Providing Context: Enriching Graph Learning Using Language Model Priors

Anonymous authors
Paper under double-blind review

Abstract

We introduce GINAT, a graph-language augmentation framework that improves message-passing GNNs by conditioning their edge messages on LM-embedded global graph descriptions. Unlike text-attributed graph methods that rely on node/edge-attached text, GINAT leverages external prompts that capture graph semantics independently of the downstream task. These prompts are injected into message passing through a cross-attention mechanism that preserves permutation equivariance, enabling global contextual guidance within standard MPNN computation. Across a broad suite of node- and graph-level benchmarks, GINAT yields consistent gains over strong GNN baselines, reaching up to 23% average relative improvement. Ablations show that the gains arise from the semantic content of prompts rather than random vectors, and an analysis of the learned context scaling reveals where and how strongly different backbones and layers rely on language-conditioned information. Overall, GINAT establishes LM-embedded prompts as a practical, interpretable graph-language signal to strengthen graph representation learning. Our code is available at `https://anonymous.4open.science/r/GINAT-1312/`

1 Introduction

Graph neural networks (GNNs) have achieved remarkable success in the processing of structured non-Euclidean data Jumper et al. (2021); Xu et al. (2019). Early progress was driven by message-passing neural networks (MPNNs) such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and others, which maintain node permutation equivariance - the fundamental inductive bias for graph learning. Recent years have seen the rise of GNNs that go beyond this paradigm, such as Graph Transformers (Ying et al., 2021) and GraphMAE (Hou et al., 2022), which borrow architectures proven in other deep learning fields. In parallel, the rise of Large Language Models (LLMs) has sparked research in the graph learning community in two prominent directions (among others, see Appendix E). First, they facilitated the introduction of text-attributed graphs (TAGs), where nodes are attributed with text (Yan et al., 2023). Second, efforts were made to utilize the generalization capabilities of LLMs to directly process graphs by serializing them Fatemi et al. (2024).

Table 1: Relative performance gain by applying GINAT to different MPNNs using selected LMs, averaged across over 20 node- and graph-level datasets.

LM → MPNN ↓	Average gain	`RoBERTa large`	`DeBERTa base`	`Qwen3 4B-Base`
GAT	23.31%	23.59%	23.55%	22.78%
GATv2	22.20%	22.63%	23.64%	20.34%
GraphGPS	9.78%	10.03%	9.44%	9.86%
GCN	4.45%	5.70%	4.29%	3.35%
GIN	3.83%	3.93%	4.02%	3.53%
GraphConv	1.89%	1.79%	1.98%	1.91%

In this work, we take a different approach: we use Language Models (LMs) to provide domain context that augments message passing, while preserving the inductive bias of MPNNs.

Our central hypothesis is that the rich semantic priors of LMs can be leveraged to enhance graph learning. To understand how, we first ask *What do graphs and natural language have*

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

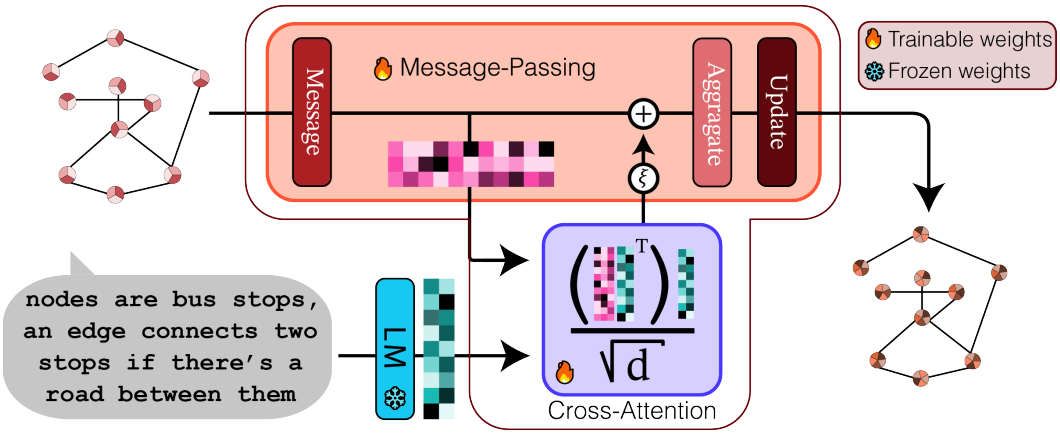

Figure 1: Illustration of a GINAT Layer. A graph with initial features is enriched with a prompt sampled from a dataset of prompts that describe its content. We augment message passing by cross-attending graph messages to the prompt embedding produced by a pretrained LM. The output of the cross-attention operation is then scaled by a learned parameter ξ and added to the original messages.

in common? Graphs offer a formal framework for decomposing concepts into explicit entities and relations, enabling rigorous representations that support mathematical analysis and computational processing. Natural language, in contrast, is a flexible medium for high-level discourse, enriching communication with context and prior knowledge (Gentner & Asmuth, 2019; Williams et al., 2017; Korta & Perry, 2024). In this work, we regard graphs and natural language as *complementary* representational paradigms: both are human-engineered abstractions devised to model, interpret, and reason about the world.

A core challenge in graph learning is that graph structure is rarely self-interpreting: the same motif can carry different semantics across domains, and typical datasets provide limited supervision to disambiguate it. As an intuitive example, consider a leaf node (a node v with $\text{degree}(v) = 1$). In an electric-circuit graph, it may represent an external connection (input, output, ground, etc.), whereas in a road-map graph, it may represent a dead-end or a cul-de-sac. More generally, the interpretation of motifs, node/edge types, and even labels depends on the domain *context* that is often not encoded in the graph. Even coarse domain-level context can substantially reduce this ambiguity: knowing whether a graph represents circuits, roads, or molecules constrains which motifs and relations can plausibly be interpreted. This suggests that supplying a natural-language description of the graph's domain can act as an interpretable prior that guides message passing.

Existing approaches only partially address this gap. TAG methods specifically process graphs defined over textual data, but they do not necessarily provide global domain/task semantics and are limited when graphs lack rich text annotations. We show they fail to generalize to our setting in Appendix F.9. Conversely, directly processing graphs via serialization can be constrained by LLM context windows and computational cost, and it often relies on ad-hoc orderings that break permutation invariance; it can also be awkward to represent fine-grained numeric attributes. Nevertheless, pretrained LMs capture broad contextual knowledge from large text corpora and can retrieve relevant information from natural-language descriptions. We elaborate on this in Section 3.

Building on this motivation, we propose GINAT - **G**raph **I**nference by **N**atural-language **A**ugmentation of **T**opology. This is a graph-language augmented model that conditions MPNN message passing on LM-embedded domain prompts while preserving permutation equivariance. Our method extends the theoretically-established MPNN paradigm, and provably maintains the permutation equivariance property by using the cross-attention mechanism (see Appendix I) for message augmentation.

First, GINAT embeds the prompt using any frozen, pretrained LM. During training, this allows learning from massive-scale linguistic priors. Then, the messages passed by each MPNN layer are cross-attended to the prompt embeddings, and the final message is constructed by

a scaled residual connection. Figure 1 illustrates the approach, and Section 2 discusses the model in-depth.

Our experiments show significant gains relative to multiple well-established MPNN backbones across multiple graph modalities, domains, and tasks. Table 1 presents these gains concisely. GINAT results are competitive with large-scale GNNs like Graphformer and GraphMAE, often surpassing them. Importantly, throughout our experiments, the prompts we use are independent of the specific task, and describe the graph data only, ensuring the gains do not stem from prior knowledge on the task itself. We discuss the construction of the prompt datasets in detail in Appendix B and Appendix A. We experiment with multiple LMs and LLMs as prompt encoders, and perform rigorous testing to establish that the gains derive from the context provided by the prompt and the LM prior, rather than the added computational benefits. To that end, we ablate the conditioning vector source (prompt embeddings vs. noise) and ensure baselines have similar parameter counts to GINAT.

Our analysis indicates that GINAT's effectiveness depends on the structural information already encoded in the messages supplied to cross-attention. When these messages distinguish individual relations, as in GAT and GATv2, the graph structure controls how each message attends to the prompt, allowing the LM prior to contribute relation-dependent contextual information. When the messages depend primarily on the source node, as in GraphConv, the prompt contribution is less structurally differentiated and consequently provides smaller marginal gains. We provide additional analysis and findings in Section 3. Together, these findings provide a mechanistic explanation for the improvements obtained by GINAT: The LM embedding improves performance by contextually transforming structural distinctions exposed by the underlying message-passing operation.

Our contributions are summarized as follows:

- We propose GINAT, a framework for augmenting MPNNs with LM-embedded prompts (Section 2). Importantly, GINAT is permutation-equivariant, which is crucial for graph learning.
- We provide an in-depth analysis of the method's inner workings by examining different success and failure modes, spanning different MPNN backbones and datasets (Section 3).
- We evaluate GINAT on node- and graph-level benchmarks, showing consistent performance gains, and use original LLM-augmented datasets of task-agnostic prompts together with ablations and controls for systematic analysis (Section 4).

2 Method

In this section, we first define GINAT, then describe how prompts are embedded and assigned, and finally discuss how we scale the augmentation and permutation equivariance. An overview of the architecture is presented in Figure 1.

Our framework extends on the classical message-passing paradigm in graph learning. Message-passing layers are conventionally divided into three abstract components, with architectures differing in their definition: message, aggregate, and update (see Appendix J). Depending on the specific MPNN architecture, a message is usually constructed as a transformation of node features, sometimes combined with edge features if present. Then, the messages received from neighbors are aggregated at each node.

GINAT accepts two inputs: the graph data and an external prompt or textual description. The core innovation of our method is that in contrast with TAG models, the textual information is not node-specific. Rather, it describes the entire graph or data domain—allowing for augmentation of existing graph data without the high labeling cost.

GINAT uses the backbone MPNN to construct the messages and embeds the text using a frozen LM. As discussed in depth in Section 3, this allows the framework to use not only the information in the prompt, but also in the learned LM prior. Then, our method conditions the messages on the embedded prompt using cross-attention (For the sake of completeness,

```
"Molecular structure where atoms are nodes and edges are
their connecting bonds"

"A graph where nodes are atoms and edges show their
covalent connections"

"Each node signifies an atom, and each edge signifies a
chemical bond."
```

Figure 2: Examples of prompts used to describe molecular graphs (e.g., OGBG-MolHIV and ZINC). See Appendix K for details on the prompt datasets.

we provide a formal introduction to the cross-attention mechanism in Appendix I). The conditioned message is then added to the residual message as defined in Equation 1. The MPNN's forward pass will proceed as originally defined using the conditioned message.

Formally, given message defined by the MPNN, an edge $(u, v) \in E$ with $u \in \mathcal{N}(v)$, and a prompt embedding $p \in \mathbb{R}^{k \times d_{LM}}$, where $\mathcal{C}$ is a cross-attention operator, we define message $m_{vu}^{(l)}$ that will be provided to the aggregate method as follows:

$$\tilde{m}_{vu}^{(l)} = \text{message}\left(h_v^{(l-1)}, h_u^{(l-1)}, e_{vu}\right), \qquad m_{vu}^{(l)} = \tilde{m}_{vu}^{(l)} + \xi^{(l)} \cdot \mathcal{C}^{(l)}\left(\tilde{m}_{vu}^{(l)}, p, p\right). \tag{1}$$

Here, $h_u^{(l-1)}, h_v^{(l-1)}$ are u, v's representations by the previous layer, and e_{vu} are the edge features, when available. $\xi^{(l)}$ is a learned scalar which we discuss in more depth in Section 2.2.

2.1 Cross-Attending Messages and Prompts

The original MPNN messages $\tilde{m}$ are used as queries and the embedded prompts p are used as keys and values in the cross-attention operation. Together with the $\xi^{(l)}$-scaled residual, this enriches messages by allowing them to selectively attend to contextualized prompt embeddings. Importantly, when $\xi^{(l)}$ is set to zero, the formulation reduces to the standard MPNN, so the MPNN's expressivity is preserved.

Prompts Embedding. In order to adapt to different variants of pretrained LMs, such as LLMs, masked LMs, and sentence encoders, we chose an embedding approach that applies the same to all of them. We tokenize each prompt and pass it through the LM, taking the last-layer output logits as an embedding $p \in \mathbb{R}^{k \times d_{LM}}$. We normalize the embedded prompts using LayerNorm and pass them to all GINAT layers. Thus, all GINAT layers receive the same (normalized) prompt embedding as input. Note that LMs play a *preprocessing role* in GINAT. We don't train or fine-tune them, and they are only used to embed the prompts in vectorized space, equipped with the LM's prior. The pairing of embedded prompts with graphs or nodes is not part of the architecture, but a task-related choice (see Appendix B).

Computational Complexity. Cross-attending an embedded prompt of k tokens with all messages passed along edges is $O(|E| \cdot k)$ per layer, thus the compute overhead is linear in the input graph and k (and linear in $|E|$ for fixed k). As we show in Figure 2 and in Appendix K, k values are small.

For node-level tasks, on each forward pass, we assign a single prompt drawn from the prompts dataset to a sampled subgraph (sampler step). We note the assignment is not fixed, and a sampled graph is attended with a different prompt on each forward pass.

2.2 Scaling the Cross-Attention Signal

$\xi^{(l)}$ in Equation 1 is a learned scalar serving two purposes. First, it prevents the cross-attention residual from dominating $m_{vu}^{(l)}$, which can degrade performance due to a scale mismatch between the cross-attention and MPNN signals. Section F.5 analyzes the impact of scaling this signal. It allows the model to learn the appropriate scale of the attention mechanism compared to the MPNN's output. Having a distinct scalar for each layer allows

Table 2: Graph-level results. GINAT results are highlighted in red. Best results for each MPNN–dataset pair are underlined. The best result on each dataset is in **bold**. Prompts are embedded using `Qwen3-4B-Base` (Yang et al., 2025). See Sec. 4.1 for more details.

Model ↓ / Metric →	ENZYMES Acc. ↑	PROTEINS Acc. ↑	NCI109 Acc. ↑	OGBG-MolHIV ROC-AUC ↑	ZINC MAE ↓	REDDIT-BINARY Acc. ↑	REDDIT-MULTI5K Acc. ↑
GCN	0.361 ±0.13	0.738 ±0.01	0.777 ±0.02	0.774 ±0.02	0.477 ±0.00	0.843 ±0.01	0.521 ±0.03
	0.361 ±0.08	0.750 ±0.06	0.786 ±0.01	0.779 ±0.01	0.473 ±0.11	0.883 ±0.03	0.547 ±0.03
GraphSAGE	0.389 ±0.10	0.756 ±0.02	0.771 ±0.01	0.775 ±0.01	0.517 ±0.02	0.510 ±0.07	0.198 ±0.02
	0.461 ±0.10	0.762 ±0.04	0.787 ±0.02	0.781 ±0.00	0.502 ±0.00	0.517 ±0.07	0.200 ±0.02
GAT	0.422 ±0.03	0.705 ±0.09	0.742 ±0.04	0.733 ±0.02	0.485 ±0.00	0.542 ±0.05	0.218 ±0.02
	0.467 ±0.09	0.717 ±0.06	0.770 ±0.00	0.751 ±0.01	0.400 ±0.07	0.923 ±0.01	0.552 ±0.03
GATv2	0.433 ±0.08	0.711 ±0.05	0.750 ±0.02	0.732 ±0.02	0.447 ±0.00	0.510 ±0.07	0.220 ±0.03
	0.456 ±0.01	0.750 ±0.07	0.758 ±0.00	0.735 ±0.01	0.385 ±0.06	0.865 ±0.03	0.531 ±0.04
GIN[1]	0.672 ±0.03	0.771 ±0.04	0.825 ±0.03	0.779 ±0.00	0.249 ±0.02	0.935 ±0.03	0.566 ±0.04
	0.683 ±0.05	**0.777** ±0.03	**0.829** ±0.02	**0.785** ±0.01	0.152 ±0.02	**0.945** ±0.01	0.568 ±0.03
GraphConv	0.517 ±0.21	0.744 ±0.04	0.807 ±0.01	0.778 ±0.00	0.260 ±0.00	0.917 ±0.02	0.570 ±0.03
	0.594 ±0.03	0.750 ±0.06	0.808 ±0.01	0.780 ±0.00	0.257 ±0.01	0.927 ±0.03	**0.572** ±0.04
GraphGPS	0.644 ±0.18	0.679 ±0.10	0.816 ±0.01	0.769 ±0.01	0.249 ±0.07	0.777 ±0.06	0.394 ±0.14
	0.694 ±0.03	0.750 ±0.02	0.816 ±0.00	0.772 ±0.01	0.181 ±0.01	0.928 ±0.01	0.510 ±0.01
Graphormer	0.422 ±0.11	0.684 ±0.05	0.789 ±0.00	0.731 ±0.01	0.422 ±0.00	0.761 ±0.01	0.466 ±0.02
GraphMAE	0.286 ±0.03	0.753 ±0.00	0.759 ±0.00	0.772 ±0.01	**0.144** ±0.01	0.880 ±0.00	0.484 ±0.01

the model to choose the point(s) in the message-passing process where the contextual augmentation contributes most.

Second, it provides insights into the model's utilization of the prompt signal. The value of $\xi^{(l)}$ indicates how important the model considers the information in the prompt embedding at layer l. The model can learn $\xi^{(l)} = 0$, implying that the added information is not relevant to the message-passing process at hop l. Furthermore, the model can learn that $\forall l$, $\xi^{(l)} = 0$, meaning that the provided external data does not provide any beneficial information. We elaborate on this idea in Appendix F.5.

2.3 GINAT Preserves Permutation Equivariance

Permutation equivariance is a core property of GNNs and graph learning techniques. While it is well-established that cross-attention is permutation equivariant, Theorem 2.1 establishes that our approach preserves the node permutation equivariance property of MPNNs, with a complete proof given in Appendix D:

Theorem 2.1 (GINAT Preserves Permutation Equivariance)**.** *Let* update, aggregate *be the update and aggregation methods of a permutation-equivariant MPNN layer (denote* aggregate *as* AGG *for brevity). Let* $h_v^{(l)}$ *be the representation of node* v *in the input to the current layer. Then for messages as defined in Eq. 1, the operation*

$$\text{update}\left(h_v^{(l)}, AGG\left(\left\{m_{vu}^{(l)} : u \in \mathcal{N}(v)\right\}\right)\right) \tag{2}$$

remains permutation-equivariant.

3 Analysis: How GINAT Uses Language Priors

As reflected in Tables 2, 3, GINAT's gains vary systematically across backbones and datasets, revealing how the language interacts with message passing. In this section, we provide the

[1]For datasets where edge features are available, we use GINE.

Table 3: Accuracy (%) on node classification benchmarks, under subgraph sampling regime (see Section 4.1 for more details). GINAT results are marked in red, and best results for each MPNN-dataset pair are underlined. The best result on each dataset is **bolded**. The prompt-embedding LM is `roberta-large`.

Model ↓	PubMed	CoCS	Computers	Photo	Chameleon	AIRPORTS EUROPE	AIRPORTS BRAZIL
GCN	70.28 ±1.15	90.45 ±1.45	84.11 ±0.93	89.70 ±1.00	37.44 ±0.48	43.00 ±1.00	57.10 ±5.00
	73.66 ±1.35	91.83 ±1.32	85.11 ±1.34	90.40 ±0.46	43.19 ±2.31	47.10 ±4.00	**71.40** ±10.00
GraphSAGE	69.41 ±6.41	94.47 ±0.98	88.87 ±0.80	94.05 ±2.08	44.91 ±1.32	31.70 ±5.00	40.40 ±8.00
	70.35 ±2.61	**95.26** ±0.20	90.03 ±1.45	94.61 ±0.95	45.79 ±3.04	34.90 ±3.00	53.50 ±7.00
GAT	71.75 ±1.55	90.16 ±1.03	84.82 ±0.99	89.39 ±0.73	40.32 ±2.10	42.20 ±6.00	54.70 ±6.00
	73.91 ±2.66	91.55 ±0.91	85.06 ±1.90	91.98 ±0.20	42.72 ±5.11	46.30 ±6.00	65.40 ±6.00
GATv2	72.64 ±4.66	91.42 ±1.55	85.95 ±0.26	91.58 ±1.57	45.46 ±2.47	36.58 ±1.99	50.00 ±0.00
	72.66 ±0.26	92.17 ±1.71	87.52 ±0.94	91.62 ±1.44	47.27 ±1.12	41.46 ±0.99	61.90 ±3.00
GIN	61.68 ±4.57	90.92 ±1.60	84.44 ±1.48	89.68 ±1.48	42.66 ±0.89	48.70 ±5.00	59.50 ±3.00
	62.51 ±2.98	91.66 ±0.71	86.92 ±1.84	89.81 ±0.50	43.03 ±2.28	**49.10** ±5.00	60.70 ±14.00
GraphConv	71.30 ±0.79	93.24 ±0.42	86.04 ±0.88	92.10 ±1.66	46.46 ±3.85	44.70 ±8.00	66.60 ±3.00
	71.46 ±1.87	94.61 ±0.93	86.70 ±0.61	92.22 ±1.23	**47.85** ±3.99	48.30 ±3.00	69.00 ±6.00
GraphGPS	58.30 ±4.12	94.15 ±0.91	87.95 ±1.01	93.96 ±1.22	44.74 ±1.54	45.50 ±4.00	52.30 ±8.00
	63.41 ±4.01	94.93 ±0.67	**90.25** ±1.77	**94.86** ±0.18	46.73 ±2.22	47.90 ±2.00	66.60 ±3.00

reasoning behind the mechanism, and show that our method is most effective when the underlying MPNN exposes structural distinctions in the messages supplied to cross-attention. For brevity and simplicity, we focus our discussion on the graph-level tasks, but the same logic applies for the node-level tasks.

3.1 Intuitive Look at the GINAT Process

The Training Procedure, Revisited. During training, GINAT is asked to infer on the input graph, given additional information on the problem domain. This information is provided as final-layer embedding an [L]LM generated for a prompt such as those listed in Figure 2. Specifically, no information is given regarding the inference task at hand, the specific features or labels, or the dataset used. Since we use a dataset of different prompts wording (yet similar in nature), this allows the model to learn a prior about the distribution of the embeddings generated by the LM. This can also be seen as a form of distillation.

The LM's Generated Embedding. Recall that we use the final-layer output logits for the input prompt as the embedding that is cross-attended with the GNN signal. These embeddings are not an arbitrary transformation of the input tokens, but a function of the prompt and the prior learned by the LM throughout its training. The literature on LM shows that this prior encodes the substantial knowledge found in the training data Shin et al. (2020); Dai et al. (2022), which is abundant. This allows the output logits to encode information that, although not explicitly provided in the prompt, helps GINAT to perform better on the same tasks than the parameter-matched vanilla-GNN counterparts.

At Inference Time. We reiterate that our evaluation procedure uses a different set of prompts than the set used in the training time. While orthogonal to the downstream task, usage of unseen prompts demonstrates that the model successfully learns the encoding of relevant information and uses it.

This is evident in Figure 3. A trained GINAT model is provided with unseen prompts and graphs, and the attention maps show meaningful extraction of information. Attention maps (a) and (b) demonstrate the same prompt paired to two different graphs (of the same

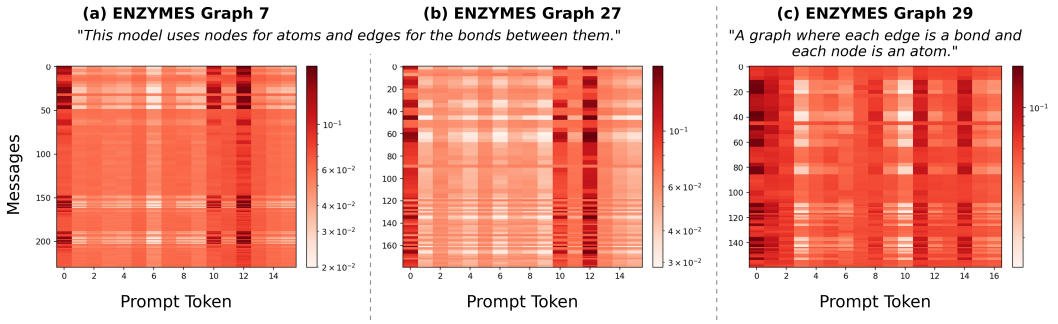

Figure 3: Example heatmaps of first layer attention maps between the MPNN signal and prompt embeddings. In these examples, GINAT is trained on ENZYMES dataset using GIN as the backbone MPNN and `roberta-large` as prompt encoder.

dataset). The consistent patterns on the token embeddings (columns) show that the model recognizes the information encoded in them, while the varying patterns across messages (rows) demonstrate that the added signal is not a constant modulation of the GNN output.

3.2 Success and Failure Modes

In our experiments, we evaluate GINAT against seven different, parameter-matched MPNN backbones on over 20 datasets. This extensive suite, documented in Tables 2, 3, 10, and 11, allows us to identify under what circumstances, settings, and configurations the method fails, where is succeeds—thus *how and why* it does so. In Appendix G we analyze the interesting case where GraphSAGE Hamilton et al. (2017) fails to utilize GINAT on featureless datasets.

Limited marginal gains for GraphConv. GraphConv Morris et al. (2019) presents a related but distinct phenomenon. Although GINAT improves its reported point estimate on nearly every benchmark, GraphConv obtains the smallest average relative gain among the evaluated backbones, with particularly small changes on molecular datasets, the Reddit benchmarks, and several node-classification datasets. Its prompt-augmented update is

$$h_v^{(\ell)} = \phi\left(\mathbf{W}_{\text{root}}^{(\ell)} h_v^{(\ell-1)} + \mathbf{W}_{\text{nbr}}^{(\ell)} \sum_{u \in \mathcal{N}(v)} \left[h_u^{(\ell-1)} + \xi^{(\ell)} \mathcal{C}^{(\ell)}\left(h_u^{(\ell-1)}, p\right)\right]\right). \tag{3}$$

When GINAT is applied to GraphConv, the cross-attention query is the untransformed source representation h_u and, in the unweighted setting, is independent of the receiver v. All outgoing messages from the same source node consequently receive the same prompt-conditioned correction. With constant features, the prompt initially supplies only a degree-scaled global vector, which is largely redundant with the degree information already captured by vanilla GraphConv. Thus, GINAT has limited marginal information to add to an already effective degree-sensitive representation. GraphConv's larger gains on datasets such as Enzymes, and the Airports benchmarks further indicate that it can utilize the contextual signal when the input messages contain sufficient variation and the task leaves greater room for improvement.

How graph-attention backbones utilize GINAT. GAT and GATv2 exhibit the largest improvements from GINAT. This behavior can be related to the position at which GINAT augments their computation. Suppressing the attention-head index, both backbones construct an attention-weighted message before applying cross-attention:

$$\alpha_{vu}^{(\ell)} = \underset{u \in \mathcal{N}(v)}{\text{softmax}}\left(e_{vu}^{(\ell)}\right) \qquad \widetilde{m}_{vu}^{(\ell)} = \alpha_{vu}^{(\ell)} \mathbf{W}^{(\ell)} h_u^{(\ell-1)} \qquad m_{vu}^{(\ell)} = \widetilde{m}_{vu}^{(\ell)} + \xi^{(\ell)} \mathcal{C}^{(\ell)}\left(\widetilde{m}_{vu}^{(\ell)}, p\right) \tag{4}$$

Thus, the graph-attention coefficient affects not only the contribution of the original message, but also the query through which the message attends to the prompt tokens. Once the coefficients differ across edges, structurally distinct messages can attend differently to the same embedded prompt and retrieve different contextual signals.

GraphConv versus graph-attention backbones. The substantially smaller gains of GraphConv can be explained by the weaker structural information contained in the message supplied to GINAT. Its cross-attention query contains neither the target-node representation nor a learned coefficient describing the importance of the $u \to v$ relation. This distinction becomes particularly pronounced on featureless benchmarks. If all initial node representations equal $\mathbf{c}$, GraphConv repeats the same prompt-conditioned message over the d_v incoming edges. By contrast, under uniform GAT attention, $\alpha_{vu}^{(1)} = d_v^{-1}$, the degree-dependent attention coefficient is incorporated into the query before cross-attention:

$$\begin{aligned} \mathbf{a}_{v,\text{GraphConv}}^{(1)} &= d_v \left[\mathbf{c} + \xi^{(1)} \mathcal{C}^{(1)} (\mathbf{c}, p) \right], \\ \mathbf{a}_{v,\text{GAT}}^{(1)} &= \mathbf{W}^{(1)} \mathbf{c} + \xi^{(1)} d_v \mathcal{C}^{(1)} \left(d_v^{-1} \mathbf{W}^{(1)} \mathbf{c}, p \right). \end{aligned} \tag{5}$$

For GraphConv, the additional signal follows the same multiplicity already captured by sum aggregation. For GAT, although attention normalization removes degree from the vanilla aggregate, degree remains encoded in the query supplied to cross-attention. Since cross-attention is nonlinear in this query, GINAT can transform an otherwise canceled structural signal into a degree-dependent contextual representation. The comparison therefore clarifies the contribution of our method to graph inference: its benefit does not arise merely from adding an LM representation, but from conditioning that representation on the structural message produced by the underlying GNN. GAT and GATv2 expose relation-dependent information before cross-attention, enabling an interaction between topology and linguistic priors, whereas GraphConv exposes primarily source-node information and consequently obtains smaller marginal gains.

4 Experiments

We empirically evaluate our method and examine its components and properties. Our experiments show that GINAT *consistently improves message passing GNNs* across graph- and node-level benchmarks.

The experimental setup and settings are detailed in Appendix B. The configurations used to obtain the reported results are listed in Appendix M.

Beyond Baseline Comparison. We experimented heavily with the different components in GINAT, seeking answers for the following questions: **(a)** Is the information in the prompts actually contributes, or is it perhaps the additional computational capacity or better gradient flow? **(b)** What is the best setting of ξ, and what are its contributions? **(c)** What alternatives for cross-attention for graph-language fusion are there, and how do TAG methods fare in our settings?

Due to lack of space, we present these experiments in Appendix F. Results of experiments designed to answer **(a)** and **(b)** are presented in Figures 4, 5, respectively.

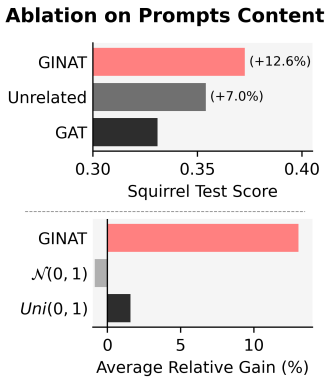

Figure 4: Comparing GINAT under different prompt-embedding sources. In both plots, the red bar represents GINAT with the benchmark's proper prompt dataset. Top: Squirrel results using proper vs. unrelated prompts, compared to MPNN baselines. Bottom: average relative gain across six benchmarks using proper prompts vs. Gaussian/uniform random vectors (See Appendix F.6).

4.1 Graph- and Node-Level Performance

Tables 2 and 3 report the primary evaluation of GINAT across diverse graph- and node-level benchmarks using standard MPNN backbones under matched training protocols and model-selection by validation. Across domains and metrics, GINAT consistently improves the underlying backbone. We present additional results in Appendix H.

A notable and interesting case is that of the attention-based networks on the featureless datasets - REDDIT-* and IMDB-*, as listed in Tables 2, 10. In these cases, the baseline models fail to learn and collapse to random guess - a known phenomenon on attentive

MPNNs Zhang & Xie (2020) - while GINAT improves significantly. These findings suggest that GINAT aids MPNNs on data they struggle with. A similar phenomenon occurring for GraphSAGE is analyzed in depth in Appendix G.

For node classification (Table 3), GINAT similarly improves performance in a transductive setting across graphs with varying feature dimensionality and homophily. While these datasets are usually evaluated under a full-graph regimen, where the entire graph is processed in one forward pass, due to the memory overhead incurred by the cross-attention mechanism, we had to settle for subgraph sampling using GraphSAINT Zeng et al. (2020). Baseline, vanilla-GNN models were trained using the same protocol for fair comparison. The reported results are aggregated across the subgraphs, per the convention in the literature under these circumstances. A consequence of this compromise is that performance on some of the benchmarks shown in the table is relatively low to the results usually reported in the literature. To complement these results, we performed experiments on small node-level benchmarks. See Appendix F.8 for results and discussion.

We observe clear improvements especially on heterophilic benchmarks such as Chameleon, suggesting that prompt-augmented message passing can complement or partially mitigate the limitations of standard neighborhood aggregation in challenging regimes. Overall, these results establish GINAT as a general plug-in framework to message passing architectures across task levels, while subsequent sections isolate the source of the gains (prompt content and LM priors) and analyze how the model uses provided context via the learned scaler $\xi^{(l)}$. Full experimental details (including prompt usage protocol, handling of featureless graphs, sampler settings) appear in Appendix F.2.

5 Discussion

We introduced a novel graph learning paradigm, *graph-language augmented models*, which augments graph data with external textual context. We instantiate this paradigm with GINAT, a framework that enhances message passing by enriching edge messages using LM-embedded prompts through cross-attention while preserving permutation equivariance. Empirically, it improves MPNN baselines by up to **23%** (relative) across graph- and node-level benchmarks, with particularly strong gains in settings where standard MPNNs struggle, such as heterophilic and featureless benchmarks.

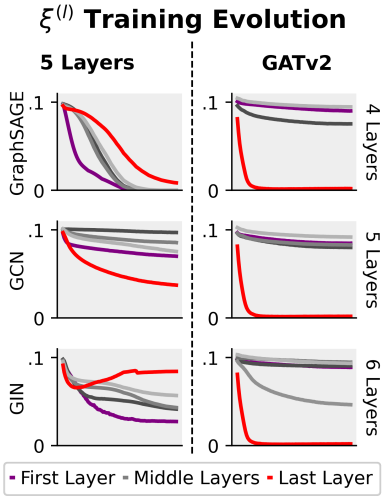

Figure 5: Training evolution of $\xi^{(l)}$ on REDDIT-MULTI-12K for MPNNs with GINAT using `DeBERTa-V3-base` (initialized at 0.1). Left: 5-layer models across backbones. Right: GATv2 with 4/5/6 layers. See App. F.4.

Our ablations indicate that the gains come from meaningful language-derived context rather than added capacity or noise: semantically relevant prompts consistently outperform random vectors, and different prompt encoders yield similar trends, suggesting that prompt alignment is the key driver. Finally, our interpretability experiments reveal *when* and *where* the model uses language-conditioned information: the layer-wise scaling profiles expose benchmark- and backbone-dependent reliance on contextual attention, offering an actionable diagnostic for understanding the mechanism behind the improvements.

Limitations. GINAT incurs non-negligible computational overhead. Specifically, this makes training and evaluation of large node-level transductive benchmarks unfeasible on our hardware, and our compromise on GraphSAINT subgraph sampling based approach, while not unique, is a deviation from the standard in GNN literature. Additionally, our experiments use small, synthetic prompt datasets designed to evaluate the proposed paradigm in a controlled setting. A systematic investigation of how prompt quality, diversity, scale, and construction strategy affect performance remains an important direction for future work.

Ethics Statements

This work does not involve human subjects, personal or sensitive data, or applications in high-risk domains, to the best of our knowledge. All datasets used in our experiments are publicly available benchmark graph datasets (e.g., OGB) and were used in compliance with their respective licenses. The large language models employed in our study are publicly released open-source models obtained through HuggingFace. Our method is designed as a general framework for improving graph learning and does not target or promote harmful applications. We acknowledge that, as with any machine learning system, fairness and bias issues may arise if the approach is applied to sensitive data; however, such applications are beyond the scope of this paper.

Reproducibility Statement

We have taken care to ensure the reproducibility of our results. Complete details of datasets, model architectures, training settings, and hyperparameters are provided in the main text and Appendix. The prompt datasets used are documented in Appendix K. All baseline implementations follow the official PyTorch Geometric library (Fey & Lenssen, 2019), and pretrained LLMs are obtained from HuggingFace repositories. All experiments used $630, 1312, 15052010$ as seeds. Code for GINAT, including training scripts and evaluation pipelines, is available in the anonymised git repository `https://anonymous.4open.science/r/GINAT-1312/`

AI Use Statement

In this work, we used generative AI tools for / to:

- Generation of synthetic data sets (as discussed in length in Appendix B).
- Assistance in the writing of proofs.
- Implement methods.
- Support qualitative and thematic data analysis.

We have not used generative AI tools for / to:

- Help develop theoretical models or conceptual frameworks.
- Formulate mathematical claims.
- Provide critical ingredients for proving mathematical claims.
- Propose or refine hypotheses.
- Design or provide feedback on research methodology or experiments.
- Assist with translation.
- Clean and reformat dataset.
- Interpret results.

Additionally, we used generative AI tools for / to:

- Create or edit software code.
- Summarize or analyse existing literature, and identify relevant literature.
- Draft parts of a research paper.
- Edit a research paper to improve readability.
- Format references.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested rigorously for validity and correctness. All written LLM-generated text appearing in the submitted manuscript was reviewed and approved by the authors. All suggested literature was reviewed and cross-referenced by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2112–2118, 2021. doi: 10.24963/ijcai.2021/291. URL https://www.ijcai.org/proceedings/2021/291.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, et al. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. doi: 10.52202/068431-1723. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract.html.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Haibo Chen, Xin Wang, Zeyang Zhang, Haoyang Li, Weigao Wen, Ling Feng, and Wenwu Zhu. Curriculum gnn-llm alignment for text-attributed graphs. In *OpenReview preprint*, 2025.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms) in learning on graphs. *SIGKDD Explor. Newsl.*, pp. 42–61, March 2024.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581/.

Keyu Duan, Qian Liu, Tat seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. Simteg: A frustratingly simple approach improves textual graph learning. *ArXiv*, abs/2308.02565, 2023.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Moshe Eliasof, Fabrizio Frasca, Beatrice Bevilacqua, Eran Treister, Gal Chechik, and Haggai Maron. Graph Positional Encoding via Random Feature Propagation. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IuXR1CCrSi.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/abs/1903.02428.

Dedre Gentner and Jennifer Asmuth. Metaphoric extension, relational categories, and abstraction. *Language, Cognition and Neuroscience*, 34(10):1298–1307, 2019. doi: 10.1080/23273798.2017.1410560. URL https://doi.org/10.1080/23273798.2017.1410560.

Yaowen Gu, Zidu Xu, and Carl Yang. Empowering graph neural network-based computational drug repositioning with large language model-inferred knowledge representation. *Interdisciplinary Sciences: Computational Life Sciences*, 17(3):698–715, 2025. doi: 10.1007/s12539-024-00654-7.

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

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Vincent J. Hellendoorn, Petros Maniatis, Rishabh Singh, Charles Sutton, and David Bieber. Global relational models of source code. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1lnbRNtwr.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

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

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020b. URL https://openreview.net/forum?id=HJlxPhHFvB.

Xin Huang, Weipeng Zhuo, Minh Phu Vuong, Shiju Li, Jongryool Kim, Bradley Rees, and Chul-Ho Lee. SDT-GNN: Streaming-based distributed training framework for graph neural networks. In *2025 IEEE International Conference on Big Data (BigData)*, 2025. URL https://ieeexplore.ieee.org/document/11401821.

John Jumper, Richard Evans, Alexander Pritzel, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Kepa Korta and John Perry. Pragmatics. In Edward N. Zalta and Uri Nodelman (eds.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2024 edition, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: Progress and future directions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8123–8131, 2024a. doi: 10.24963/ijcai.2024/898. URL https://www.ijcai.org/proceedings/2024/898.

Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor Wai Kin Chan, and Jia Li. Glbench: A comprehensive benchmark for graph with large language models. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16682–16699, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.856. URL https://aclanthology.org/2025.findings-acl.856/.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: higher-order graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020. URL https://www.graphlearning.io.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021. doi: 10.1093/comnet/cnab014. URL https://doi.org/10.1093/comnet/cnab014.

Yingxia Shao, Hongzheng Li, Xizhi Gu, Hongbo Yin, Yawen Li, Xupeng Miao, Wentao Zhang, Bin Cui, and Lei Chen. Distributed graph neural network training: A survey. *arXiv preprint arXiv:2211.00216*, 2022. URL https://arxiv.org/abs/2211.00216.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL https://aclanthology.org/2020.emnlp-main.346/.

Haoran Song, Jiarui Feng, Guangfu Li, Michael Province, Philip Payne, Yixin Chen, and Fuhai Li. Large language models meet graph neural networks for text-numeric graph reasoning. *arXiv preprint arXiv:2501.16361*, 2025.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. Paths-over-graph: Knowledge graph empowered large language model reasoning. *arXiv preprint arXiv:2410.14211*, 2024.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V. Chawla, and Panpan Xu. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

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

Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. Llms as zero-shot graph learners: Alignment of gnn representations with llm token embeddings. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Runze Wang, Mingqi Yang, and Yanming Shen. Bridging molecular graphs and large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(20): 21234–21242, 2025. doi: 10.1609/aaai.v39i20.35422. URL https://ojs.aaai.org/index.php/AAAI/article/view/35422.

Jamie Williams, Noah Russell, and Derek Irwin. On the notion of abstraction in systemic functional linguistics. *Functional Linguistics*, 4(1):13, 2017.

Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. When do LLMs help with node classification? a comprehensive analysis. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 67623–67649. PMLR, 2025. URL https://proceedings.mlr.press/v267/wu25y.html.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2024. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

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

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0755. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/37d00f567a18b478065f1a91b95622a0-Abstract-Datasets_and_Benchmarks.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. LinkBERT: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8003–8016, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.551. URL https://aclanthology.org/2022.acl-long.551/.

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? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Yuxin You, Zhen Liu, Xiangchao Wen, Yongtao Zhang, and Wei Ai. Large language models meet graph neural networks: A perspective of graph mining. *arXiv preprint arXiv:2412.19211*, 2024.

Jianxiang Yu, Yuxiang Ren, Chenghua Gong, Jiaqi Tan, Xiang Li, and Xuecang Zhang. Leveraging large language models for node generation in few-shot learning on text-attributed graphs. In *AAAI Conference on Artificial Intelligence*, 2025.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwB.

Jiasheng Zhang, Jialin Chen, Menglin Yang, Aosong Feng, Shuang Liang, Jie Shao, and Rex Ying. Dtgb: A comprehensive benchmark for dynamic text-attributed graphs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Shuo Zhang and Lei Xie. Improving attention mechanism in graph neural networks via cardinality preservation. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 1395–1402. International Joint Conferences on Artificial Intelligence Organization, July 2020. doi: 10.24963/ijcai.2020/194. URL https://doi.org/10.24963/ijcai.2020/194. Main track.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=q0nmYciuuZN.

Xiangru Zhu, Zhixu Li, Xiaodan Wang, Xueyao Jiang, Penglei Sun, Xuwu Wang, Yanghua Xiao, and Nicholas Jing Yuan. Multi-modal knowledge graph construction and application: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(2):715–735, 2024. doi: 10.1109/TKDE.2022.3224228. URL https://ieeexplore.ieee.org/document/9961954.

A Prompts Dataset Generation

Algorithm 1 Prompts Dataset Generation

Input:
Example prompts $\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\}$
Models $\mathcal{M} = \{m_1, \ldots, m_K\}$
Desired rephrasings per example N
Validity predicate $\textsc{Valid}(\cdot)$
Cleaning function $\textsc{Clean}(\cdot)$

Output:
Generated dataset $\mathcal{G}$ of rephrased prompts

Procedure:
$\mathcal{G} \leftarrow \emptyset$
for each $p \in \mathcal{P}$ **do**
  added $\leftarrow 0$
  while added $< N$ **do**
    $m \leftarrow \textsc{Next}(\mathcal{M})$ (cycle models)
    $y \leftarrow \textsc{Generate}(m, p)$ (request rephrasing)
    $r \leftarrow \textsc{Clean}(y)$ (extract single candidate)
    if $\textsc{Valid}(r, p)$ **and** $r \notin \mathcal{G}$ **then**
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{r\}$
      added $\leftarrow$ added $+ 1$
    end if
  end while
end for
return $\mathcal{G}$

Curating a diverse collection of hundreds of prompts describing a single dataset (or domain) by hand is a prohibitively time-consuming task, especially across the ten distinct domains explored in this work. Since our primary focus is not on the linguistic nuances of NLP but rather on GNNs, we generated the prompt dataset synthetically using open-source LLMs.

For each dataset, we manually wrote 5–10 "seed" descriptions of the graphs or nodes. We then employed LLMs—specifically Qwen2.5 (Qwen et al., 2024) and SmolLM3 (Bakouch et al., 2025)—to generate new descriptions by rephrasing these examples. To maximize phrasing diversity and maintain experimental integrity, we intentionally used a separate suite of LLMs for data generation, ensuring the evaluation framework remained strictly decoupled from the main experimental pipeline. While the downstream tasks are independent, this separation provides additional methodological rigor against potential leakage.

The Generation Process The high-level procedure for expanding our prompt corpus is formalized in Algorithm 1. This process began with a curated set of instruction prompts—listed in full below—designed to guide the LLMs in rephrasing the manual seeds while strictly adhering to our desired word counts. We tasked the models with iteratively rephrasing these seeds until the target number of unique descriptions was met for each domain.

To foster significant linguistic variety and prevent the models from converging on repetitive syntactic structures, we drew the generation temperature uniformly from the range $[0.7, 1.3]$ for each request. Furthermore, we employed the LLMs in a round-robin fashion, alternating between them to ensure that the resulting dataset was not skewed by the stylistic biases or unique inductive biases of a single architecture.

Following each generation, the outputs underwent a multi-stage post-processing pipeline to ensure data integrity. We first stripped the responses of any meta-discourse or conversational filler (e.g., `"Certainly! Here is a rephrased version:"`) that frequently occurs in high-

temperature settings. Each candidate was then validated against the specified length constraints and cross-referenced with the existing corpus to eliminate exact string duplicates. While we did not implement a fully automated formal verification of the semantic content, we conducted manual surveys across each domain to confirm that the generated prompts remained coherent and contextually relevant. This pipeline ultimately produced hundreds of unique prompts for every experimental domain. Although we do not argue that these synthetic datasets are optimal, they served as a robust and scalable foundation for our experiments. To facilitate reproducibility and further research, we have released the complete collection of generated prompts as part of our codebase.

Prompts Used to Instruct LLMs The specific instruction templates used to guide the LLMs in rephrasing the seed descriptions are enumerated below:

- `"Rephrase the following text in a different way while preserving its meaning: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Rewrite this prompt using different words and structure: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Express the following idea in an alternative way: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Transform this prompt while keeping the same intent: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Provide a different formulation of this prompt: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Rearrange and reword this text: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Give me a variant of the following prompt: "[PROMPT]" Keep it around [LENGTH] words."`
- `"Paraphrase this text: "[PROMPT]" Keep it around [LENGTH] words."`

B Experimental Setup and Settings

In our experiments, we apply GINAT to widely used MPNN baselines and compare their performance on many benchmarks. Specifically, we experiment with GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GATv2 (Brody et al., 2022), GraphConv (Morris et al., 2019), GraphSAGE (Hamilton et al., 2017), and GIN (Xu et al., 2019) or GINE (Hu et al., 2020b) when edge features were available. We also apply GINAT to the message passing-based graph transformer GraphGPS (Rampášek et al., 2022). For each benchmark and MPNN pair, we select the best of four depths and two widths by validation, and report mean test score over three fixed-seed runs. This evaluation protocol varies the prompt paired with each graph sample across runs, reducing the risk that performance is driven by spurious prompt-graph alignments. For benchmarks that provide a standard split, such as OGBG-MolHIV Hu et al. (2020a) or Chameleon Rozemberczki et al. (2021), we follow them. Otherwise, we generate a seeded train/validation/test split for each of the three runs and report the mean test score across them. The hyperparameters used to train GINAT and baselines for each benchmark are listed in Appendix N. Pretrained LMs were obtained from HuggingFace, and we use PyTorch Geometric implementations of all backbones (Fey & Lenssen, 2019).

Baselines' Settings and Results. We follow the benchmarking approach explicitly advocated in the landmark Dwivedi et al. (2023). We use controlled, budget-matched comparisons over chasing state-of-the-art numbers, since the latter conflates method quality with tuning effort. Since GINAT uses cross-attention over LLM embeddings with dimensionality on the order of 10^3, its layers are larger than those of standard GNN baselines. To control for capacity, we match each baseline to GINAT's parameter budget by keeping the same depth and adjusting the hidden dimension to closely match the parameter count. Our results are reproducible using the code accompanying this work.

Prompt Datasets. To experiment and establish the validity of our proposed method, we curate a small prompt dataset per domain using a few manually written examples and an LLM-based rephrasing pipeline. For completeness, we elaborate on the toy prompt dataset generation procedure in Appendix A. Overall, our toy prompt datasets contain up to 300 prompts, averaging ~ 15 words each (see Appendix K for more details). For reproducibility and future research, the complete toy datasets can be found in the codebase accompanying this work. In each training and evaluation run, we split the prompts dataset randomly in an 8-1-1 ratio to prevent memorization of prompt embeddings. Figure 2 shows example prompts; they describe graphs but never mention labels or objectives. In Appendix C, we ablate on several flavors of toy prompts dataset and show that the chosen experimental settings and datasets curation process are robust.

Assignment of Prompts. Depending on the learning task at hand, we assign the prompts differently. For graph-level tasks, on each forward pass, each graph sample is paired with a single prompt drawn from the prompts dataset. To ensure a single graph attends to a single prompt, we construct an attention mask that blocks any interaction between a graph and prompts assigned to other graphs, ensuring its messages attend only to the prompts paired with its graph.

C Ablating Prompt Dataset Construction

We further evaluate whether GINAT is sensitive to the particular construction of the prompt dataset. To this end, we compare several prompt-dataset variants on two representative benchmarks: NCI1 for graph classification and PubMed for node classification. For NCI1, we use a GAT backbone, while for PubMed we use a GATv2 backbone. In both cases, we evaluate two prompt encoders, `Qwen3-4B-Base` and `RoBERTa-large`, and six architecture configurations corresponding to three depths and two hidden widths. Each configuration is trained with three random seeds, yielding 90 runs per benchmark. Table 4 reports the mean and standard deviation across these runs.

We consider five prompt-dataset variants. *Test = Train* reuses the training prompts at evaluation time. *Single Source* constructs all prompts from a single manually written template. *Noisy* uses higher-temperature generations, producing lower-quality and less consistent prompt variations. *Small Set* restricts the prompt pool to roughly 50 prompts. Finally, *Large Set* increases the prompt pool to roughly 500 prompts.

Table 4: Ablation of prompt dataset construction on representative graph- and node-level datasets. Results are averaged over two embedding LMs, six architecture configurations, and three seeds.

Prompt Setting	NCI1 Acc. ↑	PubMed Acc. ↑
Vanilla MPNN	0.731 ± 0.047	0.630 ± 0.090
GINAT	$\mathbf{0.776 \pm 0.019}$	$\mathbf{0.644 \pm 0.107}$
Test=Train	0.742 ± 0.044	0.629 ± 0.107
Single Source	0.735 ± 0.039	0.613 ± 0.104
Noisy	0.725 ± 0.055	0.627 ± 0.101
Small Set (~ 50)	0.734 ± 0.049	0.643 ± 0.115
Large Set (~ 500)	0.735 ± 0.037	0.631 ± 0.099

C.1 Analysis and Discussion.

The results indicate that the gains obtained by GINAT are robust to variations in the prompt construction pipeline, but are strongest when using the curated prompt dataset. On NCI1, the full prompt setting improves over the matched vanilla MPNN by 4.5 absolute accuracy points, while all alternative prompt variants underperform the full setting. This suggests that the benefit does not arise merely from adding prompt-conditioned parameters or from repeatedly exposing the model to arbitrary text embeddings. Rather, the alignment and diversity of the prompt set contribute to the final performance.

The ablations show that both prompt quality and diversity are important. The *Noisy* and *Single Source* variants perform close to or below the vanilla baseline, indicating that low-quality prompts or limited linguistic variation weaken the prompt signal. The *Small Set* and *Large Set* variants further show that prompt quantity alone is insufficient: a compact prompt set can be competitive on PubMed, while a larger pool does not consistently improve over the curated setting. Finally, *Test=Train* does not improve performance, establishing that the gains are not due to prompt reuse or information leakage.

overall, these results further support the conclusion that GINAT rely on semantic information encoded in LM-embedded domain prompts rather than on the structural properties of a particular prompt set.

D Preservation of Equivariance

GNNs are inherently permutation equivariant: reordering the nodes in the input graph leads to the same reordering in the output node embeddings. Analogously, multi-modal cross-attention is also permutation equivariant with respect to the queries. In this section, we prove that our method maintains permutation equivariance - a key characteristic in any GNN.

Proposition D.1. *Let $Q \in \mathbb{R}^{n_q \times d}$ denote the query matrix, $K \in \mathbb{R}^{n_k \times d}$ the key matrix, and $V \in \mathbb{R}^{n_k \times d_v}$ the value matrix. The cross-attention operator is defined as*

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V,$$

where the softmax is applied row-wise. Then, for any permutation matrix $P \in \{0,1\}^{n_q \times n_q}$, it holds that

$$\mathrm{Attn}(PQ, K, V) = P\,\mathrm{Attn}(Q, K, V).$$

That is, cross-attention is permutation equivariant with respect to its queries.

Proof. Consider the attention with permuted queries:

$$\mathrm{Attn}(PQ, K, V) = \mathrm{softmax}\left(\frac{(PQ)K^\top}{\sqrt{d}}\right) V.$$

By associativity of matrix multiplication,

$$(PQ)K^\top = P(QK^\top).$$

Since the softmax is applied row-wise, permuting the rows before softmax is equivalent to permuting them after softmax:

$$\mathrm{softmax}\big(P(QK^\top)\big) = P\,\mathrm{softmax}(QK^\top).$$

Therefore,

$$\mathrm{Attn}(PQ, K, V) = \big(P\,\mathrm{softmax}(QK^\top)\big)V = P\,\big(\mathrm{softmax}(QK^\top)V\big).$$

This simplifies to

$$\mathrm{Attn}(PQ, K, V) = P\,\mathrm{Attn}(Q, K, V),$$

proving permutation equivariance with respect to queries. □

We now prove Theorem 2.1. For convenience, we restated it here.

Theorem (GINAT Preserves Permutation Equivariance)**.** *Let* update, aggregate *be the update and aggregation methods of a permutation-equivariant MPNN layer (denote* aggregate *as* AGG *for brevity). Let $h_v^{(l)}$ be the representation of node v in the input to the current layer. Then for messages as defined in Eq. 1, the operation*

$$\mathrm{update}\left(h_v^{(l)}, AGG\left(\left\{m_{uv}^{(l)} : u \in \mathcal{N}(v)\right\}\right)\right) \tag{6}$$

remains permutation-equivariant.

Proof. Let P be a permutation matrix applied to reorder the nodes in the input graph. Permutation equivariance requires that

$$f(PX, PAP^\top) = Pf(X, A), \tag{7}$$

where X is the node feature matrix and A the adjacency matrix.

Since the MPNN layer is permutation-equivariant, we can deduce that the message is permutation-equivariant, i.e.

$$P\,\mathrm{MSG}\,(X, A) = \mathrm{MSG}\,\left(PX, PAP^T\right) \tag{8}$$

Let p be some input to the cross-attention to be used as keys and values. In our case, these would be LM-generated prompt embeddings. By Proposition D.1 we have

$$P\,\text{CrossAttention}\left(\text{MSG}\left(X, A\right), p, p\right) = \text{CrossAttention}\left(\text{MSG}\left(PX, PAP^T\right), p, p\right) \quad (9)$$

Thus by linearity, the cross-attention-augmented message defined by GINAT in Equation 1 holds

$$\begin{aligned} &P\left(\text{MSG}\left(X, A\right) + \text{CrossAttention}\left(\text{MSG}\left(X, A\right), p, p\right)\right) = \\ &\text{MSG}\left(PX, PAP^T\right) + \text{CrossAttention}\left(\text{MSG}\left(PX, PAP^T\right), p, p\right) \end{aligned} \quad (10)$$

Meaning that the message function defined by Equation 1 preserves the permutation equivariance property. Thus, the assumptions aggregate $(\cdot)$ and update $(\cdot)$ make on the message$(\cdot)$ hold, and the permutation equivariance property is preserved. Note that the ξ scaler is dropped from this derivation for clarity and brevity.

□

E Related Work

E.1 Text-Attributed Graphs

Text-attributed graphs (TAGs) combine graph structure with textual node or edge attributes and appear in many domains, from citation networks to e-commerce. Several recent works advance TAG research. Yan et al. (2023) introduce a comprehensive benchmark with new datasets, tasks, and systematic evaluations; Zhao et al. (2023) propose GLEM, a scalable framework for learning on large TAGs using variational inference and message passing; and Zhang et al. (2024) present the first benchmark for dynamic TAGs, curating large-scale, time-evolving graphs and defining four tasks. Other work leverages LLMs to enrich TAG representations: He et al. (2024), for example, develop an LLM-to-LM interpreter that distills explanations from LLMs into smaller language models.

While this progress inspired our work, our setting differs fundamentally. TAG methods assume that text attributes are part of the dataset, whereas GINAT uses external, task-agnostic prompts that describe the semantics of graphs and their entities. These prompts are neither node nor edge attributes, nor are they sources of label- or objective-specific information.

E.2 LLMs as Enhancers for GNNs

LLMs offer rich semantic priors that can complement the structural focus of GNNs, making them especially relevant to foundation-style graph models. Prior work employs LLMs as *knowledge injectors*, filling gaps in incomplete graphs and improving link prediction by aligning textual and structural representations (Li et al., 2024a; Chen et al., 2025; Tian et al., 2024); as sources of *self-supervised signals*, generating semantic objectives that enrich pretraining beyond purely structural tasks (You et al., 2024; Li et al., 2024a); and as *interpretable enhancers*, producing natural-language rationales that improve transparency and support human-in-the-loop reasoning (Gu et al., 2025).

Unlike most LLM–GNN hybrids, GINAT does not use an LLM to impute missing edges or attributes, generate auxiliary supervision, or produce post-hoc explanations. Instead, it treats the LM as a frozen prompt embedder and uses its token-level representations as an external conditioning signal that directly augments message passing through cross-attention on edge messages.

E.3 GNNs as Enhancers for LLMs

Conversely, recent work uses graph structure to enrich language models with explicit relational reasoning. One line of research augments language-model pretraining or inference with relational information. LinkBERT leverages inter-document links during pretraining (Yasunaga et al., 2022), while knowledge-graph retrieval and encoding methods use graph-based entity representations to improve factual question answering and entity-centric reasoning (Mavromatis & Karypis, 2025).

A second line investigates hybrid graph–language architectures. Molecular graph encoders combined with LLMs enhance drug-discovery tasks (Wang et al., 2025), while abstract-syntax-tree encodings integrated with Transformers improve code reasoning (Hellendoorn et al., 2020). Graph-guided prompting offers another form of integration: GNNs extract relational substructures, such as multi-hop paths or neighborhoods, and provide them to LLMs as structured prompts, enabling multi-hop reasoning without modifying the language-model architecture (Chen et al., 2024).

These approaches face several challenges. Scaling graph processing incurs substantial communication and memory costs (Shao et al., 2022; Huang et al., 2025); semantic alignment between graph and token embeddings remains non-trivial (Wang et al., 2024; Li et al., 2024a); and evaluation protocols are still developing, with benchmarks such as GLBench and LLMNodeBed beginning to standardize comparisons (Li et al., 2024b; Wu et al., 2025).

E.4 Cross-Attention for Graph Learning

Cross-attention provides a natural interface for joint reasoning over graph structure and linguistic priors by allowing graph representations to attend selectively to relevant language signals. It has demonstrated strong performance in multimodal knowledge-graph completion under sparsity (Zhu et al., 2024), scientific discovery and graph reasoning through integration with large language models (Song et al., 2025), and multi-hop reasoning through dynamic alignment between language queries and graph substructures (Tan et al., 2024).

Despite these successes, challenges remain in balancing modality dominance, scaling attention to large graphs, and aligning the corresponding embedding spaces. Moreover, the contribution of the language signal is often difficult to isolate: performance gains may reflect implicit priors, architectural bias, or increased capacity rather than meaningful use of the input text. Our design injects prompt-conditioned context directly at the message level and enables a more controlled assessment of how language priors affect message passing through analyses of learned scaling and cross-attention patterns.

F Additional Experiments

F.1 Graph-Level Tasks

Table 2 shows the test scores of MPNNs with and without GINAT on several graph benchmarks, spanning molecular and social network data. These benchmarks vary in properties such as the presence of node features, graph size, and density.

In those experiments, we include benchmarks that lack node features, such as the REDDIT-* benchmarks. To facilitate fair comparison with GINAT, in the baseline experiments we assigned them random node features. This choice follows the established literature exploring random node features Abboud et al. (2021); Eliasof et al. (2023). We use constant features in GINAT to isolate the contribution of prompt-augmented message passing, rather than introducing additional signal via arbitrary node attributes.

Usage of Embedded Prompts. For graph-level benchmarks, we pair each graph with a prompt, randomly sampled from the prompt dataset on each forward pass. Across epochs, each graph is paired with multiple prompts, encouraging the model to identify meaningful signals from the LM prior.

Findings. The results establish that GINAT provides significant gains on graph-level tasks. We also compare against strong GNN baselines beyond message passing: Graphormer Ying et al. (2021) and GraphMAE Hou et al. (2022). In many cases, GINAT applied to an MPNN outperforms these strong models, even when the underlying MPNN alone does not. We train them from scratch on the same splits, tune hyperparameters using recommended configurations, and select the best model by validation.

Gains are specifically evident on the REDDIT-* benchmarks; in some cases, GINAT more than doubles the baseline MPNN accuracy. As mentioned earlier, these graphs are featureless and thus messages are purely structural in content. These results demonstrate that the embedded prompts provide context to the graph, and allow the MPNN to learn about the topology more effectively.

F.2 Node-Level Tasks

Table 3 presents the test results of MPNNs with and without GINAT. The benchmarks include graph datasets with varying node feature dimensions, as well as heterophilic benchmarks. All node-level models were trained under a transductive regime, and nodes were sampled using the GraphSAINT Zeng et al. (2020) sampler. For each benchmark, we chose the `num_steps` (number of samplings per epoch, similar to the number of batches in inductive learning) and `batch_size` (number of sampled nodes per step) so that in expectation, all nodes will be sampled in each epoch. For hyperparameters, we refer to Appendix N.

F.3 GINAT Outperforms Baseline Approaches

In the previous subsection, GINAT shows that our paradigm is beneficial and can indeed be used to improve graph inference. While GINAT is rooted in theoretic guarantees and previous work, it is worthwhile to examine whether less complex approaches to combining the two signals can achieve similar performance. We experiment with three different approaches: (a) summing the prompt embedding with the MPNN message (following a learned aggregation and projection to the same space), (b) concatenating the prompt embedding to the message, and (c) summing outputs from sentence-embedding LMs (so that the embedding is of fixed-size).

In addition to these approaches, we also experiment with using TAG models, as the conceptual similarities cannot be overlooked. However, conceptual similarity is not enough, and TAG methods such as TAPE He et al. (2024) and LLM4NG Yu et al. (2025) achieve near-random performance under our paradigm. We present TAG methods results in Appendix F.9.

Experiment Setting. We evaluate the baseline methods on three datasets that cover multiple tasks and homogeneity properties: IMDB-BINARY, Computers, and Chameleon. In order to isolate the contribution of the signals-fusion method, each method-dataset pair is evaluated on three different MPNN backbones (GAT, GIN, GCN) and six depth-width settings. Each configuration is tested using three predefined seeds. Overall, each result in Table 5 aggregates 54 different training runs.

Results and Discussion. We present the results of the experiment in Table 5. GINAT, summation, and concatenation techniques used `roberta-large` prompt embeddings. For sentence embedding LMs, we experiment with `BGE-Base` and `-large`. GINAT shows the best and most consistent performance. While not consistent, the relatively close results the baseline approaches achieve further suggest that our external-information approach is promising, and multiple techniques are able to harness this information for graph inference. Another advantage of GINAT over the baseline alternatives is that it offers insights on *how* the model utilizes the information in the embedded prompt signal, as we discuss in the following subsection.

Table 5: Comparing GINAT with baseline techniques. Best results are in **bold** and the second-best are underlined. GINAT performs best and most consistently. BGE-* are sentence-embedding techniques. See Section F.3 for further discussion.

Method	IMDB BINARY	Computers	Chameleon
GINAT	**0.743±.05**	**0.859±.00**	0.459±.01
Sum	0.740±.04	0.855±.00	0.430±.00
Concat	0.670±.03	0.841±.01	**0.465±.03**
BGE-b	0.720±.04	0.849±.02	0.456±.04
BGE-l	**0.743±.06**	**0.859±.00**	0.440±.02

Usage of Embedded Prompts. For node-level benchmarks, we attend each sampled subgraph (one sampler step) to a single embedded prompt. Prompts describe what nodes represent in the graph, mirroring the graph-level setting in which prompts describe the target object. We refer to Appendix K for examples.

Findings. The results in Table 3 show that GINAT extends beyond graph-level tasks. As the analysis in Table 7 shows, the GINAT framework achieves a relative performance improvement of over 10%. Importantly, the heterophilic benchmarks Squirrel and Chameleon showed a significant improvement of up to 6 percentage points.

F.4 Exploring GINAT's Contribution

Having established that the performance gains made by GINAT stem from the ability to incorporate meaningful information from the prompt signal into the message passing process, we wish to understand how it does that. To this end, we examine its trainable additions to MPNNs.

Using ξ to Measure Contribution. In Section 2.2 we introduced the learned scaler $\xi^{(l)}$, which scales the cross-attention signal to the messages. Its main purpose of the scaler ξ is to provide some insight into the inner works for GINAT. More concretely, it allows us to quantify how much (and in what layers) the model uses the context provided by the embedded prompts and the LM prior.

In Figure 5, we present the convergence of $\xi^{(l)}$ when GINAT is trained on REDDIT-MULTI-12K. These results represent two phenomena we recognized in our experiments. First, different MPNNs make different use of the signal that originates from cross-attending the messages with the embedded prompts. For instance, as can be inferred from Table 2, GraphSAGE fails to utilize this signal, and the performance increase GINAT provides on the REDDIT-MULTI-12K benchmark is negligible. In Figure 5, it's evident that GraphSAGE learns $\xi^{(l)} = 0$ for all layers. This finding supports the same conclusion: GraphSAGE is unable to leverage the added context to improve learning. Further ablations and additional study of the $\xi^{(l)}$ training dynamics are provided in Appendix F.5. On the other hand, GCN

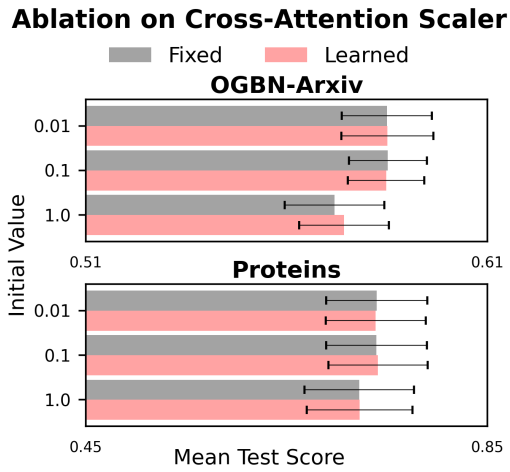

Figure 6: Ablation on using learned or fixed scaler for the cross-attention signal. Each bar represents the average of 96 configurations, spanning four different embedding LMs and three MPNN backbones (See Appendix F.5 for details).

and GIN both achieve better results on the benchmark when equipped with GINAT, and both converge to strictly non-zero $\xi^{(l)}$ values, albeit in different ways.

The second phenomenon demonstrated in Figure 5 is that the same $\xi^{(l)}$ patterns appear for different depth configurations. In the plots on the right of the figure, it is evident that GATv2 relies on the GINAT signal heavily in the early layers, but learns $\xi^{(L)} = 0$, thus not using it at all in the last layer. We note that GINAT more than doubles GATv2's accuracy on REDDIT-MULTI-12K (See Table 2).

F.5 Ablations on the Cross-Attention Residual

Ablation: Fixing vs. Learning $\xi^{(l)}$. From an expressivity standpoint, learning $\xi^{(l)}$ is redundant. Assume that after training, a specific layer learned $\xi^{(l)} = \xi$ and the projection matrix (part of the cross-attention mechanism) $W_V = W_V^0$. Fixing $\xi^{(l)} = 1$, the model could simply learn $W_V = \xi \cdot W_V^0$, achieving the exact same result. However, the learned scaler serves another purpose: providing insight to GINAT usage of the information encoded in the prompt embedding. In this experiment, we test the effect of learning the scaler in comparison to having it fixed.

Since early experimentation showed that not scaling the cross-attention signal leads to vanishing gradients and poor performance, we test four initial values for ξ (learned or not): $1, 0.1,, 0.01, 0.001$. We evaluated GINAT on the node-level benchmark OGBN-Arxiv and the graph-level benchmark Proteins, each with the respective prompts dataset used in the main experiments. For comprehensive comparison, we evaluated each scaler setting of GINAT with four different LMs used for prompt embedding: `Qwen3-0.6B-Base`, `bge-small-en-v1.5`, `deberta-v3-small`, and `deberta-v3-large`. We used three MPNN backbones: GATv2, GCN, and GIN, each with four depth and two width configuration, setting a total of 96 configurations per benchmark per scaler configuration. The results of this experiment are presented in Figure 6, averaged across said configurations for comprehensive comparison. We draw two main conclusions from the results. First, it is evident that initializing the scaler with 1 yields the worst results, as expected. Second, there is no performance gain from using a fixed scaler; it is thus justified to use a learned scaler for our means.

Ablation: Initial Value of $\xi^{(l)}$. As evident in the fixed vs. learned $\xi^{(l)}$, the initial value of the parameter impacts the performance of the model. To better understand this dependency, we experiment with a wider range of initial values: $[1, 0.1, 0.01, 0.001]$. We chose this range of values because empirically, the magnitude of the prompt encoding was about $10x$ this of

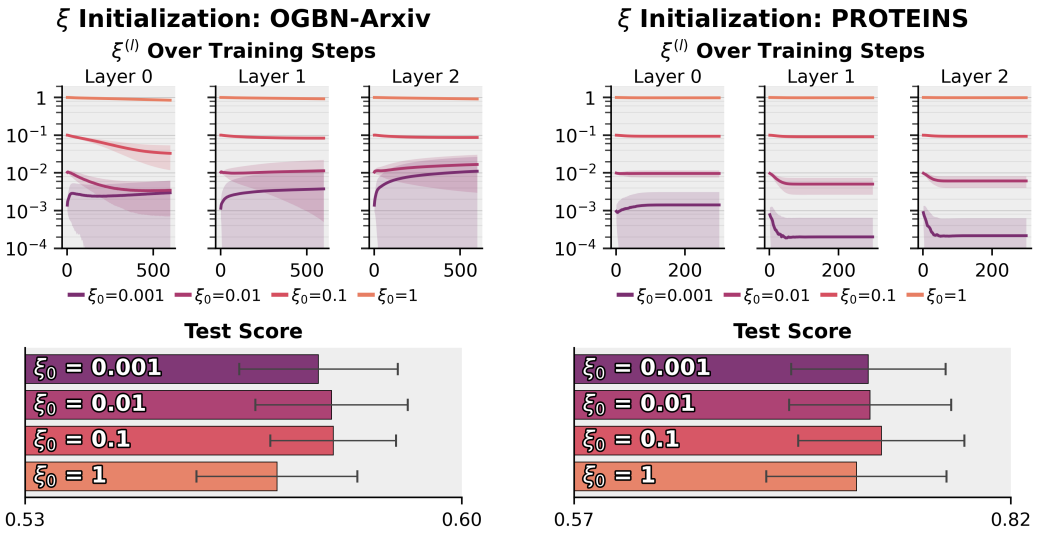

Figure 7: Results of the initial ξ_0 values ablation study on the OGBN-Arxiv and Proteins datasets. In top subfigures, we plot the training evolution of $\xi^{(l)}$, with std. range shaders. The bottom bar plots represent the test scores of the resulting experiments. Each bar and plot aggregate at least 96 runs.

the MPNN signal. In early experiments, this magnitude mismatch led to vanishing gradients to the MPNN and affected performance. We experiment on the same datasets and use the same encoder LMs, MPNN backbones and configuration variety in order to isolate the effect of the initial value. We also tracked the evolution of $\xi^{(l)}$ during training. The results are presented in Figure 7.

The results confirm our initial findings that $\xi_0 = 1$ leads to performance degradation, and that $\xi_0 = 0.1$ is the optimal choice. While the performance margin from the smaller values is negligible, the training dynamic plots present very large std. dev. values, which indicate those values are not robust. As $\xi_0 = 0.1$ performs best and is the most robust, we conclude that the optimal range for the initial $\xi^{(l)}$ value is $[0.01, 0.1]$.

F.6 Ablating the Prompts Source

The information encoded in the embedded prompts is a key element in GINAT. The aim of the following ablation study is to provide evidence for this claim. In this study, we provide the model with different types of prompt embeddings. We compare GINAT using each alternative prompt source against (i) the baseline MPNN and (ii) GINAT using the benchmark's proper prompt dataset. The proper prompts dataset for each benchmark is detailed in Appendix K.

Figure 4 shows two distinct experiments. In the first experiment, we train GINAT with GAT as the backbone MPNN and `roberta-base` as the prompt-embedding LM. The models are trained on the Squirrel node classification benchmark, using the proper prompts dataset and a dataset of prompts that are irrelevant to this benchmark.

In the second experiment, we train the same GINAT with the same backbone MPNN but use random vectors as prompt embeddings, sampled from Gaussian and Uniform distributions. In this experiment, we used several benchmarks: Chameleon, Squirrel, Computers, Photo, WikiCS, and Enzymes. For convenient comparison between the random inputs and the proper prompts dataset, we aggregate them based on the relative gains they achieved on each benchmark relative to the baseline GAT model.

Results. We provide here the results summarized in Figure 4. The prompts dataset used for each benchmark is detailed in Appendix K.

Table 6: Performance comparison against random-noise baselines. Gains (in %) are reported relative to the baseline GAT model.

Dataset	GINAT	Gaussian Noise	Uniform Noise	GAT
Chameleon	0.470 (+21.82%)	0.416 (+7.97%)	0.414 (+7.41%)	0.386
Squirrel	0.332 (+13.49%)	0.307 (+4.67%)	0.320 (+9.38%)	0.293
Computers	0.852 (+4.12%)	0.843 (+2.96%)	0.840 (+2.63%)	0.819
Photos	0.916 (+3.67%)	0.897 (+1.55%)	0.897 (+1.58%)	0.883
ENZYMES	0.517 (+30.99%)	0.322 (-18.31%)	0.361 (-8.45%)	0.394
WikiCS	0.732 (+4.22%)	0.674 (-4.04%)	0.681 (-3.11%)	0.702

Analysis and Discussion. The results of this study provide concrete evidence for our main hypothesis: pretrained LM priors can enhance graph learning. The random vectors experiment establishes that the GINAT performance gains stem from the LM distribution, and not merely from any computational or architectural gains. Taken together, these controls disentangle architecture from prior. Random Gaussian/Uniform vectors do not recover the gains, showing that improvements are not a generic effect of injecting additional features or parameters, but depend on the structure of the LM embedding manifold. Interestingly, unrelated natural-language prompts can still yield a non-trivial benefit, which we interpret as evidence for a generic distributional prior induced by pretrained LMs (e.g., smoothness/anisotropy and semantic clustering) that can regularize message passing even without benchmark-specific descriptions. However, the proper, domain-relevant prompts consistently perform best, showing that aligned content provides additional, task-relevant context beyond the generic LM prior.

F.7 Ablating the Prompt Encoder

To further establish GINAT ability to utilize the information encoded in the prompt embeddings and analyze it, we ablate the embedding LM. This experiment aims to evaluate GINAT's sensitivity to the choice of prompt encoder and to test whether different LM priors yield different gains.

Settings. In this experiment, we train GINAT using the same prompt datasets as in Section 4.1, and maintain the same overall experimental setting, changing only the LM used to embed the prompts. We use four different LM families, representing different approaches to language modeling: RoBERTa (Liu et al., 2019) and DeBERTa-V3 He et al. (2023) are encoder-only transformers, and are widely accepted as LM baselines. BGE-en-1.5v (Xiao et al., 2024) is an encoder specifically designed for text embedding, and Qwen3 (Yang et al., 2025) represents the LLM class. We evaluate GINAT with all publicly available sizes of those LMs: two for RoBERTA, three for BGE-en-1.5v, and four for Qwen3 and DeBERTa. This study evaluates GINAT's sensitivity to the choice and size of the prompt encoder; it is not intended as a benchmark for LMs.

Findings. This experiment included over 1000 different MPNN-LM-benchmark combinations (not counting hyper-parameter search for each such combination). To make sense of them, we compare the *relative gain* of GINAT with each embedding LM. For a benchmark-MPNN-LM triplet $(\text{ds}, \mathcal{M}, \mathbb{LM})$, the relative gain (formulated for higher-is-better metrics; for the opposite case, we negate this) is defined as

$$\frac{\text{test}(\text{GINAT}^{\mathbb{LM}}_{\mathcal{M}}, \text{ds}) - \text{test}(\mathcal{M}, \text{ds})}{\text{test}(\mathcal{M}, \text{ds})} \tag{11}$$

where $\text{test}(\cdot, \cdot)$ represents the test score of a model evaluated on a benchmark. This measure is useful since it allows us to compare gains on different benchmarks, which are individually measured on different scales.

The results on Table 7 are aggregated across the `Pubmed`, `CoCS`, `CoPhysics`, `Computers`, `Photo`, `Chameleon`, `Squirrel`, and `Wiki-CS` datasets. This collection represents graphs from different data modalities, as well as heterophilic graphs. The results on Table 8 are aggregated across the `OGBG-MolHIV`, `Proteins`, `IMDB-Multi`, `REDDIT-MULTI-5K`, and `ZINC`. Those represent benchmarks of different scales and domains, as well as feature-less graphs.

Table 7: Performance gain by applying GINAT to different MPNN architectures using different LMs for prompt embedding, averaged across 8 node level datasets. Note that when edge features are available, we use GINE instead of GIN. We aggregate them together.

	`BGE-1.5v small`	`BGE-1.5v base`	`BGE-1.5v large`	`DeBERTaV3 xsmall`	`DeBERTaV3 small`	`DeBERTaV3 base`	`DeBERTaV3 large`
GCN	6.19% ±7.57	7.32% ±8.96	6.92% ±8.67	6.34% ±7.21	6.21% ±7.38	7.00% ±9.48	7.89% ±10.2
GraphSAGE	6.33% ±11.1	5.22% ±9.39	5.05% ±8.65	5.11% ±9.59	4.88% ±10.9	4.64% ±9.32	5.08% ±8.3
GAT	6.22% ±6.54	5.15% ±4.96	4.80% ±4.94	4.76% ±4.86	6.32% ±6.29	5.52% ±6.17	5.35% ±5.85
GATv2	4.42% ±5.07	5.43% ±6.29	3.30% ±4.88	6.70% ±7.3	4.90% ±6.2	7.10% ±8.08	5.36% ±4.97
GIN	—	—	—	1.61% ±3.55	1.30% ±1.74	1.18% ±2.35	1.14% ±1.58
GraphConv	4.40% ±7.04	3.35% ±4.8	3.31% ±4.52	4.60% ±8.24	4.86% ±8.08	5.01% ±8.52	4.52% ±7.25
GraphGPS	2.07% ±2.36	3.42% ±4.32	1.87% ±1.32	3.56% ±3.89	2.67% ±3.19	3.51% ±5.12	2.46% ±2.23
Average	4.94%	4.98%	4.21%	4.67%	4.45%	4.85%	4.54%

	`RoBERTa base`	`RoBERTa large`	`Qwen3 0.6B-Base`	`Qwen3 1.7B-Base`	`Qwen3 4B-Base`	`Qwen3 8B-Base`	**Average**
GCN	6.03% ±7.21	7.22% ±8.83	7.03% ±7.64	6.40% ±7.57	6.85% ±8.85	7.50% ±10.4	6.84%
GraphSAGE	5.61% ±9.86	5.67% ±9.32	4.94% ±7.36	4.00% ±6.83	4.66% ±7.66	5.34% ±8.82	5.12%
GAT	6.54% ±7.17	6.59% ±6.03	6.07% ±5.97	5.94% ±6.35	6.03% ±6.44	5.11% ±5.49	5.72%
GATv2	5.63% ±6.29	8.50% ±9.79	5.22% ±5.75	4.68% ±7.13	3.43% ±9.97	5.83% ±8.12	5.42%
GIN	0.56% ±2.21	3.49% ±3.5	1.11% ±1.64	1.96% ±2.04	2.29% ±2.66	1.81% ±2.86	1.65%
GraphConv	4.94% ±7.18	4.29% ±6.29	2.94% ±5.66	3.77% ±5.37	3.87% ±4.64	3.48% ±4.94	4.10%
GraphGPS	2.13% ±1.94	3.69% ±3.32	3.24% ±2.61	2.62% ±4.07	3.41% ±4.37	3.00% ±3.48	2.90%
Average	4.49%	5.64%	4.36%	4.19%	4.36%	4.58%	4.63%

Table 1 shows a succinct summary of the results on three selected LMs, and Tables 7, 8 present them in more detail. Two main conclusions can be derived from the results. First, some MPNNs benefit more from GINAT than others. Specifically, GIN and GraphConv show the least overall improvement, while GAT and GATv2 improve significantly. Second, comparing the performance of different sizes of the same models shows no evidence that the capacity of the model improves the performance. While counterintuitive, this pattern might be explained by the relatively small and synthetic prompt datasets, which limit the benefits of higher-capacity encoders.

The significant performance gain for graph-level benchmarks is attributed to the excellent performance GINAT shows on benchmarks that lack node features, such as IMDB-Multi and the REDDIT-* benchmarks (all part of the TUDatasets collection Morris et al. (2020)). We discuss achievement in more depth in Section 4.1.

F.8 Full-Graph Transductive Setting on Node-Level Benchmarks

As explained in Section 4.1, the memory complexity of GINAT does not allow for full-graph transductive inference of most node-level benchmarks. In order to verify that the performance increases exhibited under the subgraph sampling setting, trained and evaluated GINAT on small-graph datasets

Table 8: Performance gain by applying GINAT to different MPNN architectures using different LMs for prompt embedding, averaged across 5 graph-level datasets.

	BGE-1.5v small	BGE-1.5v base	BGE-1.5v large	DeBERTaV3 xsmall	DeBERTaV3 small	DeBERTaV3 base	DeBERTaV3 large
GCN	13.34% ±10.1	12.16% ±12.3	12.73% ±12.1	15.48% ±12.7	16.97% ±14.7	16.90% ±13.2	15.92% ±13.3
GAT	48.83% ±70.7	51.27% ±72.6	46.09% ±63.5	54.32% ±70	50.38% ±61	51.03% ±64.8	50.75% ±62.7
GATv2	45.75% ±56.6	47.23% ±59.5	45.23% ±58.9	50.74% ±58	48.06% ±54.9	49.71% ±56.5	48.72% ±56.9
GIN	14.47% ±22.9	13.44% ±22.5	12.98% ±24.2	13.51% ±22.5	13.88% ±23.9	13.59% ±22.1	14.08% ±23
GraphConv	4.00% ±3.49	3.70% ±3.47	4.19% ±4.04	3.72% ±3.15	3.82% ±2.69	3.19% ±1.93	3.94% ±2.44
GraphGPS	37.31% ±35.3	37.37% ±33.8	33.32% ±27.2	32.56% ±29.8	35.72% ±30.3	36.86% ±31.4	34.30% ±28.7
Average	27.28%	27.53%	25.76%	28.39%	28.14%	28.55%	27.95%

	RoBERTa base	RoBERTa large	Qwen3 0.6B-Base	Qwen3 1.7B-Base	Qwen3 4B-Base	Qwen3 8B-Base	**Average**
GCN	15.48% ±11.3	15.15% ±12.2	12.38% ±10.3	13.62% ±12.7	12.76% ±11	11.70% ±10.2	14.20%
GAT	48.96% ±65.3	48.70% ±71.1	52.80% ±69.7	51.32% ±74.3	50.95% ±71.2	50.96% ±70.9	50.49%
GATv2	48.23% ±58.6	47.80% ±57.4	47.49% ±55.1	46.74% ±58.3	49.51% ±54.6	46.24% ±57.7	47.80%
GIN	14.47% ±23.1	14.01% ±22.5	12.97% ±23.8	13.69% ±22.9	14.08% ±22.6	11.41% ±23.6	13.58%
GraphConv	3.62% ±3.06	5.08% ±4.48	3.72% ±2.43	3.33% ±2.52	5.58% ±4.19	2.99% ±2.98	3.91%
GraphGPS	35.50% ±30.5	36.13% ±28	33.50% ±27.2	34.35% ±27.2	30.18% ±29	34.88% ±29.6	34.77%
Average	27.71%	27.81%	27.14%	27.17%	27.18%	26.36%	27.46%

F.9 Applying TAG Methods to Our Paradigm

As discussed in the Related Work and Experiments sections of our main paper, our proposed paradigm might remind some readers with *text attributed graphs* (TAGs). TAGs are a special class of graphs, where each node is associated with textual description. For example, consider the popular benchmark PubMed, whose node features stem from classic NLP techniques based on the contents of the represented research papers. A TAG version of this dataset may replace these constructed features with the abstract of each paper, for example. Then, during graph inference, an LM would be applied to each node content to generate its representation, which will be used as features for downstream GNN techniques.

Experiment Settings. Across all TAG-style experiments, each specific experiment and configuration is trained and evaluated three times using the same three seeds as in GINAT experiments. In recognition of their differences, we made particular efforts to adapt our paradigm to each of them in the most faithful way to its original architecture.

In TAPE He et al. (2024), node features comprise: (i) TA-pretrained LM features from titles and abstracts; (ii) P-top-k prediction features; and (iii) E-pretrained LM features derived mainly from explanations. For a fair comparison with GINAT, we replaced TA and E with the same prompt embeddings used by GINAT. Since P is an LM ranking output rather

than an embedding, we left it unchanged. We used the original TAPE model settings and hyperparameters, and run pre-training and inference with three random seeds, as in GINAT.

For LLM4NG Yu et al. (2025) and SIMTEG Duan et al. (2023), we replace the original prompt embeddings used by each method with our GINAT prompt embeddings while keeping the rest of the method unchanged. In both cases, we ensured dimensionality matching and restricted the modification to the prompt-embedding component only. All GNN settings and hyperparameters followed the corresponding original implementations.

Table 9: TAG Techniques performance under our paradigm. See Appendix F.9 for further discussion.

Method \ Dataset	PubMed	CoCS	CoPhysics
TAPE He et al. (2024)	0.398±.01	0.136±.00	0.354±.03
LLM4NG Yu et al. (2025)	0.384±.03	0.157±.04	0.505±.01
SIMTEG Duan et al. (2023)	0.405±.01	0.226±.00	0.502±.00
GINAT	**0.727±.01**	**0.923±.01**	**0.976±.00**

Findings. As evident in Table 9, TAG techniques cannot be utilized for our paradigm. In our proposed paradigm, we use an LM to embed external information about the graph data given as textual prompts, and those prompts are not node-specific. It is thus not surprising that while our paradigm might remind TAG techniques, TAG methods perform miserably under our setting.

G GraphSAGE: Failure to Utilize GINAT on Featureless Graphs

GraphSAGE Hamilton et al. (2017) exhibits a particularly informative interaction with GINAT. On datasets with informative node features, GraphSAGE generally benefits from the prompt signal, including substantial improvements on Enzymes and the Airports benchmarks. In contrast, its gains on the featureless Reddit benchmarks are negligible: accuracy changes from 0.510 to 0.517 on Reddit-Binary, from 0.198 to 0.200 on Reddit-Multi-5K, and remains 0.214 on Reddit-Multi-12K. The learned cross-attention scalers provide complementary evidence for this behavior. On Reddit-Multi-12K, GraphSAGE learns $\xi^{(\ell)} \approx 0$ in every layer, effectively reducing GINAT to the vanilla backbone. Thus, the negligible improvement does not merely indicate that the prompt contribution is small; it shows that the trained model actively suppresses a signal that it cannot use to improve inference in this setting.

This behavior follows from the interaction between the source-dependent cross-attention query and GraphSAGE's mean aggregation. The prompt-augmented GraphSAGE update can be written as

$$\mathbf{h}_v^{(\ell)} = \phi\left(\mathbf{W}_{\text{root}}^{(\ell)}\mathbf{h}_v^{(\ell-1)} + \mathbf{W}_{\text{nbr}}^{(\ell)}\frac{1}{d_v}\sum_{u\in\mathcal{N}(v)}\left[\mathbf{h}_u^{(\ell-1)} + \xi^{(\ell)}\mathcal{C}^{(\ell)}\left(\mathbf{h}_u^{(\ell-1)}, \mathbf{p}\right)\right]\right), \tag{12}$$

where $d_v = |\mathcal{N}(v)|$. As in GraphConv, the cross-attention query is the source representation $\mathbf{h}_u^{(\ell-1)}$, independently of the receiver v. Unlike GraphConv, however, GraphSAGE averages the incoming messages and therefore discards their multiplicity. If the initial node representations are constant, $\mathbf{h}_u^{(0)} = \mathbf{c}$, then every incoming message is identical and the first-layer update reduces to

$$\mathbf{h}_v^{(1)} = \phi\left(\mathbf{W}_{\text{root}}^{(1)}\mathbf{c} + \mathbf{W}_{\text{nbr}}^{(1)}\left[\mathbf{c} + \xi^{(1)}\mathcal{C}^{(1)}\left(\mathbf{c}, \mathbf{p}\right)\right]\right), \tag{13}$$

which is independent of d_v and of the composition of $\mathcal{N}(v)$. Every node consequently obtains the same representation. The same argument applies inductively to subsequent layers: if all representations entering a layer are identical, their prompt-conditioned messages and mean aggregates are also identical. With the virtual-node readout used in our graph-level experiments, the virtual node likewise receives the mean of identical representations and therefore cannot recover either graph size or topology.

The failure is therefore not evidence that GraphSAGE is generally incompatible with GINAT. Rather, it identifies a specific limitation of applying a global prompt signal through source-only messages followed by mean aggregation. When node features vary, the messages supplied to cross-attention also vary, allowing different nodes to retrieve different information from the prompt; this is consistent with the improvements observed on feature-attributed benchmarks. When the input is featureless, however, GINAT cannot itself break the symmetry between nodes, while mean aggregation removes the neighborhood multiplicity that could otherwise provide an initial structural distinction. GraphSAGE consequently has no graph-dependent query through which to align topology with the LM prior and learns to suppress the cross-attention residual. This result further clarifies the contribution of GINAT to graph inference: the embedded prompt enriches structural information exposed by the underlying message-passing rule, but does not replace that structural information when the backbone eliminates it before aggregation.

H Additional Results

We list here GINAT's performance against the MPNN baselines, which we did not have the room to include in Table 2 and 3. GINAT results are highlighted in light red. Best results for each MPNN-dataset pair are underlined, and the best results for a datasets are **bolded**.

H.1 Graph Level Results

Table 10: Additional graph-level results. All experiments ran under the same settings as in Table 2 and are presented here due to space limitations only.

Model ↓	IMDB-BINARY	IMDB-MULTI	NCI1	REDDIT-MULTI12K
Metric →	Acc. ↑	Acc. ↑	Acc. ↑	Acc. ↑
GCN	0.607 ±0.06	0.371 ±0.01	0.779 ±0.04	0.459 ±0.00
	0.747 ±0.06	0.489 ±0.03	0.804 ±0.04	0.476 ±0.01
GraphSAGE	0.500 ±0.00	0.327 ±0.03	0.783 ±0.03	0.214 ±0.01
	0.527 ±0.02	0.329 ±0.03	0.796 ±0.02	0.214 ±0.01
GAT	0.507 ±0.01	0.336 ±0.03	0.702 ±0.09	0.214 ±0.01
	0.753 ±0.06	0.469 ±0.05	0.783 ±0.02	0.469 ±0.02
GATv2	0.500 ±0.00	0.327 ±0.03	0.750 ±0.02	0.214 ±0.01
	0.743 ±0.05	0.496 ±0.04	0.780 ±0.03	0.471 ±0.01
GIN[1]	0.723 ±0.03	0.476 ±0.06	0.823 ±0.03	0.498 ±0.01
	0.740 ±0.03	**0.509** ±0.01	**0.828** ±0.03	**0.504** ±0.01
GraphConv	0.737 ±0.04	0.469 ±0.05	0.806 ±0.02	0.492 ±0.02
	0.753 ±0.06	0.502 ±0.03	0.817 ±0.04	0.496 ±0.01
GraphGPS	0.717 ±0.01	0.471 ±0.05	0.800 ±0.03	0.233 ±0.01
	0.747 ±0.03	0.500 ±0.02	0.799 ±0.04	0.474 ±0.03

H.2 Node Level Results

Table 11: Additional node-level results. All experiments ran under the same settings as in Table 3 and are presented here due to space limitations only.

Model ↓	CiteSeer	CoPhysics	Wiki-CS	AIRPORTS USA	Squirrel
GCN	51.78 ±1.94	96.65 ±0.44	75.19 ±0.86	44.90 ±1.00	31.15 ±0.52
	44.84 ±0.43	97.35 ±0.58	75.00 ±0.61	47.60 ±2.00	34.40 ±2.53
GraphSAGE	54.86 ±2.29	96.96 ±0.19	77.39 ±1.39	50.00 ±1.00	36.43 ±5.02
	55.04 ±1.12	97.31 ±0.39	**77.91** ±0.69	**51.60** ±0.00	**37.61** ±4.08
GAT	38.01 ±0.95	96.44 ±0.45	72.63 ±1.70	45.70 ±2.00	33.09 ±0.82
	42.98 ±1.01	97.02 ±0.40	74.36 ±0.65	48.30 ±2.00	36.84 ±0.42
GATv2	40.46 ±0.36	97.05 ±0.44	72.26 ±0.52	48.73 ±4.20	33.31 ±2.49
	41.35 ±1.44	**97.55** ±0.31	74.36 ±0.78	50.42 ±0.84	34.47 ±2.20
GIN	46.51 ±0.82	96.60 ±0.26	72.79 ±3.01	44.90 ±0.00	34.24 ±1.91
	48.32 ±3.47	97.29 ±0.81	72.83 ±2.54	49.10 ±1.00	34.85 ±2.29
GraphConv	54.44 ±1.68	96.85 ±0.14	73.97 ±0.93	49.50 ±2.00	35.06 ±3.57
	55.02 ±2.06	96.95 ±0.71	74.36 ±0.92	**51.60** ±2.00	35.39 ±0.65
GraphGPS	44.82 ±1.89	97.21 ±0.42	73.91 ±0.89	39.40 ±4.00	35.76 ±4.82
	45.76 ±1.17	97.36 ±1.11	75.47 ±1.65	41.80 ±3.00	36.20 ±2.07

I Cross-attention for GNN-LLM Integration

Cross-attention has emerged as a central mechanism for embedding and aligning multi-modal information, enabling models to selectively integrate representations across heterogeneous modalities. For example, they were proven effective in vision-language models (Li et al., 2023; Alayrac et al., 2022), where textual tokens are grounded in visual features through learned cross-modal interactions.

Formally, given two node feature matrices $X_G \in \mathbb{R}^{n_G \times d_G}$ and $X_T \in \mathbb{R}^{n_T \times d_T}$, the cross-attention layer consists of three learned weight matrices that project the features into a shared vector space of dimension d. Without loss of generality, we use X_G as the queries and X_T as the keys and values, with projection matrices $W_Q \in \mathbb{R}^{d_G \times d}$, $W_K \in \mathbb{R}^{d_T \times d}$, and $W_V \in \mathbb{R}^{d_T \times d}$. We then compute

$$Q = X_G W_Q^T \qquad K = X_T W_K^T \qquad V = X_T W_V^T$$

which are usually termed the query, key, and value matrices, respectively. The cross-attention operator is then defined as:

$$\text{CrossAttention}\left(X_G, X_T\right) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V \tag{14}$$

In our GINAT, we explore cross-attention as an interface between GNNs and LLMs. Concretely, we consider GNNs that provide embeddings $H_G \in \mathbb{R}^{|V| \times d}$ that encode structural dependencies among nodes. Specifically, as described in Section 2, we use the messages passed between nodes as queries. As keys and values, we consider LLM-generated contextualized token embeddings $H_T \in \mathbb{R}^{L \times d}$. Cross-attention is then employed to align these spaces,

$$Z_G = \text{CrossAttention}(H_G, H_T), \tag{15}$$

which enriches the messages with semantic cues from language.

J Background: Message Passing Neural Networks (MPNNS)

Let $G = (V, E)$ be a graph with node set V and (directed or undirected) edge set $E \subseteq V \times V$. Each node $v \in V$ may have a feature vector $h_v^{(0)} \in \mathbb{R}^{d_0}$ and each edge $(u, v) \in E$ may have features $e_{uv} \in \mathbb{R}^{d_e}$. MPNNs produce layer-wise node representations $\{h_v^{(l)}\}_{v \in V}$ for $l = 0, \ldots, L$ by repeatedly exchanging *messages* along edges and updating local states. A single layer $l \to l+1$ is defined by three components:

$$\textit{message}(\cdot), \quad \textit{aggregate}(\cdot), \quad \textit{update}(\cdot).$$

Message. For each incoming edge $(u, v) \in E$, node u sends a message to v:

$$m_{u \to v}^{(l)} \;=\; \textit{message}\big(h_u^{(l)},\, h_v^{(l)},\, e_{uv}\big) \;\in\; \mathbb{R}^{d_m}. \tag{16}$$

The function *message* can be a neural network (e.g., MLP) or a parametric rule (e.g., linear map, attention score). It may depend on source features $h_u^{(l)}$, target features $h_v^{(l)}$, and edge features e_{uv}.

Aggregate. Node v aggregates all incoming messages from its neighborhood $\mathcal{N}(v) = \{u \mid (u, v) \in E\}$ using a permutation-invariant operator:

$$\bar{m}_v^{(l)} \;=\; \textit{aggregate}\Big(\{\, m_{u \to v}^{(l)} : u \in \mathcal{N}(v) \,\}\Big). \tag{17}$$

Common choices include *sum*, *mean*, *max*, learned attention-weighted sums, or more expressive set functions. Permutation invariance ensures that reordering neighbors does not change the result.

Update. Node v updates its representation by combining its previous state with the aggregated message:

$$h_v^{(l+1)} \;=\; \textit{update}\big(h_v^{(l)},\, \bar{m}_v^{(l)}\big) \;\in\; \mathbb{R}^{d_{l+1}}. \tag{18}$$

Typical updates are MLPs, gated mechanisms (e.g., GRU), residual additions with normalization, or transformer-style blocks. With permutation-invariant *aggregate*, the mapping $H^{(l)} \mapsto H^{(l+1)}$ is *permutation equivariant* at the node level.

Readout for graph-level tasks. For graph-level prediction, node embeddings after L layers are pooled into a single graph representation via a permutation-invariant *readout*:

$$h_G \;=\; \textit{readout}\Big(\{\, h_v^{(L)} : v \in V \,\}\Big), \qquad \hat{y}_G \;=\; f_{\text{head}}(h_G). \tag{19}$$

Common readouts include *sum*/*mean*/*max* pooling, attention pooling, Set2Set, SortPool, or hierarchical pooling. The invariance of *readout* guarantees that $\hat{y}_G$ is independent of node ordering.

Design choices and examples.

- **GCN-style layers.** Messages linearly transform neighbors, often with degree normalization, and *aggregate* is summation/mean; *update* applies a nonlinearity.
- **GAT-style layers.** *message* uses learned attention coefficients α_{uv} over neighbors; *aggregate* is the attention-weighted sum; *update* applies projection and nonlinearity.
- **Edge-aware MPNNs.** Edge features e_{uv} modulate messages (e.g., via edge-conditioned filters).
- **Stabilization.** Residual/skip connections, normalization (batch/layer), and dropout are commonly used to improve optimization and mitigate oversmoothing.

Properties. With permutation-invariant *aggregate* (and *readout*), MPNNs are permutation-equivariant at the node level and permutation-invariant at the graph level. Expressivity grows with depth and the choice of *message*/*aggregate*/*update*, and can be related to the Weisfeiler–Leman hierarchy under suitable conditions.

Summary. An MPNN layer computes messages on edges (16), aggregates them per node (17), and updates node states (18); a graph-level predictor applies a permutation-invariant readout (19). This decomposition provides a flexible template that subsumes many popular GNN architectures while preserving the necessary symmetry constraints of graph data.

K Prompts Dataset

This appendix provides statistics and sample prompts for each prompt dataset used in our experiments. The complete datasets used will be published upon acceptance.

K.1 Summary Table

Table 12: Summary of prompt datasets used in the experiments.

Prompt Dataset	# Prompts	Word Count (min / mean / max)	Benchmarks Used
Molecular Prompts	193	10 / 12.5 / 15	OGBG-MOLHIV, MUTAG, PROTEINS, ENZYMES, NCI1, NCI109, ZINC
IMDB Prompts	197	10 / 13.0 / 16	IMDB-Binary, IMDB-Multi
Reddit Prompts	210	8 / 15.5 / 23	Reddit-Binary, Reddit-Multi-5K, Reddit-Multi-12K
Citation Prompts	300	8 / 15.5 / 23	PubMed, CiteSeer, OGBN-Arxiv, CoCS, CoPhysics
Amazon Prompts	300	8 / 12.5 / 17	Computers, Photo
Wiki Prompts	300	8 / 13.0 / 18	Chameleon, Squirrel, WikiCS
Transport Prompts	100	11 / 12.4 / 17	Airports-USA, Airports-Europe, Airports-Brazil

K.2 Moleular Prompts

Number of prompts: 193
Word count statistics: min=10, mean=12.5, max=15
Used by datasets: OGBG-MOLHIV, MUTAG, PROTEINS, ENZYMES, NCI1, NCI109, ZINC

Sample Prompts

1. "In a molecular graph, nodes are atoms and edges are chemical bonds."
2. "Each node in this graph represents an atom in a molecule."
3. "The graph structure captures the molecular topology of a chemical compound."
4. "Nodes represent atoms connected by edges representing covalent bonds."
5. "This molecular graph encodes the structural formula of a chemical compound."

K.3 IMDB Prompts

Number of prompts: 197
Word count statistics: min=10, mean=13.0, max=16
Used by datasets: IMDB-Binary, IMDB-Multi

Sample Prompts

1. "This graph represents a movie collaboration network from IMDB."
2. "Nodes in this graph are actors who appeared in the same movie."
3. "Edges connect actors who have collaborated in at least one film."

4. "The graph captures professional relationships between movie actors."
5. "Each node represents an actor and edges represent movie collaborations."

K.4 Reddit Prompts

Number of prompts: 210
Word count statistics: min=8, mean=15.5, max=23
Used by datasets: Reddit-Binary, Reddit-Multi-5K, Reddit-Multi-12K

Sample Prompts

1. "This graph represents a discussion thread from Reddit."
2. "Nodes are users who commented in the same online discussion thread."
3. "Edges connect users who replied to each other in a Reddit thread."
4. "The graph captures social interactions in an online community discussion."
5. "Each node represents a Reddit user participating in a discussion."

K.5 Citation Prompts

Number of prompts: 300
Word count statistics: min=8, mean=15.5, max=23
Used by datasets: PubMed, CiteSeer, OGBN-Arxiv, CoCS, CoPhysics

Sample Prompts

1. "This is a citation network where nodes are scientific papers."
2. "Edges represent citation relationships between academic publications."
3. "Each node is a research paper connected to papers it cites or is cited by."
4. "The graph captures the knowledge flow in academic literature."
5. "Nodes represent scholarly articles and edges represent citations."

K.6 Amazon Prompts

Number of prompts: 300
Word count statistics: min=8, mean=12.5, max=17
Used by datasets: Computers, Photo

Sample Prompts

1. "This graph represents product co-purchasing patterns from Amazon."
2. "Nodes are products and edges connect frequently co-purchased items."
3. "The graph captures consumer purchasing behavior in e-commerce."
4. "Each node represents a product in an online retail network."
5. "Edges indicate that two products are often bought together."

K.7 Wiki Prompts

Number of prompts: 300
Word count statistics: min=8, mean=13.0, max=18
Used by datasets: Chameleon, Squirrel, WikiCS

Sample Prompts

1. "This graph represents Wikipedia page hyperlinks."
2. "Nodes are Wikipedia articles connected by hyperlink references."
3. "The graph captures the knowledge structure of Wikipedia."
4. "Each node is a Wikipedia page with edges to linked pages."
5. "Edges represent hyperlinks between Wikipedia articles."

K.8 Transport Prompts

Number of prompts: 100
Word count statistics: min=11, mean=12.4, max=17
Used by datasets: Airports-USA, Airports-Europe, Airports-Brazil

Sample Prompts

1. "A singular airport within a global aviation infrastructure can be represented as a node."
2. "A node representing a specific aviation hub within a dense air transport network structure."
3. "An individual airport in a transportation network is represented by a singular node within that flight structure."
4. "A vertex in a geospatial aviation network denoting a fixed site for aircraft arrivals and departures."
5. "A node within an interconnected transport graph identifying a specific hub of civil aviation."

L Datasets information

L.1 Molecular Graph Datasets

Table 13: Summary statistics of molecular datasets. Feature columns report only dimensionality. Totals are estimated as #graphs × average edges.

Dataset	Graphs	Nodes	Edges	Tot. E	Node dim	Edge dim
OGBG-MOLHIV	41,127	25.5	27.5	1,130,992	9	3
MUTAG	188	17.9	19.8	3,721	1	1
ENZYMES	600	32.6	62.1	37,284	19	–
PROTEINS	1,113	39.1	72.8	81,049	2	–
NCI1	4,110	29.9	32.3	132,753	1	–
NCI109	4,127	29.7	32.1	132,601	1	–
ZINC (full)	249,456	23.2	24.9	6,211,454	1	1

OGBG-MOLHIV. A molecular property prediction benchmark: binary classification of whether a molecule inhibits HIV replication. Nodes are atoms with 9 categorical features (atomic number, chirality, aromaticity, etc.); edges are bonds with 3 categorical features (bond type, stereo, conjugation).

MUTAG. A dataset of nitroaromatic compounds labeled by mutagenicity. Nodes are atoms (atom type); edges are bonds (bond type).

ENZYMES. Contains 600 protein tertiary-structure graphs classified into 6 enzyme commission (EC) classes. Nodes are secondary structure elements (SSEs) with one categorical type (helix/sheet/turn) and 18 continuous attributes. Together, this yields 19 node dimensions. Edges connect interacting SSEs.

PROTEINS. Protein tertiary-structure graphs with binary labels (enzyme vs. non-enzyme). Nodes are SSEs with a categorical type and one continuous attribute (2 total dimensions). PROTEINS_full provides 29 attributes.

NCI1 / NCI109. Molecules tested for anti-cancer activity. Nodes are atoms (1 categorical type dimension); edges are unlabeled. Graph labels indicate activity against cancer cell lines.

ZINC. Large dataset of drug-like molecules from the ZINC database. Nodes are atoms (1 categorical type dimension), edges are bonds (1 categorical type dimension). Used primarily for property regression tasks.

L.2 Social Network Graph Datasets

Table 14: Summary statistics for Reddit and IMDB graph classification datasets. Feature columns report only dimensionality. Totals are estimated as #graphs × avg. edges.

Dataset	Graphs	Nodes	Edges	Tot. E	Node dim	Edge dim
REDDIT-BINARY	2,000	429.6	995.5	1,991,000	0	0
REDDIT-MULTI-5K	4,999	508.5	594.9	2,973,905	0	0
REDDIT-MULTI-12K	11,929	391.4	462.0	5,511,198	0	0
IMDB-BINARY	1,000	19.8	96.5	96,500	0	0
IMDB-MULTI	1,500	13.0	65.9	98,850	0	0

REDDIT-BINARY, REDDIT-MULTI-5K, REDDIT-MULTI-12K. Each graph represents a Reddit discussion thread. Nodes are users; edges connect users if one replied to the other. Graph labels correspond to subreddit/community categories (2, 5, and 11 classes, respectively).

IMDB-BINARY / IMDB-MULTI. Graphs are ego-networks of movie collaborations. Nodes are actors/actresses, and edges indicate co-appearance in a movie. Graph labels are genres: IMDB-BINARY distinguishes Action vs. Romance, while IMDB-MULTI has three genres (Comedy, Romance, Sci-Fi).

L.3 Node-level Datasets

Table 15: Summary statistics for node classification datasets.

Dataset	Graphs	Nodes	Edges	Tot. E	Classes	Node dim	Edge dim
PubMed	1	19,717	44,338	44,338	3	500	0
CiteSeer	1	3,327	4,732	4,732	6	3,703	0
WikiCS	1	11,701	216,123	216,123	10	300	0
Chameleon	1	2,277	36,101	36,101	5	2,325	0
Squirrel	1	5,201	217,073	217,073	5	2,089	0
Computers	1	13,752	245,866	245,866	10	767	0
Photo	1	7,650	119,081	119,081	8	745	0
CoCS	1	18,333	81,894	81,894	15	6,805	0
CoPhysics	1	34,493	247,962	247,962	5	8,415	0
Airports-USA	1	1,190	13,599	13,599	4	1,190	0
Airports-Europe	1	399	5,995	5,995	4	399	0
Airports-Brazil	1	131	1,074	1,074	4	131	0

PubMed and CiteSeer : Classic citation networks in which nodes represent scientific papers and edges represent citation links. Node features are sparse textual representations of the papers.

WikiCS : A Wikipedia network from the Computer Science domain. Nodes correspond to Wikipedia pages, edges represent hyperlinks, and node features are derived from article text.

Chameleon & Squirrel : Wikipedia networks based on specific topics. These datasets are commonly used to evaluate models on heterophilous graphs, where connected nodes often belong to different classes.

Computers & Photo : Segments of the Amazon co-purchase graph. Nodes represent products, and edges indicate that two items are frequently bought together. Node features are bag-of-words representations of product reviews.

CoCS & CoPhysics : Co-authorship networks from the Microsoft Academic Graph. Nodes represent authors, and edges indicate that two authors have co-authored at least one paper. Node features are based on paper keywords.

Airports-USA, Airports-Europe, and Airports-Brazil : Airport traffic networks from the struc2vec benchmark. Nodes represent airports, edges represent connections between airports, and labels correspond to airport activity levels. Following the PyTorch Geometric version of these datasets, node features are one-hot encoded node identifiers.

OGBN-Arxiv : A massive, directed citation network of Computer Science papers on arXiv. This is a benchmark dataset from the Open Graph Benchmark (OGB) and includes a temporal aspect (publication dates).

M Dataset Training Configuration

As discussed in Section 4, for each dataset and GNN backbone pair, we searched for the optimal layer count and layer width combination. The table below details the configuration used for each dataset. We reiterate that the same configuration was used for the baseline model and the LLM augmented model.

Table 16: Configuration used for graph-level datasets, given in a layers/hidden dimension format.

Dataset	GAT	GATv2	GCN	GIN	GPS	GraphCon	GraphSAGE
Enzymes	4 / 512	4 / 256	4 / 256	4 / 256	4 / 512	5 / 512	3 / 256
IMDB-BINARY	5 / 256	4 / 256	4 / 256	5 / 512	5 / 256	4 / 512	4 / 512
IMDB-MULTI	4 / 512	3 / 256	5 / 256	5 / 256	3 / 256	5 / 512	4 / 256
MUTAG	4 / 512	5 / 512	4 / 512	5 / 256	3 / 256	4 / 512	3 / 256
NCI1	5 / 256	4 / 256	5 / 512	4 / 256	5 / 256	5 / 512	5 / 512
NCI109	5 / 256	3 / 512	5 / 256	5 / 512	5 / 256	5 / 256	5 / 256
OGBG-MolHIV	5 / 256	4 / 512	5 / 512	3 / 256	3 / 512	5 / 512	3 / 512
Proteins	3 / 256	3 / 512	3 / 256	5 / 512	3 / 256	5 / 512	3 / 256
REDDIT-B	4 / 256	3 / 256	5 / 256	5 / 512	4 / 512	5 / 512	3 / 256
REDDIT-M12k	3 / 256	3 / 256	5 / 512	5 / 512	3 / 256	5 / 256	3 / 256
REDDIT-M5k	5 / 256	3 / 256	4 / 512	5 / 512	5 / 256	4 / 256	4 / 512
ZINC	3 / 512	3 / 512	3 / 512	3 / 512	4 / 512	4 / 512	4 / 512

Table 17: Configuration used for node-level datasets, given in a layers/hidden dimension format.

Dataset	GAT	GATv2	GCN	GIN	GPS	GraphCon	GraphSAGE
CiteSeer	5 / 512	4 / 256	4 / 256	3 / 512	3 / 512	5 / 512	3 / 256
Chameleon	4 / 256	4 / 256	3 / 256	4 / 512	4 / 256	4 / 512	3 / 512
CoCS	4 / 256	3 / 512	3 / 256	3 / 256	4 / 256	3 / 512	3 / 512
Computers	3 / 512	3 / 256	3 / 256	3 / 256	3 / 256	3 / 512	3 / 256
CoPhysics	4 / 256	4 / 256	4 / 256	4 / 256	5 / 512	5 / 256	5 / 512
Photo	3 / 512	4 / 256	3 / 512	4 / 512	3 / 512	3 / 256	4 / 256
PubMed	5 / 256	3 / 256	4 / 512	3 / 512	3 / 512	4 / 512	4 / 512
Squirrel	3 / 256	5 / 512	4 / 512	5 / 512	3 / 256	4 / 256	3 / 256
WikiCS	3 / 256	3 / 256	3 / 256	3 / 512	3 / 512	3 / 256	3 / 256
Airports-USA	4 / 1024	4 / 1024	3 / 1024	3 / 1024	4 / 128	3 / 1024	3 / 1024
Airports-Europe	3 / 128	3 / 128	4 / 512	4 / 512	4 / 1024	4 / 1024	3 / 1024
Airports-Brazil	3 / 512	3 / 512	3 / 512	3 / 1024	4 / 1024	3 / 256	4 / 128

N Hyper Parameters Settings

N.1 GINAT settings

This includes the settings for used to train GINAT and the MPNN baselines. We note that GINAT and the MPNNs made use of the exact same hyperparameters settings in each experiment. Setting are available in Tables 18, 19.

Dataset	LR	LR Scheduler	Patience	Batch Size	Epochs	Loss Function
OGBG-MOLHIV	1e-4	ReduceLROnPlateau	50	512	200	BCEWithLogitsLoss
MUTAG	1e-4	ReduceLROnPlateau	5	64	60	CrossEntropy
PROTEINS	1e-4	ReduceLROnPlateau	5	64	150	CrossEntropy
ENZYMES	1e-4	ReduceLROnPlateau	5	64	150	CrossEntropy
NCI1	1e-4	ReduceLROnPlateau	10	64	120	CrossEntropy
NCI109	1e-4	ReduceLROnPlateau	10	64	120	CrossEntropy
IMDB-Binary	1e-4	ReduceLROnPlateau	10	48	100	CrossEntropy
IMDB-Multi	1e-4	ReduceLROnPlateau	10	48	100	CrossEntropy
Reddit-Binary	1e-4	ReduceLROnPlateau	10	48	100	CrossEntropy
Reddit-Multi-5K	1e-4	ReduceLROnPlateau	10	48	100	CrossEntropy
Reddit-Multi-12K	1e-4	ReduceLROnPlateau	10	48	100	CrossEntropy
ZINC	1e-4	ReduceLROnPlateau	5	1024	100	MAELoss

Table 18: GINAT Training hyperparameters and optimization settings across graph-level datasets

Dataset	LR	LR Scheduler	Batch Size	Epochs	Num Steps	Loss Function
PubMed	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
CiteSeer	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Computers	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Photo	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Arxiv	1e-4	ReduceLROnPlateau	8192	300	10	CrossEntropy
CoCS	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
CoPhysics	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Squirrel	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Chameleon	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
WikiCS	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
USA	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Europe	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy
Brazil	1e-4	ReduceLROnPlateau	256	40	50	CrossEntropy

Table 19: GINAT Training hyperparameters for node-level classification benchmarks

N.2 Graphormer

Training settings and hyperparameters appear in Table 21 for node level datasets and Table 20 for graph level datasets.

N.3 GraphMAE

Training settings and hyperparameters appear in Table 23 for node level datasets and Table 22 for graph level datasets.

Dataset	Hidden Dim	#Layers	Heads	Dropout (In)	Dropout (Hid)	LR	Weight Decay	Epochs	Batch Size
OGBG-MOLHIV	512	3	8	0.1	0.0	$1e{-4}$	$5e{-4}$	200	512
MUTAG	256	4	8	0.1	0.0	$1e{-4}$	$5e{-4}$	60	64
PROTEINS	256	4	8	0.1	0.0	$1e{-4}$	$5e{-4}$	150	64
ENZYMES	256	4	8	0.1	0.0	$1e{-4}$	$5e{-4}$	150	64
NCI1	256	4	8	0.1	0.0	$1e{-4}$	$5e{-4}$	120	64
NCI109	256	4	8	0.1	0.0	$1e{-4}$	$5e{-4}$	120	64
IMDB-Binary	256	2	8	0.1	0.0	$1e{-4}$	$5e{-4}$	100	48
IMDB-Multi	256	2	8	0.1	0.0	$1e{-4}$	$5e{-4}$	100	48
Reddit-Binary	256	2	8	0.1	0.0	$1e{-4}$	$5e{-4}$	100	48
Reddit-Multi-5K	256	2	8	0.1	0.0	$1e{-4}$	$5e{-4}$	100	48
Reddit-Multi-12K	256	2	8	0.1	0.0	$1e{-4}$	$5e{-4}$	100	48
ZINC	256	4	8	0.1	0.0	$1e{-4}$	$5e{-4}$	100	1024

Table 20: Graphormer hyperparameter settings for all graph datasets

Dataset	Hidden Dim	#Layers	Heads	Dropout (In)	Dropout (Hid)	LR	Weight Decay	Epochs	Batch Size	Neighbors
PubMed	128	2	8	0.4	0.5	$1e{-2}$	$5e{-4}$	200	Full	–
Computers	64	2	8	0.4	0.5	$1e{-2}$	$5e{-4}$	150	Full	–
Photo	64	2	8	0.4	0.5	$1e{-2}$	$5e{-4}$	150	Full	–
OGBN-Arxiv	128	3	4	0.4	0.2	$1e{-3}$	$5e{-4}$	50	1024	[15, 10]
CoCS	64	2	8	0.4	0.5	$1e{-2}$	$5e{-4}$	100	Full	–
CoPhysics	64	2	8	0.4	0.5	$1e{-2}$	$5e{-4}$	100	Full	–
Squirrel	64	2	4	0.2	0.5	$5e{-3}$	$5e{-4}$	200	Full	–
Chameleon	64	2	4	0.2	0.5	$5e{-3}$	$5e{-4}$	200	Full	–
WikiCS	64	2	8	0.4	0.5	$1e{-2}$	$5e{-4}$	100	Full	–

Table 21: Graphormer hyperparameter settings for all node-datasets

Dataset	Hidden Dim	#Layers	Heads	Dropout (In)	Dropout (Hid)	LR	Weight Decay	Epochs	Batch Size
OGBG-MOLHIV	512	3	2	0	0	$1e{-4}$	0	200	512
MUTAG	256	4	2	0.2	0.1	$1e{-4}$	0	60	64
PROTEINS	256	4	2	0.2	0	$1e{-4}$	0	150	64
ENZYMES	256	4	2	0.2	0	$1e{-4}$	0	150	64
NCI1	256	4	2	0.2	0	$1e{-4}$	0	120	64
NCI109	256	4	2	0.2	0	$1e{-4}$	0	120	64
IMDB-Binary	256	2	2	0.2	0	$1e{-4}$	0	100	48
IMDB-Multi	256	2	2	0.2	0	$1e{-4}$	0	100	48
Reddit-Binary	256	2	2	0.2	0	$1e{-4}$	$2e{-4}$	100	48
Reddit-Multi-5K	256	2	2	0.2	0	$1e{-4}$	$2e{-4}$	100	48
Reddit-Multi-12K	256	2	2	0.2	0	$1e{-4}$	$2e{-4}$	100	48
ZINC	256	4	2	0.2	0	$1e{-4}$	0	100	1024

Table 22: GraphMAE hyperparameter settings for all graph datasets

Dataset	Hidden Dim	#Layers	Heads	Dropout (In)	Dropout (Hid)	LR	Weight Decay	Epochs	Batch Size
PubMed	1024	2	4	0.2	0.1	$1e{-3}$	$1e{-5}$	1000	Full
Squirrel	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
Chameleon	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
WikiCS	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
Computers	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
Photo	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
CoCS	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
CoPhysics	512	2	4	0.2	0.1	$1e{-3}$	$2e{-4}$	1500	Full
OGBN-Arxiv	1024	3	4	0.2	0.1	$1e{-3}$	0	1000	1024

Table 23: GraphMAE hyperparameter settings for all node datasets