A Graph Laplacian Eigenvector-based Pre-training Method for Graph Neural Networks

Howard Dai[1,*] **Nyambura Njenga**[2,*] **Hiren Madhu**[1] **Siddharth Viswanath**[1]
Ryan Pellico[3,†] **Ian Adelstein**[1,†] **Smita Krishnaswamy**[1,†]

[1]Yale University [2]Georgia Institute of Technology [3]Trinity College
[*]co-first authors [†]co-senior authors
Correspondence: smita.krishnaswamy@yale.edu

Abstract

The development of self-supervised graph pre-training methods is a crucial ingredient in recent efforts to design robust graph foundation models (GFMs). *Structure-based* pre-training methods are under-explored yet crucial for downstream applications which rely on underlying graph structure. In addition, pre-training traditional message passing GNNs to capture global and regional structure is often challenging due to the risk of oversmoothing as network depth increases. We address these gaps by proposing the Laplacian Eigenvector Learning Module (LELM), a novel pre-training module for graph neural networks (GNNs) based on predicting the low-frequency eigenvectors of the graph Laplacian. Moreover, LELM introduces a novel architecture that overcomes oversmoothing, allowing the GNN model to learn long-range interdependencies. Empirically, we show that models pre-trained via our framework outperform baseline models on downstream molecular property prediction tasks.

1 Introduction

Graph Neural Networks (GNNs) have become a powerful tool in node and graph representation learning, with successful applications across domains ranging from biomedicine (Cantürk et al., 2023; Yan et al., 2024) to social networks (Fan et al., 2019). More recently, graph foundation models (GFMs) are emerging as an exciting field; inspired by the success of large language models (LLMs), researchers are exploring the possibility of creating large graph-based models with emergent capabilities across many domains (Liu et al., 2025; Xie et al., 2022; Wang et al., 2025).

A key ingredient in this effort is the creation of self-supervised tasks which can be performed on large unlabeled graph datasets (Liu et al., 2022). A variety of pre-training methods have been proposed, with the majority of such methods being feature-based, including contrastive losses and feature reconstruction (Liu et al., 2025; Xie et al., 2022). A few structure-based methods, which precompute labels based on graph topology, have been proposed (Peng et al., 2020a; Hwang et al., 2020). However, this category of pre-training approach, recently termed *graph property prediction* (Liu et al., 2025), remains under-explored. Furthermore, limitations of GNNs in capturing long-range structural information caused by oversmoothing and underreaching (Cai & Wang, 2020; Oono & Suzuki, 2019; Keriven, 2022) pose challenges when approaching graph property prediction tasks which rely on global structure.

We propose the Laplacian Eigenvector Learning Module (LELM) for GNNs and GFMs. The low-frequency Laplacian eigenvectors capture a range of global, regional, and local graph structure, making them well-suited as a structural target. LELM utilizes a novel global MLP prediction head during pre-training that allows the GNN model to learn long-range relationships without requiring

Proceedings of the 1st Conference on Topology, Algebra, and Geometry in Data Science (TAG-DS 2025).

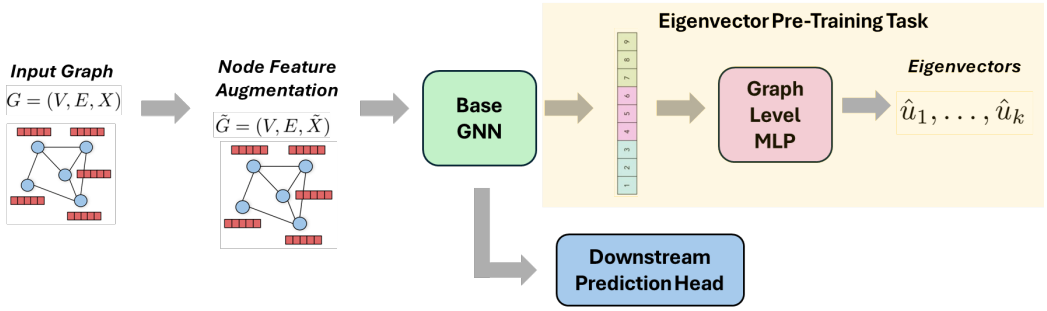

Figure 1: Overview of the LELM pre-training pipeline. Here, “Base GNN" and “Downstream Prediction Head" can be any user-defined model architecture.

excessively deep networks, and it augments pre-training data with positional features to overcome expressivity limits of GNNs. LELM is highly flexible: it can be used to pre-train any GNN to improve downstream performance across all kinds of graph-based datasets, and it can be used either as an independent pre-training method or as a plug-in addition to existing graph pre-training pipelines.

Our main contributions are as follows:

1. We propose LELM, a Laplacian eigenvector-based pre-training module for GNNs and GFMs.
2. We introduce a novel MLP head that enables GNNs to capture large-scale structure. Furthermore, we propose a set of augmented node features based on the graph diffusion operator.
3. We empirically demonstrate that LELM provides performance improvements over baseline models.

2 Background

Given a graph $\mathcal{G} = (V, E)$ with unnormalized adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$, the unnormalized Laplacian $\mathbf{L}$ of a graph $\mathcal{G}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

Let $\lambda_1, \lambda_2, \ldots, \lambda_k$ denote the k lowest eigenvalues of $\mathbf{L}$ in nondecreasing order. Let $\psi_1, \psi_2, \ldots \psi_k$ denote the corresponding eigenvectors, such that we have:

$$\mathbf{L}\boldsymbol{\psi}_i = \lambda_i \boldsymbol{\psi}_i$$

The low-frequency graph Laplacian eigenvectors have been used for positional encodings, spectral GNNs, spectral clustering, and to generate provably minimal graph cuts. We provide examples and references for each of these applications in Section B.

3 Method

The LELM pre-training framework consists of three primary components:

- **Node feature augmentation:** We provide initial features based on the graph diffusion operator. These embeddings provide the base GNN model with additional structural context which is useful for the pre-training task as well as for downstream structure-aware tasks.
- **Graph-level MLP:** We pass a graph-level aggregated representation from a base GNN into our prediction MLP head. The graph-level MLP overcomes the challenge of oversmoothing by allowing the base GNN to learn long-range node interactions without requiring an excessively deep message passing network.
- **Eigenvector prediction:** During pre-training, we task the model to predict the k lowest-frequency eigenvectors of the graph Laplacian.

Together, these components allow the base GNN to learn local and global graph structure while overcoming intrinsic expressivity limitations, making LELM a robust pre-training framework for structure-aware downstream tasks. A visual overview of our pipeline can be seen in Figure 1, and a detailed algorithmic overview of our pipeline can be found in Section C.

3.1 Node feature augmentation

We propose two kinds of embeddings: **(1) wavelet positional embeddings**, which encode relative positional information between nodes, and **(2) diffused dirac embeddings**, which encode local connectivity structures around each node. Both embeddings use the graph diffusion operator, and capture local aggregate information on each node. We provide more details as well as theoretical guarantees in Section G.

3.2 Graph-level MLP

Once we perform node augmentation, we pass the augmented graph as input into a base GNN model and graph-level MLP to output predicted eigenvectors.

Base GNN: The base GNN model takes in a graph with augmented node features and generates learned node representations via neighborhood message passing and update steps. Any GNN architecture may be selected as the base model to fit the needs of the dataset and downstream application.

Graph-level MLP: We concatenate the node-wise output of the base GNN model to form a graph-level aggregated representation. We then pass the aggregated vector through an MLP model to produce the low-frequency Laplacian eigenvectors.

Eigenvector-learning methods for other applications (such as spectral clustering) use a node-wise MLP head, processing each node's eigencoordinates independently based on their learned hidden embedding (Shaham et al., 2018; Dwivedi et al., 2021; Cantürk et al., 2023). Such methods fail to address oversmoothing as a node-wise MLP cannot learn long-range interactions; instead, one must use several layers of message passing within the base GNN (Cantürk et al., 2023).

3.3 Eigenvector prediction

During pre-training, the model output aims to minimize a weighted sum of two loss functions: **(1) eigenvector loss** and **(2) energy loss**.

To ensure the model does not output k copies of the trivial eigenvector, we impose orthogonality on the final outputs of the model via QR decomposition, as proposed by Shaham et al. (2018). Let each $\hat{\mathbf{u}}_i$ denote the ith predicted eigenvector via LELM after orthogonality has been imposed.

Energy loss, used in Shaham et al. (2018); Dwivedi et al. (2021); Ma & Zhan (2023), computes the sum of Rayleigh quotients:

$$\mathcal{L}_{energy} = \frac{1}{k}\sum_{i=1}^{k} \hat{\mathbf{u}}_i^\top \mathbf{L}\hat{\mathbf{u}}_i$$

Energy loss is minimized when the predicted eigenvectors span the same subspace as the ground-truth eigenvectors; to enforce a strict ordering and basis on the predicted eigenvectors, we additionally impose **eigenvector loss**, used in Mishne et al. (2019):

$$\mathcal{L}_{eigvec} = \frac{1}{k}\sum_{i=1}^{k} \|\mathbf{L}\hat{\mathbf{u}}_i - \lambda_i \hat{\mathbf{u}}_i\|$$

We discuss further intuition behind our loss functions in Section D and provide proofs for necessary sign and basis invariances in Section F.

4 Experimental Results

To evaluate the effectiveness of LELM, we conduct various experiments on real-world datasets.

Comparison against baseline models: We pre-train a standard Graph Isomorphism Network (GIN) (Xu et al., 2019) and GPS (Rampášek et al., 2022), a graph transformer, using LELM. Once the model has been pre-trained, we replace the graph-level MLP head with a downstream prediction MLP

Table 1: Test MAE (↓) performance comparison on ZINC (single metric) and QM9 (first seven target properties).

	ZINC full	ZINC subset	QM9						
Model	Penalized $\log p$	Penalized $\log p$	μ	α	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	Δ_ε	$\{R^2\}$	ZPVE
GIN + LELM	0.130	0.353	0.484	**0.489**	**0.00353**	**0.00371**	0.00513	**28.103**	**0.00048**
GIN (baseline)	0.228	0.438	0.472	1.132	0.00386	0.00399	0.00562	50.909	0.00240
GPS + LELM	**0.104**	**0.210**	0.502	0.592	0.00372	0.00408	**0.00511**	33.606	0.00178
GPS (baseline)	0.150	0.358	**0.413**	0.718	0.00434	0.00442	0.00592	80.503	0.00111

Table 2: Test MAE (↓) performance comparison when pre-training a GIN on ZINC to predict alternative structural training targets.

Alternative training targets	ZINC full	ZINC subset
LELM	**0.130**	**0.353**
Node degree	0.238	0.471
Local clustering coefficient	1.493	1.551
Cycle counting	0.285	0.420
Lap Eigenvalues	0.250	0.520

and fine-tune model weights. We evaluate our pre-training framework on three molecular datasets ZINC, ZINC-12k (Sterling & Irwin, 2015) and QM9 (Ramakrishnan et al., 2014). For each of these models, we compare LELM against the same GNN model without pre-training. For both models, pre-training improves performance for all but one of the downstream targets. We record results of our experiments in Table 1.

Comparison against alternative pre-training targets: We compare LELM to alternative structure-based pre-training targets, including node degree, local clustering coefficient, cycle counting, and Laplacian eigenvalues. For each alternative target, we pre-train a standard GIN (with our default settings) using L1 loss. Results are recorded in Table 2. Here, we see that LELM outperforms alternative structure-based pre-training targets, indicating that the Laplacian eigenvectors are a better pre-training target than many other natural choices for graph targets.

Ablation on MLP head: We pre-train both the GIN and GPS using LELM, replacing the graph-level MLP with a standard MLP which acts on each node's embeddings individually. Results can be seen in Table 3; for both the GIN and GPS, pre-training is more effective with the graph-level MLP.

Table 3: Test MAE (↓) performance comparison on ZINC when replacing the graph-level MLP with a basic node-wise MLP during pre-training.

Model	ZINC full	ZINC subset
GIN + LELM	0.130	0.353
GIN + LELM (no graph-level MLP)	0.152	0.435
GPS + LELM	**0.104**	**0.210**
GPS + LELM (no graph-level MLP)	0.126	0.261

Enhancing existing pre-training methods: To demonstrate the flexibility of our method, we complement an existing pre-training method with LELM and demonstrate performance improvements in the majority of downstream tasks. Full experimental details and results can be found in Section I.

5 Limitations and Future Work

There are several promising future directions toward improving the LELM pre-training framework. First, we have demonstrated the effectiveness of the framework for pre-training and fine-tuning a GNN on the same dataset, or on domain-related datasets, but we have yet to explore the effectiveness of LELM as a *transfer learning* framework. A future direction could be exploring the viability of LELM when pre-training on synthetic graphs or on cross-domain datasets.

References

Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision. In *International conference on machine learning*, pp. 524–533. PMLR, 2019.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. *arXiv preprint arXiv:2303.01028*, 2023a.

Deyu Bo, Xiao Wang, Yang Liu, Yuan Fang, Yawen Li, and Chuan Shi. A survey on spectral graph neural networks. *arXiv preprint arXiv:2302.05631*, 2023b.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. In *Forty-first International Conference on Machine Learning*, 2023.

Ziyu Chen, Yingzhou Li, and Xiuyuan Cheng. Specnet2: Orthogonalization-free spectral embedding by neural networks. In *Mathematical and Scientific Machine Learning*, pp. 33–48. PMLR, 2022.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2021.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

Mingguo He, Zhewei Wei, and Ji-Rong Wen. Convolutional neural networks on graphs with chebyshev approximation, revisited. *Advances in neural information processing systems*, 35: 7264–7276, 2022.

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*.

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*, 2019.

Dasol Hwang, Jinyoung Park, Sunyoung Kwon, KyungMin Kim, Jung-Woo Ha, and Hyunwoo J Kim. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. *Advances in neural information processing systems*, 33:10294–10305, 2020.

Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*, pp. 222–231. IEEE, 2020.

Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems*, 35:2268–2281, 2022.

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

Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2016.

Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1901.01484*, 2019.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip S Yu. Graph self-supervised learning: A survey. *IEEE transactions on knowledge and data engineering*, 35(6): 5879–5900, 2022.

George Ma, Yifei Wang, and Yisen Wang. Laplacian canonization: A minimalist approach to sign and basis invariant spectral embedding. *Advances in Neural Information Processing Systems*, 36: 11296–11337, 2023.

Yixuan Ma and Kun Zhan. Self-contrastive graph diffusion network. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 3857–3865, 2023.

Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52 (6):1686–1697, 2012.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451, 2018.

Gal Mishne, Uri Shaham, Alexander Cloninger, and Israel Cohen. Diffusion nets. *Applied and Computational Harmonic Analysis*, 47(2):259–285, 2019.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Zhen Peng, Yixiang Dong, Minnan Luo, Xiao-Ming Wu, and Qinghua Zheng. Self-supervised graph representation learning via global context prediction. *arXiv preprint arXiv:2003.01604*, 2020a.

Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of the web conference 2020*, pp. 259–270, 2020b.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1150–1160, 2020.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

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

Ann M Richard, Richard S Judson, Keith A Houck, Christopher M Grulke, Patra Volarath, Inthirany Thillainadarajah, Chihae Yang, James Rathman, Matthew T Martin, John F Wambaugh, et al. Toxcast chemical landscape: paving the road to 21st century toxicology. *Chemical research in toxicology*, 29(8):1225–1251, 2016.

Uri Shaham, Kelly Stanton, Henry Li, Ronen Basri, Boaz Nadler, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations*, 2018.

Daniel A Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of 37th conference on foundations of computer science*, pp. 96–105. IEEE, 1996.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny. Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of chemical information and modeling*, 56(10):1936–1949, 2016.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*.

Ruoxi Sun, Hanjun Dai, and Adams Wei Yu. Does gnn pretraining help molecular representation? *Advances in Neural Information Processing Systems*, 35:12096–12109, 2022.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*.

Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 889–898, 2017.

Zehong Wang, Zheyuan Liu, Tianyi Ma, Jiazheng Li, Zheyuan Zhang, Xingbo Fu, Yiyang Li, Zhengqing Yuan, Wei Song, Yijun Ma, et al. Graph foundation models: A comprehensive survey. *arXiv preprint arXiv:2505.15116*, 2025.

Yaochen Xie, Zhengyang Wang, and Shuiwang Ji. Noise2same: Optimizing a self-supervised bound for image denoising. *Advances in neural information processing systems*, 33:20320–20330, 2020.

Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):2412–2429, 2022.

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

Pengwei Yan, Kaisong Song, Zhuoren Jiang, Yangyang Kang, Tianqianjin Lin, Changlong Sun, and Xiaozhong Liu. Empowering dual-level graph self-supervised pretraining with motif discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 9223–9231, 2024.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33: 5812–5823, 2020.

A Related works: GNN pre-training methods

Towards the goal of improving graph foundation models, a variety of self-supervised graph pre-training tasks have been proposed. According to the taxonomy provided by Liu et al. (2025); Xie et al. (2022), existing graph pre-training methods can be categorized into two broad categories: *contrastive* and *predictive* methods.

Contrastive methods maximize mutual information between pairs of data views using objectives like Jensen-Shannon estimator (Nowozin et al., 2016) or InfoNCE (Oord et al., 2018). Methods can be categorized by the types of views used: graph-node (Sun et al.; Veličković et al.; Peng et al., 2020b), subgraph-node (Hu et al.; Jiao et al., 2020), and subgraph-subgraph Qiu et al. (2020). Some methods also employ graph augmentation to generate two views (You et al., 2020).

Predictive methods, also referred to as generative methods (Liu et al., 2025), self-generate labels and train to predict these labels. A first class of predictive models uses graph reconstruction, whether by using node/edge masking (Xie et al., 2020; Batson & Royer, 2019; Hu et al.) or using autoencoders (Wang et al., 2017; Kipf & Welling, 2016). A second class of predictive methods are *property prediction* methods, which precompute underlying graph properties as labels. Examples include statistical properties such as k-hop connectivity (Peng et al., 2020a) or topological properties like a meta-path (Hwang et al., 2020). Overall, there are a lack of works on property prediction-based methods, with the majority of predictive pre-training methods falling under the former category of graph reconstruction (Liu et al., 2025). Our method, LELM, is the first property prediction method to use the graph Laplacian eigenvectors as a pre-training target.

B Graph Laplacian eigenvector applications

Provably minimal graph cuts: The second-lowest eigenvector ψ_2, known as the Fiedler vector, can be used to generate a provably "good" cut on a graph; in particular, for some arbitrary threshold $s \in \mathbb{R}$, we can define a Fiedler cut C to be:

$$C = (\{i : \psi_2(i) < s\}, \{i : \psi_2(i) \geq s\})$$

On any bounded-degree n-vertex planar graph, the optimal Fiedler cut has ratio $O(\frac{1}{n})$ (Spielman & Teng, 1996).

Positional encodings: The low-frequency Laplacian eigenvectors naturally encode a global position on the graph. As a result, Laplacian positional encodings (LapPE) have been used as a standard positional encoding for graph transformer models (Dwivedi & Bresson, 2020; Rampášek et al., 2022). In practice, directly using the Laplacian eigenvectors as positional encodings creates sign and basis ambiguity issues, as ψ_i is an eigenvector of $L \iff -\psi_i$ is an eigenvector of L. Approaches to solving this problem include designing an architecture which processes the Laplacian eigenvectors in a sign- and basis-invariant manner (Lim et al., 2022) or defining canonical directions for the eigenvectors (Ma et al., 2023).

Spectral GNNs: A variety of methods, known as spectral graph neural networks, use the Laplacian eigendecomposition to learn filters in signal domain (Bo et al., 2023b). Some methods explicitly compute or approximate the k lowest-frequency Laplacian eigenvectors, learning advanced filters on the corresponding eigenvalues (Bruna et al., 2013; Liao et al., 2019; Bo et al., 2023a). Other methods instead learn polynomial filters on the graph (Defferrard et al., 2016; He et al., 2022), circumventing the expensive process of eigendecomposition by learning a k-degree polynomial function with respect to L, i.e. $p(L) = \theta_0 I + \theta_1 L + \cdots + \theta_k L^k$.

Spectral clustering: The Laplacian eigenvectors have also been used for clustering applications. Given a set of data $x_1, \ldots x_n$, Belkin & Niyogi (2001) construct a weighted graph G with n nodes using a heat kernel. Then to generate a k-dimensional embedding, Belkin & Niyogi (2001) compute the k-lowest eigenvectors $\psi_2, \ldots \psi_{k+1}$ (omitting the trivial eigenvector) of the graph Laplacian and assign data point x_i the embedding $(\psi_2(i), \psi_3(i), \ldots, \psi_{k+1}(i))$. Shaham et al. (2018); Chen et al.

(2022) learn this spectral map using a neural network, allowing for a natural extension of this map to new datapoints.

C Complete algorithms

We provide a detailed overview of the eigenvector pre-training process in Algorithm 1, and an overview of the full pre-training and fine-tuning pipeline in Algorithm 2.

Algorithm 1 Eigenvector Prediction

Input: Graph $G = (V, E)$; augmented node features $\tilde{X} = \{\tilde{x}_j\}$; Base GNN
Output: Output Pre-trained GNN model, k lowest-frequency eigenvectors

for $i <$ Pre-Training Epochs **do**
  $\vec{z}_0, \ldots, \vec{z}_n \leftarrow \textsc{BaseGNN}(G, \tilde{X})$
  $\vec{Z} \leftarrow [\vec{z}_1, \ldots, \vec{z}_n] \in \mathbb{R}^{nd}$
  $\tilde{U} \leftarrow \text{MLP}(\vec{Z})$
  $\hat{U} = \text{QR}(\tilde{U})$
  $\mathit{Loss} = \alpha \cdot \textsc{EnergyLoss}(\hat{U}) + \beta \cdot \textsc{EigvecLoss}(\hat{U})$
  Back-propagate Loss, update model weights
end for
return BaseGNN

Algorithm 2 Structure-Informed Graph Pre-training Framework

Input: Graph $G = (V, E)$; node features $X = \{x_j\}$; training labels Y; untrained Base GNN; untrained Downstream Prediction Head
Output: Trained Base GNN and Downstream Prediction Head

$\tilde{X} \leftarrow \textsc{AugmentFeatures}(G, X)$
$\textsc{BaseGNN} \leftarrow \textsc{EigvecPreTrain}(G, \tilde{X}, \textsc{BaseGNN})$
for $i <$ Fine-tuning Epochs **do**
  $\vec{z}_0, \ldots, \vec{z}_n \leftarrow \textsc{BaseGNN}(G, \tilde{X})$
  $\vec{Z} \leftarrow [\vec{z}_1, \ldots, \vec{z}_n]$
  $\hat{Y} \leftarrow \textsc{DownstreamHead}(\vec{Z})$
  $\mathit{Loss} = \textsc{LossCriterion}(\hat{Y}, Y)$
  Backpropagate Loss, update model weights
end for
return BaseGNN, DownstreamHead

D Loss function

We minimize a weighted sum of two loss functions: **(1) eigenvector loss** and **(2) energy loss**. Both loss functions respect necessary sign and basis invariances of Laplacian eigenvectors; full proofs can be found in F.

Energy loss, used by Shaham et al. (2018); Dwivedi et al. (2021); Ma & Zhan (2023), aims to minimize the sum of Rayleigh quotients:

$$\mathcal{L}_{energy} = \frac{1}{k}\text{Tr}(\hat{U}^\top L\hat{U})$$

This loss function is motivated by the iterative optimization problem following from Courant-Fischer, which states that the eigenvectors of L (and the eigenvectors of any Hermitian matrix) can be equivalently expressed as solutions to the following iterative optimization problem:

$$\psi_k \in \underset{\substack{\|x\|=1 \\ x \perp \psi_1, \ldots, \psi_{k-1}}}{\arg\min} \; x^\top Lx.$$

The term $\frac{x^\top Lx}{x^\top x}$ is known as the Rayleigh quotient; because we normalize our predicted eigenvectors, we simply treat this as $x^\top Lx$.

However, minimizing this loss function only minimizes the *sum* of the first k Rayleigh quotients, meaning the minimizer (subject to orthogonality) is any set of vectors spanning same subspace spanned by the k lowest frequency eigenvectors. For applications in clustering, this is reasonable, as the exact basis in which embeddings are expressed is often irrelevant; however, to require the model to truly predict the k-lowest eigenvectors, we must include a more explicit penalty, such as **eigenvector loss**.

Eigenvector loss, used by Mishne et al. (2019), measures the difference between each $L\hat{u}_i$ and $\lambda_i\hat{u}_i$:

$$\mathcal{L}_{eigvec} = \frac{1}{k}\|(L\hat{U} - \hat{U}\Lambda_k)\|$$

Eigenvector loss enforces both the correct basis and a strict ordering (up to eigenvalue multiplicity) on the predicted eigenvectors. Our final loss function is then:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{energy} + \beta \cdot \mathcal{L}_{eigvec}$$

E Loss function alternative expressions

Eigenvector loss, per-vector form:

$$\mathcal{L}_{eigvec} = \frac{1}{k}\sum_{i=1}^{k}\|L\hat{u}_i - \lambda_i\hat{u}_i\|$$

Eigenvector loss, matrix form:

$$\mathcal{L}_{eigvec} = \frac{1}{k}\|(L\hat{U} - \hat{U}\Lambda_k)\|$$

Energy loss, per-vector form:

$$\mathcal{L}_{energy} = \frac{1}{k}\sum_{i=1}^{k}\hat{u}_i^\top L\hat{u}_i$$

Energy loss, matrix form:

$$\mathcal{L}_{energy} = \frac{1}{k}\text{Tr}(\hat{U}^\top L\hat{U})$$

Energy loss is order-invariant and rotation invariant (see F); for applications in clustering, this is reasonable. However, we would like the model to learn the eigenvectors in their specific order, so we also define **absolute energy loss**, matching the Rayleigh quotient with the ground-truth eigenvalue:

$$\mathcal{L}_{energy\_abs} = \frac{1}{k}\sum_{i=1}^{k}|\hat{u}_i^\top L\hat{u}_i - \lambda_i|$$

This can be written as, in matrix form:

$$\mathcal{L}_{energy\_abs} = \frac{1}{k}\text{Tr}|\hat{U}^\top L\hat{U} - \Lambda_k|$$

In practice, we do not show any results using absolute energy loss, and instead linearly combine energy loss with eigenvector loss to avoid order and rotation invariance. However, absolute energy loss remains an interesting avenue to explore.

E.1 Orthogonality

To ensure the model does not output k copies of the trivial eigenvector, we must give the model orthogonality constraints on the output vectors. There are again two reasonable choices here: **(1) forced orthogonality** and **(2) orthogonality loss**.

Forced orthogonality, used in Shaham et al. (2018), imposes orthogonality on the final outputs of the model via QR decomposition. In other words, if $\hat{U}'$ is the initial output to the model, Q is an $n \times k$ matrix with orthonormal columns, and R is a $k \times k$ upper triangular matrix, then we achieve the final output $\hat{U}$ as such:

$$QR = \hat{U}'$$
$$\hat{U} = Q$$

Orthogonality loss, used in Dwivedi et al. (2021); Ma & Zhan (2023); Mishne et al. (2019) imposes a softer constraint, encouraging orthogonality by penalizing the model for producing pairwise similar vectors. This can be written as:

$$\mathcal{L}_{ortho} = \frac{1}{k}\|\hat{U}^\top \hat{U} - I\|$$

Based on preliminary testing, we found that forced orthogonality improved performance on the eigenvector-learning, and thus use forced orthogonality in all of our experiments.

F Energy and eigenvector losses are sign and basis invariant

F.1 Definition of basis invariance

Consider any eigenspace spanned by ground truth eigenvectors $[\psi_j, \psi_{j+1}, \dots \psi_{j+k-1}] = V$. Also recall that, by Spectral Theorem, we can decompose any vector u into a linear combination of all eigenvectors:

$$u = \sum_{i=1}^{n} c_i \psi_i$$

Then a loss function is basis invariant if any rotation of the projected component $VV^\top u$ does not change the loss incurred by u. In other words, u gets to arbitrarily "choose" with what basis it wants to express its $VV^\top u$ component. Sign invariance is a special case of basis invariance, where changing sign is equivalent to rotating over a one-dimensional subspace (note that this is slightly stronger than the most apparent form of sign invariance, where we would say $\mathcal{L}(u) = \mathcal{L}(-u)$; instead, we can flip any *component* $c_i\psi_i$ of u when decomposed in terms of eigenvectors).

Definition 1 (Basis invariance)**.** *Consider an eigenspace spanned by ground truth eigenvectors $[\psi_j, \psi_{j+1}, \dots \psi_{j+k-1}] = \Psi \in \mathbb{R}^{n \times k}$. Consider an eigenspace rotation R_Ψ defined as such:*

$$R_\Psi = \Psi A \Psi^\top + (I_n - \Psi\Psi^\top), A \in \mathrm{SO}(k)$$

A loss function $\mathcal{L}(u)$ is basis invariant if, for all such Ψ, R_Ψ, $u \in \mathbb{R}^n$, we have:

$$L(u) = L(R_\Psi u)$$

F.2 Proofs

Lemma 1 (Energy loss is basis invariant)**.** *For any R_Ψ and a single eigenvector prediction $u \in \mathbb{R}^n$, we have:*

$$u^\top L u = (R_\Psi u)^\top L (R_\Psi u)$$

Proof. First note that R_Ψ is orthogonal; the set of all R_Ψ describes a subset of $SO(k)$ where only the k basis vectors in Ψ are rotated. Thus, we have $R_\Psi^\top R_\Psi = I$.

In addition, because Ψ is an eigenspace, all columns are eigenvectors with a shared eigenvalue λ. Then we have:

$$R_\Psi L = \Psi A \Psi^\top L + L - \Psi\Psi^\top L = \lambda \Psi A \Psi^\top + L - \lambda \Psi\Psi^\top = L\Psi A\Psi^\top + L - L\Psi\Psi^\top = L R_\Psi$$

Then we have:

$$R_\Psi^\top L R_\Psi = R_\Psi^\top R_\Psi L = L$$

Thus, for any u, we have:

$$u^\top L u = u^\top R_\Psi^\top L R_\Psi u = (R_\Psi u)^\top L (R_\Psi u)$$

□

Lemma 2 (Eigenvector loss is basis invariant)**.** *For any R_Ψ and a single eigenvector prediction $u \in \mathbb{R}^n$ and ground truth eigenvalue λ, we have:*

$$\|Lu - \lambda u\| = \|L(R_\Psi u) - \lambda(R_\Psi u)\|$$

Proof. We know, from our proof above in Lemma 1, that $R_\Psi L = L R_\Psi$. Because $R_\Psi \in \mathrm{SO}(k)$, we have $\|R_\Psi x\| = \|x\|$ for any $x \in \mathbb{R}^n$. Then we have:

$$\|Lu - \lambda u\| = \|R_\Psi(Lu - \lambda u)\|$$
$$\|Lu - \lambda u\| = \|L(R_\Psi u) - \lambda(R_\Psi u)\|$$

□

We have an even stronger statement of invariance for energy loss: **given a predicted set of k orthogonal vectors, rotating the vectors within the same subspace does not impact loss.** In other words, a model trained on energy loss only needs to predict the correct *subspace* of k eigenvectors. This is clearly not true of eigenvector loss. Depending on the application, this kind of invariance can be good or bad.

Lemma 3 (Energy loss is rotation invariant)**.** *Let L be a Laplacian matrix and $V \subseteq \mathbb{R}^n$ be some k-dimensional subspace. Suppose $U = [u_1, u_2, \ldots, u_k], W = [w_1, w_2, \ldots, w_k] \in \mathbb{R}^{n \times k}$ are both orthonormal bases for V. Then we have:*

$$\frac{1}{k}\mathrm{Tr}(U^\top L U) = \frac{1}{k}\mathrm{Tr}(W^\top L W)$$

Proof. Note that $UU^\top = WW^\top$, as they are both orthogonal projectors for the same subspace. Then we have, by the cyclic property of trace:

$$\frac{1}{k}\mathrm{Tr}(U^\top L U) = \frac{1}{k}\mathrm{Tr}(UU^\top L) = \frac{1}{k}\mathrm{Tr}(WW^\top L) = \frac{1}{k}\mathrm{Tr}(W^\top L W)$$

□

G Node feature augmentation

G.1 Graph Diffusion Operator

The diffusion operator P of a graph G is defined as:

$$P = D^{-1}A$$

Each entry P_{ij} represents the probability of starting a random walk at node i and ending at node j after one step. One can also take powers of the diffusion operator, P^t. Each entry of the powered matrix, P_{ij}^t, represents the probability of starting a random walk at node i and ending at node j after t steps

The j^{th} wavelet operator Ψ_j of a graph G is defined as:

$$\Psi_j = P^{2^{j-1}} - P^{2^j}$$
$$\Psi_0 = I - P$$

A wavelet bank, $\mathcal{W}_J$ is a collection of wavelet operators such that:

$$\mathcal{W}_J = \{\Psi_j\}_{0 \le j \le J} \cup P^{2^J}$$

G.2 Node Embeddings

Wavelet positional embeddings encode information about the relative position of each node within the graph. We randomly select two nodes from each graph, i and j, and start dirac signals δ_i, δ_j. We then apply these signals to each wavelet, Ψ_k, in our wavelet bank. The wavelet positional embedding for node m is the m^{th} row of the resulting matrix.

$$w_{m,k} = \Psi_k(m, \cdot) \begin{bmatrix} | & | \\ \delta_i & \delta_j \\ | & | \end{bmatrix}$$
$$w_m = [w_{m,1} \quad \dots \quad w_{m,J}]$$

Diffused dirac embeddings encode information about the connectedness of each node and its neighbors. For each node, m, we apply the m^{th} row of the diffusion matrix P to each wavelet Ψ_k in our wavelet bank. As above, the diffused dirac embedding for node m is the m^{th} row of the resulting matrix.

$$d_{m,k} = \Psi_k(m, \cdot)\, P(m, \cdot)^\top$$
$$d_m = [d_{m,1} \quad \dots \quad d_{m,J}]$$

These node embeddings are unique up to co-spectrality of the graph Laplacian.

Lemma 4 (Uniqueness up to co-spectrality)**.** *Let G_1, G_2 be graphs of size n with Laplacian matrices L_1, L_2 respectively. Let d_m^1, d_m^2 represent the diffused dirac embeddings for each node in G_1, G_2. Then if L_1 and L_2 have different eigenvalues, $\{d_m^1 : m \leq n\} \neq \{d_m^2 : m \leq n\}$*

Proof. Consider the random-walk Laplacian of a graph: $L_{rw} := I - D^{-1}A = I - P$. Moreover, note that $L_{rw} = D^{-1}L$. Observe that

$$\begin{aligned} L_{rw}Dv &= D^{-1}LDv \\ &= D^{-1}U\Lambda U^\top Dv \\ &= Bv \text{ for some diagonalizable matrix } B \text{ with eigenvalues } \lambda_i, \dots, \lambda_n \end{aligned}$$

Where $U = [\psi_1 \quad \dots \quad \psi_n]$, with ψ_i orthonormal eigenvectors of L and Λ is the diagonal matrix of eigenvalues $\lambda_1, \dots, \lambda_n$ of L. Any change to the eigenspectrum of L, clearly results in a change to L_{rw}, and therefore P. Since $\Psi_0 = I - P$, any two graphs with distinct Laplacian eigenspectra will have distinct diffused dirac node embeddings. □

H Alternative structural pre-training targets

Here, we formally define and provide details for the alternative pre-training targets used in section 4.

- **Node degree:** A node-level label representing the degree of each node
- **Local clustering coefficient:** A node-level label computing the local clustering coefficient of each node. For a fixed node u, the coefficient C is given by:

$$C = \frac{2|\{e_{jk} : v_j, v_k \in N_u, e_{jk} \in E\}|}{|N_u|(|N_u| - 1)}.$$

- **RWSE:** A node-level label computing self-walk probabilities at varying step counts for the diffusion operator (Dwivedi et al., 2021). In our experiments, we use step counts from the interval $[2, 22]$.
- **Cycle counting:** A graph-level label computing cycle counts of varying lengths. In our experiments, we count cycles up to length 9.
- **Lap Eigenvalues:** A graph-level label computing the k-lowest Laplacian eigenvalues $\lambda_1, \dots, \lambda_k$. We use the same $k = 6$ as we do with LELM.

Table 4: Test ROC-AUC (%, ↑) performance on 5 molecular prediction tasks when **augmenting an existing pre-training method** on a GIN base model. *Sup.* refers to the original supervised pre-training as implemented by Hu et al. (2019), and *Sup.+* refers to supervised training with LELM. Results for *no pre-training* are taken directly from Sun et al. (2022). All methods are tuned over seven learning rates and averaged over three seeds.

Dataset		BACE	BBBP	Tox21	ToxCast	SIDER
Pretrain method	MLP Head					
ContextPred, Sup.+	Graph-level	79.62 ± 3.63	$\mathbf{70.76 \pm 1.64}$	77.94 ± 0.11	$\mathbf{66.13 \pm 0.34}$	60.05 ± 0.99
ContextPred, Sup.+	Node-wise	75.87 ± 3.11	68.74 ± 1.07	78.86 ± 0.06	63.78 ± 0.32	59.83 ± 0.53
ContextPred, Sup.	-	$\mathbf{84.98 \pm 1.28}$	68.25 ± 0.48	77.44 ± 0.19	64.01 ± 0.81	$\mathbf{62.87 \pm 0.89}$
Masking, Sup.+	Graph-level	80.71 ± 3.84	68.33 ± 0.89	$\mathit{79.09 \pm 0.25}$	$\mathit{65.96 \pm 0.20}$	$\mathit{62.41 \pm 1.77}$
Masking, Sup.+	Node-wise	$\mathit{81.02 \pm 1.67}$	$\mathit{69.94 \pm 1.76}$	$\mathbf{79.33 \pm 0.41}$	65.14 ± 0.44	59.38 ± 1.11
Masking, Sup.	-	75.42 ± 2.64	67.36 ± 4.60	78.33 ± 0.24	64.88 ± 0.82	61.6 ± 1.78
No pre-training	-	75.77 ± 4.29	69.62 ± 1.05	75.52 ± 0.67	63.67 ± 0.32	59.07 ± 1.13

For all alternative structural pre-training tasks, we use the same hyperparameters for GIN as displayed in 5, with no initial features and using a standard MAE loss instead of eigenvector + energy loss. We train on the full ZINC dataset. All structural pre-training targets are normalized to have mean 0 and standard deviation 1 across the entire dataset.

I Enhancing an existing Graph Neural Network pre-training method

I.1 Summary

We augment the existing molecular pre-training methods proposed by Hu et al. (2019) with eigenvector-learning. In particular, Hu et al. (2019) propose node-level pre-training tasks (context prediction and masking) on ZINC15 (Sterling & Irwin, 2015), followed by a graph-level supervised pre-training task on ChEMBL (Mayr et al., 2018; Gaulton et al., 2012). We augment the graph-level supervised pre-training step by adding an additional MLP head to predict eigenvectors, and we evaluate on five downstream datasets based on work by Sun et al. (2022).

Detailed results are shown in Table 4. Eigenvector-learning consistently improves performance for the masking pre-training pipeline, but achieves mixed results on the context prediction pipeline. Notably, performance for the masking pipeline was increased for all five datasets when performing eigenvector pre-training with the graph-level MLP head.

I.2 Learning rate tuning

We keep the majority of the settings from Hu et al. (2019) the same. For downstream fine-tuning, we tune over 7 learning rates for fair comparison according to Sun et al. (2022). We run each method and learning rate over 3 seeds, and select the learning rate based on mean validation accuracy over all learning rates.

I.3 Downstream datasets

We briefly list and cite the five downstream datasets here for reference. The five datasets are the datasets chosen in Sun et al. (2022), and are a subset of the eight primary downstream datasets evaluated in Hu et al. (2019).

- **BACE:** Qualitative binding results Subramanian et al. (2016)
- **BBBP:** Blood-brain barrier penetration Martins et al. (2012)
- **Tox21:** Toxicity data Mayr et al. (2016)
- **Toxcast:** Toxicology measurements Richard et al. (2016)
- **SIDER:** Database of adverse drug reactions (ADR) Kuhn et al. (2016)

J Detailed experimental settings

A complete overview of model hyperparameters and settings can be found in Table 5. Heuristically, the Graph-level MLP head hidden dimension is chosen to be the max # nodes multiplied by the hidden dimension size of the base GNN. We do NOT omit the trivial eigenvector when counting number of eigenvectors predicted.

Table 5: A comprehensive list of all model hyperparameters used during the eigenvector pre-training step. All hyperparameters highlighted in gray are specific to eigenvector-learning, while other listed configs reflect general GNN settings (and are set to match default values in each respective baseline work).

Method	GIN (4)	GPS (4)	GIN pre-training (I)
Pre-training dataset	ZINC-subset (12k), ZINC (250k), QM9 (134k)	ZINC-subset (12k), ZINC (250k), QM9 (134k)	ZINC15 (2M), ChEMBL (456K)
Base architecture	GIN	Transformer/GIN	GIN
# params	33543	157680	2252210
# layers of per-node feature update	3	3	2
# layers of message passing	4	4	5
Hidden dim	60	60	300
Activation fn	ReLU	ReLU	ReLU
Dropout	0.1	0.1	0.2
Batch size	128	128	32
Learning rate	0.001	0.001	0.001
Optimizer	Adam	Adam	Adam
Scheduler	ReduceLROnPlateau patience=5, factor=0.9	ReduceLROnPlateau patience=20, factor=0.5	None -
Pre-Training Epochs	200	100	100
Fine Tuning Epochs	500	150	100
Laplacian norm type	Unnormalized	Unnormalized	Unnormalized
# eigenvectors predicted	6	6	5
Initial features	Diffused dirac + Wavelet pos.	Diffused dirac + Wavelet pos.	Molecule features
MLP head type(s)	Graph-level	Graph-level	Graph-level, per-node
Graph-level MLP max # nodes	40	40	50
MLP head # layers	5	5	1
MLP head hidden dim	2400	2400	N/A
MLP head activation fn	ReLU	ReLU	N/A
Loss function (and coefficient)	2*Eigenvector + 1*energy	2*Eigenvector + 1*energy	0.25 * Eigenvector + 0.05 * ortho
Other features/notes		Removed graphs with less than six nodes during pre-training	

For our ablation experiments with node-wise MLP heads, we keep all settings the same, replacing the graph-level MLP, which has an graph-level input dimension of $60 \cdot 40 = 2400$, with a node-level MLP which has an input dimension and hidden dimension of 60.

For ZINC-12k and ZINC, the test/train splits are provided. The QM9 dataset is randomly split into training, validation, and test sets with a ratio of 0.8/0.1/0.1

K Runtime and compute

All experiments were run using Yale's Grace cluster. Runs used a single GPU each, with at least 11GB of RAM per GPU. The type of GPU varied between A100, V100, A5000, and RTX5000. For all runs using our main pipeline (comparison against baseline models, comparison against alternative pre-training targets, and ablation on MLP head), pre-training and fine-tuning on all downstream tasks took between 12 to 24 hours on ZINC-full, and 36 to 48 hours on QM9. For experiments enhancing an existing pre-training pipeline, the full pre-training process (including both unsupervised and supervised steps) took around 24 hours, and the downstream fine-tuning across all five datasets for a single seed and model variant took around 15 hours.

Across all results that are included in this paper, we estimate a total of 325 GPU hours spent on the main-body experiments, and an additional 400 GPU hours spent on enhancing an existing GNN pre-training method.

The complete experimental process consumed far more GPU hours, as we initially explored basic models' abilities to learn Laplacian eigenvectors, and also experimented with a variety of augmentation features before deciding on the two included in the paper. We estimate an additional 800 GPU hours used for initial experimentation and exploration.

L Visualization of predicted eigenvectors during pre-training

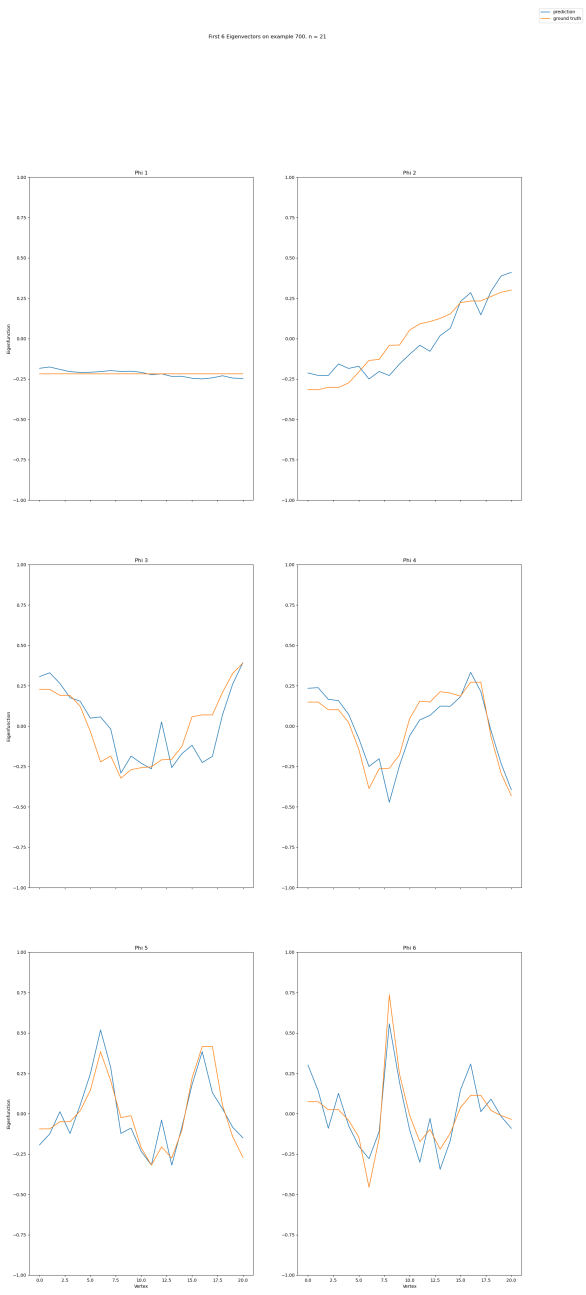

Figure 2: A comparison of predicted eigenvectors (blue) with ground-truth eigenvectors (orange) for a single molecular graph with $n = 21$ nodes. These predictions were produced by the standard GIN model on a validation example from the ZINC dataset. Node indices are sorted in increasing order of $\psi_2(i)$, and the sign orientation of the predicted vectors $\hat{u}_i$ is chosen such that $\hat{u}_i \cdot \psi_i \geq 0$.

TAG-DS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims of the abstract and introduction are that we developed a novel GNN pre-training method that improves performance on structure-based tasks. The empirical results demonstrate the improvements, and the theoretical results justify the design of the framework.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a limitations section, section 5, that discusses the question of whether the framework is suitable for transfer learning.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All proofs in the paper include the full set of assumptions and all proofs are complete and correct. Proofs are included in sections F and G

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Code is provided in supplemental material and data is publicly accessible. Moreover, all experimental settings are provided in supplemental material as well for ease of reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is provided as part of supplementary material and all data is publicly accessible.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental details, including data splits, hyperparameters, optimizer, network depth, etc., are provided in the supplemental material in section J.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide a complete overview of compute resources used and time of execution in the appendix in section K.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Paper does not violate any part of Neurips Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The framework developed in this paper is not tied to any particular application.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: All models are trained publicly accessible molecular property prediction datasets that pose no safety risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification:The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs were not used for core methods in this research. The only use of LLMs was for checking grammar in the paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.