ID-GNN: UNSUPERVISED GRAPH NEURAL NETWORK

by

Sara Lilly Mount

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

May 2024

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Sara Lilly Mount

Thesis Title: ID-GNN: Unsupervised Graph Neural Network

Date of Final Oral Examination: 15 March 2024

The following individuals read and discussed the thesis submitted by student Sara Lilly Mount, and they evaluated the student's presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Edoardo Serra, Ph.D.	Chair, Supervisory Committee
Francesca Spezzano, Ph.D.	Member, Supervisory Committee
Marion Scheepers, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Edoardo Serra, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ACKNOWLEDGMENT

I give many thanks to my advisor, Edoardo Serra, for continuous support and accommodation of my myriad setbacks.

I also owe gratitude to my husband, Dustin Mount, for the times when he shouldered more than his share of the burdens (who are named Auggie and Blaise).

ABSTRACT

Graph-structured data has increasing applicability in hundreds of domains, the majority of which produce mostly unlabeled data. For this reason, the supervised methods of graph representation learning, which dominate current methods, have limited use in many cases. Similarly, many methods rely on a graph being homophilic in which proximity implies similarity. In many applications, the structure of a node's connections carries more relevant information than the nodes to which it is connected. The proposed method is an unsupervised graph neural network that doesn't rely on homophily of the graphs. It creates an identifying signature for each node and supplements each node's features with the IDs of its neighbors to encourage representations that can be used to reconstruct the features of the neighbors. The produced representations may be used in downstream tasks such as node classification. Experiments show that ID-GNN is within 3% of the node classification accuracy of state-of-the-art methods on heterophilic graphs.

CONTENTS

ACKNOWLEDGMENT iv

ABSTRACT v

TABLE OF CONTENTS v

LIST OF TABLES viii

LIST OF FIGURES ix

LIST OF ABBREVIATIONS xi

1 INTRODUCTION 1

2 RELATED WORK 4

2.1 Graph Neural Networks 4

2.2 Heterophily vs. Homophily 5

2.3 Unsupervised Graph Representation Learning 6

2.4 Classifiers 8

3 METHOD 10

3.1 Overview 10

3.2 Setting 11

3.3 ID Codes . . . 13
3.4 Sampling Function . . . 13
3.5 Graph Neural Network . . . 14
3.6 MLP for Reconstruction . . . 14
3.7 Representation Source . . . 14

4 EXPERIMENTS . . . 15
4.1 Datasets . . . 15
4.2 Training . . . 16
4.3 Evaluation Task . . . 17
4.4 Tuning . . . 17
4.4.1 Activation Function . . . 18
4.4.2 Network Depths . . . 18
4.4.3 Dimensionality Reduction . . . 18
4.4.4 Hash Function . . . 18
4.5 Experimental Setup . . . 19
4.5.1 Embedding Source . . . 19
4.5.2 GNN Type . . . 19
4.5.3 ID Encoder Method . . . 19
4.6 Comparisons with State-of-the-art . . . 19

5 RESULTS . . . 21
5.1 Tuning . . . 21
5.1.1 Activation Function . . . 21
5.1.2 Network Depths . . . 21

5.1.3 Dimensionality Reduction 22
5.1.4 Hash Function 23
5.2 Experiments 23
5.2.1 Embedding Source 23
5.2.2 Classifier 24
5.2.3 GNN Type 24
5.2.4 ID Encoder Method 25
5.3 Comparisons with State-of-the-art 26
5.4 Discussion 26

6 CONCLUSION 28

REFERENCES 28

APPENDICES 31

A ARTIFACTS 32

LIST OF TABLES

4.1 Heterophilic dataset characteristics 16

4.2 Homophilic dataset characteristics 16

5.1 Tuning accuracy by activation combination 22

5.2 Tuning accuracy by GNN depth 22

5.3 Tuning accuracy by PCA dimension 23

5.4 Tuning accuracy by hash type 23

5.5 Accuracy by embedding source 24

5.6 Accuracy by classifier type 24

5.7 Accuracy by GNN type 25

5.8 Accuracy on homophilic datasets by ID encoder method 25

5.9 Accuracy on heterophilic datasets by ID encoder method 25

5.10 Average classification accuracy compared to state-of-the-art methods on homophilic datasets 26

5.11 Average classification accuracy compared to state-of-the-art methods on heterophilic datasets 26

LIST OF FIGURES

3.1 Diagram of Proposed Approach 12

LIST OF ABBREVIATIONS

GNN	Graph Neural Network
MPNN	Message Passing Neural Network
GCN	Graph Convolutional Network
GIN	Graph Isomorphism Network
GAT	Graph Attention Network
MLP	Multi Layer Perceptron
DGI	Deep Graph Infomax
NWR-GAE	Neighborhood Wasserstein Reconstruction Graph Auto Encoder
OT-GNN	Optimal Transport-based GNN
GREET	Graph Representation learning with Edge hEterophily discriminaTing
PaireE	Paire Embedding
ID	Identification
DR	Dimensionality Reduction
PCA	Principal Component Analysis
GRL	Graph Representation Learning

CHAPTER 1:

INTRODUCTION

The amount of data being produced each year is growing exponentially. It is projected that by 2025, 150 zetabytes of data will need analysis [3]. Generally, the predictive value of the data lies in information not inherently included in the data, such as the subject in a photo or the legality of a financial transaction. The trade-off of such vast amounts of data is the impossibility of manually labeling it. This is where unsupervised methods of machine learning are becoming more and more necessary.

Each entity in typical data is self-contained in that its characteristics give information about only itself. Graph-structured data includes another level of information in the connections between entities [15]. These connections can be mined for a host of additional knowledge. Perhaps an entity (node or vertex of a graph) does not have particularly interesting characteristics of its own—or any at all–but the way it is connected to other nodes reveals something about it. An increasing amount of real-world data includes this kind of linkage information: social networks, computer networks, citation networks, molecular structure, etc [15].

In general, there are two modalities for the patterns in which nodes of different types are connected. A node connected to many other nodes of a certain type may be more likely to be that type also. This type of graph is known as homophilic. Alternately, the pattern of connections surrounding a node might make it very similar

to another node with similar connection patterns though the two nodes have no connectivity between them, making it heterophilic [18]. There is currently a much greater body of research regarding techniques that rely on homophilic graphs than there is for heterophilic graphs [5], though there are many occasions where proximity does not imply similarity [18]. Examples are given in Section 4.1.

One way to mine a graph's connectivity information is to use it in determining a numerical representation of each node—or entire graph—in some-dimensional vector space. The goal is for two vectors to be "near" each other in vector space by some distance metric if and only if the nodes they represent are similar in some other way. In typical data, the similarity can only depend on the characteristics of individual entities. In a graph, the similarity can also take into account the characteristics of its neighbors. These vector representations can be used in various downstream tasks such as node or graph classification, node clustering, link prediction, or anomaly detection.

Often vector representations of the nodes or graphs, also known as embeddings, are produced via neural networks, specifically known as Graph Neural Networks (GNNs). In a graph setting, each layer of the neural network performs aggregation on information from each node's neighbors [15]. After each layer of the neural network has fed its results to the next layer, the representations of each node contain information about its neighbors and neighbors' neighbors, and so on, out to the desired distance. It is the methods for selecting, aggregating, and weighting this information that differentiate the types of GNN. Each network is trained so that the calculations within the neurons produce, as a whole, a desirable final embedding based on whatever loss function represents the goal. In unsupervised settings, this goal is often the ability to recreate some original component of the graph such as its connections. This

architecture is known as a Message Passing Neural Network (MPNN) [15].

There is a known limitation of GNNs to perform poorly on heterophilic graphs [18]. Since they rely heavily on incoming data from neighbor nodes, the final representations of a node and its neighbors tend to be similar. This is also related to the over-smoothing problem where, after a certain number of convolutions, the representations of all nodes tend to have amalgamated so much information from the surroundings that they are indistinguishable [18, 10]. This prevents the solution for heterophilic graphs from being a deeper network of message passing.

In the proposed approach, an identifying representation for each node is produced by some method-–either a neural network, hash algorithm, or dimensionality reduction of the features. These IDs are then combined with other available features and a GNN produces embeddings for each node. Feedback to the GNN depends on a decoder's ability to reconstruct the features of a node's neighbors from its embedding when the IDs of the neighbors are given as markers to aid the reconstruction. I dub this method ID-GNN.

CHAPTER 2:

RELATED WORK

2.1 Graph Neural Networks

A comprehensive survey on graph neural networks [15] breaks down the landscape of GNNs into four main categories: recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial–temporal GNNs. ID-GNN follows most closely the graph autoencoder structure and uses a convolutional GNN as the encoder. Often the objective of the decoder is to reconstruct the adjacency matrix or other representation of the graph. In many cases, the objective encourages nodes which are near each other to have similar representations and nodes which are farther away to be different. This is one reason why many methods do well on homophilic graphs and poorly on heterophilic ones. While the authors do include both homophilic and heterophilic graphs in their list of benchmark datasets, they do not discuss the difference between them.

In the survey of Sato [11], the author sets out to explain the properties of graph structures GNNs can and cannot distinguish. There are many examples of non-isomorphic graphs that a GNN cannot distinguish, generally due to similarity of node features. For example, when the node features are identical a GNN cannot distinguish between a hexagon and two triangles. If the features are random, the GNN

is successful. He shows that GNNs are at most as powerful as the one-dimensional Weisfeiler Leman algorithm, but proved that adding noise to the features improves the power of the GNN. This shows promise that the additional ID motif proposed in this paper can contribute to the GNN's ability to differentiate structures within a graph.

Based on convolutional neural networks, the Graph Convolutional Network (GCN) [6] is applied to the irregular structure of graph data to represent both the structure and features of each node. GCNs do so by aggregating the features of a node and its neighbors so that after each convolution, each node's representation contains information about a larger and larger portion of its surrounding neighborhood. A GCN assigns weights to the neighboring nodes during aggregation which are calculated based on each node's number of neighbors, or degree [15]. The Graph Isomorphism Network (GIN) [16] is a type of convolutional neural network that assigns a weight to the node's ego features that is updated during training [15]. These different aggregation mechanisms lead to different strengths in representation learning. In [8], a simple GCN is used as a baseline comparison and achieves higher accuracy than a Graph Attention Network (GAT), or Multi Layer Perceptron (MLP). Though they did not test GIN, it is also promising for use on heterophilic networks since it differentiates the ego node from the neighbor nodes during aggregation. GCN and GIN are selected for comparison as the GNN encoder component of ID-GNN. In addition, GIN is used as one of the methods for producing IDs for similar reasons.

2.2 Heterophily vs. Homophily

Since much real-world data is inherently heterophilic, the authors of [18] describe a metric for measuring the degree to which a graph is heterophilic vs. homophilic. A

simple average of the proportion of the nodes' neighbors which are the same class gives a value between 0 and 1 where 0 represents a perfectly heterophilic graph in which a node is never connected to nodes of the same class. This metric can also be applied to graphs in which the edges have classes as well, which is not studied in this paper. The authors divide GNN methods for heterophilic graphs into a taxonomy based on the methods they use to overcome the limitations of a message-passing system for such graphs. Broadly speaking, the lack of proximity of nodes of the same class is dealt with by changing either the definition of neighbor or the architecture of the GNN. Instead of using only one-hop neighbors in the message passing, many methods either increase the number of hops or search the graph for similar nodes. Changes to the message-passing system itself include weighting same-class and different-class neighbors differently, keeping the node's own representation separate from its neighbors', and combining local and global views in a node's representation. It is important to note that some of these techniques would not apply to unsupervised methods.

2.3 Unsupervised Graph Representation Learning

Deep Graph Infomax (DGI) [14] is a state-of-the-art method for unsupervised graph representation learning. Its main mechanism is to create representations of the small neighborhoods of the graph which share the maximum amount of information with an entire graph summary. This is accomplished by a discriminator which is given positive and negative samples of graph-neighborhood summary pairs and taught to create representations which maximize the similarity of positive samples. These representations are used for node classification on homophilic citation networks. While

this method is widely cited and implemented, it is not optimal for heterophilic data and other methods have shown improvement over DGI on both types of network.

The method proposed in [13] employs an encoder-decoder structure using a graph neural network to encode a node's neighborhood based on its degree and the distribution of its neighbors' features and a decoder based on Neighborhood Wasserstein Reconstruction (NWR-GAE). This is accomplished by a three part loss: one for the degree, one for the neighborhood representation, and one for the node's features. This multi-goal setup has the ability to encode both homophilic and heterophilic graphs. For this reason, NWR-GAE has increased node classification accuracy over DGI on both homo- and heterophilic datasets while still being unsupervised. This and several other methods, including DGI, were also tested on a synthetic dataset of graphs with increasing amounts of perturbation to find the failure point of structure discrimination power. NWR-GAE achieves better results than DGI on all but the simplest synthetic graph structures.

The Optimal Transport-based GNN (OT-GNN) [17] method is a training framework that enforces a constraint on a GNN that requires that the produced embeddings fall into equal-sized clusters. This is intended to fight the over-smoothing problem in which the node embeddings learn to become extremely similar making it difficult to distinguish between them. While this method is able to generate higher accuracy node classifications on both types of graphs, the requirement for representations to fall into equal clusters would be problematic for datasets in which the distribution of classes is not even. Thus, if similar results can be achieved without such a constraint, the new method would be applicable in many more scenarios.

GREET [8] discriminates between same-class and different-class neighbors during

message passing in a contrastive way such that nodes that are likely to be the same class are encouraged to form similar representations, whereas nodes that are likely to be different classes are encouraged to form dissimilar representations. This is made possible even in an unsupervised setting by using the node attributes and structure to infer class. Furthermore, the ad-hoc classifier and representation learning alternately train each other. In this way, it is designed to perform well on both homophilic and heterophilic graphs.

Geom-GCN [10] overcomes the distance between similar nodes in heterophilic graphs by mapping node features into latent space and including similar-yet-distant nodes as neighbors during aggregation. Similarly, the geometric relationship between nodes is given a latent representation to preserve structural relationships as well. Though this is a semi-supervised method, an extension to an unsupervised setting is easily imaginable.

The PairE method described in [7] expands the unit of embedding from a single node to node pairs of nodes to further differentiate between nodes which might result in similar representations. To preserve dissimilarity, the features of the nodes in each pair are concatenated rather than aggregated. Additionally, during the aggregation process of message passing, the neighbors of each node of the pair are aggregated per node and then concatenated. This allows each node to maintain its identity within the pair. The GNN is trained to reconstruct both the nodes' features and the aggregated features of the immediate neighbors.

2.4 Classifiers

To test ID-GNN, I compare two ensemble methods for node classification. Each creates a progression of simple classifiers which enhance each other's results. They

pass the intermediate results on to the next simple classifier, with higher weights being given to the entities that were classified wrongly by the previous classifier. In so doing, each level of the process aims to get right what its predecessors did not. The final decision is based on an average or vote from all the learners. XGBoost [1] specifically chooses splitting points within its classifiers based on those that will move the result in the direction of the gradient for maximum effect. Extra-Trees [4] uses randomly selected splitting attributes and randomly selected splitting points. For node classification, the node "attributes" generated by the embedding algorithm separate the node from others based on the splits in the tree, and the node is classified according to which leaf of the tree it becomes. These methods balance accuracy with efficiency, so make an easy choice for a simple classifier for the downstream node classification task.

CHAPTER 3:

METHOD

3.1 Overview

The goal of ID-GNN is to train an encoder to produce vector representations for each node of a graph such that the vector of two nodes will be close in vector-space if the nodes are connected to similar neighbors in similar patterns. A node's representation should contain information about its immediate neighbors in order to differentiate it from similar nodes with dissimilar neighbors. Using a Graph Neural Network as an encoder, each representation of a node is incentivized to contain information about its neighbors based on the output of a decoder. If the decoder is able to reconstruct the features of a node's neighbors using only the node's own vector representation, the message-passing mechanisms of the GNN are successful.

To this end, I supplement a node's features with an identifying signature. The method for creating this ID is the main topic of this study. These IDs subsequently follow a node's features through the GNN, acting like markers so that given the vector representation produced by the GNN and the IDs of the neighbors, the decoder can reconstruct the original features of those neighbors.

If the ID itself contains any information that aids in the reconstruction of a node's features, the decoder will be successful whether or not the encoder truly passes infor-

mation through the structure of the graph. This is because the node representations produced by the GNN are supplied to the decoder already associated with the IDs of their neighbors. Any neighbor feature information in the ID will appear to be reconstructed from the node's representation when really it was reconstructed from the ID. Conversely, the IDs must also be identifying, i.e., there are likely no two exactly alike, so that they differentiate nodes whose features are similar to begin with. So the crux of the problem is to choose a method to create the IDs that balances loss of information with retention of information.

The general steps in the architecture of this method are

1. Create ID signatures for all nodes
2. Sample a portion of each node's neighbors to a k-hop distance ($\sim$)
3. Combine IDs with node features ($X + ID$)
4. Use $X + ID$ in a Graph Neural Network to obtain representations of each node (H)
5. Combine the IDs of sampled neighbors ($\widetilde{ID}$) with H
6. Use $\widetilde{ID} + H$ in a Multi-Layer Perceptron to reconstruct neighbor node features (J)
7. Calculate loss between sampled neighbors' features ($\widetilde{X}$) and J

3.2 Setting

Let G be a graph with vertices V and edges E. Let A represent its adjacency matrix where each entry a_{ij} represents the presence or absence of an edge between

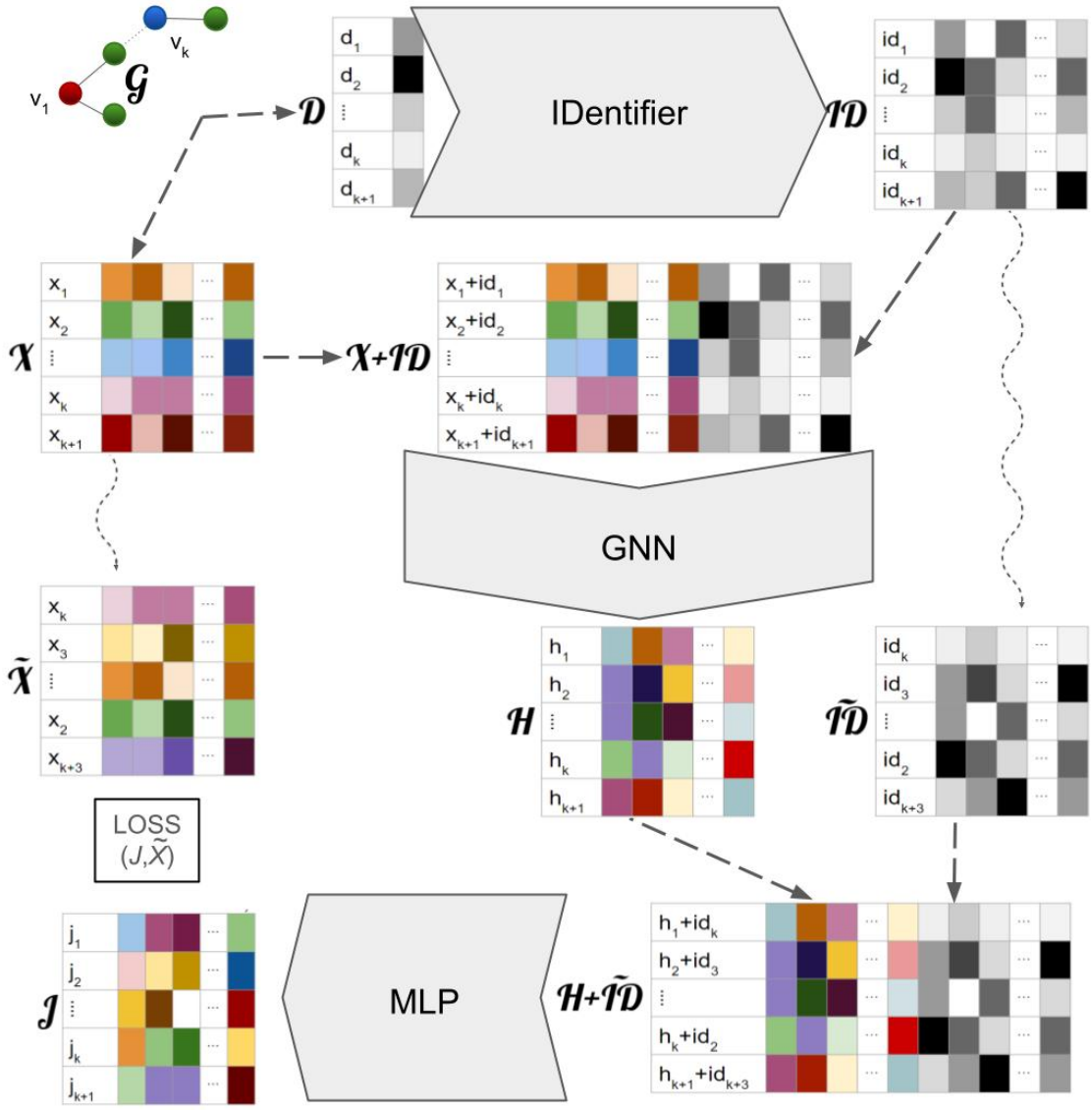

Figure 3.1: Diagram of Proposed Approach

nodes v_i and v_j. D will represent the degree matrix where d_i contains the number of neighboring nodes for node i and X will represent the attribute matrix where x_i is a vector of node attributes for node i.

3.3 ID Codes

To produce the IDs of the nodes, I propose to compare three methods: Graph Isomorphism Network (GIN), hash algorithm, or dimensionality reduction (DR) of their features using Principal Component Analysis (PCA). In the case of the GIN, it is given the degree matrix rather than node features and produces representations that encode the structural identity of each node. Alternatively, the attribute vector x_i of each node is hashed with a common hashing algorithm, creating a very likely unique code for each node. Finally, using a PCA on the features matrix produces a new set of features which are a linear combination of the original features. These submethods of ID-GNN are named for the method used for the ID: GIN-GNN, hash-GNN, and DR-GNN, respectively. These IDs are represented as *ID* in Figure 3.1. This diagram is specifically showing the flow of GIN-GNN, though for the other sub-methods, D is replaced by X. The "IDentifier" represents either the GIN, hash, or PCA step. Additional tests are performed where the ID is created from concatenation of GIN output and hashed features as well as concatenation of GIN output and PCA, dubbed GINhash-GNN and GINDR-GNN.

3.4 Sampling Function

A sampling function creates companion matrices $\widetilde{ID}$ and $\widetilde{X}$ of the IDs and features, respectively, of a randomly selected set of k neighbors of each node of the graph. Therefore $\widetilde{id}_i$ contains the IDs of k of the neighbors of v_i and $\widetilde{x}_i$ contains the

features of the same k neighbors of v_i. Given the parameters probability and hops, the function will randomly select a neighbor of a node and with a certain probability add it to the list of neighbors of the original node until it reaches a neighbor a certain number of hops from the original node. This function is represented by the wavy arrow in Figure 3.1.

3.5 Graph Neural Network

The ID is then concatenated with the features X and passed to a graph neural network. I compare the effectiveness of using a GCN to a GIN for the neural network. Either will produce a reduced-dimension embedding of each node, H.

3.6 MLP for Reconstruction

The representations produced by the GNN, H, and the identifiers of each node's sampled neighbors, $\widetilde{ID}$, are concatenated and passed as input to a Multi-Layer Perceptron. This works to reconstruct the features of each node's sampled neighbors, comparing the produced output J to the features of the neighbors selected by the sampling algorithm, $\tilde{X}$. The results of this loss are used as feedback in the backward propagation to the GNN so that the end result is node embeddings that contain enough information about its neighborhood to reconstruct that neighborhood's features.

3.7 Representation Source

Final embeddings to be used in the downstream task are taken from the model state with the lowest loss. Since the ID-GNN creates two different representations of each node (from the ID and the GNN), there are three ways to generate final representations: from the ID, from the GNN, or a concatenation of both representations.

CHAPTER 4:

EXPERIMENTS

4.1 Datasets

I select the same datasets selected in [9, 13, 17, 8, 10] which include three homophilic networks and six heterophilic networks. These are commonly used datasets to test unsupervised GNNs [15]. Cora, CiteSeer, and PubMed [12] are citation networks in which nodes represent academic papers, edges represent citations, and the papers are categorized into classes by subject. These datasets are homophilic since papers that cite or are cited by each other are likely to be about the same subject. Cornell, Texas, and Wisconsin [2] are college website networks where pages are nodes, links are edges, and the classes are the type of webpage. They are heterophilic since pages labeled "faculty" may be only distantly connected to other faculty pages, but perhaps faculty pages tend to be linked to course pages at a higher rate than student pages do. Actor, Chameleon, and Squirrel are also networks of webpages from Wikipedia. Each node represents an article from a Wikipedia category and the edges represent links between articles. In the case of Chameleon and Squirrel, the label data is continuous and represents the average monthly traffic to each page. These have been discretized into categories as in [10], in this case using quantiles. Labels for the Actor dataset represent the function of the entity in the article (actor, director,

Table 4.1: Heterophilic dataset characteristics

Dataset	**Cornell**	**Texas**	**Wisconsin**	**Actor**	**Chameleon**	**Squirrel**
Nodes	251	183	183	7,600	2,277	5,201
Edges	499	295	309	33,391	36,101	217,073
Features	1,703	1,703	1,703	932	3,132	3,148
Classes	5	5	5	5	5	5
Homophily	0.3855	0.0968	0.1498	0.2210	0.2470	0.2156

Table 4.2: Homophilic dataset characteristics

Dataset	**Cora**	**CiteSeer**	**PubMed**
Nodes	2708	3327	19717
Edges	5429	4732	44338
Features	1433	3703	500
Classes	7	6	3
Homophily	0.8258	0.7175	0.7924

film...). Wikipedia datasets are also heterophilic, likely since popular pages more often link to subcategories of the same topic which are less popular than more general topics. All nine datasets' features are vectors representing a bag-of-words where each variable indicates the presence or absence of a word in the page or document. The homophily metric is calculated using the metric described in Section 2.2. Details about these datasets are in Tables 4.1 and 4.2. All datasets are used for a node classification task.

4.2 Training

I fix the dimension of the hidden layers at 2000, employ early stopping with a patience of 300, a dropout rate of 0.3, and use an Adam optimizer with initial learning rate of 0.001 and weight decay of 0. For the sampling function, the probability is set

to 0.1 and hops to 2.

4.3 Evaluation Task

All tuning and experiments are evaluated using the task of node classification. Node representations are produced by ID-GNN without access to labels and then fixed before classification begins, ensuring an unsupervised method. In every case, the classifier divides the output of ID-GNN into 10 random 80/20 splits and the reported metric is the average accuracy for the 10 rounds of classification. XGBoost [1] and ExtraTrees [4] are used for comparision. XGBoost is used with default parameters and ExtraTrees is set to a maximum depth of 30 with 100 estimators.

4.4 Tuning

Several parameters of the ID method and GNN may influence the ability of the produced representations to differentiate nodes. I separately tune the combination of activation functions, depth of neural networks, number of components of the PCA, and type of hash function. Tuning is performed using the WebKB datasets. Each model is trained on the entire dataset and evaluated on the same 80% tuning subset of the data. Trials are performed 10 times on each combination of parameters and I report the average accuracy of the 10 trials. For each parameter and each variation, analysis is separated for the two types of GNN so that selected parameters may be different based on whether GIN or GCN is being used. Each set of parameters is fixed before experimentation, and only the 20% test subset of the WebKB datasets are used to evaluate experiments.

4.4.1 Activation Function

Activation functions in a neural network affect how the layers of a GNN pass information to one another, modifying the output of one layer before being sent to the next. I compare three options for activation function: sigmoid, ReLU, and none. There are four places where an activation function may be used within ID-GNN: the GIN (when used for the ID encoder), the GNN, the MLP, and the overall training loop. Since these may interact, I compare all 81 unique combinations of activation function. These combinations are tuned using the GIN-GNN variation since it requires activation functions in all four places.

4.4.2 Network Depths

The number of layers of a GNN also affects its ability to pass messages through the network. Deeper networks tend to oversmooth node representations, so it is important to find a balance. Since GIN-GNN uses two neural networks, I optimize the number of layers in each by testing all combinations of 2, 3, and 4 layers using the optimal combination of activation function determined in the previous tuning step. For activation function tuning, two layers were used in each network.

4.4.3 Dimensionality Reduction

For DR-GNN, the number of components of the PCA directly affects how much information ends up in the ID. I test a PCA of the node features with 5, 25, 50, and 100 components on both types of GNN. This process uses the optimal activation functions and depths discovered in the previous tuning steps.

4.4.4 Hash Function

I also compare two methods of producing a hash of the nodes' features: SHA256 and SHA3. The method by which the features are hashed will affect the uniqueness of

the ID and the MLP's ability to reconstruct the features in the hash-GNN variation. Each hash function is tuned on both hash-GIN and hash-GCN using the optimal activation functions and depths discovered in previous tuning steps.

4.5 Experimental Setup

4.5.1 Embedding Source

Throughout all iterations of tuning and experimentation, the classifier is given three different representations of the nodes from the final state of training: the ID, the output of the GNN, and the concatenation of both (ID+GNN). The ID alone should not produce useful embeddings. The question is whether including this bit of data enhances the representations produced by the GNN or not.

4.5.2 GNN Type

All tuning and experimentation combinations are performed using first a GCN and then a GIN as the encoder. Whether one or the other has an advantage from its particular method of message-passing is determined by the overall accuracy achieved during node classification.

4.5.3 ID Encoder Method

Each of five different methods for generating an ID is tested in combination with each type of GNN. The different methods are GIN, dimensionality reduction (DR), hash, and concatenations of GIN+DR and GIN+hash.

4.6 Comparisons with State-of-the-art

To eliminate the choice of classifier and train/test split points, I reverse engineer PairE and GREET to produce embeddings by their published methodologies and then perform the same classification process I use for ID-GNN as discussed in Section

4.3. Results for DGI are reported as publised by the original authors.

CHAPTER 5:

RESULTS

5.1 Tuning

5.1.1 Activation Function

The combination of activation functions was chosen by averaging the accuracy achieved on a tuning subset for all WebKB datasets. Table 5.1 shows only combinations that were in the top 5 for each type of GNN. The range of accuracy for all 81 combinations of activation function was around 20% from the worst combination to the best for both GCN and GIN. Since the type of activation function interacts with the type of GNN used, the tuning results were applied to subsequent tests per GNN, i.e., ID-GCN always uses the combination sigmoid, sigmoid, none, sigmoid and ID-GIN always uses relu, relu, none, relu.

5.1.2 Network Depths

For determining the best combination of the number of layers in ID-GNN when the ID method was GIN, all combinations of depths 2 through 4 were run on a tuning subset of the WebKB datasets. Again, table 5.2 shows that using GCN and GIN for the GNN reacted differently to depths, therefore different defaults are used for subsequent tests. Specifically, ID-GCN always uses 2 layers for the ID (if it includes a GIN) and 4 layers for the GCN while ID-GIN always uses 2 layers for both GNN's.

Table 5.1: Tuning accuracy by activation combination

Activation Functions				GNN Type	
ID	GNN	MLP	Training	GIN-GCN	GIN-GIN
relu	relu	none	relu	59.72	**69.03**
sigmoid	relu	sigmoid	sigmoid	64.61	65.09
none	relu	sigmoid	none	64.81	65.55
none	sigmoid	none	sigmoid	67.21	66.37
sigmoid	none	none	sigmoid	67.21	55.68
sigmoid	relu	none	sigmoid	67.29	56.52
none	relu	none	sigmoid	67.75	53.06
sigmoid	sigmoid	none	sigmoid	**68.21**	66.65

Table 5.2: Tuning accuracy by GNN depth

Depth		GNN Type	
GIN	GNN	GIN-GCN	GIN-GIN
2	2	69.75	**70.10**
2	3	70.13	59.43
2	4	**70.96**	61.25
3	2	65.56	67.35
3	3	65.80	64.75
3	4	64.92	62.32
4	2	65.13	70.00
4	3	65.92	65.29
4	4	65.74	64.77

5.1.3 Dimensionality Reduction

For the four values tested for the number of PCA components to include in the ID, Table 5.3 shows that the range of accuracy was not large and that both GCN and GIN benefited most from including 50 components of the PCA in the ID. Therefore the default for subsequent tuning and tests was fixed at 50 for any submethods which include dimensionality reduction in the ID.

Table 5.3: Tuning accuracy by PCA dimension

DR Dimension	DR-GCN	DR-GIN
5	69.76	69.44
25	70.53	68.26
50	**71.12**	**69.89**
100	70.36	68.50

Table 5.4: Tuning accuracy by hash type

Hash Type	hash-GCN	hash-GIN
SHA256	**54.96**	**55.91**
SHA3	54.39	54.29

5.1.4 Hash Function

Both the GCN and the GIN methods of hash-GNN had higher accuracy when SHA256 was used as the hashing algorithm. However, the benefit was minimal likely due to the similarity of the hash functions tested. Table 5.4 shows the improvement from SHA3 to SHA256 was less than 2% in both cases.

5.2 Experiments

5.2.1 Embedding Source

Table 5.5 shows the average accuracy for node classification obtained during all trials of ID-GNN by the source of the node representations. For most methods, concatenating the ID and the output from the GNN produces the highest results. Note that using the ID alone is more successful than random chance, but does not provide enough differentiation between nodes to produce useful results in all except the dimensionality reduction method. In all other tables, the accuracies shown are those obtained using the concatenation of the ID and the GNN output for classification.

Table 5.5: Accuracy by embedding source

	Source of Embedding		
Method	ID	GNN	ID+GNN
DR-GCN	63.64	57.85	**64.58**
DR-GIN	**62.79**	55.96	61.83
GIN-GCN	42.29	59.08	**60.23**
GIN-GIN	46.14	55.76	**56.76**
GINDR-GCN	43.33	63.49	**64.06**
GINDR-GIN	43.31	54.52	**55.56**
GINhash-GCN	43.24	61.10	**61.76**
GINhash-GIN	43.45	56.28	**57.71**
hash-GCN	33.87	**52.77**	52.58
hash-GIN	33.84	47.16	**47.26**

Table 5.6: Accuracy by classifier type

Method	ExtraTrees	XGBoost
DR-GNN	58.05	**68.77**
GIN-GNN	58.31	**58.68**
GINDR-GNN	59.19	**60.43**
GINhash-GNN	59.54	**59.93**
hash-GNN	49.58	**50.26**

5.2.2 Classifier

For all sub-methods, XGBoost outperforms ExtraTrees as a classifier, in some cases by quite a large margin, as seen in Table 5.6. The widest margin appears for DR-GNN.

5.2.3 GNN Type

All types of ID encoder method were used in both GCN and GIN to compare which aggregation schema creates the best node representation. Universally, GCN outperforms GIN and often by a wide margin as seen in Table 5.7. Therefore, GCN is selected as the GNN for ID-GNN.

Table 5.7: Accuracy by GNN type

	GCN	GIN
DR-GNN	**64.58**	61.83
GIN-GNN	**60.23**	56.76
GINDR-GNN	**64.06**	55.56
GINhash-GNN	**61.76**	57.71
hash-GNN	**52.58**	47.26

Table 5.8: Accuracy on homophilic datasets by ID encoder method

Method	CORA	CiteSeer	PubMed
DR-GCN	75.51	69.71	**85.18**
GIN-GCN	74.03	**73.69**	58.24
GINDR-GCN	77.07	73.28	57.51
GINhash-GCN	**78.98**	65.16	59.69
hash-GCN	42.6	46.99	55.61

5.2.4 ID Encoder Method

Using PCA to reduce the dimensions of the features shows the most promise as a method for creating the IDs to supplement the features and aid the reconstruction of the neighbor features. As seen in Tables 5.8 and 5.9, it achieves the highest accuracy by a large margin for all of the heterophilic datasets and is competitive on PubMed.

Table 5.9: Accuracy on heterophilic datasets by ID encoder method

Method	Cornell	Texas	Wisc.	Actor	Cham.	Squirrel
DR-GCN	**76.65**	**81.56**	**81.12**	**33.00**	**75.91**	**68.44**
GIN-GCN	55.67	58.33	55.97	22.71	74.85	67.37
GINDR-GCN	64.6	69.88	72.35	22.98	72.00	64.73
GINhash-GCN	61.54	69.87	66.15	22.86	71.46	64.64
hash-GCN	50.36	65.54	52.38	25.04	71.47	64.8

Table 5.10: Average classification accuracy compared to state-of-the-art methods on homophilic datasets

Method	CORA	CiteSeer	PubMed
DGI	82.30	71.8	76.8
GREET	**83.81**	**77.40**	80.29
PairE	81.78	75.68	**87.32**
ID-GNN	75.51	70.57	85.18

Table 5.11: Average classification accuracy compared to state-of-the-art methods on heterophilic datasets

Method	Cornell	Texas	Wisc.	Actor	Cham.	Squirrel
DGI	46.48	52.97	55.68	na	na	na
GREET	77.95	**85.70**	81.26	**36.16**	**79.14**	**69.85**
PairE	67.66	69.82	66.14	32.07	63.37	40.86
ID-GNN	**81.37**	80.70	**81.66**	33.00	75.85	67.35

5.3 Comparisons with State-of-the-art

In Tables 5.10 and 5.11 ID-GNN represents the proposed method which deploys a PCA for dimensionality reduction, a GCN for the neural network, using tuned parameters discussed above. DGI values are those published by the original authors, and GREET and PairE accuracies are produced by the same classification methodology employed in this paper.

5.4 Discussion

GCN is a clear winner over GIN for use as the GNN component of ID-GNN. The separate weights it applies to neighbors based on their degree may help it to encode some information about the types of neighbors being aggregated. This would be more effective for heterophilic data than GIN's paradigm of learnable weights for only the ego node.

Looking at the type of ID encoder used, it is notable that dimensionality reduction had the strongest performance though it was actually smaller in dimension than GIN or hash. By retaining some feature information, it can identify the neighbor node with a shorter vector. In combination with GCN's neighbor weights based on strucutre, the differing identities of neighbors seem to be preserved through message passing.

It is clear that ID-GNN in any form does not perform as well on homophilic datasets as any of the state-of-the-art methods, regardless of whether the method is homophily-agnostic or not. Using the most competitive combination of tuning parameters, ID encoder method, and neural network type, ID-GNN is close to matching the performance of state-of-the-art methods which are designed to work with heterophilic graphs. It would be worth exploring whether the pattern in which ID-GNN surpasses GREET on the least heterophilic dataset (Cornell) and is the furthest from GREET on the most heterophilic dataset (Texas) holds true upon further inspection.

While the goal was to create a method which performs well whether the dataset is homophilic or heterophilic, it would be interesting further work to tune parameters by type of dataset in order to hone ID-GNN's ability to perform well specifically on heterophilic graphs. Additionally, when reverse engineering GREET, I discovered that the accuracies reported within their paper were based on performing classification periodically during training. Whereas ID-GNN (and PairE) produce an embedding based on lowest loss and then use it for classification, GREET uses the embedding which produces the highest accuracy. In a truly unsupervised setting, this would not be available. After exploring the performance of GREET using loss to determine the best model state, I believe ID-GNN method could surpass GREET and produce publishable results.

CHAPTER 6:

CONCLUSION

I have presented ID-GNN, a new method for training unsupervised representations on heterophilic graph data. By generating ID codes for each node using PCA of the features, and using these IDs as flags to follow feature information through a GCN, the IDs of neighbor nodes aid in the reconstruction of neighbor features using the ego node representation. This produces node embeddings which allow for a high likelihood of different-class neighbors, meaning ID-GNN is near state-of-the-art as measured by accuracy on node classification.

BIBLIOGRAPHY

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[2] Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-Mellon University Pittsburgh PA School of Computer Science, 1998.

[3] Ogi Djuraskovic. Big data statistics 2022: How much data is in the world?, Jan 2022.

[4] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[5] Junchen Jin, Mark Heimann, Di Jin, and Danai Koutra. Toward understanding and evaluating structural node embeddings. *ACM Trans. Knowl. Discov. Data*, 16(3), nov 2021.

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[7] You Li, Bei Lin, Binli Luo, and Ning Gui. Graph representation learning beyond

node and homophily. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4880–4893, 2022.

[8] Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent CS Lee, and Shirui Pan. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 4516–4524, 2023.

[9] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

[10] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[11] Ryoma Sato. A survey on the expressive power of graph neural networks, 2020.

[12] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[13] Mingyue Tang, Pan Li, and Carl Yang. Graph auto-encoder via neighborhood wasserstein reconstruction. In *International Conference on Learning Representations*, 2022.

[14] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax, 2018.

[15] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[16] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[17] Liang Yang, Junhua Gu, Chuan Wang, Xiaochun Cao, Lu Zhai, Di Jin, and Yuanfang Guo. Toward unsupervised graph neural network: Interactive clustering and embedding via optimal transport. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1358–1363, 2020.

[18] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

APPENDIX A:

ARTIFACTS

- Code Repository
- Full Results