UGGS: A Unified Graph Generation Framework Based on Self-Supervised Learning

Sajad Ramezani
srameza1@ualberta.ca
University of Alberta

Soroor Motie
smoti088@uottawa.ca
University of Ottawa

ABSTRACT

Deep learning on graphs has gained interest in recent years. The applicability of graphs to model problems in various domains, such as chemical molecules, financial transactions, parse trees, etc., has encouraged researchers to develop and extend machine learning methods from other data modalities, such as text and image, to graphs. Generative models have been used extensively in recent years and have achieved significant milestones, especially in text and image generation. However, graph generative models have not been developed as extensively, and fundamental problems are still in the discussion phase. This work addresses some of these problems, such as the lack of an integrated framework and interpretable evaluation metric, by introducing a unified framework for the graph generation task. The base of the proposed framework is on the appropriate graph and node embeddings to estimate graphs' distribution. Hence it composes of graph neural networks to embed the nodes and graphs and also enhances the quality of graph embeddings via the introduction of pseudo tasks in a self-supervised fashion. Self-supervised techniques have proven useful in enhancing generative models to be more robust and generalizable. This work proposes several pseudo tasks and evaluates their performance on common graph datasets. It also emphasizes the problem of graph decoding and speculates that graph generation strategy matters, and one can establish more complex graph generation models to generate higher-quality graphs. It also proposes a distance metric in embedding space for generated graphs to filter out poorly generated data. In the end, the proposed framework achieves competitive results compared to previously proposed models while having fewer parameters and a shorter training time. We have also made our framework implementation available.

KEYWORDS

graph generation, graph neural networks, generative models, self-supervised learning

ACM Reference Format:
Sajad Ramezani and Soroor Motie. 2023. UGGS: A Unified Graph Generation Framework Based on Self-Supervised Learning . In *Proceedings of ACM Conference (MLG'23).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

MLG'23, August 2023, Long Beach, CA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph generation, the task of generating new graphs that embody similar characteristics to an existing set, has proven to be pivotal in a wide spectrum of applications. These range from the intricate realm of molecular design in pharmaceutical research [29] [7] to the data-intensive field of financial transactions. The promise held by graph generation models lies in their capacity to generate plausible new instances. By doing so, these models present a potent tool for simulation, augmentation, and synthesis [13]. They extend our ability to fabricate diverse datasets, project future instances, and even unearth patterns veiled within the complex structure of graph-based data. Notwithstanding its inherent complexities and challenges, graph generation underscores a vital research pursuit, driving advancements and enabling novel possibilities in various domains.

However, graph generation poses unique challenges compared to other generative tasks such as text or image generation. Unlike sequences or 2D structures, graphs are composed of discrete elements arranged in a complex, non-sequential format, often capturing intricate relationships and topologies. This complexity calls for specialized methods to model and generate novel graph structures effectively. Additionally, evaluating the quality of generated graphs is another significant challenge due to their high-dimensional, discrete nature, and the inability for visual inspections as a measure of quality.

The history of graph generation models stretches back to classical models such as Erdös-Rényi, Watts-Strogatz, and Barabasi-Albert. These models served as cornerstones in understanding the foundational properties of graph generation, based on principles of randomization, small-world phenomena, and preferential attachment respectively[20]. However, their utility has been limited due to assumptions of homogeneity and independence among nodes, failing to capture the nuanced heterogeneity and complex dependencies within real-world graph structures[30].

As a response to these shortcomings, the advent of deep learning has ushered in a new wave of methods for graph generation. Deep generative models for graphs promise to address the limitations of classical methods, introducing a level of complexity and adaptability that can better mirror the intricate and heterogeneous nature of real-world graphs.

As we venture into the complexities of various graph generation models, the need for a unified framework becomes increasingly apparent. Many of these models share common underlying mechanisms – they learn graph representations and subsequently sample or leverage the learnt embedding space to generate new graphs [31]. However, the diversity in their formulation and structure makes it challenging to integrate their unique strengths effectively [9]. The proposed framework in this work aims to resolve this issue by

combining the best aspects of these diverse methods, paving the way for a more robust graph generation model.

Building upon the conventional approach, we have introduced self-supervised learning to our framework, focusing on the formulation of innovative pseudo-tasks. This addition not only enhances the capability of our model but also allows it to encompass the essence of many preceding works. We have also highlighted the importance of the decoding step, a factor often overlooked in many models. By proposing two decoding strategies, we demonstrate the effectiveness of a more refined decoding strategy, underscoring its critical role in the generation process.

Additionally, we have addressed one of the more significant challenges in the realm of graph generation - the lack of interpretable measures for evaluating the generated graphs. To counter this, we introduced an interpretation of Maximum Mean Discrepancy (MMD) based on graph perturbation, providing a more tangible measure of performance.

In a bid to promote further research in this area, we have developed and open-sourced our framework. This enables other researchers to apply our approach easily and develop new graph generative models, fostering faster and more effective advancements in the field.

2 RELATED WORK

The goal of the graph generation task is to train a model on a graph dataset derived from a specific distribution, denoted as p, such that the model effectively captures this underlying distribution. The model then produces a new distribution, denoted as q, which ideally mirrors the original distribution p, allowing us to generate graph samples from q that echo the properties of p. Some research also introduces additional constraints to support optimization and conditional generation, thereby enhancing the overall quality of the generated graphs[9],[31].

The field has seen an evolution from classical to more advanced, deep generative models. Classical models, including Erdős-Rényi [8], Watts-Strogatz, and Barabási-Albert [3], laid down the initial groundwork but were relatively simplistic and lacked the versatility to capture complex graph structures.

Recently, deep generative models have come to the fore, showcasing improved results by more accurately capturing intricate graph structures and properties. Various approaches have been used, each with their distinct perspective and strategy. For instance, GraphRNN[30] treats graph generation as a sequential process, generating nodes and edges in a step-by-step manner akin to natural language processing. Conversely, models like spanning tree-based graph generation [1] perceive the task as a series of decisions that cumulatively construct the graph.

However, while these methods present useful modelling techniques, they operate under the assumption of a canonical ordering for the nodes and edges. This is problematic because it conflicts with the inherent permutation invariance of graphs, an attribute that these models struggle to accommodate.

2.1 Graph Generation and Generative Models

Deep graph generative models have seen a rich development and can be broadly categorized into five types: Autoregressive Models, Variational Autoencoders (VAEs), Normalizing Flows, Generative Adversarial Networks (GANs), and Diffusion Models. Here is a brief description of each:

- Autoregressive Models: Autoregressive models, like GraphRNN [30], generate graphs in a sequential manner. The generation process assumes a predefined ordering of the nodes and edges, and each element is generated conditioning on the previously generated ones. Although this method is highly effective, the assumption of canonical ordering is a limitation given that graphs are inherently permutation-invariant.
- Variational Autoencoders (VAEs): Graph VAEs [16] encode the input graph into a continuous latent space, and then decode the latent representation to generate the output graph. The encoding-decoding mechanism allows the model to learn complex patterns and distributions from the input data. However, capturing the variability in graph size and structure can be a challenge for these models.
- Normalizing Flows: These models map a simple probability distribution to a complex one through a series of invertible transformations, enabling exact computation of log-likelihood and exact sampling. In the context of graph generation, normalizing flows provide an appealing framework to model the complex distribution of graphs, with models like GraphAF [22] and GraphNVP [18] being notable examples.
- Generative Adversarial Networks (GANs) [11]: GANs for graph generation, such as GraphGAN [26] and MolGAN [5], use a game-theoretic setup where a generator and discriminator are trained simultaneously. The generator creates graphs aiming to fool the discriminator, while the discriminator learns to distinguish real graphs from generated ones. This adversarial process helps in generating high-quality graphs, but training GANs can often be unstable.
- Diffusion Models: These are a newer category of models where a data distribution is modelled as a reverse diffusion process starting from a simple prior. These models, although not yet as common in the context of graph generation, show great promise due to their ability to generate high-quality samples and compute likelihoods.Vignac et al. 25 discrete diffusion process to generate realistic graphs. It has further constrained by being permutation equivarience.

The above categorization is based on Zhu et al. 31 algorithmic taxonomy for graph generation, which bears similarity to our framework in terms of offering a unified approach to graph generation. However, our framework distinguishes itself on two major fronts. Firstly, we approach the problem from the perspective of self-supervised learning. We incorporate various pseudo tasks, such as link prediction, which allow our model to learn and fine-tune its parameters, facilitating robust and flexible representations of the graph structures. Secondly, we provide an open-source implementation of our framework, which is aimed at fostering further research in the field of graph generation. This implementation not only encapsulates previous methods, thus serving as a comprehensive tool for graph generation tasks, but also provides researchers with a platform to enhance and build upon existing techniques, thereby accelerating the development of novel and more efficient graph generative models.

2.2 Contribution

A category of proposed graph generation models is based on Encoding graphs into embedding space. While using various approaches, these approaches can be formed into a unified framework to be used together and compared effectively as described by [31]. There has not been a dedicated framework to enhance and facilitate the task of graph generation. This work proposes an implemented framework with decoupled modules such as Encoder, Self-Supervised Tasks, Decoding, and Evaluation. The modules will allow for an easier extension of these works and better benchmarking. However, latter case is not the primary concern of this work. Since evaluating generative models is a challenging task in the case of graph generation, it is substantially harder because of both the discreteness and incapability of visual inspection. We show that the proposed model accomplishes comparable results and sometimes beats state-of-the-art in the general graph generation tasks with fewer parameters. We have also emphasized the importance of decoding strategy in graph generation and introduced a similarity filter based on graph embedding to omit poor-quality generated graphs. We further justify this approach with a perturbation graph experiment . Finally, we introduce the developed tool based on the proposed framework, which includes all the modules and models discussed in this work while being compatible with several other "ML for graphs" tools developed in recent years. This tool also includes the necessary components for developing the graph generation model, including datasets, evaluation methods, and interpretation tools. We hope this tool can enhance and facilitate the work of other researchers in this field. In the end, we discuss the implication and further research direction and propose several extensions of this work.

To sum up, we try to overcome the shortcomings of previous works, particularly through the following contributions:

- Proposing a unified framework dedicated to graph generation task achieving state-of-the-art graph generation results compared to previous models w.r.t the number of parameters
- Proposing and applying a decoding strategy to refine generated graphs using graph embedding distance
- Ensuring reproducibility by offering an extensive tool for implementing the presented framework in a more comprehensive way
- Clarifying the evaluation metric of maximum mean discrepancy by offering an interpretable evaluation measure

3 APPROACH

3.1 UGGS: Framework Overview

In this work, we propose a novel framework for graph generation, broken down into two essential components. First, we utilize a guiding model, which is capable of comprehending the properties and distribution of the training graph data for graph generation purposes. An example of such a guiding model can be a link prediction model. Second, we define a graph generation strategy, employing the guiding model to generate graphs with desired properties.

Our proposed framework for graph generation employs graph neural networks to capture node and graph-level representation. These representations are subsequently refined through self-supervised tasks on graphs. The overall architecture of this framework is depicted in figure 1. Initially, the encoder captures the crucial features of the training graph distribution within an embedding space. The resultant node and graph-level representations are then fed into several self-supervised heads for further tuning of representation and model parameters. As demonstrated by previous studies, self-supervised methods have proven highly effective in encapsulating useful features within the embedding space, leading to a more robust representation.

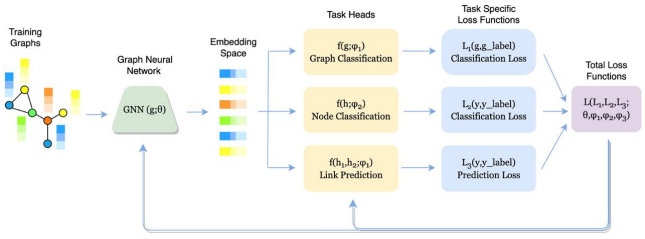

Figure 1: UGGS Framework with three pseudo task

3.2 Training

3.2.1 Encoder. First, we employ an encoder to capture the distribution of the training data. The encoder's role is to learn a mapping from nodes to a latent space, and it can also incorporate a readout function to embed the entire graph into an embedding space. For this purpose, we utilize a graph neural network. Our framework is versatile and can work with any other model that essentially learns this aforementioned mapping to capture both the node embeddings and graph embeddings.

3.2.2 Pseudo-heads. In order to train the parameters, we introduce pseudo-tasks. These tasks allow for the simultaneous training of both the encoder model and the task heads. The task heads are responsible for fine-tuning the parameters and learning a mapping from the embedding space produced by the encoder to specific tasks, such as link prediction, which proves beneficial for graph generation.

3.2.3 Training Procedure. In the training phase, the GNN encoder derives node representations of the training graphs through message passing and aggregation among nodes. A variety of GNN architectures proposed in the literature may be employed. In our study, we examined three of the most prevalent approaches in graph generation: GraphSage [12], GAT [24], and GIN [28]. To further enhance the parameters, we suggest the incorporation of self-supervised heads. Each head is a parameterized model, such as a neural network, tasked with pretext (pseudo) tasks. The underlying intuition is that these tasks will guide the parameters of both the GNN encoder (θ) and the self-supervised heads (ϕ) towards greater robustness and superior generalization.

Each self-supervised head may necessitate its own preprocessing and unique loss function. For example, the link prediction head selects pairs of positive and negative nodes. Positive node pairs are adjacent in the training graph, whereas negative pairs are not adjacent in the original training graph. These heads may also introduce hyperparameters to the overall framework. For instance, in

the case of link prediction, the ratio of negative to positive pairs to be sampled must be defined. These self-supervised heads utilize the node embeddings produced by the GNN encoder as input, as depicted in Figure 1. Graph-level tasks can also be introduced with the aid of the read-out function (GIN). The loss functions of self-supervised heads need to be combined, which can be achieved through a simple linear combination.

The framework comprises two sets of parameters: 1) θ, which represents GNN encoder parameters, and 2) ϕ, denoting self-supervised heads parameters. This decoupling permits the use of pre-trained GNN encoders such as GRAN, addressing a limitation present in previous graph generation architectures. The GNN encoder and self-supervised heads can be trained either jointly (as demonstrated in our experiments) or separately.

For instance, in our algorithm, we employed link prediction as a pseudo-task. The loss function for this task is defined through the sampling of positive edges (edges that exist in the graph) and negative edges (edges that do not exist in the graph). Another component of the loss function serves as a regularizer for the learned embeddings. In this work, we adopt the same loss function as [12]Moreover, in the results section, we explore models that incorporate degree prediction as an additional task (represented by UGGS-2), showcasing the flexibility and adaptability of our unified graph generation framework.

3.3 Inference Phase

The problem of generating a graph from a learned model, a process we refer to as 'decoding' in the context of generation, this problem has seen relatively limited exploration in the literature despite its importance. In contrast, the significant volume of research and methodologies dedicated to the problem of decoding in the realm of natural language generation has motivated us to establish a similar structure within the context of graph generation. This leads us to pose the question: how critical is a decoding strategy to the performance of a graph generation model?

One intuitive decoding strategy, as depicted in the figure 2 involves the generation of a graph with n vertices: initially, we sample n-node embeddings from the embedding space. Utilizing the Link Prediction head, we estimate the probability of adjacency between each pair of nodes. Subsequently, through thresholding, we generate the adjacency matrix for the graph. In our experiments, we employed a uniform sampling approach from the node embedding space for generating graphs. However, our framework allows for the possibility of incorporating a learned sampling module in future work, providing flexibility in the sampling strategy. as

The second decoding strategy employs a similarity measure with a reference set, a group of graphs to which we desire our generated graphs to bear similarity. Initially, we embed the generated graphs into an alternative embedding space, employing recently developed graph embedding approaches. Then, using cosine similarity in the embedding space, we search for generated graphs that show the greatest similarity to the reference set (figure). As demonstrated in the [Results Section], the choice of decoding strategy is consequential, and this research area has been somewhat under-explored in the literature. There is scope for a multitude of more sophisticated approaches to enhance the quality of generated graphs.

Algorithm 1: Training Algorithm for the proposed framework with GraphSage as the encoder and link prediction as the only pseudo task

Input: Graph dataset $D = \{g | g \text{ is } G(V, E, X)\}$,
V: vertex set, E: edge set, X:
node features,
k: graph neural network depth,
σ: non linear function,
$AGGREGATE$: differentiable aggregator function,
N: neighbourhood function,
d: embedding dimension,
$f(u, v)$: link prediction neural network u, v: edges
Output: Vector representation z_v for all vertices and edge probability $p_{u,v}$

for G *in* D **do**
  `// Embedding generation`
  $h_u^0 = G.X_u$
  for k **do**
    for v *in* $G.V$ **do**
      $h_{N(v)}^k = AGGREGATE(\{h_{u^{k-1}}, \forall u \text{ in } N(v)\})$
      $h_v^k = \sigma(W_k * \text{concat}(h_v^{k-1}, h_{N(v)}^k))$
    $h_v^k = h_v^k / \text{norm}(h_v^k), \forall v$
  $z_v = h_v^k \forall v$
  `// Link prediction`
  P = empty matrix
  G = Ground Truth matrix
  for u, v *in SamplePositiveEdge(g)* **do**
    $P[u, v] = f(z_u, z_v)$
    $G[u, v] = 1$
  for u, v *in SampleNegativeEdge(g)* **do**
    $P[u, v] = f(z_u, z_v)$
    $G[u, v] = 0$
  loss = $BCE_{emb}(Z) + BCE(P, G)$

4 EXPERIMENTS

We evaluated our proposed model on a total of four distinct datasets—two of which are synthetic and the other two originating from protein and 3D object domains respectively. This diverse selection was intentional, serving to demonstrate the broad applicability and versatility of our model across varying use-cases. The specifics of these datasets are elaborated upon in Table 1.

Our model's performance was benchmarked against a selection of alternative models from existing literature, as outlined in Table 3. The evaluation metrics utilized to assess the quality of the generated graphs include degree distribution, clustering coefficient, orbits, and spectral characteristics. These metrics were compared using the Maximum Mean Discrepancy (MMD) method.MMD quantifies the dissimilarity between two probability distributions. For our purpose, we use the squared MMD to compare sets of samples from distributions p and q. It can be derived as follows:

$$\text{MMD}^2(p||q) = \mathbb{E}_{x,y\sim p}[k(x,y)] + \mathbb{E}_{x,y\sim q}[k(x,y)] - 2\mathbb{E}_{x\sim p, y\sim q}[k(x,y)]$$

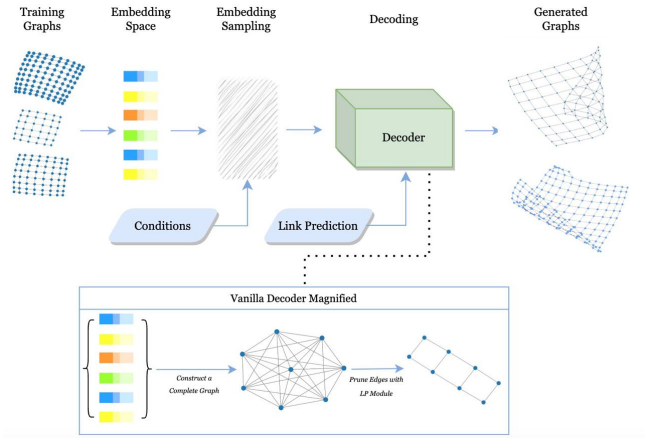

Figure 2: Proposed Vanilla Graph Generation Strategy

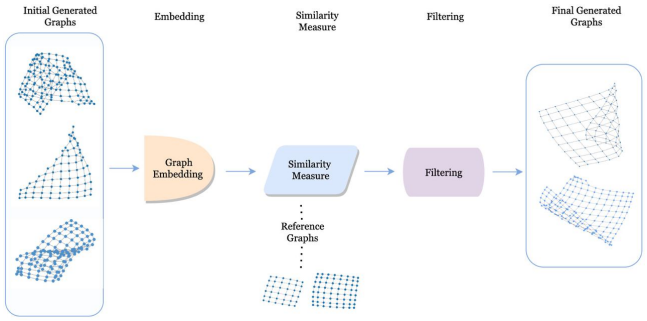

Figure 3: Proposed Filtering Graph Generation Strategy

Table 1: Dataset statistics

Dataset Name	Size	Vertecies	Type	Description
2D Grid	100	100<V<400	Synthetic	2d shaped grid
Lobster	100	10<V<100	Synthetic	tree with nodes of at most 2 nodes from the main branch
Protein	918	100<V<500	Real-world	[6]
3D Mesh	41	V<5000	Real-world	[19]

The choice of using Maximum Mean Discrepancy (MMD) as an evaluation measure is motivated by its widespread adoption in previous works and the inherent challenge of assessing the quality of generative graphs. The MMD measure, coupled with a set of graph statistics, provides an efficient approach to evaluate the quality of generated graphs.

5 RESULTS AND DISCUSSION

Across most datasets, our model achieved state-of-the-art results, while maintaining a significantly smaller number of parameters and reduced training time. For instance, in the case of the protein dataset, our model used only 259k parameters compared to 1545k in GRAN and 392k in GraphRNN, while still generating more representative protein graphs.

Moreover, the results from the 2D grid data clearly demonstrate the effectiveness of incorporating self-supervised tasks. When we compare our original model (UGGS-1) with the same model enhanced with a pseudo task of degree prediction (UGGS-2), we find a noticeable improvement. This indicates that incorporating relevant pseudo tasks can effectively boost model performance.

However, it's important to be mindful while designing these pseudo tasks, as they may not always contribute to model improvement. This is exemplified in the case of the protein dataset, where the addition of a pseudo task did not result in enhanced performance.

Lastly, the 3D mesh dataset showcases the scalability of our framework. GraphRNN-based models failed to process this dataset and returned an out-of-memory error, whereas our model successfully managed the task and produced competitive results. These findings underline the efficiency and robustness of our proposed framework for graph generation, even in demanding and large-scale scenarios.

We compared our model to the following models from the literature Classical Models:

- Erdös-Rényi [8]: This is a simple random graph model, where each edge is included in the graph independently with a given probability. It's useful for representing unstructured randomness but may not capture complex patterns or properties seen in real-world graphs.
- Barabasi-Albert [3]: This model generates random scale-free networks using a preferential attachment mechanism, where new nodes prefer to attach to existing nodes with high degree. It's often used to mimic the growth of real-world networks.
- MMSB (Stochastic Block Model) [2]: This probabilistic model generates graphs by partitioning nodes into latent blocks or communities, and then connecting nodes based on the block membership. It's often used for community detection and network analysis.
- GraphVAE (Graph Variational Autoencoder) [16]: GraphVAE learns a probabilistic mapping of graphs into a continuous latent space and can generate new graphs by decoding random points in this space. It's capable of capturing the underlying distribution of the graph data.
- GraphRNN [30]: GraphRNN formulates the graph generation process as a sequence generation problem, and uses recurrent neural networks to generate nodes and edges sequentially. It can handle graphs of arbitrary size and complexity.
- GRAN (Graph Recurrent Attention Network) [17]: GRAN applies an autoregressive approach to graph generation, using blocks of adjacency matrix and calculating probabilities with attention weights. It can handle complex patterns in graph data.
- BiGG (Scalable Deep Generative Modeling for Sparse Graphs) [4]: BiGG is an autoregressive model that leverages the sparsity of real-world graphs to efficiently generate new graphs. It avoids generating full adjacency matrices and can scale to

Table 2: Comparison of our models to other graph generative models using MMD in 2D Grid dataset.

2D Grid				
Model Name	Degree	Cluster Coeff	Orbit	Spectral
Erdos-Renyi	0.79	2.00	1.08	0.68
B-A	1.86	**0**	0.72	NR
MMSB	1.88	0.13	1.23	NR
GraphVAE	7.07e-2	7.33e-2	0.12	1.44e-2
GraphRNN-S	0.12	3.73e-2	0.18	0.19
GraphRNN	1.12e-2	**7.33e-5**	1.03e-3	1.18e-2
GRAN	8.23e-4	3.79e-5	1.59e-3	1.62e-2
GDSS	0.11	5.06e-3	7.00e-2	NR
BiGG	4.12e-4	7.25e-5	**5.10e-4**	9.28e-3
UGGS-1	**3.70e-4**	**0.0**	5.60e-4	1.20e-2
UGGS-2	**1.64e-4**	**0.0**	**2.62e-4**	**6.30e-3**

Table 3: Comparison of our models to other graph generative models using MMD in Protein dataset.

Protein				
Model Name	Degree	Cluster Coeff	Orbit	Spectral
Erdos-Renyi	5.64e-2	1.00	1.54	9.13e-2
B-A	1.40	1.70	0.92	NR
MMSB	0.23	0.49	0.77	NR
GraphVAE	0.48	7.14e-2	0.74	0.11
GraphRNN-S	4.02e-2	4.79e-2	0.23	0.21
GraphRNN	1.06e-2	0.14	0.88	1.88e-2
GRAN	1.98e-2	4.86e-2	0.13	5.13e-3
BiGG	9.51e-4	**2.25e-2**	2.26e-2	**4.51e-3**
UGGS-1	**6.02e-4**	5.80e-2	**8.70e-3**	1.20e-2
UGGS-2	4.06e-3	8.82e-2	4.03e-3	8.31e-3

significantly larger graphs compared to other deep autoregressive graph generative models.

- tree-gen [23]: Using a tree decomposition reduces the graph generation problem to multiple tree generation. It also proposes a permutation invariant model for tree generation.
- GDSS (Score-based Generative Modeling of Graphs via the System of Stochastic Differential Equations) [15]: GDSS uses a novel score matching objective and a new solver for the system of SDEs to efficiently sample from the reverse diffusion process. It has been demonstrated to generate molecules close to the training distribution without violating the chemical valency rule, showing the effectiveness of the system of SDEs in modeling the node-edge relationships.
- JT-VAE [14]: A model based on decomposing the graph to junction trees and employing an encoder network for both tree and graph and for a generation it assembles junction trees sampled from embedding space.

5.1 Decoding Result

As shown in the table 5, by employing a more sophisticated decoding scheme that filters generated graphs based on their embedding distance, we can significantly enhance the quality of the generated

Table 4: Comparison of our models to other graph generative models using MMD in Lobster dataset

Lobster				
Model Name	Degree	Cluster Coeff	Orbit	Spectral
GraphVAE	2.09e-2	7.97e-2	1.43e-2	3.94e-2
GraphRNN-S	3.48e-3	4.30e-2	2.48e-4	6.72e-2
GraphRNN	9.26e-5	**0.0**	2.19e-5	1.14e-2
GRAN	3.73e-2	**0.0**	7.67e-4	2.71e-2
JT-VAE	0.163	**0.0**	5.86e-3	8.27e-2
Tree-Gen	2.94e-4	**0.0**	**2.23e-5**	1.88e-2
BiGG	**2.94e-5**	**0.21**	**1.51e-5**	**8.57e-3**
UGGS-1	7.53e-4	1.86e-2	0.12	1.91e-2

Table 5: Comparison of our models to other graph generative models using MMD in 3D Mesh dataset

3D Mesh				
Model Name	Degree	Cluster Coeff	Orbit	Spectral
Erdos-Renyi	0.31	1.22	1.27	4.26e-2
GraphVAE	OOM	OOM	OOM	OOM
GraphRNN-S	OOM	OOM	OOM	OOM
GraphRNN	OOM	OOM	OOM	OOM
GRAN	1.75e-2	0.51	0.21	7.45e-3
BiGG	2.56e-3	**0.21**	0.21	3.40e-3
UGGS-1	1.78e-2	**0.21**	**7.90e-2**	9.30e-3
UGGS-2	1.38e-2	0.97	**6.00e-2**	4.7e-2

graphs. To validate the hypothesis that embedding distance indeed captures graph similarity, we conducted a perturbation analysis using two perturbation methods: edge rewire and node addition.

In the edge rewire method, we changed the source and destination of an edge with a certain probability, essentially rewiring the connections. In the node addition method, we added a new node to the graph and connected it to some existing nodes with a given probability.

We then increased the perturbation probability and sorted the graphs based on the amount of perturbation. Simultaneously, we also sorted the graphs based on their embedding distance from the initial unperturbed graph. We then plotted their Spearman correlation to investigate whether the ranking returned by the embedding distance aligns with the actual amount of perturbation.The result is depicted in figure 4. This analysis is essential as it provides the foundation for our distance-based decoding strategy.

In this experiment, we used two graph embedding methods, FEATHER [21] and Graph Isomorphism Network [28], for comparison. The results, as illustrated in the corresponding figure, confirm the effectiveness of our proposed decoding strategy, suggesting that embedding distance is a useful measure for graph similarity.

5.2 Interpretable Measure

With the recent advancements in deep graph generative models, there is a need for explainable methods to evaluate and interpret the produced graphs. While previous studies have predominantly relied on visual inspections, statistical measures of select graph

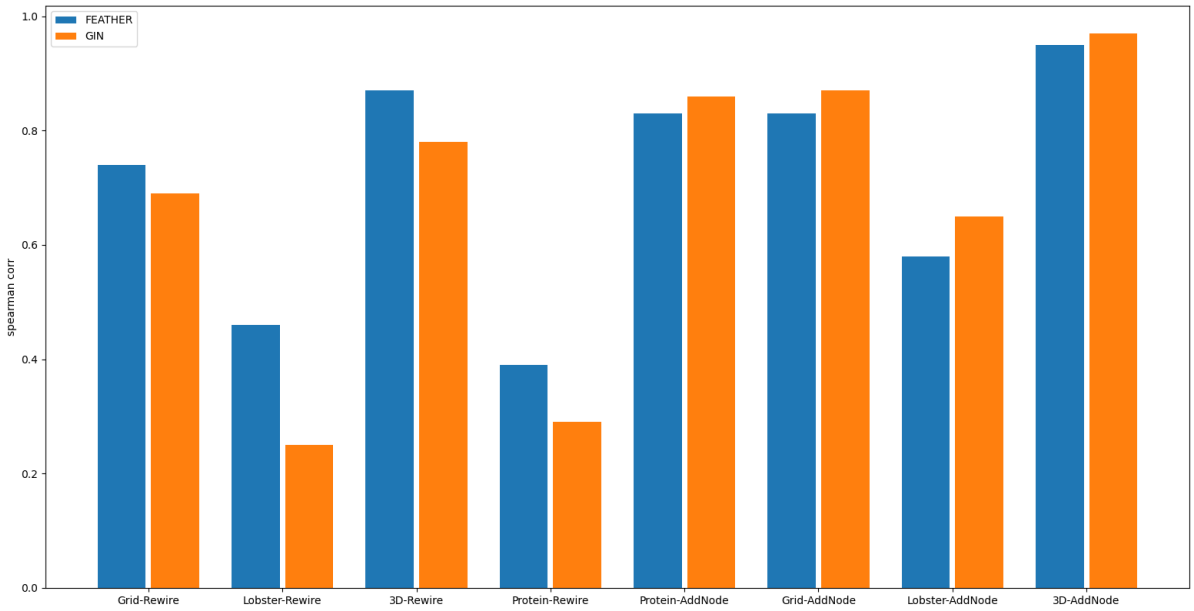

Figure 4: Comparison graph embeddings for distance-based decoding

Table 6: Comparison of our model's decoding strategy using MMD in 2D Grid, Protein, and Lobster datasets.

Model Name	Vanilla Decoding Strategy				Distance Based Decoding Strategy			
	Degree	Cluster Coef	Orbit	Spectral	Degree	Cluster Coef	Orbit	Spectral
Grid	8.27e-4	**0.0**	1.25e-3	**2.22e-3**	**1.86e-5**	**0.0**	**4.50e-5**	1.61e-2
Protein	2.09e-3	**5.63e-2**	5.91e-2	9.30e-3	**6.28e-4**	7.15e-2	**4.11e-4**	**5.92e-3**
Lobster	4.35e-3	**0.0**	2.00e-2	1.67e-2	**7.45e-4**	**0.0**	**6.28e-3**	**8.70e-3**

characteristics, and embedding-based evaluation metrics, this section focuses on introducing an interpretable evaluation measure for a typical graph comparison metric, Maximum Mean Discrepancy (MMD).

To elaborate, discussing MMD in isolation does not provide a comprehensive understanding of model performance unless it's compared with other models. Additionally, there is no clear understanding of the effect of a unit change in the MMD measure; for instance, the difference between an MMD of 0.01 and an MMD of 0.02. To address this issue, we express the MMD measure relative to the percentage of perturbation, such as edge or node perturbation, in a set of generated graphs. Therefore, every reported MMD is paired with a perturbation value, resulting in a more interpretable and accountable metric.

6 DEVELOPED FRAMEWORK AND CODEBASE

In an effort to propel further research and development in the field of graph generation, we have implemented a comprehensive framework that takes a modular and user-friendly approach. The framework is meticulously designed with simplicity and flexibility in mind, rendering it an accessible and valuable tool for researchers and practitioners alike. Our implementation encapsulates the entire graph generation process, from input to output. The conciseness and comprehensibility of our implementation are illustrated through an end-to-end example provided in the appendix.

Our framework is also designed to be highly compatible with a variety of popular graph deep learning libraries, including DGL [27] and PyTorch Geometric [10]. This feature enhances its flexibility and broadens its potential applications, as it can leverage the strengths and capabilities of these libraries to cater to a variety of graph-related tasks.

Moreover, we have open-sourced our framework and warmly welcome contributions from the broader research community. We believe that this collaborative approach will fuel innovation and accelerate progress in the domain of graph generation. By making our work openly accessible, we aim to catalyze further advancements in this exciting field and facilitate the discovery of novel solutions to complex graph generation problems.

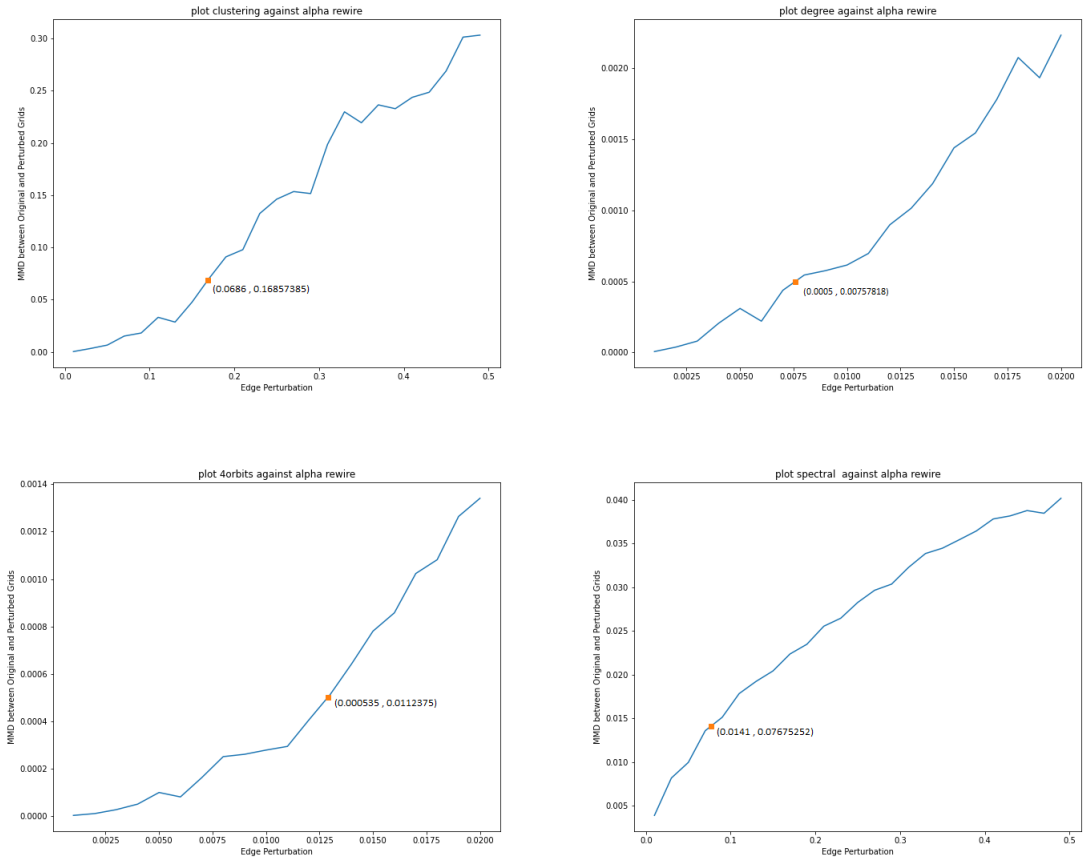

Figure 5: Proposed interpretable measure for four different criteria. The horizontal axis represents the percentage of perturbation in the edges of the training graphs, for example [0%,20%,...,100%], and the vertical axis represents the MMD between perturbed and reference for 100 grids. The highlighted points also represent the MMD of the proposed framework and the corresponding percentage of perturbation as the score in proposed measure. For example, suppose the model reports an MMD value of 0.0005 for the degree distribution. In that case, it is equivalent to perturbing about 0.75% of the edges of a reference graphs or, in other words, there is a 0.75% deviation from the training graphs.

7 CONCLUSION AND FUTURE WORK

In this work, we introduced a Unified Framework for Graph Generation based on Self-supervised Learning (UGGS), that encapsulates and expands upon existing methodologies. Our approach leverages the concept of pseudo-tasks to facilitate robust and efficient graph generation, demonstrating competitive performance metrics in terms of model parameters and training time, when benchmarked against state-of-the-art alternatives. Additionally, we shed light on the often-underestimated importance of decoding strategy, proposing a more sophisticated approach that is based on distance measures. Through rigorous testing and empirical analysis, we showcased the effectiveness of this approach, contributing valuable insights to the field.

Looking ahead, we identify several promising avenues for future research. First, our work emphasizes the significance of the decoding strategy in the graph generation process. As such, the development of more sophisticated decoding methodologies could yield significant improvements in model performance and the quality of generated graphs. Moreover, evaluating the quality and validity of generated graphs remains a nontrivial task. While we proposed an alternative, interpretable measure to MMD in this work, it is clear that more research effort is needed in this area to develop comprehensive, interpretable, and robust evaluation metrics. By advancing in these directions, we believe the field can continue to make substantial strides toward more powerful and efficient graph generation methodologies.

REFERENCES

[1] Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. 2022. Spanning Tree-based Graph Generation for Molecules. In *International Conference on Learning*

Representations. https://openreview.net/forum?id=w60btE_8T2m
[2] Edo M Airoldi, David Blei, Stephen Fienberg, and Eric Xing. 2008. Mixed membership stochastic blockmodels. *Advances in neural information processing systems* 21 (2008).
[3] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1. https://doi.org/10.1103/RevModPhys.74.47
[4] Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. 2020. Scalable Deep Generative Modeling for Sparse Graphs. *arXiv preprint arXiv:2006.15502* (2020).
[5] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models* (2018).
[6] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.
[7] Yasha Ektefaie, George Dasoulas, Ayush Noori, Maha Farhat, and Marinka Zitnik. 2023. Multimodal learning with graphs. *Nature Machine Intelligence* (2023), 1–11.
[8] P Erdos and A Enyi. 1959. On random graphs I. *Publicationes Mathematicae (Debrecen)* 6 (1959), 290–297.
[9] Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, and Hamid R Rabiee. 2021. Deep graph generators: A survey. *IEEE Access* 9 (2021), 106675–106702.
[10] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
[12] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.
[13] José Jiménez-Luna, Francesca Grisoni, and Gisbert Schneider. 2020. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence* 2, 10 (2020), 573–584.
[14] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*. PMLR, 2323–2332.
[15] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*. PMLR, 10362–10383.
[16] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[17] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. 2019. Efficient Graph Generation with Graph Recurrent Attention Networks. In *NeurIPS*.
[18] Kaushalya Madhawa, Katsuhiko Ishiguro, Kosuke Nakago, and Motoki Abe. 2020. Graph{NVP}: an Invertible Flow-based Model for Generating Molecular Graphs. https://openreview.net/forum?id=ryxQ6T4YwB
[19] Marion Neumann, Plinio Moreno, Laura Antanas, Roman Garnett, and Kristian Kersting. 2013. Graph kernels for object category prediction in task-dependent robot grasping. In *Online proceedings of the eleventh workshop on mining and learning with graphs*. 0–6.
[20] Mark Newman. 2018. *Networks* (2 ed.). Oxford University Press, London, England.
[21] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, 1325–1334.
[22] Chence Shi*, Minkai Xu*, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1esMkHYPr
[23] Hamed Shirzad, Hossein Hajimirsadeghi, Amir H. Abdi, and Greg Mori. 2022. TD-GEN: Graph Generation Using Tree Decomposition. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 151)*, Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.). PMLR, 5518–5537. https://proceedings.mlr.press/v151/shirzad22a.html
[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ
[25] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2023. DiGress: Discrete Denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=UaAD-Nu86WX
[26] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
[27] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).
[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km
[29] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems* 31 (2018).
[30] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*. PMLR, 5708–5717.
[31] Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. 2022. A Survey on Deep Graph Generation: Methods and Applications. In *The First Learning on Graphs Conference*. https://openreview.net/forum?id=Im8G9R1boQi

A LEARNED REPRESENTATION

figure 6 depicts the comparison of the learned embeddings between the reference and generated graphs, and those of the randomly generated graphs. The results demonstrate that randomly generated graphs cannot be compared to the graphs generated using the proposed framework, as evidenced by the significant distance between their embeddings.

B CODE SNIPPET

```
import torch
import numpy as np
from preprocess.dataprovider import GranDataProvider
from preprocess.taskdataprovider import LPDataProvider,
NodeDegreeDataProvider
from model.gnn_model import GraphGIN
from model.prediction_model import MLPPredictor
from model.loss import compute_loss
from train.lp_train import VanillaLPTrain
from decode.decode import GraphSamplingDecoder
from evaluation.mmd import evaluate_mmd
#### CONFIGS
device = torch.device("cuda:0" if torch.cuda.is_available()
else "cpu")
GRAPH_TYPE = "lobster"
max_n = 1000
dim = 128
num_class = 50
#### PREPARE DATA
graph_provider = GranDataProvider(GRAPH_TYPE)
task_provider = LPDataProvider(feat="onehot", max_n=max_n,
neg_over_sample=8)
n_provider = NodeDegreeDataProvider(num_class)
### Create Models
model = GraphGIN(max_n, dim, "mean")
pred = MLPPredictor(dim)
ts = VanillaLPTrain(model,
pred,graph_provider,task_provider,compute_loss,
save_dir="Exp/experiemt1/",
)
```

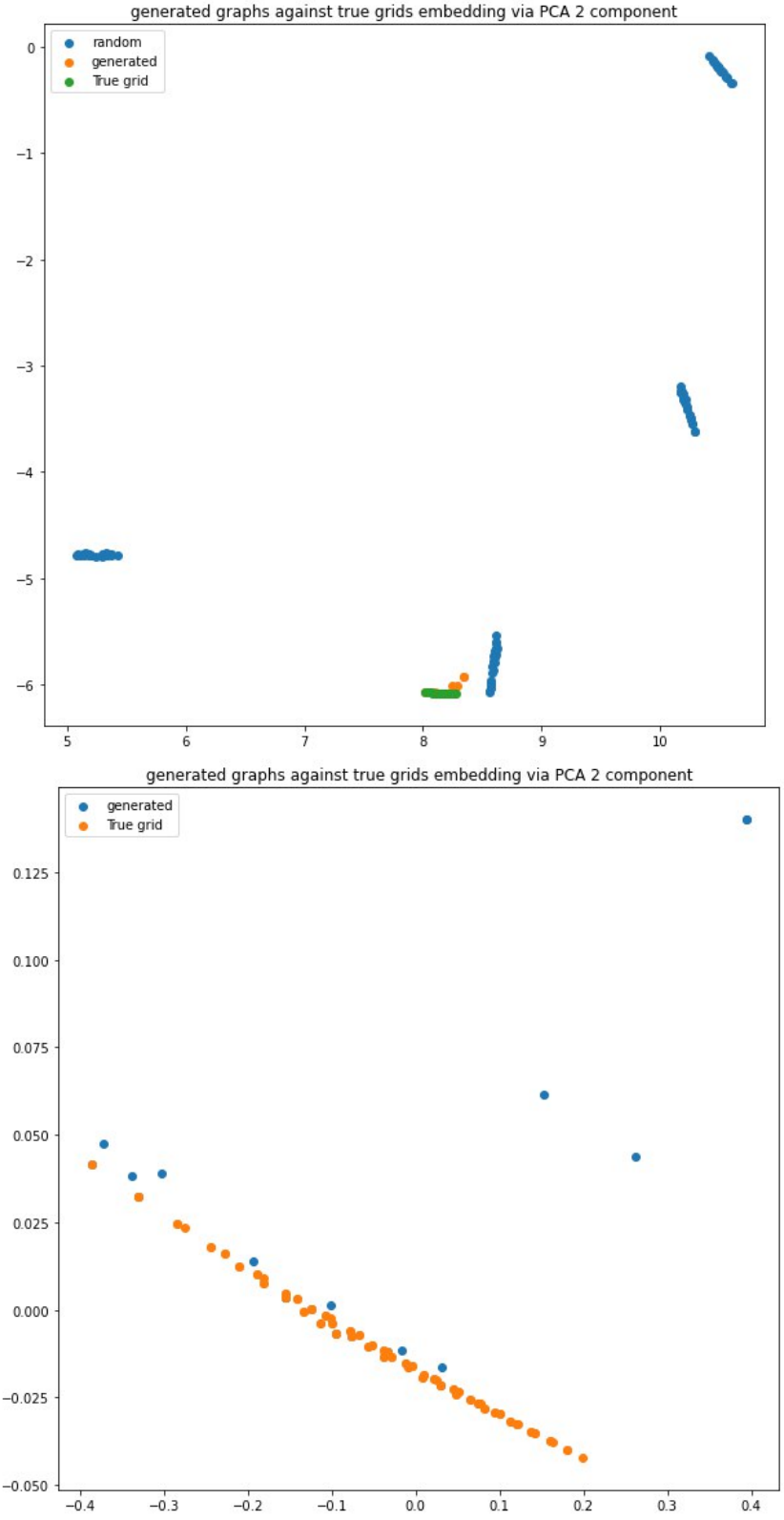

Figure 6: Comparison of the learned embeddings between the generated and reference graph datasets using principal component analysis (PCA). The embeddings of 20 graphs generated using the proposed framework were compared to the representations of 100 training 2D Grids. The results show that a significant proportion of the generated graphs are in close proximity to the reference graph embeddings line.

```
model, pred = ts(num_epochs=3000, save_step=50)
### Generation
graph_provider.set_mode("train")
graph_decoder = GraphSamplingDecoder(
graph_provider,task_provider,
gnn_model=model,
prediction_model=pred,
plot=False,
num_gen=20,
)
gen, time_dif = graph_decoder()
graph_provider.set_mode("test")
(
mmd_degree_test,mmd_clustering_test,mmd_4orbits_test,mmd_spectral_
) = evaluate_mmd(graph_provider, gen, degree_only=False)
```