KGPFN: UNLOCKING THE POTENTIAL OF KNOWLEDGE GRAPH FOUNDATION MODEL VIA IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge graph (KG) foundation models aim to generalize to graphs with unseen entities and relations by learning transferable relational structure. Most existing methods, however, focus on relation-level universality, leaving in-context learning, the other pillar of foundation models, largely unexplored for KG reasoning. Context in KGs is structured and heterogeneous: accurate prediction requires conditioning both on the local neighborhood of the query entities and on global context that summarizes how the query relation behaves across many instances. We propose KGPFN, a KG foundation model built on a Prior-Data Fitted Network (PFN) that combines transferable relational representations with inference-time in-context learning over structured context. KGPFN learns relation representations by message passing on relation graphs and extracts multi-scale local context from the intermediate head representations of a multi-layer NBFNet. It then builds relation-specific global context from positive and negative examples of the query relation, together with their local structural representations, and aggregates this context with feature-level and sample-level attention. Through multi-graph pretraining, KGPFN learns to combine structural representations with labeled contextual evidence without inference-time parameter updates. On 57 knowledge graphs, KGPFN achieves the best average MRR both without and with fine-tuning, and context sensitivity analyses highlight the value of negative context examples.

1 INTRODUCTION

Foundation models have become a dominant paradigm, generalizing through large-scale pretraining and adapting to new tasks through context at inference time. Large language models (OpenAI, 2023; Liu et al., 2024; Touvron et al., 2023) show that a single pretrained model can perform diverse tasks specified by the prompt and a few in-context demonstrations (Agarwal et al., 2024). Foundation models for structured data increasingly follow the Prior-Data Fitted Network (PFN) (Nagler, 2023) paradigm, in which a model pretrained on synthetic tasks infers a new task from a small labeled context set at inference time. Examples include tabular models (Hollmann et al., 2022; Grinsztajn et al., 2025; Qu et al., 2025; 2026; Zhang et al., 2025a), time-series models (Taga et al., 2025), and extensions to relational data (Wang et al., 2026). These models share two properties: *broad transferability* across tasks and domains with little or no retraining, and *strong in-context learning* from labeled examples provided at inference time.

Knowledge graphs (KGs) represent relational data in applications such as question answering (Bai et al., 2023; Gao et al., 2026), recommendation (Cao et al., 2019), and drug discovery (Gao et al., 2025), and underpin neural graph databases that manage and query graphs with learned representations (Bai et al., 2025; Besta et al., 2022; Ren et al., 2023; Xie et al., 2026; Li et al., 2026). Traditional KG embeddings (Bai et al., 2022; Ren et al., 2020; Bordes et al., 2013) learn graph-specific entity representations and must be retrained for each new graph. KG foundation models (Galkin et al., 2023; Huang et al., 2025; Cui et al., 2024; Zhang et al., 2025b) instead learn transferable relational knowledge that generalizes across KGs with different entity and relation vocabularies, enabling zero-shot or few-shot adaptation to unseen graphs.

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.

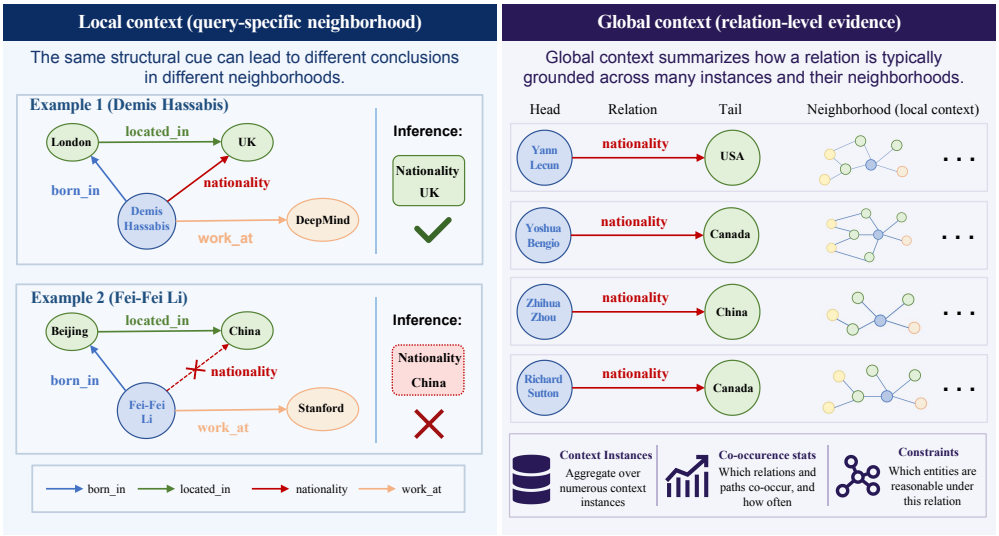


Figure 1: (a) **Local context.** The same structural cue may support different conclusions depending on the query-specific neighborhood. (b) **Global context.** Global context summarizes how a relation is typically instantiated across many examples and their neighborhoods.

Existing KG foundation models have advanced transferable relational reasoning, and KGICL (Cui et al., 2024) further incorporates in-context examples in the form of prompt graphs. Yet transferable structural patterns alone may not capture how a relation is instantiated in a particular graph, which calls for conditioning explicitly on relation-specific positive and negative examples at inference time. Given the success of PFN-style structured foundation models (Hollmann et al., 2022; Grinsztajn et al., 2025; Zhang et al., 2025a), extending this paradigm to KGs is a natural direction, but not a straightforward one. Unlike text or tables, KG context is structured and heterogeneous: rather than a single surrounding window, it combines (i) *local context*, the query-specific neighborhood in which reasoning takes place, and (ii) *global context*, the relation-specific evidence accumulated across many instances.

Local context is the subgraph around the query entities. The same relation can call for different predictions depending on this neighborhood: a reusable structural cue may be reliable in one subgraph and misleading in another. As shown in Fig. 1 (left), the pattern $(\texttt{person}, \texttt{born_in}, \texttt{city}) \wedge (\texttt{city}, \texttt{located_in}, \texttt{nation})$ often indicates $(\texttt{person}, \texttt{nationality}, \texttt{nation})$. For Demis Hassabis, $(\texttt{Demis Hassabis}, \texttt{born_in}, \texttt{London})$ and $(\texttt{London}, \texttt{located_in}, \texttt{UK})$ correctly suggest $(\texttt{Demis Hassabis}, \texttt{nationality}, \texttt{UK})$. For Fei-Fei Li, the analogous path through $\texttt{Beijing}$ points to $\texttt{China}$, yet nationality is not determined by birthplace alone: other edges in her neighborhood, such as $(\texttt{Fei-Fei Li}, \texttt{works_at}, \texttt{Stanford})$, provide complementary evidence. A model must therefore condition on the local neighborhood to decide when a generic pattern applies and when it should be overridden.

Global context captures how a relation behaves across many instances rather than within a single neighborhood. As shown in Fig. 1 (right), it can be formed from representative instances of the query relation together with their local neighborhoods. This collection offers a distributional view of how the relation is grounded in graph structure: which auxiliary relations and multi-hop paths tend to co-occur with it, how often they do so, and how they constrain plausible heads and tails. Relational universality and context are thus complementary: the former captures reusable relational structure, while local and global context indicate how that structure applies to a particular graph and query. This motivates combining transferable structural representations with relation-specific in-context examples.

Building on this analysis, we propose **KGPFN**, which unlocks the potential of **K**nowledge **G**raph foundation models through in-context learning with a **P**rior-data **F**itted **N**etwork. KGPFN couples transferable structural representations with inference-time use of context. It first runs message passing over the relation graph to obtain relation representations that capture transferable, relation-level

structure. For local context, a multi-layer NBFNet computes neighborhood-aware representations of the query entities. For global context, we retrieve positive and negative instances of the query relation together with their local neighborhoods, and process them with a PFN that combines feature-level and sample-level attention. Pretrained on multiple KGs, KGPFN learns to jointly exploit structural regularities and contextual evidence, enabling zero-shot transfer and in-context adaptation to unseen graphs. Our main contributions are as follows:

- **Local and global context for KG foundation models.** We formulate KG context as local context, the query-specific neighborhood, and global context, labeled positive and negative instances of the query relation, moving beyond reliance on transferable relational patterns alone.
- **In-context learning over KG context with Prior-data Fitted Networks.** KGPFN performs in-context learning over this structured context: feature-level attention models structural signals within each instance, and sample-level attention captures regularities across instances, enabling adaptation to unseen graphs without parameter updates.
- **Evaluation on 57 knowledge graphs.** Across transductive, inductive, and fully inductive settings, KGPFN achieves the best average MRR both without and with fine-tuning. Sensitivity analyses clarify the roles of local-context depth and of positive and negative global examples.

2 Related Work

Knowledge graph foundation models. KG foundation models aim to capture relational invariances that transfer across heterogeneous KGs, enabling fully inductive, zero-shot completion on unseen entities and relations. ULTRA (Galkin et al., 2023) builds a relation graph from four fundamental interaction patterns and learns transferable, query-conditioned relation representations. TRIX (Zhang et al., 2025b) increases expressiveness by encoding which entities participate in relation interactions, and jointly updates entity and relation embeddings, which also enables efficient relation prediction. MOTIF (Huang et al., 2025) shows that many KG foundation models rely on binary motifs, and that higher-order motifs on relational hypergraphs yield greater expressiveness. KGICL (Cui et al., 2024) frames KG reasoning as in-context learning, using a few query-centered prompt graphs (e.g., 5-shot) and a unified tokenizer to align entities and relations across KGs. FLOCK (Kim et al., 2026) instead captures local context by learning from random walks.

In-context learning for structured data via Prior-data Fitted Networks. Prior-data Fitted Networks (PFNs) (Nagler, 2023) cast in-context learning as approximate Bayesian inference: a model pretrained on synthetic tasks drawn from a prior predicts query labels from labeled context examples in a single forward pass, without task-specific optimization. TabPFN (Hollmann et al., 2022) applies this idea to tables by treating rows as context examples; TabPFN-v2.5 (Grinsztajn et al., 2025), TabICL (Qu et al., 2025), and Limix (Zhang et al., 2025a) further improve robustness and scalability. PFN-based models have also been extended to time series (Taga et al., 2025), relational databases (Wang et al., 2026), and graphs (Eremeev et al., 2025; Liao et al., 2026). Compared with KGICL's few-shot prompt graphs, KGPFN conditions on a larger labeled set of positive and negative examples of the query relation. Compared with PFN-based graph models, it targets multi-relational KGs whose entity and relation vocabularies change across graphs.

3 Preliminary

3.1 Task Definition

A knowledge graph (KG) is a directed multi-relational graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ with entity set $\mathcal{E}$, relation set $\mathcal{R}$, and observed triple set $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Each triple $(h, r, t) \in \mathcal{T}$ is a directed edge from head entity h to tail entity t under relation r. We study link prediction: given an incomplete query $(h, r, ?)$, predict the missing tail entity t. A model assigns a plausibility score $s(h, r, t)$ to each candidate $t \in \mathcal{E}$ and ranks candidates accordingly. We distinguish the visible support triples used for graph encoding from the held-out triples to be predicted. Our benchmarks cover three split settings: *transductive*, with shared entity and relation vocabularies; *inductive*, with new test entities; and *fully inductive*, with new test entities and relations.

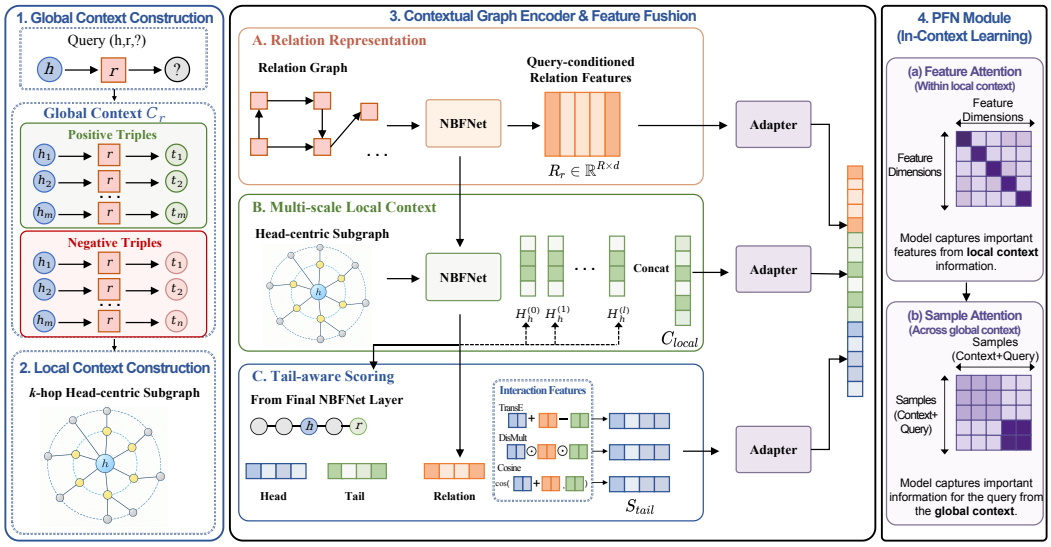


Figure 2: The framework of KGPFN.

3.2 Prior-data Fitted Networks

Posterior predictive distribution. In the Bayesian view of supervised learning, a prior $p(\phi)$ is placed over a hypothesis space Φ of mappings from inputs x to labels y. Each hypothesis $\phi \in \Phi$ induces a data-generating distribution from which a dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$ is sampled. Given a context set D and a test input x, the posterior predictive distribution marginalizes over hypotheses:

$$p(y \mid x, D) \;\propto\; \int_{\Phi} p(y \mid x, \phi)\, p(D \mid \phi)\, p(\phi)\, d\phi, \tag{1}$$

where $p(D \mid \phi)$ is the likelihood of D under ϕ.

Prior-data fitted networks. PFNs cast in-context learning as amortized Bayesian inference: a single network is trained to approximate the posterior predictive distribution given labeled context examples. Given a support set $\mathcal{C} = \{(x_c^{(i)}, y_c^{(i)})\}_{i=1}^{n}$ and a query x_q, a PFN predicts

$$f_\theta(x_q, \mathcal{C}) \;\approx\; p(y_q \mid x_q, \mathcal{C}). \tag{2}$$

The prior over tasks is encoded implicitly in the pretrained weights, and conditioning on the observed context $\mathcal{C}$ yields an approximate posterior predictive distribution in a single forward pass, without parameter updates.

4 Method

This section presents KGPFN, a knowledge graph foundation model with in-context learning capabilities; its framework is shown in Fig. 2. Sections 4.1 and 4.2 construct the global and local context, Section 4.3 encodes them, and Section 4.4 describes the PFN, which combines feature-level and sample-level attention.

4.1 Global Context Construction

To show the model how the query relation is typically grounded, we construct a relation-specific global context. Let $\mathcal{G}_{\text{obs}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_{\text{obs}})$ denote the graph of visible support edges under the evaluation protocol; context construction and graph encoding use only this graph, so held-out query edges are excluded. For a query $(h, r, ?)$, we collect observed triples of the same relation, $\{(h_i, r, t_i)\}_{i=1}^{m^+}$, from $\mathcal{T}_{\text{obs}}$, and generate negative triples of that relation, $\{(h_j, r, t_j)\}_{j=1}^{m^-}$, by negative sampling (Appendix C.2). Together they form the global context $\mathcal{C}_r = \{((h_k, r, t_k), y_k)\}_{k=1}^{m}$, where $m = m^+ + m^-$ and $y_k \in \{0, 1\}$ labels each example as positive or negative.

4.2 LOCAL CONTEXT CONSTRUCTION

Whereas global context provides relation-level evidence, local context captures the neighborhood structure around each query or context triple. Extracting an enclosing subgraph around both endpoints of every candidate triple would be expensive, so we use a head-centric local context shared by all candidate tails of a given (h, r). Conceptually, this context is the k-hop neighborhood of h in $\mathcal{G}_{\text{obs}}$. We obtain its representation through head-conditioned NBFNet message passing (Section 4.3): because each layer aggregates evidence from one additional hop, the head states of the first k layers provide multi-scale local-context features. The same graph-wide forward pass also yields final-layer representations of all candidate tails, so local-context extraction and all-entity scoring share their graph computation.

4.3 CONTEXTUAL GRAPH ENCODER

We next encode both kinds of context into vector representations.

Relation representations. Following Galkin et al. (2023) and Zhang et al. (2025b), we build a relation graph $\mathcal{G}_r = (\mathcal{R}, \mathcal{R}_{\text{fund}}, \mathcal{E}_r)$ from $\mathcal{G}_{\text{obs}}$. Each node is a relation type in $\mathcal{R}$, and edges in $\mathcal{E}_r$ connect relations that share endpoints, typed by four fundamental interactions $\mathcal{R}_{\text{fund}} = \{\text{t2h}, \text{h2h}, \text{h2t}, \text{t2t}\}$. Equivalently, the relation graph is an adjacency tensor $\mathbf{A}_r \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{R}|\times 4}$.

Given a query relation r, we initialize its node with an indicator ($\mathbf{h}^{(0)}_{r|r} = \mathbf{1}^d$ and $\mathbf{h}^{(0)}_{v|r} = \mathbf{0}$ for $v \neq r$) and run message passing on $\mathcal{G}_r$, where each layer aggregates messages over the four interaction types $s \in \mathcal{R}_{\text{fund}}$:

$$\mathbf{h}^{(\ell+1)}_{v|r} \;=\; \text{Upd}\Big(\mathbf{h}^{(\ell)}_{v|r},\;\; \text{Agg}\Big(\big\{\text{Msg}_s(\mathbf{h}^{(\ell)}_{u|r}) : u \in \mathcal{N}_s(v),\; s \in \mathcal{R}_{\text{fund}}\big\}\Big)\Big). \tag{3}$$

After L_r layers, we obtain the query-conditioned relation embeddings $\mathbf{R}_r = \big[\mathbf{h}^{(L_r)}_{v|r}\big]_{v\in\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|\times d}$.

Multi-scale local context representation. We run an L-layer NBFNet on $\mathcal{G}_{\text{obs}}$ conditioned on (h, r), initializing the head node with the query relation embedding and all other nodes with zeros: $\mathbf{H}^{(0)}_h = \mathbf{r}$ and $\mathbf{H}^{(0)}_v = \mathbf{0}$ for $v \neq h$. Writing $\mathbf{H}^{(\ell)}_v$ for the state of node v at layer ℓ (with the conditioning on (h, r) left implicit), each layer propagates relation-aware messages along visible typed edges:

$$\mathbf{H}^{(\ell+1)}_v \;=\; \text{Upd}\Big(\mathbf{H}^{(\ell)}_v,\;\; \text{Agg}\Big(\big\{\text{Msg}(\mathbf{H}^{(\ell)}_u, r_{u\to v}, \mathbf{r}) : (u, r_{u\to v}, v) \in \mathcal{T}_{\text{obs}}\big\}\Big)\Big). \tag{4}$$

Given the relation representations, each layer extends the receptive field by one hop (Lemma 2). We therefore use the head states as local-context summaries, $\mathbf{c}^{(\ell)}_{\text{loc}}(h, r) = \mathbf{H}^{(\ell)}_h$, and stack them as $\mathbf{C}_{\text{loc}}(h, r) = [\mathbf{c}^{(1)}_{\text{loc}}; \cdots; \mathbf{c}^{(k)}_{\text{loc}}]$ with $1 \leq k \leq L$.

Tail-aware scoring representation. Let $\mathbf{z}^{(L)}_v = \mathbf{H}^{(L)}_v$. To discriminate among candidate tails, we combine the final-layer tail embedding $\mathbf{z}^{(L)}_t$, the head embedding $\mathbf{z}^{(L)}_h$, and the query relation embedding $\mathbf{r}$ into three interaction features:

$$\begin{aligned} \phi_{\text{transe}}(h, r, t) &= \mathbf{z}^{(L)}_h + \mathbf{r} - \mathbf{z}^{(L)}_t, \\ \phi_{\text{distmult}}(h, r, t) &= \mathbf{z}^{(L)}_h \odot \mathbf{r} \odot \mathbf{z}^{(L)}_t, \\ \phi_{\cos}(h, r, t) &= \cos\big(\mathbf{z}^{(L)}_h + \mathbf{r},\; \mathbf{z}^{(L)}_t\big), \end{aligned} \tag{5}$$

which are concatenated as $S_{\text{tail}}(h, r, t) = [\phi_{\text{transe}}(h, r, t), \phi_{\text{distmult}}(h, r, t), \phi_{\cos}(h, r, t)]$. For the query and each context example $((h_k, r, t_k), y_k) \in \mathcal{C}_r$, three adapters project the local context, the triple embedding, and S_{tail} into a shared space, and their outputs are concatenated:

$$\mathbf{x}_k = \Big[\text{Adapter}\big(\mathbf{C}_{\text{loc}}(h_k, r)\big)\;;\; \text{Adapter}\big([\mathbf{z}^{(L)}_{h_k}; \mathbf{r}; \mathbf{z}^{(L)}_{t_k}]\big)\;;\; \text{Adapter}\big(S_{\text{tail}}(h_k, r, t_k)\big)\Big], \tag{6}$$

where each $\text{Adapter}(\cdot)$ is a lightweight MLP.

4.4 PFN ARCHITECTURE

Given the fused representations in Eq. 6, the PFN (Eq. 2) scores a query $\mathbf{x}_q$ conditioned on the labeled context $\mathcal{C}_r = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $y_i \in \{0, 1\}$ indicates whether context triple i is positive. Feature-level self-attention, with RoPE (Su et al., 2024) positional encoding, lets the features of each example interact. Sample-level attention then aggregates the context: context examples attend to one another through self-attention, and each candidate query retrieves relation-specific evidence through cross-attention to $\mathcal{C}_r$. Keys and values come only from context examples, while residual connections preserve each candidate's own representation. The contextualized query representation is mapped to the score $s(h, r, t)$. Appendix B details the formulation, inference complexity, and caching strategy.

Data prior construction. Tabular PFNs (Hollmann et al., 2022; Grinsztajn et al., 2025; Zhang et al., 2025a) often synthesize tasks from random generative processes. We instead build an empirical task prior by sampling support–query episodes from the training KGs, which preserves their multi-relational structure. Each episode consists of a support graph $\mathcal{G}_\text{s}$ and a query graph $\mathcal{G}_\text{q}$ drawn from the remaining training triples whose endpoints appear in $\mathcal{G}_\text{s}$. The support graph plays the role of $\mathcal{G}_\text{obs}$ for context construction and graph encoding, and the held-out query triples serve as prediction targets. To sample $\mathcal{G}_\text{s}$, we draw a base keep ratio ρ shared across relations, add a small relation-specific perturbation δ, and retain each relation's edges at the resulting rate. Training over many such episodes yields a predictor that transfers across graphs by conditioning on labeled graph context.

Training objective. We train KGPFN with two complementary losses under negative sampling. For a batch of B positive triples with scores s_i^+, each paired with N negatives with scores $s_{i,j}^-$, a binary cross-entropy (BCE) loss separates positives from negatives:

$$\mathcal{L}_\text{BCE} = -\frac{1}{B}\sum_{i=1}^{B}\left[\log\sigma\big(s_i^+\big) + \sum_{j=1}^{N} w_{i,j}\,\log\big(1-\sigma\big(s_{i,j}^-\big)\big)\right], \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid function and $w_{i,j} = \frac{\exp(s_{i,j}^-/\tau)}{\sum_{k=1}^N \exp(s_{i,k}^-/\tau)}$ is a self-adversarial weight with temperature τ that emphasizes hard negatives. A softmax cross-entropy loss further encourages each positive to rank above its sampled negatives:

$$\mathcal{L}_\text{CE} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp\big(s_i^+\big)}{\exp\big(s_i^+\big)+\sum_{j=1}^{N}\exp\big(s_{i,j}^-\big)}. \tag{8}$$

The overall objective is $\mathcal{L} = \mathcal{L}_\text{BCE} + \mathcal{L}_\text{CE}$.

An idealized view of context-dependent scoring. To illustrate how labeled context can modulate structural scoring without parameter updates, we analyze an idealized bilinear surrogate of the context interaction.

Proposition 1 (Context-dependent feature reweighting in a bilinear surrogate)**.** *Fix a feature map $\mathbf{c} : \mathcal{X} \to \mathbb{R}^n$ on triples, a bias $b \in \mathbb{R}$, weights $\mathbf{w}_0 \in \mathbb{R}^n$, and a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. For a query $q \in \mathcal{X}$ and a labeled support set $\mathcal{S}_r = \{(s_i, y_i)\}_{i=1}^K$, with $K \geq 1$, $y_i \in \{0, 1\}$, and $\widetilde{y}_i = 2y_i - 1$, define*

$$s_\text{bil}(q \mid \mathcal{S}_r) := b + \mathbf{w}_0^\top \mathbf{c}(q) + \frac{1}{K}\sum_{i=1}^{K}\widetilde{y}_i\,\mathbf{c}(q)^\top\mathbf{A}\mathbf{c}(s_i). \tag{9}$$

Writing $\boldsymbol{\mu}_{\mathcal{S}_r} := K^{-1}\sum_{i=1}^K \widetilde{y}_i\mathbf{c}(s_i)$, the score satisfies

$$s_\text{bil}(q \mid \mathcal{S}_r) = b + \big(\mathbf{w}_0 + \mathbf{A}\boldsymbol{\mu}_{\mathcal{S}_r}\big)^\top\mathbf{c}(q). \tag{10}$$

Proposition 1 shows that labeled support examples shift the effective structural weights by $\mathbf{A}\boldsymbol{\mu}_{\mathcal{S}_r}$, while the feature map and model parameters stay fixed. A diagonal $\mathbf{A}$ reweights features coordinate-wise, a general $\mathbf{A}$ also couples different query and support features, and $\mathbf{A} = \mathbf{0}$ recovers a context-independent structural scorer. Appendix A gives the proof and the exact condition under which changing the support set changes a query score.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets and metrics. Following prior work (Galkin et al., 2023; Zhang et al., 2025b), we evaluate on 57 knowledge graphs from diverse domains, grouped by their benchmark splits: *transductive* (16 datasets), where training and test graphs share entity and relation vocabularies; *inductive* (18 datasets), with new test entities; and *fully inductive* (23 datasets), with new test entities and relations. These categories describe each benchmark's own training/test split; whether pretrained parameters are further fine-tuned is determined by the evaluation protocols below. We use filtered ranking metrics: for each test triple (h, r, t), the tail is ranked against all candidate entities after filtering out other true triples. For KGPFN, we report mean MRR and Hits@10 over five inference runs with different context-sampling seeds and a fixed checkpoint per dataset and setting; per-dataset standard deviations are given in Appendix C.3. Benchmark averages weight each graph equally. Dataset statistics are listed in Appendix C.1.

Baselines. We compare KGPFN with five KG foundation models: **ULTRA** (Galkin et al., 2023), which pretrains transferable relation representations on relation graphs for inductive reasoning across graphs; **KGICL** (Cui et al., 2024), which performs in-context reasoning over retrieved prompt graphs; **TRIX** (Zhang et al., 2025b), a more expressive fully inductive model for zero-shot transfer; **MOTIF** (Huang et al., 2025), which captures higher-order relational motifs beyond pairwise relation interactions; and **FLOCK** (Kim et al., 2026), which learns from random walks.

Implementation details. Following Galkin et al. (2023), the relation encoder and the local-context encoder are each 6-layer NBFNets with hidden dimension 64. The PFN Transformer is initialized from pretrained TabICL (Qu et al., 2026) weights, with its feature preprocessing module removed. Local context is read from the head-node states of the first three NBFNet layers, and the sixth layer provides representations of all candidate tails in the visible support graph. The global context contains 20 positive and 80 negative triples. We train on 8 NVIDIA A800 80GB GPUs and run inference on a single A800 GPU. Further training details are given in Appendix C.2.

5.2 EVALUATION WITHOUT AND WITH FINE-TUNING

Following prior KG foundation models (Galkin et al., 2023; Zhang et al., 2025b; Huang et al., 2025), we pretrain KGPFN on FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), and CoDEx-Medium (Safavi & Koutra, 2020), and evaluate under two protocols. Without fine-tuning, the pretrained model retrieves positive and negative context examples from each target graph's visible support edges and predicts with fixed parameters. This measures zero-shot transfer on the 54 graphs outside the pretraining set and held-out link prediction on the three pretraining graphs. With fine-tuning, the model is further trained on the target dataset's training split. Because KGICL uses a different pretraining setup, we include it only in the fine-tuned comparison.

Table 1 summarizes the average results. Without fine-tuning, KGPFN achieves the best MRR in all three settings and overall, and ties with FLOCK for the best overall Hits@10. Its results in the two inductive settings show that the pretrained in-context predictor transfers to graphs with unseen entities and relations. With fine-tuning, KGPFN achieves the best average MRR and Hits@10 in every setting, raising overall MRR from 0.396 to 0.434. Fine-tuning brings large gains on several graphs whose domains differ markedly from the pretraining graphs, such as ConceptNet100k (0.116 to 0.291), AristoV4 (0.205 to 0.342), and Hetionet (0.276 to 0.405). Per-dataset results are reported in Appendix C.3, and inference efficiency is compared in Appendix B.2.

5.3 CONTEXT SENSITIVITY ANALYSIS

We study how global-context size and local-context depth affect performance, retraining the model for each configuration. Since the PFN predicts from a nonempty labeled support set, we vary the numbers of positive and negative examples rather than removing global context entirely.

Global context. We vary the number of positive examples over $\{5, 10, 15, 20\}$ and the number of negative examples over $\{20, 40, 60, 80\}$, evaluate all 16 combinations, and report the average MRR over the 57 graphs (Fig. 3, left). Within this range, performance is relatively insensitive to

Table 1: Average MRR and Hits@10 over 57 KGs. The 16-graph transductive and 57-graph overall averages include the three pretraining graphs (FB15k-237, WN18RR, and CoDEx-Medium); the "Pretraining" columns report this subset separately. W/o FT: without fine-tuning. (**Bold**: best; Underline: runner-up.)

Setting	Model	Inductive e, r (23 graphs)		Inductive e (18 graphs)		Transductive (16 graphs)		Total Avg (57 graphs)		Pretraining (3 graphs)	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
W/o FT	ULTRA	0.345	0.513	0.431	0.566	0.329	0.479	0.366	0.518	–	–
	MOTIF	0.349	0.525	0.436	0.577	0.332	0.476	0.372	0.528	–	–
	TRIX	0.368	0.540	0.455	0.592	0.353	0.512	0.391	0.551	–	–
	FLOCK	0.369	0.554	0.456	**0.604**	0.352	0.518	0.392	**0.560**	–	–
	KGPFN	**0.373**	**0.558**	**0.460**	0.597	**0.355**	**0.521**	**0.396**	**0.560**	–	–
Fine-tuned	ULTRA	0.397	0.556	0.442	0.582	0.384	0.548	0.408	0.562	0.407	0.568
	KGICL	0.410	0.560	0.467	0.604	0.397	0.554	0.419	0.576	–	–
	MOTIF	0.401	0.558	0.455	0.594	0.394	0.552	0.417	0.569	0.415	0.565
	TRIX	0.401	0.556	0.459	0.594	0.395	0.559	0.418	0.569	0.415	0.563
	FLOCK	0.417	0.576	0.473	0.619	0.389	0.547	0.427	0.582	0.415	0.561
	KGPFN	**0.421**	**0.590**	**0.482**	**0.623**	**0.398**	**0.567**	**0.434**	**0.594**	**0.416**	**0.580**

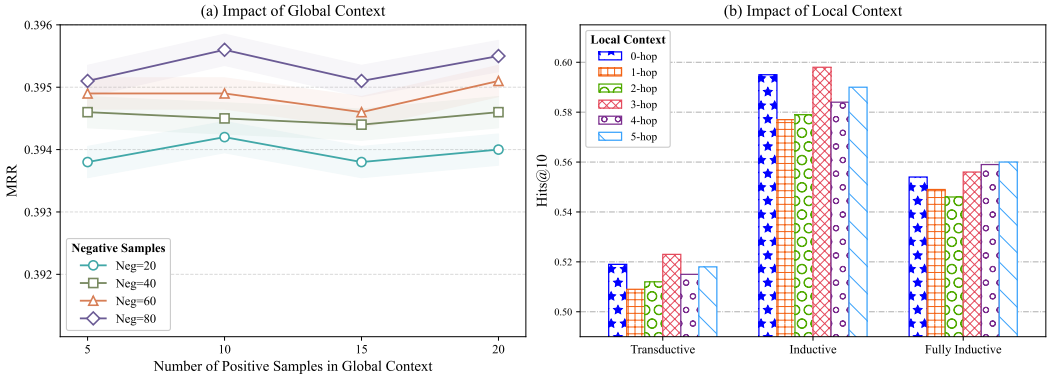


Figure 3: Context sensitivity. (a) Average MRR over 57 KGs for different numbers of positive and negative global-context examples. (b) Hits@10 for different local-context depths.

the number of positives, whereas adding negatives consistently improves MRR. This suggests that negative examples provide useful evidence for discriminating among candidates. A likely reason is that about 80% of negatives are drawn from the head's two-hop neighborhood (Appendix C.2), so they act as hard, structurally similar contrasts to the true tails.

Local context. We vary the local-context depth from 0 to 5 hops (Fig. 3, right); depth 0 removes the local-context input $\mathbf{C}_{\text{loc}}$ and its adapter branch from Eq. 6. Three-hop context gives the best Hits@10 in the transductive and inductive settings and remains competitive in the fully inductive setting. With only 1 or 2 hops, performance can even fall below the variant without local context, suggesting that very shallow summaries are insufficient for disambiguation. Going beyond 3 hops brings no consistent further gain.

5.4 Ablation Study

To assess each tail-aware scoring feature, we remove the TransE, DistMult, and cosine features one at a time (Table 2). Removing any single feature lowers MRR in all three settings, indicating that the features are complementary. Removing the TransE-style feature causes the largest drop, from 0.396 to 0.389 in average MRR.

Table 2: Ablation of tail-aware scoring features (MRR). (**Bold**: best.)

Model	Inductive e, r (23 graphs)	Inductive e (18 graphs)	Transductive (16 graphs)	Average (57 graphs)
KGPFN	**0.373**	**0.460**	**0.355**	**0.396**
KGPFN (w/o TransE score)	0.365	0.454	0.351	0.389
KGPFN (w/o DistMult score)	0.368	0.458	0.353	0.392
KGPFN (w/o cosine score)	0.370	0.456	0.352	0.392

5.5 Inference Efficiency

We measure per-query latency and peak GPU memory for all-entity ranking on FB15k-237 (Table 3, Appendix B.2). KGPFN takes 62.1 ms per query with 2.78 GB of peak memory. Its latency matches that of KGICL (61.8 ms), which also builds query-specific context, while using about $9\times$ less memory (24.9 GB), and it is about $8\times$ faster than FLOCK (491 ms, 50.9 GB). Direct structural scorers remain faster (0.95–18.7 ms for ULTRA, TRIX, and MOTIF) because they skip context retrieval and encoding. KGPFN keeps this overhead moderate by sharing one graph-encoder pass across all candidate tails and caching the encoded support set across candidate chunks.

5.6 Visualization

To illustrate how the model uses labeled global context, we present a case study on CoDEx-Large (Safavi & Koutra, 2020) with the query ($\texttt{Imperial}, \texttt{Record label}^{-1}$, ?), where $^{-1}$ denotes the inverse relation. The candidate set contains 5 correct and 20 incorrect tails, and the global context contains 5 positive and 20 negative examples. Fig. 4 shows the cross-attention weights from the last sample-attention layer, i.e., how each query candidate attends to the context examples. Since candidate rows contribute no keys or values, these weights show directly which labeled examples each candidate draws on. Correct candidates concentrate their attention on positive context examples, whereas incorrect candidates spread their attention more diffusely over negative examples. The case study thus illustrates that attention to labeled global context is candidate-specific.

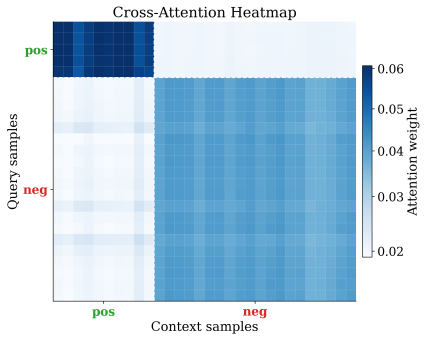


Figure 4: Last-layer sample attention from query candidates to context examples.

6 Conclusion

Knowledge graph foundation models have mainly pursued transferable relational patterns, while in-context learning has received less attention. Effective KG reasoning draws on two kinds of context: the local neighborhood of the query, and a global view of how the query relation behaves across many instances. KGPFN combines both. It learns relation representations by message passing on the relation graph, reads multi-scale local context from intermediate NBFNet head states, and aggregates positive and negative examples of the query relation with a Prior-Data Fitted Network that uses feature-level and sample-level attention. Labeled contextual evidence thus informs prediction at inference time without parameter updates. On 57 knowledge graphs spanning transductive, inductive, and fully inductive settings, KGPFN performs strongly both without and with fine-tuning, and the context analyses highlight the value of negative examples and three-hop local context. Retrieving and encoding context makes inference slower than direct structural scorers such as ULTRA (Section 5.5); reducing this cost is a natural direction for future work. Other promising extensions include selecting context examples adaptively for each query instead of sampling them, and applying the same in-context formulation to complex logical queries and to temporal or multimodal knowledge graphs.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.

Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. Query2particles: Knowledge graph reasoning with particle embeddings. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 2703–2714, 2022.

Jiaxin Bai, Xin Liu, Weiqi Wang, Chen Luo, and Yangqiu Song. Complex query answering on eventuality knowledge graph with implicit logical constraints. *Advances in Neural Information Processing Systems*, 36:30534–30553, 2023.

Jiaxin Bai, Zihao Wang, Yukun Zhou, Hang Yin, Weizhi Fei, Qi Hu, Zheye Deng, Jiayang Cheng, Tianshi Zheng, Hong Ting Tsang, et al. Top ten challenges towards agentic neural graph databases. *arXiv preprint arXiv:2501.14224*, 2025.

Maciej Besta, Patrick Iff, Florian Scheidl, Kazuki Osawa, Nikoli Dryden, Michal Podstawski, Tiancheng Chen, and Torsten Hoefler. Neural graph databases. In *Learning on Graphs Conference*, pp. 31–1. PMLR, 2022.

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.

Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pp. 151–161, 2019.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. Meta relational learning for few-shot link prediction in knowledge graphs. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4217–4226, 2019.

Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. *Advances in Neural Information Processing Systems*, 37:7095–7124, 2024.

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.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. Graphpfn: A prior-data fitted graph foundation model. *arXiv preprint arXiv:2509.21489*, 2025.

Mikhail Galkin, Max Berrendorf, and Charles Tapley Hoyt. An open challenge for inductive link prediction on knowledge graphs. *arXiv preprint arXiv:2203.01520*, 2022.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. *arXiv preprint arXiv:2310.04562*, 2023.

Yisen Gao, Jiaxin Bai, Tianshi Zheng, Qingyun Sun, Ziwei Zhang, Xingcheng Fu, Jianxin Li, and Yangqiu Song. Controllable logical hypothesis generation for abductive reasoning in knowledge graphs. *arXiv preprint arXiv:2505.20948*, 2025.

Yisen Gao, Jiaxin Bai, Yi Huang, Xingcheng Fu, Qingyun Sun, and Yangqiu Song. Unifying deductive and abductive reasoning in knowledge graphs with masked diffusion model. In *Proceedings of the ACM Web Conference 2026*, pp. 3600–3611, 2026.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. Tabpfn-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*, 2017.

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.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Xingyue Huang, Pablo Barceló, Michael M Bronstein, Ismail Ilkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? *arXiv preprint arXiv:2502.13339*, 2025.

Jinwoo Kim, Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and Ismail I Ceylan. Flock: A knowledge graph foundation model via learning on random walks. In *International Conference on Learning Representations*, volume 2026, pp. 36664–36705, 2026.

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.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

Yufei Li, Yisen Gao, Jiaxin Bai, Jiaxuan Xiong, Haoyu Huang, Zhongwei Xie, Hong Ting Tsang, and Yangqiu Song. Towards neural graph data management. *arXiv preprint arXiv:2603.05529*, 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.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 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.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, pp. 25660–25676. PMLR, 2023.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Jingang Qu, David HolzmÃžller, GaÃl Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*, 2020.

Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. Neural graph reasoning: Complex logical query answering meets graph databases. *arXiv preprint arXiv:2303.14617*, 2023.

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.

Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

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

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, 2007.

Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Timepfn: Effective multivariate time series forecasting with synthetic data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 20761–20769, 2025.

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

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yanbo Wang, Jiaxuan You, Chuan Shi, and Muhan Zhang. Relational in-context learning via synthetic pre-training with structural prior. *arXiv preprint arXiv:2603.03805*, 2026.

Zhongwei Xie, Jiaxin Bai, Shujie Liu, Haoyu Huang, Yufei Li, Yisen Gao, Hong Ting Tsang, and Yangqiu Song. Ngdb-zoo: Towards efficient and scalable neural graph databases training. *arXiv preprint arXiv:2602.21597*, 2026.

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*, 2025a.

Yucheng Zhang, Beatrice Bevilacqua, Mikhail Galkin, and Bruno Ribeiro. Trix: A more expressive model for zero-shot domain transfer in knowledge graphs. *arXiv preprint arXiv:2502.19512*, 2025b.

Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. An ood multi-task perspective for link prediction with new relation types and nodes. *arXiv preprint arXiv:2307.06046*, 23, 2023.

A Theoretical Analysis of an Idealized Context Interaction

This appendix establishes conditional locality of entity-level message passing and analyzes context-dependent feature reweighting in an idealized bilinear surrogate.

A.1 Finite-hop Structural Locality

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ be the graph on which entity-level message passing is performed. Write $u \xrightarrow{\rho} v$ when an edge of type ρ carries a message from u to v. Define $\mathcal{N}_L(v)$ to contain all nodes that can reach v in at most L such message-passing steps, including v, together with the induced typed edges.

Lemma 2 (Finite-hop Locality of Conditional Message Passing)**.** *Fix the model parameters and all conditioning inputs $\boldsymbol{\eta}$, including query-conditioned relation embeddings. Suppose the entity updates have the form*

$$\mathbf{h}_v^{(\ell+1)} = F_\ell\Big(\mathbf{h}_v^{(\ell)},\ \{\{(\rho, \mathbf{h}_u^{(\ell)}) : u \xrightarrow{\rho} v\}\}; \boldsymbol{\eta}\Big), \tag{11}$$

where F_ℓ is a deterministic function of the displayed arguments and the braces denote a multiset. Conditional on $\boldsymbol{\eta}$, $\mathbf{h}_v^{(L)}$ is determined by the initial node features and typed edges in $\mathcal{N}_L(v)$.

Proof. We prove the claim by induction on the layer index. At layer 0, the state of v is its given initial feature, contained in $\mathcal{N}_0(v)$. Suppose the claim holds at layer ℓ for every node. Equation 11 uses only the current state of v, the types of its incoming message edges, and the current states of the corresponding neighbors u. By the induction hypothesis, these states are determined by $\mathcal{N}_\ell(v)$ and the neighborhoods $\mathcal{N}_\ell(u)$. Every node that can reach such a neighbor u in at most ℓ steps can reach v in at most $\ell + 1$ steps. Thus all required nodes, typed edges, and initial features lie in $\mathcal{N}_{\ell+1}(v)$. Since F_ℓ and $\boldsymbol{\eta}$ are fixed, they determine $\mathbf{h}_v^{(\ell+1)}$, completing the induction. □

Scope of locality. The lemma applies to the local entity-update form in Eq. 11, conditional on the initial features and relation representations. In KGPFN, the latter are computed from a graph-wide relation graph. Thus the intermediate head states combine local entity-level propagation with graph-wide relational conditioning.

A.2 Bilinear Surrogate and Proof of Feature Reweighting

For Proposition 1, let $\mathcal{X}$ be a domain of triples and fix a feature map $\mathbf{c} : \mathcal{X} \to \mathbb{R}^n$. The observed encoding graph and relation-conditioning inputs are held fixed when comparing support sets. This isolates the effect of context aggregation on a fixed structural representation.

Fix $b \in \mathbb{R}$, $\mathbf{w}_0 \in \mathbb{R}^n$, and $\mathbf{A} \in \mathbb{R}^{n \times n}$. For a nonempty support set $\mathcal{S} = \{(s_i, y_i)\}_{i=1}^K$, let $\widetilde{y}_i = 2y_i - 1$ and

$$\boldsymbol{\mu}_\mathcal{S} := \frac{1}{K} \sum_{i=1}^K \widetilde{y}_i \mathbf{c}(s_i). \tag{12}$$

The score s_{bil} is defined by Eq. 9. No symmetry, positive-semidefiniteness, or diagonal constraint on $\mathbf{A}$ is required.

Proof of Proposition 1. By linearity of matrix multiplication and finite summation,

$$\begin{aligned} \frac{1}{K} \sum_{i=1}^K \widetilde{y}_i\, \mathbf{c}(q)^\top \mathbf{A} \mathbf{c}(s_i) &= \mathbf{c}(q)^\top \mathbf{A} \left(\frac{1}{K} \sum_{i=1}^K \widetilde{y}_i \mathbf{c}(s_i) \right) \\ &= \mathbf{c}(q)^\top \mathbf{A} \boldsymbol{\mu}_\mathcal{S} \\ &= (\mathbf{A} \boldsymbol{\mu}_\mathcal{S})^\top \mathbf{c}(q). \end{aligned} \tag{13}$$

The last equality holds because both sides are the same real scalar; it does not require $\mathbf{A}$ to be symmetric. Adding $b + \mathbf{w}_0^\top \mathbf{c}(q)$ yields Eq. 10. □

When does context change the score? For any two nonempty support sets $\mathcal{S}_1, \mathcal{S}_2$, which may have different sizes, subtraction gives the exact identity

$$s_{\text{bil}}(q \mid \mathcal{S}_1) - s_{\text{bil}}(q \mid \mathcal{S}_2) = \mathbf{c}(q)^\top \mathbf{A} \big(\boldsymbol{\mu}_{\mathcal{S}_1} - \boldsymbol{\mu}_{\mathcal{S}_2} \big). \tag{14}$$

Thus the query score changes if and only if the right-hand side is nonzero: the change in effective weights must have a nonzero projection onto the query feature vector.

Coordinate-wise and cross-feature effects. The effective coefficient of query feature $c_k(q)$ is

$$w_{\mathrm{eff},k}(\mathcal{S}) = w_{0,k} + \sum_{j=1}^{n} A_{kj} \left(\frac{1}{K} \sum_{i=1}^{K} \widetilde{y}_i c_j(s_i) \right). \tag{15}$$

For $\mathbf{A} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, only the matching support coordinate contributes to each correction. A general matrix additionally permits cross-feature contributions. The statistic $\boldsymbol{\mu}_{\mathcal{S}}$ is a signed empirical feature average that reflects both the support features and their positive/negative class proportions.

Relationship to projected interactions. A projected dot product is one instance of the bilinear interaction. If $\mathbf{z}_x = \mathbf{W}_{\mathcal{M}}\mathbf{c}(x)$, where $\mathbf{W}_{\mathcal{M}} \in \mathbb{R}^{d\times n}$, and $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{a\times d}$, then

$$(\mathbf{W}_Q\mathbf{z}_q)^\top(\mathbf{W}_K\mathbf{z}_s) = \mathbf{c}(q)^\top \underbrace{\mathbf{W}_{\mathcal{M}}^\top\mathbf{W}_Q^\top\mathbf{W}_K\mathbf{W}_{\mathcal{M}}}_{\mathbf{A}} \mathbf{c}(s). \tag{16}$$

This identity motivates the surrogate through projected query–support interactions. Proposition 1 describes the resulting feature reweighting under label-weighted bilinear aggregation. KGPFN implements context interaction using the multi-layer attention architecture detailed below.

B Detailed Formulations of PFN Architecture

In our framework, the PFN acts as an in-context relational inference engine. Given an in-context support set $\mathcal{C}_r = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ and a set of query candidate triples $\{\mathbf{x}_{q,j}\}_{j=1}^N$ conditioned on relation r, the PFN estimates the plausibility score $s(h, r, t)$. Following the design principles of TabICL (Qu et al., 2025; 2026), we decouple the processing into two orthogonal Transformer modules: **Feature-Level Attention** (row-wise feature interaction) and **Sample-Level Attention** (dataset-wise in-context learning).

Feature-Level Attention (Row-wise Feature Interaction). For each triple k (whether an in-context triple from $\mathcal{C}_r$ or a query candidate), the multi-source fusion in Eq. 6 produces a structured set of F feature tokens:

$$\mathbf{X}_k = [\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \ldots, \mathbf{x}_{k,F}] \in \mathbb{R}^{F\times d_{\mathrm{feat}}}, \tag{17}$$

which encapsulates the multi-hop local context $\mathbf{C}_{\mathrm{loc}}(h_k, r)$, the conditional entity-relation representations $[\mathbf{z}_{h_k}^{(L)}; \mathbf{r}; \mathbf{z}_{t_k}^{(L)}]$, and the structural plausibility features $S_{\mathrm{tail}}(h_k, r, t_k)$.

Following TabICL (Qu et al., 2025; 2026), we first project the feature channels into the internal embedding dimension d_e via a linear adapter $\mathbf{W}_{\mathrm{adapt}} \in \mathbb{R}^{d_{\mathrm{feat}}\times d_e}$. To aggregate disparate topological signals into a compact sample-level representation, we prepend C learnable class tokens $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_C]^\top \in \mathbb{R}^{C\times d_e}$ (C=4 by default) to the feature sequence:

$$\mathbf{Z}_k^{(0)} = \big[\mathbf{C}\ ;\ \mathbf{X}_k\mathbf{W}_{\mathrm{adapt}}\big] \in \mathbb{R}^{(C+F)\times d_e}. \tag{18}$$

The sequence $\mathbf{Z}_k^{(0)}$ is then processed by a stack of L_{feat} Transformer encoder layers equipped with Rotary Position Embedding (RoPE) (Su et al., 2024). At each layer $l \in \{1, \ldots, L_{\mathrm{feat}}\}$, the multi-head self-attention operation is defined as:

$$\mathbf{Q}_k^{(l)} = \mathbf{R}_\Theta\big(\mathbf{Z}_k^{(l-1)}\mathbf{W}_Q^{(l)}\big), \quad \mathbf{K}_k^{(l)} = \mathbf{R}_\Theta\big(\mathbf{Z}_k^{(l-1)}\mathbf{W}_K^{(l)}\big), \quad \mathbf{V}_k^{(l)} = \mathbf{Z}_k^{(l-1)}\mathbf{W}_V^{(l)}, \tag{19}$$

$$\mathbf{A}_{k,\mathrm{feat}}^{(l)} = \mathrm{Softmax}\left(\frac{\mathbf{Q}_k^{(l)}(\mathbf{K}_k^{(l)})^\top}{\sqrt{d_k}}\right)\mathbf{V}_k^{(l)}, \tag{20}$$

followed by standard residual connections, LayerNorm, and feed-forward networks:

$$\mathbf{Z}_k^{(l)} = \mathbf{Z}_k^{(l-1)} + \mathrm{Dropout}\big(\mathbf{A}_{k,\mathrm{feat}}^{(l)}\mathbf{W}_O^{(l)}\big), \quad \mathbf{Z}_k^{(l)} \leftarrow \mathbf{Z}_k^{(l)} + \mathrm{FFN}\big(\mathrm{LayerNorm}(\mathbf{Z}_k^{(l)})\big). \tag{21}$$

In the final layer L_{feat}, the C updated class tokens act as queries attending over the full feature context. The resulting class representations $[\mathbf{z}_{k,1}^{(L_{\mathrm{feat}})}, \ldots, \mathbf{z}_{k,C}^{(L_{\mathrm{feat}})}]$ are normalized with LayerNorm and concatenated to construct the final sample-level representation vector:

$$\mathbf{u}_k = \mathrm{Concat}\Big(\mathbf{z}_{k,1}^{(L_{\mathrm{feat}})}, \mathbf{z}_{k,2}^{(L_{\mathrm{feat}})}, \ldots, \mathbf{z}_{k,C}^{(L_{\mathrm{feat}})}\Big) \in \mathbb{R}^{d_{\mathrm{icl}}}, \tag{22}$$

where $d_{\mathrm{icl}} = C \cdot d_e$. This effectively condenses the multi-faceted graph structure of each triple into a fixed-dimensional latent embedding.

Sample-Level Attention (Dataset-wise In-Context Learning). Once every triple k is mapped to $\mathbf{u}_k \in \mathbb{R}^{d_{\text{icl}}}$, the sample-level Transformer models relational dependencies between the query triple and the support context triples.

First, ground-truth binary labels $y_i \in \{0, 1\}$ for context triples are mapped into the latent space via a linear target encoder $\mathbf{W}_y \in \mathbb{R}^{d_{\text{icl}} \times 1}$ with bias $\mathbf{b}_y$ and added to the context representations, while query representations remain unaltered:

$$\tilde{\mathbf{u}}_i = \begin{cases} \mathbf{u}_i + \mathbf{W}_y y_i + \mathbf{b}_y, & \text{if } i \in \{1, \dots, m\} \quad \text{(context triple in } \mathcal{C}_r\text{)} \\ \mathbf{u}_i, & \text{if } i \in \{m+1, \dots, m+N\} \quad \text{(candidate query triple)} \end{cases} \tag{23}$$

The combined sequence is formed as:

$$\mathbf{H}^{(0)} = \left[\tilde{\mathbf{u}}_1, \dots, \tilde{\mathbf{u}}_m, \tilde{\mathbf{u}}_{q,1}, \dots, \tilde{\mathbf{u}}_{q,N}\right]^\top \in \mathbb{R}^{(m+N) \times d_{\text{icl}}}. \tag{24}$$

The sequence passes through L_{icl} Transformer layers. At each layer, attention queries are computed from all rows, while keys and values are computed only from the context rows. Thus, context examples perform bidirectional self-attention, and each candidate query performs cross-attention exclusively to the context pool. Candidate query rows contribute no keys or values, so they attend neither to themselves nor to other candidate queries. Each candidate has its own attention query vector and retrieves a candidate-specific weighted combination of the shared context values. Formally, let $\mathbf{H}_{\text{ctx}}^{(l-1)} \in \mathbb{R}^{m \times d_{\text{icl}}}$ denote the context representations and $\mathbf{h}_{q,j}^{(l-1)} \in \mathbb{R}^{d_{\text{icl}}}$ denote a query triple representation at layer $l-1$. The sample-level cross-attention is computed as:

$$\mathbf{q}_{q,j}^{(l)} = \mathbf{h}_{q,j}^{(l-1)} \mathbf{W}_{Q,\text{icl}}^{(l)}, \quad \mathbf{K}_{\text{ctx}}^{(l)} = \mathbf{H}_{\text{ctx}}^{(l-1)} \mathbf{W}_{K,\text{icl}}^{(l)}, \quad \mathbf{V}_{\text{ctx}}^{(l)} = \mathbf{H}_{\text{ctx}}^{(l-1)} \mathbf{W}_{V,\text{icl}}^{(l)}, \tag{25}$$

$$\mathbf{a}_{q,j,\text{sample}}^{(l)} = \text{Softmax}\left(\frac{\mathbf{q}_{q,j}^{(l)} (\mathbf{K}_{\text{ctx}}^{(l)})^\top}{\sqrt{d_{\text{icl}}}} \odot \mathbf{\Gamma}\big(\mathbf{q}_{q,j}^{(l)}, m\big) \right) \mathbf{V}_{\text{ctx}}^{(l)}, \tag{26}$$

where $\mathbf{\Gamma}(\mathbf{q}, m) = \text{MLP}_{\text{base}}(\log m) \cdot \big(1 + \tanh(\text{MLP}_{\text{query}}(\mathbf{q}))\big)$ represents TabICL's query-aware scalable softmax (QASSMax) modulation (Qu et al., 2025), preventing attention entropy collapse across varied context sizes m. The attention output is combined with that row's incoming representation through a residual connection, preserving candidate-specific information alongside the retrieved evidence. Layer normalization and feed-forward transformations are applied within each row. After L_{icl} layers, only the candidate query rows are passed to the scoring head. Each contextualized query representation $\mathbf{h}_{q,j}^{(L_{\text{icl}})}$ is projected by a lightweight two-layer MLP to produce the scalar plausibility prediction:

$$s(h, r, t_j) = \mathbf{w}_2^\top \, \text{GELU}\big(\mathbf{W}_1 \mathbf{h}_{q,j}^{(L_{\text{icl}})} + \mathbf{b}_1\big) + b_2 \in \mathbb{R}. \tag{27}$$

B.1 Theoretical Inference Complexity Analysis

We separate graph encoding, support preparation, and candidate scoring. Let $n = |\mathcal{V}|$ be the number of candidate entities, $m = |\mathcal{C}_r|$ the support size, and $p = C + F$ the number of feature tokens including class tokens. Write $D = d_{\text{icl}}$, and assume the feed-forward widths scale proportionally to their respective Transformer embedding widths. We likewise take the QASSMax modulation-network widths to be $\mathcal{O}(D)$, so their per-query cost is included in the $\mathcal{O}(D^2)$ dense-computation term.

Graph encoding. Let T_{rel} be the cost of preparing query-conditioned relation representations on the visible graph. Let $T_{\text{enc}}(h, r)$ be the cost of one head-conditioned entity-encoder pass, including local-context readout and final entity representations. One such pass serves all candidate tails for a given head and relation. Each context triple also requires representations conditioned on its own head. If $\mathcal{H}_r$ denotes the distinct context heads, the cold graph-encoding cost, grouping identical head–relation pairs, is

$$T_{\text{graph}} = T_{\text{rel}} + \sum_{u \in \{h\} \cup \mathcal{H}_r} T_{\text{enc}}(u, r). \tag{28}$$

This separates context graph encoding from the query pass. The exact encoder cost depends on the message and update operators; the support representations can be reused across candidate chunks once computed. This expression assumes that static graph structures and retrieval indices have already been built, matching the preprocessing convention of the empirical benchmark below.

Feature encoding and support prefill. For each triple, input projection and the feature Transformer cost

$$T_{\text{feat}} = \mathcal{O}\big(F d_{\text{feat}} d_e + L_{\text{feat}}(p^2 d_e + p d_e^2)\big)\,. \tag{29}$$

The first term is the feature projection, while the two layer terms account for attention products and dense projections/feed-forward operations. Let T_{adapt} denote the per-triple cost of constructing and projecting the structural features in Eq. 6. Preparing all support examples costs $m(T_{\text{adapt}} + T_{\text{feat}})$. Their sample-level self-attention and KV-cache preparation additionally cost

$$T_{\text{prefill}} = \mathcal{O}\big(L_{\text{icl}}(m^2 D + m D^2)\big)\,. \tag{30}$$

All-entity candidate scoring. Each candidate independently encodes its features and attends to the cached support. Across all n candidates, feature construction and encoding cost $n(T_{\text{adapt}} + T_{\text{feat}})$; sample-level attention, dense projections, and feed-forward operations cost

$$T_{\text{query}} = \mathcal{O}\big(n L_{\text{icl}}(mD + D^2) + nD d_{\text{head}}\big)\,, \tag{31}$$

where d_{head} is the hidden width of the final two-layer scoring MLP. Combining these terms gives

$$\begin{aligned} T_{\text{infer}} = {} & T_{\text{retrieve}} + T_{\text{graph}} + (m+n)(T_{\text{adapt}} + T_{\text{feat}}) \\ & + T_{\text{prefill}} + T_{\text{query}}, \end{aligned} \tag{32}$$

where T_{retrieve} accounts for retrieving and sampling the support examples. For fixed model dimensions and support size, the candidate-scoring stage scales linearly with n; graph encoding and support preparation are accounted for separately.

Acceleration via context caching and chunking. The support set is held fixed while scoring all candidates for a query. We encode its feature representations once and run support self-attention once per layer, storing the resulting keys and values. Because support states do not attend to candidate queries, these cached states are reusable across candidate chunks. Caching amortizes the support prefill over the $\lceil n/N_c \rceil$ chunks, where N_c is the candidate chunk size.

The per-layer support KV cache requires $\mathcal{O}(mD)$ storage, totaling $\mathcal{O}(L_{\text{icl}} mD)$ across layers. Dense attention weights, when materialized, occupy $\mathcal{O}(m^2)$ space during support prefill and $\mathcal{O}(N_c m)$ during candidate cross-attention for a fixed number of heads. Feature attention uses $\mathcal{O}(B_{\text{feat}} p^2)$ attention-weight storage for a feature batch of B_{feat} triples. These workspaces coexist with model weights, graph-encoder state, and projected features. The attention mask makes candidate queries independent of one another, allowing cross-attention scoring without an $n \times n$ candidate-attention matrix; chunking limits the number processed simultaneously. The end-to-end peak memory is measured below.

B.2 Empirical Efficiency Experiment

We compare the inference cost of KGPFN against ULTRA (Galkin et al., 2023), TRIX (Zhang et al., 2025b), MOTIF (Huang et al., 2025), FLOCK, and KGICL (Cui et al., 2024) on FB15k-237 (Toutanova & Chen, 2015). Tail-only all-entity ranking queries $(h, r, ?)$ are timed in groups of $B{=}64$, reporting the mean across three non-overlapping groups after one warmup group. Graph-level preprocessing (e.g., relation graphs, index building) is excluded, while per-query costs (random walks, context retrieval/encoding, scoring) are included. For method settings, KGPFN uses 20 positive and 80 negative context triples; KGICL constructs 3 hop-2 prompt graphs per tail relation; and FLOCK runs with its standard 128-walk configuration under an evaluation microbatch of 8.

Table 3: Inference efficiency comparison on FB15k-237.

Model	ms/query	Peak GPU (GB)
ULTRA	0.95	2.34
TRIX	5.00	1.46
MOTIF	18.7	2.48
KGICL	61.8	24.9
KGPFN	62.1	2.78
FLOCK	491	50.9

Table 4: 16 Transductive datasets used in the experiments.

Dataset	Entities	Rels	Train	Valid	Test
CoDEx Small	2034	42	32888	1827	1828
CoDEx Medium	17050	51	185584	10310	10311
CoDEx Large	77951	69	551193	30622	30622
WDsinger	10282	135	16142	2163	2203
FB15k237_10	11512	237	27211	15624	18150
FB15k237_20	13166	237	54423	16963	19776
FB15k237_50	14149	237	136057	17449	20324
FB15k237	14541	237	272115	17535	20466
NELL23k	22925	200	25445	4961	4952
WN18RR	40943	11	86835	3034	3134
AristoV4	44949	1605	242567	20000	20000
Hetionet	45158	24	2025177	112510	112510
NELL995	74536	200	149678	543	2818
ConceptNet100k	78334	34	100000	1200	1200
DBpedia100k	99604	470	597572	50000	50000
YAGO310	123182	37	1079040	5000	5000

As shown in Table 3, ULTRA, TRIX, and MOTIF are fastest after graph-level preprocessing, using direct structure-based scoring. KGICL, KGPFN, and FLOCK additionally retrieve or sample graph evidence for inference. Among these three methods, KGPFN achieves latency close to KGICL with substantially lower peak memory, and is faster and more memory-efficient than FLOCK in this benchmark. KGPFN amortizes all-tail scoring through a shared structure encoding and cached PFN context.

C More Experiments

C.1 Datasets

Transductive datasets (16). The transductive split contains graphs where test entities appear in the training graph. It includes FB15k-237 Toutanova & Chen (2015) and its sparse subsets (FB15k-237-10%, FB15k-237-20%, FB15k-237-50%), WN18RR Dettmers et al. (2018), CoDEx-Small Safavi & Koutra (2020), CoDEx-Medium Safavi & Koutra (2020), CoDEx-Large Safavi & Koutra (2020), NELL-995 Mitchell et al. (2018), ConceptNet100k Speer et al. (2017), DBpedia100k Lehmann et al. (2015), YAGO3-10 Suchanek et al. (2007), AristoV4 Chen et al. (2019), Hetionet Himmelstein et al. (2017), WDsinger Galkin et al. (2023), and NELL23k. The details for these datasets are reported in Table 4.

Inductive datasets — new entities, no new relations (18). These datasets introduce unseen entities at inference time while keeping the relation vocabulary fixed. They comprise the GraIL benchmark graphs Teru et al. (2020): FB15k-237-Inductive (v1–v4), WN18RR-Inductive (v1–v4), and NELL-Inductive (v1–v4); the ILPC 2022 graphs (small and large) Galkin et al. (2022); and 4 INDIGO graphs Liu et al. (2021); Hamaguchi et al. (2017): HM-1k, HM-3k, HM-5k, and HM-Indigo. The details for these datasets are reported in Table 5.

Fully inductive datasets — new entities and new relations (23). These datasets require generalization to both unseen entities and unseen relation types. They include the InGram graphs Lee et al. (2023): NL-Ingram (splits 0/25/50/75/100), FB-Ingram (25/50/75/100), and WK-Ingram (25/50/75/100); the MTDEA WikiTopics graphs Zhou et al. (2023): WikiTopics-MT1 (health, tax), WikiTopics-MT2 (org, sci), WikiTopics-MT3 (art, infra), and WikiTopics-MT4 (sci, health); and two additional MTDEA graphs, Metafam and FBNELL. The details for these datasets are reported in Table 6.

Table 5: 18 Inductive entity (e) datasets used in the experiments.

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
NELL v1	14	3103	4687	3103	4687	414	225	833	201
NELL v2	88	2564	8219	2564	8219	922	2086	4586	935
NELL v3	142	4647	16393	4647	16393	1851	3566	8048	1620
NELL 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
IndigoBM	229	12721	121601	12797	121601	14121	14775	250195	14904

Table 6: 23 Inductive entity and relation (e, r) datasets used in the experiments.

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

C.2 Training Details

Pretraining. We pretrain on FB15K-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), and CoDEx-Medium (Safavi & Koutra, 2020) with AdamW, a learning rate of 5×10^{-5}, and weight decay 10^{-6}. The batch size is 32, and each positive query is paired with 512 negative samples. The keep ratio ρ is drawn uniformly from $[0.5, 0.9]$. For each relation, the perturbation δ is drawn uniformly from $[0, 0.08]$ and added to ρ; this perturbed ratio is the rate at which that relation's edges

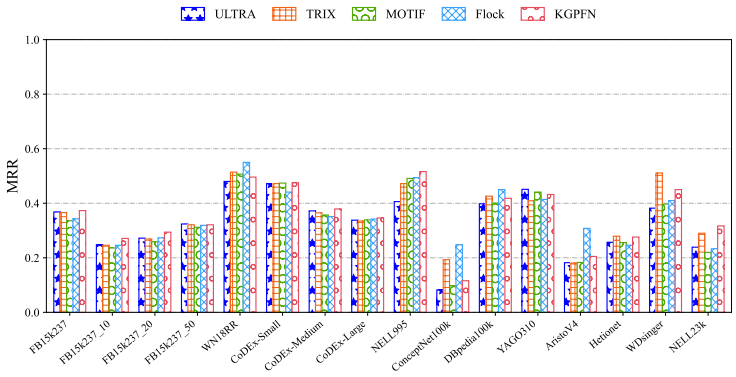
are retained in the support graph. Every 10 steps we resample the training episode: one of the three graphs is drawn first, and a support graph together with a query graph is then sampled from it. We count 100 steps as one epoch and train for 100 epochs.

Fine-tuning. Fine-tuning also uses AdamW and has two stages on each target dataset's training split. We first train the structure encoder for 3 epochs with learning rate 1×10^{-5}. We then freeze the encoder and train the PFN for 20 epochs, using the same optimizer settings as pretraining. At evaluation, context construction and graph encoding use the visible support edges of the target graph, excluding held-out prediction targets.

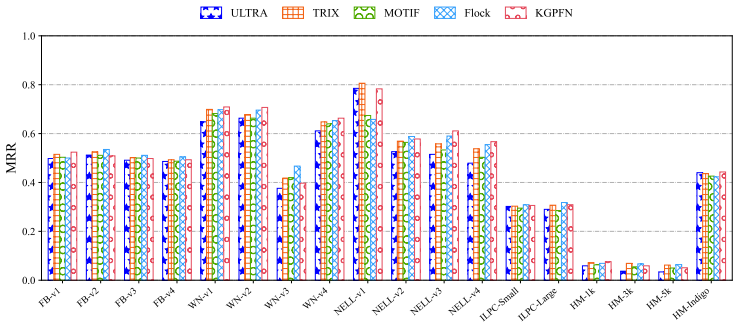
Global-context sampling. The default context budget is 20 positive and 80 negative triples. If fewer than 20 positive examples are available for the relation, the missing positive slots are reassigned to negative examples. For a given head h and relation r, the negative sampler targets approximately 80% of its tail candidates from the head's two-hop neighborhood, shuffling the neighbors before selection. The remaining approximately 20% are sampled uniformly from the graph's entity set. A shortfall in neighborhood candidates is filled by uniform sampling; if further candidates are needed, the sampler scans a shuffled entity list. At each stage, it rejects triples already present in the visible support graph, the current query triple, and positive triples selected for the context.

C.3 More results

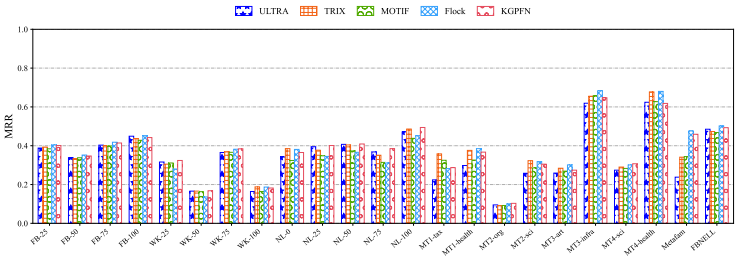
This section reports per-dataset MRR comparisons on the 57-graph benchmark. Figures 5a, 5b, and 5c show competitive performance without fine-tuning across the three settings.



(a) 16 transductive KGs.



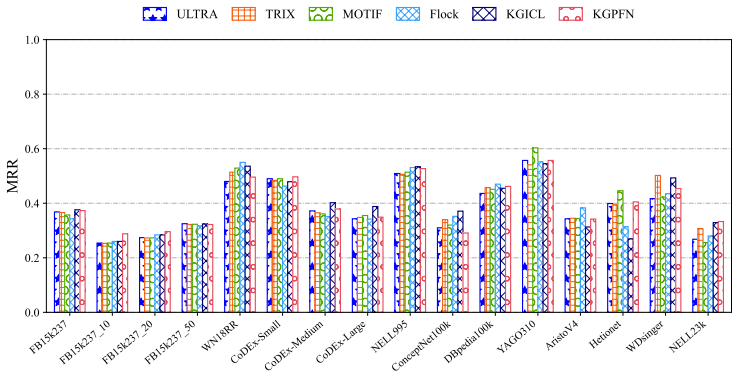
(b) 18 inductive KGs.



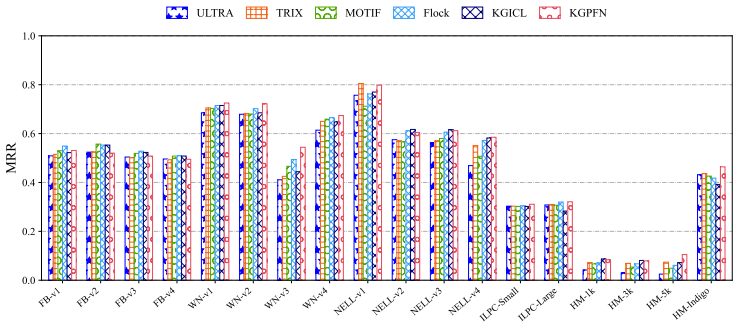
(c) 23 fully inductive KGs.

Figure 5: MRR without fine-tuning.

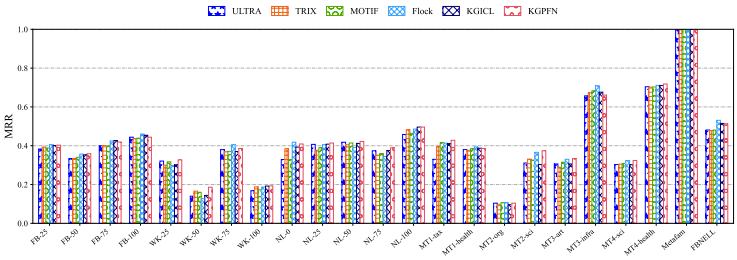
Fig. 6a, Fig. 6b and Fig. 6c show the same per-dataset comparison after fine-tuning.



(a) 16 transductive KGs.



(b) 18 inductive KGs.



(c) 23 fully inductive KGs.

Figure 6: MRR after fine-tuning.

Here, we report KGPFN's per-dataset mean and standard deviation over five inference runs with different context-sampling seeds, using a fixed checkpoint for each dataset and evaluation setting. W/o FT denotes inference without fine-tuning. The three pretraining graphs appear in Table 7, the other 13 transductive graphs in Table 8, and the inductive graphs in Tables 9 and 10. The main table's 16-graph transductive average includes both transductive subsets, and its 57-graph overall average includes every graph once. Per-dataset results are rounded to three decimal places; averaging these displayed values can introduce small rounding differences relative to averaging the unrounded results.

Table 7: Results on the three pretraining datasets (MRR / Hits@10).

	KGPFN (w/o FT)		KGPFN (fine-tuned)	
Dataset	**MRR**	**Hits@10**	**MRR**	**Hits@10**
FB15k237	0.373±0.004	0.570±0.003	0.373±0.004	0.570±0.003
WN18RR	0.496±0.026	0.634±0.029	0.496±0.026	0.634±0.029
CoDExMedium	0.379±0.004	0.535±0.004	0.379±0.004	0.535±0.004

Table 8: Results on the 13 transductive datasets outside the three-graph pretraining set (MRR / Hits@10).

Dataset	KGPFN (w/o FT) MRR	KGPFN (w/o FT) Hits@10	KGPFN (fine-tuned) MRR	KGPFN (fine-tuned) Hits@10
FB15k237_10	0.271±0.008	0.434±0.002	0.288±0.010	0.457±0.011
FB15k237_20	0.294±0.029	0.444±0.039	0.296±0.033	0.445±0.033
FB15k237_50	0.321±0.010	0.526±0.020	0.322±0.012	0.532±0.015
CoDExSmall	0.475±0.021	0.685±0.014	0.497±0.013	0.694±0.013
CoDExLarge	0.346±0.006	0.479±0.008	0.349±0.004	0.482±0.010
NELL995	0.516±0.007	0.666±0.009	0.527±0.010	0.670±0.007
ConceptNet100k	0.116±0.008	0.203±0.011	0.291±0.003	0.527±0.004
DBpedia100k	0.418±0.009	0.607±0.004	0.462±0.012	0.623±0.008
YAGO310	0.432±0.010	0.654±0.014	0.557±0.008	0.717±0.016
AristoV4	0.205±0.027	0.327±0.037	0.342±0.015	0.494±0.023
Hetionet	0.276±0.011	0.438±0.018	0.405±0.014	0.546±0.012
WDsinger	0.450±0.010	0.585±0.017	0.454±0.016	0.589±0.017
NELL23k	0.317±0.019	0.555±0.011	0.333±0.019	0.559±0.012

Table 9: Results on inductive datasets (MRR / Hits@10).

Dataset	KGPFN (w/o FT) MRR	KGPFN (w/o FT) Hits@10	KGPFN (fine-tuned) MRR	KGPFN (fine-tuned) Hits@10
FB15k237v1	0.524±0.002	0.686±0.003	0.531±0.001	0.703±0.006
FB15k237v2	0.509±0.009	0.723±0.008	0.520±0.010	0.736±0.006
FB15k237v3	0.498±0.008	0.663±0.004	0.508±0.005	0.672±0.006
FB15k237v4	0.493±0.009	0.693±0.007	0.495±0.006	0.698±0.010
WN18RRv1	0.709±0.003	0.812±0.007	0.725±0.000	0.826±0.002
WN18RRv2	0.707±0.008	0.809±0.006	0.722±0.008	0.816±0.008
WN18RRv3	0.397±0.004	0.512±0.007	0.544±0.005	0.647±0.004
WN18RRv4	0.663±0.004	0.738±0.009	0.674±0.006	0.744±0.004
NELLv1	0.783±0.005	0.886±0.005	0.798±0.002	1.000±0.001
NELLv2	0.578±0.010	0.775±0.010	0.604±0.009	0.777±0.006
NELLv3	0.611±0.014	0.787±0.012	0.612±0.015	0.797±0.013
NELLv4	0.567±0.016	0.811±0.020	0.585±0.009	0.811±0.009
ILPC-small	0.306±0.006	0.461±0.008	0.311±0.009	0.467±0.013
ILPC-large	0.309±0.007	0.439±0.013	0.321±0.005	0.443±0.008
HM-1k	0.076±0.001	0.119±0.002	0.084±0.000	0.146±0.001
HM-3k	0.059±0.002	0.096±0.012	0.079±0.004	0.101±0.007
HM-5k	0.051±0.006	0.082±0.002	0.105±0.003	0.146±0.002
HM-indigo	0.443±0.011	0.662±0.014	0.464±0.006	0.684±0.005

Table 10: Results on fully inductive datasets (MRR / Hits@10).

Dataset	KGPFN (w/o FT) MRR	KGPFN (w/o FT) Hits@10	KGPFN (fine-tuned) MRR	KGPFN (fine-tuned) Hits@10
NL-0	0.365±0.008	0.606±0.008	0.409±0.009	0.615±0.013
NL-25	0.401±0.004	0.607±0.009	0.414±0.009	0.618±0.005
NL-50	0.409±0.009	0.574±0.008	0.422±0.009	0.618±0.004
NL-75	0.385±0.003	0.570±0.005	0.389±0.003	0.576±0.007
NL-100	0.494±0.007	0.695±0.011	0.496±0.004	0.730±0.004
FB-25	0.401±0.018	0.654±0.035	0.403±0.012	0.662±0.029
FB-50	0.347±0.012	0.553±0.020	0.359±0.013	0.578±0.017
FB-75	0.414±0.009	0.619±0.003	0.419±0.007	0.629±0.005
FB-100	0.442±0.007	0.655±0.011	0.444±0.007	0.656±0.009
WK-25	0.324±0.002	0.535±0.007	0.327±0.003	0.537±0.012
WK-50	0.168±0.019	0.330±0.006	0.186±0.019	0.343±0.008
WK-75	0.384±0.012	0.542±0.012	0.386±0.011	0.559±0.007
WK-100	0.182±0.015	0.299±0.022	0.193±0.014	0.324±0.016
MT1-health	0.367±0.008	0.477±0.012	0.386±0.009	0.478±0.010
MT1-tax	0.287±0.008	0.348±0.002	0.428±0.005	0.576±0.003
MT2-org	0.103±0.003	0.181±0.003	0.104±0.002	0.186±0.005
MT2-sci	0.305±0.003	0.438±0.004	0.374±0.009	0.548±0.006
MT3-art	0.274±0.007	0.454±0.020	0.334±0.011	0.512±0.016
MT3-infra	0.647±0.009	0.813±0.012	0.661±0.007	0.823±0.011
MT4-sci	0.307±0.006	0.489±0.005	0.324±0.009	0.512±0.011
MT4-health	0.618±0.001	0.768±0.001	0.718±0.002	0.805±0.003
Metafam	0.459±0.001	0.932±0.000	1.000±0.001	1.000±0.000
FBNELL	0.493±0.007	0.686±0.007	0.513±0.004	0.689±0.001

D AI USE STATEMENT

We used generative AI tools to polish the language of the manuscript and to assist with writing, as well as to help implement experimental and analysis code. We also used AI assistance in drafting the proofs of the theoretical results; the authors have independently checked the correctness of all statements and proofs. We have reviewed all AI-assisted text and code, and take full responsibility for the final content of this paper, including its claims, wording, and experimental results.