ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
DATA SCIENCE AND COMPUTATION
Ciclo XXXVI

Settore Concorsuale: 09/H1 – Sistemi di elaborazione delle informazioni

Settore Scientifico Disciplinare: IINF-05/A – Sistemi di elaborazione delle informazioni

Graph neural network methods for representation and generation in drug discovery

Presentata da: Carlo Abate

Coordinatore Dottorato
Prof. Daniele Bonacorsi

Supervisore
Prof. Andrea Cavalli

Co-supervisore
Dr. Sergio Decherchi

Esame Finale Anno 2025

Contents

Abstract

Drug discovery is a time-consuming and expensive process, often spanning over a decade and costing billions of dollars. This thesis advances graph-based machine learning approaches to accelerate this process, making three main contributions. First, we provide a comprehensive review of graph neural networks for conditional molecular generation, establishing a framework for understanding and comparing different methods. Building on these insights, we introduce AMCG (Atomic-Molecular Conditional Generator), a novel generative framework that achieves state-of-the-art performance while offering one-shot generation capability and effective property optimization via gradient ascent. Motivated by the heterophilic nature of molecular graphs — where connected atoms often have dissimilar features — we then develop MaxCutPool, a differentiable graph pooling technique based on the `MAXCUT` problem. By combining graph-theoretical principles with deep learning, MaxCutPool demonstrates superior performance on heterophilic graphs while remaining competitive on standard benchmarks and maintaining computational efficiency. Together, these contributions advance both the theoretical foundations of graph representation learning and provide practical tools for accelerating drug discovery.

Chapter 1

Introduction

1.1 The landscape of drug discovery

Drug discovery is a complex, multidisciplinary task aimed at identifying and developing new therapeutic compounds. This process is characterized by its resource requirements in terms of time and money, often spanning over a decade and costing billions of dollars from initial concept to market approval [1, 2]. One of the key challenges the pharmaceutical industry faces is to reduce such costs. The difficulty of this operation is to be found in the inherent complexity of disease mechanisms, the necessity to meet regulatory requirements, and the need to develop drugs for increasingly specific patient populations [3]. The traditional drug discovery pipeline consists of several key stages [4]: the initial step consists of identifying a biological target that is believed to play a central role in a disease when modulated by a drug. This target can be a protein, a gene, RNA, or any biological molecule. Once a target is validated, the following step is to find initial *hits*, i.e. compounds that interact with the target in a desirable way. This process traditionally involves screening techniques – *in vitro* (see Section 1.2.1) or *in silico* (see Section 1.2.2) – of large chemical libraries. Following hit discovery, the identified compounds are optimized to improve their efficacy, selectivity, and pharmacokinetic properties. This phase is called lead optimization and involves cycles of synthesis and testing. Once a lead compound has been optimized, it enters preclinical development, where its safety,

efficacy, and pharmacokinetics are tested *in vitro* (in cell cultures) and *in vivo* (in animal models). The resulting compounds undergo a clinical trials stage, which is one of the most critical phases of drug development since it involves testing the drug on humans and where, often, potential drugs fail for lack of efficacy and/or side effects. Upon successful clinical trials, an NDA is submitted to regulatory bodies such as the FDA (U.S. Food and Drug Administration) or EMA (European Medicines Agency). This application includes all data from preclinical and clinical studies, as well as information on manufacturing, labeling, and proposed marketing strategies.

During this long and complex development process a significant amount of information is produced, additionally there are many stages at which the pipeline can fail. Due to the complexity of the process and the wealth of produced information [5], computational methods, including deep learning based ones, can support the discovery process.

1.2 Classical approaches in hit discovery

Over the years, several approaches have been developed to reduce the costs of the initial phases of the drug discovery pipeline.

1.2.1 High-throughput screening

High-throughput screening (HTS) has been a cornerstone of drug discovery for decades [6]. It typically utilizes automated equipment to perform testing on thousands or even millions of compounds in a relatively short time. The advantages of HTS include the ability to quickly test a large number of compounds and to identify good potential drug candidates. Furthermore, as a by-product, it generates extensive structure-activity relationship (SAR) data that can be of relevant interest. The disadvantages, on the other hand, include the high costs associated with the necessary equipment and compound libraries, often induced by low hit rates – typically less than 0.1% [7]. Another problem is given by the limited ability that this technique has in exploring the full chemical space. Despite these chal-

lenges, HTS has led to the discovery of numerous drugs and continues to be an important tool in many drug discovery programs [8, 9]. Recent advancements in miniaturization, robotics, and data analysis have further improved the efficiency and effectiveness of HTS [10].

1.2.2 Virtual screening

Virtual screening (VS) emerged as a computational alternative to HTS, allowing researchers to evaluate large libraries of compounds *in silico* [11]. This approach leverages computational power to predict the likelihood of compounds binding to a target or exhibiting desired properties, thereby prioritizing specific compounds for experimental testing. VS can be broadly categorized into two main approaches: structure-based and ligand-based. Structure-based VS utilizes the three-dimensional structure of the target protein to dock and score potential ligands [12, 13], and has been particularly successful in identifying hits for targets with well-characterized binding sites, such as enzymes and receptors [14]. Ligand-based VS relies on the principle that molecules with similar structures or properties are likely to have similar biological activities [15]. This approach is particularly useful when the three-dimensional structure of the target is unknown. Common ligand-based VS techniques include:

- pharmacophore modeling [16];
- quantitative structure-activity relationship (QSAR) analysis [17, 18];
- shape-based similarity searching [19, 20];
- machine learning (ML) models trained on known active compounds [21].

VS has proven to be a cost-effective method for identifying promising lead compounds, often complementing and enhancing traditional HTS approaches [22].

1.3 The emergence of deep learning in drug discovery

In recent years, the field of drug discovery has experienced a transformative shift with the emergence of deep learning (DL) techniques [23]. Such techniques can be utilized to accelerate many stages of the drug discovery process, ranging from target identification and hit discovery to lead optimization and preclinical testing [24], by leveraging vast amounts of data and increasingly complex algorithms. DL offers several key advantages in drug discovery: the ability to learn complex, non-linear relationships from large datasets [25, 26], automated feature extraction, reducing the need for manual feature engineering [23, 27], potential for end-to-end learning, integrating multiple steps of the drug discovery process [5, 24], capability to generate novel molecular structures [28, 29], and improved predictive power for molecular properties and activities [30, 31]. These capabilities have led to applications across the drug discovery pipeline. For instance, deep learning models have been used to analyze large-scale genomic and proteomic data to identify potential drug targets [32], and DL-based VS methods have shown improved performance in identifying active compounds compared to traditional approaches [33].

Another important aspect of molecular modeling in drug discovery is the accurate representation of potential energy. The potential energy of a molecular system describes the energy landscape that governs molecular interactions and conformations. DL approaches have shown promise in predicting and modeling potential energy surfaces more accurately and efficiently than traditional methods [34, 35]. For instance, *neural network potentials* can capture complex many-body interactions that are challenging for classical force fields [36]. These potentials can be trained on high-level quantum mechanical calculations and then used to predict energies and forces for larger systems or longer timescales, bridging the gap between quantum accuracy and classical efficiency [37]. This capability is particularly valuable in drug discovery for tasks such as protein-ligand binding affinity prediction, conformational analysis, and molecular dynamics (MD) simulations [38].

1.3.1 Graph neural networks in drug discovery

Graph neural networks (GNNs) are a class of neural networks that operate directly on graph-structured data. Their peculiar mechanism of action, *message-passing*, makes them particularly well-suited for representing and analyzing molecules. The use of GNNs in drug discovery offers several key advantages over traditional methods and other DL approaches: first and foremost, molecules are inherently graph-based entities, where atoms are nodes and bonds are edges. GNNs naturally align with this representation, allowing the model to directly learn from the structural and relational information encoded in molecular graphs. Unlike traditional vector-based methods, which may lose some of this relational context, GNNs preserve the molecular topology, capturing both the connectivity and the nature of the interactions between different parts of a molecule [39]. Unlike fixed-dimensional input methods, GNNs can adapt to varying graph sizes and can process molecules with different numbers of atoms and bonds without the need of extensive preprocessing [40]. Furthermore, the features learnt by GNNs can often be directly interpreted in terms of molecular substructures or motifs, such as functional groups or specific bonding patterns. This interpretability is a significant advantage, as it allows researchers to gain insights into which parts of a molecule contribute most to its predicted properties or activities [41]. Another promising aspect of GNNs is their potential for transfer learning. In drug discovery, the ability to transfer knowledge from one task to another – such as from predicting molecular properties to predicting biological activity – can save significant time and resources. GNNs, trained on large datasets to learn generalizable chemical features, can be fine-tuned for specific tasks, enhancing performance on related problems and enabling more efficient drug discovery processes [42, 43].

The versatility of GNNs has led to their application across a wide range of tasks in drug discovery: they have been successfully employed to predict various molecular properties, such as solubility, toxicity, and bioavailability [44], becoming the *de facto* standard architecture utilized for such task. Another significant application of GNNs is in predicting the binding affinity between proteins and potential ligands [45]. Accurately predicting binding affinity is crucial for identifying promising

drug candidates, and GNNs have shown promise in this area by effectively modeling the interactions between molecules and target proteins. In addition to molecular property and activity prediction, GNNs have been applied to predict chemical reactions and assist in retrosynthetic analysis [46]. These applications help chemists understand how to synthesize target molecules, further accelerating the drug development process. GNNs are also being used to generate new molecular structures from scratch, a process known as *de novo* molecular design [47]. By learning the principles of chemical bonding and molecular stability, GNNs can suggest novel compounds that are likely to exhibit desired behaviours, thus expanding the chemical space available for drug discovery. A graph-based AI model [48] demonstrated its effectiveness in de novo drug design by creating a JAK1 inhibitor that was later independently patented (WO2020182159A1), underscoring the model's ability to generate novel and biologically active molecules. Lastly, GNN approaches have been successfully applied to drug repurposing efforts, leveraging large-scale biomedical data to identify new indications for existing drugs [49, 50]. These models can integrate diverse data types, including drug structures, gene expression profiles, and clinical data, to find novel therapeutic applications for known compounds.

Despite these advances, several key challenges remain in the application of GNNs to drug discovery. First, many existing generative models lack explicit control over molecular properties and atomic composition, limiting their practical utility in targeted drug design. Second, current approaches often struggle with the inherent heterophilic nature of molecular graphs, where connected atoms typically have different properties. Third, the interpretability of these models remains a significant challenge, particularly in understanding how structural modifications affect molecular properties.

1.4 Thesis structure

This thesis contributes to both the theoretical foundations of GNNs and their practical applications in de novo molecular design. The content of this work is largely based on the works [P1, P4, P2, P3] and is organized as follows:

Table 1.1: Works that form the basis of this thesis

Chapter	Publication	Description	Status
2	[P1]	Comprehensive review of GNNs for conditional molecular design	Published
3	[P2]	AMCG: A novel GNN-based generative model for molecular graphs	Published
4	[P3]	MaxCutPool: A new graph pooling technique for heterophilic graphs	Accepted
5	[P4]	Review of ligandability and druggability assessment methods	Published

- Chapter 2 presents a comprehensive literature review of GNNs for conditional molecular design. It provides an overview of the current state-of-the-art in graph-based approaches to address this task. The chapter also sets the mathematical foundation for the subsequent technical contributions. This chapter is based on our comprehensive review of GNNs for conditional molecular design [P1].

- Chapter 3 introduces AMCG (Atomic-Molecular Conditional Generator), a novel GNN-based generative model for molecular graphs. Here, we detail the theoretical foundations of the model, its architectural features and training process, and present comprehensive experimental results demonstrating its effectiveness in both unconditional and conditional molecular generation tasks. The model presented in this chapter was originally introduced in our publication [P2].

- Chapter 4 introduces MaxCutPool, a new graph pooling technique. This chapter is more theoretical, addressing the need for specialized pooling methods when dealing with heterophilic graphs. It covers the mathematical foundations of the `MAXCUT` problem at the basis of our method, describes the MaxCutPool pooling layer in detail, and provides extensive experimental evaluations. MaxCutPool technique discussed in this chapter was first proposed in our work [P3].

- Chapter 5 explores complementary projects in drug discovery. This chapter begins with a review of ligandability and druggability assessment methods,

covering various machine learning approaches, datasets, and benchmarks used in this field [P4]. It also describes the ongoing development of a web server for protein pocket detection.

- Chapter 6 synthesizes the thesis contributions and discusses future directions for integrating these components into a comprehensive de novo drug design pipeline. It explores the potential integration of MaxCutPool into the AMCG framework and outlines a vision for a more holistic approach to drug discovery.

Chapter 2

Graph neural networks for molecular design: a literature review

2.1 Introduction

The field of machine learning has witnessed a significant advancement since the advent of GNNs. These powerful models have shown great capability for handling structured data, making them a valuable tool in drug discovery and molecular design [47]. GNNs extend the concept of traditional, feed-forward neural networks to graph domains, allowing for the processing of data that can be represented as nodes connected by edges.

Introduced in 2005 [51] and further developed in 2009 [52], the original Graph Neural Network (GNN) model laid the foundation for processing graph-structured data using neural networks. However, it wasn't until the mid-2010s that GNNs became well established in the machine learning community. This surge in popularity coincided with the increasing availability of graph-structured datasets and computational resources, marking a new era in the analysis of complex, interconnected data.

Being the main focus of this thesis on generative models for molecular graphs, we will direct our attention specifically to the development and application of this family of methodologies. As the field has evolved, in fact, a diverse set of approaches has emerged, each with its own strengths and challenges. To provide a comprehensive overview of this landscape, we present a taxonomical forest of generative models for drug design (Figure 2.1).

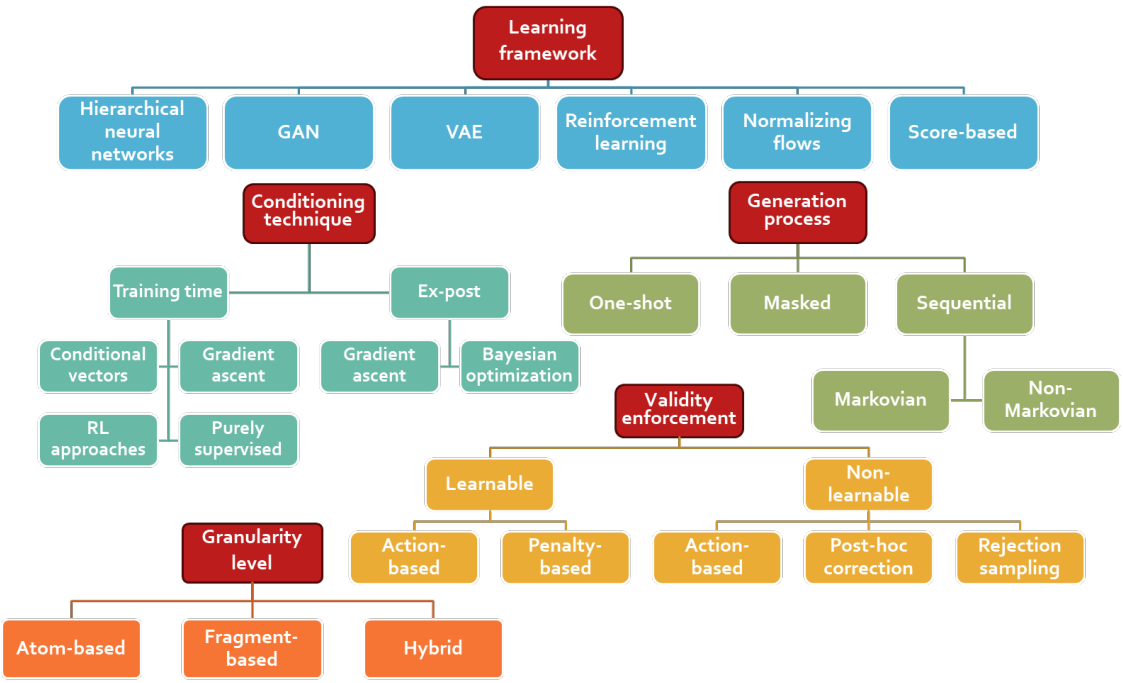

Figure 2.1: Taxonomy of generative models for de novo drug design, from general choices such as the employed learning framework, the conditioning technique and the generation process modeling to domain-specific ones, such as the validity enforcement techniques and the granularity level.

In this chapter we provide a comprehensive review of GNNs for molecular design, structured as follows:

- we begin Section 2.2 by discussing the fundamentals of graphs and the graph generation problem, establishing the mathematical foundation for molecular representation using GNNs. Here, we compare graph-based approaches to traditional SMILES-based methods, highlighting the advantages and challenges of each representation;

- we then provide an overview of GNN architectures in Section 2.3, describing the structure and role of their main components – i.e. propagation modules (Section 2.3.1) and pooling modules (Section 2.3.4);
- in Section 2.4 we describe the most utilized learning frameworks, which form the foundational architecture of generative models. Choosing the learning framework is common to every deep learning application design. This operation is interconnected with the other task-specific modeling decisions, each influencing the other;
- in Section 2.5 we deeply investigate such choices. In particular, in Section 2.5.1 we focus on the various ways to model the generation process. This can range from one-shot approaches that produce entire molecules simultaneously to sequential methods that build molecules step-by-step. The generation can be carried out at different granularity levels, which depend on the utilized molecular dictionary. This aspect is discussed in Section 2.5.2. Another critical aspect of molecular generation is ensuring that the produced structures adhere to chemical rules. We describe them in Section 2.5.3. In conclusion, in Section 2.5.4 we discuss conditioning techniques allowing for targeted generation of molecules with specific properties;
- finally, in Section 2.6 we review key datasets and benchmarks used in the field, discussing evaluation metrics and challenges in assessing model performance.

2.2 Foundations of molecular graph learning

2.2.1 Graph fundamentals and graph generation problem

A *graph* is a mathematical structure defined as a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_N\}$ is a set of N nodes and $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_M\}$ is a set of M edges. Each edge $\varepsilon = (v_i, v_j)$ connects two nodes v_i and v_j. The connectivity of a graph is represented by its *adjacency matrix* $\boldsymbol{A} \in \{0, 1\}^{N \times N}$, where its element a_{ij} is 1 if nodes v_i and v_j are adjacent (connected by an edge), and 0 otherwise. If the adjacency matrix is symmetric, the graph is said to be *undirected*, otherwise it

is said to be *directed*. In this context, we primarily focus on undirected graphs. The *neighborhood* of a node v_i, denoted as $\mathcal{N}(v_i)$, is the set of all nodes adjacent to it, and the *degree* of v_i is $|\mathcal{N}(v_i)|$. The *Laplacian matrix* is another important matrix representation of a graph. For an undirected graph, the Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is the *degree matrix* (a diagonal matrix with $d_{ii} = |\mathcal{N}(v_i)|$) and $\boldsymbol{A}$ is the adjacency matrix. The Laplacian matrix plays a crucial role in various graph-theoretic analyses and algorithms, including spectral clustering, graph partitioning, and the study of graph dynamics [53].

Graphs have proven to be versatile structures for describing complex data in various fields, including chemistry, software engineering, and image processing [54, 55]. Their ability to represent relational information makes them particularly valuable in domains such as drug design and network science [56, 57].

The graph generation problem has attracted significant attention as well due to its wide-ranging applications [58]. Historically, several algorithms have been developed to generate graphs with specific properties:

- the Erdős–Rényi model [59] for random graphs with a constant probability of edge connection;
- the Watts–Strogatz model [60] for random social networks with high edge density;
- the Barabási–Albert model [61] for scale-free graphs with power law degree distribution.

These models primarily focus on generating adjacency matrices based on structural properties, treating nodes and edges as indistinguishable. However, to represent molecules and other complex structures, richer graph representations are needed.

Annotated graphs extend the basic graph structure by associating additional information with nodes and edges. They are the main data structure used in this thesis. Let us consider a graph $(\mathcal{V}, \mathcal{E})$ with N nodes and M edges. The corresponding annotated graph is a 5-ple $(\mathcal{V}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{E})$ where: $\boldsymbol{X} \in \mathbb{R}^{N \times c}$ is the node feature matrix having c as the node feature dimension; $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency

matrix, now allowing for continuous edge weights; $\boldsymbol{E} \in \mathbb{R}^{M \times d}$ is the edge feature matrix having d as the edge feature dimensions. In this notation, $\boldsymbol{x}_i$ represents the feature vector of node v_i, while a_{ij} and $\boldsymbol{e}_{ij}$ represent the scalar weight and the feature vector of edge (v_i, v_j), respectively. In the following, we will refer to annotated graphs as $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{E})$. We will refer to annotated graphs with no edge features as $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$, and we will refer to annotated graphs with a binary adjacency matrix as $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{E})$.

Using this enhanced structure, a molecule can be fully represented as an annotated graph where nodes correspond to atoms, edges to bonds, and the feature matrices contain relevant information such as atom types, bond types, Cartesian coordinates, and chemical properties.

2.2.2 Graphs vs SMILES representations

In the field of drug discovery and molecular design, the choice of molecular representation plays a crucial role in the effectiveness of ML models. Two prominent approaches have emerged: SMILES strings and graph-based representations. Each has its strengths and limitations, which have significant implications for model design and performance.

SMILES-based Approaches

SMILES (Simplified Molecular-Input Line-Entry System) strings [62] are a linear textual representation of molecular structures, encoding the connectivity and atom types of a molecule as a sequence of characters. This representation allowed the first wave of deep learning approaches in drug discovery to leverage powerful existing sequence learning architectures from natural language processing. These models, primarily including recurrent neural networkss (RNNs) (such as LSTM networks [63]) and attention-based networks (like Transformers [64]), were quickly adapted to the domain of drug discovery [65, 66, 67, 68, 69, 70, 71, 72, 73, 74].

While SMILES-based models have demonstrated the ability to generate realistic molecules, they face several limitations: these models, in fact, must learn both

chemistry and SMILES grammar, potentially wasting representational power on rules that are not the true target of the learning process. Moreover, SMILES strings have both canonical and non-canonical representations, which may lead to imprecise learning [75]. Additionally, there's a lack of continuity in SMILES representation: small perturbations in a SMILES string can result in significant changes to the corresponding molecular entity and, conversely, similar chemical entities can have vastly different SMILES representations.

Efforts have been made to address some limitations of SMILES-based approaches, such as the SELFIES (SELF-referencing Embedded Strings) representation [76] which ensures that every grammatically correct string corresponds to a chemically valid molecule. However, these approaches still require the model to learn a grammar.

Advanced techniques: LLMs. Despite the above mentioned inherent limitations of SMILES representations, recent breakthroughs in Large Language Models (LLMs) [77] have revitalized interest in sequence-based approaches to molecular design. The remarkable ability of transformer architectures to capture long-range dependencies and complex patterns has enabled new possibilities in molecular generation and property prediction. Recent work has shown that LLMs can serve as foundation models for Chemical Language Models (CLMs) that perform at or above the level of models trained solely on chemical SMILES string data [78]. Building on this foundation, approaches like ChemBERTa-2 [79] have demonstrated the potential of chemical foundation models by leveraging self-supervised learning on massive datasets of up to 77M compounds from PubChem. These models can effectively learn salient molecular representations that transfer well to downstream tasks.

Nevertheless, these approaches still face some of the fundamental challenges inherent to sequence-based representations, leading researchers to explore alternative approaches that might better capture the inherent structure of molecular data.

Graph-based Approaches

Graph-based representations address many of the limitations of SMILES strings, offering several key advantages in molecular modeling. Unlike SMILES-based approaches, graph-based models focus entirely on chemical properties and structures, eliminating the need to learn grammar rules. This allows the network to concentrate its representational power on the underlying chemistry. Furthermore, graph representations are naturally invariant to node ordering, a critical feature in molecular modeling that ensures consistent representation of a molecule regardless of how its atoms are numbered, thus simplifying the learning process.

The interpretability of graph-based approaches is another significant advantage. Sub-graphs can be directly interpreted as molecular fragments, providing more intuitive representations. Additionally, chemical structural constraints, such as molecular validity, can be more easily enforced on both full molecules and fragments in graph representations compared to SMILES representations. Graph representations also provide a more robust space of molecular structures, where similar graphs generally correspond to similar molecules. This stability is particularly beneficial for tasks like molecular optimization and property prediction, as it allows for smoother navigation of the chemical space.

Advanced techniques: Geometric and equivariant models. While molecular graphs effectively capture the topological structure, they can be naturally extended to include spatial information by equipping nodes with 3D coordinates. This geometric representation becomes crucial when dealing with tasks that depend on the spatial arrangement of atoms, such as protein-ligand binding free energy or conformer generation. Similarly, molecules can be represented as point clouds, where each atom is a point in 3D space with associated features.

These spatial representations introduce new challenges in model design, as predictions should remain consistent regardless of the molecule's orientation or position in space. Recent developments in geometric deep learning have addressed this aspect through equivariant architectures. While traditional GNNs already ensure equivariance to node permutations - meaning predictions remain unchanged when atoms

are reordered - modern architectures extend this to include spatial transformations. The E(n)-equivariant GNNs [80] pioneered this approach by designing neural networks that respect both permutation and Euclidean space symmetries. This ensures that molecular predictions remain consistent regardless of the molecule's orientation or position.

The key insight of these approaches lies in their ability to maintain both permutational and geometric equivariance, eliminating the need for extensive data augmentation and at the same time ensuring more robust predictions across different molecular conformations. This has proven particularly valuable in tasks involving protein-ligand interactions, conformer generation, and molecular dynamics, where both topological and spatial relationships play crucial roles.

These advantages have led to a growing interest in using GNN models for molecular representation and generation. However, the task of generating a molecular graph remains challenging, as it requires generating both the adjacency matrix and the feature matrices. This complexity is further amplified by the vast size of the drug-like chemical space, estimated at 10^{33} [81]. Nonetheless, the ability of graph representations to naturally capture molecular structures continues to drive new developments in the field, motivating researchers to address these challenges.

2.3 Graph neural networks: an overview

A typical GNN architecture consists of a sequence of graph processing modules [57], which can be categorized into three main types:

- *propagation modules*: they maintain the graph structure while propagating information between nodes following the graph topology. Most of the propagation modules can be ascribed to the Message-Passing Neural Networks framework (see Section 2.3.1);
- *pooling modules*: they modify the graph structure, usually generating a coarsened version, trying to keep the relevant information. Most of the pooling layers can be ascribed to the Select-Reduce-Connect framework (see Section

2.3.4);

- *sampling modules*: less common, they determine a subset of nodes from which to collect information when dealing with large graphs.

2.3.1 Message-passing neural networks

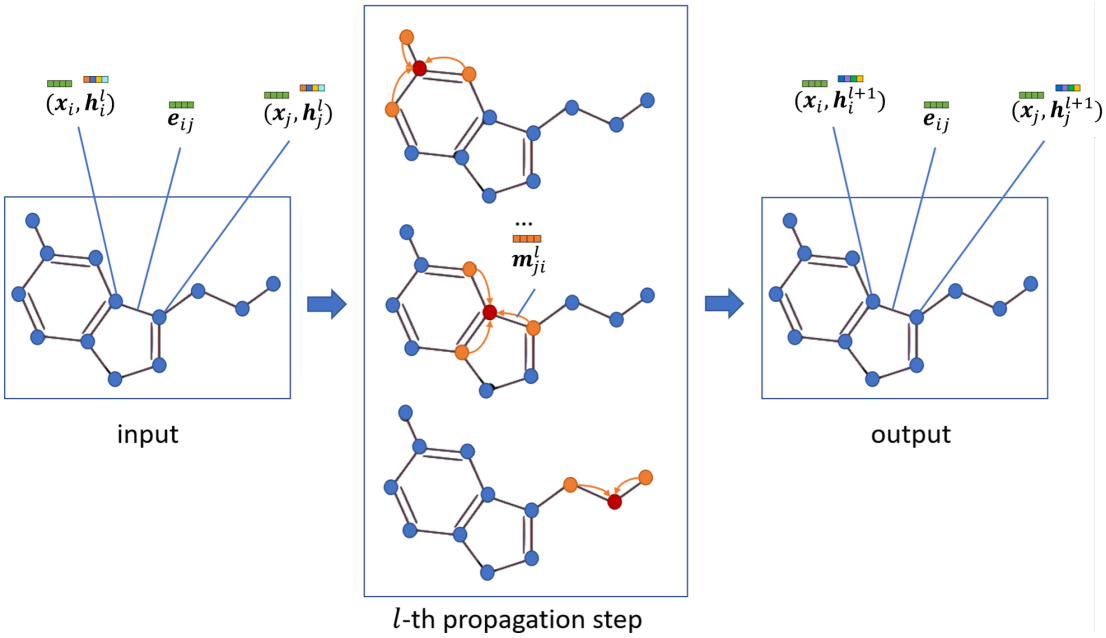

Figure 2.2: A visual representation of message-passing step to a generic node (in red) from its neighbor nodes.

A peculiarity of graph data structures is that they do not possess a specific node ordering for each neighbourhood. This means that the functions implemented by graph processing modules need to be invariant with respect to permutations of the nodes belonging to the same neighbourhood. The Message-Passing Neural Network (MPNN) [40] is a general GNN framework that gives an abstract structure to most standard permutation-invariant graph propagation modules.

Let us consider a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{E})$ and let us equip each node v_i with an additional, hidden representation $\boldsymbol{h}_i$. What the MPNN module does is to collect information from the node's neighborhood to update its hidden representation. This is generally defined as a Message-Passing (MP) step, an iterative process that enables the network to learn representations of nodes, edges, and entire graphs.

Through this mechanism, GNNs can capture both local and global structural information. For each MP step l, depicted in Figure 2.2, a MPNN module updates the hidden representation $\boldsymbol{h}_i^l$ as follows:

$$\begin{aligned} \boldsymbol{m}_{ji}^l &= M_l(\boldsymbol{x}_i, \boldsymbol{x}_j, a_{ij}, \boldsymbol{e}_{ij}, \boldsymbol{h}_i^l, \boldsymbol{h}_j^l) \\ \boldsymbol{m}_i^l &= A_l(\boldsymbol{m}_{ji}^l, v_j \in \mathcal{N}(v_i)) \\ \boldsymbol{h}_i^{l+1} &= U_l(\boldsymbol{x}_i, \boldsymbol{h}_i^l, \boldsymbol{m}_i^l) \end{aligned} \tag{2.1}$$

Here, M_l is a message function, A_l is a permutation-invariant aggregator function, and U_l is an update function. A readout function R can be used to compute a graph-level feature vector.

Propagation modules can be categorized into two main approaches: recurrent and convolutional.

2.3.2 Recurrent approaches

Recurrent approaches view message propagation as a dynamical system. The original GNN and Gated Graph Neural Networks (GG-NNs) [82] fall into this category. In these models, the symbol l represents the current time step of the state dynamical system.

The GNN message-passing scheme is the following:

$$\boldsymbol{m}_{ji}^l = \text{MLP}(\boldsymbol{x}_i, \boldsymbol{x}_j, a_{ij}, \boldsymbol{e}_{ij}, \boldsymbol{h}_j^l) \tag{2.2}$$

$$\boldsymbol{m}_i^l = \sum_{v_j \in \mathcal{N}(v_i)} \boldsymbol{m}_{ji}^l \tag{2.3}$$

$$\boldsymbol{h}_i^{l+1} = \boldsymbol{m}_i^l \tag{2.4}$$

Here, M_l is implemented as a multilayer perceptron (MLP) with shared weights across all iterations. GNNs are based on an iterative process that must converge to a fixed point. GG-NNs address this by stopping iterations after a set number of steps, without guaranteeing convergence.

2.3.3 Convolutional approaches

Convolutional approaches, here represented by Graph Convolutional Networks (GCNs) [83] and Graph Attention Networks (GATs) [84], extend the concept of convolution to the graph domain. Unlike recurrent approaches, each convolutional module corresponds to a single message-passing step, with consecutive steps using independent sets of weights.

The GCN message-passing scheme is:

$$\boldsymbol{m}_{ji}^{l} = \frac{1}{\sqrt{d_i d_j}} \boldsymbol{W}^{l} \boldsymbol{h}_{j}^{l} \tag{2.5}$$

$$\boldsymbol{m}_{i}^{l} = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} \boldsymbol{m}_{ji}^{l} \tag{2.6}$$

$$\boldsymbol{h}_{i}^{l+1} = \sigma(\boldsymbol{m}_{i}^{l}) \tag{2.7}$$

where d_i and d_j are the degrees of nodes v_i and v_j, σ is an activation function, and $\boldsymbol{W}^{l}$ is the learnt weight matrix.

GAT generalizes this further by introducing learnt attention weights. The GAT message-passing scheme thus becomes:

$$\boldsymbol{m}_{ji}^{l} = \alpha_{ij} \boldsymbol{W}^{l} \boldsymbol{h}_{j}^{l} \tag{2.8}$$

$$\boldsymbol{m}_{i}^{l} = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} \boldsymbol{m}_{ji}^{l} \tag{2.9}$$

$$\boldsymbol{h}_{i}^{l+1} = \sigma(\boldsymbol{m}_{i}^{l}) \tag{2.10}$$

where $\alpha_{ij} = \alpha(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with α any node features-dependent function.

GCNs have been extensively improved and adapted to various graph subdomains, including molecular data. Notable variants include Relational Graph Convolutional Networks (R-GCN) [85], Edge-Conditioned Graph Convolutional Networks

(EC-GCN) [86] and Spatial Graph Convolutional Networks (Spatial-GCN) [87]. R-GCN and EC-GCN learn separate message-passing functions for different edge types, while Spatial-GCN equips each node with a vector of coordinates. These advancements have significantly enhanced our ability to model and generate complex molecular structures.

It's worth noting that the challenge of molecular representation learning contributed to the early development of GCNs. Models like Molecular Graph Convolutions [88] and Neural Fingerprints [39] were specifically designed for the molecular domain.

2.3.4 Graph pooling

Graph pooling is a crucial operation in GNNs, allowing the network to hierarchically learn graph representations. Analogous to pooling operations in convolutional neural networks (CNNs), graph pooling aims to reduce the size of the graph while retaining important structural and feature information. This process is key to building deep GNNs capable of capturing both local and global graph properties, which is essential for tasks for graph classification [89], node classification [90, 91], graph matching [92], and spatio-temporal forecasting [93, 94].

Designing effective pooling layers for GNNs presents several unique challenges. Unlike grid-structured data where pooling operations are well-defined (e.g., max pooling in CNNs), graphs lack a regular structure, making it difficult to determine which nodes should be grouped together. Additionally, the pooling operation must preserve both local and global graph properties while being invariant to node permutations. Another significant challenge is maintaining the balance between information preservation and dimensionality reduction - aggressive pooling can lead to loss of critical structural information, while insufficient pooling may not effectively capture hierarchical patterns. Furthermore, in molecular graphs specifically, the pooling operation must respect chemical constraints and preserve important substructures that determine molecular properties.

Most pooling methods can be expressed through the Select-Reduce-Connect (SRC)

framework [95]. In its original formulation, the SRC framework takes also into account edge features. Here, we present a slightly simplified version that fits better to the scope of this thesis.

This framework decomposes the pooling operation $\texttt{POOL} : \mathcal{G} = (\boldsymbol{X}, \boldsymbol{A}) \mapsto \mathcal{G}' = (\boldsymbol{X}', \boldsymbol{A}')$ into three key steps: `SELECT`, `REDUCE` and `CONNECT`.

- `SEL` (selection): $\mathcal{G} \mapsto \boldsymbol{S} \in \mathbb{R}^{N \times K}$, a selection operator that maps the N original nodes to K pooled nodes, often referred to as *supernodes*. The entries in $\boldsymbol{S}$ represent the assignment of original nodes to supernodes;
- `RED` (reduction): $(\mathcal{G}, \boldsymbol{S}) \mapsto \boldsymbol{X}' \in \mathbb{R}^{K \times F}$, a reduction operator that computes the features of the supernodes. A common way to implement `RED` is $\boldsymbol{X}' = \boldsymbol{S}^\top \boldsymbol{X}$. This operation computes the features of each supernode by summing the features of all nodes assigned to it;
- `CON` (connection): $(\mathcal{G}, \boldsymbol{S}) \mapsto \boldsymbol{A}' \in \mathbb{R}^{K \times K}$, a connection operator that generates the new adjacency matrix for the pooled graph. Typically, `CON` is implemented as $\boldsymbol{A}' = \boldsymbol{S}^\top \boldsymbol{A} \boldsymbol{S}$. This operation creates the adjacency matrix of the pooled graph, where a'_{ij} represents the edge weight between supernodes i and j. This weight is the sum of all edge weights in the original graph between nodes assigned to supernode i and nodes assigned to supernode j.

The design of `SEL`, `RED`, and `CON` operations leads to a classification of pooling operators, categorized as *trainable* if all three operations are learnt end-to-end, *non-trainable* otherwise.

Most pooling methods can be ascribed to three main families: *soft-clustering* methods, *scoring-based* methods, and *one-every-K* methods.

Soft-clustering methods, also known as dense pooling [95], allow nodes to belong to multiple supernodes via soft memberships – i.e. each row in $\boldsymbol{S}$ can have multiple non-zero values. Examples include DiffPool [96], MinCutPool [97], StructPool [98], HoscPool [99], and Deep Modularity Networks (DMoN) [100]. These methods typically use an MLP or MPNN to process node features, followed by a softmax operation, resulting in a soft cluster assignment matrix. Each method incorpo-

rates unsupervised auxiliary loss functions to shape cluster formation. Trainable soft-clustering methods offer flexibility and expressive power, preserving all informative content from the original graph [101]. However, they face challenges such as memory constraints for large graphs, interpretability issues due to dense, less interpretable pooled graphs, and potential generalization problems when mapping each graph to a fixed number of supernodes.

Scoring-based methods select supernodes based on a scoring vector, choosing the top K scoring nodes. Examples include Top-k Pooling (Top-k) [90, 102], ASAPool [103], SAGPool [104], PanPool [91], TAPool [105], CGIPool [106], and IPool [107]. These methods differ in their scoring computation techniques and auxiliary tasks used to enhance pooled graph quality. The value of K can be set as a proportion of the original node count, allowing adaptability to graph size. However, the scores are generally obtained by node features that become locally similar after MP operations. This can lead to clustered selection, incomplete representation of the graph, and reduced performance in downstream tasks due to diminished expressiveness [108]. Some attempts have been made to promote diversity in node selection [109, 110], but challenges remain.

One-every-K methods employ graph-theoretical properties to uniformly subsample the graph for supernode selection. Examples include k-Maximal Independent Sets Pooling (k-MIS) [111], Graclus [112, 113], SEP [114], and Node Decimation Pooling (NDP) [115]. These methods use various approaches such as maximal K-independent sets, merging connected node pairs, or partitioning based on graph structural entropy. While these methods are adaptive and produce clear cluster assignments, they lack fine control over pooled graph size, are non-trainable as they precompute the pooled graph based solely on topology, and have limited scope as they don't account for node features or downstream task requirements.

Recent research has focused on developing methods that combine the strengths of these different approaches. For example, the MaxCutPool method [P3], which will be introduced in Chapter 4, aims to leverage graph-theoretical principles (inspired by one-every-K methods) while maintaining trainability and feature-awareness (similar to scoring-based and soft-clustering methods).

2.4 Learning frameworks

While in the past single GNNs were mainly used alone to learn molecular representations – e.g. for property prediction [44], nowadays they act as core component of complex learning frameworks. These frameworks form the foundational architecture of generative models, defining how the model learns to represent and generate molecular structures. Many of these approaches build upon the concept of hierarchical neural networks [116], where the overall model can be split into subnets, each focused on a specific task. In this section, we will explore the most prominent learning frameworks used in molecular generation.

2.4.1 Variational autoencoders

Variational Autoencoders (VAEs) [117] have become one of the most popular generative architectures in molecular design. They are particularly attractive due to their ability to learn a continuous latent space of molecules, which can be sampled to generate new structures.

The VAE architecture consists of two main components: an *encoder* network that, during training, compresses the input examples into the parameters of a latent probability distribution; a *decoder* network whose goal is to reconstruct the original input from samples drawn from the latent distribution [118]. The objective function of a VAE is the variational lower bound of the log-likelihood of the data, also known as the Evidence Lower BOund (ELBO):

$$\mathcal{L}_{VAE} = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{2.11}$$

where $q(\boldsymbol{z}|\boldsymbol{x})$ is the approximate posterior learnt by the encoder, $p(\boldsymbol{x}|\boldsymbol{z})$ is the likelihood of the data given the latent representation (learnt by the decoder), and $p(\boldsymbol{z})$ is the prior distribution of the latent space (typically a standard Gaussian). The first term in the ELBO is the reconstruction loss, which measures how well the model can reconstruct the input data. The second term is the Kullback-Leibler (KL) divergence, which encourages the learnt latent distribution to be close to

the prior distribution; this promotes a well-structured latent space that facilitates meaningful interpolation and sampling.

The application of VAEs to molecular generation offers several advantages. Perhaps most significantly, VAEs induce the encodings of the elements of the training dataset to belong to a continuous, latent manifold. This representation allows for smooth interpolation between different molecular structures, providing a powerful tool for exploring chemical space [119]. One can navigate this latent space to discover new molecules with desired properties, effectively *steering* the generation process towards specific objectives. By sampling from the learnt latent distribution, VAEs can generate diverse sets of molecules, potentially uncovering novel structures that might be overlooked by more deterministic approaches. On the other hand, when dealing with VAEs one can face several problems. One significant issue is the phenomenon known as *posterior collapse* [120]. This occurs when the model learns to ignore part of the latent space, effectively reducing the diversity of molecules it can generate. The inherent trade-off between reconstruction accuracy and latent space regularity in VAEs also presents challenges [121]. If the model prioritizes reconstruction too heavily, it may learn a latent space that is highly irregular and difficult to sample from meaningfully. Conversely, if the KL divergence term is weighted too strongly, the model may learn an overly simplified latent space that fails to capture the complexity of molecular structures. Finding the right balance is often problem-dependent and can require significant tuning.

Despite these facts, researchers have made significant progress in adapting VAEs to solve such problems [122, 123, 124]. As a result, VAEs remain one of the most widely used generative models today.

2.4.2 Generative adversarial networks

Generative Adversarial Networks (GANs) [125], represent another powerful approach to molecular generation. GANs consist of two competing neural networks: a generator and a discriminator. The generator produces artificial samples and has 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 trained simultaneously in a minimax game, formalized as:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.12}$$

where G is the generator, D is the discriminator, $p_{data}(\boldsymbol{x})$ is the true data distribution, and $p_z(\boldsymbol{z})$ is a prior distribution on the input noise variables.

In molecular generation, GANs are well-known for producing high-quality samples, with the adversarial framework pushing the generator to create increasingly realistic molecular structures. Unlike VAEs, GANs do not require explicit density estimation of the data, which simplifies certain aspects of the generative process.

Nevertheless, GANs notably suffer from training instability, which can be particularly problematic for discrete data such as molecular graphs [126]. GANs are also susceptible to *mode collapse* [127], where the generator produces a limited variety of molecules, failing to capture the full diversity of the underlying data distribution. Additionally, standard GANs do not offer an explicit latent space, which can make tasks like property optimization more difficult.

CycleGAN

The CycleGAN [128] is an extension of the GAN framework designed for unpaired image-to-image translation.

The CycleGAN architecture consists of two GANs trained jointly:

- generator $\mathrm{G} : X \to Y$ and discriminator D_Y;
- generator $\mathrm{F} : Y \to X$ and discriminator D_X,

where X and Y are two different molecular domains.

The key innovation of CycleGAN is the introduction of a cycle consistency loss:

$$\mathcal{L}_{cyc}(\mathrm{G}, \mathrm{F}) = \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[||\mathrm{F}(\mathrm{G}(\boldsymbol{x})) - \boldsymbol{x}||_1] + \mathbb{E}_{\boldsymbol{y}\sim p_{data}(\boldsymbol{y})}[||\mathrm{G}(\mathrm{F}(\boldsymbol{y})) - \boldsymbol{y}||_1]. \tag{2.13}$$

This loss ensures that the transformations are reversible, i.e., $\mathrm{F}(\mathrm{G}(\boldsymbol{x})) \approx \boldsymbol{x}$ and $\mathrm{G}(\mathrm{F}(\boldsymbol{y})) \approx \boldsymbol{y}$.

In molecular generation, the CycleGAN is particularly advantageous for tasks such as transforming molecules between different property regimes (e.g., low to high solubility), generating analogs of existing drugs, and translating between molecular representations. However, challenges again arise due to the discrete nature of molecular graphs, difficulties in ensuring cycle consistency for graphs with different node orderings, and the need to maintain the chemical validity of transformed molecules.

2.4.3 Normalizing flows

Normalizing flows [129] are a powerful class of generative models that allow for both efficient inference and sampling. At their core, they are maximum likelihood methods that define a bijective mapping between $\boldsymbol{z}$ belonging to a latent space and $\boldsymbol{x}$ belonging to the data space.

More specifically, we define:

$$\boldsymbol{x} = f(\boldsymbol{z}) \tag{2.14}$$

where $\boldsymbol{z}$ is drawn from a simple prior distribution (typically a standard Gaussian), and f is an invertible, differentiable function parameterized by some $\boldsymbol{\theta}$. The function f is constructed as a composition of simpler bijective functions $f = f_1 \circ f_2 \circ \cdots \circ f_K$:

$$\boldsymbol{x} \overset{f_1}{\longleftrightarrow} \boldsymbol{h}_1 \overset{f_2}{\longleftrightarrow} \boldsymbol{h}_2 \cdots \overset{f_K}{\longleftrightarrow} \boldsymbol{z} \tag{2.15}$$

Under this change of variables, the log-density of the data can be expressed as:

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}) + \sum_{i=1}^{K} \log \left| \det \left(\frac{\partial f_i}{\partial \boldsymbol{h}_{i-1}} \right) \right| \tag{2.16}$$

The clever design of the component functions f_i allows for efficient computation of this log-likelihood. Typically, these functions are constructed to have triangular Jacobians, making the log-determinant term computationally tractable. This enables the use of standard optimization techniques to train the model by minimizing the negative log-likelihood of the observed data.

In the context of molecular generation, the invertibility of normalizing flows is a

particularly useful feature. The ability to move between data space (molecular structures) and latent space in both directions allows for interesting applications, such as molecular optimization through latent space manipulation or the analysis of how changes in latent space correspond to structural changes in molecules [130]. However, normalizing flows are designed for continuous data while molecules are inherently discrete objects, with atoms and bonds forming a graph structure. This mismatch requires careful considerations and often necessitates the development of specialized techniques to bridge the gap between continuous and discrete representations [131].

2.4.4 Score-based models

Score-based models, introduced in [132], offer a novel approach to generative modeling. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. The model then learns a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model.

Let us consider a process that gradually transforms data into noise described by the following stochastic differential equation (SDE):

$$d\boldsymbol{x} = f(\boldsymbol{x}, t)dt + g(t)d\boldsymbol{w} \tag{2.17}$$

where $f(\boldsymbol{x}, t)$ is the drift coefficient, $g(t)$ is the diffusion coefficient, and $\boldsymbol{w}$ is a standard Wiener process [133].

The reverse process is also described by a SDE:

$$d\boldsymbol{x} = [f(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]dt + g(t)d\bar{\boldsymbol{w}} \tag{2.18}$$

where $p_t(\boldsymbol{x})$ is the probability density at time t, and $\bar{\boldsymbol{w}}$ is a standard Wiener process running backwards in time.

The crucial term in this reverse SDE is the gradient of the log-density with respect

to the data $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$, that is called *score function*; incidentally in physics this is just a force.

Once the score function is known, it is possible to use Langevin dynamics [134, 135] sampling process, a Markov chain Monte Carlo (MCMC) method that solely relies on the score function. Discretizing Langevin dynamics via a first integrator one can obtain the associated sampling process. The resulting update rule is:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \epsilon \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_t) + \sqrt{2\epsilon}\boldsymbol{z}_t \tag{2.19}$$

where ϵ is a small step size, and $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is standard Gaussian noise.

In the context of score-based models, the true, unknown, gradient $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_t)$ is replaced with a learnt score function $s_\theta(\boldsymbol{x}, t)$:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \epsilon s_\theta(\boldsymbol{x}_t) + \sqrt{2\epsilon}\boldsymbol{z}_t. \tag{2.20}$$

This process is iterated for a number of steps, gradually transforming random noise into samples from the target distribution. The noise term $\sqrt{2\epsilon}\boldsymbol{z}_t$ allows the sampler to explore the distribution and avoid getting stuck in local modes.

A key strength of score-based models is their training stability compared to adversarial approaches like GANs [136]. This stability has been leveraged in various domains, including image generation [137] and molecular design [138] . However, the computational cost associated with the reverse process and Langevin dynamics sampling [134] can result problematic. This cost can be a limiting factor in high-throughput virtual screening scenarios. Despite these challenges, score-based models continue to advance, with recent work exploring their application in 3D structure generation [139] and conditional generation tasks [140].

2.4.5 Reinforcement learning

Reinforcement learning (RL) provides a framework for learning to make decisions in complex environments. In molecular generation, RL can be used to optimize molecules for specific properties or objectives.

The RL framework typically consists of:

- an agent (the generative model);
- an environment (the chemical space);
- a state space (molecular representations);
- an action space (modifications to molecules);
- a reward function (based on desired molecular properties).

The goal is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{2.21}$$

where θ are the policy parameters, γ is a discount factor, and r_t is the reward at time step t [141].

RL offers significant advantages for molecular generation: the model can be trained to directly optimize desired molecular properties (such as solubility or binding affinity) and handle complex, non-differentiable reward functions, which are common in molecular tasks [142]. Additionally, RL enables effective exploration of the chemical space, allowing the model to discover novel molecular structures by balancing exploration (trying new structures) with exploitation (refining known structures). The design of reward functions, however, can be a complex task [143]. Moreover, learning in high-dimensional action spaces can be unstable, making it difficult to converge to a solution [144, 145].

2.5 Guided generation of molecular graphs

Formally speaking, the most common approach to molecular generation and conditioning involves creating a dataset of molecular graphs $\mathcal{D}_{self} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_M\}$ and devising a self-supervised model to learn from this data. Less frequently, the task is framed as a supervised problem, where the input dataset comprises pairs of molecules $\mathcal{D}_{sup} = \{(\mathcal{G}_1, \widehat{\mathcal{G}}_1), ..., (\mathcal{G}_M, \widehat{\mathcal{G}}_M)\}$, and the model learns to map from a

known unoptimized molecule $\mathcal{G}_i$ to an optimized target molecule $\widehat{\mathcal{G}}_i$ [146, 147, 148, 149, 150].

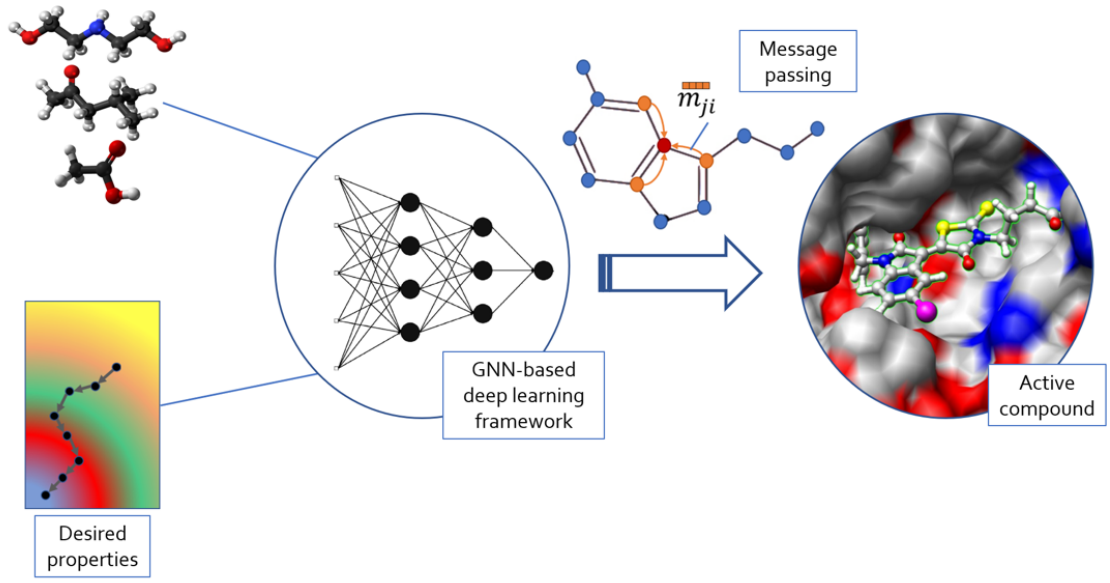

Figure 2.3: Overview of a GNN-based molecular design framework. The process integrates structural information from input molecules, desired molecular properties, and GNN-based deep learning to generate optimized compounds with potential biological activity.

Beyond generating molecular structures that reasonably belong to the training dataset, a crucial objective in practical applications is to guide the generation process towards molecules with specific desired properties (Figure 2.3). This involves skewing the distribution of generated molecules towards structures optimized for a set of prescribed (numerical) properties. The methods used to achieve this conditioning are varied and depend on both the underlying generation approach and the specific deep learning models employed.

Table 2.1 provides a comprehensive overview of recent approaches in the literature, highlighting their key characteristics. In the following subsections, we will explore them in detail, examining how different approaches address the challenges of generating valid, diverse, and property-optimized molecular structures. We will discuss how the molecular graph generation process is modeled (Section 2.5.1), its granularity level (Section 2.5.2), the techniques for enforcing chemical validity (Section

Ref	Learning Framework	GNN Model	Conditioning	Validity Constraint	Generation Process	Granularity Level	Code Availability
[151]	VAE	GGNN	gradient ascent	action-based constraints	sequential	atom-wise	link
[152]	VAE	MPNN	gradient ascent bayesian optimization	learnt	sequential	fragment-based	link
[153]	VAE	EC-GCN	conditional vector	learnt	one-shot	-	link
[154]	GAN	R-GCN	RL	learnt	one-shot	-	link
[155]	Hierarchical	R-GCN	conditional vector	learnt	sequential	atom-wise	link
[156]	GAN	GCN	RL	rejection sampling	sequential	hybrid	link
[157]	Hierarchical	GraphRNN	RL	rejection sampling	sequential	atom-wise	-
[158]	Normalizing Flow	R-GCN	RL	rejection sampling	sequential	atom-wise	link
[159]	Normalizing Flow	R-GCN	RL	action-based constraints	sequential	atom-wise	-
[147]	VAE + GAN	MPNN	conditional vector	learnt	sequential	fragment-based	-
[160]	VAE + Attention	MPNN	conditional vector	learnt	sequential	fragment-based	link
[161]	Hierarchical	GGNN	RL	learnt	sequential	atom-wise	link
[162]	Hierarchical	MPNN	conditional vector	learnt	sequential	atom-wise	link
[163]	GAN	GCN	RL	rejection sampling	sequential	hybrid	link
[164]	VAE	MPNN	bayesian optimization	action-based constraints	sequential	atom-wise	link
[165]	Hierarchical	MPNN	conditional vector	learnt	masked	-	link
[166]	VAE	MPNN	conditional vector	action-based learning	one-shot	-	link
[167]	Hierarchical	GCN	conditional vector	learnt	sequential	atom-wise	link
[168]	Normalizing Flow	R-GCN	gradient ascent	post hoc correction	one-shot	-	link
[169]	VAE + Normalizing Flow	R-GCN	gradient ascent	post hoc correction	one-shot	-	link
[130]	Normalizing Flow	R-GCN	gradient ascent	learnt	one-shot	-	link
[170]	VAE	GAT	conditional vector	learnt	one-shot	-	-
[171]	Normalizing Flow	GAT	gradient ascent	learnt	sequential	atom-wise	-
[148]	Normalizing Flow	R-GCN	supervised	learnt	one-shot	-	-
[149]	CycleGAN	JT-VAE	supervised	learnt	one-shot	-	link
[172]	Hierarchical	GGNN	gradient ascent	learnt	reaction-based	-	link
[48]	Hierarchical	MPNN	supervised	learnt	sequential (SMILES)	-	-
[173]	Hierarchical	MPNN	markov chain	learnt	sequential	fragment-based	link
[174]	VAE	GGNN	structural embedding	action-based constraints	sequential	atom-wise	link
[175]	VAE	MPNN	structural embedding	learnt	sequential	atom-wise	link

Table 2.1: 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, motif) and the code availability

2.5.3) and the methods for conditioning the generation on desired molecular properties (Section 2.5.4). Through this analysis, we aim to provide a comprehensive understanding of the current state of the art in guided molecular graph generation and highlight promising directions for future research in this rapidly evolving field.

2.5.1 Generation process

In molecular graph generation, the primary goal is to learn the characteristics of a given dataset $\mathcal{D}_{self} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_M\}$. The underlying assumption is that this dataset is sampled from an unknown distribution $p^*(\mathcal{G})$, which we aim to model with an explicit distribution $p_\theta(\mathcal{G})$. During training, the network learns the parameters θ of a function $f_\theta(\mathcal{G})$ that approximates $p_\theta(\mathcal{G})$. After training, new molecules are generated by sampling from f_θ.

The generation process can be conducted in different ways, as illustrated in Figure 2.4. The main approaches are:

- one-shot models, where a molecule is produced in a single inference step;
- sequential models, where a molecule is built via several inference steps of the same model;
- masked graph modeling, inspired by masked language models like BERT [176].

One-shot generation

In one-shot generation, we typically assume that $p_\theta(\mathcal{G}) = p_\theta(\mathcal{G}|\boldsymbol{z})$, where $\boldsymbol{z} = z_1, ..., z_n$ is a vector of latent variables with a known prior distribution. The generation process involves sampling these latent variables and generating the entire graph based on the sampled vectors.

Early models like MolGAN [154] and GraphVAE [153] employed this strategy. MolGAN uses a GAN framework where the generator is an MLP that directly samples the adjacency matrix of the molecular graph, while the discriminator is based on R-GCN. GraphVAE, on the other hand, models molecules using a probabilistic graph

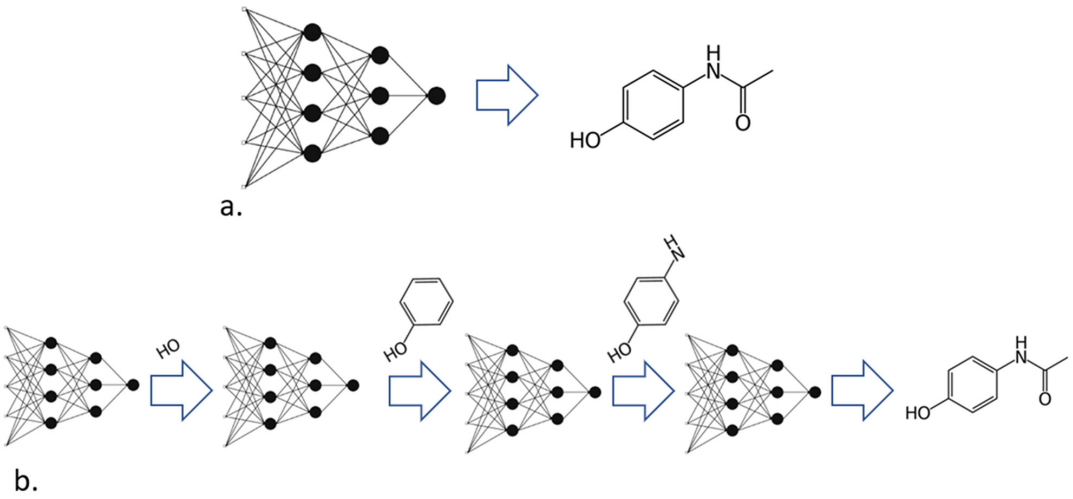

Figure 2.4: a. One-shot generation. **b.** (Fragment-based) Sequential generation.

where node/edge existence is represented by Bernoulli variables, and node/edge attributes are multinomial variables. It uses an EC-GCN as the encoder and a deterministic MLP as the decoder.

Another approach, MPGVAE [170], models the bond/atom joint latent distribution as:

$$p_\theta(\mathcal{G}|\boldsymbol{z}) = p_\theta(\boldsymbol{X}, \boldsymbol{E}|\boldsymbol{z}) = \prod_{v_i \in \mathcal{V}} p_\theta(\boldsymbol{x}_i|\boldsymbol{z}) \prod_{\varepsilon_{ij} \in \mathcal{E}} p_\theta(\boldsymbol{e}_{ij}|\boldsymbol{z}) \tag{2.22}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{e}_{ij}$ are categorical distributions over atom and bond types, respectively. Here, the absence of a bond is seen as one of the edge categories. The generation is a three-step process. First, an initial graph structure is generated from latent variables in a one-shot fashion. Second, a MPNN iteratively refines node and edge representations. Third, the final molecular graph is sampled from the node/edge distributions $\boldsymbol{x}_i$ and $\boldsymbol{e}_{ij}$.

Alternative formulations introduce dependencies between bonds and atoms:

$$p_\theta(\mathcal{G}|\boldsymbol{z}) = p_\theta(\boldsymbol{X}, \boldsymbol{E}|\boldsymbol{z}) = p_\theta(\boldsymbol{X}|\boldsymbol{E}, \boldsymbol{z}_{\boldsymbol{X}}) p_\theta(\boldsymbol{E}|\boldsymbol{z}_{\boldsymbol{E}}) \tag{2.23}$$

or:

$$p_\theta(\mathcal{G}|\boldsymbol{z}) = p_\theta(\boldsymbol{X}, \boldsymbol{E}|\boldsymbol{z}) = p_\theta(\boldsymbol{E}|\boldsymbol{X}, \boldsymbol{z}_{\boldsymbol{E}}) p_\theta(\boldsymbol{V}|\boldsymbol{z}_{\boldsymbol{V}}) \tag{2.24}$$

where $\boldsymbol{z_V}$ and $\boldsymbol{z_E}$ are latent variables for vertices and edges, respectively. Here, the models first focus on one of the two sets and then use that information to complete the generation. In [177], 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 [168], 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.

In the context of score-based models for graph generation [178], the diffusion process is modeled as a system of two coupled SDEs:

$$\begin{cases} d\boldsymbol{X}_t = \left[f_{1,t}(\boldsymbol{X}_t) - g_{1,t}^2 \nabla_{\boldsymbol{X}_t} \log p_t(\boldsymbol{X}_t, \boldsymbol{A}_t)\right] d\tilde{t} + g_{1,t} d\tilde{\boldsymbol{w}}_1 \\ d\boldsymbol{A}_t = \left[f_{2,t}(\boldsymbol{A}_t) - g_{2,t}^2 \nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{X}_t, \boldsymbol{A}_t)\right] d\tilde{t} + g_{2,t} d\tilde{\boldsymbol{w}}_2 \end{cases} \tag{2.25}$$

Here, two GNNs learn the gradients $\nabla_{\boldsymbol{X}_t} \log p_t(\boldsymbol{X}_t, \boldsymbol{A}_t)$ and $\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{X}_t, \boldsymbol{A}_t)$. New samples are generated by using these learnt gradients in the SDEs and solving them with standard SDE integrators.

Sequential generation

Sequential generation addresses the limitation of one-shot methods that require a maximum size for the output molecule. In this approach, generation is modeled as a sequence of *graph transition actions*. A typical action set includes vertex and edge addition and removal [155, 179]. Although some sequential models still require an upper bound on the number of nodes in the generated molecule [151], this approach is generally more flexible than one-shot generation.

The most common way to model sequential generation is to decompose $p_\theta(\mathcal{G})$ into a product of conditional distributions:

$$p_\theta(\mathcal{G}) = \prod_t p_\theta(a_t | \mathcal{G}_t, ..., \mathcal{G}_0) \tag{2.26}$$

where $a_t \in T(\mathcal{G}_t, ..., \mathcal{G}_0)$, that is the action set available on graph $\mathcal{G}_t$ at step t.

In practice, each graph $\mathcal{G}$ is transformed into a sequence of operations $S_{\mathcal{G}}$, 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 [180]. For the generative task, there is currently no theoretically grounded preferred ordering. The most effective approach is architecture-dependent and chiefly determined by empirical evidence.

Sequential generation methods can be categorized into non-Markovian and Markovian approaches. For Markovian strategies, one restricts T to depend only on the latest graph, namely $T = T(\mathcal{G}_t)$. These approaches will be detailed in the following subsections.

Non-Markovian methods JT-VAE [152] was one of the first sequential approaches. In this method, each molecule is assembled by linking chemical fragments from a predefined dictionary, which is derived in a chemically principled way from the training set without a learning process. Specifically, a VAE-based model generates a *junction tree* for each molecule. The junction tree is the basic tree-structured scaffold representing connections among nodes, where each node represents a set of candidate fragments. The action set for the junction tree is the sequential addition of nodes while maintaining the tree topology. To assemble the final molecule, a single fragment is chosen for each node.

Other methods, such as those presented in [155] and [157], can be ascribed to hierarchical neural network frameworks. In [155], a stack of R-GCN blocks with a `BN-ReLU-Conv` structure [181] is used to extract representations. 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. This approach employs a DFS graph visit,

enhanced by the additional move of going to a random node, thus generating several building paths for the same molecule. Due to the non-uniqueness of the sequence, a sampling strategy is also adopted. The GraphRNN framework [182] uses a BFS ordering together with classical RNN networks to directly generate the adjacency matrix. This approach was extended to molecular graphs in [157], where 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 it remains relevant to the field of molecular graph generation. Non-Markovian approaches offer flexibility in capturing long-range dependencies in the graph generation process, potentially leading to more coherent and diverse molecular structures. However, they may also be more computationally intensive and challenging to train compared to Markovian methods.

Markov-process-based methods These methods simplify the generation process by assuming $p_\theta(a_t|\mathcal{G}_t, ..., \mathcal{G}_0) = p_\theta(a_t|\mathcal{G}_t)$. This assumption is reasonable because the generation history that led to a specific graph should not influence subsequent decisions. Examples include GCPN [156], MG^2N^2 [183], and GraphAF [158].

MG^2N^2 is based on three GNNs arranged 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 determines the edge label;
- the third network determines the existence of other edges connecting the new node to preexisting nodes.

GraphAF is based on normalizing flows [129]. It adapts to the graph domain a variant called autoregressive flows [184], which scale linearly with respect to the length of the sequence to be learnt. Within the GraphAF framework an R-GCN calculates the embedding for the current graph at each step, while two separate

MLPs learn the parameters of the distributions from which the next node and its edges are sampled. A recent addition to Markov-based methods is MolGrow [171], which uses a graph normalizing flow for hierarchical molecular generation. This model generates molecular graphs by progressively adding atoms and bonds, allowing for the generation of larger and more complex molecules.

Markov-process-based methods offer a balance between computational efficiency and the ability to capture local dependencies in the graph generation process.

Masked graph modeling

This approach, introduced in [165], uses graph masking for generation. Instead of learning $p(\mathcal{G})$, the model learns $p(\eta|\mathcal{G}_{\setminus\eta})$, where η is a subset of $\mathcal{G}$ and $\mathcal{G}_{\setminus\eta}$ is $\mathcal{G}$ with that subset masked out.

Generation starts from an initial graph. At each step, a part of the graph is masked and resampled according to the learnt conditional distribution. This method offers a unique approach to molecular graph generation, drawing inspiration from successful masked language models in natural language processing [176].

Each of these generation processes offers distinct advantages and challenges in the context of molecular design. One-shot methods can be faster but may struggle with larger molecules, while sequential methods offer more control but can be computationally intensive. Masked graph modeling provides a novel middle ground, potentially offering benefits of both approaches.

2.5.2 Granularity level

The granularity level in molecular generation refers to the size of the building blocks used in the generation process. This choice can impact the model's flexibility, validity, and exploration capabilities. Two main approaches dominate the field: atom-based and fragment-based methods.

Atom-based methods In atom-based approaches, molecules are generated one atom at a time. This fine-grained approach offers maximum flexibility, allowing

for the exploration of the entire chemical space. Models like GraphAF [158] and MolDQN [185] exemplify this approach, generating molecules atom by atom and bond by bond. The advantage of atom-based methods lies in their ability to potentially create novel molecular structures that might not be easily accessible through fragment-based approaches. However, atom-based methods can struggle with generating larger molecules efficiently and may require additional constraints to ensure chemical validity. To address this, one might rely on fragment dictionaries.

Fragment-based methods These methods generate molecules by combining predefined molecular fragments, typically consisting of up to six atoms. This approach, exemplified by JT-VAE [152], offers several advantages. Firstly, it inherently ensures a higher degree of chemical validity, as the fragments themselves are chemically valid structures. Secondly, it can lead to more efficient generation of drug-like molecules, as many common molecular substructures are directly incorporated. JT-VAE learns its fragment dictionary from the training set, which allows for adaptation to specific chemical spaces but may limit reconstruction accuracy on unseen molecules. The authors extended this approach in [160], incorporating larger subgraphs (structural motifs) to achieve higher reconstruction accuracy.

Hybrid and adaptive approaches Recent research has explored hybrid and adaptive approaches that aim to combine the strengths of both atom-based and fragment-based methods. For example, GCPN [156] provides a flexible framework that can adapt to different granularity levels, from atoms to larger fragments, depending on the task at hand. CORE [146] introduces an interesting compromise by using a hierarchical approach. It first generates a molecular scaffold using larger fragments and then refines the molecule at the atomic level, combining the efficiency of fragment-based methods with the flexibility of atom-based approaches.

2.5.3 Validity constraints enforcement

Ensuring the validity of generated molecules is crucial in molecular graph generation. While some models can learn chemical validity implicitly from data [155, 154, 153, 152, 147, 160, 165, 167, 149, 172, 48], many approaches incorporate

explicit strategies to enforce validity constraints. These strategies can be broadly categorized into learnable and non-learnable methods.

Learnable methods

Learnable methods incorporate validity constraints into the model's architecture or learning process, allowing the model to improve its ability to generate valid structures over time.

Action-based learning These methods enforce validity by penalizing incorrect actions during training, and preventing them from being taken during generation. CGVAE [151] learns to implement valency checks, preventing the addition of bonds when an atom has reached its maximum valency. An extension in [186] adapts the CGVAE decoder to learn to use valency histograms as additional inputs to guide the generation process. RL based methods, such as those in [156], [157], and [158], learn to incorporate validity checks as part of their reward structure. These methods use valency checks as intermediate feedback signals during training and learn to employ validity-constrained sampling during inference. The approach in [187] learns state transition dynamics where each possible action leads to valid substructures. These action-based methods allow full control of the model's tolerance for invalid actions, with the potential to produce 100% valid molecules. They can be configured to allow intermediate invalid steps to increase exploration abilities or to strictly forbid any invalid actions.

Penalty-based learning These techniques, differently from the action-based ones, allow incorrect actions during generation while still learning to penalize them. This trades the structural difficulty of enforcing correct moves for the relative ease of optimization. The Regularized Graph VAE [188] employs a penalty technique [189] to transform the constrained problem into an unconstrained one. The model learns to minimize constraint violations through a tunable hyperparameter in the cost function. A similar approach is used in [190]. In [166], validity is encouraged through a reinforcement learning framework. The model learns a reward function for valid samples, and the GraphVAE decoder acts as a learnt policy network.

While these approaches may not always generate 100% valid molecules, they are often easier to implement and can approximate constraint enforcement as a learnt restraint.

Non-learnable methods

Non-learnable methods enforce validity through fixed rules or post-processing steps that do not adapt or improve over time.

Action-based constraints These methods use fixed rules to prevent invalid actions during generation. Valency checks are used to avoid adding bonds when an atom has reached its maximum valency. If atoms have missing bonds after generation, a fixed number of hydrogen atoms is added [177].

Post-hoc correction These methods apply fixed corrections to the generated molecules after the generation process. The generated graph is modified according to predetermined valency rules, preserving a maximal valid subgraph from the original output. This approach is used in flow-based models such as MoFlow [168] and GF-VAE [169]. While these methods may lead to some undesirable modifications, they are widely applicable across different generation approaches.

Validity rejection sampling This method involves generating molecules and then discarding those that fail validity checks. It is particularly effective for models with fast inference times which can quickly generate and filter a large number of candidates. The models that utilize this approach include MolGAN [154], GraphNVP [130], JTVAE [152], and MoFlow [168]. The main advantage of validity rejection sampling is its simplicity and effectiveness, especially for models with fast generation times. It allows the generative model to explore a wider range of structures without being constrained by explicit validity rules during the generation process. This can potentially lead to more diverse and novel molecular structures. However, this method has limitations in terms of efficiency for models with slower generation times, potential bias if the model tends to produce certain types of invalid structures, and scalability issues for very large or complex molecules.

2.5.4 Conditioning

Conditioning the graph generation is a powerful way to skew the learnt distribution towards a desired bias. This approach is crucial for targeted molecular design, allowing the creation of molecules with specific properties or characteristics. In this section, we first introduce a novel framework for categorizing conditioning tasks, then detail such tasks, and finally describe the associated conditioning techniques.

Conditioning tasks

We propose a comprehensive framework that encapsulates the various conditioning tasks found in the literature. Any conditional task can be represented as one or a subset of the following quadruplet:

$$(\underbrace{\mathcal{R}}_{1)}, \underbrace{\boldsymbol{p} \gtrless \boldsymbol{p_0}}_{2)}, \underbrace{\boldsymbol{p} = \boldsymbol{p_0}}_{3)}, \underbrace{\min(\boldsymbol{p})}_{4)}) \tag{2.27}$$

where:

1) $\mathcal{R}$ represents a set of additional entities given as input to the model, such as drugs, proteins, or any relevant biological information;

2) $\boldsymbol{p} \gtrless \boldsymbol{p_0}$ denotes a set of constraints, where $\boldsymbol{p}$ is a set of numerical properties and $\boldsymbol{p_0}$ is the corresponding vector of thresholds;

3) $\boldsymbol{p} = \boldsymbol{p_0}$ encodes numerical equality between desired and actual properties;

4) $\min(\boldsymbol{p})$ represents a set of desired optimal values.

This framework provides a unified way to categorize and understand the various conditioning tasks in molecular generation:

Property optimization This task involves generating new molecules to optimize the value of one or more numerical properties. Typical properties include:

- Quantitative Estimate of Drug-likeness (QED, introduced in [191]) [164, 168, 169, 130, 151, 154, 156, 157, 158, 147, 160, 161];

- Synthetic Accessibility Score (SAS, introduced in [192]) [154, 155];
- penalized logP [164, 177, 169, 171, 152, 156, 157, 158, 160], defined as the octanol-water partition coefficient minus the synthetic accessibility score and the number of cycles of length bigger than 6. This metric balances lipophilicity with synthetic feasibility and structural complexity;
- biological activity [162, 163, 147, 160, 161].

This task can be ascribed to component 4) in our framework.

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 towards molecules with a specified set of attributes, which can be numerical or categorical values for molecular properties. For example, [153] and [170] provide the desired histogram of atom types of the output molecule, while [162, 165, 166, 156] condition the generation towards molecules with a specific weight. This task can be ascribed to component 3) in our framework.

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 $\mathcal{G}$, and the goal is to maintain the similarity $sym(\mathcal{G}, \hat{\mathcal{G}}) \geq \delta$, where $\hat{\mathcal{G}}$ 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 [177, 168, 169, 167, 171, 152, 156, 157, 158, 160]. This task is a combination of components 1) and 2).

Scaffold hopping In scaffold hopping, the goal is to minimize a chemical similarity and maximize a physical (field effects) similarity with respect to a given drug [193, 48, 174]. This task is a combination of components 1) and 3).

Conditioning methods

Conditioning techniques in molecular graph generation can be broadly categorized into two main approaches: training time conditioning and ex post conditioning.

Training time conditioning Training time conditioning methods incorporate property objectives directly into the model's learning process, allowing the generative model to inherently understand and produce molecules with desired characteristics. One of the most prevalent approaches in this category is the use of conditional vectors [153, 155, 162, 165, 166, 167, 147, 160]. In this method, the model learns to generate molecules conditioned on a vector of desired property values, effectively learning a conditional distribution $p(\mathcal{G}|\boldsymbol{c})$, where $\boldsymbol{c}$ is a vector containing the values of the properties of interest. For each molecular graph $\mathcal{G}$ in the dataset, the ground-truth value for the vector $\boldsymbol{c}$ is calculated. During training, the model is fed with input couples $(\mathcal{G}, \boldsymbol{c})$. During inference, the desired values for $\boldsymbol{c}$ are given to the model to guide the generation. This approach allows for fine-grained control over multiple molecular properties simultaneously.

Another powerful training time technique is substructure-based conditioning, employed by [174] and [175]. Here, the model learns to generate molecules that incorporate specific substructures, which is particularly useful for tasks like generating molecular linkers or multi-objective optimization.

Gradient ascent in the latent space of VAEs is another effective strategy for property optimization [151, 152, 177]. This method jointly trains an additional regressor to predict property values from latent space points. As a consequence, the latent space is modeled and regularized taking account of the additional information. During inference, the latent representation and its property value are calculated, optimized through gradient ascent with respect to the property predictor model, and then decoded from the local optimum (see Figure 2.5).

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. One of the first, prominent examples of RL approaches for molecular graph generation is GCPN [156]. The authors modeled the generation as a Markov process (see Section 2.5.1). At each step, a GCN extracts a representation of the partially generated molecule, and then four MLP models decide the next action. By mod-

elling the generation as a sequence of decisions one can define short-term (based on the single action) and long-term (based on the final result) rewards. In GCPN the short-term rewards are a combination of validity rewards (see 2.5.3) and adversarial rewards (in a GAN fashion). In contrast, long-term rewards are used to condition the generation towards the desired property. A similar approach is used in [157, 158, 161, 163].

Finally, some researchers have also explored supervised learning approaches, treating property optimization as a direct mapping problem from input molecules to optimized output molecules [149, 193].

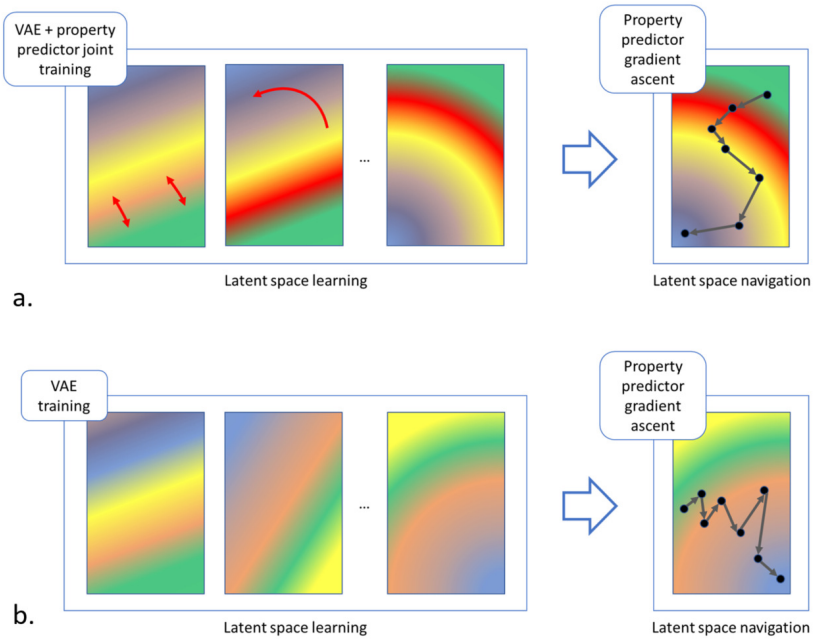

Figure 2.5: Properties of interest can modify the latent space at learning time (**a.**). Red arrows represent the *steering force* of the properties predictors in optimizing the latent space during training. Alternatively properties predictors can be used just to navigate the latent space a posteriori via gradient ascent (**b.**); this time the distortion force is absent.

Ex post conditioning Ex post conditioning methods, on the other hand, separate the generation and optimization processes. These techniques first train a general molecular generation model and then navigate the learnt space to find molecules with desired properties. A common approach in this category is to per-

form gradient ascent on a fixed latent space [169, 130, 168, 172]. This is done by first training a VAE to learn a general latent space of molecules, then train a separate regressor on this fixed space to predict properties of interest. During inference, gradient ascent is used to navigate the latent space towards regions corresponding to desired properties. This process provides flexibility, allowing for optimization of properties not considered during the initial training phase, but the latent space navigation might be more difficult.

For scenarios where property evaluation is computationally expensive, Bayesian optimization (BO) [194] techniques have proven valuable [152, 164] used BO to efficiently explore the latent space of their models, employing surrogate functions (often sparse Gaussian processes [195]) to approximate complex property landscapes. The BO approach is particularly useful when dealing with properties that are expensive to compute or require experimental validation, as it can efficiently guide the search towards promising regions of the chemical space with fewer evaluations.

2.6 Datasets and benchmarks

The development and evaluation of GNN-based molecular generation models rely heavily on appropriate datasets and benchmarks. These resources play a crucial role in training models, assessing their performance, and enabling fair comparisons between different approaches.

Several large-scale datasets have become standard resources for training and evaluating generative models:

- the QM9 dataset [196] contains approximately 134,000 small organic molecules with up to nine heavy atoms (C, O, N, F), along with their associated quantum mechanical properties calculated at the DFT level of theory. This dataset is particularly useful for tasks involving quantum chemical properties and has been instrumental in developing models that can generate molecules with specific quantum properties or predict these properties for novel structures;

Table 2.2: Details of the most common benchmark molecular datasets.

Dataset	Size	Molecule Types	Typical Use Cases
ZINC	Millions	Drug-like	Molecular generation Property prediction
QM9	∼134,000	Small organics	Quantum property prediction Molecular generation
ChEMBL	∼2 million	Bioactive	Bioactivity prediction Target-specific generation
DrugBank	∼14,000	Approved drugs Experimental drugs	Drug repurposing Drug-target interaction
PubChem	>100 million	Diverse	Large-scale molecular analysis Data mining

- the ZINC database [197] is a free resource of commercially-available compounds for virtual screening. ZINC contains millions of drug-like molecules, making it an excellent resource for training generative models aimed at drug discovery applications. A commonly used subset is ZINC250k, containing about 250,000 molecules with atom types including C, O, N, F, P, S, Cl, I, and Br;
- the ChEMBL database [198] contains manually curated bioactive molecules with drug-like properties. ChEMBL is particularly useful for tasks involving property prediction or generation of molecules with specific bioactivities, as it includes detailed information about the biological activities of compounds against various targets;
- DrugBank [199] is a comprehensive database containing detailed information about approved drugs, experimental drugs, and drug targets. It includes chemical, pharmacological, and pharmaceutical data, making it valuable for tasks related to drug repurposing, drug-target interaction prediction, and generating drug-like molecules.

When evaluating molecular generation models, several key metrics have become standard in the field:

- *validity*, the proportion of generated molecules that are chemically valid;

- *uniqueness*, the proportion of unique molecules in a set of generated structures;
- *novelty*, the proportion of generated molecules not present in the training set;
- *diversity*, often defined as the average pairwise Tanimoto distance between Morgan fingerprints [200].

The product of validity, uniqueness, and novelty, often referred to as VUN, serves as a global aggregated metric for model performance. However, it's crucial to recognize that the VUN metric may not fully capture the genuine task objective of reproducing the data distribution. Some trivial models can achieve near-perfect VUN scores without truly capturing the complexity of molecular space [201].

One significant challenge in evaluating novelty is the *copy problem* where generated molecules differ from original ones by only minor changes, such as a single bond type, atom, or SMILES string symbol. This can artificially inflate novelty scores [157]. Additionally, issues with molecular representation and processing can affect these metrics. For instance, inconsistencies in handling aromaticity between the training data and generated molecules can lead to artificially increased novelty scores.

To address these limitations, researchers have developed more sophisticated evaluation approaches. The Fréchet ChemNet Distance (FCD) [202] provides a more holistic assessment of the similarity between generated and reference molecular distributions. In general, in addition to VUN, an inspection of the molecular property distribution can give a hint of the soundness of a model. Some researchers have also proposed task-specific metrics, such as measures of synthetic accessibility or drug-likeness, and benchmarking suites like GuacaMol [203] and MOSES [204] have been developed.

In conclusion, while current datasets, benchmarks, and evaluation suites have been instrumental in advancing the field of molecular generation, there is still room for improvement. Future efforts should focus on developing more comprehensive and unbiased datasets, more sophisticated evaluation metrics that capture practical relevance, and benchmarks that can assess performance on increasingly complex

and realistic molecular design tasks.

Building upon these foundations and addressing the challenges identified in the current state of molecular generation models, the next chapter introduces Atomic-Molecular Conditional Generator (AMCG), a VAE-like graph-based conditional generative model for molecular design. AMCG represents an initial step in addressing some of the key limitations of existing approaches, offering improved flexibility, control, and performance in generating drug-like molecules.

Chapter 3

AMCG: a graph dual Atomic-Molecular Conditional Generator

3.1 Introduction

Building on the foundations of GNNs and generative models discussed in Chapter 2, this chapter introduces AMCG (Atomic-Molecular Conditional Generator), a novel framework for molecular graph generation. AMCG addresses several key challenges in the field of de novo drug design:

- it provides a flexible, one-shot generation process that allows for rapid sampling of molecular structures;
- unlike many existing methods, it imposes no upper limit on the number of atoms in generated molecules;
- it incorporates an explicit atomic type histogram assignation, enabling fine-grained control over the generated molecules' composition;
- it offers multiple sampling strategies to balance between diversity and adherence to the training distribution;

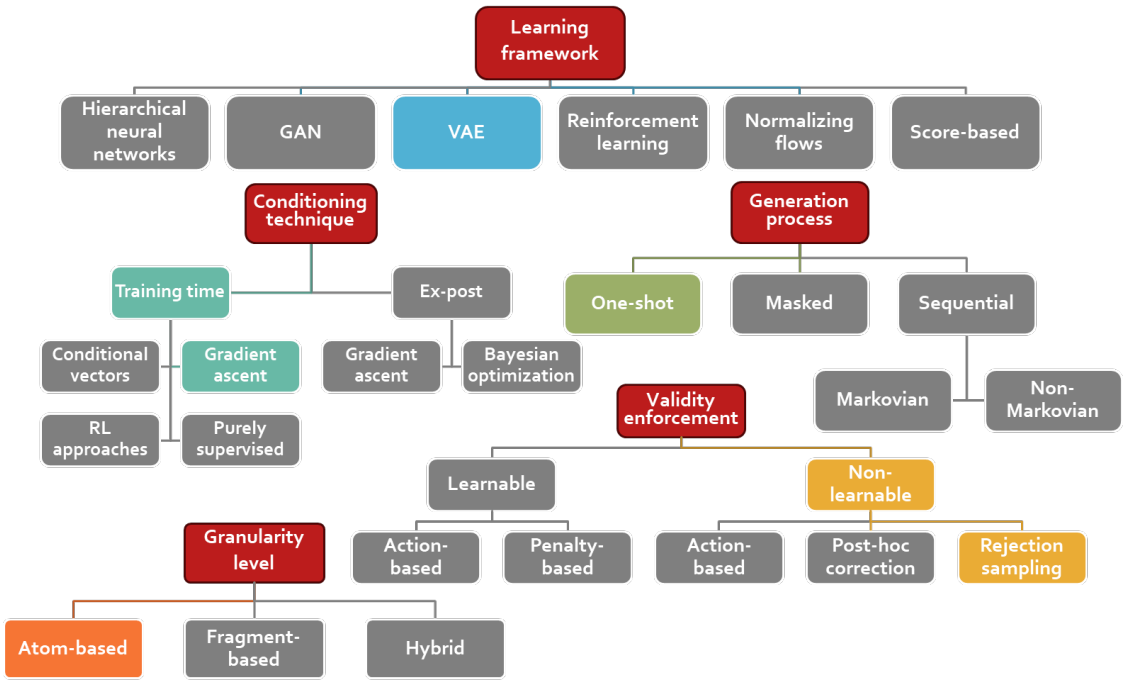

Figure 3.1: The design choices made for AMCG.

- it allows for property optimization through gradient ascent in the latent space;
- it achieves state-of-the-art performance on benchmark datasets while maintaining computational efficiency.

AMCG architecture employs a dual atomic-molecular representation, leveraging a distillation strategy during training to enhance the model's performance. This chapter will detail the structure of AMCG framework, its implementation, and its performance. The content of this chapter, the tables and most of the figures are taken from [P2]. The code to reproduce the experiments presented in this chapter is available at `https://github.com/carloabate/amcg`.

3.2 Model architecture

3.2.1 Overview of AMCG

AMCG is designed as a VAE-like framework that supports both unconditional and conditional generation of molecular graphs. While VAE-based approaches have been quite applied to molecular generation, they face a fundamental challenge in balancing reconstruction ability with latent space regularization. A model able to reconstruct very well the input data often creates a sparse latent space where empty regions may not correspond to meaningful molecular structures. On the other hand, overweighting the regularization term can help compact the space [122] but may also lead to an overly constrained mapping due to the shape of the prior distribution. To address these limitations, various approaches have been proposed. Score-based methods in the latent space [205, 206] offer a rigorous solution but can be computationally expensive. At the other extreme, graphical tools for manual latent space navigation [207] provide intuitive control but lack algorithmic rigor and automation. AMCG proposes a reasonable middle ground: instead of sampling from the standard normal prior, it samples from a modified Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, where $\boldsymbol{\mu}$ is the mean of the latent embeddings and $\boldsymbol{\sigma}$ is empirically optimized. Furthermore, AMCG refines this strategy by employing Gaussian Mixture Models (GMMs) [208] to better capture the learnt prior distribution. By incorporating these sampling strategies, AMCG aims to overcome the limitations of traditional VAE approaches while maintaining computational efficiency. In Figure 3.1 we highlighted the main design choices made for our model, in relationship to the generative framework taxonomy introduced in Chapter 2.

AMCG comprises two main branches: an atomic branch and a molecular branch, as illustrated in Figure 3.2. Its key components are:

- an *encoder*: it processes the input molecular graph to generate atomic and molecular representations;
- a *combiner*: it merges the atomic and molecular representations;
- a *molecular decoder*: it generates a surrogate atom-wise representation from

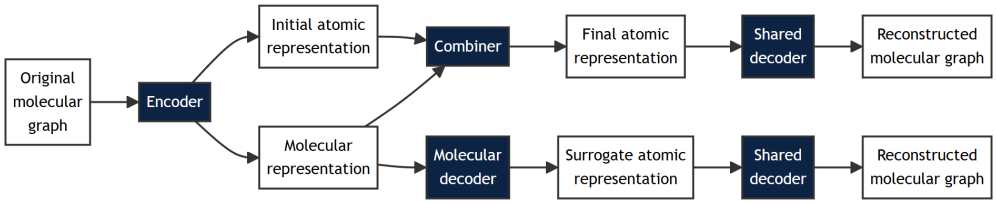

Figure 3.2: High-level representation of the AMCG framework. White blocks represent data objects, while blue blocks represent network components. The upper and lower branches are the atomic and molecular branches, respectively.

the molecular embedding;

- a *shared decoder*: it reconstructs the original molecule from the atom-wise representations, either coming from the atomic or the molecular branch.

During training, both branches aim to reconstruct the input molecular graph. However, during generation, only the molecular branch is used, sampling from the learnt latent space to produce new molecules.

3.2.2 Encoder

The encoder consists of two parts: an atomic encoder and a molecular encoder.

Atomic encoder

The atomic encoder uses GAT layers [84] with HeteroLinear (HL) activation functions, that is each atom type has its own linear activation. Formally, for an input $\boldsymbol{x}$ and atom type t, a HL function is defined as:

$$\mathrm{HL}_t(\boldsymbol{x}) = \boldsymbol{W}_t\boldsymbol{x} + \boldsymbol{b}_t \tag{3.1}$$

where $\boldsymbol{W}_t$ and $\boldsymbol{b}_t$ are learnable parameters specific to atom type t. This allows the network to learn different transformations for different atom types, enhancing its ability to capture type-specific chemical properties.

It generates two feature vectors per atom, $\boldsymbol{\mu}^A$ and $\boldsymbol{\sigma}^A$, which parameterize a mul-

tivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^A, \text{diag}(\boldsymbol{\sigma}^A))$. The output is sampled as:

$$\boldsymbol{x}^A = \boldsymbol{\mu}^A + \boldsymbol{\varepsilon} \odot \boldsymbol{\sigma}^A \tag{3.2}$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\odot$ denotes the Hadamard product.

This is done to increase the robustness of the downstream network (i.e. the decoder part) [209] and to break the symmetry in reconstructing the original graph [210].

The atomic encoder has an input size of 54 plus the number of atom types of the dataset. It then has embedding size of 512, with two GAT heads, a hidden size and an output size of 1024.

Molecular encoder

The molecular encoder transforms the atomic features using an MLP and then aggregates them using a global sum operator. The resulting molecular embedding is used to parameterize another multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^M, \text{diag}(\boldsymbol{\sigma}^M))$. The final molecular representation is sampled as:

$$\boldsymbol{x}^M = \boldsymbol{\mu}^M + \boldsymbol{\varepsilon} \odot \boldsymbol{\sigma}^M \tag{3.3}$$

A KL divergence term is added to the loss function to encourage the learnt distribution to adhere to a standard Gaussian prior:

$$\mathcal{L}_{KL}(\boldsymbol{\mu^M}, \boldsymbol{\sigma^M}) = \frac{1}{2}\left[-\sum_{i=1}^{m}(\log(\sigma_i^M)^2 + 1) + \sum_{i=1}^{m}(\sigma_i^M)^2 + \sum_{i=1}^{m}(\mu_i^M)^2\right]. \tag{3.4}$$

The molecular encoder has an input size and a first hidden channels number of 1024, while a second hidden channels number and an output size of 1536.

3.2.3 Combiner

The combiner merges the atomic and molecular representations by concatenating the molecular representation $\boldsymbol{x}^M$ after each atomic embedding vector $\boldsymbol{x}^A$. This combined representation is then processed by a four-layer MLP with HL activation

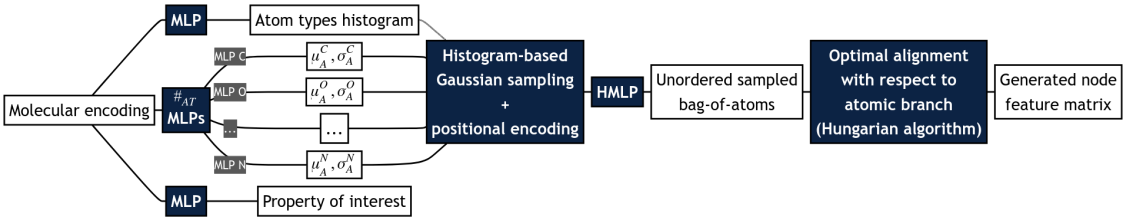

Figure 3.3: Architectural details of the molecular decoder.

(HMLP):

$$\boldsymbol{x}_F^A = N(\boldsymbol{x}^A \parallel \boldsymbol{x}^M) \tag{3.5}$$

where $\parallel$ denotes the concatenation operation. This combination operator proved to be more flexible than a sum or averaging operator, allowing also for different atomic and molecular embedding size.

The combiner has an input size, a hidden size and an output size of 2560, 1024 and 1024 respectively.

Let us then denote with $\boldsymbol{X}^F$ the resulting atomic feature matrix.

3.2.4 Molecular decoder

The molecular decoder, illustrated in Figure 3.3, is a crucial component of AMCG that generates new atomic representations from the molecular embedding. It consists of several key components:

1. **Histogram predictor**: this is an MLP regressor that predicts the histogram H of atom types in the molecule. The loss function for this component is the mean squared error (MSE), denoted as $\mathcal{L}_H$. This histogram prediction allows for explicit control over the atomic composition of generated molecules, providing a built-in conditioning mechanism. The histogram predictor has an input size of 1536, a hidden size of 1024 and an output size that is equal to the number of atom types present in the dataset;

2. **Atom type-specific MLPs**: for each atom type AT present in the dataset, the molecular representation is passed through a separate 2-layer MLP. Every MLP outputs the parameters of a normal distribution $\mathcal{N}(\boldsymbol{\mu}^{AT}, \text{diag}(\boldsymbol{\sigma}^{AT}))$.

The atom-type specific MLPs have an input size, a hidden size and an output size of 1536, 2048 and 2048, respectively;

3. **Sampling step**: the atom type-specific distributions are sampled according to the predicted histogram. To ensure diversity in the output, a positional bias constant term (based on the sampling order) is added to each atom. This step produces a matrix $\boldsymbol{X}$ where the number of rows equals the number of atoms;

4. **Post-processor**: the atomic embeddings are refined through an HMLP, resulting in a matrix $\widehat{\boldsymbol{X}}_F$. The post-processor has an input size, a hidden size and an output size of 2048, 1024 and 1024, respectively.

During training, the atomic branch guides the alignment of $\widehat{\boldsymbol{X}}_F$ with its counterpart $\boldsymbol{X}_F$ from the atomic branch. This alignment is necessary because the sampling process in the molecular branch does not preserve the original atom ordering. The alignment is achieved using the Hungarian algorithm [211], which matches atoms of the same type between $\boldsymbol{X}_F$ and $\widehat{\boldsymbol{X}}_F$. This process reduces the computational complexity from $O(n^3)$ to $O((\max_k n_k)^3)$, where n_k is the number of atoms of type k. An alignment loss $\mathcal{L}_{AL}$ is added to the global loss function, defined as the MSE between the original atom-wise representation and the surrogate one. Importantly, backpropagation through the Hungarian algorithm is not necessary, as the same graph is obtained regardless of the atom order due to the design of the shared decoder.

3.2.5 Shared decoder

The shared decoder, illustrated in Figure 3.4, is responsible for reconstructing the molecular graph from either the atomic or molecular representations. It performs the following steps:

1. First, the input $\boldsymbol{X}$ (either the atomic or surrogate embedding) is passed to an MLP regressor that predicts the number of hydrogen atoms connected to each heavy atom (i.e., each row of matrix $\boldsymbol{X}$). A hydrogen prediction loss $\mathcal{L}_{HY}$ is added to the global loss function. The hydrogen atoms predictor has

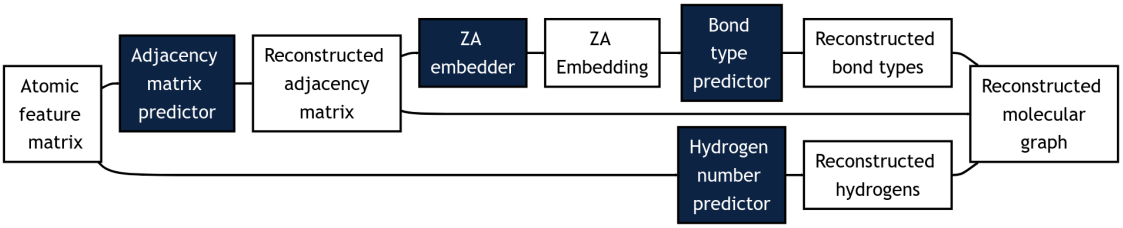

Figure 3.4: Architectural details of the shared decoder.

an input size of 1024, followed by a pyramidal structure of sizes 1024-512-1;

2. Next, X is fed to an adjacency matrix decoder, originally introduced in [212], that reconstructs the adjacency matrix of the molecular graph:

$$\overline{\boldsymbol{A}} = \sigma(\alpha \boldsymbol{X}\boldsymbol{X}^T + \beta) \tag{3.6}$$

where α and β are learnable scalar parameters, and σ is the sigmoid activation function. A reconstruction loss term is added to the global loss:

$$\mathcal{L}_A(\overline{\boldsymbol{A}}, \boldsymbol{A}) = \frac{\sum_{i,j} l(a_{ij}, \overline{a}_{ij})}{N^2} \tag{3.7}$$

where N is the number of atoms and l is defined as:

$$l(\overline{a}_{ij}, a_{ij}) = \begin{cases} \log(\overline{a}_{ij}), & \text{if } a_{ij} = 1 \\ 1 - \log(\overline{a}_{ij}), & \text{if } a_{ij} = 0. \end{cases}$$

3. The decoder then uses $\boldsymbol{X}$ and $\boldsymbol{A}$ to generate new atomic features via a composition of two linear transformations (*XA embedder* in Figure 3.4):

$$\begin{aligned} \boldsymbol{X}' &= \text{ReLU}(\boldsymbol{H}_0 \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{b}_0) \\ \boldsymbol{X}'' &= \boldsymbol{H}_1 \boldsymbol{A}\boldsymbol{Z}' + \boldsymbol{b}_1 \end{aligned} \tag{3.8}$$

where $\boldsymbol{H}_0, \boldsymbol{H}_1, \boldsymbol{b}_0, \boldsymbol{b}_1$ are learnable parameters.

4. Finally, an MLP classifier C is used to predict the bond types. For each edge ε_{ij} predicted by the adjacency matrix decoder, with $\boldsymbol{x}_i''$ and $\boldsymbol{x}_j''$ being the i-th

and j-th rows of $\boldsymbol{X}''$ respectively, the bond type is defined as:

$$B(i,j) = \text{argmax}\, C(\boldsymbol{x}_i'', \boldsymbol{x}_j'') \tag{3.9}$$

A bond reconstruction loss term $\mathcal{L}_B$, defined as the cross-entropy between the predicted and target bond types, is added to the global loss. The bond type classifier has an input size of 2048 for QM9 dataset, 4096 for ZINC dataset, a hidden size of 2048 and an output size of 4. For the QM9 dataset, the bond classifier takes as input $\boldsymbol{x}_i'' + \boldsymbol{x}_j''$, while for the ZINC dataset, it takes as input $\boldsymbol{x}_i'' || \boldsymbol{x}_j''$.

The output of the shared decoder consists of an adjacency matrix, bond types, atom types, and hydrogen atom counts for each heavy atom. This information is used to attempt the generation of a valid molecule. The process involves first adding all bonds and hydrogen atoms, then validating the consistency of the resulting structure. If inconsistencies are found, the model retains only the hydrogen atoms associated with aromatic bonds and makes another attempt. If issues persist, the hydrogen atoms are disregarded, and only bond information is used.

This process ultimately generates a reconstructed molecular graph $\overline{\mathcal{G}} = (\boldsymbol{X}_{\overline{\mathcal{G}}}, \boldsymbol{A}_{\overline{\mathcal{G}}}, \boldsymbol{E}_{\overline{\mathcal{G}}})$, where $\boldsymbol{X}_{\overline{\mathcal{G}}}$ represents the atom features, $\boldsymbol{A}_{\overline{\mathcal{G}}}$ is the adjacency matrix, and $\boldsymbol{E}_{\overline{\mathcal{G}}}$ contains the edge (bond) features.

3.3 Training and loss functions

AMCG model is trained using a weighted sum of several loss components:

$$\begin{aligned}\mathcal{L}_{UNC} = w_{KL}\mathcal{L}_{KL} + w_{AL}\mathcal{L}_{AL} + w_H\mathcal{L}_H + \\ + w_{HY}^A\mathcal{L}_{HY}^A + w_{HY}^M\mathcal{L}_{HY}^M + w_A^A\mathcal{L}_A^A + w_B^A\mathcal{L}_B^A + w_A^M\mathcal{L}_A^M + w_B^A\mathcal{L}_B^M\end{aligned} \tag{3.10}$$

where the superscripts A and M denote losses associated with the atomic and molecular branches, respectively.

To manage the complexity of this multi-component loss, a curriculum learning

approach is employed. The training process focuses first on the atomic branch, then shifts to the molecular branch, and finally promotes latent space compaction by increasing the weight of the KL divergence term. The specific weight schedules used for the QM9 and ZINC datasets are provided in Table 3.1.

Table 3.1: The weights schedule used to train AMCG. (a) was utilized to train the model on QM9 dataset, while (b) was utilized to train the model on ZINC dataset.

(a)

Epoch	w_A^A	w_A^M	w_{KL}	Other w
0	20	5	0	1
50	5	20	0	1
100	5	20	1/5	1
150	5	20	1	1
200	5	20	5	1

(b)

Epoch	w_A^A	w_A^M	w_{KL}	Other w
0	20	5	0	1
50	5	20	0	1
150	5	20	1/5	1
200	5	20	1	1
250	5	20	2	1

3.4 Molecular generation and sampling strategies

3.4.1 Unconditional generation

For unconditional generation, AMCG employs various sampling strategies from the learnt latent space. These strategies aim to balance between generating diverse, novel molecules and maintaining chemical validity. The following approaches were investigated:

- `VAE`: sampling from a standard VAE Gaussian prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$;
- `VAE-like`: sampling from a Gaussian prior $\mathcal{N}(\overline{\boldsymbol{\mu}}, \mathrm{diag}(\alpha \cdot \overline{\boldsymbol{\sigma}}))$, where $\overline{\boldsymbol{\mu}}$ and $\overline{\boldsymbol{\sigma}}$ are the mean and standard deviation of the latent representations of the dataset, respectively. α is a scalar hyperparameter;
- `GMM-F`: training a full covariance matrix GMM in the latent space;
- `GMM-D1/D2`: training a GMM in the latent space with diagonal covariance matrices scaled by a hyperparameter β.

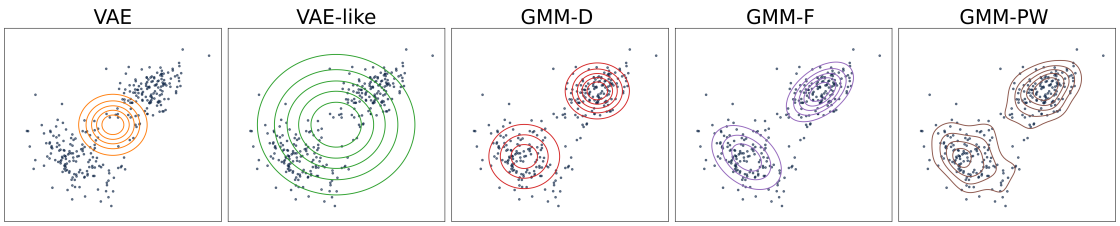

Figure 3.5: Visual representation of the different prior distributions in the latent space.

Figure 3.5 illustrates the different prior distributions in a 2D projection of the latent space. These sampling strategies offer different trade-offs between exploration of the latent space and adherence to the training set distribution.

3.4.2 Conditional generation

AMCG supports two main approaches for conditional generation:

Structural conditioning

Our model allows for explicit control over the atom type histogram of the generated molecules. This feature is particularly useful for targeted molecule design, where specific atomic compositions are desired.

One significant application of this flexibility is in the realm of patent evasion and chemical space exploration. For instance, the substitution of one atom with an electronically similar atom (e.g., replacing an oxygen with a sulfur) can lead to molecules with similar properties but distinct enough to escape existing patents [213, 214, 215]. This capability allows for greater freedom in molecular design and can be crucial in developing new drug candidates that don't infringe on existing intellectual property.

Furthermore, the ability to perturb the histogram brings to light the role of *activity cliffs* in molecular design [216]. Activity cliffs refer to instances where relatively small changes in the atomic composition of a molecule lead to significant changes in its potency profile. By allowing controlled modifications to the atom type histogram, AMCG can potentially explore these activity cliffs, enabling the discovery of molecules with dramatically improved properties through minor structural

changes.

Property-based conditioning

AMCG employs gradient ascent in the latent space to optimize for specific molecular properties (see Section 2.5.4). This is achieved by training an additional property predictor alongside the main model. Upon training the predictor the property optimization process described in Algorithm 1 takes place. This strategy allows for the generation of molecules with optimized properties while maintaining chemical validity through a rejection sampling approach (see Section 2.5.3).

Algorithm 1 Property optimization strategy

Input: $N \geq 0$, $V \geq 0$, $S \geq 0$
Input: The molecule to optimize $\boldsymbol{x}$.
Output: A molecular optimization path M.

```
M ← [ ]
x_0 ← x
for i ← 1 to N do                          ▷ N steps of gradient ascent
    x_i ← ascend(x_{i−1})
    if i mod S == 0 then                   ▷ Sampling each S steps
        for j ← 1 to V do                  ▷ V validity rejection trials
            y ← sample(x_i)
            if y is valid then
                Add y to M and exit the loop.
            end if
        end for
    end if
end for
return M
```

3.5 Experimental results

3.5.1 Datasets and preprocessing

AMCG was evaluated on QM9 and ZINC, two widely used benchmark datasets for molecular generation (see Section 2.6).

The preprocessing of molecular data for AMCG is an extensive process that gen-

erates a rich set of chemical, topological, and physical descriptors for both atoms and bonds, detailed in the following:

- **chemical descriptors**: generated using RDKit v.2023.03.3 [217]. These include standard chemical features such as atom type, degree, formal charge, and bond type;
- **physical descriptors**: generated using AmberTools suite (version 22.3) and BiKi Life Sciences [218]. This process involves:
  - converting `.pdb` files to `.mol2` format using the antechamber tool to obtain Generalized Amber Force Field (GAFF) atom types;
  - for the ZINC dataset, where 3D coordinates are not available, SMILES strings are first converted to 3D structures using the method of Riniker and Landrum [219], followed by minimization using UFF force field parameters [220];
  - matching atoms to GAFF atom types to recover equilibrium distance and spring constant for each bond;
  - when exact GAFF parameters are not available, the `parmchk2` tool is used to generate missing bond parameters by chemical analogy;
- **topological descriptors**: a graph positional embedding is computed to capture the positional information of each node in the graph, as done in [221]. This feature is extracted using a torch-geometric built-in function.

The full list of descriptors used is provided in Table 3.2, while the preprocessed datasets are available at `https://doi.org/10.5281/zenodo.11109939`.

3.5.2 Unconditional generation results

As a first evaluation of our model, we sampled 10000 molecules and assessed their validity, uniqueness, novelty, and VUN (see Section 2.6). We compared our framework to other state-of-the-art latent variable generators. Results are presented in Table 3.3, while molecular graphs sampled from our model can be seen in Figure

Table 3.2: The descriptors used to train AMCG.

Feature	Obtained from	# descriptors	Type
atom type	`Atom.GetSymbol()`	4 (QM9) 9 (ZINC)	one-hot
degree	`Atom.GetDegree()`	6	one-hot
formal charge	`Atom.GetFormalCharge()`	1	integer
radical electrons	`Atom.GetNumRadicalElectrons()`	1	integer
hybridization	`Atom.GetHybridization()`	6	one-hot
hydrogens	`Atom.GetTotalNumHs()`	5	one-hot
aromaticity	`Atom.GetIsAromatic()`	1	boolean
chirality type	`try: Atom.GetProp('_CIPCode')`	2	one-hot ['R','S'] else [0,0]
has property of chiral center	`Atom.HasProp('_ChiralityPossible')`	1	boolean
possible chiral center	`Atom.GetProp('_ChiralityPossible')`	1	boolean
graph positional encoding	`transforms.AddRandomWalkPE`	30	real

Node descriptors

Feature	Obtained from	# descriptors	Type
bond type	`Bond.Get.BondType()`	4	one-hot
conjugation	`Bond.GetIsConjugated()`	1	bool
wether bond is in ring	`Bond.IsInRing()`	1	bool
stereochemistry	`Bond.GetStereo()`	4	one-hot
equilibrium distance	see Section 3.5.1	1	real
spring constant	see Section 3.5.1	1	real
geometric distance	see Section 3.5.1	1	real

Edge descriptors

3.6.

Table 3.3: Comparison of AMCG models with competing latent variable methodologies on QM9 dataset. The size of the generated sample set was 10^4. The `VAE` model employs a regular Gaussian prior $\mathcal{N}(0, 1)$ and collapses in terms of uniqueness. The `VAE-like` uses a Gaussian prior $\mathcal{N}(\mu, \sigma)$ and improves uniqueness significantly. `GMM` models employ various flavors of mixture models and obtain the best results.

Model	Validity	Validity w/o check	Uniqueness	Novelty	VUN
MPG-VAE [170]	-	0.9100	0.6800	0.540	0.3340
GraphNVP [130]	-	0.8310	0.9920	0.582	0.4797
GRF [222]	-	0.8450	0.6600	0.586	0.3268
GraphAF [158]	1.000	0.6700	0.9451	0.8883	0.8395
MoFlow [168]	1.000	0.8896	0.9853	0.9604	0.9462
GraphDF [159]	1.000	0.8267	0.9762	0.9810	0.9576
Ours - `VAE`	1.000	0.4006	0.1293	0.8987	0.1162
Ours - `VAE-like`	1.000	0.5803	0.7756	0.8829	0.6848
Ours - `GMM-F`	1.000	0.4075	0.9428	0.8001	0.7543
Ours - `GMM-D1`	1.000	0.1653	0.9693	0.9640	0.9344
Ours - `GMM-D2`	1.000	0.0555	0.9982	0.9964	0.9946

The results show that AMCG achieves competitive or superior performance compared to existing methods. Notably, the `GMM-D2` variant achieves near-perfect VUN scores, indicating its ability to generate valid, unique, and novel molecules.

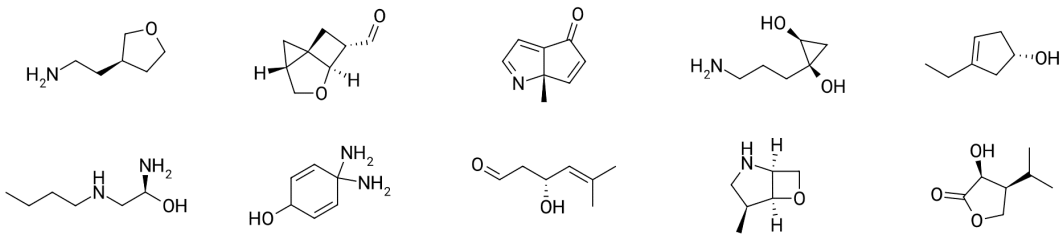

Figure 3.6: Randomly generated molecules from the `GMM-F` prior.

To provide a more comprehensive evaluation of the model's performance, we introduced the concept of a UN persistence plot (Figure 3.7). This plot shows how the product of uniqueness and novelty (UN) changes as more valid molecules are generated. The UN persistence plot reveals that the GMM-based sampling strategies (`GMM-D1`, `GMM-D2`, and `GMM-F`) maintain high UN values even as the number of generated molecules K increases (up to $K = 200000$), indicating their ability to produce diverse and novel molecules consistently.

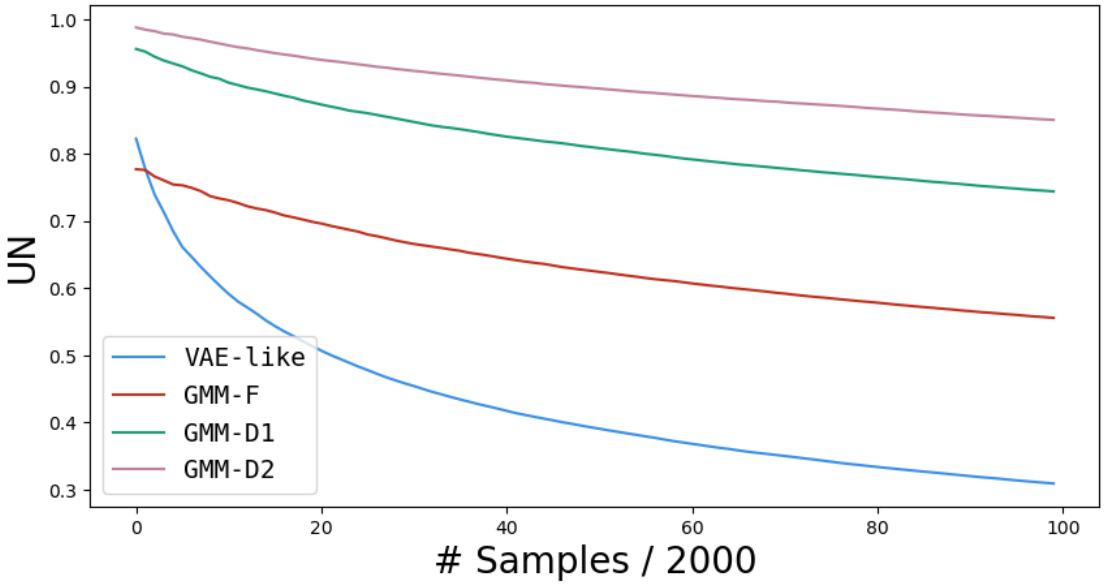

Figure 3.7: UN persistence plot on QM9 dataset. The x-axis shows the number of valid generated molecules, and the y-axis shows the product of uniqueness and novelty (UN).

To further assess the quality of the generated molecules, we compared the distribution of various molecular properties (logP, QED, Synthetic Accessibility Score, and molecular weight) between the generated molecules and the original dataset (Figure 3.8). These results demonstrate that the `GMM-F` and `GMM-D1` variants of AMCG produce molecular property distributions that closely match those of the original dataset, indicating the model's ability to capture and reproduce the underlying chemical space effectively.

Means and standard deviations of the properties of interest are summarized in Table 3.4, together with the evaluation of internal diversity of the generated samples. Again, the less explorative approach(es) better capture the behaviour of the properties of the dataset, whereas the others produce different results. Accordingly to the nature of the chosen priors, the diversity value also increases when moving to the less conservative ones.

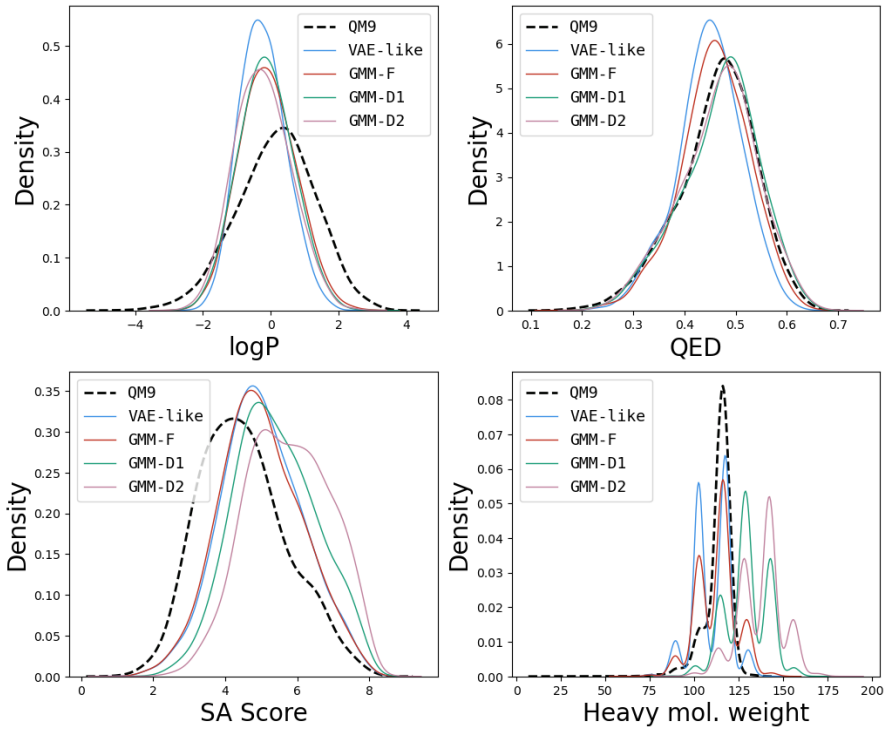

Figure 3.8: Molecular property distributions for 10,000 valid, novel, and unique generated molecules trained on the QM9 dataset. The dashed line represents the original dataset, and continuous lines represent the developed models.

Table 3.4: Mean values and standard deviations (in brackets) for molecular properties of 10000 valid, novel and unique generated samples by training on the QM9 dataset. The first line, named QM9, shows reference values. In terms of mimicking the original manifold the `GMM-F` model is the best one.

Model	logP	QED	SA Score	Mol. weight	Diversity
QM9	0.15 (1.16)	0.46 (0.08)	4.50 (1.21)	123.12 (7.60)	0.1038
Ours - `VAE-like`	-0.23 (0.68)	0.45 (0.06)	5.04 (1.12)	119.41 (11.34)	0.1695
Ours - `GMM-F`	-0.09 (0.83)	0.46 (0.07)	4.98 (1.13)	123.13 (12.25)	0.1511
Ours - `GMM-D1`	-0.14 (0.79)	0.47 (0.08)	5.41 (1.12)	141.43 (12.77)	0.2159
Ours - `GMM-D2`	-0.23 (0.82)	0.47 (0.08)	5.74 (1.13)	149.65 (13.81)	0.2561

3.5.3 Conditional generation results

Structural conditioning

To evaluate AMCG's ability to perform structural conditioning and its robustness to histogram modifications, we conducted an experiment with perturbed atom type histograms.

We applied four different types of perturbations to the predicted histograms when sampling from the most conservative priors (`VAE-like` and `GMM-F`):

- `±1`: the predicted histogram was modified by randomly adding or subtracting 1 to each atom type count;
- `±2`: the predicted histogram was modified by randomly adding or subtracting 2 to each atom type count;
- `random-p`: a new histogram was generated randomly according to the atom type probabilities observed in the dataset;
- `random-u`: a new histogram was generated randomly with uniform probability across atom types.

Table 3.5 presents the results of this experiment, showing the impact of these perturbations on validity, uniqueness, novelty, and the combined VUN score. The results demonstrate that AMCG can generate valid, unique, and novel molecules even when the atom type histograms are significantly perturbed. As we move from minor perturbations (`±1`) to more drastic changes (`random-u`), we observe a trade-off between validity and novelty/uniqueness. The *validity w/o check* column shows that the initial validity of the generated molecules decreases as perturbations become more extreme, especially for the random uniform perturbation.

Interestingly, as we depart further from the original manifold through random perturbations of the histogram, we see a general trend of decreasing validity (before checks) but increasing uniqueness and novelty. This suggests that while more extreme perturbations lead to a higher proportion of initially invalid molecules, they also push the model to explore new regions of chemical space, resulting in

Table 3.5: VUN evaluation for histogram perturbation. Here we perturb the histogram step of the method with several random techniques. The more we depart from the original manifold through random perturbation of the histogram the more we lose in validity and gain in uniqueness and novelty.

Perturbation	Validity	Validity w/o check	Uniqueness	Novelty	VUN
None	1.000	0.5803	0.7756	0.8829	0.6848
±1	1.000	0.3811	0.7533	0.8877	0.6686
±2	1.000	0.2521	0.6528	0.8759	0.5717
random-p	1.000	0.3653	0.7794	0.8361	0.6514
random-u	1.000	0.0333	0.7047	0.9833	0.6929

VAE-like prior

Perturbation	Validity	Validity w/o check	Uniqueness	Novelty	VUN
None	1.000	0.4075	0.9428	0.8001	0.7543
±1	1.000	0.2827	0.8733	0.8555	0.7471
±2	1.000	0.1932	0.7438	0.8546	0.6353
random-p	1.000	0.1973	0.9086	0.8243	0.7486
random-u	1.000	0.0279	0.8356	0.9844	0.8225

GMM-F prior

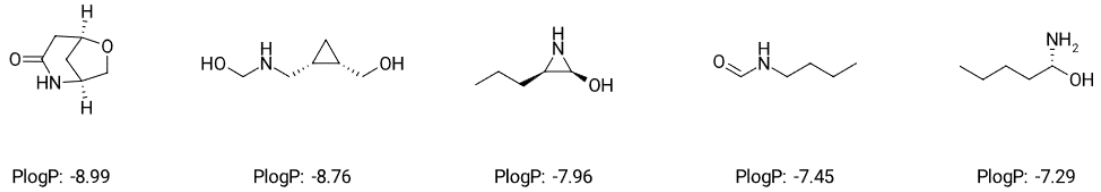

Figure 3.9: Property optimization path example. The optimization progresses from left to right, leading to the opening of the ring system and an increase in the dipole moment.

more unique and novel structures.

These findings highlight the robustness and flexibility of AMCG in handling structural conditioning. The model's ability to generate chemically sound molecules even with significantly altered atom type distributions demonstrates its potential for targeted molecular design and exploration of diverse chemical spaces.

Property-based conditioning

AMCG employs gradient ascent in the latent space to optimize for specific molecular properties. We evaluated its performance in optimizing two commonly used properties in literature: penalized logP and QED (see Section 2.5.4).

The experiment was conducted by attempting to maximize the value of the desired property for 10000 random molecules from the training set. To determine the optimal gradient ascent hyperparameters (learning rate, number of steps, and decoding step size), we first optimized 1000 molecules from the training set and recorded the maximum number of steps that could be taken without encountering a failed decoding.

Using these optimized hyperparameters, we then generated the complete set of molecules along the optimization path for each of the 10000 test molecules. The final output for each molecule was determined to be the molecule along its path with the best property value.

Figure 3.9 shows an example of an optimization path for penalized logP: in this example, we can observe that the optimization process first opens the ring system and then increases the dipole moment of the molecule, also rendering it more

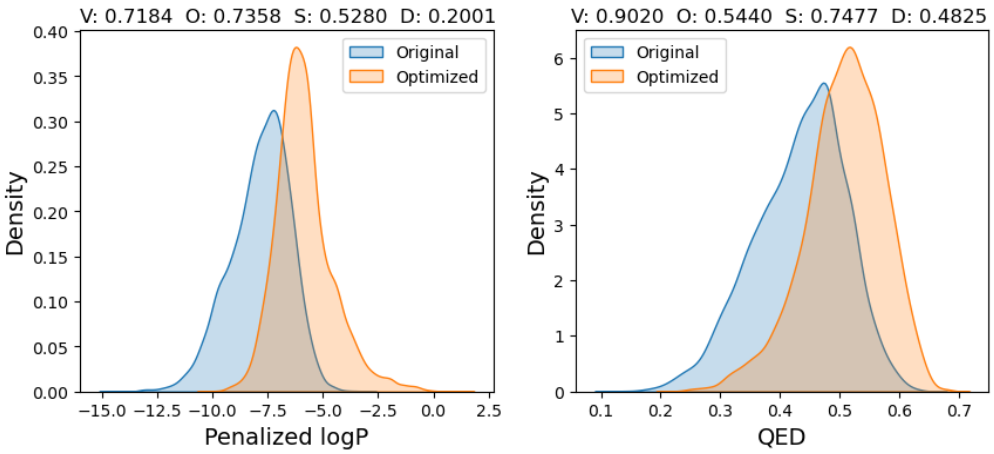

Figure 3.10: Property optimization results on QM9 dataset. V: validity, O: optimization rate, S: success rate, D: diversity. Blue: original distribution, Orange: optimized distribution.

linear. This demonstrates AMCG's ability to make significant structural changes while optimizing for a specific property.

Figure 3.10 presents the overall results of property optimization for both penalized logP and QED:

In this figure, we report several key metrics:

- V (validity): the percentage of decoded samples along the optimization path that are chemically valid;
- O (optimization rate): the ratio of molecules for which we found a molecule with an improved property value;
- S (success rate): the ratio of optimized molecules that are also novel (not present in the training set);
- D (diversity): the Tanimoto distance between the initial molecule and the optimized one, measuring the structural diversity introduced by the optimization process.

The blue distributions represent the original property values in the dataset, while the orange distributions show the values after optimization. We can observe a clear shift towards higher values for both penalized logP and QED, demonstrating

AMCG's ability to effectively optimize these properties. The high success rates and diversity scores indicate that AMCG is not simply memorizing known molecules but is able to explore novel chemical space guided by the property optimization objective.

3.5.4 Results on ZINC Dataset

To assess AMCG's performance on a more diverse and challenging dataset, we evaluated it on the ZINC250k dataset. This evaluation revealed interesting challenges, particularly in compacting the latent space, which proved significantly more difficult than with QM9. Due to this issue, we were unable to fit the Gaussian mixture of the `GMM-F` model during training. Consequently, in addition to the previously described `VAE`, `VAE-like`, and `GMM-D` priors, we introduced a `GMM-PW` (pointwise) prior, defined as a normalized sum of Gaussians centered on the embedding vectors of the training data. Table 3.6 presents our evaluation results, comparing AMCG with other state-of-the-art methods. Despite the overall latent space not

Table 3.6: Comparison of AMCG models with competing latent variable methodologies on the ZINC dataset. The size of the generated sample set was 10^4. The more we depart from the manifold the more we reduce validity. The `GMM-PW` model, as it samples in the close vicinity of the training points, is the most conservative model.

Model	Validity	Validity w/o check	Uniqueness	Novelty	VUN
GraphNVP	-	0.426	0.948	1.000	0.4038
GRF	-	0.734	0.537	1.000	0.3942
GraphAF	1.000	0.68	0.991	1.000	0.9910
MoFlow	1.000	0.5030	0.9999	1.000	0.9999
GraphDF	1.000	0.8903	0.9916	1.000	0.9916
Ours - `VAE`	1.000	0.2323	0.0437	0.8902	0.0389
Ours - `VAE-like`	1.000	0.0262	0.7054	1.000	0.7054
Ours - `GMM-D`	1.000	0.0144	0.9900	1.000	0.9900
Ours - `GMM-PW`	1.000	0.2630	0.9190	0.7636	0.7017

being well-structured, we found that sampling could occur safely in the vicinity of the learnt embedded training molecules. Consistent with our QM9 findings, the `VAE-like` prior showed clear benefits over the standard `VAE` approach. Interestingly, the `GMM-D` prior exhibited poor validity without resampling but achieved excellent results when validity rejection was applied. Figure 3.11 showcases ran-

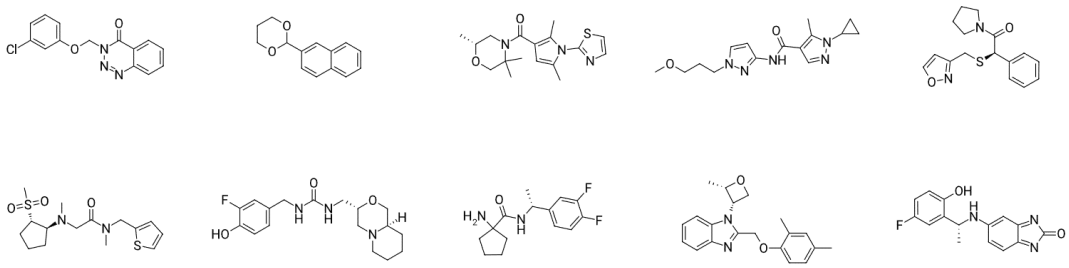

Figure 3.11: Randomly generated molecules from the `GMM-PW` prior.

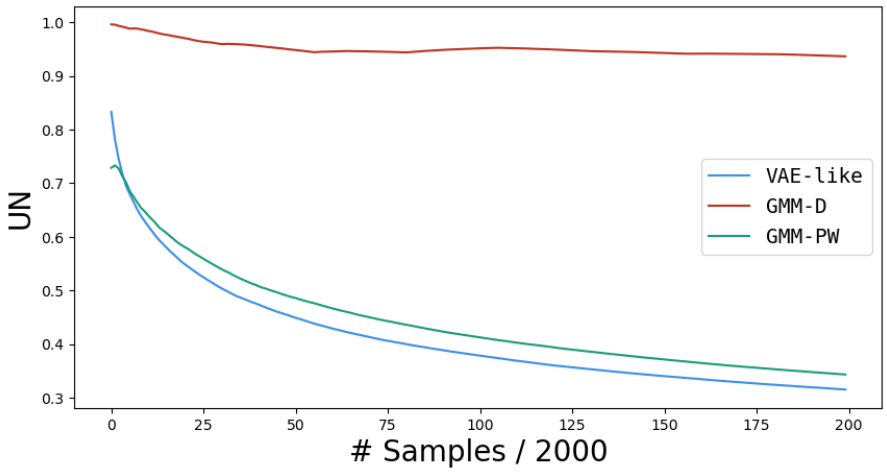

Figure 3.12: UN persistence plot on the ZINC dataset. The x-axis shows the number of valid generated molecules, and the y-axis shows the product of uniqueness and novelty. The `GMM-PW` and `VAE-like` lead to quick degradation, whereas `GMM-D` is able to sustain generation more effectively.

domly generated samples from the `GMM-PW` model, demonstrating AMCG's ability to generate diverse and realistic molecular structures. To improve molecular quality, we applied a post-hoc algorithm to remove chords from our generated graphs. To further validate our findings, we generated a UN persistence plot for the ZINC dataset (Figure 3.12), evaluating uniqueness and novelty for 400000 valid samples. We omitted results for the `VAE` prior due to its poor performance in the small-sized sample set regime. The plot revealed that the `GMM-D` prior is remarkably robust, showing no degradation even with extensive use of the validity rejection strategy. Our comprehensive evaluation of the generated molecules' properties (Table 3.7 and Figure 3.13) revealed several key insights. Sampling from the `VAE-like` prior

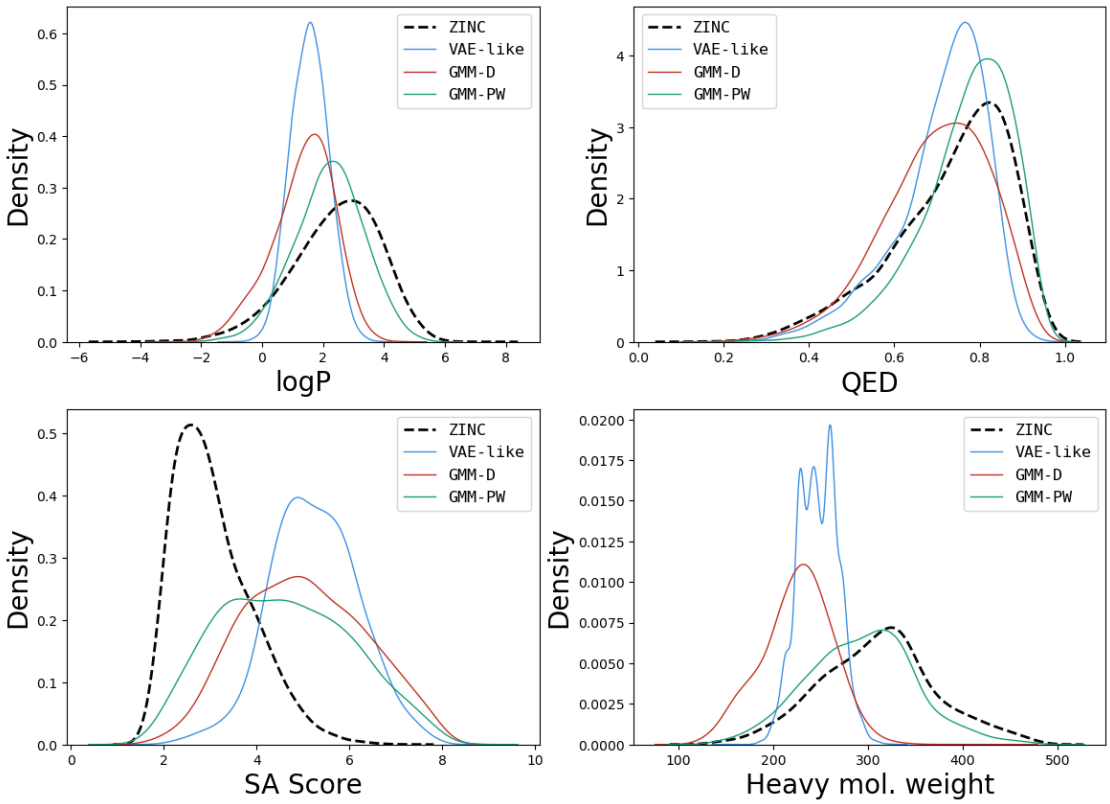

Figure 3.13: Molecular property distributions for 10,000 valid, novel, and unique generated molecules by training on the ZINC dataset. The dashed line represents the original dataset, while continuous lines represent our models. `GMM-PW` is the most conservative model, whereas `GMM-D` tends to produce smaller molecules.

resulted in overly concentrated property distributions, likely due to sampling from a reduced portion of a hard-to-compact latent space. The `GMM-PW` model provided the best performance in mimicking the original dataset's behavior, confirming its conservative nature. Conversely, molecules generated by the `GMM-D` model tended to be slightly smaller than those in the training distribution, attributed to the inherent difficulty in building larger molecules when departing from the embedding. In conclusion, our experiments on the ZINC dataset demonstrate AMCG's

Table 3.7: Mean values and standard deviations (in brackets) for molecular properties of 10000 valid, novel and unique generated samples by training on the ZINC dataset. `GMM-PW` confirms to be the most conservative model whereas `GMM-D` produces systematically smaller molecules.

Model	logP	QED	SA Score	Mol. weight	Diversity
ZINC	2.47 (1.43)	0.73 (0.14)	3.04 (0.83)	331.97 (61.53)	0.2396
VAE-like	1.55 (0.62)	0.71 (0.11)	5.23 (0.96)	265.18 (21.12)	0.3705
GMM-D	1.33 (1.03)	0.69 (0.13)	5.04 (1.30)	245.51 (39.82)	0.2684
GMM-PW	2.20 (1.15)	0.77 (0.11)	4.59 (1.42)	313.11 (57.23)	0.3093

flexibility and effectiveness on complex and diverse molecular datasets. For applications requiring limited *fantasy* and tight adherence to the dataset, the `GMM-PW` model performs best. However, for better fantasy and sustained UN persistence, the `GMM-D` model is more suitable. It's worth noting that to achieve converging models, we used logP as a property to guide the training process. These results highlight areas for potential future improvement, particularly in generating larger molecules consistently.

3.6 Conclusions

The AMCG framework introduced in this chapter represents a significant advancement in molecular graph generation, offering a flexible and powerful approach for de novo drug design. By combining a dual atomic-molecular representation with innovative sampling strategies, AMCG achieves state-of-the-art performance on standard benchmarks while providing several key advantages over existing methods. AMCG's primary strength lies in its one-shot generation capability, allowing

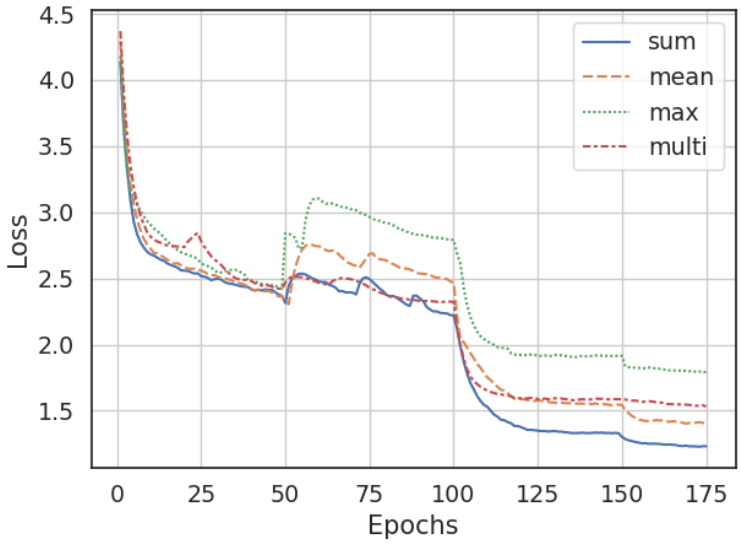

Figure 3.14: Training loss comparison for different global aggregators. The multi-aggregator approach combines global sum, mean, and max operations. The loss curve shape reflects the curriculum learning strategy used (see Table 3.1).

for rapid sampling and efficient exploration of chemical space. This approach, coupled with flexible conditioning mechanisms - both structural and property-based - provides a versatile tool for targeted molecular design. The explicit control over atom type histograms and the gradient-based property optimization demonstrate AMCG's potential for fine-tuned molecular generation. Our comprehensive evaluation, including the novel UN persistence plots, reveals that AMCG, particularly with GMM-based sampling, can maintain high uniqueness and novelty over extended generation runs. This feature is crucial for practical applications in drug discovery, where exploring a wide range of chemical possibilities is often necessary. An important aspect of AMCG's architecture that contributed to its performance is the choice of global aggregator used to extract the molecular representation. Our ablation study, comparing different aggregator types including global sum, global mean, global max, and a multi-aggregator approach, demonstrated that the global sum aggregator consistently outperformed other options. Figure 3.14 illustrates the training loss for each aggregator type. The global sum aggregator exhibited more stable behavior and achieved lower overall loss, supporting our architectural choice and highlighting the importance of carefully selecting each component in the

model design. Its superior performance may be attributed to its ability to preserve more information about the overall molecular structure, which is crucial for accurate reconstruction and generation of molecules. However, our experiments also revealed some limitations, particularly when dealing with more complex datasets like ZINC. The latent space for ZINC molecules was not compact enough to allow for consistent gradient ascent during property optimization, likely due to the model keeping embeddings too well-distanced to ensure accurate reconstruction during training. Additionally, generated ZINC molecules tended to be smaller than the original ones, which can be attributed to our one-shot generation and decoding strategy, which tends to produce hyperconnected (and thus often invalid) graphs when attempting to generate larger molecules. These limitations, combined with our ablation study results, point to several promising directions for future research. These include exploring more expressive encoders that can compress information from larger molecules into smaller latent spaces without losing critical structural information, investigating different decoding approaches that can better handle the generation of larger molecules without sacrificing validity, developing advanced aggregation strategies that can better handle the complexity of larger molecules while maintaining stability, and exploring more advanced optimization techniques or incorporating multi-objective optimization to balance different molecular properties simultaneously during conditional generation.

The insights gained from working with molecular graphs, particularly their heterophilic nature, sparked a broader investigation into fundamental graph neural network operations. Molecules are inherently heterophilic structures: connected atoms are often of different types with varying properties, forming diverse bonds that are crucial to the molecule's overall characteristics and function. This heterophily poses unique challenges for traditional graph-based machine learning models, which often assume some degree of homophily in graph data. Our ablation study on AMCG once again confirmed the importance of a powerful encoder, particularly in capturing and preserving this complex heterophilic information. These observations led us to develop MaxCutPool, a novel graph pooling method based on the `MAXCUT` problem. While initially motivated by the challenges posed by molecular

data, MaxCutPool is a general-purpose theoretical advancement in graph neural networks. It aims to identify diverse subsets of nodes in any graph structure, making it applicable far beyond just molecular graphs. The following chapter introduces MaxCutPool in detail, presenting it as a significant contribution to the fundamental theory of graph neural networks, with potential applications across various domains where graph-structured data exhibits heterophilic properties.

Chapter 4

MaxCutPool: differentiable feature-aware MAXCUT for pooling in graph neural networks

4.1 Introduction

Building upon the foundations of graph pooling discussed in Chapter 2 and the insights given from Chapter 3, we now introduce MaxCutPool, a novel graph pooling technique that addresses key limitations of existing approaches. While previous methods have focused on either trainable feature-based pooling or graph-theoretical objectives, MaxCutPool uniquely combines these aspects to create a powerful and flexible pooling solution.

Our contributions in this chapter are the following:

1. We present a robust, GNN-based method for computing MAXCUT partitions in attributed graphs. This approach not only works with feature-rich graphs but also demonstrates improved performance on non-attributed graphs compared to traditional MAXCUT algorithms.
2. We present a new benchmark dataset specifically designed to test the performance of graph neural networks on heterophilic graphs. This dataset fills a

gap in existing benchmarks and provides a valuable resource for evaluating GNN performance on complex, heterophilic graph structures.

3. We apply this MAXCUT optimization to create a new graph pooling layer. MaxCutPool represents the first pooling method which combines graph-theoretical MAXCUT objectives with differentiable, feature-aware operations. This results in a pooling layer that can adapt to both graph structure and node features, leading to improved performance in downstream tasks.

In the following sections, we will expand on the theoretical background of MaxCutPool, detail its architecture, and present comprehensive experimental results. These experiments demonstrate MaxCutPool's effectiveness across various graph types and learning tasks, highlighting its potential to advance the state-of-the-art in graph neural network architectures. The content of this chapter, its figures, and its tables, are taken from [P3]. Our implementation of MaxCutPool layer can be found at https://github.com/NGMLGroup/MaxCutPool.

4.2 Background

4.2.1 The MAXCUT problem and its continuous relaxations

The MAXCUT problem is a fundamental concept in graph theory and combinatorial optimization. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes and non-negative weights on the edges – i.e. the adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{N \times N}$, the objective of MAXCUT is to find a *cut* – a partition of nodes $(\mathcal{S}, \mathcal{V} \setminus \mathcal{S})$ where $\mathcal{S} \subset \mathcal{V}$ – that maximizes the total volume of edges connecting nodes in $\mathcal{S}$ with those in $\mathcal{V} \setminus \mathcal{S}$.

Mathematically, the MAXCUT problem can be expressed as an integer quadratic problem:

$$\max_{\boldsymbol{z}} \sum_{i,j \in V} a_{ij}(1 - z_i z_j) \quad \text{s.t.} \quad z_i \in \{-1, 1\} \tag{4.1}$$

where $\boldsymbol{z} \in \{-1, 1\}^N$ is an assignment vector indicating to which side of the partition each node is assigned, and w_{ij} is the weight of the edge connecting nodes i and j.

The MAXCUT problem is known to be NP-hard, making it computationally intractable

for large graphs. To address this challenge, several continuous relaxations have been proposed. One of the most notable is the Goemans-Williamson (GW) algorithm [223], which provides a semidefinite relaxation of the integer quadratic problem:

$$\max_{\boldsymbol{X}} \sum_{i,j \in V} a_{ij}(1 - \boldsymbol{x}_i \cdot \boldsymbol{x}_j) \quad \text{s.t.} \quad \|\boldsymbol{x}_i\| = 1 \tag{4.2}$$

where $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ is a matrix whose rows $\boldsymbol{x}_i$ are continuous embeddings of size d of the nodes in $\mathcal{G}$. The GW algorithm guarantees an expected cut size of .868 of the maximum cut. Another effective continuous relaxation is the Largest Eigenvector Vertex Selection (LEVS) method [224]. This approach uses the eigenvector $\boldsymbol{u}_{\max}$ associated with the largest eigenvalue $\lambda_{\max}$ of the graph Laplacian matrix $\boldsymbol{L}$. A cut in $\mathcal{G}$ can then be found based on the polarity of the components of $\boldsymbol{u}_{\max}$, for instance by letting $\mathcal{S} = \{i : \boldsymbol{u}_{\max}[i] \geq 0\}$.

However, these continuous relaxations face significant challenges when applied to complex graph structures. While a `MAXCUT` partition that cuts every edge exists for bipartite graphs, in fully connected graphs no more than half of the edges can be cut. As graph topologies depart from the bipartite case, algorithms relying on continuous relaxations tend to become unstable and perform poorly [225]. Figure

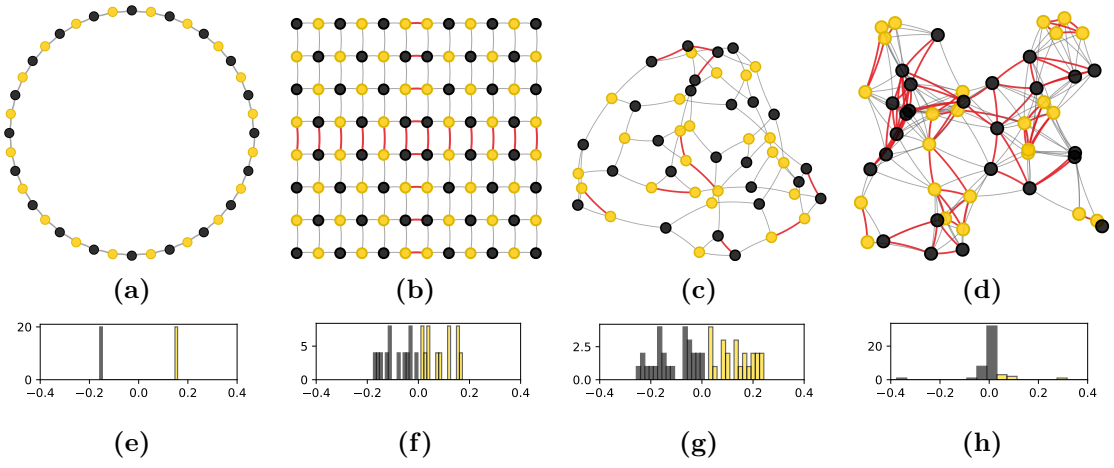

Figure 4.1: Top row: partitions induced by the sign of the elements in $\boldsymbol{u}_{\max}$. The nodes are colored based on the partition and the red edges are those not cut (the less, the better). **Bottom row:** histograms of $\boldsymbol{u}_{\max}$ inducing the partitions above. While in bipartite graphs the separation is sharp, the more a graph is irregular and dense the more the values are clustered around zero, making it difficult to find the optimal `MAXCUT`.

4.1 illustrates this challenge, showing the performance of the LEVS method on bipartite and non-bipartite graphs. As graphs become more dense and irregular, the values in $\boldsymbol{u}_{\max}$ cluster around zero, making it increasingly difficult to identify the optimal `MAXCUT` solution.

It's worth noting that the `MAXCUT` problem is closely related to graph coloring, particularly the 2-color approximate coloring problem [226]. This problem aims to identify subsets of nodes such that the connections within each subset are minimized. The resulting coloring represents a high-frequency graph signal and induces a partition that is orthogonal to spectral clustering [227].

4.2.2 Heterophilic message-passing

Message-passing (MP) is a fundamental operation in GNNs, allowing for the propagation of information across the graph structure (see Section 2.3.1). Let us consider a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ with $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ its node feature matrix and its adjacency matrix, respectively. A message-passing operator can be described in a compact way as:

$$\boldsymbol{X}' = \sigma(\boldsymbol{P}\boldsymbol{X}\boldsymbol{\Theta}) \tag{4.3}$$

where σ is a non-linear activation function, $\boldsymbol{\Theta}$ are the trainable parameters, and $\boldsymbol{P}$ is a propagation operator matching the sparsity pattern of $\boldsymbol{A}$. Different GNN architectures employ various propagation operators. For instance, in GCNs the propagation operator is defined as $\boldsymbol{P} = \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}$, where $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ and $\hat{d}_{ii} = \sum_{j=0} \hat{a}_{ij}$.

A key challenge with traditional message-passing operators is the tendency towards *oversmoothing*. With repeated application of the operator $\boldsymbol{P}$, which is usually non-negative, node features can become increasingly similar, compromising the ability to learn and represent diverse graph structures [228, 229]. This is particularly problematic when dealing with heterophilic graphs, where connected nodes often have different properties.

In contrast, by using a *sharpening* propagation operator, any kind of graph signal

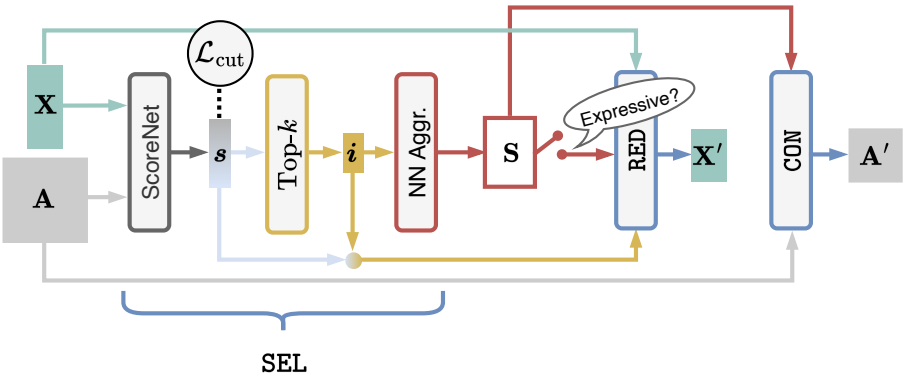

Figure 4.2: Schematic representation of MaxCutPool layer highlighting the `SELECT`, `REDUCE`, and `CONNECT` operations

can be learnt [230]. In this context, a sharpening operator is defined by an operator

$$\boldsymbol{P} = \boldsymbol{I} - \delta \left(\boldsymbol{I} - \boldsymbol{D}^{\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{\frac{1}{2}} \right) = \boldsymbol{I} - \delta \boldsymbol{L}^{\text{sym}} \tag{4.4}$$

where δ is a smoothness hyperparameter and $\boldsymbol{L}^{\text{sym}}$ is the symmetrically normalized Laplacian of $\mathcal{G}$. As observed by [231], when $\delta = 0$ the MP behaves like an MLP. Instead, when $\delta = 1$ the behavior is close to that of a GCN. Finally, as noted by [230], when $\delta > 1$ the operator $\boldsymbol{P}$ favors the realization of non-smooth signals on the graph. We refer to this variant as a Heterophilic Message Passing (HetMP) operator. We note that this can be seen as a graph-equivalent of the Laplacian sharpening kernels for images, mapping connected nodes to different values [232].

4.3 MaxCutPool overview

MaxCutPool can be described using the SRC framework (see Section 2.3.4), which provides a unified way to understand and compare different pooling methods. Figure 4.2 provides a schematic overview of the MaxCutPool layer, illustrating how these components work together to perform graph pooling.

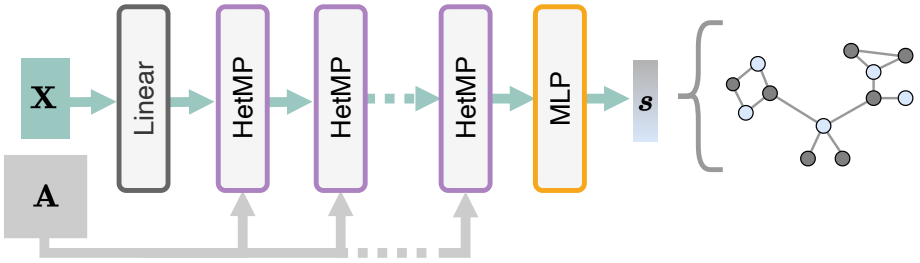

Figure 4.3: Schematic representation of the ScoreNet

4.3.1 `SELECT` operation

The `SEL` operation in MaxCutPool is the core of its novelty, leveraging a `MAXCUT`-inspired approach to choose supernodes. It consists of the following steps:

Node scoring

MaxCutPool can be ascribed to the family of scoring-based pooling methods (see Section 2.3.4). In this approach, a scoring vector assigning a score to each node is calculated, and the nodes with the highest scores are selected.

In our method, the scoring vector $\boldsymbol{s} \in [-1, 1]^N$ is computed via a ScoreNet, illustrated in Figure 4.3, that is an auxiliary GNN consisting of:

- a linear layer that maps the input features to a desired hidden dimension;
- a stack of HetMP layers that gradually transform the node features;
- an MLP that produces the final score vector.

The use of HetMP layers is crucial in overcoming the tension between standard message-passing (which tends to smooth features across adjacent nodes) and the `MAXCUT` objective (which aims to make adjacent nodes as different as possible). By setting $\delta > 1$ in the HetMP operation, we effectively create a high-pass filter on the graph, amplifying differences between adjacent nodes.

Supernode selection

The K nodes with the highest scores in $\boldsymbol{s}$ are selected as supernodes, forming one side of the MAXCUT partition $\mathcal{S}$. This is achieved through a top-K operation:

$$\boldsymbol{i} = \text{top}_K(\boldsymbol{s}) \tag{4.5}$$

where $\boldsymbol{i}$ are the indices of the selected supernodes.

Nearest neighbor aggregation

Once the K supernodes are selected, the remaining $N - K$ nodes are assigned to one of the supernodes via a nearest neighbor aggregation scheme. This process creates an assignment matrix $\boldsymbol{S}$, where $[\boldsymbol{S}]_{ij} = 1$ if and only if node i is assigned to supernode j. Formally:

$$\texttt{SEL} : [\boldsymbol{S}]_{ij} = 1 \iff j = \phi(\mathcal{S}, \boldsymbol{A}, i) \tag{4.6}$$

where $\phi(\mathcal{S}, \boldsymbol{A}, i)$ returns the nearest supernode of node i. Figure 4.4 illustrates

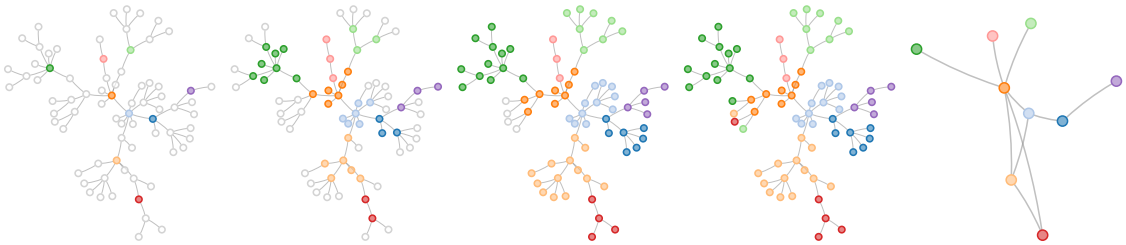

Figure 4.4: **(a)** The nodes with the $K = 9$ highest scores are selected. **(b-c)** Their ID is propagated to the unselected nodes until all are covered or until a maximum number of iterations (2 here) is reached. **(d)** The 4 remaining nodes are assigned randomly. **(e)** The pooled graph is obtained by aggregating the nodes with the same ID and coalescing the edges connecting nodes from different groups.

this process. The assignment is implemented through a breadth-first search of the graph, starting from the selected supernodes. This aggregation scheme aims at preserving local graph structure while achieving the desired graph coarsening.

The nearest neighbor association procedure is designed to be efficient and paral-

lelizable on GPUs and is described in Algorithm 2: the input consists of the graph $\mathcal{G}$ (in particular, its topology described by the adjacency matrix $\boldsymbol{A}$), the set of K supernodes $\mathcal{S}$ identified by the `SEL` operation, and a maximum number of iterations ($MaxIter$), which represent the maximum number of steps a node can traverse the graph to reach its closest supernode before being assigned at random.

Algorithm 2 Pseudo-code for the assignment scheme to the supernodes

```
1:  procedure AssignNodesToSupernodes(G, S, MaxIter)
2:      E ← InitializeEncodings(G, S)                          ▷ One-hot encoding
3:      m ← InitializeMask(G, S)
4:      Assignments ← InitializeEmptyList()
5:      for i = 1 to MaxIter do
6:          if AllNodesAssigned(m) then
7:              break
8:          end if
9:          E' ← ParallelMessagePassing(G, E)                  ▷ E' = AE
10:         Assignments ← ParallelAssignment(E', S, m)
11:         m ← UpdateMask(m, Assignments)
12:         E ← E'
13:     end for
14:     if not AllNodesAssigned(m) then
15:         RndAssignmentss ← ParallelRandomAssignment(UnassignedNodes, S)

16:     end if
17:     FinalAssignments ← GetFinalAssignments(Assignments, RndAssignments)

18:     return FinalAssignments
19: end procedure
```

In line 2, an encoding matrix $\boldsymbol{E}$ of size $N \times K + 1$ is initialized so that row i is a one-hot vector with the non-zero entry in position $k+1$, if the node i of the original graph is the k-th supernode. Otherwise, row i in a zero-vector of size $K + 1$. This matrix will be gradually populated when supernodes are encountered during the BFS. It's important to note that the 0-th column in matrix $\boldsymbol{E}$ (and subsequently in $\boldsymbol{E}'$) serves a special purpose. This column represents a "fake" supernode, which plays a crucial role in the assignment process.

A Boolean mask $\boldsymbol{m} \in \{0, 1\}^N$ indicating whether a node already encountered the closest supernode is initialized in line 3 with 1 in position i is nodes i is a supernode

and 0 otherwise. Finally, an empty list indicating to which supernode each node is assigned is initialized (line 4).

Until the maximum number of iterations is reached or until all nodes are assigned (line 6), the encoding matrix $\boldsymbol{E}$ is propagated with an efficient message-passing operation (line 9) that can be parallelized on a GPU. As soon as a 1 appears in position k within a line i of $\boldsymbol{E}$ previously full of zeros, node i is assigned to supernode k and the assignments and mask $\boldsymbol{m}$ are updated accordingly (lines 10 and 11). The *ParallelAssignment* function (line 10), in particular, takes the rows of the newly generated embeddings $\boldsymbol{E}'$ that have not yet been assigned and performs an `argmax` operation on the last dimension. If the `argmax` doesn't find any valid supernode for a node (*i.e.*, all values in the row are zero), it returns 0, effectively assigning the node to the "fake" supernode represented by the 0-th column. This allows to filter out the unassigned nodes in line 11.

If there are still unassigned nodes at the end of the iterations, the remaining nodes are randomly assigned to one of the K supernodes (line 15). Finally, all the assignments are merged (line 17).

4.3.2 `REDUCE` operation

MaxCutPool offers two variants for the `RED` operation:

1. MaxCutPool: $[\boldsymbol{X}']i := s_i \odot [\boldsymbol{X}]i :$
2. MaxCutPool-E: $\boldsymbol{X}' = \boldsymbol{s} \odot \boldsymbol{S}^T \boldsymbol{X}$

The first variant (standard MaxCutPool) only uses the features of the selected supernodes, while the second variant (MaxCutPool-E) combines information from all nodes assigned to each supernode. The "-E" suffix indicates that this variant satisfies the sufficient conditions for expressiveness as defined in [101]. The Hadamard product $\odot$ with the score vector $\boldsymbol{s}$ plays a crucial role in the learning process, as it ensures that gradients can flow back through the ScoreNet during backpropagation. This allows the model to learn an effective scoring function that is directly influenced by the downstream task.

4.3.3 `CONNECT` operation

For both variants, the `CON` operation, which determines the adjacency matrix of the pooled graph, is implemented as:

$$\texttt{CON} : \boldsymbol{A}' = \boldsymbol{S}^T \boldsymbol{A} \boldsymbol{S} \tag{4.7}$$

where $\boldsymbol{S}$ is the assignment matrix produced in the `SEL` operation (see Section 2.3.4).

4.3.4 Auxiliary loss

One of the central features of MaxCutPool is the presence of an auxiliary loss based on the `MAXCUT` objective

$$L_{\text{cut}} = \frac{\boldsymbol{s}^T \boldsymbol{A} \boldsymbol{s}}{W} \tag{4.8}$$

where $W = \sum_{ij} a_{ij}$ is the total edge weight of the graph. This loss encourages the selected nodes to belong to opposite sides of the `MAXCUT` partition. By minimizing L_{cut}, we push the model to assign nodes to different partitions if and only if they are connected, effectively maximizing the number of cut edges. The total loss for a GNN model incorporating MaxCutPool layers is then defined as:

$$L = L_{\text{task}} + \sum_{l} \beta L_{\text{cut}}^{(l)} \tag{4.9}$$

where L_{task} is the task-specific loss (e.g., classification loss), and β is a scalar weighting each auxiliary loss $L_{\text{cut}}^{(l)}$ associated with the l-th MaxCutPool layer. The auxiliary loss can be derived from the original `MAXCUT` objective as follows.

Let us consider the `MAXCUT` objective in Equation 4.1. It can be rewritten as

$$\max_{\boldsymbol{z}} \left(\sum_{i,j \in \mathcal{V}} a_{ij} - \sum_{i,j \in \mathcal{V}} z_i z_j a_{ij} \right) = \max_{\boldsymbol{z}} \left(W - \sum_{i,j \in \mathcal{V}} z_i z_j a_{ij} \right),$$

which is equivalent to

$$\max_{\boldsymbol{z}} \left(1 - \sum_{i,j \in \mathcal{V}} \frac{z_i z_j a_{ij}}{W} \right).$$

The solution $\boldsymbol{z}^*$ for the original objective is thus the solution for

$$\min_{\boldsymbol{z}} \frac{\boldsymbol{z}^\top A \boldsymbol{z}}{W},$$

leading to our auxiliary loss L_{cut}.

4.3.5 Hyperparameters and optimization

MaxCutPool and its ScoreNet component have several hyperparameters that can be tuned to optimize performance for specific tasks and datasets. We use the notation $[a, b, c]$ to indicate a model with three layers with hidden sizes a, b, and c, respectively. We also use the notation $[a] \times L$ to indicate L layers with a units each. The main hyperparameters are:

- the structure of ScoreNet HetMP block: the default configuration is $[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$;
- the activation function of the HetMP layers: By default, we use TanH;
- the structure of ScoreNet MLP block: the default configuration is $[16, 16]$;
- the activation function of the MLP layers: by default, we use ReLU;
- the smoothness hyperparameter δ: this controls the degree of heterophilic message-passing. The default value is 2;
- the auxiliary loss weight β: this balances the influence of the `MAXCUT` objective against the task-specific loss. The default value is 1.

In our experiments, we employ different grid search strategies to find the optimal hyperparameter configurations for each task and dataset. The specific hyperparameters optimized and their search ranges vary depending on the experiment and the GNN architecture used, and are detailed in the next sections.

4.4 Experimental evaluation

To demonstrate the effectiveness of MaxCutPool, we conducted extensive experiments on three main tasks: `MAXCUT` partition computation, graph classification, and node classification. These experiments were designed to evaluate MaxCutPool's performance on a variety of graph types, including heterophilic graphs.

4.4.1 Computation of the `MAXCUT` partition

Our first experiment focused on computing a `MAXCUT` partition by training a simple GNN consisting of an MP layer followed by the ScoreNet, which returns the score vector $\boldsymbol{s}$, as illustrated in Figure 4.5. We used a Graph Isomorphism Network (GIN) layer [233] as the MP layer with 32 units and ELU activation function. The model was trained by minimizing only the auxiliary loss L_{cut} defined in Equation 4.8, for 2000 epochs, using the Adam optimizer with an initial learning rate of 8e-4. We used a learning rate scheduler that reduces the learning rate by 0.8 when the auxiliary loss does not improve for 100 epochs. The `MAXCUT` partition is obtained by rounding the values in the score vector as follows;

$$y_i = \begin{cases} 1 & \text{if } s_i > 0, \\ -1 & \text{otherwise.} \end{cases}$$

The best configuration was found via a grid search on the following set of hyperparameters:

- smoothness hyperparameter δ: $\{2, 3, 5\}$;
- ScoreNet HetMP structure:
  - $[32] \times 4$,
  - $[4] \times 32$,
  - $[8] \times 16$,
  - $[16] \times 8$,
  - $[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$;
- HetMP layers activation: {ReLU, TanH}

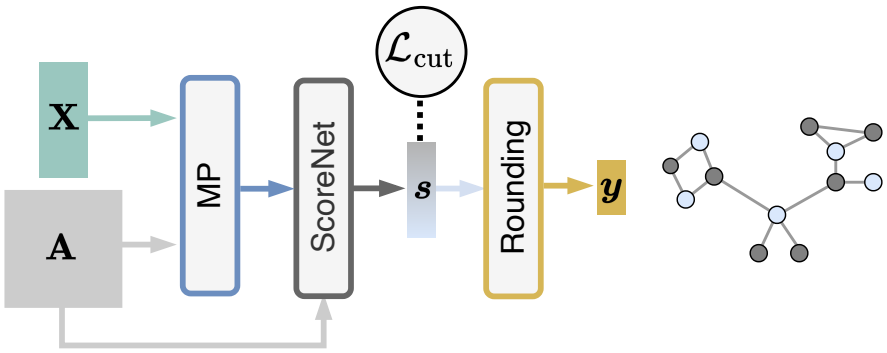

Figure 4.5: Schematic representation of the architecture used for `MAXCUT` evaluation.

In Table 4.1 we report the configurations of the ScoreNet used for the different graphs in the `MAXCUT` experiment. The performance of MaxCutPool was compared

Table 4.1: Hyperparameters configurations of the ScoreNet for the `MAXCUT` task.

Dataset	MP units	MP Act	δ
G14	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	ReLU	2.0
G15	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	ReLU	2.0
G22	$[4] \times 32$	TanH	2.0
G49	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	TanH	2.0
G50	$[8] \times 16$	ReLU	2.0
G55	$[4] \times 32$	ReLU	2.0
G70	$[8] \times 16$	ReLU	2.0
BarabasiAlbert	$[4] \times 32$	TanH	2.0
Community	$[4] \times 32$	TanH	2.0
ErdősRenyi	$[4] \times 32$	TanH	2.0
Grid2d (10×10)	$[4] \times 32$	TanH	2.0
Grid2d (60×40)	$[4] \times 32$	ReLU	2.0
Minnesota	$[4] \times 32$	TanH	2.0
RandRegular	$[4] \times 32$	TanH	2.0
Ring	$[4] \times 32$	ReLU	2.0
Sensor	$[4] \times 32$	TanH	2.0

against the LEVS approach based on $\boldsymbol{u}_{\max}$ and the GW algorithm, introduced in Section 4.2.1, and a GNN with GCN propagation operator that minimizes a `MAXCUT` loss, as proposed in [234]. To ensure a fair comparison, our MaxCutPool-based model and the GCN-based model were designed with a comparable number of learnable parameters.

We evaluated these methods on two sets of graphs:

- 9 graphs generated via the PyGSP library [235], including bipartite graphs such as Grid2D and Ring, as well as more complex structures;
- 7 graphs from the GSet dataset [236], including random, planar, and toroidal graphs, which are typically used as benchmarks for evaluating `MAXCUT` algorithms.

Table 4.2: Size of the graph cuts obtained with MaxCutPool, a GNN with GCN layers, and two common algorithms to compute the `MAXCUT`. GW results are absent for some entries of the PyGSP datasets and for GSet because the solver failed to converge.

(a) PyGSP datasets

Dataset	GW	LEVS	GCN	MaxCutPool
BarabasiAlbert	0.6875	0.6589	0.7240	**0.7292**
Community	0.6767	0.6429	0.6805	**0.6814**
ErdősRenyi	0.6920	0.6858	0.6797	**0.7105**
Grid (10×10)	**1.0000**	**1.0000**	0.9222	**1.0000**
Grid (60×40)	-	0.9787	0.1862	**0.9815**
Minnesota	-	0.9104	0.8904	**0.9130**
RandRegular	0.4827	0.8760	0.8733	**0.9040**
Ring	**1.0000**	**1.0000**	0.4200	**1.0000**
Sensor	0.6000	0.5719	0.6281	**0.6406**

(b) GSet datasets

Dataset	LEVS	GCN	MaxCutPool
G14	0.6155	0.6323	**0.6412**
G15	0.5945	0.6288	**0.6424**
G22	0.6441	0.6409	**0.6577**
G49	**1.0000**	0.9683	**1.0000**
G50	**0.9800**	0.9610	0.9750
G55	0.7568	0.7865	**0.8068**
G70	0.8803	0.8945	**0.9086**

Table 4.2 presents the results of this experiment. The performance is measured in terms of the ratio of cut edges, with higher values indicating better performance. As we can see, MaxCutPool consistently outperforms the baselines, finding the best cut in almost all cases. The use of heterophilic message-passing, in fact, allows the model to effectively capture and amplify differences between adjacent nodes, which is crucial for finding good `MAXCUT` partitions. These results demonstrate the effectiveness of our approach in solving the `MAXCUT` problem, even on complex and heterogeneous graph structures.

4.4.2 Multipartite dataset

As part of our contributions, we introduce a novel dataset for graph classification that specifically targets heterophilic graph structures. The Multipartite dataset is, to our knowledge, the first benchmark dataset of this kind.

The Multipartite dataset consists complete C-partite graphs. Each graph in the dataset is constructed such that its nodes can be partitioned into C groups, where nodes within each group are disconnected, but are connected to all nodes in every other group, as shown in Figure 4.6. This structure creates a highly heterophilic environment, challenging traditional GNN approaches that often rely on the assumption of homophily.

Algorithm 3 Multipartite graph dataset generation

Input: num_clusters, max_nodes_per_cluster, graphs_per_class
Output: dataset

```
cluster_centers ← GeneratePolygonVertices(num_clusters)  ▷ Initial arrangement of centers
dataset ← {}
for class_label ← 0 to num_clusters - 1 do
    for 1 to graphs_per_class do
        graph ← GenerateMultipartiteGraph(cluster_centers, max_nodes_per_cluster)
        graph.label ← class_label                ▷ Label based on current rotation
        Add graph to dataset
    end for
    cluster_centers ← RotateClockwise(cluster_centers)        ▷ Rotate for next class
end for
return dataset

function GenerateMultipartiteGraph(cluster_centers, max_nodes_per_cluster)
    for each center in cluster_centers do
        num_nodes ← RandomInt(1, max_nodes_per_cluster)
        node_positions ← GenerateNodesAroundCenter(center, num_nodes)
        node_color ← GetColorForCluster(center)  ▷ Each cluster has a unique color
        AddNodesToGraph(node_positions, node_color)
    end for
    ConnectNodesAcrossClusters()              ▷ Create complete multipartite graph
    return graph
end function

function RotateClockwise(centers)
    return [centers[-1]] + centers[:-1]                ▷ Move last center to front
end function
```

The generation procedure is described in Algorithm 3 and is detailed in the following:

1. A set of C cluster centers with 2D coordinates (x, y) is initially arranged in

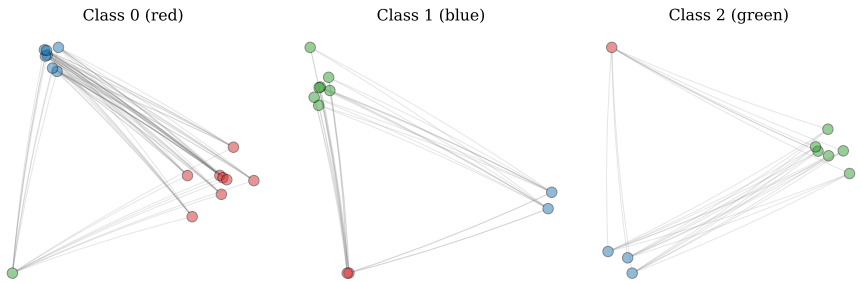

Figure 4.6: Examples of multipartite graphs with 3 cluster centers. The graph class corresponds to the color of the nodes from the group on the right.

a polygon shape. Each center is associated with a label, *i.e.*, a color.

2. The graph class is determined by the position and the color of the cluster centers. Specifically, the graph class is given by the color of the cluster whose center is on the positive x-axis.

3. For each class, we generate multiple graphs using these cluster centers. A graph is created by drawing at random the position of the nodes around each cluster center. The number of nodes per cluster varies randomly up to a maximum. Nodes within a cluster share the same color, which is determined by the cluster center.

4. The topology of each graph is obtained by connecting nodes from one cluster to the nodes of all the other clusters, but not to the nodes of the same cluster. Therefore, a node is connected only to nodes with different colors, making the graphs highly heterophilic.

5. After generating graphs for one class, the cluster centers are rotated, and this rotated configuration is used for the next class. Indeed, each rotation brings a different cluster to the positive x-axis.

6. The rotation process continues until the graphs for all the C different classes, whose number is equal to the number of clusters, are generated.

Most interestingly, the Multipartite dataset is designed in a way that the graph classification label does not depend on the graph topology and can, in principle, be inferred by a standard MLP. This makes the dataset particularly valuable for eval-

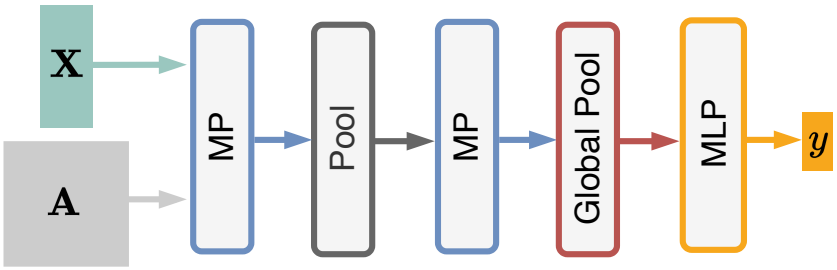

Figure 4.7: Schematic representation of the architecture used for graph classification.

uating GNNs because it challenges models to distinguish between relevant node features (color and position) and potentially misleading topological information, providing a controlled environment to assess how well GNNs can isolate and leverage relevant information in complex graph structures.

The specific instance of the Multipartite dataset used in our experiments, consisting of 5000 graphs with 10 centers, 500 graphs per center, and a maximum of 20 nodes per cluster, is available online at: https://zenodo.org/doi/10.5281/zenodo.11616515.

4.4.3 Graph classification

Building upon the promising results in `MAXCUT` partition computation, we next evaluated MaxCutPool's performance on the more practical task of graph classification. This experiment aims to assess how well the `MAXCUT`-inspired pooling operation translates to improved performance on downstream tasks.

For this experiment, we used a GNN classifier with the following architecture: `MP`(32)-`POOL`-`MP`(32)-`READOUT` (Figure 4.7). Here, `MP` represents GIN layer, chosen for its expressive power in capturing graph structures. The `POOL` operation is where we apply MaxCutPool or one of the competing pooling methods for comparison.

In our experiments, we evaluated three variants of MaxCutPool:

- MaxCutPool and MaxCutPool-E, described in Section 4.3.2;
- MaxCutPool-NL: a version without the `MAXCUT` auxiliary loss, serving as an

ablation study to assess its importance.

We tested our model against a range of state-of-the-art pooling methods:

- DiffPool [96], DMoN [100], MinCutPool [97] from the soft-clustering pooling family (see Section 2.3.4);
- Top-k [90] from the scoring-based family (see Section 2.3.4);
- Graclus [112] and k-MIS [111] from the one-every-k family (see Section 2.3.4);
- Edge-Contraction Pooling (ECPool) [237]: A pooling method that iteratively contracts edges in the graph.

Table 4.3: Details of the graph classification datasets.

Dataset	#Samples	#Classes	Avg. #vert.	Avg. #edg.	V. attr.	V. lab.	$\bar{h}(\mathcal{D})$
EXPWL1	3,000	2	76.96	186.46	–	yes	0.2740
NCI1	4,110	2	29.87	64.60	–	yes	0.6245
PROTEINS	1,113	2	39.06	72.82	1	yes	0.6582
Mutagenicity	4,337	2	30.32	61.54	–	yes	0.3679
COLLAB	5,000	3	74.49	4,914.43	–	no	1
REDDIT-B	2,000	2	429.63	995.51	–	no	1
GCB-H	1,800	3	148.32	572.32	–	yes	0.8440
DD	1,178	2	284.32	1,431.32	–	yes	0.0688
MUTAG	188	2	17.93	19.79	–	yes	0.7082
ENZYMES	600	6	32.63	62.14	18	yes	0.6687
Multipartite	5000	10	99.79	4,477.43	3	yes	0.1101

We tested these methods on a diverse set of graph classification datasets:

- eight datasets from the TU Dataset collection [238]: COLLAB, DD, NCI1, ENZYMES, MUTAG, Mutagenicity, PROTEINS, and REDDIT-BINARY;
- the Graph Classification Benchmark-Hard (GCB-H) [239], designed to be particularly challenging for GNNs;
- EXPWL1 [101], a dataset specifically created to test the expressive power of GNNs;
- the Multipartite dataset introduced in Section 4.4.2.

The statistics of the graph classification datasets are summarized in Table 4.3. Typical homophily metrics are designed for node classification tasks or rely on node labels, which are not always available in graph classification settings. To quantify

the degree of homophily in our datasets, we introduce a surrogate homophily score $\bar{h}(\mathcal{D})$ based on node features, where $\mathcal{D}$ denotes the whole dataset. This score is defined as the absolute value of the average cosine similarity between the node features of connected nodes in each graph of the dataset:

$$\bar{h}(\mathcal{D}) = \left| \frac{1}{|\mathcal{D}|} \sum_{\mathcal{G} \in \mathcal{D}} \frac{1}{|\mathcal{E}_{\mathcal{G}}|} \sum_{(i,j) \in \mathcal{E}_{\mathcal{G}}} \frac{\boldsymbol{x}_i \boldsymbol{x}_j}{|\boldsymbol{x}_i||\boldsymbol{x}_j|} \right|$$

where $|\mathcal{D}|$ is the number of graphs in the dataset, $\mathcal{E}_{\mathcal{G}}$ is the set of edges of the graph $\mathcal{G}$ and $\boldsymbol{x}_i, \boldsymbol{x}_j$ are the feature vectors of the i-th and j-th node respectively. This measure allows us to assess the degree of feature similarity between connected nodes across the dataset, providing insight into the homophilic nature of the graphs.

For our experiments, whenever node features were not available, we used node labels. If node labels were also unavailable, we used a constant as a surrogate node feature. The datasets were split via a 10-fold cross-validation procedure. The training dataset was further partitioned into a 90-10% train-validation random split. This approach is similar to the procedure described by [240]. The models were trained using a batch size of 32 for 1000 epochs, using the Adam optimizer with an initial learning rate of 1e-4. We used early stopping with a patience of 300 epochs, monitoring the validation loss. The best configuration was found via a grid search on the following set of hyperparameters:

- auxiliary loss weight β: $\{1, 2, 5\}$;
- ScoreNet HetMP structure:
  - $[32] \times 8$,
  - $[32] \times 4$,
  - $[8] \times 16$
  - $[16] \times 8$,
  - $[32, 32, 16, 16, 8, 8]$,
  - $[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$.

In Table 4.4 we report the configurations of the ScoreNet used in the graph classification architecture for the different datasets in the expressive and non-expressive variant of MaxCutPool.

Table 4.4: Hyperparameters configurations of the ScoreNet for the graph classification task.

	MaxCutPool		MaxCutPool-E	
Dataset	**MP units**	β	**MP units**	β
GCB-H	$[8] \times 16$	3.0	$[32] \times 8$	5.0
COLLAB	$[32] \times 8$	1.0	$[32] \times 8$	1.0
DD	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	1.0	$[8] \times 16$	5.0
ENZYMES	$[8] \times 16$	3.0	$[16] \times 8$	3.0
EXPWL1	$[32, 32, 16, 16, 8, 8]$	1.0	$[16] \times 8$	1.0
MUTAG	$[8] \times 16$	1.0	$[16] \times 8$	3.0
Multipartite	$[32] \times 8$	3.0	$[32] \times 8$	1.0
Mutagenicity	$[32, 32, 16, 16, 8, 8]$	1.0	$[32] \times 8$	5.0
NCI1	$[32, 32, 16, 16, 8, 8]$	1.0	$[8] \times 16$	3.0
PROTEINS	$[32, 32, 32, 32, 16, 16, 16, 8, 8, 8, 8]$	3.0	$[32, 32, 16, 16, 8, 8]$	5.0
REDDIT-B	$[32] \times 8$	1.0	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	1.0

Table 4.5: Mean and standard deviations of the graph classification accuracy. For each dataset the best performing method and those that are not significantly different from it are colored in green. If a method is in the top-performing group is assigned with a score of 1, 0 otherwise.

Pooler	**GCB-H**	**COLLAB**	**EXPWL1**	**Mult.**	**Mutag.**	**NCI1**	**REDDIT-B**	**Score**
No pool	*74*$_{\pm 4}$	*74*$_{\pm 2}$	*87*$_{\pm 2}$	*14*$_{\pm 12}$	*79*$_{\pm 2}$	*78*$_{\pm 3}$	*90*$_{\pm 2}$	-
DiffPool	51$_{\pm 8}$	70$_{\pm 2}$	69$_{\pm 3}$	9$_{\pm 1}$	78$_{\pm 2}$	75$_{\pm 2}$	90$_{\pm 2}$	1
DMoN	74$_{\pm 3}$	68$_{\pm 2}$	73$_{\pm 3}$	52$_{\pm 2}$	80$_{\pm 2}$	77$_{\pm 2}$	88$_{\pm 2}$	3
EdgePool	75$_{\pm 4}$	72$_{\pm 3}$	90$_{\pm 2}$	55$_{\pm 3}$	80$_{\pm 2}$	77$_{\pm 3}$	91$_{\pm 2}$	4
Graclus	75$_{\pm 3}$	72$_{\pm 3}$	90$_{\pm 2}$	25$_{\pm 18}$	80$_{\pm 2}$	77$_{\pm 2}$	90$_{\pm 3}$	4
k-MIS	75$_{\pm 4}$	71$_{\pm 2}$	99$_{\pm 1}$	58$_{\pm 2}$	79$_{\pm 2}$	75$_{\pm 3}$	90$_{\pm 2}$	4
MinCutPool	75$_{\pm 5}$	70$_{\pm 2}$	71$_{\pm 3}$	56$_{\pm 3}$	78$_{\pm 3}$	73$_{\pm 3}$	87$_{\pm 2}$	1
Top-k	56$_{\pm 5}$	72$_{\pm 2}$	73$_{\pm 2}$	43$_{\pm 3}$	75$_{\pm 3}$	73$_{\pm 2}$	77$_{\pm 2}$	0
MaxCutPool	73$_{\pm 3}$	77$_{\pm 2}$	100$_{\pm 0}$	90$_{\pm 2}$	77$_{\pm 2}$	75$_{\pm 2}$	89$_{\pm 3}$	5
MaxCutPool-E	74$_{\pm 3}$	77$_{\pm 2}$	100$_{\pm 0}$	87$_{\pm 5}$	79$_{\pm 1}$	76$_{\pm 2}$	89$_{\pm 2}$	7
MaxCutPool-NL	61$_{\pm 6}$	77$_{\pm 3}$	100$_{\pm 0}$	91$_{\pm 1}$	76$_{\pm 3}$	74$_{\pm 2}$	86$_{\pm 3}$	3

Table 4.5 presents the results of our graph classification experiments. For each dataset, we report the mean and standard deviation of the classification accuracy across 10-fold cross-validation. We also include the performance of the GNN model without any pooling layers as a baseline (*No pool*). We conducted a preliminary ANOVA test (p-value 0.05) for each dataset followed by a pairwise Tukey-HSD test (p-value 0.05) to group models whose performance is not significantly different. Those belonging to the top-performing group are colored in green. The ANOVA test failed on ENZYMES, PROTEINS, MUTAG, and DD, meaning that the difference in the performance of the GNNs equipped with different poolers is

not significant. For this reason, the results on these datasets are omitted from Table 4.5 and reported in Table 4.6.

Table 4.6: Graph classification accuracy values (subset)

Pooler	DD	MUTAG	ENZYMES	PROTEINS
No pool	$73_{\pm 5}$	$78_{\pm 13}$	$33_{\pm 6}$	$71_{\pm 4}$
Diffpool	$77_{\pm 4}$	$81_{\pm 11}$	$36_{\pm 7}$	$75_{\pm 3}$
DMoN	$78_{\pm 5}$	$82_{\pm 11}$	$37_{\pm 7}$	$76_{\pm 4}$
ECPool	$73_{\pm 5}$	$84_{\pm 12}$	$35_{\pm 8}$	$74_{\pm 5}$
Graclus	$73_{\pm 4}$	$82_{\pm 12}$	$33_{\pm 7}$	$73_{\pm 4}$
k-MIS	$75_{\pm 3}$	$83_{\pm 10}$	$33_{\pm 8}$	$73_{\pm 5}$
MinCutPool	$78_{\pm 5}$	$81_{\pm 12}$	$34_{\pm 9}$	$77_{\pm 5}$
Top-k	$72_{\pm 5}$	$82_{\pm 10}$	$29_{\pm 7}$	$74_{\pm 5}$
MaxCutPool	$77_{\pm 4}$	$84_{\pm 10}$	$31_{\pm 6}$	$74_{\pm 4}$
MaxCutPool-E	$77_{\pm 3}$	$85_{\pm 9}$	$34_{\pm 5}$	$74_{\pm 4}$
MaxCutPool-NL	$74_{\pm 4}$	$83_{\pm 11}$	$31_{\pm 4}$	$70_{\pm 4}$

The results reveal several key insights: first of all, MaxCutPool consistently ranks among the top-performing methods across all evaluated datasets. This demonstrates its versatility and effectiveness across a wide range of graph types and classification tasks. Interestingly, on the EXPWL1 dataset, designed to test GNN expressiveness, even the non-expressive variant of MaxCutPool achieves perfect accuracy (100%), outperforming all competitors. This is a significant result, as it represents the first known instance of a non-expressive pooler passing this challenging expressiveness test. As expected, MaxCutPool shows particularly strong performance on the Multipartite dataset, significantly outperforming all other pooling methods. This highlights MaxCutPool's effectiveness in handling highly heterophilic graph structures, where traditional pooling methods often struggle.

When compared to the *No pool* baseline, MaxCutPool improves classification performance on most datasets. This suggests that MaxCutPool is effectively increasing the receptive field of the message-passing layers while retaining necessary information and enhancing the overall expressive power of the GNN model. The MaxCutPool-E variant, which satisfies theoretical expressiveness conditions, generally exhibits similar or better performance compared to the standard MaxCutPool across most datasets. This indicates that the added expressiveness can indeed translate to improved practical performance. The performance decline observed in

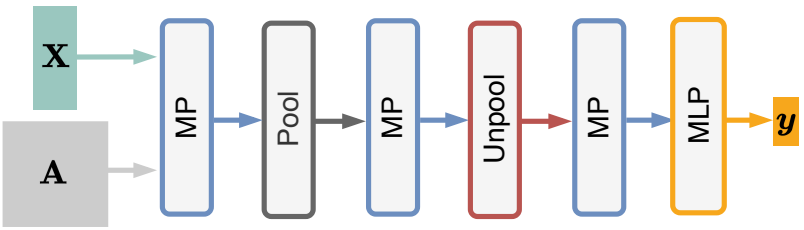

Figure 4.8: Schematic representation of the architecture used for node classification.

the MaxCutPool-NL variant (without the auxiliary loss), on the other hand, demonstrates the importance of the `MAXCUT`-based loss in guiding the pooling process. This ablation result validates our design choice of incorporating graph-theoretical principles into the learning process. On datasets like COLLAB, however, all MaxCutPool variants achieve top performance, showing a statistically significant improvement over other methods. Interestingly, when learning this dataset, the auxiliary loss term plateaued around 0, making the performance equivalent to the MaxCutPool-NL variant. This suggests that our method remains robust even when the auxiliary loss is not needed for the downstream task.

4.4.4 Node classification

To further evaluate the effectiveness of MaxCutPool, particularly on heterophilic graphs, we conducted experiments on node classification tasks.

For this task, we employed a simple auto-encoder architecture for node classification with the following structure: `MP`(32)-`POOL`-`MP`(32)-`UNPOOL`-`MP`(32)-`READOUT` (Figure 4.8). Here, `MP` is a GIN layer, `POOL` is implemented by MaxCutPool or one of the baseline methods, and `UNPOOL` (also referred to as *lifting*) is implemented by copying the value of each supernode to all the nodes that were assigned to it during the pooling phase.

We compared MaxCutPool against Top-k [90], k-MIS [111] and NDP [115] methods. It's worth noting that we did not include soft-clustering poolers (like DiffPool or MinCutPool) or Graclus in this comparison due to their high VRAM and RAM memory requirements for large graphs, respectively. In contrast, MaxCut-

Pool demonstrated excellent scalability, efficiently handling large graph structures with minimal memory overhead (see Section 4.4.6).

For this experiment, we used five heterophilic datasets introduced in [241], whose statistics are summarized in Table 4.7.

Table 4.7: Statistics of node classification datasets.

Dataset	# Nodes	# Edges	# Classes	$h(\mathcal{G})$
Roman-Empire	22,662	32,927	18	0.021
Amazon-Ratings	24,492	93,050	5	0.127
Minesweeper	10,000	39,402	2	0.009
Tolokers	11,758	519,000	2	0.180
Questions	48,921	153,540	2	0.079

These datasets represent a range of heterophilic graph structures, with varying levels of homophily as measured by the class insensitive edge homophily ratio $h(\mathcal{G})$ [242]. The Roman-Empire dataset, in particular, has the lowest homophily ratio among all datasets, making it an excellent test case for methods designed to handle heterophilic graphs.

The node classifier was trained for 20000 epochs, using the Adam optimizer with an initial learning rate of 5e-4. We used a learning rate scheduler that reduces the learning rate by 0.5 when the validation loss does not improve for 500 epochs. We used early stopping with a patience of 2000 epochs, monitoring the validation loss. The best configuration was found via a grid search on the following set of hyperparameters:

- ScoreNet HetMP structure:
  - $[32] \times 4$,
  - $[4] \times 32$,
  - $[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$;
- MLP layers activation: {ReLU, TanH}

The configuration of the ScoreNet for the MaxCutPool pooler used in the different datasets is reported in Table 4.8.

Table 4.9 presents the results of our node classification experiments. For Roman-

Table 4.8: Hyperparameters configurations of the ScoreNet in the node classification task.

Dataset	MP units	MLP Act.
Roman-Empire	$[32, 32, 32, 32]$	ReLU
Amazon-Ratings	$[32, 32, 32, 32]$	ReLU
Minesweeper	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	ReLU
Tolokers	$[32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8]$	ReLU
Questions	$[32, 32, 32, 32]$	ReLU

Table 4.9: Node classification accuracy (Roman-empire, Amazon-ratings) and AUROC (Minesweeper, Tolokers, Questions). The best performing models in each dataset are in green and get 1 score point, 0 otherwise.

Pooler	**Roman-e.**	**Amazon-r.**	**Minesw.**	**Tolokers**	**Questions**	**Score**
Top-k	$26_{\pm 7}$	$46_{\pm 4}$	$94_{\pm 1}$	$89_{\pm 5}$	$64_{\pm 3}$	1
k-MIS	$23_{\pm 3}$	$48_{\pm 2}$	$75_{\pm 2}$	$84_{\pm 2}$	$83_{\pm 1}$	1
NDP	$22_{\pm 5}$	$53_{\pm 2}$	$98_{\pm 0}$	$88_{\pm 6}$	$68_{\pm 4}$	3
MaxCutPool	$56_{\pm 3}$	$53_{\pm 1}$	$96_{\pm 1}$	$87_{\pm 3}$	$82_{\pm 4}$	4
MaxCutPool-E	$60_{\pm 4}$	$53_{\pm 2}$	$97_{\pm 1}$	$91_{\pm 2}$	$85_{\pm 5}$	**5**

empire and Amazon-ratings, we report the mean and standard deviation of the classification accuracy. For Minesweeper, Tolokers, and Questions, we report the ROC AUC, following the same evaluation protocol used in [241]. MaxCutPool, particularly in its expressive variant (MaxCutPool-E), achieves superior performance across these heterophilic datasets. This is especially evident on the Roman-Empire dataset, where both MaxCutPool and MaxCutPool-E significantly outperform all other methods. Nonetheless, MaxCutPool-E consistently ranks in the top tier across all datasets, showing robust performance across varying degrees of heterophily. This consistency is noteworthy, as other methods tend to excel only on a subset of the datasets.

4.4.5 Node classification with skip connections

In addition to the basic node classification architecture, we also evaluated a different model incorporating skip (residual) connections. This architecture, inspired by the Graph U-Net [90], aims to preserve low-level features throughout the network, potentially improving performance on heterophilic graphs.

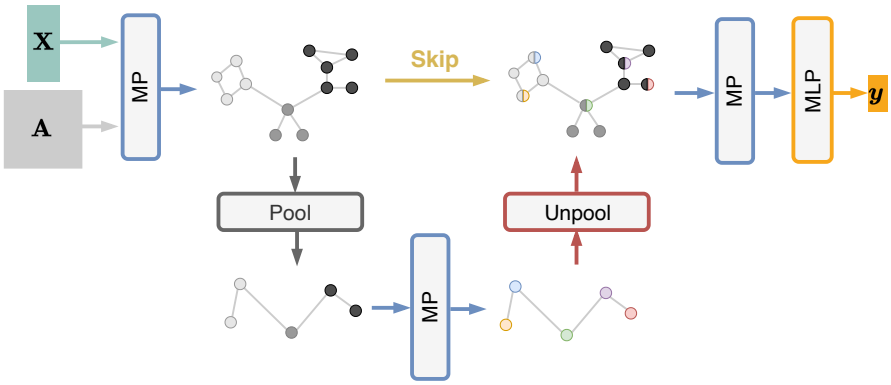

Figure 4.9: Schematic representation of the architecture with skip connections used for node classification.

The architecture with skip connections is depicted in Figure 4.9. In this architecture, the node features obtained after the first MP layer are concatenated with the node features yielded by the unpooling step. This allows the network to combine high-level abstract features with low-level structural information. We used the same datasets and evaluation protocol as in the previous node classification experiment. The hyperparameters for the MaxCutPool layer in this architecture are reported in Table 4.10. For the Minesweeper dataset, we used a GIN layer

Table 4.10: Hyperparameters configurations for the node classification task based on the architecture with skip connections.

Dataset	MP units	MLP Act.	Expressive
Roman-Empire	[32, 32, 32, 32]	ReLU	✗
Amazon-Ratings	[32, 32, 32, 32]	ReLU	✗
Minesweeper	[32, 32, 32, 32]	Tanh	✗
Tolokers	[32, 32, 32, 32]	ReLU	✗
Questions	[32, 32, 32, 32]	ReLU	✗

with 16 units as the MP layer, instead of the usual 32 units. This adjustment was necessary because the architecture with skip connections consistently achieved nearly 100% ROC AUC with 32 units, regardless of the pooling method applied.

The results for node classification using the architecture with skip connections (Table 4.11) keep showing the effectiveness of our layer in handling heterophilic graphs. MaxCutPool maintains strong performance across datasets, with particu-

Table 4.11: Node classification accuracy (Roman-empire, Amazon-ratings) and AUROC (Minesweeper, Tolokers, Questions) obtained when using the architecture with skip connections.

Pooler	Roman-e.	Amazon-r.	Minesw.*	Tolokers	Questions	Score
Top-k	$20_{\pm 11}$	$49_{\pm 7}$	$91_{\pm 1}$	$96_{\pm 0}$	$70_{\pm 3}$	1
k-MIS	$19_{\pm 2}$	$53_{\pm 3}$	$90_{\pm 0}$	$91_{\pm 2}$	$82_{\pm 4}$	2
NDP	$19_{\pm 4}$	$56_{\pm 5}$	$94_{\pm 0}$	$90_{\pm 8}$	$69_{\pm 7}$	2
MaxCutPool	$67_{\pm 2}$	$53_{\pm 1}$	$92_{\pm 1}$	$96_{\pm 1}$	$82_{\pm 2}$	**3**

larly significant improvements on the highly heterophilic Roman-Empire dataset. However, the performance gap between MaxCutPool and other methods narrows compared to the basic architecture, suggesting that skip connections benefit other pooling methods in handling heterophilic graphs to some extent. These findings highlight the importance of considering architectural choices when designing GNNs for heterophilic graphs.

4.4.6 Memory usage and scalability

An important aspect of any pooling method is its memory usage and scalability to large graphs. We conducted an experimental evaluation of the GPU VRAM usage for different pooling methods, including MaxCutPool. The results, shown in Figure 4.10, demonstrate that MaxCutPool scales efficiently with graph size. The plot reveals that soft-clustering methods exhibit exponential growth in GPU VRAM usage as graph size increases. In contrast, scoring-based methods, including MaxCutPool, show sublinear growth. This makes MaxCutPool particularly suitable for working with large graphs, a common scenario in molecular and biological datasets.

4.5 Conclusions

MaxCutPool represents a significant advancement in graph pooling techniques. By leveraging the `MAXCUT` loss and incorporating heterophilic message-passing, unlike existing graph pooling and coarsening approaches that aim to preserve low-frequencies, it performs exceptionally well on heterophilic datasets. Our proposed pooling strategy combines advantages of different approaches while addressing their

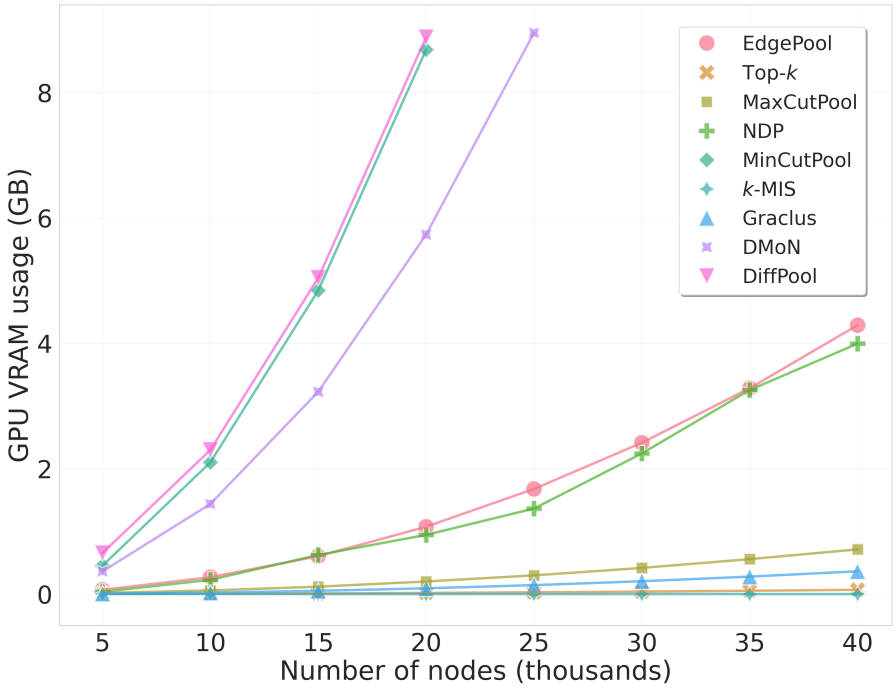

Figure 4.10: The GPU VRAM usage of different poolers.

limitations. Like score-based methods, it allows for flexible pooling ratios adaptable to graph size. Its HetMP layers enable uniform supernode distribution similar to one-every-K methods, while maintaining the flexibility to choose any set of supernodes. Uniquely among scoring-based poolers, MaxCutPool incorporates a graph-theoretical auxiliary regularization loss. Its strong performance across tasks from `MAXCUT` partition computation to graph and node classification demonstrates its versatility and effectiveness. This makes it a promising tool for molecular graph analysis, with potential to improve property prediction and drug discovery tasks.

Chapter 5

Other Works

This chapter describes two complementary projects related to computational drug discovery: a comprehensive review of machine learning approaches for ligandability and druggability prediction, and the development of a web server interface for molecular surface computation and pocket detection.

5.1 Ligandability and druggability assessment via machine learning

A critical step in early-stage drug discovery is the identification and characterization of ligand binding sites on target proteins. The concepts of ligandability and druggability aim to quantify the likelihood that a given binding site or target protein can bind small molecules with high affinity and be modulated by a drug-like compound, respectively. In recent years, machine learning approaches have emerged as powerful tools for predicting ligandability and druggability.

We conducted a comprehensive review of machine learning methods for ligandability and druggability assessment [P4]. This review provides a thorough analysis of the current state-of-the-art in ML-based ligandability and druggability prediction, serving as a valuable resource for researchers in the field of computer-aided drug discovery.

5.1.1 Machine learning tasks and architectures

Our review categorizes ML approaches for ligandability and druggability prediction into two main groups:

- **two-step approaches**: these methods first detect pockets using geometric or energetic algorithms, then apply ML models to score the identified pockets.
- **direct approaches**: these methods use ML to simultaneously detect and score potential binding sites.

For each category, we discuss representative algorithms, their strengths, and limitations.

Two-step methods like DoGSiteScorer [243] and PockDrug [244] showed advantages in interpretability since they separate the geometric detection and scoring steps. The scoring step in this methods is typically implemented via classical shallow learners such as support vector machines and random forests. For this reason, they offer advantages in terms of interpretability.

Direct approaches like DeepSite [245], DeepSurf [246] and PointSite [247] show promise in their ability to learn complex features directly from structural data. These methods typically make use deep learning models such as 3D convolutional neural networks and graph neural networks, requiring significant computational resources and large training datasets.

5.1.2 Feature engineering and representation

A significant portion of the review is dedicated to examining different ways of representing protein structures and pockets for ML models. We discuss several key representation approaches:

- **voxel-based representations**: these methods discretize the 3D space around a protein into a grid, encoding physicochemical properties in each voxel. This approach is particularly suited for 3D convolutional neural networks;
- **graph-based representations**: these methods represent proteins as graphs,

with atoms or residues as nodes and chemical bonds or spatial proximity as edges. Graph neural networks can effectively process these representations;

- **point cloud representations**: these methods represent proteins as sets of points in 3D space, often used with specialized architectures like PointNet [248];
- **surface-based representations**: these methods focus on the molecular surface, using techniques like ray-casting to capture surface properties.

Our analysis reveals distinct trade-offs between these representation approaches. While voxel-based methods provide a natural way to apply powerful deep learning architectures, they can be computationally intensive for large proteins. Graph-based approaches offer a more compact representation and can naturally capture chemical connectivity, but may require more sophisticated architectures to process spatial information effectively. Point cloud representations provide an interesting middle ground, offering both spatial accuracy and computational efficiency. Surface-based methods are particularly effective for analyzing protein-protein interfaces and shallow binding sites, though they may miss information about deep pockets.

5.1.3 Incorporation of molecular dynamics

An important aspect covered in our review is the integration of MD simulations into ligandability prediction methods. MD-based methods offer significant advantages in identifying transient and cryptic binding sites that may not be visible in static structures. Tools like TRAPP [249] effectively leverage MD trajectories to estimate average pocket distributions and detect transient pockets, while methods like CryptoSite [250] achieve comparable results to full MD simulations with reduced computational cost. The JEDI [251] method stands out by providing an analytically derivable ligandability potential that can be directly used in MD through enhanced sampling methods, offering a unique approach to pocket optimization.

5.1.4 Current challenges and future directions

Our review identifies several key areas for improvement in the field of ML-based ligandability and druggability prediction. One of the most pressing needs is the development of more rigorous labeling strategies. Current methods often rely on simple binary classifications, which fail to capture the nuanced nature of ligand binding. We argue for the adoption of labeling approaches based on thermodynamic or kinetic observables, which would provide a more accurate representation of a pocket's ligandability. Another significant challenge lies in handling protein flexibility. While some methods have begun to incorporate molecular dynamics simulations, there is still considerable room for improvement in efficiently capturing and representing protein dynamics. This is crucial for accurately predicting ligandability, as proteins are not static entities but constantly fluctuate between different conformations. Finally, the growing interest in RNA as a drug target necessitates the extension of ligandability prediction methods to RNA structures. This presents unique challenges due to the distinct structural and chemical properties of RNA compared to proteins, requiring the development of new algorithms and feature representations tailored to nucleic acid structures.

Looking ahead, we see great potential in the integration of ligandability prediction with other drug discovery tasks. End-to-end learning systems that combine ligandability prediction with de novo drug design, ADMET prediction, and other related tasks could significantly streamline the drug discovery process. Such integrated approaches could leverage the interrelationships between these different aspects of drug discovery to improve overall predictive power and efficiency.

5.2 Development of a web server for molecular surface analysis

NanoShaper [252] is a powerful tool for computing and analyzing molecular surfaces, offering various surface definitions and functionalities relevant to computational drug discovery. It provides a comprehensive suite of tools for molecular sur-

face analysis, including the computation of Solvent Excluded Surface (SES), Skin surface, and Gaussian surface. Beyond surface computation, NanoShaper offers sophisticated cavity detection algorithms, allowing researchers to identify potential binding sites within protein structures.

One of NanoShaper's key strengths is its ability to generate grid-consistent triangulations of molecular surfaces. This feature is particularly useful for applications in computational physics, such as solving the Poisson-Boltzmann equation for electrostatics calculations. NanoShaper can also color grid points based on their location relative to the molecular surface, facilitating the setup of volumetric calculations.

As an ongoing project, we are developing a user-friendly web interface for NanoShaper. This web server aims to make these powerful capabilities more accessible to researchers without extensive computational expertise. The interface allows users to upload protein structures in PDB format and easily specify the type of surface they want to compute and analyze. Users can select from various surface types, adjust parameters like probe radius for SES or blobbyness for Gaussian surfaces, and specify additional analysis options such as cavity detection thresholds.

Once the calculations are complete, the web server provides interactive visualization of the results. Users can explore the computed molecular surface, view detected cavities, and examine surface properties. The interface will also allow for easy download of the computed surfaces and associated data, enabling further analysis or integration with other computational workflows. In Figure 5.1 we show a snapshot of the preliminary home page of the web server.

The availability of this web server will facilitate the usage of NanoShaper, enabling a broader range of researchers to benefit from its capabilities in analyzing protein surfaces and potential binding sites. It will be particularly useful for structural biologists and medicinal chemists as this approach won't require to run NanoShaper locally.

We envision several future potential enhancements for the NanoShaper web server. One promising direction is extending support to RNA structures, addressing the growing interest in RNA as a drug target. This would involve adapting the un-

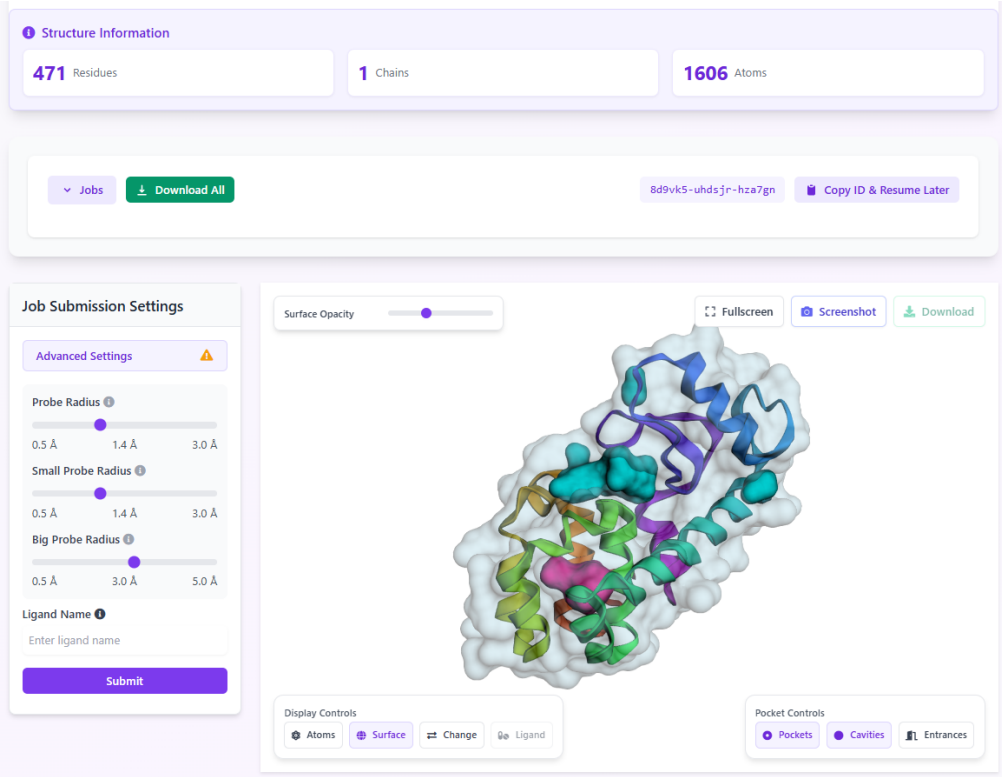

Figure 5.1: A snapshot of the NanoShaper web server interface.

derlying algorithms to handle the unique structural features of RNA molecules. Additionally, we could incorporate RNA-specific scoring functions for pocket analysis, taking into account factors like the presence of potential metal ion binding sites or the accessibility of the major and minor grooves.

Another potential area for improvement is the integration of the web server with other computational tools in the drug discovery pipeline. For example, we could provide direct links to docking software or electrostatics calculations, allowing users to seamlessly move from surface analysis to more detailed binding site characterization or virtual screening.

In conclusion, the development of the NanoShaper web server represents an important step in making advanced molecular surface analysis tools more accessible to the broader scientific community. By providing an intuitive interface to NanoShaper capabilities, we hope to accelerate research in structure-based drug design and related fields.

Chapter 6

Conclusions and future perspectives

This thesis has explored graph-based machine learning approaches for drug discovery, focusing on de novo molecular design and advanced graph representation techniques. Our work spans theoretical developments, practical applications of graph neural networks, and the creation of tools to support researchers in the drug discovery pipeline. We began with a comprehensive review of conditional generative models for de novo drug discovery, synthesizing recent advances and identifying key challenges in the field. This review provided the foundation for our subsequent research. Subsequently, we developed the Atomic-Molecular Conditional Generator (AMCG), a novel graph-based generative model for molecular design. AMCG offers several key advantages, including a dual atomic-molecular representation, one-shot generation capabilities, explicit control over atom type histograms, and property optimization through gradient ascent in the latent space. AMCG demonstrated state-of-the-art performance on standard benchmarks, showcasing its potential to generate valid, diverse, and property-optimized molecules efficiently. Our work on AMCG led us to recognize the inherently heterophilic nature of molecular graphs, where connected nodes (atoms) often have dissimilar features. This realization prompted us to explore more fundamental aspects of graph representation learning, resulting in the development of MaxCutPool. This novel graph pooling technique,

based on the MAXCUT problem, proved particularly effective for heterophilic graphs. MaxCutPool's superior performance on heterophilic graph classification and node classification tasks demonstrated how advances in graph theory could translate into improved performance on real-world tasks in drug discovery.

A natural consequence of the work conducted in this thesis is to integrate these components into a more comprehensive de novo drug design framework. A straightforward extension of our work would be to incorporate the MaxCutPool pooling layer into the AMCG framework, potentially enhancing its ability to efficiently encode molecular graphs into a compact latent space. However, to truly advance the field of de novo drug design, we must move beyond optimizing individual components and work towards more integrated, context-aware systems. Figure 6.1 illustrates our vision for an ideal drug design model that leverages all available in silico and experimental information. This comprehensive framework would integrate molecular

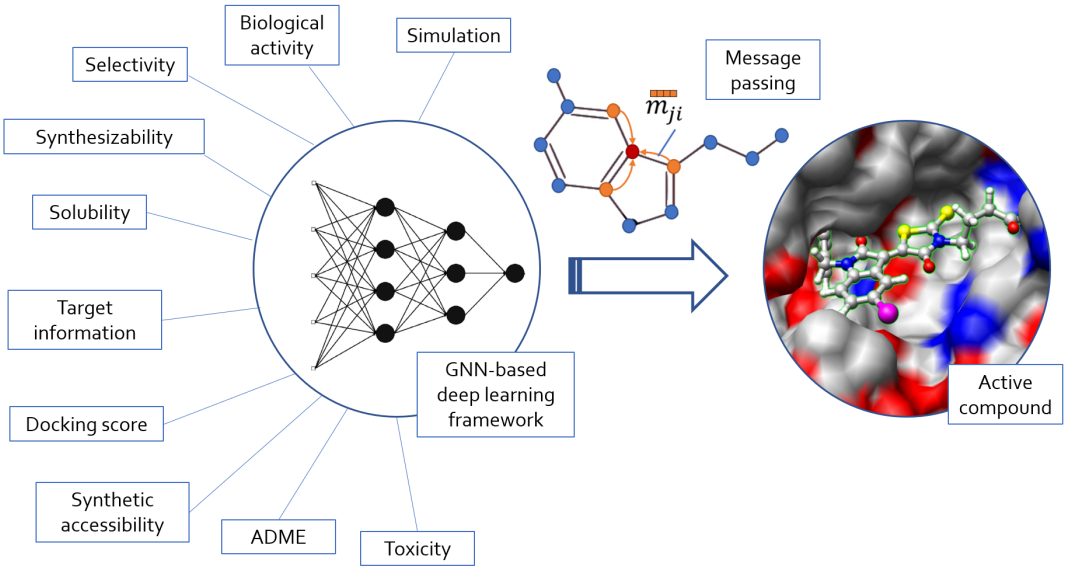

Figure 6.1: An ideal model for designing drug candidates should incorporate all available in silico information (including simulations and machine learning predictors) as well as experimental data. This can be achieved through conditioning and context awareness.

generation, target binding prediction, ADMET property prediction, synthetic accessibility assessment, ligandability prediction, experimental feedback mechanisms, and multi-objective optimization. Such an integrated approach could significantly

accelerate the identification of promising drug candidates by simultaneously optimizing for multiple, often conflicting objectives while maintaining chemical validity and synthetic feasibility. Realizing this vision presents several challenges, including balancing model complexity and interpretability, efficiently integrating experimental validation, ensuring data quality and quantity, optimizing computational efficiency, and effectively incorporating domain knowledge.

In conclusion, the work presented in this thesis contributes to the current transformation of drug discovery via artificial intelligence and machine learning. By combining theoretical advancements in graph representation learning with practical tools for molecular generation and analysis, we aim to pave the way for more efficient and effective drug discovery processes. The path towards comprehensive, target-aware de novo drug design systems is challenging, but the potential impact on drug discovery and human health is profound.

Publications

[P1] Abate C, Decherchi S, Cavalli A. Graph neural networks for conditional de novo drug design. WIREs Computational Molecular Science. 2023;13(4):e1651. Available from: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1651.

[P2] Abate C, Decherchi S, Cavalli A. AMCG: a graph dual atomic-molecular conditional molecular generator. Machine Learning: Science and Technology. 2024 jul;5(3):035004. Available from: https://dx.doi.org/10.1088/2632-2153/ad5bbf.

[P3] Abate C, Bianchi FM. MaxCutPool: differentiable feature-aware Maxcut for pooling in graph neural networks. In: The Thirteenth International Conference on Learning Representations; 2025. Available from: https://openreview.net/forum?id=xlbXRJ2XCP.

[P4] Di Palma F, Abate C, Decherchi S, Cavalli A. Ligandability and druggability assessment via machine learning. WIREs Computational Molecular Science. 2023;13(5):e1676. Available from: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1676.

Bibliography

[1] DiMasi JA, Grabowski HG, Hansen RW. Innovation in the pharmaceutical industry: New estimates of R&D costs. J Health Econ. 2016 Feb;47:20-33.

[2] Paul SM, Mytelka DS, Dunwiddie CT, Persinger CC, Munos BH, Lindborg SR, et al. How to improve R&D productivity: the pharmaceutical industry's grand challenge. Nature Reviews Drug Discovery. 2010 Mar;9(3):203-14. Available from: https://doi.org/10.1038/nrd3078.

[3] Jameson JL, Longo DL. Precision medicine–personalized, problematic, and promising. N Engl J Med. 2015 May;372(23):2229-34.

[4] Hughes JP, Rees S, Kalindjian SB, Philpott KL. Principles of early drug discovery. Br J Pharmacol. 2011 Mar;162(6):1239-49.

[5] Schneider P, Walters WP, Plowright AT, Sieroka N, Listgarten J, Goodnow RA, et al. Rethinking drug design in the artificial intelligence era. Nature Reviews Drug Discovery. 2020 May;19(5):353-64. Available from: https://doi.org/10.1038/s41573-019-0050-3.

[6] Macarron R, Banks MN, Bojanic D, Burns DJ, Cirovic DA, Garyantes T, et al. Impact of high-throughput screening in biomedical research. Nat Rev Drug Discov. 2011 Mar;10(3):188-95.

[7] Inglese J, Auld DS, Jadhav A, Johnson RL, Simeonov A, Yasgar A, et al. Quantitative high-throughput screening: a titration-based approach that efficiently identifies biological activities in large chemical libraries. Proc Natl Acad Sci U S A. 2006 Jul;103(31):11473-8.

[8] Roy R, Singh SK, Ahmad N, Misra S. Challenges and advancements in high-throughput screening strategies for cancer therapeutics. Global Translarional Medicine. 2024;3(1):2448.

[9] Longwell CK, Labanieh L, Cochran JR. High-throughput screening technologies for enzyme engineering. Current Opinion in Biotechnology. 2017;48:196-202. Chemical biotechnology • Pharmaceutical biotechnology. Available from: https://www.sciencedirect.com/science/article/pii/S0958166917300708.

[10] Wildey MJ, Haunso A, Tudor M, Webb M, Connick JH. Chapter Five - High-Throughput Screening. In: Goodnow RA, editor. Platform Technologies in Drug Discovery and Validation. vol. 50 of Annual Reports in Medicinal Chemistry. Academic Press; 2017. p. 149-95. Available from: https://www.sciencedirect.com/science/article/pii/S0065774317300076.

[11] Lyne PD. Structure-based virtual screening: an overview. Drug Discov Today. 2002 Oct;7(20):1047-55.

[12] Kitchen DB, Decornez H, Furr JR, Bajorath J. Docking and scoring in virtual screening for drug discovery: methods and applications. Nat Rev Drug Discov. 2004 Nov;3(11):935-49.

[13] Fan J, Fu A, Zhang L. Progress in molecular docking. Quantitative Biology. 2019 Jun;7(2):83-9. Available from: https://doi.org/10.1007/s40484-019-0172-y.

[14] Kroemer RT. Structure-based drug design: docking and scoring. Curr Protein Pept Sci. 2007 Aug;8(4):312-28.

[15] Ripphausen P, Nisius B, Bajorath J. State-of-the-art in ligand-based virtual screening. Drug Discovery Today. 2011;16(9):372-6. Available from: https://www.sciencedirect.com/science/article/pii/S1359644611000626.

[16] Schaller D, Šribar D, Noonan T, Deng L, Nguyen TN, Pach S, et al. Next generation 3D pharmacophore modeling. WIREs Computational Molecular

Science. 2020;10(4):e1468. Available from: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1468.

[17] Cherkasov A, Muratov EN, Fourches D, Varnek A, Baskin II, Cronin M, et al. QSAR Modeling: Where Have You Been? Where Are You Going To? Journal of Medicinal Chemistry. 2014 Jun;57(12):4977-5010. Available from: https://doi.org/10.1021/jm4004285.

[18] Gramatica P. Principles of QSAR models validation: internal and external. QSAR & Combinatorial Science. 2007;26(5):694-701. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/qsar.200610151.

[19] Nicholls A, McGaughey GB, Sheridan RP, Good AC, Warren G, Mathieu M, et al. Molecular Shape and Medicinal Chemistry: A Perspective. Journal of Medicinal Chemistry. 2010 May;53(10):3862-86. Available from: https://doi.org/10.1021/jm900818s.

[20] Ballester PJ, Richards WG. Ultrafast shape recognition to search compound databases for similar molecular shapes. J Comput Chem. 2007 Jul;28(10):1711-23.

[21] Gorgulla C. Recent Developments in Ultralarge and Structure-Based Virtual Screening Approaches. Annu Rev Biomed Data Sci. 2023 May;6:229-58.

[22] Schneider G. Virtual screening: an endless staircase? Nature Reviews Drug Discovery. 2010 Apr;9(4):273-6. Available from: https://doi.org/10.1038/nrd3139.

[23] Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T. The rise of deep learning in drug discovery. Drug Discovery Today. 2018;23(6):1241-50. Available from: https://www.sciencedirect.com/science/article/pii/S1359644617303598.

[24] Vamathevan J, Clark D, Czodrowski P, Dunham I, Ferran E, Lee G, et al. Applications of machine learning in drug discovery and development. Nature Reviews Drug Discovery. 2019 Jun;18(6):463-77. Available from: https://doi.org/10.1038/s41573-019-0024-5.

[25] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015 May;521(7553):436-44. Available from: https://doi.org/10.1038/nature14539.

[26] Gawehn E, Hiss JA, Schneider G. Deep Learning in Drug Discovery. Molecular Informatics. 2016;35(1):3-14. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.201501008.

[27] Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE transactions on pattern analysis and machine intelligence. 2013 August;35(8):1798—1828. Available from: http://www.cs.princeton.edu/courses/archive/spring13/cos598C/RepresentationLearning-AReviewandNewPerspectives.pdf.

[28] Meyers J, Fabian B, Brown N. De novo molecular design and generative models. Drug Discovery Today. 2021;26(11):2707-15. Available from: https://www.sciencedirect.com/science/article/pii/S1359644621002531.

[29] Zeng X, Wang F, Luo Y, gu Kang S, Tang J, Lightstone FC, et al. Deep generative molecular design reshapes drug discovery. Cell Reports Medicine. 2022;3(12):100794. Available from: https://www.sciencedirect.com/science/article/pii/S2666379122003494.

[30] Deng J, Yang Z, Wang H, Ojima I, Samaras D, Wang F. A systematic study of key elements underlying molecular property prediction. Nature Communications. 2023 Oct;14(1):6395. Available from: https://doi.org/10.1038/s41467-023-41948-6.

[31] Li Z, Jiang M, Wang S, Zhang S. Deep learning methods for molecular representation and property prediction. Drug Discovery Today. 2022;27(12):103373. Available from: https://www.sciencedirect.com/science/article/pii/S135964462200366X.

[32] Mamoshina P, Vieira A, Putin E, Zhavoronkov A. Applications of Deep Learning in Biomedicine. Mol Pharm. 2016 Mar;13(5):1445-54.

[33] Lenselink EB, ten Dijke N, Bongers B, Papadatos G, van Vlijmen HWT, Kowalczyk W, et al. Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. Journal of Cheminformatics. 2017 Aug;9(1):45. Available from: https://doi.org/10.1186/s13321-017-0232-0.

[34] Smith JS, Isayev O, Roitberg AE. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. Chem Sci. 2017 Feb;8(4):3192-203.

[35] Martire S, Decherchi S, Cavalli A. OBIWAN: An Element-Wise Scalable Feed-Forward Neural Network Potential. J Chem Theory Comput. 2024 Jul;20(14):6287-302.

[36] Behler J, Parrinello M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. Phys Rev Lett. 2007 Apr;98:146401. Available from: https://link.aps.org/doi/10.1103/PhysRevLett.98.146401.

[37] Unke OT, Chmiela S, Sauceda HE, Gastegger M, Poltavsky I, Schütt KT, et al. Machine Learning Force Fields. Chemical Reviews. 2021 Aug;121(16):10142-86. Available from: https://doi.org/10.1021/acs.chemrev.0c01111.

[38] Noé F, Tkatchenko A, Müller KR, Clementi C. Machine Learning for Molecular Simulation [Journal Article]. Annual Review of Physical Chemistry. 2020;71(Volume 71, 2020):361-90. Available from: https://www.annualreviews.org/content/journals/10.1146/annurev-physchem-042018-052331.

[39] Duvenaud D, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T, Aspuru-Guzik A, et al. Convolutional networks on graphs for learning molecular fingerprints. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2. NIPS'15. Cambridge, MA, USA: MIT Press; 2015. p. 2224–2232.

[40] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for Quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17. JMLR.org; 2017. p. 1263–1272.

[41] Ying R, Bourgeois D, You J, Zitnik M, Leskovec J. GNNExplainer: Generating Explanations for Graph Neural Networks. Adv Neural Inf Process Syst. 2019 Dec;32:9240-51.

[42] Hu W, Liu B, Gomes J, Zitnik M, Liang P, Pande V, et al. Strategies for Pre-training Graph Neural Networks. International Conference on Learning Representations. 2020. Available from: https://openreview.net/forum?id=HJlWWJSFDH.

[43] Batatia I, Benner P, Chiang Y, Elena AM, Kovács DP, Riebesell J, et al.. A foundation model for atomistic materials chemistry; 2024. Available from: https://arxiv.org/abs/2401.00096.

[44] 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 Discovery Today: Technologies. 2020;37:1-12. Available from: https://www.sciencedirect.com/science/article/pii/S1740674920300305.

[45] Lim J, Ryu S, Park K, Choe YJ, Ham J, Kim WY. Predicting Drug–Target Interaction Using a Novel Graph Neural Network with 3D Structure-Embedded Graph Representation. Journal of Chemical Information and Modeling. 2019 Sep;59(9):3981-8. Available from: https://doi.org/10.1021/acs.jcim.9b00387.

[46] Coley C, Jin W, Rogers L, Jamison TF, Jaakkola TS, Green WH, et al. A graph-convolutional neural network model for the prediction of chemical reactivity. Chem Sci. 2019;10:370-7. Available from: http://dx.doi.org/10.1039/C8SC04228D.

[47] Xiong J, Xiong Z, Chen K, Jiang H, Zheng M. Graph neural networks for

automated de novo drug design. Drug Discovery Today. 2021;26(6):1382-93. Available from: https://www.sciencedirect.com/science/article/pii/S1359644621000787.

[48] 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 9;6(35):22945-54. Available from: https://doi.org/10.1021/acsomega.1c03613.

[49] Aliper A, Plis S, Artemov A, Ulloa A, Mamoshina P, Zhavoronkov A. Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data. Molecular Pharmaceutics. 2016 Jul;13(7):2524-30. Available from: https://doi.org/10.1021/acs.molpharmaceut.6b00248.

[50] Menestrina L. Network analysis and machine learning assist drug repurposing and safety assessment in neurological diseases [PhD Thesis]. Bologna, Italy: Alma Mater Studiorum - Università di Bologna; 2024. Keywords: Drug Research, Graph Theory, Knowledge Graph, Knowledge Graph Embedding Model, Machine Learning, Neurological Diseases. Available from: http://amsdottorato.unibo.it/11318/.

[51] Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.. vol. 2; 2005. p. 729-34 vol. 2.

[52] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The Graph Neural Network Model. IEEE Transactions on Neural Networks. 2009;20(1):61-80.

[53] Spielman DA. Spectral Graph Theory and its Applications. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07); 2007. p. 29-38.

[54] Zhang Z, Cui P, Zhu W. Deep Learning on Graphs: A Survey. IEEE Transactions on Knowledge and Data Engineering. 2020;14(8):1-1.

[55] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems. 2021;32(1):4-24.

[56] Frasconi P, Gori M, Sperduti A. A general framework for adaptive processing of data structures. IEEE Transactions on Neural Networks. 1998;9(5):768-86.

[57] 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. Available from: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[58] Chakrabarti D, Faloutsos C. Graph mining: Laws, generators, and algorithms. ACM Comput Surv. 2006 Jun;38(1):2–es. Available from: https://doi.org/10.1145/1132952.1132954.

[59] Erdös P, Rényi A. On Random Graphs I. Publicationes Mathematicae Debrecen. 1959;6:290.

[60] Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. Nature. 1998 6;393(6684):440-2. Available from: https://doi.org/10.1038/30918.

[61] Albert R, Barabási AL. Statistical mechanics of complex networks. Reviews of Modern Physics. 2002;74(1):47-97.

[62] Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. Journal of Chemical Information and Computer Sciences. 1988 Feb;28(1):31-6. Available from: https://doi.org/10.1021/ci00057a005.

[63] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Comput. 1997 Nov;9(8):1735–1780. Available from: https://doi.org/10.1162/neco.1997.9.8.1735.

[64] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Advances in Neural Infor-

mation Processing Systems. vol. 30. Curran Associates, Inc.; 2017. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[65] Gupta A, Müller AT, Huisman BJH, Fuchs JA, Schneider P, Schneider G. Generative Recurrent Networks for De Novo Drug Design. Molecular Informatics. 2018;37(1-2):1700111. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.201700111.

[66] 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. Available from: https://doi.org/10.1021/acsomega.1c05145.

[67] Yu L, Zhang W, Wang J, Yu Y. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI'17. AAAI Press; 2017. p. 2852–2858.

[68] Ertl P, Lewis R, Martin E, Polyakov V. In silico generation of novel, drug-like chemical matter using the LSTM neural network. arXiv preprint arXiv:171207449. 2017.

[69] 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. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.201700123.

[70] Olivecrona M, Blaschke T, Engkvist O, Chen H. Molecular de-novo design through deep reinforcement learning. Journal of Cheminformatics. 2017 9;9(1):48. Available from: https://doi.org/10.1186/s13321-017-0235-x.

[71] Segler MHS, Kogej T, Tyrchan C, Waller MP. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ACS Central Science. 2018 1;4(1):120-31. Available from: https://doi.org/10.1021/acscentsci.7b00512.

[72] Dai H, Tian Y, Dai B, Skiena S, Song L. Syntax-Directed Variational Autoencoder for Structured Data. CoRR. 2018;abs/1802.08786. Available from: http://arxiv.org/abs/1802.08786.

[73] Alperstein Z, Cherkasov A, Rolfe JT. All smiles variational autoencoder. arXiv preprint arXiv:190513343. 2019.

[74] 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. Journal of Cheminformatics. 2019 11;11(1):71. Available from: https://doi.org/10.1186/s13321-019-0393-0.

[75] Bjerrum EJ, Sattarov B. Improving chemical autoencoder latent space and molecular de novo generation diversity with heteroencoders. Biomolecules. 2018;8(4):131.

[76] Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. Machine Learning: Science and Technology. 2020;1(4):045024.

[77] Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language Models are Few-Shot Learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in Neural Information Processing Systems. vol. 33. Curran Associates, Inc.; 2020. p. 1877-901. Available from: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[78] Cavanagh JM, Sun K, Gritsevskiy A, Bagni D, Bannister TD, Head-Gordon T. SmileyLlama: Modifying Large Language Models for Directed Chemical Space Exploration; 2024. Available from: https://arxiv.org/abs/2409.02231.

[79] Ahmad W, Simon E, Chithrananda S, Grand G, Ramsundar B. ChemBERTa-2: Towards Chemical Foundation Models; 2022. Available from: https://arxiv.org/abs/2209.01712.

[80] Satorras VG, Hoogeboom E, Welling M. E(n) Equivariant Graph Neural Networks. In: Meila M, Zhang T, editors. Proceedings of the 38th International Conference on Machine Learning. vol. 139 of Proceedings of Machine Learning Research. PMLR; 2021. p. 9323-32. Available from: https://proceedings.mlr.press/v139/satorras21a.html.

[81] Polishchuk PG, Madzhidov TI, Varnek A. Estimation of the size of drug-like chemical space based on GDB-17 data. Journal of Computer-Aided Molecular Design. 2013;27(8):675-9.

[82] Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated Graph Sequence Neural Networks. In: Proceedings of the 4th International Conference on Learning Representations. ICLR '16; 2016. Available from: http://arxiv.org/abs/1511.05493.

[83] Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. In: Proceedings of the 5th International Conference on Learning Representations. ICLR '17; 2017. Available from: https://openreview.net/forum?id=SJU4ayYgl.

[84] Veličković P, Casanova A, Liò P, Cucurull G, Romero A, Bengio Y. Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings; 2018. Available from: https://openreview.net/forum?id=rJXMpikCZ.

[85] Schlichtkrull M, Kipf TN, Bloem P, Berg Rvd, Titov I, Welling M. Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. Springer; 2018. p. 593-607. Available from: https://doi.org/10.1007/978-3-319-93417-4_38.

[86] Simonovsky M, Komodakis N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 3693-702. Available from: https://doi.org/10.48550/arXiv.1704.02901.

[87] Danel T, Spurek P, Tabor J, Smieja M, Struski L, Slowik A, et al. Spatial Graph Convolutional Networks. In: 5th International Conference on Neural Information Processing. vol. 1333 of Communications in Computer and Information Science. Springer; 2020. p. 668-75. Available from: https://doi.org/10.1007/978-3-030-63823-8_76.

[88] Kearnes S, McCloskey K, Berndl M, Pande V, Riley P. Molecular graph convolutions: moving beyond fingerprints. Journal of Computer-Aided Molecular Design. 2016;30(8):595-608. Available from: http://dx.doi.org/10.1007/s10822-016-9938-8.

[89] Khasahmadi AH, Hassani K, Moradi P, Lee L, Morris Q. Memory-Based Graph Networks. In: International Conference on Learning Representations; 2020. .

[90] Gao H, Ji S. Graph u-nets. In: international conference on machine learning. PMLR; 2019. p. 2083-92.

[91] Ma Z, Xuan J, Wang YG, Li M, Liò P. Path integral based convolution and pooling for graph neural networks. Advances in Neural Information Processing Systems. 2020;33:16421-33.

[92] Liu N, Jian S, Li D, Zhang Y, Lai Z, Xu H. Hierarchical adaptive pooling by capturing high-order dependency for graph representation learning. IEEE Transactions on Knowledge and Data Engineering. 2021;35(4):3952-65.

[93] Cini A, Mandic D, Alippi C. Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting. International Conference on Machine Learning. 2024.

[94] Marisca I, Alippi C, Bianchi FM. Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling. In: Proceedings of the 41st International Conference on Machine Learning. vol. 235 of Proceedings of Machine Learning Research. PMLR; 2024. p. 34846-65.

[95] Grattarola D, Zambon D, Bianchi FM, Alippi C. Understanding pooling in graph neural networks. IEEE Transactions on Neural Networks and Learning

Systems. 2022.

[96] Ying Z, You J, Morris C, Ren X, Hamilton W, Leskovec J. Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems. 2018;31.

[97] Bianchi FM, Grattarola D, Alippi C. Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning. PMLR; 2020. p. 874-83.

[98] Yuan H, Ji S. Structpool: Structured graph pooling via conditional random fields. In: Proceedings of the 8th International Conference on Learning Representations; 2020. .

[99] Duval A, Malliaros F. Higher-order clustering and pooling for graph neural networks. In: Proceedings of the 31st ACM international conference on information & knowledge management; 2022. p. 426-35.

[100] Tsitsulin A, Palowitch J, Perozzi B, Müller E. Graph Clustering with Graph Neural Networks. J Mach Learn Res. 2023;24:127:1-127:21.

[101] Bianchi FM, Lachi V. The expressive power of pooling in Graph Neural Networks. In: Advances in Neural Information Processing Systems. vol. 36; 2023. p. 71603-18.

[102] Knyazev B, Taylor GW, Amer M. Understanding attention and generalization in graph neural networks. Advances in neural information processing systems. 2019;32.

[103] Ranjan E, Sanyal S, Talukdar P. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34; 2020. p. 5470-7.

[104] Lee J, Lee I, Kang J. Self-attention graph pooling. In: International conference on machine learning. PMLR; 2019. p. 3734-43.

[105] Gao H, Liu Y, Ji S. Topology-Aware Graph Pooling Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021 dec;43(12):4512-8.

[106] Pang Y, Zhao Y, Li D. Graph pooling via coarsened graph infomax. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2021. p. 2177-81.

[107] Gao X, Dai W, Li C, Xiong H, Frossard P. iPool—Information-Based Pooling in Hierarchical Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems. 2022;33(9):5032-44.

[108] Wang P, Luo J, Shen Y, Heng S, Luo X. A Comprehensive Graph Pooling Benchmark: Effectiveness, Robustness and Generalizability. arXiv preprint arXiv:240609031. 2024.

[109] Zhang Z, Bu J, Ester M, Zhang J, Yao C, Yu Z, et al. Hierarchical graph pooling with structure learning. arXiv preprint arXiv:191105954. 2019.

[110] Noutahi E, Beaini D, Horwood J, Giguère S, Tossou P. Towards interpretable sparse graph representation learning with laplacian pooling. arXiv preprint arXiv:190511577. 2019.

[111] Bacciu D, Conte A, Landolfi F. Graph Pooling with Maximum-Weight k-Independent Sets. In: Thirty-Seventh AAAI Conference on Artificial Intelligence; 2023. .

[112] Dhillon IS, Guan Y, Kulis B. Weighted graph cuts without eigenvectors a multilevel approach. IEEE transactions on pattern analysis and machine intelligence. 2007;29(11):1944-57.

[113] Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems. 2016;29.

[114] Wu J, Chen X, Xu K, Li S. Structural entropy guided graph hierarchical pooling. In: International conference on machine learning. PMLR; 2022. p. 24017-30.

[115] Bianchi FM, Grattarola D, Livi L, Alippi C. Hierarchical representation learning in graph neural networks with node decimation pooling. IEEE Transactions on Neural Networks and Learning Systems. 2020;33(5):2195-207.

[116] Mavrovouniotis ML, Chang S. Hierarchical neural networks. Computers & Chemical Engineering. 1992;16(4):347-69. Available from: https://doi.org/10.1016/0098-1354(92)80053-C.

[117] Kingma DP, Welling M. Auto-Encoding Variational Bayes. In: 2nd International Conference on Learning Representations; 2014. Available from: http://arxiv.org/abs/1312.6114v10.

[118] Doersch C. Tutorial on Variational Autoencoders; 2016.

[119] Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, et al. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Central Science. 2018 Feb;4(2):268-76. Available from: https://doi.org/10.1021/acscentsci.7b00572.

[120] Lucas J, Tucker G, Grosse R, Norouzi M. Understanding Posterior Collapse in Generative Latent Variable Models. DeepGenStruct workshop, ICLR. 2019. Available from: https://openreview.net/forum?id=r1xaVLUYuE.

[121] Asperti A, Trentin M. Balancing Reconstruction Error and Kullback-Leibler Divergence in Variational Autoencoders. IEEE Access. 2020;8:199440-8.

[122] Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, et al. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In: International Conference on Learning Representations; 2017. Available from: https://openreview.net/forum?id=Sy2fzU9gl.

[123] Kim Y, Wiseman S, Miller A, Sontag D, Rush A. Semi-Amortized Variational Autoencoders. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. PMLR; 2018. p. 2678-87. Available from: https://proceedings.mlr.press/v80/kim18e.html.

[124] Zhao S, Song J, Ermon S. InfoVAE: balancing learning and inference in variational autoencoders. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Arti-

ficial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'19/IAAI'19/EAAI'19. AAAI Press; 2019. Available from: https://doi.org/10.1609/aaai.v33i01.33015885.

[125] 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. 2014;27. Available from: https://doi.org/10.48550/arXiv.1406.2661.

[126] Wiatrak M, Albrecht S. Stabilizing Generative Adversarial Network Training: A Survey; 2019.

[127] Kossale Y, Airaj M, Darouichi A. Mode Collapse in Generative Adversarial Networks: An Overview. In: 2022 8th International Conference on Optimization and Applications (ICOA); 2022. p. 1-6.

[128] Zhu JY, Park T, Isola P, Efros AA. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In: 2017 IEEE International Conference on Computer Vision; 2017. p. 2242-51. Available from: https://doi.org/10.1109/ICCV.2017.244.

[129] Kobyzev I, Prince SJD, Brubaker MA. Normalizing Flows: An Introduction and Review of Current Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021;43(11):3964-79. Available from: https://doi.org/10.1109/TPAMI.2020.2992934.

[130] Madhawa K, Ishiguro K, Nakago K, Abe M. GraphNVP: An Invertible Flow Model for Generating Molecular Graphs. arXiv; 2019. Available from: https://arxiv.org/abs/1905.11600.

[131] 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.

[132] Song Y, Ermon S. Generative Modeling by Estimating Gradients of the Data Distribution. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc.; 2019. Available from: https://dl.acm.org/doi/abs/10.5555/3454287.3455354.

[133] Karatzas I, Shreve S. Brownian Motion and Stochastic Calculus. Graduate Texts in Mathematics (113) (Book 113). Springer New York; 1991. Available from: https://books.google.it/books?id=ATNy_Zg3PSsC.

[134] Welling M, Teh YW. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. ICML'11. Madison, WI, USA; 2011. p. 681–688. Available from: https://dl.acm.org/doi/10.5555/3104482.3104568.

[135] Leimkuhler B, Matthews C. Molecular Dynamics: With Deterministic and Stochastic Numerical Methods. Interdisciplinary Applied Mathematics. Springer International Publishing; 2015. Available from: https://books.google.it/books?id=-5oxrgEACAAJ.

[136] Dhariwal P, Nichol A. Diffusion models beat GANs on image synthesis. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21. Red Hook, NY, USA: Curran Associates Inc.; 2021. .

[137] Ho J, Jain A, Abbeel P. Denoising Diffusion Probabilistic Models. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20. Red Hook, NY, USA: Curran Associates Inc.; 2020. Available from: https://dl.acm.org/doi/abs/10.5555/3495724.3496298.

[138] Alakhdar A, Poczos B, Washburn N. Diffusion Models in De Novo Drug Design. Journal of Chemical Information and Modeling. 2024 Sep. Available from: https://doi.org/10.1021/acs.jcim.4c01107.

[139] Xu M, Yu L, Song Y, Shi C, Ermon S, Tang J. GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation. In: International Conference on Learning Representations; 2022. Available from: https://openreview.net/forum?id=PzcvxEMzvQC.

[140] Igashov I, Stärk H, Vignac C, Schneuing A, Satorras VG, Frossard P, et al. Equivariant 3D-conditional diffusion model for molecular linker design. Nature Machine Intelligence. 2024 Apr;6(4):417-27. Available from: https://doi.org/10.1038/s42256-024-00815-9.

[141] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press; 2018.

[142] Wang Q, Wei Z, Hu X, Wang Z, Dong Y, Liu H. Molecular generation strategy and optimization based on A2C reinforcement learning in de novo drug design. Bioinformatics. 2023 11;39(11):btad693. Available from: https://doi.org/10.1093/bioinformatics/btad693.

[143] Devidze R, Radanovic G, Kamalaruban P, Singla A. Explicable Reward Design for Reinforcement Learning Agents. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Vaughan JW, editors. Advances in Neural Information Processing Systems. vol. 34. Curran Associates, Inc.; 2021. p. 20118-31. Available from: https://proceedings.neurips.cc/paper_files/paper/2021/file/a7f0d2b95c60161b3f3c82f764b1d1c9-Paper.pdf.

[144] Müller N, Glasmachers T. Challenges in High-Dimensional Reinforcement Learning with Evolution Strategies. In: Auger A, Fonseca CM, Lourenço N, Machado P, Paquete L, Whitley D, editors. Parallel Problem Solving from Nature – PPSN XV. Cham: Springer International Publishing; 2018. p. 411-23.

[145] Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In: Proceedings of the International Conference on Learning Representations (ICLR); 2016. .

[146] Fu T, Xiao C, Sun J. CORE: Automatic Molecule Optimization Using Copy & Refine Strategy. Proceedings of the AAAI Conference on Artificial Intelligence. 2020 Apr;34(01):638-45. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/5404.

[147] Jin W, Yang K, Barzilay R, Jaakkola T. Learning Multimodal Graph-to-Graph Translation for Molecule Optimization. In: 7th International Conference on Learning Representations; 2019. Available from: https://openreview.net/forum?id=B1xJAsA5F7.

[148] Tan C, Gao Z, Li SZ. Target-aware Molecular Graph Generation. arXiv; 2022. Available from: https://arxiv.org/abs/2202.04829.

[149] Maziarka Ł, Pocha A, Kaczmarczyk J, Rataj K, Danel T, Warchoł M. Mol-CycleGAN: A generative model for molecular optimization. Journal of Cheminformatics. 2020;12(1):1-18. Available from: https://doi.org/10.1186/s13321-019-0404-1.

[150] Fu T, Xiao C, Glass LM, Sun J. MOLER: Incorporate Molecule-Level Reward to Enhance Deep Generative Model for Molecule Optimization. IEEE Transactions on Knowledge and Data Engineering. 2022;34(11):5459-71.

[151] Liu Q, Allamanis M, Brockschmidt M, Gaunt AL. Constrained Graph Variational Autoencoders for Molecule Design. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18. Red Hook, NY, USA: Curran Associates Inc.; 2018. p. 7806–7815. Available from: https://dl.acm.org/doi/10.5555/3327757.3327877.

[152] Jin W, Barzilay R, Jaakkola T. Junction tree variational autoencoder for molecular graph generation. In: International conference on machine learning. PMLR; 2018. p. 2323-32.

[153] Simonovsky M, Komodakis N. GraphVAE: Towards generation of small graphs using variational autoencoders. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lec-

ture Notes in Bioinformatics). 2018;11139 LNCS:412-22. Available from: https://doi.org/10.1007/978-3-030-01418-6_41.

[154] 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.

[155] Li Y, Zhang L, Liu Z. Multi-objective de novo drug design with conditional graph generative model. Journal of Cheminformatics. 2018;10(1):1-24. Available from: https://doi.org/10.1186/s13321-018-0287-6.

[156] You J, Liu B, Ying R, Pande V, Leskovec J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18. Red Hook, NY, USA: Curran Associates Inc.; 2018. p. 6412–6422. Available from: https://dl.acm.org/doi/10.5555/3327345.3327537.

[157] Popova M, Shvets M, Oliva J, Isayev O. MolecularRNN: Generating realistic molecular graphs with optimized properties; 2019. Available from: https://arxiv.org/abs/1905.13372.

[158] Shi C, Xu M, Zhu Z, Zhang W, Zhang M, Tang J. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. In: 8th International Conference on Learning Representations; 2020. Available from: https://openreview.net/forum?id=S1esMkHYPr.

[159] Luo Y, Yan K, Ji S. GraphDF: A Discrete Flow Model for Molecular Graph Generation. In: Meila M, Zhang T, editors. Proceedings of the 38th International Conference on Machine Learning. vol. 139 of Proceedings of Machine Learning Research. PMLR; 2021. p. 7192-203. Available from: https://proceedings.mlr.press/v139/luo21a.html.

[160] Jin W, Barzilay R, Jaakkola T. Hierarchical Generation of Molecular Graphs Using Structural Motifs. In: Proceedings of the 37th International Conference

on Machine Learning. ICML'20. JMLR.org; 2020. Available from: https://dl.acm.org/doi/abs/10.5555/3524938.3525387.

[161] Atance SR, Diez JV, Engkvist O, Olsson S, Mercado R. De Novo Drug Design Using Reinforcement Learning with Graph-Based Deep Generative Models. Journal of Chemical Information and Modeling. 2022 10;62(20):4863-72. Available from: https://doi.org/10.1021/acs.jcim.2c00838.

[162] Lim J, Hwang SY, Moon S, Kim S, Kim WY. Scaffold-based molecular design with a graph generative model. Chemical Science. 2020;11:1153-64. Available from: http://dx.doi.org/10.1039/C9SC04503A.

[163] 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. Journal of Cheminformatics. 2020;12(1):1-17. Available from: https://doi.org/10.1186/s13321-020-00454-3.

[164] Samanta B, De A, Jana G, Gomez V, Chattaraj PK, Ganguly N, et al. NEVAE: A Deep Generative Model for Molecular Graphs. Journal of Machine Learning Research. 2022 06;21(1). Available from: https://doi.org/10.1609/aaai.v33i01.33011110.

[165] Mahmood O, Mansimov E, Bonneau R, Cho K. Masked graph modeling for molecule generation. Nature Communications. 2021;12(1). Available from: http://dx.doi.org/10.1038/s41467-021-23415-2.

[166] Kwon Y, Yoo J, Choi YS, Son WJ, Lee D, Kang S. Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation. Journal of Cheminformatics. 2019;11(1):1-10. Available from: https://doi.org/10.1186/s13321-019-0396-x.

[167] Assouel R, Ahmed M, Segler MH, Saffari A, Bengio Y. DEFactor: Differentiable Edge Factorization-based Probabilistic Graph Generation; 2018. Available from: https://arxiv.org/abs/1811.09766.

[168] Zang C, Wang F. MoFlow: An Invertible Flow Model for Generating Molecular Graphs. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 617–626. Available from: https://doi.org/10.1145/3394486.3403104.

[169] Ma C, Zhang X. GF-VAE: A Flow-Based Variational Autoencoder for Molecule Generation. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. CIKM '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 1181–1190. Available from: https://doi.org/10.1145/3459637.3482260.

[170] Flam-Shepherd D, Wu TC, Aspuru-Guzik A. MPGVAE: improved generation of small organic molecules using message passing neural nets. Machine Learning: Science and Technology. 2021 07;2(4):045010. Available from: https://doi.org/10.1088/2632-2153/abf5b7.

[171] Kuznetsov M, Polykovskiy D. MolGrow: A Graph Normalizing Flow for Hierarchical Molecular Generation. In: Proceedings of the Thirty-Fifth Conference on Association for the Advancement of Artificial Intelligence (AAAI); 2021. p. 8226-34. Available from: https://doi.org/10.48550/arXiv.2106.05856.

[172] 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. vol. 32. Curran Associates, Inc.; 2019. Available from: https://proceedings.neurips.cc/paper/2019/file/46d0671dd4117ea366031f87f3aa0093-Paper.pdf.

[173] Xie Y, Shi C, Zhou H, Yang Y, Zhang W, Yu Y, et al. MARS: Markov Molecular Sampling for Multi-objective Drug Discovery. In: International Conference on Learning Representations; 2021. Available from: https://openreview.net/forum?id=kHSu4ebxFXY.

[174] Imrie F, Bradley AR, van der Schaar M, Deane CM. Deep Generative Models for 3D Linker Design. Journal of Chemical Information and Modeling. 2020;60(4):1983-95. Available from: https://doi.org/10.1021/acs.jcim.9b01120.

[175] Jin W, Barzilay R, Jaakkola T. Multi-Objective Molecule Generation Using Interpretable Substructures. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20. JMLR.org; 2020. Available from: https://dl.acm.org/doi/abs/10.5555/3524938.3525388.

[176] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis, Minnesota: Association for Computational Linguistics; 2019. p. 4171-86. Available from: https://aclanthology.org/N19-1423.

[177] Bresson X, Laurent T. A Two-Step Graph Convolutional Decoder for Molecule Generation; 2019. Available from: https://arxiv.org/abs/1906.03412.

[178] Jo J, Lee S, Hwang SJ. Score-based Generative Modeling of Graphs via the System of Stochastic Differential Equations. In: Thirty-ninth International Conference on Machine Learning; 2022. Available from: https://doi.org/10.48550/arXiv.2202.02514.

[179] Mercado R, Rastemo T, Lindelöf E, Klambauer G, Engkvist O, Chen H, et al. Graph Networks for Molecular Design. Machine Learning: Science and Technology. 2020. Available from: https://doi.org/10.1088/2632-2153/abcf91.

[180] Randić M. On Canonical Numbering of Atoms in a Molecule and Graph Isomorphism. Journal of Chemical Information and Computer Sciences. 1977 08;17(3):171-80. Available from: https://doi.org/10.1021/ci60011a013.

[181] He K, Zhang X, Ren S, Sun J. Identity mappings in deep residual

networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2016;9908 LNCS:630-45. Available from: https://doi.org/10.1007/978-3-319-46493-0_38.

[182] 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. PMLR; 2018. p. 5708-17. Available from: https://proceedings.mlr.press/v80/you18a.html.

[183] Bongini P, Bianchini M, Scarselli F. Molecular generative Graph Neural Networks for Drug Discovery. Neurocomputing. 2021;450:242-52. Available from: https://doi.org/10.1016/j.neucom.2021.04.039.

[184] Papamakarios G, Pavlakou T, Murray I. Masked Autoregressive Flow for Density Estimation. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 2335–2344. Available from: https://dl.acm.org/doi/10.5555/3294771.3294994.

[185] Zhou Z, Kearnes S, Li L, Zare RN, Riley P. Optimization of Molecules via Deep Reinforcement Learning. Scientific Reports. 2019 Jul;9(1):10752. Available from: https://doi.org/10.1038/s41598-019-47148-x.

[186] Rigoni D, Navarin N, Sperduti A. Conditional Constrained Graph Variational Autoencoders for Molecule Design. In: IEEE Symposium Series on Computational Intelligence (SSCI); 2020. p. 729-36. Available from: https://doi.org/10.1109/SSCI47803.2020.9308554.

[187] Kearnes S, Li L, Riley P. Decoding Molecular Graph Embeddings with Reinforcement Learning. arXiv; 2019. Available from: https://arxiv.org/abs/1904.08915.

[188] Ma T, Chen J, Xiao C. Constrained Generation of Semantically Valid Graphs via Regularizing Variational Autoencoders. In: Proceedings of the 32nd In-

ternational Conference on Neural Information Processing Systems. NIPS'18. Red Hook, NY, USA: Curran Associates Inc.; 2018. p. 7113–7124. Available from: https://dl.acm.org/doi/10.5555/3327757.3327814.

[189] Courant R. Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Society. 1943;49(1):1 23. Available from: https://doi.org/10.1090/S0002-9904-1943-07818-4.

[190] Pölsterl S, Wachinger C. Adversarial Learned Molecular Graph Inference and Generation. In: Machine Learning and Knowledge Discovery in Databases: European Conference, Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag; 2020. p. 173–189. Available from: https://doi.org/10.1007/978-3-030-67661-2_11.

[191] Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL. Quantifying the chemical beauty of drugs. Nature Chemistry. 2012 1;4(2):90-8. Available from: https://doi.org/10.1038/nchem.1243.

[192] Ertl P, Schuffenhauer A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. Journal of Cheminformatics. 2009 6;1(1):8. Available from: https://doi.org/10.1186/1758-2946-1-8.

[193] Zheng S, Lei Z, Ai H, Chen H, Deng D, Yang Y. Deep scaffold hopping with multimodal transformer neural networks. Journal of Cheminformatics. 2021;13(1):1-15. Available from: https://doi.org/10.1186/s13321-021-00565-5.

[194] Snoek J, Larochelle H, Adams RP. Practical Bayesian Optimization of Machine Learning Algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2. NIPS'12. Red Hook, NY, USA: Curran Associates Inc.; 2012. p. 2951–2959. Available from: https://dl.acm.org/doi/10.5555/2999325.2999464.

[195] Snelson E, Ghahramani Z. Sparse Gaussian Processes Using Pseudo-Inputs.

In: Proceedings of the 18th International Conference on Neural Information Processing Systems. NIPS'05. Cambridge, MA, USA: MIT Press; 2005. p. 1257–1264. Available from: https://dl.acm.org/doi/10.5555/2976248.2976406.

[196] Ramakrishnan R, Dral PO, Rupp M, von Lilienfeld OA. Quantum chemistry structures and properties of 134 kilo molecules. Scientific Data. 2014 Aug;1(1):140022. Available from: https://doi.org/10.1038/sdata.2014.22.

[197] Irwin JJ, Shoichet BK. ZINC - A Free Database of Commercially Available Compounds for Virtual Screening. Journal of Chemical Information and Modeling. 2005 Jan;45(1):177-82. Available from: https://doi.org/10.1021/ci049714+.

[198] Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, et al. ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic Acids Research. 2011 09;40(D1):D1100-7. Available from: https://doi.org/10.1093/nar/gkr777.

[199] Knox C, Wilson M, Klinger CM, Franklin M, Oler E, Wilson A, et al. DrugBank 6.0: the DrugBank Knowledgebase for 2024. Nucleic Acids Res. 2024 Jan;52(D1):D1265-75.

[200] Bajusz D, Rácz A, Héberger K. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? Journal of Cheminformatics. 2015 May;7(1):20. Available from: https://doi.org/10.1186/s13321-015-0069-3.

[201] Renz P, Van Rompaey D, Wegner JK, Hochreiter S, Klambauer G. On failure modes in molecule generation and optimization. Drug Discovery Today: Technologies. 2019;32-33:55-63. Artificial Intelligence. Available from: https://www.sciencedirect.com/science/article/pii/S1740674920300159.

[202] Preuer K, Renz P, Unterthiner T, Hochreiter S, Klambauer G. Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. Journal of Chemical Information and Modeling. 2018 Sep;58(9):1736-41. Available from: https://doi.org/10.1021/acs.jcim.8b00234.

[203] Brown N, Fiscato M, Segler MHS, Vaucher AC. GuacaMol: Benchmarking Models for de Novo Molecular Design. Journal of Chemical Information and Modeling. 2019 Mar;59(3):1096-108. Available from: https://doi.org/10.1021/acs.jcim.8b00839.

[204] Polykovskiy D, Zhebrak A, Sanchez-Lengeling B, Golovanov S, Tatanov O, Belyaev S, et al. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. Frontiers in Pharmacology. 2020;11. Available from: https://www.frontiersin.org/journals/pharmacology/articles/10.3389/fphar.2020.565644.

[205] Vahdat A, Kreis K, Kautz J. Score-based Generative Modeling in Latent Space. In: Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, editors. Advances in Neural Information Processing Systems; 2021. Available from: https://openreview.net/forum?id=P9TYG0j-wtG.

[206] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models; 2021.

[207] Zhang Y, Li J, Xu C. Graph-Based Latent Space Traversal for New Molecules Discovery. In: Proceedings of the 16th International Symposium on Visual Information Communication and Interaction. VINCI '23. New York, NY, USA: Association for Computing Machinery; 2023. Available from: https://doi.org/10.1145/3615522.3615548.

[208] Ghosh P, Sajjadi MSM, Vergari A, Black M, Scholkopf B. From Variational to Deterministic Autoencoders. In: International Conference on Learning Representations; 2020. Available from: https://openreview.net/forum?id=S1g7tpEYDS.

[209] Godwin J, Schaarschmidt M, Gaunt AL, Sanchez-Gonzalez A, Rubanova

Y, Veličković P, et al. Simple GNN Regularisation for 3D Molecular Property Prediction and Beyond. In: International Conference on Learning Representations; 2022. Available from: https://openreview.net/forum?id=1wVvweK3oIb.

[210] Satorras VG, Hoogeboom E, Welling M. E(n) Equivariant Graph Neural Networks; 2021.

[211] Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 1955;2(1-2):83-97. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[212] Kipf TN, Welling M. Variational Graph Auto-Encoders; 2016.

[213] Saito–Tarashima N, Kinoshita M, Igata Y, Kashiwabara Y, Minakawa N. Replacement of oxygen with sulfur on the furanose ring of cyclic dinucleotides enhances the immunostimulatory effect via STING activation. RSC Med Chem. 2021;12:1519-24. Available from: http://dx.doi.org/10.1039/D1MD00114K.

[214] Nagy PI. Replacement of Oxygen by Sulfur in Small Organic Molecules. 3. Theoretical Studies on the Tautomeric Equilibria of the 2OH and 4OH-Substituted Oxazole and Thiazole and the 3OH and 4OH-Substituted Isoxazole and Isothiazole in the Isolated State and in Solution. Int J Mol Sci. 2016 Jul;17(7).

[215] Ficarra F, Silvi M. Atom-swap chemistry could aid drug discovery. Nature. 2023 November;623(7985):36-7. Available from: https://ideas.repec.org/a/nat/nature/v623y2023i7985d10.1038_d41586-023-03297-8.html.

[216] Stumpfe D, Hu H, Bajorath J. Evolving Concept of Activity Cliffs. ACS Omega. 2019 Aug;4(11):14360-8.

[217] RDKit contributors. RDKit: Open-source cheminformatics; 2023. Version 2023.03.3. Available from: https://www.rdkit.org.

[218] Decherchi S, Bottegoni G, Spitaleri A, Rocchia W, Cavalli A. BiKi Life Sciences: A New Suite for Molecular Dynamics and Related Methods in Drug Discovery. Journal of Chemical Information and Modeling. 2018 Feb;58(2):219-24. Available from: https://doi.org/10.1021/acs.jcim.7b00680.

[219] Riniker S, Landrum GA. Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation. Journal of Chemical Information and Modeling. 2015 Dec;55(12):2562-74. Available from: https://doi.org/10.1021/acs.jcim.5b00654.

[220] Rappe AK, Casewit CJ, Colwell KS, Goddard III WA, Skiff WM. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. Journal of the American Chemical Society. 1992 Dec;114(25):10024-35. Available from: https://doi.org/10.1021/ja00051a040.

[221] Dwivedi VP, Luu AT, Laurent T, Bengio Y, Bresson X. Graph Neural Networks with Learnable Structural and Positional Representations. In: International Conference on Learning Representations; 2022. Available from: https://openreview.net/forum?id=wTTjnvGphYj.

[222] Honda S, Akita H, Ishiguro K, Nakanishi T, Oono K. Graph Residual Flow for Molecular Graph Generation; 2019.

[223] Goemans MX, Williamson DP. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM. 1995 Nov;42(6):1115–1145. Available from: https://doi.org/10.1145/227683.227684.

[224] Shuman DI, Faraji MJ, Vandergheynst P. A multiscale pyramid transform for graph signals. IEEE Transactions on Signal Processing. 2015;64(8):2119-34.

[225] Trevisan L. Max cut and the smallest eigenvalue. In: Proceedings of the forty-first annual ACM symposium on Theory of computing; 2009. p. 263-72.

[226] Aspvall B, Gilbert JR. Graph Coloring Using Eigenvalue Decomposition. SIAM Journal on Algebraic Discrete Methods. 1984;5(4):526–538.

[227] von Luxburg U. A Tutorial on Spectral Clustering; 2007.

[228] Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K. Simplifying Graph Convolutional Networks. 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. 6861-71.

[229] Wang G, Ying R, Huang J, Leskovec J. Improving Graph Attention Networks with Large Margin-based Constraints; 2019.

[230] Eliasof M, Ruthotto L, Treister E. Improving graph neural networks with learnable propagation operators. In: International Conference on Machine Learning. PMLR; 2023. p. 9224-45.

[231] Bianchi FM. Simplifying clustering with graph neural networks. arXiv preprint arXiv:220708779. 2022.

[232] Mather PM, Koch M. Computer processing of remotely-sensed images. John Wiley & Sons; 2022.

[233] Xu K, Hu W, Leskovec J, Jegelka S. How Powerful are Graph Neural Networks? In: International Conference on Learning Representations; 2019. .

[234] Schuetz MJ, Brubaker JK, Katzgraber HG. Combinatorial optimization with physics-inspired graph neural networks. Nature Machine Intelligence. 2022;4(4):367-77.

[235] Defferrard M, Martin L, Pena R, Perraudin N. PyGSP: Graph Signal Processing in Python. Zenodo; 2017. Available from: https://doi.org/10.5281/zenodo.1003158.

[236] Ye Y. The gset dataset. Stanford; 2003.

[237] Diehl F. Edge contraction pooling for graph neural networks. arXiv preprint arXiv:190510990. 2019.

[238] Morris C, Kriege NM, Bause F, Kersting K, Mutzel P, Neumann M. TU-Dataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020); 2020. .

[239] Bianchi FM, Gallicchio C, Micheli A. Pyramidal Reservoir Graph Neural Network. Neurocomputing. 2022;470:389-404.

[240] Errica F, Podda M, Bacciu D, Micheli A. A Fair Comparison of Graph Neural Networks for Graph Classification. In: International Conference on Learning Representations; 2020. .

[241] Platonov O, Kuznedelev D, Diskin M, Babenko A, Prokhorenkova L. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In: The Eleventh International Conference on Learning Representations; 2023. .

[242] Lim D, Hohne F, Li X, Huang SL, Gupta V, Bhalerao O, et al. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. Advances in Neural Information Processing Systems. 2021;34:20887-902.

[243] Volkamer A, Kuhn D, Rippmann F, Rarey M. DoGSiteScorer: a web server for automatic binding site prediction, analysis and druggability assessment. Bioinformatics. 2012 May;28(15):2074-5.

[244] Borrel A, Regad L, Xhaard H, Petitjean M, Camproux AC. PockDrug: A Model for Predicting Pocket Druggability That Overcomes Pocket Estimation Uncertainties. Journal of Chemical Information and Modeling. 2015 Apr;55(4):882-95. Available from: https://doi.org/10.1021/ci5006004.

[245] Jiménez J, Doerr S, Martínez-Rosell G, Rose AS, De Fabritiis G. DeepSite: protein-binding site predictor using 3D-convolutional neural networks. Bioinformatics. 2017 05;33(19):3036-42. Available from: https://doi.org/10.1093/bioinformatics/btx350.

[246] Mylonas SK, Axenopoulos A, Daras P. DeepSurf: a surface-based deep learning approach for the prediction of ligand binding sites on proteins. Bioinformatics. 2021 01;37(12):1681-90. Available from: https://doi.org/10.1093/bioinformatics/btab009.

[247] Yan X, Lu Y, Li Z, Wei Q, Gao X, Wang S, et al. PointSite: A Point Cloud Segmentation Tool for Identification of Protein Ligand Binding Atoms. Journal of Chemical Information and Modeling. 2022 Jun;62(11):2835-45. Available from: https://doi.org/10.1021/acs.jcim.1c01512.

[248] Charles RQ, Su H, Kaichun M, Guibas LJ. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 77-85.

[249] Kokh DB, Richter S, Henrich S, Czodrowski P, Rippmann F, Wade RC. TRAPP: A Tool for Analysis of Transient Binding Pockets in Proteins. Journal of Chemical Information and Modeling. 2013 May;53(5):1235-52. Available from: https://doi.org/10.1021/ci4000294.

[250] Cimermancic P, Weinkam P, Rettenmaier TJ, Bichmann L, Keedy DA, Woldeyes RA, et al. CryptoSite: Expanding the Druggable Proteome by Characterization and Prediction of Cryptic Binding Sites. J Mol Biol. 2016 Feb;428(4):709-19.

[251] Cuchillo R, Pinto-Gil K, Michel J. A Collective Variable for the Rapid Exploration of Protein Druggability. Journal of Chemical Theory and Computation. 2015 Mar;11(3):1292-307. Available from: https://doi.org/10.1021/ct501072t.

[252] Decherchi S, Rocchia W. A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale. PLoS One. 2013 Apr;8(4):e59744.

Ringraziamenti

Ringrazio il Prof. Andrea Cavalli e il Dott. Sergio Decherchi per avermi accolto nel loro gruppo di ricerca, per avermi guidato e supervisionato durante questi lunghi anni. Ringrazio anche il Prof. Filippo Maria Bianchi per avermi accolto all'UiT e per avermi dato la possibilità di lavorare in un ambiente stimolante e produttivo.

Sergio, al di là della supervisione e del lavoro, hai osservato tutto il percorso dall'inizio alla fine e sai quanto sia stato difficile e provante per me. Quando ho visto solo nero hai capito i miei bisogni e hai empatizzato con le mie difficoltà, dandomi spazio e tempo per riprendermi. Il mondo accademico è un mondo difficile, e sono sicuro che non sarei arrivato in fondo senza questa libertà. Per questo, ti sono profondamente grato.

Filippo, hai reso il freddo norvegese molto meno freddo, e lavorare con te mi ha dato la possibilità di capire il mio valore e fare pace con la ricerca. Grazie per aver creduto in me quando io ho smesso di farlo.

Ele, questi anni non sarebbero stati gli stessi senza di te. Mentirei se dicessi che sono pronto a lavorare in un ufficio in cui non ci sei. Sei riuscita a starmi vicino anche quando ero lontano e non so nemmeno da dove partire per ringraziarti. Mi sento fortunato di poterti chiamare amica.

Stefano, per sempre mio brother-in-science, e tutti i "colleghi" di rn.ai, non cambiate mai. Le persone come voi sono una boccata d'aria fresca.

Sami, grazie per avermi insegnato l'assertività :P

Spostandoci fuori dall'accademia, un grazie sentito va a tutti i compagni di bevute, di tennis, di ping pong, di chiacchiere e risate conosciuti a Genova in questi anni. In particolare Eleonora, che anche se ormai a 10 fusi orari di distanza è sempre vicina come il primo giorno. Mi avete alleggerito le giornate anche quando sembrava impossibile.

Ale, mi hai conosciuto forse nel momento peggiore di tutti. Nonostante questo, per qualche strano allineamento di pianeti hai deciso di camminare con me. Assieme è tutto più colorato.

Mamma, papà, zia, zio, Rò, Giorgio, Alessio, Fabio, Chiara, Ilaria, Teresa, Salvatore. Mi avete visto perdere pezzi, mi avete visto rimetterli insieme, mi avete visto cadere e rialzarmi cambiato, mi avete visto allontanarmi quasi al punto di sparire, eppure siete sempre stati là per me.

A voi tutti, grazie.

Infine, non l'ho mai fatto ma penso che stavolta sia necessario e importante.

Grazie a me stesso.

:wq