Comparative Study of Deep Generative Models on Chemical Space Coverage

Jie Zhang, Rocío Mercado, Ola Engkvist, Hongming Chen

Submitted date: 13/11/2020 • Posted date: 16/11/2020

In recent years, deep molecular generative models have emerged as novel methods for de novo molecular design. Thanks to the rapid advance of deep learning techniques, deep learning architectures such as recurrent neural networks, generative autoencoders, and adversarial networks, to give a few examples, have been employed for constructing generative models. However, so far the metrics used to evaluate these deep generative models are not discriminative enough to separate the performance of various state-of-the-art generative models. This work presents a novel metric for evaluating deep molecular generative models; this new metric is based on the chemical space coverage of a reference database, and compares not only the molecular structures, but also the ring systems and functional groups, reproduced from a reference dataset of 1M structures. In this study, the performance of 7 different molecular generative models was compared by calculating their structure and substructure coverage of the GDB-13 database while using a 1M subset of GDB-13 for training. Our study shows that the performance of various generative models varies significantly using the benchmarking metrics introduced herein, such that generalization capability of the generative model can be clearly differentiated. Additionally, the coverage of ring systems and functional groups existing in GDB-13 was also compared between the models. Our study provides a useful new metric that can be used for evaluating and comparing generative models.

Comparative study of deep generative models on chemical space coverage

Jie Zhang[&, $, ‖], *Rocío Mercado*[€], *Ola Engkvist*[€], *Hongming Chen*[&, ‖, *]

[&]Guangdong Laboratory Animals Monitoring Institute, Guangzhou, 510663, P. R. China

[$]State Key Laboratory of Respiratory Disease, Guangzhou Institutes of Biomedicine and Health, Chinese Academy of Sciences, Guangzhou 510530, P. R. China

[‖]Bioland Laboratory (Guangzhou Regenerative Medicine and Health - Guangdong Laboratory)，Guangzhou 510530, P. R. China

[€]Hit Discovery, Discovery Sciences, R&D, AstraZeneca, Gothenburg, Sweden

[*]Correspondence e-mail: chen_hongming@grmh-gdl.com

Abstract

In recent years, deep molecular generative models have emerged as novel methods for *de novo* molecular design. Thanks to the rapid advance of deep learning techniques, deep learning architectures such as recurrent neural networks, generative autoencoders, and adversarial networks, to give a few examples, have been employed for constructing generative models. However, so far the metrics used to evaluate these deep generative models are not discriminative enough to separate the performance of various state-of-the-art generative models. This work presents a novel metric for evaluating deep molecular generative models; this new metric is based on the chemical space coverage of a reference database, and compares not only the molecular structures, but also the ring systems and functional groups, reproduced from a reference dataset of 1M structures. In this study, the performance of 7 different molecular generative models was compared by calculating their structure and substructure coverage of the GDB-13 database while using a 1M subset of GDB-13 for training. Our study shows that the performance of various generative models varies significantly using the benchmarking metrics introduced herein, such that generalization capability of the generative model can be clearly differentiated. Additionally, the coverage of ring systems and functional groups existing in GDB-13 was also compared between the models. Our study provides a useful new metric that can be used for evaluating and comparing generative models.

Keywords: deep generative models, chemical space coverage, ring systems, functional groups

Introduction

Deep learning has been successfully used in many fields to learn relationships that are too complex to learn using traditional computer algorithms, including early image classification,[1, 2] facial recognition, and music recognition.[3] Deep learning even surpasses the performance of human experts in some challenging tasks, such as playing GO.[4] Moreover, deep generative models play important roles in tasks like music composition,[5] image generation,[6] and language translation.[7] In the last five years, deep generative modeling has also been applied in the fields of cheminformatics and molecular design. One interesting example is using deep neural networks for compound structure generation. [8-11]

The number of chemically feasible, drug-like molecules has been estimated to be on the order of $10^{60} – 10^{100}$ compounds.[12] For such a large chemical space, it is clearly impossible to synthesize and test every compound for pharmaceutical applications. To efficiently explore the space, molecular generative models have emerged in recent years with the aim of better navigating through this huge chemical space for *de novo* molecule design.

De novo molecular design has long been put forward as a way to accelerate the drug discovery process as it is expected to save time and resources in drug discovery, where it can take over a decade and billions of dollars in investment to bring a single drug to market.[13] Historically, *de novo* design methods have been mainly rule-driven and used brute force algorithms to achieve their goal. For example, creating a virtual library using fixed rules and building blocks, then scoring each compound in the virtual library to find the best compound. In contrast, deep generative molecular design is the concept of generating molecules using deep neural networks. Deep generative models are data-driven methods which generate compound structures by learning the underlying probability distributions in a compound dataset instead of screening existing databases for molecules that fit the desired profile. Deep generative models are powerful as they allow chemists to bypass models using hard-coded chemical rules which do not scale to larger datasets. Furthermore, not all chemical rules are easy to define. Using deep generative models, one can avoid enumerating all possible structures for a given application and then screening them (a daunting task). Instead, one can simply train a model using known compounds, and sample the model for the desired set of properties (e.g. ADMET profile) to get out promising structures. *De novo* generative models can generate structures that are in significantly narrower, but more promising, regions of chemical space. Moreover, deep learning methods can take advantage of all the information available in ever-increasing large public datasets, thanks to automation technologies used in high-throughput screening and parallel synthesis.[14]

In recent years, many molecular generative models have been published, such as CharRNN, VAE, and REINVENT, which are remarkable at sampling molecules both in- and outside the training set used to learn chemistry rules.[11, 15-18] Notably, many of these generative models have been benchmarked using existing "distribution-based" metrics implemented in open-source programs such as MOSES[10] or GuacaMol.[19] However, these methods are in general non-discriminative as many of these state-of-the-art (SOTA) models perform quite well across all the included metrics, such that it is difficult to compare them and gain a deep understanding of each model's strengths and weaknesses. We previously proposed a new metric: the percent coverage of functional groups present in GDB-13.[20] GDB-13 contains 977,468,314 structures, which enumerate small organic molecules containing up to 13 atoms of C, N, O, S, and Cl by following simple chemical stability and synthetic feasibility rules.[21] The generalization capability of deep generative models is assessed by computing how much of the whole GDB-13 can be recovered by a model trained from a small GDB-13 subset. In the current study, as an extension of our previous work,[20] we apply the idea as a way for benchmarking the performance of multiple generative models.

Substructure has long been used to characterize the composition of compounds. One concept is the so-called *functional group*, frequently used in many fields in chemistry, including medicinal chemistry. A functional group is defined as a subset of connected atoms in a molecule that in some way endow specific intrinsic properties (or *functions*) to a molecule. Furthermore, the presence or absence of a functional group in a molecule could determine whether a molecule will react in a given reaction. Some of the most common groups in medicinal chemistry include amides (RC(=O)NR'R"), ethers (R–O–R'), and amines (RR'NR"), where R, R', and R" represent organic groups or hydrogen atoms.[22] Another substructure-based concept is the *ring system*; ring systems are the key components of molecular scaffolds. They play an important role in a molecule's observed properties, such as the electronics, scaffold rigidity, molecular reactivity, and toxicity. On average six new ring systems enter the drug space each year and approximately 28% of new drugs contain a new ring system.[23]

In this study, we extend our previous analysis on chemical space coverage of generative models from compound coverage of GDB-13 to include coverage of functional groups and ring systems. We investigated the percentage of chemical space covered in terms of full structures, functional groups, and ring systems by published SOTA generative models. The size of GDB-13 was hypothesized to be large enough to highlight differences between the various models.

Three major classes of deep generative models are benchmarked and studied in this work, including those based on recurrent neural networks (RNNs), autoencoder (AE) based networks , generative adversarial networks (GANs), and graph neural networks (GNNs). The deep generative models based on RNNs include REINVENT[15, 20, 24] and CharRNN,[25] which use SMILES as the input and output strings. VAE,[26] AAE,[18, 27] ORGAN,[17] and LatentGAN[11] adopt either an AE or GAN for structure generation using SMILES. Besides the SMILES-based generative models, one graph-based generative model, GraphINVENT,[28] which uses GNNs in its core architecture, is also included in the benchmark study. An effort was made to cover most of the major types of generative model architectures in this study, in the hopes that this would provide a comprehensive comparison for existing generative models.

Methods

Extraction of functional groups and ring systems. To identify functional groups (FG) in the various sets of molecules in this work (generated molecule sets, and GDB-13), the RDKit functional group identification package,[29] which is based on an algorithm introduced by Peter

Ertl for automatically identifying functional groups, was used.[30] The advantage of the method is that it is not based on manually curated lists of functional groups, and thus can be applied to any chemical series. It is important to note that different chemists have slightly different definitions of what is a functional group; however, as the benchmark introduced here calculated ratios of functional groups in the generated and reference sets, a difference in opinions between chemists shouldn't be relevant. The extraction of compound ring system (RS) was done using RDKit. First, all monocyclic rings were retrieved; monocyclic rings were then fused depending on if individual ring systems shared atoms or not.

Generative models. The models studied in this study include CharRNN, REINVENT, AAE, VAE, ORGAN, LatentGAN, and GraphINVENT. The REINVENT code available at the github.com/undeadpixel/reinvent-randomized repo[24, 31] was used; the CharRNN, AAE, VAE, and ORGAN codes available at the MOSES GitHub repository[10, 32] were used; the LatentGAN code available at the github.com/Dierme/latent-gan repo[11, 33] was used; finally, the GraphINVENT code available at the github.com/MolecularAI/GraphINVENT repo[28, 34, 35] was used. All methods except for GraphINVENT are string-based generative models, whereas GraphINVENT is a graph-based generative model.

Training. The GDB-13 database is used as the reference chemical space for this study.[20] A one million (1M) molecule subset of GDB-13 was randomly selected and used as the training set for all the generative models. Another 200K molecules of GDB-13 were selected as the validation set for calculating the validation loss. During training, a check point model was saved at every epoch. The check point model with the lowest validation loss was chosen as the final model for sampling 1 billion (1B) SMILES.

Hyperparameters. For REINVENT, hyperparameters were taken from the GitHub repo.[31] For CharRNN, AAE, VAE, and ORGAN, the parameters were taken from the models' config file in MOSES GitHub repo without further optimization. For LatentGAN, , the default values of parameters in the GitHub repo were adopted.[11, 33] For GraphINVENT, parameters and hyperparameters for the best performing model (cGGNN) in the original publication were used and not further optimized.[35] Detailed hyperparameters for each model can be found in SI Table S2.

Sampling. Once each model was trained, 1B compounds were sampled from each trained model. The functional groups and ring systems were then identified for all sample sets as well as the full GDB-13 set. All compounds in the analysis were standardized by converting to RDKit canonical SMILES. Molecular graphs generated using GraphINVENT were further sanitized during the conversion to canonical RDKit SMILES for a more fair comparison to the other models.

Technical details. For models in the MOSES repository and REINVENT, the training was done using Python 3.6[36] and PyTorch 1.4[37]. To accelerate sampling for 1B SMILES, the largest batch size allowed by the GPU memory was adopted; for example, ORGAN, AAE, and VAE adopted a sampling batch size of 25, 000, and CharRNN adopted a sampling batch size of 20,000. Also, LatentGAN was trained using tensorflow-gpu 2.2, which adopted a sampling batch size of 50,000. All the computations were performed on Linux workstations with GeForce RTX 2080Ti graphic cards using CUDA 10.1. Canonical SMILES and dataset analysis were carried out using RDKit.[29] The Wasserstein distances[38] between distributions in Figure 2 were calculated with an in-house script using SciPy.[39] Finally, GraphINVENT runs using Python 3.6 and PyTorch 1.2.

Results and Discussions

Analysis of the GDB-13 database

GDB-13 contains theoretically drug-like compounds whose heavy atom count is less than or equal to 13 and, in total, comprises of 975,820,210 molecules, 21,852,845 ring systems, and 4,401,506 functional groups. The distribution of the occurrence frequency of these ring systems and functional groups is shown in Figure 1. Figure 1 indicates that ~80% of ring systems and ~66% functional groups in GDB-13 occur in compounds only 1-2 times, while only ~1% of ring systems and functional groups are observed in GDB-13 molecules more than 200 times. In general, most of the ring systems (~93%) and functional groups (~91%) appear in GDB-13 less than 20 times.

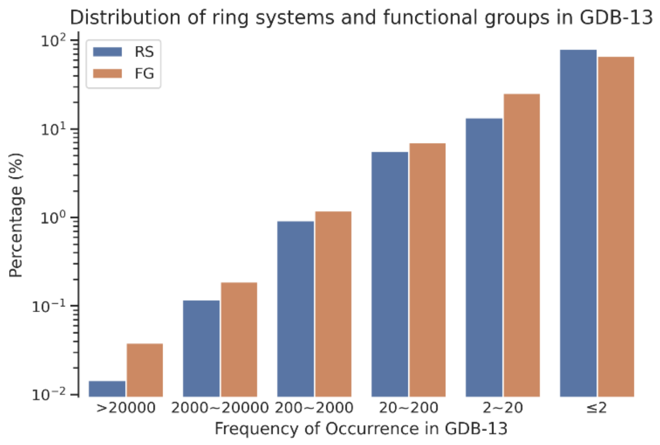


Figure 1. Distribution of ring systems (RS) and functional groups (FG) in GDB-13 according to the frequency of occurrence. Y-axis is the percentage plotted on a logarithmic scale.

Analysis of the 1M training dataset

One million SMILES were randomly selected from the GDB-13 database for the training set, which corresponds to roughly 0.1% of the total GDB-13 dataset. The training set contains around 0.9% of the ring systems and functional groups in the whole GDB-13 database (Table 1). The coverage of the ring systems and functional groups is nine times as high as the coverage of compounds, which is obviously due to the fact that some ring systems and functional groups occur far more than once in GDB-13, as shown in Figure 1.

Table 1. Summary of GDB-13 coverage in the training set, consisting of 1M randomly selected molecules.

Item	Counts in the training dataset (1M)	Coverage of GDB-13
Compounds	1,000,000	~0.1%
Ring systems	202,848	~0.9%
Functional groups	38,209	~0.9%

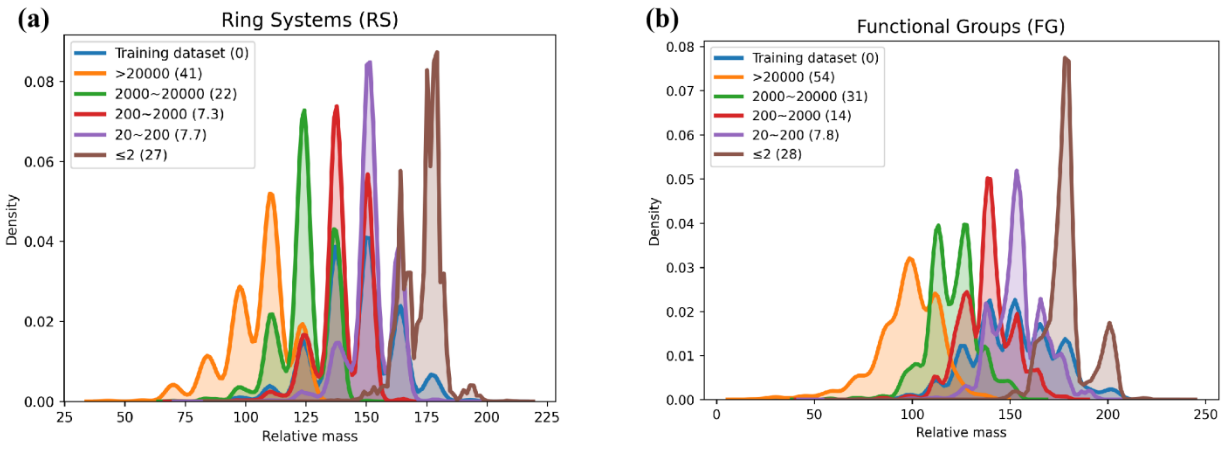


Figure 2. The molecular weight distributions for the different substructures in GDB-13. The different colors indicate distributions involving substructures that occur in GDB-13 a similar number of times (i.e. orange is substructures that occur >20,000 times in GDB-13, brown is substructures that occur ≤ 2 times). In the key, the numbers in parentheses indicate the Wasserstein distance between the training set distribution and the indicated distribution. (a) Ring systems (RS). (b) Functional groups (FG).

The molecular weights (MWs) of ring systems and functional groups in the training set, grouped by frequency of occurrence, are shown in Figure 2. The MWs here were calculated from the composition of specific ring systems and functional groups rather than the full compound. It is observed that their probability of occurrence decreases with increasing MW. For example, the mean MW of RS and FG which occur with a frequency >20,000 is around 100; however, for RS and FG which occur <=2 times in GDB-13, the mean MW is around 170. More basic RS and FG, such as C1CC1 (cyclopropyl) and C=O (carbonyl), respectively, tend to have smaller MW compared to complex RS and FG. Furthermore, many complex RS and FG can be built from the basic components via enumeration and combination following the chemical rules extracted from the training dataset.

Training and sampling speed

All deep molecular generative models were trained with the same training set of 1M compounds. Each epoch of training took 17-20 min for most models (Figure 3), except CharRNN (28 min) and GraphINVENT (36 min). In general, the training speed of all the models is acceptable. We observed that training SMILES-based models is faster than the graph-based model; this is understandable because the action space of the graph-based model is much larger than any of the SMILES-based models.

Nonetheless, the sampling speed of the generative models was observed to vary significantly. The sampling speed of REINVENT, AAE, ORGAN, and VAE were all above 7000 compounds per second, while the sampling speed of CharRNN, LatentGAN, and GraphINVENT were only 200, 100, and 1100 compounds per second, respectively. It should be noted that both training and sampling speeds are strongly related to the batch size that is limited by the memory of the GPU.

In current work, the default batch size as specified in the code was used during the training, while for sampling, the largest batch size allowed by the GPU memory was chosen.

Given the relatively small size of the training set (1M molecules), all the deep generative modes had a tractable training speed. In terms of sampling, the sampling speed was limited by each model's architecture and size; using a larger sampling batch size allowed by a greater GPU memory could boost the sampling speed.

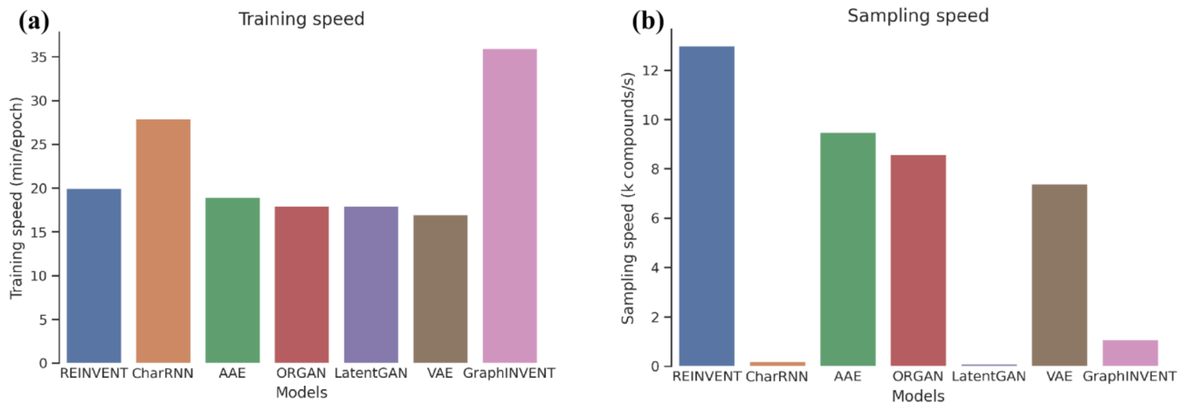


Figure 3. Training and sampling speeds of the generative models benchmarked in this work. (a) Training speed, which is the time required per epoch during training. (b) Sampling speed, which is the number of SMILES/graphs generated per second (including invalid ones).

Validity and repetition rate of sampled molecules

We first check the validity of the molecules generated by all the deep generative models, which is defined as the percentage of chemically valid SMILES/graphs in the 1B generated set. Table 1 shows that the validity in general is satisfactory for all models, where all models achieve validity higher than 90 percent. RNN based models (REINVENT and CharRNN) have the highest validity which is above 99.3% (Table 2). The validity of LatentGAN and GraphINVENT are 94.7% and 95.3% respectively, which are lowest among all the models. In order to check how much duplication is generated among the sample sets, the repetition rate (R_rept) was calculated via the formula below:

$$R_{rept} = \frac{N_{valid} - N_{unique}}{N_{unique}}, \tag{1}$$

where, N_{valid} is the number of total valid molecules in the 1B generated set and N_{unique} is the number of unique molecules in the 1B generated set (i.e. duplicates removed). The compound repetition rates of most deep generative models were around 1.0, that is to say, most compounds were sampled twice on the average. ORGAN and CharRNN have the highest repetition rates, which are 3.6 and 1.4 respectively, whereas GraphINVENT has the lowest (0.7).

It seems that all the deep generative models had a satisfactory high percent validity that was above 94% in this study. The validity of CharRNN reached as high as 99.7%. Most models had a repetition rate around 1 except CharRNN and ORGAN. ORGAN had a repetitive rate as high as 3.6, which means that each generated compound was sampled 4.6 times on average. The high repetition rate resulted in a low overall compound coverage for ORGAN, where the coverage was as low as 16%.

Table 2. Percentage of the valid molecules and molecular repetition rate in the 1B generated set for each model in this study. The uncertainty in the percent validity was less than a fraction of a percentage point for each model.

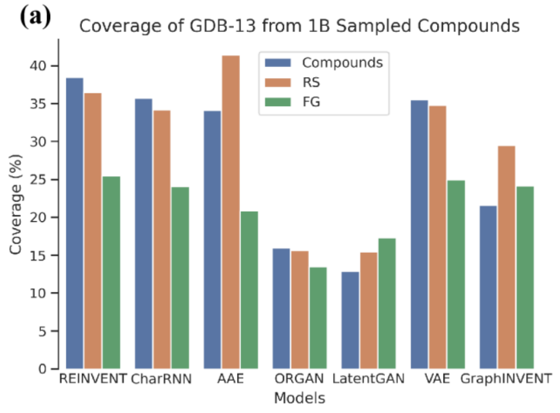
Model	REINVENT	CharRNN	AAE	ORGAN	LatentGAN	VAE	GraphINVENT
Validity (%)	99.3	99.7	97.8	97.2	94.7	98.2	95.3
Repetition rate	0.9	1.4	0.8	3.6	0.9	1.0	0.7

Coverage of GDB-13 chemical space

The molecule and substructure coverage of GDB-13 space for all generative models studied herein is shown in Figure 4a. It can be seen that all the models possess good capabilities for generalization, surpassing the coverage of the 1M training set used, which has a ~0.1% coverage of GDB-13 compounds, ~0.9% coverage of GDB-13 ring systems, and ~0.9% coverage of GDB-13 functional groups. REINVENT achieves the highest compound and FG coverage (39% and 25%, respectively), while AAE achieves best RS coverage (41%). The GAN models (ORGAN and LatentGAN) have lowest coverage at all three levels.

Using these new metrics, the difference in performance among these models is more pronounced; this is in contrast to a previous benchmarking study using the MOSES metrics,[10] where the two GAN models appear to perform similarly with the CharRNN, AAE, and VAE models.

Overall, REINVENT, CharRNN, AAE, and VAE are the top-ranking models in this benchmarking study. They have a compound coverage, RS coverage, and FG coverage around 34%, 34%, and 21%, respectively, in all cases. The performance of GraphINVENT is between the two classes of models discussed above, and demonstrates coverage scores of 22%, 30%, and 24% for compound coverage, RS coverage, and FG coverage, respectively.



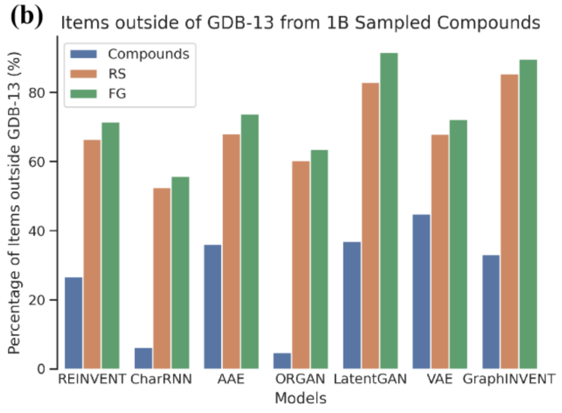


Figure 4. Coverage of GDB-13 chemical space using 1B sampled molecules. (a) Coverage of compounds, ring systems (RS), and functional groups (FG) in GDB-13 ($P_{covered}$). (b) Percentage of sampled molecules, RS, and FG that are outside the chemical space of GDB-13 (P_{out}).

Coverage of compounds, RS, and FG in GDB-13 was calculated via the formula below:

$$P_{covered} = \frac{N_{unique\_in}}{N_{GDB13}} * 100\% \,, \tag{1}$$

where N_{unique_in} is the number of unique sampled compounds, RS, or FG that are also found in GDB-13, and N_{GDB13} is the total number of compounds, RS, or FG present in GDB-13.

The percentage of sampled compounds, RS, or FG that are outside the chemical space of GDB-13 was calculated via the formula below:

$$P_{out} = \frac{N_{unique\_out}}{N_{unique}} * 100\% \,, \tag{2}$$

where N_{unique_out} is the number of unique sampled compounds, RS, or FG that are *not* found in GDB-13, and N_{unique} is the total number of unique compounds, RS, or FG in the generated sets.

There are four major metrics mentioned above, namely validity, repetition rate, coverage of GDB-13 chemical space, and percentage outside GDB-13. Validity represents how good a generative model has learned the chemical rules for constructing compounds; repetition rate represents how much structure duplication exists in the generated compound set; generalization capacities of models can be measured with the coverage of GDB-13 after being trained on a smaller fraction of chemical space. As a supplement to above metrics, percentage outside GDB-13 shows how many sampled compounds fall outside the scope of GDB-13 (which are usually non drug-like compounds).. Also, these four metrics are not independent from each other. For example, if a model has a high validity and a small percentage sampled outside GDB-13, given that exactly 1B compounds are sampled, the only reasonable explanation for a low GDB-13 coverage is a high repetition rate.

Figure 4b shows the generated structures outside GDB-13. As GDB-13 uses filters to remove molecules that do not satisfy simple chemical stability and synthetic feasibility rules, such as ring-strain criteria and valency rules, there are many structures that can be generated which violate the filters used by GDB-13. For example, there are around 30% valid SMILES generated by REINVENT which fall outside the scope of GDB-13 chemical space. However, for CharRNN and ORGAN, only 6% and 5% of their respective generated sets fall outside GDB-13, which is much lower than other models in this study. As the percent validity of the structures generated by both models is above 97%, we conclude that the lower fraction of compounds outside GDB-13 is due to the high repetition rate of compounds for these models, as shown in Table 2. As for the percentage of RS and FG outside of the scope of GDB-13, more than 50% of all FG and RS found in the generated sets for each model are outside the GDB-13 chemical space.

After training with a subset of the GDB-13 database (0.1%), all the generative models showed promising performance in terms of compound coverage. Around 13% compounds of GDB-13 were covered with 1B SMILES sampled by the LatentGAN, which is 130 times greater than the coverage of the training dataset itself. The model with the best performance in this study is REINVENT, which has an observed compound coverage as high as 39%. Thus, we conclude that deep generative models in general have satisfactory learning and generalization capacities. In terms of overall GDB-13 compound coverage, the rank of performance in descending order is REINVENT > CharRNN > VAE > AAE > GraphINVENT > ORGAN > LatentGAN.

The GDB-13 coverage of RS and FG was generally less than the coverage of compounds, except in the cases of AAE, LatentGAN, and GraphINVENT. However, in these cases, greater than 60%

RS and FG in the generated set were outside the scope GDB-13 chemical space, while less than 40% of generated molecules were outside GDB-13. In terms of RS coverage of GDB-13, the rank of performance in descending order is AAE > REINVENT > VAE > CharRNN > GraphINVENT > ORGAN > LatentGAN. In terms of FG coverage of GDB-13, the rank of performance in descending order is REINVENT > VAE > GraphINVENT > CharRNN > AAE > LatentGAN > ORGAN. Examples of the most commonly observed groups in structures generated by the two best models, REINVENT and AAE, are shown in Figures 6 & 8. Examples of the most commonly observed groups in structures generated by one of the worst models, LatentGAN, are shown in Figures 7 & 9.

Relationship between the coverage of GDB-13 and occurrence frequency

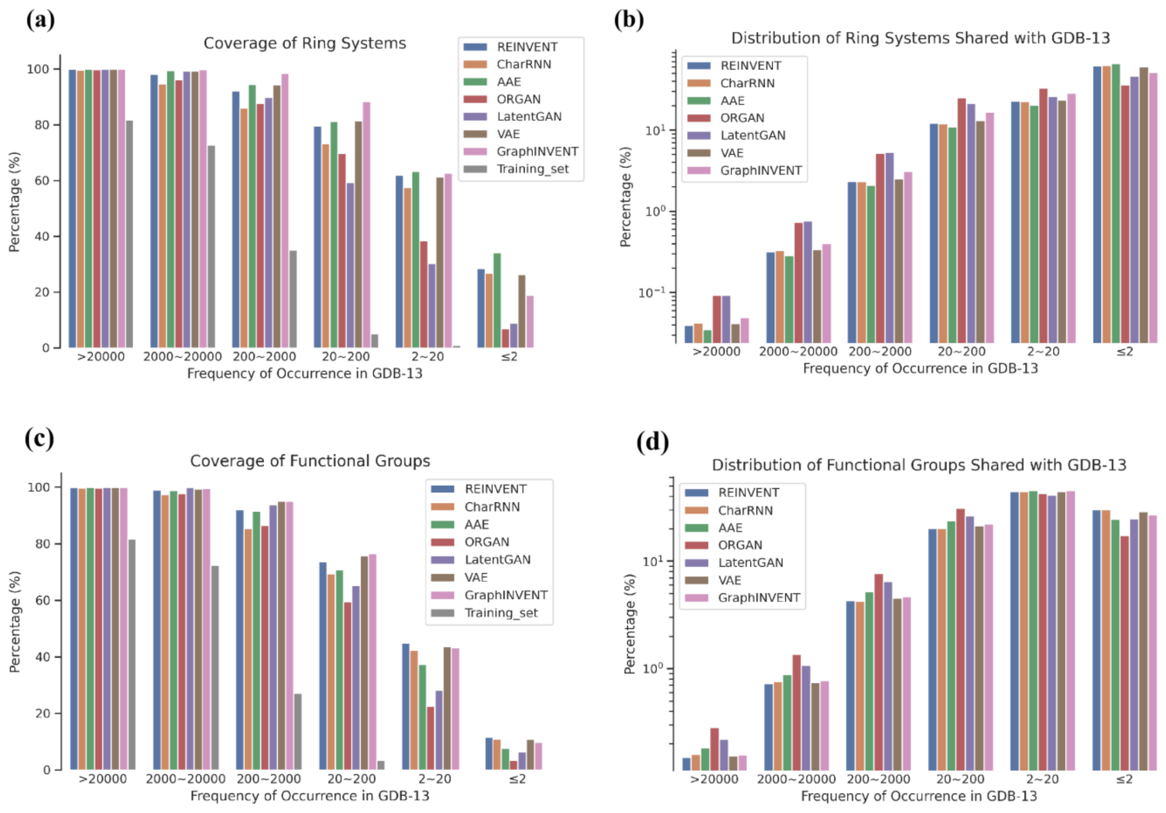


Figure 5. Coverage of GDB-13 chemical space from 1B sampled molecules, grouped by the occurrence frequency of molecules in GDB-13. (a & c) Coverage of RS and FG. (b & d) Distribution of generated RS and FG that are shared with the chemical space of GDB-13. The y-axes for (b) and (d) are displayed in logarithmic scale.

The coverage of GDB-13 chemical space from 1B sampled molecules, grouped by the occurrence frequency of molecules in GDB-13, was calculated via the formula below:

$$P_{covered} = \frac{N_{unique\_in}(R_m,\ R_n)}{N_{GDB13}(R_m,\ R_n)} * 100\% \, , \tag{3}$$

where $N_{unique\_in}(R_m,\ R_n)$ is the number of unique RS or FG in the sampled set that have an occurrence frequency in the range of R_m to R_n in GDB-13, and $N_{GDB13}(R_m,\ R_n)$ is the total number of RS or FG in GDB-13 with an occurrence frequency in the range of R_m to R_n. As such, $P_{covered}$ represents the coverage of specific set of substructures $N_{GDB13}(R_m,\ R_n)$ of GDB-13 from the 1B generated set.

In Figures 5a and 5c, the RS and FG coverage of various models is broken down into different frequency sections to examine the coverage performance for different types of substructures. Figure 5 shows that for high frequency RS and FG, the coverage is high and quite similar among all models, while for less frequent RS and FG, the coverage reveals differences between models. On the other hand, comparing with the training set, all models demonstrate clear enrichment of RS and FG coverage, and the enrichment gets bigger as the RS and FG frequency is lower. As for RS and FG at the occurrence ranges of ">20000", "2000~20000", and "200~2000", the coverage is close to 100% for all models, while the coverage of the training dataset is around 82%, 73%, and 31% at these respective occurrence frequency ranges. As for RS at the occurrence range of "20~200", "2~20" and "≤2", most generative models have an RS coverage of around 80%, 60%, and 30%, compared to only 5%, 1%, and 0% for the training dataset. The coverage of FG at the different occurrence frequency ranges has a similar pattern to the RS coverage.

Similarly, distribution of generated RS and FG that are shared with the chemical space of GDB-13 was calculated via the formula below:

$$P_{dist} = \frac{N_{unique\_in}(R_m,\ R_n)}{N_{unique\_in}} * 100\% \,, \tag{4}$$

where $N_{unique\_in}(R_m,\ R_n)$ is the number of unique RS or FG in the sampled set that have an occurrence frequency in the range of R_m to R_n in GDB-13, and N_{unique_in} is the total number of unique RS or FG in the generated set, which are also included in GDB-13. Thus, P_{dist} is a metric of the distribution of RS or FG that are shared with GDB-13 at different occurrence ranges.

The distributions of generated RS and FG corresponding to occurrence frequency in GDB-13 are shown in Figures 5b & 5d. Given that most RS and FG have an occurrence frequency below 20 in the GDB-13 database (as shown in Figure 1), the overall coverage of RS and FG is thus dominated by ones with low occurrence frequency.

The most frequent and least frequent ring systems and functional groups sampled by the deep generative models are listed in Figures 6-9. The most often sampled ring systems are simple carbon cycles or aromatic heterocycles containing O and N atoms, such as C1CC1 (cyclopropane), which was sampled up to 78M times in the 1B sample set, and C1COC1 (oxetane), which were sampled up to 26M times in the 1B sample set. For comparison, the benzene ring ranked 85th among the most common sampled ring systems. As for the least common sampled ring systems, they were usually complex macrocycles that were only sampled once out of the 1B compounds generated.

The most commonly sampled functional groups are ordinary small ones, such as single oxygen and nitrogen atoms, C-C double bonds, and C-C triple bonds. The least commonly sampled functional groups are those with complex structures formed by a combination of simple ones. The ring systems and functional groups that are not included in GDB-13 usually do not conform to simple chemical stability and synthetic feasibility rules. It is worthwhile to mention that none of the GDB-13 compounds contain Br or F, although Br and F were obtained as functional groups up to 1.7M and 0.1M times when 1B compounds were generated by LatentGAN. This is due to the fact that LatentGAN decoder was trained on another prior dataset[11], such that LatentGAN could theoretically sample outside of applicability domain of the training set.

Most of the RS (~93%) and FG (~91%) found in the generated sets that are also found in GDB-13 are seen less than 20 times. As the results show in Figure 2, RS and FG that occur more frequently in GDB-13 tend to have smaller molecular weights. The building blocks of RS and FG are basic rings and functional groups with simple structures and small molecular weights. More complex RS and FG can be built via the combination of these basic components.

The coverage of RS and FG with an occurrence frequency in GDB-13 greater than 200 was nearly 100%. This is because these RS and FG can be easily obtained via combinations of smaller fragments. However, given that as many as up to 13 heavy atoms were considered in constructing the GDB-13 database, most RS and FG possess complex structures and were included in compounds of GDB-13 less than 20 times. RS and FG that occur less than 20 times in the generated sets dominate the coverage of the deep generative models.

Model comparison

It is interesting to observe that these models describe the chemical space so differently, although trained with the same training set. It seems that the RS and FG coverage of GraphINVENT is higher than its overall molecular coverage, one reason could be due to its large action space; that is, the number of possible "correct" sampled actions at any stage during graph generation is much larger than it is for SMILES-based methods which must use only tokens sampled in the training set. As such, given that GraphINVENT samples actions probabilistically, it is possible that sequences of actions are sampled which have never been seen in the training set, thus leading to new molecules. Another interesting observation is that GAN based models generally perform worst in terms of GDB-13 coverage on all three metrics, one reason could be due to that, in the adversarial training, the generator is supposed to mimic the true data as much as possible to fool the discriminator, which deteriorates its generalization capability to a certain extent. We also noticed that the performance of REINVENT and CharRNN is somehow similar, while their sampling speed has very large difference. This is strange given that both models are based on the same RNN architecture, suggesting that the technical implementation of CharRNN is suboptimal.

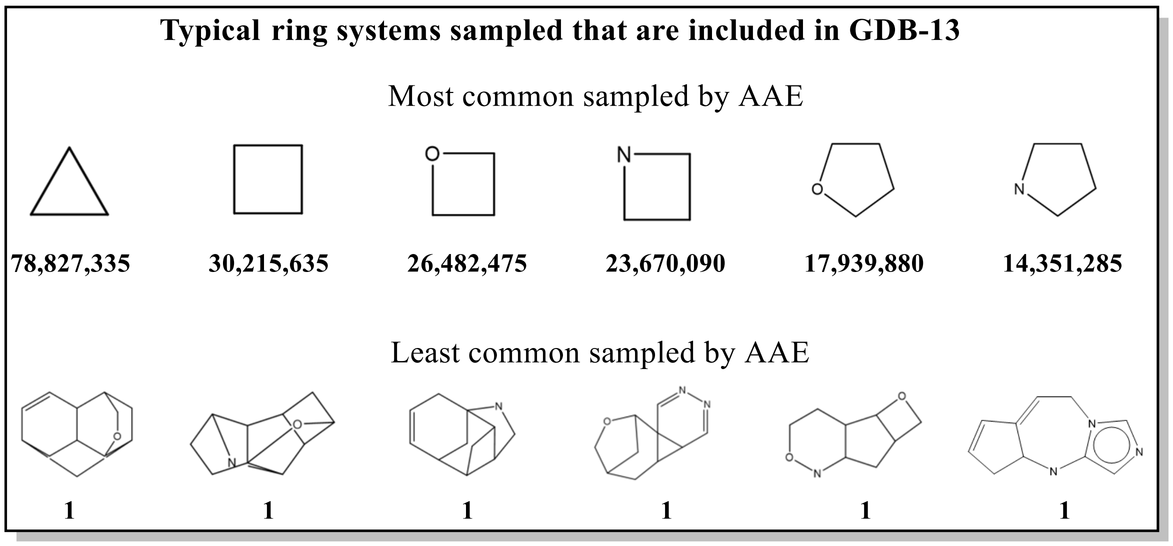
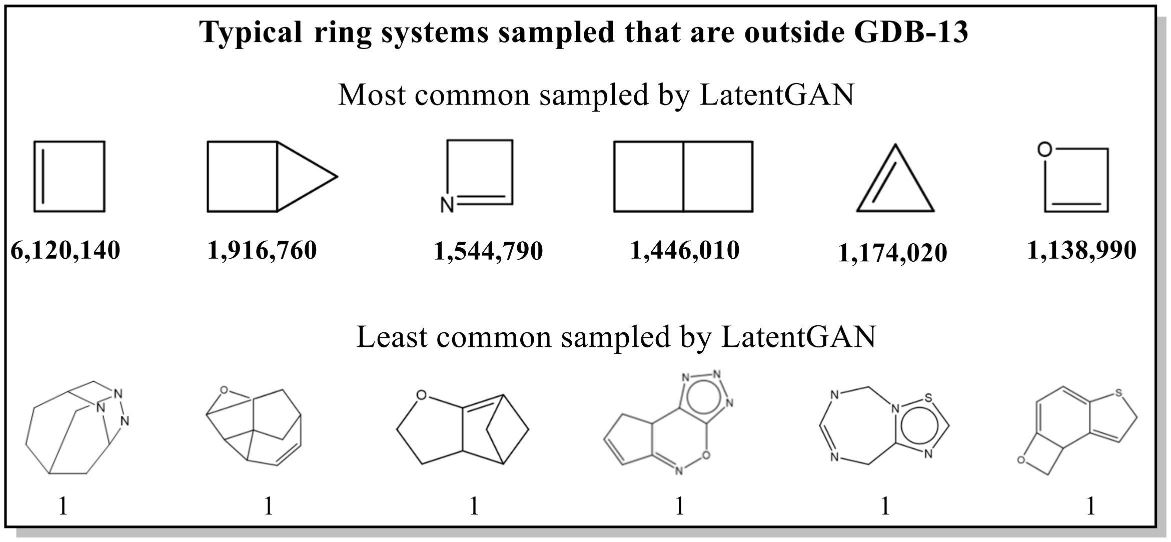
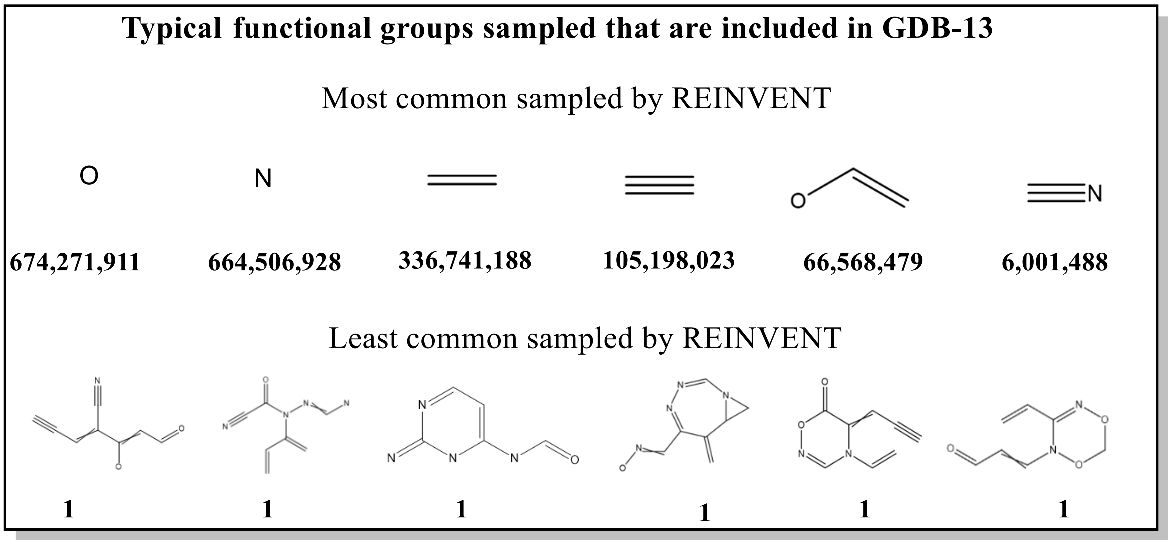

Typical ring systems sampled that are included in GDB-13
Most common sampled by AAE
78,827,335
30,215,635
26,482,475
23,670,090
17,939,880
14,351,285
Least common sampled by AAE
1
1
1
1
1
1

Figure 6. Typical ring systems that are sampled by AAE, which are included in GDB-13. The numbers below the structures in the figure are the occurrence frequency of ring systems in the 1B sampled compounds.



Typical ring systems sampled that are outside GDB-13
Most common sampled by LatentGAN
6,120,140
1,916,760
1,544,790
1,446,010
1,174,020
1,138,990
Least common sampled by LatentGAN
1
1
1
1
1
1

Figure 7. Typical ring systems that are sampled by LatentGAN, which are outside GDB-13.



Typical functional groups sampled that are included in GDB-13
Most common sampled by REINVENT
674,271,911
664,506,928
336,741,188
105,198,023
66,568,479
6,001,488
Least common sampled by REINVENT
1
1
1
1
1
1

Figure 8. Typical functional groups that are sampled by REINVENT, which are included in GDB-13.

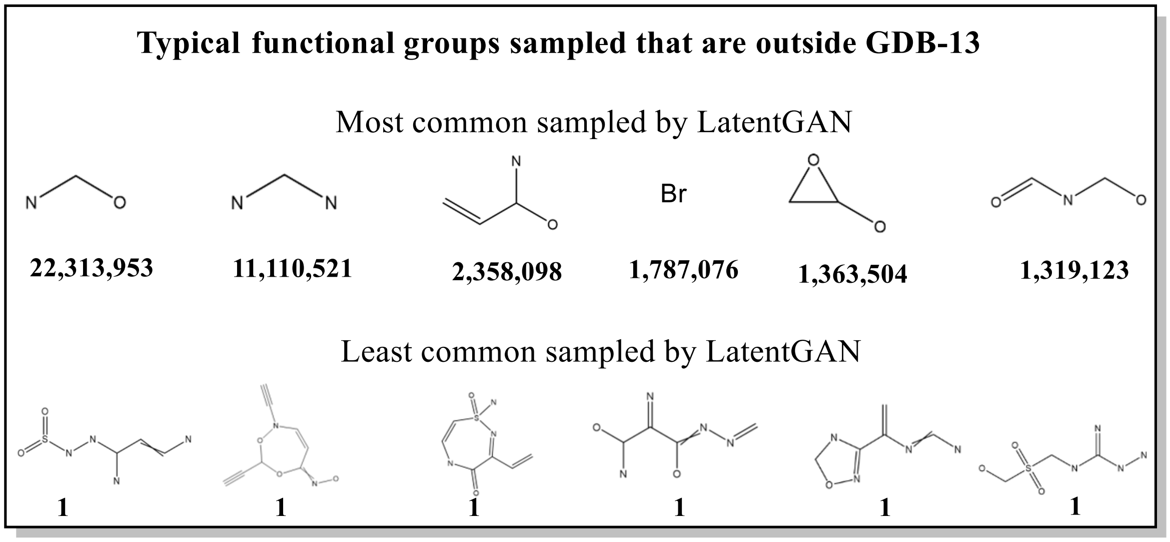

Typical functional groups sampled that are outside GDB-13
Most common sampled by LatentGAN
22,313,953
11,110,521
2,358,098
1,787,076
1,363,504
1,319,123
Least common sampled by LatentGAN
1
1
1
1
1
1

Figure 9. Typical functional groups that are sampled by LatentGAN, which are outside GDB-13.

Conclusions

Molecules consist of a variety of ring systems and functional groups, which are connected in different ways to form molecules. The most basic ring systems and functional groups have simple structures and small molecular weights; these can be found in GDB-13 molecules over dozens of times. More complex ring systems and functional groups have complicated structures and large molecular weights, and might only occur in GDB-13 a handful of times. However, due to their structural variety and enormous quantity (>90%), complex ring systems and functional groups are strong components affecting the coverage of GDB-13.

All the deep generative models studied in this work have over 100 times greater chemical space coverage for GDB-13 using 1B samples than the training set (1M) used to train the models. In terms of compound coverage of GDB-13, the best model (REINVENT) reached ~39% coverage, far beyond the coverage of LatentGAN (~13%), which ranked lowest amongst the models in this study. Depending on the generative task, the deep generative model used should thus be chosen carefully, as there are differences in how all these seemingly similar models sample the chemical space.

Associated Content

Author Information

Corresponding Author

Hongming Chen, E-mail: chen_hongming@grmh-gdl.cn

Author Contributions

J. Z. ran training and generation jobs using REINVENT, CharRNN, VAE; LatentGAN, and ORGAN. R. M. ran training and generation jobs using GraphINVENT. R. M. and J. Z. ran benchmarking calculations for this work, and J. Z. made all figures. The manuscript was written through the contributions of all authors. All authors have given approval to the final version of the manuscript.

Acknowledgements

J. Z. would like to acknowledge funding provided by XtalPi Inc, and R. M. thanks the Postdoc Program at AstraZeneca.

Supplementary Materials

The detailed hyperparameters and training loss curves of all models can be found in supplementary materials. The training, sampling, and analysis script could found in the GitHub repository, https://github.com/jeah-z/Generative_Models_benchmark_gdb13.

Abbreviations

RS, Ring system(s); FG, Functional group(s); GAN, Generative adversarial network; GNN, graph neural network; RNN, recurrent neural network.

References

[1] Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification; proceedings of the 2012 IEEE conference on computer vision and pattern recognition, F, 2012 [C]. IEEE.

[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks; proceedings of the Advances in neural information processing systems, F, 2012 [C].

[3] Taigman Y, Yang M, Ranzato M A, Wolf L. Deepface: Closing the gap to human-level performance in face verification; proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, F, 2014 [C].

[4] Silver D, Huang A, Maddison C J, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M. Mastering the game of Go with deep neural networks and tree search [J]. nature, 2016, 529(7587): 484-9.

[5] Hadjeres G, Pachet F, Nielsen F. Deepbach: a steerable model for bach chorales generation; proceedings of the International Conference on Machine Learning, F, 2017 [C]. PMLR.

[6] Garg S, Rish I, Cecchi G, Lozano A. Neurogenesis-inspired dictionary learning: Online model adaption in a changing world [J]. arXiv preprint arXiv:170106106, 2017.

[7] Johnson M, Schuster M, Le Q V, Krikun M, Wu Y, Chen Z, Thorat N, Viégas F, Wattenberg M, Corrado G. Google's multilingual neural machine translation system: Enabling zero-shot translation [J]. Transactions of the Association for Computational Linguistics, 2017, 5(339-51.

[8] Bjerrum E J, Threlfall R. Molecular generation with recurrent neural networks (RNNs) [J]. arXiv preprint arXiv:170504612, 2017.

[9] Kotsias P-C, Arús-Pous J, Chen H, Engkvist O, Tyrchan C, Bjerrum E J. Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks [J]. Nature Machine Intelligence, 2020, 2(5): 254-65.

[10] Polykovskiy D, Zhebrak A, Sanchez Lengeling B, Golovanov S, Tatanov O, Belyaev S, Kurbanov R, Artamonov A, Aladinskiy V, Veselov M, Kadurin A, Johansson S, Chen H, Nikolenko S I, Aspuru-Guzik A, Zhavoronkov A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models [J]. Frontiers in Pharmacology, 2020, DOI: 10.3389/fphar.2020.565644 (Accepted).

[11] Prykhodko O, Johansson S V, Kotsias P-C, Arús-Pous J, Bjerrum E J, Engkvist O, Chen H. A de novo molecular generation method using latent vector based generative adversarial network [J]. Journal of Cheminformatics, 2019, 11(1): 74.

[12] Schneider G, Fechner U. Computer-based de novo design of drug-like molecules [J]. Nat Rev Drug Discov, 2005, 4(8): 649-63.

[13] DiMasi J A, Grabowski H G, Hansen R W. Innovation in the pharmaceutical industry: new estimates of R&D costs [J]. Journal of health economics, 2016, 47(20-33.

[14] Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T. The rise of deep learning in drug discovery [J]. Drug discovery today, 2018, 23(6): 1241-50.

[15] Blaschke T, Arús-Pous J, Chen H, Margreitter C, Tyrchan C, Engkvist O, Papadopoulos K, Patronov A. REINVENT 2.0–an AI tool for de novo drug design [J]. 2020,

[16] Blaschke T, Olivecrona M, Engkvist O, Bajorath J, Chen H. Application of generative autoencoder in de novo molecular design [J]. Molecular informatics, 2018, 37(1-2): 1700123.

[17] Guimaraes G L, Sanchez-Lengeling B, Outeiral C, Farias P L C, Aspuru-Guzik A. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models [J]. arXiv preprint arXiv:170510843, 2017.

[18] Kadurin A, Aliper A, Kazennov A, Mamoshina P, Vanhaelen Q, Khrabrov K, Zhavoronkov A. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology [J]. Oncotarget, 2017, 8(7): 10883.

[19] Brown N, Fiscato M, Segler M H, Vaucher A C. GuacaMol: benchmarking models for de novo molecular design [J]. Journal of chemical information and modeling, 2019, 59(3): 1096-108.

[20] Arus-Pous J, Blaschke T, Ulander S, Reymond J L, Chen H, Engkvist O. Exploring the GDB-13 chemical space using deep generative models [J]. J Cheminform, 2019, 11(1): 20.

[21] Blum L C, Reymond J-L. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13 [J]. Journal of the American Chemical Society, 2009, 131(25): 8732-3.

[22] Ertl P, Altmann E, McKenna J M. The Most Common Functional Groups in Bioactive Molecules and How Their Popularity Has Evolved over Time [J]. Journal of Medicinal Chemistry, 2020, 63(15): 8408-18.

[23] Taylor R D, MacCoss M, Lawson A D. Rings in drugs [J]. J Med Chem, 2014, 57(14): 5845-59.

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

[25] Segler M H, Kogej T, Tyrchan C, Waller M P. Generating focused molecule libraries for drug discovery with recurrent neural networks [J]. ACS central science, 2018, 4(1): 120-31.

[26] Gómez-Bombarelli R, Wei J N, Duvenaud D, Hernández-Lobato J M, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel T D, Adams R P, Aspuru-Guzik A. Automatic chemical design using a data-driven continuous representation of molecules [J]. ACS central science, 2018, 4(2): 268-76.

[27] Makhzani A, Shlens J, Jaitly N, Goodfellow I, Frey B. Adversarial autoencoders [J]. arXiv preprint arXiv:151105644, 2015.

[28] Mercado R, Rastemo T, Lindelöf E, Klambauer G, Engkvist O, Chen H, Bjerrum E J. Practical Notes on Building Molecular Graph Generative Models [J]. ChemRxiv, 2020, (Preprint).

[29] Landrum G. RDKit: Open-source cheminformatics [J]. 2006.

[30] Ertl P. An algorithm to identify functional groups in organic molecules [J]. Journal of cheminformatics, 2017, 9(1): 1-7.

[31] Arus-Pous J. reinvent-randomized [M]. GitHub repository [https://githubcom /undeadpixel/reinvent-randomized], Github, 2020.

[32] Polykovskiy D, Zhebrak A, Sanchez-Lengeling B, Golovanov S, Tatanov O, Belyaev S, Kurbanov R, Artamonov A, Aladinskiy V, Veselov M. moses [M]. Github, Github repository [https://github.com/molecularsets/moses], 2020.

[33] Johansson S, Prykhodko O. latent-gan [M]. Github repository [https://githubcom/ Dierme/latent-gan], Github, 2019.

[34] Mercado R, Rastemo T, Lindelöf E. GraphINVENT [M]. GitHub repository [https://githubcom/MolecularAI/GraphINVENT/], Github, 2020.

[35] Rocío M, Tobias R, Edvard L, Günter K, Ola E, Hongming C, Esben Jannik B. Graph Networks for Molecular Design [J]. ChemRxiv, 2020, (Preprint).

[36] Oliphant T E. Python for scientific computing [J]. Computing in Science & Engineering, 2007, 9(3): 10-20.

[37] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L. Pytorch: An imperative style, high-performance deep learning library; proceedings of the Advances in neural information processing systems, F, 2019 [C].

[38] Panaretos V M, Zemel Y. Statistical aspects of Wasserstein distances [J]. Annual review of statistics and its application, 2019, 6(405-31.

[39] Virtanen P, Gommers R, Oliphant T E, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J. SciPy 1.0: fundamental algorithms for scientific computing in Python [J]. Nature methods, 2020, 17(3): 261-72.

Supporting Information

Table S1. Source codes for studied generative models.

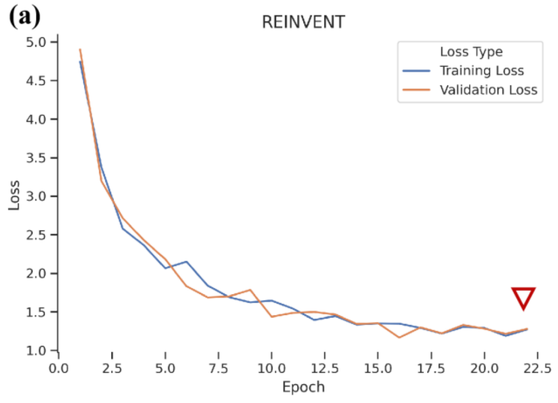
Model	GitHub repository
REINVENT	https://github.com/undeadpixel/reinvent-randomized
CharRNN	https://github.com/molecularsets/moses/tree/master/moses/char_rnn
AAE	https://github.com/molecularsets/moses/blob/master/moses/aae
ORGAN	https://github.com/molecularsets/moses/tree/master/moses/organ
LatentGAN	https://github.com/Dierme/latent-gan
VAE	https://github.com/molecularsets/moses/tree/master/moses/vae
GraphINVENT	https://github.com/MolecularAI/GraphINVENT

Table S2. Hyperparameters of generative models in this study.

Model	Parameters & hyperparameters	Sampling epoch
REINVENT	num-layers = 3； layer-size = 512； embedding-layer-size = 256； dropout = 0; layer-normalization = False; max-sequence-length = 256; learning-rate-mode = exp; learning-rate-start = 1E-4; learning-rate-min = 1E-5; learning-rate-gamma = 0.8; learning-rate-threshold = 1E-4; learning-rate-average-steps = 1; learning-rate-patience = 8; batch-size = 128 (training); batch-size = 12,800 (sampling);	22
CharRNN	num_layers=3; hidden=768; dropout=0.2; n_batch=64; lr=1E-3; step_size=10; gamma=0.5; n_jobs=1; n_workers=1; max_len=100; batch-size = 25,000 (sampling);	20

AAE	embedding_size=32; encoder_hidden_size=512; encoder_num_layers=1; encoder_bidirectional=True; encoder_dropout=0; decoder_hidden_size=512; decoder_num_layers=2; decoder_dropout=0; latent_size=128; discriminator_layers=[640, 256]; n_batch=512(training); lr=1E-3; step_size=20; gamma=0.5; n_jobs=1; n_workers=1; discriminator_steps=1; weight_decay=0	20
ORGAN	embedding_size=32; hidden_size=512; num_layers=2; dropout=0; discriminator_layers= [(100, 1), (200, 2), (200, 3), (200, 4), (200, 5), (100, 6), (100, 7), (100, 8), (100, 9), (100, 10), (160, 15), (160, 20)]; discriminator_dropout=0; reward_weight=0.7; generator_pretrain_epochs=50; discriminator_pretrain_epochs=50; pg_iters=1000; n_batch=64; lr=1E-4; n_jobs=8; n_workers=1; max_length=100; clip_grad=5; rollouts=16; generator_updates=1; discriminator_updates=1; discriminator_epochs=10; pg_smooth_const=0.1; n_ref_subsample=500; additional_rewards= MetricsReward.supported_metrics; batch-size = 25,000 (sampling);	50
LatentGAN	encode="chembl"; batch-size=64; n-critic-number=5; lr=2E-4; b1=0.5; b2=0.9; sample-after-training=False; save-interval=1; number-samples=50,000; decode-sampled=True;	1990
VAE	q_d_h=256; q_n_layers=1; q_dropout=0.5; d_cell='gru'; d_n_layers=3; d_dropout=0; d_z=128; d_d_h=512; freeze_embeddings=False; n_batch=512; clip_grad=50; kl_start=0; kl_w_start=0; kl_w_end=0; lr_start=3E-4; lr_n_period=10; lr_n_restarts=10; lr_n_mult=1; lr_end=3E-4; n_last=1,000; n_jobs=1; n_workers=1;	99
GraphINVENT	activation_function=SELU; batch_size=1000; block_size=100,000; *_dropout_p=0.0; group_size=1000; gru_bias=True; hidden_node_features=100; init_lr=1e-4; lrdf=0.9999; lrdi=10,000; message_passes=3; message_size=100; min_rel_lr=5e-2; mlp_bias=True; *_depth=4; *_hidden_dim=500; ramp_up_lr=False; tune_lr=True; weight_decay=0.0; weights_initialization=uniform; gather_width=100	20

Training loss:

(a)


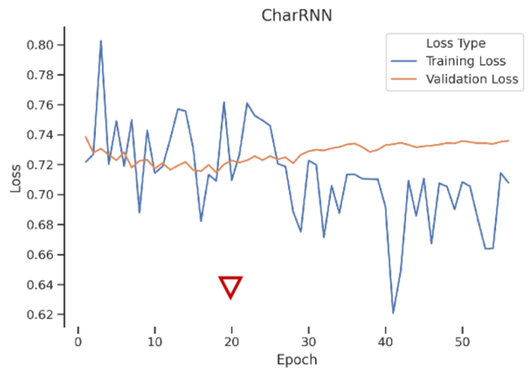
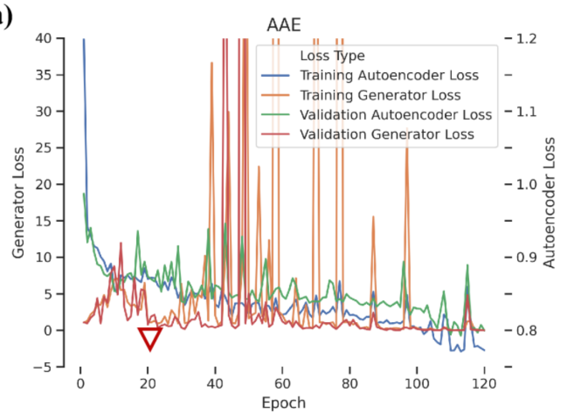
(b)


Fig. S1 Losses of generative models during training. (a) REINVENT. (b) CharRNN. The saved model used for sampling is denoted with symbol ∇ in each figure.

(a)


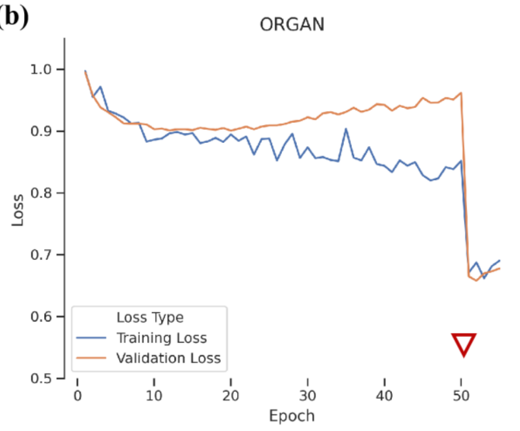
(b)


Fig. S2 Losses of generative models during training. (a) AAE. (b) ORGAN.

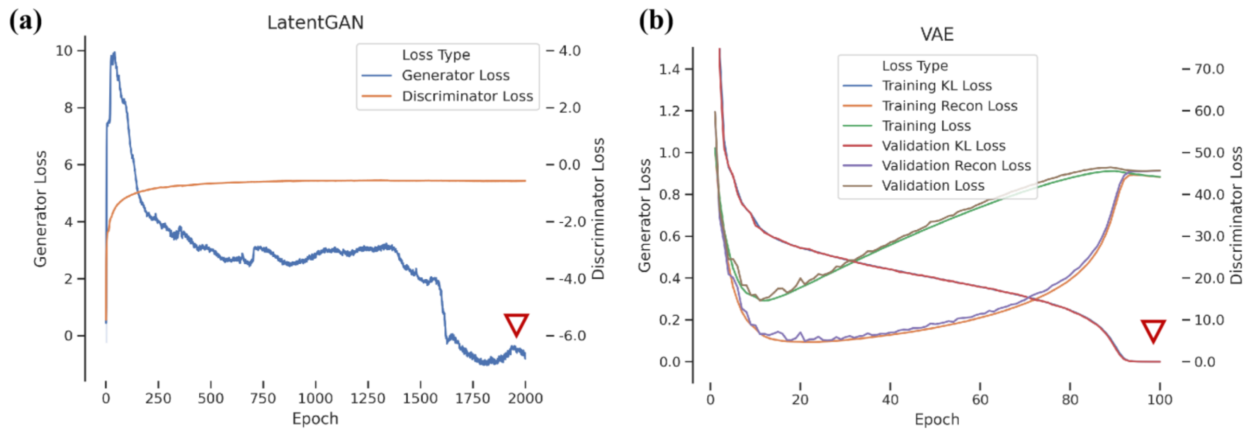


Fig. S3 Losses of generative models during training. (a) LatentGAN. (b) VAE.

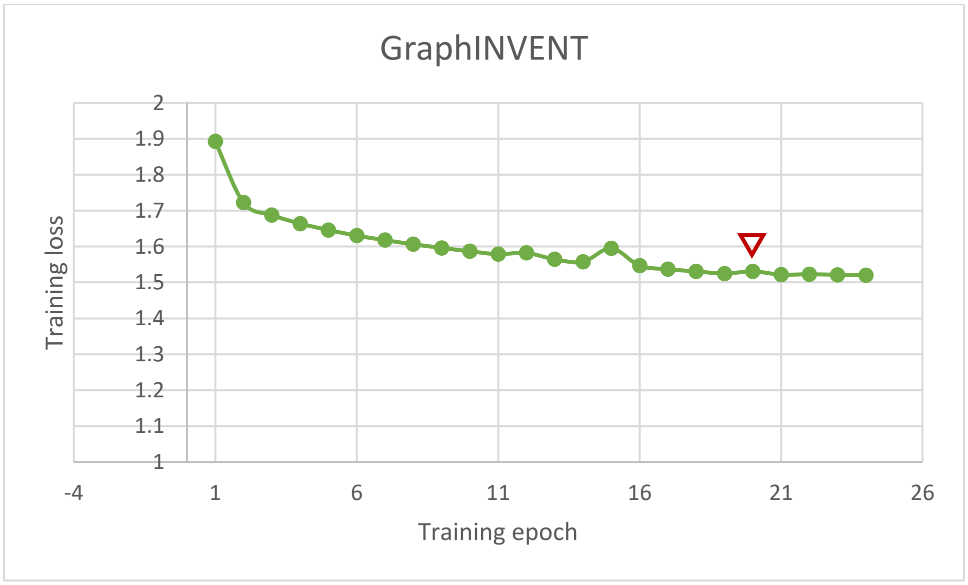


Fig. S4. Loss of generative model GraphINVENT during training.