# Graph Representation Ensemble Learning

by

Palash Goyal, Sachin Raja, Di Huang, sujit Rokka Chhetri, Arquimedes Canedo, Ajoy Mondal, Jaya shree, C V Jawahar

in

*IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*
: 1
-8

Report No: IIIT/TR/2020/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2020

2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

# Graph Representation Ensemble Learning

Palash Goyal*
*University of Southern California*
palashgo@usc.edu

Sachin Raja*
*IIIT-Hyderabad*
sachinraja13@gmail.com

Di Huang*
*University of Southern California*
dh_599@usc.edu

Sujit Rokka Chhetri*
*University of California-Irvine*
schhetri@uci.edu

Arquimedes Canedo*
*Siemens Corporate Technology*
arquimedes.canedo@siemens.com

Ajoy Mondal
*IIIT-Hyderabad*
ajoy.mondal@iiit.ac.in

Jaya Shree
*University of Southern California*
shree@usc.edu

CV Jawahar
*IIIT-Hyderabad*
jawahar@iiit.ac.in

***Abstract*—Representation learning on graphs has been gaining attention due to its wide applicability in predicting missing links and classifying and recommending nodes. Most embedding methods aim to preserve specific properties of the original graph in the low dimensional space. However, real-world graphs have a combination of several features that are difficult to characterize and capture by a single approach. In this work, we introduce the problem of graph representation ensemble learning and provide a first of its kind framework to aggregate multiple graph embedding methods efficiently. We provide analysis of our framework and analyze – theoretically and empirically – the dependence between state-of-the-art embedding methods. We test our models on the node classification task on four real-world graphs and show that proposed ensemble approaches can outperform the state-of-the-art methods by up to 20% on macro-F1. We further show that the strategy is even more beneficial for underrepresented classes with an improvement of up to 40%.**

***Index Terms*—Graph Representation, Node Embedding, Ensemble Learning, Greedy Search, Node Classification, Distance Correlation, Prediction Diversity**

## I. Introduction

Graphs are used to represent data in various scientific fields, including social sciences, biology, and physics [1]–[4]. Such representation allows researchers to gain insights about their problems. The most common tasks on graphs are link prediction, node classification, and visualization. For example, link prediction in the social domain is used to determine friendships between people. Node classification in the biology domain is used to identify genes of proteins. Similarly, visualization is used to identify communities and the structure of a graph. Recently, a significant amount of work has been devoted to learning low dimensional representation of nodes in the graphs to allow the use of machine learning techniques to perform the tasks on graphs. Graph representation learning techniques embed each node in the network in low dimensional space and map link prediction and node classification in the network space to the nearest neighbor search and vector classification in the embedding space [5]. Several of these techniques have shown state-of-the-art performance on graph tasks [6], [7].

State-of-the-art techniques in graph representation learning define some characteristics of the graphs. They aim to capture and define an objective function to learn these features in the low-dimensional embedding. For example, HOPE [7] preserves higher-order proximity between nodes using the singular value decomposition of the similarity matrix. Similarly, Node2Vec [6] captures the similarity of nodes using random walks on the graph. However, real-world graphs do not follow a simple structure and can be layered with several categories of properties with complex interactions between them. It has been shown that no single method outperforms other methods on all network tasks and datasets [5]. We further illustrate this by the example in Figure 1 with a social network from two classrooms (represented by the pink color). We also show the family links of individual students in the classroom and represent family members outside the classroom (represented by the blue color). Here, we consider the task of multi-label node classification with the classes classroom and role in the family. This network is complex and has both community and structural properties. Methods such as HOPE [7], which preserve community, can effectively classify the nodes into classrooms but perform poorly on family links that follow structure. On the other hand, structure preserving methods can classify the role of an individual student in the family but puts nodes in the same classroom into separate categories.

In this work, we introduce graph representation ensemble learning. Given a graph and a list of methods capturing various

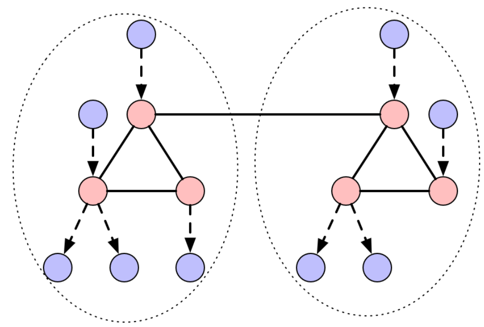


Fig. 1: It illustrates the example motivating the need for ensemble learning. The graph represents interactions in classrooms (in red) and family trees of students (in purple). Such a complex combination of community and structure requires a combination of multiple embedding methods for accuracy.

*These authors contributed equally to this work.

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 IEEE

properties of the graph, we aim to learn a representation of nodes which can combine embeddings from each method such that it outperforms each of the constituent methods in terms of prediction performance. We also show, through our experiments, that ensembling embedding methods by combining embeddings from individual methods outperform a standard way of ensembling using majority voting. Ensemble methods have been very successful in the field of machine learning. Methods such as AdaBoost [8] and Random Forest [9] have shown to be much more accurate than the individual classifiers that compose them. It has been shown that combining even the simplest but diverse classifiers can yield high performance. However, to the best of our knowledge, no work has focused on ensemble learning on graph representation learning.

Here, we formally introduce ensemble learning on graph representation methods and provide a framework for it. We first give a motivation example to show that a single embedding approach is not enough for accurate predictions on a graph task, and combining methods can yield improvement in performance. We then formalize the problem and define a method to measure the correlations of embeddings obtained from various approaches. Then, we provide an upper bound on the correlation assuming certain properties of the graph. The upper bound is used to establish the utility of our framework.

We focus our experiments on the task of node classification. We compare our method with state-of-the-art embedding methods and show its performance on four real-world networks, including collaboration networks, social networks, and biological networks. Our experiments show that the proposed ensemble approaches outperform the state-of-the-art methods by 20% on macro-F1. We further show that the approach is even more beneficial for underrepresented classes and get an improvement of 40%.

Overall, our paper makes the following contributions:

1) We introduce ensemble learning in the field of graph representation learning.
2) We propose a framework for ensemble learning on given a variety of graph embedding methods.
3) We provide a theoretical analysis of the proposed framework and show its utility theoretically and empirically.
4) We demonstrate that combining multiple diverse methods through ensemble achieves state-of-the-art accuracy and outperforms majority voting strategy to the ensemble for node classification.

## II. Related Work

Methods for graph representation learning (aka graph embedding) typically vary in properties preserved by the approach and the objective function used to capture these properties. Based on the properties, embedding methods can be divided into two broad categories: (i) community preserving, and (ii) structure preserving. Community preserving approaches aim to capture the distances in the original graph in the embedding space. Within this category, methods vary on the level of distances captured. For example, Graph Factorization [10] and Laplacian Eigenmaps [11] preserve shorter distances (i.e., low order proximity) in the graph, whereas more recent methods such as Higher Order Proximity Embedding (HOPE) [7] and GraRep [12] capture longer distances (i.e., high order proximity). Structure preserving methods aim to understand the structural similarity between nodes and capture the role of each node. Node2Vec [6] uses a mixture of breadth first and depth first search for this. Deep learning methods such as Structural Deep Network Embedding (SDNE) [13] and Deep Network Graph Representation (DNGR) [12] use deep autoencoders to preserve distance and structure.

Based on the objective function, embedding methods can be broadly divided into two categories: (i) matrix factorization, and (ii) deep learning methods. Matrix factorization techniques represent a graph as a similarity matrix and decompose it to get the embedding. Graph Factorization and HOPE use adjacency matrix and higher order proximity matrix for this. Deep learning methods, on the other hand, use multiple non-linear layers to capture the underlying manifold of the interactions between nodes. SDNE, DNGR, and VGAE [14] are examples of these methods. Some other recent approaches use graph convolutional networks to learn graph structure [15]–[17]. As an example, Geometric GCN [18] maps the graph to a continuous latent space using node embedding and then uses the geometric relationships defined in the latent space to build structural neighborhoods for aggregation. Some more recent methods suggested augmenting deep networks with attention mechanism. One such method is Graph Attention Networks [19], which uses a novel attention model on the power series of the transition matrix, which guides the random walk to optimize an upstream objective.

In machine learning, ensemble approaches [20] are algorithms that combine the outputs of a set of classifiers. It has been shown that the ensemble of classifiers are more accurate than any of its members if the classifiers are accurate and diverse [21]. There are several ways individual classifiers can be combined. Broadly, they can be divided into four categories: (i) Bayesian voting, (ii) random selection of training examples, (iii) random selection of input features, and (iv) random selection of output labels. Bayesian voting methods combine the predictions from the classifiers weighted by their confidence. On the other hand, methods such as Random Forest [9] and Adaboost [8] divide the training data into multiple subsets, train classifiers on each subset, and combine the output. The third category of approaches divides the input set of features available to the learning algorithm [22]. Finally, for data with a large number of output labels, some methods divide the set of output labels and learn individual classifiers to learn their corresponding label subset [23].

In this work, we extend the concept of ensemble learning to graph representation learning and get insights into the correlations between various graph embedding methods. Based on this, we propose ensemble methods for them and show the improvement in performance on node classification task.

## III. Motivating Example

This section presents a motivational case study to highlight the effectiveness of the proposed graph representation ensemble learning on a synthetic dataset. We present the analysis by utilizing four synthetically generated graphs: (a) Barabasi-Albert, (b) Random Geometry (c) Stochastic Block Model, and (d) Watts Strogatz graph (see Figure 2). Each of these graphs exhibits a specific structural property. We use a spring layout to further elucidate the difference in the structural properties of the four different synthetic graphs. The Barabasi-Albert graph makes new connections through preferential attachment using the degree of the existing nodes. Watts Strogatz graph generates a ring of $n$ graphs with the addition of edges of each node with its $k$ neighbors. Stochastic Block Model creates community clusters by preserving the community structure. The Random Geometry graph generates $n$ nodes and adds $m$ edges by utilizing the spatial proximity among the nodes as a measure.

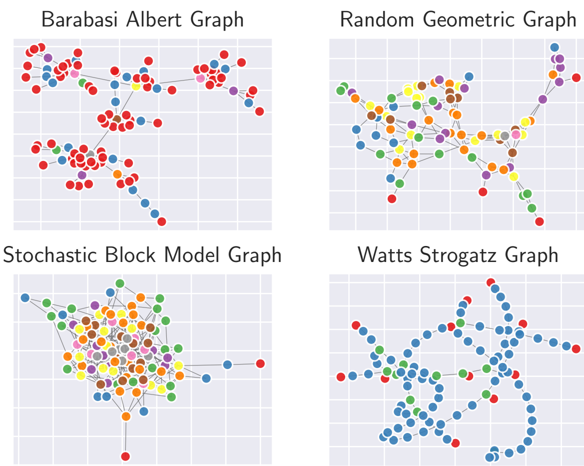


Fig. 2: Four synthetic graph with different graph properties (with node color representing different degrees) initially drawn using the spring layout.

We have generated each of the synthetic graphs with 100 nodes each. As mentioned earlier, different embedding algorithms such as Graph Factorization, Laplacian Eigenmaps, High Order Proximity Preserving, Structural Deep Network Embedding, Node2Vec, Geometric Graph Convolution Networks, Graph Autoencoders, and Graph Attention Learning Networks capture various characteristics of the graphs. Hence, a single embedding algorithm may not be able to capture the entire complex interaction. To test this hypothesis, we have created two node labels for the synthetic graph. The first label is based on the degree of the graph, whereas the second label is based on the closeness centrality measure [24] of the graph. The centrality values are binned, and the respective bins are used as node labels.

To simulate the interaction between different synthetic graphs, we have randomly selected node pairs (equal to 40% of the total number of nodes) and added edges between them

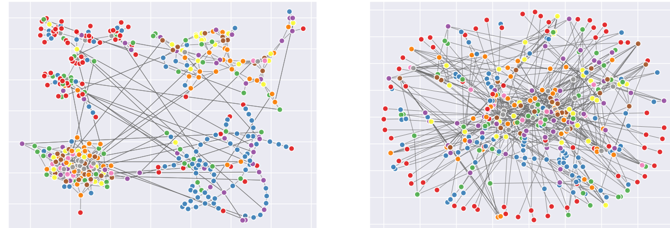

Fig. 3: It illustrates the merge graph of four synthetically generated graphs using two approaches(different colors represent the updated node degree). **Left graph:** shows the original layout by adding edges of four graphs. **Right graph:** shows a new layout by adding four graphs using Spring techniques.

(with a probability threshold of 0.3). The addition of the edges are shown in Figure 3.

| **Method** | **Macro-F1 deg↑** | **Macro-F1 cent↑** |
|---|---|---|
| GF | 0.052 | 0.137 |
| LAP | <u>0.221</u> | 0.157 |
| HOPE | 0.172 | 0.163 |
| SDNE | 0.136 | <u>0.216</u> |
| Node2Vec | 0.203 | 0.186 |
| Graph AE | 0.175 | 0.141 |
| Geom GCN | 0.194 | 0.174 |
| Graph Attn | 0.113 | 0.114 |
| Majority Vote All | **0.235 (6.33%)** | **0.221 (2.31%)** |
| Optimal Concatenation | **0.252 (14.02%)** | **0.241 (11.57%)** |

TABLE I: Ensemble performance on motivating example.

The result of the node classification for the degree labels of the merged synthetic graph is shown in column 2 of Table I. The embedding obtained from the state-of-the-art methods and the ensemble approach is utilized to predict the degree labels. It can be seen that compared to the state-of-the-art algorithms, the ensemble based approach is able to achieve 14.02% improvement in macro F1 score. It is a significant improvement in the classification accuracy as compared to all individual methods independently and a majority voting ensembling scheme of all methods together.

The classification accuracy results for classifying the centrality measures are shown in column 3 of Table I. For this label, it can be observed that the ensemble based method can achieve 11.57% improvement in macro F1-score. Both the macro F1-score proves that the ensemble based approach can utilize the best characteristic of different graph embedding algorithm's ability to capture the structure of the network.

## IV. Graph Representation Ensemble Learning

In this section, we define the notations and provide the graph ensemble problem statement. We then explain multiple variations of deep learning models capable of capturing temporal patterns in dynamic graphs. Finally, we design the loss functions and optimization approach.

### A. Notations

We define a directed graph as $G = (V, E)$, where $V$ is the vertex set, and E is the directed edge set. The adjacency matrix is denoted as $A$. We define the embedding matrix from a method $m$ as $X^m$. The embedding matrix can be used to reconstruct the distance between all pairwise nodes in the graph. We denote this as $S^m$, in which $S^m_{i,j} = \|X^m_{i,.} - X^m_{j,.}\|$.

### B. Problem Statement

In this paper, we introduce the problem of ensemble learning on graph representation learning. We define it as follows: *Given a set of embedding methods $\{m_1, \ldots, m_k\}$ with corresponding embeddings for a graph $G$ as $\{X^{m_1}, \ldots, X^{m_k}\}$ and errors $\{\epsilon_1, \ldots, \epsilon_k\}$ on a graph task $\mathcal{T}$, a graph ensemble learning approach aims to learn an embedding $X^m$ with error $\epsilon$ such that $\epsilon < min(\epsilon_1, \ldots, \epsilon_k)$.*

### C. Measuring Graph Embedding Diversity

Different graph embedding techniques vary in the types of properties of the graphs preserved by them and the model defined. Broadly, embedding techniques can be divided into (i) structure preserving, and (ii) community preserving models, defined as follows:

**Definition 1.** (Community Preserving Models): It aims to embed nodes with the lower distance between them closer in the embedding space.

**Definition 2.** (Structure Preserving Models): It aims to embed structurally similar nodes closer in the embedding space.

As the ensemble accuracy of a combination of methods depends on the diversity of the input methods [25], we now establish bounds on the diversity of embedding models. Graph embedding of a graph $G$ is a matrix $X \in \mathbb{R}^{n \times d}$ where $n$ is the number of nodes, and $d$ is the dimension of the embedding. Thus, we require a diversity measure that can quantify diversity between matrices. Pearson correlation [26] is a popular metric traditionally used to measure diversity of two univariate random variables. It can be generalized to a multivariate case and defined as RV coefficient [27].

As RV Coefficient measures linear dependence between the variables and embedding methods can be non-linear in construction, we can use a distance based metric to capture such non-linearity between embeddings:

**Definition 3.** [28] (Distance Covariance): Suppose that $X$ and $Y$ are matrices of centered random vectors (column vectors). Let the $n \times n$ distance matrices $(a_{j,k})$ and $(bj, k)$ containing all pairwise distances, $a_{j,k} = \|X_j - X_k\|$ and $b_{j,k} = \|Y_j - Y_k\|$. We compute the doubly centered distance matrices $(A_{j,k})$ and $(B_{j,k})$, where $A_{j,k} = a_{j,k} - a_{j,.} - a_{.,k} + a_{.,.}$ and $B_{j,k} = b_{j,k} - b_{j,.} - b_{.,k} + b_{.,.}$. The distance covariance is defined as follows:

$$dCov^2(X, Y) = \frac{1}{n^2} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{j,k} B_{j,k}.$$

**Definition 4.** [29] (Distance Correlation): The distance correlation between random variables $X$ and $Y$ is given as follows:

$$dCor(X, Y) = \frac{dCov(X, Y)}{\sqrt{dCov(X, X) dCov(Y, Y)}}$$

Based on this, we obtain the following bound:

**Theorem 1.** *Consider two embedding methods $m_1$ and $m_2$ with corresponding embeddings for a graph $G = (V, E)$ as $X^{m_1}$ and $X^{m_2}$, where $|V| = n$. Let $G$ have a set $V_1$ of structurally similar nodes with $|V_1| = n_1$ and a set $V_2 = V \setminus V_1$ with nodes in multiple communities. If $m_1$ is a purely structural preserving method and $m_2$ preserves both structural and community properties, then distance correlation between the the embeddings has the following bound:*

$$dCor(X^{m_1}, X^{m_2}) < 1 - \frac{n_1}{n}.$$

*Proof.* Let $S^{m_1}$ and $S^{m_2}$ denote the pairwise distance matrices for methods $m_1$ and $m_2$, and $S'^{m_1}$ and $S'^{m_2}$ denote their doubly centered versions. We now have,

$$dCov(X^{m_1}, X^{m_2}) = \frac{1}{n^2} \sum_{v,w \in V} S'^{m_1}_{v,w} S'^{m_2}_{v,w} \tag{1}$$

$$dCov(X^{m_1}, X^{m_1}) = \frac{1}{n^2} \sum_{v,w \in V} (S'^{m_1}_{v,w})^2 \tag{2}$$

We can divide the first summation (eqn. 1) into four parts:

$$\sum_{v,w \in V} S'^{m_1}_{v,w} S'^{m_2}_{v,w} = \sum_{v,w \in V_1} S'^{m_1}_{v,w} S'^{m_2}_{v,w} + \sum_{v \in V_1, w \in V_2} S'^{m_1}_{v,w} S'^{m_2}_{v,w} + \sum_{v \in V_2, w \in V_1} S'^{m_1}_{v,w} S'^{m_2}_{v,w} + \sum_{v,w \in V_2} S'^{m_1}_{v,w} S'^{m_2}_{v,w}$$

As $m_2$ preserves structural similarity, the distance between each pair of nodes in set $V_1$ will be 0 yielding the first term of above equation 0. Also, since $V_1$ and $V_2$ do not have a specified relation, the embedding distances by $m_1$ and $m_2$ will be randomly distributed and uncorrelated. Thus, the second and third terms become 0. We can get similar results for second summation (eqn. 2) as well. From this, we get

$$dCor(X^{m_1}, X^{m_2}) = \frac{\sum_{v,w \in V_2} S'^{m_1}_{v,w} S'^{m_2}_{v,w}}{\sqrt{\sum_{v,w \in V_2} (S'^{m_1}_{v,w})^2} \sqrt{\sum_{v,w \in V_2} (S'^{m_2}_{v,w})^2}}$$

As correlation between two variables is bounded by 1, from the above we get

$$dCor(X^{m_1}, X^{m_2}) \leq \frac{(n - n_1)^2}{n^2} = 1 + \frac{n_1^2}{n^2} - \frac{2n_1}{n}$$

Also, $n_1 < n$ and thus $\frac{n_1^2}{n^2} < \frac{n_1}{n}$. We thus get

$$dCor(X^{m_1}, X^{m_2}) < 1 - \frac{n_1}{n}.$$

□

**Corollary 1.** *For a graph $G$ with $s$ sets of structurally similar nodes $\{V_1 \ldots V_k\}$ with $|V_i| = n_i$ and embedding methods $m_1$ and $m_2$ preserving purely structural and both structural and* *community properties respectively, the distance correlation bound is:*

$$dCor(X^{m_1}, X^{m_2}) < 1 - \sum_{i=1}^{s} \frac{n_i}{n}.$$

*Proof.* The summation in Theorem 1, eqn. 2, can be broken down into $2s$ parts and the rest follows as above. $\square$

### *D. Measuring Label Prediction Diversity*

We have now established the upper bound on correlation between the embeddings. We also know the following about predictions using Logistic Regression:

**Theorem 2.** *Consider two sets of feature spaces for data $\mathcal{D}$ represented as $X \in \mathbb{R}^{n \times d_1}$ and $Y \in \mathbb{R}^{n \times d_2}$ with labels for individual data points as $Z \in \mathbb{R}^n$. If logistic regression models trained on $(X, Z)$ and $(Y, Z)$ obtain accuracy of $a_X$ and $a_Y$ respectively, then we have the following bound for the model trained on $(X \parallel Y, Z)$, where $\parallel$ denotes concatenation operation:*

$$a_{X \parallel Y} \geq \max(a_X, a_Y)$$

*Proof.* Without loss of generality, assume that $a_X > a_Y$. As logistic regression is an additive model, setting weights of the model corresponding to $Y$ would yield the accuracy of the concatenated model $a_X$. $\square$

From the above theorem, we note that adding embeddings of method $m_2$ on $m_1$ would not decrease the performance. Further, the equality in Theorem 2 is realized when $Y$ is a linear scaling of $X$ or distances in $Y$ are exactly correlated with $X$. But from Theorem 1 we have an upper bound on the correlation between the embeddings. Thus, we get $a_{X \parallel Y} > \max(a_X, a_Y)$. Tighter bounds are left as future work.

Further, to empirically measure good candidates for embeddings concatenation, we compute a prediction diversity score $div_{(p_{m_1}, p_{m_2})}$, defined as the ratio of data points which are predicted differently by $m_1$ and $m_2$ with one of the methods making correct prediction to the total number of data points.

### *E. Runtime Optimization Techniques*

Given a set of $k$ embedding methods $\{m_1 \ldots m_k\}$ with optimal hyperparameters $\{\langle_1 \ldots \langle_k\}$ and the maximum time complexity from the methods as $T$ per unit dimension, a naive implementation of finding the optimal combination of methods would take a time complexity of $O(2^k \times T \times d)$, where $d$ is the embedding dimensionality. To optimize this, we do an approximation by greedily adding the next method's embedding to the current set of embeddings. This yields a time complexity of $O(k \times T \times d)$.

### *F. Algorithm*

Algorithm 1 provides the pseudo-code for the framework. Given an input graph $G$, we split the graph nodes into training, validation and test. We then use the validation set to get an accuracy score for each embedding method. Based on the evaluation score and prediction divergence metric, we greedily add the next best embedding approach to evaluate the performance of the ensemble of methods. Finally, we report the performance on a held-out test set. In the experiments below the above step is performed 5 times and the average is reported. Please note that in order to evaluate the performance of our greedy algorithm, we also compare the results with the optimal ensemble found by searching the entire search space. We call that ensemble as the optimal ensemble.

**Algorithm 1:** graphensemble

**Function** *graphensemble (Graph G, Embedding methods $\mathcal{M} = \{m_1, .., m_K\}$)*
- train, val, test $\leftarrow$ splitNodeIndexes($G$);
- **for** $i = 1 \ldots K$ **do**
  - $X^{m_i} \leftarrow$ getEmbedding($G$, $m_i$);
  - training(model, $X^{m_i}$[train]);
  - $a_i \leftarrow$ evaluate(model, $X^{m_i}$[val]);
- sortedmethods $\leftarrow$ Sort $\mathcal{M}$ based on $a$;
- $ens \leftarrow$ sortedmethods[0];
- **for** $i = 1 \ldots K$ **do**
  - **if** $m_i$ == $ens$ **then**
    - continue;
  - $div_{(p_{m_i}, p_{ens})} \leftarrow$ predDiv($p_{m_i}$, $p_{ens}$);
  - $dcor_{m_i,ens} \leftarrow$ dcor($m_i$, $ens$);
  - $greedy\_c_{m_i,ens} \leftarrow div_{(p_{m_i}, p_{ens})}$ * $a_{m_i}$ * $a_e ns$;
- ensembleset $\leftarrow$ set();
- ensembleset.add($ens$);
- remainingmethods $\leftarrow \mathcal{M}$ - ensembleset;
- $a_{prev} \leftarrow 0$;
- **for** $m \in$ *remainingmethods sorted by* $greedy\_c_{m_i,ens}$ **do**
  - ensembleset.add(m);
  - remainingmethods $\leftarrow \mathcal{M}$ - ensembleset;
  - ens $\leftarrow$ set($m_i$ $\forall$ $m_i$ $\in$ ensembleset);
  - $X \leftarrow$ concat($X^{m_i}$ $\forall$ $m_i$ $\in$ ensembleset);
  - training(model, $X$[train]);
  - $a_e ns \leftarrow$ evaluate(model, $X$[val]);
  - **if** $a_e ns < a_{prev}$ **then**
    - ensembleset.remove(m);
  - **for** $i = 1 \in remainingmethods - m$ **do**
    - $div_{(p_{m_i}, p_{ens})} \leftarrow$ predDiv($p_{m_i}$, $p_{ens}$);
    - $greedy\_c_{m_i,ens} \leftarrow div_{(p_{m_i}, p_{ens})}$ * $a_{m_i}$ * $a_{ens}$;
  - $a_{prev} \leftarrow a_e$;
- $X \leftarrow$ concat($X^{m_i}$ $\forall$ $m_i$ $\in$ ensembleset);
- training(model, $X$[train]);
- $a_{test} \leftarrow$ evaluate(model, $X$[test]);
- return $a_{test}$

## V. Experiments

In this section, we establish the Graph Ensemble approach against eight state-of-the-art baseline embedding methods to evaluate their multi-label node classification performance on four benchmark datasets. In addition, we yield insights into the correlation of graph embedding methods.

### A. Datasets

Table II shows used four benchmark real-life graphs for node classification tasks in our experiment. For each dataset, we derive the largest weakly connected component.

| Dataset | Node | Edge | Class |
|---|---|---|---|
| PPI [30] | 03,890 | 038,839 | 50 |
| BlogCatalog [31] | 10,312 | 333,983 | 39 |
| Citeseer [32] | 03,312 | 004,660 | 06 |
| Wikipedia [33] | 04,777 | 092,512 | 40 |

TABLE II: Statistics of benchmark datasets in the experiment.

### B. Baseline Graph Embedding Methods

We compare our Graph Ensemble method with the eight baseline models - Graph Factorization (GF) [34], Laplacian Eigenmaps (LAP) [35], High Order Proximity Preserving (HOPE) [36], Structural Deep Network Embedding (SDNE) [37], Node2Vec [38], Geometric Graph Convolutional Networks (Geom GCN) [18], Variational Graph Autoencoders (GraphAE) [14] and Graph Attention Networks [19].

### C. Graph Ensemble Approach

Our graph representation ensemble learning mechanism leverages a bag of single embedding methods and achieves an optimal embedding combination for graph feature learning. First, we run an individual graph embedding method on the original graph to get the best embedding at each dimension. Then, we use the greedy approximated search to add embedding generated by other methods iteratively to the embedding given by the best single method. In the end, we feed the ensemble concatenation embedding and baseline method embedding to the downstream multi-label node classification task. At each experiment round, we split the nodes of a graph into training data (50%), validation data (20%), and test data (30%). Using training data is intended to find the best hyperparamter for single methods. We choose the optimal ensemble embedding combination based on the validation data. And we report the performance of our graph ensemble methods and five baseline methods on test data.

*1) Hyperparameter Search:* In order to get the best embedding for each single graph embedding model, we employ a best hyperparamter search on the training dataset. Among three embedding dimensions 32, 64 and 128, we select the best hyperparameter set respectively at each dimension. Except for LAP which does not contain hyperparamters, we use grid search on a range of hyperparameter sets for the other four methods. For GF, we search parameters including learning rate from {1e-3, 1e-2, 1e-1} and regularization from {1e-1, 1, 10}. For HOPE, we select a decaying factor from {1e-4, 1e-3, 1e-2, 1e-1} and similarity function from Katz Index, PageRank, Common Neighbours and Adamic-Adar. For SDNE, we fix the auto-encoder structure 500, 1000, 300 nodes in each layer, and set first loss function parameter $\alpha$ to 1e-5 and penalty $\beta$ to 10. We select two regularization factors and $xeta$ from {1e-3, 1e-2} respectively. As for Node2Vec, we set walk length to 80, number of walks to 10, context size to 10. We select return $p$ and in-and-out $q$ from {0.25, 0.5, 1, 2, 4} respectively.

*2) Ensemble Combination Search:* After obtaining the best hyperparameter set for each method at each dimension, we evaluate their performance on a multi-label node classification task with the validation dataset and select the optimal ensemble combination. First, we choose the best method, which has the best performance on the training data. We test its performance on validation data under the best setting with respect to three dimensions 32, 64, and 128, and then select its best dimension based on Macro $F_1$ score. Secondly, we append the embedding of the second best method at three dimensions separately to the best embedding so far and repeat the evaluation process. The criteria we use to select the next best method is based on the highest value of the product between the prediction divergence and the individual method's macro averaged F-score. If the performance improves, we keep the second embedding at the chosen dimension. Otherwise, we abandon this method and continue the appending process. In the end, we will obtain the best combination iteratively via such greedy approximation.

### D. Embedding Correlation and Prediction Divergence

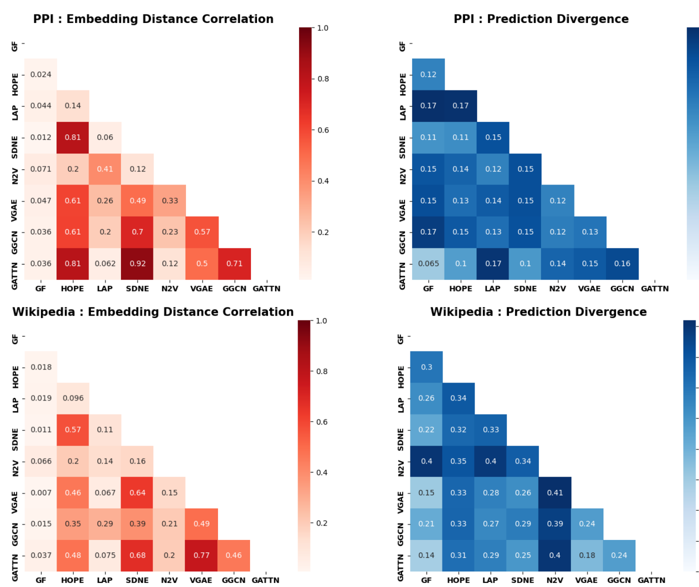


Fig. 4: Distance correlations of embedding methods on real networks (dimensions set to 128).

The distance correlations between the embeddings obtained by different embedding methods are presented in Figure 4. We observe that the correlation between the embeddings varies significantly with the underlying dataset. For PPI and Wikipedia, we see that most methods are weakly correlated. This strengthens our claim in Theorem 1 that embedding methods preserve different properties and if the underlying graph is complex, then the embeddings will be diverse. For both Wikipedia and PPI datasets, we observe that SDNE and HOPE have high correlation values, as both preserve higher order proximity in a non-linear way, further strengthening our claim. We also observe, in general, that for the pair of embedding methods that have high correlation values, prediction divergence score is low. A high value of prediction divergence score directly provides an indication of how much gain can be obtained by ensembling the pair of embeddings. However, if the distance correlation values for those methods is very low, not much performance gain was observed because of induction of high non-linearity in the joint embedding space. To combat this, we use a product of distance correlation and prediction divergence as the criteria to greedily select embedding methods for ensembling.

### *E. Multi-label Node Classification*

In the multi-label node classification task, we are given a graph as well as labels of a proportion of nodes as training data. And we aim to predict the unknown labels for the rest of the nodes in the test data. Each node in the graph has one or multiple labels. To evaluate the graph ensemble embedding and baseline methods embedding, we utilize the same One-Vs-the-Rest multi-label strategy and Logistic Regression (with $L2$ regularization and class balancing using sample weights) to build classifiers. To ensure the robustness of our proposed graph ensemble methods and stability of the experiments, we repeat the whole process for five rounds and report the average results. We use Macro $F_1$ and Micro $F_1$ as evaluation metrics. Micro $F_1$ has a similar performance like Macro $F_1$; thus, it is not reported in the paper. We care more about the minority class prediction, and Macro $F_1$ is preferably considered.

We summarize multi-label classification results in Table III. Overall, we observe that the ensemble of methods outperforms individual methods significantly except for Citeseer. Geometric GCN gives the highest accuracy for all PPI and BlogCat datasets, Graph AE for Citeseer and HOPE for Wikipedia. The performance improvement with embeddings concatenation can be attributed to the interplay of embeddings when concatenated together, and the amount of information shared.

### *F. Minority Class*

Figure 5 highlights the $F_1$ score of our graph ensemble methods on smaller classes is higher than the best individual methods. Our graph ensemble strategy combines the captured features derived by all single methods and generates a comprehensive graph embedding, which can improve the performance of less represented classes. In Wikipedia, we observe that for minimal classes, none of the individual methods perform well. However, the combination ensemble performs well and gives $F_1$ upto 0.8. Similarly, in Citeseer, we see an improvement of about 40% for less represented labels.

| **Dataset** | **Method** | **Dimension** | **Macro-F1↑** |
|---|---|---|---|
| PPI | GF | 128 | 0.908 |
| | LAP | 128 | 0.051 |
| | HOPE | 128 | 0.104 |
| | SDNE | 32 | 0.139 |
| | Node2Vec | 128 | 0.146 |
| | Geom GCN | 128 | <u>0.167</u> |
| | Graph AE | 128 | 0.155 |
| | Graph Attn | 128 | 0.114 |
| | **Majority Vote** All | All | **0.169 (1.2%)** |
| | **Optimal Concatenation** of Geom GCN, Graph AE, Node2Vec, SDNE | 128, 128, 128, 32 | **0.198 (18.56%)** |
| | **Greedy Concatenation** of Geom GCN, Graph AE, Node2Vec, SDNE, Graph Attn | 128, 128, 128, 32, 128 | **0.193 (15.57%)** |
| BlogCat | GF | 128 | 0.042 |
| | LAP | 32 | 0.047 |
| | HOPE | 128 | 0.127 |
| | SDNE | 128 | 0.198 |
| | Node2Vec | 128 | 0.214 |
| | Geom GCN | 128 | <u>0.223</u> |
| | Graph AE | 128 | 0.116 |
| | Graph Attn | 128 | 0.084 |
| | **Majority Vote** All | All | **0.232 (4.04%)** |
| | **Optimal Concatenation** of Geom GCN, HOPE, LAP, Node2Vec, SDNE | 128, 128, 32, 128, 128 | **0.256 (14.80%)** |
| | **Greedy Concatenation** of Geom GCN, HOPE, Node2Vec, SDNE | 128, 128, 128, 128 | **0.254 (13.90%)** |
| Citeseer | GF | 128 | 0.114 |
| | LAP | 128 | 0.666 |
| | HOPE | 64 | 0.636 |
| | SDNE | 128 | 0.398 |
| | Node2Vec | 128 | 0.643 |
| | Geom GCN | 64 | 0.580 |
| | Graph AE | 64 | <u>0.669</u> |
| | Graph Attn | 128 | 0.355 |
| | **Majority Vote** All | All | **0.676 (1.05%)** |
| | **Optimal Concatenation** of GF, HOPE, LAP, SDNE, Graph AE, Node2Vec | 128, 64, 128, 128, 64, 128 | **0.697 (4.2%)** |
| | **Greedy Concatenation** of HOPE, LAP, Graph AE, Node2Vec | 64, 128, 64, 128 | **0.692 (3.44%)** |
| Wikipedia | GF | 64 | 0.041 |
| | LAP | 128 | 0.032 |
| | HOPE | 128 | <u>0.159</u> |
| | SDNE | 128 | 0.030 |
| | Node2Vec | 128 | 0.098 |
| | Geom GCN | 64 | 0.102 |
| | Graph AE | 128 | 0.064 |
| | Graph Attn | 64 | 0.058 |
| | **Majority Vote** All | All | **0.168 (5.66%)** |
| | **Optimal Concatenation** of GF, HOPE, LAP, Node2Vec Geom GCN, Graph AE | 64, 128, 128, 128, 64, 128 | **0.191 (20.12%)** |
| | **Greedy Concatenation** of HOPE, LAP, Node2Vec Geom GCN | 128, 128, 128, 64 | **0.187 (17.61%)** |

TABLE III: Macro $F_1$ score of Graph Ensemble methods. The percentage inside parentheses indicates the performance gain of Graph Ensemble over the best single method.

## VI. CONCLUSION

In this paper, we proposed a framework which can create an ensemble of graph embedding approaches outperforming each method. We provided a theoretical analysis of the framework and established the upper bound on the correlations between graph embedding techniques. Further, we compared our method with state-of-the-art embedding methods and showed improvement in four real-world networks. We also showed that the model is even more useful for underrepresented classes. There are several research directions for future work: (1) tighter ensemble bound, (2) information theoretic approaches which can take into account the mutual information between

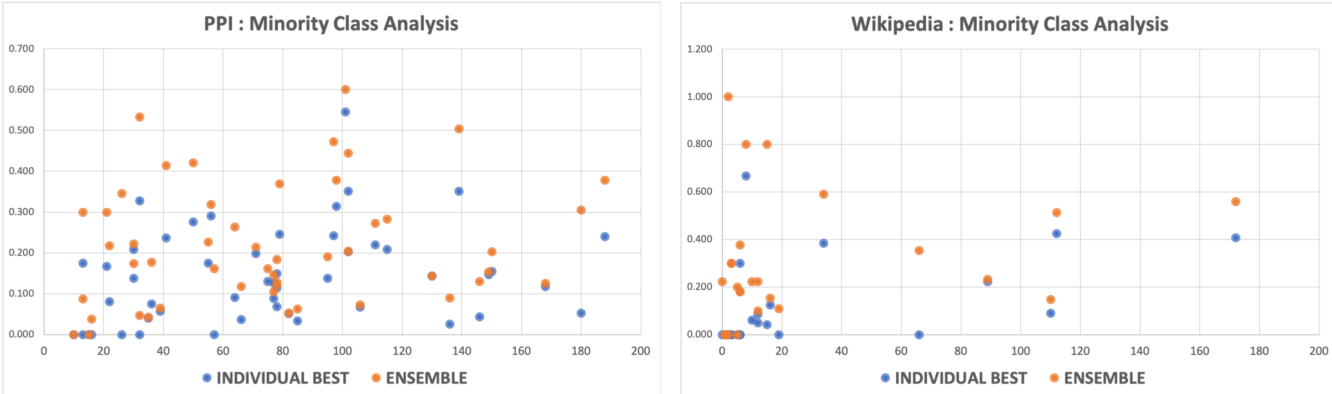


Fig. 5: Node classification results on PPI and Wikipedia Datasets. Y-axis is $F_1$ score on each class.

embeddings, and (3) dynamic ensembles for evolving graphs.

## References


[1] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 kdd cup," *ACM SIGKDD Explorations*, vol. 5, no. 2, 2003.

[2] L. C. Freeman, "Visualizing social networks," *Journal of social structure*, vol. 1, no. 1, p. 4, 2000.

[3] A. Theocharidis, S. Van Dongen, A. Enright, and T. Freeman, "Network visualization and analysis of gene expression data using biolayout express3d," *Nature protocols*, vol. 4, pp. 1535–1550, 2009.

[4] P. Goyal, A. Sapienza, and E. Ferrara, "Recommending teammates with deep neural networks," in *Proceedings of the 29th on Hypertext and Social Media*. ACM, 2018, pp. 57–61.

[5] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705118301540

[6] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. of ACM SIGKDD*, 2016, pp. 1105–1114.

[8] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine learning*, vol. 42, no. 3, pp. 287–320, 2001.

[9] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[10] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 37–48.

[11] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, vol. 14, 2001, pp. 585–591.

[12] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1145–1152.

[13] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1225–1234.

[14] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[15] ——, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[16] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[17] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[18] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," *arXiv preprint arXiv:2002.05287*, 2020.

[19] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. A. Alemi, "Watch your step: Learning node embeddings via graph attention," in *Advances in Neural Information Processing Systems*, 2018, pp. 9180–9190.

[20] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[21] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 993–1001, 1990.

[22] D. W. Opitz, "Feature selection for ensembles," *AAAI/IAAI*, vol. 379, p. 384, 1999.

[23] F. Ricci and D. W. Aha, "Extending local learners with error-correcting output codes," *Naval Center for Applied Research in Artificial Intelligence, Washington, DC*, 1997.

[24] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.

[25] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[26] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[27] P. Robert and Y. Escoufier, "A unifying tool for linear multivariate statistical methods: the rv-coefficient," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 25, no. 3, pp. 257–265, 1976.

[28] G. J. Székely, M. L. Rizzo *et al.*, "Brownian distance covariance," *The annals of applied statistics*, vol. 3, no. 4, pp. 1236–1265, 2009.

[29] G. J. Székely, M. L. Rizzo, N. K. Bakirov *et al.*, "Measuring and testing dependence by correlation of distances," *The annals of statistics*, vol. 35, no. 6, pp. 2769–2794, 2007.

[30] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood *et al.*, "The biogrid interaction database: 2008 update," *Nucleic acids research*, vol. 36, no. suppl 1, pp. D637–D640, 2008.

[31] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of the 15th international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 817–826.

[32] Q. Lu and L. Getoor, "Link-based classification," in *ICML*, vol. 3, no. 2003, 2003, pp. 496–503.

[33] M. Mahoney, "Large text compression benchmark," *URL: http://www. mattmahoney. net/text/text. html*, 2011.

[34] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 37–48.

[35] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[36] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1105–1114.

[37] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[38] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.