RETHINKING LINK PREDICTION EVALUATION: FULL-SPACE REFERENCE AND MODEL-SCREENED CANDIDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Link prediction aims to infer unobserved edges in graphs and underlies many real-world applications. In practice, industrial systems typically use learned screening models to narrow vast candidate spaces and retain high-likelihood candidates before downstream prediction. However, academic evaluation protocols often construct candidate sets by randomly sampling or heuristically selecting non-edges, which may omit plausible and challenging negatives and lead to overly optimistic estimates of predictive capability. To quantify the effect of such candidate omission without imposing an additional selection rule, we use full-space evaluation, which ranks each test edge against the complete valid candidate space, as an idealized selection-free reference. Across 20 methods and seven datasets, we find that heuristic candidate selection can substantially inflate measured performance and distort method comparisons. Because full-space evaluation becomes costly on large graphs, we propose model-screened evaluation, which adapts the learned screening principle used in industrial systems to academic benchmarking and uses a learned model to retain a controllable number of high-likelihood candidates. With a fixed screening architecture and training configuration across all datasets, our protocol closely preserves the evaluation conclusions of the full-space reference. For example, on ogbl-ddi, model-screened evaluation reduces the average MRR deviation from full-space evaluation from 14.8 to 0.16 percentage points and the average pairwise MRR-gap deviation from 4.5 to 0.08.

1 INTRODUCTION

Link prediction aims to infer unobserved edges in graphs and underlies a wide range of real-world applications, including recommendation systems, social network analysis, biological interaction discovery, and financial risk detection. Recent methods have achieved strong performance on academic benchmarks Wang et al. (2024); Shomer et al. (2024). Despite this progress, academic evaluation and large-scale deployment differ substantially in how candidate sets are constructed. Real-world recommender systems typically use one or more learned screening models to filter out unlikely candidates and retain a small set of high-likelihood candidates for downstream prediction Covington et al. (2016); Cheng et al. (2023); Liu et al. (2024), as illustrated in Figure 1. In contrast, academic benchmarks commonly evaluate test links against small sets of randomly sampled or heuristically selected non-edges Li et al. (2023). This mismatch raises a key concern: can academic candidate construction omit challenging negatives and thereby produce overly optimistic estimates of link prediction capability? To answer this question, we first examine three limitations of current academic evaluation.

Limitation 1: Academic candidate sets may underrepresent plausible and challenging negatives in realistic prediction scenarios. Random negative sampling has been shown to produce many easy negatives that can be readily distinguished from true links Li et al. (2023). Heuristic-based protocols attempt to construct harder candidates, but rely on manually designed structural heuristics as coarse proxies for candidate difficulty (see Section 2.2 for details). As shown in Section 5.2, such rules may still fail to retain sufficiently challenging candidates. In contrast, candidates reaching downstream predictors in industrial systems have already survived one or more stages of learned screening, making them substantially more plausible than arbitrary non-edges Covington

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.

et al. (2016); Cheng et al. (2023); Liu et al. (2024). Consequently, existing academic evaluation protocols may overestimate link prediction capability when their candidate construction omits plausible and challenging candidates that learned screening would retain for downstream prediction.

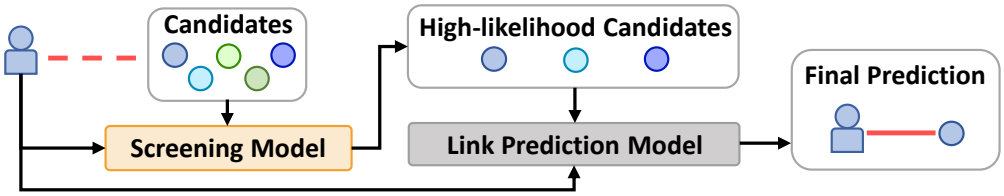


Figure 1: A typical industrial pipeline with learned candidate screening before downstream prediction or ranking Covington et al. (2016); Cheng et al. (2023).

Limitation 2: Heuristic candidate selection relies on multiple design choices. Heuristic-based evaluation requires manual choices about which heuristics to use, how each heuristic is configured, and how the selection strategy is adapted across datasets. These design choices may lead to inconsistent heuristic configurations across studies, resulting in different candidate sets and potentially compromising the fairness and reliability of comparisons in link prediction research.

Limitation 3: Test-time efficiency receives limited attention. Academic link prediction studies primarily report predictive performance, while test-time efficiency is rarely evaluated under consistent candidate settings. Yet inference efficiency is critical in real-world systems, where large candidate spaces must be processed under strict latency constraints. For example, industrial recommender systems may narrow millions or billions of candidates to a small set of relevant items within tens of milliseconds Liu et al. (2024). It therefore remains unclear whether improvements in predictive performance justify the added inference cost in practical deployment.

To address these limitations, we make the following contributions.

Contribution 1: We establish full-space evaluation as a selection-free reference for candidate construction. Existing protocols first select a small set of non-edges for evaluation, either by random sampling or heuristic selection, as exemplified by HeaRT Li et al. (2023), which forms candidate pools by pairing each endpoint with non-connected nodes and selects challenging negatives using heuristic scores. Full-space evaluation instead skips this selection step and ranks the positive edge against its entire candidate pool. This ensures that no potentially challenging negatives are excluded and eliminates variability introduced by candidate selection: the candidate set is uniquely determined by the benchmark graph and test edge, providing a fair and consistent basis for comparison. Using this reference, we show that heuristic evaluation can substantially inflate reported performance and distort the apparent performance gaps among methods. For example, on Squirrel, HeaRT deviates from full-space evaluation by an average of 22.1 percentage points in MRR across methods and 8.6 percentage points in pairwise MRR gaps.

Contribution 2: We propose model-screened evaluation to approximate the full-space reference with a small learned candidate set. When full-space evaluation becomes computationally costly, we develop model-screened evaluation by adapting the learned screening principle widely used in practical systems (addressing **Limitation 1**). Specifically, we use a simple learned screening model to filter unlikely candidates and retain a small, controllable number of high-likelihood candidates for downstream evaluation. To minimize protocol-level design choices, we fix the model architecture, hyperparameters, and training configuration across all datasets without dataset-specific tuning (addressing **Limitation 2**). Despite the substantially reduced candidate space, model-screened evaluation consistently maintains high agreement with the full-space reference and substantially outperforms heuristic screening in evaluation fidelity. For example, on Squirrel, it reduces the average MRR deviation from full-space evaluation from 22.1 to 0.33 percentage points and the average pairwise MRR-gap deviation from 8.6 to 0.25 percentage points.

Contribution 3: We conduct a broad evaluation of predictive performance and inference efficiency. We extensively evaluate 20 existing link prediction methods across diverse graph domains, including social, co-purchase, citation, biomedical, and Wikipedia page networks. Beyond predictive performance, we measure their inference efficiency under consistent candidate settings (address-

ing **Limitation 3**), enabling us to assess whether performance improvements justify the additional computational cost required for practical deployment.

The remainder of this paper is organized as follows. Section 2 introduces the link prediction task and evaluation background. Sections 3 and 4 present the full-space and model-screened evaluation, respectively. Section 5 evaluates link prediction methods under a representative existing protocol, full-space evaluation, and model-screened evaluation. Section 6 concludes the paper.

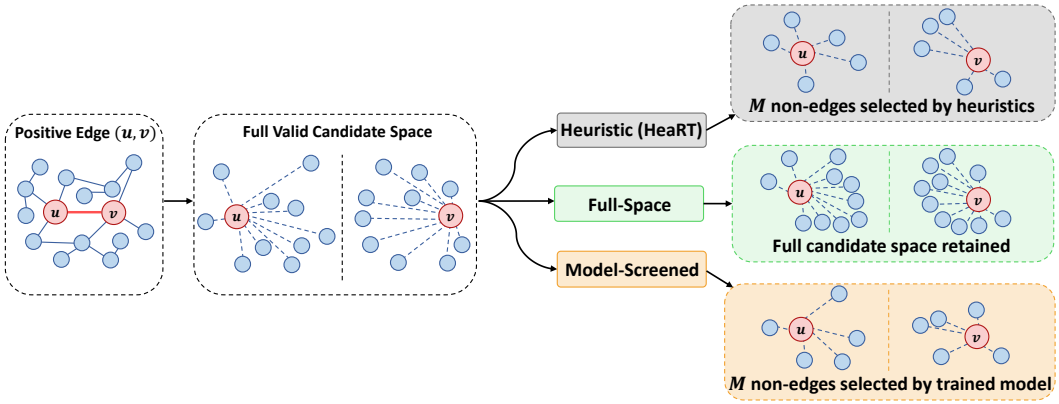


Figure 2: Three link prediction evaluation protocols: heuristic-based evaluation (represented by HeaRT Li et al. (2023)), Full-Space, and Model-Screened. Given a positive test edge (u, v), HeaRT ranks non-edges using predefined heuristics and retains M candidates; Full-Space retains all valid candidates; and Model-Screened ranks non-edges using a trained model and retains M candidates.

2 Preliminaries

This section introduces the link prediction task and reviews representative candidate selection protocols, evaluation settings, and prediction methods.

2.1 Problem Definition

Let $G = (V, E, X)$ denote a graph with node set V, edge set E, and node features X. We split the observed edges into training, validation, and test sets E_{tr}, E_{val}, and E_{te}, and define the training graph as $G_{\text{tr}} = (V, E_{\text{tr}}, X)$. We focus on undirected graphs, while the formulation readily extends to directed graphs. Given G_{tr}, a link prediction method assigns a score $s(u, v)$ to each candidate node pair (u, v), with higher scores indicating more likely edges. For each test edge $(u, v) \in E_{\text{te}}$, it is ranked against a set of candidate non-edges $\mathcal{N}(u, v)$, and performance is measured based on its resulting rank. The construction of $\mathcal{N}(u, v)$ therefore determines which non-edges a method is evaluated against and directly affects the measured predictive performance. This work studies this effect and develops evaluation protocols that better account for the challenging candidates relevant to downstream prediction.

2.2 Candidate Selection Protocols

Random candidate selection. Random candidate selection Li et al. (2023) uniformly samples M non-edges from arbitrary node pairs in V and uses the same negative set for all positive test edges. Because the sampled negatives are independent of the corresponding positives, they are often structurally or semantically unrelated and thus easy to distinguish from true links.

Heuristic candidate selection (HeaRT Li et al. (2023)). Heuristic selection has been increasingly adopted in recent work Shomer et al. (2024) as an alternative to random sampling, with HeaRT being a representative approach. As shown at the top of Figure 2, for each test edge (u, v), HeaRT constructs two endpoint-specific candidate pools by replacing either endpoint with every node in V, excluding self-loops and positive edges in the dataset. It ranks the remaining candidates using

predefined heuristics, including resource allocation (RA) and personalized PageRank (PPR), and additionally uses node-feature cosine similarity for specific datasets. Each heuristic ranks the candidates independently, and HeaRT assigns each candidate its best (minimum) rank across heuristics. It then selects the top $M/2$ candidates for each endpoint, yielding M negatives per test edge.

Candidate selection in industry. Large-scale recommender systems commonly use learned models to screen plausible links from candidate spaces containing millions or billions of possibilities before downstream prediction or ranking. Given one endpoint, these models score possible counterparts and retain only hundreds or thousands of high-scoring candidates, often through one or more progressively refined screening stages. This paradigm has been widely used in video recommendation, short-video platforms, and e-commerce systems Covington et al. (2016); Liu et al. (2024); Cheng et al. (2023). Despite implementation differences, these systems share the same principle of using learned models to identify plausible candidates before downstream prediction or ranking. This motivates us to examine whether academic evaluation protocols that do not use learned screening may omit challenging candidates that materially affect measured performance.

2.3 Link Prediction Methods

We broadly categorize representative link prediction methods into five groups, summarized in Table 4 in Appendix A. **(1) Heuristic-based methods:** Common Neighbors (CN) Newman (2001), Adamic Adar (AA) Adamic & Adar (2003), Resource Allocation (RA) Zhou et al. (2009), Katz Katz (1953), and Shortest Path (SP) Liben-Nowell & Kleinberg (2003), which score node pairs using hand-crafted graph structural properties. **(2) Embedding-based methods:** Matrix Factorization (MF) Menon & Elkan (2011), Multilayer Perceptron (MLP), and Node2Vec Grover & Leskovec (2016), which learn low-dimensional node representations for link prediction. **(3) Standard GNN methods:** GCN Kipf & Welling (2017), GraphSAGE Hamilton et al. (2017), GAE Kipf & Welling (2016), and GAT Veličković et al. (2018), which learn node representations through neighborhood aggregation. **(4) GNN methods with pairwise information:** SEAL Zhang & Chen (2018) and BUDDY Chamberlain et al. (2023) exploit local subgraph information, NBFNet Zhu et al. (2021) models paths between node pairs, Neo-GNN Yun et al. (2021), NCN, and NCNC Wang et al. (2024) exploit neighborhood overlap, and PEG Wang et al. (2022) incorporates graph positional encodings. **(5) Transformer-based methods:** LPFormer Shomer et al. (2024) models pairwise graph context through attention, while LinkGPT He et al. (2025) uses LLMs to model textual and structural information for link prediction.

3 Full-Space Evaluation

To determine how much candidate selection itself affects evaluation, we need a reference that does not depend on any particular candidate selection rule. As discussed in Sections 1 and 2.2, existing candidate selection protocols may exclude challenging negatives, leading to overly optimistic evaluations and altering comparisons among prediction methods. Moreover, heuristic selection introduces various protocol-level design choices that determine the evaluated candidate set and can affect comparability across studies. We highlight two major sources of such variability below.

(1) Choice and adaptation of heuristics. Candidate selection depends on which heuristics are used, how their rankings are combined, and how the heuristic set is adapted across datasets. For example, HeaRT Li et al. (2023), a representative heuristic evaluation protocol, primarily relies on resource allocation (RA) and personalized PageRank (PPR). However, on some datasets, it additionally incorporates feature similarity because feature-based prediction methods were observed to perform strongly. Thus, heuristic selection is informed by observed prediction results, and alternative heuristics such as common neighbors are also possible. The choice of heuristics may also be constrained by dataset characteristics; for example, feature similarity cannot be applied to featureless graphs such as ogbl-ddi Hu et al. (2020). Therefore, there is no canonical rule for selecting or combining heuristics across datasets.

(2) Configuration of individual heuristics. Even with a fixed heuristic set, individual heuristics may involve additional configuration choices. Personalized PageRank depends on parameters such as the restart probability and potentially the number of propagation steps. Feature-based heuristics

can also use different similarity or distance measures, such as cosine similarity, dot product, or L_1/L_2 distance.

These choices matter because they directly determine the test candidate set rather than merely tuning an evaluated model. Different configurations can therefore change performance and relative comparisons across methods, reducing reproducibility and comparability across studies. This motivates a reference that removes candidate selection choices altogether.

Why use full-space as a reference? Conceptually, link prediction can be viewed as a conditional ranking problem: given one endpoint, the model seeks to identify its true counterpart from the complete set of valid alternatives. For example, in real-world recommendation systems, the idealized prediction target is to rank the most relevant item for a given user among all valid candidate items. This complete set of alternatives therefore provides a natural underlying candidate universe. Full-space evaluation retains this entire universe without applying an additional candidate selection rule. Consequently, no potentially challenging negative can be excluded by candidate construction, and deviations from full-space evaluation can be attributed to the candidate selection step. We therefore use full-space evaluation as an idealized, selection-free reference.

3.1 Full-Space Evaluation Protocol

As illustrated in the middle of Figure 2, for each positive test edge $(u, v) \in E_{\text{te}}$, full-space evaluation replaces either endpoint with every node in V, removes self-loops and known positive edges, and retains all remaining non-edges without further selection. The resulting candidate set is uniquely determined by the graph and test edge and covers the complete valid candidate space, ensuring that no potentially challenging negative is excluded. We therefore use full-space evaluation as an idealized reference for comparing candidate selection protocols.

4 Model-Screened Evaluation

Full-space evaluation provides a selection-free reference, but scoring the complete candidate space can be costly for large graphs, especially when evaluation is repeated during validation and testing. As discussed in Section 2.2, practical systems address this issue using trained predictive models to screen large candidate spaces and retain plausible candidates before downstream prediction. Rather than adapting hand-crafted heuristics from dataset-specific empirical observations, we draw on this widely deployed paradigm, which has been extensively validated in successful real-world systems serving massive user populations.

Directly reproducing industrial screening pipelines on academic benchmarks is neither feasible nor necessary. Industrial systems can exploit rich proprietary information, such as detailed entity attributes, long interaction histories, and behavioral context, whereas academic link prediction benchmarks typically provide only a graph and limited node information Hu et al. (2020). They also operate under substantially greater scale and stricter system constraints, often requiring complex multi-stage pipelines and extensive feature engineering. Such complexity is unnecessary for academic evaluation and would introduce additional protocol-level variability. We therefore adapt the core principle of industrial learned screening to academic benchmarks, using a learned screener to retain plausible candidates while avoiding dataset-specific heuristic design. We then evaluate whether these compact candidate sets preserve the conclusions obtained from the selection-free full-space reference in Section 5.

4.1 Model-Screened Evaluation Protocol

We develop the Context-Augmented MLP (CAM), a simple screening model inspired by industrial candidate generation and deliberately designed to minimize protocol-level design choices. We first describe its architecture and design rationale, and then present the fixed training configuration and candidate selection procedure.

Model architecture. Following industrial screening models that combine trainable ID embeddings with interaction history Covington et al. (2016); Cheng et al. (2023), CAM associates each node with a trainable embedding and, when scoring a candidate pair, augments each endpoint embedding with the aggregated embeddings of its one-hop neighbors. Let $\mathbf{H}$ stack all node embeddings, and

let $\mathbf{A}_{\text{tr}}$ and $\mathbf{D}$ denote the training adjacency matrix and its degree matrix, respectively. We use the standard normalized training adjacency matrix Kipf & Welling (2017): $\hat{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}_{\text{tr}}\mathbf{D}^{-1/2}$. As illustrated in Figure 3, CAM first encodes the context-augmented endpoint representations into latent embeddings and then uses a decoder to produce the prediction score:

$$\begin{aligned} \text{Context augmentation:} \quad & \mathbf{x}_w = \mathbf{h}_w \oplus \{\hat{\mathbf{A}}\mathbf{H}\}_w, \quad w \in \{u, v\}, \\ \text{Endpoint encoding:} \quad & \mathbf{z}_w = \text{Enc}(\mathbf{x}_w), \quad w \in \{u, v\}, \\ \text{Decoder scoring:} \quad & s(u, v) = \sigma(\text{Dec}(\mathbf{z}_u \otimes \mathbf{z}_v)) \,. \end{aligned}$$

Here, $\oplus$ and $\otimes$ denote concatenation and element-wise multiplication, respectively. Both Enc and Dec are 2-layer MLPs with a hidden dimension of 256.

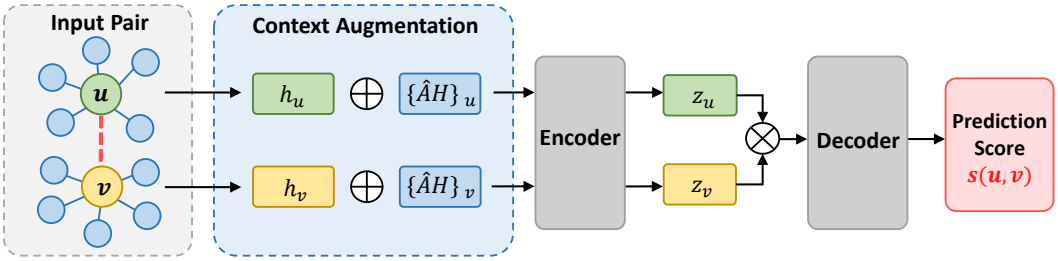


Figure 3: Architecture of the screening model CAM.

Design rationale. To instantiate the core learned screening principle used in practical systems while minimizing dataset-specific modeling choices, CAM adopts three simple design choices. **(1) MLP-based backbone:** MLP-based encoders are widely used in industrial screening models Covington et al. (2016); Cheng et al. (2023) and introduce fewer architectural and tuning choices than more expressive graph models. **(2) Trainable node embeddings:** Trainable embeddings provide a unified node representation across datasets without relying on potentially heterogeneous or unavailable node attributes. **(3) One-hop context:** One-hop neighbors provide a natural graph analogue of direct interaction histories commonly used in industrial screening, while avoiding additional multi-hop modeling choices. We intentionally avoid optimizing the screening architecture separately for each dataset, as this would reintroduce protocol-level variability. As shown in Section 5.2, this simple design already produces evaluations close to full-space evaluation.

Fixed configuration. To minimize screening model variability, we use the same CAM architecture, training objective, and hyperparameter configuration across all datasets, without dataset-specific tuning. CAM requires no specialized training procedure and is trained independently of downstream methods using the standard link prediction setup: BCE on observed edges and randomly sampled non-edges Li et al. (2023). For each dataset, the resulting model is fixed and shared across all downstream evaluations. Detailed training settings are provided in Appendix B.3.2.

Candidate selection. As illustrated at the bottom of Figure 2, for each positive test edge (u, v), CAM scores all valid non-edges obtained by replacing either endpoint, excluding self-loops and known positive edges. Following HeaRT, candidates from the two endpoints are ranked separately, with the top $M/2$ retained from each side, where $M = 500$ is the total number of negative candidates per positive edge.

Overall, model-screened evaluation learns dataset-specific predictive patterns while keeping the screening architecture and training configuration fixed across datasets, thereby limiting protocol-level design choices. Section 5 evaluates its fidelity to full-space evaluation.

5 Experimental Results

We evaluate three protocols: full-space, model-screened, and HeaRT, a representative heuristic-based protocol; we omit random selection because prior work has shown that it provides unreliable evaluation results Li et al. (2023).

Table 1: Mapping of evaluated methods to compact IDs, shown as method (ID). H, E, G, P, and T denote heuristic, embedding-based, standard GNN, pair-aware GNN, and Transformer-based methods, respectively.

CN (H1)	AA (H2)	RA (H3)	Katz (H4)	SP (H5)	MF (E1)	MLP (E2)
Node2Vec (E3)	GCN (G1)	SAGE (G2)	GAE (G3)	GAT (G4)	SEAL (P1)	BUDDY (P2)
NBFNet (P3)	Neo-GNN (P4)	NCN (P5)	NCNC (P6)	PEG (P7)	LPFormer (T1)	

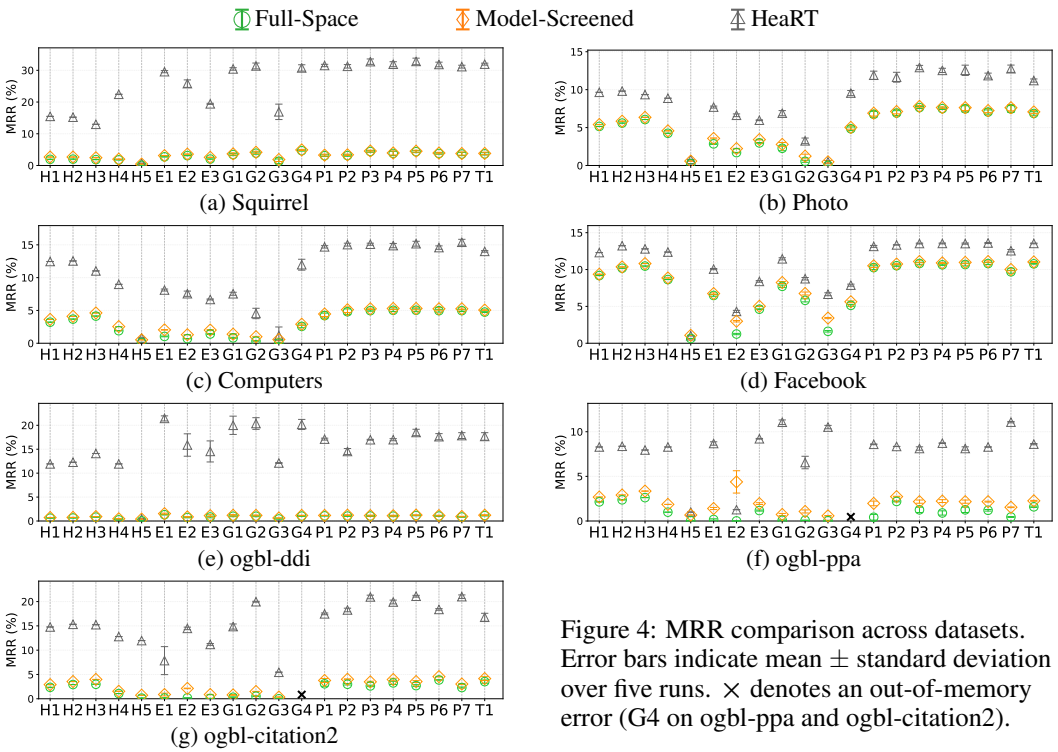


Figure 4: MRR comparison across datasets. Error bars indicate mean ± standard deviation over five runs. × denotes an out-of-memory error (G4 on ogbl-ppa and ogbl-citation2).

5.1 Experimental Settings

Datasets and methods. We evaluate seven widely used graph datasets spanning social, co-purchase, biological, citation, and Wikipedia networks: Squirrel, Photo, Computers, Facebook, ogbl-ddi, ogbl-ppa, and ogbl-citation2 Li et al. (2023); Shchur et al. (2018); Pei et al. (2020); Hu et al. (2020). Dataset statistics and split details are provided in Appendix B.1. We evaluate 20 link prediction methods introduced in Section 2.3 and listed in Table 1; LinkGPT is excluded because the textual node information it requires is unavailable in the adopted datasets.

Evaluation metrics. Following prior link prediction benchmarks Li et al. (2023), we report mean reciprocal rank (MRR) and Hits@K for $K \in \{5, 10, 20, 50\}$. To quantify evaluation fidelity relative to full-space evaluation, we report average deviations in individual-method MRR (MRR-Diff) and pairwise MRR gaps (Gap-Diff). Because MRR values and pairwise gaps are generally small, we also report normalized versions of these metrics relative to their corresponding full-space scales for easier interpretation. Detailed metric definitions are provided in Appendix B.2.

Implementation details. For the downstream link prediction models, we follow the training and hyperparameter selection procedure of HeaRT Li et al. (2023). We keep each trained model fixed across evaluation protocols so that observed differences can be attributed specifically to candidate construction rather than differences in model training or hyperparameter selection. CAM is trained with a single configuration shared across all datasets. Following HeaRT, we set $M = 500$ candidates per positive test edge for both HeaRT and model-screened evaluation unless otherwise specified. All reported results are averaged over five runs with different random seeds. All experiments are conducted on one NVIDIA H100 GPU. Full implementation details are provided in Appendix B.3.

5.2 Comparison of Evaluation Protocols

Figure 4 compares the evaluated link prediction methods under full-space, model-screened, and HeaRT evaluation using MRR. The corresponding Hits@K results, which exhibit consistent trends, are reported in Appendix Figure 6. Table 2 summarizes the corresponding MRR-Diff, Gap-Diff, and their normalized counterparts relative to full-space evaluation.

Table 2: MRR and pairwise gap deviations from full-space evaluation under HeaRT and model-screened protocols. Absolute deviations are reported in percentage points and normalized deviations in percentages. Lower values indicate closer alignment with full-space results.

Metric	Protocol	Squirrel	Photo	Computers	Facebook	ogbl-ddi	ogbl-ppa	ogbl-citation2
MRR-Diff	HeaRT	22.1	4.1	7.6	3.0	14.8	7.0	13.8
	Model-Screened	**0.33**	**0.34**	**0.39**	**0.47**	**0.16**	**1.0**	**0.74**
Normalized MRR-Diff	HeaRT	752	87.7	254	38.4	1759	698	737
	Model-Screened	**11.3**	**7.3**	**12.8**	**6.0**	**19.5**	**105**	**39.6**
Gap-Diff	HeaRT	8.6	1.4	3.1	0.83	4.5	2.8	4.1
	Model-Screened	**0.25**	**0.25**	**0.30**	**0.40**	**0.08**	**0.67**	**0.31**
Normalized Gap-Diff	HeaRT	645	45.9	135	21.9	1318	278	252
	Model-Screened	**19.1**	**8.1**	**13.0**	**10.6**	**23.0**	**66.7**	**19.2**

Observation 1: Heuristic screening can substantially affect both absolute performance and relative model comparisons. As shown in Figure 4 and Table 2, HeaRT deviates substantially from full-space evaluation, generally reporting considerably higher MRR and larger performance gaps among methods. For example, on Squirrel, ogbl-ddi, and ogbl-citation2, HeaRT exhibits average MRR deviations of 22.1, 14.8, and 13.8 percentage points, respectively, and average pairwise MRR-gap deviations of 8.6, 4.5, and 4.1 percentage points, respectively. Moreover, on datasets such as ogbl-ppa and ogbl-citation2, methods that appear among the strongest under HeaRT fall behind or perform similarly to multiple competitors under full-space evaluation. These results support our concern that heuristic candidate construction can omit challenging negatives and consequently provide overly optimistic estimates of predictive capability.

Observation 2: Model-screened evaluation closely tracks the full-space reference across diverse graphs using a fixed configuration. As shown in Figure 4 and Table 2, with $M = 500$ negatives per positive, model-screened evaluation closely tracks the full-space reference across datasets and prediction methods, substantially reducing the deviations introduced by heuristic screening. For example, on Squirrel and ogbl-ddi, model-screened evaluation reduces MRR-Diff from 22.1 and 14.8 to 0.33 and 0.16 percentage points, respectively, while reducing Gap-Diff from 8.6 and 4.5 to 0.25 and 0.08 percentage points, respectively. Similar improvements hold across the remaining datasets, suggesting that model-screened evaluation retains plausible and challenging candidates that better preserve the performance patterns observed under full-space evaluation, despite using only a small subset of the full candidate space. Importantly, we use the same screening architecture, training objective, and hyperparameters across all datasets, without dataset-specific tuning. By learning predictive patterns from the training data under this fixed configuration, model-screened evaluation reduces dataset-specific design choices and provides a more consistent evaluation protocol.

Observation 3: Effective candidate selection can outweigh a larger candidate set. We additionally vary the number of retained candidates $M \in \{100, 250, 500, 1000\}$ in model-screened evaluation, with complete results reported in Appendix Figure 7. Even with $M = 100$, model-screened evaluation remains substantially closer to full-space evaluation than HeaRT with $M = 500$ for most methods, and progressively approaches full-space as M increases.

Observation 4: The benefit of learned screening persists across screening architectures. We further replace CAM with three representative learned screeners from different model families, including MLP (embedding-based), GCN (standard GNN), and SEAL (pairwise GNN), with the complete results in Appendix Figure 5 and Table 6. Different learned screeners yield broadly consistent downstream MRR and remain substantially closer to full-space evaluation than heuristic screening, indicating that the advantage of model-screened evaluation is not specific to CAM. MLP consistently exhibits larger deviations from full-space evaluation, suggesting that neighborhood context helps

identify challenging candidates. GCN and SEAL introduce additional graph-specific design choices, such as message-passing depth and pair-specific subgraph modeling, respectively, but achieve only comparable or modestly better evaluation fidelity than CAM. In contrast, CAM fixes structural context to a single neighborhood aggregation step. Notably, although MLP, GCN, and SEAL are each tuned separately for every dataset following Appendix B.3.3, none provides a substantial improvement in fidelity over CAM. We therefore use CAM as the default screener not because it universally achieves the lowest deviation, but because it adapts the core principle of industrial learned screening to academic benchmarks through a learnable model with direct interaction context, while deliberately limiting graph-specific modeling choices.

Overall, these results support model-screened evaluation as a more practically grounded and consistent alternative to heuristic screening.

Table 3: Per-run inference time (seconds) of link prediction models on a test set of 10,000 positive links under model-screened evaluation (500 candidates) and full-space evaluation. × denotes an out-of-memory error.

Dataset	Protocol	H1	H2	H3	H4	H5	E1	E2	E3	G1	G2	G3	G4	P1	P2	P3	P4	P5	P6	P7	T1
Computers	Model-Screened	1.2	1.0	1.0	0.43	1.1	0.055	0.14	1.0	0.14	0.15	0.12	0.13	0.9	0.8	0.8	0.56	0.8	1.0	0.48	4.5
	Full-Space	5.1	5.0	5.0	4.7	4.9	4.9	7.3	7.2	9.1	9	4.9	11.2	26	23	23	21	23	27	15	126.2
ogbl-ppa	Model-Screened	1.6	1.6	1.6	32	26	0.07	0.13	0.14	0.33	0.17	0.26	×	1.7	1.4	1.6	0.7	1.6	1.7	0.48	4.0
	Full-Space	12	12	12	40	56	127	128	139	128	128	30	×	800	654	654	597	655	939	385	5634
ogbl-citation2	Model-Screened	1.1	1.3	1.5	88.1	307.1	0.1	0.12	0.12	0.30	0.20	0.3	×	0.6	0.33	0.52	0.49	0.5	0.7	0.42	1.8
	Full-Space	22	22	22	104	507	327	338	354	330	339	69	×	2279	1785	1776	1567	1749	2959	981	13710

5.3 Inference Time Comparison

Here we focus on the inference cost of downstream link prediction methods under controlled numbers of candidates. Table 3 reports the per-run inference time on 10,000 positive links using either $M = 500$ candidates or the full valid candidate space for each positive link. Transformer-based and pair-aware GNN methods are generally more expensive than simpler predictors. The heuristic-based methods H4 and H5 are also costly on large graphs, as their computation is dominated by multi-hop graph traversal and is therefore less sensitive to the number of candidate links; consequently, their runtime increases only modestly from 500 candidates to full-space evaluation. These differences are further amplified under full-space evaluation, where several methods require hundreds or even thousands of seconds per run, making repeated validation and testing, such as hyperparameter tuning and checkpoint selection, particularly costly. Importantly, these substantially higher inference costs often yield only modest or no improvements in predictive performance, as shown in Figure 4. In practical systems, inference latency can directly affect deployment feasibility. For example, in social link recommendation and fraud or risk detection, large volumes of candidate links may need to be scored under strict latency constraints. These results therefore suggest that predictive gains should be considered together with their inference cost, especially when substantially more expensive models provide only limited accuracy improvements.

6 Conclusion

We revisit link prediction evaluation from the perspective of candidate construction, motivated by the mismatch between learned screening in practical systems and heuristic selection in academic benchmarks. Using full-space evaluation as a selection-free reference, we show across 20 methods and seven datasets that heuristic selection can omit challenging negatives, yielding overly optimistic performance and altered method comparisons. Because repeatedly scoring the full candidate space with downstream models is costly, we propose model-screened evaluation, which adapts learned screening to academic benchmarks and retains a controllable set of high-likelihood candidates. With a fixed configuration across datasets, it closely preserves full-space results while substantially reducing downstream scoring cost. We further show that greater computational cost does not always yield commensurate gains. Overall, these results motivate learned screening as a more practically grounded alternative to heuristic selection.

AI USE STATEMENT

Generative AI tools were used to assist in improving the clarity, organization, and presentation of the manuscript. They were also used to provide feedback on experimental design and to assist with code modification and debugging as part of the research process. All AI-assisted suggestions and code changes were reviewed and verified by the authors. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies evaluation protocols for link prediction using existing benchmark datasets. It does not involve new data collection, human-subject studies, or interaction with users. Our proposed protocols modify only how candidate links are constructed and evaluated, without changing the underlying benchmark data or introducing new sources of personal or sensitive information. The work is intended for methodological evaluation and benchmarking rather than direct deployment in real-world decision-making systems. We adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 5 describes the evaluated datasets, link prediction methods, evaluation metrics, candidate-set sizes, and experimental environment. Appendix B.1 provides dataset statistics and split settings, Appendix B.2 gives complete definitions of the evaluation metrics, and Appendix B.3 details the training and evaluation configurations for HeaRT, CAM, alternative screening models, and downstream link prediction methods. Anonymous source code for reproducing the experiments is available at `https://anonymous.4open.science/r/realistic-link-prediction-D1FC`. A copy of the source code is also included as a ZIP file in the supplementary material submitted with the paper.

REFERENCES

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

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

Mengli Cheng, Yue Gao, Guoqiang Liu, and Hongsheng Jin. Easyrec: An easy-to-use, extendable and efficient framework for building industrial recommendation systems. In *AAAI*, pp. 16419–16421, 2023.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pp. 191–198, 2016.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pp. 855–864, 2016.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1025–1035, 2017.

Zhongmou He, Jing Zhu, Shengyi Qian, Joyce Chai, and Danai Koutra. Linkgpt: Leveraging large language models for enhanced link prediction in text-attributed graphs. In *CIKM*, pp. 843–853, 2025.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: datasets for machine learning on graphs. In *NeurIPS*, pp. 22118–22133, 2020.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

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

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: current pitfalls and new benchmarking. In *NeurIPS*, pp. 3853–3866, 2023.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM*, pp. 556–559, 2003.

Chi Liu, Jiangxia Cao, Rui Huang, Kai Zheng, Qiang Luo, Kun Gai, and Guorui Zhou. Kuaiformer: Transformer-based retrieval at kuaishou. *arXiv preprint arXiv:2411.10057*, 2024.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 437–452. Springer, 2011.

Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

Hongbin Pei, Bingzhe Wei, Kevin Chen Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

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

Harry Shomer, Yao Ma, Haitao Mao, Juanhui Li, Bo Wu, and Jiliang Tang. LPFormer: An adaptive graph transformer for link prediction. In *KDD*, pp. 2686–2698, 2024.

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

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *ICLR*, 2022.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *ICLR*, volume 2024, pp. 14573–14589, 2024.

Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. In *NeurIPS*, pp. 13683–13694, 2021.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS*, pp. 5171–5181, 2018.

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

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: a general graph neural network framework for link prediction. In *NeurIPS*, pp. 29476–29490, 2021.

A Link Prediction Method Characteristics

Table 4: Comparison of representative link prediction method categories. *Train* indicates whether model training is required; *Neigh. agg.* indicates whether neighborhood aggregation is used; *Pair structure* indicates whether pair-specific structural information is modeled; *Pair cost* characterizes the computation for each candidate pair; and *Scalability* indicates suitability for large candidate sets.

Category	Methods	Train	Neigh. agg.	Pair structure	Pair cost	Scalability
Heuristic	CN, AA, RA, Katz, SP	✗	✗	✓	Low–Med.	High
Embedding	MF, MLP, Node2Vec	✓	✗	✗	Low	High
Standard GNN	GCN, SAGE, GAE, GAT	✓	✓	✗	Low	High
Pair-aware GNN	SEAL, BUDDY, NBFNet, Neo-GNN, NCN, NCNC, PEG	✓	✓	✓	Varies	Varies
Transformer-based	LPFormer, LinkGPT	Varies	Varies	Optional	High	Limited

B Additional Experimental Materials

B.1 Dataset Details

Table 5 summarizes the dataset statistics and split ratios. For Squirrel, Photo, Computers, and Facebook, we randomly split observed edges into training, validation, and test sets with an 85/5/10 ratio Li et al. (2023). For the OGB datasets, we use their official benchmark splits Hu et al. (2020).

Table 5: Statistics of the datasets. "–" denotes unavailable information.

Statistic	Squirrel	Photo	Computers	Facebook	ogbl-ddi	ogbl-ppa	ogbl-citation2
Nodes	5,201	7,650	13,752	22,470	4,267	576,289	2,927,963
Edges	217,073	238,162	491,722	342,004	1,334,889	30,326,273	30,561,187
Features	2,089	745	767	128	–	58	128
Train/Val/Test (%)	85/5/10	85/5/10	85/5/10	85/5/10	80/10/10	70/20/10	98/1/1
Content	Wikipedia page	Co-purchase	Co-purchase	Social	Drugs	Proteins	Citation

B.2 Metric Details

Let N_{pos} denote the number of positive test edges and r_i denote the rank of the i-th positive edge within its corresponding candidate pool.

MRR. Mean reciprocal rank (MRR) averages the reciprocal rank of each positive edge:

$$\text{MRR} = \frac{1}{N_{\text{pos}}} \sum_{i=1}^{N_{\text{pos}}} \frac{1}{r_i}.$$

Hits@K. Hits@K measures the proportion of positive edges that appear among the top-K ranked candidates:

$$\text{Hits@}K = \frac{1}{N_{\text{pos}}} \sum_{i=1}^{N_{\text{pos}}} \mathbb{I}(r_i \le K),$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.

To quantify the overall effect of candidate construction relative to full-space evaluation, we further measure deviations in both method performance and pairwise performance gaps. Let Q denote the number of prediction methods, s_m^P denote the MRR of method m under protocol P, and s_m^{F} denote its MRR under full-space evaluation.

Mean absolute MRR difference. This metric measures the average absolute MRR deviation across prediction methods:

$$\text{MRR-Diff}(P) = \frac{1}{Q} \sum_{m=1}^{Q} \left| s_m^P - s_m^{\text{F}} \right|.$$

Normalized MRR difference. We further normalize the MRR difference by the MRR scale under full-space evaluation:

$$\text{Normalized MRR-Diff}(P) = \frac{\text{MRR-Diff}(P)}{\text{Average MRR under full-space}} \times 100\%.$$

Formally,

$$\text{Normalized MRR-Diff}(P) = \frac{\sum_{m=1}^{Q} \left| s_m^P - s_m^{\mathrm{F}} \right|}{\sum_{m=1}^{Q} s_m^{\mathrm{F}}} \times 100\%.$$

Mean absolute pairwise MRR gap difference. To capture changes in the relative performance differences between methods, we compare the MRR gap of every method pair under protocol P with that under full-space evaluation:

$$\text{Gap-Diff}(P) = \frac{2}{Q(Q-1)} \sum_{m<n} \left| \left(s_m^P - s_n^P\right) - \left(s_m^{\mathrm{F}} - s_n^{\mathrm{F}}\right) \right|.$$

This metric first computes the absolute change in the MRR gap for each method pair and then averages it over all pairs.

Normalized pairwise MRR gap difference. We further normalize the pairwise MRR gap difference by the gap scale under full-space evaluation:

$$\text{Normalized Gap-Diff}(P) = \frac{\text{Gap-Diff}(P)}{\text{Average absolute pairwise MRR gap under full-space}} \times 100\%.$$

Formally,

$$\text{Normalized Gap-Diff}(P) = \frac{\sum_{m<n} \left| \left(s_m^P - s_n^P\right) - \left(s_m^{\mathrm{F}} - s_n^{\mathrm{F}}\right) \right|}{\sum_{m<n} \left| s_m^{\mathrm{F}} - s_n^{\mathrm{F}} \right|} \times 100\%.$$

For both normalized metrics, we normalize the aggregate deviation rather than averaging method- or pair-specific percentage changes, as our goal is to characterize the overall effect of a candidate protocol across the evaluated methods and method pairs rather than emphasize changes associated with any individual method or pair.

B.3 Implementation Details

B.3.1 Configuration of HeaRT

For HeaRT evaluation, we use RA and PPR as the structural heuristics, following the original protocol Li et al. (2023). We follow the official HeaRT implementation and use its default configurations for both heuristics.

B.3.2 Configuration of CAM

CAM is trained following the standard link prediction setup, using BCE on observed edges and randomly sampled non-edges as in Li et al. (2023), and is trained independently of the downstream link prediction methods. Its encoder and decoder each contain two layers, with both the initial embedding dimension and hidden dimension set to 256. We use a learning rate of 0.01 and a dropout rate of 0.1 for all datasets. During training, we evaluate CAM every five epochs by computing the mean MRR of validation edges against 500 randomly sampled negative edges. Training is stopped if the validation MRR does not improve for 10 consecutive evaluations or if the number of training epochs reaches 500.

B.3.3 Configuration of Alternative Screening Models

To examine the sensitivity of model-screened evaluation to the choice of screening model, we additionally consider MLP, GCN, and SEAL as alternative screeners in place of CAM. Each screening model is trained independently of the downstream link prediction methods. We tune their model hyperparameters over the same search spaces as those specified for the corresponding link prediction models in Appendix B.3.4. Validation and early stopping follow the same procedure as CAM, described in Appendix B.3.2.

B.3.4 Configuration of Link Prediction Models

For all trainable link prediction methods, we follow the general training protocol of Li et al. (2023). We optimize models using binary cross-entropy loss and the Adam optimizer with default PyTorch settings, except that the learning rate is selected from $0.01, 0.001$. During training, we randomly sample one negative edge for each positive edge. Candidate construction is used only for validation and test evaluation and does not alter the training data or optimization procedure. We search the dropout rate over $\{0, 0.1, 0.3, 0.5\}$ and the hidden dimension over $\{128, 256\}$. When applicable, the number of encoder layers is selected from $\{1, 2, 3\}$, while the predictor uses two layers. Method-specific architectural settings not covered by this common search space follow their original implementations. For model selection, we use validation MRR computed with 500 negative candidates per validation edge selected by HeaRT. We use HeaRT as a common protocol for model selection so that the downstream models are fixed before comparing different candidate construction protocols. Once selected, each trained model is kept unchanged across all evaluation protocols, allowing us to isolate the effect of candidate construction rather than differences in model training or tuning.

B.3.5 Evaluation Settings

For computational tractability, we randomly sample 100,000 positive test edges when a dataset contains more than 100,000 test positives. This follows the subsampling practice used by Li et al. (2023) for large-scale evaluation. The same sampled test edges are used across all candidate construction protocols to ensure a fair comparison.

B.4 MRR Comparison Among Screening Models

Table 6 reports the MRR and pairwise-gap deviations from full-space evaluation for different screening models using the metrics defined in Appendix B.2, while Figure 5 presents their downstream MRRs. The broadly consistent results indicate that the benefit of learned screening is not specific to CAM, while CAM remains comparable to the alternatives with a simpler fixed design.

Table 6: Comparison of negative-screening models in the model-screened protocol. Absolute deviations are reported in percentage points and normalized deviations in percentages. Lower values indicate closer agreement with full-space evaluation.

Metric	Selector	Squirrel	Photo	Computers	Facebook	ogbl-ddi	ogbl-ppa	ogbl-citation2
MRR-Diff	**CAM**	**0.33**	**0.34**	**0.39**	**0.47**	**0.16**	**1.0**	**0.74**
	MLP	0.35	0.41	0.41	0.88	0.22	1.2	2.4
	GCN	0.31	0.33	0.32	0.47	0.15	0.85	0.57
	SEAL	0.31	0.33	0.30	0.35	0.15	0.77	0.46
Normalized MRR-Diff	**CAM**	**11.3**	**7.3**	**12.8**	**6.0**	**19.5**	**105**	**39.6**
	MLP	11.9	8.8	13.5	11.2	25.8	116	128
	GCN	10.5	7.1	10.6	6.0	17.7	85.6	30.6
	SEAL	10.7	7.1	10.0	4.5	18.4	76.9	24.7
Gap-Diff	**CAM**	**0.25**	**0.25**	**0.30**	**0.40**	**0.08**	**0.67**	**0.31**
	MLP	0.25	0.24	0.26	0.39	0.09	0.67	1.0
	GCN	0.25	0.27	0.32	0.46	0.09	0.74	0.28
	SEAL	0.25	0.27	0.32	0.36	0.09	0.76	0.41
Normalized Gap-Diff	**CAM**	**19.1**	**8.1**	**13.0**	**10.6**	**23.0**	**66.7**	**19.2**
	MLP	18.6	7.9	11.5	10.4	27.7	67.0	63.3
	GCN	18.6	8.9	13.8	12.1	25.3	73.6	17.2
	SEAL	18.5	8.8	14.0	9.4	26.4	75.9	25.3

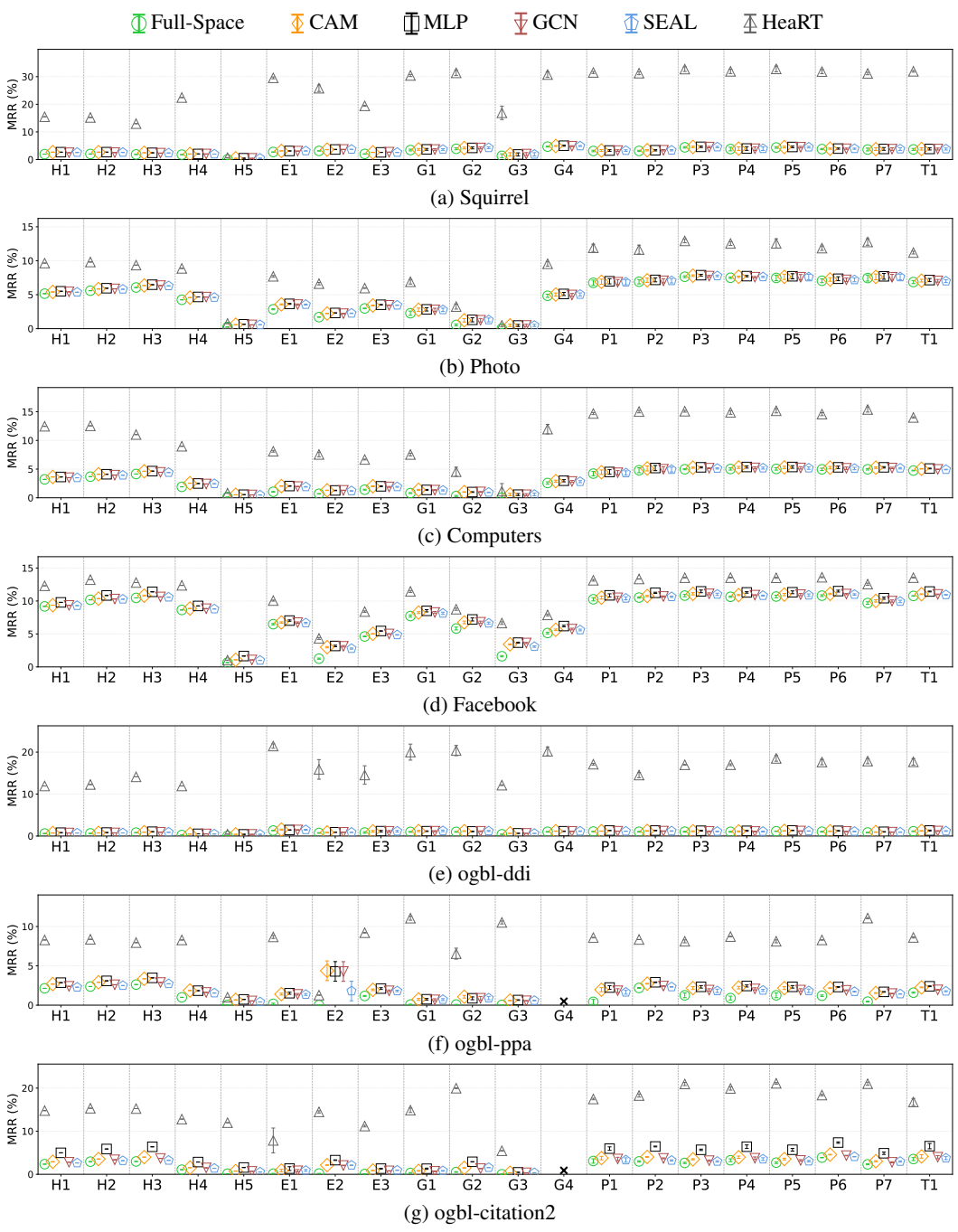


Figure 5: MRR comparison among different models used in model-screened evaluation. For visual clarity, markers within each method are horizontally offset to avoid overlap; only their vertical positions encode the metric values. Vertical black grid lines separate different methods. × denotes an out-of-memory error (e.g., G4 on ogbl-citation2).

B.5 HITS COMPARISON

Figure 6 reports Hits@K for varying K. The results exhibit trends consistent with those observed for MRR in Section 5.2 and confirm that the effect of candidate construction is not specific to MRR.

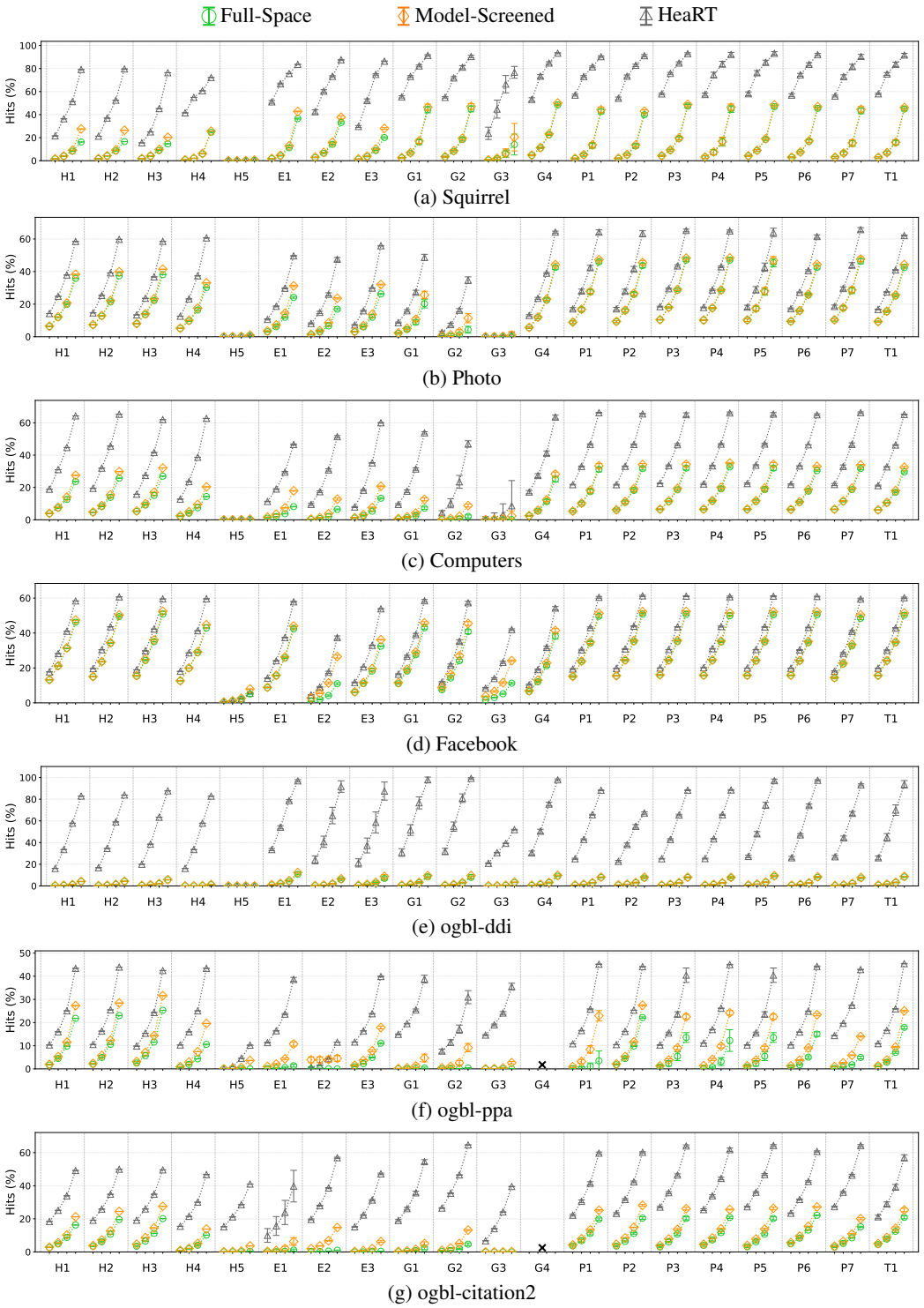
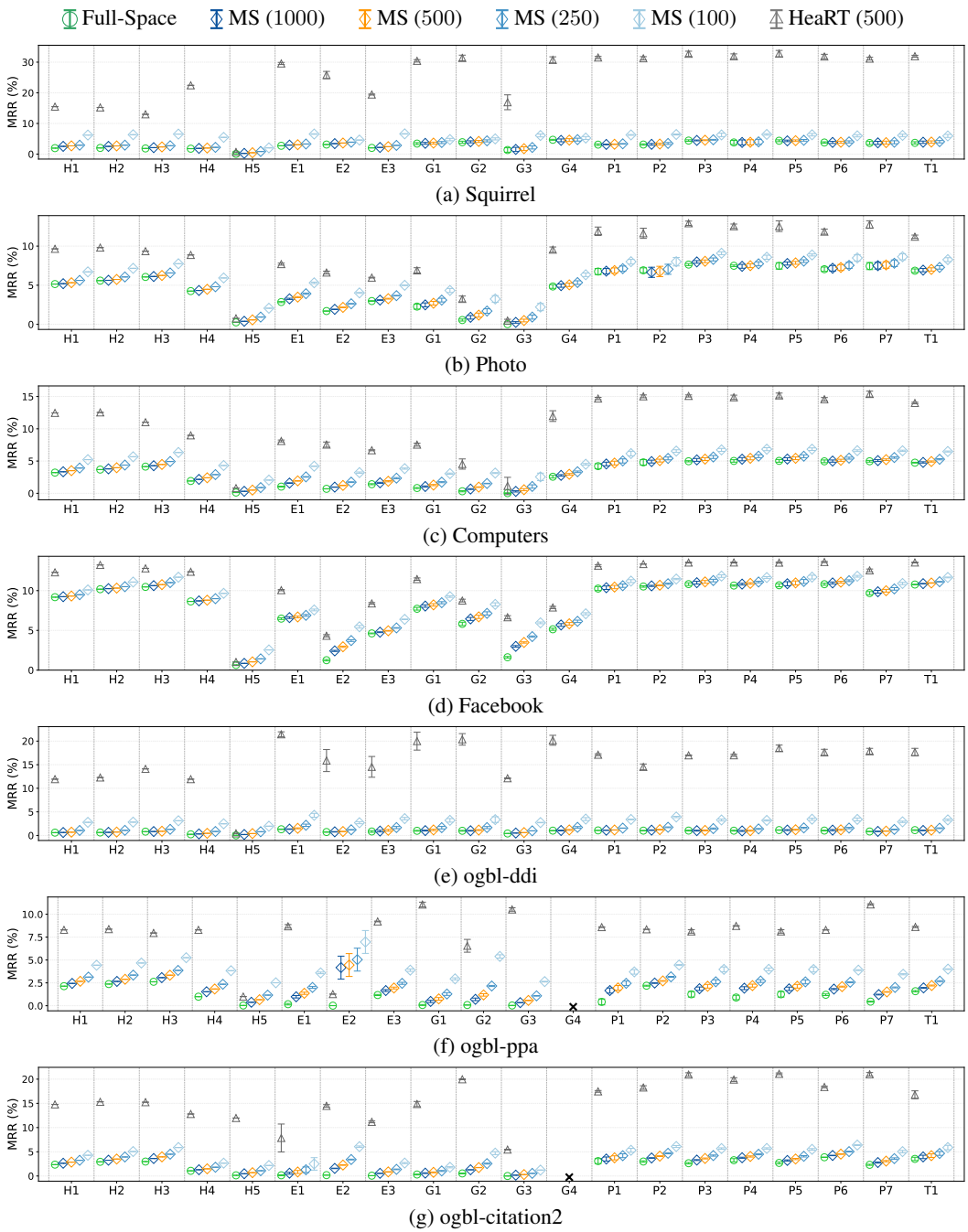


Figure 6: Hits@K comparison across datasets. For each method, the points from left to right indicate $K = 5, 10, 20, 50$. $\times$ denotes an out-of-memory error (e.g., G4 on ogbl-citation2).

B.6 VARYING NUMBER OF CANDIDATES

Figure 7 shows that model-screened evaluation generally approaches full-space evaluation as M increases. Even with $M = 100$, it is closer to full-space than HeaRT with $M = 500$ for most methods.



(a) Squirrel

(b) Photo

(c) Computers

(d) Facebook

(e) ogbl-ddi

(f) ogbl-ppa

(g) ogbl-citation2

Figure 7: MRR comparison among model-screened, HeaRT, and full-space evaluation, where model-screened uses $M \in \{100, 250, 500, 1000\}$ and is denoted as MS (M). For visual clarity, markers within each method are horizontally offset to avoid overlap; only their vertical positions encode the metric values. Vertical black grid lines separate different methods. $\times$ denotes an out-of-memory error (e.g., G4 on ogbl-citation2).