Transfer Learning of Graph Neural Networks with Ego-graph Information Maximization

Qi Zhu[1]*, **Carl Yang**[2]*, **Yidan Xu**[3], **Haonan Wang**[1], **Chao Zhang**[4], **Jiawei Han**[1]
[1]University of Illinois Urbana-Champaign, [2]Emory University,
[3]University of Washington, [4]Georgia Institute of Technology
[1]{qiz3,haonan3,hanj}@illinois.edu, [2]j.carlyang@emory.edu,
[3]yx2516@uw.edu, [4]chaozhang@gatech.edu

Abstract

Graph neural networks (GNNs) have achieved superior performance in various applications, but training dedicated GNNs can be costly for large-scale graphs. Some recent work started to study the pre-training of GNNs. However, none of them provide theoretical insights into the design of their frameworks, or clear requirements and guarantees towards their transferability. In this work, we establish a theoretically grounded and practically useful framework for the transfer learning of GNNs. Firstly, we propose a novel view towards the *essential graph information* and advocate the capturing of it as the goal of transferable GNN training, which motivates the design of EGI (*Ego-Graph Information maximization*) to analytically achieve this goal. Secondly, when node features are structure-relevant, we conduct an *analysis of* EGI *transferability* regarding the difference between the local graph Laplacians of the source and target graphs. We conduct controlled synthetic experiments to directly justify our theoretical conclusions. Comprehensive experiments on two real-world network datasets show consistent results in the analyzed setting of direct-transfering, while those on large-scale knowledge graphs show promising results in the more practical setting of transfering with fine-tuning.[1]

1 Introduction

Graph neural networks (GNNs) have been intensively studied recently [29, 26, 39, 68], due to their established performance towards various real-world tasks [15, 69, 53], as well as close connections to spectral graph theory [12, 9, 16]. While most GNN architectures are not very complicated, the training of GNNs can still be costly regarding both memory and computation resources on real-world large-scale graphs [10, 63]. Moreover, it is intriguing to transfer learned structural information across different graphs and even domains in settings like few-shot learning [56, 44, 25]. Therefore, several very recent studies have been conducted on the transferability of GNNs [21, 23, 22, 59, 31, 3, 47]. However, it is unclear in what situations the models will excel or fail especially when the pre-training and fine-tuning tasks are different. To provide rigorous analysis and guarantee on the transferability of GNNs, we focus on the setting of direct-transfering between the source and target graphs, under an analogous setting of "domain adaptation" [7, 59].

In this work, we establish a theoretically grounded framework for the transfer learning of GNNs, and leverage it to design a practically transferable GNN model. Figure 1 gives an overview of our framework. It is based on a novel view of a graph as samples from the joint distribution of its k-hop ego-graph structures and node features, which allows us to define graph information and similarity,

*These two authors contribute equally.
[1]Code and processed data are available at https://github.com/GentleZhu/EGI.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online.

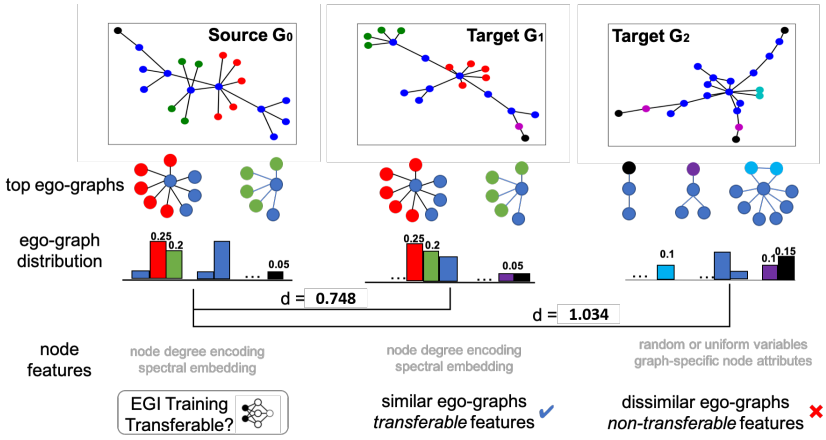


Figure 1: Overview of our GNN transfer learning framework: (1) we represent the toy graph as a combination of its 1-hop ego-graph and node feature distributions; (2) we design a transferable GNN regarding the capturing of such essential graph information; (3) we establish a rigorous guarantee of GNN transferability based on the node feature requirement and graph structure difference.

so as to analyze GNN transferability (§3). This view motivates us to design EGI, a novel GNN training objective based on ego-graph information maximization, which is effective in capturing the graph information as we define (§3.1). Then we further specify the requirement on transferable node features and analyze the transferability of EGI that is dependent on the local graph Laplacians of source and target graphs (§3.2).

All of our theoretical conclusions have been directly validated through controlled synthetic experiments (Table 1), where we use structural-equivalent role identification in an direct-transfering setting to analyze the impacts of different model designs, node features and source-target structure similarities on GNN transferability. In §4, we conduct real-world experiments on multiple publicly available network datasets. On the Airport and Gene graphs (§4.1), we closely follow the settings of our synthetic experiments and observe consistent but more detailed results supporting the design of EGI and the utility of our theoretical analysis. On the YAGO graphs (§4.2), we further evaluate EGI on the more generalized and practical setting of transfer learning with task-specific fine-tuning. We find our theoretical insights still indicative in such scenarios, where EGI consistently outperforms state-of-the-art GNN representation and transfer learning frameworks with significant margins.

2 Related Work

Representation learning on graphs has been studied for decades, with earlier spectral-based methods [6, 46, 52] theoretically grounded but hardly scaling up to graphs with over a thousand of nodes. With the emergence of neural networks, unsupervised network embedding methods based on the Skip-gram objective [37] have replenished the field [51, 14, 42, 45, 66, 62, 65]. Equipped with efficient structural sampling (random walk, neighborhood, *etc.*) and negative sampling schemes, these methods are easily parallelizable and scalable to graphs with thousands to millions of nodes. However, these models are essentially transductive as they compute fully parameterized embeddings only for nodes seen during training, which are impossible to be transfered to unseen graphs.

More recently, researchers introduce the family of graph neural networks (GNNs) that are capable of inductive learning and generalizing to unseen nodes given meaningful node features [29, 12, 15, 67]. Yet, most existing GNNs require task-specific labels for training in a semi-supervised fashion to achieve satisfactory performance [29, 15, 53, 64], and their usage is limited to single graphs where the downstream task is fixed. To this end, several unsupervised GNNs are presented, such as the auto-encoder-based ones like VGAE [28] and GNFs [35], as well as the deep-infomax-based ones like DGI [54] and InfoGraph [50]. Their potential in the transfer learning of GNN remains unclear when the node features and link structures vary across different graphs.

Although the architectures of popular GNNs such as GCN [29] may not be very complicated compared with heavy vision and language models, training a dedicated GNN for each graph can still

be cumbersome [10, 63]. Moreover, as pre-training neural networks are proven to be successful in other domains [13, 18], the idea is intriguing to transfer well-trained GNNs from relevant source graphs to improve the modeling of target graphs or enable few-shot learning [59, 31, 3] when labeled data are scarce. In light of this, pioneering works have studied both generative [22] and discriminative [21, 23] GNN pre-training schemes. Though Graph Contrastive Coding [43] shares the most similar view towards graph structures as us, it utilizes contrastive learning across all graphs instead of focusing on the transfer learning between any specific pairs. On the other hand, unsupervised domain adaptive GCNs [59] study the domain adaption problem only when the source and target tasks are homogeneous.

Most previous pre-training and self-supervised GNNs lack a rigorous analysis towards their transferability and thus have unpredictable effectiveness. The only existing theoretical work on GNN transferability studies the performance of GNNs across different permutations of a single original graph [33, 34] and the tradeoff between discriminability and transferability of GNNs [47]. We, instead, are the first to rigorously study the more practical setting of transferring GNNs across pairs of different source and target graphs.

3 Transferable Graph Neural Networks

In this paper, we design a more transferable training objective for GNN (EGI) based on our novel view of essential graph information (§3.1). We then analyze its transferability as the gap between its abilities to model the source and target graphs, based on their local graph Laplacians (§3.2).

Based on the connection between GNN and spectral graph theory [29], we describe the output of a GNN as a combination of its input node features X, fixed graph Laplacian L and learnable graph filters Ψ. The goal of training a GNN is then to improve its utility by learning the graph filters that are compatible with the other two components towards specific tasks.

In the graph transfer learning setting where downstream tasks are often unknown during pre-training, we argue that the general utility of a GNN should be optimized and quantified *w.r.t.* its ability of capturing the essential graph information in terms of the joint distribution of its topology structures and node features, which motivates us to design a novel ego-graph information maximization model (EGI) (§3.1). The general transferability of a GNN is then quantified by the gap between its abilities to model the source and target graphs. Under reasonable requirements such as using *structure-respecting* node features as the GNN input, we analyze this gap for EGI based on the structural difference between two graphs *w.r.t.* their local graph Laplacians (§3.2).

3.1 Transferable GNN via Ego-graph Information Maximization

In this work, we focus on the *direct-transfering setting* where a GNN is pre-trained on a source graph G_a in an unsupervised fashion and applied on a target graph G_b without fine-tuning.[2] Consider a graph $G = \{V, E\}$, where the set of nodes V are associated with certain features X and the set of edges E form graph structures. Intuitively, the transfer learning will be successful only if both the features and structures of G_a and G_b are similar in some ways, so that the graph filters of a GNN learned on G_a are compatible with the features and structures of G_b.

Graph kernels [57, 8, 30, 38] are well-known for their capability of measuring similarity between pair of graphs. Motivated by k-hop subgraph kernels [4], we introduce a novel view of a graph as *samples from the joint distribution of its k-hop ego-graph structures and node features*. Since GNN essentially encodes such k-hop ego graph samples, this view allows us to give concrete definitions towards *structural information* of graphs in the transfer learning setting, which facilitates the measuring of similarity (difference) among graphs. Yet, none of the existing GNN training objectives are capable of recovering such distributional signals of ego graphs. To this end, we design *Ego-Graph Information maximization* (EGI), which alternatively reconstructs the k-hop ego-graph of each center node via mutual information maximization [20].

Definition 3.1 (K-hop ego-graph)**.** *We call a graph $g_i = \{V(g_i), E(g_i)\}$ a k-hop ego-graph centered at node v_i if it has a k-layer centroid expansion [4] such that the greatest distance between v_i and*

[2] In the experiments, we show our model to be generalizable to the more practical settings with task-specific pre-training and fine-tuning, while the study of rigorous bound in such scenarios is left as future work.

any other nodes in the ego-graph is k, i.e. $\forall v_j \in V(g_i), |d(v_i, v_j)| \leq k$, *where* $d(v_i, v_j)$ *is the graph distance between* v_i *and* v_j.

In this paper, we use directed k-hop ego-graph and its direction is decided by whether it is composed of incoming or outgoing edges to the center node, *i.e.*, g_i and $\tilde{g}_i$. The results apply trivially to undirected graphs with $g_i = \tilde{g}_i$.

Definition 3.2 (Structural information)**.** *Let* $\mathcal{G}$ *be a topological space of sub-graphs, we view a graph* G *as samples of k-hop ego-graphs* $\{g_i\}_{i=1}^n$ *drawn* i.i.d. *from* $\mathcal{G}$ *with probability* μ*,* i.e.*,* $g_i \overset{\text{i.i.d.}}{\sim} \mu \ \forall i = 1, \cdots, n$*. The structural information of* G *is then defined to be the set of k-hop ego-graphs of* $\{g_i\}_{i=1}^n$ *and their empirical distribution.*

As shown in Figure 1, three graphs G_0, G_1 and G_2 are characterized by a set of 1-hop ego-graphs and their empirical distributions, which allows us to quantify the structural similarity among graphs as shown in §3.2 (*i.e.*, G_0 is more similar to G_1 than G_2 under such characterization). In practice, the nodes in a graph G are characterized not only by their k-hop ego-graph structures but also their associated node features. Therefore, G should be regarded as samples $\{(g_i, x_i)\}$ drawn from the joint distribution $\mathbb{P}$ on the product space of $\mathcal{G}$ and a node feature space $\mathcal{X}$.

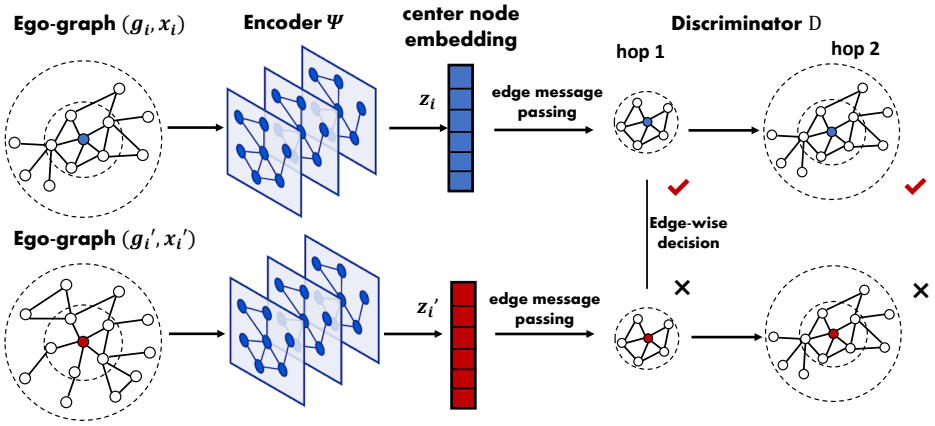


Figure 2: The overall EGI training framework.

Ego-Graph Information Maximization. Given a set of ego-graphs $\{(g_i, x_i)\}_i$ drawn from an empirical joint distribution $(g_i, x_i) \sim \mathbb{P}$. We aim to train an GNN encoder Ψ to maximize the mutual informaion (MI $(g_i, \Psi(g_i, x_i))$) between the defined structural information g_i[3] (*i.e.* k-hop ego-graph) and node embedding $z_i = \Psi(g_i, x_i)$. To maximize the MI, another discriminator $\mathcal{D}(g_i, z_i) : E(g_i) \times z_i \to \mathbb{R}^+$ is introduced to compute the probability of an edge e belongs to the given ego-graph g_i. We use the Jensen-Shannon MI estimator [20] in the EGI objective,

$$\mathcal{L}_{\text{EGI}} = -\text{MI}^{(\text{JSD})}(\mathcal{G}, \Psi) = \frac{1}{N} \sum_{i=1}^{N} \left[\text{sp}\left(\mathcal{D}(g_i, z_i')\right) + \text{sp}\left(-\mathcal{D}(g_i, z_i)\right) \right], \tag{1}$$

where $\text{sp}(x) = \log(1 + e^x)$ is the softplus function and (g_i, z_i') is randomly drawn from the product of marginal distributions, *i.e.* $z_i' = \Psi(g_{i'}, x_{i'}), (g_{i'}, x_{i'}) \sim \mathbb{P}, i' \neq i$. In general, we can also randomly draw negative g_i' in the topological space, while enumerating all possible graphs $g_{i'}$ leads to high computation cost.

In Eq. 1, the computation of $\mathcal{D}$ on $E(g_i)$ depends on the node orders. Following the common practice in graph generation [70], we characterize the decision process of $\mathcal{D}$ with a fixed graph ordering, *i.e.*, the BFS-ordering π over edges $E(g_i)$. $\mathcal{D} = f \circ \Phi$ is composed by another GNN encoder Φ and scoring function f over an edge sequence $E^\pi : \{e_1, e_2, ..., e_n\}$, which makes predictions on the BFS-ordered edges.

[3]Later in section 3.2, we will discuss the equivalence between MI(g_i, z_i) and MI$((g_i, x_i), z_i)$ when node feature is structure-respecting.

Recall our previous definition on the direction of k-hop ego-graph, the center node encoder Ψ receives pairs of (g_i, x_i) while the neighbor node encoder Φ in discriminator $\mathcal{D}$ receives $(\tilde{g}_i, x_i)$. Both encoders are parameterized as GNNs,

$$\Psi(g_i, x_i) = \text{GNN}_\Psi(A_i, X_i), \Phi(\tilde{g}_i, x_i) = \text{GNN}_\Phi(A'_i, X_i),$$

where A_i, A'_i is the adjacency matrix with self-loops of g_i and $\tilde{g}_i$, respectively. The self-loops are added following the common design of GNNs, which allows the convolutional node embeddings to always incorporate the influence of the center node. $A_i = A'^{\intercal}_i$. The output of Ψ, *i.e.*, $z_i \in \mathbb{R}^n$, is the center node embedding, while Φ outputs representation $H \in \mathbb{R}^{|g_i| \times n}$ for neighbor nodes in the ego-graph.

Once node representation H is computed, we now describe the scoring function f. For each of the node pair $(p, q) \in E^\pi$, h_p is the source node representation from Φ, x_q is the destination node features. The scoring function is,

$$f(h_p, x_q, z_i) = \sigma\left(U^T \cdot \tau\left(W^T[h_p || x_q || z_i]\right)\right), \tag{2}$$

where σ and τ are Sigmoid and ReLU activation functions. Thus, the discriminator $\mathcal{D}$ is asked to distinguish a positive $((p, q), z_i)$ and negative pair $((p, q), z'_i))$ for each edge in g_i.

$$\mathcal{D}(g_i, z_i) = \sum_{(p,q) \in E^\pi} \log f(h_p, x_q, z_i), \;\; \mathcal{D}(g_i, z'_i) = \sum_{(p,q)}^{E^\pi} \log f(h_p, x_q, z'_i). \tag{3}$$

There are two types of edges (p, q) in our consideration of node orders, *type-a* - the edges across different hops (from the center node), and *type-b* - the edges within the same hop (from the center node). The aforementioned BFS-based node ordering guarantees that Eq. 3 is sensitive to the ordering of type-a edges, and invariant to the ordering of type-b edges, which is consistent with the requirement of our theoretical analysis on $\Delta_\mathcal{D}$. Due to the fact that the output of a k-layer GNN only depends on a k-hop ego-graph for both encoders Ψ and Φ, EGI can be trained in parallel by sampling batches of g_i's. Besides, the training objective of EGI is transferable as long as (g_i, x_i) across source graph G_a and G_b satisfies the conditions given in §3.2. More model details in Appendix §B and source code in the Supplementary Materials.

Connection with existing work. To provide more insights into the EGI objective, we also present it as a dual problem of ego-graph reconstruction. Recall our definition of ego-graph mutual information $\text{MI}(g_i, \Psi(g_i, x_i))$. It can be related to an ego-graph reconstruction loss $R(g_i|\Psi(g_i, x_i))$ as

$$\max \text{MI}(g_i, \Psi(g_i, x_i)) = H(g_i) - H(g_i|\Psi(g_i, x_i)) \leq H(g_i) - R(g_i|\Psi(g_i, x_i)). \tag{4}$$

When EGI is maximizing the mutual information, it simultaneously minimizes the upper error bound of reconstructing an ego-graph g_i. In this view, the key difference between EGI and VGAE [28] is they assume each edge in a graph to be observed independently during the reconstruction. While in EGI, edges in an ego-graph are observed jointly during the GNN decoding. Moreover, existing mutual information based GNNs such as DGI [54] and GMI [41] explicitly measure the mutual information between node features x and GNN output Ψ. In this way, they tend to capture node features instead of graph structures, which we deem more essential in graph transfer learning as discussed in §3.2.

Use cases of EGI framework. In this paper, we focus on the classical domain adaption (direct-transferring) setting [7], where no target domain labels are available and transferability is measured by the performance discrepancy without fine-tuning. In this setting, the transferability of EGI is theoretically guaranteed by Theorem 3.1. In §4.1, we validated this with the airport datasets. Beyond direct-transferring, EGI is also useful in the more generalized and practical setting of transfer learning with fine-tuning, which we introduced in §4.2 and validated with the YAGO datasets. In this setting, the transferability of EGI is not rigorously studied yet, but is empirically shown promising.

Supportive observations. In the first three columns of our synthetic experimental results (Table 1), in both cases of transfering GNNs between similar graphs (F-F) and dissimilar graphs (B-F), EGI significantly outperforms all competitors when using node degree one-hot encoding as transferable node features. In particular, the performance gains over the untrained GIN show the effectiveness of training and transfering, and our gains are always larger than the two state-of-the-art unsupervised GNNs. Such results clearly indicate advantageous structure preserving capability and transferability of EGI.

3.2 Transferability analysis based on local graph Laplacians

We now study the transferability of a GNN (in particular, with the training objective of $\mathcal{L}_{\text{EGI}}$) between the source graph G_a and target graph G_b based on their graph similarity. We firstly establish the requirement towards node features, under which we then focus on analyzing the transferability of EGI *w.r.t.* the structural information of G_a and G_b.

Recall our view of the GNN output as a combination of its input node features, fixed graph Laplacian and learnable graph filters. The utility of a GNN is determined by the compatibility among the three. In order to fulfill such compatibility, we require the node features to be *structure-respecting*:

Definition 3.3 (Structure-respecting node features)**.** *Let g_i be an ordered ego-graph centered on node v_i with a set of node features $\{x^i_{p,q}\}^{k,|V_p(g_i)|}_{p=0,q=1}$, where $V_p(g_i)$ is the set of nodes in p-th hop of g_i. Then we say the node features on g_i are structure-respecting if $x^i_{p,q} = [f(g_i)]_{p,q} \in \mathbb{R}^d$ for any node $v_q \in V_p(g_i)$, where $f : \mathcal{G} \to \mathbb{R}^{d\times|V(g_i)|}$ is a function. In the strict case, f should be injective.*

In its essence, Def 3.3 requires the node features to be a function of the graph structures, which is sensitive to changes in the graph structures, and in an ideal case, injective to the graph structures (*i.e.*, mapping different graphs to different features). In this way, when the learned graph filters of a transfered GNN is compatible to the structure of G, they are also compatible to the node features of G. As we will explain in Remark 2 of Theorem 3.1, this requirement is also essential for the analysis of EGI transferability which eventually only depends on the structural difference between two graphs.

In practice, commonly used node features like node degrees, PageRank scores [40], spectral embeddings [11], and many pre-computed unsupervised network embeddings [42, 51, 14] are all structure-respecting in nature. However, other commonly used node features like random vectors [68] or uniform vectors [60] are not and thus non-transferable. When raw node attributes are available, they are transferable as long as the concept of *homophily* [36] applies, which also implies Def 3.3, but we do not have a rigorous analysis on it yet.

Supportive observations. In the fifth and sixth columns in Table 1, where we use same fixed vectors as non-transferable node features to contrast with the first three columns, there is almost no transferability (see $\delta(acc.)$) for all compared methods when non-transferable features are used, as the performance of trained GNNs are similar to or worse than their untrained baselines. More detailed experiments on different transferable and non-transferable features can be found in Appendix §C.1.

With our view of graphs and requirement on node features both established, now we derive the following theorem by characterizing the performance difference of EGI on two graphs based on Eq. 1.

Theorem 3.1 (GNN transferability)**.** *Let $G_a = \{(g_i, x_i)\}^n_{i=1}$ and $G_b = \{(g_{i'}, x_{i'})\}^m_{i'=1}$ be two graphs, and assume node features are structure-relevant. Consider GCN Ψ_θ with k layers and a 1-hop polynomial filter ϕ. With reasonable assumptions on the local spectrum of G_a and G_b, the empirical performance difference of Ψ_θ evaluated on $\mathcal{L}_{\text{EGI}}$ satisfies*

$$|\mathcal{L}_{\text{EGI}}(G_a) - \mathcal{L}_{\text{EGI}}(G_b)| \leq \mathcal{O}\left(\Delta_{\mathcal{D}}(G_a, G_b) + C\right). \tag{5}$$

On the RHS, C is only dependent on the graph encoders and node features, while $\Delta_{\mathcal{D}}(G_a, G_b)$ measures the structural difference between the source and target graphs as follows,

$$\Delta_{\mathcal{D}}(G_a, G_b) = \tilde{C}\frac{1}{nm}\sum_{i=1}^{n}\sum_{i'=1}^{m}\lambda_{\max}(\tilde{L}_{g_i} - \tilde{L}_{g_{i'}}) \tag{6}$$

where $\lambda_{\max}(A) := \lambda_{\max}(A^TA)^{1/2}$, and $\tilde{L}_{g_i}$ denotes the normalised graph Laplacian of $\tilde{g_i}$ by its in-degree. $\tilde{C}$ is a constant dependant on $\lambda_{\max}(\tilde{L}_{g_i})$ and $\mathcal{D}$.

Proof. The full proof is detailed in Appendix §A. □

The analysis in Theorem 3.1 naturally instantiates our insight about the correspondence between structural similarity and GNN transferability. It allows us to tell how well an EGI trained on G_a can work on G_b by only checking the local graph Laplacians of G_a and G_b without actually training any model. In particular, we define the *EGI gap* as $\Delta_{\mathcal{D}}$ in Eq. 6, as other term C is the same for different methods using same GNN encoder. It can be computed to bound the transferability of EGI regarding its loss difference on the source and target graphs.

Remark 1. *Our view of a graph G as samples of k-hop ego-graphs is important, as it allows us to obtain node-wise characterization of GNN similarly as in [55]. It also allows us to set the depth of ego-graphs in the analysis to be the same as the number of GNN layers (k), since the GNN embedding of each node mostly depends on its k-hop ego-graph instead of the whole graph.*

Remark 2. *For Eq. 1, Def 3.3 ensures the sampling of GNN embedding at a node always corresponds to sampling an ego-graph from $\mathcal{G}$, which reduces to uniformly sampling from $G = \{g_i\}_{i=1}^n$ under the setting of Theorem 3.1. Therefore, the requirement of Def 3.3 in the context of Theorem 3.1 guarantees the analysis to be only depending on the structural information of the graph.*

Supportive observations. In Table 1, in the $\bar{d}$ columns, we compute the average structural difference between two Forest-fire graphs ($\Delta_{\mathcal{D}}$(F,F)) and between Barabasi and Forest-fire graphs ($\Delta_{\mathcal{D}}$(B,F)), based on the RHS of Eq. 5. The results validate the topological difference between graphs generated by different random-graph models, while also verifying our view of graph as k-hop ego-graph samples and the way we propose based on it to characterize structural information of graphs. We further highlight in the δ(acc) columns the accuracy difference between the GNNs transfered from Forest-fire graphs and Barabasi graphs to Forest-fire graphs. Since Forest-fire graphs are more similar to Forest-fire graphs than Barabasi graphs (as verified in the $\Delta_{\mathcal{D}}$ columns), we expect δ(acc.) to be positive and large, indicating more positive transfer between the more similar graphs. Indeed, the behaviors of EGI align well with the expectation, which indicates its well-understood transferability and the utility of our theoretical analysis.

Use cases of Theorem 3.1. Our Theorem 3.1 naturally allows for two practical use cases among many others: *point-wise pre-judge* and *pair-wise pre-selection* for EGI pre-training. Suppose we have a target graph G_b which does not have sufficient training labels. In the first setting, we have a single source graph G_a which might be useful for pre-training a GNN to be used on G_b. The EGI gap $\Delta_{\mathcal{D}}(G_a, G_b)$ in Eq. 6 can then be computed between G_a and G_b to pre-judge whether such transfer learning would be successful before any actual GNN training (*i.e.*, yes if $\Delta_{\mathcal{D}}(G_a, G_b)$ is empirically much smaller than 1.0; no otherwise). In the second setting, we have two or more source graphs $\{G_a^1, G_a^2, \ldots\}$ which might be useful for pre-training the GNN. The EGI gap can then be computed between every pair of G_a^i and G_b to pre-select the best source graph (*i.e.*, select the one with the least EGI gap).

In practice, the computation of eigenvalues on the small ego-graphs can be rather efficient [2], and we do not need to enumerate all pairs of ego-graphs on two compared graphs especially if the graphs are really large (*e.g.*, with more than a thousand nodes). Instead, we can randomly sample pairs of ego-graphs from the two graphs, update the average difference on-the-fly, and stop when it converges. Suppose we need to sample M pairs of k-hop ego-graphs to compare two large graphs, and the average size of ego-graphs are L, then the overall complexity of computing Eq. 5 is $\mathcal{O}(ML^2)$, where M is often less than 1K and L less than 50. In Appendix §C.4, we report the approximated $\Delta_{\mathcal{D}}$'s *w.r.t.* different sampling frequencies, and they are indeed pretty close to the actual value even with smaller sample frequencies, showing the feasible efficiency of computing $\Delta_{\mathcal{D}}$ through sampling.

Limitations. EGI is designed to account for the structural difference captured by GNNs (*i.e.*, k-hop ego-graphs). The effectiveness of EGI could be limited if the tasks on target graphs depend on different structural signals. For example, as Eq. 6 is computing the average pairwise distances between the graph Laplacians of local ego-graphs, $\Delta_{\mathcal{D}}$ is possibly less effective in explicitly capturing global graph properties such as numbers of connected components (CCs). In some specific tasks (such as counting CCs or community detection) where such properties become the key factors, $\Delta_{\mathcal{D}}$ may fail to predict the transferability of GNNs.

4 Real Data Experiments

Baselines. We compare the proposed model against existing self-supervised GNNs and pre-training GNN algorithms. To exclude the impact of different GNN encoders Ψ on transferability, we always use the same encoder architecture for all compared methods (*i.e.*, GIN [60] for direct-transfering experiments, GCN [29] for transfering with fine-tuning).

The self-supervised GNN baselines are GVAE [28], DGI [54] and two latest mutual information estimation methods GMI [41] and MVC [17]. As for pre-training GNN algorithms, MaskGNN

Table 1: Synthetic experiments of identifying structural equivalent nodes. We randomly generate 40 graphs with the Forest-fire model (F) [32] and 40 graphs with the Barabasi model (B) [1], The GNN model is GIN [60] with random parameters (baseline with only the neighborhood aggregation function), VGAE[28], DGI [54], and EGI with GIN encoder. We train VGAE, DGI and EGI on one graph from either set (F and B), and test them on the rest of Forest-fire graphs (F). Transferable feature is node degree one-hot encoding and non-transferable feature is uniform vectors. More details about the results and dataset can be found in Appendix §C.1

Method	**transferable features**			**non-transferable feature**			**structural difference**	
	F-F	B-F	δ(acc.)	F-F	B-F	δ(acc.)	$\Delta_{\mathcal{D}}$(F,F)	$\Delta_{\mathcal{D}}$(B,F)
GIN (untrained)	0.572	0.572	/	0.358	0.358	/		
VGAE (GIN)	0.498	0.432	+0.066	0.240	0.239	0.001		
DGI (GIN)	0.578	0.591	-0.013	0.394	0.213	+0.181	0.752	0.883
EGI (GIN)	**0.710**	0.616	+0.094	0.376	0.346	+0.03		

and ContextPredGNN are two node-level pre-training models proposed in [21] Besides, Structural Pre-train [23] also conducts unsupervised node-level pre-training with structural features like node degrees and clustering coefficients.

Experimental Settings. The main hyperparameter k is set 2 in EGI as a common practice. We use Adam [27] as optimizer and learning rate is 0.01. We provide the experimental result with varying k in the Appendix §C.4. All baselines are set with the default parameters. Our experiments were run on an AWS g4dn.2xlarge machine with 1 Nvidia T4 GPU. By default, we use node degree one-hot encoding as the transferable feature across all different graphs. As stated before, other transferable features like spectral and other pre-computed node embeddings are also applicable. We focus on the setting where the downstream tasks on target graphs are unspecified but assumed to be structure-relevant, and thus pre-train the GNNs on source graphs in an unsupervised fashion.[4] In terms of evaluation, we design two realistic experimental settings: (1) Direct-transfering on the more structure-relevant task of role identification without given node features to directly evaluate the utility and transferability of EGI. (2) Few-shot learning on relation prediction with task-specific node features to evaluate the generalization ability of EGI.

4.1 Direct-transfering on role identification

First, we use the role identification without node features in a *direct-transfering* setting as a reliable proxy to evaluate transfer learning performance regarding different pre-training objectives. Role in a network is defined as nodes with similar structural behaviors, such as *clique members*, *hub* and *bridge* [19]. Across graphs in the same domain, we assume the definition of role to be consistent, and the task of role identification is highly structure-relevant, which can directly reflect the transferability of different methods and allows us to conduct the analysis according to Theorem 3.1. Upon convergence of pre-training each model on the source graphs, we directly apply them to the target graphs and further train a multi-layer perceptron (MLP) upon their outputs. The GNN parameters are frozen during the MLP training. We refer to this strategy as *direct-transfering* since there is no fine-tuning of the models after transfering to the target graphs.

We use two real-world network datasets with role-based node labels: (1) Airport [45] contains three networks from different regions– Brazil, USA and Europe. Each node is an airport and each link is the flight between airports. The airports are assigned with external labels based on their *level of popularity*. (2) Gene [68] contains the gene interactions regarding 50 different cancers. Each gene has a binary label indicating whether it is a *transcription factor*. More details about the results and dataset can be found in Appendix C.2.

The experimental setup on the Airport dataset closely resembles that of our synthetic experiments in Table 1, but with real data and more detailed comparisons. We train all models (except for the untrained ones) on the Europe network, and test them on all three networks. The results are presented in Table 2. We notice that the node degree features themselves (with MLP) show reasonable performance in all three networks, which is not surprising since the popularity-based airport role labels are highly relevant to node degrees. The untrained GIN encoder yields a significant margin over just node features, as GNN encoder incorporates structural information to node representations.

[4]The downstream tasks are unspecified because we aim to study the general transferability of GNNs that is not bounded to specific tasks. Nevertheless, we assume the tasks to be relevant to graph structures.

While training of the DGI can further improve the performance on the source graph, EGI shows the best performance there with the structure-relevant node degree features, corroborating the claimed effectiveness of EGI in capturing the essential graph information (*i.e.* recover the k-hop ego-graph distributions) as we stress in §3.

When transfering the models to USA and Brazil networks, EGI further achieves the best performance compared with all baselines when structure relevant features are used (64.55 and 73.15), which reflects the most significant positive transfer. Interestingly, direct application of GVAE, DGI and MVC that do not capture the input k-hop graph jointly, leads to rather limited and even negative transferrability (through comparison against the untrained GIN encoders). The recently proposed transfer learning frameworks for GNN like MaskGNN and Structural Pre-train are able to mitigate negative transfer to some extent, but their performances are still inferior to EGI. We believe this is because their models are prone to learn the graph-specific information that is less transferable across different graphs. GMI is also known to capture the graph structure and node features, so it achieves second best result comparing with EGI.

Similarly as in Table 1, we also compute the structural differences among three networks *w.r.t.* the EGI gap in Eq. 6. The structural difference is 0.869 between the Europe and USA networks, and 0.851 between the Europe and Brazil datasets, which are pretty close. Consequently, the transferability of EGI regarding its performance gain over the untrained GIN baseline is 4.8% on the USA network and 4.4% on the Brazil network, which are also close. Such observations again align well with our conclusion in Theorem 3.1 that the transferability of EGI is closely related to the structural differences between source and target graphs.

Table 2: Results of role identification with direct-transfering on the Airport dataset. We report mean and standard deviation over 100 runs. The scores marked with ** passed t-test with $p < 0.01$ over the second runners.

Method	**Airport [45]**		
	Europe	USA	Brazil
features	0.528±0.052	0.557±0.028	0.671±0.089
GIN (random-init)	0.558±0.050	0.616±0.030	0.700±0.082
GVAE (GIN) [28]	0.539±0.053	0.555±0.029	0.663±0.089
DGI (GIN) [54]	0.578±0.050	0.549±0.028	0.673±0.084
Mask-GIN [21]	0.564±0.053	0.608±0.027	0.667±0.073
ContextPred-GIN [21]	0.527±0.048	0.504±0.030	0.621±0.078
Structural Pre-train [23]	0.560±0.050	0.622±0.030	0.688±0.082
MVC [17]	0.532±0.050	0.597±0.030	0.661±0.093
GMI [41]	0.581±0.054	0.593±0.031	0.731±0.107
EGI (GIN)	**0.592±0.046****	**0.646±0.029** **	**0.732±0.078**

On the Gene dataset, with more graphs available, we focus on EGI to further validate the utility of Eq. 5 in Theorem 3.1, regarding the connection between the EGI gap (Eq. 6) and the performance gap (micro-F1) of EGI on them. Due to severe label imbalance that removes the performance gaps, we only use the seven brain cancer networks that have a more consistent balance of labels. As shown in Figure 3, we train EGI on one graph and test it on the other graphs. The x-axis shows the EGI gap, and y-axis shows the improvement on micro-F1 compared with an untrained GIN. The negative correlation between two quantities is obvious. Specifically, when the structural difference is smaller than 1, positive transfer is observed (upper left area) as the performance of transferred EGI is better than untrained GIN, and when the structural difference becomes large (> 1), negative transfer is observed. We also notice a similar graph pattern, *i.e.* single dense cluster, between source graph and positive transferred target graph G_2.

4.2 Few-shot learning on relation prediction

Here we evaluate EGI in the more generalized and practical setting of *few-shot learning* on the less structure-relevant task of relation prediction, with task-specific node features and fine-tuning. The source graph contains a cleaned full dump of 579K entities from YAGO [49], and we investigate 20-shot relation prediction on a target graph with 24 relation types, which is a sub-graph of 115K entities sampled from the same dump. In *post-fine-tuning*, the models are pre-trained with an unsupervised loss on the source graph and fine-tuned with the task-specific loss on the target graph. In *joint-fine-tuning*, the same pre-trained models are jointly optimized *w.r.t.* the unsupervised pre-training loss

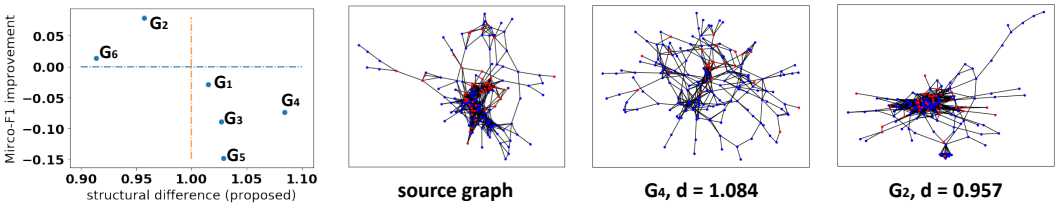


Figure 3: Transfer learning performance of role identification on the Gene dataset. We visualize the source graph G_0 and two example target graphs that are relatively more different (G_4) or similar (G_2) with G_0.

and task-specific fine-tuning loss on the target graph. In Table 3, we observe most of the existing models fail to transfer across pre-training and fine-tuning tasks, especially in the *joint-fine-tuning* setting. In particular, both Mask-GIN and ContextPred-GIN rely a lot on task-specific fine-tuning, while EGI focuses on the capturing of similar ego-graph structures that are transferable across graphs. The mutual information based method GMI also demonstrates considerable transferability and we believe the ability to capture the graph structure is the key to the transferability. As a consequence, EGI significantly outperforms all compared methods in both settings. More detailed statistics and running time are in Appendix §C.3.

Table 3: Performance of few-shot relation prediction on YAGO. The scores marked with ** passed t-test with $p < 0.01$ over the second best results.

Method	post-fine-tuning		joint-fine-tuning	
	AUROC	MRR	AUROC	MRR
No pre-train	0.687±0.002	0.596±0.003	N.A.	N.A.
GVAE	0.701±0.003	0.601±0.007	0.679±0.004	0.568±0.008
DGI	0.689±0.011	0.586±0.025	0.688±0.012	0.537±0.023
MaskGNN	0.713±0.009	0.631±0.015	0.712±0.005	0.560±0.010
ContextPredGNN	0.692±0.030	0.662±0.030	0.705±0.011	0.575±0.021
GMI	0.728±0.005	0.625±0.009	0.721±0.007	0.643±0.011
Structural Pre-train	OOM	OOM	OOM	OOM
MVC	OOM	OOM	OOM	OOM
EGI	**0.739± 0.009****	**0.670±0.014**	**0.787 ± 0.011****	**0.729 ± 0.016****

5 Conclusion

To the best of our knowledge, this is the first research effort towards establishing a theoretically grounded framework to analyze GNN transferability, which we also demonstrate to be practically useful for guiding the design and conduct of transfer learning with GNNs. For future work, it is intriguing to further strengthen the bound with relaxed assumptions, rigorously extend it to the more complicated and less restricted settings regarding node features and downstream tasks, as well as analyze and improve the proposed framework over more transfer learning scenarios and datasets. It is also important to protect the privacy of pre-training data to avoid potential negative societal impacts.

Acknowledgments and Disclosure of Funding

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004, SocialSim Program No. W911NF-17-C-0099, and INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897. Chao Zhang is supported NSF IIS-2008334, IIS-2106961, and ONR MURI N00014-17-1-2656. We would like to thank AWS Machine Learning Research Awards program for providing computational resources for the experiments in this paper. This work is also partially supported by the internal funding and GPU servers provided by the Computer Science Department of Emory University. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government.

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *FOCS*, pages 339–348, 2005.

[3] Jinheon Baek, Dong Bok Lee, and Sung Ju Hwang. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. *Advances in Neural Information Processing Systems*, 33, 2020.

[4] Lu Bai and Edwin R Hancock. Fast depth-based subgraph kernels for unattributed graphs. *Pattern Recognition*, 50:233–245, 2016.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[6] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, pages 585–591, 2002.

[7] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NIPS*, pages 137–144, 2007.

[8] Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *arXiv preprint arXiv:2011.03854*, 2020.

[9] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[10] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

[11] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, pages 4171–4186, 2019.

[14] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[16] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *ACHA*, 30(2):129–150, 2011.

[17] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[19] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *KDD*, pages 1231–1239, 2012.

[20] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2019.

[22] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*, pages 1857–1867, 2020.

[23] Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. Pre-training graph neural networks for generic structural feature extraction. *arXiv preprint arXiv:1905.13728*, 2019.

[24] Suk-Geun Hwang. Cauchy's interlace theorem for eigenvalues of hermitian matrices. *The American Mathematical Monthly*, 111(2):157–159, 2004.

[25] Xuan Kan, Hejie Cui, and Carl Yang. Zero-shot scene graph relation prediction through commonsense knowledge integration. In *ECML-PKDD*, 2021.

[26] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *NIPS*, pages 7090–7099, 2019.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[30] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

[31] Lin Lan, Pinghui Wang, Xuefeng Du, Kaikai Song, Jing Tao, and Xiaohong Guan. Node classification on graphs with few-shot novel labels via meta transformed network embedding. *Advances in Neural Information Processing Systems*, 33, 2020.

[32] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

[33] Ron Levie, Wei Huang, Lorenzo Bucci, Michael M Bronstein, and Gitta Kutyniok. Transferability of spectral graph convolutional neural networks. *arXiv preprint arXiv:1907.12972*, 2019.

[34] Ron Levie, Elvin Isufi, and Gitta Kutyniok. On the transferability of spectral graph filters. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pages 1–5. IEEE, 2019.

[35] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. In *Advances in Neural Information Processing Systems*, pages 13556–13566, 2019.

[36] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[38] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *arXiv preprint arXiv:1904.12218*, 2019.

[39] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

[40] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[41] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *WWW*, pages 259–270, 2020.

[42] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[43] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*, pages 1150–1160, 2020.

[44] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[45] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *KDD*, pages 385–394, 2017.

[46] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[47] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[48] Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2190–2199, 2018.

[49] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[50] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2019.

[51] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[52] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[53] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[54] Petar Velickovic, William Fedus, William L Hamilton, Pietro Lio, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

[55] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *KDD*, 2019.

[56] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NIPS*, pages 3630–3638, 2016.

[57] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[58] Boris Weisfeiler and Andrei A Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[59] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *WWW*, pages 1457–1467, 2020.

[60] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[61] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[62] Carl Yang, Yichen Feng, Pan Li, Yu Shi, and Jiawei Han. Meta-graph based hin spectral embedding: Methods, analyses, and insights. In *ICDM*, 2018.

[63] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Chuck Rosenberg, and Jure Leskovec. Multisage: Empowering graphsage with contextualized multi-embedding on web-scale multipartite networks. In *KDD*, 2020.

[64] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: A unified framework with survey and benchmark. In *TKDE*, 2020.

[65] Carl Yang, Chao Zhang, Xuewen Chen, Jieping Ye, and Jiawei Han. Did you enjoy the ride? understanding passenger experience via heterogeneous network embedding. In *ICDE*, 2018.

[66] Carl Yang, Jieyu Zhang, and Jiawei Han. Co-embedding network nodes and hierarchical labels with taxonomy based generative adversarial nets. In *ICDM*, 2020.

[67] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. Relation learning on social networks with multi-modal graph edge variational autoencoders. In *WSDM*, 2020.

[68] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *NIPS*, pages 1338–1349, 2019.

[69] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NIPS*, 2018.

[70] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5708–5717. PMLR, 2018.

A Theory Details

From the $\mathcal{L}_{\text{EGI}}$ objective, we have assumed $g_i \overset{\text{i.i.d.}}{\sim} \mu$, $x_i \overset{\text{i.i.d.}}{\sim} \nu$, and $(g_i, x_i) \overset{\text{i.i.d.}}{\sim} p$, for $(g_i, x_i) \in \mathcal{G} \times \mathcal{X}$. Then for a sample $\{(g_i, x_i)\}_i$, we have access to the empirical distributions of the three. In the procedure of evaluating the objective, we sample uniformly.

Note that, in Eq. 2 of the main paper, we used a d dimensional hidden state h_p to denote an edge's source node representation and x_q as destination node features from the structure of the ego-graph and the associated source node feature with GNN. In our proof, we denote $v_{p,q}$ as the q-th node in the p-th layer of the ego-graph and let $h_{p,q} = h_p$ and $x_{p,q} = x_q$. For simplicity, in i-th layer, we denote $f(x^i) = h^i_{p,q} \| x^i_{p,q}$, where $[\cdot\|\cdot]$ is the concatenation operation.

Finally, as we are considering GNN with k layers, its computation only depends on the k-hop ego-graphs of G, which is an important consideration when unfolding the embedding of GNN at a centre node with Lamma A.1.

Lemma A.1. *For any $A \in \mathbb{R}^{m \times n}$, where $m \geq n$, and A is a submatrix of $B \in \mathbb{R}^{m' \times n}$, where $m < m'$, we have*

$$\|A\|_2 \leq \|B\|_2.$$

Proof. Note that, AA^T is a principle matrix of BB^T, *i.e.*, AA^T is obtained by removing the same set of rows and columns from BB^T. Then, by Eigenvalue Interlacing Theorem [24] and the fact that A^TA and AA^T have the same set of non-zero singular values, the matrix operator norm satisfies $\|A\|_2 = \sqrt{\lambda_{\max}(A^TA)} = \sqrt{\lambda_{\max}(AA^T)} \leq \sqrt{\lambda_{\max}(BB^T)} = \|B\|_2$. □

A.1 Center-node view of GCN

Recall that $V_p(g_i)$ denotes the set of nodes in the pth hop of k-hop ego-graph g_i, and $x^i_{p,q}$ denotes the feature for qth node in pth hop of g_i, for any $p = 0, \ldots, k;\ q = 1, \ldots, |V_p(g_i)|$. Similarly, $V(g_i)$ denotes the entire set of nodes in g_i. In each ego-graph sample $\{g_i, x_i\}$, the layer-wise propagation rules for the center node embedding in encoder Ψ and discriminator $\mathcal{D}$ can be written into the form of GCN as followed

$$Z^{(l)} = \text{ReLU}(D^{-\frac{1}{2}}(I + A)D^{-\frac{1}{2}}Z^{(l-1)}\theta^{(l)})$$

where A is adjacency matrix of G. I adds the self-loop and $D_{ii} = \sum_j A_{ij}$ is the degree matrix.

We focus on the center node's embedding obtained from a k-layer GCN with 1-hop polynomial filter $\phi(L) = Id - L$. Inspired by the characterization of GCN from a node-wise view in [55], we similarly denote the embedding of node $x_i\ \forall i = 1, \cdots, n$ in the final layer of the GCN as

$$z_i^{(k)} = z_i = \Psi_\theta(x_i) = \sigma\Big(\sum_{j \in \mathcal{N}(x_i)} e_{ij} {z_j^{(k-1)}}^T \theta^{(k)}\Big) \in \mathbb{R}^d,$$

where $e_{ij} = [\phi(L)]_{ij} \in \mathbb{R}$ the weighted link between node i and j; and $\theta^{(k)} \in \mathbb{R}^{d \times d}$ is the weight for the kth layer sharing across nodes. Then $\theta = \{\theta^{(\ell)}\}_{\ell=1}^k$. We may denote $z_i^{(\ell)} \in \mathbb{R}^d$ similarly for $\ell = 1, \cdots, k-1$, and $z_i^0 = x_i \in \mathbb{R}^d$ as the node feature of center node x_i. With the assumption of GCN in the statement, it is clear that only the k-hop ego-graph g_i centered at x_i is needed to compute $z_i^{(k)}$ for any $i = 1, \cdots, n$ instead of the whole of G. Precisely, p-hop of subgraph corresponds to the $\ell = (k - p)$th layer in the model.

With such observation in mind, let us denote the matrix of node embeddings of g_i at the ℓth layer as $[z_i^{(\ell)}] \in \mathbb{R}^{|V(g_i)| \times d}$ for $\ell = 1, \cdots, k$; and let $[z_i^{(0)}] \equiv [x_i] \in (\mathbb{R}^d)^{|V(g_i)|}$ denote the matrix of node features in the k-hop ego-graph g_i. In addition, denote $[z_i^{(\ell)}]_p$ as the principle submatrix, which includes embeddings for nodes in the 0 to pth hop of g_i, $0 \leq p \leq k$.

We denote L_{g_i} as the out-degree normalised graph Laplacian of g_i. Here, the out-degree is defined with respect to the direction from leaves to centre node in g_i. Similarly, denote $\tilde{L}_{g_i}$ as the in-degree normalised graph Laplacian of g_i, where the direction is from centre to leaves.

WLOG, we write the ℓth layer embedding in matrix notation of the following form

$$[z_i^{(\ell)}]_{k-\ell+1} = \sigma([\phi(L_{g_i})]_{k-\ell+1}[z_i^{(\ell-1)}]_{k-\ell+1}\theta^{(\ell)}),$$

where the GCN only updates the embedding of nodes in the 0 to $(k-\ell)$th hop. We also implicitly assume the embedding of nodes in $(k-\ell+1)$ to kth hop are unchanged through the update, due to the directed nature of g_i. Hence, we obtain $z_i \equiv [z_i^{(k)}]_0$ from the following

$$[z_i^{(k)}]_1 = \sigma([\phi(L_{g_i})]_1[z_i^{(k-1)}]_1\theta^{(k)}).$$

Similarly, we are able to write down the form of discriminator using matrix representation for GCN. The edge information at ℓth time point for nodes in $V(g_i)$ can be described as follows

$$[h_i^{(\ell)}] = ReLU(\phi(\tilde{L}_{g_i})[h_i^{(\ell-1)}]\tilde{\theta}^{(\ell)}),$$

A.2 Proof for Theorem 4.1

We restate Theorem 4.1 from the main paper as below.

Theorem A.2. *Let $G_a = \{(g_i, x_i)\}_{i=1}^n$ and $G_b = \{(g_{i'}, x_{i'})\}_{i'=1}^m$ be two graphs and node features are structure-respecting with $x_i = f(L_{g_i}), x_{i'} = f(L_{g_{i'}})$ for some function $f : \mathbb{R}^{|V(g_i)|\times|V(g_i)|} \to \mathbb{R}^d$. Consider GCN Ψ_θ with k layers and a 1-hop polynomial filter ϕ,the empirical performance difference of Ψ_θ with $\mathcal{L}_{\text{EGI}}$ satisfies*

$$|\mathcal{L}_{\text{EGI}}(G_a) - \mathcal{L}_{\text{EGI}}(G_b)| \leq \mathcal{O}\left(\frac{1}{nm}\sum_{i=1}^{n}\sum_{i'=1}^{m}[M + C\lambda_{\max}(L_{g_i} - L_{g_{i'}}) + \tilde{C}\lambda_{\max}(\tilde{L}_{g_i} - \tilde{L}_{g_{i'}}))]\right), \tag{7}$$

where M is dependant on Ψ, $\mathcal{D}$, node features, and the largest eigenvalue of L_{g_i} and $\tilde{L}_{g_i}$. C is a constant dependant on the encoder, while $\tilde{C}$ is a constant dependant on the decoder. With a slight abuse of notation, we denote $\lambda_{\max}(A) := \lambda_{\max}(A^TA)^{1/2}$. Note that, in the main paper, we have $C := M + C\lambda_{\max}(L_{g_i} - L_{g_{i'}})$, and $\Delta_{\mathcal{D}}(G_a, G_b) := \tilde{C}\lambda_{\max}(\tilde{L}_{g_i} - \tilde{L}_{g_{i'}})$.

Proof. Now,

$$\begin{aligned}
&|\mathcal{L}_{\text{EGI}}(G) - \mathcal{L}_{\text{EGI}}(G')| \\
=&\left|\frac{1}{n^2}\sum_{i,j=1}^{n}(\mathcal{D}(g_i, z_j)) - \frac{1}{n}\sum_{i=1}^{n}(-(-\mathcal{D}(g_i, z_i)) - (\frac{1}{m^2}\sum_{i',j'=1}^{m}(\mathcal{D}(g_{i'}, z_{j'})) - \frac{1}{m}\sum_{i'=1}^{m}(-(-\mathcal{D}(g_{i'}, z_{i'}))))\right| \\
\leq&\frac{1}{n^2m^2}\sum_{i,j=1}^{n}\sum_{i',j'=1}^{m}|\mathcal{D}(g_i, z_j) - \mathcal{D}(g_{i'}, z_{j'})| + \frac{1}{nm}\sum_{i=1}^{n}\sum_{i'=1}^{m}|\mathcal{D}(g_i, z_i) - \mathcal{D}(g_{i'}, z_{i'})| \\
=&\frac{1}{n^2m^2}\sum_{i,j=1}^{n}\sum_{i',j'=1}^{m}A + \frac{1}{nm}\sum_{i=1}^{n}\sum_{i'=1}^{m}B.
\end{aligned}$$

We make the following assumptions in the proof,

1. Assume the size of the neighborhood for each node is bounded by $0 < r < \infty$, then the maximum number of node for p-th layer subgraph is bounded by r^p. WLOG, let $1 \leq |V_p(g_i)| \leq |V_p(g_{i'})| \leq r^p$;
2. Assume $h_{p,q}^i \| x_{p,q}^i = 0$ if $|V_p(g_i)| < q$, i.e. assume non-informative edge information and node features for non-existed nodes in the smaller neighborhood with no links;

From Assumption 2, we add isolated nodes to the smaller neighborhood $V_p(g_i)$ such that the neighborhood size at each hop match. It can be found in our code to compute EGI gap as pad_nbhd. For the following proof, we WLOG assume $|V_p(g_i)| = |V_p(g_{i'})|\ \forall p$.

First we consider B. Recall that, $V_p(g_i)$ is the set of nodes in layer p of g_i,

$$\mathcal{D}(g_i, z_i) = \sum_{p=1}^{k}\sum_{q=1}^{|V_p(g_i)|}\log(\sigma_{sig}\left(U^T\tau\left(W^T[f(x^i)\|z_i]\right)\right)),$$

where $\sigma_{sig}(t) = \frac{1}{1+e^{-t}}$ is the sigmoid function, τ is some γ_τ-Lipschitz activation function and $[\cdot\|\cdot]$ denotes the concatenation of two vectors. Then we obtain

$$U^T\tau\left(W^T[f(x^i)\|z_i]\right) = U^T\tau\left(W_1^T f(x^i) + W_2^T z_i\right).$$

Since $\log(\sigma_{sig}(t)) = -\log(1+e^{-t})$, which is 1-Lipschitz, it gives

$$\begin{aligned}
B &\le \sum_p^k \Big| \sum_q^{|V_p(g_{i'})|} \sigma_s(U^T\tau\left(W_1^T f(x^i) + W_2^T z_i\right)) - \sigma_s(U^T\tau\left(W_1^T f(x^{i'}) + W_2^T z_{i'}\right))\Big| \\
&\le \gamma_\tau \|U\|_2 \sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} (\|W_1^T f(x^i) - W_1^T f(x^{i'})\|_2 + \|W_2^T z_i - W_2^T z_{i'}\|_2) \\
&\le \gamma_\tau \|U\|_2 s_w \left(\sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} \left[\|h_{p,q}^i - h_{p,q}^{i'}\|_2 + \|x_{p,q}^i - x_{p,q}^{i'}\|_2\right] + \sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} \|z_i - z_{i'}\|_2 \right) \\
&\le C_1 \left(\sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} \left[\|h_{p,q}^i - h_{p,q}^{i'}\|_2 + \|x_{p,q}^i - x_{p,q}^{i'}\|_2\right] / \sum_{p=1}^k r^p + \|z_i - z_{i'}\|_2 \right) \\
&= C_1 (I_1 + I_2)
\end{aligned} \tag{8}$$

We provide the derivation for the unfolding of ℓth layer GCN with the centre-node view in Lemma A.3. This will be used in the derivation of I_1 and I_2.

Lemma A.3. *For any $\ell = 1, \cdots, k$, we have an upper bound for the hidden representation difference between g_i and g_i',*

$$\begin{aligned}
\|[z_i^{(\ell)}]_{k-\ell} - [z_{i'}^{(\ell)}]_{k-\ell}\|_2 \le (\gamma_\sigma c_\theta)^\ell \|\phi(L_{g_i})\|_2^\ell \|[x_i] - [x_{i'}]\|_2 \\
+ \frac{(\gamma_\sigma c_\theta)^\ell \|\phi(L_{g_i})\|_2^\ell + 1}{\gamma_\sigma c_\theta \|\phi(L_{g_i})\|_2 - 1} \gamma_\sigma c_\theta c_z \|\phi(L_{g_i}) - \phi(L_{g_{i'}})\|_2.
\end{aligned} \tag{9}$$

Specifically, for $\ell = k$, we obtain the expansion for center node embedding $\|[z_i^{(k)}]_0 - [z_{i'}^{(k)}]_0\| \equiv \|z_i - z_{i'}\|$.

Proof. By Lemma A.1, for any $\ell = 1, \cdots, k$, the following holds

$$\|[z_i^{(\ell)}]_{k-\ell} - [z_{i'}^{(\ell)}]_{k-\ell}\|_2 \le \|[z_i^{(\ell)}]_{k-\ell+1} - [z_{i'}^{(\ell)}]_{k-\ell+1}\|_2.$$

Assume $\max_\ell \|[z_i^{(\ell)}]\|_2 \le c_z < \infty \; \forall i$, and $\max_\ell \|\theta^{(\ell)}\|_2 \le c_\theta < \infty$, where $c_\theta = \vee_\ell s_{\theta^{(\ell)}}$ the largest singular value.

Then, for $\ell = 1, \cdots, k-1$, we have

$$\begin{aligned}
&\|[z_{i'}^{(\ell)}]_{k-\ell} - [z_{i'}^{(\ell)}]_{k-\ell}\|_2 \\
\le &\|[\sigma([\phi(L_{g_i})]_{k-\ell+1}[z_i^{(\ell-1)}]_{k-\ell+1}\theta^{(\ell)}) - \sigma([\phi(L_{g_{i'}})]_{k-\ell+1}[z_{i'}^{(\ell-1)}]_{k-\ell+1}\theta^{(\ell)})]_{k-\ell})\|_2 \\
\le &\gamma_\sigma \|[\phi(L_{g_i})]_{k-\ell+1}[z_i^{(\ell-1)}]_{k-\ell+1} - [\phi(L_{g_{i'}})]_{k-\ell+1}[z_{i'}^{(\ell-1)}]_{k-\ell+1}\|_2 \|\theta^{(k)}\|_2 \\
\le &\gamma_\sigma c_\theta \|[\phi(L_{g_i})]_{k-\ell+1}\|_2 \|[z_i^{(\ell-1)}]_{k-\ell+1} - [z_{i'}^{(\ell-1)}]_{k-\ell+1}\|_2 + \gamma_\sigma c_\theta \|[z_{i'}^{(\ell-1)}]_{k-\ell+1}\|_2 \|[\phi(L_{g_i})]_{k-\ell+1} - [\phi(L_{g_{i'}})]_{k-\ell+1}\|_2 \\
\le &\gamma_\sigma c_\theta \|\phi(L_{g_i})\|_2 \|[z_i^{(\ell-1)}]_{k-\ell+1} - [z_{i'}^{(\ell-1)}]_{k-\ell+1}\|_2 + \gamma_\sigma c_\theta c_z \|\phi(L_{g_i}) - \phi(L_{g_{i'}})\|_2.
\end{aligned} \tag{10}$$

since $[\phi(L_{g_i})]_{k-\ell+1}$ is the principle submatrix of $\phi(L_{g_i})$. Then we equivalently write the above equation as $E_\ell \le bE_{\ell-1} + a$, which gives

$$E_\ell \le b^\ell E_1 + \frac{b^\ell + 1}{b-1} a.$$

With $[x_i] = [z_i^{(0)}]_k$, we see the following is only dependant on the structure of g_i and $g_{i'}$,

$$\begin{aligned}\|[z_{i'}^{(\ell)}]_{k-\ell} - [z_{i'}^{(\ell)}]_{k-\ell}\|_2 &\leq (\gamma_\sigma c_\theta)^\ell \|\phi(L_{g_i})\|_2^\ell \|[x_i] - [x_{i'}]\|_2 \\ &+ \frac{(\gamma_\sigma c_\theta)^\ell \|\phi(L_{g_i})\|_2^\ell + 1}{\gamma_\sigma c_\theta \|\phi(L_{g_i})\|_2 - 1} \gamma_\sigma c_\theta c_z \|\phi(L_{g_i}) - \phi(L_{g_{i'}})\|_2.\end{aligned}$$

□

Since the the graph Laplacians are normalised, we have $\|\phi(L_{g_i})\|_2 \leq c_L < \infty \; \forall i$. In addition, let

$$\|x_{p,q}^i - x_{p,q}^{i'}\|_2 \leq \sup_i \sup_{p,q} \|x_{p,q}^i - x_{p,q}^{i'}\|_2 = \sup_i \|f(L_{g_i}) - f(L_{g_{i'}})\|_2 := \delta_x.$$

Hence, $\|[x_i] - [x_{i'}]\|_2 \leq \delta_x (\sum_{p=1}^k r^p)^{1/2} := c_x$. From Lemma A.3, it is clear that we obtain the following at the final layer

$$\begin{aligned}I_2 = \|z_i - z_{i'}\|_2 &\leq (\gamma_\sigma c_\theta c_L)^k c_x + \frac{(\gamma_\sigma c_\theta c_L)^k + 1}{\gamma_\sigma c_\theta c_L - 1} \gamma_\sigma c_\theta c_z \|\phi(L_{g_i}) - \phi(L_{g_{i'}})\|_2 \\ &\leq C(M c_x + \|L_{g_i} - L_{g_{i'}}\|_2) \\ &= C(M c_x + \lambda_{\max}(L_{g_i} - L_{g_{i'}})^{1/2}).\end{aligned} \tag{11}$$

since ϕ is a linear function for L. Indeed, this can be generalised to polynomial function ϕ of higher powers.

Now, consider the following term that is related with discriminator $\mathcal{D}$,

$$I_1 = \sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} \left[\|h_{p,q}^i - h_{p,q}^{i'}\|_2 + \|x_{p,q}^i - x_{p,q}^{i'}\|_2 \right] / \sum_{p=1}^k r^p$$

Firstly, we denote $\tilde{L}_{p,q}$ as the in-degree graph Laplacian derived with the subgraph g_q of g_i centred at $q \in V_p(g_i)$. Different from the encoder, we utilize every node's hidden embedding in the computation. Specifically, g_q is obtained by retrieving links in g_i that connects to the qth node in the pth layer. This is a principle submatrix of the in-degree graph Laplacian $\tilde{L}_{g_i}$ of g_i.

Just as defined in §A.1, we denote $[h_q^{(p)}]_\ell$ as the pth layer GCN embedding for nodes in hop 0 to hop $\ell \in [0, p]$ of g_q. Note that in this case, $[h_q^{(p)}]_0 = h_q^{(p)}$, which is one row of $[h_i^{(p)}]$, corresponding to the q-th node in the neighborhood. So we may write the first term in I_1 as

$$\sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} \|h_q^{(p)} - h_{q'}^{(p)}\|$$

where $h_{q'}^{(p)} := h_{p,q}^{i'}$ for short. In this way, we regard each node $q \in V_p(g_i)$ as the centre node, which allows us to unfold the convolution similarly as expanding the I_2 term. Now, for any $q \in V_k(g_i)$, i.e. when $p = k$, we apply Lemma A.3 similarly as for $\|z_i - z_{i'}\|_2$. Then,

$$\begin{aligned}\|h_q^{(k)} - h_{q'}^{(k)}\| &\leq (\gamma_\sigma c_{\tilde{\theta}} c_{\tilde{L}})^k c_x + \frac{(\gamma_\sigma c_{\tilde{\theta}} c_{\tilde{L}})^k + 1}{\gamma_\sigma c_{\tilde{\theta}} c_{\tilde{L}} - 1} \gamma_\sigma c_{\tilde{\theta}} c_h \|\phi(\tilde{L}_{k,q}) - \phi(\tilde{L}_{k,q'})\|_2 \\ &\leq \tilde{C}_k (\tilde{M}_k c_x + \|\phi(\tilde{L}_{g_i}) - \phi(\tilde{L}_{g_{i'}})\|_2)\end{aligned}$$

where $\tilde{L}_{p,q}$ is the principle submatrix of $\tilde{L}_{g_i}$ and Lemma A.1 can be applied iin the last inequality. In addition, $\tilde{C}_k$ and $\tilde{M}_k$ are taken to be the maximum over any $q \in V_k(g_i)$. In general, for $q \in V_p(g_i)$, $0 < p < k$, we have

$$\|h_q^{(p)} - h_{q'}^{(p)}\|_2 \leq \tilde{C}_p (\tilde{M}_p c_x + \|\phi(\tilde{L}_{g_i}) - \phi(\tilde{L}_{g_{i'}})\|_2)$$

Take a common upper bound for $\tilde{C}_p, \tilde{M}_p$ over $0 < p \leq k$, we obtain

$$\begin{aligned}\sum_{p=1}^k \sum_{q=1}^{|V_p(g_{i'})|} \|h_q^{(p)} - h_{q'}^{(p)}\| / \sum_{p=1}^k r^p &\leq \tilde{C}(\tilde{M} c_x + \|\tilde{L}_{g_i} - \tilde{L}_{g_{i'}}\|_2) \\ &= \tilde{C}(\tilde{M} c_x + \lambda_{\max}(\tilde{L}_{g_i} - \tilde{L}_{g_{i'}})^{1/2})\end{aligned}$$

In addition, for the other half of I_1, we have

$$\sum_{p=1}^{k}\sum_{q=1}^{|V_p(g_{i'})|}\|x_{p,q}^{i}-x_{p,q}^{i'}\|_2/\sum_{p=1}^{k}r^p \leq \sup_i \sup_{p,q}\|x_{p,q}^{i}-x_{p,q}^{i'}\|_2=\delta_x=c_x/(\sum_{p=1}^{k}r^p)^{1/2}$$

We can write $\mathcal{B}$ in terms of weights C and $\tilde{C}$, which is dependant on the activation function σ, k and $\sup_i \lambda_{\max}(L_{g_i})$. Hence,

$$\begin{aligned}B &\leq (CM+\tilde{C}\tilde{M}+1/(\sum_{p=1}^{k}r^p))c_x + C\lambda_{\max}(L_{g_i}-L_{g_{i'}})+\tilde{C}\lambda_{\max}(\tilde{L}_{g_i}-\tilde{L}_{g_{i'}})\\ &= M'c_x + C\lambda_{\max}(L_{g_i}-L_{g_{i'}})+\tilde{C}\lambda_{\max}(\tilde{L}_{g_i}-\tilde{L}_{g_{i'}})\end{aligned}$$

Note that the derived I_1 for B is the same for A, since the node features, edge information and embedded features are bounded by separate terms in Eq. 8. The only difference is given by I_2, where a different set of graph Laplacians L_{g_j}, $L_{g_{j'}}$ and node features (x_j) are used. Therefore,

$$A \leq M'c_x + C\lambda_{\max}(L_{g_j}-L_{g_{j'}})+\tilde{C}\lambda_{\max}(\tilde{L}_{g_i}-\tilde{L}_{g_{i'}})$$

Hence the result. □

Note that, our view of structural information is closely related to graph kernels [4] and graph perturbation [55]. Specifically, our Definition on k-hop ego-graph is motivated by the concept of k-layer expansion sub-graph in [4]. However, [4] used the Jensen-Shannon divergence between pairwise representations of sub-graphs to define a depth-based sub-graph kernel, while we depict G as samples of its ego-graphs. In this sense, our view is related to the setup in [55], which derived a uniform algorithmic stability bound of a 1-layer GNN under 1-hop structure perturbation of G.

In the setting of domain adaptation, [7] draws a connection between the difference in the distributions of source and target domains and the model transferability, and learns a transferable model by minimizing such distribution differences. This coincides with our approach of connecting the structure difference of two graphs in terms of k-hop subgraph distributions and the transferability of GNNs in the above theory.

B Model Details

Following the same notations used in the main paper, EGI consists of a GNN encoder Ψ and a GNN discriminator $\mathcal{D}$. In general, the GNN encoder Ψ and discriminator $\mathcal{D}$ can be any existing GNN models. For each ego-graph and its node features $\{g_i, x_i\}$, the GNN encoder returns node embedding z_i for the center node v_i. As mentioned in Eq. 2 in the main paper, the GNN discriminator $\mathcal{D}$ makes edge-level predictions as follows,

$$\mathcal{D}(e_{\tilde{v}v}|h_{p,q}^{\tilde{q}}, x_{p,q}^{i}, z_i) = \sigma\left(U^T\cdot\tau\left(W^T[h_{p,q}^{\tilde{q}}||x_{p,q}^{i}||z_i]\right)\right), \tag{12}$$

where $e_{\tilde{v}v}\in E(g_i)$ and $h_{p,q}^{\tilde{q}}\in\mathbb{R}^d$ (simplified as h_p in the main paper, same for $x_{p,q}^{i}=x_q$) is the representation for edge $e_{\tilde{v}v}$ between node $v_{p-1,\tilde{q}}$ in hop $p-1$ and $v_{p,q}$ in hop p. The prediction relies on the combination of center node embedding z_i, destination node feature $x_{p,q}^{i}$ and source node representation $h_{p,q}^{\tilde{q}}$. And now we describe how we calculate the source node representation in $\mathcal{D}$.

To obtain the source node representation representations h, the GNN in discriminator $\mathcal{D}$ operates on a reversed ego-graph $\tilde{g}_i$ while encoder Ψ performs forward propagation on g_i. The discriminator GNN starts from the center node v_i and compute the hidden representation $m_{p-1,\tilde{q}}$ for node $v_{p-1,q}$ at each hop. We denote the source node at $p-1$ hop as $\tilde{q}\in\tilde{Q}_{p,q}, \tilde{Q}_{p,q}=\{\tilde{q}: v_{p-1,\tilde{q}}\in V_{p-1}(g_i), e_{(p-1,\tilde{q})(p,q)}\in E(g_i)\}$. Although $h_{p,q}$ is calculated as node embedding, in reversed ego graph $\tilde{g}_i$, node only has one incoming edge. Thus, we can also interpret $h_{p,q}^{\tilde{q}}$ as the edge embedding as it combines source node's hidden representation $m_{p-1,\tilde{q}}$ and destination node features $x_{p,q}$ as follows,

$$h_{p,q}^{\tilde{q}} = \text{ReLU}\left(W_p^T\left(m_{p-1,\tilde{q}}+x_{p,q}^{i}\right)\right),\ m_{p-1,\tilde{q}}=\frac{1}{|\tilde{Q}_{p-1,\tilde{q}}|}\sum_{q'\in\tilde{Q}_{p-1\tilde{q}}}h_{p-1,\tilde{q}}^{q'} \tag{13}$$

Algorithm 1: Pseudo code for training EGI

The GNN encoder Ψ and the GNN discriminator $\mathcal{D}$, k-hop ego graph and features $\{g_i, x_i\}$;
/* EGI-training starts */
while $\mathcal{L}_{\text{EGI}}$ *not converges* **do**
  Sample M ego-graphs $\{(g_1, x_1), ..., (g_M, x_M)\}$ from empirical distribution $\mathbb{P}$ without replacement, and obtained their positive and negative node embeddings z_i, z_i' through Ψ

$$z_i = \Psi(g_i, x_i), z_i' = \Psi(g_i', x_i'),$$

  /* Initialize positive and negative expectation in Eq. 1 in the main paper*/
  $E_{pos} = 0, E_{neg} = 0$
  for p = *1 to* k **do**
    /* Compute JSD on edges at each hop*/
    for $e_{(p-1,\tilde{q})(p,q)} \in E(g_i)$ **do**
      generate source node embedding $h_{p,q}^{\tilde{q}}$ in Eq. 13 ;
      $E_{\text{pos}} = E_{\text{pos}} + \sigma\left(U^T \cdot \tau\left(W^T[h_{p,q}^{\tilde{q}} || x_{p,q}^i || z_i]\right)\right)$
      $E_{\text{neg}} = E_{\text{neg}} + \sigma\left(U^T \cdot \tau\left(W^T[h_{p,q}^{\tilde{q}} || x_{p,q}^i || z_i']\right)\right)$
    end
  end
  /* Compute batch loss*/
  $\mathcal{L}_{\text{EGI}} = E_{\text{neg}} - E_{\text{pos}}$
  /* Update Ψ, $\mathcal{D}$ */
  $\theta_\Psi \xleftarrow{+} -\nabla_\Psi \mathcal{L}_{\text{EGI}}, \theta_\mathcal{D} \xleftarrow{+} -\nabla_\mathcal{D} \mathcal{L}_{\text{EGI}}$
end

When $p = 1$, every edge origins from the center node v_i and $m_{0,q'}$ is the center node feature x_{v_i}. Note that we the elaborated aggregation rule is equivalent as layer-wise propagation rules (different in-degree matrix for each $h_{p,q}$) of EGI earlier in §A.1.

In every batch, we sample a set of ego-graphs and their node features $\{g_i, x_i\}$. During the forward pass of encoder Ψ, it aggregates from neighbor nodes to the center node v_i. Then, the discriminator calculates the edge embedding in Eq. 13 from center node v_i to its neighbors and make edge-level predictions– *fake* or *true*. Besides training framework Figure 2 in the main paper, the algorithm EGI is depicted in Algorithm 1.

We implement our method and all of the baselines using the same encoders Ψ: 2-layer GIN [60] for synthetic and role identification experiments, 2-layer GraphSAGE [15] for the relation prediction experiments. We set hidden dimension as 32 for both synthetic and role identification experiments, For relation prediction fine-tuning task, we set hidden dimension as 256. We train EGI in a mini-batch fashion since all the information for encoder and discriminators are within the k-hop ego-graph g_i and its features x_i. Further, we conduct neighborhood sampling and set maximum neighbors as 10 to speed up the parrallel training. The space and time complexity of EGI is $O(BN^K)$, where B is the batch size, N is the number of the neighbors and k is the number of hops of ego-graphs. Notice that both the encoder Ψ and discriminator $\mathcal{D}$ propagate message on the k-hop ego-graphs, so the extra computation cost of $\mathcal{D}$ compared with a common GNN module is a constant multiplier over the original one. The scalability of EGI on million scale YAGO network is reported in section C.3.

B.1 Transfer Learning Settings

The goal of transfer learning is to train a model on a dataset or task, and use it on another. In our graph learning setting, we focus on training the model on one graph and using it on another. In particular, we focus our study on the setting of *unsupervised-transfering*, where the model learned on the source graph is directly applied on the target graph without *fine-tuning*. We study this setting because it allows us to directly measure the transferability of GNNs, which is not affected by the fine-tuning process on the target graph. In other words, the fine-tuning process introduces significant uncertainty to the analysis, because there is no guarantee on how much the fine-tuned GNN is different from the pre-trained one. Depending on specific tasks and labels distributions on the two graphs, the fine-tuned

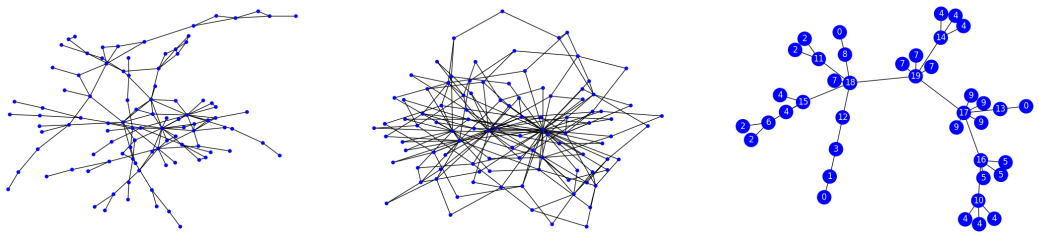
GNN might be quite similar to the pre-trained one, or it can be significantly different. It is then very hard to analyze how much the pre-trained GNN itself is able to help. Another reason is about efficiency. The fine-tuning of GNNs requires the same environment set-up and computation resource as training GNNs from scratch, although it may take less training time eventually if pre-training is effective. It is intriguing if this whole process can be eliminated when we guarantee the performance with unsupervised-transfering.

In our experiments, we also study the setting of transfer learning with fine-tuning, particularly on the real-world large-scale YAGO graphs. Since we aim to study the general transferability of GNNs not bounded to specific tasks, we always pre-train GNNs with the unsupervised pre-training objective on source graphs. Then we enable two types of fine-tuning. The first one is *post-fine-tuning* ($\mathcal{L} = \mathcal{L}_s$), where the pre-trained GNNs are fine-tuned with the supervised task specific objective $\mathcal{L}_s$ on the target graphs. The second on is *joint-fine-tuning* ($\mathcal{L} = \mathcal{L}_s + \mathcal{L}_u$), where pre-training is the same, but fine-tuning is done *w.r.t.* both the pre-training objective $\mathcal{L}_u$ and task specific objective $\mathcal{L}_s$ on target graphs in a semi-supervised learning fashion. The unsupervised pre-training objective $\mathcal{L}_u$ of EGI is Algorithm 1, while those of the compared algorithms are as defined in their papers. The supervised fine-tuning objective $\mathcal{L}_s$ is the same as in the DistMult paper [61] for all algorithms.

C Additional Experiment Details

C.1 Synthetic Experiments

Data. As mentioned in the main paper, we use two traditional graph generation models for synthetic data generation: (1) barabasi-albert graph [5] and (2) forest-fire graph [32]. We generate 40 graphs each with 100 nodes with each model. We control the parameters of two models to generate two graphs with different ego-graph distributions. Specifically, we set the number of attached edges as 2 for barabasi-albert model and set $p_{\text{forward}} = 0.4$, $p_{\text{backward}} = 0.3$ for forest-fire model. In Figure 4a and 4b, we show example graphs from two families in our datasets. They have the same size but different appearance which leads to our study on the transferability gap $\Delta_{\mathcal{D}}$(F, F) and $\Delta_{\mathcal{D}}$(B, F) in Table 1 in the main paper. The accuracy of this task defined as the percentage of nearest neighbors for target node in the embedding space $z = \Psi(\cdot)$ that are structure-equivalent, *i.e.* #correct k-nn neighbors / #ground truth equivalent nodes.



(a) Forest-fire graph example **(b)** Barabasi-albert graph example **(c)** structural label example

Figure 4: Visualizations of the graphs and labels we use in the synthetic experiments.

Results. The structural equivalence label is obtained by a 2-hop WL-test [58] on the ego-graphs. If two nodes have the same 2-hop ego-graphs, they will be assigned the same label. In the example of Figure 4c, the nodes labeled with same number (*e.g.* 2, 4) have the isomorphic 2-hop ego-graphs. Note that this task is exactly solvable when node features and GNN architectures are powerful enough like GIN [60]. In order to show the performance difference among different methods, we set the length of one-hot node degree encoding to 3 (all nodes with degrees higher than 3 have the same encoding). Here, we present the performance comparison with different length of degree encodings (d) in Table 4. When the capacity of initial node features is high (d=10), the transfer learning gap diminishes between different methods and different graphs because the structural equivalence problem can be exactly solved by neighborhood aggregations. However, when the information in initial node

features is limited, the advantage of EGI in learning and transfering the graph structural information is obvious. In Table 5, we also show the performance of different transferable and non-transferable features discussed after Definition 4.3 in the main paper, *i.e.* node embedding [42] and random feature vectors. The observation is similar with Table 1 in the main paper: the transferable feature can reflect the performance gap between similar and dissimilar graphs while non-transferable features can not.

In both Table 4 and 8 here as well as Table 1 in the main paper, we report the structural difference among graphs in the two sets ($\bar{d}$) calculated *w.r.t.* the term $\Delta_{\mathcal{D}}(G_a, G_b)$ on the RHS of Theorem 4.1 in the main paper. This indicates that the Forest fire graphs are structurally similar to the other Forest fire graphs, while less similar to the Barabasi graphs, as can be verified from Figure 4a and 4b. Our bound in Theorem 4.1 then tells us that the GNNs (in particular, EGI) should be more transferable in the F-F case than B-F. This is verified in Table 4 and 5 when using the transferable node features of degree encoding with limited dimension (d=3) as well as DeepWalk embedding, as EGI pre-trained on Forest fire graphs performs significantly better on Forest fire graphs than on Barabasi graphs (with +0.094 and +0.057 differences, respectively).

Table 4: Synthetic experiments of identifying structural-equivalent nodes with different degree encoding dimensions.

Method	**#dim degree encoding d = 3**			**# dim degree encoding d = 10**			**structural difference**	
	F-F	B-F	δ(acc.)	F-F	B-F	δ(acc.)	Δ_D(F,F)	Δ_D(B,F)
GCN (untrained)	0.478	0.478	/	0.940	0.940	/		
GIN (untrained)	0.572	0.572	/	0.940	0.940	/		
VGAE (GIN)	0.498	0.432	+0.066	0.939	0.937	0.002	0.752	0.883
DGI (GIN)	0.578	0.591	-0.013	0.939	0.941	-0.002		
EGI (GIN)	**0.710**	0.616	+0.094	0.942	0.942	0		

Table 5: Synthetic experiments of identifying structural-equivalent nodes with different transferable and non-transferable features.

Method	**DeepWalk embedding**			**random vectors**			**structural difference**	
	F-F	B-F	δ(acc.)	F-F	B-F	δ(acc.)	Δ_D(F,F)	Δ_D(B,F)
GCN (untrained)	0.658	0.658	/	0.246	0.246	/		
GIN (untrained)	0.663	0.663	/	0.520	0.520	/		
GVAE (GIN)	0.713	0.659	+0.054	0.266	0.264	0.002	0.752	0.883
DGI (GIN)	0.640	0.613	+0.027	0.512	0.576	-0.064		
EGI (GIN)	**0.772**	0.715	+0.057	0.507	0.485	+0.022		

C.2 Real-world Role Identification Experiments

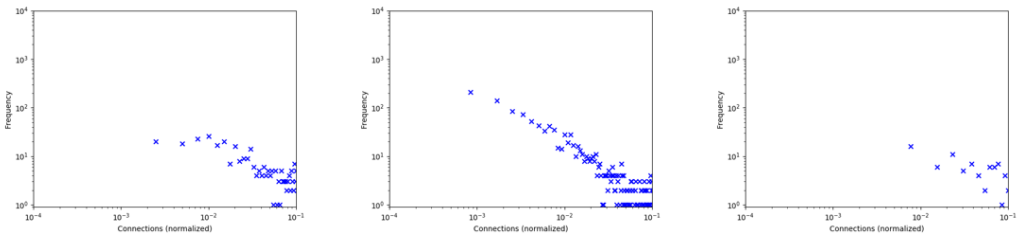
Data. We report the number of nodes, edges and classes for both airport and gene dataset. The numbers for the Gene dataset are the aggregations of the total 52 gene networks in the dataset. For the three airport networks, Figure 5 shows the power-law degree distribution on log-log scale. The class labels are between 0 to 3 reflecting the level of the airport activities [45]. For the Gene dataset, we matched the gene names in the TCGA dataset [68] to the list of transcription factors on wikipedia[5]. 75% of the genes are marked as 1 (transcription factors) and some gene graphs have extremely imbalanced class distributions. So we conduct experiments on the relatively balanced gene graphs

[5] `https://en.wikipedia.org/wiki/Transcription_factor`

Table 6: Overall Dataset Statistics

Dataset	# Nodes	# Edges	# Classes
Europe	399	5,995	4
USA	1,190	13,599	4
Brazil	131	1,074	4
Gene	9,228	57,029	2

of brain cancers (Figure 2 in the main paper). Both datasets do not have organic node attributes. The role-based node labels are highly relevant to their local graph structures, but are not trivially computable such as from node degrees.



(a) Europe airport log-log plot **(b)** USA airport log-log plot **(c)** Brazil airport log-log plot

Figure 5: Visualizations of power-law degree distribution on three airport dataset.

Results. As we can observe from Figure 5, the three airport graphs have quite different sizes and structures (*e.g.*, regarding edge density and connectivity pattern). Thus, the absolute classification accuracy in both Table 2 in the main paper and Table 8 here varies across different graphs. However, as we mention in the main paper, the structural difference we compute based on Eq. 5 in Theorem 3.1 is close among the Europe-USA and Europe-Brazil graph pairs (0.869 and 0.851), which leads to close transferability of EGI from Europe to USA and Brazil. This indicates the effectiveness of our view over essential structural information. We also provide detailed standard deviations of Table 2 (main paper) when using node degree as features.

Table 7: Results of role identification with direct-transfering on the Airport dataset (Table 2, main paper). The performance reported (%) are the average over 100 runs. We set all node features same as non-transferable features.

Method	Europe (source)		USA (target)		Brazil (target)	
	node degree	same feat.	node degree	same feat.	node degree	same feat.
features	52.81	20.59	55.67	20.22	67.11	19.63
GIN (untrained)	55.75	53.88	61.56	58.32	70.04	70.37
GVAE	53.90	21.12	55.51	22.39	66.33	17.70
DGI	57.75	22.13	54.90	21.76	67.93	18.78
MaskGNN	56.37	55.53	60.82	54.64	66.71	74.54
ContextPredGNN	52.69	49.95	50.38	54.75	62.11	70.66
Structural Pre-train	56.00	53.83	62.17	57.49	68.78	72.41
MVC	53.16	51.69	59.66	50.42	66.07	61.55
GMI	58.12	46.25	59.28	47.64	73.07	62.96
EGI (GIN)	**59.15****	54.98	**64.55****	57.40	**73.15**	70.00

Besides that, the results present in Table 8 are the accuracy of GNNs directly trained and evaluated on each network without transfering. Therefore, only the Europe column has the same results as in Table 2 in the main paper, while the USA and Brazil columns can be regarded as providing an upper-bound performance of GNN transfered from other graphs. As we can see, EGI gives the closest results from Table 2 (main paper) to Table 8 here, demonstrating the its plausible transferability. The scores are so close, showing a possibility to skip fine-tuning when the source and target graphs are similar enough. Also note that, although the variances are pretty large (which is also observed in other works like [45] since the networks are small), our t-tests have shown the improvements of EGI to be significant.

C.3 Real-world large-scale Relation Prediction Experiments

Data. As shown in Table 9, the source graph we use to pre-train GNNs is the full graph cleaned from the YAGO dump [49], where we assume the relations among entities are unknown. The target

Table 8: Role identification that identifies structurally similar nodes on real-world networks. The performance reported are the average and standard deviation for 10 runs. Our classification accuracy on three datasets all passed the t-test ($p<0.01$) with the second best result in the table.

Method	Airport [45]		
	Europe	USA	Brazil
node degree	52.81% ± 5.81%	55.67% ± 3.63%	67.11% ± 7.58%
GCN (random-init)	52.96% ± 4.51%	56.18% ± 3.82%	55.93% ± 1.38%
GIN (random-init)	55.75% ± 5.84%	62.77% ± 2.35%	69.26% ± 9.08%
GVAE (GIN)	53.90% ± 4.65%	58.99% ± 2.44%	55.56% ± 6.83%
DGI (GIN)	57.75% ± 4.47%	62.44% ± 4.46%	68.15% ± 6.24%
Mask-GIN	56.37% ± 5.07%	63.78% ± 2.79%	61.85% ± 10.74%
ContextPred-GIN	52.69% ± 6.12%	56.22% ± 4.05%	58.52% ± 10.18%
Structural Pre-train	56.00% ± 4.58%	62.29% ± 3.51%	71.48% ± 9.38 %
MVC	53.16% ± 4.07%	62.81 % ± 3.12%	67.78 % ± 4.79%
GMI	58.12 % ± 5.28%	63.36 % ± 2.92%	73.70% ± 4.21%
EGI (GIN)	**59.15%** ± **4.44**%	**65.88%** ± **3.65**%	**74.07%** ± **5.49**%

Table 9: dataset statistics and running time of EGI

Dataset	# Nodes	# Edges	# Relations	# Train/Test	Training time per epoch
YAGO-Source	579,721	2,191,464	/	/	338 seconds
YAGO-Target	115,186	409,952	24	480/409,472	134 seconds

graph we use is a subgraph uniformed sampled from the same YAGO dump (we sample the nodes and then include all edges among the sampled nodes). The similar ratio between number of nodes and edges can be observed in Table 9. On the target graph, we also have the access to 24 different relations [48] such as *isAdvisedBy*, *isMarriedTo* and so on. Such relation labels are still relevant to the graph structures, but the relevance is lower compared with the structural role labels. We use the 256-dim degree encoding as node features for pre-training on the source graph, then we use the 128-dim positional embedding generated by LINE [51] for fine-tuning on the target graph, to explicitly make the features differ across source and target graphs.

Results. In Section B.1, we introduced two different types of fine-tuning, *i.e.*, *post-fine-tuning* and *joint-fine-tuning*. For both types of fine-tuning, we add one feature encoder $\mathcal{E}$ before feeding it into the GNNs for two purposes. First, the target graph fine-tuning feature usually has different dimensions with the pre-training features, such as the node degree encoding we use. Second, the semantics and distributions of fine-tuning features can be different from pre-training features. The feature encoder aims to bridge the gap between feature difference in practice. The supervised loss used in this experiment is the same as in DistMult [61]. In particular, the bilinear score function is calculated as $s(h, r, t) = z_h^T M_r z_t$, where M_r is a diagonal matrix for each relation r, z_h and z_t the the embedding of GNN encoder Ψ for head and tail entities. The experiments were run on GTX1080 with 12G memories. We report the average training time per epoch of our algorithm in pre-training and fine-tuning stage in Table 9 as well. The pre-training and fine-tuning takes about 40 epochs and 10 epochs to converge, respectively. In Table 9, we also present the per-epoch training time of EGI. EGI takes about 338 seconds per epoch for optimizing the ego-graph information maximization objective on YAGO-source. As we can see, fine-tuning also takes significant time compared to pre-training, which strengthens our arguments about avoiding or reducing fine-tuning through structural analysis. We implement all baselines within the same pipeline, and the running times are all in same scale.

C.4 Parameter study

In this section, we provide additional parameter analysis towards proposed EGI model - choices of k, and efficiency study on EGI gap $\Delta_{\mathcal{D}}$ - sampling frequencies.

Performance of different size of ego-graphs. In our Theorem 3.1 and EGI algorithm (Eq. 1), number of hops k determines the size of ego-graphs. In principle, k may affect the transferability of EGI in two ways: (1) larger k may make the EGI model (both center node encoder Ψ and neighbor node encoder Φ) more expressive (better precision) and the EGI gap $\Delta_{\mathcal{D}}$ more accurate (better

Table 10: Comparison of EGI with different k. Accuracy and EGI gap $\Delta_{\mathcal{D}}$ are reported.

	Europe (source)	USA (target)		Brazil (target)	
	acc.	acc.	$\Delta_{\mathcal{D}}$	acc.	$\Delta_{\mathcal{D}}$
EGI (k=1)	58.25	60.08	0.385	60.74	0.335
EGI (k=2)	59.15	64.55	0.869	73.15	0.851
EGI (k=3)	57.63	64.12	0.912	72.22	0.909

predictiveness); (2) However, the GNN encoders may suffer from the over-smoothing problem and the computations may suffer from more noises. Therefore, it is hard to determine the influence of k without empirical analysis. As we can observe in , when $k = 1$ or $k = 3$, the classification accuracy of the source graph is worse than $k = 2$, likely because the GNN encoder is either less powerful or over-smoothed. As a result, $k = 2$ obtains the best transferability to both the USA and Brazil networks. When $k = 3$, $\Delta_{\mathcal{D}}$ likely accounts for too subtle/noisy ego-graph differences and may become less effective in predicting the transferability. Therefore, we choose $k = 2$ to conduct experiments in main paper.

Precision of $\Delta_{\mathcal{D}}$ under different sampling frequencies. In Table 11, we present the estimated $\Delta_{\mathcal{D}}$ versus sampling frequency for 10 runs on airport dataset. A theoretical study on its convergence could be an interesting future direction. As we can observe, large sample frequency leads to more accurate and robust estimation of $\Delta_{\mathcal{D}}$. Between Europe and USA, although 100 pairs of ego-graphs are only equivalent as 2.1% of the total pair-wise enumerations, the estimated $\Delta_{\mathcal{D}}$ is pretty close.

Table 11: EGI gap $\Delta_{\mathcal{D}}$ on airport dataset with different sampling frequencies.

Sampling frequency	$\Delta_{\mathcal{D}}$(Europe, USA)	$\Delta_{\mathcal{D}}$(Europe, Brazil)
100 pairs	0.872±0.039	0.854±0.042
1000 pairs	0.859±0.012	0.848±0.007
All pairs	0.869 ±0.000	0.851 ±0.000

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Mainly see Sections 3 and 4
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.2 and Appendix §A
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix §A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 4 and supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Due to space limit, but we conducted significant tests on all claimed improvements
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4
   (b) Did you mention the license of the assets? [No] They are all public
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide our models and code in the supplemental material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]