TOWARDS INVARIANCE TO NODE IDENTIFIERS IN GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Message-Passing Graph Neural Networks (GNNs) are known to have limited expressive power, due to their message passing structure. One mechanism for circumventing this limitation is to add unique node identifiers (IDs), which break the symmetries that underlie the expressivity limitation. In this work, we highlight a key limitation of the ID framework, and propose an approach for addressing it. We begin by observing that the final output of the GNN should clearly not depend on the specific IDs used. We then show that in practice the network does in fact depend on the IDs, which is undesirable. Invariance to node IDs may be enforced in several ways, and we discuss their theoretical properties. We then propose a novel regularization method that effectively enforces ID invariance to the network. Extensive evaluations on both real-world and synthetic tasks demonstrate that our approach significantly improves ID invariance and, in turn, often boosts generalization performance.

1 INTRODUCTION

Graph Neural Networks (GNNs), and in particular Message-Passing Graph Neural Networks (MPGNNs) (Morris et al., 2019) are limited in their expressive power because they may represent distinct graphs in the same way (Garg et al., 2020). However, when unique node identifiers (IDs) are provided, this limitation is overcome and the MPGNNs become Turing complete (Loukas, 2020; Garg et al., 2020).

There are several ways to attach unique IDs to nodes. A simple approach is to add features with random values as identifiers and resample them during the training process (Sato et al., 2021; Abboud et al., 2021). This strategy, which we refer to as Random Node Initialization (RNI), almost surely assigns unique IDs to each node. Consequently, RNIs transform MPGNNs into universal approximators of invariant and equivariant graph functions (Abboud et al., 2021). Nonetheless, RNI has been shown to not improve generalization on real-world datasets (Murphy et al., 2019; You et al., 2021; Cantürk et al., 2024; Eliasof et al., 2024; Papp et al., 2021; Bevilacqua et al., 2022; Eliasof et al., 2023).

Note that although IDs are used as GNN features, specific ID values should not affect network output. Namely, two different assignments to the IDs should ideally result in the same output. Otherwise, different ID assignments would result in different predictions for the graph label, and therefore at least one of them would be wrong. We refer to this desirable property as "invariance to ID values". Intuitively, if such an invariance does not hold, generalization is negatively impacted.

Surprisingly, we find that in practice this invariance often does not hold for GNNs learned with RNIs. This is shown across multiple GNN architectures, real-world, and synthetic datasets, even when resampling IDs per batch. These results align with prior research on the implicit bias of GNNs trained with gradient-based methods, which demonstrated a tendency to overfit to graph structures even in scenarios where such structures should be disregarded (Bechler-Speicher et al., 2024a). Consequently, although GNNs are theoretically capable of converging to ID-invariant solutions, they fail to do so. This lack of invariance leads to overfitting. Thus, we set out to improve the ID-invariance of learned GNNs.

There are several ways of constraining models to be invariant to IDs. For example, one can constrain the output of the first GNN layer to have the same output for all assignments of node IDs. However,

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

we prove that this severely limits the expressive power of the GNN. Consequently, we conclude that improving expressiveness requires including layers that are not invariant to IDs. Additionally, we demonstrate that a three-layer architecture, with only the final layer being invariant to IDs, suffices to achieve invariance to IDs and expressiveness beyond that of the 1-WL test.

Building on our theoretical findings, we introduce ICON: ID-Invariance through Contrast, an approach that achieves ID invariance through explicit regularization. We evaluate ICON on both real-world and synthetic datasets across various GNN architectures, demonstrating that ICON significantly enhances ID invariance and generalization in many cases. Furthermore, we show that ICON improves extrapolation capabilities and accelerates training convergence.

Main Contributions: The goal of this work is to improve the utilization of node IDs in GNNs by making models ID-invariant. Our key contributions are:

1. We show, on multiple real-world datasets, that GNNs with RNI fail to learn invariance to IDs, despite resampling. This is an undesirable property, potentially leading to overfitting.
2. We theoretically analyze the properties of GNNs with IDs and derive concrete and practical requirements to achieve invariance to IDs while improving expressiveness with respect to GNNs without IDs.
3. Based on our theoretical analysis, we present ICON- an efficient method, that is compatible with any GNN, to achieve invariance to IDs while improving expressiveness.
4. We evaluate ICON on real-world and synthetic datasets with a variety of GNNs, and show that it significantly improves invariance to IDs and often improves generalization. Finally, we demonstrate how ICON can improve extrapolation capabilities and accelerate training convergence.

2 Related work

In this section, we review relevant prior work, focusing on the use of IDs in GNNs, and neural networks with invariance properties.

2.1 Graph neural networks

The key idea of GNNs (Kipf & Welling, 2017; Gilmer et al., 2017; Veličković et al., 2018; Hamilton et al., 2017; Bechler-Speicher et al., 2024b) is to use neural networks that combine node features with graph structure to obtain useful graph representations. This combination is done in an iterative manner, which can capture complex properties of the graph and its node features. See Wu et al. (2021) for a recent survey.

2.2 Node identifiers in GNNs

Many GNNs, such as MPGNNs, are inherently limited in their ability to distinguish between nodes with structurally similar neighborhoods. This limitation makes many GNNs only as powerful as the 1-Weisfeiler-Lehman (WL) isomorphism test (Morris et al., 2019).

To address this limitation, the assignment of node IDs has emerged as a practical technique to enhance the expressiveness of GNNs (Sato et al., 2021; Abboud et al., 2021; Bouritsas et al., 2023; Pellizzoni et al., 2024; Cantürk et al., 2024; Eliasof et al., 2023). The addition of node IDs can assist the GNN to distinguish between nodes that would otherwise be indistinguishable by message-passing schemes. For example, You et al. (2021) showed that expressiveness improves by adding binary node IDs. Recently, Pellizzoni et al. (2024) suggested a method for assigning IDs using Tinhofer-based node ordering (Tinhofer, 1991; Pellizzoni et al., 2024). Another approach to add node IDs is to calculate new structural node features, such as counting triangles (Bouritsas et al., 2023) or computing Laplacian eigenvalues (Dwivedi & Bresson, 2021), which can sometimes separate nodes that were otherwise indistinguishable. These approaches are deterministic, yet do not guarantee the uniqueness of node representation. To achieve uniqueness almost surely, Sato et al. (2021) and Abboud et al. (2021) suggested sampling random node features with sufficient dimension at every

epoch. This approach was shown to make MPGNNs universal approximators of invariant and equivariant graph functions (Abboud et al., 2021). In this work we focus on these non-deterministic approaches, and claim that the model should be invariant to the values of IDs.

2.3 INVARIANT NEURAL NETWORKS

The topic of invariance in deep learning has been explored in various domains, including Convolutional Neural Networks (CNNs) and GNNs (Simard et al., 1998). In CNNs, translational invariance is achieved by design. However, models such as Spatial Transformer Networks (STN) enable CNNs to handle more complex geometric transformations, such as invariances to rotations and zooming (Jaderberg et al., 2015). Furthermore, the Augerino approach, which learns invariances by optimizing over a distribution of data augmentations, is an effective strategy to learn invariances in tasks such as image classification (Benton et al., 2020). TI-Pooling, introduced in transformation-invariant CNN architectures, further addresses the need for invariance in vision-based tasks (Laptev et al., 2016). In graph tasks, learning invariances has seen rapid development, particularly with GNNs. Traditionally, GNNs are designed to be permutation invariant, but recent research has also explored explicit mechanisms to learn invariances rather than relying solely on built-in properties. For example, Xia et al. (2023) introduced a mechanism that explicitly learns invariant representations by transferring node representations across clusters, ensuring generalization under structure shifts in the test data. This method is particularly effective in scenarios where training distributions and test distributions may differ, a problem known as out-of-distribution generalization. Jia et al. (2024) proposed a strategy to extract invariant patterns by mixing environment-related and invariant subgraphs. In this work, we present ICON which promotes ID invariance while maintaining their added expressive power.

3 GNNS OVERFIT IDS

In this section, we demonstrate that GNNs trained with RNIs fail to converge to solutions that are invariant to the values of the IDs. We hypothesize that this lack of invariance may contribute to the poor empirical performance of RNIs in practice, and show in Section 6 that generalization improves when improving invariance to IDs.

Experimental setting We evaluate GraphConv (Morris et al., 2019), GIN (Xu et al., 2019), and GAT (Veličković et al., 2018) with RNIs, on four Open Graph Benchmark (OGB) datasets (Hu et al., 2020), which cover node and graph classification tasks. For each dataset and model, we sampled 64 random features as IDs. We compute the average *invariance ratio* with respect to the training set S, defined as follows. The invariance ratio of a model on an example x is $\max_{i \in L} P[f(x) = i]$ where P is the distribution over random ID assignments, and L is the set of possible model outputs (e.g., classes). This measures the degree to which all random ID assignments result in the same label. The maximum value of the invariance ratio is 1 (i.e., same output label for all random ID assignments) and the minimal value is $1/|L|$. For link prediction tasks, the invariance ratio is computed over the predicted binary edge labels (link or no-link), analogously to comparing predicted class labels in classification. We define the invariance ratio of S as the average of the invariance ratios for all $x \in S$. To estimate this ratio, we resampled IDs 10,000 times per sample after each epoch, on a random subset of up to 1,000 samples per dataset. All evaluations were performed in inference mode without gradient computation.

Datasets For node classification, we use ogbn-arxiv (Hu et al., 2020), a citation network where nodes represent academic papers and edges denote citation relationships. Node features are derived from word embeddings of paper titles and abstracts, and the task involves predicting the subject area of each paper. For graph classification, we used three molecular property prediction datasets (Hu et al., 2020). In ogbg-molhiv the task was to predict whether a molecule inhibits HIV replication, a binary classification task based on molecular graphs with atom-level features and bond-level edge features. ogbg-molbbbp involves predicting blood-brain barrier permeability, a crucial property for drug development, while ogbg-molbace focuses on predicting the ability of a molecule to bind to the BACE1 enzyme, associated with Alzheimer's disease. See more details in the Appendix.

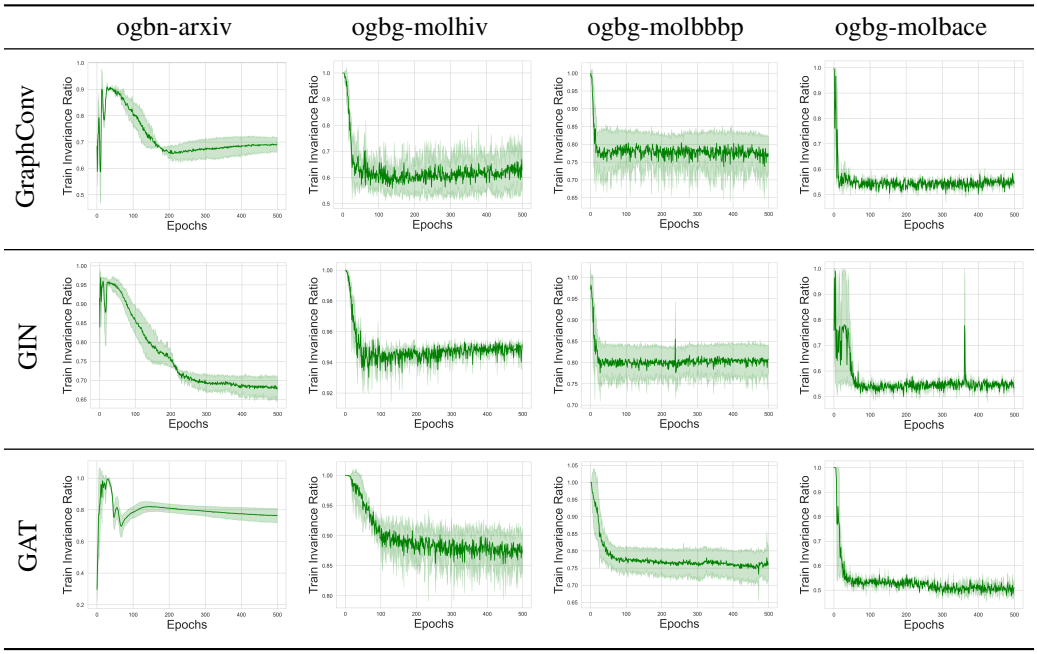


Figure 1: Invariance ratios over different GNNs with RNI, on real-world datasets. Despite the ID resampling that is applied in every epoch, invariance is not achieved and, in most cases, does not increase during training.

Evaluation protocol We used the training-validation-test splits introduced in Hu et al. (2020). Models were trained on the training set and hyperparameters were tuned by evaluating performance on the validation set. For testing, each model was trained with the best hyperparameters using three random seeds, and we report the average performance with these three seeds. Each network was trained for 500 epochs. Detailed hyperparameters for each model and dataset are provided in the Appendix.

Results Figure 1 presents the invariance ratios for the different GNNs and datasets. The results indicate that the models did not achieve invariance with respect to the IDs. Notably, most invariance ratios continue to decrease rather than increase throughout the training process. This suggests that the GNNs not only fail to converge toward invariance but also tend to overfit as training progresses.

We observe that the invariance ratios are relatively high at the beginning of training. It is important to note that a model can achieve invariance simply by ignoring the IDs (i.e., not using them at any part of the computation). As training progresses, the invariance ratio drops, which suggests that the model in fact uses the IDs, but in a non-invariant manner. This suggests that reassignment of random IDs during training is insufficient to guarantee invariance. In the next section, we theoretically analyze GNNs with IDs and consider approaches to achieve invariance.

We repeat this experiment with additional dimension sizes of random features and large weight decay and observe the same trends. Due to space limitations, these results are deferred to the Appendix.

4 Theoretical analysis of invariance to IDs and expressivity

In this section, we establish theoretical results on GNNs with unique IDs. We examine how GNNs can be invariant to IDs while still leveraging their expressiveness potential. Our objective is to derive practical guidelines for introducing ID invariance into the model. Due to space constraints, all proofs are deferred to the Appendix.

Preliminaries Throughout this paper, we denote a graph by $G = (V, E)$ over n nodes and its corresponding node features by $X \in \mathbb{R}^{n \times d}$, where d is the input feature dimension. We assume that

a constant feature 1 is always attached to the nodes, which is a common practice (Xu et al., 2019; Morris et al., 2019). We consider message-passing GNNs Morris et al. (2019) defined as follows. The initial node representations $H^{(0)}$ of a graph G over n nodes are the input node features X. Then, each layer of the GNN updates node representations via:

$$H^{(l+1)} = \text{GNN}_{\theta^{(l)}}(H^{(l)}; G),$$

where $\text{GNN}_{\theta^{(l)}}$ is the l-th layer, and it depends on the input graph structure G and the latest node representations $H^{(l-1)}$. The layer is associated with a set of learnable weights $\theta^{(l)}$ at each layer $l \in \{0, 1, \ldots, L-1\}$. The final node representations $\{h_i^{(L)}\}_{i=1}^n$ obtained in the last layer can then be used for downstream tasks such as node or graph labeling. We denote the embedding function realized by the k'th layer of a GNN by GNN_k.

Our focus is on GNNs that use additional *node IDs*. Namely, each node also takes an additional r dimensional vector, which serves as its ID. Let the matrix $I \in \mathbb{R}^{n \times r}$ denote the set of node IDs for all nodes. Node IDs are simply concatenated to the input node features, so that $H^{(0)} \in \mathbb{R}^{n \times (d+r)}$. We denote such GNNs as *GNN-R*, and note that they take three inputs: a graph G, features X, and I, a matrix of node IDs. Their output is the output label.

As noted earlier, a key desirable property of such models is that their output is invariant to the IDs, as formally defined below.

Definition 4.1 (ID-invariant function)**.** Let f be a GNN-R model, and G be a graph. Then f is *ID-invariant* with respect to G if for all X and unique node IDs I, I', $f(G, X, I) = f(G, X, I')$. In words, the output of f on G with any node features X does not depend on the values of the node IDs I. Similarly, f is invariant with respect to a set of graphs S, if the definition above holds for all $G \in S$.

We shall be specifically interested in the case where layers of a GNN are invariant to the IDs. Thus, we say that the k'th layer of a GNN is ID invariant if $GNN_k(G; X; I)$ is ID invariant. Namely, the layer is invariant if the embedding it produces does not change when I changes.

Note that a GNN-R that is not ID-invariant is prone to error, because there are certain inputs X for which two different IDs I, I' will result in different labels, and one of these will necessarily be wrong. Therefore, ID-invariant solutions are preferred. Note also that a function may be ID-invariant with respect to some graphs, but non-ID-invariant with respect to others, as the following theorem shows:

Theorem 4.2. *A function f can be ID-invariant with respect to a set of graphs S and non-ID-invariant with respect to another set of graphs S'.*

To prove Theorem 4.2 we simply construct a function where ID-invariance is constrained on a set of graphs. A direct implication of Theorem 4.2 is that it is possible for a GNN-R to be ID-invariant on a training set, but non-invariant on a test set, leading to potentially high test error.

Since achieving invariance is desirable, we next ask how this can be accomplished. A natural idea is to enforce invariance at every layer. However, Theorem 4.3 below shows that doing so actually eliminates the benefits of using IDs, reducing the model's expressiveness to that of a standard GNN without IDs. This is somewhat counterintuitive. It suggests that for expressiveness to be improved in the presence of unique IDs, at least some of the layers of the GNN must, in fact, be sensitive to the values of the IDs.

Theorem 4.3. *Let $\mathbb{G}$ be a GNN without unique node IDs. Let $\mathbb{G}'$ be a GNN-R with the same parameterization as $\mathbb{G}$, that is also ID-invariant in every layer. Then the sets of functions realized by $\mathbb{G}$ and $\mathbb{G}'$ are the same.*

To prove Theorem 4.3 we take a GNN-R and a GNN without IDs with the same parameterization and show by induction that the output of every layer of the GNN-R can be realized by the corresponding layers in the GNN without IDs (the reverse containment is clear).

Theorem 4.3 implies that in order to design a network that is both ID-invariant and expressive, we must allow it to not be invariant to IDs in at least one hidden layer. The next theorem shows that for three-layer models, invariance in the third layer results in enhanced expressiveness over a model that does not use IDs.

Table 1: Invariance ratios for train and test sets when training GNNs with RNI vs. ICON. Across all models and datasets, ICON consistently achieves high invariance ratios, sometimes reaching an invariance ratio of 1. In contrast, RNI often yields significantly lower invariance ratios, in some cases approaching 0.5 which is the lowest possible value for binary classification tasks.

		GraphConv		GIN		GAT		TransformerConv		GraphGPS	
DATASET	SPLIT	RNI	ICON	RNI	ICON	RNI	ICON	RNI	ICON	RNI	ICON
ARXIV	Train	$0.69_{\pm 0.02}$	$\mathbf{0.79}_{\pm 0.05}$	$0.68_{\pm 0.03}$	$\mathbf{1.00}_{\pm 0.00}$	$0.76_{\pm 0.05}$	$\mathbf{0.90}_{\pm 0.01}$	$0.73_{\pm 0.04}$	$\mathbf{0.94}_{\pm 0.01}$	$0.78_{\pm 0.03}$	$\mathbf{0.98}_{\pm 0.00}$
	Test	$0.67_{\pm 0.05}$	$\mathbf{0.85}_{\pm 0.11}$	$0.63_{\pm 0.02}$	$\mathbf{1.00}_{\pm 0.00}$	$0.66_{\pm 0.03}$	$\mathbf{0.85}_{\pm 0.01}$	$0.70_{\pm 0.03}$	$\mathbf{0.91}_{\pm 0.02}$	$0.75_{\pm 0.03}$	$\mathbf{0.97}_{\pm 0.01}$
MOLHIV	Train	$0.64_{\pm 0.10}$	$\mathbf{0.95}_{\pm 0.00}$	$0.95_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.88_{\pm 0.04}$	$\mathbf{0.94}_{\pm 0.00}$	$0.79_{\pm 0.02}$	$\mathbf{0.97}_{\pm 0.00}$	$0.82_{\pm 0.02}$	$\mathbf{0.99}_{\pm 0.00}$
	Test	$0.62_{\pm 0.09}$	$\mathbf{0.98}_{\pm 0.00}$	$0.97_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.86_{\pm 0.03}$	$\mathbf{0.96}_{\pm 0.00}$	$0.76_{\pm 0.03}$	$\mathbf{0.98}_{\pm 0.00}$	$0.80_{\pm 0.02}$	$\mathbf{1.00}_{\pm 0.00}$
MOLBACE	Train	$0.76_{\pm 0.02}$	$\mathbf{0.82}_{\pm 0.01}$	$0.78_{\pm 0.00}$	$\mathbf{0.85}_{\pm 0.11}$	$0.74_{\pm 0.02}$	$\mathbf{0.80}_{\pm 0.01}$	$0.70_{\pm 0.03}$	$\mathbf{0.83}_{\pm 0.02}$	$0.74_{\pm 0.02}$	$\mathbf{0.88}_{\pm 0.02}$
	Test	$0.74_{\pm 0.08}$	$\mathbf{0.90}_{\pm 0.03}$	$0.82_{\pm 0.03}$	$\mathbf{0.85}_{\pm 0.14}$	$0.72_{\pm 0.04}$	$\mathbf{0.86}_{\pm 0.02}$	$0.68_{\pm 0.02}$	$\mathbf{0.85}_{\pm 0.01}$	$0.73_{\pm 0.03}$	$\mathbf{0.91}_{\pm 0.02}$
MOLBBBP	Train	$0.55_{\pm 0.00}$	$\mathbf{0.52}_{\pm 0.06}$	$0.55_{\pm 0.02}$	$\mathbf{0.93}_{\pm 0.07}$	$0.51_{\pm 0.03}$	$\mathbf{0.58}_{\pm 0.02}$	$0.50_{\pm 0.02}$	$\mathbf{0.66}_{\pm 0.04}$	$0.52_{\pm 0.01}$	$\mathbf{0.73}_{\pm 0.03}$
	Test	$0.54_{\pm 0.01}$	$\mathbf{0.67}_{\pm 0.08}$	$0.51_{\pm 0.04}$	$\mathbf{0.96}_{\pm 0.04}$	$0.56_{\pm 0.02}$	$\mathbf{0.60}_{\pm 0.07}$	$0.53_{\pm 0.01}$	$\mathbf{0.70}_{\pm 0.03}$	$0.55_{\pm 0.02}$	$\mathbf{0.76}_{\pm 0.01}$
COLLAB	Train	$0.61_{\pm 0.03}$	$\mathbf{0.89}_{\pm 0.02}$	$0.62_{\pm 0.03}$	$\mathbf{0.97}_{\pm 0.01}$	$0.63_{\pm 0.03}$	$\mathbf{0.90}_{\pm 0.02}$	$0.65_{\pm 0.03}$	$\mathbf{0.93}_{\pm 0.02}$	$0.66_{\pm 0.02}$	$\mathbf{0.97}_{\pm 0.01}$
	Test	$0.59_{\pm 0.04}$	$\mathbf{0.92}_{\pm 0.03}$	$0.60_{\pm 0.02}$	$\mathbf{0.98}_{\pm 0.01}$	$0.61_{\pm 0.02}$	$\mathbf{0.92}_{\pm 0.02}$	$0.62_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.02}$	$0.64_{\pm 0.03}$	$\mathbf{0.98}_{\pm 0.01}$
SQUIRREL	Train	$0.57_{\pm 0.02}$	$\mathbf{0.83}_{\pm 0.04}$	$0.58_{\pm 0.03}$	$\mathbf{0.91}_{\pm 0.03}$	$0.59_{\pm 0.02}$	$\mathbf{0.84}_{\pm 0.03}$	$0.60_{\pm 0.02}$	$\mathbf{0.87}_{\pm 0.02}$	$0.61_{\pm 0.02}$	$\mathbf{0.92}_{\pm 0.02}$
	Test	$0.56_{\pm 0.03}$	$\mathbf{0.87}_{\pm 0.02}$	$0.55_{\pm 0.04}$	$\mathbf{0.94}_{\pm 0.02}$	$0.58_{\pm 0.03}$	$\mathbf{0.86}_{\pm 0.03}$	$0.59_{\pm 0.02}$	$\mathbf{0.89}_{\pm 0.02}$	$0.60_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.02}$

Theorem 4.4. *There exists a function $f(G; X)$ over graphs that no message passing GNN without IDs can realize, but $f(G; X)$ can be realized by an ID-invariant GNN-R with 3 layers, where only the last layer is ID-invariant.*

To prove Theorem 4.4 we construct an ID-invariant solution for the task of detecting a triangle. We show that it can be implemented by passing IDs in the first two layers, and then only checking if two IDs are the same or not, which is invariant to their specific values.

Together, Theorems 4.3 and 4.4 answer not only whether to enforce ID-invariance, but where to enforce it: regularizing all layers toward invariance forfeits the expressive advantage of IDs, while regularizing only the final layer is sufficient for ID-invariance and compatible with expressiveness beyond 1-WL. This directly motivates the design of ICON in Section 5.

In the next section, we build upon the results discussed in this section to design an approach that enhances the utilization of IDs in GNNs by explicitly enforcing ID-invariance to the model.

5 Learning invariance to IDs

In this section, we build on our analysis in Section 4 and propose ICON, an approach to improve the ability of GNNs to use IDs. While regularizing toward invariance is a natural idea, the question remains how to regularize: Theorem 4.3 shows that requiring invariance in all layers eliminates the expressive benefit of IDs, whereas Theorem 4.4 shows that enforcing invariance solely at the final layer is sufficient for ID-invariance while preserving beyond-1-WL expressiveness. Based on these insights, we propose to explicitly enforce invariance at the last layer only, using a regularizer. By avoiding invariance constraints on intermediate layers, ICON maintains the model's capacity to leverage IDs effectively.

The key idea in ICON is to add a term that regularizes models towards invariance to the random ID values. Before describing the regularizer term, we recall the "standard" supervised task-loss term.

Given a downstream task, such as graph classification or regression, we compute the task loss $\mathcal{L}_{\text{task}}$ based on the output of the final GNN layer $H^{(L)}$. Specifically, we use a standard linear classifier $g : \mathbb{R}^d \to \mathbb{R}^{d_{\text{out}}}$, where d_{out} is the desired dimension of the output (e.g., number of classes), to obtain the final prediction $\hat{y} = g(H^{(L)})$. This output is fed into a standard supervised loss (e.g., mean-squared error or cross-entropy). Formally, the task loss reads:

$$\mathcal{L}_{\text{task}} = \text{Loss}(\hat{y}, y),$$

where y is the ground truth label.

Table 2: Performance across OGB datasets for different models. We report mean ROC-AUC and std for the molhiv, molbace and molbbbp datasets, mean accuracy and std for the arxiv and Squirrel dataset, and Hits@50 and std for collab. For each model we report the performance with no IDs, with RNI, and with ICON. The best-performing approach within the std margin is highlighted in bold. In all cases, ICON outperforms RNI.

MODEL	ARXIV	MOLHIV	MOLBACE	MOLBBBP	COLLAB	SQUIRREL
GraphConv	$72.05_{\pm 1.13}$	$\mathbf{70.27}_{\pm 2.51}$	$62.58_{\pm 1.98}$	$63.39_{\pm 0.17}$	$46.15_{\pm 1.01}$	$32.4_{\pm 1.7}$
GraphConv+RNI	$71.34_{\pm 1.50}$	$56.44_{\pm 2.29}$	$52.29_{\pm 2.01}$	$59.08_{\pm 2.03}$	$40.09_{\pm 3.49}$	$28.9_{\pm 2.2}$
GraphConv+ICON	$\mathbf{73.66}_{\pm 1.19}$	$68.59_{\pm 1.96}$	$\mathbf{65.72}_{\pm 1.10}$	$\mathbf{67.02}_{\pm 0.97}$	$\mathbf{48.15}_{\pm 1.71}$	$\mathbf{34.8}_{\pm 1.3}$
GIN	$76.30_{\pm 1.58}$	$\mathbf{72.09}_{\pm 1.01}$	$73.02_{\pm 1.27}$	$\mathbf{60.86}_{\pm 1.05}$	$43.87_{\pm 0.94}$	$30.2_{\pm 1.5}$
GIN+RNI	$71.25_{\pm 1.91}$	$71.06_{\pm 1.95}$	$61.94_{\pm 2.51}$	$53.35_{\pm 1.73}$	$39.42_{\pm 2.31}$	$26.7_{\pm 1.9}$
GIN+ICON	$\mathbf{79.16}_{\pm 0.82}$	$\mathbf{72.01}_{\pm 1.30}$	$\mathbf{76.78}_{\pm 0.99}$	$\mathbf{59.96}_{\pm 2.00}$	$\mathbf{45.76}_{\pm 1.12}$	$\mathbf{33.1}_{\pm 1.1}$
GAT	$\mathbf{72.76}_{\pm 1.30}$	$72.60_{\pm 2.49}$	$\mathbf{71.01}_{\pm 2.83}$	$\mathbf{59.42}_{\pm 2.52}$	$44.23_{\pm 1.23}$	$29.8_{\pm 2.0}$
GAT+RNI	$61.85_{\pm 3.11}$	$54.52_{\pm 2.10}$	$63.43_{\pm 1.39}$	$58.75_{\pm 1.98}$	$40.87_{\pm 1.98}$	$27.3_{\pm 1.6}$
GAT+ICON	$\mathbf{71.77}_{\pm 2.96}$	$\mathbf{76.60}_{\pm 1.18}$	$66.08_{\pm 1.35}$	$\mathbf{61.93}_{\pm 1.29}$	$\mathbf{45.02}_{\pm 1.35}$	$\mathbf{32.7}_{\pm 1.2}$
TransformerConv	$73.15_{\pm 1.39}$	$\mathbf{70.11}_{\pm 1.49}$	$74.91_{\pm 2.04}$	$\mathbf{60.92}_{\pm 1.37}$	$44.67_{\pm 1.10}$	$33.6_{\pm 1.4}$
TransformerConv+RNI	$70.62_{\pm 1.60}$	$67.92_{\pm 1.40}$	$65.33_{\pm 1.98}$	$55.38_{\pm 1.73}$	$42.21_{\pm 1.92}$	$30.9_{\pm 1.6}$
TransformerConv+ICON	$\mathbf{76.09}_{\pm 2.10}$	$\mathbf{73.25}_{\pm 1.10}$	$\mathbf{75.01}_{\pm 1.50}$	$\mathbf{59.18}_{\pm 1.82}$	$\mathbf{46.39}_{\pm 1.18}$	$\mathbf{35.3}_{\pm 1.3}$
GraphGPS	$74.80_{\pm 1.99}$	$\mathbf{72.75}_{\pm 2.11}$	$69.94_{\pm 1.51}$	$\mathbf{58.17}_{\pm 2.01}$	$47.12_{\pm 0.88}$	$\mathbf{37.4}_{\pm 1.2}$
GraphGPS+RNI	$71.03_{\pm 2.08}$	$61.15_{\pm 3.79}$	$68.18_{\pm 1.40}$	$56.93_{\pm 2.40}$	$44.90_{\pm 1.30}$	$33.7_{\pm 1.7}$
GraphGPS+ICON	$\mathbf{77.25}_{\pm 2.15}$	$\mathbf{74.19}_{\pm 1.72}$	$\mathbf{72.01}_{\pm 1.49}$	$\mathbf{60.35}_{\pm 1.72}$	$\mathbf{48.26}_{\pm 0.91}$	$36.6_{\pm 1.0}$

To achieve invariance to the IDs, ICON utilizes a regularizer $\mathcal{L}_{\text{reg}}$ that is computed as follows. Assume our training sample is $G_1 = (V, E, X)$ with node identifiers R_1. We sample additional random node IDs R_2. Our goal is to make the network output similar for these two random IDs. Namely, the model should be invariant to the specific values of the node IDs. Let $H_i^{(L)}$ denote the final embeddings for the model with node IDs R_i. Namely $H_i^{(L)}$ is obtained by propagating the inputs with IDs R_i through the network up to the last GNN layer (prior to the final prediction layer). To impose ID-invariance, we consider a loss that aims to reduce the difference between these embeddings. There are several options and metrics possible here, and we choose the mean squared error for simplicity. Namely, we consider the regularizer:

$$\mathcal{L}_{\text{reg}} = \frac{1}{n \cdot d_L} \|H_1^{(L)} - H_2^{(L)}\|_2^2,$$

where n is the number of nodes and d_L is the embedding dimension, so that $\mathcal{L}_{\text{reg}}$ is normalized and does not scale with the graph size.

Then, the overall loss per training example in ICON is: $\mathcal{L} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{reg}}$.
The task loss $\mathcal{L}_{\text{task}}$ is computed using the predictions from the first ID assignment R_1 only, while $\mathcal{L}_{\text{reg}}$ compares the embeddings from both assignments. The loss is minimized via standard stochastic gradient descent, where this is the per-sample loss. We note that no additional weighting coefficient is needed between the two terms, as the normalization in $\mathcal{L}_{\text{reg}}$ ensures that both terms are on a comparable scale. Note that it is possible to sample more than two random IDs per training example and adapt the loss accordingly. Here we opt for the simplest version where the sample regularizer only uses two samples.

6 Experimental evaluation

In this section, we evaluate ICON on diverse real-world and synthetic datasets.[1] We demonstrate its ability to consistently improve invariance to IDs and, as a result, to often enhance generalization.

6.1 Real-world datasets

To assess the effectiveness of ICON on real datasets, we evaluate it on diverse tasks from the Open Graph Benchmark (OGB) (Hu et al., 2020). We test ICON across multiple GNN architectures, including GraphConv (Morris et al., 2019), GIN (Xu et al., 2019), GAT (Veličković et al., 2018),

[1]Code is provided in Supplementary Material

TransformerConv (Shi et al., 2021) and GraphGPS (Rampášek et al., 2022). We note that ICON is simple to add to any existing method. We focus on these architectures due to their widespread use and to demonstrate the efficacy of ICON across GNN architectures.

Datasets As node-level tasks we used ogbn-arxiv and the heterophilous Squirrel dataset for node classification. We used ogbg-molhiv, ogbg-molbbbp, and ogbg-molbace for binary graph classification, and ogbl-collab for link prediction (Hu et al., 2020; Rozemberczki et al., 2021). The datasets are described in detail with their dataset statistics in the Appendix.

Setup For each dataset and GNN, we evaluated three configurations: the baseline model without IDs, the model with RNI, and the model with ICON. For each configuration, we conducted a grid search by training on the training set and evaluating on the validation set. We selected the best-performing model over the validation set. We then report the average metric scores and std of the selected configuration with 3 seeds. We use ROC-AUC for binary graph classification, Accuracy for node classification, and Hits@50 for link prediction, following the common metrics for these datasets in the literature. We also calculate the final invariance ratios as defined in Section 3 for both RNI and ICON. We use 64 random features as the IDs. We repeat this experiment with additional sizes of ID dimensions and observe the same trends. Due to space limitations, these results are deferred to the Appendix.

Results Table 1 presents the invariance ratios of ICON and RNI across all datasets and models, with respect to the train and test sets. Across all datasets and models, ICON significantly improves invariance with respect to both the train and the test set. In some cases, it reaches full invariance. Table 2 presents the average ROC-AUC scores for each model and dataset. Across all models and datasets, ICON improves upon RNI, with the maximal margin being 22.08 ROC-AUC points on the ogbg-molhiv datasets with GAT. In 23 out of the 30 cases, ICON improved generalization also upon the baseline model and was on par with the baseline in 4 other cases, which shows that improving invariance can improve generalization.

Table 3: Invariance ratios with respect to the train set and test set when training GNNs with RNI vs. with ICON over the isInTriangle task. ICON achieves perfect invariance.

METHOD	Train	Test
GraphConv+RNI	$0.95_{\pm 0.01}$	$0.93_{\pm 0.08}$
GraphConv+ICON	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$
GIN+RNI	$0.95_{\pm 0.02}$	$0.94_{\pm 0.02}$
GIN+ICON	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.99}_{\pm 0.00}$
GAT+RNI	$0.88_{\pm 0.09}$	$0.88_{\pm 0.13}$
GAT+ICON	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$

6.2 Synthetic datasets

We next examine the performance of ICON and RNI on synthetic datasets with three synthetic tasks where GNNs without IDs are provably unable to solve these tasks (Chen et al., 2020; Garg et al., 2020; Abboud et al., 2021) but GNNs that employ unique IDs and are sufficiently large can solve these tasks (Abboud et al., 2021).

6.2.1 The "Is In Triangle" task

Dataset The isInTriangle task is a binary node classification where the goal is to determine whether a given node is part of a triangle. It was shown in Chen et al. (2020); Garg et al. (2020) that MPGNNs without IDs cannot solve the isInTriangle task. With IDs, GNNs can solve the task, yet they can do so by either overfitting to the IDs or with an IDs-invariant solution. We show the existence of an IDs-invariant solution for this task in the proof of Theorem 4.4. The dataset consists of 100 graphs with 100 nodes each, generated using the preferential attachment (BA) model (Barabasi & Albert, 1999), in which graphs are constructed by incrementally adding new nodes with m edges and connecting them to existing nodes with a probability proportional to the degrees of those nodes. We adopt an inductive setting, where the graphs used during testing are not included in the training set. We used $m = 2$ for the training graphs and evaluated

Table 4: Accuracy on the isInTriangle task in interpolation and extrapolation settings.

Method	Interp.	Extrap.
GraphConv+Const.	$75.35_{\pm 2.09}$	$53.70_{\pm 1.67}$
GraphConv+RNI	$74.87_{\pm 3.06}$	$57.02_{\pm 3.39}$
GraphConv+ICON	$\mathbf{88.45}_{\pm 2.04}$	$\mathbf{78.20}_{\pm 2.53}$
GIN+Const.	$76.10_{\pm 1.83}$	$56.29_{\pm 1.55}$
GIN+RNI	$72.15_{\pm 2.01}$	$55.01_{\pm 1.30}$
GIN+ICON	$\mathbf{90.95}_{\pm 1.27}$	$\mathbf{71.20}_{\pm 1.94}$
GAT+Const.	$74.99_{\pm 2.10}$	$56.07_{\pm 2.11}$
GAT+RNI	$71.20_{\pm 1.48}$	$58.92_{\pm 1.57}$
GAT+ICON	$\mathbf{89.71}_{\pm 1.27}$	$79.30_{\pm 1.90}$

two test sets: an interpolation setting in which the graphs are drawn from the BA distribution with $m = 2$ and an extrapolation setting in which the graphs are drawn from a different distribution than the train graphs, a BA distribution with $m = 3$.

Setup We repeat the setup from Sections 3 and 6.1. As in this task nodes have no features, for the baseline models where IDs are not used, we use a constant 1 feature for all nodes. For each task, we conducted a grid search by training on the training set and evaluating on the validation set. We selected the best-performing model over the validation set. We then report the average accuracy and std of the selected configuration with 3 seeds. We also calculate the final invariance ratios as in Sections 3 and 6.1 for RNI and ICON.

Results Table 3 presents the invariance ratios of ICON and RNI across all models, for the interpolation and extrapolation settings. Across all models, ICON substantially improves invariance both with respect to the train and the test set. In some cases, it reaches full invariance. Table 4 presents the average accuracy for each model for the interpolation and extrapolation setup. Across all models ICON improves upon RNI and the baseline, with the maximal margin being 18.80 accuracy percentage in the interpolation setting and 21.18 accuracy percentage in the extrapolation setting.

6.2.2 EXP AND CEXP

Abboud et al. (2021) presented the EXP and CEXP datasets as an example of the improved expressiveness RNI provides, as these tasks cannot be solved with MPGNNs without IDs. However, it was shown that to improve upon the MPGNNs accuracy, much longer training time is required. In the current experiment, we set out to test if ICON can achieve the same accuracy of RNI while improving the training time. The datasets contain 600 pairs of graphs 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. CEXP is a modified version of EXP, where 50% of pairs are slightly modified to be distinguishable by 1-WL. It was shown in Abboud et al. (2021) that a GraphConv (Abboud et al., 2021) with RNI reaches perfect accuracy on the test set. We repeat the protocol from Abboud et al. (2021), who evaluated RNI. We use a GraphConv GNN (Morris et al., 2019) with RNI and ICON with 8 layers, 64 random features, 64 hidden dimensions, over 500 epochs, with 3 random seeds. Figure 2 presents the learning curves of accuracy during training for ICON and RNI. Both methods reach a test accuracy of almost 100%, but ICON convergence is faster in both tasks, highlighting the effect of explicit regularization towards IDs-invariance as introduced by ICON vs. an implicit one as done in RNI using random resampling of IDs during training.

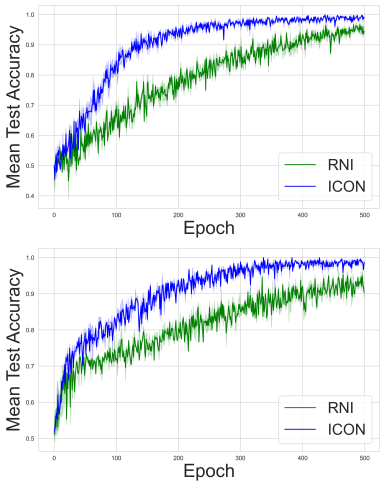


Figure 2: The test accuracy learning curves of RNI and ICON on the EXP and CEXP datasets.

7 CONCLUSIONS

In this paper, we harness the theoretical power of unique node identifiers (IDs) to enhance the effectiveness of Graph Neural Networks (GNNs). We show that, in practice, the final output of GNNs that incorporate node IDs is not invariant to the specific choice of IDs. We provide a theoretical analysis that motivates an approach that explicitly enforces invariance to IDs in GNNs while taking advantage of their expressive power. Building on our theoretical results, we propose ICON, an explicit regularization technique, which is GNN agnostic. Through comprehensive experiments, we demonstrate across real-world and synthetic datasets that ICON considerably improves invariance to node IDs, as well as generalization, extrapolation, and training convergence time. These results highlight ICON as an efficient solution to improve the performance of GNNs through the utilization of IDs, making it a valuable tool for use with any GNN. We discuss directions for future work in Appendix E.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure that our results are reproducible. The proposed regularizer is fully specified in Section 5, and complete proofs of all theoretical claims, together with their assumptions, are given in the Appendix. The datasets, splits, architectures, hyperparameters, and training protocols used in every experiment are described in Section 6 and detailed further in the Appendix. Source code to reproduce our experiments is provided as anonymous supplementary material.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the tasks for which ICLR requires disclosure. Specifically, generative AI was not used to create synthetic datasets or to clean or reformat data, to build theoretical models or conceptual frameworks, to state mathematical claims, supply key proof ingredients, or write proofs, to propose or refine hypotheses, to design the experiments or methodology or to give feedback on them, to implement the method proposed in this paper, or to interpret results. Translation and qualitative or thematic analysis are not applicable to this work. Among the tasks for which disclosure is recommended, we used generative AI tools for two purposes: to improve readability through editing, namely grammar and phrasing improvements to text written by the authors; and to write and edit supporting software code, such as experiment scripts, data processing utilities, and plotting and analysis code. We have reviewed all AI-assisted work. Every AI-edited sentence was checked by the authors against the intended meaning, and all AI-assisted code was read, tested, and verified for correctness by the authors before any result reported here was produced with it. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In Zhi-Hua Zhou (ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, pp. 2112–2118. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/291. URL https://doi.org/10.24963/ijcai.2021/291. Main Track.

Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. Science, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509. URL http://www.sciencemag.org/cgi/content/abstract/286/5439/509.

Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. arXiv, 2018. URL https://arxiv.org/pdf/1806.01261.pdf.

Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2024a.

Maya Bechler-Speicher, Amir Globerson, and Ran Gilad-Bachrach. The intelligible and effective graph neural additive networks. In Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24, Red Hook, NY, USA, 2024b. Curran Associates Inc. ISBN 9798331314385.

Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks. In Proceedings of NeurIPS, 2020. URL https://arxiv.org/abs/2010.11882.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation

networks. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=dFbKQaRk15w.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1):657–668, January 2023. ISSN 1939-3539. doi: 10.1109/tpami.2022.3154319. URL http://dx.doi.org/10.1109/TPAMI.2022.3154319.

Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), Proceedings of the 41st International Conference on Machine Learning, volume 235 of Proceedings of Machine Learning Research, pp. 5533–5566. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/canturk24a.html.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 10383–10395. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/75877cb75154206c4e65e76b88a12712-Paper.pdf.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. URL https://arxiv.org/abs/2012.09699.

Moshe Eliasof, Fabrizio Frasca, Beatrice Bevilacqua, Eran Treister, Gal Chechik, and Haggai Maron. Graph positional encoding via random feature propagation. In International Conference on Machine Learning, pp. 9202–9223. PMLR, 2023.

Moshe Eliasof, Beatrice Bevilacqua, Carola-Bibiane Schönlieb, and Haggai Maron. Granola: Adaptive normalization for graph neural networks. In Proceedings of NeurIPS, 2024.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In Hal Daumé III and Aarti Singh (eds.), Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pp. 3419–3430. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/garg20c.html.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In Advances in neural information processing systems, pp. 2017–2025, 2015.

Tianrui Jia, Haoyang Li, Cheng Yang, Tao Tao, and Chuan Shi. Graph invariant learning with subgraph co-mixup for out-of-distribution generalization. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 8562–8570, 2024.

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

Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 289–297, 2016.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=B1l2bp4YwS.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pp. 4602–4609, 2019. doi: 10.1609/aaai.v33i01.33014602. URL https://ojs.aaai.org/index.php/AAAI/article/view/4384.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 4663–4673. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/murphy19a.html.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), Advances in Neural Information Processing Systems, volume 34, pp. 21997–22009. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b8b2926bd27d4307569ad119b6025f94-Paper.pdf.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Paolo Pellizzoni, Till Hendrik Schulz, Dexiong Chen, and Karsten Borgwardt. On the expressivity and sample complexity of node-individualized graph neural networks. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=8APPypS0yN.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), Advances in Neural Information Processing Systems, volume 35, pp. 14501–14515. Curran Associates, Inc., 2022. doi: 10.52202/068431-1054. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/5d4834a159f1547b267a05a4e2b7cf5e-Paper-Conference.pdf.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. Journal of Complex Networks, 9(2):cnab014, 04 2021. ISSN 2051-1329. doi: 10.1093/comnet/cnab014. URL https://doi.org/10.1093/comnet/cnab014.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In Proceedings of the 2021 SIAM International Conference on Data Mining, SDM, 2021.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou (ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, pp. 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/214. URL https://doi.org/10.24963/ijcai.2021/214. Main Track.

Patrice Y. Simard, Yann A. LeCun, John S. Denker, and Bernard Victorri. Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation, pp. 239–274. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49430-0. doi: 10.1007/3-540-49430-8_13. URL https://doi.org/10.1007/3-540-49430-8_13.

G. Tinhofer. A note on compact graphs. Discrete Applied Mathematics, 30(2):253–264, 1991. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(91)90049-3. URL https://www.sciencedirect.com/science/article/pii/0166218X91900493.

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

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Donglin Xia, Xiao Wang, Nian Liu, and Chuan Shi. Learning invariant representations of graph neural networks via cluster generalization. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=zrCmeqV3Sz.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Jiaxuan You, Jonathan M. Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 10737–10745, 2021.

A Proofs

A.1 Proof of Theorem 4.2

Let S and S' be two disjoint sets of graphs ($S \cap S' = \varnothing$). Fix any ID-invariant graph property $P : \mathcal{G} \to \{0, 1\}$ (e.g., "G has an Eulerian path"). For an assignment I that gives each node $v \in V(G)$ a unique ID vector $I_v \in \mathbb{R}^r$, define

$$f(G, I) := \begin{cases} P(G), & \text{if } G \in S, \\ \sum_{v \in V(G)} \mathbf{1}^\top I_v, & \text{if } G \in S', \end{cases}$$

where $\mathbf{1} \in \mathbb{R}^r$ is the all-ones vector.

ID-invariance on S: If $G \in S$, then $f(G, I) = P(G)$ depends only on G and is independent of I. Hence f is ID-invariant on S.

Non-invariance on S': Suppose $G \in S'$ with $V(G) \neq \varnothing$ Pick any node $u \in V(G)$ and any two ID assignments I, I' that are identical on $V(G) \setminus \{u\}$ but satisfy $\mathbf{1}^\top I_u \neq \mathbf{1}^\top I'_u$ (e.g., change one coordinate of I_u). Then

$$f(G, I) - f(G, I') = \sum_{v \in V(G)} \mathbf{1}^\top I_v - \sum_{v \in V(G)} \mathbf{1}^\top I'_v = \mathbf{1}^\top I_u - \mathbf{1}^\top I'_u \neq 0,$$

so $f(G, I) \neq f(G, I')$. Hence f is not ID-invariant on S'.

Therefore a single function f is ID-invariant on S and not ID-invariant on S'.

A.2 Proof of direct implication of Theorem 4.2

In the main text we mention that a direct implication of Theorem 4.2 states that a GNN can be ID-invariant to a train set and non ID-invariant to a test set. This setting only differs from the setting

in Theorem 4.2 in that the learned function is now restricted to one that can be realized by a GNN. Indeed, as proven in Abboud et al. (2021), a GNN with IDs is a universal approximator. Specifically, with enough layers and width, an MPGNN can reconstruct the adjacency matrix in some hidden layer, and then compute over it the same function as in the proof of Theorem 4.2.

A.3 Proof of Theorem 4.3

Let $\mathbb{G}$ be a message-passing GNN as in the preliminaries, without node IDs. Let $\mathbb{G}'$ be a GNN-R with node IDs $I \in \mathbb{R}^{n\times r}$ concatenated to the features, sharing the same architecture and parameterization as $\mathbb{G}$. Let g be a fixed graph.

We will show that if $\mathbb{G}'$ is ID-invariant in every layer, then $\mathbb{G}$ and $\mathbb{G}'$ can realize the same set of functions. Specifically, $\mathbb{G}'$ is not more expressive than $\mathbb{G}$. We prove this by induction on the number of layers.

We first consider GNNs that output embeddings for each node in the graph. The argument for GNNs performing graph-level prediction (e.g., via aggregation and readout over node embeddings) follows immediately and is discussed at the end of the proof. We assume, without loss of generality, that the hidden dimension h is fixed across all layers, including the final node embeddings.

Base step: one-layer GNN. Let $\mathbb{G}'$ be a one-layer GNN-R and $\mathbb{G}$ a one-layer GNN with the same architecture and parameterization as $\mathbb{G}'$. The inputs to the first layer are $(A, [X\ I])$ in $\mathbb{G}'$ and (A, X) in $\mathbb{G}$. Recall that both models include a constant feature 1 as part of X (see Section 4).

$\mathbb{G}$ can realize anything $\mathbb{G}'$ can. Assume $\mathbb{G}'$ is ID-invariant and implements an ID-invariant function f. In a message-passing layer, the update for each node v takes the form

$$h_v^{(1)} = u\Big(h_v^{(0)}, \sum_{w\in N(v)} m(h_w^{(0)}, h_v^{(0)})\Big),$$

where m is the message function and u is the update function. Since each message $m(h_w^{(0)}, h_v^{(0)})$ is a function of a single neighbor-receiver pair, and the aggregation is a sum, it suffices to show that the pairwise message function does not depend on the ID components.

Since f is ID-invariant on the entire graph, it is also ID-invariant when restricted to any induced subgraph. In particular, consider any edge (w, v) and the induced subgraph on just these two nodes. On this subgraph, the message from w to v is:

$$m([X_w; I_w], [X_v; I_v]) = m([X_w; I'_w], [X_v; I'_v])$$

for all valid unique IDs I, I'. Since this holds for arbitrary ID values, the message function effectively depends only on (X_w, X_v), i.e., there exists a function $\tilde{m}$ such that $m([X_w; I_w], [X_v; I_v]) = \tilde{m}(X_w, X_v)$ for all unique IDs. The same argument applies to the update function u with respect to the self-features.

Since each individual message ignores the IDs, the full aggregation $\sum_{w\in N(v)} \tilde{m}(X_w, X_v)$ is a permutation-invariant function over the neighbor multiset that depends only on node features X, regardless of the neighborhood size. Therefore, $\mathbb{G}$ can realize the same function by setting its weights for the feature components equal to those of $\mathbb{G}'$.

$\mathbb{G}'$ can realize anything $\mathbb{G}$ can. Given any function realized by $\mathbb{G}$ operating on features X (which include the constant 1), $\mathbb{G}'$ receives the concatenated input $[X; I]$. Since both models share the same architecture and parameterization, $\mathbb{G}'$ can simply set the weights corresponding to the I channels to zero, reducing to the same computation as $\mathbb{G}$.

Inductive step. Assume the theorem holds for any GNN and GNN-R with up to $L-1$ layers. We now prove it for an L-layer GNN and GNN-R.

Let $\mathbb{G}'$ be an L-layer GNN-R and $\mathbb{G}$ an L-layer GNN with the same architecture and parameterization as $\mathbb{G}'$. Let f be a function implemented by $\mathbb{G}'$. We can decompose it as $f = f_L \circ f_{[L-1]}$, where $f_{[L-1]}$ is the function implemented by the first $L-1$ layers of $\mathbb{G}'$, and f_L is the function implemented by the L-th layer.

By the inductive hypothesis, $f_{[L-1]}$ can be realized by some $\mathbb{G}_{(L-1)}$ (a GNN without IDs) with the same architecture and parameterization up to layer $L-1$. We therefore set $\mathbb{G}_{(L-1)}$ to implement $f_{[L-1]}$. Then, since the architectures of $\mathbb{G}$ and $\mathbb{G}'$ are identical, $\mathbb{G}$ can realize the same L-th layer transformation f_L. Hence, we define $\mathbb{G}$ such that its first $L-1$ layers realize $f_{[L-1]}$ and its L-th layer realizes f_L.

Consequently,

$$\mathbb{G} = f_L \circ f_{[L-1]} = f,$$

and thus $\mathbb{G}$ can realize f.

The reverse direction is the same: let $\mathbb{G}$ be an L-layer GNN without IDs, and $\mathbb{G}'$ an L-layer GNN-R with the same architecture and parameterization. Assume that $\mathbb{G}$ realizes some function $f = f_L \circ f_{[L-1]}$, where $f_{[L-1]}$ corresponds to its first $L-1$ layers and f_L to its last layer.

By the inductive hypothesis, there exists a GNN-R $\mathbb{G}'_{(L-1)}$ (ID-invariant) whose first $L-1$ layers realize $f_{[L-1]}$. Since the architectures and parameters of $\mathbb{G}$ and $\mathbb{G}'$ are identical, we can define the final layer of $\mathbb{G}'$ to implement the same transformation f_L. Thus, the full $\mathbb{G}'$ realizes

$$\mathbb{G}' = f_L \circ f_{[L-1]} = f.$$

Therefore, any function realizable by $\mathbb{G}$ can also be realized by $\mathbb{G}'$, which completes the inductive step and establishes the equivalence for L layers. □

A.4 Proof of Theorem 4.4

It is known that isInTriangle is not realizable by MPGNNs without IDs (Garg et al., 2020) (i.e., not within 1-WL expressivity). We construct a 3-layer GNN-R f that realizes isInTriangle, where only the last layer is ID-invariant. Crucially, the construction works for any set of unique node IDs, not just a specific assignment.

Let $G = (V, E)$ with $|V| = n$. Each node $u \in V$ is assigned a unique scalar ID $I_u \in \mathbb{R}$ (no other features are used; the extension to $I_u \in \mathbb{R}^r$ is immediate by projecting to a scalar first). Write $N(u)$ for the neighbors of u and let $h_v^{(\ell)}$ denote the representation of node v at layer ℓ.

For the sake of clarity, we define the message-passing steps as commonly done in the literature (Gilmer et al., 2017; Battaglia et al., 2018; Xu et al., 2019; Morris et al., 2019). A layer ℓ is specified by a message function $m^{(\ell)}$, an aggregation operator AGG (we use summation), and an update function $u^{(\ell)}$:

$$\text{messages:} \quad M_{u\to v}^{(\ell)} \;=\; m^{(\ell)}\big(h_u^{(\ell-1)},\, h_v^{(\ell-1)}\big), \qquad \text{aggregate:} \quad a_v^{(\ell)} \;=\; \sum_{u\in N(v)} M_{u\to v}^{(\ell)},$$

$$\text{update:} \quad h_v^{(\ell)} \;=\; u^{(\ell)}\big(h_v^{(\ell-1)},\, a_v^{(\ell)}\big).$$

We now construct a three-layer GNN-R. The construction uses a collection of K nonlinear basis functions $\psi_1, \ldots, \psi_K : \mathbb{R} \to \mathbb{R}$ (e.g., random Fourier features or shifts of a ReLU), chosen so that for any set of n distinct scalars $\{I_1, \ldots, I_n\}$, the vectors $\Psi(I_u) \coloneqq (\psi_1(I_u), \ldots, \psi_K(I_u))^\top \in \mathbb{R}^K$ are linearly independent. Such a collection exists for $K \geq n$ by standard arguments (e.g., using analytic functions, the vectors are linearly independent for any set of distinct inputs). In particular, for distinct IDs, the inner product $\langle \Psi(I_u), \Psi(I_w) \rangle$ is large when $I_u = I_w$ and generically small otherwise, allowing the detection of shared neighbors.

We set $h_v^{(0)} \coloneqq \Psi(I_v) \in \mathbb{R}^K$.

Layer 1: neighbor signature vectors. Each node collects the nonlinear embeddings of its neighbors' IDs:

$$m^{(1)}(h_u^{(0)}, h_v^{(0)}) \;=\; h_u^{(0)} = \Psi(I_u), \qquad u^{(1)}(h_v^{(0)}, a_v^{(1)}) \;=\; a_v^{(1)}.$$

Thus

$$h_v^{(1)} \;=\; \sum_{u\in N(v)} \Psi(I_u) \;\in\; \mathbb{R}^K.$$

This is a "neighborhood signature" vector that encodes the multiset of neighbor IDs. Note that this layer is not ID-invariant: the representation $h_v^{(1)}$ depends on the specific ID values of v's neighbors.

Layer 2: common-neighbor detection. Define the message as the inner product of the two endpoints' Layer-1 signatures:

$$m^{(2)}(h_u^{(1)}, h_v^{(1)}) \;=\; \langle h_u^{(1)},\, h_v^{(1)} \rangle \;=\; \sum_{w \in N(u)} \sum_{w' \in N(v)} \langle \Psi(I_w),\, \Psi(I_{w'}) \rangle \;\in\; \mathbb{R},$$

and the update

$$u^{(2)}(h_v^{(1)}, a_v^{(2)}) \;=\; a_v^{(2)} \;=\; \sum_{u \in N(v)} \langle h_u^{(1)},\, h_v^{(1)} \rangle.$$

By the linear independence of $\{\Psi(I_w)\}$, the Gram matrix $G_{ww'} = \langle \Psi(I_w), \Psi(I_{w'}) \rangle$ is positive definite. In particular, the diagonal entries satisfy $G_{ww} = \|\Psi(I_w)\|^2 > 0$, while the off-diagonal terms $G_{ww'}$ for $w \neq w'$ are generically nonzero but can be made arbitrarily small relative to the diagonal by choosing the basis functions ψ_k appropriately (e.g., with sufficiently separated random frequencies).

Therefore, the dominant contribution to $h_v^{(2)}$ comes from terms where $w = w'$, i.e., common neighbors:

$$h_v^{(2)} \;=\; \sum_{u \in N(v)} \sum_{w \in N(u)} \sum_{w' \in N(v)} G_{ww'} \;\approx\; c \cdot \sum_{u \in N(v)} |N(u) \cap N(v)|,$$

where $c = \|\Psi(I_w)\|^2 > 0$ (which can be made constant across nodes by normalizing Ψ). With a suitable choice of basis functions, the gap between the dominant (common-neighbor) terms and the cross terms can be made arbitrarily large, so that $h_v^{(2)} > 0$ if and only if v participates in at least one triangle.

We emphasize that the value of $h_v^{(2)}$ depends on the specific IDs (through the Gram matrix), so this layer is not ID-invariant. However, the sign of $h_v^{(2)}$ — which is all that matters for the final decision — is determined solely by the graph topology.

Layer 3: ID-invariant decision. Let $g : \mathbb{R} \to \{0, 1\}$ be a function that realizes the threshold $g(t) = \mathbb{1}\{t > \tau\}$ for a threshold τ chosen between the cross-term noise level and the common-neighbor signal level (which is always possible by the gap argument above). This can be implemented by a ReLU MLP. Define a dummy message,

$$m^{(3)}(h_u^{(2)}, h_v^{(2)}) \;=\; 0, \qquad a_v^{(3)} \;=\; 0,$$

and the update

$$u^{(3)}(h_v^{(2)}, a_v^{(3)}) \;=\; g\big(h_v^{(2)}\big) \in \{0, 1\}.$$

Set the network output to $f(G)_v \coloneqq h_v^{(3)}$.

By the construction above,

$$f(G)_v \;=\; 1 \quad\Longleftrightarrow\quad h_v^{(2)} \;>\; \tau \quad\Longleftrightarrow\quad \exists\, u \in N(v) \text{ with } N(u) \cap N(v) \neq \varnothing,$$

which holds if and only if node v lies on a triangle. Thus f realizes IsInTriangle.

ID-invariance of the construction. The output $f(G)_v \in \{0, 1\}$ depends only on whether v participates in a triangle, which is a topological property independent of the ID values. Hence, for any two sets of unique IDs I and I', the network produces the same output: $f(G, I) = f(G, I')$. Layers 1 and 2 produce ID-dependent intermediate representations (their numerical values change when IDs are resampled), but the final binary output is invariant. Only Layer 3 is constrained to be ID-invariant; Layers 1 and 2 are not.

Remark on the hidden dimension. The construction uses a hidden dimension $K \geq n$ (the number of nodes). While this depends on the graph size, this is standard in expressivity proofs for GNNs

(cf. (Abboud et al., 2021)), which establish the existence of networks with sufficient capacity, not practical architectures.

B Additional Experimental Results

In this section, we provide complementary experiments to those presented in the main paper, aimed at verifying the robustness of the observed phenomena with respect to the number of random ID channels and to the regularization strength during training. The results consistently support the conclusions drawn in the main text regarding both the invariance and generalization benefits achieved by ICON.

Varying the number of random ID channels. Table 5 reports results obtained when using 8 random channels as node identifiers (IDs), compared to the default configuration of 64 channels used in the main experiments. We observe that the trends remain consistent across all models and datasets. Specifically, ICON continues to achieve substantially higher ID-invariance ratios than RNI and, in most cases, leads to improved generalization performance. While using fewer ID channels slightly reduces the overall representational capacity of the model, the relative improvement achieved by ICON remains stable, demonstrating its robustness to the dimensionality of the identifier space.

Invariance under reduced weight decay. To further assess the stability of ICON under different regularization regimes, we repeat all experiments with a reduced weight decay coefficient of 10^{-5} (compared to 10^{-4} in the main experiments). The corresponding invariance ratios are illustrated in Figure 4. Once again, the results align with the findings in the main paper: models trained with ICON exhibit markedly higher invariance ratios throughout training, even when the weight decay is reduced. This suggests that the observed invariance improvement is a direct result of the ICON regularization term rather than a by-product of stronger implicit regularization.

Invariance evolution during training. Figure 3 depicts the invariance trajectories over epochs for the case of 8-channel IDs. Similar to the 64-channel case, we observe that standard RNI-based training initially exhibits partial invariance, which gradually degrades as training progresses. In contrast, ICON maintains high invariance ratios across epochs, often approaching full invariance by convergence. This consistent pattern confirms that ICON successfully constrains the model to remain invariant to the values of random IDs during optimization, independent of architectural choices or ID dimensionality.

C Additional Experimental Details

Real-world datasets The datasets' statistics are summarized in Table 6. They include both node- and graph-level classification tasks, as well as a link prediction benchmark. For the node classification task, we use OGBN-ARXIV and the heterophilous SQUIRREL dataset. For graph-level prediction, we use the molecular property datasets OGBG-MOLHIV, OGBG-MOLBACE, and OGBG-MOLBBBP. Finally, OGBL-COLLAB is employed for link prediction. All OGB datasets follow the official OGB train/validation/test splits (Hu et al., 2020). For the SQUIRREL dataset (Rozemberczki et al., 2021), we use the standard train/validation/test splits provided by Pei et al. (2020).

isInTriangle The isInTriangle task is a binary node classification where the goal is to determine whether a given node is part of a triangle. The dataset consists of 100 graphs with 100 nodes each, generated using the preferential attachment (BA) model (Barabasi & Albert, 1999), in which graphs are constructed by incrementally adding new nodes with m edges and connecting them to existing nodes with a probability proportional to the degrees of those nodes. We adopt an inductive setting, where the graphs used during testing are not included in the training set. We used $m = 2$ for all the training graphs.

EXP and CEXP We followed the protocol of Abboud et al. (2021) and used Adam optimizer with a learning rate of $1e-4$. The network has 8 GraphConv layers followed by a sum-pooling and a 3 layer readout function. All hidden layers have 64 dimensions, and we used 64 random features and we did not discard the original features in the data.

Table 5: Performance across OGB datasets for different models with 8 random channels as IDs. We report mean ROC-AUC and standard deviation for the MOLHIV, MOLBACE, and MOLBBBP datasets, mean accuracy and standard deviation for the ARXIV and SQUIRREL datasets, and Hits@50 and standard deviation for the COLLAB dataset. For each model we report the performance with no IDs, with RNI, and with ICON. The best-performing approach within the standard deviation margin is highlighted in bold. In all cases, ICON outperforms RNI and maintains the same trends observed in the main paper.

Model	OGBN-ARXIV	OGBG-MOLHIV	OGBG-MOLBACE	OGBG-MOLBBBP	OGBL-COLLAB	SQUIRREL
GraphConv	$72.05_{\pm 1.13}$	$\mathbf{70.27}_{\pm 2.51}$	$62.58_{\pm 1.98}$	$63.39_{\pm 0.17}$	$46.15_{\pm 1.01}$	$32.4_{\pm 1.7}$
GraphConv + RNI	$70.89_{\pm 1.42}$	$55.87_{\pm 2.33}$	$51.78_{\pm 2.14}$	$58.92_{\pm 1.91}$	$40.02_{\pm 3.42}$	$28.8_{\pm 2.0}$
GraphConv + ICON	$\mathbf{73.41}_{\pm 1.14}$	$\mathbf{70.19}_{\pm 2.01}$	$\mathbf{65.19}_{\pm 1.13}$	$\mathbf{66.78}_{\pm 1.02}$	$\mathbf{47.92}_{\pm 1.68}$	$\mathbf{34.7}_{\pm 1.4}$
GIN	$76.30_{\pm 1.58}$	$\mathbf{72.09}_{\pm 1.01}$	$73.02_{\pm 1.27}$	$\mathbf{60.86}_{\pm 1.05}$	$43.87_{\pm 0.94}$	$30.2_{\pm 1.5}$
GIN + RNI	$71.46_{\pm 1.88}$	$70.83_{\pm 1.92}$	$62.19_{\pm 2.39}$	$53.12_{\pm 1.78}$	$39.12_{\pm 2.25}$	$26.9_{\pm 1.8}$
GIN + ICON	$\mathbf{79.03}_{\pm 0.87}$	$71.79_{\pm 1.33}$	$\mathbf{76.52}_{\pm 1.04}$	$60.21_{\pm 2.08}$	$\mathbf{45.61}_{\pm 1.09}$	$\mathbf{33.0}_{\pm 1.1}$
GAT	$\mathbf{72.76}_{\pm 1.30}$	$72.60_{\pm 2.49}$	$\mathbf{71.01}_{\pm 2.83}$	$\mathbf{59.42}_{\pm 2.52}$	$44.23_{\pm 1.23}$	$29.8_{\pm 2.0}$
GAT + RNI	$62.04_{\pm 3.04}$	$54.11_{\pm 2.04}$	$63.02_{\pm 1.48}$	$58.62_{\pm 1.93}$	$40.73_{\pm 1.96}$	$27.2_{\pm 1.5}$
GAT + ICON	$70.59_{\pm 2.91}$	$\mathbf{76.33}_{\pm 1.20}$	$68.10_{\pm 1.42}$	$\mathbf{63.14}_{\pm 1.23}$	$\mathbf{45.02}_{\pm 1.33}$	$\mathbf{32.4}_{\pm 1.2}$
TransformerConv	$73.15_{\pm 1.39}$	$\mathbf{70.11}_{\pm 1.49}$	$74.91_{\pm 2.04}$	$\mathbf{60.92}_{\pm 1.37}$	$44.67_{\pm 1.10}$	$33.6_{\pm 1.4}$
TransformerConv + RNI	$70.34_{\pm 1.57}$	$67.21_{\pm 1.44}$	$64.78_{\pm 1.89}$	$55.01_{\pm 1.81}$	$42.02_{\pm 1.90}$	$30.8_{\pm 1.6}$
TransformerConv + ICON	$\mathbf{75.62}_{\pm 2.02}$	$\mathbf{72.98}_{\pm 1.18}$	$\mathbf{74.26}_{\pm 1.51}$	$\mathbf{59.01}_{\pm 1.79}$	$\mathbf{46.21}_{\pm 1.15}$	$\mathbf{35.1}_{\pm 1.2}$
GraphGPS	$74.80_{\pm 1.99}$	$\mathbf{72.75}_{\pm 2.11}$	$69.94_{\pm 1.51}$	$\mathbf{58.17}_{\pm 2.01}$	$47.12_{\pm 0.88}$	$\mathbf{37.4}_{\pm 1.2}$
GraphGPS + RNI	$70.91_{\pm 2.11}$	$61.02_{\pm 3.74}$	$67.91_{\pm 1.43}$	$56.71_{\pm 2.36}$	$44.72_{\pm 1.27}$	$33.6_{\pm 1.6}$
GraphGPS + ICON	$\mathbf{76.81}_{\pm 2.12}$	$\mathbf{73.92}_{\pm 1.68}$	$\mathbf{71.84}_{\pm 1.47}$	$\mathbf{60.12}_{\pm 1.78}$	$\mathbf{48.02}_{\pm 0.96}$	$\mathbf{36.5}_{\pm 1.0}$

Table 6: Summary Statistics of OGB Datasets

Dataset	**# Graphs**	**Avg # Nodes**	**Avg # Edges**	**# Node Features**	**# Classes**
OGBN-ARXIV	1	169,343	1,166,243	128	40
OGBG-MOLHIV	41,127	25.5	27.5	9	2
OGBG-MOLBACE	1,513	34.1	36.9	9	2
OGBG-MOLBBBP	2,039	24.1	26.0	9	2
OGBL-COLLAB	1	235,868	1,285,465	128	$2^{\dagger}$
SQUIRREL	1	5,201	217,073	2,089	5

† For OGBL-COLLAB, the task is link prediction rather than classification, where edges between authors are predicted based on collaboration features.

Hyper-Parameters For all experiments, we use a fixed dropout rate of 0.1 and ReLU activations. We tuned the learning rate in $\{10^{-3}, 5 \cdot 10^{-4}\}$, batch size in $\{32, 64\}$ (for graph-level tasks), number of layers in $\{3, 5\}$, hidden dimensions in $\{32, 64\}$, and weight decay in $\{10^{-4}, 10^{-5}\}$. For node-level and link prediction tasks (OGBN-ARXIV, SQUIRREL, OGBL-COLLAB), we use full-graph training. Random seeds were fixed for reproducibility, and results are reported across three independent runs. All models were trained on a single NVIDIA V100 32GB GPU.

D Time Complexity Analysis

In this section, we analyze the computational complexity introduced by ICON relative to a standard GNN trained with RNI. Recall that ICON adds an explicit regularization term encouraging invariance to random ID values by comparing embeddings produced under two independent ID assignments.

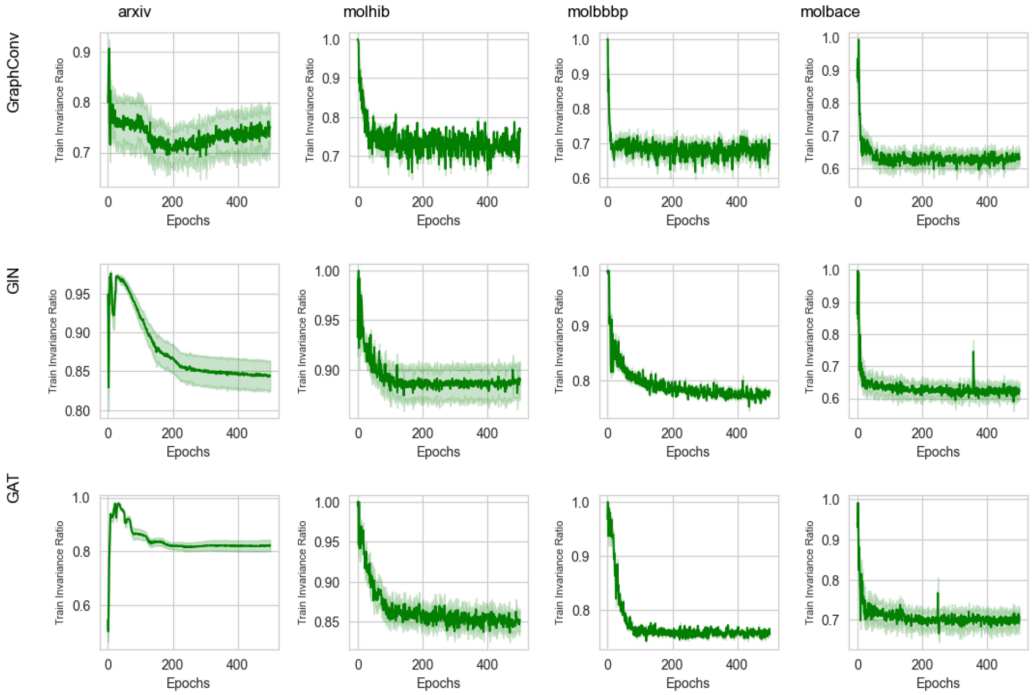


Figure 3: Invariance ratios over different GNNs with RNI, on real-world datasets, using 8 random channels as IDs. Despite the ID resampling that is applied in every epoch, invariance is not achieved and, in most cases, does not increase during training.

Let the base GNN-R (with random IDs) consist of L message-passing layers with hidden dimensions $\{d_\ell\}_{\ell=1}^{L}$. Denote by $|V|$ and $|E|$ the number of nodes and edges in the input graph, respectively. The computational complexity of a single forward pass of a message-passing GNN is

$$O\left(\sum_{\ell=1}^{L} |E| d_{\ell-1} + |V| d_\ell\right),$$

which corresponds to the cost of aggregating and updating node representations across all layers.

Additional cost of ICON. ICON introduces one additional forward pass per training batch to evaluate the same network under a second random ID assignment I'. Let T_f denote the cost of one forward pass and T_b the cost of one backward pass. The total training cost per iteration can then be expressed as:

$$T_{\text{RNI}} = T_f + T_b, \qquad T_{\text{ICON}} = 2T_f + 2T_b + O(|V| d_L),$$

where the additional $O(|V| d_L)$ term arises from computing the ℓ_2 regularization term $\|H_1^{(L)} - H_2^{(L)}\|_2^2$ at the final layer. The backward pass cost doubles because the regularization term depends on embeddings from both forward passes, requiring gradient computation through both computation graphs.

Therefore, the overall time complexity of ICON remains in the same asymptotic order as the base GNN, that is,

$$O(|E| d),$$

but with a multiplicative constant of approximately 2–3$\times$ due to the extra forward and backward passes. The precise overhead depends on implementation details and hardware-level parallelization efficiency. In practice, the cost can be mitigated because both forward passes share the same graph structure and model parameters, allowing for partial reuse of message computations and efficient batched evaluation on GPUs.

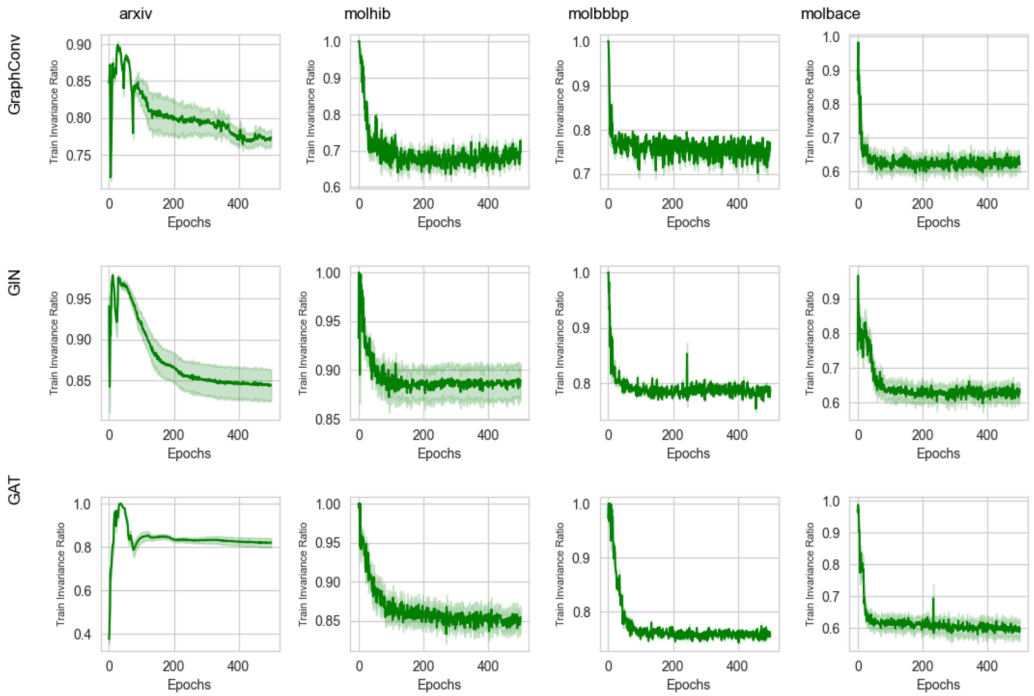


Figure 4: Invariance ratios over different GNNs with RNI, on real-world datasets, with weight decay 0.00001. Despite the ID resampling that is applied in every epoch, invariance is not achieved and, in most cases, does not increase during training.

Overall, ICON preserves the asymptotic time complexity of standard message-passing GNNs, with only a modest constant factor overhead. Specifically:

- **Asymptotic complexity:** $O(|E|d)$ per forward pass, same as standard GNNs.
- **Constant overhead:** one extra forward pass, one extra backward pass, and a lightweight vector difference at the final layer.
- **Empirical runtime:** In our experiments, ICON increased the total wall-clock training time by approximately 2–3$\times$, consistent with the doubling of both forward and backward computations, while often converging in fewer epochs due to improved invariance and optimization stability.

Overall, ICON provides a favorable trade-off between a small computational overhead and substantial gains in ID-invariance, generalization, and convergence speed.

E Future work

An intriguing open question is whether two layers of a GNN with unique identifiers (IDs) suffice to achieve additional expressiveness beyond the 1-WL test. We showed that using ID-invariance on all layers does not enhance expressiveness, whereas enforcing invariance solely in the final layer does. Fully understanding the conditions on layers regarding ID-invariance and identifying which functions can be computed under various settings presents an interesting direction for future research. Our experiments show that ID utilization can be improved with ICON, by *learning* to produce an invariant network. Another interesting open question is how to design GNNs that benefit from the added theoretical expressiveness of IDs *and* are invariant to IDs *by design*.