Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 246 (2024) 4186–4193

Procedia Computer Science

www.elsevier.com/locate/procedia

28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2024)

NeuraGED: A GNN estimation for Graph–Edit Distance

Sara Bacconi[a], Filippo Costanti[a,b,*], Monica Bianchini[a], Niccolò Pancino[a], Pietro Bongini[a]

[a] *University of Siena, Department of Information Engineering and Mathematics, Via Roma, 56, 53100, Siena, Italy*
[b] *University of Florence, Department of Information Engineering, Via Santa Marta, 3, 50139, Florence, Italy*

Abstract

Graph generative models often lack a proper reconstruction loss to evaluate the distance between the generated graph and the target graph. This is particularly important for molecular graph generators based on autoencoders, which should reconstruct the input graphs as precisely as possible. Though, the distance estimation can be useful for any graph generator based on reconstruction, including sequential methods relying on Graph Neural Networks. Since graphs are discrete entities by nature, general graph spaces lack a reliable, general, and computationally affordable distance function. Graph Edit Distance is of course an exact, general, and permutation–invariant method for evaluating the difference between two graphs defined in the same graph space. Since it needs all the possible combinations of pairs of nodes from the two graphs, its exact computation is a NP-complete problem and cannot be carried out for graphs larger than ten nodes. As a consequence, a comprehensive soft–estimation method for Graph Edit Distance based on siamese Graph Neural Networks is proposed. A theoretical discussion is carried out, showing that the proposed method can provide a reliable and precise soft–estimation of the Graph Edit Distance on molecular graphs. Molecular graph generators can therefore use this distance estimation as a powerful non–permutation–invariant reconstruction loss. Moreover, the experimental results show that the distance estimation is accurate, with a very low Mean Squared Error loss value.

Keywords: Graph Edit Distance; Graph Neural Networks; Machine Learning; Deep Learning; Siamese Networks; ZINC dataset;

1. Introduction

The problem of generating graph data involves training a generative model that can produce new graph instances resembling those observed in a given dataset. Unlike other structured data, such as images or text, graphs exhibit unique challenges, due to their complexity and heterogeneity. This makes graph generation a difficult and crucial task in many domains, including social network analysis, recommendation systems, biological network modeling, and design of new chemical compounds. The main issue when dealing with graphs is, of course, their discrete nature: graphs or their encodings must be brought into a continuous domain, because discrete spaces hinder the possibility of

* Corresponding author.
E-mail address: filippo.costanti@unifi.it

calculating gradients and therefore applying Backpropagation–like algorithms to train generative models. Traditional generative approaches [1] struggle to capture intricate topological patterns, community structures, and attribute distributions present in complex graphs. By using neural networks to operate directly on graph–structured data, GNNs [2] can capture hierarchical relationships and local dependencies present in graphs, enabling effective generative modeling. However, generating realistic graph data poses several key challenges. First, the generated graphs must possess statistical properties and structural characteristics similar to those of the input dataset, including the degree distribution, average path length, and clustering coefficient. Additionally, the generated graphs should preserve the semantic meaning and functional properties encoded in the node and edge attributes, if available. Several approaches have been proposed to address these challenges, including sequential graph generation based either on recurrent models [3, 4, 5] or reinforcement learning [6, 7], Variational Graph Autoencoders (VGAEs [8, 9, 10]), Graph Generative Adversarial Networks (GGANs [11, 12]), and diffusion models for graph generation [13, 14]. Variational graph autoencoders learn a low–dimensional latent space representation of graphs, allowing for efficient sampling and interpolation between graph instances. Graph generative adversarial networks employ an adversarial training mechanism to generate realistic graphs by competing against a discriminator network. Sequential models iteratively generate nodes and edges conditioned on previously generated components, capturing the sequential nature of graph generation.

One of the main problems in training generative models is the choice of an adequate loss function. For one–shot models, such as variational autoencoders, the most commonly used loss functions is the Kullback-Leibler (KL) divergence [15] combined with a reconstruction loss, such as the mean squared error (MSE). MSE can compute the distance between two matrices but, in general, it does not guarantee to calculate an adequate topological distance for two adjacency matrices representing the connectivity of two graphs. In fact, there are many possible adjacency matrices for every single graph, one for each enumeration of the nodes. This means that comparing different adjacency matrices of the same graph can lead to inadequate values of the reconstruction error. Two main functions are designed for solving this problem: Graph Edit Distance (GED [16]) and Sub–graph Edit Distance (SED [17]). Unlike the SED, which violates the properties of identity and symmetry, the GED is considered a distance metric [18, 19] invariant to node order permutations. GED has an important application in inexact graph matching, especially useful in fault–tolerant pattern recognition [20]. Additionally, it has interesting applications in bioinformatics: for instance, it can be used to measure the distance between two molecular graphs [21, 22], a particular kind of labeled graph in which the node labels indicate the type of each atom and the edge labels indicate the type of each chemical bond. Moreover, having a model which computes the similarities or dissimilarities between molecular graphs could be functional also for implementing an efficient reconstruction loss for deep generative models. Unfortunately, the tasks of computing GED and SED have been demonstrated to be NP–hard [23]. Recently, some works have employed GNNs to estimate GED. In [18]iamese structure with Graph Isomorphism Networks (GINs) is used to predict GED and SED, with multiple MLPs for performing features encoding, decoding, and distance of embeddings. More complex is the structure of the solution reported in [24] here, graphs are processed by two Graph Convolutional Networks (GCNs [25] to produce node–focused embeddings, which are subsequently processed by a graphs global attention network to obtain graph–focused embeddings, and by a sub–module to obtain a node wise attention embedding. These embeddings are then concatenated and processed by an MLP to predict a GED–like similarity score. Despite achieving good performance, those approaches cannot be used in a generation task, since the number of nodes in the generated graphs is constrained by the dataset used to train the model (e.g. LINUX [26], AIDS[1] and IMDB [27] datasets), making the computation impossible for new datasets. Approximation of GED and SED could also be useful in many application fields, such as social sciences or bioinformatics [28], for example to locate similar subgraphs within larger graphs — to find similar communities in social networks [29] — or for matching similar and complementary substructures in drug discovery [30, 31, 32] and for protein–protein interaction prediction [33].

In this paper, a procedure to be used in graph generative frameworks is proposed, which consists in training a neural network to calculate a topological soft estimation of the GED using a simpler architecture compared with the literature, setting up the task as a supervised regression problem on a synthetic graph dataset generated for this purpose. The model consists of a GNN–based Siamese network [34, 35, 36], a type of architecture used for tasks involving similarity comparison or metric learning. Specifically, it is constituted by two identical subnetworks sharing the same parameters and architecture (for this reason, they are called "Siamese"), each taking a different input. The

[1] https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data

key aspect of the Siamese model lies in its ability to learn a similarity function, thus representing the ideal choice for the computation of the GED. The paper is organized as follows. In section 2, the GED learning problem and the dataset generation are described, together with the model structure; in Section 3, the experimental set–up is reported while the obtained results are discussed in Section 4. Finally, Section 5 collects some conclusions and depicts future perspectives.

2. Materials and Methods

2.1. Problem definition

A graph is a non–linear, non–Euclidean data structure defined as a pair $G = (V, E)$, where V represents the set of nodes and $E \subseteq V \times V$ is the set of edges. Nodes describe entities, while edges stand for the relationships between them. Nodes and edges can be associated with values or vectors of values describing their features — $x_v \in \mathbb{R}^{d_v}$, $\forall v \in V$, and $e_{u,v} \in \mathbb{R}^{d_e}$, $\forall (u, v) \in E$, respectively. The *stacked* versions of all the node and edge feature vectors constitute, respectively, the node feature matrix $\mathbf{X}_V \in \mathbb{R}^{|V| \times d_v}$ and the edge feature matrix $\mathbf{X}_E \in \mathbb{R}^{|E| \times d_e}$.

Let $G = (V, E, X_V, X_E)$ be a generic undirected, labeled graph, s.t. $G \in \mathcal{G}$, where $\mathcal{G}$ is a set of graphs. The GED between two graphs G_1 and G_2 in $\mathcal{G}$, denoted as GED(G_1, G_2), is defined as the minimum number of edit operations which transform G_1 in G_2, where an edit operation on the graph G_1 can be either a single insertion or a single deletion of a node or an edge, or even a node relabeling. The GED between two identical graphs, i.e. between two isomorphic graphs with the same node and edge features, is therefore defined as zero.

In this study, *NeuraGED* — a GNN–based approximation of the graph edit distance — is proposed. The model is trained on a dataset $\mathcal{L}_{\mathcal{G}} = \{\langle G_1, G_2, y\rangle : y = \text{GED}(G_1, G_2) \in \mathbb{R} \mid G_1, G_2 \in \mathcal{G}\}$ of tuples, where $y \in \mathbb{R}$ is the ground–truth value for GED(G_1, G_2), with $y(G_1, G_2) = y(G_2, G_1)$.

2.2. Dataset generation

The primary challenge associated with the GED calculation is its computational complexity. It has been proven that this problem belongs to the NP–hard class of complexity [23]: the NP–hardness of GED implies that computing the exact distance between two graphs is intractable for large or complex graphs, since it requires a prohibitive amount of time and resources. Hence, in order to obtain a soft estimation of this distance, the generation of a specific dataset, in which the target distance for each couple of samples is known, is proposed.

The ZINC dataset [37], a common and widely used dataset in drug discovery and molecular modeling, was utilized as a reference in this study [38]. ZINC contains information on the molecular structures, properties, and features of various chemical compounds, comprising molecular graphs with a maximum of 38 atoms, described by a set of 27 features. The generated dataset is composed of ordered triplets $\langle G_1, G_2, y\rangle$, where $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ describe two simple undirected and node–labeled graphs, i.e. undirected graphs which do not allow for parallel edges nor self–loops, while y represents the GED between G_1 and G_2. In order to simulate the insertion and deletion edit operations, and to guarantee a general balance of the dataset, the first half of the data was constructed so that G_2 is generated as an extension of G_1, i.e. with only the addition operation allowed in the generation procedure, while the second half of the dataset is built by exchanging the position of G_2 and G_1 inside the ordered triplets, so that only the deletion operation is considered. Let $\mathbf{G}$ be the set of undirected simple graphs and let $\mathbb{B}^{n \times m}$ be the set of all $n \times m$ binary matrices. The structure of the graphs is then constructed through a random generation of their adjacency matrices by means of a function $f : \mathbb{R} \to \mathbf{G}$, which generates a squared matrix $\mathbf{A} = [a_{ij}] \in \mathbb{B}^{k_1 \times k_1}$, given a number of nodes $k_1 = |V_1|$, sampled from a specific normal distribution. Moreover, since $G_1, G_2 \in \mathbf{G}$, the generated adjacency matrices are binary symmetric matrices with all zeros on the diagonal. In this context, isolated nodes are not allowed: if there are one or more nodes with no incident edges (i.e. null rows and therefore null columns in $\mathbf{A}_1$), random non–diagonal elements are set to 1. Given the two adjacency matrices $\mathbf{A}_1 \in \mathbb{B}^{k_1 \times k_1}$ and $\mathbf{A}_2 \in \mathbb{B}^{k_2 \times k_2}$, with $k_1 \leq k_2$, the topological estimation of the GED is computed as defined in Eq. (1):

$$y = n + \sum_{\substack{k_1 \le i \le k_2 - 1 \\ 0 \le j \le k_1 - 1}} a_{i,j}^2 + \frac{1}{2} \sum_{\substack{k_1 \le i \le k_2 - 1 \\ k_1 \le j \le k_2 - 1}} a_{i,j}^2. \tag{1}$$

The two summations in Eq. (1) count the number of elements equal to 1 in the two submatrices that can be extracted from $\mathbf{A}_2$ if only the rows associated to the added vertices are considered, hence the rows not belonging to $\mathbf{A}_1$. Fig. 2 shows the GED estimation from a graphical point of view: the sum of the second submatrix, colored in red, is divided by two since undirected graphs are considered and because of the symmetrical nature of $\mathbf{A_2}$, otherwise the corresponding edges would be counted twice.

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Fig. 1. Adjacency matrix $\mathbf{A}_1$

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Fig. 2. Adjacency matrix $\mathbf{A}_2$

In order to measure the performance of the proposed model with different initialization methods for node labels, and to generate the final samples $\langle G_1, G_2, y \rangle$, the associated feature matrices $\mathbf{X}_1 \in \mathbb{R}^{k_1 \times d_v}$ and $\mathbf{X}_2 \in \mathbb{R}^{k_2 \times d_v}$ are generated as well, together with the target y and the maximum diameter D for each pair. On the one hand, if a generic graph G is connected, then its diameter $diam(G)$ is computed. On the other hand, if G is not connected, its diameter would be infinite, therefore the relative sample is discarded. Hence, four possible situations can occur:

- Both G_1, G_2 are disconnected, then $D = 0$;
- G_1 is connected and G_2 is disconnected, then $D = diam(G_1)$;
- G_1 is disconnected and G_2 is connected, then $D = diam(G_2)$;
- Both G_1, G_2 are connected, then $D = max\{diam(G_1),\ diam(G_2)\}$.

2.3. NeuraGED Model

The GED estimation is carried out by NeuraGED, a hybrid model consisting of a GNN–based Siamese neural network followed by a Multilayer Perceptron. In particular, Siamese networks for graphs [39, 40] approximate well the embedding function $f_1 : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^h \times \mathbb{R}^h$, where $\mathcal{G}$ is the set of graphs and h is the dimension of the embedding space.

GNNs have been demonstrated to be universal approximators on graphs [41]. Moreover, it has been proved that a GNN with a sufficient number of message passing iterations and an injective readout function is as powerful as the first–order Weisfeiler–Lehman test. This holds true both for recurrent GNNs and for some convolutional GNN models [42]. As a consequence, the choice of GINs — which were purposefully designed for maximizing the isomorphism recognition capability — seems to be natural. GINs calculate message–passing iterations in stacked convolutional layers which do not share the weights, according to Eq. (2):

$$h_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right) h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right), \tag{2}$$

where $h_v^{(k)}$ is the state of the node v at layer k, $\epsilon^{(k)}$ is a learnable parameter at layer k, and $\mathcal{N}(v)$ is the one–hop neighborhood of node v. Once the two graphs G_1 and G_2 are fed in input to the Siamese GIN, after k message–passing iterations (carried out by the same number of layers), the state of every node represents a natural node embedding (k is a hyperparameter of the network). The two graph embeddings are obtained using a global pooling with max, average or sum aggregation mode.

After the graph embeddings E_1 and E_2 are computed, they are concatenated and fed in input to a MLP which approximates the function $f_2 : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$, such that $f_2(E_1, E_2)$ approximates GED(G_1, G_2). Setting the activation function of the last layer of the MLP with a *ReLU* guarantees the non negativity of f_2. Moreover, given that GNNs are universal approximators on graphs [43, 44], it can be concluded that f_2 maps the two embeddings in the GED value of the correspondent pair of graphs [45]. Fig. 3 shows the structure of the NeuraGED model.

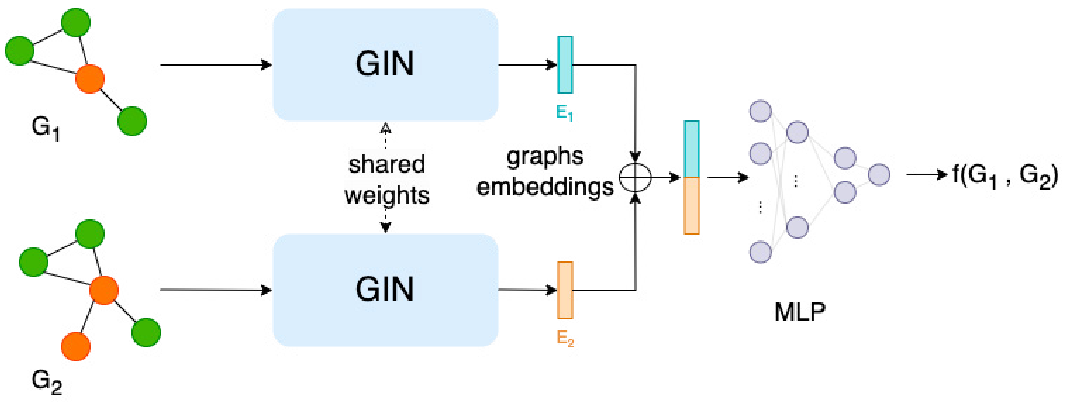

Fig. 3. Architecture of NeuraGED

3. Experimental Setup

Experiments have been performed on a Intel(R) Core(TM) i7-9800X CPU with 3.80GHz equipped with 64 GB RAM. The dataset consists of 120.000 pairs of graphs, 70% of which compose the training set, 10% the validation set, and 20% the test set. Once the number of message passing layers has been fixed to 3, equal to the maximum diameter of connected graphs in the dataset, the first step of the training procedure consists in a random search of the network hyperparameters. For the GIN state updating function, a MLP with only one hidden layer has been chosen. The number of neurons in the hidden layer depends on the node state dimension, varying between $2h$ and $3h$, h being the node state dimension. The activation function of the MLP module inside the GIN layers is fixed to *SeLU* and the global pooling method is always *sum*.

Since the MLP implementing the distance function f_2 takes in input the concatenated embeddings of the two graphs, its layout depends on the node state dimension h. While the input layer is always of dimension $2h$ and the output layer is always composed of one unit, the number of hidden layers change within 2 and 3 and their units within $2 \cdot h$, $10 \cdot h$, for 2 layers, and $2h$, $10h$, $10h$, for 3 layers. The activation function is *LeakyReLU* for all layers except the last one, which is equipped with *ReLU* activation. In Table 1 are reported the combinations of node state dimension, GIN hidden layers, pooling type, and number of hidden layer neurons for the MLP.

The training of the model is carried out using *MSE* as a loss function and the *Adam* optimizer, with a learning rate of 10^{-6}, a batch size of 512, and a number of epochs equal to 1000, with early stopping based on the validation loss.

To measure the performance of the network configuration with each chosen set of parameters, a 5–*fold cross validation* has been applied, with patience for *early stopping* equal to 15. Each training inside the cross validation has been performed changing the initialization of feature matrices of the graphs.

Table 1. Experimental setup for NeuraGED. The table summarizes the parameters used in the experiments. As it can be seen, values from the second and fourth columns depend on the values of the first column, i.e. GIN hidden layer and MLP hidden layer sizes are multiples of the node state dimension. The best setup is reported in bold.

State dimension	**GIN hidden layer**	**Pooling type**	***MLP* hidden layers**
64	128	sum	[128 , 640]
64	192	sum	[128 , 640]
128	**256**	**sum**	[**256** , **1280**]
128	384	sum	[256 , 1280]
128	256	sum	[256 , 1280, 1280]
128	384	sum	[256 , 1280, 1280]

4. Results and Discussion

Experiments show that the best model is the one having node state dimension of 128, GIN hidden layer size of 256, *sum* global pooling and MLP with two hidden layers of 256 and 1280 neurons.

In Table 2, the best results of the cross–validation are reported.

Table 2. Comparison of results obtained with different feature matrix initialization methods. For each method, the average MSE on the test set and its standard deviation are reported.

Feature matrix	**Average test MSE**	**Standard deviation**
ones	**0.1343**	**0.0754**
random uniform	1.9211	0.2822
random normal	14.4601	2.5295

As it can be appreciated from the results shown in Table 2, NeuraGED can identify with a low error a broad variety of graph structures. This behavior can be obtained with different methods of feature matrix initialization, indicating that the model is capable of filtering out (at least to a certain degree) this unrelated information, focusing only on the topology. However, the predictions obtained with feature matrices sampled from a distribution in the interval $[0, 1)$, referred in Table 2 to as *random uniform*, or a standard normal distribution, referred in Table 2 to as *random normal*, are not as accurate as those obtained in the case of feature matrices initialized with all the entries equal to 1. This indicates that the features introduce some noise in the GED estimation process, even if the GED is purely topological. Additionally, it is possible to observe that the sparsity of the random normal feature matrices can hinder the learning process, but this could be avoided by shifting the mean of the normal distribution.

5. Conclusion

In this paper, NeuraGED, a method for computing a soft–estimation of the Graph Edit Distance (GED) based on siamese Graph Neural Networks (GNNs) was presented. An exact computation of the GED is unfeasible because it constitutes a NP–hard problem and cannot be carried out for graphs larger than 10 nodes. The proposed model provides a reliable and computationally affordable estimation of the GED, with very low error rates, that can be applied also to graphs with more than 10 nodes. This method can be useful for several tasks on graph data that require an evaluation of a topological distance between two structures. For instance, it could be used to compute the distance between two molecular graphs, even when dealing with large molecules. This is potentially very useful when building graph generators, as the proposed method could be used for calculating a reliable, fast, informative, and permutation–invariant reconstruction loss. Such loss could substantially improve the training procedure of many graph generative models, which typically rely on less informative and non–permutation–invariant reconstruction loss functions.
Regarding future research, it would be interesting to add a term accounting for the dissimilarities between the feature vectors to the target estimation equation. For instance, this term could be based on the euclidean distance between

features. Of course, another possible development to pursue could be the addition of edge labels, consequently obtaining an estimation of the distances between the edge feature vectors. These latter two steps would extend the proposed computation to labeled graphs, making the distance much more informative in this domain. Moreover, it could be functional to add a normalized precomputed supervision, for instance by applying a logarithmic formulation. Finally, this promising model could be extended to produce a more precise approximation of the *GED*, thus becoming a plausible reconstruction loss function for deep generative models.

Acknowledgements

This study was co-funded by the European Union - Next Generation EU, in the context of The National Recovery and Resilience Plan - Investment 1.5 Ecosystems of Innovation, Project Tuscany Health Ecosystem (THE), Spoke 3 - Advanced technologies, methods and materials for human health and well-being. ECS00000017, CUP: B63C22000680007.

References

[1] Lim SH, Lee SM, Powers S, Shankar M, Imam N. Survey of approaches to generate realistic synthetic graphs. Oak ridge national laboratory. 2015.
[2] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The Graph Neural Network Model. IEEE Transactions on Neural Networks. 2009;20(1):61–80.
[3] You J, Ying R, Ren X, Hamilton W, Leskovec J. Graphrnn: Generating realistic graphs with deep auto-regressive models. In: International conference on machine learning. PMLR; 2018. p. 5708–5717.
[4] Bongini P, Bianchini M, Scarselli F. Molecular generative graph neural networks for drug discovery. Neurocomputing. 2021;450:242–252.
[5] Bongini P. Graph Neural Networks for Drug Discovery: An Integrated Decision Support Pipeline. In: 2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE). IEEE; 2023. p. 218–223.
[6] De Cao N, Kipf T. MolGAN: An implicit generative model for small molecular graphs. arXiv preprint arXiv:180511973. 2018.
[7] You J, Liu B, Ying Z, Pande V, Leskovec J. Graph convolutional policy network for goal-directed molecular graph generation. Advances in neural information processing systems. 2018;31.
[8] Kipf TN, Welling M. Variational graph auto-encoders. arXiv preprint arXiv:161107308. 2016.
[9] Simonovsky M, Komodakis N. Graphvae: Towards generation of small graphs using variational autoencoders. In: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27. Springer; 2018. p. 412–422.
[10] Jin W, Barzilay R, Jaakkola T. Junction tree variational autoencoder for molecular graph generation. In: International conference on machine learning. PMLR; 2018. p. 2323–2332.
[11] Wang H, Wang J, Wang J, Zhao M, Zhang W, Zhang F, et al. Graphgan: Graph representation learning with generative adversarial nets. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32; 2018. .
[12] Ma D, Yuan D, Huang M, Dong L. Vgc-gan: a multi-graph convolution adversarial network for stock price prediction. Expert Systems with Applications. 2024;236:121204.
[13] Chamberlain B, Rowbottom J, Gorinova MI, Bronstein M, Webb S, Rossi E. Grand: Graph neural diffusion. In: International Conference on Machine Learning. PMLR; 2021. p. 1407–1418.
[14] Cao H, Tan C, Gao Z, Xu Y, Chen G, Heng PA, et al. A survey on generative diffusion models. IEEE Transactions on Knowledge and Data Engineering. 2024.
[15] Shlens J. Notes on kullback-leibler divergence and likelihood. arXiv preprint arXiv:14042000. 2014.
[16] Bunke H, Allermann G. Inexact graph matching for structural pattern recognition. Pattern Recognition Letters. 1983;1(4):245–253.
[17] Bunke H. On a relation between graph edit distance and maximum common subgraph. Pattern recognition letters. 1997;18(8):689–694.
[18] Ranjan R, Grover S, Medya S, Chakaravarthy V, Sabharwal Y, Ranu S. Greed: A neural framework for learning graph distance functions. Advances in Neural Information Processing Systems. 2022;35:22518–22530.
[19] Gao X, Xiao B, Tao D, Li X. A survey of graph edit distance. Pattern Analysis and applications. 2010;13:113–129.
[20] Dory M, Parter M. Fault-tolerant labeling and compact routing schemes. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing; 2021. p. 445–455.
[21] Ibragimov R, Malek M, Guo J, Baumbach J. Gedevo: an evolutionary graph edit distance algorithm for biological network alignment. In: German conference on bioinformatics 2013. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik; 2013. .
[22] Deng S, Yu Y. Predicting Structural Similarity between Molecules Using Graph Neural Networks. In: 2022 10th International Conference on Bioinformatics and Computational Biology (ICBCB). IEEE; 2022. p. 78–84.
[23] Zeng Z, Tung AK, Wang J, Feng J, Zhou L. Comparing stars: On approximating graph edit distance. Proceedings of the VLDB Endowment. 2009;2(1):25–36.
[24] Bai Y, Ding H, Bian S, Chen T, Sun Y, Wang W. Simgnn: A neural network approach to fast graph similarity computation. In: Proceedings of the twelfth ACM international conference on web search and data mining; 2019. p. 384–392.

[25] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:160902907. 2016.
[26] Wang X, Ding X, Tung AK, Ying S, Jin H. An efficient graph indexing method. In: 2012 IEEE 28th International Conference on Data Engineering. IEEE; 2012. p. 210–221.
[27] Maas A, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies; 2011. p. 142–150.
[28] Bongini P, Pancino N, Scarselli F, Bianchini M. BioGNN: how graph neural networks can solve biological problems. In: Artificial Intelligence and Machine Learning for Healthcare: Vol. 1: Image and Data Analytics. Springer International Publishing Cham; 2022. p. 211–231.
[29] Su X, Xue S, Liu F, Wu J, Yang J, Zhou C, et al. A comprehensive survey on community detection with deep learning. IEEE Transactions on Neural Networks and Learning Systems. 2022.
[30] Bongini P, Messori E, Pancino N, Bianchini M. A Deep Learning Approach to the Prediction of Drug Side–Effects on Molecular Graphs. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2023.
[31] Pancino N, Perron Y, Bongini P, Scarselli F. Drug side effect prediction with deep learning molecular embedding in a graph-of-graphs domain. Mathematics. 2022;10(23):4550.
[32] Bongini P, Scarselli F, Bianchini M, Dimitri GM, Pancino N, Lio P. Modular multi–source prediction of drug side–effects with DruGNN. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2022;20(2):1211–1220.
[33] Pancino N, Rossi A, Ciano G, Giacomini G, Bonechi S, Andreini P, et al. Graph Neural Networks for the Prediction of Protein-Protein Interfaces. In: ESANN; 2020. p. 127–132.
[34] Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using a" siamese" time delay neural network. Advances in neural information processing systems. 1993;6.
[35] Serrano N, Bellogín A. Siamese neural networks in recommendation. Neural Computing and Applications. 2023;35(19):13941–13953.
[36] Chicco D. Siamese neural networks: An overview. Artificial neural networks. 2021:73–94.
[37] Irwin JJ, Shoichet BK. ZINC- a free database of commercially available compounds for virtual screening. Journal of chemical information and modeling. 2005;45(1):177–182.
[38] Kim J, Park S, Min D, Kim W. Comprehensive Survey of Recent Drug Discovery Using Deep Learning. International Journal of Molecular Sciences. 2021;22(18):9983.
[39] Xu X, Liu C, Feng Q, Yin H, Song L, Song D. Neural network-based graph embedding for cross-platform binary code similarity detection. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security; 2017. p. 363–376.
[40] Ma G, Ahmed NK, Willke TL, Yu PS. Deep graph similarity learning: A survey. Data Mining and Knowledge Discovery. 2021;35:688–725.
[41] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. Computational capabilities of graph neural networks. IEEE Transactions on Neural Networks. 2008;20(1):81–102.
[42] Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? arXiv preprint arXiv:181000826. 2018.
[43] Xu K, Li J, Zhang M, Du SS, Kawarabayashi Ki, Jegelka S. What can neural networks reason about? arXiv preprint arXiv:190513211. 2019.
[44] Li P, Leskovec J. The expressive power of graph neural networks. Graph Neural Networks: Foundations, Frontiers, and Applications. 2022:63–98.
[45] Serratosa F. Redefining the graph edit distance. SN Computer Science. 2021;2(6):438.