Small Language Models as Graph Classifiers: Evaluating and Improving Permutation Robustness

Michal Podstawski *

Posted Date: 25 February 2026

doi: 10.20944/preprints202602.0946.v2

Keywords: graph classification; small language models; graph representation learning

Article

Small Language Models as Graph Classifiers: Evaluating and Improving Permutation Robustness

Michal Podstawski

NASK National Research Institute, Warsaw, Poland; michal.podstawski@nask.pl

Abstract

Graph classification is dominated by permutation-invariant graph neural networks. We revisit this problem from a different perspective: can small language models (SLMs) act as graph classifiers when graphs are serialized as text? Unlike GNNs, sequence-based transformers do not encode permutation invariance by construction, raising a fundamental question about structural stability under node relabeling. We provide the first systematic study of permutation robustness in small graph-as-text models. We introduce an evaluation protocol based on Flip Rate and KL-to-Mean divergence to quantify prediction instability across random node permutations. To enforce structural consistency, we propose Permutation-Invariant Training (PIT), a multi-view regularization scheme that aligns predictions across relabeled graph views, and examine its interaction with degree-aware token embeddings as a minimal inductive bias. Across benchmark datasets using parameter-efficient fine-tuning, we show that SLMs achieve competitive classification accuracy, yet standard fine-tuning exhibits non-trivial permutation sensitivity. PIT consistently reduces instability and in most evaluated settings improves accuracy, demonstrating that structural invariance in sequence-based graph models can emerge through explicit regularization.

Keywords: graph classification; small language models; graph representation learning

---

1. Introduction

Graphs are a fundamental representation for structured data across domains including chemistry, biology, social networks, and program analysis. Graph classification - the task of assigning a label to an entire graph - is traditionally addressed using graph neural networks (GNNs), which explicitly encode permutation-invariant inductive biases through message passing and aggregation mechanisms. These architectural constraints ensure that graph predictions are invariant under node relabeling, a core structural property of graphs.

Recently, transformer-based language models have demonstrated surprising versatility beyond natural language processing. In particular, there has been growing interest in applying language models to structured inputs by serializing non-textual data into textual form. In the graph domain, this approach - often referred to as *graph-as-text* - represents nodes and edges as sequences of tokens and applies standard language model fine-tuning procedures. This paradigm raises a fundamental question:

> To what extent can small language models act as graph classifiers when graphs are represented as text, and how robust are such models to graph symmetries?

Unlike GNNs, language models do not inherently encode permutation invariance. When graphs are serialized, the ordering of node identifiers and adjacency lists becomes part of the textual input. Consequently, two isomorphic graphs with different node labelings may yield different serialized sequences. If a graph-as-text model relies on serialization artifacts, its predictions may vary under node relabeling-even though the underlying graph is unchanged.

 doi:10.20944/preprints202602.0946.v2

In this work, we systematically investigate permutation robustness in small language models (SLMs) used for graph classification. We formulate graph classification as next-token prediction: after a serialized adjacency-list representation, the model predicts a single class token. We employ parameter-efficient fine-tuning via LoRA and evaluate performance using stratified cross-validation on benchmark graph datasets.

Beyond accuracy, we introduce an explicit evaluation protocol for permutation robustness. For each graph, we generate multiple random node relabelings and measure: (i) *Flip Rate*, the fraction of relabelings that change the predicted class; and (ii) *KL-to-Mean divergence*, which quantifies confidence instability across relabelings. These metrics allow us to analyze not only whether graph-as-text models are accurate, but whether their predictions are structurally stable.

We compare two strategies for improving permutation robustness: (1) *Permutation - Invariant Training (PIT)*, which uses multi-view consistency regularization across relabeled graph views; and (2) a *minimal structural inductive bias*, implemented as degree-aware token embeddings that inject permutation-invariant structural information into the model.

Our empirical results reveal a nuanced picture. On simpler graph datasets, consistency-based regularization is sufficient to substantially reduce permutation sensitivity while improving accuracy. On more complex datasets, combining structural bias with consistency training further improves robustness and predictive performance. In contrast, structural bias alone is insufficient to guarantee invariance.

These findings provide three key insights. First, small language models can serve as competitive graph classifiers under adjacency-list serialization. Second, permutation robustness does not automatically emerge from fine-tuning and must be explicitly addressed. Third, output-level regularization and architectural inductive bias interact in nontrivial ways, revealing an accuracy–permutation tradeoff in graph-as-text models.

By quantifying this tradeoff and evaluating minimal interventions, this work contributes to understanding how structural invariance can be enforced in compact transformer architectures operating on serialized graphs.

Theoretical Perspective on Emergent Invariance.

From a theoretical standpoint, permutation invariance in graph learning can be viewed as a symmetry constraint over the input space: for any permutation π acting on node indices, the target function must satisfy $f(G) = f(\pi(G))$. In graph neural networks, this constraint is enforced architecturally through symmetric aggregation operators. In contrast, transformer-based language models operating on serialized graphs do not encode this symmetry *a priori*; instead, invariance must emerge, if at all, from the training objective and data distribution.

Multi-view consistency training can be interpreted as minimizing an empirical estimate of prediction variance over the orbit of a graph under the permutation group. In this sense, Permutation-Invariant Training (PIT) approximates group-averaged risk minimization, encouraging the learned classifier to collapse permutation-equivalent representations into a shared decision region. However, because the model processes ordered sequences, invariance is not guaranteed outside the distribution of sampled permutations. Structural inductive biases, such as degree-aware embeddings, partially restrict the hypothesis space by injecting permutation-invariant signals, thereby reducing the burden on optimization. Together, these mechanisms suggest that permutation invariance in sequence-based graph learners is not an inherent architectural property but an emergent regularized symmetry shaped by objective design and inductive bias.

2. Related Work

2.1. Graph Neural Networks and Structural Inductive Bias

Graph Neural Networks (GNNs) have become the dominant paradigm for graph representation learning due to their inherent permutation invariance. Architectures such as Graph Convolutional Net-

 doi:10.20944/preprints202602.0946.v2

works (GCN) [1], GraphSAGE [2], and the Graph Isomorphism Network (GIN) [3] achieve invariance through symmetric neighborhood aggregation. These models incorporate graph-specific inductive bias directly into the architecture.

More recently, transformer-based models have been adapted to graphs. For instance, Graphormer [5] introduces structural encodings such as node degrees and shortest-path distances as additive biases in attention, significantly improving graph transformer performance. These approaches demonstrate that structural inductive bias plays a crucial role in graph learning.

In contrast, our work investigates graph classification in a *graph-as-text* setting, where no explicit permutation-invariant mechanism is built into the architecture. This raises the question of whether invariance can emerge through fine-tuning or must be explicitly encouraged.

2.2. Large Language Models for Graph-Structured Data

Recent work explores applying large language models (LLMs) to graph-structured problems by serializing graphs into textual sequences. GraphGPT [9] aligns LLMs with graph tasks through instruction tuning.

While these works primarily focus on reasoning capabilities or instruction alignment, systematic analysis of *permutation robustness* in graph-as-text classification settings remains limited, especially for compact language models fine-tuned using parameter-efficient methods.

Our study differs in three respects: (i) we focus on small language models rather than frontier-scale LLMs, (ii) we evaluate graph classification under stratified cross-validation on TU benchmarks [4], and (iii) we explicitly quantify permutation sensitivity under random node relabeling.

3. Method

3.1. Problem Formulation

Let $G = (V, E, X)$ denote a graph with node set V, edge set E, and node features X. In graph classification, the objective is to learn a function $f(G)$ that predicts a graph-level label y.

A fundamental property of graph learning is permutation invariance: for any node relabeling (permutation) π,

$$f(G) = f(\pi(G)).$$

In a graph-as-text setting, however, this invariance is not guaranteed, since different node orderings produce different serialized sequences.

Our goal is to evaluate and improve permutation robustness of small language models used for graph classification via textual serialization.

3.2. Graph-to-Text Serialization

Each graph is converted into an adjacency-list textual representation:

```
GRAPH n=|V| m=|E|
NODE u <node_repr(u)> : v1 v2 ... vk
...
```

Node representations include either: (i) discrete node labels encoded as `lab=<id>`, or (ii) truncated continuous feature vectors formatted as `feat=[...]`.

This format exposes both topology and node attributes while remaining compatible with standard transformer language models.

3.3. Classification via Next-Token Prediction

Graph classification is cast as next-token prediction. Given a serialized graph $\mathcal{S}(G)$, we append the prompt suffix:

```
Class:
```

The model predicts a single-token label (e.g., “0” or “1”).

 doi:10.20944/preprints202602.0946.v2

Let $h(G)$ denote the hidden state at the final token position. Class probabilities are computed as:

$$p(y \mid G) = \text{softmax}(Wh(G)),$$

where W selects logits corresponding to label tokens.

Training minimizes class-weighted cross-entropy:

$$\mathcal{L}_{\text{sup}} = -\sum_i w_{y_i} \log p(y_i \mid G_i),$$

where weights w_{y_i} compensate for class imbalance within each fold.

3.4. Parameter-Efficient Fine-Tuning

We fine-tune pretrained small language models using Low-Rank Adaptation (LoRA) [6]. LoRA modules are applied to attention and MLP projection layers. All base model parameters remain frozen, and only low-rank adapters are trained.

Quantized 4-bit loading (QLoRA-style [7]) is used to reduce memory footprint, allowing experimentation on modest hardware.

3.5. Permutation-Invariant Training (PIT)

To encourage robustness under node relabeling, we introduce Permutation-Invariant Training (PIT).

For each graph, we generate K random relabeled views:

$$\{G^{(1)}, \ldots, G^{(K)}\}.$$

In addition to supervised loss, we enforce prediction consistency across views using KL divergence to the mean distribution:

$$\mathcal{L}_{\text{cons}} = \frac{1}{K} \sum_{k=1}^{K} \text{KL}\Big(p(y \mid G^{(k)}) \parallel \bar{p}(y)\Big),$$

where $\bar{p}(y)$ is the average prediction across views.

The total training objective becomes:

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{cons}},$$

with λ controlling consistency strength.

3.6. Structural Bias: Degree-Aware Token Embedding

We further investigate minimal structural inductive bias through degree-aware token embeddings.

For each node u, its degree $\deg(u)$ is bucketed into discrete intervals. All tokens corresponding to the line

```
NODE u ...
```

receive an additional learned embedding $e_{\deg(u)}$ added to the token embedding:

$$\tilde{x}_t = x_t + e_{\deg(u)}.$$

This modification preserves the transformer architecture while injecting permutation-invariant structural information.

3.7. Permutation Robustness Metrics

To quantify permutation sensitivity, we evaluate each trained model under multiple random node relabelings.

 doi:10.20944/preprints202602.0946.v2

For each graph G, we generate R relabeled variants and compute:

Flip Rate.

Let $\{\hat{y}^{(1)}, \ldots, \hat{y}^{(R)}\}$ denote the predicted class labels obtained from R independently relabeled versions of a graph G. Let

$$\hat{y}^{\star} = \text{mode}\left(\hat{y}^{(1)}, \ldots, \hat{y}^{(R)}\right)$$

denote the modal (most frequent) predicted label across relabelings.[1]

The Flip Rate of G is defined as

$$\text{FlipRate}(G) = \frac{1}{R} \sum_{r=1}^{R} \mathbf{1}\left[\hat{y}^{(r)} \neq \hat{y}^{\star}\right],$$

where $\mathbf{1}[\cdot]$ denotes the indicator function.

Flip Rate measures the fraction of node relabelings that produce a prediction different from the modal class, thereby quantifying decision instability under permutation.

KL-to-Mean

Confidence instability measured by:

$$\frac{1}{R} \sum_{r=1}^{R} \text{KL}\left(p^{(r)}(y) \parallel \bar{p}(y)\right),$$

where $\bar{p}(y)$ is the mean predicted distribution.

All results are reported as mean $\pm$ standard deviation across cross-validation folds.

4. Experiments and Results

4.1. Datasets

We evaluate on benchmark graph classification datasets from the TU collection [4]. In this study, we report results across all considered datasets, which differ in graph size, average node degree, feature availability, and structural variability. The collection spans datasets containing relatively small graphs with limited structural diversity as well as datasets composed of larger graphs exhibiting more complex and heterogeneous connectivity patterns. This diversity enables a comprehensive evaluation of model performance under varying structural regimes. Our framework is dataset-agnostic and readily extends to additional graph classification datasets beyond those reported here.

4.2. Base Model

All experiments use pretrained small instruction-tuned language models as backbone architectures. In particular, we employ **Qwen2.5-1.5B-Instruct** [10] and **Llama-3.2-1B-Instruct** [12] as base models. We select instruction-tuned variants because graph classification is formulated as prompt completion (next-token prediction of a class label), and instruction tuning improves stability and consistency in conditional generation settings.

Unless otherwise stated, experimental settings and fine-tuning procedures are kept identical across backbone models to ensure fair comparison.

4.3. Experimental Protocol

All experiments use stratified K-fold cross-validation (default $K = 10$). Within each fold, a validation subset is extracted from the training partition for model selection. We report mean $\pm$ standard deviation across folds.

We evaluate four configurations:

[1] In the rare case of ties, one of the tied labels is selected arbitrarily.

 doi:10.20944/preprints202602.0946.v2

1. **Baseline**: single-view fine-tuning without consistency regularization.
2. **PIT**: multi-view consistency training with $\lambda > 0$.
3. **Degree Bias**: degree-aware token embedding without consistency regularization.
4. **Degree Bias + PIT**: combination of structural bias and multi-view consistency.

All models are fine-tuned using LoRA adapters. Training hyperparameters are kept fixed across configurations for fairness.

4.4. GNN Baselines

To provide a permutation-invariant reference, we evaluate four standard message-passing architectures: GCN [1], GraphSAGE [2], GIN [3], and GAT [11]. These models perform symmetric neighborhood aggregation and are therefore invariant to node relabeling by construction, in contrast to our graph-as-text language models.

All models use three message-passing layers with hidden dimension 128, ReLU activations, dropout 0.5, and a two-layer MLP classifier. Graph-level representations are obtained via global mean pooling for GCN, GraphSAGE, and GAT. For GIN, we employ global sum pooling, consistent with its original formulation and common TU benchmark practice. Each GIN layer uses a two-layer MLP update function, and GAT uses four attention heads. Hyperparameters are kept fixed across datasets to avoid dataset-specific tuning bias.

When node attributes are unavailable, node degree is used as a simple structural feature.

Training follows the same stratified 10-fold cross-validation protocol as in our language-model experiments to ensure direct comparability. Within each fold, 10% of the training portion is reserved for validation, and early stopping is performed based on validation accuracy. Models are optimized using AdamW with learning rate 10^{-3} and weight decay 10^{-4}. We report mean test accuracy and standard deviation across folds.

These architectures represent widely adopted TU baselines and serve as structurally invariant performance references for assessing the accuracy–robustness tradeoffs of small language models.

Table 1. 10-fold cross-validation accuracy (mean ± std) across datasets.

Model	PROTEINS	MUTAG	BZR	PTC_MR
GNN Baselines				
GCN	0.7071 ± 0.0466	0.6865 ± 0.0961	0.8099 ± 0.0419	0.5608 ± 0.0517
GIN	0.7134 ± 0.0372	0.7822 ± 0.0795	0.7951 ± 0.0194	0.5810 ± 0.0508
SAGE	0.6901 ± 0.0767	0.7018 ± 0.0731	0.7924 ± 0.0416	0.5603 ± 0.0809
GAT	0.7035 ± 0.0579	0.6754 ± 0.0840	0.7927 ± 0.0149	**0.5903** ± 0.0495
Small Language Models (Graph-as-Text)				
Qwen2.5-1.5B-Instruct				
Baseline	0.6775 ± 0.0731	0.8085 ± 0.0715	0.7684 ± 0.1028	0.5469 ± 0.0744
PIT	0.7332 ± 0.0385	**0.8632** ± 0.0721	0.8052 ± 0.0363	0.5588 ± 0.0409
Degree	0.7017 ± 0.0534	0.7982 ± 0.0769	0.6498 ± 0.1490	0.5231 ± 0.0673
Degree + PIT	**0.7509** ± 0.0320	0.8148 ± 0.1228	0.8032 ± 0.0401	0.5663 ± 0.0208
Llama-3.2-1B-Instruct				
Baseline	0.6909 ± 0.0414	0.8082 ± 0.0366	0.7401 ± 0.0876	0.5292 ± 0.0508
PIT	0.7284 ± 0.0376	0.8058 ± 0.0684	0.8100 ± 0.0412	0.5774 ± 0.0798
Degree	0.6963 ± 0.0345	0.7982 ± 0.0932	0.7135 ± 0.1001	0.5229 ± 0.0719
Degree + PIT	0.7375 ± 0.0361	0.8315 ± 0.0569	**0.8144** ± 0.0270	0.5765 ± 0.0627

 doi:10.20944/preprints202602.0946.v2

Table 2. Permutation robustness metrics across datasets.

Method	PROTEINS		MUTAG		BZR		PTC_MR	
	Flip	KL	Flip	KL	Flip	KL	Flip	KL
Qwen2.5-1.5B-Instruct								
Baseline	0.0805	0.0818	0.0313	0.0361	0.0872	0.1037	0.1388	0.1420
PIT	0.0190	0.0055	0.0087	0.0003	0.0082	0.0017	0.0388	0.0015
Degree	0.0897	0.0807	0.0983	0.1374	0.1430	0.1464	0.1851	0.1605
Degree + PIT	0.0140	0.0043	0.0168	0.0043	0.0239	0.0053	0.0588	0.0062
Llama-3.2-1B-Instruct								
Baseline	0.1163	0.1380	0.0513	0.0722	0.0781	0.1320	0.2708	0.3385
PIT	0.0151	0.0030	0.0039	0.0019	0.0075	0.0018	0.0378	0.0024
Degree	0.1089	0.1084	0.1202	0.1427	0.1263	0.1773	0.2500	0.2687
Degree + PIT	0.0177	0.0044	0.0367	0.0097	0.0252	0.0143	0.1062	0.0110

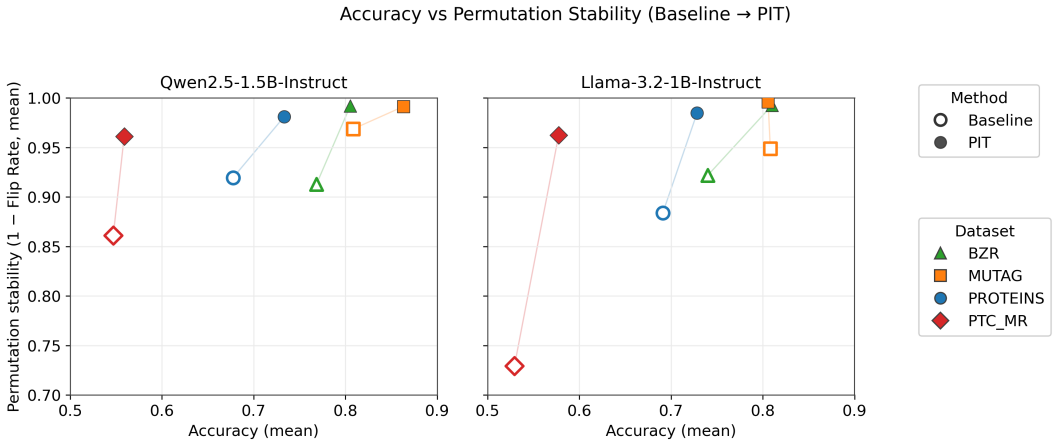

Figure 1. Accuracy versus permutation stability (1 – Flip Rate) for Baseline and PIT across datasets. Hollow markers denote standard fine-tuning (Baseline), while filled markers denote Permutation-Invariant Training (PIT). Line segments connect paired results on the same dataset, illustrating the effect of PIT. Higher values on both axes indicate better performance.

4.5. Analysis

Accuracy–Robustness Tradeoff.

Across datasets and backbone models, a consistent pattern emerges: permutation robustness does not arise automatically from standard fine-tuning, even when classification accuracy is competitive. Baseline graph-as-text models often achieve strong predictive performance, particularly on structurally simple datasets such as MUTAG, yet exhibit non-negligible Flip Rates and elevated KL-to-Mean divergence. This indicates that correct predictions may still depend on serialization artifacts rather than purely structural information.

Permutation-Invariant Training (PIT) substantially reduces both Flip Rate and confidence instability across all evaluated datasets. In several cases, robustness often improves several-fold and, in some cases, approaches an order-of-magnitude reduction relative to the baseline. Importantly, these gains do not come at the expense of predictive performance. On the contrary, PIT frequently improves mean cross-validation accuracy. This suggests that enforcing consistency across relabeled views acts as a beneficial regularizer, encouraging the model to rely on structural signals that generalize across permutations rather than memorizing idiosyncratic token orderings.

Dataset-Dependent Effects.

The magnitude of robustness improvements varies across datasets. On structurally simple datasets such as MUTAG, baseline models already exhibit relatively low permutation sensitivity, and

 doi:10.20944/preprints202602.0946.v2

PIT primarily refines stability while modestly increasing accuracy. In contrast, on more structurally diverse datasets such as PROTEINS and BZR, baseline models display substantially higher Flip Rates and KL divergence. In these regimes, PIT produces pronounced reductions in permutation sensitivity and often yields larger accuracy gains.

This pattern suggests that structural heterogeneity amplifies serialization-induced variance. When graph topology is simple and graphs are small, multiple permutations may lead to similar token-level patterns, limiting instability. As structural complexity increases—through larger graphs, higher degree variability, or richer connectivity patterns—the space of possible serializations expands, increasing the likelihood that the model exploits order-specific correlations unless explicitly regularized.

Role of Structural Inductive Bias.

Degree-aware token embeddings alone do not reliably improve robustness. In several cases, structural bias without consistency regularization yields comparable or even worse Flip Rates relative to the baseline. This indicates that injecting permutation-invariant features into token embeddings is insufficient to override the model's sensitivity to sequence ordering.

However, when combined with PIT, structural bias consistently strengthens robustness on more complex datasets. The combination often produces the lowest Flip Rates and KL divergence values while maintaining or improving classification accuracy. This interaction suggests that structural bias and consistency regularization address complementary aspects of the invariance problem. PIT aligns output distributions across permuted views, while degree-aware embeddings reduce representational variance at the input level. Together, they constrain both the hypothesis space and the optimization trajectory.

Backbone Model Comparison.

Both Qwen2.5-1.5B-Instruct and Llama-3.2-1B-Instruct exhibit qualitatively similar behavior under permutation perturbations. Although absolute accuracy differs slightly across datasets, the effect of PIT is consistent across backbones: permutation sensitivity decreases dramatically once multi-view regularization is introduced. This consistency suggests that permutation instability is not idiosyncratic to a specific architecture but rather a general property of sequence-based models applied to serialized graphs.

Moreover, the fact that similar robustness patterns appear across two independently developed pretrained models indicates that invariance does not spontaneously emerge during instruction tuning or pretraining on natural language. Instead, invariance must be explicitly encouraged during task-specific fine-tuning.

Comparison to GNN Baselines.

Graph neural networks serve as a permutation-invariant reference, as their message-passing and aggregation operators guarantee invariance by construction. While GNNs remain competitive and often strong on TU benchmarks, small language models trained with PIT approach comparable accuracy levels on several datasets. This demonstrates that explicit architectural invariance is not strictly necessary for competitive performance, provided that appropriate regularization is applied.

However, unlike GNNs, invariance in graph-as-text models is approximate rather than guaranteed. Robustness depends on the diversity of sampled permutations during training and may not generalize to unseen relabelings outside the training distribution. This distinction highlights a fundamental difference between architectural and regularized invariance.

Emergent Regularized Symmetry.

Taken together, the empirical findings support the view that permutation invariance in sequence-based graph learners is an emergent property induced by training objectives rather than a built-in symmetry of the architecture. Multi-view consistency training approximates risk minimization over permutation orbits, encouraging the classifier to collapse equivalent serialized representations into

 doi:10.20944/preprints202602.0946.v2

a shared decision region. Structural bias further reduces representational variability but cannot, in isolation, enforce invariance.

These observations reveal a nuanced accuracy–permutation tradeoff. Without explicit regularization, models may exploit serialization-specific cues that improve apparent accuracy while reducing structural stability. Enforcing permutation consistency aligns predictions with graph symmetries and often improves generalization, particularly in structurally heterogeneous settings.

5. Conclusion

We investigated graph classification using small transformer language models in a graph-as-text setting, where permutation invariance is not encoded architecturally but must instead emerge through training. Beyond predictive accuracy, we introduced explicit permutation robustness metrics—Flip Rate and KL-to-Mean divergence—to quantify sensitivity to node relabeling and expose instability that standard evaluation protocols overlook.

Across TU benchmark datasets, we observed that standard fine-tuning can yield competitive accuracy while still exhibiting non-negligible permutation sensitivity, particularly on structurally complex graphs. Permutation-Invariant Training (PIT), implemented via multi-view consistency regularization, consistently and substantially improves robustness, often while simultaneously improving classification accuracy. On simpler datasets, PIT alone is sufficient to achieve near-perfect permutation stability. On more heterogeneous datasets, combining PIT with minimal structural inductive bias further strengthens the accuracy–robustness tradeoff. Structural bias in isolation, however, does not guarantee invariance.

These findings suggest that permutation invariance in sequence-based graph learners is not an inherent architectural property but an emergent symmetry shaped by objective design and inductive bias. While architectural approaches such as GNNs guarantee invariance by construction, our results demonstrate that compact transformer models can approximate this symmetry through regularization, albeit without formal guarantees beyond the sampled permutation distribution.

More broadly, this work highlights a fundamental distinction between architectural and regularized invariance. In sequence models operating on serialized structured data, symmetry must be explicitly evaluated and encouraged. Our framework provides both practical guidance for designing robust graph-as-text classifiers and a methodological template for studying invariance emergence in transformer-based models. Future work will investigate larger model scales, richer structural encodings, and theoretical characterizations of invariance under group-averaged training objectives.

Acknowledgments: This manuscript acknowledges the use of ChatGPT [13], powered by the GPT-5 language model developed by OpenAI, to improve language clarity, refine sentence structure, and enhance overall writing precision.

References

1. Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. In: ICLR (2017)
2. William L. Hamilton, Rex Ying, Jure Leskovec: Inductive Representation Learning on Large Graphs. In: NeurIPS (2017)
3. Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka: How Powerful are Graph Neural Networks? In: ICLR (2019)
4. Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, Marion Neumann: TUDataset: A Collection of Benchmark Datasets for Learning with Graphs. In: ICML Workshop on Graph Representation Learning (2020)
5. Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, Tie-Yan Liu: Do Transformers Really Perform Badly for Graph Representation? In: NeurIPS (2021)
6. Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen: LoRA: Low-Rank Adaptation of Large Language Models. In: ICLR (2022)

 doi:10.20944/preprints202602.0946.v2

7. Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer: QLoRA: Efficient Finetuning of Quantized LLMs. In: NeurIPS (2023)
8. Samuli Laine, Timo Aila: Temporal Ensembling for Semi-Supervised Learning. In: ICLR (2017)
9. Jiabin Tang and Yuhao Yang and Wei Wei and Lei Shi and Lixin Su and Suqi Cheng and Dawei Yin and Chao Huang: GraphGPT: Graph Instruction Tuning for Large Language Models. arXiv preprint arXiv:2310.13023 (2023)
10. Qwen Team: Qwen2.5 Technical Report arXiv preprint arXiv:2412.15115 (2022)
11. Petar Veličković and Guillem Cucurull and Arantxa Casanova and Adriana Romero and Pietro Liò and Yoshua Bengio: Graph Attention Networks. In: International Conference on Learning Representations (ICLR) (2018).
12. Meta Llama Team: The Llama 3 Herd of Models arXiv preprint arXiv:2407.21783 (2024)
13. OpenAI: ChatGPT https://openai.com/chatgpt