arXiv:2209.05582v2 [cs.LG] 6 Feb 2023

Graph Neural Networks for Molecules

A Chapter for Book "Machine Learning in Molecular Sciences"

Yuyang Wang[12], Zijie Li[1], and Amir Barati Farimani[123⋆]

1 Department of Mechanical Engineering, Carnegie Mellon University
2 Machine Learning Department, Carnegie Mellon University
3 Department of Chemical Engineering, Carnegie Mellon University
Pittsburgh, PA 15213, USA
{yuyangw,zijieli,barati}@cmu.edu

Abstract. Graph neural networks (GNNs), which are capable of learning representations from graphical data, are naturally suitable for modeling molecular systems. This review introduces GNNs and their various applications for small organic molecules. GNNs rely on message-passing operations, a generic yet powerful framework, to update node features iteratively. Many researches design GNN architectures to effectively learn topological information of 2D molecule graphs as well as geometric information of 3D molecular systems. GNNs have been implemented in a wide variety of molecular applications, including molecular property prediction, molecular scoring and docking, molecular optimization and *de novo* generation, molecular dynamics simulation, etc. Besides, the review also summarizes the recent development of self-supervised learning for molecules with GNNs.

Keywords: Graph Neural Network, Molecular Modeling, Quantitative Structure-activity Relationship, Molecular Generation, Molecular Simulation, Self-supervised Learning

⋆ Corresponding author.

Contents

1 Message Passing Graph Neural Networks

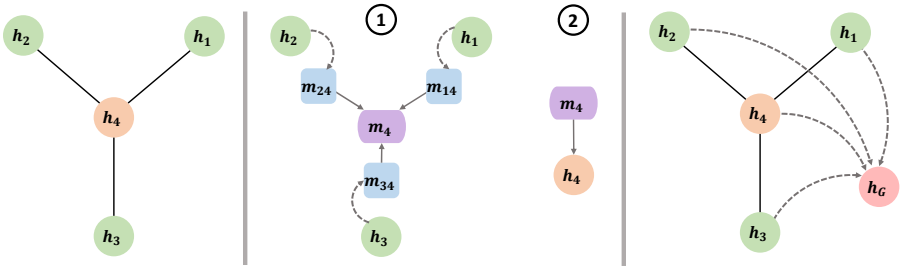
Graphs are ubiquitous data structure that expresses a set of objects and the relationships between them. Formally, a graph $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denotes the set of objects (nodes) and their relationships (edges), respectively [1]. Fig. 1(a) shows an example of a graph containing four nodes and three edges. Graphs are powerful expressions that can model a wide range of systems. For example, the social network can be modeled as a graph where each node represents a person and each edge represents social connections between them, e.g., friendship, spouseship, colleagueship, etc [2–4]. Other graphical systems include chemical compounds [5, 6], knowledge graphs [7, 8], physical systems [9, 10], and various other domains [11–13]. Recently, there is growing attention from the machine learning (ML) community to develop ML models, especially deep neural networks (DNNs) [14], to analyze graphical data [15, 16]. Previous deep learning models, including convolutional neural networks (CNNs) [17, 18] and recurrent neural networks (RNNs) [19], fail to directly operate on graphical structures. Graph neural networks (GNNs), a deep learning method, are developed to learn representations from graphs directly [20, 21]. GNNs have been prevalent in various domains and many different tasks. In what follows, we will introduce the basic concepts and operations of GNNs.

Modern GNNs are built upon the message-passing layer that aggregates neighboring information to update each node in an iterative manner. The framework is first formalized by Gilmer et al [22]. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the i-th node is $u_i \in \mathcal{V}$ and the edge between u_i and u_j is $e_{ij} \in \mathcal{E}$. Each node u_i is initialized as a feature vector $\boldsymbol{h}_i^{(0)} = \mathrm{Emb}_n(u_i)$ via an node embedding function. Similar to node embedding, each edge e_{ij} is mapped to a feature vector via an edge embedding function $\boldsymbol{a}_{ij} = \mathrm{Emb}_e(e_{ij})$. The message-passing function is demonstrated in Fig. 1(b). Each message-passing layer contains two operations: (1) computation and aggregation of the messages from neighboring nodes and edges, and (2) update of the node feature based on the message and old feature. The two operations at k-th layers in a GNN is given in Equation 1 and 2, respectively.

$$\boldsymbol{m}_i^{(k)} = \sum_{u_j \in \mathcal{N}(u_i)} \phi_m^{(k)}(\boldsymbol{h}_i^{(k-1)}, \boldsymbol{h}_j^{(k-1)}, \boldsymbol{a}_{ij}), \tag{1}$$

$$\boldsymbol{h}_i^{(k)} = \phi_f^{(k)}(\boldsymbol{h}_i^{(k-1)}, \boldsymbol{m}_i^{(k)}), \tag{2}$$

where $\boldsymbol{h}_i^{(k)} \in \mathbb{R}^F$ denotes the feature vector of node u_i at k-th layer, $\mathcal{N}(u_i)$ models all the neighbors of node u_i. $\phi_m^{(k)}$ and $\phi_u^{(k)}$ are the message and update functions. By conducting the two operations iteratively, node features within the graph are updated and can be utilized for node-level tasks. To conduct graph-level tasks, like molecular property prediction, directly utilizing all the node features might be infeasible as different graphs have different numbers of nodes. To obtain the representation of the whole graph, the readout operation is introduced to down-sample the node features as illustrated in Fig. 1(c). For a



(a) An example graph (b) Message-passing function (c) Readout function

Fig. 1. Illustration of (a) an example graph defined by nodes and edges, (b) messaging-passing function containing ① an aggregation operation from $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{h}_3$ to obtain message m_4 and ② an update operation to update the node feature $\boldsymbol{h}_4$ from $\boldsymbol{m}_4$, and (c) readout function to obtain the graph feature $\boldsymbol{h}_G$.

GNN with K layers, after the last message-passing layer, the readout function $R(\cdot)$ is given in Equation 3.

$$\boldsymbol{h}_G = R(\{\boldsymbol{h}_i^{(K)} \mid u_i \in \mathcal{V}\}). \tag{3}$$

The design of the message-passing operation is essential to learning graph representations [23]. Since proposed, different GNNs have been developed with different aggregation and update functions for expressive graph representation learning [24–28]. Early spectral methods are built upon the spectral representation of graphs. Such methods compute the eigenvectors of the graph Laplacian for aggregating neighboring information and apply nonlinear activation functions on the aggregated feature to update each node [29, 30]. Graph convolutional network (GCN) [31] introduces a simple yet generic GNN framework, where the aggregation is implemented as an element-wise mean pooling over the node and its neighbors, and the update is implemented as a linear transformation $\boldsymbol{W}$ followed by nonlinear ReLU function [32] as given in Equation 4.

$$\boldsymbol{h}_i^{(k)} = \text{ReLU}(\boldsymbol{W} \cdot \text{MEAN}(\boldsymbol{h}_j^{(k-1)}) \mid u_j \in \mathcal{N}(u_i) \cup \{u_i\})). \tag{4}$$

GraphSAGE [33] formulates aggregation via multiplication of learnable weight matrix and neighboring features followed by ReLU and max-pooling over the aggregated features as given in Equation 5.

$$\boldsymbol{h}_i^{(k)} = \sigma\left(\boldsymbol{W}' \cdot \left[\text{MAX}\left(\text{ReLU}(\boldsymbol{W} \cdot \boldsymbol{h}_j^{(k-1)}) \mid u_j \in \mathcal{N}(u_i)\right) \parallel h_i^{(k-1)}\right]\right), \tag{5}$$

where $[\cdot\|\cdot]$ is the concatenation and σ is a nonlinear activation function like sigmoid. The update in GraphSAGE contains a concatenation of the node and aggregated features succeeded by a linear transformation $\boldsymbol{W}'$. Equation 6 shows the message-passing in graph isomorphism network (GIN) [34], another widely used GNN architecture. GIN proposes to sum the node features and all the

neighboring features and applies a multi-layer perceptron (MLP) to update the node in the message-passing layer.

$$\boldsymbol{h}_i^{(k)} = \text{MLP}((1+\epsilon)\boldsymbol{h}_i^{(k-1)} + \sum\nolimits_{u_j \in \mathcal{N}(u_i)} \boldsymbol{h}_j^{(k-1)}), \tag{6}$$

Further, graph attention network (GAT) [35] introduces the attention mechanism to message-passing via computing the attention score as the weight coefficient in aggregation. The standard attention follows $\text{Attn}(Q, K, V) = \frac{QK^\top}{\sqrt{d_k}}V$, where Q, K, and V are quey, key, and value matrices of the embedding of each token, respectively, with the square root of the embedding dimension d_k as the scaling factor. The message-passing in GAT adjusts the attention as given in Equation 7 and 8.

$$\boldsymbol{h}_j^{(k)} = \sigma(\frac{1}{K}\sum\nolimits_{k=1}^{K}\sum\nolimits_{u_j \in \mathcal{N}(u_i)\cup\{u_i\}} \alpha_{ik}^k \boldsymbol{W}^k \boldsymbol{h}_j^{(k-1)}), \tag{7}$$

$$\alpha_{ij}^k = \frac{\exp(\text{ReLU}(\boldsymbol{a}^\top[\boldsymbol{W}^k h_i^{(k-1)} \parallel \boldsymbol{W}^k h_j^{(k-1)}]))}{\sum_{u_j \in \mathcal{N}(u_i)} \exp(\text{ReLU}(\boldsymbol{a}^\top[\boldsymbol{W}^k h_i^{(k-1)} \parallel \boldsymbol{W}^k h_j^{(k-1)}]))}, \tag{8}$$

where α_{ij}^k is the scalar that measures attention score, K is the number of attention heads, and $\boldsymbol{a} \in \mathbb{R}^{2F}$ is the weight vector. Multiple works also leverage RNNs to model the message-passing function. Gated graph neural networks (GCNN) by Li et al. [25] employs the gated recurrent unit (GRU) [36] to aggregate neighboring information and update node features as shown in Equation 9.

$$\boldsymbol{h}_i^{(k)} = \text{GRU}\left(\boldsymbol{h}_i^{(k-1)}, \sum\nolimits_{u_j \in \mathcal{N}(u_i)} \boldsymbol{W}\boldsymbol{h}_j^{(k-1)}\right). \tag{9}$$

Long short-term memory (LSTM) [19] is implemented similarly as GRU for message-passing functions. Tai et al. [37] propose Tree-LSTM that extends LSTM to tree-structured data. Such a model is adapted to general graphs by Peng et al. [38] as well as Zayats and Ostendorf [39].

Recently, there are also efforts on adapting the transformer architecture to graphical data [40]. The standard transformer is designed to learn from sequential data like text [41]. How to encode the structural information of the graph is the major challenge in applying the transformer on graphs. To this end, Graphormer [42] is directly built upon the standard transformer with an effective encoding of the structural information. Specifically, Graphormer includes spatial encoding of the shortest path, edge encoding of edge features on the shortest path, and centrality encoding of the degree centrality of each node. Each encoding is corporated with learnable parameters and added to the attention score. Dwivedi et al. [43] introduce a graph transformer that learns from the local attention of neighboring connections. It also implements the Laplacian eigenvectors of the graph as the positional encoding in place of sinusoidal positional encoding. Recently, tokenized graph transformer (TokenGT) is proposed to demonstrate that pure transformers can be powerful graph representation learners [44]. Unlike previous works which only tokenize nodes and integrate edge information in attention

updates, TokenGT tokenizes both nodes and edges with orthonormal node identifiers encoding the connectivity of the tokens and trainable type identifiers that encode whether a token is a node or an edge. TokenGT allows adaption of linear attention (e.g., Performer [45]) introduced for pure transformer and reaches $\mathcal{O}(N+M)$ cost, where N and M are the number of nodes and edges, respectively.

For the readout operation, the primitive choices are pooling over all the node features, including mean-pooling, max-pooling, and summation-pooling. Xu et al. [34] point out that summation-pooling is more expressive than mean- and max-pooling, which can capture the full multiset while the other two poolings fail. Works have also investigated other readout functions to improve representation learning and computational efficiency. Attention mechanisms have been implemented in place of summation- or mean-pooling as the readout operation [22, 25]. Vinyals et al. [46] propose set2set function which implements an LSTM for unordered and size-variant input sets. Other methods have probed the rearrangement of the nodes to down-sample the features. Defferrard et al. [30] coarsen the graph into multi-levels via the Graclus algorithm and then rearrange the nodes into a balanced binary tree. The readout is conducted by aggregating the node features in a bottom-up manner. Following the insight, Zhang et al. [47] propose SortPool which ranks nodes based on their structural roles within the input graph and truncates the size of the graph after ranking. DiffPool [48], on the other hand, develops a differentiable pooling module that generates hierarchical graph representations. It learns a soft clustering assignment at each layer to aggregate node features. SAGPool [49] combines self-attention with end-to-end hierarchical representation learning which includes both node feature and graph topology. In general, the readout operation can be considered as a special message-passing layer that aggregates all the updated node features within the graph. It also plays an important role in learning expressive graph representations.

2 Molecular Graph Neural Networks

The previous section has introduced the basic concepts of the message-passing GNN and its prevalent variants. This section focuses on the GNN architectures that are adapted and designed for molecular representation learning. As the focus of this chapter, molecules, can be naturally viewed as graphs [50]. Fig. 2 illustrates examples of how molecule graphs are built. Within the graph, each node models an atom, and each edge models the interatomic interactions. Interactions include two-dimensional (2D) topological information like covalent bonds and 3D geometric information like distances and angles. Graph representations have demonstrated advantages over other molecular featurization techniques. Some methods develop a language that converts a molecule into a one-dimensional (1D) string, e.g., simplified molecular-input line-entry system (SMILES) [51], SMILES arbitrary target specification (SMARTS) [52], and recently developed self-referencing embedded strings (SELFIES) [53]. Other works propose rules that can embed each molecule into a feature vector, namely a molecular finger-

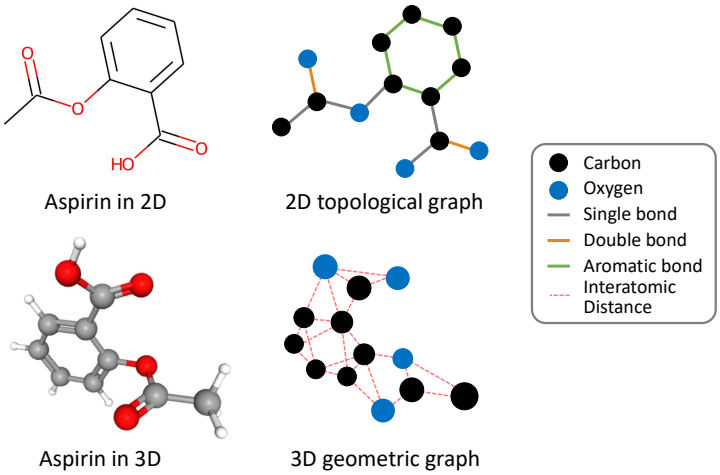


Fig. 2. Illustration of building graphs from molecules using 2D topological and 3D geometric information. The aspirin molecule is shown as an example.

print or a descriptor, that usually encodes the presence or absence of certain substructures in the molecule. Examples include extended-connectivity fingerprint (ECFP) [54] and molecular access system (MACCS) keys [55]. Both string-based and fingerprint-based methods have been widely and successfully implemented in many applications including molecular similarity search, clustering, and virtual screening. With the rise of deep learning in recent years, deep learning methods have been applied to learn molecular representations from molecular languages and fingerprints. Though these featurization techniques follow certain rules to encode substructure information, they, however, still struggle to directly model the topological and geometric information of molecules. GNNs, which learn representation from molecule graphs, are capable of encoding the graphical structures. Many works have investigated the GNN architectures for learning from molecule graphs. Also, atomic and bond descriptors, like atomic number, charge, chirality, hybridization, bond type, etc., can be added to the node and edge features. This chapter focuses on the message-passing architecture design for molecule graphs.

Duvenaud et al. [5] introduce one of the pioneering works that learning molecular representations via a GNN. In this work, node features are aggregated through concatenation and are updated through a learnable matrix conditioned on the node degree followed by a sigmoid function. The readout operation computes a weighted summation over node features from all the layers. Another trailblazing model, named Weave, from Kearnes et al. [6] proposes molecular graph convolutions that update edge features in each message-passing layer. Specifically, at each layer, one first updates the edge features from connected nodes and then updates the node features by aggregating the edges. Hu et al.

[56] extend the standard GIN by including the edge features in message-passing as given in Equation 10.

$$h_v^{(k)} = \text{MLP}\left(h_v^{l-1} + \sum\nolimits_{u_j \in \mathcal{N}(u_i)} (h_j^{(k-1)} + a_{ij})\right), \tag{10}$$

where a_{ij} is the embedding of the edge between nodes u_i and u_j, containing the information of bond type and direction. Glimer et al. [22] elucidate a simple yet unified GNN framework named message passing neural network (MPNN) which most of the previous models fall into. They then propose enn-s2s, a variant of MPNN that aggregates neighboring information as given in Equation 11.

$$a_i^{(k)} = \sum_{u_j \in \mathcal{N}(u_i)} \text{MLP}(a_{ij}) h_j. \tag{11}$$

Besides, it follows set2set [46] as the readout function to obtain molecule representations. Yang et al. [57] further proposed directed MPNN (D-MPNN) that uses information associated with directed edges instead of information associated with vertices in standard MPNN. In addition, D-MPNN implements rich atom and bond features to improve the expressiveness of molecule graphs. Attention mechanisms have been developed for molecular GNNs. Xiong et al. [58] introduce AttentiveFP containing atom embedding layers and molecule embedding layers. The atom embedding layers borrow the attention mechanism introduced in GAT to aggregate local messages and the output attention context is fed into a gated recurrent unit (GRU) together with atom features from the previous layer to obtain updated context. The molecule embedding layers assume a virtual node that connects to all atoms and follows the same pattern as atom embedding layers. The final output of the virtual node is used as the representation of the whole molecule graph. Not only the attention mechanisms but also the transformer architectures are investigated for molecule graphs. Rong et al. [59] introduce GTransformer which combines the message-passing framework with transformer. GTransformer applies a bi-level message-passing strategy that aggregates and updates the information on both nodes and edges. Besides, it employs the residue connection and dynamic message-passing by randomly choosing the number of aggregation hops.

So far, we have discussed GNNs that model molecules as 2D graphs. Namely, these models only consider topological information while ignoring geometric information like distances and angles in the 3D Euclidean space. Such 2D GNNs have demonstrated promising performance in many applications. However, 3D information is crucial as it is closely related to the energy landscape and molecules rely on 3D conformation to function in practice. Nevertheless, simply adding the positional coordinates into GNNs can be problematic as translation or rotation of the molecule will change the output of the models. One would expect to design GNN architectures whose output is immutable to the rotations and translations of 3D molecular structures [60]. To formalize, we borrow the concept from group theory and denote the set of proper rigid transformations (i.e., translations and rotations) in n-dimensional Euclidean space as SE(n) [61]. Many works thus

explore SE(3)-invariant GNNs for molecules existing in 3D Euclidean space. Schütt et al. [62] introduce the deep tensor neural network (DTNN) that model the distances between atoms in message-passing. DTNN extract the graph-level representation by feeding each node feature into an MLP and summing them up. Following DTNN, Schütt et al. further propose SchNet [63] that is composed of well-designed layers to model local correlations between atoms. Equation 12 elaborate the atom-wise update while Equation 13 and 14 elaborate continuous-filter convolution at the $(k+1)$-th layer in SchNet.

$$h_i^{(k)} = W h_i^{(k)} + b \tag{12}$$

$$h_i^{(k)} = \sum\nolimits_{u_j \in \mathcal{N}(u_i)} h_j^{(k)} \circ W_{\text{cf}}^{(k-1)}, \tag{13}$$

$$W_{\text{cf}}^{(k)} = \text{SoftPlus}(W_2^{(k)} \cdot \text{SoftPlus}(W_1^{(k)} \cdot \big|\big|_{u_j \in \mathcal{N}(u_i)} \text{RBF}(d_{ij}))), \tag{14}$$

where $\circ$ is an element-wise multiplication, d_{ij} is the distance in Euclidean space between nodes u_i and u_j, and $\text{RBF}(d_{ij}) = \big|\big|_{k=1}^{K} \exp(-\gamma \|d_{ij} - \mu_k\|^2)$ concatenates radial basis functions with $\mu_k = 0.1k$Å and $\gamma = 10$Å that expands the interatomic distance from a scalar to a vector of $\mathbb{R}^K$. Besides, it implements SoftPlus, a smooth approximation to ReLU, as nonlinear activation functions. Each message-passing layer starts with an atomise-wise update followed by a continuous-filter convolution operation. It then conducted two atom-wise updates with a SoftPlus activation in between to obtain the combination term $v_i^{(k)}$. The final output node feature at the $(k+1)$-th layer is updated by $h_i^{(k+1)} = h_i^{(k)} + v_i^{(k)}$. SchNet effectively encodes 3D distance information to molecular GNN and inspires many follow-up works in this domain. PhysNet by Unke et al. [64] also leverages the interatomic distances to build an SE(3)-invariant GNN, which adapts the interaction blocks to update node features from distances and the residual blocks to learn representations with deeper neural networks. DimeNet by Gasteiger et al. [65] follows the architecture of PhysNet while integrating additional angular information expanded with Fourier-Bessel representations. The message-passing at the k-th layer from node u_j to u_i takes in not only the distance d_{ij} but also angles $\angle u_k u_j u_i$ as well as $h_k^{(k-1)}$, where $u_k \in \mathcal{N}(u_j) \backslash \{u_i\}$. The same team further proposes an improved version named DimeNet++ [66] with fast interactions and embedding hierarchy. Fang et al. [67] report GeoGNN containing the atom-bond graph and bond-angle graph to incorporate both interatomic distances and angles. In the atom-bond graph, each node represents an atom and each edge represents a covalent bond, while in the bond-angle graph, each node represents an atom-pair and each edge represents the angle between two pairs. GeoGNN builds the message-passing function based on GIN [34] and updates the atom-bond and bond-angle graphs iteratively. Adams et al. [68] introduce a model that is invariant to rotations of rotatable bonds.

Though SE(3)-invariant has merits for GNN in various molecular property predictions, it is still limited for expressing graph representations in some aspects. One limitation is that such invariance requires the message-passing to

contain only features of distances or angles but is unable to encode directional information. Besides, for applications like force field prediction where the output is expected as a set of 3D vectors, SE(3)-invariant GNNs give the same output for molecules with different rotations in 3D. However, the force field is expected to rotate together with the rotation of the molecular system. To this end, SE(3)-equivariance is introduced which generalizes the concept of invariance. Formally, equivariance is a kind of symmetry for function. For function $\Phi : \mathcal{X} \to \mathcal{Y}$ (e.g., Φ can be a deep neural network), it is equivariant with respect to group G such that it commutes with any group actions $g \in G$ on $\mathcal{X}$ and $\mathcal{Y}$ as shown in Equation 15.

$$\rho_g^{\mathcal{Y}}(\Phi(x)) = \Phi(\rho_g^{\mathcal{X}}(x)), \forall x \in \mathcal{X}, g \in G, \tag{15}$$

where $\rho^{\mathcal{X}}(g)$ and $\rho^{\mathcal{Y}}(g)$ are the group representation of group action g on $\mathcal{X}$ and $\mathcal{Y}$ space, respectively. The group representation of group action $g \in G$ on vector space $\mathcal{V}$ is defined as:

$$\rho : G \mapsto GL(V), \tag{16}$$

such that $\rho(g_1 g_2) = \rho(g_1)\rho(g_2), \forall g_1, g_2 \in G$ (function satisfies this property is called group homomorphism). The group representation allows operating abstract mathematical object-group action, on the vector space of particular interest. More specifically, for 3D molecules' modeling, we are interested in studying the equivariance with respect to 3D rotations/translations, which corresponds to the 3D special Euclidean group, SE(3). A useful property of the equivariant functions is that composing them will yield another equivariant function, which means the whole network is equivariant if each layer is equivariant. In addition, translation invariance (invariance is also equivariance) is straightforward to achieve in most neural networks as long as they do not use global coordinates as features. Therefore in most of the models, the 3D rotation group SO(3) is the only group that requires special care.

The first major category of equivariant networks is based on irreducible representation, tensor product, and spherical harmonics. For 3D rotations, its representations are orthogonal matrices and can always be decomposed into the irreducible representation of the following form:

$$\rho(g) = Q^T \left[\bigoplus_l D^{(l)}(g) \right] Q, \tag{17}$$

where Q is an $N \times N$ orthogonal matrix (for change of basis), $\bigoplus$ is the direct sum, and $D^{(l)}(g)$ is the *Wigner D-matrices* for group action g [69]. Vectors transformed under $D^{(l)}(g)$ are called l-th order vectors, which have a length of $2l + 1$. Based on these representations, useful learnable equivariant layers can be developed. A common recipe is first building learnable filters with spherical harmonics and then composing them with input features through the tensor product, which ascribes to the fact that spherical harmonics are the equivariant basis for SO(3) and the tensor product is equivariant. Moreover, the tensor product of the irreducible representations of the vector space can be evaluated by

looking up a set of pre-calculated coefficients, which are called Clebsch-Gordon coefficients.

Tensor Field Network [10] proposes a SO(3) equivariant layer under this recipe:

$$h^l_{\text{out},i} = W^{ll} h^l_{\text{in},i} + \sum_{k \geq 0} \sum_{j \in \mathcal{N}(i)}^{n} W^{lk}(\vec{r}_j - \vec{r}_i) h^k_{\text{in},j}, \tag{18}$$

where h^l_i denotes the l-th order vector of input/output features, $W^{lk}(\vec{r}_j - \vec{r}_i)$ is a learnable filter conditioned on relative position $\vec{r}_j - \vec{r}_i$. More specifically, the learnable filter is derived by multiplying a learnable scalar with Clebsch Gordon coefficients and spherical harmonic basis function:

$$W^{lk}(\vec{r}_j - \vec{r}_i) = \sum_{J=|k-l|}^{k+l} \psi^{lk}_J(||r_{ij}||) \sum_{m=-J}^{J} Y^{(J)}_m \left(\frac{\vec{r}_j - \vec{r}_i}{||r_{ij}||} \right) Q^{lk}_{Jm}, \tag{19}$$

where $\psi^{lk}_J(\cdot)$ is a learnable function conditioned on the interatomic distance: $||r_{ij}|| = ||\vec{r}_j - \vec{r}_i||_2$, $Y^{(J)}_m$ denotes the m-th dimension of J-th spherical harmonics, and Q^{lk}_{Jm} denotes the (J, m)-th coefficient in Clebsch-Gordon matrice. The composed learnable filter $W^{lk}(\vec{r}_j - \vec{r}_i)$ with a shape $(2l+1) \times (2k+1)$ will map type-k features to type-l features. Tensor Field Network, there is an active line of works [70–72] in using operation on irreducible representation to build SE(3) equivariant network.

Another direction to build an equivariant network is to exploit the properties of vectorial features instead of leveraging tools from group representation theory. The principle is that the operation on the directional features (e.g. velocities) can only be linear. Therefore, when building the message passing layer, the operations on vector features $\vec{h}$ are restricted to: (a) Linear projection $W\vec{h}$; (b) Dot product: $< \vec{h}_1, \vec{h}_2 >$; (c) Tensor product: $\vec{h}_1 \otimes \vec{h}_2$. An example of a such recipe is PaiNN [73], where it proposes an equivariant message passing block by extending the continuous and invariant message passing layer proposed in [63]. Each layer in the PaiNN comprises two sub-blocks, a message passing block, and an update block. Inside the message passing block, the update of scalar features Δs_i are calculated as:

$$\Delta s_i = \sum_{j \in \mathcal{N}(i)} \phi_s(s_j) \odot W_s(||r_{ij}||), \tag{20}$$

where $\odot$ denotes Hadamard product, $\phi_s(\cdot)$ is a learnable atom-wise function, $W_s(||r_{ij}||)$ is the learnable continuous filter conditioned on interatomic distance, which is a linear combination of the radial basis function (PaiNN adopts the radial basis function proposed in [74]). Similarly, the update of vector features

is defined as:

$$\Delta\vec{h}_i = \sum_{j\in\mathcal{N}(i)} \vec{h}_i \odot \phi_{vv}(s_j) \odot W_{vv}(||r_{ij}||) \tag{21}$$

$$+ \sum_{j\in\mathcal{N}(i)} \phi_{vs}(s_j) W_{vs}(||r_{ij}||) \frac{\vec{r}_j - \vec{r}_i}{||r_{ij}||}, \tag{22}$$

where the first half of the equation is a convolution with respect to the vector features, and the second half is a convolution with respect to the scalar features using an equivariant filter. After the message passing block, the scalar and vector features are updated with the calculated residuals: $s_i \leftarrow s_i + \Delta s_i, \quad \vec{h}_i \leftarrow \vec{h}_i + \Delta\vec{h}_i$ and then fed into update block. Next, in the update block, the update of scalar features is calculated as:

$$\Delta s_i = a_{ss}(s_i, ||W_v\vec{h}_i||) + a_{sv}(s_i, ||W_v\vec{h}_i||) < W_u\vec{h}_i, W_v\vec{h}_i >, \tag{23}$$

where a_{ss} and a_{sv} are learnable function, and W_v, W_u are learnable linear projection matrices. And then the update of vector features is calculated as:

$$\Delta\vec{h}_i = a_{vv}(s_i, ||W_v\vec{h}_i||) W_u\vec{h}_i. \tag{24}$$

In practice, a_{ss}, a_{sv}, a_{vv} are derived from the same network a by splitting along the feature dimension of output. The scalar features and vector features are again updated with residuals. Equivariant Transformer [75] extends the above message passing layer to attention. GVP-GNN [76] leverages a similar idea with different design choices of the message passing layer. As demonstrated in Soledad Villar et al. [77], such formulation is expressive enough for approximating SE(3) equivariant functions.

It is worth noting that the above models are just non-exhaustive instances of equivariant graph neural networks. There are several other lines of works that leverage different principles such as Lie algebra [78, 79] , or other message passing protocols designed for vector features [80–83] and more general groups [84].

3 Graph Neural Networks on Molecular Applications

GNNs have been widely implemented to various applications in molecular sciences [85–87]. This section reviews the molecular applications empowered by GNNs, including molecular property prediction, molecular scoring and docking, molecular dynamics simulation, molecular optimization and generation, and others. Each subsection introduces the formulation of the problem, prevalent datasets and metrics, as well as works to solve the problem using GNNs.

3.1 Molecular Property Prediction

The most straightforward utilization of GNNs is to predict the molecular properties given the molecule graphs [86]. Such a task can be considered as a graph

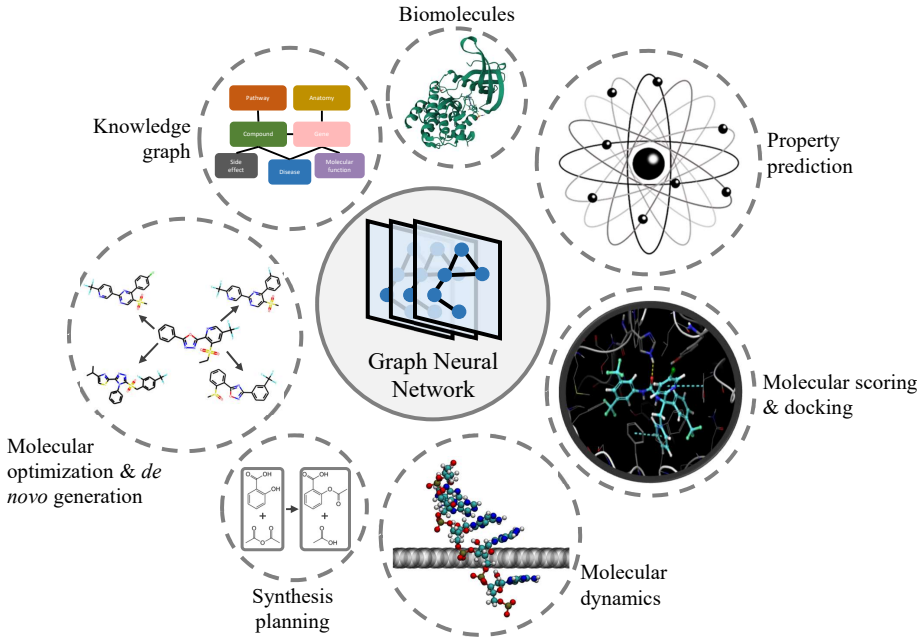


Fig. 3. Various applications of GNNs on molecular sciences.[1]

classification or regression following the empirical risk minimization in supervised learning. For a dataset of N molecule graph and property pairs $\{(\mathcal{G}_1, y_1), \ldots, (\mathcal{G}_N, y_N)\}$, the objective to optimize is given in Equation 25.

$$\min_{\theta} \sum_{i} \ell(\text{GNN}_\theta(\mathcal{G}_i), y_i), \tag{25}$$

where $\text{GNN}_\theta(\cdot)$ is a GNN model parameterized by θ to predict a certain property from an input molecule graph $\mathcal{G}_i$ and $\ell(\cdot, \cdot)$ measures the difference between the prediction and ground truth label, like cross-entropy loss for classification tasks and mean squared error for regression tasks [88–90]. Researchers have collected multiple databases containing a wide variety of molecular properties so that different GNN models can be benchmarked. Table 1 summarizes the domain, the number of compounds, the number of tasks, task type, whether containing 3D information and the sources of the popular databases for molecular property predictions. MoleculeNet [93] is a widely used benchmark for molecular property prediction built upon multiple public databases. It contains multi-level properties, including physiology, biophysics, physical chem-

[1] Resources of some fragments in Fig. 3: [91–93], `https://www.dgl.ai/news/2020/06/09/covid.html`, `https://www.gla.ac.uk/news/archiveofnews/2021/september/headline_812517_en.html`, `https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology`.

istry, and quantum mechanics. Besides, benchmarking-GNN [94] creates ZINC and AQSOL datasets for predicting the constrained and aqueous solubility. It should be pointed out that benchmarking-GNN only uses a subset of the original ZINC database (250,000 molecules) [95]. Recent efforts have focused on generating labeled molecular datasets via density functional theory (DFT) calculations. QM9, which is also included in MoleculeNet, contains approximately 134k molecules with their geometric, energetic, electronic, and thermodynamic properties. Alchemy [96] extends the previous dataset to include 12 quantum mechanical properties of 119,487 molecules with up to 14 heavy atoms. PCQM4M and its latest version PCQM4Mv2 in the OGB challenge aim at predicting the DFT-calculated HOMO-LUMO energy gap from 2D or 3D molecule graphs. ANI-1 [97] extends this concept of such datasets by including the energies of off-equilibrium conformations. Specifically, ANI-1 contains 24,687,809 conformations from 57,462 molecules. There are also works that create curated datasets for specific tasks, including quantitative structure-odor relationship (QSOR) [98], Carbon absorption [99], etc. Molecular property prediction is challenging since different datasets in it have numbers of instances of different magnitudes, from less than 200 to about 4,000,000 molecules. Also, most classification tasks in it have quite imbalanced labels, therefore, ROC-AUC and PRC-AUC are widely used instead of accuracy to measure the performance of different models. Regression benchmarks adapt either root-mean-square error (RMSE) or mean absolute error (MAE) as evaluation metrics. Besides, Hu et al. [93] introduce scaffold splitting to separate structurally different molecules into training, validation, and test subsets. This strategy provides a more challenging yet realistic setting than random splitting and has been leveraged in many works on molecular property prediction [56–59, 67, 100].

Most of the GNN architectures introduced in Section 1 and 2 can be leveraged on molecular property predictions. Apart from those models, a wide range of GNN variants have been developed for the application. Wieder et al. [86] provide a comprehensive survey for the molecular property predictions via GNNs till the year 2020. Early works utilize spectral GNNs for molecular property prediction [124–127]. Since the proposal of GCN [31], a simple yet generic algorithm, many GNNs have been built upon it for molecular property predictions [128–132]. Some works have also explored using RNNs for aggregation in this area [133–135]. After the formalization of the message-passing framework by Gilmer et al. [22], researchers have built GNNs systematically following the insight to predict a wide range of molecular properties [57, 136–138].

Recently, there are emerging researches that leverage DNNs, especially SE(3)-invariant or SE(3)-equivariant GNNs with 3D geometric information, for quantum mechanics (QM) predictions [139–141]. Many works are evaluated on QM9. Early SE(3)-invariant GNNs, including SchNet [63], PhysNet [64], HIP-NN [142], MGCN [143], DimeNet [65], DimeNet++ [66], Cormorant [144], SphereNet [145], are built upon interatomic distances and angles. Some works have leveraged orbital information to better predict QM properties [146–149]. GNNs built upon SE(3)-equivariant operations have also been widely applied to this applica-

Domain	Dataset	# Molecules	# Tasks	Task type	3D	Source
Physiology	MUTAG	188	1	Classification	No	[101]
	SIDER	1,427	27	Classification	No	[93, 102]
	ClinTox	1,478	2	Classification	No	[93, 103]
	BBBP	2,039	1	Classification	No	[93, 104]
	NCI1	4,110	1	Classification	No	[105]
	Tox21	7,831	12	Classification	No	[93, 106]
	ToxCast	8,575	617	Classification	No	[93, 107]
Biophysics	BACE	1,513	1	Classification	No	[93, 108]
	CCRF-CEM	3,047	1	Regression	No	[109]
	PC-3	4,294	1	Regression	No	[109]
	PDBBind	19,443	1	Regression	Yes	[93, 110, 111]
	HIV	41,127	1	Classification	No	[93, 112]
	MUV	93,087	17	Classification	No	[93, 113]
	PCBA	437,929	128	Classification	No	[93, 114]
Physical Chemistry	FreeSolv	642	1	Regression	No	[93, 115]
	ESOL	1,128	1	Regression	No	[93, 116]
	Lipophilicity	4,200	1	Regression	No	[93, 117]
	AQSOL	9,982	1	Regression	No	[94, 118]
	ZINC	12,000	1	Regression	No	[94, 95]
Quantum Mechanics	QM7	7,165	1	Regression	Yes	[93, 119]
	QM7b	7,211	14	Regression	Yes	[93, 120]
	QM8	21,786	12	Regression	Yes	[93, 121]
	QM9	133,885	12	Regression	Yes	[93, 122]
	Alchemy	119,487	12	Regression	Yes	[96]
	PCQM4M	3,803,453	1	Regression	No	[123]
	PCQM4Mv2	3,378,606	1	Regression	Yes	[123]
	ANI-1	24,687,809 (conf.)	1	Regression	Yes	[97]

Table 1. Summary of benchmarks for molecular property predictions.

tion. Examples include SE(3)-Transformer [70], E(n)-GNN [150], LieTransformer [151], L0/L1Net [152], GemNet [153], PaiNN [154], and TorchMD-Net [75]. Many SE(3)-equivariant GNNs have also been benchmarked MD17 [155] which contains energy and force field predictions from molecular dynamics simulations of eight molecules. We will discuss GNN applications on MD simulations in Section 3.3 extensively. Recent works have investigated *ab-initio* solution of the Schrödinger equation to acquire fundamental QM properties equipped with GNNs. Hermann et al. [156] propose PauliNet, a GNN-based wave function ansatz. PauliNet adapts SchNet as the GNN backbone and exploits multireference Hartree-Fock solution as the optimization start point. The whole framework is trained using variational quantum Monte Carlo (VMC). Gao et al. [157] introduce a framework combining a GNN and a neural wave function to solve the Schrödinger equation for multiple geometries via the VMC method simultaneously.

3.2 Molecular Scoring and Docking

Molecular scoring mainly refers to the prediction of pharmacological activity of a molecule candidate, which is essential in *in silico* drug discovery [87]. Such a

method is widely applied in the virtual screening of drug candidates. It can also be leveraged to narrow down the search space for *de novo* molecular generation. There are two major categories of molecular scoring: ligand-based scoring and structure-based scoring. The ligand-based scoring requires no information about the receptor in pharmacological activities. Such a setting can be considered as a special case of molecular property predictions that model the quantitative structure-activity relationships (QSAR), like drug-likeness, toxicity, solubility, etc. On the contrary, structure-based scoring is based on the structure of the biological target (usually a protein) of a molecule. Such a strategy predicts the drug-target interactions given the drug candidates and the target receptors. Molecular docking predicts the preferred pose of a molecule when bound with a binding site to form a stable protein-ligand complex. It is one of the most popular techniques in structure-based drug discovery. Several works leverage 3D CNN to model the protein-ligand complex [158–162]. However, 3D CNN requires voxelization of 3D space which adds burdens to the memory. Moreover, such a voxelization loses resolution of the exact pose in 3D and lacks explicit modeling of interatomic interactions. This chapter reviews the structure-based molecular scoring and molecular docking empowered by GNNs.

PDBBind [110, 163, 164] collects biomolecular complexes with their experimental binding affinity data and is widely used in structure-based scoring with deep learning as a regression task. DUD-E [165] is another dataset that contains 102 targets across different protein families. For each target, both positive (active) and negative (decoy) ligands are provided, which are formalized as classification tasks. Also, BindingDB [166, 167] is a public database of measured binding affinities of proteins and ligands. Drugbank [168, 169], a public online database, that collects FDA-approved drugs with drug target and drug action information. ChEMBL [117] also provides plenty of protein-ligand information including chemical compound and activity assay data. Several works have also created curated databases [170, 171].

Structure-based scoring aims at predicting the binding affinity of drug-target interactions (DTIs) [172]. The key to designing GNN models for structure-based molecular scoring is how to integrate the information of the protein and/or the binding pocket to the ligand [173–175]. Feinberg et al. [176] introduce PotentialNet, a pioneering GNN framework that models the protein-ligand complexes. PotentialNet takes two stages in message-passing: (1) intra-molecular message-passing based only on covalent bonds, and (2) intra- and inter-molecular message-passing based on Euclidean distance and bonds. Gomes et al. [177] introduce a framework contain three parameter-sharing Siamese GNNs that encode ligand, protein, and protein-ligand complex, to features G_{ligand}, G_{protein}, and G_{complex}, respectively. The prediction of binding affinity then utilizes $\Delta G = G_{\text{complex}} - G_{\text{ligand}} - G_{\text{protein}}$. Lim et al. [178] improve the framework by conducting intra-molecular message passing and inter-molecular massage-passing simultaneously. The model utilizes the subtraction between the features of a target protein and the ligand in their complex to predict the binding affinity. Such a strategy that models the interactions within the ligands and interactions

between ligands and targets is widely used in later works. InteractionGraphNet [179] by Jiang et al. models the intra- and inter-molecular interactions sequentially with two independent GNN models. Morrone et al. [180] adapt a dual-GNN framework to encode the ligand graph and protein-ligand interactions separately. The output of the two GNNs is concatenated and fed to an MLP to predict the binding affinity of the complex. Son et al. [181] propose GraphBAR, a framework containing multiple graphs whose adjacency matrices cover neighbors within different distance cutoffs. Knutson et al. [182] propose two parallel GNNs with one incorporating domain knowledge of proteins and ligands while the other learning interactions with no domain prior. The works listed above are built upon protein-ligand complexes. However, these models may suffer to make accurate predictions for those complexes without experimental measurements. To this end, Torng and Altman [183] propose to encode protein pocket graphs and 2D molecule graphs independently and combine the output features to predict the DTIs. Recent works have introduced language models to encode protein information in predicting DTIs [170, 184–188].

Unlike structure-based scoring which predicts binding affinity directly, molecular docking focuses on predicting the posture of a protein-ligand complex. Conventional docking contains a search module to generate massive potential protein-ligand binding poses and a scoring module to evaluate the interaction of binding poses [189–191]. Such methods rely on heavy candidate sampling and usually involve empirical scoring functions, thus may be time-consuming and inaccurate [192, 193]. Recently, there are emerging works that apply GNNs for molecular docking [194–197]. Jiang et al. [198] develop a GNN to refine the initial docking pose from conventional docking software. The GNN can be run multiple times to gradually get optimal predictions. Mendez et al. [199] report DeepDock, which first learns a statistical potential based on distance likelihood and then samples ligand conformations based on learned potential. Specifically, DeepDock models statistical potentials on torsion angles to generate favorable molecular conformations in the binding site [200]. However, these methods assume prior knowledge of the binding site for the ligands. Some works have investigated predicting protein binding sites using GNNs [11, 201–204]. To this end, GNN methods have been developed to both locate the binding site and predict the preferred binding conformations. EquiBind [205] by Stärk et al. adapts SE(3) GNN to predict the binding site and applies constraints that only allow change of torsion angles while keeping bond lengths and angles fixed. TANKBind [206] by Lu et al. incorporates trigonometry constraints to GNNs which avoid unrealistic conformations like atom overlapping. It also leverages contrastive learning to build the energy landscape for the inter-molecular interaction of different binding sites.

3.3 Molecular Dynamics Simulation

Molecular Dynamics (MD) has a wide range of applications in material science, chemistry, and biophysics [207–209]. It provides a numerical way to study and predict intricate molecular systems. In essence, MD simulation calculates atomic

forces and then updates the system states with a discretized equation of motion. The forces can either be modeled by *ab initio* approaches (AIMD) like density functional theory (DFT) [210] that considers the electronic structure of atoms or through empirical potentials which bypasses electronic structures [211]. AIMD is highly accurate but its computation is prohibitively expensive, which limits its scope of application. On the contrary, empirical force fields are much more efficient, yet with worse accuracy and limited generalizability. The major limitation of MD simulations using empirical force fields stems from the difficulty in how to describe complex interatomic potentials accurately with the appropriate functional form given the diverse types of interactions in the system. In this regard, neural networks have the potential to close the accuracy gap between empirical potentials and AIMD with its function approximation capability [212–216].

Depending on how the environment of the atom is described, neural networks can be categorized into two broad categories. The first kind of approach relies on hand-designed featurization, which builds tailored descriptors that exploit domain knowledge [139, 212, 217–220]. One of the first examples is Behler-Parrinello Neural Networks (BPNN) [221–223], which proposes atom-centered symmetry functions (ACSFs) for describing the local neighborhood around an atom. Graph neural networks, on the other hand, provide a framework to directly learn the atomic representations from the raw atomic coordinates and low-level atomic features, offering an alternative to the manually tailored atomic descriptors [5, 22, 62–64, 73, 74].

Based on the above frameworks, neural networks can be used to address the efficiency-accuracy tradeoff in MD from the following aspects. The first aspect is to use neural networks to learn the forcefield (i.e. interatomic potential) of a molecular system. In this case, neural networks trained on data with ab initio level accuracy can serve as fast and efficient surrogate models for AIMD. The output of the network can either be the potential energy surface (PES) or atomic forces, in turn forming an energy-based model (also known as neural network potentials) [62, 63, 74, 139, 155, 223–227] or force-based model [228–231]. These two variants are similar in principle (Chmiela et al. [155] demonstrate that force-based model essentially learns a linearization of the PES), but force-based models are more accurate at predicting forces in practice and can bypass the process of calculating energy gradients. However, force-based models are generally not energy conserving and thus can generate unphysical predictions. Another perspective is learning the pattern of molecular trajectories instead of the dynamics, which treats the simulation as a sampling problem. Here neural networks are used as a probabilistic generative model [232–235] to model the distribution $p(x_{t+1}|x_t)$ with x_t denotes the state of the system at time t. Since this paradigm no longer depends on dynamics to update the system, it does not suffer from the truncation error that arises from the discretization error of the equation of motion and thus can adopt a very large time step size (e.g. at the scale of a nanosecond). The downside is such a method can lead to unphysical prediction and cannot be used to study properties related to energy and dynamics.

3.4 Molecular Optimization and Generation

Efficient and effective molecular generation is of crucial importance in practice [236]. Especially, in the pharmaceutical industry, the discovery of new drugs is a long and expensive process that costs more than $2.5 billion and 10–15 years on average [237]. Therefore. it is appealing to develop techniques that automatically and effectively generate plausible molecule candidates. The deep generative models, which learn to approximate the distribution of observed data, have been leveraged for molecule graph optimization and *de novo* generation [87, 238–242]. The optimization of the molecule usually starts with a hit compound and manipulates the graph to achieve better target properties like toxicity, drug-likeness, binding affinity, etc. The *de novo* molecule generative models generate novel molecules from scratch or conditioned on desired properties or specific fragments. In this case, molecule optimization can be considered as a conditional generation problem conditioned on the hit molecule. Thus, we review these two applications together in this section. There are two major frameworks in molecule graph generation: (1) fragment-based generative models and (2) molecule-based generative models [87, 243]. The fragment-based models generate molecule graph iteratively, where at each step the model choose an action, including adding, deleting, or editing one or multiple atoms, bonds, or functional groups. Such a fragment-based framework can be incorporated with reinforcement learning (DRL) that defines a Markov decision process (MDP) and learns the best generative policy by maximizing the expected accumulated rewards. It can also be implemented with autoregressive models like RNNs, which determine the action based on statuses at previous steps [244, 245]. The molecule-based model, on the other hand, generates all the attributes (i.e., atoms and bonds) of a molecule at once. Many generative methods equipped with GNNs fall into this category, including variational autoencoder (VAE) [246], generative adversarial network (GAN) [247], flow-based generative model [248], and score-based or diffusion generative model [249, 250]. VAE usually contains an encoder and a decoder that are trained to maximize the evidence lower bound concerning the log-likelihood of training data. GAN introduces two jointly trained components: a generator and a discriminator, where the generator generates samples while the discriminator attempts to distinguish between samples from the training data and those from the generator. Flow-based models are built upon a sequence of invertible operations that directly models the likelihood of the observed data. Score-based or diffusion model defines a Markov chain that adds random noise to data at each step and learns to denoise from the noise to recover the data. The following part in this section introduces molecule graph generative methods in detail.

Metrics and Benchmarks To measure the performance of deep generative models on molecule graphs, one needs to define comprehensive metrics and benchmarks [251, 252]. Simple but effective metrics include validity which assesses whether the generated molecules are valid, uniqueness/diversity which evaluates if a model generates a different molecule at each sampling, and novelty which measures whether generated molecules exist in the training set. Besides,

Preuer et al. propose the Fréchet ChemNet distance (FCD) [253] following the widely-used Fréchet Inception distance (FID) [254] in image synthesis, which is built upon the difference of hidden representations from ChemNet, a deep neural network, between generated molecules and those in the training set. Kullback-Leibler (KL) divergence is also used to measure whether the generative model approximates the distribution of the training set. In conditional generation or optimization, certain properties are used to evaluate the generative models. Many works build evaluation metrics on drug-likeness [255]. For example, the synthetic accessibility score (SAS) describes the ease of synthesis of molecules based on fragment contributions and a complexity penalty [256], the octanol-water partition coefficient $\log P$ characterizes the drug-likeness of a molecule, the penalized $\log P$ is the subtraction of original $\log P$ and SAS, and quantitative estimate of drug-likeness (QED) applies desirability functions which provides a multicriteria metric to assess drug-likeness [257]. Maximum mean discrepancy (MMD) [258, 259] is another metric to evaluate the generated graphs. MMD is employed to measure the distribution difference between generated molecule graphs and training set on degrees, clustering coefficients, orbit counts, as well as bond lengths of different types. For candidate drug optimization or generative, the binding affinity of molecules with respect to the target protein pocket computed by molecular docking tools [260, 261] or molecular dynamic (MD) simulations [262–264] are also used as the objective [265, 266]. Energy-related properties, like HOMO-LUMO gap and dipole moment, have also been leveraged as the optimization or generation targets [267]. Additionally, Gao et al. [268] point out that sampling efficiency should be another important consideration in real molecular generation applications.

Several works have contributed to creating molecular generation benchmarks and databases. GuacaMol [251] by Brown et al. employ a standardized subset from ChEMBL database [117]. It includes validity, uniqueness, novelty, FCD, and KL divergence as the evaluation measurements. MOSES [252] by Polykovskiy et al. contains 1,936,962 molecular structures selected from ZINC [95] and is split into the training, test, and scaffold test sets containing approximately 1.6M, 176k, and 176k structure, respectively. Aside from the metrics included on GuacaMol, MOSES implements the similarity of fragments and scaffolds and properties distribution to evaluate the generative models. Besides, ZINC [95], ChEMBL [117], GDB databases [121, 269] are also used in training 2D molecule graph generative models. QM9 [122] which contains more than 130K 3D molecular structures, is also widely used for 3D molecular generation. Recently, GEOM [270] by Axelrod et al. introduces GEOM-QM9, an extension to QM9, containing multiple conformations for most molecules, and GEOM-Drugs containing 304,466 drug-like species up to a maximum of 91 heavy atoms.

Fragment-based Generative Model Fragment-based generative model modifies a molecule graph by adding, removing, or substituting a fragment (i.e., an atom or a motif) sequentially [271]. Such a framework can be directly modeled as an RL problem [265, 272–276]. At each time step t, an agent receives a reward

r_t and predicts an action $a_t \in \mathcal{A}$ given the current state $s_t \in \mathcal{S}$, and the next state $s_{t+1} \in \mathcal{S}$ is depended solely on s_t and a_t following the MDP setting. Here, $\mathcal{A}$ and $\mathcal{S}$ denote the action and state space, respectively. In this case, the state is an incomplete molecule graph, the action is what fragment to add, remove, or substitute, and the agent is usually modeled by a deep neural network, including GNNs. Besides, the reward r_t evaluates how well the molecule is generated which could be a chemical metric or an output from property prediction ML models that approximate empirical measurements. The optimization objective of an RL agent (Equation 26) is to maximize the expected accumulated return R_t of a policy defined by the agent.

$$\mathbb{E}_\pi[R_t] = \mathbb{E}_\pi \left[\sum_t^T \gamma^t r_t \right], \tag{26}$$

where $\gamma \in (0, 1]$ is a discount rate and T is the maximal length of a trajectory. RL provides a generic framework for molecular design of different targets [277]. You and Liu et al. [278] propose a graph convolutional policy network (GCPN) that successively constructs a molecule by adding an atom, a substructure, or a bond, and is trained via the policy gradient algorithm. It built its agent on GCN [31]. Experiments show that GCPN can generate molecules with optimized or targeted penalized $\log P$ or QED properties. Jin et al. [279] present RationaleRL composed of rationale extraction, graph completion, and rationale distribution. RationaleRL is evaluated on GNK3β and JNK3 that measures the inhibition against glycogen synthase kinase-3 c-Jun N-terminal kinase-3 [280]. It also measures the validity, diversity, and novelty of the generative model. DeepGraphMolGen proposed by Khemchandani et al. [281] adapts GNNs for action prediction as well as property prediction in an RL setting. It designs a multi-objective reward function containing SAS, QED, and predicted binding affinity at the dopamine and norepinephrine transporters. Besides RL, autoregressive models can be incorporated with fragment-based molecule graph generation [244, 282]. GraphINVENT by Mercado et al. [282] leverages a tiered GNN architecture to generate a single bond at each step. Podda et al. [283] and Chen et al. [284] develop autoregressive generative models that operate on the fragments. Lim et al. [285] propose to extend a given molecule scaffold by sequentially adding nodes and edges, which guarantees the generated molecules preserve the certain scaffold. Xie et al. [286] propose a framework named MARS which combines molecule graph editing with MCMC sampling. The method employs a GNN to sample fragment-editing actions at each step adaptively. Shi et al. [287] introduce GraphAF built upon an autoregressive normalized flow-based generative model that adds nodes and edges to a molecule graph sequentially. GraphAF can also be fine-tuned with RL for molecular optimization of certain properties. Following the normalized flow model, GraphDF by Luo et al. [288] samples discrete latent variables which are mapped to additional nodes and edges via invertible modulo shift transforms.

Fragment-based generative models introduced so far have demonstrated the effectiveness of generating valid, unique, and diverse molecules as well as gen-

erating or optimizing towards desired SAS, $\log P$, and QED. However, these models only adapt 2D topological information without 3D geometric features. Many recent works have investigated generative models on 3D molecular structures as many properties, like energy and protein-ligand binding, are related to 3D geometries. Gebauer et al. [289] introduce G-SchNet, an autoregressive generative model for 3D molecular structure generation via placing atoms in 3D Euclidean space one by one. Gebauer et al. further introduce cG-SchNet [290], a conditional version of G-SchNet to generate molecules with certain motifs or low-energy (e.g., small HOMO-LUMO gap). Simm et al. [291] propose an RL formulation that adds atoms in 3D euclidean space sequentially and designs a reward function based on the fast quantum-chemical calculated electronic energy. The team also introduce MolGym, an RL environment comprising several molecular design tasks along with baseline models. Flam-Shepherd et al. [292] further introduce a 3D molecule generative RL framework to add fragments instead of single atoms at each step, which is more efficient in creating larger and complex molecules. Luo et al. [293] present G-SphereNet follows the flow-based generation while determining atoms in 3D space by predicting distances, angles, and torsion to preserve equivariance. Besides, researchers have investigated generating 3D molecules given a designated protein binding site. Docking scores along with other generic metrics are utilized to evaluate the deep generative models. Luo et al. [294] and GraphBP by Liu et al. [295] develop 3D autoregressive generative methods to sample the type and position of a new atom sequentially in the 3D binding pocket. Powers et al. [296] propose a molecular optimization method that expands a small fragment into a larger molecule within a protein site. At each step, a GNN trained by imitation learning selects the connecting point as well as the type and dihedral angle of a fragment to be connected. To preserve the equivariance in the 3D molecular generation, many works adopt a local spherical coordinate system in the generation process [289–291, 294, 295]. Besides, Imrie et al. [297] introduce DeLinker which generates linkers given two fragments in 3D Euclidean space.

Molecule-based Generative model Unlike fragment-based generative models, molecule-based generative models create the node features, edge features, and adjacency matrix of a molecule graph simultaneously. VAE, a popular generative model, has been leveraged for molecule-based generation [255, 298]. A VAE is composed of two sub-models: an encoder that encodes the input into a latent feature domain, and a decoder that maps the feature back to the input [246, 299]. The loss for training a VAE is given in Equation 27.

$$\ell = -\mathbb{E}_{z\sim q_\theta(z|x)}[\log p_\phi(x'|z)] + \mathrm{KL}(q_\theta(z|x)\|p(z)), \tag{27}$$

where x is a input data, z is the latent vector, q_θ is the encoder, and p_ϕ is the decoder. The first term measures how well the model reconstructs the data and the second term regularizes the latent vector to be similar to a prior Gaussian distribution $p(z)$ through a KL divergence. Such a framework has been leveraged to generate molecule graphs [299–301]. Simonovsky et al. [302] propose Graph-