Open access • Posted Content • DOI:10.33774/CHEMRXIV-2021-CG9P8

MoleGuLAR: Molecule Generation using Reinforcement Learning with Alternating Rewards (preprint) — Source link

Manan, Goel, Shampa, Raghunathan, Siddhartha, Laghuvarapu, Priyakumar, U. Deva

Institutions: International Institute of Information Technology, Hyderabad

Published on: 27 Jul 2021 - ChemRxiv

Topics: Reinforcement learning and Virtual screening

MoleGuLaR: Molecule Generation using Reinforcement Learning with Alternating Rewards

Manan Goel, Shampa Raghunathan, Siddhartha Laghuvarapu, and U Deva Priyakumar*

Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad 500 032, India

E-mail: deva@iiit.ac.in

Abstract

Design of new inhibitors for novel targets is a very important problem especially in the current scenario with the world being plagued by COVID-19. Conventional approaches undertaken to this end, like, high-throughput virtual screening require extensive combing through existing datasets in the hope of finding possible matches. In this study we propose a computational strategy for *de novo* generation of molecules with high binding affinities to the specified target. A deep generative model is built using a stack augmented recurrent neural network for initially generating drug like molecules and then it is optimized using reinforcement learning to start generating molecules with desirable properties–primarily the binding affinity. The reinforcement learning section of the pipeline is further extended to multi-objective optimization showcasing the model's ability to generate molecules with a wide variety of properties desirable for drug like molecules, like, LogP, Quantitative Estimate of Drug Likeliness etc.. For multi-objective optimization, we have devised a novel strategy in which the

property being used to calculate the reward is changed periodically. In comparison to the conventional approach of taking a weighted sum of all rewards, this strategy shows enhanced ability to generate a significantly higher number of molecules with desirable properties.

1 Introduction

The advent of data driven techniques across multiple domains of computer science, like, robotics, natural language processing and computer vision has found immense success and this has led to their application in natural sciences.[1,2] The curation of large datasets[3–5] has increased the relevance of machine learning based approaches in problems like molecular property prediction, conceiving retrosynthetic pathways, protein structure prediction and drug disovery.[6–9]

Drug discovery is a long, expensive and arduous process which combines a wide range of disciplines including chemistry, biology and pharmocology. For a novel target, the conventional approach is to perform high-throughput screening on chemical libraries to identify small molecules that bind well to the target. The identified hits are then optimized to get higher binding affinity, reduce toxicity and improve oral bioavailibity.[10,11] The time and expense involved in this process gave rise to alternate *in silico* approaches like virtual screening wherein small molecules from existing drug libraries are computationally evaluated by generating protein ligand complexes and ranking them using a scoring function.[12,13] However, these also come with the caveat that finding the most stable conformation of the complex is a non-convex optimization problem and it can take a very large amount of time ($\approx$ 10 minutes for large molecules) to find the most optimal conformation. Even the most exhaustive studies[14] have been able to find binding affinities of $\approx 10^8$ molecules on a single target which is minuscule in comparison to the vast magnitude of the chemical space with about 10^{60} synthesizable molecules.[15] This posits the argument for the *de novo* generation of molecules with high binding affinities to the required target instead of searching in existing libraries.

Machine learning based approaches like recurrent neural networks, generative adverserial networks (GANs) and variational autoencoders have recently been adopted for molecule generation. Gupta et al. used long short term memory recurrent neural networks, generally used for natural language processing tasks to generate molecules in the form of SMILES (simplified molecular-input line-entry system) which is a string representation of molecules and has it's own grammar and semantics.[16] GANs are generative models that learn the probability distribution of the training data and sampling from the distribution can then be used to generate synthetic data points. This model has also been applied to generation of molecules with desirable properties in works by De Cao and Kipf, Prykhodko et al. and Maziarka et al..[17–19] Jin and coworkers used the graph representation of molecules to train a variational autoencoder that could then generate graphs of new molecules.[20] Kusner et al., Griffiths and Hernández-Lobato and Lim et al. used SMILES representations for generating molecules through the variational autoencoder architecture.[21,23] In fact, applications of deep learning models for molecule generations have proven to be very successful, in the recent years.[24–31]

The next challenge is to generate molecules with desirable properties for which the two major approaches being adopted are reinforcement learning and latent space optimization. Variational autoencoders are capable of learning a continuous space representation of molecules which can then be optimized to get molecules with target properties through techniques like Bayesian optimization and swarm optimization.[32,33] Reinforcement learning can be used to generate desirable molecules by decomposing the process as a sequence of states and actions to maximize a reward which in this case is the desirable property. Popova and coworkers used stack-augmented gated recurrent units (GRUs) to generate molecules followed by reinforcement learning guided optimization on properties like LogP, quantitative estimate of drug likeliness (QED) and synthetic accessibility.[34] Guimaraes et al. combined the GAN and reinforcement learning frameworks for the task while You and coworkers used a graph based policy network to generate molecular graphs.[35,36] Boitreaud et al. combined a

variational autoencoder model with reinforcement learning to generate molecules with high binding affinities to the specified target.[37]

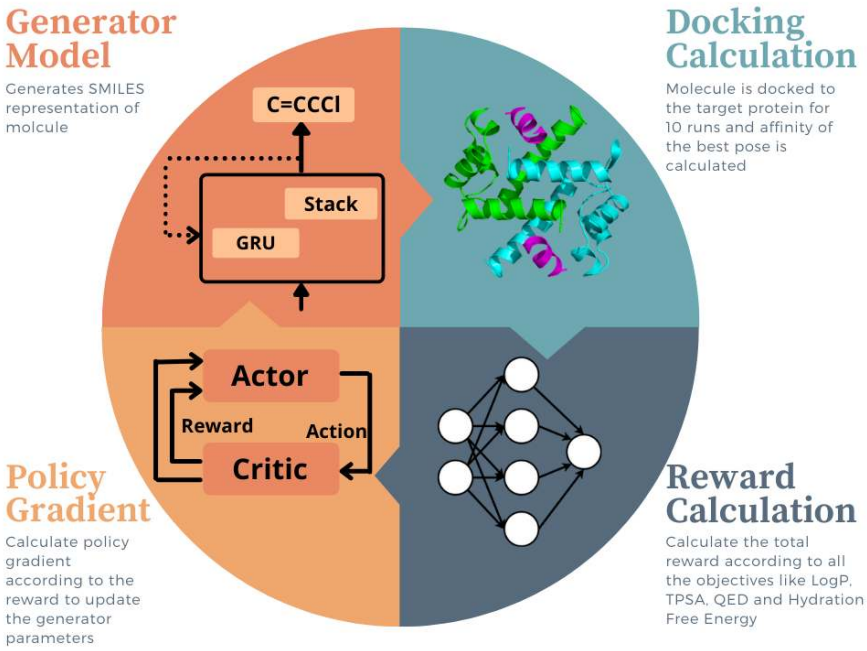


Figure 1: Pipeline used by MoleGuLaR for generating molecules with high binding affinity to a specified drug target.

In this work we propose a molecule generation pipeline (Figure 1) which uses a stack augmented recurrent neural network (RNN) initially trained to generate valid drug-like molecules which is then optimized to generate molecules with a high binding affinity to the specified target. For the binding affinity calculation, we tried two methodologies:

1. Performing docking calculation to find the most stable complex and the corresponding binding affinity.

2. Using a machine learning model trained to predict binding affinities.

We draw a comparison between the two and report their pros and cons. In case of multi-objective optimization we found that using a weighted sum of the rewards from different properties may not be effective in some cases because it is possible that one or more properties dominate others leading to poor results. Hence, we also propose a novel optimization strategy

in which the reward is alternated so that the model changes the region it samples from in the chemical space. When the reward is changed, the generator starts from a better position with respect to one property when optimization for another property is started. We also showcase it's application on two proteins: M_{pro} of SARS-CoV-2 and TTBK1 with a wide set of target properties. The robustness of this strategy is further showcased by using it to optimize the model for conflicting properties along with the binding affinity.

2 Theory and Methods

This section describes the various components of the proposed framework (Figure 1). The formulation of the stack-augmented RNN as the generator model is detailed in Section 2.1 followed by methods for binding affinity calculation and hydration free energy calculation in Sections 2.2 and 2.3, respectively. The formulation of the molecule generation as a Markov Decision Process, use of reinforcement learning to maximize a given reward function using policy gradient and the two optimization strategies used in this study are described in Section 2.4.

2.1 Generator

The generator module makes use of a stack augmented GRU which outputs molecules as SMILES strings.[34,38] A valid SMILES string must have correct valency for all atoms and all ring openings and closures must be counted and hence, conventional RNNs do not work well on this task because of their inability to count. Therefore, the addition of a memory unit along with the RNN forms an appropriate model.

The added stack has 3 operations: PUSH adds a new element at the top of the stack, POP removes the top element and NO-OP does not change the state of the stack. The operation at every step a_t is given by

$$a_t = f(Ah_t)$$

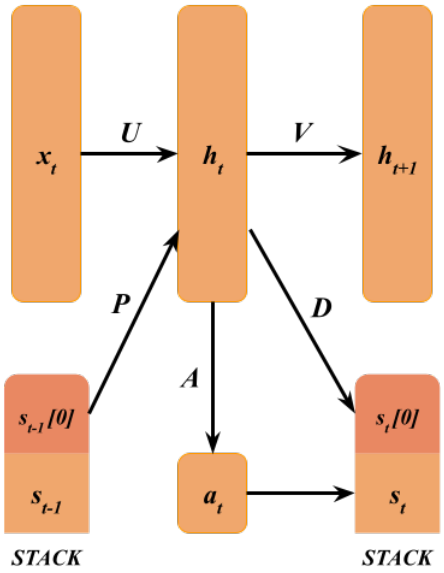


Figure 2: Schematic of push down stack augmented RNN.

where A is a learned parameter and f is the softmax function. Update rules for the stack are:

$$s_t[0] = a_t[PUSH]\sigma(Dh_t) + a_t[POP]s_{t-1}[1] + a_t[NO-OP]s_{t-1}[0]$$
$$s_t[i] = a_t[PUSH]s_{t-1}[i-1] + a_t[POP]s_{t-1}[i+1]$$

If $a_t[PUSH] = 1$, then a new value is pushed to the top of the stack and all existing values are pushed down. While if $a_t[POP] = 1$, the value at the top is removed, and all other values are moved up. The top of the stack is then used to update the hidden state given by

$$h_t = \sigma(Ux_t + Vh_{t-1} + Ps^k_{t-1})$$

where s^k_{t-1} represents the top k elements of the stack and P is a $m \times k$ matrix when m is the hidden state size.

The stack RNN is initially trained on $\approx$ 1.5 million drug-like molecules from the ChEMBL21 database[5] to learn the rules and grammar of SMILES strings.

2.2 Binding Affinity Calculation

The two methodologies being used for binding affinity calculation are detailed in the following sections.

2.2.1 Docking Calculations

The generator model once trained is then used to produce ligands which are then docked to the specified target to find the most stable conformation of the complex and find the corresponding binding affinity further referred as BA in the manuscript. The 3D structure of the molecule from the SMILES string is obtained using the RDKit toolkit.[39] This structure is then converted to a format suitable for the input to the docking software using AutoDockTools4 with the final docking being done using AutoDock-GPU.[40,41] This tool is referred as AutoDock in the rest of the manuscript.

2.2.2 Machine Learning Models

We also experimented with making use of machine learning models to predict the binding affinities of the generated ligands instead of performing a docking calculations to reduce the computation time. In order to do this we obtained a dataset of $\approx$ 2 million molecules docked with the TTBK1 protein.

Figure 3a shows that a significant number of molecules lie in a small range of binding affinities and hence, using that for the predictor model tends to overfit (Figure S1 of Supplementary Material). In order to tackle this issue we split the entire dataset into smaller bins of 1 kcal/mol and sampled 25K molecules from each bin and all the molecules if the number of molecules is less than 25K. Figure 3b shows the distribution of the obtained subset consisting of $\approx$ 200K molecules. This is then split into training, testing and validation sets in the ratio 80:10:10.

We further experimented with various machine learning models for this regression task. Jaeger et al. proposed the Mol2vec[42] model for learning vector representations of SMILES

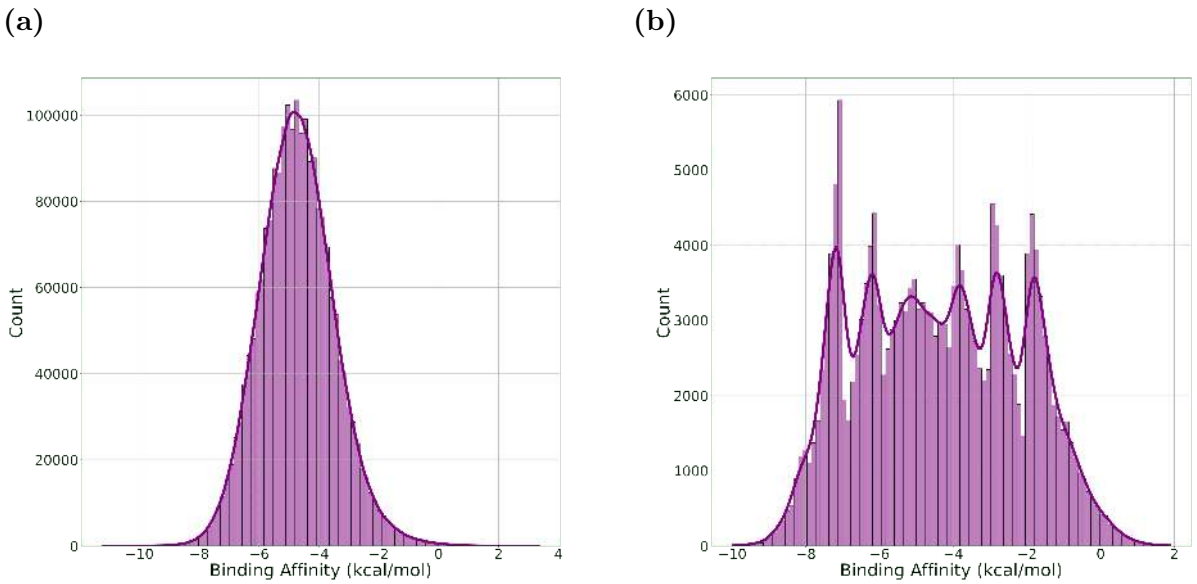


Figure 3: **Distribution of binding affinities.** (a) 2 million molecules with TTBK1, and (b) selected molecules from buckets.

strings that can then be used for further downstream tasks like property prediction. Using these embeddings, a random forest model with 250 decision trees was trained for predicting the BA. The aforementioned model with input features from the embeddings obtained from Graph Isomorphism Networks (GIN) proposed by Xu et al.[43,44] were also used for the task. The drawback of these approaches is that the embeddings being used are constant (Figure S2 of Supplementary Material). Fine tuning these representations during the training process helps improve the accuracy for which three linear layers were added with the GIN embeddings and the model is then trained end-to-end.

2.3 Hydration Free Energy Prediction

The hydration free energy (ΔG_{Hyd}) of a molecule measures it's interaction with water and forms an important part of the drug delivery system. Recently, Pathak et al. proposed a neural network model for predicting hydration free energies in any generic organic solvent.[45] To predict ΔG_{Hyd}, presently, we train a message passing neural network (MPNN) proposed by Gilmer et al. on the FreeSolv dataset which consists of 643 molecules.[46] The MPNN

model has shown to perform extremely well across a wide variety of quantum mechanical prediction tasks as well as ΔG_{Hyd} prediction as shown by Wu et al..[3]

2.4 Reinforcement Learning

A reinforcement learning pipeline generally consists of two modules: the actor and the critic. The actor takes the current state (s_t) of the system and performs an action (a_t) that should maximize the reward. The critic sees a_t, s_t and the state obtained by performing the action (s_{t+1}) and penalizes or rewards the actor.

Generation of a SMILES string can be modelled as a Markov Decision Process where s_t denotes the SMILES string constructed so far, a_t denotes the addition of a token to s_t. We also define a terminal state s_T which signifies the end of the molecule and initial state s_0. The whole generation process can be represented by the trajectory:

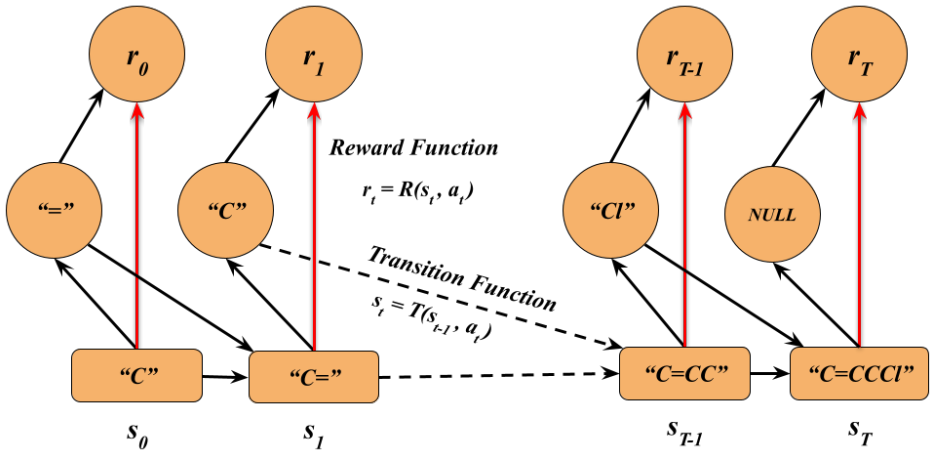


Figure 4: Illustration of SMILES string generation as a Markov Decision Process. At each state s_t, the agent performs an action a_t to give the updated state s_{t+1} and provide a reward according to the state.

The generator model parameterized by Θ estimates the probability $p(a_t|s_t, \Theta)$, samples a_t from the probability distribution and updates the state until s_T is reached. Rewards of all intermediate states s_t with $t < T$ is 0 since it is possible that the intermediate SMILES strings may not represent a valid molecule. s_T is then sent to the critic which returns the

reward $r(s_T)$. Hence, the task here is to find Θ such that the expected reward is maximized.

$$R(\Theta) = \mathbb{E}[r(s_T)|s_0, \Theta] = \sum_{s_T \epsilon S} p_\Theta(s_T) r(s_T)$$

The set S containing all terminal states of length T is exponential in size and hence, needs to be approximated as a mathematical expectation.

$$R(\Theta) = \mathbb{E}[r(s_T)|s_0, \Theta] = \mathbb{E}_{a_1 \sim p_\Theta(a_1|s_0)} .... \mathbb{E}_{a_T \sim p_\Theta(a_T|s_{T-1})} r(s_T) \tag{1}$$

To maximize $R(\Theta)$ we need to calculate its gradient which can be done using the REINFORCE algorithm[47] and the following differentiation rule –

$$\delta_\Theta f(\Theta) = f(\Theta) \frac{\delta_\Theta f(\Theta)}{\delta \Theta} = f(\Theta) \delta_\Theta \log f(\Theta) \tag{2}$$

From Equations (1) and (2)

$$\implies \delta_\Theta R(\Theta) = \sum_{s_T \epsilon S} [\delta_\Theta p_\Theta(s_T)] r(s_T) = \sum_{s_T \epsilon S} p_\Theta(s_T) [\delta_\Theta \log p_\Theta(s_T)] r(s_T)$$

$$= \sum_{s_T \epsilon S} p_\Theta(s_T) [\sum_{t=1}^{T} \delta_\Theta \log p_\Theta(a_t|s_{t-1})] r(s_T)$$

$$\implies \delta_\Theta R(\Theta) = \mathbb{E}_{a_1 \sim p_\Theta(a_1|s_0)} .... \mathbb{E}_{a_T \sim p_\Theta(a_T|s_{T-1})} [\sum_{t=1}^{T} \delta_\Theta \log p_\Theta(a_t|s_{t-1})] r(s_T)$$

For the multiobjective set up, the reward function $r(s_T)$ is composed of multiple components from the different properties that model is being optimized for. The two reward strategies that we propose are –

- **Weighted Sum Rewards**: The total reward $r(s_T)$ is expressed as a weighted sum of

all other components

$$r(s_T) = w_1 D(s_T) + w_2 L(s_T) + w_3 Q(s_T) + w_4 T(s_T) + w_5 H(s_T)$$

where D fetches the reward for BA, L for LogP, Q for QED, T for topological polar surface area and H for ΔG_{Hyd}. Weights are kept as hyperparameters and remain constant throughout the optimization process. The functional forms of the reward for each property is given in Table S1 of the supplementary material.

- **Alternating Rewards**: The aforementioned approach does not work especially in cases where properties are conflicting like high TPSA and more negative hydration free energy would be contradictory in nature. In such cases we have devised a strategy wherein one of the weights is changed to 0 while keeping the other as 1. This takes the generator model into the space where one property is optimal providing a better starting point when optimization for the other property is started. The current strategy works extremely well across most of the tasks and outperforms previous works. Further details are mentioned in the Results and Discussion section.

3 Results and Discussion

This section describes the performance of machine learning models for predicting binding affinity and ΔG_{Hyd} as well as application of the proposed pipeline on the targets:

- SARS-CoV-2 M_{pro} (PDB ID: 6LU7): With the world in the midst of a global pandemic caused by COVID-19, the main protease (M_{pro}) has been identified as an important target due its vital role in viral transcription and replication.[48]

- TTBK1 (PDB ID: 4BTK): Neurodegenerative diseases have become extremely common over the past few years and the tau kinase tau-tubulin kinase 1 has proved to be an attractive target to combat a wide variey of neurodegenerative diseases.[49]

3.1 Machine Learning Models

The performance of machine learning models mentioned in Section 2.2.2 for BA prediction on the test set is reported in Table 1.

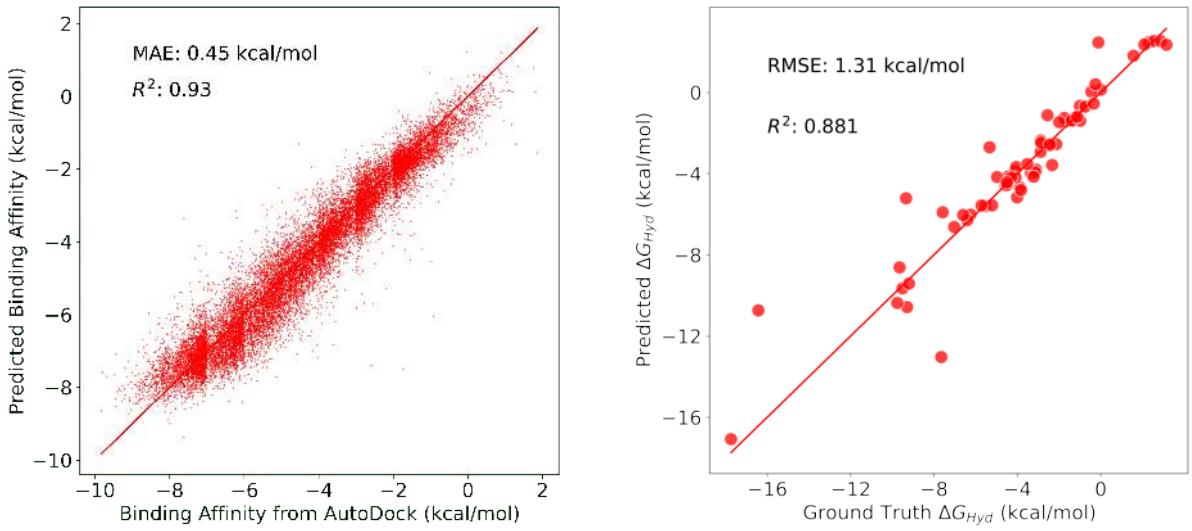
Table 1: Performance of predictor models for BA in terms of performance metrics MAE (kcal/mol) and coefficent of determination (R^2).

Model	MAE (kcal/mol)	R^2
Graph Embeddings + Random Forest	0.87	0.55
Mol2vec + Random Forest	0.47	0.91
Graph Isomorphism Network (GIN)	**0.45**	**0.93**

The use of constant embeddings for the random forest models leads to a higher mean absolute error (MAE) in comparison to the GIN model because in the latter, the model learns to automatically extract more relevant information from the molecular graph. The former also showed comparatively poor performance in the desirable region i.e. where BA is high due to the less number of samples in that range in the dataset. The correlation between the predicted values and ground truth values is shown in Figure 5a. Figure 5b shows the correlation of predicted and ground truth ΔG_{Hyd} in the test set obtained from the FreeSolv dataset. The MPNN model succeeds in achieving a high degree of accuracy with root mean squared error (RMSE) of 1.31 kcal/mol and close correlation characterized by the R^2 score of 0.881 which should be as close to 1 as possible.

3.2 Single Objective Optimization

The initial experiments were performed to analyze the ability of the generator model based only on SMILES to learn to generate molecules whose structure is complementary to the binding pocket without any explicit 3D information. Refer to Section S3 of the supplementary material for the statistics and generated molecules from each of the experiments in Section 3.2.



(a) BA predicted by GIN vs. ground truth values from AutoDock

(b) ΔG_{Hyd} predicted by MPNN vs. ground truth values from FreeSolv

Figure 5: Correlation plots between predicted values from the machine learning models and ground truth values from the respective datasets.

3.2.1 Docking Calculations

For both TTBK1 and SARS-CoV-2 M_{pro}, the generator model was optimized for 175 iterations with 15 policy gradient steps in each and a batch of 10 molecules. At the end of every iteration, 100 molecules are generated and their docking scores are calculated. After the completion of 175 iterations, 500 molecules were generated from the initial model and the optimized model. The distribution of the binding affinities was then plotted in Figures 6a and 6b which show a significant shift towards the more desirable regions.

The above approach shows great performance in optimization for BA but if other properties of the molecule are monitored like LogP, QED etc. (Figures 7a and 7b), they do not stay within the desirable range for drug-like molecules which is between 0 and 5 for LogP and as close to 1 as possible for QED. Hence, there is a need for multi-objective optimization in order to keep the other properties in check as well.

3.2.2 Using GIN

Usage of a GIN for BA prediction leads to an approximately 10 fold speed up in optimization taking only about 5 hours while still keeping the high performance. To further validate the generator, 500 molecules were generated and docked to TTBK1. The shift in distribution of BA, LogP and QED are shown in Figures 6c and 7c. However, the undesirable LogP and QED still persist hence, multi-objective optimization is used further.

3.3 Multi-Objective Optimization

In order to address the shortcomings of single objective optimization, the rewards from different properties were also integrated into the policy gradient calculation and the performance was tested for the two proteins using different target values and both optimization strategies. These are listed in Figure 8 and the results have been discussed in the subsequent sections. Refer to Section S4 of the supplementary material for the statistics and generated molecules from each of the experiments in Section 3.3.

3.3.1 Weighted Sum Reward

Tests were done to optimize the BA calculated using AutoDock along with target LogP = 2.5. While there was improvement in the distribution of LogP in comparison to single objective optimization and BA in comparison to the initial model (Figure 9), however the target was not achieved yet (Figure 10). A similar observation was seen when GIN was used for BA calculation instead of AutoDock in the same setting (Figures 9c and 10c). Further experimentation was done using the GIN BA predictor (Figure 11), in which the generator was optimized to generate molecules with various simultaneous targets i.e. LogP = 2.5, maximum QED, TPSA = 100 Å^2 and $\Delta G_{Hyd} = -10$ kcal/mol and the weighted sum of all rewards was taken. This worked well for BA, TPSA and ΔG_{Hyd} but failed to achieve the target LogP and showed a very low QED. This is in concordance with the OptiMol pipeline by Boitreaud et al. who showed that optimizing for BA led to a reduction in QED.[37] In order

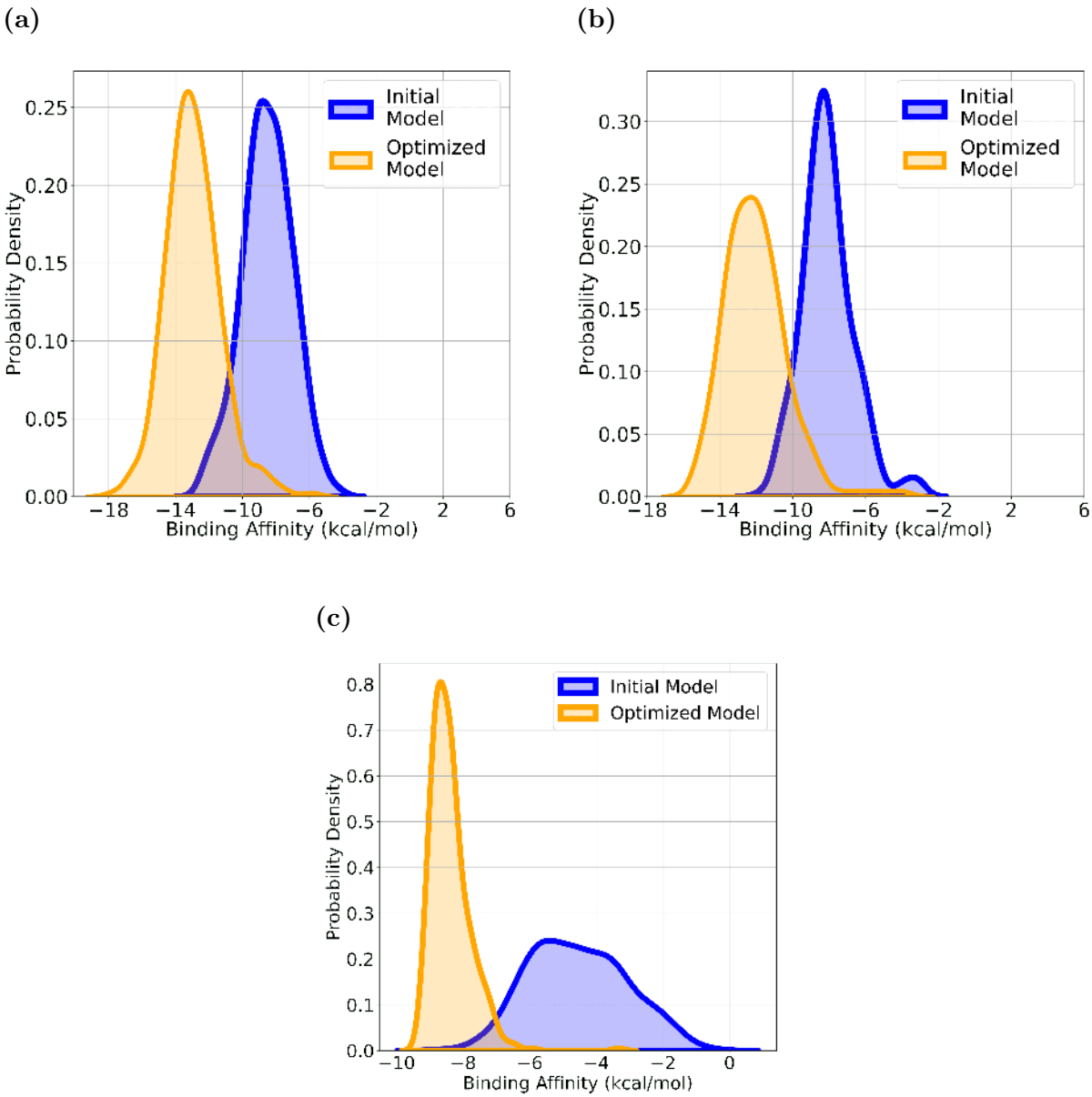


Figure 6: **Distribution of BA of generated molecules.** BA of generated molecules before and after optimizing the generator for reward from BA with (a) SARS-CoV-2 M_{pro}, (b) TTBK1 and (c) TTBK1 calculated using GIN.

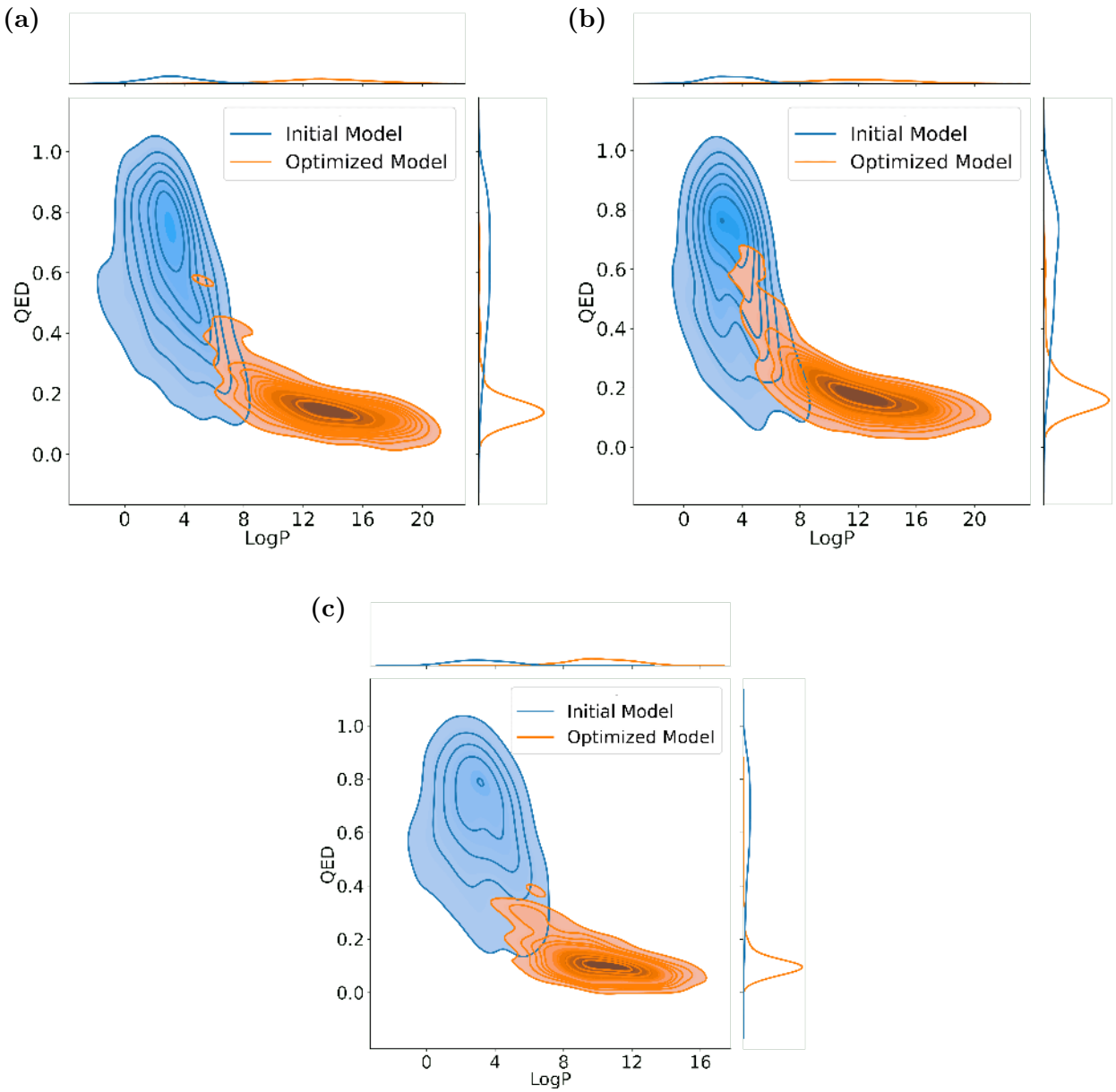


Figure 7: Joint distribution of LogP and QED of generated molecules before and after optimizing the model for reward from BA with (a) SARS-CoV-2 M_{pro}, (b) TTBK1 and (c) TTBK1 calculated using GIN.

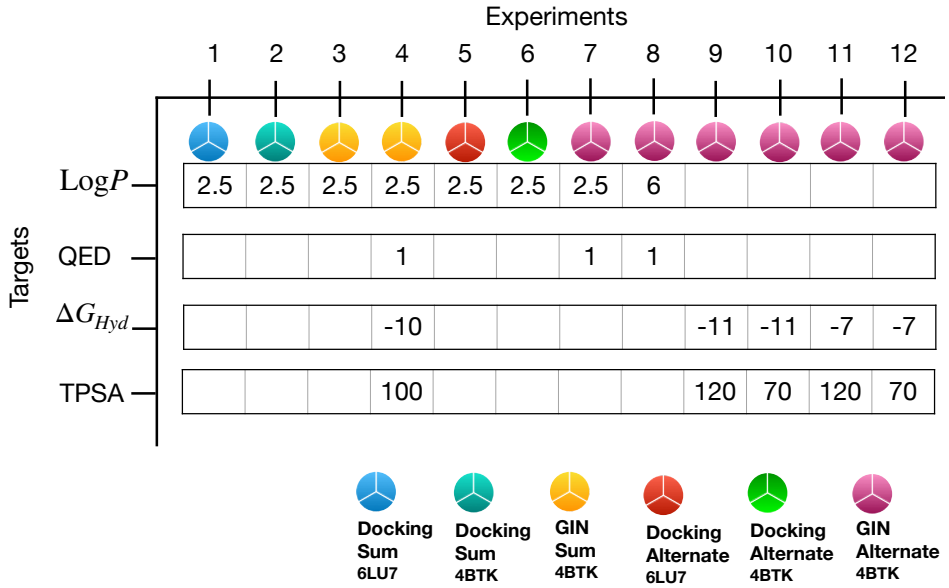


Figure 8: Experiments performed for multi-objective optimization: Protein PDB ID, tools used for BA calculation, different target values of respective properties and optimization strategies. ΔG_{Hyd} and TPSA are in kcal/mol and Å^2, respectively.

to tackle this, we propose the following **alternating rewards strategy** for optimization.

3.3.2 Alternating Rewards

The pipeline's exceptional performance on single objective tasks helped formulate the strategy that only one objective be optimized at a time and the objective be changed at regular intervals. Taking the example of LogP and BA, initially the model moves to generating molecules with better BA but after a few iterations, the reward is switched to optimize for LogP. When the reward is switched, the model is already sampling from the space with high BA and moves towards the region close to the target LogP. Figures 12 and 13 show the application of this strategy on SARS-CoV-2 M_{pro} and TTBK1, respectively, using AutoDock. We can see a better distribution for BA as well as a significant overlap in the most desirable and optimized regions of LogP and QED. Furthermore Table 2 shows the ability of the current strategy to consistently generate a higher percentage of hit molecules in comparison to the weighted sum approach. More detailed information is given in Figure S6 of the supplementary material.

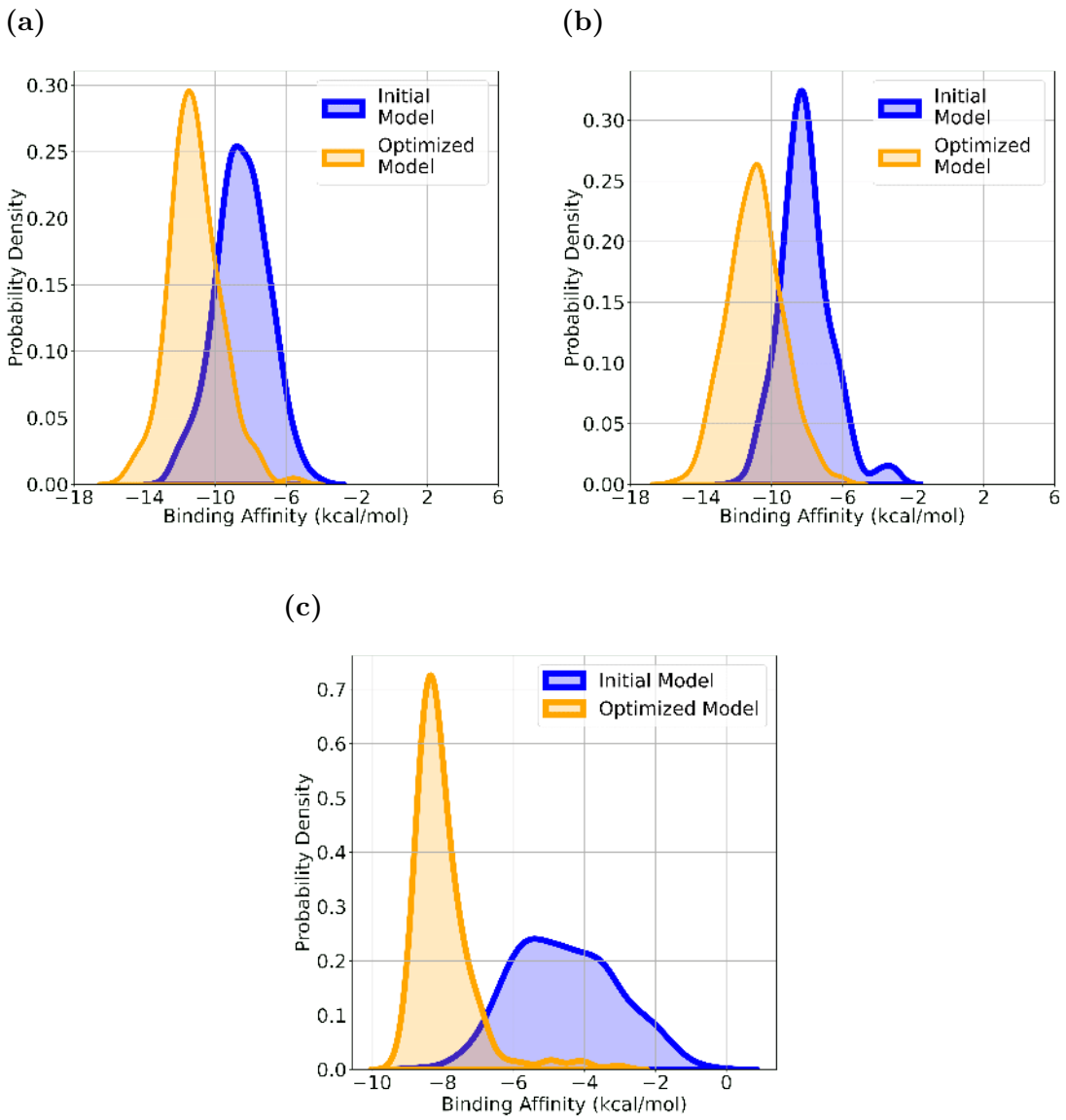


Figure 9: Distribution of BA of generated molecules before and after optimizing the generator for sum of rewards from target LogP = 2.5 and high BA calculated with (a) SARS-CoV-2 M_{pro}, (b) TTBK1 and (c) TTBK1 calculated using GIN.

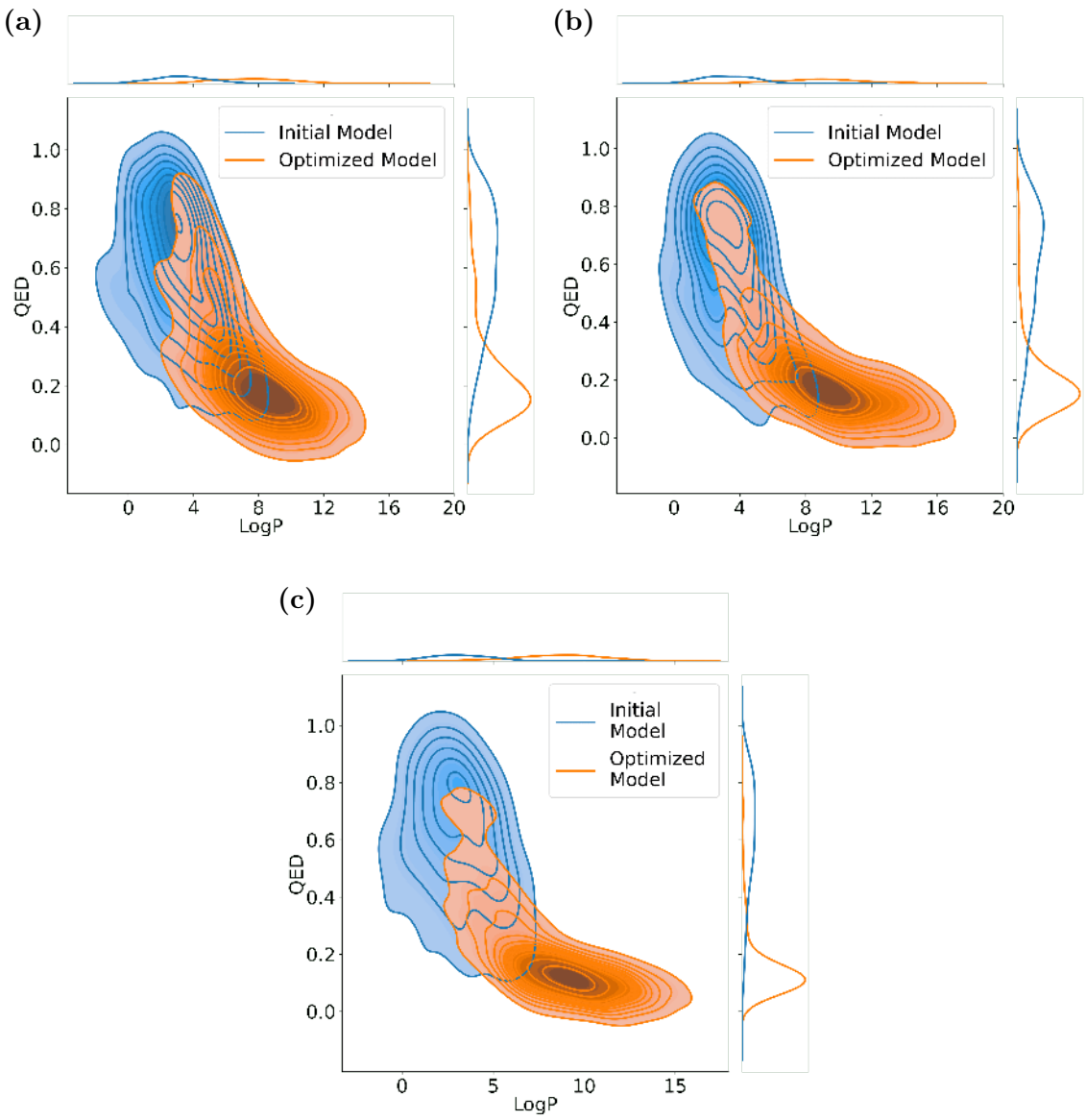


Figure 10: Joint distribution of LogP and QED of generated molecules before and after optimizing the generator for sum of rewards from target LogP = 2.5 and high BA with (a) SARS-CoV-2 M_{pro}, (b) TTBK1 and (c) TTBK1 calculated using GIN.

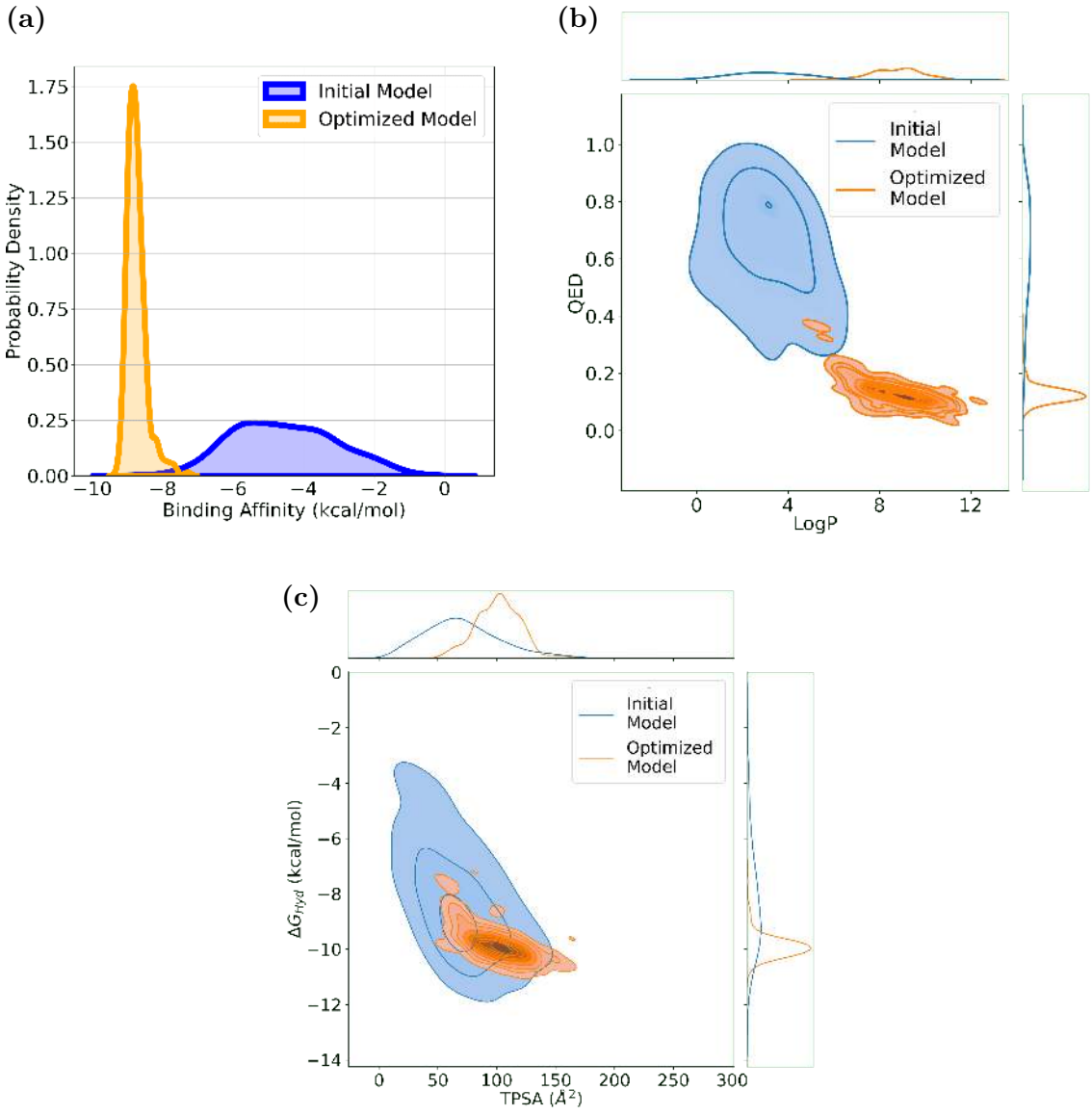


Figure 11: (a) Distribution of BA, (b) joint distribution of QED and LogP, and (c) joint distribution of ΔG_{Hyd} and TPSA before and after optimizing the generator for high BA with TTBK1 calculated using GIN; target values are LogP = 2.5, QED = 1, TPSA = 100 Å^2 and $\Delta G_{Hyd} = -10$ kcal/mol.

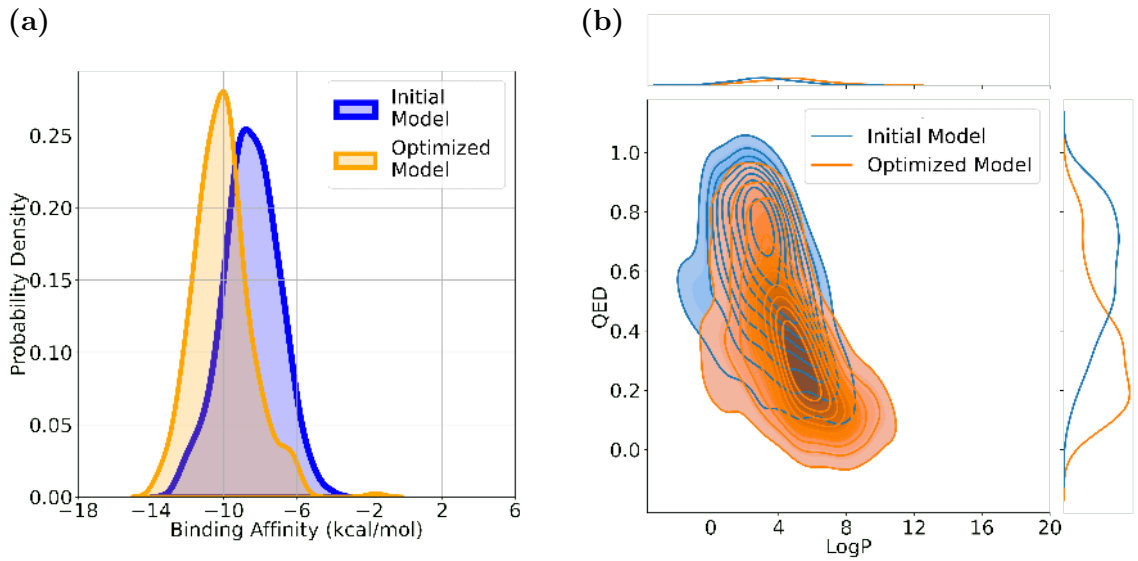


Figure 12: (a) Distribution of BA, and (b) joint distribution of LogP and QED before and after optimizing the model for LogP = 2.5 and high BA with SARS-CoV-2 M_{pro} calculated using AutoDock by alternating rewards.

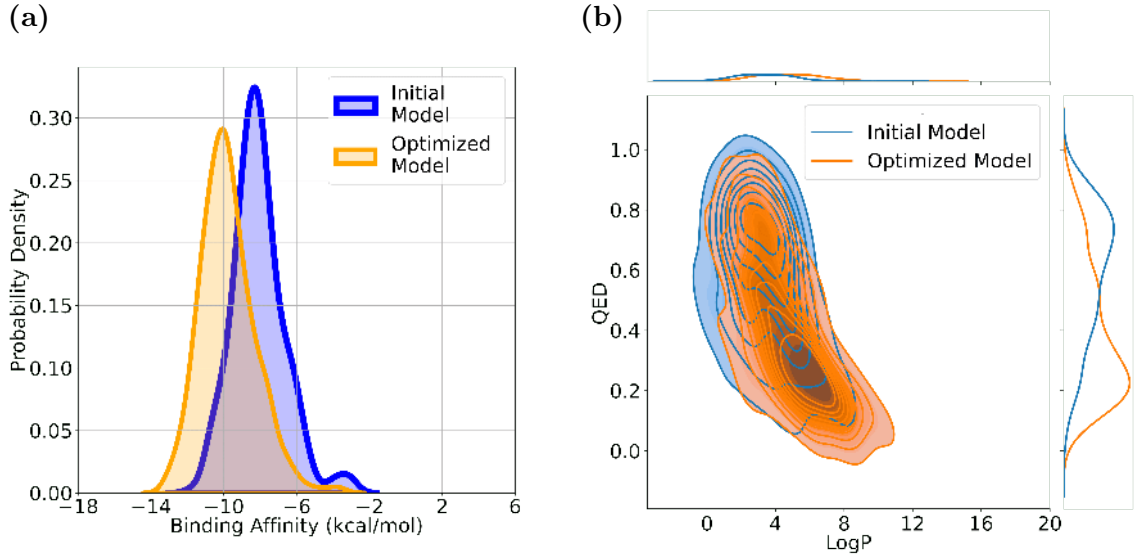


Figure 13: (a) Distribution of BA, and (b) joint distribution of LogP and QED before and after optimizing the model for LogP = 2.5 and high BA with TTBK1 calculated using AutoDock by alternating rewards.

The GIN model was used for further experimentation. Different targets were kept for different properties to evaluate the model's capability of achieving all targets simultaneously. In Figure 14 and Table 3, it is visible that the model is capable of generating molecules with high BA along with maximizing QED subject to the LogP being constrained to 2.5 and 6. Furthermore in Figure 14c, there is a clear separation of the distributions in three

Table 2: Percentage of molecules with desirable properties for SARS-CoV-2 M_{pro} and TTBK1 using the two optimization strategies.

PDB ID	Optimization strategy	Percentage of molecules with BA < -9 kcal/mol and $1 < LogP < 5$	Percentage of molecules with BA < -10 kcal/mol and $1 < LogP < 5$	Percentage of molecules with BA < -11 kcal/mol and $1 < LogP < 5$
6LU7	Sum	11.4 %	6.5 %	2.6 %
4BTK	Sum	6.3 %	1.1 %	0.4 %
6LU7	Alternate	39.1 %	18.9 %	7.1 %
4BTK	Alternate	30.3 %	14.2 %	3.0 %

dimensions showing the model's ability of navigating different regions of the chemical space where molecules possess the desired properties.

Table 3: LogP and QED targets along with obtained mean values of BA, LogP and QED of the corresponding generated data as well as the best BA.

Target LogP	Target QED	Mean BA (kcal/mol)	Best BA (kcal/mol)	Mean LogP	Mean QED
2.5	1	-6.76	-8.18	2.9	0.42
6	1	-7.64	-8.41	5.87	0.19

A similar experiment was repeated with TPSA and ΔG_{Hyd} along with BA to see how the optimization strategy handles conflicting targets since higher the TPSA, more negative the ΔG_{Hyd}. The target pairs are mentioned in Table 4.

Table 4: TPSA and ΔG_{Hyd} targets along with mean values of BA, TPSA and ΔG_{Hyd} of the corresponding generated data as well as the best BA.

Target TPSA (Å^2)	Target ΔG_{Hyd} (kcal/mol)	Mean TPSA (Å^2)	Mean ΔG_{Hyd} (kcal/mol)	Mean BA (kcal/mol)	Best BA (kcal/mol)
70	-11	88.77	-10.13	-6.11	-8.36
120	-11	117.25	-10.75	-6.65	-8.32
70	-7	71.64	-7.42	-7.4	-8.90
120	-7	99.16	-8.42	-6.85	-8.64

TPSA of 120 Å^2 and 70 Å^2 with ΔG_{Hyd} of -11 kcal/mol and -7 kcal/mol respectively are feasible simultaneously. The distinction in the distributions is shown in Figure 15 where

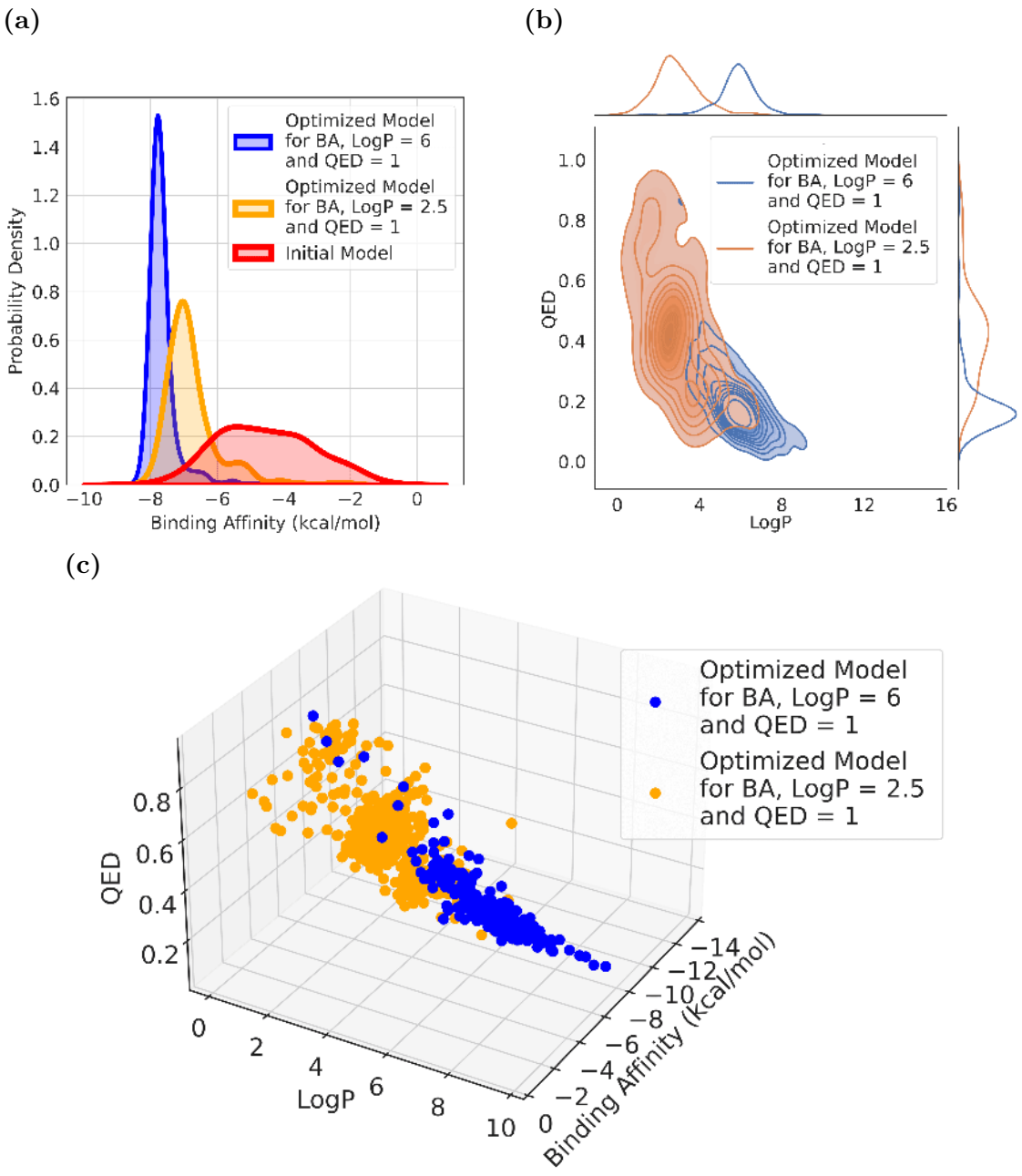


Figure 14: (a) Distribution of BA, (b) joint distribution of LogP and QED, and (c) 3D representation of properties of generated molecules before and after optimizing the generator for high BA with TTBK1 calculated using GIN.

the pipeline successfully generates molecules with high BA as well as the target TPSA and ΔG_{Hyd}. The generator learns to add Oxygen, Nitrogen and Sulphur atoms to the molecular structure to increase TPSA and improve ΔG_{Hyd} in the first case while in the second it adds long aliphatic chains and benzene rings to keep the TPSA and ΔG_{Hyd} low. However, both the tasks are achieved while keeping a high BA to the target protein.

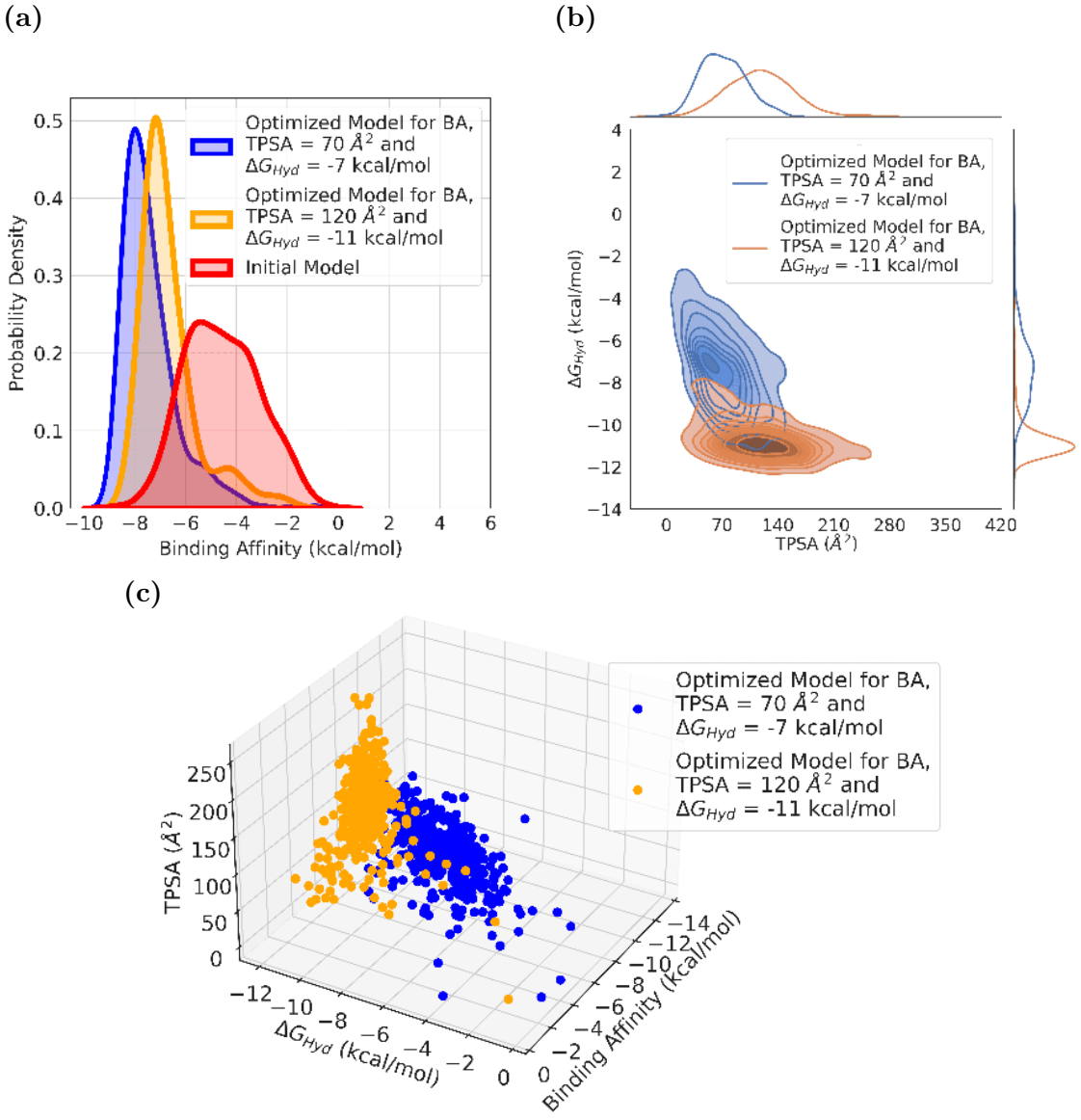


Figure 15: (a) Distribution of BA, (b) joint distribution of LogP and QED, and (c) 3D representation of properties of generated molecules before and after optimizing the generator for high BA with TTBK1 calculated using GIN.

The difficulty arises in the other two cases since it is extremely hard to achieve the

mentioned TPSA and ΔG_{Hyd} simultaneously. However, switching the rewards after every 35 iterations does help reach a common consensus between the properties. In Figures 16b and 16c, though the peaks of the distributions do not lie exactly at the target but they show a wider distribution which leads to a higher fraction of molecules being closer to the target properties. The molecules with ΔG_{Hyd}= -7 kcal/mol and TPSA = 120Å^2 are much larger than those generated for other cases with both a large number of Oxygen, Nitrogen and Sulphur atoms to increase TPSA as well as long aliphatic chains and benzene rings to make the ΔG_{Hyd} less negative which shows that the generator automatically learns to create large molecules so that both properties can be satisfied simultaneously. The molecules with ΔG_{Hyd} = -11 kcal/mol and TPSA = 70Å^2 are smaller but contain at least 3 benzene rings causing a low TPSA but a large number of Oxygen, Nitrogen and Sulphur atoms to take the ΔG_{Hyd} close to -11 kcal/mol. This further showcases the robustness of the the alternating reward strategy to generate molecules with simultaneous targets of different nature and find an intersection between the sets of molecules.

The best hit from 500 molecules generated from each of the aforementioned experiments using alternating rewards are shown in Figure 17.

4 Conclusion

In this study, we propose a pipeline for *de novo* generation of drug like molecules with high BA to novel targets along with other desirable properties. Reinforcement learning is used to optimize the generator model weights to maximize the rewards obtained from calculated properties. A novel optimization strategy is also proposed for the multi-objective set up in which the reward function is switched to optimize for a different property at regular intervals instead of the conventional approach in which sum of rewards from all properties is taken. We also show the performance of two ways of calculating BA i.e. using AutoDock and using a predictor model while also weighing the merits and demerits of both approaches as a part

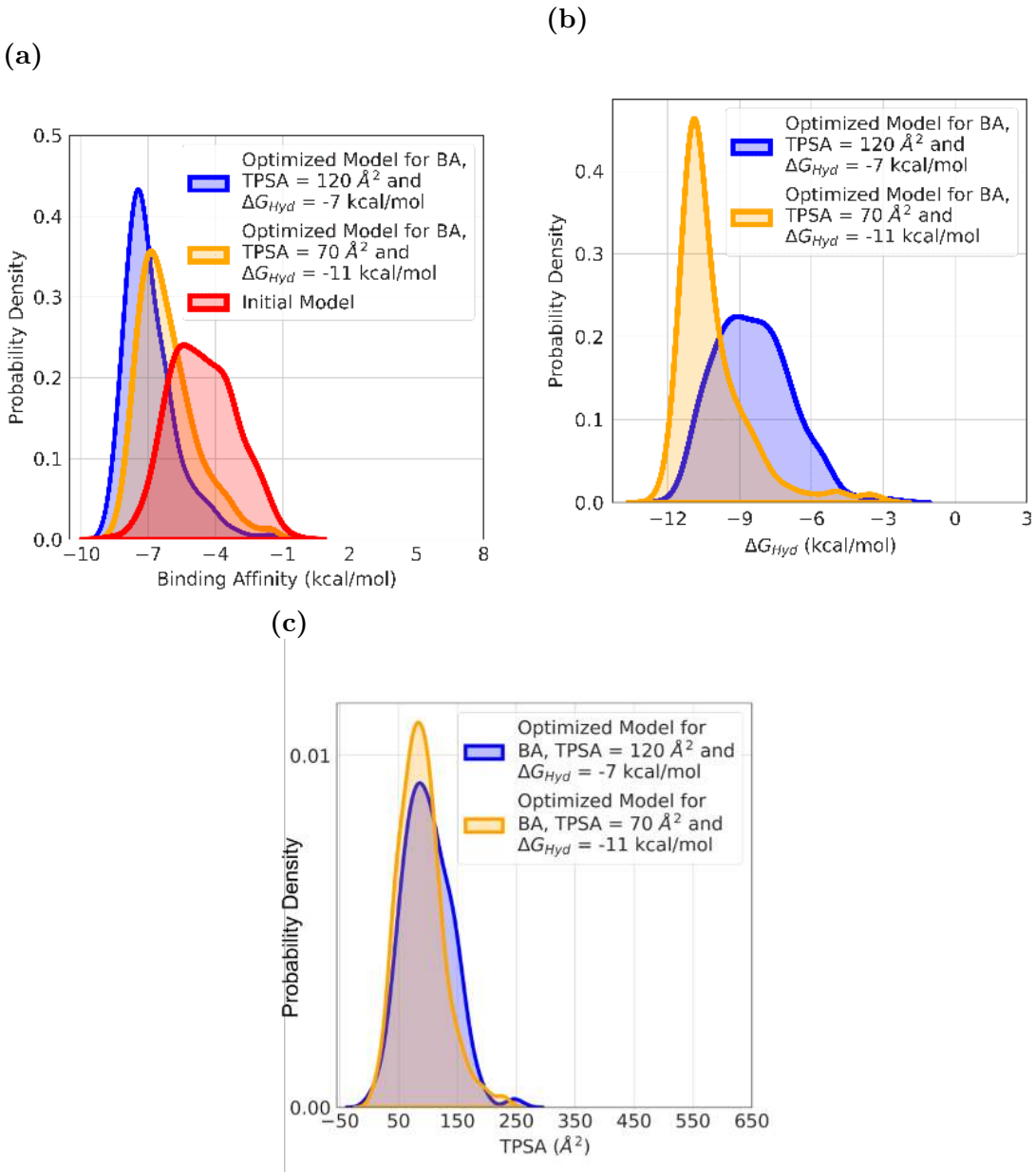
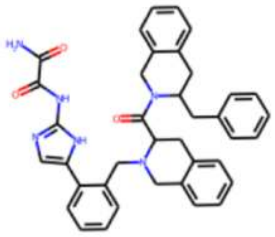
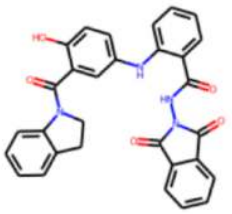


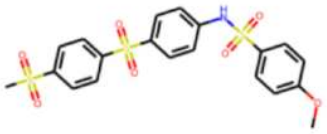
Figure 16: Distribution of (a) BA, (b) ΔG_{Hyd}, and (c) TPSA before and after optimizing the model for high binding affinity with TTBK1 calculated using GIN.



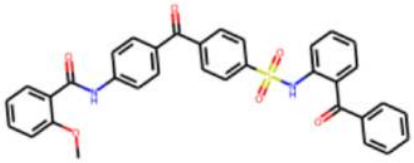
BA with SARS-CoV-2 M_{pro}: -13.53 kcal/mol
LogP: 4.62



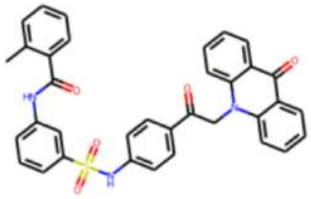
BA with TTBK1: -12.14 kcal/mol
LogP: 4.28



BA with TTBK1 calculated using GIN: -7.12 kcal/mol
LogP: 2.73
QED: 0.55



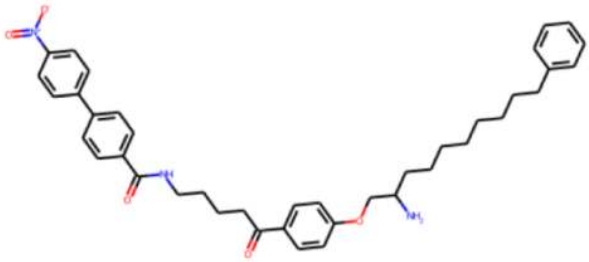
BA with TTBK1 calculated using GIN: -8.17 kcal/mol
LogP: 6.21
QED: 0.19



BA with TTBK1 calculated using GIN: -8.32 kcal/mol
TPSA = 114.34 $Å^2$
ΔG_{Hyd} = -11.03 kcal/mol

BA with TTBK1 calculated using GIN: -7.98 kcal/mol
TPSA = 79.81 $Å^2$
ΔG_{Hyd} = -10.45 kcal/mol

BA with TTBK1 calculated using GIN: -8.38 kcal/mol
TPSA = 71.52 $Å^2$
ΔG_{Hyd} = -7.13 kcal/mol



BA with TTBK1 calculated using GIN: -7.5 kcal/mol
TPSA = 124.56 $Å^2$
ΔG_{Hyd} = -7.0 kcal/mol

Figure 17: The top hit from 500 molecules generated after each experiment done using alternating rewards.

of the pipeline. Further work can include training the BA predictor models on the fly using techniques like active learning to make the pipeline more robust and efficient. The use of this architecture significantly reduces the number of docking calculations required to identify potential drugs for a novel target removing a major bottleneck in the drug discovery process and can potentially be used to generate targeted drug libraries. We show that the alternating reward strategy is extremely robust in finding potential hits for the target across a wide set of target properties.

References

(1) Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. Machine learning for molecular and materials science. *Nature* **2018**, *559*, 547–555.

(2) Sadowski, P.; Baldi, P. *Braverman Readings in Machine Learning. Key Ideas from Inception to Current State*; Springer, 2018; pp 269–297.

(3) Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; Pande, V. MoleculeNet: A benchmark for molecular machine learning. *Chem. Sci.* **2018**, *9*, 513–530.

(4) Irwin, J. J.; Shoichet, B. K. ZINC–A free database of commercially available compounds for virtual screening. *J. Chem. Inf. Model.* **2005**, *45*, 177–182.

(5) Mendez, D.; Gaulton, A.; Bento, A. P.; Chambers, J.; De Veij, M.; Félix, E.; Magariños, M. P.; Mosquera, J. F.; Mutowo, P.; Nowotka, M., et al. ChEMBL: Towards direct deposition of bioassay data. *Nucleic Acids Res.* **2019**, *47*, D930–D940.

(6) Senior, A. W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Žídek, A.; Nelson, A. W.; Bridgland, A., et al. Improved protein structure prediction using potentials from deep learning. *Nature* **2020**, *577*, 706–710.

(7) Segler, M. H.; Waller, M. P. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chem.–Eur. J.* **2017**, *23*, 5966–5971.

(8) Schreck, J. S.; Coley, C. W.; Bishop, K. J. Learning retrosynthetic planning through simulated experience. *ACS Cent. Sci.* **2019**, *5*, 970–981.

(9) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural message passing for quantum chemistry. International Conference on Machine Learning. 2017; pp 1263–1272.

(10) Szymański, P.; Markowicz, M.; Mikiciuk-Olasik, E. Adaptation of high-throughput screening in drug discovery—toxicological screening tests. *Int. J. Mol. Sci.* **2012**, *13*, 427–452.

(11) Broach, J. R.; Thorner, J. High-throughput screening for drug discovery. *Nature* **1996**, *384*, 14–16.

(12) Maia, E. H. B.; Assis, L. C.; de Oliveira, T. A.; da Silva, A. M.; Taranto, A. G. Structure-based virtual screening: From classical to artificial intelligence. *Front. Chem.* **2020**, *8*.

(13) Sliwoski, G.; Kothiwale, S.; Meiler, J.; Lowe, E. W. Computational methods in drug discovery. *Pharmacol. Rev.* **2014**, *66*, 334–395.

(14) Lyu, J.; Wang, S.; Balius, T. E.; Singh, I.; Levit, A.; Moroz, Y. S.; O'Meara, M. J.; Che, T.; Algaa, E.; Tolmachova, K., et al. Ultra-large library docking for discovering new chemotypes. *Nature* **2019**, *566*, 224–229.

(15) Reymond, J.-L. The chemical space project. *Acc. Chem. Res.* **2015**, *48*, 722–730.

(16) Gupta, A.; Müller, A. T.; Huisman, B. J.; Fuchs, J. A.; Schneider, P.; Schneider, G. Generative recurrent networks for de novo drug design. *Mol. Inf.* **2018**, *37*, 1700111.

(17) De Cao, N.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973* **2018**,

(18) Prykhodko, O.; Johansson, S. V.; Kotsias, P.-C.; Arús-Pous, J.; Bjerrum, E. J.; Engkvist, O.; Chen, H. A de novo molecular generation method using latent vector based generative adversarial network. *J. Cheminf.* **2019**, *11*, 1–13.

(19) Maziarka, Ł.; Pocha, A.; Kaczmarczyk, J.; Rataj, K.; Danel, T.; Warchoł, M. Mol-CycleGAN: a generative model for molecular optimization. *J. Cheminf.* **2020**, *12*, 1–18.

(20) Jin, W.; Barzilay, R.; Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. International Conference on Machine Learning. 2018; pp 2323–2332.

(21) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar variational autoencoder. International Conference on Machine Learning. 2017; pp 1945–1954.

(22) Griffiths, R.-R.; Hernández-Lobato, J. M. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chem. Sci.* **2020**, *11*, 577–586.

(23) Lim, J.; Ryu, S.; Kim, J. W.; Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *J. Cheminf.* **2018**, *10*, 1–9.

(24) Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; Aspuru-Guzik, A. Self-Referencing Embedded Strings (SELFIES): A 100% robust molecular string representation. *Mach. Learn.: Sci. Technol.* **2020**, *1*, 045024.

(25) Born, J.; Manica, M.; Cadow, J.; Markert, G.; Mill, N. A.; Filipavicius, M.; Martínez, M. R. PaccMann^{RL} on SARS-CoV-2: Designing antiviral candidates with conditional generative models. *arXiv preprint arXiv:2005.13285* **2020**,

(26) Born, J.; Manica, M.; Cadow, J.; Markert, G.; Mill, N. A.; Filipavicius, M.; Janakarajan, N.; Cardinale, A.; Laino, T.; Martínez, M. R. Data-driven molecular design for

discovery and synthesis of novel ligands: a case study on SARS-CoV-2. *Mach. Learn.: Sci. Technol.* **2021**, *2*, 025024.

(27) Khemchandani, Y.; O'Hagan, S.; Samanta, S.; Swainston, N.; Roberts, T. J.; Bollegala, D.; Kell, D. B. DeepGraphMolGen, a multi-objective, computational strategy for generating molecules with desirable properties: a graph convolution and reinforcement learning approach. *J. Cheminf.* **2020**, *12*, 1–17.

(28) Kell, D. B.; Samanta, S.; Swainston, N. Deep learning and generative methods in cheminformatics and chemical biology: navigating small molecule space intelligently. *Biochem. J.* **2020**, *477*, 4559–4580.

(29) Jiménez-Luna, J.; Grisoni, F.; Weskamp, N.; Schneider, G. Artificial intelligence in drug discovery: Recent advances and future perspectives. *Expert Opin. Drug Discovery* **2021**, 1–11.

(30) Mehta, S.; Laghuvarapu, S.; Pathak, Y.; Sethi, A.; Alvala, M.; Priyakumar, U. D. MEMES: Machine learning framework for Enhanced MolEcular Screening. *chemrxiv preprint 10.33774/chemrxiv-2021-nr0vn-v2* **2021**,

(31) Bagal, V.; Aggarwal, R.; Vinod, P.; Priyakumar, U. D. LigGPT: Molecular Generation using a Transformer-Decoder Model. *chemrxiv preprint 10.26434/chemrxiv.14561901.v1* **2021**,

(32) Winter, R.; Montanari, F.; Steffen, A.; Briem, H.; Noé, F.; Clevert, D.-A. Efficient multi-objective molecular optimization in a continuous latent space. *Chem. Sci.* **2019**, *10*, 8016–8024.

(33) Korovina, K.; Xu, S.; Kandasamy, K.; Neiswanger, W.; Poczos, B.; Schneider, J.; Xing, E. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. International Conference on Artificial Intelligence and Statistics. 2020; pp 3393–3403.

(34) Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci. Adv.* **2018**, *4*, eaap7885.

(35) Guimaraes, G. L.; Sanchez-Lengeling, B.; Outeiral, C.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *arXiv preprint arXiv:1705.10843* **2017**,

(36) You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473* **2018**,

(37) Boitreaud, J.; Mallet, V.; Oliver, C.; Waldispuhl, J. Optimol: Optimization of binding affinities in chemical space for drug discovery. *J. Chem. Inf. Model.* **2020**, *60*, 5658–5666.

(38) Joulin, A.; Mikolov, T. Inferring algorithmic patterns with stack-augmented recurrent nets. *arXiv preprint arXiv:1503.01007* **2015**,

(39) Landrum, G. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling. 2013.

(40) Morris, G. M.; Huey, R.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J. Comput. Chem.* **2009**, *30*, 2785–2791.

(41) Santos-Martins, D.; Solis-Vasquez, L.; Tillack, A. F.; Sanner, M. F.; Koch, A.; Forli, S. Accelerating AutoDock4 with GPUs and gradient-based local search. *J. Chem. Theory Comput.* **2021**, *17*, 1060–1073.

(42) Jaeger, S.; Fulle, S.; Turk, S. Mol2vec: unsupervised machine learning approach with chemical intuition. *J. Chem. Inf. Model.* **2018**, *58*, 27–35.

(43) Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* **2018**,

(44) Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; Leskovec, J. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* **2019**,

(45) Pathak, Y.; Mehta, S.; Priyakumar, U. D. Learning Atomic Interactions through Solvation Free Energy Prediction Using Graph Neural Networks. *J. Chem. Inf. Model.* **2021**, *61*, 689–698.

(46) Mobley, D. L.; Guthrie, J. P. FreeSolv: a database of experimental and calculated hydration free energies, with input files. *J. Comput.-Aided Mol. Des.* **2014**, *28*, 711–720.

(47) Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **1992**, *8*, 229–256.

(48) Zhang, L.; Lin, D.; Sun, X.; Curth, U.; Drosten, C.; Sauerhering, L.; Becker, S.; Rox, K.; Hilgenfeld, R. Crystal structure of SARS-CoV-2 main protease provides a basis for design of improved α-ketoamide inhibitors. *Science* **2020**, *368*, 409–412.

(49) Sato, S.; Cerny, R. L.; Buescher, J. L.; Ikezu, T. Tau-tubulin kinase 1 (TTBK1), a neuron-specific tau kinase candidate, is involved in tau phosphorylation and aggregation. *J. Neurochem.* **2006**, *98*, 1573–1584.

Graphical TOC Entry