Software Engineering Department
Braude College

Capstone Project Phase B
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

1. General Description

The project focuses on improving Graph Autoencoders (GAEs), a type of machine learning model that learns from graph data. GAEs work by simplifying complex graphs into low-dimensional representations (compressed versions), and then trying to rebuild the original graphs as accurately as possible. However, traditional GAEs sometimes concentrate too much on the direct connections between closely linked nodes, which can lead to missing the overall structure of the graph.

This project explores a new method called Masked Graph Modeling (MGM) to help GAEs better understand the entire graph, not just the closely connected parts. The goal is to make GAEs more effective in capturing the big picture of the graph while still preserving important details.

2. Implementation

To implement this MGM task, the model is built upon the well-known encoder-decoder framework, combined with a classical group of generative models which are the GAE's family. This approach enables the model to perceive an input of a neural network graph in a way of encoding nodes to low dimensional representations, and later, reconstructing (decoding) the graph.

The encoder is initialized as a type of GNN architecture (Graph Neural Network) and tested on various choices from this architecture (GCN, SAGE, GAT from torch_geometric [7]) for implementing the message passing process. As a default option, the encoder's operation is set to the Graph Convolutional Networks (GCN).

For the Decoder, the model actually contains 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. Both decoders utilize MLP architecture to make predictions based on node features. They also use dropout and activation functions (like ReLU) to learn from node features and make predictions.

As discussed in Part A of our project, the superiority of the MaskGAE over the traditional GAE model is embodied in the masking strategy. The implementation of this component in the model is by random walks from the torch_cluster library, which selectively masks edges based on local graph structures. Additionally, a Bernoulli distribution is employed in the masking process to probabilistically determine which edges to mask, ensuring a balance between retaining and removing edges during training.

Eventually, the project involves developing and testing this model to see if it improves the accuracy and effectiveness of GAEs for tasks like predicting relationships between

nodes or classifying them into categories. In literature, the model is tested on various datasets, such as Cora, CiteSeer, Pubmed, arXiv, MAG, and Collab alongside co-purchase graphs like Photo and Computer [6].

3. Solution

3.1 Structure

The system is based on the MaskGAE framework, which modifies the traditional Graph Autoencoder (GAE) approach by incorporating Masked Graph Modeling (MGM). The key components of this structure include:

- **Encoder-Decoder Framework**: The encoder (GCN) processes the masked graph, producing a compact, low-dimensional representation of each node. The decoder then attempts to reconstruct the original graph, including the masked edges, from these representations.
- **Graph Convolutional Network (GCN)**: The encoder of the system uses a Graph Convolutional Network, a well-known architecture for processing graph data. GCN effectively captures the local structure of a graph by aggregating information from a node's immediate neighbors. This helps in creating low-dimensional representations of the nodes, which preserve important local features.
- **Masking Strategy**: A unique feature of the structure is the masking of certain edges in the graph. By hiding parts of the graph, the system forces the encoder to learn richer representations that are not overly dependent on direct connections.
- **Learning Objective**: The system is designed to understand both the local connections (as captured by GCN) and the global structure of the graph. This dual focus ensures that the model doesn't just focus on nearby nodes but also understands the broader relationships across the entire graph.

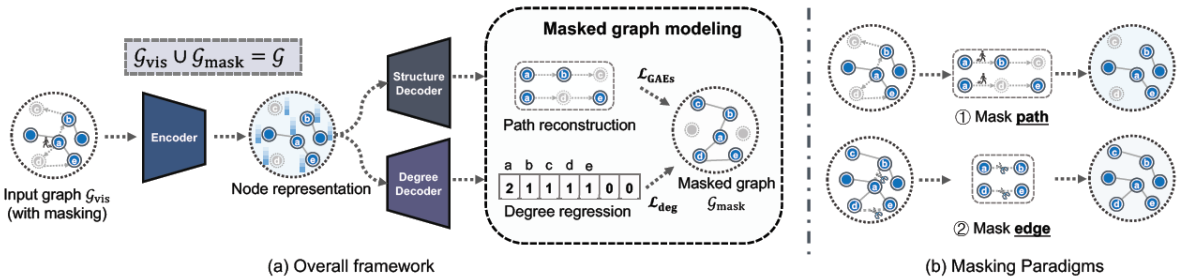


Figure 1. MaskGAE Structure[6]

3.2 System Setup

The MaskGAE system improves traditional Graph Autoencoders (GAEs) by using Masked Graph Modeling (MGM) to capture both local and global graph structures. By strategically masking parts of the graph, the model learns to reconstruct missing elements, reducing redundancy and improving its representation of complex graphs. This approach combines edge-wise and path-wise masking with a strong encoder-decoder framework built on Graph Convolutional Networks (GCNs) and Multi-Layer Perceptrons (MLPs), creating a more effective solution for graph-based tasks. To explore deeper into the core workings of this project, we will now examine the technical aspects that form the foundation of the system.

3.2.1 Masking Strategy

Edge-wise Random Masking (T_{edge}):

The purpose is to create a masked version of the graph by randomly masking a subset of edges. This edge mask is done by sampling edges using a Bernoulli distribution with a masking ratio P<1. The masked edges E_{mask} are a subset of the original edges E, and the masking process is denoted as $G_{mask}=T_{edge}(G)$.

Path-wise Random Masking (T_{path}):

The purpose is to create a masked graph by masking paths instead of individual edges, breaking short-range connections and forcing the model to infer missing information based on more complex patterns. The path-wise masking strategy involves random walks starting from a set of root nodes R, sampled using a Bernoulli distribution with a sample ratio q. The walk length is defined as l_{walk}. Masked edges Emask are generated through these random walks, with the process denoted as $G_{mask}=T_{path}(G)$.

3.2.2 Encoder-Decoder Framework

Encoder:

The encoder is composed of two Graph Convolutional Network (GCN) layers. It processes the masked graph G_{mask} and generates low-dimensional representations Z of the nodes, capturing both local node features and the broader graph structure.

Decoder:

The decoder consists of two Multi-Layer Perceptron (MLP) layers, specifically designed for both structure and degree reconstruction. It takes the encoded representations Z and attempts to reconstruct the original graph, including the masked

parts. This includes both reconstructing the connectivity (structure decoder) and the node degrees (degree decoder) to ensure a comprehensive graph recovery.

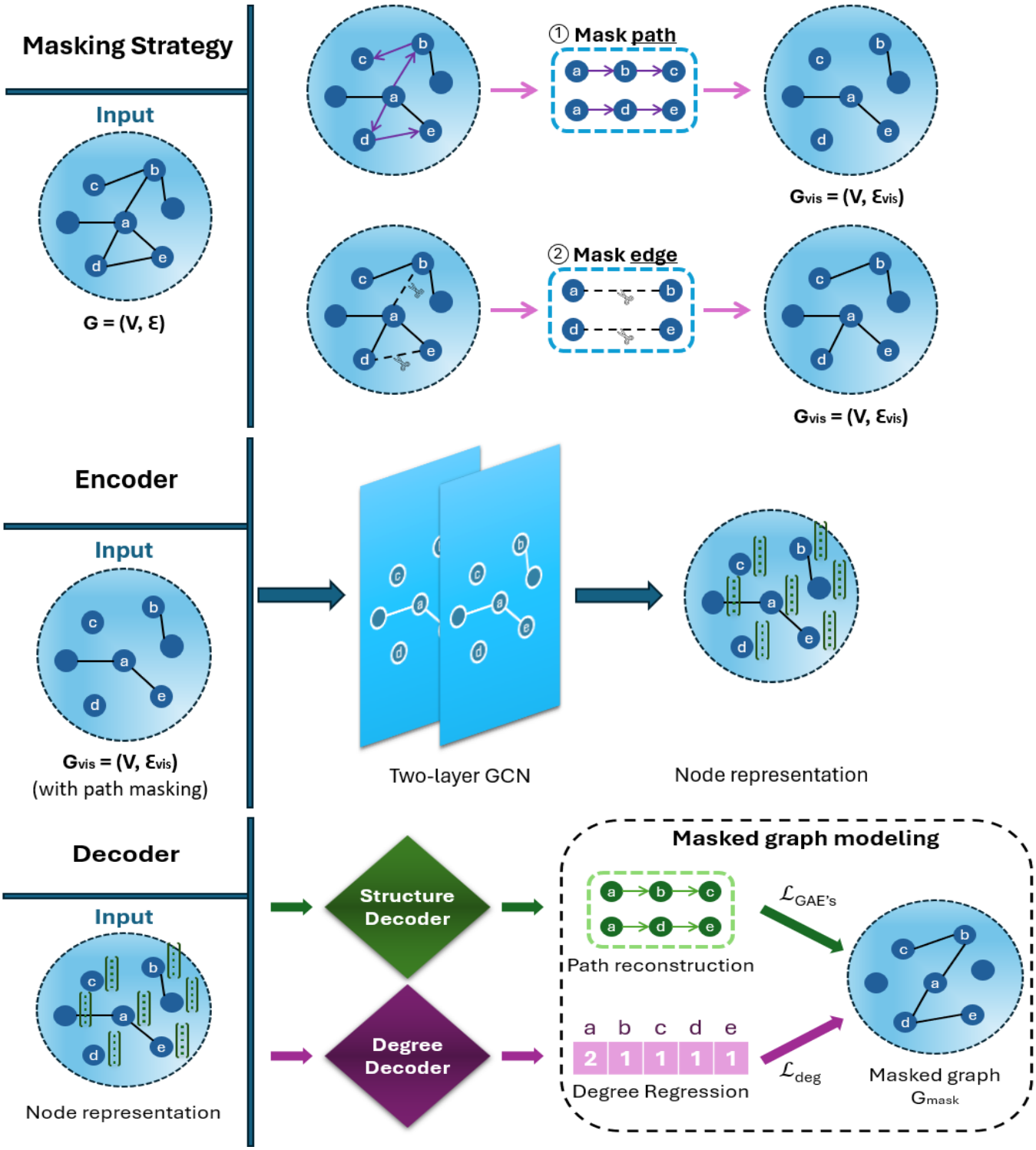
3.2.3 Learning Objective

The combined objective of MaskGAE is to find a balance between reconstructing the masked graph's edges (Reconstruction Loss) and accurately predicting node degrees (Regression Loss). The parameter α controls the trade-off between these two terms and influences the regularization aspect of the learning.

- **Binary Cross-Entropy Loss (BCE):** This loss function is used to measure the accuracy of the reconstructed edges compared to the original edges in the graph. The loss is calculated separately for positive edges (those present in the original graph) and negative edges (those not present in the original graph). The goal is to minimize the difference between the original and reconstructed graphs.
- **Redundancy Reduction:** As part of the learning process, the system also aims to reduce redundancy between paired subgraphs. This is achieved through the masking strategies, which help in minimizing the overlap between subgraphs and thus reducing redundant information.
- **Mutual Information Optimization:** The model is designed to optimize the mutual information between different views of the graph (represented by subgraphs) to ensure that the learned representations are both relevant and non-redundant for downstream tasks.

3.3 Flow

The flow begins with the input graph, where certain edges (connections between nodes) are masked based on a predefined strategy. This masked graph is input into the encoder, which processes only the visible parts of the graph and generates low-dimensional representations of the nodes. The decoder then takes these representations and tries to reconstruct the original graph, including the masked parts. The reconstructed graph is compared to the original to measure how well the model has learned to represent and infer the hidden parts. This process is repeated multiple times, with the model gradually improving its ability to reconstruct the graph accurately.


Masking Strategy
Input
G = (V, Ɛ)
① Mask path
Gvis = (V, Ɛvis)
② Mask edge
Gvis = (V, Ɛvis)
Encoder
Input
Gvis = (V, Ɛvis)
(with path masking)
Two-layer GCN
Node representation
Decoder
Input
Node representation
Structure Decoder
Degree Decoder
Masked graph modeling
Path reconstruction
$\mathcal{L}_{GAE's}$
a b c d e
2 1 1 1 1
Degree Regression
$\mathcal{L}_{deg}$
Masked graph
Gmask

4. Research Process

4.1 Exploring Theoretical Topics

Before exploring the MaskGAE model itself, we needed to comprehend several topics that serve as a platform for a deeper understanding of the model. Our research began with self-supervised learning, an approch relevant for understanding how models can learn from data without explicit labels [1]. We then made sure to understand the idea of contrastive learning: a technique that enhances unsupervised representation learning by maximizing mutual information between different augmented views of data [8]. This approach was particularly relevant given the fact that the model is working most exclusively with graphs [2].

Understanding Graph Autoencoders (GAEs) and the concept of masking in graphs was also essential, as this model's innovation is the combination between traditional GAEs and masking. GAEs operate within an encoder-decoder framework, learning to reconstruct missing parts of a graph by encoding the structure and decoding it back to its original form [5]. Masked autoencoding, widely applied in natural language processing and image modeling, provided further insights into how selectively hiding data elements and then predicting the missing parts could be leveraged for graph data, an area that has seen limited exploration [3].

Finally, we explored Graph Convolutional Networks (GCNs), which was the dominant approach for the encoder in the model's structure. We learned that GCN is a method to propagate information through graph structures in a way that is parallel to how convolutional networks process image data [5], setting the stage for understanding how these components integrate into the MaskGAE model. This foundational knowledge was necessary as it provided the appropriate context and theoretical knowledge for the mechanisms underlying the MaskGAE model.

4.2 Understanding MaskGAE Innovation

Throughout the exploration of the MaskGAE model, we realized that the innovation of the MaskGAE model lies in its ability to address the issue of redundancy within graph representations. This is a critical limitation found in traditional Graph Autoencoders (GAEs): While GAEs have been effective in learning node representations by maximizing mutual information among nodes, they often fall short by capturing excessive similarities between subgraphs. This redundancy can lead to the model overemphasizing overlapping information, which may not contribute significantly to accurate predictions or within the graph [4].

MaskGAE introduces a novel approach to overcome this by incorporating Masked Graph Modeling (MGM), where specific edges within the graph are strategically

removed. By reducing the overlap between subgraphs, MaskGAE minimizes the redundancy that obstructs traditional GAEs [6]. For example, by removing an edge between two connected nodes, the resulting subgraphs associated with each node become more distinct, thereby allowing the model to capture more diverse and meaningful representations. As a core conclusion, we realized that this ability to selectively mask parts of the graph and still accurately reconstruct the hidden elements is what sets MaskGAE apart, offering enhanced precision and utility in learning from complex graph networks.

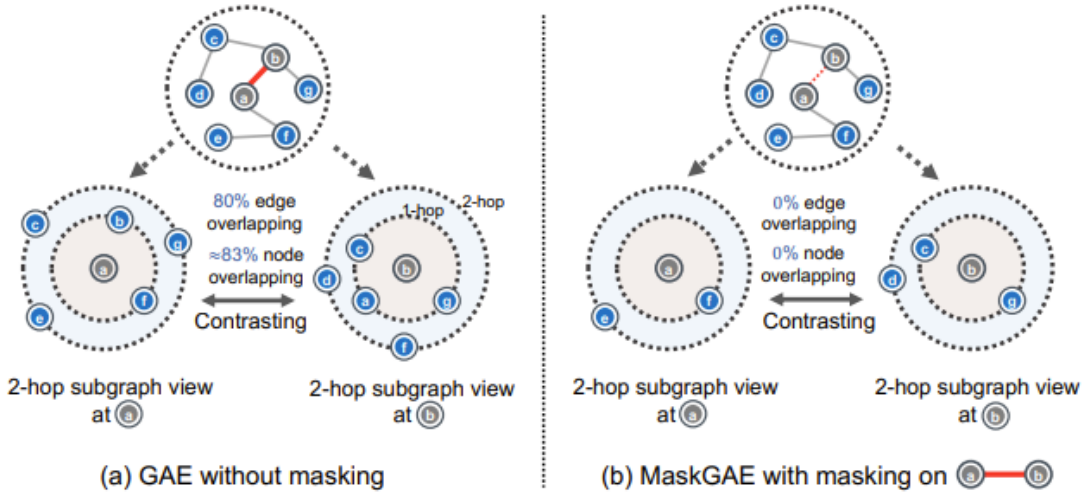


Figure 2. Benefits of MaskGAE [6]

4.3 Practical Research

In this section, we detail the practical research conducted to evaluate the MaskGAE model's performance under various conditions and configurations. For our own understanding, various parameters were modified throughout different iterations of the MaskGAE model. Among other parameters, we experimented with the degree of masking applied to subgraphs, the depth of the encoder-decoder architecture, and the masking ratio for the masking strategies. After several experiments on each set of defined parameters, we compared the metrics for evaluation of the model's abilities.

Chosen parameter: masking ratio (P) - explained in 3.2.1

	masking strategy	P	AUC	AP
run 1	Path	0.7 (default)	96.81% ± 0.12%	97.07% ± 0.11%
run 2	Edge	0.7 (default)	96.96% ± 0.11%	97.23% ± 0.08%
run 3	Path	0.5	96.21% ± 0.07%	96.39%± 0.09%
run 4	Edge	0.5	96.07% ± 0.08%	96.35% ± 0.11%

run 5	Path	0.9	95.95% ± 0.10%	96.00% ± 0.07%
run 6	Edge	0.9	95.80% ± 0.18%	95.96% ± 0.16%

The results show that at the default P = 0.7, both Path and Edge masking strategies deliver similar performance in terms of AUC and AP metrics. However, as the masking ratio deviates from the default (both higher and lower), the differences between the two strategies become even less prominent.

Chosen parameter: regularization factor (α) - explained in 3.2.3

	masking strategy	α	AUC	AP
run 1	Path	0.0001	96.03% ± 0.10%	96.17% ± 0.12%
run 2	Edge	0.0001	96.08% ± 0.10%	96.25% ± 0.12%
run 3	Path	0.003 (default)	96.00% ± 0.12%	96.16% ± 0.10%
run 4	Edge	0.003 (default)	96.07% ± 0.07%	96.27% ± 0.07%
run 5	Path	0.1	95.87% ± 0.08%	95.94% ± 0.11%
run 6	Edge	0.1	95.75% ± 0.16%	95.99% ± 0.15%

Regarding parameter α, the overall differences remain minimal and consistent across different values. The results show that both the Path and Edge masking strategies produce very similar performance across different runs, with only minor variations in AUC and AP scores.

Chosen parameter: regularization factor (decoder-dropout) - explained in 3.2.2

	masking strategy	decoder-dropout	AUC	AP
run 1	Path	0.2	92.84% ± 0.40%	94.22% ± 0.21%
run 2	Edge	0.2	93.25% ± 0.38%	94.49% ± 0.22%
run 3	Path	0.5	94.80% ± 0.23%	95.63% ± 0.17%
run 4	Edge	0.5	95.14% ± 0.27%	95.78% ± 0.21%
run 5	Path	0.8 (default)	96.00% ± 0.12%	96.16% ± 0.10%
run 6	Edge	0.8 (default)	96.07% ± 0.07%	96.27% ± 0.07%

The results indicate that as the dropout increases, both masking strategies show improvements in performance, with the Edge strategy maintaining a slight advantage in terms of both AUC and AP across all dropout settings.

In conclusion, although the general results showed only minor differences—typically half a percent or less in performance metrics—these variations allowed us to gain deeper insights into the relations between the model's components. This exploration proved valuable for our research, as it gave us a better understanding of how each parameter affects the model's behavior and overall performance.

4.4 Plan for Numerical Study

We chose the Cora dataset for our practical research on the MaskGAE model. The Cora dataset is a citation network dataset where nodes represent papers. Edges (links) represent citation relationships between papers (e.g., paper A cites paper B). Each edge in this dataset is a directed link between two nodes (papers). Since this dataset uses edge indices (two node identifiers for each edge), each link can be uniquely identified by the pair of nodes (source, target). Thus, we tracked and computed statistics for each unique link across multiple test sets, even though the test sets were randomized in each iteration. We monitored how many times a link appears in the test set and how often it was correctly predicted.

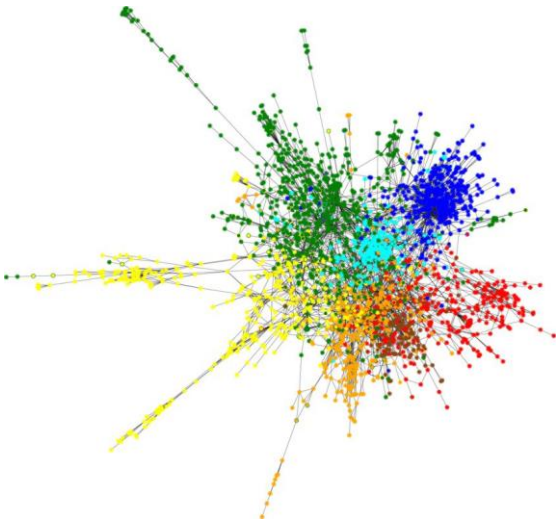


Figure 3. Partial visualization of the CORA dataset [10]

Each Iteration, we randomly divided the dataset into train, validation, and test sets. Also, we made sure that for each run, the test set will be different from the previous test sets. Each record of every link was maintained in the dataset, counting how many times the link appeared in the test set and how many times the link was predicted correctly. After multiple runs, we generated a histogram to visualize the distribution of correctly predicted links.

We used a Google Colab environment in Python so the chosen library for plot generation was matplotlib. The data (links statistics) was saved as CSV files, saved locally in the Colab environment, which can be managed using standard Colab

commands.

Afterwards, we created a dictionary data structure of “Pandas DataFrame” [9] that holds all the links in the dataset, initially setting counters for how often each link appears in the test set and how often it is predicted correctly. Every time we ran an iteration, we checked if a link was in the test set and updated our statistics accordingly. After enough iterations, we generated a histogram showing the distribution of correct predictions for each link.

5. Tools and Environments

During the development and implementation of the MaskGAE model, we utilized a variety of Python libraries and tools to facilitate the research process. The core environment for this research was based on Google Colab in Python, with significant use of libraries specifically designed for deep learning and graph processing.

First, **NumPy** was essential for performing numerical operations and handling array manipulation, which was necessary for managing the underlying data structures in the graph models. Another critical component was **PyTorch**, the deep learning framework that served as the backbone of the model development. PyTorch provided a flexible and robust environment for building, training, and optimizing the MaskGAE model.

Another crucial library is **torch-geometric** which was employed for handling graph-based data structures and implementing Graph Neural Networks (GNNs). This specialized library was crucial for performing operations like node and edge transformations, which are fundamental to graph-based deep learning models. Additionally, **SciPy** played an important role by offering a wide range of scientific computing tools. It was particularly useful for mathematical and statistical operations needed during data preprocessing and transformation.

Finally, to ensure efficient processing, we relied on **CUDA** and **CUDNN** to leverage GPU acceleration during model training. These libraries allowed the model to handle large-scale computations quickly and effectively, which was critical for maintaining performance, especially when working with large datasets. Using CUDA 10.2 and CUDNN 7.6.0 ensured compatibility with the available hardware and optimized performance for deep learning tasks. Several custom scripts were developed to manage different aspects of the research:

- **train_linkpred.py and train_linkpred_ogb.py:** These scripts were used for training the MaskGAE model, focusing on link prediction tasks. They allowed for the adjustment of key parameters such as batch size, learning rate, masking strategy, and dropout rates during training.
- **train_nodeclas.py:** This script was dedicated to node classification tasks, another key area of performance evaluation for the MaskGAE model. It also supported pretext training for link prediction.

- **model.py:** Contained the core MaskGAE model architecture, including the encoder and decoder components, leveraging GNN layers and implementing the masking strategies critical to the research.
- **mask.py:** Managed the masking strategies such as MaskEdge and MaskPath, which were key innovations in the MaskGAE model, helping to selectively mask subgraphs and edges.
- **loss.py:** Defined the various loss functions used for model training, including custom loss functions critical for evaluating the model's performance during graph reconstruction.
- **utils.py:** Provided utility functions for setting seeds, loading datasets, and formatting results, ensuring consistency and ease of use across different scripts.

6. Challenges

In our project to improve Graph Autoencoders (GAEs) with Masked Graph Modeling (MGM), we encountered several key challenges:

- **Masking Strategy Complexity:** Implementing an effective masking strategy was challenging, as we needed to balance the amount of information hidden without losing essential graph structures. Deciding the right proportions for edge-wise and path-wise masking required fine-tuning and experimentation to ensure the model learned useful features without oversimplifying the graph.

- **Model Scalability:** Scaling the model to handle larger graphs continues to be a significant challenge. The computational resources required for training on large graph datasets often result in extended processing times. While some optimizations have been made, managing scalability remains an ongoing difficulty that requires further exploration and improvement.

- **Maintaining Global and Local Structure:** One of the key goals was to capture both local (node-level) and global (whole graph) patterns. This balance proved difficult, as traditional methods tend to focus too much on local connections.

- **Tuning the MaskGAE Model:** Fine-tuning the Graph Convolutional Networks (GCN) and Multi-Layer Perceptrons (MLP) for the encoder and decoder required multiple iterations to avoid overfitting and ensure the model was learning meaningful patterns. This process involved frequent adjustments to hyperparameters and required significant computational resources.

- **Evaluation Metrics:** Identifying the most relevant metrics to measure the model’s performance is a challenging task. Due to the complexity of graph structures, it is difficult to find the right balance between reconstruction accuracy and preserving meaningful graph features.

7. Results

7.1 Goals

The primary goal of our research was to gain a deep understanding of the MaskGAE model, its architecture, and how its various parameters interact to influence overall performance. We aimed to thoroughly investigate the role of the model's components, such as the encoder, decoder, and masking strategies, in improving the accuracy of graph-based predictions. By testing different elements, we sought to optimize the model and better understand the underlying mechanisms that drive its performance on complex network data.

The second goal of our research was to develop a link tracking mechanism for prediction accuracy of individual links within the graph. We created and maintained detailed statistics for each link: the number of times it appeared in test sets and the frequency with which it was correctly predicted. These details aimed to evaluate the consistency and reliability of the MaskGAE model across multiple runs.

This approach allowed us to better understand how certain edges in the graph are more challenging to predict, and how the model's ability to generalize, varied depending on test set size, randomness, and model’s components. Our goal was to generate insights that could inform future improvements to both the model and the data preparation process, ultimately leading to more accurate and reliable graph-based predictions.

7.2 Train 1

The first training process we decided to view link prediction performance through two experiments, focusing on the two masking strategies. Both experiments involved the Cora dataset (test set size of 30%) with 40 runs (500 epochs per run) and all other parameters set to default. The first experiment employed the Path Masking Strategy, while the second iteration employed the Edge Masking Strategy. The results of these iterations are visualized in the histograms below, which show the distribution of correctly predicted links across all 40 runs.

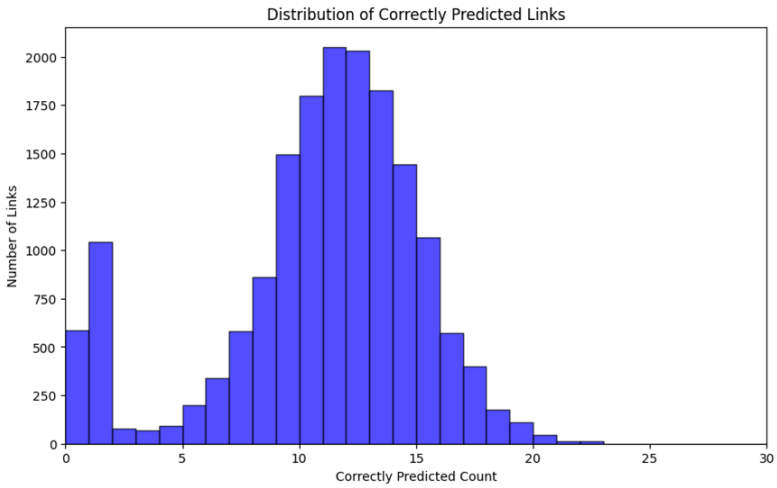


Figure 4. Histogram of Path Masking Strategy (30% test set)

The histogram generated for this experiment (fig. 4) shows a negatively skewed distribution, with most links being predicted correctly around 10 to 15 times. The peak of the distribution occurs at around 10 to 12 correct predictions, indicating that a substantial portion of the links is moderately well-predicted by the model. However, the tail of the distribution on the left side reveals a significant number of links that were predicted correctly between 0-1 times. This suggests that a certain subset of links consistently presents difficulties for the model.

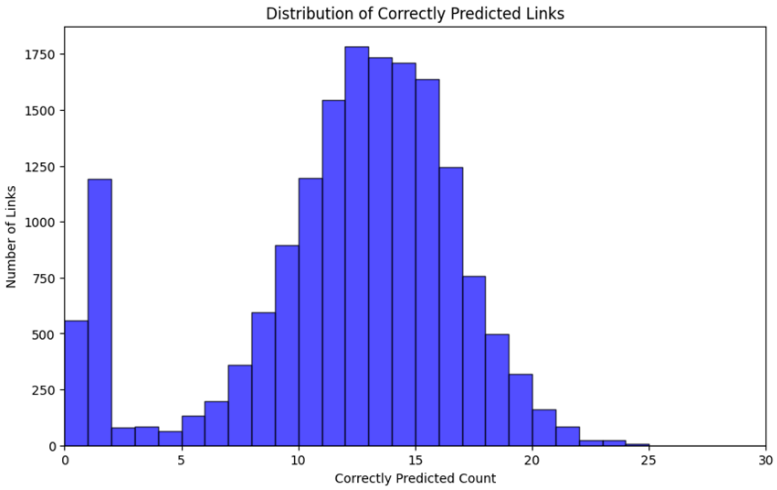


Figure 5. Histogram of Edge Masking Strategy (30% test set)

In this experiment (fig. 5), we shifted to the Edge Masking Strategy, where individual edges (rather than paths) are masked. The histogram shows a similar negatively skewed distribution, but with some important distinctions. The peak of the distribution in this case is around 12 correct predictions, with the majority of links being predicted correctly between 12 and 16 times. The left tail of the distribution still shows a subset of links that were difficult for the model to predict correctly between 0-1 times.

7.3 Train 2

For the next training we decided to view link prediction performance when the test set is taken at 50% from the Cora dataset. Like Train 1, we conducted two experiments: one for each of the two masking strategies (Path Masking and Edge Masking). Both experiments are set to 40 runs (500 epochs per run) and all other parameters are set to default. The results of these iterations are visualized in the histograms below.

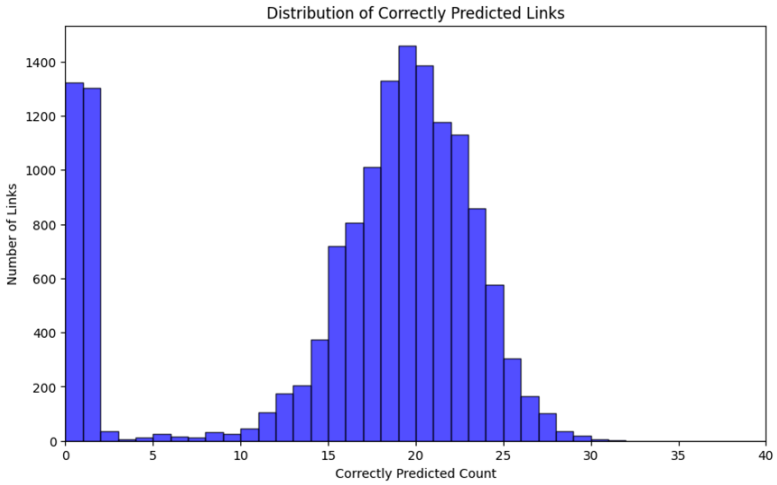


Figure 6. Histogram of Path Masking Strategy (50% test set)

In the first histogram (fig. 6), we observe a large concentration of links correctly predicted between 18 and 22 times (with around 1000-1400 links), and a significant number of links that were not predicted correctly even once (approximately 1300). This suggests that while a portion of the links is consistently well-predicted, there is a substantial subset of links that the model struggles to predict accurately.

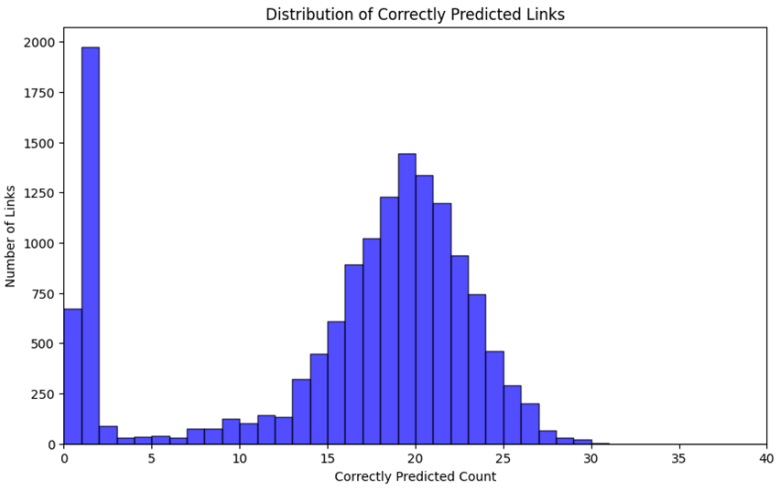


Figure 7. Histogram of Edge Masking Strategy (50% test set)

In the second histogram (fig. 7), we see a similar distribution, but with some differences. The overall peak for correctly predicted links is slightly shifted, showing more consistency in correct predictions around the 19-22 times (1250-1500 links). Like the Path Masking Strategy, there is a sizable group of links that are never predicted

correctly (around 600) and a large group of links that were predicted once (around 2000). The peak of correctly predicted links in the Edge Masking is more prominent.

Overall, both strategies show similar behavior, with many links being difficult for the model to predict. However, the Edge Masking Strategy demonstrates slightly better performance, with a stronger concentration of links in the higher range of correct predictions, indicating that it might allow for more consistent performance across different types of links.

7.4 Train 3

Another training process of ours is focusing on the masking ratio (P), explained in 3.2.1. In these iterations, the masking ratio was varied while keeping other parameters constant, including a 30% test set from the Cora dataset, the Path Masking Strategy, and 40 runs (500 epochs per run). The goal was to evaluate how different levels of masking, p = 0.3, p = 0.7 (default), and p = 0.9, impact the model's link prediction performance. The results are visualized in the histograms, which display the distribution of correctly predicted links across all runs for each masking ratio.

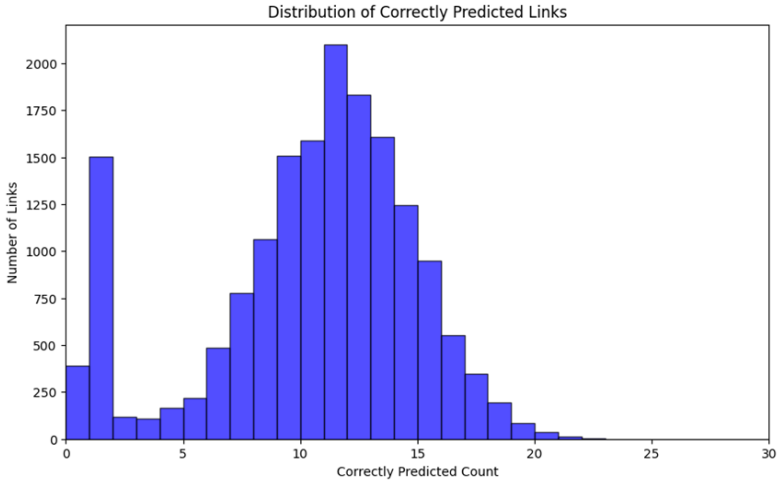


Figure 8. Histogram of masking ratio P = 0.3

The histogram for p = 0.3 (fig. 8) demonstrates a right-skewed distribution with a peak at around 12 correct predictions. Most of the links are predicted correctly between 8 and 16 times, and the number of links predicted correctly fewer than 5 times is relatively small. This suggests that at a lower masking ratio (p = 0.3), the model has enough information to make moderately accurate predictions for the majority of links. However, there is still a notable subset of links that remain challenging, as indicated by the small left tail of links with 0-1 correct predictions.

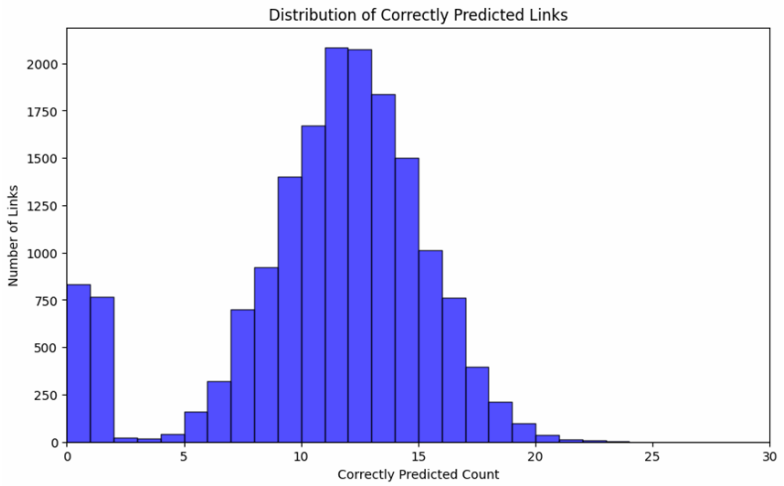


Figure 9. Histogram of masking ratio P = 0.9

The histogram for p = 0.9 (fig. 9) shows a distribution that is very similar in shape to the previous two iterations but with a slightly less pronounced left tail. The peak of the distribution is still at around 11 to 14 correct predictions, but the number of links predicted correctly between 0-1 times has diminished. This suggests that at a higher masking ratio, the model can predict more certain links. Also, the right side of the distribution remains strong, with a notable portion of links still being predicted correctly more than 10 times.

For p = 0.7 (fig. 4) we can see that the default value is sufficient in predicting a robust number of links from the test set. Especially, when compared to other values of the masking ratio.

8. General Conclusions

Throughout the various iterations and experiments, we have observed several key patterns related to the performance of the MaskGAE model in the link prediction task.

Impact of Masking Strategies:

Both the Path Masking Strategy and Edge Masking Strategy showed strong, comparable results across all experiments. However, a slight performance improvement was consistently noted with the Edge Masking Strategy. This improvement is reflected in the shift of the distribution of correctly predicted links towards the right, indicating that more links were consistently predicted correctly with the Edge Masking Strategy. This strategy, by masking individual edges rather than entire paths, seems to allow the model to better generalize across different graph regions, resulting in more reliable predictions for a wider range of links. Despite this, the differences between the two strategies remain small, with both exhibiting a high level of predictive capability, as evidenced by similar AUC and AP scores.

Test Set Size:

The experiments with different test set sizes (30% and 50%) revealed that the test set size has a clear influence on the distribution of correctly predicted links. The larger test set (50%) resulted in a more balanced distribution (more links falling into the middle ranges of correct predictions), but with more links predicted 0 or 1 times. This suggests that with a larger test set, the model benefits from increased exposure to a more diverse subset of links during evaluation, improving its generalization across runs. On the other hand, smaller test sets (30%) showed a lower concentration of poorly predicted links (0-1 times), likely due to the reduced variety of links in the test set.

Masking Ratio (P):

The experiments with varying masking ratios (p = 0.3, 0.7, 0.9) demonstrate that the P parameter significantly impacts the model's ability to predict links accurately. At p = 0.3, the model performs well but tends to struggle with a subset of links, as shown by the spike in links with 0-1 correct predictions. This suggests that certain links remain challenging to predict regardless of how much information is masked. As the masking ratio increases to p = 0.7 and p = 0.9, the model shows a more balanced distribution of correct predictions, suggesting that this ratio strikes an effective balance between providing enough information to the model while still challenging it.

Analysis of Poorly Predicted Links:

Across all experiments, a consistent observation was the spike in links that were predicted correctly 0-1 times. These poorly predicted links are likely associated with sparsely connected nodes or those located in more structurally complex regions of the graph, where the model struggles to identify strong predictive patterns. In addition, insufficient local information or the altering structural properties of nodes could also be a factor that contributed to the creation of these unpredicted links.

For example, this factor is known in literature as studies have found that any alteration in a citation network data could have a major influence on the ability of recognizing different links in the network. In the research about the GMAE model, it was concluded that around 30-50% of the total edges (citations) that went through data transformation were later deviate from the inner structure of the citation network and thus went unnoticed [12].

9. Further Investigation

As we discussed in our results, each plot contains skewness in the prediction accuracy distribution. In this regard, an analysis is required for exploring this outcome. First, an analysis of the hard-to-predict links, particularly those with low prediction accuracy (e.g., 0-1 correct predictions), could reveal important characteristics that challenge the MaskGAE model. It is important to investigate whether these links occur between specific types of papers or node classes, or whether they are more isolated or weakly connected within the graph.

Furthermore, examining the node and edge attributes associated with these difficult-to-predict links may uncover whether certain nodes possess unique or rare features that make their link prediction more challenging. Understanding whether the feature representation is insufficient to capture the underlying structure could inform future improvements in data preprocessing and feature engineering.

Additionally, it is essential to study the overall graph structure, focusing on the distinction between well-predicted and poorly predicted links. Links that are consistently well-predicted may belong to dense clusters of nodes (such as groups of papers within the same research domain) while hard-to-predict links may appear from more sparsely connected regions of the graph. This possible observation can provide insights into the MaskGAE model's ability to generalize across different graph regions.

After investigating these aspects, we believe that there is a primary potential for extending the MaskGAE model to new domains. More precisely, apply it to social networks that represent connections between individuals. Although we did not have the opportunity to explore this direction in our current research, we see significant promise in adapting the model to analyze social graphs. Social networks present unique challenges, with dynamic interactions and evolving relationships, and we believe that the MaskGAE framework could be a powerful tool for tasks like community detection, influence modeling, and friend recommendation. This remains a future research avenue, as further modifications and experiments would be needed to tailor the MaskGAE model to effectively handle the specific nuances of social graph structures.

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

[6] 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.

[7] https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html

[8] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449802

[9] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[10] https://paperswithcode.com/dataset/cora

[11] R. Avros, M.B. Haim, A. Madar, E. Ravve, Z. Volkovich, Spotting Suspicious Academic Citations Using Self-Learning Graph Transformers. Mathematics 2024, 12, 814. https://doi.org/10.3390/math12060814 (Q1)