]$ and $r \in R \cup \{\texttt{start}\}$.

Readout. For node classification, we apply a shared two-layer MLP $\mathrm{out}_{\mathrm{node}} : \mathbb{R}^d \to \mathbb{R}$ independently to each final label token. Specifically, for a query node v and class $c \in [d_Y]$, we obtain the class logit

$$\ell_{v,c} = \mathrm{out}_{\mathrm{node}}\Big(\mathbf{v}^{(L)}(v, d_X + c)\Big).$$

A softmax over the d_Y class logits yields the corresponding class probabilities.

For link prediction with query $q = (h_q, r_q, ?)$, we score every candidate tail $t \in V$ using its final structure token together with the final state of the query relation. A shared two-layer MLP $\mathrm{out}_{\mathrm{link}} : \mathbb{R}^d \to \mathbb{R}$ produces

$$\ell_{h_q,r_q,t} = \mathrm{out}_{\mathrm{link}}\Big(\mathbf{v}^{(L)}(t, d_X + d_Y + 1) + \mathbf{r}^{(L)}(r_q)\Big).$$

We convert this logit into a link probability $p_\theta(h_q, r_q, t) = \sigma(\ell_{h_q,r_q,t})$.

Loss. For node-classification training steps, we minimize the cross-entropy loss over the predicted class logits of all query nodes V_q. The loss is

$$\mathcal{L}_{\mathrm{NC}} = -\frac{1}{|V_q|}\sum_{v \in V_q} \log \frac{\exp(\ell_{v,Y_v})}{\sum_{c=1}^{d_Y}\exp(\ell_{v,c})}.$$

For link-prediction steps, each positive triple (h, r, t) is paired with k negative tails $t_1^-, \ldots, t_k^-$. Following Kim et al. (2026), we use self-adversarial negative weighting, which assigns larger weight to negative candidates receiving larger model scores. Writing $\tau_{\mathrm{adv}} > 0$ for the adversarial temperature, we define

$$w_i^{\mathrm{adv}} = \frac{\exp\Big(\ell_{h,r,t_i^-}/\tau_{\mathrm{adv}}\Big)}{\sum_{j=1}^{k} \exp\Big(\ell_{h,r,t_j^-}/\tau_{\mathrm{adv}}\Big)}.$$

The resulting binary cross-entropy loss is

$$\mathcal{L}_{\mathrm{LP}} = -\log p_\theta(h, r, t) - \sum_{i=1}^{k} w_i^{\mathrm{adv}} \log\big(1 - p_\theta(h, r, t_i^-)\big)\,.$$

The losses are averaged over all link-prediction queries in the batch. Note that, due to the self-adversarial weighting used for link prediction, the practical objective no longer has $M_p^\star$ exactly as its population minimizer.

B Comparison to related (graph) foundation models

While the framework stated in Section 5 is stated more generally, in our implementation we instantiate the intra-node processor with self-attention and the ICL update with masked attention (see Appendix A). The resulting first two updates therefore follow the two-dimensional attention pattern common to recent tabular foundation models, alternating interactions within samples and across samples (Hollmann et al., 2025).

In the node-classification setting, this also yields an architectural connection to GraphPFN (Eremeev et al., 2026). Both architectures combine within-node feature processing and in-context information exchange with a graph-structural update. However, GraphPFN incorporates structure through adjacency-masked attention over neighbors, whereas Wander uses sampled random walks and a shared sequence processor as its structural interface. This distinction provides two notable advantages. First, random walks allow Wander to capture structural information beyond the expressive scope of standard message-passing architectures, as formalized in Theorem 3 and empirically investigated in Section D.11. Second, the amount of structural information available to the model can be adjusted at inference time by varying the number and length of sampled walks, without modifying the learned parameters.

In the setting of knowledge graph link prediction, Wander is instead closely related to the knowledge graph foundation model Flock (Kim et al., 2026). While Flock likewise uses random walks as its primary structural inductive bias, it does not include an explicit in-context update that allows query representations to attend to context nodes.

C Theoretical analysis

In this section, we provide proofs of the statements from Sections 4 and 6.

C.1 Bayes-Optimal Predictor

We start by proving Theorem 1.

Proof of Theorem 1. For an observed graph G_{obs}, let $Z_p(G_{\mathrm{obs}})$ and $Z_r(G_{\mathrm{obs}})$ denote its marginal likelihood (probability mass or density) under p and r, respectively. For $p_\lambda = (1-\lambda)p + \lambda r$, its marginal likelihood is

$$Z_{p_\lambda}(G_{\mathrm{obs}}) = (1-\lambda) Z_p(G_{\mathrm{obs}}) + \lambda Z_r(G_{\mathrm{obs}}).$$

Hence, whenever $Z_p(G_{\mathrm{obs}}) > 0$, the posterior under the mixture prior can be written as

$$p_\lambda(H \mid G_{\mathrm{obs}}) = \big(1 - \beta_{\lambda,r}(G_{\mathrm{obs}})\big) p(H \mid G_{\mathrm{obs}}) + \beta_{\lambda,r}(G_{\mathrm{obs}}) r(H \mid G_{\mathrm{obs}}),$$

where

$$\beta_{\lambda,r}(G_{\mathrm{obs}}) = \frac{\lambda Z_r(G_{\mathrm{obs}})}{(1-\lambda)Z_p(G_{\mathrm{obs}}) + \lambda Z_r(G_{\mathrm{obs}})}.$$

If $Z_r(G_{\mathrm{obs}}) = 0$, then $\beta_{\lambda,r}(G_{\mathrm{obs}}) = 0$, and by convention $M_r^\star = M_p^\star$ on such observations. Taking the posterior expectation of the query target therefore gives

$$M_{p_\lambda}^\star(G_{\mathrm{obs}}, q) = \big(1 - \beta_{\lambda,r}(G_{\mathrm{obs}})\big) M_p^\star(G_{\mathrm{obs}}, q) + \beta_{\lambda,r}(G_{\mathrm{obs}}) M_r^\star(G_{\mathrm{obs}}, q).$$

For brevity, write

$$\beta := \beta_{\lambda,r}(G_{\mathrm{obs}}), \qquad M_p := M_p^\star(G_{\mathrm{obs}}, q), \qquad M_r := M_r^\star(G_{\mathrm{obs}}, q),$$

and $T := T_q(G^\star)$. By convexity of ℓ_q in its prediction argument,

$$\ell_q\big((1-\beta)M_p + \beta M_r, T\big) \le (1-\beta)\ell_q(M_p, T) + \beta \ell_q(M_r, T).$$

Subtracting $\ell_q(M_p, T)$ and taking expectation under ν yields

$$\begin{aligned} R_\nu(M_{p_\lambda}^\star) - R_\nu(M_p^\star) &\le \mathbb{E}_\nu\big[\beta_{\lambda,r}(G_{\mathrm{obs}})\,\big(\ell_q(M_r^\star(G_{\mathrm{obs}}, q), T_q(G^\star)) - \ell_q(M_p^\star(G_{\mathrm{obs}}, q), T_q(G^\star))\big)\big] \\ &= \mathbb{E}_\nu\left[\beta_{\lambda,r}(G_{\mathrm{obs}})\,\mathbb{E}_\nu\left[\begin{matrix}\ell_q(M_r^\star(G_{\mathrm{obs}}, q), T_q(G^\star)) \\ \quad - \ell_q(M_p^\star(G_{\mathrm{obs}}, q), T_q(G^\star))\end{matrix}\,\middle|\, G_{\mathrm{obs}}, q\right]\right] \\ &= \mathbb{E}_\nu[\beta_{\lambda,r}(G_{\mathrm{obs}})\Delta_\nu(G_{\mathrm{obs}}, q)]\,. \end{aligned}$$

where the second equality uses that $\beta_{\lambda,r}(G_{\mathrm{obs}})$ depends only on the observation. The support assumption on p ensures that the above quantities are defined ν-almost surely, which proves the claim. □

The bound makes explicit that an added graph family matters only to the extent that it receives posterior weight on the target observation. In particular, adding r cannot increase the target risk if $\Delta_\nu(G_{\mathrm{obs}}, q) > 0$ only on observations for which $\beta_{\lambda,r}(G_{\mathrm{obs}}) = 0$. More generally, beneficial and harmful contributions are weighted by the posterior relevance of the added family.

Proof of Proposition 2. Let $\pi = (\pi_V, \pi_R, \pi_X, \pi_Y)$. Since p is a distribution over isomorphism classes, $p(\pi(H)) = p(H)$ for every H. Moreover,

$$G \preceq H \quad \Longleftrightarrow \quad \pi(G) \preceq \pi(H),$$

so $H \mapsto \pi(H)$ gives a bijection between the completions of G and those of $\pi(G)$. Hence

$$p(\pi(H) \mid \pi(G)) = p(H \mid G).$$

By definition of the query targets,

$$T_{\pi(q)}(\pi(H)) = \pi_{\mathrm{out}}(T_q(H))\,.$$

Therefore,

$$\begin{aligned} M_p^\star(\pi(G), \pi(q)) &= \mathbb{E}_{H\sim p(\cdot\mid\pi(G))}[T_{\pi(q)}(H)] \\ &= \mathbb{E}_{H\sim p(\cdot\mid G)}[T_{\pi(q)}(\pi(H))] \\ &= \pi_{\mathrm{out}}\big(\mathbb{E}_{H\sim p(\cdot\mid G)}[T_q(H)]\big) \\ &= \pi_{\mathrm{out}}\big(M_p^\star(G, q)\big)\,, \end{aligned}$$

which proves the claim. □

C.2 Universality

Before we can prove Theorem 3, we need to formalize the notion of *admissible priors.*

Admissible graph prior. Let p be a probability distribution over the isomorphism classes of $\mathbb{G}^{\star}_{n,m,b}$. For every isomorphism class G, we separate its discrete attributes from its continuous attributes. Denote by κ_G its *discrete isomorphism class*, which fixes the graph structure, relation types, feature dimension, and all discrete-valued attributes, and by

$$z_G \in \mathcal{Z}_{\kappa_G} \subseteq \mathbb{R}^{d_{\kappa_G}}$$

its continuous attributes. We can then write the prior as

$$p(G) = p(\kappa_G)\, p_{\kappa_G}(z_G), \qquad p_{\kappa_G}(z_G) := p(z_G \mid \kappa_G).$$

We call p an *admissible graph prior* if, for every discrete isomorphism class κ with $p(\kappa) > 0$, the domain $\mathcal{Z}_\kappa$ is a compact product domain and p_κ admits a continuous and strictly positive density on $\mathcal{Z}_\kappa$. Moreover, for each fixed discrete observed graph-query class $\bar{\kappa}$, all compatible positive-mass completions share the same observed-attribute domain: after partitioning $z = (x, u)$ into observed and unobserved attributes,

$$\mathcal{Z}_\kappa = \mathcal{X}_{\bar{\kappa}} \times \mathcal{U}_{\bar{\kappa},\kappa}.$$

Throughout, we fix an observation mechanism that reveals a subset of attributes, with selection probabilities depending only on the discrete completion, not on the continuous attributes.

Continuous Bayes-optimal predictors. We now show that the Bayes-optimal predictor introduced in Section 3 is well-behaved under admissible priors. This is required later to show that it can be approximated.

Lemma 5. *Let p be an admissible graph prior. Then, within every fixed discrete isomorphism class of observed graph-query pairs, the Bayes-optimal predictor*

$$M^{\star}_p(G_{\mathrm{obs}}, q) = \mathbb{E}_{G^{\star} \sim p(\cdot \mid G_{\mathrm{obs}})}\left[T_q(G^{\star})\right]$$

is continuous in the continuous attributes observed in G_{obs} for link prediction and node classification.

Proof. Fix a discrete observed graph-query class $\bar{\kappa}$ and write $x \in \mathcal{X}_{\bar{\kappa}}$ for its observed continuous attributes. Let $\mathcal{K}$ be the finite, nonempty set of positive-mass discrete completions that can produce this observation. For each $\kappa \in \mathcal{K}$, define

$$p_{\kappa,X}(x) := \int_{\mathcal{U}_{\bar{\kappa},\kappa}} p_\kappa(x, u)\, du.$$

Continuity on the compact product domain implies that $p_{\kappa,X}$ is continuous on the common domain $\mathcal{X}_{\bar{\kappa}}$. Strict positivity of p_κ implies $p_{\kappa,X}(x) > 0$ there.

Let $a_\kappa > 0$ denote $p(\kappa)$ multiplied by the probability of the fixed discrete observation given κ. For link prediction and node classification, each target coordinate is determined by the discrete completion; denote its value by $c_{\kappa,j} \in \{0, 1\}$. Bayes' rule gives

$$M^{\star}_{p,j}(x, q) = \frac{\sum_{\kappa \in \mathcal{K}} a_\kappa c_{\kappa,j} p_{\kappa,X}(x)}{\sum_{\kappa \in \mathcal{K}} a_\kappa p_{\kappa,X}(x)}.$$

The numerator and denominator are finite sums of continuous functions on $\mathcal{X}_{\bar{\kappa}}$, and the denominator is strictly positive there. Hence every output coordinate is continuous. □

Proof of Theorem 3. We are now able to prove Theorem 3 from Section 6, here stated in a more detailed form. Note that we prove universality of the WANDER *framework* under the assumptions below. This does not necessarily correspond to the implementation details given in Appendix A.

Theorem 3. *Fix $n, m, b \in \mathbb{N}$. Assume a* WANDER *configuration with at least two layers, a walk length S of at least n, and which fulfills the following assumptions on the random walk sampler and model capacity:*

(A1) *each $v \in V$ and each $r \in R$ has positive probability of being chosen as the starting node or the first traversed relation type of a walk, respectively, every non-backtracking walk of length at most S has positive probability of occurring as a prefix of a sampled walk and all walks are sampled independently and have probabilities bounded away from zero by constants depending only on n and m;*

(A2) *after the intra-node update, every node token can retain an injective encoding of the complete collection of tokens of that node;*

(A3) *at every sequence position, the sequence processor can retain an injective encoding of the complete recorded walk together with the distinguished position;*

(A4) *the consensus protocol can retain an injective encoding of the multiset of update proposals associated with a hidden state, while preserving its previous state; and*

(A5) *the output head is a universal approximator of continuous functions on compact domains.*

Then WANDER *is a universal graph prior approximator for graph priors over* $\mathbb{G}^{\star}_{n,m,b}$.

Note that the context-attention module may be chosen as the identity and plays no role in this argument.

We start with a simple observation, which will help us in proving that with high probability, random walks break all symmetries.

Lemma 6 (Random counts break finite symmetries)**.** *Let U be a finite set and let $X_1, \dots, X_K$ be i.i.d. samples from a distribution p satisfying $p(u) > 0$ for every $u \in U$. For $u \in U$, define*

$$C_u^{(K)} := \sum_{j=1}^{K} \mathbf{1}_{\{X_j = u\}}.$$

Then

$$\Pr\Big[C_u^{(K)} \neq C_{u'}^{(K)} \text{ for all distinct } u, u' \in U\Big] \longrightarrow 1 \qquad \text{as } K \to \infty.$$

Proof. Fix distinct $u, u' \in U$.

First suppose $p(u) \neq p(u')$. By the strong law of large numbers,

$$\frac{C_u^{(K)} - C_{u'}^{(K)}}{K} \longrightarrow p(u) - p(u') \neq 0 \qquad \text{almost surely.}$$

Hence $\Pr[C_u^{(K)} = C_{u'}^{(K)}] \to 0$. Now suppose $p(u) = p(u') = \beta > 0$. Let $M_K := C_u^{(K)} + C_{u'}^{(K)}$. Then $M_K \sim \mathrm{Bin}(K, 2\beta)$ and, conditional on $M_K = \ell$, $C_u^{(K)} \sim \mathrm{Bin}(\ell, 1/2)$. Consequently,

$$\Pr\Big[C_u^{(K)} = C_{u'}^{(K)} \mid M_K = \ell\Big] \leq \binom{\ell}{\lfloor \ell/2 \rfloor} 2^{-\ell} = O(\ell^{-1/2}).$$

Since $M_K \to \infty$ in probability, it follows that $\Pr[C_u^{(K)} = C_{u'}^{(K)}] \to 0$. Since U is finite, a union bound over all unordered pairs $\{u, u'\} \subseteq U$ gives the claim. □

Now we first show that with sufficiently many walks, all nodes and relation types are uniquely identifiable after a single layer.

Lemma 7 (First-layer node and relation identifiers)**.** *Under assumptions* (A1)–(A4)*, for every $n, m \in \mathbb{N}$ and every $\delta > 0$, there exists a finite number $K_1 = K_1(n, m, \delta)$ of first-layer walks such that, with probability at least $1 - \delta$, after the first structural update:*

1. *every node $v \in V$ contains a unique identifier $\iota_V(v)$,*
2. *every relation type $r \in R$ contains a unique identifier $\iota_R(r)$, and*
3. *the original observed node attributes and labels are preserved.*

Proof. For each node $v \in V$, let

$$C_v := \#\{j \in [K_1] : \eta_j \text{ starts at } v\},$$

and for every relation type $r \in R$, let

$$D_r := \#\{j \in [K_1] : r \text{ is the first relation traversed by } \eta_j\}.$$

By assumption (A1), the starting node of a sampled walk follows a categorical distribution assigning strictly positive probability to every $v \in V$. Similarly, the first traversed relation follows a categorical distribution assigning strictly positive probability to every $r \in R$. Lemma 6 therefore implies

$$\Pr[C_v \neq C_{v'} \text{ for all } v \neq v'] \longrightarrow 1$$

and

$$\Pr[D_r \neq D_{r'} \text{ for all } r \neq r'] \longrightarrow 1$$

as $K_1 \to \infty$.

The beginning of a walk is explicitly distinguished by the start-relation record, while the first traversed relation is identifiable from its position in the recorded sequence. By assumption (A3), the corresponding update proposals may therefore encode whether a node occurs as the starting node and whether a relation occurs as the first traversed relation. By assumption (A4), the consensus protocol may retain the multiplicity of these proposals. Consequently, the updated node state may retain C_v, and the updated relation state may retain D_r.

We may therefore take

$$\iota_V(v) := C_v, \qquad \iota_R(r) := D_r.$$

With probability tending to one, these values are injective on V and R, respectively.

Finally, assumption (A2) ensures that the complete local node state, including all observed features and labels, can be retained before the structural update, while assumption (A4) allows the structural update to preserve this information. Hence the random identifiers do not overwrite the original node information.

Since $|V| \leq n$ and $|R| \leq m$, the required number of walks can be chosen uniformly over all considered graphs in $\mathcal{G}_{n,m,b}$. □

Next we will show that when choosing the walk length to be at least the diameter of the graph, for any node-edge pair there always exists a walk that traverses both.

Lemma 8 (Short non-backtracking edge witnesses)**.** *Let $G = (V, E, R, X, Y)$ have connected underlying undirected graph with diameter D. For every target node $t \in V$ and every edge $e = (u, r, v) \in E$, there exists a non-backtracking walk starting at t, of length at most*

$$D + 1 \leq |V| \leq n,$$

that traverses e in one of its two directions.

Proof. Fix $t \in V$ and $e = (u, r, v) \in E$. Let

$$P = (t = v_0, v_1, \ldots, v_\ell = u)$$

be a shortest path in the underlying undirected graph from t to u. Then $\ell \leq D$. Since P is a shortest path, it contains no immediate backtracking.

If either $\ell = 0$ or $v_{\ell-1} \neq v$, append the traversal from u to v along e. The resulting walk is non-backtracking and has length

$$\ell + 1 \leq D + 1.$$

If instead $v_{\ell-1} = v$, then the final step of P already traverses the edge e, namely from v to u. Since traversal direction is recorded, this already identifies the original directed edge (u, r, v). In this case P itself is the required witness and has length at most D.

Thus in either case there exists a non-backtracking witness walk of length at most $D + 1$. Since a connected graph on at most n nodes satisfies $D \leq |V| - 1$, the witness has length at most n. □

Lemma 9 (Simultaneous edge witnessing)**.** *Assume that the walk length S is at least n. For every $n, m \in \mathbb{N}$ and every $\delta > 0$, there exists a finite number $K_2 = K_2(n, m, \delta)$ of second-layer walks such that, with probability at least $1 - \delta$, for every node $t \in V$ and every edge $e \in E$, at least one sampled walk starts at t and contains a traversal of e.*

Proof. By Lemma 8, for every pair $(t, e) \in V \times E$ there exists an admissible non-backtracking witness walk of length at most n.

By assumption (A1) on the sampler, a walk that contains such a prefix has probability of at least $\beta_{n,m} > 0$.

Therefore,

$$\Pr[(t, e) \text{ is not witnessed in } K_2 \text{ walks}] \leq (1 - \beta_{n,m})^{K_2}.$$

Since $|E| \leq |V|^2|R| \leq n^2 m$, there are at most $n^3 m$ target-edge pairs. A union bound yields

$$\Pr[\exists (t, e) \in V \times E \text{ that is not witnessed}] \leq n^3 m (1 - \beta_{n,m})^{K_2}.$$

The right-hand side converges to zero as $K_2 \to \infty$, proving the claim. □

Lemma 10 (Two-layer graph reconstruction)**.** *Suppose the events in Lemmas 7 and 9 both occur. Then, after the second structural update, the hidden state associated with every node $t \in V$ can retain an encoding from which the complete observed graph G and, in the case of link prediction, the query q can be reconstructed up to isomorphism.*

Proof. By Lemma 7, after the first layer every node and relation type carries a persistent identifier

$$\iota_V : V \to \mathbb{N}, \qquad \iota_R : R \to \mathbb{N},$$

which is injective on V and R, respectively. The first-layer node states also retain all observed node attributes and labels.

Consider now the second structural layer and fix a target node t. By Lemma 9, for every edge $e = (u, r, v) \in E$ there exists a sampled walk η_e that starts at t and traverses e.

Because the first-layer identifiers are part of the hidden states used by the recording protocol, the recording of η_e contains enough information to identify the traversal of e as

$$\big(\iota_V(u), \iota_R(r), \iota_V(v)\big),$$

together with its orientation. In addition, whenever a node occurs in a walk, its recorded hidden state contains its observed local attributes.

By assumption (A3), the update proposal produced at the starting position t may retain an injective encoding of the complete recorded walk η_e. Thus, the proposal associated with t contains the information that the edge

$$\big(\iota_V(u), \iota_R(r), \iota_V(v)\big)$$

exists.

By assumption (A4), the consensus update at t can retain injectively the multiset of all such proposals. Since every edge is witnessed by at least one walk starting at t, the resulting state can therefore recover

$$\left\{ \big(\iota_V(u), \iota_R(r), \iota_V(v)\big) : (u, r, v) \in E \right\}.$$

Because ι_V and ι_R are injective, this determines the complete edge-typed graph up to a renaming of nodes and relation types.

Since the underlying graph is connected, every node is incident to an edge. Hence the collection of edge-witnessing walks also contains the first-layer state of every node and therefore determines all observed node features and labels.

Finally, the recording protocol contains the link query indicators. It follows that the second-layer state at t determines G, and in the case of link prediction q, up to isomorphism. □

We are now ready to prove the main result.

Proof of Theorem 3. Fix $\epsilon, \delta > 0$. By Lemmas 7 and 9, the probabilities of successful symmetry breaking in the first layer and complete edge witnessing in the second layer both converge to one as the number of sampled walks grows. Hence, there exists a common finite sampling budget $K \in \mathbb{N}$ such that, when K walks are sampled in each layer, the event in Lemma 7 and the event in Lemma 9 each

occur with probability at least $1 - \delta/2$. By a union bound, both events therefore occur simultaneously with probability at least $1 - \delta$.

Condition on such a successful realization of the sampled walks. By Lemma 10, the second-layer hidden state used by the output head determines the observed graph-query pair (G, q) up to isomorphism.

Fix now a discrete isomorphism class $\bar{\kappa}$ of connected observed graph-query pairs. Let $\mathcal{K}(\bar{\kappa})$ denote the set of discrete isomorphism classes of fully specified graphs that are compatible with $\bar{\kappa}$ and have positive prior probability. Since n, m, and b are bounded, $\mathcal{K}(\bar{\kappa})$ is finite.

For every $\kappa \in \mathcal{K}(\bar{\kappa})$, let $\mathcal{X}_{\bar{\kappa},\kappa}$ denote the projection of $\mathcal{Z}_\kappa$ onto the continuous coordinates observed in $\bar{\kappa}$, and define

$$\mathcal{X}_{\bar{\kappa}} := \bigcup_{\kappa \in \mathcal{K}(\bar{\kappa})} \mathcal{X}_{\bar{\kappa},\kappa}.$$

Each $\mathcal{X}_{\bar{\kappa},\kappa}$ is compact as the projection of the compact set $\mathcal{Z}_\kappa$. Since the union is finite, $\mathcal{X}_{\bar{\kappa}}$ is compact. Let $x \in \mathcal{X}_{\bar{\kappa}}$ denote the continuous observed attributes.

For a fixed successful realization ω of the discrete random choices made by the model, let

$$\Phi_{\bar{\kappa},\omega} : \mathcal{X}_{\bar{\kappa}} \longrightarrow \mathbb{R}^d$$

denote the resulting representation supplied to the output head. The model modules are continuous, so $\Phi_{\bar{\kappa},\omega}$ is continuous. Moreover, by Lemma 10, the representation determines the corresponding graph-query pair up to isomorphism. After fixing a canonical representative of $\bar{\kappa}$, we may therefore regard $\Phi_{\bar{\kappa},\omega}$ as injective.

A continuous injection from a compact space into $\mathbb{R}^d$ is a homeomorphism onto its image. Hence

$$\Phi_{\bar{\kappa},\omega}^{-1}$$

is continuous on $\Phi_{\bar{\kappa},\omega}(\mathcal{X}_{\bar{\kappa}})$.

Let $M^\star_{p,\bar{\kappa}}$ denote the restriction of the Bayes-optimal predictor to the observed discrete class $\bar{\kappa}$, viewed as a function of its continuous observed attributes x. By Lemma 5, $M^\star_{p,\bar{\kappa}}$ is continuous. Consequently,

$$h \longmapsto M^\star_{p,\bar{\kappa}}\big(\Phi^{-1}_{\bar{\kappa},\omega}(h)\big)$$

is continuous on the compact representation domain $\Phi_{\bar{\kappa},\omega}(\mathcal{X}_{\bar{\kappa}})$.

Because n, m, and b are bounded, there are only finitely many discrete isomorphism classes of connected observed graph-query pairs. For a fixed number and length of sampled walks, there are also only finitely many discrete walk realizations. Hence the successful representation domains form a finite union of compact sets. Whenever two such domains intersect, Lemma 10 implies that the shared representation determines the same graph-query pair up to isomorphism, and therefore the same Bayes-optimal output. The continuous functions defined above thus agree on intersections and together define a continuous function on this finite union of compact representation domains.

By assumption (A5), the output head can approximate this function uniformly to accuracy ϵ. Thus, whenever the two successful sampling events occur,

$$\big\| M_{\theta,K}(G, q) - M^\star_p(G, q) \big\| < \epsilon.$$

Since the numbers of walks were chosen uniformly over $\mathcal{G}_{n,m,b}$ and the two successful events occur jointly with probability at least $1 - \delta$, we obtain

$$\Pr\big[\big\| M_{\theta,K}(G, q) - M^\star_p(G, q) \big\| < \epsilon\big] \geq 1 - \delta$$

uniformly over all connected $G \in \mathcal{G}_{n,m,b}$ and all valid queries q. Thus, for every $\epsilon, \delta > 0$, there exist network parameters θ and a finite sampling budget K satisfying the definition of a universal graph prior approximator. □

C.3 Permutation equivariance

Proof of Proposition 4. Let $\pi = (\pi_V, \pi_R, \pi_X, \pi_Y)$ and let $(G', q') = (\pi(G), \pi(q))$. We show inductively over the layers that the joint distribution of the hidden states is equivariant under π:

node states are permuted according to π_V, their feature and label channels according to π_X and π_Y, respectively, while relation states are permuted according to π_R.

The claim holds at initialization by construction: feature and label values are carried along with their corresponding channels, structural states depend only on the node's role relative to the query, and relation states are independent of relation identities.

Assume now that the claim holds at the beginning of layer l. By assumption, the intra-node processor μ_θ is equivariant in distribution under permutations of feature and label channels. Since the same processor is applied independently to every node, the intra-node update also commutes with π_V.

The in-context update acts independently on each channel and aggregates over the corresponding context nodes. Since

$$V_{\text{context}}(G', q') = \pi_V(V_{\text{context}}(G, q)),$$

permuting nodes and channels therefore only permutes the corresponding queries, keys, and values, and the update remains equivariant.

For the structural update, permutation invariance of A implies that the walks sampled on (G', q') have the same distribution as the images under π of the walks sampled on (G, q). The recording protocol consequently produces correspondingly permuted walk recordings. Since the same sequence processor is applied to every recording, corresponding walk positions produce corresponding update proposals. Finally, the consensus protocol is permutation invariant in the collection of proposals associated with each state, so the resulting node and relation states are again equivariant in distribution. This completes the induction.

After the final layer, the shared node head maps permuted label channels to the correspondingly permuted classification outputs. Likewise, the shared link head maps permuted candidate-node and relation states to the correspondingly permuted link-prediction outputs. Hence

$$M_\theta(\pi(G), \pi(q)) \overset{d}{=} \pi_{\text{out}}(M_\theta(G, q)),$$

which proves the claim. □

C.4 Computational complexity

Let $N = |V|$, $C = d_X + d_Y + 1$ be the number of channels, $N_{\text{ctx}} = |V_{\text{context}}|$, K the number of sampled walks per layer, S their length, and d the hidden dimension. For a single WANDER layer, intra-node self-attention costs

$$\mathcal{O}\big(N(C^2 d + C d^2)\big),$$

since attention is applied independently across the C tokens of each node. The ICL update uses the N node tokens as queries and the N_{ctx} context tokens as keys and values, independently for each channel, and therefore costs

$$\mathcal{O}\big(CNN_{\text{ctx}} d + C(N + N_{\text{ctx}}) d^2\big).$$

Finally, the stochastic structural update processes KC sequences of length $S + 1$. With the bidirectional GRU and feed-forward network used in our implementation, this requires

$$\mathcal{O}\big(KCSd^2\big)$$

operations. Sampling the walks and aggregating their proposals add only $\mathcal{O}(KS)$ and $\mathcal{O}(KCSd)$, respectively, and are therefore dominated by sequence processing.

Consequently, the overall time complexity of one layer is

$$\mathcal{O}\big(NC^2 d + CNN_{\text{ctx}} d + \big(NC + CN_{\text{ctx}} + KCS\big) d^2\big)$$

which simplifies to

$$\mathcal{O}\big(NC^2 + CNN_{\text{ctx}} + KCS\big)$$

for fixed hidden dimension d. In particular, for a fixed architecture and hidden dimension, the structural update scales linearly in both the number and length of the sampled walks. This makes its computation independent of the total number of edges except for adjacency-list construction and permits the accuracy–cost trade-offs obtained by varying the walk budget at inference time.

D EXPERIMENTAL DETAILS

D.1 GRAPH PRIOR

The training objective stated in Section 4 assumes access to samples from a graph prior p. Ideally, this prior would capture the distribution of real-world graphs directly. In practice, however, available graph datasets form a comparatively small and heterogeneous collection. We therefore follow the prior-data fitted network paradigm introduced by Müller et al. (2022) and construct most of our pretraining distribution synthetically.

A central ingredient of TABPFN (Hollmann et al., 2023) is a prior over *structural causal models* (SCMs), which generate variables by sampling a causal dependency graph together with functions mapping parent variables and noise to their descendants. Selecting some generated variables as features and another as the target yields a broad synthetic distribution over supervised learning problems.

Synthetic attributed graphs. Recent GFMs extend this idea to attributed graphs, but make different assumptions about how graph structure and node attributes arise. NODEPFN (Choi et al., 2026) follows an *attribute-first* construction: features and labels are first sampled from an MLP-based SCM, after which the graph is generated using either a label-dependent stochastic block model (Holland et al., 1983) or an Erdős–Rényi (Erdős & Rényi, 1959) model. GRAPHPFN (Eremeev et al., 2026) instead follows a *structure-first* construction: the topology is sampled from multi-level degree-corrected stochastic block models and preferential-attachment graphs, and features and targets are subsequently generated by a graph-aware SCM.

These constructions capture complementary dependencies: attributes may influence which relations form, while graph structure may itself influence node attributes. We therefore include both generation directions in our prior. We additionally broaden the structural distribution with Watts–Strogatz (Watts & Strogatz, 1998), random geometric (Gilbert, 1961), and kernel-based graph models (Lovász & Szegedy, 2006) to cover structural regimes beyond block-community graphs, including highly clustered and geometry-driven networks. Concretely, each synthetic graph is attribute-first with probability 0.4 and structure-first with probability 0.6. Structure-first topologies are a multi-level degree-corrected stochastic block model with preferential attachment (0.45), a random geometric graph in latent dimension 2–10 (0.30), or a Watts–Strogatz graph with rewiring probability in $[0.01, 0.3]$ (0.25). Attribute-first topologies are a label-ordered Watts–Strogatz graph (0.45), a label stochastic block model (0.35), or an RBF kernel graph (0.20). The label stochastic block model and the RBF kernel use a homophily ratio that is homophilous (centered at 0.8) with probability 0.7 and heterophilous (centered at 0.1) otherwise. Graph size is mixed log-uniform, half in $[200, 3000]$ nodes and half in $[3000, 16000]$, and the average degree is log-uniform on $[2.5, 20]$ with probability 0.9. Homogeneous link-prediction episodes reuse these undirected graphs with labels removed; node features are dropped with probability 0.5, and a fraction of edges drawn from $[0.7, 0.9]$ is kept as context.

The resulting attributed graphs are used for both node-classification and homogeneous link-prediction pretraining.

Comparison with other synthetic priors. To characterize the structural regimes covered by our prior, we sample 1,000 graphs from each of the WANDER, NODEPFN, and GRAPHPFN priors and compare their statistics with those of the real-world evaluation graphs. Figure 3 shows that WANDER covers a wider range of clustering coefficients than the other synthetic priors, including graphs with substantial clustering at low average degree. It also spans both homophilous and heterophilous regimes, although its samples are concentrated toward homophily. The distributions in Figure 4 provide complementary comparisons of graph size, average degree, clustering, homophily, and number of classes. To test the effect of these differences on the transfer to real world graphs we replace the synthetic prior during pretraining in Appendix D.13.

Multi-relational graphs. Extending such synthetic mechanisms to multi-relational graphs is less straightforward, as relation types in knowledge graphs follow different and often domain-specific generative processes. We therefore augment the synthetic attributed-graph prior with three real-world knowledge graphs: FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), and

CoDEx Medium (Safavi & Koutra, 2020). These graphs provide multi-relational link-prediction episodes during pretraining, exposing WANDER jointly to synthetic attributed graphs and real-world multi-relational structure. Statistics about these three real-world knowledge graphs are provided in Table 11.

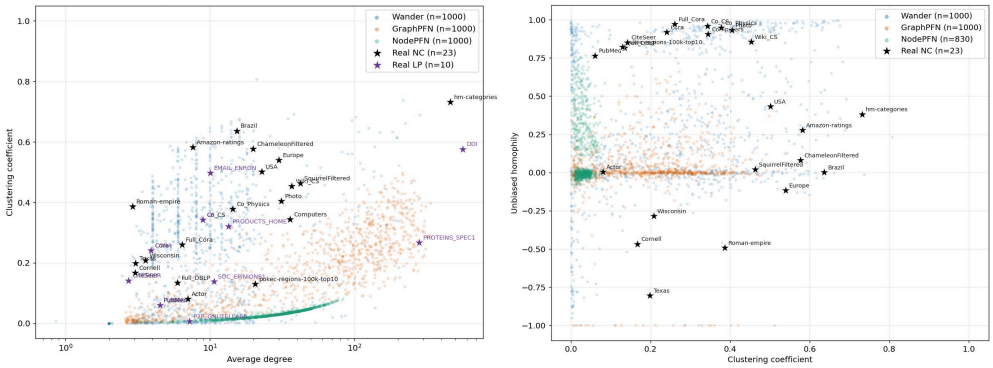


Figure 3: Structural coverage of the synthetic graph priors compared with the real-world evaluation graphs.

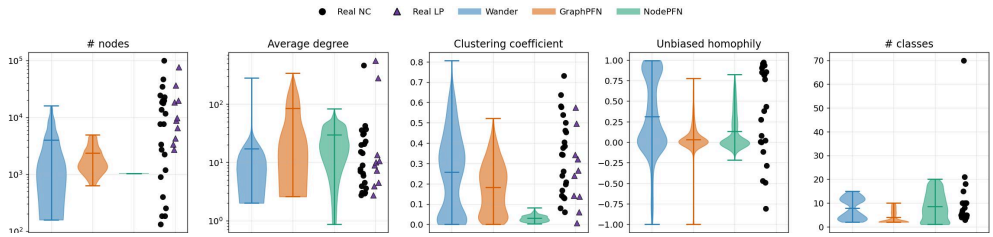


Figure 4: Distributions of graph statistics for 1,000 graphs sampled from each synthetic prior, alongside the real-world evaluation graphs.

D.2 IMPLEMENTATION AND PRETRAINING DETAILS

The model and optimization hyperparameters are summarized in Table 8. We train WANDER for 350 epochs using a four-stage curriculum that progressively increases the complexity of the pretraining tasks.

During the first 200 epochs, we disable the stochastic structural update in every layer and retain only the intra-node and ICL updates. Pretraining during this stage uses exclusively synthetic graphs, with 80% of training steps devoted to node classification and 20% to homogeneous link prediction. This initial stage is intended to encourage the model to learn to exploit node attributes and in-context supervision before introducing the random-walk-based structural signal, thereby reducing the incentive to rely primarily on graph structure or label homophily. We additionally increase the feature dimensionality over the course of this stage: during the first 100 epochs, the maximum feature dimension is 16, while during epochs 101–200 it is increased to 32. Whenever a sampled synthetic task contains more features than the respective limit, we reduce the feature dimension using PCA.

For the remaining 150 epochs, we activate the stochastic structural update and train the complete WANDER architecture. At this stage, 50% of training steps use knowledge-graph link prediction, 40% use synthetic node classification, and 10% use synthetic homogeneous link prediction. The maximum feature dimension is increased to 64. During the final 50 epochs, we additionally increase the maximum size of synthetic graphs from $8{,}000$ to $16{,}000$ nodes.

Table 8: Pretraining hyperparameters across the four pretraining stages of WANDER.

	Phase 1	Phase 2	Phase 3	Phase 4
Epoch	1–100	101–200	201–300	301–350
Architecture				
Hidden dimension d	128 *(all phases)*			
Layers L	6 *(all phases)*			
Heads (intra-node)	4 *(all phases)*			
Heads (ICL)	4 *(all phases)*			
Heads (structural update)	4 *(all phases)*			
Structural update	off	off	on	on
Base walk count K_{base} ($n < 1$k)	—	—	32	32
Base walk count K_{base} (1k–8k)	—	—	64	64
Base walk count K_{base} ($n \geq 8$k)	—	—	128	128
Walk length S ($n < 1$k)	—	—	32	32
Walk length S (1k–8k)	—	—	64	64
Walk length S ($n \geq 8$k)	—	—	128	128
Data				
Synthetic NC	80%	80%	40%	40%
Synthetic LP	20%	20%	10%	10%
Real KG	0%	0%	50%	50%
Max. size synthetic	8,000	8,000	8,000	16,000
Max. features synthetic	16	32	64	64
Optimizer & training				
Optimizer	AdamW *(all phases)*			
Learning rate	5×10^{-5} *(all phases)*			
Weight decay	0.01 *(all phases)*			
Batches per epoch	2,000 *(all phases)*			
Graphs per batch	2 *(all phases)*			
LP batch size (per graph)	4 *(all phases)*			
NC batch size (per graph)	all test	all test	all visited	all visited
Negatives (link prediction)	512 *(all phases)*			

Each optimization step contains two graphs drawn from the same task family: either two synthetic node-classification tasks, two synthetic homogeneous link-prediction tasks, or two knowledge graphs. In the latter case, the two graphs may correspond to the same underlying knowledge graph.

The pretrained model has approximately 13M parameters.

WANDER is stochastic because of random-walk sampling and the random permutations of feature and label channels applied during inference. We refer to each independently randomized forward pass as an *inference sample*. Unless stated otherwise, we average the logits from 16 inference samples before applying the task-specific output transformation. For knowledge-graph link prediction, we follow Kim et al. (2026) for comparability and use the dataset-specific numbers of inference samples reported in Table 13.

Following prior work (Kim et al., 2026), we add in each knowledge graph for each triple (h, r, t) the corresponding inverse (t, r^{-1}, h), where r^{-1} is a new relation symbol.

D.3 KNOWLEDGE GRAPH LINK PREDICTION

To evaluate zero-shot transfer for knowledge graph link prediction, we consider the entity-prediction setting, where the task is to rank candidate tail entities given a query head entity and relation type.

Datasets. We follow prior work on knowledge graph foundation models (Galkin et al., 2024; Zhang et al., 2024; Kim et al., 2026) and evaluate on the same benchmark suite, comprising 54 knowledge graphs grouped into three transfer settings:

- **Inductive e, r.** This setting evaluates transfer to knowledge graphs containing both unseen entities and unseen relation types. It comprises the 13 datasets introduced by InGram (Lee et al., 2023): FB-25, FB-50, FB-75, FB-100, WK-25, WK-50, WK-75, WK-100, NL-0, NL-25, NL-50, NL-75, and NL-100; as well as 10 datasets from MTDEA (Zhou et al., 2023): MT1 tax, MT1 health, MT2 org, MT2 sci, MT3 art, MT3 infra, MT4 sci, MT4 health, Metafam, and FBNELL. Dataset statistics are provided in Table 9.

- **Inductive e.** This setting evaluates transfer to previously unseen entities while the relation vocabulary is shared with the training graph. It includes the 12 GraIL benchmarks (Teru et al., 2020), consisting of WN-v1 through WN-v4, FB-v1 through FB-v4, and NL-v1 through NL-v4; four INDIGO benchmarks (Liu et al., 2021): HM 1k, HM 3k, HM 5k, and HM Indigo; and the two NodePiece benchmarks (Galkin et al., 2022), ILPC Small and ILPC Large. Dataset statistics are provided in Table 10.
- **Transductive.** In the transductive setting, both entities and relation types are observed during training, and evaluation concerns held-out links over the same entity and relation vocabularies. This group comprises CoDEx Small and CoDEx Large (Safavi & Koutra, 2020), NELL-995 (Xiong et al., 2017), YAGO3-10 (Mahdisoltani et al., 2014), WDsinger, NELL23k, FB15k-237(10), FB15k-237(20), and FB15k-237(50) (Lv et al., 2020), AristoV4 (Chen et al., 2021), DBpedia100k (Ding et al., 2018), ConceptNet100k (Malaviya et al., 2020), and Hetionet (Himmelstein et al., 2017). Dataset statistics are provided in Table 11.

Baselines and evaluation protocol. We compare WANDER against the knowledge graph foundation models ULTRA (Galkin et al., 2024), TRIX (Zhang et al., 2024), and FLOCK (Kim et al., 2026). All three baselines are designed for zero-shot transfer across knowledge graphs and are evaluated on the benchmark suite described above.

We follow the filtered ranking protocol used in the corresponding prior work. For each query, candidate entities corresponding to other known true facts are removed from the ranking before computing the evaluation metrics. For evaluating head prediction we convert the query into a tail prediction, by using the inverse relation type. We report mean reciprocal rank (MRR) and Hits@10. As the benchmark datasets and evaluation protocol coincide with those used in the respective baseline papers, we directly use their reported results rather than reevaluating the models. Full per-dataset results are reported in Table 12.

For WANDER, we use the same link-prediction random-walk sampling protocol as during pretraining (Appendix A). Following FLOCK (Kim et al., 2026), we use walks of length $S = 128$ and adopt its dataset-specific entity-prediction walk and inference sample budgets, reported in Table 13. Here, K_{base} denotes the base walk count, so each structural update samples $3K_{\text{base}}$ walks according to the three starting distributions defined in Appendix A.

D.4 HOMOGENEOUS LINK PREDICTION

Datasets. We follow the homogeneous link-prediction benchmark introduced by ANYGRAPH (Xia & Huang, 2026), using the dataset versions and train/validation/test splits released with its official implementation. Our evaluation comprises ten graphs: CiteSeer, Cora, PubMed, Coauthor CS, DDI, Products Home, P2P Gnutella06, Email-Enron, Proteins Spec1, and Soc Epinions. These datasets cover citation, coauthorship, e-commerce, peer-to-peer, communication, biological, and social networks, and include both attributed and unattributed graphs. Dataset statistics are reported in Table 15.

We adopt a slightly stricter evaluation protocol than the original ANYGRAPH benchmark. Although the graphs are treated as undirected, we found that for some test edges (u, v), the reverse edge (v, u) occurs in the training split. Such cases effectively reveal the test edge during training. We therefore remove all test edges whose reverse is present in the training graph. For attributed datasets, we apply the same feature preprocessing as in the node-classification experiments described below.

Baselines and hyperparameter selection. We compare WANDER against the end-to-end link-prediction models NBFNET (Zhu et al., 2021) and BUDDY (Chamberlain et al., 2023), as well as the graph foundation model ANYGRAPH (Xia & Huang, 2026). For NBFNET and BUDDY, we perform dataset-specific hyperparameter selection based on validation performance. Both models are trained for at most 40 epochs with early stopping using a patience of five epochs.

For NBFNET, we fix the hidden dimension to 32 following the official implementation and search over learning rates $\{10^{-3}, 5 \cdot 10^{-3}\}$, network depths $\{3, 5, 6\}$, and numbers of sampled negative examples $\{1, 32, 256\}$. For BUDDY, we search over hidden dimensions $\{128, 256\}$, learning rates $\{10^{-3}, 5 \cdot 10^{-3}\}$, and the same set of negative-sampling budgets $\{1, 32, 256\}$. The validation-selected hyperparameters for each dataset are reported in Table 14.

Table 9: Dataset statistics for inductive-e, r link prediction datasets. Triples are the number of edges given at training, validation, or test, respectively, whereas *Valid* and *Test* denote triples to be predicted in the validation and test graphs.

Dataset	**Training Graph**			**Validation Graph**				**Test Graph**			
	Entities	Rels	Triples	Entities	Rels	Triples	Valid	Entities	Rels	Triples	Test
FB-25	5190	163	91571	4097	216	17147	5716	4097	216	17147	5716
FB-50	5190	153	85375	4445	205	11636	3879	4445	205	11636	3879
FB-75	4659	134	62809	2792	186	9316	3106	2792	186	9316	3106
FB-100	4659	134	62809	2624	77	6987	2329	2624	77	6987	2329
WK-25	12659	47	41873	3228	74	3391	1130	3228	74	3391	1131
WK-50	12022	72	82481	9328	93	9672	3224	9328	93	9672	3225
WK-75	6853	52	28741	2722	65	3430	1143	2722	65	3430	1144
WK-100	9784	67	49875	12136	37	13487	4496	12136	37	13487	4496
NL-0	1814	134	7796	2026	112	2287	763	2026	112	2287	763
NL-25	4396	106	17578	2146	120	2230	743	2146	120	2230	744
NL-50	4396	106	17578	2335	119	2576	859	2335	119	2576	859
NL-75	2607	96	11058	1578	116	1818	606	1578	116	1818	607
NL-100	1258	55	7832	1709	53	2378	793	1709	53	2378	793
Metafam	1316	28	13821	1316	28	13821	590	656	28	7257	184
FBNELL	4636	100	10275	4636	100	10275	1055	4752	183	10685	597
Wiki MT1 tax	10000	10	17178	10000	10	17178	1908	10000	9	16526	1834
Wiki MT1 health	10000	7	14371	10000	7	14371	1596	10000	7	14110	1566
Wiki MT2 org	10000	10	23233	10000	10	23233	2581	10000	11	21976	2441
Wiki MT2 sci	10000	16	16471	10000	16	16471	1830	10000	16	14852	1650
Wiki MT3 art	10000	45	27262	10000	45	27262	3026	10000	45	28023	3113
Wiki MT3 infra	10000	24	21990	10000	24	21990	2443	10000	27	21646	2405
Wiki MT4 sci	10000	42	12576	10000	42	12576	1397	10000	42	12516	1388
Wiki MT4 health	10000	21	15539	10000	21	15539	1725	10000	20	15337	1703

Unlike NBFNET and BUDDY, ANYGRAPH is used as a pretrained graph foundation model. Its authors provide two checkpoints pretrained on disjoint collections of real-world graphs. For each evaluation dataset, we use the checkpoint whose pretraining set does not contain that dataset, thereby avoiding overlap between pretraining and evaluation data.

WANDER inference. We use the same link-prediction random-walk sampling protocol as during pretraining (Appendix A). For attributed graphs, we use a base walk count of $K_{\text{base}} = 64$ and walk length $S = 64$; for graphs without node features, we use $K_{\text{base}} = 128$ and $S = 128$. Thus, each structural update samples $3K_{\text{base}}$ walks according to the three starting distributions described in Appendix A. Unless stated otherwise, we average predictions over 16 independently sampled forward passes.

Comparison to UniLP. UNILP (Dong et al., 2024) and TFMLINKER (Liao et al., 2026) are two recently proposed graph foundation models for homogeneous link prediction. We do not include them in our main comparison for three reasons. First, both models score links conditioned on a specific pair of endpoints, whereas WANDER conditions on a source node and jointly scores candidate targets. Consequently, evaluating source-conditioned Recall@20 over all candidate nodes would require pairwise models to separately evaluate a large number of candidate edges, making our main evaluation protocol computationally impractical. Second, both methods are designed for unattributed graphs and therefore cannot exploit the node features available in several of our benchmarks. Third, for TFMLinker, we were unable to identify an official implementation at the time of evaluation.

To nevertheless provide a direct comparison, we evaluate WANDER and UNILP on the datasets considered by UNILP using a sampled, source-conditioned ranking protocol. UNILP originally reports Hits@50 by comparing positive edges against sampled negative edges whose endpoints may differ. Such comparisons are less natural for WANDER, whose scores are produced jointly for candidate tails conditioned on a fixed source and are therefore primarily meaningful relative to other candidates for the same query. As a middle ground, we use sampled Recall@20: for each source

Table 10: Dataset statistics for inductive-e link prediction datasets. Triples are the number of edges given at training, validation, or test, respectively, whereas *Valid* and *Test* denote triples to be predicted in the validation and test graphs.

Dataset	**Rels**	**Training Graph**		**Validation Graph**			**Test Graph**		
		Entities	Triples	Entities	Triples	Valid	Entities	Triples	Test
FB-v1	180	1594	4245	1594	4245	489	1093	1993	411
FB-v2	200	2608	9739	2608	9739	1166	1660	4145	947
FB-v3	215	3668	17986	3668	17986	2194	2501	7406	1731
FB-v4	219	4707	27203	4707	27203	3352	3051	11714	2840
WN-v1	9	2746	5410	2746	5410	630	922	1618	373
WN-v2	10	6954	15262	6954	15262	1838	2757	4011	852
WN-v3	11	12078	25901	12078	25901	3097	5084	6327	1143
WN-v4	9	3861	7940	3861	7940	934	7084	12334	2823
NL-v1	14	3103	4687	3103	4687	414	225	833	201
NL-v2	88	2564	8219	2564	8219	922	2086	4586	935
NL-v3	142	4647	16393	4647	16393	1851	3566	8048	1620
NL-v4	76	2092	7546	2092	7546	876	2795	7073	1447
ILPC Small	48	10230	78616	6653	20960	2908	6653	20960	2902
ILPC Large	65	46626	202446	29246	77044	10179	29246	77044	10184
HM 1k	11	36237	93364	36311	93364	1771	9899	18638	476
HM 3k	11	32118	71097	32250	71097	1201	19218	38285	1349
HM 5k	11	28601	57601	28744	57601	900	23792	48425	2124
HM Indigo	229	12721	121601	12797	121601	14121	14775	250195	14904

Table 11: Dataset statistics for transductive link prediction datasets. Entity task denotes the entity-prediction task: h/t is predicting both heads and tails, and t is predicting only tails.

Dataset	**Entities**	**Rels**	**Train**	**Valid**	**Test**	**Entity Task**
CoDEx Small	2034	42	32888	1827	1828	h/t
CoDEx Large	77951	69	551193	30622	30622	h/t
NELL995	74536	200	149678	543	2818	h/t
YAGO3-10	123182	37	1079040	5000	5000	h/t
WDsinger	10282	135	16142	2163	2203	h/t
NELL23k	22925	200	25445	4961	4952	h/t
AristoV4	44949	1605	242567	20000	20000	h/t
DBpedia100k	99604	470	597572	50000	50000	h/t
ConceptNet100k	78334	34	100000	1200	1200	h/t
FB15k-237(10)	11512	237	27211	15624	18150	t
FB15k-237(20)	13166	237	54423	16963	19776	t
FB15k-237(50)	14149	237	136057	17449	20324	t
Hetionet	45158	24	2025177	112510	112510	h/t
Pretraining						
FB15k-237	14541	237	272115	17535	20466	h/t
WN18RR	40943	11	86835	3034	3134	h/t
CoDEx Medium	17050	51	185584	10310	10311	h/t

node, we rank all of its held-out positive neighbors together with 100 sampled non-edges and report the fraction of positive neighbors appearing among the top 20 candidates. We evaluate on two random $70/10/20$ train/validation/test splits.

Table 16 shows that, without node features, the two models achieve essentially identical average performance (85.4% Recall@20), with WANDER outperforming UNILP on Celegans, USAir, and Cora, and UNILP performing better on the remaining four datasets. At the same time, WANDER is substantially faster (under same hardware, A100 GPU) under this evaluation protocol. For the three datasets with node attributes, we additionally report WANDER with features enabled. This improves performance on Cora, CS, and Facebook, respectively, exceeding the featureless UNILP results on all three datasets.

Table 12: Zero-shot knowledge graph link prediction results. Best in each row is **bold**, second-best is underlined.

	Dataset	ULTRA		TRIX		FLOCK		**WANDER**	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Inductive (entity, relation)	FB-25	0.388	0.640	0.393	0.650	**0.404**	**0.664**	**0.404**	0.663
	FB-50	0.338	0.543	0.334	0.547	**0.352**	0.566	**0.352**	**0.568**
	FB-75	0.403	0.604	0.401	0.611	**0.418**	**0.622**	0.408	0.617
	FB-100	0.449	0.642	0.436	0.635	0.452	**0.663**	**0.458**	**0.663**
	WK-25	**0.316**	**0.532**	0.305	0.496	0.280	0.491	0.288	0.477
	WK-50	**0.166**	**0.324**	**0.166**	0.313	0.136	0.278	0.147	0.289
	WK-75	0.365	0.537	0.368	0.513	**0.382**	**0.538**	0.372	0.517
	WK-100	0.164	0.286	0.188	0.299	0.187	0.304	**0.190**	**0.309**
	NL-0	0.342	0.523	**0.385**	0.549	0.381	**0.606**	0.351	0.553
	NL-25	**0.395**	0.569	0.377	0.589	0.345	**0.590**	0.338	0.546
	NL-50	**0.407**	0.570	0.404	0.548	0.366	0.565	0.391	**0.579**
	NL-75	**0.368**	0.547	0.351	0.525	0.311	0.524	0.357	**0.555**
	NL-100	0.471	0.651	0.486	0.676	0.452	0.692	**0.487**	**0.707**
	MT1 tax	0.224	0.305	**0.358**	**0.452**	0.282	0.383	0.281	0.410
	MT1 health	0.298	0.374	0.376	0.457	0.385	0.481	**0.391**	**0.483**
	MT2 org	0.095	0.159	0.091	0.156	0.100	0.163	**0.102**	**0.167**
	MT2 sci	0.258	0.354	0.323	0.465	0.318	0.458	**0.359**	**0.526**
	MT3 art	0.259	0.402	0.284	0.441	0.301	**0.466**	**0.304**	**0.466**
	MT3 infra	0.619	0.755	0.655	0.797	**0.684**	0.821	0.677	**0.822**
	MT4 sci	0.274	0.449	0.290	0.460	0.301	0.463	**0.310**	**0.480**
	MT4 health	0.624	0.737	0.677	0.775	0.680	0.780	**0.701**	**0.787**
	Metafam	0.238	0.644	0.341	0.815	0.476	0.935	**0.504**	**0.941**
	FBNELL	0.485	0.652	0.473	0.660	0.502	0.700	**0.526**	**0.712**
Inductive (entity)	FB-v1	0.498	0.656	0.515	0.682	0.500	0.697	**0.524**	**0.701**
	FB-v2	0.512	0.700	0.525	0.730	0.535	0.737	**0.540**	**0.742**
	FB-v3	0.491	0.654	0.501	0.669	0.511	0.685	**0.514**	**0.693**
	FB-v4	0.486	0.677	0.493	0.687	**0.505**	0.702	0.504	**0.703**
	WN-v1	0.648	0.768	**0.699**	0.791	0.698	0.803	0.690	**0.807**
	WN-v2	0.663	0.765	0.678	0.781	**0.696**	0.790	0.687	**0.793**
	WN-v3	0.376	0.476	0.418	0.541	0.467	0.608	**0.478**	**0.627**
	WN-v4	0.611	0.705	0.648	0.723	**0.653**	0.729	0.650	**0.731**
	NL-v1	0.785	**0.913**	**0.806**	0.898	0.658	0.863	0.677	0.878
	NL-v2	0.526	0.707	0.569	0.768	**0.588**	**0.797**	0.587	0.794
	NL-v3	0.515	0.702	0.558	0.743	0.590	0.783	**0.610**	**0.796**
	NL-v4	0.479	0.712	0.538	0.765	0.555	0.786	**0.572**	**0.795**
	HM 1k	0.059	0.092	**0.072**	**0.128**	0.069	0.119	0.062	0.119
	HM 3k	0.037	0.077	**0.069**	0.119	0.067	0.118	**0.069**	**0.126**
	HM 5k	0.034	0.071	0.062	0.110	0.064	0.116	**0.067**	**0.126**
	HM Indigo	**0.440**	**0.648**	0.436	0.645	0.423	0.638	0.416	0.636
	ILPC Small	0.302	0.443	0.303	0.455	0.309	**0.459**	**0.310**	**0.459**
	ILPC Large	0.290	0.424	0.307	0.428	0.318	0.438	**0.321**	**0.440**
Transductive	NELL995	0.406	0.543	0.472	0.629	0.494	0.655	**0.519**	**0.662**
	NELL23k	0.239	0.408	**0.290**	**0.497**	0.233	0.398	0.249	0.413
	WDsinger	0.382	0.498	**0.511**	**0.609**	0.410	0.528	0.402	0.521
	ConceptNet100k	0.082	0.162	0.193	0.345	0.248	0.453	**0.258**	**0.477**
	CoDEx Small	**0.472**	0.667	**0.472**	**0.670**	0.441	0.644	0.457	0.658
	CoDEx Large	0.338	0.469	0.335	0.469	0.342	0.464	**0.345**	**0.475**
	YAGO310	0.451	0.615	0.409	0.627	0.414	**0.674**	**0.499**	0.666
	AristoV4	0.182	0.282	0.181	0.286	**0.308**	**0.443**	0.295	0.424
	DBpedia100k	0.398	0.576	0.426	0.603	**0.450**	**0.627**	0.436	0.625
	Hetionet	0.257	0.379	**0.279**	**0.420**	0.246	0.371	**0.279**	0.415
	FB15k-237(10)	**0.248**	0.398	0.246	0.393	0.246	0.402	**0.248**	**0.404**
	FB15k-237(20)	0.272	0.436	0.269	0.430	0.273	0.444	**0.277**	**0.451**
	FB15k-237(50)	**0.324**	**0.526**	0.321	0.521	0.319	0.518	0.323	0.523
	Mean	0.366	0.518	0.390	0.548	0.391	0.560	**0.399**	**0.565**
	Mean rank	3.250	3.417	2.657	2.750	2.343	2.222	**1.750**	**1.611**

Table 13: Random-walk configuration used for knowledge-graph link prediction. We follow the entity-prediction evaluation configuration of FLOCK (Kim et al., 2026). Each WANDER structural update samples $3K_{\text{base}}$ walks. The walk length is $S = 128$ for all datasets.

Dataset(s)	**Base walks K_{base}**	**Inference samples**
Inductive (e, r)		
FB-25, FB-50	16	16
FB-75, FB-100	8	16
WK-25, WK-75	4	16
WK-50, WK-100	16	16
NL-0, NL-25, NL-75	2	16
NL-50, NL-100	4	16
MT1-tax	4	16
MT1-health	16	16
MT2-org	32	16
MT2-sci	16	16
MT3-art	32	16
MT3-infra	32	16
MT4-sci	16	16
MT4-health	16	16
Metafam, FBNELL	16	16
Inductive (e)		
FB-v1–v4	16	16
WN-v1–v4	16	16
NL-v1–v4	16	16
ILPC Small	16	16
ILPC Large	64	16
HM 1k	16	16
HM 3k	32	16
HM 5k	64	16
HM Indigo	128	16
Transductive		
CoDEx Small	16	16
CoDEx Large	512	2
NELL-995	128	16
YAGO3-10	512	8
WDsinger	16	16
NELL23k	32	16
FB15k-237(10)	32	16
FB15k-237(20)	64	16
FB15k-237(50)	64	16
Hetionet	512	2
DBpedia100k	512	2
AristoV4	256	4
ConceptNet100k	128	16

Table 14: Val-selected hyperparameters for the NBFNET and BUDDY link-prediction baselines.

	NBFNET			BUDDY		
Dataset	Depth	lr	#Neg.	Hidden	lr	#Neg.
CiteSeer	3	0.005	1	256	0.005	1
Cora	5	0.001	32	128	0.005	32
PubMed	5	0.005	1	256	0.001	256
CS	3	0.005	32	256	0.001	256
DDI	5	0.005	256	256	0.001	256
P2P Gnutella06	6	0.005	1	256	0.005	1
Email Enron	5	0.005	256	256	0.001	256
Proteins Spec1	6	0.005	32	128	0.001	256
SOC Epinions	6	0.005	256	256	0.001	32
Products Home	6	0.005	32	256	0.005	32

D.5 NODE CLASSIFICATION ON SIMPLE GRAPHS

Detailed results for the zero-shot node-classification evaluation are reported in Table 19. We report test accuracy averaged over all available data splits. The reported standard deviation reflects variation across these splits and, for stochastic models (NODEPFN, GRAPHPFN, and WANDER), additionally across three inference passes with different random seeds.

Table 15: Statistics of the homogeneous link-prediction datasets. Test sources are unique heads of test edges. Original is the AnyGraph split. Remaining drops a test edge when either direction already appears in the training graph, which is stricter than AnyGraph's directed protocol; a source is kept only if at least one test edge survives.

Dataset	#Nodes	#Edges	Avg. degree	Clustering Coeff.	Diameter	#Features	#Test sources (original)	#Test sources (filtered)
CiteSeer	3,327	4,552	2.7	0.141	28	3,703	423	175
Cora	2,708	5,278	3.9	0.241	18	1,433	399	192
PubMed	19,717	44,324	4.5	0.060	16	500	2,041	895
CS	18,333	81,894	8.9	0.343	24	6,805	1,973	1,377
DDI	4,267	1,201,400	563.1	0.576	5	–	1,587	1,587
P2P Gnutella06	8,717	31,525	7.2	0.007	9	–	2,107	2,107
Email Enron	36,692	183,831	10.0	0.497	11	–	13,137	4,400
Proteins Spec1	6,568	922,980	281.1	0.267	5	–	6,124	5,442
SOC Epinions	75,888	405,740	10.7	0.138	12	–	17,695	14,511
Products Home	9,790	65,919	13.5	0.321	22	100	4,106	1,681

Table 16: Sampled Recall@20 and test-only wall-clock for UNILP vs. WANDER on two random $70/10/20$ splits. Per source: rank that source's eval neighbors plus 100 sampled non-edges.

	Celegans		USAir		NS		PB		Cora		CS		Facebook	
	R@20	min	R@20	min	R@20	min	R@20	min	R@20	min	R@20	min	R@20	min
UNILP	70.9	57	85.2	45	**94.2**	112	**83.4**	450	75.4	269	**93.1**	187	**95.5**	438
WANDER (w/o feat.)	**77.2**	**3**	**90.6**	**3**	89.4	**6**	80.7	**13**	**76.4**	**15**	91.3	**45**	92.2	**52**
WANDER (w/ feat.)	–	–	–	–	–	–	–	–	90.6	115	97.0	142	95.8	159

Datasets. We evaluate on a benchmark suite of 26 multiclass node-classification datasets, using official data splits whenever available. The suite comprises Cornell, Texas, Wisconsin, and Actor from Pei et al. (2020); the Brazil, Europe, and USA airport networks from Ribeiro et al. (2017); Chameleon and Squirrel from Rozemberczki et al. (2021); Cora, CiteSeer, and PubMed from Yang et al. (2016); Photo, Computers, Coauthor CS, and Coauthor Physics from Shchur et al. (2018); Wiki CS from Mernyei & Cangea (2020); Full DBLP and Full Cora from Bojchevski & Günnemann (2018); Roman Empire and Amazon Ratings from Platonov et al. (2023); HM Categories and Pokec Regions from Bazhenov et al. (2026); and Paris, Shanghai and LA from Liang et al. (2026).

As Pokec Regions contains approximately 1.6 million nodes, we construct a smaller variant, *Pokec 100k*, to facilitate evaluation. We randomly select a training node as the root and retain the first 100,000 nodes visited by a breadth-first search. Within the resulting connected subgraph, we retain labels for the ten most frequent classes and treat nodes belonging to all remaining classes as unlabeled. Labeled nodes retain their original GraphLand random-low split assignments, while unlabeled nodes are excluded from all splits.

Following the concerns about duplicate nodes in Chameleon and Squirrel raised by Platonov et al. (2023), we use the filtered versions of both datasets proposed by the authors.

Whenever official splits are provided, we evaluate on all available splits and average the resulting test performance. For HM Categories, we use the random-low split provided by GraphLand. The Brazil, Europe, USA, Photo, Computers, Full DBLP, Full Cora, Coauthor CS, and Coauthor Physics datasets do not provide official splits. For these datasets, following Zhao et al. (2025); Finkelshtein et al. (2026), we sample 20 training nodes per class and divide the remaining nodes equally between validation and test sets. We repeat this procedure for five random seeds and report results aggregated across the resulting splits.

Table 17 summarizes the main statistics of the evaluation datasets. In addition to standard graph and task statistics, we report *unbiased homophily* as proposed by Mironov & Prokhorenkova (2025).

Baselines. We compare against a strongly tuned three-layer GCN (Kipf & Welling, 2017) with hidden dimension 256, trained for 1000 epochs. For each dataset, we perform a grid search over the learning rate $\{3\cdot10^{-5}, 10^{-4}, 3\cdot10^{-4}, 10^{-3}, 3\cdot10^{-3}\}$, the use of layer normalization and skip connections, and dropout rates in $\{0, 0.1, 0.2\}$. Hyperparameters are selected based on validation

Table 17: Statistics of considered node classification datasets. "# official splits" counts train/val/test masks shipped with the dataset. "–" denotes datasets without an official split; we use a custom protocol there (5 seeds, 20 labeled nodes per class).

Group	Dataset	# nodes	# features	# classes	% train	avg. deg.	unbiased homophily	# official splits
WebKB	Cornell	183	1.7k	5	47.5%	3.0	-0.47	10
	Texas	183	1.7k	5	47.5%	3.0	-0.81	10
	Wisconsin	251	1.7k	5	47.8%	3.6	-0.29	10
Airports	Air Brazil	131	131	4	61.1%	15.3	0.00	–
	Air Europe	399	399	4	20.1%	30.0	-0.12	–
	Air USA	1.2k	1.2k	4	6.7%	22.9	0.43	–
Wiki filtered	Chameleon filt.	890	2.3k	5	46.0%	19.9	0.08	10
	Squirrel filt.	2.2k	2.1k	5	47.4%	42.3	0.02	10
	Actor	7.6k	932	5	48.0%	7.0	0.01	10
Planetoid	Cora	2.7k	1.4k	7	5.2%	3.9	0.92	1
	CiteSeer	3.3k	3.7k	6	3.6%	2.7	0.85	1
	PubMed	19.7k	500	3	0.3%	4.5	0.77	1
Amazon	Amz. Photos	7.7k	745	8	2.1%	31.1	0.93	–
	Amz. Computers	13.8k	767	10	1.5%	35.8	0.91	–
	Wiki CS	11.7k	300	10	5.0%	36.9	0.86	20
Citation full	Full DBLP	17.7k	1.6k	4	0.5%	6.0	0.82	–
	Full Cora	19.8k	8.7k	70	7.1%	6.4	0.97	–
	Roman Empire	22.7k	300	18	50.0%	2.9	-0.49	10
	Amz. Ratings	24.5k	300	5	50.0%	7.6	0.28	10
Coauthor	Co. CS	18.3k	6.8k	15	1.6%	8.9	0.96	–
	Co. Physics	34.5k	8.4k	5	0.3%	14.4	0.95	–
	HM Categories	46.6k	32	21	10.0%	460.9	0.38	1
	Pokec 100k	100k	56	10	4.0%	20.6	0.82	1
City networks	Paris	114.1k	37	10	10.0%	3.2	0.87	1
	Shanghai	183.9k	37	10	10.0%	2.9	0.89	1
	LA	240.6k	37	10	10.0%	2.8	0.89	1
Synth.	Cycles	5.0k	0	4	10.0%	3.0	0.75	1
	Grids	3.1k	0	5	0.3%	3.8	1.00	1

Table 18: GCN grid setting chosen for each dataset. Shared architecture: 3 layers, hidden dimension 256, weight decay 5×10^{-4}.

Group	Dataset	Learning rate	Dropout rate	Layer norm	Skip connection
WebKB	Cornell	10^{-3}	0.1	–	–
	Texas	10^{-3}	0.1	–	–
	Wisconsin	10^{-3}	0.0	✓	✓
Airports	Brazil	10^{-4}	0.1	✓	–
	Europe	3×10^{-4}	0.1	–	–
	USA	10^{-3}	0.2	–	–
Wiki filtered	Chameleon filt.	10^{-3}	0.0	✓	✓
	Squirrel filt.	3×10^{-4}	0.1	✓	–
	Actor	3×10^{-3}	0.2	–	✓
Planetoid	Cora	3×10^{-3}	0.1	–	–
	CiteSeer	3×10^{-3}	0.1	✓	–
	PubMed	3×10^{-3}	0.0	–	–
Amazon	Amz. Photos	10^{-3}	0.2	✓	–
	Amz. Computers	3×10^{-4}	0.0	✓	–
	Wiki CS	10^{-3}	0.1	–	✓
Citation full	Full DBLP	3×10^{-3}	0.1	✓	–
	Full Cora	3×10^{-3}	0.2	✓	–
	Roman Empire	3×10^{-3}	0.2	–	✓
	Amz. Ratings	3×10^{-3}	0.2	✓	✓
Coauthor	Co. CS	3×10^{-3}	0.1	–	✓
	Co. Physics	10^{-3}	0.1	✓	–
	HM Categories	3×10^{-3}	0.2	✓	–
	Pokec 100k	3×10^{-3}	0.2	–	–
City networks	Paris	3×10^{-3}	0.2	✓	–
	Shanghai	10^{-3}	0.2	✓	–
	LA	3×10^{-3}	0.2	✓	–

performance on a single split and subsequently fixed for evaluation on all available splits of the corresponding dataset. The selected hyperparameters are reported in Table 18.

We further consider the graph foundation models GRAPHANY (Zhao et al., 2025), NODEPFN (Choi et al., 2026), and GRAPHPFN (Eremeev et al., 2026), as well as the tabular foundation model TABPFNV3 (Grinsztajn et al., 2026). We reevaluate all models using their official implementations. NODEPFN supports at most 20 classes and can therefore not be evaluated on Full Cora (70 classes) or HM Categories (21 classes). GRAPHPFN is architecturally restricted to 10 output classes, but circumvents this restriction through error-correcting output codes (ECOC); following its default evaluation protocol, we use an ensemble of 10 models. WANDER natively supports an arbitrary number of classes.

Preprocessing. For GraphPFN and WANDER, we follow the transductive preprocessing pipeline provided with GraphPFN. On the GraphLand datasets HM Categories and Pokec 100k, numerical features are transformed using a quantile-normal transformation, fraction-valued features using a quantile-uniform transformation, and categorical features are retained as ordinal encodings. On all remaining datasets, we remove feature columns that are constant over the training nodes and standardize each remaining feature column. For datasets with more than 64 feature dimensions, we subsequently apply PCA to reduce the representation to 64 dimensions. This dimensionality reduction limits memory consumption and more closely matches the feature dimensionalities encountered by GraphPFN and WANDER during pretraining. An evaluation without dimension reduction (PCA) for GraphPFN resulted in a worse mean performance and was not used for the results in Table 4.

For NodePFN and GraphAny, we retain the preprocessing procedures provided by their respective official implementations. NODEPFN additionally performs model selection over truncated-SVD feature dimensions $\{10, 15, 20, 25, 30\}$ and different numbers of smoothing layers. We also preserve this procedure for datasets that are not included in the original NODEPFN evaluation code.

For TabPFNv3 we also apply PCA to reduce feature representations to 64 dimensions. An evaluation without dimension reduction resulted in a worse mean performance and was not used for the results in Table 4.

Inference parameters and scalable evaluation of WANDER. In a standard WANDER forward pass, predictions can be obtained for nodes that are visited by the sampled random walks. At the same time, the in-context learning mechanism relies on sufficiently broad coverage of labeled context nodes. Obtaining such coverage with a single forward pass becomes computationally expensive on large graphs, as it would require sampling a very large number of walks.

We therefore implement a batch-wise node-classification inference procedure that separates the processing of context and query nodes. First, we perform a forward pass in which random walks are initialized exclusively from training nodes, sampled in a class-stratified manner, and cache the resulting intermediate representations of all visited training nodes. We then partition the test nodes into batches of at most 128 by greedy BFS: starting from the highest-degree remaining test node, we collect nearby test nodes in hop order (paths may go through non-test nodes) until the batch is full. This keeps the query nodes in a batch graph-local, so their random walks share neighborhood coverage. For each batch, random walks are initialized exclusively from its query nodes, while the cached training-node representations are supplied as context during the in-context learning update. This allows test nodes to be processed in batches without repeatedly recomputing representations for the labeled context.

Although this procedure is primarily motivated by scalability on large graphs, we use it consistently for all node-classification experiments. We adjust the random-walk budget according to graph size. For graphs with at most $1{,}000$, $1{,}001$–$10{,}000$, $10{,}001$–$40{,}000$, and more than $40{,}000$ nodes, we use base walk counts of 32, 64, 128, and 256, respectively. We use a walk length of 32 for graphs with at most $1{,}000$ nodes and 128 otherwise. In both passes, we retain the total budget of $3K_{\text{base}}$ walks per structural update, only their starting-node distribution differs from the standard sampling scheme of Appendix A.

D.6 Analysis of dataset-specific performance gaps

While WANDER performs strongly on average, its advantage is not uniform across datasets. We investigate two possible contributors to these performance gaps: insufficient random-walk coverage on particularly dense graphs and mismatch between the pretraining prior and the structural regime of the evaluation graph.

Dense graphs. HM Categories and DDI are among the datasets on which WANDER trails the strongest task-specific baseline. Both graphs are unusually dense relative to most of the evaluation suite. In such graphs, a fixed number of sampled walks explores only a small fraction of the many possible local transitions, potentially yielding insufficient structural coverage. Consistent with this explanation, increasing the base walk count on HM Categories from 256 to 2048, using walks of length 32, raises node-classification accuracy from 48.7% to 52.1%. Similarly, increasing the base walk count on DDI from 128 to 512 improves Recall@20 from 13.0% to 14.1%. These results indicate that part of the performance gap on dense graphs can be reduced by allocating more inference-time computation. They also suggest that adapting the walk budget not only to graph size, but also to graph density, may provide a better inference policy.

Prior mismatch. A second possible source of weaker performance is mismatch between the synthetic pretraining prior and particular evaluation graphs. As discussed in Appendix D.13, samples from the WANDER prior span both homophilous and heterophilous regimes but are concentrated toward homophily. Correspondingly, replacing the synthetic component with the GRAPHPFN prior improves performance on several heterophilous node-classification datasets, despite reducing aggregate accuracy. This suggests that the comparatively weaker results on some heterophilous graphs may partly reflect the relative frequency of these regimes during pretraining. However, this evidence is only suggestive: the priors also differ in their graph structures and attribute-generation mechanisms, so the effect cannot be attributed to homophily alone.

D.7 Pretraining on single task

For the single-task variants in Table 5, we follow the same pretraining setup and curriculum as described in Appendix D.2, but restrict training to the respective task. In each curriculum stage, we train for exactly the number of optimization steps that this task receives in the joint setting. For example, a task assigned 40% of the steps in a given stage is trained for 40% of that stage's total number of steps. All other hyperparameters and task-specific data distributions are kept unchanged.

D.8 Compositional generalization to attributed knowledge graphs

During pretraining, WANDER encounters node features on homogeneous graphs and relation types on knowledge graphs, but never both jointly. We test whether these capabilities compose at inference time by evaluating the same pretrained checkpoint zero-shot on ConceptNet100k, WN18RR, and the four inductive WN-v1–v4 graphs in a 2×2 design with MiniLM node features and relation types enabled or disabled. Without relation types, all relations, including inverses, are mapped to a single dummy type; without features, entities are attribute-free. We use filtered head and tail ranking and the default knowledge-graph inference settings from Appendix D.3 and Table 13. Table 6 reports the unweighted mean MRR across WN-v1–v4.

We encode ConceptNet entity strings and WordNet synsets using `all-MiniLM-L6-v2` Wang et al. (2020), representing each synset as "*lemma*: *gloss*." The resulting 384-dimensional embeddings are reduced to 32 dimensions using PCA.

As shown in Table 6, node features and relation types each improve MRR individually, while their combination performs best on every benchmark. Relation types contribute most strongly, particularly on WordNet, whereas MiniLM features provide the largest gains on ConceptNet100k and the inductive WN-v1–v4 graphs.

Table 19: Multiclass node classification results (test accuracy %, mean ± std over all available splits). NODEPFN is shown but excluded from mean and mean rank (missing Full Cora and HM Categories). TABPFNv3 uses node features only (ignores the graph).

Group	Dataset	GCN	TABPFNv3	GRAPHANY	NODEPFN	GRAPHPFN	**WANDER**
WebKB	Cornell	38.9 ± 5.8	71.4 ± 4.2	58.6 ± 6.4	67.3 ± 4.6	70.3 ± 3.0	**76.9 ± 3.4**
	Texas	54.9 ± 6.4	71.1 ± 8.4	67.0 ± 6.0	76.8 ± 5.3	70.5 ± 12.1	**77.8 ± 7.1**
	Wisconsin	60.2 ± 6.5	77.5 ± 2.5	63.5 ± 4.7	79.4 ± 5.2	80.0 ± 5.2	**80.8 ± 5.7**
Airports	Air Brazil	46.2 ± 18.0	33.1 ± 9.3	35.4 ± 15.1	63.9 ± 7.9	**74.6 ± 6.7**	74.1 ± 4.6
	Air Europe	47.0 ± 0.7	39.1 ± 3.0	41.6 ± 6.4	52.6 ± 4.3	**58.0 ± 4.3**	55.5 ± 4.1
	Air USA	47.4 ± 1.6	23.8 ± 2.3	43.5 ± 1.9	60.9 ± 1.4	**63.0 ± 1.7**	62.4 ± 1.8
Wiki filtered	Chameleon filt.	41.6 ± 4.9	46.9 ± 3.4	32.6 ± 3.7	46.0 ± 2.6	**50.9 ± 4.1**	47.9 ± 3.8
	Squirrel filt.	33.2 ± 2.1	41.1 ± 2.4	25.7 ± 1.3	42.5 ± 0.9	**48.0 ± 2.8**	43.9 ± 1.7
	Actor	33.1 ± 0.5	**39.0 ± 1.0**	29.1 ± 0.8	32.7 ± 1.4	38.7 ± 1.1	36.4 ± 0.9
Planetoid	Cora	77.6	66.3	79.4	**82.1 ± 0.3**	74.6 ± 0.5	80.7 ± 1.1
	CiteSeer	61.0	64.9	68.4	67.9 ± 0.2	**69.2 ± 0.4**	66.7 ± 1.0
	PubMed	73.9	70.3	76.3	78.6 ± 0.3	72.0 ± 0.7	**78.9 ± 0.1**
Amazon	Amz. Photos	90.2 ± 0.6	87.2 ± 0.9	90.7 ± 0.8	89.6 ± 0.8	92.5 ± 0.4	**93.1 ± 0.2**
	Amz. Computers	**85.5 ± 1.0**	78.9 ± 1.2	83.0 ± 1.1	81.8 ± 0.7	83.9 ± 1.1	83.5 ± 1.2
	Wiki CS	**79.4 ± 0.6**	73.8 ± 0.6	75.4 ± 0.7	76.0 ± 0.7	79.1 ± 0.6	79.3 ± 0.4
Citation full	Full DBLP	73.7 ± 0.3	69.3 ± 1.2	70.9 ± 0.8	75.3 ± 1.3	74.8 ± 0.6	**79.9 ± 1.2**
	Full Cora	58.7 ± 0.7	49.0 ± 0.7	57.4 ± 0.4	—	57.2 ± 0.6	**59.9 ± 0.4**
	Roman Empire	**79.6 ± 0.4**	65.7 ± 0.5	64.2 ± 0.6	48.8 ± 0.5	67.8 ± 0.4	65.4 ± 0.5
	Amz. Ratings	**51.2 ± 0.7**	49.1 ± 0.5	42.5 ± 0.5	44.2 ± 0.5	48.6 ± 0.5	49.0 ± 0.5
Coauthor	Co. CS	91.4 ± 0.5	90.6 ± 0.3	90.6 ± 0.5	90.8 ± 0.2	91.3 ± 0.4	**92.0 ± 0.2**
	Co. Physics	92.6 ± 0.6	90.9 ± 0.9	92.7 ± 0.5	92.5 ± 0.6	90.9 ± 0.9	**92.9 ± 0.9**
	HM Categories	**66.4**	42.1	37.1	—	55.2 ± 0.7	48.7 ± 0.1
	Pokec 100k	78.4	43.2	44.0	78.1 ± 0.5	83.8 ± 0.2	**86.1 ± 0.5**
City	City Paris	40.6	48.5	20.2	44.8	46.9 ± 0.6	**52.8 ± 0.2**
	City LA	35.9	44.8	17.1	44.3	44.3 ± 0.8	**55.2 ± 0.2**
	City Shanghai	38.1	47.6	19.3	44.7	44.2 ± 0.5	**53.5 ± 0.1**
	Mean accuracy (%)	60.64	58.66	54.85	—	66.55	**68.20**
	Mean rank	3.12	3.69	3.96	—	2.42	**1.73**

D.9 INFERENCE-TIME ADAPTATION

Synthetic GRIDS experiment. To study how the structural range of WANDER can be adapted at inference time, we construct the featureless synthetic benchmark GRIDS as the disjoint union of five 25×25 four-neighbor grids. The resulting graph contains $3{,}125$ nodes and $6{,}000$ undirected edges, with each connected component corresponding to one class and having diameter 48. We sample two training nodes per grid, for 10 labeled nodes in total, and randomly split the remaining nodes into 400 validation and $2{,}715$ test nodes. The nearest labeled node is on average 11 hops from a test node and at most 28 hops away. Since no node features are provided and all five components are structurally identical, prediction requires propagating the sparse supervision available elsewhere in the same connected component.

We evaluate the pretrained WANDER checkpoint zero-shot, without any dataset-specific finetuning, using the node-classification inference procedure from Appendix D.5. To vary the available structural context without changing the model parameters, we evaluate the same checkpoint with walk lengths 4, 16, and 64, while keeping the remaining inference settings fixed. For the context-aware sampling variant, we use length-64 walks, but retain only walks that visit at least one labeled context node. The visualizations in Figure 2 show the predicted probability of the true class for one of the five grid components.

Featureless real-world graphs. We additionally evaluate on PubMed, Full DBLP, Coauthor CS, and Coauthor Physics, whose training splits provide comparatively sparse supervision (see Table 17). We remove all node features and retain only graph structure and the observed training labels. We compare GRAPHANY, NODEPFN, GRAPHPFN, and WANDER in the zero-shot setting, without dataset-specific finetuning. The baseline GFMs are evaluated using their official implementations and preprocessing procedures, while WANDER uses the graph-size-dependent random-walk budget described in Appendix D.5. We report test accuracy averaged over all available splits.

Table 20: Test accuracy on the synthetic GRIDS dataset and selected real-world graphs without node features.

	GRIDS	PubMed	Full DBLP	Co. CS	Co. Physics
GRAPHANY	20.0	18.0	44.7 ± 0.0	22.6 ± 0.0	50.5 ± 0.0
NODEPFN	26.6 ± 0.0	39.5 ± 0.0	45.9 ± 1.0	28.3 ± 0.5	51.2 ± 0.3
GRAPHPFN	23.3 ± 0.7	49.9 ± 2.6	61.8 ± 1.6	59.2 ± 3.5	69.4 ± 1.2
WANDER	96.6 ± 0.5	$\mathbf{73.7 \pm 0.7}$	65.9 ± 3.9	84.1 ± 0.6	82.2 ± 5.0
WANDER (filtered)	$\mathbf{100.0 \pm 0.0}$	72.8 ± 0.5	$\mathbf{70.0 \pm 4.6}$	$\mathbf{84.6 \pm 0.6}$	$\mathbf{85.8 \pm 2.7}$

D.10 INFERENCE-TIME EFFICIENCY AND COMPUTE–PERFORMANCE TRADE-OFFS

Runtime. Table 21 compares the wall-clock cost of zero-shot WANDER inference with that of training and evaluating dataset-specific end-to-end models. We consider Cora link prediction with NBFNET and Full DBLP node classification with a GCN. All measurements were obtained on a single NVIDIA RTX PRO 6000 GPU. For WANDER, we use the default inference configuration: batches of eight source nodes with 64 walks of length 64 for link prediction, and batches of 128 nodes with 128 walks of length 128 for node classification. The reported end-to-end runtimes include hyperparameter selection.

Scoring an individual batch with WANDER takes only a few seconds. Nevertheless, evaluating an entire graph remains computationally substantial, particularly for node classification. With the default 16 inference samples, a full evaluation takes approximately 11 minutes on Cora and 21 minutes on Full DBLP. The number of inference samples can, however, be reduced without retraining the model. With a single inference sample, the corresponding runtimes decrease to 37 seconds and 78 seconds, respectively, making zero-shot WANDER inference faster than training and selecting the dataset-specific baselines on both datasets.

Compute–performance trade-offs. A useful property of WANDER is that its inference-time computation can be adjusted without retraining: we may vary the number of sampled walks, their length, and the number of inference samples whose logits are averaged. Figure 5 quantifies the resulting trade-off between predictive performance and inference speed. Starting from the default configuration, we independently reduce the number of walks and the walk length to one half and one quarter of their dataset-specific defaults, or reduce the number of inference samples from 16 to 8 or 1. For node classification, changes to the random-walk configuration are applied to both the

Table 21: Wall-clock runtimes on a single NVIDIA RTX PRO 6000 GPU. Times are reported for one seed and, for Full DBLP, one data split.

	Cora (LP)		Full DBLP (NC)	
	WANDER	NBFNET	WANDER	GCN
Grid search	–	13 min 36 s	–	1 min 28 s
Single batch (1 inference sample)	1.70 s	1.1 ms	1.13 s	2.0 ms
Single batch (16 inference sample)	27.2 s	–	18.1 s	–
Full eval (1 inference sample)	37 s	13 min 36 s	1 min 18 s	1 min 28 s
Full eval (16 inference samples)	10 min 58 s	–	21 min 16 s	–

context and query passes of the scalable inference procedure. All other settings remain fixed. We report performance and wall-clock speedup relative to the default configuration, averaged across the homogeneous link-prediction and node-classification benchmarks, respectively.

Reducing the number of inference samples offers the most favorable trade-off: eight samples roughly halve runtime with little performance loss, while one sample provides a larger speedup at a higher cost in performance. Reducing the number of walks also degrades performance gradually, whereas shorter walks cause the largest drop, particularly for link prediction.

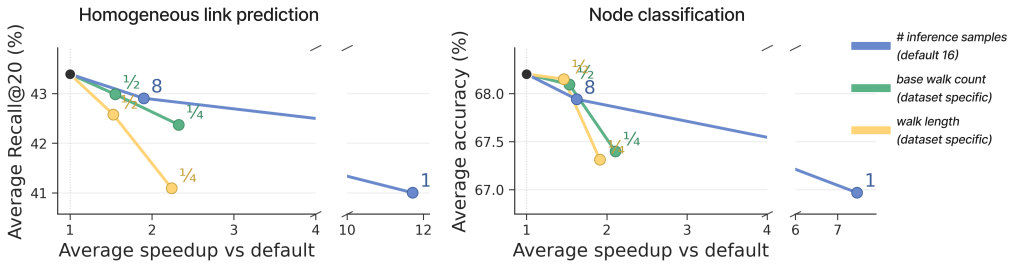


Figure 5: Compute–performance trade-offs obtained by reducing the number of walks, walk length, or inference samples relative to the default configuration. Results are averaged across the datasets of each benchmark.

D.11 HIGHER-ORDER STRUCTURAL PATTERNS

To evaluate WANDER's ability to identify higher-order structural motifs that are challenging for GNNs bounded by the expressive power of 1-WL, we construct a synthetic four-class node-classification benchmark, CYCLES.

Dataset. The graph is featureless, exactly 3-regular, and contains $n = 5000$ nodes. We plant 200 triangles, 150 four-cycles, and 120 five-cycles as pairwise vertex-disjoint motifs, yielding 600 nodes in each motif class. Each motif node is connected by one edge to a distinct node in the remaining 3200-node backbone.

The backbone is completed by matching the remaining degree stubs while rejecting self-loops, duplicate edges, and edges between nodes at graph distance of most three. We further reject graphs containing any C_3, C_4, or C_5 outside the planted motifs, ensuring that all such cycles are known by construction. Nodes are labeled by membership in the backbone, C_3, C_4, or C_5. We use stratified $10\%/20\%/70\%$ train/validation/test splits.

Evaluation protocol. We compare against GCN (Kipf & Welling, 2017) and GRAPHPFN (Eremeev et al., 2026). GCN uses the same hyperparameter search as in Appendix D.5, while GRAPHPFN is finetuned using the authors' official procedure. For WANDER, we use walk length 16 and base walk count 32 during finetuning and evaluation, and finetune for 200 epochs with 32 batches per epoch and batch size 16. All other hyperparameters match pretraining, only training nodes contribute to the supervised loss, and checkpoint selection is based on validation performance.

Table 22: Accuracy on CYCLES (finetuned).

	CYCLES
Majority	64.0
GCN	63.3
GRAPHPFN	67.4
WANDER	**97.5**

As shown in Table 22, GRAPHPFN reaches 67.4%, only modestly above the 64.0% majority baseline, whereas WANDER reaches 97.5%, almost perfectly separating nodes by their cycle participation.

D.12 FINETUNING

We additionally evaluate how much WANDER benefits from dataset-specific finetuning on the link-prediction benchmarks.

Knowledge graph link prediction. We finetune using the pretraining learning rate of 5×10^{-5}, no weight decay, and 256 negative samples per query. Random-walk sampling follows the default knowledge-graph evaluation protocol described in Appendix D.3. We train on the full training split for up to five epochs and select the checkpoint with the highest validation MRR. The resulting performance is reported in Table 23 and compared to the finetuned performance of ULTRA, TRIX and FLOCK as reported in the respective papers.

Finetuning increases mean MRR from 0.399 to 0.433 and Hits@10 from 0.565 to 0.590. The finetuned model thereby surpasses the strongest specialized baseline on average, showing that WANDER retains its advantage in knowledge-graph reasoning when all methods are adapted to each target graph.

Homogeneous link prediction. We use the same finetuning protocol for homogeneous link prediction, with random-walk sampling following the corresponding default evaluation setup. Since not all datasets provide a validation split, we do not perform dataset-specific checkpoint selection and instead evaluate the checkpoint after five finetuning epochs for every dataset. Results are reported in Table 24.

Finetuning improves mean Recall@20 from 43.39% to 46.34%, with gains on nine of the ten datasets. The largest improvements occur on PubMed, Coauthor CS, and Proteins, whereas performance decreases slightly on DDI. Thus, the pretrained model generally adapts effectively to individual graphs in the homogeneous link-prediction setting, although the benefit depends on the target graph.

Node classification. We finetune the pretrained WANDER checkpoint separately on each dataset using only the official training nodes of split 0. Each epoch comprises five optimizer steps with 16 query nodes each. To provide additional supervision, at every step we also hide the labels of a randomly sampled 25% of the training nodes, capped at $1{,}024$, and compute the loss on those hidden nodes encountered in at least one forward pass. During training, we use 128 walks and the size-dependent walk length specified in Appendix D.5. We train for at most 100 epochs, with early stopping after 15 epochs without improvement in validation accuracy. For each dataset, we select the learning rate from 10 logarithmically spaced values between 5×10^{-6} and 5×10^{-4} based on validation accuracy. The selected checkpoint is evaluated on the split-0 test nodes using the same size-tier walk budget as in the zero-shot evaluation. We report the mean and standard deviation over three seeds in Table 25.

Finetuning raises mean accuracy from 68.26% to 70.37%, improving performance on 20 of the 26 graphs. Note that zero-shot mean accuracy differs from earlier tables, since we only considered split 0 here. The largest gains occur on Roman Empire and several Amazon, HM Categories, and city-network datasets, indicating that dataset-specific supervision can substantially adapt the pretrained representations when the target graph differs from the pretraining distribution. However, gains are not uniform, with the largest decreases occurring on the two small datasets Cornell and Wisconsin. Overall, finetuning provides useful adaptation from an already strong zero-shot initialization. Future

Table 23: Finetuned knowledge graph link prediction results. Best in each row is **bold**, second-best is underlined.

	Dataset	ULTRA (ft.)		TRIX (ft.)		FLOCK (ft.)		WANDER		WANDER (ft.)	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Inductive (entity, relation)	FB-25	0.383	0.635	0.393	0.650	0.405	**0.666**	0.404	0.663	**0.406**	0.665
	FB-50	0.334	0.538	0.334	0.547	**0.357**	**0.570**	0.352	0.568	0.345	0.562
	FB-75	0.400	0.598	0.401	0.611	**0.425**	0.630	0.408	0.617	0.424	**0.631**
	FB-100	0.444	0.643	0.436	0.633	**0.460**	**0.668**	0.458	0.663	0.458	0.666
	WK-25	**0.321**	**0.535**	0.300	0.493	0.298	0.506	0.288	0.477	0.293	0.493
	WK-50	0.140	0.280	**0.166**	**0.313**	0.127	0.260	0.147	0.289	0.126	0.272
	WK-75	0.380	0.530	0.368	0.513	**0.405**	**0.556**	0.372	0.517	0.401	**0.556**
	WK-100	0.168	0.286	0.188	0.299	0.187	0.306	**0.190**	**0.309**	0.182	0.297
	NL-0	0.329	0.551	0.385	0.549	**0.418**	0.619	0.351	0.553	0.403	**0.628**
	NL-25	**0.407**	0.596	0.377	0.589	0.405	**0.626**	0.338	0.546	0.390	0.596
	NL-50	**0.418**	0.595	0.405	0.555	0.391	0.562	0.391	0.579	0.403	**0.605**
	NL-75	**0.374**	0.570	0.351	0.525	0.344	0.544	0.357	0.555	0.360	**0.582**
	NL-100	0.458	0.684	0.482	0.691	0.486	**0.714**	0.487	0.707	**0.495**	0.706
	MT1 tax	0.330	0.459	0.397	0.508	**0.413**	0.497	0.281	0.410	0.405	**0.519**
	MT1 health	0.380	0.467	0.376	0.457	**0.394**	**0.493**	0.391	0.483	**0.394**	0.492
	MT2 org	0.104	0.170	0.098	0.162	0.107	0.174	0.102	0.167	**0.108**	**0.176**
	MT2 sci	0.311	0.451	0.331	0.526	0.366	0.525	0.359	0.526	**0.378**	**0.542**
	MT3 art	0.306	0.473	0.289	0.441	**0.330**	0.483	0.304	0.466	0.329	**0.491**
	MT3 infra	0.657	0.807	0.672	0.810	0.709	0.838	0.677	0.822	**0.713**	**0.846**
	MT4 sci	0.303	0.478	0.305	0.482	0.324	0.509	0.310	0.480	**0.327**	**0.512**
	MT4 health	0.704	0.785	0.702	0.785	0.711	**0.790**	0.701	0.787	**0.714**	**0.790**
	Metafam	0.997	**1.000**	0.997	**1.000**	0.992	**1.000**	0.504	0.941	**1.000**	**1.000**
	FBNELL	0.481	0.661	0.478	0.655	**0.531**	**0.714**	0.526	0.712	0.525	0.712
Inductive (entity)	FB-v1	0.509	0.670	0.515	0.682	**0.549**	**0.721**	0.524	0.701	**0.549**	**0.721**
	FB-v2	0.524	0.710	0.525	0.730	0.553	0.754	0.540	0.742	**0.560**	**0.755**
	FB-v3	0.504	0.663	0.501	0.669	**0.528**	**0.696**	0.514	0.693	0.525	0.695
	FB-v4	0.496	0.684	0.493	0.687	0.510	0.702	0.504	0.703	**0.515**	**0.705**
	WN-v1	0.685	0.793	0.705	0.798	**0.715**	0.811	0.690	0.807	0.713	**0.814**
	WN-v2	0.679	0.779	0.682	0.780	0.702	0.795	0.687	0.793	**0.706**	**0.801**
	WN-v3	0.411	0.546	0.425	0.543	0.494	**0.627**	0.478	**0.627**	**0.496**	0.625
	WN-v4	0.614	0.720	0.650	0.722	**0.665**	0.741	0.650	0.731	0.663	**0.744**
	NL-v1	0.757	0.878	**0.804**	0.899	0.762	**0.928**	0.677	0.878	0.789	0.888
	NL-v2	0.575	0.761	0.571	0.764	0.612	0.806	0.587	0.794	**0.616**	**0.812**
	NL-v3	0.563	0.755	0.571	0.759	0.606	0.803	0.610	0.796	**0.620**	**0.809**
	NL-v4	0.469	0.733	0.551	0.772	0.572	**0.801**	0.572	0.795	**0.587**	0.798
	HM 1k	0.042	0.100	**0.072**	0.128	0.071	**0.153**	0.062	0.119	0.066	0.132
	HM 3k	0.030	0.090	0.069	0.119	0.067	**0.153**	0.069	0.126	**0.078**	0.150
	HM 5k	0.025	0.068	**0.074**	0.118	0.061	0.130	0.067	0.126	0.060	**0.136**
	HM Indigo	0.432	0.639	**0.436**	**0.645**	0.418	0.633	0.416	0.636	0.411	0.636
	ILPC Small	0.303	0.453	0.303	0.455	0.305	0.454	**0.310**	0.459	**0.310**	**0.462**
	ILPC Large	0.308	0.431	0.310	0.431	0.320	0.441	**0.321**	0.440	**0.321**	**0.442**
Transductive	NELL995	0.509	0.660	0.506	0.648	0.531	0.665	0.519	0.662	**0.565**	**0.681**
	NELL23k	0.268	0.450	0.306	0.536	0.280	0.465	0.249	0.413	**0.327**	**0.552**
	WDsinger	0.417	0.526	**0.502**	0.620	0.435	0.543	0.402	0.521	**0.502**	**0.633**
	ConceptNet100k	0.310	0.529	0.340	0.564	0.352	0.580	0.258	0.477	**0.381**	**0.614**
	CoDEx Small	**0.490**	**0.686**	0.484	0.676	0.463	0.648	0.457	0.658	0.473	0.663
	CoDEx Large	0.343	0.478	**0.348**	**0.481**	0.342	0.467	0.345	0.475	0.347	0.477
	YAGO310	0.557	0.710	0.541	0.702	0.552	0.700	0.499	0.666	**0.575**	**0.718**
	AristoV4	0.343	0.496	0.345	0.499	0.383	0.523	0.295	0.424	**0.387**	**0.530**
	DBpedia100k	0.436	0.603	0.457	0.619	0.470	0.623	0.436	0.625	**0.472**	**0.630**
	Hetionet	**0.399**	**0.538**	0.394	0.534	0.314	0.465	0.279	0.415	0.338	0.478
	FB15k-237(10)	0.254	0.411	0.253	0.408	**0.260**	0.420	0.248	0.404	**0.260**	**0.422**
	FB15k-237(20)	0.274	0.445	0.273	0.441	**0.284**	**0.459**	0.277	0.451	0.282	0.457
	FB15k-237(50)	**0.325**	**0.528**	0.322	0.522	0.317	0.517	0.323	0.523	0.322	0.523
	Mean	0.407	0.562	0.418	0.569	0.427	0.583	0.399	0.565	**0.433**	**0.590**
	Mean rank	3.796	3.898	3.435	3.657	2.380	2.324	3.463	3.370	**1.926**	**1.750**

Table 24: Finetuned homogeneous link prediction (Recall@20 %).

	CiteSeer	Cora	PubMed	CS	DDI	Products Home	P2P Gnutella	Email Enron	Proteins Spec1	SOC Epinions	Mean	Mean rank
WANDER	65.88	56.78	31.73	64.62	12.98	71.12	9.06	69.78	31.40	20.59	43.39	1.90
WANDER (ft.)	69.20	60.90	39.70	70.00	11.90	72.60	9.30	71.90	36.60	21.30	46.34	1.10

Table 25: Node classification finetuning versus zero-shot WANDER on split 0 only (test accuracy %, mean ± std over 3 seeds).

Group	Dataset	Zero-shot	Finetuned
WebKB	Cornell	**74.77 ± 1.27**	72.07 ± 1.27
	Texas	75.68 ± 2.21	**80.18 ± 1.27**
	Wisconsin	**76.47 ± 0.00**	71.24 ± 0.92
Airports	Air Brazil	78.21 ± 1.81	**80.77 ± 0.00**
	Air Europe	57.29 ± 2.06	57.29 ± 0.29
	Air USA	61.20 ± 0.08	**61.62 ± 0.67**
Wiki filtered	Chameleon filt.	**50.52 ± 1.11**	49.31 ± 0.64
	Squirrel filt.	43.58 ± 0.79	**45.80 ± 0.65**
	Actor	37.61 ± 0.20	**39.65 ± 0.38**
Planetoid	Cora	80.80 ± 1.02	**81.90 ± 0.80**
	CiteSeer	66.50 ± 1.28	**67.00 ± 1.04**
	PubMed	78.87 ± 0.24	**82.10 ± 0.90**
Amazon	Amz. Photos	92.89 ± 0.09	**93.20 ± 0.09**
	Amz. Computers	84.72 ± 0.07	**86.87 ± 0.14**
	Wiki CS	79.32 ± 0.08	**80.59 ± 0.06**
Citation full	Full DBLP	81.11 ± 0.17	**81.58 ± 0.14**
	Full Cora	**59.94 ± 0.12**	59.27 ± 0.22
	Roman Empire	64.41 ± 0.11	**83.18 ± 0.19**
	Amz. Ratings	49.35 ± 0.12	**55.45 ± 0.11**
Coauthor	Co. CS	92.21 ± 0.06	**92.84 ± 0.08**
	Co. Physics	93.01 ± 0.24	**93.62 ± 0.24**
	HM Categories	48.72 ± 0.11	**54.13 ± 0.12**
	Pokec 100k	**86.05 ± 0.49**	85.95 ± 0.07
City	City Paris	52.83 ± 0.20	**55.44 ± 0.12**
	City LA	55.16 ± 0.20	**58.30 ± 0.18**
	City Shanghai	53.47 ± 0.10	**60.35 ± 0.05**
	Mean accuracy (%)	68.26	**70.37**

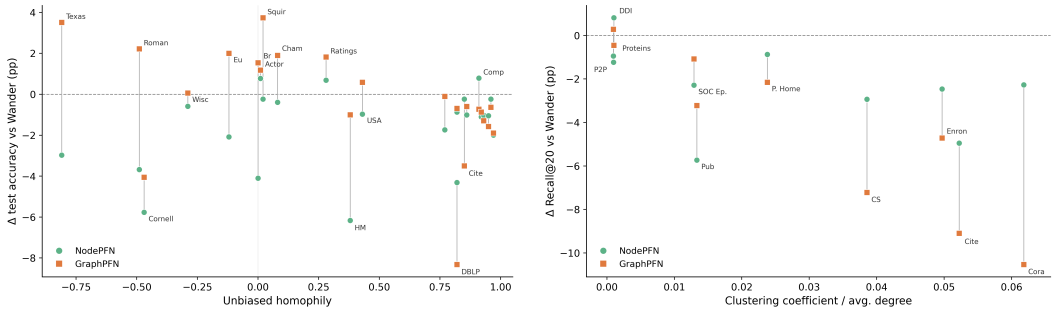
work could investigate protocols that adapt the optimization and sampling budgets to graph size and the amount of available supervision.

D.13 PRIOR ABLATIONS

To assess the effect of the synthetic graph distribution, we repeat WANDER's pretraining with either the NODEPFN or GRAPHPFN prior in place of our synthetic attributed-graph prior. In both ablations, we retain the original mixture of synthetic and real-world knowledge-graph training episodes, including the same three knowledge graphs; only the generator of synthetic graphs changes. The architecture and remaining training setup are held fixed.

Table 26 reports the aggregate results. The WANDER prior yields the highest mean node classification accuracy and homogeneous link-prediction Recall@20; knowledge graph link-prediction MRR changes only marginally.

Figure 6 shows where the differences arise. In homogeneous link prediction, both alternative priors underperform particularly on graphs with high clustering relative to their average degree, consistent with the broader coverage of this structural regime in Figure 3. For node classification, the pattern is less uniform: GRAPHPFN's prior improves accuracy on several heterophilous graphs, even though its aggregate accuracy is lower. One possible explanation is that homophilous graphs make



(a) Change in node-classification accuracy versus unbiased homophily.

(b) Change in link-prediction Recall@20 versus clustering coefficient divided by average degree.

Figure 6: Effect of replacing WANDER's synthetic attributed-graph prior with the NODEPFN or GRAPHPFN prior. Points show results on individual real-world datasets; vertical differences are measured relative to the model trained with the WANDER prior, in percentage points.

Table 26: Ablation of the synthetic attributed-graph pretraining prior.

Synthetic prior	KG LP MRR	Node cls. Acc. (%)	Hom. LP R@20 (%)
WANDER prior	**0.399**	**68.20**	**43.39**
NODEPFN prior	0.397	66.44	<u>41.16</u>
GRAPHPFN prior	<u>0.398</u>	<u>67.33</u>	39.53

Table 27: Ablations of the three layer components of WANDER.

Model	KG LP MRR	Node cls. Acc. (%)	Hom. LP R@20 (%)
WANDER	**0.399**	**68.20**	**43.39**
w/o intra-node update	0.398	60.23	38.1
w/o ICL update	0.393	61.02	37.1
w/o stochastic structural update	0.001	51.11	15.6

up a larger share of the WANDER prior, leaving its pretrained model less adapted to heterophily. These interpretations remain tentative, since the priors differ in other aspects of graph and attribute generation as well.

D.14 ARCHITECTURE ABLATIONS

We assess the contribution of the three update operations that constitute each WANDER layer: the intra-node update, the in-context-learning (ICL) update, and the stochastic structural update. We train three variants in which one of these operations is removed from every layer. Without the intra-node update, information cannot be exchanged between the feature, label, and structure tokens associated with the same node. Without the ICL update, tokens cannot directly attend to the labeled context nodes within their respective channels. Finally, without the stochastic structural update, no graph information can be used, and the model falls back to a tabular model.

The results confirm that all three updates contribute, although their importance depends on the task. Removing the intra-node update has almost no effect on KG link prediction, which is expected because the knowledge graphs contain no node features and prediction is driven primarily by relation-conditioned structural information. In contrast, the same ablation substantially degrades node classification and homogeneous link prediction, where intra-node mixing is needed to combine attributes with label and structural tokens. Removing the ICL update similarly hurts both attributed tasks and causes a smaller reduction on knowledge graphs. The stochastic structural update is essential across all settings: without it, performance drops sharply, and KG link prediction becomes effectively impossible in the absence of node attributes.