IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 May 2023, accepted 30 May 2023, date of publication 2 June 2023, date of current version 15 June 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3282248*

RESEARCH ARTICLE

# Validity Improvement in MolGAN-Based Molecular Generation

**JIAYI FAN, SEUL KI HONG, AND YONGKEUN LEE, (Member, IEEE)**
Department of Semiconductor Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea
Corresponding author: Yongkeun Lee (yklee@seoultech.ac.kr)

This work was supported by the Seoul National University of Science and Technology.

**ABSTRACT** Designing molecules that have desired properties is one of the challenging tasks of drug design. Among the many molecular generative models, a generative adversarial network (GAN), is able to generate molecule structures with desirable chemical properties via reinforcement learning. Generating valid molecules is the foremost task of any molecular generative model, since invalid molecules cannot be synthesized. We base our research on a molecular generative adversarial network (MolGAN) architecture to investigate how the validity score is influenced in different scenarios. First, we verify that the Vanilla GAN structure can produce valid molecules in measure, and that the reward network, along with Vanilla GAN, can further increase the validity score in a reinforcement learning manner. Then, the procedure for solely optimizing the validity score is tested, followed by an assessment of validity score maintenance while other chemical properties are being optimized. We found that multiple aspects, including loss functions, hyper parameters, and training sequences, must be carefully considered and optimized to raise the validity score of molecular generation alone or in concurrence with the optimizing of other chemical property scores.

**INDEX TERMS** Drug design, molecular generation, generative adversarial network (GAN), molecular generative adversarial network (MolGAN).

## I. INTRODUCTION

In the drug design process, finding valid and synthesizable molecules is a rather daunting task; it normally takes decades and costs billions of dollars to bring new drugs to the market [1], [2], [3], [4]. The reason behind this delay and expense is that the realm of possible molecules is enormous, and it is impractical to assess them all. To mitigate this challenge, a de novo design is often adopted. One of the main approaches in computer-aided drug design, de novo design uses generative models to produce novel molecules with desired chemical properties. In recent years, the use of deep learning models to perform molecular generation and molecular optimization has emerged at a fast pace. The deep learning models feature a deep neural network architecture that usually possesses more than three layers.

To utilize a deep molecular generation model, the molecules must be represented in an encoding that the model can recognize. Furthermore, an encoding scheme should capture the essential structural information of the molecules. One of the most popular ways to do this is called a simplified molecular-input line-entry system (SMILES) string representation [5], [6], which encodes the molecular graph into a sequence of ASCII characters using a depth-first graph traversal. However, there are some major flaws in using SMILES representations for molecules: the molecule can vary drastically with changes to the order of the string or even to a single character, and the generative model can generate invalid sequences based on the SMILES. Therefore, researchers rely on molecular graphs [7], [8], [9], [10] to strengthen the molecular generation so that all the generated graphs can be valid graphs.

Several popular deep molecular generation models have been proposed. A recurrent neural network (RNN) is one of the fundamentals of deep molecular generation [11], [12]. It is inherently suitable for molecular generation due to its capabilities of sequence modeling and generation and its use of the SMILES representation of molecules. Another popular deep molecular generative model type is the variational autoencoder (VAE) [13], [14]. However, it has been

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

reported that the molecules generated by VAE-based models suffer from low validity scores. The generative adversarial network (GAN) is a promising deep molecular generative model [15], [16], [17], [18], [19], [20]. It includes two major components: a discriminator and a generator. The goal of a GAN model is to generate samples with distributions that approximate those of the training set. In molecular generation, the goal of the generator is to generate molecules that are similar to those in the training set in order to fool the discriminator, while the discriminator tries to distinguish between generated and real molecules. They can be considered to be like players in a minimax game. This process is very useful, even when we do not know the real distribution.

MolGAN [21] applies GAN to the molecular generation problem, adapting GAN to operate on graph-structured data. It combines a reinforcement learning objective to lead the generation toward certain desired chemical properties. This approach is the first to discard SMILES representation and embrace the proposed graph representation in GAN molecule synthesis. It achieves significantly better validity scores compared to those of several VAE-based generative models, and higher chemical property scores than those of several SMILE-based GAN models. However, the indispensable mandate of any molecular generative model is to produce chemically valid molecules. Only then can we discuss other desired chemical properties. Therefore, how to generate molecules that are as chemically valid as possible must be investigated. Furthermore, the perfect discriminator problem and the mode collapse problem, which also hinder the training of a GAN model, must be solved.

An improved version of MolGAN, called L-MolGAN, was proposed in [22] to address the problems of the original MolGAN. With the original MolGAN model, in attempts to generate molecules with more than nine atoms, the generated graphs tend to be disconnected, resulting in invalid molecules. L-MolGAN introduces a graph expansion mechanism that suppresses the generation of disconnected graphs by penalizing them, using depth-first searches to check the connectivity of the generated graphs. However, L-MolGAN still need to be optimized further. For simplicity, we based our research on the original MolGAN model, but our ideas can also be applied to L-MolGAN.

In this paper, multiple factors are studied to show how the validity score can be improved based on the MolGAN model. We first evaluate the model with only a discriminator to train the generator (i.e., Vanilla GAN architecture), without explicitly optimizing the validity score. Next, we incorporate a reward network to output a validity score, and then train the generator to generate more valid molecules, considering multiple aspects, such as the loss functions, hyper parameters, reward network architecture, and training sequences. Then, we investigate how to improve the validity score, and then how to maintain it. We assess parameters for maintaining validity by concurrently optimizing other property scores, because validity is a prerequisite for the optimization of other

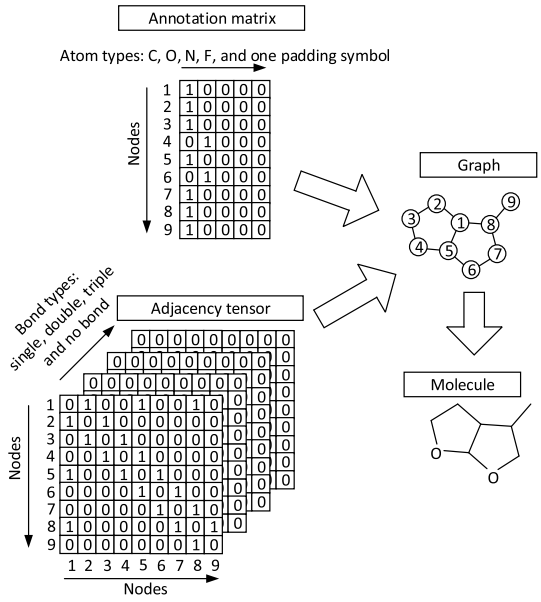


**FIGURE 1.** Molecule representation in graph and matrix forms.

properties. Two reward networks are used simultaneously: one reward network is for improving validity, and the other is for improving other chemical properties. The conditions in which the reward networks perform better in training the generator are then evaluated. Next, the loss function is modified from the original MolGAN. Finally, the experimental results are provided and discussed to show how validity can be improved.

## II. MOLGAN ARCHITECTURE

### A. MOLECULES AS GRAPHS AND MATRIX

The first step when using MolGAN is to represent molecules as graphs (Fig. 1). A molecule graph is an undirected graph with nodes $V$ and edges $E$. The nodes represent the atoms, and the edges represent the atomic bonds between different atoms in a molecule. A node feature matrix $X$ (annotation matrix) is used that aggregates one-hot vectors from each atom in a molecule to indicate the different types of atoms. In this study, the maximum number of atoms, atomic types, and atomic bonding types in a molecule are restricted to 9, 5, and 4, respectively, for the sake of simplicity. This results in one $9 \times 5$ node feature matrix $X$ and one $4 \times 9 \times 9$ adjacent tensor $A$. In a node feature matrix $X$, the row indicates the atomic node, and the column indicates the atomic type in that atomic node. Similarly, an adjacent tensor $A$ (adjacency tensor), which stores information regarding the edges in the graph, is used to indicate the atomic bond types, such as single or double atomic bonding. In adjacency tensor $A$, each $9 \times 9$ adjacent matrix provides information on the bonds between atomic nodes. If a node is bonded with another, the corresponding element in the adjacent matrix is marked as 1; otherwise, 0 is placed. In our case, with four different bonding types, there will be four $9 \times 9$ matrixes in the

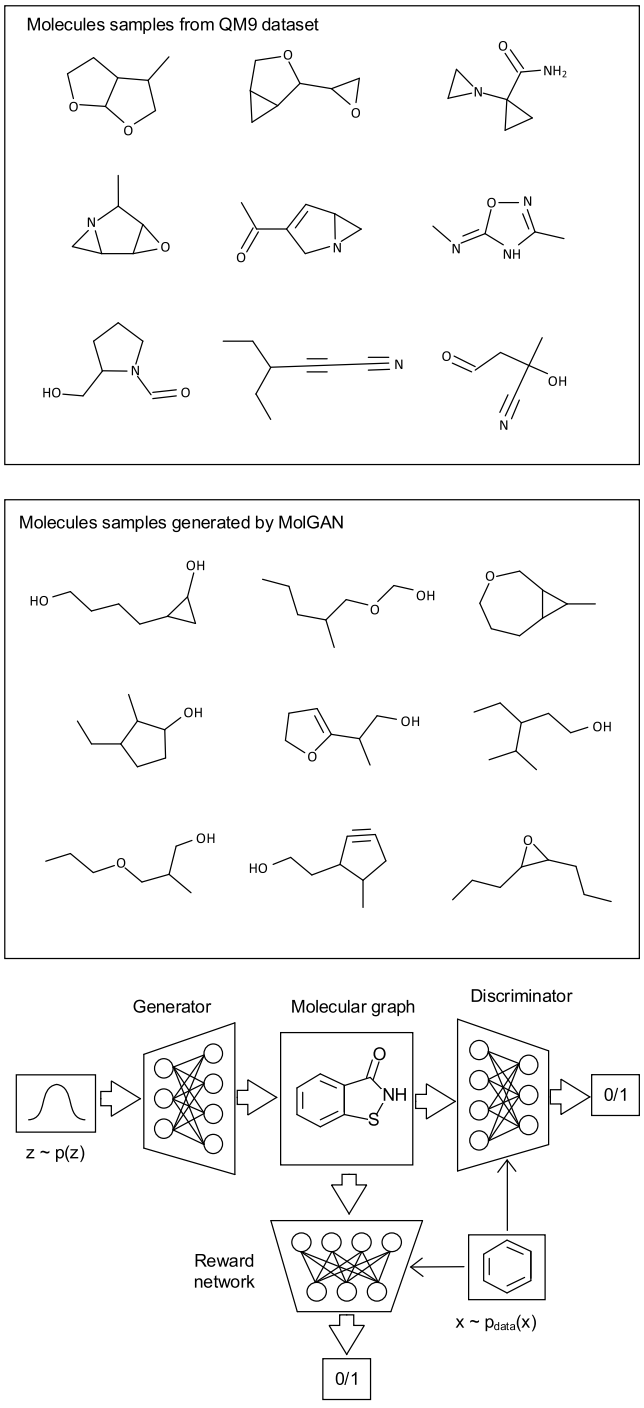


**FIGURE 2.** MolGAN architecture with molecule samples [21].

adjacent tensor $A$. The architecture of MolGAN, the chemical structures of some molecules generated by MolGAN, and the QM9 dataset are shown in Fig. 2.

### B. GENERATOR ARCHITECTURE

The architecture of the generator in MolGAN is a multilayer perceptron (MLP) with three layers that takes as an input a 32-dimensional noise vector $z$, sampled from a standard normal distribution, and uses tanh as an activation function. The numbers of hidden units in the MLP are 128, 256, and 512, respectively. The output is a probabilistic complete graph $\tilde{G} = (\tilde{A}, \tilde{X})$, which includes an adjacency matrix $\tilde{A}$ and a node feature matrix $\tilde{X}$. The generator in MolGAN is trained by a linear combination of the Wasserstein GAN (WGAN) loss and the reinforcement learning (RL) loss, as follows:

$$L_G = \lambda L_{G(WGAN)} + (1-\lambda)L_{G(RL)}, \tag{1}$$

$$L_{G(WGAN)} = -L_{fake}, \tag{2}$$

where $L_G$ is the generator loss; $L_{G(WGAN)}$ is the WGAN loss; the mean value of the discriminator output for one batch of the generated molecules, $L_{G(RL)}$, is the RL loss; and $\lambda$ is the parameter which determines the ratio between $L_{G(WGAN)}$ and $L_{G(RL)}$. By minimizing $L_G$, the generator is trying to fool the discriminator and generate molecules with a similar distribution as those in the true/real dataset. By minimizing $L_{G(WGAN)}$, $L_{fake}$ will be maximized, and $L_{fake}$ will be close to $L_{real}$.

### C. DISCRIMINATOR ARCHITECTURE

The discriminator is used to discriminate the input molecules as either real or fake; its task is to accept a molecule graph as input, and output a scalar ranging within $(-\infty, +\infty)$. To handle the graph structure data as an input, the discriminator and the reward network use a relational graph convolutional network to support graphs with multiple edge types. This architecture uses a relational graph convolution operation to convolve the feature representation of nodes $\tilde{X}$ using the adjacency tensor $\tilde{A}$, as follows:

$$h_i^{(l+1)} = tan\, h(f_s^{(l)}(h_i^{(l)}, x_i) + \sum_{j=1}^{N}\sum_{y=1}^{Y}\frac{\tilde{A}_{ijy}}{|N_i|}f_y^{(l)}(h_j^{(l)}, x_i)), \tag{3}$$

where $h_i^{(l)}$ is the signal of the node $i$ at layer $l$ and $f_s^{(l)}$ is a linear transformation function between layers and $f_y^{(l)}$ is an edge type specific affinity function for each layer. $N_i$ denotes the set of neighbors for node $i$. After successive graph convolution operations are applied, the node embeddings are aggregated into a graph-level representation vector, as follows:

$$h'_G = tan\, h(\sum_{v\in V}\sigma(i(h_v^L, x_v))\odot tan(j(h_v^L, x_v))), \tag{4}$$

where $\sigma$ is a logistic sigmoid function, $i$ and $j$ are MLPs with a linear output layer and $\odot$ denotes element-wise multiplication. An MLP is then used to make the discriminator and reward network output a scalar value.

In MolGAN, the discriminator is trained using the WGAN- Gradient Penalty (GP) loss function, as shown in

$$L_D = L_{D(WGAN)} = L_{fake} - L_{real} + L_{gp}, \tag{5}$$

where $L_D$ is the discriminator loss; $L_{fake}$ is the mean value of the discriminator output for one batch of the generated molecules; $L_{real}$ is the mean value of the discriminator output for one batch of the real molecules; and $L_{gp}$ is the gradient

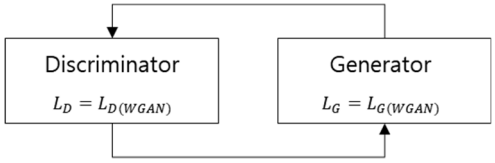


**FIGURE 3.** Training loop for the first scenario.

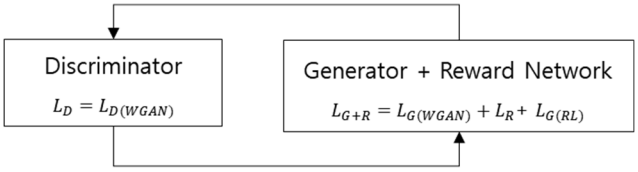


**FIGURE 4.** Training loop for the second and third scenarios.

penalty. The discriminator is trying to distinguish the generated molecules from the real ones by minimizing $L_D$, which will minimize $L_{fake}$ and maximize $L_{real}$, making $L_{fake}$ and $L_{real}$ as dissimilar as possible.

### D. REINFOREMENT LEARNING WITH REWARD NETWORK

Vanilla GAN can be used to generate molecules. However, the properties of the generated molecules are of the most importance in drug design. Validity, novelty, uniqueness, druglikeness, solubility, and synthesizability are some of the desired objectives in drug discovery. If the generated molecules are invalid and not synthesizable, they are completely useless. RL is adopted to optimize molecule generation toward desirable chemical properties. To assess the chemical properties of the generated molecules, MolGAN uses the external chemistry software RDKit to calculate a molecule's chemical properties, such as validity, druglikeness, solubility, and synthesizability. However, to provide gradients for the GAN training process, the MolGAN architecture creatively incorporates a reward network into the Vanilla GAN. The job of the reward network is to replace RDKit and provide accurate chemical scores for the generated molecules. Thus, the reward network should be trained to accept a molecule as input and output a chemical score that is close to RDKit's professional evaluation Conveniently, the reward network is designed to have the same architecture as the discriminator network. However, the reward network's output score ranges within $(0, 1)$ instead of $(-\infty, +\infty)$ to match the output score from the external chemistry software. The discriminator and reward network work independently; they do not share parameters.

By minimizing the root-mean-square error (RMS error) between the score obtained from the reward network and the score calculated from RDKit, the reward network can be trained to match the score from RDKit. Therefore, the RMS error is used as the loss to train the reward network, as shown in:

$$L_R = \frac{\sum (s_R - s_{RDKit})^2}{B}, \tag{6}$$

where $L_R$ is the loss for the reward network; $s_R$ is the output of the reward network; $s_{RDKit}$ is the score provided by RDKit; and $B$ is the batch size.

## III. PROPOSED VALIDITY IMPROVEMENT

The most important task of a molecular generative model is to output valid molecules. Only valid molecules are valuable for further discussion: A molecule must be valid to allow the molecular generative model to be trained for other chemical properties.

Several scenarios were considered to improve the validity score. The first scenario uses Vanilla GAN to generate molecules, as shown in Fig. 3. In this case, the only tutor for the generator is a discriminator. The generated molecules are likely to be valid because only real molecules are used in the training set. The role of the discriminator is to teach the generator to produce valid molecules. While the accuracy of the discriminator determines the validity score, it only indirectly guides the generator.

In the second scenario, as shown in Fig. 4, the reward network is added in a reinforcement learning manner because the discriminator was not sufficient enough to produce a high validity score in the first scenario. Therefore, adopting an idea proposed in the original MolGAN paper [21], a reward network is added for validity. It will directly guide/teach the generator on top of the first scenario. In addition to the first tutor (the discriminator), a second tutor (the reward network) is used to ensure a higher validity score.

The third scenario (Fig. 4) is used in the original MolGAN paper. The authors came up with a smart way to improve validity further: By optimizing several chemical property scores, they use the product of multiple scores (synthesizability score, solubility score, etc.) as the expected value in the loss function. A zero score is given to invalid molecules. Thus, the validity score is indirectly optimized when optimizing other chemical properties.

In the fourth scenario, shown in Fig. 4, the original MolGAN is modified. In the original MolGAN method, the generator and the reward network are jointly trained by adding the WGAN loss (for the generator) and the RMS error loss (for the reward network), as shown in (7). At first glance, the training of the model seems compact. However, the generator will collapse the reward network, or vice versa; thus, neither will be trained properly.

$$L_{G+R} = L_{G(WGAN)} + L_R \tag{7}$$

In the fifth scenario (Fig. 5), validity score improvement is studied using cases where the other chemical property scores are optimized. In these cases, the validity score is inherently maintained at a high level, otherwise the optimizing of the other scores would not occur. The loss function is formulated in (8), where $s_R$ is the output of the reward network (i.e., the solubility score), and $B$ is the batch size. The solubility score is averaged over all the generated molecules, whether or not they are valid. It should be noted that invalid molecules

are also counted when calculating the solubility score, even though their scores are all zeros. We use (8) (divided by batch size) instead of (9) (divided by the number of valid generated molecules $n_{valid}$ only) because the latter only improves the solubility score, but the former improves both the validity score and the solubility score.

$$L_{G(RL)} = -\frac{\sum s_R}{B} \tag{8}$$

$$L_{G(RL)} = -\frac{\sum s_R}{n_{valid}} \tag{9}$$

In the sixth scenario (Fig. 6), to prevent the validity score collapse problem from occurring while optimizing another chemical property (e.g., solubility), we incorporate two reward networks in parallel. One reward network is for solubility, and the other reward network is used to penalize situations that cause the validity score to decrease. The two reward networks share exactly the same architecture, and the loss function for both of these reward networks is (6). In the original MolGAN paper, first, a given molecule is fed into RDKit to obtain the molecule's validity score $s_{RDKit(validity)}$ and solubility score $s_{RDKit(solubility)}$. These two scores are then multiplied to yield a joint score, $s_{joint}$, with which the reward net is trained. However, it is problematic that, whether or not the molecule is valid, $s_{joint}$ becomes dependent only on $s_{RDKit(solubility)}$, and there is no effect at all from introducing the validity score from RDKit to train the reward network. To solve this problem, a separate reward network for the validity score is used. We first calculate the average score for each reward network and then multiply these average scores to construct the RL loss batch by batch, as shown in (10). The benefit of using the average score $\frac{\sum s_{R2(solubility)}}{B}$ is that it can capture both the solubility and validity scores, as explained in the fifth scenario. The average score $\frac{\sum s_{R1(validity)}}{B}$ is used to penalize the low validity score without losing the effect of the reward network for validity.

$$L_{G(RL)} = -\frac{\sum s_{R1(validity)}}{B} \times \frac{\sum s_{R2(solubility)}}{B} \tag{10}$$

The training loops and the corresponding loss functions for the aforementioned scenarios are shown in Figs. 3–6.

## IV. EXPERIMENTAL RESULTS AND DISCUSSION

The experiments were carried out on the QM9 dataset. This dataset is a subset of the GDB-17 chemical database, which enumerates 166.4 billion molecules of up to 17 atoms of C, N, O, S, and halogens. GDB-17 covers a molecular size range that is typical of many drugs and lead compounds. The QM9 subset has become the golden standard for machine learning predictions of various chemical properties. It contains 133,885 organic compounds and up to nine heavy atoms [23], [24].

(Experiment 1) Vanilla GAN (Fig. 3) was tested using the hypermeters in Table 2. The best validity score was 73.8, which was slightly lower than expected. It was concluded that indirect instruction by the discriminator is not enough, and

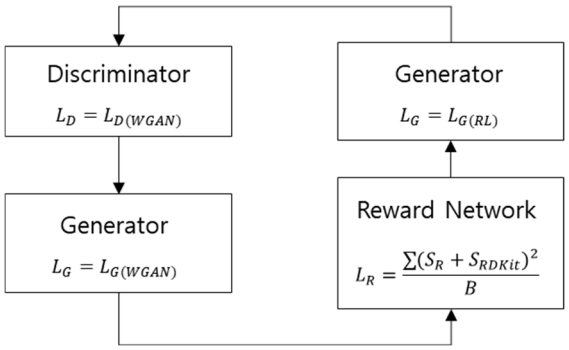


**FIGURE 5.** Training loop for the fourth and fifth scenarios.

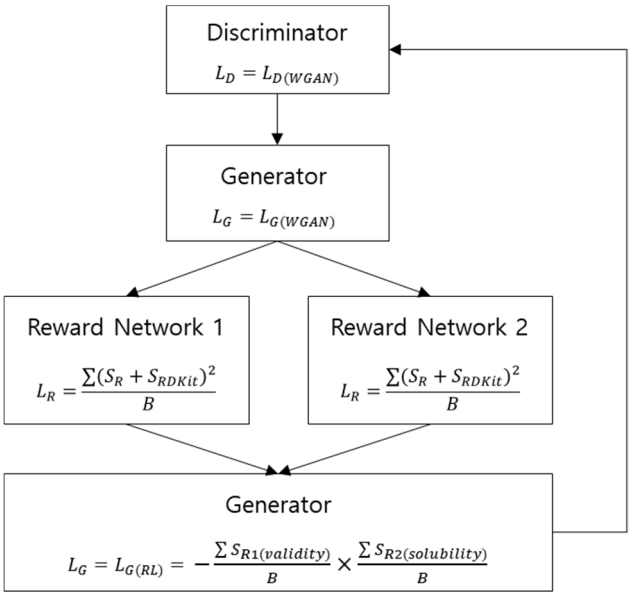


**FIGURE 6.** Training loop for the sixth scenario.

that additional direct instructions are necessary to guide the generator. Therefore, one reward network was added on top of Vanilla GAN in a reinforcement learning manner.

(Experiment 2) The original MolGAN model (Fig. 4) was adopted for this experiment. The best validity score obtained was 90.0, clearly indicating that an additional reward network can increase the validity score. The reward network must cope with any changes that occur in the model. Thus, the accuracy of the pretrained reward network worsens as the iteration progresses, eventually resulting in a low validity score.

(Experiment 3) Instead of using the pretrained reward network from Experiment 2, we renewed the reward network in every iteration. Then, a universal setup for the training of the reward network was tested, i.e., with a fixed learning rate and a fixed iteration number. A number of possible setups for the reward network were tested, as shown in Table 1. A learning rate of 0.005 and an iteration number of 5 were chosen because they provided comparable accuracies to those obtained using RDKit and the training time can be reduced by a higher learning rate and a lower iteration number. The reward network was incorporated into the Vanilla GAN

**TABLE 1.** Validity error using different learning rates and numbers of iterations.

| Iteration \ Learn Rate | 0.001 | 0.002 | 0.003 | 0.004 | 0.005 | 0.006 |
|---|---|---|---|---|---|---|
| **5** | 0.2743 | 0.1704 | 0.1352 | 0.0367 | 0.0231 | 0.5338 |
| **10** | 0.0869 | 0.0294 | 0.0176 | 0.0238 | 0.0133 | 0.0753 |
| **15** | 0.0418 | 0.0326 | 0.0127 | 0.0090 | 0.0130 | 0.0250 |
| **20** | 0.0344 | 0.0223 | 0.0117 | 0.0067 | 0.0099 | 0.0336 |
| **25** | 0.0246 | 0.0158 | 0.0080 | 0.0062 | 0.0080 | 0.0131 |
| **30** | 0.0275 | 0.0134 | 0.0075 | 0.0057 | 0.0068 | 0.0111 |

**TABLE 2.** Comparison of different experiments.

| Experiment | Scenario | Objective | Learning Rate | Ratio | Iterations | Batch Size | Best Validity Score |
|---|---|---|---|---|---|---|---|
| 1 | 1 | validity | D (0.0001)<br>G (0.0001) | 5:1 | 5000 | 16 | 73.8 |
| 2 | 2 | validity | D (0.0001)<br>G+$R_{validity}$ (0.0001) | 5:1 | 5000 | 16 | 90.0 |
| 3 | 3 | validity | D (0.0001)<br>G (0.0001)<br>$R_{validity}$ (0.005)<br>G (0.0001) | 1:1:5:1 | 5000 | 16 | 99.6 |
| 4 | 3 | validity | D (0.0001)<br>G (0.0001)<br>$R_{validity}$ (0.0001)<br>G (0.0001) | 1:1:1:1 | 5000 | 16 | 15.1 |
| 5 | 5 | validity,<br>druglikeness | D (0.0001)<br>G (0.0001)<br>$R_{druglikeness}$ (0.001)<br>G (0.0001) | 10:1:10:1 | 6000 | 16 | 91.2 |
| 6 | 5 | validity,<br>synthesizability | D(0.0001)<br>G (0.0001)<br>$R_{synthesizability}$ (0.004)<br>G (0.0001) | 10:1:10:1 | 6000 | 64 | 65.4 |

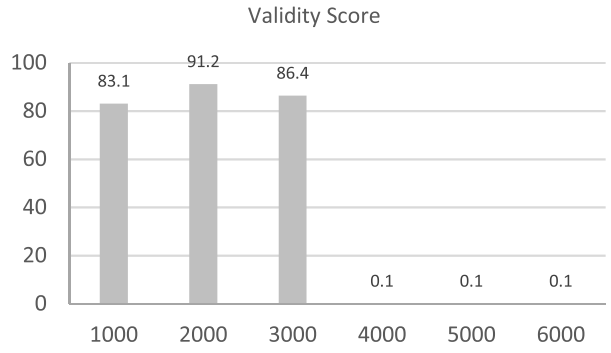


**FIGURE 7.** Validity score as a function of iteration. It is destroyed while optimizing druglikeness.

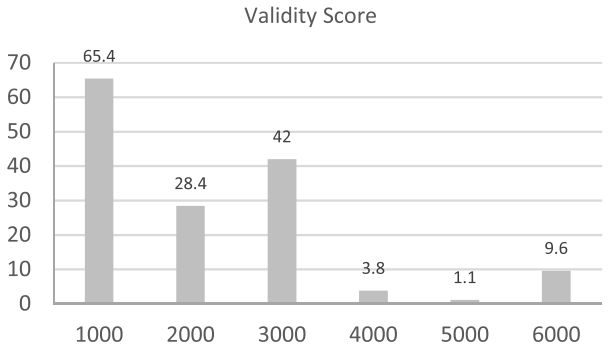


**FIGURE 8.** Validity score as a function of iteration. It is destroyed while optimizing synthesizability.

model to allow the generator to be jointly but sequentially trained by the discriminator and the reward network, as shown in Fig. 5. The reward network was optimized in this case. The best validity score obtained was 99.6, which outperformed the Vanilla GAN and the original MolGAN methods used in Experiments 1 and 2, respectively. We believe that optimizing the reward network increases the validity score further, potentially achieving state-of-the-art results, but compromises the training time for the reward network.

(Experiment 4) Based on the model introduced in Scenario 3, where the reward network was not optimized, this model was trained in a loop, as shown in Fig. 5, with the added hyperparameters shown in Table 2. The best validity score was 15.1, which suggests that the reward network was optimized.

(Experiments 5 and 6) These two experiments used the training loop described in Fig. 5 and the hyperparameters in Table 2. They revealed the phenomena that the validity score could not maintain while optimizing druglikeness and synthesizability, as shown in Fig. 7 and Fig. 8, respectively. It is observed in Fig. 7 that the validity was maintained for only the first 3000 iterations before a cliff-like drop occurred

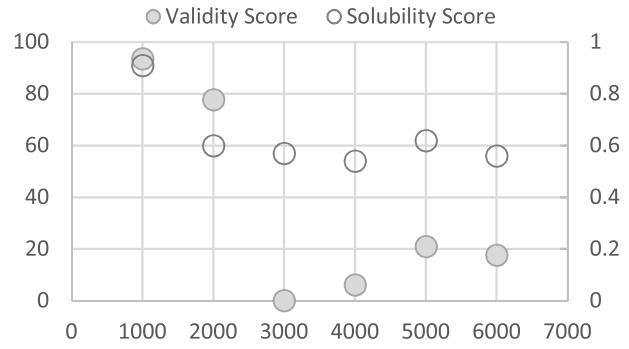


**FIGURE 9.** Validity and solubility score using the proposed method as a function of iteration.

in the scores. This drop was due to the mutual interference and destruction in the model that occurred during the optimization of the other chemical property scores. Similarly, as seen in Fig. 8, the model was trained for a different target, i.e., synthesizability, and the validity score collapsing problem occurred again. To solve the validity score collapsing problem seen in Experiments 5 and 6, we propose using the model from the sixth scenario.

(Experiment 7) The validity improvement method shown in Fig. 6 was tested here. The main target was to optimize solubility and maintain a high validity score. In this model, two reward networks were trained, one for the validity score and the other for the solubility score. The results are shown in Fig. 9. The best solubility score was 0.91, while the validity score was 93.6. The way we solved the validity score collapse problem was explained in the previous section.

## V. CONCLUSION

In this paper, potential ways to improve the validity score of the generated molecules based on MolGAN were investigated. It was found that Vanilla GAN can produce molecules with a moderate validity score of 73.8. This score was obtained by only using valid molecules in the training set and implicitly optimizing the validity score. Then, a reward network was added to the Vanilla GAN in a reinforcement learning manner. It was forced to be optimized at every iteration rather than being pretrained to explicitly improve the validity score to 99.6. It outperformed Vanilla GAN (73.8) and the original MolGAN method (90.0). Multiple aspects, including loss function, hyperparameters, and training sequences, were evaluated and optimized to further raise the validity score. We also investigated the validity score collapsing problem that occurs during the optimization of other chemical property scores. It was found that two separate reward networks in parallel could postpone the validity score collapsing problem. A good validity score of 93.6 and a moderate chemical property score of 0.91 were obtained.

## APPENDIX

See Tables 1 and 2.

## REFERENCES

[1] D. C. Elton, Z. Boukouvalas, M. D. Fuge, and P. W. Chung, "Deep learning for molecular design—A review of the state of the art," *Mol. Syst. Des. Eng.*, vol. 4, no. 4, pp. 828–849, 2019.

[2] V. D. Mouchlis, A. Afantitis, A. Serra, M. Fratello, A. G. Papadiamantis, V. Aidinis, I. Lynch, D. Greco, and G. Melagraki, "Advances in de novo drug design: From conventional to machine learning methods," *Int. J. Mol. Sci.*, vol. 22, no. 4, p. 1676, Feb. 2021.

[3] D. Xue, Y. Gong, Z. Yang, G. Chuai, S. Qu, A. Shen, J. Yu, and Q. Liu, "Advances and challenges in deep generative models for de novo molecule generation," *WIREs Comput. Mol. Sci.*, vol. 9, no. 3, p. e1395, May 2019.

[4] M. Popova, M. Shvets, J. Oliva, and O. Isayev, "MolecularRNN: Generating realistic molecular graphs with optimized properties," 2019, *arXiv:1905.13372*.

[5] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *J. Chem. Inf. Comput. Sci.*, vol. 28, no. 1, pp. 31–36, Feb. 1988.

[6] N. O'Boyle and A. Dalke, "DeepSMILES: An adaptation of SMILES for use in machine-learning of chemical structures," in *ChemRxiv*. Cambridge, U.K.: Cambridge Open Engage, 2018.

[7] Y. Kwon, D. Lee, Y. S. Choi, K. Shin, and S. Kang, "Compressed graph representation for scalable molecular graph generation," *J. Cheminformatics*, vol. 12, no. 1, pp. 1–8, 2020.

[8] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, "GraphAF: A flow-based autoregressive model for molecular graph generation," 2020, *arXiv:2001.09382*.

[9] P. Bongini, M. Bianchini, and F. Scarselli, "Molecular generative graph neural networks for drug discovery," *Neurocomputing*, vol. 450, pp. 242–252, Aug. 2021.

[10] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2323–2332.

[11] F. Grisoni, M. Moret, R. Lingwood, and G. Schneider, "Bidirectional molecule generation with recurrent neural networks," *J. Chem. Inf. Model.*, vol. 60, no. 3, pp. 1175–1183, Mar. 2020.

[12] M. Langevin, H. Minoux, M. Levesque, and M. Bianciotto, "Scaffold-constrained molecular generation," *J. Chem. Inf. Model.*, vol. 60, no. 12, pp. 5637–5646, Dec. 2020.

[13] C. Ma and X. Zhang, "GF-VAE: A flow-based variational autoencoder for molecule generation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2021, pp. 1181–1190.

[14] J. Mitton, H. M. Senn, K. Wynne, and R. Murray-Smith, "A graph VAE and graph transformer approach to generating molecular graphs," 2021, *arXiv:2104.04345*.

[15] O. Prykhodko, S. V. Johansson, P.-C. Kotsias, J. Arus-Pous, E. J. Bjerrum, O. Engkvist, and H. Chen, "A de novo molecular generation method using latent vector based generative adversarial network," *J. Cheminformatics*, vol. 11, no. 1, pp. 1–3, Dec. 2019.

[16] Y. J. Lee, H. Kahng, and S. B. Kim, "Generative adversarial networks for de novo molecular design," *Mol. Informat.*, vol. 40, no. 10, Oct. 2021, Art. no. 2100045.

[17] Y. Bian, J. Wang, J. J. Jun, and X.-Q. Xie, "Deep convolutional generative adversarial network (dcGAN) models for screening and design of small molecules targeting cannabinoid receptors," *Mol. Pharmaceutics*, vol. 16, no. 11, pp. 4451–4460, Nov. 2019.

[18] E. Lin, C.-H. Lin, and H.-Y. Lane, "Relevant applications of generative adversarial networks in drug design and discovery: Molecular de novo design, dimensionality reduction, and de novo peptide and protein design," *Molecules*, vol. 25, no. 14, p. 3250, Jul. 2020.

[19] A. Ramesh, A. S. Rao, S. Moudgalya, and K. S. Srinivas, "GAN based approach for drug design," in *Proc. 20th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2021, pp. 825–828.

[20] I. Jacobs and M. Maragoudakis, "De novo drug design using artificial intelligence applied on SARS-CoV-2 viral proteins ASYNT-GAN," *BioChem*, vol. 1, no. 1, pp. 36–48, Apr. 2021.

[21] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," 2018, *arXiv:1805.11973*.

[22] Y. Tsujimoto, S. Hiwa, Y. Nakamura, Y. Oe, and T. Hiroyasu, "L-MolGAN: An improved implicit generative model for large molecular graphs," in *ChemRxiv*. Cambridge, U.K.: Cambridge Open Engage, 2021.

[23] L. Ruddigkeit, R. van Deursen, L. C. 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, no. 11, pp. 2864–2875, Nov. 2012.
[24] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific Data*, vol. 1, no. 1, Aug. 2014, Art. no. 140022.

**JIAYI FAN** was born in Guangxi, China, in 1991. He received the B.S. degree in electrical engineering and its automation from Hunan University, Changsha, China, in 2014, and the M.S. degree in mechatronics engineering from Kyungsung University, Busan, South Korea, in 2019. He is currently a Researcher with the Department of Semiconductor Engineering, Seoul National University of Science and Technology, Seoul, South Korea.

**SEUL KI HONG** received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2009, 2011, and 2015, respectively. He was a Senior Engineer with Samsung Electronics, South Korea. He is currently an Assistant Professor with the Department of Semiconductor Engineering, Seoul National University of Science and Technology, Seoul, South Korea.

**YONGKEUN LEE** (Member, IEEE) received the B.S. degree in material engineering from Iowa State University, Ames, IA, USA in 1991, the M.S. degree in material science from Columbia University, New York City, NY, USA, in 1993, and the Ph.D. degree in materials engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1996. He was with Nanyang Technological University, Singapore, as an Assistant Professor, and a Principal Engineer with Samsung Electronics LCD Business, Giheung, South Korea. Since 2007, he has been the Dean of the Graduate School of Nano IT Design Fusion, Seoul National University of Science and Technology, Seoul, South Korea, where he is currently a Professor with the Department of Semiconductor Engineering.