Molecule Design by Latent Space Energy-Based Modeling and Gradual Distribution Shifting

Abstract

Generation of molecules with desired chemical and biological properties such as high drug-likeness, high binding affinity to target proteins, is critical for drug discovery. In this paper, we propose a probabilistic generative model to capture the joint distribution of molecules and their properties. Our model assumes an energy-based model (EBM) in the latent space. Conditional on the latent vector, the molecule and its properties are modeled by a molecule generation model and a property regression model respectively. To search for molecules with desired properties, we propose a sampling with gradual distribution shifting (SGDS) algorithm, so that after learning the model initially on the training data of existing molecules and their properties, the proposed algorithm gradually shifts the model distribution towards the region supported by molecules with desired values of properties. Our experiments show that our method achieves competitive performances on various molecule design tasks.

1 INTRODUCTION

In drug discovery, it is of vital importance to find or design molecules with desired pharmacologic or chemical properties such as high drug-likeness and binding affinity to a target protein. It is challenging to directly optimize or search over the drug-like molecule space since it is discrete and enormous, with an estimated size on the order of 10^{33} [Polishchuk et al., 2013].

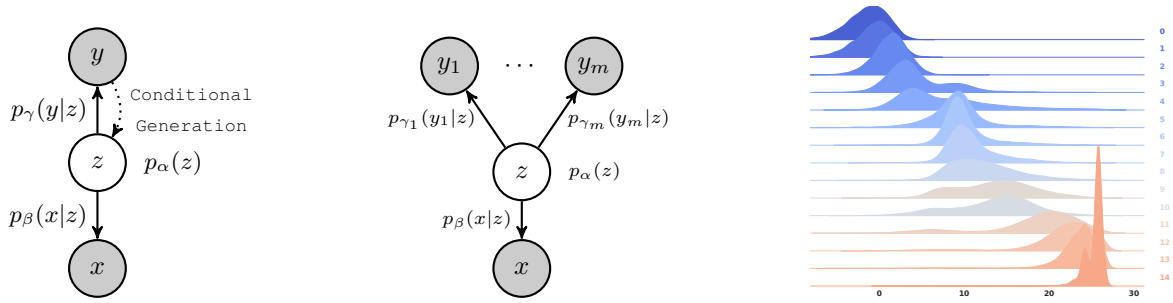
Recently, a large body of work attempts to tackle this problem. The first line of work leverages deep generative models to map the discrete molecule space to a continuous latent space, and optimizes molecular properties in the latent space with methods such as Bayesian optimization [Gómez-Bombarelli et al., 2018, Kusner et al., 2017, Jin et al., 2018]. The second line of work recruits reinforcement learning algorithms to optimize properties in the molecular graph space directly [You et al., 2018, De Cao and Kipf, 2018, Zhou et al., 2019, Shi et al., 2020, Luo et al., 2021]. A number of other methods have been proposed to optimize molecular properties with genetic algorithms [Nigam et al., 2020], particle-swarm algorithms [Winter et al., 2019], and specialized MCMC methods [Xie et al., 2021].

In this work, we propose a method along the first line mentioned above, by learning a probabilistic latent space generative model of molecules and optimizing molecular properties in the latent space. Given the central role of latent variables in this approach, we emphasize that it is critical to learn a latent space model that captures the data regularities of the molecules. Thus, instead of assuming a simple Gaussian distribution in the latent space as in prior work [Gómez-Bombarelli et al., 2018, Jin et al., 2018], we assume a flexible and expressive energy-based model (EBM) [LeCun et al., 2006, Ngiam et al., 2011, Kim and Bengio, 2016, Xie et al., 2016, Kumar et al., 2019, Nijkamp et al., 2019, Du and Mordatch, 2019, Grathwohl et al., 2019, Finn et al., 2016] in latent space. This leads to a *latent space energy-based model* (LSEBM) as studied in Pang et al. [2020], Nie et al. [2021], where LSEBM has been shown to model the distributions of natural images and text well.

Going beyond existing latent space energy-based models Pang et al. [2020], Nie et al. [2021], our work makes two innovations:

First, given our goal of property optimization, we learn a joint distribution of molecules and their properties. Our model consists of (1) an energy-based model (EBM) in a low-dimensional continuous latent space, (2) a molecule generation model that generates molecule given the latent vector, and (3) a property regression model that predicts the value of the property given the latent vector. See Figure 1a for an illustration of the model. We first learn the initial model on the training data that consist of existing molecules

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.



(a) Single-Objective Optimization. (b) Multi-Objective Optimization. (c) Shift of distribution of y.

Figure 1: An illustration of the joint distribution of molecule with its single property (a) or multiple properties (b). x represents a molecule, z is the latent vector, y is a molecular property of interest, $\{y_j\}_{j=1}^m$ indicates m properties. (c) illustrates the shift of the distribution of a single property in sampling with gradual distribution shifting (SGDS).

and their properties. All three components in our model are learned jointly by an approximate maximum likelihood algorithm.

Second, and more importantly, we propose a *sampling with gradual distribution shifting* (SGDS) method for molecule design. We first sample molecules and their property values from the initial model learned on the training data mentioned above. Then we gradually shift the joint distribution towards the region supported by molecules with high property values. Specifically, our method iterates the following steps. (1) Shift the sampled property values by a small constant towards the desired target value. (2) Generate molecules given the shifted property values. (3) Obtain the ground-truth property values of the generated molecules by querying the software. (4) Update the model parameters by learning from the generated molecules and their ground-truth property values. Because of the flexibility of the latent space energy-based model, the model can be updated to account for the change of the joint distribution of the generated molecules and their ground-truth property values in the gradual shifting process. Figure 1c illustrates the shifting of the distribution of the property values of the generated molecules.

In drug discovery, most often we need to consider multiple properties simultaneously. Our model can be extended to this setting straightforwardly. With our method, we only need to add a regression model for each property, while the learning and sampling methods remain the same. (see Figure 1c). We can then simultaneously shift the values of the multiple properties for multi-objective optimization.

We evaluate our method in various settings including single-objective and multi-objective optimization. Our method outperforms prior methods by significant margins.

In summary, our contributions are as follows:

- We propose to learn a latent space energy-based model for the joint distribution of molecules and their properties.
- We develop a sampling with gradual distribution shifting method, which enables us to extrapolate the data distribution and sample from the region supported by molecules with high property values.
- Our methods are versatile enough to be extended to optimizing multiple properties simultaneously.
- Our model achieves state-of-the-art performances on a range of molecule optimization tasks.

Caveat. As in most existing work on molecule design, we assume that the value of a property of interest of a given molecule can be obtained by querying an existing software. There are two research problems in this endeavor. (1) Developing software that can output biologically or chemically accurate value of the property for an input molecule. (2) Developing method that can optimize the property values output by a given software. While problem (1) is critically important, our work is exclusively about problem (2). We duly acknowledge that existing software may need much improvements. Meanwhile our method can be readily applied to the improved versions of software.

2 RELATED WORK

Optimization with Generative Models. Deep generative models approximate the distribution of molecules with desired biological or non-biological properties. Existing approaches for generating molecules include applying variational autoencoder (VAE) [Kingma and Welling, 2014] and generative adversarial network (GAN) [Goodfellow et al., 2014] etc. to molecule data [Gómez-Bombarelli et al., 2018, Jin et al., 2018, De Cao and Kipf, 2018, Honda et al., 2019, Madhawa et al., 2019, Shi et al., 2020, Zang and Wang, 2020, Kotsias et al., 2020, Chen et al., 2021, Fu et al., 2020, Liu et al., 2021, Bagal et al., 2021, Eckmann et al., 2022, Segler et al., 2018]. After learning continuous representations for molecules, they are further able to optimize using different methods. [Segler et al., 2018] proposes to optimize by simulating design-synthesis-test cycles. [Gómez-Bombarelli et al., 2018, Jin et al., 2018, Eckmann et al., 2022] propose to learn a surrogate function to predict properties, and then

use Bayesian optimization to optimize the latent vectors. However, the performance of this latent optimization is not satisfactory due to three major issues. First, it is difficult to train an accurate surrogate predictor especially for those novel molecules with high properties along the design trajectories. Second, as the learned latent space tries to cover the fixed data space, its ability to explore the targets out of the distribution is limited [Brown et al., 2019b, Huang et al., 2021]. Third, those methods are heavily dependent on the quality of learned latent space, which requires non-trivial efforts to design encoders when dealing with multiple properties. To address the above issues, [Eckmann et al., 2022] use VAE to learn the latent space and train predictors separately using generated molecules, and then leverage latent inceptionism, which involves the decoder solely, to optimize the latent vector with multiple predictors. In this paper, we propose an encoder-free model in both training and optimization to learn the joint distribution of molecules and properties. We then design an efficient algorithm to shift the learned distribution gradually.

Optimization with Reinforcement Learning and Evolutionary Algorithms. Reinforcement learning (RL) based methods directly optimize and generate molecules in an explicit data space [You et al., 2018, Zhou et al., 2019, Jin et al., 2020, Gottipati et al., 2020]. By formulating the property design as a discrete optimization task, they can modify the molecular substructures guided by an oracle reward function. However, the training of those RL-based methods can be viewed as rejection sampling which is difficult and inefficient due to the random-walk search behavior in the discrete space. Evolutionary algorithms (EA) also formulate the optimization in a discrete manner [Nigam et al., 2020, Jensen, 2019, Xie et al., 2021, Fu et al., 2021a,b]. By leveraging carefully-crafted combinatorial search algorithms, they can search the molecule graph space in a flexible and efficient way. However, the design of those algorithms is non-trivial and domain specific.

3 METHODS

3.1 PROBLEM SETUP AND OVERVIEW

We use the SELFIES representation for molecules [Krenn et al., 2020]. It encodes each molecule as a string of characters and ensures validity of all SELFIES strings. Let $x = (x^{(1)}, ..., x^{(t)}, ..., x^{(T)})$ be a molecule string encoded in SELFIES, where $x^{(t)} \in \mathcal{V}$ is the t-th character and $\mathcal{V}$ is the vocabulary.

Suppose $y \in \mathbb{R}$ represents a molecular property of interest. Then the problem we attempt to tackle is to optimize x such that its property $y = y^*$ where y^* is some desirable value for y. We take a probabilistic approach and treat the optimization problem as a sampling problem, that is,

$$x^* \sim p(x|y = y^*). \tag{1}$$

This is a *single-objective optimization* problem since only one property is targeted. In real-world drug design settings, we are more likely to optimize multiple properties simultaneously, that is, *multi-objective optimization*. Suppose we optimize for $\{y_j \in \mathbb{R}\}_{j=1}^m$, then our task is to sample

$$x^* \sim p(x|y_1 = y_1^*, ..., y_m = y_m^*). \tag{2}$$

To address these problems, we propose a solution within a unified probabilistic framework. As a first step, we need to model or approximate the data distribution of molecules and their properties, $p_{\text{data}}(x, y)$. To this end, we recruit latent space energy-based model (LSEBM) [Pang et al., 2020, Nie et al., 2021] to model the molecule and properties. LSEBM assumes that a latent vector $z \in \mathbb{R}^d$ in a low dimensional latent space follows an energy-based prior model $p(z)$. Conditional on z, the molecule x and the property y are independent, so that the joint distribution $p(x, y, z)$ can be factorized as $p(z)p(x|z)p(y|z)$, which leads to $p(x, y) = \int p(z)p(x|z)p(y|z)dz$ as an approximation to $p_{\text{data}}(x, y)$. The latent space energy-based prior model $p(z)$, the molecule generation model $p(x|z)$, and the property regression model $p(y|z)$ can be jointly learned by an approximate maximum likelihood algorithm (see §3.3 and Algorithm 1). LSEBM within the context of molecule data is presented in §3.2.

For the purpose of property optimization, we are required to generate molecules x with some desirable property y^*. Rather than direct optimization in the molecule space, we choose to optimize z in the latent space. We first consider the single-objective optimization problem (Equation (1)). With the learned model, we propose to optimize x given $y = y^*$ by ancestral sampling,

$$z^* \sim p(z|y = y^*), \quad x^* \sim p(x|z = z^*). \tag{3}$$

However, if y^* deviates from the observed data distribution of y, this naive solution involves sampling in an extrapolated regime (or out of distribution regime) where y^* is not in the effective support of the learned distribution. To address this problem, we propose a *Sampling with Gradual Distribution Shifting* (SGDS) approach where we gradually shift the learned distribution to a region where it is supported by high property values (see §3.4 and Algorithm 2).

Our model is designed to be versatile such that it admits straightforward extension to multi-objective optimization. To optimize x given $\{y_j = y_j^*\}_{j=1}^m$, we can simply augment the joint distribution with more regression models, i.e., $p(x, z, y_1, ..., y_m) = p(z)p(x|z)\prod_{j=1}^m p(y_j|z)$. The optimization procedure follows the same SGDS approach. See §3.5 for more details on multi-objective optimization.

3.2 JOINT DISTRIBUTION OF MOLECULE AND MOLECULAR PROPERTY

Suppose $x = (x^{(1)}, ..., x^{(t)}, ..., x^{(T)})$ is a molecule string in SELFIES, $y \in \mathbb{R}$ is the target property of interest, and $z \in \mathbb{R}^d$ is the latent vector. Consider the following model,

$$z \sim p_\alpha(z), \quad [x \mid z] \sim p_\beta(x|z), \quad [y \mid z] \sim p_\gamma(y|z), \quad (4)$$

where $p_\alpha(z)$ is a prior model with parameters α, $p_\beta(x|z)$ is a molecule generation model with parameters β, and $p_\gamma(y|z)$ is a property regression model with parameter γ. In VAE [Kingma and Welling, 2014], the prior is simply assumed to be an isotropic Gaussian distribution. In our model, $p_\alpha(z)$ is formulated as a learnable energy-based model,

$$p_\alpha(z) = \frac{1}{Z(\alpha)} \exp(f_\alpha(z)) p_0(z), \quad (5)$$

where $p_0(z)$ is a reference distribution, assumed to be isotropic Gaussian as in VAE. $f_\alpha : \mathbb{R}^d \to \mathbb{R}$ is a scalar-valued negative energy function and is parameterized by a small multi-layer perceptron (MLP) with parameters α. $Z(\alpha) = \int \exp(f_\alpha(z)) p_0(z) dz = \mathbb{E}_{p_0}[\exp(f_\alpha(z))]$ is the normalizing constant or partition function.

The molecule generation model $p_\beta(x|z)$ is a conditional autoregressive model,

$$p_\beta(x|z) = \prod_{t=1}^{T} p_\beta(x^{(t)}|x^{(1)}, ..., x^{(t-1)}, z) \quad (6)$$

which is parameterized by a one-layer LSTM Hochreiter and Schmidhuber [1997] with parameters β. Note that the latent vector z controls every step of the autoregressive model. It is worth pointing out the simplicity of the molecule generation model of our method considering that those in prior work involve complicated graph search algorithm or alternating generation of atoms and bonds with multiple networks.

Given a molecule x, suppose y is the chemical property of interest, such as drug likeliness or protein binding affinity. The ground-truth property value can be computed for an input x via open-sourced software such as RDKit [Landrum et al., 2013] and AutoDock-GPU [Santos-Martins et al., 2021]. We assume that given z, x and y are conditionally independent, so that

$$p_\theta(x, y, z) = p_\alpha(z) p_\beta(x|z) p_\gamma(y|z), \quad (7)$$

where $\theta = (\alpha, \beta, \gamma)$. We use the model $p_\theta(x, y) = \int p_\theta(x, y, z) dz$ to approximate the data distribution $p_{\text{data}}(x, y)$. See Supplement for a detailed discussion.

The property regression model can be written as

$$p_\gamma(y|z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - s_\gamma(z))^2\right), \quad (8)$$

where $s_\gamma(z)$ is a small MLP, with parameters γ, predicting y based on the latent z. The variance σ^2 is set as a constant or a hyperparameter in our work.

3.3 LEARNING JOINT DISTRIBUTION

Suppose we observe training examples $\{(x_i, y_i), i = 1, ..., n\}$. The log-likelihood function is $L(\theta) = \sum_{i=1}^{n} \log p_\theta(x_i, y_i)$. The learning gradient can be calculated according to

$$\begin{aligned} \nabla_\theta \log p_\theta(x, y) &= \mathbb{E}_{p_\theta(z|x,y)} \left[\nabla_\theta \log p_\theta(x, y, z)\right] \\ &= \mathbb{E}_{p_\theta(z|x,y)} \left[\nabla_\theta (\log p_\alpha(z) + \log p_\beta(x|z) + \log p_\gamma(y|z))\right]. \end{aligned} \quad (9)$$

For the prior model,

$$\nabla_\alpha \log p_\alpha(z) = \nabla_\alpha f_\alpha(z) - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]. \quad (10)$$

The learning gradient given an example (x, y) is

$$\begin{aligned} \delta_\alpha(x, y) &= \nabla_\alpha \log p_\theta(x, y) \\ &= \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\alpha f_\alpha(z)] - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]. \end{aligned} \quad (11)$$

Thus α is updated based on the difference between z inferred from empirical observation (x, y), and z sampled from the current prior.

For the molecule generation model,

$$\delta_\beta(x, y) = \nabla_\beta \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\beta \log p_\beta(x|z)]. \quad (12)$$

Similarly, for the property regression model,

$$\delta_\gamma(x, y) = \nabla_\gamma \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\gamma \log p_\gamma(y|z)]. \quad (13)$$

Estimating expectations in equations 11, 12, and 13 requires MCMC sampling of the prior model $p_\alpha(z)$ and the posterior distribution $p_\theta(z|x, y)$. We recruit Langevin dynamics [Neal, 2011]. For a target distribution $\pi(z)$, the dynamics iterates

$$z_{\tau+1} = z_\tau + s\nabla_z \log \pi(z_\tau) + \sqrt{2s}\epsilon_\tau, \quad (14)$$

where τ indexes the time step of the Langevin dynamics, s is step size, and $\epsilon_\tau \sim \mathcal{N}(0, I_d)$ is the Gaussian white noise. $\pi(z)$ can be either the prior $p_\alpha(z)$ or the posterior $p_\theta(z|x, y)$. In either case, $\nabla_z \log \pi(z)$ can be efficiently computed by back-propagation.

We initialize $z_0 \sim \mathcal{N}(0, I_d)$, and we run Γ steps of Langevin dynamics (e.g. $\Gamma = 20$) to approximately sample from the prior and the posterior distributions. The resulting learning algorithm is an approximate maximum likelihood learning algorithm. See [Pang et al., 2020] for a theoretical understanding of the learning algorithm based on the finite-step MCMC. The learning algorithm is summarized in Algorithm 1.

Algorithm 1: Learning joint distribution.

input : Learning iterations T, learning rates for the prior, generation, and regression models $\{\eta_0, \eta_1, \eta_2\}$, initial parameters $\theta_0 = (\alpha_0, \beta_0, \gamma_0)$, observed examples $\{(x_i, y_i)\}_{i=1}^n$, batch size m, number of prior and posterior sampling steps $\{\Gamma_0, \Gamma_1\}$, and prior and posterior sampling step sizes $\{s_0, s_1\}$.

output : $\theta_T = (\alpha_T, \beta_T, \gamma_T)$.

for $t = 0 : T-1$ **do**

1. **Mini-batch**: Sample observed examples $\{(x_i, y_i)\}_{i=1}^m$.
2. **Prior sampling**: For each i, sample $z_i^- \sim p_{\alpha_t}(z)$ using equation Equation (14), where the target distribution $\pi(z) = p_{\alpha_t}(z)$, and $s = s_0$, $\Gamma = \Gamma_0$.
3. **Posterior sampling**: For each (x_i, y_i), sample $z_i^+ \sim p_{\theta_t}(z|x_i, y_i)$ using Equation (14), where the target distribution $\pi(z) = p_{\theta_t}(z|x_i, y_i)$, and $s = s_1$, $\Gamma = \Gamma_1$.
4. **Update prior model**: $\alpha_{t+1} = \alpha_t + \eta_0 \frac{1}{m} \sum_{i=1}^m [\nabla_\alpha f_{\alpha_t}(z_i^+) - \nabla_\alpha f_{\alpha_t}(z_i^-)]$.
5. **Update generation model**: $\beta_{t+1} = \beta_t + \eta_1 \frac{1}{m} \sum_{i=1}^m \nabla_\beta \log p_{\beta_t}(x_i|z_i^+)$.
6. **Update regression model**: $\gamma_{t+1} = \gamma_t + \eta_2 \frac{1}{m} \sum_{i=1}^m \nabla_\gamma \log p_{\gamma_t}(y_i|z_i^+)$.

3.4 SAMPLING WITH GRADUAL DISTRIBUTION SHIFTING (SGDS)

To tackle the single-objective optimization problem (Equation (1)), one naive approach is to perform ancestral sampling with two steps, given some desirable property value y^*,

$$\text{(i) } z^* \sim p_\theta(z|y = y^*) \propto p_\alpha(z) p_\gamma(y = y^*|z), \tag{15}$$

$$\text{(ii) } x^* \sim p_\beta(x|z = z^*), \tag{16}$$

where (i) is an application of Bayes rule, with $p_\alpha(z)$ as the prior and $p_\gamma(y|z)$ as the likelihood. Sampling from $p_\theta(z|y)$ can be carried out by Langevin dynamics in Equation (14) by replacing the target distribution $\pi(z)$ with $p_\theta(z|y)$.

Our model $p_\theta(x, y, z)$ is learned to capture the data distribution. In real-world settings, y^* might not be within the support of the data distribution. Therefore, sampling following Equation (15) does not work well since it involves extrapolating the learned distribution.

We propose an iterative updating method called *sampling with gradual distribution shifting* (SGDS) to address this issue. In particular, we first leverage the n samples collected from the common dataset $\{(x_i^0, y_i^0)\}_{i=1}^n$ (e.g. ZINC, $n = 250,000$) to learn the initial joint distribution $p_{\theta_0}(x, y)$ as a valid starting point. Then we shift the joint distribution progressively using a smaller number k (e.g., $k = 10,000$) of synthesized samples $\{(x_i^t, y_i^t)\}_{i=1}^k$ from distribution $p_{\theta_{t-1}}$ at the previous iteration, where $k \ll n$. Therefore, by progressively learning the joint distribution with T (e.g., $T = 30$) shift iterations, $p_{\theta_1}, \ldots, p_{\theta_T}$, at the last several iterations, we expect to generate molecules with desirable properties that are significantly distant from the initial distribution as shown in Figure 1c.

Next, we shall explain in detail the steps to generate $\{(x_i^t, y_i^t)\}_{i=1}^k$ given $p_{\theta_{t-1}}$. In a property maximization task, we shift the support slightly by adding a small Δ_y to all y's,

$$\tilde{y}^t = y^{t-1} + \Delta_y, \tag{17}$$

and generate x^t conditional on shifted $\tilde{y}^t$, following Equation (15),

$$\text{(i) } z^t \sim p_{\theta_{t-1}}(z|y = \tilde{y}^t), \tag{18}$$

$$\text{(ii) } x^t \sim p_{\beta_{t-1}}(x|z = z^t). \tag{19}$$

Δ_y can be chosen as a fixed small value. After generating x^t, its ground-truth property value y^t can be computed by calling the corresponding engines such as RDKit and AutoDock-GPU. In Equation (18), sampling can be achieved by langevin dynamics as in Equation (14). For the sake of efficiency, we propose to run persistent chain by initializing the Langevin dynamics from the latent vectors generated in the previous iteration. This is also called warm start. Specifically, we have

$$\begin{aligned} z_0^t &= z_\Gamma^{t-1}, \\ z_{\tau+1}^t &= z_\tau^t + s\nabla_z \log p_{\theta_{t-1}}(z|\tilde{y}^t) + \sqrt{2s}\epsilon_\tau, \end{aligned} \tag{20}$$

for $\tau = 1, ..., \Gamma$, where Γ is the length of Markov chain in each iteration. With warm start, we use $\Gamma = 2$ in our experiments.

For more efficient optimization via distribution shifting, we further introduce a rank-and-select scheme by maintaining a buffer of top-k samples of (z, x, y) (where z is the sampled latent vector, x is the generated molecule, and y is the ground-truth property value of x). Specifically, we maintain a buffer which consists of k samples of (z, x, y) with the highest values of y in the past shifting iterations. In each shift iteration, conditioned on the shifted property values, with warm start, initialized from these k vectors z in the buffer, a new batch of k latent vectors z, molecules x, and their ground-truth values y can be produced in the new shift iteration. We rank all the $2k$ samples of (z, x, y) (including k newly generated ones and k samples in the buffer) and select the top-k samples of (z, x, y) based on the ground-truth values y. The k samples of (z, x, y) in the buffer are then updated by those newly selected k samples of (z, x, y). We call this procedure as *rank-and-select*. This rank-and-select procedure can also be applied to constrained optimization tasks, where we select those sampled molecules that satisfy the given constraints. With the selected samples, we then shift the model distribution by learning from these samples with several learning iterations. The SGDS algorithm is summarized in Algorithm 2.

Algorithm 2: SGDS for single property optimization.

input : Shift iterations T, initial pretrained parameters $\theta_0 = (\alpha_0, \beta_0, \gamma_0)$, initial examples $\{(x_i^0, y_i^0)\}_{i=1}^k$ from the data distribution boundary, shift magnitude Δ_y, $\text{PropertyComputeEngine} = \text{RDKit or AutoDock-GPU}$, $\text{LearningAlgorithm} = \text{Algorithm 1}$.

output : $\{(x_i^T, y_i^T)\}_{i=1}^k$.

for $t = 1 : T$ **do**

1. **Property shift**: For each y_i^{t-1}, $\tilde{y}_i^t = y_i^{t-1} + \Delta_y$.
2. **Latent sampling with warm start**: For each $\tilde{y}_i^t$, sample $z_i^t \sim p_{\theta_{t-1}}(z|\tilde{y}_i^t)$ using Equation (20).
3. **Molecule generation**: For each z_i^t, sample $x_i^t \sim p_{\theta_{t-1}}(x|z_i^t)$.
4. **Property computation**: For each x_i^t, compute $y_i^t = \text{PropertyComputeEngine}(x_i^t)$.
5. **Rank-and-select:** Update the buffer of top-k samples $\{z_i^t, x_i^t, y_i^t\}_{i=1}^k$ by rank-and-select.
6. **Distribution shift**: $\theta_t = \text{LearningAlgorithm}(\{(x_i^t, y_i^t)\}_{i=1}^k, \theta_{t-1})$.

3.5 MULTI-OBJECTIVE OPTIMIZATION

We next consider the multi-objective optimization problem. Suppose we optimize for a set of properties $\{y_j\}_{j=1}^m$, then we learn a property regression model for each property y_j,

$$p_{\gamma_j}(y_j|z) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2\sigma_j^2}(y_j - s_{\gamma_j}(z))^2\right), \tag{21}$$

where each s_{γ_j} is a small MLP with parameters γ_j. We assume that given z the properties are conditionally independent, so the joint distribution is

$$p_\theta(x, z, y_1, ..., y_m) = p_\alpha(z) p_\beta(x|z) \prod_{j=1}^m p_{\gamma_j}(y_i|z). \tag{22}$$

Under our framework, both the learning and the sampling algorithm for the single-objective problem can be straightforwardly extended to the multi-objective setting. In SGDS, we shift the values of the multiple properties simultaneously, and generate molecules conditional on the multiple properties.

4 EXPERIMENTS

To demonstrate the effectiveness of our proposed method, SGDS, we compare our model with previous SOTA methods for molecule design including single-objective optimization (§4.2), multi-objective optimization (§4.3) and constrained optimization (§4.4). In molecule design experiments, we consider both non-biological and biological properties. Finally, we add ablation studies to analyze the effects of different components in SGDS. We also conduct unconditional molecule generation experiments, multiple property optimization (MPO) experiments in Guacamol benchmark [Brown et al., 2019a], and these results are included in the Supplement.

4.1 EXPERIMENTAL SETUP

Datasets. For the molecule property optimization task, we report results on ZINC [Irwin et al., 2012], which consists of around 2.5×10^4 molecules. The SMILES representations of molecules are changed into SELFIES using [Krenn et al., 2020]. The non-biological properties (such as penalized logP, QED, etc.) can be computed using RDKit [Landrum et al., 2013]. Following [Eckmann et al., 2022], we use the docking scores from AutoDock-GPU [Santos-Martins et al., 2021] to approximate the binding affinity to two protein targets, human estrogen receptor (ESR1) and Human peroxisomal acetyl-CoA acyl transferase 1 (ACAA1).

Training Details. There are three modules in our method, the molecule generation model $p_\beta(x|z)$, the energy-based prior model $p_\alpha(z)$, and property regression model $\{p_{\gamma_j}(y|z)\}_{j=1}^m$, where m is the total number of properties we aim to optimize. The generation model $p_\beta(x|z)$ is parameterized by a single-layer LSTM with 1024 hidden units where the dimension of latent vector z is 100. The energy-based prior model $p_\alpha(z)$ is a 3-layer MLP. Each of the property regression model $p_{\gamma_j}(y|z)$ is a 3-layer MLP. It is worth mentioning that compared to most previous models, SGDS is characterized by its simplicity without adding inference networks for sampling, or RL-related modules for optimization. In order to get valid initial distribution θ_0 for SGDS, we first train our model for 30 epochs on ZINC. We use Adam optimizer [Kingma and Ba, 2015] to train our models with learning rates 10^{-4} for energy-based prior model, and 10^{-3} for the molecule generation model and property regression model. During SGDS, we use 30 shifting iterations for single-objective optimization and 20 for multi-objective optimization. For each iteration of distribution shifting, we sample 10^4 boundary examples except binding affinity, where 2×10^3 examples are used to speed up calculation, and then we update the model parameters θ for 10 iterations using Algorithm 1 with Adam optimizer and the same learning rates mentioned above. All experiments are conducted on Nvidia Titan XP GPU.

4.2 SINGLE-OBJECTIVE OPTIMIZATION

Penalized logP and QED maximization. For non-biological properties, we are interested in Penalized logP and QED, both of which can be calculated by RDKit [Landrum et al., 2013].

Since we know Penalized logP scores have a positive relationship with the length of molecules, we maximize Pe-

nalized logP either with or without maximum length limit. Following [Eckmann et al., 2022], the maximum length is set to be the maximum length of molecules in ZINC using SELFIES. From Table 1, we can see that with length limit, SGDS outperforms previous methods by a large margin. We also achieve the highest QED with/without length limit. These observations demonstrate the effectiveness of our method. We also illustrate our distribution shifting method in Figure 1c. One can notice that, the distribution of the property is gradually shifted towards the region with higher values, and the final distribution is significantly distant from the initial one.

Table 1: Non-biological single-objective optimization. Report top-3 highest scores found by each model. LL (Length Limit) denotes whether it has the maximum length limit. Baseline results obtained from [Eckmann et al., 2022, You et al., 2018, Luo et al., 2021, Xie et al., 2021].

Method	LL	Penalized logP (↑)			QED (↑)		
		1st	2rd	3rd	1st	2rd	3rd
JT-VAE	✗	5.30	4.93	4.49	0.925	0.911	0.910
GCPN	✓	7.98	7.85	7.80	**0.948**	0.947	0.946
MolDQN	✓	11.8	11.8	11.8	**0.948**	0.943	0.943
MARS	✗	45.0	44.3	43.8	**0.948**	**0.948**	**0.948**
GraphDF	✗	13.7	13.2	13.2	**0.948**	**0.948**	**0.948**
LIMO	✓	10.5	9.69	9.60	0.947	0.946	0.945
SGDS	✓	**26.4**	**25.8**	**25.5**	**0.948**	**0.948**	**0.948**
SGDS	✗	**158.0**	**157.8**	**157.5**	**0.948**	**0.948**	**0.948**

Biological Property Optimization. ESR1 and ACAA1 are two human proteins. We aim to design ligands (molecules) that have the maximum binding affinities towards those target proteins. ESR1 is well-studied, which has many existing binders. However, we do not use any binder-related information in SGDS. Binding affinity is measured by the estimated dissociation constants $\mathrm{K_D}$, which can be approximated by docking scores from AutoDock-GPU [Santos-Martins et al., 2021]. Large binding affinities corresponds to small $\mathrm{K_D}$. That is, we aim to minimize $\mathrm{K_D}$. Table 2 shows that our model outperforms previous methods on both ESR1 and ACAA1 binding affinity maximization tasks by large margins. Comparing to existing methods, much more molecules with high binding affinities can be directly sampled from the last several shifting iterations. See Supplement for more examples. Producing those ligands with high binding affinity plays a vital role in the early stage of drug discovery.

4.3 MULTI-OBJECTIVE OPTIMIZATION

Multi-objective Binding Affinity Optimization. We consider maximizing binding affinity, QED and minimizing synthetic accessibility score (SA) simultaneously. Following Eckmann et al. [2022], we exclude molecules with abnormal behaviors [1] to encourage the joint distribution shifts towards a desirable region in terms of pharmacologic and synthetic properties. Those heuristics can be conveniently added in our *rank-and-select* step. Table 3 shows our multi-objective results compared to LIMO and GCPN. From the results, we can see that SGDS is able to find the ligands with desired properties while keeping the pharmacologic structures. For ESR1, we have two existing binders on the market, Tamoxifen and Raloxifene. Our designed ligands have similar QED and SA, with very low $\mathrm{K_D}$. Compared to existing methods, SGDS obtains better results in overall adjustments. For ACAA1, we do not have any existing binders. Compared with prior SOTA methods, our optimized ligands outperform those by a large margin in terms of $\mathrm{K_D}$. When comparing with single-objective optimization, we find that multi-objective optimization is more complicated, but it may be more useful in real world molecule design. While we still need domain expertise to determine the effectiveness of those ligands discovered by SGDS, we believe the ability of SGDS to generate many high quality molecules given multiple metrics is extremely useful in the early stage of drug discovery.

[1] In particular, we exclude molecules with QED↑ smaller than 0.4, SA↓ larger than 5.5, and too small (less than 5 atoms) or too large (more than 6 atoms) chemical rings.

Table 2: Biological single-objective optimization. Report top-3 lowest $\mathrm{K_D}$ (in nanomoles/liter) found by each model. Baseline results obtained from [Eckmann et al., 2022].

Method	ESR1 $\mathrm{K_D}$ (↓)			ACAA1 $\mathrm{K_D}$ (↓)		
	1st	2rd	3rd	1st	2rd	3rd
GCPN	6.4	6.6	8.5	75	83	84
MolDQN	373	588	1062	240	337	608
MARS	25	47	51	370	520	590
GraphDF	17	64	69	163	203	236
LIMO	0.72	0.89	1.4	37	37	41
SGDS	**0.03**	**0.03**	**0.04**	**0.11**	**0.11**	**0.12**

Table 3: Muli-objective Optimization for both ESR1 and ACAA1. Report Top-2 average scores related to $\mathrm{K_D}$ (in nmol/L), QED and SA. Baseline results obtained from [Eckmann et al., 2022].

Ligand	ESR1			ACAA1		
	$\mathrm{K_D}$ ↓	QED ↑	SA ↓	$\mathrm{K_D}$ ↓	QED ↑	SA ↓
Tamoxifen	87	0.45	2.0	–	–	–
Raloxifene	7.9×10^6	0.32	2.4	–	–	–
GCPN 1st	810	0.43	4.2	8500	0.69	4.2
GCPN 2nd	27000	0.80	3.7	8500	0.54	4.3
LIMO 1st	4.6	0.43	4.8	28	0.57	5.5
LIMO 2nd	2.8	0.64	4.9	31	0.44	4.9
SGDS 1st	**0.36**	0.44	3.99	**4.55**	0.56	4.07
SGDS 2nd	1.28	0.44	3.86	5.67	0.60	4.58

4.4 CONSTRAINED OPTIMIZATION

To optimize single-objective under some constrains, we use the original SGDS steps and in *rank-and-select*, we only keep the molecules that satisfy the constraints.

Similarity-constrained Penalized logP Maximization Following JT-VAE [Jin et al., 2018], this experiment aims to generate molecules with high penalized logP while being similar to the target molecules. Similarity is measured by Tanimoto similarity between Morgan fingerprints with a cutoff value δ. We compare our results with previous SOTA method in Table 4. The results show that SGDS tends to obtain better results with weak constraints (i.e. $\delta = 0, 0.2$) with 100% success rate, since different from optimized property, the constraints are added implicitly.

Table 4: Similarity-constrained optimization results. LIMO results obtained from [Eckmann et al., 2022, Luo et al., 2021].

δ	GraphDF		LIMO		SGDS	
	Improv.	% Succ.	Improv.	% Succ.	Improv.	% Succ.
0.0	5.9 ± 2.0	100	10.1 ± 2.3	100	$\mathbf{19.1} \pm 2.1$	100
0.2	5.6 ± 1.7	100	5.8 ± 2.6	99.0	$\mathbf{7.4} \pm 1.9$	100
0.4	$\mathbf{4.1} \pm 1.4$	100	3.6 ± 2.3	93.7	3.8 ± 1.4	97.5
0.6	1.7 ± 1.2	93.0	1.8 ± 2.0	85.5	$\mathbf{2.6} \pm 2.0$	95.6

logP targeting In Table 5, comparing to previous methods, SGDS is able to get competitive diversity scores with significantly better success rate in both ranges. That is because after SGDS, our model is shifted towards the region that is supported by molecules satisfying the logP constraints. Due to the flexibility of our EBM prior, SGDS achieves high diversity scores while keeping most of the sampled molecules within the logP range.

Table 5: logP targeting to a certain range [Eckmann et al., 2022, You et al., 2018, Luo et al., 2021, Xie et al., 2021].

Method	$-2.5 \leq \text{logP} \leq -2$		$5 \leq \text{logP} \leq 5.5$	
	Success	Diversity	Success	Diversity
ZINC	0.4%	0.919	1.3%	0.901
JT-VAE	11.3%	0.846	7.6%	0.907
ORGAN	0	–	0.2%	**0.909**
GCPN	85.5%	0.392	54.7%	0.855
LIMO	10.4%	**0.914**	–	–
SGDS	**86.0%**	0.874	**62.2%**	0.858

4.5 ABLATION STUDIES

SGDS outperforms previous methods by a significant margin, especially on binding affinity related experiments. Hence, we conduct ablations on the key components of our method on a challenging single-objective ACAA1 maximization experiment. Since SGDS is optimized based on shifting the joint distribution rather than per-molecule based optimization method (such as [Eckmann et al., 2022]), we use the summarized statistics (i.e. the mean and standard deviation of 100 lowest $\mathrm{K_d}$ from uniquely generated molecules) of last three shifted distributions as our metric to compare the key components rather than top-3 optimized $\mathrm{K_D}$ in §4.2. Ablation studies are discussed as follows.

(1) *Without EBM Prior*: for joint distribution, we replace the learnable EBM prior by a fixed $\mathcal{N}(0, I_d)$. (2)*Without Property Regression* $p_\gamma(y|z)$: we only learn the distribution of molecules as $p_\alpha(z)p_\beta(x|z)$. For each iteration of distribution shifting, we only use rank-and-select and update model parameters based on those molecules with high values. The molecule can be generated by first sampling $z \sim p_\alpha(z)$ and then $x \sim p_\beta(x|z)$. (3) *Without Gradual Shifting*: rather than iterative distribution shifting in SGDS, we directly sample $z \sim p_\theta(z|y = y^\star)$, where $y^\star$ is set to be the minimal value we can get in §4.2. (4) *Without Rank-and-Select:* we skip the rank-and-select step in Algorithm 2. (5) *Without Warm Start:* when sampling $z \sim p_\theta(z|y)$ in current iteration, we replace the warm start algorithm in Equation (20) by 20-step langevin dynamics in Equation (14) with the same step size.

Table 6: Ablation Studies. Report the mean and stand deviation of 100 uniquely generated molecules with the lowest $\mathrm{K_D}$(in 10^{-9}mol/L) from last three shifted iterations (i.e. the 28th, 29th, 30th iterations of total 30 iterations).

Method	28th	29th	30th
SGDS	0.74 ± 0.04	0.61 ± 0.03	0.59 ± 0.03
Without EBM Prior	47.8 ± 26.9	38.8 ± 21.1	35.1 ± 20.2
Without Property Regression	140 ± 74.8	114 ± 67.0	103 ± 56.3
Without Gradual Shifting	211 ± 125	166 ± 97.9	137 ± 74.8
Without Rank-and-Select	9.71 ± 5.52	5.75 ± 3.39	3.91 ± 2.17
Without Warm Start	6.27 ± 3.92	3.27 ± 2.99	2.37 ± 1.35

The ablation studies are displayed in Table 6. It is clear that all the proposed components contribute significantly to the good performance of our method.

5 CONCLUSION AND DISCUSSION

We propose a deep generative model for the joint distribution of molecules and their properties. It assumes an energy-based prior model in a low-dimensional continuous latent space, and the latent vector can generate the molecule and predict its property value. We then design a sampling with gradual distribution shifting (SGDS) method to shift the learned distribution to a region with high property values. Molecule design can then be achieved by conditional sampling. Our experiments demonstrate that our method outperforms previous SOTA methods on some tasks by significant margins.

A limiting factor is that we assume the properties of interest can be readily and accurately computed by existing software, which is an assumption made by most of the existing methods. In reality, much work still needs to be done to define and compute biologically and chemically meaningful objectives or metrics. In our work, we focus on optimizing given objectives or metrics, and our method can be readily applied to improved versions of objectives or metrics.

References

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Liggpt: Molecular generation using a transformer-decoder model. 2021.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019a.

Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019b. doi: 10.1021/acs.jcim.8b00839. URL https://doi.org/10.1021/acs.jcim.8b00839. PMID: 30887799.

Binghong Chen, Tianzhe Wang, Chengtao Li, Hanjun Dai, and Le Song. Molecule optimization by explainable evolution. In *International Conference on Learning Representation (ICLR)*, 2021.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *CoRR*, abs/1903.08689, 2019. URL http://arxiv.org/abs/1903.08689.

Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael K Gilson, and Rose Yu. Limo: Latent inceptionism for targeted molecule generation. *arXiv preprint arXiv:2206.09010*, 2022.

Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, 2016. URL http://arxiv.org/abs/1611.03852.

Tianfan Fu, Cao Xiao, and Jimeng Sun. Core: Automatic molecule optimization using copy & refine strategy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 638–645, 2020.

Tianfan Fu, Wenhao Gao, Cao Xiao, Jacob Yasonik, Connor W Coley, and Jimeng Sun. Differentiable scaffolding tree for molecular optimization. *arXiv preprint arXiv:2109.10469*, 2021a.

Tianfan Fu, Cao Xiao, Xinhao Li, Lucas M Glass, and Jimeng Sun. Mimosa: Multi-constraint molecule sampling for molecule optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 125–133, 2021b.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.

Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Simon Blackburn, Karam Thomas, Connor Coley, Jian Tang, et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International Conference on Machine Learning*, pages 3668–3679. PMLR, 2020.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521*, 2019.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International conference on machine learning*, pages 4849–4859. PMLR, 2020.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *CoRR*, abs/1606.03439, 2016. URL `http://arxiv.org/abs/1606.03439`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

Panagiotis-Christos Kotsias, Josep Arús-Pous, Hongming Chen, Ola Engkvist, Christian Tyrchan, and Esben Jannik Bjerrum. Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. *Nature Machine Intelligence*, 2(5):254–265, 2020.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Rithesh Kumar, Anirudh Goyal, Aaron C. Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *CoRR*, abs/1901.08508, 2019. URL `http://arxiv.org/abs/1901.08508`.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954, 2017.

Greg Landrum et al. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 2013.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pages 7192–7203. PMLR, 2021.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Radford M Neal. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

Jiquan Ngiam, Zhenghao Chen, Pang Wei Koh, and Andrew Y. Ng. Learning deep energy models. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1105–1112, 2011. URL `https://icml.cc/2011/papers/557_icmlpaper.pdf`.

Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34:13497–13510, 2021.

AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *ICLR 2020*, 2020.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run MCMC toward energy-based model. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, Canada*, 2019.

Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. *arXiv preprint arXiv:2006.08205*, 2020.

Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

Diogo Santos-Martins, Leonardo Solis-Vasquez, Andreas F Tillack, Michel F Sanner, Andreas Koch, and Stefano Forli. Accelerating autodock4 with gpus and gradient-based local search. *Journal of chemical theory and computation*, 17(2):1060–1073, 2021.

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Robin Winter, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert. Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science*, 10(34):8016–8024, 2019.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2635–2644, 2016. URL `http://proceedings.mlr.press/v48/xiec16.html`.

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=kHSu4ebxFXY`.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pages 6410–6421, 2018.

Chengxi Zang and Fei Wang. 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, 2020.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.