MOLEBM: MOLECULE GENERATION AND DESIGN BY LATENT SPACE ENERGY-BASED MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generation of molecules with desired chemical and biological properties such as high drug-likeness, high binding affinity to target proteins, is critical in 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. Given the latent vector sampled from the latent space EBM, both molecules and molecular properties are conditionally sampled via a molecule generator model and a property regression model respectively. The EBM in a low dimensional latent space allows our model to capture complex chemical rules implicitly but efficiently and effectively. Due to the joint modeling with chemical properties, molecule design can be conveniently and naturally achieved by conditional sampling from our learned model given desired properties, in both single-objective and multi-objective optimization settings. The latent space EBM, molecule generator, and property regression model are learned jointly by approximate maximum likelihood, while optimization of properties is accomplished by gradual shifting of the model distribution towards the region supported by molecules with high property values. Our experiments show that our model outperforms state-of-the-art models 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 is 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 like 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 & Kipf, 2018; Zhou et al., 2019; Shi et al., 2020; Luo et al., 2021). A number of other efforts have been made to optimize molecular properties with genetic algorithms (Nigam et al., 2020), particle-swarm algorithms (Winter et al., 2019) 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 generative model of molecule distributions and optimizing chemical 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 & Bengio, 2016; Xie et al., 2016; Kumar et al., 2019; Nijkamp et al., 2019; Du & 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. For molecule modeling, without any explicit validity constraints in generation, our model generates molecules with high validity with simple SMILES representation (Weininger, 1988).

Given our goal of property optimization, we learn a joint distribution of molecules and their properties. Our model consists of 1) an EBM in a low-dimensional continuous latent space, 2) a generator mapping from the latent space to the observed molecule space, and 3) a property regression model mapping from the latent space to the property values (see Figure 1). We call our model as *MolEBM*. All three components in our model are learned jointly by approximate maximum likelihood. A learned model generates a molecule with a high property value in two steps: 1) given the property value, sample the latent vector; 2) given the sampled latent vector, generate a molecule (see the top-to-bottom path in Figure 1a). Since the learned model approximates the data distribution, directly sampling from the learned model conditional on a high property value does not work well since a molecule with a high property value is most likely not in the original data distribution. We thus design a method to gradually shift the learned distribution towards the region supported by molecules with high property values, and sample molecules with desirable properties from the shifted distribution.

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. Learning the model involves inferring the latent vector given both the molecule and the property value, and we recruits Langevin dynamics instead of amortized inference network for inference computation. This design makes our approach versatile in dealing with varying number of properties.

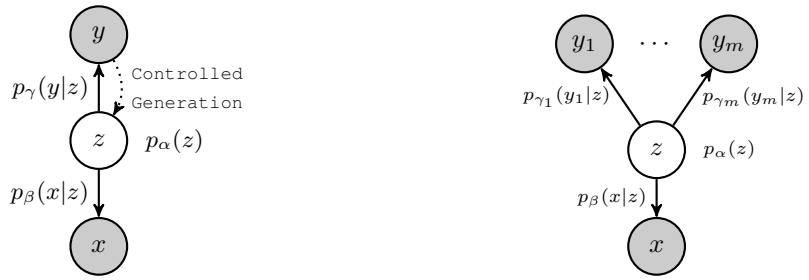
We evaluate our method in various settings including single-objective optimization and multi-objective optimization. Our method outperforms prior methods by significant margins.

In summary, our contributions are as follows:

- We propose to learn a latent space energy-based model for the joint distribution of molecules and molecular properties.
- We develop a *sampling with gradual distribution shifting* method, enabling 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 together.
- Our model achieves state-of-the-art performances on a wide range of molecule optimization tasks.

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 & 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 & Kipf, 2018; Honda et al., 2019; Madhawa et al., 2019; Shi et al., 2020; Zang & 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., 2019; 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 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, and make it possible to obtain several adequate predictors. We then design an efficient algorithm to shift the learned distribution iteratively.



(a) Single-Objective Optimization. (b) Multi-Objective Optimization.

Figure 1: An illustration of MolEBM. x represents a molecule, z is the latent vector, y is a molecular property of interest, $\{y_j\}_{j=1}^m$ indicates m properties.

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 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 need 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 under a unified probabilistic framework. As a first step, we need to model the data distribution of molecules, $p_{\text{data}}(x)$. To this end, we recruit latent space energy-based model (Pang et al., 2020; Nie et al., 2021) for its expressiveness. LSEBM assumes latent vector $z \in \mathbb{R}^d$ in a low dimensional latent space follows an energy-based model, while the observed x is generated by a generator conditional z, that is, $p(x, z) = p(x|z)p(z)$. We introduce LSEBM within the context of molecule data in §3.2.

For the purpose of property optimization, we propose to model the joint distribution of molecules and molecular properties (§3.3). We first consider the single-objective optimization problem (eq. 1). Given the expressiveness of LSEBM, we assume that the latent vector z captures data regularities in x (evaluated in §4.2 and §A.4). Thus, we can learn a simple regression model from the low-dimensional latent space, that is, $p(y|z)$, and the joint distribution is $p(x, y, z) = p(z)p(x|z)p(y|z)$. The model is learned with maximum likelihood (see §3.4 and Algorithm 1).

With the learned model, we can optimize x given $y = y^*$ by ancestral sampling $z^* \sim p(z|y = y^*)$ and $x^* \sim p(x|z = z^*)$. 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. We propose a *Sampling with Gradual Distribution Shifting* (SGDS) approach where we 1) sample on the boundary of the effective support of the learned distribution, and 2) gradually shift the learned distribution with these boundary samples to a region where it is supported by high property values (see §3.5 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 sampling procedure follows the same SGDS approach. See §3.6 for more details on multi-objective optimization.

3.2 Latent Space Energy-Based Model

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

$$z \sim p_\alpha(z), \quad x \sim p_\beta(x|z), \tag{3}$$

where $p_\alpha(z)$ is a prior model with parameters α, and $p_\beta(x|z)$ is a generation model with parameters β. In VAE (Kingma & Welling, 2014), the prior is simply assumed to be an isotropic Gaussian distribution. In our model, $p_\alpha(z)$ is formulated as an energy-based model,

$$p_\alpha(z) = \frac{1}{Z(\alpha)} \exp(f_\alpha(z))p_0(z), \tag{4}$$

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 the 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 partition function.

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

$$p_\beta(x|z) = \prod_{t=1}^{T} p_\beta(x^{(t)}|x^{(1)}, ..., x^{(t-1)}, z) \tag{5}$$

which is parameterized by a one-layer LSTM with parameters β. Note that the latent vector z controls every step of the autoregressive model.

3.3 Joint Distribution of Molecule and Molecular Property

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

$$p_\theta(x, y, z) = p_\alpha(z)p_\beta(x|z)p_\gamma(y|z), \tag{6}$$

where $p_\alpha(z)$ is the EBM prior, $p_\beta(x|z)$ is the generation model, and $p_\gamma(y|z)$ is the property regression model, and $\theta = (\alpha, \beta, \gamma)$. We use the model $p_\theta(x, y, z)$ to approximate the data distribution of (x, y). See Appendix §A.1 for details.

The property regression model can be written as

$$p_\gamma(y|z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - s_\gamma(z))^2\right), \tag{7}$$

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 hyperparameter in our work.

3.4 Learning Algorithm

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

$$\nabla_\theta \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}\left[\nabla_\theta \log p_\theta(x, y, z)\right] \tag{8}$$

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

For the prior model, $\nabla_\alpha \log p_\alpha(z) = \nabla_\alpha f_\alpha(z) - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]$. Thus the learning gradient given an example (x, y) is

$$\delta_\alpha(x, y) = \nabla_\alpha \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\alpha f_\alpha(z)] - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]. \tag{10}$$

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

$$\delta_\beta(x, y) = \nabla_\beta \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\beta \log p_\beta(x|z)]. \tag{11}$$

Similarly, for the regression model,

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

Estimating expectations in equations 17, 18, and 19 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

$$z_{k+1} = z_k + s\nabla_z \log \pi(z_k) + \sqrt{2s}\epsilon_k, \tag{13}$$

where k indexes the time step of the Langevin dynamics, s is a small step size, and $\epsilon_k \sim \mathcal{N}(0, I_d)$ is the Gaussian white noise. $\pi(z)$ can be either $p_\alpha(z)$ or $p_\theta(z|x, y)$. In either case, $\nabla_z \log \pi(z)$ can be efficiently computed by back-propagation. See Appendix §A.1 for more details.

Algorithm 1: Learning MolEBM.

input : Learning iterations T, learning rates for the prior, generation, and regression model $\{\eta_0, \eta_1, \eta_2\}$, 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 $\{K_0, K_1\}$, 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 (13), where the target distribution $\pi(z) = p_{\alpha_t}(z)$, and $s = s_0$, $K = K_0$.
3. **Posterior sampling**: For each (x_i, y_i), sample $z_i^+ \sim p_{\theta_t}(z|x_i, y_i)$ using equation (13), where the target distribution $\pi(z) = p_{\theta_t}(z|x_i, y_i)$, and $s = s_1$, $K = K_1$.
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**: $\gamma_{t+1} = \gamma_t + \eta_2 \frac{1}{m} \sum_{i=1}^m \nabla_\gamma \log p_{\gamma_t}(y_i|z_i^+)$.

3.5 Sampling with Gradual Distribution Shifting

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

$$(1)\ z^* \sim p_\theta(z|y = y^*) \propto p_\alpha(z)p_\gamma(y = y^*|z) \quad (2)\ x^* \sim p_\beta(x|z = z^*), \tag{14}$$

where (1) is an application of Bayes' rule, with $p_\alpha(z)$ as the prior and $p_\gamma(y|z)$ as the likelihood.

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 14 does not work well since it involves extrapolating the learned distribution. We propose a method called *sampling with gradual distribution shifting* (SGDS) to address this issue. In particular, we find examples from the training data on the boundary of the distribution support by sorting them according to the values of y and taking top-k samples, $\{(x_i^{(\text{old})}, y_i^{(\text{old})})\}_{i=1}^k$, with high property values. We shift the support slightly by adding some small Δ_y to all y's, and sample x conditional on shifted y's, following equation 14. Given the generated x's, we compute their groundtruth y's with RDKit or AutoDock-GPU. We then shift the learned model by finetuning it with the new data, $\{(x_i^{(\text{new})}, y_i^{(\text{new})})\}_{i=1}^k$, for a few steps (e.g., 10). This completes one iteration of distribution shift. The molecule with desired high property value is sampled after T shifting iterations. We summarize the algorithm in Algorithm 2.

Algorithm 2: Sampling with Gradual Distribution Shifting (SGDS).

input : Shift iterations T, initial 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 $t = 0 : T-1$ **do**

1. **Property shift**: For each y_i^t, $\tilde{y}_i^{t+1} = y_i^t + \Delta_y$.
2. **Latent sampling**: For each $\tilde{y}_i^{t+1}$, sample $z_i^{t+1} \sim p_{\theta_t}(z|\tilde{y}_i^{t+1})$.
3. **Molecule sampling**: For each z_i^{t+1}, sample $x_i^{t+1} \sim p_{\theta_t}(x|z_i^{t+1})$.
4. **Property computation**: For each x_i^{t+1}, compute $y_i^{t+1} = \text{PropertyComputeEngine}(x_i^{t+1})$.
5. **Distribution shift**: $\theta_{t+1} = \text{LearningAlgorithm}(\{(x_i^{t+1}, y_i^{t+1})\}_{i=1}^k, \theta_t)$.

For constrained optimization, in step 3, we only keep the sampled molecules that satisfy the given constraints.

3.6 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,

$$p_{\gamma_j}(y_j|z) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2\sigma_j^2}(y_j - s_{\gamma_j}(z))^2\right), \tag{15}$$

where each s_{γ_j} is a small MLP with parameters γ_j. Then the joint distribution is,

$$p_\theta(x, z, y_1, ..., y_m) = p_\alpha(z) p_\beta(x|z) \prod_{j=1}^m p_{\gamma_j}(y_i|z). \tag{16}$$

Under our framework, the learning algorithm and the sampling algorithm for the single-objective problem can be straightforwardly extended to the multi-objective setting. In both settings, the same types of properties are provided in both the initial training stage and the distribution shifting stage.

4 Experiments

To demonstrate the effectiveness of our proposed model, we compare our model with previous SOTA methods for unconditional generation (§4.2) and molecule design including single-objective optimization (§4.3) and multi-objective optimization (§4.4). In molecule design experiments, we consider both non-biological and biological properties.

4.1 Experimental Setup

Datasets. For the unconditional generation task, we report results on ZINC (Irwin et al., 2012) and MOSES datasets (Polykovskiy et al., 2020). ZINC consists of around $250k$ molecules, and MOSES comprises around 2 million molecules. For optimization tasks, we conduct experiments with ZINC.

Training Details. There are three modules in our model, the top-down generation model $p_\beta(x|z)$, the EBM prior $p_\alpha(z)$, and the regression model $p_\gamma(y|z)$. $p_\beta(x|z)$ is parameterized by a single-layer LSTM with 1024 hidden units. The dimension of latent vector z is 100. $p_\alpha(z)$ is a 3-layer MLP with 100 input dimension and 200 hidden units. The property regression model $p_\gamma(y|z)$ is a 3-layer MLP with 100 input dimension and 100 hidden units. In multi-objective optimization settings, we recruit one MLP for each property and these MLPs have the same architecture as mentioned above. It is worth mentioning that compared to most previous models, our model is characterized by its simplicity without adding inference networks, RL-related modules, and graph neural networks. Adam (Kingma & Ba, 2015) optimizer is used to train our models with learning rates 0.0001 for EBM and 0.001 for the rest. We train our models for 30 epochs. $10,000$ (non-biological) and $2,000$

(biological) boundary examples are selected for distribution shifting. The numbers of shift iterations are 50 (non-biological single-objective), 20 (non-biological single-objective) and 10 (multi-objective). Within each SGDS iteration, the model is finetuned with only 10 parameter updates. All experiments are conducted on Nvidia Titan XP GPU. See more details in Appendix §A.6.

4.2 Unconditional Generation

Three types of encoding systems are used to encode molecules in prior work: SMILES (Weininger, 1988), SELFIES (Krenn et al., 2020), and graph. SMILES and SELFIES linearize a molecule graph into a string of characters. Most previous models using SMILES struggle to generate molecules with high validity, which is the percentage of molecules that satisfy the chemical valency rule. Thus graph representations become popular since explicit valency constraints can be imposed. However, perfect validity in this approach does not imply that the model captures the chemical rules since it is achieved with external constraints. Recently, SELFIES is developed where every SELFIES string corresponds to a valid molecule due to the nature of the encoding system. Thus, validity for generated SMILES strings is a good indicator on how well a learned model captures the basic chemical rules implicitly. Besides validity, we also compare models on uniqueness (the percentage of unique molecules in all generated samples) and novelty (the percentage of generated molecules that are not in the training set).

Following previous work, we randomly sample $10k$ molecules for ZINC and $30k$ for MOSES, and compare on the three aforementioned metrics. Generations results on ZINC and MOSES are shown in Table 1 and Table 2 respectively. `str-smi` denotes string-based SMILES representations and `str-sfi` denotes string-based SELFIES representations. In Table 1, we show generation results for both SMILES and SELFIES. SMILES does not have a validity constraint during generation. However, our model, MolEBM, still achieves 95.5% validity, which outperforms other SMILES-based methods and is also comparable to those with valency check. This result demonstrates that our model can capture those valency rules effectively and implicitly (also see Appendix §A.4.). Samples from our model also achieve perfect uniqueness and novelty.

Model	**Representation**	**Validity**	**Novelty**	**Uniqueness**
JT-VAE (Jin et al., 2018)	Graph	1.000*	1.000	1.000
GCPN (You et al., 2018)	Graph	1.000*	1.000	1.000
GraphNVP (Madhawa et al., 2019)	Graph	0.426	1.000	0.948
GraphAF (Shi et al., 2020)	Graph	1.000*	1.000	0.991
GraphDF (Luo et al., 2021)	Graph	1.000*	1.000	1.000
ChemVAE (Gómez-Bombarelli et al., 2018)	`str-smi`	0.170	0.980	0.310
GrammarVAE (Kusner et al., 2017)	`str-smi`	0.310	1.000	0.108
SDVAE (Dai et al., 2018)	`str-smi`	0.435	-	-
MolEBM	`str-smi`	0.955	1.000	1.000
MolEBM	`str-sfi`	1.000	1.000	1.000

Table 1: Unconditional generation on ZINC. * denotes valency check.

Model	**Representation**	**Validity**	**Novelty**	**Uniqueness**
JT-VAE (Jin et al., 2018)	Graph	1.000*	0.914	1.000
GraphAF (Shi et al., 2020)	Graph	1.000*	1.000	0.991
GraphDF (Luo et al., 2021)	Graph	1.000*	1.000	1.000
LIMO (Eckmann et al., 2022)	`str-sfi`	1.000	1.000	0.976
MolEBM	`str-sfi`	1.000	1.000	1.000

Table 2: Unconditional generation on MOSES. * denotes valency check. Results obtained from (Polykovskiy et al., 2020; Eckmann et al., 2022).

4.3 Single-Objective Optimization

Non-Biological Property Optimization. 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 the Penalized logP scores have a positive relationship with the length of the molecules, we

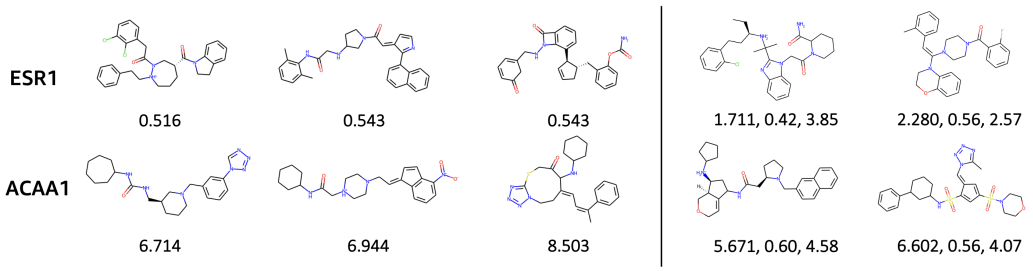


Figure 2: Generated molecules with high binding affinities. Left: Top-3 single-objective optimized molecules. The number denotes $\mathbf{K_D}$ in nmol/L. Right: Top-2 multi-objective optimized molecules. Numbers denote $\mathbf{K_D}$ in nmol/L, QED and SA respectively.

maximize Penalized 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 9, we can see that with length limit, MolEBM outperforms previous methods by a large margin. Maximizing penalized logP without length limit leads to the highest Penalized logP, 158.0. 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 Appendix §A.3.

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
MolEBM	✓	**26.4**	**25.1**	**24.4**	**0.948**	**0.948**	**0.948**
MolEBM	✗	**158.0**	**157.8**	**157.5**	**0.948**	**0.948**	**0.948**

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

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, while ACAA1 does not. However, we did not use any binder-related information in the design process. Binding affinity is measured by the estimated dissociation constants $\mathbf{K_D}$, which can be computed with AutoDock-GPU (Santos-Martins et al., 2021) given a molecule. Large binding affinities corresponds to small $\mathbf{K_D}$. That is, we aim to minimize $\mathbf{K_D}$. Table 4 shows that our model outperforms previous methods on both ESR1 and ACAA1 binding affinity maximization tasks. Producing those ligands with high binding affinity plays a vital role in the early stage of drug discovery.

Method	**ESR1** $\mathbf{K_D}$ (↓)			**ACAA1** $\mathbf{K_D}$ (↓)		
	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
MolEBM	**0.52**	**0.54**	**0.54**	**6.71**	**6.94**	**8.50**

Table 4: Biological single-objective optimization. Report top-3 lowest $\mathbf{K_D}$ (in nanomoles/liter) found by each model. Baseline results obtained from (Eckmann et al., 2022).

4.4 MULTI-OBJECTIVE OPTIMIZATION

We next consider optimizing binding affinity, QED and SAS simultaneously for multi-objective optimization. Following Eckmann et al. (2022), we exclude molecules with abnormal behaviors [1] in the generation process to make sure the learned distribution shifts towards a desirable region in terms of pharmacologic and synthetic properties.

Table 5 shows our multi-objective binding affinity maximization results comparing to LIMO (Eckmann et al., 2022) and GCPN (You et al., 2018). From the results, we can see that MolEBM 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 rather low $\mathbf{K_D}$. Compared to existing methods, MolEBM 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. When comparing ACAA1 multi-objective setting with its corresponding single-objective one, we find multi-objective results even outperforms the single-objective one, which may be counter-intuitive since multi-objective optimization is assumed to be a harder task than single-objective optimization. However, our current results indicate that with proper prior knowledge (e.g. multiple objectives are positively aligned), multi-objective settings could be more plausible in *de novo* design, and those complicated prior knowledge can indeed be captured by our expressive latent space EBM. While we still need domain expertise to determine the effectiveness of those ligands discovered by MolEBM, we believe our model could shed light on the optimization procedure in the early drug discovery.

Ligand	**ESR1**			**ACAA1**		
	$\mathbf{K_D}$ (↓)	**QED** (↑)	**SA** (↓)	$\mathbf{K_D}$ (↓)	**QED** (↑)	**SA** (↓)
GCPN 1st	810	0.43	4.2	8500	0.69	4.2
GCPN 2nd	2.7×10^4	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
Tamoxifen	87	0.45	2.0	–	–	–
Raloxifene	7.9×10^6	0.32	2.4	–	–	–
MolEBM 1st	**1.71**	0.42	3.85	**5.67**	0.60	4.58
MolEBM 2nd	**2.28**	0.56	2.56	**6.60**	0.56	4.07

Table 5: Muli-Objective Binding Affinity Maximization for both ESR1 and ACAA1. Report Top-2 average scores related to $\mathbf{K_D}$ (in nmol/L), QED and SA. Baseline results obtained from (Eckmann et al., 2022).

Figure 2 shows generated molecules with high binding affinities. See Appendix for more examples.

5 CONCLUSION AND DISCUSSION

We propose a deep generative model, MolEBM, which models the joint distribution of molecules and molecular properties. It assumes an energy-based prior for a low-dimensional continuous latent space, which effectively captures data regularities of the discrete molecule data. We then design a distribution shifting method (SGDS) 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 by a significant margin.

A limiting factor is that the sampling with gradual distribution shifting (SGDS) requires that the properties of interest can be easily computed. In our future work, we shall explore semi-supervised learning methods using our model where the properties are available only for a small sample of molecules.

Another limitation is that, for our optimization method, good property scores may not translate to useful molecules for drug design. This problem may be partially addressed by multi-objective optimization as studied in this paper. In our work, we focus on optimizing given objectives. Designing good objectives or metrics is an equally or perhaps even more important problem that deserves careful investigation.

[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.

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 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, 2019. 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.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

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, pp. 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, pp. 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*, pp. 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*, pp. 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.

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*, pp. 2323–2332. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International conference on machine learning*, pp. 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*, pp. 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*, pp. 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*, pp. 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.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

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.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

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*, pp. 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*, pp. 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*, pp. 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.

A APPENDIX

A.1 DETAILS ABOUT MODEL AND LEARNING

Our model is of the form $p_\alpha(z)p_\beta(x|z)p_\gamma(y|z)$. The marginal distribution of (x, y) is

$$p_\theta(x, y) = \int p_\theta(x, y, z)dz = \int p_\alpha(z)p_\beta(x|z)p_\gamma(y|z)dz.$$

We use $p_\theta(x, y)$ to approximate the data distribution of (x, y).

For the data distribution of (x, y), y is a deterministic function of x. However, a machine learning method usually cannot learn the deterministic function exactly. Instead, we can only learn a probabilistic $p_\theta(y|z)$. Our model $p_\theta(x, y)$ seeks to approximate the data distribution $p(x, y)$ by maximum likelihood. A learnable and flexible prior model $p_\alpha(z)$ helps to make the approximate more accurate than a fixed prior model such as that in VAE.

Let the training data be $\{(x_i, y_i), i = 1, ..., n\}$. The log-likelihood function is $L(\theta) = \sum_{i=1}^n \log p_\theta(x_i, y_i)$. The learning gradient is $L'(\theta) = \sum_{i=1}^n \nabla_\theta \log p_\theta(x_i, y_i)$. In the following, we provide details for calculating $\nabla_\theta \log p_\theta(x, y)$ for a single generic training example (x, y) (where we drop the subscript $_i$ for notation simplicity).

$$\begin{aligned}
\nabla_\theta \log p_\theta(x, y) &= \frac{1}{p_\theta(x, y)} \nabla_\theta p_\theta(x, y) \\
&= \frac{1}{p_\theta(x, y)} \int \nabla_\theta p_\theta(x, y, z)dz \\
&= \frac{1}{p_\theta(x, y)} \int p_\theta(x, y, z) \nabla_\theta \log p_\theta(x, y, z)dz \\
&= \int \frac{p_\theta(x, y, z)}{p_\theta(x, y)} \nabla_\theta \log p_\theta(x, y, z)dz \\
&= \int p_\theta(z \mid x, y) \nabla_\theta \log p_\theta(x, y, z)dz \\
&= \mathbb{E}_{p_\theta(z|x,y)} \left[\nabla_\theta \log p_\theta(x, y, z)\right] \\
&= \mathbb{E}_{p_\theta(z|x,y)} \left[\nabla_\theta (\log p_\alpha(z) + \log p_\beta(x|z) + \log p_\gamma(y|z))\right].
\end{aligned}$$

For the prior model,

$$\begin{aligned}
\nabla_\alpha \log p_\alpha(z) &= \nabla_\alpha f_\alpha(z) - \nabla_\alpha \log Z(\alpha) \\
&= \nabla_\alpha f_\alpha(z) - \frac{1}{Z(\alpha)} \nabla_\alpha Z(\alpha) \\
&= \nabla_\alpha f_\alpha(z) - \frac{1}{Z(\alpha)} \int \nabla_\alpha \exp(f_\alpha(z)) p_0(z)dz \\
&= \nabla_\alpha f_\alpha(z) - \int \nabla_\alpha f_\alpha(z) \frac{1}{Z(\alpha)} \exp(f_\alpha(z)) p_0(z)dz \\
&= \nabla_\alpha f_\alpha(z) - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)].
\end{aligned}$$

Thus the learning gradient for α given an example (x, y) is

$$\delta_\alpha(x, y) = \nabla_\alpha \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\alpha f_\alpha(z)] - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]. \tag{17}$$

The above equation has an empirical Bayes nature. $p_\theta(z|x, y)$ is based on the empirical observation (x, y), while p_α is the prior model. For the generation model,

$$\delta_\beta(x, y) = \nabla_\beta \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\beta \log p_\beta(x|z)]. \tag{18}$$

Similarly, for the regression model,

$$\delta_\gamma(x, y) = \nabla_\gamma \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\gamma \log p_\gamma(y|z)]. \tag{19}$$

Estimating expectations in the above equations requires Monte Carlo sampling of the prior model $p_\alpha(z)$ and the posterior distribution $p_\theta(z|x, y)$. If we can draw fair samples from the two distributions, and use these Monte Carlo samples to approximate the expectations, the the gradient ascent algorithm based on the Monte Carlo samples is the stochastic gradient ascent algorithm or the stochastic approximation algorithm of Robbins and Monro (Robbins & Monro, 1951), who established the convergence of such an algorithm to a local maximum of the log-likelihood.

For MCMC sampling using Langevin dynamics, the finite step or short run Langevin dynamics may cause bias in Monte Carlo sampling. The bias was analyzed in Pang et al. (2020). The resulting algorithm is an approximate maximum likelihood learning algorithm.

A.2 Ablation Studies

We conduct ablations on the key components of our method: 1) EBM Prior (EBM vs. standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$), (2) SGDS (shift distributions with SGDS vs. no shift), and (3) joint training (joint training of molecule and molecular property distribution vs. train molecule with LSEBM and learn property regression model in a second step). The ablation results are displayed in Table 6. It is clear that all the proposed components contribute significantly to the good performance of our method.

EBM Prior	**SGDS**	**Joint Training**	**Penalized logP**		
			1st	2nd	3rd
✗	✓	✓	13.81	13.78	13.75
✓	✗	✓	12.27	12.02	11.93
✓	✓	✗	12.79	12.70	12.41
✓	✓	✓	**26.37**	**25.05**	**24.38**

Table 6: Ablation studies.

A.3 Illustration of Sampling with Gradual Distribution Shifting (SGDS)

Figure 3 shows property score densities of sampled molecules from our MolEBM in the distribution shifting process. We can see the model distribution is gradually shifting towards the region supported by molecules with high property values.

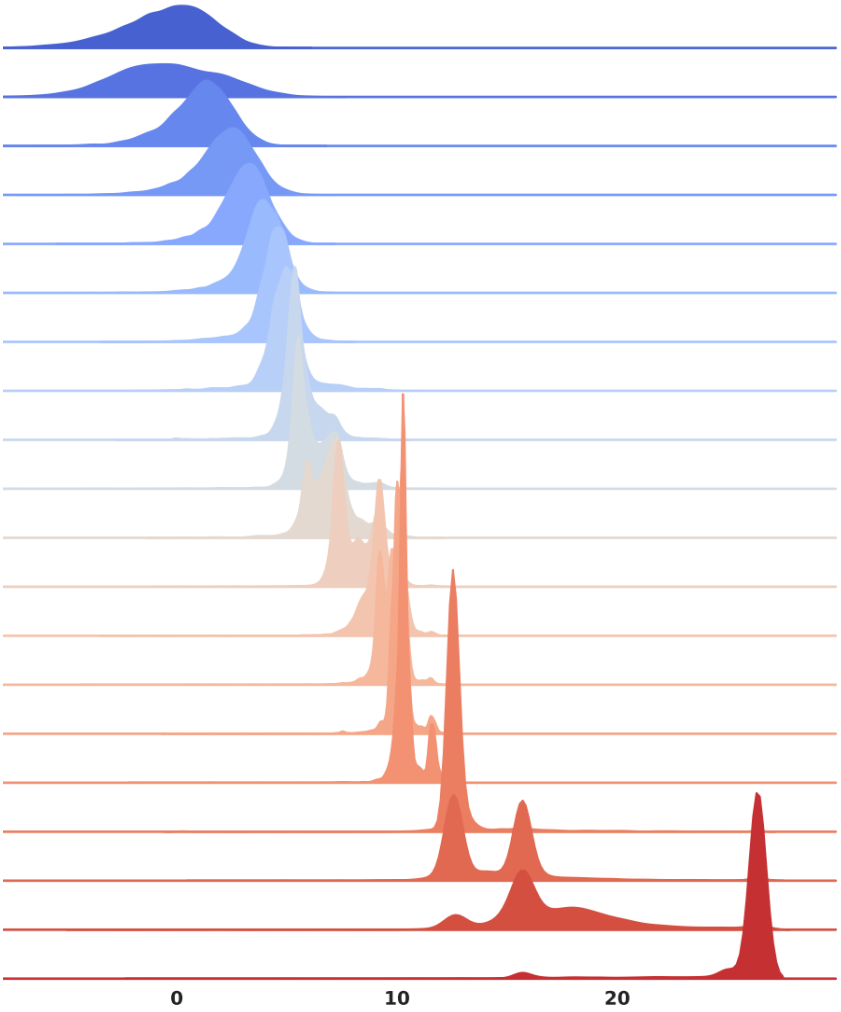


Figure 3: Illustration of SGDS in a single-objective penalized logP optimization experiment.

A.4 Unconditional Generation

For unconditional generation task, we use uniqueness, novelty and validity to compare our generation results with existing methods. Meanwhile, for SELFIES-based ZINC dataset, we split this dataset into train split ($240k$) and test split ($10k$ samples). Here, we randomly sample $10k$ molecules from the learned latent EBM and calculate their logP, QED and SA scores using RDKit. We compare these property densities with molecule property densities in test split. The results are shown in Figure 4. We can see that the marginal distributions (property score densities) of the learned model match those of the data quite well, implying that our model indeed captures the chemical rules implicitly.

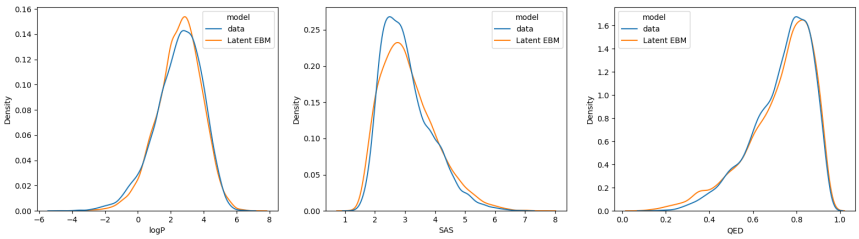


Figure 4: Property score distributions.

A.5 GENERATION WITH DIFFERENT LENGTH OF MARKOV CHAIN

In our experiments, we use short-run MCMC (i.e. $s = 20$) in Equation 13 for all experiments. From Figure 5, we can see with the increasing length of Markov chains, the molecules change accordingly, showing that Markov Chain doesn't get stuck in the local mode.

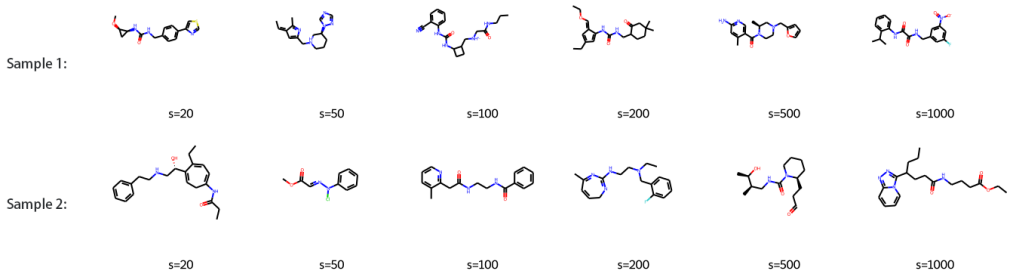


Figure 5: Sampled molecules with the different length of Markov chain.

A.6 TRAINING TIME

Maximum likelihood learning of MolEBM takes 4 hours, to train 30 iterations on a single Nvidia Titan XP GPU with batch size 2048. For non-biological single-objective property optimization, it takes around 0.5 hours to do 50 distribution shifting (SGDS) iterations. For biological binding affinity maximization, the optimization time is mainly dependent on the number of queries of AutoDock-GPU. We do 20 and 10 SGDS iterations for the single-objective and multi-objective tasks, respectively, which cost 10 hours and 5 hours. For biological property optimization tasks, we use two Nvidia Titan XP GPUs, one for running our code and another one for running AutoDock-GPU. We have added a table to compare with previous methods.

Model	Penalized-logP/QED
JT-VAE	24
GCPN	8
MolDQN	24
GraphDF	8
Mars	12
LIMO	1
MolEBM	4.5

Table 7: Comparison of molecule generation time in (hrs). Results obtained from (Eckmann et al., 2022).

Even if we intensively use MCMC based sampling methods, our training speed is affordable comparing to existing methods. That's due to our designed latent space EBM is low-dimensional and we use short-run MCMC (i.e. with fixed iteration steps $s = 20$) in our experiments. The sampling results with respects to the length of Markov chain is discussed in previous section.

A.7 OPTIMIZATION WITH CONSTRAINTS

In the unconditional generation task, even with SMILES representation, our model can capture the valency constraints in chemical space. This idea can further be extended to constrained optimizations, and those constraints in the chemical space can be directly imposed by only keeping the sampled molecules that satisfy the constraints during the SGDS molecule sampling step in Algorithm 2. We include logP targeting and similarity-constrained optimization experiments in the following.

For example, in logP targeting experiments, during each iteration, we only keep the molecules within the target logP range; in similarity-constrained experiments, we only use those molecules which satisfy similarity constraints. By iteratively updating the model with those selected molecules, we shift the joint distribution towards the target that satisfies the constraints.

A.7.1 LOGP TARGETING

Comparing to previous methods, MolEBM is able to get second best diversity scores with significantly better success rate in both ranges. That's 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, MolEBM achieves rather high diversity scores while keeping most of the sampled molecules within the logP range.

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**	–	–
MolEBM	**86.0%**	0.874	**62.2%**	0.858

Table 8: logP targeting to a certain range (Eckmann et al., 2022; You et al., 2018; Luo et al., 2021; Xie et al., 2021).

A.7.2 SIMILARITY-CONSTRAINED OPTIMIZATION

Following previous procedures for JT-VAE (Jin et al., 2018), we select 800 molecules with the lowest penalized-logP scores in ZINC250k dataset. This experiments aim to generative novel molecules with high penalized-log while similarity to the target molecules. We first randomly select one molecule as the target, and then optimize the p-logP. Within each SGDS step, we only keep the molecules that have similarity score greater than minimum value δ.

δ	GCPN		GraphDF		LIMO		MolEBM	
	Improv.	% Succ.	Improv.	% Succ.	Improv.	% Succ.	Improv.	% Succ.
0.0	4.2 ± 1.3	**100**	5.9 ± 2.0	**100**	10.1 ± 2.3	**100**	$\mathbf{19.11} \pm 2.12$	**100**
0.2	4.1 ± 1.2	**100**	5.6 ± 1.7	**100**	5.8 ± 2.6	99.0	$\mathbf{7.41} \pm 1.89$	**100**
0.4	2.5 ± 1.3	**100**	$\mathbf{4.1} \pm 1.4$	**100**	3.6 ± 2.3	93.7	3.80 ± 1.44	97.5

Table 9: Non-biological single-objective optimization. Report top-3 highest scores found by each model. LL (Length Limit) denotes whether the model has the limit of maximum length. Baseline results obtained from (Eckmann et al., 2022; Luo et al., 2021).

A.8 ADDITIONAL EXPERIMENTS ON GUACAMOL BENCHMARKS

We further evalute our MolEBM on several distribution learning and goal-directed optimization tasks in GuacaMol benchmark (Brown et al., 2019).

To be specific, for distribution learning, we use the validity, uniqueness and novelty to evaluate our model. The results are shown in Table 10. Comparing to existing methods, our MolEBM trained using SELFIES representations achieve highest scores among all three tasks.

For goal-directed benchmarks, we select five multiple property optimization tasks (MPO). Due to time limit, we just run the experiments without carefully fine-tuning the hyper-parameters. However, our MolEBM is also able to achieve comparable results to the strong baselines as Graph GA.

Again, GuacaMol is an extremely useful benchmark since it provides the normalized scoring functions which is a weighed sum of multiple diverse properties of interest. The normalized scoring functions enable the fast convergence of MolEBM. With GuacaMol, we can also investigate the design choice between single property regression network to predict the predefined scores and multiple property regression networks to predict original chemical properties individually. We leave this for future studies.

A.8.1 DISTRIBUTION-LEARNING BENCHMARKS

Benchmark	AAE	Graph MCTS	Random Sampler	SMILES LSTM	VAE	**MolEBM**
Validity	0.822	**1.000**	**1.000**	**1.000**	0.959	**1.000**
Uniqueness	**1.000**	**1.000**	0.997	**1.000**	0.999	**1.000**
Novelty	0.998	0.994	0.000	0.912	0.971	**0.999**

Table 10: Distribution learning results on GuacaMol benchmarks (Brown et al., 2019).

A.8.2 GOAL-DIRECTED BENCHMARKS

Benchmark	Best of Dataset	SMILES GA	Graph MCTS	Graph GA	SMILES LSTM	**MolEBM**
Osimertinib MPO	0.839	0.886	0.784	**0.953**	0.907	*0.933*
Fexofenadine MPO	0.817	0.931	0.695	**0.998**	0.959	*0.971*
Ranolazine MPO	0.792	0.881	0.616	*0.920*	0.855	**0.924**
Sitagliptin MPO	0.509	0.689	0.458	**0.891**	0.545	*0.829*

Table 11: Distribution learning results on GuacaMol benchmarks (Brown et al., 2019). Top-2 results are highlighted as **bold** and *italic* respectively.

A.9 GENERATED SAMPLES

Figure 6 and Figure 7 show generated molecules with high binding affinities towards ESR1 and ACAA1 respectively in single-objective property design.

Figure 8 and Figure 9 show generated molecules with high binding affinities towards ESR1 and ACAA1 respectively in multi-objective property design.

Comparing to the previous state-of-the-art methods, our MolEBM is able to produce more high quality molecules than top-3 only because after sampling with gradual distribution shifting (SGDS), MolEBM is able to locate at the area supported by molecules with high binding affinities.

In single-objective design, we find that few generated molecules may be of less practical use due to undesired properties (e.g. the first one in Figure 7 has a large circle). This observation is in accordance with (Eckmann et al., 2022), which is the case where the single-objective optimization is not sufficient. Thus we need multi-objective binding affinities design settings since in contrast to non-biological properties, binding affinities are hard to compute and optimize.

In multi-objective design settings, we find that those issues mentioned above can be largely solved by optimizing binding affinities, QED and SA at the same time.

Meanwhile, compared to previous generative model based methods, we use Langevin dynamics to infer the posterior distribution $p(z|x, y_1, \ldots, y_n)$ without bothering to design different encoders which is not trivial.

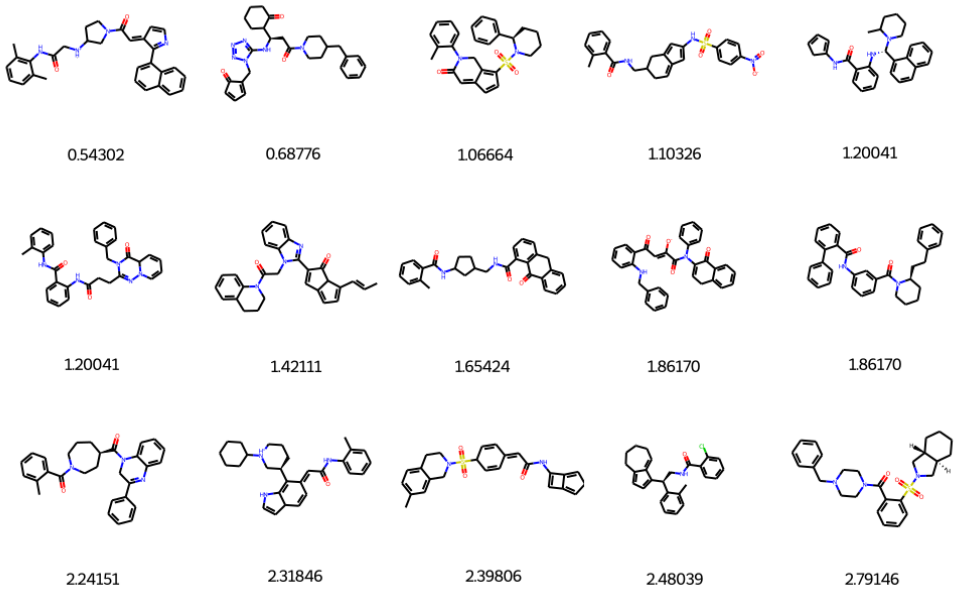


Figure 6: Generated molecules in **singe-objective** esr1 binding affinity maximization experiments with corresponding $\mathbf{K_D}(\downarrow)$ in nmol/L.

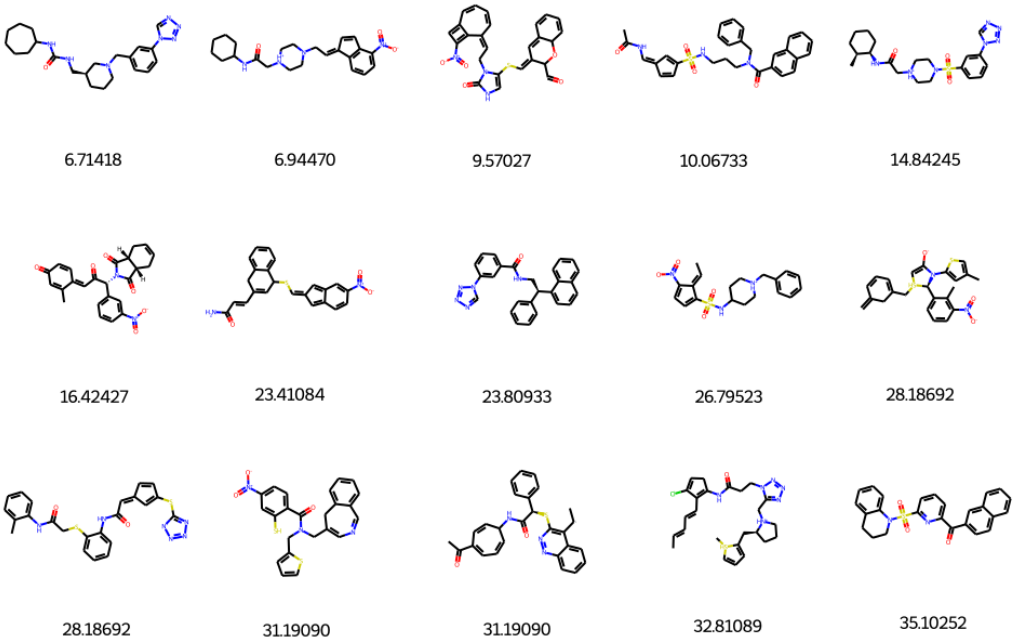


Figure 7: Generated molecules in **singe-objective** acaa1 binding affinity maximization experiments with corresponding $\mathbf{K_D}(\downarrow)$ in nmol/L.

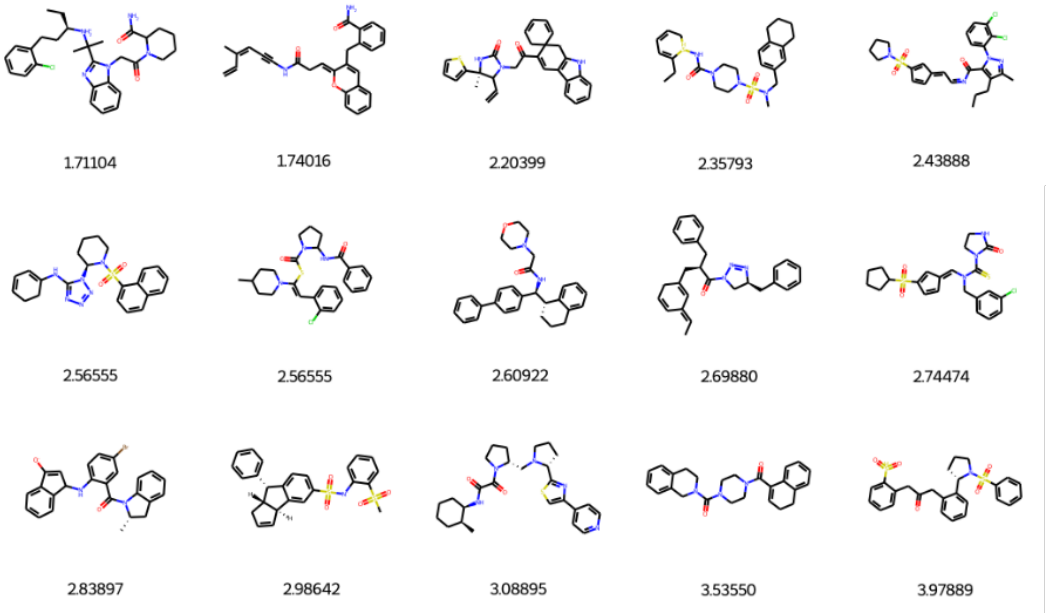


Figure 8: Generated molecules in **multi-objective** esr1 binding affinity maximization experiments with corresponding $\mathbf{K_D}(\downarrow)$ in nmol/L.

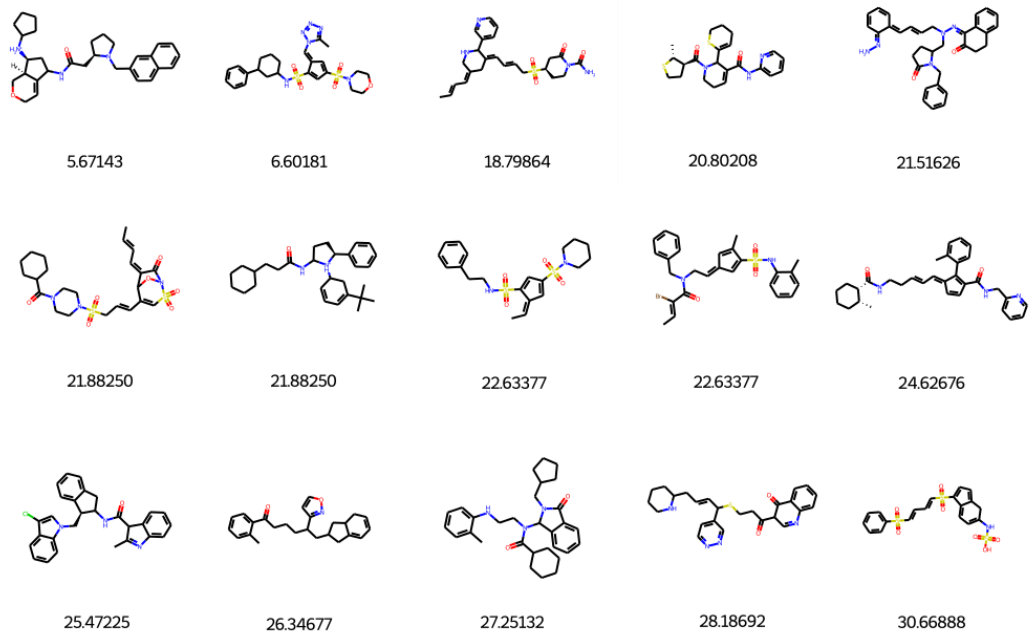


Figure 9: Generated molecules in **multi-objective** acaa1 binding affinity maximization experiments with corresponding $\mathbf{K_D}(\downarrow)$ in nmol/L.

A.10 TOP-3 MOLECULES IN P-LOGP AND QED OPTIMIZATION

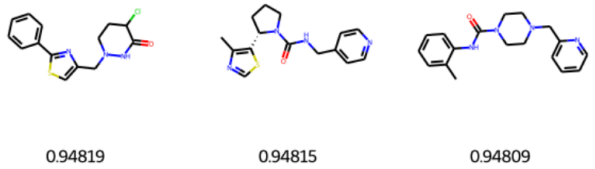


Figure 10: Top-3 molecules in single-objective QED maximization.

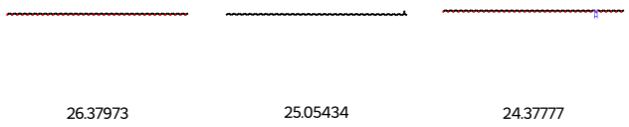


Figure 11: Top-3 molecules in single-objective p-logP maximization.

A.11 REPRODUCIBILITY

Our code and saved checkpoints can be found here [2].

[2] https://drive.google.com/drive/folders/1UQcXrLWo20wuBocCIEIq7RRm1p2-bb8H?usp=sharing