Inductive Graph Embeddings through Locality Encodings

Nurudin Alvarez-Gonzalez^{1,3,*} Andreas Kaltenbrunner^{2,3} Vicenç Gómez^{3}
nuralgon@gmail.com, andreas.kaltenbrunner@isi.it, vicen.gomez@upf.edu
^{1}NTENT Hispania, Barcelona, Spain
^{2}ISI Foundation, Turin, Italy
^{3}Universitat Pompeu Fabra, Barcelona, Spain

Abstract

This work aims to find inductive network embeddings in large networks without domain-dependent node/edge attributes. We propose to use a set of basic predefined local encodings as the basis of a learning algorithm. In particular, we consider the degree frequencies at different distances from a node, which can be computed efficiently for relatively short distances and a large number of nodes. The resulting embeddings generalize well across unseen or distant regions in the network, both in unsupervised settings, when combined with language model learning, as well as in supervised tasks, when used as additional features in a neural network. Despite its simplicity, this method achieves state-of-the-art performance in tasks such as role detection, link prediction and node classification, and represents an inductive network embedding method directly applicable to large unattributed networks.

Introduction

Graph structures are the natural way to represent arbitrary relationships in complex domains. Because of their generality, they can be applied to problems in which represented entities display rich interactions between each other. However, compactly capturing such interactions is a non-trivial task, often made unfeasible, for example, by the high-dimensionality and complexity of real-world networks.

Representation learning has been successfully applied to graphs, capturing useful interactions in compact ways [6]. The learned representations can be used for downstream machine learning tasks, and have seen applications as varied as molecule generation for drug design [17], content recommendation [23], or social network analysis [15].

A wide array of learning approaches have focused on local node features and interactions. *Transductive* approaches require all nodes in the graph to be available at training time to learn a representation for each of them. Transductive methods can capture rich relationships within networks in a scalable manner. However, the resulting representations do not generalize to unseen nodes or edges, and require additional attention to deal with label permutations.

In contrast, *inductive* approaches [5] aim to compactly represent network interactions so that the specific identity of a node is irrelevant to its representation. Inductive representations are typically derived from attributes of nodes and edges within their neighbourhoods. In either case, inductive models tend to be more demanding than their transductive counterparts, particularly when entire graphs or node neighbourhoods ought to be sampled during training [9].

In this paper, we introduce **IGEL** (**I**nductive **G**raph **E**mbeddings through **L**ocality Encoding), a novel inductive method for learning structural graph embeddings. IGEL aims to overcome the aforementioned issues in previous works by representing network structures explicitly, in the form of neighbourhood structure dictionaries. Dense vector representations are then learned from the structural dictionaries, in either an unsupervised graph-dependent process or under full supervision for a target task.

Related Work

In this section, we briefly review the contributions most related to our work, grouping them according to their setting and their associated limitations. We then describe IGEL in this context, and give an introduction to how the method serves as a solution to existing problems. For a more comprehensive review of graph embedding techniques at large, we refer the reader to existing surveys [3, 22].

Unattributed Graph Embeddings

Initial approaches to learning graph representations focused on unattributed graphs. Models such as DeepWalk [15] learn node representations by training a word2vec [12] skip-gram language model over sequences of node labels sampled with random walks generated from a graph. Later, node2vec [4] extended this idea by introducing parameters to control the locality of the random walks and the properties of the learned latent space. LINE [19] tailored the representations towards graph structures by learning to distinguish adjacent nodes and nodes with similar edges. VERSE [20] introduces similarity measures such as personalized PageRank or SymRank into the loss function, to capture explicit properties from underlying network relationships in contrast to context-based language models. Finally, models like struct2vec [16], directly adapt the representation of a node with the substructures surrounding it.

*Work prior to joining Amazon.com Inc.

Convolutional Graph Networks

On attributed networks, research has focused on inductive graph representations. Inductive models generalize to unseen graphs from a given domain if the same features are available. Current inductive representation research has focused on graph convolutional networks (GCNs) [14] and their variants. GCNs aggregate features across nodes and edges through neighbourhood-wide pooling operations and transforms. GCNs have been successfully applied for semi-supervised learning [9] or node classification [21].

However, training GCNs on massive graphs remains a challenge. Some work focuses on sampling the graph in some fashion rather than considering complete node neighbourhoods. For instance, LGCL [2] starts from a randomly chosen set of nodes, and incrementally adds nodes by sampling according to a BFS strategy, until reaching a budget of nodes. GraphSAGE [5] represents a node by sampling a fixed number of neighbours at each distance from the node. Sampling approaches bound computational complexity to a constant factor, trading computational costs for predictive performance. Extensions such as PinSAGE [23] have shown that sampling can cope with web-scale problems by adapting existing models to distributed environments.

More recently, Position-aware Graph Neural Networks (P-GNNs) [24] have been proposed to extend GNNs representations with the relative position of nodes in the network. P-GNNs learn to represent nodes conditioned by anchor sets of nodes, aggregating information not only from its neighbourhoods, but also against each anchor-set to capture its position in the graph. P-GNNs have been derived for networks with node features, whose predictive power can be confounded with the positional information. In this work, we take the opposite direction: starting from a network with no attributes, we explore the role of position information regarding structural similarity within node neighbourhoods.

Our Contribution in Context

We tackle the problems about existing graph representation methods in unattributed inductive settings, namely:

A *The difficulty in translating transductive network embeddings such as DeepWalk or node2vec to unseen graphs from the same domain*. Rather than working with individual nodes, we learn embeddings of graph structures, which capture more general properties of the graph.

B *Graph Convolutional Networks on unattributed graphs leverage node and edge attributes*. We propose an inductive yet unattributed representation learning method. The resulting embeddings can serve as additional input features to deep neural networks or graph convolutions.

C *The generalization problems of existing structural embedding methods*. Methods such as struct2vec [16] do not transfer to unseen graphs. We propose a similar structural representation, but avoid computing node-to-node meta-graph structures and instead learning a random-walk based inductive representation.

The IGEL Algorithm

This section provides an overview of IGEL and drills down into the two steps that compose its high level algorithm.

Overview

The main idea of IGEL is to represent local graph structures inductively by learning dense embeddings over a sparse space of structural attributes that appear in the surroundings of nodes in a graph. These dense embeddings can be learned in supervised or unsupervised ways, with the overall process encompassing two main steps. For a given graph $\mathcal{G} = \langle V, E \rangle$ of vertices V and edges E, representing a node requires:

1. **Encoding.** The first step encodes local graph structures in the form of a sparse vector recording the frequencies of structural attributes found for every given node $n \in V$.
2. **Embedding.** The sparse encoding is used as input to an embedding model, that transforms the vector into a dense representation of the aggregated structural attributes.

Encoding Local Graph Structures

We capture the local structure by considering the neighbourhood of a node. For a given node $n \in V$ and encoding distance $\alpha \in \mathbb{N}$, let $\mathcal{G}_\alpha(n)$ denote the sub-graph of $\mathcal{G}$ containing all nodes found at distance at most α from n, and the edges connecting them. From the neighbourhood graph $\mathcal{G}_\alpha(n)$, we represent the structural features of node n by considering the number of times a degree δ is found at distance c in the graph $\mathcal{G}_\alpha(n)$, $c = 0, \dots, \alpha$.

The structural features can be understood as a succession of distance $c \in \mathbb{N}$ and degree $\delta \in \mathbb{N}$ tuples (c, δ) and their respective counts. Given the max-degree δ_{max} of $\mathcal{G}$, a sparse feature vector $\mathbf{x}_n \in \mathbb{R}^\ell$, with $\ell = \delta_{\text{max}} \times (\alpha + 1)$, can be constructed such that the ith component of $\mathbf{x}_n$, $i = c\delta_{\text{max}} + \delta$, contains the number of times τ the tuple (c, δ) appears in $\mathcal{G}_\alpha(n)$.

To construct a robust encoding, we log-transform each count as $\log_2(1 + \tau)$ and normalize the values at every encoding distance from the source by dividing by the sum of counts at that distance. At inference time, we use the max-degree bin δ_{max} to store the frequencies of nodes with degree $\delta' > \delta_{\text{max}}$, while unseen frequencies such that $\delta' < \delta_{\text{max}}$ are mapped to their corresponding (randomly initialised) bin. Figure 1 shows a representation of the encoding step prior to normalization.

Motivating the Encoding Scheme

The main motivation for our encoding is the idea that isomorphic sub-graphs should share the same encodings. This is similar to the meta-graph in struct2vec [16]. Our approach is directly inspired by graph isomorphism tests [18, 1] that use iterated degree sequences [8]. However, rather than packing the sequences into labels, our objective is to use the encodings to learn a dense representation downstream. We thus keep uncompressed counts of the degrees.

Additionally, our encoding only captures structural relationships by using "local" degrees, e.g., computed only in the sub-graph containing the neighbours of the source node

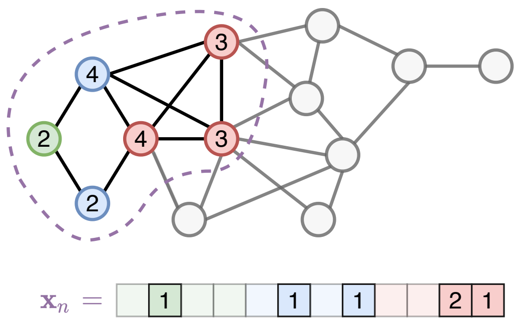


Figure 1: Encoding example. Dashed region denotes the neighbourhood of the green node n for distance $\alpha = 2$. Green nodes are at distance 0, blue nodes at 1 and red nodes at 2. Numbers indicate the degrees in the induced graph. The structural representation for the green node is the frequency mapping of distance-degree (c, δ) pairs: $\{(0,2):1, (1,2):1, (1,4):1, (2,3):2, (2,4):1\}$, which translates into the sparse structural feature vector $\mathbf{x}_n$.

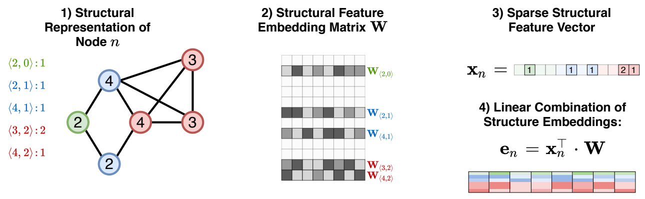
at distance α. We introduce this constraint to prevent the representation from focusing on specific degree values from the whole network that act as "landmarks", which we expect to generalise poorly to unseen graphs.

Embedding of the Sparse Representations

We can now represent a node in terms of its immediate surroundings, but this representation is sparse and high dimensional. A simple way to transform the sparse structural features into a dense, lower dimensional, representation is through a linear transformation:

$$\mathbf{e}_n = \mathbf{x}_n^\top \cdot \mathbf{W}, \tag{1}$$

where $\mathbf{e}_n$ is d-dimensional, and $\mathbf{W} \in \mathbb{R}^{\ell \times d}$ is a learnable matrix of parameters, $d \ll \ell$. Each row in $\mathbf{W}$ captures the vector representation of each single structural feature. This can be understood as a weighted average of the encoded structural features, representing the node in terms of its immediate surroundings. Figure 2 provides a visual overview.



1) Structural Representation of Node n
$\langle 2,0\rangle : 1$
$\langle 2,1\rangle : 1$
$\langle 4,1\rangle : 1$
$\langle 3,2\rangle : 2$
$\langle 4,2\rangle : 1$
4
3
2
4
3
2
2) Structural Feature Embedding Matrix $\mathbf{W}$
$\mathbf{W}_{\langle 2,0\rangle}$
$\mathbf{W}_{\langle 2,1\rangle}$
$\mathbf{W}_{\langle 4,1\rangle}$
$\mathbf{W}_{\langle 3,2\rangle}$
$\mathbf{W}_{\langle 4,2\rangle}$
3) Sparse Structural Feature Vector
$\mathbf{x}_n =$
1
1
1
2
1
4) Linear Combination of Structure Embeddings:
$\mathbf{e}_n = \mathbf{x}_n^\top \cdot \mathbf{W}$

Figure 2: Structural embedding example (cont). A sparse vector $\mathbf{x}_n$ is created from the (degree, distance) frequency pairs of the neighborhood of a node n. An embedding matrix $\mathbf{W}$ is used as a linear mapping of the sparse vector $\mathbf{x}_n$ into the final embedding $\mathbf{e}_n$, a dense representation of node n. Shaded regions refer to the zero values, both in $\mathbf{x}_n$ and $\mathbf{W}$.

Learning the Embeddings

We now describe two possible ways for learning the embedding matrix $\mathbf{W}$. We first consider exploratory network analysis, where the aim is to learn representations that capture general graph properties, and then consider classification tasks, where the objective is to identify attributes useful to discriminate between the classes.

Unsupervised Setting The objective here is to capture the global network features in terms of the local structures surrounding every node. We follow the approach of skip-gram based methods [4, 15], which optimize a likelihood function from sample data generated through random walks on the graph. Our approach is composed of two distinct methods:

Distributional Sampling through Random Walks— First, we sample random walks over the graph, selecting the next node uniformly from the neighbours of the current node. Figure 3 illustrates a possible random walk of length 9 in the graph of Figure 1. Randomly sampling the graph structure, produces traversed node sequences which become the input for the following negative sampling optimization objective.

Negative Sampling Optimization Objective— Given a random walk ω, defined as a sequence of nodes of length s, we define the context $\mathcal{C}(n, \omega)$ associated to the occurrence of a node n in ω as the the sub-sequence in ω containing the nodes that appear close to n, including repetitions. Closeness is determined by the hyper-parameter p, the size of the positive context window, i.e., the context contains all nodes that appear at most p steps before/after the node within ω.

Like DeepWalk, we define a skip-gram negative sampling objective that learns to identify nodes appearing in similar contexts within random walks. Given a node $n_t \in V$ in random walk ω, our task is to learn embeddings that assign high probability for nodes $n_o \in V$ appearing in the context $\mathcal{C}(n_t, \omega)$, and lower probability for nodes not appearing in the context. As we focus on the learned representation capturing these symmetric relationships, the probability of n_o being in $\mathcal{C}(n_t, \omega)$ is given by:

$$p\left(n_o \in \mathcal{C}(n_t, \omega)\right) = \sigma(\mathbf{e}_{n_t}^\top \cdot \mathbf{e}_{n_o}), \tag{2}$$

where $\sigma(\cdot)$ is the logistic function.

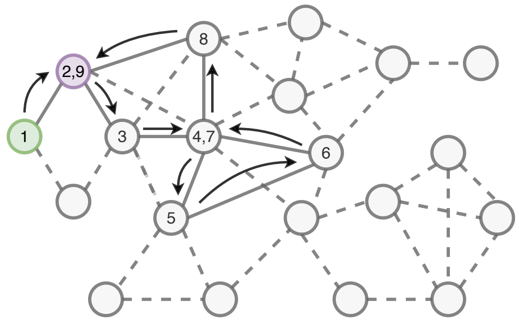


Figure 3: Example of random walk. Nodes contain the timestep when they were visited. The context of nodes seen besides a target node across the walk serves to learn the representations given by the approach shown in Figure 1.

Our global objective function is the following negative sampling log-likelihood. For each random walk and each node in the random walk, we sum the term corresponding to the positive cases for structures found in the context, and the expectation over z negative, randomly sampled, structures:

$$\mathcal{L}_{\text{u}}(\mathbf{W}) = \sum_{j=1}^{w} \sum_{\substack{n_t \in \omega_j, \\ n_o \in \mathcal{C}(n_t, \omega_j)}} \Bigg[\log \sigma(\mathbf{e}_{n_t}^\top \cdot \mathbf{e}_{n_o}) + \sum_{i=1}^{z} \mathbb{E}_{n_i \sim P_n(V)} \Big[\log \sigma(-\mathbf{e}_{n_t}^\top \cdot \mathbf{e}_{n_i}) \Big] \Bigg], \quad (3)$$

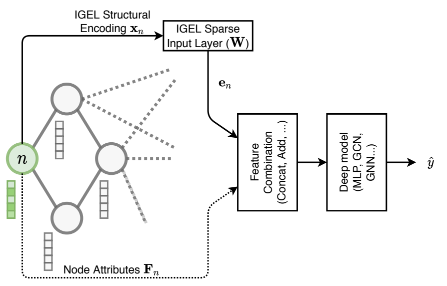
where $P_n(V)$ is the noise distribution from which the z negative samples are drawn. We maximise (3) through gradient ascent. Table 1 summarizes the hyper-parameters.

Table 1: Hyper-Parameters used in unsupervised tasks.

Hyper-Parameter	**Description**
$w \in \mathbb{N}$	Number of random walks per node
$s \in \mathbb{N}$	Number of steps in the Random Walks
$z \in \mathbb{N}$	Number of negative examples
$p \in \mathbb{N}$	Positive samples context window size

Supervised Setting IGEL can also be used as a node embedding layer in a deep neural network. The embedding of Eq. (1) can be fed into a downstream model as additional input features for a multilayer perceptron (MLP) or a graph neural network (GNN). In this case, the weight matrix $\mathbf{W}$ is learned with the rest of the model parameters $\mathbf{\Theta}$. Instead of capturing structural properties through random walks, the model in the supervised setting captures salient structural features that have predictive power in a particular task.

Figure 4 illustrates the general model for a standard node classification task, where IGEL embeddings are used in combination with possibly additional node attributes $\mathbf{F}_n$ to generate an output $\hat{y}$ used to predict the node class label.



IGEL Structural Encoding $\mathbf{x}_n$
IGEL Sparse Input Layer ($\mathbf{W}$)
$\mathbf{e}_n$
n
Feature Combination (Concat, Add, ...)
Deep model (MLP, GCN, GNN...)
$\hat{y}$
Node Attributes $\mathbf{F}_n$

Figure 4: An example of IGEL applied in a supervised setting. The node embedding $\mathbf{e}_n$ provides a structural representation that can optionally be combined with additional node or edge features as an input to a deep learning model which outputs $\tilde{y}$. The weight matrix $\mathbf{W}$ is jointly learned with the rest of the parameters $\mathbf{\Theta}$. See text for details.

Using IGEL as an embedding layer allows for pre-training and fine-tuning when not enough training data is available. The IGEL layer can first be trained in the unsupervised manner, and subsequently fine-tuned to a specific inference task.

Without loss of generality, we focus on multi-label classification. For M independent labels, we minimise the binary cross-entropy of the predicted labels associated to nodes:

$$\mathcal{L}_{\text{s}}(\mathbf{W}, \mathbf{\Theta}) = -\sum_{n \in V} \sum_{i=1}^{M} \Big[y_n^i \log \sigma(\hat{y}_n^i) + (1 - y_n^i) \log \big(1 - \sigma(\hat{y}_n^i)\big) \Big], \quad (4)$$

where y_n^i is one if i is the true class label of node n, and zero otherwise, and $\hat{y}_n^i$ is the network output corresponding to the class with label i.

Algorithmic Complexity

Finally, we describe the computational complexity of IGEL in relation to other methods. We focus on time and parameter complexities as described in [3, 22]. We differentiate time complexity, expressed in terms of computational steps needed to train given an input graph, and parameter complexity, which refers to the number of learnable parameters in a model. We ignore parameters that act as a constant for either complexity, such as the number of random walks or the number of negative samples.

We first consider the *parameter complexity*. Let γ be the number of bins required to encode the local degree-based representation, which we set to the maximum degree δ_{max} in this work[1]. Since each embedding vector has dimension d and we have γ bins for each encoding distance, the number of parameters scales as $\mathcal{O}(\alpha \times \gamma \times d)$.

On the other hand, the *time complexity* of IGEL is given by two steps: encoding and optimization. The encoding step requires to visit all nodes at distance α from each node n, and repeat that for each node in the graph. Time complexity is therefore exponential in α, $\mathcal{O}(|V| \times \delta_{\text{max}}^{\alpha})$. Once local representations are computed, the embedding step can be done efficiently, involving a linear sparse product as a forward-then-backward-pass in a neural network. Table 2 compares the time and parameter complexities of several models.

We find that IGEL's complexity lies between transductive and inductive methods. Its parameter complexity aligns with node embedding methods, bounded by the maximum degree in the graph. In contrast, IGEL's time complexity depends on the number of nodes and the size of their surrounding neighbourhoods, similar to sampling-based GCNs. However, IGEL lacks the fixed number of operations that is guaranteed by sampling a constant number of nodes. Thus, we expect IGEL's performance to degrade on denser graphs with neighbourhoods containing a wider range of different degrees among their nodes.

[1]For highly skewed degree distributions, our algorithm can be modified to require less parameters $\gamma \ll \delta_{\text{max}}$ by using log-sized bins or by exploiting the sparsity of the feature vectors.

Table 2: Comparison of graph representation methods in terms of parameter and time complexities. All but IGEL taken from [3, 22]. r denotes the sampling factor for sampling GCNs, ρ is the convolutional distance in GCNs, and d is the embedding size or the number of features, depending on the method.

Name	Time Compl.	Param. Compl.
DeepWalk, node2vec	$\mathcal{O}(\|V\|)$	$\mathcal{O}(\|V\|d)$
struct2vec*	$\mathcal{O}(\|V\|\log\|V\|)$	$\mathcal{O}(\|V\|d)$
GCN	$\mathcal{O}(\rho\|V\|\|E\|)$	$\mathcal{O}(\rho d^2)$
GraphSAGE	$\mathcal{O}(r^\rho\|V\|)$	$\mathcal{O}(\rho d^2)$
IGEL	$\mathcal{O}(\delta_{\text{max}}^{\alpha}\|V\|)$	$\mathcal{O}(\alpha\gamma d)$

*Struct2vec involves a $\mathcal{O}(|V|\log|V|)$ pre-processing step to generate the meta-graph and the embedding training process, equivalent to DeepWalk, $\mathcal{O}(|V|)$.

Experimental Evaluation

We evaluate IGEL in link prediction and node classification tasks which uses the IGEL encodings and using the graphs of Table 3. In Appendix A we additionally illustrate the unsupervised setting and show that IGEL is effective discovering non-trivial embeddings that are interpretable. Additional experimental details can be found in Appendix B.

Table 3: Overview of the graphs used in the experiments. $\bar{\mathbf{c}}$ is their average clustering coefficient.

	# Nodes	# Edges	$\bar{\mathbf{c}}$
Facebook [11]	4 039	88 234	0.52
ArXiv AstroPhysics [11]	18 772	198 050	0.32
PPI (train) [5]	44 906	613 184	0.12

Link Prediction (Unsupervised Embedding)

We follow the evaluation methodology proposed in [4] that allows us to compare with previous transductive algorithms. For a given graph, we generate negative examples (non-existing edges) by sampling random pairs of nodes that are not connected in the original graph. Positive examples (existing edges) are obtained by removing half of the edges while keeping the mutilated graph after the edge removals connected[2]. Both sets of node pairs are chosen to have the same cardinality. We first learn *unsupervised* IGEL embeddings on the mutilated graph using Equation (3), and subsequently treat link prediction as a binary classification task. We train a logistic regression classifier whose input is the Hadamard (element-wise) product of the frozen embedding vectors of nodes at each end of an edge. This representation is the best combination of features reported in [4]. Note that this setting can be seen as a special case of Eq. (4) with $M = 1$, a fixed $\mathbf{W}$ and the deep architecture replaced by a single layer of additional weights.

[2]The constraint of keeping the graph connected is only required to compare with transductive methods and is not necessary for IGEL.

Table 4: Area Under the ROC Curve (AUROC) results for the Facebook and AstroPhysics arXiv graphs. IGEL outperforms transductive methods as reported in [4] using only local degree structure.

Method	Facebook	arXiv
DeepWalk [15]	0.968	0.934
LINE [19]	0.949	0.890
node2vec [4]	0.968	0.937
IGEL ($\alpha = 2$)	**0.976**	**0.984**

We optimize all hyper-parameters described in Table 1 plus α using grid search. For link prediction, we target two disparate settings with a social network (Facebook ego-network), and a citation network (arXiv). Model performance depends strongly on two parameters: the encoding distance α and, to a lesser extent, the number of negative samples z. We find that $\alpha = 2$ yields the best performance on both graphs and observe a diminishing returns effect for larger values, e.g. results under $\alpha = 3$ are comparable to $\alpha = 2$ but training is an order-of-magnitude slower, in agreement with observations from [5]. Table 4 shows the results of the best IGEL configuration on five independent experiments outperforming previous transductive approaches.

Node Classification (Supervised)

We now evaluate how IGEL is able to capture structural attributes that generalize to unseen graphs from the same domain. For that, we consider an inductive multi-label node classification task, where we explore the performance of IGEL without unsupervised pre-training and fine-tuning. The objective is to predict 121 different binary labels capturing protein functions in a series of Protein-to-Protein Interaction (PPI) graphs. Aside from the graph structures, every node has 50 attributes, allowing us to evaluate how the performance of IGEL differs in attributed networks.

We use the supervised baselines proposed by GraphSAGE [5] and LCGL [2], and further compare with a fully supervised version of GraphSAGE [21]. We use randomized grid search to identify the best possible set of parameters for an multi-layer perceptron (MLP) containing IGEL and (optionally) feature inputs.

Table 5 shows the classification results in terms of micro-averaged F1 score. Our model using structural embeddings and node features significantly outperforms both LGCL and every GraphSAGE instantiation with $\alpha = 1$. We observe that a higher value of $\alpha = 2$ produces worse results than $\alpha = 1$. We attribute this to the fact that the unsupervised step involving random walks is not used in this fully supervised setting, which makes it more prone to overfitting. We believe that using regularization would help further improve these results. Remarkably, a model trained using only graph structural features at $\alpha = 1$ is able to outperform *every* GraphSAGE configuration that uses node/edge attributes and sampling. This highlights the strong predictive power of structural features alone, which to our knowledge has not been explored before in the presence of node or edges attributes.

Table 5: Micro-F1 scores for the multilabel classication task on the PPI graphs. IGEL combined with node features used as input for an MLP outperforms LGCL and supervised GraphSAGE both with and without sampling.

Method		**PPI**
Only Features (MLP, ours)		0.558
GraphSAGE-GCN[†]		0.500
GraphSAGE-mean[†]		0.598
GraphSAGE-LSTM[†]		0.612
GraphSAGE-pool[†]		0.600
GraphSAGE (no sampling)[‡]		0.768
LGCL[*]		0.772
IGEL ($\alpha = 1$)	Graph Only	0.736
	Graph + Feats	**0.850**
IGEL ($\alpha = 2$)	Graph Only	0.506
	Graph + Feats	0.741

Results as reported by †: [5] ‡: [21] $*$: [2]

Scalability Analysis

We finish with an empirical analysis of the empirical scalability of IGEL. For that, we evaluate our implementation in the unsupervised setting using synthetic data generated according to the Erdös-Renyi model. We control for three aspects that condition the runtime of the algorithm: the number of nodes $|V|$, the encoding distance α, and the average degree per node $\langle\delta\rangle$.

Figure 5 shows the run-time as a function of $|V|$ for different values of α, keeping $\langle\delta\rangle$ fixed. We observe that α does not significantly affect the cost in this setting. Furthermore all curves show an approximate linear scaling for $|V| \geq 1,024$ (note the log-scale in both axes), showing that IGEL can be applied to large-scale graphs in this regime.

We also analyze the influence of $\langle\delta\rangle$ for different values of α in Figure 6. For $\alpha = 1$, the runtime is almost constant as a function of $\langle\delta\rangle$, suggesting that the direct structural features

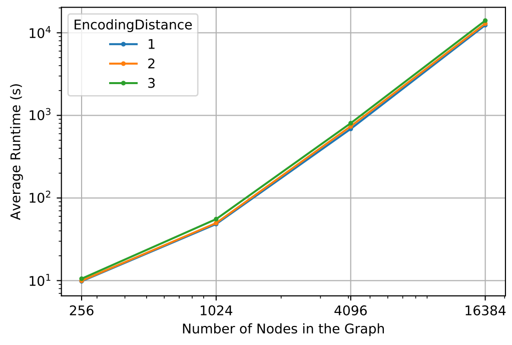


Figure 5: Log-log plot showing the average runtime as a function of the graph size $|V|$ for different values of encoding distance α with fixed average degree of $\langle\delta\rangle = 8$.

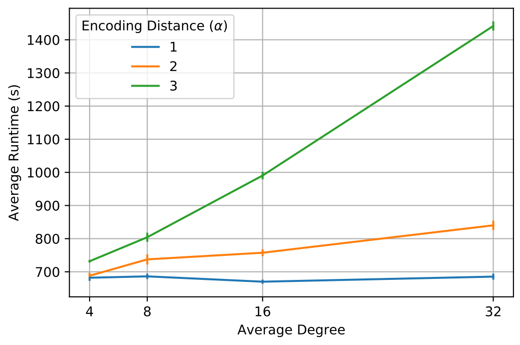


Figure 6: Average runtime as a function of the average degree for different values of α with fixed size of $|V| = 4,096$.

can be feasibly computed in very dense graphs. However, runtime grows linearly with $\langle\delta\rangle$ for values of $\alpha > 2$.

Overall, these results show that IGEL is a scalable method for values of $\alpha = 1, 2$. We believe that ideas from sampling-based GCN literature can be extended to purely structural methods like IGEL to allow for the encoding and learning of very distant structural features.

Conclusions

We have presented IGEL, a novel inductive network embedding method that compactly captures graph structures in a dense latent space [3]. To the best of our knowledge, IGEL is the first inductive embedding algorithm explicitly designed for unattributed graphs that captures structural features without the need for node attributes.

We show that IGEL is scalable and can compete with transductive approaches in tasks such as link prediction on unattributed graphs. Additionally, IGEL can inductively generalize to unseen graphs, outperforming models based on GCNs in a multi-label classification setting.

Our results highlight interesting research directions. First, it is not clear how explicit encodings like IGEL differ from GNNs trained over similar encodings as node features. Second, IGEL encodings do not account for evolution in the network, limiting their application to temporal graphs. Finally, IGEL encodings are not able to capture global graph structures.

Acknowledgements. We thank the anonymous reviewers for their comments and suggestions. The project received funding from “La Caixa” Foundation (ID 100010434), under the agreement LCF/PR/PR16 /51110009. Nurudin was supported by project 2018DI047 of the Catalan Industrial Doctorates Plan. Vicenç Gómez is supported by the Ramon y Cajal program RYC-2015-18878 (AEI/MINEICO/FSE,UE).

[3]Our Python code is available at `https://github.com/nur-ag/IGEL`.

References

[1] Arvind, V., Köbler, J., Rattan, G., Verbitsky, O.: Graph isomorphism, color refinement, and compactness. computational complexity **26**(3), 627–685 (2017)

[2] Gao, H., Wang, Z., Ji, S.: Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1416–1424 (2018)

[3] Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78 – 94 (2018)

[4] Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864 (2016)

[5] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems 30 (2017)

[6] Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications (2017), IEEE Data Engineering Bulletin, September 2017

[7] Kaltenbrunner, A., Aragón, P., Laniado, D., Volkovich, Y.: Not all paths lead to Rome: Analysing the network of sister cities. In: Proceedings of the 7th International Workshop on Self-Organizing Systems (2013)

[8] Kersting, K., Mladenov, M., Garnett, R., Grohe, M.: Power iterated color refinement. In: Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)

[9] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR (2017)

[10] Knuth, D.E.: The Stanford GraphBase: A Platform for Combinatorial Computing. Association for Computing Machinery, New York, NY, USA (1993)

[11] Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection (Jun 2014), `http://snap.stanford.edu/data`

[12] Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of the International Conference on Learning Representations (ICLR) (2013)

[13] Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (Feb 2004)

[14] Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: Proceedings of The 33rd International Conference on Machine Learning. vol. 48, pp. 2014–2023. New York, USA (2016)

[15] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 701–710 (2014)

[16] Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: Struc2vec: Learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 385–394. KDD '17, ACM (2017)

[17] Samanta, B., De, A., Jana, G., Gómez, V., Chattaraj, P., Ganguly, N., Gomez-Rodriguez, M.: NEVAE: A deep generative model for molecular graphs. Journal of Machine Learning Research **21**(114), 1–33 (2020)

[18] Shervashidze, N., Schweitzer, P., Jan, E., Leeuwen, V., Mehlhorn, K., Borgwardt, K.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **1**, 1–48 (01 2010)

[19] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on World Wide Web. pp. 1067–1077 (2015)

[20] Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference. pp. 539–548. WWW '18 (2018)

[21] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)

[22] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems pp. 1–21 (2020)

[23] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining. pp. 974–983 (2018)

[24] You, J., Ying, R., Leskovec, J.: Position-aware graph neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7134–7143. PMLR, Long Beach, California, USA (09–15 Jun 2019), `http://proceedings.mlr.press/v97/you19b.html`

Appendix A: Experiments in the Unsupervised Setting

We also evaluated IGEL in the unsupervised setting using networks that are amenable for visual analysis. We focus on analyzing how the encoding distance parameter α influences the learned representations.

Les Misérables Network— Our first analysis considers a variation of the original Les Misérables network [10], where two identical copies are connected by a single edge between two nodes chosen at random. This allows us understand whether or not IGEL can capture structural similarity within areas in the graph. Figure 7 shows the composite network with nodes colored based on their clusters. For $\alpha = 1$, the local encodings only consider direct neighbours. In this case, the resulting embeddings capture the standard notion of community or homophily, i.e., nodes with the same color are densely connected between each other and weakly connected to the rest. Furthermore, the embeddings corresponding to nodes within the two copies of each community also appear clustered together, although some nodes are not directly connected. This shows that IGEL embeddings can capture structural similarity, not only locally, but also from distant nodes in the graph. Larger encoding distances allow IGEL to capture relationships on larger portions of the graph. For $\alpha = 2$, shown in Figure 7(bottom), the embeddings are clustered in two major groups, separating frequently visited nodes—with higher degrees and closeness centrality, and hence more commonly appearing in the sampling process—from connecting bridges or edges.

We obtain our results on the Les Misérables Network running unsupervised IGEL for 5 epochs with a batch size of 50 000, an Adam learning rate of 0.01, and 200 steps per random walk. The positive context window size is 10, with a proportion of 10 negative samples per positive and 8-component embedding vectors. We direct the reader to the code repository, in which `configs/clustering.json` contains the exact hyperparameter configuration used.

To select the number of clusters used in Figure 7, we use modularity score [13]. Figure 8 shows that highest modularity is obtained consistently with 5 or 6 clusters for different values of the encoding distance α.

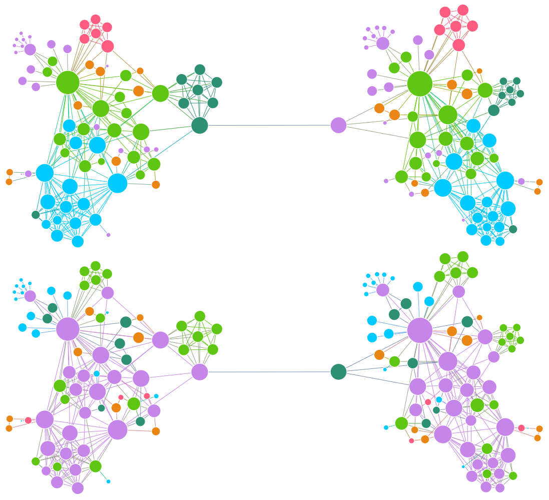


Figure 7: Visualization of the embeddings in the (cloned) Les Misérables network: node colors reflect the clustering of the embeddings obtained for two different encoding distances, $\alpha = 1$ (top) or $\alpha = 2$ (bottom). In addition to capture community structure, the IGEL embeddings also generalize structurally similar nodes between distant regions. Node sizes correspond to their harmonic closeness centrality score.

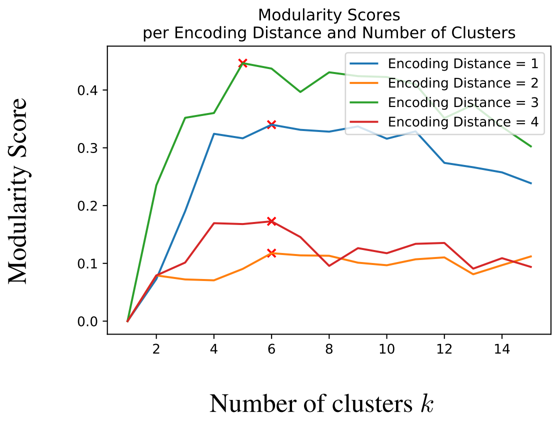


Figure 8: Modularity score for the different encoding distance values when compared with the total number of clusters. The red markings represent the scores of the best performing total communities for each encoding distance value.

The result of running a standard community detection method in the graph is shown in Figure 9, where we can observe that the local communities correspond to clustered embeddings found by IGEL for $\alpha = 1$. However, IGEL is able to capture the additional structural similarity due to the cloned nature of the two subgraphs.

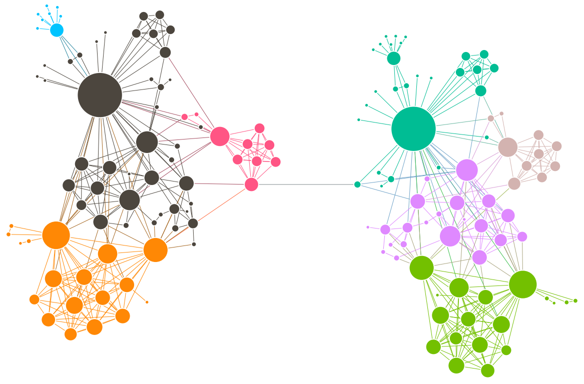


Figure 9: Node clusters produced by the Louvain Community Detection method. In contrast with IGEL, Louvain directly optimizes modularity over the whole graph. This noticeably makes the clusters across copies of the graph different, whereas IGEL can only identify the same representations given the available structural information in the neighbourhood of a node.

Sister Cities Network— We further detail our process illustrate how IGEL can be used for knowledge discovery using a larger network: the network of Sister Cities [7], a graph mined from Wikipedia where nodes correspond to cities and links to partnerships. We follow a similar approach as before and train IGEL embeddings for each node. We then perform a clustering analysis of the embeddings.

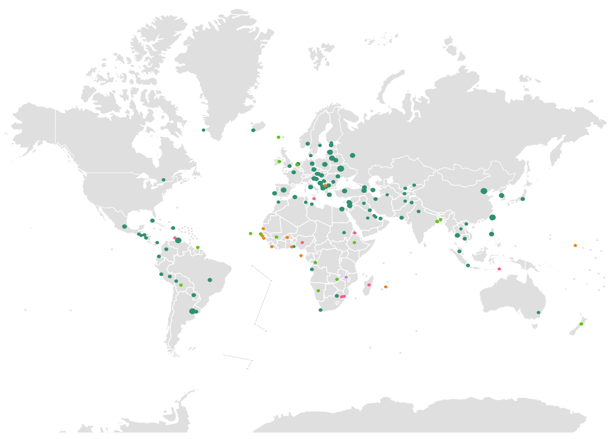


Figure 10: Geo-plot of the learned clusters on the Sister Cities dataset, filtering only cities that are capitals. Each point is a city, with a radius proportional to its degree, and colored by the group assigned by the clustering model. To learn the clusters, we use an IGEL encoding distance $\alpha = 4$ and rum KMeans with $k = 6$.

The obtained clusterings allow us to identify interpretable node roles, e.g., cities that are country capitals. To quantitatively test this, we use an external labeled dataset that assigns a total of 179 nodes as country capitals, and then evaluate IGEL as a binary classifier by treating the nodes that belong to the cluster containing the highest number of capital cities as positive, and the rest of the nodes as negative.

Without supervision, we evaluate whether the unsupervised embeddings can capture country capitals directly from the data using unsupervised K-means clustering. Figure 10 shows a geo-plot with label colors corresponding to a subset of 872 capital cities. Figure 11 shows all the learned clusters over the same geo-plot. Remarkably, the cluster composed of IGEL embeddings consistently outperforms heuristic methods by capturing up to 28% more country capitals, as measured by F1-score (see Table 6 for details). From these results, we can conclude that IGEL is effective discovering non-trivial embeddings that are interpretable.

We obtain our results on the Sister Cities Network running unsupervised IGEL for 2 epochs with a batch size of 25000, an Adam learning rate of 0.01, and 30 steps per random walk. The positive context window size is 7, with a proportion of 10 negative samples per positive and 30-component embedding vectors. We direct the reader to the code repository, in which `configs/clusteringLarge.json` contains the exact hyperparameter configuration used to train the embeddings.

Analysis of Capitals—To compare the clustering model, we set a baseline with a heuristic approach based on node degree and PageRank score. We rank the nodes by either metric and take as many nodes as there are in the positive cluster. We then compute F1-scores for the cluster, degree and PageRank approaches, as shown in Table 6.

Upon qualitative inspection, we additionally notice that the clusters associated with capitals usually include large,

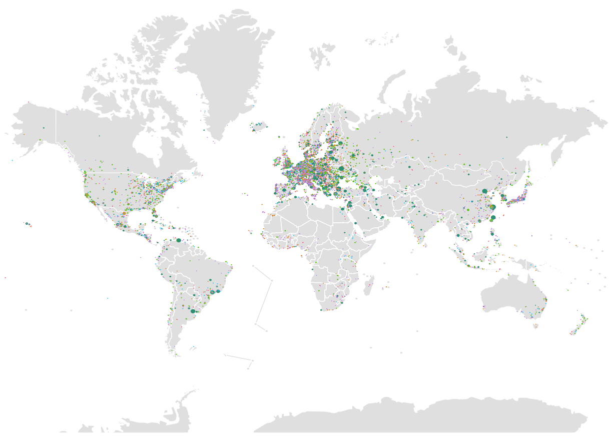


Figure 11: Clusters of Sister Cities found using IGEL embeddings with $\alpha = 4$ and $k = 6$.

densely populated cities. Additionally, we found that the model often discriminates against capital cities located in Africa. This may be caused by a reduced coverage of sister relationships in Wikipedia, or from the lack of such links between African cities and the rest of the world.

Table 6: Comparison between F1-scores of the cluster containing the most capitals C and the result taking the top $|C|$ nodes as ranked by degree (Deg.) or PageRank (PR). **Bold** means that the configuration is the best in a row, *italic* that it is the best result for the encoding distance in that row.

α	k	$\lvert C\rvert$	C **F1**	**Top-$\lvert C\rvert$ Deg. F1**	**Top-$\lvert C\rvert$ PR F1**	**%vs Deg. F1**
1	5	854	**0.197**	0.174	0.070	13.32
	6	848	**0.201**	0.175	0.070	14.44
	7	839	***0.202***	0.177	0.071	14.42
2	5	578	**0.248**	0.198	0.095	25.34
	6	566	***0.258***	0.201	0.097	28.02
	7	568	**0.257**	0.201	0.096	27.99
3	5	1 892	**0.118**	0.116	0.075	1.73
	6	886	***0.195***	0.178	0.068	9.47
	7	1 360	**0.144**	0.140	0.070	2.78
4	5	2 181	**0.108**	0.105	0.074	2.48
	6	872	***0.196***	0.173	0.069	13.23
	7	1 224	0.141	**0.151**	0.068	-6.62

Correlation with node indicators— Finally, we analyse how the embeddings themselves relate to traditional graph metrics. For every node, we compute a score given only the embeddings of nodes in the graph, and then study its correlation to PageRank, Closeness, Betweenness and Degree centralities. To do so, we start with the intuition that a highly central node should be highly similar to its neighbours, since its representation should minimise the distance with any of the surrounding nodes. With this in mind, we compute the per-node scores as the sum of similarity scores between the node embedding and the embeddings of every one of its neighbours. Finally, our similarity function is the dot product between embedding vectors.

Since we are comparing distributions that are defined over different domains, such as probabilities in the case of PageRank or integers in the case of degree, we focus on the rank rather than the actual values between scores and metrics. Thus, we use Spearman's ρ correlation statistic to measure the monotonicity between both. The results for different encoding distances are shown in Table 7.

Table 7: Spearman Correlations (with all $p < 10^{-20}$) between the node-to-graph self-similarity scores and graph centrality metrics.

α	**PageRank**	**Betweenness**	**Closeness**	**Degree**
1	0.754	0.791	0.319	0.899
2	0.740	0.842	0.299	0.665
3	0.772	0.769	0.284	0.907
4	0.776	0.769	0.281	0.903

The correlation between embedding scores and results show several patterns. First, the embedding scores between a node and its neighbours is highly correlated with Degree, PageRank and Betweenness centralities. Second, closeness centrality is the least correlated of the metrics, as it measures how near a node is to every other node, in effect capturing a global property that needs to look beyond immediate node neighbourhoods. This falls in line with the hypothesis that we discussed in the previous section, in which larger graphs should display scores that are less correlated with global metrics. We direct the reader to Table 8 for the correlations in the Les Misérables graph, which exhibit higher correlations with the metrics across the board, and more so with closeness in particular.

Appendix B: Additional settings and results

In this section we summarise the hyperparameters, hardware and experimental details for the Link Prediction, Node Classificaton and Scalability experiments. We direct the reader to the `configs/replication/` folder in the code repository for the exact configurations used to run the experiments.

Link Prediction– The best performing hyperparameter configuration on the Facebook graph including $\alpha = 2$, learning $d = 256$ component vectors with $e = 10$ walks per node, each of length $s = 150$ and $p = 8$ negative samples per positive for the unsupervised negative sampling. Respectively on the arXiv citation graph, we find the best configuration at $\alpha = 2$, $d = 256$, $e = 2$, $s = 100$ and $p = 9$. Finally, we note that we run each experiment on a machine with 8 cores, 16GB of memory and a single GPU with 8GB of memory.

Node Classification– In the node classification experiment, we analyze both encoding distances $\alpha \in \{1, 2\}$. Other IGEL hyper-parameters are fixed after a small greedy search based on the best configurations in the link prediction experiments. For the MLP model, we perform a greedy architecture search, including the number of hidden units, activation functions and depth. Our results show scores averaged over five different seeded runs with the same configuration, as obtained from the parameter search. We run each experiment on a machine with 8 cores, 16GB of memory and a single GPU with 8GB of device memory.

Table 8: Spearman Correlations (with $p < 10^{-3}$) between the node-to-graph self-similarity scores and graph centrality metrics on the Les Miserables cloned graph.

α	**PageRank**	**Betweenness**	**Closeness**	**Degree**
1	0.728	0.355	0.246	0.719
2	0.964	0.760	0.590	0.974
3	0.889	0.550	0.459	0.872
4	0.973	0.758	0.598	0.982

For reproducibility, we note that the best performing hyperparameter configuration on the node classification is found with $\alpha = 2$ on $d = 256$ length embedding vectors, concatenated with node features as the input layer for 1000 epochs in a 3-layer MLP using ELU activations with a learning rate of 0.005. Additionally, we apply 100 epoch patience for early stopping, monitoring the F1-score on the validation set.

Scalability experiments– The training process in the scalability experiments is performed with a realistic parameter configuration similar to the one used in the link prediction task, with encoding distances $\alpha = 1, 2, 3$. For each training configuration, we execute five independent experiments. Each run is performed on a Slurm-based cluster requesting 48 CPU cores and 200GB of memory on each setting. We record the overall training times, and analyze the run-time as a function of the size of the graph and the average degree.