Generative Model for Synthesizing Ionizable Lipids: A Monte Carlo Tree Search Approach

Jingyi Zhao
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ
jz610@cam.ac.uk

Yuxuan Ou
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ
yuxuan.ou@trinity.ox.ac.uk

Austin Tripp
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ
ajt212@cam.ac.uk

Morteza Rasoulianboroujeni
School of Pharmacy
University of Wisconsin-Madison
Madison, WI 53705
rasoulianbor@wisc.edu

José Miguel Hernández-Lobato
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ
jmh233@cam.ac.uk

Abstract

Ionizable lipids are essential in developing lipid nanoparticles (LNPs) for effective messenger RNA (mRNA) delivery. While traditional methods for designing new ionizable lipids are typically time-consuming, deep generative models have emerged as a powerful solution, significantly accelerating the molecular discovery process. However, a practical challenge arises as the molecular structures generated can often be difficult or infeasible to synthesize. This project explores Monte Carlo tree search (MCTS)-based generative models for synthesizable ionizable lipids. Leveraging a synthetically accessible lipid building block dataset and two specialized predictors to guide the search through chemical space, we introduce a policy network guided MCTS generative model capable of producing new ionizable lipids with available synthesis pathways.

1 Introduction

The development of messenger RNA (mRNA)-based therapeutics marks a transformative advance in the treatment and prevention of a wide range of diseases, including genetic disorders, infectious diseases, and cancer [37, 23]. Given the intrinsic instability of mRNA, a robust delivery mechanism is crucial, with ionizable lipid nanoparticles (LNPs) emerging as the leading technology for this purpose [32, 24]. These LNPs comprise four distinct lipid types, among which the ionizable amine-containing lipids are pivotal. They primarily facilitate the encapsulation of mRNA within LNPs and enhance its delivery into the cellular cytoplasm, where it can be translated into therapeutic proteins [19, 8, 51, 52]. Notably, different LNPs often feature unique ionizable lipid structures, emphasizing the importance of developing a diverse array of ionizable lipids. This diversity is crucial for effectively delivering mRNA to various target cells and tissues, highlighting the need for continued innovation in the design of ionizable lipids to facilitate mRNA-based therapeutics.

Designing novel ionizable lipids is traditionally time-consuming and labor-intensive. However, recent advancements in the integration of deep learning with combinatorial chemistry have shown great promise in accelerating this development [53]. One of the primary challenges in ionizable lipid generation is the effective exploration of a vast combinatorial chemical space. The SyntheMol approach has demonstrated considerable success in this area, utilizing Monte Carlo tree search

AI for New Drug Modalities at NeurIPS 2024.

(MCTS) to efficiently navigate through expansive search spaces and generate desired molecules, complete with synthesis pathways, particularly in the field of antibiotic development [48].

This work extends the application of the MCTS-based generative model to the domain of ionizable lipid generation. We introduce a policy network guided MCTS method which leverages the strengths of MCTS and integrates it with the strategic direction provided by the policy network, aiming to optimize the generation process and yield new ionizable lipids more efficiently. Our main contributions are:

- We compile a synthetically accessible dataset of molecules suitable for use as ionizable lipid heads or lipid tails, which can facilitate future ionizable lipid generation tasks.
- We develop reliable lipid property predictors to assess whether a candidate molecule possesses lipid-like characteristics or ionizable properties.
- We propose and implement a generative model that leverages a policy network to guide the MCTS in exploring the chemical space. This model effectively generates high-quality products, complete with available synthesis pathways.

2 Related Work

Deep generative models have emerged as a powerful solution to the inverse molecular design challenge, enabling the translation of desired molecular properties into specific molecular structures [25, 14, 44, 3, 4, 41, 29]. A practical problem that obstructs the usefulness of these generative algorithms is that proposed molecular structures may be challenging or infeasible to synthesize [12]. While post-hoc synthesis planning for generated molecules is feasible [12, 39, 40], a more effective approach is to incorporate synthesis instructions directly into the design phase. One effective solution is to adopt a bottom-up approach which begins with existing building blocks and strategically determines pathways to synthesize product molecules possessing desired properties [5, 48].

Identifying new ionizable lipids remains a bottleneck of LNP development. Current state-of-the-art approaches still primarily rely on combinatorial chemistry techniques. Even when synthesized, an ionizable lipid often fails to exhibit transfection capabilities, with a low likelihood of achieving high transfection efficiency [53, 27]. Despite the vast structural design space of ionizable lipids, small modifications in chemical structure can lead to substantial differences in biological performance [27]. Existing approaches that incorporate machine learning into ionizable lipid development primarily focus on building transfection efficiency predictors to aid in lipid screening [53, 27, 9, 33]. However, no current approaches directly utilize machine learning for the generative design of ionizable lipids.

3 Background

This section presents the theoretical background of this work.

3.1 Monte Carlo Tree Search in Molecular Generation

MCTS is a robust algorithm that integrates the stochastic nature of Monte Carlo simulations with the structured decision-making processes inherent in tree searches [22, 7]. A recent study has shown that MCTS-based generative models can perform successfully in small-molecule antibiotic development [48]. In the context of SyntheMol, the algorithm iteratively builds a tree structure, where each node represents one or more potential molecular structures, and branches represent possible synthesis steps using 13 well-validated chemical reactions [48]. The application of SyntheMol to lipid generation is detailed in Appendix B.

3.2 Guided Monte Carlo Tree Search

The pioneering work that integrates neural networks with MCTS is AlphaGo [42]. Building upon that, AlphaZero refined this approach by merging the policy and value networks into a single, more efficient neural network [43]. What's more, AlphaZero was trained exclusively through self-play, enabling the system to adapt and optimize its game play without relying on external data. AlphaZero provides a highly relevant framework for the application to molecular generation. Although our

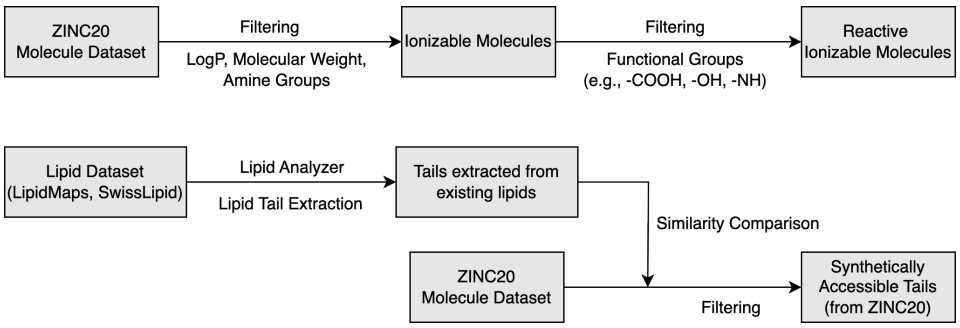


Figure 1: Roadmap of lipid building block datasets construction. The upper flowchart illustrates the process of filtering ionizable lipid head building block dataset. The lower flowchart illustrates the process of constructing lipid tail building block dataset.

approach diverges from AlphaZero in that it does not integrate a value network within the MCTS framework—instead relying on separate property predictors—the role of the policy network remains pivotal in both contexts. In molecular generation, our policy network is instrumental in selecting the next building block for synthesis, which directly influences the structure and properties of the resultant molecule, akin to how each move in Go influences the progression and outcome of the game.

3.3 Retrosynthesis Evaluation

While the ideal method to validate the synthesis pathway of our generated products would be through experimental testing in a laboratory, practical constraints currently prevent us from doing so. Consequently, we rely on in silico evaluations to estimate synthesizability. We first calculate the synthetic accessibility score (SA score) to predict synthesizability in an automated manner [10]. Additionally, we seek to validate our proposed synthesis pathways using Syntheseus [31], a python package for retrosynthetic planning.

The SA score is calculated through a combination of fragment contributions and a complexity penalty [10]. The resulting SA score provides a metric for synthetic accessibility, ranging from 1 (indicating easy synthesis) to 10 (indicating high difficulty). However, while the SA score differentiates between feasible and infeasible molecules to some extent, it does not provide specific insights into the actual synthesis pathways. Syntheseus operates by recursively decomposing a target molecule into increasingly simpler molecules through a backward reaction prediction model, continuing until it identifies a set of synthetically accessible building blocks [31]. In our experiments, we employ the Molecule Edit Graph Attention Network (MEGAN) backward reaction prediction model [36].

4 Methodology

This section presents the lipid building block dataset construction, the lipid property predictors, the reaction prediction model, and the policy network guided MCTS approach for lipid generation.

4.1 Dataset Construction

An ionizable lipid consists of an ionizable lipid head and several lipid tails. Our building block datasets contain synthetically accessible molecules that can act as lipid heads or lipid tails. We filter valid lipid building blocks from the ZINC20 database [17], a large-scale chemical database designed for drug discovery and virtual screening.

Figure 1 shows the flowcharts for constructing lipid head and lipid tail building block datasets. For the ionizable lipid head dataset, we first filter ionizable molecules from the ZINC20 molecule dataset based on specific criteria of LogP value, molecular weight, and amine groups. Since the lipid head building blocks are expected to react with lipid tail building blocks, we further filter reactive molecules by identifying whether or not the molecule contains certain functional groups that participate in common reactions. Detailed filtering criteria are presented in Appendix A.

We identify a lipid tail building block if the molecule is similar to an actual lipid tail (i.e., the tail component of an existing lipid). Taking advantage of the Lipid Analyzer[1], an implemented toolkit for lipid tail extraction, we extract lipid tails from a lipid dataset sourced from [46, 2]. We then conduct a similarity comparison to filter those molecules from the ZINC20 dataset that are similar to a real lipid tail. The resulting molecules make up our lipid tail building block dataset. Detailed information of the dataset construction process and selective examples of lipid building blocks are shown in Appendix A.

4.2 Lipid Property Prediction

Our generative approach relies on molecular property predictors to evaluate the potential of generated molecules to be an ionizable lipid. We employ a lipid classifier for binary assessment of lipid-likeness and an ionizability predictor to evaluate whether the generated product is ionizable.

Lipid Classifier Utilizing the Chemprop graph neural network framework [54], our lipid classifier features three message passing layers that integrate molecular features, complemented by two feed-forward layers dedicated to predicting molecular properties. The dataset we use to train the lipid classifier contains 180 000 lipid samples and 180 000 non-lipid samples. The non-lipid samples are small molecules from the PubChem database [20]. Part of the lipid samples come from publicly accessible lipid dataset LipidMaps and SwissLipids [46, 2]. The rest of the lipid samples are generated using a hierarchical graph encoder-decoder that employs significantly large and flexible graph motifs as basic building blocks [18].

Ionizability Predictor An ionizable molecule has neutral charge in physiological pH and becomes positively charged in acidic environments [6, 45]. For simpler filtering purpose, we only consider the net charge under pH = 7.4 (i.e., represents the physiological pH) and pH = 5 (i.e., represents the acidic environment). We utilize the MolGpKa module to determine acidic and basic groups within a given molecule and to estimate their respective pKa values [34]. The net charge of each molecule under a given pH value can then be calculated using the Henderson-Hasselbalch equation [35].

4.3 Reaction Prediction

We employ a template-based reaction prediction model which applies predefined reaction templates—derived from known chemical reaction mechanisms—to reactants. The advantages of this approach include faster computation, adherence to established chemical rules, and higher synthesis success potential. We adopt the same reaction templates as SyntheMol, our baseline model. The combinatorial chemical space explored by SyntheMol was the Enamine REadily AccessibLe (REAL) Space [13]. SyntheMol employs 13 reactions that account for 93.9% of the REAL Space [48]. Although our lipid building block dataset differs from the REAL space, the selected reactions are widely applicable across a broad range of chemicals, making them suitable for our project as well.

4.4 Guided Monte Carlo Tree Search for Lipid Generation

Building upon the SyntheMol and inspired by AlphaZero [48, 43], we apply the policy network guided MCTS to ionizable lipid generation. The integration of a policy network with MCTS forms the cornerstone of our approach, enabling a strategic exploration of chemical spaces through guided decision-making. In our problem formulation, the state is defined as the current molecule, and the action as the selection of the next building block. The policy network assigns probabilities to potential actions for each state. The training of the policy network is a cyclic process, aimed at progressively improving the network's ability to predict and prioritize effective synthetic pathways.

Figure 2 illustrates the workflow of the policy network training procedure. We start from a randomly initialized policy network which assigns equal probabilities to all the actions in the provided action space. This policy network will be used to guide the MCTS. We conduct the tree search for a number of simulations, and the search data (i.e., visit counts of all state-action pairs involved) of the tree search will be used as the training data to train the policy network for several epochs. We propose a customized policy network training technique, which is detailed in Appendix D. This process serves as one iteration of policy network training. In the next iteration, we use the updated policy network to guide MCTS and repeat the process.

[1]The Lipid Analyzer toolkit is an unpublished work.

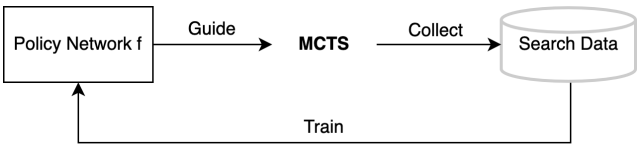


Figure 2: Workflow of policy network training.

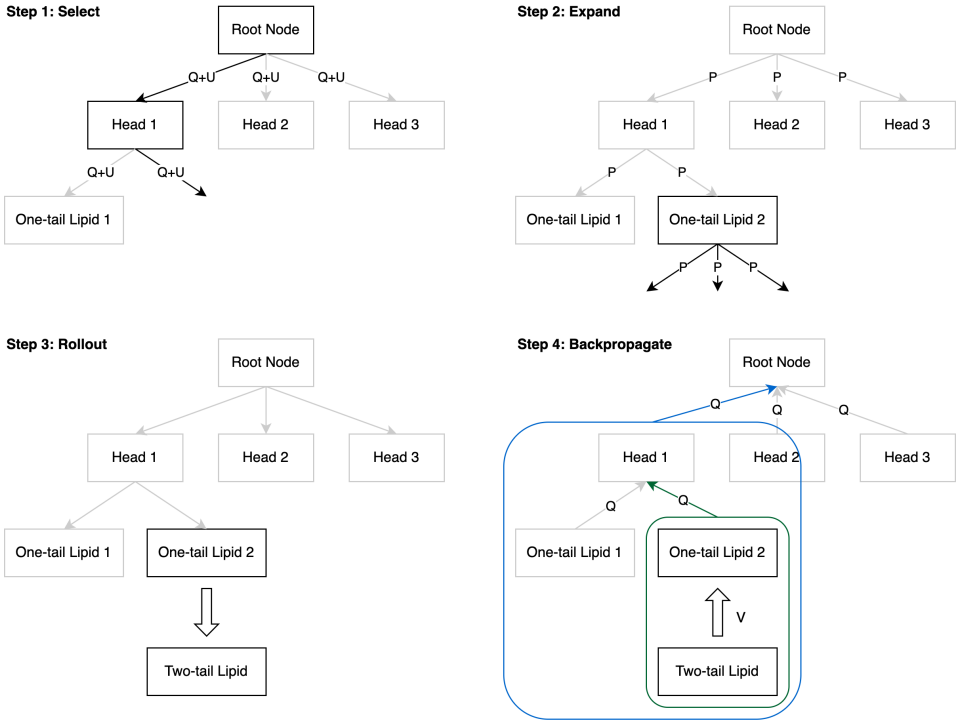


Figure 3: Policy network guided Monte Carlo tree search for lipid generation.

A step-wise depiction of each simulation of the MCTS is shown in Figure 3, with the detailed algorithm presented in Appendix C. Each simulation of the MCTS consists of four steps: select, expand, rollout, and backpropagate. We define the policy network that guides the MCTS simulations to be f_θ with parameters θ. Each edge in the tree search represents a state-action pair (s, a) where state s is the current molecule we have and action a is the next building block molecule to choose. Each edge stores a set of statistics $\{N(s,a), W(s,a), P(s,a)\}$ where $N(s,a)$ is the visit count, $W(s,a)$ is the total action value (i.e., sum of values of final products reached after taking action a from state s), and $P(s,a)$ is the prior probability of selecting that edge. Note that $P(s,\cdot) = f_\theta(s)$ is given by the current policy network.

In the selection step, each simulation traverses the tree by selecting the edge with the maximum Upper Confidence Bound (UCB) score until a leaf node is reached. At each time step t, our action selection criterion follows

$$a_t = \arg\max_a UCB\_score(s_t, a) = \arg\max_a (Q(s_t, a) + U(s_t, a)) \tag{1}$$

The UCB score is defined to be

$$UCB\_score(s,a) = Q(s,a) + U(s,a) \tag{2}$$

$$= \frac{W(s,a)}{N(s,a)} + c \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)} \tag{3}$$

where c is a constant parameter controlling the level of exploration.

In the expansion step, the leaf node selected in the previous step will be expanded. First, the state s of the leaf node will be calculated as the chemical product of the state and action molecules represented

by the edge which directs to this leaf node. We then find the action space $A(s)$ of the leaf node and calculate $P(s, a)$ values for all $a \in A(s)$ via the policy network, i.e., $P(s, \cdot) = f_\theta(s)$. The P values will be stored in the newly-added outgoing edges from the selected leaf node.

Meanwhile, the selected leaf node will be evaluated by the rollout step. This step aims to get a value for the selected leaf node. The rollout means performing random actions until we reach the end of the play (i.e., until we generate a two-tail lipid). This randomly generated product will be evaluated by the property predictor and this property score will act as the value of the selected leaf node.

Once we have the value of the selected leaf node, we perform the last step of the simulation, backpropagation. The value will be backpropagated along the chosen path to update action values Q.

5 Experiments

Our experimental investigation focuses on comparing the performance of the SyntheMol approach, served as the baseline, with our proposed policy network guided MCTS approach. We demonstrate the enhanced efficiency of the guided MCTS approach in identifying and synthesizing high-potential ionizable lipids.

5.1 Experimental Setups

Lipid Building Block Dataset Following the procedure described in Section 4.1, we construct a lipid building block dataset consisting of over 2.7 million lipid head building blocks and 5310 lipid tail building blocks. While directly incorporating such huge dataset into the MCTS would result in a predominantly exploratory behavior akin to random selections, we extract a subset of approximately 12 000 head building blocks to define our actual head search space. The entire tail building block dataset is utilized in subsequent levels of the search. For the evaluation of the policy network in the guided MCTS approach, an additional set of 200 testing head building blocks is employed.

SyntheMol Configuration We constrain the maximum number of child node expansions to be 2 000, meaning that each MCTS explores a head search space of this size. The MCTS is executed over 10 000 simulations to ensure comprehensive analysis of the generated lipid products. Additionally, the exploration weight c used in the UCB score calculation, as detailed in Equation 5, is set at 10.

Guided MCTS Configuration We operate the guided MCTS across 10 iterations. For every iteration, the MCTS is executed 10 times, each exploring a head space of 200. This setup allows the 10 runs of MCTS collectively to explore a total head search space size of 2 000, aligning with the head search space used in the SyntheMol approach. Each MCTS runs 10 000 simulations to gather substantial search data. The data from these 10 MCTS runs are pooled to train the policy network, and the accumulated generated products are analyzed as the iteration's output. The policy network undergoes 20 epochs of training in each iteration. The exploration weight c used in the UCB score calculation, as outlined in Equation 2, is set at 20.

Policy Network Configuration The policy network processes state-action pairs, where each state is the current molecule and the action is the next building block molecule selected. Each molecule is represented using a Morgan fingerprint with 1024-bit binary digit and radius 2 [38], and the fingerprints of both the state and action molecules are concatenated to form a 2048-bit feature vector, serving as the input to the policy network. Our policy network architecture consists of four linear layers, each followed by a ReLU activation function [11] and dropout layers with a dropout rate of 0.5 [49] to prevent overfitting. The Adam optimizer is used for network training, with a learning rate of 0.001 [21].

Computing Resources Our computational setup includes a GPU server equipped with 8 Tesla P-100 GPUs, each featuring 16 GB of memory. It is important to note that within our experimental framework, only the MolGpKa module is configured to utilize GPU resources [34]. All other components of our algorithms are designed to run efficiently on CPU.

5.2 Results and Discussions

We first present the performance of our lipid property predictors and then compare the generative results of the SyntheMol approach with our proposed policy network guided MCTS approach.

5.2.1 Lipid Property Prediction

As previously mentioned, we utilize a Chemprop based binary lipid classifier to determine whether a candidate molecule is lipid-like [54]. After training for a single epoch, our model demonstrates exceptional performance, achieving both Receiver Operating Characteristic Area Under the Curve (ROC-AUC) and Precision-Recall Area Under the Curve (PR-AUC) scores above 0.9999. Together, these metrics, along with a test accuracy of 99.89%, underscore the robustness and predictive accuracy of our lipid classifier.

In terms of ionizability, we utilize the MolGpKa module to identify ionizable groups within a target molecule and calculates their corresponding pKa values. We then determine the net charge of the target molecule at specific pH levels and conduct the ionizability filtering. According to [34], the MolGpKa predictor has undergone rigorous testing, thus justifying the credibility of our ionizability predictions.

To better validate our lipid property predictors, we curated an ionizable lipid dataset with more than 2 500 ionizable lipids that have been experimentally synthesized from published works [53, 26, 15, 28, 1, 55, 50]. We evaluated our predictors on this dataset, achieving an accuracy of 98.32% for our lipid classifier. Notably, all ionizable lipids were correctly classified by the ionizability predictor.

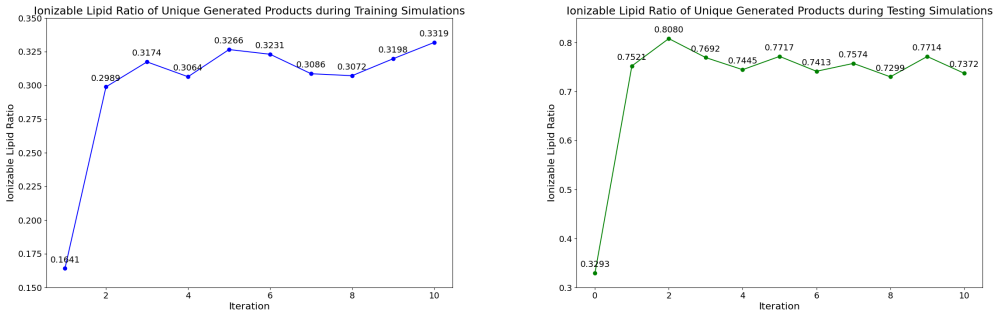
5.2.2 Generative Results

Our primary objective is to generate products that are potent candidates for ionizable lipids, meaning we aim for molecules with high property scores. We begin by examining the products generated via the SyntheMol approach, which serves as a baseline for comparing the performance enhancements achieved with the guided MCTS approach.

The property score combines a lipid classifier score, which typically hovers near 0 or 1, and a binary ionizability score. A property score approaching 2 typically identifies the molecule as a likely ionizable lipid. After conducting 10 000 simulations, a total of 16 477 two-tail lipids were generated. We observe that only 4513 of the 16 477 generated products are predicted to be ionizable lipids, providing an ionizable lipid rate at 0.2739. To evaluate the effectiveness of MCTS, we conducted experiments with random generation. Specifically, we generated 10 000 unique two-tail lipids through random combinations using the same lipid head and tail building block subsets, achieving an ionizable lipid rate of 0.1547. This result justifies the effectiveness of MCTS in guiding the generation of ionizable lipids.

We now turn our attention to the outcomes of the policy network guided MCTS approach. By comparing these results with those from the SyntheMol approach, we seek to illustrate the enhanced effectiveness of the policy network guided MCTS in producing ionizable lipids. We analyze the unique generated products from both training and testing simulations. The primary distinction between these simulations lies in their respective head building block search spaces. However, it is important to note that the selection of head molecules significantly influences the quality of the generated products. Nonetheless, observing the trend in quality changes within a specific head set, as guided by different iterations of the policy network, can still effectively illustrate the efficiency and impact of policy network training.

Figure 4a depicts the ionizable lipid rate cross training iterations. The initial results in iteration 1 are derived from MCTS simulations guided by a randomly initialized policy network, whereas the results from iteration 2 onward are influenced by successively trained instances of the policy network. A distinct increase in the ionizable lipid generation rate is evident after the first training session, as is shown in Figure 4a, which underscores a significant enhancement in quality attributable to the training of the policy network, affirming its efficacy in refining the generation process. Figure 4b presents the ionizable lipid rate during testing simulations, each guided by policy networks trained across different iterations. The initial iteration (i.e., iteration 0) features products generated by MCTS simulations directed by a randomly initialized policy network, with subsequent iterations using progressively trained policy networks. The results demonstrate a marked improvement in the ionizable lipid rate following the initial training iteration. Subsequent iterations show the rate stabilizing, with only minor fluctuations observed, oscillating between 0.73 and 0.8. In either case, these rates significantly surpass the ionizable lipid rate of below 0.3 achieved by the SyntheMol approach, clearly demonstrating the superior performance of our model over the baseline. A detailed



(a) Ionizable lipid rate during training simulations (b) Ionizable lipid rate during testing simulations

Figure 4: Ionizable lipid rate of the generated products during guided MCTS simulations.

Table 1: Evaluation of generated products and synthesis pathways.

	No. Unique Ionizable Lipids	Unique Ionizable Lipid Rate	Average SA Score	Retro Valid Rate by Syntheseus
Guided MCTS Train	5058	0.3319	4.62	0.4881
Guided MCTS Test	545	0.7372	4.24	0.2679
SyntheMol	4513	0.2739	4.77	0.2719
Random Combination	1547	0.1547	4.44	0.3103
Published Lipids	2563	/	4.12	0.0433

comparison of the ionizable lipid generation rate is presented in Table 1, where the rate for the guided MCTS corresponds to the value recorded after 10 iterations of policy network training.

5.2.3 Retrosynthesis Evaluation

Table 1 presents the evaluation results of our generated ionizable lipids and their corresponding synthesis pathways. We consider all unique products generated during both training and testing phases across all iterations of the guided MCTS approach. For better comparison, we also include the generated products from random combinations and the dataset of published ionizable lipids synthesized in previous works [53, 16, 30].

The SA score, which ranges from 1 to 10 with lower values indicating easier synthesis [10], reveals similar results for both the SyntheMol and guided MCTS approaches. Notably, there is no significant difference between the SA scores of our generated products and those of the published ionizable lipids. While the SA score justifies the synthesizability of the generate products to some extent, it does not offer any insights into their specific synthesis pathways.

Using Syntheseus, we aim to identify synthesis pathways for the generated products using a backward reaction prediction model. Given that a single generated product may have multiple possible synthesis pathways, we do not strictly adhere to those suggested by the generative model. Instead, we focus on identifying reactants for the generated products and checking whether these reactants are present in our lipid building block dataset. We record the proportion of generated products that can be synthesized using molecules from the building block dataset.

We observe that the retrosynthesis rate validated by Syntheseus is not high, with none of the cases exceeding 50%. The low rate may result from the unique characteristics of the lipid building block dataset and the reaction templates we adopted. For retrosynthesis planning, we utilize the MEGAN backward reaction prediction model trained on the USPTO-50k dataset [36]. Our lipid building block molecules may differ significantly from the reactants in the USPTO reactions, and our reaction templates may not align well with those in the USPTO dataset. Additionally, a lot of the reaction templates involve using linker molecules, while such reactions might not be present in the USPTO dataset. These mismatches make retrosynthesis planning challenging and contribute to the low retrosynthesis rate. Regarding the published ionizable lipids that have been experimentally

synthesized, we observe a retrosynthesis valid rate below 5%. This is likely because these lipids were synthesized using building blocks outside our building block dataset. Our building block dataset may lack many structures not included in the ZINC dataset. Additionally, our choice of weight cutoff and restrictions on amine-containing structures during lipid head dataset construction further reduce the search space.

Selected examples of generated ionizable lipids, along with their synthesis pathways validated by the retrosynthesis tool, are presented in Appendix E.

6 Conclusion

In this work, we have explored the Monte Carlo tree search (MCTS)-based generative models for ionizable lipid generation. We constructed a lipid building block dataset, featuring synthetically accessible ionizable lipid heads and tails, which is well-suited for future lipid generation tasks. We developed two lipid property predictors: a lipid classifier and an ionizability predictor, both designed to accurately assess whether a candidate molecule is lipid-like or ionizable. Adapting the SyntheMol approach, originally utilized for antibiotic discovery, we tailored this method for lipid generation to serve as our baseline model. We further innovated by developing a policy network guided MCTS-based generative model which is capable of producing high-quality ionizable lipids with available synthesis paths, outperforming our baseline. Our achievements indicate that this project offers a promising direction for ionizable lipid generation, also contributing to the broader field of drug delivery.

Nonetheless, it is important to acknowledge that the work has its limitations. If condition allowed, we may conduct experimental validations of the existing reaction templates with lipid building blocks to ensure their applicability and efficiency in lipid synthesis. It will be helpful to develop and integrate additional reaction templates that are specific to lipid chemistry. To address the limitation of not having practical synthesis validations, future research should prioritize establishing collaborations with chemical laboratories. This will enable empirical testing and validation of the synthesized molecules, providing a direct assessment of their practical viability and safety. Develop or customize computer-aided retrosynthesis tools specifically for ionizable lipid generation can also be helpful.

Acknowledgments and Disclosure of Funding

The authors would like to thank Asal Mehradfar and Mohammad Shahab Sepehri for curating the lipid dataset used for training lipid classifier and for building the Lipid Analyzer toolkit.

References

[1] Mahmoud M. Abd Elwakil, Ryota Suzuki, Alaa M. Khalifa, Rania M. Elshami, Takuya Isono, Yaser H.A. Elewa, Yusuke Sato, Takashi Nakamura, Toshifumi Satoh, and Hideyoshi Harashima. Harnessing topology and stereochemistry of glycidylamine-derived lipid nanoparticles for in vivo mrna delivery to immune cells in spleen and their application for cancer vaccination. *Advanced Functional Materials*, 33(45):2303795, 2023.

[2] Lucila Aimo, Robin Liechti, Nevila Hyka-Nouspikel, Anne Niknejad, Anne Gleizes, Lou Götz, Dmitry Kuznetsov, Fabrice P. A. David, F. Gisou van der Goot, Howard Riezman, Lydie Bougueleret, Ioannis Xenarios, and Alan Bridge. The swisslipids knowledgebase for lipid biology. *Bioinformatics*, 31(17):2860–2866, 2015.

[3] Rim Assouel, Mohamed Ahmed, Marwin H. S. Segler, Amir Saffari, and Yoshua Bengio. Defactor: Differentiable edge factorization-based probabilistic graph generation. *CoRR*, abs/1811.09766, 2018.

[4] Yuemin Bian, Junmei Wang, Jaden Jungho Jun, and Xiang-Qun Xie. Deep convolutional generative adversarial network (dcGAN) models for screening and design of small molecules targeting cannabinoid receptors. 16(11):4451–4460, 2019. Publisher: American Chemical Society.

[5] John Bradshaw, Brooks Paige, Matt J. Kusner, Marwin H. S. Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags, 2020.

[6] Manuel J. Carrasco, Suman Alishetty, Mohamad-Gabriel Alameh, Hooda Said, Lacey Wright, Mikell Paige, Ousamah Soliman, Drew Weissman, Thomas E. Cleveland, Alexander Grishaev, and Michael D. Buschmann. Ionization and structural properties of mRNA lipid nanoparticles influence expression in intramuscular and intravascular administration. 4(1):956, 2021.

[7] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, 2006.

[8] Isabelle M. S. Degors, Cuifeng Wang, Zia Ur Rehman, and Inge S. Zuhorn. Carriers break barriers in drug delivery: Endocytosis and endosomal escape of gene delivery vectors. 52(7):1750–1760, 2019. Publisher: American Chemical Society.

[9] Daisy Yi Ding, Yuhui Zhang, Yuan Jia, and Jiuzhi Sun. Machine learning-guided lipid nanoparticle design for mrna delivery, 2023.

[10] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. 1(1):8, 2009.

[11] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Trans. Syst. Sci. Cybern.*, 5:322–333, 1969.

[12] Wenhao Gao and Connor W. Coley. The synthesizability of molecules proposed by generative models, 2020.

[13] Oleksandr O. Grygorenko, Dmytro S. Radchenko, Igor Dziuba, Alexander Chuprina, Kateryna E. Gubina, and Yurii S. Moroz. Generating multibillion chemical space of readily accessible screening compounds. *iScience*, 23(11):101681, 2020. Published correction appears in iScience. 2020 Dec 04;23(12):101873. doi: 10.1016/j.isci.2020.101873.

[14] 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, January 2018.

[15] Zepeng He, Zhicheng Le, Yi Shi, Lixin Liu, Zhijia Liu, and Yongming Chen. A multidimensional approach to modulating ionizable lipids for high-performing and organ-selective mrna delivery. *Angewandte Chemie International Edition*, 62(43):e202310401, 2023.

[16] Sebastian G. Huayamares, Melissa P. Lokugamage, Regina Rab, Alejandro J. Da Silva Sanchez, Hyejin Kim, Afsane Radmand, David Loughrey, Liming Lian, Yuning Hou, Bhagelu R. Achyut, Annette Ehrhardt, Jeong S. Hong, Cory D. Sago, Kalina Paunovska, Elisa Schrader Echeverri, Daryll Vanover, Philip J. Santangelo, Eric J. Sorscher, and James E. Dahlman. High-throughput screens identify a lipid nanoparticle that preferentially delivers mrna to human tumors in vivo. *Journal of Controlled Release*, 357:394–403, 2023.

[17] John J. Irwin, Khanh G. Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R. Wong, Munkhzul Khurelbaatar, Yurii S. Moroz, John Mayfield, and Roger A. Sayle. ZINC20—a free ultralarge-scale chemical database for ligand discovery. *Journal of Chemical Information and Modeling*, 60(12):6065–6073, 2020. PMID: 33118813.

[18] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. *CoRR*, abs/2002.03230, 2020.

[19] M. Kim, M. Jeong, S. Hur, Y. Cho, J. Park, H. Jung, Y. Seo, H. A. Woo, K. T. Nam, K. Lee, and H. Lee. Engineered ionizable lipid nanoparticles for targeted delivery of rna therapeutics into different types of cells in the liver. *Science Advances*, 7(9):eabf4398, 2021.

[20] Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, Jiyao Wang, Bo Yu, Jian Zhang, and Stephen H. Bryant. Pubchem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, 2016.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[22] Levente Kocsis and Csaba Szepesvari. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, 2006.

[23] Byoungjae Kong, Yelee Kim, Eun Hye Kim, Jung Soo Suk, and Yoosoo Yang. mrna: A promising platform for cancer immunotherapy. *Advanced Drug Delivery Reviews*, 199:114993, 2023.

[24] Ruvanthi Kularatne, Rachael Crist, and Stephan Stern. The future of tissue-targeted lipid nanoparticle-mediated nucleic acid delivery. *Pharmaceuticals*, 15:897, 07 2022.

[25] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder, 2017.

[26] Bowen Li, Rajith Singh Manan, Shun-Qing Liang, Akiva Gordon, Allen Jiang, Andrew Varley, Guangping Gao, Robert Langer, Wen Xue, and Daniel Anderson. Combinatorial design of nanoparticles for pulmonary mRNA delivery and genome editing. 41(10):1410–1415, 2023.

[27] Bowen Li, Idris Raji, Akiva Gordon, Lizhuang Sun, Theresa Raimondo, Favour Oladimeji, Allen Jiang, Andrew Varley, Robert Langer, and Daniel Anderson. Accelerating ionizable lipid discovery for mrna delivery using machine learning and combinatorial chemistry. *Nature Materials*, 23:1–7, 05 2024.

[28] Shuai Liu, Qiang Cheng, Tuo Wei, Xueliang Yu, Lindsay T. Johnson, Lukas Farbiak, and Daniel J. Siegwart. Membrane-destabilizing ionizable phospholipids for organ-selective mRNA delivery and CRISPR–cas gene editing. 20(5):701–710, 2021.

[29] Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19784–19795. Curran Associates, Inc., 2021.

[30] Han Han Ly, Simon Daniel, Shekinah K. V. Soriano, Zoltán Kis, and Anna K. Blakney. Optimization of lipid nanoparticles for sarna expression and cellular activation using a design-of-experiment approach. *Molecular Pharmaceutics*, 19(6):1892–1905, 2022. PMID: 35604765.

[31] Krzysztof Maziarz, Austin Tripp, Guoqing Liu, Megan Stanley, Shufang Xie, Piotr Gaiński, Philipp Seidl, and Marwin Segler. Re-evaluating retrosynthesis algorithms with syntheseus. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[32] Michael J. Mitchell, Margaret M. Billingsley, Rebecca M. Haley, Marissa E. Wechsler, Nicholas A. Peppas, and Robert Langer. Engineering precision nanoparticles for drug delivery. 20(2):101–124, 2021.

[33] Saeed Moayedpour, Jonathan Broadbent, Saleh Riahi, Michael Bailey, Hoa Thu, Dimitar Dobchev, Akshay Balsubramani, Ricardo Santos, Lorenzo Kogler-Anele, Alejandro Corrochano-Navarro, Sizhen Li, Fernando Montoya, Vikram Agarwal, Ziv Bar-Joseph, and Sven Jager. Representations of lipid nanoparticles using large language models for transfection efficiency prediction. *Bioinformatics (Oxford, England)*, 40, 05 2024.

[34] Xiaolin Pan, Hao Wang, Cuiyu Li, John Zeng Hui Zhang, and Changge Ji. MolGpka: A web server for small molecule pka prediction using a graph-convolutional neural network. *Journal of chemical information and modeling*, 2021.

[35] Ralph H. Petrucci, William S. Harwood, and F. Geoffrey Herring. *General Chemistry: Principles and Modern Applications*. Prentice Hall, Upper Saddle River, NJ, 8th edition, 2002. 1 volume (various pagings): illustrations (some color); 26 cm.

[36] Mikolaj Sacha, Mikolaj Blaz, Piotr Byrski, Pawel Wlodarczyk-Pruszynski, and Stanislaw Jastrzebski. Molecule edit graph attention network: Modeling chemical reactions as sequences of graph edits. *CoRR*, abs/2006.15426, 2020.

[37] Ugur Sahin, Katalin Karikó, and Özlem Türeci. mRNA-based therapeutics — developing a new class of drugs. 13(10):759–780, 2014.

[38] Nadine Schneider, Daniel Lowe, Roger Sayle, and Gregory Landrum. Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity. *Journal of chemical information and modeling*, 55, 02 2015.

[39] Marwin H. S. Segler, Mike Preuß, and Mark P. Waller. Towards "alphachem": Chemical synthesis planning with tree search and deep neural network policies. *CoRR*, abs/1702.00020, 2017.

[40] Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. 555(7698):604–610, 2018.

[41] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation, 2021.

[42] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[43] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

[44] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders, 2018.

[45] Stephan T. Stern and Scott E. McNeil. Nanotechnology safety concerns revisited. 101(1):4–21, 2007. _eprint: https://academic.oup.com/toxsci/article-pdf/101/1/4/10976607/kfm169.pdf.

[46] Manish Sud, Eoin Fahy, Deirdre Cotter, H. Alex Brown, Edward A. Dennis, Christopher K. Glass, Alfred H. Jr. Merrill, Robert C. Murphy, Christian R. H. Raetz, David W. Russell, and Shankar Subramaniam. LMSD: Lipid maps® structure database. Nucleic Acids Research, 2007. PMID: 17098933.

[47] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[48] Kyle Swanson, Gary Liu, Denise B. Catacutan, Autumn Arnold, James Zou, and Jonathan M. Stokes. Generative ai for designing and validating easily synthesizable and structurally novel antibiotics. *Nature Machine Intelligence*, 6:338–353, 2024.

[49] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[50] Tuo Wei, Yehui Sun, Qiang Cheng, Sumanta Chatterjee, Zachary Traylor, Lindsay T. Johnson, Melissa L. Coquelin, Jialu Wang, Michael J. Torres, Xizhen Lian, Xu Wang, Yufen Xiao, Craig A. Hodges, and Daniel J. Siegwart. Lung SORT LNPs enable precise homology-directed repair mediated CRISPR/cas genome correction in cystic fibrosis models. 14(1):7322, 2023.

[51] Anders Wittrup, Angela Ai, Xing Liu, Peter Hamar, Radiana Trifonova, Klaus Charisse, Muthiah Manoharan, Tomas Kirchhausen, and Judy Lieberman. Visualizing lipid-formulated siRNA release from endosomes and target gene knockdown. 33(8):870–876, 2015.

[52] Emily Xu, W. Mark Saltzman, and Alexandra S. Piotrowski-Daspit. Escaping the endosome: assessing cellular trafficking mechanisms of non-viral vehicles. *Journal of Controlled Release*, 335:465–480, 2021.

[53] Yue Xu, Shihao Ma, Haotian Cui, Jingan Chen, Shufen Xu, Fanglin Gong, Alex Golubovic, Muye Zhou, Kevin Chang Wang, Andrew Varley, Rick Xing Ze Lu, Bo Wang, and Bowen Li. AGILE platform: a deep learning powered approach to accelerate LNP development for mRNA delivery. 15(1):6305, 2024.

[54] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019. Correction published: J Chem Inf Model. 2019 Dec 23;59(12):5304-5305. doi: 10.1021/acs.jcim.9b01076.

[55] Xueliang Yu, Shuai Liu, Qiang Cheng, Tuo Wei, Sang Lee, Di Zhang, and Daniel J. Siegwart. Lipid-modified aminoglycosides for mrna delivery to the liver. *Advanced Healthcare Materials*, 9(7):1901487, 2020.

A Lipid Building Block Dataset Construction and Selected Examples

We here discuss in detail how the lipid building block dataset construction process is performed and present selected examples of lipid building blocks.

For the ionizable lipid head dataset, we first filter ionizable molecules from the ZINC20 molecule dataset based on the following three criteria:

1. Molecular weight < 500 g/mol.
2. Log P < 0 where the log P is the logarithm (base 10) of the partition coefficient P, which is the ratio of the concentrations of a compound in a mixture of two immiscible phases: typically a hydrophobic solvent and water.
3. Molecules with amine functional groups but not ammonium based molecules.

Since the lipid head building blocks are expected to react with lipid tail building blocks to generate product lipids, we further filter reactive molecules from our ionizable molecule set. We identify reactive molecules by filtering whether or not the molecule contains certain functional groups that participate in common reactions. Specifically, we check whether our candidate molecule contains carboxyl group (i.e., -COOH), hydroxyl group (i.e., -OH), or amine group (i.e., -N or -NH or -NH2). If the candidate molecule contains any one or more of the target functional groups, we consider the molecule to be reactive. Our filtered reactive ionizable molecule set becomes our ionizable lipid head dataset.

In terms of lipid tail building block dataset, we again filter from the ZINC20 molecule dataset so that our lipid tail building blocks are purchasable. We identify a lipid tail building block if the molecule is similar to an actual lipid tail (i.e., the tail component of an existing lipid). We use a lipid dataset sourced from the LipidMaps database and the SwissLipid database [46, 2]. Leveraging the Lipid Analyzer, an implemented toolkit for lipid tail extraction, we extract lipid tails from this lipid dataset. The Lipid Analyzer finds the lipid head for a given lipid, we then extract lipid tails by removing the head substructure. For a given molecular structure, the algorithm first recognizes ring structures. For all possible arrangements of the rings, the algorithm then removes carbon atoms that are not near any hydrophilic atom and not forming any ring. If only one substructure remains after the operation, and if the log P value of this substructure is low enough, this substructure will be identified as the lipid head. It's then easy to extract tail substructures by removing the head substructure.

We then conduct a similarity comparison to filter those molecules from the ZINC20 dataset that are similar to a real lipid tail. The similarity is measured by the Graph Edit Distance (GED) between two molecular structures, quantifying the minimum number of operations required to transform one graph into another. We only select molecules that has GED value smaller or equal to 1 with a real lipid tail. Meanwhile, we also consider the similarity between molecular fingerprint representations, Daylight fingerprint and Extended Connectivity Fingerprint 4 (ECFP4) are considered. The resulting molecules make up our lipid tail building block dataset.

Selected examples of our lipid head building blocks with different functional groups are shown in Figure 5. The first row presents examples with no carboxyl group, one hydroxyl group, and two amine groups; the second row presents examples with one carboxyl group, no hydroxyl group, and one amine group; the third row presents examples with two carboxyl groups, no hydroxyl group, and one amine group. Note that we only consider independent amine groups that are linked to only simple carbon atoms. For example, in the first example of the first row, we find a secondary amine group (i.e., -NH) linked to a carbonyl group (i.e., C=O, a carbon atom double-bonded to an oxygen atom). The linkage with the carbonyl group forms a more complicated substructure, which influences the reactive performance of the amine group. We therefore exclude this secondary amine group when counting our target functional groups. Similarly, the nitrogen-nitrogen bond (i.e., N-N) which appears in a five-membered nitrogen-containing ring influences the reactive property of the nitrogen, and we also exclude these nitrogen atoms when counting the occurrence of amine functional groups.

Figure 6 shows selected examples of lipid tail building blocks. As we can see, a lipid tail usually consists of a carbon chain and a functional group. The functional group may be hydroxyl group, amine group, or a halogen atom.

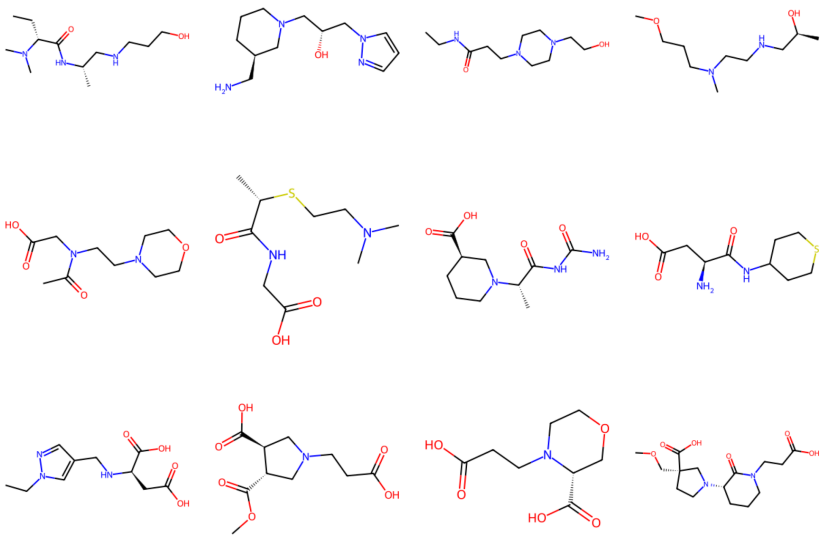


Figure 5: Selected examples of lipid head building blocks.

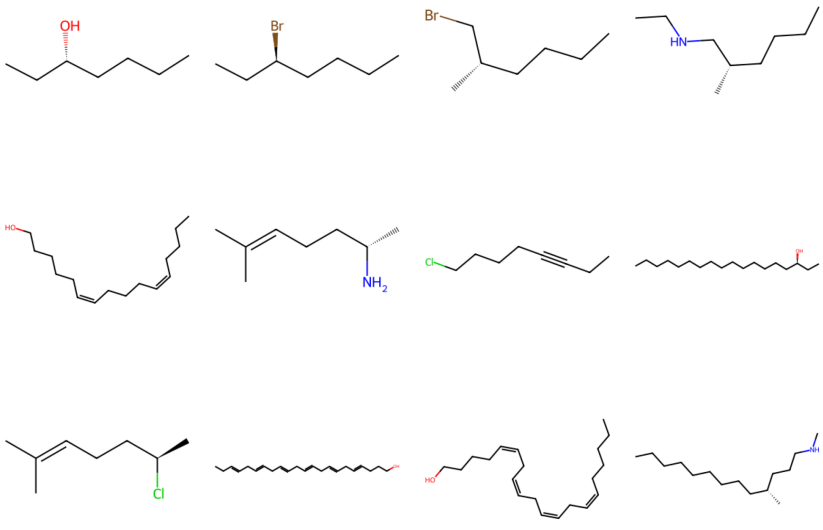


Figure 6: Selected examples of lipid tail building blocks.

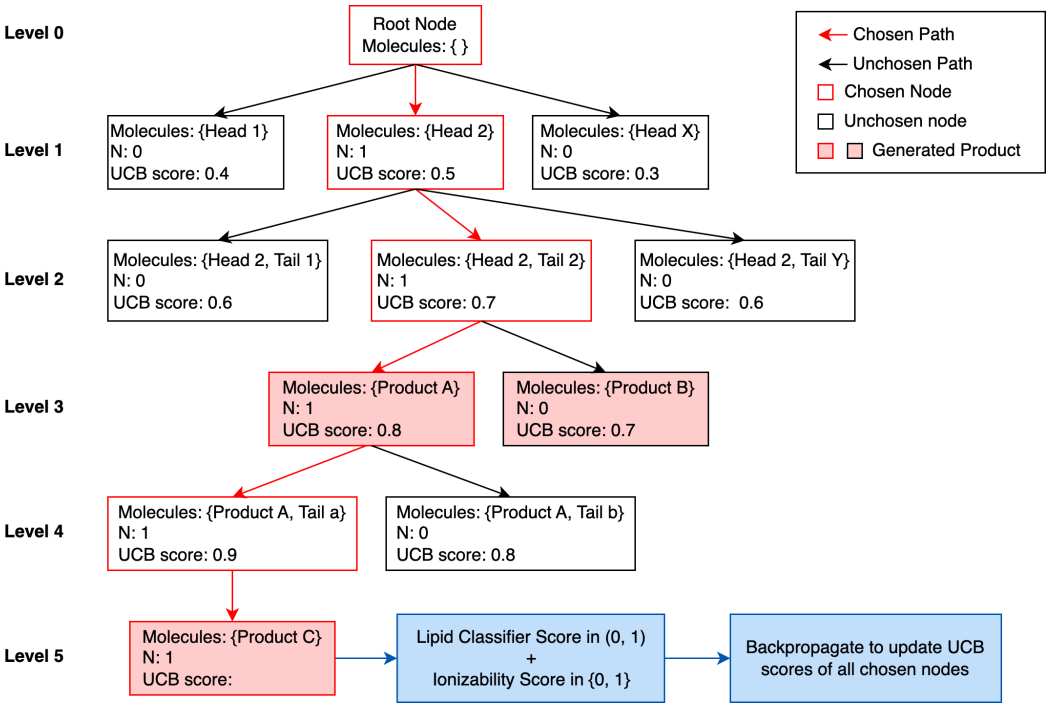


Figure 7: SyntheMol for lipid generation.

B SyntheMol for Lipid Generation

Figure 7 illustrates a typical simulation of the MCTS process in our lipid generation. In the search tree, each node stores crucial statistics: the molecules represented by the node, the node's visit count N, and the Upper Confidence Bound (UCB) score. The UCB score guides the node selection process using the UCB action selection criterion [47, 22]. The UCB score combines an action value Q, which represents the average of the total action values W (i.e., the sum of the values of the final products passing through the node), with an upper confidence term U. This term is defined as $U(node) \propto \frac{P(node)}{1+N(node)}$, where $P(node)$ indicates the property scores assigned to the node. The complete formula for the UCB score is as follows:

$$UCB\_score(node) = Q(node) + U(node) \tag{4}$$

$$= \frac{W(node)}{N(node)} + c \cdot P(node) \cdot \frac{\sqrt{1+n}}{1+N(node)} \tag{5}$$

where n is the total visit count of all nodes at the same level as the target node and c is a constant exploration parameter controlling the level of exploration. Note that the P value is the property score calculated by our property predictors. When a node represents multiple molecules, the P value for that node is computed as the average of the property scores for each constituent molecule.

It is important to note that the assignment of property scores to nodes at non-terminal levels—representing either building block molecules or intermediates—may initially seem illogical, as direct evaluation of these entities' properties does not typically yield meaningful insights. However, this approach does not compromise the long-term efficacy of the algorithm. Over time, as more product molecules are synthesized and action values refined, the utility of early-stage evaluations is validated. SyntheMol's findings corroborate this approach, demonstrating that despite low scores of individual building blocks, the synthesized molecules often exhibit significantly higher scores, which are effectively identified by MCTS, highlighting its capability to uncover promising molecules overlooked by simpler scoring methods [48].

The MCTS algorithm commences at a root node (level 0), an empty initial node, which is then expanded with child nodes representing available lipid head building blocks defined within our chemical space (level 1). Each child node receives an UCB score, with the node exhibiting the highest

UCB score selected for further expansion. This selected node is expanded to include a lipid tail building block, generating second-level nodes where each combination represents potential reactants (level 2).

Subsequently, the node with the highest UCB score from level 2 is selected and expanded. This stage differs from prior expansions in that it now entails actual chemical reactions between the two building blocks within the node. The resulting third-level nodes embody all conceivable products predicted by forward reaction mechanisms (level 3). This iterative expansion continues until either a valid final product (i.e., a two-tail lipid in our case) is synthesized, or further reactions become untenable with the existing building blocks in our dataset.

Upon terminating the simulation with a valid final product, the synthesized molecule is assessed using our property predictors. The resultant property value is then backpropagated to update the UCB scores along the simulated pathway. For analysis purpose, we meticulously document all products, including intermediates, generated during each simulation.

C Algorithm of Guided Monte Carlo Tree Search for Lipid Generation

We define a `Node` class to represent the nodes appeared in the tree search. The `Node` class has the following attributes:

- `state`: Stores the SMILES representation of the molecule represented by this node.
- `N`: A visit count of the node, initialized to zero.
- `P`: The prior probability assigned based on predictions from a policy network.
- `W`: The cumulative value sum, representing the total assessed value of this node's state.
- `children`: A dictionary to hold child nodes, with the keys of the dictionary to be actions (i.e., the next building blcoks to select) and the values to be corresponding child nodes.

Algorithm 1 Policy Network Guided MCTS

Require: Product(): reaction predictor
Require: PropertyScore(): property score predictor
Require: SearchProbability(): search probability calculator
$f_\theta \leftarrow$ randomly initialized neural network with parameter θ
$\lambda \leftarrow$ regularization constant
$\alpha \leftarrow$ learning rate
$c \leftarrow$ exploration weight
$D = \emptyset$

for each iteration do
  for each play do
    $S_0 \leftarrow$ empty root state
    $D_{play} \leftarrow \text{MCTS}(S_0)$
    $D = D \cup D_{play}$
  end for
  for each epoch do
    Train f_θ using D
  end for
  Reset $D = \emptyset$
end for

function MCTS($root\_state$)
  Initialize $root\_node$ with $root\_state$
  EXPAND($root\_node$)
  $generation = \emptyset$
  for each simulation do
    $leaf\_node, search\_path \leftarrow$ SELECT($root\_node$)
    if $leaf\_node$ is two-tail lipid **then**
      v = PropertyScore($leaf\_node$)
      Add $search\_path$ to $generation$
    else
      EXPAND($leaf\_node$)
      $v \leftarrow$ ROLLOUT($leaf\_node$)
    end if
    BACKPROPAGATE(v, $search\_path$)
  end for
  Write $generation$ to log file
  return visit counts of all state-action pairs
end function

Algorithm 2 Select Function in MCTS

function SELECT($root\_node$)
  $search\_path = [\,]$
  $node \leftarrow root\_node$
  while $node.children$ is not empty **do**
    $selected\_action = \arg\max_a$ UCB_SCORE($node, node.children[a]$)
    $node \leftarrow node.children[selected\_action]$
    Add $node$ to $search\_path$
  end while
  Update $node.state$ with Product($node.state, selected\_action$)
  return $node$, $search\_path$
end function

Algorithm 3 Expand Function in MCTS

function EXPAND($node$)
    $A(node) \leftarrow$ NEXT_BUILDING_BLOCKS($node$)
    $P(node, \cdot) = f_\theta(node.state, A(node))$
    for a in $A(node)$ **do**
        Initialize $child\_node$ with $P(node, a)$
        $node.children[a] = child\_node$
    end for
end function

Algorithm 4 Backpropagate Function in MCTS

function BACKPROPAGATE(v, $search\_path$)
    for $node$ in $search\_path$ **do**
        $node.N \leftarrow node.N + 1$
        $node.W \leftarrow node.W + v$
        $root\_node.N \leftarrow root\_node.N + 1$
    end for
end function

Algorithm 5 Rollout Function in MCTS

function ROLLOUT($node$)
    while $node.state$ is not two-tail lipid **do**
        $A(node) \leftarrow$ NEXT_BUILDING_BLOCKS($node$)
        return 0 if $A(node)$ is empty
        $a \leftarrow$ a random choice from $A(node)$
        $node \leftarrow$ new node with Product($node.state$, a)
    end while
    return PropertyScore($node$)
end function

Algorithm 6 UCB Score Calculation Function in MCTS

function UCB_SCORE($parent\_node$, $child\_node$)
    if $child\_node.N = 0$ **then**
        $Q = 0$
    else
        $Q = \frac{child\_node.W}{child\_node.N}$
    end if
    $U = c \cdot child\_node.P \cdot \frac{\sqrt{parent\_node.N}}{child\_node.N+1}$
    return $Q + U$
end function

Algorithm 7 Get Action Space Function in MCTS

Require: max_expand_num a pre-determined number
function NEXT_BUILDING_BLOCKS($node$)
    if $node$ is empty node **then**
        $A(node) \leftarrow max\_expand\_num$ random samples from head search space
    else
        $reactive\_tail\_set \leftarrow$ tails which can react with $node.state$
        $A(node) \leftarrow max\_expand\_num$ random samples from $reactive\_tail\_set$
    end if
    return $A(node)$
end function

D Policy Network Training

As is mentioned before, the visit counts of all state-action pairs that appear in the tree search are recorded to be the training data of the policy network. However, the search data we collect are highly imbalanced in the sense that there will be a lot more state-action pairs in later levels. The number of state-action pairs from the first level (i.e., when state is the empty state and actions are the chosen head building blocks) is very limited, being the number of different head building blocks that are expanded by the root node. Meanwhile, search data from later levels will be very sparse in the sense that the majority of the state-action pairs will have zero visit count. In order to better utilize our limited data, we propose a customized method for training the policy network.

Let $f(s, a)$ denote the naive policy network output (before softmax) for the state-action pair (s, a), $p(s, a)$ denote the corresponding predicted prior (which will be used in the UCB score calculation), and $\pi(s, a)$ be the search probability. We here adopt the search probability definition as proposed in the AlphaZero algorithm [43].

$$p(s, a_1) = \frac{e^{f(s,a_1)}}{\sum_a e^{f(s,a)}} \tag{6}$$

$$\pi(s, a_1) = \frac{N(s, a_1)^{\frac{1}{\tau}}}{\sum_a N(s, a)^{\frac{1}{\tau}}} \tag{7}$$

where τ is a temperature parameter controlling the level of exploration. When τ tends to infinity, the search probability is the same as random selections. When τ is small, say, when $\tau = 1$, the search probability strictly follows the actual visit count distribution. The objective of the policy network training is to maximize the similarity of $p(s, a)$ and $\pi(s, a)$.

Now, suppose we take two state-action pairs, (s, a_1) and (s, a_2), at a time. We apply the log-ratio and get:

$$\log \frac{p(s, a_1)}{p(s, a_2)} = f(s, a_1) - f(s, a_2) \tag{8}$$

$$\log \frac{\pi(s, a_1)}{\pi(s, a_2)} = \frac{1}{\tau} \log N(s, a_1) - \frac{1}{\tau} \log N(s, a_2) \tag{9}$$

The previous objective is equivalent to maximize the similarity of $f(s, a_1) - f(s, a_2)$ and $\frac{1}{\tau} \log N(s, a_1) - \frac{1}{\tau} \log N(s, a_2)$.

We therefore customize the loss function to be the error between $f(s, a_1) - f(s, a_2)$ and their corresponding $\frac{1}{\tau} \log N(s, a_1) - \frac{1}{\tau} \log N(s, a_2)$, for any two state-action pairs, (s, a_1) and (s, a_2). This can be considered as a regression problem, MAE or MSE loss may be applied.

E Selected Examples of Synthesis Planning

Figures 8 and 9 give two examples of our generated ionizable lipids with their corresponding synthesis paths. Note that these synthesis paths have been validated by the retrosynthesis tool Syntheseus [31]. In the figures below, the blue nodes represent product molecules (i.e., intermediate product or final product) and the green nodes represent building block molecules (i.e., lipid head building block or lipid tail building block).

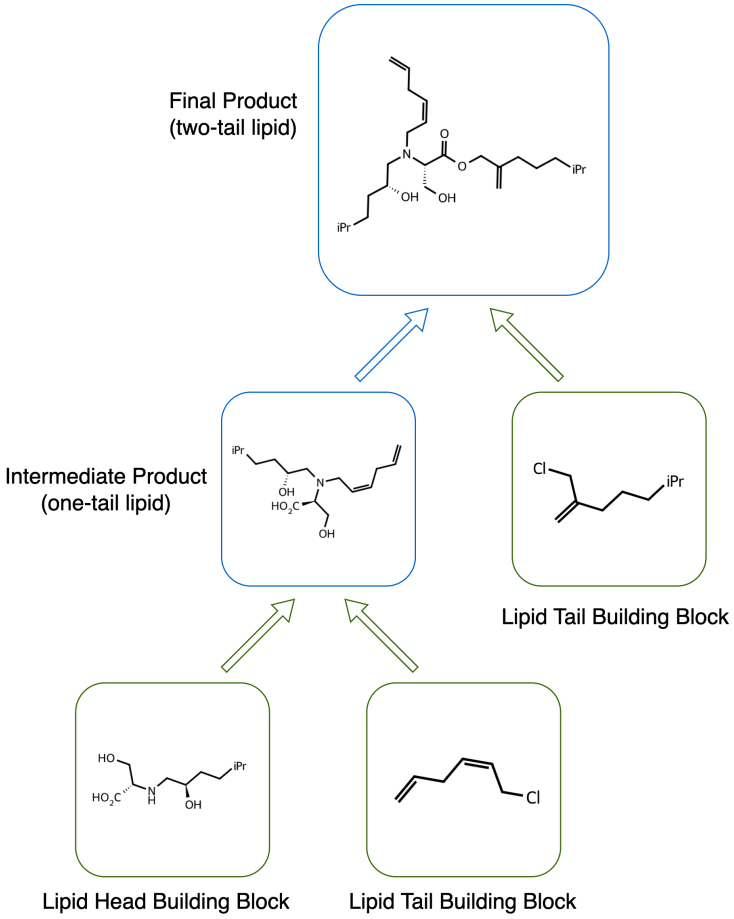


Figure 8: Example 1 of generated ionizable lipid with synthesis path.

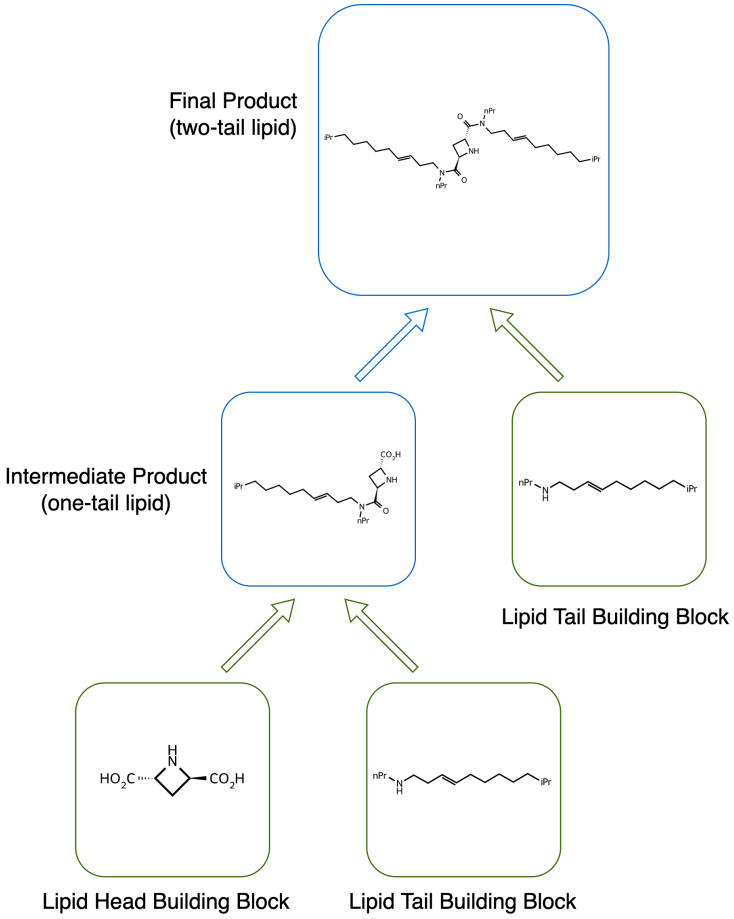


Figure 9: Example 2 of generated ionizable lipid with synthesis path.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The authors clearly stated the paper's contributions and scope in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer:[NA]

Justification: We do not have theoretical results that need theoretical proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer:[Yes]

   Justification: All experiment details are stated in section 5.1 experimental setups.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All source codes are publicly available

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experiment details are stated in section 5.1 experimental setups.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because of computation cost.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All computational details are stated in section 5.1 experimental setups.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that we followed, in every respect, with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential impact of the research in section 6 conclusion.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All source codes, models, and dataset we used are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All source codes of the paper are publicly available and well documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.