Graph Neural Network for Molecular Structure: Application in HIV Inhibitor Molecule Prediction

Quynh Nguyen[1]

Department of Finance and Banking

Abstract. The application of Graph Neural Networks (GNNs) in computational chemistry provides a powerful approach to modeling and predicting the properties of molecular compounds. GNNs represent atoms as nodes and bonds as edges, capturing the complex interactions within molecular graphs. This approach offers a robust method for predicting chemical properties, including molecular stability, reactivity, and toxicity. In this paper, we explore various GNN architectures and their ability to generalize across different molecular datasets, such as QM9 and MoleculeNet. As a specific application, we propose a novel framework that utilizes GNNs to predict and identify potential HIV inhibitor molecules by analyzing their graph-based representations. This research aims to contribute to the discovery and design of effective HIV inhibitors, offering a promising direction for future antiviral drug development.

Keywords: Deep learning · Molecules · reinforcement learning · Graph neural network (GNNs) · Graph autoencoder (GAE) · MolGAN

1 Introduction

In recent years, the development of machine learning (ML) methods has significantly advanced the biomedical field. A key area of focus within biomedical fields where ML and deep learning (DL) methods have been extensively studied is molecular structures, which are central to numerous biomedical applications, including drug discovery and prediction. Understanding and predicting the properties of molecular structures is crucial for identifying new therapeutic compounds, designing effective drugs, and advancing personalized medicine. As a result, there has been a surge in the application of ML/DL techniques, particularly those that can model and analyze the intricate details of molecular interactions and behaviors. For instance, basic ML algorithms like Support Vector Machines (SVM) and XGBoost have been explored in the context of molecular structure analysis to predict various molecular properties and behaviors [8].

Beyond these foundational methods, more advanced techniques have been developed for molecular structure-related problems. For example, [1] developed a reinforcement learning (RL) framework in a continuous setting, which is based on a stochastic parametrized Hamiltonian version of the Pontryagin maximum principle (PMP), to solve complex problems like side-chain packing and protein

folding. Additionally, Thomas Kipf combined generative AI mechanisms with reinforcement learning to create [4] **MolGAN**, an implicit, likelihood-free generative model for small molecular graphs that can generate molecules with specific desired chemical properties. Kipf and Welling [10] also study other graph neural networks such as variational graph auto-encoders and their possible application to molecular structures.

Among these advanced and novel frameworks, one of the most popular and effective methods for capturing and investigating the complex structures of molecules is the Graph Neural Network (GNN). GNNs have proven to be especially powerful in the biomedical domain and have been extensively studied [12]. For instance, [15], [2], and [14] have utilized GNNs to develop knowledge-enhanced models for information extraction from biomedical texts, demonstrating the versatility of GNNs beyond purely molecular applications.

In this work, we explore various GNN architectures and their ability to generalize across different molecular datasets. Moreover, we also propose a novel framework leveraging GNNs to predict and identify potential HIV inhibitor molecules via their graph-based representations. This research seeks to contribute to the discovery and design of effective HIV inhibitors, paving the way for new directions in antiviral drug development.

2 Background on autoencoder and reinforcement learning

2.1 Variational autoencoder

We wish to learn an encoder that map our data $\mathbf{x}$ to a continuous latent variable $\mathbf{z}$, and a decoder that maps back $\mathbf{z}$ to $\mathbf{x}$. The variational autoencoder [9] provides a formulation in which the encoding $\mathbf{z}$ is interpreted as a latent variable in a probabilistic generative model; a probabilistic decoder is defined by a likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$ parameterized by θ, while the encoder represented by $q_\phi(\mathbf{z}|\mathbf{x})$ attempts to approximate the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ while at the same time maintains a certain nice functional form (nice function q_ϕ) to make sampling $\mathbf{x}$ via $\mathbf{z}$ simpler. Using variational Bayes approach, this task can be done by maximizing the evidence lower bound (ELBO):

$$\mathcal{L}(\phi, \theta; \mathbf{x}) = \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \tag{1}$$

2.2 Generative adversarial network

Generative adversarial network (GAN) [6] is a neural network architecture that is used for data generation and consists of a generator G and a discriminator D. G_θ is a generator parameterized by θ and is trained to produce synthetic data x from certain latent variable z that can be easier to sample. On the other hand, discriminator D try to differentiate between real sample and sample generated from G. In other words, D and G play the minimax game $\min_G \max_D V(D, G)$ with value function $V(G, D)$:

$$V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

2.3 Reinforcement learning

In the reinforcement learning, we consider an agent interacting with the environment to learn the best action to take. At time t, the agent choose an action $a_t \in \mathcal{A}$ according to its policy $\pi_\theta(a|s_t)$ parameterized by θ given its current state $s_t \in \mathcal{X}$. The environment then produces a reward $r(s_t, a_t)$ and transition to the next states s_{t+1} according to the transition probability $\mathbb{P}(s_{t+1}|s_t, a_t)$. The goal of the agent is to maximize the expected λ-discounted cumulative return $\mathcal{J}(\theta) = \mathbb{E}_\pi[R_t] = \mathbb{E}_\pi[\sum_{i\geq 0} \gamma^i r(s_{t+i}, a_{t+i})]$. In the policy gradient methods, we directly parameterize a policy $\pi_\theta(a|s_t)$ and update its parameter θ so as to maximize the objective function $\mathcal{J}(\theta)$. There are many popular reinforcement learning algorithms. Two of such algorithms are REINFORCE [20] and DDPG [13]

3 Graph Neural Network (GNN)

3.1 General framework

Definition 1 *A graph is represented by $G = (V, E)$, where V is the set of vertices or nodes, and E is the set of nodes. Let $v_i \in V$ to denote a node and $e_{ij} = (v_i, v_j) \in E$ to denote an edge pointing from v_i to v_j. The neighborhood of a node v is defined as $N(v) = \{u \in V : (v, u) \in E\}$. The adjacency matrix $\mathbf{A}$ is a $n \times n$ matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. A graph may have node attribute $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is a node feature matrix with rows $\mathbf{x}_v \in \mathbb{R}^d$ representing the feature vector of a node v. A graph may also have edge attribute $\mathbf{X}^e \in \mathbb{R}^{m\times c}$ with feature vector $\mathbf{x}^e_{v,u} \in \mathbb{R}^c$ of an edge (v, u).*

The main idea of graph neural network (GNN) is that instead of taking an usual input vector, a GNN will take a graph G defined above as the input. In this way, GNN can deal with graph data, where not only content but relation between nodes are also important (see Figure 1).

3.2 Convolutional graph neural networks (ConvGNNs)

This type of network generalizes the usual convolutional network on grid (or image) data. The main idea is to generate a node v's representation from its own feature $\mathbf{x}_v$ and neighbors' features $\mathbf{x}_u$, where $u \in N(v)$. There are, in fact, two main types of ConvGNNs: spectral-based ConvGNN and spatial-based ConvGNN. The first type depend on the spectral represenation of the normalized Laplacian matrix of the graph. We will focus on the second type instead as it allows different graph structure G and is a direct generalization of the popular convolutional neural network (CNN) on graph data. Many spatial-based ConvGNNs follows the architecture of the general message-passing neural network (MPNN). The convolutional layers of MPNN can be described by the following equation:

$$\mathbf{h}_v^{(k)} = U_k \left(\mathbf{h}_v^{(k-1)}, \sum_{u\in N(v)} M_k(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{x}^e_{vu}) \right) \tag{3}$$

Here each $\mathbf{h}^{(k)}$ is the hidden vector at layer k whose entries are vectors $\mathbf{h}_v^{(k)}$ for nodes v in V. $U_k(.)$ and $M_k(.)$ are function with learnable parameters, and there parameter x_{vu}^e is an optional argument for M_k and is the edge feature vector described earlier. Also, note that $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ (zeroth hidden layer is actually input).

The network can consist of an optional readout layer that give a representation for the entire graph as a vector $\mathbf{h}_G$ based on the final hidden node layer:

$$\mathbf{h}_G = R(\mathbf{h}_v^{(K)} | v \in V) \tag{4}$$

where $R(.)$ is the readout function with learnable parameter.

Finally, similar to the pooling layer of CNNs that follows convolutional layers to downsample the image, the pooling layer of GNNs also has the form:

$$\mathbf{h}_G = \text{mean/max/sum}(\mathbf{h}_1, \mathbf{h}_2, \cdots \mathbf{h}_n) \tag{5}$$

where $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \cdots \mathbf{h}_n)$ is the hidden variable of the convolutional layer right before this pooling layer.

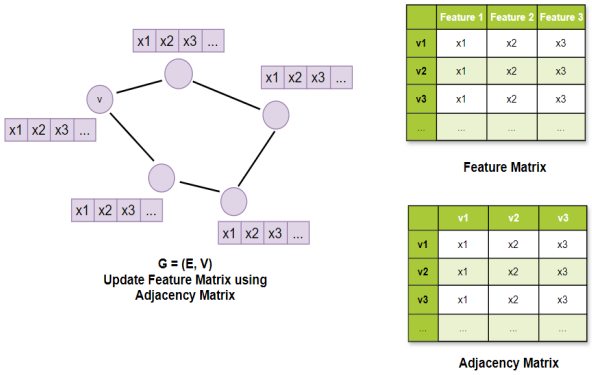

Fig. 1. Mechanism of Graph Neural Network (GNN)

3.3 Variational graph auto-encoders

ConvGNNs described in the last section are building block to construct the so-called varational graph auto-encoders (VGAE). This neural network is similar to variational neural network (VAE) [9], but with input being a graph data $(\mathbf{X}, \mathbf{A})$ where $\mathbf{A}$ is the adjacency matrix and $\mathbf{X}$ is the node feature matrix. Similar to VAE, VGAE also consist of two parts, which are inference and generative models:

Inference model:

$$q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i | \boldsymbol{\mu}_i, diag(\boldsymbol{\sigma}_i^2)) \tag{6}$$

where $\boldsymbol{\mu} = \text{GNN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$, $\log \boldsymbol{\sigma} = \text{GNN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$ so that $\text{GNN}_{\boldsymbol{\mu}}$ and $\text{GNN}_{\boldsymbol{\sigma}}$ are two-layer graph neural networks that share the first layer parameters.

Generative model:

$$p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{n} \prod_{j=1}^{n} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) \tag{7}$$

$$p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^T \mathbf{z}_j) \tag{8}$$

where σ is the sigmoid function.

Finally, similar to the VAE, the goal of VGAE is to optimize the variational lower bound $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] - KL[q(\mathbf{Z}|\mathbf{X}, \mathbf{A})||p(\mathbf{Z})] \tag{9}$$

Here the prior distribution $p(\mathbf{Z})$ can be taken to be Gaussian.

4 Previous works on applying autoencoder to molecule generation problem

4.1 Molecules representation and desired properties

Molecules representation We mention here two important data representations of a molecule:

1. SMILES string representation of molecule. Simplified molecular-input line-entry system (SMILES) refers to a line notation for encoding molecular structures. Each SMILES string encodes one particular molecular structure:
   (a) Atom are represented by standard abbreviation of the chemical elements. They are put inside brackets except common elements such as B, C, N, O, P, S, F, Cl, Br, or I.
   (b) A bond is represented using one of the symbols: ., -, =, #, $, :, / or \.
   (c) Ring structure is represented by putting labels to show connectivity between non-adjacent atoms (in SMILES string). For example, C1CCCC2-C1CCCC2 represents two ring structure here.
   (d) SMILES string also show many other properties and structures of a molecule such as aromaticity, branching, stereochemistry or isotopes
2. One can also represent molecule as a graph: its atoms can be considered as nodes in a graph, and edges correspond to bonds between atoms inside a molecule. One can reconstruct the molecule and its properties from the graph representation. Cheminformatics packages such as RDKit [17] provides a way to do this task.

Properties Here are important properties that when one tries to choose a desirable drug-like molecule

– Solubility: a property that measures how likely a molecule is able to mix with water, also known as the water octanol partition coefficients (LogP).

- Synthetizability: estimates how hard (0) or how easy (1) it is to synthesize a given molecule
- Druglikeness: how likely a molecule is a viable candidate for a drug, an estimate the captures the abstract notion of aesthetics in medicinal chemistry. This property is correlated to the previous two metrics.

4.2 Character variational autoencoder

Character variational autoencoder (CVAE) [5] uses VAE to learn a latent representation for SMILES strings of molecules. Two databases considered are ZINC and QM9. Both encoder and decoder are RNNs and are paired to perform sequence-to-sequence learning (SMILES to SMILES). In particular, the encoder consists of three 1D convolutional layers followed by on fully connected layer. The decoder has three layers of gated recurrent unit (GRU) networks. The last layer of the RNN decoder defines a probability distribution over all possible characters at each position in the SMILES string.

4.3 Grammar variational autoencoder

One issue with the character-based VAE is that it may map latent points to sequences that are not valid. Thus, Grammar VAE (GVAE) [11] proposes to give the VAE explicit knowledge about how to produce valid sequences by using a grammar for the sequences. The input and output in this cases are also SMILES string. The SMILES grammar will play an important role in this auto-encoder.

The encoder first uses SMILES grammar to parse the SMILES string (correponding to a molecule) into a parse tree. Then the tree is decomposed into a sequence of production rules by performing a pre-order tranversonal on parse tree. Then these rules are converted into 1-hot vectors where each dimension (of the vectors) correspond to a rule in SMILES grammar. These 1-hot vectors combine into a matrix $\mathbf{X}$ that is then passed via a convolutional neural network to a continous latent vector $\mathbf{z}$

Using a recurrent neural network, the decoder first maps $\mathbf{z}$ to a matrix $\mathbf{F}$ of size $T_{max} \times K$ consisting of logit vectors. Here T_{max} is the maximum number of production rules allowed by the decoder (to create the SMILES string) and K is the number of production rules in SMILES grammar. We now will use this logit matrix to produce a production rule matrix $\mathbf{X}$, which yields a SMILES string based on the SMILES grammar. The idea is to maintain stack of unprocessed non-terminal symbols (We first push smiles onto the stack). Then every time a non-terminal symbol is processed (or pop from the stack), a product rule is produced based on the logits $\mathbf{T}$ and added to $\mathbf{X}$ until the number of times (or steps) reaches T_{max}.

On molecular datasets, GVAE produces about twice more valid sequences than CVAE (Character VAE). Bayesian optimization is then performed on the learned continuous latent space to problem of finding new drug-like molecules. In the end, GVAE also yields higher score than CVAE in this problem.

4.4 ORGAN

Objective-Reinforced Generative Adversarial Network (ORGAN) [7] combines reinforcement learing with GAN to generate valid SMILES string representations for molecules. ORGAN models the generator G_θ as a reinforcement-learning agent. At time t, state s_t is the incomplete sequence (string) $Y_{1:t}$, and the generator G_θ must produces an action a_t that gives the next token y_{t+1}. The agent's stochastic policy is then given by $G_\theta(y_t|Y_{1:(t-1)})$, and we wish to maximize:

$$J(\theta) = \mathbb{E}[R(Y_{1:T})|s_0, \theta] \tag{10}$$

The reward R in ORGAN model is defined as:

$$R(Y_{1:T}) = \lambda D_\phi(Y_{1:T}) + (1-\lambda)O_i(Y_{1:T}) \tag{11}$$

where O_i is the desired objectives. In molecule generation problem, O_i can be a combination of solubility, synthetizability, and druglikeness.

4.5 GraphVAE

Unlike previous works that use SMILES representation of molecule, GraphVAE [19] takes input as the graph representation of molecules instead and then use a variational graph auto-encoders to both learn latent space of molecules and generate new molecule data.

5 MolGAN

5.1 Graph representation of molecules

As we mentioned before, one can consider graph representation of a molecule: each node v_i is an atom, with one-hot feature vector $\mathbf{x}_i$ of dimension T, and each edge (v_i, v_j) describes a bond type $y \in \{1, \cdots, Y\}$. This gives us the node attribute matrix $\mathbf{X} \in \mathbb{R}^{n \times T}$ and the adjacency tensor $\mathbf{A} \in \mathbb{R}^{n \times n \times Y}$. This adjacency tensor is similar to the usual adjacency matrix, but each $\mathbf{A}[i, j, :]$ is a one-hot vector corresponds to the bond type $y \in \{1, \cdots, Y\}$.

MolGAN is a deep generative network that is inspired by generative adversarial networks (GAN) and its improved version WGAN, that take a random Gaussian vector as input z and generate a graph structure for some molecule correspoding to z.

5.2 MolGAN architecture

The MolGAN consists of three main components: a generator G_θ, a discriminant D_ϕ, and a reward network $\hat{R}_\psi$. Similar to GAN, the generator try to generate a sample that is close to the one in the actual dataset, and the discriminator try to differentiate between the artificially generated and the actual molecule.

The reward network learns to assign a reward to each molecule to match a score provided by an external software.

The generator consist of a simple multi-layer perceptron (MLP) that takes a random Gaussian vector $z \in \mathbb{R}^D$ and outputs the attribute matrix $\mathbf{X}$ that defines atom types and the adjacency tensor $\mathbf{A}$ that defines atom types. From here, via categorical sampling, two new discrete, sparse objects $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{A}}$ are generated from $\mathbf{X}$ and $\mathbf{A}$ respectively.

The discriminator D_ϕ and the reward network $\hat{R}_\psi$ then both shares a convolutional graph neural networks that outputs $\mathbf{h}_G$. From here, two seperate MLPs D_ϕ and $\hat{R}_\psi$ that take the same $\mathbf{h}_G$ and output a level scalar output for the discriminator and a number between $(0, 1)$ for the reward network. Finally, the loss function being minimized is:

$$L(\theta) = \lambda L_{WGAN} + (1 - \lambda) L_{RL} \tag{12}$$

where L_{WGAN} is the loss function in the WGAN's framework, and L_{RL} is the reinforcement learning loss corresponding to the reward generated by $\hat{R}_\psi$. De Cao and Thomas Kipf [4] also introduces an additional gradient penalty for the GAN network:

$$\alpha(\|\nabla_{\hat{x}} D_\phi(\hat{x})\| - 1)^2 \tag{13}$$

where $\hat{x}$ is a linear combination of the actual data x and the generated data $G_\theta(z)$.

6 Molecular structures via graph

In this section, we investigate the graph data from the molecular structure via graph autoencoder's framework. The graph neural network architecture that we focus on is the MolGAN by De Cao and Kipf [4]. We study other graph neural networks such as variational graph auto-encoders by Kipf and Welling [10], and propose our GNN framework with application to molecular structures in the next section.

6.1 Reimplement MolGAN

One of the first goal is to understand more carefully MolGAN framework and architecture and then reimplement this network and test on the same dataset QM9 [18] [16]. Our discriminator D_ϕ also consists of two graph convolutional layers (GCNs) followed by an aggregation layer that combines output from GCNs with nodes input and then followed by dense layers. Generator G_θ only consists of dense layers and then resize the hidden variable to output the corresponding graph representation for molecules. We work on QM9 data consists of 133,885 compounds up to 9 heavy atoms: carbon (C), oxygen (O), nitrogen (N) and fluorine (F). We split data set into: train (80%), validation (10%) and test (10%). In the training phase, for the GAN implementation, we run for about 28 epochs with batch size 32.

A generated molecule is **valid** if its SMILES string is nonempty and doesn't contain . (non-bond) and *. In the validation phase, the generator produces valid molecules from a single random Gaussian vector of size 32 with the success rate close to 80%. Among valid molecules, scores for chemical properties of generated molecules are all close to the standard scores for real molecules. During test phase, however, the success rate drop to only 30%, but all other chemical indices remain in good shapes. Certain modification of hyperparameters and network architecture together with longer training time may improve the result. However, due to time constraint and computational power together with a few training failures (due to network architecture) prevent us from having a better result.

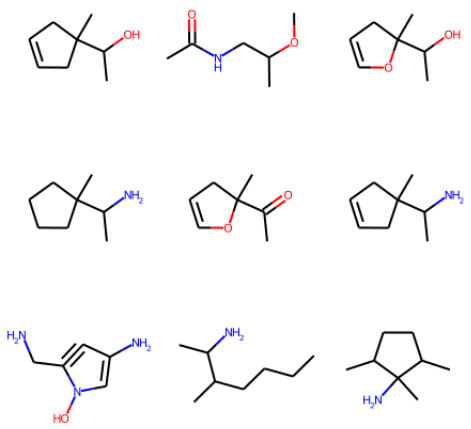

Fig. 2. Nine distinct generated molecules from MolGAN architecture

6.2 Develop a VGAE model for generating molecules

Since MolGAN doesn't have an encoder, we built an additional encoder to enhance molecule generation. In our implementation, the encoder has the same graph convolutional layers as the discriminator, but with different dense layers. After training only the discriminator and generator for 28 epochs, we train the entire auto-encoder, which include an encoder and the generator, which is regarded as the decoder, for an additional 11 epochs. The success rate of generating a valid molecule is 84%, with all chemical indices in good shapes. Again, this rate can be increased as we improve hyperparameters and network architectures and increase training time.

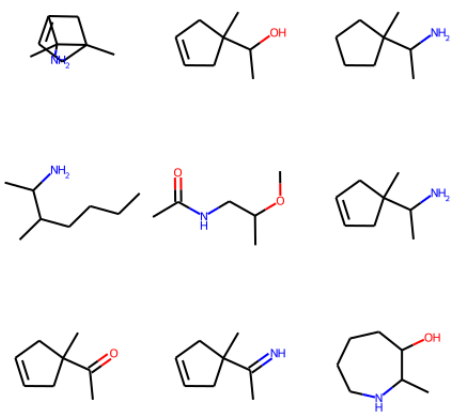

Fig. 3. Nine distinct generated molecules from MolGANVAE architecture

7 Applications: HIV molecules inhibitor prediction

7.1 Datasets

The data used in this study to identify potential HIV inhibitor molecules was sourced from the MoleculeNet data [21], a well-established resource for benchmark datasets in molecular machine learning. For our analysis, we specifically utilized the HIV.csv file from this repository. This file contains experimentally measured data on the ability of various molecules to inhibit HIV replication. The dataset includes the following fields:

- SMILES String: a text-based representation of the molecular structure.
- Activity: Numerical variable representing the experimentally measured ability of the molecule to inhibit HIV replication.
- HIV Active Status: A binary target indicating whether a molecule is active (1) or inactive (0) in inhibiting HIV.

It is important to note that the dataset is significantly imbalanced. Specifically, there are 39,684 samples in the negative class (not HIV active) compared to only 1,443 samples in the positive class (HIV active). This imbalance presents a challenge for machine learning models, as they may become biased towards the majority class and not accurately capture the characteristics of the minority class. To mitigate this issue, we employed several techniques including data augmentation, oversampling of the minority class, and class-weight adjustments, and graph batching (see Figure 4).

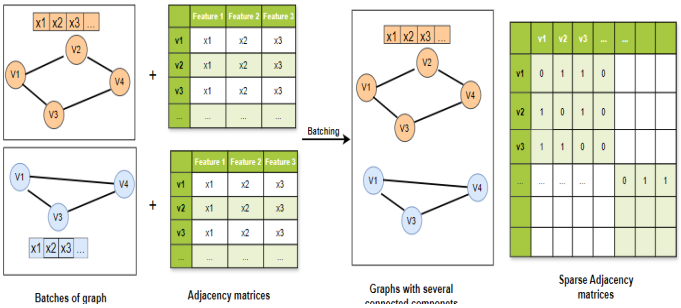

Fig. 4. Overview of Graph batching technique

7.2 Methods

To accurately predict whether a molecule is suitable as an HIV inhibitor, we introduce Dual-Level GCN, a simple yet effective framework consisting of two distinct stages of Graph Convolutional Networks (GCNs). The first stage, a node-level GCN, is designed to detect and highlight the most important components of a molecule by learning rich and informative representations of individual atoms and their immediate environments. This stage is crucial for capturing the local structural features that may be critical for the molecule's inhibitory activity.

The second stage, a graph-level GCN, builds upon the node-level representations to assess the entire molecular graph. This stage focuses on identifying molecules that are highly likely to function as HIV inhibitors. Additionally, it proposes regions of interest (ROI) within the molecular structure that could be key to the molecule's inhibitory potential.

This two-stage framework (see Figure 5) not only addresses the challenge of data imbalance by leveraging detailed information at both the node and graph levels, but it also integrates mini-batching techniques to enhance computational efficiency. By utilizing information from both levels, the framework offers a more robust and comprehensive approach to predicting HIV inhibitors, ultimately improving the model's predictive accuracy.

7.3 Experimental Results

In this section, we present the experimental results obtained from applying our proposed dual-stage Graph Convolutional Network (GCN) framework to the task of identifying potential HIV inhibitor molecules. We evaluate the performance of the framework across multiple metrics, comparing it with baseline models and other state-of-the-art methods. The experiments were conducted using the HIV dataset from the MoleculeNet repository, which was described in the previous sections.

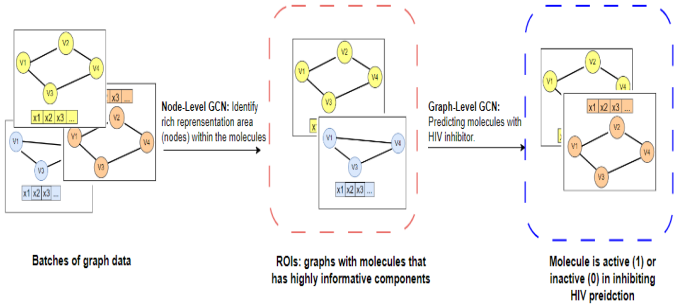

Fig. 5. Dual-Level GCN: with graph-level GCN and node-level GCN

Table 1. Results of Dual-Level GCN on MoleculeNet

Model	F1-Score	AUC ROC
Training	0.823	0.812
Validation	0.820	0.803
Testing	0.791	0.785

8 Conclusion and Future directions

In this paper, we have demonstrated the efficacy of Graph Neural Networks (GNNs) in studying the intricate nature of molecule structures, particularly in predicting the properties of molecular compounds. Furthermore, we proposed a novel GNN-based framework specifically designed with two stages to identify potential HIV inhibitor molecules. This two-stage approach allows the model to capture both fine-grained local details and broad, global features, improving its ability to predict effective HIV inhibitors. Additionally, the framework is designed to handle data imbalance and efficiently process large datasets through mini-batching and adaptive learning techniques. By combining insights from both the node and graph levels, the model provides a robust and accurate prediction mechanism, contributing valuable tools for HIV inhibitor discovery.

While our current framework has shown promising results in identifying potential HIV inhibitor molecules, there is still room for improvement. In the near future, we plan to further refine the model by adjusting hyperparameters and experimenting with different network architectures to enhance its predictive accuracy and robustness.

Additionally, we are exploring the integration of advanced reinforcement learning algorithms into our framework. These algorithms could allow the model to learn more dynamically and make better decisions during the training process, potentially leading to improved identification of key molecular features associated with HIV inhibition.

We are also considering the incorporation of alternative graph neural network (GNN) architectures, which may offer more efficient ways to extract and represent the complex information embedded within molecular graphs. These

alternatives could include attention-based GNNs or those that focus on specific substructures within the molecule, offering new perspectives on how to best capture the essential characteristics for HIV inhibition.

By pursuing these avenues, we aim to further advance our framework's capabilities and contribute to the ongoing efforts in computational drug discovery, particularly in the context of HIV research.

References

1. Bajaj, C., Li, C., Nguyen, M.: Solving the side-chain packing arrangement of proteins from reinforcement learned stochastic decision making. arXiv preprint arXiv:2212.03320 (2022)
2. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, Stroudsburg, PA, USA (2019)
3. Dai, H., Tian, Y., Dai, B., Skiena, S., Song, L.: Syntax-directed variational autoencoder for structured data (2018)
4. De Cao, N., Kipf, T.: MolGAN: An implicit generative model for small molecular graphs (2018)
5. Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules. ACS Cent. Sci. **4**(2), 268–276 (2018)
6. Goodfellow, I.: NIPS 2016 tutorial: Generative adversarial networks (2016)
7. Guimaraes, G.L., Sanchez-Lengeling, B., Outeiral, C., Farias, P.L.C., Aspuru-Guzik, A.: Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models (2017)
8. Jiang, D., Wu, Z., Hsieh, C.Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., Hou, T.: Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. J. Cheminform. **13**(1) (2021)
9. Kingma, D.P., Welling, M.: An introduction to variational autoencoders (2019)
10. Kipf, T.N., Welling, M.: Variational graph Auto-Encoders (2016)
11. Kusner, M.J., Paige, B., Hernández-Lobato, J.M.: Grammar variational autoencoder (2017)
12. Li, M.M., Huang, K., Zitnik, M.: Graph representation learning in biomedicine and healthcare. Nat. Biomed. Eng. **6**(12), 1353–1369 (2022)
13. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015)
14. Nguyen, M., Le, P.: Generalized knowledge-enhanced framework for biomedical entity and relation extraction. arXiv preprint arXiv:2408.06618 (2024)
15. Peters, M.E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., Smith, N.A.: Knowledge enhanced contextual word representations. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, Stroudsburg, PA, USA (2019)

16. Ramakrishnan, R., Dral, P.O., Rupp, M., von Lilienfeld, O.A.: Quantum chemistry structures and properties of 134 kilo molecules. Sci. Data **1**(1), 140022 (2014)
17. RDKit: Open-source cheminformatics. http://www.rdkit.org, [Online; accessed 11-April-2013]
18. Ruddigkeit, L., van Deursen, R., Blum, L.C., Reymond, J.L.: Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. J. Chem. Inf. Model. **52**(11), 2864–2875 (2012)
19. Simonovsky, M., Komodakis, N.: GraphVAE: Towards generation of small graphs using variational autoencoders. In: Artificial Neural Networks and Machine Learning – ICANN 2018, pp. 412–422. Springer International Publishing, Cham (2018)
20. Sutton, R.S., Barto, A.G.: An Reinforcement Learning: Introduction. Mit Press (2012)
21. Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V.: Moleculenet: A benchmark for molecular machine learning. arXiv preprint 1703.00564 (2018)
22. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2021)