ONE INTERACTION, TWO CONSEQUENCES: RETHINKING THE COMPUTATIONAL ROLE OF EDGES IN GRAPH LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

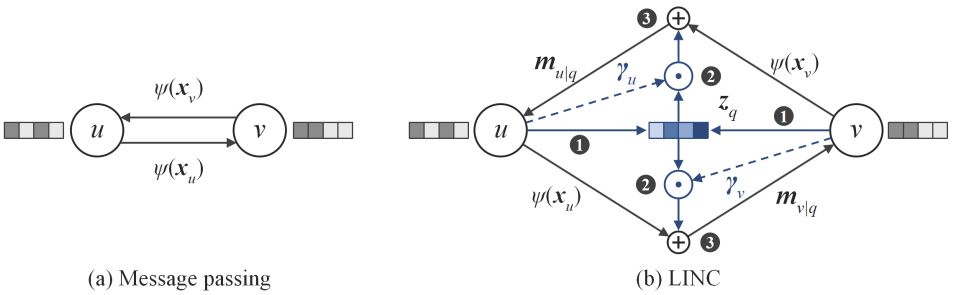
Graph neural networks (GNNs) are widely used to learn from graphs, and most of their designs focus on what information a node should receive. In real graph data, however, an edge usually records an interaction, such as a paper written by a student and a professor, and the same interaction can mean different things to the two participants. Such designs use an edge only to shape the message sent to one receiving node and do not model the interaction as one fact with a consequence for each participant. Graphs that differ in homophily, heterogeneity, and directionality have been handled by separate specialized GNNs that fix the meaning of an edge in advance. We instead make the interaction itself the unit of computation and propose a simple model, Learning from Interaction Consequences (LINC). LINC forms one representation of an interaction from both participants, and the two participants use this representation through scales computed from their own embeddings. As a result, one interaction has two consequences, one for each participant. Since a consequence also carries the other participant's features, ordinary message passing is a special case of LINC. We prove that the interaction representation of an undirected edge does not depend on the order of the participants, while the two consequences can still differ. On 16 benchmarks covering homophilic, heterophilic, heterogeneous, and directed graphs, LINC uses one architecture throughout and achieves the best average rank in every group against 38 baselines. In ablation studies, the interaction representation alone carries most of what a node needs from its neighbors. Different consequences for the two participants help on most graphs, especially on directed and heterophilic graphs, where the participants often differ in role or label.

1 INTRODUCTION

Graph neural networks (GNNs) have been widely applied to social networks (Borisyuk et al., 2024), e-commerce (Chen et al., 2024), financial transactions (Egressy et al., 2024), and biomedicine (Huang et al., 2024), where graph data are large in scale and diverse in form (homophilic and heterophilic, homogeneous and heterogeneous, undirected and directed). Most of the progress in GNNs can be viewed as answers to the question of *what information a node should receive*. Over the years, the range from which a node receives information has grown from its neighborhood to the whole graph. Neighborhood aggregation passes the transformed features of neighbors to a node (Kipf & Welling, 2017; Hamilton et al., 2017), attention decides how much a node takes from each neighbor (Veličković et al., 2018; Brody et al., 2022), and heterophilic GNNs change from whom and how a node receives information when linked nodes tend to be dissimilar (Zhu et al., 2020; Qiu et al., 2026). Relations, the receiving node, and edge directions further decide the form in which a message arrives (Schlichtkrull et al., 2018; Brockschmidt, 2020; Rossi et al., 2024). Graph transformers go one step further and allow each node to attend to all other nodes (Ying et al., 2021; Deng et al., 2024).

Whether the range is a neighborhood or the whole graph, the graph enters these models through its edges. Specifically, an edge serves as a route (Kipf & Welling, 2017), a weight (Veličković et al., 2018), a condition (Schlichtkrull et al., 2018), or a state (Battaglia et al., 2018) for the messages addressed to one receiving node. Building on this view of edges, specialized GNNs have been

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.



(a) Message passing (b) LINC

Figure 1: Illustration of message passing and our proposed LINC. (a) In message passing, the two directions of an edge carry two messages, and each message is transformed from one endpoint by a map ψ. (b) In LINC, one interaction gives each participant its own consequence, built from one shared interaction representation (blue) and the features of the other participant (black). ❶ The two participants form one interaction representation $\boldsymbol{z}_q$. ❷ Each participant scales $\boldsymbol{z}_q$ with a vector computed from its own embedding (dashed, $\boldsymbol{\gamma}_u$ or $\boldsymbol{\gamma}_v$). ❸ The scaled representation is added to ψ of the other participant, which gives the consequence for that participant ($\boldsymbol{m}_{u|q}$ or $\boldsymbol{m}_{v|q}$).

developed for graphs that differ in homophily (Gu et al., 2025), heterogeneity (Li et al., 2026), and directionality (Pahng & Hormoz, 2025).

However, graph data in the real world are formed in a different way. An edge usually records an event or a relationship in which two nodes take part together, such as a purchase or a collaboration. We call such an edge an *interaction* and the two nodes its *participants*. An interaction has a consequence for each participant, and the two consequences often differ. For instance, a purchase tells us what the user wants and whom the product attracts (Hou et al., 2026). Node types and edge directions make this difference easy to see, but it also exists in graphs without them. A paper written by a student and a professor is an undirected edge between two nodes of the same type, yet it tells us which area the student is entering and whom the professor is training (Hu et al., 2020a). More generally, useful signals for a node come from its neighbors and global features, and also from why two nodes are connected and what the connection means to each of them. Better fusion of neighbor or global features alone may not extract such signals.

The observations above reveal a gap between how models are designed and how graph data are formed. Models ask what a node should receive, whereas graph data record which interactions a node takes part in. Existing designs do use both endpoints of an edge, for example to weight a neighbor (Veličković et al., 2018; Brody et al., 2022), to modulate a message (Brockschmidt, 2020), to compute a message from both the sender and the receiver (Gilmer et al., 2017), or to pass a representation of the edge to the receiving node (Jiang et al., 2019; Barceló et al., 2026). In all these designs, the two endpoints only serve the message that one receiving node gets. As a result, the interaction is reduced to a message for one node (Figure 1a) and is not modeled as one fact with a consequence for each participant. Instead, each kind of graph has been addressed separately, and each line of work fixes the meaning of an interaction in advance through a single factor, such as similarity (Kipf & Welling, 2017), difference (Zhu et al., 2020; Bo et al., 2021), types (Schlichtkrull et al., 2018; Zhou et al., 2023), or directions (Zhang et al., 2021b; Rossi et al., 2024). Viewed through interactions, these lines of work are separate responses to the same gap.

To bridge this gap, we start from the interaction itself and ask two questions. ***What does each interaction tell each of its two participants? Starting only from the interactions recorded in real graph data, can a simple model learn more useful node representations?*** To answer these questions, we propose LINC (**L**earning from **IN**teraction **C**onsequences, Figure 1b), which learns node representations from the consequences of interactions for their participants. For every edge, LINC forms one representation of the interaction from the embeddings of both participants. The two participants then use this representation through scales computed from their own embeddings, and the same interaction therefore has two consequences, one for each participant. Since each consequence also carries the other participant's features, ordinary message passing is the special case of LINC without the interaction representation. The same layer applies to graphs of any homophily,

heterogeneity, and directionality without changing its form. Moreover, LINC computes no attention weights, carries no edge states between layers, and needs neither metapaths nor relation-specific weight matrices.

Our main contributions are summarized as follows.

- **New perspective.** We point out a gap between how GNNs are designed and how graph data are formed, and we view an edge as an interaction with one consequence per participant. Under this view, one model can serve graphs of any homophily, heterogeneity, and directionality, which specialized GNNs have handled separately.
- **Proposed method.** We propose LINC, a simple yet effective model where the two participants of an interaction share one representation and use it through scales computed from their own embeddings.
- **Theoretical properties.** We prove that the representation of an undirected interaction is invariant to the order of the participants while the two consequences can differ, and that ordinary message passing is a special case of LINC.
- **Empirical validation.** With the same architecture on every graph, LINC achieves an average rank of 1.88 among 20 methods on eight homophilic and heterophilic graphs, 1.25 among 16 on four heterogeneous graphs, and 1.25 among eight on four directed graphs. In ablation studies, the interaction representation alone carries most of what a node needs from its neighbors, and the two participants benefit from different consequences.

2 Related Work

Computational roles of an edge. A message-passing layer updates a node with the messages it receives along its edges (Gilmer et al., 2017), and existing models give the edge one or more of four computational roles in these messages. As a *route*, the edge only decides which neighbors send messages to a node, as in classic GNNs (Kipf & Welling, 2017; Hamilton et al., 2017) and many heterophilic GNNs (Zhu et al., 2020; Qiu et al., 2026). As a *weight*, the edge scales the message of a neighbor by a coefficient computed from the two endpoints, and this coefficient can be positive (Veličković et al., 2018) or signed (Bo et al., 2021). As a *condition*, the edge selects or modulates the transformation of a message according to edge attributes (Simonovsky & Komodakis, 2017), relations and node types (Schlichtkrull et al., 2018; Zhou et al., 2023), directions (Rossi et al., 2024), the embedding of the receiving node (Brockschmidt, 2020), or learned maps at both endpoints whose outputs are pulled toward agreement by sheaf diffusion (Bodnar et al., 2022). As a *state*, the edge carries its own representation across layers and passes it to both endpoints (Battaglia et al., 2018; Barceló et al., 2026). In all four roles, the edge shapes the message that one receiving node gets, and a model may combine several roles. In contrast, LINC rethinks the role of the edge and treats it as an interaction, which is one fact shared by two participants. Each participant then takes its own consequence from this shared fact.

Joint treatments of the pair. When a task involves the pair itself, models treat the two endpoints of an edge together. For example, physics-informed interaction models give an edge two effects tied by conservation laws (Han et al., 2022; Sharma & Fink, 2026), temporal interaction models update both participants at every event with one message per participant (Kumar et al., 2019; Rossi et al., 2020), and models for link prediction and recommendation take the pair as the target and score it from the two endpoints (He et al., 2020; Chamberlain et al., 2023). Each of these lines keeps the part of an interaction that its task needs, that is, two effects fixed by a law, two updates computed separately, or one score for the pair. Both lines of work have become more specialized, and the simpler question of what one interaction tells each of its two participants has been passed over. LINC answers this question directly by learning the two consequences of each interaction from one shared representation and using them to update both participants. Please refer to Appendix A for additional related work.

3 Preliminaries

3.1 Notation

We use bold italic letters for matrices and vectors ($\boldsymbol{W}$, $\boldsymbol{x}$), calligraphic letters for sets and graphs ($\mathcal{V}$, $\mathcal{G}$), italic letters for scalars and maps (n, τ), and upright letters for fixed functions (GELU).

We denote a graph as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{Q}, \mathcal{X}, \tau, \phi)$ of the node set $\mathcal{V}$ with $n = |\mathcal{V}|$, the interaction set $\mathcal{Q}$ with $m = |\mathcal{Q}|$, the raw features $\mathcal{X} = \{\boldsymbol{x}_i^{\text{raw}}\}_{i \in \mathcal{V}}$, whose dimension may differ between node types, the node type map $\tau : \mathcal{V} \to \mathcal{T}$, and the relation map $\phi : \mathcal{Q} \to \mathcal{R}$, where a relation is an edge type. Each node i has a label $y_i \in \mathcal{Y}$.

Definition 1 (Interaction)**.** An *interaction* is an edge $q = (u, v) \in \mathcal{Q}$ with $u \neq v$, written as an ordered pair whose *participants* u and v take the roles of source and target. It is *unordered* if the two roles are interchangeable, and we then write $q = \{u, v\}$.

Properties of graph data. A graph $\mathcal{G}$ is *homogeneous* if $|\mathcal{T}| = |\mathcal{R}| = 1$ and *heterogeneous* otherwise. It is *undirected* if all of its interactions are unordered and *directed* otherwise. We measure the *homophily* of $\mathcal{G}$ by the adjusted homophily $\hom(\mathcal{G})$ of Platonov et al. (2023a), which is high on homophilic graphs and close to zero or negative on heterophilic graphs (see Appendix C.2 for its formula). Node types, relations, and the source and target roles are given as input. In contrast, $\hom(\mathcal{G})$ depends on the labels to be predicted, and a model cannot know it in advance.

3.2 Problem definition

Problem 1 (Node classification across graph properties)**.** Given a graph $\mathcal{G}$ of any homophily, heterogeneity, and directionality, and the labels $\{y_i\}_{i \in \mathcal{V}_L}$ of a node set $\mathcal{V}_L \subset \mathcal{V}$, learn a model F_θ, with the same form for every such $\mathcal{G}$, whose output $F_\theta(\mathcal{G})_i$ predicts y_i for every $i \in \mathcal{V} \setminus \mathcal{V}_L$.

4 LINC: Learning from Interaction Consequences

In this section, we introduce LINC, whose unit of computation is the interaction. In each layer, every interaction gives one consequence to each participant, and every node is updated from the consequences of the interactions it takes part in (Figure 1b and Section 4.3). A consequence has two terms. The direct term carries the features of the other participant, and the interaction term describes what the interaction means to the receiving participant. The interaction term comes from one representation formed from both participants (Section 4.1) and the own scale of the receiving participant (Section 4.2). All matrices and bias vectors below are learnable, and LN, Drop, and $\odot$ are layer normalization, dropout, and the Hadamard product.

4.1 One representation per interaction

Since an interaction is one fact shared by two participants, we form its representation once from both of them. We first embed the raw features with one input map per node type, $\boldsymbol{h}_i^0 = \mathrm{GELU}(\mathrm{Drop}(\boldsymbol{W}_{\tau(i)}\boldsymbol{x}_i^{\text{raw}} + \boldsymbol{b}_{\tau(i)}))$. At layer ℓ, node i has the embedding $\boldsymbol{h}_i^\ell \in \mathbb{R}^d$, where d is the hidden dimension, and the layer works on the normalized embedding $\boldsymbol{x}_i^\ell = \mathrm{LN}(\boldsymbol{h}_i^\ell)$. We drop the layer index ℓ below. Given an interaction $q = (u, v)$, we project the source and the target with the role matrices $\boldsymbol{P}_{\mathrm{s}}, \boldsymbol{P}_{\mathrm{t}} \in \mathbb{R}^{r \times d}$, where r is the interaction dimension, and combine them into the *interaction representation*

$$\boldsymbol{z}_q = \sqrt{r}\ \mathrm{norm}(\boldsymbol{P}_{\mathrm{s}}\boldsymbol{x}_u) \odot \mathrm{norm}(\boldsymbol{P}_{\mathrm{t}}\boldsymbol{x}_v) \in \mathbb{R}^r, \tag{1}$$

where $\mathrm{norm}(\boldsymbol{a}) = \boldsymbol{a}/\max(\|\boldsymbol{a}\|_2, \varepsilon)$ scales a vector to unit Euclidean length and ε is a small constant. For an unordered interaction, the two role matrices are tied, i.e., $\boldsymbol{P}_{\mathrm{s}} = \boldsymbol{P}_{\mathrm{t}} = \boldsymbol{P}$. Due to the normalization, $\boldsymbol{z}_q$ depends only on the directions of the two projections, and the sum of its entries equals $\sqrt{r}$ times their cosine similarity. In other words, $\boldsymbol{z}_q$ keeps entry by entry what a similarity score would collapse into a single number. Moreover, since each row of $\boldsymbol{P}_{\mathrm{s}}$ and $\boldsymbol{P}_{\mathrm{t}}$ mixes all features, every feature of one participant can interact with every feature of the other in $\boldsymbol{z}_q$.

4.2 Two consequences per interaction

An interaction often has a different consequence for each participant. To capture this, we first map $\boldsymbol{z}_q$ to the content $\boldsymbol{c}_q$ shared by both participants. Each participant i then scales this content feature by feature with its *own scale* $\boldsymbol{\gamma}_i$, computed from its embedding,

$$\boldsymbol{c}_q = \boldsymbol{B}\boldsymbol{z}_q, \qquad \boldsymbol{\gamma}_i = \mathbf{1} + \boldsymbol{C}\frac{\boldsymbol{x}_i}{\sqrt{d}}, \tag{2}$$

where $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ are shared by all interactions and nodes, $\mathbf{1}$ is the all-ones vector, and the factor $1/\sqrt{d}$ keeps the scales close to one at initialization. As a result, what a participant takes from an interaction depends on its own embedding. Formally, the two consequences of q are

$$\boldsymbol{m}_{u|q} = \mathrm{GELU}(\boldsymbol{D}_{\mathrm{s}}\boldsymbol{x}_v + \boldsymbol{\gamma}_u \odot \boldsymbol{c}_q)\,, \qquad \boldsymbol{m}_{v|q} = \mathrm{GELU}(\boldsymbol{D}_{\mathrm{t}}\boldsymbol{x}_u + \boldsymbol{\gamma}_v \odot \boldsymbol{c}_q)\,, \tag{3}$$

where $\boldsymbol{D}_{\mathrm{s}}, \boldsymbol{D}_{\mathrm{t}} \in \mathbb{R}^{d\times d}$, and $\boldsymbol{D}_{\mathrm{s}} = \boldsymbol{D}_{\mathrm{t}} = \boldsymbol{D}$ for an unordered interaction. Inside the GELU, the first term $\boldsymbol{D}_{\mathrm{s}}\boldsymbol{x}_v$, the *direct term*, carries the features of v to u, like a message in message passing. The second term $\boldsymbol{\gamma}_u \odot \boldsymbol{c}_q$, the *interaction term*, captures what the interaction means to u. Each consequence therefore brings together the two sides of the gap in Section 1, that is, what a node receives and which interactions it takes part in. The own scales make the same interaction mean different things to its two participants. Without them ($\boldsymbol{C} = \mathbf{0}$), both participants would take the same interaction term, and their consequences would differ only through the direct term.

Unlike an edge state, the interaction representation is not carried across layers and is recomputed from both participants at every layer. The own scale $\boldsymbol{\gamma}_u$ works like a weight or a condition, but it acts on the interaction instead of the neighbor. In message passing, each direction of an edge gets a separate message (Figure 1a), and models for directed and heterogeneous graphs add a message along the reverse of each edge (Schlichtkrull et al., 2018; Rossi et al., 2024). In contrast, LINC stores each interaction only once, in either order if it is unordered, and updates both participants from the same $\boldsymbol{z}_q$. To show that this order does not matter while the two consequences can still differ, we prove the following theorem in Appendix B.1.

Theorem 1 (Exchange symmetry and two consequences)**.** *Let $q = \{u, v\}$ be an unordered interaction. For every parameter setting, exchanging $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ leaves $\boldsymbol{z}_q$ unchanged and exchanges $\boldsymbol{m}_{u|q}$ and $\boldsymbol{m}_{v|q}$. The interaction terms $\boldsymbol{\gamma}_u \odot \boldsymbol{c}_q$ and $\boldsymbol{\gamma}_v \odot \boldsymbol{c}_q$ are equal for all inputs when $\boldsymbol{C} = \mathbf{0}$ and can differ when $\boldsymbol{C} \neq \mathbf{0}$.*

Theorem 1 shows that the interaction representation is invariant and the pair of consequences equivariant. As a result, an undirected edge can have different consequences for its participants, even though neither is marked as source or target. To show that each consequence depends on both participants, unlike a message from the neighbor, we prove the following proposition in Appendix B.2.

Proposition 1 (Dependence on both participants)**.** *Fix the embedding $\boldsymbol{x}_v$ of the neighbor. The message $\mathrm{GELU}(\boldsymbol{D}_{\mathrm{s}}\boldsymbol{x}_v)$ of the direct term is the same for every embedding $\boldsymbol{x}_u$ of u, while the consequence $\boldsymbol{m}_{u|q}$ can change with $\boldsymbol{x}_u$.*

4.3 Aggregation and update

Every interaction now has one consequence for each participant. Node i collects the consequences of the n_i interactions $\mathcal{Q}(i) = \{q \in \mathcal{Q} \mid i \in q\}$ it takes part in and averages them with its embedding,

$$\boldsymbol{g}_i = \frac{\boldsymbol{D}_0\boldsymbol{x}_i + \sum_{q\in\mathcal{Q}(i)} \omega_{i,q}\, \boldsymbol{m}_{i|q}}{1 + n_i} + \boldsymbol{b}, \qquad \omega_{i,q} = \frac{n_i}{K_i\, n_{i,\phi(q)}}, \tag{4}$$

where $\boldsymbol{D}_0\boldsymbol{x}_i$ is the self term with $\boldsymbol{D}_0 \in \mathbb{R}^{d\times d}$, $\boldsymbol{b} \in \mathbb{R}^d$ is a bias, K_i is the number of distinct relations in $\mathcal{Q}(i)$, and $n_{i,\phi(q)}$ is the number of interactions in $\mathcal{Q}(i)$ with the same relation as q. These relation-balanced weights add up to n_i and give every relation at node i the same total weight, which keeps a frequent relation from dominating the average. On a homogeneous graph, every $\omega_{i,q}$ equals one, and Eq. (4) averages over node i and its neighbors with equal weights, like the row-normalized adjacency matrix with self-loops. Relations enter the layer only through these weights and node types only through the input maps, while all other matrices are shared across relations. Finally, a residual block with $\boldsymbol{W}_o \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_o \in \mathbb{R}^d$ updates the embedding,

$$\boldsymbol{h}_i^{\ell+1} = \boldsymbol{h}_i^{\ell} + \mathrm{Drop}_2[\boldsymbol{W}_o\,\mathrm{GELU}(\mathrm{Drop}_1(\boldsymbol{g}_i)) + \boldsymbol{b}_o]\,. \tag{5}$$

After L layers, we compute the logits as $\boldsymbol{W}_{\text{cls}}\,\text{LN}(\boldsymbol{h}_i^L) + \boldsymbol{b}_{\text{cls}}$ with $\boldsymbol{W}_{\text{cls}} \in \mathbb{R}^{|\mathcal{Y}|\times d}$, and we train the model with the cross-entropy loss on the labeled nodes. Without the interaction term, a consequence depends on the other participant alone and acts as a message. To show that ordinary message passing is then a special case of LINC, we prove the following proposition in Appendix B.3.

Proposition 2 (Reduction to message passing)**.** *With $\boldsymbol{B} = \boldsymbol{0}$, the layer equals the message-passing layer with messages $\boldsymbol{m}_{u|q}^{(0)} = \text{GELU}(\boldsymbol{D}_{\text{s}}\boldsymbol{x}_v)$ and $\boldsymbol{m}_{v|q}^{(0)} = \text{GELU}(\boldsymbol{D}_{\text{t}}\boldsymbol{x}_u)$ and with the same self term, aggregation, and residual block.*

The layer is equivariant to type-preserving node permutations (see Proposition 3 in Appendix B.4).

Why the GELU acts on each interaction. If the GELU were moved after the sum, all interactions of a node would collapse into one product with the summed neighborhood. The layer could then no longer distinguish two neighborhoods with the same size and sum (see Proposition 4 in Appendix B.5).

Complexity. We compute all projections of node embeddings once per node and gather them for the interactions. With hidden dimension d and interaction dimension r, a layer takes $O(nd^2+ndr+mrd)$ time and $O((n+m)(d+r))$ activation memory, both linear in the numbers of nodes and interactions.

5 Experiments

We conduct experiments to answer four research questions. **RQ1:** How does LINC compare with state-of-the-art methods across graph properties? **RQ2:** Where does the gain of LINC come from? **RQ3:** How sensitive is LINC to its hyperparameters? **RQ4:** How efficient is LINC?

5.1 Experimental settings

Datasets. We evaluate LINC on 16 benchmarks in three groups. The undirected graphs include two homophilic graphs (Amazon-Photo and WikiCS) and six heterophilic graphs (Chameleon-F, Roman-Empire, Amazon-Ratings, Minesweeper, Tolokers, and Questions), where "-F" denotes the filtered version of Platonov et al. (2023b). The heterogeneous graphs are IMDB and Freebase from HGB (Lv et al., 2021) and the multiplex graphs Alibaba (Fu et al., 2023) and Taobao (Yu et al., 2022). The directed graphs are Cora-ML (Bojchevski & Günnemann, 2018), the directed Roman-Empire (Rossi et al., 2024), MGTAB (Shi et al., 2023), and ogbn-arxiv (Hu et al., 2020a).

Baselines. On undirected graphs, we compare LINC with 19 baselines, including MLP, classic GNNs, attention models, heterophilic GNNs, graph transformers, and edge-aware models, where GCN, GAT, and GraphSAGE use the stronger implementations of Platonov et al. (2023b) with residual connections and normalization. On heterogeneous graphs, we compare with 15 baselines, including homogeneous GNNs, metapath-based and metapath-free models, and multiplex models. On directed graphs, we compare with seven directed GNNs. Appendix C describes all datasets and baselines.

Training and evaluation. To compare methods as fairly as possible, all methods within each table share the same data, splits, and evaluation code, and the splits follow prior work. For each baseline, we use the configuration reported by its authors for the same dataset version, split, and task, and otherwise tune it over the authors' search space under a cap of 128 configurations, or five on ogbn-arxiv. We tune LINC under the same caps (Appendix C.5). All methods select hyperparameters on the validation set. We report the mean and standard deviation over runs and the average rank.

5.2 Main results

To answer RQ1, we compare LINC with the baselines on the three groups of graphs. As shown in Tables 1, 2, and 3, LINC achieves the best average rank on **homophilic and heterophilic graphs**, **heterogeneous graphs**, and **directed graphs**. With the same architecture, it ranks in the top three on all 16 graphs and first on 10 of them. In contrast, the strongest baseline changes from graph to graph. For example, GNN-FiLM ranks second on Roman-Empire and Minesweeper but 18th on Chameleon-F. In the second metric, LINC also ranks first on average in Macro-F1 on undirected and directed graphs (Appendix D.1) and second in Micro-F1 on heterogeneous graphs.

Discussion. LINC gains the most on graphs where an interaction means different things to its two participants. On the two homophilic graphs, where neighbors mostly share a label, LINC stays within

Table 1: Node classification results on homophilic and heterophilic graphs (mean ± standard deviation, %). The top rows give the size and the adjusted homophily (hom) of each graph. We report accuracy, except ROC-AUC for Minesweeper, Tolokers, and Questions. Best and second-best results are in **bold** and underlined. OOM (out of memory) is unranked. EB-GNN† adds a node-classification head to the edge encoder and edge-to-node readout of Barceló et al. (2026).

	Homophilic		Heterophilic						
Method	Amazon-Photo	WikiCS	Chameleon-F	Roman-Empire	Amazon-Ratings	Minesweeper	Tolokers	Questions	Avg. rank
# Nodes	7,650	11,701	890	22,662	24,492	10,000	11,758	48,921	
# Edges	119,081	215,603	8,854	32,927	93,050	39,402	519,000	153,540	
hom	0.79	0.58	0.03	−0.05	0.14	0.01	0.09	0.02	
MLP	71.09±1.94	67.59±1.30	28.92±4.00	61.67±0.59	37.96±0.35	50.32±1.22	70.43±0.76	60.81±4.02	17.38
GCN	94.92±0.08	78.88±0.57	**43.66±3.52**	72.86±0.57	48.91±0.58	89.88±0.54	82.85±0.83	76.25±1.19	7.25
GraphSAGE	95.36±0.07	78.65±0.43	39.72±5.43	82.75±0.48	**53.57±0.65**	90.87±0.60	82.28±0.41	75.48±0.92	5.88
APPNP	94.20±0.10	77.05±0.69	36.63±2.49	51.02±0.81	41.65±0.49	67.11±1.13	65.21±1.20	53.17±1.32	15.31
GAT	94.20±0.04	79.09±0.79	41.05±3.42	80.14±0.42	49.13±0.67	90.32±0.61	83.53±0.45	77.15±0.86	6.56
GATv2	82.18±3.76	78.53±0.69	35.36±3.81	48.19±1.12	41.26±0.38	72.49±1.16	69.65±1.12	71.44±1.27	15.00
GATE	93.12±0.14	77.72±0.52	37.47±2.30	72.65±0.66	48.96±0.69	89.44±1.81	81.15±0.69	73.63±0.95	10.75
MixHop	92.77±0.04	76.71±0.49	39.27±3.12	38.72±0.72	41.60±0.45	80.05±1.12	64.80±1.16	50.98±1.12	15.62
GPR-GNN	95.12±0.21	79.45±0.60	32.75±6.17	65.37±0.61	44.76±0.49	86.87±0.81	68.76±1.12	57.36±0.68	12.75
FSGNN	96.34±0.17	80.14±0.47	38.63±4.50	79.06±0.63	44.77±0.92	90.38±0.56	84.33±0.84	77.88±1.34	5.75
GloGNN	95.80±0.10	71.36±0.97	40.65±4.12	67.05±0.88	51.17±0.32	54.73±1.42	81.16±0.92	75.76±1.26	10.25
ACM-GCN	**96.54±0.07**	75.50±2.13	39.45±4.22	77.75±0.69	41.18±0.32	90.72±0.57	82.07±0.95	73.60±0.87	8.75
GAMMA	96.06±0.08	61.44±2.81	37.72±2.74	39.36±2.58	36.79±0.01	90.67±0.59	79.81±0.79	66.74±5.10	13.44
SGFormer	96.06±0.32	79.28±0.75	42.82±4.12	76.24±0.63	52.14±0.53	84.34±0.82	82.95±0.90	77.94±1.35	5.56
CoBFormer	95.25±0.37	**80.39±0.66**	42.08±3.89	65.95±0.45	52.45±0.42	59.52±1.26	81.44±0.79	73.51±0.94	8.50
CensNet	90.78±0.75	69.06±7.99	33.75±3.23	57.33±1.63	44.11±0.64	85.36±3.20	66.10±7.52	69.14±5.98	15.38
GNN-FiLM	95.62±0.36	75.15±0.82	29.44±4.03	89.50±0.45	46.46±0.80	93.99±0.66	81.96±1.68	77.14±1.94	8.38
EGAT	94.47±0.16	74.25±1.79	27.84±8.41	69.75±1.23	36.73±2.31	91.66±0.57	84.37±0.65	72.51±1.74	12.00
EB-GNN†	OOM	OOM	37.07±3.32	81.01±0.63	47.87±0.68	92.45±0.71	OOM	OOM	7.25
LINC	96.45±0.16	79.80±0.62	42.73±2.99	**91.40±0.65**	52.40±0.61	**95.47±0.55**	**86.03±0.81**	**79.41±1.02**	**1.88**

Table 2: Node classification results on heterogeneous graphs (mean ± standard deviation, %). Top rows give graph sizes, counting inverse relations. We report Macro-F1 (Ma-F1) and Micro-F1 (Mi-F1). Best and second-best results are in **bold** and underlined. OOM (out of memory) is unranked.

	IMDB		Freebase		Alibaba		Taobao		Avg. rank	
Method	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1
# Nodes	21,420		180,098		15,218		21,318			
# Edges	86,642		1,057,688		54,024		41,194			
# Relations	6		36		6		8			
MLP	51.19±1.73	61.10±0.44	9.44±0.00	49.32±0.00	19.49±1.24	28.84±0.78	18.81±0.00	39.30±0.00	14.75	10.00
GCN	58.87±0.73	65.35±0.37	31.14±0.40	60.37±0.29	29.88±0.26	35.69±0.32	20.33±3.13	39.20±0.22	9.75	5.75
GAT	57.10±4.54	64.20±2.51	40.59±3.14	63.47±0.87	12.51±2.72	19.86±4.05	22.14±3.09	37.96±2.37	11.25	9.00
R-GCN	55.64±0.98	58.84±0.76	45.87±0.83	56.03±2.39	31.88±0.56	36.92±1.07	30.00±1.42	35.87±2.43	6.50	9.75
HAN	57.45±0.66	64.23±0.40	21.48±0.96	55.36±1.47	19.87±2.98	30.39±2.69	20.19±1.97	**39.30±0.58**	12.25	8.00
HetGNN	48.02±1.07	51.20±0.97	20.91±1.09	36.77±0.70	28.55±0.91	31.03±1.04	**35.02±2.68**	36.27±2.67	10.25	12.25
RSHN	54.74±3.13	58.34±2.15	37.18±2.68	59.91±0.56	30.43±1.49	35.03±1.60	28.19±2.75	38.01±1.95	8.75	8.75
HetSANN	48.81±2.28	56.60±1.03	21.26±3.67	57.59±1.02	20.77±5.43	30.39±2.65	29.68±2.46	34.03±3.27	11.50	12.50
HGT	60.57±3.19	66.21±1.18	23.77±2.85	57.05±2.02	36.91±3.70	43.35±4.07	24.33±3.57	32.99±4.58	7.75	7.75
Simple-HGN	64.08±0.51	67.66±0.18	46.01±1.88	**66.11±0.21**	14.71±3.05	26.01±1.98	28.96±5.21	36.37±2.99	6.75	6.50
HINormer	60.58±1.06	63.72±0.44	46.15±2.57	65.06±0.56	26.70±6.81	35.42±4.51	19.94±2.52	38.81±1.11	8.75	6.75
SlotGAT	63.63±0.82	**68.39±0.41**	44.46±1.64	65.92±0.66	36.51±1.20	43.23±1.19	31.08±0.75	36.67±2.58	3.75	**3.25**
MHGCN	62.77±0.32	65.83±1.36	23.71±5.66	45.43±5.34	30.40±1.32	36.70±0.88	27.38±1.37	33.68±3.25	8.00	9.75
BPHGNN	62.42±0.46	65.49±0.91	OOM	OOM	32.68±0.72	38.79±0.70	28.85±5.01	34.08±3.12	5.67	7.67
MRiemGNN	61.26±0.72	64.18±0.60	31.52±1.37	34.38±2.27	29.01±0.91	30.48±0.53	33.81±2.25	34.18±2.53	6.50	12.00
LINC	**64.69±1.17**	67.13±1.09	**52.32±0.90**	65.26±0.52	**44.86±1.38**	**48.07±1.38**	33.85±3.02	36.07±2.83	**1.25**	4.25

0.6 points of the best baseline. On the heterophilic Roman-Empire, Tolokers, and Minesweeper, it improves over the best baseline by 1.90, 1.66, and 1.48 points, although the best baselines GNN-FiLM and EGAT shape each message with both endpoints of an edge. For instance, in Roman-Empire, a syntactic dependency tells a head word and its dependent different things about their roles. On heterogeneous graphs, the two participants often have different types, and LINC improves by 7.95 points on Alibaba without metapaths or relation-specific weight matrices. On directed graphs, the best baseline changes among magnetic-Laplacian models, CoED, and IBG-NN, while LINC learns the roles of the source and the target from each interaction. The ablation in Section 5.3 supports this view, since different consequences bring their four largest Macro-F1 gains on Alibaba, Tolokers, Minesweeper, and Roman-Empire. Finally, the enhanced GCN and GraphSAGE rank first on Chameleon-F and Amazon-Ratings, confirming that simple models are strong baselines (Platonov

Table 3: Node classification results on directed graphs (mean ± standard deviation, %). Top rows give the size and adjusted homophily (hom). We report accuracy. Best and second-best results are in **bold** and underlined.

Method	Cora-ML	Roman-Empire	MGTAB	ogbn-arxiv	Avg. rank
# Nodes	2,995	22,662	10,199	169,343	
# Edges	8,416	39,143	425,863	1,166,243	
hom	0.75	−0.05	−0.04	0.59	
DiGCN-IB	87.27±1.23	43.89±0.66	83.24±1.40	63.53±0.47	6.75
MagNet	87.61±1.10	84.94±0.47	89.57±1.04	68.93±0.15	3.50
LightDiC	**88.16±0.72**	71.83±0.32	85.55±0.97	66.72±0.17	4.50
FaberNet	79.25±1.58	92.35±0.21	73.11±0.93	68.92±0.26	5.25
Dir-GNN	86.48±1.72	91.61±0.72	89.80±1.12	67.67±0.31	4.00
CoED	86.23±1.32	92.58±0.45	83.43±0.94	67.45±0.10	4.75
IBG-NN	75.45±2.23	65.06±0.73	90.43±0.68	64.14±0.30	6.00
LINC	87.75±1.03	**94.68±0.24**	**90.64±0.94**	**71.58±0.19**	**1.25**

Table 4: Ablation study results (mean, %) on one graph of each kind. A check mark means the part is kept. Small gray numbers give the difference from LINC in percentage points, and Avg. drop is the mean drop over the eight columns. ‡Widened to the parameter count of LINC.

			Homophilic Amazon-Photo		Heterophilic Tolokers		Heterogeneous Freebase		Directed Roman-Empire		
Direct term	Interaction representation	Own scales	Acc	Ma-F1	ROC-AUC	Ma-F1	Ma-F1	Mi-F1	Acc	Ma-F1	Avg. drop
			90.41 −6.03	88.49 −7.09	76.79 −9.25	60.32 −11.90	18.64 −33.67	46.31 −18.95	67.73 −26.95	55.57 −38.12	19.00
✓			95.58 −0.87	94.57 −1.01	85.20 −0.83	71.90 −0.32	51.97 −0.34	64.83 −0.43	93.42 −1.26	92.07 −1.62	0.84
✓‡			95.58 −0.87	94.53 −1.05	85.05 −0.98	71.53 −0.69	52.02 −0.30	64.90 −0.36	93.94 −0.75	92.68 −1.01	0.75
	✓		95.36 −1.09	94.19 −1.39	85.13 −0.91	70.42 −1.80	49.05 −3.26	63.49 −1.77	92.89 −1.79	91.47 −2.21	1.78
	✓	✓	95.25 −1.20	94.12 −1.47	85.66 −0.37	72.47 +0.25	47.95 −4.36	63.35 −1.90	93.52 −1.16	92.31 −1.38	1.45
✓	✓		96.27 −0.17	95.50 −0.09	85.89 −0.15	72.30 +0.08	52.44 +0.13	65.70 +0.44	94.49 −0.19	93.46 −0.23	0.02
✓	✓	✓	**96.45**	**95.58**	**86.03**	**72.22**	**52.32**	**65.26**	**94.68**	**93.69**	–

et al., 2023b; Luo et al., 2024). LINC is equally simple and shows that a model built on what an interaction means to each participant performs strongly on every kind of graph.

5.3 Ablation study

To answer RQ2, we ablate the three parts of a consequence in Eq. (3). The *direct term* carries the other participant's features, the *interaction representation* $\boldsymbol{z}_q$ enters through the shared content $\boldsymbol{c}_q$, and the *own scales* adapt $\boldsymbol{c}_q$ to each participant. We remove them alone and in pairs by setting $\boldsymbol{D}_\mathrm{s} = \boldsymbol{D}_\mathrm{t} = \boldsymbol{0}$, $\boldsymbol{B} = \boldsymbol{0}$, or $\boldsymbol{C} = \boldsymbol{0}$, keeping the own scales only with $\boldsymbol{c}_q$. Every variant gives each participant its own consequence and differs only in what the consequence contains. Without the interaction representation, the layer becomes message passing (Proposition 2), which we also widen to match LINC. All variants reuse the configuration of LINC (see Appendix D.2 for all graphs).

One interaction. The interaction representation alone carries most of what a node needs from its neighbors. In Table 4, removing both the direct term and the interaction representation costs 19.00 points on average, and keeping the latter alone, 1.45 points. On Tolokers, it performs on par with LINC and beats the enlarged message-passing variant in both metrics. Since $\boldsymbol{z}_q$ is formed from both participants, it carries the features of the neighbor along with the pair, and it needs enough dimensions to replace the direct term. Across the 16 graphs, removing the direct term costs 1.16 points at the median when $r \geq d$ and 3.50 points when $r < d$, as on Freebase with $r = 128 < d = 256$.

Two consequences. Different consequences help where the two participants differ. Without the direct term and the own scales, both participants take the same consequence (Theorem 1), and the interaction adds the same term to both of their aggregations. A shared term likely suits participants with the same label and conflicts with different labels or roles. Thus, the own scales should matter on heterophilic and directed graphs and little on homophilic graphs. In Table 4, fixing the own scales lowers the results on Tolokers and the directed Roman-Empire in both metrics. Across the 16 graphs, it lowers the results on all four directed graphs and on five of six heterophilic graphs in each metric, and changes them by at most 0.11 points on homophilic graphs. With the direct term in place, the participants already receive different messages, and the own scales cost 0.02 points in Table 4.

Complementary parts. The direct term and the interaction representation carry different information. The direct term depends on the other participant alone, while the interaction term multiplies the features of both participants (Eq. (1)) and changes with the embedding of the receiver (Proposition 1). Without the interaction representation, the results in Table 4 drop on all four graphs in both metrics,

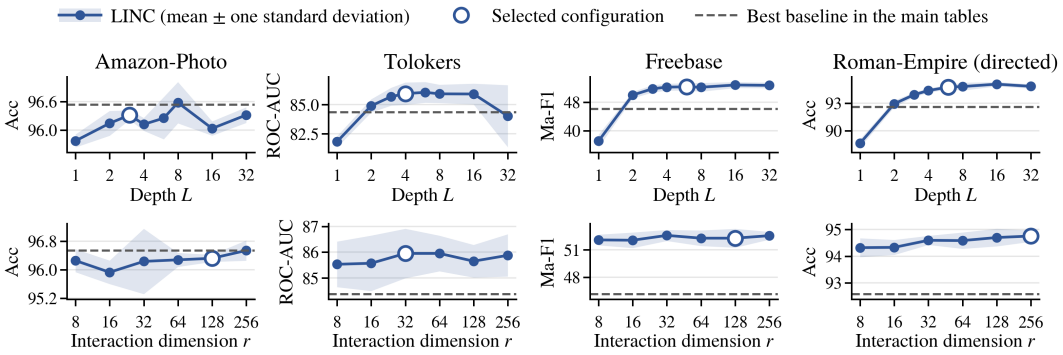


Figure 2: Hyperparameter analysis (%) of the depth L (top) and the interaction dimension r (bottom).

and widening closes the gap on none of them. Keeping both parts gives the best main metric on all four graphs, and the margin over the variants with one part is often larger in Macro-F1. As Macro-F1 weights every class equally, the larger margin suggests the combination helps small classes most.

5.4 Hyperparameter and efficiency analysis

To answer RQ3, we vary the depth L and the interaction dimension r one at a time around the selected configuration. As shown in Figure 2, LINC is robust to r. The main metric varies by less than one point across r on all four graphs and on 12 of 16 graphs in Appendix D.3. The robustness agrees with Section 5.3, since the direct term and the interaction representation share the work of carrying the features of the neighbor. In contrast, depth matters where labels depend on distant nodes, as each layer crosses one interaction. On the directed Roman-Empire, a sparse word graph, the accuracy rises from 88.63 to 95.07 with 16 layers, and on the grid of Minesweeper, the ROC-AUC rises from 88.61 to 98.96 with 32 layers. On Tolokers, with 88 neighbors per node on average, more than four layers add little. We cap the depth at six layers. Six layers are deeper than most GNNs used in practice and keep every configuration affordable on the largest graphs. LINC reaches all main results within this cap, and the deeper models above, which the main tables omit, show that it can improve further.

To answer RQ4, we measure the training time per epoch on the same GPU (see Appendix D.4). LINC ranks 6th of 16 methods on Taobao, 7th of 8 on Cora-ML, and 16th of 19 on the dense Tolokers, since its cost grows with the number of interactions (see Section 4.3).

6 Conclusion

In this paper, we start from the gap between how GNNs are designed and how graph data are formed, and propose LINC. In LINC, the two participants of an interaction share one representation and use it through scales computed from their own embeddings. As a result, one interaction has a consequence for each participant. With one architecture, LINC achieves the best average rank in every group of benchmarks. In ablation studies, the interaction representation alone carries most of what a node needs from its neighbors, and the two participants benefit from different consequences.

Limitations and future work. Following common practice in node classification, we evaluate LINC on graphs whose interactions carry no features or timestamps. However, real interactions often come with both, such as the amount and time of a transaction. The interaction representation is a natural place to add them, and it can also serve link and graph prediction. LINC also uses a basic form of interaction learning and aggregates the consequences of a node with a simple degree-normalized average. Attention over consequences may further improve it. The interaction representation also adds a cost of $O(mrd)$ per layer (see Section 4.3), and sampling can reduce it on large graphs. Our view also extends to other fields, where a follow is attention given by one user and influence gained by the other, and a rating, a transaction, or a binding event likewise means different things to its two participants. Among agents built on large language models (Zhang et al., 2025), one critique means one thing to the agent that gives it and another to the agent that receives it. These settings often need task-specific designs, which we leave for future work.

AI USE STATEMENT

We used generative AI tools for translation, language editing, related work discovery, and code development. The ideas, the model, and the experimental design are the authors' own. The authors take full responsibility for the content, including the authenticity of the AI-assisted code and of every reported result.

ETHICS STATEMENT

This study uses public graph-learning benchmarks and reports aggregate predictive results. We do not attempt to identify individuals represented in social-network data or to make decisions about them, and we make no claim about deployment. Please see Appendix C.1 for the description and source of every dataset.

REPRODUCIBILITY STATEMENT

Section 4 defines the layer, and Appendix B provides all proofs. Appendix C describes the datasets, the baselines, and the settings of every method, and Appendix D provides the settings and additional results of every experiment. To reproduce our results, we upload the anonymized code of LINC as supplementary material, with the configuration of every dataset and the scripts that prepare the data, search the hyperparameters, train and evaluate the models, and run the ablation and sensitivity analyses.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 21–29, 2019.

Amir Ghazizadeh Ahsaei, Rickard Ewetz, and Hao Zheng. GAMMA: Gated multi-hop message passing for homophily-agnostic node representation in GNNs. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Pablo Barceló, Fabian Jogl, Alexander Kozachinskiy, Matthias Lanzinger, Stefan Neumann, and Cristóbal Rojas. Message passing on the edge: Towards scalable and expressive GNNs. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2510.13615.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

Cristian Bodnar, Francesco Di Giovanni, Benjamin P. Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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

Fedor Borisyuk, Shihai He, Yunbo Ouyang, Morteza Ramezani, Peng Du, Xiaochen Hou, Chengming Jiang, Nitin Pasumarthy, Priya Bannur, Birjodh Singh Tiwana, Ping Liu, Siddharth Dangi, Daqi Sun, Zhoutao Pei, Xiao Shi, Sirou Zhu, Qianqi Shen, Kuang-Hsuan Lee, David Stein, Baolei Li, Haichao Wei, Amol Ghoting, and Souvik Ghosh. LiGNN: Graph neural networks at LinkedIn. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4793–4803, 2024.

Marc Brockschmidt. GNN-FiLM: Graph neural networks with feature-wise linear modulation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1144–1152, 2020.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

David Buterez, Jon Paul Janet, Dino Oglic, and Pietro Liò. An end-to-end attention-based approach for learning on graphs. *Nature Communications*, 16:5244, 2025. doi: 10.1038/s41467-025-60252-z.

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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m1oqEOAozQU.

Hao Chen, Yuanchen Bei, Qijie Shen, Yue Xu, Sheng Zhou, Wenbing Huang, Feiran Huang, Senzhang Wang, and Xiao Huang. Macro graph neural networks for online billion-scale recommender systems. In *Proceedings of the ACM Web Conference 2024*, pp. 3598–3608, 2024. doi: 10.1145/3589334.3645517.

Jialong Chen, Bowen Deng, Zhen Wang, Chuan Chen, and Zibin Zheng. Graph neural Ricci flow: Evolving feature from a curvature perspective. In *International Conference on Learning Representations*, 2025.

Jun Chen and Haopeng Chen. Edge-featured graph attention network. *arXiv preprint arXiv:2101.07671*, 2021.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. In *International Conference on Learning Representations*, 2021.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *International Conference on Learning Representations*, 2024.

Béni Egressy, Luc von Niederhäusern, Jovan Blanuša, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. Provably powerful graph neural networks for directed multigraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11838–11846, 2024. doi: 10.1609/aaai.v38i10.29069.

Chaofan Fu, Guanjie Zheng, Chao Huang, Yanwei Yu, and Junyu Dong. Multiplex heterogeneous graph neural network with behavior pattern modeling. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 482–494, 2023.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, 2017.

Ming Gu, Zhuonan Zheng, Sheng Zhou, Meihan Liu, Jiawei Chen, Qiaoyu Tan, Liangcheng Li, and Jiajun Bu. Making classic GNNs strong baselines across varying homophily: A smoothness–generalization perspective. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yutong Guo, Wenrui Guan, Qihang Guo, Yuge Wang, Keyu Liu, and Xibei Yang. Extending graph neural networks to edge-level neighborhood awareness via line graph. *Electronic Research Archive*, 34(2):866–889, 2026. doi: 10.3934/era.2026040.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Zhichao Han, David S. Kammer, and Olga Fink. Learning physics-consistent particle interactions. *PNAS Nexus*, 1(5):pgac264, 2022. doi: 10.1093/pnasnexus/pgac264.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, 2020.

Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4132–4139, 2020.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking LLMs as semantic encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, San Diego, California, United States, 2026. Association for Computational Linguistics. URL https://aclanthology.org/2026.acl-long.147/.

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 *Advances in Neural Information Processing Systems*, volume 33, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pp. 2704–2710, 2020b. doi: 10.1145/3366423.3380027.

Kexin Huang, Payal Chandak, Qianwen Wang, Shreyas Havaldar, Akhil Vaid, Jure Leskovec, Girish N. Nadkarni, Benjamin S. Glicksberg, Nils Gehlenborg, and Marinka Zitnik. A foundation model for clinician-centered drug repurposing. *Nature Medicine*, 30(12):3601–3613, 2024. doi: 10.1038/s41591-024-03233-x.

Xiaodong Jiang, Pengsheng Ji, and Sheng Li. CensNet: Convolution with edge-node switching in graph neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 2656–2662, 2019. doi: 10.24963/ijcai.2019/369.

Peter Bjørn Jørgensen, Karsten Wedel Jacobsen, and Mikkel N. Schmidt. Neural message passing with edge updates for predicting properties of molecules and materials. *arXiv preprint arXiv:1806.03146*, 2018.

Wagner A. Junior, Fábio Ramos, Alex B. Vieira, José Augusto M. Nacif, and Ricardo S. Ferreira. A survey on edge-aware graph learning methods. *IEEE Transactions on Network Science and Engineering*, 13:5680–5697, 2026. doi: 10.1109/TNSE.2025.3649386.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016. doi: 10.1007/s10822-016-9938-8.

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

Christian Koke and Daniel Cremers. HoloNets: Spectral convolutions do extend to directed graphs. In *International Conference on Learning Representations*, 2024.

Jonathan Kouchly, Ben Finkelshtein, Michael Bronstein, and Ron Levie. Efficient learning on large graphs using a densifying regularity lemma. In *International Conference on Learning Representations*, 2026.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. JODIE: Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1269–1278, 2019. doi: 10.1145/3292500.3330895.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13242–13256, 2022.

Xiang Li, Yuan Cao, Zhongying Zhao, Guoqing Chao, and Yanwei Yu. Multiplex heterogeneous graph neural networks with Euclidean-Riemannian mutual space synergy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 2026. doi: 10.1609/aaai.v40i18.38536.

Xunkai Li, Meihao Liao, Zhengyu Wu, Daohan Su, Wentao Zhang, Rong-Hua Li, and Guoren Wang. LightDiC: A simple yet effective approach for large-scale digraph representation learning. *Proceedings of the VLDB Endowment*, 17(7):1542–1551, 2024.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are strong baselines: Reassessing GNNs for node classification. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, 2024.

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2021.

Yanni Ma, Hao Liu, Yulan Guo, Theo Gevers, and Martin R. Oswald. Edge-centric relational reasoning for 3D scene graph prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 7847–7855, 2026. doi: 10.1609/aaai.v40i10.37728.

Qiheng Mao, Zemin Liu, Chenghao Liu, and Jianling Sun. HINormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM Web Conference 2023*, pp. 599–610, 2023. doi: 10.1145/3543507.3583493.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design. *arXiv preprint arXiv:2105.07634*, 2021.

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

Nimrah Mustafa and Rebekka Burkholz. GATE: How to keep out intrusive neighbors. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Seong Ho Pahng and Sahand Hormoz. Improving graph neural networks by learning continuous edge directions. In *International Conference on Learning Representations*, 2025.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily–heterophily dichotomy and beyond. In *Advances in Neural Information Processing Systems*, volume 36, pp. 523–548, 2023a.

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

Ruizhong Qiu, Ting-Wei Li, Gaotang Li, and Hanghang Tong. Graph homophily booster: Reimagining the role of discrete features in heterophilic graph learning. In *International Conference on Learning Representations*, 2026.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515, 2022.

Dhruv Rohatgi, Tanya Marwah, Zachary Chase Lipton, Jianfeng Lu, Ankur Moitra, and Andrej Risteski. Towards characterizing the value of edge embeddings in graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 51905–51923, 2025.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M. Bronstein. Edge directionality improves learning on heterophilic graphs. In *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pp. 25:1–25:27, 2024.

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

Celia Rubio-Madrigal and Rebekka Burkholz. Fixed aggregation features can rival GNNs. In *International Conference on Machine Learning*, 2026.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pp. 593–607, 2018. doi: 10.1007/978-3-319-93417-4_38.

Vinay Sharma and Olga Fink. A physics-informed graph neural network conserving linear and angular momentum for dynamical systems. *Nature Communications*, 17:1045, 2026. doi: 10.1038/s41467-025-67802-5.

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

Shuhao Shi, Kai Qiao, Jian Chen, Shuai Yang, Jie Yang, Baojie Song, Linyuan Wang, and Bin Yan. MGTAB: A multi-relational graph-based twitter account detection benchmark. *arXiv preprint arXiv:2301.01123*, 2023.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31613–31632, 2023.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3693–3702, 2017. doi: 10.1109/CVPR.2017.11.

Ying Song, Shuangjia Zheng, Zhangming Niu, Zhang-Hua Fu, Yutong Lu, and Yuedong Yang. Communicative representation learning on attributed molecular graphs. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 2831–2838, 2020. doi: 10.24963/ijcai.2020/392.

Benedict Aaron Tjandra, Federico Barbero, and Michael M. Bronstein. Enhancing the expressivity of temporal graph networks through source-target identification. *arXiv preprint arXiv:2411.03596*, 2024.

Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David S. Rosenblum, and Andrew Lim. Digraph inception convolutional networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 165–174, 2019a.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pp. 2022–2032, 2019b.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sNFLN3itAd.

Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. ACMP: Allen-Cahn message passing with attractive and repulsive forces for graph neural networks. In *International Conference on Learning Representations*, 2023.

Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. NodeFormer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27387–27401, 2022.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. Less is more: on the over-globalizing problem in graph transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54656–54672, 2024.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Hansheng Xue, Luwei Yang, Vaibhav Rajan, Wen Jiang, Yi Wei, and Yu Lin. Multiplex bipartite network embedding using dual hypergraph convolutional networks. In *Proceedings of the Web Conference 2021*, pp. 1649–1660, 2021.

Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10816–10824, 2023.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, pp. 28877–28888, 2021.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In *Advances in Neural Information Processing Systems*, volume 36, pp. 67686–67700, 2023.

Pengyang Yu, Chaofan Fu, Yanwei Yu, Chao Huang, Zhongying Zhao, and Junyu Dong. Multiplex heterogeneous graph convolutional network. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2377–2387, 2022.

Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 793–803, 2019.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-Designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76678–76692, 2025.

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. In *Advances in Neural Information Processing Systems*, volume 34, 2021a.

Qiannan Zhang and Shichao Pei. Edge-centric heterogeneous graph transformer with vector-quantized memory. In *ECAI 2025*, pp. 2850–2857, 2025. doi: 10.3233/FAIA251142.

Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. MagNet: A neural network for directed graphs. In *Advances in Neural Information Processing Systems*, volume 34, 2021b.

Zhen Zhang, Mohammed Haroon Dupty, Fan Wu, Javen Qinfeng Shi, and Wee Sun Lee. Factor graph neural networks. *Journal of Machine Learning Research*, 24(181):1–54, 2023.

Ziang Zhou, Jieming Shi, Renchi Yang, Yuanhang Zou, and Qing Li. SlotGAT: Slot-based message passing for heterogeneous graphs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 42644–42657, 2023.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, and Bin Wang. Relation structure-aware heterogeneous graph neural network. In *2019 IEEE International Conference on Data Mining*, pp. 1534–1539, 2019.

Aaron Zweig and Joan Bruna. Exponential separations in symmetric neural networks. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

A Extended Related Work

A.1 Computational roles of an edge

Routes. Most GNNs for node classification are built from message-passing layers (Gilmer et al., 2017). GCN uses a fixed normalized average over the neighborhood (Kipf & Welling, 2017), GraphSAGE concatenates the embedding of a node with an aggregate of its sampled neighbors (Hamilton et al., 2017), and GIN studies which aggregators are injective on multisets (Xu et al., 2019). On heterophilic graphs, H2GCN keeps the embedding of a node and its neighbors separate and uses higher-order neighborhoods (Zhu et al., 2020), GPR-GNN learns signed weights over propagation steps (Chien et al., 2021), and LINKX encodes node features and adjacency separately

before combining them (Lim et al., 2021). ACM-GCN mixes low-pass, high-pass, and identity channels (Luan et al., 2022), IGNN treats neighborhoods of different hops separately to ease the dilemma between smoothness and generalization (Gu et al., 2025), and GRAPHITE adds feature nodes to increase homophily (Qiu et al., 2026). Benchmark studies show that simply keeping the embedding of a node and its neighbors separate is already strong on heterophilic graphs (Platonov et al., 2023b). In multi-agent systems built on large language models, the graph decides which agents exchange messages, and G-Designer uses a GNN to design this topology for each task (Zhang et al., 2025).

Weights. GAT and GATv2 compute a positive weight for each neighbor from both endpoints (Veličković et al., 2018; Brody et al., 2022), and FAGCN allows negative weights (Bo et al., 2021). Graph transformers extend attention to all nodes and keep the graph as local message-passing layers or as a bias on the attention weights (Ying et al., 2021; Rampášek et al., 2022; Wu et al., 2022; 2023; Shirzad et al., 2023; Deng et al., 2024).

Conditions. Edge-Conditioned Convolution generates the transformation of a message from edge attributes (Simonovsky & Komodakis, 2017). R-GCN selects a weight matrix by relation (Schlichtkrull et al., 2018), and CompGCN composes node and relation embeddings before the transformation (Vashishth et al., 2020). HGT conditions attention and transformation on node types and relations (Hu et al., 2020b), Simple-HGN adds learnable relation embeddings to graph attention (Lv et al., 2021), and HINormer and SlotGAT encode node types in a graph transformer and in type-specific slots (Mao et al., 2023; Zhou et al., 2023). Spectral models encode direction in a complex or non-symmetric operator (Zhang et al., 2021b; Koke & Cremers, 2024), and Dir-GNN aggregates incoming and outgoing neighbors separately (Rossi et al., 2024). Interaction Networks compute an effect from the sender, the receiver, and the edge attributes and aggregate the effects at the receiver to predict the dynamics of physical systems (Battaglia et al., 2016). GNN-FiLM computes a feature-wise scale and shift from the receiving node and applies them to each incoming message (Brockschmidt, 2020). Neural sheaf diffusion attaches learned maps to the endpoints of each edge and diffuses node features through the resulting sheaf Laplacian (Bodnar et al., 2022).

States. Weave networks update atom and pair states together (Kearnes et al., 2016). Graph Networks extend Interaction Networks with edge, node, and global updates (Battaglia et al., 2018), and several molecular models keep directed edge states across layers or couple node and edge updates (Jørgensen et al., 2018; Song et al., 2020). CensNet alternates convolutions on the graph and on its line graph (Jiang et al., 2019), and ELNA-Net propagates approximate edge states over a line graph before fusing them into the nodes (Guo et al., 2026). EGAT updates node embeddings and edge states jointly through attention (Chen & Chen, 2021). Factor graph neural networks introduce factor nodes connected to two or more variables and learn to perform inference in probabilistic graphical models (Zhang et al., 2023). Recent work studies when edge states add representational power and when edges should be the main tokens of the computation (Rohatgi et al., 2025; Buterez et al., 2025; Barceló et al., 2026). EHFormer represents a heterogeneous neighborhood as a sequence of edges (Zhang & Pei, 2025), and LEO reasons over object pairs on a line graph and writes the result back to the objects (Ma et al., 2026). A recent survey covers many more edge-aware methods (Junior et al., 2026).

A.2 Joint treatments of the pair

Updating both endpoints. Some models compute separate updates for the two endpoints of an edge. Physics-informed interaction models decode a pair of forces or torques from one edge and tie the two through conservation laws (Han et al., 2022; Sharma & Fink, 2026). Temporal interaction models process a sequence of timestamped events and update both participants of each event. JODIE couples two recurrent networks for this (Kumar et al., 2019), DyRep models how events drive node representations (Trivedi et al., 2019), and TGN builds one message for the source and one for the target of each event and writes them into node memories (Rossi et al., 2020). TGNv2 shows that the TGN message cannot tell the source from the target and adds node identifiers to solve the problem (Tjandra et al., 2024).

Scoring the pair. Other models give the pair a single output, usually a score. The product of the two node embeddings is the standard decoder in link prediction and knowledge graph completion. Work on the labeling trick shows that node representations computed independently of each other cannot

distinguish some pairs of nodes, and link prediction methods therefore label the target pair before message passing (Zhang et al., 2021a). Later link predictors add structural features of the target pair, such as subgraph sketches (Chamberlain et al., 2023) or learned representations of common neighbors (Wang et al., 2024). On temporal graphs, GraphMixer encodes the recent links of a node with a fixed time encoding and an MLP mixer and pools the features of its neighbors (Cong et al., 2023). DyGFormer feeds the interaction histories of both endpoints to a Transformer with a neighbor co-occurrence encoding (Yu et al., 2023). Both predict future links from these encodings. In collaborative filtering, NGCF adds the element-wise product of user and item embeddings to its messages to pass more between a user and an item whose embeddings agree (Wang et al., 2019a). The product enters the message to each receiver, whereas LINC forms one representation per interaction and lets each participant read it through its own scale. LightGCN keeps only the normalized neighborhood aggregation for identifier embeddings (He et al., 2020).

B PROOFS

Properties used in the proofs. The proofs below refer to the layer in Section 4 and use a few properties of the GELU and of norm. The GELU acts entrywise as $\mathrm{GELU}(t) = t\,\Phi(t)$, where Φ is the cumulative distribution function of the standard normal distribution and $\varphi = \Phi'$ is its density. Since $\Phi(-t) = 1 - \Phi(t)$,

$$\mathrm{GELU}(t) - \mathrm{GELU}(-t) = t, \qquad \mathrm{GELU}(t) + \mathrm{GELU}(-t) = t\,\big(2\Phi(t) - 1\big), \tag{6}$$

and the second function is even and strictly increasing in $|t|$, because its derivative $2\Phi(t) - 1 + 2t\,\varphi(t)$ is positive for $t > 0$. The GELU is strictly increasing on $[0, \infty)$, where its derivative $\Phi(t) + t\,\varphi(t)$ is positive. Since norm divides a vector by a positive number, it keeps the sign of every entry and is odd, $\mathrm{norm}(-\boldsymbol{a}) = -\,\mathrm{norm}(\boldsymbol{a})$.

B.1 PROOF OF THEOREM 1

Theorem 1 (Exchange symmetry and two consequences)**.** *Let $q = \{u, v\}$ be an unordered interaction. For every parameter setting, exchanging $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ leaves $\boldsymbol{z}_q$ unchanged and exchanges $\boldsymbol{m}_{u|q}$ and $\boldsymbol{m}_{v|q}$. The interaction terms $\boldsymbol{\gamma}_u \odot \boldsymbol{c}_q$ and $\boldsymbol{\gamma}_v \odot \boldsymbol{c}_q$ are equal for all inputs when $\boldsymbol{C} = \boldsymbol{0}$ and can differ when $\boldsymbol{C} \neq \boldsymbol{0}$.*

Proof. Throughout the proof, $q = \{u, v\}$ is unordered. Hence $\boldsymbol{P}_{\mathrm{s}} = \boldsymbol{P}_{\mathrm{t}} = \boldsymbol{P}$ and $\boldsymbol{D}_{\mathrm{s}} = \boldsymbol{D}_{\mathrm{t}} = \boldsymbol{D}$, and the matrix $\boldsymbol{C}$ is shared by all nodes. To make the dependence on the two embeddings explicit, we write

$$\begin{aligned} \boldsymbol{z}(\boldsymbol{x}_u, \boldsymbol{x}_v) &= \sqrt{r}\ \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u) \odot \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v), \\ \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v) &= \boldsymbol{B}\,\boldsymbol{z}(\boldsymbol{x}_u, \boldsymbol{x}_v), \qquad \boldsymbol{\gamma}(\boldsymbol{x}) = \boldsymbol{1} + \frac{\boldsymbol{C}\boldsymbol{x}}{\sqrt{d}}, \end{aligned} \tag{7}$$

Then $\boldsymbol{z}_q = \boldsymbol{z}(\boldsymbol{x}_u, \boldsymbol{x}_v)$, $\boldsymbol{c}_q = \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v)$, and $\boldsymbol{\gamma}_i = \boldsymbol{\gamma}(\boldsymbol{x}_i)$. By Eq. (3),

$$\boldsymbol{m}_{u|q} = \mathrm{GELU}\big(\boldsymbol{D}\boldsymbol{x}_v + \boldsymbol{\gamma}(\boldsymbol{x}_u) \odot \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v)\big), \quad \boldsymbol{m}_{v|q} = \mathrm{GELU}\big(\boldsymbol{D}\boldsymbol{x}_u + \boldsymbol{\gamma}(\boldsymbol{x}_v) \odot \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v)\big). \tag{8}$$

Step 1: the interaction representation is invariant. For every entry $j \in \{1, \dots, r\}$,

$$\begin{aligned} [\boldsymbol{z}(\boldsymbol{x}_v, \boldsymbol{x}_u)]_j &= \sqrt{r}\,[\mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v)]_j\,[\mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u)]_j \\ &= \sqrt{r}\,[\mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u)]_j\,[\mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v)]_j = [\boldsymbol{z}(\boldsymbol{x}_u, \boldsymbol{x}_v)]_j, \end{aligned} \tag{9}$$

because the product of two real numbers does not depend on their order. Hence $\boldsymbol{z}(\boldsymbol{x}_v, \boldsymbol{x}_u) = \boldsymbol{z}(\boldsymbol{x}_u, \boldsymbol{x}_v)$, and multiplying both sides by $\boldsymbol{B}$ gives $\boldsymbol{c}(\boldsymbol{x}_v, \boldsymbol{x}_u) = \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v)$. This step uses the tied matrix $\boldsymbol{P}$. With $\boldsymbol{P}_{\mathrm{s}} \neq \boldsymbol{P}_{\mathrm{t}}$, the two factors would be projected by different matrices and could not be exchanged.

Step 2: the two consequences are exchanged. Exchanging the embeddings means that the layer receives $\boldsymbol{x}_v$ as the embedding of u and $\boldsymbol{x}_u$ as the embedding of v. Denote the resulting consequences

by $\boldsymbol{m}'_{u|q}$ and $\boldsymbol{m}'_{v|q}$. Substituting the exchanged embeddings into Eq. (8) and applying Step 1,

$$\begin{aligned}\boldsymbol{m}'_{u|q} &= \mathrm{GELU}\big(\boldsymbol{D}\boldsymbol{x}_u + \boldsymbol{\gamma}(\boldsymbol{x}_v) \odot \boldsymbol{c}(\boldsymbol{x}_v, \boldsymbol{x}_u)\big) \\ &= \mathrm{GELU}\big(\boldsymbol{D}\boldsymbol{x}_u + \boldsymbol{\gamma}(\boldsymbol{x}_v) \odot \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v)\big) = \boldsymbol{m}_{v|q}, \qquad (10)\\ \boldsymbol{m}'_{v|q} &= \mathrm{GELU}\big(\boldsymbol{D}\boldsymbol{x}_v + \boldsymbol{\gamma}(\boldsymbol{x}_u) \odot \boldsymbol{c}(\boldsymbol{x}_v, \boldsymbol{x}_u)\big) \\ &= \mathrm{GELU}\big(\boldsymbol{D}\boldsymbol{x}_v + \boldsymbol{\gamma}(\boldsymbol{x}_u) \odot \boldsymbol{c}(\boldsymbol{x}_u, \boldsymbol{x}_v)\big) = \boldsymbol{m}_{u|q}. \qquad (11)\end{aligned}$$

This step relies on $\boldsymbol{D}$ and $\boldsymbol{C}$ being the same for both participants. Storing the interaction as (v, u) instead of (u, v) has the same effect. The layer then computes $\boldsymbol{c}(\boldsymbol{x}_v, \boldsymbol{x}_u) = \boldsymbol{c}_q$, node v takes the role of the source and receives $\mathrm{GELU}(\boldsymbol{D}_\mathrm{s}\boldsymbol{x}_u + \boldsymbol{\gamma}_v \odot \boldsymbol{c}_q) = \boldsymbol{m}_{v|q}$, and node u takes the role of the target and receives $\mathrm{GELU}(\boldsymbol{D}_\mathrm{t}\boldsymbol{x}_v + \boldsymbol{\gamma}_u \odot \boldsymbol{c}_q) = \boldsymbol{m}_{u|q}$. The order in which an unordered interaction is stored thus changes neither $\boldsymbol{z}_q$ nor the consequence for either participant.

Step 3: the interaction terms differ only through $\boldsymbol{C}$. By Eq. (7), the scales of the two participants differ by $\boldsymbol{\gamma}(\boldsymbol{x}_u) - \boldsymbol{\gamma}(\boldsymbol{x}_v) = \boldsymbol{C}(\boldsymbol{x}_u - \boldsymbol{x}_v)/\sqrt{d}$, and both multiply the same $\boldsymbol{c}_q$. It follows that

$$\boldsymbol{\gamma}_u \odot \boldsymbol{c}_q - \boldsymbol{\gamma}_v \odot \boldsymbol{c}_q = \frac{1}{\sqrt{d}}\big(\boldsymbol{C}(\boldsymbol{x}_u - \boldsymbol{x}_v)\big) \odot \boldsymbol{c}_q. \qquad (12)$$

If $\boldsymbol{C} = \boldsymbol{0}$, the right-hand side vanishes for all inputs, and the two interaction terms are equal. Otherwise, entry k of the difference is nonzero exactly when $[\boldsymbol{C}(\boldsymbol{x}_u - \boldsymbol{x}_v)]_k \neq 0$ and $[\boldsymbol{c}_q]_k \neq 0$, that is, when the two participants have different scales on a feature that the shared content uses.

Step 4: the interaction terms can differ when $\boldsymbol{C} \neq \boldsymbol{0}$. We construct an example. Let $\boldsymbol{e}_1 \in \mathbb{R}^r$ and $\boldsymbol{e}_k \in \mathbb{R}^d$ be standard basis vectors, $\boldsymbol{p}$ a nonzero vector, and $\boldsymbol{P}$ a matrix with first row $\boldsymbol{p}$. Take

$$\boldsymbol{x}_u = \boldsymbol{p}, \qquad \boldsymbol{x}_v = 2\boldsymbol{p}, \qquad \boldsymbol{B} = \boldsymbol{w}\boldsymbol{e}_1^\top, \qquad \boldsymbol{C} = \alpha\, \boldsymbol{e}_k(\boldsymbol{x}_u - \boldsymbol{x}_v)^\top, \qquad (13)$$

where $\boldsymbol{w} \in \mathbb{R}^d$ has $w_k > 0$ and $\alpha > 0$. Since $\boldsymbol{p}^\top\boldsymbol{x}_u > 0$, $\boldsymbol{p}^\top\boldsymbol{x}_v > 0$, and norm keeps signs, the first entry of $\boldsymbol{z}_q$ is positive. It follows that

$$\boldsymbol{c}_q = [\boldsymbol{z}_q]_1\, \boldsymbol{w}, \qquad [\boldsymbol{c}_q]_k > 0, \qquad [\boldsymbol{C}(\boldsymbol{x}_u - \boldsymbol{x}_v)]_k = \alpha\|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2^2 > 0, \qquad (14)$$

and Eq. (12) gives

$$[\boldsymbol{\gamma}_u \odot \boldsymbol{c}_q - \boldsymbol{\gamma}_v \odot \boldsymbol{c}_q]_k = \frac{\alpha}{\sqrt{d}}\,\|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2^2\,[\boldsymbol{c}_q]_k > 0. \qquad (15)$$

The consequences differ as well. The k-th entries of the two scales are

$$[\boldsymbol{\gamma}_u]_k = 1 + \frac{\alpha}{\sqrt{d}}\,(\boldsymbol{x}_u - \boldsymbol{x}_v)^\top\boldsymbol{x}_u, \qquad [\boldsymbol{\gamma}_v]_k = 1 + \frac{\alpha}{\sqrt{d}}\,(\boldsymbol{x}_u - \boldsymbol{x}_v)^\top\boldsymbol{x}_v, \qquad (16)$$

and both are positive for α small enough. The k-th entries of the two interaction terms are therefore two different positive numbers. With $\boldsymbol{D} = \boldsymbol{0}$, the two numbers pass through the GELU, which is strictly increasing on $[0, \infty)$, and hence $[\boldsymbol{m}_{u|q}]_k \neq [\boldsymbol{m}_{v|q}]_k$. Since the interaction is unordered and every parameter is shared by the two participants, the difference comes from their embeddings alone. □

B.2 Proof of Proposition 1

Proposition 1 (Dependence on both participants)**.** *Fix the embedding $\boldsymbol{x}_v$ of the neighbor. The message $\mathrm{GELU}(\boldsymbol{D}_\mathrm{s}\boldsymbol{x}_v)$ of the direct term is the same for every embedding $\boldsymbol{x}_u$ of u, while the consequence $\boldsymbol{m}_{u|q}$ can change with $\boldsymbol{x}_u$.*

Proof. The message $\mathrm{GELU}(\boldsymbol{D}_\mathrm{s}\boldsymbol{x}_v)$ applies the linear map $\boldsymbol{D}_\mathrm{s}$ and then the entrywise GELU to $\boldsymbol{x}_v$. It does not involve $\boldsymbol{x}_u$ and therefore takes the same value for every embedding $\boldsymbol{x}_u$.

For the second claim, we show that replacing $\boldsymbol{x}_u$ by $-\boldsymbol{x}_u$ can change the consequence while $\boldsymbol{x}_v$ stays fixed. Take $\boldsymbol{C} = \boldsymbol{0}$ and $\boldsymbol{D}_\mathrm{s} = \boldsymbol{0}$, which gives $\boldsymbol{\gamma}_u = \boldsymbol{1}$ and $\boldsymbol{m}_{u|q} = \mathrm{GELU}(\boldsymbol{c}_q)$, and take any $\boldsymbol{P}_\mathrm{s}$, $\boldsymbol{P}_\mathrm{t}$, $\boldsymbol{B}$, $\boldsymbol{x}_u$, and $\boldsymbol{x}_v$ with $\boldsymbol{c}_q \neq \boldsymbol{0}$, such as the choice in Step 4 of the proof of Theorem 1. Since norm is odd, replacing $\boldsymbol{x}_u$ by $-\boldsymbol{x}_u$ turns $\boldsymbol{z}_q$ into $-\boldsymbol{z}_q$ and $\boldsymbol{c}_q = \boldsymbol{B}\boldsymbol{z}_q$ into $-\boldsymbol{c}_q$. By Eq. (6), the two consequences then differ by

$$\mathrm{GELU}(\boldsymbol{c}_q) - \mathrm{GELU}(-\boldsymbol{c}_q) = \boldsymbol{c}_q \neq \boldsymbol{0}, \qquad (17)$$

whereas the message of the direct term is $\mathrm{GELU}(\boldsymbol{D}_\mathrm{s}\boldsymbol{x}_v)$ in both cases. □

B.3 Proof of Proposition 2

Proposition 2 (Reduction to message passing)**.** *With $\boldsymbol{B} = \mathbf{0}$, the layer equals the message-passing layer with messages $\boldsymbol{m}^{(0)}_{u|q} = \mathrm{GELU}(\boldsymbol{D}_{\mathrm{s}}\boldsymbol{x}_v)$ and $\boldsymbol{m}^{(0)}_{v|q} = \mathrm{GELU}(\boldsymbol{D}_{\mathrm{t}}\boldsymbol{x}_u)$ and with the same self term, aggregation, and residual block.*

Proof. Let $\boldsymbol{B} = \mathbf{0}$. For every interaction q, Eq. (2) gives $\boldsymbol{c}_q = \boldsymbol{B}\boldsymbol{z}_q = \mathbf{0}$ whatever the value of $\boldsymbol{z}_q$, and hence $\boldsymbol{\gamma}_u \odot \boldsymbol{c}_q = \boldsymbol{\gamma}_v \odot \boldsymbol{c}_q = \mathbf{0}$ whatever the values of $\boldsymbol{\gamma}_u$ and $\boldsymbol{\gamma}_v$. Substituting into Eq. (3),

$$\boldsymbol{m}_{u|q} = \mathrm{GELU}(\boldsymbol{D}_{\mathrm{s}}\boldsymbol{x}_v + \mathbf{0}) = \boldsymbol{m}^{(0)}_{u|q}, \qquad \boldsymbol{m}_{v|q} = \mathrm{GELU}(\boldsymbol{D}_{\mathrm{t}}\boldsymbol{x}_u + \mathbf{0}) = \boldsymbol{m}^{(0)}_{v|q}. \tag{18}$$

The aggregation in Eq. (4) then reads

$$\boldsymbol{g}_i = \frac{\boldsymbol{D}_0\boldsymbol{x}_i + \sum_{q\in\mathcal{Q}(i)} \omega_{i,q}\, \boldsymbol{m}^{(0)}_{i|q}}{1 + n_i} + \boldsymbol{b}, \tag{19}$$

which is the aggregation of the message-passing layer with messages $\boldsymbol{m}^{(0)}$, self term $\boldsymbol{D}_0\boldsymbol{x}_i$, weights $\omega_{i,q}$, and bias $\boldsymbol{b}$. The residual block in Eq. (5) acts on $\boldsymbol{g}_i$ and $\boldsymbol{h}^{\ell}_i$ only and does not involve $\boldsymbol{P}_{\mathrm{s}}$, $\boldsymbol{P}_{\mathrm{t}}$, $\boldsymbol{B}$, or $\boldsymbol{C}$. Hence the layer of LINC with $\boldsymbol{B} = \mathbf{0}$ and the message-passing layer give the same output for every input and every value of the remaining parameters, and with dropout the two layers coincide for every realization of the dropout masks. The matrices $\boldsymbol{P}_{\mathrm{s}}$, $\boldsymbol{P}_{\mathrm{t}}$, and $\boldsymbol{C}$ have no effect once $\boldsymbol{B} = \mathbf{0}$, since the three matrices enter the layer only through $\boldsymbol{c}_q$ and $\boldsymbol{\gamma}_i \odot \boldsymbol{c}_q$. □

B.4 Permutation equivariance

Let π be a permutation of the nodes that maps every node to a node of the same type on heterogeneous graphs. Let $\pi \cdot \boldsymbol{H}$ place the embedding of node i at node $\pi(i)$, and let $\pi \cdot \mathcal{Q}$ replace every interaction (u, v) by $(\pi(u), \pi(v))$ with the same relation and the same roles.

Proposition 3 (Permutation equivariance)**.** *With dropout disabled, a layer f_θ of LINC satisfies*

$$f_\theta(\pi \cdot \boldsymbol{H}, \pi \cdot \mathcal{Q}) = \pi \cdot f_\theta(\boldsymbol{H}, \mathcal{Q}). \tag{20}$$

Proof. Write $\boldsymbol{H}' = \pi \cdot \boldsymbol{H}$ and $\mathcal{Q}' = \pi \cdot \mathcal{Q}$, and mark every quantity computed on $(\boldsymbol{H}', \mathcal{Q}')$ with a prime. By definition, $\boldsymbol{h}'_{\pi(i)} = \boldsymbol{h}^{\ell}_i$ for every node i. Since layer normalization acts on each node with shared parameters, $\boldsymbol{x}'_{\pi(i)} = \mathrm{LN}(\boldsymbol{h}'_{\pi(i)}) = \mathrm{LN}(\boldsymbol{h}^{\ell}_i) = \boldsymbol{x}_i$.

Interactions. Every interaction $q = (u, v) \in \mathcal{Q}$ corresponds to $q' = (\pi(u), \pi(v)) \in \mathcal{Q}'$ with the same relation and roles, and hence with the same matrices $\boldsymbol{P}_{\mathrm{s}}$, $\boldsymbol{P}_{\mathrm{t}}$, $\boldsymbol{D}_{\mathrm{s}}$, and $\boldsymbol{D}_{\mathrm{t}}$. By Eq. (1),

$$\boldsymbol{z}_{q'} = \sqrt{r}\ \mathrm{norm}(\boldsymbol{P}_{\mathrm{s}}\boldsymbol{x}'_{\pi(u)}) \odot \mathrm{norm}(\boldsymbol{P}_{\mathrm{t}}\boldsymbol{x}'_{\pi(v)}) = \sqrt{r}\ \mathrm{norm}(\boldsymbol{P}_{\mathrm{s}}\boldsymbol{x}_u) \odot \mathrm{norm}(\boldsymbol{P}_{\mathrm{t}}\boldsymbol{x}_v) = \boldsymbol{z}_q, \tag{21}$$

It follows that $\boldsymbol{c}_{q'} = \boldsymbol{c}_q$ and $\boldsymbol{\gamma}'_{\pi(i)} = \mathbf{1} + \boldsymbol{C}\boldsymbol{x}'_{\pi(i)}/\sqrt{d} = \boldsymbol{\gamma}_i$ for every node. By Eq. (3),

$$\begin{aligned} \boldsymbol{m}'_{\pi(u)|q'} &= \mathrm{GELU}\big(\boldsymbol{D}_{\mathrm{s}}\boldsymbol{x}'_{\pi(v)} + \boldsymbol{\gamma}'_{\pi(u)} \odot \boldsymbol{c}_{q'}\big) = \boldsymbol{m}_{u|q}, \\ \boldsymbol{m}'_{\pi(v)|q'} &= \mathrm{GELU}\big(\boldsymbol{D}_{\mathrm{t}}\boldsymbol{x}'_{\pi(u)} + \boldsymbol{\gamma}'_{\pi(v)} \odot \boldsymbol{c}_{q'}\big) = \boldsymbol{m}_{v|q}. \end{aligned} \tag{22}$$

Aggregation. The interactions of $\pi(i)$ in $\mathcal{Q}'$ are the images of the interactions of i in $\mathcal{Q}$, with the same relations. Therefore $n'_{\pi(i)} = n_i$, $K'_{\pi(i)} = K_i$, and $n'_{\pi(i),\phi} = n_{i,\phi}$ for every relation ϕ, which gives $\omega'_{\pi(i),q'} = \omega_{i,q}$. By Eq. (4),

$$\begin{aligned} \boldsymbol{g}'_{\pi(i)} &= \frac{\boldsymbol{D}_0\boldsymbol{x}'_{\pi(i)} + \sum_{q'\in\mathcal{Q}'(\pi(i))} \omega'_{\pi(i),q'}\, \boldsymbol{m}'_{\pi(i)|q'}}{1 + n'_{\pi(i)}} + \boldsymbol{b} \\ &= \frac{\boldsymbol{D}_0\boldsymbol{x}_i + \sum_{q\in\mathcal{Q}(i)} \omega_{i,q}\, \boldsymbol{m}_{i|q}}{1 + n_i} + \boldsymbol{b} = \boldsymbol{g}_i. \end{aligned} \tag{23}$$

Update. With dropout disabled, the embedding of $\pi(i)$ at the next layer is $\boldsymbol{h}'_{\pi(i)} + \boldsymbol{W}_o\,\mathrm{GELU}(\boldsymbol{g}'_{\pi(i)}) + \boldsymbol{b}_o = \boldsymbol{h}^{\ell}_i + \boldsymbol{W}_o\,\mathrm{GELU}(\boldsymbol{g}_i) + \boldsymbol{b}_o = \boldsymbol{h}^{\ell+1}_i$. The output of node $\pi(i)$ on $(\boldsymbol{H}', \mathcal{Q}')$ thus equals the output of node i on $(\boldsymbol{H}, \mathcal{Q})$, which is Eq. (20). On heterogeneous graphs, the input map of a node depends only on its type, which π preserves, and the same argument therefore covers the input embeddings. With dropout, the identity holds in distribution, and it holds for each sample if the dropout masks are permuted together with the nodes. □

B.5 THE FACTORIZED VARIANT

To keep the notation light, we state the result on an undirected homogeneous graph, where every weight $\omega_{i,q}$ equals one and the role matrices are tied. Part (i) carries over to other graphs with the role matrix of each neighbor in place of $\boldsymbol{P}$. Consider the variant of the layer that applies the GELU to the direct term alone and adds the interaction terms after the sum over interactions,

$$\tilde{\boldsymbol{g}}_i = \frac{\boldsymbol{D}_0\boldsymbol{x}_i + \sum_{q=\{i,v\}\in\mathcal{Q}(i)} \mathrm{GELU}(\boldsymbol{D}\boldsymbol{x}_v)}{1+n_i} + \frac{1}{1+n_i}\sum_{q=\{i,v\}\in\mathcal{Q}(i)} \boldsymbol{\gamma}_i \odot \boldsymbol{c}_q + \boldsymbol{b}. \tag{24}$$

The variant has the same parameters as LINC and differs only in where the interaction terms meet the GELU. Its first term is the aggregation of the message-passing layer in Proposition 2, and we call its second term the aggregated interaction term.

Proposition 4 (The factorized variant)**.** *(i) The aggregated interaction term of Eq.* (24) *at node u depends on the neighborhood of u only through its size n_u and $\boldsymbol{s}_u = \sum_{q=\{u,v\}\in\mathcal{Q}(u)} \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v)$. (ii) There exist parameters, an embedding $\boldsymbol{x}_u$, and two neighborhoods of equal size with equal $\boldsymbol{s}_u$ and equal $\sum_v \boldsymbol{x}_v$ for which the sum $\sum_{q\in\mathcal{Q}(u)} \boldsymbol{m}_{u|q}$ of the consequences of LINC differs between the two neighborhoods.*

Proof. (i) Fix a node u. Since the vectors $\boldsymbol{\gamma}_u$ and $\mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u)$ do not depend on the interaction, the map

$$\boldsymbol{w} \;\mapsto\; \boldsymbol{\gamma}_u \odot \boldsymbol{B}\big[\sqrt{r}\ \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u) \odot \boldsymbol{w}\big] \tag{25}$$

is the same for every $q \in \mathcal{Q}(u)$. The map is linear in $\boldsymbol{w}$, since the Hadamard product with a fixed vector and the multiplication by $\boldsymbol{B}$ are both linear, and a linear map commutes with a finite sum. Taking $\boldsymbol{w} = \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v)$ for each interaction,

$$\begin{aligned}\sum_{q=\{u,v\}\in\mathcal{Q}(u)} \boldsymbol{\gamma}_u \odot \boldsymbol{c}_q &= \sum_{q=\{u,v\}\in\mathcal{Q}(u)} \boldsymbol{\gamma}_u \odot \boldsymbol{B}\big[\sqrt{r}\ \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u) \odot \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v)\big] \\ &= \boldsymbol{\gamma}_u \odot \boldsymbol{B}\Big[\sqrt{r}\ \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u) \odot \sum_{q=\{u,v\}\in\mathcal{Q}(u)} \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_v)\Big] \\ &= \boldsymbol{\gamma}_u \odot \boldsymbol{B}\big[\sqrt{r}\ \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u) \odot \boldsymbol{s}_u\big].\end{aligned} \tag{26}$$

The aggregated interaction term is this vector divided by $1+n_u$ and hence depends on the neighbors of u only through n_u and $\boldsymbol{s}_u$. The argument holds for any interaction weights that do not depend on the embeddings of the neighbors.

(ii) Take $\boldsymbol{C} = \boldsymbol{0}$ and $\boldsymbol{D} = \boldsymbol{0}$. Then $\boldsymbol{\gamma}_u = \boldsymbol{1}$, the direct term vanishes, and $\boldsymbol{m}_{u|q} = \mathrm{GELU}(\boldsymbol{c}_q)$ for every interaction of u. For a neighbor with embedding $\boldsymbol{a}$, the shared content of its interaction with u is

$$\boldsymbol{c}_{\boldsymbol{a}} = \boldsymbol{B}\big[\sqrt{r}\ \mathrm{norm}(\boldsymbol{P}\boldsymbol{x}_u) \odot \mathrm{norm}(\boldsymbol{P}\boldsymbol{a})\big]. \tag{27}$$

Since norm is odd, $\boldsymbol{c}_{-\boldsymbol{a}} = -\boldsymbol{c}_{\boldsymbol{a}}$. For two neighbor embeddings $\boldsymbol{a}$ and $\boldsymbol{b}$, consider the neighborhoods

$$\mathcal{N}_1 = \{\boldsymbol{a}, -\boldsymbol{a}\}, \qquad \mathcal{N}_2 = \{\boldsymbol{b}, -\boldsymbol{b}\}. \tag{28}$$

Both have two nodes and $\sum_v \boldsymbol{x}_v = \boldsymbol{0}$, and both have $\boldsymbol{s}_u = \boldsymbol{0}$ because $\mathrm{norm}(-\boldsymbol{P}\boldsymbol{a}) = -\mathrm{norm}(\boldsymbol{P}\boldsymbol{a})$. By part (i), the aggregated interaction term of the variant is therefore the same for both neighborhoods, and its first term is zero for both, since $\mathrm{GELU}(\boldsymbol{D}\boldsymbol{x}_v) = \mathrm{GELU}(\boldsymbol{0}) = \boldsymbol{0}$. For LINC, Eq. (6) gives, entry by entry,

$$\sum_{\boldsymbol{x}_v\in\mathcal{N}_1} \boldsymbol{m}_{u|q} = \mathrm{GELU}(\boldsymbol{c}_{\boldsymbol{a}}) + \mathrm{GELU}(-\boldsymbol{c}_{\boldsymbol{a}}) = g(\boldsymbol{c}_{\boldsymbol{a}}), \qquad \sum_{\boldsymbol{x}_v\in\mathcal{N}_2} \boldsymbol{m}_{u|q} = g(\boldsymbol{c}_{\boldsymbol{b}}), \tag{29}$$

with $g(t) = t\,(2\Phi(t) - 1)$. Since g is even and strictly increasing in $|t|$, the two sums differ whenever $|\boldsymbol{c}_{\boldsymbol{a}}|$ and $|\boldsymbol{c}_{\boldsymbol{b}}|$ differ in some entry. This holds, for instance, for the choice of $\boldsymbol{P}$, $\boldsymbol{B}$, $\boldsymbol{x}_u$, and $\boldsymbol{a} = \boldsymbol{x}_v$ in Step 4 of the proof of Theorem 1 with all other rows of $\boldsymbol{P}$ set to zero. Then $\boldsymbol{c}_{\boldsymbol{a}} \neq \boldsymbol{0}$, $\boldsymbol{P}$ has rank one, and any nonzero $\boldsymbol{b}$ with $\boldsymbol{P}\boldsymbol{b} = \boldsymbol{0}$ gives $g(\boldsymbol{c}_{\boldsymbol{b}}) = \boldsymbol{0} \neq g(\boldsymbol{c}_{\boldsymbol{a}})$. A neighborhood of the form $\{\boldsymbol{a}, -\boldsymbol{a}\}$ thus cancels the odd part of the GELU and keeps its even part, which records the size of each interaction term, whereas the variant sums before any nonlinearity and gives the same output for both neighborhoods. □

A related separation is known for functions on sets, where networks that process pairs of elements before pooling can represent functions that need exponential width when the elements are processed one at a time (Zweig & Bruna, 2022).

C Experimental Settings

C.1 Datasets

Homophilic undirected graphs.

- **Amazon-Photo** (Shchur et al., 2018). Nodes are products in the Photo category of the Amazon co-purchase graph, and edges connect products that are often bought together. Node features are bag-of-words vectors of product reviews, and the task is to predict the product category.
- **WikiCS** (Mernyei & Cangea, 2020). Nodes are Wikipedia articles on computer science, and edges are hyperlinks between them. Node features are averaged GloVe embeddings of the article text, and the task is to predict the branch of computer science of an article.

Heterophilic undirected graphs.

- **Chameleon-F** (Rozemberczki et al., 2021; Platonov et al., 2023b). Nodes are Wikipedia pages on chameleons, and edges are hyperlinks between them. Node features indicate informative nouns in the pages, and the task is to predict the average monthly traffic of a page, grouped into classes. The filtered version removes duplicated nodes.
- **Roman-Empire** (Platonov et al., 2023b). Nodes are the words of the Wikipedia article on the Roman Empire, and edges connect words that are adjacent in the text or linked by a syntactic dependency. Node features are fastText embeddings of the words, and the task is to predict the syntactic role of a word.
- **Amazon-Ratings** (Platonov et al., 2023b). Nodes are products of the Amazon co-purchase graph, such as books, music CDs, DVDs, and VHS tapes, and edges connect products that are often bought together. Node features are fastText embeddings of the product descriptions, and the task is to predict the average rating of a product.
- **Minesweeper** (Platonov et al., 2023b). Nodes are the cells of a 100×100 grid, and edges connect neighboring cells. Node features are one-hot encodings of the number of neighboring mines, which are hidden for a random half of the nodes, and the task is to predict whether a cell is a mine.
- **Tolokers** (Platonov et al., 2023b). Nodes are workers of the Toloka crowdsourcing platform, and edges connect two workers who worked on the same task. Node features are based on the profile and task performance of a worker, and the task is to predict whether a worker has been banned in one of the projects.
- **Questions** (Platonov et al., 2023b). Nodes are users of the question-answering website Yandex Q, and an edge connects two users if one answered a question of the other. Node features are averaged fastText embeddings of the user descriptions, and the task is to predict whether a user remained active on the website.

Heterogeneous graphs.

- **IMDB** (Lv et al., 2021). Nodes are the movies, directors, actors, and keywords of a movie database, linked by several relations. The task is to predict the genres of a movie, and a movie can have several genres.
- **Freebase** (Lv et al., 2021). Nodes are the books, films, music, sports, people, locations, organizations, and businesses of a subgraph of the Freebase knowledge graph, linked by several relations. The nodes carry no raw features, and the task is to predict the category of a book.
- **Alibaba** (Fu et al., 2023). We use the small Alibaba graph of DualHGCN (Xue et al., 2021) with the labels, features, and split released with BPHGNN. Nodes are items and users, linked by clicks, inquiries, and contacts, and the task is to predict the category of an item. The released labels give several categories to 474 of the 1,869 items. Following the released loader, every method takes the first category with the largest label value, and the task is therefore single-label classification over five classes.

- **Taobao** (Yu et al., 2022). We use the Taobao subset released with MHGCN. Nodes are items and users, linked by purchases, clicks, favorites, and additions to the cart, and the task is to classify items into three categories.

Directed graphs.

- **Cora-ML** (Bojchevski & Günnemann, 2018). Nodes are machine learning papers, and each edge points from a citing paper to a cited paper. Node features are bag-of-words vectors of the papers, and the task is to predict the topic of a paper.
- **Roman-Empire** (Rossi et al., 2024). We use the directed release of Dir-GNN, which keeps the native directed edges between the words. The release differs slightly from the undirected version in labels, masks, and topology.
- **MGTAB** (Shi et al., 2023). Nodes are annotated Twitter accounts, and each edge points from an account to an account it follows. The edges combine the friend lists and the reversed follower lists of the release. Node features are the 788 features of the release, and the task is to predict whether an account is a bot.
- **ogbn-arxiv** (Hu et al., 2020a). Nodes are arXiv computer science papers, and each edge points from a citing paper to a cited paper. Node features are averaged word embeddings of the title and the abstract, and the task is to predict the subject area of a paper.

C.2 Dataset settings

Table 5 lists the statistics and settings of every dataset.

Table 5: Statistics and settings of the datasets. Types and Relations count node types and relations, and hom is the adjusted homophily, computed on directed graphs without the edge directions. Split gives the rounded percentages of labeled nodes for training, validation, and test. Undirected edges count each node pair once, and directed edges count each arc once, both without duplicates and self-loops. Edges and relations of a heterogeneous graph count the stored relation arcs and types, including inverse relations and, on Freebase, 88,049 self-loop records. Features of a heterogeneous graph refer to its target node type.

Dataset	Nodes	Edges	Features	Classes	Types	Relations	hom	Split	Runs	Metric
Amazon-Photo	7,650	119,081	745	8	1	1	0.79	60/20/20	3 seeds	Acc
WikiCS	11,701	215,603	300	10	1	1	0.58	5/45/50	10 splits	Acc
Chameleon-F	890	8,854	2,325	5	1	1	0.03	48/32/20	10 splits	Acc
Roman-Empire	22,662	32,927	300	18	1	1	-0.05	50/25/25	10 splits	Acc
Amazon-Ratings	24,492	93,050	300	5	1	1	0.14	50/25/25	10 splits	Acc
Minesweeper	10,000	39,402	7	2	1	1	0.01	50/25/25	10 splits	ROC-AUC
Tolokers	11,758	519,000	10	2	1	1	0.09	50/25/25	10 splits	ROC-AUC
Questions	48,921	153,540	301	2	1	1	0.02	50/25/25	10 splits	ROC-AUC
IMDB	21,420	86,642	3,489	5	4	6	n/a	24/6/70	5 splits	Ma-F1, Mi-F1
Freebase	180,098	1,057,688	none	7	8	36	n/a	24/6/70	5 splits	Ma-F1, Mi-F1
Alibaba	15,218	54,024	7	5	2	6	n/a	20/20/60	5 seeds	Ma-F1, Mi-F1
Taobao	21,318	41,194	19	3	2	8	n/a	80/10/10	5 seeds	Ma-F1, Mi-F1
Cora-ML	2,995	8,416	2,879	7	1	1	0.75	50/25/25	10 splits	Acc
Roman-Empire	22,662	39,143	300	18	1	1	-0.05	50/25/25	10 splits	Acc
MGTAB	10,199	425,863	788	2	1	1	-0.04	70/20/10	10 splits	Acc
ogbn-arxiv	169,343	1,166,243	128	40	1	1	0.59	54/18/29	3 seeds	Acc

Metrics. Accuracy (Acc) is the fraction of correctly classified test nodes. Macro-F1 (Ma-F1) is the unweighted mean of the per-class F1 scores, and Micro-F1 (Mi-F1) is the F1 score pooled over all test nodes and labels. ROC-AUC is the area under the receiver operating characteristic curve.

Homophily. The adjusted homophily (Platonov et al., 2023a) of a graph is

$$\begin{aligned} \hom(\mathcal{G}) &= \frac{\hom_{\mathrm{e}}(\mathcal{G}) - \sum_{c\in\mathcal{Y}} p_c^2}{1 - \sum_{c\in\mathcal{Y}} p_c^2}, \\ \hom_{\mathrm{e}}(\mathcal{G}) &= \frac{1}{m} \sum_{(u,v)\in\mathcal{Q}} \mathbb{I}[y_u = y_v], \qquad p_c = \frac{1}{2m} \sum_{i\in\mathcal{V}:\, y_i=c} n_i, \end{aligned} \tag{30}$$

where $\hom_e$ is the edge homophily, the share of interactions whose participants have the same label, n_i is the number of interactions node i takes part in, and p_c is the share of all node degrees held by class c. It is high on homophilic graphs and close to zero or negative on heterophilic graphs.

Splits. Amazon-Photo uses the fixed 60/20/20 split provided in the code release of Rubio-Madrigal & Burkholz (2026), with three initialization seeds. WikiCS uses its first ten official splits, and its official validation and stopping sets together serve for validation (Mernyei & Cangea, 2020). Chameleon-F uses the ten splits distributed with the filtered graph, and Roman-Empire, Amazon-Ratings, Minesweeper, Tolokers, and Questions use the ten 50/25/25 splits of Platonov et al. (2023b). IMDB and Freebase use the official test sets of HGB (Lv et al., 2021), which hold out about 70% of the labeled nodes, and five random 80/20 splits of the remaining labeled nodes into training and validation sets. Alibaba uses the 20/20/60 split in the files released with BPHGNN (Fu et al., 2023), although its paper states 20/40/40, and Taobao uses the 80/10/10 split released with MHGCN (Yu et al., 2022), each with five seeds. Cora-ML uses ten random 50/25/25 splits generated as in Dir-GNN (Rossi et al., 2024), and the directed Roman-Empire uses the ten splits released with Dir-GNN. MGTAB uses ten random 70/20/10 splits following its benchmark (Shi et al., 2023), and ogbn-arxiv uses the official time split with three seeds (Hu et al., 2020a).

Preprocessing. Self-loops stored in the directed files are removed, and a model that needs a self term adds its own. IMDB is a multilabel task trained with sigmoid outputs and binary cross-entropy, and the other tasks use softmax outputs and cross-entropy.

C.3 Baselines

Within each group below, the baselines are listed in chronological order.

Undirected graphs.

- *Structure-agnostic model.* MLP applies a multilayer perceptron to the node features and ignores the graph.
- *Classic GNNs.* GCN (Kipf & Welling, 2017) averages the transformed features of the neighbors with a symmetric degree normalization. GraphSAGE (Hamilton et al., 2017) concatenates the embedding of a node with an aggregate of its sampled neighbors. APPNP (Gasteiger et al., 2019) propagates the predictions of an MLP with personalized PageRank.
- *Attention models.* GAT (Veličković et al., 2018) weights each neighbor by an attention coefficient computed from both endpoints. GATv2 (Brody et al., 2022) applies the non-linearity before the attention vector, which makes the ranking of the neighbors depend on the receiving node. GATE (Mustafa & Burkholz, 2024) allows a node to switch off the aggregation of its neighbors when the neighbors do not help.
- *Heterophilic GNNs.* MixHop (Abu-El-Haija et al., 2019) mixes the features aggregated over different powers of the adjacency matrix. GPR-GNN (Chien et al., 2021) learns signed weights over the steps of a generalized PageRank propagation. FSGNN (Maurya et al., 2021) selects the features aggregated over different hops with soft weights and normalizes each hop. GloGNN (Li et al., 2022) aggregates from all nodes with a learned coefficient matrix that captures global homophily. ACM-GCN (Luan et al., 2022) mixes low-pass, high-pass, and identity channels adaptively for each node. GAMMA (Ahsaei et al., 2025) combines multi-hop messages through node-dependent gates to handle varying levels of homophily.
- *Graph transformers.* SGFormer (Wu et al., 2023) combines one layer of global attention in linear time with a GNN. CoBFormer (Xing et al., 2024) applies attention within and across clusters of nodes and trains it together with a GCN by collaborative learning.
- *Edge-aware models.* CensNet (Jiang et al., 2019) alternates convolutions on the graph and on its line graph to update node embeddings and edge states. GNN-FiLM (Brockschmidt, 2020) modulates each incoming message with a feature-wise scale and shift computed from the receiving node. EGAT (Chen & Chen, 2021) updates node embeddings and edge states jointly through attention. EB-GNN (Barceló et al., 2026) keeps edge representations and passes messages between edges, and EB-GNN† adds a node-classification head to its edge encoder and edge-to-node readout.

Heterogeneous graphs. Following Yang et al. (2023), a model is metapath-based if it first aggregates neighbors within each relation or metapath and then fuses the results, and metapath-free if it embeds the types into the messages of one aggregation.

- *Structure-agnostic model.* MLP is the same model as on undirected graphs.
- *Homogeneous GNNs.* GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018) are the same models as on undirected graphs.
- *Metapath-based models.* R-GCN (Schlichtkrull et al., 2018) aggregates the neighbors of each relation with a separate weight matrix and sums the results. HAN (Wang et al., 2019b) applies node-level attention within each metapath and semantic-level attention across metapaths. HetGNN (Zhang et al., 2019) samples heterogeneous neighbors by random walks with restart and aggregates them by type with recurrent networks.
- *Metapath-free models.* RSHN (Zhu et al., 2019) encodes the relation structure with a coarsened line graph and uses the resulting relation representations in heterogeneous message passing. HetSANN (Hong et al., 2020) applies type-aware attention directly to neighbors of different types. HGT (Hu et al., 2020b) parameterizes attention and message transformations by node types and relations. Simple-HGN (Lv et al., 2021) extends GAT with learnable relation embeddings, residual connections, and output normalization. HINormer (Mao et al., 2023) encodes the local structure and the heterogeneity of a node and applies a graph transformer to its context. SlotGAT (Zhou et al., 2023) keeps a separate slot of the representation for each node type and passes messages slot by slot with attention.
- *Multiplex models.* MHGCN (Yu et al., 2022) learns relation weights and aggregates multiplex heterogeneous structures with graph convolutions along relation paths of different lengths. BPHGNN (Fu et al., 2023) models depth and breadth behavior patterns in multiplex heterogeneous graphs and aligns the two representations through contrastive learning. MRiemGNN (Li et al., 2026) combines relation-aware Euclidean and Riemannian representations through kernel-based encoding and mutual learning across the two spaces.

Directed graphs.

- *Spectral models.* DiGCN-IB (Tong et al., 2020) convolves over higher-order directed proximity and combines several orders in inception blocks. MagNet (Zhang et al., 2021b) encodes the direction of edges in the complex phase of a magnetic Laplacian. LightDiC (Li et al., 2024) computes a digraph convolution based on the magnetic Laplacian before training and trains a light model on the result. FaberNet (Koke & Cremers, 2024) applies spectral filters built from Faber polynomials to the non-symmetric adjacency matrix.
- *Spatial models.* Dir-GNN (Rossi et al., 2024) aggregates incoming and outgoing neighbors separately. CoED (Pahng & Hormoz, 2025) learns a continuous direction for every edge and aggregates incoming and outgoing messages separately. IBG-NN (Kouchly et al., 2026) approximates the graph with intersecting blocks under a densifying objective and computes efficiently on the resulting representation.

C.4 Baseline settings

Protocol. All methods share the task, the data version, the splits, and the evaluation code, and the test set never enters model selection. Each method keeps its own preprocessing, optimizer, architecture, and stopping rule. For GCN, GAT, and GraphSAGE on the undirected graphs, we use the implementations and configurations of the heterophily benchmark (Platonov et al., 2023b), with residual connections, layer normalization, and feed-forward modules. Features with signed continuous values are L2-normalized instead of row-normalized, and dense embeddings stay continuous.

Configurations. When the authors of a baseline published a configuration for the same dataset version, split, graph, and task, we use that configuration without search. Otherwise, we tune the baseline over the search space given in the paper or the official code, with at most 128 valid configurations, or five on the large ogbn-arxiv. A space of this size is enumerated, and a larger or continuous space is searched with the tree-structured Parzen estimator (TPE) (Bergstra et al., 2011), of which the first ten trials are random, or the first two on ogbn-arxiv. Every candidate is run three times, and its mean validation score guides the search. The cap is an upper limit, and smaller grids are enumerated in full, such as the 36 configurations of GATv2 and the six of GAMMA. Runs that fail, for example with out-of-memory errors or non-finite losses, receive no score and do not count toward the cap, and proposals that repeat an evaluated configuration are skipped. One configuration is selected per model and dataset and used for all splits. EB-GNN† runs out of memory on Amazon-Photo, WikiCS, Tolokers, and Questions, and BPHGNN on Freebase, even with their smallest configurations. On Amazon-Ratings, 12 of the 18 configurations of EB-GNN† run out of memory, and we select the best

of the six valid configurations on the validation set. On heterogeneous graphs, GCN and GAT receive the graph with all node types and relations merged, and MRiemGNN uses our splits instead of its own resplitting.

C.5 Settings of LINC

Configurations. LINC is tuned with TPE over 128 valid configurations, and every candidate is run three times and ranked by its mean validation score. The search space covers the depth $L \in \{1, 2, 3, 4, 6\}$, the hidden dimension $d \in \{32, 64, 128, 256\}$, the interaction dimension $r \in \{16, 32, 64, 128, 256\}$, and the weight decay in $\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. The learning rate is optimized over the continuous interval $[10^{-4}, 10^{-2}]$ on a logarithmic scale, as in prior graph learning studies (Wang et al., 2023; Chen et al., 2025), and the dropout rate over the continuous interval $[0, 0.6]$. On ogbn-arxiv, LINC is tuned over five configurations, like the baselines. Table 6 lists the selected configurations.

Graph input. Every interaction is stored once. On undirected graphs LINC ties the matrices of the two roles, and on directed graphs it keeps them separate and receives the original edge directions. On heterogeneous graphs LINC uses one input map per node type and the relation-balanced weights, and keeps the matrices of the two roles separate. A relation and its reverse are merged into one relation when their edge sets are exact reverses of each other, and node types without features receive a constant input.

Training. We train LINC with AdamW (Loshchilov & Hutter, 2019) in single precision and clip gradients at norm one. Training runs for at most 3,000 epochs and stops early after 80 epochs without improvement on the validation criterion. Checkpoints are selected by the main validation metric, with ties broken by Macro-F1 and then the negative log-likelihood, or on heterogeneous graphs by Micro-F1 and then the negative loss.

Table 6: Configurations selected for LINC on the validation set.

Dataset	L	d	r	Dropout	Learning rate	Weight decay
Amazon-Photo	3	128	128	0.50	3.2e-3	1e-5
WikiCS	3	64	64	0.56	2.5e-3	0
Chameleon-F	6	256	64	0.09	2.4e-4	1e-5
Roman-Empire	6	64	128	0.34	7.8e-3	1e-4
Amazon-Ratings	3	256	256	0.41	1.2e-3	0
Minesweeper	6	32	32	0.32	3.5e-3	1e-6
Tolokers	4	32	32	0.43	9.5e-3	1e-6
Questions	6	256	128	0.33	1.0e-4	1e-4
IMDB	6	256	32	0.48	4.7e-4	0
Freebase	6	256	128	0.30	9.9e-4	0
Alibaba	6	256	16	0.60	9.2e-3	1e-6
Taobao	3	64	128	0.01	9.0e-3	1e-4
Cora-ML	3	256	128	0.50	2.3e-4	1e-6
Roman-Empire	6	64	256	0.43	2.7e-3	1e-3
MGTAB	4	128	32	0.34	1.2e-3	1e-2
ogbn-arxiv	4	128	128	0.27	1.7e-3	0

C.6 Computing environment

All experiments run on NVIDIA A100-PCIE-40GB GPUs. The reported LINC models were trained with PyTorch 2.12.1 (CUDA 13.0 build), and each baseline runs in its official code base with the libraries it requires. The efficiency measurements in Appendix D.4 use PyTorch 2.12.1, PyG 2.7.0, and CUDA runtime 13.0.

D Additional Results

D.1 Main results

Setting. We report Macro-F1 on the undirected and directed graphs, computed from the same runs and checkpoints as the accuracy.

Table 7: Node classification results on homophilic and heterophilic graphs (mean ± standard deviation, %). We report Macro-F1. Best and second-best results are in **bold** and underlined.

	Homophilic		Heterophilic						
Method	Amazon-Photo	WikiCS	Chameleon-F	Roman-Empire	Amazon-Ratings	Minesweeper	Tolokers	Questions	Avg. rank
MLP	55.66±1.85	51.97±3.69	24.19±4.83	41.45±2.01	16.42±1.07	44.44±0.00	43.87±0.00	49.27±0.09	17.31
GCN	94.23±0.24	76.10±0.86	**40.67±4.78**	66.31±0.95	41.85±0.66	74.03±1.06	68.11±2.28	59.34±1.16	6.38
GraphSAGE	94.33±0.05	75.91±0.52	34.30±5.81	78.20±0.62	**47.59±0.87**	77.62±0.82	57.99±1.25	60.34±1.42	5.50
APPNP	91.68±0.24	71.67±2.54	23.40±3.32	38.82±1.49	20.58±0.29	44.53±0.15	44.81±2.96	49.24±0.00	15.44
GAT	93.27±0.17	76.45±1.06	37.89±2.94	75.15±0.74	42.25±0.84	75.80±1.05	66.28±1.25	60.13±0.82	6.00
GATv2	67.48±6.55	75.29±1.14	22.17±3.21	38.23±1.41	20.39±0.37	44.44±0.00	43.92±0.11	49.24±0.00	16.75
GATE	91.33±0.17	75.16±0.62	32.26±2.36	64.84±0.97	41.77±1.49	74.41±3.21	65.29±1.31	56.78±1.00	9.88
MixHop	89.82±0.15	70.58±1.23	33.62±4.20	17.57±0.34	20.22±0.28	51.04±4.14	46.23±0.84	49.24±0.00	15.06
GPR-GNN	94.20±0.31	76.72±0.85	22.28±8.04	48.00±0.79	28.06±1.15	53.09±2.46	48.06±3.80	55.06±1.12	12.12
FSGNN	95.49±0.31	77.85±0.59	32.06±4.94	72.57±0.79	28.86±1.64	74.96±0.93	65.35±1.95	52.34±0.94	7.62
GloGNN	95.00±0.14	67.43±1.28	37.94±4.40	58.32±0.87	43.30±0.41	51.55±1.23	55.95±1.82	58.73±0.97	9.38
ACM-GCN	**95.80±0.05**	67.87±6.06	35.25±3.94	70.38±0.78	21.12±0.33	75.79±0.78	59.13±2.71	56.28±2.42	8.50
GAMMA	95.29±0.20	40.74±5.43	31.66±3.70	15.90±2.47	10.76±0.00	75.04±3.36	52.22±1.43	49.24±0.00	14.19
SGFormer	95.14±0.52	76.46±1.05	33.16±3.75	70.07±0.77	45.87±0.95	65.01±1.13	66.12±1.85	56.11±1.04	7.25
CoBFormer	94.17±0.35	**78.04±0.79**	38.82±4.24	53.65±1.02	42.26±0.58	44.59±0.15	62.22±3.08	56.16±1.64	8.38
CensNet	86.39±3.00	60.10±11.79	24.81±8.23	38.68±3.23	25.73±1.26	56.99±6.61	43.92±0.16	49.24±0.00	15.88
GNN-FiLM	94.81±0.53	69.95±1.37	24.42±3.23	86.90±0.62	35.74±2.07	81.35±1.12	59.71±3.44	59.99±1.57	7.88
EGAT	93.64±0.31	70.84±2.00	15.00±7.89	59.64±2.62	15.75±5.12	77.51±1.53	68.05±1.82	51.66±1.52	11.88
EB-GNN†	OOM	OOM	33.47±3.15	75.83±0.68	41.83±1.01	78.57±2.28	OOM	OOM	6.00
LINC	95.58±0.30	77.11±0.83	38.94±2.53	**89.36±0.80**	46.80±0.70	**84.60±0.91**	**72.22±1.87**	**60.62±1.56**	**1.62**

Table 8: Node classification results on directed graphs (mean ± standard deviation, %). We report Macro-F1. Best and second-best results are in **bold** and underlined.

Method	Cora-ML	Roman-Empire	MGTAB	ogbn-arxiv	Avg. rank
DiGCN-IB	86.77±1.28	17.73±0.80	77.05±2.33	31.46±0.51	6.50
MagNet	86.91±0.96	79.67±0.73	86.83±1.22	44.51±0.08	3.50
LightDiC	**87.28±0.85**	63.25±0.49	81.01±1.43	39.16±0.13	4.75
FaberNet	77.10±1.41	90.87±0.34	45.92±7.87	43.76±0.24	5.75
Dir-GNN	85.20±2.08	89.69±0.94	87.12±1.39	45.23±0.49	3.75
CoED	85.27±1.54	90.97±0.71	77.77±1.29	44.39±0.48	4.25
IBG-NN	73.41±2.45	50.48±0.86	87.88±0.92	40.09±0.70	5.75
LINC	86.49±1.10	**93.69±0.37**	**88.29±1.07**	**52.36±0.20**	**1.75**

Results. Tables 7 and 8 give the results. LINC achieves an average rank of 1.62 among 20 methods on homophilic and heterophilic graphs and 1.75 among eight methods on directed graphs.

D.2 Ablation study

Setting. We remove the direct term ($\boldsymbol{D}_\mathrm{s} = \boldsymbol{D}_\mathrm{t} = \mathbf{0}$), the interaction representation ($\boldsymbol{B} = \mathbf{0}$), and the own scales ($\boldsymbol{C} = \mathbf{0}$) alone and in pairs. Without the interaction representation, the layer is the message-passing layer of Proposition 2, and its enlarged version widens the hidden dimension until the parameter count is within 5% of LINC. Since the own scales act on the interaction representation, no variant keeps them without it. None of the variants is tuned again. They use the configuration selected on the validation set for LINC, apart from the wider hidden dimension of the enlarged variant, and are retrained from scratch with the same splits, seeds, and early stopping rule. The results thus show what the removed parts contribute under this configuration. For every graph, we report both metrics of all models.

Results. Tables 9 and 10 give both metrics of all models on every graph. Removing both the direct term and the interaction representation lowers the main metric on every graph. The full model is not the best variant on every graph. With the direct term in place, removing the own scales changes the main metric by at most 0.75 points and raises it slightly on 10 of the 16 graphs, by at most 0.48 points. Removing the interaction representation raises the main metric on five graphs, by at most 1.02 points on Chameleon-F, whose standard deviation is about 3 points, and lowers it by up to 4.52 points elsewhere. Most per-graph differences below 0.3 points lie within the run-to-run variation, and the conclusions in Section 5.3 rest on directions that hold across many graphs.

D.3 Hyperparameter analysis

Setting. We vary the depth $L \in \{1, 2, 3, 4, 6, 8, 16, 32\}$ and the interaction dimension $r \in \{8, 16, 32, 64, 128, 256\}$ one at a time on all 16 graphs, with all other settings fixed at the configuration selected for LINC. Models of different depths are trained separately with the same splits and seeds. The run at the selected configuration is repeated in the analysis, and its value can differ

Table 9: Ablation study results on all 16 graphs (mean, %) on the main metric, namely accuracy, ROC-AUC on Minesweeper, Tolokers, and Questions, and Macro-F1 on heterogeneous graphs. A check mark means the part is kept, and ‡marks the variant widened to the parameter count of LINC.

	Direct term		✓	✓‡			✓	✓
	Interaction representation				✓	✓	✓	✓
	Own scales					✓		✓
Homophilic	Amazon-Photo	90.41	95.58	95.58	95.36	95.25	96.27	96.45
	WikiCS	73.05	79.77	79.84	78.09	78.11	79.92	79.80
Heterophilic	Chameleon-F	37.07	43.75	41.57	39.83	39.68	43.02	42.73
	Roman-Empire	68.04	89.24	89.85	89.22	90.30	91.41	91.40
	Amazon-Ratings	39.89	50.24	50.76	49.72	50.03	51.98	52.40
	Minesweeper	50.80	95.75	95.72	93.44	94.04	95.51	95.47
	Tolokers	76.79	85.20	85.05	85.13	85.66	85.89	86.03
	Questions	75.19	79.35	79.39	75.89	75.91	79.41	79.41
Heterogeneous	IMDB	61.48	64.62	65.18	61.96	61.91	64.73	64.69
	Freebase	18.64	51.97	52.02	49.05	47.95	52.44	52.32
	Alibaba	21.82	44.74	44.93	29.07	31.67	44.10	44.86
	Taobao	29.49	29.32	32.54	34.15	34.50	34.22	33.85
Directed	Cora-ML	78.79	87.79	87.73	83.41	83.67	87.79	87.75
	Roman-Empire	67.73	93.42	93.94	92.89	93.52	94.49	94.68
	MGTAB	89.33	90.75	90.69	90.42	90.69	90.78	90.64
	ogbn-arxiv	55.14	71.69	71.66	70.80	71.01	72.06	71.58

Table 10: Ablation study results on all 16 graphs (mean, %) on the second metric, namely Macro-F1, or Micro-F1 on heterogeneous graphs. A check mark means the part is kept, and ‡marks the variant widened to the parameter count of LINC.

	Direct term		✓	✓‡			✓	✓
	Interaction representation				✓	✓	✓	✓
	Own scales					✓		✓
Homophilic	Amazon-Photo	88.49	94.57	94.53	94.19	94.12	95.50	95.58
	WikiCS	68.99	77.06	77.03	74.97	74.94	77.12	77.11
Heterophilic	Chameleon-F	32.01	40.86	37.75	33.35	34.11	39.38	38.94
	Roman-Empire	56.46	86.78	87.28	86.64	87.92	89.35	89.36
	Amazon-Ratings	26.04	43.84	44.63	44.29	44.44	46.29	46.80
	Minesweeper	44.44	85.16	85.02	80.88	82.26	84.88	84.60
	Tolokers	60.32	71.90	71.53	70.42	72.47	72.30	72.22
	Questions	57.57	60.13	59.47	59.33	59.33	60.55	60.62
Heterogeneous	IMDB	64.95	67.13	67.49	65.14	65.07	67.18	67.13
	Freebase	46.31	64.83	64.90	63.49	63.35	65.70	65.26
	Alibaba	27.83	47.74	47.95	34.99	37.25	46.90	48.07
	Taobao	37.31	34.03	35.42	36.52	37.36	35.72	36.07
Directed	Cora-ML	77.74	86.70	86.82	82.69	82.91	86.57	86.49
	Roman-Empire	55.57	92.07	92.68	91.47	92.31	93.46	93.69
	MGTAB	86.54	88.43	88.34	88.07	88.34	88.45	88.29
	ogbn-arxiv	35.05	51.63	52.06	50.24	51.02	52.26	52.36

slightly from Tables 1, 2, and 3. Because the configuration is selected on the validation set within a search space of at most six layers (see Appendix C.5), deeper settings are not part of Tables 1, 2, and 3, even though some of them reach higher scores, such as 32 layers on Minesweeper, 16 layers on the directed Roman-Empire, and 8 layers on Amazon-Photo.

Results. Figures 3 and 4 give the results on both metrics for all 16 graphs. The second metric moves with the main metric, and their Pearson correlation across the eight depths exceeds 0.8 on 14 of the 16 graphs.

D.4 Efficiency analysis

Setting. All measurements use the same GPU model and data (see Appendix C.6). We report the training time per epoch averaged after warm-up, the inference time with a fixed checkpoint, and the peak allocated GPU memory. On graphs with more than 80,000 nodes, LINC recomputes the activations of each layer in the backward pass, which lowers the peak memory at the cost of extra computation and leaves the results unchanged.

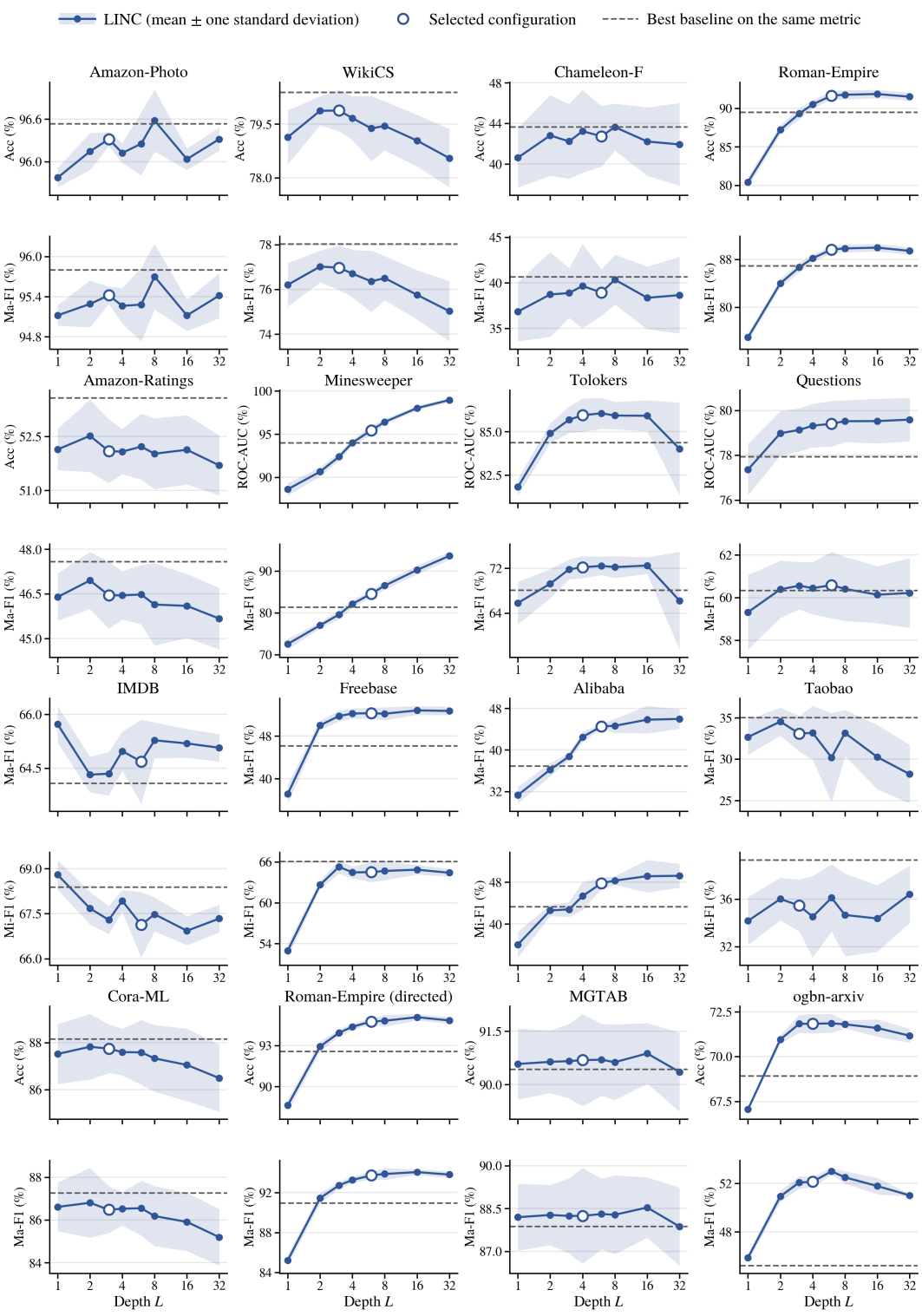


Figure 3: Hyperparameter analysis with the depth L on all 16 graphs. For each graph, the upper panel shows the main metric and the lower panel the second metric, namely Macro-F1, or Micro-F1 on heterogeneous graphs. The graphs are homophilic and heterophilic in the first four rows, heterogeneous in the next two rows, and directed in the last two rows.

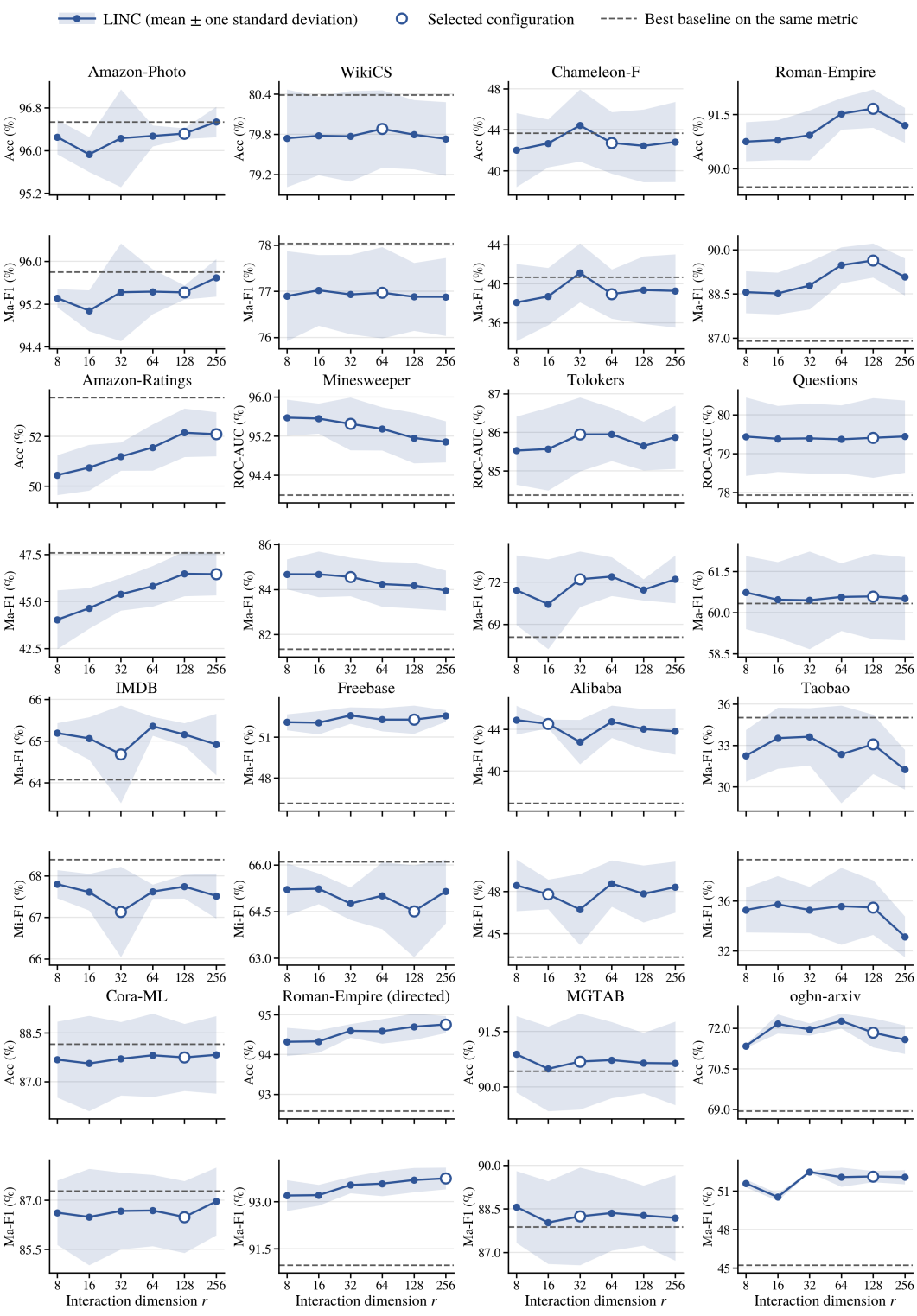


Figure 4: Hyperparameter analysis with the interaction dimension r on all 16 graphs, arranged as in Figure 3.

Results. Table 11 gives the results on Tolokers, Taobao, and Cora-ML, one graph of each group, and on ogbn-arxiv, the largest graph, with every method measured on the same GPU model. On ogbn-arxiv, the graph with the most interactions, LINC recomputes activations to save memory and

takes 514 ms per training epoch, of the same order as the 333 ms of IBG-NN, while its peak memory of 8,186 MiB stays below the 10,379 MiB of IBG-NN.

Table 11: Efficiency analysis results on one GPU model (see Appendix C.6), averaged over runs. Train and Inference are the training time per epoch and the inference time in ms, and Memory is the peak GPU memory in MiB. Lower is better, and best and second-best results are in **bold** and underlined. For HetGNN, Train is one outer iteration of its unsupervised training, and Inference includes generating its representations and running its classifier. On ogbn-arxiv, LINC recomputes activations to save memory, with a cost of the same order as IBG-NN and a lower peak memory.

Tolokers			
Method	Train	Inference	Memory
MLP	10.6	**0.2**	**36**
GCN	15.3	5.5	407
GraphSAGE	13.8	4.1	452
APPNP	**6.6**	2.1	76
GAT	33.1	9.0	2,588
GATv2	21.3	9.3	1,724
GATE	82.1	29.1	3,705
MixHop	10.4	1.7	141
GPR-GNN	7.3	2.3	78
FSGNN	7.8	1.9	249
GloGNN	32.3	4.7	171
ACM-GCN	9.6	1.1	160
GAMMA	12.7	3.9	299
SGFormer	38.6	14.4	2,340
CoBFormer	21.5	6.7	599
CensNet	18.3	3.0	821
GNN-FiLM	180.4	62.0	17,669
EGAT	112.7	43.7	8,216
EB-GNN†	OOM	OOM	OOM
LINC	58.6	18.4	2,760

Taobao			
Method	Train	Inference	Memory
MLP	**6.3**	**0.3**	**28**
GCN	19.3	2.2	117
GAT	34.5	7.9	723
R-GCN	36.8	1.3	358
HAN	15.7	2.9	66
HetGNN	212.7	42.9	426
RSHN	23.2	2.5	37
HetSANN	122.4	37.3	311
HGT	287.8	21.8	1,458
Simple-HGN	73.0	10.2	928
HINormer	299.9	91.0	14,222
SlotGAT	63.1	9.4	822
MHGCN	14.1	4.8	335
BPHGNN	98.2	23.4	7,435
MRiemGNN	47.6	20.4	1,161
LINC	26.7	3.8	440

Cora-ML			
Method	Train	Inference	Memory
DiGCN-IB	7.8	1.7	219
MagNet	13.9	5.4	352
LightDiC	8.2	**0.5**	**116**
FaberNet	21.1	4.2	328
Dir-GNN	**7.2**	1.8	186
CoED	16.5	3.2	482
IBG-NN	47.0	7.7	496
LINC	27.9	4.1	371

ogbn-arxiv			
Method	Train	Inference	Memory
DiGCN-IB	46.9	21.1	1,089
MagNet	61.1	37.8	1,194
LightDiC	**21.5**	23.8	**448**
FaberNet	74.6	36.1	1,817
Dir-GNN	26.1	**13.6**	945
CoED	99.2	51.8	2,657
IBG-NN	332.8	123.4	10,379
LINC	514.1	135.4	8,186