SMALL MOLECULE OPTIMIZATION WITH LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements in large language models (LLMs) have opened new possibilities for generative molecular drug design. In molecular optimization, LLMs are promising candidates to augment traditional modeling and rule-based approaches for refining molecular structures toward design criteria. We present a novel approach to molecular optimization using LLMs trained on a hand-crafted corpus of over 100 million molecules and their properties. We trained three new models, Chemlactica-125M, Chemlactica-1.3B, and Chemma-2B, with a demonstrated ability to generate molecules with specified properties and learn new molecular characteristics from limited samples, competitive with the state-of-the-art (SOTA) in property prediction tasks on experimental data. Our optimization method, elucidated by these capabilities, combines the models' generative power with concepts from prompt optimization, evolutionary algorithms, and rejection sampling to solve molecular optimization problems more efficiently. The approach surpasses previous SOTA results on the Practical Molecular Optimization (PMO) benchmark and exceeds or is competitive with the SOTA in multi-property optimization tasks involving docking simulations. We release the training data, language models, and optimization algorithm to facilitate further research and reproducibility.

1 INTRODUCTION

Molecular optimization is a cornerstone of drug discovery, involving identifying compounds with specific desirable properties(Hughes et al., 2011). This process traditionally requires extensive laboratory experimentation, making it time-consuming and costly. Computational methods have emerged as powerful tools to accelerate this process, yet they often struggle with the vast and discrete nature of chemical space (Wu et al., 2018; Schneider, 2018).

Large language models (LLMs) have recently demonstrated remarkable capabilities across various domains, from natural language processing to code generation (Brown et al., 2020; OpenAI, 2023; Zhang et al., 2023). While there have been initial attempts to apply LLMs to chemical tasks (Irwin et al., 2022; Edwards et al., 2022; Chilingaryan et al., 2024), these efforts were often limited in scope or performance. Our work represents a significant leap forward, leveraging the full power of LLMs to revolutionize molecular optimization for drug discovery.

We present a novel approach that harnesses LLMs to generate and optimize small molecules with unprecedented efficiency and accuracy. Our method uniquely combines LLMs' generative capabilities with evolutionary strategies, enabling more effective exploration of chemical space than traditional graph-based or SMILES-based models.

Our research makes several contributions to the field:

1. We develop a comprehensive molecular corpus derived from PubChem (Kim et al., 2015), encompassing over 110 million molecules and their properties. This corpus, richer in chemical information compared to SMILES-only corpora used in previous studies, serves as the foundation for training our specialized LLMs: Chemlactica (125M and 1.3B parameters) and Chemma (2B parameters). These models demonstrate a deep understanding of molecular structures and properties, enabling more accurate predictions and generations.
2. We illustrate the adaptability of our models through efficient fine-tuning for various molecular property prediction tasks. With just a few hundred training examples, our models achieve

competitive performance on standard benchmarks like ESOL and FreeSolv, showcasing their potential for rapid adaptation to new tasks in drug discovery pipelines incorporating experimental data.

3. We introduce a new molecule optimization algorithm that unifies concepts from genetic algorithms, rejection sampling, and prompt optimization. This algorithm leverages our trained LLMs to navigate the vast chemical space efficiently, generating molecules with targeted properties with high sample efficiency. It achieves state-of-the-art performance on multiple molecular optimization benchmarks. On the PMO benchmark tasks (Gao et al., 2022), we achieved an average improvement of 8% over the previous best method. In drug discovery case studies involving protein-ligand docking, our method generates viable drug candidates up to 4 times faster than existing approaches.

2 Related Work

Language Models for Molecular Representation While graph-based representations are common for molecules, string-based representations, particularly the Simplified Molecular Input Line Entry System (SMILES) (Weininger, 1988), have gained new traction in molecular modeling due to their compatibility with language models(Guo et al., 2023c; Ramos et al., 2024). This approach leverages the power of pretrained language models and enables efficient processing of molecular data. Notable examples include ChemFormer (Irwin et al., 2022), MolT5 (Edwards et al., 2022), BARTSmiles (Chilingaryan et al., 2024), and LM-Desgin for proteins (Zheng et al., 2023), which adapt traditional language model architectures to chemical tasks. These models demonstrate the potential of applying natural language processing techniques to molecular design and property prediction.

Molecular Optimization Techniques Molecular optimization, a key challenge in drug discovery(Schneider and Fechner, 2005; Zhou et al., 2019), involves navigating a vast combinatorial space of potential drugs while satisfying multiple constraints. Traditional approaches include genetic algorithms adapted for molecular graphs (Yoshikawa et al., 2018) and Monte Carlo tree search over molecular graphs (Jensen, 2019). Recent methods increasingly make use of machine learning, especially deep learning techniques(van Tilborg et al., 2024). For instance, variational autoencoders (Kingma and Welling, 2013) have been applied to generate and optimize molecules in latent space, including (Gómez-Bombarelli et al., 2018) and (Jin et al., 2018). The GFlowNet (Bengio et al., 2021) represents a novel approach designed to sample compositional objects (like molecules) with reward-proportional probability, making it well-suited for optimization tasks. Extensions of GFlowNets (Kim et al., 2024) incorporating genetic search have shown promising results in molecular optimization.

Large Language Models in Optimization The success of large language models (LLMs) has led to their application in various optimization tasks beyond text generation. For instance, Chen et al. (2023) combined prompt tuning with evolutionary algorithms to design neural network architectures, outperforming human experts on specific tasks. Similarly, EvoPrompt (Guo et al., 2023b) developed a general evolutionary algorithm using language models, optimizing task-specific prompts for various downstream applications. Recently, generalized methods from discrete prompt optimization have been introduced(Guo et al., 2023a). Another work where optimization is performed via human-machine dialogue with a finetuned LM is DrugAssist (Ye et al., 2023). Our method uniquely combines discrete prompt optimization style evolutionary methods and LLM-based optimization to the molecular optimization problem setting. The work most similar and parallel to ours is (Wang et al., 2024), which also combines evolutionary algorithms with prompt tuning for molecular generation but does not train models on custom corpora or perform additional finetuning during optimization as is done in this work. These studies demonstrate the potential of LLMs in complex optimization problems, paving the way for their application in molecular design and optimization.

Our work builds upon these foundations, uniquely combining the strengths of large language models with evolutionary strategies for molecular optimization. We extend the application of LLMs beyond simple property prediction or generation, developing a comprehensive framework for navigating the complex landscape of molecular design.

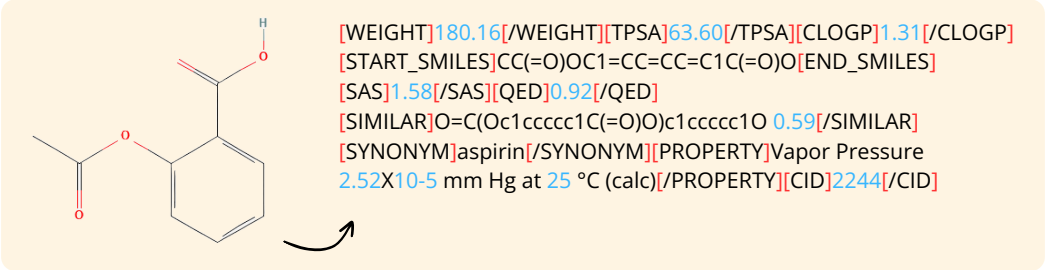


Figure 1: Aspirin's molecular structure (left) and it's representation in our dataset (right)

3 Training Corpus

For our training data, we extract information on molecules from PubChem(Kim et al., 2015), encompassing information on the molecules, similar molecule pairs, experimental properties, and bioassays; we store the data in a database. We randomly reserve approximately 10,000 molecules to monitor during training and for select experiments in 4, where the validation set is mentioned. We used *rdkit* (Landrum et al., 2013) to compute key molecular properties, get more precise similarity measurements, and standardize SMILES strings. We transformed our database into a corpus of molecular documents with key-value pairs representing identifiers and information for a given molecule. To provide this as input to the language models, we developed a template system using paired tags to delimit each property and data point for the final string representation of molecules that we derive from their intermediate key-value representations. For instance, the string representation for a molecule's quantitative estimated drug-likeness (QED) value is `[QED]0.84[/QED]`. To enable property prediction and property-conditioned molecular generation, we randomized the property order and either set the position of the primary molecule at the start of the document or in between other tags with equal probability. Figure 1 illustrates the document corresponding to aspirin. We provide more details on the training data in the appendix section A.4 and open-source the key-value representation of our data on huggingface.

4 Model Training and Evaluation

Selection of Pretrained Language Models We chose models for continued pretraining based on their general-purpose performance and domain-specific knowledge. At its release, Galactica (Taylor et al., 2022) outperformed models like OPT (Zhang et al., 2022), Chinchilla (Hoffmann et al., 2022), and BLOOM (Workshop et al., 2022) on tasks such as BIG-bench (bench authors, 2023), MMLU (Hendrycks et al., 2020), and TruthfulQA (Lin et al., 2021). Its pretraining included two million PubChem molecules, SMILES-specific tagging, and a scientific corpus, making it well-suited for molecular data. Gemma (Team et al., 2024), while not explicitly trained on molecular data, underwent extensive pretraining (2 trillion tokens for Gemma-2B) and demonstrated state-of-the-art performance on benchmarks like MMLU, HellaSwag (Zellers et al., 2019), and Human eval (Chen et al., 2021), comparable to larger models like LLaMA 2 (Touvron et al., 2023) and Mistral 7B (Jiang et al., 2023). We used the Galactica and Gemma tokenizers with minor modifications, and performed standard language model training for both models. Additional details on tokenization and training are supplied in appendix sections A.5.

4.1 Benefits of Continued Pretraining

To assess the efficacy of continued pretraining, we conducted two experiments designed to demonstrate (a) the potential advantages of initiating from a pretrained model versus training from scratch and (b) whether our continued pretraining methodology enhances molecular comprehension. For the first experiment, we randomly initialized a 125M parameter Galactica model and trained it following the protocol recommended in the Galactica paper (Taylor et al., 2022) on our entire dataset. The results of the downstream evaluations, presented in Table 1, demonstrate that the randomly initialized model yielded inferior performance on all conditional generation and property prediction tasks

Table 1: RMSE (RSME corrected for mean) ↓ for Property Prediction (PP) and Conditional Generation (CG) for different tasks and models.

	QED		SIM		SAS	
	PP	CG	PP	CG	PP	CG
Random Init-125M	0.030	0.110 (0.119)	0.068	0.239	0.092	0.442 (0.736)
Chemlactica-125M	0.016	0.084 (0.084)	0.046	0.181	0.082	0.432 (0.432)
Chemlactica-1.3B	0.004	0.069 (0.096)	**0.043**	0.172	0.064	0.281 (0.281)
Chemma-2B-2.1B	0.015	0.095 (0.095)	0.049	0.147	0.057	0.482 (0.482)
Chemma-2B-39B	**0.003**	**0.059 (0.059)**	0.045	**0.167**	**0.049**	**0.277 (0.277)**

	CLOGP		TPSA		WEIGHT	
	PP	CG	PP	CG	PP	CG
Random Init-125M	0.543	1.096 (1.096)	2.415	9.628 (10.218)	11.934	58.091 (58.091)
Chemlactica-125M	0.101	0.429 (0.482)	1.326	7.876 (8.774)	6.996	17.267 (17.267)
Chemlactica-1.3B	0.094	0.507 (0.507)	1.032	**5.579 (5.579)**	5.699	**13.494 (13.679)**
Chemma-2B-2.1B	0.055	0.700 (0.700)	1.662	6.307 (6.307)	4.187	19.565 (19.565)
Chemma-2B-39B	**0.037**	**0.454 (0.454)**	**0.933**	7.091 (7.091)	**0.640**	15.429 (15.429)

compared to Chemlactica-125M, which leveraged Galactica's pretraining. To address the second point, we performed supervised fine-tuning on a molecular property classification task using both Galactica-125M and Chemlactica-125M, demonstrating the superior capacity of our model to adapt to new downstream tasks. More details on this experiment can be found in Appendix A.7

4.2 Evaluation of Computed Property Prediction and Conditional Generation

To assess our models' proficiency in learning computed properties, we conducted two comprehensive experiments:

Property Prediction We randomly sampled a fixed set of 100 molecules from the validation set. For each property, we prompted the models with `[START_SMILES]`m_i^{smiles}`[END_SMILES][QED]`, where m_i^{smiles} represents the SMILES string of the molecule. We then calculated the Root Mean Square Error (RMSE) between predicted and actual property values to evaluate performance.

Conditional Generation For each property, we sampled 100 values v_i from the distribution of PubChem molecules. We then prompted the models to generate molecules with `[QED]`v_i`[/QED][START_SMILES]`. Using rdkit, we computed the actual property values of the generated SMILES and calculated the RMSE against the target v_i. To account for potential invalid generations, we compute a corrected RMSE by substituting the property values of invalid SMILES with the mean value of the respective property's distribution in our dataset.

Table 1 demonstrates the results of these experiments for the three models and a compute-controlled version of the Chemma-2B model (Chemma-2B-2.1B) and another controlled for the number of molecules to match the 125M model trained on the full training set(Chemma-2B-39B). We have utilized several techniques, including Chain-of-Thought (CoT)(Wei et al., 2022), repetition penalty(Keskar et al., 2019), and undesired token suppression to enhance the quality of generations. The details of these techniques, alongside an ablation study of the effect of each on generations, are included in Appendix A.6.3. Furthermore, we show that our models are well calibrated in predicting these properties in A.12.3. These experiments comprehensively demonstrate our models' capabilities in predicting molecular properties and generating molecules with specified properties. These are crucial tasks in molecular design and will become the building blocks for our optimization algorithm.

Supervised Fine-Tuning A notable capability of our models is their adaptability to new datasets and ability to learn novel molecular properties through supervised fine-tuning. To assess this feature, we fine-tuned our models on 6 ADME tasks introduced by Fang et al. (2023a) and 3 others from

Table 2: Regression tasks from MoleculeNet, all values are RMSE ↓.

	ESOL	FreeSolv	Lipophilicity	Avg
MoleculeNet GC	0.970	1.400	0.655	1.008
Chemformer	0.633	1.230	0.598	0.820
MoLFormer-XL	0.279	**0.231**	0.529	0.346
GROVER large	0.831	1.544	0.560	0.978
MolCLR	1.110	2.200	0.650	1.320
iMolCLR	1.130	2.090	0.640	1.287
BARTSmiles	0.308	0.338	0.540	0.395
Chemlactica-125M	0.276 ± 0.027	0.312 ± 0.016	0.486 ± 0.003	0.358 ± 0.012
Chemlactica-1.3B	$\mathbf{0.251} \pm \mathbf{0.004}$	0.286 ± 0.009	$\mathbf{0.463} \pm \mathbf{0.006}$	$\mathbf{0.333} \pm \mathbf{0.005}$
Chemma-2B	0.297 ± 0.018	0.368 ± 0.010	0.531 ± 0.014	0.404 ± 0.020

MoleculeNet Wu et al. (2018). These tasks require the model to learn and predict different molecular properties, such as hydration-free energy, water solubility, and human liver microsomal stability. Our models demonstrate competitive performance, achieving state-of-the-art results for some tasks, as shown in Table 2. Unlike some comparable methods Sirumalla et al. (2024), Glenn Northcutt (2005) and Chilingaryan et al. (2024), we train and evaluate the model on regression tasks using next token prediction, without utilizing a dedicated regression head. We provide further details regarding data structuring, parameter choices, and results for ADME datasets in Appendix A.12.2.

5 Molecular Optimization Algorithm

We present a novel population-based algorithm for molecular optimization that leverages our trained language models. The algorithm addresses the challenging task of navigating the vast chemical space to find molecules with desired properties, subject to a limited evaluation budget. Formally, we define the molecular optimization problem as:

$$m^* = \arg\max_{m \in \mathcal{M}} O(m)$$

where m represents a molecule, $\mathcal{M}$ is the set of valid candidate molecules (estimated to be around 10^6 (Bohacek et al., 1996)), and $O : \mathcal{M} \to \mathbb{R}$ is a black-box oracle function that evaluates molecular properties. This oracle could represent complex processes such as lab experiments or quantum simulations.

Our approach maintains a pool of P high-performing molecules and iteratively generates new candidates using a language model. It is built on three key innovations:

LLM-enhanced genetic algorithm We leverage our language models to generate molecules similar to the current pool. This process functions analogously to a genetic algorithm where language model generations replace traditional crossover/mutation operations. For S randomly selected molecules from the pool, we generate a new molecule using the prompt:

[SIMILAR]m_1^{smiles} 0.9[/SIMILAR]...[SIMILAR]m_S^{smiles} 0.8[/SIMILAR][START_SMILES]

Explicit oracle modeling Inspired by the rejection sampling technique (Bai et al., 2022; Touvron et al., 2023), we incorporate oracle feedback directly into the language model by fine-tuning on high-performing molecules. To accomplish this, we use prompts of the form:

[PROPERTY]$O(m)$[/PROPERTY][START_SMILES]m^{smiles}[END_SMILES]

This explicit modeling allows the language model to learn the relationship between molecular structure and oracle scores, enabling more targeted generation.

Algorithm 1 presents our complete optimization procedure, which includes the initialization of an empty molecule pool, iterative generation of new molecules using the language model, evaluation of new molecules using the oracle function, updating the pool to maintain the top-P molecules, and periodic fine-tuning of the

Algorithm 1 molecular_optimization

Input: P, S, N, K
Initialize an empty $Pool \leftarrow \{\}$
repeat
1. Generate prompts for molecule generation.
for $i = 1$ **to** N **do**
$(m_{i,1}, m_{i,2}, \ldots, m_{i,S}) \leftarrow random_subset(Pool)$
$p_i \leftarrow molecules2prompt((m_{i,1}, m_{i,2}, \ldots, m_{i,S}), null)$
end for

2. Generate N new and unique molecules with the language model.
$m_i \leftarrow LM(p_i), i = 1, \ldots, N$

3. Update the pool with m_is and keep only the top-P molecules.
$Pool \leftarrow Pool \cup \{m_1, \ldots, m_N\}$
$Pool \leftarrow$ top-$P(Pool)$

4. Fine-tune if necessary.
if the best molecule (in terms of oracle score) has not improved for K iterations **then**

5. Take all the molecules from the $Pool$ with their corresponding similar molecules (using which they have been generated), $m_i, (m_{i,1}, m_{i,2}, \ldots, m_{i,S}), i = 1, \ldots, P$ respectively.
$train_samples_i \leftarrow molecules2prompt((m_{i,1}, m_{i,2}, \ldots, m_{i,S}), m_i), i = 1, \ldots, P$

6. Train LM on $train_samples_i, i = 1, \ldots, P$.
end if
until optim. problem stopping condition

language model when progress stagnates. Algorithm 2 details our prompt construction process, which is crucial for effective molecule generation and model fine-tuning. For generation, vanilla temperature sampling is used.

We employ a dynamic fine-tuning strategy to adapt the language model throughout the optimization process. Fine-tuning is triggered if the best molecule does not improve for K consecutive iterations, with the maximum number of fine-tuning rounds limited by the oracle budget. We use a learning rate scheduler with warm-up steps, and each fine-tuning step consists of multiple epochs with a portion of data reserved for validation to prevent overfitting.

Given the complexity of our algorithm, we adopt a focused hyperparameter tuning strategy, prioritizing the most sensitive parameters while keeping others fixed. This approach balances computational efficiency with optimization performance. Appendix A.6 provides the methodology and results of our hyperparameter tuning experiments. By combining these elements, our algorithm effectively leverages the power of large language models for molecular optimization, showing strong performance across a range of tasks as detailed in Section 6.

6 Experiments

6.1 Practical Molecular Optimization

Problem formulation. Inspired by real-world molecular design settings Gao et al. (2022) proposes the practical molecular optimization (PMO) benchmark consisting of 23 molecular optimization problems. PMO focuses on sample efficiency, generalizability to different optimization objectives, and robustness to hyperparameter selection of molecular optimization algorithms. To assess optimization ability and sample efficiency, Gao et al. (2022) put a limit on the number of oracle calls for each task to 10000 and measures the area under the curve (AUC) of the top-10 average property values versus the number of oracle calls as the performance metric. AUC values are calculated after every 100 oracle calls, combined, and normalized to the $[0, 1]$ range.

Our approach. Using our proposed optimization algorithm we evaluate the Chemlactica-125M, Chemlactica-1.3B and Chemma-2B models. The optimization algorithm's hyperparameters are tuned for each model separately according to the hyperparameter tuning methodology described in (Gao et al., 2022) and A.6. For this experiment, we keep model parameters in bfloat16 for more rapid evaluation.

Table 3: PMO benchmark with Chemlactica-125M, Chemlactica-1.3B and Chemma-2B in comparison with other methods. REINVENT results are taken from Gao et al. (2022), Augmented memory is taken from Guo and Schwaller (2023a), and Genetic-guided (GG) GFlowNets are taken from Kim et al. (2024). Values are the average of 5 runs with different seeds, metric is Top-10 AUC ↑ ± standard deviation

	jnk3	median1	scaffold_hop	sitagliptin_mpo	sum of 4	sum of 23
REINVENT	0.783 ± 0.023	0.356 ± 0.009	0.560 ± 0.019	0.021 ± 0.003	1.720	14.196
Augmented memory	0.739 ± 0.110	0.326 ± 0.013	0.567 ± 0.008	0.284 ± 0.050	1.916	15.002
GG GFlowNets	0.764 ± 0.069	0.379 ± 0.010	0.615 ± 0.100	0.634 ± 0.039	2.392	16.213
Chemlactica-125M	0.881 ± 0.058	0.359 ± 0.060	0.626 ± 0.016	$\mathbf{0.649 \pm 0.051}$	2.515 ± 0.119	17.170 ± 0.424
Chemlactica-1.3B	0.866 ± 0.021	$\mathbf{0.382 \pm 0.047}$	$\mathbf{0.673 \pm 0.080}$	0.586 ± 0.062	2.506 ± 0.155	17.284 ± 0.284
Chemma-2B	$\mathbf{0.891 \pm 0.032}$	$\mathbf{0.382 \pm 0.022}$	0.669 ± 0.110	0.613 ± 0.018	$\mathbf{2.555 \pm 0.099}$	$\mathbf{17.534 \pm 0.214}$

Results. Our method performs strongly, surpassing the existing approaches. The algorithm powered by our smallest model (Chemlactica-125M) already improves over the state-of-the-art by a significant margin, with an AUC Top-10 of 17.170 (Chemlactica-125M) vs 16.213 (Genetic-guided GFlowNets). Additionally, strengthening the generator model improves the performance. Chemlactica-1.3B and Chemma-2B achieve AUC Top-10 of 17.284 and 17.534, respectively. For a more comprehensive understanding of optimization dynamics, Figures 5-7 illustrate visualizations of the optimization processes for `sitagliptin_mpo` task with different seeds and different models.

Furthermore, to investigate the novelty of generated molecules, we visualize the distance of the molecules generated by the model from the closest molecule in the training dataset (PubChem) throughout the optimization procedure. Figure 2 represents the molecules generated during the optimization along with their oracle scores and distances from the closest molecule in PubChem computed with Tanimoto similarity. Even though we did not explicitly guide the model to generate molecules distant from PubChem, we observe that the model generates molecules far from PubChem to optimize the given objective. In both multi-property optimization (MPO) tasks across all seeds, the model finds high-scoring molecules with less than 0.3 similarity to their nearest neighbor in PubChem, and we observe a similar pattern in nearly all MPO problems from PMO. We conclude that our method does not retrieve molecules from the training dataset and is able to explore the chemical space beyond PubChem to solve molecular design tasks.

Unlike most other methods, our language models can leverage additional information about the oracle if the oracle internally calculates common molecular properties. These properties can be explicitly written in the prompts used in the optimization loop. In Appendix A.11.2, we show that such enriched prompts can significantly improve the metrics for several PMO tasks.

6.2 Multi-property Optimization with Docking

6.2.1 Introduction to MPO with Docking

Molecular optimization tasks incorporating docking simulation evaluate a model's capability to generate viable molecules for practical drug discovery. These benchmarks assess the model's ability to generate plausible molecules that optimize docking scores (minimize docking energy) against specified protein targets while adhering to other desired structural or physicochemical constraints. The additional constraints help to prevent exploitation of the docking algorithms used. The primary objective of these benchmarks is maximizing the reward function with minimal oracle calls, emphasizing sample efficiency. Below, we present our approach and results on two benchmarks involving suites of MPO tasks with docking components, using different hyperparameter tuning approaches and evaluation metrics. We illustrate how generated molecules' docking scores change throughout the optimization process in A.13.2. In other experiments, we found that numerical precision is important for molecular optimization tasks (see A.11.3), so we keep model parameters in full floating-point precision for docking MPO experiments.

6.2.2 Docking MPO on DRD2, AChE, and MK2

Problem formulation. This benchmark was initially proposed in the Augmented Memory paper (Guo and Schwaller, 2023a). It focuses on three targets with extensive real-world applications: the dopamine type 2 receptor (DRD2), MK2-kinase, and acetylcholinesterase (AChE). To ensure the generation of realistic molecules, the oracle reward function incorporates additional constraints, including the maximization of QED and a molecular weight limit of 500 Da.

Consistent with other works, we quantify sample efficiency using two metrics: oracle burden and generative yield. Oracle burden measures the number of oracle calls required to generate N unique molecules above a

Table 4: Docking MPO experiments run with a maximum oracle budget of 5000 oracle calls. Note that both oracle burden and generative yield values are reward-threshold dependent, and mean values from the reported baseline works are reported. The numbers next to the metrics correspond to the thresholds, and the values in parentheses for oracle burden indicate how many unique molecules need to be generated. The best performance on each task-metric combination is bolded. We use the best-performing hyperparameters from the PMO benchmark.

Metric	Target	Reinvent Baseline	Beam Structure 15	Chemlactica 125M	Chemlactica 1.3B	Chemma 2B
Generative Yield 0.7 ↑	DRD2	1879 ± 16	3474 ± 158	3733 ± 512	3659 ± 288	**3848** ± 98
	MK2	879 ± 10	3127 ± 138	**3772** ± 578	3660 ± 535	3578 ± 452
	AChE	2437 ± 53	3824 ± 162	4108 ± 67	**4193** ± 128	4092 ± 284
Generative Yield 0.8 ↑	DRD2	102 ± 6	1780 ± 439	2827 ± 510	2621 ± 614	**2985** ± 194
	MK2	2 ± 0	987 ± 211	**2569** ± 1156	2216 ± 522	1058 ± 465
	AChE	147 ± 11	2059 ± 327	3246 ± 168	**3652** ± 349	3096 ± 372
Oracle burden 0.8 (1) ↓	DRD2	168 ± 149	126 ± 90	20 ± 29	**11** ± 10	74 ± 62
	MK2	1724 ± 802	736 ± 166	345 ± 312	**78** ± 125	189 ± 278
	AChE	83 ± 29	105 ± 29	22 ± 28	**15** ± 23	74 ± 72
Oracle burden 0.8 (10) ↓	DRD2	883 ± 105	582 ± 83	**114** ± 08	160 ± 130	240 ± 11
	MK2	Failed	1122 ± 154	493 ± 418	**248** ± 261	440 ± 548
	AChE	481 ± 108	462	224 ± 17	**91** ± 103	168 ± 94
Oracle burden 0.8 (100) ↓	DRD2	4595 ± 0	1120 ± 25	**364** ± 119	430 ± 250	518 ± 41
	MK2	Failed	2189 ± 181	865 ± 533	**486** ± 346	934 ± 918
	AChE	3931 ± 286	1110 ± 265	497 ± 58	**333** ± 131	433 ± 143

predefined reward threshold, and generative yield represents the number of unique molecules generated above a reward threshold for a fixed number of oracle calls. To maintain consistency in implementations, we adopt the molecular preprocessing, conformational generation, docking parameters, and aggregate reward function from the (Guo and Schwaller, 2023b), specifically comparing our results with the beam structure 15 method, which demonstrated superior average-case performance relative to other benchmarked methods. We used the same hyperparameters as those selected for the PMO experiment with no modifications.

Results. Table 4 presents our approach's performance on this benchmark. None of our models consistently outperforms the others for generative yield across the evaluated receptors. Conversely, Chemlactica-1.3B generally demonstrates superior performance for oracle burden, aside from oracle burden 1 and 10 for DRD2, where Chemlactica-125M is superior. Appendix A.13.3 shows the set of molecules generated at the beginning and at the end of the optimization trajectory for DRD2 docking. Furthermore, in A.13.2, we show that the average docking score of molecules consistently decreases throughout optimization, suggesting that the model learns to not only generate molecules that improve the scores of the easier properties but can generate molecules with low docking energies for the DRD2 pocket as well.

6.2.3 DOCKING MPO; HITS ON PARP1, FA7, 5HT1B, BRAF, AND JAK2

Problem formulation. Following Lee et al. (2024), we formulate the MPO objective as the product of a normalized docking score, normalized SA score, and QED score. The docking is performed on the `parp1`, `fa7`, `5ht1b`, `braf`, and `jak2` target proteins.

Our approach. To ensure comparability of the results with other methods, we employ the oracle function implementation used in Guo and Schwaller (2024), run our method with ten different seeds (0-9), and use an oracle budget of 3000 for each task. For hyperparameter tuning, we use an illustrative experiment that does not use the same oracle function as the docking MPO tasks (for a more detailed discussion on hyperparameter selection, refer to the Appendix A.6. Following Lee et al. (2024) and Guo and Schwaller (2024), we compute the Hit Ratio (%), that is, the percentage of molecules with QED > 0.5, SA score < 5, and docking score better than the median of the known actives, as well as the Strict Hit Ratio (%), which requires QED > 0.7 and SA score < 3.).

Results. Table 5 and 6 illustrate the results of our method in comparison with the baselines. We observe that our algorithm powered by our language models performs comparably with others. In terms of the Hit Ratio (%), our approach performs significantly better with the targets `fa7` and `jak2`, while in terms of the Strict Hit Ratio (%), it is significantly better with the target `fa7`. Notably, for the target `5ht1b`, our approach is inferior to Saturn for both metrics and shows statistically similar performance with the rest of the target proteins. The

outcomes illustrate the efficiency of our approach and its transferability to physics-based simulations using the illustrative experiment. The results also demonstrate the applicability of our method to MPO problems with a costly oracle, which are more prevalent in industrial drug discovery settings. We present analyses of generated molecule diversity in A.9.

Table 5: Comparision of our approach with other methods. The values represent the Hit Ratio (%) $\uparrow \pm$ standard deviation across 10 independent runs. The results of Augmented Memory, GEAM, and Saturn are taken from Guo and Schwaller (2023a), Lee et al. (2024), and Guo and Schwaller (2024), respectively. Results within one standard deviation from the best one are bolded.

Method	Target Protein				
	parp1	fa7	5ht1b	braf	jak2
Augmented Memory	16.966 ± 3.224	2.637 ± 0.860	52.016 ± 2.302	8.307 ± 1.714	21.548 ± 4.938
GEAM	**45.158 ± 2.408**	20.552 ± 2.357	47.664 ± 1.198	30.444 ± 1.610	46.129 ± 2.073
Saturn	**57.981 ± 18.537**	14.527 ± 9.961	**68.185 ± 3.400**	**38.999 ± 10.114**	60.827 ± 11.502
Chemlactica-125M	37.117 ± 7.325	70.392 ± 17.584	31.927 ± 2.697	30.107 ± 8.071	54.657 ± 12.759
Chemlactica-1.3B	**52.333 ± 12.669**	**87.03 ± 5.37**	40.541 ± 3.373	**49.471 ± 12.055**	**75.02 ± 8.04**
Chemma-2B	**52.063 ± 6.935**	79.97 ± 11.154	38.233 ± 5.846	**44.81 ± 12.14**	**71.03 ± 9.074**

Table 6: Comparison of our approach with other methods. The values represent the Strict Hit Ratio (%) $\uparrow \pm$ standard deviation across 10 independent runs. The results of GEAM and Saturn are taken from Lee et al. (2024) and Guo and Schwaller (2024), respectively. Results within one standard deviation from the best one are bolded.

Method	Target Protein				
	parp1	fa7	5ht1b	braf	jak2
GEAM	6.510 ± 1.087	2.106 ± 0.958	8.719 ± 0.903	3.685 ± 0.524	7.944 ± 1.157
Saturn	**55.102 ± 18.027**	13.887 ± 9.723	**64.730 ± 3.717**	**37.250 ± 9.615**	**55.903 ± 13.613**
Chemlactica-125M	28.907 ± 7.185	58.596 ± 18.698	25.523 ± 2.865	22.86 ± 7.239	45.943 ± 12.408
Chemlactica-1.3B	35.33 ± 12.837	**68.22 ± 6.438**	28.074 ± 3.641	**34.583 ± 9.856**	**58.133 ± 8.404**
Chemma-2B	**37.117 ± 5.712**	**63.933 ± 11.794**	26.757 ± 5.345	**31.783 ± 10.171**	**54.79 ± 8.946**

6.2.4 Discussion of MPO with Docking

Our findings validate the effectiveness of our approach, demonstrating that our models can leverage pretraining information and iterative fine-tuning to optimize complex reward functions, even with limited data not seen during pretraining. Furthermore, successfully transferring training parameters and sampling strategies from the PMO benchmark and illustrative experiment to the MPO docking tasks in 6.2.2 and 6.2.3, respectively, underscores our method's flexibility and robustness. This adaptability suggests that our approach could be particularly valuable when extensive hyperparameter tuning is impractical or undesirable. Specifically, the results demonstrate the applicability of our method to MPO problems with costly oracles, which are more prevalent in industrial drug discovery settings. However, our method does not directly generate 3D conformations used by the docking scoring function, has not universally outperformed the baselines across tasks, and is sensitive to numerical precision. Future work will further improve sample efficiency and apply the algorithm to even more challenging MPO problems inspired by applications within the industry.

7 Conclusion

This paper presents three language models: Chemlactica-125M, Chemlactica-1.3B, and Chemma-2B, trained on a novel corpus encompassing over 100 million molecules and their properties. We demonstrate the efficacy of these models on multiple tasks inspired by industrial drug design, with a particular focus on molecular optimization. Our proposed optimization algorithm combines the capabilities of language models with concepts from genetic algorithms. This approach has shown strong performance across various benchmarks, indicating its potential for addressing complex molecular design challenges. We publicly release our training corpus, pretrained models, optimization algorithm, and associated training recipes to support reproducibility, making an early step toward applying language models to chemical research. We hope our contributions provide a valuable foundation for future work in this domain, enabling new approaches for molecular design and analysis.

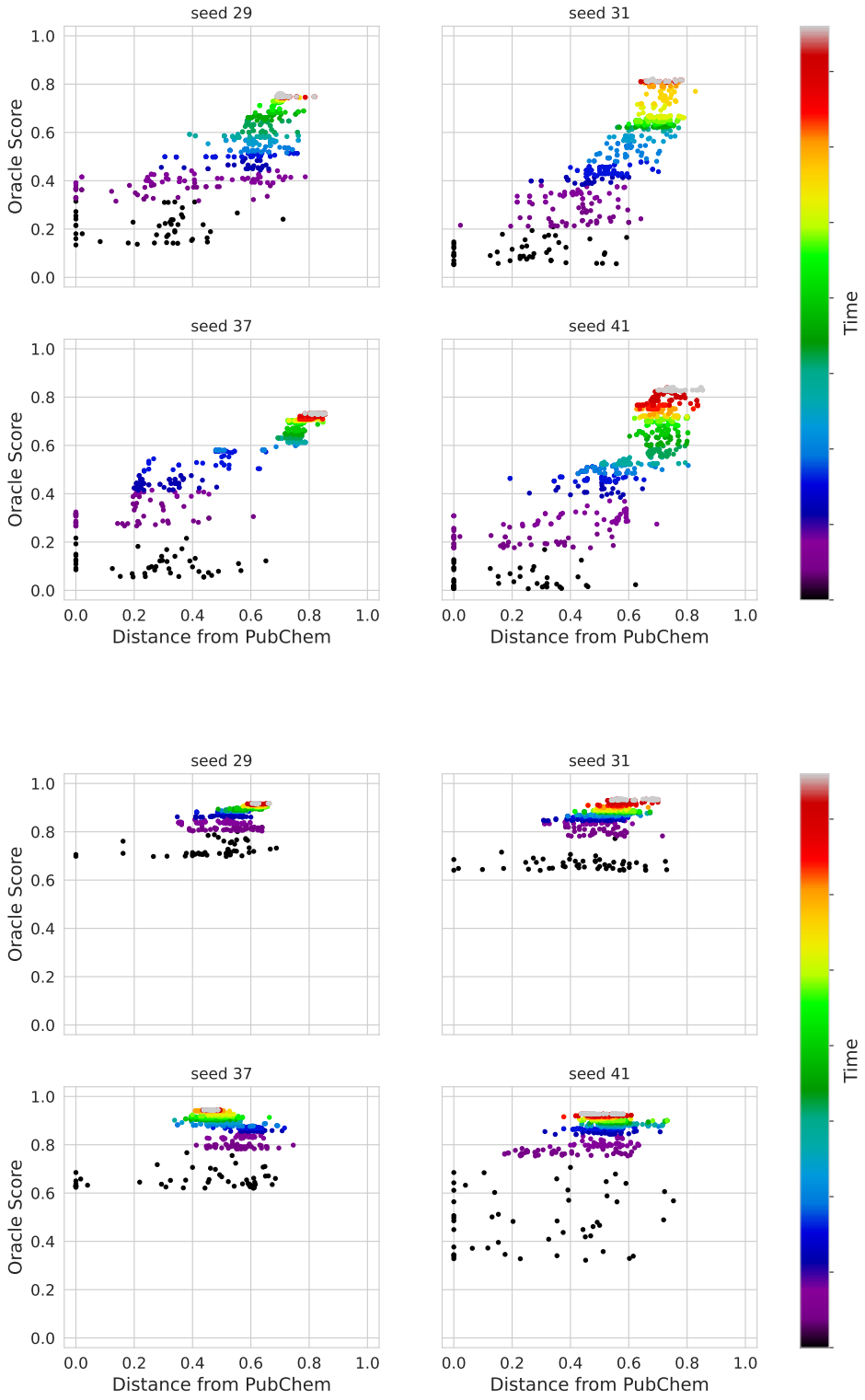


Figure 2: Visualization of the oracle score vs. distance from the closest molecule in PubChem of the current 50 best molecules throughout the optimization process. The plots are obtained for `sitagliptin_mpo` (top) and `ranolazine_mpo` (bottom) tasks from PMO benchmark with the Chemma-2B model with four different seeds.

REFERENCES

Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

B. bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

R. S. Bohacek, C. McMartin, and W. C. Guida. The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews*, 16(1):3–50, 1996.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

A. Chen, D. Dohan, and D. R. So. Evoprompting: Language models for code-level neural architecture search. *ArXiv*, abs/2302.14838, 2023. URL https://api.semanticscholar.org/CorpusID:257232765.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

G. Chilingaryan, H. Tamoyan, A. Tevosyan, N. Babayan, K. Hambardzumyan, Z. Navoyan, A. Aghajanyan, H. Khachatrian, and L. Khondkaryan. Bartsmiles: Generative masked language models for molecular representations. *Journal of Chemical Information and Modeling*, 2024. URL https://doi.org/10.1021/acs.jcim.4c00512.

T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

C. N. Edwards, T. Lai, K. Ros, G. Honke, and H. Ji. Translation between molecules and natural language. *ArXiv*, abs/2204.11817, 2022. URL https://api.semanticscholar.org/CorpusID:248376906.

C. Fang, Y. Wang, R. Grater, S. Kapadnis, C. Black, P. Trapa, and S. Sciabola. Prospective validation of machine learning algorithms for absorption, distribution, metabolism, and excretion prediction: An industrial perspective. *Journal of Chemical Information and Modeling*, 63(11):3263–3274, 2023a.

C. Fang, Y. Wang, R. Grater, S. Kapadnis, C. Black, P. Trapa, and S. Sciabola. Prospective validation of machine learning algorithms for absorption, distribution, metabolism, and excretion prediction: An industrial perspective. *Journal of Chemical Information and Modeling*, 63(11):3263–3274, 2023b.

W. Gao, T. Fu, J. Sun, and C. W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization. *ArXiv*, abs/2206.12411, 2022. URL https://api.semanticscholar.org/CorpusID:250072218.

R. Glenn Northcutt. The new head hypothesis revisited. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, 304(4):274–297, 2005.

R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

J. Guo and P. Schwaller. Augmented memory: Capitalizing on experience replay to accelerate de novo molecular design. *ArXiv*, abs/2305.16160, 2023a.

J. Guo and P. Schwaller. Beam enumeration: Probabilistic explainability for sample efficient self-conditioned molecular design. *ArXiv*, abs/2309.13957, 2023b.

J. Guo and P. Schwaller. Saturn: Sample-efficient generative molecular design using memory manipulation. *arXiv preprint arXiv:2405.17066*, 2024.

Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*, 2023a.

Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, Y. Yang, T. University, and M. Research. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *ArXiv*, abs/2309.08532, 2023b. URL https://api.semanticscholar.org/CorpusID:262012566.

T. Guo, B. Nan, Z. Liang, Z. Guo, N. Chawla, O. Wiest, X. Zhang, et al. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688, 2023c.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

R. Irwin, S. Dimitriadis, J. He, and E. J. Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

N. Jain, P.-y. Chiang, Y. Wen, J. Kirchenbauer, H.-M. Chu, G. Somepalli, B. R. Bartoldson, B. Kailkhura, A. Schwarzschild, A. Saha, et al. Neftune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*, 2023.

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

A. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. Singh Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. arxiv. *arXiv preprint arXiv.2310.06825*, 2023.

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

N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

H.-S. Kim, M. Kim, S. Choi, and J. Park. Genetic-guided gflownets: Advancing in practical molecular optimization benchmark. *ArXiv*, abs/2402.05961, 2024.

S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, and S. H. Bryant. Pubchem substance and compound databases. *Nucleic Acids Research*, 44: D1202 – D1213, 2015. URL https://api.semanticscholar.org/CorpusID:9567253.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL https://api.semanticscholar.org/CorpusID:216078090.

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

S. Lee, S. Lee, K. Kawaguchi, and S. J. Hwang. Drug discovery with dynamic goal-aware fragments. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=xuX2rDSSco.

S. Lin, J. Hilton, and O. Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

X. Liu, S. Jiang, A. Vasan, A. Brace, O. Gokdemir, T. Brettin, and R. Stevens. Drugimprover: Utilizing reinforcement learning for multi-objective alignment in drug optimization. In *NeurIPS 2023 Workshop on New Frontiers of AI for Drug Discovery and Development*, 2023.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. *CoRR*, abs/1710.03740, 2017. URL http://arxiv.org/abs/1710.03740.

M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9, 2017. URL https://api.semanticscholar.org/CorpusID:2978311.

OpenAI. Gpt-4 technical report. 2023.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

M. Popova, O. Isayev, and A. Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

M. C. Ramos, C. J. Collison, and A. D. White. A review of large language models and autonomous agents in chemistry. *arXiv preprint arXiv:2407.01603*, 2024.

B. Ramsundar, P. Eastman, P. Walters, V. Pande, K. Leswing, and Z. Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh, and P. Das. Do large scale molecular language representations capture important structural information? *arXiv preprint arXiv:2106.09553*, 2021.

G. Schneider. Automating drug discovery. *Nature reviews drug discovery*, 17(2):97–113, 2018.

G. Schneider and U. Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005.

S. K. Sirumalla, D. S. Farina Jr, Z. Qiao, D. A. Di Cesare, F. C. Farias, M. B. O'Connor, P. J. Bygrave, F. Ding, T. Dresselhaus, M. G. P. de Lacerda, et al. Multi-modal and multi-task transformer for small molecule drug discovery. In *ICML'24 Workshop ML for Life and Material Science: From Theory to Industry Applications*, 2024.

R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

D. van Tilborg, H. Brinkmann, E. Criscuolo, L. Rossen, R. Özçelik, and F. Grisoni. Deep learning for low-data drug discovery: hurdles and opportunities. *Current Opinion in Structural Biology*, 86:102818, 2024.

A. Vardanian. USearch by Unum Cloud, Oct. 2023. URL https://github.com/unum-cloud/usearch.

H. Wang, M. Skreta, C. T. Ser, W. Gao, L. Kong, F. Streith-Kalthoff, C. Duan, Y. Zhuang, Y. Yu, Y. Zhu, Y. Du, A. Aspuru-Guzik, K. Neklyudov, and C. Zhang. Efficient evolutionary search over chemical space with large language models. *ArXiv*, abs/2406.16976, 2024. URL https://api.semanticscholar.org/CorpusID:270711201.

Z. Wang, W. Nie, Z. Qiao, C. Xiao, R. Baraniuk, and A. Anandkumar. Retrieval-based controllable molecule generation. *International Conference on Learning Representations*, 2023.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. URL https://arxiv.org/abs/2201.11903.

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

C. Wognum, L. Zhu, H. Mary, Larissa, J. St-Laurent, A. Quirke, K. Zhu, S. Whitfield, K. McLean, and felix. polaris-hub/polaris: 0.8.4, Sept. 2024. URL https://doi.org/10.5281/zenodo.13652588.

B. Workshop, T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Y. Xie, Z. Xu, J. Ma, and Q. Mei. How much space has been explored? measuring the chemical space covered by databases and machine-generated molecules, 2023. URL https://arxiv.org/abs/2112.12542.

G. Ye, X. Cai, H. Lai, X. Wang, J. Huang, L. Wang, W. Liu, and X. Zeng. Drugassist: A large language model for molecule optimization. *arXiv preprint arXiv:2401.10334*, 2023.

N. Yoshikawa, K. Terayama, M. Sumita, T. Homma, K. Oono, and K. Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47(11):1431–1434, 2018.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

S. Zhang, Z. Chen, Y. Shen, M. Ding, J. B. Tenenbaum, and C. Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023.

Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, aug 2023. ISSN 2150-8097. doi: 10.14778/3611540.3611569. URL https://doi.org/10.14778/3611540.3611569.

Z. Zheng, Y. Deng, D. Xue, Y. Zhou, F. Ye, and Q. Gu. Structure-informed language models are protein designers. In *International conference on machine learning*, pages 42317–42338. PMLR, 2023.

C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, et al. Lima: Less is more for alignment. corr abs/2305.11206 (2023). *arXiv preprint arXiv:2305.11206*, 10, 2023.

Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):10752, 2019.

A Appendix

A.1 Broader Related Work

Recurrent Neural Networks in Molecular Design Recurrent neural networks (RNNs) have been applied to molecular optimization as parameter-efficient alternatives to transformer architectures. A notable example is REINVENT (Olivecrona et al., 2017), which uses policy-based reinforcement learning to generate molecules with desired properties. Another attempt at employing RNNs in drug design is ReLeaSE (Popova et al., 2018), which combines multi-phase training with small RNNs for molecular drug design. Another work that proposes a framework for drug optimization and utilizes finetuned RNNs is DrugImprover (Liu et al., 2023). Finally, recent enhancements to REINVENT, such as augmented memory and beam enumeration (Guo and Schwaller, 2023b), have further improved its performance. These approaches combine surrogate models, molecular diversity filters, experience replay mechanisms, and substructure filtering to increase sample efficiency; but these methods do not leverage the scale, generative capabilities and flexibility afforded by LLMs.

A.2 Limitations

The language models introduced in this paper operate only on SMILES representations and do not support 3D coordinates of atoms, limiting their reliability in scenarios where 3D conformation is critical. Furthermore, the models have a very limited understanding of biological entities like proteins, constraining their practical applicability in particular biochemistry and drug discovery use-cases. While effective, the optimization algorithms presented in this paper are not exhaustively tuned, suggesting potential room for improvement. Finally, our current approach does not directly account for or consider synthetic accessibility or other practical considerations in drug design, which limits its immediate applicability in real-world drug discovery pipelines.

A.3 BROADER IMPACT

The methods presented in this work have the potential for both positive and negative societal impacts. On the positive side, these models could significantly benefit the drug discovery and healthcare industries by accelerating the development of new therapeutic compounds. This acceleration may lead to faster responses to emerging health challenges and potentially reduce the cost of drug development.

However, as with many dual-use technologies, there is a risk that sufficiently advanced versions of these models could lower the barriers for malicious actors attempting to develop chemical or biological weapons. This risk underscores the importance of responsible development and deployment of such technologies.

Given these potential impacts, we recommend that future work in this area include rigorous evaluation of these algorithms and language models in designing potentially harmful substances to better understand and mitigate risks. Developing safeguards and ethical guidelines for using and disseminating molecular optimization models is crucial. Collaboration with experts in biosecurity and ethics will be essential to ensure that the development of these technologies proceeds in a manner that maximizes benefits while minimizing the potential for harm.

A.4 DATASET GENERATION

Data Collection We first constructed a comprehensive SQL database using PubChem dumps to generate our training corpus. Then, using *rdkit* (Landrum et al., 2013), we computed key molecular properties, including synthesizability score (SAS), quantitatively estimated drug-likeness (QED), molecular weight (MW), total polar surface area (TPSA), partition coefficient (CLogP), and various structural features such as hydrogen donors/acceptors and ring counts. Due to differences in SMILES canonicalization between PubChem and rdkit, we standardized all SMILES strings using rdkit's implementation.

Our dataset's cutoff date is January 26th, 2023, thus excluding any subsequent additions or modifications to PubChem. To ensure data integrity, molecules that failed rdkit's MolFromSmiles parsing were discarded. To incorporate similarity information, we utilized PubChem's related molecule data, which includes pairs with Tanimoto similarity $\geq$0.8 based on PubChem fingerprints. From the resulting 200 billion pairs, we sampled 4 billion and recalculated their similarities using the ECFC4 fingerprint for improved accuracy and consistency with other methods.

JSONL Corpus Generation We transformed our database into a corpus of JSONL files, with each molecule represented as a single JSON object. This representation includes molecular identifiers, computed properties, similarity data, synonyms, experimental properties, and the PubChem compound identifier (CID). This representation allows for more flexible manipulation of molecules' text representation, which we describe in 3.

A.5 TRAINING DETAILS

A.5.1 TOKENIZATION

We utilized the original tokenizers from Gemma and Galactica, adding chemistry-specific tokens `[START_SMILES]` and `[END_SMILES]` to Gemma's tokenizer for consistency. To optimize training efficiency, we included all opening and closing tags as special tokens (e.g., `[QED]`). Samples of varying lengths were tokenized and grouped into blocks of 2048 tokens, separated by model-specific separator tokens (EOS "</s>" for Chemlactica, BOS "<bos>" for Chemma).

A.5.2 TRAINING IMPLEMENTATION

Chemma and Chemlactica were trained using the AdamW optimizer (Loshchilov and Hutter, 2019) with cross-entropy loss and a causal language modeling objective. We applied dropout only to Chemlactica, maintaining consistency with the original model training regimes. For computational efficiency, we train Chemma-2B in full bfloat16. We leveraged PyTorch's (Paszke et al., 2019) Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023) and Flash Attention (Dao, 2024) for optimized training. The training was conducted locally (Chemlactica-125M: 306 A100 hours) and on Nebius.ai cloud (Chemma-2B: 488 H100 GPU hours, Chemlactica-1.3B: 288 H100 GPU hours). Preparatory work before the final training runs consumed multiple thousands of A100 hours.

A.5.3 MODEL TRAINING HYPERPARAMETERS

Table 7 lists the hyperparameters we used for pretraining the aforementioned models.

A.6 HYPERPARAMETER TUNING AND SELECTION

A.6.1 OPTIMIZATION ALGORITHM HYPERPARAMETER TUNING FOR PMO.

Given our optimization algorithm's large number of hyperparameters, we adopt a two-phase approach. First, we identify and freeze the hyperparameters that empirically show less sensitivity to the algorithm's performance. Then, we focus on tuning the more sensitive hyperparameters using grid search. We tune the hyperparameters separately for Chemlactica-125M, Chemlactica-1.3B, and Chemma-2B to account for model-specific optimal settings. For searching the nearest neighbor with Tanimoto similarity in PubChem, we utilized the USearch similarity search engine for vectors (Vardanian, 2023).

Selection criteria. For tuning, we utilize the `perindopril_mpo` and `zaleplon_mpo` tasks from the PMO benchmark, following the methodology in (Gao et al., 2022). We report the AUC Top-10 metric from three independent runs with different seeds for each hyperparameter configuration.

Fixed hyperparameters and grid. A key hyperparameter, N, which determines the number of molecules generated before updating the pool, is set to 200. We implement a dynamic temperature scheduling strategy to increase the diversity of generated molecules. The sampling temperature starts at 1 and linearly increases to 1.5 as the number of oracle evaluations grows. This gradual temperature increase promotes the generation of more diverse molecules over time, reducing repetition and encouraging exploration of the chemical space.

We perform grid search on P (pool size), S (number of similar molecules), K (fine-tuning tolerance level), and lr (fine-tuning peak learning rate) with the following grid:

- $P = [10, 30, 50]$
- $S = [0, 1, 2, 5]$
- $K = [3, 5, 7]$
- $lr = [10^{-4}, 10^{-5}]$

A.6.2 OPTIMIZATION ALGORITHM HYPERPARAMETER TUNING FOR MPO WITH DOCKING ON PARP1, FA7, 5HY1B, BRAF, AND JAK2 TARGETS.

Consistent with the hyperparameter tuning used for PMO, we select the most sensitive hyper-parameters and then tune them separately for each model via grid search.

Selection criteria. Motivated by the illustrative experiment for hyperparameter tuning used in Guo and Schwaller (2023b), we formulate a problem of maximizing the TPSA of a molecule while keeping its weight under 350 Da and having more than one ring. We use this molecular design task and the oracle burden metric for hyperparameter selection.

Hyperparameter grid. Since the illustrative experiment takes less time to evaluate (compared to the PMO tasks), we expand the number of hyperparameters used in the grid to allow for a more extensive search.

We perform a grid search on P (pool size), S (number of similar molecules), N (the number of molecules generated to update the pool), $temp_schedule$ (the starting and ending temperature for linearly changing it

Table 7: Hyperparameters of our language models. All cross-entropy losses use mean reduction.

	Chemlactica-125M	Chemlactica-1.3B	Chemma-2B
Peak learning rate	1.4e-3	1.0e-4	1.0e-3
Warmup steps	500	500	500
Context length	2048	2048	2048
ADAM β_1	0.9	0.9	0.9
ADAM β_2	0.95	0.95	0.95
ADAM ϵ	1e-8	1e-8	1e-8
Weight Decay	0.1	0.1	0.1
Dropout	0.1	0.1	None
Attention Dropout	0.1	0.1	None
Precision	Mixed	Mixed	BF16
Loss Function	CE Loss	CE Loss	CE Loss
Vocabulary Size	50066	50066	256000
Gradient Clipping	1.0	1.0	1.0

during the optimization process), K (fine-tuning tolerance level) and lr (fine-tuning peak learning rate) with the following grid:

- $P = [10, 30, 50]$
- $S = [1, 2]$
- $N = [100, 200]$
- $temp_schedule = [[1.5, 1.0], [1.3, 1.0], [1.0, 1.0], [1.0, 1.3], [1.0, 1.5]]$
- $K = [2, 3, 5]$
- $lr = [10^{-4}, 10^{-5}]$

A.6.3 CONDITIONAL GENERATION HYPERPARAMETERS

Our generation process benefits from the following techniques to improve output quality:

- **Chain-of-Thought (CoT):** We omit [START_SMILES] from the initial prompt, enabling the model to generate more property values before the molecule itself.
- **Repetition Penalty:** Applied to discourage repetitive outputs.
- **Undesired Token Suppression:** Employed to ensure the model eventually generates [START_SMILES].

Table 9 provides an ablation study of these sampling components across our three models, demonstrating their individual and combined impacts on generation quality. Surprisingly, the best combination of hyperparameters, as chosen by lowest corrected RMSE (RMSE(c)), coincides with all three models. DNF (Did Not Finish) trial exceeded 30 minutes of runtime when it was manually terminated.

A.7 CHEMLACTICA VS. GALACTICA COMPARISON

To evaluate the efficacy of our pretraining approach in comparison to the base Galactica model, we utilized the BBBP task, introduced by Wu et al. (2018). The BBBP task is a binary classification problem that requires the model to predict whether a given molecule can penetrate the blood-brain barrier. Notably, the pretraining dataset for the base Galactica model included the training set of the BBBP dataset in a question answering format, hence enabling BBBP prediction without further fine-tuning. We show the model's performance as reported in the original publication as well as our reproduction. We conducted supervised fine-tuning on base Galactica-125M and Chemlactica-125M, each using their respective data formats. It is important to note that supervised fine-tuning is necessary, as the general capabilities of our models do not inherently enable the prediction of specific downstream tasks. The results, presented in Table 8, demonstrate that although hyperparameter tuning of Galactica improved model performance significantly, Chemlactica demonstrated better performance without any tuning. With the best hyperparameters, Chemlactica's results further improve. These results suggest that our continued pretraining improves the models' ability to adapt to downstream tasks with fine-tuning.

Table 8: Comparison of Galactica-125M vs Chemlactica-125M on BBBP binary classification.

	ROC ↑
Galactica-125M (original paper)	0.393
Galactica-125M (our reproduction)	0.417
Galactica-125M (SFT)	0.512
Chemlactica-125M (SFT)	0.729
Galactica-125M (SFT - HP tuned)	0.695
Chemlactica-125M (SFT - HP tuned)	**0.739**

A.8 THE ALGORITHM FOR CONVERTING MOLECULES TO PROMPT

Algorithm 2 shows the procedure of converting a molecule and its similar molecule into either a prompt for new molecule generation or a training sample. The separator token represented by <sep> corresponds to the eos token "</s>" for the Chemlactica models and the bos token "<bos>" for Chemma. This keeps consistency with models' training data described in 4.

Table 9: Ablation study on Conditional Generation hyperparameters. Each row represents one combination of Chain-of-Thought (CoT), repetition penalty (rep.), and suppression (supp.). All experiments are done on the molecular weight (top) and SAS (bottom) prediction tasks.

			Chemlactica-125M		Chemlactica-1.3B		Chemma-2B	
CoT	rep.	supp.	RMSE (c) ↓	Invalids ↓	RMSE (c) ↓	Invalids ↓	RMSE (c) ↓	Invalids ↓
No	1.0	No	70.11 (70.11)	0/100	15.81 (65.32)	1/100	12.15 (64.54)	1/100
Yes	1.0	No	112.52 (112.52)	0/100	187.26 (187.26)	0/100	198.48 (191.89)	46/100
Yes	1.010	No	82.28 (82.28)	0/100	137.19 (137.19)	0/100	170.02 (170.02)	0/100
Yes	1.0	Yes	33.46 (33.46)	0/100	18.53 (25.22)	1/100	31.98 (31.85)	1/100
Yes	1.005	Yes	34.52 (34.52)	0/100	17.14 (17.14)	0/100	29.71 (29.71)	0/100
Yes	**1.010**	**Yes**	**30.27 (30.27)**	**0/100**	**16.87 (16.87)**	**0/100**	**18.93 (20.39)**	**1/100**
Yes	1.015	Yes	30.27 (30.27)	0/100	18.07 (19.61)	1/100	18.99 (20.44)	1/100
Yes	1.020	Yes	31.17 (31.17)	1/100	16.33 (18.03)	1/100	24.16 (25.27)	1/100
Yes	1.050	Yes	45.38 (45.38)	1/100	16.49 (34.48)	1/100	74.78 (130.11)	63/100
Yes	1.100	Yes	35.20 (35.20)	0/100	16.61 (32.37)	1/100	740.28 (488.73)	59/100
No	1.0	No	0.268 (0.769)	19/100	0.395 (0.395)	1/100	0.391 (0.431	4/100
Yes	1.0	No	0.887 (0.887)	0/100	0.866 (0.866)	0/100	DNF	46/100
Yes	1.010	No	0.951 (0.951)	0/100	0.691 (0.691)	0/100	0.769 (0.769)	0/100
Yes	1.0	Yes	0.436 (0.436)	0/100	0.470 (0.470)	2/100	0.253 (0.253)	1/100
Yes	1.005	Yes	0.439 (0.439)	0/100	0.475 (0.475)	0/100	0.348 (0.363)	1/100
Yes	1.010	Yes	0.432 (0.432)	0/100	**0.281 (0.281)**	**0/100**	**0.275 (0.275)**	**0/100**
Yes	1.015	Yes	0.373 (0.378)	1/100	0.540 (0.536)	2/100	0.331 (0.347)	2/100
Yes	1.020	Yes	0.432 (0.432)	0/100	0.369 (0.369)	0/100	0.325 (0.341)	2/100
Yes	1.050	Yes	0.294 (0.733)	3/100	0.843 (0.951)	10/100	DNF	63/100
Yes	**1.100**	**Yes**	**0.381 (0.381)**	0/100	0.449 (0.449)	1/100	DNF	59/100

Algorithm 2 molecules2prompt

Input: $(m_1, m_2, \ldots, m_S), m$
1. Check if the outcome should be a molecule generation prompt or a training sample.
if m is $null$ **then**
  1.1. Sample similarity values for molecules in the prompt, desirable oracle score and set the suffix for a molecule generation.
  $v_i^{sim} \sim \mathcal{U}(0.4, 0.9), i = 1, \ldots, S$
  $v^{max} \leftarrow$ the maximum oracle score achieved thus far
  $v^{prop} \sim \mathcal{U}(v^{max}, oracle_max)$
  $suffix \leftarrow$ [START_SMILES]
else
  1.3. Compute the correct similarity values for the molecules in the prompt and the correct oracle score, set the suffix for a training sample.
  $v_i^{sim} = similar(m_i, m), i = 1, \ldots, S$
  $v^{prop} = O(m)$
  $suffix \leftarrow$ [START_SMILES] m^{smiles} [END_SMILES]<sep>
end if
2. Concatenate all molecules in the prompt with their similarity values.
$p \leftarrow$ [SIMILAR] m_1^{smiles} v_1^{sim} [/SIMILAR] ... [SIMILAR] m_S^{smiles} v_S^{sim} [/SIMILAR]
if at least one fine-tuning has been performed **then**
  2.1. Add the oracle score to the prompt.
  $p \leftarrow concat(p,$ [PROPERTY] v^{prop} [/PROPERTY]$)$
end if
3. Add the appropriate suffix.
return $concat(p, suffix)$

A.9 Diversity Results for Docking MPO; parp1, fa7, 5ht1b, braf, and jak2

Table 10 presents the #Circles metric for molecules satisfying the Strict Hit Ratio conditions(Xie et al., 2023). We display the results for our methods alongside the baselines to facilitate comparison. The results demonstrate that our approach generates more diverse high-reward molecules than other methods.

Table 10: Comparison of our approach with other methods. The values represent the #Circles (↑) ± standard deviation for molecules which satisfy the criteria for the Strict Hit Ratio across 10 independent runs. The results of GEAM and Saturn are taken from Lee et al. (2024) and Guo and Schwaller (2024), respectively. Results within one standard deviation from the best one are bolded.

Method	Target Protein				
	parp1	fa7	5ht1b	braf	jak2
GEAM	14 ± 3	7 ± 2	25 ± 3	11 ± 2	18 ± 2
Saturn	5 ± 0	3 ± 1	17 ± 3	4 ± 0	7 ± 1
Chemlactica-125M	41 ± 7	40 ± 15	38 ± 7	42 ± 6	41 ± 5
Chemlactica-1.3B	**59 ± 10**	**61 ± 9**	**53 ± 7**	**62 ± 7**	**63 ± 15**
Chemma-2B	**66 ± 16**	**73 ± 14**	**64 ± 14**	**72 ± 14**	**72 ± 19**

Table 11: Comparision of different methods on PMO. The values represent the AUC Top-10 ↑ metric averaged over five independent runs with different seeds.

Oracle	REINVENT	Augmented Memory	Genetic GFN	Chemlactica 125M	Chemlactica 1.3B	Chemma 2B
albuterol_similarity	0.882 ± 0.006	0.913 ± 0.009	0.949 ± 0.010	**0.951 ± 0.011**	0.947 ± 0.012	**0.951 ± 0.009**
amlodipine_mpo	0.635 ± 0.035	0.691 ± 0.047	0.761 ± 0.019	**0.772 ± 0.091**	0.769 ± 0.083	0.766 ± 0.107
celecoxib_rediscover	0.713 ± 0.067	0.796 ± 0.008	0.802 ± 0.029	0.906 ± 0.046	0.911 ± 0.013	**0.920 ± 0.011**
deco_hop	0.666 ± 0.044	0.658 ± 0.024	0.733 ± 0.109	0.801 ± 0.101	**0.836 ± 0.117**	0.831 ± 0.123
drd2	0.945 ± 0.007	0.963 ± 0.006	**0.974 ± 0.006**	0.965 ± 0.007	0.968 ± 0.005	0.972 ± 0.006
fexofenadine_mpo	0.784 ± 0.006	0.859 ± 0.009	0.856 ± 0.039	0.881 ± 0.031	0.891 ± 0.039	**0.931 ± 0.014**
gsk3	0.865 ± 0.043	0.881 ± 0.021	0.881 ± 0.042	0.926 ± 0.022	0.916 ± 0.027	**0.928 ± 0.021**
isomers_c7h8n2o2	0.852 ± 0.036	0.853 ± 0.087	0.969 ± 0.003	**0.951 ± 0.012**	0.933 ± 0.017	0.947 ± 0.009
isomers_c9h10n2o2pf2cl	0.642 ± 0.054	0.736 ± 0.051	0.897 ± 0.007	0.927 ± 0.006	**0.929 ± 0.012**	0.914 ± 0.017
jnk3	0.783 ± 0.023	0.739 ± 0.110	0.764 ± 0.069	0.881 ± 0.058	0.866 ± 0.021	**0.891 ± 0.032**
median1	0.356 ± 0.009	0.326 ± 0.013	0.379 ± 0.010	0.359 ± 0.060	**0.382 ± 0.047**	**0.382 ± 0.022**
median2	0.276 ± 0.008	0.291 ± 0.008	0.294 ± 0.007	0.328 ± 0.032	0.329 ± 0.016	**0.366 ± 0.018**
mestranol_similarity	0.618 ± 0.048	0.750 ± 0.049	0.708 ± 0.057	0.896 ± 0.064	0.850 ± 0.051	**0.926 ± 0.023**
osimertinib_mpo	0.837 ± 0.009	0.855 ± 0.004	0.860 ± 0.008	**0.907 ± 0.015**	0.892 ± 0.013	0.879 ± 0.016
perindopril_mpo	0.537 ± 0.016	0.613 ± 0.015	0.595 ± 0.014	0.709 ± 0.052	**0.755 ± 0.066**	0.711 ± 0.062
qed	0.941 ± 0.000	**0.942 ± 0.000**	**0.942 ± 0.000**	**0.942 ± 0.000**	**0.942 ± 0.000**	0.941 ± 0.000
ranolazine_mpo	0.760 ± 0.009	0.801 ± 0.006	0.819 ± 0.018	0.864 ± 0.014	**0.883 ± 0.017**	0.868 ± 0.015
scaffold_hop	0.560 ± 0.019	0.567 ± 0.008	0.615 ± 0.100	0.626 ± 0.016	**0.673 ± 0.080**	0.669 ± 0.110
sitagliptin_mpo	0.021 ± 0.003	0.284 ± 0.050	0.634 ± 0.039	**0.649 ± 0.051**	0.586 ± 0.062	0.613 ± 0.018
thiothixene_rediscovery	0.534 ± 0.013	0.550 ± 0.041	0.583 ± 0.034	0.624 ± 0.102	0.693 ± 0.119	**0.698 ± 0.121**
troglitazone_rediscovery	0.441 ± 0.032	0.540 ± 0.048	0.511 ± 0.054	0.734 ± 0.130	0.765 ± 0.138	**0.824 ± 0.049**
valsartan_smarts	**0.178 ± 0.358**	0.000 ± 0.000	0.135 ± 0.271	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zaleplon_mpo	0.358 ± 0.062	0.394 ± 0.026	0.552 ± 0.033	0.569 ± 0.047	0.569 ± 0.020	**0.608 ± 0.055**
sum	14.196	15.002	16.213	17.170 ± 0.424	17.284 ± 0.284	**17.534 ± 0.214**

A.10 Detailed Results for Practical Molecular Optimization

Table 11 shows the evaluations of Chemlactica-125M, Chemlactica-1.3B and Gemma-2B, along with other methods on 23 tasks of the PMO benchmark. No method uniformly beats all others on every task. Our (and many other) methods get a zero result on `valsartan_smarts`. The reason is that the oracle has a binary multiplier term usually equal to zero, so there is no supervision signal for the entire generation process.

A.11 Analysis of Molecular Optimization

A.11.1 Ablation on Finetuning During Optimization

A key component of our proposed optimization algorithm is the fine-tuning step, initiated when the algorithm's progress stagnates. To assess the impact of this fine-tuning step, we conducted a comparative analysis of optimization processes both with and without this feature. For this evaluation, we selected four representative tasks from the PMO benchmark: `jnk3`, `median1`, `sitagliptin_mpo`, and `scaffold_hop`. We select these tasks to provide diverse challenges and adequately represent the full suite of PMO tasks.

Table 12: Illustration of the results of ablation study on the fine-tuning step in the optimization algorithm. The values represent AUC Top-10 ↑ obtained from five independent runs.

	Chemlactica-125M		Chemlactica-1.3B		Chemma-2B	
	fine-tuning	no fine-tuning	fine-tuning	no fine-tuning	fine-tuning	no fine-tuning
jnk3	0.881 ± 0.058	0.878 ± 0.040	0.866 ± 0.021	0.867 ± 0.036	0.891 ± 0.032	0.869 ± 0.033
median1	0.359 ± 0.060	0.371 ± 0.006	0.382 ± 0.047	0.395 ± 0.027	0.382 ± 0.022	0.380 ± 0.034
scaffold_hop	0.626 ± 0.016	0.648 ± 0.017	0.673 ± 0.080	0.721 ± 0.121	0.669 ± 0.110	0.700 ± 0.122
sitagliptin_mpo	0.649 ± 0.051	0.607 ± 0.051	0.586 ± 0.062	0.576 ± 0.082	0.613 ± 0.018	0.563 ± 0.059
sum	$\mathbf{2.515 \pm 0.119}$	2.504 ± 0.068	2.506 ± 0.155	$\mathbf{2.559 \pm 0.062}$	$\mathbf{2.555 \pm 0.099}$	2.512 ± 0.160

Table 13: The performance of the extended version of our optimization algorithm on selected PMO tasks. The prompts used in the optimization contain the description of the tasks in the format our language models has seen during pretraining. See Table 14 for the additional tags used in the prompts.

	Chemlactica-125M		Chemlactica-1.3B		Chemma-2B	
	no add. props.	add. props.	no add. props.	add. props.	no add. props.	add. props.
jnk3	0.881 ± 0.058	0.881 ± 0.058	0.866 ± 0.021	0.866 ± 0.021	0.891 ± 0.032	0.891 ± 0.032
median1	0.359 ± 0.060	0.479 ± 0.004	0.382 ± 0.047	0.488 ± 0.000	0.382 ± 0.022	0.479 ± 0.002
scaffold_hop	0.626 ± 0.016	0.983 ± 0.004	0.673 ± 0.080	0.975 ± 0.006	0.669 ± 0.110	0.983 ± 0.003
sitagliptin_mpo	0.649 ± 0.051	0.534 ± 0.041	0.586 ± 0.062	0.495 ± 0.035	0.613 ± 0.018	0.576 ± 0.055
sum	2.515 ± 0.119	$\mathbf{2.920 \pm 0.096}$	2.506 ± 0.155	$\mathbf{2.824 \pm 0.034}$	2.555 ± 0.099	$\mathbf{2.887 \pm 0.040}$

Table 12 presents the quantitative results of these experiments. To provide a more comprehensive understanding of the fine-tuning effect, we visualize the optimization trajectories in Figures 8 through 10. These visualizations aggregate data from five independent runs, offering insights into both the mean performance and its variance across different initializations.

This ablation study allows us to isolate the impact of the fine-tuning step and understand its contribution to the overall performance of our optimization algorithm across different types of molecular optimization tasks.

A.11.2 Leveraging Known Molecular Properties in Optimization Tasks

Our language models possess knowledge of various molecular properties such as QED, CLogP, and TPSA. However, we deliberately avoid utilizing this information in Algorithm 1 to maintain fair comparison with other methods. This decision stems from the fact that our models have been trained on properties that are components of the oracle functions we optimize against (e.g., those in PMO). Exploiting this partial oracle information could potentially give our method an unfair advantage.

We conducted a separate set of experiments to explore the models' capacity to utilize additional information in solving optimization problems using four tasks from the PMO benchmark: `jnk3`, `median1`, `sitagliptin_mpo`, and `scaffold_hop`. For these tasks, we modified Algorithm 2 to incorporate relevant known properties into the prompt p between steps 2 and 3.

Table 13 presents a performance comparison between our standard approach and this property-augmented version. The specific syntax used for adding these properties to the prompts is detailed in Table 14. Notably, no additional properties were added for the `jnk3` docking function as our models lack specific knowledge about this component.

The results demonstrate a significant performance improvement across all models when these additional properties are incorporated. This finding suggests that our models can effectively leverage their pre-existing knowledge of molecular properties to enhance their performance in molecular design tasks. However, it is important to note that while this approach showcases the potential of our models, it may not provide a fair comparison with methods that do not have access to such property information.

A.11.3 The Impact of Floating Point Precision on Molecular Optimization

Numerical Precision in Model Training Lower precision training, including mixed and half-precision methods, is commonly used to increase training throughput. These techniques, employed during our models' pretraining stages, typically have negligible impact on performance and may even provide a regularizing effect(Micikevicius et al., 2017). However, in molecular optimization involving multiple rounds of fine-tuning, lower numerical precision leads to significantly degraded performance. Several factors contribute to this phenomenon in the specific case of molecular optimization with language models.

Table 14: The descriptions of tasks used in the prompts in the extended version of our optimization algorithm. The results are in Table 13. See Section A.11.2 for details.

	the syntax of additional properties added to the prompts
jnk3	(nothing added)
median1	[SIMILAR]*camphor_smiles* 0.55[/SIMILAR][SIMILAR]*menthol_smiles* 0.55[/SIMILAR]
scaffold_hop	[SIMILAR]*pharmacophor_smiles* 0.80[/SIMILAR]
sitagliptin_mpo	[SIMILAR]*sitagliptin_smiles* 0.99[/SIMILAR][CLOGP]2.02[/CLOGP][TPSA]77.04[/TPSA]

Table 15: Impact of numerical precision on Docking MPO experiments from 6.2.2. Oracle burden and generative yield values are reward-threshold dependent. The numbers next to the metrics correspond to the thresholds, and the values in parentheses for oracle burden indicate how many unique molecules need to be generated. The best performance on each task-metric combination is bolded. We use the best-performing hyperparameters from the PMO benchmark.

Metric	Target	Chemlactica-125M BF16	Chemlactica-125M FP32
Generative Yield 0.7 ↑	DRD2	3501 ± 252	**3733 ± 512**
	MK2	3000 ± 80	**3772 ± 578**
	AChE	**4337 ± 133**	4108 ± 67
Generative Yield 0.8 ↑	DRD2	2574 ± 103	**2827 ± 510**
	MK2	1223 ± 519	**2569 ± 1156**
	AChE	3877 ± 272	3246 ± 168
Oracle burden 0.8 (1) ↓	DRD2	156 ± 100	**20 ± 29**
	MK2	**320 ± 83**	345 ± 312
	AChE	**10 ± 8**	22 ± 28
Oracle burden 0.8 (10) ↓	DRD2	283 ± 61	**114 ± 08**
	MK2	631 ± 100	**493 ± 418**
	AChE	**123 ± 119**	224 ± 17
Oracle burden 0.8 (100) ↓	DRD2	577 ± 71	**364 ± 119**
	MK2	1134 ± 178	**865 ± 533**
	AChE	**350 ± 137**	497 ± 58

Challenges in Batched Generation Molecular optimization pipelines require repeated model calls for generation, followed by oracle function scoring. While batched processing accelerates this process through GPU parallelization, it introduces complications. The necessary padding for batch processing alters matrix sizes, affecting multiply-accumulate operations within the model. These small errors accumulate as they propagate through the model's layers. Lower precision exacerbates these errors, leading to larger discrepancies in logit values and, consequently, more significant impacts on the generated molecules.

Cascading Effects of Sub-optimal Generations In our approach, high-scoring generated molecules are leveraged for fine-tuning and generating similar structures that steer the optimization processs. Thus, when lower precision leads to sub-optimal molecule generation, it creates a positive feedback loop. The model is fine-tuned on and guided by these lower-quality molecules, hindering the generation of higher-scoring molecules in subsequent iterations. This causal relationship between successive generations underlies the adverse effects of low-precision training and inference in molecular optimization pipelines.

Precision Ablation Study To quantify the impact of numerical precision on the optimization process, we conducted an ablation study comparing 32-bit floating point precision with bfloat16 precision. Table 15 presents the results of this comparison across all drug discovery case studies described in Section 6.2.2. We show that for the majority of task-metric combinations, optimization results were better when model parameters were in full floating point precision. Despite the potential computational costs, these results demonstrate the importance of maintaining higher numerical precision in molecular optimization tasks.

A.12 Additional Experiments

A.12.1 QED Maximization with Similarity Constrained Molecular Design

Problem formulation. This optimization problem aims to generate a molecule with a high QED, similar to another given molecule. More formally, given a molecule M, the objective is to generate a new molecule M' such that $sim(M', M) \geq 0.4$ and $qed(M') \geq 0.9$. Following Wang et al. (2023), 800 molecules are selected

with QED in the range $[0.7, 0.8]$ as the inputs to the optimization problem, and the measure of performance is the percentage of the molecules that have been optimized (satisfy the QED and similarity constraints). In addition, a maximum number of QED evaluations is chosen to optimize each lead molecule.

Our approach. Since this is a lead optimization problem, we add the lead molecule to all prompts in addition to the molecules added from the pool. The lead molecule is added by enclosing it in `[SIMILAR]` tag. For this task, we design an oracle function by combining the QED value of the generated molecule with the similarity value of the lead molecule with the generated molecule. Additionally, we decreased the maximum number of QED evaluations to 10000, compared to the baselines, which used 50000.

Results. For this task, we only evaluate the Chemlactica-125M model, which achieves better success rates compared to the best existing approaches, 99.0% (Chemlactica-125M) versus 94.6% (RetMol), while being constrained to use five times less QED evaluations at maximum. Since the performance of the Chemlactica-125M saturates the benchmark, we have not evaluated other models for this task. Table 16 illustrates the performance of different algorithms.

A.12.2 Property Prediction

Supervised fine-tuning recipe. Inspired by instruction tuning methodologies and Zhou et al. (2023), we generated a specialized training corpus formatted as follows:

`[START_SMILES]`m^{smiles}`[END_SMILES][PROPERTY]activity <VALUE>[/PROPERTY].`

Hyperparameters. We only trained the model on generated responses following the [PROPERTY] tag during the fine-tuning process. Our initial experiments indicated that a general fine-tuning recipe of 15 epochs yielded satisfactory results with a peak learning rate of $10e-4$, 3 epochs of warmup and a NEFTune noise (Jain et al., 2023) of 5. To further improve model performance, we conducted an extensive hyperparameter tuning study, exploring a grid of values within the following ranges: Learning rate: [0.00001, 0.00005, 0.0001, 0.0002], Number of epochs: [10, 15, 20], Warmup epochs: [0, 1, 2, 3], NEFTune noise : [0.0, 5.0, 10.0]. In our experiments, we employed a batch size of 32 and a maximum sequence length of 128, except in cases where GPU memory limitations necessitated reducing the batch size to 16 while maintaining the established sequence length. Table 17 shows the best values for all tasks and models.

Results. Table 2 lists the results for three regression tasks from MoleculeNet (Wu et al., 2018) alongside other comparable methods like Chilingaryan et al. (2024) and Ross et al. (2021). For all Moleculenet tasks, we have utilized the DeepChem library Ramsundar et al. (2019) and the original recommended splits to load the datasets. .Fang et al. (2023b) introduces a novel dataset encompassing six ADME targets. The assessment of ADME properties is crucial for understanding how potential drug candidates interact with the human body, aspects of which are absorption, distribution, metabolism, and excretion. This knowledge is essential for evaluating efficacy, safety, and clinical potential, guiding drug development toward optimal therapeutic outcomes. The authors have disclosed DMPK datasets collected over a 20-month period, focusing on six ADME in vitro endpoints: human and rat liver microsomal stability, MDR1-MDCK efflux ratio, solubility, and human and rat plasma protein binding. The dataset comprises between 885 and 3087 measurements for each corresponding endpoint. For this series of tasks, we utilized Polaris Hub Wognum et al. (2024) as a centralized platform for dataset loading and result sharing. To promote standardized benchmarks in the field, we employed the datasets as presented, with default preprocessing. We limited our comparisons to the results available at the time, including the baselines provided by the original authors. We generated a randomly split validation set for this series of tasks, comprising 20 percent of the training data. After identifying the optimal hyperparameters, we trained on the entire training set to maximize performance. Table 18 presents the results for ADME tasks. The presented results showcase the abilities of our models after the hyperparameter tuning stage.

Table 16: Performance comparison of different algorithms on QED and Similarity constrained molecular optimization problem.

	Success Rate (%) $\uparrow$
QMO	92.8
RetMol	94.5
Chemlactica-125M	**99.0**

A.12.3 MODEL CALIBRATION

Methodology Model calibration in language modeling refers to the alignment between a model's predicted probabilities for generating specific text and the actual likelihood of that text being correct. To assess the calibration of our models, we developed a suite of multiple-choice property prediction questions based on our training data format.

We generated 2000 questions for each computed property, resulting in 10,000 responses. Each question presented a SMILES string as input:

[START_SMILES]m^{smiles}[END_SMILES]

and was followed by five potential continuations, with only one being correct. An example of such a continuation for a question testing QED could be: [QED]0.78[/QED].

This methodology is inspired by the calibration analysis in the GPT-4 technical report (OpenAI, 2023), which highlights calibration as a key indicator of high-quality pretraining. For each response, we calculated the model's predicted probability from the perplexity of the text, normalizing it against other responses for the same question. These probabilities were then aggregated and sorted into 10 equal-width bins. We plotted the fraction of correct responses for each bin, allowing us to visualize the relationship between the model's confidence and accuracy.

Results Figures 3a and 3b present the calibration plots for Chemma-2B and Chemlactica-125M, respectively. The x-axis represents the 10 probability bins, while the left y-axis shows the correct response fraction. The right y-axis and red bars indicate the number of occurrences within each bin.

Chemlactica and Chemma models demonstrate robust calibration, as evidenced by the near-linear relationship between assigned probabilities and correct outcomes across all computed properties. This relationship closely follows the diagonal grey line, which represents perfect calibration.

These results suggest that the perplexity scores generated by our models serve as reliable confidence indicators for molecular data predictions (averaged over a set of molecules), provided the data falls within the distribution of the training corpus. This calibration is crucial for practical applications, as it allows users to accurately gauge the reliability of the models' outputs in simple molecular prediction and generation tasks. However, finetuning, like that performed in the optimization algorithm, likely leads to a loss of model calibration(OpenAI, 2023).

A.13 ADDITIONAL FIGURES

A.13.1 VISUALIZATION OF THE MODEL OUTPUTS ON PROPERTY PREDICTION AND CONDITIONAL GENERATION TASKS

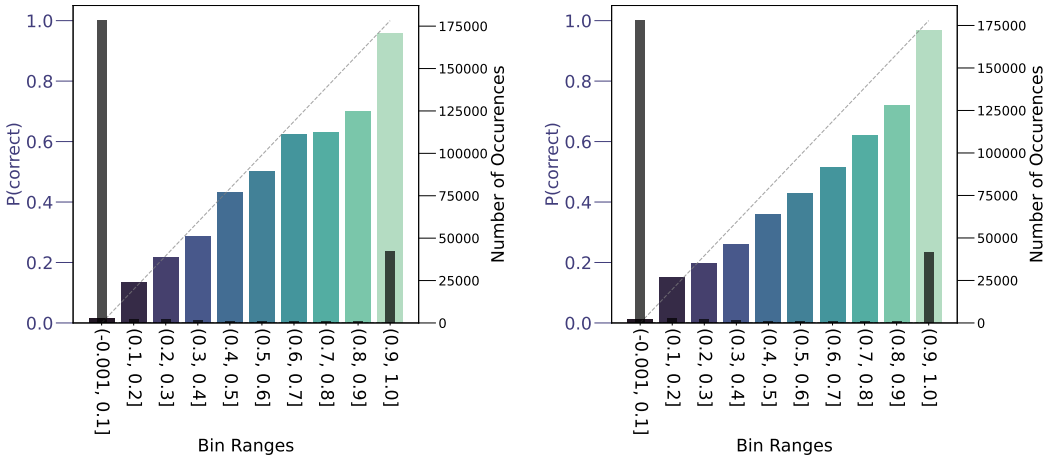
Figures 4e-4e show the performance of Chemma-2B for property prediction and conditional molecular generation tasks. Each dot in the scatter plot corresponds to one molecule. The histogram in the background is the distribution of those properties in our training set. The purple line represents the RMSE between the property's ground truth and predicted values.

Table 17: Selected hyperparameters for property prediction tasks as a result of the grid search. We report learning rate (LR), warmup ratio (WU), number of epochs (Ep.) and Neftune noise (Nef.).

	Chemlactica-125M				Chemlactica-1B				Chemma-2B			
Task	LR	WU	Ep.	Nef.	LR	WU	Ep.	Nef.	LR	WU	Ep.	Nef.
RLM	5.0e-5	0	15	0	1.0e-5	3	10	5	2.0e-4	1	20	10
HLM	5.0e-5	1	10	5	1.0e-5	3	10	0	1.0e-4	3	20	0
MDR	5.0e-5	2	20	0	1.0e-5	1	15	0	2.0e-4	1	20	5
RPPB	1.0e-4	1	15	0	1.0e-5	3	10	0	1.0e-4	1	15	5
HPPB	2.0e-4	3	20	10	1.0e-4	2	15	10	2.0e-4	2	15	0
SOL	1.0e-4	2	10	0	5.0e-5	0	20	5	2.0e-4	1	15	10
FREESOLV	1.0e-4	0	15	5	1.0e-5	3	10	10	1.0e-4	2	15	0
ESOL	1.0e-4	0	10	0	1.0e-5	3	10	5	2.0e-4	3	10	0
LIPO	5.0e-5	1	2	0	1.0e-5	0	10	0	2.0e-4	2	10	0

Table 18: Regression tasks from the ADME benchmark. All numbers are Pearson correlation ↑. RandomForestRegressors use task-specific features: [a] desc2D, [b] fcfp4, [c] atompair.

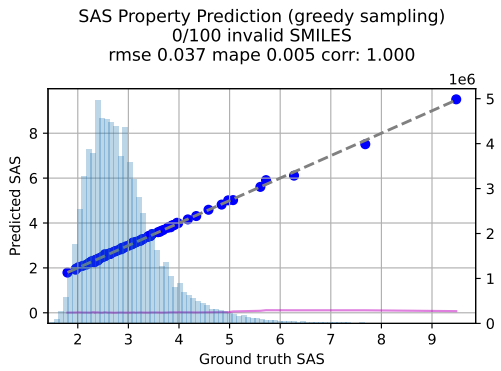
	HLM	MDR	SOL	RLM	HPPB	RPPB
1B_MPNN (LargeMix-and-Phenomics)	**0.778**	**0.860**	**0.764**	**0.784**	**0.888**	**0.908**
adme-fang-RandomForestRegressor	0.639[a]	0.716[a]	0.439[b]	0.640[a]	0.690[c]	0.722[a]
Chemlactica-125M	0.717	0.714	0.608	0.714	0.774	0.442
Chemlactica-1.3B	0.720	0.762	0.574	0.698	0.635	0.614
Chemma-2B	0.674	0.709	0.558	0.660	0.636	0.747



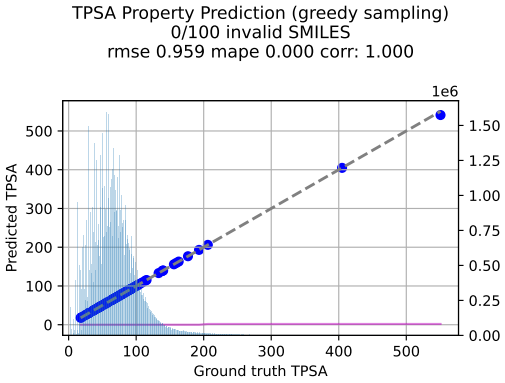
(a) Calibration of Chemma-2B.

(b) Calibration of Chemlactica-125M.

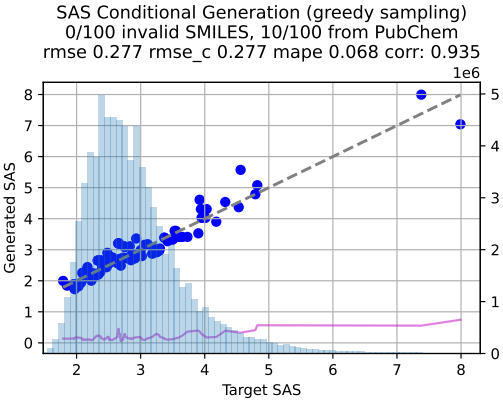
Figure 3: Model calibration on synthetic multiple choice question where y=x represents perfect calibration.



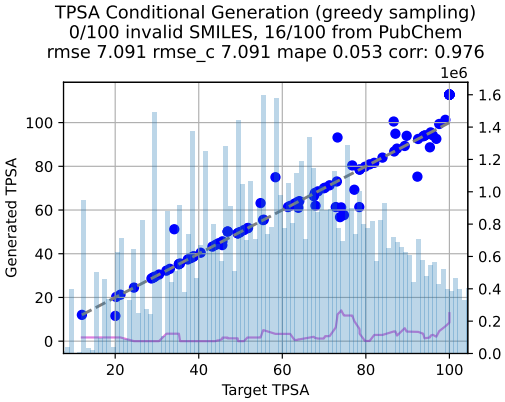
(a) SAS prediction.



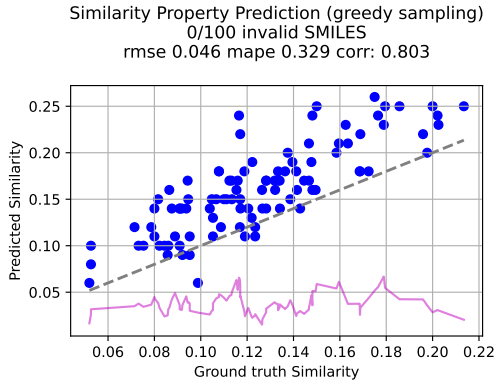
(b) TPSA Prediction.



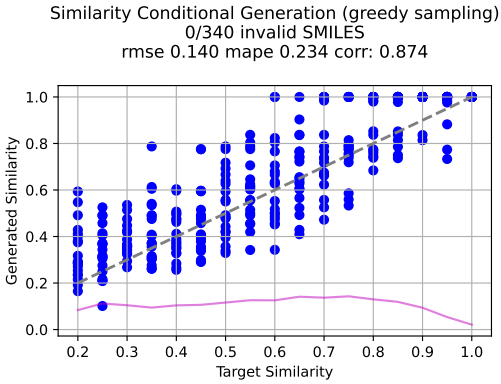
(c) SAS-conditioned generation of molecules.



(d) TPSA-conditioned generation of molecules.



(e) Prediction of similarity between two molecules.



(f) Similarity-conditioned generation of molecules.

Figure 4: Illustration of errors made by Chemma-2B during property prediction and conditional generation for various properties.

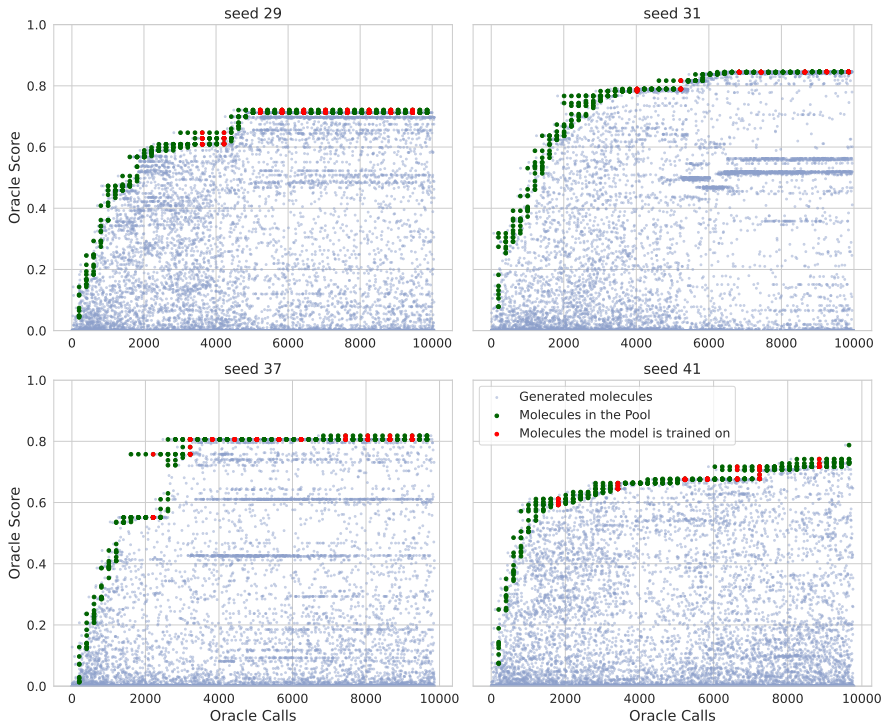


Figure 5: Optimization process visualization using Chemlactica-125M model for `sitagliptin_mpo` task with four different seeds.

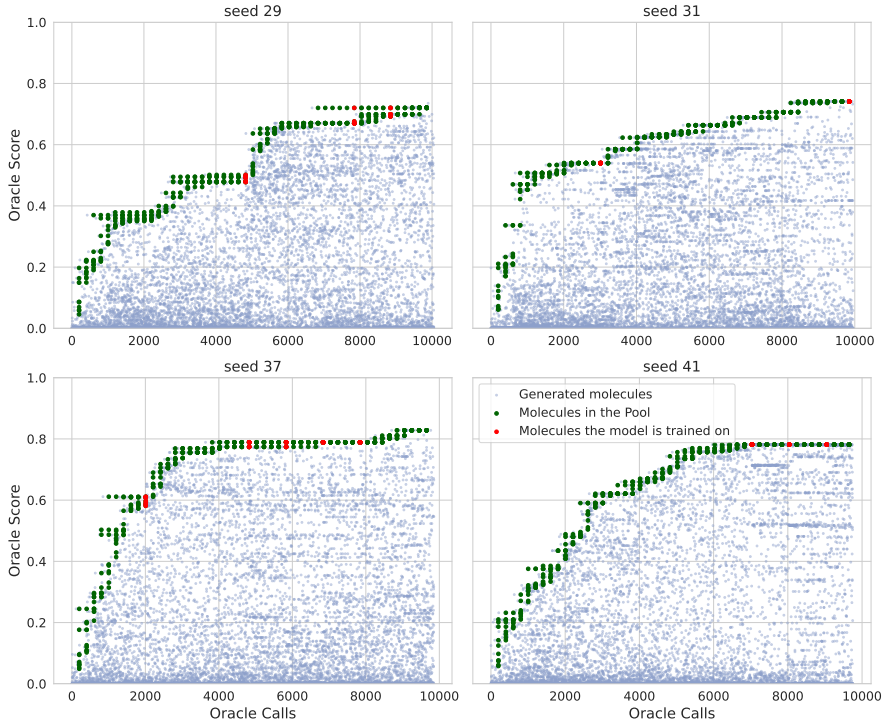


Figure 6: Optimization process visualization using Chemlactica-1.3B model for `sitagliptin_mpo` task with four different seeds.

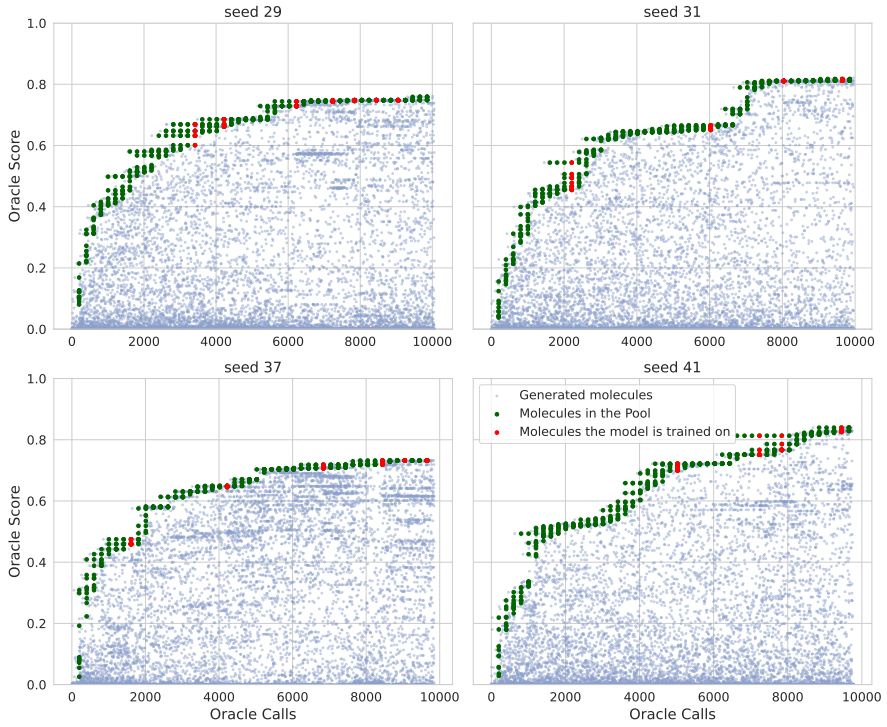


Figure 7: Optimization process visualization using Chemma-2B model for `sitagliptin_mpo` task with four different seeds.

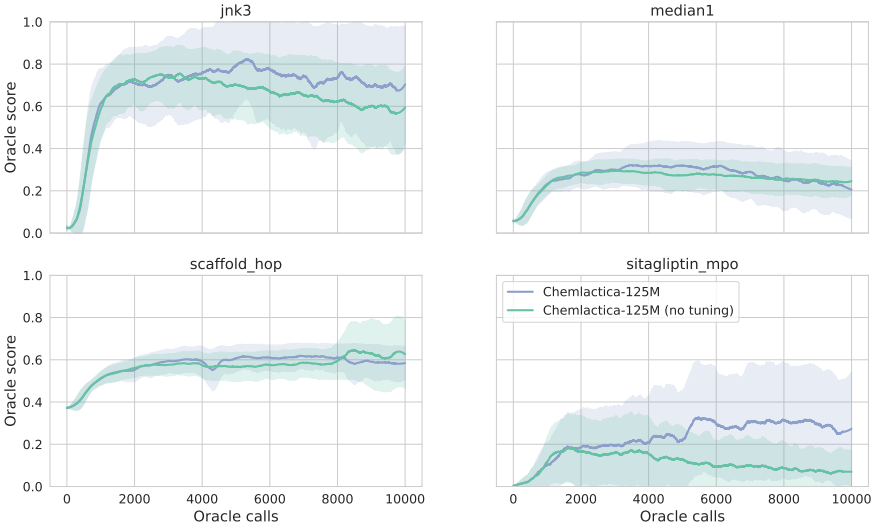


Figure 8: Mean oracle score $\pm$ standard deviation of the generated molecule for Chemlactica-125M.

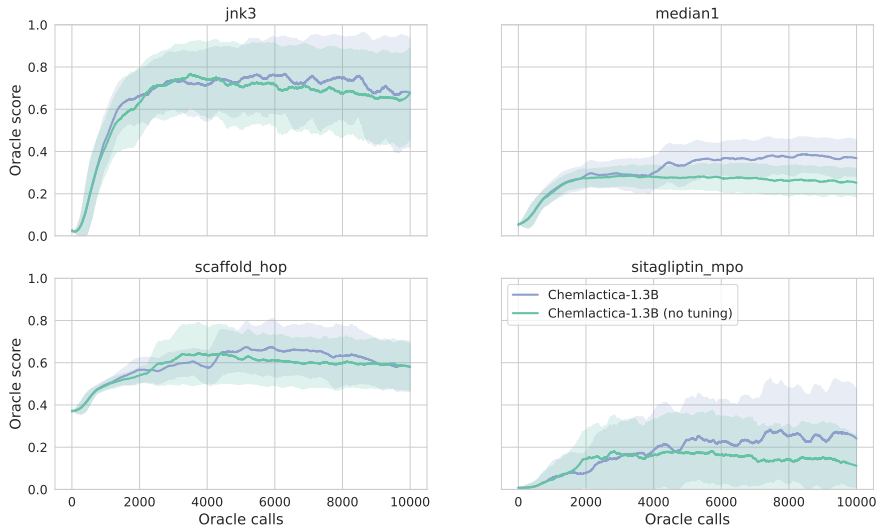


Figure 9: Mean oracle score ± standard deviation of the generated molecule for Chemlactica-1.3B.

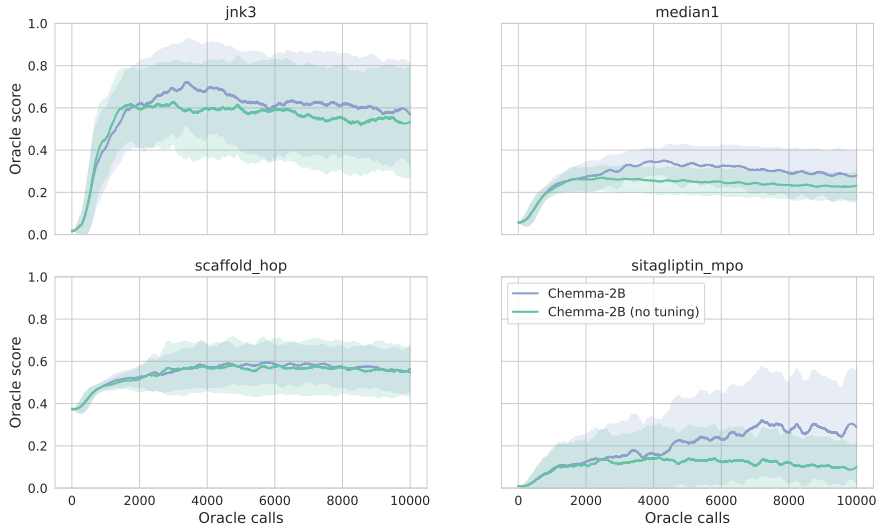


Figure 10: Mean oracle score ± standard deviation of the generated molecule for Chemma-2B.

A.13.2 DOCKING SCORES THROUGHOUT DRD2 MPO

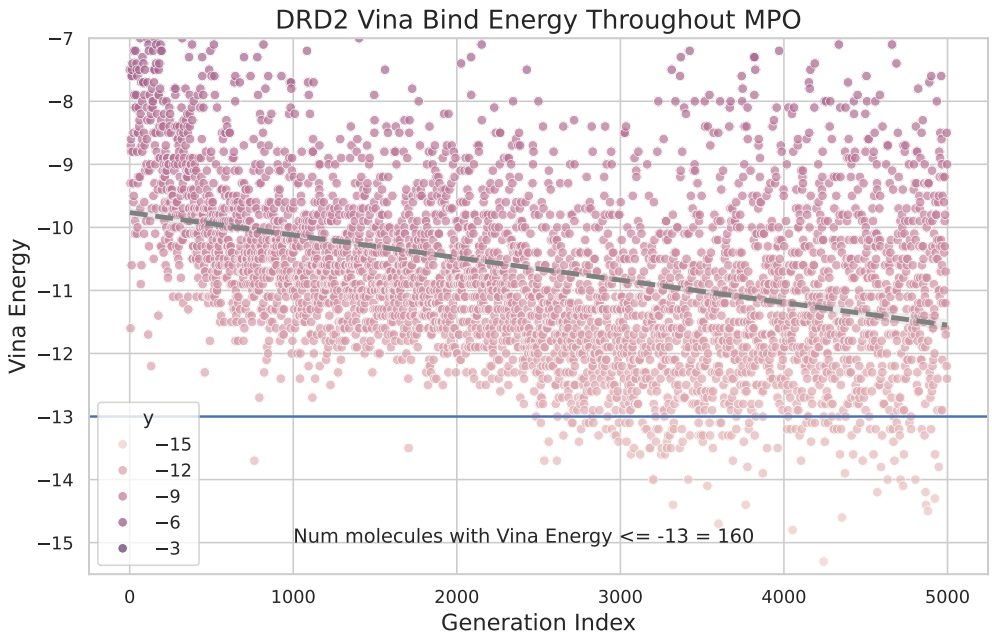
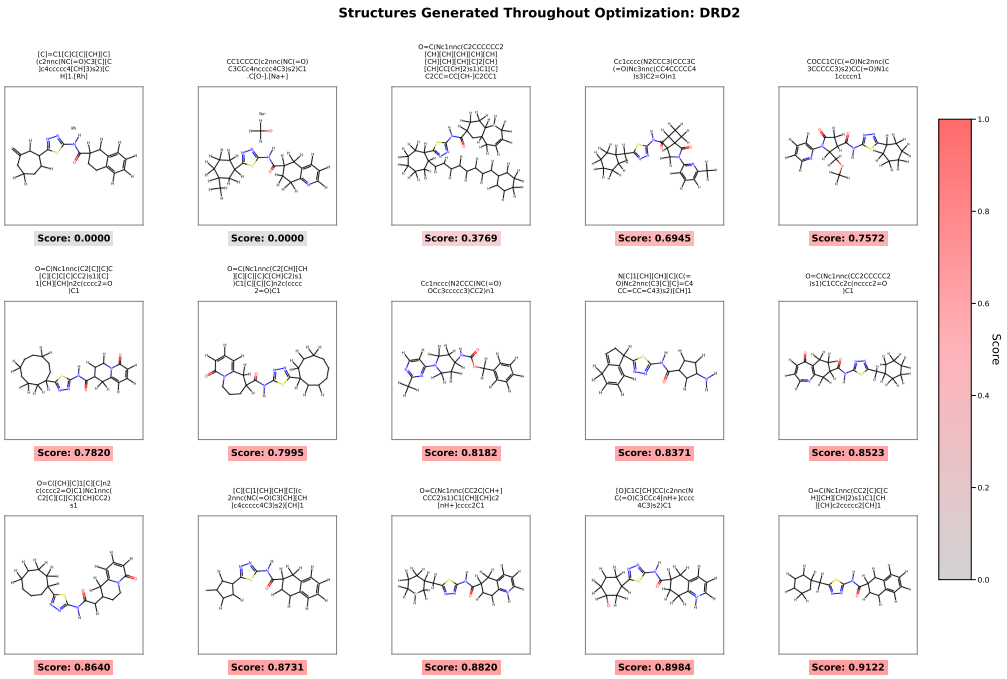
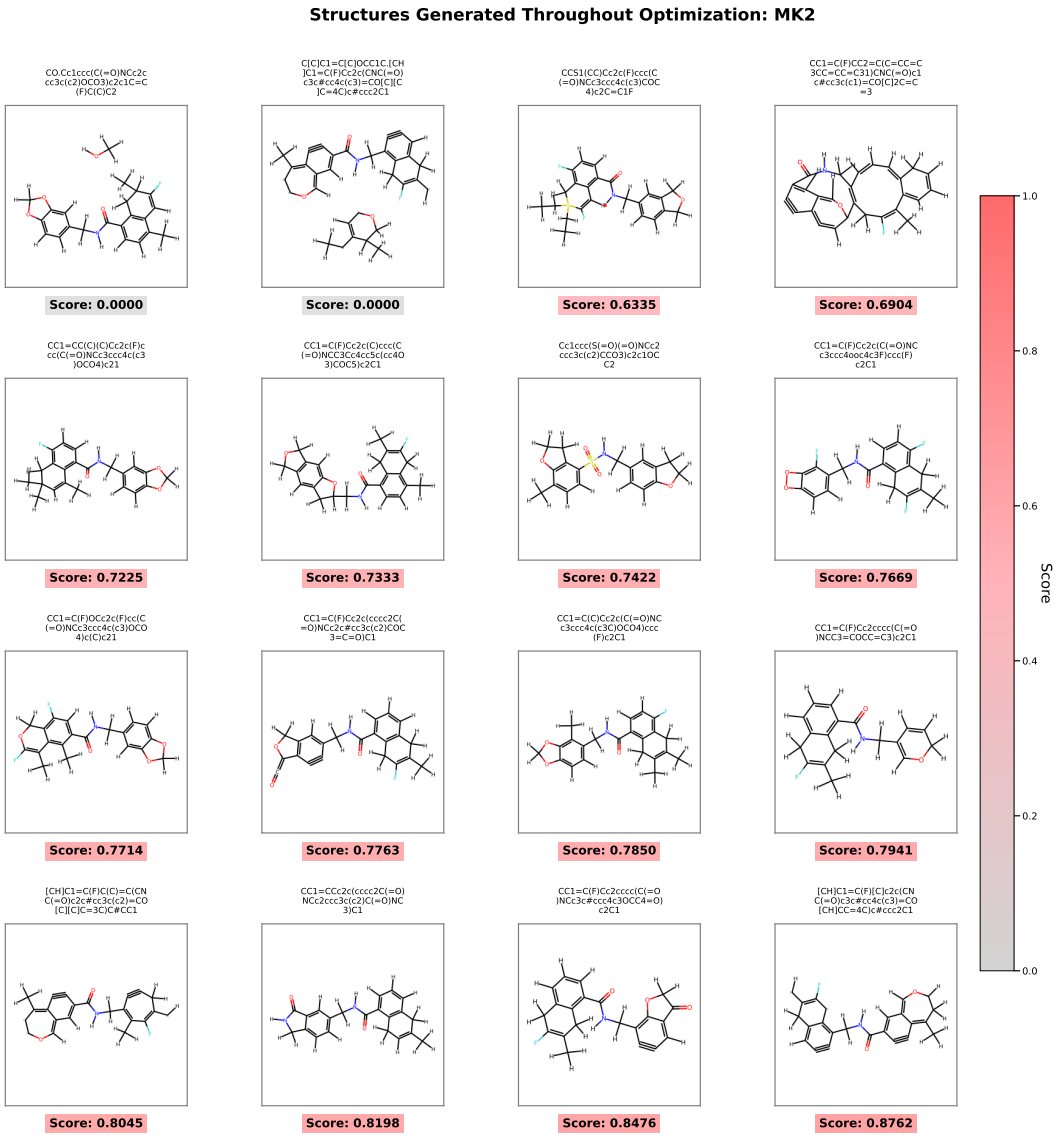
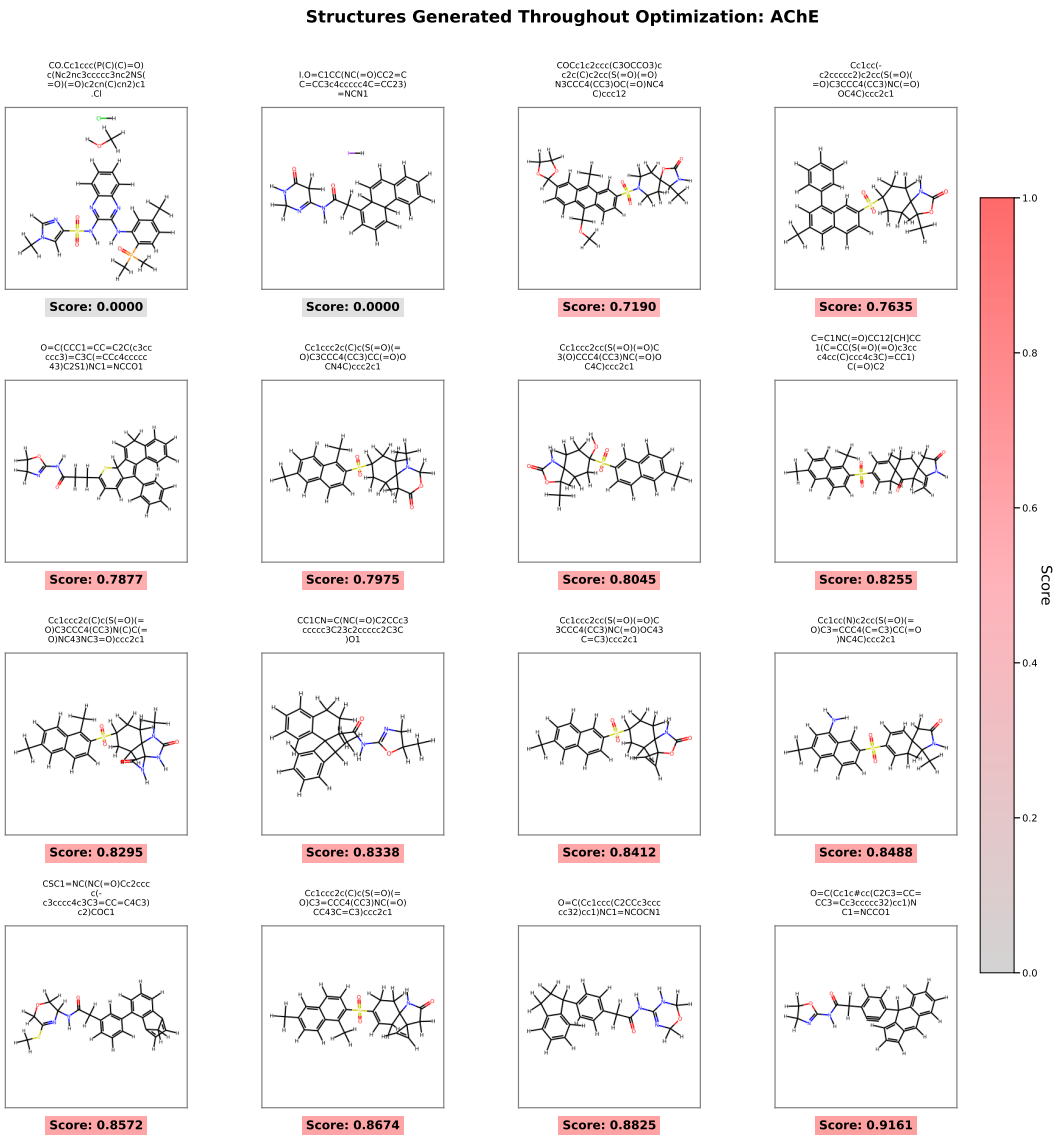


Figure 11: Autodock Vina energy scores for molecules generated through DRD2 MPO process; note that molecules which did not dock at all are not included.

A.13.3 GENERATED MOLECULES FROM THE DOCKING EXPERIMENTS




Structures Generated Throughout Optimization: MK2
CO.Cc1ccc(C(=O)NCc2cc3c(c2)OCO3)c2c1C=C(F)C(C)C2
Score: 0.0000
C[C]C1=C[C]OCC1C.[CH]C1=C(F)Cc2c(CNC(=O)c3c#cc4c(c3)=CO[C][C]=4C)c#ccc2C1
Score: 0.0000
CCS1(CC)Cc2c(F)ccc(C(=O)NCc3ccc4c(c3)COC4)c2C=C1F
Score: 0.6335
CC1=C(F)CC2=C(C=CC=C3CC=CC=C31)CNC(=O)c1c#cc3c(c1)=CO[C]2C=C=3
Score: 0.6904
CC1=CC(C)(C)C)c2c(F)ccc(C(=O)NCc3ccc4c(c3)OCO4)c21
Score: 0.7225
CC1=C(F)Cc2c(C)ccc(C(=O)NCC3Cc4cc5c(cc4O3)COC5)c2C1
Score: 0.7333
Cc1ccc(S(=O)(=O)NCc2ccc3c(c2)CCO3)c2c1OCC2
Score: 0.7422
CC1=C(F)Cc2c(C(=O)NCc3ccc4ooc4c3F)ccc(F)c2C1
Score: 0.7669
CC1=C(F)OCc2c(F)cc(C(=O)NCc3ccc4c(c3)OCO4)c(C)c21
Score: 0.7714
CC1=C(F)Cc2c(cccc2C(=O)NCc2c#cc3c(c2)COC3=C=O)C1
Score: 0.7763
CC1=C(C)Cc2c(C(=O)NCc3ccc4c(c3C)OCO4)ccc(F)c2C1
Score: 0.7850
CC1=C(F)Cc2cccc(C(=O)NCC3=COCC=C3)c2C1
Score: 0.7941
[CH]C1=C(F)C(C)=C(CNC(=O)c2c#cc3c(c2)=CO[C][C]C=3C)C#CC1
Score: 0.8045
CC1=CCc2c(cccc2C(=O)NCc2ccc3c(c2)C(=O)NC3)C1
Score: 0.8198
CC1=C(F)Cc2cccc(C(=O)NCc3c#ccc4c3OCC4=O)c2C1
Score: 0.8476
[CH]C1=C(F)[C]c2c(CNC(=O)c3c#cc4c(c3)=CO[CH]CC=4C)c#ccc2C1
Score: 0.8762
Score
1.0
0.8
0.6
0.4
0.2
0.0


Structures Generated Throughout Optimization: AChE
CO.Cc1ccc(P(C)(C)=O)c(Nc2nc3ccccc3nc2NS(=O)(=O)c2cn(C)cn2)c1.Cl
Score: 0.0000
I.O=C1CC(NC(=O)CC2=CC=CC3c4ccccc4C=CC23)=NCN1
Score: 0.0000
COCc1c2ccc(C3OCCO3)cc2c(C)c2cc(S(=O)(=O)N3CCC4(CC3)OC(=O)NC4C)ccc12
Score: 0.7190
Cc1cc(-c2ccccc2)c2cc(S(=O)(=O)C3CCC4(CC3)NC(=O)OC4C)ccc2c1
Score: 0.7635
O=C(CCC1=CC=C2C(c3cccc3)=C3C(=CCc4ccccc43)C2S1)NC1=NCCO1
Score: 0.7877
Cc1ccc2c(C)c(S(=O)(=O)C3CCC4(CC3)CC(=O)OCN4C)ccc2c1
Score: 0.7975
Cc1ccc2cc(S(=O)(=O)C3(O)CCC4(CC3)NC(=O)OC4C)ccc2c1
Score: 0.8045
C=C1NC(=O)CC12[CH]CC1(C=CC(S(=O)(=O)c3cc c4cc(C)ccc4c3C)=CC1)C(=O)C2
Score: 0.8255
Cc1ccc2c(C)c(S(=O)(=O)C3CCC4(CC3)N(C)C(=O)NC43NC3=O)ccc2c1
Score: 0.8295
CC1CN=C(NC(=O)C2CCc3ccccc3C23c2ccccc2C3C)O1
Score: 0.8338
Cc1ccc2cc(S(=O)(=O)C3CCC4(CC3)NC(=O)OC43C=C3)ccc2c1
Score: 0.8412
Cc1cc(N)c2cc(S(=O)(=O)C3=CCC4(C=C3)CC(=O)NC4C)ccc2c1
Score: 0.8488
CSC1=NC(NC(=O)Cc2ccc c(-c3cccc4c3C3=CC=C4C3)c2)COC1
Score: 0.8572
Cc1ccc2c(C)c(S(=O)(=O)C3=CCC4(CC3)NC(=O)CC43C=C3)ccc2c1
Score: 0.8674
O=C(Cc1ccc(C2CCc3ccc cc32)cc1)NC1=NCOCN1
Score: 0.8825
O=C(Cc1c#cc(C2C3=CC=CC3=Cc3ccccc32)cc1)NC1=NCCO1
Score: 0.9161
Score
1.0
0.8
0.6
0.4
0.2
0.0