TenGAN: Pure Transformer Encoders Make an Efficient Discrete GAN for De Novo Molecular Generation

Chen Li **Yoshihiro Yamanishi**
Graduate School of Informatics, Nagoya University, Chikusa, Nagoya, 464-8602, Japan
li.chen.z2@a.mail.nagoya-u.ac.jp, yamanishi@i.nagoya-u.ac.jp

Abstract

Deep generative models for *de novo* molecular generation using discrete data, such as the simplified molecular-input line-entry system (SMILES) strings, have attracted widespread attention in drug design. However, training instability often plagues generative adversarial networks (GANs), leading to problems such as mode collapse and low diversity. This study proposes a pure transformer encoder-based GAN (TenGAN) to solve these issues. The generator and discriminator of TenGAN are variants of the transformer encoders and are combined with reinforcement learning (RL) to generate molecules with the desired chemical properties. Besides, data augmentation of the variant SMILES is leveraged for the TenGAN training to learn the semantics and syntax of SMILES strings. Additionally, we introduce an enhanced variant of TenGAN, named Ten(W)GAN, which incorporates mini-batch discrimination and Wasserstein GAN to improve the ability to generate molecules. The experimental results and ablation studies on the QM9 and ZINC datasets showed that the proposed models generated highly valid and novel molecules with the desired chemical properties in a computationally efficient manner.

1 Introduction

The primary goal of *de novo* molecular generation is to find novel molecules with the desired chemical properties (Schneider and Fechner, 2005). However, finding such molecules in a vast chemical space without any prior knowledge is extremely difficult. In recent years, deep learning-based methods have been used in biomedicine and drug discovery to significantly reduce the time and cost of drug design. Generally, deep learning-based molecular generation can be divided into the following two categories: string-based methods (Grisoni et al., 2020; Li and Yamanishi, 2023) and graph-based methods (Merkwirth and Lengauer, 2005; Jin et al., 2018b; Xia et al., 2019). For the former, molecules can be represented as strings based on the simplified molecular-input line-entry system (SMILES), which makes it possible to use deep generative models of the natural language process (NLP) for calculation. While the serialized representations of molecules in SMILES strings are more concise than the graph representations, they contain less information (David et al., 2020). Additionally, unlike NLP, the semantic and syntax rules for SMILES strings are not strict, leading to various representations for a molecule depending on the order in which atoms are traversed (Li et al., 2022). Therefore, generating molecules with SMILES strings poses a significant challenge.

Generative adversarial networks (GANs) (Goodfellow et al., 2014) can generate highly realistic content, and they have recently been used in various NLP applications, such as machine translation (Wu et al., 2018) and question answering (Jiang et al., 2020). Unfortunately, GANs are mainly used to generate continuous data. There are two main drawbacks of using GAN for generating discrete data such as SMILES strings. First, the gradient of the loss function in the discriminator of a GAN helps guide the training of the generator (i.e., making slight changes to the parameters so that the generator can generate realistic data). The discriminator cannot effectively guide the generator in the case of discrete data because there are no corresponding labels in the dictionary space (Huszár, 2015). Second, the discriminator cannot evaluate an incomplete string. Different intermediate tokens during SMILES string generation may lead to different scoring results for the discriminator (Yu et al., 2017).

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

A reinforcement learning (RL)-based GAN called ORGAN (Guimaraes et al., 2017) has been proposed to solve the inability of GANs to generate molecules from discrete SMILES strings. In ORGAN, the generator is a variant of recurrent neural networks (RNNs), known as long short-term memory (LSTM), which produces SMILES-like data to confuse the discriminator. The discriminator is a convolutional neural network (CNN), which tries to differentiate samples correctly. RL is used to improve the scores for desired molecular properties. Intuitively, the scores of desired properties can be increased by shifting the reward distribution of molecules generated during each iteration to the right in a stepwise manner. A larger shift to the right increases the score of the molecular properties while decreasing the amount of data in the training set. Therefore, after multiple iterations, the diversity of the molecules decreases, and the repetition rate of the generated molecules increases. The prediction of RNNs at the current time step based on previous time steps limits its application to large molecular corpora on GPUs (Li et al., 2019). Furthermore, the convolution operator of CNNs has a local receptive field. In other words, a CNN can handle long-range dependencies only when a sufficient number of layers is set (Jiang et al., 2021). However, the difficulty of optimization increases with the number of layers.

The transformer (Vaswani et al., 2017) abandons the recurrent structure of RNNs and only uses a self-attention mechanism to capture features in sequences. The self-attention of a transformer has a global receptive field that can capture long-range dependencies, which inspired us to create a GAN that only uses transformers. Transformers for goal-directed tasks can encode a given starting molecule and decode it into a new molecule (He et al., 2021). However, their suitability in distribution learning tasks for GANs (e.g., generating molecular distributions from scratch and evaluating their quality) is unclear, particularly with discrete data. In this study, we propose a pure transformer encoder-based GAN, called TenGAN, to solve the above issues. The generator and discriminator of TenGAN are variants of the transformer encoders. Atoms can access each other at each layer of the encoder to capture the complex semantic and syntactic rules of SMILES strings. To sufficiently train the generator, variant SMILES (Arús-Pous et al., 2019) is used in the pretraining phase. We also propose an enhanced algorithm that incorporates the mini-batch discrimination (Salimans et al., 2016) and Wasserstein GAN (WGAN) (Gulrajani et al., 2017) to stabilize GAN training. To our knowledge, we are the first to generate molecules with chemical properties from SMILES strings using only transformer encoders. The main contributions of this study are as follows:

- **Novel design architecture:** To solve the problems of expansion and parallelization of the sequential network structure and the problem of long-range dependencies of the local receptive field of CNNs, we propose a GAN based only on the transformer encoders to generate SMILES strings.

- **Alleviation of training instability:** Variant SMILES can help the generator sufficiently learn the semantic and syntactic features. Mini-batch discrimination constructs a diversity measure in a mini-batch, and the WGAN solves the training imbalance between the generator and discriminator.

- **Performance improvement:** The experimental results demonstrated the usefulness of the proposed models for generating molecules with different properties. Ablation studies also demonstrated the effectiveness of the proposed techniques.

2 Related Work

2.1 Variational Autoencoders-based Models

Variational autoencoders (VAEs) (Kingma and Welling, 2014) have been used for molecular generation (Kusner et al., 2017; Jin et al., 2018a). However, traditional VAEs cannot generate valid molecules without constraints because SMILES and graphical representations of molecules are discrete data and their structures mainly contain semantic and syntax information. A parse tree with context-free grammar has been constructed to represent discrete data in producing valid molecules from SMILES strings (Kusner et al., 2017). By using context-free grammar to describe a syntactically valid SMILES string set, a grammar-based model for parsing and validation can be used. Two tasks of molecular generation are discussed: character-based VAE (CharVAE) and grammar-based VAE (GramVAE). The former samples any possible character at every step without any stack and masking operation, whereas the latter is constrained by a grammar tree to select syntactically valid SMILES strings. However, the grammar cannot fully capture the chemical validity. JTVAE is a junction tree-based VAE that generates molecules from molecular graphs instead of linear SMILES strings (Jin et al., 2018a). JTVAE can generate molecules from graphs in two steps: a tree-structured scaffold has been generated on a chemical substructure and the tree-structured scaffolds were combined into a molecule using a graph message-passing network (Dai et al., 2016). JTVAE guaranteed the validity of the molecules generated in each step.

2.2 Flow-based Autoregressive Models

GraphAF is a flow-based autoregressive model that generates molecules from graph structures (Shi et al., 2020). It can be used not only to produce valid molecular structures but also to optimize chemical properties. GraphAF can dynamically generate edges and nodes based on the existing subgraph structure during sampling. The sequential generation of GraphAF allows operations such as chemical knowledge and valence check to be imported in each generating step to ensure the validity of the generated molecules, which is similar to the graph convolutional policy network (You et al., 2018) and molecular recurrent neural network (Popova et al., 2019). Additionally, a feedforward neural network has been defined to calculate the likelihood of data from the molecular graphs to the base distribution in parallel. Experiments have demonstrated the effectiveness of GraphAF. GraphNVP (Madhawa et al., 2019) stands out as an invertible, normalizing flow-based model tailored for generating molecular graphs. It adeptly dissects the generation process into two key components: the creation of an adjacency tensor and the generation of node attributes. MoFlow (Zang and Wang, 2020) uses graphical conditional flow to generate sequences of atoms for the given chemical bonds. Finally, the generated atoms and chemical bonds are combined to obtain molecular graphs.

2.3 Transformer-based Models

Unlike our goal of generating molecules from scratch, using a transformer to optimize the desired chemical properties from a given starting molecule is called a goal-directed task (He et al., 2021). Generally, a transformer must match the input and target output for training. Matched molecular pair analysis (Tyrchan and Evertsson, 2017) was used to generate paired SMILES strings for the transformer. The two similar strings were then used as an input and label for the transformer. The desired properties were input into the transformer as an additional condition with the input molecule, allowing the transformer to design molecules with chemical properties from the given starting molecules. A geometry-aware transformer was proposed to predict the molecular properties, and the long-range interaction problem of the graph neural networks was solved (Kwak et al., 2023). This problem has always existed in CNNs. For molecular generation, a model that integrates transformer and VAE was proposed (Dollar et al., 2021), with the encoder and decoder of a transformer serving as the two components of the VAE. The goal-directed model was converted into a distributed learning model, and some new molecules were sampled from the latent space of the VAE. However, without an adversarial training mechanism, the VAE generates unnatural molecules, resulting in low validity. Furthermore, the objective function determined that the decoder cannot perform property optimization when generating molecules.

Many transformer-based models have achieved SOTA performances because of the success of transformers in machine translation and other fields. In recent years, many researchers have focused on using only transformer encoders or transformer decoders to solve problems. The most well-known models are bidirectional encoder representations from transformers (BERT) (Kenton and Toutanova, 2019) and generative pretrained transformers (GPT) (Floridi and Chiriatti, 2020). However, no previous work has considered integrating a transformer and GAN to generate molecules, particularly when using discrete SMILES strings. In this study, we propose two property-optimized GANs called TenGAN and Ten(W)GAN, which use only the transformer-encoder variants to generate molecules with the chemical properties.

3 Models

3.1 TenGAN

Data augmentation with variant SMILES. Generally, the atoms of a molecule are traversed according to a set of rules to ensure that the generated molecular string is standardized and unique. However, the same molecule can be represented by various SMILES strings through different traversal orders of the molecular graph, which are called variant SMILES (Li et al., 2022). For example, the canonical SMILES string "CCNCc1cncnc1" can be represented by the variant strings "c1ncncc1CNCC" and "n1cncc(CNCC)c1." Using a variant string instead of a canonical string makes the model learn more syntactic and semantic features. A sufficiently trained generator can capture additional features of SMILES strings. Therefore, variant SMILES are produced to pretrain the generator. Note that the generated strings are converted into the canonical strings after pretraining for evaluation. An example of a variant SMILES production is provided in Appendix A.

Transformer encoder as the generator. Figure 1 demonstrates the architecture of TenGAN. In contrast to traditional transformers that use an encoder with a multi-head attention layer to extract features from an input sequence and decode with a decoder, TenGAN generates SMILES strings from scratch/noise. Therefore, we discard the transformer decoder and use only the encoder variant with a masked multi-head attention layer as the generator to generate SMILES strings. Formally, let $< bos >$ and $< eos >$ denote

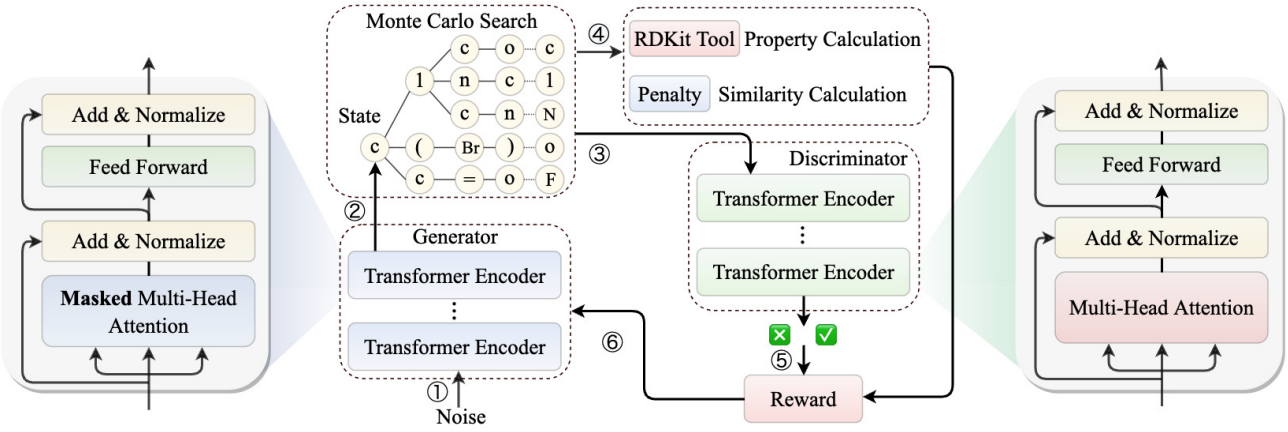


Figure 1: Overview of TenGAN. TenGAN comprises two main components: a generator G_θ and a discriminator D_ϕ, where θ and ϕ represent weight parameters. The generator and discriminator consist of only the transformer-encoder variants. The generator first produces SMILES substrings from scratch/noise for new molecules (①). Then, the generated substrings are complemented using the Monte Carlo (MC) search (②). Next, the complete strings are shuffled with the training data as the input to the discriminator (③). The discriminator uses the entire SMILES string as the input to distinguish between the generated and training strings in TenGAN or to regress the generated distribution to the training distribution in Ten(W)GAN. The current input is penalized according to the repetition rate to avoid producing the same molecule, and the RDKit tool is used to calculate the value of their chemical properties (④). Subsequently, the discriminator takes the probability of the input SMILES being real as part of the reward (⑤). Finally, the probabilities, penalties, and property scores are jointly used as rewards to update the parameters of the generator with RL (⑥).

the first and last tokens of a SMILES string, respectively. For a SMILES string $X_{1:T} = [x_1, x_2, \cdots, x_T]$ of string length T, x_i refers to the i-th token. We define $[< bos >, x_1, \cdots, x_T]$ and $[x_1, \cdots, x_T, < eos >]$ as pretraining input and output, and mask the multi-head attention layer of the traditional transformer encoder to avoid exposing the generator to future information. Only the starting token $< bos >$ is given in the generation phase after pretraining, and the next token is sampled from the current one. Details of the calculation are provided in Appendix B.

Transformer encoder as the discriminator. The discriminator is a binary classifier that uses the transformer encoders to determine whether the string is from the generator or the training set of SMILES strings. The discriminator uses a padding mask to ensure that the transformer encoders do not pay any attention to padding tokens. After obtaining the feature vector of the attention scores of each atom in the SMILES string to others, these feature vectors are summed, averaged, and input to the last fully connected layer to calculate the probability. For additional details, refer to Appendix B.

Reinforcement learning. Because SMILES represents molecules as discrete data, GANs cannot differentiate samples directly. RL is an alternative method that can be used to solve the non-differentiable problem. The objective of the generator G_θ is to produce a SMILES string that started from noise and maximizes the reward of its expected chemical properties. Generally, let $Y_{1:T} = \{y_1, y_2, \cdots, y_T\}$ and s_0 be the generated SMILES string and initial state, respectively. The objective function of the generator is given by

$$J(\theta) = \sum_{Y_{1:T}} G_\theta(Y_{1:T}|s_0) \cdot R^{G_\theta}(Y_{1:T-1}, y_T), \quad (1)$$

where R^{G_θ} denotes the action-value function that accumulates the reward at state $Y_{1:T-1}$, takes action y_T, and follows policy G_θ. R^{G_θ} mainly comprises three parts: the probability that the generated SMILES string is real according to the discriminator, the property scores of the generated SMILES strings as calculated by RDKit tool (Landrum, 2013), and the penalty for generating nonunique SMILES strings.

$$\begin{aligned} R^{G_\theta}(Y_{1:T-1}, y_T) =& \lambda D_\phi(Y_{1:T}) + (1-\lambda)\cdot \\ & O(Y_{1:T}) \cdot P(Y_{1:T}) - b(Y_{1:T}), \end{aligned} \quad (2)$$

Here, O and P are the property scores and the ratio of the number of unique SMILES strings to the product of the number of strings and the number of repeated strings, respectively. b is defined with a baseline value to provide a molecule with a large reward. For simplicity, it is computed by the average of the reward. λ is a

hyperparameter that is used to balance the strengths of GAN and RL. Particularly, TenGAN is fully trained by RL when $\lambda = 0$ and by GAN when $\lambda = 1$. D_ϕ, O, and P can only score the complete SMILES string of $Y_{1:T}$ and cannot provide a reward for substrings. MC search was used for sampling to evaluate the action-value function for an incomplete string. Let N be the number of MC searches. An N-time search can generate N samples by using $Y_{1:t}$:

$$Y^i_{1:T} \in \mathrm{MC}^{G_\theta}(Y_{1:t}, N), \text{and } i \in [1, N]. \tag{3}$$

The probabilities of N samples being real are calculated using D_ϕ, and the average value is used as the final calculation result of the intermediate state of $Y_{1:t}$:

$$\begin{aligned} R^{G_\theta}(Y_{1:t-1}, y_t) = \frac{1}{N}\sum_{n=1}^{N} \lambda D_\phi(Y^n_{1:T}) + (1-\lambda)\cdot \\ O(Y^n_{1:T}) \cdot P(Y^n_{1:T}) - b(Y^n_{1:T}), \quad \text{if} \quad t < T, \end{aligned} \tag{4}$$

$$\begin{aligned} R^{G_\theta}(Y_{1:t-1}, y_t) = \lambda D_\phi(Y_{1:t}) + (1-\lambda)\cdot O(Y_{1:t}) \cdot \\ P(Y_{1:t}) - b(Y_{1:t}), \quad \text{if} \quad t = T. \end{aligned} \tag{5}$$

Then, the gradient of the objective function of Eq.(1) can be rewritten by

$$\nabla J(\theta) \simeq \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t}[R^{G_\theta}(Y_{1:t-1}, y_t) \cdot \nabla \log G_\theta(y_t|Y_{1:t-1})]. \tag{6}$$

Finally, the generator G_θ is updated to $G_{\theta'}$ using the gradient $\nabla J(\theta)$ by $\theta' \leftarrow \theta + \nabla J(\theta)$.

Algorithm 1: Procedures for TenGAN

Data: a SMILES dataset $S_{real} = \{X_{1:T}\}$
Initialization: generator G_θ, discriminator D_ϕ
`// Pretrain the generator`
for $i = 1$ ***to*** *g-epochs* **do**
  Update θ with the MLE
end
Generate a fake dataset $S_{fake} = \{Y_{1:T}\}$
`// Pretrain the discriminator`
for $i = 1$ ***to*** *d-epochs* **do**
  Update ϕ with the cross-entropy or Wasserstein distance (③ in Fig. 1)
end
`// Adversarial training`
for $i = 1$ ***to*** *epochs* **do**
  `// Train the generator`
  for $j = 1$ ***to*** *g-steps* **do**
    Input noise (① in Fig. 1) and generate $\{Y_{1:t}\}$
    Complete $\{Y_{1:t}\}$ and produce S_{fake} (② in Fig. 1)
    Calculate $R^{G_\theta}(Y_{1:t-1}, y_t)$ by Eq. (4) and Eq. (5) (④ and ⑤ in Fig. 1) Update θ by Eq. (6) (⑥ in Fig. 1)
  end
  `// Train the discriminator`
  for $k = 1$ ***to*** *d-steps* **do**
    Update ϕ of the discriminator (③ in Fig. 1)
  end
end

3.2 Ten(W)GAN

Mini-batch discrimination. In RL, the SMILES strings used for training decreases as the reward increases, reducing the diversity of the generated data. Additionally, the discriminator discriminates each input string independently, and there is no correlation between the gradients. Therefore, the discriminator cannot guide the generator to generate diverse molecules. Mini-batch discrimination is used to mitigate these issues. Unlike the original approach, we consider a variant mini-batch discrimination, which does not require new parameter learning. Formally, the output of the last layer is defined as $\mathbf{M} \in \mathbb{R}^{B \times d_{model}}$, where B is the mini-batch size. We first calculate the standard deviation of the feature matrix $\mathbf{M}$ in the d_{model} dimension. Then, we concatenate the mean of the estimates to $\mathbf{M}$. Note that the mini-batch discrimination mainly acts on the last layer of discriminator.

Wasserstein GAN. To further mitigate training instability, we consider the Ten(W)GAN, a WGAN based on the mini-batch discrimination. Ten(W)GAN, in contrast to TenGAN's binary classification, uses the WGAN in the discriminator. The WGAN uses the Wasserstein distance as the cost function, which has a smoother gradient than a GAN.

Algorithm 1 summarizes the procedures of TenGAN and Ten(W)GAN. In the pretraining phase, maximum likelihood estimation (MLE) is used to pretrain the generator with the SMILES dataset S_{real}. Then, the generated dataset S_{fake} is combined with S_{real} to pretrain the discriminator. The real dataset has the same number of SMILES strings as the fake dataset to avoid the problem of imbalanced datasets. In the training phase, the generator and discriminator are trained alternately. The gradients of the generator are updated by the MC policy gradient method.

4 Experiments

4.1 Experimental Setup

Datasets. Two randomly selected subsets of 5000 and 10000 SMILES strings from QM9 (Ramakrishnan et al., 2014) and ZINC (Irwin et al., 2012) were used

Table 1: Statistics for the QM9 and ZINC datasets.

Dataset	LEN	MIN	MAX	QED	SA	logP
QM9	15	5	31	0.48	0.23	0.29
ZINC	38	22	69	0.79	0.76	0.63

* LEN, MIN, and MAX indicate the average, minimum, and maximum length of the SMILES strings.

as the training datasets. All strings in both datasets contained up to nine heavy atoms, such as carbon (C), oxygen (O), nitrogen (N), and fluorine (F).

Hyperparameters. Both the QM9 and ZINC datasets had a vocabulary size of 40 characters and the molecules had a maximum length of 60 and 70 characters, respectively. In the training of the two datasets, each iteration generated 5000 and 10000 samples, respectively, with a batch size of 64. The generator had four encoder layers. Each encoder had four attention heads, and the value and feedforward layers had dimensions of 128 and 1024, respectively. The dropout probability (Srivastava et al., 2014) was 0.1. The discriminator had four encoder layers, each with five attention heads. The value and feedforward layers had dimensions of 100 and 200, respectively. The dropout probability was set to 0.25. The generator and discriminator were trained by using the Adam optimizer (Kingma and Ba, 2015). The learning rates of the generator and discriminator in the pretraining were 8e−4 and 8e−5, respectively. The learning rate of the generator in the training was set to 8e−5. Additionally, unless other specified, λ was set to 0.5 and the rollout times N for QM9 and ZINC datasets were set to 16 and 8 respectively. The generator was pretrained for 150 epochs using MLE, and the discriminator was pretrained for 10 epochs. Up to 100 adversarial training epochs were performed. Note that all experiments were performed on GPU using CUDA. Our source code is publicly available on GitHub [1].

4.2 Metrics

Evaluation measures. To evaluate the performance and compare those to baseline models, we used the following statistics (Simonovsky and Komodakis, 2018). *Validity* is the percentage of chemically valid molecules to all generated molecules. Note that the validity was checked by the RDKit tool. *Uniqueness* is the percentage of non-repeated molecules to the number of valid molecules. *Novelty* is the percentage of valid molecules not included in the training dataset for all unique molecules. *Diversity* is the average Tanimoto distance (Rogers and Tanimoto, 1960) between Morgan fingerprints (Morgan, 1965) of any two molecules with a radius of 4 and 2,048 bits. We calculated the execution time in the GPU environment.

Optimized properties. For controlled generation and optimization, models need to be trained to learn some chemical properties of molecules. We focused on the following chemical properties for property optimization. *Drug-likeness* is a measure of the likelihood that a molecule is a drug, which is usually expressed by the quantitative estimate of drug-likeness (QED) score (Bickerton et al., 2012). *Synthesizability* is a measure of the difficulty of synthesizing a molecule that can be described by the synthetic accessibility (SA) score (Ertl and Schuffenhauer, 2009). *Solubility* is a measure of the hydrophilicity of a molecule that is usually calculated by the log octanol-water partition coefficient (logP) (Comer and Tam, 2001). The calculations of the chemical properties are detailed in Appendix C. All the above scores are in the range of [0, 1]. A score of zero indicates an invalid molecule or a valid molecule with a score of zero for a given property. Note that a higher score indicates a better performance regarding the corresponding property. Table 1 provides some average statistics for molecules in the two datasets.

4.3 Comparison with RNN-based Methods

Table 2 and Table E.1 in the Appendix show the comparisons of TenGAN and Ten(W)GAN with Naïve RL, ORGAN, and OR(W)GAN for the QM9 and ZINC datasets, respectively. These baseline algorithms are the closest related works of training RNNs for optimizing different properties in generating molecules. The LSTM and transformer encoder (TransEn) baselines were trained by using MLE without property optimization and stopped immediately after pretraining. These results suggest that the proposed models performed better than the baselines on most measures.

For the drug-likeness in Table 2, TenGAN improved the validity and uniqueness by 11.0% and 7.6%, respectively. Ten(W)GAN showed improvements of 14.7% and 133.6%, respectively. Both TenGAN and Ten(W)GAN had the novelty scores close to 100.0%. The QED scores were improved by 3.6% and 17.6%, respectively. Additionally, TenGAN and Ten(W)GAN performed significantly better in terms of diversity. The execution times show the superiority of the transformer during training with respect to computational efficiency. Similar trends were also observed on the ZINC dataset as shown in Table E.1 in the Appendix.

Figure C.1 demonstrates examples of the top-12 molecules of Naïve RL, ORGAN, and OR(W)GAN for the QM9 dataset. For example, the molecules generated by Naïve RL, ORGAN, OR(W)GAN include

[1] Our code: `https://github.com/naruto7283/TenGAN`

Table 2: Performance evaluation on property optimizations compared to the closest related work on QM9 dataset.

Property	Algorithm	Validity	Uniqueness	Novelty	QED	SA	logP	Diversity	Time (h)
	LSTM	74.4%	98.3%	96.9%	0.48	0.23	0.29	0.93	0.50
	TransEn	89.5%	93.9%	82.7%	0.48	0.24	0.30	0.92	0.33
Drug-likeness	Naïve RL	97.0%	59.0%	**100.0%**	0.57	0.54	**0.47**	0.89	7.39
	ORGAN	88.1%	65.7%	97.9%	0.55	0.45	0.41	0.86	7.16
	OR(W)GAN	85.8%	35.7%	98.1%	0.51	**0.57**	0.29	0.89	7.31
	TenGAN	97.8%	70.7%	98.0%	0.57	0.50	0.40	**0.90**	**5.06**
	Ten(W)GAN	**98.4%**	**83.4%**	99.8%	**0.60**	0.45	0.44	0.89	5.75
Synthesizability	Naïve RL	**97.2%**	18.6%	98.2%	0.48	0.75	0.33	0.88	7.54
	ORGAN	96.5%	**32.7%**	**99.4%**	0.49	0.71	0.41	0.87	7.78
	OR(W)GAN	94.8%	23.8%	99.3%	0.50	0.64	0.34	0.86	7.81
	TenGAN	96.7%	24.2%	97.5%	**0.52**	0.71	**0.50**	**0.90**	**4.67**
	Ten(W)GAN	96.8%	21.7%	98.0%	**0.52**	**0.76**	0.48	0.87	6.73
Solubility	Naïve RL	92.8%	**63.2%**	**100.0%**	0.44	**0.59**	**0.80**	0.86	7.08
	ORGAN	93.3%	41.3%	99.5%	0.51	0.56	0.54	0.89	7.20
	OR(W)GAN	92.1%	16.9%	97.2%	0.48	0.52	0.44	0.90	7.53
	TenGAN	97.5%	61.6%	97.6%	**0.51**	0.47	0.55	**0.91**	**2.99**
	Ten(W)GAN	**98.2%**	30.1%	99.7%	**0.51**	0.57	0.64	0.90	4.37

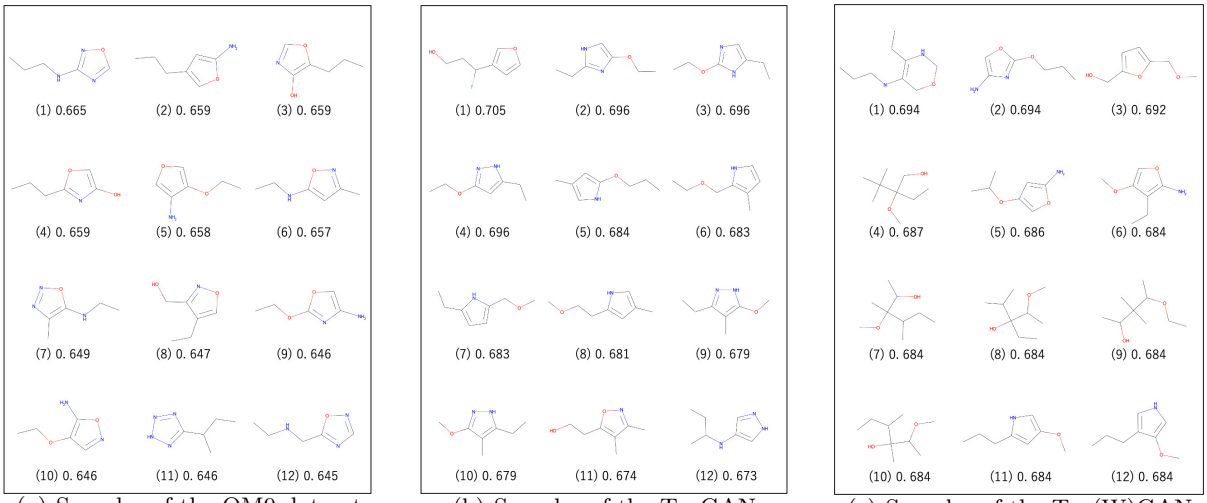
* Red boxes indicate the directly optimized properties. Bold values indicate the highest scores among different methods. The values in gray cells indicate that TenGAN / Ten(W)GAN outperformed the corresponding baselines.

unnatural substructures such as big ring structures (e.g., nine-, ten-membered rings). Figure 2 shows the top-12 molecule structures by QED scores for the QM9 dataset, TenGAN, and Ten(W)GAN. All compounds generated by TenGAN have pyrrole rings (i.e., five-member rings containing nitrogen atoms), which are in many organic compounds. Intuitively, the molecules generated by TenGAN are not similar to the QM9 training dataset, but their QED scores are much higher. Additionally, most compounds generated by Ten(W)GAN have pyrrole rings. The highest QED score in the dataset was 0.665 [Fig. 2 (a)]. Our proposed TenGAN and Ten(W)GAN improved the highest QED score to 0.705 [Fig. 2 (b)] and 0.694 [Fig. 2 (c)], respectively. Compared with the top-12 molecules generated by Naïve RL, ORGAN, and OR(W)GAN, TenGAN and Ten(W)GAN generated more stable and feasible structures for drug candidates. Similarly, Figs. E.1, E.2, and E.3 in the Appendix depict newly valid molecules with high property scores on the ZINC dataset.

In the property optimization of synthesizability and solubility, TenGAN and Ten(W)GAN also worked better than the other models. The uniqueness and diversity of the molecules decreased with the training epoch because of fundamental problems in synthesizability and solubility functions. However, in the optimization of synthesizability, TenGAN tended to generate more molecules that are easier to synthesize, such as benzene rings or carbon chains, containing only carbon to maximize the SA score. Similarly, in the optimization of solubility, TenGAN generated more oily molecules that contained long carbon chains, such as "CCCCCCCC." In contrast to mode collapse, the fundamental problems in synthesizability and solubility functions made the generated molecules more repetitive, reducing their uniqueness and diversity.

Figure 3, Appendix Fig. D.1 and Appendix Fig. E.4) show similar trajectories for property scores according to the training epochs of TenGAN and Ten(W)GAN on the QM9 and ZINC datasets. The increase in the property scores with the training epochs implies that TenGAN and Ten(W)GAN generated molecules with the desired chemical properties on the both datasets.

Figure 4, Appendix Figs.D.2, D.3, D.4, and E.5 show the property (i.e., QED, SA, and logP) distributions of molecules in the original training dataset and molecules generated by TenGAN and Ten(W)GAN on the QM9 and ZINC datasets. Our proposed models shifted the distributions of the optimized properties to the right. TenGAN successfully generated molecules with higher property scores than the original molecules in the training dataset. Ten(W)GAN outperformed TenGAN owing to mini-batch discrimination to improve the diversity of the generated molecules and the



(a) Samples of the QM9 dataset. (b) Samples of the TenGAN. (c) Samples of the Ten(W)GAN.

Figure 2: Top-12 molecule structures with the QED scores (drug-likeness) generated by the proposed models.

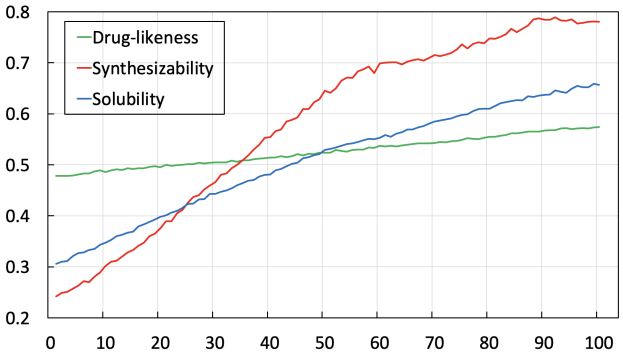


Figure 3: Changes in property scores according to the training epochs for TenGAN on the QM9 dataset.

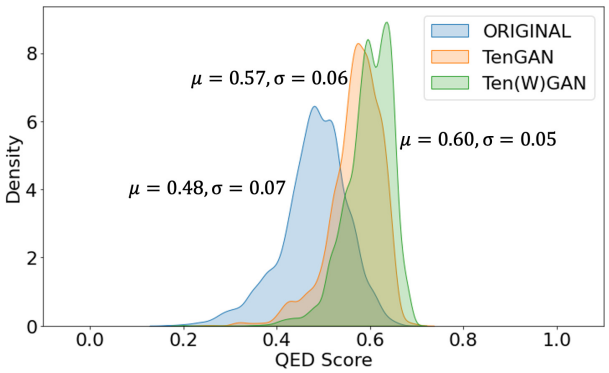


Figure 4: QED Distributions on the QM9 dataset.

Wasserstein distance in the loss function to overcome the drawbacks of the KL divergence function.

4.4 Comparison with Other Algorithms

To further validate the effectiveness of our proposed models, we also conducted comparisons with various graph-based algorithms, including JTVAE, GraphAF, GraphNVP, MoFlow, and MolGAN (De Cao and Kipf, 2018), as well as GAN-based algorithms like CharVAE, GramVAE, and TransVAE. Note that due to the richer information content in molecular graphs, graph-based models typically outperform SMILES string-based models. Specifically, discrete GANs relying on SMILES strings often exhibit training instability, leading to a reduced capacity for molecular generation.

The results of the comparison are presented in Table 3. Although JTVAE realized the highest validity (1.6% higher than Ten(W)GAN), the uniqueness was only 67.4% of Ten(W)GAN. GraphAF realized a 9.9% improvement in uniqueness but had much lower

Table 3: Comparison with graph-, VAE-, Flow-, and GAN-based algorithms on the QM9 dataset.

Algorithm	QED	Validity	Uniqueness	Novelty	Total
JTVAE	0.46	**100%**	55.7%	97.1%	54.1%
GraphAF	0.47	37.0%	91.7%	99.6%	33.8%
CharVAE	0.50	17.2%	**99.9%**	94.9%	16.3%
GramVAE	0.48	38.0%	98.8%	93.7%	35.2%
TransVAE	0.52	17.2%	25.2%	97.2%	42.1%
GraphNVP	0.58	83.0%	99.2%	-	-
MoFlow	0.44	95.0%	93.7%	89.0%	79.2%
MolGAN	0.59	99.3%	2.3%	99.7%	2.3%
TenGAN	**0.60**	97.8%	82.6%	**99.8%**	80.6%
Ten(W)GAN	**0.60**	98.4%	83.4%	**99.8%**	**81.9%**

* Total is the product of validity, uniqueness, and novelty.

Table 4: Chemical property optimization of drug-likeness (QED) in TenGAN and Ten(W)GAN, followed by a comparison of the top-k generated molecules with baselines on the QM9 dataset.

Algorithm	Top-1	Top-5	Top-10	Top-100	Top-1000
ORGAN	0.68	0.66	0.66	0.65	0.61
GraphNVP	0.63	0.62	0.62	0.59	0.42
MoFlow	0.66	0.65	0.65	0.62	0.57
TenGAN	**0.71**	**0.70**	**0.69**	0.66	0.63
Ten(W)GAN	0.69	0.69	**0.69**	**0.68**	**0.65**

validity and QED scores that were only 37.6% and 78.3% of Ten(W)GAN, respectively. Similarly, CharVAE also had much lower validity and QED scores that were only 17.5% and 83.3% of Ten(W)GAN, respectively. The validity and QED scores of GramVAE were only 38.6% and 80.0% of Ten(W)GAN, respectively. Although TransVAE combined a VAE with a transformer, it had the lowest validity and uniqueness among these models, which were only 17.5% and 30.0% of Ten(W)GAN, respectively. GraphNVP demonstrated higher uniqueness compared to both TenGAN and Ten(W)GAN, but its validity was considerably lower than that of our models. MoFlow had a lower Total score than both TenGAN and Ten(W)GAN, despite being the current SOTA model. MolGAN had high validity, but it had low uniqueness (2.3%) because of mode collapse. The results with similar trends on the ZINC dataset are demonstrated in Table E.2 in the Appendix. Overall, both TenGAN and Ten(W)GAN performed as the top two in terms of novelty, Total, and QED score, while maintaining high validity and uniqueness. These results demonstrate that our proposed discrete GANs, relying on SMILES strings and containing less molecular information compared to graph-based models, enhance the performance of molecular generation with property optimization.

Table 4 presents the chemical property optimization of drug-likeness in TenGAN and Ten(W)GAN, along with a comparison of the top-k generated molecules with baseline models (ORGAN, GraphNVP, and MoFlow) on the QM9 dataset. TenGAN demonstrated superior performance in property optimization compared to the three baseline models. Specifically, TenGAN outperformed these baseline models in Top-1, Top-5, and Top-10, showcasing improvements in QED scores by 4.4%, 6.1%, and 4.5%, compared to the second-ranked ORGAN model. Moreover, Ten(W)GAN exhibited enhancements of 4.6% and 6.5% in QED scores for Top-100 and Top-1000, respectively. The findings indicate that, compared to TenGAN, Ten(W)GAN has the ability to enhance the overall chemical property values of more generated molecules (Top-100 and Top-1000). Overall, the comprehensive evaluation across various metrics and comparisons with baseline models highlight the outstanding performance of both TenGAN and Ten(W)GAN in the realm of chemical property optimization.

4.5 Ablation Studies

In our pursuit of a thorough examination of each component within the proposed models, we present comprehensive ablation studies that scrutinize the impacts of different factors, including λ, N, and variant SMILES, across both TenGAN and Ten(W)GAN.

Tables F.1, F.2, and F.3 in the Appendix showcase the results. The findings in Table F.1 illustrate that λ effectively balances the trade-off between GAN and RL for property optimization. Additionally, Table F.2 demonstrates that increasing the rollout times N stabilizes validity, uniqueness, and novelty, albeit with an increase in computational time. Lastly, the results in Table F.3 validate the effectiveness of the variant SMILES in enhancing chemical properties, as well as improving validity, uniqueness, and novelty of the molecular structures generated by the models.

5 Conclusion

We proposed TenGAN and Ten(W)GAN, which are based only on transformer encoders to generate molecules and optimize the desired chemical properties. Variant SMILES, mini-batch discrimination, and WGAN were leveraged to mitigate the problem of training instability. The experimental results demonstrated that the proposed models performed better than RNN-, graph-, and VAE-based methods. Additionally, the evaluation results of ablation studies validated the effectiveness of the proposed techniques.

A major limitation of this study is that MC search requires a trade-off between performance stability and time consumption, as shown in Section F. A small number of searches may result in unstable training, whereas a large number of searches increases the time cost. We will consider this limitation and address it in future work by employing a solution such as the soft actor-critic (Haarnoja et al., 2018).

Acknowledgements

This research was supported by the International Research Fellow of Japan Society for the Promotion of Science (Postdoctoral Fellowships for Research in Japan [Standard]), AMED under Grant Number JP22nk0101111, and JSPS KAKENHI [grant numbers 20H05797, 21K18327].

References

Arús-Pous, J., Johansson, S. V., Prykhodko, O., Bjerrum, E. J., Tyrchan, C., Reymond, J.-L., Chen, H., and Engkvist, O. (2019). Randomized SMILES strings improve the quality of molecular generative models. *Journal of Cheminformatics*, 11(1):1–13.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. (2012). Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98.

Comer, J. and Tam, K. (2001). Lipophilicity profiles: Theory and measurement. *Pharmacokinetic Optimization in Drug Research: Biological, Physicochemical and Computational Strategies*, pages 275–304.

Dai, H., Dai, B., and Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pages 2702–2711.

David, L., Thakkar, A., Mercado, R., and Engkvist, O. (2020). Molecular representations in AI-driven drug discovery: A review and practical guide. *Journal of Cheminformatics*, 12(1):1–22.

De Cao, N. and Kipf, T. (2018). MolGAN: An implicit generative model for small molecular graphs. In *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Dollar, O., Joshi, N., Beck, D. A., and Pfaendtner, J. (2021). Attention-based generative models for de novo molecular design. *Chemical Science*.

Ertl, P. and Schuffenhauer, A. (2009). Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):1–11.

Floridi, L. and Chiriatti, M. (2020). GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

Grisoni, F., Moret, M., Lingwood, R., and Schneider, G. (2020). Bidirectional molecule generation with recurrent neural networks. *Journal of Chemical Information and Modeling*, 60(3):1175–1183.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. (2017). Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *ArXiv Preprint ArXiv:1705.10843*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870.

He, J., You, H., Sandström, E., Nittinger, E., Bjerrum, E. J., Tyrchan, C., Czechtizky, W., and Engkvist, O. (2021). Molecular optimization by capturing chemist's intuition using deep neural networks. *Journal of Cheminformatics*, 13(1):1–17.

Huszár, F. (2015). How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *ArXiv Preprint ArXiv:1511.05101*.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. (2012). ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768.

Jiang, H., Misra, I., Rohrbach, M., Learned-Miller, E., and Chen, X. (2020). In defense of grid features for visual question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10267–10276.

Jiang, Y., Chang, S., and Wang, Z. (2021). TransGAN: Two pure transformers can make one strong GAN, and that can scale up. In *Advances in Neural Information Processing Systems*.

Jin, W., Barzilay, R., and Jaakkola, T. (2018a). Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. (2018b). Learning multimodal graph-to-graph translation for molecular optimization. In *International Conference on Learning Representations*.

Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 24171–4186.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954.

Kwak, B., Park, J., Kang, T., Jo, J., Lee, B., and Yoon, S. (2023). GeoT: A geometry-aware transformer for reliable molecular property prediction and chemically interpretable representation learning. *ACS Omega*, 8(42):39759–39769.

Landrum, G. (2013). RDKit documentation. *Release*, 1(1-79):4.

Li, C., Yamanaka, C., Kaitoh, K., and Yamanishi, Y. (2022). Transformer-based objective-reinforced generative adversarial network to generate desired molecules. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 3884–3890.

Li, C. and Yamanishi, Y. (2023). SpotGAN: A reverse-transformer GAN generates scaffold-constrained molecules with property optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 323–338.

Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. (2019). Neural speech synthesis with transformer network. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713.

Madhawa, K., Ishiguro, K., Nakago, K., and Abe, M. (2019). GraphNVP: An invertible flow model for generating molecular graphs. *ArXiv Preprint ArXiv:1905.11600*.

Merkwirth, C. and Lengauer, T. (2005). Automatic generation of complementary descriptors with molecular graph networks. *Journal of Chemical Information and Modeling*, 45(5):1159–1168.

Morgan, H. L. (1965). The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113.

Popova, M., Shvets, M., Oliva, J., and Isayev, O. (2019). MolecularRNN: Generating realistic molecular graphs with optimized properties. *ArXiv Preprint ArXiv:1905.13372*.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):1–7.

Rogers, D. J. and Tanimoto, T. T. (1960). A computer program for classifying plants. *Science*, 132(3434):1115–1118.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, volume 29, pages 2234–2242.

Schneider, G. and Fechner, U. (2005). Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. (2020). GraphAF: A flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*.

Simonovsky, M. and Komodakis, N. (2018). GraphVAE: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Tyrchan, C. and Evertsson, E. (2017). Matched molecular pair analysis in short: Algorithms, applications and limitations. *Computational and Structural Biotechnology Journal*, 15:86–90.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wu, L., Xia, Y., Tian, F., Zhao, L., Qin, T., Lai, J., and Liu, T.-Y. (2018). Adversarial neural machine translation. In *Asian Conference on Machine Learning*, pages 534–549.

Xia, X., Hu, J., Wang, Y., Zhang, L., and Liu, Z. (2019). Graph-based generative models for de novo drug design. *Drug Discovery Today: Technologies*, 32:45–53.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, volume 31.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence*, volume 31.

Zang, C. and Wang, F. (2020). MoFlow: An invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626.

Supplementary Material

TenGAN: Pure Transformer Encoders Make an Efficient Discrete GAN for De Novo Molecular Generation

A Variant SMILES

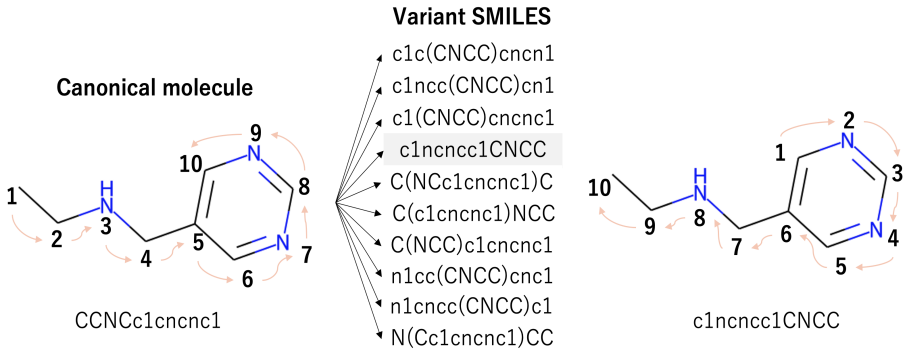


Figure A.1: Example of producing variant SMILES strings.

Figure A.1 demonstrates an example of producing variant SMILES strings. Generally, a molecular graph has a canonical SMILES string (left figure). The numbers and arrows in the figure denote the traversal order of the atoms. Unlike NLP, the canonical molecule has various SMILES representations (middle figure) according to different traversal orders (right figure) called variant SMILES. However, the same molecular graph can be represented using these ten variant SMILES strings. Therefore, variant SMILES strings can be produced to improve the pretraining of the generator and prevent it from learning only a single semantic and syntactic feature. Several representations of the same molecule can be used to fully train the generator. This also alleviates the problem of mode collapse caused by a "perfect discriminator."

B Transformer Encoder

The input representations consist of three parts: the query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$, and value matrix $\mathbf{V}$. The self-attention sublayer utilizing $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ is expressed as follows:

$$\text{self-attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{model}}})\mathbf{V}, \tag{7}$$

where d_{model} denotes the dimension of the transformer encoder. Usually, the self-attention can be refined into multi-head self-attention so that the information of different representation subspaces from different positions can be jointly considered:

$$\text{multi-head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat}(\text{head}_1, \cdots, \text{head}_H)\mathbf{W}, \tag{8}$$

and

$$\text{head}_i = \text{self-attention}(\mathbf{Q}\mathbf{W}^Q, \mathbf{K}\mathbf{W}^K, \mathbf{V}\mathbf{W}^V), \tag{9}$$

where $\mathbf{W} \in \mathbb{R}^{d_v \times d_{model}}$, $\mathbf{W}^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d_{model} \times d_k}$, and $\mathbf{W}^V \in \mathbb{R}^{d_{model} \times d_v}$ are the parameter matrices, and

$$d_k = d_v = \frac{d_{model}}{h}, \tag{10}$$

where h is the number of heads. Then, the output of the multi-head self-attention layer is fed into a position-wise feedforward network to generate the final source representation $H_{1:T} = \{h_1, \cdots, h_T\}$:

$$H_{1:T} = \text{feed-forward}(\text{multi-head}(\mathbf{Q}, \mathbf{K}, \mathbf{V})) + \mathbf{V}. \tag{11}$$

Note that the multi-head attention is masked to avoid exposing the generator to future information during the sampling process. A padding mask is used to ensure that the transformer encoders do not pay any attention to the padding tokens in the discriminator.

C Property Calculation

C.1 Diversity

The diversity was calculated from the Tanimoto similarity between the Morgan fingerprints of any two molecules in the generated set. Let V_i and V_j be the Morgan fingerprints of two arbitrary generated molecules. The Tanimoto similarity is then defined as

$$\mathrm{Sim}(V_i, V_j) = \frac{|V_i \& V_j|}{|V_i| + |V_j| - |V_i \& V_j|},$$

where $|\cdot|$ represents the number of bits set in the fingerprints, and & is the common bits in the two fingerprints. The diversity is then calculated as

$$\mathrm{Div}(\mathcal{D}_z) = 1 - \frac{1}{|\mathcal{D}_z|} \sum_{V_i, V_j \in \mathcal{D}_z} \mathrm{Sim}(V_i, V_j).$$

C.2 Drug-likeness

Drug-likeness is evaluated by the QED scores. We generally assign different weights to eight molecular descriptors: the molecular weight (MW), octanol-water partition coefficient (ALOGP), number of hydrogen bond donors (HBDs), number of hydrogen bond acceptors (HBAs), molecular polar surface area (PSA), number of rotatable bonds (ROTBs), number of aromatic rings (AROMs), and number of structural alerts (ALERTS). The calculation is as follows:

$$\mathrm{QED} = \exp\left(\frac{\sum_{i=1}^{8} W_i \ln d_i)}{\sum_{i=1}^{8} W_i}\right.,$$

where d_i and W_i represent the desirability function and weight of the i-th descriptor, respectively. Usually, the weights of the eight molecular descriptors are obtained through chemical experiments. In practice, the QED score is calculated by a function in the RDKit tool. The larger the QED score, the more drug-like the molecule.

C.3 Synthesizability

Synthesizability is evaluated by the SA score defined as

$$\mathrm{SA} = r_s - \sum_{i=1}^{5} p_i,$$

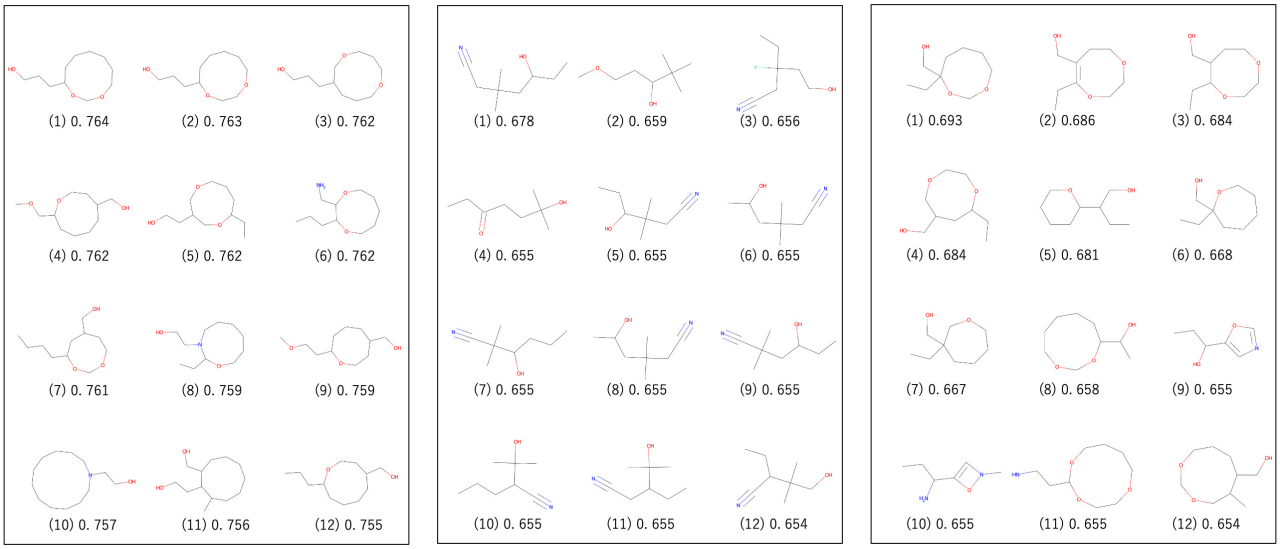
where r_s indicates the "synthetic knowledge" gained by analyzing the features of synthetic molecules. r_s is the ratio of the summed contributions from all fragments to the number of fragments in the molecule. In this work, we calculated r_s from the experimental results (Ertl and Schuffenhauer, 2009). p_i ($i \in \{1, \cdots, 5\}$) represents the ring complexity, stereo complexity, macrocycle penalty, size penalty, and bridge penalty, which were calculated using the RDKit tool. The larger the SA score, the easier the synthesis of the molecule.

C.4 Solubility

In the physical sciences, solubility is quantified by logP, where P is the partition coefficient (defined as the ratio of concentrations of a molecule in a mixture of two immiscible solvents at equilibrium). The logP can be calculated as follows:

$$\mathrm{logP} = \log \frac{c_o}{c_w},$$

where c_o and c_w indicate the substance activity in the organic and water phases, respectively. In practice, we calculate the logP of a molecule using the RDKit tool. The larger the logP value, the higher the lipophilicity of the molecule to the organic phase.



(a) Samples of the Naïve RL. (b) Samples of the ORGAN. (c) Samples of the OR(W)GAN.

Figure C.1: Top-12 molecule structures with the QED scores (drug-likeness) generated by the baseline models.

D Experimental Results on QM9 Dataset

D.1 Molecule Structures

Figure C.1 shows the top-12 molecule structures by QED score for the following molecular generative models on the QM9 dataset: Naïve RL, ORGAN, and OR(W)GAN. In organic chemistry, molecular structures follow certain rules, in which existing cyclic organic compounds generally have five- or six-member rings because of their high stability. The molecules generated by Naïve RL [Fig. C.1 (a)] have ring structures consisting of more than seven atoms, which are not stable. Additionally, most of them violate Hückel's rule, which is a fundamental rule in organic chemistry. Meanwhile, all of the top-12 compounds generated by ORGAN [Fig. C.1 (b)] are acyclic and the QED scores are lower than those for the compounds generated by Ten(W)GAN. Similar to Naïve RL, most of the compounds generated by OR(W)GAN [Fig. C.1 (c)] have ring structures consisting of more than seven atoms. A four-member ring compound (10) that does not satisfy H́uckel's rule was also generated.

D.2 Property Optimization of Ten(W)GAN

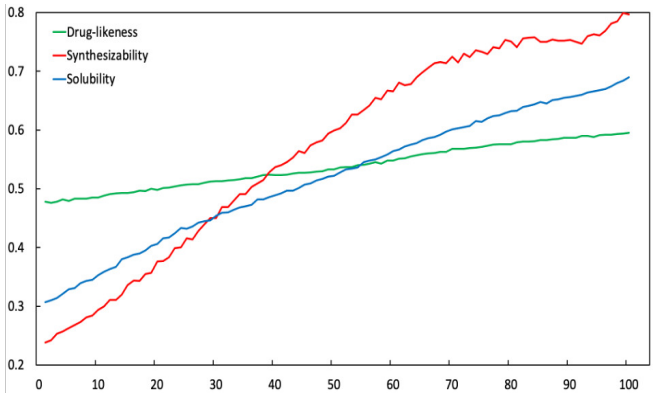
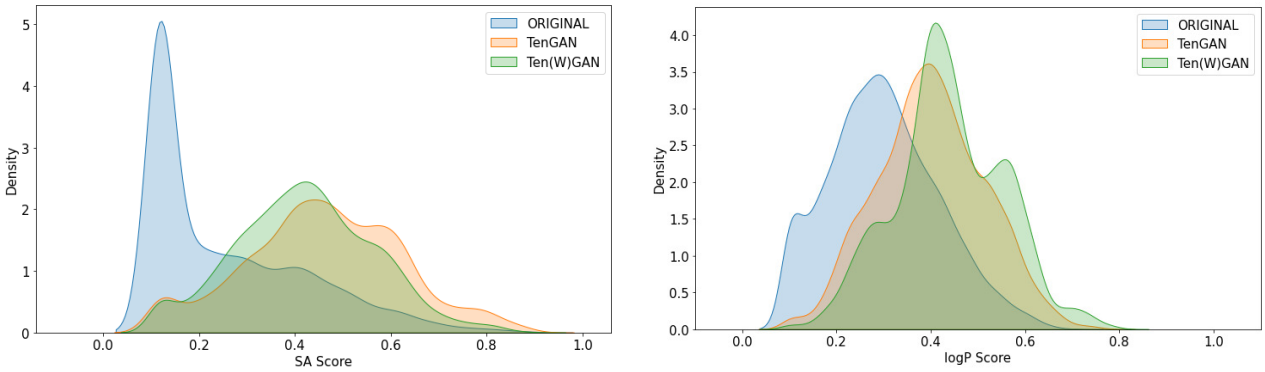


Figure D.1: Changes in the property scores according to the training epoch for Ten(W)GAN on the QM9 dataset.

Figure D.1 shows the changes in the property scores according to the training epoch for Ten(W)GAN on the QM9

dataset. Similar to TenGAN, the optimized property scores increased with the training epoch for Ten(W)GAN. When the optimized property was drug-likeness, the curve of Ten(W)GAN increased smoothly and slowly. The curves for the synthesizability and solubility increased faster than the drug-likeness because more semantic and syntactic features were required to generate molecules with high drug-likeness. Fewer semantic and syntactic features were required to generate molecules with high synthesizability because simple molecules are easier to synthesize. For solubility, molecules with higher logP values tend to have longer carbon chains. Only a few semantic and syntactic features are required to generate molecules with a high logP. Therefore, the curve for the drug-likeness increased slowly, whereas the curves for the synthesizability and solubility increased rapidly. In summary, Ten(W)GAN can also generate molecules with desired chemical properties.

D.3 Property Distributions



(a) Distribution of SA scores (synthesizability).

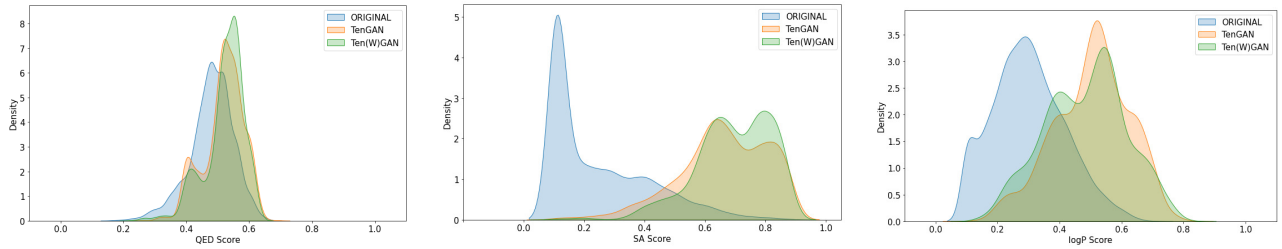
(b) Distribution of logP scores (solubility).

Figure D.2: Distributions of SA and logP with drug-likeness as the optimized property on the QM9 dataset.

Figure D.2 shows the SA and logP distributions of the generated molecules when the drug-likeness was the optimized property for TenGAN and Ten(W)GAN on the QM9 dataset. Figure D.2 (a) shows that not only the QED distribution but also the SA distribution was significantly improved with the proposed models. The SA distribution of the training set (ORIGINAL) had a peak at 0.1. TenGAN and Ten(W)GAN shifted the peaks to the right. Although the two distributions of the proposed model became more scattered with the distribution of the original training set, the mean value of the SA increased overall. Similarly, Fig. D.2 (b) shows that the logP distributions improved with the proposed models. The difference between the SA and logP scores is that the QED score could not reach the maximum value of one.

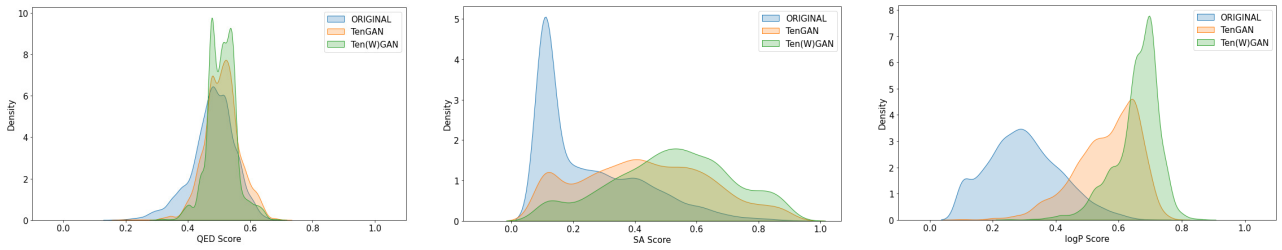
When the property of synthesizability was optimized, TenGAN and Ten(W)GAN significantly improved the distributions of the generated molecules, and the SA score reached the maximum value of one. Figure D.3 (b) shows that the distribution peak of the training set (0.1) shifted to the right to around 0.75. Therefore, TenGAN and Ten(W)GAN generated more molecules that are easy to synthesize when the optimized property was synthesizability. Meanwhile, solubility is improved by generating carbon chain molecules, which are also easy to synthesize. Therefore, Fig. D.3 (c) shows that the proposed models improved the logP distribution. Unlike synthesizability and solubility, drug-likeness generated high-quality molecules. A molecule that is easy to synthesize and has a high logP value and generally has low drug-likeness. Figure D.3 (a) shows that the QED distributions of the proposed models did not improve much when synthesizability was the optimized property.

Similarly, the distributions of QED scores were not much improved when solubility was the optimized property. Figure D.4 (b) and Fig. D.4 (c) show that TenGAN and Ten(W)GAN effectively improved the SA and logP distributions of the generated molecules, whereas the QED distribution remained the same [Fig. D.4] (a) because the solubility and drug-likeness properties are in conflict when the length of the SMILES string is small.



(a) Distribution of QED (drug-likeness). (b) Distribution of SA (synthesizability). (c) Distribution of logP (solubility).

Figure D.3: Distributions of QED, SA and logP with synthesizability as the optimized property on the QM9 dataset.



(a) Distribution of QED (drug-likeness). (b) Distribution of SA (synthesizability). (c) Distribution of logP (solubility).

Figure D.4: Distributions of QED, SA and logP with solubility as the optimized property on the QM9 dataset.

E Experimental Results on ZINC Dataset

E.1 Comparison with RNN-based Methods

Table E.1 compares the results of TenGAN and Ten(W)GAN with Naïve RL, ORGAN, and OR(W)GAN for the ZINC dataset. The results shows that the proposed models performed better than the baselines on most measures on the ZINC dataset.

For the drug-likeness, TenGAN improved the validity and uniqueness to 95.3% and 80.3%, respectively. Ten(W)GAN improved to 95.3% and 81.2%, respectively. Both TenGAN and Ten(W)GAN had the novelty scores larger than 96.0%. The QED scores were improved to 0.84. Additionally, TenGAN and Ten(W)GAN performed significantly better in terms of diversity and computational cost. When the optimized properties were synthesizability and solubility, TenGAN and Ten(W)GAN also worked better than the other models.

E.2 Top-12 Molecule Structures

Figures. E.1, E.2 and E.3 provide examles of the top-12 molecules of TenGAN and Ten(W)GAN for the ZINC dataset. For example, TenGAN and Ten(W)GAN in Fig. E.1 generated drug-like chemical structures.

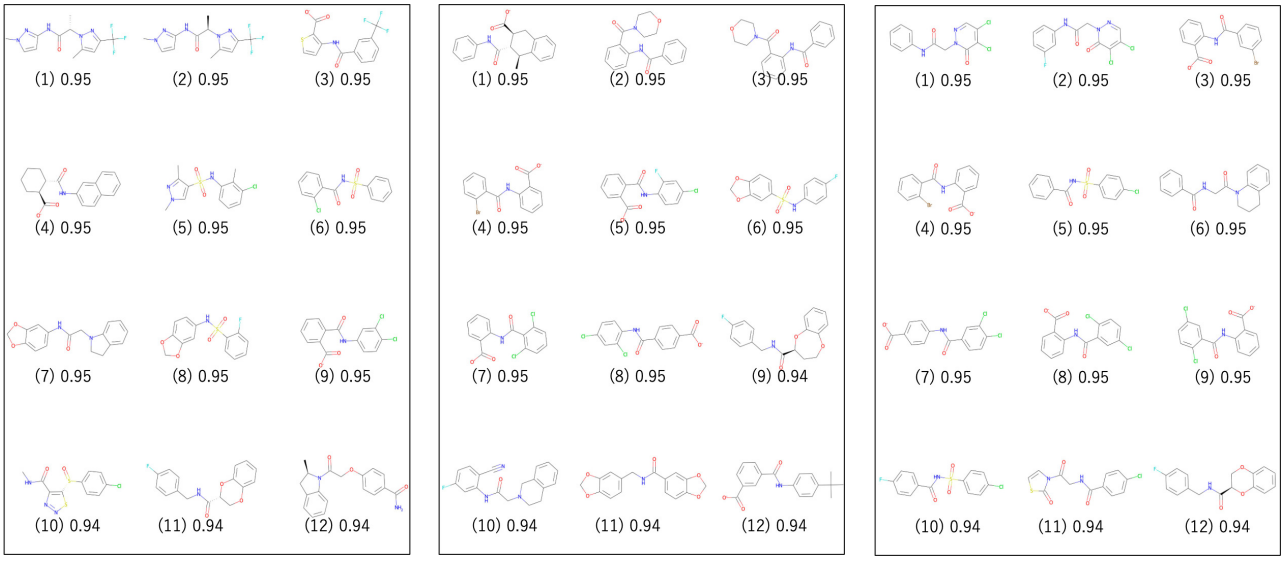
E.3 Property Optimization of Ten(W)GAN

Figures E.4 shows the two curves of the property scores according to the training epochs for TenGAN and Ten(W)GAN on the ZINC dataset. The increase in the property scores with the training epochs indicated that TenGAN and Ten(W)GAN generated molecules with the desired chemical properties.

Table E.1: Evaluation results with chemical properties compared with the closest related works on the ZINC dataset.

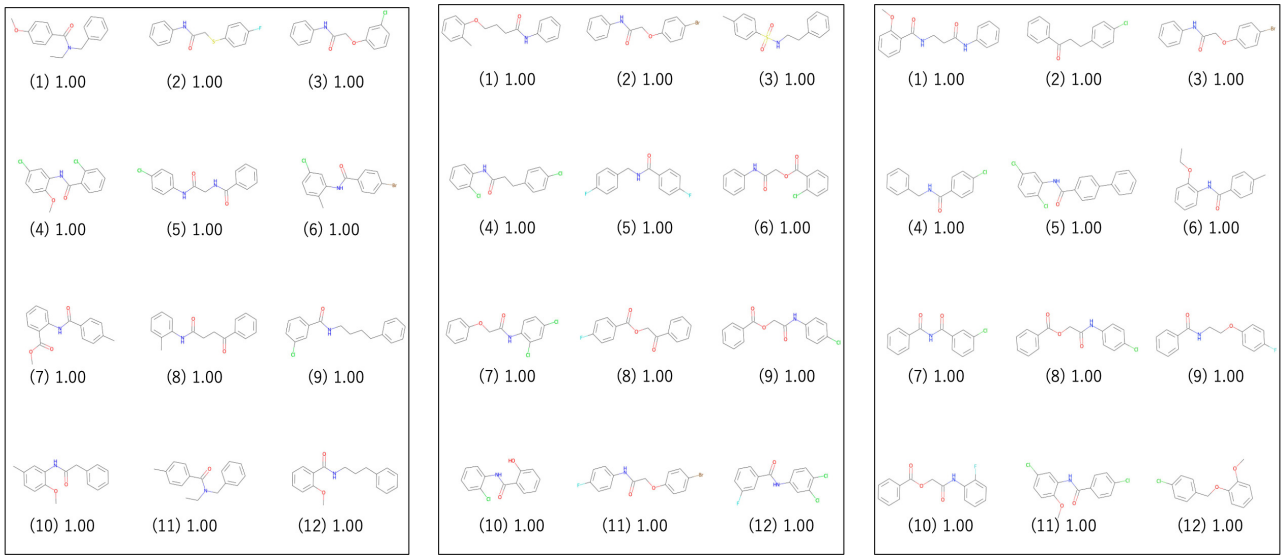
Property	Algorithm	Validity	Uniqueness	Novelty	QED	SA	logP	Diversity	Time (h)
	LSTM	51.9%	97.6%	98.6%	0.76	0.78	0.65	0.89	0.92
	TransEn	85.2%	97.8%	97.6%	0.79	0.78	0.63	0.89	0.30
Drug-likeness	Naïve RL	89.2%	65.5%	**98.5%**	0.81	0.53	0.64	0.86	12.79
	ORGAN	91.7%	54.9%	98.4%	0.80	0.48	**0.66**	0.85	12.40
	OR(W)GAN	91.5%	54.1%	98.2%	0.79	0.48	**0.66**	0.85	11.58
	TenGAN	**95.3%**	80.3%	96.2%	**0.84**	0.87	0.64	0.86	**5.21**
	Ten(W)GAN	**95.3%**	**81.2%**	96.5%	**0.84**	**0.88**	0.65	**0.87**	5.31
Synthesizability	Naïve RL	88.5%	39.3%	98.7%	0.79	0.91	0.66	0.86	13.22
	ORGAN	81.7%	44.0%	98.9%	0.77	0.87	0.61	0.86	13.29
	OR(W)GAN	76.1%	26.6%	**99.3%**	0.74	0.89	**0.75**	0.84	13.00
	TenGAN	**91.7%**	85.4%	96.5%	**0.81**	0.90	0.67	**0.86**	**3.25**
	Ten(W)GAN	90.5%	**86.5%**	96.3%	0.80	**0.91**	0.69	**0.86**	3.70
Solubility	Naïve RL	84.2%	76.4%	98.9%	0.74	**0.85**	0.74	0.86	13.58
	ORGAN	68.1%	44.2%	99.3%	0.78	0.82	0.67	0.84	13.55
	OR(W)GAN	83.8%	64.6%	98.3%	0.78	0.53	0.71	0.85	13.38
	TenGAN	**93.0%**	**80.7%**	**99.6%**	0.62	0.82	**0.92**	**0.86**	**4.96**
	Ten(W)GAN	91.7%	79.2%	**99.6%**	0.63	0.83	**0.92**	0.85	5.06

* Red boxes indicate the directly optimized properties. Bold values indicate the highest scores among different methods. The values in gray cells indicate that TenGAN / Ten(W)GAN outperformed the corresponding baseline.



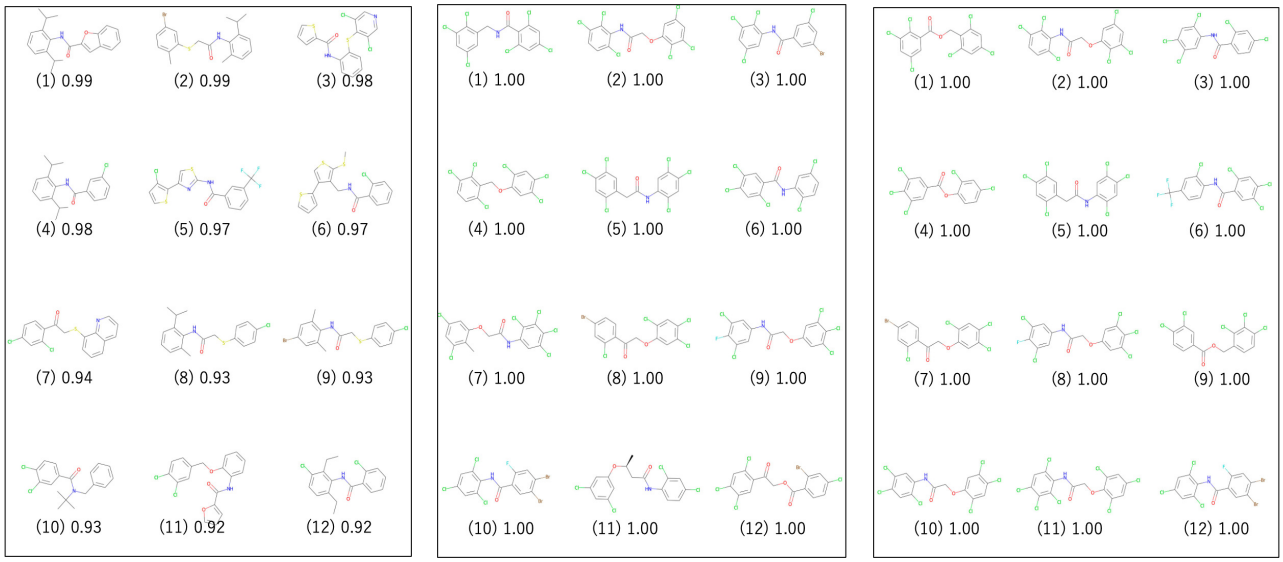
(a) Samples of the ZINC dataset. (b) Samples of the TenGAN. (c) Samples of the Ten(W)GAN.

Figure E.1: Top-12 molecule structures with the QED scores (drug-likeness) on the ZINC dataset.



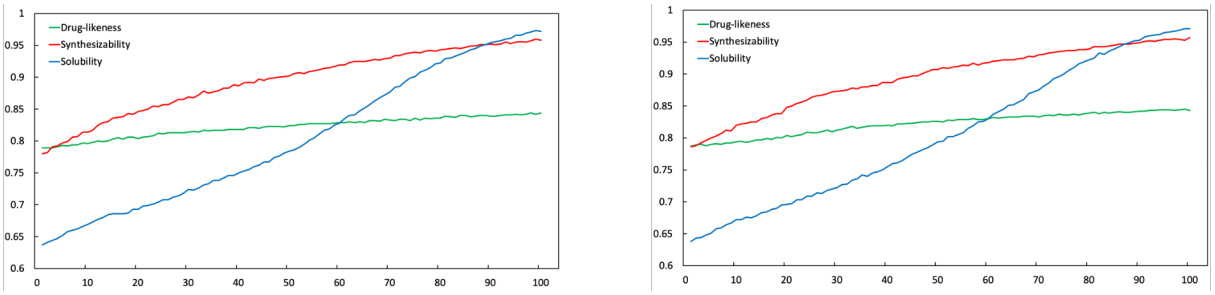
(a) Samples of the ZINC dataset. (b) Samples of the TenGAN. (c) Samples of the Ten(W)GAN.

Figure E.2: Top-12 molecule structures with the SA scores (synthesizability) on the ZINC dataset.



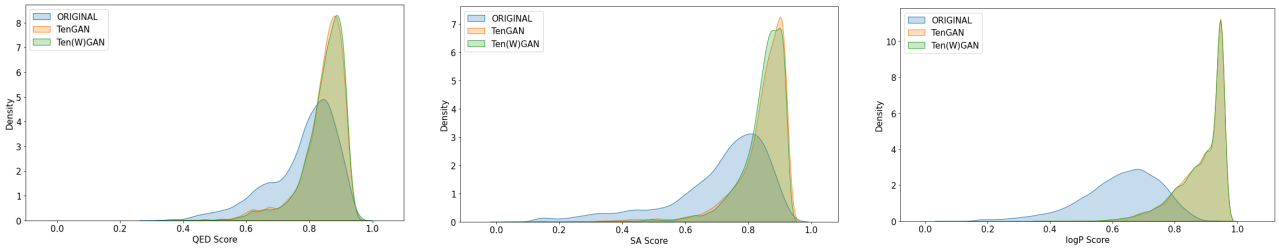
(a) Samples of the ZINC dataset. (b) Samples of the TenGAN. (c) Samples of the Ten(W)GAN.

Figure E.3: Top-12 molecule structures with the logP scores (solubility) on the ZINC dataset.



(a) Property change curve for TenGAN. (b) Property change curve for Ten(W)GAN.

Figure E.4: Changes in the property scores according to the training epoch on the ZINC dataset.



(a) Distribution of QED (drug-likeness). (b) Distribution of SA (synthesizability). (c) Distribution of logP (solubility).

Figure E.5: Distributions with drug-likeness, synthesizability, and solubility as the optimized properties on the ZINC dataset.

Table E.2: Comparison of TenGAN with graph-based and VAE-based algorithms on the ZINC dataset.

Algorithm	Validity	Uniqueness	Novelty	Total	logP
JTVAE	**100.0%**	19.8%	99.8%	19.8%	0.51
GraphAF	**100.0%**	83.2%	100.0%	83.2%	0.61
CharVAE	86.7%	81.2%	26.4%	18.6%	0.55
GramVAE	91.9%	77.2%	11.9%	8.5%	0.58
TransVAE	25.4%	**100.0%**	**100.0%**	25.4%	0.14
MoFlow	26.0%	**100.0%**	**100.0%**	26.0%	0.71
MolGAN	95.3%	4.3%	**100.0%**	4.1%	0.62
TenGAN	91.6%	95.4%	97.5%	**85.2%**	**0.73**
Ten(W)GAN	91.2%	95.7%	97.2%	84.8%	0.71

E.4 Property Distributions

Figure E.5 shows the distributions with drug-likeness, synthesizability, and solubility as the optimized properties on the ZINC dataset. From the figures, it was observed that TenGAN and Ten(W)GAN effectively improved the QED, SA, and logP distributions of the generated molecules.

E.5 Comparison with Graph and VAE Methods

Table E.2 shows that although JTVAE and GraphAF realized the highest validity (i.e., 100%), the uniqueness and logP score were much lower than TenGAN. Similarly, CharVAE and GramVAE had much lower novelty and logP scores than TenGAN. Although TransVAE combined a VAE with a transformer, it had the smallest validity among these baseline models (i.e., 25.4%). In summary, TenGAN performed best in terms of Total and logP value while maintaining high validity, uniqueness, and novelty.

Table F.1: Effect of different λ on the QM9 dataset.

λ	QED	Validity	Uniqueness	Novelty	Diversity
0	0.59	98.1%	82.6%	**99.8%**	0.90
0.1	**0.60**	97.8%	82.6%	**99.8%**	0.89
0.3	0.57	**98.7%**	60.2%	96.7%	0.89
0.5	0.57	97.8%	70.7%	98.0%	0.90
0.7	0.52	96.8%	71.5%	92.9%	0.92
0.9	0.47	91.3%	**91.6%**	86.6%	**0.93**
1.0	0.46	82.0%	90.2%	85.8%	**0.93**

Table F.2: Effect of different N on the QM9 dataset.

N	QED	Validity	Uniqueness	Novelty	Time (h)
1	0.48	89.5%	**93.9%**	82.7%	0.33
2	**0.57**	98.1%	77.9%	**99.4%**	2.16
4	0.55	98.2%	60.2%	94.5%	2.99
8	**0.57**	97.8%	62.0%	98.7%	4.73
16	**0.57**	97.8%	70.7%	98.0%	5.06
32	0.56	**98.6%**	71.9%	97.4%	11.62

Table F.3: Effect of the variant SMILES on the ZINC dataset.

Algorithm	QED	Validity	Uniqueness	Novelty
TenGAN w/o	0.83	93.6%	69.2%	96.1%
TenGAN	**0.84**	**95.3%**	**80.3%**	**96.2%**
Ten(W)GAN w/o	0.84	93.7%	74.2%	95.9%
Ten(W)GAN	**0.84**	**95.3%**	**81.2%**	**96.5%**

* "w/o" indicates model training without the variant SMILES.

F Ablation Studies

Effect of λ. λ represents the trade-off between the GAN and RL. Different $\lambda \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$ were used on the drug-likeness to explore the effect of λ on the performance of TenGAN. The results are presented in Table F.1. Increasing λ decreased the QED score, validity, and novelty while increasing the uniqueness and diversity. These results are reasonable because RL mainly controls the training process when λ is small. Particularly, the training is fully executed by RL when $\lambda = 0$. Because RL updates the generator by improving the average reward of generated molecules and invalid molecules receive zero rewards, the generator tends to generate mostly valid molecules in this case. However, a higher validity and QED score mean that the model must use less data to learn strategies (i.e., the overlap between the training dataset distribution and the generated data distribution shown in Fig. 4). Therefore, when RL dominates the training process, the uniqueness and diversity decrease.

Effect of N. The rollout times N controls the number of MC searches, and it also affects the performance of TenGAN. Intuitively, when N is small, the randomness of the strings generated based on substrings and the variance of rewards are relatively large, which may result in model instability. Table F.2 shows the effect of N on the performance of TenGAN with regard to drug-likeness (QED score). Increasing N stabilizes the validity, uniqueness, and novelty, but the computational time increased as N increases.

Effect of Variant SMILES. We also evaluated the effect of the variant SMILES on the performance of TenGAN and Ten(W)GAN, respectively. The evaluation results are shown in Table F.3. All metrics (i.e., validity, uniqueness, novelty, and QED scores) of TenGAN and Ten(W)GAN improved after using the variant SMILES technique. Therefore, variant SMILES can help TenGAN and Ten(W)GAN to fully train and learn more syntactic and semantic rules from SMILES strings.