AlajVAE: Molecule Design Using Variational Autoencoders

Yassaman Ommi
Department of Mathematics and Computer Science
Amirkabir University of Technology (Tehran Polytechnic)
yassi.ommi@aut.ac.ir

Amin Gheibi
Department of Mathematics and Computer Science
Amirkabir University of Technology (Tehran Polytechnic)
amin.gheibi@aut.ac.ir

***Abstract*—Deep generative models have recently become a sought-after research criterion with many applications; i.e., graph generation methods are progressively being bespoken for drug discovery. Despite the availability of current graph generative models, they are often too broad and generic, and computationally demanding, which restricts their application to small molecules. On top of that, the majority of them are based on SMILES strings, which significantly increases the computational overhead, not to mention the burden of dealing with intricate grammatical rules. Therefore, we propose a VAE-based approach, ensuring that both encoder and decoder are graph-structured. Additionally, we have implemented state-of-the-art graph-mathcing algorithms to handle the issue of node-ordering and isomorphic graphs. Eventually, we compare our results with several renowned models, as baselines, further illustrating its potencies and shortcomings. An open-source python implementation of our code is accessible through GitHub.**

***Index Terms*—Generative Models, De Novo Design, Deep Learning, Variational Autoencoder, Molecular Graph Generation, Drug Generation.**

I. Introduction

Graphs are the natural data structure to represent relational and structural information in many domains, ranging from social networks to molecule structures. Therefore the ability to generate graphs can have many applications. The problem of random graph generation is a problem dating back to several decades ago, to the early work by Erdös and Rényi in the 1960s [1]. However, due to the simplicity and hand-engineered processes they have limitations regarding modeling the complex dependencies and can only create graphs with predefined statistical properties. For example, Erdös–Rényi graphs do not have the heavy-tailed degree distribution that is typical for many real-world complex networks. More recently, with the growing popularity of deep generative models, building deep graph generators has also attracted an increasing attention. This is mainly due to the fact that deep graph generators compared to older algorithms, have a greater capacity to learn structural information from data. Hence, they are effectively capable of modeling graphs with complicated topologies and constrained structural properties, such as molecules.

BSc. Final Project.

Recently, deep graph generation is utilized for facilitating the procedure of drug discovery. The principal goal of the drug design process is to find new chemical compounds that are able to modulate the activity of a given target (typically a protein) in a desired way. However, chemical space, the expanse spanning all possible molecules, is vast, and finding such molecules in it without any prior knowledge is nearly impossible. *De Novo* molecular design, mostly referred to as generative chemistry, aims at assisting this task with computer-based methods, arising from the increased popularity of generative models in AI. As such, generating molecules can be decomposed in two subtasks: First, to learn to represent molecules in a continuous space that facilitates the prediction and optimization of their properties (i.e., encoding), and second, to learn a mapping between the optimized continuous representation and molecular graphs with improved properties (i.e., decoding).
There has been some prior work [2], [3] on drug design that formulated the graph generation task as a string generation problem, using the SMILES representation, a linear string notation used in chemistry to describe molecular structures. However, this approach has major shortcomings. Not only SMILES does not capture the similarity between molecules, but also it does not express essential chemical properties as well as graphs. To address this issue, we are using molecular graphs to represent molecules, which we will define in the following sections.

In this work, we propose a VAE-based graph generator, suitable for chemical molecules. It is specifically designed to incorporate the chemical characteristics of drug molecules. Also, it could be easily modified to better suit the desired applications. Our method only uses the graphical representations of molecules, eliminating the overhead induced by dealing with SMILES strings and their respective formal grammar. Furthermore, it facilitates the use of many newly proposed methods—which are based on graphical representations of molecules—to be applied on top of our model, providing expandability and a platform for future endeavors.

II. Related Work

A generative model is a statistical model of the joint probability distribution $P(X, Y)$ on given observable variable X and target variable Y [4]; consequently, deep generative models (DGMs) are neural networks with many hidden layers trained to approximate complex, high-dimensional probability distributions using a large number of observations. When trained appropriately, DGMs can be used to estimate the likelihood of each observation and to create new samples from the underlying distribution [5]. It is a powerful scheme of learning any kind of data distribution using unsupervised learning and since its introduction, DGMs have become a hot trend in AI research criteria, as a result of their tremendous achievements in recent years. Despite the success in these domains, it is still challenging to correctly generate discrete structured data, such as graphs, molecules and computer programs.

Most popular approaches for implementing DGMs, consist mainly of Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). While VAE aims to maximize the lower bound of the data log-likelihood [6], GAN seeks to achieve an equilibrium between the generator and the discriminator [7].

Various molecular graph generation approaches have been proposed that mainly differ by the molecular graph representation. Li et al. [8] gives a comprehensive description of various models proposed for drug generation.

Most studies are based on SMILES[1] [9] strings. However, SMILES strings require a huge overhead for understanding their complex grammar. Moreover, a small change in the arrangement of characters can lead to a different molecule that has a significantly unalike structure or a molecule that is invalid. In this case, neighboring strings can not be used to predict the chemical characteristics of a given input string.

The logical stride to address this problem is to represent a chemical molecule through its molecular graph, since graph-based approaches typically do not suffer from the problem of invalidity of generated molecules. Several paradigms have been introduced to consolidate the use of graphs as the primary mean for representing distinct domains of data [10], [11], including but not limited to networks, molecules, etc. This subject area is otherwise known as Graph Representation Learning.

Many studies have been conducted that incorporate the use of graph-based methods to generate valid chemical molecules; for instance, [12] proposes a general graph convolutional network based model for goal-directed graph generation through reinforcement learning. Some other approaches use autoencoders to produce such graphs; i.e., [13] proposes a new hierarchical graph encoder-decoder that employs significantly larger and more flexible graph motifs as basic building blocks. Another similar application of this approached is implemented in [14], which generates molecular graphs with a guaranty that certain scaffolds will be present in the generated molecule, with a high probability. Nevertheless, property control poses a plethora of challenges in scaffold-based molecule generation owing to the fact that fixing a scaffold confines the chemical space, decreasing the possibility of finding desirable molecules.

[1]Simplified Molecular-Input Line-Entry System

Additionally, randomized models can also yield interesting results similar to those of real-world data. This is due to an important advantage of using random walks, which is their invariance under node reordering. Also, random walks only include the nonzero entries of a graph's adjacency matrix, thus efficiently exploiting the sparsity of real-world graphs. Taking that into account, [15] proposes *NetGAN*, claiming to be the first implicit generative model for graphs capable of mimicking real-world networks. In this approach, the problem of graph generation is presumed to be as learning the distribution of biased random walks over the input graph. On that grounds, the model has demonstrated the ability of producing graphs that exhibit well-known network patterns without explicitly specifying them in the model definition.

Besides the model generating valid molecular graphs, the possibility of optimizing the chemical properties of generative model should also be considered. A number of recently developed models have tackled the issue. Popova et al. [16] introduces the MolecularRNN based on GraphRNN [17], that generates valid molecules bundled with property optimization features. For each step, it employs two separate RNNs, *NodeRNN* for atom generation, and *EdgeRNN* for bond generation, together with valency-based rejection sampling. Consequently, in each step, it ensures that the current sum of all bonds does not exceed the allowed valency.

Furthermore, to improve the compound design process, [18] introduces Mol-CycleGAN—a CycleGAN-based model that generates optimized compounds with high structural similarity to the original ones. In order to do this task, it learns from the sets of molecules with and without a specific property.

III. Problem Formulation and Notation

In this section, we review the notations used in the paper and provide a problem formulation for the problem of graph generation.

A. Notation

We represent a molecule as a graph $G = (V, E)$, with V and E as the graph's set of nodes and edges, respectively. The nodes are viewed as the molecule's atoms and the edges as its bonds. Taking $|V| = n$, there are $n!$ possible node orderings for the graph; thus, if we choose an ordering π, the graph can be represented by the corresponding adjacency matrix $A^{\pi} \in R^{n \times n}$. The generated graph is also shown as $\tilde{G}$ with $\tilde{A}$ as its reconstructed adjacency matrix.

B. Problem Definition

Given a set of observed graphs $\mathcal{G} = \{G_1, G_2, \ldots\}$ sampled from the underlying data distribution $p(G)$, the goal of learning a generative models is to learn the distribution $p(G)$ of the observed set of graphs, and obtain new graphs by sampling from it. This can be done by either estimating $p(G)$ and then

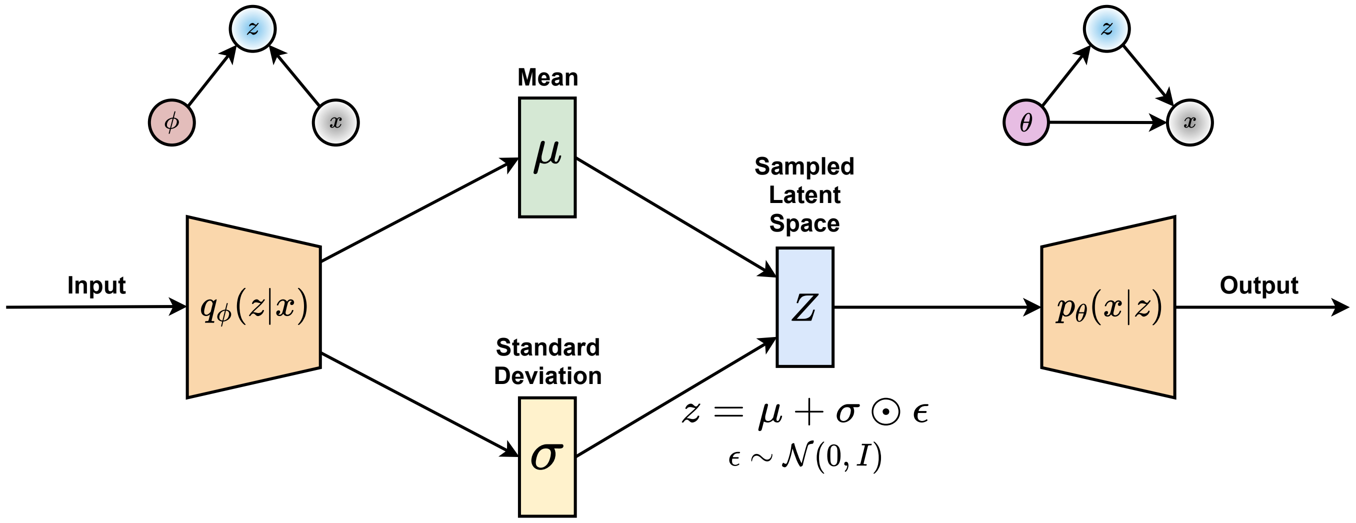

Figure 1: Schema of a Variational Autoencoder

sampling from it, or just implicitly sampling from it without explicitly having the distribution. The approach we use in our work is to encode the given graphs into a latent space, and use the embeddings to generate new samples.

IV. Method

Our approach extends the variational autoencoder framework, with the idea of outputting a probabilistic fully-connected graph and using a standard graph matching algorithm to align it to the ground truth. Initially, we briefly restate VAEs, and then continue by thoroughly presenting our method.

A. VAE

Variational Autoencoder is a latent space-based generative model, proposed in [19]. Suppose we have a dataset of samples x from a distribution parameterized by ground truth generative latent codes $z \in R^c$, where c refers to the length of the latent codes. VAE aims to learn a joint distribution between the latent space $z \sim p(z)$ and the input space $x \sim p(x)$. Specifically, the encoder is defined by a variational posterior $q_\Phi(z|x)$, and the decoder is defined by a generative distribution $p_\theta(x|z)$, with Φ and θ as parameters to be learned. A schema of the described model is shown in Figure 1. Furthermore, there is a prior distribution $p(z)$ imposed on the latent code representation as a regularization, which we have set $p(z) = \mathcal{N}(0, I)$. The whole model is trained by maximizing the lower bound of the true objective:

$$\mathcal{L}(\phi, \theta; x, z) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)). \tag{1}$$

where the second term, D_{KL}, stands for the non-negative Kullback–Leibler divergence (also known as relative entropy) [20] between the true and the approximate posterior.

B. AlajVAE

1) Graph Encoding: The goal of graph encoding is to generate a latent vector z of the entire graph G as a whole molecule. The input to our encoder is the graph's adjacency matrix A and node-feature matrix F, which indicates the class/feature of nodes as one-hot vectors, which also can be atom types in our context. Moreover, the encoder itself has two convolution layers to extract the nodes' features, which are then summed to represent the whole graph. Then, the encoder outputs are interpreted as mean $\mu(G)$ and variance $\sigma(G)$, which are then used to sample $z \sim \mathcal{N}(\mu(G), \sigma(G))$ via the re-parameterization trick [19]. Nevertheless, any other graph embedding method is applicable. Note that to sidestep the size mismatch problem in matrix multiplications of layers, the adjacency matrix A is padded with zero rows and columns so that all matrices have the same size N. Since they represent isolated nodes, they will not interfere with the embedding process.

2) Graph Decoding: The goal of this step is to generate the graph, which can be done either in an autoregressive style, generating one node at a time [8], [21], edge by edge [22], or to generate the whole graph at one step [23], [24]. Considering the discrete connected nature of graphs, they can have some connections with hidden meanings, which makes it hard to linearize their construction in a sequence. Additionally, iterative construction of discrete structures during training without step-wise supervision involves discrete decisions, which are not differentiable and therefore complicated for the back-propagation.
To dodge this issue, we have our decoder to output a probabilistic fully-connected graph $\tilde{G}$ with all the nodes at once, where the existence nodes and edges is modeled as Bernoulli variables. We reconstruct the adjacency matrix, $\tilde{A}$, of $\tilde{G}$, using the decoder's output. In this way, there is a probabilistic way to understand the predicted matrix $\tilde{A} \in [0, 1]^{N \times N}$, where the element $\tilde{A}_{a,a}$ denotes the probability of existence of node a,

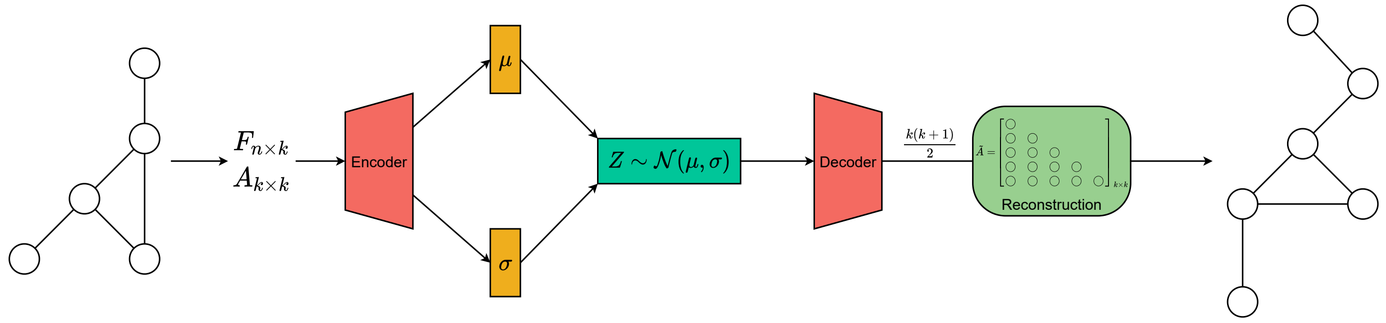

Figure 2: Schema of AlajVAE

and the element $\tilde{A}_{a,b}$ denotes the probability of existence of the edge between a, and b (with $a \neq b$).

C. Matching

Comparing two graphs G and $\tilde{G}$ having no specific node ordering can be so complicated, as their matrices are not invariant to permutations of nodes. To overcome this challenge, we have used the max-pooling graph matching algorithm proposed in [25], to align the mentioned graphs. See algorithm 1 for a more comprehensive perspective.

Algorithm 1 Max-Pooling Matching

```
Require: Affinity matrix $A$
Ensure: Soft-assignment $x$
1: Initialize the starting assignment $x$ as uniform
2: repeat
3:     for each candidate match $(i, a)$ do
4:         $x_{ia} \leftarrow x_{ia}A_{ia;ia} + \sum_{j\in\mathcal{N}_i} \max_{b\in\mathcal{N}_a} x_{jb}A_{ia;jb}$
5:     end for
6:     $x \leftarrow \frac{x}{\|x\|_2}$
7: until $x$ converges
8: Discritize $x$ if necessary
9: return $x$
```

1) Standard Graph Matching Formulation: Graph matching is a NP-hard problem, widely used in computer vision problems for tasks such as shape matching [26] or object recognition [27]. In these tasks, both a reference and a test scene are represented as graphs using visual features, and graph matching finds correspondences by minimizing the structural distortions between the two graphs. Suppose we are given $G = (V, E)$ and $G' = (V', E')$ where G is a target object model with its features as nodes and their relations as edges, and graph G' represents a scene. The goal of graph matching is to find correspondences $\mathcal{X} \in \{0, 1\}^{N\times N}$ between nodes of graphs G and G' based on the calculated similarity function, as a matrix defined as $S : (i, j) \times (a, b) \rightarrow \mathbb{R}^+$ for $i, j \in V$ and $a, b \in V'$. We have proposed our similarity function as follows:

$$\begin{aligned} &S((i,j),(a,b)) = \\ &\qquad = \frac{1}{|F_i - \tilde{F}_a| + 1} \cdot \tilde{A}_{a,a}[i = j \wedge a = b] \\ &\qquad + A_{i,j} \cdot \tilde{A}_{a,b} \cdot \tilde{A}_{a,a} \cdot \tilde{A}_{b,b}[i \neq j \wedge a \neq b] \end{aligned} \tag{2}$$

where F is the feature tensor obtained by summing the nodes' features. It is quite obvious that the first term evaluates the similarity between node pairs by comparing their features, in a way that the exact same features will yield the value $\tilde{A}_{a,a}$, which is the probability of presence of node a. Moreover, the second term evaluates the similarity between edge pairs by multiplying the probability of the existence of each node and the presence of the connection between them.

The max-pooling matching algorithm in [25], is a simple but effective algorithm following the iterative scheme of power methods, which outputs a continuous similarity matrix S^*. A more detailed description of the algorithm can be found in [25], and within the implementation of AlajVAE, which is available on GitHub.

V. Experiments

In this section, we empirically evaluate our model on both synthetic and real graph datasets. An overview of the characteristics of corresponding datasets is shown in Table I.

A. Datasets

Protein Graphs: We used the ENZYMES dataset for this category, which contains protein tertiary structures[2] representing 600 enzymes from the BRENDA enzyme database [28]. Nodes in a graph (protein) represent secondary structure elements, and two nodes are connected if the corresponding elements are interacting. Due to our limitations in computational resources, we have used a subset of the dataset, containing graphs with the most repeated number of nodes.

B. Baseline

We compare the performances of our model to three baseline models. One of them is the classical generative model of graphs coming from graph theory literature, namely the ER[3]

[2]The three dimensional shape of a protein
[3]Erdös–Rényi

Name	Domain	No. of Graphs	$\|V\|$	$\|E\|$	$\|\mathbb{V}\|$	$\|\mathbb{E}\|$
ENZYMES	Protein	575	$[2, 125]$	$[2, 149]$	3	X

Table I: The attributes of the datasets used in our experiments

model. The rationale behind this choice is to assess whether our model is able to perform better than random models that do not take into account edge dependencies (as the case in ER). We have also used the models proposed in [22] and [29], a deep graph generator, as baseline.

C. Metrics

Evaluating the task of graph generation can be challenging due to several reasons. First, graph generation includes stochastic outputs and additional aspects, making its evaluation different from problems with deterministic predictions. Moreover, graph structured data is much more difficult to evaluate than simple data with matrix/vector structures or structured signals such as images and texts. However, there are some metrics defined both to generally evaluate the sample and to evaluate based on the applications, which we will explore in the following subsections. We have reported the accessible metrics in Table II.

1) General Metrics: Generally, the quality of the generated graphs can be measured by computing the distance between some graph-based statistic distribution of the graphs in the test set and graphs that are generated. In doing so, we first select a statistic, and then compare its value in the ground truth distribution of the dataset with the generated samples' distribution. In this way, by selecting different statistics the represent various statistical properties of graphs, we can determine the similarity between our generated graphs and the original graphs. These statistics include, but are not limited to:

1) Node Degree Distribution: the empirical node degree distribution of a graph, which could encode its local connectivity patterns.
2) Clustering Coefficient: the empirical clustering coefficient distribution of a graph. Intuitively, the clustering coefficient of a node is calculated as the ratio of the potential number of triangles the node could be part of to the actual number of triangles the node is part of.
3) Orbit Count: intuitively, an orbit count specifies how many of these 4-orbits substructures the node is part of. This measure is useful in understanding if the model is capable of matching higher-order graph statistics, as opposed to node degree and clustering coefficient, which represent measures of local (or close to local) proximity.
4) Largest connected component: the size of the largest connected component of the graphs.
5) Triangle Count: the number of triangles counted in the graph.
6) Characteristic path length: the average number of steps along the shortest paths for all node pairs in the graph.
7) Gini Coefficient: The Gini Coefficient or Gini Index measures the inequality among the values of a variable. Higher the value of an index, more dispersed is the data. Alternatively, the Gini coefficient can also be calculated as the half of the relative mean absolute difference. Originally proposed by *Corrado Gini* [30] as a gauge of economic inequality, it is a statistical measure of distribution aiming to represent the income inequality or the wealth inequality within a nation or a social group. The coefficient ranges from 0 (or 0%) to 1 (or 100%), with 0 representing perfect equality and 1 representing perfect inequality.

The calculation can be done either by using their averaged value, or the distribution of the statistic, which we have adopted the former in our work.

In addition to the evaluation by measuring the similarity between the real and generated graphs, there are some other metrics that directly evaluate the quality of the generated graphs, such as:

1) Uniqueness: is utilized to capture the diversity of generated graphs, since high-quality generated samples should be diverse and similar, but not identical. To calculate uniqueness we first remove the samples that are subgraph isomorphic to some other generated samples, then the percentage of the remaining graphs is the Uniqueness. However, uniqueness is highly dependent on the pool size, and may significantly drop for a large generated library. (Note that identical graphs are subgraph isomorphic to each other.)
2) Novelty: measures if the model has learned to generalize unseen graphs, and is estimated as the percentage of generated graphs that are not subgraphs of the training graphs and vice versa.
3) Validity: can be a domain specific property to check, when the generated samples are desired to preserve some properties. These can be figured either visually in some cases (like generating tree, grid, lobster, cycle, etc.graphs), or computationally in other cases (like generating protein structures, SAT instance generation, drug molecules, etc.)

2) Application-based Metrics: In our case, we are interested in assessing special chemical properties of the molecules. These properties can be synthetic accessibility score (SA score), quantitative estimate of drug-likeness (QED), molecular weight (MW), log partition coefficient (logP), etc. SA score [31] is an estimation of how hard it is to synthesize a given molecule, which also reflects its structural complexity. Molecules with a higher score will be more complex and harder to synthesize. However, molecules with very low scores might be not complex enough to have the desired property. QED is another metric proposed in [32], which measures the drug-likeness of a given molecule, an important feature to consider in early stages of drug discovery. Furthermore, in

Dataset	Models	L.C.C.	Clustering Coef.	Mean Deg.	Max. Deg.	Min. Deg.	Gini Coef.	No. of C.C.	Training Time	No. of Epochs
ENZYMES	AlajVAE	2.233	1.062	3.84568	5.16666	1.83333	0.00631	0.016666	6h*	6*
	GraphGen	X	0.198	0.243	X	X	X	X	3h	4000
	GraphRNN	X	0.151	0.090	X	X	X	X	15h	20900

Table II: Comparison of AlajVAE and the baseline model's generated graphs' statistical properties. The rows include the absolute values of the differences between the generated and the original graphs. Smaller values of the absolute differences indicate better performance.

the task of molecule generation, Validity can be defined as the percentage of chemically valid molecules regarding some domain specific rules.

D. Limitations

AlajVAE, is supposed to be more effective for generating relatively small graphs (like molecules), due to the growth of GPU memory requirements and size of the graphs, as well as the high complexity of the matching algorithm.

VI. Future Works

Our future direction can focus on the scalability in generating larger graphs, improving the quality of the generated samples, and making the model more tailored for molecule design. There are ideas worth trying such as adding atom and bond types to the graphs' nodes and edges respectively or reconstructing feature tensors as well as the adjacency matrix. We would also like to extend its applications in real problems in chemistry, such as optimization of certain properties or predicting chemical reactions.

References

[1] P. Erdös and A. Rényi, "On random graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959. [Online]. Available: https://snap.stanford.edu/class/cs224w-readings/erdos59random.pdf

[2] A. Gupta, A. T. Müller, B. J. H. Huisman, J. A. Fuchs, P. Schneider, and G. Schneider, "Generative recurrent networks for De Novo drug design," *Molecular Informatics*, vol. 37, no. 1-2, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1002/minf.201700111

[3] J. Arús-Pous, A. Patronov, E. J. Bjerrum, C. Tyrchan, J.-L. Reymond, H. Chen, and O. Engkvist, "SMILES-based deep generative scaffold decorator for de-novo drug design," *Journal of Cheminformatics*, vol. 12, no. 1, p. 38, May 2020. [Online]. Available: https://doi.org/10.1186/s13321-020-00441-8

[4] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01. Cambridge, MA, USA: MIT Press, 2001, pp. 841–848. [Online]. Available: https://dl.acm.org/doi/10.5555/2980539.2980648

[5] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *GAMM-Mitteilungen*, vol. 44, no. 2, 2021. [Online]. Available: https://doi.org/10.1002/gamm.202100008

[6] C. Doersch, "Tutorial on variational autoencoders," 2021. [Online]. Available: https://arxiv.org/pdf/1606.05908

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680. [Online]. Available: https://dl.acm.org/doi/10.5555/2969033.2969125

[8] Y. Li, L. Zhang, and Z. Liu, "Multi-objective de novo drug design with conditional graph generative model," *Journal of Cheminformatics*, vol. 10, no. 1, p. 33, Jul. 2018. [Online]. Available: https://doi.org/10.1186/s13321-018-0287-6

[9] D. Weininger, "SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, Feb. 1988. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/ci00057a005

[10] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, vol. 40, no. 3, pp. 52–74, 2017. [Online]. Available: http://sites.computer.org/debull/A17sept/p52.pdf

[11] F. Chen, Y.-C. Wang, B. Wang, and C.-C. J. Kuo, "Graph representation learning: a survey," *APSIPA Transactions on Signal and Information Processing*, vol. 9, p. e15, 2020. [Online]. Available: https://doi.org/10.1017/ATSIP.2020.13

[12] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 6412—-6422. [Online]. Available: https://dl.acm.org/doi/pdf/10.5555/3327345.3327537

[13] W. Jin, R. Barzilay, and T. Jaakkola, "Hierarchical generation of molecular graphs using structural motifs," 2020. [Online]. Available: https://arxiv.org/pdf/2002.03230v2

[14] J. Lim, S.-Y. Hwang, S. Moon, S. Kim, and W. Y. Kim, "Scaffold-based molecular design with a graph generative model," *Chemical Science*, vol. 11, no. 4, pp. 1153–1164, 2020. [Online]. Available: http://dx.doi.org/10.1039/C9SC04503A

[15] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," 2018. [Online]. Available: https://arxiv.org/pdf/1803.00816v2

[16] M. Popova, M. Shvets, J. Oliva, and O. Isayev, "MolecularRNN: Generating realistic molecular graphs with optimized properties," 2019. [Online]. Available: https://arxiv.org/pdf/1905.13372

[17] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018. [Online]. Available: https://arxiv.org/pdf/1802.08773v3

[18] Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel, and M. Warchoł, "Mol-CycleGAN: a generative model for molecular optimization," *Journal of Cheminformatics*, vol. 12, no. 1, p. 2, Jan. 2020. [Online]. Available: https://doi.org/10.1186/s13321-019-0404-1

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014. [Online]. Available: https://arxiv.org/pdf/1312.6114v10

[20] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729694

[21] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, "Constrained graph variational autoencoders for molecule design," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 7806–7815. [Online]. Available: https://arxiv.org/pdf/1805.09076v2

[22] D. Bacciu, A. Micheli, and M. Podda, "Edge-based sequential graph generation with recurrent neural networks," *Neurocomputing*, vol. 416, pp. 177–189, Nov. 2020. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2019.11.112

[23] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, pp. 2434–2444. [Online]. Available: https://arxiv.org/pdf/1803.10459v4

[24] X. Guo, L. Zhao, Z. Qin, L. Wu, A. Shehu, and Y. Ye, "Interpretable deep graph generation with node-edge co-disentanglement," in *Proceedings of the 26th ACM SIGKDD International Conference on*

Knowledge Discovery & Data Mining, ser. KDD'20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1697–1707. [Online]. Available: https://doi.org/10.1145/3394486.3403221

[25] M. Cho, J. Sun, O. Duchenne, and J. Ponce, "Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers," in *2014 Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2014, pp. 2091–2098. [Online]. Available: https://doi.org/10.1109/CVPR.2014.268

[26] A. Sharma, R. Horaud, J. Cech, and E. Boyer, "Topologically-robust 3D shape matching based on diffusion geometry and seed growing," in *2011 Conference on Computer Vision and Pattern Recognition*, ser. CVPR'11, IEEE. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2011, pp. 2481–2488. [Online]. Available: https://doi.org/10.1109/CVPR.2011.5995455

[27] O. Duchenne, A. Joulin, and J. Ponce, "A graph-matching kernel for object categorization," in *2011 International Conference on Computer Vision*, ser. ICCV'11, IEEE. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2011, pp. 1792–1799. [Online]. Available: https://doi.org/10.1109/ICCV.2011.6126445

[28] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "BRENDA, the enzyme database: updates and major new developments," *Nucleic Acids Research*, vol. 32, pp. D431–D433, Jan. 2004. [Online]. Available: https://doi.org/10.1093/nar/gkh081

[29] N. Goyal, H. V. Jain, and S. Ranu, "GraphGen: A scalable approach to domain-agnostic labeled graph generation," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1253–1263. [Online]. Available: https://doi.org/10.1145/3366423.3380201

[30] L. Ceriani and P. Verme, "The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini," *The Journal of Economic Inequality*, vol. 10, no. 3, pp. 421–443, Sep. 2012. [Online]. Available: https://doi.org/10.1007/s10888-011-9188-x

[31] P. Ertl and A. Schuffenhauer, "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions," *Journal of Cheminformatics*, vol. 1, no. 1, p. 8, Jun. 2009. [Online]. Available: https://doi.org/10.1186/1758-2946-1-8

[32] G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan, and A. L. Hopkins, "Quantifying the chemical beauty of drugs," *Nature Chemistry*, vol. 4, no. 2, pp. 90–98, Feb. 2012. [Online]. Available: https://doi.org/10.1038/nchem.1243