Machine Learning for Natural Sciences

Monsoon 2019

Multi-Objective de-novo Molecular Generation using Deep Reinforcement Learning

Submitted by
Jyotish P (20161217) Siddhartha L (201564122)

Contents

1 Introduction **2**
1.1 Objectives of the work 3

2 Methods **4**
2.1 Molecular generation using a single objective 4
2.1.1 Generative model 4
2.1.2 Predictive model 5
2.1.3 Using the predictive and the generative models 6
2.2 Molecular generation on multiple objectives 7
2.2.1 Scaling of rewards 7
2.2.2 Pareto Frontier 7
2.3 A trial using Actor-Critic 8

3 Results **10**
3.1 Single Objective Optimization 10
3.1.1 Partition Coefficient (LogP) 10
3.1.2 Benzene Rings 11
3.1.3 Melting Point 14
3.2 Multi-Objective Optimization 16
3.2.1 LogP - SA score - Benzene Rings 16
3.2.2 LogP - Benzene Rings - Melting Point 18
3.2.3 Test for parity optimality 20

4 Conclusion and Further Work **21**

1 Introduction

Getting a new drug to the market is a long and tedious process; it can take many years or even decades. There are many sorts of experiments, clinical studies, and clinical trials that one has go through. Also, about 90% of all clinical trials in humans fail even after the molecules have been successfully tested in animals. But to a first approximation, the process is as follows:

- The doctors study medical literature, in particular associations between drugs, diseases, and proteins published in other papers and clinical studies, and find out what the target for the drug should be, i.e., which protein it should bind with;
- Then, they can formulate what kind of properties they want from the drug: how soluble it should be, which specific structures it should have to bind with this protein, should it treat this or that kind of cancer
- They then think about which molecules might have these properties; there is a lot to choose from on this stage: e.g., one standard database lists 72 million molecules, complete with their formulas, some properties and everything; unfortunately, it doesn't always say whether a given molecule cures cancer, this we have to find out for ourselves;
- A set of molecules called lead molecules are actually sent to the lab for experimental validation;

The lab validates if the substance actually works and then the whole clinical trial procedure can be initiated; it is still very long and tedious, and only a small percentage of drugs actually go all the way through the funnel and reach the market, but at least there is hope. See figure 1

As one can see, this is an extremely time taken and costly process. We cannot hope to replace the entire pipeline (since live experiments will be essential for any drug to get released). But we can try to narrow down the initial search or get a smaller set of lead molecules, thereby reducing the cost and time invested in the early stages. Finding a set of feasible molecules from billions of options can be pretty daunting for any of the classical methods in drug discovery. Recent advances

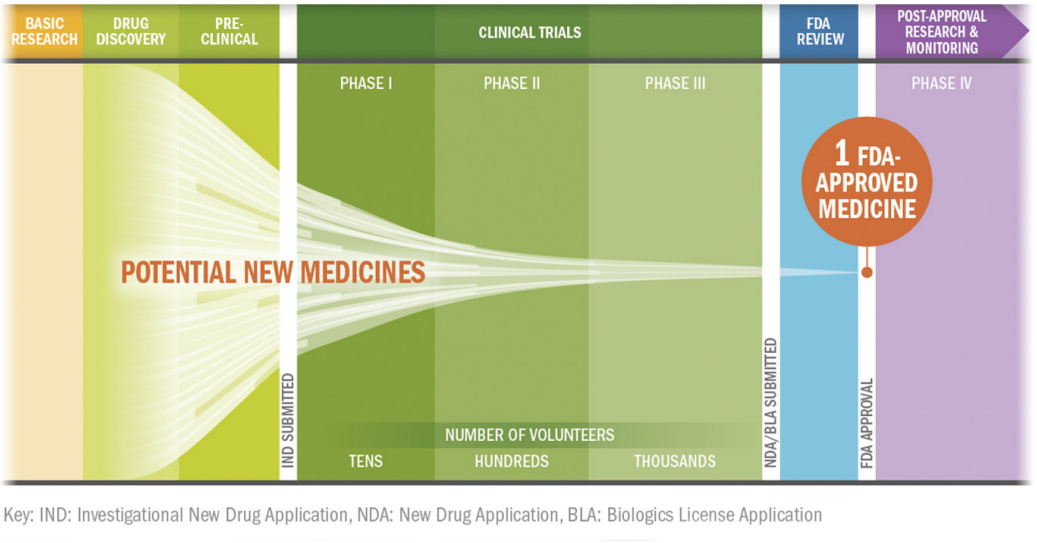


Figure 1: Procedure for drug discovery

in machine learning have shown promising results when it comes to such type of problems. So, machine learning models can be used to try and choose the molecules that are most likely to have desired properties.

1.1 Objectives of the work

- Given a set of molecular properties, generate a set of molecules that satisfy the given properties.
- The candidate molecules generated should satisfy multiple properties and not just a single property.
- The properties chosen for the study are logP (Octanol/Water partition coefficient), Synthetic accessibility score, Number of benzene rings, Melting point.

2 Methods

2.1 Molecular generation using a single objective

We first explain novel molecular generation based on a single target objective. For eg, say we want to generate a set of molecules that are in a certain range of partition coefficient. There are two models: A generative model and a predictive model.

2.1.1 Generative model

This model is used for generation of novel molecules as SMILES. The simplified molecular-input line-entry system (SMILES) is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings. Each character in the sequence is one-hot encoded and fed as an input to the generative model (or the generator).

GRU: Traditional Recurrent Neural Networks face the problem of vanishing gradients on large sequences due to gradients propagated over many layers. GRUs are modified recurrent neural networks that have additional gates to control gradients. The following are the operations of gru at a certain time step t. The input at the time is x_t and the updated hidden state of the RNN is h_t. r is the reset gate and z is the update gate.

$$z_t = \sigma(x_t U^z + h_{t-1} W^z) \tag{1a}$$

$$r_t = \sigma(x_t U^r + h_{t-1} W^r) \tag{1b}$$

$$\sim h_t = tanh(x_t U^h + (r_t * h_{t-1}) W^h) \tag{1c}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \sim h_t \tag{1d}$$

Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. If set the reset to all 1's and update gate to all 0's, it will arrive at the vanilla RNN model.

Pre-training of the generator: The generator is first pre-trained on SMILES present in the ChEMBL21 data-set that has about 1.5M smile molecules. In this stage, a SMILE is fed in as an input to the generator RNN and is trained to predict the input with a one time step delay. This is to say that, on providing start tag as an input to the generator, it must predict the first character of the SMILE sequence as the output and so on. The objective of this activity is to train the RNN network to learn the structure and grammar of SMILE sequences. See figure 2

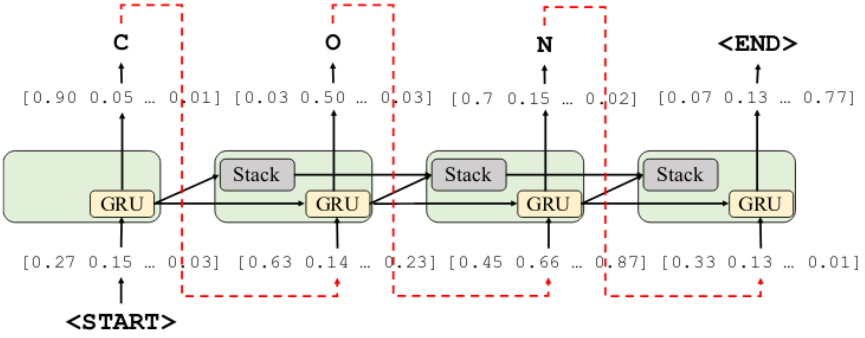


Figure 2: Pre-training of the generator

2.1.2 Predictive model

Apart from the pre-training of the generator, a different network called the predictive network is also trained. The predictive network is trained to predict a target property from a SMILE sequence input. The choice of network could vary but is typically chosen to be an RNN similar to the generator. The objective of training a predictive model is to evaluate the smiles generated by the generator which can be used to bias the generator. (see figure 3) The predictive model doesn't necessarily have to be a trained model. It could be a simple mathematical formula that can calculate the property, output of a molecular dynamics simulation or an ab-initio calculation. It is a procedure that can evaluate a smile with respect to a property.

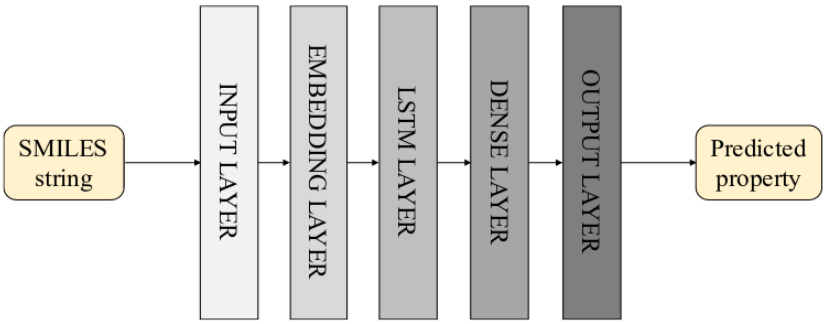


Figure 3: Predictive model for predicting target properties

2.1.3 Using the predictive and the generative models

Once the generator is pre-trained to predict valid smiles sequences and a predictive model available for evaluation of a property, the task now is to bias the generator to predict sequences that satisfy the target objective. For this, the policy gradient is used. See figure 4 **REINFORCE:** This is the algorithm that is used to optimize

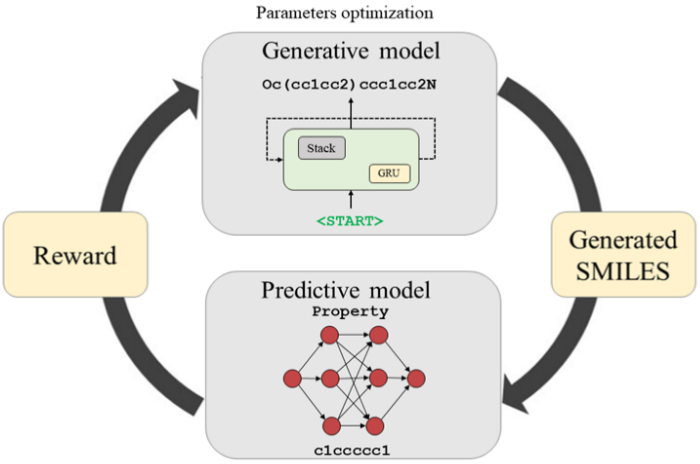


Figure 4: Combining the genrative and the predictive models with Policy Gradient

the parameters of the generator network. It has the following steps:

- Initialize the policy parameter θ at random.
- Generate one trajectory on policy π_θ : $S_1, A_1, R_2, S_2, A_2, \ldots, S_T$.

- For t=1, 2, . . . , T:
    1. Estimate the the return G_t
    2. Update policy parameters: $\theta \leftarrow \theta + \alpha\gamma t G_t \nabla_\theta ln\pi_\theta(At|St)$

For molecular generation, the policy network is the generator. The rewards G_t are calculated from the predictions of the predictive network. The policy network would eventually be biased to generate for the target objectives.

2.2 Molecular generation on multiple objectives

We experiment with multiple objectives using two techniques. One is a simple scaling of rewards technique and in the other technique, we try to generate a Pareto frontier

2.2.1 Scaling of rewards

The procedure is explained in the figure 5. We have the molecule generated from the generator. Now, the performance is evaluated by multiple predictive models. And finally, the rewards are scaled and fed as input to the generator.

2.2.2 Pareto Frontier

Here, we generate a Pareto frontier of the target objectives. Pareto optimality is a state of allocation of resources from which it is impossible to reallocate so as to make any one individual or preference criterion better off without making at least one individual or preference criterion worse off. Pareto frontier is the set of all Pareto optimal allocations. The frontier will be useful in determining the trade offs present while optimizing the objectives. For generating the Pareto frontier, we sample a multiple sets of random weights(sum to one) and use these weights while scaling the rewards. Now, we plot a 3D surface on the objectives optimized with different combination of weights to obtain the Pareto frontier.

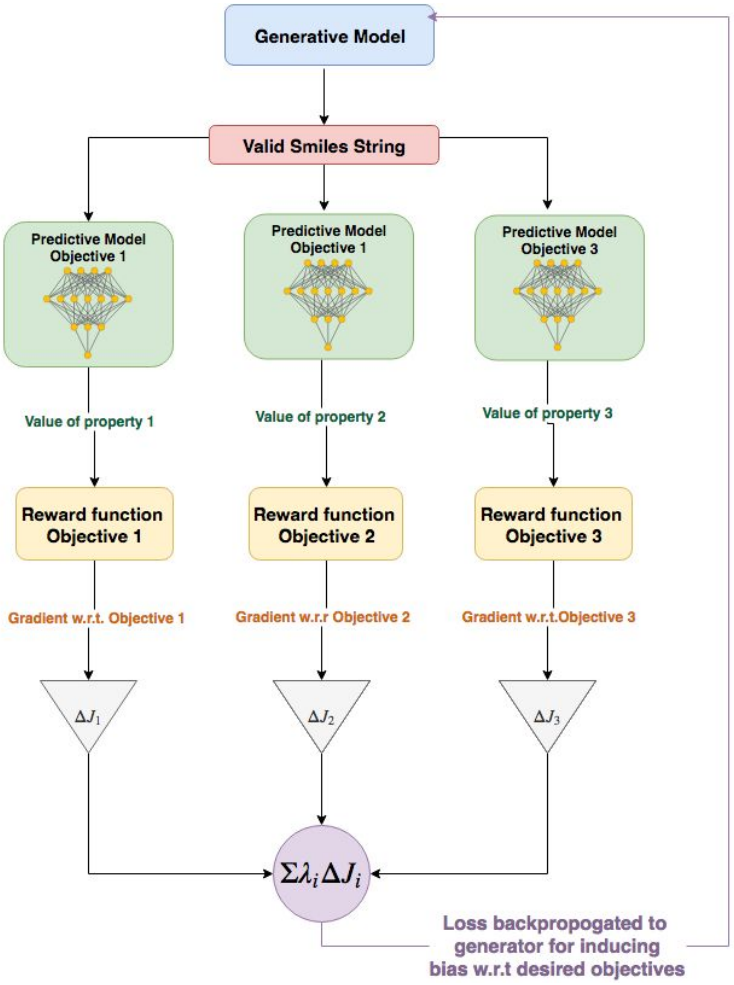


Figure 5: Multi-objective optimization through a simple scaling of rewards

2.3 A trial using Actor-Critic

Instead of optimizing using simple policy gradient, we also tried to optimize using the Actor-critic method. In this class of methods, instead of a simple Policy generator Network (Actor), we also have a critic Network. Here, the "Critic" estimates the value function whereas, the actor updates the policy distribution in the direction suggested by the critic. The algorithm 1 illustrates a simple Actor-Critic algorithm.

Algorithm 1 Q Actor Critic

Initialize parameters s, θ, w and learning rates α_θ, α_w

Sample $a \sim \pi_\theta(\alpha|s)$

for $i = 1, \ldots, T$ do **do**

Sample reward $R_t \sim R(s, a)$ and next state $s' \sim P(s'|s, a)$

Then sample the next action $a' \sim \pi_\theta(a', s')$

Update the policy parameters: $\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta log \pi_{theta}(a, s)$

Compute the correction (TD error) for action-value at time t:

$\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$

and use it to update the parameters of Q function:

$w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$

Move to $a \leftarrow a'$ and $s \leftarrow s'$

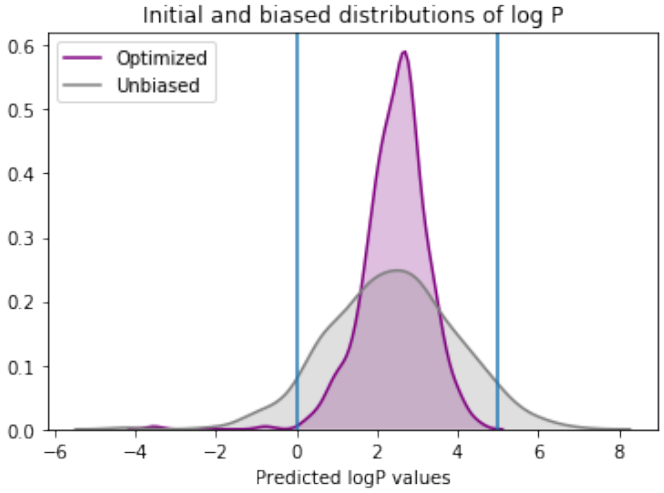
end for

3 Results

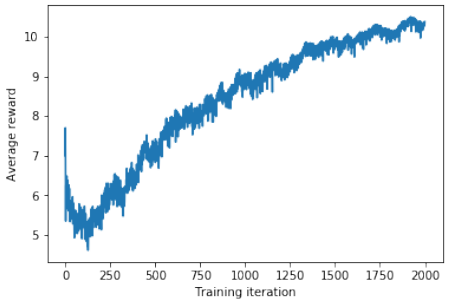
3.1 Single Objective Optimization

3.1.1 Partition Coefficient (LogP)

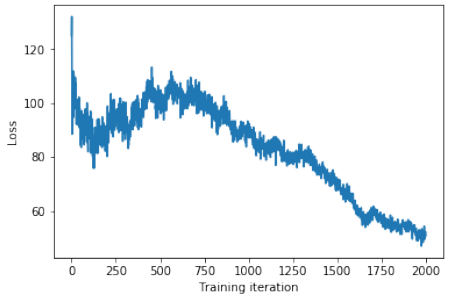
Restrict the logP value of the generated set within the range 0-5 (drug-like region).



(a) Distribution of logP with respect to the generated molecules



(b) Average rewards during training



(c) Average loss during training

Figure 6: Single objective optimization over LogP

% of molecules			Average similarity
Unique	**Invalid**	**In drug region**	
42.5	2.5	99.5	0.50

Table 1: Performance Metrics

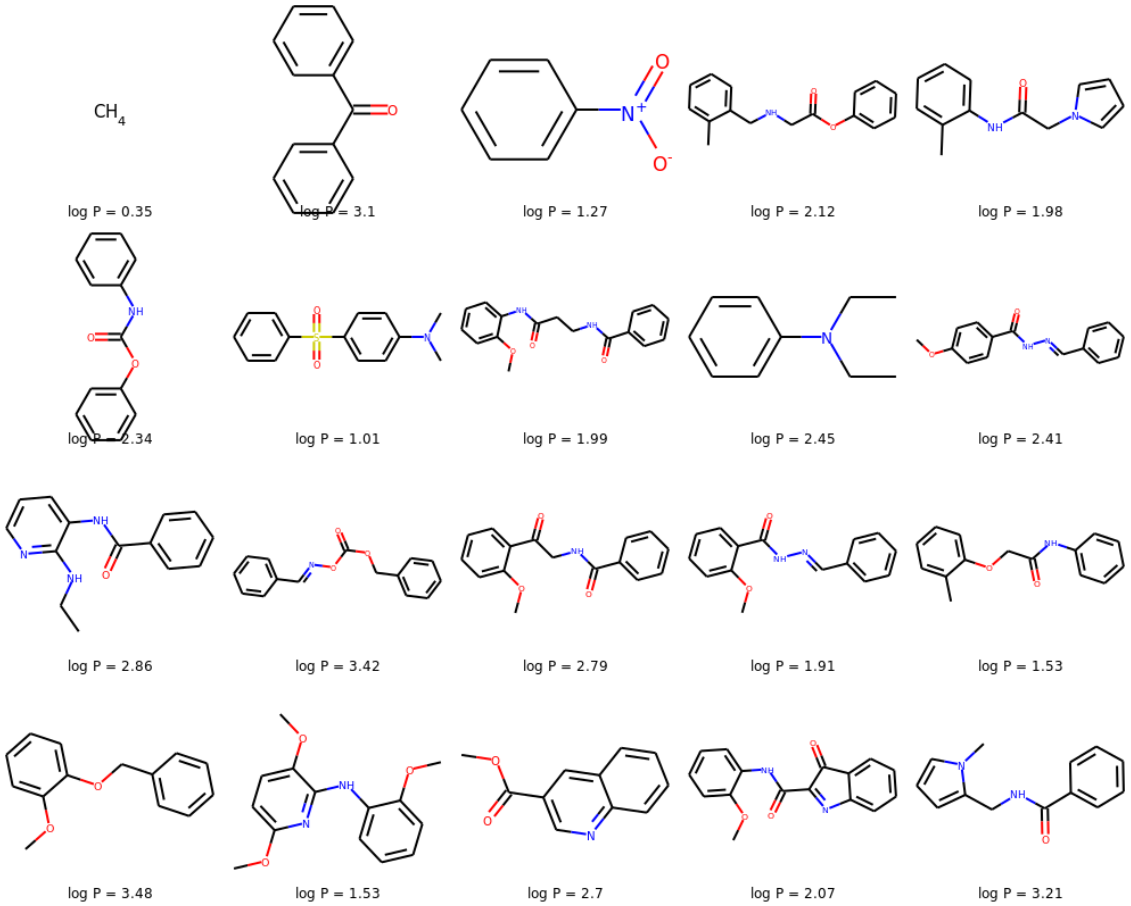
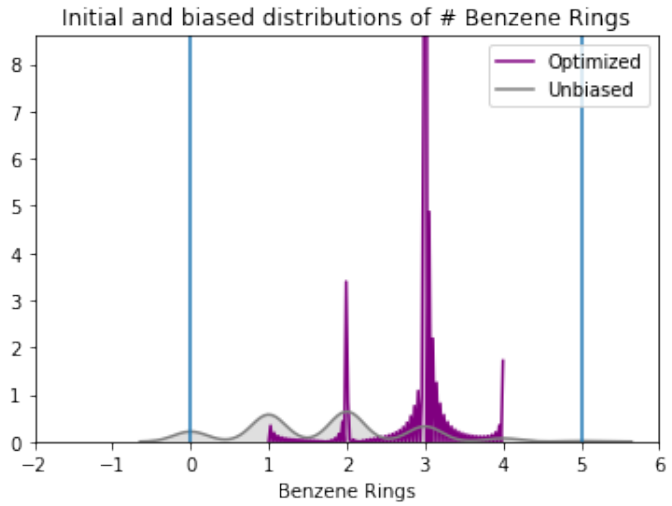


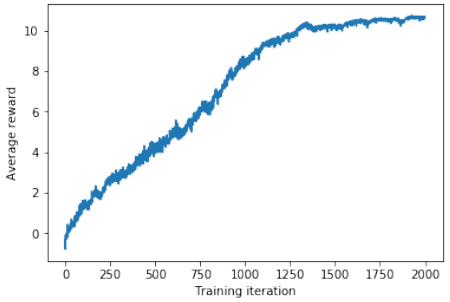
Figure 7: Molecules generated by biased generator

3.1.2 Benzene Rings

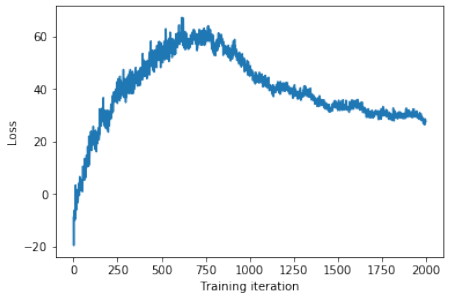
Generate Molecules that contain 3 benzene rings.



(a) Distribution of number of benzene rings with respect to the generated molecules



(b) Average rewards during training



(c) Average loss during training

Figure 8: Single objective optimization over number of benzene rings

% of molecules		Average similarity
Unique	**Invalid**	
23.3	4.5	0.70

Table 2: Performance metrics

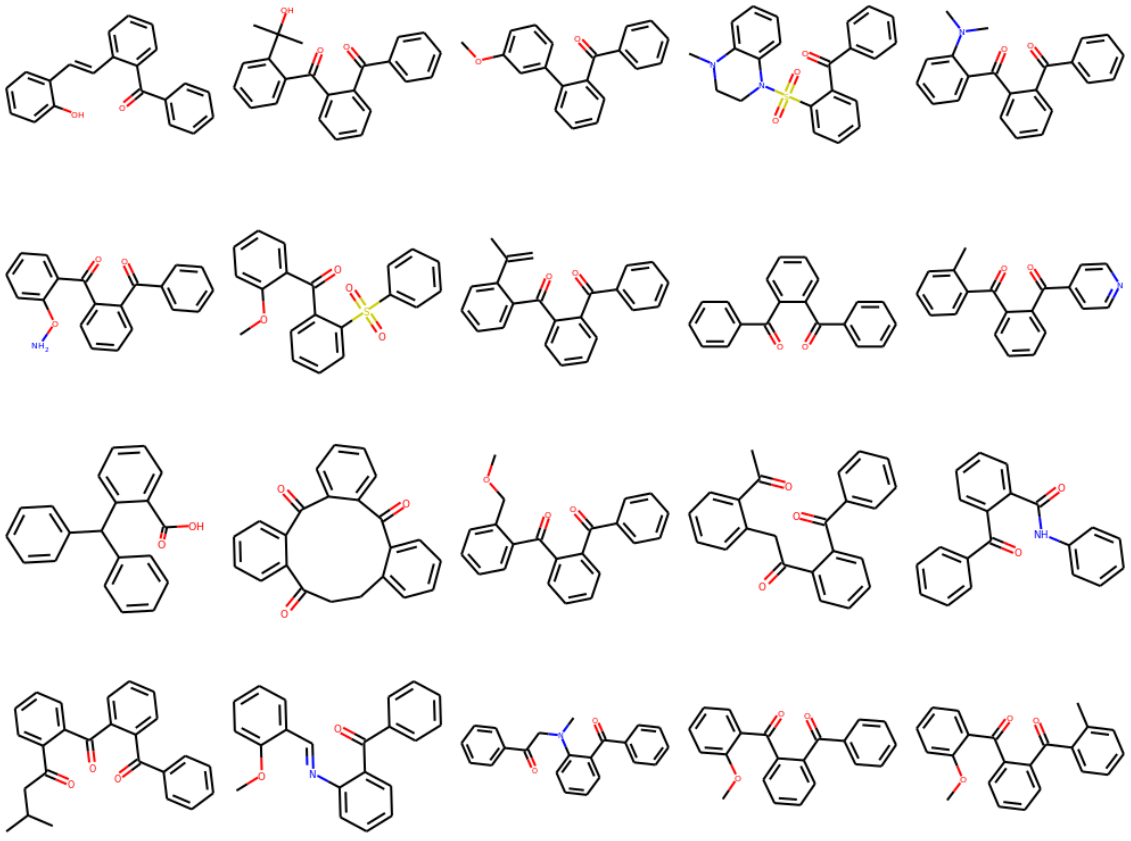
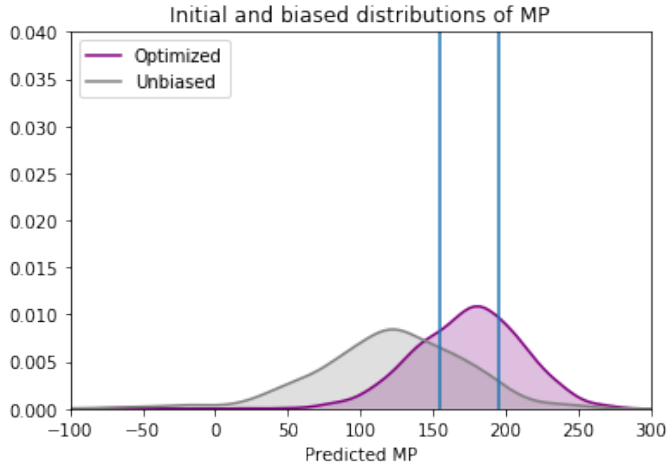


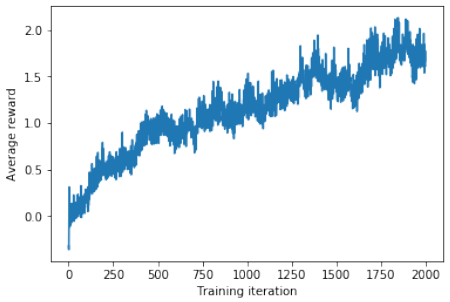
Figure 9: Molecules generated by biased generator

3.1.3 Melting Point

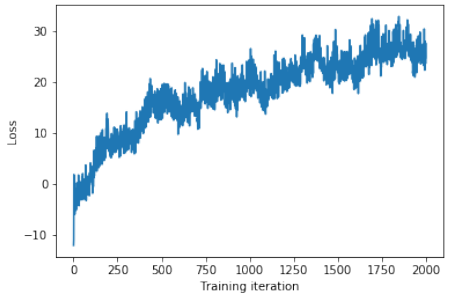
Restrict Melting Point of the generated molecules in between the region: 155 - 195 degrees Celsius.



(a) Distribution of melting point with respect to the generated molecules



(b) Average rewards during training



(c) Average loss during training

Figure 10: Single objective optimization over melting point

% of molecules		Average similarity
Unique	**Invalid**	
89.1	1	0.15

Table 3: Performance metrics

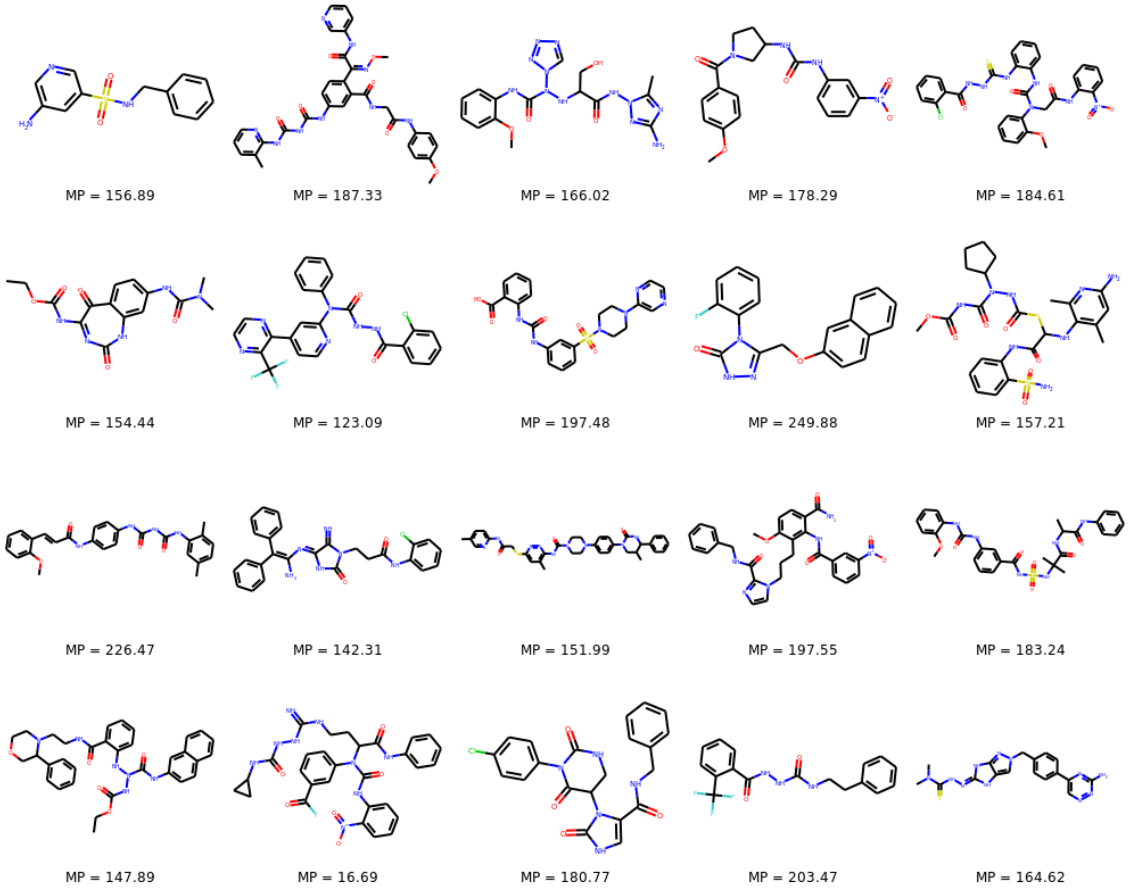


Figure 11: Molecules generated by biased generator

3.2 Multi-Objective Optimization

The experiments are performed over

- LogP - SA score - Benzene Rings
- LogP - Melting Point - Benzene Rings

3.2.1 LogP - SA score - Benzene Rings

% of molecules		Average similarity
Unique	**Invalid**	
32	2.5	0.42

Table 4: Performance metrics

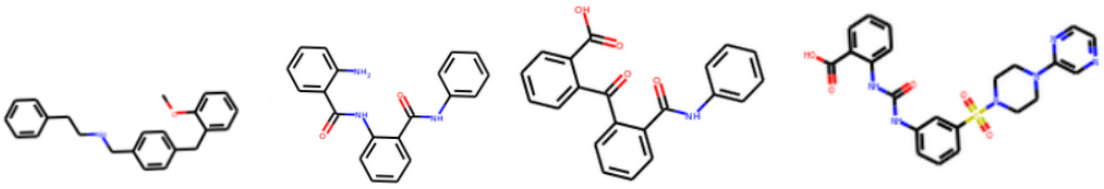
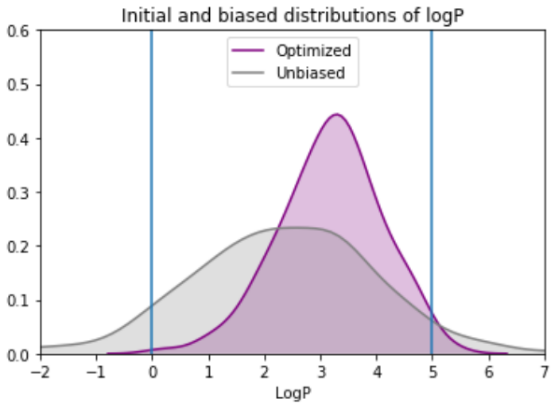
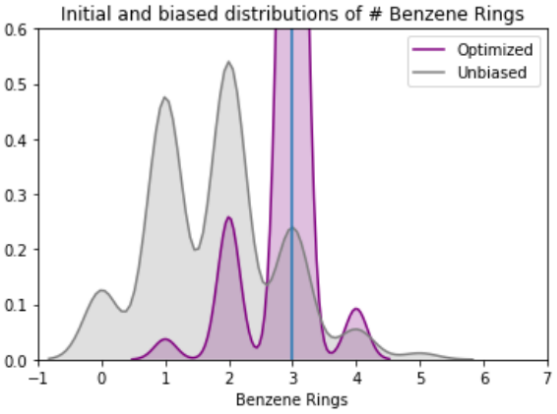


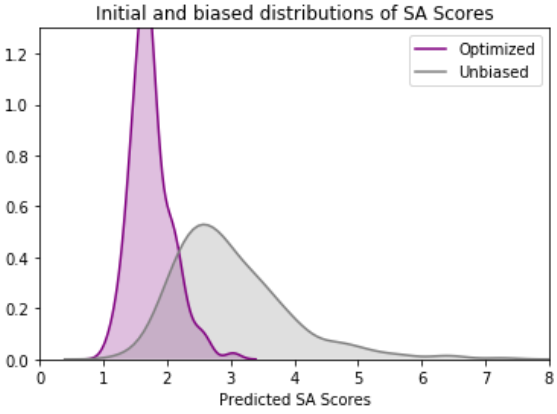
Figure 12: Molecules generated by biased generator



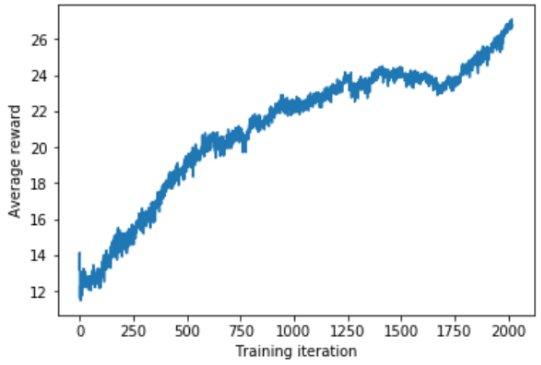
(a) Distribution of LogP



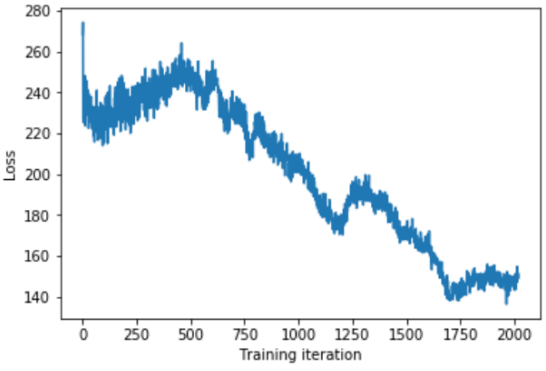
(b) Distribution of number of benzene rings



(c) Distribution of SA scores



(d) Average rewards during training



(e) Average loss during training

Figure 13: Multi-objective optimization with logP, number of benzene rings and synthetic accessibilty scores

3.2.2 LogP - Benzene Rings - Melting Point

% of molecules		Average similarity
Unique	Invalid	
18.5	0	0.76

Table 5: Performance metrics

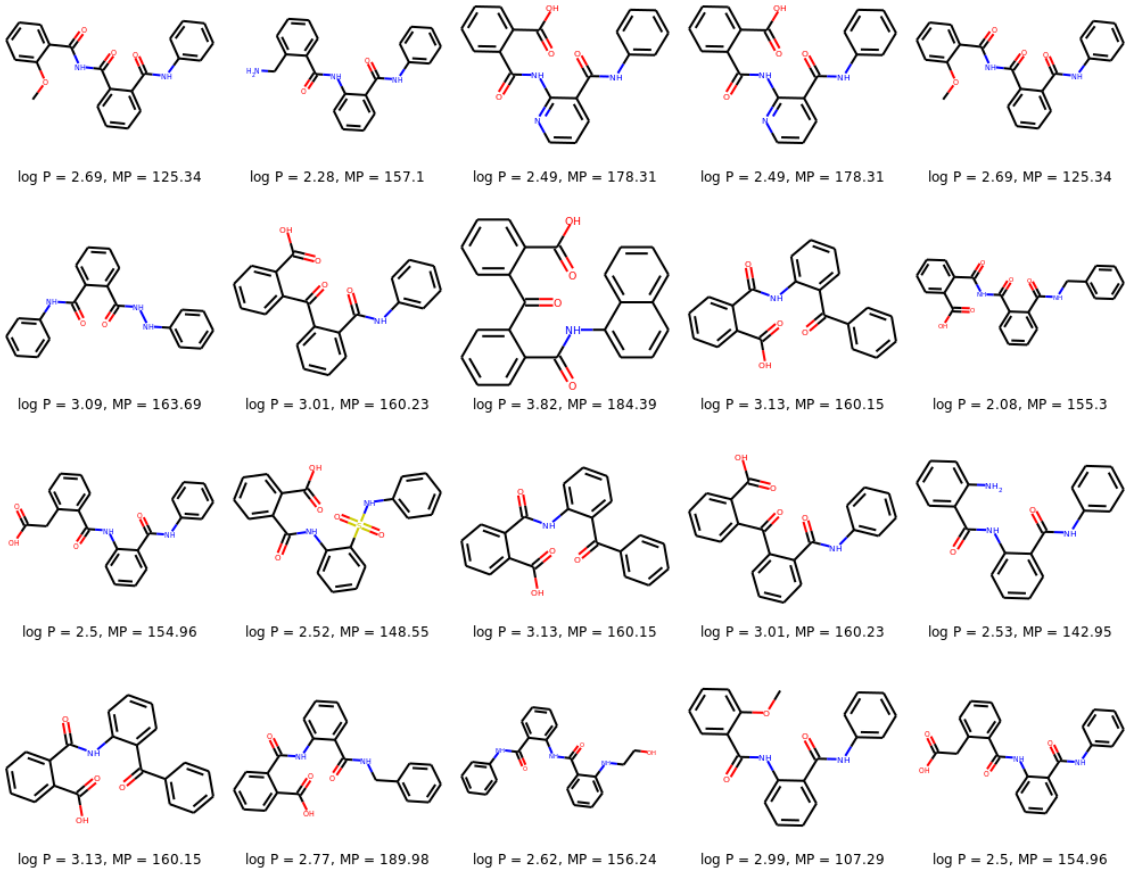
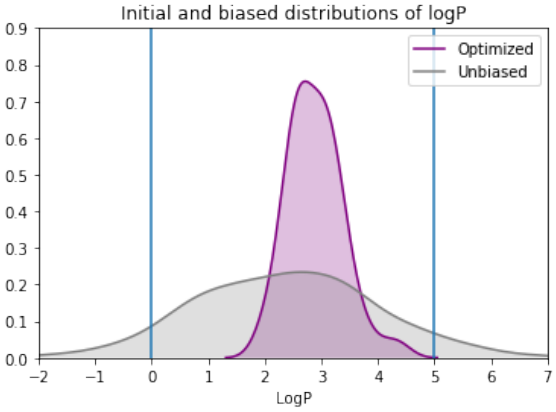
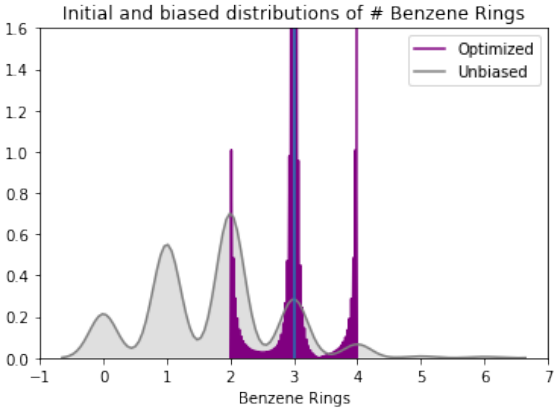


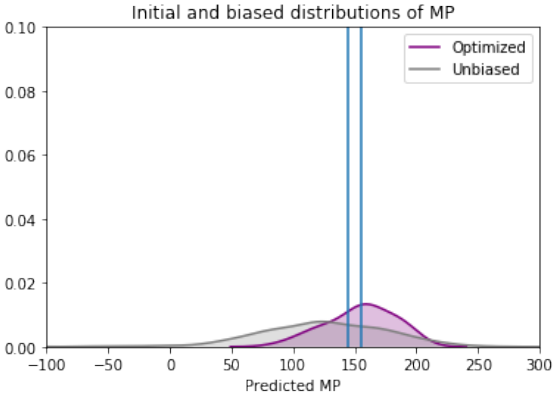
Figure 14: Molecules generated by biased generator



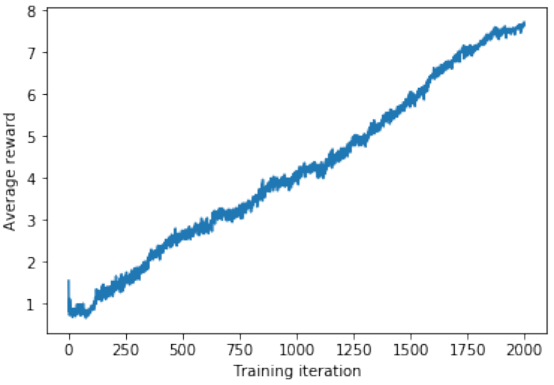
(a) Distribution of LogP



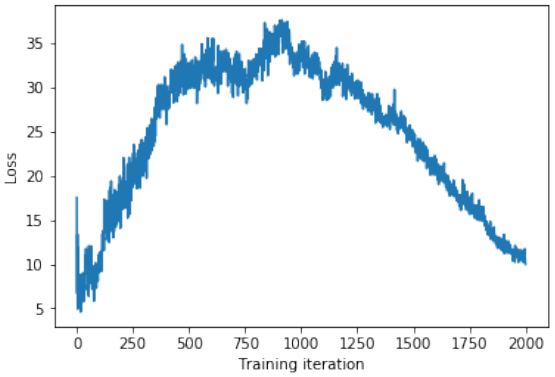
(b) Distribution of number of benzene rings



(c) Distribution of melting points



(d) Average rewards during training



(e) Average loss during training

Figure 15: Multi-objective optimization with logP, number of benzene rings and melting point

3.2.3 Test for parity optimality

The algorithm mentioned in 2 uses multiple start points with different sets of λ to generate a Pareto Frontier.

Algorithm 2 Radial algorithm

Input: $\theta^{(0)}$
$\{\lambda_i\}_{i=1}^{p} \leftarrow$ uniform sampling of R^d
$d_i^{(0)} \leftarrow S(\lambda_i, \theta^{(0)})$
for $i = 1, \ldots, p$ do **do**
    t = 1
    while $\theta_i^{(t-1)}$ not Pareto-optimal **do**
        $\theta_i^{(t)} \leftarrow \theta_i^{(t-1)} + \alpha d_i^{(t-1)}$
        $d_i^{(t)} \leftarrow S(\lambda, \theta_i^{(t)})$
        $t \leftarrow t + 1$
    end while
end for

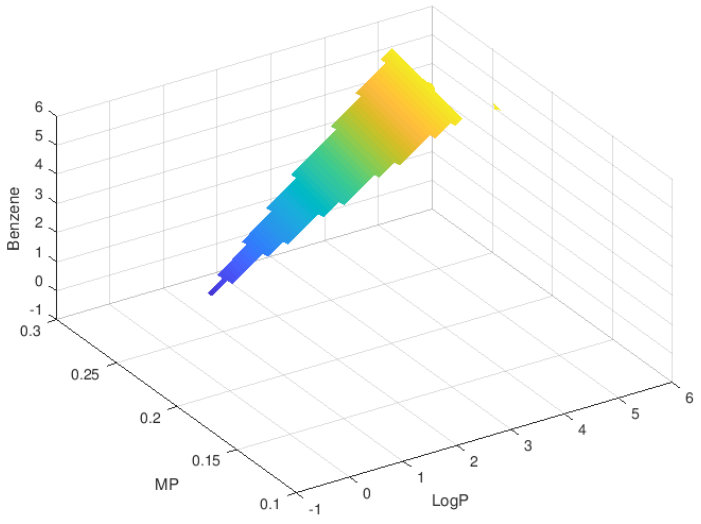


Figure 16: Pareto Frontier for MP-LogP-Benzene Optimization

The plane in figure 16 indicates that the objectives are not conflicting and can all at once reach their optimalities.

4 Conclusion and Further Work

In this project, we have demonstrated that the single objective molecular generation can be extended to multi-objective based setting using a simple scaling of rewards. Future directions and extensions that can be made to this work:

- Improve on Unique Smile Generation
- Graph Generation(instead of smiles) based methods
- Actor-critic techniques
- Value function based multi-objective optimisation methods.