Directional diffusion models for graph representation learning - written report

Marlon Kapala
Chair 9, Department of Computer Science
TU Dortmund University
Dortmund, Germany
marlon.kapala@tu-dortmund.de

Abstract—**Graph representation learning deals with learning meaningful representations from given graphs to facilitate downstream tasks like graph or node classification. Meanwhile, diffusion models are a set of novel machine learning algorithms that have brought groundbreaking successes in the image domain.**

In this report I present the paper "Directional diffusion models for graph representation learning" for the Proseminar "Hot topics of Generative AI: Applications of LLMs and Diffusion Models", which aims to combine these two. The authors explore the inherent anisotropic structure of graph data and subsequently introduce a new kind of diffusion model, the directional diffusion model. This model takes into account that structure by employing the novel directional noise and shows state-of-the-art performance in graph and node classification benchmarks.

This report provides an in-depth analysis of the paper, including background context, a detailed description of the proposed model, an evaluation of the results and a discussion on its place within the broader research landscape.[1]

I. Introduction

Graphs are a fundamental data structure. From molecules to maps or social networks, when connections between different entities are important, a graph is usually the right data structure.

This makes graph learning such an important machine learning field as it has a direct impact on several decisive industries such as the health care industry. A fundamental part of graph learning is graph representation learning. Graphs are complex data structures, so preprocessing the data before feeding it into another model is usually necessary. Diffusion models have revolutionized image synthesis [13] and perform well at multiple other machine learning tasks; however, while having found use as graph-generating models, they have not yet brought much value to graph representation learning.

This report examines the work of Yang et al. [22], who propose a directional diffusion model (DDM) and are thus among the first to introduce the concept of diffusion models to graph representation learning. Their key innovation is directional noise, which preserves structural information in graphs more effectively than standard gaussian noise. By evaluating their approach and its implications, this report aims to provide a critical perspective on the potential and limitations of DDMs in graph learning.

II. Related Work

The number of papers published on diffusion models has increased exponentially in recent years. Originally from thermodynamics, the idea was introduced to machine learning in 2015 [16]. In 2020, Ho et al. presented fundamental improvements in the algorithm in a landmark paper that paved the way for contemporary research [4]. Initially, the purpose of the model was to use its reverse process to generate images. Not long after, improvements in the architecture lead to the model outperforming generative adversarial networks (GANs), the state-of-the-art at that time [12] [13].

Over time, diffusion models have been adapted to different tasks, such as visual representation learning. However, these representation learning methods primarily operate in the image domain [2]. In the graph representation learning domain, diffusion models have not yet been influential [6].

[1] This work was supervised by Tim Katzke, TU Dortmund University.

Yet, a new series of graph diffusion models have emerged, which deals with the task of graph generation, achieving significant successes in areas such as molecule generation [9].

III. Background

A. Graphs

A graph is a data structure defined as $G = (V, E)$. Here, $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes in the graph, while $E \subseteq V^2$ is the set of edges between these nodes. An edge exists between two nodes v_i and v_j iff $(v_i, v_j) \in E$.

In graph learning, it is common to encode these edges in an adjacency matrix $A \in \{0, 1\}^{n \times n}$ while $n = |V|$. The adjacency matrix is defined as follows:

$$A_{ij} := \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{if } (v_i, v_j) \notin E \end{cases} \tag{1}$$

There are many ways to make the data structure more complex. We will discuss two important ones. Firstly, adding weights to the edges is usually desired. For example, in a road network, it is not only important to determine the existence of a connection between two locations, but also the distance or travel time associated with that connection. When modeling molecules, it may not only be interesting if there is a bond between two atoms, but it may also be interesting how many electrons these two atoms share. Thus, A is usually defined as $A \in \mathbb{R}^{n \times n}$ so that any real value can describe an edge instead of 1.

Secondly, it almost always makes sense to give the nodes additional attributes. Hence, we introduce the node embeddings $X \in \mathbb{R}^{n \times d}$. Thus, each of the n nodes of the graph is associated to a d-dimensional vector which contains further information about it. Eventually, we can then define a graph as follows:

$$G := (X, A), \qquad X \in \mathbb{R}^{n \times d}, \qquad A \in \mathbb{R}^{n \times n} \tag{2}$$

Note that the adjacency matrix A and node features X are indexed by the set of nodes and thus V can be omitted here. We can also introduce node labels and graph labels. For example, for a molecule that is modeled by a graph, it may be interesting how it smells, so we give it a corresponding label. These will be the labels we can then perform classification or regression on.

To illustrate the applicability of graphs in machine learning, we briefly examine an exemplary graph dataset. The IMDB-Binary dataset is a benchmark dataset introduced in 2015 [21] that is also used in Yang et al.'s paper [22]. The creators assembled two collaboration graphs for the action and romance genre. Each node represents an anonymous actor or actress and is connected to another node if the two actors/actresses have co-starred in the same movie. The creators then created a large set of graphs by splitting these collaboration graphs into many ego networks where at least one node, the ego, is connected to all other nodes. As the dataset anonymizes all actors/actresses and does not provide further information, there is only information on the structure of the collaborations given in the graphs. However, only from the structure of those ego networks models exist that can predict whether these ego networks belong to the romance or to the action genre with an accuracy of up to 96% [11]. This demonstrates the high utility of graph learning.

As graphs are complex data structures, the effective employment of machine learning methods usually requires converting the graphs to another form of data before-hand. Graph representation learning methods provide algorithms to do that. Here, graph neural networks are fundamental. Graph neural networks are a set of neural networks that deal with training on graphs. There are many different kinds of GNNs, but one prominent one is the *graph convolutional network* (GCN) [7]. It works similarly to convolutional neural networks (CNNs), which work on images. In fact, an image can also be represented as a graph where each pixel corresponds to a node and edges connect neighboring pixels. Thus, the GCN can be seen as a generalization of a CNN. Thus, similar to CNNs, information is also passed between nodes. A GCN layer looks like this [7]:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{3}$$

Here, $W^{(l)}$ are trainable weights while $H^{(l)}$ represents the node embeddings at step l ($H^{(0)} = X$). The identity matrix I_N is added to A, yielding $\tilde{A} = A + I_N$ to preserve information of the nodes‘

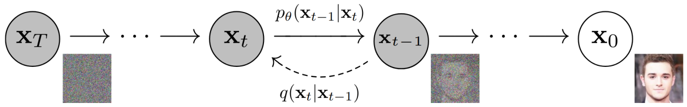

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Fig. 1. Adapted from [4]. The arrows facing left represent the forward process where on clean data, as visible on the right, noise is gradually being added until the data resembles pure noise, visible on the left. The arrows facing right represent the reverse process where the model tries to remove the noise again.

own embeddings during the message-passing process. $\tilde{D}$ is a diagonal matrix with $\tilde{D}_{ii} = \sum_j A_{ij}$ so $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ normalizes the adjacency matrix. The matrix multiplication $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}$ passes the node embeddings between the neighbors and σ eventually is a non-linearity function such as ReLU. Thus, information is propagated between neighboring nodes and trainable weights make it possible to optimize the network.

This neural network works well for node-related tasks such as node classification, but for tasks such as graph classification, the node embeddings must be aggregated. Graph pooling is used for this, which represents another conceptual similarity to the image domain [8]. Here, we differentiate between flat pooling and hierarchical pooling. Flat pooling directly aggregates the node embeddings using functions such as averaging or summation, while hierarchical pooling gradually creates smaller graphs by clustering or dropping nodes.

B. Diffusion models

Diffusion models are a class of powerful image synthesis algorithms [13] and the starting point for the directional diffusion model we will introduce later.

In its form introduced by Ho et al. [4], where the purpose is to create artificial intelligence, the diffusion model comprises a forward process that adds noise to the image data and a backward process that tries to remove that noise again, visible in Figure 1. Eventually, the trained model should be able to generate a proper image from purely random noise using the reverse process, therefore generating a new image.

The forward process is a Markov chain that gradually adds gaussian noise to the data:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{4}$$

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{5}$$

Here, x_0 stands for the original data, while $x_1, ..., x_T$ stand for the same data with noise with x_t representing the data to which noise was added t times. q is the respective probability density function. $\beta_1, ..., \beta_T$ are hyperparameters that control how much noise is added at each time step. Ho et al. made them linearly increasing [4], while later other scheduling techniques such as cosine scheduling [12] were introduced to add noise more slowly.

The forward process can also be described in the closed form

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I) \tag{6}$$

where $\alpha_t := 1 - \beta_t$ and $\overline{\alpha_t} = \prod_{s=1}^{t} \alpha_s$, making sampling at any timestamp t possible which is important for the training algorithm.

For the data with the most noise, x_T, the forward process is designed so that all information in the data is destroyed and it resembles a standard gaussian distribution. This can be expressed using the Kullback-Leibler divergence:

$$D_{KL}(q(x_T|x_0)\,||\,\mathcal{N}(0, I)) \approx 0 \tag{7}$$

By performing a gaussian transformation, equation (6) can be represented as:

$$q(x_t|x_0) = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon \tag{8}$$

$$\epsilon \sim \mathcal{N}(0, I) \tag{9}$$

This makes it possible to train a denoising neural network ϵ_θ to predict the noise ϵ from that noisy data. Thus, during training, the model tries to minimize the objective $||\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t)||^2$. This is the reverse process of the model where it tries to reconstruct the state of the data before noise was added. After training, that neural network can be used to generate a new image from new data by sampling pure noise $x_T \sim \mathcal{N}(0, I)$ and then gradually removing the noise by doing:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x, t)\right) + \sigma_t z \tag{10}$$

We can see here that we gradually remove the noise our model ϵ_θ predicts from the data until we arrive at x_0. We also add the product of $z \sim \mathcal{N}(0, I)$ and σ_t, another hyperparameter, to introduce controlled randomness to stabilize training and improve sample diversity. Ho et al. [4] use the successful U-Net architecture [15] for the denoising neural network ϵ_θ, but transformers have also been used for that [14].

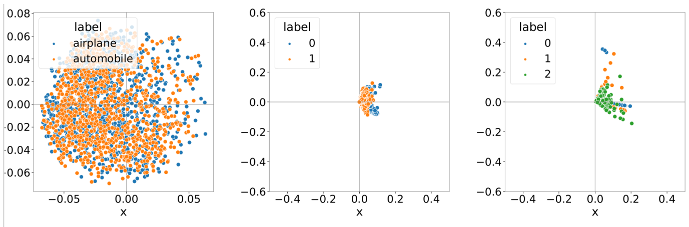
IV. Method

A. The anisotropic structure of graphs

In their paper, Yang et al. aim to develop a diffusion model which can handle graphs well. Thus, they start by exploring the structural differences between images - the kind of data structure that diffusion models were initially designed for - and graphs. To better understand the distribution of information in graphs and images, they first apply singular value decomposition to graph and image data to reduce it to two dimensions. As can be seen in Figure 2, this reveals that images are distributed much more *isotropically*, meaning that they fill up the entire space along both axes and almost form a perfect circle. Observing the 2D projection of the graph data reveals its *anisotropic* structure. Here, the distribution differs by direction; in fact, the data seems to be distributed only along certain directions. Yang et al. thus assume that there may be a structure in graphs inherently different from images.

B. Directional noise

With this discovery, Yang et al. develop a new kind of noise. As we have seen in the previous chapter, former diffusion models use *white noise* sampled from a standard gaussian distribution, i. e. $\epsilon \sim \mathcal{N}(0, I)$. This approach works well for image data, but it leads to rapid loss of directional information when applied to anisotropic distributions, such as graph data.



(a) CIFAR-10 (b) Amazon-Photo (c) IMDB-M

Fig. 2. Adapted from [22]. Three plots of image and graph data mapped to 2D using SVD. The data points of the CIFAR-10 image dataset seem to be equally distributed in all directions in a circle while the data points of the Amazon-Photo and IMDB-M datasets seem to be distributed only along certain directions.

Thus, Yang et al. propose a new kind of noise, the *directional noise*:

$$x_{t,i} = \sqrt{\overline{\alpha}_t} x_{0,i} + \sqrt{1 - \overline{\alpha}_t} \epsilon' \tag{11}$$

$$\epsilon' = \operatorname{sgn}(x_{0,i}) \odot |\overline{\epsilon}| \tag{12}$$

$$\overline{\epsilon} = \mu + \sigma \odot \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, I) \tag{13}$$

These equations may seem surprising at first, but in essence, it is still similar to what we have seen in the vanilla diffusion model [4]. The noise ϵ is still sampled from a standard gaussian and then transformed. However, the noise is here transformed using the mean μ and the component-wise standard deviation σ that were calculated from all data or, in reality, from the current data batch. Then, component-wise signs of the current data point $x_{0,i}$ are applied to that noise. Thus, the direction of the distribution is encoded in the noise and that information remains present even after many iterations.

Yang et al. repeatedly apply white noise and directional noise to graph data and analyze the signal-to-noise-ratio to further stress the superiority of directional noise when dealing with graphs. The ratio between signal and noise is crucial for diffusion models. If it drops too quickly during the forward process, this suggests that noise is added too fast, which makes it hard for the denoising neural network to successfully learn how to remove the noise.

Yang et al. calculate the signal-to-noise ratio by first training a feature extracting GNN to project the graph data to a linearly separable space before optimizing a weight vector $w \in \mathbb{R}^d$ using linear discriminant analysis. Then

$$SNR = \frac{w^T S_B w}{w^T S_W w} \tag{14}$$

where S_B is the scatter between different classes and S_W is the scatter between data points of the same class. Using this metric, the results Yang et al. present clearly indicate that directional noise seems to be much better at keeping information in the noise. At the same time, white noise seems to destroy information too fast if applied to graphs.

C. Model architecture

With that discovery, Yang et al. propose the *directional diffusion model* that directly incorporates directional noise. Thus, directional noise is applied in the forward process during training instead of white noise.

Since directional noise cannot be expressed in a closed form, the denoising neural network f_θ directly predicts the data X_0 and the objective function to minimize becomes $||f_\theta(X_t, A, t) - X_0||^2$. X_0, in this case, are the node embeddings mentioned earlier. We then pass the noisy node embeddings X_t and the adjacency matrix A so that the denoising neural network can make use of the graph's edges into it. Thus, the network is trained to reproduce the node embeddings of a given graph.

Previous diffusion models typically employ a U-Net architecture as a denoising neural network [4] and the DDM follows a similar approach. Similar to the U-Net [15] architecture, the denoising neural network consists of an encoder that reduces the dimensionality of the data and a decoder that increases it again. Another analogy is that it uses skip connections to transfer information of the data from early layers to late layers. However, the fundamental difference between the denoising neural network of the DDM and the U-Net proposed by Ronneberger et al. [15] is the kind of layers used. Ronneberger et al. initially designed the U-Net to solve the task of image segmentation. Hence, it contained convolutional neural networks. As the DDM handles graph data, it instead uses two layers of graph neural networks each in the encoder and in the decoder.

The implementation of the original publication reveals that these GNNs are *graph attention networks* (GATs). GATs are a kind of GNNs that work similarly to the graph convolutional network explained earlier [18]. However, instead of just normalizing the adjacency matrix, GATs use the attention mechanism to allow each node to dynamically weigh the connections to its neighbors for the message passing process. Attention is a concept which has already been very successful in other domains such as in transformers [17]. Between the encoder and the decoder and after the decoder, there are also two multilayer-perceptron blocks (MLP-blocks) in the network.

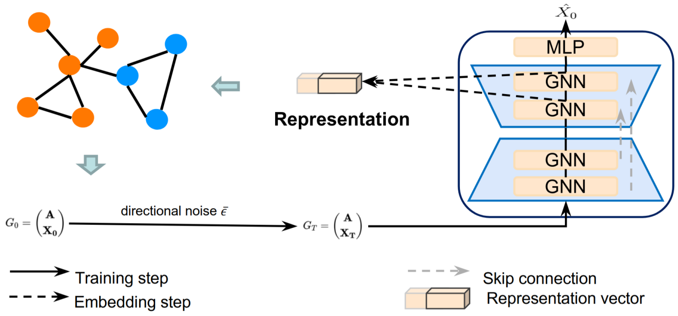

$\hat{X}_0$
MLP
GNN
GNN
GNN
GNN
Representation
$G_0 = \begin{pmatrix} \mathbf{A} \\ \mathbf{X_0} \end{pmatrix}$
directional noise $\bar{\epsilon}$
$G_T = \begin{pmatrix} \mathbf{A} \\ \mathbf{X_T} \end{pmatrix}$
Training step
Embedding step
Skip connection
Representation vector

Fig. 3. Adapted from [22]. First, directional noise is added to a graph G_0. Then, the noisy node embeddings and the adjacency matrix are passed into the denoising neural network. The result after the last MLP block is used for training while after training the latent code obtained from the intermediate GNN results is the representation.

The denoising neural network has two outputs. The first output is the result after the last MLP block and it is used to train the neural network. Hence, if training is successful, it will gradually become more similar to the node embeddings X_0. The second output is a concatenation of the intermediate results of the GNN layers of the decoder, which Yang et al. call the *latent code*. In the model's implementation, this concatenation is preceded by stacking of the intermediate results. Eventually, they are used as the generated representation. Figure 3 illustrates the overall architecture.

After training to convergence, the network can be used to generate representations from the graph. This is done by defining a set of forward steps $\{T_0, ..., T_K\}$. At each forward step k, noise corresponding to timestep k is added to the node embeddings of the input graph followed by denoising using the trained network. The latent codes of the results from each iteration are then concatenated to form the final representation.

V. Evaluation

A. Paper Benchmarks

While the way the model generates graph representations may seem arbitrary at first, the benchmark results suggest that it is a valid approach.

To evaluate their proposed model, Yang et al. let the DDM and several state-of-the-art representation learning algorithms generate representations for graphs of different benchmark datasets. They evaluate the model in two tasks: graph classification and node classification, both using support vector machines for classifying the representations.

The results, presented as the classification accuracies, are promising. Six benchmark datasets each are used for graph and node classification. The DDM achieves the highest accuracy in 5 out of 6 times of graph classification and in 4 out of 6 times of node classification. For the datasets where the DDM does not achieve the best accuracy, it still scores competitively.

Notably, despite being an unsupervised method, the DDM outperforms the supervised representation learning methods in 9 out of 12 experiments. This suggests that the DDM's structure may inherently align well with graph data.

Despite the promising benchmark results, several critical points need to be addressed.

Firstly, the benchmarks are limited to graph and node classification tasks. Thus, it is uncertain whether the DDM can perform similarly well on other graph learning tasks such as clustering or regression.

Another thing that raises suspicion is that the paper‘s code only provides functionality to reproduce the benchmarks of the DDM. This could imply that the researchers just took the benchmark results of the other models from their respective papers. At least for GraphMAE, this assumption seems to be accurate as the benchmarks in that paper are identical to those reported by Yang et al., both in graph and node classification [5].

This calls into question whether the benchmarks really present a fair comparison of the performances of the models. After all, Yang et al. even explain that they generate multiple representations with their DDM and then apply SVM separately to obtain a classification by majority vote. This gives the DDM an advantage during the benchmarking process and it is unlikely that all other models the model was compared to received the same advantage.

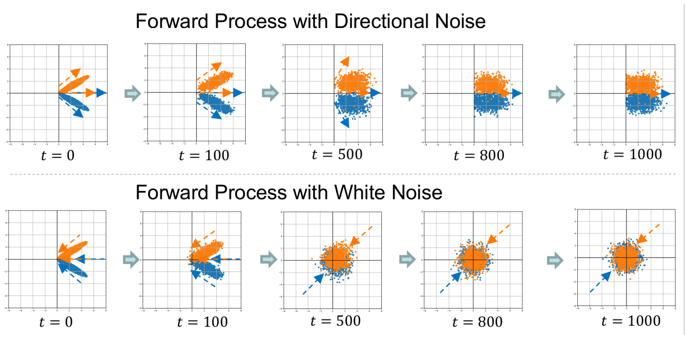


Fig. 4. Adapted from [22]. We can see that if white noise is added to these ellipses, all information is lost after 1000 forward steps, while the information of the direction is kept even after 1000 forward steps if directional noise is used.

However, a more profound investigation is necessary to fully conclude whether the conducted benchmarks are fair or not.

B. *A critical look on the directional noise*

It is uncertain whether it is actually a good idea to replace the original noise sampling process. The forward process of a diffusion model was originally designed to gradually destroy the information until it resembles a standard gaussian distribution in the end, as previously discussed.

However, this is not the case when using directional noise. Figure 4 shows that some information is always kept. In the experiments that were conducted for this report, this behavior also occurred[2]. This effect is also evident in the signal-to-noise ratio. With white noise the SNR always converges to zero, but this is not always the case with directional noise. In some of the experiments conducted by Yang et al., the SNR does not only fall slower than if white noise is applied, but it even rises (see Figure 5). Thus, in this case, the behavior is contrary to what is usually desired in diffusion models. Nevertheless, this observation should not be overstated. Unlike image synthesis diffusion models, which require the destruction of the signal to complete noise to be able to sample new data [4], this is different for DDMs. This is because there is no need to sample pure noise, since the input graph that a representation shall be created from is already the starting point.

[2] https://github.com/JavaLangMarlon/ddm-proseminar-tu-dortmund

VI. Research Context

A. Other Graph Representation Learning Techniques

Graph representation learning can be approached in various ways. While traditional approaches such as the neighborhood-based dimensionality reduction algorithm *Locally Linear Embedding (LLE)* exist, approaches based on GNNs currently dominate the field [6].

While approaches such as the DDM, which integrate GNNs into their methodology, have gained success, even standard GNNs remain competitive due to their effectiveness in extracting representations from graphs. In fact, the graph attention network that the DDM itself contains in the denoising neural network is used as a stand-alone comparison to the DDM in the benchmarks of the paper [22].

However, using diffusion models with directional noise remains a novelty in graph representation learning.

B. Alternatives to directional noise

Although directional noise is a novel addition to diffusion models, Yang et al. are not the first to explore the idea to sample ϵ from something other than a standard gaussian distribution. For instance, binomial diffusion was already introduced as an alternative to gaussian diffusion in the foundational work that introduced diffusion models to machine learning [16].

In recent years, diffusion models with various distributions have been introduced, such as the denoising diffusion gamma model that samples the noise from a gamma distribution [10].

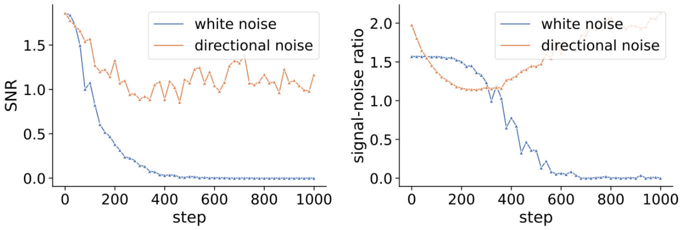


Fig. 5. Adapted from [22]. If white noise is applied to the graph data, the signal-to-noise ratio still converges to zero. This is not the case if directional noise is used. In fact, on the right side the signal-to-noise ratio even rises with additional directional noise, which is contradictory to the expected behavior of the forward process of vanilla diffusion models [4].

In the graph generation domain, it has also become a common methodology to sample noise from discrete distributions instead of continuous distributions, for example, by using a categorical distribution with predefined hyperparameters [3] [19].

Another approach that is even designed for graph representation learning - albeit limited to molecular representation learning - is the SubGDiff model [24]. During the forward process, it only applies noise to some regions of the graph. One could argue that this is also already a way of encoding structural information of the graphs in the noise, similarly to what is done in the DDM.

C. Follow-up papers

Directional noise, as introduced by Yang et al. [22], has already been applied in several domains.

In one case, it has been used to help solve the top-k recommendation task, which is another typical graph learning task where a model is trained to predict the k most favorable items to a user in a graph of users and items [23]. Similarly to Yang et al., the researchers initially observed a suboptimal performance of their diffusion graph transformer, which was significantly improved once directional noise was applied instead of standard gaussian noise.

Other applications of directional noise include the identification of drug-gene associations [20], the automated segmentation of stroke lesions on non-contrast CT images [25] and graph generation with diffusion models [1]. Yang et al. further suggest that the application of DDMs in natural language processing and computer vision provides exciting opportunities [22].

VII. Conclusion

This report has provided a comprehensive overview of the work by Yang et al. which introduces directional diffusion models, an approach that bridges two fields that have not yet had much contact with one another - diffusion models and graph representation learning. By introducing directional noise based on their observation about graph structure, the researchers make efficient use of diffusion models in graph representation learning possible.

While the reported benchmarks demonstrate impressive performance, there are potential concerns regarding fairness, comparability and reproducibility.

Despite these concerns, the DDM presents a promising alternative for unsupervised graph representation learning with early adoption in several fields such as recommendation systems.

The introduction of directional noise marks a significant step in the evolution of diffusion models. Moving forward, future research will show its long-term impact on diffusion models and graph representation learning and whether it can establish itself as a foundational technique in graph representation learning.

REFERENCES

[1] Fu, X., Gao, Y., Wei, Y., Sun, Q., Peng, H., Li, J., & Li, X. (2024). Hyperbolic Geometric Latent Diffusion Model for Graph Generation. *arXiv preprint arXiv:2405.03188.*

[2] Fuest, M., Ma, P., Gui, M., Fischer, J. S., Hu, V. T., & Ommer, B. (2024). Diffusion models and representation learning: A survey. *arXiv preprint arXiv:2407.00783.*

[3] Haefeli, K. K., Martinkus, K., Perraudin, N., & Wattenhofer, R. (2022). Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549.*

[4] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems, 33, 6840-6851.*

[5] Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., & Tang, J. (2022, August). Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 594-604).

[6] Ju, W., Fang, Z., Gu, Y., Liu, Z., Long, Q., Qiao, Z., ... & Zhang, M. (2024). A comprehensive survey on deep graph representation learning. *Neural Networks*, 106207.

[7] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*

[8] Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., ... & Tao, D. (2022). Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321.*

[9] Morehead, A., & Cheng, J. (2024). Geometry-complete diffusion for 3D molecule generation and optimization. *Communications Chemistry*, 7(1), 150

[10] Nachmani, E., Roman, R. S., & Wolf, L. (2021). Denoising diffusion gamma models. *arXiv preprint arXiv:2110.05948.*

[11] Nguyen, D. Q., Nguyen, T. D., & Phung, D. (2022, April). Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022* (pp. 193-196).

[12] Nichol, A. Q., & Dhariwal, P. (2021, July). Improved denoising diffusion probabilistic models. In *International conference on machine learning* (pp. 8162-8171). PMLR.

[13] Dhariwal, P., & Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34, 8780-8794.

[14] Peebles, W., & Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 4195-4205).

[15] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18* (pp. 234-241). Springer International Publishing.

[16] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015, June). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (pp. 2256-2265). PMLR.

[17] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems.*

[18] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903.*

[19] Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., & Frossard, P. (2022). Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734.*

[20] Wu, J., Gan, W., Lai, J., Chen, G., & Yu, P. S. (2024, November). Drug-gene Associations with Graph Learning. In *Proceedings of the 15th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics* (pp. 1-6).

[21] Yanardag, P., & Vishwanathan, S. V. N. (2015, August). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1365-1374).

[22] Yang, R., Yang, Y., Zhou, F., & Sun, Q. (2023). Directional diffusion models for graph representation learning. *arXiv preprint arXiv:2306.13210.*

[23] Yi, Z., Wang, X., & Ounis, I. (2024). A Directional Diffusion Graph Transformer for Recommendation. *arXiv preprint arXiv:2404.03326.*

[24] Zhang, J., Liu, Z., Wang, Y., & Li, Y. (2024). SubGDiff: A Subgraph Diffusion Model to Improve Molecular Representation Learning. *arXiv preprint arXiv:2405.05665.*

[25] Zhang, J., Wan, T., MacDonald, E., Menon, B., Ganesh, A., & Wu, Q. (2023). Synchronous Image-Label Diffusion Probability Model with Application to Stroke Lesion Segmentation on Non-contrast CT. *arXiv preprint arXiv:2307.01740.*