Generative Model for Drug-like Small Molecules using SeqGAN

Xuyang Steven Zhou

E-mail: zhouxuyang@hsefz.cn

Accepted for Publication: 2023
Published Date: October 2023

Abstract

Nowadays, generative models can be used for molecule generation in current drug discovery process. Among them, the family of GAN (Generative Adversarial Network) models are widely used for generative works. In this research, we tried to apply SeqGAN to generate SMILES strings of drug-like molecules, which uses Reinforcement Learning in generator training to bypass the problem that discrete sequences are not differentiable. The model went through 10 epochs of pre-training and 8 epochs of RL adversarial training. After pre-training, the decoding rate of generator is up to 18%, and the generated distribution is also close to the real distribution. But the adversarial training process was a failure, with the output model can only generate "olefin sticks". We attribute this to the too harsh penalties for generating invalid SMILES strings.

Keywords: SeqGAN, Reinforcement learning, Molecules generating

1.0 INTRODUCTION

Drug discovery initiates because there are medical needs, to start a program the first step is to develop target identification and validation, this step is to determine our problem we are going to solve, our medicine needs to have certain effect on the target. HTS (High-throughput Screening) techniques can screen many potential compounds in a short period of time, becoming the dominant way of drug development today. The effectiveness of discovering lead compounds through high-throughput drug screening depends on the number and quality of compounds in the compound sample library. Since Hit is simply a potentially active compound, but may be not sufficiently active, disorganized, and possibly toxic. Next step is to optimize lead compounds and identify drug candidates, pharmacological chemists will extend the structure of the emerging compound, and if a series of similarly structured compounds all exhibit activity on the selected target, we call the initially screened compound Lead. A lead compound is a compound with some biological activity and chemical structure. Next, the research and development of new drugs enters the development stage, and the primary goal is to complete the preclinical toxicology re-search and submit the application for experimental new drugs to the drug regulatory department. To carry out these studies requires many APIs (Active Pharmaceutical Ingredient), so the first step is the development of the API synthesis process, which requires continuous improvement and refinement. This is followed by

pharmacokinetic studies and safety pharmacological evaluations to guide clinical administration forms, doses, and evaluation of side effects. As the project progresses, it is necessary to continuously improve the mode of administration and the prescription of preparations. Then Medicinal chemists will finally come to clinical research. Compound library is a collection of entity compounds with specific structure or function and their related information under a specific standard. It is usually used as a tool for new drug discovery or cell induction to help scientists find the emerging compounds of new drug research and development or the implementers of specific functions of cells. And due to the inefficiency of high-throughput screening and the inadequacy of the molecular library of compounds that have been increasingly discovered with the development of medical treatment, The lack of compound library will lead to the inability of pharmacists to effectively complete the discovery of drug lead compounds, which is likely to lead to the slow development process in the case of urgent drug achievements. People can no longer rely on the accidental discovery of new compound components as before. SeqGAN model can be used to extend compound library. SeqGAN is first pro-posed because GAN model can't be applied in generative tasks for sequences that include discrete tokens because the Gan model needs to calculate the derivative of the sequence to complete the gradient transfer, and the adjustment of the reward function is completed through the real-time gradient transfer. The discrete non derivative data makes the Gan model unable to complete the gradient transfer, resulting in the inability of training. The GAN model was originally pro-posed in Generative Adversarial Nets in 2014 by Prof. Ian J. Goodfellow and his team. As an un-supervised learning method, GAN is composed of two independent neural networks, generator G and discriminator D. The generator G is used to generate fake samples, and the discriminator D is used to distinguish whether the samples generated by the generator G. G and D are trained together. The training continues until the two have entered a state of balance and harmony. The goal of the GAN model is to obtain a high-quality automatic generator. In GAN, Generator starts with random sampling and then performs a deterministic transformation based on the parameters of the model. Through the output of generator model G, discriminative model D calculates the loss value, and according to the resulting loss gradient, the generator model G is in-structed to make slight changes, so that G pro-duces more realistic data. GAN model also has two problems. In text generation tasks, G usually adopts RNN (Recurrent Neural Net-work) architecture, then what G passes to D is a bunch of discrete value sequences, that is, the output of each LSTM unit passes through soft-max and then takes argmax or samples a specific word based on probability, which makes gradient removal impossible to handle. Secondly, GAN can only evaluate the score/loss of the entire generated sequence and cannot be refined to evaluate the quality of the currently generated token and the impact on subsequent generation. RL (Reinforcement learning) can solve the above two points very well. Policy gradient is one of the training frameworks for decision makers in RL. The basic idea of Policy Gradient is to use reward as feedback to increase the probability of an action with a large reward and reduce the probability of an action with a small reward. If we have a reward, we can perform gradient training and up-date parameters. Using the Policy Gradient algorithm, when G generates a molecule, if we can get a feedback Reward, we can use this reward to update the parameters of G, and we no longer need to rely on the back propagation of D to up-date the parameters, so it can solve the first problem. For the second question, when a word is generated, we can use the Monte Carlo tree search to evaluate the quality of the current word, without waiting until the end of the entire sequence to evaluate the quality of the word. Therefore, the combination of reinforcement learning, and adversarial ideas can theoretically solve the problem of discrete sequence generation, and the SeqGAN model is a model that can be used for text sequence generation due to the collision of these two ideas. As far as the results are concerned, the training of the SeqGAN model in this research project was not successful, and only the pre-training was in line with expectations. Due to the difficulty of training the SeqGAN model and the lack of time and funds, part of the intensive RL training was not effective in my study.

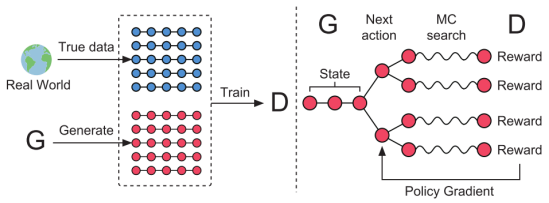

FIGURE 1. Basic concept of SeqGAN

As shown in the figure1, the real data is add-ed to the generated data of G to train D. But from the content described in the previous background chapter, we can know the discrete output of G, making it impossible for D to return a gradient to update G, so some changes need to be made, see the figure1 (right part), the policy network in the paper As G, the existing red dot is called the cur-rent state, and the next red dot to be generated can be viewed as action, because D needs to score a complete sequence, so MCTS (Monte Carlo tree search) to complete the various possibilities of

each action, D generates rewards for these complete sequences, returns to G, and updates G through reinforcement learning. The training process will alternate on G and D iteratively, until the model converges, where it can be expected that G will generate molecules so close to real ones that D cannot tell them apart.

METHODS

The data set used in this experiment is from the public data set ZINC, which contains about 750 million molecules. RDkit is used to read and write molecules, and the program is interpreted by Python. Before adversarial, training there are pre-training procedures for both generator and discriminator which makes our model converge faster. For Generator, the loss functions used in the pre-training and adversarial processes are different. In the pre-training process, the Generator uses the cross-entropy loss function, and in the adversarial process. For Discriminator, the loss function in both processes is the same.

$$J(\theta) = E\left[s_0, \theta\right] = \sum_{y_1 \in Y} G_\theta\left(s_0\right) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

EQUATION 1. Reward function of SeqGAN

Here the expectation of this reward is defined by $J_{(\theta)}$. It is the expectation of generating a reward of a complete sequence under the conditions of s_0 and θ. The G_θ part is the Generator Model. And $Q_{D_\phi}^{G_\theta}$ is an action-value function of a sequence in the text.

$$Q_{D_\phi}^{G_\theta}\left(a = y_T, s = Y_{1:T-1}\right) = D_\phi\left(Y_{1:T}\right)$$

EQUATION 2. Action value function

It's hard to obtain precise action-value function $Q_{D_\phi}^{G_\theta}$, but we can estimate it through a MC manner. Because the return value produced by an incomplete sequence has no

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t}, a = y_t) = \{\frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n), Y_{1:T}^n \in MC^{G_\beta}(Y_1$$

EQUATION 3. Different action value calculation methods in different steps

practical significance, the Q value of the resulting y_t cannot be calculated directly after the y_t is generated in the case of the original y_t to y_{t-1}, unless the y_t is the last of the entire sequence. The study then used Monte Carlo searches to complete the y_t. In this study, all possible sequences completed by using Monte Carlo search after the same y_t were calculated as rewards and then averaged.

$$\nabla_\theta J(\theta) = E_{Y_{1:t-1} \sim G_\theta}\left[\sum_{y_t \in Y} \nabla_\theta G_\theta\left(y_t | Y_{1:t-1}\right) \cdot Q_{D_\phi}^{G_\theta}\left(Y_{1:t-1}, y_t\right)\right]$$

EQUATION 4. Policy Gradient

During this test the result is obtained when the parameters in the program are "d_filter_size":20,"d_num_filters":3, "g_ltent_size":20 ,"g_hiden_size":20,"lambda":0.2, "alpha":0.03,"pretrain_g_epoches":15,"pretrain_d_epoches":10,"epoches":100,"g_steps":1,"d_steps":5,"num_rollout":20,"batch_size":50,"lr": 1e-3.

In algorithm1 describes SeqGAN model. First, initialize the G network and D network parameters randomly. The G network is pre-trained by MLE in order to improve the search efficiency of the G network. Use the pretrained G to generate some data to pretrain D by minimizing the cross-entropy.

Algorithm1. Sequence Generative Adversarial Network

Require: generator policy ; roll-out policy
;discriminator ; a sequence dataset S= {}

Initialize , with random weights ,
Pre-train using MLE on S
β«θ
Generate negative samples using for training
Pre-train via minimizing the cross entropy
Repeat
For g-steps do
Generate a sequence = ()~
For t in 1: T do
Compute Q (a=; s=)
End for
Update generator parameters via policy gradient
End for
For d-step do
Use current to generate negative examples and combine with given positive examples S
Train discriminator for 1 epoch
End for
β«θ
until G and D converge.

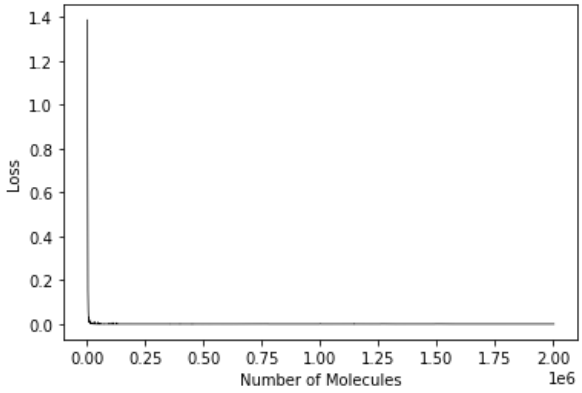

FIGURE 3. Loss curve in pre-train D

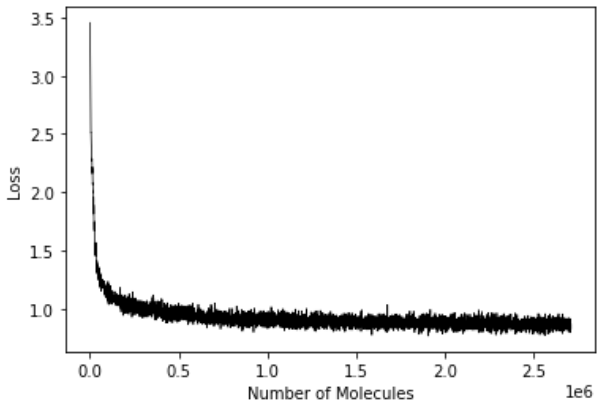

FIGURE 4. Loss curve in pre-train G

2.0 RESULT & DISCUSSION

In the end, I completed the project with 13 rounds of generator pre-training, 10 rounds of discriminator pre-training, and finally 8 rounds of reinforcement learning training. (figure3 &4 are preliminary results)

Figure 8 is the loss function of discriminator in pretraining and the picture on the right is the loss function of generator. At the same time, in the pre-training, the loss function converges very obviously, in about 1500000 molecules the loss is converged, and good results are obtained.

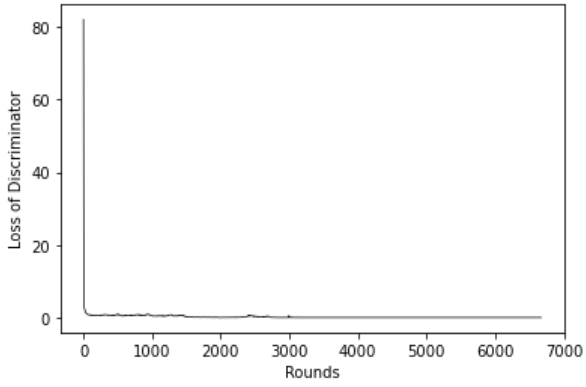

FIGURE 5. Loss curve of Discriminator in RL training

The loss function of reinforcement learning is very unstable, the generated molecules also have more small molecules, and the generated molecules are not stable enough. For example, in RL training generate molecules in figure6 the first one of line 3 is unstable because although this structure can exist in the air, it is easy to hydrolyze and lose its effect in liquids such as human gastric juice.

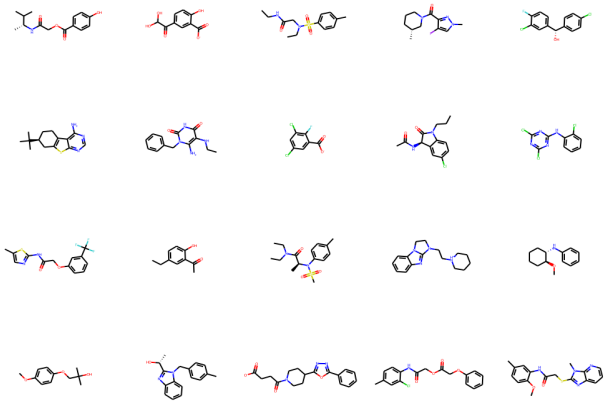

FIGURE 6. Real molecules samples

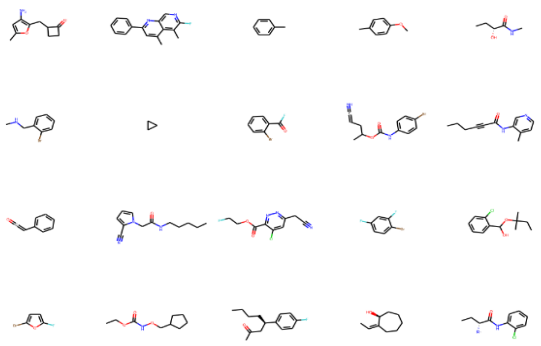

FIGURE 7. RL training generate molecules

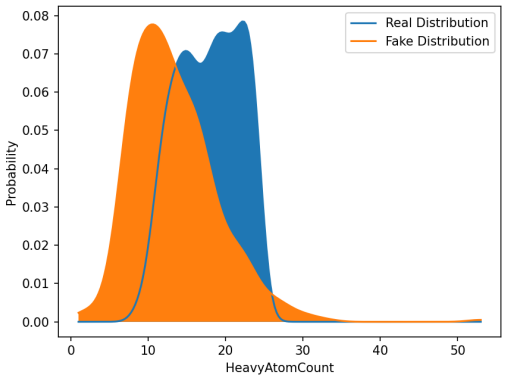

FIGURE 8. Heavy Atom Count real and fake distribution

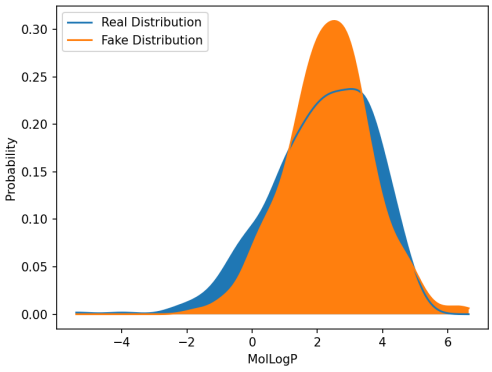

FIGURE 9. oil-water partition coefficient distribution

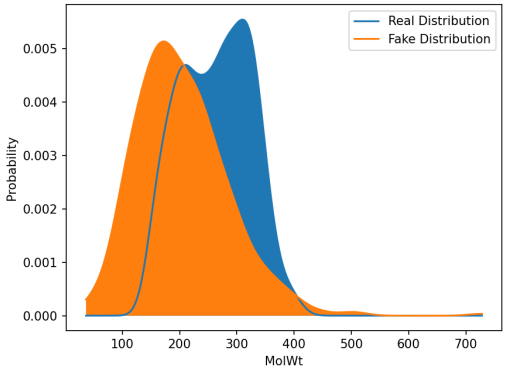

FIGURE 10. Molecular weight distribution

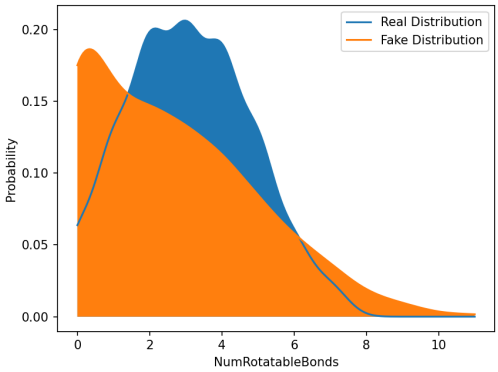

FIGURE 11. Number of molecular rotatable bonds distribution

From figures 8 to 11, it is discovered that the average fake distribution is lower than real distribution, the fake distribution trend in figure 9 is similar to the real trend, which determined the water solubility of generated molecules are close to the real molecules. Lower fake distribution in figures 8 to 11 cause generator more likely to generate small rigid molecules.

In this research, after RDkit solve the molecules, successful decoded molecules can turn into generated molecules, and the decoding rate in 0124_temp is 18 %.

3.0 CONCLUSION

In this research, I spent about two months on research, including programming and tuning parameters, and after 8 rounds of reinforcement learning training, I got relatively normal pre-training data and failed reinforcement learning training Data,

the reasons for the failure are the difficulty of training the SeqGAN model due to the content of machine learning, and the lack of time, because it often takes nearly a week to perform a round of training of the SeqGAN model, which is left for me to adjust the parameters. There is not enough time and opportunity for trial and error, which leads to the fact that the molecules generated by reinforcement learning are mostly small molecules and the molecular structure is unstable. This may be because the failure penalty I gave the program was too strict, which caused the program to focus on generating legal SMILES strings. Thus avoiding penalties and ignoring the utility of generating molecules.

4.0 ACKNOWLEDGMENT

This work was carried out by the Yuan Pei Official organizing committee. I gratefully acknowledge their invaluable cooperation in preparing this application note.

5.0 REFERENCES

[1] Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu; SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient *arXiv:1609.05473v6* 25 Aug **2017**