Graph Positional and Structural Encoder

Semih Cantürk [*1 2] Renming Liu [*3] Olivier Lapointe-Gagné [1 2] Vincent Létourneau [1 2] Guy Wolf [1 2]
Dominique Beaini [1 2 4] Ladislav Rampášek [5]

Abstract

Positional and structural encodings (PSE) enable better identifiability of nodes within a graph, rendering them essential tools for empowering modern GNNs, and in particular graph Transformers. However, designing PSEs that work optimally for all graph prediction tasks is a challenging and unsolved problem. Here, we present the Graph Positional and Structural Encoder (GPSE), the first-ever graph encoder designed to capture rich PSE *representations* for augmenting any GNN. GPSE learns an efficient common latent representation for multiple PSEs, and is highly transferable: The encoder trained on a particular graph dataset can be used effectively on datasets drawn from markedly different distributions and modalities. We show that across a wide range of benchmarks, GPSE-enhanced models can significantly outperform those that employ explicitly computed PSEs, and at least match their performance in others. Our results pave the way for the development of foundational pre-trained graph encoders for extracting positional and structural information, and highlight their potential as a more powerful and efficient alternative to explicitly computed PSEs and existing self-supervised pre-training approaches. Our framework and pre-trained models are publicly available[1]. For convenience, GPSE has also been integrated into the PyG library to facilitate downstream applications.

[*]Equal contribution [1]DIRO, Université de Montréal, Montréal, Canada [2]Mila – Quebec AI Institute, Montréal, Canada [3]Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, United States [4]Valence Labs, Montréal, Canada [5]Isomorphic Labs, London, UK. Correspondence to: Semih Cantürk <semih.canturk@mila.quebec>, Renming Liu <liurenmi@msu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]https://github.com/G-Taxonomy-Workgroup/GPSE

1. Introduction

Graph neural networks (GNN) (Scarselli et al., 2009) are the dominant paradigm in graph representation learning (Hamilton et al., 2017b; Bronstein et al., 2021), spanning diverse applications across many domains in biomedicine (Yi et al., 2022), molecular chemistry (Xia et al., 2022), and more (Dwivedi et al., 2022a; Hu et al., 2020a; 2021; Liu et al., 2022). For most of its relatively short history, GNN algorithms were developed within the message-passing neural network (MPNN) framework (Gilmer et al., 2017), where vertices exchange internal states within their neighborhoods defined by the typically sparse graph structure. This standard MPNN framework has several fundamental limits, such as the 1-WL bounded expressiveness (Xu et al., 2019; Morris et al., 2019), under-reaching (Barceló et al., 2020), and over-squashing (Alon & Yahav, 2021; Topping et al., 2022). Leveraging the success of the Transformer model in natural language processing (Vaswani et al., 2017), graph Transformer (GT) models were developed as a new paradigm for GNNs to address the above limitations by attending to all node pairs in a graph (Dwivedi & Bresson, 2021). This full attention inevitably discards the inductive biases related to the graph structure (Battaglia et al., 2018), which MPNNs leverage well to excel. Consequently, positional and structural encodings (PSE) have played the quintessential role in reintroducing such inductive biases, leading to the remarkable success of GTs (Rampášek et al., 2022; Dwivedi & Bresson, 2021; Chen et al., 2022; Ying et al., 2021).

While many different types of hand-crafted PSEs have been proposed in the literature and used by various GT models, there is no *one-size-fits-all PSE* that performs optimally for all tasks. For example, random walk encodings are more effective for molecular property prediction tasks (Rampášek et al., 2022; Dwivedi et al., 2022b). Conversely, graph Laplacian eigenvectors are more useful for tasks involving long-range dependencies (Dwivedi et al., 2022c;a). Moreover, naively stacking different PSEs together does not yield the expected gains. As a result, researchers have to rely on a combination of heuristics, trial-and-error and domain know-how to select the single best encoding for their respective tasks. Developing a universal encoding that combines the benefits of diverse hand-crafted PSEs is thus fundamental to bringing the most performance out of existing GT models.

Here, we present GPSE, an MPNN that is trained to extract graph encodings as latent representations of diverse PSEs. Once trained on a pre-training graph dataset by learning to reconstruct different PSEs using only the graph structures, GPSE can then extract PSE representations from *any* graph dataset to augment GT models. However, designing an MPNN that extracts PSE representations effectively poses a fundamental challenge: *Ensuring that the MPNN suffices to capture properties necessitated by all target PSEs.* First, encodings derived from random walks require beyond 1-WL expressivity (Li et al., 2020), while standard MPNNs are known to be bound by 1-WL (Xu et al., 2019; Morris et al., 2019). Second, graph Laplacian eigenvectors, especially those associated with low frequencies, need access to a global view of the graph structure, which simple MPNNs fail to capture (Fürer, 2010). Furthermore, naive solutions to obtain global structural information, such as stacking more MPNN layers, suffer from well-known issues of over-smoothing and over-squashing (Alon & Yahav, 2021; Li et al., 2019). Through careful architectural design, we mitigate the aforementioned pitfalls and successfully achieve the goal of extracting rich representations from diverse PSEs.

GPSE represents a leap forward in building foundational graph encoders as a one-stop shop for extracting general-purpose PSEs from any graph, alleviating the burden of trial-and-error-based feature engineering associated with conventional PSEs. By demonstrating that GPSE is computationally efficient and highly performant on a large variety of tasks, we usher in a new paradigm in graph learning without manual PSE engineering and open up exciting opportunities toward more powerful PSE extractors.

We summarize our main contributions as follows.

1. We propose GPSE, the first attempt at training a foundation graph encoder that extracts rich *positional and structural representations* solely from graph structures, which can be applied to any MPNN or GT model as a replacement for explicitly constructed PSEs.
2. We show that GPSE provides significant performance improvements over traditional hand-crafted PSEs across a variety of benchmarks.
3. Through extensive experiments, we demonstrate that GPSE is highly *transferable* across graphs of different sizes, connectivity patterns, and modalities.

1.1. Related work

Several approaches have been proposed to overcome the aforementioned limitations of standard MPNNs: Morris et al. (2019) propose higher-order MPNNs, Gutteridge et al. (2023) and Barbero et al. (2024) optimize information flow on graphs through graph rewiring, while Ding et al. (2024) draw inspiration from deep state-space models and RNNs.

Positional encodings were originally implemented as a series of sinusoidal functions in the Transformer model to capture the ordinal position of words in a sentence (Vaswani et al., 2017). However, capturing the positions of nodes in a graph is harder, as nodes of a graph lack such a canonical ordering. Many recent works on graph Transformers (GT) thus use the graph Laplacian eigenvectors as the positional encodings (Rampášek et al., 2022; Kreuzer et al., 2021), which are direct analogues to the sinusoids in Euclidean space (Spielman, 2012). Other methods for encoding positional information include electrostatic potential encodings (Kreuzer et al., 2021), shortest-path distances (Ying et al., 2021), and tree-based encodings (Shiv & Quirk, 2019). *Structural encodings*, on the other hand, have been developed to encode rich local and global connectivity patterns on graph-structured data. The random walk encoding, for example, has shown great success when used with GTs, particularly on small molecular graph benchmarks (Rampášek et al., 2022; Dwivedi & Bresson, 2021; Dwivedi et al., 2022b). Other notable structural encodings include the heat kernel (Kreuzer et al., 2021; Mialon et al., 2021), subgraphs (Bouritsas et al., 2022; Zhao et al., 2022; Chen et al., 2022), and node degree centralities (Ying et al., 2021).

PSEs are also useful for MPNNs, besides GTs. They can be directly used as additional node features for an MPNN (Dwivedi et al., 2022a; Lim et al., 2022; Wang et al., 2022b). Other works designed approaches to process the PSEs separately from the original node features. For example, LSPE (Dwivedi et al., 2022b) processes the PSEs with a separate channel and applies an auxiliary loss. SignNet and BasisNet explicitly design components in the prediction model to handle the graph Laplacian eigenvectors, aiming to resolve the sign- and basis-ambiguity issues (Lim et al., 2022). Despite this great amount of work in developing methods to better utilize hand-crafted PSEs, it is still unclear how to systematically encode information from multiple types of PSEs to effectively augment GNNs for diverse applications. GPSE thus represents the first of its kind in an attempt to tackle this fundamental problem.

2. Methods

Our core idea is to train an MPNN as a graph encoder to extract rich positional and structural representations of any query graph based *solely* on its graph structure (Figure 1A). To achieve this, we design a collection of PSEs encompassing a broad range of encodings and use them as self-supervision to train the encoder via reconstruction (Figure 1B). Once the encoder is trained, it can then be used in inference mode to extract PSE representations for augmenting any downstream dataset (Figure 1C).

For downstream tasks, we primarily use the powerful graph Transformer model GPS (Rampášek et al., 2022) that lever-

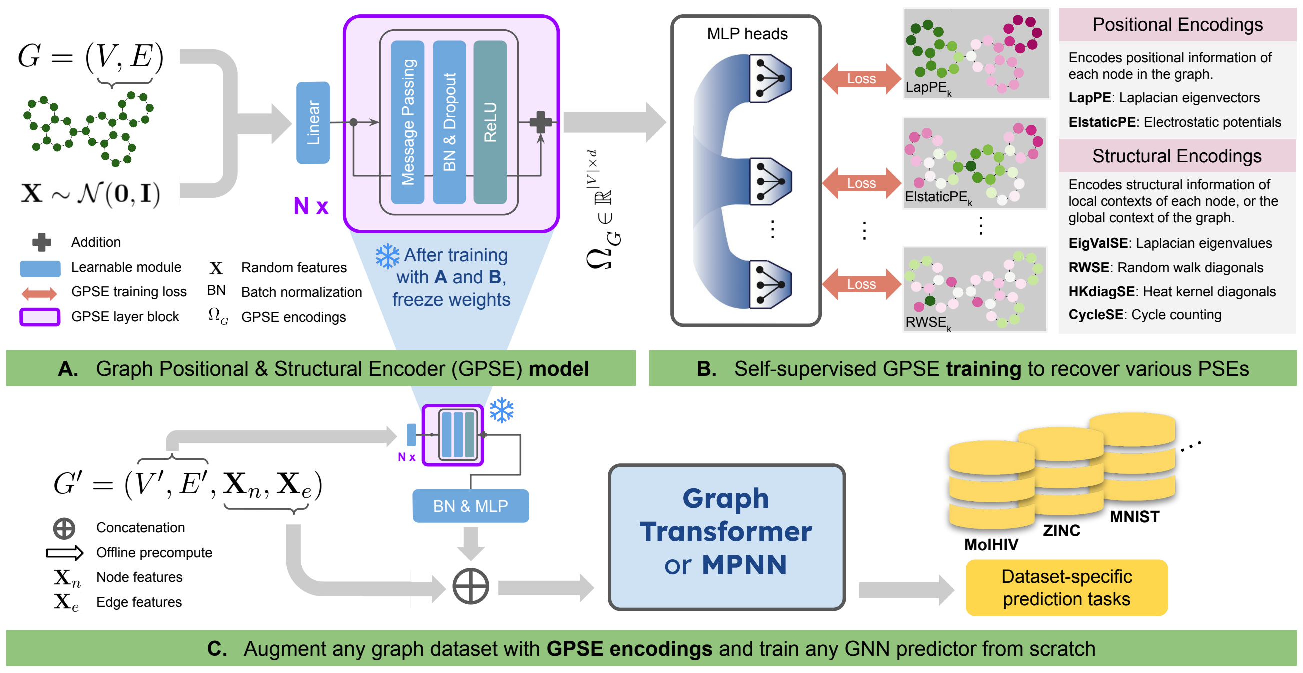

$G = (V, E)$
$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
Linear
Message Passing
BN & Dropout
ReLU
N x
After training with A and B, freeze weights
$\Omega_G \in \mathbb{R}^{|V| \times d}$
Addition
Learnable module
GPSE training loss
GPSE layer block
X Random features
BN Batch normalization
Ω_G GPSE encodings
MLP heads
Loss
LapPE
ElstaticPE
RWSE
Positional Encodings
Encodes positional information of each node in the graph.
LapPE: Laplacian eigenvectors
ElstaticPE: Electrostatic potentials
Structural Encodings
Encodes structural information of local contexts of each node, or the global context of the graph.
EigValSE: Laplacian eigenvalues
RWSE: Random walk diagonals
HKdiagSE: Heat kernel diagonals
CycleSE: Cycle counting
A. Graph Positional & Structural Encoder (GPSE) model
B. Self-supervised GPSE training to recover various PSEs
$G' = (V', E', \mathbf{X}_n, \mathbf{X}_e)$
Concatenation
Offline precompute
$\mathbf{X}_n$ Node features
$\mathbf{X}_e$ Edge features
BN & MLP
Graph Transformer or MPNN
MolHIV
ZINC
MNIST
Dataset-specific prediction tasks
C. Augment any graph dataset with GPSE encodings and train any GNN predictor from scratch

Figure 1. Overview of Graph Positional and Structural Encoder (GPSE) training and application.

ages the advantages of both the inductive bias of the local message passing (Battaglia et al., 2018) and the expressiveness of the global attention (Vaswani et al., 2017). As it has previously attained SOTA results on a variety of benchmarks using hand-crafted PSEs, GPS is a natural baseline model to demonstrate the effectiveness of GPSE. We also validate GPSE for other graph Transformers & MPNNs in our experiments, and thus show that utility of GPSE is not bound to any particular architecture.

2.1. Self-supervision via positional and structural encodings (PSE)

We design a diverse collection of six PSEs for GPSE to learn against, including the Laplacian eigenvectors (4) and eigenvalues (4), the electrostatic positional encodings (7), the random walk structural encodings (20), the heat kernel structural encodings (20), and the cycle counting graph encodings (7). In short, positional encodings inform the relative position of each node in the graph, while structural encodings describe the local connectivity patterns around a node (Figure 1B). See Appendix A for precise mathematical definitions of all PSEs used in GPSE training.

2.2. GPSE architecture

At a high level, our GPSE model is an MPNN consisting of stacked graph convolution blocks with residual gating, and skip-connections in-between. The mathematical formulation of the architecture can be found in Appendix B.1. We illustrate below that this careful design enables GPSE to excel at learning powerful PSE representations. Particularly, we present different architectural choices and *how* they remedy the two key challenges of *over-smoothing* and *over-squashing* (see Appendix C for technical details).

Evidence suggests that over-smoothing and over-squashing in graph networks relate to the graph's curvature (Topping et al., 2022; Nguyen et al., 2022; Giraldo et al., 2022). Formulations of graph curvature (Forman, 2003; Ollivier, 2009; Sreejith et al., 2016; Topping et al., 2022) aim to encode how a node's neighborhood looks like a clique (positive curvature), a grid (zero curvature), or a tree (negative curvature). A clique's high connectivity leads to rapid smoothing, while a tree's exponentially increasing size of k-hop neighborhood causes over-squashing. These phenomena are in competition, and negating both is impossible in an MPNN using graph rewiring (architectures using modified adjacency for node aggregation). However, there seems to be a *sweet spot* where the two effects are not minimized by themselves, but their sum is minimized. This minimum is sought in Giraldo et al. (2022) and Nguyen et al. (2022). Some of our following choices of architecture are justified by this search of a sweet spot in the smoothing-squashing trade-off.

Deep GNN As several of the target PSEs, such as the Laplacian eigenvectors, require having a global view of the query graph, it is crucial for the encoder to capture long-range dependencies accurately. To accomplish this, we need to use an unconventionally deep MPNN with 20 layers. However, if a graph network suffers from over-smoothing, having this many layers will result in approximately uniform node features (Oono & Suzuki, 2020; Li et al., 2019).

Residual connections & gating mechanism A first attempt at reducing the smoothing is to exploit the proven ability of residual connections in reducing over-smoothing (Li et al., 2019). Using a gating mechanism in aggregation helps reduce the over-smoothing even further. Indeed, gating allows the network to reduce the weight of some edges and in the limit effectively re-wire the graph by completely or partially ignoring some edges. We argue that it is possible for gating to act as a graph sparsification device, which decreases the graph curvature and have been shown to alleviate over-smoothing (Giraldo et al., 2022; Rong et al., 2020).

Virtual node In addition, we use a virtual node (VN) (Gilmer et al., 2017) to enable global message passing; as the virtual node has access to the states of all nodes, it allows for (a) better representation of graph-level information and (b) faster propagation of information between nodes that are further apart, and thus faster convergence of states. In technical terms, adding the virtual node drastically increases the connectivity of the graph and in turn its curvature (Appendix C, Prop. C.4), and consequently decreases over-squashing. Alternatively, one can see that the Cheeger constant (another measure of *bottleneckness* (Topping et al., 2022)) of the graph increases after adding the virtual node.

Random node features One critical question is whether an MPNN is expressive enough to learn all the target PSEs. In particular, some PSEs, such as the Laplacian eigenvalues, may require distinguishability beyond 1-WL (Fürer, 2010). Despite the known 1-WL expressiveness limitation of a standard MPNN when using constant node features (Xu et al., 2019; Morris et al., 2019), random node features can help MPNNs surpass 1-WL expressiveness (Sato et al., 2021; Abboud et al., 2021; Kanatsoulis & Ribeiro, 2022). Thus, we base our encoder architecture on an MPNN coupled with random input node features, as shown in Figure 1A.

We empirically validate that together, the above architectural design choices lead to an effective graph encoder that finds the balance between smoothing and squashing (§3.5), and even has an elevated expressiveness due to random features (§3.3). A detailed ablation study highlighting the importance of our architectural choices is available in Table H.1.

2.3. Training GPSE

Given a query graph structure $G = (V, E)$, we first generate a k-dimensional feature from a standard normal distribution for each node, $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is then passed through the GPSE model to extract the final representations (Figure 1A). The representations are then decoded into the target PSEs using multiple independent MLP heads, one per PSE (Figure 1B). We compute the reconstruction loss based on the sum of ℓ_1 and cosine similarity losses (Appendix B.2), and optimize GPSE by minimizing this reconstruction loss. Details about hyperparameters can be found in Table B.4.

Loss function The aforementioned combination of ℓ_1 and cosine similarity losses ensures that the model captures both (1) the *direction* of a particular PSE as a signal on the graph (via cosine similarity loss), and (2) the *magnitude* of the PSE (via ℓ_1 loss). From a graph signal processing perspective, both types of information are crucial to describe the characteristics of PSEs in the form of a graph signal.

Training dataset PCQM4Mv2 (Hu et al., 2021) is a typical choice of pre-training dataset for molecular tasks. However, since GPSE only extracts features from graph structures (e.g., methane, CH4, would be treated as the same graph as silane, SiH4), the amount of training samples reduces to 273,920 after extracting unique graphs. Instead, we train GPSE with MolPCBA (Hu et al., 2020a) with 323,555 unique molecular graphs and an average number of 25 nodes. We randomly select 5% validation and 5% testing data fixed across runs, and use the remaining data for training GPSE. An ablation study on training datasets considered is available in Table H.5.

3. Experiments

GPSE successfully predicts a wide range of target PSEs The self-supervised goal of GPSE is to learn graph representations from which it is possible to recover predefined positional and structural encodings. For each PSE type, we quantify prediction performance in terms of the coefficient of determination (R^2) scores, as presented in Table 1. When trained on a 5% (16,177) subset of MolPCBA molecular graphs, GPSE achieves 0.9790 average test R^2 score across the 6 PSEs. Further, we show that test performance improves asymptotically as the number of training samples increases (§3.5), achieving 0.9979 average test R^2 when trained on 90% (291,199) of unique MolPCBA graphs. These results demonstrate the ability of GPSE to extract rich positional and structural information from a query graph, as well as its ability to learn from increasing amount of data.

Table 1. Held-out PSE prediction performance of GPSE on 5% MolPCBA.

PSE	$R^2 \uparrow$
ElstaticPE	0.964
LapPE	0.973
RWSE	0.984
HKdiagSE	0.981
EigValSE	0.982
CycleSE	0.977
Overall	0.979

3.1. Enhancing performance on molecular graph data

In these experiments, we demonstrate that GPSE provides more performance improvements over traditional PSEs for a wide range of GNN models. Additionally, we show competitive performance achieved by GPSE against the complementary self-supervised learning (SSL) pre-training approaches.

GPSE-augmented GPS is highly competitive on molecular graph benchmarks We compare the performance of the GPS model augmented with our GPSE encodings versus the same model using (a) no PSE, (b) random fea-

Table 2. Performance in four molecular property prediction tasks, averaged over 10 seeds.

	ZINC (subset) MAE ↓	**PCQM4Mv2** (subset) MAE ↓	**MolHIV** AUROC ↑	**MolPCBA** AP ↑
GCN (Kipf & Welling, 2017)	0.3670 ± 0.0110	–	0.7599 ± 0.0119	0.2424 ± 0.0034
GIN (Xu et al., 2019)	0.5260 ± 0.0510	–	0.7707 ± 0.0149	0.2703 ± 0.0023
CIN (Bodnar et al., 2021)	0.0790 ± 0.0060	–	**0.8094 ± 0.0057**	–
CRaWl (Toenshoff et al., 2021)	0.0850 ± 0.0040	–	–	**0.2986 ± 0.0025**
PR-MPNN$_{\text{SIM}}$ (Qian et al., 2024)	0.0850 ± 0.0020	–	0.7950 ± 0.0090	–
K-ST SAT+RWSE (Chen et al., 2022)	0.1020 ± 0.0050	–	–	–
K-SG SAT+RWSE (Chen et al., 2022)	0.0940 ± 0.0080	–	–	–
K-ST SAT+**GPSE**	0.0830 ± 0.0023	–	–	–
K-SG SAT+**GPSE**	0.0873 ± 0.0008	–	–	–
GPS+rand	0.8766 ± 0.0107	0.4768 ± 0.0171	0.6210 ± 0.0444	0.0753 ± 0.0045
GPS+none	0.1182 ± 0.0049	0.1329 ± 0.0030	0.7798 ± 0.0077	0.2869 ± 0.0012
GPS+LapPE	0.1078 ± 0.0084	0.1267 ± 0.0004	0.7736 ± 0.0097	0.2939 ± 0.0016
GPS+RWSE	0.0700 ± 0.0040	0.1230 ± 0.0008	0.7880 ± 0.0101	0.2907 ± 0.0028
GPS+LapPE+RWSE	0.0822 ± 0.0040	0.1273 ± 0.0006	0.7719 ± 0.0129	0.2854 ± 0.0029
GPS+AllPSE	0.0734 ± 0.0030	0.1254 ± 0.0011	0.7645 ± 0.0236	0.2826 ± 0.0001
GPS+**GPSE**	**0.0648 ± 0.0030**	**0.1196 ± 0.0004**	0.7815 ± 0.0133	0.2911 ± 0.0036

tures as PSE, (c) LapPE and RWSE, and (d) concatenation of PSEs from §2.1 on four common molecular property prediction benchmarks (Dwivedi et al., 2022a; Hu et al., 2020a; 2021). For ZINC (Gómez-Bombarelli et al., 2018) and PCQM4Mv2 (Hu et al., 2020a), we use their subset versions following Dwivedi et al. (2022a) and Rampášek et al. (2022), respectively.

We first highlight that GPSE-augmented GPS achieves a remarkable 0.0648 MAE on ZINC (Table 2), not only significantly outperforming other PSEs but even challenging SOTA results. Similarly, GPS+GPSE also achieves the best result on PCQM4Mv2 amongst models that (a) are not ensemble methods and (b) do not have access to 3D information. These two strategies are highly engineering- and domain-oriented, thus do not reflect their utilities on general graph learning tasks, which we aim to demonstrate instead.

Moreover, we note that GPSE always performs better than, or at least on par with, standard PSEs, while concatenating multiple PSEs (AllPSE/LapPE+RWSE) always leads to worse performance than using individual PSEs. On ZINC & PCQM4Mv2, GPSE improves results comfortably beyond standard deviation, while in the worst case (e.g., MolHIV) it recovers the best PSE result, with differences well within a standard deviation. We discuss why GPSE works better on some datasets than others in Appendix I.

GPSE as a universal PSE augmentation The utility of GPSE encodings is not specific to GPS. In Table 2, we show that GPSE also significantly enhances two variants of SAT (Chen et al., 2022) on ZINC. Additionally, we show that augmenting different MPNN methods and the graph Transformer with GPSE universally results in remarkable improvements on ZINC: 56.24% reduction in test MAE on average compared to baselines that do not make use of any PSE, outperforming all other PSEs and their combinations (Table 3). Notably, concatenating LapPE and RWSE does not yield any benefit beyond using either one of them, demonstrating the intricacy of effectively leveraging information from multiple PSEs. We additionally perform a similar set of experiments on PCQM4Mv2 (Table H.2), and obtain the same improvements over explicitly computed PSEs, validating the success of GPSE further.

Feature augmentation using GPSE vs. SSL pre-training Our GPSE feature augmentation is related to self-supervised learning (SSL) pre-training approaches (Hu et al., 2020b; You et al., 2020b; Xie et al., 2022; Xu et al., 2021) in that both transfer knowledge from a large pre-training dataset to another for downstream evaluation. However, our approach is a substantial departure from previous SSL approaches in two distinct aspects:

1. The trained GPSE model only serves as a feature extractor that can be coupled with *any* type of downstream prediction model, which will be trained from scratch.
2. GPSE extracts representations *solely* from the graph structure and does not make use of the domain-specific features such as atom and bond types (Hu et al., 2020b), allowing GPSE to be utilized on any graph dataset.

To compare the performance of SSL pre-training and GPSE feature augmentation, we use the MoleculeNet (Wu et al., 2018; Hu et al., 2020b) datasets. For the downstream model, we use the identical GINE architecture (Hu et al., 2020b) from Sun et al. (2022). Finally, the extracted representations from GPSE are concatenated with the atom embeddings and are then fed into the GINE model.

Table 3. Six PSE augmentations combined with five different GNN models evaluated on ZINC (12k subset) dataset. Performance is evaluated as MAE (↓) and averaged over 4 seeds.

	GCN	GatedGCN	GIN	GINE	Transformer	Avg. MAE reduction
none	0.288 ± 0.004	0.236 ± 0.008	0.285 ± 0.004	0.118 ± 0.005	0.686 ± 0.017	–
rand	1.277 ± 0.340	1.228 ± 0.012	1.239 ± 0.011	0.877 ± 0.011	1.451 ± 0.002	N/A
LapPE	0.209 ± 0.008	0.194 ± 0.006	0.214 ± 0.004	0.108 ± 0.008	0.501 ± 0.145	21.12%
RWSE	0.181 ± 0.003	0.167 ± 0.002	0.175 ± 0.003	0.070 ± 0.004	0.219 ± 0.007	42.75%
LapPE+RWSE	0.184 ± 0.008	0.163 ± 0.009	0.174 ± 0.003	0.082 ± 0.004	0.205 ± 0.007	41.32%
AllPSE	0.150 ± 0.007	0.143 ± 0.007	0.153 ± 0.006	0.073 ± 0.003	0.190 ± 0.008	50.85%
GPSE	**0.129 ± 0.003**	**0.113 ± 0.003**	**0.124 ± 0.002**	**0.065 ± 0.003**	**0.189 ± 0.016**	**56.24%**

Table 4. Performance on MoleculeNet datasets (scaffold split), evaluated in AUROC (%) ↑. Red indicates worse than baseline performance.

		BBBP	BACE	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV
Pre-training	Self-supervised pre-trained (Hu et al., 2020b)†	68.8 ± 0.8	79.9 ± 0.9	76.7 ± 0.4	64.2 ± 0.5	61.0 ± 0.7	71.8 ± 4.1	75.8 ± 1.7	77.3 ± 1.0
	GraphCL pre-trained (You et al., 2020b)	69.7 ± 0.7	75.4 ± 1.4	73.9 ± 0.7	62.4 ± 0.6	60.5 ± 0.9	76.0 ± 2.7	69.8 ± 2.7	**78.5 ± 1.2**
	InfoGraph pre-trained (Wang et al., 2022a)	66.3 ± 0.6	64.8 ± 0.8	68.1 ± 0.6	58.4 ± 0.6	57.1 ± 0.8	66.3 ± 0.6	44.3 ± 0.6	70.2 ± 0.6
	JOAOv2 pre-trained (Wang et al., 2022a)	66.4 ± 0.9	67.4 ± 0.7	68.2 ± 0.8	57.0 ± 0.5	59.1 ± 0.7	64.5 ± 0.9	47.4 ± 0.8	68.4 ± 0.5
	GraphMAE (Hou et al., 2022)	72.0 ± 0.6	83.1 ± 0.9	75.5 ± 0.6	64.1 ± 0.3	60.3 ± 1.1	**82.3 ± 1.2**	76.3 ± 2.4	77.2 ± 1.0
	GraphLoG (Xu et al., 2021)	**72.5 ± 0.8**	**83.5 ± 1.2**	75.7 ± 0.5	63.5 ± 0.7	61.2 ± 1.1	76.7 ± 3.3	76.0 ± 1.1	77.8 ± 0.8
No pre-training	No augmentation (baseline) (Hu et al., 2020b)	65.8 ± 4.5	70.1 ± 5.4	74.0 ± 0.8	63.4 ± 0.6	57.3 ± 1.6	58.0 ± 4.4	71.8 ± 2.5	75.3 ± 1.9
	GraphLoG augmented	65.6 ± 1.0	82.5 ± 1.2	73.2 ± 0.5	63.6 ± 0.4	60.9 ± 0.7	72.5 ± 3.5	72.4 ± 1.5	74.4 ± 1.5
	LapPE augmented	67.1 ± 1.6	80.4 ± 1.5	76.6 ± 0.3	65.9 ± 0.7	59.3 ± 1.7	76.4 ± 2.3	75.6 ± 0.8	75.6 ± 1.1
	RWSE augmented	67.0 ± 1.4	79.6 ± 2.8	76.3 ± 0.5	65.6 ± 0.3	58.5 ± 1.4	74.5 ± 4.4	75.0 ± 1.0	78.1 ± 1.5
	AllPSE augmented	67.6 ± 1.2	77.0 ± 4.4	75.9 ± 1.0	63.9 ± 0.3	**63.0 ± 0.6**	72.6 ± 4.3	67.9 ± 0.7	75.4 ± 1.5
	GPSE augmented	66.2 ± 0.9	80.8 ± 3.1	**77.4 ± 0.8**	**66.3 ± 0.8**	61.1 ± 1.6	78.8 ± 3.8	**76.6 ± 1.2**	77.2 ± 1.5

† Best test performance reported out of four pre-training strategies (see Table H.7 for expanded results).

We note that GPSE-augmented GINE achieves the best performance on three out of the eight MoleculeNet datasets against previously reported performances (Table 4). Moreover, GPSE augmentation improves performance over the baseline across *all* eight datasets, unlike some of the previously reported results that show negative transfer. Together, these results corroborate with the findings from Sun et al. (2022) that rich features can make up for the benefits of SSL pre-training. In our case, the GPSE encodings act as the rich features that contain positional and structural information from the graphs.

We also highlight that the Table 4 results are achieved in a setup where GPSE is at a comparative disadvantage: As a general-purpose feature extractor trained on a separate dataset, GPSE cannot leverage atom and bond features of the downstream graphs unlike typical molecular graph SSL methods. When GraphLoG (Xu et al., 2021) is similarly used as a feature extractor only, for a fair comparison, it is well outperformed by GPSE and even suffers from negative transfer, highlighting the power of GPSE as a feature extractor. With this in mind, GPSE can potentially be combined with other SSL methods to enhance them in future work.

3.2. Transferability across diverse graph benchmarks

GPSE can be used on arbitrary types of graphs as it is trained using the graph structures alone, in contrast to the SSL pre-training methods. Here, we show that GPSE is transferable to general graph datasets beyond molecular data, even under extreme out-of-distribution (OOD) cases.

Transferability to molecular graph sizes We use Peptides-struct and Peptides-func from the Long Range Graph Benchmark (Dwivedi et al., 2022c) to test whether GPSE can still work when the downstream (macro-)molecular graphs are substantially larger than those used for training. Despite this difference in graph sizes, GPSE outperforms explicitly computed PSEs when used with GPS as well as a GCN architecture optimized for long-range benchmarks by Tönshoff et al. (2023) (Table 5). More strikingly, GPS+GPSE challenges SOTA results for Peptides-struct, surpassing Graph MLP-Mixer (He et al., 2022). The improved performance emphasizes the ability of GPSE to better extract global information from query graphs by providing a more informative initial encoding for the global attention mechanism in GPS.

Transferability to graph connectivity patterns We further test if GPSE generalizes to graph connectivity patterns distinct from its training dataset. Particularly, we use the superpixel graph benchmarking datasets CIFAR10 and MNIST from Dwivedi et al. (2022a), which are k-nearest neighbor graphs with $k = 8$, and employ connectivity patterns that significantly differ from those found in molecular graph datasets. Impressively, GPS+GPSE again achieves comparable results against GPS + computed PSEs (Table 5).

Transferability to extreme OOD node-classification benchmarks Taking the evaluation of GPSE one step fur-

Table 5. OOD transferability of MolPCBA-trained GPSE to datasets that vary in graph size and connectivity patterns. Dataset information and statistics are available in Appendix D.

	Peptides-struct MAE ↓	**Peptides-func** AP ↑	**CIFAR10** ACC (%) ↑	**MNIST** ACC (%) ↑
Avg. # nodes	150.9	150.9	117.6	70.6
Avg. connectivity	0.022	0.022	0.069	0.117
GIN	–	–	55.26 ± 1.53	96.49 ± 0.25
GINE	0.3547 ± 0.0045	0.5498 ± 0.0079	–	–
GatedGCN (Bresson & Laurent, 2017)	0.3420 ± 0.0013	0.5864 ± 0.0077	67.31 ± 0.31	97.34 ± 0.14
Graph MLP-Mixer (He et al., 2022)	0.2475 ± 0.0020	0.6920 ± 0.0054	**72.46 ± 0.36**	**98.35 ± 0.10**
DRew+GCN (Gutteridge et al., 2023)	0.2781 ± 0.0028	**0.6996 ± 0.0076**	–	–
PR-MPNN (Qian et al., 2024)	0.2477 ± 0.0005	0.6825 ± 0.0086	–	–
GCN+LapPE (Tönshoff et al., 2023)	0.2492 ± 0.0019	0.6218 ± 0.0055	–	–
GCN+RWSE	0.2574 ± 0.0020	0.6067 ± 0.0069	–	–
GCN+**GPSE**	0.2487 ± 0.0011	0.6316 ± 0.0085	–	–
GPS+none	0.3817 ± 0.0207	0.6231 ± 0.0252	71.67 ± 0.01	98.05 ± 0.00
GPS+(RWSE/LapPE) (Rampášek et al., 2022)	0.2500 ± 0.0005	0.6535 ± 0.0041	72.30 ± 0.36	98.05 ± 0.13
GPS+AllPSE	0.2509 ± 0.0028	0.6397 ± 0.0092	72.05 ± 0.35	98.08 ± 0.12
GPS+**GPSE**	**0.2464 ± 0.0025**	0.6688 ± 0.0151	72.31 ± 0.25	98.08 ± 0.13

ther, we test its ability to provide useful information to transductive node classification tasks, where the graphs contain hundreds of thousands of nodes, which are completely out of distribution from the GPSE training dataset. We use both MPNN (GCN, GraphSAGE (Hamilton et al., 2017a), GATv2 (Veličković et al., 2018; Brody et al., 2022)) and graph Transformer baselines. Remarkably, on 8/10 model-dataset combinations, GPSE attains the best or equal-best result, as well as the best overall results, while LapPE performs below the no-encoding baseline for MPNNs (Table 6).

Meanwhile, the indifference in performance on the Proteins dataset for MPNNs is not unexpected, as the connectivity structures of the protein interaction network do not contribute to the proteins' functions as meaningfully. Instead, the identity of the proteins' interacting partners is of importance, commonly referred to as *homophily* in the graph representation learning community (Zhu et al., 2020) or more generally known as the *Guilt-by-Association* principle in network biology (Cowen et al., 2017). This result provides valuable insights into the usefulness of GPSE as an augmentation: It is more beneficial when the underlying graph structure is informative for the downstream tasks.

3.3. Expressiveness of GPSE encodings

Given that GPSE can recover different PSEs so well (Table 1), it is natural to wonder whether it boosts standard MPNN expressiveness. We first confirm that GPSE encodings surpass 1-WL distinguishability by observing a clear visual separation of GPSE encodings on 1-WL indistinguishable graph pairs (Figure E.1). More information regarding graph isomorphism, the WL test and their connections to GNN expressivity is discussed in Appendix E.

Table 6. OOD transferability to OGB node classification benchmarks. Best model in each model-dataset category is underlined, best overall model for each dataset is indicated in **bold**.

	arXiv ACC (%) ↑	**Proteins** AUROC (%) ↑
GCN+none	71.62 ± 0.23	80.44 ± 0.56
GCN+LapPE	70.89 ± 0.20	80.38 ± 0.16
GCN+**GPSE**	71.70 ± 0.26	80.25 ± 0.19
SAGE+none	**72.36 ± 0.43**	80.35 ± 0.07
SAGE+LapPE	71.63 ± 0.16	80.27 ± 0.45
SAGE+**GPSE**	**72.34 ± 0.19**	80.14 ± 0.22
GAT(E)v2+none	71.69 ± 0.21	83.47 ± 0.13
GAT(E)v2+LapPE	71.30 ± 0.27	83.25 ± 0.05
GAT(E)v2+**GPSE**	72.17 ± 0.42	**83.51 ± 0.11**
Transformer+none	57.00 ± 0.79	73.93 ± 1.44
Transformer+LapPE	57.21 ± 0.25	74.05 ± 0.11
Transformer+**GPSE**	59.17 ± 0.21	74.67 ± 0.74
GPS+none	70.60 ± 0.28	69.55 ± 5.67
GPS+LapPE	70.62 ± 0.41	70.80 ± 4.14
GPS+**GPSE**	70.89 ± 0.36	72.05 ± 3.75

Table 7. Synthetic graph benchmarks with ten times stratified five-fold CV evaluated on ACC (%) ↑.

	CSL		**EXP**	
	Train	Test	Train	Test
GIN	10.0 ± 0.0	10.0 ± 0.0	49.8 ± 1.8	48.7 ± 2.2
GIN+rand	11.6 ± 3.7	12.7 ± 6.4	51.0 ± 2.0	51.3 ± 2.9
GIN+**GPSE**	98.2 ± 1.5	42.9 ± 7.9	84.6 ± 6.8	68.3 ± 7.5
GIN+LapPE	100.0 ± 0.0	92.5 ± 4.2	99.9 ± 0.2	99.5 ± 0.8
GIN+RWSE	100.0 ± 0.0	100.0 ± 0.0	99.7 ± 0.2	99.7 ± 0.6

To rigorously and systematically illustrate the expressivity of GPSE encodings further, we employ two synthetic benchmarks (Dwivedi et al., 2022a; Abboud et al., 2021) that require beyond 1-WL power, and use an MPNN model with 1-WL expressivity in GIN. Indeed, we find that GPSE provides extra power to the base MPNN to correctly distinguish graph isomorphism classes (Table 7). This expressivity boost by GPSE is particularly impressive, considering that (1) GPSE is pre-trained on MolPCBA, whose graph structures are not designed to be 1-WL indistinguishable like these synthetic graphs, and (2) naively adding random features to the input does not provide the same improvement.

We further point out that, in fact, augmenting the base GIN model using common PSEs like LapPE and RWSE readily archives nearly perfect graph isomorphism classification, corroborating with previous theoretical results on distance-encodings (Li et al., 2020) and spectral invariants (Fürer, 2010). This finding partially explains why GPSE provides additional power and also why previous methods using LapPE achieve perfect classification on these tasks (He et al., 2022). Finally, we note the performance difference between LapPE/RWSE and GPSE is not unexpected, as random input features only act as a patch to the MPNN expressivity limitation, rather than fully resolving it. Thus, developing more powerful and practically scalable GPSE models that losslessly capture the latent semantics of various PSEs is a vital avenue to explore in the future.

3.4. Efficiency & scaling of GPSE

We demonstrate the efficiency and scalability advantages of GPSE over hand-crafted PSEs through two sets of scaling experiments. In the first, we compare the computation time for different PSEs against GPSE as we vary the number of graphs, represented as a percentage of the MolPCBA dataset. In the latter set of experiments, we instead generate four datasets of 1000 synthetic graphs, but scale up the individual graph sizes in each dataset. Our results are presented in Appendix G, where we show that GPSE is not only considerably faster to compute than explicitly computing PSEs, but also scales much better than explicit PSE computation as both the number of graphs and graph sizes increase.

The benefit of GPSE further compounds and leads to orders-of-magnitude faster computation times when we compute and combine all PSEs required for a fair comparison (AllPSE). One primary computational advantage of GPSE is that its complexity remains unchanged at inference time, regardless of the number of PSE types used to train the model. In our study, we restricted ourselves to six different PSEs, but future work could easily include more complex and specialized PSEs yet claim the same efficiency properties for PSE extraction. However, for AllPSE, the computational costs of stacking an increasing number of encodings would quickly become untenable.

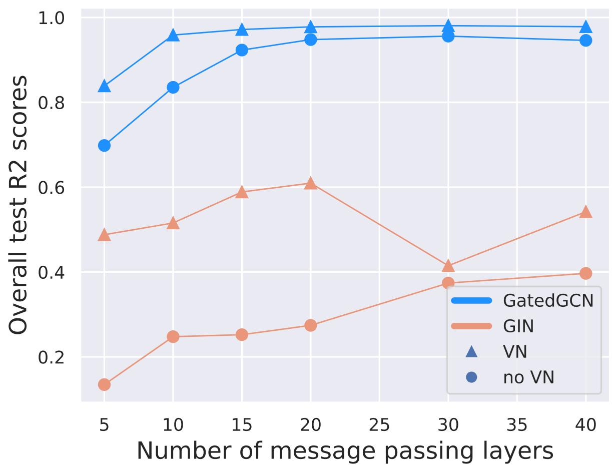


Figure 2. Virtual node (VN), convolution type, and layers ablation using 5% MolPCBA for training GPSE. The y-axis denotes the GPSE average test R^2 score over all six PSEs, as per Table 1.

3.5. Ablation studies

GPSE makes good use of depth and global message passing from the VN Despite the commonly known issue with MPNN over-smoothing as the number of message passing layers increases, we observe that GPSE does not oversmooth thanks to the gating mechanism and the residual connections in GatedGCN (Bresson & Laurent, 2017), benefiting from both the global message passing by VN and the model depth as indicated in Figure 2 and §2.2.

Downstream tasks benefit from the wide variety of PSEs used in GPSE pre-training Since GPSE is trained to capture latent semantics for recovering a wide range of PSEs, it mitigates the reliance on manually selecting task-specific PSEs, a major shortcoming of graph Transformers such as GPS and Graphormer (Ying et al., 2021). For instance, RWSE typically performs well for molecular tasks, while LapPE could be more useful for long-range dependency tasks (Rampášek et al., 2022; Dwivedi et al., 2022c). In Table H.4, we investigate whether a particular type of PSE contributes more or less to GPSE by testing the downstream performance of PCQM4Mv2 and MolHIV using different variants of GPSE that excludes one type of PSE during training. We observe that excluding any type of PSE generally reduces its performance in the downstream tasks, indicating the usefulness of different PSEs' semantics to the downstream tasks at various levels.

Asymptotic behavior with respect to the GPSE training sample sizes We perform a scaling law experiment with respect to the training sample sizes, from 5% to 80% of the MolPCBA. As shown in Figure H.1, the test loss (Appendix B) reduces as the size of the training set increases. This asymptotic behavior suggests that GPSE can further benefit from the increasing amount of training data.

The impact of pre-training dataset choice and fine-tuning on GPSE performance We reevaluate the performance of GPSE on PCQM4Mv2 and ZINC when trained on several other choices of molecular graph datasets, including GEOM (Axelrod & Gomez-Bombarelli, 2022), ZINC 250k (Gómez-Bombarelli et al., 2018), PCQM4Mv2 (Hu et al., 2021), and ChEMBL (Gaulton et al., 2012). On ZINC, GPSE performance is variable across different training datasets (Table H.5). Particularly, training GPSE on ChEMBL and MolPCBA, two largest datasets here, results in much better performances than using other, smaller datasets. The superior downstream performance achieved using larger pre-training datasets aligns well with our asymptotic results above, where a larger amount of training samples results in a more accurate GPSE for capturing PSEs and better downstream performance. However, we did not observe the same performance difference in the PCQM4Mv2-subset downstream task, indicating that the training size is not always the most crucial factor for good performance, an observation similar to Sun et al. (2022).

Finally, we investigate whether fine-tuning the GPSE model specifically on the downstream dataset could further improve its downstream performance (Table H.5). Similar to the above findings, we see that further fine-tuning GPSE may help in a task-specific manner, generally providing slight improvements on ZINC but less so on PCQM4Mv2. Together, the fact that using different pre-training datasets (provided they are sufficiently diverse) and further fine-tuning to specific downstream datasets having limited effects on GPSE performance reemphasize that GPSE learns *general* and *transferable* knowledge about various PSEs.

The success of GPSE as a general-purpose PSE encoding raises important discussion points regarding its effectiveness. Of particular interest is how and why GPSE outperforms individual and concatenated PSEs, even under significant distribution shifts. In Appendix I, we address several such points to both shed light on several aspects regarding GPSE's effectiveness, and encourage future work towards a better theoretical foundation of GPSE-like PSE extractors.

4. Conclusion

We have introduced GPSE, a unifying graph positional and structural encoder for augmenting any graph learning dataset while being applicable to all graph Transformers and message-passing GNNs. GPSE extracts rich node encodings by learning to predict a diverse collection of predefined PSEs in the initial self-supervised training phase on a set of unattributed graphs. We demonstrate the superior performance of GPSE encodings over explicitly constructed PSEs on a variety of graph learning benchmarks. Furthermore, GPSE shows great transferability across diverse benchmarking datasets and even challenges the SOTA on the Peptides-struct long-range benchmark, whose graph structures are vastly different from those in the MolPCBA dataset used to train the GPSE model.

Our study proposes a powerful, transferable and scalable alternative to hand-crafted PSEs to enhance GNNs. We therefore hope our work will motivate a shift from the limitations of PSE-based feature-engineering to developing more powerful encoders and foundation models for PSEs as feature extractors to advance the field of graph learning.

Limitations and future directions Despite the effectiveness of our GPSE model, it is currently prohibitively large to be trained on graph datasets with over one million graphs efficiently. As GPSE exhibits data scaling laws, where it asymptotically achieves perfect PSE recovery, it is a promising future direction to make GPSE more efficient and thus allow it to be trained on billion scale molecular graph datasets (Patel et al., 2020; Irwin et al., 2020).

Acknowledgements and Disclosure of Funding

We would like to thank Michael Galkin for discussions essential to the development of this paper. This work was partially funded by Bourse en intelligence artificielle des Études supérieures et postdoctorales (ESP) 2023-2024 [Semih Cantürk]; Canada CIFAR AI Chair, IVADO (Institut de valorisation des données) grant PRF-2019-3583139727, FRQNT (Fonds de recherche du Québec - Nature et technologies) grant 299376, NSERC Discovery grant 03267, and NSF DMS grant 2327211 [Guy Wolf]. This research was also enabled in part by compute resources provided by Mila (mila.quebec). The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *International Joint Conference on Artificial Intelligence*, 2021.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Axelrod, S. and Gomez-Bombarelli, R. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, 2022.

Barbero, F., Velingker, A., Saberi, A., Bronstein, M. M., and Giovanni, F. D. Locality-aware graph rewiring in GNNs. In *International Conference on Learning Representations*, 2024.

Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J., and Silva, J.-P. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Bodnar, C., Frasca, F., Otter, N., Wang, Y., Lio, P., Montufar, G. F., and Bronstein, M. Weisfeiler and Lehman go cellular: CW networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

Bresson, X. and Laurent, T. Residual Gated Graph ConvNets. *arXiv:1711.07553*, 2017.

Brody, S., Alon, U., and Yahav, E. How attentive are Graph Attention Networks? In *International Conference on Learning Representations*, 2022.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, 2022.

Cowen, L., Ideker, T., Raphael, B. J., and Sharan, R. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18(9):551–562, 2017.

Ding, C. H., He, X., Zha, H., Gu, M., and Simon, H. D. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings 2001 IEEE international conference on data mining*, pp. 107–114. IEEE, 2001.

Ding, Y., Orvieto, A., He, B., and Hofmann, T. Recurrent distance filtering for graph representation learning, 2024.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 23 (43):1–48, 2022a.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022b.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022c.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Forman, R. Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete and Computational Geometry*, 29(3):323–374, 2003.

Fürer, M. On the power of combinatorial and spectral invariants. *Linear algebra and its applications*, 432(9): 2373–2380, 2010.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Giraldo, J. H., Malliaros, F. D., and Bouwmans, T. Understanding the relationship between over-smoothing and over-squashing in graph neural networks. *arXiv:2212.02374*, 2022.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Gutteridge, B., Dong, X., Bronstein, M. M., and Di Giovanni, F. DRew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pp. 12252–12267. PMLR, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017a.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*. arXiv:1709.05584, 2017b.

He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. A generalization of ViT/MLP-Mixer to graphs. *arXiv:2212.13350*, 2022.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *34th Conference on Neural Information Processing Systems*, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. OGB-LSC: A large-scale challenge for machine learning on graphs. In *35th Conference on Neural Information Processing Systems: Datasets and Benchmarks Track*, 2021.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Irwin, J. J., Tang, K. G., Young, J., Dandarchuluun, C., Wong, B. R., Khurelbaatar, M., Moroz, Y. S., Mayfield, J., and Sayle, R. A. Zinc20—a free ultralarge-scale chemical database for ligand discovery. *Journal of chemical information and modeling*, 60(12):6065–6073, 2020.

Kanatsoulis, C. I. and Ribeiro, A. Graph neural networks are more powerful than we think. *arXiv preprint arXiv:2205.09801*, 2022.

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

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Li, G., Muller, M., Thabet, A., and Ghanem, B. DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H., and Jegelka, S. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

Liu, R., Cantürk, S., Wenkel, F., McGuire, S., Wang, X., Little, A., O'Bray, L., Perlmutter, M., Rieck, B., Hirn, M., et al. Taxonomy of benchmarks in graph representation learning. In *Learning on Graphs Conference*, pp. 6–1. PMLR, 2022.

Mialon, G., Chen, D., Selosse, M., and Mairal, J. GraphiT: Encoding graph structure in transformers. *CoRR*, abs/2106.05667, 2021.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and Leman go neural: higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2019. doi: 10.1609/aaai.v33i01.33014602.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

Nakata, M. and Shimazaki, T. Pubchemqc project: a large-scale first-principles electronic structure database for data-driven chemistry. *Journal of chemical information and modeling*, 57(6):1300–1308, 2017.

Nguyen, K., Nguyen, T., Ho, N., Nguyen, K., Nong, H., and Nguyen, V. Revisiting over-smoothing and over-squashing using Ollivier-Ricci curvature. *arXiv:2211.15779*, 2022.

Ollivier, Y. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009. ISSN 0022-1236.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

Patel, H., Ihlenfeldt, W.-D., Judson, P. N., Moroz, Y. S., Pevzner, Y., Peach, M. L., Delannée, V., Tarasova, N. I., and Nicklaus, M. C. Savi, in silico generation of billions of easily synthesizable compounds through expert-system type rules. *Scientific data*, 7(1):384, 2020.

Qian, C., Manolache, A., Ahmed, K., Zeng, Z., den Broeck, G. V., Niepert, M., and Morris, C. Probabilistically rewired message-passing neural networks. In *International Conference on Learning Representations*, 2024.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a General, Powerful, Scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021.

Scarselli, F., Gori, M., Ah Chung Tsoi, Hagenbuchner, M., and Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, January 2009. doi: 10.1109/TNN.2008.2005605.

Shiv, V. and Quirk, C. Novel positional encodings to enable tree-based transformers. *Advances in neural information processing systems*, 32, 2019.

Singh, S., Chaudhary, K., Dhanda, S. K., Bhalla, S., Usmani, S. S., Gautam, A., Tuknait, A., Agrawal, P., Mathur, D., and Raghava, G. P. SATPdb: a database of structurally annotated therapeutic peptides. *Nucleic acids research*, 44(D1):D1119–D1126, 2016.

Spielman, D. *Spectral graph theory*, volume 18. CRC Press Boca Raton, Florida, 2012.

Sreejith, R., Mohanraj, K., Jost, J., Saucan, E., and Samal, A. Forman curvature for complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(6): 063206, 2016.

Sun, R., Dai, H., and Yu, A. W. Does GNN pretraining help molecular representation? *Advances in Neural Information Processing Systems*, 35:12096–12109, 2022.

Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Graph learning with 1D convolutions on random walks. *arXiv:2102.08786*, 2021.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Tönshoff, J., Ritzert, M., Rosenbluth, E., and Grohe, M. Where did the gap go? reassessing the long-range graph benchmark, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

Wang, H., Kaddour, J., Liu, S., Tang, J., Kusner, M., Lasenby, J., and Liu, Q. Evaluating self-supervised learning for molecular graph embeddings. *arXiv:2206.08005*, 2022a.

Wang, H., Yin, H., Zhang, M., and Li, P. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022b.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xia, J., Zhu, Y., Du, Y., and Li, S. Z. Pre-training graph neural networks for molecular representations: retrospect and prospect. In *ICML 2022 2nd AI for Science Workshop*, 2022.

Xie, Y., Xu, Z., Zhang, J., Wang, Z., and Ji, S. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Xu, M., Wang, H., Ni, B., Guo, H., and Tang, J. Self-supervised graph-level representation learning with local and global structure. In *International Conference on Machine Learning*, pp. 11548–11558. PMLR, 2021.

Yi, H.-C., You, Z.-H., Huang, D.-S., and Kwoh, C. K. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1): bbab340, 2022.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, 2021.

You, J., Ying, Z., and Leskovec, J. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020a.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020b.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

A. Positional and structural encoding tasks

The success of GPSE relies on reliably learning diverse positional and structural encodings (PSE) for graphs during training. Here, we elaborate on the formulations of the PSEs introduced in §2.1.

We consider a simple undirected and unweighted graph $G = (V, E)$ as a tuple of the vertex set V and the edge set E, with no node or edge features. We denote the number of nodes and the number of edges as $n = |V|$ and $m = |E|$, respectively. Then, the corresponding adjacency matrix representing the graph G is a symmetric matrix $\mathbf{M} \in \{0,1\}^{n\times n}$, where $\mathbf{M}_{ij} = 1$ if $(v_i, v_j) \in E$ and 0 otherwise. The graph Laplacian $\mathbf{L}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{M} \tag{A.1}$$

where $\mathbf{D} \in \mathbb{N}^{n\times n}$ is a diagonal matrix whose entries correspond to the degree of a vertex in the graph, $\mathbf{D}_{ii} = deg(v_i) = |\mathcal{N}(v_i)| = |\{u|(v_i, u) \in E\}|$.

The graph Laplacian is a real symmetric matrix, thus having a full eigendecomposition as

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \tag{A.2}$$

where, $\mathbf{\Lambda}_{ii} = \lambda_i$ and $\mathbf{U}_{[:,i]} = u_i$ are the i^{th} eigenvalue and eigenvector (an eigenpair) of the graph Laplacian. We follow the convention of indexing the eigenpair from the smallest to the largest eigenvalue, i.e., $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. We further denote $\hat{\mathbf{U}}$ (and analogously the subdiagonal matrix $\hat{\mathbf{\Lambda}}$) as the matrix of Laplacian eigenvectors corresponding to non-trivial eigenvalues.

$$\hat{\mathbf{U}} = \mathbf{U}_{[:,\{i|\lambda_i \neq 0\}]} \tag{A.3}$$

Finally, we denote the (ℓ_2) normalization operation as $normalize(x) := \frac{x}{\|x\|_2}$

Laplacian eigenvector positional encodings (LapPE) LapPE is defined as the absolute value of the ℓ_2 normalized eigenvectors associated with non-trivial eigenvalues. We use the first four LapPE to train GPSE by default.

$$\text{LapPE}_i = |normalize(\hat{\mathbf{U}}_{[:,i]})| \tag{A.4}$$

The absolute value operation is needed to counter the sign ambiguity of the graph Laplacian eigenvectors, a known issue to many previous works that use the Laplacian eigenvectors to augment the models (Dwivedi et al., 2022a; Lim et al., 2022). However, common strategies to overcome the sign ambiguity issue such as random sign flipping (Dwivedi et al., 2022a) or constructing sign invariant function (Lim et al., 2022) do not resolve our issue here as we are trying to *recover* the PEs rather than using them as features.

We have, however, conducted an ablation study to find if better strategies may exist than taking the absolute value of the eigenvectors. new set of experiments to demonstrate this aspect further. In this study, we pretrained four different versions of GPSE using 5% MolPCBA-subset:

1. *GPSE-abs* takes the absolute value of the LapPE (default setting in our paper)
2. *GPSE-noabs* does not take the absolute value of the LapPE,
3. *GPSE-signinvar* uses a sign invariant loss function for LapPE by taking the minimum of the losses from both signs,
4. *GPSE-SignNet* uses a randomly initialized SignNet (Lim et al., 2022) model to generate sign invariant features as the training target for GPSE.

Our results in Table H.6 indicate that using the default absolute handling of LapPE results in similar or better performance than other strategies, indicating the effectiveness of using the absolute LapPE for training GPSE. Nevertheless, investigating better strategies for learning *invariant* representations for eigenvectors is an interesting venue for future studies.

We additionally use the eigenvalues as a graph-level regression task for training GPSE.

Electrostatic potential positional encodings (ElstaticPE) The pseudoinverse of the graph Laplacian $\mathbf{L}^\dagger$ has a physical interpretation that closely relates to the electrostatic potential between two nodes in the graph G when each node is treated as a charged particle (Kreuzer et al., 2021) and can be computed as

$$\mathbf{L}^\dagger = \mathbf{U}\mathbf{\Lambda}^\dagger\mathbf{U}^\top = \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}^{-1}\hat{\mathbf{U}}^\top \tag{A.5}$$

We further subtract each column of $\mathbf{L}^\dagger$ by its diagonal value to set zero ground state such that each node's potential on itself is 0.

$$\mathbf{Q} = \mathbf{L}^\dagger - diag(\mathbf{L}^\dagger)\mathbf{1}_n \tag{A.6}$$

The final ElstaticPE is a collection of aggregated values for each node, that summarizes the electrostatic interaction of a node with all other nodes:

1. Minimum potential from v_i to v_j: ElstaticPE$_1(i) = \text{min}(\mathbf{Q}_{[:,i]})$
2. Average potential from v_i to v_j: ElstaticPE$_2(i) = \text{mean}(\mathbf{Q}_{[:,i]})$
3. Standard deviation of potential from v_i to v_j: ElstaticPE$_3(i) = \text{std}(\mathbf{Q}_{[:,i]})$
4. Minimum potential from v_j to v_i: ElstaticPE$_4(i) = \text{min}(\mathbf{Q}_{[i,:]})$
5. Standard deviation of potential from v_j to v_i: ElstaticPE$_5(i) = \text{std}(\mathbf{Q}_{[i,:]})$
6. Average interaction on direct neighbors: ElstaticPE$_6(i) = \text{mean}\left((\mathbf{MQ})_{[:,i]}\right)$
7. Average interaction from direct neighbors: ElstaticPE$_7(i) = \text{mean}\left((\mathbf{MQ})_{[i,:]}\right)$

Random walk structural encodings (RWSE) Define the random walk matrix as the row-normalized adjacency matrix $\mathbf{P} := \mathbf{D}^{-1}\mathbf{M}$. Then $\mathbf{P}_{i,j}$ corresponds to the one-step transition probability from v_i to v_j.

The k^{th} RWSE (Dwivedi et al., 2022b) is defined as the probability of returning back to the starting state of a random walk after exactly k step of random walks:

$$\text{RWSE}_k = diag(\mathbf{P}^k) \tag{A.7}$$

Heat kernel diagonal structural encodings (HKdiagSE)

$$\text{HKdiagSE}_k = \sum_{i:\lambda_i \neq 0} e^{-k\lambda_i} normalize(\mathbf{U}_{[:,i]})^2 \tag{A.8}$$

Cycle counting structural encodings (CycleSE) CycleSE encodes global structural information of the graph by counting the number of k-cycles in the graph. For example, a 2-cycle corresponds to an undirected edge, and a 3-cycle corresponds to a triangle.

$$\text{CycleSE}_k = |\{\text{Cycles of length k}\}| \tag{A.9}$$

CycleSE is used as a graph-level regression task for training `GPSE`.

Normalizing PSEs tasks Finally, we perform graph-wide normalization preprocessing step on each node-level PSE task so that they have zero mean and unit standard deviation. This normalization step ensures all PSE targets are on the same scale, making the training process more stable.

B. Implementation details

B.1. GPSE computation

The GPSE model is built using a GatedGCN backbone (Bresson & Laurent, 2017) with PSE-specific MLP decoding heads. GPSE uses random noise drawn from a 20-dimensional standard Gaussian as the input node features. The random features are then projected to the match the hidden dimension, d, of the model, resulting in the hidden representations of the first layer:

$$h_i^{(0)} = \text{ReLU}\Big(x_i W_{\text{inp}}\Big) \tag{B.1}$$

where $h_i^{(0)} \in \mathbb{R}^{1\times d}$ indicates the hidden feature of node i in the first layer, $W_{\text{inp}} \in \mathbb{R}^{20\times d}$ is the linear projection layer, and $x_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^{1\times 20}$ is the random noise. Next, the model enters L layers of GatedGCN convolution layers, where each layer is defined as:

$$h_i^{(l+1)} = \text{ReLU}\left(h_i^{(l)} W_1^{(l)} + \sum_{j\in\mathcal{N}(i)} \sigma\Big(h_i^{(l)} W_2^{(l)} + h_j^{(l)} W_3^{(l)}\Big) \odot \Big(h_j^{(l)} W_4^{(l)}\Big)\right) \tag{B.2}$$

where $W_1^{(l)}, W_2^{(l)}, W_3^{(l)}, W_4^{(l)} \in \mathbb{R}^{d\times d}$ are learnable parameters for layer l, σ is the sigmoid function, and $\odot$ is the elementwise multiplication operator. Finally, the processed hidden feature $h_i^{(L)}$ is decoded via a two-layer MLP to predict the k^{th} node-level PSEs, such as LapPE and RWSE.

$$\hat{y}_{i,k} = \text{ReLU}\Big(h_i^{(L)} W_{k,1}\Big) W_{k,2} \tag{B.3}$$

where $W_{k,1} \in \mathbb{R}^{d\times d}$ and $W_{k,2} \in \mathbb{R}^{d\times 1}$ are learnable parameters for projecting the final hidden representation to the PSE prediction. For graph-level PSEs, such as CycleSE, we use sum-pooling to reduce the hidden representations to graph-level first, and similarly apply a two-layer MLP afterwards. Once trained, we apply GPSE to extract $h^{(L)}$ for the graphs in the downstream dataset and use it in-place of the traditional PSEs. We set L to 20, and d to 512 for our final GPSE architecture. We also present an ablation study on various architectural choices to demonstrate the effectiveness of our final model setting (Table H.1).

B.2. GPSE training loss function

We use a combination of ℓ_1 loss and cosine similarity loss for training GPSE using the PSE self-supervision defined in Appendix A. More specifically, given M number of graphs, and K number of target PSE tasks, we compute the loss as follows:

$$\mathcal{L} = \sum_{k=1}^{K}\sum_{i=1}^{M}\left[\left(\sum_{j=1}^{|V(G_i)|}\left|y_{j,k}^{(i)} - \hat{y}_{j,k}^{(i)}\right|\right) + \left(1 - \sum_{j=1}^{|V(G_i)|} \tilde{y}_{j,k}^{(i)} \tilde{\hat{y}}_{j,k}^{(i)}\right)\right] \tag{B.4}$$

where $y_{j,k}^{(i)}$, $\hat{y}_{j,k}^{(i)}$ are the true and predicted values of the j^{th} node of i^{th} graph for the k^{th} PSE task. $\tilde{y}$ and $\tilde{\hat{y}}$ are the ℓ_2 normalized version of y and $\hat{y}$, respectively. Note that in practice, we compute the loss over mini-batches of graphs rather than over all of the training graphs.

B.3. Compute environment and resources

Our codebase is based on GraphGPS (Rampášek et al., 2022), which uses PyG and its GraphGym module (Fey & Lenssen, 2019; You et al., 2020a). All experiments are run using Tesla V100 GPUs (32GB), with varying numbers of CPUs from 4 to 8 and up to 48GB of memory (except for two cases: (i) 80GB of memory is needed when performing downstream evaluation on MolPCBA, and (ii) 128GB is needed when pre-training GPSE on the ChEMBL dataset).

We recorded the run time for both the GPSE pre-computation and the downstream evaluation training loop using Python's `time.perf_counter()` function and reported them in Table B.1, B.2, B.3. We did not report the GPSE pre-computation time for other downstream benchmarks since they are all within five minutes.

B.4. Hyperparameters

B.4.1. Downstream tasks

In most of the downstream task hyperparameter searches, we followed the best settings from previous studies (Rampášek et al., 2022), and primarily tuned the GPSE encoding parameters, including the GPSE processing encoder type, the encoded dimensions, the input and output dropout rate of the processing encoder, and the application of batch normalization to the input GPSE encodings. For completeness, we list all hyperparameters for our main benchmarking studies in Tables B.1 and B.2.

Table B.1. GPS+GPSE hyperparameters for molecular property prediction benchmarks

Hyperparameter	**ZINC** (subset)	**PCQM4Mv2** (subset)	**MolHIV**	**MolPCBA**
# GPS Layers	10	5	10	5
Hidden dim	64	304	64	384
GPS-MPNN	GINE	GatedGCN	GatedGCN	GatedGCN
GPS-SelfAttn	–	Transformer	Transformer	Transformer
# Heads	4	4	4	4
Dropout	0.00	0.00	0.05	0.20
Attention dropout	0.50	0.50	0.50	0.50
Graph pooling	mean	mean	mean	mean
PE dim	32	128	20	48
PE encoder	2-Layer MLP	2-Layer MLP	Linear	Linear
Input dropout	0.50	0.50	0.30	0.30
Output dropout	0.00	0.20	0.10	0.10
Batchnorm	yes	no	yes	yes
Batch size	32	256	32	512
Learning rate	0.001	0.0002	0.0001	0.0005
# Epochs	2000	100	100	100
# Warmup epochs	50	5	5	5
Weight decay	1.00e-5	1.00e-6	1.00e-5	1.00e-5
# Parameters	292,513	6,297,345	573,025	9,765,264
PE precompute	2 min	1.5 hr	8 min	1.3 hr
Time (epoch/total)	10s/5.78h	102s/2.82h	121s/3.37h	185s/5.15h

MoleculeNet small benchmarks settings We used the default GINE architecture following previous studies (Hu et al., 2020b), which has five hidden layers and 300 hidden dimensions. For all five benchmarks, we use the same GPSE processing encoder settings as shown in Table B.4a.

CSL & EXP synthetic graph benchmarks settings We follow He et al. (2022) and use GIN (Xu et al., 2019) as the underlying MPNN model, with five hidden layers and 128 dimensions. We use the same GPSE processing encoder settings for both CSL and EXP as shown in Table B.4b.

Table B.2. GPS+GPSE hyperparameters for transferability benchmarks

Hyperparameter	**Peptides-struct**	**Peptides-func**	**CIFAR10**	**MNIST**
# GPS Layers	4	4	3	3
Hidden dim	96	96	52	52
GPS-MPNN	GatedGCN	GatedGCN	GatedGCN	GatedGCN
GPS-SelfAttn	Transformer	Transformer	Transformer	Transformer
# Heads	4	4	4	4
Dropout	0.00	0.00	0.00	0.00
Attention dropout	0.50	0.50	0.50	0.50
Graph pooling	mean	mean	mean	mean
PE dim	8	24	8	8
PE encoder	Linear	2-Layer MLP	2-Layer MLP	Linear
Input dropout	0.10	0.10	0.30	0.50
Output dropout	0.05	0.00	0.00	0.00
Batchnorm	yes	yes	no	no
Batch size	128	128	16	16
Learning rate	0.0005	0.0003	0.001	0.001
# Epochs	200	200	100	100
# Warmup epochs	10	10	5	5
Weight decay	1.00e-4	0	1.00e-5	1.00e-4
# Parameters	510,435	529,250	120,886	119,314
PE precompute	3 min	3 min	14 min	16 min
Time (epoch/total)	12s/0.65h	12s/0.67h	88s/2.44h	104s/2.90h

Table B.3. Downstream MPNN hyperparameters for node-level benchmarks.

Hyperparameter	**arXiv**	**Proteins**
MPNN	SAGE	GATEv2
# MPNN Layers	3	3
Hidden dim	256	256
Dropout	0.50	0.00
PE dim	32	32
PE encoder	2-Layer MLP	Linear
Input dropout	0.50	0.40
Output dropout	0.20	0.05
Batchnorm	no	no
Learning rate	0.01	0.01
# Epochs	500	1000
Weight decay	0	0
# Parameters	534,888	910,448
PE precompute	8 sec	3 min
Time (epoch/total)	0.25s/0.07h	35s/9.40h

Table B.4. GPSE processing encoder hyperparameters for MoleculeNet small benchmarks and synthetic WL graph benchmarks.

(a) MoleculeNet small benchmarks settings.

Hyperparameter	
PE dim	64
PE encoder	Linear
Input dropout	0.30
Output dropout	0.10
Batchnorm	yes
Learning rate	0.003
# Epochs	100
# Warmup epochs	5
Weight decay	0

(b) Synthetic WL graph benchmarks settings.

Hyperparameter	
PE dim	128
PE encoder	Linear
Input dropout	0.00
Output dropout	0.00
Batchnorm	yes
Learning rate	0.002
# Epochs	200
Weight decay	0

C. Theory details

Message-passing GNNs have receptive fields that grow exponentially with the number of layers. Given two nodes, the influence of one onto the other might become too weak over long graph distances, hindering the learning task. This phenomenon has been referred to as *over-squashing* (Alon & Yahav, 2021). A similar problem also occurs as the number of layers increases, where the nodes' hidden representations become increasingly similar as the number of layers increase, also known as *over-smoothing* (Li et al., 2019).

C.1. Relevance to GPSE

The over-smoothing and over-squashing problems are essential to overcome to effectively learn the positional and structural encodings, especially for those that require *global views of the graph*. For example, the Laplacian eigenvector corresponding to the first non-trivial eigenvalue, also known as the Fiedler vector, corresponds to the solution of the graph min-max cut problem (Ding et al., 2001). Intuitively, this problem requires accessing the global view of the entire graph as it, colloquially, aims to partition the entire graph into two parts with minimal connections.

A straightforward solution to incorporating more global information into the model is by *stacking more message-passing layers* to increase the receptive field and thus effectively expose the model to information beyond the local structure. However, simply stacking more message-passing layers easily leads to *over-smoothing*, where the messages of each node become increasingly uniform as the number of layers increases. Our usage of the gating mechanism, along with residual connections, effectively mitigates this issue while still exposing the model to more non-local information.

Meanwhile, the model may still have difficulty incorporating global information, even after fixing over-smoothing and stacking more layers, due to *over-squashing*. Informally, over-squashing can be understood as the difficulty in losslessly sending messages between two nodes across the network. This difficulty is primarily because there are only a few possible routes between the two nodes compared to all other available routes to each of the nodes. We mitigate this problem using a *virtual node* that serves as the global information exchange hub to enable global information exchange, bypassing the "few routes" limitation.

C.2. Formal analysis

Definition C.1 (Over-squashing)**.** The squashing of a GNN is measured by the influence of one node on the features of another which we interpret as the partial derivative

$$\frac{\partial h_i^{(r+1)}}{\partial x_j}$$

for $h_i^{(r)}(x_1, ..., x_n)$ the r-th hidden feature at node i, and x_j the input feature at node j. If this quantity converges to 0 as r increases, then the network is said to suffer from *over-squashing*.

Another common problem with MPNNs is known as *over-smoothing*. It has often been observed that MPNNs with many layers produce node features that are very close or even identical, which limits expressivity and prevents learning. This stems from message-passing being equivalent to a local smoothing operation; too many smoothing iterations result in all nodes converging to identical states.

Definition C.2 (Over-smoothing)**.** The smoothing of a network can be measured by the norm (for example the ℓ_1-norm) of the state difference between neighbors, i.e.

$$\sum_{(i,j)\in E} |h_i^{(r)} - h_j^{(r)}|$$

where the sum is taken over the edges of the graph. If this quantity converges to 0 as r increases, the network is said to suffer from *over-smoothing*.

In the following section, we will refer to the relationships between over-squashing and over-smoothing with graph curvature. There are many versions of graph curvature (Forman, 2003; Ollivier, 2009; Sreejith et al., 2016; Topping et al., 2022), all closely related. Here we will only consider the balanced Forman curvature from Topping et al. (2022).

Definition C.3 (Graph curvature)**.** For any edge (i, j) in a simple, unweighted graph G, its contribution to graph curvature is given by

$$\text{Ric}(i,j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + |\#_\Delta(i,j)| \left(\frac{2}{\max\{d_i, d_j\}} + \frac{1}{\min\{d_i, d_j\}} \right) + \frac{\gamma_{\max}}{\max\{d_i, d_j\}} \left(|\#^i_\Box| + |\#^j_\Box| \right)$$

where $\#^i_\Box$ is the number of 4-cycles containing the node i (diagonals not allowed), $\#^i_\Delta$ is the number of 3-cycles containing i, d_i is the degree of i and $\gamma_{\max}$ is the maximum over nodes k of the number of 4-cycles that pass through the nodes i, j and k.

It can then be shown that negative curvature causes over squashing (Topping et al., 2022; Nguyen et al., 2022) and positive curvature causes over smoothing (Nguyen et al., 2022; Giraldo et al., 2022).

Next, we show that rewiring the graph by adding a virtual node increases the balanced Forman curvature of the graph at most edges.

Proposition C.4. *The balanced Forman Curvature is increased for most edges when adding a virtual node such that*

$$\text{Ric}(i,j) - \text{Ric}^{+\text{VN}}(i,j) \leq \frac{1}{(d_i - \delta)^2 + d_i - \delta} - \frac{2\delta}{d_i^2 + d_i},$$

where d_i is the degree of the most connected node of the edge (i,j) and $\delta = d_i - d_j$.

Proof. $\#_\Box$ is invariant when adding virtual node because it automatically creates diagonals in the new 4-cycles. Therefore, $\gamma_{\max}$ is also invariant. As for d_i, d_j and $\#_\Delta$, they are all increased by 1:

$$\text{Ric}(i,j) - \text{Ric}^{+}(i,j) \approx 2\left(\frac{1}{d_i} + \frac{1}{d_j} - \frac{1}{d_i+1} - \frac{1}{d_j+1}\right) + |\#_\Delta(i,j)| \left(\frac{2}{\max\{d_i, d_j\}} + \frac{1}{\min\{d_i, d_j\}} \right)$$
$$- \left(|\#_\Delta(i,j)| + 1)(\frac{2}{\max\{d_i, d_j\}+1} + \frac{1}{\min\{d_i, d_j\}+1} \right)$$

We can let $d_i \geq d_j$ without loss of generality. The inequality is not influenced by the introduction of a virtual node:

$$\text{Ric}(i,j) - \text{Ric}^{+}(i,j) \approx 2\left(\frac{1}{d_i^2 + d_i} + \frac{1}{d_j^2 + d_j}\right) + |\#_\Delta(i,j)| \left(\frac{2}{d_i} + \frac{1}{d_j}\right) - (|\#_\Delta(i,j)| + 1)\left(\frac{2}{d_i+1} + \frac{1}{d_j+1}\right)$$

$$\text{Ric}(i,j) - \text{Ric}^{+}(i,j) \approx 2\left(\frac{1}{d_i^2 + d_i} + \frac{1}{d_j^2 + d_j}\right) + |\#_\Delta(i,j)| \left(\frac{2}{d_i^2 + d_i} + \frac{1}{d_j^2 + d_j}\right) - \left(\frac{2}{d_i+1} + \frac{1}{d_j+1}\right)$$

The number of triangles is upper bounded by the least connected node's degree minus 1, $|\#_\Delta| \leq d_j - 1$. We then have:

$$\begin{aligned}
\text{Ric}(i,j) - \text{Ric}^{+}(i,j) &\leq 2\left(\frac{1}{d_i^2 + d_i} + \frac{1}{d_j^2 + d_j}\right) + (d_j - 1)\left(\frac{2}{d_i^2 + d_i} + \frac{1}{d_j^2 + d_j}\right) - \left(\frac{2}{d_i+1} + \frac{1}{d_j+1}\right) \\
&= -\frac{2(d_i - 1)}{d_i^2 + d_i} - \frac{d_j - 2}{d_j^2 + d_j} + (d_j - 1)\left(\frac{2}{d_i^2 + d_i} + \frac{1}{d_j^2 + d_j}\right) \\
&= -\frac{2(d_i - 1 - d_j + 1)}{d_i^2 + d_i} - \frac{d_j - 2 - d_j + 1}{d_j^2 + d_j} \\
&= \frac{1}{d_j^2 + d_j} - \frac{2(d_i - d_j)}{d_i^2 + d_i}
\end{aligned}$$

Let's call the difference between the two nodes' degrees δ. We get:

$$\text{Ric}(i,j) - \text{Ric}^{+}(i,j) \leq \frac{1}{(d_i - \delta)^2 + d_i - \delta} - \frac{2\delta}{d_i^2 + d_i}$$

This upper bound gives us a good insight on the general behavior of the curvature when adding a virtual node.

First case: The upper bound of the difference is negative for $\delta \neq 0$ and $d_j \neq 1$. This means that for the most cases, the addition of the virtual node clearly increases the curvature.

Second case: The upper bound is positive for $d_j = 1$ (ie. $d_i - \delta = 1$). However, the isolated edges are not responsible for bottleneckness. It is to be noted that such *isolated* edges never have negative curvature, neither before nor after the addition of the virtual node. This is a direct consequence of the curvature definition.

Third case: The upper bound is positive for $\delta = 0$. This comes as a surprise and might need future work. It is to be noted that the upper bound tends toward 0 pretty quickly as (as $\frac{1}{d_i^2}$), thus, by the nature of the upper bound, the addition of the virtual node should still increase curvature for most of the cases where $\delta = 0$.

Note that we didn't include the 4-cycle term, because this term is inversely proportional to the number of triangles, and is therefore equal to 0 when the number of triangle is maximal. Otherwise, as the number of triangle decreases, the upper bound on the 4-cycle term increases, in a slower fashion. Thus, the upper bound still holds.

D. Datasets

Table D.1. Task information for datasets used in transferability experiments.

Dataset	Num. graphs	Num. nodes	Num. edges	Pred. level	Pred. task	Num. tasks	Metric
ZINC-subset	12,000	23.15	24.92	graph	reg.	1	MAE
CIFAR10	60,000	117.63	469.10	graph	class. (10-way)	1	ACC
MNIST	70,000	70.57	281.65	graph	class. (10-way)	1	ACC
MolHIV	41,127	25.51	27.46	graph	class. (binary)	1	AUROC
MolPCBA	437,929	25.97	28.11	graph	class. (binary)	128	AP
MolBBBP	2,039	24.06	25.95	graph	class. (binary)	1	AUROC
MolBACE	1,513	34.09	36.86	graph	class. (binary)	1	AUROC
MolTox21	7,831	18.57	19.29	graph	class. (binary)	21	AUROC
MolToxCast	8,576	18.78	19.26	graph	class. (binary)	617	AUROC
MolSIDER	2,039	33.64	35.36	graph	class. (binary)	27	AUROC
PCQM4Mv2-subset	446,405	14.15	14.58	graph	reg.	1	MAE
Peptides-func	15,535	150.94	153.65	graph	class. (binary)	10	AP
Peptides-struct	15,535	150.94	153.65	graph	reg.	11	MAE
CSL	150	41.00	82.00	graph	class. (10-way)	1	ACC
EXP	1,200	48.70	60.44	graph	class. (binary)	1	ACC
arXiv	1	169K	40M	node	class. (40-way)	1	ACC
Proteins	1	133K	1.2M	node	class. (binary)	112	AUROC

Table D.2. Classical graph properties of graph-level datasets used in transferability experiments.

	Num. nodes	Num. edges	Density	Connectivity	Diameter	Approx. max clique	Centrality	Cluster. coeff.	Num. triangles
ZINC-subset	23.15	24.92	0.101	1.00	12.47	2.06	0.101	0.006	0.06
CIFAR10	117.63	469.10	0.068	3.56	9.14	5.65	0.068	0.454	502.66
MNIST	70.57	281.65	0.116	3.71	6.83	5.56	0.116	0.478	316.65
MolHIV	25.51	27.46	0.103	0.927	11.06	2.02	0.103	0.002	0.03
MolPCBA	25.97	28.11	0.093	0.998	13.56	2.02	0.093	0.002	0.02
MolBBBP	24.06	25.95	0.114	0.950	10.75	2.03	0.114	0.003	0.03
MolBACE	34.09	36.86	0.070	1.00	15.22	2.10	0.070	0.007	0.10
MolTox21	18.57	19.29	0.157	0.976	9.37	2.02	0.159	0.003	0.03
MolToxCast	18.78	19.26	0.154	0.803	7.57	2.02	0.154	0.003	0.03
MolSIDER	33.64	35.36	0.103	0.856	12.45	2.02	0.120	0.004	0.04
PCQM4Mv2-subset	14.15	14.58	0.163	1.00	7.95	2.06	0.163	0.010	0.07
Peptides-func	150.94	153.65	0.022	0.990	56.42	2.00	0.022	0.000	0.001
Peptides-struct	150.94	153.65	0.022	0.990	56.42	2.00	0.022	0.000	0.001
CSL	41.00	82.00	0.100	3.98	6.00	2.10	0.100	0.050	4.10
EXP	48.70	60.44	0.054	0.00	1.00	2.00	0.054	0.000	0.00

D.1. Pre-training datasets

MolPCBA (Hu et al., 2020a) (MIT License) contains 400K small molecules derived from the MoleculeNet benchmark (Wu et al., 2018). There are 323,555 unique molecular graphs in this dataset.

ZINC (Gómez-Bombarelli et al., 2018) (Apache 2.0 License) contains 250K drug-like commercially available small molecules sampled from the full ZINC (Irwin et al., 2012) database. There are 219,2384 unique molecular graphs in this dataset.

GEOM (Axelrod & Gomez-Bombarelli, 2022) (CC0 1.0 license) consists of 300K drug-like small molecules. There are 169,925 unique molecular graphs in this dataset.

ChEMBL (Gaulton et al., 2012)[2] (CC BY-SA 3.0 License) consists of 1.4M drug-like bioactive small molecules. There are 970,963 unique molecular graphs in this dataset.

PCQM4Mv2 (Hu et al., 2021) (CC BY 4.0 License) contains 3.4M small molecules from the PubChemQC (Nakata & Shimazaki, 2017) project. The ground-state electronic structures of these molecules were calculated using Density Functional Theory. There are 273,920 unique molecular graphs in this dataset.

D.1.1. Extracting unique molecular graph structures

To extract unique molecular graphs, we use RDKit with the following steps:

1. For each molecule, convert all its heavy atoms to carbon and all its bonds to single-bond.
2. Convert the modified molecules into a list of SMILES strings.
3. Reduce the list to unique SMILES strings using the `set()` operation in Python.

D.2. Downstream evaluation datasets

ZINC-subset (Dwivedi et al., 2022a) (Custom license, free to use) is a 12K subset of the ZINC250K dataset (Gómez-Bombarelli et al., 2018). Each graph is a molecule whose nodes are atoms (28 possible types) and whose edges are chemical bonds (3 possible types). The goal is to regress the constrained solubility (Dwivedi et al., 2022a) (logP) of the molecules. This dataset comes with a pre-defined split with 10K training, 1K validation, and 1K testing samples.

MolHIV & MolPCBA (Hu et al., 2020a) (MIT License) are molecular property prediction datasets derived from the MoleculeNet benchmarks (Wu et al., 2018). Each graph represents a molecule whose nodes are atoms (9-dimensional features containing atom type, chirality, etc.) and whose edges are chemical bonds. The goal for MolHIV is to predict molecules' ability to inhibit HIV virus replication as a binary classification task. On the other hand, MolPCBA consists of 128 binary classification tasks that are derived from high-throughput bioassay measurements. Both datasets come with pre-defined splits based on the *scaffold splitting* procedure (Hu et al., 2020b).

PCQM4Mv2-subset (Hu et al., 2021; Rampášek et al., 2022) (CC BY 4.0 License) is a subsampled version of PCQM4Mv2 (Hu et al., 2021) using random 10% for training, 33% for validation, and the original validation set for testing. The molecular graphs are processed the same way as for MolHIV and MolPCBA, where each node is an atom, and each edge is a chemical bond. The task is to regress the HOMO-LUMO energy gap (in electronvolt) given the molecular graph. We note that the subsetted splits used in this work could be different from those used in Rampášek et al. (2022) as the `numpy` random generator may not be persistent across `numpy` versions[3]. To enable reproducibility, we also make our split indices available for future studies to benchmark against.

MoleculeNet small datasets (Hu et al., 2020b) (MIT License) We follow Sun et al. (2022) and use the selection of five small molecular property prediction datasets from the MoleculeNet benchmarks, including BBBP, BACE, Tox21, ToxCast, and SIDER. Each graph is a molecule, and it is processed the same way as for MolHIV and MolPCBA. All these datasets adopt the *scaffold splitting* strategy that is similarly used on MolHIV and MolPCBA.

Peptides-func & Peptides-struct (Dwivedi et al., 2022c) (CC BY-NC 4.0 License) both contain the same 16K peptide graphs retrieved from SAT-Pdb (Singh et al., 2016), whose nodes are residues. The two datasets differ in the graph-level tasks associated with them. Peptides-func aims to predict the functions of each peptide (10-way multilabel classification), while Peptides-struct aims to regress 11 structural properties of each peptide. Splitting is done via meta-class holdout based on the original labels of the peptides.

CIFAR10 & MNIST (Dwivedi et al., 2022a) (CC BY-SA 3.0 and MIT License) are derived from the CIFAR10 and MNIST image classification benchmarks by converting the images into SLIC superpixel graphs with 8 nearest neighbors for each node (superpixel). The 10-class classification and the splitting follow the original benchmarks (MNIST 55K/5K/10K, CIFAR10 45K/5K/10K train/validation/test splits).

CSL (Dwivedi et al., 2022a) (MIT License) contains 150 graphs that are known as circular skip-link graphs (Murphy et al., 2019). The goal is to classify each graph into one of ten isomorphism classes. The dataset is class-balanced, where each

[2]We used release 32 of ChEMBL: http://doi.org/10.6019/CHEMBL.database.32

[3]`https://stackoverflow.com/a/71790820/12519564`

isomorphism class contains 15 graph instances. Splitting is done by stratified five-fold cross-validation.

EXP (Abboud et al., 2021) (*unknown* license) contains 600 pairs of graphs (1,200 graphs in total) that cannot be distinguished by 1&2-WL tests. The goal is to classify each graph into one of two isomorphism classes. Splitting is done by stratified five-fold cross-validation.

arXiv (Hu et al., 2020a) (ODC-BY License) is a directed citation graph whose nodes are arXiv papers and whose edges are citations. Each node is featured by a 128-dimensional embedding obtained by averaging over the word embeddings of the paper's title and abstract. The goal is to classify the papers (nodes) into one of 40 subject areas of arXiv CS papers. Papers published before 2017 are used for training, while the remaining papers that are published before and after 2019 are used for validation and testing.

Proteins (Hu et al., 2020a) (CC0 License) is an undirected and weighted graph representing the interactions (edges) between proteins (nodes) obtained from eight species. Each edge has eight channels, corresponding to different types of protein interaction evidence. The task is to predict proteins' functions (112-way multilabel classification). The splitting is done by holding out proteins that correspond to specific species.

E. Visualization of GPSE encodings on 1-WL indistinguishable graph pairs

Definition E.1 (Graph isomorphism)**.** Two graphs G and H are isomorphic if there exists a bijection f between their vertex sets

$$f : V(G) \to V(H)$$

s.t. any two vertices $u, v \in G$ are adjacent in G if and only if $f(u), f(v) \in H$ are adjacent in H.

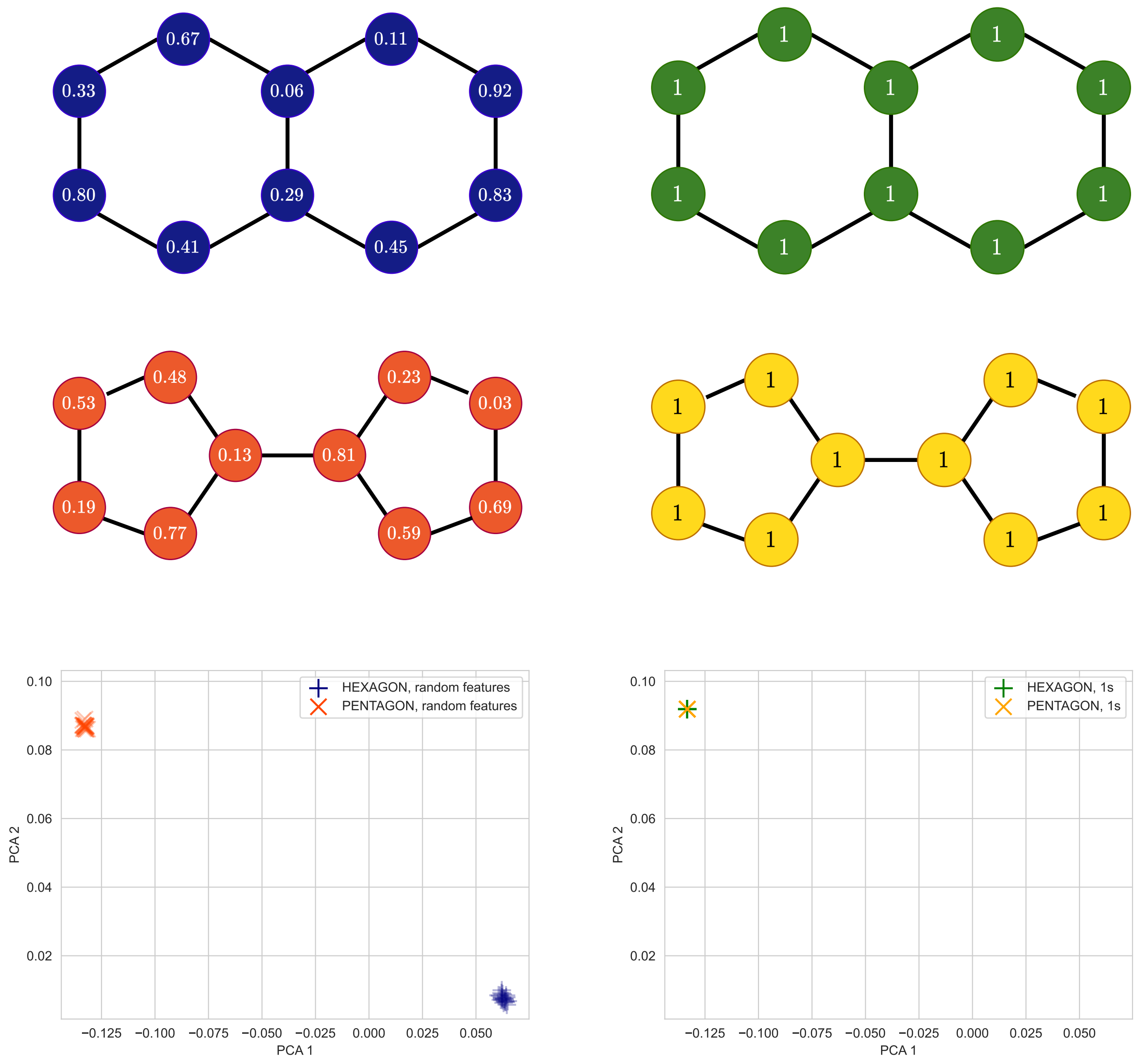


Figure E.1. 2D PCA visualization of `GPSE` encodings on 1-WL indistinguishable graph pairs. Applying `GPSE` with randomly initialized node features results in distinct encodings for HEXAGON (indigo) and PENTAGON (orange) graphs (left). The same graphs cannot be distinguished by our encoder when the node features are set to 1 for each node (right).

The 1-Weisfeiler-Leman (WL) test is an algorithm akin to message-passing that is commonly used to detect non-isomorphic graphs. It can also be viewed as a measure of expressivity: A GNN that can distinguish all pairs of non-isomorphic graphs that can also be distinguished by the 1-WL test is called "1-WL expressive".

In §3.3, we discussed that our `GPSE` is expressive enough to discern graphs that are 1-WL indistinguishable, a well-known limitation of MPNNs (Xu et al., 2019). However, Sato et al. (2021) show that the 1-WL expressivity limitation exists only when each node employs identical features; appending random features to the nodes is sufficient to achieve expressivity that goes beyond 1-WL. GPSE leverages precisely this property by replacing the node features by vectors drawn from a standard normal distribution, such that no two graphs have identical node features.

Here, we demonstrate the importance of these random node features empirically. Consider the following two graphs displayed in Figure E.1: One resembles two hexagons sharing an edge (referred to as HEXAGON), while the other resembles two pentagons connected by an edge (PENTAGON). These are a well-known pair of non-isomorphic but 1-WL indistinguishable graphs. Non-isomorphism implies that these graphs do not share the same connectivity (see Def. E.1 for a formal definition). The WL test also has its limitations: While two graphs that are deemed non-isomorphic are guaranteed to be so, there are cases where it cannot detect non-isomorphism as in the HEXAGON-PENTAGON case.

In our experiment, we create two sets of graphs, both consisting of 20 copies of HEXAGON and PENTAGON graphs each. The two sets are identical except one has all node features set to 1, while the other has features drawn from a random Normal assigned to each node, thus mirroring the actual GPSE training pipeline. We then apply an already trained GPSE encoder (trained on ZINC) to both sets and for each graph we generate aggregated (graph-level) encodings by averaging the obtained 512-dimensional node encodings from GPSE. For visualization purposes, we then apply dimensionality reduction to these graph-level encodings by first fitting a 2-dimensional PCA to GPSE encodings generated on ZINC, and then applying it to the encodings from the synthetic data.

As shown in Figure E.1, applying GPSE to graphs with randomly initialized node features results in distinct encodings for HEXAGON (indigo) and PENTAGON (orange) graphs (Fig. E.1, left). The same graphs cannot be distinguished by our encoder when the node features are 1 for each node (Fig. E.1, right). The same result is observed when analysing the graph-level PCA embeddings, that can clearly separate the two types of graphs when random node features are used by GPSE, but not otherwise. This underlines the importance of randomized node features in GPSE.

F. GPSE training and inference times

Table F.1. GPSE training times. Target PSE pre-computation included.

Training dataset	Num. unique graphs	Target PSE pre-comp time	Time (epoch/total)	Full training time
MolPCBA (default)	323,555	1.58h	596s / 19.88h	21.46h
PCQM4Mv2-full	273,920	0.87h	429s / 14.30h	15.17h
ZINC-full	219,384	0.89h	398s / 13.26h	14.15h
GEOM	169,925	0.78h	321s / 10.69h	11.47h
ChEMBL	970,963	5.97h	2509s / 83.65h	89.62h

Table F.2. GPSE inference times. LapPE and RWSE computation times are included for comparison. Missing entries are due to experimental settings not included in the benchmarking experiments.

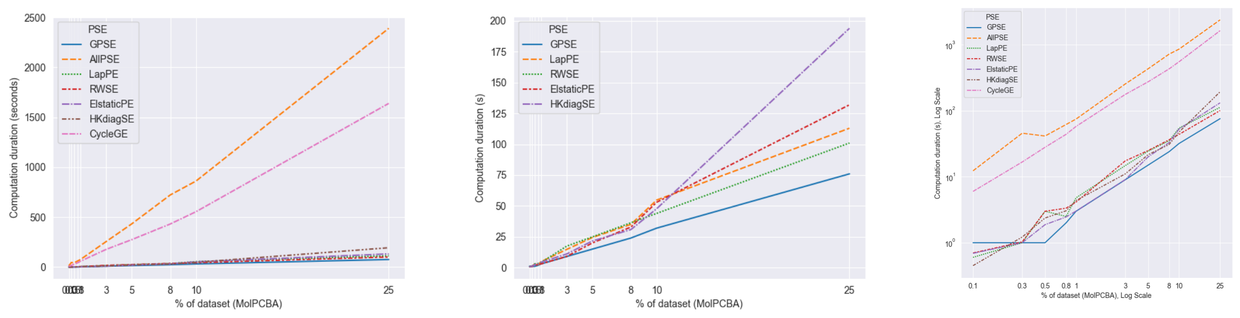
Dataset	Num. graphs	Time (GPSE)	Time (LapPE)	Time (RWSE)
ZINC-subset	12,000	6 sec	25 sec	11 sec
PCQM4Mv2-subset	446,405	3.57 min	3.88 min	7.32 min
PCQM4Mv2-full	3,746,620	31.15 min	–	51 min *
MolHIV	41,127	23 sec	37 sec	58 sec *
MolPCBA	437,929	4.6 min	6.13 min	8.33 min *
Peptides	15,535	28 sec	73 sec *	–
CIFAR10	60,000	2.15 min	2.55 min *	–
MNIST	70,000	100 sec	96 sec *	–
arXiv	1	4 sec	–	–
Proteins	1	6.68 min **	–	–

* Obtained from the GPS paper
** Neighbor batched computation (batch size: 1024, neighbor sizes: 30, 20, 10, 5, . . . , 5)

G. GPSE vs. conventional PSE scaling experiments

Table G.1. Runtimes of each PSE computation with respect to percentage of dataset used.

PSE / % MolPCBA	0.1%	0.3%	0.5%	0.8%	1%	3%	5%	8%	10%	25%
GPSE	1s	1s	1s	2s	3s	9s	15s	24s	32s	1m 16s
AllPSE	12s	46s	41s	1m 2s	1m 15s	4m 15s	7m 13s	12m 3s	14m 22s	39m 52s
LapPE	1s	1s	3s	3s	5s	15s	24s	35s	55s	1m 53s
RWSE	1s	1s	3s	3s	4s	17s	25s	36s	44s	1m 41s
ElstaticPE	1s	1s	2s	2s	3s	10s	20s	33s	53s	2m 12s
HKdiagSE	1s	1s	2s	3s	4s	11s	22s	31s	48s	3m 14s
CycleGE	6s	17s	28s	44s	58s	2m 57s	4m 35s	7m 12s	9m 15s	27m 20s



(a) GPSE + individual PSEs + combined PSEs (AllPSE)

(b) GPSE + individual PSEs only

(c) Log-log plot of GPSE + individual PSEs + combined PSEs (AllPSE)

Figure G.1. Scaling of PSE computation time with respect to number of graphs as % of MolPCBA dataset used. Visualization of Table G.1.

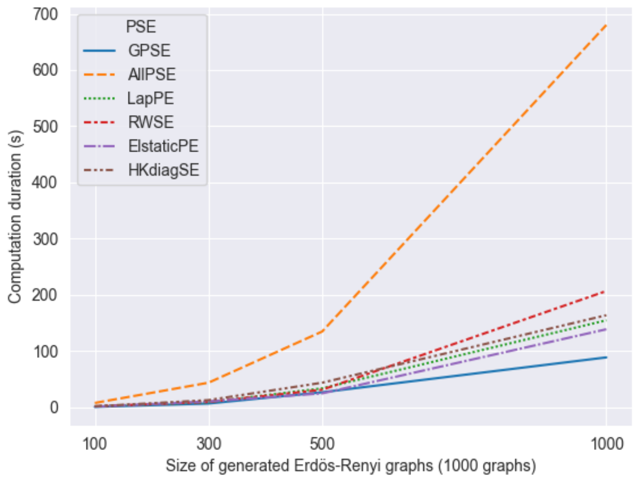
Table G.2. Runtimes of each PSE computation with respect to average graph sizes in dataset. CycleGE is excluded both in itself and as part of AllPSE, as cycle counting on large, dense and regular Erdős-Rényi graphs become computationally infeasible.

PSE / Graph size	100	300	500	1000
GPSE	1s	7s	27s	1m 29s
AllPSE (No CycleGE)	8s	44s	2m 15s	11m 20s
LapPE	2s	9s 250ms	34s	2m 35s
RWSE	2s	9s 760ms	31s 480ms	3m 27s
ElstaticPE	1s 500ms	10s 670ms	25s 30ms	2m 19s
HKdiagSE	2s	13s	44s	2m 44s

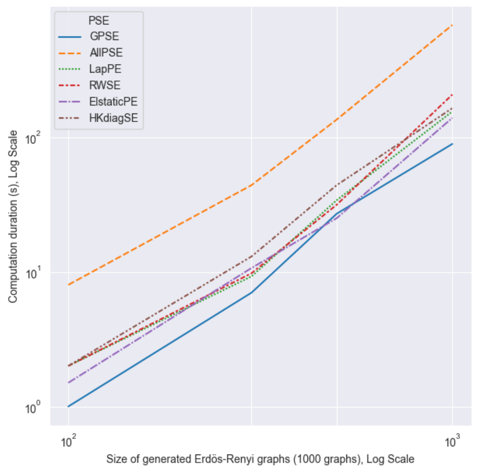
In these experiments, we measure and compare the computation time of GPSE with those of individual PSEs used in the pre-training of the GPSE model, as well as their combination (AllPSE). We conducted two sets of experiments. In the first, we used a dataset of similarly sized graphs in MolPCBA, but ran PSE computation for an increasing percentage of the dataset (Figure G.1). In the second, we generated multiple datasets of 1000 synthetic (Erdős-Rényi) graphs, scaling up the number of nodes per graph (Figure G.2) in each.

In both sets of experiments, GPSE is considerably faster to compute than the individual PSEs, and orders-of-magnitude faster than computing and combining all PSEs as AllPSE. Additionally, we observed that GPSE scales better than individual and combined PSEs. The better scaling properties of GPSE are particularly evident in scaling graph sizes (Figure G.2): As we scaled up the number of nodes in a graph to 1000, the advantage of GPSE became more apparent. This is somewhat an expected result: Regardless of graph size, inference of GPSE involves $O(Lm)$ computations, where L is the number of layers, and m is the number of edges; thus GPSE scales linearly on the number of edges in a graph. On the other hand, LapPE, for example, is expected to have polynomial complexity, as it involves eigendecomposing the graph Laplacian.

Another important point to highlight is that at inference time, *the complexity of GPSE remains unchanged* regardless of how many types of PSEs were used to train the model. This leads to significant advantages over AllPSE, which relies on



(a) GPSE + individual PSEs + combined PSEs (AllPSE)



(b) Log-log plot of GPSE + individual PSEs + combined PSEs (AllPSE)

Figure G.2. Scaling experiments with respect to size of graphs, keeping the number of graphs in each dataset constant. Visualization of Table G.2.

computing and concatenating all PSEs. The scalability issues of AllPSE is additionally exacerbated when useful but highly expensive PSEs such as CycleGE are used.

H. Additional results

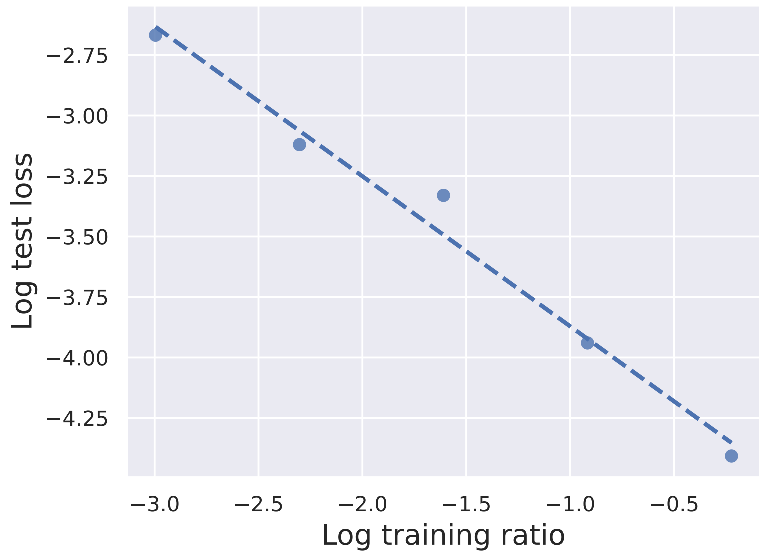


Figure H.1. Training size scaling law for `GPSE` on MolPCBA.

Table H.1. `GPSE` architecture ablation. Held-out positional and structural encodings prediction performance when trained on 5% MolPCBA (R^2 scores ↑). Ablated settings of the `GPSE` architecture are listed and compared to the full `GPSE` settings. The performance of the full `GPSE` architecture is shown in the bottom row.

Ablated setting	`GPSE` default	Overall	ElstaticPE	LapPE	RWSE	HKdiagSE	EigValSE	CycleSE
10 layers	20 layers	0.9585	0.9376	0.9302	0.9645	0.9622	0.9543	0.9701
128 dim	512 dim	0.9688	0.9484	0.9501	0.9729	0.9734	0.9706	0.9739
GCN	GatedGCN	0.0409	0.0408	0.0325	0.0424	0.0396	0.0483	0.0410
GIN	GatedGCN	0.6095	0.6953	0.4180	0.6237	0.6349	0.4002	0.6391
GATv2	GatedGCN	0.9580	0.9560	0.9476	0.9643	0.9530	0.9679	0.9561
No VN	VN	0.9478	0.9340	0.9359	0.9552	0.9479	0.9314	0.9568
Shared MLP head	Indep. MLP heads	0.9751	0.9619	0.9644	0.9802	0.9764	0.9714	**0.9778**
`GPSE`		**0.9790**	**0.9638**	**0.9725**	**0.9837**	**0.9808**	**0.9818**	0.9774

Table H.2. Five PSE augmentations combined with five different GNN models evaluated on the PCQM4Mv2-subset dataset. Performance is evaluated as mean absolute error (MAE ↓) and averaged over 4 seeds. Red indicates worse than baseline (none) performance.

	GCN	GatedGCN	GIN	GINE	Transformer	Avg. reduction
none	0.1934 ± 0.0012	0.1845 ± 0.0031	0.1790 ± 0.0011	0.1364 ± 0.0011	0.4193 ± 0.0167	–
rand	0.7604 ± 0.0019	0.7515 ± 0.0027	0.7532 ± 0.0021	0.4269 ± 0.0068	0.9810 ± 0.0064	N/A
LapPE	0.1834 ± 0.0023	0.1757 ± 0.0010	0.1720 ± 0.0018	0.1338 ± 0.0006	0.2433 ± 0.0056	11.55%
RWSE	0.1877 ± 0.0025	0.1782 ± 0.0012	0.1695 ± 0.0007	0.1317 ± 0.0005	0.1930 ± 0.0016	13.82%
LapPE+RWSE	0.1895 ± 0.0022	0.1632 ± 0.0016	0.1705 ± 0.0016	0.1370 ± 0.0005	0.1884 ± 0.0011	14.76%
AllPSE	0.1903 ± 0.0010	0.1595 ± 0.0014	0.1656 ± 0.0004	0.1416 ± 0.0040	0.2133 ± 0.062	13.59%
`GPSE`	**0.1822 ± 0.0028**	**0.1495 ± 0.0015**	**0.1615 ± 0.0015**	**0.1294 ± 0.0006**	**0.1805 ± 0.0021**	**19.32%**

Table H.3. GPSE pre-training ablations.

(a) Virtual node, convolution type, and layers ablation using 5% MolPCBA for training.

	GPSE (GatedGCN)		GPSE (GIN)	
Layers	VN	no VN	VN	no VN
5	0.8387	0.6982	0.4879	0.1347
10	0.9585	0.8353	0.5156	0.2476
15	0.9716	0.9231	0.5887	0.2523
20	0.9778	0.9478	0.6095	0.2743
30	0.9806	0.9559	0.4149	0.3740
40	0.9782	0.9459	0.5420	0.3968

(b) Training size scaling law for GPSE on MolPCBA.

Training size	Overall test loss (MAE + cosine loss) ↓
5%	0.06939
10%	0.04414
20%	0.03579
40%	0.01945
80%	0.01219

Table H.4. GPSE training task ablation. The colors indicate whether a particular PSE task for training GPSE improves or worsens the downstream performance.

Excluded task	**PCQM4Mv2** (subset) **MAE ↓**	**ogbg-molhiv AUROC ↑**
–	0.1196 ± 0.0004	0.7815 ± 0.0133
LapPE & EigVals	0.1200 ± 0.0006	0.7849 ± 0.0067
ElstaticPE	0.1197 ± 0.0007	0.7681 ± 0.0146
RWSE	0.1205 ± 0.0006	0.7771 ± 0.0105
HKdiagSE	0.1202 ± 0.0004	0.7787 ± 0.0198
CycleSE	0.1199 ± 0.0011	0.7739 ± 0.0240

Table H.5. GPSE training dataset ablation. Performance measured in MAE ↓. Green indicates fine-tuning GPSE on specific downstream dataset helps improve the performance. **Bold** indicates the best performance achieved on a particular downstream task.

			ZINC (subset)		**PCQM4Mv2** (subset)	
Training dataset	# unique graphs	Avg. # nodes	Not finetuned	Finetuned	Not finetuned	Finetuned
GEOM	169,925	18	0.0707 ± 0.0086	0.0685 ± 0.0055	0.1196 ± 0.0005	0.1194 ± 0.0002
ZINC	219,384	23	0.0700 ± 0.0041	–	0.1202 ± 0.0005	0.1197 ± 0.0007
PCQM4Mv2	273,920	14	0.0721 ± 0.0042	0.0713 ± 0.0014	**0.1192 ± 0.0005**	–
ChEMBL	970,963	30	0.0667 ± 0.0079	**0.0643 ± 0.0036**	0.1195 ± 0.0003	0.1195 ± 0.0005
MolPCBA	323,555	25	0.0648 ± 0.0030	0.0668 ± 0.0076	0.1196 ± 0.0004	0.1195 ± 0.0007

Table H.6. Ablation study on strategies to learn LapPE for GPSE.

	ZINC (subset) MAE ↓	**PCQM4Mv2 (subset) MAE ↓**	**Peptides-struct MAE ↓**	**Peptides-func AP ↑**
GPSE-abs (default)	**0.0957 ± 0.0044**	**0.1216 ± 0.0002**	**0.2516 ± 0.0018**	0.6584 ± 0.0042
GPSE-noabs	0.1051 ± 0.0046	0.1229 ± 0.0006	0.2554 ± 0.0025	**0.6687 ± 0.0119**
GPSE-signinvar	0.1116 ± 0.0072	0.1243 ± 0.0004	0.2594 ± 0.0019	0.6619 ± 0.0097
GPSE-SignNet	0.1035 ± 0.0052	0.1232 ± 0.0006	0.2568 ± 0.0020	0.6647 ± 0.0093

Table H.7. Extended results for Table 4: Performance on MoleculeNet small datasets (scaffold test split), evaluated in AUROC (%) ↑. Red indicates worse than baseline performance.

	BBBP	**BACE**	**Tox21**	**ToxCast**	**SIDER**	**ClinTox**	**MUV**	**HIV**
No pre-training (baseline) (Hu et al., 2020b)	65.8 ± 4.5	70.1 ± 5.4	74.0 ± 0.8	63.4 ± 0.6	57.3 ± 1.6	58.0 ± 4.4	71.8 ± 2.5	75.3 ± 1.9
SSL InfoMax pre-trained (Hu et al., 2020b)	68.8 ± 0.8	75.9 ± 1.6	75.3 ± 0.5	62.7 ± 0.4	58.4 ± 0.8	69.9 ± 0.3	75.3 ± 2.5	76.0 ± 0.7
SSL EdgePred pre-trained (Hu et al., 2020b)	67.3 ± 2.4	79.9 ± 0.9	76.0 ± 0.6	64.1 ± 0.6	60.4 ± 0.7	64.1 ± 3.7	74.1 ± 2.1	76.3 ± 1.0
SSL AttrMasking pre-trained (Hu et al., 2020b)	64.3 ± 2.8	79.3 ± 1.6	76.7 ± 0.4	64.2 ± 0.5	61.0 ± 0.7	71.8 ± 4.1	74.7 ± 1.4	77.2 ± 1.1
SSL ContextPred pre-trained (Hu et al., 2020b)	68.0 ± 2.0	79.6 ± 1.2	75.7 ± 0.7	63.9 ± 0.6	60.9 ± 0.6	65.9 ± 3.8	75.8 ± 1.7	77.3 ± 1.0
GraphCL pre-trained (You et al., 2020b)	69.7 ± 0.7	75.4 ± 1.4	73.9 ± 0.7	62.4 ± 0.6	60.5 ± 0.9	76.0 ± 2.7	69.8 ± 2.7	**78.5 ± 1.2**
InfoGraph pre-trained (Wang et al., 2022a)	66.3 ± 0.6	64.8 ± 0.8	68.1 ± 0.6	58.4 ± 0.6	57.1 ± 0.8	66.3 ± 0.6	44.3 ± 0.6	70.2 ± 0.6
JOAOv2 pre-trained (Wang et al., 2022a)	66.4 ± 0.9	67.4 ± 0.7	68.2 ± 0.8	57.0 ± 0.5	59.1 ± 0.7	64.5 ± 0.9	47.4 ± 0.8	68.4 ± 0.5
GraphMAE (Hou et al., 2022)	72.0 ± 0.6	83.1 ± 0.9	75.5 ± 0.6	64.1 ± 0.3	60.3 ± 1.1	**82.3 ± 1.2**	76.3 ± 2.4	77.2 ± 1.0
GraphLoG (Xu et al., 2021)	**72.5 ± 0.8**	**83.5 ± 1.2**	75.7 ± 0.5	63.5 ± 0.7	61.2 ± 1.1	76.7 ± 3.3	76.0 ± 1.1	77.8 ± 0.8
GraphLoG augmented	65.6 ± 1.0	82.5 ± 1.2	73.2 ± 0.5	63.6 ± 0.4	60.9 ± 0.7	72.5 ± 3.5	72.4 ± 1.5	74.4 ± 1.5
LapPE augmented	67.1 ± 1.6	80.4 ± 1.5	76.6 ± 0.3	65.9 ± 0.7	59.3 ± 1.7	76.4 ± 2.3	75.6 ± 0.8	75.6 ± 1.1
RWSE augmented	67.0 ± 1.4	79.6 ± 2.8	76.3 ± 0.5	65.6 ± 0.3	58.5 ± 1.4	74.5 ± 4.4	75.0 + 1.0	78.1 ± 1.5
AllPSE augmented	67.6 ± 1.2	77.0 ± 4.4	75.9 ± 1.0	63.9 ± 0.3	**63.0 ± 0.6**	72.6 ± 4.3	67.9 ± 0.7	75.4 ± 1.5
GPSE augmented	66.2 ± 0.9	80.8 ± 3.1	**77.4 ± 0.8**	**66.3 ± 0.8**	61.1 ± 1.6	78.8 ± 3.8	**76.6 ± 1.2**	77.2 ± 1.5

I. Additional discussions

Why does GPSE improve over precomputed PSEs? Our results demonstrate that GPSE encodings can improve upon augmenting GNNs with precomputed PSEs in downstream tasks. The fact that we can *recover* the target PSEs in pretraining (Table 1) accounts for why we can match the original PSEs. Why we *improve* upon them, meanwhile, can be attributed to our joint encoding: Learning to encode a diverse collection of PSEs leads to a general embedding space that abstracts both local and global perspectives of the query graph, which are more readily useable by the downstream model compared to the unprocessed PSEs. This also explains why a joint encoding outperforms the concatenation of all encodings, AllPSE. Concatenating many encodings very likely to leads to redundant representations, and also introduces significant noise to node features particularly in datasets/downstream tasks where only a small portion of the encodings are useful: This is reflected in most experiments where not only GPSE, but also RWSE- or LapPE-only configurations outperform AllPSE.

What advantages does GPSE bring over traditional SSL pre-training? In addition to being less prone to negative transfer and having competitive performance (Table 4), GPSE provides a few more advantages over traditional SSL pre-training methods: (1) GPSE uses randomly generated features instead of dataset-specific graph features, thus can be applied to arbitrary graph datasets; (2) GPSE is only used as a feature extractor and hence does not impose any constraint on the downstream model. Despite these differences, GPSE can be complementary to traditional SSL to further enhance the prediction performance, for example, by using GPSE encodings as input features to the SSL pre-training.

Why is GPSE transferable to OOD data? The transferability of GPSE to OOD data is uncommon in the graph SSL pre-training literature, particularly for molecular applications. We hypothesize that GPSE's transferability is a consequence of the choice of its predictive self-supervision tasks, which contain a mixture of both local and global intrinsic graph information. This encourages GPSE to capture global invariances using local information, hence allowing it to extract valuable representations on graphs that are different in sizes and connectivity from the training graphs.

When does GPSE help, and when does it not? GPSE provides essential information to the model when the downstream task requires positional or structural information of the graph or better node identifiability in general, which is typically the case for molecular property predictions (Hu et al., 2020b). Conversely, for downstream tasks that do not rely on such information, e.g. protein function prediction using the protein interaction network (Table 6), the benefits from GPSE are not as apparent.