Identity-aware Graph Neural Networks

Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, Jure Leskovec
Department of Computer Science, Stanford University
{jiaxuan, jgs8, rexy, jure}@cs.stanford.edu

Abstract

Message passing Graph Neural Networks (GNNs) provide a powerful modeling framework for relational data. However, the expressive power of existing GNNs is upper-bounded by the 1-Weisfeiler-Lehman (1-WL) graph isomorphism test, which means GNNs that are not able to predict node clustering coefficients and shortest path distances, and cannot differentiate between different d-regular graphs. Here we develop a class of message passing GNNs, named *Identity-aware Graph Neural Networks (ID-GNNs)*, with greater expressive power than the 1-WL test. ID-GNN offers a minimal but powerful solution to limitations of existing GNNs. ID-GNN extends existing GNN architectures by inductively considering nodes' identities during message passing. To embed a given node, ID-GNN first extracts the ego network centered at the node, then conducts rounds of *heterogeneous message passing*, where different sets of parameters are applied to the center node than to other surrounding nodes in the ego network. We further propose a simplified but faster version of ID-GNN that injects node identity information as augmented node features. Altogether, both versions of ID-GNN represent general extensions of message passing GNNs, where experiments show that transforming existing GNNs to ID-GNNs yields on average 40% accuracy improvement on challenging node, edge, and graph property prediction tasks; 3% accuracy improvement on node and graph classification benchmarks; and 15% ROC AUC improvement on real-world link prediction tasks. Additionally, ID-GNNs demonstrate improved or comparable performance over other task-specific graph networks.

Introduction

Graph Neural Networks (GNNs) represent a powerful learning paradigm that have achieved great success (Scarselli et al. 2008; Li et al. 2016; Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Velickovic et al. 2018; Xu et al. 2019; You, Ying, and Leskovec 2020). Among these models, messaging passing GNNs, such as GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and GAT (Velickovic et al. 2018), are dominantly used today due to their simplicity, efficiency and strong performance in real-world applications (Zitnik and Leskovec 2017; Ying et al. 2018; You et al. 2018a, 2019b, 2020a,b). The central idea behind message passing GNNs is to learn node embeddings via the repeated aggregation of information from local node neighborhoods using non-linear transformations (Battaglia et al. 2018).

Although GNNs represent a powerful learning paradigm, it has been shown that the expressive power of existing GNNs is upper-bounded by the 1-Weisfeiler-Lehman (1-WL) test (Xu et al. 2019). Concretely, a fundamental limitation of existing GNNs is that two nodes with different neighborhood structure can have the same computational graph, thus appearing indistinguishable. Here, a computational graph specifies the procedure to produce a node's embedding. Such failure cases are abundant (Figure 1): in node classification tasks, existing GNNs fail to distinguish nodes that reside in d-regular graphs of different sizes; in link prediction tasks, they cannot differentiate node candidates with the same neighborhood structures but different shortest path distance to the source node; and in graph classification tasks, they cannot differentiate d-regular graphs (Chen et al. 2019; Murphy et al. 2019). While task-specific feature augmentation can be used to mitigate these failure modes, the process of discovering meaningful features for different tasks is not generic and can, for example, hamper the inductive power of GNNs.

Several recent methods aim to overcome these limitations in existing GNNs. For graph classification tasks, a collection of works propose novel architectures more expressive than the 1-WL test (Chen et al. 2019; Maron et al. 2019a; Murphy et al. 2019). For link level tasks, P-GNNs are proposed to overcome the limitation of existing GNNs (You, Ying, and Leskovec 2019). While these methods have a rich theoretical grounding, they are often task specific (either graph or link level) and often suffer from increased complexity in computation or implementation. In contrast, message passing GNNs have a track record of high predictive performance across node, link, and graph level tasks, while being simple and efficient to implement. Therefore, extending message passing GNNs beyond the expressiveness of 1-WL test, to overcome current GNN limitations, is a problem of high importance.

Present work. Here we propose Identity-aware Graph Neural Networks (ID-GNNs), a class of *message passing GNNs with expressive power beyond the 1-WL test*[1]. ID-GNN provides a universal extension and makes *any* existing message passing GNN more expressive. ID-GNN embeds each node

[1]Project website with code: http://snap.stanford.edu/idgnn

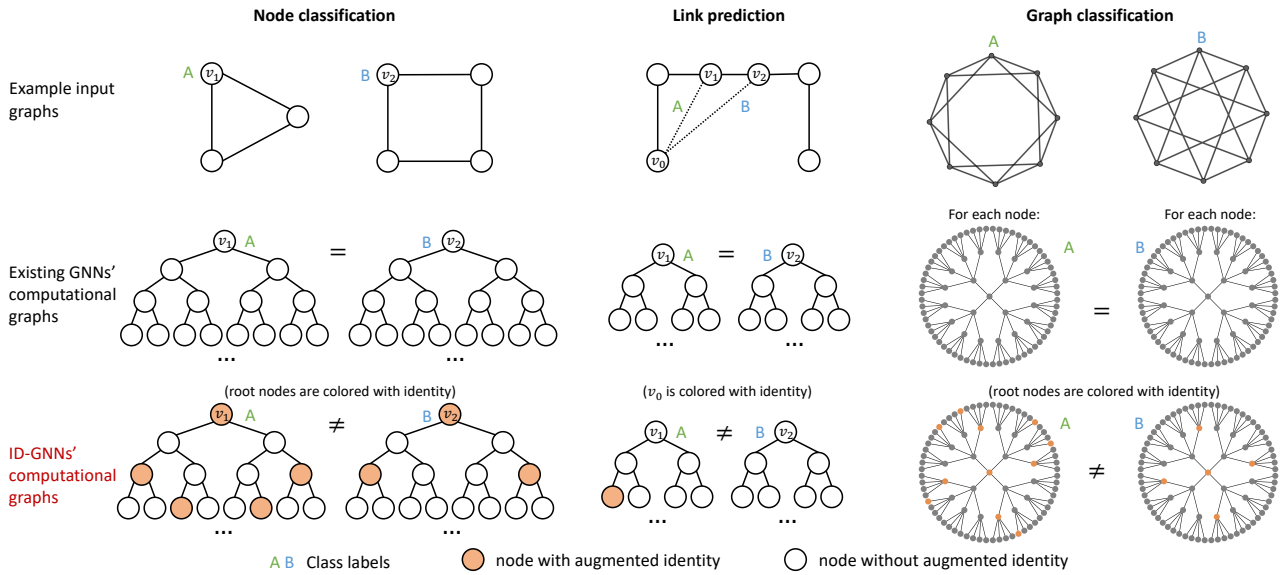

Figure 1: An overview of the proposed ID-GNN model. We consider node, edge and graph level tasks, and assume nodes do not have discriminative features. Across all examples, the task requires an embedding that allows for the differentiation of nodes labeled A vs. B in their respective graphs. However, across all tasks, existing GNNs, regardless of their depth, will *always* assign the same embedding to both nodes A and B, because for all tasks the computational graphs are identical (middle row). In contrast, the colored computational graphs provided by ID-GNN allow for clear differentiation between the nodes of label A and label B, as the colored computational graphs are no longer identical across the tasks.

by *inductively* taking into account its identity during message passing. The approach is different from labeling each node with a one-hot encoding, which is *transductive* (cannot generalize to unseen graphs). As shown in Figure 1, we use an *inductive identity coloring* technique to distinguish a node itself (the root node in the computational graph) from other nodes in its local neighborhood, within its respective computational graph. This added identity information allows ID-GNN to distinguish what would be identical computational graphs across node, edge and graph level tasks, and this way overcome the previously discussed limitations.

We propose two versions of ID-GNN. As a general approach, identity information is incorporated by applying rounds of *heterogeneous message passing*. Specifically, to embed a given node, ID-GNN first extracts the ego network centered at that node, then applies message passing, where the messages from the center node (colored nodes in Figure 1) and the rest of the nodes are computed using *different sets of parameters*. This approach naturally applies to applications involving node or edge features. We also consider a simplified version of ID-GNN, where we inject identity information via cycle counts originating from a given node as augmented node features. These cycle counts capture node identity information by counting the colored nodes within each layer of the ID-GNN computational graph, and can be efficiently computed by powers of a graph's adjacency matrix.

We compare ID-GNNs against GNNs across 8 datasets and 6 different tasks. First, we consider a collection of challenging graph property prediction tasks where existing GNNs fail, including predicting node clustering coefficient, predicting shortest path distance, and differentiating random d-regular graphs. Then, we further apply ID-GNNs to real-world datasets. Results show that transforming existing GNNs to their ID-GNN versions yields on average 40% accuracy improvement on challenging node, edge, and graph property prediction tasks; 3% accuracy improvement on node and graph classification benchmarks; and 15% ROC AUC improvement on real-world link prediction tasks. Additionally, we compare ID-GNNs against other expressive graph networks that are specifically designed for edge or graph-level tasks. ID-GNNs demonstrate improved or comparable performance over these models, further emphasizing the versatility of ID-GNNs.

Our key contribution includes: **(1)** We show that message passing GNNs can have expressive power beyond 1-WL test. **(2)** We propose ID-GNNs as a general solution to the limitations in existing GNNs, with rich theoretical and experimental results. **(3)** We present synthetic and real world tasks to reveal the failure modes of existing GNNs and demonstrate the superior performance of ID-GNNs over both existing GNNs and other powerful graph networks.

Related Work

Expressive neural networks beyond 1-WL test. Recently, many neural networks have been proposed with expressive power beyond the 1-WL test, including (Chen et al. 2019; Maron et al. 2019a; Murphy et al. 2019; You, Ying, and Leskovec 2019; Li et al. 2020). However, these papers introduce extra, often task/domain specific, components beyond standard message passing GNNs. For example, P-GNN's embeddings are tied with random anchor-sets and, thus, are not applicable to node/graph level tasks which require deter-

ministic node embeddings (You, Ying, and Leskovec 2019). In this paper we emphasize the advantageous characteristics of message passing GNNs, and show that GNNs, after incorporating inductive identity information, can surpass the expressive power of the 1-WL test while maintaining benefits of efficiency, simplicity, and broad applicability.

Graph Neural Networks with inductive coloring. Several models color nodes with augmented features to boost existing GNNs' performance (Xu et al. 2020; Veličković et al. 2020; Zhang and Chen 2018). However, existing coloring techniques are problem and domain-specific (*i.e.* link prediction, algorithm execution), and are not generally applicable to node and graph-level tasks. In contrast, ID-GNN is a general model that can be applied to any node, edge, and graph level task. It further adopts a heterogeneous message passing approach, which is fully compatible to cases where nodes or edges have rich features.

GNNs with anisotropic message passing. We emphasize that ID-GNNs are fundamentally different from GNNs based on anisotropic message passing, where different attention weights are applied to different incoming edges (Bresson and Laurent 2017; Hamilton, Ying, and Leskovec 2017; Monti et al. 2017; Velickovic et al. 2018). Adding anisotropic message passing does not change the underlying computational graph because the same message passing function is symmetrically applied across all nodes. Therefore, these models still exhibit the limitations summarized in Figure 1.

Preliminaries

A graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the node set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. Nodes can be paired with features $\mathcal{X} = \{\mathbf{x}_v | \forall v \in \mathcal{V}\}$, and edges can have features $\mathcal{F} = \{\mathbf{f}_{uv} | \forall e_{uv} \in \mathcal{E}\}$. As discussed earlier, we focus on message passing GNNs throughout this paper. We follow the definition of GNNs in (Xu et al. 2019). The goal of a GNN is to learn meaningful node embeddings $\mathbf{h}_v$ based on an iterative aggregation of local network neighborhoods. The k-th iteration of message passing, or the k-th layer of a GNN, can be written as:

$$\begin{aligned} \mathbf{m}_u^{(k)} &= \textsc{Msg}^{(k)}(\mathbf{h}_u^{(k-1)}), \\ \mathbf{h}_v^{(k)} &= \textsc{Agg}^{(k)}\big(\{\mathbf{m}_u^{(k)}, u \in \mathcal{N}(v)\}, \mathbf{h}_v^{(k-1)}\big) \end{aligned} \tag{1}$$

where $\mathbf{h}_v^{(k)}$ is the node embedding after k iterations, $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, $\mathbf{m}_v^{(k)}$ is the message embedding, and $\mathcal{N}(v)$ is the local neighborhood of v. Different GNNs have varied definitions of $\textsc{Msg}^{(k)}(\cdot)$ and $\textsc{Agg}^{(k)}(\cdot)$. For example, a GraphSAGE uses the definition ($\mathbf{W}^{(k)}$, $\mathbf{U}^{(k)}$ are trainable weights):

$$\begin{aligned} \mathbf{m}_u^{(k)} &= \textsc{ReLU}(\mathbf{W}^{(k)}\mathbf{h}_u^{(k-1)}), \\ \mathbf{h}_v^{(k)} &= \mathbf{U}^{(k)}\textsc{Concat}\big(\textsc{Max}\big(\{\mathbf{m}_u^{(k)}, u \in \mathcal{N}(v)\}\big), \mathbf{h}_v^{(k-1)}\big) \end{aligned} \tag{2}$$

The node embeddings $\mathbf{h}_v^{(K)}, \forall v \in \mathcal{V}$ are then used for node, edge, and graph level prediction tasks.

Identity-aware Graph Neural Networks

ID-GNNs: GNNs beyond the 1-WL test

We design ID-GNN so that it can make *any* message passing GNN more expressive. ID-GNN is built with two important components: **(1)** *inductive identity coloring* where identity information is injected to each node, and **(2)** *heterogeneous message passing* where the identity information is utilized in message passing. Algorithm 1 provides an overview.

Algorithm 1 ID-GNN embedding computation algorithm

Input: Graph $\mathcal{G}(\mathcal{V}; \mathcal{E})$, input node features $\{x_v, \forall v \in \mathcal{V}\}$; Number of layers K; trainable functions $\textsc{Msg}_1^{(k)}(\cdot)$ for nodes with identity coloring, $\textsc{Msg}_0^{(k)}(\cdot)$ for the rest of nodes; $\textsc{Ego}(v, k)$ extracts the K-hop ego network centered at node v, indicator function $\mathbb{1}[s = v] = 1$ if $s = v$ else 0

Output: Node embeddings $\mathbf{h}_v$ for all $v \in \mathcal{V}$

for $v \in \mathcal{V}$ **do**
  $\mathcal{G}_v^{(K)} \leftarrow \textsc{Ego}(v, K)$, $\quad \mathbf{h}_u^{(0)} \leftarrow \mathbf{x}_u, \forall u \in \mathcal{G}_v^{(K)}$
  for $k = 1, \ldots, K$ **do**
    for $u \in \mathcal{G}_v^{(K)}$ **do**
      $\mathbf{h}_u^{(k)} \leftarrow \textsc{Agg}^{(k)}\big(\{\textsc{Msg}_{\mathbb{1}[s=v]}^{(k)}(\mathbf{h}_s^{(k-1)}), s \in \mathcal{N}(u)\}, \mathbf{h}_u^{(k-1)}\big)$
  $\mathbf{h}_v \leftarrow \mathbf{h}_v^{(K)}$

Inductive identity coloring. To embed a given node $v \in \mathcal{G}$ using a K-layer ID-GNN, we first extract the K-hop ego network $\mathcal{G}_v^{(K)}$ of v. We then assign a unique coloring to the central node of the ego network $\mathcal{G}_v^{(K)}$. Altogether, nodes in $\mathcal{G}_v^{(K)}$ can be categorized into two types throughout the embedding process: nodes with coloring and nodes without coloring. This coloring technique is *inductive* because even if nodes are permuted, the center node of the ego network can still be differentiated from other neighboring nodes.

Heterogeneous message passing. K rounds of message passing are then applied to all the extracted ego networks. To embed node $u \in \mathcal{G}_v^{(K)}$, we extend Eq. 1 to enable heterogeneous message passing:

$$\begin{aligned} \mathbf{m}_s^{(k)} &= \textsc{Msg}_{\mathbb{1}[s=v]}^{(k)}(\mathbf{h}_s^{(k-1)}), \\ \mathbf{h}_u^{(k)} &= \textsc{Agg}^{(k)}\big(\{\mathbf{m}_s^{(k)}, s \in \mathcal{N}(u)\}, \mathbf{h}_u^{(k-1)}\big) \end{aligned} \tag{3}$$

where only $\mathbf{h}_v^{(K)}$ is used as the embedding representation for node v after applying K rounds of Eq. 3. Different from Eq. 1, two sets of $\textsc{Msg}^{(k)}$ functions are used, where $\textsc{Msg}_1^{(k)}(\cdot)$ is applied to nodes with identity coloring, and $\textsc{Msg}_0^{(k)}(\cdot)$ is used for node without coloring. The indicator function $\mathbb{1}[s = v] = 1$ if $s = v$ else 0 is used to index the selection of these functions. This way, the inductive identity coloring is *encoded into the ID-GNN computational graph*.

A benefit of this heterogeneous message passing approach is that it is *applicable to any message passing GNN*. For example, consider the following message passing scheme, which extends the definition of GNNs in Eq. 3 by including edge attributes $\mathbf{f}_{su}$ during message passing:

$$\begin{aligned} \mathbf{m}_{su}^{(k)} &= \textsc{Msg}_{\mathbb{1}[s=v]}^{(k)}(\mathbf{h}_s^{(k-1)}, \mathbf{f}_{su}), \\ \mathbf{h}_u^{(k)} &= \textsc{Agg}^{(k)}\big(\{\mathbf{m}_{su}^{(k)}, s \in \mathcal{N}(u)\}, \mathbf{h}_u^{(k-1)}\big) \end{aligned} \tag{4}$$

Algorithmic complexity. Besides adding the identity coloring and applying two types of message passing instead of

one, the computation of ID-GNN is almost identical to the widely used mini-batch version of GNNs (Hamilton, Ying, and Leskovec 2017; Ying et al. 2018). In our experiments, by matching the number of trainable parameters, the computation FLOPS used by ID-GNNs and mini-batch GNNs can be the same (shown in Table 4).

Extension to edge-level tasks. Here we discuss how to extend the ID-GNN framework to properly resolve existing GNN limitations in edge-level tasks (Figure 1, middle). Suppose we want to predict the edge-level label for a node pair u, v. For ID-GNN, the prediction is made from a *conditional node embedding* $\mathbf{h}_{u|v}$, which is computed by assigning node v, rather than u, identity coloring in node u's computation graph, as illustrated in Figure 1. In the case where node v does not lie within u's K-hop ego network, no identity coloring is used and ID-GNNs will still suffer from existing failure cases of GNNs. Therefore, we use deeper ID-GNNs for edge-level prediction tasks in practice.

ID-GNNs Expressive Power: Theoretical Results

ID-GNNs are strictly more expressive than existing message passing GNNs. It has been shown that existing message passing GNNs have an expressive power upper bound by the 1-WL test, where the upper bound can be instantiated by the Graph Isomophism Network (GIN) (Xu et al. 2019).

Proposition 1. *ID-GNN version of GIN can differentiate any graph that GIN can differentiate, while being able to differentiate certain graphs that GIN fails to distinguish.*

By setting $\textsc{Msg}_0^{(k)}(\cdot) = \textsc{Msg}_1^{(k)}(\cdot)$, Eq. 3 becomes identical to Eq. 1 which trivially proves the first part. The d-regular graph example given in Figure 1 then proves the second part.

ID-GNNs can count cycles. Proposition 1 provides an overview of the added expressive power of ID-GNNs. Here, we reveal one concrete aspect of this added expressive power, *i.e.*, ID-GNN's capability to count cycles. We observe that the ability of counting cycles is intuitive to understand; moreover, it is crucial for useful tasks such as predicting node clustering coefficient, which we elaborate in the next section.

Proposition 2. *For any node v, there exists a K-layer ID-GNN instantiation that can learn an embedding $\mathbf{h}_v^{(K)}$ where the j-th dimension $\mathbf{h}_v^{(K)}[j]$ equals the number of length j cycles starting and ending at node v, for $j = 1, ..., K$.*

We prove this by showing that ID-GNNs can count paths from any node u to the identity node v. Through induction, we show that a 1-layer ID-GNN embedding $\mathbf{h}_u^{(1)}$ can count length 1 paths from u to v. Then, given a K-layer ID-GNN embedding $\mathbf{h}_u^{(K)}$ that counts paths of length $1, \ldots, K$ between u and v, we show the $K+1$-th layer of ID-GNN can accurately update $\mathbf{h}_v^{(K+1)}$ to account for paths of length $K+1$. Detailed proofs are provided in the Appendix.

ID-GNNs Expressive Power: Case Studies

Node-level: Predicting clustering coefficient. Here we show that existing message passing GNNs fail to inductively predict clustering coefficients purely from graph structure, while ID-GNNs can. Clustering coefficient is a widely used metric that characterizes the proportion of closed triangles in a node's 1-hop neighborhood (Watts and Strogatz 1998). The node classification failure case in Figure 1 demonstrates GNNs' inability to predict clustering coefficients, as GNNs fail to differentiate nodes v_1 and v_2 with clustering coefficient 1 and 0 respectively. By using one-hot node features, GNNs can overcome this failure mode (Hamilton, Ying, and Leskovec 2017). However, in this case GNNs are *memorizing* the clustering coefficients for each node, since one-hot encodings prevent generalization to unseen graphs.

Based on Proposition 2, ID-GNNs can learn node embeddings $\mathbf{h}_v^{(K)}$, where $\mathbf{h}_v^{(K)}[j]$ equals the number of length j cycles starting and ending at node v. Given these cycle counts, we can then calculate clustering coefficient c_v of node v:

$$\begin{aligned} c_v &= \frac{|\{e_{su} : s, u \in \mathcal{N}(v), e_{su} \in \mathcal{E}\}|}{(d_v)(d_v-1)/2} \\ &= \frac{\mathbf{h}_v^{(K)}[3]}{\mathbf{h}_v^{(K)}[2] * (\mathbf{h}_v^{(K)}[2]-1)} \end{aligned} \tag{5}$$

where d_v is the degree of node v. Since c_v is a continuous function of $\mathbf{h}_v^{(K)}$, we can approximate it to an arbitrary ϵ precision with an MLP due to the universal approximation theorem (Hornik et al. 1989).

Edge-level: Predicting reachability or shortest path distance. Vanilla GNNs make edge-level predictions from pairs of node embeddings (Hamilton, Ying, and Leskovec 2017). However, this type of approaches fail to predict reachability or shortest path distance (SPD) between node pairs. For example, two nodes can have the same GNN node embedding, independent of whether they are located in the same connected component. Although (Veličković et al. 2020) shows that proper node feature initialization allows for the prediction of reachability and SPD, ID-GNNs present a general solution to this limitation through the use of conditional node embeddings. As discussed in "Extension to edge-level tasks", we re-formulate edge-level prediction as conditional node-level prediction; consequently, a K-layer ID-GNN can predict if node $u \in \mathcal{G}$ is reachable from $v \in \mathcal{G}$ within K hops by using the conditional node embedding $\mathbf{h}_{u|v}^{(K)}$ via:

$$\begin{aligned} \mathbf{m}_{s|v}^{(k)} &= \begin{cases} 1 & \text{if } \mathbb{1}[s=v]=1 \\ \mathbf{h}_{s|v}^{(k-1)} & \text{else} \end{cases}, \\ \mathbf{h}_{u|v}^{(k)} &= \textsc{Max}\big(\{\mathbf{m}_{s|v}^{(k)}, s \in \mathcal{N}(u)\}\big) \end{aligned} \tag{6}$$

where $\mathbf{h}_{u|v}^{(0)} = 0, \forall u \in \mathcal{G}$, and the output $\mathbf{h}_{u|v}^{(K)} = 1$ if an ID-GNN predicts u are reachable from v.

Graph-level: Differentiating random d-regular graphs. As is illustrated in Figure 1, existing message passing GNNs cannot differentiate random d-regular graphs purely from graph structure, as the computation graphs for each node are identical, regardless of the number of layers. Here, we show that ID-GNNs can differentiate a significant proportion of random d-regular graphs. Specifically, we generate 100 non-isomorphic random d-regular graphs and consider 3 settings with different graph sizes (n) and node degree (d). We use up to length K cycle counts, which a K-layer ID-GNN can

Table 1: Percentage of random d-regular graphs that ID-GNNs can differentiate (unique graph representations / total graphs). Note, none of the graphs can be differentiated by 1-WL test or GNNs regardless of the number of layers.

	ID-GNNs			
	Layer=3	Layer=4	Layer=5	Layer=6
n=64, d=4, 100 graphs	11%	64%	94%	100%
n=40, d=5, 100 graphs	14%	82%	100%	100%
n=96, d=6, 100 graphs	21%	88%	100%	100%

successfully represent (shown in Proposition 2), to calculate the percentage of these d-regular graphs that can be differentiated. Results in Table 1 confirm that the addition of identity information can greatly help differentiate d-regular graphs.

ID-GNN-Fast: Injecting Identity via Augmented Node Features

Given that: **(1)** mini-batch implementations of GNNs have computational overhead when extracting ego networks, which is required by ID-GNNs with heterogeneous message passing, and **(2)** cycle count information explains an important aspect of the added expressive power of ID-GNNs over existing GNNs, we propose ID-GNN-Fast, where we inject identity information by using cycle counts as augmented node features. Similar cycle count information is also shown to be useful in the context of graph kernels (Zhang et al. 2018). Following the definition in Proposition 3, we use the count of cycles with length $1, \ldots, K$ starting and ending at the node v as augmented node feature $\mathbf{x}_v^+ \in \mathbb{R}^K$. These additional features $\mathbf{x}_v^+$ can be computed efficiently with sparse matrix multiplication via $\mathbf{x}_v^+[k] = \text{Diag}(A^k)[v]$, where A is the adjacency matrix. We then update the input node attributes for all nodes by concatenating this augmented feature $\mathbf{x}_v = \textsc{Concat}(\mathbf{x}_v, \mathbf{x}_v^+)$.

Experiments

Experimental setup

Datasets. We perform experiments over 8 different datasets. We consider the synthetic graph datasets **(1)** `ScaleFree` (Holme and Kim 2002) and **(2)** `SmallWorld` (Watts and Strogatz 1998), each containing 256 graphs, with average degree of 4 and average clustering coefficient in the range $[0, 0.5]$. For real-world datasets we explore 3 protein datasets: **(3)** `ENZYMES` (Borgwardt et al. 2005) with 600 graphs, `(4)` `PROTEINS` (Schomburg et al. 2004) with 1113 graphs, and **(5)** `BZR` (Sutherland, O'brien, and Weaver 2003) with 405 graphs. We also consider citation networks including **(6)** `Cora` and **(7)** `CiteSeer` (Sen et al. 2008), and a large-scale molecule dataset **(8)** `ogbg-molhiv` (Hu et al. 2020) with 41K graphs.

Tasks. We evaluate ID-GNNs over two task categories. First, we consider challenging graph property prediction tasks: **(1)** classifying nodes by clustering coefficients, **(2)** classifying pairs of nodes by their shortest path distances, and **(3)** classifying random graphs by their average clustering coefficients. We bin over continuous clustering coefficients to make task (1) and (3) 10-way classification tasks and threshold the shortest path distance to make task (2) a 5-way classification task. We also consider more common tasks with real-world labels, including **(4)** node classification, **(5)** link prediction, and **(6)** graph classification. For the `ogbg-molhiv` dataset we use provided splits, while for all the other tasks, we use a random 80/20% train/val split and average results over 3 random splits. Validation accuracy (multi-way classification) or ROC AUC (binary classification) in the final epoch is reported.

Models. We present a standardized framework for fairly comparing ID-GNNs with existing GNNs. We use 4 widely adopted GNN models as base models: GAT (Velickovic et al. 2018), GCN (Kipf and Welling 2017), GIN (Xu et al. 2019), and GraphSAGE (Hamilton, Ying, and Leskovec 2017). We then transform each GNN model to its ID-GNN variants, ID-GNN-Full (based on heterogeneous message passing) and ID-GNN-Fast, holding all the other hyperparameters fixed. To further ensure fairness, we adjust layer widths, so that all the models match the number of trainable parameters of a standard GCN model (i.e., match computational budget). In summary, we run 12 models for each experimental setup, including 4 types of GNN architectures, each with 3 versions.

We use 3-layer GNNs for node and graph level tasks, and 5-layer GNNs for edge level tasks, where GCNs with 256-dim hidden units are used to set the computational budget for all 12 model variants. For ID-GNNs-Full, each layer has 2 sets of weights, thus each layer has fewer number of hidden units; for ID-GNNs-Fast, 10-dim augmented cycle counts features are used. We use ReLU activation and Batch Normalization for all the models. We use Adam optimizer with learning rate 0.01. Due to the different nature of these tasks, tasks (1)(3)(6) excluding the `ogbg-molhiv` dataset, are trained for 1000 epochs, while the rest are trained for 100 epochs. For node-level tasks, GNN / ID-GNN node embeddings are directly used for prediction; for edge-level tasks, ID-GNNs-Full make predictions with conditional node embeddings, while GNNs and ID-GNNs-Fast make predictions by concatenating pairs of node embeddings and then passing the result through a 256-dim MLP; for graph-level tasks, predictions are based on a global sum pooling over node embeddings.

Overall, these comprehensive and consistent experimental settings reveal the general improvement of ID-GNNs compared with existing GNNs.

Graph Property Prediction Tasks

Node clustering coefficient prediction. In Table 2 we observe that across all models and datasets, both ID-GNN formulations perform at the level of or significantly outperform GNN counterparts, with an average absolute performance gain of 36.8% between the best ID-GNN and best GNN. In each dataset, both ID-GNN methods perform with near 100% accuracy for at least one GNN architecture. ID-GNN-Fast shows the most consistent improvements across models with greatest improvement in GraphSAGE. These results align with the previous discussion of using cycle counts alone to learn clustering coefficients. We defer discussion until later on ID-GNN-Full sometimes showing minimal improvement, to present a general understanding of this behavior.

Table 2: **Comparing ID-GNNs with GNNs on graph property prediction tasks**. For each column, all the 12 models have *the same computational budget*. The best performance in each family of models is bold. Results are averaged over 3 random splits.

		Node classification: predict clustering coefficient				Edge classification: predict shortest path distance				Graph classification: predict clustering coefficient	
		ScaleFree	SmallWorld	ENZYMES	PROTEINS	ScaleFree	SmallWorld	ENZYMES	PROTEINS	ScaleFree	SmallWorld
GNNs	GCN	0.679±0.01	**0.589±0.04**	0.596±0.02	0.540±0.00	0.522±0.00	0.558±0.02	**0.557±0.02**	**0.722±0.01**	0.270±0.06	0.433±0.03
	SAGE	0.470±0.03	0.271±0.03	0.572±0.04	0.444±0.03	0.297±0.01	0.360±0.16	0.550±0.02	0.722±0.01	0.047±0.03	0.077±0.01
	GAT	0.470±0.03	0.274±0.06	0.464±0.03	0.400±0.02	0.451±0.00	0.551±0.02	0.556±0.02	0.722±0.01	0.127±0.04	0.093±0.03
	GIN	**0.693±0.00**	0.571±0.04	**0.660±0.02**	**0.558±0.02**	**0.551±0.01**	**0.575±0.02**	0.541±0.03	0.722±0.01	**0.280±0.01**	**0.453±0.03**
ID-GNNs Fast	GCN	0.897±0.01	0.812±0.02	0.786±0.04	0.805±0.02	0.521±0.00	0.576±0.02	0.553±0.03	0.722±0.01	0.823±0.04	**0.850±0.06**
	SAGE	**0.954±0.01**	**0.994±0.00**	**0.958±0.04**	**0.985±0.01**	0.527±0.01	**0.583±0.02**	0.551±0.04	**0.722±0.01**	**0.827±0.02**	0.810±0.04
	GAT	0.889±0.01	0.739±0.03	0.675±0.04	0.675±0.05	0.471±0.00	0.574±0.02	0.545±0.03	0.722±0.01	0.620±0.01	0.800±0.06
	GIN	0.895±0.00	0.822±0.03	0.798±0.06	0.790±0.01	**0.546±0.00**	0.576±0.02	**0.556±0.03**	0.722±0.01	0.730±0.02	0.840±0.06
ID-GNNs Full	GCN	***0.985±0.01***	***0.994±0.00***	0.984±0.02	***0.995±0.00***	0.999±0.00	1.000±0.00	0.994±0.00	0.998±0.00	***0.830±0.03***	***0.877±0.05***
	SAGE	0.588±0.01	0.400±0.12	0.591±0.04	0.474±0.03	***1.000±0.00***	***1.000±0.00***	***1.000±0.00***	1.000±0.00	0.247±0.01	0.250±0.10
	GAT	0.638±0.01	0.847±0.20	***0.994±0.00***	0.994±0.00	0.984±0.00	0.989±0.01	0.963±0.01	0.993±0.01	0.047±0.03	0.067±0.01
	GIN	0.716±0.01	0.572±0.04	0.655±0.02	0.570±0.03	1.000±0.00	0.964±0.05	1.000±0.00	***1.000±0.00***	0.273±0.02	0.490±0.01
Best ID-GNN over best GNN		29.3%	40.6%	33.4%	43.7%	44.9%	42.5%	44.3%	27.8%	55.0%	42.3%

Table 3: **Comparing GNNs with ID-GNNs on real-world prediction tasks**. For each column, all the 12 models have *the same computational budget*. The best performance in each family of models is bold. Results are averaged over 3 random splits.

		Node classification: real-world labels		Edge classification: link prediction				Graph classification: real-world labels			
		Cora	CiteSeer	ScaleFree	SmallWorld	ENZYMES	PROTEINS	ENZYMES	PROTEINS	BZR	ogbg-molhiv
GNNs	GCN	0.848±0.01	0.709±0.01	0.796±0.01	0.709±0.00	0.651±0.01	0.659±0.01	0.547±0.01	0.695±0.02	0.844±0.04	0.747±0.02
	SAGE	**0.868±0.01**	**0.726±0.01**	0.541±0.00	0.512±0.00	0.546±0.01	0.582±0.01	0.542±0.01	0.692±0.01	0.852±0.04	0.758±0.01
	GAT	0.857±0.01	0.716±0.01	0.500±0.00	0.500±0.00	0.478±0.01	0.491±0.01	**0.555±0.02**	**0.723±0.00**	0.848±0.03	0.742±0.01
	GIN	0.858±0.01	0.719±0.01	**0.802±0.01**	**0.722±0.01**	**0.654±0.01**	**0.667±0.00**	0.553±0.02	0.721±0.00	**0.856±0.02**	**0.762±0.03**
ID-GNNs Fast	GCN	0.851±0.02	0.715±0.00	0.856±0.03	0.719±0.00	0.649±0.01	0.671±0.01	0.600±0.01	***0.741±0.02***	0.807±0.02	0.772±0.02
	SAGE	0.866±0.02	***0.742±0.01***	***0.898±0.01***	0.743±0.02	0.671±0.04	0.701±0.01	***0.639±0.00***	0.724±0.03	0.835±0.06	**0.780±0.01**
	GAT	**0.870±0.02**	0.719±0.02	0.731±0.02	0.537±0.00	0.490±0.01	0.502±0.01	0.619±0.03	0.715±0.03	0.848±0.05	0.740±0.01
	GIN	0.864±0.01	0.719±0.01	0.837±0.01	**0.759±0.01**	**0.718±0.02**	**0.724±0.00**	0.567±0.01	0.723±0.01	**0.864±0.03**	0.755±0.02
ID-GNNs Full	GCN	0.863±0.01	0.719±0.01	0.771±0.04	0.798±0.03	0.838±0.01	0.878±0.02	**0.586±0.04**	0.715±0.02	0.881±0.04	0.769±0.01
	SAGE	0.875±0.01	**0.730±0.02**	0.741±0.01	0.724±0.03	0.819±0.01	0.863±0.01	0.547±0.02	0.721±0.01	0.864±0.02	***0.783±0.02***
	GAT	***0.878±0.01***	0.729±0.01	0.749±0.01	0.742±0.03	0.824±0.01	0.859±0.03	0.567±0.05	**0.738±0.01**	***0.881±0.04***	0.739±0.01
	GIN	0.851±0.00	0.725±0.01	**0.815±0.01**	***0.810±0.03***	***0.846±0.01***	***0.886±0.02***	0.544±0.02	0.730±0.03	0.852±0.03	0.756±0.00
Best ID-GNN over best GNN		1.0%	1.6%	9.6%	8.7%	19.2%	21.9%	8.3%	1.8%	2.5%	2.0%

Shortest path distance prediction. In the pairwise shortest path prediction task, ID-GNNs-Full outperform GNNs by an average of 39.9%. Table 2 reveals that ID-GNN-Full performs with 100% or near 100% accuracy under all GNN architectures, across all datasets. This observation, along with the comparatively poor performance of ID-GNNs-Fast and GNNs, confirms the previously discussed conclusion that traditional edge-level predictions, through pairwise node embeddings, fail to accurately make edge-level predictions.

Average clustering coefficient prediction for random graphs. In Table 2, we observe that adding identity information results in a 55% and 42.3% increase in best performance over `ScaleFree` and `SmallWorld` graphs respectively. ID-GNN-Fast shows the most consistent improvement (56.9% avg. model gain), which aligns with previous intuitions about the utility of cycle count information in predicting clustering coefficients and differentiating random graphs.

Real-world Prediction Tasks

Node classification. In node classification we see smaller but still significant improvements when using ID-GNNs. Table 3 shows an overall 1% and 1.6% improvement for `Cora` and `CiteSeer` respectively. In all cases except for GIN and GraphSAGE on `Cora`, adding identity information improves performance. In regards to the relatively small improvements, we hypothesize that the richness of node features (over 1000-dim for both datasets) greatly dilutes the importance of graph structure in these tasks, and thus the added expressiveness from identity information is diminished.

Link prediction. As shown in Table 3, we observe consistent improvement in ID-GNNs over GNNs, with 9.2% and 20.6% ROC AUC improvement on synthetic and real-world graphs respectively. Moreover, we observe that ID-GNN-Full nearly always performs the best, aligning with previous edge-level task results in Table 2 and intuitions on the importance of reformulating edge-level tasks as conditional node prediction tasks. We observe that performance improves less for random graphs, which we hypothesize is due to the randomness within these synthetic graphs causing the distinction between positive and negative edges to be much more vague.

Graph classification. Across each dataset, we observe that the best ID-GNN consistently outperforms the best GNN of the same computational budget. However, model to model improvement is less clear. For the `ENZYMES` dataset, ID-GNN-Fast shows strong improvements under each GNN architecture, with gains as large as 10% in accuracy for the

Table 4: Runtime analysis for GCN and ID-GNN equivalents given the same computational budget. For each model, average time (milisecond) per batch of 128 ENZYME graphs is reported for the forward and the forward + backward pass.

	GCN	ID-GNN-Fast	GCN (mini-batch)	ID-GNN-Full
forward	4.8±0.1	4.9±0.1	28.1±0.1	24.2±4.0
forward + backward	8.9±0.7	10.0±0.6	33.3±0.9	31.1±0.8

GraphSAGE model. In PROTEIN and BZR, ID-GNN-Full shows improvements for each GNN model (except GIN on BZR), with greatest performance increases in GCN and GAT (avg. 3.6% and 3.0% respectively).

Computational Cost Analysis

We compare the runtime complexity (excluding mini-batch loading time) of ID-GNNs vs. existing GNNs, where we hold the computational budget constant across all models. Table 4 reveals that when considering the forward and backward pass, ID-GNN-Full runs 3.8x slower than its GNN equivalent but has an equivalent runtime complexity to the mini-batch implementation of GNN, while ID-GNN-Fast runs with essentially zero overhead over existing GNN implementations.

Summary of Comparisons with GNNs

Overall, ID-GNN-Full and ID-GNN-Fast demonstrate significant improvements over their message passing GNN counterparts, of the same computational budget, on a variety of tasks. In all tasks, the best ID-GNNs outperforms the best GNNs; moreover, out of 160 model-task combinations, ID-GNNs fail to improve accuracy in fewer than 10 cases. For the rare cases where there is no improvement from ID-GNN-Full, we suspect that the model *underfits* since we control the complexity of models: given that ID-GNN-Full has two sets of weights (heterogeneous message passing), fewer weights are used for each message passing. For verification, if we double the computational budget, we observe that ID-GNN versions again outperform GNN counterparts.

Comparisons with Expressive Graph Networks

We provide additional experimental comparisons against other expressive graph networks in both edge and graph-level tasks. For edge-level task, we further compare with P-GNN (You, Ying, and Leskovec 2019) over the ENZYMES and PROTEINS datasets using the protocol introduced previously. For graph-level comparison, we include experimental results over 3 datasets: MUTAG with 182 graphs (Debnath et al. 1991), PTC with 344 graphs (Helma and Kramer 2003), and PROTEINS. We follow PPGN's (Maron et al. 2019a) 10-fold 90/10 data splits and compare against 5 other expressive graph networks. We report numbers in the corresponding papers, and report the best ID-GNNs out of the 4 variants.

Link prediction. We compare against P-GNNs on 2 link prediction datasets. As shown in the Table 5, we observe significant improvements using ID-GNNs compared to both its GNN counterpart and P-GNNs. These results both demonstrate ID-GNNs' competitive performance as a general graph learning method against a task-specific model, while also highlighting ID-GNN's improved ability to incorporate node-features compared with P-GNNs.

Table 5: Comparisons with P-GNN on link prediction task.

	Edge classification: link prediction	
	ENZYMES	PROTEINS
Best GNN	0.654±0.015	0.667±0.002
P-GNN (You, Ying, and Leskovec 2019)	0.715±0.024	0.810±0.013
Best ID-GNN-Fast	0.718±0.010	0.724±0.015
Best ID-GNN-Full	**0.846±0.010**	**0.886±0.015**

Table 6: Comparisons with other expressive graph networks on graph classification tasks. We use evaluation setup from (Maron et al. 2019a), and the reported numbers in corresponding papers are shown.

	Graph classification: real-world labels		
	MUTAG	PTC	PROTEINS
Best GNN	0.905±0.057	0.617±0.046	0.773±0.037
PPGN (Maron et al. 2019a)	0.906±0.087	0.662±0.065	0.772±0.047
CCN (Kondor et al. 2018)	0.916±0.072	**0.706 ±0.07**	NA
1-2-3 GNN (Morris et al. 2019)	0.861	0.609	0.755
Invariant GNs (Maron et al. 2019b)	0.846±0.10	0.595 ± 0.073	0.752 ± 0.043
GSN (Bouritsas et al. 2020)	0.922 ± 0.075	0.682 ± 0.072	0.766 ± 0.050
Best ID-GNN-Fast	**0.965±0.032**	0.619±0.054	**0.780±0.035**
Best ID-GNN-Full	**0.930±0.056**	0.625±0.053	**0.779±0.024**

Graph classification. We compare ID-GNNs against several other more powerful graph networks in the task of graph classification. Table 6 demonstrates the strong performance of ID-GNNs. ID-GNNs outperform other graph networks on the MUTAG and PROTEINS datasets; Although ID-GNNs performance then drops on the PTC dataset, they are still comparable to two out of the four powerful graph models. These strong results further demonstrate the ability of ID-GNN to outperform not only message passing GNNs, but also other powerful, task specific graph networks across a range of tasks.

Conclusion

We have proposed ID-GNNs as a general and powerful extension to existing GNNs with rich theoretical and experimental results. Specifically, ID-GNNs have expressive power beyond the 1-WL test. When runtime efficiency is the primary concern, we also present a feature augmented version of ID-GNN that maintains theoretical guarantees and empirical success of heterogeneous message passing, while only requiring one-time feature pre-processing. We recommend that this cycle-count feature augmentation be the new go-to node feature initialization when additional node attributes are not available. Additionally, as direct extensions to message passing GNNs, ID-GNNs can be easily implemented and extended via existing code platform. Overall, ID-GNNs outperform corresponding message passing GNNs, while both maintaining the attractive proprieties of message passing GNNs and demonstrating competitive performance compared with other powerful/expressive graph networks. We hope ID-GNNs' added expressive power and proven practical applicability can enable exciting new applications and further development of message passing GNNs.

Ethics Statement

GNNs represent a promising family of models for analyzing and understanding relational data. A broad range of application domains, such as network fraud detection (Akoglu, Chandy, and Faloutsos 2013; Kumar, Cheng, and Leskovec 2017; Akoglu and Faloutsos 2013), molecular drug structure discovery (You et al. 2018a,b; Jin, Barzilay, and Jaakkola 2018), recommender systems (Ying et al. 2018; You et al. 2019a), and network analysis (Kumar, Cheng, and Leskovec 2017; Morris et al. 2019; Fan et al. 2019; Ying et al. 2019) stand to be greatly impacted by the use and development of GNNs. As a direct extension of existing message passing GNNs, ID-GNNs represent a simple but powerful transformation to GNNs that re-frames the discussion on GNN expressive power and thus their performance in impactful problem domains. In comparison to other models that have expressive power beyond 1-WL tests, ID-GNNs are easy to implement with existing graph learning packages; therefore, ID-GNNs can be easily used as extensions of existing GNN models for tackling important real-world tasks, as well as themselves extended and further explored in the research space.

The simplicity of ID-GNNs presents great promise for further exploration into the expressiveness of GNNs. In particular, we believe that our work motivates further research into heterogeneous message passing and coloring schemes, as well as generic, but powerful forms of feature augmentation. By further increasing the expressiveness of message passing GNNs, we hopefully enable new, important tasks to be solved across a wide range of disciplines or significant improvement on previously defined and widely adopted GNN models. Through ease of use and strong preliminary results, we believe that our work opens the doors for new explorations into the study of graphs and graph based tasks, with the potential for great improvement in existing GNN models.

Acknowledgments

We gratefully acknowledge the support of DARPA under Nos. FA865018C7880 (ASED), N660011924033 (MCS); ARO under Nos. W911NF-16-1-0342 (MURI), W911NF-16-1-0171 (DURIP); NSF under Nos. OAC-1835598 (CINES), OAC-1934578 (HDR), CCF-1918940 (Expeditions), IIS-2030477 (RAPID); Stanford Data Science Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Biohub, Amazon, Boeing, JPMorgan Chase, Docomo, Hitachi, JD.com, KDDI, NVIDIA, Dell. J. L. is a Chan Zuckerberg Biohub investigator. Jiaxuan You is supported by JPMorgan Chase PhD Fellowship and Baidu Scholarship. Rex Ying is supported by Baidu Scholarship.

References

Akoglu, L.; Chandy, R.; and Faloutsos, C. 2013. Opinion fraud detection in online reviews by network effects. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Akoglu, L.; and Faloutsos, C. 2013. Anomaly, event, and fraud detection in large network datasets. In *ACM International Conference on Web Search and Data Mining (WSDM)*.

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

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S.; Smola, A. J.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21(suppl_1): i47–i56.

Bouritsas, G.; Frasca, F.; Zafeiriou, S.; and Bronstein, M. M. 2020. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *arXiv preprint arXiv:2006.09252* .

Bresson, X.; and Laurent, T. 2017. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553* .

Chen, Z.; Villar, S.; Chen, L.; and Bruna, J. 2019. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34(2): 786–797.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The Web Conference (WWW)*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Helma, C.; and Kramer, S. 2003. A survey of the predictive toxicology challenge 2000–2001. *Bioinformatics* 19(10): 1179–1182.

Holme, P.; and Kim, B. J. 2002. Growing scale-free networks with tunable clustering. *Physical review E* 65(2): 026107.

Hornik, K.; Stinchcombe, M.; White, H.; et al. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5): 359–366.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *Advances in Neural Information Processing Systems (NeurIPS)* .

Jin, W.; Barzilay, R.; and Jaakkola, T. 2018. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning (ICML)* .

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

Kondor, R.; Son, H. T.; Pan, H.; Anderson, B.; and Trivedi, S. 2018. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144* .

Kumar, S.; Cheng, J.; and Leskovec, J. 2017. Antisocial behavior on the web: Characterization and detection. In *The Web Conference (WWW)*.

Li, P.; Wang, Y.; Wang, H.; and Leskovec, J. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. *Advances in Neural Information Processing Systems (NeurIPS)* .

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2016. Gated graph sequence neural networks. *International Conference on Learning Representations (ICLR)* .

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019a. Provably powerful graph networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Maron, H.; Ben-Hamu, H.; Shamir, N.; and Lipman, Y. 2019b. Invariant and Equivariant Graph Networks. In *International Conference on Machine Learning (ICML)*.

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Murphy, R. L.; Srinivasan, B.; Rao, V.; and Ribeiro, B. 2019. Relational pooling for graph representations. *International Conference on Machine Learning (ICML)* .

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1): 61–80.

Schomburg, I.; Chang, A.; Ebeling, C.; Gremse, M.; Heldt, C.; Huhn, G.; and Schomburg, D. 2004. BRENDA, the enzyme database: updates and major new developments. *Nucleic acids research* 32(suppl_1): D431–D433.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3): 93–93.

Sutherland, J. J.; O'brien, L. A.; and Weaver, D. F. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *Journal of chemical information and computer sciences* 43(6): 1906–1915.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. *International Conference on Learning Representations (ICLR)* .

Veličković, P.; Ying, R.; Padovano, M.; Hadsell, R.; and Blundell, C. 2020. Neural execution of graph algorithms. *International Conference on Learning Representations (ICLR)* .

Watts, D. J.; and Strogatz, S. H. 1998. Collective dynamics of 'small-world'networks. *Nature* 393(6684): 440.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? *International Conference on Learning Representations (ICLR)* .

Xu, K.; Li, J.; Zhang, M.; Du, S. S.; Kawarabayashi, K.-i.; and Jegelka, S. 2020. What Can Neural Networks Reason About? *International Conference on Learning Representations (ICLR)* .

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* .

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

You, J.; Leskovec, J.; He, K.; and Xie, S. 2020a. Graph Structure of Neural Networks. *International Conference on Machine Learning (ICML)* .

You, J.; Liu, B.; Ying, R.; Pande, V.; and Leskovec, J. 2018a. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *Advances in Neural Information Processing Systems (NeurIPS)* .

You, J.; Ma, X.; Ding, D.; Kochenderfer, M.; and Leskovec, J. 2020b. Handling Missing Data with Graph Representation Learning. *Advances in Neural Information Processing Systems (NeurIPS)* .

You, J.; Wang, Y.; Pal, A.; Eksombatchai, P.; Rosenburg, C.; and Leskovec, J. 2019a. Hierarchical temporal convolutional networks for dynamic recommender systems. In *The Web Conference (WWW)*.

You, J.; Wu, H.; Barrett, C.; Ramanujan, R.; and Leskovec, J. 2019b. G2SAT: Learning to Generate SAT Formulas. In *Advances in Neural Information Processing Systems (NeurIPS)*.

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware graph neural networks. *International Conference on Machine Learning (ICML)* .

You, J.; Ying, R.; and Leskovec, J. 2020. Design Space for Graph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

You, J.; Ying, R.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018b. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning (ICML)*.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, Z.; Wang, M.; Xiang, Y.; Huang, Y.; and Nehorai, A. 2018. Retgk: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 3964–3974.

Zitnik, M.; and Leskovec, J. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33(14): i190–i198.

Proof of Proposition 2

Consider an arbitrary node v in a graph $G = (V, E)$ for which we want to compute the embedding under ID-GNN. Without loss of generality assume that $x_u = [1], \forall u \in V$. Additionally, let $\text{MSG}_0^{(k)}(\cdot)$, $\text{MSG}_1^{(k)}(\cdot)$, and $\text{AGG}^{(k)}(\cdot)$, from main paper Eq. 3 (the general formulation for the k-th layer of heterogeneous message passing), be defined as follows:

$$\begin{aligned} &\text{MSG}_0^{(k)}(\cdot) = W_0^k h_u^{k-1} + b_0^k, \quad \text{MSG}_1^{(k)}(\cdot) = W_1^k h_u^{k-1} + b_1^k, \\ &\text{AGG}^{(k)}(\cdot) = \text{SUM}(\{m_u^k, u \in N(w)\}) \end{aligned} \tag{7}$$

where W_0^k, b_0^k are trainable weights for nodes without identity coloring, and W_1^k, b_1^k are for nodes with identity coloring. Assume the following weight matrix assignments for different layers k:

k = 1: Let $W_0^1 = [0]$ (i.e the 0 matrix), $b_1^0 = [0]$, $W_1^1 = [0]$, and $b_1^1 = [1]$.

k = 2, . . . , K: Let $W_0^k = W_1^k = [0, I]^T \in \mathbb{R}^{k \times (k-1)}$ with identity matrix $I \in \mathbb{R}^{(k-1)\times(k-1)}$, $b_0^k = [0, ..., 0]^T \in \mathbb{R}^k$, and $b_1^k = [1, 0, ..., 0]^T \in \mathbb{R}^k$.

We will first prove Lemma 1 by induction.

Lemma 1: *After n-layers of heterogeneous message passing w/r to the identity colored node v, the embedding $h_u^n \in R^n, \forall u \in V$ is such that $h_u^n[j]$ = the number of paths of length (j) starting at node u and ending at the identity node v for $j = 1, ..., n$.*

Proof of Lemma 1:

Base case: We consider the result after 1 layer of message passing. For an arbitrary node u, we see that:

$$h_u^1 = \sum_{w \in N(u)} \big([0]h_w^{k-1} + [1]\mathbb{1}[w = v]\big) = \sum_{w \in N(u)} [1]\mathbb{1}[w = v] \tag{8}$$

where the indicator function $\mathbb{1}[w = v] = 1$ if $w = v$ else 0 is used to reflect node coloring. We see that $h_u^1[1] =$ exactly the number of paths of length 1 from u to the identity node v.

Inductive Hypothesis: We assume that after k-layers of heterogeneous message passing the embedding $h_u^k \in R^k, \forall u \in V$ is such that $h_u^k[j]$ = the number of paths of length j starting at node u and ending at the identity node v for $j = 1, ..., k$. We will prove that after one more layer of message passing or the $(k + 1)$th layer of ID-GNN the desired property still holds for the updated embeddings $h_u^{k+1} \in R^{k+1}, \forall u \in V$. To do so we consider the update for an arbitrary node u:

$$\begin{aligned} h_u^{k+1} &= \sum_{w \in N(u)} \big([0, I]^T h_w^k + [1, 0, ..., 0]^T \mathbb{1}[w = v]\big) \\ &= \sum_{w \in N(u)} [\mathbb{1}[w = v], h_w^k]^T \end{aligned} \tag{9}$$

We see that $h_u^{k+1}[1] =$ the number of length 1 paths to the identity node v, and for $j = 2, ..., k + 1 \rightarrow h_u^{k+1}[j] = \sum_{w \in N(u)} h_w^k[j - 1]$, which by our inductive hypothesis is exactly the number of paths of length j from node u to v. To see this, we notice that for the node u, we sum up all of the paths of length $j - 1$ from the neighboring nodes of u to the destination node v in order to get the number of paths of length j from u to v. Moreover, we see that each path of length $j - 1$ is now 1 step longer after an extra layer of message passing giving the desired paths of length j. We have thus shown that after $k + 1$ layers of message passing the embedding h_u^{k+1} has the desired properties completing induction.

Proposition 2 directly follows from the result of Lemma 1. Namely, if we choose the identity node v itself, by Lemma 1 we can learn an embedding such that h_v^k satisfies $h_v^k[j] =$ the number of length j paths from v to v, or equivalently the number of cycles starting and ending at node v, for $j = 1, \ldots, k$.

Memory consumption of ID-GNN

Here we discuss the potential memory consumption overhead of ID-GNNs compared with mini-batch GNNs. In ID-GNN, we adopted GraphSAGE (Hamilton, Ying, and Leskovec 2017) style of mini-batch GNN implementation, where disjoint extracted ego-nets are fed into a GNN. Given that we control the computational complexity, ID-GNN-Full has no memory overhead compared with GraphSAGE-style mini-batch GNN. This argument is supported by Table 4 (run-time analysis) in the main manuscript as well.

Mini-batch GNNs can be made more memory efficient by leveraging overlap within the extracted ego-nets; consequently, the embeddings of these nodes only need to be computed once. Note that while this approach saves the embedding computation of GNNs, it increases the time for processing mini-batches, since nodes from different ego-nets need to be aligned and deduplicated. Comparing the memory usage of ID-GNNs with this memory-efficient mini-batch GNN, the increase in memory usage is moderate. Comparison over `CiteSeer` reveals that a 3-layer ID-GNN takes 34%, 79% and 162% more memory under batch sizes 16, 32 and 64. Moreover, since mini-batch GNNs are often used for graphs larger than `CiteSeer`, where the percentage of common nodes between ego-nets is likely small, the overhead of ID-GNN's memory consumption will be even lower.