# SURF: EXPRESSIVE STRUCTURAL ADAPTATION FOR BILLION-SCALE LINK PREDICTION WITH GRAPH FOUNDATION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In real-world recommendation systems, link prediction often takes the form of retrieval: identifying relevant items or entities from massive candidate populations. Graph Foundation Models (GFMs) offer a way to share predictive knowledge across the relationships, tasks, and graph domains underlying these recommendations. To serve such retrieval workloads, however, they must capture diverse structural linking patterns while preserving efficient indexed candidate search. We identify three bottlenecks in retrieval-compatible GFMs: neighborhood sampling can discard predictive structural evidence, standard similarity readouts restrict how this evidence can be used, and shared structural mechanisms can suffer negative transfer across relationships. To address these limitations, we introduce SURF[1], which augments a pretrained graph-transformer backbone (HGT by default) with precomputed full-neighborhood node signatures, lightweight relation-conditioned structural operators, and score fusion. Each domain–relationship task adapts and combines structural evidence without updating the shared backbone. By confining relation-dependent computation to the query side, SURF preserves reusable candidate representations and indexed retrieval without online graph traversal or query–candidate-specific neural computation. Our analysis shows how suitable signature channels and lightweight query-side operators can express neighborhood-overlap and finite random-walk heuristics. Across 16 datasets, SURF improves MRR over its HGT backbone on 14 under sampled large-candidate retrieval, including all six held-out domains after adaptation. On three real-world production tasks, SURF achieves 73.4–183.6% relative MRR gains across three backbones, each evaluated on over one billion query–candidate pairs. These results demonstrate that structural expressivity and large-scale indexed retrieval need not be competing objectives for GFMs.

## 1 INTRODUCTION

Large-scale online platforms operate on massive heterogeneous graphs, connecting diverse entities through billions of interactions (Badrinath et al., 2025; Yoon et al., 2022; Hang et al., 2024). Recommendation tasks such as suggesting friends, content, or advertisements can be formulated as link prediction, enabling shared models to reuse knowledge across tasks. At serving time, this often means retrieving relevant target entities from massive candidate populations. However, different relationships might not depend on the same evidence: shared neighbors may be especially informative for friendship recommendation, whereas content attributes and user interests may be more informative for content recommendation. This motivates models that share useful representations while allowing the linking patterns to vary across relationships.

Graph Foundation Models (GFMs) offer a natural starting point for such representation sharing by reusing pretrained models across graphs and tasks. Scalable GFMs such as GraphBFF extend this approach to massive heterogeneous graphs (Bechler-Speicher et al., 2026). When paired with an inner-product readout, node-level encoders also provide a practical interface for large-scale retrieval: candidate embeddings can be computed offline, cached, and indexed using maximum inner-product search (MIPS). An incoming query can then retrieve candidates without running a graph encoder separately for every query–candidate pair. This separation between offline representation learning and online candidate retrieval makes these backbones attractive for applications with large candidate populations and strict latency requirements.

[1] Source code: `https://anonymous.4open.science/r/SURF_GFM`



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.

However, a retrieval-compatible representation does not by itself ensure that the scoring function captures the structural patterns needed for link prediction. Predictive evidence can depend on structural information between the two endpoints' neighborhoods, such as shared neighbors and connecting paths, rather than on their individual attributes alone. Link-prediction models such as NBFNet and ULTRA exploit query-conditioned propagation to capture topological heuristics, with ULTRA additionally supporting relation-conditioned reasoning across graphs (Zhu et al., 2021; Galkin et al., 2024). Their query-conditioned graph computation, however, cannot be directly replaced by a search over fixed, cached candidate embeddings. Conversely, scalable node-embedding GFMs such as GraphBFF (Bechler-Speicher et al., 2026) provide this retrieval interface, but sampled encoders and standard similarity readouts do not automatically capture the same structural information. The challenge is therefore to enrich retrieval-compatible GFMs with relation-conditioned structural reasoning while retaining cached candidate representations and avoiding online graph traversal.

We introduce SURF (Semantic and strUctural Relation-Aware Fusion), a model that augments retrieval-compatible graph transformer backbones with lightweight, relation-conditioned structural adaptation (Figure 1). SURF combines pretrained node embeddings with expressive structural signatures computed offline using full neighborhood information from the observed graph. Compact domain–relationship-specific operators learn how to combine structural information across neighborhood scales, while learned fusion weights balance the scores resulting from structure and the backbone model. These operators act only on the query representation, allowing the same cached candidate vectors to be reused across relationships. The resulting score remains an inner product, $s_d(u,r,v) = \mathbf{q}_d(u,r)^\top \mathbf{c}(v)$, enabling indexed retrieval without online graph traversal or query–candidate-specific neural inference. We use HGT as the default backbone (Hu et al., 2020), keeping its embeddings frozen while fitting the structural operators and fusion weights.

**Contributions.** Our contributions are threefold:

1. **Expressive, relation-conditioned structural adaptation.** We introduce SURF, which augments retrieval-compatible graph transformer backbones with lightweight structural adaptation. We characterize the structural functions representable by its signature-based scoring mechanism, showing how relation-conditioned operators can recover neighborhood-overlap and finite random-walk heuristics while preserving a reusable candidate index.
2. **Bridging structural expressivity and retrieval compatibility.** Across 16 datasets and two backbone architectures, SURF substantially improves retrieval-compatible GFMs, including on held-out graphs after adaptation. Our experiments further show that relation-specific structural adaptation mitigates interference from sharing structural mechanisms across heterogeneous relationships, while retaining an inner-product scoring interface for indexed retrieval.
3. **Billion-scale evaluation on production data.** On a real-world production dataset, we evaluate three backbones on more than one billion query–candidate pairs each, spanning 3 different retrieval tasks. SURF improves MRR and Hits@1 in all nine backbone–task combinations, with aggregate relative MRR gains of 73.4–183.6%.

## 2 Related Work

**Pretrained graph models and scalable backbones.** Graph Foundation Models seek reusable representations across graphs and tasks (Xia & Huang, 2026; Bechler-Speicher et al., 2026; Wang et al., 2024b;c; Zhao et al., 2024; Huang et al., 2023). GraphBFF emphasizes large-scale pretraining on heterogeneous graphs (Bechler-Speicher et al., 2026), while AnyGraph uses a mixture-of-experts architecture to accommodate graph diversity (Xia & Huang, 2026). Graph diversity also motivates architectural and theoretical studies of transfer (Xia & Huang, 2026; Shi et al., 2026; Zhu et al., 2026). Complementary encoder architectures include HGT, which incorporates node and edge types into heterogeneous graph attention (Hu et al., 2020), and GraphGPS, which combines local message passing with global attention and positional or structural encodings (Rampášek et al., 2023).

**Expressive link prediction and transferable reasoning.** Expressive link predictors model structural dependencies between target endpoints through node labeling, enclosing-subgraph processing, and query-conditioned propagation (Zhang et al., 2021; Lachi et al., 2025). SEAL learns from labeled enclosing subgraphs (Zhang & Chen, 2018), while NBFNet uses source-conditioned message passing to learn path-based scores (Zhu et al., 2021). Building on query-conditioned reasoning, ULTRA learns transferable relational representations for prediction across knowledge graphs (Galkin et al., 2024), while UniLP explores cross-graph link prediction through in-context learning (Dong et al., 2024b). These methods

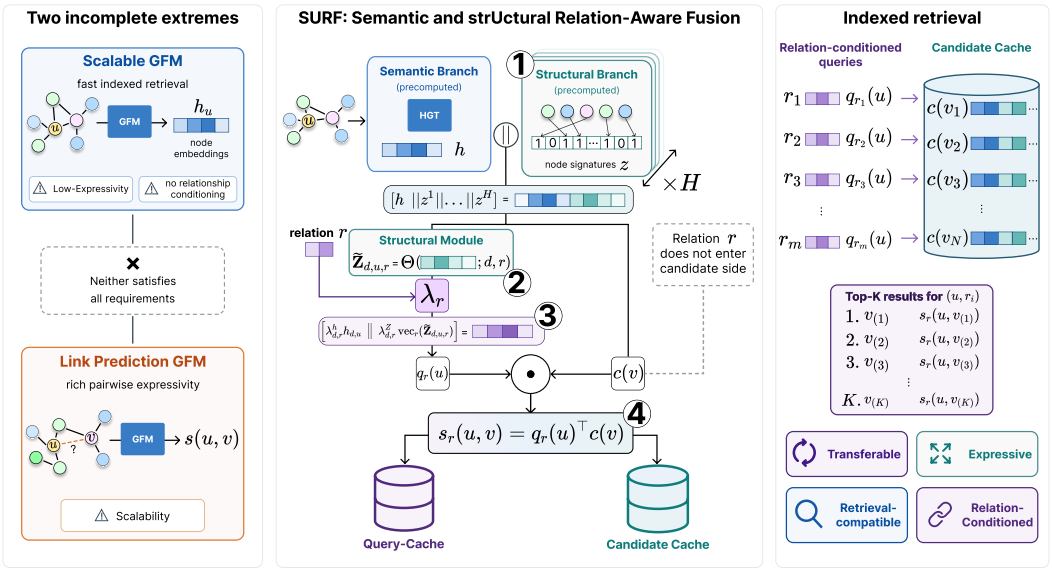


Figure 1: **SURF bridges the expressivity–retrieval gap in graph foundation models.** Left: cached-embedding GFMs versus pairwise link predictors. Numbered stages show SURF: **(1) Precomputation** (Section 4.1): cache backbone embeddings $\mathbf{h}_v$ and full-neighborhood signatures $\mathbf{Z}_v$, retaining structural information. **(2) Structural adaptation** (Section 4.2): apply task-local $\Theta_{d,r}$ to query signatures while keeping candidate signatures fixed. **(3) Fusion and query construction** (Sections 4.2–4.3): combine calibrated backbone, raw, and L2-normalized structural scores with $\boldsymbol{\lambda}_{d,r}$, compiling task-specific transformations into $\mathbf{q}_d(u,r)$. **(4) Indexed retrieval** (Section 4.3): score via $s_d(u,r,v) = \mathbf{q}_d(u,r)^\top \mathbf{c}(v)$, reusing one candidate index per graph snapshot across relationships. The MIPS factorization is exact; ANN search approximates its ranking. The schematic abbreviates domain indices and omits branch and calibration details.

demonstrate the value of structural reasoning beyond independent endpoint representations, but their query- or pair-dependent computation differs from retrieval over fixed, cached candidate vectors.

**Structural signatures and topology-aware predictors.** A complementary line of work captures link-predictive structure through neighborhood summaries and learned topological heuristics. Bloom signatures compress node neighborhoods into hashing-based representations from which overlap-related structural features can be estimated (Zhang et al., 2024b). MPLP uses message passing over quasi-orthogonal node identifiers to estimate common-neighbor information while retaining node-level computational complexity (Dong et al., 2024a). NCN and NCNC incorporate common-neighbor information into neural link predictors (Wang et al., 2024a), while HL-GNN learns combinations of structural heuristics through graph propagation (Zhang et al., 2024a).

**Relation-conditioned scoring and structural specialization.** Knowledge-graph embedding models such as RESCAL (Nickel et al., 2011) and TuckER (Balažević et al., 2019) combine relation-dependent bilinear scoring with shared entity representations. SURF specializes a lightweight structural readout rather than the backbone: domain–relationship-specific operators act on precomputed signatures, and learned fusion weights combine structural and pretrained backbone scores. Its query-side factorization retains a common cached candidate representation across relationships.

## 3 Problem Setting

**Domains and prediction tasks.** We consider a collection of graph domains $d \in \mathcal{D}$. Each graph domain $d \in \mathcal{D}$ has an observed heterogeneous graph $G_d = (V_d, E_d)$ with nodes $V_d$ and typed edges $E_d \subseteq V_d \times \mathcal{R}_d \times V_d$. Here, $\mathcal{R}_d$ is the set of relationship types in domain $d$, and $\tau_d(v)$ denotes the type of node $v$. A prediction task is identified by a domain–relationship pair $(d,r)$. Different relationships can share endpoint types: viewing and sharing both link users to content but define different tasks.

**Candidate ranking and observed information.** For a query $(u,r,?)$, we rank candidates $v \in \mathcal{C}_{d,r} \subseteq V_d$ compatible with relationship $r$ using a score $s_d(u,r,v)$. Training, validation, and test positives form pairwise disjoint supervision sets $E_d^{\text{train}}$, $E_d^{\text{val}}$, and $E_d^{\text{test}}$. The observed graph may include other relationships and need not coincide with training supervision. For missing-link evaluation, we exclude validation and test positives from the observed graph used by both backbone message passing and signature preprocessing.

**Pretraining and downstream adaptation.** A backbone is jointly pretrained across source domains and reused on source or held-out domains. Downstream training supervision may be used to adapt it; transfer therefore does not assume zero-shot prediction. In SURF, the resulting embeddings are held fixed while structural operators and fusion weights are fitted for each domain–relationship task.

**Cached, relation-conditioned retrieval.** We consider scoring functions that admit the factorization

$$s_d(u,r,v) = \mathbf{q}_d(u,r)^\top \mathbf{c}(v),$$

where the query vector depends on $u$ and $r$, while the candidate vector is independent of both. For a fixed graph snapshot, candidate vectors are computed offline and reused across relationships. Retrieval uses exact maximum inner-product search (MIPS) or an approximate nearest-neighbor index over compatible candidates $\mathcal{C}_{d,r}$. Online query construction uses cached representations, not graph traversal or query-specific subgraph extraction. Relation-conditioned scoring need not reduce to relation-independent embedding similarity (Balažević et al., 2019; Wang et al., 2014; Nickel et al., 2011). We omit $d$ from node representations and graph operators when the domain is clear; they still depend on its graph snapshot.

# 4 SURF: SEMANTIC AND STRUCTURAL RELATION-AWARE FUSION

SURF combines pretrained node representations with a lightweight, task-local structural model (Figure 1). We precompute backbone embeddings and separate structural signatures from the observed graph, keeping both fixed while optimizing the operators and fusion weights. Task-specific transformations are compiled into query vectors, leaving candidate vectors unchanged.

## 4.1 BACKBONE REPRESENTATIONS AND STRUCTURAL SIGNATURES

**Pretrained backbone embeddings.** For each node $v \in V_d$, let $\mathbf{h}_v \in \mathbb{R}^{p_h}$ denote the embedding produced by the pretrained Graph Transformer backbone (HGT by default) from node features and the observed graph (Hu et al., 2020). These embeddings capture feature and sampled-neighborhood information and are cached before fitting task-local operators and fusion weights. Their inner-product score, $s_d^h(u,v) = \langle \mathbf{h}_u, \mathbf{h}_v \rangle$, is shared across relationships within a domain.

**Structural node signatures.** We precompute fixed-width structural signatures from full observed neighborhoods, independently of backbone sampling and query–candidate pairs. The information they retain depends on the choice of encoding or sketch (Rampášek et al., 2023; Maskey et al., 2022; Zhang et al., 2024b; Airale et al., 2025; Dong et al., 2024a). We represent each node $v$ through $H$ channels $\mathbf{z}_v^{(\ell)} \in \mathbb{R}^{p_z}$, indexed by $\ell \in \{1,...,H\}$, and stack their transposes as rows of $\mathbf{Z}_v \in \mathbb{R}^{H \times p_z}$. We write $\mathbf{Z}^{(\ell)}$ for the all-node matrix of channel $\ell$. Our implementation uses $H = 2$, corresponding to one- and two-hop aggregation.

**Log-count Bloom signatures.** We construct structural signatures using a log-count variant of the Bloom signatures of Zhang et al. (2024b), which compress neighborhood information by hashing node identities into a fixed number of buckets. Let $\kappa : V_d \to \{1,...,p_z\}$ be a hash function and $R \in \mathbb{R}^{|V_d| \times p_z}$ the corresponding identifier matrix, with $R_{vb} = \mathbf{1}\{\kappa(v) = b\}$. Different nodes may share a bucket, so compressed signatures need not preserve neighborhood overlap exactly. Using the observed graph adjacency matrix $A$, we compute

$$\mathbf{Z}^{\text{raw},(\ell)} = \log\Big(1 + A\mathbf{Z}^{\text{raw},(\ell-1)}\Big), \quad \ell \in \{1,...,H\}, \tag{1}$$

where $\mathbf{Z}^{\text{raw},(0)} = R$ initializes the recurrence, the logarithm is applied elementwise, and row $v$ of $\mathbf{Z}^{\text{raw},(\ell)}$ is $(\mathbf{z}_v^{\text{raw},(\ell)})^\top$. The initialization is not included in the signature bank $\mathbf{Z}_v$, which contains channels $1,...,H$. We compute all $H$ channels offline and use "raw" to distinguish these log-count signatures from their L2-normalized counterparts. Unlike linear propagation $A^\ell R$, this repeated nonlinear aggregation does not produce exact walk counts. The aggregation steps are described in Appendix E.

**Normalized structural branch.** Alongside the raw log-count signatures, SURF uses an L2-normalized branch that divides each nonzero channel vector by its Euclidean norm and leaves zero vectors unchanged. The raw branch provides magnitude-sensitive structural evidence, whereas the normalized branch emphasizes similarity in the distribution of that evidence. For each branch $e \in \{\mathrm{raw}, \mathrm{L2}\}$, we stack the two channels into $\mathbf{Z}_v^e \in \mathbb{R}^{2 \times p_z}$. Both branches derive from the same precomputed signatures and remain fixed during structural adaptation; the operators and fusion weights determine how their scores contribute to prediction.

### 4.2 Relation-Conditioned Structural Adaptation

**Task-local structural operator.** We describe structural adaptation for one signature branch, omitting its branch superscript; the same construction is applied independently to the raw and L2-normalized branches. For each domain–relationship task $(d,r)$, a structural module $\Theta_{d,r}$ maps the query's signature bank $\mathbf{Z}_u$ to a task-conditioned representation of the same shape. This module may use different parameterizations, including an MLP or a Transformer over signature channels, provided that it operates on cached query information without inspecting a candidate or traversing the graph online. We instantiate it as a linear channel-mixing operator—a simple, lightweight choice that learns interactions between neighborhood scales:

$$\Theta_{d,r}(\mathbf{Z}_u) = \mathbf{K}_{d,r}^{\top} \mathbf{Z}_u, \qquad \mathbf{K}_{d,r} \in \mathbb{R}^{H \times H}. \tag{2}$$

With $H = 2$, this implementation requires only four learned coefficients per branch and task, independently of signature width or graph size. The candidate signature $\mathbf{Z}_v$ remains unchanged across queries and relationships, while the structural module determines how the query's structural information is compared with it.

**Structural score and hop-pair interactions.** For a single structural branch, the score is $s_d^{\mathrm{str}}(u,r,v) = \langle \Theta_{d,r}(\mathbf{Z}_u), \mathbf{Z}_v \rangle_F$, where $\langle \cdot,\cdot \rangle_F$ denotes the Frobenius inner product. For our linear operator, this becomes

$$s_d^{\mathrm{str}}(u,r,v) = \sum_{i=1}^{H} \sum_{j=1}^{H} (\mathbf{K}_{d,r})_{ij} \left\langle \mathbf{z}_u^{(i)}, \mathbf{z}_v^{(j)} \right\rangle. \tag{3}$$

Diagonal coefficients weight same-hop comparisons, while off-diagonal coefficients weight cross-hop comparisons. For example, $(\mathbf{K}_{d,r})_{12}$ weights the interaction between the query's one-hop signature and the candidate's two-hop signature. Each domain–relationship task learns its own coefficients, so the same cached signatures can support different structural linking rules.

**Fusion with the backbone score.** SURF combines the backbone score $s_d^h(u,v)$ with the raw and normalized structural scores, denoted by $s_d^{\mathrm{raw}}(u,r,v)$ and $s_d^{\mathrm{L2}}(u,r,v)$, through learned late-score fusion (Song et al., 2025). For each domain–relationship task, the final score is

$$s_d(u,r,v) = \lambda_{d,r}^{h} \frac{s_d^h(u,v)}{\rho_{d,r}^{h}} + \lambda_{d,r}^{\mathrm{raw}} \frac{s_d^{\mathrm{raw}}(u,r,v)}{\rho_{d,r}^{\mathrm{raw}}} + \lambda_{d,r}^{\mathrm{L2}} \frac{s_d^{\mathrm{L2}}(u,r,v)}{\rho_{d,r}^{\mathrm{L2}}}, \tag{4}$$

where $\rho_{d,r}^{e}$ is a fixed, training-derived calibration scale for score component $e$, and $\boldsymbol{\lambda}_{d,r} = \mathrm{softmax}(\boldsymbol{\eta}_{d,r})$ converts three learned task-specific logits into nonnegative fusion weights that sum to one. These weights balance backbone, magnitude-sensitive structural, and normalized structural evidence. Fusion operates on the scores rather than injecting signatures into the backbone's attention or message-passing layers. We describe calibration and fitting in the next subsection.

### 4.3 Training and Indexed Retrieval

**Staged structural adaptation.** With backbone embeddings and signatures cached and frozen, we first fit the raw and L2-normalized operators independently for each task $(d,r)$. Each fit minimizes a softplus pairwise ranking loss over training positives and sampled negative candidates, updating only the corresponding operator. We then calibrate each component $e \in \{h, \mathrm{raw}, \mathrm{L2}\}$ by the root-mean-square of its positive–negative training-score differences. With operators and calibration scales fixed, we fit only the three fusion logits $\boldsymbol{\eta}_{d,r}$ from the calibrated training scores; the scales also remain fixed during retrieval. The two-channel linear implementation has eight kernel coefficients and three fusion logits, totaling 11 trainable scalars per task; fitting does not backpropagate through the backbone. Appendix E gives the loss, calibration, and optimization details.

**Compiling query and candidate representations.** Let $\mathrm{vec}(\cdot)$ denote row-wise vectorization and $\|$ denote vector concatenation. We compile all task-dependent transformations into the query vector, leaving the candidate vector unchanged:

$$\begin{aligned}\mathbf{q}_d(u,r) &:= \left[\frac{\lambda^h_{d,r}}{\rho^h_{d,r}}\mathbf{h}_u \middle\| \frac{\lambda^{\mathrm{raw}}_{d,r}}{\rho^{\mathrm{raw}}_{d,r}}\mathrm{vec}(\Theta^{\mathrm{raw}}_{d,r}(\mathbf{Z}^{\mathrm{raw}}_u)) \middle\| \frac{\lambda^{\mathrm{L2}}_{d,r}}{\rho^{\mathrm{L2}}_{d,r}}\mathrm{vec}(\Theta^{\mathrm{L2}}_{d,r}(\mathbf{Z}^{\mathrm{L2}}_u))\right], \\ \mathbf{c}(v) &:= \left[\mathbf{h}_v \middle\| \mathrm{vec}(\mathbf{Z}^{\mathrm{raw}}_v) \middle\| \mathrm{vec}(\mathbf{Z}^{\mathrm{L2}}_v)\right].\end{aligned} \tag{5}$$

Their inner product recovers the fused score, $s_d(u,r,v) = \mathbf{q}_d(u,r)^\top \mathbf{c}(v)$. Candidate vectors can therefore be indexed once per graph snapshot and reused across relationships. Query construction uses cached inputs and task-local parameters, requiring neither online graph traversal nor pairwise neural inference. This factorization also holds for nonlinear structural modules, provided that they operate only on cached query information and produce outputs compatible with the candidate signature banks.

### 4.4 Structural Expressivity

Appendix F.1 gives an example where both implemented log-count branches, with fixed first-hop operators, distinguish neighborhood overlaps in expectation over hashing despite equal backbone scores. The following theorem instead characterizes recovery of classical heuristics and approximation error for linear signature channels, separately from the nonlinear log-count implementation.

**Theorem 1** (Expressivity of query-side signature operators)**.** *Fix a domain–relationship task on $n$ nodes. Let $B_0,\ldots,B_H \in \mathbb{R}^{n\times n}_{\geq 0}$ store uncompressed structural signatures as rows, with $B_0 = I$; $(B_i)_{uv}$ is node $v$'s weight in node $u$'s signature channel $i$. Let $R \in \mathbb{R}^{n\times p_z}$ be the identifier matrix, with unit-norm rows. Form the fixed-width signatures $\mathbf{z}^{(i)}_v = (B_i R)^\top_{v,:}$ and stack their transposes into the signature bank $\overline{\mathbf{Z}}_v$, including channel 0. For any channel-mixing operator $\mathbf{K} \in \mathbb{R}^{(H+1)\times(H+1)}$, define the structural score $s_{\mathbf{K}}(u,v) = \langle \mathbf{K}^\top \overline{\mathbf{Z}}_u, \overline{\mathbf{Z}}_v \rangle_F$. If $RR^\top = I$, then*

$$s_{\mathbf{K}}(u,v) = \sum_{i,j=0}^{H} K_{ij}(B_i B_j^\top)_{uv}. \tag{6}$$

*Consequently, suitable signature banks recover:*

- *(i) weighted common-neighbor scores $(AWA^\top)_{uv}$, using $B_i = AW^{1/2}$ for nonnegative diagonal $W$. These include Common Neighbors, Resource Allocation, and Adamic–Adar.*
- *(ii) finite random-walk scores $\sum_{i=0}^{H}\beta_i(P^i)_{uv}$, using $B_i = P^i$, including $B_0 = I$, where $P$ is a transition matrix.*

*Let $\mu = \max_{a\neq b}|(RR^\top)_{ab}|$ denote identifier coherence and $m_i(u) = \|(B_i)_{u,:}\|_1$ signature mass. Then*

$$\left| s_{\mathbf{K}}(u,v) - \sum_{i,j=0}^{H} K_{ij}(B_i B_j^\top)_{uv} \right| \leq \mu \sum_{i,j=0}^{H} |K_{ij}| \left[ m_i(u) m_j(v) - (B_i B_j^\top)_{uv} \right]. \tag{7}$$

The theorem shows that cached structural signatures can recover familiar link-prediction heuristics with fixed candidate vectors and no online graph traversal. Orthogonal node identifiers give exact recovery; otherwise, similarities between different identifiers can introduce spurious overlap. The bound quantifies how this error depends on identifier similarity, uncompressed signature mass, and channel-mixing coefficients. Appendix F provides the proof and the step-by-step bound derivation.

**Task-conditioned structural comparisons.** A dot product between concatenated signature channels fixes $\mathbf{K}_{d,r} = I$, allowing only same-channel comparisons shared across tasks. Learning $\mathbf{K}_{d,r}$ adds task-specific weighting of both same-channel and cross-channel evidence.

## 5 Experiments

We evaluate whether structural adaptation improves link prediction and how it affects preprocessing and online retrieval costs. We use HGT for the academic evaluation, test cumulative adaptations on GraphBFF, and evaluate GraphBFF, HGT, and GraphGPS on production data.

| Dataset | AnyGraph | | GraphGPS | | GraphBFF | | HGT | | SURF (GraphBFF) | | SURF (HGT) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) |
| *Train–test datasets* | | | | | | | | | | | | |
| amazon-book | 0.00±0.00 | 0.01±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | **5.64±0.07** | **6.62±0.09** | **5.60±0.07** | **6.57±0.09** |
| fitness | 0.15±0.05 | 1.03±0.65 | 0.01±0.00 | 3.10±0.00 | 0.00±0.00 | 4.60±0.03 | 0.01±0.00 | 2.66±0.01 | **22.29±0.53** | **25.11±0.50** | **21.69±0.36** | **24.44±0.33** |
| goodreads | 0.01±0.01 | 0.14±0.11 | 0.00±0.00 | 0.80±0.06 | 0.00±0.00 | 1.13±0.07 | 0.00±0.00 | 0.47±0.02 | **25.74±1.84** | **28.67±1.92** | **28.53±1.61** | **31.80±1.62** |
| movielens-10m | 0.00±0.00 | 0.02±0.01 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | **6.14±0.46** | **7.35±0.46** | **6.12±0.46** | **7.31±0.49** |
| ogbl-biokg | 0.05±0.02 | 0.24±0.04 | 0.40±0.29 | 0.68±0.28 | 0.13±0.01 | 0.24±0.03 | 0.00±0.00 | 0.03±0.01 | **3.16±0.15** | **4.00±0.19** | **2.83±0.02** | **3.62±0.01** |
| rel-amazon | 0.01±0.00 | 0.16±0.03 | 0.02±0.03 | 0.14±0.06 | 0.02±0.02 | **0.19±0.05** | **0.07±0.01** | **0.32±0.04** | 0.04±0.00 | 0.08±0.01 | **0.10±0.00** | 0.17±0.00 |
| rel-hm | 0.04±0.04 | 0.26±0.20 | 0.08±0.06 | **0.28±0.16** | 0.01±0.00 | 0.18±0.03 | 0.04±0.03 | **0.28±0.05** | **0.13±0.03** | 0.24±0.05 | **0.11±0.01** | 0.22±0.01 |
| rel-stack | 0.00±0.00 | 0.01±0.00 | 0.00±0.00 | 0.83±0.24 | 0.56±0.15 | 1.20±0.06 | 0.00±0.00 | 0.39±0.05 | **5.40±0.69** | **5.49±0.63** | **5.89±0.00** | **6.09±0.08** |
| rel-trial | 0.04±0.03 | 2.39±1.21 | 0.64±0.82 | 2.02±1.73 | 0.75±0.14 | 3.17±0.21 | 1.99±0.31 | 4.14±0.18 | **11.05±0.54** | **11.67±0.42** | **9.62±0.94** | **10.46±0.80** |
| yelp2018 | 0.00±0.00 | 0.02±0.01 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | **8.70±0.49** | **10.16±0.52** | **8.71±0.51** | **10.17±0.53** |
| *Mean Rank* | 4.70 | 4.50 | 4.45 | 4.45 | 4.30 | 3.70 | 4.25 | 3.95 | **1.60** | **2.20** | **1.70** | **2.20** |
| *Fine-tuned / held-out datasets* | | | | | | | | | | | | |
| amazon-text | 0.00±0.00 | 8.82±0.08 | 0.00±0.00 | 0.88±0.01 | 0.00±0.00 | 0.40±0.04 | 0.00±0.00 | 1.06±0.01 | **35.76±1.82** | **38.67±1.99** | **36.10±1.38** | **39.02±1.54** |
| gowalla | 0.00±0.00 | 0.02±0.00 | 0.00±0.00 | 0.01±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | **14.94±0.19** | **16.70±0.18** | **14.89±0.05** | **16.64±0.03** |
| ogbl-wikikg2 | 0.24±0.01 | 0.29±0.01 | 0.43±0.00 | 0.48±0.00 | 0.01±0.00 | 0.03±0.00 | 1.43±0.29 | 1.44±0.28 | **11.80±0.00** | **12.83±0.05** | **11.22±1.16** | **12.21±1.19** |
| photo | 0.08±0.00 | 8.58±0.01 | 0.00±0.00 | 3.34±0.05 | 0.00±0.00 | 3.39±0.12 | 0.00±0.00 | 3.16±0.25 | **29.34±0.04** | **32.29±0.04** | **29.32±0.07** | **32.36±0.07** |
| rel-avito | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.01±0.00 | 0.00±0.00 | 0.01±0.00 | 0.00±0.00 | 0.01±0.01 | **0.93±0.02** | **0.98±0.02** | **1.13±0.14** | **1.17±0.14** |
| steam-text | 0.00±0.00 | 3.41±0.23 | 0.00±0.00 | 0.08±0.01 | 0.00±0.00 | 0.06±0.00 | 0.00±0.00 | 0.43±0.01 | **10.65±3.72** | **11.95±3.91** | **11.80±2.43** | **13.05±2.64** |
| *Mean Rank* | 4.27 | 3.83 | 4.44 | 4.50 | 5.02 | 5.38 | 4.40 | 4.42 | **1.44** | **1.60** | **1.44** | **1.27** |

Table 1: Sampled retrieval on ten train–test and six fine-tuned/held-out datasets with 10% of test positives and 10% of compatible candidates. H@1 and MRR (%) are means ± sample standard deviations across runs. ULTRA is omitted because scoring each dataset under this protocol is estimated to require more than 24 hours; completed runs on `rel-stack` and `amazon-text` exceeded this duration. **Blue**/**red** highlight first/second-place reported means per dataset and metric; ties share highlights and consume places. Mean ranks are computed over the displayed models (lower is better).

### 5.1 Academic benchmark setup

**Datasets.** Following GraphBFF (Bechler-Speicher et al., 2026), we evaluate on 16 datasets spanning recommendation, knowledge graphs, social networks, and relational prediction: ten are jointly pretrained on their training splits and evaluated on held-out test splits; six are excluded from pretraining and evaluated after target-domain adaptation, not strictly zero-shot. Table 10 summarizes the datasets and tasks (Appendix E).

**Baselines.** Baselines are the foundation models AnyGraph (Xia & Huang, 2026) (mixture of experts) and GraphBFF (Bechler-Speicher et al., 2026) (a scalable heterogeneous Graph Transformer), plus GraphGPS (Rampášek et al., 2023) and HGT (Hu et al., 2020) trained in the same multi-dataset setting. Unaugmented HGT is the direct backbone-only comparison for the default SURF. ULTRA's sampled pairwise implementation uses NBFNet-style query-conditioned message passing (Galkin et al., 2024; Zhu et al., 2021); large-candidate results are limited to evaluations estimated within 24 hours.

**Training and adaptation.** Backbones use AdamW for 25,000 pretraining and 10,000 held-out adaptation steps, with two random seeds and two-hop fanout [10,10] for AnyGraph, GraphBFF, GraphGPS, and HGT. For SURF, we then freeze the backbone, cache node embeddings, construct one- and two-hop signatures from the observed graph, and fit task-local structural operators and fusion weights as in Section 4.3. Appendix E gives optimizer settings, signature dimensions, fitting budgets, and validation.

**Evaluation.** Large-candidate retrieval uniformly samples 10% of test positives and 10% of compatible candidates without replacement, retaining the imbalance between links and compatible nonedges. Hits@1 and MRR use the minimum-bias sampled-metric estimator of Krichene & Rendle (2022) (Appendix E). We also report these metrics on balanced test sets with equal numbers of positive and negative pairs in Appendix C.

### 5.2 Link-prediction performance

Table 1 reports performance under the sampled retrieval protocol. With HGT as its backbone, SURF improves MRR over the unaugmented backbone on 14 of 16 datasets, including all six held-out domains after adaptation, under both sampled retrieval and balanced-negative evaluation (Table 8 in Appendix C). On Goodreads and Fitness, MRR increases from 0.47% to 31.80% and from 2.66% to 24.44%, respectively. Improvements also extend to held-out domains: on Amazon-Text, MRR rises from 1.06% to 39.02%. Gains are not universal: MRR decreases on Rel-Amazon and Rel-HM, although Hits@1 improves on both.

| Backbone | MRR lift (%) | Hits@1 lift (%) |
|---|---|---|
| GraphBFF | +103.5 | +325.3 |
| HGT | +183.6 | +399.4 |
| GraphGPS | +73.4 | +246.0 |

Table 2: Production results: lift is $100(m_{\mathrm{SURF}}/m_{\mathrm{base}}-1)$ after macro-averaging metric $m$ over Tasks A–C. Confidentiality requires masking task identities and absolute scores. Blue/red denote gains/decreases. Both models use identical test groups with 4,092 fixed negatives per positive.

| | Added components | | | Balanced negatives | | | Sampled full candidates | | | All 32 cells | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cumulative GraphBFF stage | Full-graph signatures $\mathbf{Z}_u$ | Relation readout $\mathbf{K}_{d,r}$ | Task-local fusion weights $\lambda^c_{d,r}$ | Macro MRR (%) | Mean rank ↓ | Wins /16 | Macro MRR (%) | Mean rank ↓ | Wins /16 | Mean rank ↓ | Wins |
| GraphBFF | × | × | × | 61.93 | 2.63 | 2 | 0.91 | 3.19 | 0 | 2.91 | 2/32 |
| → + full-graph signatures | ✓ | × | × | 58.83 | 3.19 | 0 | 2.54 | 2.88 | 0 | 3.03 | 0/32 |
| → + relation readout | ✓ | ✓ | × | 61.01 | 3.00 | 1 | 0.83 | 2.75 | 2 | 2.88 | 3/32 |
| → **SURF (GraphBFF)** | ✓ | ✓ | ✓ | **92.81** | **1.19** | **13** | **13.57** | **1.19** | **14** | **1.19** | **27/32** |

Table 3: **Cumulative GraphBFF ablation.** The cumulative ablation uses independent runs from the dataset-level result tables. Macro MRR averages 16 dataset means per protocol; ranks and wins are within dataset–protocol groups. Final columns aggregate all 32 groups. Augmented configurations use two runs; GraphBFF uses two for balanced-negative train/test evaluation and three otherwise.

### 5.3 Real-world evaluation on production data

We evaluate GraphBFF, HGT, and GraphGPS on three production prediction tasks from a large online platform, masking task identities as A–C and reporting only relative metric improvements for confidentiality. This production workload evaluates many queries against sampled candidates, totaling 1.105 billion scored pairs per backbone, compatible with industrial retrieval settings (Badrinath et al., 2025) (Appendix E.1). Table 2 shows aggregate relative MRR gains of 73.4–183.6%; MRR and Hits@1 improve in all nine backbone–task combinations. Per-task results are reported in Appendix E.1, Table 11.

### 5.4 Structural adaptation ablations

**Cumulative components and backbone generality.** Table 3 evaluates cumulative structural adaptations to GraphBFF, testing the approach beyond the default HGT backbone. Starting from GraphBFF, the successive stages introduce full-graph signatures, a relation-specific readout, and calibrated, learned fusion. The final stage replaces the uncalibrated sum with unit weights by RMS-calibrated scores and learned task-local fusion weights, leaving structural-kernel fitting unchanged. The complete configuration increases macro MRR from 0.91% to 13.57% under sampled retrieval, and from 61.93% to 92.81% under explicit-negative evaluation, achieving the highest MRR in 27 of 32 dataset–protocol combinations among the compared configurations. The tested partial configurations do not match the complete model's aggregate performance, highlighting the importance of combining structural signatures, relation-specific readout, and task-local fitting to address the limitations of the unaugmented backbone.

We also motivate task-local fitting by showing cross-dataset kernel sharing also degrades accuracy as the cohort grows (see Appendix B, Figure 4). Finally, Appendix D separately examines whether scaling GraphBFF closes the fusion gap.

**Relationship-specific fitting.** Table 4 evaluates SURF on `ogbl-biokg`, which contains 51 relationship types. Structural adaptation improves both backbones even with a dataset-wide structural model: MRR increases from 0.030% to 3.366% for HGT and from 0.240% to 3.143% for GraphBFF. Relationship-specific fitting further increases MRR to 3.621% and 3.995%, respectively, giving the highest performance among the compared configurations for both backbones. These results support combining structural adaptation with relationship-specific specialization, rather than imposing the same structural scoring function on every relationship in a graph.

**Per-relationship evidence.** Relationship-specific structural kernels outperform the dataset-wide kernel on 38 of BioKG's 51 relationships. Table 7 in Appendix A details these differences: structural-only MRR

| Backbone | Backbone only | SURF–Dataset | SURF |
|---|---|---|---|
| HGT | 0.030 | 3.366 | **3.621** |
| GraphBFF | 0.240 | 3.143 | **3.995** |

Table 4: MRR (%) on `ogbl-biokg` (51 relationship types), using each row's backbone. Backbone-only means come from Table 1; SURF values retain original precision after percentage conversion. SURF–Dataset fits a dataset-wide structural model ($\mathbf{K}_d$); SURF fits relationship-specific models ($\mathbf{K}_{d,r}$).

| Method | Latency for 100k candidates | Search result |
|---|---|---|
| ULTRA, sampled | 633.8 s (extrapolated) | Exhaustive candidate scoring |
| SURF, exact MIPS | 68.5 ms | Exact top-100 |
| SURF, HNSW | 0.253 ms | — |

Table 5: Retrieval latency for 100k candidates. ULTRA is extrapolated from measured 128-candidate batches, fanout [10,10], on an NVIDIA A100 80 GB GPU. Both SURF benchmarks use one AMD EPYC 7B13 CPU thread; exact MIPS uses shape-matched vectors with all 4,224 fused dimensions and includes top-100 selection. Exactness concerns model-score ranking, not predictive accuracy. The dash marks unreported search quality for untuned HNSW, whose latency is not quality-matched.

increases from 0.27% to 18.04% for protein–protein expression and from 12.26% to 32.28% for drug–drug structural similarity. These results support allowing the same cached signatures to be compared differently across relationships, rather than imposing a common structural scoring function throughout the graph.

### 5.5 Preprocessing and retrieval costs

**Online retrieval latency.** Over 100,000 candidates (Table 5), SURF's cached-vector exact-MIPS scoring and top-100 selection take 68.5 ms, approximately 9,250 times faster than ULTRA's estimated 633.8 seconds under the evaluated configurations. ULTRA takes 810.4 ms per 128-candidate batch, with 96.8% spent on neighborhood sampling; SURF requires no online graph traversal. Untuned HNSW returns results in 0.253 ms, without quality matching. The factorization supports standard inner-product indexes (such as FAISS (Johnson et al., 2021) and HNSW (Malkov & Yashunin, 2020)); deployment-specific optimization of index memory, latency, and recall is beyond this work's scope.

Appendix E, Table 9, reports industrial preprocessing costs. Appendix A examines structural-fitting budgets, signature width, precision, and normalization (Table 6; Figure 3).

## 6 Conclusion

We introduced SURF, a structural adaptation model that combines pretrained graph representations with cached node signatures and task-local structural operators. SURF incorporates relationship-specific structural evidence while preserving a query–candidate inner-product score suitable for indexed retrieval. Experiments with HGT as the default backbone show improved retrieval performance across most evaluated datasets, while the GraphBFF ablation demonstrates benefits on an alternative backbone. On a sampled production subgraph, SURF improves all three evaluated backbones in MRR and Hits@1 across three tasks, with over one billion scored pairs per backbone. The serving comparison illustrates how cached-vector scoring avoids the online graph computation required by the evaluated ULTRA pipeline. These results support structural adaptation as a practical way to strengthen graph foundation models for retrieval without moving graph reasoning into the online scoring path.

**Limitations.** Cached signatures enlarge node representations and retrieval indexes, and graph updates require refreshing affected signatures. Their construction also requires efficient access to the observed message graph. Target-domain adaptation uses training data and may require schema-specific input adapters; task-local structural fitting does not eliminate transfer limitations inherited from the shared backbone. SURF therefore trades additional offline computation and storage for low-latency structural scoring, making signature compression and incremental updates important directions for future work.

### AI USE STATEMENT

In this work, we used generative AI tools for literature search, particularly related to information retrieval evaluation techniques, and related work on Universal Graph Foundation Models for Large Scale Retrieval. We also made use of generative AI tools for polishing the manuscript: improving the writing of the introduction and abstract, removing redundancies, shortening sentences, and fixing typos, as well as checking for mathematical notation consistency. Finally, we used AI-based methods for anonymization of the code made available.

We have reviewed all AI-assisted work. All works suggested by AI models were carefully read, then manually incorporated by the authors (there are no citations introduced by generative AI methods in the manuscript). We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Louis Airale, Antonio Longa, Mattia Rigon, Andrea Passerini, and Roberto Passerone. Simple path structural encoding for graph transformers. *arXiv preprint arXiv:2502.09365*, 2025.

Anirudhan Badrinath, Alex Yang, Kousik Rajesh, Prabhat Agarwal, Jaewon Yang, Haoyu Chen, Jiajing Xu, and Charles Rosenberg. OmniSage: Large Scale, Multi-Entity Heterogeneous Graph Representation Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 4261–4272, August 2025. doi: 10.1145/3711896.3737253. URL http://arxiv.org/abs/2504.17811.

Ivana Balažević, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. 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. 5185–5194, 2019.

Maya Bechler-Speicher, Yoel Gottlieb, Andrey Isakov, David Abensur, Ami Tavory, Daniel Haimovich, Ido Guy, and Udi Weinsberg. Billion-Scale Graph Foundation Models, 2026. URL https://arxiv.org/abs/2602.04768.

Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. Pure message passing can estimate common neighbor for link prediction. *Advances in Neural Information Processing Systems*, 37:73000–73035, 2024a.

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

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 31598–31619, 2024.

Jinquan Hang, Zhiqing Hong, Xinyue Feng, Guang Wang, Guang Yang, Feng Li, Xining Song, and Desheng Zhang. Paths2pair: Meta-path based link prediction in billion-scale commercial heterogeneous graphs. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 5082–5092, 2024.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous Graph Transformer. In *Proceedings of The Web Conference 2020*, pp. 2704–2710, Taipei Taiwan, April 2020. ACM. ISBN 978-1-4503-7023-3. doi: 10.1145/3366423.3380027. URL https://dl.acm.org/doi/10.1145/3366423.3380027.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. Prodigy: Enabling in-context learning over graphs. *Advances in Neural Information Processing Systems*, 36:16302–16317, 2023.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. *Commun. ACM*, 65(7):75–83, June 2022. ISSN 0001-0782. doi: 10.1145/3535335. URL https://doi.org/10.1145/3535335.

Veronica Lachi, Francesco Ferrini, Antonio Longa, Bruno Lepri, Andrea Passerini, and Manfred Jaeger. Bridging theory and practice in link representation with graph neural networks. *Advances in Neural Information Processing Systems*, 38:122617–122646, 2025.

Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, April 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

Sohir Maskey, Ali Parviz, Maximilian Thiessen, Hannes Stärk, Ylli Sadikaj, and Haggai Maron. Generalized laplacian positional encoding for graph representation learning. *arXiv preprint arXiv:2210.15956*, 2022.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer, January 2023. URL http://arxiv.org/abs/2205.12454.

Junhua Shi, Qingyun Sun, Haonan Yuan, and Xingcheng Fu. Sa$^2$gfm: Enhancing robust graph foundation models with structure-aware semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15716–15724, 2026.

Yu Song, Zhigang Hua, Harry Shomer, Yan Xie, Jingzhe Liu, Bo Long, and Hui Liu. A scalable pretraining framework for link prediction with efficient adaptation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2621–2632, 2025.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *International conference on learning representations*, volume 2024, pp. 14573–14589, 2024a.

Zehong Wang, Zheyuan Zhang, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. Gft: Graph foundation model with transferable tree vocabulary. *Advances in neural information processing systems*, 37: 107403–107443, 2024b.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. Towards graph foundation models: Learning generalities across graphs via task-trees. *arXiv preprint arXiv:2412.16441*, 2024c.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

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, 2026.

Hongxia Yang. Aligraph: A comprehensive graph neural network platform. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3165–3166, 2019.

Minji Yoon, John Palowitch, Dustin Zelle, Ziniu Hu, Ruslan Salakhutdinov, and Bryan Perozzi. Zero-shot transfer learning within a heterogeneous graph via knowledge transfer networks. *Advances in Neural Information Processing Systems*, 35:27347–27359, 2022.

Juzheng Zhang, Lanning Wei, Zhen Xu, and Quanming Yao. Heuristic learning with graph neural networks: A unified framework for link prediction. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 4223–4231, 2024a.

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

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021.

Tianyi Zhang, Haoteng Yin, Rongzhe Wei, Pan Li, and Anshumali Shrivastava. Learning scalable structural representations for link prediction with bloom signatures. In *Proceedings of the ACM Web Conference 2024*, pp. 980–991, 2024b.

Jianan Zhao, Hesham Mostafa, Michael Galkin, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphany: A foundation model for node classification on any graph. *arXiv preprint arXiv:2405.20445*, 29, 2024.

Jiajun Zhu, Ying Chen, Peihao Wang, Yixuan He, Pan Li, Aditya Akella, and Zhangyang Wang. When do graph foundation models transfer? a data-centric theory. *arXiv preprint arXiv:2605.29828*, 2026.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in neural information processing systems*, 34:29476–29490, 2021.

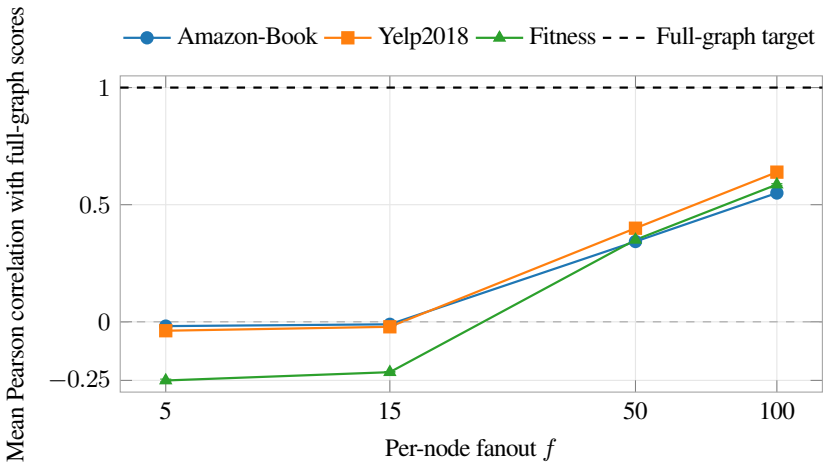


Figure 2: **Neighborhood sampling distorts structural pair scores.** For the same 3,000 held-out positive pairs in each dataset, we compute structural signatures using either the full observed graph or a sampled graph that retains at most $f$ uniformly sampled neighbors per node. We compare the resulting signature dot-product scores across pairs using Pearson correlation. Smaller fanouts correspond to more aggressive sampling. The dashed line at one denotes perfect correlation with the full-graph scores.

| | Adaptation-positive budget | | | | | Subsampling repetition | | Hash seed | | Precision | Normalization | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | 50 | 100 | 250 | 500 | 1,000 | 1 | 2 | 1 | 2 | FP16 | None | Robust |
| HGT | +0.48 ±0.85 | +0.11 ±0.92 | −0.27 ±1.01 | −0.13 ±0.21 | 0.00 ±0.00 | +0.20 ±1.35 | +0.23 ±1.03 | +0.09 ±0.38 | −0.23 ±0.45 | +0.06 ±0.05 | −2.54 ±1.44 | −1.56 ±2.05 |
| GraphBFF | +0.55 ±0.86 | +0.29 ±0.87 | −0.46 ±1.09 | −0.08 ±0.22 | 0.00 ±0.00 | +0.29 ±1.31 | +0.14 ±0.93 | +0.27 ±0.26 | −0.55 ±0.68 | +0.06 ±0.04 | −1.97 ±1.07 | −1.29 ±1.83 |

Table 6: Task-level SURF adaptation robustness at 4,096 evaluation negatives. Each entry is the paired mean ΔMRR in percentage points, with the 95% confidence-interval half-width shown underneath. Blue and red indicate positive and negative mean changes, respectively, whose 95% confidence intervals exclude zero; entries whose intervals contain zero are shown in black. Vertical lines separate the five types of changes evaluated: adaptation-positive budget, subsampling repetition, hash seed, numerical precision, and normalization. Budget interventions use the 1,000-positive cell as reference, while subsampling repetitions use the matched 100-positive cell. Most interventions are statistically unresolved at this sample size; the consistent exceptions are the small positive FP16 changes and the degradation from removing RMS normalization.

## A Additional Structural Diagnostics and Ablations

This appendix provides additional diagnostics of structural signatures and their adaptation. We examine how neighborhood sampling affects retained structural information (Figure 2), sensitivity to structural-fitting choices (Table 6), and the trade-off between signature width and prediction quality (Figure 3).

**Effect of neighborhood sampling.** We investigate whether structural scores computed from sampled neighborhoods preserve the variation across node pairs observed when using the full graph. We fix 3,000 held-out positive pairs per dataset on Amazon-Book, Yelp2018, and Fitness and compute one- and two-hop raw log-count signatures (Zhang et al., 2024b) on the full observed graph and on graphs sampled with fanouts $f \in \{5,15,50,100\}$. Figure 2 reports Pearson correlations between full-graph and sampled-graph scores across the fixed pairs, averaged over hop combinations and sampling seeds. At fanout 15, the mean correlation is negative for all three datasets; increasing the fanout to 100 improves it to only 0.55–0.64. Thus, sampling changes more than the scale of structural scores: it also distorts their variation across pairs. This motivates computing structural signatures from the full observed graph rather than the neighborhoods sampled for backbone training. This diagnostic does not directly measure final link-ranking quality.

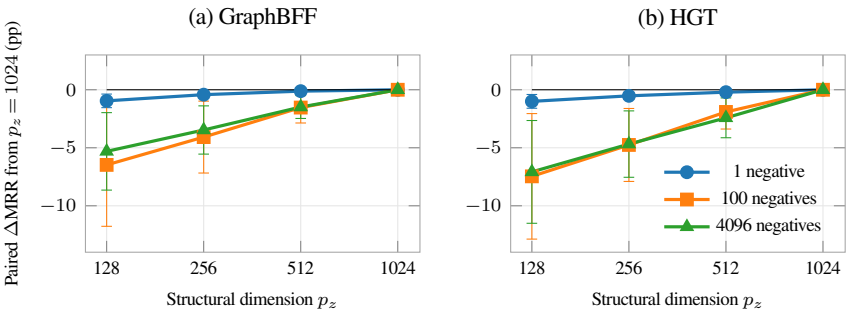


Figure 3: **Signature-width trade-off for SURF with HGT and GraphBFF backbones.** Values are paired MRR changes relative to $p_z = 1024$, averaged over 28 dataset–task–seed cases; error bars show 95% confidence intervals. Reducing signature width incurs larger losses with 100 and 4,096 negatives than with one negative.

**Robustness of structural adaptation.** Table 6 reports paired MRR changes under variations in structural-fitting configuration, evaluated with 4,096 negatives per positive. Varying the adaptation budget between 50 and 1,000 training positives produces no consistent improvement, with the reported confidence intervals overlapping with zero. Repeated subsampling likewise produces small, statistically unresolved changes. In contrast, removing RMS score calibration decreases MRR by 2.54 percentage points for HGT and 1.97 points for GraphBFF. These results support calibrating the score components before fusion, while providing little evidence that larger fitting budgets improve performance within the tested range.

**Signature width.** Figure 3 examines structural signature widths $p_z \in \{128, 256, 512, 1024\}$, keeping the backbones frozen and refitting the structural parameters and fusion weights at each width. Smaller signatures reduce storage requirements but incur larger prediction losses in more demanding candidate pools. Reducing $p_z$ from 1,024 to 128 decreases mean MRR by 7.08 percentage points with HGT and 5.31 points with GraphBFF when evaluated against 4,096 negatives, compared with approximately one point for either backbone with one negative. These results expose a storage–quality trade-off that becomes more pronounced as the candidate pool grows.

**BioKG per-relationship results.** Table 7 compares dataset-wide and relationship-specific structural scores for representative BioKG relationships.

| **Relation** | $\mathbf{K}_d$ | $\mathbf{K}_{d,r}$ | **Gain (pp)** |
|---|---|---|---|
| drug–drug personality disorder | 1.79 | **33.51** | +31.71 |
| drug–drug monogenic disease | 30.25 | **50.40** | +20.14 |
| drug–drug structural similarity | 12.26 | **32.28** | +20.02 |
| protein–protein expression | 0.27 | **18.04** | +17.76 |
| drug–drug substance-related disorder | 2.05 | **9.55** | +7.49 |
| protein–protein reaction | 4.51 | **10.88** | +6.37 |
| disease–protein | 2.59 | **5.97** | +3.38 |
| drug–disease | 6.48 | **9.86** | +3.37 |

Table 7: **Relationship-specific structural comparisons improve selected BioKG relationships.** Structural-only MRR (%) for representative relationships in `ogbl-biokg`, comparing a dataset-wide kernel $\mathbf{K}_d$ with relationship-specific kernels $\mathbf{K}_{d,r}$. Gains are reported in percentage points (pp). Scores and gains are converted from the reported source values; rounding can produce a 0.01-point difference between a reported gain and the difference of the displayed scores.

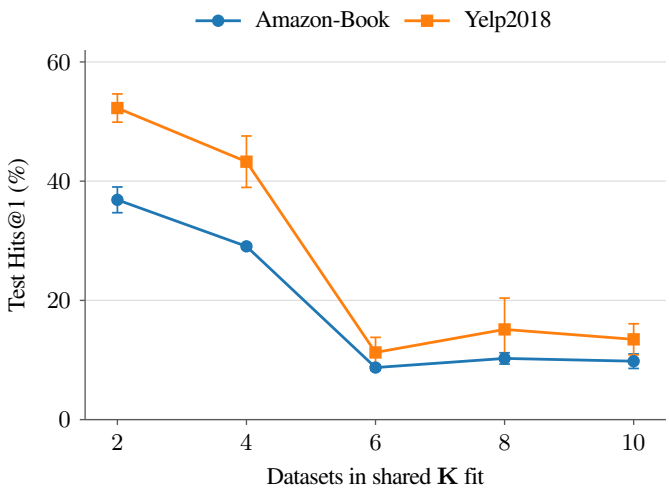


Figure 4: Shared-kernel fitting on growing dataset cohorts, with signatures and evaluation pairs fixed. Error bars are standard deviations over three runs.

## B Negative Transfer Experiment

**Effect of sharing.** Figure 4 isolates the effect of cross-dataset sharing by fitting a single structural kernel $\mathbf{K} \in \mathbb{R}^{2\times 2}$ jointly across cohorts of 2, 4, 6, 8, and 10 datasets, starting with Amazon-Book and Yelp2018, while keeping precomputed one- and two-hop signatures and evaluation pairs fixed. The structural score is $s_d^{\mathrm{str}}(u,r,v;\mathbf{K}) = \sum_{i,j\in\{1,2\}} K_{ij}\langle \mathbf{z}_u^{(i)}, \mathbf{z}_v^{(j)}\rangle$. Each fit optimizes the four kernel entries using pairwise logistic-ranking loss, with up to 1,000 training positives per task and 16 negatives per positive. We use Adam for 1,000 steps and select checkpoints using up to 500 validation positives with 100 negatives each, prioritizing Hits@1, then MRR, then pairwise loss. Curves report mean test Hits@1 over three runs, with error bars showing the corresponding standard deviations. As the cohort grows from two to ten datasets, Hits@1 falls from 36.87% to 9.80% on Amazon-Book and from 52.27% to 13.47% on Yelp2018. These results motivate task-local structural fitting rather than forcing a common kernel across domains.

## C Additional Link Prediction Results

Table 8 reports Hits@1 and MRR for the baselines and SURF backbone variants on test sets containing equal numbers of positive and negative pairs. Results cover the ten train/test datasets and six held-out datasets, complementing the large-candidate retrieval evaluation in the main text.

## D Scaling Diagnostics Beyond GraphBFF

**Question.** We test whether increasing GraphBFF's model size, training data, or optimization budget closes the performance gap between backbone-only and structurally augmented scoring. At each checkpoint, we compare the backbone alone with the same frozen backbone combined with a fixed relation-conditioned structural expert.

**Protocol.** All runs use GraphBFF pretrained on the pretraining dataset collection, with a fixed seed, bf16 training, batch size 128, two-hop fanout [10,10], and the same task schedule. We vary three axes separately: model size across T1–T6 (30.4M–662.1M trainable parameters), training data for T5 (1%, 10%, and 100%, corresponding to 0.38M, 3.8M, and 38.1M distinct edges), and optimization budget (5k, 15k, and 25k steps). A three-learning-rate pilot selects the learning rate before each selected run resumes to its target checkpoint. The scaling experiments use full-adjacency transductive reconstruction. Every checkpoint is evaluated on the same materialized 4,096-negative corpus, using both backbone-only and relation-conditioned fusion scoring.

| Dataset | AnyGraph | | GraphGPS | | Ultra | | GraphBFF | | HGT | | SURF (GraphBFF) | | SURF (HGT) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) | H@1 (%) | MRR (%) |
| *Train–test datasets* | | | | | | | | | | | | | | |
| amazon-book | 11.50±6.78 | 26.93±5.48 | 3.24±0.00 | 17.61±0.00 | 42.41±0.17 | 56.36±0.27 | 0.70±0.24 | 12.36±0.90 | 2.86±0.00 | 19.80±0.00 | **83.78±0.00** | **90.32±0.00** | **83.66±0.05** | **90.25±0.03** |
| fitness | 30.59±8.28 | 45.89±7.03 | 83.94±0.00 | 90.71±0.00 | 97.89±0.07 | 98.79±0.04 | 95.41±0.25 | 97.46±0.17 | 79.30±0.66 | 87.49±0.55 | **99.05±0.03** | **99.49±0.02** | **99.21±0.06** | **99.59±0.03** |
| goodreads | 19.87±3.10 | 41.55±2.91 | 87.83±0.00 | 93.11±0.00 | 96.11±0.30 | 97.63±0.20 | 91.27±0.41 | 95.06±0.23 | 80.83±0.38 | 88.77±0.24 | **98.90±0.13** | **99.39±0.08** | **99.02±0.00** | **99.96±0.00** |
| movielens-10m | 2.33±0.41 | 13.15±0.33 | 0.00±0.00 | 3.53±0.00 | 22.21±1.32 | 41.21±1.28 | 0.10±0.04 | 3.60±0.06 | 0.01±0.00 | 3.66±0.00 | **58.72±0.13** | **70.55±0.16** | **58.58±0.28** | **70.39±0.34** |
| ogbl-biokg | 10.13±0.71 | 24.75±1.08 | 13.00±0.00 | 28.51±0.00 | **80.20±0.07** | **87.77±0.02** | 37.68±4.86 | 54.38±4.31 | 52.09±13.49 | 65.79±10.02 | **78.07±0.37** | **83.31±0.25** | 77.24±0.19 | 82.67±0.14 |
| rel-amazon | 68.51±3.07 | 84.25±1.54 | 84.39±0.00 | 92.19±0.00 | 72.58±0.14 | 86.29±0.07 | 86.45±0.66 | 93.22±0.33 | **87.57±0.10** | **93.79±0.05** | 84.29±2.34 | 92.15±1.17 | **86.56±0.11** | **93.28±0.05** |
| rel-hm | 77.82±0.72 | 88.91±0.36 | 91.58±0.00 | 95.79±0.00 | 88.90±0.25 | 94.45±0.12 | **92.79±0.27** | **96.39±0.14** | **92.79±0.23** | **96.40±0.12** | 90.95±0.29 | 95.48±0.14 | 90.43±0.25 | 95.22±0.12 |
| rel-stack | 54.17±3.22 | 77.08±1.61 | 62.65±0.00 | 81.32±0.00 | 65.70±0.07 | 82.85±0.03 | 69.03±0.91 | 84.51±0.45 | **74.66±1.46** | **87.33±0.73** | 71.70±0.07 | 85.85±0.03 | **76.97±0.56** | **88.48±0.28** |
| rel-trial | 61.81±17.56 | 80.90±8.78 | 90.25±0.00 | 95.12±0.00 | 82.72±0.32 | 91.36±0.16 | 91.82±0.50 | 95.91±0.25 | **93.12±0.14** | **96.56±0.07** | 92.14±0.10 | 96.07±0.05 | **93.24±0.14** | **96.62±0.07** |
| yelp2018 | 10.94±4.14 | 27.66±3.28 | 2.99±0.00 | 18.53±0.00 | 53.49±0.14 | 68.32±0.11 | 0.28±0.01 | 11.72±0.02 | 1.77±0.00 | 18.68±0.00 | **90.49±0.34** | **94.74±0.19** | **90.47±0.34** | **94.73±0.19** |
| *Mean Rank* | 6.10 | 6.10 | 5.10 | 5.30 | 3.90 | 3.90 | 4.45 | 4.60 | 4.05 | 3.70 | **2.40** | **2.40** | **2.00** | **2.00** |
| *Fine-tuned / held-out datasets* | | | | | | | | | | | | | | |
| gowalla | 11.90±0.90 | 31.00±0.60 | 5.54±0.00 | 22.09±0.00 | 69.16±0.58 | 79.36±0.39 | 1.20±0.05 | 15.80±0.10 | 4.91±0.00 | 24.46±0.00 | **94.33±0.09** | **97.01±0.05** | **94.31±0.12** | **97.00±0.06** |
| amazon-text | 91.60±0.10 | 95.03±0.07 | 63.34±0.00 | 76.85±0.00 | 67.86±0.36 | 80.39±0.28 | 64.00±2.00 | 77.00±1.00 | 59.00±0.80 | 73.20±0.50 | **96.94±0.60** | **98.35±0.32** | **97.02±0.48** | **98.38±0.26** |
| steam-text | 78.40±0.50 | 86.60±0.20 | 21.85±0.00 | 44.59±0.00 | 81.42±1.01 | 88.96±0.60 | 33.00±1.00 | 52.00±1.00 | 43.00±1.00 | 61.40±0.60 | **91.10±3.20** | **95.09±1.87** | **91.28±3.24** | **95.21±1.88** |
| rel-avito | 49.00±2.00 | 74.00±1.00 | 81.65±0.00 | 90.83±0.00 | 48.12±24.07 | 74.06±12.04 | 84.90±0.80 | 92.50±0.40 | 84.00±2.00 | 92.10±0.80 | **86.55±0.66** | **93.27±0.33** | **85.42±1.40** | **92.71±0.70** |
| photo | 97.79±0.05 | 98.82±0.03 | 86.03±0.00 | 91.92±0.00 | 97.15±0.05 | 98.43±0.02 | 88.70±0.20 | 93.60±0.10 | 81.90±0.20 | 89.28±0.09 | **98.21±0.04** | **99.05±0.03** | **98.65±0.06** | **99.30±0.03** |
| ogbl-wikikg2 | 52.00±2.00 | 71.00±1.00 | 19.26±0.00 | 45.30±0.00 | **96.99±0.03** | **98.19±0.01** | 18.20±0.70 | 43.50±0.60 | 50.00±20.00 | 70.00±10.00 | 95.00±0.70 | 96.95±0.44 | **95.54±0.34** | **97.31±0.23** |
| *Mean Rank* | 4.00 | 4.17 | 5.83 | 6.00 | 3.67 | 3.50 | 5.50 | 5.50 | 5.67 | 5.50 | **1.83** | **1.83** | **1.50** | **1.50** |

Table 8: Results on the ten train–test datasets and six fine-tuned/held-out datasets under the balanced-negatives protocol. H@1 and MRR are reported as percentages. The five backbone baselines appear before the divider; the corresponding SURF variants appear after it. Values are mean ± sample standard deviation over the supplied runs. The first- and second-place reported means for each metric on each dataset are highlighted in **blue** and **red**, respectively. Ties are highlighted jointly and consume the corresponding places. Mean ranks are computed separately for each dataset group from the displayed dataset-level means across the seven methods (lower is better); tied methods share the average of the ranks they occupy.

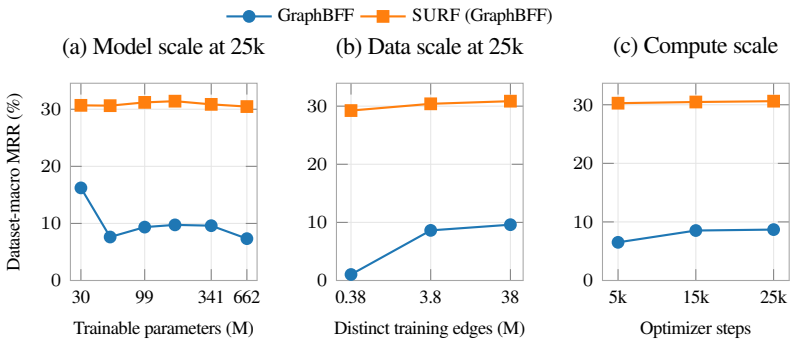


Figure 5: **Scaling GraphBFF alone does not close the fixed structural-fusion gap in this diagnostic.** Dataset-macro MRR on the shared 4,096-negative evaluation corpus when scaling (a) model size at 25k steps, (b) T5 training data at 25k steps, and (c) optimizer steps. SURF uses the corresponding frozen GraphBFF backbone plus its fixed relation-conditioned structural expert.

**Results.** Figure 5 shows that scaling GraphBFF alone does not close the structural-fusion gap within the tested ranges. Across the model, data, and optimization sweeps, SURF (GraphBFF) achieves 29.2–31.4% dataset-macro MRR, compared with 1.0–16.2% for the backbone alone. Increasing training from 5k to 25k steps raises backbone MRR from 6.51% to 8.69%, but SURF (GraphBFF) reaches 30.61% at 25k steps, retaining a 21.92-percentage-point advantage. Figure 6 reports the corresponding validation-BCE curves.

## E Experiment Details

This appendix records the implementation, training, and evaluation settings that complement the main experimental setup. The model and retrieval equations are defined in Section 4.2; here we focus on their experimental configuration.

**Software stack.** We implemented the pipeline in Python 3.11 and PyTorch (Paszke et al., 2019). Academic message-passing backbones use PyTorch Geometric (Fey & Lenssen, 2019), while AnyGraph consumes precomputed dense SGC features and edge batches (Xia & Huang, 2026). For the production graph, an in-memory graph store backed by GraphLearn-for-PyTorch supports partitioned graph storage and distributed neighborhood sampling (Yang, 2019). Training and evaluation jobs run in containerized environments on Google Cloud Vertex AI. Dataset materialization and large-scale signature computation use BigQuery, with intermediate and final artifacts stored in Google Cloud Storage.

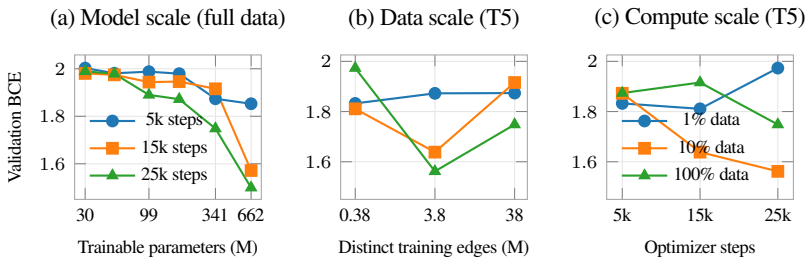


Figure 6: **Validation-loss scaling diagnostics.** Validation BCE as model size, distinct training edges, and optimizer steps vary. These single-seed curves describe optimization behavior; they are not fitted power laws or estimates of a universal scaling law.

**Datasets and supervision.** The pretraining collection contains ten datasets: `amazon-book`, `fitness`, `goodreads`, `movielens-10m`, `ogbl-biokg`, `rel-amazon`, `rel-hm`, `rel-stack`, `rel-trial`, and `yelp2018`. We additionally evaluate on six held-out datasets: `amazon-text`, `gowalla`, `ogbl-wikikg2`, `photo`, `rel-avito`, and `steam-text`. RelBench tasks retain their temporal supervision and seed timestamps, while `ogbl-biokg` uses the official OGB split and all 51 original relationship labels. We use deterministic training, validation, and test partitions. For the academic link-prediction experiments, validation and test positives are excluded from the message graph. During multi-dataset training, tasks are sampled with probability proportional to their number of training positives.

Table 10 summarizes the materialized datasets used in our experiments, reporting message-edge records and positive supervision records separately. For RelBench, the positive counts correspond to the bounded temporal samples used in our experiments, rather than all target rows available in the original benchmark at the corresponding cutoff.

**Backbone training.** We train the academic backbones with AdamW for 25,000 pretraining steps and 10,000 adaptation steps on held-out datasets, reporting results over two seeds. AnyGraph, GraphBFF, HGT, and GraphGPS use fixed-fanout sampling $[10,10]$. ULTRA uses its sampled pairwise edge-scoring implementation; large-candidate results are available for `rel-stack` and `amazon-text`, while the remaining datasets are estimated to require more than 24 hours.

**Structural signatures.** We precompute one- and two-hop log-count Bloom signatures from the full observed message graph, following Equation (1). Unless stated otherwise, each hop has dimension 1,024, with each node identifier hashed into one bucket. We apply $\log(1+x)$ element-wise after each propagation and use the resulting signatures to construct both the raw and per-hop $\ell_2$-normalized structural branches.

**Signature intuition and construction.** For an uncompressed neighborhood indicator $\mathbf{b}_v$, $[\mathbf{b}_v]_w = \mathbf{1}\{w \in \mathcal{N}(v)\}$ and $\mathbf{b}_u^\top \mathbf{b}_v = |\mathcal{N}(u) \cap \mathcal{N}(v)|$. Thus node-level summaries can recover common-neighbor counts without fetching neighborhoods during scoring, although compression need not preserve this exactly. In the log-count construction, the first aggregation counts neighboring identifiers in each bucket before logarithmic compression; the second aggregates the resulting one-hop signatures and applies the logarithm again. Keeping channels separate permits same-scale and cross-scale comparisons between query and candidate neighborhoods. A sparse implementation of the signature recurrence requires $O(H(|E|+|V|)p_z)$ time for all nodes, including neighborhood aggregation and elementwise transformations. For fixed $H$ and $p_z$, this is linear in graph size.

**Structural kernel and fusion fitting.** For a training positive $(u,r,v^+)$ and a sampled negative candidate $v^-$, define the score difference for component $e \in \{h,\text{raw},\text{L2}\}$ as

$$\Delta^e_{d,r} = s^e_d(u,r,v^+) - s^e_d(u,r,v^-).$$

With the backbone and signatures frozen, we independently fit the four kernel coefficients of each structural branch $e \in \{\text{raw},\text{L2}\}$ by minimizing the pairwise logistic (softplus) objective

$$\mathcal{L}^e_{d,r} = \mathbb{E}_{(u,v^+,v^-)}\left[\log\left(1+\exp(-\Delta^e_{d,r})\right)\right],$$

| Stage | Scope | Time |
|---|---|---|
| Graph preparation | 3.69 billion message arcs | 582 s |
| Two-hop signature computation | Approximately 50,000 nodes | 42–43 s |
| **Total (sum of stages)** | | **624–625 s** |

Table 9: **Offline preprocessing on the industrial graph.** Graph preparation constructs the message graph used by the experiment, including its edge exclusions. Signature computation materializes the requested nodes' one- and two-hop signatures using complete neighborhoods. The total sums the recorded stage times; graph preparation is reusable for a fixed snapshot.

where the expectation is over the task's training positives and sampled negatives. For the canonical component experiments, each task uses up to 1,000 training positives and 500 validation positives, with 16 training negatives and 4,096 validation negatives per positive. Each kernel is optimized for 1,000 steps using Adam, batch size 16,384, learning rate $3\times10^{-2}$, and weight decay $10^{-5}$, with validation every 25 steps.

After fitting the structural kernels, we compute a calibration scale for each component $e\in\{h,\mathrm{raw},\mathrm{L2}\}$:

$$\rho^{e}_{d,r}=\sqrt{\mathbb{E}_{(u,v^{+},v^{-})}\left[(\Delta^{e}_{d,r})^{2}\right]}.$$

With the kernels and calibration scales fixed, we optimize the three fusion logits for 1,000 steps, using the same batch size, learning rate, weight decay, and validation interval as the structural kernels. The fitted parameters are compiled into the query representation using Equation (5).

**Evaluation protocols.** We report Hits@$k$ and MRR. We construct a balanced test set containing equal numbers of positive and negative pairs and score every pair. Each positive is ranked against the shared pool of all sampled test negatives, irrespective of source node. We compute MRR and Hits@$k$ from these ranks. The sampled-full-candidate protocol uniformly samples 10% of test positives and 10% of compatible candidates without replacement. We estimate full-candidate Hits@$k$ and MRR using the minimum-bias correction of Krichene & Rendle (2022). Full-candidate runs instead evaluate all test positives against the full candidate population. Candidate scores are streamed in blocks of 200,000 using FP16, with query batches of 128. We report means and sample standard deviations across two training seeds.

**Compute environment.** Most academic experiments use four NVIDIA A100 GPUs with 40GB memory each. ULTRA uses one NVIDIA A100 80GB GPU for training and batched candidate-score compilation. Structural kernel and fusion-weight fitting can run on a single CPU worker.

**Separate industrial-scale preprocessing benchmark.** In this separate benchmark, graph preparation produces 3.69 billion message arcs. We compute width-1,024 one- and two-hop signatures using full neighborhood information for approximately 50,000 validation and test nodes, not every node. Table 9 reports 582 seconds for graph preparation and 42–43 seconds for signature computation, or approximately 10.4 minutes combined. For a fixed snapshot, the prepared graph and signatures are reusable across backbone checkpoints and relationship-specific fitting, keeping traversal outside online retrieval.

### E.1 Production-subgraph evaluation

**Data and prediction tasks.** We sample a production subgraph from a large online platform and evaluate three link-prediction tasks, anonymized as Tasks A, B, and C. Confidentiality requirements prevent disclosure of task identities and absolute metric values; we report only relative improvements over each backbone. GraphBFF, HGT, and GraphGPS use the same subgraph and supervision splits. Validation and test positives are held out from base-model weight optimization, and each augmented model reuses its corresponding backbone checkpoint.

**Sampled ranking protocol.** For each task, we evaluate 90,000 test-positive groups, each containing one positive and 4,092 fixed negatives. Every backbone and scoring variant uses the same groups: 270,000 positive groups and 1,105,110,000 scored pairs per backbone. The backbone-only semantic decoder is an inner product, and negatives tied with a positive do not rank ahead of it. MRR and Hits@$k$ are computed directly from these sampled ranks, without the sampled-metric corrections used in the academic

Table 10: Dataset statistics of the materialized experimental artifacts. Message edges are directed records in the stored message graph before split-specific leakage masking. Positive links sum the train, validation, and test supervision records used by our protocol.

| Dataset | # Nodes | # Message edges | # Positive links | Features | Task |
|---|---|---|---|---|---|
| *Train–test datasets* | | | | | |
| amazon-book | 144,242 | 2,984,108 | 2,984,108 | None | Recommendation / LP |
| fitness | 173,055 | 1,773,594 | 2,672,772 | Text (768) | Link prediction |
| goodreads | 676,084 | 8,582,306 | 9,482,056 | Text (768) | Link prediction |
| movielens-10m | 80,555 | 10,000,054 | 10,000,054 | None | Recommendation / LP |
| ogbl-biokg | 93,773 | 5,088,434 | 5,088,434 | None | Knowledge Graph Completion / LP (51 relations) |
| rel-amazon | 15,000,713 | 50,578,032 | 5,843,968 | Mixed tabular | LP: user-item-purchase |
| | | | 4,198,400 | | LP: user-item-rate |
| | | | 2,552,832 | | LP: user-item-review |
| rel-hm | 16,664,809 | 60,749,148 | 3,992,576 | Mixed tabular | LP: user-item-purchase |
| rel-stack | 4,247,264 | 11,625,774 | 22,822 | Mixed tabular | LP: user-post-comment |
| | | | 6,339 | | LP: post-post-related |
| rel-trial | 5,434,924 | 15,335,102 | 31,744 | Mixed tabular | LP: condition-sponsor-run |
| | | | 720,896 | | LP: site-sponsor-run |
| yelp2018 | 69,716 | 1,561,406 | 1,561,406 | None | Recommendation / LP |
| *Fine-tuned / held-out datasets* | | | | | |
| amazon-text | 20,332 | 200,860 | 200,860 | Text (1,536) | Recommendation / LP |
| gowalla | 70,839 | 1,027,370 | 1,027,370 | None | Recommendation / LP |
| ogbl-wikikg2 | 2,500,604 | 17,137,181 | 22,491,710 | None | Knowledge Graph Completion / LP (535 relations) |
| photo | 48,362 | 500,939 | 781,295 | Text (768) | Link prediction |
| rel-avito | 20,679,117 | 86,108,942 | 150,528 | Mixed tabular | LP: user-ad-visit |
| steam-text | 28,547 | 525,922 | 525,922 | Text (1,536) | Recommendation / LP |

Counts are taken from the exact union-index manifests used for training and evaluation. Message-edge counts are materialized directed rows, not necessarily unique unordered edges. Positive-link counts are supervision instances and may therefore differ from the number of message edges. For RelBench, they are the bounded temporal samples used by our experiments, not the full target-row totals reported by the source benchmark. The ogbl-biokg row uses the corrected relation-aware overlay with the official split and 51 relations.

Datasets listed with no raw node features use a one-dimensional constant placeholder where required by the encoder interface; this placeholder is not treated as an informative feature. Fitness and Photo provide source node-classification labels, but our experiments use their graph edges for link prediction. Mixed-tabular feature dimensionality varies by node type and encoding.

benchmark. Aggregate metrics are macro-averages over the three equally sized task test sets; relative improvements are computed from these macro-averages.

**Structural adaptation and evaluation.** We freeze backbone embeddings and fit the structural operators and score-calibration scales using 50,000 training-positive groups per task. An independent set of 20,000 validation-positive groups per task selects the fusion configuration. Both splits use 4,092 negatives per positive. The implementation uses hop-separated log-count signatures with raw and L2-normalized structural branches. Test metrics are computed only after validation selection. Table 11 gives the per-task relative improvements.

## F Proof of Theorem 1

*Proof.* For each linear signature channel $i = 0, \ldots, H$, the identifier matrix $R$ maps an uncompressed signature to a fixed-width signature:

$$\mathbf{z}_u^{(i)} = R^\top B_i^\top \mathbf{e}_u,$$

| Backbone | MRR lift (%) | Hits@1 lift (%) | Hits@10 lift (%) |
|---|---|---|---|
| *Task A* | | | |
| GraphBFF | +29.2 | +129.7 | +14.3 |
| HGT | +89.7 | +284.5 | +84.9 |
| GraphGPS | +10.3 | +76.5 | -2.3 |
| *Task B* | | | |
| GraphBFF | +38.3 | +123.7 | +44.4 |
| HGT | +162.7 | +488.8 | +292.3 |
| GraphGPS | +9.7 | +65.5 | +7.9 |
| *Task C* | | | |
| GraphBFF | +200.0 | +608.8 | +193.9 |
| HGT | +361.9 | +590.4 | +631.9 |
| GraphGPS | +177.0 | +531.8 | +167.8 |

Table 11: Per-task relative gains from SURF over each backbone. Task identities and absolute scores are withheld for confidentiality. Lift is $100(m_{\text{SURF}}/m_{\text{base}}-1)$ for metric $m$; blue/red denote gains/decreases, not statistical significance. Each task uses 90,000 test-positive groups with 4,092 fixed negatives per positive. Values are single-run point estimates.

where $\mathbf{e}_u$ is the $u$-th standard basis vector. Stack these signatures as rows of the bank $\overline{\mathbf{Z}}_u \in \mathbb{R}^{(H+1)\times p_z}$, including channel 0, and let $\mathbf{K}_{d,r} \in \mathbb{R}^{(H+1)\times(H+1)}$ be the channel-mixing operator. Expanding the Frobenius inner product gives

$$\begin{aligned}\left\langle \mathbf{K}_{d,r}^\top \overline{\mathbf{Z}}_u, \overline{\mathbf{Z}}_v \right\rangle_F &= \sum_{i=0}^{H}\sum_{j=0}^{H} (\mathbf{K}_{d,r})_{ij} \left\langle \mathbf{z}_u^{(i)}, \mathbf{z}_v^{(j)} \right\rangle \\ &= \sum_{i=0}^{H}\sum_{j=0}^{H} (\mathbf{K}_{d,r})_{ij} \mathbf{e}_u^\top B_i R R^\top B_j^\top \mathbf{e}_v \\ &= \sum_{i=0}^{H}\sum_{j=0}^{H} (\mathbf{K}_{d,r})_{ij} (B_i B_j^\top)_{uv},\end{aligned}$$

where the final equality uses $RR^\top = I$. This proves Equation (6).

Let $W_g$ be diagonal with nonnegative entries $g(w)$. Choose uncompressed signatures $B_i = AW_g^{1/2}$, set $(\mathbf{K}_{d,r})_{ii} = 1$, and set all other coefficients to zero. The resulting score is

$$(B_i B_i^\top)_{uv} = (AW_g A^\top)_{uv} = \sum_w A_{uw} A_{vw} g(w).$$

For distinct nodes in an undirected graph, choosing $g(w) = 1$, $g(w) = 1/\deg(w)$, or $g(w) = 1/\log\deg(w)$ recovers Common Neighbors, Resource Allocation, or Adamic–Adar, respectively. Undefined degree-based weights can be set to zero, since nodes of degree zero or one cannot be common neighbors of distinct nodes.

For finite random-walk scores, let $P$ be a transition matrix and choose $B_i = P^i$ for $i = 0,\ldots,H$, so that $B_0 = I$. Set $(\mathbf{K}_{d,r})_{i0} = \beta_i$ and all remaining coefficients to zero. Then

$$\sum_{i=0}^{H} (\mathbf{K}_{d,r})_{i0} (B_i B_0^\top)_{uv} = \sum_{i=0}^{H} \beta_i (P^i)_{uv}.$$

Because $\mathbf{K}_{d,r}$ is not constrained to be symmetric, this construction selects forward transition terms without requiring their reverse counterparts.

To bound the error for nonorthogonal identifiers, assume each row of $R$ has unit norm and let $E = RR^\top - I$. Then $E_{xx} = 0$ and $|E_{xy}| \le \mu$ for $x \neq y$, where $\mu$ is the identifier coherence. For the nonnegative uncompressed signatures $a = (B_i)_{u,:}$ and $b = (B_j)_{v,:}$, the channel-pair error satisfies

$$|aEb^\top| = \left| \sum_{x\neq y} a_x E_{xy} b_y \right| \le \mu \sum_{x\neq y} a_x b_y = \mu\left(\|a\|_1 \|b\|_1 - ab^\top\right).$$

Applying the triangle inequality across channel pairs, with weights $|(\mathbf{K}_{d,r})_{ij}|$, gives Equation (7). Orthogonal identifiers have $\mu=0$, recovering the exact score identity. □

### F.1 Capability witnesses

We give structural and relation-ranking witnesses for the implemented score in Section 4.2, separately from the preceding linear-bank proof.

**Structural witness.** Consider the undirected six-cycle on nodes $\{0,\dots,5\}$, with edges between consecutive nodes modulo six, identical node attributes and types, and one edge type. All nodes belong to the same automorphism orbit, so a deterministic permutation-equivariant endpoint encoder without symmetry-breaking inputs assigns them identical embeddings. Consequently, the nonedges $(0,2)$ and $(0,3)$ receive equal dot-product scores, although their neighborhood overlaps differ:

$$|\mathcal{N}(0)\cap\mathcal{N}(2)|=1, \qquad |\mathcal{N}(0)\cap\mathcal{N}(3)|=0.$$

**Actual log-count channels.** Hash each node independently and uniformly into $p_z \geq 2$ buckets, let $t=1/p_z$, and set $\mathbf{K}^{\mathrm{raw}}=\mathrm{diag}(1,0)$ to select the first-hop channel. For a fixed bucket $b$, let $X_b$ indicate whether the shared neighbor of nodes 0 and 2 hashes to that bucket. Conditional on $X_b$, their signature coordinates are independent, with means

$$m_0=t\log 2, \qquad m_1=(1-t)\log 2+t\log 3.$$

The shared neighbor therefore contributes $t(1-t)(m_1-m_0)^2$ to their covariance. In contrast, nodes 0 and 3 have disjoint neighborhoods, giving independent coordinates with the same marginal means. Summing over buckets yields

$$\begin{aligned}\Delta_{\mathrm{raw}} &:= \mathbb{E}_\xi[s^{\mathrm{raw}}(0,r,2)-s^{\mathrm{raw}}(0,r,3)]\\ &= (1-t)\left[(1-2t)\log 2+t\log 3\right]^2>0.\end{aligned} \tag{8}$$

For $p_z=2$, this gap is $(\log 3)^2/8$.

**Normalized channels.** Set $\mathbf{K}^{\mathrm{L2}}=\mathrm{diag}(1,0)$. If a node's two neighbors hash to the same bucket, its normalized first-hop signature is that bucket's basis vector; otherwise, it is the sum of their basis vectors divided by $\sqrt{2}$. The conditional coordinate means are therefore

$$n_0=\frac{t}{\sqrt{2}}, \qquad n_1=t+\frac{1-t}{\sqrt{2}}.$$

Applying the same conditional-independence argument gives

$$\begin{aligned}\Delta_{\mathrm{L2}} &:= \mathbb{E}_\xi[s^{\mathrm{L2}}(0,r,2)-s^{\mathrm{L2}}(0,r,3)]\\ &= (1-t)\left[t+\frac{1-2t}{\sqrt{2}}\right]^2>0.\end{aligned} \tag{9}$$

For $p_z=2$, this gap is $1/8$. With equal backbone scores and positive fusion weights and calibration scales held fixed across hash realizations, the raw and normalized branches yield a positive expected fused score gap.

**Relation witness.** A positive expected fused score gap implies that at least one hash realization has a positive gap. Fix such a realization, with equal backbone contributions for candidates 2 and 3. For two relations $r_1$ and $r_2$ sharing these compatible candidates, use the first-hop structural kernels above for $r_1$ and their negatives for $r_2$, with identical positive fusion weights and calibration scales across relations. Candidate 2 then ranks above candidate 3 for $r_1$ and below it for $r_2$, while the cached candidate vectors and message graph remain unchanged.

**Local adaptation.** With the backbone and signatures fixed, updating one task's structural kernels, fusion weights, or calibration scales leaves every other task's scoring function unchanged. This isolates task-local adaptation, but does not guarantee improved generalization or remove interference already present in the shared backbone.