Received: 15 September 2022 | Revised: 9 November 2022 | Accepted: 30 November 2022

DOI: 10.1002/wcms.1651

OVERVIEW

WIREs COMPUTATIONAL MOLECULAR SCIENCE WILEY

Graph neural networks for conditional de novo drug design

Carlo Abate[1,2] | **Sergio Decherchi**[1] | **Andrea Cavalli**[1,2]

[1]Fondazione Istituto Italiano di Tecnologia, Genoa, Italy

[2]Università degli Studi di Bologna, Bologna, Italy

Correspondence
Sergio Decherchi, Fondazione Istituto Italiano di Tecnologia, Genoa, Italy.
Email: sergio.decherchi@iit.it

Edited by: Peter Schreiner, Editor-in-Chief

Abstract

Drug design is costly in terms of resources and time. Generative deep learning techniques are using increasing amounts of biochemical data and computing power to pave the way for a new generation of tools and methods for drug discovery and optimization. Although early methods used SMILES strings, more recent approaches use molecular graphs to naturally represent chemical entities. Graph neural networks (GNNs) are learning models that can natively process graphs. The use of GNNs in drug discovery is growing exponentially. GNNs for drug design are often coupled with conditioning techniques to steer the generation process towards desired chemical and biological properties. These conditioned graph-based generative models and frameworks hold promise for the routine application of GNNs in drug discovery.

This article is categorized under:

Data Science > Artificial Intelligence/Machine Learning

Data Science > Chemoinformatics

Data Science > Computer Algorithms and Programming

KEYWORDS

deep learning, drug discovery, generative models, graph neural networks

1 | INTRODUCTION

Drug discovery is expensive and challenging. The capitalized cost for a single drug can easily reach 2 billion dollars.[1] Failure can occur during many phases of drug development. Factors leading to failure include: poor solubility, poor blood–brain-barrier penetration, lack of efficacy, off-target activity, difficult or low yield synthesis, poor kinetics, high toxicity, and/or suboptimal metabolism (ADMET).[2–8] During drug design, it is therefore important to generate several molecules with a high probability of hitting a given target. Deep learning and generative methods can maximize this probability. Deep learning is a computational technique that, in principle, can deal with many aspects of in silico drug design. It therefore may contribute in future to a largely automated drug discovery process, from hypothesis generation to synthesis to high-throughput automated screening.[9,10]

The first deep learning approaches to drug discovery used SMILES strings[11] as input representation for molecules. This strategy quickly leveraged powerful existing sequence learning architectures and adapted them to drug discovery.[12–20] These architectures were originally designed for natural language processing tasks. They include chiefly recurrent neural networks (such as long short-term memory networks[21]) and attention-based networks (such as Transformers[22]). These frameworks were capable of generating realistic molecules, yet, the SMILES representation has several intrinsic limitations relative to graph-based approaches. These models must learn both the chemistry and the SMILES grammar, which means that they use representation power to learn rules that are not the true target of the learning process. If the learning is not completely correct, this can lead to structurally invalid SMILES strings. SMILES

strings have other limitations. First, they have canonical and noncanonical representations, which may lead to imprecise learning.[23] Second, small perturbations of a SMILES string (in terms of edit distance) can lead to big modifications of the corresponding molecular entity. The opposite is also true, with similar chemical entities potentially leading to very different SMILES strings.

The above limitations can be tackled by describing a molecule with its molecular graph. A graph-based model does not have to learn grammar rules, so the corresponding network does not waste representational resources on this auxiliary task. Moreover, subgraphs can be interpreted as molecular fragments. Chemical structural constraints, such as molecular validity, can be explicitly enforced on full molecules and fragments more easily than SMILES. SMILES substrings do not necessarily have a chemical interpretation and a validity check policy is necessary; SELFIES[24] representation aims to mitigate this last issue. This is because every grammatically correct SELFIES string is also a chemically valid molecule, yet they still require the network to learn a grammar. Despite these nice theoretical features, still SMILES representations have practically led to valuable results also in comparison with graphs approaches[25,26] rendering the research on this topic particularly intriguing. For these reasons, there is growing interest in using graph neural network (GNN) models. GNN models have been successfully applied to various tasks, including recommendation, computer vision, natural language processing,[27] and drug discovery.[28]

In this review, we describe and summarize the recent evolution of GNN-based generative models that perform conditional generation for drug discovery. We focus on the main conditioning techniques (see Section 3.4.2). We also provide a formalization to represent all the conditioning tasks in the literature in a single framework (see Section 3.4.1). The paper is structured as follows: in Section 2, we introduce the graph generation problem, GNNs, and the main learning frameworks. In Section 3, we describe the main aspects of the molecular generative models, focusing on the conditioning methods and other details of the generation process. Finally, in Section 4, we briefly discuss the future directions of GNN-based approaches.

2 | GNNs AND LEARNING FRAMEWORKS

2.1 | Graph generation problem

A graph is a pair $G = (V, E)$, where $V = \{v_1, ..., v_N\}$ is a set of N nodes and $E = \{e_1, ..., e_M\}$ is a set of M edges. Each edge $e = (v_i, v_j)$ connects two nodes v_i and v_j for the indices i and j. A graph is said to be directed if $(v_i, v_j) \neq (v_j, v_i)$. Otherwise, it is undirected. Unless stated otherwise, the graphs considered from now on are undirected. Nodes v_i and v_j are adjacent if $(v_i, v_j) \in E$. For a given graph G, the corresponding adjacency matrix is denoted as $A \in \{0, 1\}^{N \times N}$. The i,j-th element of A, indicated as A_{ij}, represents the connectivity between the nodes v_i and v_j. $A_{ij} = 1$ if v_i is adjacent to v_j, otherwise $A_{ij} = 0$. The set of all nodes that are adjacent to node v_i is called the neighborhood of v_i, and is denoted with $N(v_i)$. $|N(v_i)|$ is called the degree of v_i. Finally, we denote with $N(i)$ the set of indices of the nodes belonging to $N(v_i)$.

Due to their versatility in describing structured data, graphs are widely used in many fields, including chemistry, software engineering, and image processing.[29,30] Since graphs are data structures that naturally represent relational information in many domains, the ability to generate them also has many applications. For example, a generative model of molecular graphs can be used for drug design, while graph generation also plays a significant role in network science. Historically, several methods and algorithms have been designed to generate graphs with certain properties. Examples include the Erdős–Rényi model[31] to generate random graphs, that is, where two nodes have a constant probability of being connected; the Watts–Strogatz model[32] to generate random social nets, that is, graphs with high edge density; and the Barabási–Albert model[33] to generate scale-free graph, that is, graphs whose degree distribution follows a power law. These generative algorithms focus on different criteria to generate the adjacency matrix, namely structural properties. In this case, the nodes are indistinguishable from each other and so are the edges. In order to represent molecules, one needs a richer graph data structure, such as an annotated graph. In annotated graphs, nodes and edges can possess additional features. They can be summarized in two matrices $F^V \in \mathbb{R}^{n \times c}$, $F^E \in \mathbb{R}^{m \times d}$, where c and d are node and edge feature dimensions, respectively. We denote with F_i the feature vector related to node v_i, and with F_{ij} the feature vector related to edge (v_i, v_j). Using this structure, a molecule can be represented by an annotated graph (V, E, F^V, F^E), identifying nodes with atoms, edges with bonds, and including any other relevant information in the feature matrices (e.g., atom types, bond types, Cartesian coordinates, chemical properties). This is not the only possibility since molecular graphs can be also represented in other ways (e.g., edge lists instead of adjacency matrix) or bear different semantics (e.g., identifying nodes with bonds and edges with atoms). However, the majority of the works considered in

this review follow the above-mentioned representation. To generate a molecular graph, one must therefore generate its adjacency matrix and predict the feature matrices. Generating a random chemical graph is a simple task, whereas it becomes very challenging when one desires to generate a distribution of molecules equipped with specific chemical properties (e.g., activity) and within the possible $\sim 10^{33}$[33,34] drug-like compounds of the chemical space. With the rise of artificial intelligence and the increasing availability of biochemical data, research interest has moved from handcrafted algorithms to end-to-end architectures such as GNN models.

2.2 | Graph neural networks

The term "graph neural networks" usually encompasses the entire class of deep learning approaches that aim to process graphs.[27,35] In general, a typical GNN architecture is a sequence of graph processing modules[30]: if the structure of the graph is not modified (input vs. output), it is called a propagation module and its role is to propagate information between nodes following the graph topology. More specifically, each node v_i, together with its feature vector F_i, is equipped with a hidden representation h_i. Propagation modules collect information from the neighborhood of each node and use this information to update the node's hidden representation. If the structure of the graph is modified, it is called a pooling module instead. When the graph is large, a sampling module is used to determine the subset of nodes (in the neighborhood of the current one) from which to collect the information. By combining these modules, GNNs can solve graph-focused and node-focused tasks. A detailed description of the different kinds of basic modules can be found in the study of Zhou et al.[30] In the following section, we briefly describe the rather general setting of message-passing neural networks as the common background of most of the currently available propagation modules.

2.2.1 | Message-passing neural networks

Unlike other data structures, graphs do not have a specific node ordering for each neighborhood. This means that the functions that are implemented by graph processing modules must be permutation invariant. The message-passing neural network[36] (MPNN) is a general GNN framework that gives an abstract structure to most standard permutation invariant graph processing modules. For each node v_i, for each message-passing step l, an MPNN module updates its hidden representation h_i^l as follows*:

$$m_{ji}^{l+1} = M_l\left(F_i, F_j, F_{ij}, h_i^l, h_j^l\right), \tag{1}$$

$$m_i^{l+1} = A_l\left(m_{ji}^{l+1}, v_j \in N(v_i)\right), \tag{2}$$

$$h_i^{l+1} = U_l\left(F_i, h_i^l, m_i^{l+1}\right), \tag{3}$$

where F_{ij} is the feature vector related to edge (v_i, v_j), $M_l()$ is a message function, generating the messages passing from node v_j to node v_i, $A_l()$ is an aggregator function, which is permutation-invariant with respect to the neighbors of node v_i, and $U_l()$ is an update function that generates the new hidden vector. At the end of the process, the final hidden representations can be fed to node-wise output functions $O_i()$ or to a global graph-wise readout function $R()$, that will produce, respectively, the new node-level and graph-level feature vectors. It is important to remark that, to maximize the expressive power of GNNs, the aggregator functions $A_l()$ and the graph readout function $R()$ should be injective (i.e., they do not map different neighborhoods or different graphs to the same representation).[39]

Propagation modules can be divided in two groups: recurrent approaches and convolutional methods. The graph propagation modules that can be ascribed to the MPNN framework[28,30,37] include: the first proposed GNN (denoted as GNN to distinguish it from general models),[38] gated graph neural networks (GG-NNs),[40] graph attention networks (GATs),[41] and graph convolutional networks (GCNs).[42] GNN and GG-NN represent the recurrent approach, in which message propagation is seen as a dynamical system. In a recurrent propagation module, the symbol l also represents the current time step of the state dynamical system. The GNN message-passing scheme is as follows:

$$m_{ji}^{l+1} = \mathrm{MLP}\left(F_i, F_j, F_{ij}, h_j^l\right), \tag{4}$$

$$m_i^{l+1} = \sum_{j \in N(i)} m_{ji}^{l+1}, \tag{5}$$

$$h_i^{l+1} = m_i^{l+1}. \tag{6}$$

In this case, the operator $M_l()$ in Equation 1 is a multilayer perceptron that has the same weights as all the iterations of the message passing through this module. GNNs are based on an iterative process, which must converge to a fixed point. To tackle this issue, GG-NNs[40] stop the iterations after a certain number of steps, without guaranteeing convergence.

GCN and GAT represent the convolutional approach: as the name suggests, these modules seek to extend the concept of convolution to the graph domain. The key difference between recurrent and convolutional approaches is as follows: in recurrent approaches, each recurrent module has several message-passing steps that rely on the same network with the same weights. In convolutional approaches, each module is related to a single message-passing step. This means that two consecutive convolutional propagation steps rely on two independent sets of weights, achieving better performances in terms of scalability and prediction accuracy. Hence, the symbol l indicates the convolutional module (layer) here. The GCN message-passing scheme is as follows:

$$m_{ji}^{l+1} = \frac{1}{\sqrt{d_i d_j}} W^l h_j^l, \tag{7}$$

$$m_i^{l+1} = \sum_{j \in N(i) \cup \{i\}} m_{ji}^{l+1}, \tag{8}$$

$$h_i^{l+1} = \sigma\left(m_i^{l+1}\right), \tag{9}$$

where d_i and d_j are the degrees of node v_i and v_j, σ is an activation function, and W^l is the learnt weight matrix. GCN algorithm is a special version of the Cheby-Filter,[43] a spectral method taking advantage of truncated Chebyshev polynomials. In its original formulation the coefficient $1/\sqrt{d_i d_j}$ was used to scale only the messages coming from the neighbor nodes, while the hidden vector of the center node v_i was not scaled. This led to some numerical instabilities that are solved in Equation 7 in which now the scaling is done also for $i = j$.

The calculation of the message m_{ji} is generalized in GAT modules. In this case, $m_{ji}^{l+1} = \alpha_{ij} W^l h_j^l$, where $\alpha_{ij} = \alpha(v_i, v_j)$ are learnt attention weights that try to grasp the underlying relationships between each pair of nodes. Figure 1 illustrates an example of one propagation step.

GCNs have been widely improved and adapted to several types of graph subdomains, including those used to model molecular data. Some of the most relevant are relational GCNs (R-GCNs),[44] edge-conditioned GCNs (EC-GCNs),[45] and spatial GCNs (spatial-GCNs).[46] In R-GCNs and EC-GCNs, separate message-passing functions are learnt with respect to different edge types whereas in spatial-GCN each node is equipped with a vector of coordinates. Importantly, the molecular representation learning problem contributed to the early development of GCN modules, with molecular graph convolutions[47] and neural fingerprints[48] being promising models designed specifically for the molecular domain.

2.3 | Learning frameworks

Nowadays, GNNs are often the main building blocks of big generative models. In the past, they were mainly used to learn molecular representations, for example, property prediction.[49] In this review, we analyze molecular graph generative models that refer to some well-established deep learning approaches. These core building blocks are: variational autoencoders (VAEs), generative adversarial networks (GANs), CycleGAN, normalizing flows, score-based models, transformers, and reinforcement learning (RL). We summarize them below.

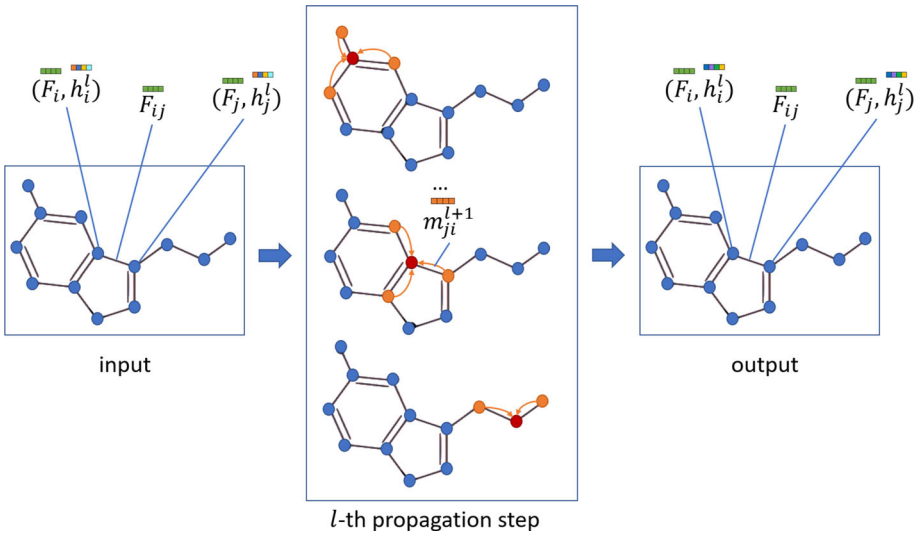

FIGURE 1 An example of message passing. Message propagation to a generic node (in red) from a neighborhood during one propagation step

2.3.1 | Variational autoencoders

VAEs[50] are one of the most common generative architectures. The VAE architecture comprises two subnets. The first subnet is an inference network. During training, it compresses the input examples into a latent easy-to-sample probability distribution (e.g., normal). During the learning phase, a latent sample vector is drawn from this distribution and given to a generator network, which tries to reconstruct the original input sample. The loss function is a variational lower bound of the log-likelihood of the data, which is the sum of the reconstruction loss (i.e., the distance between the input and the output) and the similarity loss (i.e., the KL-divergence between the latent distribution and a reference distribution, usually normal).

2.3.2 | Generative adversarial networks

GANs[51] comprise two subnets, a generator network and a discriminator. The generator produces artificial likely samples to fool the discriminator. The discriminator is fed with real and generated data, and must recognize if a given sample is original or artificial. The two networks are jointly trained in a minimax game, in which they try to improve themselves until the discriminator can no longer distinguish real from generated data.

2.3.3 | CycleGAN

The CycleGAN[52] technique learns a map between the elements of two datasets, without giving any paired examples to the model. The CycleGAN architecture comprises two GANs that are jointly trained. As input, the generators take samples from each dataset and must then generate samples that could belong to the other dataset's domain. The discriminators must recognize if the generated images belong to the desired domain or not. A cycle consistency loss is added to the two usual GAN losses to ensure the reversibility of each transformation, that is, $f_i(f_j(x_j)) = x_j$ and $f_j(f_i(x_i)) = x_i$, where i and j refers to the two domains.

2.3.4 | Normalizing flow

A normalizing flow[53] is a sequence of invertible and differentiable functions, which transform a simple probability distribution into a target one. Generation is then achieved by sampling points from the simple distribution and applying

the learnt transformation. Due to the discrete nature of atom types and bond types in the specific case of molecular generation, real-valued noise is added to convert them into continuous data in order to fit them into a flow-based architecture.[54]

2.3.5 | Score-based models

Instead of learning the data probability distribution directly, score-based models[55] learn its gradient with respect to the data. In particular, the data samples are transformed according to a predetermined stochastic differential equation (SDE, called diffusion process) into (typically) Gaussian noise. Thanks to SDE properties, learning such gradient is sufficient to allow reverting this process, similarly to what happens in denoising diffusion models,[56] as a coupled time reverse SDE can be associated to the original one. Generation is then achieved by integrating this time reverse SDE, starting from a random sample and then following the direction of the gradient through stochastic gradient Langevin dynamics[57] (or other SDE ensembles).

2.3.6 | Transformers

Transformers were introduced by Vaswani et al.[22] They have achieved state-of-the-art performances in several tasks involving sequential data. In contrast to recurrent neural networks, this encoder–decoder architecture can process the entire input sequence at once, allowing for massive parallelization. Transformers rely on a (multihead self)-attention mechanism[58] to model long-range interactions between the sequence elements.

2.3.7 | Reinforcement learning

In a RL framework, an agent learns how to make decisions in a complex environment. The decisions are chosen according to the output of a policy network. The parameters of this network are modified to optimize the negative or positive feedback received by the environment. When modeling a generation task, the policy network can be any generative model, and the action space can be arbitrarily complex. RL-based techniques are usually trained using policy gradient optimization algorithms,[59] a widely used family of algorithms for reward function optimization.

3 | GUIDED MOLECULAR GRAPH GENERATION

A molecular generator's first goal is to learn to build realistic molecules belonging to the dataset distribution (e.g., drug-like molecules, fragments, proteins). To learn this skill, the molecular generator should also explore the chemical space and guide the generation towards compounds that are not in the training set. The strategies adopted to learn the dataset distribution are strongly related to the way the generation process is modeled. We discuss this in Section 3.1. In Section 3.3, we discuss ways to enforce validity.

A molecular generator's second goal is to skew the distribution of the generated molecules towards structures that are optimized for a prescribed set of (numerical) properties; from the simply achievable quantitative estimate of drug-likeliness (QED) to the more cogent synthetic accessibility (SA) and octanol–water partition coefficient (logp).

This *conditioning* step can be achieved in different ways, depending on the modeling approach of the generation task and on the deep learning models used (see Section 3.4). This modeling step is crucial for the real-world application of these methodologies. Indeed, while learning to successfully sample from a complex distribution is an excellent result for machine learning, it is not enough for in silico drug discovery. Molecule hypothesis generation must be coupled with a biasing process to find the desired chemical entity. This is achieved through conditioning and is one of the most important medicinal chemistry aspects for the success of future generative models.

The most popular technique for the generation and conditioning setup is to create a dataset of molecular graphs $D_{\text{self}} = \{G_1, G_2, ..., G_M\}$ to be learnt, and to devise a self-supervised model. A less common approach is to model the generation of optimized molecules as a supervised task. In this case, the input dataset comprises pairs

$D_{\text{sup}} = \left\{ \left(G_1,\ \widehat{G}_1\right), ..., \left(G_M,\ \widehat{G}_M\right) \right\}$ and the model tries to directly learn a map from a known unoptimized input molecule G_i to a known optimized target molecule $\widehat{G}_i$.

The methodologies reviewed here are representative of the approaches in the recent literature. Their key differences are summarized in Table 1. The learning framework refers to the generative architecture used. The GNN model specifies the main graph-processing block in each architecture. The conditioning technique is used to optimize molecular properties. The validity constraint refers to the method used (if present) to enforce the generation of valid molecules. The generation process details how the graph is built and how the latent distribution is modeled. This can be carried out at different granularity levels, depending on the molecular dictionary (typically involving fragments or atoms) used. In the following sections, we discuss these aspects in detail.

3.1 | Generation process

In most cases, molecular graph generation architectures use a self-supervised block to learn the characteristics of a given dataset $D_{\text{self}} = \{G_1, G_2, ..., G_M\}$. The underlying assumption is that the dataset is sampled from an unknown distribution $p^*(G)$, which is modeled with an explicit distribution $p_\theta(G)$. During training, the network learns the parameters θ of a function $f_\theta(G)$ that approximates $p_\theta(G)$. After training, the generation is conducted by sampling from f_θ. The generation can be conducted in different ways (see Figure 2). In one-shot models, a molecule is produced in a single inference step. In sequential models, a molecule is built via several inference steps of the same model. Another approach is masked graph modeling, which was inspired by masked language models (e.g., BERT[92]) and which we will describe later.

3.1.1 | One-shot generation

To model p_θ, one can assume that $p_\theta(G) = p_\theta(G|\boldsymbol{z})$, where $\boldsymbol{z} = z_1, ..., z_n$ is a vector of latent variables whose prior distribution is known. The generation is then conducted by sampling the latent variables and generating the entire graph according to the sampled latent vectors. This one-shot strategy is used by the older models MolGAN[63] and GraphVAE,[62] which rely on a GAN and a VAE learning framework, respectively. In MolGAN, the GAN generative model is an MLP, which is used to directly sample the adjacency matrix of the molecular graph in a one-shot fashion. The discriminator network is based on R-GCN.[44] In GraphVAE, a molecule is modeled using a probabilistic graph, in which the existence of a node/edge is represented by a Bernoulli variable, while node and edge attributes are multinomial variables. The encoder is an edge-conditioned graph convolutional network,[45] and the decoder is a deterministic MLP. Scalability is the key limitation here because the network is trained on a scoring algorithm (i.e., graph matching) that is computationally tractable with small graphs only.

Another VAE-based latent variable approach is MPGVAE.[82] Here, the assumption for the bond/atoms joint latent distribution is that:

$$p_\theta(G|\mathbf{z}) = p_\theta(V, E \mid \mathbf{z}) = \prod_{v \in V} p_\theta(F_v|\mathbf{z}) \prod_{e \in E} p_\theta(F_e|\mathbf{z}), \tag{10}$$

where the feature vectors of the nodes and edges are categorical distributions over atom and bond types. The generation has three steps. First, an initial graph structure is generated from latent variables in a one-shot fashion. Second, an MPNN[36] iteratively refines node and edge representations. Third, the final molecular graph is sampled from the edge/node representations (i.e., distributions).

Other ways to model the latent distribution are to assume:

$$p_\theta(G|\mathbf{z}) = p_\theta(V, E \mid \mathbf{z}) = p_\theta(V|E, \mathbf{z_V}) p_\theta(E|\mathbf{z_E}) \tag{11}$$

or

$$p_\theta(G|\mathbf{z}) = p_\theta(V, E \mid \mathbf{z}) = p_\theta(E|V, \mathbf{z_E}) p_\theta(V|\mathbf{z_V}), \tag{12}$$

TABLE 1 Literature methods and their main features: core learning framework, GNN model, the molecular properties conditioning technique, the validity constraint technique, the generation process of new entities, the granularity level (e.g., atomic, fragment), and the code availability

Ref	Learning framework	GNN model	Conditioning	Validity constraint	Generation process	Granularity level	Code availability
60	VAE	GG-NN	Gradient ascent	Valency masking	Sequential	Atom wise	https://github.com/Microsoft/constrained-graph-variational-autoencoder
61	VAE	MPNN	Gradient ascent Bayesian optimization	Learnt	Sequential	Fragment based	https://github.com/wengong-jin/icml18-jtnn
62	VAE	EC-GCN	Conditional vector	Learnt	One-shot	—	https://github.com/JiaxuanYou/graph-generation/tree/master/baselines/graphvae
63	GAN	R-GCN	RL	Learnt	One-shot	—	https://github.com/nicola-decao/MolGAN
64	Hierarchical	R-GCN	Conditional vector	Learnt	Sequential	Atom wise	https://github.com/kevinid/molecule_generator
65	Transformer	3D-GCN	Supervised	Learnt	Sequential (SMILES)	—	https://github.com/prokia/deepHops
66	GAN	GCN	RL	Valency-based rejection sampling	Sequential	Hybrid	https://github.com/bowenliu16/rl_graph_generation
67	Hierarchical	GraphRNN	RL	Valency-based rejection sampling	Sequential	Atom wise	—
68	Normalizing Flow	R-GCN	RL	Valency-based rejection sampling	Sequential	Atom wise	https://github.com/DeepGraphLearning/GraphAF
69	VAE + GAN	MPNN	Conditional vector	Learnt	Sequential	Fragment based	—
70	VAE + Attention	MPNN	Conditional vector	Learnt	Sequential	Fragment based	https://github.com/wengong-jin/hgraph2graph/
71	Hierarchical	GGNN	RL	Learnt	Sequential	Atom wise	https://github.com/olsson-group/RL-GraphINVENT
72	Hierarchical	MPNN	Conditional vector	Learnt	Sequential	Atom wise	https://github.com/jaechanglim/GGM
73	GAN	GCN	RL	Valency-based rejection sampling	Sequential	Hybrid	https://github.com/dbkgroup/prop_gen
74	VAE	MPNN	Bayesian optimization	Valency masking	Sequential	Atom wise	https://github.com/Networks-Learning/nevae
75	Hierarchical	MPNN	Conditional vector	Learnt	Masked	—	https://github.com/nyu-dl/dl4chem-mgm
76	VAE	MPNN	Conditional vector	RL	One-shot	—	https://github.com/seokhokang/graphvae_approx/
77	Hierarchical	GCN	Conditional vector	Learnt	Sequential	Atom-wise	https://github.com/Laboratoire-de-Chemoinformatique/hyfactor
78	VAE	GCN	Gradient ascent	Beam search	One-shot	—	—
79	Normalizing Flow	R-GCN	Gradient ascent	Post hoc validity correction	One-shot	—	https://github.com/calvin-zcx/moflow

TABLE 1 (Continued)

Ref	Learning framework	GNN model	Conditioning	Validity constraint	Generation process	Granularity level	Code availability
80	VAE + Normalizing Flow	R-GCN	Gradient ascent	Post hoc validity correction	One-shot	—	https://github.com/chshm/GF-VAE
81	Normalizing Flow	R-GCN	Gradient ascent	Learnt	One-shot	—	https://github.com/hlzhang109/PyTorch-GraphNVP
82	VAE	GAT	Conditional vector	Learnt	One-shot	—	—
83	Normalizing Flow	GAT	Gradient ascent	Learnt	Sequential	Atom-wise	—
84	Normalizing Flow	R-GCN	Supervised	Learnt	One-shot	—	—
85	CycleGAN	JT-VAE	Supervised	Learnt	One-shot	—	https://github.com/ardigen/mol-cycle-gan
86	Hierarchical	GGNN	Gradient ascent	Learnt	Reaction based	—	https://github.com/john-bradshaw/molecule-chef
87	Hierarchical	MPNN	Supervised	Learnt	Sequential (SMILES)	—	—
88	Hierarchical	MPNN	Supervised	SELFIES	Sequential (SMILES)	—	https://github.com/shiwentao00/Pocket2Drug
89	Hierarchical	MPNN	Markov chain	Learnt	Sequential	Fragment based	https://github.com/yutxie/mars
90	VAE	GGNN	Structural embedding	Valency masking	Sequential	Atom wise	https://github.com/oxpig/DeLinker
91	VAE	MPNN	Structural embedding	Learnt	Sequential	Atom wise	https://github.com/wengong-jin/multiobj-rationale

Abbreviations: EC-GCN, edge-conditioned graph convolutional network; GAN, generative adversarial network; GAT, graph attention network; GCN, graph convolutional network; GG-NN, gated graph neural network; GNN, graph neural network; MPNN, message-passing neural network; R-GCN, relational graph convolutional network; RL, reinforcement learning; VAE, variational autoencoder.

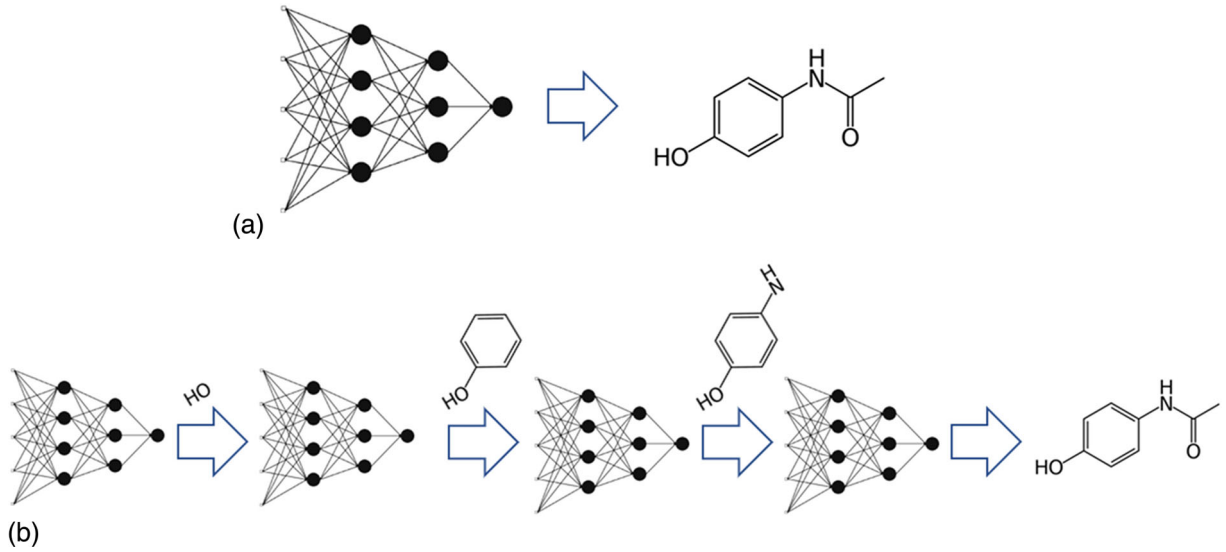

FIGURE 2 (a) One-shot generation. (b) (Fragment-based) sequential generation.

where $\boldsymbol{z_V}$ and $\boldsymbol{z_E}$ are the latent variables. This approach introduces a dependency between the bonds and the atoms. Here, the models first focus on one of the two sets and then use that information to complete the generation. In the study of Bresson and Laurent,[78] for example, the VAE decoder's first step is to generate a molecular formula in a "bag-of-atoms" representation. Then, a fully connected graph with the generated set of atoms is fed into a GCN-based network to properly learn the bond structure. In the study of Zang and Wang,[79] the first step is to generate the adjacency matrix, which is then fed into the atom feature generator together with the latent variables of the nodes.

Finally, in the study of Jo et al.,[93] graph generation is achieved through a score-based framework. In this case an SDE models a diffusion process in time $t \in [0, T]$ from the data distribution to noise, according to:

$$dG_t = f_t(G_t)dt + g_t(G_t)dw, \;\; G_0 \sim p^*, \tag{13}$$

where f_t is the drift coefficient, g_t is the diffusion coefficient and w is a standard Wiener process. As anticipated before, one can associate to this SDE a time reverse SDE of the form:

$$dG_t = \left[f_t(G_t) - g_t^2 \nabla_{G_t} \log p_t(G_t)\right] d\tilde{t} + g_t d\tilde{w}. \tag{14}$$

When applying this method to graphs a possibility[93] is to separate the diffusion over the features vectors from the diffusion over the adjacency matrix, thus instantiating a system of two coupled SDEs:

$$\begin{cases} dF_t = \left[f_{1,t}(F_t) - g_{1,t}^2 \nabla_{F_t} \log p_t(F_t,\; A_t)\right] d\tilde{t} + g_{1,t} d\tilde{w_1} \\ dA_t = \left[f_{2,t}(A_t) - g_{2,t}^2 \nabla_{A_t} \log p_t(F_t,\; A_t)\right] d\tilde{t} + g_{2,t} d\tilde{w_2}. \end{cases} \tag{15}$$

Here two GNNs learn the gradients $\nabla_{F_t} \log p_t(F_t, A_t)$ and $\nabla_{A_t} \log p_t(F_t, A_t)$; new samples are generated by replacing them in Equations 15 and using standard SDE integrators.

3.1.2 | Sequential generation

One limitation of these just described methods is that they require a maximum size of the output molecule. To address this problem, the generation can be modeled as a sequence of operations, called graph transition actions. A prescribed set of actions is associated with a given neural model. A typical action set is: atom (vertex) and bond (edge) addition and bond (edge) removal.[64,94] Other action sets are possible, such as when the vertex represents a molecular fragment.

Although some sequential models still require an upper bound to the number of nodes of the generated molecule,[60] this approach is more flexible than one-shot generation.

JT-VAE[61] was one of the first sequential approaches. Here, each molecule is assembled by linking chemical fragments from a predefined dictionary, which is derived in a chemically principled way from the training set but without a learning process. More specifically, a VAE-based model generates a junction tree for each molecule. The junction tree is the basic tree-structured scaffold representing the connections among nodes, where each node represents a set of candidate fragments. Here, the action set for the junction tree is the sequential addition of nodes maintaining the tree topology. To assemble the final molecule, a single fragment is chosen for each node, leading to the selection within the nodes.

In contrast to JT-VAE, the most common way to model sequential generation is to decompose $p_\theta(G)$ into the product of a sequence of conditional distributions $p_\theta(a_t \mid G_t, ..., G_0)$, where $a_t \in T(G_t, ..., G_0)$ is the action set available on graph G_t at step t. In practice, each graph G is transformed into a sequence of operations S_G, and the learning framework must learn how to replicate these sequences in an autoregressive fashion. Since the number of sequences for each graph can increase rapidly, different strategies are adopted to address this issue. To reduce the number of generative sequences, a graph node ordering must be defined a priori. Typical orderings are based on breadth-first search (BFS) and depth-first search (DFS) graph visit algorithms, although any custom ordering can be used. Finding a canonical ordering for molecular graphs is an old problem.[95] For the generative task, there is currently no theoretical grounded ordering. The most effective approach is architecture-dependent and chiefly determined by empirical evidence. Here, we detail non-Markovian and Markovian strategies. For Markovian strategies, one restricts T to depend on the latest graph only, namely $T = T(G_t)$.

Non-Markovian methods

Some examples of the non-Markovian approach are mentioned by Li et al.[64] and Popova et al.[67]; these can be ascribed to a hierarchical neural network framework.[96] This methodology involves splitting a net into subnets each focused on a specific task.

In the study of Li et al.,[64] a stack of R-GCN[44] blocks with a "BN-ReLU-Conv" structure[97] extracts the representation. An MLP then estimates the probability distribution of all possible actions. A variant of this architecture includes a recurrent unit, in which the molecule representation is stored and updated during the generation. Here, a DFS graph visit is used, enhanced by the additional move of going to a random node and thus generating several building paths for the same molecule. Due to the non-uniqueness of the sequence, a sampling strategy is also adopted.

In the GraphRNN[98] framework, a BFS ordering is used together with classical RNN networks to directly generate the adjacency matrix. This approach was extended to molecular graphs,[67] in which external MLPs predict node and edge labels. Strictly speaking, this model is never directly aware of a graph topology and does not use GNNs, but is still relevant.

Markov-process-based methods

To reduce the space of possible sequences, one can model the graph generation as a Markov process, assuming $p_\theta(a_t \mid G_t, ..., G_0) = p_\theta(a_t \mid G_t)$. This is reasonable because the generation history that led to that specific graph should not influence the subsequent decision. Some examples include GCPN[66] (a GAN-based model that will be discussed later), MG^2N^2,[99] and GraphAF.[68]

MG^2N^2 is based on three in a hierarchical fashion. Each network focuses on a specific decision. The first network is a classifier that decides whether to add a new node to the graph and whether this new node is bound to the node added in the previous step. If the decision is positive, the second network decides its edge label, and the third network determines the existence of other edges connecting it to other preexisting nodes.

GraphAF is based on normalizing flows.[53] It adapts to the graph domain a variant of normalizing flows called autoregressive flows.[100] These scale linearly with respect to the length of the sequence to be learnt. Inside the framework, an R-GCN calculates the embedding for the current graph for each step. Then, two separate MLPs learn the parameters of the distributions from which the next node and its edges are sampled.

3.1.3 | Masked graph modeling

This approach in the study of Mahmood et al.[75] uses graph masking (subgraph selection) for the generation. Instead of learning $p(G)$, the MPNN-based model tries to learn $p(\eta|G_{\setminus\eta})$, where η is a subset of G and $G_{\setminus\eta}$ is G with that subset masked out. After training, sampling is conducted starting from an initial graph. For each generation step, a part of the

graph is masked and resampled according to the learned conditional distribution. To the best our knowledge, Mahmood et al.'s[75] study is the only paper that uses this approach.

3.2 | Granularity level

The generation can be modeled in different ways depending on the granularity of the representation. In atom-based methods, the molecule is generated with one atom per step. With fragment-based methods, the molecule is generated by adding a small fragment (usually up to six atoms) from a predefined dictionary. The advantage of atom-based methods is flexibility, since they enable the exploration of the entire chemical space. In contrast, relying on a predefined set of fragments helps the validity (since fragments are valid by definition), but prevents a full chemical exploration. There is no a priori best choice for building the dictionary. In the study of Jin et al.,[61] for example, the fragment dictionary is learnt from the training set, so there is no guarantee that each molecule in the test set can be perfectly reconstructed. The fragment-based approach by Jin et al.[61] has a 100% validity score without having to hardwire any explicit chemical constraint. The authors extended this approach in another study of Jin et al.,[70] building a dictionary of bigger subgraphs (structural motifs) and achieving a higher reconstruction accuracy. In the study of You et al.,[66] the generative framework is instead adaptable to any kind of dictionary and can be potentially used with fragments, but generating molecules in an atom-based fashion leads to better results.

3.3 | Validity constraints enforcement

Regardless of the generation approach, the validity of the resulting molecule is also important. One advantage of molecular graph representations is that one can explicitly enforce this constraint on the generated samples. Although the model can learn chemical validity from data,[61–64,69,70,75,77,85–87] there are also several strategies for structurally imposing these constraints on the network training.

The most common strategies for obtaining valid molecules can be grouped into: (i) methods that take proper actions during generation; (ii) penalty methods that minimize the probability of generating invalid molecules; and (iii) ex post molecular correction methods.

The first group comprises methods that model the molecular generation as a sequence of operations. Here, one uses valency checks to avoid possibly wrong actions. If a given atom type already has the maximum number of bonds, then adding another bond is prevented. If some atoms have missing bonds after generation, the necessary number of hydrogen atoms is added. This is done in CGVAE.[60] In Rigoni et al.'s study,101 the authors adapted the CGVAE decoder to use valency histograms as additional inputs to guide the generation. In RL-based methods,[66–68] valency checks are used as intermediate feedback signals during training, and a validity-constrained sampling is conducting during inference. The generative model can thus only take valid actions, producing 100% valid molecules. Notably, these techniques allow full control of the model's tolerance for invalid actions. Intermediate invalid steps can be allowed to increase the model's exploration abilities. Alternatively, this can be forbidden by defining state transition dynamics, in which each possible action can lead to valid substructures, as shown by Kearnes et al.[102] Similarly, during the edge construction phase in the study of Bresson and Laurent,[78] only edges that do not break valency rules are added to the molecule.

The second group comprises methods that trade the structural difficulty of making correct moves for the relative ease of optimization. For instance, the Regularized Graph VAE[103] uses a penalty technique[104] to enforce constraints. The original constrained problem is transformed into an unconstrained problem, where the constraint violation is penalized via a tunable hyperparameter in the newly defined cost function. Although the approach does not always generate valid molecules, it is easy to implement and tries to approximately enforce a constraint as a restraint. This approach was also used by Pölsterl and Wachinger.[105] Another penalty-based technique was used by Kwon et al.,[76] where validity is encouraged through RL. More specifically, a new component is added to the GraphVAE architecture to assign a reward to valid samples, and the GraphVAE decoder is used as a policy network.

The third group comprises methods with an ex post fixing of the generated molecule. This method is not structurally embedded in the generation and may lead to several undesirable generated molecules. However, it is largely usable. The generated graph is modified according to valency rules in order to preserve a maximal subgraph belonging to the

original output. This ex post validity correction step is found in flow-based models mentioned by Zang and Wang[79] and Ma and Zhang.[80]

3.4 | Conditioning

Conditioning the graph generation is a powerful way to skew the learnt distribution towards a desired bias. Conditioning can be used for several tasks. It is implemented with specific conditioning methods, ranging from simple supervised algorithms to elaborated RL methods. Here, we describe in detail the tasks for which conditioning is used and then the associated techniques.

3.4.1 | Conditioning tasks

The literature contains several conditioning tasks. All these tasks have a common conditional framework. Namely, any conditional task can be ascribed to one or a subset of the following quadruplet:

$$(\underbrace{\mathcal{R}}_{1)}, \underbrace{\boldsymbol{p} \gtrless \boldsymbol{p_0}}_{2)}, \underbrace{\boldsymbol{p} = \boldsymbol{p_0}}_{3)}, \underbrace{\min(\boldsymbol{p})}_{4)}).$$

Here, (1) is a set of additional entities, which are given as input to the model. They can be drugs, proteins, or any additional information (chiefly biological information) to which the model must refer during generation; (2) is a set of *constraints*, where $\boldsymbol{p}$ is a set of numerical properties and $\boldsymbol{p_0}$ is the corresponding vector of thresholds; (3) encodes numerical equality between desired and actual properties, namely $\boldsymbol{p_0}$ and $\boldsymbol{p}$. In detail, $\boldsymbol{p}$ could be any a set of numerical and categorical attributes. Finally, (4) encodes a set of desired optimal values. Below, we describe the most common conditioning tasks, ascribing them to specific instances of the proposed framework.

Property optimization
This task involves generating new molecules to optimize the value of one or more numerical properties. Typical properties include: (i) QED (introduced by Bickerton et al.[106])[60,63,66–71,74,79–81]; (ii) SA score (SAS, introduced by Ertl and Schuffenhauer[107])[63,64]; (iii) penalized logp[61,66–68,70,74,78,80,83]; and (iv) biological activity.[69–73] This task can be ascribed to (4).

Property targeting
Instead of optimizing a given property, one can give the model a target value for the output molecule. With property targeting, the generation is guided toward molecules with a specified set of attributes, which can be numerical or categorical values for molecular properties. For example, Simonovsky and Komodakis[62] and Flam-Shepherd et al.[82] provide the desired histogram of atom types of the output molecule, whereas You et al.,[66] Lim et al.,[72] Mahmood et al.,[75] Kwon et al.[76] condition the generation toward molecules with a specific weight. Here, the task can be ascribed to (3).

Constrained property optimization
This task involves perturbing a molecule while optimizing one or more numerical properties. More formally, the generator is fed with a molecule G, and the goal is to maintain the similarity $\mathrm{sym}\left(G, \widehat{G}\right) \geq \delta$, where $\widehat{G}$ is the generated molecule and δ is a predefined similarity threshold. The most common choice for the property to be optimized is the penalized logp.[61,66–68,70,77–80,83] Here, the task is a combination of (1) and (2).

Scaffold hopping
In scaffold hopping, the goal is to minimize a 2D similarity and maximize a 3D similarity with respect to a given drug.[65,87,90] Here, the task is a combination of (1) and (3). Below, we describe some representative techniques for conditioning the models for these common tasks.

3.4.2 | Conditioning methods

Conditioning techniques can be divided into two groups: the generation can be guided during training or ex post. For training time conditioning, the model directly learns the latent distribution specific to the optimization task. For ex post conditioning, the model learns how to *navigate* a previously learnt generic molecular space in order to reach the region containing the (locally) optimized molecular features of interest. In principle training time guidance should generate a better manifold, yet a systematic comparison on this topic in literature is not available to authors knowledge.

Training time conditioning

The most common way to condition the molecular generation is to use conditional vectors[62,64,69,70,72,75–77]: instead of learning $p(G)$ (which can itself be modeled, as discussed in Section 3.1), the framework learns $p(G \mid \boldsymbol{c})$, where $\boldsymbol{c}$ is a vector containing the values of the properties of interest. More specifically, for each molecular graph G in the dataset, the ground-truth value for the vector $\boldsymbol{c}$ is calculated. During training, the model is fed with input couples $(G, \boldsymbol{c})$. During inference, the desired values for c are given to the model to guide the generation. The implementation depends on the specific architecture.

In a similar procedure mentioned by Imrie et al.[90] and Jin et al.,[91] the framework learns $p(G \mid S)$ where S is a substructure that the final molecule must contain. In practice, this involves progressively extending S by adding new atoms and bonds. In Imrie et al.'s study,[90] this strategy is used to generate linkers between two molecular structures in order to create a final bigger molecule that incorporates both. In Jin et al.'s study,[91] this strategy is used for multi-objective conditioning: each condition is translated into a *rationale*, and the union of the required rationales is used as starting graph for the generated molecule.

Another approach used in VAE-based methods[60,61,78] is conditioning through gradient ascent. The framework is extended with an additional regressor that predicts, from each point of the latent space, the value of the property to be optimized. The VAE and the regressor are jointly trained in order to model and regularize the latent space. During inference, the latent representation is calculated (i.e., the latent variables are sampled), optimized through gradient ascent with respect to the property predictor model, then decoded from the local optimum (see Figure 3).

RL approaches are interesting for conditional (and task-oriented) generation. RL is particularly suitable for molecular generation because one can estimate molecular properties with external tools, transform these estimates into feedback for the model, and combine this with RL algorithms to deal with discrete structures. In the study of De Cao and Kipf,[63] for example, the RL environment comprises two networks, both based on R-GCNs[44]: the GAN discriminator network and a reward network, assigning a score to each generated molecule, depending on the property to be optimized. First, the generative model is trained as part of a GAN only. Then, it is trained with an objective function combining the GAN loss and the RL loss. Here, the generation is one-shot, with a single action generating an entire molecule. Thus, the full potential of an RL approach is not exploited. A more detailed modeling is conducted in GCPN.[66] The authors modeled molecular generation as a Markov process, in which the generative model becomes a policy network that iteratively adds fragments to the molecule. For each step, a GCN-based network extracts a representation of the partially generated molecule, then four MLP models decide the next action. Modeling the generation as a sequence of decisions allows one to define short-term (based on the single action) and long-term (based on the final result) rewards. Here, the short-term rewards are a combination of validity rewards (see Section 3.3) and adversarial rewards, obtained following the GAN framework approach. In contrast, long-term rewards are domain-specific and are used to condition the generation toward the desired property. A similar approach was used by Popova et al.,[67] Shi et al.,[68] Atance et al.,[71] Khemchandani et al.[73]

Finally, some works model the property optimization task as a supervised learning problem. Here, the input dataset is made of pairs $D_{\text{sup}} = \left\{ \left(G_1, \widehat{G_1}\right), ..., \left(G_N, \widehat{G_N}\right) \right\}$, and the model is a map from an input to an output graph. This approach was used by Zheng etal.[65] and Maziarka et al.[85] The work of Zheng et al.[65] is a transformer-based model[22] used to perform the scaffold hopping task. Here, both SMILES and 3D coordinate graph representations are used as input formats to map the molecule into one of its "hopped" output versions in SMILES form.

In contrast, Maziarka et al.[85] extend the CycleGAN[52] paradigm to molecular graphs. The model learns to map points belonging to the JT-VAE latent space, sending each representation to its improved version in terms of the chosen properties.

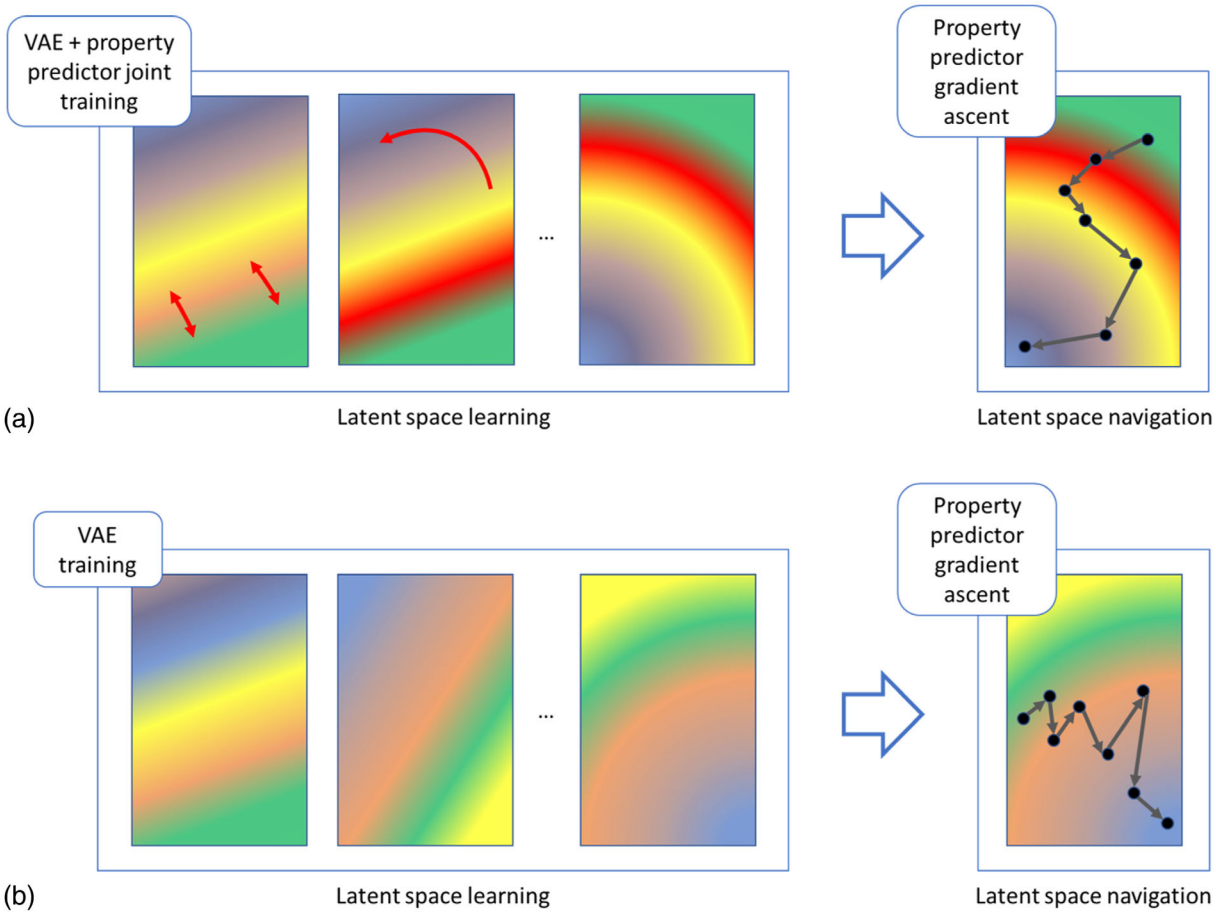

FIGURE 3 Properties of interest can modify the latent space at learning time (a). Red arrows encode the *force* of the properties predictors in optimizing the latent space during training. Alternatively properties predictors can be used just to navigate the latent space a posteriori via gradient ascent (b), this time the distortion force is absent.

Ex post conditioning

In contrast to the "training time" version described above, the regressor here is not jointly trained with the variational autoencoder. Instead, the VAE is trained independently in the first phase and a latent distribution of the dataset is learnt.[79–81,86] Then, the regressor is trained on the fixed latent space to predict the property of interest. Finally, the latent space is navigated through gradient ascent (see Figure 3).

When it is computationally demanding to evaluate the scoring function, one can use the Bayesian optimization (BO) technique.[108] The BO technique approximates the property scoring function by evaluating it on a bag of points. More specifically, the BO algorithm is an iterative loop. For each iteration, (i) a surrogate function tries to approximate the function using its values on the bag of points; and (ii) an acquisition function is used to choose which points to add to the bag. This is repeated until a stop criterion is met. In the study of Jin et al.[61] and Samanta et al.,[74] the surrogate function is a sparse Gaussian process[109] trained on the latent space, and the acquisition function is the expected improvement heuristics.[110]

4 | FUTURE DIRECTIONS AND CONCLUSIONS

Property-optimized molecular graph generation is a complex task. Despite the growing number of algorithms, many possibilities have not yet been explored.

Several recent GNN-based methods[99,102,103,105,111–113] offer interesting insights about learning molecular manifolds. These works offer a viewpoint, which can be enhanced by the above-mentioned conditioning techniques. Jiang et al.[114] systematically compared traditional descriptor-based methods and GNN-based ones for property prediction. For a single

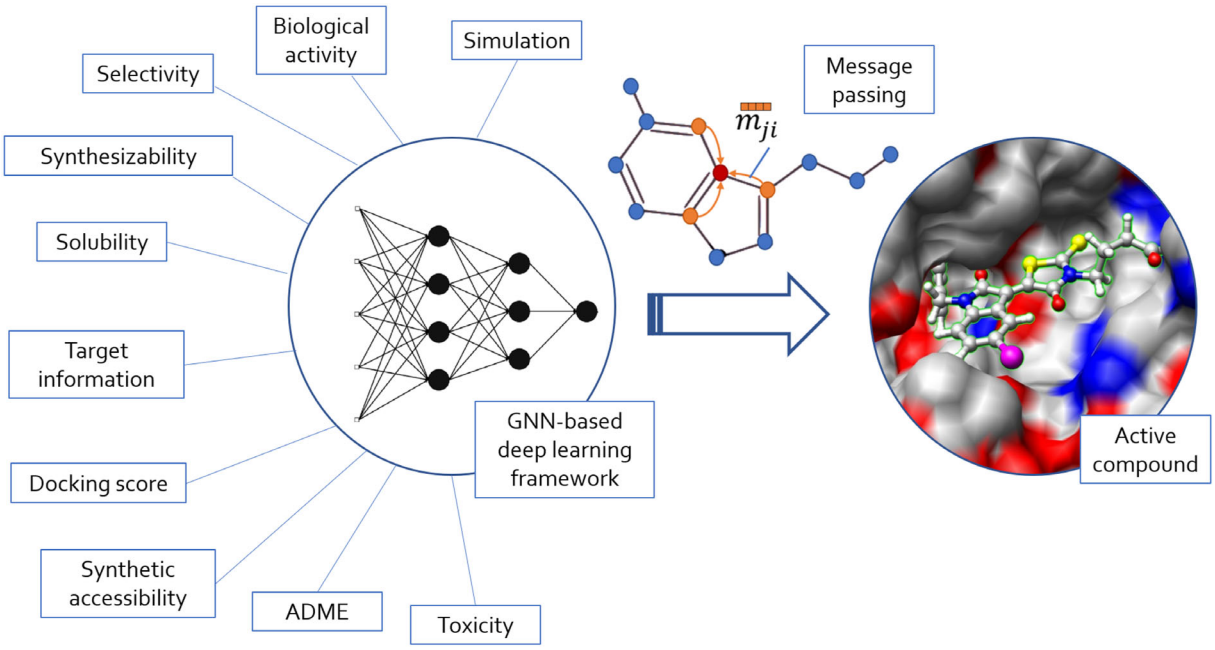

FIGURE 4 An ideal model able to design a drug candidate should take into consideration all the available in-silico information (simulative, machine learning predictors) and experimental data. This can be achieved via conditioning and context awareness.

objective function, there was no clear winner, but in multi-objective property prediction tasks GNNs achieved significantly higher performances. As GNN are effective in obtaining multi-task oriented representations one can also employ these representations to power purely RL-based methods that do not rely on a preexisting dataset or GNNs.[115–118]

Graph-based models have got their own limitations: on the algorithmic side, even simple structures such as decalin and bicyclopentyl are not distinguishable by naive message passing neural networks.[119] For this reason, some recent MPNNs do not propagate the information following the original molecular graph connectivity but instead allow for different, more complex interactions.[120–122] Also, 2D molecular graphs do not carry with themselves information about the conformational landscape of a molecule and its dynamics. One possibility is to represent molecules as 3D point clouds. Some efforts have been made to design generative models able to condition the generation according to 3D scores or directly output 3D molecular graphs. In this case, since many of the molecular properties are invariant with respect to rotations and translations, the same should hold for the underlying models which try to learn and optimize them.[123] Such models also include GNNs[124] or take into account physics,[125] but their performances are still similar to 2D ones.[126] In general, whichever is the underlying representation (2D molecular graph or 3D point cloud), there is still much space in terms of scalability improvements for these models.[127] Generating molecules sequentially might lead to a potential bias related to the chosen ordering of the atoms, while one-shot generative models are mainly available for small molecules.[127]

Most generative models described in this review try to produce molecules that are likely good drug candidates (within the limits of each framework's generative power). Nevertheless, although the potential drug-likeness of the generated samples can be increased by learning the dataset's latent distribution and optimizing certain molecular properties, it makes also sense to provide the generator with additional information (or context) and thus to fully exploit the learning capacity of modern deep learning frameworks. Ideally one should take advantage of all the experimental and in-silico available information (both physics-based simulation[128] and machine learning predictors) (Figure 4) toward gray box modeling.[129] For real-world applicability to drug discovery, one must therefore be somewhat "target-aware." Thus, instead of the input graphs G_1, ..., G_M, the model is given input couples (G_1, T_1), ...,(G_M, T_M), where T_i is the molecular (e.g., protein) target to which the molecule G_i binds. Alternatively, the built model is given a single target ad hoc. In the work of Zheng et al.,[65] a pretrained protein embedder adds the target information to the molecular feature vector to improve the performance of the scaffold hopping architecture. In Tan et al.'s study,[84] a normalizing flow-based architecture instead uses the target information to condition the generation of a new molecule with an optimized drug-target interaction. In the study of Shi et al.,[88] a supervised encoder-decoder model maps each pocket of the dataset to a suitable molecule. Here, GNNs produce the embedding of the binding pocket used by the framework. This is another

demonstration of the versatility of these networks and graph representations, which can be used to describe both drug-like molecules and targets. We believe that target awareness (which can be achieved via conditioning) will be critical to achieving maturity for this class of methods. The synthesizability of the molecules[86,130–132] is another kind of contextual information that can guide the generation. In particular, one might want to determine both the final molecule and a feasible route to its synthesis. An interesting proposal is MoleculeChef,[86] a GNN-based framework in which the network does not directly sample the final molecule. Instead, the network produces a bag of reactants, which are then synthesized with a reaction predictor.[133] This model only predicts single-step reactions. In a subsequent improvement,[130] the network generates synthesis-directed acyclic graphs (DAGs). The model is then used as part of an autoencoder to navigate the synthesis DAG space, and as the target function of a hill climbing algorithm for molecular optimization.

Assessing the performances of de novo molecular generative models is not a trivial task. QED optimization is often used as reference objective; however, first for current generative models it is not difficult to optimize this value and second, it represents only a tiny fraction of a real molecular design process.[127] Furthermore, the tasks proposed by the most utilized benchmarks can be easily solved using trivial models and the baselines models reach quite high performances as well,[134,135] spoiling the value of the benchmarks themselves. For this reason, despite the ongoing developments of more challenging benchmarks[136] the most reliable way to measure the effectiveness of a method is currently in vitro validation. A recent graph-based model[87] successfully generated a novel JAK1 inhibitor, which was then patented (WO2020182159A1) independently 2 months later, demonstrating this approach's ability to generate drugs like a human expert. However, there currently is a need to find a middle ground between toy tasks and in vitro validation.[126]

In general, many intriguing and diverse architectures have been proposed to solve de novo molecular generation. Using these models in everyday drug discovery could boost performance and save time. Target awareness and synthesis awareness (and guidance) are two key directions that have not yet been fully explored. Both could boost the use of these models for everyday drug discovery. Together with improved scoring functions to predict molecular properties, we expect that developments in this area will make these approaches competitive with simple database scanning, as is currently routine in virtual screening and docking campaigns or computational free energy and kinetics estimation methods.[128] One should consider that there is not yet clear evidence of the superiority of these methods with respect to traditional drug design. This brings the opportunity of merging deep learning techniques with established approaches and hence using GNNs more like boosters[137] than fully fledged alternate methods. Furthermore, the scientific community is dedicating considerable efforts to explainable AI and drug discovery is one of the fields in which adding knowledge on how and why the models work can be fundamental.[138] Being able to interpret and explain graph-based frameworks[139] can allow researchers to embed more complex state transition dynamics or to model the latent probability distribution in a more appropriate way. As a result, this can help to hardwire chemical domain knowledge that goes beyond simple chemical validity constraints.

AUTHOR CONTRIBUTIONS

Carlo Abate: Conceptualization (equal); formal analysis (lead); writing – original draft (lead); writing – review and editing (supporting). **Sergio Decherchi:** Conceptualization (equal); formal analysis (supporting); supervision (lead); writing – original draft (supporting); writing – review and editing (lead). **Andrea Cavalli:** Conceptualization (supporting); supervision (supporting); writing – review and editing (supporting).

CONFLICT OF INTEREST

Sergio Decherchi and Andrea Cavalli declare a conflict of interest as they are co-founders of BiKi Technologies s.r.l., a company selling the BiKi Life Sciences software suite for computational drug discovery.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study

ORCID

Carlo Abate https://orcid.org/0000-0001-7024-3540
Sergio Decherchi https://orcid.org/0000-0001-8371-2270

RELATED WIREs ARTICLES

From machine learning to deep learning: Advances in scoring functions for protein-ligand docking
Learning molecular potentials with neural networks
Application advances of deep learning methods for de novo drug design and molecular dynamics simulation

ENDNOTE

* The definition of MPNN that we give here is taken from the study of Oneto et al.[37] The original MPNN formulation[36] does not include F_i and F_j in Equation (1), but following the more general definition of message passing operation in the study of Oneto et al.,[37] the original GNNs[38] can be seen as a special case of such framework.

REFERENCES

1. Wouters OJ, McKee M, Luyten J. Estimated research and development investment needed to bring a new medicine to market, 2009–2018. JAMA. 2020;323(9):844–53. https://doi.org/10.1001/jama.2020.1166
2. Smith GF. Designing drugs to avoid toxicity. Prog Med Chem. 2011;50:1–47. https://doi.org/10.1016/B978-0-12-381290-2.00001-X
3. Caldwell GW, Yan Z, Tang W, Dasgupta M, Hasting B. ADME optimization and toxicity assessment in early- and late-phase drug discovery. Curr Top Med Chem. 2009;9(11):965–80. https://doi.org/10.2174/156802609789630929
4. Caldwell GW, Ritchie DM, Masucci JA, Hageman W, Yan Z. The new pre-preclinical paradigm: compound optimization in early and late phase drug discovery. Curr Top Med Chem. 2001;1(5):353–66. https://doi.org/10.2174/1568026013394949
5. Liu X, Chen C, Smith BJ. Progress in brain penetration evaluation in drug discovery and development. J Pharmacol Exp The. 2008;325(2):349–56. https://doi.org/10.1124/jpet.107.130294
6. He Q, Liu J, Liang J, Liu X, Li W, Liu Z, et al. Towards improvements for penetrating the blood-brain barrier—recent progress from a material and pharmaceutical perspective. Cells. 2018;7(4):24. https://doi.org/10.3390/cells7040024
7. Savjani KT, Gajjar AK, Savjani JK. Drug solubility: importance and enhancement techniques. Pharmacology. 2012;2012(7):195727. https://doi.org/10.5402/2012/195727
8. Hughes JP, Rees S, Kalindjian SB, Philpott KL. Principles of early drug discovery. Br J Pharmacol. 2011;162(6):1239–49. https://doi.org/10.1111/j.1476-5381.2010.01127.x
9. Schneider G. Automating drug discovery. Nat Rev Drug Discov. 2018;17(2):97–113. https://doi.org/10.1038/nrd.2017.232
10. Paul D, Sanap G, Shenoy S, Kalyane D, Kalia K, Tekade RK. Artificial intelligence in drug discovery and development. Drug Discov Today. 2020;26(1):80–93. https://doi.org/10.1016/j.drudis.2020.10.010
11. Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comput Sci. 1988;28(1):31–6. https://doi.org/10.1021/ci00057a005
12. Gupta A, Müller AT, Huisman BJH, Fuchs JA, Schneider P, Schneider G. Generative recurrent networks for de novo drug design. Mol Inform. 2018;37(1-2):1700111. https://doi.org/10.1002/minf.201700111
13. Yang L, Yang G, Bing Z, Tian Y, Niu Y, Huang L, et al. Transformer-based generative model accelerating the development of novel BRAF inhibitors. ACS Omega. 2021;6(49):33864–73. https://doi.org/10.1021/acsomega.1c05145
14. Yu L, Zhang W, Wang J, Yu Y. SeqGAN: sequence generative adversarial nets with policy gradient. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI'17. Washington, DC: Association for the Advancement of Artificial Intelligence Press; 2017. p. 2852–8. https://doi.org/10.48550/arXiv.1609.05473
15. Ertl P, Lewis R, Martin E, Polyakov V. In silico generation of novel, drug-like chemical matter using the LSTM neural network. arXiv. 2017. https://arxiv.org/abs/1712.07449
16. Blaschke T, Olivecrona M, Engkvist O, Bajorath J, Chen H. Application of generative autoencoder in de novo molecular design. Molecular Informatics. 2018;37(1-2):1700123. https://doi.org/10.1002/minf.201700123
17. Olivecrona M, Blaschke T, Engkvist O, Chen H. Molecular de-novo design through deep reinforcement learning. J Cheminform. 2017;9(1):48. https://doi.org/10.1186/s13321-017-0235-x
18. Segler MHS, Kogej T, Tyrchan C, Waller MP. Generating focused molecule libraries for drug discovery with recurrent neural networks. ACS Central Sci. 2018;4(1):120–31. https://doi.org/10.1021/acscentsci.7b00512
19. Dai H, Tian Y, Dai B, Skiena S, Song L. Syntax-directed Variational autoencoder for structured data. arXiv. 2018. https://arxiv.org/abs/1802.08786
20. Arús-Pous J, Johansson SV, Prykhodko O, Bjerrum EJ, Tyrchan C, Reymond JL, et al. Randomized SMILES strings improve the quality of molecular generative models. J Cheminform. 2019;11(1):71. https://doi.org/10.1186/s13321-019-0393-0
21. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;12(9):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735
22. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Red Hook, NY: Curran Associates Inc; 2017. p. 6000–10. https://doi.org/10.5555/3295222.3295349
23. Bjerrum EJ, Sattarov B. Improving chemical autoencoder latent space and molecular de novo generation diversity with heteroencoders. Biomolecules. 2018;8(4):131. https://doi.org/10.3390/biom8040131
24. Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A. Self-referencing embedded strings (SELFIES): a 100% robust molecular string representation. Mach Learn Sci Technol. 2020;1(4):045024. https://doi.org/10.1088/2632-2153/aba947
25. Jiang J, Zhang R, Zhao Z, Ma J, Liu Y, Yuan Y, et al. MultiGran-SMILES: multi-granularity SMILES learning for molecular property prediction. Bioinformatics. 2022;38:4573–80. https://doi.org/10.1093/bioinformatics/btac550
26. Root-aligned SMILES: a tight representation for chemical reaction prediction. Chem Sci. 2022;8(13):9023–34. https://doi.org/10.1039/D2SC02763A

27. Zhang Z, Cui P, Zhu W. Deep learning on graphs: a survey. IEEE Trans Knowl Data Eng. 2020;14(8):1–270. https://doi.org/10.1109/tkde.2020.2981333
28. Gaudelet T, Day B, Jamasb AR, Soman J, Regep C, Liu G, et al. Utilizing graph machine learning within drug discovery and development. Brief Bioinform. 2021;22(6):1–22. https://doi.org/10.1093/bib/bbab159
29. Frasconi P, Gori M, Sperduti A. A general framework for adaptive processing of data structures. IEEE Trans Neural Netw. 1998;9(5):768–86. https://doi.org/10.1109/72.712151
30. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: a review of methods and applications. AI Open. 2020;1:57–81. https://doi.org/10.1016/j.aiopen.2021.01.001
31. Erdös P, Rényi A. On random graphs I. Public Math Debrecen. 1959;6:290. https://www.renyi.hu/~p_erdos/1959-11.pdf
32. Watts DJ, Strogatz SH. Collective dynamics of "small-world" networks. Nature. 1998;393(6684):440–2. https://doi.org/10.1038/30918
33. Albert R, Barabási AL. Statistical mechanics of complex networks. Rev Mod Phys. 2002;74(1):47–97. https://doi.org/10.1103/RevModPhys.74.47
34. Polishchuk PG, Madzhidov TI, Varnek A. Estimation of the size of drug-like chemical space based on GDB-17 data. J Comput Aided Mol Des. 2013;27(8):675–9. https://doi.org/10.1007/s10822-013-9672-4
35. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst. 2021;32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386
36. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. Proceedings of the 34th International Conference on Machine Learning—Vol. 70. ICML'17; 2017. p. 1263–72. JMLR.org. https://dl.acm.org/doi/10.5555/3305381.3305512
37. Oneto L, Navarin N, Biggio B, Errica F, Micheli A, Scarselli F, et al. Towards learning trustworthily, automatically, and with guarantees on graphs: an overview. Neurocomputing. 2022;493:217–43, 243. https://doi.org/10.1016/j.neucom.2022.04.072
38. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw. 2009;20(1):61–80. https://doi.org/10.1109/TNN.2008.2005605
39. Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? International Conference on Learning Representations; 2019. Available from. https://openreview.net/forum?id=ryGs6iA5Km
40. Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. Proceedings of the 4th International Conference on Learning Representations. ICLR'16; 2016. Available from. http://arxiv.org/abs/1511.05493
41. Veličković P, Casanova A, Liò P, Cucurull G, Romero A, Bengio Y. Graph attention networks. 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings; 2018. Available from. https://openreview.net/forum?id=rJXMpikCZ
42. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. Proceedings of the 5th International Conference on Learning Representations. ICLR '17; 2017. Available from. https://openreview.net/forum?id=SJU4ayYgl
43. Hammond DK, Vandergheynst P, Gribonval R. Wavelets on graphs via spectral graph theory. Appl Comput Harmon Anal. 2011;30(2):129–50.
44. Schlichtkrull M, Kipf TN, Bloem P, Rvd B, Titov I, Welling M. Modeling relational data with graph convolutional networks. European Semantic Web Conference. New York, NY: Springer; 2018. p. 593–607. https://doi.org/10.1007/978-3-319-93417-4_38
45. Simonovsky M, Komodakis N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos, CA: IEEE Computer Society; 2017. p. 29–38. https://doi.org/10.1109/CVPR.2017.11
46. Danel T, Spurek P, Tabor J, Smieja M, Struski L, Slowik A, et al. Spatial graph convolutional networks. 5th International Conference on Neural Information Processing. Vol. 1333 of Communications in Computer and Information Science. New York, NY: Springer; 2020. p. 668–75. https://doi.org/10.1007/978-3-030-63823-8_76
47. Kearnes S, McCloskey K, Berndl M, Pande V, Riley P. Molecular graph convolutions: moving beyond fingerprints. J Comput Aid Mol Des. 2016;30(8):595–608. https://doi.org/10.1007/s10822-016-9938-8
48. Duvenaud DK, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A, et al. Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems 28. Red Hook, NY: Curran Associates, Inc; 2015. p. 2224–32. Available from. http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints.pdf
49. Wieder O, Kohlbacher S, Kuenemann M, Garon A, Ducrot P, Seidel T, et al. A compact review of molecular property prediction with graph neural networks. Drug Discov Today Technol. 2020;37:1–12. https://doi.org/10.1016/j.ddtec.2020.11.009
50. Kingma DP, Welling M. Auto-encoding variational Bayes. 2nd International Conference on Learning Representations; 2014. Available from. http://arxiv.org/abs/1312.6114v10
51. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Advances in neural information processing systems. Red Hook, NY: Curran Associates, Inc; 2014. https://doi.org/10.48550/arXiv.1406.2661
52. Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. 2017 IEEE International Conference on Computer Vision. Los Alamitos, CA: IEEE Computer Society; 2017. p. 2242–51. https://doi.org/10.1109/ICCV.2017.244
53. Kobyzev I, Prince SJD, Brubaker MA. Normalizing flows: an introduction and review of current methods. IEEE Trans Pattern Anal Mach Intell. 2021;43(11):3964–79. https://doi.org/10.1109/TPAMI.2020.2992934

54. Ho J, Chen X, Srinivas A, Duan Y, Abbeel P. Flow++: improving flow-based generative models with variational dequantization and architecture design. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning. Vol. 97 of Proceedings of Machine Learning Research. PMLR; 2019. p. 2722–30. Available from. https://proceedings.mlr.press/v97/ho19a.html
55. Song Y, Ermon S. Generative modeling by estimating gradients of the data distribution. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc; 2019. https://doi.org/10.5555/3454287.3455354
56. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20. Red Hook, NY: Curran Associates Inc; 2020. https://doi.org/10.5555/3495724.3496298
57. Welling M, Teh YW. Bayesian learning via stochastic gradient langevin dynamics. Proceedings of the 28th International Conference on International Conference on Machine Learning. ICML'11. Madison, WI: Omnipress; 2011. p. 681–8. https://doi.org/10.5555/3104482.3104568
58. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations; 2015. https://doi.org/10.48550/arXiv.1409.0473
59. Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. Proceedings of the 12th International Conference on Neural Information Processing Systems. NIPS'99. Cambridge, MA: MIT Press; 1999. p. 1057–63. Available from. https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf
60. Liu Q, Allamanis M, Brockschmidt M, Gaunt AL. Constrained graph variational autoencoders for molecule design. Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18. Red Hook, NY: Curran Associates Inc; 2018. p. 7806–15. https://doi.org/10.5555/3327757.3327877
61. Jin W, Barzilay R, Jaakkola T. Junction tree variational autoencoder for molecular graph generation. RSC Drug Discov Ser. 2021;75:228–49. https://doi.org/10.1039/9781788016841-00228
62. Simonovsky M, Komodakis N. GraphVAE: Towards generation of small graphs using variational autoencoders. Lect Notes Comput Sci. 2018;11139 LNCS:412–22. https://doi.org/10.1007/978-3-030-01418-6_41
63. De Cao N, Kipf T. MolGAN: An implicit generative model for small molecular graphs. ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models; 2018. Available from. https://arxiv.org/abs/1805.11973
64. Li Y, Zhang L, Liu Z. Multi-objective de novo drug design with conditional graph generative model. J Chem. 2018;10(1):1–24. https://doi.org/10.1186/s13321-018-0287-6
65. Zheng S, Lei Z, Ai H, Chen H, Deng D, Yang Y. Deep scaffold hopping with multimodal transformer neural networks. J Chem. 2021;13(1):1–15. https://doi.org/10.1186/s13321-021-00565-5
66. You J, Liu B, Ying R, Pande V, Leskovec J. Graph convolutional policy network for goal-directed molecular graph generation. Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18. Red Hook, NY: Curran Associates Inc; 2018. p. 6412–22. https://doi.org/10.5555/3327345.3327537
67. Popova M, Shvets M, Oliva J, Isayev O. MolecularRNN: Generating realistic molecular graphs with optimized properties. arXiv. 2019. http://arxiv.org/abs/1905.13372
68. Shi C, Xu M, Zhu Z, Zhang W, Zhang M, Tang J. GraphAF: a flow-based autoregressive model for molecular graph generation. 8th International Conference on Learning Representations; 2020. Available from. https://openreview.net/forum?id=S1esMkHYPr
69. Jin W, Yang K, Barzilay R, Jaakkola T. Learning multimodal graph-to-graph translation for molecule optimization. 7th International Conference on Learning Representations; 2019. Available from. https://openreview.net/forum?id=B1xJAsA5F7
70. Jin W, Barzilay R, Jaakkola T. Hierarchical generation of molecular graphs using structural motifs. Proceedings of the 37th International Conference on Machine Learning. ICML'20. JMLR.org; 2020. https://doi.org/10.5555/3524938.3525387
71. Atance SR, Diez JV, Engkvist O, Olsson S, Mercado R. De novo drug design using reinforcement learning with graph-based deep generative models. J Chem Inform Model. 2022;62(20):4863–72. https://doi.org/10.1021/acs.jcim.2c00838
72. Lim J, Hwang SY, Moon S, Kim S, Kim WY. Scaffold-based molecular design with a graph generative model. Chem Sci. 2020;11:1153–64. https://doi.org/10.1039/C9SC04503A
73. Khemchandani Y, O'Hagan S, Samanta S, Swainston N, Roberts TJ, Bollegala D, et al. DeepGraphMolGen, a multi-objective, computational strategy for generating molecules with desirable properties: a graph convolution and reinforcement learning approach. J Chem. 2020;12(1):1–17. https://doi.org/10.1186/s13321-020-00454-3
74. Samanta B, De A, Jana G, Gomez V, Chattaraj PK, Ganguly N, et al. NEVAE: a deep generative model for molecular graphs. J Mach Learn Res. 2022;21(1):1110–7. https://doi.org/10.1609/aaai.v33i01.33011110
75. Mahmood O, Mansimov E, Bonneau R, Cho K. Masked graph modeling for molecule generation. Nat Commun. 2021;12(1):3156. https://doi.org/10.1038/s41467-021-23415-2
76. Kwon Y, Yoo J, Choi YS, Son WJ, Lee D, Kang S. Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation. J Chem. 2019;11(1):1–10. https://doi.org/10.1186/s13321-019-0396-x
77. Assouel R, Ahmed M, Segler MH, Saffari A, Bengio Y. DEFactor: Differentiable edge factorization-based probabilistic graph generation. arXiv. 2018;1–14. http://arxiv.org/abs/1811.09766
78. Bresson X, Laurent T. A two-step graph convolutional decoder for molecule generation. arXiv. 2019. https://arxiv.org/abs/1906.03412
79. Zang C, Wang F. MoFlow: an invertible flow model for generating molecular graphs. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '20. New York, NY: Association for Computing Machinery; 2020. p. 617–26. https://doi.org/10.1145/3394486.3403104

80. Ma C, Zhang X. GF-VAE: a flow-based variational autoencoder for molecule generation. Proceedings of the 30th ACM International Conference on Information & Knowledge Management. CIKM '21. New York, NY: Association for Computing Machinery; 2021. p. 1181–90. https://doi.org/10.1145/3459637.3482260
81. Madhawa K, Ishiguro K, Nakago K, Abe M. GraphNVP: an invertible flow model for generating molecular graphs. arXiv. 2019. https://arxiv.org/abs/1905.11600
82. Flam-Shepherd D, Wu TC, Aspuru-Guzik A. MPGVAE: improved generation of small organic molecules using message passing neural nets. Mach Learn Sci Technol. 2021;2(4):045010. https://doi.org/10.1088/2632-2153/abf5b7
83. Kuznetsov M, Polykovskiy D. MolGrow: a graph normalizing flow for hierarchical molecular generation. Proceedings of the Thirty-Fifth Conference on Association for the Advancement of Artificial Intelligence (AAAI). Washington, DC: Association for the Advancement of Artificial Intelligence Press; 2021. p. 8226–34. https://doi.org/10.48550/arXiv.2106.05856
84. Tan C, Gao Z, Li SZ. Target-aware molecular graph generation. arXiv. 2022. https://arxiv.org/abs/2202.04829
85. Maziarka Ł, Pocha A, Kaczmarczyk J, Rataj K, Danel T, Warchoł M. Mol-CycleGAN: a generative model for molecular optimization. Journal of. Chem. 2020;12(1):1–18. https://doi.org/10.1186/s13321-019-0404-1
86. Bradshaw J, Paige B, Kusner MJ, Segler M, Hernández-Lobato JM. A model to search for synthesizable molecules. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems. Volume 32. Red Hook, NY: Curran Associates, Inc; 2019. Available from. https://proceedings.neurips.cc/paper/2019/file/46d0671dd4117ea366031f87f3aa0093-Paper.pdf
87. Yu Y, Xu T, Li J, Qiu Y, Rong Y, Gong Z, et al. A novel Scalarized scaffold hopping algorithm with graph-based variational autoencoder for discovery of JAK1 inhibitors. ACS Omega. 2021;6(35):22945–54. https://doi.org/10.1021/acsomega.1c03613
88. Shi W, Singha M, Srivastava G, Pu L, Ramanujam J, Brylinski M. Pocket2Drug: an encoder-decoder deep neural network for the target-based drug design. Front Pharmacol. 2022;13:13. https://doi.org/10.3389/fphar.2022.837715
89. Xie Y, Shi C, Zhou H, Yang Y, Zhang W, Yu Y, et al. MARS: Markov molecular sampling for multi-objective drug discovery. International Conference on Learning Representations; 2021. Available from. https://openreview.net/forum?id=kHSu4ebxFXY
90. Imrie F, Bradley AR, van der Schaar M, Deane CM. Deep generative models for 3D linker design. J Chem Inf Model. 2020;60(4):1983–95. https://doi.org/10.1021/acs.jcim.9b01120
91. Jin W, Barzilay R, Jaakkola T. Multi-objective molecule generation using interpretable substructures. Proceedings of the 37th International Conference on Machine Learning. ICML'20. JMLR.org; 2020. https://doi.org/10.5555/3524938.3525388
92. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis, MN: Association for Computational Linguistics; 2019. p. 4171–86. Available from. https://aclanthology.org/N19-1423
93. Jo J, Lee S, Hwang SJ. Score-based generative modeling of graphs via the system of stochastic differential equations. Thirty-ninth International Conference on Machine Learning. JMLR.org; 2022. https://doi.org/10.48550/arXiv.2202.02514
94. Mercado R, Rastemo T, Lindelöf E, Klambauer G, Engkvist O, Chen H, et al. Graph networks for molecular design. Mach Learn Sci Technol. 2020;2(2):1–37. https://doi.org/10.1088/2632-2153/abcf91
95. Randić M. On canonical numbering of atoms in a molecule and graph isomorphism. J Chem Inform Comput Sci. 1977;17(3):171–80. https://doi.org/10.1021/ci60011a013
96. Mavrovouniotis ML, Chang S. Hierarchical neural networks. Comput Chem Eng. 1992;16(4):347–69. https://doi.org/10.1016/0098-1354(92)80053-C
97. He K, Zhang X, Ren S, Sun J. Identity mappings in deep residual networks. Lect Notes Comput Sci. 2016;9908 LNCS:630–45. https://doi.org/10.1007/978-3-319-46493-0_38
98. You J, Ying R, Ren X, Hamilton W, Leskovec J. GraphRNN: generating realistic graphs with deep auto-regressive models. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. Vol. 80 of Proceedings of Machine Learning Research. JMLR.org; 2018. p. 5708–17. Available from. https://proceedings.mlr.press/v80/you18a.html
99. Bongini P, Bianchini M, Scarselli F. Molecular generative graph neural networks for drug discovery. Neurocomputing. 2021;450:242–52. https://doi.org/10.1016/j.neucom.2021.04.039
100. Papamakarios G, Pavlakou T, Murray I. Masked autoregressive flow for density estimation. Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Red Hook, NY: Curran Associates Inc; 2017. p. 2335–44. https://doi.org/10.5555/3294771.3294994
101. Rigoni D, Navarin N, Sperduti A. Conditional constrained graph variational autoencoders for molecule design. IEEE Symposium Series on Computational Intelligence (SSCI). New York, NY: IEEE; 2020. p. 729–36. https://doi.org/10.1109/SSCI47803.2020.9308554
102. Kearnes S, Li L, Riley P. Decoding molecular graph embeddings with reinforcement learning. arXiv. 2019. https://arxiv.org/abs/1904.08915
103. Ma T, Chen J, Xiao C. Constrained generation of semantically valid graphs via regularizing variational autoencoders. Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18. Red Hook, NY: Curran Associates Inc; 2018. p. 7113–24. https://doi.org/10.5555/3327757.3327814
104. Courant R. Variational methods for the solution of problems of equilibrium and vibrations. Bull Am Math Soc. 1943;49:1–23. https://doi.org/10.1090/S0002-9904-1943-07818-4

105. Pölsterl S, Wachinger C. Adversarial learned molecular graph inference and generation. Machine Learning and Knowledge Discovery in Databases: European Conference, Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag; 2020. p. 173–89. https://doi.org/10.1007/978-3-030-67661-2_11
106. Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL. Quantifying the chemical beauty of drugs. Nat Chem. 2012;4(2):90–8. https://doi.org/10.1038/nchem.1243
107. Ertl P, Schuffenhauer A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. J Cheminform. 2009;1(1):8. https://doi.org/10.1186/1758-2946-1-8
108. Snoek J, Larochelle H, Adams RP. Practical Bayesian optimization of machine learning algorithms. Proceedings of the 25th International Conference on Neural Information Processing Systems—Vol. 2. NIPS'12. Red Hook, NY: Curran Associates Inc; 2012. p. 2951–9. https://doi.org/10.5555/2999325.2999464
109. Snelson E, Ghahramani Z. Sparse Gaussian processes using pseudo-inputs. Proceedings of the 18th International Conference on Neural Information Processing Systems. NIPS'05. Cambridge, MA: MIT Press; 2005. p. 1257–64. https://doi.org/10.5555/2976248.2976406
110. Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. J Global Optim. 1998;13(4):455–92. https://doi.org/10.1023/A:1008306431147
111. Dai H, Nazi A, Li Y, Dai B, Schuurmans D. Scalable deep generative modeling for sparse graphs. Proceedings of the 37th International Conference on Machine Learning. ICML'20. JMLR.org; 2020. https://doi.org/10.5555/3524938.3525153
112. Cofala T, Kramer O. Transformers for molecular graph generation. ESANN 2021 Proceedings, European Symposium on Artificial Neural Networks. Louvain-la-Neuve: i6doc.com; 2021. p. 123–8. https://doi.org/10.14428/esann/2021.es2021-112
113. Maziarka Ł, Danel T, Mucha S, Rataj K, Tabor J, Jastrzębski S. Molecule attention transformer. arXiv. 2020. http://arxiv.org/abs/2002.08264
114. Jiang D, Wu Z, Hsieh CY, Chen G, Liao B, Wang Z, et al. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. J Cheminform. 2021;13(1):12. https://doi.org/10.1186/s13321-020-00479-8
115. Jensen JH. A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. Chem Sci. 2019;10(12):3567–72. https://doi.org/10.1039/c8sc05372c
116. Jeon W, Kim D. Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors. Sci Rep. 2020;10(1):22104. https://doi.org/10.1038/s41598-020-78537-2
117. Spiegel JO, Durrant JD. AutoGrow4: an open-source genetic algorithm for de novo drug design and lead optimization. J Cheminform. 2020;12(1):25. https://doi.org/10.1186/s13321-020-00429-4
118. Simm GNC, Pinsler R, Hernández-Lobato JM. Reinforcement learning for molecular design guided by quantum mechanics. Proceedings of the 37th International Conference on Machine Learning. ICML'20. JMLR.org; 2020. https://doi.org/10.5555/3524938.3525769
119. Balcilar M, Héroux P, Gaüzère B, Vasseur P, Adam S, Honeine P. Breaking the limits of message passing graph neural networks. Proceedings of the 38th International Conference on Machine Learning (ICML). JMLR.org; 2021. https://doi.org/10.48550/arXiv.2106.04319
120. Bevilacqua B, Frasca F, Lim D, Srinivasan B, Cai C, Balamurugan G, et al. Equivariant subgraph aggregation networks. International Conference on Learning Representations; 2022. https://openreview.net/forum?id=dFbKQaRk15w
121. Valsesia D, Fracastoro G, Magli E. RAN-GNNs: breaking the capacity limits of graph neural networks. IEEE Trans Neural Netw Learn Syst. 2021;1–10. https://doi.org/10.1109/TNNLS.2021.3118450
122. Bodnar C, Frasca F, Otter N, Wang Y, Liò P, Montufar GF, et al. Weisfeiler and Lehman go cellular: CW networks. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Vaughan JW, editors. Advances in neural information processing systems. Volume 34. Red Hook, NY: Curran Associates, Inc; 2021. p. 2625–40. Available from. https://openreview.net/forum?id=uVPZCMVtsSG
123. Brandstetter J, Hesselink R, van der Pol E, Bekkers EJ, Welling M. Geometric and physical quantities improve E(3) equivariant message passing. International Conference on Learning Representations; 2022. Available from. https://openreview.net/forum?id=_xwr8gOBeV1
124. Hoogeboom E, Satorras VG, Vignac C, Welling M. Equivariant diffusion for molecule generation in 3D. In: Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, Sabato S, editors. Proceedings of the 39th International Conference on Machine Learning. Vol. 162 of Proceedings of Machine Learning Research. JMLR.org; 2022. p. 8867–87. Available from. https://proceedings.mlr.press/v162/hoogeboom22a.html
125. Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys Ther. 2021 Jun;3(6):422–40. https://doi.org/10.1038/s42254-021-00314-5
126. Meyers J, Fabian B, Brown N. De novo molecular design and generative models. Drug Discov Today. 2021;11(26):2707–15. https://doi.org/10.1016/J.DRUDIS.2021.05.019
127. Bilodeau C, Jin W, Jaakkola T, Barzilay R, Jensen KF. Generative models for molecular discovery: recent advances and challenges. Mach Learn. 2022;12(5):e1608. https://doi.org/10.1002/wcms.1608
128. Decherchi S, Cavalli A. Thermodynamics and kinetics of drug-target binding by molecular simulation. Chem Rev. 2020;120(23):12788–833. https://doi.org/10.1021/acs.chemrev.0c00534
129. Decherchi S, Grisoni F, Tiwary P, Cavalli A. Editorial: molecular dynamics and machine learning in drug discovery. Front Mol Biosci. 2021;4(8):673773. https://doi.org/10.3389/fmolb.2021.673773

130. Bradshaw J, Paige B, Kusner MJ, Segler MHS, Hernández-Lobato JM. Barking up the right tree: an approach to search over molecule synthesis DAGs. Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20. Red Hook, NY: Curran Associates Inc; 2020. https://doi.org/10.5555/3495724.3496299
131. Gao W, Mercado R, Coley CW. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. International Conference on Learning Representations; 2022. Available from. https://openreview.net/forum?id=FRxhHdnxt1
132. Nguyen DH, Tsuda K. Generating reaction trees with cascaded variational autoencoders. J Chem Phys. 2022;156(4):044117. https://doi.org/10.1063/5.0076749
133. Schwaller P, Laino T, Gaudin T, Bolgar P, Hunter CA, Bekas C, et al. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. ACS Cent Sci. 2019;5(9):1572–83. https://doi.org/10.1021/acscentsci.9b00576
134. Grant LL, Sit CS. De novo molecular drug design benchmarking. RSC Med Chem. 2021;8(12):1273–80. https://doi.org/10.1039/D1MD00074H
135. Zhang J, Mercado R, Engkvist O, Chen H. Comparative study of deep generative models on chemical space coverage. J Chem Inform Model. 2021;61(6):2572–81. https://doi.org/10.1021/acs.jcim.0c01328
136. Gao W, Fu T, Sun J, Coley CW. Sample efficiency matters: A benchmark for practical molecular optimization. Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track. Red Hook, NY: Curran Associates, Inc; 2022. Available from. https://openreview.net/forum?id=yCZRdI0Y7G
137. Kimber TB, Chen Y, Volkamer A. Deep learning in virtual screening: recent applications and developments. Int J Mol Sci. 2021;22(9):4435. https://doi.org/10.3390/ijms22094435
138. Jiménez-Luna J, Grisoni F, Schneider G. Drug discovery with explainable artificial intelligence. Nat Mach Intell. 2020;2(10):573–84. https://doi.org/10.1038/s42256-020-00236-4
139. Yuan H, Yu H, Gui S, Ji S. Explainability in graph neural networks: a taxonomic survey. IEEE Trans Pattern Anal Mach Intell. 2022;1–19. https://doi.org/10.1109/TPAMI.2022.3204236

How to cite this article: Abate C, Decherchi S, Cavalli A. Graph neural networks for conditional de novo drug design. WIREs Comput Mol Sci. 2023;13(4):e1651. https://doi.org/10.1002/wcms.1651