GraphGANFed: A Federated Generative Framework for Graph-Structured Molecules Towards Efficient Drug Discovery

Daniel Manu, Jingjing Yao, *Member, IEEE*, Wuji Liu, and Xiang Sun, *Member, IEEE*,

***Abstract*—Recent advances in deep learning have accelerated its use in various applications, such as cellular image analysis and molecular discovery. In molecular discovery, a generative adversarial network (GAN), which comprises a discriminator to distinguish generated molecules from existing molecules and a generator to generate new molecules, is one of the premier technologies due to its ability to learn from a large molecular data set efficiently and generate novel molecules that preserve similar properties. However, different pharmaceutical companies may be unwilling or unable to share their local data sets due to the geo-distributed and sensitive nature of molecular data sets, making it impossible to train GANs in a centralized manner. In this paper, we propose a Graph convolutional network in Generative Adversarial Networks via Federated learning (GraphGANFed) framework, which integrates graph convolutional neural Network (GCN), GAN, and federated learning (FL) as a whole system to generate novel molecules without sharing local data sets. In GraphGANFed, the discriminator is implemented as a GCN to better capture features from molecules represented as molecular graphs, and FL is used to train both the discriminator and generator in a distributive manner to preserve data privacy. Extensive simulations are conducted based on the three benchmark data sets to demonstrate the feasibility and effectiveness of GraphGANFed. The molecules generated by GraphGANFed can achieve high novelty (≈ 100) and diversity (> 0.9). The simulation results also indicate that 1) a lower complexity discriminator model can better avoid mode collapse for a smaller data set, 2) there is a tradeoff among different evaluation metrics, and 3) having the right dropout ratio of the generator and discriminator can avoid mode collapse.**

***Index Terms*—Generative adversarial networks, Graph convolutional networks, Federated learning, Drug discovery**

I. Introduction

The discovery of new organic and inorganic molecules remains a challenge in medicine, chemistry, and materials sciences. Traditional approaches to molecular discovery involve mathematical frameworks derived from related properties calculated from chemical structures with different physical or biological reactions [1, 2]. However, these mathematical frameworks may not fully capture the properties of the chemical structures, limiting the ability to fully explore novel molecules. To address this challenge, computer-aided solutions have been developed that utilize deep learning methods to learn highly complex representations of chemical structures, extract features from existing molecules, and generate new molecules based on the learned features. These approaches have shown promise in facilitating molecular generation [3–6] and *de novo* drug design [7].

Deep generative models, one of the deep learning models widely applied in molecular discovery, generate molecules based on features/patterns learned from the existing molecule sets with desired chemical properties. There are two common types of input data for generative models: Simplified Molecular Input Line Entry System (SMILES) representations [8] and molecular graphs [9]. SMILES is a string-based representation, has been applied as the input data type for Natural Language Processing (NLP) deep learning models such as Recurrent Neural Networks (RNNs) [10, 11] and Auto-Encoders [12–17]. Recent works have explored the use of RNN-based generative models to learn the properties of the existing molecules represented as SMILES, and then generate novel molecules with the same properties. Auto-Encoder based models comprise an encoder and a decoder, where the encoder encodes the molecules into a latent vector representation and the decoder transforms the latent vectors back to molecules. In contrast to SMILES, molecular graphs use edges and nodes to represent the structures of molecules. Molecular graphs can represent more detailed structures of molecules than SMILES, thus attracting much attention [12]. Fig. 1 shows different molecular representations for generative modeling.

Graph Convolutional Neural Network (GCN) is a typical deep learning model that can learn molecular graph representations and analyze the related properties [18–20]. For example, [21] applies GCN to estimate a specific metric (e.g., novelty, validity, or uniqueness) of a molecule by analyzing the structure of the molecule represented as a molecular graph. Generative Adversarial Networks (GANs) have emerged as another popular approach for molecular discovery [1, 3, 4]. A GAN model consists of a generator and a discriminator that compete with each other. The generator generates realistic new samples that preserve the features of the existing samples to deceive the discriminator, while the discriminator tries to differentiate between the generated and existing samples. The generator and discriminator are trained iteratively, and the generator becomes better at generating realistic samples that deceive the discriminator over time. GANs have been widely used in various image generation tasks and have demonstrated

D. Manu and X. Sun are with the SECNet Lab., Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA. E-mail: {dmanu,sunxiang}@unm.edu.

J. Yao is with the Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA. E-mail: jingjing.yao@ttu.edu.

W. Liu is with Amazon, San Diego, CA, 92121, USA. E-mail: liuwujicoding@gmail.com.

This work was supported by the National Science Foundation under Award CNS-2148178.

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

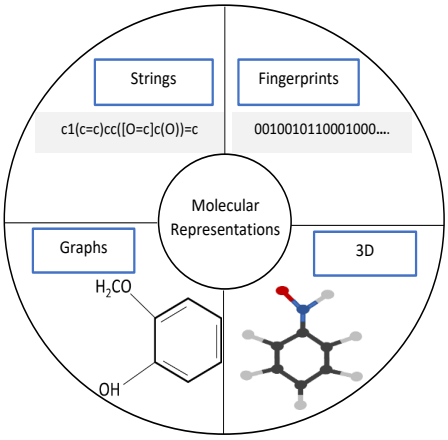


Fig. 1: Different molecular representations.

outstanding performance [22–24]. The performance of GANs for molecular generation can be further improved by integrating GCNs since GCNs can learn more detailed properties of molecules represented as graphs. Hence, the integration of GCN and GAN in drug discovery is worth investigating.

To derive an accurate deep learning model for generating new molecules, it is essential to train the model on a large and diverse data set. However, it is not realistic to obtain this data set because the sharing of different pharmaceutical companies' molecular databases is not always feasible due to the compound intellectual property and strict data regulation [25]. One of the existing solutions is that pharmaceutical companies encrypt their data samples and share the encrypted data samples with a third party, which trains the model over the encrypted data samples by using Secure Multiparty Computation (MPC) [26]. The encryption method, however, cannot fully ensure data privacy, thus hindering collaborative drug discovery. Recently, Federated Learning (FL) has emerged as a promising solution to train deep learning models in a distributed and privacy-preserving manner [27]. In FL, a centralized FL server broadcasts a global model to all the clients in terms of pharmaceutical companies. Each client trains its received global model over its local data set for several epochs and then uploads its trained local model to the FL server. After that, the FL server aggregates the received local models from the clients to generate a new global model, and then broadcasts the new global model to the clients to start the next global round. This process continues until the global model is converged. The clients only share their local models and do not share their local data sets, which ensures data privacy. Hence, FL can enable pharmaceutical companies to collaborate and train accurate deep learning models on their collective data while preserving data privacy, which is important to accelerate the drug discovery process.

In this work, we propose the Graph convolutional network in Generative Adversarial Networks via Federated learning (GraphGANFed) framework for molecular discovery, which leverages FL to train a GAN model that uses GCNs to represent molecules as graphs. In this framework, the discriminator is implemented by a GCN that learns the features/properties of the existing molecules, while the generator is implemented by a multilayer perceptron (MLP) or a multilayer neural network that generates new molecules while preserving the properties of the existing molecules to deceive the discriminator. The discriminator and generator are trained based on FL, which ensures data privacy as each client keeps its data locally. In GraphGANFed, the discriminator evaluates the molecules generated by the generator. This evaluation guides the generator to generate new molecules that preserve features of existing molecules but have different structures, so that the generator can deceive the discriminator later on. Hence, it is critical to design a good discriminator model that can perform an accurate evaluation to guide the generator in generating more realistic new molecules. Therefore, we investigate how different discriminator model structures affect the performance metrics (e.g., Validity, Novelty, Diversity, etc.) in our framework GraphGANFed. We also analyze how different dropout ratios affect the evaluation metrics. The main contributions of our work are summarized as follows.

- We propose a GraphGANFed framework that combines GCN, GAN, and FL to generate novel and effective molecules while preserving data privacy.
- We implement GraphGANFed and demonstrate its performance based on three benchmark datasets. The results indicate that GraphGANFed generates molecules with high novelty (≈ 100) and diversity (> 0.9).
- We investigate the impact of various factors on the performance of GraphGANFed, including the complexity of the discriminator, number of clients, and dropout ratios via extensive simulations.

The rest of the paper is organized as follows. Section II presents a brief background and related works regarding molecular discovery, GAN, and GCN. Section III introduces the GraphGANFed architecture design and the metrics to evaluate generated molecules. Section IV discusses the simulation setups and results. Finally, Section V concludes the paper.

II. Background and Related Works

In this section, we provide a brief overview of GANs, state-of-the-art generative models for molecular discovery, challenges and proposed solutions in GANs, and GCN frameworks for molecular discovery.

A. Basis of GANs

GANs are generative models that aim to minimize the divergence between the real data distribution P_d and the generative model distribution P_z. Given a training data distribution P_d, the generator generates samples x from the distribution P_z with the random noise z. The discriminator discriminates between real samples from P_d and generated samples from P_z [28]. The generator aims to minimize the loss $\log(1 - D(\tilde{x}))$, where $\tilde{x}$ is the generator's output when given noise z and $D(\tilde{x})$ is is the discriminator's estimate of the probability that a generated sample is an existing sample. On the other hand, the discriminator tries to maximize the probability of correctly

labeling both real and generated samples. Overall, GANs can be viewed as a two-player min-max game:

$$\min_{\theta}\max_{\phi}\mathbb{E}_{x\sim P_d}[logD_\phi(x)]+\mathbb{E}_{\tilde{x}\sim P_z}[log(1-D_\phi(G_\theta(z))], \quad (1)$$

where $D_\phi(\cdot)$ and $G_\theta(\cdot)$ are the discriminator and generator models parameterized by ϕ and θ, respectively. $\mathbb{E}_{x\sim P_d}[logD_\phi(x)]$ is the expected value over all the existing samples for all the probabilities that the discriminator correctly classifies the existing samples and $\mathbb{E}_{\tilde{x}\sim P_z}[log(1-D_\phi(G_\theta(z))]$ is the expected value over all the generated samples for all the probabilities that the discriminator correctly classifies the generated samples from the generator.

B. Generative models for molecular discovery

GANs have emerged as a highly promising approach for generating novel and diverse molecules, and have become the most commonly used generative models for state-of-the-art molecular generations. Several recent studies have demonstrated the power of GANs in the field. Guimaraes *et al.* [4] proposed an objective-reinforced generative adversarial network (ORGAN) by integrating reinforcement learning (RL) into the GAN. They used the SeqGAN (sequence based Generative Adversarial Network) architecture based on RNNs and GAN, and applied a policy gradient method to learn the generative network parameters. Putin *et al.* [15] proposed a deep neural network Adversarial Threshold Neural Computer (ATNC) architecture. This architecture is based on GAN and RL, where the generator is implemented as a Differential Neural Computer (DNC) with new specific blocks known as adversarial threshold (AT). Their framework successfully generated good molecules whilst overcoming the negative reward problem in ORGANIC.

C. Challenges and solutions in GANs

GANs are widely known for generating expected samples but they suffer from the mode collapse issue (i.e., generate limited variety of samples) and may be difficult to train. Hence, several studies have explored methods to stabilize GAN training. Arjovsky *et al.* [29] proposed a Wasserstein distance $W(P_z, P_d)$ to measure the distance between the existing data distribution P_d and generated data distribution P_z. Under soft conditions (i.e., generator is locally Lipschitz and continuous in θ), $W(P_z, P_d)$ is continuous and differentiable almost everywhere. By minimizing the Wasserstein distance, a GAN can generate more realistic samples and better approximate the existing data distribution.

The min-max game for Wasserstein GAN (WGAN) can be expressed by the Kantorovich-Rubinstein duality [30], i.e.,

$$\min_{\theta}\max_{\phi\in\mathcal{D}}\mathbb{E}_{x\sim P_d}[D_\phi(x)]-\mathbb{E}_{\tilde{x}\sim P_z}[D_\phi(\tilde{x})], \quad (2)$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions, P_z is the model distribution implicitly defined by $\tilde{x}=G(z)$, and $z\sim p_z$ is a random noise vector generated from a prior distribution $p(z)$. Hence, for an optimal discriminator (which is also called a critic since it is not to classify samples, but to evaluate the Wassertein distance between generated samples and existing samples), reducing the value function with respect to the generator parameters, i.e., from Eq. (2), reduces $W(P_d, P_z)$.

To ensure that the discriminator satisfies the Lipschitz constraint, they proposed weight clipping of the critic within a dense space $[-c, c]$. However, they demonstrated that the weight clipping approach leads to optimization problems in [31]. To address the stability, vanishing, and exploding gradient issues, Gulrajani *et al.* [31] proposed an alternative approach to enforce the Lipschitz constraint. They introduced a soft constraint with a penalty on the gradient norm for random samples $\hat{x}\sim P_{\hat{x}}$, where $P_{\hat{x}}$ is the distribution of random samples from the generated and existing data distributions. Thus, the new loss function of the discriminator becomes

$$L=\underbrace{-\mathbb{E}_{x\sim P_d}[D_\phi(x)]+\mathbb{E}_{\tilde{x}\sim P_z}[D_\phi(\tilde{x})]}_{\text{original critic loss}}+\underbrace{\gamma\mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}}D(\hat{x})\|_2-1)^2]}_{\text{gradient penalty}}, \quad (3)$$

where γ is the gradient penalty coefficient. In our work, we adopt the loss function in Eq. (3) to stabilize the training process and mitigate model collapse.

D. Graph Convolutional Networks

GCNs have been applied in various drug discovery applications such as biological property [32], quantum mechanical property [33], interaction prediction [34, 35], synthesis prediction [36], and *de novo* molecular design [12]. In [32], a graph convolutional framework was proposed to learn the molecular representations using both node and edge features. Each layer is comprised of atoms (nodes) in a molecule and pair-wise representations. The framework uses a Weave unit to establish relations between different layers through propagation between atoms to atoms, atoms to pairs, pairs to atoms, and pairs to pairs. Every layer adopts the weave unit architecture, except for the last convolution layer where only the atom representations are utilized for downstream applications, such as estimating solubility or drug efficacy. The framework uses neural networks to model the transitions over the same representations (i.e., atom-atom and pair-pair). However, for the transitions over different representations (i.e., atom-pair and pair-atom), an additional order-invariant aggregation operation is used for the feature transition. The graph convolutional framework inputs are molecular graphs containing atoms and bond. The authors evaluated their proposed framework for biological activities on 259 data sets consisting of PubChem BioAssay (PCBA) [37], the training set for the Tox21 challenge [38], the enhanced directory of useful decoys [39], and the "maximum unbiased validation" datasets built by Rohrer and Baumann [40], in a multi-task setting where biological activities are predicted. However, the results show that the proposed framework does not always outperform the baseline methods that use Morgan fingerprints.

In [12], the authors proposed a framework named GraphVAE for molecular generations, which directly generates a probabilistic fully-connected graph with a predefined maximum size. GraphVAE is based on the variational autoencoder method, where the stochastic graph encoder is used to process discrete attributed graphs $G=(A, E, F)$, representing the adjacency matrix A, edge attribute tensor E, and node

attribute tensor F with a predefined number of nodes. The encoder is implemented by a feed-forward network with edge-conditioned graph convolutions. The stochastic graph encoder embeds the graphs into a continuous representation z. The graph decoder outputs a probabilistic fully-connected graph on the predefined nodes, where the discrete samples can be drawn. To ensure that the output graph matches the ground truth, they used a standard matching algorithm to align the output graph to the ground truth. GraphVAE was evaluated based on two molecular data sets, QM9 and ZINC, and the results demonstrated that it can learn appropriate embedding properties for small molecules, but it is difficult to capture complex chemical interactions for larger molecules.

All the above existing works about molecular generation and drug discovery are performed in a centralized setting which results in mode collapse and hinders the ability of the models to generalize on diverse datasets. Our proposed framework aims to solve this model collapse and diversity issues in the centralized setting by ensuring that models are trained in a federated setting without sharing the client dataset thus preserving data privacy.

III. Graph Convolutional Network in Generative Adversarial Networks via Federated learning (GraphGANFed)

In this section, we explain the key concepts in molecular graphs, provide the detailed design of our proposed GraphGANFed architecture, and analyze the metrics to evaluate the performance of our proposed GraphGANFed architecture.

A. Molecular graphs

In GraphGANFed, every molecule is represented as a non-directional graph $\mathcal{G}$ with a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each atom in the molecule corresponds to a node in the graph, and the type of each atom is specified using a node label matrix $\boldsymbol{V}$. The matrix $\boldsymbol{V} = \{v_{ij} \,|1 \leq i \leq N,\ 1 \leq j \leq B\}$, where N is the total number of atoms in a molecule, B is the number of possible atom types, and v_{ij} implies if atom i is the type j (i.e., v_{ij}=1) or not (i.e., v_{ij}=0). Here, we consider generating molecular graphs with the fixed maximum number of atoms $N = 10$. Each atom belongs to one of the 10 possible types (i.e., $B = 10$): Carbon (C), Nitrogen (N), Oxygen (O), Fluorine (F), Bromine (Br), Phosphorus (P), Sulfur (S), Chlorine (Cl), Iodine (I), and one padding symbol (*). The padding symbol is used to represent an atom other than the types mentioned above, and it allows for flexibility in the molecular structure to accommodate the individual design preferences of different clients.

In GraphGANFed, the bonds between atoms in a molecule are represented as non-directional edges $(i, i') \in \mathcal{E}$, where $i, i' \in \mathcal{V}$. An adjacency matrix $\boldsymbol{A}$ is used to depict all the bonds between the atoms in a molecule. The matrix $\boldsymbol{A} = \{a_{ii'k} \,|1 \leq i, i' \leq N, 1 \leq k \leq T\}$, where T is the number of bond types, and $a_{ii'k}$ implies if the bond between atoms i and i' is bond type k (i.e., $a_{ii'k} = 1$) or not (i.e., $a_{ii'k} = 0$). Here, we consider 5 types of bonds (i.e., $T = 5$): zero, single, double, triple, and aromatic bonds.

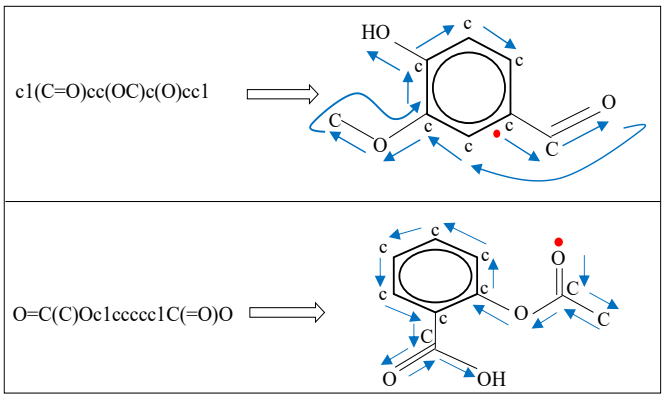


Fig. 2: Examples of converting SMILES to molecular graphs.

B. The GraphGANFed framework

In general, GraphGANFed consists of three main parts, i.e., the generator, discriminator, and FL system. Fig. 3 shows the overview of GraphGANFed.

The Generator is implemented using a multi-layer perceptron (MLP) neural network. It takes a random noise vector z, which is sampled from a prior distribution $p(z)$, as an input and outputs the molecular graph represented by two types of continuous matrices, i.e., a node label matrix $\boldsymbol{V}$ and an adjacency matrix $\boldsymbol{A}$. Each hidden layer in the MLP neural network applies the tanh activation function. The output of the final hidden layer of the MLP is linearly projected to match the dimensions of $\boldsymbol{V}$ and $\boldsymbol{A}$, and then normalized using the softmax function. Fig. 10 shows some molecules generated by the described MLP neural network. The goal of the generator is to minimize the following loss function [6]

$$\mathcal{L}^{gen} = -\log D(\tilde{\boldsymbol{V}}^{gen}, \tilde{\boldsymbol{A}}^{gen}), \tag{4}$$

where $\tilde{\boldsymbol{V}}^{gen}$ and $\tilde{\boldsymbol{A}}^{gen}$ are the node label and adjacency matrices of a generated molecule, respectively, and $D(\tilde{\boldsymbol{V}}^{gen}, \tilde{\boldsymbol{A}}^{gen})$ is the output of the discriminator for the generated molecule.

It is worth noting that both the generated and existing molecules will be used to train the discriminator, which will be explained later on. However, the node label and adjacency matrices of a generated molecule (i.e., $\tilde{\boldsymbol{V}}^{gen}$ and $\tilde{\boldsymbol{A}}^{gen}$) are continuous, while the node label and adjacency matrices of an existing molecule (denoted as $\boldsymbol{V}^{exit}$ and $\boldsymbol{A}^{exit}$) are discrete, as they are derived based on the definition of molecular graphs described in Section III-A. To make the types of matrices consistent (so that they can be fed into the same discriminator model for training), we discretize $\tilde{\boldsymbol{V}}^{gen}$ and $\tilde{\boldsymbol{A}}^{gen}$ by applying categorical sampling [41]. Specifically, we obtain the discrete node label and adjacency matrices of a generated molecule by applying categorical sampling as follows: $\boldsymbol{V}^{gen} = \text{cat}(\tilde{\boldsymbol{V}}^{gen})$ and $\boldsymbol{A}^{gen} = \text{cat}(\tilde{\boldsymbol{A}}^{gen})$, where $\boldsymbol{V}^{gen}$ and $\boldsymbol{A}^{gen}$.

The Discriminator aims to distinguish the molecules generated by the generator from the those in the existing data set, and so the loss function of the discriminator is [31]

$$\begin{aligned}\mathcal{L}^{dis} = &-\log D(\boldsymbol{V}^{gen}, \boldsymbol{A}^{gen}) + \log D(\boldsymbol{V}^{exist}, \boldsymbol{A}^{exist}) \\ &+ \gamma(||\nabla_{\boldsymbol{\Omega_A}, \boldsymbol{\Omega_V}} D(\boldsymbol{\Omega_A}, \boldsymbol{\Omega_V})|| - 1)^2,\end{aligned} \tag{5}$$

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

where $\boldsymbol{V}^{exist}$ and $\boldsymbol{A}^{exist}$ are the node label and adjacency matrices of a molecule in the existing data set, respectively, $D(\boldsymbol{V}^{exist}, \boldsymbol{A}^{exist})$ is the output of the discriminator for the existing molecule, and $\gamma(||\nabla_{\boldsymbol{\Omega_A},\boldsymbol{\Omega_V}} D(\boldsymbol{\Omega_A}, \boldsymbol{\Omega_V})|| - 1)^2$ is the gradient penalty term to stabilize the gradients of the discriminator. Here, γ is the gradient penalty coefficient, $\boldsymbol{\Omega_A} = \varepsilon \boldsymbol{A}^{exist} + (1-\varepsilon)\boldsymbol{A}^{gen}$, $\boldsymbol{\Omega_V} = \varepsilon \boldsymbol{V}^{exist} + (1-\varepsilon)\boldsymbol{V}^{gen}$, and $\varepsilon \in [0, 1]$ is a predefined hyperparameter. The intuition of the loss function $\mathcal{L}^{dis}$ is to maximize the difference between the output of an existing molecule and the output of a generated molecule i.e., $D(\boldsymbol{V}^{exist}, \boldsymbol{A}^{exist}) - D(\boldsymbol{V}^{gen}, \boldsymbol{A}^{gen})$. A larger difference implies the two molecules can be easily discriminated. Note that the two outputs $D(\boldsymbol{V}^{exist}, \boldsymbol{A}^{exist})$ and $D(\boldsymbol{V}^{gen}, \boldsymbol{A}^{gen})$ are positive scalar values capturing the features of the molecules.

The discriminator is implemented using a GCN based on the architecture in [6], which convolves the node label matrix $\boldsymbol{V}$ using the adjacency matrix $\boldsymbol{A}$. Here, we use the relational GCN (R-GCN) encoder, which is a convolutional network that supports non-directional and multiple-edge graphs as inputs. Note that the reason why GCN is applied as the architecture of the discriminator is that GAN has been demonstrated to effectively capture the representations from the molecular graphs [6], and other models, such as MLP, may not effectively evaluate the relationship among atoms and their connections. For example, the authors in [42] have quantified the performance of using GCN and MLP for layer feature propagation on graph-structured data (e.g., citation networks) for the node classification task. From their results, the MLP achieved the lowest classification accuracy compared to different variants of GCN propagation models. The discriminator consists of two convolution layers to extract the atom features, followed by one layer MLP to compute a one-dimensional feature representation, which will be parallelly fed into two pieces of one hidden layer, as illustrated in Fig. 3. Note that the design of two parallel hidden layers with different activation functions is followed by the discriminator model in the MolGAN framework [6]. Other GAN architectures such as [43, 44] used multiple parallel convolutional layers with different activation functions to extract different multiscale features from the input data. Based on our understanding, using more parallel hidden layers with different activation functions can improve the performance of the GAN model. This is because different parallel hidden layers with different activation functions achieve different nonlinear characteristics, which enables the discriminator to extract diverse and non-overlapping features of molecules. For example, one MLP might focus on identifying positive features that indicate a molecule's desirability, while the other might highlight negative features that indicate undesired properties. This enables a more accurate evaluation of the input molecules. Having more MLPs tends to better capture different features of training molecular graphs but increases the risk of overfitting, i.e., the model is over-complicated to memorize the noise in the training molecule sets instead of the underlying features. Hence, how to identify the optimal number of parallel hidden layers is still unclear and will be our future work. The final output layer is used to aggregate the features from the two hidden layers followed by the tanh activation function to generate a scalar value representing the features of the input molecule. Specifically, the atom feature representations are propagated through each of the first two layers based on [3, 6]

$$\begin{aligned} h_i^{'(l+1)} &= f_s^{(l)}(h_i^{(l)}, \boldsymbol{v}_i) + \sum_{i'=1}^{N} \sum_{k=1}^{T} \frac{a_{ii'k}}{|\mathcal{N}_i|} f_t^{(l)}(h_{i'}^{(l)}, \boldsymbol{v}_i), \\ h_i^{(l+1)} &= \tanh(h_i^{'(l+1)}), \end{aligned} \tag{6}$$

where $h_i^{(l)}$ is the feature of atom i at layer l, $\boldsymbol{v_i}$ is the type of atom i (i.e., the i^{th} row in the node label matrix $\boldsymbol{V}$), $f_s^{(l)}$ is the residual or skip connection between layer l and the next layer $l+1$, $f_t^{(l)}$ is an edge-type affine function of layer l, $\mathcal{N}_i$ is the set of atom i's neighboring atoms in the molecule[1], $|\mathcal{N}_i|$ implies the number of neighbors for atom i. After propagating the node feature representations through 2 convolution layers, the output is converted into a 1-dimensional graph representation, denoted as $h_i^{(L)}$, which is simultaneously fed into two parallel hidden layers. Denote $\sigma(g_1(h_i^{(L)}, \boldsymbol{v}_i))$ and $tanh(g_2(h_i^{(L)}, \boldsymbol{v}_i))$ as the outputs of the two parallel hidden layers after applying the sigmoid and tanh activation functions, respectively, where $g_1(h_i^{(L)}, \boldsymbol{v}_i)$ and $g_2(h_i^{(L)}, \boldsymbol{v}_i)$ are the outputs of the two parallel hidden layers before their activation functions. The final output layer will aggregate the results from the two parallel hidden layers by conducting element-wise multiplication followed by the tanh activation function, i.e.,

$$\begin{aligned} h_{\mathcal{G}}^{'} &= \sum_{i=1}^{N} \sigma(g_1(h_i^{(L)}, \boldsymbol{v}_i)) \odot \tanh(g_2(h_i^{(L)}, \boldsymbol{v}_i)), \\ h_{\mathcal{G}} &= \tanh(h_{\mathcal{G}}^{'}), \end{aligned} \tag{7}$$

where $h_{\mathcal{G}}$ and $h'_{\mathcal{G}}$ are the results before and after the tanh activation function, and $\odot$ is element-wise multiplication. Here, $h'_{\mathcal{G}}$ is a one-dimensional vector feature representation of molecular graph $\mathcal{G}$, and $h_{\mathcal{G}} \in [0, +\infty]$ is a scalar feature representation. Note that the notations $h_{\mathcal{G}}$ and $D(\boldsymbol{V}^{gen}, \boldsymbol{A}^{gen})$/$D(\boldsymbol{V}^{exist}, \boldsymbol{A}^{exist})$ have the same physical meaning, i.e., the output of the discriminator given a molecular graph $\mathcal{G}$ represented by $\langle \boldsymbol{V}^{gen}, \boldsymbol{A}^{gen} \rangle$ or $\langle \boldsymbol{V}^{exist}, \boldsymbol{A}^{exist} \rangle$.

The FL system enables distributed training of the generator and discriminator based on the local datasets in different clients in terms of pharmaceutical companies, ensuring that these clients do not need to share their data sets with others and thus preserving data privacy. Fig. 3 illustrates how the FL system distributively trains the generator and discriminator. In general, the system consists of a centralized FL server and a number of clients. Similar to the traditional FL system, there are four steps in each global round. Owing to the fact that there are two models, i.e., a generator and discriminator in GraphGANFed, the local model calculation step is different from the traditional FL. Specifically,

1) **Global model broadcasting:** at the beginning of each global round, the FL server broadcasts the current global generator θ^{global} and discriminator ϕ^{global} to the clients.

[1]Here, we define two atoms in a molecule are the neighbors if there is an edge (i.e., single, double, triple, or aromatic bond) between the two atoms.

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

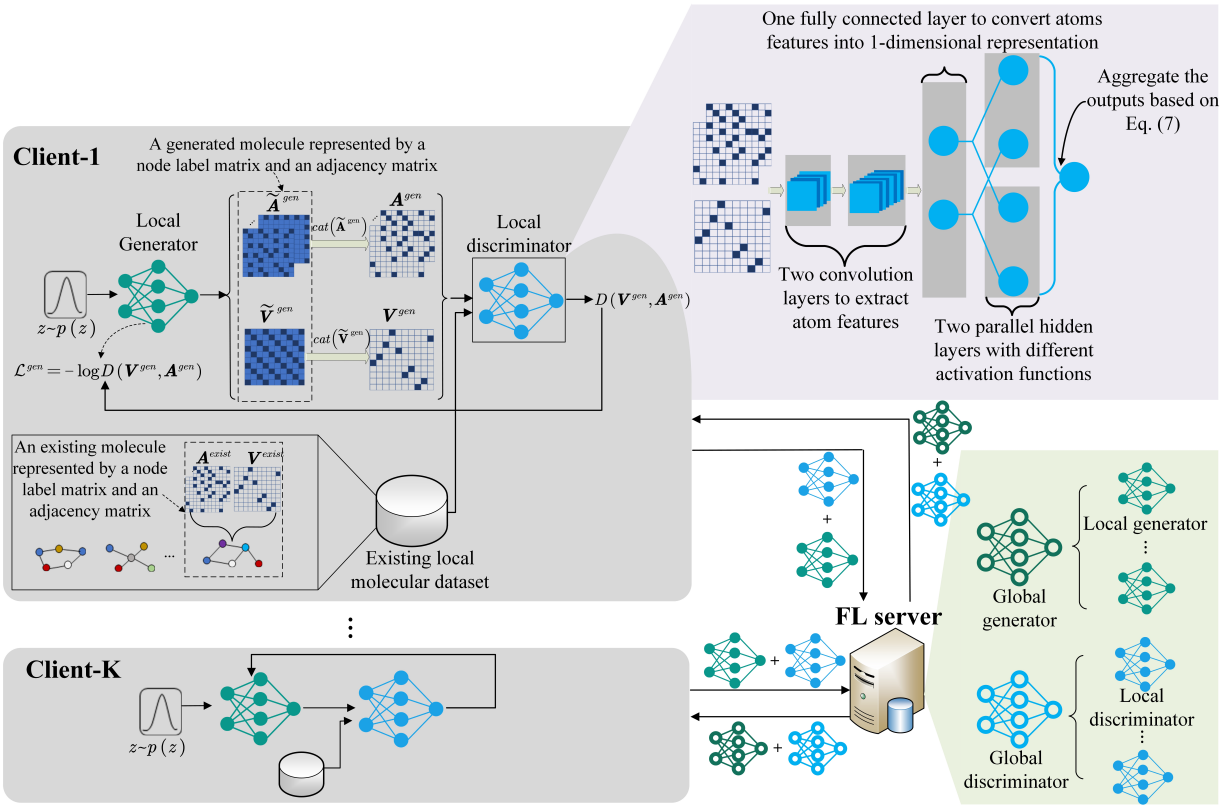


Fig. 3: Overview of the GraphGANFed framework.

2) **Local model calculation:** upon receiving the global generator and discriminator, as shown in Fig. 4, each client i) divides its local dataset of existing molecules into several batches with the same size, where M is the number of batches, and ii) trains the local discriminator and generator over E epochs. In each epoch, a client
   a) generates a batch of new molecules based on its local generator;
   b) trains the local discriminator ϕ_k^{local} based on the batch of generated molecules and the 1^{st} batch of the existing molecules in M via batched gradients decent. Note that the node label/adjacency matrices of both generated and existing molecules, $\boldsymbol{V}^{gen}/\boldsymbol{A}^{gen}$ and $\boldsymbol{V}^{exist}/\boldsymbol{A}^{exist}$ are discrete and non-differentiable, meaning that back-propagation cannot be performed. Hence, we apply a gradient estimator [41], which converts the discrete non-differentiable samples in $\boldsymbol{V}^{gen}/\boldsymbol{A}^{gen}$ and $\boldsymbol{V}^{exist}/\boldsymbol{A}^{exist}$ into continuous differentiable samples based on the Gumbel-Softmax distribution to enable back-propagation to be performed;
   c) trains the local generator θ_k^{local} based on the outputs of the discriminator with respect to the batch of generated molecules in Step a);
   d) repeats Steps a)-c) to train the local generator θ_k^{local} and discriminator ϕ_k^{local} until the last batch of the existing molecules in M has been used for training;
3) **Local model uploading:** once a client trained its local generator θ_k^{local} and discriminator ϕ_k^{local} after $E-1$ epochs, it uploads θ_k^{local} and ϕ_k^{local} to the FL server.
4) **Local model aggregation:** once the FL server receives θ_k^{local} and ϕ_k^{local} from all the clients, it performs model aggregation based on, for example, FedAvg [45] to derive the new global generator and discriminator, i.e.,

$$\begin{cases} \theta^{global} = \sum_k \frac{|\boldsymbol{S}_k^{gen}|}{|\boldsymbol{S}^{gen}|}\theta_k^{local}, \\ \phi^{global} = \sum_k \frac{|\boldsymbol{S}_k^{dis}|}{|\boldsymbol{S}^{dis}|}\phi_k^{local}, \end{cases} \tag{8}$$

where $|\boldsymbol{S}_k^{gen}|$ and $|\boldsymbol{S}_k^{dis}|$ are the numbers of molecules are used to train local generator and discriminator, respectively, for client k, $|\boldsymbol{S}^{gen}| = \sum_k |\boldsymbol{S}_k^{gen}|$, and $|\boldsymbol{S}^{dis}| = \sum_k |\boldsymbol{S}_k^{dis}|$.

The global round continues until the global generator and discriminator converge.

C. Metrics

We apply the same metrics as the MOSES [46] benchmark to evaluate the performance of the molecules generated by GraphGANFed. These metrics can be used to identify the performance and some common issues (such as mode collapse and overfitting) of the generative model in GraphGANFed. Specifically, below are the details of all the metrics used to evaluate the generated molecules.

- **Validity** refers to the fraction of generated molecules that are valid. RDkit's molecular structure parser [47] is used to compute the validity, which measures whether the generated molecules maintain explicit chemical constraints,

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

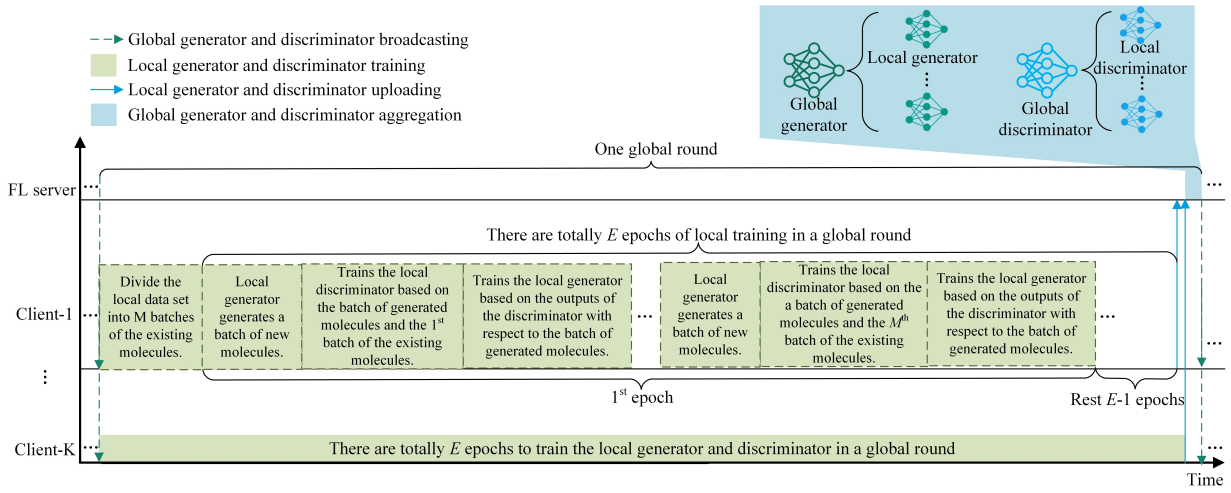


Fig. 4: Procedure of a global round in FL.

such as proper atom valency and bond consistency. The **Validity** value is a percentage in the range $[0, 100]$.

- **Uniqueness** refers to the fraction of valid molecules that are unique. Low uniqueness means repetitive molecular generation, which may imply mode collapse, i.e., it produces a limited variety of molecules. The value of **Uniqueness** can be a percentage in the range $[0, 100]$.
- **Novelty** refers to the fraction of valid molecules that are different from the ones in the existing dataset. Low novelty indicates that the generative model tens to overfit to the existing data. The value of **Novelty** can be a percentage in the range $[0, 100]$.
- **Internal Diversity (IntDiv$_p$)** estimates the chemical diversity in the generated molecules and is computed as the (p^{th}) power of the mean of the Tanimoto similarity (T) between the fingerprints of all the pairs of molecules $(m1, m2)$ in the generated molecules set (G), i.e.,

$$\mathrm{IntDiv_p(G)} = 1 - \sqrt[p]{\frac{1}{|G|^2} \sum_{m_1, m_2 \in G} T(m_1, m_2)^p}. \quad (9)$$

  IntDiv$_p$ can be used to identify normal failure cases in the generative model, such as mode collapse. During mode collapse, the generative model keeps generating similar structures or a small variety of molecules, excluding some parts of the chemical space. A higher **IntDiv**$_p$ means a higher diversity in the generated molecule set. The value of **IntDiv**$_p$ can be $[0, 1]$.
- **Quantitative Estimation of Drug-likeliness (QED)** refers to how likely a molecule is a viable candidate for a drug. The value of **QED** can be $[0, 1]$.
- **Octanol-water partition coefficient (LogP)** refers to a chemical's concentration in the octanol phase to its concentration in the water phase in a two-phase octanol/water system. **LogP** was computed based on RDKit's Crippen estimation. The value of **LogP** can be $[0, 1]$.
- **Similarity to a nearest neighbor (SNN)** measures an average Tanimoto similarity (m_G, m_R) between fingerprints of a molecule m_G from the generated set G and its nearest neighbor molecule m_R in the real dataset R. It is computed by [46]:

$$\mathrm{SNN(G, R)} = \frac{1}{|G|} \sum_{m_G \in R} \max_{m_R \in R} T(m_G, m_R) \quad (10)$$

  Here, we use the Morgan (extended connectivity) fingerprints estimation via the RDKit library. First, we compute the Morgan fingerprints of the generated molecule and randomly sample a number of molecules from the existing dataset. Then, we compare the similarities between the fingerprints of the generated molecule and each of the fingerprints of the existing molecules, and compute the average similarity scores. The similarity value can be represented as a measure of precision, and if the generated molecules are distinct from the manifold of the existing dataset, the similarity to the nearest neighbor will be very low. The range of **Similarity** is $[0, 1]$.

Note that all metrics, except for the **Validity**, are computed on only the valid molecules from the generated molecule set.

IV. Simulation Setups and Results

In this section, we conduct extensive simulations by using benchmark datasets to evaluate GraphGANFed.

A. Benchmark datasets

We use three data sets from MoleculeNet. [48]: ESOL, QM8, and QM9. These datasets have different properties in terms of physical chemistry and quantum mechanics. Table I presents the details of these datasets.

- ESOL [49] is a small dataset comprising water solubility data for 1,128 compounds. It has been used to train models which directly predict solubility from chemical structures that are encoded in SMILES string.
- QM8 [50] originated from a recent survey on representing quantum mechanical calculations of electronic fields and excited state energy of small molecules. There are 22,000 molecules, each of which comprises up to eight heavy atoms, that are collected using techniques, such

TABLE I: Benchmark datasets.

Category	Dataset	#Compds.	Avg. #Atoms	Avg. #Bonds
Physical Chem.	ESOL	1,128	13.29	40.65
Quantum Mech.	QM8	21,786	7.77	23.95
	QM9	133,885	8.80	27.60

as second-order approximate coupled-cluster (CC2) and time-dependent density functional theories (TDDFT).

- QM9 [51] is an extensive dataset that provides geometric, electronic, energetic, and thermodynamic properties for a section of the GDB-17 database. It contains 134,000 stable organic molecules with up to 9 heavy atoms. The molecules in the dataset are reproduced using density functional theory (B3LYP/6-31G (2df,p) based DFT).

We randomly split each dataset into training, validation, and testing sets based on a ratio of 80:10:10. The training set is used to train the models via FL settings, and the validation and test sets are used for hyperparameter tuning and model evaluation, respectively. We distribute the samples in the training dataset to the clients based on the independent and identical distribution (IID) and non-IID, and analyze the performance of GraphGANFed under these two scenarios. Specifically, each sample in terms of a molecule is labeled by its molecular formula (e.g., CH_3, CH_4, NH_3), and molecules with the same label are in the same class. For IID, molecules in each class are divided into K groups of equal size. Each group is assigned to a client and K is the total number of clients in FL. For non-IID, which basically comprises two aspects, i.e., 1) the numbers of data samples from different classes are unbalanced among clients, and 2) the total number of data samples among the clients is different. To implement non-IID, we pick a random number of molecules in each class and allocate them to each client.

B. Hardware and hyperparameter configurations

In all simulations, we use a fixed 16-dimensional random noise vector $\boldsymbol{z}$ sampled after every $1,000$ local epochs for ESOL and QM8, and 100 local epochs for QM9. The batch size is 16 and the number of local epochs is $E = 1,000$ for each participated client. The gradient penalty coefficient is $\gamma = 10$ to calculate the loss function of the discriminator in Eq. (5). Also, both the generator and discriminator models are trained by using the Adam optimizer with the exponential decay rates $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and the learning rate (LR) equals 0.0001. Here, the LR for both generator and discriminator are decayed by a value of 100 after each $1,000$ epochs. All the simulations are conducted on a computer with 2 NVIDIA Tesla K40m GPUs and 12 GB memory. The source code of GraphGANFed and simulation results are available at https://github.com/danielmanu93/GraphGANFed.

C. Simulation results

1) Model convergence analysis: Assume that the structures of the discriminator and generator are $([64, 128], 64, [128, 1])$ and $([32, 64, 128])$ respectively, where each value indicates the number of neurons/channels in a layer for the GCN model described in Fig. 3. For example, $([64, 128], 64, [128, 1])$ implies

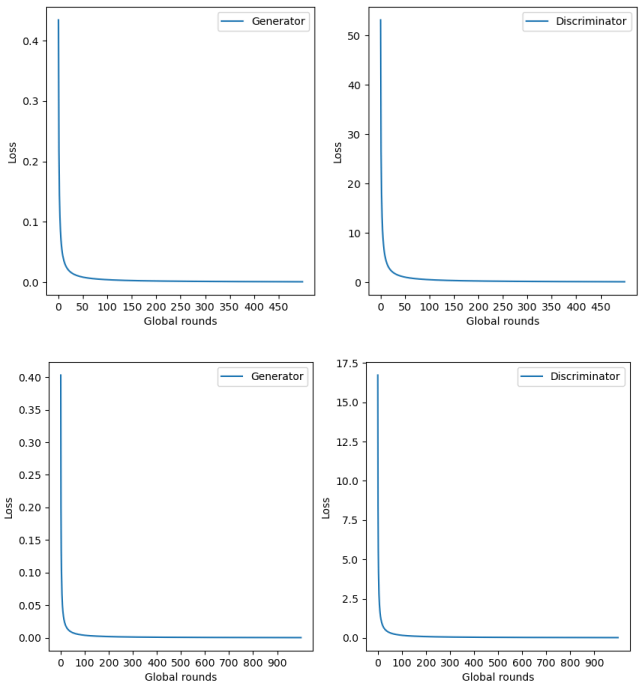


Fig. 5: Learning curves for the global generator and discriminator in IID (top) and non-IID (bottom).

that, in the discriminator, there are 64 and 128 channels in the first two convolution layers for atom feature extraction, 64 neurons in the hidden layer for feature dimension reduction, and 128 and 1 neurons in the last two layers for feature aggregation. There are $K = 4$ clients to train the discriminator and generator based on FL, and Fig. 5 shows the loss function curves of the discriminator and generator over different global rounds for the IID and non-IID settings. We can observe that both the generator and discriminator models converged after 60 global iterations and 150 global iterations in IID and non-IID settings, respectively, which demonstrates the feasibility of our proposed GraphGANFed framework. Also, it is common to have a slower convergence rate in the non-IID setting as compared to IID because of the high data distribution bias among the clients caused by non-IID.

2) Effect of different generator architectures on the metrics: The generator, which is one critical component in GraphGANFed, is responsible for generating novel and effective molecules. Hence, we will analyze the performance of GraphGANFed by varying the complexity of the generator (i.e., the hidden dimensions and number of hidden layers). Tables II and III show the results of applying different hidden dimensions and layers in terms of the complexities of the generator for the IID and non-IID settings, respectively, where the second column in each table indicates different hidden dimensions and layers of the generator. The 5th-11th columns imply the values of seven metrics mentioned in Section III.C to evaluate the performance of the molecules generated by GraphGANFed. For ESOL, which is a small dataset consisting of only 1,128 compounds, we observe the results for the IID setting in Table II and find that as the generator complexity

increases, the values of QED, Uniqueness, and LogP decrease, i.e. $(0.52, 0.34, 0.34)$, $(98.8, 0.9, 0.9)$ and $(0.68, 0.26, 0.26)$, respectively, but the values of Validity and Similarity increase, i.e., $(73.2, 100.0, 100.0)$ and $(0.0001, 0.0001, 0.0003)$, respectively. Similar results are observed in the non-IID setting, as shown in Table III. The reason why increasing the generator complexity increases Validity is that a high-complexity generator can capture complex patterns in the data and represent hierarchical features (from bonds to atoms to functional groups) effectively. However, as the generator complexity becomes higher than the dataset, overfitting occurs and leads to mode collapse. Hence, we can observe that as the complexity increases, the values of QED, Uniqueness, and LogP reduce. Thus, there is a tradeoff between Validity/Similarity and QED/Uniqueness/LogP, and the tradeoff can be adjusted by modifying the generator complexity. Moreover, a low generator complexity, i.e., [32, 64] is recommended for a small dataset like ESOL to generate molecules that have good values for most of the metrics whilst eliminating mode collapse. Note that Validity and Uniqueness have negative dependence, i.e., as Validity increases, Uniqueness decreases, and vice versa. This is because generating valid molecules often requires the model to follow well-established chemical rules, which limits the diversity and uniqueness of the generated molecular structures since the valid molecules tend to share common chemical motifs and features with molecules in the existing dataset. Hence, it is impossible to simultaneously maximize both Validity and Uniqueness. *In our simulation, without further specification, we consider Validity as the most critical metric. That is, one model structure is better than the other iff its Validity value is higher than the other.* The reason for Validity being the most critical metric is that other metrics are calculated based on Validity, and Validity shows positive dependence on most of the other metrics, except for Uniqueness.

For QM8, which is a medium dataset that has more compounds than ESOL but less than QM9, we observe the results for the IID setting in Table II and find that as the generator complexity increase, the values for QED, Validity, and LogP increase, i.e., $(0.33, 0.46, 0.48)$, $(0.1, 61.1, 91.6)$ and $(0.30, 0.72, 0.74)$, respectively, but the values for Diversity, Uniqueness, and Similarity decrease, i.e., $(1.00, 0.99, 0.99)$, $(100.0, 24.1, 9.4)$ and $(0.0141, 0.0078, 0.0059)$, respectively. The tradeoff changes from "between Validity/Similarity and QED/Uniqueness/LogP" for ESOL to "between QED/Validity/LogP and Diversity/Uniqueness/Similarity" for QM8. Also, the results for non-IID in Table III show similar results. Also, we can observe that the low complexity generator, i.e., [64, 128] results in mode collapse in terms of Validity close to 0 for both the IID and non-IID settings. A slightly higher complexity generator, i.e., [128,256] or [256,512], can generate molecules with good values for all the metrics whilst eliminating mode collapse.

For QM9, which is a large dataset, we can observe for the IID setting in Table II that as we increase the generator complexity, the values of QED, Validity and LogP increase, i.e., $(0.46, 0.46, 0.50)$, $(1.4, 28.5, 72.2)$ and $(0.60, 0.72, 0.69)$, respectively, but the values of Uniqueness and Similarity decreases, i.e., $(96.3, 34.3, 10.9)$ and $(0.0378, 0.0007, 0.0297)$, respectively. In the non-IID setting, we observe the same results. Hence, if Validity is considered the most critical metric, then a high-complexity generator is recommended to apply for a large training dataset, such as QM9.

3) Effect of different discriminator architectures on the metrics: The discriminator is the key component to determine the performance of GraphGANFed, and so we will analyze the performance in terms of the values of the metrics (that we mentioned in Section III.C) by varying the complexity of the discriminator. Tables IV and V show the results of applying different dimensions in terms of complexities of the discriminator for the IID and non-IID settings, respectively, where the third column in each table indicates different dimensions of the discriminator. For ESOL, we observe the results for the IID setting in Table IV and find that as the discriminator complexity increases, the values of QED, Uniqueness, LogP, and Similarity decrease, i.e., $(0.54, 0.34, 0.34)$, $(100, 0.9, 0.9)$, $(0.73, 0.26, 0.26)$, and $(0.0146, 0.0004, 0.0004)$, respectively, but the value of Validity increases, i.e., $(70.5, 100, 100)$. The same results are observed in the non-IID setting, as shown in Table V. The reason for having the increasing Validity as the discriminator complexity increases is that training a higher complexity model over a small dataset, such as ESOL, leads to discriminator overfitting. This overfitting pushes the generator to generate molecules with the same or very similar structure as the molecules in the dataset, resulting in higher Validity. On the other hand, training a higher complexity model over a small dataset may lead to mode collapse, which reduces the values of QED, Uniqueness, and LogP. Thus, there is a tradeoff among Validity and QED/Uniqueness/LogP/Similarity, and the tradeoff can be adjusted by changing the discriminator complexity. In addition, for a small dataset like ESOL, a low discriminator complexity, i.e., [32, 64], 32, [64,1], is preferred to generate molecules that have considerable metric values.

For QM8, we observe the results for the IID setting in Table IV and find that as the discriminator complexity increases, the values of QED, Diversity, and Validity decrease, i.e., $(0.49, 0.46, 0.45)$, $(1.00, 0.99, 0.99)$, and $(78.4, 26.9, 13.7)$, respectively, but the values of Uniqueness, LogP, and similarity increases, i.e., $(18.1, 51.4, 67.9)$, $(0.54, 0.59, 0.62)$, and $(0.0067, 0.0261, 0.0336)$, respectively. The tradeoff changes from "between Validity and QED/Uniqueness/LogP" for ESOL into "between QED/Diversity/Validity and Uniqueness/LogP/Similarity" for QM8. Also, if we observe the results for non-IID in Table V, we can find that as the discriminator complexity increases, only Uniqueness decreases, while QED, Validity, LogP, and Similarity increase. Hence, we conclude that the tradeoff among different evaluation metrics may change based on the size of the dataset and the data sample distribution among clients. In addition, for a medium dataset like QM8, a medium discriminator dimension, such as $[64, 128], 64, [128, 1]$ or slightly smaller, is preferred to generate the molecules that have considerable metric values.

For QM9, we can observe from the results that as we increase the discriminator complexity, QED, Validity, and Similarity increase, but Uniqueness reduces in the IID setting as shown in Table IV. In the non-IID setting, as shown in Table

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

TABLE II: Performance of GraphGANFed with different generators and the same discriminator in IID.

Datasets	Generator Dimension	Discriminator Dimension	Number of Clients	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
ESOL	[32,64]	[32,64],32,[64,1]	3	0.52	1.00	73.2	98.8	100.0	0.68	0.0001
	[32,128]			0.34	1.00	100.0	0.9	100.0	0.26	0.0001
	[16,32,64]			0.34	1.00	100.0	0.9	100.0	0.26	0.0003
QM8	[64,128]	[64,128],64,[128,1]	3	0.33	1.00	0.1	100.0	100.0	0.30	0.0141
	[128,256]			0.46	0.99	61.1	24.1	100.0	0.72	0.0078
	[64,128,256]			0.48	0.99	91.6	9.4	100.0	0.74	0.0059
QM9	[64,128]	[256,512],256,[512,1]	3	0.46	0.99	1.4	96.3	100.0	0.60	0.0378
	[32,64,128]			0.46	0.99	28.5	34.3	100.0	0.72	0.0007
	[64,128,256]			0.50	0.99	72.2	10.9	100.0	0.69	0.0297

TABLE III: Performance of GraphGANFed with different generators and the same discriminator in non-IID.

Datasets	Generator Dimension	Discriminator Dimension	Number of Clients	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
ESOL	[32,64]	[32,64],32,[64,1]	3	0.50	1.00	9.8	100.0	100.0	0.84	0.0014
	[32,128]			0.45	1.00	55.4	91.9	100.0	0.48	0.0015
	[16,32,64]			0.34	1.00	100.0	0.9	100.0	0.26	0.0000
QM8	[64,128]	[256,512],256,[512,1]	3	0.43	0.99	0.5	100.0	100.0	0.50	0.0428
	[128,256]			0.47	0.99	56.7	31.7	100.0	0.60	0.0008
	[64,128,256]			0.49	0.99	88.9	10.4	100.0	0.62	0.0378
QM9	[64,128]	[256,512],256,[512,1]	3	0.44	0.99	0.8	99.0	100.0	0.62	0.0484
	[32,64,128]			0.45	0.99	22.9	36.4	100.0	0.74	0.0008
	[64,128,256]			0.50	0.99	73.5	10.8	100.0	0.67	0.0078

TABLE IV: Performance of GraphGANFed with different discriminators and the same generator in IID.

Datasets	Generator Dimension	Discriminator Dimension	Number of Clients	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
ESOL	[32,128]	[32,64],32,[64,1]	4	0.54	1.00	70.5	100.0	100.0	0.73	0.0146
		[32,128],32,[128,1]		0.34	1.00	100.0	0.9	100.0	0.26	0.0004
		[32,256],32,[256,1]		0.34	1.00	100.0	0.9	100.0	0.26	0.0004
QM8	[32,64,128]	[64,128],64,[128,1]	4	0.49	1.00	78.4	18.1	100.0	0.54	0.0067
		[128,256],128,[256,1]		0.46	0.99	26.9	51.4	100.0	0.59	0.0261
		[256,512],256,[512,1]		0.45	0.99	13.7	67.9	100.0	0.62	0.0336
QM9	[64,128,256]	[64,128],64,[128,1]	4	0.45	0.99	17.5	47.5	100.0	0.72	0.0004
		[128,256],128,[256,1]		0.46	0.99	32.2	22.1	100.0	0.73	0.0004
		[256,512],256,[512,1]		0.50	0.99	72.9	12.6	100.0	0.72	0.0056

TABLE V: Performance of GraphGANFed with different discriminators and the same generator in non-IID.

Datasets	Generator Dimension	Discriminator Dimension	Number of Clients	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
ESOL	[32,128]	[32,64],32,[64,1]	3	0.45	1.00	55.4	91.9	100.0	0.48	0.0015
		[32,128],32,[128,1]		0.34	1.00	100.0	0.9	100.0	0.26	0.0004
		[32,256],32,[256,1]		0.34	1.00	100.0	0.9	100.0	0.26	0.0004
QM8	[32,64,128]	[64,128],64,[128,1]	3	0.46	0.99	28.2	54.7	100.0	0.43	0.0013
		[128,256],128,[256,1]		0.49	0.99	66.3	24.6	100.0	0.56	0.0053
		[256,512],256,[512,1]		0.49	0.99	88.9	10.4	100.0	0.62	0.0378
QM9	[64,128,256]	[64,128],64,[128,1]	3	0.49	0.99	10.3	75.0	100.0	0.46	0.0070
		[128,256],128,[256,1]		0.50	0.99	56.2	18.3	100.0	0.54	0.0073
		[256,512],256,[512,1]		0.50	0.99	73.5	10.8	100.0	0.67	0.0078

V, only Uniqueness decreases, but QED, Validity, LogP, and Similarity increase as we increase the discriminator dimension. Hence, we can derive the same conclusion that the tradeoff among different metrics may change based on the size of the data set and the data sample distribution among clients. In addition, for a large dataset like QM9, a large discriminator complexity is preferred to generate the molecules that have considerable metric values.

4) Effect of different client settings on the metrics: In this section, we evaluate how different numbers of clients affect the performance of GraphGANFed. Tables VI and VII show the metrics of GraphGANFed under different numbers of clients in IID and non-IID settings, respectively. For ESOL, we observed that as we increase the number of clients, QED, and Validity decrease, i.e. $(0.46, 0.43, 0.15)$ and $(91.9, 4.5, 1.8)$, respectively, but Uniqueness, LogP, and Similarity increase, i.e. $(80.6, 100, 100)$, $(0.95, 0.88, 1)$, and $(0.0007, 0.0202, 0.0453)$, respectively, for the IID setting. For non-IID, we observe that as we increase the number of clients, Validity, QED, Uniqueness, LogP, and Similarity decrease, i.e. $(69.6, 55.4, 100)$, $(0.41, 0.45, 0.34)$, $(74.4, 91.9, 0.9)$, $(1, 0.48, 0.26)$ and $(0.0047, 0.0015, 0.0011)$, respectively. Specifically, Validity decreases as the number of clients k increases from 1 to 3 (where $k = 1$ is equivalent to centralized training). The reason for Validity decreasing as k increases from 1 to 3 is that ESOL is a small dataset and as k increases to 3, the number of molecules allocated to the clients reduces as compared to the scenario when $k = 1$. Thus, the local models in each client will be trained based on fewer training samples, which leads to low model generalization and causes the generator to produce molecules with low Validity. Note that when $k = 7$, Validity increases to 100, which does not seem to follow the trend that Validity decreases as k increases. This is because when $k = 7$, the number of samples

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

TABLE VI: Performance of GraphGANFed with different numbers of clients in IID.

Datasets	Generator Dimension	Discriminator Dimension	Number of Clients	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
ESOL	[32,128]	[32,64],32,[64,1]	1	0.46	1.00	91.9	80.6	100.0	0.95	0.0007
			5	0.43	1.00	4.5	100.0	100.0	0.88	0.0202
			10	0.15	1.00	1.8	100.0	100.0	1.00	0.0453
QM8	[32,64,128]	[64,128],64,[128,1]	1	0.45	0.99	21.9	59.9	100.0	0.64	0.0202
			5	0.44	0.99	4.6	84.0	100.0	0.64	0.0281
			10	0.44	0.99	4.6	99.1	100.0	0.56	0.0108
QM9	[128,256,512]	[128,128],256,128,1	1	0.51	1.00	90.2	3.9	100.0	0.71	0.0071
			5	0.45	1.00	87.9	11.9	100.0	0.34	0.0068
			10	0.41	1.00	75.4	13.9	100.0	0.24	0.0064

TABLE VII: Performance of GraphGANFed with different numbers of clients in non-IID.

Datasets	Generator Dimension	Discriminator Dimension	Number of Clients	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
ESOL	[32,128]	[32,64],32,[64,1]	1	0.41	1.00	69.6	74.4	100.0	1.00	0.0047
			3	0.45	1.00	55.4	91.9	100.0	0.48	0.0015
			7	0.34	1.00	100.0	0.9	100.0	0.26	0.0011
QM8	[32,64,128]	[64,128],64,[128,1]	1	0.46	0.99	48.9	37.9	100.0	0.68	0.0103
			3	0.46	0.99	35.6	47.5	100.0	0.66	0.0108
			7	0.44	0.99	20.9	59.3	100.0	0.65	0.0309
QM9	[128,256,512]	[128,128],256,[128,1]	1	0.44	1.00	92.3	1.3	100.0	0.90	0.0253
			3	0.50	0.99	85.6	4.8	100.0	0.48	0.0078
			7	0.50	0.99	77.3	7.1	100.0	0.42	0.0039

allocated to each client further reduces and results in model collapse, where the generator produces molecules with limited variety in terms of low Uniqueness. In fact, the generator only produces the molecules that contain repetitive padding symbols $(*)$. By comparing the results between IID and non-IID, we can observe that the tradeoff changes from "between QED/Validity and Uniqueness/LogP/Similarity" in IID to "between QED/Uniqueness/LogP/Similarity and Validity" in non-IID as the clients are increased for ESOL.

For QM8, we observed that as the number of clients increases, QED, Validity, and LogP decrease, but Uniqueness increases in both IID and non-IID settings. However, the Similarity decrease for IID, i.e. $(0.0202, 0.0281, 0.0108)$ and increase for non-IID, i.e. $(0.0103, 0.0108, 0.0309)$, as the number of clients increases. Hence, we can derive that the tradeoff changes from "between QED/Validity/LogP/Similarity and Uniqueness" in IID to "between QED/Validity/LogP and Uniqueness/Similarity" in non-IID as the number of clients increases for QM8. Moreover, we are aware that the Validity values for QM8 are low (i.e., < 50) in both IID and non-IID cases. This is because we use the small discriminator architecture, which cannot effectively learn and evaluate the features of the molecules in QM8, a medium dataset. Applying a higher complexity discriminator can significantly increase Validity, which has been demonstrated in Table V, where Validity increases from 28.2 to 88.9 when the discriminator structure increases from $[64, 128], 64, [128, 1]$ to $[256, 512], 256, [512, 1]$.

In the IID setting for QM9, we observed that as the number of clients increases, QED, Validity, LogP, and Similarity decrease, i.e. $(0.51, 0.45, 0.41)$, $(90.2, 87.9, 75.4)$, $(0.71, 0.34, 0.24)$, and $(0.0071, 0.0068, 0.0064)$, respectively, and Uniqueness increases, i.e., $(3.9, 11.9, 13.9)$. However, in the non-IID setting, QED and Uniqueness increase, i.e. $(0.44, 0.50, 0.50)$ and $(1.3, 4.8, 7.1)$, respectively, and Diversity, Validity, LogP, and Similarity decrease, i.e. $(1.00, 0.99, 0.99)$, $(93.1, 85.6, 77.3)$, $(0.90, 0.48, 0.42)$, and $(0.0253, 0.0078, 0.0039)$, respectively. Hence, the tradeoff changes from "between QED/Validity/LogP/Similarity and Uniqueness" in IID to "between QED/Uniqueness and Diversity/Validity/LogP/Similarity" in non-IID as the number of clients increases. Therefore, we can derive that the tradeoff among different evaluation metrics may change by varying the number of clients in the system and the distributions of data samples among the clients.

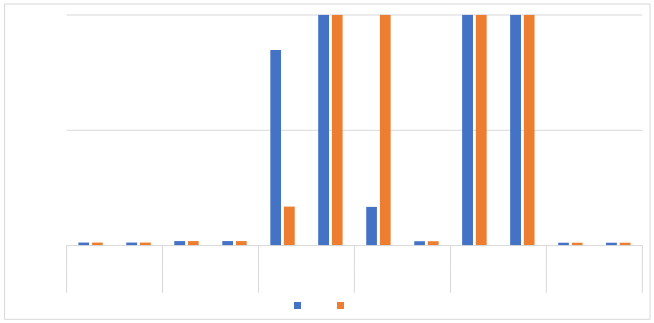


Fig. 6: Performance of GraphGANFed by varying the dropout ratios for ESOL.

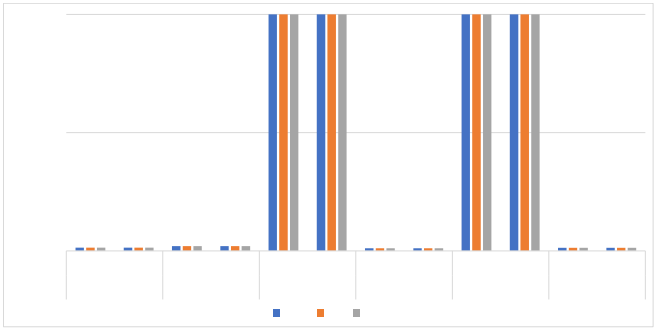


Fig. 7: Performance of GraphGANFed by varying the dropout ratios for QM8.

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

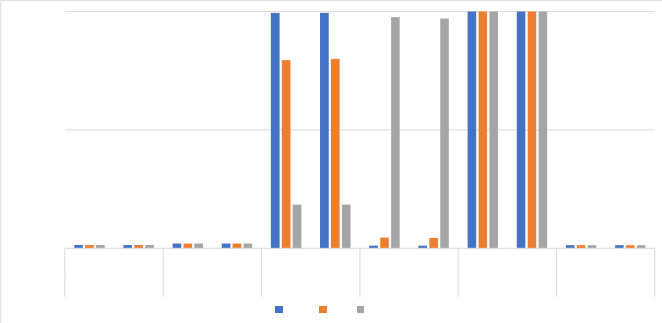


Fig. 8: Performance of GraphGANFed by varying the dropout ratios for QM9.

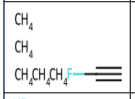
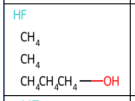
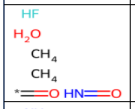
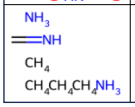
Molecules	QED	Diversity	Validity	Uniqueness	Novelty	LogP	Similarity
	0.46	1.00	62.3	23.4	100.0	0.73	0.0031
	0.48	0.99	58.2	31.6	100.0	0.60	0.0318
	0.43	0.99	40.8	62.9	100.0	0.28	0.0090
	0.48	1.00	92.1	10.4	100.0	0.74	0.0058

Fig. 9: Molecule samples and their metric results.

5) Effect of different dropout ratios on generation metrics: Applying dropout layers can reduce the model complexity without significantly affecting the model performance. We then analyze the performance of GraphGANFed if dropout layers are applied to both the generator and discriminator. For the generator, we apply dropout after the last layer. For the discriminator, we apply dropout after the first two convolution layers and the last two parallel hidden layers. Assume that the structure of the generator is $[32, 128]$ for ESOL, $[32, 64, 128]$ for QM8, and $[64, 128, 256]$ for QM9, and the structure of the discriminator is $[32, 64], 32, [64, 1]$ for ESOL, $[64, 128], 64, [128, 1]$ for QM8, and $[256, 512], 256, [512, 1]$ for QM9. The results are shown in Figs. 6-8.

Fig. 6 illustrates how different metrics change when the dropout ratio varies for ESOL. We can observe that as the dropout ratio increases from 0.25 to 0.6 in IID, Validity decreases from 84.8 to 16.9, and QED, Uniqueness, and LogP increase from 0.34 to 0.35, from 16.8 to 100, and from 0.26 to 0.27, respectively. For IID, the data distributions for the clients are the same. As the dropout ratio increases, more neurons are dropped in the generator and discriminator to reduce model complexity, but the models may not be able to learn some molecular representations in the dataset to generate valid samples, thus leading to a decrease in Validity. For non-IID, we observe that, as the dropout ratio increases from 0.25 to 0.6, the values of QED, Diversity, Validity, Uniqueness, Novelty, and LogP do not change. However, mode collapse happens on Uniqueness. Since it has been demonstrated that the high complexity of the discriminator lead to mode collapse on Uniqueness for ESOL in Table IV, we can speculate that the complexity of the discriminator is still too high even if the dropout ratio is 0.6, thus leading to mode collapse. We also observe that even though the validity is 100%, the generator could only generate padding symbols $(*)$.

Fig. 7 shows how different metrics change when the dropout ratio varies for QM8. We observe that increasing the dropout ratios does not affect the metrics for both IID and non-IID. In addition, we also observe mode collapse on Uniqueness, which may be caused by the high model complexity even if the dropout ratio is 0.75. Similar to the non-IID setting for ESOL, even though the Validity is 100%, the generator could only generate padding symbols $(*)$.

Fig. 8 shows how different metrics change when the dropout ratio varies for QM9. For IID, we can observe that as the dropout ratio increases from 0.25 to 0.5 to 0.75, QED and Uniqueness increase, i.e. $(0.34, 0.34, 0.36)$ and $(0.1, 3.5, 97.6)$, respectively. Yet, Validity decreases, i.e. $(99.4, 79.4, 18.4)$. Similar results are observed in the non-IID setting that, as the dropout ratio increases, QED and Uniqueness increase, but Validity decreases. Diversity and Novelty remain the same for all the dropout ratios in both IID and non-IID settings. Based on the observations in Figs. 6, 7 and 8, we can conclude that the model complexity can be adjusted by varying the dropout ratio. Also, having the right dropout ratio can avoid mode collapse. In addition, from all the experimental results presented above, we can observe that GraphGANFed can generate molecules to ensure high novelty (≈ 100) and diversity (> 0.9).

D. Case study

In this section, we present a case study to demonstrate the real-world practicality of GraphGANFed by analyzing some representative molecules generated by the model, and discuss their potential applications in drug discovery. The molecules are randomly selected among the set of molecules that are produced by the well-trained generator. The selected molecules are evaluated by using the mentioned seven metrics. As shown in Fig. 9, the results demonstrate that all the selected molecules have high QED and Diversity but low Similarity, which indicates that these molecules are different from the existing molecules in the data set and have a high probability of being developed into effective drugs or becoming novel and effective compounds to construct effective drugs. Moreover, the 1st, 2nd, and 4th molecules have high Validity and LogP scores, which indicate promising pharmacokinetic properties and good membrane permeability. These metric results can help pharmaceutical scientists select the best candidate(s) during the virtual screening process to accelerate drug discovery and reduce the overall cost. For instance, if only one molecule can be selected to conduct further laboratory and clinic testing, then, based on the results in Fig. 9, the 4th sample molecule seems to be the best candidate since it achieves high QED, LogP, and Validity scores.

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3349990

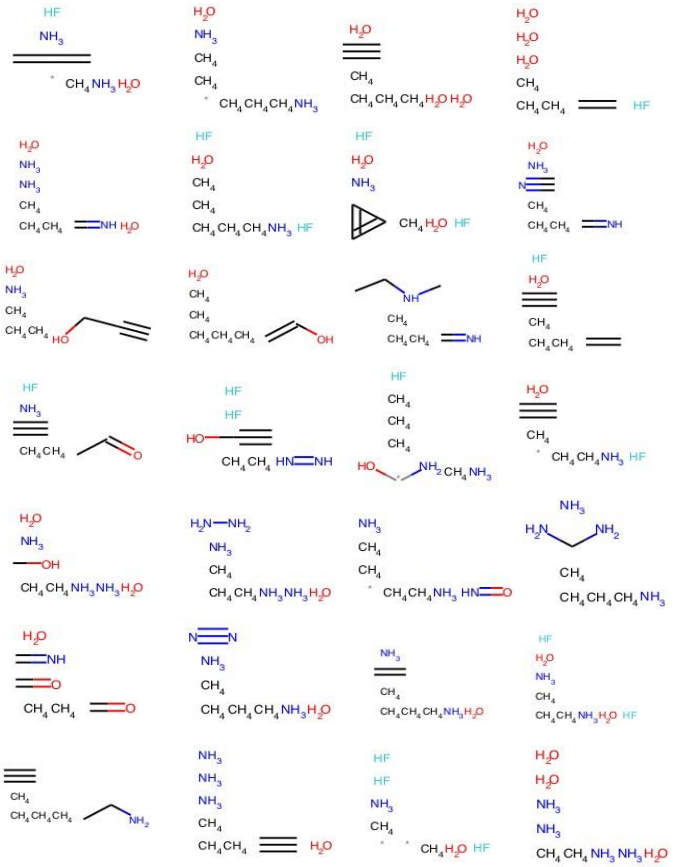


Fig. 10: Examples of molecules generated by GraphGANFed.

V. Conclusion

In this paper, we have proposed the GraphGANFed framework, which integrates GCN, GAN, and FL to generate novel and effective molecules for drug discovery, while preserving privacy. In GraphGANFed, an FL server aims to train the global generator and discriminator in a distributed manner, where the global generator generates new molecules that can preserve the properties of the existing molecules to deceive the global discriminator and the global discriminator distinguishes generated molecules from the existing molecules. In each global iteration, the FL server enables all the clients to train their local generators and discriminators over local datasets. The local generators and discriminators are uploaded and aggregated to derive a new global generator and global discriminator for the next global iteration. Extensive simulations have been made to prove the feasibility and effectiveness of GraphGANFed. The molecules generated by GraphGANFed can ensure high novelty (≈ 100) and diversity (> 0.9). Also, the simulation results suggest that 1) a lower complexity discriminator model can better avoid mode collapse for a smaller dataset in IID and non-IID, 2) there is a tradeoff among different metric values and the tradeoff may change based on the size of the dataset and the data sample distribution, 3) model complexity of the generator and discriminator can be adjusted by varying the dropout ratio, and having the right dropout ratio can avoid mode collapse.

Our future work has two major directions. First, the generator in GraphGANFed applies MLP as the model architecture, which may not be the optimal option as compared to other sophisticated model architectures, such as Variational Autoencoder and graph neural networks. We will evaluate the performance of GraphGANFed by applying different sophisticated model architectures. Second, as we mentioned before, Validity and Uniqueness have negative dependence, and so it is impossible to maximize both metrics. The current GraphGANFed architecture is unable to adjust the balance or tradeoff between Validity and Uniqueness. In our future work, we will modify the GraphGANFed architecture to enable the system to adjust the balance among different metrics. A possible solution is to modify the loss function of the generator to integrate the corresponding metric values, e.g., $\mathcal{L}^{gen} = -\lambda_1 \times \log D(\tilde{\boldsymbol{V}}^{gen}, \tilde{\boldsymbol{A}}^{gen}) - \lambda_2 \times Validity - \lambda_3 \times Uniqueness$, where λ_1, λ_2, and λ_3 are the weights associated with the performance of generator, Validity, and Uniqueness. If the generator aims to minimize the loss value $\mathcal{L}^{gen}$, then it has to generate the molecules with high Validity and Uniqueness, and adjusting the balance between Validity and Uniqueness can be achieved by calibrating the values of λ_2 and λ_3.

References

[1] A. Nouira, J. Crivello, and N. Sokolovska, "Crystalgan: Learning to discover crystallographic structures with generative adversarial networks," *CoRR*, vol. abs/1810.11203, 2018.

[2] W. Walters and R. Barzilay, "Applications of deep learning in molecule generation and molecular property prediction," *Accounts of Chemical Research*, vol. 54, 12 2020.

[3] D. Manu, Y. Sheng, J. Yang, J. Deng, T. Geng, A. Li, C. Ding, W. Jiang, and L. Yang, "Fl-disco: Federated generative adversarial network for graph-based molecule drug discovery: Special session paper," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–7.

[4] G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, and A. Aspuru-Guzik, "Objective-reinforced generative adversarial networks (organ) for sequence generation models," 2017. [Online]. Available: https://arxiv.org/abs/1705.10843

[5] V. Bagal, R. Aggarwal, P. K. Vinod, and U. Priyakumar, "Molgpt: Molecular generation using a transformer-decoder model," *J. Chem. Inf. Model.*, vol. 62, 10 2021.

[6] N. De Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," 2018. [Online]. Available: https://arxiv.org/abs/1805.11973

[7] V. Mouchlis, A. Afantitis, A. Serra, M. Fratello, A. Papadiamantis, V. Aidinis, I. Lynch, D. Greco, and G. Melagraki, "Advances in de novo drug design: From conventional to machine learning methods," *International Journal of Molecular Sciences*, p. 1676, 02 2021.

[8] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *J. Chem. Inf. Comput. Sci.*, vol. 28, pp. 31–36, 1988.

[9] T. Gaudelet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. R. Hayter, R. Vickers, C. Roberts, J. Tang, D. Roblin, T. L. Blundell, M. M. Bronstein, and J. P. Taylor-King, "Utilising graph machine learning within drug discovery and development," 2020. [Online]. Available: https://arxiv.org/abs/2012.05716

[10] E. J. Bjerrum and R. Threlfall, "Molecular generation with recurrent neural networks (rnns)," *CoRR*, vol. abs/1705.04612, 2017. [Online]. Available: http://arxiv.org/abs/1705.04612

[11] S. Huang, S. Chen, H. Peng, D. Manu, Z. Kong, G. Yuan, L. Yang, S. Wang, H. Liu, and C. Ding, "Hmc-tran: A tensor-core inspired hierarchical model compression for transformer-based dnns on gpu," ser. GLSVLSI '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 169–174. [Online]. Available: https://doi.org/10.1145/3453688.3461740

[12] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders,"

CoRR, vol. abs/1802.03480, 2018. [Online]. Available: http://arxiv.org/abs/1802.03480

[13] W. Jin, R. Barzilay, and T. S. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," *CoRR*, vol. abs/1802.04364, 2018. [Online]. Available: http://arxiv.org/abs/1802.04364

[14] J. Lim, S. Ryu, J. W. Kim, and W. Y. Kim, "Molecular generative model based on conditional variational autoencoder for de novo molecular design," *CoRR*, vol. abs/1806.05805, 2018. [Online]. Available: http://arxiv.org/abs/1806.05805

[15] E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. Aladinskaia, A. Aliper, and A. Zhavoronkov, "Adversarial threshold neural computer for molecular de novo design," *Molecular Pharmaceutics*, vol. 15, 03 2018.

[16] D. Polykovskiy, A. Zhebrak, D. Vetrov, Y. Ivanenkov, V. Aladinskiy, M. Bozdaganyan, P. Mamoshina, A. Aliper, A. Zhavoronkov, and A. Kadurin, "Entangled conditional adversarial autoencoder for de novo drug discovery," *Molecular Pharmaceutics*, vol. 15, 09 2018.

[17] D. Manu, P. M. Tshakwanda, Y. Lin, W. Jiang, and L. Yang, "Seismic waveform inversion capability on resource-constrained edge devices," *Journal of Imaging*, vol. 8, no. 12, 2022. [Online]. Available: https://www.mdpi.com/2313-433X/8/12/312

[18] X. Bresson and T. Laurent, "A two-step graph convolutional decoder for molecule generation," *CoRR*, vol. abs/1906.03412, 2019. [Online]. Available: http://arxiv.org/abs/1906.03412

[19] J. You, B. Liu, R. Ying, V. S. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *CoRR*, vol. abs/1806.02473, 2018. [Online]. Available: http://arxiv.org/abs/1806.02473

[20] D. Manu, S. Huang, C. Ding, and L. Yang, "Co-exploration of graph neural network and network-on-chip design using automl," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 175–180. [Online]. Available: https://doi.org/10.1145/3453688.3461741

[21] X. Bresson and T. Laurent, "A two-step graph convolutional decoder for molecule generation," *CoRR*, vol. abs/1906.03412, 2019. [Online]. Available: http://arxiv.org/abs/1906.03412

[22] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "CVAE-GAN: fine-grained image generation through asymmetric training," *CoRR*, vol. abs/1703.10155, 2017. [Online]. Available: http://arxiv.org/abs/1703.10155

[23] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "Gan-based synthetic brain mr image generation," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 734–738.

[24] P. Andreini, S. Bonechi, M. Bianchini, A. Mecocci, and F. Scarselli, "Image generation by gan and style transfer for agar plate image segmentation," *Computer Methods and Programs in Biomedicine*, vol. 184, p. 105268, 12 2019.

[25] S. Chen, D. Xue, G. Chuai, Q. Yang, and Q. Liu, "FL-QSAR: a federated learning-based QSAR prototype for collaborative drug discovery," *Bioinformatics*, vol. 36, no. 22-23, pp. 5492–5498, 12 2020. [Online]. Available: https://doi.org/10.1093/bioinformatics/btaa1006

[26] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," pp. 70–246, 01 2018.

[27] J. Yao and N. Ansari, "Secure federated learning by power control for internet of drones," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1021–1031, 2021.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[29] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[30] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017. [Online]. Available: http://arxiv.org/abs/1704.00028

[32] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, aug 2016. [Online]. Available: https://doi.org/10.1007\%2Fs10822-016-9938-8

[33] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature Communications*, vol. 8, no. 1, jan 2017. [Online]. Available: https://doi.org/10.1038\%2Fncomms13890

[34] Q. Feng, E. V. Dueva, A. Cherkasov, and M. Ester, "PADME: A deep learning-based framework for drug-target interaction prediction," *CoRR*, vol. abs/1807.09741, 2018. [Online]. Available: http://arxiv.org/abs/1807.09741

[35] T. Ma, C. Xiao, J. Zhou, and F. Wang, "Drug similarity integration through attentive multi-view graph auto-encoders," *CoRR*, vol. abs/1804.10850, 2018. [Online]. Available: http://arxiv.org/abs/1804.10850

[36] W. Jin, C. W. Coley, R. Barzilay, and T. S. Jaakkola, "Predicting organic reaction outcomes with weisfeiler-lehman network," *CoRR*, vol. abs/1709.04555, 2017. [Online]. Available: http://arxiv.org/abs/1709.04555

[37] Y. Wang, J. Xiao, T. O. Suzek, J. Zhang, J. Wang, Z. Zhou, L. Han, K. Karapetyan, S. Dracheva, B. A. Shoemaker *et al.*, "Pubchem's bioassay database," *Nucleic acids research*, vol. 40, no. D1, pp. D400–D412, 2012.

[38] T. Unterthiner, A. Mayr, G. Klambauer, and S. Hochreiter, "Toxicity prediction using deep learning," 2015. [Online]. Available: https://arxiv.org/abs/1503.01445

[39] M. Mysinger, M. Carchia, J. Irwin, and B. Shoichet, "Directory of useful decoys, enhanced (dud-e): Better ligands and decoys for better benchmarking," *Journal of medicinal chemistry*, vol. 55, pp. 6582–94, 06 2012.

[40] S. Rohrer and K. Baumann, "Maximum unbiased validation (MUV) data sets for virtual screening based on pubchem bioactivity data," *J. Chem. Inf. Model.*, vol. 49, pp. 169–84, 03 2009.

[41] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016. [Online]. Available: https://arxiv.org/abs/1611.01144

[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[43] H. Aetesam and S. K. Maji, "Perceptually-motivated adversarial training for deep ensemble denoising of hyperspectral images," *Remote Sensing Letters*, vol. 13, no. 8, pp. 767–777, 2022.

[44] D. Mu, H. Li, H. Liu, L. Dong, and G. Zhang, "Underwater image enhancement using a mixed generative adversarial network," *IET Image Processing*, vol. 17, no. 4, pp. 1149–1160, 2023.

[45] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[46] D. Polykovskiy, A. Zhebrak, B. Sánchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. I. Nikolenko, A. Aspuru-Guzik, and A. Zhavoronkov, "Molecular sets (MOSES): A benchmarking platform for molecular generation models," *CoRR*, vol. abs/1811.12823, 2018. [Online]. Available: http://arxiv.org/abs/1811.12823

[47] G. Landrum, "Rdkit documentation," *Release*, vol. 1, no. 1-79, p. 4, 2013.

[48] Z. Wu, B. Ramsundar, E. Feinberg, J. Gomes, C. Geniesse, A. Pappu, K. Leswing, and V. Pande, "Moleculenet: A benchmark for molecular machine learning," *Chemical Science*, vol. 9, 03 2017.

[49] J. Delaney, "Esol: Estimating aqueous solubility directly from molecular structure," *Journal of chemical information and computer sciences*, vol. 44, pp. 1000–5, 05 2004.

[50] R. Ramakrishnan, M. Hartmann, E. Tapavicza, and O. A. von Lilienfeld, "Electronic spectra from TDDFT and machine learning in chemical space," *The Journal of Chemical Physics*, vol. 143, no. 8, p. 084111, aug 2015. [Online]. Available: https://doi.org/10.1063\%2F1.4928757

[51] L. Ruddigkeit, R. Deursen, L. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," *J. Chem. Inf. Model.*, vol. 52, 10 2012.