Transferable Representation Learning for Drug Discovery

Peizhen Bai

School of Computer Science

University of Sheffield

This thesis is submitted for the degree of

Doctor of Philosophy

June 2025

Declaration

All sentences or passages quoted in this thesis from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this thesis have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in the degree examination as a whole.

Name: Peizhen Bai

Signature: Peizhen Bai

Date: 23 June 2025

Acknowledgements

First and foremost, I would like to express my special gratitude to my MSc and PhD supervisor Prof. Haiping Lu, for his unwavering support, professional guidance, and consistent encouragement throughout my doctoral journey. His insightful advice, vision, and dedication have been invaluable in shaping my development and growth as an academic researcher. Equally, I am pretty grateful to Dr. Filip Miljković, Dr. Bino John, Dr. Xianyuan Liu and Prof. Val Gillet for their crucial advice, guidance and strong research collaborations. Furthermore, I sincerely appreciate the financial and scholarship support provided by the University of Sheffield and the School of Computer Science.

I want to extend my sincere thanks to my labmates and friends in the Machine Learning Group at Lab 136, Regent Court: Dr. Chunchao Ma, Dr. Yan Ge, Dr. Shuo Zhou, Dr. Chao Han, Dr. Lawrence Schöbs, Sina Tabakhi, Pawel Pukowski, Haolin Wang, Jiayang Zhang, Wenrui Fan, Mohammod Naimul Islam Suvon, Alan Thomas and Lalu Muhammad Riza Rizky, who provided me with constant help and made the lab a place of daily joy throughout my PhD journey.

Finally, I extend my deepest gratitude to my family—my wife, Jia, and my parents—for their unwavering love and support. I am especially grateful to my wife for her enduring companionship and unconditional trust in me. You have brought immense joy and brightness to my life. Meeting you and sharing our life journey together is my greatest fortune.

I am grateful to everyone who supported, encouraged, and helped me throughout this journey. This achievement is as much yours as it is mine. Thank you.

Abstract

Drug discovery seeks to identify new candidate medications that can effectively treat human diseases with acceptable developability. Traditional computational and machine learning methods leverage handcrafted domain features for drug screening but suffer from poor transferability due to the vast chemical search space. In this thesis, we introduce transferable representation learning, a prominent approach within deep learning, to address different domain transferability challenges in drug discovery. Specifically, we develop three deep learning-based frameworks to learn transferable representations that adapt to key drug-related tasks, improving specific transferability for drug-target interaction prediction and enhancing generic transferability for molecular property prediction and inverse protein folding.

For drug-target interaction prediction, we first propose a low-bias evaluation strategy to effectively validate specific transferability. After that, we develop a bilinear attention network-based framework incorporating domain adaptation to improve performance under both in-domain and cross-domain settings. Furthermore, we design a molecular self-supervised pre-training framework aimed at improving generic transferability for molecular property prediction. The pre-trained model fully captures 2D topological and 3D geometric information of molecules, enabling fine-tuning for different downstream property prediction tasks. Finally, we design a mask prior-guided denoising diffusion framework that improves generic transferability for inverse protein folding, which involves iteratively generating feasible amino acid sequences that can fold into a given protein structure. In this thesis, extensive experiments are conducted to demonstrate the effectiveness of our proposed frameworks compared to related state-of-the-art methods. We also identify potential research directions in this emerging field for future exploration.

Table of Contents

List of Figures

List of Tables

Symbols and Notations

General Symbols

a	A real or integer scalar
$\mathbf{a}$	A vector
$\mathbf{A}$	A matrix
$\mathcal{A}$	A tensor
$\mathbb{A}$	A set
$\mathbb{R}$	The set of all real numbers
$\mathbb{P}$	Probability
$\mathbb{E}$	Expectation
$\mathbf{I}$	An identity matrix

Indexing

$\mathbf{a}_i$	The i-th element of a vector $\mathbf{a}$
$\mathbf{A}_{i,j}$	The element at the i-th row and j-th column of a matrix $\mathbf{A}$
$\mathbf{A}_{i,:}$	The i-th row of a matrix $\mathbf{A}$
$\mathbf{A}_{:,j}$	The j-th column of a matrix $\mathbf{A}$
$\mathcal{A}_{i,j,k}$	The element at the i-th, j-th and k-th position of a 3D tensor $\mathcal{A}$
$\mathcal{A}_{:,j,:}$	A 2D slice indexed by j at the second dimension of a 3D tensor $\mathcal{A}$

Graphs, Sets and Intervals

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	A graph consisting of the vertices and edges
$\{0, 1\}$	The set containing only 0 and 1
$\{0, 1, \ldots, n\}$	The set containing all integers between 0 and n
$[a, b]$	The real interval including a and b
$[a, b)$	The real interval including a but excluding b

Functions and Operations

$\sigma(\cdot)$	Non-linear activation function
$\otimes$	Outer product
$\circ$	Hadamard (element-wise) product
$\cdot$	Dot product
$\sum$	The sum of a sequence of terms
$\prod$	The multiplication of a sequence of terms
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\|\|\mathbf{A}\|\|^F$	Frobenius norm of matrix $\mathbf{A}$
$\mathcal{L}$	Loss function
$\|\mathbb{A}\|$	The size of set $\mathbb{A}$
$\text{sim}(\cdot, \cdot)$	Dot similarity between two vectors
$\frown$	Concatenation operator over multiple scalars
$\text{SumPool}(\cdot)$	Mean pooling operation
$\text{MeanPool}(\cdot)$	Mean pooling operation
$\text{Cat}(\mathbf{x}; \mathbf{p})$	Categorical distribution over $\mathbf{x}_t$ with probabilities $\mathbf{p}$

Abbreviations

AA	Amino Acid
BAN	Blinear Attention Network
BLOSUM	Blocks Substitution Matrix
CDAN	Conditional Domain Adversarial Network
CNN	Convolutional Neural Network
CV	Computer Vision
DB	Davies Bouldin
DDIM	Denoising Diffusion Implicit Model
DDPM	Denoising Diffusion Probabilistic Model
DL	Deep Learning
DNN	Deep Neural Network
DTI	Drug-Target Interaction
EGNN	Equivariant Graph Neural Network
GCN	Graph Convolutional Network
GIN	Graph Isomorphism Networks
GNN	Graph Neural Network
HDBSCAN	Hierarchical Density-Based Spatial Clustering
IPF	Inverse Protein Folding
MLP	Multi-Layer Perceptron
MRD	Mutual Reachability Distance
MST	Minimum Spanning Tree
NLP	Natural Language Processing

NSSR	Native Sequence Similarity Recovery
RBF	Radial Basis Function
SSL	Self-Supervised Learning
SVM	Support Vector Machine
TRL	Transferable Representation Learning
VLB	Variational Lower Bound

Chapter 1

Introduction

The goal of drug discovery is to identify safe and effective compounds for the treatment of human diseases. Owing to the enormous chemical space (~10^{60} active molecules) (Kirkpatrick and Ellis, 2004) and complex screening process (Kim et al., 2021), traditional drug discovery has notably high cost and time-consuming development cycle. The average cost of a new medicine is estimated at 2.6 billion USD, and the whole development cycle takes over 12 years (Chan et al., 2019). To decrease the cost and failure rate of *in vitro* experiments, many computational methods, such as quantitative structure-activity relationship (QSAR), were utilized to accelerate the identification of high-confidence drug candidates (Sliwoski et al., 2014). However, these methods typically rely on a set of handcrafted features that require extensive and expert domain knowledge. If the predefined rules lack the right representative information for the specific task, computational methods will have poor generalization. In contrast, deep learning (DL)-based methods can adaptively learn data-driven representation from input data and flexible objectives, allowing to greatly enhance model performance on drug discovery-related tasks.

With the accumulation of large amounts of biological and chemical data over the last few decades, considerable effort has been invested to automating and accelerating drug screening and design through DL. These techniques now play a vital role across multiple stages of the drug discovery pipeline. For instance, DL-based methods can predict interactions between drug molecules and target proteins (Lee et al., 2019; Nguyen et al., 2021; Öztürk et al.,

2018; Huang et al., 2021), enabling biologists and chemists to prioritize the most promising candidates and identify potential opportunities for drug repurposing at an early stage. By representing molecules as graphs, researchers have developed deep models for molecular property prediction using graph representation learning (You et al., 2020; Subramonian, 2021; Stärk et al., 2022; Liu et al., 2022). This enables the prediction of key properties such as solubility, toxicity and stability, reducing reliance on time-consuming wet-lab assays. Furthermore, deep generative models are increasingly developed for de novo design tasks, such as generating novel proteins or small molecules with desired properties (Jumper et al., 2021; Peng et al., 2022). They are also used for inverse protein folding, which designs feasible amino acid sequences with specific protein structures and functions (Yi et al., 2024; Zheng et al., 2023). Consequently, the emergence of DL techniques has progressively transformed the early stage of drug discovery, leading to the exploration of more efficient, precise and cost-effective methodologies in relevant applications.

1.1 Motivation and Research Questions

Despite the great success and breakthroughs of DL across multiple fields, most mainstream DL models are considered to be *data-hungry*, which require a huge amount of training data to achieve satisfactory and generalizable performance. However, the cost of data annotation and collection is quite expensive in drug discovery-related tasks. Although there has been accumulated considerable big data over many years of chemical research, the data volume and efficiency are still inadequate in the face of enormous magnitude of chemical space. Therefore, the learned deep representations always lack transferability to other relevant tasks, or the same task but with different distributed data (Bengio, 2012).

In the process of real-world drug design and screening, it is crucial to consider transferability in modelling as the drug molecules to be predicted are often novel and out of learned distribution. In this thesis, we aim to explore model transferability in drug discovery and develop new DL-based methods. The proposed methods can learn transferable representations that adapt to multiple prediction tasks, and enhance generalization to out-of-distribution data.

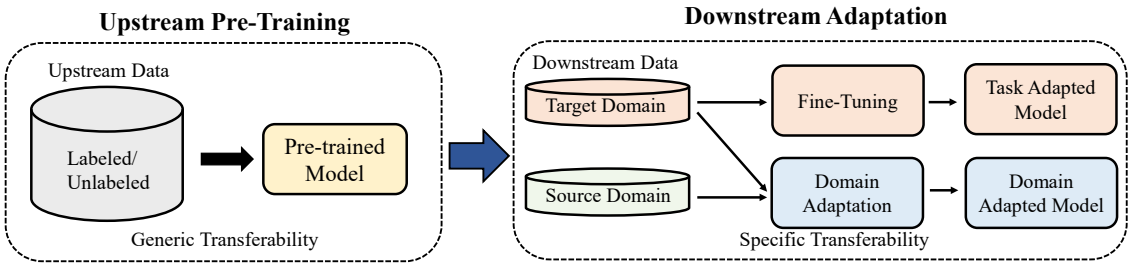

Fig. 1.1 The two-stage TRL lifecycle in most transferability research of DL. In upstream pre-training, models mainly focus on learning generic transferability from large-scale labeled/unlabeled data. In downstream adaptation, the pre-trained models can be directly fine-tuned to adapt to the target task if labeled data in the target domain is available. If there is only unlabeled data in the target domain, additional labeled data from the source domain will be introduced to improve target performance.

Towards gaining transferability in DL, as shown in Figure 1.1, the whole lifecycle of transferable representation learning (TRL) can be divided into two stages: upstream pre-training and downstream adaptation (Jiang et al., 2022). Upstream pre-training utilizes large-scale labeled/unlabeled data to learn general representations or reusable weights that can be transferable to diverse downstream tasks. The goal of this stage is to enhance *generic* transferability of models. Pre-training strategies have been widely studied in many fields, including computer vision (CV) (Liu et al., 2021) and natural language processing (NLP) (Devlin et al., 2019). They have gradually become a paradigm in large-scale industrial applications, such as foundational models, owing to their decent performance.

Moreover, the stage of downstream adaptation aims to enhance *specific* transferability to a target task using downstream data. When labeled data is available from the target domain, pre-trained models can be directly fine-tuned to adapt to the target task. However, if only unlabeled data is available in the target domain, additional labeled data from the identical prediction task but with a different distribution (i.e. source domain data) can be employed to enhance target performance. This process is known as domain adaptation (Ganin and Lempitsky, 2014), which is a significant component in TRL. As a result, the majority of DL

transferability research falls within a specific stage of the TRL lifecycle. Next, I will describe my TRL research questions in drug discovery and give a brief background about them.

- Many DL-based methods have significantly achieved high accuracies in the field of drug discovery. However, an obvious performance gap still remains between academic research and industrial applications, where academic results tend to be over-optimistic for industrial settings. An essential reason is that the general evaluation framework for DL assumes that test data follows the same distribution as training data, which rewards more memorization rather than transferability. To effectively employ the models in real-world drug discovery, it is crucial to demonstrate their generalization that can transfer learned knowledge to novel drugs. Therefore, the first research question (Q1) is: *how can we design a low-bias evaluation to fairly measure model transferability and reduce the gap with real performance in drug discovery?*

- In the downstream adaptation of TRL, current deep models perform weak specific transferability, leading to a significant performance decrease when faced with biological test data that falls outside the learned distribution. Different from the fields of CV and NLP, drug discovery typically lacks sufficient labeled data to learn the full distribution for the target domain. Consequently, the weak specific transferability heavily impedes the application of deep models in wider scenarios. The second research question (Q2) is: *how can we enhance specific transferability of deep models to novel drugs that are out of learned distribution?*

- In the upstream pre-training of TRL, a well-trained model learns transferable representations and reusable parameters with generic transferability. This enables the model to effectively adapt to various discriminative and generative tasks, such as molecular property prediction and inverse protein folding. In drug discovery, the number of labeled data is often far from sufficient to obtain a pre-trained model with enough generic transferability. The third research question (Q3) is: *how can we design the self-supervised learning frameworks that can enhance generic transferability in both drug-related discriminative and generative tasks?*

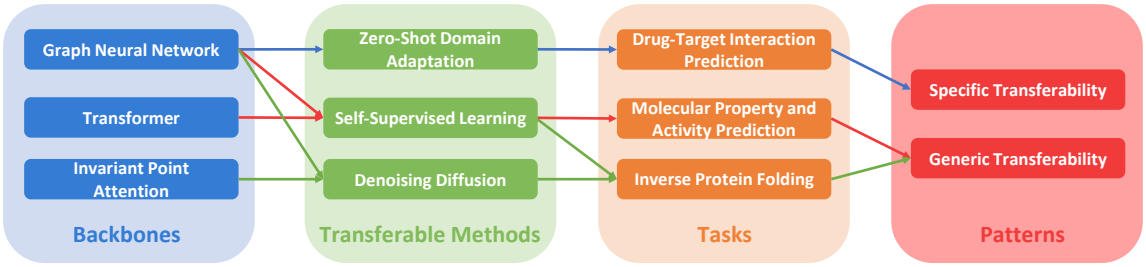

Fig. 1.2 The structured relationships of different components in this thesis. The main focus is exploring transferable representation learning in drug discovery, which is applied to relevant tasks and transferability patterns.

1.2 Structure and Contributions

This section presents the structure of this thesis and highlights the novel research contributions. Figure 1.2 illustrates the structured relationships between TRL and drug discovery tasks in this thesis. Chapter 2 presents a survey of the literature related to the research topics. In Chapter 3, we propose a low-bias evaluation method to measure the model generalization for drug-target interaction (DTI), specifically targeting Q1. Chapter 4 answers Q2 by presenting a new deep model, the bilinear attention network with domain adaptation, to comprehensively improve DTI prediction. After that, Chapters 5-6 utilize the newly developed pre-training strategies to enhance the generic transferability of Q3 in molecular property prediction and protein design, respectively. Chapter 7 summarises the findings of this thesis and presents the potential future work. The main contents of the following chapters are listed below.

Chapter 2: Background presents a survey of the literature related to the research topics in this thesis. Since molecules can be naturally viewed as 2D topology graphs, where each node represents an atom and each edge denotes a chemical bond. Graph representations can effectively encode structural information in molecules and play a crucial role in the development of our proposed methods. This chapter first introduces the background knowledge about deep graph learning, including Graph Neural Networks (GNNs) and Graph Transformer. Then, it provides a fundamental review of TRL, which can be divided into self-supervised pre-training and domain adaptation. Additionally, this chapter shows a key generative method: denoising diffusion probabilistic models (DDPMs) that will be used in

Chapter 6. Finally, three important tasks in drug discovery: drug-target interaction, molecular property prediction and molecule generation, are introduced as the focus of this thesis.

Chapter 3: Toward Low-Bias Evaluation for Drug-Target Interaction identifies the common pitfalls that lead to high-bias performance evaluation in drug-target interaction (DTI) prediction. It then presents two hierarchical-clustering-based split strategies to achieve low-bias evaluation in realistic settings. We specifically study the data bias in a widely used DTI dataset, BindingDB, and re-evaluate the prediction performance of three state-of-the-art deep models using five data split strategies. The experimental results confirm the overoptimism of the general random and cold splits. Meanwhile, this study reveals that hierarchical-clustering-based splits are far more challenging and can provide a potentially more valuable evaluation of model transferability in real-world DTI prediction scenarios.

Contribution 1: We empirically validate the general evaluation bias for deep models in DTI prediction, and propose two hierarchical-clustering-based splits to solve Q1. The new split strategies lead to more challenging tasks that can better reward model transferability (or generalization) rather than memorization. Additionally, we point out the common pitfalls in the construction of the DTI dataset, which can serve as guidelines for mitigating data bias. The low-bias evaluation framework provides an effective transferability comparison of deep models in realistic settings.

Chapter 4: Bilinear Attention Network for Drug-Target Interaction Prediction presents an interpretable bilinear attention network-based model (DrugBAN) to improve DTI prediction. It works on 2D drug molecular graphs and 1D target protein sequences to perform prediction. BAN is employed to learn local pairwise interactions between drugs and targets, while conditional domain adversarial learning is introduced to align the learned representations across different distributions for better generalization on novel drug-target pairs. Experiments on three benchmark datasets under both in-domain and cross-domain settings show that DrugBAN achieves the best overall performance against other state-of-the-art models. Moreover, visualizing the learned bilinear attention map provides interpretable insights from prediction results.

Contribution 2: We propose DrugBAN, an end-to-end bilinear attention deep model to address Q2. To enhance the specific transferability to out-of-distribution data, we have integrated an adversarial domain adaptation module into the cross-domain modelling. Moreover, we create a quantitative cross-domain scenario by utilizing a clustering-based split strategy. Distinct from previous methods, DrugBAN can leverage its bilinear attention map to capture pairwise local interactions between drugs and targets, resulting in more interpretable insights for prediction results and improved generalization capability.

Chapter 5: Geometry-aware Line Graph Transformer Pre-training proposes to apply Graph Transformer to develop a novel molecular self-supervised pre-training framework for molecular representation learning, and adapt to various downstream molecular property prediction tasks. We first design a geometry-aware line graph transformer (Galformer) backbone to adaptively encode the 2D topological and 3D geometric line graphs of a molecule. The designed backbone has a high capacity and can capture critical structural information from both 2D and 3D modalities. Next, we devise two complementary pre-training tasks, that is dual-view contrastive learning and masked line node prediction, to achieve comprehensive understanding at both inter and intra-modality levels. To evaluate the model performance of Galformer, we compare it against six state-of-the-art baselines on twelve property prediction benchmarks, using downstream fine-tuning. Experimental results show that Galformer consistently outperforms all baselines on both classification and regression tasks, demonstrating its effectiveness.

Contribution 3: We propose a novel molecular self-supervised pre-training framework, Galformer, to address Q3 in the context of molecular property prediction. Galformer leverages a unified graph transformer architecture to encode the topological and geometric information of a molecule simultaneously, which fully captures structural information from different modalities. Furthermore, we build two complementary pre-training tasks from both inter- and intra-modality levels as self-supervised objectives, allowing the model to understand generic domain knowledge in the learning process.

Chapter 6: Mask Prior-Guided Denoising Diffusion Improves Inverse Protein Folding presents a mask prior-guided denoising diffusion (MapDiff) framework that accurately captures structural and residue information for inverse protein folding. MapDiff is a discrete diffusion probabilistic model that iteratively generates amino acid sequences with reduced noise, conditioned on a given protein backbone. Additionally, we develop a two-step denoising network incorporating residue interactions through a mask prior pre-training strategy. For sampling inference, we combine the denoising diffusion implicity model with Monte-Carlo dropout to improve the generative process. Experiments on four sequence design benchmarks demonstrate that MapDiff significantly outperforms existing methods, achieving state-of-the-art performance. Furthermore, the protein sequences generated by our method exhibit biological relevance and a high degree of similarity in folding into structures closely resembling native proteins.

Contribution 4: We propose MapDiff, a novel discrete denoising diffusion-based framework to solve Q3 in the context of inverse protein folding. This framework incorporates a two-step denoising network that adaptively conditions the diffusion trajectories to produce feasible amino acid sequences from a fixed protein structure. Particularly, we design a masked sequence designer that explicitly considers residue interactions through a mask prior pre-training strategy. Furthermore, MapDiff leverages denoising diffusion implicity model with Monte-Carlo dropout to accelerate the generative process and improve uncertainty estimation.

Chapter 7: Conclusion and Future Work summarises the findings and contributions involved in this thesis. We then point out the potential directions for future research in the area, expanding upon the work presented in the prior chapters.

The contents of this thesis are based on the following three publications and one manuscript on arXiv, for which I am the leading author. The technical contents of Chapter 4 and Chapter 6 have appeared in Nature copyrighted materials, with the permission to preprint version granted by Nature.

1. Peizhen Bai, Filip Miljković, Yan Ge, Nigel Greene, Bino John, and Haiping Lu. "Hierarchical clustering split for low-bias evaluation of drug-target interaction prediction." *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 641-644, 2021.

2. Peizhen Bai, Filip Miljković, Bino John, and Haiping Lu. "Interpretable bilinear attention network with domain adaptation improves drug–target prediction." *Nature Machine Intelligence*, 5, 126-136, 2023.

3. Peizhen Bai, Xianyuan Liu, and Haiping Lu. "Geometry-aware Line Graph Transformer Pre-training for Molecular Property Prediction." arXiv preprint: 2309.00483, 2023.

4. Peizhen Bai, Filip Miljković, Xianyuan Liu, Leonardo De Maria, Rebecca Croasdale-Wood, Owen Rackham, and Haiping Lu. "Mask prior-guided denoising diffusion improves inverse protein folding." *Nature Machine Intelligence*, 1-13, 2025.

The following three papers were also co-authored during the period of research, however do not align with the main narrative of this thesis and therefore have not been included in the text.

1. Lu, Haiping, Xianyuan Liu, Shuo Zhou, Robert Turner, Peizhen Bai, Raivo E. Koot, Mustafa Chasmai, Lawrence Schobs, and Hao Xu. "PyKale: Knowledge-aware machine learning from multiple sources in Python." *ACM International Conference on Information & Knowledge Management (CIKM)*, 4274-4278. 2022.

2. Xu, Hao, Shengqi Sang, Peizhen Bai, Ruike Li, Laurence Yang, and Haiping Lu. "GripNet: Graph information propagation on supergraph for heterogeneous graphs." *Pattern Recognition*, 133, 108973, 2023.

3. Mu, Yida, Peizhen Bai, Kalina Bontcheva, and Xingyi Song. "Addressing Topic Granularity and Hallucination in Large Language Models for Topic Modelling." arXiv preprint:2405.00611, 2024.

Chapter 2

Background

In Chapter 1, we have introduced the importance of transferable representation learning (TRL) and presented a basic research framework for drug discovery. In this chapter, we will further provide a more comprehensive background on our research topics. Firstly, we start with the introduction of TRL methods based on different techniques and gain insights into their applications to drug discovery. After that, we discuss classical graph-based models in deep graph learning, which have shown great influence in modeling molecular data. Then, we introduce a promising class of deep generative methods called denoising diffusion probabilistic models. Finally, this chapter presents the related work on three application-specific tasks: drug-target interaction prediction, molecular property prediction, and 3D molecule generation.

2.1 Transferable Representation Learning

Although deep learning models are widely used across various fields due to their powerful performance, they typically require large-scale labeled data during training, and thus can generalize well to test data. However, in drug discovery, there are many specific tasks for which only unlabeled or limited labeled data is available. The nature of data scarcity restricts the transferability of deep learning models to unseen data that is out of the learned distribution. In this condition, pre-training and domain adaptation have become classic paradigms for

tackling problems with limited data and annotations. These methods can learn transferable representations from a relevant upstream task or the same downstream task but with different distributed data. In this section, we will discuss important pre-training and domain adaptation approaches as background for our study.

2.1.1 Introduction to Pre-training

Pre-training aims to train deep learning models using large-scale upstream data, and transfer the learned knowledge to specific downstream tasks to improve performance. In recent years, it has been widely applied to enhance the generic transferability of large and foundational models (Radford et al., 2018; Devlin et al., 2019; Ouyang et al., 2022). Generally, the pre-training paradigm is considered to be effective as it enables implicitly learning inductive bias (Torrey et al., 2010) and reducing intrinsic dimensions (Aghajanyan et al., 2021) from large amounts of data.

Inductive bias refers to the set of assumptions that a machine learning algorithm makes to generalize unseen data. For example, in supervised learning, convolutional neural networks utilize the local connectivity assumption as its strong inductive bias for image recognition. When training data is scarce, a fixed but strong inductive bias can enhance data efficiency and improve model generalization. However, it will also limit the model's hypothesis space, potentially reducing its transferability and expressiveness to specific tasks (Battaglia et al., 2018). During pre-training, important inductive bias can be acquired from large-scale data itself rather than manual definition. This data-driven approach ensures that the learned inductive bias is more transferable and representative of the underlying data distribution. As the most commonly used architecture for pre-training, Transformer (Vaswani et al., 2017) has the least assumptions on the input data structure, thus providing the most expressive framework for adaptively learning transferable knowledge and inductive bias.

Intrinsic dimension describes the minimum number of parameters required to solve the optimization problem without significant loss in performance (Li et al., 2018). A pre-trained model typically contains millions of learnable parameters, but it can generalize well to downstream tasks using only a few hundred or thousand labeled samples through fine-tuning

(Devlin et al., 2019) or low-rank adaptation (Hu et al., 2022). The phenomenon demonstrates that pre-training can effectively reduce the intrinsic dimension of the model. Aghajanyan et al. (2021) provides empirical evidence that larger models can have lower intrinsic dimension after a certain number of pre-training iterations, leading to efficient model compression and capturing the essential features of the data. Therefore, pre-training can achieve rapid adaptation and enhance generalization on new tasks by leveraging the low intrinsic dimension nature.

Pre-training methods can be classified into supervised pre-training and self-supervised (unsupervised) pre-training, depending on the use of labeled or unlabeled training data. Due to the scarcity and high annotation cost for the labeled data in the field of drug discovery, this thesis primarily focuses on the development of self-supervised pre-training. Next, we will introduce two classic self-supervised learning frameworks based on the types of specifically designed pre-training tasks: contrastive self-supervised pre-training and generative self-supervised pre-training (Jiang et al., 2022).

2.1.2 Contrastive Self-supervised Pre-training

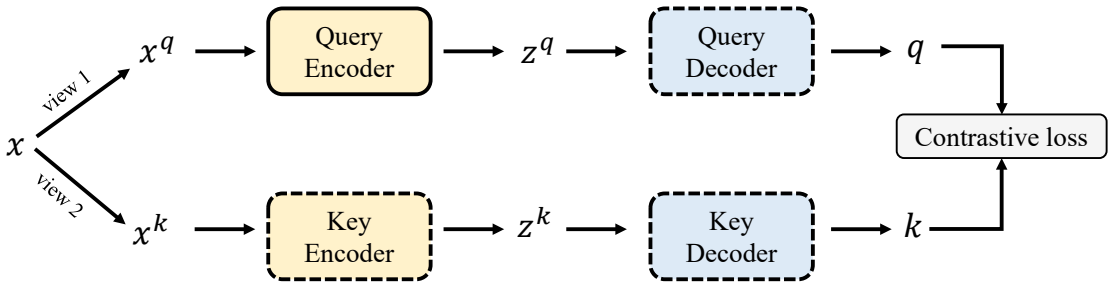

Fig. 2.1 A general pipeline of contrastive learning. Two augmented views x^q and x^k are generated from the original input x. x^q serves as an anchor, while x^k acts as its positive sample. The two views x^q and x^k are first encoded to their latent representations $\mathbf{z}^q$ and $\mathbf{z}^k$, respectively. Then the decoders map the latent representations into metric space to calculate the contrastive loss. The model is optimized by minimizing the contrastive loss.

Contrastive learning is a self-supervised pre-training approach that involves discriminative tasks. It applies an instance-based comparison to group similar samples closer together and pushes diverse samples farther apart (Jaiswal et al., 2020), as shown in Figure 2.1. For each input data x, contrastive learning first creates two augmented views x^q and x^k using domain-specific augmentation strategies, such as random cropping or color transformation for image data. In pre-training, one of the augmented view x^q acts as an anchor, while another view x^k serves as its positive sample. All other samples in the training set are considered as negative samples. The encoders embed and map different views into their latent representations, and the decoders further map the latent representations into common metric space to calculate specific contrastive loss, such as InforNCE (Oord et al., 2018) or NT-Xent (Chen et al., 2020b). Despite having different forms, the primary goal of contrastive losses is to maximize the mutual information between the anchor and positive sample in the metric space. In this way, contrastive learning creates pseudo labels for self-supervised pre-training and enables the model to adaptively capture intrinsic and transferable knowledge from data.

There are two key components in contrastive learning: augmentation strategy and contrastive framework. The design of the augmentation strategy is highly flexible and relies on specific types of data. An effective augmentation strategy should have two basic properties: (1) Generating sufficient diverse variations to enhance model generalization across different scenarios. (2) Preserving semantic invariance to ensure that positive pairs maintain their essential features even after augmentation. For image data, many geometric transformations, such as flipping, random cropping and scaling, are often used as augmentation strategies to generate anchors and positive samples (Jaiswal et al., 2020). Such methods encourage generative diversity while keeping semantic invariance in original images. For text data, two consecutive sentences from the same document can be regarded as natural positive pairs. Lan et al. (2020) utilize the textual nature to construct sentence-order prediction tasks as contrastive self-supervised pre-training. In the field of drug discovery, MolCLR (Wang et al., 2021b) designs three molecular data augmentation strategies (atom masking, bond deletion and subgraph removal) to develop 2D molecular contrastive learning, improving the model performance on various downstream tasks for molecular property prediction.

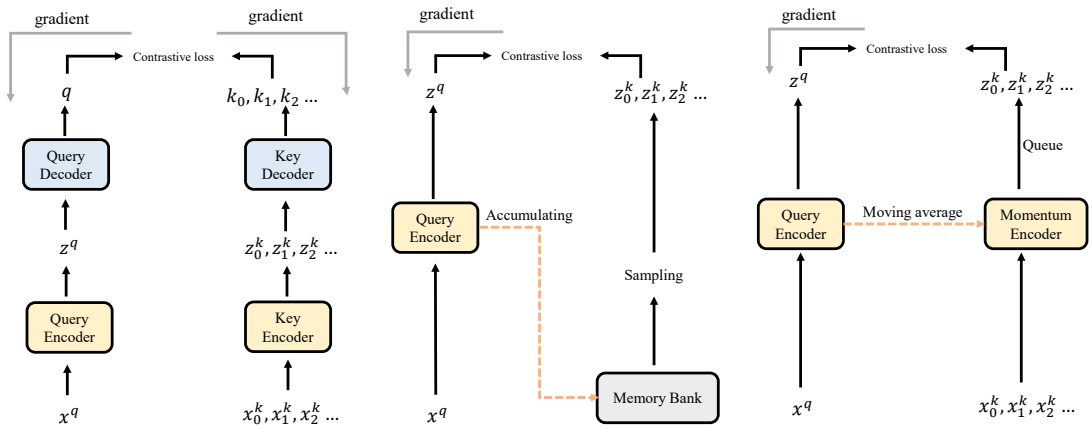

Fig. 2.2 Three classic contrastive frameworks. (a) SimCLR generates query and key representations separately with a large batch size. (b) Memory Bank records various key representations in a look-up dictionary and the negative data is sampled from previous recorded representations at each batch training. (c) MoCo maintains a large and consistent dictionary using a memory bank and momentum update mechanisms. The momentum encoder is iteratively updated from the query encoder with a small momentum coefficient.

The second key component is the contrastive framework used to propagate information. Figure 2.2 illustrates three classic contrastive frameworks: SimCLR (Chen et al., 2020b), Memory Bank (Wu et al., 2018b) and MoCo (He et al., 2020), which are primarily different from the strategies for negative sample collection. SimCLR proposes a simple end-to-end contrastive framework, in which query and key encoders individually process input data and perform the reverse gradient update. During the training phase, SimCLR usually employs a large batch size to provide more diverse negative samples and make the discriminative task more challenging. However, in certain scenarios, using a large batch size can exceed the memory limit of hardware resources and suffer from optimization issues in regular backpropagation. To solve the issues, Memory Bank introduces a dictionary of negative samples by accumulating the generated negative representations from a series of previous batch iterations. It significantly reduces the memory cost in mini-batch training and enhances the framework scalability in applications. On the basis of Memory Bank, MoCo maintains a

dynamic dictionary of negative samples through a momentum update. It slowly updates the momentum encoder from the query encoder using a small momentum coefficient. Meanwhile, the dynamic dictionary uses a queue mechanism to remove outdated negative representations regularly. As a result, MoCo can relieve the representation inconsistency problem in the dictionary of negative samples and improve memory efficiency with small batch training.

In comparison to supervised pre-training, contrastive self-supervised pre-training can learn more transferable representations with unlabeled data, leading to competitive or even better performance on a range of downstream tasks. To explain this phenomenon, Zhao et al. (2021b) reveals that contrastive self-supervised pre-training usually learns low-level and mid-level representations that can be transferred, while supervised pre-training tends to extract high-level semantic information to specific properties. In drug discovery, researchers aim to comprehensively evaluate various properties of a potential drug molecule, such as toxicity, stability and lipophilicity. If the downstream property prediction is not connected to the labels in supervised pre-training, the pre-trained models may suffer from overfitting, potentially limiting their generic transferability. In contrast, contrastive pre-training methods need to discriminate all instances rather than labels. This makes the model learn more essential and transferable representations at the low and mid-levels, which enhances transferability to various downstream tasks.

2.1.3 Generative Self-supervised Pre-training

Generative learning is a self-supervised pre-training approach that learns to approximate underlying data distribution with generative pre-training tasks. Figure 2.3 illustrates a general pipeline for generative learning. The perturbed sample x^p is first generated from the original input x. Then it employs an encoder to map x^p into a latent representation $\mathbf{z}$, and a decoder further maps $\mathbf{z}$ back to a reconstructed output $\hat{x}$. The objective of generative pre-training tasks is to minimize the reconstructed loss between the original input x and the reconstructed output $\hat{x}$. After pre-training, the encoder and learned latent representations can be adapted to downstream tasks. Based on the types of objective functions, most generative learning

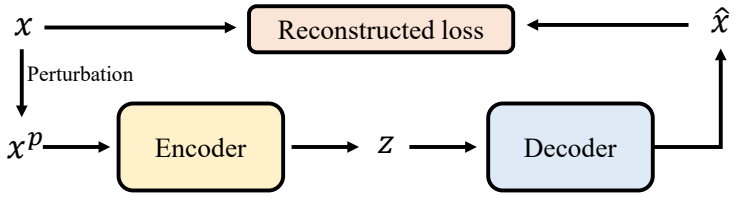

Fig. 2.3 A general pipeline of generative learning. The perturbed sample x^p is generated from the original input x. An encoder is employed to map x^p into a latent representation $\mathbf{z}$, and a decoder further maps $\mathbf{z}$ back to a reconstructed output $\hat{x}$. The objective of generative pre-training tasks is to minimize the reconstructed loss between the original input x and the reconstructed output $\hat{x}$.

methods are classified into two categories: autoregressive pre-training and autoencoding pre-training.

Autoregressive pre-training learns the data distribution by predicting future information on the condition of the previous context. For example, language modeling aims to predict the probability of the next word in a given sequence, depending on the previous contextual words in a fixed window size. The autoregressive mechanism can facilitate contextual understanding and capture rich semantics in the face of vast amounts of data. A well-known autoregressive model is the generative pre-trained transformer (GPT) (Radford et al., 2018; Brown et al., 2020), which combines the transformer architecture and language modeling task for autoregressive pre-training. With the expansion of the training corpus and model size, GPT models have demonstrated impressive transferability across a variety of downstream NLP tasks, particularly in the breakthroughs of generative tasks.

Autoencoding pre-training learns the data distribution by reconstructing the original input from the latent representation of the mask sample. The hypothesis of autoencoding models is that transferable representations should be robust to reconstruct all information from the input with partial corruption. Recently, He et al. (2022) proposes masked autoencoder (MAE) to train a vision transformer with unlabeled image data. For data perturbation, MAE randomly masks over 70% patches of each input image. The model must capture essential and rich information to reconstruct the entire image with a small fraction of input patches. In the field

of drug discovery, Hu et al. (2020) propose autoencoding pre-training strategies for molecular representation learning. By viewing each molecule as a 2D graph, they randomly mask the attributes of atoms or chemical bonds, and then employ graph neural networks (GNNs) to predict them based on local structures. The pre-trained GNNs can capture both structural information and domain-specific knowledge, thereby enhancing model transferability on downstream molecular tasks. Due to the great progress of large-scale generative learning across various domains, there has been an increasing interest in developing relevant pre-training techniques for molecules within the giant chemical space (Rong et al., 2020; Zhang et al., 2021; Yang et al., 2022a).

2.1.4 Domain Adaptation

Self-supervised pre-training improves the generic transferability of deep learning models by utilizing large-scale unlabeled data. The pre-trained models can then be adapted to a target task with a few labeled data from the target domain. Nevertheless, in many real scenarios, the data in the target domain is novel and lacks label information, with only some labeled data from a source domain being available. Due to the distribution gap between the source and target domains, a well-trained model using the labeled source data often performs poorly in predicting the target domain. In this situation, domain adaptation techniques have been proposed to reduce the distribution shift between the source domain and target domain, thus improving the model performance and specific transferability.

Early domain adaptation methods reweight sample importance or find common feature subspace in the shallow regime, using labeled data in the source domain and unlabeled data in the target domain (Sugiyama et al., 2007). In recent years, deep domain adaptation methods apply the adaptation module into deep neural networks to adaptively learn domain-invariant features and reduce the distribution gap between the source and target domains (Gong et al., 2013; Huang et al., 2006). For instance, Long et al. (2018) propose a conditional domain adversarial network (CDAN) that combines adversarial networks with multilinear conditioning to learn transferable representation for domain adaptation. By incorporating

classifier prediction information into adversarial learning, CDAN can effectively align data distributions across different domains.

Domain adaptation has gained more attention for improving specific transferability in drug discovery. Abbasi et al. (2019) combine deep domain adaptation with graph convolutional networks for cross-domain molecule property prediction. For another instance, we develop a bilinear attention network-based framework for drug-target interaction prediction. In addition to focusing on in-domain prediction, we have incorporated deep adversarial domain adaptation into our framework to enhance cross-domain prediction performance. A detailed description and experimental validation will be presented in Chapter 4.

2.2 Deep Graph Learning

Transferable representation learning provides pre-training and domain adaptation frameworks for leveraging information across tasks and domains. However, its direct application to drug discovery requires deep architectures that can capture complex structural relationships. To address this, deep graph learning has emerged as a robust approach to represent and learn from graph-structured data such as molecules and proteins. This section introduces the basics of graphs and explains two main types of architectures: graph neural networks and graph transformers. As a result, these provide the foundation for applying transfer learning in drug discovery.

2.2.1 Fundamentals of Graph

Definition 2.2.1. (Graph). A graph with N nodes can be denoted as $\mathcal{G}$= $(\mathcal{V}, \mathcal{E})$, where each node $v_i \in \mathcal{V}$ and each edge $(v_i, v_j) \in \mathcal{E}$ for $i, j = 1, ..., N$. Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ represent the node feature matrix that contains the attributes of the nodes in D dimensions. The adjacency matrix is represented as $\mathbf{P} \in \{0,1\}^{N \times N}$, indicating the connectivity between nodes. For any pair of nodes v_i and v_j, $P_{ij} = 1$ if there is an edge connecting them, otherwise $P_{ij} = 0$

Graphs, such as social networks, recommendation systems, and knowledge graphs, are ubiquitous and natural data structures that can capture pairwise interactions between a set of

nodes, as defined in Definition 2.2.1. In the field of life science, graphs are also commonly used to describe many real-life complex relationships, ranging from molecular structures to biological systems (Yi et al., 2022). For instance, at the molecular scale, biomolecules can be represented as graphs that capture the structural and spatial relationships between their component atoms and chemical bonds. At the biological level, interactomes are viewed as graphs that capture specific interactions between biomolecular species (e.g. nucleic acids, metabolites or proteins). Among biological networks, protein-protein interaction graphs with biologically meaningful associations are the most commonplace (Gaudelet et al., 2021; Han, 2008). The rapid advancement of deep graph learning has led to an increasing interest in utilizing related graph methods within drug discovery and development.

There are various computational tasks studied on graphs, which can be briefly classified into two categories: node-focused tasks and graph-focused tasks (Ma and Tang, 2021). In node-focused tasks, the entire dataset is generally represented in a single graph, with each node as a data sample. These tasks involve node and link analysis within the graph, such as node classification, link prediction and community clustering. In graph-focused tasks, the dataset consists of a set of observed graphs, with each graph as a data sample. These tasks involve graph analysis and the predictions are made for individual graphs, such as graph classification, graph matching and graph generation (Zhu et al., 2022b). In this thesis, we mainly explore graph-focused tasks within drug discovery, which is the most common task category at the molecular scale. Here, we provide the formal definitions for the two primary graph-focused tasks, i.e., graph classification and graph generation.

Definition 2.2.2. (Graph Classification). Given a set of labeled graphs $\mathcal{D} = \{\mathcal{G}_i, y_i\}_{i=1}^{M}$, where $\mathcal{G}_i$ is a graph and y_i is its corresponding label, the goal of graph classification is to learn a mapping function $\mathcal{F} : \mathcal{G} \rightarrow y$ with the set $\mathcal{D}$. The learned function $\mathcal{F}$ can generalize to predict the labels of new graphs.

Definition 2.2.3. (Graph Generation). Given a set of observed graphs $\mathcal{D} = \{\mathcal{G}_i\}_{i=1}^{M}$, where G_i represents a graph, the goal of graph generation is to learn the distribution $p(\mathcal{D})$ from the set $\mathcal{D}$. The new graphs can be generated by sampling from the learned distribution, i.e., $\mathcal{G}_{new} \sim p(\mathcal{D})$.

In the two definitions above, we do not consider potential contexts or conditions associated with the graphs to ensure clarity. In specific scenarios, a graph can be associated with additional information that is utilized for graph classification or conditional generation with desired properties.

2.2.2 Graph Neural Networks

Graph neural networks (GNNs) are a class of deep learning methods that operate deep neural networks on graph-structured data. The first GNN model was introduced around two decades ago (Gori et al., 2005), but this area has received considerable attention in recent years. Modern GNNs can be treated as a process of graph representation learning. They usually follow a neural message-passing (or neighbour aggregation) framework and leverage both node features and graph structure in the learning process (Gilmer et al., 2017). A typical GNN iteratively updates each node representation by aggregating and transforming node-wise information from its local neighbourhood. By stacking multiple GNN layers, the updated node representations can capture rich semantic and structural information within their multi-hop neighbourhoods. Formally, the l-th GNN layer can be defined as:

$$\mathbf{n}_v^{(l)} = \text{Aggregate}\left(\left\{\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v)\right\}\right), \tag{2.1}$$

$$\mathbf{h}_v^{(l)} = \text{Combine}\left(\mathbf{h}_u^{(l-1)}, \mathbf{n}_v^{(l)}\right), \tag{2.2}$$

where $\mathbf{h}_v^{(l)} \in \mathbb{R}^H$ denotes the hidden representation of node v at the k-th layer, and $\mathcal{N}(v)$ is a set of neighbour nodes adjacent to v. We initialize the node representation $\mathbf{h}_v^{(0)}$ by its input node feature $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. The choice of Aggregate$(\cdot)$ and Combine$(\cdot)$ plays a critical role in the effectiveness of GNNs, and many architectures have been proposed (Kipf and Welling, 2017; Veličković et al., 2018; Xu et al., 2019; Hamilton et al., 2017) with efficient Aggregate$(\cdot)$ functions.

For node-focused tasks, the node representation $\mathbf{h}_v^{(L)}$ at the final layer can be utilized for prediction. Specifically, we can extract the individual node representation or concatenate any pair of node representations $[\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)}]$ with a simple classifier to perform node

classification or link prediction. For graph-focused tasks, it is common to incorporate a READOUT$(\cdot)$ function that can aggregate all final node representations to generate the entire graph representation $\mathbf{h}_{\mathcal{G}}$:

$$\mathbf{h}_{\mathcal{G}} = \text{Readout}\left(\left\{\mathbf{h}_v^{(L)} | v \in \mathcal{G}\right\}\right), \tag{2.3}$$

where Readout$(\cdot)$ can be a simple pooling function that satisfies permutation invariant. Next, we will introduce some important GNN architectures that have been proposed in recent years.

Graph Convolutional Networks. Convolutional neural networks (CNNs) have shown powerful expressive ability and led to many breakthroughs in the field of computer vision. The key point of CNNs is the use of shared and learnable convolutional kernels to detect local patterns over the input data. However, the traditional convolution operation can only be applied to data on regular grids. Inspired by this, Kipf and Welling (2017) first proposed graph convolutional networks (GCN) that generalize the convolution operation on irregular graph-structured data. Specifically, the receptive field for each node is defined as its one-hop neighbourhood within a single GCN layer. In the AGGREGATE$(\cdot)$ function, GCN applies a weighted mean aggregation scheme that utilizes the normalized node degree for assigning appropriate weights to adjacency nodes:

$$\mathbf{h}_v^{(l)} = \text{ReLU}\left(W^l \cdot \sum_{u \in \widetilde{\mathcal{N}}(v)} (\text{Deg}(v)\text{Deg}(u))^{-\frac{1}{2}} \mathbf{h}_u^{(l-1)}\right), \tag{2.4}$$

where W^l is a learnable weight matrix at the l-th GCN layer, ReLU$(\cdot)$ is the non-linear activation function, Deg(v) and Deg(u) represent the degree of node v and u respectively, and $\widetilde{\mathcal{N}}(v)$ is a set of self node v and its neighbour nodes, i.e., $v \cup \mathcal{N}(v)$. The Combine$(\cdot)$ step is not necessary for GCN, as the target node information has been integrated into the self-connected $\widetilde{\mathcal{N}}(v)$ during neighbourhood aggregation.

Graph Isomorphism Networks. Weisfeiler-Lehman (WL) test (Weisfeiler and Lehman, 1968; Shervashidze et al., 2011) is an effective and time-efficient algorithm to distinguish whether two graphs are topologically identical, known as the graph isomorphism problem. The one-dimensional form of WL test is similar to the neighbourhood aggregation in GNNs.

Based on this, Xu et al. (2019) developed graph isomorphism networks (GIN) that achieve the powerful expressive (or discriminative) ability within the message-passing framework. The l-th GIN layer updates node representations as:

$$\mathbf{h}_v^{(l)} = \text{MLP}^{(l)}\left(\left(1+\varepsilon^{(l)}\right)\cdot h_v^{(l-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{l-1}\right), \tag{2.5}$$

where $\varepsilon^{(l)}$ can be a learnable weight or a fixed scalar, and $\text{MLP}^{(l)}(\cdot)$ is a learnable multi-layer perception with a non-linear activation function. GIN applies a simple summation scheme for neighbourhood aggregation, but it can achieve maximum expressiveness compared to other common schemes like mean and max. To preserve distinct adjacency information in aggregation, the feature transformation is parameterized by an MLP according to the universal approximation theorem (Hornik et al., 1989). After that, the entire graph representation can be derived by:

$$\mathbf{h}_{\mathcal{G}} = \text{Concat}\left(\text{Readout}\left(\left\{\mathbf{h}_v^{(l)} | v\in\mathcal{G}\right\}\right) | l = 0,1,...,L\right), \tag{2.6}$$

where $\text{Concat}(\cdot)$ is the concatenation of the graph representations across all GIN layers.

There are many other GNN variants proposed with different aggregation, combination and pooling schemes. GraphSAGE (Hamilton et al., 2017) designs an inductive aggregation scheme for large graphs, which updates node representations from the sampled neighbours at each iteration. It can generate inductive representations for unseen nodes during training. In addition, graph attention networks (GAT) (Veličković et al., 2018) leverage the attention mechanism Bahdanau et al. (2015) to dynamically compute the aggregation weights for different neighbours. The adaptive attention module allows GAT to focus on significant parts of the graph based on the specific context and attributes of nodes. The proposal of GNNs has laid the foundations for the rapid development of graph representation learning in many applications.

2.2.3 Graph Transformers

Transformer architecture (Vaswani et al., 2017) has achieved significant success in various fields, such as natural language and computer vision. Recently, there have been many successful Transformer variants (Ying et al., 2021; Fuchs et al., 2020; Rong et al., 2020) proposed for adapting graph-structured data, commonly known as Graph Transformers. In comparison to most GNNs, Graph Transformers demonstrate their great potential to break through some inherent limitations within the message-passing framework. For instance, GNNs are known to suffer from the over-squashing problem due to the information distortion during long-distance interactions, as well as the over-smoothing problem caused by repeated local aggregation (Topping et al., 2021; Min et al., 2022). Additionally, the expressive power of popular GNNs is inherently limited by the Weisfeiler-Lehman isomorphism test (Xu et al., 2019; Maron et al., 2019). A central cause of the above problems is that the fixed local aggregation in message passing naturally restricts the model's computation graph and flexibility. To alleviate these issues, there has been an increasing interest in developing Graph Transformers that consider graph structural information as a soft inductive bias (Kreuzer et al., 2021).

A vanilla Transformer architecture is s comprised of stacked Transformer layers, each of which consists of a multi-head self-attention module followed by a feed-forward network. Both modules embed layer normalization (Ba et al., 2016) and skip connection (He et al., 2016) to reduce the internal covariate shift and to avoid the vanishing gradient problem, respectively. For any graph $\mathcal{G}$, we can first consider the node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ as the input without other structural information. The self-attention module projects $\mathbf{X}$ into the three representations $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ with the corresponding learnable projection matrices $\mathbf{W}_q \in \mathbb{R}^{D_k \times D}$, $\mathbf{W}_k \in \mathbb{R}^{D_k \times D}$ and $\mathbf{W}_v \in \mathbb{R}^{D_v \times D}$. Next, the attention output is calculated using the scaled dot-product self-attention mechanism, which can be written as:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q^T, \mathbf{K} = \mathbf{X}\mathbf{W}_k^T, \mathbf{V} = \mathbf{X}\mathbf{W}_v^T, \tag{2.7}$$

$$\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{d_k}}, \tag{2.8}$$

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}(\mathbf{A})\mathbf{V}, \tag{2.9}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ is an attention weight matrix and each element A_{ij} indicates the importance of the j-th position while computing the output representation of the i-th position in Equation (2.9). Assume there are m attention heads in the multi-head self-attention module, then the output matrix can be denoted as:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{Concat}(Head_1,...,Head_m)\mathbf{W}_o, \\ Head_i &= \text{Attention}(\mathbf{X}(\mathbf{W}_q^i)^T,\mathbf{X}(\mathbf{W}_k^i)^T,\mathbf{X}(\mathbf{W}_v^i)^T), \end{aligned} \tag{2.10}$$

where $\mathbf{W}_q^i \in \mathbb{R}^{D_k\times D}$, $\mathbf{W}_k^i \in \mathbb{R}^{D_k\times D}$ and $\mathbf{W}_v^i \in \mathbb{R}^{D_v\times D}$ are the projection matrices of the i-th head, $\mathbf{W}_o \in \mathbb{R}^{mD_v\times D}$ is the multi-head output projection matrix and $\text{Concat}(\cdot)$ is the concatenation operation across all heads.

The standard self-attention mechanism in Transformer actually regards the input nodes as a fully connected graph, ignoring the inherent structures within the actual graph. As a result, the performance is naturally poor for graph-specific tasks. In this situation, many Graph Transformers have been proposed to incorporate graph structural information as a necessary inductive bias into the vanilla Transformer. These methods can be broadly categorized into three groups (Min et al., 2022): (1) Using GNNs as auxiliary modules for Transformers. (2) Designing graph structure-aware positional encoding. (3) Improving self-attention matrices from graphs.

Using GNNs as auxiliary modules for Transformer. An intuitive solution for introducing structural information is to directly combine the vanilla Transformer architecture with existing GNNs. The GNN modules are considered as auxiliary modules to learn local structure representations by neighbourhood aggregation, while the Transformer modules can further extract global relations by computing pairwise interactions between all nodes. There are general common combination types as shown in Figure 2.4: (1) Building Transformer modules on top of GNN modules (Rong et al., 2020; Wu et al., 2021a; Mialon et al., 2021). (2) Stacking Transformer and GNN layers as a unified module (Lin et al., 2021). (3) Parallelizing Transformer modules and GNN modules (Zhang et al., 2020).

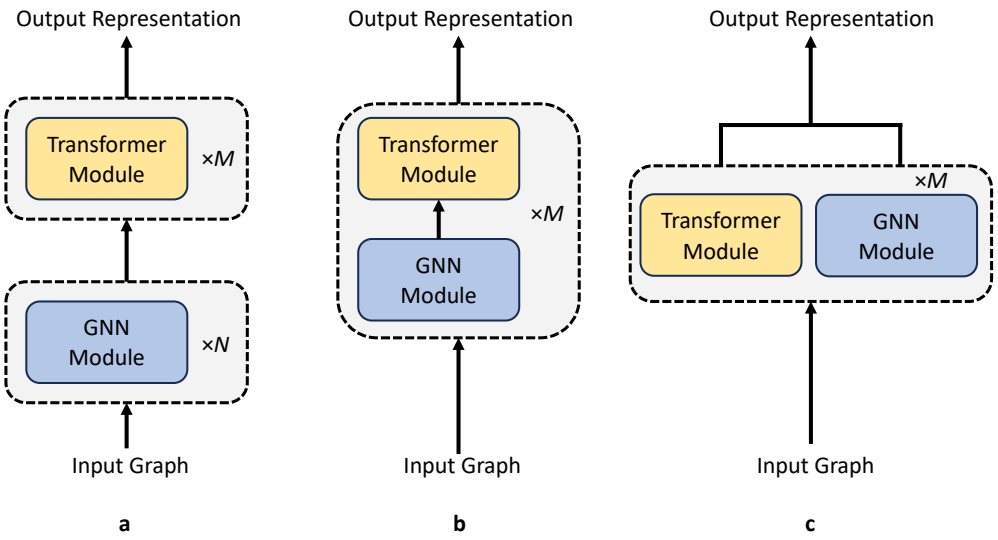

Fig. 2.4 Three general types of using GNNs as auxiliary modules. (a) Building GNNs before Transformer. (b) Stacking GNNs with Transformer as a unified module. (3) Building GNNs and Transformer in parallel.

Designing graph-aware positional encoding. Although the auxiliary GNN modules have demonstrated their effectiveness in introducing graph structures, these methods significantly increase model complexity, and present additional challenges in optimization due to the vast hyper-parameter search space. The second group of methods involves developing graph-aware positional encoding. Positional encoding is a critical component of the vanilla Transformer architecture used for modeling sequence data, Specifically, the positional embedding vectors are added to the input tokens prior to being passed into the self-attention module. Similar to the mechanism, some graph structure-aware positional encodings are proposed to inject graph inductive bias into Transformer (Cai and Lam, 2020; Kreuzer et al., 2021; Dwivedi et al., 2023). For example, Ying et al. (2021) leverages the node degree centrality as structural signals, and adds the centrality encoding of each node to its feature vector as the model input. Dwivedi et al. (2023) factorizes the graph Laplacian matrix of each graph in the dataset, and uses the q smallest non-trivial eigenvectors as the positional encoding for Graph Transformer. Graph Laplacian matrix is defined as the difference between the degree matrix and the adjacency matrix. its eigenvectors can distinguish the local structures of graph nodes while preserving long-range dependency.

Improving self-attention matrices from graphs. Graph-aware positional encoding is an effective approach to inject graph inductive bias into Transformer. However, different from sequence data, the compression of complex graph structures into individual node vectors can lead to information loss, potentially limiting model performance. The third group of methods aims to incorporate more accurate graph information into the self-attention matrix computation. Some methods (Yao et al., 2020; Dwivedi and Bresson, 2021) design masked self-attention mechanism that restricts the scope of global attention to a local domain with a specific graph structure, which can be defined as:

$$\mathbf{A} = (\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}) \odot \phi_M(\mathbf{P}_\mathcal{G}), \tag{2.11}$$

where $\mathbf{P}_\mathcal{G}$ is the adjacency matrix of graph $\mathcal{G}$ and $\phi_M(\cdot)$ is a mapping function to determine the specific mask region. If $\phi_M(\cdot)$ is an identity function, this mask self-attention is similar to a GNN architecture that considers the one-hop neighbourhood aggregation.

Another line of methods aims to add graph priors as bias terms into the computation of the self-attention matrix (Ying et al., 2021; Zhao et al., 2021a; Li et al., 2022a). They measure crucial structural features, such as the shortest path between node pairs, and integrate them as bias matrices in the self-attention module. Thus, the attention weight matrix can be denoted as:

$$\mathbf{A} = (\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}) + \sum_{i=1}^{n} \mathbf{B}_\mathcal{G}^i, \tag{2.12}$$

where $\mathbf{B}_\mathcal{G}^i$ is the i-th bias matrix associated with structural information. Many distinct bias encodings have been developed to inject graph priors. For instance, Graphormer (Ying et al., 2021) designs two simple but effective techniques: spatial encoding and edge encoding. In spatial encoding, they first measure the shortest path distance between any node pairs. If two nodes are not connected, their distance is set to a specific value, such as -1. After that, each distance value is assigned a learnable scalar as a bias term in the self-attention module. In edge encoding, they search the edges on the shortest path of node pairs, and compute the average of dot-products between edge features and a learnable weight matrix as the second graph bias term.

2.3 Denoising Diffusion Probabilistic Models

Diffusion models represent a new state-of-the-art class of deep generative models. They have shown remarkable potential across diverse domains, including computer vision (Brempong et al., 2022; Ho et al., 2020), natural language processing (Austin et al., 2021; Yu et al., 2022b), and molecule generation in drug discovery (Hoogeboom et al., 2022; Jing et al., 2022). Intuitively, a diffusion model defines the diffusion process that gradually adds random noise into data, thereby transforming complex data distributions into a simple prior distribution (Yang et al., 2022b). Then, it learns to reverse the diffusion process to generate new data by sampling noise from the prior distribution. In this section, we will mainly introduce denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020; Sohl-Dickstein et al., 2015; Kingma et al., 2021), a prevalent type of diffusion models. In particular, DDPMs have been widely used in generative tasks for drug discovery and serve as the foundation of our subsequent research in molecule generation.

Formally, a DDPM defines two Markov chains to perform the diffusion and denoising process: a forward diffusion chain that gradually injects Gaussian noise to perturb original data, and a reverse denoising chain that iteratively removes noise to generate new data. The forward diffusion process transforms the original data distribution into standard Gaussians by constant noise schedules, while the reverse denoising process learns the Markov transition kernels that are parameterized by deep neural networks. Given a data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and a fixed number of diffusion steps T, the joint distribution and transition kernel in the forward diffusion process can be written as follows (Ho et al., 2020):

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \tag{2.13}$$

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}), \tag{2.14}$$

where $\beta_t \in (0,1)$ is a pre-defined noise schedule that controls the weight of Gaussian noise added at the t-th diffusion step. The noise schedules $\{\beta_1, ..., \beta_T\}$ are incremental and ensure

that $\mathbf{x}_T$ can nearly converge to a standard Gaussian distribution, i.e., $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. A remarkable property of the Gaussian transition kernel is that we can easily generate a sample of $\mathbf{x}_t$ at an arbitrary diffusion step t when $\mathbf{x}_0$ is given (Sohl-Dickstein et al., 2015). By defining the notations $\alpha_t \coloneqq 1 - \beta_t$ and $\bar{\alpha}_t \coloneqq \prod_{s=0}^{t} \alpha_s$, we have:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.15}$$

In the reverse denoising process, DDPM begins by sampling a random noise vector from the prior Gaussian distribution, i.e., $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. It then intuitively removes noise by the reverse denoising Markov chain and its parameterized transition kernel, which can be defined as:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \tag{2.16}$$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}), \tag{2.17}$$

where the mean $\mu_\theta(\mathbf{x}_t, t)$ is typically parameterized by deep neural networks, and the variance σ_t is a hyperparameter.

As the forward diffusion process $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$ can be regarded as a fixed posterior distribution, the reverse process is optimized by maximizing the variational lower bound (VLB) on the log-likelihood of the observed data:

$$\begin{aligned} \log p_\theta(\mathbf{x}_0) &\overset{(i)}{=} \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right] \\ &\overset{(ii)}{\geq} \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right] \\ &\overset{(iii)}{=} \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right], \end{aligned} \tag{2.18}$$

where (ii) is derived from Jensen's inequality. The VLB maximization is a typical objective for the training of probabilistic generative models (Yang et al., 2022b). However, the direct

estimation of the mean $\mu_\theta(\mathbf{x}_t, t)$ suffers from training instability (Nichol and Dhariwal, 2021). Instead, DDPMs propose to use a surrogate objective that simply predicts the noise ε added to each diffusion step for optimization:

$$\mathcal{L}_{DDPM} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \varepsilon \sim \mathcal{N}(0,\mathbf{I}), t} \left[\lambda(t) || \varepsilon - \varepsilon_\theta(\mathbf{x}_t, t) ||^2 \right], \tag{2.19}$$

where $\varepsilon_\theta(\mathbf{x}_t, t)$ represents an estimated noise produced by deep neural networks. $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ as described in Equation (2.15). $\lambda(t)$ denotes a positive weighting term. It is simply set to 1 in practice, which can generally lead to better sample quality. Figure 2.5 illustrates the training flow of DDPM.

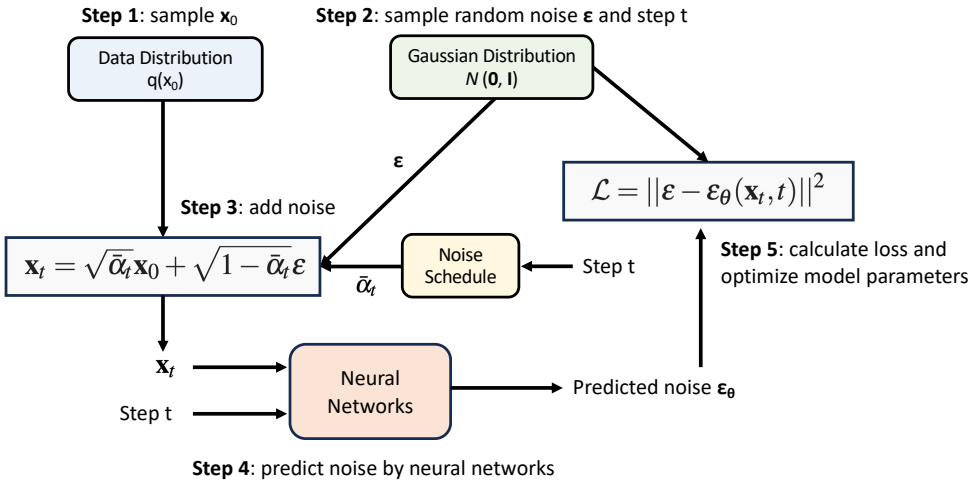

Fig. 2.5 Illustration of the DPMM training flow.

After training, DDPM can generate new data with ε_θ by the iterative denoising sampling:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}, \tag{2.20}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the reverse denoising chain starts from the Gaussian prior $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

DDPM models the generative process as a discrete-time Markov chain that gradually adds noise to clean data and then learns reverse this process to generate novel samples. Beyond this discrete-time formulation, subsequent studies have provided a more general

theoretical framework by formulating diffusion models in continuous time as stochastic differential equations (SDEs) and their equivalent deterministic probability flow ordinary differential equations (ODEs). This continuous-time interpretation, developed by Song et al. (2020), unifies score-based generative modelling and enables exact likelihood computation. More recently, the flow-matching framework (Lipman et al., 2022) extends these ideas by directly estimating a transport vector field that deterministically maps data to noise along more straight trajectories. It provides an alternative to stochastic diffusion-based processes and has inspired efficient generative models such as latent diffusion models (Rombach et al., 2022).

2.4 Applications in Drug Discovery

2.4.1 Drug-target Interaction Prediction

The identification of drug-target interaction (DTI) is a fundamental task in drug discovery. Traditional biomedical methods for measuring DTI are reliable, but they are also costly and time-consuming due to the requirement of conducting in vitro experiments over enormous drug compound space. In comparison, computational methods employing in silico techniques can significantly narrow down the search space at an early stage, and have attracted growing attention over the past few years.

Owing to the promising advancements in machine learning, many effective computational methods have been developed for DTI prediction. Similarity-based methods design distance score schemes that utilize known drug-drug and protein-protein similarities to perform predictions (Buza and Peka, 2017; Perlman et al., 2011). However, these methods have limited generalization to structurally dissimilar drug compounds and proteins. Feature-based methods use classic machine learning models, such as random forest and support vector machine, on the combination of numerical drug and protein descriptors (Cao et al., 2012; Geppert et al., 2009; Ning et al., 2009). Network-based methods integrate multiple data sources of drugs and proteins into a heterogeneous network, and apply graph-based techniques to identify interactions (Luo et al., 2017; Yu et al., 2022a). Following a chemogenomics

perspective (Bredel and Jacoby, 2004; Yamanishi et al., 2008), many deep learning-based methods integrate both chemical and genomic space into a unified end-to-end framework to make DTI predictions (Tsubaki et al., 2019; Huang et al., 2021; Bai et al., 2023).

2.4.2 Molecular Property Prediction

Recent years have witnessed significant progress in applying deep learning in molecular property prediction. As the foundation of deep learning-based property prediction, many studies have been devoted to improving molecular representation learning. Traditional feature-based methods depend on hand-crafted representations generated by chemical fingerprints or molecular descriptors, which lack the ability to capture potential structural information and require extensive feature engineering (Rogers and Hahn, 2010; Cao et al., 2008). Recently, the emergence of GNNs has allowed for more intuitive and efficient learning of molecular graph representations. Gilmer et al. (2017) first proposes to use a message-passing framework to capture atomic interactions in molecular graphs, and then Yang et al. (2019) extends the framework by considering the directional bond interactions. Xiong et al. (2019) leverages graph attention networks to learn the data-driven fingerprints in a dense vector space. Nevertheless, these approaches face the challenge of capturing long-range dependencies, which has led to the exploration of transformer-based architectures to preserve graph structures and improve the expressiveness of learned molecular representations (Ying et al., 2021; Pyzer-Knapp et al., 2022; Mialon et al., 2021).

Self-supervised learning has become a fundamental paradigm for developing generalizable molecular representations. This approach typically involves pre-training a deep model on a large dataset of unlabeled molecules, followed by fine-tuning on downstream property tasks using limited labeled data. Molecular pre-training tasks are broadly classified into contrastive methods and generative methods. The objective of contrastive methods is to maximize the mutual information between the augmented views from the same molecular graph. These approaches suggest various augmentation strategies, such as atom masking and bond deletion, to create semantically similar views (You et al., 2020; Sun et al., 2021; Subramonian, 2021). In contrast, generative methods consider reconstructing the information of a single molecular

graph at different levels, or creating certain pseudo labels for pre-training tasks (Rong et al., 2020; Zhang et al., 2021; Li et al., 2022a). For example, Zhang et al. (2021) proposes a framework to capture molecular information from both node-level and graph-level fragmentations, which predicts masked atom/bond-based attributes and reconstructs motif-based graph trees. More recently, a few studies have focused on integrating geometric knowledge into the pre-training framework to obtain more informative representations. In particular, Liu et al. (2022) and Stärk et al. (2022) propose to encode the 2D topology and 3D geometry by different GNN backbone models, and then implicitly inject 3D information into 2D GNN with a hybrid method.

2.4.3 Inverse Protein Folding

Inverse protein folding (IPF), also known as structure-based protein design, is a fundamental problem in computational biology and medicine. It aims to generate possible amino acid (AA) sequences that can fold into a desired 3D backbone structure, enabling the creation of novel proteins with specific functions. There are enormous applications for IPF in both academic and industrial scenarios, ranging from therapeutic protein engineering, enzyme design and targeted drug design.

Traditional physics-based approaches consider IPF as an energy optimization problem (Alford et al., 2017), which suffers from high computational cost and insufficient accuracy. In recent years, deep learning has emerged as the preferred paradigm for solving protein structure problems, due to its robust ability to adaptively learn complex non-linear patterns from data. There have been many efforts in IPF with deep learning models. Early MLP-based and CNN-based methods regard each residue in proteins as an isolated unit, or the whole as point cloud data when modelling, without sufficiently considering structural information and interactions between residues (Wu et al., 2021b; Li et al., 2014; O'Connell et al., 2018; Anand et al., 2022). Lately, graph-based methods have emerged that utilize proximity graphs to represent 3D protein structures, and then apply graph neural networks (GNNs) to capture residue features while incorporating structural constraints. Owing to the intrinsic

advantage of GNNs in local information aggregation and exchange within graph-structured data, graph-based methods have achieved substantial performance improvement in this field.

Chapter 3

Toward Low-Bias Evaluation for Drug-Target Interaction

3.1 Introduction

Predicting drug-target interaction (DTI) plays an important role in the process of drug discovery, where drugs are chemical compounds and targets are usually target proteins. However, as described in Chapter 1, a large performance gap still exists between academic research and industrial application in drug discovery, where academic results tend to be over-optimistic for industrial settings. Therefore, this chapter focuses on studying the low-bias evaluation of machine learning within the context of DTI prediction. Similar to the three pitfalls in machine learning pointed out by Riley (2019), here we identify three common pitfalls that cause high-bias DTI performance evaluation:

- **Inappropriate data splitting.** A common practice in machine learning research is to split training and test sets at random and evaluate model performance by the accuracy on the test set (under the assumption that the training and test data have the same distribution). In the context of drug discovery, such random split tends to overestimate model performance in real-world settings. One important reason is that drug compounds in the same series share the same scaffold or large substructure, which is easy to learn as long as a few molecules of this series are contained in the training set.

However, in real applications, chemists often need DTI prediction on a new compound series different from known compounds, which is more challenging and makes random split inappropriate.

- **Low-confidence negative samples.** Although many public databases exist for generating DTI data, researchers often ignore hidden bias during model training. The lack of highly confident negative samples is one of them. Machine learning is a data-driven technique and model performance depends heavily on data quality. DTI papers often randomly generate negative samples from unobserved pairs for training and evaluation, which leads to low confidence because they may include some unknown true positive samples.

- **Drug-wise pair imbalance.** The final important pitfall is the hidden imbalance between positive and negative pairs for each drug. In public DTI datasets (Gilson et al., 2016; Wishart et al., 2018; Kim et al., 2019), it is common that most drugs have only one type of pairs (positive or negative). For these drugs, models can make correct prediction using only drug information without learning appropriate DTI patterns, leading to significant drug-wise pair imbalance. Subsequently, the evaluation result is over-optimistic and the model has poor generalization.

We aim to address the three pitfalls above to reduce bias in training and evaluating machine learning models. We compare five different DTI data splitting strategies: random split, cold drug split, scaffold split, single-linkage split, and our newly proposed density-based hierarchical clustering split. The results demonstrate that the choice of splitting strategy can significantly impact model performance evaluation. The first three strategies do not adequately consider real-world scenarios in drug discovery, leading to over-optimistic performance estimation. In contrast, the two clustering-based splits create more challenging and realistic tasks that that can better reward model's generalization rather than memorization. In addition, we strictly control the hidden data bias in a benchmark dataset and experimentally validate all negative DTI pairs to ensure both interaction types exist for each drug. This leads

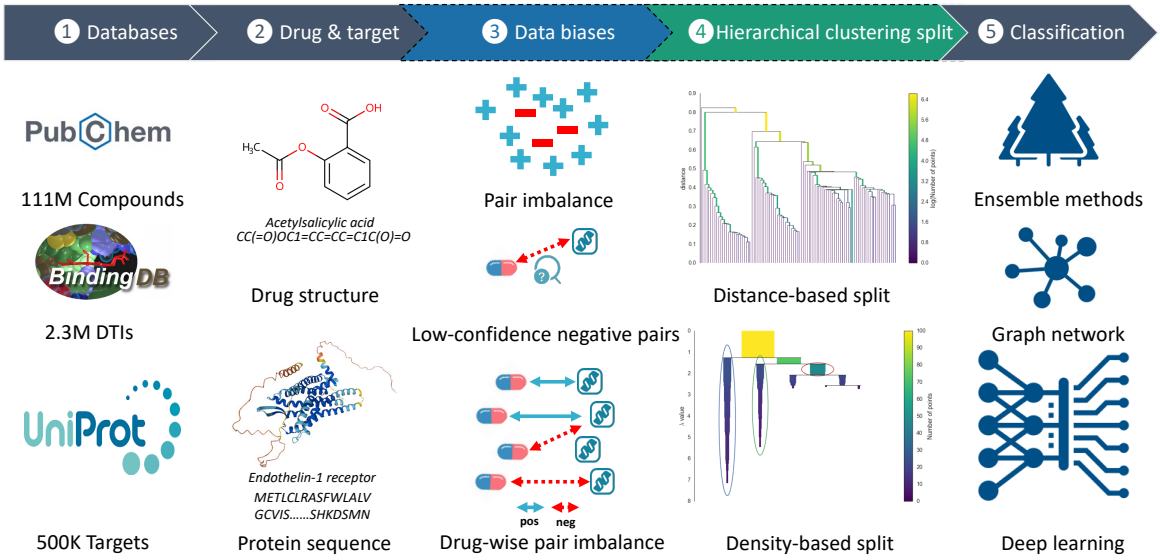

Fig. 3.1 Bias control in a general DTI prediction workflow.

to a low-bias dataset to encourage learning correct interaction information for better DTI prediction.

3.2 Methodology

The DTI prediction task can be viewed as a binary classification of whether a drug forms biological interaction with a target of interest or not. Drugs are commonly encoded as 1D sequences (SMILES) or 2D molecular graphs, and target proteins are typically represented as amino acid sequences. Figure 3.1 shows a general DTI prediction workflow and our bias-controlled evaluation.

3.2.1 Low-Bias Dataset Construction

Most DTI datasets are not originally designed for training machine learning models. They have hidden data bias and tend to produce over-optimistic results. We propose to reconstruct the experimental datasets following two bias removal guidelines:

- *High-confidence negative samples should be used.* The best option is to select experimentally validated pairs. We can set a safe margin in measured binding affinity to select negative samples (Gao et al., 2018). We can also employ some similarity-based DTI negative sampling algorithms (Liu et al., 2015).

- *The number of drugs containing only one interaction type should be removed or reduced.* Many DTI experiments only consider the imbalance between positive and negative pairs across the whole dataset, e.g., by keeping a fixed ratio without considering the pair imbalance for individual drugs, leading to prediction based only on drug features rather than drug-target interaction.

3.2.2 Classic Data Split Strategies

We introduce three classical and one clustering-based data split strategies: random split, cold drug split, scaffold split and single-linkage split (Mayr et al., 2018).

- *Random split* is the most popular data split strategy. DTI pairs are randomly split into train, validation, and test with given ratios.

- *Cold drug split* first randomly splits drugs into train/validation/test, and then puts all DTI pairs associated with individual drugs in corresponding sets as the final splits.

- *Scaffold split* is based on 2D molecular structures that partition drugs into different bins according to their Murcko scaffolds. These bins are then randomly split into train/validation/test sets so that all drugs associated with a bin are part of the same set. Next, all DTI pairs associated with the drugs in a bin are assigned to the corresponding sets.

- *Single-linkage split* is a clustering-based strategy to ensure that the distances between clusters are always larger than a pre-defined threshold. This strategy can cluster drugs by their chemical fingerprints such as ECFP4 (Rogers and Hahn, 2010) in this study, and then apply Jaccard distance on binarized ECFP4 to measure the pairwise distance

between drugs. For single-linkage hierarchical clustering, each cluster of drugs will only be assigned to one of the train, validation and test sets.

3.2.3 HDBSCAN for Data Split

However, single-linkage split uses a single distance threshold that cannot separate clusters of different densities, which are common in drug compound series. Therefore, we propose a Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (McInnes et al., 2017) strategy to split data. HDBSCAN is a hierarchical clustering method that transforms the original distances between data points to density. It is designed for clusters of varying densities. We investigate the performance differences between HDBSCAN split and other split strategies. We review HDBSCAN briefly below:

1. *Calculate the mutual reachability distance (MRD).* To find clusters, HDBSCAN first computes the MRD between data points a and b as:

$$d_k(a,b) = \max\left\{\text{Core}_k(a), \text{Core}_k(b), \text{Distance}(a,b)\right\}, \tag{3.1}$$

where k is a hyperparameter indicating the number of nearest neighbors, $\text{Core}_k(a)$ is the core distance between the core a and its k-th nearest neighbor, and $\text{Distance}(a,b)$ is the distance between a and b with the original metric. This MRD metric allows dense points with low core distance to remain at the same distance from each other while sparse points with high core distance are pushed away.

2. *Build a minimum spanning tree (MST).* After getting MRD, an MST is built from a weighted graph. In this graph, vertices are the data points and the weight of an edge between any two points is their MRD. Then a tree is built one edge at a time by adding the edge with the lowest weight. Meanwhile this added edge needs to bridge the current tree and a vertex that is not in this tree. Given the MST, the next step is to convert it into cluster hierarchy. HDBSCAN sorts edges in MST by distance with ascending order and then iterates to create a new merged cluster for each edge.

3. *Condense the cluster tree*. a minimum cluster size is introduced as a hyperparameter. At each hierarchy split, the sizes of newly generated clusters are compared with the minimum cluster size. If a new cluster has fewer points than the minimum cluster size, HDBSCAN declares them to be "points falling out of a cluster". If the size of a new cluster is equal to or larger than the minimum cluster size, HDBSCAN treats it as a true cluster split to persist in the tree.

4. *Compute the stability and extract clusters*. To extract clusters from the condensed cluster tree, HDBSCAN defines a stability, which aims to choose clusters that persist and have a longer lifetime. First, for a given cluster in hierarchy, HDBSCAN defines a measure λ inversely proportional to the distance threshold, so λ_{birth} denotes the λ when the cluster is split from its parent cluster. Each falling point p in the cluster has a value λ_p so that each cluster c has a stability s:

$$s(c) = \sum_{p \in c} (\lambda_p - \lambda_{\text{birth}}) . \tag{3.2}$$

   Now traverse the condensed cluster tree from all leaf nodes to root. If the sum of the stabilities of child clusters is greater than the stability of their parent cluster, the parent cluster stability is set to be the sum of the child stabilities. Otherwise, the parent cluster is selected to be one of the final clusters.

3.3 Experiments

3.3.1 Dataset Construction

We construct a low-bias version of binary BindingDB (Gilson et al., 2016) dataset in this experiment. Following the IC_{50} threshold used by Gao et al. (2018), we consider a drug-target pair to be positive if its IC_{50} is less than 100 nm, and negative if its IC_{50} is greater than 10,000 nm, which gives a 100-fold difference to reduce class label noise.

Bias-reducing Preprocessing. Due to the drug-wise pair imbalance, 91% of drugs only have one type of pairs (positive or negative) in the binary BindingDB dataset. This implies that we can train a model to make right classification without considering protein information for DTI pairs associated with the 91% of drugs. Therefore, high classification accuracy does not indicate successful learning of correct DTI patterns. We further process the data by removing all DTI pairs of drugs containing only one pair type. This gives us a low-bias dataset with 29,674 positive samples and 32,752 negative samples. Figure 3.2a shows the drug probability distribution in terms of log ratios of positive to negative samples in the dataset, which is calculated as:

$$ln^i_{ratio} = ln\frac{N^i_{pos}}{N^i_{neg}}, \tag{3.3}$$

where N^i_{pos} is the number of positive interactions for drug i, and N^i_{neg} is the number of negative interactions. Following the steps above, our constructed dataset addresses common pitfalls and has three benefits: (1) The number of positive and negative samples is balanced. (2) All negative samples are experimentally validated and highly confident. (3) The drug-wise pair imbalance in DTI pairs is removed.

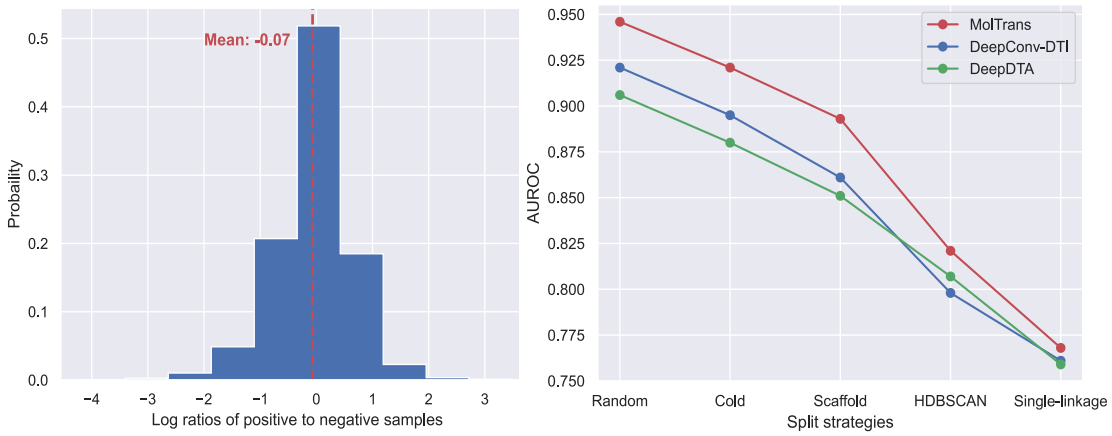

Fig. 3.2 (**a**) Drug probability distribution in terms of $ln(N^i_{pos}/N^i_{neg})$ in the constructed low-bias dataset. The red line indicates the mean log ratio for all drugs (mean=-0.07). $ln(N^i_{pos}/N^i_{neg}) = 0$ when the number of positive and negative interactions are equal for drug i. (**b**) Comparison of three models using five different split strategies.

3.3.2 Metrics

We use AUROC, AUPRC, and accuracy as the major metrics to measure model performance. We also report the sensitivity and specificity metrics at the best F1 score.

3.3.3 Split Strategies

We study five split strategies in model performance evaluation: random split, cold drug split, scaffold split, single-linkage split, and HDBSCAN split, as detailed in Section II.

We set the distance threshold to 0.5 for single-linkage split and minimum cluster size to 5 for HDBSCAN split. Each split strategy keeps a 7:1:2 ratio for training/validation/test sets. We conduct five independent runs with different random seeds for each split. We compare the three algorithms below on the same splits.

3.3.4 Learning Algorithms

Three state-of-the-art deep learning DTI models are selected for performance comparison:

- **DeepConv-DTI** (Lee et al., 2019) models DTI using convolutional neural network (CNN) and one global max-pooling layer to learn protein sequence features, and one fully connected layer to encode drug fingerprints ECFP4.
- **DeepDTA** (Öztürk et al., 2018) uses CNN on both drug SMILES string and protein amino acid sequence to extract local residue patterns. As the original DeepDTA is a regression model to predict binding affinity. A sigmoid function is added after the last layer of the decoder for binary classification.
- **MolTrans** (Huang et al., 2021) adapts transformer architectures to encode drug and protein information, and introduces an interaction map module with a CNN layer to learn the interaction between molecular substructures.

Table 3.1 MolTrans performance comparison with five data split strategies (**Best**, Worst).

Split strategy	AUROC	AUPRC	Accuracy	Sensitivity	Specificity	Test loss
Random	**0.946±0.003**	**0.935±0.004**	**0.874±0.003**	**0.838±0.01**	**0.914±0.007**	**0.469±0.019**
Cold drug	0.921±0.003	0.909±0.006	0.841±0.004	0.798±0.009	0.889±0.005	0.670±0.052
Scaffold	0.893±0.006	0.874±0.004	0.804±0.005	0.736±0.012	0.882±0.014	0.797±0.099
HDBSCAN	0.821±0.024	0.778±0.031	0.724±0.037	0.581±0.091	0.891±0.03	0.936±0.328
Single linkage	0.768±0.024	0.717±0.025	0.676±0.032	0.483±0.077	0.894±0.023	0.959±0.224

3.3.5 Implementation Details

We follow the same model hyper-parameter settings described in the original papers. The batch size is 64 and each model is allowed to run 100 epochs in each independent training. The learning rate is set to $1e^{-5}$ with the Adam optimizer. The test model is selected at the epoch giving the best AUROC on the validation set. The selected model is evaluated on the test set and metrics are reported.

3.3.6 Performance Gap between Different Split Strategies

To investigate whether there is a significant performance gap for the same method with different data split strategies, Table 3.1 shows the performance of MolTrans with different metrics on the test sets generated by different strategies. As expected, random split has significant information overlap between training and test sets, so it achieves the best performance across all metrics. However, random split's good performance on the test set does not imply the same good performance in real drug discovery, where it is unlikely to have so much prior information while predicting a novel drug-target interaction pair in reality. The performance declines differently in other split strategies. Compared with random split, single-linkage split has the largest performance drop of 18.8% in AUROC and 23.3% in AUPRC.

For other split strategies, the performance drop is also evident. Cold drug split has just a slight drop since it randomly selects drugs without considering drug similarities. Scaffold split further divides drugs by their shared scaffold, so it is more challenging than the cold split. However, as scaffold split considers only well-defined cyclic substructures connected by the linkers, scaffolds that share the topology may still be considered dissimilar. Thus, scaffolds proximal in the fingerprint space can cause information leakage and model bias

if not found in either training or test set. As clustering-based split strategies cluster similar compounds irrespective of their scaffolds, and as part of the same subset, the potential for data leakage is reduced.

We further study the influence of data split strategy on performance of different models. Figure 3.2b plots AUROC of MolTrans, DeepConv-DTI and DeepDTA on the five split strategies. Although MolTrans always outperforms the other two models, their performance gap gradually decreases with the change of split strategies. Comparing random and single-linkage split, the improvement of MolTrans over DeepDTA drops from 4.4% to 1.1%. Moreover, DeepConv-DTI outperforms DeepDTA with random split, but DeepDTA outperforms DeepConv-DTI with HDBSCAN split. This shows that the better performance on random split strategy is over-optimistic because the test set rewards more memorization rather than generalization.

3.4 Summary

In this chapter, we study the low-bias evaluation of machine learning models in DTI prediction. Experimental results showed that traditional split strategies tend to overestimate predictive performance and exaggerate performance gaps between different models. We constructed a low-bias dataset, and adopted two clustering-based split strategies toward more realistic evaluation in drug discovery. Clustering-based splits created the most challenging prediction tasks for evaluating real-world DTI prediction performance.

However, while clustering-based splits can reduce bias in performance evaluation, they potentially increase the variance of evaluation results due to the inclusion of more diverse test samples. Therefore, when adopting low-bias evaluation protocols, researchers should also consider the potential increase in variance and aim to balance both aspects to ensure reliable results. For example, strategies such as repeated cross-validation and robust performance averaging can mitigate high-variance effects and improve the reliability of DTI model evaluation in realistic scenarios.

Chapter 4

Bilinear Attention Network with Domain Adaptation improves Drug-Target Prediction

4.1 Introduction

Recently, deep learning (DL) has rapidly progressed for computational DTI prediction due to its successes in various areas, enabling large-scale validation in a relatively short time (Gao et al., 2018). In Chapter 3, we mainly investigate the low-bias DTI evaluation strategies and show that the predictive performance of existing DL-based methods is overestimated. In this chapter, we follow the low-bias evaluation principle and develop a bilinear attention network-inspired framework with domain adaptation for DTI prediction. Our framework enhances local drug-target modeling and improves specific transferability in both in-domain and cross-domain scenarios.

As pointed out in Section 2.4.1, many DL-based methods are constructed from a chemogenomics perspective, which integrates the chemical space, genomic space, and interaction information into a unified end-to-end framework. They treat DTI prediction as a binary classification task, and make predictions by feeding the inputs into different deep encoding and decoding modules such as deep neural network (DNN) (Lee et al., 2019; Hinnerichs

and Hoehndorf, 2021), graph neural network (GNN) (Gao et al., 2018; Nguyen et al., 2021; Tsubaki et al., 2019; Feng et al., 2018) or transformer architectures (Chen et al., 2020a; Huang et al., 2021).

Despite these promising developments, two challenges remain in DL-based methods. The first challenge is explicit learning of interactions between local structures of drug and protein. DTI is essentially decided by mutual effects between important molecular substructures in the drug compound and binding sites in the protein sequence (Schenone et al., 2013). However, many previous studies learn global representations in their separate encoders, without explicitly learning local interactions (Öztürk et al., 2018; Nguyen et al., 2021; Öztürk et al., 2019; Zheng et al., 2020; Lee et al., 2019). Consequently, drug and protein representations are learned for the whole structures first and mutual information is only implicitly learned in the black-box decoding module. Interactions between drug and target are particularly related to their crucial substructures, thus separate global representation learning tends to limit the modeling capacity and prediction performance. Moreover, without explicit learning of local interactions, the prediction result is hard to interpret, even if the prediction is accurate.

The second challenge is generalizing prediction performance across domains, i.e. out of learned distribution. Due to the vast regions of chemical and genomic space, drug-target pairs that need to be predicted in real-world applications are often unseen and dissimilar to any pairs in the training data. They have different distributions and thus need cross-domain modeling (Abbasi et al., 2020; Kao et al., 2021). A robust model should be able to transfer learned knowledge to a new domain that only has unlabeled data. In this case, we need to align distributions and improve cross-domain generalization performance by learning transferable representations, e.g. from "source" to "target". To the best of our knowledge, this is an underexplored direction in drug discovery (Abbasi et al., 2021).

To address these challenges, we propose an interpretable bilinear attention network-based model (DrugBAN) for DTI prediction, as shown in Figure 4.1a. DrugBAN is a deep learning framework with explicit learning of local interactions between drug and target, and conditional domain adaptation for learning transferable representations across domains.

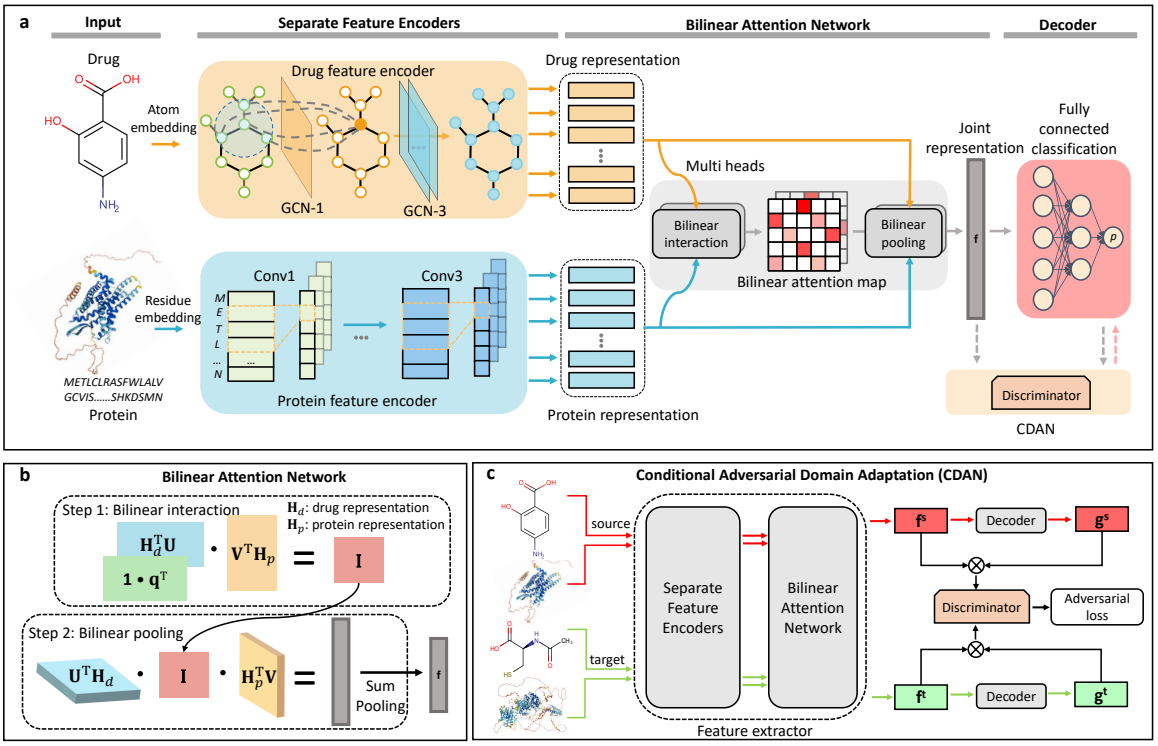

Fig. 4.1 Overview of the DrugBAN framework. **(a)** The input drug molecule and protein sequence are separately encoded by graph convolutional networks and 1D-convolutional neural networks. Each row of the encoded drug representation is an aggregated representation of adjacent atoms in the drug molecule, and each row of the encoded protein representation is a subsequence representation in the protein sequence. The drug and protein representations are fed into a bilinear attention network to learn their pairwise local interactions. The joint representation $\mathbf{f}$ is decoded by a fully connected decoder module to predict the DTI probability p. If the prediction task is cross-domain, the conditional domain adversarial network (CDAN) module is employed to align learned representations in the source and target domains. **(b)** The bilinear attention network architecture. $\mathbf{H}_d$ and $\mathbf{H}_p$ are encoded drug and protein representations. In Step 1, the bilinear attention map matrix $\mathbf{I}$ is obtained by a low-rank bilinear interaction modeling via transformation matrices $\mathbf{U}$ and $\mathbf{V}$ to measure the substructure-level interaction intensity. Then $\mathbf{I}$ is utilized to produce the joint representation $\mathbf{f}$ in Step 2 by bilinear pooling via the shared transformation matrices $\mathbf{U}$ and $\mathbf{V}$. **(c)** CDAN is a domain adaptation technique to reduce the domain shift between different distributions of data. We use CDAN to embed joint representation $\mathbf{f}$ and softmax logits $\mathbf{g}$ for source and target domains into a joint conditional representation via the discriminator, a two-layer fully connected network that minimizes the domain classification error to distinguish the target domain from the source domain.

Specifically, we first use graph convolutional network (Kipf and Welling, 2017) (GCN) and convolutional neural network (CNN) to encode local structures in 2D drug molecular graph and 1D protein sequence, respectively. Then the encoded local representations are fed into a pairwise interaction module that consists of a bilinear attention network (Yu et al., 2018; Kim et al., 2018) to learn local interaction representations, as depicted in Figure 4.1b. The local joint interaction representations are decoded by a fully connected layer to make a DTI prediction. In this way, we can utilize the pairwise bilinear attention map to visualize the contribution of each substructure to the final predictive result, improving the interpretability. For cross-domain prediction, we apply conditional domain adversarial network (CDAN) (Long et al., 2018) to transfer learned knowledge from source domain to target domain to enhance cross-domain generalization, as illustrated in Figure 4.1c. We conduct a comprehensive performance comparison against five state-of-the-art DTI prediction methods on both in-domain and cross-domain settings of drug discovery. The results show that our method achieves the best overall performance compared to state-of-the-art methods, while providing interpretable insights for the prediction results.

To summarize, DrugBAN differs from previous works by (i) capturing pairwise local interactions between drugs and targets via a bilinear attention mechanism, (ii) enhancing cross-domain generalization with an adversarial domain adaptation approach; and (iii) giving an interpretable prediction via bilinear attention weights instead of black-box results.

4.2 Methodology

4.2.1 Problem Formulation

In DTI prediction, the task is to determine whether a pair of a drug compound and a target protein will interact. For target protein, denoting each protein sequence as $\mathcal{P} = (a_1, ..., a_n)$, where each token a_i represents one of the 23 amino acids. For drug compound, most existing deep learning-based methods represent the input by the Simplified Molecular Input Line Entry System (SMILES) (Weininger, 1988b), which is a 1D sequence describing chemical atom and bond token information in the drug molecule. The SMILES format allows encoding

drug information with many classic deep learning architectures. However, since the 1D sequence is not a natural representation for molecules, some important structural information of drugs could be lost, degrading model prediction performance. Our model converts input SMILES into its corresponding 2D molecular graph. Specifically, a drug molecule graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (atoms) and $\mathcal{E}$ is the set of edges (chemical bonds).

Given a protein sequence $\mathcal{P}$ and a drug molecular graph $\mathcal{G}$, DTI prediction aims to learn a model $\mathcal{M}$ to map the joint feature representation space $\mathcal{P} \times \mathcal{G}$ to an interaction probability score $p \in [0, 1]$.

4.2.2 Framework Overview

Figure 4.1a shows the proposed DrugBAN framework. Given an input drug-target pair, firstly, we employ separate graph convolutional network (GCN) and 1D-convolutional neural network (1D-CNN) blocks to encode molecular graph and protein sequence information, respectively. Then we use a bilinear attention network module to learn local interactions between encoded drug and protein representations. The bilinear attention network consists of a bilinear attention step and a bilinear pooling step to generate a joint representation, as illustrated in Figure 4.1b. Next, a fully connected classification layer learns a predictive score indicating the probability of interaction. For improving model generalization performance on cross-domain drug-target pairs, we further embed CDAN into the framework to adapt representations for better aligning source and target distributions, as depicted in Figure 4.1c.

4.2.3 Protein Sequence Encoder

The protein feature encoder consists of three consecutive 1D-convolutional layers, which transforms an input protein sequence to a matrix representation in the latent feature space. Each row of the matrix denotes a subsequence representation in the protein. Drawing on the concept of word embedding, we first initialize all amino acids into a learnable embedding matrix $\mathbf{E}_p \in \mathbb{R}^{23 \times D_p}$, where 23 is the number of amino acid types and D_p is the latent

space dimensionality. By looking up $\mathbf{E}_p$, each protein sequence $\mathcal{P}$ can be initialized to corresponding feature matrix $\mathbf{X}_p \in \mathbb{R}^{\Theta_p \times D_p}$. Here Θ_p is the maximum allowed length of a protein sequence, which is set to align different protein lengths and make batch training. Following previous works (Huang et al., 2021; Öztürk et al., 2018; Tsubaki et al., 2019), protein sequences with maximum allowed length are cut, and those with smaller length are padded with zeros.

The CNN-block protein encoder extracts local residue patterns from the protein feature matrix $\mathbf{X}_p$. Here a protein sequence is considered as an overlapping 3-mer amino acids such as "METLCL...DSMN" $\rightarrow$ "MET", "ETL", "TLC",..., "DSM", "DLK". The first convolutional layer is utilized to capture the 3-mer residue-level features with kernel size $= 3$. Then the next two layers continue to enlarge the receptive field and learn more abstract features of local protein fragments. The protein encoder is described as follows:

$$\mathbf{H}_p^{(l+1)} = \sigma(\mathrm{CNN}(\mathbf{W}_c^{(l)}, \mathbf{b}_c^{(l)}, \mathbf{H}_p^{(l)})), \tag{4.1}$$

where $\mathbf{W}_c^{(l)}$ and $\mathbf{b}_c^{(l)}$ are the learnable weight matrices (filters) and bias vector in the l-th CNN layer. $\mathbf{H}_p^{(l)}$ is the l-th hidden protein representation and $\mathbf{H}_p^{(0)} = \mathbf{X}_p$. $\sigma(\cdot)$ denotes a non-linear activation function, with $\mathrm{ReLU}(\cdot)$ used in our experiments.

4.2.4 Molecular Graph Encoder

For drug compound, we convert each SMILES string to its 2D molecular graph $\mathcal{G}$. To represent node information in $\mathcal{G}$, we first initialize each atom node by its chemical properties, as implemented in the DGL-LifeSci (Li et al., 2021) package. Each atom is represented as a 74-dimensional integer vector describing eight pieces of information: the atom type, the atom degree, the number of implicit Hs, formal charge, the number of radical electrons, the atom hybridization, the number of total Hs and whether the atom is aromatic. Similar to the maximum allowed length setting in a protein sequence above, we set a maximum allowed number of nodes Θ_d. Molecules with less nodes will contain virtual nodes with zero padded. As a result, each graph's node feature matrix is denoted as $\mathbf{M}_d \in \mathbb{R}^{\Theta_d \times 74}$. Moreover, we use

a simple linear transformation to define $\mathbf{X}_d = \mathbf{W}_0\mathbf{M}_d^\top$, leading to a real-valued dense matrix $\mathbf{X}_d \in \mathbb{R}^{\Theta_d \times D_d}$ as the input feature.

We employed a three-layer GCN-block to effectively learn the graph representation on drug compounds. GCN generalizes the convolutional operator to an irregular domain. Specifically, we update the atom feature vectors by aggregating their corresponding sets of neighborhood atoms, connected by chemical bonds. This propagation mechanism automatically captures substructure information of a molecule. We keep the node-level drug representation for subsequent explicit learning of local interactions with protein fragments. The drug encoder is written as:

$$\mathbf{H}_d^{(l+1)} = \sigma(\mathrm{GCN}(\tilde{\mathbf{A}}, \mathbf{W}_g^{(l)}, \mathbf{b}_g^{(l)}, \mathbf{H}_p^{(l)})), \tag{4.2}$$

where $\mathbf{W}_g^{(l)}$ and $\mathbf{b}_g^{(l)}$ are the GCN's layer-specific learnable weight matrix and bias vector, $\tilde{\mathbf{A}}$ is the adjacency matrix with added self-connections in molecular graph $\mathcal{G}$, and $\mathbf{H}_d^{(l)}$ is the l-th hidden node representation with $\mathbf{H}_d^{(0)} = \mathbf{X}_d$.

4.2.5 Bilinear Attention Network

Bilinear attention network (BAN) was first proposed to solve the problem of visual question answering (VQA) (Kim et al., 2018). It uses a bilinear attention map to gracefully extend unitary attention networks for adapting multimodal learning, which considers every pair of multimodal input channels. Compared to using a unitary attention mechanism directly on multimodal data, BAN can provide richer joint information but keep the computational cost at the same scale. Due to the problem similarity between VQA and DTI, we design a BAN-inspired pairwise interaction module to integrate drug molecule and target protein encodings. This module consists of two layers: (i) A bilinear interaction map to capture pairwise attention weights and (ii) a bilinear pooling layer over the interaction map to extract joint drug-target representation.

Given the third layer's hidden protein and drug representations $\mathbf{H}_p^{(3)} = \{\mathbf{h}_p^1, \mathbf{h}_p^2, ..., \mathbf{h}_p^M\}$ and $\mathbf{H}_d^{(3)} = \{\mathbf{h}_d^1, \mathbf{h}_d^2, ..., \mathbf{h}_d^N\}$ after separate CNN and GCN encoders, where M and N denote

the number of encoded substructures in a protein and atoms in a drug. The bilinear interaction map can obtain a single head pairwise interaction $\mathbf{I} \in \mathbb{R}^{N \times M}$:

$$\mathbf{I} = ((\mathbf{1} \cdot \mathbf{q}^\top) \circ \sigma((\mathbf{H}_d^{(3)})^\top \mathbf{U})) \cdot \sigma(\mathbf{V}^\top \mathbf{H}_p^{(3)}), \tag{4.3}$$

where $\mathbf{U} \in \mathbb{R}^{D_d \times K}$ and $\mathbf{V} \in \mathbb{R}^{D_p \times K}$ are learnable weight matrices for drug and protein representations, $\mathbf{q} \in \mathbb{R}^K$ is a learnable weight vector, $\mathbf{1} \in \mathbb{R}^N$ is a fixed all-ones vector, and $\circ$ denotes Hadamard (element-wise) product. The elements in $\mathbf{I}$ indicate the interaction intensity of respective drug-target sub-structural pairs, with mapping to potential binding sites and molecular substructures. To intuitively understand bilinear interaction, an element $\mathbf{I}_{i,j}$ in Equation (4.3) can also be written as:

$$\mathbf{I}_{i,j} = \mathbf{q}^\top (\sigma(\mathbf{U}^\top \mathbf{h}_d^i) \circ \sigma(\mathbf{V}^\top \mathbf{h}_p^j)), \tag{4.4}$$

where $\mathbf{h}_d^i$ is the i-th column of $\mathbf{H}_d^{(3)}$ and $\mathbf{h}_p^j$ is the j-th column of $\mathbf{H}_p^{(3)}$, respectively denoting the i-th and j-th sub-structural representations of drug and protein. Therefore, we can see a bilinear interaction as first mapping representations $\mathbf{h}_d^i$ and $\mathbf{h}_p^j$ to a common feature space with weight matrices $\mathbf{U}$ and $\mathbf{V}$, then learn an interaction on Hadamard product and the weight of vector $\mathbf{q}$. In this way, pairwise interactions provide interpretability on the contribution of sub-structural pairs to the predicted result.

To obtain the joint representation $\mathbf{f}' \in \mathbb{R}^K$, we introduce a bilinear pooling layer over the interaction map $\mathbf{I}$. Specifically, the k-th element of $\mathbf{f}'$ is computed as:

$$\begin{aligned} \mathbf{f}'_k &= \sigma((\mathbf{H}_d^{(3)})^\top \mathbf{U})_k^\top \cdot \mathbf{I} \cdot \sigma((\mathbf{H}_p^{(3)})^\top \mathbf{V})_k \\ &= \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{I}_{i,j} (\mathbf{h}_d^i)^\top (\mathbf{U}_k \mathbf{V}_k^\top) \mathbf{h}_p^j, \end{aligned} \tag{4.5}$$

where $\mathbf{U}_k$ and $\mathbf{V}_k$ denote the k-th column of weight matrices $\mathbf{U}$ and $\mathbf{V}$. Notably, there are no new learnable parameters at this layer. The weight matrices $\mathbf{U}$ and $\mathbf{V}$ are shared with the previous interaction map layer to decrease the number of parameters and alleviate over-fitting. Moreover, we add a sum pooling on the joint representation vector to obtain a compact

feature map:

$$\mathbf{f} = \text{Sumpool}(\mathbf{f}', s), \tag{4.6}$$

where the Sumpool$(\cdot)$ function is a one-dimensional and non-overlapped sum pooling operation with stride s. It reduces the dimensionality of $\mathbf{f}' \in \mathbb{R}^K$ to $\mathbf{f} \in \mathbb{R}^{K/s}$. Furthermore, we can extend the single pairwise interaction to a multi-head form by calculating multiple bilinear interaction maps. The final joint representation vector is a sum of individual heads. As the weight matrices $\mathbf{U}$ and $\mathbf{V}$ are shared, each additional head only adds one new weight vector $\mathbf{q}$, which is parameter-efficient. In our experiments, the multi-head interaction has a better performance than a single one.

Thus, using the novel bilinear attention mechanism, the model can explicitly learn pairwise local interactions between drug and protein. To compute the interaction probability, we feed the joint representation $\mathbf{f}$ into the decoder, which is one fully connected classification layer followed by a sigmoid function:

$$p = \text{Sigmoid}(\mathbf{W}_o\mathbf{f} + \mathbf{b}_o), \tag{4.7}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are learnable weight matrix and bias vector.

Finally, we jointly optimize all learnable parameters by backpropagation. The training objective is to minimize the cross-entropy loss as follows:

$$\mathcal{L} = -\sum_i (y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) + \frac{\lambda}{2} \|\Theta\|_2^2, \tag{4.8}$$

where Θ is the set of all learnable weight matrices and bias vectors above, y_i is the ground-truth label of the i-th drug-target pair, p_i is its output probability by the model, and λ is a hyperparameter for L2 regularization.

4.2.6 Cross-domain Adaptation

Machine learning models tend to perform well on similar data from the same distribution (i.e. in-domain), but poorer on dissimilar data with different distribution (i.e. cross-domain).

It is a key challenge to improve model performance on cross-domain DTI prediction. In our framework, we embed conditional adversarial domain adaptation (CDAN) to enhance generalization from a source domain with sufficient labeled data to a target domain where only unlabeled data is available.

Given a source domain $\mathcal{S}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s}$ of N_s labeled drug-target pairs and a target domain $\mathcal{S}_t = \{x_i^t\}_{j=1}^{N_t}$ of N_t unlabeled drug-target pairs, we leverage CDAN to align their distributions and improve prediction performance across domains. Figure 4.1c shows the CDAN workflow in our framework, including three key components: the feature extractor $F(\cdot)$, the decoder $G(\cdot)$, and the domain discriminator $D(\cdot)$. We use $F(\cdot)$ to denote the separate feature encoders and bilinear attention network together to generate joint representations of input domain data, i.e., $\mathbf{f}_i^s = F(x_i^s)$ and $\mathbf{f}_j^t = F(x_j^t)$. Next, we use the fully connected classification layer mentioned above followed by a softmax function as $G(\cdot)$ to get a classifier prediction $\mathbf{g}_i^s = G(\mathbf{f}_i^s) \in \mathbb{R}^2$ and $\mathbf{g}_j^t = G(\mathbf{f}_j^t) \in \mathbb{R}^2$. Furthermore, we apply a multilinear map to embed joint representation $\mathbf{f}$ and classifier prediction $\mathbf{g}$ into a joint conditional representation $\mathbf{h} \in \mathbb{R}^{2K/s}$, which is defined as the flattening of the outer product of the two vectors:

$$\mathbf{h} = \text{Flatten}(\mathbf{f} \otimes \mathbf{g}), \tag{4.9}$$

where $\otimes$ is the outer product.

The multilinear map captures multiplicative interactions between two independent distributions (Song et al., 2009; Song and Dai, 2013). Following the CDAN mechanism, we simultaneously align the joint representation and predicted classification distributions of source and target domains by conditioning the domain discriminator $D(\cdot)$ on the $\mathbf{h}$. The domain discriminator $D(\cdot)$, consisting of a three-layer fully connected networks, learns to distinguish whether a joint conditional representation $\mathbf{h}$ is derived from the source domain or the target domain. On the other hand, the feature extractor $F(\cdot)$ and decoder $G(\cdot)$ are trained to minimize the source domain cross-entropy loss $\mathcal{L}$ with source label information, and simultaneously generate indistinguishable representation $\mathbf{h}$ to confuse the discriminator

$D(\cdot)$. As a result, we can formulate the two losses in the cross-domain modeling:

$$\mathcal{L}_s(F,G) = \mathbb{E}_{(x_i^s,y_i^s)\sim\mathcal{S}_s}\mathcal{L}(G(F(x_i^s)),y_i^s), \tag{4.10}$$

$$\mathcal{L}_{adv}(F,G,D) = \mathbb{E}_{x_i^t\sim\mathcal{S}_t}\log(1-D(\mathbf{f}_i^t,\mathbf{g}_i^t)) + \mathbb{E}_{x_j^s\sim\mathcal{S}_s} log(D(\mathbf{f}_j^s,\mathbf{g}_j^s)), \tag{4.11}$$

where $\mathcal{L}_s$ is the cross-entropy loss on the labeled source domain and $\mathcal{L}_{adv}$ is the adversarial loss for domain discrimination. The optimization problem is written as a minimax paradigm:

$$\max_{\mathrm{D}}\min_{\mathrm{F,G}}\mathcal{L}_{\mathrm{s}}(\mathrm{F},\mathrm{G}) - \omega\mathcal{L}_{\mathrm{adv}}(\mathrm{F},\mathrm{G},\mathrm{D}), \tag{4.12}$$

where $\omega > 0$ is a hyperparameter to weight $\mathcal{L}_{adv}$. By introducing the adversarial training on $\mathcal{L}_{adv}$, our framework can reduce the data distribution shift between source and target domains, leading to the improved generalization on cross-domain prediction.

4.3 Experiments

4.3.1 Datasets

We evaluate DrugBAN on three public DTI datasets: BindingDB, BioSNAP and Human. The BindingDB dataset is a web-accessible database (Gilson et al., 2016) of experimentally validated binding affinities, focusing primarily on the interactions of small drug-like molecules and proteins. We use a low-bias version of the BindingDB dataset constructed in Section 3.3.1 using the bias-reducing preprocessing steps. The BioSNAP dataset is created from the DrugBank database (Wishart et al., 2008) by Marinka Zitnik and Leskovec (2018), consisting of 4,510 drugs and 2,181 proteins. It is a balanced dataset with validated positive interactions and an equal number of negative samples randomly obtained from unseen pairs. The Human dataset is constructed by Liu et al. (2015), including highly credible negative samples via an *in silico* screening method. Following previous studies (Chen et al., 2020a; Tsubaki et al., 2019; Zheng et al., 2020), we also use the balanced version of Human dataset containing the same number of positive and negative samples. To mitigate the influence of

the hidden data bias (Chen et al., 2020a), we use additional cold pair split for performance evaluation on the Human dataset. Table 4.1 shows statistics of the three datasets.

Table 4.1 Experimental dataset statistics

Dataset	# Drugs	# Proteins	# Interactions
BindingDB (Gilson et al., 2016)	14,643	2,623	49,199
BioSNAP (Huang et al., 2021)	4,510	2,181	27,464
Human (Liu et al., 2015)	2,726	2,001	6,728

4.3.2 Evaluation Strategies and Metrics

We use two different split strategies for in-domain and cross-domain settings. For in-domain evaluation, each experimental dataset is randomly divided into training, validation, and test sets with a 7:1:2 ratio. For cross-domain evaluation, we propose a clustering-based pair split strategy to construct cross-domain scenario. We conduct cross-domain evaluation on the large-scale BindingDB and BioSNAP datasets. For each dataset, we firstly use the single-linkage algorithm to cluster drugs and proteins by ECFP4 (extended connectivity fingerprint, up to four bonds) (Rogers and Hahn, 2010) fingerprint and pseudo amino acid composition (PSC) (Cao et al., 2013), respectively. After that, we randomly select 60% drug clusters and 60% protein clusters from the clustering result, and consider all drug-target pairs between the selected drugs and proteins as source domain data. All the pairs between drugs and proteins in the remaining clusters are considered to be target domain data. Under the clustering-based pair split strategy, the source and target domains are non-overlapping with different distributions. Following the general setting of domain adaptation, we use all labeled source domain data and 80% unlabeled target domain data as the training set, and the remaining 20% labeled target domain data as the test set. The cross-domain evaluation is more challenging than in-domain random split but provides a better measure of model generalization ability in real-world drug discovery.

In the clustering-based pair split, we use the Jaccard distance and cosine distance on ECFP4 and PSC for accurately measuring the pairwise distance. We set the minimum

distance threshold $\gamma = 0.5$ in both drug and protein clusterings since this choice can prevent over-large clusters and be ensure separate dissimilar samples. We obtain 2,780 clusters of drugs and 1,693 clusters of proteins for the BindingDB dataset, and 2,387 clusters of drugs and 1,978 clusters of proteins for the BioSNAP dataset. Table 4.2 shows the number of samples in the ten largest clusters of the clustering results. It shows that BindingDB has a more balanced cluster distribution than BioSNAP in drug clustering. In addition, the protein clustering result tends to generate many small clusters with only a few proteins in both datasets, indicating that the average similarity between proteins is lower than that between drugs. We randomly select 60% drug clusters and 60% protein clusters from clustering result, and regard all associated drug-target pairs with them as source domain data. The associated pairs in the remaining clusters are considered to be source domain data. This split strategy allows quantitatively constructing cross-domain tasks by considering the similarity between drugs or proteins.

Table 4.2 Size of the ten largest clusters in the BindingDB and BioSNAP datasets generated by the clustering-based pair split.

Dataset	Object	# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
BindingDB	Drug	598	460	304	290	253	250	203	202	198	158
BioSNAP	Drug	294	267	75	68	36	35	28	26	24	24
BindingDB	Protein	17	15	15	12	10	10	10	9	9	8
BioSNAP	Protein	8	8	8	6	5	4	4	4	4	4

The AUROC (area under the receiver operating characteristic curve) and AUPRC (area under the precision-call curve) are used as the major metrics to evaluate model classification performance. In addition, we also report the accuracy, sensitivity, and specificity at the threshold of the best F1 score. We conduct five independent runs with different random seeds for each dataset split. The best performing model is selected to be the one with the best AUROC on the validation set. The selected model is then evaluated on the test set to report the performance metrics.

4.3.3 Baselines

We compare DrugBAN with the following five models on DTI prediction: (1) Two shallow machine learning methods, support vector machine (SVM) and random forest (RF) applied on the concatenated fingerprint ECFP4 and PSC features; (2) DeepConv-DTI (Lee et al., 2019) that uses CNN and one global max-pooling layer to extract local patterns in protein sequence and a fully connected network to encode drug fingerprint ECFP4; (3) GraphDTA (Nguyen et al., 2021) that models DTI using graph neural networks to encode drug molecular graph and CNN to encode protein sequence. The learned drug and protein representation vectors are combined with a simple concatenation. To adapt GraphDTA from the original regression task to a binary classification task, we follow the steps in earlier literature (Chen et al., 2020a; Huang et al., 2021) to add a Sigmoid function in its last fully connected layer, and then optimize its parameters with a cross-entropy loss. (4) MolTrans (Huang et al., 2021), a deep learning model adapting transformer architecture to encode drug and protein information, and a CNN-based interactive module to learn sub-structural interaction. For the above deep DTI models, we follow the recommended model hyper-parameter settings described in their original papers.

4.3.4 Implementation Details

DrugBAN is implemented in Python 3.8 and PyTorch 1.7.1 (Paszke et al., 2017), along with functions from DGL 0.7.1 (Wang et al., 2019), DGLlifeSci 0.2.8 (Li et al., 2021), Scikit-learn 1.0.2 (Pedregosa et al., 2011), Numpy 1.20.2 (Harris et al., 2020) and RDKit 2021.03.2 (Greg Landrum et al, 2006). The batch size is set to be 64 and the Adam optimizer is used with a learning rate of 5e-5. We allow the model to run for at most 100 epochs for all datasets. The best performing model is selected at the epoch giving the best AUROC score on the validation set, which is then used to evaluate the final performance on the test set. The protein feature encoder consists of three 1D-CNN layers with the number of filters [128, 128, 128] and kernel sizes [3, 6, 9]. The drug feature encoder consists of three GCN layers with hidden dimensions [128, 128, 128]. The maximum allowed sequence length for protein is set to be

Table 4.3 In-domain performance comparison on the BindingDB and BioSNAP datasets with random split (**Best**, Second Best).

Method	AUROC	AUPRC	Accuracy	Sensitivity	Specificity
		BindingDB			
SVM (Cortes and Vapnik, 1995)	0.939±0.001	0.928±0.002	0.825±0.004	0.781±0.014	0.886±0.012
RF (Ho, 1995)	0.942±0.011	0.921±0.016	0.880±0.012	0.875±0.023	0.892±0.020
DeepConv-DTI (Lee et al., 2019)	0.945±0.002	0.925±0.005	0.882±0.007	0.873±0.018	0.894±0.009
GraphDTA (Nguyen et al., 2021)	0.951±0.002	0.934±0.002	0.888±0.005	0.882±0.012	0.897±0.008
MolTrans (Huang et al., 2021)	0.952±0.002	0.936±0.001	0.887±0.006	0.877±0.016	0.902±0.009
DrugBAN	**0.960±0.001**	**0.948±0.002**	**0.904±0.004**	**0.900±0.008**	**0.908±0.004**
		BioSNAP			
SVM (Cortes and Vapnik, 1995)	0.862±0.007	0.864±0.004	0.777±0.011	0.711±0.042	0.841±0.028
RF (Ho, 1995)	0.860±0.005	0.886±0.005	0.804±0.005	**0.823±0.032**	0.786±0.025
DeepConv-DTI (Lee et al., 2019)	0.886±0.006	0.890±0.006	0.805±0.009	0.760±0.029	0.851±0.013
GraphDTA (Nguyen et al., 2021)	0.887±0.008	0.890±0.007	0.800±0.007	0.745±0.032	**0.854±0.025**
MolTrans (Huang et al., 2021)	0.895±0.004	0.897±0.005	0.825±0.010	0.818±0.031	0.831±0.013
DrugBAN	**0.903±0.005**	**0.902±0.004**	**0.834±0.008**	0.820±0.021	0.847±0.010

1200, and the maximum allowed number of atoms for drug molecule is 290. In the bilinear attention module, we only employ two attention heads to provide better interpretability. The latent embedding size k is set to be 768 and the sum pooling window size s is 3. The number of hidden neurons in the fully connected decoder is 512.

4.3.5 In-domain Performance Comparison

Here we compare DrugBAN with five baselines under the random split setting: support vector machine (SVM), random forest (RF), DeepConv-DTI, GraphDTA, and MolTrans. This is the in-domain scenario so we use vanilla DrugBAN without embedding the CDAN module. Table 4.3 shows the comparison on the BindingDB and BioSNAP datasets. DrugBAN has consistently outperformed baselines in AUROC, AUPRC, and accuracy, while the performance in sensitivity and specificity is also competitive. The deep learning-based methods achieve higher performance than the shallow machine learning methods. The results indicate that data-driven representation learning can capture more important information than pre-defined descriptor features in in-domain DTI prediction. Moreover, DrugBAN can capture interaction patterns via its pairwise interaction module, further improving prediction performance.

Figure 4.2 shows the in-domain results on the Human dataset. Under the random split, the deep learning-based models all achieve similar and promising performance (AUROC $>$

0.98). However, Chen et al. (2020a) pointed out that the Human dataset had some hidden ligand bias, resulting in the correct predictions being made only based on the drug features rather than interaction patterns. The high accuracy could be due to bias and overfitting, not indicating a model's real-world performance on prospective prediction. Therefore, we further use a cold pair split strategy to evaluate models to mitigate the overoptimism of performance estimation under random split due to the data bias. This cold pair split strategy guarantees that all test drugs and proteins are not observed during training so that prediction on test data cannot rely only on the features of known drugs or proteins. We randomly assign 5% and 10% DTI pairs into the validation and test sets respectively, and remove all their associated drugs and proteins from the training set. Figure 4.2 indicates that all models have a significant performance drop from random split to cold pair split, especially for SVM and RF. However, we can see that DrugBAN still achieves the best performance against other state-of-the-art deep learning baselines.

We also conduct experiments to clarify how the proposed framework performs with high fraction of missing data on BindingDB and BioSNAP. Following the missing data setting in MolTrans (Huang et al., 2021), we train DrugBAN and deep learning baselines with only 5%, 10%, 20% and 30% of each dataset, and evaluate predictive performance on the rest of data (90% as test set and 10% as validation set for determining early stopping). Table 4.4 presents the obtained results, showing DrugBAN has the best performance in all settings. In particular, the improvement is more significant on the larger dataset, BindingDB.

4.3.6 Cross-domain Performance Comparison

In-domain classification under random split is an easier task and of less practical importance. Therefore, next, we study more realistic and challenging cross-domain DTI prediction, where training data and test data have different distributions. To imitate this scenario, the original data is divided into source and target domains by the clustering-based pair split. We turn on the CDAN module of DrugBAN to get $\text{DrugBAN}_{\text{CDAN}}$ for studying knowledge transferability in cross-domain prediction.

Table 4.4 AUROC Performance comparison on the BindingDB and BioSNAP datasets with high fraction of missing data (**Best**, Second Best)

Missing (%)	DeepConv-DTI	GraphDTA	MolTrans	DrugBAN
		BindingDB		
95	0.773±0.005	0.831±0.002	0.846±0.004	**0.856±0.003**
90	0.840±0.002	0.867±0.002	0.874±0.003	**0.887±0.004**
80	0.877±0.002	0.897±0.003	0.905±0.001	**0.920±0.003**
70	0.890±0.005	0.916±0.002	0.923±0.001	**0.934±0.001**
		BioSNAP		
95	0.710±0.005	0.768±0.005	0.767±0.006	**0.770±0.008**
90	0.781±0.003	0.798±0.003	0.800±0.004	**0.802±0.003**
80	0.816±0.003	0.829±0.003	0.835±0.001	**0.836±0.002**
70	0.839±0.002	0.851±0.002	0.853±0.002	**0.860±0.003**

Figure 4.3 presents the performance evaluation on the BindingDB and BioSNAP datasets with clustering-based pair split. Compared to the previous in-domain prediction results, the performance of all DTI models drops significantly due to much less information overlap between training and test data. In this scenario, vanilla DrugBAN still outperforms other state-of-the-art models on the whole. Specifically, it outperforms MolTrans by 2.9% and 7.4% in AUROC on the BioSNAP and BindingDB datasets, respectively. The results show that DrugBAN is a robust method under both in-domain and cross-domain settings. Interestingly, RF achieves good performance and even consistently outperforms other deep learning baselines (DeepConv, GraphDTA and MolTrans) on the BindingDB dataset. The results indicate that deep learning methods are not always superior to shallow machine learning methods under the cross-domain setting.

Domain adaptation techniques have received increasing attention due to the ability of transferring knowledge across domains, but they are mainly applied to computer vision and natural language processing problems. We combine vanilla DrugBAN with CDAN to tackle cross-domain DTI prediction. As shown in Figure 4.3, $\text{DrugBAN}_{\text{CDAN}}$ has significant performance improvements with the introduction of a domain adaptation module. On the BioSNAP dataset, it outperforms vanilla DrugBAN by 4.6% and 16.9% in AUROC and AUPRC, respectively. By minimizing the distribution discrepancy across domains, CDAN can effectively enhance DrugBAN generalization ability and provide more reliable results.

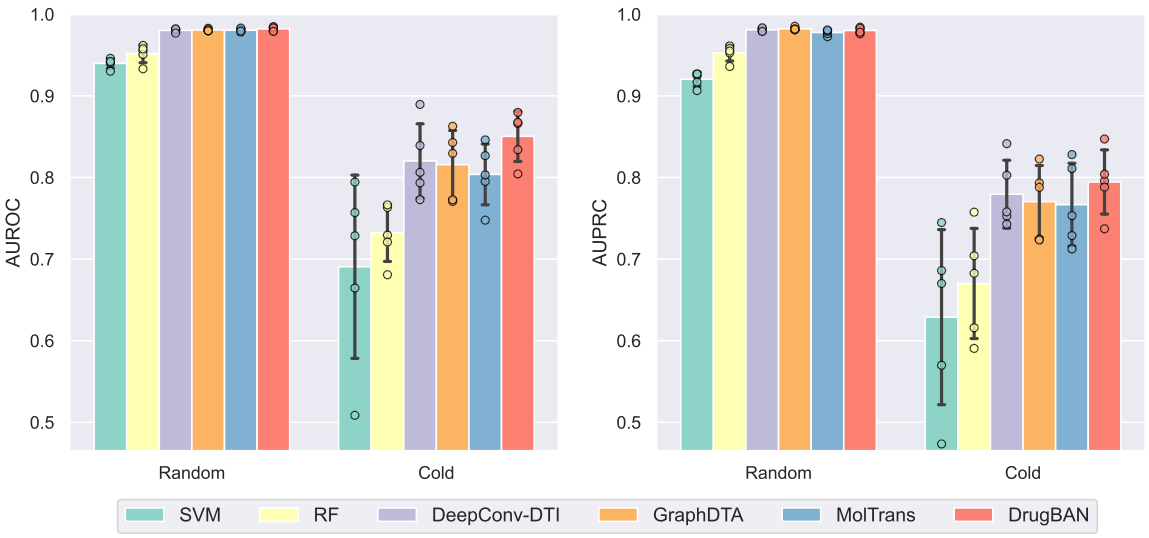

Fig. 4.2 In-domain performance comparison on the Human dataset with random split and cold pair split over five independent runs. Left: AUROC scores. Right: AUPRC scores. The vertical bars represent mean, and the black lines are error bars indicating standard deviation. The dots indicates performance scores in each random run of models.

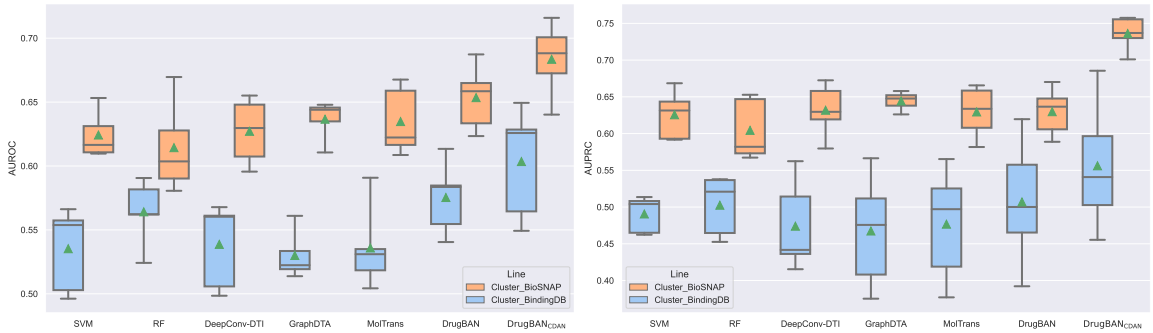

Fig. 4.3 Cross-domain performance comparison on the BindingDB and BioSNAP datasets with clustering-based pair split over five independent runs. Left: AUROC scores. Right: AUPRC scores. The box plots show the median as the center lines, and the mean as the green triangles. The minima and lower percentile represent the worst and second-worst scores. The maxima and upper percentile indicate the best and second-best scores.

These results demonstrate the strength of DrugBAN in generalizing prediction performance across domains.

Table 4.5 Ablation study in AUROC on the BindingDB and BioSNAP datasets with random and clustering-based split strategies. The first four models show the effectiveness of our bilinear attention module, and the last three models show the strength of $\text{DrugBAN}_{\text{CDAN}}$ on cross-domain prediction (**Best**, Second Best).

Ablation tests	$\text{BindingDB}_{\text{random}}$	$\text{BioSNAP}_{\text{random}}$	$\text{BindingDB}_{\text{cluster}}$	$\text{BioSNAP}_{\text{cluster}}$
Linear concatenation (Öztürk et al., 2018)	0.949±0.002	0.887±0.007	-	-
One-side target attention (Tsubaki et al., 2019)	0.950±0.002	0.890±0.005	-	-
One-side drug attention (Tsubaki et al., 2019)	0.953±0.002	0.892±0.004	-	-
DrugBAN	**0.960±0.001**	**0.903±0.005**	0.575±0.025	0.654±0.023
$\text{MolTrans}_{\text{CDAN}}$	-	-	0.575±0.038	0.656±0.028
$\text{DrugBAN}_{\text{DANN}}$	-	-	0.592±0.042	0.667±0.030
$\text{DrugBAN}_{\text{CDAN}}$	-	-	**0.604±0.039**	**0.684±0.026**

4.3.7 Ablation Study

Here we conduct an ablation study to investigate the influences of bilinear attention and domain adaptation modules on DrugBAN. The results are shown in Table 4.5. To validate the effectiveness of bilinear attention, we study three variants of DrugBAN that differ in the joint representation computation between drug and protein: one-side drug attention, one-side protein attention, and linear concatenation. The one-side attention is equivalent to the neural attention mechanism introduced by Tsubaki et al. (2019), which is used to capture the joint representation between a drug vector representation and a protein subsequence matrix representation. We replace the bilinear attention in DrugBAN with one-side attention to generate the two variants. Linear concatenation is a simple vector concatenation of drug and protein vector representations after a max-pooling layer. As shown in the first four rows of Table 4.5, the results demonstrate that bilinear attention is the most effective method to capture interaction information for DTI prediction. To examine the effect of CDAN, we study two variants: DrugBAN with domain-adversarial neural network (DANN) (Ganin et al., 2016) (i.e. $\text{DrugBAN}_{\text{DANN}}$) and MolTrans with CDAN (i.e. $\text{MolTrans}_{\text{CDAN}}$). DANN is another adversarial domain adaptation technique without considering classification distribution. The last four rows of Table 4.5 indicate that $\text{DrugBAN}_{\text{CDAN}}$ still achieves the best performance improvement in cross-domain prediction.

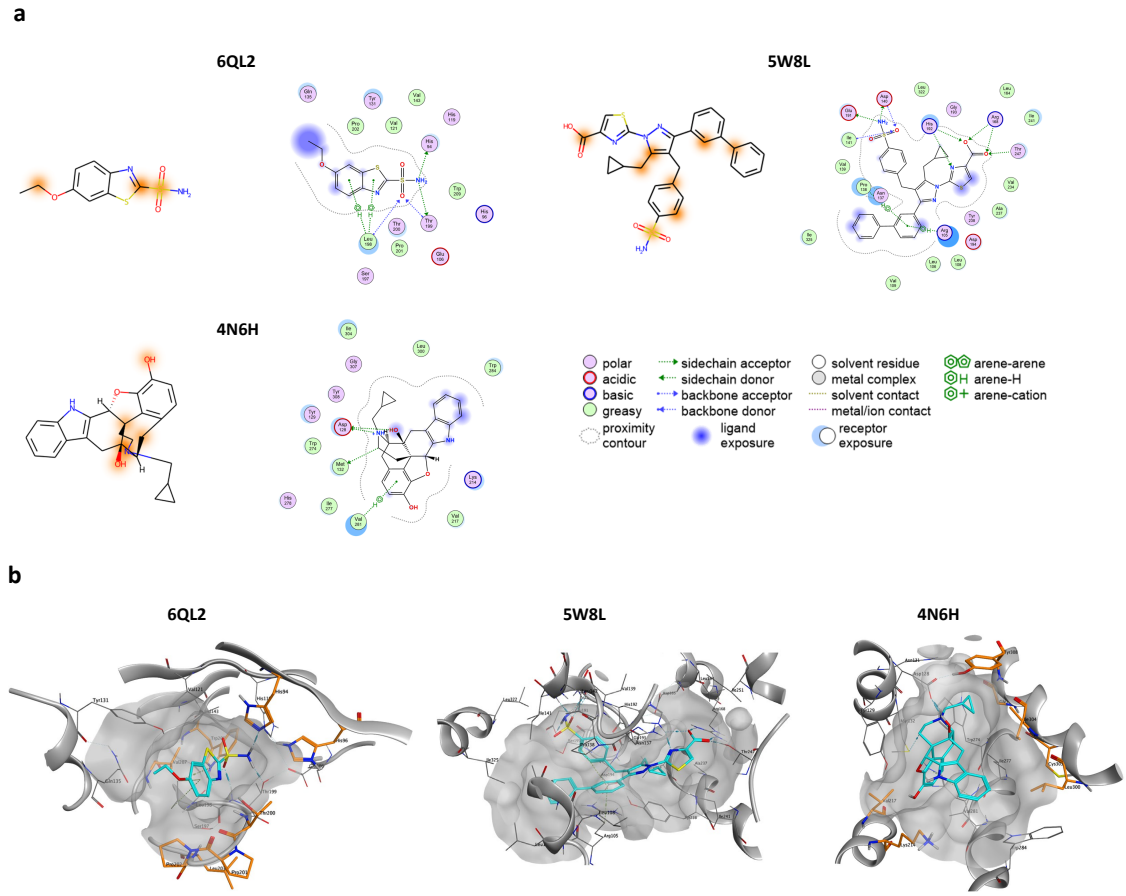

Fig. 4.4 Importance visualization of ligands and binding pockets. **(a)** Interpretability of co-crystalized ligands. The left-hand side of each panel shows the two-dimensional structures of ligands with highlighted atoms (orange) that were predicted to contribute to protein binding. All structures were visualized using RDKit (Greg Landrum et al, 2006). In addition, ligand-protein interaction maps (right-hand side of each panel) from the corresponding crystal structures of these ligands are provided. At the right bottom, the legend panel for the ligand-protein interaction maps is displayed. **(b)** Interpretability of binding pocket structures. The three-dimensional representations of ligand-protein binding pockets are provided highlighting the correctly predicted amino acid residues (orange) that surround the corresponding ligands (cyan). Remaining amino acid residues, secondary structure elements, and surface maps are colored in grey. All ligand-protein interaction maps and three-dimensional representations of X-ray structures were visualized using the Molecular Operating Environment (MOE) software (Molecular Operating Environment (MOE), 2022).

4.3.8 Interpretability with Bilinear Attention Visualization

A further strength of DrugBAN is to enable molecular level insights and interpretation critical for drug design efforts, utilizing the components of the bilinear attention map to visualize

the contribution of each substructure to the final predictive result. Here, we examine the top three predictions (PDB ID: 6QL2, 5W8L and 4N6H) of co-crystalized ligands from Protein Data Bank (PDB) (Burley et al., 2019). Only X-ray structures with resolution greater than 2.5 Å that corresponded to human protein targets were proceeded for selection. In addition, co-crystalized ligands were required to have $pIC_{50} \leq 100$ nM and not to be part of the training set. The visualization results are shown in Figure 4.4a alongside the ligand-protein interaction maps originating from the corresponding X-ray structures. For each molecule, we colored its top 20% weighted atoms in bilinear attention map with orange.

For PDB structure 6QL2 (ethoxzolamide complexed with human carbonic anhydrase 2), our model correctly interpreted sulfonamide region as essential for ligand-protein binding (in 6QL2: sulfonamide oxygen as a hydrogen bond acceptor to the backbone of Leu198 and Thr199, and amino group as a hydrogen bond donor to the side chains of His94 and Thr199). On another hand, ethoxy group of ethoxzolamide was incorrectly predicted to form specific interactions with the protein, although its exposure to the solvent may promote further binding (blue highlight). In addition, benzothiazole scaffold, which forms an arene-H interaction with Leu198, is only partly highlighted by our interpretability model. It is worth mentioning that though top 20% of interacting atoms of ethoxzolamide only corresponded to three highlighted atoms, all of them indicated different ligand-protein interaction sites corroborated by the X-ray structure.

In 5W8L structure (9YA ligand bound to human L-lactate dehydrogenase A), the interpretability feature once more highlighted important interaction patterns for ligand-protein binding. For example, sulfonamide group was once more indicated to form specific interactions with the protein (in 5W8L: amino group as a hydrogen bond donor to the side chains of Asp140 and Glu191, and sulfonamide oxygen as a hydrogen bond acceptor to the backbone of Asp140 and Ile141). Similarly, we noted that carboxylic acid group was also partly highlighted (in 5W8L: carboxylic acid oxygens act as hydrogen bond acceptors to the side chains of Arg168, His192, and Thr247). Moreover, biphenyl rings were correctly predicted to participate in ligand-protein binding (in 5W8L: arene-H interaction with Arg105 and Asn137). Although 9YA (bound to 5W8L) was much larger and complex than ethoxzolamide

(bound to 6QL2), the model showed good interpretability potential for the majority of the experimentally confirmed interactions.

In the third example, 4N6H X-ray complex of human delta-type opioid receptor with EJ4 ligand, main interacting functional groups of EJ4 were once more highlighted correctly. Here, a hydroxyl group of the aliphatic ring complex and a neighboring tertiary amine (in 4N6H: both as hydrogen bond donors to the side chain of Asp128) were correctly interpreted to form specific interactions. On the other hand, phenol group was wrongly predicted to participate in protein binding.

As for the more challenging protein sequence interpretability, the results were overall weaker than those for the ligand interpretability. Although many amino acid residues that were predicted to potentially participate in ligand binding were in fact distantly located to the respective compounds, a number of amino acid residues forming the binding sites were yet correctly predicted, which is shown in Figure 4.4b. For example, in 6QL2 complex the following residues were highlighted: His94, His96, Thr200, Pro201, Pro202, Leu203, Val207, Trp209. Among these, only His94 forms specific interaction with ethoxzolamide. In 5W8L, none of the residues that constitute the ligand-protein binding site were highlighted. However, in 4N6H structure, there were several correctly predicted residues within the binding site: Lys214, Val217, Leu300, Cys303, Ile304, Gly307, and Tyr308. Unfortunately, none of the residues participated in the specific interactions with the ligand. Given these results, it is expected that protein sequence interpretability would be less confident because the one-dimensional protein sequence (used as protein information input in our model) does not necessarily imply the three-dimensional configuration and locality of the binding pocket. However, the results from the primary protein sequence are encouraging enough to safely assume that the further incorporation of three-dimensional protein information into the modeling framework would eventually improve the model interpretability of drug-target interaction networks.

In addition, as the interpretability provided by DrugBAN is adaptively learned from DTI data itself, such interpretation has potential to find some hidden knowledge of local

interactions that has not been explored, and could help drug hunters to improve binding properties of a given scaffold, or to reduce the off-target liabilities of a compound.

4.4 Summary

In this chapter, we present DrugBAN, an end-to-end bilinear attention deep learning framework for DTI prediction. We have integrated CDAN, an adversarial domain adaptation network, into the modeling process to enhance cross-domain generalization ability. Compared with other state-of-the-art DTI models and conventional machine learning models, the experimental results show that DrugBAN consistently achieves improved DTI prediction performance in both in-domain and cross-domain settings. Furthermore, by mapping attention weights to protein subsequences and drug compound atoms, our model can provide biological insights for interpreting the nature of interactions. The proposed ideas are general in nature and can be extended to other interaction prediction problems, such as the prediction of drug-drug interaction and protein-protein interaction.

Chapter 5

Graph Transformer Pre-training Improves Molecular Property Prediction

In Chapter 4, we leveraged domain adaptation to improve model's specific transferability to novel drug-target pairs that are out of learned distribution. However, this paradigm primarily enhance model performance for the identical prediction task across different distributions. It is not intended to improve generic transferability, which involves fine-tuning a pre-trained model for adapting diverse downstream tasks. In drug discovery, a common challenge is to predict various properties of a drug-like molecule. Therefore, this chapter focuses on developing a novel self-supervised learning framework to improve model's specific transferability to downstream molecular property tasks.

5.1 Introduction

Predicting molecular properties plays a crucial role in drug discovery and computational chemistry. However, since only a small fraction of molecules are labeled compared to the giant chemical space, supervised learning methods are prone to over-fitting and poorly generalize to dissimilar data and multiple property tasks (Rong et al., 2020; Wang et al., 2021b). To enhance the generic transferability of molecular representation learning, we develop a self-supervised learning (SSL) framework for molecular property prediction. By

utilizing large-scale 2D and 3D molecules, our framework has the potential to generate more comprehensive molecular representations that can be advantageous for various property prediction tasks.

With the advancement of deep learning, many recent works exploit the SSL paradigm to learn generalizable molecular representations (You et al., 2020; Liu et al., 2022; Stärk et al., 2022; Rong et al., 2020). This involves pre-training a model with unlabeled molecules, and then fine-tuning it for downstream property predictions using limited labeled data. SSL has shown its efficacy in capturing molecular information and improving predictive performance.

In recent years, most molecular SSL methods have utilized graph neural networks (GNNs) for encoding 2D molecular graphs, where atoms are represented as nodes and chemical bonds as edges. Depending on the message-passing mechanism, GNNs can effectively preserve topological information in molecules. One line of these works develops graph contrastive learning-based methods, which adopt data augmentation strategies to generate correlated graph pairs from the same molecule (You et al., 2020; Suresh et al., 2021). However, these works have only considered the 2D modality of molecular data, despite its multimodal nature. More recently, there has been a growing trend towards integrating 3D geometric information into 2D molecular representation learning, and designing new contrastive methods across modalities (Stärk et al., 2022; Zhu et al., 2022a; Liu et al., 2022). The incorporation of 3D geometry can provide rich energy knowledge that is vital in determining molecular functionalities.

Despite the fruitful developments, two challenges remain in 2D-3D molecular SSL: limited model capacity and incomplete pre-training tasks. Firstly, previous studies primarily employ existing 2D and 3D GNNs as backbone networks for pre-training, which have consistently suffered from the over-smoothing problem and expressive power limitation (Xu et al., 2019). These issues limit the model capacity to capture rich multimodal information and learn expressive representations from molecular data. Secondly, most dual-modality molecular pre-training methods only focus on learning inter-modality relation between 2D and 3D molecular structures, without simultaneous consideration of intra-modality relation. The intra-modality relation within each modality is complementary to the inter-modality

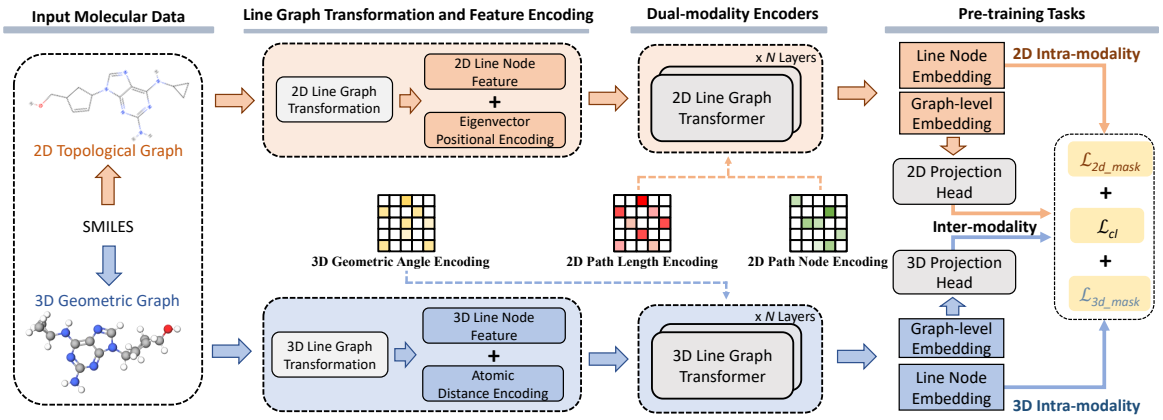

Fig. 5.1 Overview of the Galformer pre-training framework. The input 2D topological graph and 3D geometric graph are derived from the same molecule in the SMILES Weininger (1988a) format. Galformer provides a dual-modality line graph transformer architecture to encode both 2D and 3D molecular information. In each modality, the original molecular graph is first transformed into its line graph. Then the line node feature and positional/distance encoding are derived. Furthermore, the 2D and 3D encoders learn structural information by incorporating 2D path length, 2D path node and 3D geometric angle encodings in their self-attention modules. For pre-training tasks, Galformer creates the inter-modality mask on line node embedding, and the intra-modality contrast on graph-level embedding.

relation (Gao et al., 2019; Wei et al., 2020). For instance, each 2D molecular representation should not only obtain geometric knowledge from its associated 3D modality, but also preserve the topological information within the 2D structure. By considering both inter- and intra-modality knowledge extraction in a unified molecular SSL framework, we can design the pre-trained model to be more robust and generalizable to various downstream tasks.

To tackle the two challenges, we propose the Geometry-aware line graph transformer (Galformer) in this chapter. It is a unified 2D-3D pre-training framework to learn generalizable molecular representations. Figure 5.1 presents an overview of Galformer. We first transform the input 2D and 3D molecular graphs into their line graphs (i.e. edge-adjacency graphs). This transformation preserves the inherent adjacency information in molecules while emphasizing structural information at finer granularity levels, such as node-pair distances and edge-pair angles. Subsequently, we design a dual-modality line graph transformer architecture to model both 2D and 3D molecular information flow simultaneously. To effectively consider the inter- and intra-modality relations in SSL, we design two complementary

pre-training tasks: masked line node prediction and dual-modality contrastive learning. At the intra-modality level, we randomly mask a proportion of 2D and 3D line nodes, and then predict their types from the context embedding within the individual modality. Owing to the property of the line graph, this strategy considers local atom-pair, bond, and angle information being masked, thus capturing richer structural patterns in molecules. At the inter-modality level, we apply contrastive learning to maximize the mutual information between the graph-level 2D and 3D molecular representations. The contrastive pre-training task bridges the modality gap and is complementary to inter-modality mask learning. The pre-trained 2D encoder incorporates rich 3D geometric knowledge to improve downstream property prediction. Our contributions are three-fold:

- We propose a dual-modality line graph transformer architecture to encode both 2D topological and 3D geometric information of molecules. By considering molecular structures as graph inductive bias into the transformer, the designed backbone can extract discriminative knowledge across modalities.
- We design two complementary pre-training tasks: masked line node prediction at the inter-modality level and dual-modality contrastive learning at the intra-modality level. Both contribute to generating more robust and generalizable molecular representations.
- We evaluate Galformer performance on twelve downstream molecular property datasets, including both classification and regression tasks. The experimental results show its superiority over six state-of-the-art baselines.

5.2 Methodology

5.2.1 Dual-Modality Molecular Graphs

Definition 5.2.1. (2D Topological Graph) Given a 2D molecule, its topological graph can be defined as $\mathcal{G}^{2d} = (\mathcal{V}^{2d}, \mathcal{E}^{2d})$, where each node $u \in \mathcal{V}^{2d}$ denotes an atom and each edge $(u, v) \in \mathcal{E}$ represents a chemical bond between node (atom) u and v. Meanwhile, we initialize

the node feature with atom attributes as $\mathbf{h}_u^{2d} \in \mathbb{R}^{\Theta_v}$ and edge feature with chemical bond attributes as $\mathbf{e}_{uv}^{2d} \in \mathbb{R}^{\Theta_e}$, where Θ_v and Θ_e denotes the dimensions of initial node and edge features, respectively.

Definition 5.2.2. (3D Geometric Graph) A geometric graph of 3D molecule can be represented as $\mathcal{G}^{3d} = (\mathcal{V}^{3d}, \mathcal{E}^{3d}, \mathcal{A}^{3d})$, where $(u, v, w) \in \mathcal{A}$ denotes the geometric angle between edge (u, v) and (v, w). Following previous studies (Li et al., 2022b; Fang et al., 2022), we consider the atomic distance $d_{uv} > 0$ and angle $\theta_{uvw} \in [0, \pi]$ as invariant spatial features in $\mathcal{G}^{3d}$, regardless of how the same molecular conformation rotates or translates in 3D space. To emphasize the geometric information, each 3D node feature is denoted as a one-hot vector $\mathbf{h}_u^{3d} \in \mathbb{R}^{\Theta_t}$, where Θ_t is the number of atomic types without additional topological attributes.

The dual-modality molecular graphs capture high-level topological and geometric relationships, which are significant to the functionality of molecules. For each atom in a given molecule, we create a 101-dimensional one-hot vector to represent the atom type, and 36-dimensional integer vector to indicate the atom properties. These atom properties includes formal charge, number of hydrogen atoms, presence of a chiral center, chirality type, number of radical electrons, hybridization state, bond degree, atomic mass, and aromaticity. We concatenate the atom type vector and atom property vector to create the initial node feature in a 2D topological graph. Meanwhile, we utilize the individual atom type vector as the initial node feature in a 3D geometric graph.

For each bond in a given molecule, we generate a 5-dimensional one-hot vector to encode the bond type, and a 9-dimensional integer vector to indicate the bond properties, including ring membership, stereochemistry, and conjugation status. The two vectors are concatenated as the initial edge feature in a 2D topological graph. For a 3D geometric graph, we calculate the pairwise atomic distance and represent it as a vector using a Gaussian Basis Kernel function, which serves as the initial edge feature.

5.2.2 Problem Formulation

Given a set of unlabeled molecules $\mathcal{M} = \{\mathcal{G}_i^{2d}, \mathcal{G}_i^{3d}\}_{i=1}^{|\mathcal{M}|}$, where each molecule $M_i \in \mathcal{M}$ has its 2D topological graph $\mathcal{G}_i^{2d}$ and 3D geometric graph $\mathcal{G}_i^{3d}$. The study aims to pre-train a dual-modality SSL model that generates discriminative molecular representations and then adapts to molecular property prediction by fine-tuning. Due to the scarcity of the 3D conformers in downstream tasks, following the paradigm of previous works (Liu et al., 2022; Stärk et al., 2022), we regard the knowledge of 3D geometry as privileged information only used during pre-training. Then, the pre-trained 2D encoder is subsequently fine-tuned for downstream property prediction tasks, with only 2D topological graphs available.

5.2.3 Framework Overview

Figure 5.1 presents an overview of the Galformer framework, which consists of three key components: line graph transformation, dual-modality line graph transformer, and complementary pre-training tasks. In this methodology, we first describe the procedure for transforming 2D and 3D molecular graphs into their corresponding line graphs. Then we elaborate on the design of dual-modality line graph transformers that can effectively encode both topological and geometric information. Finally, we introduce the pre-training tasks that enable learning of inter-modality and intra-modality relations.

5.2.4 Molecular Line Graphs

Topological Line Graph. Given a 2D topological graph $\mathcal{G}^{2d} = (\mathcal{V}^{2d}, \mathcal{E}^{2d})$, we can transform it into a topological line graph $\hat{\mathcal{G}}^{2d} = (\hat{\mathcal{V}}^{2d}, \hat{\mathcal{E}}^{2d})$ following the two steps. (i) For each edge in $\mathcal{G}^{2d}$, create a node in $\hat{\mathcal{G}}^{2d}$. This step involves mapping the edges in the original graph to nodes in the line graph. (ii) For every pair of edges in $\mathcal{G}^{2d}$ that share a common node, create an edge between their corresponding nodes in $\hat{\mathcal{G}}^{2d}$. This step involves making connections between the nodes in the line graph based on the common nodes in the original graph. After the transformation, we can initialize each node feature in $\hat{\mathcal{G}}^{2d}$ with the atom and bond attributes

as follows:

$$\mathbf{n}_{uv}^{2d} = \text{Concat}(\mathbf{W}_g \mathbf{h}_u^{2d} + \mathbf{W}_g \mathbf{h}_v^{2d}, \mathbf{W}_e \mathbf{e}_{uv}^{2d}), \tag{5.1}$$

where $\mathbf{W}_g \in \mathbb{R}^{\frac{\Theta_{\hat{v}}}{2} \times \Theta_v}$ and $\mathbf{W}_e \in \mathbb{R}^{\frac{\Theta_{\hat{v}}}{2} \times \Theta_e}$ are learnable projection matrices. To avoid ambiguity and simplify notations, we denote $\hat{\mathcal{V}}^{2d} = \{\hat{v}_i\}_{i=1}^{|\hat{\mathcal{V}}^{2d}|}$ as the set of nodes in topological line graph $\hat{\mathcal{G}}^{2d}$, where $|\hat{V}^{2d}|$ is equivalent to the number of edges $|\mathcal{E}^{2d}|$ in $\mathcal{G}^{2d}$ by the definition of line graph.

Geometric Line Graph. A geometric line graph $\hat{\mathcal{G}}^{3d} = (\hat{\mathcal{V}}^{3d}, \hat{\mathcal{E}}^{3d})$ can be derived from its corresponding 3D geometric graph $\mathcal{G}^{3d}$ by the similar transformation steps above. In contrast, we emphasize integrating the invariant spatial features (i.e., atomic distance and angle) within the line graph framework. As the atomic distance d_{uv} and angle θ_{uvw} are continuous scalar variables, it is difficult to directly incorporate them into the subsequent deep architecture. Therefore, we use Gaussian Basis Kernel function (Scholkopf et al., 1997) to map the atomic distance and angle into high-dimensional vectors, which can be written as:

$$\mathbf{d}_{uv} = \overset{M}{\underset{m=1}{\frown}} \left(-\frac{1}{\sqrt{2\pi}|\sigma_d^m|} \exp\left(-\frac{1}{2} \left(\frac{\alpha_d^t d_{uv} + \beta_d^t - \mu_d^m}{|\sigma_d^m|} \right)^2 \right) \right), \tag{5.2}$$

$$\theta_{uvw} = \overset{M}{\underset{m=1}{\frown}} \left(-\frac{1}{\sqrt{2\pi}|\sigma_\theta^m|} \exp\left(-\frac{1}{2} \left(\frac{\alpha_\theta^t \theta_{uvw} + \beta_\theta^t - \mu_\theta^m}{|\sigma_\theta^m|} \right)^2 \right) \right), \tag{5.3}$$

where $\frown$ denotes the concatenation operator over multiple scalars and M is the number of Gaussian Basis Kernels, that is, the dimension of the mapped vectors. (σ_d^m, μ_d^m) and (σ_θ^m, μ_θ^m) are the k-th learnable kernel center and scaling factor for the atomic distance and angle, respectively. (α_d^t, β_d^t) and (α_θ^t, β_θ^t) are also learnable scalars but indexed by the type of atomic pair in (u, v) and edge pair in (u, v, w), respectively. The mapped pair-wise encodings can preserve invariant spatial features and adapt to further deep modeling. Next, the node and edge features in geometric line graph $\hat{G}^{3d}$ are defined as follows:

$$\mathbf{n}_{uv}^{3d} = \text{Concat}(\mathbf{W}_l \mathbf{h}_u^{3d} + \mathbf{W}_l \mathbf{h}_v^{3d}, \mathbf{W}_d \mathbf{d}_{uv}), \tag{5.4}$$

$$\mathbf{e}_{uvw}^{3d} = \mathbf{W}_\mathbf{p} \theta_{\mathbf{uvw}}, \tag{5.5}$$

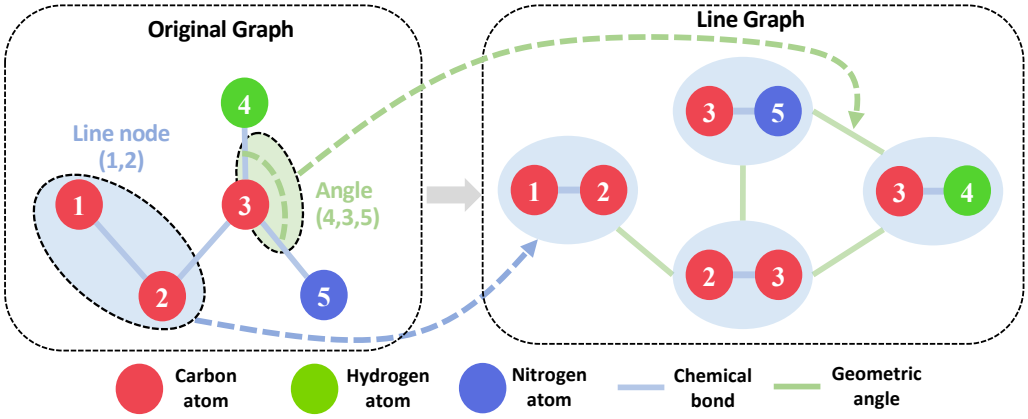

Fig. 5.2 An illustration of transforming a molecular graph to its line graph. This transformation preserves the inherent adjacency information in the original graph while emphasizing structural information at finer granularity levels.

where $\mathbf{W}_l \in \mathbb{R}^{\frac{\Theta_{\hat{v}}}{2} \times \Theta_t}$, $\mathbf{W}_d \in \mathbb{R}^{\frac{\Theta_{\hat{v}}}{2} \times M}$ and $\mathbf{W}_p \in \mathbb{R}^{\Theta_{\hat{e}} \times M}$ are learnable projection matrices. By definition, the edges in the line graph represent the pairs of edges in the original graph sharing a common node (i.e. adjacent edges). The relationship derives that each line edge corresponds to a unique angle in the original graph. Therefore, we can adopt the projected angle vector $\mathbf{e}^{3d}_{uvw}$ as edge feature in $\hat{G}^{3d}$. Finally, we can define $\hat{\mathcal{V}}^{3d} = \{\hat{v}_i\}_{i=1}^{|\hat{\mathcal{V}}^{3d}|}$ as the set of nodes and $\hat{\mathcal{E}}^{3d} = \{\hat{e}_{ij}\}_{i,j \in [1,|\hat{\mathcal{V}}^{3d}|]}$ as the set of edges in $\hat{\mathcal{G}}^{3d}$. Figure 5.2 illustrates the line graph transformation.

5.2.5 Dual-modality Line Graph Transformers

Although transformer architecture has demonstrated remarkable performance in various tasks involving sequence data, its direct application to graph-structured data still remains challenging. The primary obstacle lies in the fact that the self-attention and feed-forward network modules are order-agnostic to the input features, and the classic positional encoding (Vaswani et al., 2017) fails to capture graph structural information. To handle this problem, we develop dual-modality line graph transformers, which introduce efficient structural encoding methods to incorporate the topological and geometric structures, respectively.

2D Line Graph Transformer. We introduce three distinct structural encoding methods: eigenvector positional encoding, path length encoding, and path node encoding, thus inte-

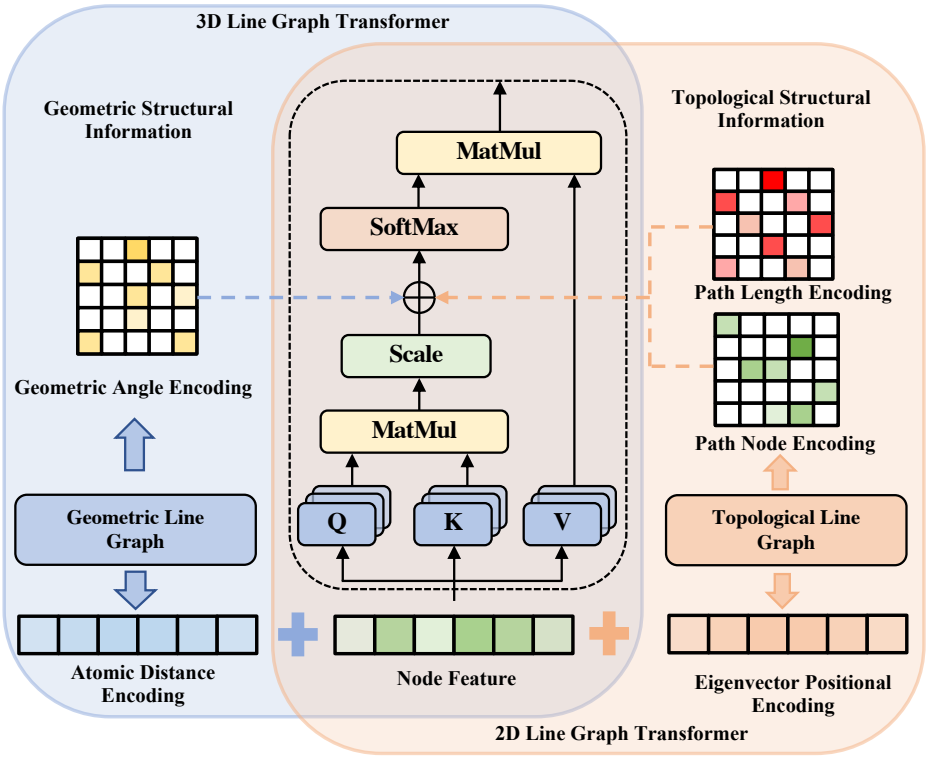

Fig. 5.3 An illustration of the dual-modality line graph transformer architecture. The blue part is the 3D line graph transformer capturing the geometric structural information, and the orange part is the 2D line graph transformer incorporating the topological structural information. Both of them are derived from the vanilla transformer backbone.

grating the topological information of 2D line graphs into transformer architecture. These encoding methods serve as the inductive bias to convey structural information, leading to more robust and discriminative graph representations.

Eigenvector Positional Encoding. For the topological line graph, we use the eigenvectors of its normalized Laplacian matrix as positional encoding. Laplacian matrix is defined as the difference between the degree matrix and the adjacency matrix, representing the topology of a graph. Furthermore, Laplacian eigenvectors can distinguish the local position information of different nodes, while preserving the global topology structure (Dwivedi et al., 2023; Belkin and Niyogi, 2003). For each node in the topology line graph, its corresponding eigenvector is viewed as a node positional encoding and added to the node feature for position-aware

feature input:

$$\mathbf{x}_i^{2d} = \mathbf{n}_i^{2d} + \mathbf{W}_v \mathbf{v}_i^{2d}, \tag{5.6}$$

where $\mathbf{v}_i^{2d} \in \mathbb{R}^K$ is the K smallest non-trivial eigenvectors for the i-th node in the topological line graph, and $\mathbf{W}_v \in \mathbb{R}^{\Theta_v \times K}$ denotes a learnable projection matrix. The eigenvector positional encoding for graph-structured data can be regarded as a natural generalization of the sine-cosine positional encoding for sequence-structured data. By adding it to the input node features, the transformer can effectively capture their positional information in the graph.

Path Length Encoding. Different from general structured data in Euclidean space, e.g., texts and images, graphs do not have a canonical order and can only lie in a non-Euclidean space connected by edges. To model full structural information, the spatial relations between nodes should be measured. Following previous studies (Ying et al., 2021; Li et al., 2022a), we derive the shortest path between each pair of nodes, and then leverage the path length and the node features along the path to capture spatial information. For the path length encoding, we assign a learnable embedding vector for each length scalar, then project it as a bias term in the self-attention module. Given a node pair $(\hat{v}_i, \hat{v}_j)$ in the topological line graph, the path length encoding can be written as:

$$\begin{aligned} \mathbf{p}_{ij} &= \mathrm{Spl}(\hat{v}_i, \hat{v}_j), \\ b_{ij} &= \mathbf{p}_{ij}^T \mathbf{w}_{\mathbf{p}}, \end{aligned} \tag{5.7}$$

where $\mathrm{Spl}(\cdot)$ denotes an embedding function that determines a learnable embedding vector $\mathbf{p}_{ij} \in \mathbb{R}^p$ indexed by the length of the shortest path between node $\hat{v}_i$ and $\hat{v}_j$, and $\mathbf{w}_{\mathbf{p}} \in \mathbb{R}^p$ is a learnable projection vector.

Path Node Encoding. As the different nodes along the shortest path can distinguish spatial relations in graphs, we encode the path node features as additional structural information into modeling. For each node pair $(\hat{v}_i, \hat{v}_j)$, we compute an average bias term of the dot-products of the node feature and a learnable projection vector along the path, which is defined as:

$$c_{ij} = \frac{1}{N_{ij}} \sum_{n=1}^{N_{ij}} (\mathbf{x}_n^{2d})^T \mathbf{w}_n, \tag{5.8}$$

where N_{ij} is the shortest path length between node $\hat{v}_i$ and $\hat{v}_j$, $\mathbf{x}_n^{2d}$ is the n-th 2D node feature along the path, and $\mathbf{w}_n \in \mathbb{R}^{\Theta_{\hat{v}}}$ denotes the n-th learnable projection vector indexed by the path position.

Topological Bias in 2D Attention. We incorporate the proposed three structural encodings into the transformer architecture, which captures full structural information in the topological line graph. Specifically, the eigenvector positional encoding is added to the original node feature as the input feature. The path length and path node encodings are viewed as two structural bias terms to the self-attention module. We can compute each element (i, j) in the attention matrix as follows:

$$\mathbf{A}_{ij}^{2d} = \frac{(\mathbf{W}_q^{2d}\mathbf{x}_i^{2d})^T(\mathbf{W}_k^{2d}\mathbf{x}_j^{2d})}{\sqrt{d_k}} + b_{ij} + c_{ij}, \tag{5.9}$$

where $\mathbf{W}_q^{2d} \in \mathbb{R}^{d_k \times \Theta_{\hat{v}}}$ and $\mathbf{W}_k^{2d} \in \mathbb{R}^{d_k \times \Theta_{\hat{v}}}$ are learnable projection matrices. Similar to the vanilla transformer, we augment the self-attention module by extending to the multi-head mechanism. Note that the structural bias terms b_{ij} and c_{ij} should also be computed by separate projection vectors within different heads. Then, we adapt the feed-forward network with layer normalization (Ba et al., 2016) and skip connection operations (He et al., 2016) over the output of the self-attention module.

3D Line Graph Transformer. To incorporate the geometric graph information into transformer architecture, we consider two novel structural encoding methods: atomic distance encoding and geometric angle encoding. By leveraging the constructed geometric line graph, the atomic distance and geometric angle can be embedded into the node and edge representations in a natural manner. For the atomic distance encoding, we concatenate the atomic feature embedding and pair-wise distance embedding in Equation (5.4), which provides a relative position encoding as the input feature to the following modeling, that is, $\mathbf{x}_i^{3d} = \mathbf{h}_{uv}^{3d}$.

Geometric Angle Encoding. As an invariant spatial feature, the geometric angle between the adjacent bonds plays an important role in determining the spatial structure of 3D molecules. Similarly, we derive the shortest path between any two nodes in the geometric

line graph. Since the angle embedding is viewed as the line edge feature by Equation (5.5), we can encode the edge features along the path as the structural information. Given a node pair $(\hat{v}_i, \hat{v}_j)$, an average bias term is computed by taking the dot-product of the edge feature and a learnable projection vector along the shortest path:

$$g_{ij} = \frac{1}{N_{ij}-1} \sum_{m=1}^{N_{ij}-1} (\mathbf{e}_m^{3d})^T \mathbf{w}_m, \tag{5.10}$$

where N_{ij} is the shortest path length, $\mathbf{e}_m^{3d}$ is the m-th 3D edge feature along the path, and $\mathbf{w}_m \in \mathbb{R}^{\Theta_{\hat{e}}}$ denotes the m-th learnable projection vector indexed by the path position.

Geometric Bias in 3D Attention. Next, we introduce the pair-wise distance and angle encodings into transformer architecture, capturing the geometric structural information. For each element (i, j) in the attention matrix of 3D line graph transformer, we have:

$$\mathbf{A}_{ij}^{3d} = \frac{(\mathbf{W}_q^{3d}\mathbf{x}_i^{3d})^T(\mathbf{W}_k^{3d}\mathbf{x}_j^{3d})}{\sqrt{d_k}} + g_{ij}, \tag{5.11}$$

where $\mathbf{W}_q^{3d} \in \mathbb{R}^{d_k \times \Theta_{\hat{v}}}$ and $\mathbf{W}_k^{3d} \in \mathbb{R}^{d_k \times \Theta_{\hat{v}}}$ are learnable projection matrices. As with the 2D line graph transformer, we leverage the multi-head attention mechanism to enhance the self-attention module, then feed the output into a feed-forward network with layer normalization and skip connection operations.

Architecture Advantages. Here we discuss two advantages of the designed dual-modality architecture: (i) Both 2D and 3D line graph transformers are derived from the vanilla transformer backbone but encode different topological and geometric information, respectively. The consistent backbone across two modalities ensures that the encoded representations are directly comparable, and the knowledge can be transferred in the subsequent inter-modality learning. (ii) By transforming original molecular graphs into line graphs, the structural information is preserved in the node and edge features. This transformation ensures that graph structures can be more effectively captured by most graph-based machine learning methods. Despite different terminologies, the concept of molecular line graph has already been applied in previous studies. However, they separate the line graph into atom-bond and

bond-angle two graphs (Fang et al., 2022; Li et al., 2022b), or consider the transformation only on 2D molecules (Li et al., 2022a)

5.2.6 Pre-training Task Construction

The efficiency of the pre-training framework significantly depends on the construction of pre-training tasks. In this work, we design two complementary tasks from both intra-modality and inter-modality levels: masked line node prediction and dual-modality contrastive learning.

Masked Line Node Prediction. For the pre-training at the intra-modality level, we leverage node masking to let the model learn the regularities of the atom/bond attributes. Following the "masked language model (MLM)" training objective in BERT (Devlin et al., 2019), we randomly sample a proportion of line nodes in the topological and geometric line graphs, and replace them with the masking procedures: (i) masking 80% of the selected nodes using a special *MASK* indicator, (ii) replacing 10% of the selected nodes with other random nodes, and (iii) keeping the remaining 10% nodes unchanged. Then we apply 2D/3D line graph transformers to obtain the corresponding line node embeddings. Finally, a multi-layer perceptron (MLP) is used on top of the masked embeddings to predict the types of the original nodes with a cross-entropy loss. The mask losses on the 2D and 3D modalities are denoted as $\mathcal{L}^{2d}_{mask}$ and $\mathcal{L}^{3d}_{mask}$, respectively.

Dual-modality Contrastive Learning. As the 2D and 3D modalities of the same molecule are correlated and can provide complementary knowledge to each other, we can enhance molecular self-supervised learning by designing an inter-modality pre-training task. Specifically, we leverage contrastive learning to maximize the mutual information between the 2D and 3D representations of a molecule. In each line graph, we create a virtual node connected to all other nodes. Its output represents the graph-level embedding and can be used for inter-modality contrast. For each molecule, we first extract the graph-level embeddings from the masked topological and geometric line graphs, and then adapt the individual projection heads to map them into a common representation space, resulting in the derivation of $\mathbf{z}^{2d}_i$ and $\mathbf{z}^{3d}_i$. The 2D-3D latent vectors from the same molecule are regarded as positive pairs, and negative pairs otherwise. Based on the InfoNCE loss (Oord et al., 2018),

our dual-modality contrastive learning is written as:

$$\mathcal{L}_{cl}^{2d} = -\sum_{i=1}^{N} \log \frac{\exp(\text{Sim}(\mathbf{z}_i^{2d}, \mathbf{z}_i^{3d})/\tau)}{\sum_{j=1}^{M} \exp(\text{Sim}(\mathbf{z}_i^{2d}, \mathbf{z}_j^{3d})/\tau)}, \tag{5.12}$$

$$\mathcal{L}_{cl}^{3d} = -\sum_{i=1}^{N} \log \frac{\exp(\text{Sim}(\mathbf{z}_i^{3d}, \mathbf{z}_i^{2d})/\tau)}{\sum_{j=1}^{M} \exp(\text{Sim}(\mathbf{z}_i^{3d}, \mathbf{z}_j^{2d})/\tau)}, \tag{5.13}$$

$$\mathcal{L}_{cl} = \frac{1}{N}(\mathcal{L}_{cl}^{2d} + \mathcal{L}_{cl}^{3d}), \tag{5.14}$$

where $\mathbf{z}_i^{2d}$ and $\mathbf{z}_i^{3d}$ denotes two latent vectors in 2D and 3D modalities from the same molecule, τ is a temperature coefficient, $\text{Sim}(\cdot)$ measures the dot similarity between two vectors, M is the batch size, and N is the number of molecules in the dataset. Dual-modality contrastive learning can simultaneously align the positive pairs of the same molecule and distinguish them from negative pairs, thereby enabling the 2D/3D representation to extract complementary information from its 3D/2D counterpart.

Finally, we combine the intra-modality mask losses and inter-modality contrastive loss into an overall loss function as follows:

$$\mathcal{L}_{obj} = \lambda_1 \left(\mathcal{L}_{mask}^{2d} + \mathcal{L}_{mask}^{3d} \right) + \lambda_2 \mathcal{L}_{cl}, \tag{5.15}$$

where λ_1 and λ_2 are weighting coefficients. The training objective is to minimize the overall loss $\mathcal{L}_{obj}$. As a result, the mask generative losses capture useful semantics of molecules in two modalities, while the contrastive loss can simultaneously provide complementary information from each other. Consequently, our SSL framework can produce a robust and discriminative molecular representation that encodes both topological and geometric information.

Table 5.1 Statistics of Molecular Property Datasets

Dataset	# Molecules	# Tasks	Source	Type
BBBP	2,039	1	MoleculeNet	Classification
Sider	1,427	27	MoleculeNet	Classification
ClinTox	1,478	2	MoleculeNet	Classification
BACE	1,513	1	MoleculeNet	Classification
Tox21	7,831	12	MoleculeNet	Classification
MUV	93,087	17	MoleculeNet	Classification
HIV	41,127	1	MoleculeNet	Classification
Estrogen	3,122	2	ChEMBL	Classification
MetStab	2,267	2	MetStabOn	Classification
ESOL	1,128	1	MoleculeNet	Regression
Freesolv	642	1	MoleculeNet	Regression
Lipo	4,200	1	MoleculeNet	Regression
Malaria	9,999	1	Antimalarial	Regression
CEP	29,978	1	Clean Energy Project	Regression

5.3 Experiments

5.3.1 Datasets

For pre-training, we use unlabeled GEOM dataset (Axelrod and Gomez-Bombarelli, 2022) comprising 304k molecules, which contains precise 2D and 3D structures. Most molecules have multiple 3D conformers, but the conformer with the lowest energy is deemed to be the most stable and has the highest possibility. Therefore, we take the lowest-energy conformer of each molecule as its 3D structure. For downstream fine-tuning, we evaluate model performance on fourteen labeled 2D molecular datasets from MoleculeNet Wu et al. (2018a) and other public sources (Gaulton et al., 2012; Podlewska and Kafel, 2018; Gamo et al., 2010; Hachmann et al., 2011), including nine classification tasks and five regression tasks. These benchmarks have been widely used in previous studies (You et al., 2020; Li et al., 2022a; Wang et al., 2021b; Liu et al., 2022) and provide comprehensive coverage of molecular properties in domains, such as physiology, biophysics and physical chemistry. The statistics of these datasets are summarized in Table 5.1. For detailed descriptions of molecular properties, please refer to Appendix B.1.

5.3.2 Evaluation Strategies and Metrics

As described in previous chapters, it is a common practice to divide a dataset into training, validation and test sets using random split for machine learning tasks. However, this split strategy fails to distinguish molecules with similar structures. As suggested in MoleculeNet (Wu et al., 2018a), we adopt scaffold split to create a more challenging yet realistic evaluation for molecular property prediction. Each downstream dataset is divided into training/validation/test with an 8:1:1 ratio.

We use two metrics to evaluate the performance of downstream tasks: the area under the receiver operating characteristic curve (AUROC) for classification tasks and RMSE for regression tasks. For classification datasets with multi-binary labels, we calculate the average AUROC of all independent labels as the final metric. For each downstream task, we conduct three independent scaffold splitting runs with different random seeds, and report the means and standard deviations of evaluation metrics.

5.3.3 Baselines

We comprehensively evaluate the performance of Galformer against seven state-of-the-art self-supervised learning methods on molecules. Among these, GraphCL (You et al., 2020) and JOAO (You et al., 2021) are 2D contrastive methods that leverage data augmentation strategies on 2D molecular graphs to facilitate representation learning. GROVER Rong et al. (2020) and KPGT (Li et al., 2022a) are 2D generative methods that incorporate domain knowledge for the design of self-supervised learning tasks. DMP (Zhu et al., 2023) effectively combines the 1D sequence and 2D graph of a molecule to learn dual-modality representations. 3DInformax (Stärk et al., 2022) and GraphMVP (Liu et al., 2022) are 2D-3D pre-training methods with GNN-based backbones. 3DInformax aims to maximize the mutual information between the 2D and 3D representations via contrastive learning, while GraphMVP adopts a hybrid strategy to learn complementary 2D-3D information.

Table 5.2 Test AUROC performance of different methods on nine downstream classification datasets (**best in bold**, second best in underline). A larger value indicates better performance (marked by ↑). Each experiment is conducted independently three times with different random seeds for scaffold split.

	Classification accuracy (AUROC) ↑								
Dataset	BBBP	Sider	ClinTox	BACE	Tox21	MUV	HIV	Estrogen	MetStab
# Molecules	2,039	1,427	1,478	1,513	7,831	93,087	41,127	3,122	2,267
# Tasks	1	27	2	1	12	17	1	2	2
GraphCL	$0.901_{(0.033)}$	$0.625_{(0.023)}$	$0.642_{(0.120)}$	$0.868_{(0.025)}$	$0.823_{(0.015)}$	$0.762_{(0.027)}$	$0.776_{(0.007)}$	$0.891_{(0.020)}$	$0.794_{(0.037)}$
JOAO	$0.900_{(0.031)}$	$0.629_{(0.020)}$	$0.707_{(0.068)}$	$0.866_{(0.035)}$	$0.825_{(0.011)}$	$0.766_{(0.048)}$	$0.769_{(0.004)}$	$0.869_{(0.050)}$	$0.814_{(0.031)}$
GROVER	$0.923_{(0.029)}$	$0.645_{(0.010)}$	$0.894_{(0.032)}$	$0.882_{(0.030)}$	$0.840_{(0.016)}$	$0.831_{(0.035)}$	$0.773_{(0.014)}$	$0.903_{(0.030)}$	$0.822_{(0.039)}$
3DInfomax	$0.905_{(0.033)}$	$0.634_{(0.029)}$	$0.724_{(0.104)}$	$0.862_{(0.023)}$	$0.819_{(0.021)}$	$0.803_{(0.007)}$	$0.750_{(0.010)}$	$0.871_{(0.039)}$	$0.811_{(0.043)}$
GraphMVP	$0.918_{(0.019)}$	$0.652_{(0.027)}$	$0.705_{(0.092)}$	$0.866_{(0.028)}$	$0.832_{(0.017)}$	$0.787_{(0.039)}$	$0.791_{(0.004)}$	$0.892_{(0.033)}$	$0.827_{(0.043)}$
DMP	$\underline{0.930}_{(0.031)}$	$0.655_{(0.025)}$	$0.908_{(0.032)}$	$0.890_{(0.033)}$	$0.830_{(0.022)}$	$\underline{0.835}_{(0.020)}$	$\mathbf{0.798}_{(0.015)}$	$0.910_{(0.025)}$	$\underline{0.853}_{(0.046)}$
KPGT	$0.927_{(0.028)}$	$\underline{0.658}_{(0.013)}$	$\mathbf{0.915}_{(0.027)}$	$\underline{0.893}_{(0.032)}$	$\underline{0.847}_{(0.013)}$	$0.829_{(0.047)}$	$0.768_{(0.007)}$	$\underline{0.915}_{(0.028)}$	$0.846_{(0.052)}$
Galformer	$\mathbf{0.933}_{(0.027)}$	$\mathbf{0.681}_{(0.011)}$	$\underline{0.910}_{(0.030)}$	$\mathbf{0.897}_{(0.026)}$	$\mathbf{0.852}_{(0.015)}$	$\mathbf{0.841}_{(0.023)}$	$\underline{0.796}_{(0.017)}$	$\mathbf{0.936}_{(0.010)}$	$\mathbf{0.875}_{(0.029)}$

5.3.4 Implementation Details

We implement Galformer mainly in PyTorch (Paszke et al., 2017) and DGL (Wang et al., 2019), along with useful functions from RDKit (Greg Landrum et al, 2006), DGLlifeSci (Li et al., 2021) and Scikit-learn (Pedregosa et al., 2011). The Adam optimizer (Kingma and Ba, 2014) with a polynomial decay learning rate scheduler is used for training optimization. We employ 12-layer 2D and 3D line graph transformers as backbone networks to encode the dual-modality molecular information. The hidden size is set to 768 and the number of attention heads is set to 12. We select a batch size of 256 and pre-train the model for 50 epochs. The peak learning rate is set to $2e^{-4}$ and Adam weight decay is set to $1e^{-6}$. Additionally, we set the mask ratio as 0.4 and temperature coefficient τ as 0.1. The multi-task weighting coefficients λ_1 and λ_2 are all set to 1 for balanced optimization. After pre-training, we concatenate the graph-level embedding and averaged line node embedding as the molecular representation for downstream tasks.

5.3.5 Downstream Task Evaluation

Tables 5.2 and 5.3 present the testing performance of all methods on classification and regression tasks, respectively. The results provide the following observations: (1) Galformer consistently achieves the best performance on 11 out of 14 downstream datasets, demonstrating its effectiveness. Compared to previous SOTA results from baselines, Galformer

has an overall relative improvement of 1.2% on the classification tasks and 1.8% on the regression tasks. (2) Among contrastive learning baselines, GraphMVP and 3DInformax, which leverage both 2D and 3D graphs in pre-training, perform better on the whole average than GraphCL and JOAO, which only use 2D graphs. It indicates that incorporating 3D geometry can effectively facilitate molecular self-supervised learning. Furthermore, Galformer achieves more significant improvement compared with the 2D-3D contrastive baselines. This result can be attributed to our complementary pre-training tasks, which can adaptively capture both inter-modality and intra-modality information. (3) KPGT, DMP and GROVER outperform other baselines for most tasks. One potential explanation is that they both employ transformer-based architectures as backbone networks, leading to higher model capacity and more expressive power than message passing models. Nevertheless, the lack of leveraging 3D geometry limits their ability to fully learn molecular representations. Therefore, Galformer exhibits a notable capability to surpass the inherent limitations in previous methods, consequently achieving more accurate molecular property prediction.

Table 5.3 Test RMSE Performance of different methods on five downstream regression datasets (**best in bold**, second best in underline). A lower value indicates better performance (marked by ↓). Each experiment is conducted independently three times with different random seeds for scaffold split.

	Regression error (RMSE) ↓				
Dataset	ESOL	Freesolv	Lipo	Malaria	CEP
# Molecules	1,128	642	4,200	9,999	29,978
# Tasks	1	1	1	1	1
GraphCL	$1.253_{(0.023)}$	$2.216_{(0.117)}$	$0.762_{(0.029)}$	$1.115_{(0.127)}$	$1.251_{(0.029)}$
JOAO	$1.203_{(0.061)}$	$2.010_{(0.189)}$	$0.759_{(0.015)}$	$1.117_{(0.122)}$	$1.278_{(0.021)}$
GROVER	$0.928_{(0.106)}$	$1.998_{(0.280)}$	$0.703_{(0.023)}$	$1.092_{(0.086)}$	$1.064_{(0.085)}$
3DInfomax	$1.167_{(0.148)}$	$2.275_{(0.391)}$	$0.755_{(0.025)}$	$\underline{1.075}_{(0.103)}$	$1.277_{(0.019)}$
GraphMVP	$0.995_{(0.113)}$	$1.806_{(0.164)}$	$0.736_{(0.051)}$	$1.101_{(0.107)}$	$1.248_{(0.010)}$
DMP	$0.815_{(0.048)}$	$1.335_{(0.251)}$	$0.658_{(0.020)}$	$1.095_{(0.085)}$	$1.128_{(0.042)}$
KPGT	$\underline{0.802}_{(0.096)}$	$\underline{1.230}_{(0.231)}$	$\mathbf{0.621}_{(0.018)}$	$1.083_{(0.082)}$	$\underline{1.020}_{(0.100)}$
Galformer	$\mathbf{0.741}_{(0.087)}$	$\mathbf{1.213}_{(0.200)}$	$\underline{0.628}_{(0.015)}$	$\mathbf{1.066}_{(0.092)}$	$\mathbf{1.015}_{(0.084)}$

5.3.6 Ablation Study

We conduct ablation studies to analyze the effectiveness of three critical designs in Galformer.

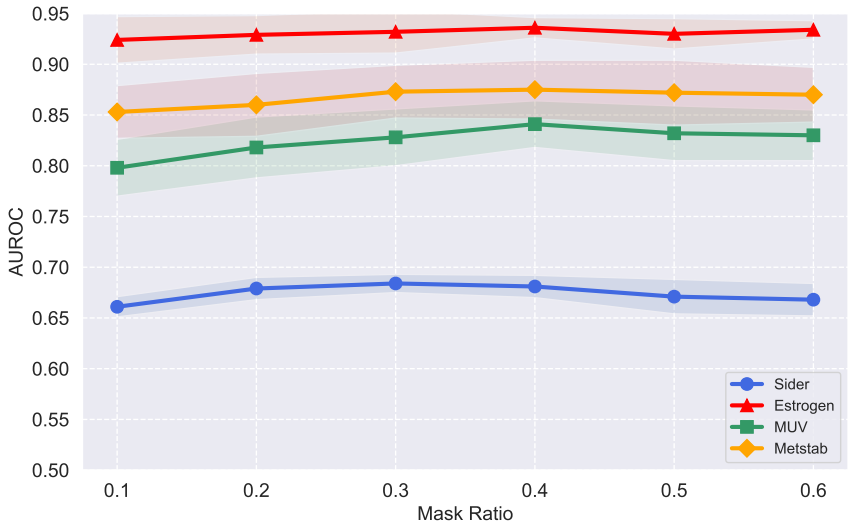

Fig. 5.4 Performance variation of Galformer over different mask ratios on four benchmark datasets.

Impact of Different Mask Ratios. We randomly mask a proportion of nodes in the 2D and 3D line graphs during pre-training. Figure 5.4 shows the impact of different mask ratios on four benchmark datasets. We can observe that Galformer has the best overall performance with a mask ratio of 40%. This ratio is significantly higher than the commonly employed ratios in previous works (Devlin et al., 2019; Hu et al., 2020; Wang et al., 2021b), which were typically below 25%. As our additional contrastive learning provides complementary knowledge from different modalities, Galformer can adapt to challenging tasks with a higher mask ratio. This shares similarity with the success of masked autoencoders (MAE) (He et al., 2022) in the vision domain, where a high mask ratio can better capture long-range dependencies and improve model generalization.

Effect of Pre-training Strategy and Backbone Network. To verify the effectiveness of our pre-training strategy and backbone network, we study Galformer without pre-training, and replace 2D line graph transformer backbone with vanilla transformer (Vaswani et al., 2017) and Graphormer (Ying et al., 2021). Table 5.4 shows the performance comparison. We can observe that Galformer achieves significant improvement after pre-training. Specifically, our pre-training strategy yields an average improvement of 5.9% in AUROC and 10.2% in

Table 5.4 Ablation study with the absence of pre-training and different backbones in averaged AUROC on classification datasets and RMSE on regression datasets (**best in bold**, second best in underline)

Backbone	Classification **Avg. AUROC** ↑	Regression **Avg. RMSE** ↓
No Pre-training	0.810	1.048
Vanilla Transformer	0.825	1.012
Graphormer	0.854	0.951
Galformer	**0.858**	**0.941**

Table 5.5 Ablation study of different contrastive losses in averaged AUROC on classification datasets and RMSE on regression datasets (**best in bold**, second best in underline).

Contrastive Loss (CL)	Classification **Avg. AUROC** ↑	Regression **Avg. RMSE** ↓
w/o CL	0.838	0.986
NT-Xent	0.850	0.954
EMB-NCE	0.852	0.956
InfoNCE	**0.858**	**0.941**

RMSE. Compared to the other pre-trained variants with different backbones, Galformer still achieves the best performance. This improvement can be attributed to our well-designed graph structural encodings in 2D line graph transformer, which can capture key structural information in molecules.

Choice of Contrastive Loss. For modeling inter-modality relations, Galformer leverages contrastive learning to bridge the modality gap between 2D and 3D representations. To validate the effectiveness of our equipped InfoNCE loss, we implement three model variants: one without contrastive learning, and two others with NT-Xent (Chen et al., 2020b) and EMB-NCE (Liu et al., 2022) contrastive losses, respectively. As observed from Table 5.5, applying contrastive loss can result in improved performance on both classification and regression tasks. It indicates that the incorporation of 3D information during pre-training can significantly enhance model performance toward downstream tasks. Moreover, InfoNCE loss consistently outperforms the other variants, which verifies the effectiveness of InfoNCE on cross-modality contrastive learning.

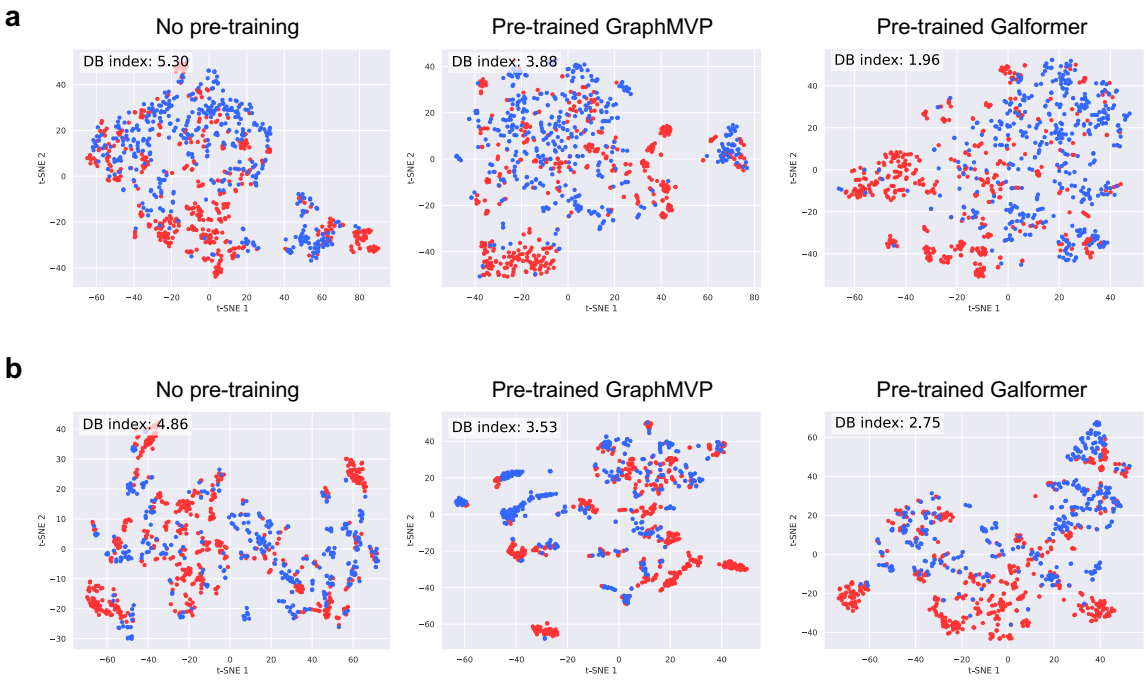

Fig. 5.5 Visualization of the pre-trained molecular representations on BBBP and ESOL datasets via t-SNE. All results are without fine-tuning by ground-truth labels. (a) BBBP dataset: color represents binary labels for the barrier permeability property. (b) ESOL dataset: color represents binary labels for water solubility. DB index measures the degree of cluster separation. The lower the DB index, the better the separation.

5.3.7 Molecular Representation Visualization

To intuitively investigate the impact of pre-trained molecular representations without fine-tuning, we randomly select 800 molecules from BBBP and ESOL datasets respectively, and utilize t-SNE (Van der Maaten and Hinton, 2008) to project these representations onto a 2D embedding space for visualization. On BBBP, each molecule is associated with a binary class label that indicates its barrier permeability property. On ESOL, the original label is a continuous variable that denotes the water solubility of the molecule. We simply use the median 0.057 mols/L in ESOL as a threshold. A molecule is considered to be positive if its water solubility is more than the threshold, and negative otherwise. Figure 5.5 shows the visualization results, using the Davies Bouldin (DB) index (Davies and Bouldin, 1979) as a metric to measure the cluster separation. The lower the DB index, the better the separation. We can observe that pre-trained GraphMVP and Galformer can better separate the labeled

molecules than the model without pre-training. In particular, Galformer further decreases the DB index to 1.96 on BBBP, compared with 3.88 of GraphMVP. Note that this is a zero-shot learning scenario without fine-tuning. Therefore, Galformer can effectively incorporate generalizable domain knowledge during pre-training, and thus significantly improve learned molecular representations.

5.4 Summary

In this chapter, we propose a Geometry-aware line graph transformer (Galformer), a dual-modality pre-training framework for molecular representation learning. Specifically, we first transform original 2D and 3D molecular graphs into line graphs, and then design a dual-modality line graph transformer backbone to fully encode both topological and geometric information. Furthermore, we build two complementary pre-training tasks as self-supervised objectives, learning complete knowledge from both intra-modality and inter-modality levels. Extensive experiments on downstream tasks demonstrate that Galformer has improved over SOTA baselines.

Chapter 6

Mask Prior-Guided Denoising Diffusion Improves Inverse Protein Folding

In Chapter 5, we introduced a dual-modality pre-training framework for molecular property prediction. The pre-trained model can be adapted to downstream discriminative tasks via supervised fine-tuning. However, we have not yet explored the potential benefits of self-supervised learning for generative tasks, which are common in drug discovery, such as de novo protein design and inverse protein folding (IPF). In this chapter, we develop a diffusion-based generative model for IPF prediction. In particular, we enhance the model's generic transferability via mask prior-guided pre-training.

6.1 Introduction

Proteins are complex, three-dimensional (3D) structures folded from linear amino acid (AA) sequences. They play a critical role in essentially all biological processes, including metabolism, immune response and cell cycle control. As described in Section 2.4.3, The IPF problem is a fundamental structure-based protein design problem in computational biology and medicine. It aims to generate valid AA sequences with the potential to fold into a desired 3D backbone structure, enabling the creation of novel proteins with specific functions

(Dauparas et al., 2022). Its enormous applications range from therapeutic protein engineering, lead compound optimization and antibody design.

Traditional physics-based approaches consider IPF as an energy optimization problem (Alford et al., 2017), suffering from high computational cost and limited accuracy. In recent years, deep learning has emerged as the preferred paradigm for solving protein structure problems due to its strong ability to learn complex non-linear patterns from data adaptively. In deep learning for IPF, early convolutional neural network-based models view each protein residue as an isolated unit or the whole as point cloud data, with limited consideration of structural information and interactions between residues (Wu et al., 2021b; Li et al., 2014; O'Connell et al., 2018; Anand et al., 2022). Recently, graph-based methods have represented 3D protein structures as proximity graphs, and then use graph neural networks (GNNs) to model residue representations and incorporate structural constraints. GNNs can aggregate and exchange local information within graph-structured data, enabling substantial performance improvement in graph-based methods.

Despite the advances in graph-based methods, structural information alone cannot determine the residue identities of some challenging regions, such as loops and intrinsically disordered regions (Towse and Daggett, 2012; Zheng et al., 2023). In such uncertain, low-confidence cases, interactions with other accurately predicted residues can provide more reliable guidance for mitigating uncertainty in these regions. Moreover, existing deep learning-based IPF methods typically employ autoregressive decoding or uniformly random decoding to generate AA sequences, prone to accumulate prediction errors (Li et al., 2020; Martínez-González et al., 2021) and limited in capturing global and long-range dependencies in protein evolution Starr and Thornton (2016); Xu et al. (2021). Recently, several non-autoregressive alternatives have shown the potential to outperform the autoregressive paradigm in related contexts (Li et al., 2020; Martínez-González et al., 2021; Lyu et al., 2024). Additionally, protein structure prediction methods, such as the AlphaFold series (Jumper et al., 2021; Abramson et al., 2024), often take an iterative generation process to refine non-deterministic structures by integrating well-predicted information. These raise the question: can combining residue interactions with an iterative refinement and an efficient

non-autoregressive decoding improve IPF prediction performance to generate more plausible protein sequences?

Recently, denoising diffusion models, an innovative class of deep generative models, have gained growing attention in various fields. They learn to generate conditional or unconditional data by iteratively denoising random samples from a prior distribution. Diffusion-based models have been adopted for de novo protein design and molecule generation, achieving state-of-the-art performance. For instance, RFdiffusion (Watson et al., 2023) fine-tunes the protein structure prediction network RoseTTAFold (Baek et al., 2021) under a denoising diffusion framework to generate 3D protein backbones, and Torsional Diffusion (Jing et al., 2022) implements a diffusion process on the space of torsion angles for molecular conformer generation. In structure-based drug design, DiffSBDD (Schneuing et al., 2022) proposes an equivariant 3D-conditional diffusion model to generate novel small-molecule binders conditioned on target protein pockets. While diffusion models have a widespread application in computational biology, most existing methods primarily focus on generating structures in continuous 3D space. The potential of diffusion models in inverse folding has not been fully exploited yet.

In this chapter, we propose a Mask prior-guided denoising Diffusion (MapDiff) framework (Figure 6.1) to accurately capture structure-to-sequence mapping for IPF prediction. Unlike previous graph-based methods, MapDiff models IPF as a discrete denoising diffusion problem that iteratively generates less-noisy AA sequences conditioned on a target protein structure. Due to the property of denoising diffusion, MapDiff can also be viewed as an iterative refinement that enhances the accuracy of the generated sequences over time. Moreover, we design a novel denoising network with non-autoregressive decoding to adaptively improve the denoising trajectories using a pre-trained mask prior. Our denoising network effectively leverages the structural information and residue interactions to reduce prediction error on low-confidence residue prediction. To further improve uncertainty estimation and the denoising speed, we combine the denoising diffusion implicit model (DDIM) (Song et al., 2021) with Monte-Carlo dropout (Gal and Ghahramani, 2016) in the discrete generative process. We conduct comprehensive experimental comparisons against state-of-the-art methods

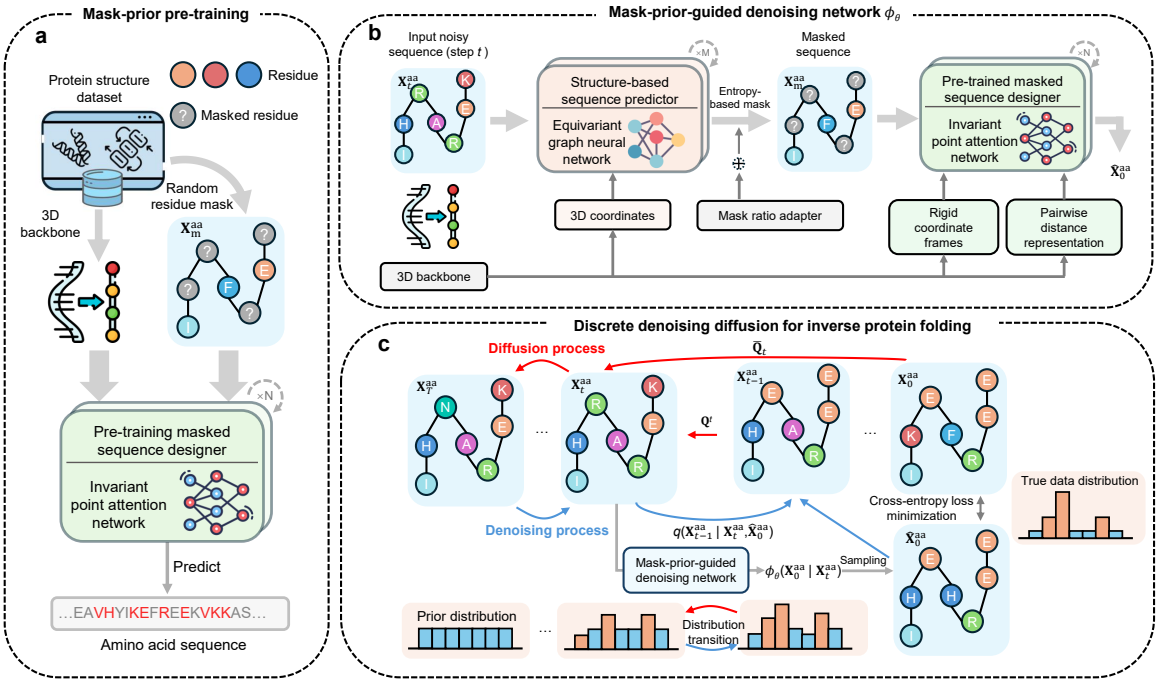

Fig. 6.1 Mask prior-guided denoising diffusion (MapDiff) for inverse protein folding. **(a)** The mask-prior pre-training stage randomly masks residues within the AA sequence and pre-trains an invariant point attention (IPA) network with the masked sequence and the 3D backbone structure to learn prior structural and sequence knowledge, using BERT-like masked language modelling objectives. **(b)** The mask prior-guided denoising network ϕ_θ takes an input noisy AA sequence $\mathbf{X}^{aa}$ to predict the native AA sequence $\mathbf{X}_0^{aa}$ via three operations in every iterative denoising step: it first initializes a structure-based sequence predictor as an equivariant graph neural network to denoise the noisy sequence $\mathbf{X}^{aa}$ conditioned on the provided 3D backbone structure. Then, combining an entropy-based mask strategy with a mask ratio adaptor identifies and masks low-confidence residues in the denoised sequence in the first step to produce a masked sequence $\mathbf{X}_m^{aa}$. Next, the pre-trained masked sequence designer in (a) takes the masked sequence $\mathbf{X}_m^{aa}$ and its 3D backbone information for refinement (fine-tuning) to better predict the native sequence $\mathbf{X}_0^{aa}$. **(c)** The MapDiff denoising diffusion framework iteratively alternates between two processes: diffusion and denoising. The diffusion process progressively adds random discrete noise to the native sequence $\mathbf{X}_0^{aa}$ according to a transition probability matrix $\overline{\mathbf{Q}}_t$ at the diffusion step t so that the real data distribution can gradually transition to a uniform prior distribution. The denoising process randomly samples an initial noisy AA sequence $\mathbf{X}_T^{aa}$ from the prior distribution and iteratively uses the denoising network ϕ_θ in (b) to denoise it, learning to predict the native sequence $\mathbf{X}_0^{aa}$ from $\mathbf{X}_t^{aa}$ at each denoising step t. The prediction $\hat{\mathbf{X}}_0^{aa}$ facilitates the computation of the posterior distribution $q(\mathbf{X}_{t-1}^{aa} \mid \mathbf{X}_t^{aa}, \hat{\mathbf{X}}_0^{aa})$ for predicting a less noisy sequence $\mathbf{X}_{t-1}^{aa}$.

for IPF prediction, demonstrating the effectiveness of MapDiff across multiple metrics and benchmarks, outperforming even those incorporating external knowledge. Moreover, when we use AlphaFold2 (Jumper et al., 2021) to fold the sequences generated by MapDiff back to 3D structures, such AlphaFold2-folded structures are highly similar to the native protein templates, even for cases of low sequence recovery rates.

Overall, this chapter shows the high potential of utilizing discrete denoising diffusion models with mask prior pre-training for IPF prediction. Our main contributions are three-fold: (i) we propose a discrete denoising diffusion-based framework named MapDiff to explicitly consider the structural information and residue interactions in the diffusion and denoising processes; (ii) we design a mask prior-guided denoising network that adaptively denoise the diffusion trajectories to produce feasible and diverse sequences from a fixed structure; (iii) MapDiff incorporates discrete DDIM with Monte-Carlo dropout to accelerate the generative process and improve uncertainty estimation.

6.2 Methodology

6.2.1 Discrete Denoising Diffusion Models

Denoising diffusion models are a class of deep generative models trained to create new samples by iteratively denoising sampled noise from a prior distribution. The training stage of a diffusion model consists of a forward diffusion process and a reverse denoising process. Given an original data distribution $q(\mathbf{x}_0)$, the forward diffusion process gradually corrupts a data point $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ into a series of increasingly noisy data points $\mathbf{x}_{1:T} = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$ over T time steps. This process follows a Markov chain, where $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. Conversely, the reverse denoising process, denoted as $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, aims to progressively reduce noise towards the original data distribution $q(\mathbf{x}_0)$ by predicting $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$. The initial noise $\mathbf{x}_T$ is sampled from a pre-defined prior distribution $p(\mathbf{x}_T)$, and the denoising inference p_θ can be parameterized by a learnable neural network. While the diffusion and denoising processes are agnostic to the data modality, the choice of prior distributions and Markov transition operators varies between continuous and discrete spaces.

We follow the settings of the discrete denoising diffusion proposed by Austin et al. (2021) and Vignac et al. (2022). In contrast to typical Gaussian diffusion models that operate in continuous state space, discrete denoising diffusion models introduce noise to categorical data using transition probability matrices in discrete state space. Let $x_t \in \{1, \cdots, K\}$ denote the categorical data with K categories and its one-hot encoding represented as $\mathbf{x}_t \in \mathbb{R}^K$. At time step t, the forward transition probabilities can be denoted by a matrix $\mathbf{Q}_t \in \mathbb{R}^{K \times K}$, where $[\mathbf{Q}_t]_{ij} = q(x_t = j \mid x_{t-1} = i)$ is the probability of transitioning from category i to category j. Therefore, the discrete transition kernel in the diffusion process is defined as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \text{Cat}(\mathbf{x}_t; \mathbf{p} = \mathbf{x}_{t-1}\mathbf{Q}_t), \tag{6.1}$$

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \text{Cat}(\mathbf{x}_t; \mathbf{p} = \mathbf{x}_0\overline{\mathbf{Q}}_t), \text{ with } \overline{\mathbf{Q}}_t = \mathbf{Q}_1\mathbf{Q}_2 \cdots \mathbf{Q}_t, \tag{6.2}$$

where $\text{Cat}(\mathbf{x}; \mathbf{p})$ represents a categorical distribution over $\mathbf{x}_t$ with probabilities determined by $\mathbf{p} \in \mathbb{R}^K$. As the diffusion process has a Markov chain, the transition matrix from $\mathbf{x}_0$ to $\mathbf{x_t}$ can be written as a closed form in Equation 6.2 with $\overline{\mathbf{Q}}_t = \mathbf{Q}_1\mathbf{Q}_2 \cdots \mathbf{Q}_t$. This property enables efficient sampling of $\mathbf{x}_t$ at arbitrary time steps without recursively applying noise. Following the Bayesian theorem, the calculation of posterior distribution from time step t to $t-1$ can be written as:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \propto \mathbf{x}_t\mathbf{Q}_t^T \odot \mathbf{x}_0\overline{\mathbf{Q}}_{t-1}, \tag{6.3}$$

where $\odot$ is a Hadamard (element-wise) product. The derivation of Equation 6.3 is detailed in Appendix A.1. The posterior $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is equivalent to $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ due to its Markov property. Thus, the clean data $\mathbf{x}_0$ is introduced for denoising estimation and can be used as the target of the neural network p_θ. The transition matrix $\mathbf{Q}_t$ has multiple choices. A simple yet effective approach is the marginal transition parametrized by $\mathbf{Q}_t = (1-\beta_t)\mathbf{I} + \beta_t\mathbf{1}_K\mathbf{p}^T$, where $\mathbf{p} \in \mathbb{R}^{20}$ denotes the marginal probability distribution of AA types in the training data. As $\lim_{t \to T}\beta_t = 1$, $q(\mathbf{x}_t)$ converges to an independent marginal distribution, regardless of the initial $\mathbf{x}_0$. Consequently, the marginal distribution can serve as the prior distribution for noise sampling in the generative process.

6.2.2 Problem Formulation

IPF prediction aims to generate a feasible AA sequence that can fold into a desired backbone structure. Given a target protein of length L, we present it as a proximity residue graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}, \mathbf{E})$, where each node denotes an AA residue within the protein. The node features $\mathbf{X} = [\mathbf{X}^{aa}, \mathbf{X}^{pos}, \mathbf{X}^{prop}]$ encodes the AA residue types, 3D spatial coordinates, and geometric properties. The adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$ is constructed using the k-nearest neighbor algorithm. Specifically, each node is connected to a maximum of k other nodes within a cutoff distance smaller than 30 Å. The edge feature matrix $\mathbf{E} \in \mathbb{R}^{M \times 93}$ illustrates the spatial and sequential relationships between the connected nodes. For sequence generation, we define a discrete denoising process on the types of noisy AA residues $\mathbf{X}_t^{aa} \in \mathbb{R}^{N \times 20}$ at time t. Conditioned on the noise graph $\mathcal{G}_t$, this process is subject to iteratively refine noise $\mathbf{X}_t^{aa}$ towards a clean $\mathbf{X}_0^{aa} = \mathbf{X}^{aa}$, which is predicted by our mask prior-guided denoising network.

6.2.3 Residue Graph Feature Construction

The protein is represented as a residue graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}, \mathbf{E})$ to reflect its geometric structure and topological relationships. The node feature $\mathbf{X} \in \mathbb{R}^{N \times 42}$ is decomposed into three components: $\mathbf{X}^{aa} \in \mathbb{R}^{N \times 20}, \mathbf{X}^{pos} \in \mathbb{R}^{N \times 3}$ and $\mathbf{X}^{prop} \in \mathbb{R}^{N \times 19}$. Each row of $\mathbf{X}^{aa}$ represents the one-hot encoded vector that indicates the amino acid type of a residue. $\mathbf{X}^{pos}$ denotes the spatial backbone coordinates, specifically represented by the backbone C_α atoms. Following Yi et al. (2024) and Ganea et al. (2022), we describe the structural and geometric properties of the residues by $\mathbf{X}^{prop}$, including the solvent-accessible surface area (SASA), crystallographic B-factor, surface-aware node features, protein secondary structure and backbone dihedral angles. Protein secondary structure reflects the local folding patterns of AA residues in a protein chain. We utilize the DSSP (Kabsch and Sander, 1983) algorithm to calculate the secondary structures for each residue and represent them by one-hot encoding. We derive the sin and cos values for the backbone dihedral angles ϕ and ψ, denoting the spatial arrangement within the backbone atoms. The surface-aware node features are the weighted

average distance of each residue node to its one-hop neighbors, defined as follows:

$$w_{i,i',\lambda} = \frac{\exp(-||\mathbf{x}_i^{pos} - \mathbf{x}_{i'}^{pos}||^2/\lambda)}{\sum_{j\in\mathcal{N}_i}\exp(-||\mathbf{x}_i^{pos} - \mathbf{x}_j^{pos}||^2/\lambda)}, \tag{6.4}$$

$$\rho_i(\mathbf{x}_i;\lambda) = \frac{\|\sum_{i'\in\mathcal{N}_i} w_{i,i',\lambda}(\mathbf{x}_i^{pos} - \mathbf{x}_{i'}^{pos})\|}{\sum_{i'\in\mathcal{N}_i} w_{i,i',\lambda}\|\mathbf{x}_i^{pos} - \mathbf{x}_{i'}^{pos}\|}, \tag{6.5}$$

where $\lambda \in \{1.,2.,5.,10.,30.\}$.

The edge feature $\mathbf{E} \in \mathbb{R}^{M\times 93}$ describes connections between pairs of residue nodes, including the relative spatial distances, local spatial positions and relative sequential positions. The relative spatial distance is defined as the Euclidean distance between the C_α coordinates of two residues. This distance is then projected into a 15-dimensional kernel space using a radial basis function (RBF). In addition, a binary contact signal (Ingraham et al., 2019) is used to indicate if the spatial distance between two residues is less than 8 Å. The local spatial positions (Ganea et al., 2022) have 12 dimensions and are created from a local coordinate system. They represent the relative positions and local orientations among the backbone atoms of residues. Next, the relative sequential position is defined as a 65-dimensional one-hot feature, based on the difference in the sequential indices along the AA chain.

For the IPA network, we further construct a pairwise distance representation $\mathbf{Z} = \{\mathbf{z}_{ij} \in \mathbb{R}^{1\times d_z} \mid 1 \le i \le N, 1 \le j \le N\}$ and rigid coordinate frames $\mathcal{T} = \{T_i := (\mathbf{R}_i \in \mathbb{R}^{3\times 3}, \mathbf{t}_i \in \mathbb{R}^3) \mid 1 \le i \le N\}$ as geometric features. The representation $\mathbf{Z}$ includes the RBF-based spatial distances between C_α, N, C, and a virtual C_β (Dauparas et al., 2022) estimated by other backbone atoms, as well as the relative sequential positions for all pairs of residues. The rigid coordinate frames (Bryant et al., 2022) are constructed using a Gram-Schmidt process from the coordinates of C_α, N, and C, ensuring the invariance of IPA with respect to global Euclidean transformations.

6.2.4 Framework Overview

Overall, the MapDiff framework formulates IPF prediction as a denoising diffusion problem (Figure 6.1c). The diffusion process progressively adds random discrete noise to the native

AA sequence according to the transition probability matrices to facilitate the training of a denoising network. In the denoising process, this denoising network iteratively denoises a noisy, randomly sampled AA sequence conditioned on the 3D structural information to predict or reconstruct the native AA sequence. The diffusion and denoising processes iterate alternately to capture the sampling diversity of native sequences from their complex distribution and refine the predicted AA sequences.

We propose a novel mask prior-guided denoising network to adaptively adjust the discrete denoising trajectories towards generating more valid AA sequences via three operations within each iterative denoising step (Figure 6.1b). Firstly, a structure-based sequence predictor employs an equivariant graph neural network (EGNN) (Satorras et al., 2021) to denoise the noisy sequence conditioned on the backbone structure. Secondly, we use an entropy-based mask strategy (Zhou et al., 2023) and a mask ratio adaptor to identify and mask low-confidence or uncertain (e.g. structurally undetermined) residues in the denoised sequence in the first operation to produce a masked sequence. Thirdly, a pre-trained masked sequence designer network predicts the masked residues to obtain their refined prediction, leveraging structural information. The pre-training of the masked sequence designer is done before the diffusion and denoising processes via an invariant point attention (IPA) network (Jumper et al., 2021) using masked language modelling (Figure 6.1a), incorporating prior structural and sequence knowledge. Both the structure-based sequence predictor and masked sequence designer network use non-autoregressive decoding to fully leverage structural information and residue interactions to enhance the denoising trajectories. To improve uncertainty estimation further, we use DDIM (Song et al., 2021) to skip some denoising steps and Monte Carlo dropout (Gal and Ghahramani, 2016), following common regularization practices.

6.2.5 IPF Denoising Diffusion Process

Discrete diffusion process. In the diffusion process, we incrementally introduce discrete noise to the clean AA residues over a number of time steps $t \in \{1, \cdots, T\}$, resulting in transforming the original data distribution to a simple uniform distribution. Given a clean AA sequence $\mathbf{X}_0^{aa} = \{\mathbf{x}_0^i \in \mathbb{R}^{1\times 20} \mid 1 \leq i \leq N\}$, we utilize a cumulative uniform transition

matrix $\overline{\mathbf{Q}}_t$ to independently add noise to each AA residue at arbitrary step t:

$$q(\mathbf{x}_t^i \mid \mathbf{x}_0^i) = \text{Cat}(\mathbf{x}_t^i; \mathbf{p} = \mathbf{x}_0^i \overline{\mathbf{Q}}_t), \text{with } \overline{\mathbf{Q}}_t = \mathbf{Q}_1 \mathbf{Q}_2 \cdots \mathbf{Q}_t, \tag{6.6}$$

$$q(\mathbf{X}_t^{aa} \mid \mathbf{X}_0^{aa}) = \prod_{1 \le i \le N} q(\mathbf{x}_t^i \mid \mathbf{x}_0^i), \tag{6.7}$$

where $\mathbf{Q}_t = (1-\beta_t)\mathbf{I} + \beta_t \mathbf{1}_K \mathbf{1}_K^T / K$, and K denotes the number of native AA types (i.e. 20). The weight of the noise, $\beta_t \in [0,1]$ is determined by a common cosine schedule (Nichol and Dhariwal, 2021).

Training objective of denoising network. The denoising neural network, denoted as ϕ_θ, is an essential component to reverse the noise process in diffusion models. In our framework, the network takes a noise residue graph $\mathcal{G}_t = (\mathbf{X}_t, \mathbf{A}, \mathbf{E})$ as input and aims to predict the real AA residues $\mathbf{X}_0^{aa}$. Specifically, we design a mask prior-guided denoising network as ϕ_θ, which effectively captures inherent structural information and learns the underlying data distribution. To train the learnable network ϕ_θ, the objective is to minimize the cross-entropy loss between the predicted AA probabilities and the real AA types over all nodes.

Reverse denoising process. Once the denoising network has been trained, it can be utilized to generate new AA sequences through an iterative denoising process. In this study, we first use the denoising network ϕ_θ to estimate the generative distribution $\hat{p}_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t^i)$ for each AA residue. Then the reverse denoising distribution $p_\theta(\mathbf{x}_{t-1}^i | \mathbf{x}_t^i)$ is parameterized by combining the posterior distribution with the marginalized network predictions as follows:

$$p_\theta(\mathbf{x}_{t-1}^i | \mathbf{x}_t^i) \propto \sum_{\hat{\mathbf{x}}_0^i} q(\mathbf{x}_{t-1}^i \mid \mathbf{x}_t^i, \hat{\mathbf{x}}_0^i) \hat{p}_\theta(\hat{\mathbf{x}}_0^i \mid \mathbf{x}_t^i), \tag{6.8}$$

$$p_\theta(\mathbf{X}_{t-1}^{aa} \mid \mathbf{X}_t^{aa}) = \prod_{1 \le i \le N} p_\theta(\mathbf{x}_{t-1}^i \mid \mathbf{x}_t^i), \tag{6.9}$$

where $\hat{\mathbf{x}}_0^i$ represents the predicted probability distribution for the i-th residue $\mathbf{x}_0^i$. The posterior distribution is defined as:

$$q(\mathbf{x}_{t-1}^i \mid \mathbf{x}_t^i, \hat{\mathbf{x}}_0^i) = \frac{q(\mathbf{x}_t^i \mid \mathbf{x}_{t-1}^i, \hat{\mathbf{x}}_0^i) q(\mathbf{x}_{t-1}^i \mid \hat{\mathbf{x}}_0^i)}{q(\mathbf{x}_t^i \mid \hat{\mathbf{x}}_0^i)}, \tag{6.10}$$

$$= \mathrm{Cat}(\mathbf{x}_{t-1}^i; \mathbf{p} = \frac{\mathbf{x}_t^i \mathbf{Q}_t^T \odot \hat{\mathbf{x}}_0^i \overline{\mathbf{Q}}_{t-1}}{\hat{\mathbf{x}}_0^i \overline{\mathbf{Q}}_t (\mathbf{x}_t^i)^T}). \tag{6.11}$$

By applying the reverse denoising process, the generation of less noise $\mathbf{X}_{t-1}^{aa}$ from $\mathbf{X}_t^{aa}$ is feasible. The derivation is detailed in Appendix A.1. The denoised result is determined by the predicted residues from the denoising neural network, as well as the predefined transition matrices at steps t and $t-1$. To generate a new AA sequence, the complete generative process begins with a random noise from the independent prior distribution $p(\mathbf{x}_T)$. The initial noise is then iteratively denoised at each time step using the reverse denoising process, gradually converging to a desired sequence conditioned on the given graph $\mathcal{G}$.

DDIM with Monte-Carlo dropout. Although discrete diffusion models have demonstrated significant generation ability in many fields, the generative process suffers from two limitations that hinder their success in IPF prediction. Firstly, the generative process is inherently computational inefficiency due to the numerous denoising steps involved, which require a sequential Markovian forward pass for the iterative generation. Secondly, the categorical distribution utilized for denoising sampling lacks sufficient uncertainty estimation. Many studies indicate that the logits produced by deep neural networks do not accurately represent the true probabilities. Typically, the predictions tend to be overconfident, leading to a discrepancy between the predicted probabilities and actual distribution. As the generative process iteratively draws samples from the estimated categorical distribution, insufficient uncertainty estimation will accumulate sampling errors and result in unsatisfactory performance.

To accelerate the generative process and improve uncertainty estimation, we propose a novel discrete sampling method by combining DDIM (Song et al., 2021) with Monte-Carlo dropout. DDIM, known as Denoising Diffusion Implicit Model, is a widely used method that improves the generation efficiency of diffusion models in continuous space. It defines the generative process as the reverse of a deterministic and non-Markovian diffusion process,

making it possible to skip certain denoising steps during generation. As discrete diffusion models possess analogous properties, Yi et al. (2024) extended DDIM into discrete space for IPF prediction. Similarly, we define the discrete DDIM sampling to the posterior distribution as follows:

$$q(\mathbf{x}_{t-k}^i \mid \mathbf{x}_t^i, \hat{\mathbf{x}}_0^i) = \text{Cat}(\mathbf{x}_{t-k}^i; \mathbf{p} = \frac{\mathbf{x}_t^i \mathbf{Q}_t^T \cdots \mathbf{Q}_{t-k}^T \odot \hat{\mathbf{x}}_0^i \overline{\mathbf{Q}}_{t-k}}{\hat{\mathbf{x}}_0^i \overline{\mathbf{Q}}_t (\mathbf{x}_t^i)^T}), \tag{6.12}$$

where k is the number of skipping steps.

Then we introduce the application of Monte-Carlo dropout within the generative process, a technique designed to enhance prediction uncertainty in neural networks. Specifically, we utilize dropout not only to prevent overfitting during the training of our denoising network, but also maintain its activation in the inference stage. By keeping dropout enabled and running multiple forward passes (Monte-Carlo samples) during inference, we generate a prediction distribution for each input, as opposed to a single-point estimation. To improve uncertainty estimation, we aggregate the predictions by taking a mean pooling over all output logits corresponding to the same input. This operation leads to the predicted logits that perform reduced estimation bias, and their normalized probabilities can more accurately reflect the actual distribution. Therefore, we can leverage Monte-Carlo dropout to enhance the generative process toward more reliable samplings. To clarify the algorithm details of MapDiff, we provide training and sampling inference implementation in Algorithm 6.1 and Algorithm 6.2.

As described in Algorithm 6.1, our denoising network follows a two-step approach, where we utilize the global-aware EGNN and pre-trained IPA network to instantiate the base sequence predictor h_ϕ and masked sequence designer f_θ, respectively. The training objective aims to minimize the cross-entropy losses L_b and L_m in the two stages, while the learnable weights ϕ and θ are jointly optimized through backpropagation. For the sampling inference procedure in Algorithm 6.2. It is a combination of DDIM with Monte-Carlo dropout to accelerate the sampling process and improve uncertainty estimation. The DDIM posterior computation is designed to skip a specific time step k at each denoising sampling.

Algorithm 6.1 Denoising Training Algorithm of MapDiff

Input: residue graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}, \mathbf{E})$, where $\mathbf{X} = [\mathbf{X}_0^{aa}, \mathbf{X}^{pos}, \mathbf{X}^{prop}]$, pairwise representation $\mathbf{Z}$, rigid coordinate frames $\mathcal{T}$, denoising time steps T
Initial: base sequence predictor h_ϕ, masked sequence designer f_θ
while ϕ, θ have not converged **do**
  First step: base sequence prediction
  Sample time step $t \sim \mathcal{U}(1, T)$ and noise sequence $\mathbf{X}_t^{aa} \sim q(\mathbf{X}_t^{aa} \mid \mathbf{X}_0^{aa})$
  Predict $p_b(\hat{\mathbf{X}}_0^{aa}) = h_\phi(\mathbf{X}_t^{aa}, \mathbf{X}^{pos}, \mathbf{X}^{prop}, \mathbf{E}, t)$ {Global-aware EGNN}
  Compute base cross-entropy loss $L_b = L_{CE}(p(\hat{\mathbf{X}}_0^{aa}), \mathbf{X}_0^{aa})$
  Second step: masked sequence refinement
  Compute base entropy $\{ent_1^b, \cdots ent_N^b\} = E(p(\hat{\mathbf{X}}_0^{aa}))$
  Compute mask ratio $mr_t = \sin\left(\frac{\pi}{2}\beta_t \cdot \sigma\right) + m$
  Generate entropy-based masked sequence and residue representations $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N]$
  Predict $p_m(\{\hat{\mathbf{X}}_0^{aa}\}_{mask}) = f_\theta(\mathbf{S}, \mathbf{Z}, \mathcal{T})$ {IPA network with mask pre-training}
  Compute mask cross-entropy loss $L_m = L_{CE}(p(\{\hat{\mathbf{X}}_0^{aa}\}_{mask}), \{\mathbf{X}_0^{aa}\}_{mask})$
  Compute total loss $L = L_b + L_m$
  Update $\phi, \theta \leftarrow \text{optimizer}(L, \phi, \theta)$
end while
return h_ϕ, f_θ

Additionally, by incorporating Monte-Carlo dropout, we can compute the mean probability prediction by passing multiple stochastic forwards. The experimental results demonstrate that this novel sampling inference significantly enhances the generative performance.

6.2.6 Mask Prior-Guided Denoising Network

In diffusion model applications, the denoising network significantly influences the generation performance. We develop a mask prior-guided denoising network, integrating both structural information and residue interactions for enhanced protein sequence prediction. Our denoising network architecture encompasses a structure-based sequence predictor, a pre-trained mask sequence designer, and a mask ratio adapter.

Structure-based sequence predictor. We adopt an equivariant graph neural network (EGNN) with a global-aware module as the structure-based sequence predictor, which generates a full AA sequence from the backbone structure. EGNN is a type of graph neural network that satisfies equivariance operations for the special Euclidean group SE(3). It preserves geometric and spatial relationships of 3D coordinates within the message-passing

Algorithm 6.2 Sampling Inference Algorithm of MapDiff

Input: residue graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}, \mathbf{E})$, where $\mathbf{X} = [\mathbf{X}^{pos}, \mathbf{X}^{prop}]$, pairwise representation $\mathbf{Z}$, rigid coordinate frames $\mathcal{T}$, denoising time steps T, skipping step k, stochastic forward passes C
Initial: base sequence predictor h_ϕ, masked sequence designer f_θ
Sample noise from a uniform prior $\mathbf{X}_T^{aa} \sim p(\mathbf{X}_T)$
Activate dropout for h_ϕ and f_θ during inference {Monte-Carlo dropout}
for c in $\{1, 2, ..., C\}$ **do**
  for t in $\{T, T-k, ..., 0\}$ **do**
    Predict $p_f(\hat{\mathbf{X}}_0^{aa})$ by neural networks h_ϕ, f_θ and $\mathbf{X}_t^{aa}$
    Compute $p_t^c(\mathbf{X}_{t-k}^{aa} | \mathbf{X}_t^{aa}, \hat{\mathbf{X}}_0^{aa})$ {DDIM posterior computation}
  end for
end for
Compute mean prediction $p_0^m(\mathbf{X}_0^{aa}) = \frac{1}{C} \sum p_0^c(\mathbf{X}_0^{aa} | \mathbf{X}_k^{aa}, \hat{\mathbf{X}}_0^{aa})$ {Monte-Carlo estimation}
return $\mathbf{X}_0^{aa} \sim p_0^m(\mathbf{X}_0^{aa})$

framework. Given a noise residue graph , we use $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N]$ to denote the initial node embeddings, which are derived from the noise AA types and geometric properties. The coordinates of each node is represented as $\mathbf{X}^{pos} = [\mathbf{x}_1^{pos}, \mathbf{x}_2^{pos}, \cdots \mathbf{x}_N^{pos}]$, while the edge feature are denoted as $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \cdots \mathbf{e}_M]$. In this setting, EGNN consists of a stack of equivariant graph convolutional layers (EGCL) for the node and edge information propagation, which are defined as:

$$\mathbf{e}_{ij}^{(l+1)} = \phi_e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\|^2, \mathbf{e}_{ij}^{(l)}), \tag{6.13}$$

$$\hat{\mathbf{h}}_i^{(l+1)} = \phi_h(\mathbf{h}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{e}_{ij}^{(l+1)}), \tag{6.14}$$

$$\mathbf{x}_i^{(l+1)} = \mathbf{x}_i^{(l)} + \frac{1}{N_i} \sum_{j \in \mathcal{N}(i)} (\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) \phi_x(\mathbf{e}_{ij}^{(l+1)}), \tag{6.15}$$

where l denotes the l-th EGCL layer, $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{pos}$, and $w_{ij} = \text{Sigmoid}(\phi_w(\mathbf{e}_{ij}^{(l+1)}))$ is a soft estimated weight assigned to the specific edge representation. All components (ϕ_e, ϕ_h, ϕ_x, ϕ_w) are learnable and parametrized by fully connected neural networks. In the information propagation, EGNN achieves equivariance to translations and rotations on the node coordinates $\mathbf{X}^{pos}$, and preserves invariant to group transformations on the node features $\mathbf{H}$ and edge features $\mathbf{E}$.

However, the vanilla EGNN only considers local neighbour aggregation while neglecting the global context. Some recent studies (Tan et al., 2023; Gao et al., 2023) have demonstrated the significance of global information in protein design. Therefore, we introduce a global-aware module in the EGCL layer, which incorporates the global pooling vector into the update of node representations:

$$\mathbf{m}^{(l+1)} = \text{Meanpool}(\{\hat{\mathbf{h}}_i^{(l+1)}\}_{i\in\mathcal{G}}), \tag{6.16}$$

$$\mathbf{h}_i^{(l+1)} = \hat{\mathbf{h}}_i^{(l+1)} \odot \text{Sigmoid}(\phi_m(\mathbf{m}^{(l+1)}, \hat{\mathbf{h}}_i^{(l+1)})), \tag{6.17}$$

where MeanPool$(\cdot)$ is the mean pooling operation over all nodes within a residue graph. The global-aware module effectively integrates global context into modelling and only increases a linear computational cost. To predict the probabilities of residue types, the node representations from the last EGCL layer are fed into a fully connected classification layer with softmax function, which is defined as:

$$\mathbf{l}_i^b = \mathbf{h}_i^{(L)}\mathbf{W}_o + \mathbf{b}_o, \tag{6.18}$$

$$\mathbf{p}_i^b = \text{Softmax}(\mathbf{l}_i^b), \tag{6.19}$$

where $\mathbf{W}_o \in \mathbb{R}^{D_h \times 20}$ and $\mathbf{b}_o \in \mathbb{R}^{1\times 20}$ are learnable weight matrix and bias vector.

Low-confidence residue selection and mask ratio adapter. As previously mentioned, structural information alone can sometimes be insufficient to determine all residue identities. Certain flexible regions display a weaker correlation with the backbone structure but are significantly influenced by their sequential context. To enhance the denoising network's performance, we introduce a masked sequence designer module. This module refines the residues identified with low confidence in the base sequence predictor. We adopt an entropy-based residue selection strategy, as proposed by Zhou et al. (2023), to identify these low-confidence residues. The entropy for the i-th residue of the probability distribution $\mathbf{p}_i^b$ is calculated as:

$$ent_i^b = -\sum_j \mathrm{p}_{ij}^b \log(\mathrm{p}_{ij}^b), \tag{6.20}$$

Given that entropy quantifies the uncertainty in a probability distribution, it can be utilized to locate the low-confidence predicted residues. Consequently, residues with the most entropy are masked, while the rest remain in a sequential context. The masked sequence designer aims to reconstruct the entire sequence by using the masked partial sequence in combination with the backbone structure. In addition, to account for the varying noise levels of the input sequence in diffusion models, we design a simple mask ratio adapter to dynamically determine the entropy mask percentage at different denoising steps:

$$\mathrm{mr}_t = \sin\left(\frac{\pi}{2}\beta_t \cdot \sigma\right) + m, \tag{6.21}$$

where $\beta_t \in [0,1]$ represents the noise weight at step t derived from the noise schedule, and σ and m are predefined deviation and minimum mask ratio, respectively. With the increase of β_t, the mask ratio is proportional to its time step.

Mask prior pretraining. To incorporate prior knowledge of sequential context, we pretrain the masked sequence designer by applying the masked language modelling objective proposed in BERT (Devlin et al., 2019). It is important to clarify that we use the same training data in the diffusion models for pretraining purposes, in order to avoid any information leakage from external sources. In this process, we randomly sample a proportion of residues in the native AA sequences, and replace them with the masking procedures: (i) masking 80% of the selected residues using a special MASK type, (ii) replacing 10% of the selected residues with other random residue types, and (iii) keeping the remaining 10% residues unchanged. Subsequently, we input the partially masked sequences, along with structural information, into the masked sequence designer. The objective of the pretraining stage is to predict the original residue types from the masked residue representations using a cross-entropy loss function.

Masked sequence designer. We use an invariant point attention (IPA) network as the masked sequence designer. IPA is a geometry-aware attention mechanism designed to facilitate the fusion of residue representations and spatial relationships, enhancing the structure generation within AlphaFold2 (Bryant et al., 2022). In this study, we repurpose the IPA

module to refine low-confidence residues in the base sequence predictor. Given a mask AA sequence, we denote its residue representation as $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N]$, which is derived from the residue types and positional encoding. To incorporate geometric information, as with the IPA implementation in Frame2seq (Akpinaroglu et al., 2023), we construct a pairwise distance representation $\mathbf{Z} = \{\mathbf{z}_{ij} \in \mathbb{R}^{1\times d_z} \mid 1 \le i \le N, 1 \le j \le N\}$ and rigid coordinate frames $\mathcal{T} = \{T_i := (\mathbf{R}_i \in \mathbb{R}^{3\times 3}, \mathbf{t}_i \in \mathbb{R}^3) \mid 1 \le i \le N\}$. The pairwise representation $\mathbf{Z}$ is obtained by calculating inter-residue spatial distances and relative sequence positions. The rigid coordinate frames are constructed from the coordinates of backbone atoms using a Gram-Schmidt process, ensuring the invariance of IPA with respect to the global Euclidean transformations. Subsequently, we take the residue representation, pairwise distance representation and rigid coordinate frames as inputs, and feed them into a stack of IPA layers for representation learning, which is defined as:

$$\mathbf{S}^{(l+1)}, \mathbf{Z}^{(l+1)} = \mathrm{IPA}(\mathbf{S}^{(l)}, \mathbf{Z}^{(l)}, \mathcal{T}^{(l+1)}). \tag{6.22}$$

The IPA network follows the self-attention mechanism. However, it enhances the general attention queries, keys, and values by incorporating 3D points that are generated in the rigid coordinate frame of each residue. This operation ensures that the updated residue and pair representations remain invariant by global rotations and translations.

The algorithm implementation of IPA is provided in Algorithm 6.3. In the original implementation of IPA, each layer regards the protein as a fully connected graph and only updates the node residue representations. Following the message passing proposed by Yim et al. (2023), we also perform an edge update by propagating the updated node information to pairwise edge representations. Finally, we extract the node representations from the last IPA layer to predict the masked residue types. For the i-th residue, the predicted probability distribution and entropy in the masked sequence designer are calculated as:

$$\mathbf{p}_i^m = \mathrm{Softmax}(\mathbf{l}_i^m), \quad \mathbf{l}_i^m = \mathbf{h}_i^{(L)}\mathbf{W}_m + \mathbf{b}_m, \tag{6.23}$$

Algorithm 6.3 Invariant Point Attention (IPA) Layer

Input: residue representation $\mathbf{S}^{(l)} = \{\mathbf{s}_i \in \mathbb{R}^{d_s} \mid 1 \le i \le N\}$, pairwise representation $\mathbf{Z}^{(l)} = \{\mathbf{z}_{ij} \in \mathbb{R}^{d_z} \mid 1 \le i \le N, 1 \le j \le N\}$, rigid coordinate frames $\mathcal{T} = \{T_i := (\mathbf{R}_i \in \mathbb{R}^{3\times 3}, \mathbf{t}_i \in \mathbb{R}^3) \mid 1 \le i \le N\}$

Parameters: $N_{\text{head}} = 4, c = 32, N_{\text{query point}} = 4, N_{\text{value point}} = 8$

$\mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h = \text{LinearNoBias}(\mathbf{s}_i)$ $\quad \mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h \in \mathbb{R}^{d_h}, h \in \{1, \cdots, N_{\text{head}}\}$

$\vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$ $\quad \mathbf{q}_i^{hp}, \mathbf{k}_i^{hp} \in \mathbb{R}^3, p \in \{1, \cdots, N_{\text{query point}}\}$

$\vec{\mathbf{v}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$ $\quad \mathbf{v}_i^{hp} \in \mathbb{R}^3, p \in \{1, \cdots, N_{\text{value point}}\}$

$b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

$w_C = \sqrt{\frac{2}{9N_{\text{query point}}}}$

$w_L = \sqrt{\frac{1}{3}}$

$a_{ij}^h = \text{softmax}\left(w_L\left(\frac{1}{\sqrt{c}}\mathbf{q}_i^{hT}\mathbf{k}_j^h + b_{ij}^h - \frac{\tau^h w_C}{2}\sum_p \|T_i \circ \vec{\mathbf{q}}_i^{hp} - T_j \circ \vec{\mathbf{k}}_j^{hp}\|^2\right)\right)$

$\tilde{\mathbf{o}}_i^h = \sum_j a_{ij}^h \mathbf{z}_{ij}$

$\mathbf{o}_i^h = \sum_j a_{ij}^h \mathbf{v}_j$

$\vec{\mathbf{o}}_i^{hp} = T_i^{-1} \circ \sum_j a_{ij}^h \left(T_j \circ \vec{\mathbf{v}}_j^{hp}\right)$

$\tilde{\mathbf{s}}_i = \text{Linear}\left(\text{concat}_{h,p}(\tilde{\mathbf{o}}_i^h, \mathbf{o}_i^h, \vec{\mathbf{o}}_i^{hp}, \|\vec{\mathbf{o}}_i^{hp}\|)\right)$

$\hat{\mathbf{s}}_i = \text{Linear}(\tilde{\mathbf{s}}_i)$

$\tilde{\mathbf{z}}_{ij} = \text{Linear}(\text{concat}(\hat{\mathbf{s}}_i, \hat{\mathbf{s}}_j, \mathbf{z}_{ij}))$

return $\mathbf{S}^{(l+1)} = \{\tilde{\mathbf{s}}_i \in \mathbb{R}^{d_s} \mid 1 \le i \le N\}$, $\mathbf{Z}^{(l+1)} = \{\tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{d_z} \mid 1 \le i \le N, 1 \le j \le N\}$

$$ent_i^m = -\sum_j \mathrm{p}_{ij}^m \log(\mathrm{p}_{ij}^m), \tag{6.24}$$

where $\mathbf{W}_m \in \mathbb{R}^{D_s \times 20}$ and $\mathbf{b}_m \in \mathbb{R}^{1\times 20}$ are learnable weight matrix and bias vector. The training objective is to jointly minimize the cross-entropy losses for both the base sequence predictor and masked sequence designer. In the inference stage, we calculate the final predicted probability by weighting the output logits based on their entropy as follows:

$$\mathbf{l}_i^{\mathrm{f}} = \frac{\exp(-ent_i^b)}{\exp(-ent_i^b) + \exp(-ent_i^m)}\mathbf{l}_i^b + \frac{\exp(-ent_i^m)}{\exp(-ent_i^b) + \exp(-ent_i^m)}\mathbf{l}_i^m. \tag{6.25}$$

$$\mathbf{p}_i^{\mathrm{f}} = \text{Softmax}(\mathbf{l}_i^{\mathrm{f}}). \tag{6.26}$$

By incorporating the mask-prior denoising network into the discrete denoising diffusion process, our framework enhances the diffusion trajectories, leading to more accurate predictions of protein sequences.

6.3 Experiments

6.3.1 Datasets

We mainly evaluate MapDiff against state-of-the-art baselines on the CATH 4.2 and CATH 4.3 datasets (Orengo et al., 1997). Following previous studies (Ingraham et al., 2019; Hsu et al., 2022; Gao et al., 2023), the proteins are partitioned based on the CATH topology classification, resulting in 18,024 structures for training, 608 structures for validation and 1,120 proteins for testing on CATH 4.2. Similarly, we conduct the topology classification following ESM-IF (Hsu et al., 2022) on CATH 4.3, which assigns 16,630 proteins to training set, 1,516 proteins to the validation set, and 1,864 proteins to the test set. As a result, there is no overlap of protein topology codes among the different sets. To study the generative quality of different proteins, we report evaluations in three specific categories: short proteins (up to 100 residues in length), single-chain proteins (labeled with 1 chain in CATH) and all proteins within the test sets. The zero-shot generalization is evaluated on TS50 and PDB2022. TS50 (Li et al., 2014) is a commonly used benchmark set in protein sequence design, comprising 50 individual protein chains. PDB2022 includes recently published single-chain structures from Protein Data Bank (Berman et al., 2000) curated by Zhou et al. (2023), with protein length ≤ 500 and resolution ≤ 2.5 Å after 2022. It consists of 1,975 protein structures and has no overlap with other experimental datasets.

6.3.2 Evaluation Strategies and Metrics

We evaluate the accuracy of generated sequences using three metrics: perplexity, recovery rate and native sequence similarity recovery (NSSR) (Löffler et al., 2017). Perplexity measures the alignment between a model's predicted AA probabilities and the native AA types at each residue position. The recovery rate indicates the proportion of accurately predicted AAs in the protein sequence. The NSSR evaluates the similarity between the predicted and native residues via the Blocks Substitution Matrix (BLOSUM) (Henikoff and Henikoff, 1992), where each residue pair contributes to a positive prediction if their BLOSUM score is greater than zero. We use BLOSUM42, BLOSUM62, BLOSUM80 and BLOSUM90

to account for AA similarities at four different cut-off levels for NSSR computation. To evaluate the foldability, i.e., the quality of refolded protein structures, we use five metrics: predicted local distance difference test (pLDDT), predicted aligned error (PAE), predicted template modelling (pTM), template modelling score (TM-score) and root mean square deviation (RMSD), where pLDDT, PAE and pTM measure the confidence and reliability of residue positions produced by the refolding algorithm, and TM-score and RMSD measure the discrepancies between the predicted 3D structures and their native counterparts.

6.3.3 Baselines

We compare MapDiff with recent deep graph models for inverse protein folding, including StructGNN (Ingraham et al., 2019), GraphTrans (Ingraham et al., 2019), GVP (Jing et al., 2020), AlphaDesign (Gao et al., 2022), ProteinMPNN (Dauparas et al., 2022), PiFold (Gao et al., 2023), LM-Design (Zheng et al., 2023) and GRADE-IF (Dauparas et al., 2022). To ensure a reliable and fair comparison, we reproduce the open-source and four most state-of-the-art baselines (ProteinMPNN, PiFold, LM-Design, and GRADE-IF) under identical settings in our experiments. ProteinMPNN uses a message-passing neural network to encode structure features, and a random decoding scheme to generate protein sequences. PiFold introduces a residue featurizer to extract distance, angle, and direction features. It proposes a PiGNN encoder to learn expressive residue representations, enabling the generation of protein sequences in a one-shot manner. LM-Design uses structure-based models as encoders and incorporates the protein language model ESM as a protein designer to refine the generated sequences. GRADE-IF employs EGNN to learn residue representations from protein structures, and adopts the graph denoising diffusion model to iteratively generate feasible sequences. All baselines are implemented following the default hyperparameter settings in their original papers.

6.3.4 Implementation Details

MapDiff is implemented in Python 3.8 and PyTorch 1.13.1 (Paszke et al., 2017), along with functions from BioPython 1.81 (Cock et al., 2009), PyG 2.4.0 (Fey and Lenssen, 2019), Scikit-learn 1.0.2 (Pedregosa et al., 2011), NumPy 1.22.3 (Harris et al., 2020), and RDKit 2023.3.3 (Greg Landrum et al, 2006). It consists of two training stages: mask prior pre-training and denoising diffusion model training, both of which use the same CATH 4.2/4.3 training set. The batch size is set to 8, and the models are trained up to 200 epochs in pre-training and 100 epochs in denoising training. We employ the Adam optimizer with a one-cycle scheduler for parameter optimization, setting the peak learning rate to 5e-4. In the denoising network, the structure-based sequence predictor consists of six global-aware EGCL layers, each with 128 hidden dimensions. Meanwhile, the masked sequence designer stacks six layers of IPA, each with 128 hidden dimensions and 4 attention heads. The dropout rate is set to 0.2 in both EGCL and IPA layers. A cosine schedule is applied to control the noise weight at each time steps, with a total of 500 time steps. During sampling inference, the skip steps for DDIM are configured to 100, and the Monte-Carlo forward passes are set to 50. For the mask ratio adaptor, we set the minimum mask ratio to 0.4 and the deviation to 0.2. All experiments are conducted on a single Tesla A100 GPU. Following the regular evaluation in deep learning, the best-performing model is selected based on the epoch that provides the highest recovery on the validation set. Finally, this selected model is subsequently used to evaluate performance on the test set.

6.3.5 Sequence Recovery Performance Comparison

Firstly, we evaluate the sequence recovery performance of MapDiff against state-of-the-art baselines on the CATH datasets. Table 6.1 presents the prediction perplexity and median recovery rate on the full test set, along with short and single-chain subsets. The results demonstrate that MapDiff achieves the best performance across different metrics and subsets of data, highlighting its effectiveness in generating valid protein sequences. Specifically, we can observe that: (1) MapDiff achieves a recovery rate of 60.93% and 60.68% on the full

Table 6.1 Performance comparison on the CATH 4.2 and CATH 4.3 datasets with topology classification split. The results include the perplexity and median recovery on the full test set, as well as on short and single-chain subsets. External knowledge entails the utilization of additional training data or protein language models. We also quoted partial baseline results from Gao et al. (2023) and Hsu et al. (2022) for comparative analysis, marked with †. The best result for each dataset and metric is marked in bold and the second-best result is underlined.

	Models	External Knowledge	Total Params	Perplexity (↓)			Median Recovery (%, ↑)		
				Short	Single-chain	Full	Short	Single-chain	Full
CATH 4.2	†StructGNN	×	1.4M	8.29	8.74	6.40	29.44	28.26	35.91
	†GraphTrans	×	1.5M	8.39	8.83	6.63	28.14	28.46	35.82
	†GVP	×	2.0M	7.09	7.49	6.05	32.62	31.10	37.64
	†AlphaDesign	✓	6.6M	7.32	7.63	6.30	34.16	32.66	41.31
	ProteinMPNN	×	1.9M	6.90	7.03	4.70	36.45	35.29	48.63
	PiFold	×	6.6M	5.97	<u>6.13</u>	4.61	39.17	42.43	51.40
	LM-Design	✓	659M	6.86	6.82	4.55	37.66	38.94	<u>53.19</u>
	GRADE-IF	×	7.0M	<u>5.65</u>	6.46	<u>4.40</u>	<u>45.84</u>	<u>42.73</u>	52.63
	MapDiff	×	14.7M	**3.96**	**4.41**	**3.43**	**54.04**	**49.34**	**60.93**
CATH 4.3	†GVP-GNN-Large	×	21M	7.68	6.12	6.17	32.60	39.40	39.20
	†+ AF2 predicted data	✓	142M	6.11	4.09	4.08	38.30	50.08	50.08
	†GVP-Transformer	×	21M	8.18	6.33	6.44	31.30	38.50	38.30
	†+ AF2 predicted data	✓	142M	6.05	4.00	4.01	38.10	51.50	51.60
	ProteinMPNN	×	1.9M	6.12	6.18	4.63	40.00	39.13	47.66
	PiFold	×	6.6M	5.52	<u>5.00</u>	<u>4.38</u>	43.06	45.54	51.45
	LM-Design	✓	659M	6.01	5.73	4.47	44.44	45.31	<u>53.66</u>
	GRADE-IF	×	7.0M	<u>5.30</u>	6.05	4.58	<u>48.21</u>	<u>45.94</u>	52.24
	MapDiff	×	14.7M	**3.90**	**3.83**	**3.52**	**55.56**	**54.99**	**60.68**

CATH 4.2 and CATH 4.3 test sets, significantly outperforming the state-of-the-art baselines by 14.5% and 13.1%, respectively. Furthermore, MapDiff shows recovery improvements of 17.9% and 15.5% on the short and single-chain test sets of CATH 4.2. (2) MapDiff consistently achieves the lowest perplexity compared to previous methods and produce high-confidence probability distribution to facilitate accurate predictions. (3) MapDiff is a highly accurate IPF model that operates independently of external knowledge. In some of the compared baselines, external knowledge sources, such as additional training data or protein language models, are utilized to enhance prediction accuracy. Due to its well-designed architecture and diffusion-based generation mechanism, MapDiff effectively utilizes limited training data to capture relevant patterns to achieve superior generalizability.

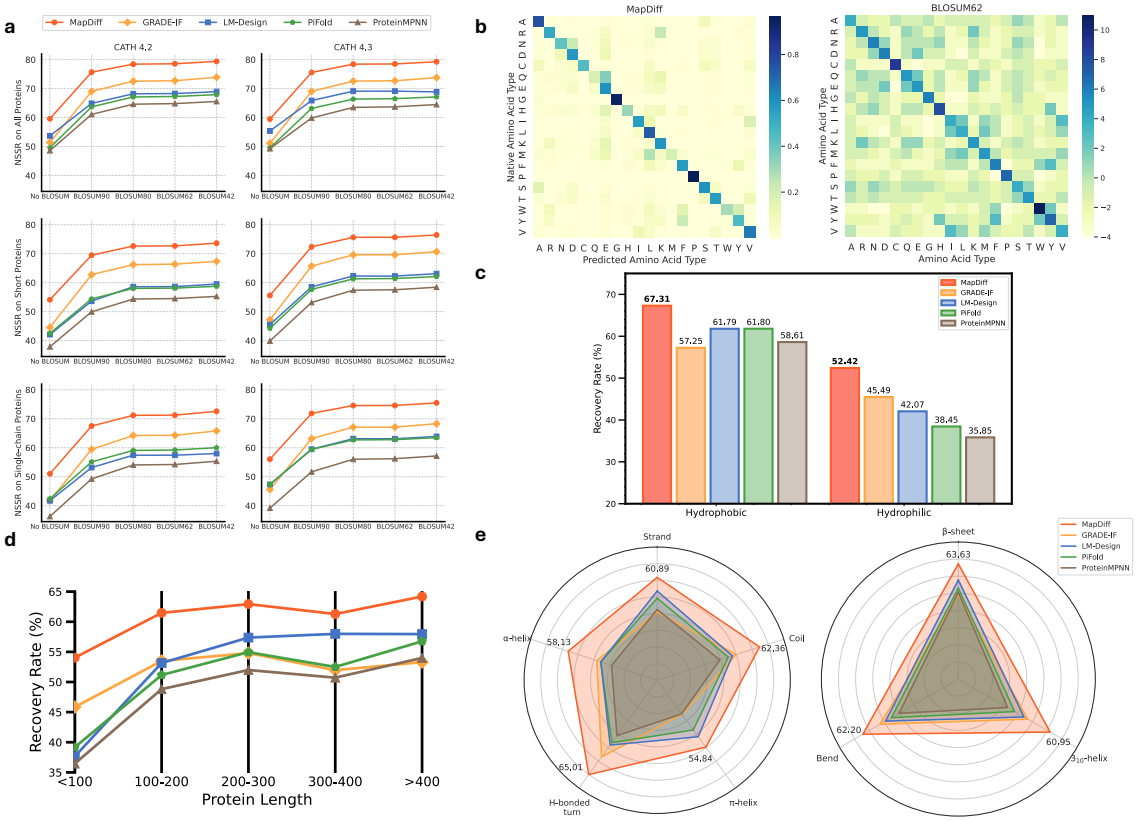

Fig. 6.2 Model performance comparison and sensitivity analysis across different scenarios on the CATH datasets. **(a)** NSSR scores for MapDiff and baseline methods Yi et al. (2024); Gao et al. (2023); Zheng et al. (2023); Dauparas et al. (2022) on the full test sets and the short and single-chain protein subsets for four different BLOSUM matrices and no BLOSUM matrix. **(b)** The predicted confusion matrix for MapDiff with the native BLOSUM62 matrix, where darker colors indicate a higher prediction likelihood. **(c)** Breakdown of the recovery rates into hydrophilic and hydrophobic residues. **(d)** Median sequence recovery rates across different protein lengths. **(e)** Residue recovery performance across different secondary structures visualized in two groups for clarity. For example, the coils represent residues without specific second structures, where MapDiff outperforms the baselines significantly.

Table 6.2 Zero-shot performance comparison on knowledge transfer from CATH to PDB2022 and TS50 datasets. We report the test results of models trained on CATH 4.2 and CATH 4.3 (in brackets), respectively. The best result for each dataset and metric is marked in bold and the second-best result is underlined.

Models	PDB2022			TS50		
	Recovery (%)	NSSR62 (%)	NSSR90 (%)	Recovery (%)	NSSR62 (%)	NSSR90 (%)
ProteinMPNN	56.75 (56.65)	72.50 (72.59)	69.96 (69.95)	52.34 (51.80)	70.31 (70.13)	66.77 (66.80)
PiFold	60.63 (60.26)	75.55 (75.30)	72.96 (72.86)	58.39 (58.90)	73.55 (74.52)	70.33 (71.33)
LM-Design	66.03 (66.20)	79.55 (80.12)	77.60 (78.20)	57.62 (58.27)	73.74 (75.69)	71.22 (73.12)
GRADE-IF	58.09 (58.35)	77.44 (77.51)	74.57 (74.97)	57.74 (59.27)	77.77 (79.11)	74.36 (76.24)
MapDiff	**68.03** (**68.00**)	**84.19** (**84.30**)	**82.13** (**82.29**)	**68.76** (**69.77**)	**84.10** (**85.27**)	**81.76** (**83.08**)

We further conduct model performance and sensitivity analysis across different scenarios. Figure 6.2a presents the mean NSSR scores for MapDiff and the baselines on the CATH datasets. MapDiff consistently achieves the best NSSR scores across different test sets. Figure 6.2b compares the confusion matrix of MapDiff on CATH 4.2 with the native BLOSUM62 matrix. The confusion matrix denotes proportions for specific combinations of actual and predicted amino acid types, with darker cells indicating greater proportion. Non-diagonal darker cells indicate the alignment between closely related residue pairs according to the BLOSUM62 matrix. This observation indicates that MapDiff can effectively capture the practical substitutions between residues. Figures 6.2c and 6.2e show the sequence recovery performance across different amino acid types, as well as eight secondary structures. Notably, MapDiff is the only model achieving over 50% recovery rate in predicting hydrophobic amino acids and significant improvements in recovering α-helix and β-sheet secondary structures. Figure 6.2d presents a sensitivity analysis of the recovery performance for varying protein lengths. For short proteins (less than 100 amino acids in length), several baselines show a significant decrease in performance. For instance, the recovery rate of LM-Design falls below 40% for the short proteins. This could be due to the protein language model used in LM-Design being sensitive to protein length. In contrast, MapDiff, which employs non-autoregressive decoding and an iterative denoising process, consistently outperforms all baselines and maintains high performance across all protein lengths.

To validate the zero-shot transferability of our method, we compare the model's performance on two independent test datasets, TS50 and PDB2022, which do not overlap with the

CATH data, as shown in Table 6.2. The results demonstrate that MapDiff achieves the highest recovery and NSSR scores on both datasets. We can conclude that, even though LM-Design reaches a high recovery (66%) that is approaching our method on PDB2022, the performance gap widens on NSSR62 and NSSR90. In contrast, GRADE-IF and MapDiff can generalize better while considering the possibility of similar residue substitution. This suggests that diffusion-based models more effectively capture residue similarity in IPF prediction. For the TS50 dataset, MapDiff significantly improves state-of-the-art methods by 8.1% on NSSR62, and is the best model, achieving a recovery rate of 68%.

6.3.6 Foldability of Generated Protein Sequences

Table 6.3 Comparison of foldability quality for the generated sequences on the CATH 4.2 test set using AlphaFold2. The results are presented as mean±standard deviation. The best result for each metric is marked in bold and the second-best result is underlined.

Models	**pLDDT (↑)**	**PAE (↓)**	**PTM (↑)**	**TM-Score (↑)**	**RMSD (↓)**
ProteinMPNN	87.13±9.79	5.85±3.17	77.42±14.96	86.27±16.32	<u>3.08±4.25</u>
PiFold	87.42±9.82	5.81±3.22	77.75±15.03	<u>86.56±16.21</u>	3.10±4.29
LM-Design	<u>88.04±9.00</u>	<u>5.78±3.27</u>	<u>78.00±14.80</u>	85.36±16.98	3.54±5.00
GRADE-IF	85.32±9.27	6.30±3.10	75.63±13.80	85.80±14.93	3.11±3.96
MapDiff	**88.63±8.27**	**5.42±2.76**	**79.00±13.04**	**88.77±13.48**	**2.57±3.50**

Foldability is a crucial property that evaluates whether a protein sequence can fold into the desired structure. In this study, we evaluate the foldability of generated protein sequences by predicting their structures with AlphaFold2 and comparing the discrepancies against the native crystal structures. Table 6.3 presents five foldability metrics for the 1,120 structures in the CATH 4.2 test set. The results indicate that the generated protein sequences by MapDiff exhibit superior foldability, the highest confidence, and minimal discrepancy compared to their native structures. Notably, the foldability and sequence recovery results do not always positively correlate. For instance, while ProteinMPNN performs poorly in sequence recovery, it achieves the best RMSD among baseline methods. Therefore, it is essential to comprehensively evaluate IPF models from both sequence and structure perspectives.

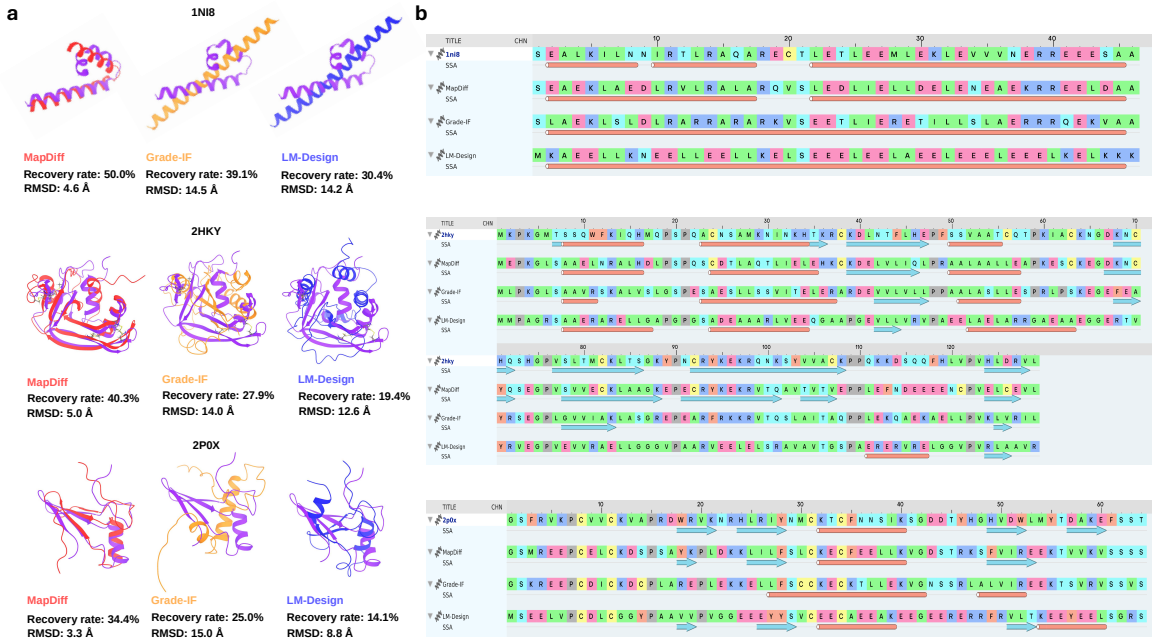

Fig. 6.3 Comparison of three refolded structures (left) and the respective model-designed sequences (right) for proteins with PDB IDs 1NI8, 2HKY, and 2P0X. **(a)** Refolded tertiary structure visualization of the sequences designed by three models MapDiff (red), GRADE-IF (orange) and LM-Design (blue). The refolded structures are generated by AlphaFold2 and superposed against the ground-truth structures (purple). For each model and structure, the recovery rate and RMSD value are indicated for foldability comparison. **(b)** The alignment of the three native sequences and the respective model-designed sequences. The results are shown with secondary structure elements marked below each sequence: α helices are in red, β strands are in blue, and loops and disordered regions are the rest. The refolded structures and alignments of the predicted protein sequences were visualized using the Schrödinger Maestro software (Schrödinger, LLC, 2023).

In Figure 6.3a, we illustrate exemplary 3D structures refolded by AlphaFold2 from IPF-derived sequences generated by MapDiff, GRADE-IF, and LM-Design for three different protein folds (PDB ID: 1NI8, 2HKY, 2P0X) with a pre-selected monomer pTM prediction argument. In addition to estimating the sequence recovery rate and foldability of derived 3D structures using the RMSD metric, we also inspect the alignment of native and generated sequences, including the agreement between refolded secondary structures and individual pairs of amino acids in Fig. 6.3b. The first example is a 46 amino acid-long monomer of the 1NI8 structure (purple) representing an N-terminal fragment of the H-NS dimerization

domain, a protein composed of three α helices that is involved in structuring the chromosome of Gram-negative bacteria, hence acting as a global regulator for the expression of different genes (Bloch et al., 2003). Two monomers form a homodimer which requires the presence of K^5, R^{11}, R^{14}, R^{18}, and K^{31} residues to engage in the prokaryotic DNA binding. MapDiff (red) managed to retrieve two out of the three α helices, with an interhelical turn present at the same position as in the original structure (A17-R18), whereas GRADE-IF (orange) and LM-Design (blue) models only consisted of a single continuous α helix. Moreover, MapDiff and GRADE-IF obtained four out of five (K^5, R^{11}, R^{14}, and R^{18}) amino acids required for DNA binding and LM-Design obtained none. MapDiff and LM-Design generate glutamic acid (E) and GRADE-IF isoleucine (I) which, in comparison to the corresponding positively charged K^{31} in the original structure, are negatively charged and neutral residues, respectively. The single continuous α helix displayed by GRADE-IF and LM-Design AlphaFold2 models hence produces much worse RMSD values (14.5 Åand 14.2 Å, respectively) than the MapDiff model, which retrieved two helices at the right positions (RMSD = 4.6 Å). Consistent with this, MapDiff obtained a 10% higher recovery rate than GRADE-IF and LM-Design.

The second example is the 2HKY structure of 109 amino-acid long human ubiquitous ribonuclease 7 (hRNase7) rich in positively charged residues, that possesses antimicrobial activity (Huang et al., 2007). This α/β mixed protein contains 22 cationic residues (18 K and 4 R) distributed into three surface-exposed clusters which promote binding to the bacterial membrane, thus rendering it permeable, which consequently elicits membrane disruption and death. In addition, it contains four disulfide bridges ($C^{24} - C^{82}$, $C^{38} - C^{92}$, $C^{56} - C^{107}$, and $C^{63} - C^{70}$) critical for its secondary and tertiary structure, three of which were successfully retrieved by MapDiff, whereas no cysteines were found in either GRADE-IF or LM-Design sequences. Furthermore, all secondary structure elements were nearly entirely recovered by MapDiff, unlike GRADE-IF and LM-Design solutions which contained little resemblance to the native structure, particularly in the C-terminus half. These structural findings were reflected in a fair recovery rate of 40.3% and a RMSD value of 5.0 Å for MapDiff, which was considerably better than in GRADE-IF and LM-Design structures (14.0 Åand 12.6 Å, respectively).

A third example displays AlphaFold2 refolded structures obtained from generated sequences with relatively low recovery rates that used 2P0X structure of an optimized non-biological (de novo) ATP-binding protein as a template (Mansy et al., 2007). Here, MapDiff retrieved all detected secondary structure elements, except for the C-terminus β-strand which was replaced by a loop. LM-Design was the second best with an α helix substituting the aforementioned β-strand. Even if nearly all secondary structure elements were retrieved by both MapDiff and LM-Design AlphaFold2 models, MapDiff model obtained by far the best RMSD (3.3 Åas opposed to 8.8 Å). Despite having a better recovery rate than LM-Design, GRADE-IF generated a sequence that folded poorly compared to the experimentally confirmed structure (15.0 Å).

In these cases, MapDiff achieved low RMSD values to successfully replicate the majority of secondary structure elements elucidated through experiments, including other structural features such as the disulfide bonds (2HKY) or positively charged residues that were suspected to participate in protein function (1NI8). In contrast, Grade-IF and LM-Design predicted sequences that not only had lower recovery rates than MapDiff but also exhibited partially or entirely absent secondary structure elements, as shown by the experimentally derived 3D structures, resulting in significantly worse RMSDs. Although the structures predicted by AlphaFold2 cannot entirely substitute the structural elucidation by experimental techniques such as X-ray or NMR (nuclear magnetic resonance), they provide the first glance at the foldability potential of de novo generated protein sequences by IPF models. A natural next step in future work would be to express the de novo designed protein sequences and experimentally determine their tertiary structures.

6.3.7 Model Analysis and Ablation Study

We perform a model analysis and ablation study to assess the effectiveness of key components in MapDiff. We specifically investigate the contributions of edge feature updating, node coordinate updating, and global context learning within the base sequence predictor (G-EGNN) to the model performance. Additionally, we examine the impact of the mask ratio adapter and the pre-trained IPA component in the residue refinement module on the

Table 6.4 Ablation study of the denoising network modules in MapDiff. We study five model variants and investigate how much performance decreases when key components are removed. The results indicate the model performance on CATH 4.2. The best result for each metric is marked in bold.

Module	Component	MapDiff	Variant 1	Variant 2	Variant 3	Variant 4	Variant 5
G-EGNN	EdgeUpdate	✓	✓	✓		✓	✓
	CoordinateUpdate	✓	✓	✓			✓
	GlobalContext	✓	✓		✓	✓	✓
Refinement	MaskAdapter	✓		✓	✓	✓	
	IPA	✓	✓	✓	✓	✓	
Results	Recovery (%)	**60.93**	58.64	59.76	58.38	60.16	56.46
	NSSR62 (%)	**78.57**	76.73	77.32	77.04	77.80	75.69
	NSSR90 (%)	**75.66**	73.52	74.82	74.24	75.02	72.58
Summary	Change	-	↓↓	↓↓	↓↓	↓	↓↓↓

predictions. As shown in Table 6.4, we study five variants of MapDiff, each with different key components removed, and compare their performance on the CATH 4.2 test set. The results show that each component positively enhances the sequence recovery performance. For instance, we observe that the IPA-based refinement mechanism (variant 5) achieves the most significant improvement, increasing recovery by 7.9%. Meanwhile, the global context learning and coordinate updating (variant 2, 4) in G-EGNN also improve the recovery by 1.2% and 1.9%, respectively.

In Figure 6.4, we analyze the scalability of MapDiff across denoising training and sampling inference modes. Figure 6.4a illustrates the runtimes of model training and inference against the protein sequence length. For each protein length, we conduct ten independent runs using a single Tesla A100 GPU. We empirically observe that the runtimes of MapDiff increase almost linearly with the protein sequence length. For instance, MapDiff takes around 0.13 seconds for a short protein sequence with 100 residues during denoising training, while the runtime reaches around 0.28 seconds for a long protein sequence with 500 residues. Figure 6.4b shows the peak GPU memory usage against the protein sequence length. Similar to the runtime, the memory usage exhibits a linear increase as the sequence length grows. During sampling inference, MapDiff only requires around 2GB of RAM for a protein sequence with 500 residues. The study highlights the scalability of MapDiff in both training and inference.

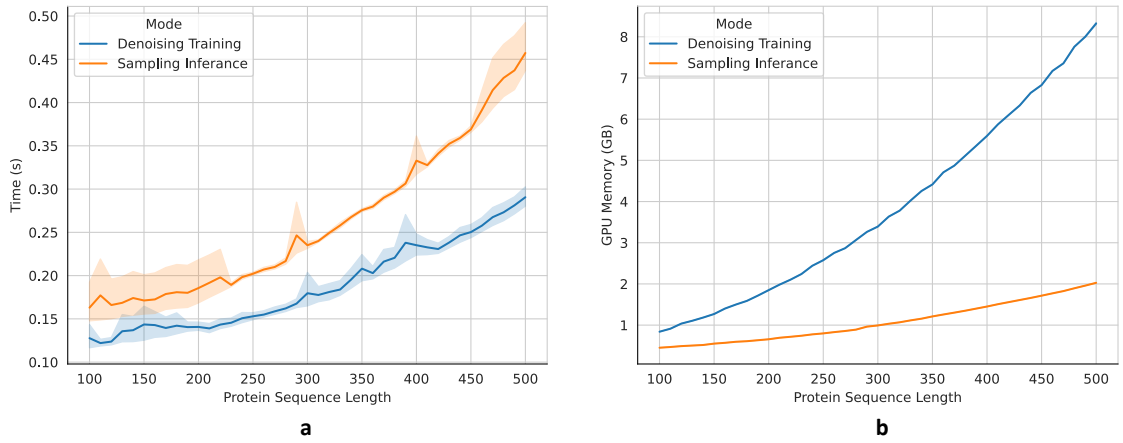

Fig. 6.4 Runtime and GPU memory usage of MapDiff. (a) Analysis of training and sampling runtimes for different protein sequence lengths, based on ten independent runs for each length. The sampling inference runtime represents the duration of one Monte-Carlo sample with DDIM (skipping steps=100) in MapDiff. The shaded area indicates the 95% confidence interval for the runtime estimation at each protein sequence length. (b) Correlation between GPU memory usage and protein sequence length during training and sampling modes.

In Figure 6.5, we present a sensitivity analysis of MapDiff with respect to the number of Monte-Carlo samples and DDIM skipping steps. Figure 6.5a demonstrates that the sequence recovery and NSSR62 significantly improve as the number of Monte Carlo samples increases. Notably, the model performance stabilizes and remains high only if the number of samples reaches 20. This finding indicates that enhancing prediction uncertainty through Monte-Carol dropout is an effective strategy for improving the diffusion-based generation process. Figure 6.5b illustrates the model performance against the number of DDIM skipping steps. To isolate the performance gain from Monte-Carlo dropout, we set the number of Monte-Carlo samples to 2 in this validation. The results show that even as the number of skipping steps increases, MapDiff achieves higher recovery compared to the model variant without IPA. This can be attributed to the prior knowledge provided by the pre-trained IPA component, which guides the diffusion trajectories toward more reliable sampling and consequently leads to more accurate sequence generation.

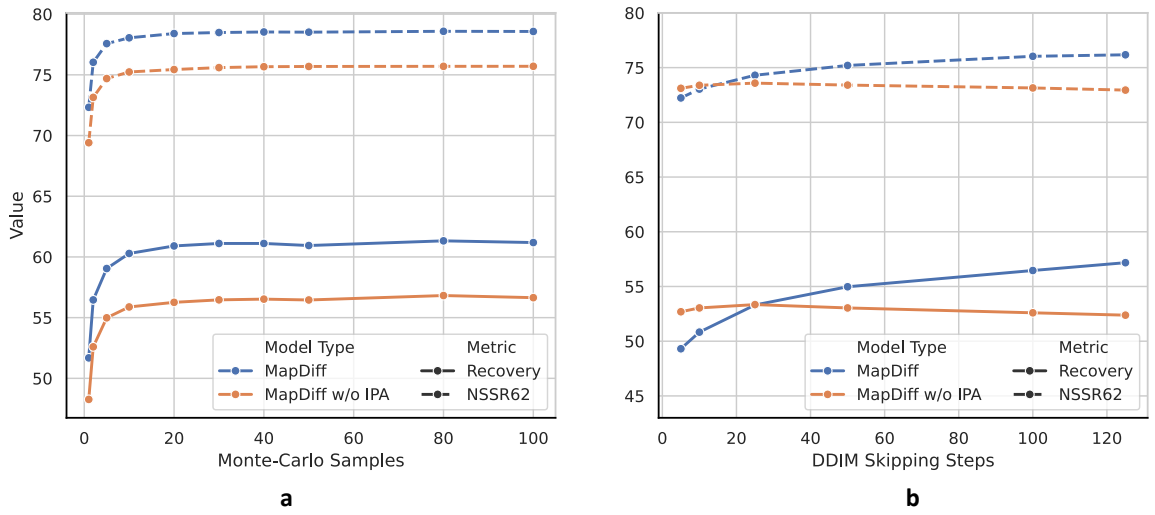

Fig. 6.5 Model sensitivity analysis with different Monte-Carol samples and DDIM skipping steps on CATH 4.2. (a) Median recovery and NSSR62 score in relation to the number of Monte Carlo samples. (b) Median recovery and NSSR62 score with respect to the number of DDIM skipping steps.

6.4 Summary

In this chapter, we present MapDiff, a mask prior-guided denoising diffusion framework for structure-based protein design. Specifically, we regard IPF prediction as a discrete denoising diffusion problem, and develop a graph-based denoising network to capture structural information and residue interactions. At each denoising step, we utilize a G-EGNN module to generate clean sequences from input structures and a pre-trained IPA module to refine low-confidence residues, ensuring reliable denoising trajectories. Moreover, we integrate DDIM with Monte-Carlo dropout to accelerate generative sampling and enhance uncertainty estimation. Experiments demonstrate that MapDiff consistently outperforms the state-of-the-art IPF models across multiple benchmarks and scenarios. At the same time, the generated protein sequences exhibit a high degree of similarity to their native counterparts. Even in cases where the overall sequence similarity was low, these sequences can often refold into their native structures, as demonstrated by the AlphaFold2-refolded models. We also conduct a comprehensive ablation study to analyze the importance of different model components for the prediction results. MapDiff demonstrates transferability and robustness

in generating novel protein sequences, even with limited training data. Promising future directions include verifying the applicability of MapDiff in practical domains such as de novo antibody design and protein engineering, as well as improving the model interpretability.

Chapter 7

Conclusion and Future Work

In this thesis, we have explored transferable representation learning in the context of drug discovery. As a result, we summarize the thesis by addressing the three research questions presented in Chapter 1 and discussing potential work worthy of further investigation in future.

7.1 Conclusion

Here, we aim to answer the proposed three research questions in Chapter 1.

- **Q1: How can we design a low-bias evaluation to fairly measure model transferability and reduce the gap with real performance in drug discovery?** As we have discussed, there always exists a significant gap between the performance reported in academic papers and that in practical drug discovery settings. The results in academic research often appear overly optimistic when applied to real-world industrial contexts. In Chapter 3, we analyze the common pitfalls that cause high-bias evaluations in DTI prediction. By eliminating hidden data biases and employing appropriate split strategies, we can enhance the distribution divergence between the training and test sets, thereby constructing a more challenging and realistic task with low-bias evaluation. We follow the proposed low-bias evaluation principles across different drug discovery applications. In Chapter 4, we develop a clustering-based pair split strategy to construct a cross-domain scenario in DTI prediction. Furthermore, scaffold split and topology

split strategies are employed to augment the distribution gap between training and test sets in molecular property prediction (in Chapter 5) and inverse protein folding (in Chapter 6), respectively. Thus, our evaluation methods exhibit low bias and reward more on model transferability rather than memorization.

- **Q2: How can we enhance specific transferability of deep models to novel drugs that are out of learned distribution?** Specific transferability refers to the capability of a deep model to effectively transfer learned knowledge from a source domain to a target domain. Due to the vast chemical space and limited drug-target data, enhancing a model's specific transferability is crucial for improving DTI prediction. To overcome this problem, we propose DrugBAN with an adversarial domain adaptation module in Chapter 4, which enhances generalization performance on out-of-distribution data. Our framework develops a bilinear attention-inspired network to capture substructure-level interactions between drug molecules and target proteins. Furthermore, the learned joint representations can be aligned across different distributions using conditional domain adversarial learning. Consequently, DrugBAN consistently generates specific transferable representations and achieves significant prediction performance in both source and target domain settings.

- **Q3: How can we design the self-supervised learning frameworks that can enhance generic transferability in both drug-related discriminative and generative tasks?** Generic transferability refers to the ability of a pre-trained deep model to apply its learned knowledge and representations across a wide range of downstream tasks. This thesis presents two self-supervised learning frameworks designed to improve the generic transferability of models in molecular property prediction and inverse protein folding. In Chapter 5, we develop Galformer, a dual-modality self-supervised learning framework for molecular representation learning. Galformer integrates discriminative 2D and 3D knowledge from large-scale unlabeled molecules, and can effectively adapt to discriminative molecular property predictions via fine-tuning. In Chapter 6, we propose MapDiff, a mask prior-guided denoising diffusion model designed for

inverse protein folding. We first develop a mask prior pre-training strategy to learn residue interactions and structural information from proteins, and then incorporate the pre-trained model into a conditional discrete denoising diffusion framework for structure-based protein sequence generation. Our results demonstrate the effectiveness in enhancing generic transferability for both discriminative and generative tasks in drug discovery.

7.2 Future Work

Although this thesis has addressed several fundamental problems of transferable representation learning in drug discovery, many open challenges still remain to be considered. Here, we point out three crucial research directions that are worth to be explored in future work.

- **3D structure-based DTI prediction.** In Chapter 3 and Chapter 4, we only focus on chemogenomics-based DTI prediction using a 1D protein sequence and 2D molecular graph as input. Given that the number of highly accurate 3D structured proteins accounts for only a small fraction of the known protein sequences, our proposed framework does not consider modelling with such structural information. Nevertheless, DeepMind's AlphaFold (Jumper et al., 2021) is making great progress in protein 3D structure prediction, recently generating 2 billion protein 3D structure predictions from 1 million species. Such progress opens doors for utilizing generated 3D structural information in chemogenomics-based DTI prediction. Following the idea of pairwise local interaction learning and domain adaptation, we believe that extending our method further on complex 3D structures can lead to even better performance and interpretability in future work.

- **Lead compound optimization.** Lead compound optimization involves refining and enhancing a lead compound molecule that has shown potential to become a successful drug candidate, which plays an important role in the early drug discovery and development process. The primary objectives of this process are to enhance the potency and

optimize the drug-like properties of lead compounds, while minimizing any potential adverse effects. Therefore, computational methods for molecular property prediction can significantly accelerate the optimization process and reduce high monetary costs. In Chapter 5, we demonstrate the effectiveness of our self-supervised learning framework in capturing structural and chemical information from unlabeled molecules. Compared to traditional molecular descriptors, data-driven molecular representations from self-supervised learning are more informative, and can adapt to different molecule-related tasks. We propose that one future direction should focus on applying deep molecular representation learning to the optimization of lead compounds, which could significantly enhance the identification of successful drug candidates.

- **De novo drug design.** De novo drug design involves the creation of new pharmaceutical compounds from scratch, tailored to interact with specific biological targets. This process is inherently multidisciplinary, integrating critical steps such as target identification, structure-based design and molecular modeling. In Chapter 6, we validate the significant performance achieved by denoising diffusion models in the context of inverse protein folding. Different from traditional biology methods, which usually modify existing compounds, de novo drug design employing deep generative methods can more accurately and efficiently explore the vast chemical space, potentially generating novel structures with the desired biological activity. We believe that incorporating deep generative methods, particularly the widely used denoising diffusion technique, into de novo drug design can significantly improve success rates and shorten development timelines.

- **Multi-objective Bayesian Experimental Design.** Multi-objective Bayesian Experimental Design (BED) (Rainforth et al., 2024; Chaloner and Verdinelli, 1995) provides a flexiable strategy to further improve data efficiency and experimental optimization in molecular and protein design tasks (Lambrinidis and Tsantili-Kakoulidou, 2021). By integrating the developed models for molecular property prediction or de novo design, with practical toolkits like NEXTorch (Wang et al., 2021a), multi-objective

BED can actively select candidate molecules or protein sequences that balance multiple criteria, including stability, potency and developability, while maximizing the expected information gain. This approach can guide iterative experimental validation more effectively with limited resources and accelerate the process of drug discovery that meet diverse design requirements.

References

Abbasi, K., Poso, A., Ghasemi, J., Amanlou, M., and Masoudi-Nejad, A. (2019). Deep transferable compound representation across domains and tasks for low data drug discovery. *Journal of Chemical Information and Modeling*, 59(11):4528–4539.

Abbasi, K., Razzaghi, P., Poso, A., Amanlou, M., Ghasemi, J. B., and Masoudi-Nejad, A. (2020). DeepCDA: deep cross-domain compound–protein affinity prediction through lstm and convolutional neural networks. *Bioinformatics*, 36(17):4633–4642.

Abbasi, K., Razzaghi, P., Poso, A., Ghanbari-Ara, S., and Masoudi-Nejad, A. (2021). Deep learning in drug target interaction prediction: current and future perspectives. *Current Medicinal Chemistry*, 28(11):2100–2113.

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. (2024). Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3.

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2021). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Association for Computational Linguistic*.

Akpinaroglu, D., Seki, K., Guo, A., Zhu, E., Kelly, M. J., and Kortemme, T. (2023). Structure-conditioned masked language models for protein sequence design generalize beyond the native sequence space. *bioRxiv*, pages 2023–12.

Alford, R. F., Leaver-Fay, A., Jeliazkov, J. R., O'Meara, M. J., DiMaio, F. P., Park, H., Shapovalov, M. V., Renfrew, P. D., Mulligan, V. K., Kappel, K., et al. (2017). The rosetta all-atom energy function for macromolecular modeling and design. *Journal of Chemical Theory and Computation*, 13(6):3031–3048.

Anand, N., Eguchi, R., Mathews, I. I., Perez, C. P., Derry, A., Altman, R. B., and Huang, P.-S. (2022). Protein sequence design with a learned potential. *Nature Communications*, 13(1):746.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Axelrod, S. and Gomez-Bombarelli, R. (2022). Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., et al. (2021). Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Bai, P., Miljković, F., John, B., and Lu, H. (2023). Interpretable bilinear attention network with domain adaptation improves drug–target prediction. *Nature Machine Intelligence*, 5(2):126–136.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *ICML Unsupervised and Transfer Learning*.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The protein data bank. *Nucleic Acids Research*, 28(1):235–242.

Bloch, V., Yang, Y., Margeat, E., Chavanieu, A., Augé, M. T., Robert, B., Arold, S., Rimsky, S., and Kochoyan, M. (2003). The h-ns dimerization domain defines a new fold contributing to dna recognition. *Nature Structural & Molecular Biology*, 10(3):212–218.

Bredel, M. and Jacoby, E. (2004). Chemogenomics: an emerging strategy for rapid target and drug discovery. *Nature Reviews Genetics*, 5:262–275.

Brempong, E. A., Kornblith, S., Chen, T., Parmar, N., Minderer, M., and Norouzi, M. (2022). Denoising pretraining for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4175–4186.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bryant, P., Pozzati, G., and Elofsson, A. (2022). Improved prediction of protein-protein interactions using alphafold2. *Nature Communications*, 13(1):1265.

Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Costanzo, L. D., Christie, C. H., Dalenberg, K., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranovic, V., Guzenko, D., Hudson, B. P., Kalro, T., Liang, Y., Lowe, R., Namkoong, H., Peisach, E., Periskova, I., Prlić, A., Randle, C., Rose, A. S., Rose, P. W., Sala, R., Sekharan, M., Shao, C., Tan, L., Tao, Y.-P., Valasatava, Y., Voigt, M., Westbrook, J. D., Woo, J., Yang, H., Young, J. Y., Zhuravleva, M., and Zardecki, C. (2019). Rcsb protein data bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Research*, 47:D464 – D474.

Buza, K. and Peka, L. (2017). Drug-target interaction prediction with bipartite local models and hubness-aware regression. *Neurocomputing*, 260:284–293.

Cai, D. and Lam, W. (2020). Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 34, pages 7464–7471.

Cao, D., Liu, S., Xu, Q., Lu, H., Huang, J.-H., Hu, Q.-N., and Liang, Y. (2012). Large-scale prediction of drug-target interactions using protein sequences and drug topological structures. *Analytica Chimica Acta*, 752:1–10.

Cao, D., Xu, Q., and Liang, Y. (2013). Propy: a tool to generate various modes of chou's pseaac. *Bioinformatics*, 29 7:960–2.

Cao, Y., Jiang, T., and Girke, T. (2008). A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics*, 24(13):i366–i374.

Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design: A review. *Statistical science*, pages 273–304.

Chan, H. C. S., Shan, H., Dahoun, T., Vogel, H., and Yuan, S. (2019). Advancing drug discovery via artificial intelligence. *Trends in Pharmacological Sciences*.

Chen, L., Tan, X., Wang, D., Zhong, F., Liu, X., Yang, T., Luo, X., Chen, K., Jiang, H., Zheng, M., and Elofsson, A. (2020a). TransformerCPI: improving compound-protein interaction prediction by sequence-based deep learning with self-attention mechanism and label reversal experiments. *Bioinformatics*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. (2009). BioPython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I., Courbet, A., de Haas, R. J., Bethel, N., et al. (2022). Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):224–227.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186.

Dwivedi, V. P. and Bresson, X. (2021). A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2023). Benchmarking graph neural networks. *ArXiv*, abs/2003.00982.

Fang, X., Liu, L., Lei, J., He, D., Zhang, S., Zhou, J., Wang, F., Wu, H., and Wang, H. (2022). Geometry-enhanced molecular representation learning for property prediction. *Nature Machine Intelligence*, pages 1–8.

Feng, Q., Dueva, E., Cherkasov, A., and Ester, M. (2018). PADME: A deep learning-based framework for drug-target interaction prediction. *arXiv preprint arXiv:1807.09741*.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. (2020). Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

Gamo, F.-J., Sanz, L. M., Vidal, J., De Cozar, C., Alvarez, E., Lavandera, J.-L., Vanderwall, D. E., Green, D. V., Kumar, V., Hasan, S., et al. (2010). Thousands of chemical starting points for antimalarial lead identification. *Nature*, 465(7296):305–310.

Ganea, O.-E., Huang, X., Bunne, C., Bian, Y., Barzilay, R., Jaakkola, T. S., and Krause, A. (2022). Independent SE(3)-equivariant models for end-to-end rigid protein docking. In *International Conference on Learning Representations*.

Ganin, Y. and Lempitsky, V. S. (2014). Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. S. (2016). Domain-adversarial training of neural networks. In *J. Mach. Learn. Res.*

Gao, K. Y., Fokoue, A., Luo, H., Iyengar, A., Dey, S., and Zhang, P. (2018). Interpretable drug target prediction using deep neural representation. In *International Joint Conference on Artificial Intelligence*, pages 3371–3377.

Gao, P., Jiang, Z., You, H., Lu, P., Hoi, S. C., Wang, X., and Li, H. (2019). Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6639–6648.

Gao, Z., Tan, C., and Li, S. Z. (2022). Alphadesign: A graph protein design method and benchmark on alphafolddb. *arXiv preprint arXiv:2202.01079*.

Gao, Z., Tan, C., and Li, S. Z. (2023). Pifold: Toward effective and efficient protein inverse folding. In *The Eleventh International Conference on Learning Representations*.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B., Vickers, R., Roberts, C., Tang, J., et al. (2021). Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 22(6):bbab159.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., et al. (2012). Chembl: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107.

Gayvert, K. M., Madhukar, N. S., and Elemento, O. (2016). A data-driven approach to predicting successes and failures of clinical trials. *Cell Chemical Biology*, 23(10):1294–1301.

Geppert, H., Humrich, J., Stumpfe, D., Gärtner, T., and Bajorath, J. (2009). Ligand prediction from protein sequence and small molecule information using support vector machines and fingerprint descriptors. *Journal of Chemical Information and Modeling*, 49 4:767–79.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR.

Gilson, M. K., Liu, T., Baitaluk, M., Nicola, G., Hwang, L., and Chong, J. (2016). BindingDB in 2015: a public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic Acids Research*, 44(D1):D1045–D1053.

Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International Conference on Machine Learning*.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.

Greg Landrum et al (2006). RDKit: Open-source cheminformatics. http://www.rdkit.org.

Hachmann, J., Olivares-Amaya, R., Atahan-Evrenk, S., Amador-Bedolla, C., Sánchez-Carrera, R. S., Gold-Parker, A., Vogt, L., Brockway, A. M., and Aspuru-Guzik, A. (2011). The harvard clean energy project: large-scale computational screening and design of

organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*.

Han, J.-D. J. (2008). Understanding biological functions through molecular networks. *Cell Research*, 18(2):224–237.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. (2020). Momentum contrast for unsupervised visual representation learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9726–9735.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919.

Hinnerichs, T. and Hoehndorf, R. (2021). DTI-Voodoo: machine learning over interaction networks and ontology-based background knowledge predicts drug–target interactions. *Bioinformatics*, 37:4835 – 4843.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hsu, C., Verkuil, R., Liu, J., Lin, Z., Hie, B., Sercu, T., Lerer, A., and Rives, A. (2022). Learning inverse folding from millions of predicted structures. In *International Conference on Machine Learning*, pages 8946–8970. PMLR.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2020). Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

Huang, J., Smola, A., Gretton, A., Borgwardt, K. M., and Schölkopf, B. (2006). Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*.

Huang, K., Xiao, C., Glass, L., and Sun, J. (2021). MolTrans: Molecular interaction transformer for drug–target interaction prediction. *Bioinformatics*, 37:830 – 836.

Huang, Y.-C., Lin, Y.-M., Chang, T.-W., Wu, S.-J., Lee, Y.-S., Chang, M. D.-T., Chen, C., Wu, S.-H., and Liao, Y.-D. (2007). The flexible and clustered lysine residues of human ribonuclease 7 are critical for membrane permeability and antimicrobial activity. *Journal of Biological Chemistry*, 282(7):4626–4633.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. (2019). Generative models for graph-based protein design. *Advances in Neural Information Processing Systems*, 32.

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2020). A survey on contrastive self-supervised learning. *ArXiv*, abs/2011.00362.

Jiang, J., Shu, Y., Wang, J., and Long, M. (2022). Transferability in deep learning: A survey. *ArXiv*, abs/2201.05867.

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. (2022). Torsional diffusion for molecular conformer generation. *Advances in Neural Information Processing Systems*, 35:24240–24253.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L., and Dror, R. (2020). Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*.

Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589.

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules*, 22(12):2577–2637.

Kao, P.-Y., Kao, S.-M., Huang, N.-L., and Lin, Y.-C. (2021). Toward drug-target interaction prediction via ensemble modeling and transfer learning. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2384–2391. IEEE.

Kim, J., Park, S., Min, D., and Kim, W. (2021). Comprehensive survey of recent drug discovery using deep learning. *International Journal of Molecular Sciences*, 22.

Kim, J.-H., Jun, J., and Zhang, B.-T. (2018). Bilinear Attention Networks. In *Advances in Neural Information Processing Systems*.

Kim, S., Chen, J., Cheng, T., et al. (2019). PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109.

Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *Advances in Neural Information Processing Systems*, 34:21696–21707.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kirkpatrick, P. and Ellis, C. (2004). Chemical space. *Nature*, 432(7019):823–824.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. (2021). Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629.

Lambrinidis, G. and Tsantili-Kakoulidou, A. (2021). Multi-objective optimization methods in novel drug design. *Expert opinion on drug discovery*, 16(6):647–658.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Lee, I., Keum, J., and Nam, H. (2019). DeepConv-DTI: Prediction of drug-target interactions via deep learning with convolution on protein sequences. *PLoS Computational Biology*, 15.

Li, B., Tian, J., Zhang, Z., Feng, H., and Li, X. (2020). Multitask non-autoregressive model for human motion prediction. *IEEE Transactions on Image Processing*, 30:2562–2574.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Li, H., Zhao, D., and Zeng, J. (2022a). Kpgt: Knowledge-guided pre-training of graph transformer for molecular property prediction. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Li, M., Zhou, J., Hu, J., Fan, W., Zhang, Y., Gu, Y., and Karypis, G. (2021). DGL-LifeSci: An open-source toolkit for deep learning on graphs in life science. *ACS Omega*.

Li, S., Zhou, J., Xu, T., Dou, D., and Xiong, H. (2022b). Geomgcl: Geometric graph contrastive learning for molecular property prediction. In *Proceedings of the Thirty-Six AAAI Conference on Artificial Intelligence*, pages 4541–4549.

Li, Z., Yang, Y., Faraggi, E., Zhan, J., and Zhou, Y. (2014). Direct prediction of profiles of sequences compatible with a protein structure by neural networks with fragment-based local and energy-based nonlocal profiles. *Proteins: Structure, Function, and Bioinformatics*, 82(10):2565–2573.

Lin, K., Wang, L., and Liu, Z. (2021). Mesh graphormer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12939–12948.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022). Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Liu, H., Sun, J., Guan, J., Zheng, J., and Zhou, S. (2015). Improving compound–protein interaction prediction by building up highly credible negative samples. *Bioinformatics*, 31:i221 – i229.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. (2022). Pre-training molecular graph representation with 3d geometry. In *International Conference on Learning Representations*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002.

Löffler, P., Schmitz, S., Hupfeld, E., Sterner, R., and Merkl, R. (2017). Rosetta: Msf: a modular framework for multi-state computational protein design. *PLoS Computational Biology*, 13(6):e1005600.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. (2018). Conditional Adversarial Domain Adaptation. In *NeurIPS*.

Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., Peng, J., Chen, L., and Zeng, J. (2017). A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nature Communications*, 8.

Lyu, S., Sowlati-Hashjin, S., and Garton, M. (2024). Variational autoencoder for design of synthetic viral vector serotypes. *Nature Machine Intelligence*, pages 1–14.

Ma, Y. and Tang, J. (2021). *Deep Learning on Graphs*. Cambridge University Press.

Mansy, S. S., Zhang, J., Kümmerle, R., Nilsson, M., Chou, J. J., Szostak, J. W., and Chaput, J. C. (2007). Structure and evolutionary analysis of a non-biological atp-binding protein. *Journal of Molecular Biology*, 371(2):501–513.

Marinka Zitnik, Rok Sosič, S. M. and Leskovec, J. (2018). BioSNAP Datasets: Stanford biomedical network dataset collection.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019). Provably powerful graph networks. *Advances in Neural Information Processing Systems*, 32.

Martínez-González, A., Villamizar, M., and Odobez, J.-M. (2021). Pose transformers (potr): Human motion prediction with non-autoregressive transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2276–2284.

Mayr, A., Klambauer, G., Unterthiner, T., Steijaert, M., Wegner, J., Ceulemans, H., Clevert, D.-A., and Hochreiter, S. (2018). Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical Science*, 9:5441 – 5451.

McInnes, L., Healy, J., and Astels, S. (2017). HDBSCAN: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205.

Mialon, G., Chen, D., Selosse, M., and Mairal, J. (2021). Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*.

Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. (2022). Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*.

Molecular Operating Environment (MOE) (2022). 2020.09 Chemical Computing Group ULC, 1010 Sherbooke St. West, Suite 910, Montreal, QC, Canada, H3A 2R7.

Nguyen, T., Le, H., Quinn, T., Nguyen, T. M., Le, T. D., and Venkatesh, S. (2021). GraphDTA: Predicting drug-target binding affinity with graph neural networks. *Bioinformatics*, 37(8):1140–1147.

Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR.

Ning, X., Rangwala, H., and Karypis, G. (2009). Multi-assay-based structure-activity relationship models: Improving structure-activity relationship models by incorporating activity information from related targets. *Journal of Chemical Information and Modeling*, 49 11:2444–56.

O'Connell, J., Li, Z., Hanson, J., Heffernan, R., Lyons, J., Paliwal, K., Dehzangi, A., Yang, Y., and Zhou, Y. (2018). Spin2: Predicting sequence profiles from protein structures using deep neural networks. *Proteins: Structure, Function, and Bioinformatics*, 86(6):629–633.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., and Thornton, J. M. (1997). Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Öztürk, H., Olmez, E. O., and Özgür, A. (2018). DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics*, 34:i821 – i829.

Öztürk, H., Ozkirimli, E., and Özgür, A. (2019). WideDTA: prediction of drug-target binding affinity. *arXiv preprint arXiv:1902.04166*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830.

Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., and Ma, J. (2022). Pocket2mol: Efficient molecular sampling based on 3d protein pockets. In *International Conference on Machine Learning*, pages 17644–17655. PMLR.

Perlman, L., Gottlieb, A., Atias, N., Ruppin, E., and Sharan, R. (2011). Combining drug and gene similarity measures for drug-target elucidation. *Journal of Computational Biology*, 18(2):133–145.

Podlewska, S. and Kafel, R. (2018). Metstabon—online platform for metabolic stability predictions. *International Journal of Molecular Sciences*, 19(4):1040.

Pyzer-Knapp, E. O., Pitera, J. W., Staar, P. W., Takeda, S., Laino, T., Sanders, D. P., Sexton, J., Smith, J. R., and Curioni, A. (2022). Accelerating materials discovery using artificial intelligence, high performance computing and robotics. *npj Computational Materials*, 8(1):84.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Rainforth, T., Foster, A., Ivanova, D. R., and Bickford Smith, F. (2024). Modern bayesian experimental design. *Statistical Science*, 39(1):100–114.

Riley, P. F. (2019). Three pitfalls to avoid in machine learning. *Nature*, 572:27–29.

Rogers, D. and Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. (2020). Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR.

Schenone, M., Dancík, V., Wagner, B. K., and Clemons, P. A. (2013). Target identification and mechanism of action in chemical biology and drug discovery. *Nature Chemical Biology*, 9 4:232–40.

Schneuing, A., Du, Y., Harris, C., Jamasb, A., Igashov, I., Du, W., Blundell, T., Lió, P., Gomes, C., Welling, M., et al. (2022). Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*.

Scholkopf, B., Sung, K.-K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. (1997). Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765.

Schrödinger, LLC (2023). *Schrödinger Release 2023-4: Maestro, Schrödinger, LLC, New York, NY, 2023*.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9).

Sliwoski, G., Kothiwale, S., Meiler, J., and Lowe, E. W. (2014). Computational methods in drug discovery. *Pharmacological Reviews*, 66(1):334–395.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Song, J., Meng, C., and Ermon, S. (2021). Denoising diffusion implicit models. In *International Conference on Learning Representations*.

Song, L. and Dai, B. (2013). Robust low rank kernel embeddings of multivariate distributions. In *NIPS*.

Song, L., Huang, J., Smola, A., and Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *ICML*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Stärk, H., Beaini, D., Corso, G., Tossou, P., Dallago, C., Gunnemann, S., and Lio', P. (2022). 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*.

Starr, T. N. and Thornton, J. W. (2016). Epistasis in protein evolution. *Protein Science*, 25(7):1204–1218.

Subramonian, A. (2021). Motif-driven contrastive learning of graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15980–15981.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P., and Kawanabe, M. (2007). Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems*, 20.

Sun, M., Xing, J., Wang, H., Chen, B., and Zhou, J. (2021). Mocl: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Suresh, S., Li, P., Hao, C., and Neville, J. (2021). Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:15920–15933.

Tan, C., Gao, Z., Xia, J., Hu, B., and Li, S. Z. (2023). Global-context aware generative protein design. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. (2021). Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*.

Torrey, L., Shavlik, J., Walker, T., and Maclin, R. (2010). Transfer learning via advice taking. *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski*, pages 147–170.

Towse, C.-L. and Daggett, V. (2012). When a domain is not a domain, and why it is important to properly filter proteins in databases: conflicting definitions and fold classification systems for structural domains make filtering of such databases imperative. *Bioessays*, 34(12):1060–1069.

Tsubaki, M., Tomii, K., and Sese, J. (2019). Compound-protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35:309–318.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations*.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. (2022). Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. (2019). Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Wang, Y., Chen, T.-Y., and Vlachos, D. G. (2021a). Nextorch: a design and bayesian optimization toolkit for chemical sciences and engineering. *Journal of Chemical Information and Modeling*, 61(11):5312–5319.

Wang, Y., Wang, J., Cao, Z., and Farimani, A. B. (2021b). Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4:279–287.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. (2023). De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100.

Wei, X., Zhang, T., Li, Y., Zhang, Y., and Wu, F. (2020). Multi-modality cross attention network for image and sentence matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10941–10950.

Weininger, D. (1988a). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36.

Weininger, D. (1988b). SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36.

Weisfeiler, B. and Lehman, A. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16.

Wishart, D. S., Feunang, Y. D., Guo, A., et al. (2018). DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46:D1074 – D1082.

Wishart, D. S., Knox, C. K., Guo, A., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., and Hassanali, M. (2008). DrugBank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Research*, 36:D901 – D906.

Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J. E., and Stoica, I. (2021a). Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279.

Wu, Z., Johnston, K. E., Arnold, F. H., and Yang, K. K. (2021b). Protein sequence design with deep generative models. *Current Opinion in Chemical Biology*, 65:18–27.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018a). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018b). Unsupervised feature learning via non-parametric instance discrimination. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., et al. (2019). Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*, 63(16):8749–8760.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.

Xu, Y., Song, Y., Garg, S., Gong, L., Shu, R., Grover, A., and Ermon, S. (2021). Anytime sampling for autoregressive models via ordered autoencoding. In *International Conference on Learning Representations*.

Yamanishi, Y., Araki, M., Gutteridge, A., Honda, W., and Kanehisa, M. (2008). Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24:i232 – i240.

Yang, F., Wang, W., Wang, F., Fang, Y., Tang, D., Huang, J., Lu, H., and Yao, J. (2022a). scbert as a large-scale pretrained deep language model for cell type annotation of single-cell rna-seq data. *Nature Machine Intelligence*, 4(10):852–866.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. (2019). Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Cui, B., and Yang, M.-H. (2022b). Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*.

Yao, S., Wang, T., and Wan, X. (2020). Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154.

Yi, H.-C., You, Z.-H., Huang, D.-S., and Kwoh, C. K. (2022). Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1):bbab340.

Yi, K., Zhou, B., Shen, Y., Liò, P., and Wang, Y. (2024). Graph denoising diffusion for inverse protein folding. *Advances in Neural Information Processing Systems*, 36.

Yim, J., Trippe, B. L., De Bortoli, V., Mathieu, E., Doucet, A., Barzilay, R., and Jaakkola, T. (2023). Se(3) diffusion model with application to protein backbone generation. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021). Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.

You, Y., Chen, T., Shen, Y., and Wang, Z. (2021). Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823.

Yu, L., Qiu, W., Lin, W., Cheng, X., Xiao, X., and Dai, J. (2022a). Hgdti: predicting drug–target interaction by using information aggregation based on heterogeneous graph neural network. *BMC Bioinformatics*, 23(1):126.

Yu, P., Xie, S., Ma, X., Jia, B., Pang, B., Gao, R., Zhu, Y., Zhu, S., and Wu, Y. N. (2022b). Latent diffusion energy-based model for interpretable text modelling. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162, pages 25702–25720. PMLR.

Yu, Z., Yu, J., Xiang, C., Fan, J., and Tao, D. (2018). Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Trans. Neural Netw. Learn. Syst.*, 29:5947–5959.

Zhang, J., Zhang, H., Xia, C., and Sun, L. (2020). Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Zhang, Z., Liu, Q., Wang, H., Lu, C., and Lee, C.-K. (2021). Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems*, 34:15870–15882.

Zhao, J., Li, C., Wen, Q., Wang, Y., Liu, Y., Sun, H., Xie, X., and Ye, Y. (2021a). Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*.

Zhao, N., Wu, Z., Lau, R. W. H., and Lin, S. (2021b). What makes instance discrimination good for transfer learning? *ArXiv*, abs/2006.06606.

Zheng, S., Li, Y., Chen, S., Xu, J., and Yang, Y. (2020). Predicting drug–protein interaction using quasi-visual question answering system. *Nature Machine Intelligence*, 2:134–140.

Zheng, Z., Deng, Y., Xue, D., Zhou, Y., Ye, F., and Gu, Q. (2023). Structure-informed language models are protein designers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

Zhou, X., Chen, G., Ye, J., Wang, E., Zhang, J., Mao, C., Li, Z., Hao, J., Huang, X., Tang, J., et al. (2023). Prorefiner: an entropy-based refining strategy for inverse protein folding with global graph attention. *Nature Communications*, 14(1):7434.

Zhu, J., Xia, Y., Wu, L., Xie, S., Qin, T., Zhou, W., Li, H., and Liu, T.-Y. (2022a). Unified 2d and 3d pre-training of molecular representations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2626–2636.

Zhu, J., Xia, Y., Wu, L., Xie, S., Zhou, W., Qin, T., Li, H., and Liu, T.-Y. (2023). Dual-view molecular pre-training. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3615–3627.

Zhu, Y., Du, Y., Wang, Y., Xu, Y., Zhang, J., Liu, Q., and Wu, S. (2022b). A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*. PMLR.

Appendix A

Proofs and Algorithms

A.1 Discrete Posterior Distribution

We present the derivations of the discrete posterior distribution in the reverse denoising process. To clarify the derivation in Equation 6.11, we remove the residue index i and assume $\mathbf{x}_t = \mathbf{x}_t^i$, as it is not relevant to the theoretical derivation.

Propsition 1. For $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ defined in Equation 6.3, we have:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \propto \mathbf{x}_t \mathbf{Q}_t^T \odot \mathbf{x}_0 \overline{\mathbf{Q}}_{t-1}, \tag{A.1}$$

Proof. By Bayesian theorem, the distribution can be written as:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \propto q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1} \mid \mathbf{x}_0). \tag{A.2}$$

Due to the Markov property, $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0)$ is equivalent to $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. Similarly, we have:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \propto q(\mathbf{x}_{t-1} \mid \mathbf{x}_t) q(\mathbf{x}_t). \tag{A.3}$$

Given that $q(\mathbf{x}_t)$ is an independent distribution, it can be viewed as a normalizing constant. By the definition of the discrete transition matrix $\mathbf{Q}_t$, we can derive:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) = \text{Cat}(\mathbf{x}_{t-1}; \mathbf{p} = \mathbf{x}_0 \overline{\mathbf{Q}}_{t-1}), \tag{A.4}$$

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \text{Cat}(\mathbf{x}_{t-1}; \mathbf{p} = \mathbf{x}_t \mathbf{Q}_t^T). \tag{A.5}$$

By integrating the above terms, we obtain $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \propto \mathbf{x}_t \mathbf{Q}_t^T \odot \mathbf{x}_0 \overline{\mathbf{Q}}_{t-1}$ as intended.

Propsition 2. For $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0)$ defined in Equation 6.11, we have:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0) = \text{Cat}(\mathbf{x}_{t-1}; \mathbf{p} = \frac{\mathbf{x}_t \mathbf{Q}_t^T \odot \hat{\mathbf{x}}_0 \overline{\mathbf{Q}}_{t-1}}{\hat{\mathbf{x}}_0 \overline{\mathbf{Q}}_t \mathbf{x}_t^T}). \tag{A.6}$$

Proof. By Bayesian theorem, we write the distribution as:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0) = \frac{q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \hat{\mathbf{x}}_0) q(\mathbf{x}_{t-1} \mid \hat{\mathbf{x}}_0)}{q(\mathbf{x}_t \mid \hat{\mathbf{x}}_0)}. \tag{A.7}$$

According to the definition of the reverse denoising process, both random variables $X_0 = \mathbf{x}_t$ and $X_t = \hat{\mathbf{x}}_0$ are observed. Here, $\mathbf{x}_t$ represents the discrete noise state at step t, while $\hat{\mathbf{x}}_0$ denotes the predicted clean state by the denoising neural network. Therefore, we can rewrite Equation A.8 in terms of the observed variables:

$$q(X_{t-1} \mid X_t = \mathbf{x}_t, X_0 = \hat{\mathbf{x}}_0) = \frac{q(X_t = \mathbf{x}_t \mid \mathbf{x}_{t-1}, X_0 = \hat{\mathbf{x}}_0) q(\mathbf{x}_{t-1} \mid X_0 = \hat{\mathbf{x}}_0)}{q(X_t = \mathbf{x}_t \mid X_0 = \hat{\mathbf{x}}_0)}. \tag{A.8}$$

Due to the Markov property, we have:

$$q(X_{t-1} \mid X_t = \mathbf{x}_t, X_0 = \hat{\mathbf{x}}_0) = \frac{q(X_t = \mathbf{x}_t \mid X_{t-1}) q(\mathbf{x}_{t-1} \mid X_0 = \hat{\mathbf{x}}_0)}{q(X_t = \mathbf{x}_t \mid X_0 = \hat{\mathbf{x}}_0)}. \tag{A.9}$$

Now we can individually define each term in Equation A.9. Regarding the distribution $q(X_t = \mathbf{x}_t \mid X_{t-1})$, it is important to note that the value of X_{t-1} is not observed. Consequently, it is unfeasible to compute a determined result for the conditional distribution. Instead, we need to enumerate all possible $\mathbf{x}_{t-1}$ to compute the probability distribution over $\mathbf{x}_t$, defined as follows:

$$q(X_t = \mathbf{x}_t \mid X_{t-1}) = \mathbf{x}_t [\mathbf{I}_k \mathbf{Q}_t]^T, \tag{A.10}$$

$$= \mathbf{x}_t \mathbf{Q}_t^T. \tag{A.11}$$

For the two terms $q(\mathbf{x}_{t-1} \mid X_0 = \hat{\mathbf{x}}_0)$ and $q(X_t = \mathbf{x}_t \mid X_0 = \hat{\mathbf{x}}_0)$, we can extend them following the definition of the forward diffusion process:

$$q(\mathbf{x}_{t-1} \mid X_0 = \hat{\mathbf{x}}_0) = \hat{\mathbf{x}}_0 \overline{\mathbf{Q}}_{t-1}, \tag{A.12}$$

$$q(X_t = \mathbf{x}_t \mid X_0 = \hat{\mathbf{x}}_0) = \hat{\mathbf{x}}_0 \overline{\mathbf{Q}}_t \mathbf{x}_t^T. \tag{A.13}$$

By putting them all together, we obtain:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0) = \text{Cat}(\mathbf{x}_{t-1}; \mathbf{p} = \frac{\mathbf{x}_t \mathbf{Q}_t^T \odot \hat{\mathbf{x}}_0 \overline{\mathbf{Q}}_{t-1}}{\hat{\mathbf{x}}_0 \overline{\mathbf{Q}}_t \mathbf{x}_t^T}). \tag{A.14}$$

Appendix B

Additional Experimental Details

B.1 Molecular Property Datasets

As described in Section 5.3.1, we leverage fourteen labeled molecular property datasets from MoleculeNet (Wu et al., 2018a) and other public sources (Gaulton et al., 2012; Podlewska and Kafel, 2018; Gamo et al., 2010; Hachmann et al., 2011) as downstream prediction tasks, including nine classification datasets and five regression datasets.

- **BBBP** is a dataset about the blood-brain barrier penetration, and it has binary labels indicating the permeability property.
- **Sider** is a collection of marketed drugs and their adverse drug reactions. The drug side-effects are grouped into 27 system organ classes following MedDRA classifications.
- **ClinTox** is a dataset containing drugs approved by the FDA but eliminated from clinical trials owing to their toxicity (Gayvert et al., 2016). It has two classification tasks that indicate the clinical trial toxicity and the FDA approval status.
- **BACE** provides qualitative (binary label) binding results for a set of inhibitors targeting human β-secretase 1 (BACE-1).
- **Tox21** is a public database measuring the toxicity of compounds on 12 different targets.
- **MUV** is the maximum unbiased validation dataset selected from PCBA. It consists of 17 challenging tasks and is exclusively designed to validate of virtual screening techniques.
- **HIV** is a dataset obtained from the Drug Therapeutics Program AIDS Antiviral screen. It contains compounds that were tested for their ability to inhibit HIV replication.

- **Estrogen** is a dataset derived from ChEMBL (Gaulton et al., 2012) and consists of molecules with activities towards estrogen receptors.
- **MetStab** is a collection of molecules that have specific half-life measurements within an organism (Podlewska and Kafel, 2018).
- **ESOL** dataset measures the water solubility of molecules.
- **Freesolv** is a collection of experimental and calculated hydration free energies of small molecules in water.
- **Lipo** is a dataset that provides experimental results of the octanol/water distribution coefficient for molecules.
- **Malaria** is a dataset measuring the efficacy of drugs against the parasite that causes malaria (Gamo et al., 2010).
- **CEP** is a dataset from the Harvard Clean Energy Project (Hachmann et al., 2011) that measures the organic photovoltaic efficiency of molecules.