Permutation-Regularized Expander Graph Propagation

Anonymous Authors[1]

Abstract

Expander-based message passing has been proposed as an effective mechanism to mitigate oversquashing in graph neural networks (GNNs) by enabling rapid global information flow. However, fixed expander structures can overfit and generalize poorly on realistic long-range graph benchmarks. We propose Permuted Expander Graph Propagation (P-EGP), a simple modification that introduces controlled permutations of the expander connectivity as a form of structural regularization. Empirically, permutation improves performance at shallow depths, yielding large gains on TreeNeighborMatch and consistent 4–10% absolute improvements over EGP and CGP on Peptides-func and Peptides-struct, with results stable across multiple seeds. At larger depths, we uncover a trade-off between rapid mixing and layer-to-layer coherence, and show that applying permutation selectively—rather than uniformly across layers— recovers and further improves generalization. Overall, P-EGP consistently improves generalization with negligible additional cost (up to constant factors).

1. Introduction

Oversquashing(Alon & Yahav, 2021) in Graph Neural Networks (GNNs) refers to the phenomenon where information from an exponentially expanding k-hop neighborhood must be compressed into fixed-size node representations as messages propagate across layers, causing long-range dependencies to be distorted or lost. This makes long-range reasoning hard because, as the graph distance increases, signals from faraway nodes become indistinguishable and overwhelmed, even if they are theoretically reachable. Simply stacking more layers in vanilla GNNs does not solve this problem: deeper architectures exacerbate oversquashing by funneling more information into the same constrained node representations, while also introducing related issues like oversmoothing, where node embeddings become increasingly similar. As a result, naïvely increasing depth often degrades performance rather than enabling effective global reasoning, highlighting a structural limitation of standard message-passing GNNs rather than a lack of expressive power per layer.

Expanders(Krebs & Shaheen, 2011) are attractive for mitigating oversquashing because they achieve a rare but crucial combination: they are sparse, yet they exhibit strong global connectivity. This means information can propagate between any two nodes in very few hops without creating bottlenecks. Expander Graph Propagation (EGP)(Deac et al., 2022) is a successful instantiation of this idea, introducing fixed auxiliary graphs with expander properties that are interleaved with standard message passing, allowing GNNs to efficiently mix global information while remaining scalable and preprocessing-free. Cayley Graph Propagation (CGP)(Wilson et al., 2025) refines this approach by more faithfully preserving the expander structure: instead of truncating the auxiliary Cayley graph to match the input graph size, CGP propagates over the complete Cayley graph, using additional virtual nodes to retain its theoretical expansion guarantees. This refinement strengthens the bottleneck-free communication promised by expanders, leading to more robust long-range information flow in GNNs.

Despite these advances, we empirically observe that models with fixed expanders achieve high training performance but exhibit weaker generalization and robustness. As a possible explanation, we hypothesize that the fixed expanders introduce a stable shortcut, which may lead to over-specialized representations.

Contributions:

1. We introduce permutation-regularized variants of expander graph propagation with negligible overhead.
2. We achieve large and consistent improvements over EGP and CGP on LRGB and synthetic oversquashing benchmarks.
3. We empirically demonstrate that permutation regularization yields representations more robust to expander

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

removal on real long-range graph datasets.

2. Related Work

2.1. Oversquashing

Oversquashing refers to the failure of graph neural networks (GNNs) to transmit long-range information because signals from an exponentially growing receptive field are compressed into fixed-size node representations; Alon & Yahav formalized this bottleneck and showed it is distinct from over-smoothing, especially in tasks with large problem radii. To diagnose this failure mode, they introduced oversquashing stress tests—controlled synthetic benchmarks (e.g., tree-structured message-passing tasks) that are theoretically solvable but empirically break standard GNNs as depth increases—demonstrating that architectural bottlenecks, rather than optimization issues, fundamentally limit long-range reasoning in GNNs.

2.2. Expander-based propagation

2.2.1. EGP

Expander Graph Propagation (EGP) was proposed as a structural response to over-squashing in message passing GNNs, aiming to enable rapid global information flow without incurring the cost of dense connectivity. The central idea is to interleave standard GNN layers with propagation over a fixed auxiliary expander graph, allowing information to mix globally in a small number of steps.

The design of EGP is guided by several constraints: global propagation in logarithmic depth, resistance to information bottlenecks, linear time and space complexity on sparse graphs, and the absence of input-dependent preprocessing. By relying on precomputed sparse expanders that are independent of the input graph, EGP satisfies these constraints while remaining scalable and simple to integrate into existing GNN architectures.

2.2.2. CGP

Cayley Graph Propagation (CGP) builds on the expander-based message passing framework introduced by EGP by revisiting how Cayley graphs are aligned with the input graph. While EGP constructs a truncated subgraph of a Cayley graph to match the input graph size, CGP observes that this truncation can weaken the desirable expansion properties of the underlying expander. To address this, CGP proposes propagating over the complete Cayley graph structure, introducing additional virtual nodes(Southern et al., 2024; Wilson et al., 2025) where necessary to preserve the graph's theoretical guarantees.

CGP retains the alternating layer schema of EGP — interleaving input-graph and auxiliary-graph propagation, ensuring that message passing occurs on a bottleneck-free expander at each auxiliary layer. This formulation preserves the design goals of EGP, including sparsity, scalability, and preprocessing-free operation, while offering a theoretically grounded alternative for expander-based propagation.

2.2.3. Other graph editing methods

The approaches discussed above focus on auxiliary expander graphs that are used exclusively during message passing, without modifying the topology of the input graph itself. Accordingly, this work focuses exclusively on expander-based message passing and directly builds on Expander Graph Propagation (EGP) and Cayley Graph Propagation (CGP).

Accordingly, we restrict our comparisons to expander-based methods and do not consider general graph rewiring approaches, which are orthogonal to the setting studied here.

3. Method

3.1. Expander Graph Propagation

Notation (following EGP). Let the input graph be defined by node features $X \in \mathbb{R}^{|V|\times d}$ and an adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$, where $x_u \in \mathbb{R}^d$ denotes the feature vector of node $u \in V$. Let $A_{\mathrm{Cay}(n)} \in \mathbb{R}^{|V|\times|V|}$ denote the adjacency matrix induced by a (possibly truncated) Cayley graph constructed with size parameter n. We write a generic graph neural network layer as $\mathrm{GNN}(\cdot,\cdot)$, mapping node features and an adjacency matrix to updated node features.

Expander Graph Propagation alternates layers operating on the input graph and the expander graph. For example, a two-layer block is written as

$$H = \mathrm{GNN}\big(\mathrm{GNN}(X, A),\, A_{\mathrm{Cay}(n)}\big),$$

and this alternating pattern is repeated for a fixed number of layers, with the final node representations used for downstream prediction.

Since our goal is to isolate the effect of the proposed mechanism, we restrict our experiments to the Graph Isomorphism Network (GIN)(Xu et al., 2018). Unless stated otherwise, the term GNN will henceforth refer to a GIN layer. So, for a node u, the update rule is given by

$$h_u = \phi\left((1+\epsilon)x_u + \sum_{v\in\mathcal{N}(u)} x_v\right), \tag{1}$$

where $\mathcal{N}(u)$ denotes the neighborhood of node u, $\epsilon \in \mathbb{R}$ is a learnable parameter, and $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ is implemented as a multilayer perceptron with two hidden layers.

Additionally, our method adopts the same expander graph family introduced in Expander Graph Propagation.

3.2. Permutation-Regularized Expander Propagation

In our model, we retain the alternating two-layer structure of Expander Graph Propagation, but randomise the expander connectivity at each layer via node permutations. A two-layer block is given by,

$$H = \mathrm{GNN}\big(\mathrm{GNN}(X, A),\ P^\top A_{\mathrm{Cay}(n)} P\big)\,,$$

where P is a random permutation matrix applied at every other layer. This operation preserves the expander graph up to isomorphism, while reassigning node-to-node correspondences across layers, thereby preventing a fixed alignment between expander vertices and input graph nodes.

We summarise the resulting forward pass in Algorithm 1.

Algorithm 1 Permutation-Regularized Expander Propagation

Input: Node features $X \in \mathbb{R}^{|V|\times d}$, adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$
Output: Node embeddings H
Choose the smallest expander graph $G^{\mathrm{Cay}(n)}$ with $|V(G^{\mathrm{Cay}(n)})| \geq |V|$
Let $A^{\mathrm{Cay}(n)}$ denote its adjacency matrix
$H^{(0)} \leftarrow X$
for $t = 1 \dots T$ **do**
  if t is odd **then**
    $H^{(t)} \leftarrow \mathrm{GNN}^{(t)}(H^{(t-1)}, A)$ {Propagation over input graph}
  else
    $P \leftarrow$ GENERATEPERMUTATION()
    $H^{(t)} \leftarrow \mathrm{GNN}^{(t)}\Big(H^{(t-1)},\ P^\top A^{\mathrm{Cay}(n)}_{1:|V|,1:|V|} P\Big)$
    {Permuted expander propagation}
  end if
end for
return $H^{(T)}$

Implementation details. In practice, permutation is applied to the expander graph's edge list(Fey et al., 2021) rather than explicitly forming the permuted adjacency matrix $P^\top AP$. Since the Cayley graph is 4-regular, this operation can be performed in $O(|E|) = O(|V|)$ time by relabelling node indices in the adjacency list, preserving the overall linear-time complexity of the method.

For a fixed epoch and mini-batch, given an initial seed s_0, we iteratively apply

$$(P^{(t+1)}, s_{t+1}) = \textsc{GeneratePermutation}(s_t),$$

where s_t denotes the seed at the expander layer t.

3.2.1. VARIANTS

The frequency at which GENERATEPERMUTATION is invoked defines different variants of our method. Specifically, we consider:

- **P-EGP (per-epoch, batch, layer):** a fresh permutation is generated for every combination of epoch, mini-batch, and expander layer;
- **EP-EGP (per-epoch, layer):** a fresh permutation is generated for each epoch and expander layer, and shared across all mini-batches within the epoch;
- **F-EGP (fixed per-layer):** a fixed permutation is generated once per expander layer and reused throughout training.

While **P-EGP** introduces the highest degree of stochasticity and maximally disrupts fixed routing patterns across layers and epochs, this may come at the cost of reduced stability in long-range information propagation. The **EP-EGP** and **F-EGP** variants therefore represent intermediate trade-offs between stochasticity and the ability to learn consistent global structures.

4. Experimental Setup

4.1. Datasets

4.1.1. PEPTIDES

Peptides-func and Peptides-struct are part of the Long Range Graph Benchmark (LRGB)(Dwivedi et al., 2022), which was explicitly designed to evaluate graph models under long-range dependency constraints. In contrast to common molecular benchmarks with small graph diameters, peptide graphs are substantially larger and sparser, with an average shortest path length of approximately 20.9 and an average diameter of about 57, while maintaining a low average degree close to 2. These structural properties make information propagation across distant nodes inherently challenging for local message-passing GNNs and render the datasets particularly sensitive to oversquashing effects. As discussed in the LRGB benchmark, strong performance on these tasks typically requires models to effectively capture and integrate long-range dependencies beyond local neighborhoods.

Notably, Peptides-func (multi-label graph classification) and Peptides-struct (graph regression) share the same underlying molecular graphs and differ only in their prediction tasks, enabling evaluation of whether a method generalizes across distinct learning regimes under identical graph structures. Peptides-func is evaluated using **Average Precision (AP)** across 10 binary labels, while Peptides-struct is evaluated using **Mean Absolute Error (MAE)** for predicting aggregated 3D structural properties.

4.1.2. TREENEIGHBORMATCH

TreeNeighborMatch is a synthetic benchmark introduced by Alon and Yahav(Alon & Yahav, 2021) to explicitly probe the effects of oversquashing in graph neural networks under controlled conditions. The task is constructed such that correct prediction requires aggregating information from an exponentially growing number of leaf nodes toward a single target node, making it inherently sensitive to information bottlenecks as graph depth increases. In this benchmark, **depth** refers to the depth of the underlying tree used to generate the dataset, which directly controls the radius over which information must be propagated.

The task is formulated as a node-level classification problem, where the objective is to predict the label of a designated target node based on matching information propagated from distant leaf nodes. Performance is measured using classification accuracy. Throughout our experiments, **layers** denote the number of message-passing layers in the GNN model, allowing us to study the interaction between architectural depth and dataset-imposed long-range dependencies.

4.2. Models and Baselines

A. Base GNN:
   (a) We use GIN as the backbone
   (b) It is used consistently across all models

B. EGP & CGP
   (a) For Peptides, we compare our models with both the methods.
   (b) For the TreeNeighborMatch benchmark, we use EGP as the primary expander-based baseline and compare it against its permutation-regularized variants to isolate the effect of permutation on expander-based propagation.

C. Permutation based variants
   (a) Randomly reassigns expander connectivity across layers and acts as structural regularization.
   (b) Dataset-specific application: For Peptides, permutations are applied at every expander layer. For TreeNeighborMatch, permutation is applied only at the final expander layer to isolate its effect at the global mixing stage.

4.3. Training and Hyperparameters

4.3.1. PEPTIDES

For the Peptides datasets, we adopt the hyperparameters and training protocol of CGP without modification whenever specified. For settings not explicitly reported in CGP (e.g., batch dimension), we follow the default configuration used in the LRGB benchmark. Additionally, early stopping(Prechelt, 2012) is applied with a patience of 50 epochs to prevent overfitting.

4.3.2. TREENEIGHBORMATCH

For TreeNeighborMatch, we use a computationally feasible training configuration chosen to satisfy time and resource constraints, as the original settings are prohibitively expensive for this benchmark. All hyperparameters are fixed across models and reported in Appendix.

4.4. Evaluation Protocol

1. Official splits and dataloaders were used for both Peptides and TreeNeighborMatch
2. For a fixed dataset, all the experimental conditions (seeds, GPUs, hyperparameters etc) were identical for all models.
3. All results are averaged over 5 random seeds (0–4).

We build on publicly available implementations released by the authors of CGP, the LRGB benchmark, and TreeNeighborMatch.

5. Results and Discussion

5.1. Peptides Results

5.1.1. PEPTIDES-FUNC

Table 1 reports test Average Precision (AP) on the Peptides-func benchmark. Among expander-based methods, all permutation-regularized variants substantially outperform both EGP and CGP under identical training protocols. In particular, F-EGP achieves the best performance with **0.6145 ± 0.0041**, corresponding to an absolute improvement of **+0.112** over EGP and **+0.113** over CGP. This translates to a relative gain of approximately **22%** over both baselines.

Table 1. Peptides-func results (AP ↑).

Model	AP ↑
GIN	0.5572 ± 0.0044
+EGP	0.5023 ± 0.0084
+CGP	0.5016 ± 0.0058
+P-EGP	0.6064 ± 0.0036
+EP-EGP	$\mathit{0.6118 \pm 0.0038}$
+F-EGP	$\mathbf{0.6145 \pm 0.0041}$

5.1.2. PEPTIDES-STRUCT

Table 2 reports test Mean Absolute Error (MAE) on the Peptides-struct benchmark, where lower values indicate bet-

ter performance. All permutation-regularized variants significantly improve over both EGP and CGP. In particular, EP-EGP achieves the lowest error with **0.2670 ± 0.0015 MAE**, compared to **0.3062 ± 0.0068** for EGP and **0.3067 ± 0.0031** for CGP. This corresponds to an absolute error reduction of approximately **0.039 MAE**, or a relative improvement of about **13%** over both baselines.

Table 2. Peptides-struct results (MAE ↓).

Model	MAE ↓
GIN	0.3610 ± 0.0061
+EGP	0.3062 ± 0.0068
+CGP	0.3067 ± 0.0031
+P-EGP	0.2681 ± 0.0017
+EP-EGP	$\mathbf{0.2670 \pm 0.0015}$
+F-EGP	$\mathit{0.2680 \pm 0.0018}$

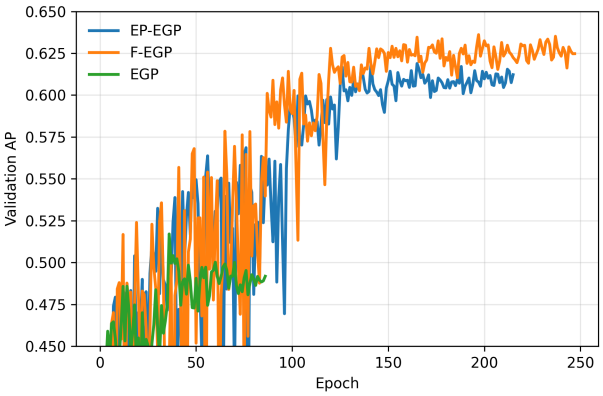


Figure 1. Peptides-func, seed=0

Taken together, the results on Peptides-func and Peptides-struct demonstrate that the proposed permutation-regularized expander propagation consistently improves performance under identical training settings, including the same backbone architecture, hyperparameters, and data splits. Since both tasks are defined over the same underlying graphs but differ in supervision — classification versus regression — the observed gains indicate that the proposed approach yields more accurate and stable representations rather than task-specific improvements. This consistency across learning regimes suggests that permutation regularization provides a generally effective inductive bias for long-range graph learning, rather than benefits tied to a specific task formulation.

5.2. TreeNeighborMatch

TreeNeighborMatch is a controlled synthetic benchmark designed to directly stress oversquashing, as successful prediction requires aggregating information from an exponentially growing receptive field. Increasing network depth exacerbates this bottleneck, making performance highly sensitive to long-range propagation quality. Although TreeNeighborMatch is defined in terms of training accuracy, we report test accuracy to evaluate generalization beyond the training trees. We report results at fixed depth 4 while varying the number of GNN layers.

In this setting, we compare EGP with its permutation-regularized variants. To isolate the effect of permutation at the global mixing stage, permutation is applied only at the final expander layer, while all earlier layers remain identical to EGP. As shown in Table 3, permutation-regularized variants consistently outperform EGP under severe oversquashing conditions, with EP-EGP achieving the strongest gains.

Table 3. TreeNeighborMatch results at fixed depth 4. Higher is better.

Model	Depth $= 4$	
	Layers $= 6$	Layers $= 7$
GIN	0.0954 ± 0.0143	0.1136 ± 0.0295
+EGP	0.3575 ± 0.0508	0.3136 ± 0.0315
+EP-EGP	$\mathbf{0.4593 \pm 0.0508}$	$\mathbf{0.5664 \pm 0.1065}$
+F-EGP	$\mathit{0.4421 \pm 0.0490}$	$\mathit{0.5210 \pm 0.0845}$

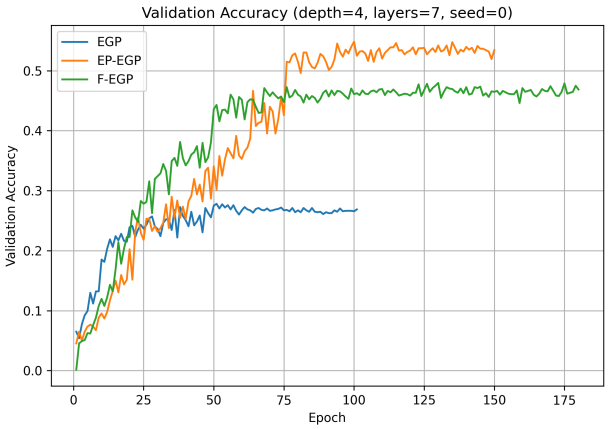


Figure 2. TreeNeighborMatch

5.3. Linear Probe Analysis (Representation Quality)

Motivation. We use linear probing as a diagnostic tool rather than as a new prediction task, with the goal of evaluating representation quality independent of expander-based propagation. Linear probes are a standard technique for this purpose, allowing performance to be attributed to the learned embeddings rather than to architectural components

used during training. In our setting, the probe is used *to isolate representation quality from architectural effects* introduced by auxiliary expander layers(Bechler-Speicher et al., 2025).

Protocol. To evaluate representation quality independently of expander-based propagation, we perform a linear probing analysis following standard practice. For each trained model, we freeze the encoder parameters and disable all expander layers, so that message passing is performed solely over the original input graph edges. Batch normalization layers are fixed in evaluation mode and dropout is disabled to ensure deterministic embeddings.

We then reinitialize and train a single linear classifier on top of the frozen graph-level embeddings, using the same data splits, loss function, and evaluation metric as the original task. During this phase, only the parameters of the linear classifier are updated. Performance is reported on the validation and test sets, without any additional finetuning of the encoder.

Results. Table 4 summarizes linear probe performance on Peptides-func. Permutation-regularized variants achieve consistent absolute gains of approximately **3–4%** over EGP and CGP in probe accuracy. These improvements are stable across random seeds and evaluation splits, indicating that the effect is not driven by variance in training. Overall, the results indicate higher-quality representations for permutation-regularized models under a fixed linear readout.

Table 4. Linear probe results on Peptides-func (AP ↑).

Model	Test Probe AP
EGP	0.3725 ± 0.0099
CGP	0.3697 ± 0.0064
P-EGP	$\mathit{0.4050 \pm 0.0087}$
EP-EGP	$\mathbf{0.4105 \pm 0.0088}$
F-EGP	$\mathit{0.4050 \pm 0.0055}$

Interpretation. These findings suggest that permutation regularization acts as a form of structural regularization, reducing sensitivity to specific expander alignments and yielding representations that generalize beyond the training-time propagation structure. The benefits persist even when the auxiliary expander graph is removed, indicating that permutation regularization improves the learned representations themselves rather than merely aiding propagation at inference time.

5.4. Discussion

Across both real-world molecular benchmarks and controlled synthetic stress tests, permutation-regularized expander propagation consistently improves performance in settings that require long-range information integration. On the Peptides benchmarks, these gains are observed under identical training protocols and persist across both classification and regression tasks, indicating improved robustness beyond task-specific effects. On TreeNeighborMatch, permutation regularization further improves performance as depth increases, suggesting enhanced stability under extreme oversquashing conditions.

Taken together, these results support the view that introducing controlled stochasticity into expander-based propagation mitigates sensitivity to fixed auxiliary routing patterns. Rather than relying on a single deterministic expander structure across layers, varying expander alignments appears to act as an effective form of structural regularization, improving generalization in long-range graph learning scenarios. Importantly, these benefits are achieved without increasing model complexity or altering the underlying input graph structure.

6. Limitations

- **Lack of theoretical guarantees.** This work is empirical in nature and does not provide formal theoretical analysis or guarantees explaining when or why permutation-regularized expander propagation should be optimal. While the observed trends are consistent across multiple benchmarks, developing a principled theoretical understanding of permutation effects remains an open direction.

- **Limited model backbones.** Experiments are conducted using a restricted set of GNN backbones and message-passing architectures. Although results are consistent across these settings, it remains to be studied how permutation regularization interacts with a broader class of architectures, including attention-heavy or non-message-passing models.

- **Fixed expander family.** We focus on a single family of expander constructions, following prior work on EGP and CGP. Other expander families or alternative auxiliary graph constructions may exhibit different behavior under permutation, which we do not explore in this study.

- **No hyperparameter search.** To ensure fair comparison, all methods are evaluated under identical hyperparameters inherited from prior work. While this isolates the effect of permutation, it may understate the potential performance of individual variants under task-specific tuning.

7. Conclusion

We studied the limitations of fixed expander-based propagation for long-range graph learning and proposed permutation-regularized variants that introduce controlled stochasticity without altering model complexity or input graph structure. Across real-world molecular benchmarks from the Long Range Graph Benchmark and the TreeNeighborMatch stress test, permutation regularization consistently improves performance in regimes that require long-range information integration. Additional analysis further indicates that these gains are not solely attributable to expander-based propagation at inference time, but reflect improvements in the learned representations.

Overall, our results suggest that varying expander alignments provides a simple and effective form of structural regularization for expander-augmented GNNs. An important direction for future work is to develop a theoretical understanding of how permutation interacts with auxiliary graph structure and to explore adaptive strategies for expander construction in long-range graph settings.

Impact Statement

This work proposes a modification to expander-based graph neural network architectures that improves robustness and generalization in long-range graph learning tasks. By introducing permutation-based regularization into auxiliary expander propagation, the method enhances representation quality without increasing model complexity or altering the underlying input graph structure.

The proposed approach is intended to support more reliable learning in applications that require long-range information integration, such as molecular property prediction and structured reasoning on large graphs. Since the method does not rely on task-specific preprocessing or domain-sensitive assumptions, it is broadly applicable across graph learning settings.

Potential negative impacts are limited. The method does not introduce new data sources, does not alter the semantics of input graphs, and does not directly affect decision-making systems involving humans. As with any machine learning model, misuse or deployment without appropriate validation could lead to erroneous predictions, but this risk is not unique to the proposed approach. We do not anticipate broader societal harms arising from this work.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications, 2021. URL https://arxiv.org/abs/2006.05205.

Bechler-Speicher, M., Finkelshtein, B., Frasca, F., Müller, L., Tönshoff, J., Siraudin, A., Zaverkin, V., Bronstein, M. M., Niepert, M., Perozzi, B., Galkin, M., and Morris, C. Position: Graph learning will lose relevance due to poor benchmarks. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. URL https://openreview.net/forum?id=nDFpl2lhoH.

Deac, A., Lackenby, M., and Veličković, P. Expander graph propagation, 2022. URL https://arxiv.org/abs/2210.02997.

Dwivedi, V. P., Rampášek, L., Galkin, M., Sanyal, S., Luan, Y., Khan, S., Zhang, B., Zhang, M., Hooi, B., Günnemann, S., et al. Long range graph benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Fey, M., Sunil, J., Nitta, A., Puri, R., Shah, M., Stojanović, B., and Rozemberczki, B. PyG 2.0: Scalable learning on real world graphs. *arXiv preprint arXiv:2110.02905*, 2021. URL https://arxiv.org.

Krebs, M. and Shaheen, A. *Expander families and Cayley graphs: a beginner's guide*. Oxford University Press, 2011.

Prechelt, L. *Early Stopping — But When?*, pp. 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_5. URL https://doi.org/10.1007/978-3-642-35289-8_5.

Southern, J., Di Giovanni, F., Bronstein, M., and Lutzeyer, J. F. Understanding virtual nodes: Oversquashing and node heterogeneity. *arXiv preprint arXiv:2405.13526*, 2024.

Wilson, J., Bechler-Speicher, M., and Veličković, P. Cayley graph propagation, 2025. URL https://arxiv.org/abs/2410.03424.

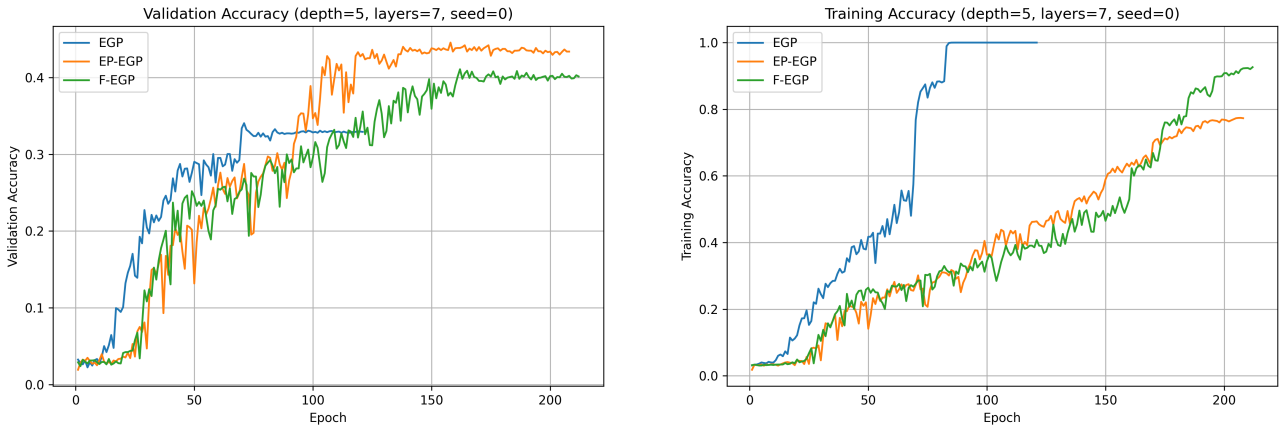
Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

A. TreeNeighborMatch, hyperparameters at depth=4

Models are trained for 200 epochs using the Adam optimizer with a learning rate of 10^{-3}, weight decay 5×10^{-4}, and batch size 1024. A ReduceLROnPlateau scheduler is used with mode `max`, threshold mode `abs`, reduction factor 0.5, and patience of 10 epochs. Early stopping is applied with a patience of 50 epochs. The hidden dimension is set to 64, dropout is disabled (0.0), and a single training iteration is used.

B. Train vs Validation Accuracy at depth=5

In this section we notice that EGP achieves higher train accuracy even at depth=5, whereas our variants have a slower training curve but the validation accuracy is higher. The hyperparameters are same as that of depth=4, except we increased the number of epochs from 200 to 400 and hidden dimension from 64 to 128.



(a) Validation accuracy

(b) Training accuracy

Figure 3. TreeNeighborMatch performance during training and validation.