Generating Graphs with Specified Properties

Clementine Lauvergne et Thomas Gravier

Introduction

The objective of this challenge is to generate graphs with specific properties leveraging the Neural Graph Generator (NGG) architecture described in [1]. This approach utilizes conditioned latent diffusion models for graph generation, combining a Variational Autoencoder (VAE) to process the graph structure with a diffusion process in the latent vector space. The diffusion is guided by conditioning vectors that encode graph statistics, ensuring the generated graphs align with desired properties.

1 Overall Method

We started by familiarizing ourselves with the model and understanding each of its components. We red the article on the baseline, as well as documentation on the state of the art in graph generation. As the paper is already a state-of-art in the field, we decided to focus on improving the already existing architecture instead of trying other models such as GAN's and AutoRegressive methods which are known to be more unstable to train and require huge computational resources.

To systematically improve the model's architecture, we adopted a rigorous scientific methodology. For each iteration, we made a single modification to an element of the baseline model and carefully analyzed the results in comparison to the original one. This controlled approach ensured that we could isolate the effects of each change, providing clear insights into how individual adjustments influenced the model's performance.

2 Experiments

2.1 Encoder

The encoder for the baseline model is GIN (Graph Isomorphism Network). While GIN has proven to be a powerful choice, we still decided to explore alternative Graph Neural Network (GNN) encoders to assess whether they could improve the model's performance. We firstly tried a classical GCN but it did not improve the results.

As discussed in class, attention mechanisms provide a significant advantage by assigning interpretable importance scores to neighbors, highlighting which nodes exert greater influence.

Thus, we decided to try the Graph Attention Network (GAT) especially since the graph size are relatively small (number of nodes capped at 50) so it would not increase too much the memory and computational resources needed. The GAT Encoder improved a lot our performance. We also tried TransformerConv and SAGEConv. Actually, SAGEConv has been the one performing the best.

2.2 Decoder

We observed a significant mismatch between the hidden dimensions of the decoder, which were set to the default value of 128, and the shape of the adjacency matrix, which was 2450. This discrepancy suggested that the decoder might struggle to effectively bridge the dimensional gap. To address this, we implemented a "growing" layer MLP (Multi-Layer Perceptron) that progressively expands the hidden dimensions to align more closely with the adjacency matrix, ensuring a smoother and more consistent transformation process.

Additionally, when training the Variational Autoencoder (VAE), we recognized the importance of incorporating information about the graph's features into the decoder. By injecting these features, we aimed to enable the decoder to "learn" to generate outputs that are informed by the inherent structure and attributes of the graph. This approach was intended to improve the decoder's ability to generalize effectively to unseen data in the test set, fostering better overall performance.

We also experimented with integrating attention mechanisms into the MLP conditioning within the decoder. The goal was to leverage attention to prioritize and dynamically weight critical aspects of the input features during decoding. However, despite its theoretical promise, this modification did not yield any noticeable improvements in our results, suggesting that the current architecture might already capture the necessary information effectively, or that the added complexity was not well-suited to our specific task.

We also decided to implement a temperature scheduler mechanism for the Gumbel-Softmax operation. By gradually lowering the temperature over the course of training, we aimed to transition from more exploratory predictions to increasingly confident and deterministic outputs.

2.3 Denoiser

Inspired by the Stable Diffusion paper [2], we decided to experiment with a 1D-UNet as a denoiser. The objective was to use a two-channel input: one channel representing the embedded condition and the other corresponding to the generated latent variable. For implementation, we utilized the 1D-UNet model available in the Hugging Face library.

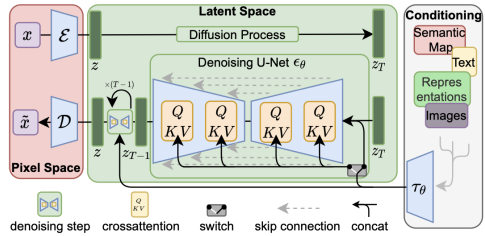

FIGURE 1. *Stable diffusion pipeline*

This approach led to a significant improvement in our denoiser's performance. However, training the 1D-UNet proved to be extremely time-consuming and demanded substantial computational resources, making it impractical for generalization. Due to these limitations, we ultimately decided not to include the 1D-UNet in our final model, despite its promising results.

2.4 Conditioning

At one point, we considered encoding the graph's features using a language model. Given its reputation for effectiveness, we decided to try the RoBERTa model. However, we encountered several issues with this approach. Our sequences contained a total of only a few words, typically resulting in tokenization ranging from 60 to 75 tokens and the embedding dimension of the RoBERTa model is 768. Thus, it appeared unreasonable to retain such huge matrices for encoding the condition in our baseline model, particularly given that the features in our case could be extracted more deterministically.

However, we recognize that for more complex tasks, such as those involving diverse sentence structures, incorporating such embeddings could be beneficial. Thought, a method like spaCy that only extract the features would likely outperform a large language model (LLM) in this context, where the linking words of a sentence doesn't matter, while also being significantly more computationally efficient for sustained use.

2.5 Inference

We also realized that monitoring the loss alone was not sufficient for evaluating our model's performance. Indead, even if we had the lowest validation possible for both AutoEncoder and Denoiser, we still didn't obtained good results during the evaluation phase. We needed a way to directly compare the generated results to determine which one was the best.

Initially, we considered retraining the denoiser and decoder together with a very low learning rate, using a score derived from graph features as the optimization objective. However, this approach proved challenging. Calculating metrics like the exact

number of nodes or clustering coefficients from differentiable functions turned out to be difficult. Additionally, using the F1 adjacency matrix loss would not have improved our results, as two graphs can share the same 7 given properties but have different adjacency matrices. Therefore, we were unable to successfully implement this method.

As an alternative, we drew inspiration from the approach outlined in the paper to evaluate the method's accuracy. We developed a custom solution to calculate the Mean Absolute Error (MAE) between the features of the generated graph and those of the original graph. Thus, we were able to implement a Rejection Mechanism, where the objective is to generate multiple candidate graphs and select the one whose features most closely matched the ground truth.

Thought we recognize that it is not scalable, this strategy turned out to be highly effective, providing a reliable way to evaluate and select the best results.

3 Final Model

In the end, we decided to retain the simplest baseline. Indeed, we believe the best aspect of the paper lies in its scalability compared to other models, while maintaining comparable effectiveness. Consequently, we preserved the original architecture but introduced a few key modifications: replacing the GIN Encoding with SAGE Encoding, adding growing layers concatenated with the condition to the decoder, and adjusting the F1 loss calculation from mean to sum.

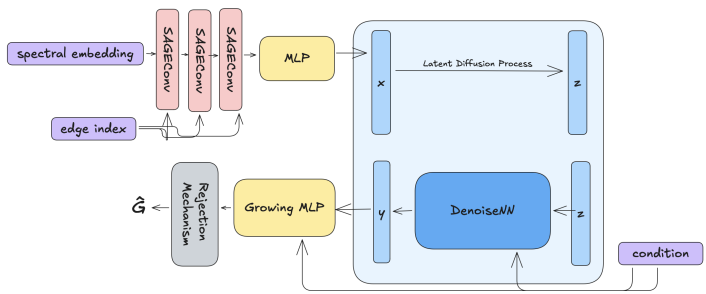

FIGURE 2. *Our final pipeline*

Additionally, we performed extensive fine-tuning of the model's parameters to further optimize its performance. The hyperparameters we focused on included the number of epochs, the number of layers of our modified decoder, the learning rate with a scheduling strategy, and the beta parameters for the autoencoder's loss function. We systematically explored a wide range of parameter values, adjusting one parameter at a time while closely monitoring the Mean Absolute Error (MAE) to assess the impact of each change.

Epochs AutoEncoder	Epochs Denoiser	Modification	Nb of Graphs	MAE Score
200	200	Base	10	0.029
200	200	Base	10	0.014
200	200	Denoiser (5 layers)	10	0.021
200	200	Denoiser (3 layers)	10	0.014
400	400	Denoiser (4 layers), scheduler after 200 epochs, added epochs	10	0.015
500	400	LR = 1e-4, Denoiser (4 layers), added epochs	10	0.017

TABLE 1. *Results of fine-tuning the AutoEncoder*

4 Critics and Results

Understanding the relationship between the loss function and the model's performance score, represented by MAE, proved to be challenging. It would have been beneficial to have a more intuitive loss function that could directly support back-propagation since simply relying on the adjacency matrix would not have been effective in this case.

Also, implementing a method for extracting graph features from any text would have been a valuable, since it would allow to generate graph from any sentence. It would have facilitated the generalization of our model to other graph-based problems.

At the end, we obtained a final score equal to 0.05.

References

[1] Iakovos Evdaimon, Giannis Nikolentzos, Christos Xypolopoulos, Ahmed Kammoun, Michail Chatzianastasis, Hadi Abdine, and Michalis Vazirgiannis. Neural graph generator: Feature-conditioned graph generation using latent diffusion models. 2024.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany, Runway ML, 2021.