Software Engineering Department
Braude College

Capstone Project Phase A
24-1-R-10

Evaluation of edges readiness using masked graph autoencoders - MASKGAE

Students:

Gilad Segal, gilad.segal@e.braude.ac.il

Chen Hanzin, chen.hanzin@e.braude.ac.il

Supervisors:

Dr. Renata Avros

Prof. Zeev Volkovich

Table Of Contents

Abstract

This study explores a new way to enhance how we learn from complex network data using a method called Masked Graph Autoencoding (MaskGAE). Unlike traditional approaches, MaskGAE improves learning by selectively hiding parts of the network (like connections between nodes) and then trying to figure out what was hidden using the remaining visible data.

Our main goal is to understand which parts of a network structure are effective and which parts are redundant (repeating themselves during the process). We do this by hiding certain parts of the network (sub-graphs), analyzing how the network behaves with these parts missing, and then attempting to reconstruct the hidden parts. By repeating this process, we can determine how well the network's structure is based on how accurate it can replicate the hidden elements consistently.

From a theoretical standpoint, we connect our method with established learning theories, showing that MaskGAE enhances how networks learn on their own. We test our approach on various types of network data, comparing it with other leading methods. Our tests show that MaskGAE not only performs well but often exceeds other methods in tasks like predicting missing connections and classifying nodes within the network.

Through this research, we aspire to provide a clearer understanding of how masking parts of a network can lead to better data analysis and learning outcomes, offering a significant improvement over traditional methods.

1. Introduction

In recent years, self-supervised learning has emerged as a leading methodology within the domain of graph neural networks (GNNs), facilitating the extraction of meaningful representations from unlabeled data without the need for specific tasks. This approach is based on the creation of auxiliary tasks (or pretext tasks) which explore data directly for the extraction of supervisory signals [1]. Contrastive learning is a technique that involves comparing different perspectives of the same dataset to ensure a consistent understanding of the information across various views. However, the effectiveness of these methods often depends on complicated pretext tasks and data augmentation methodologies to capture various structural nuances [2].

Another promising field of self-supervised learning includes the utilization of graph autoencoders (GAEs), a subset of models dedicated to reconstructing graph structures solely from input graph data. While GAEs offer streamlined implementation, their tendency to overly emphasize local information poses challenges beyond fundamental tasks like link prediction, requiring the exploration of more versatile pretext tasks [3].

Additionally, masked autoencoding, a technique for unveiling hidden patterns within data, has drawn significant attention for its transformative impact on language and image processing domains. Despite its widespread application, particularly through methods like masked language modeling (MLM) and masked image modeling (MIM) exemplified by BERT and MAE, its potential within graph-related contexts remains largely unexplored. Masked autoencoding has advantages in analyzing graph structures, as it allows for simple masking or unmasking of components, thereby presenting a compelling opportunity to enhance the performance of GAEs in self-supervised learning tasks.

In this work, we plan to build upon the success of masked language modeling (MLM) and masked image modeling (MIM) by introducing masked graph modeling (MGM) as a principled pretext task for graph-structured data. MGM revolves around the core idea of selectively removing a portion of the input graph and training the model to predict the removed content, such as edges. Building upon this concept, we propose masked graph autoencoder (MaskGAE), a self-supervised learning framework that incorporates the principles of masking and prediction through node and edge-level reconstruction. This framework demonstrates the advantages of MGM in enhancing the self-supervised learning paradigm of GAEs.

Specifically, we reveal that the learning objective of GAEs is similar to contrastive learning, where paired subgraphs naturally form two structural views for contrast. Importantly, by masking edges, our MaskGAE framework reduces redundancy in the contrastive subgraph views within GAEs, thereby improving the effectiveness of contrastive learning.

2. Background and Related Work

2.1 Contrastive learning

Contrastive Learning is a deep learning technique for unsupervised representation learning. The essence of this approach is to create different augmentation views for maximizing the mutual information between different representations of data.

For graphs, contrast learning is considered a prominent paradigm for having modified versions of the original graph data instances (such as nodes, subgraphs, or entire graphs) to generate multiple views of the same instance. These versions are designed to preserve the essential properties or features of the data while introducing variations that a model should learn to recognize as fundamentally the same, regarding structure or pattern.

2.2 Graph autoencoders

Graph Autoencoders (GAE) are a group of generative models that work according to the **encoder-decoder framework.** GAE models are trained on an incomplete version of datasets where parts of the edges have been removed, while all node features are kept intact [3]. removed edges are often defined as negative edges while the untouched edges are defined as positive edges.

2.2.1 Encoder

The **encoder** is the first step for the GAE model. It maps the nodes (often with complex high-dimensional information) into more compact features that are learned through a process of dimensionality reduction. These latent representations aim to capture the essential information of the graph in a more manageable form, often referred to as node embeddings.

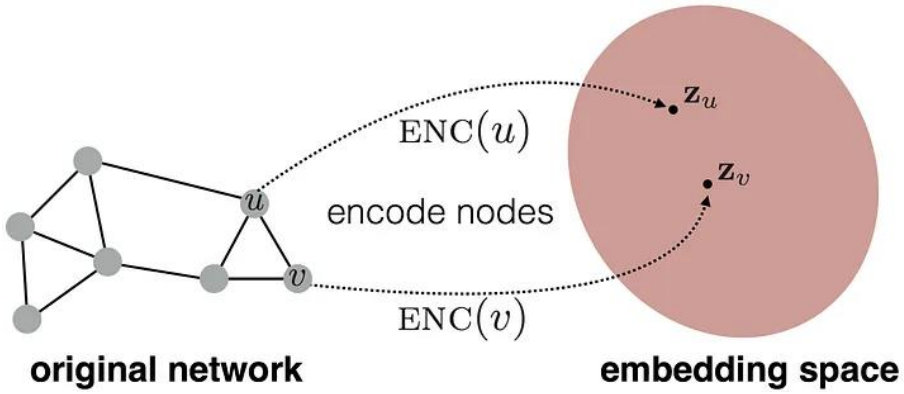


Figure 1. Example of the encoder function [5]

In the past, GAE's were considered not effective in learning node representations and thus leading to over-emphasize proximity information at the expense of structural information [4]. This was due to the common graph models that mostly fixated on the concept of random walks for node embeddings. Today, more and more encoders in GAE's are using well-established GNN architecture to improve the self-supervised learning scheme of GAEs.

2.2.2 Decoder

After the encoder generates node embeddings, the decoder reconstructs the graph using those embeddings. The decoder predicts the relationship or similarity between pairs of nodes in the graph. Thus, for each two nodes *U* and *V*, their pair embeddings are passed to the decoder which outputs the likelihood for a connection between these two nodes.

2.2.3 Optimizing the encoder-decoder framework

The encode-decoder framework can be optimized using different methods. The most common method is by using **reconstruction loss** which is defined as how well the

decoder can predict the similarity between two nodes in the graph. To optimize the model, the goal is to minimize the reconstruction loss, using binary cross-entropy loss ($\mathcal{L}_{GAEs}$):

$$\mathcal{L}^{+} = \frac{1}{|\varepsilon^{+}|} \sum_{(u,v)\in\varepsilon^{+}} log h_{\omega}(z_u, z_v),$$

$$\mathcal{L}^{-} = \frac{1}{|\varepsilon^{-}|} \sum_{(u',v')\in\varepsilon^{-}} \log\left(1 - h_{\omega}(z_{u'}, z_{v'})\right),$$

$$\mathcal{L}_{GAEs} = -(\mathcal{L}^{+} + \mathcal{L}^{-})$$

Equation 1. binary cross-entropy loss [7]

z is the node representation obtained from the encoder.

ε^{+} is a set of positive edges.

ε^{-} is a set of negative edges.

h_{ω} is the decoder with parameters ω.

2.3 Masked autoencoding

Masked autoencoding is a widely known learning task designed for different approaches of data representation. Mostly, this task is used in natural language processing with Masked Language Modeling (MLM) and in image processing with Masked Image Modeling (MIM). The main principle of the masked autoencoder (MAE) is to remove certain data instances (words in MLM and pixels in MIM) and learn how to reconstruct the missing content through predictions.

The core design of masked autoencoding is prominently based on the encoder-decoder framework. The encoder receives only the visible subset of data instances, after others were removed during the process of masking; and the decoder is supposed to reconstruct the original content from the latent representations. In both image and language, the encoder's function is the same as before [3.2.1]: preparing the dimensionally reduced data representations. For example, the masking in vision processing is hiding vast portions of pixels so the decoder's objective is to output those pixels back. In language, the task involves masking words, leading to a decoder that predicts missing words that contain semantic information.

While there is a widespread use of masked autoencoding in language and vision research, this learning task has been relatively unexplored in the field of graphs. Only few efforts were made recently to address this gap by applying masked autoencoding to graph data as a self-supervised learning approach. These efforts require masking

strategies that take into account graph components such as nodes, edges and even sub-graphs.

2.4 GCN - Graph convolutional networks

Graph Convolutional Networks (GCNs), is a specific type of Graph Neural Network (GNN) that uses convolutional operations to propagate information between nodes in a graph. GCN examines the local neighborhoods within a graph, similar to how Convolutional Neural Networks (CNNs) analyze image regions. GCNs are proficient at identifying meaningful patterns in these neighborhoods, which is crucial for tasks such as node classification or relationship prediction between nodes.

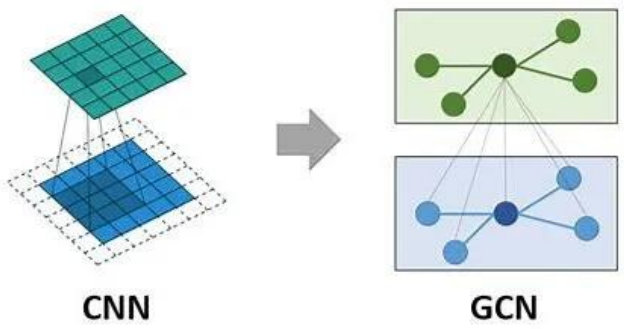


Figure 2. Graph Convolutional Networks [6]

2.4.1 Graph convolutional networks setup

Graph Convolutional Networks are structured to manage data represented as graphs. For example, In a citation network, nodes represent academic papers, and edges represent citations between these papers. The goal is often to classify each paper into a category based on its content and citation patterns, such as "Computer Science," "Physics," "Mathematics," etc.

Data Setup: The first step is data preparation and it involves formatting the graph into a representation suitable for GCN. This typically includes creating a node adjacency matrix (A) that outlines the graph's topology (connections between nodes) and a node feature matrix (X) that details the attributes of each node (such as text embeddings or bag-of-words representations) [3].

Convolution Operations: After preparing the data, the GCN can perform specialized convolution operations on the node feature matrix using adjustable weight matrices. For instance, the feature representation of a paper in a citation network might be updated by summing the features of papers that cite it, weighted by the learned parameters of the GCN.

Non-linear Activation: Following convolution, GCNs apply a non-linear activation function (such as ReLU) to the transformed node features, enabling the network to capture more complex patterns and relationships.

Training and Prediction: Through backpropagation and using loss functions, GCN is trained on tasks like node classification or link prediction. Once trained, they can predict outcomes on new graph data by employing the same convolution and aggregation techniques.

3. Expected Achievements

The anticipated goals of this project involve creating a smart learning model that can improve self-learning for graphs. The model's main goal is to develop a new algorithm called MaskGAE, designed to mitigate issues associated with data redundancy. We're going to check how effective the model is at rebuilding hidden connections in the graph. We'll mask some connections and see if the model can find them correctly. This will help us see how well the model understands the graph.

4. Research Process

4.1 Exploring MaskGAE

In recent advancements in graph machine learning, the MaskGAE model has emerged as a promising approach for learning robust graph representations through the innovative use of masking strategies. During our research we studied the potential of this model and how it got implemented in the domain of citation networks. We familiarize ourselves with the key ingredients of the model and learned how it accurately predicts the structure of these networks after selective masking of their components.

This study will focus on several critical research questions: How effectively can the MaskGAE model reconstruct masked citation networks? What are the optimal masking strategies for maximizing model performance in this specific context? By delving into these questions, We plan to not only implement the MaskGAE model efficiently but also enhance its applicability and accuracy in predicting missing or unseen parts of the graph. Through this research, we intend to contribute to the broader understanding of masked graph autoencoders and their potential in complex network analysis, building on the foundational theories and recent empirical findings in the field.

4.1.1 MaskGAE advantages (vs GAE)

Graph Autoencoders (GAEs) have proven effective in learning node representations within graphs by maximizing mutual information among nodes. However, GAEs often exhibit a significant flaw: they tend to capture excessive similarities between subgraphs, leading to redundancy that can disrupt the learning process. This

redundancy may result in the model overemphasizing information that does not necessarily aid in making accurate predictions or noticing future behaviors within the graph.

To handle these challenges, Masked Graph Modeling (MGM) introduces a strategic approach by selectively removing certain edges from the graph. This process aims to produce fewer overlapping subgraphs, thereby mitigating the issue of redundancy observed in traditional GAEs.

For instance, consider an edge (a, b) which is identified as a positive connection. By removing this edge, the resulting subgraphs focused on nodes a and b respectively will exhibit reduced overlap, with fewer common nodes and edges. This reduction in overlap helps in diversifying the information captured by the model, enhancing its ability to learn more distinct and useful representations that are important for the tasks at hand. The MaskGAE model leverages this approach to effectively minimize redundancy, offering a more refined and potentially more accurate graph analysis tool compared to standard GAE.

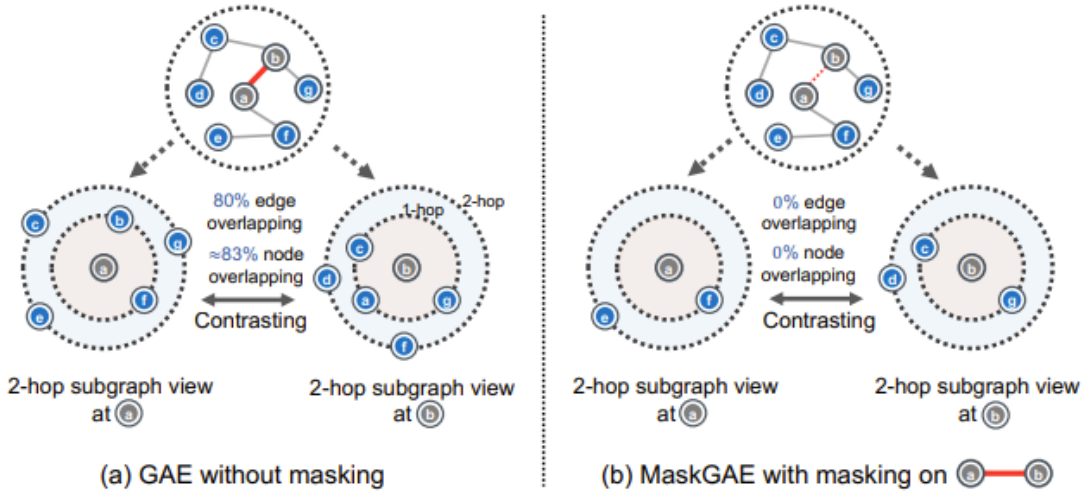


Figure 3. Benefits of MaskGAE [7]

4.1.2. Potential challenges

Exploring the complexity of implementing MaskGAE, a novel approach to self-supervised learning on graphs, reveals several potential challenges. One significant issue is the risk of masking distorting the inherent meanings of specific graphs, which could compromise the accuracy of the model's learned representations. Additionally, MaskGAE operates on the assumption that connected nodes generally share similar labels, an assumption that may not always be true in complex real-world networks, thereby challenging the efficiency of the model.

Moreover, while self-supervised learning on graphs such as MaskGAE has advantages across various domains (including financial networks, healthcare

systems, and social media analysis), it also raises significant privacy concerns. The nature of these models, which learn detailed representations of graph data, could potentially lead to the exposure of sensitive information if not handled correctly.

For example, in a social network, even if direct attributes like names and locations are masked, the structure of the graph itself could reveal personal relationships or preferences that are considered private. This issue is common in financial networks, where the unauthorized prediction or revelation of connections and trends could have serious implications for market integrity or individual privacy.

4.2 MaskGAE framework

MaskGAE is a straightforward framework based on the asymmetric design of encoder-decoder architecture. The framework is similar to the well-known GAE framework with mild additions which are designed to reduce redundancy between pairs of subgraphs in the network. Thus, MaskGAE is combined from four critical components: masking strategy, encoder, decoder, and learning objective.

Masking Strategy

Prominent approaches for masking in graphs are focusing on edge masking as the optimal way to facilitate the Masked Graph Modeling (MGM) task. There are two approaches that output the masked graph and thus are appropriate for this task.

The first approach is plain and based on edge masking: direct selection of a subset of edges, using a specific probability value to decide to include it as part of the set. In the presented algorithm of MaskGAE, the probability function is the Bernoulli distribution, and we define the masking ratio as ‘p’ ($p<1$) and the set of edges is defined as: $\mathcal{E}_{mask} \sim Bernoulli(p)$.

The second approach is wider and considers a whole path of edges in the graph (Path-based Masking). A path in a graph is a series of consecutive edges that joins a sequence of neighboring nodes. In this approach, the path of edges originates from a “root node” which functions as a starting node of the path. The root nodes themselves are sampled with a specific distribution in a similar manner to the first approach. From each root node, a path is created using random walk: moving past the root node neighbors, where in each step a neighbor could be taken into the path under a certain probability.

In this approach the root nodes are denoted as ‘R’ and sampled using the Bernoulli distribution with a sample ratio ‘q’, defined as: $R \sim Bernoulli(q)$; and the set of edges is defined as: $\mathcal{E}_{mask} \sim RandomWalk(R, l_{walk})$. l_{walk} represents the random walk length.

Encoder

The encoder receives as input the visible, unmasked sub-graph and it views the graph in small pieces at a time. It maps the nodes to low dimensional representations (embeddings) for the decoder.

The embeddings learned by the encoder are extracted for each node in the test set. These embeddings serve as input features for a linear classifier, such as logistic regression. The linear classifier is trained using the embeddings of the labeled nodes in the training set.

The encoder is a specific type of neural network architecture called Graph Convolutional Networks (GCN) that is a type of GNN (Graph Neural Network) architecture. Two GCN layers are applied for the encoder which outputs node embeddings denoted as z.

Decoder

The decoder is a combination of two separate decoders: structure decoder and degree decoder. The structure decoder predicts connections between nodes in the graph and the degree decoder assists in understanding the number of connections each node has. According to the MaskGAE setup, the decoder main operation lies in the structure reconstruction whereas the degree decoder is mostly designed to assist the model in training, by forcing it to approximate the node degree.

The structure decoder receives the node embeddings from the encoder, and for each pair of them (z_u, z_v) it performs an element-wise product of the vectors. The result is transferred to MLP (multilayer perceptron) for learning complex patterns and relationships. Afterwards, the output from the MLP is transferred to the Sigmoid function as input for binary classification of link connection. Eventually the structure decoder provides the probability for a connection between every two nodes.

$$h_\omega(z_u, z_v) = Sigmoid(MLP(z_u \circ z_v))$$

Equation 2. structure decoder [7]

For the degree decoder there is only use of MLP to generate the number of neighbors for each node. The graph itself has a lot of information about the number of connections each node has. So, the model uses the degree decoder to make sure the model is learning this aspect as well.

$$g_\Phi(z_v) = MLP(z_v)$$

Equation 3. degree decoder [7]

Learning Objective

The combined objective of MaskGAE is to find a balance between reconstructing the masked graph's edges, using **Reconstruction Loss**, and accurately predicting node

degrees, using **Regression Loss**. To regulate the trade-off between these two losses, the model uses new parameter denoted as α. This parameter influences the regularization aspect of the learning.

The reconstruction loss evaluates how well the model reconstructs the masked graph, specifically focusing on the edges that were masked in the original graph. This loss is similar to the one used in traditional GAE (the binary cross-entropy loss), but in this case, it considers only the masked edges ($\mathcal{E}_{mask}$). A common function to implement this method is Mean Square Error (MSE) which takes the difference between the model's predictions and the original degree, square it, and average it out across the whole dataset.

$$\mathcal{L}_{deg} = \frac{1}{|V|}\sum_{v \in V} \left|\left| g_{\Phi}(z_v) - deg_{mask(v)} \right|\right|^2$$

Equation 4. regression loss [7]

$g_{\Phi}(z_v)$ is the degree decoder.

$deg_{mask(v)}$ is the original degree of node v

$$\mathcal{L} = \mathcal{L}_{GAEs} + \alpha\mathcal{L}_{deg}$$

Equation 5. learning objective [7]

$\mathcal{L}_{deg}$ contributes to the regularization, i.e., guides the model to learn more abstract representations that are beneficial for tasks beyond the training data. This helps prevent the model from becoming overly specialized (basically to prevent overfitting) and enhances its ability to handle a variety of graph structures effectively.

Algorithm Masked Graph Autoencoder (MaskGAE)

Input: Graph G=(V,E) , encoder $f_{\theta}(\cdot)$, structure decoder $h_{\omega}(\cdot)$, degree decoder $g_{\Phi}(\cdot)$, masking strategy M $\epsilon\{$ M_{edge} , M_{path} $\}$, hyperparameter α.

Output: Learned encoder $f_{\theta}(\cdot)$;

1. **While** not converged **do**;
2. $G_{mask} \leftarrow$ M(G);
3. $G_{vis} \leftarrow G - G_{mask}$;
4. $Z \leftarrow f(G_{vis})$;
5. Calculate L_{GAEs} over G_{mask} according to Eq.(1);
6. Calculate L_{deg} over G_{mask} according to Eq.(10);
7. Calculate loss function L$\leftarrow L_{GAEs}$ + α L_{deg};
8. Update θ, ω, ϕ by gradient descent;
9. **end while;**
10. **return** $f_{\theta}(\cdot)$;

Figure 4. MaskGAE Algorithm [7].

4.3 Datasets statistics

Our research builds upon the foundational work conducted by previous researchers on the MaskGAE model, particularly their detailed methodology in data collection and preprocessing [7]. As we prepare to advance this body of knowledge, we are particularly influenced by their use of diverse and robust datasets, which have demonstrated significant relevance in various domains. Among these datasets they used renowned citation networks such as CORA, CiteSeer, PubMed, arXiv, MAG, and Collab alongside co-purchase graphs like Photo and Computer.

Dataset	#Nodes	#Edges	#Features	#Classes	Density
Cora	2,708	10,556	1,433	7	0.144%
CiteSeer	3,327	9,104	3,703	6	0.082%
Pubmed	19,717	88,648	500	3	0.023%
Photo	7,650	238,162	745	8	0.407%
Computer	13,752	491,722	767	10	0.260%
arXiv	16,9343	2,315,598	128	40	0.008%
MAG	736,389	10,792,672	128	349	0.002%
Collab	235,868	1,285,465	128	-	0.002%

Figure 4. Dataset Statistics [7].

4.3.1 Data collection

Inspired by their approach, we are focusing our efforts on specific citation networks, primarily the CORA and the Pubmed datasets, which have been considered in earlier studies. These datasets offer not only a broad application spectrum but are also critical for a comprehensive evaluation of the MaskGAE model's performance. Our next step will be to follow a structured approach to data management, replicating the preprocessing steps that have proven effective in prior research. These steps include feature normalization, necessary modifications to the graph structure, and meticulous division of the datasets into training, validation, and testing sets.

Additionally, we will thoroughly address any data integrity issues such as missing or incorrect data points to ensure the highest quality and reliability of our datasets. Our objective is to prepare a clean and well-structured dataset, particularly from the CORA and PubMed citation networks, to effectively train and evaluate the MaskGAE model. By doing so, we aim to explore new dimensions within these specific domains, potentially uncovering novel insights that could contribute to the broader understanding of graph autoencoders.

4.3.2 Graph representation and preprocessing

The dataset is represented as a graph, where each learned article acts as a node, and citations between articles are edges connecting these nodes. This creates a network of interconnected nodes, forming a visual graph representation. Unique identifiers or

labels are assigned to each article, and corresponding nodes are created in the graph to represent them. Citations between articles are modeled as directed edges, indicating the direction of the citation relationship.

For example, if article A cites article B, there is a directed edge from the node representing article A to the one representing article B. Additionally, attributes such as publication year, authors, keywords, and other metadata associated with each article can be incorporated as node or edge attributes. These attributes define the graph representation, providing contextual information for effective utilization by the subsequent link prediction algorithm.

Preprocessing steps are then applied to the graph to ensure its suitability for training and evaluation. These steps include feature normalization, graph structure modifications, and data splitting into training, validation and test sets. They aim to handle missing or not accurate data points, enhancing the dataset's quality and reliability for analysis and model training.

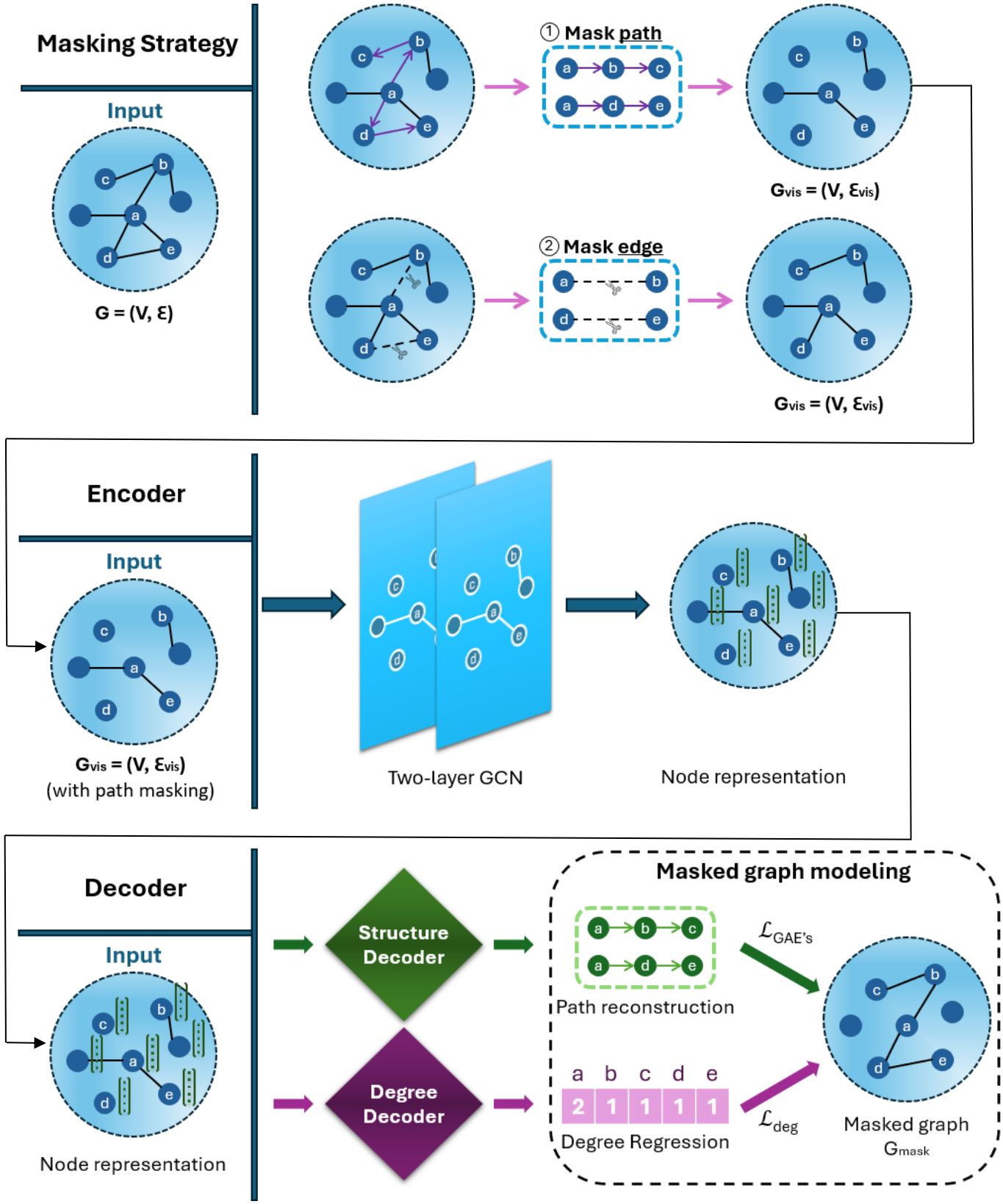
4.4 Training process

Training the MaskGAE model involves two primary tasks: link prediction and node classification.

In the link prediction task, the model aims to predict the presence or absence of edges between pairs of nodes in a graph. The process involves iterations over mini-batches of masked edges and computes the loss using cross-entropy loss (correctly identifying actual edges). Also, the model parameters are optimized using gradient-based optimization algorithms, such as SGD or Adam. This process continues iteratively until convergence. The metrics that are used in the process are ROC AUC score and Average Precision score are used to evaluate the model's performance on link prediction.

For node classification, the model predicts class labels or properties associated with individual nodes in the graph. The training process involves optimizing a suitable loss function, such as cross-entropy loss. The model's parameters are updated iteratively based on the computed gradients, aiming to minimize the difference between predicted class probabilities and true labels. Most prominent performance metrics that are used are accuracy and precision to evaluate the model's node classification performance.

4.5 MaskGAE flow diagram



5. Evaluation/Verification Plan

To effectively evaluate the performance of the MaskGAE model, we will perform two primary tasks: link prediction and node classification. Each task will have specific setups for sampling, training, and evaluation metrics.

5.1 Link prediction

Setup and Sampling: We will prepare the data for the link prediction task, by sampling 10% of existing edges from each dataset to serve as positive examples. An equal number of non-existent edges (unconnected node pairs) will also be sampled to serve as negative examples, ensuring a balanced dataset for testing.

Model Training: The model will be trained through iterations over mini-batches of these masked edges, where the primary objective is to predict whether a link (edge) should exist between pairs of nodes. The training will use cross-entropy loss to evaluate the accuracy of the edge predictions.

Optimization: Gradient-based optimization algorithms, such as Stochastic Gradient Descent (SGD) or Adam, will be used to adjust the model parameters. This process continues iteratively until convergence.

Evaluation Metrics:

- ROC AUC Score: This metric will measure the model's ability to distinguish between the classes of linked and non-linked node pairs over the graph network. A higher ROC AUC score indicates better discrimination, with 1.0 representing perfect result and 0.5 suggesting no better than random guessing.
- Average Precision Score: This will summarize the precision-recall curve as the weighted mean of precision achieved at each threshold. It reflects the model's ability to correctly identify positive (actual linked) pairs across different recall levels, crucial for handling unbalanced data where positive links are less common.

5.2 Node classification

Setup and Training: Initially, the model will be trained on a graph using self-defined supervisions and pretext tasks, which help in guiding the learning process towards useful node representations.

Feature Concatenation: Post initial training, the output from each layer of the model will be concatenated to form a comprehensive node representation. This enriched feature set captures multi-scale information encoded at different layers of the network.

Model Evaluation: The evaluation will involve fitting a logistic regression model on the frozen learned embeddings, ensuring that no gradients flow back to the encoder.

This step estimates the quality of the embeddings in terms of their utility for classification without further adapting the model.

Evaluation Metrics:

- Accuracy: This metric will measure the percentage of correctly predicted labels over the total number of nodes in the test set. It provides a straightforward indicator of the model's overall effectiveness in correctly classifying nodes.
- Precision: This will assess the proportion of positive identifications that were actually correct, particularly important in situations where the cost of a false positive is high. It helps in understanding the reliability of the predictions in terms of class-specific performance.

5.3 Test plan

Data Preparation: Organize the dataset by separating it into a training, validation and testing sets. Use the training set to teach the model, and the testing set to check how well the model works. Make sure the testing set includes a balanced mix of both relevant and irrelevant citations.

Model Training: Use the training set to educate the link prediction model. This step involves giving the model the necessary data and labels (whether citations are relevant or not). The model will learn to recognize patterns and relationships between citations to make informed predictions.

Prediction Generation: Use the model trained with the training set to predict outcomes on the testing set, determining whether citations are relevant or irrelevant. The model will provide a probability or a straightforward yes/no prediction for each citation in the testing set.

Evaluation Metrics Calculation: Measure the model's effectiveness using predefined metrics (accuracy, precision, ROC AUC Score, etc.). This involves comparing the model's predictions to the actual outcomes in the testing set and using specific formulas to calculate these metrics.

Interpretation and Analysis: Review the calculated metrics to understand how well the model performs. Evaluate these metrics to determine the model's effectiveness at distinguishing between irrelevant and relevant citations.

Fine-tuning and Iteration: If the metrics suggest improvements are needed, adjust the model's settings, select different features, or incorporate more data. Repeat the training and testing cycle to see how these changes affect the model's accuracy.

Reporting and Conclusion: Summarize the findings into a detailed report. Include the performance metrics, discuss what the results mean, and conclude how capable the model is at predicting the relevance of citations in academic articles.

6. Future Plan

As we progress with our research on the MaskGAE model, our future plan is designed to build upon the existing findings and extend our understanding of this new approach to graph analysis. Our immediate next steps involve refining the model's architecture and exploring its applicability across different types of graph datasets, particularly focusing on citation networks like CORA and potentially expanding to others such as PubMed.

We plan to optimize the MaskGAE model’s parameters and masking strategies to better accommodate the specific characteristics of different graph structures. This includes experimenting with various rates and patterns of masking to determine the optimal configuration that maximizes model performance in both link prediction and node classification tasks.

7. References

[1] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self supervised learning on graphs: Contrastive, generative, or predictive. TKDE (2021).

[2] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In ICML, Vol. 119. PMLR, 4116–4126.

[3] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. CoRR abs/1611.07308 (2016).

[4] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. ICLR (Poster) 2, 3 (2019), 4.

[5] Graph Neural Networks Series | Part 3 | Node embedding The Modern Scientist

[6] https://python.plainenglish.io/graph-convolutional-network-simplified-88028e87c04d

[7] Li, J., Wu, R., Sun, W., Chen, L., Tian, S., Zhu, L., Meng, C., Zheng, Z., & Wang, W. (2023). What's behind the mask: Understanding masked graph modeling for graph autoencoders. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23) (pp. 1-13). ACM.