Small Molecule Optimization with Large Language Models

Menua Bedrosian
YerevaNN
Yerevan State University

Philipp Guevorguian
YerevaNN
Yerevan State University

Tigran Fahradyan
YerevaNN
American University of Armenia

Gayane Chilingaryan
YerevaNN

Hrant Khachatrian
YerevaNN
Yerevan State University

Armen Aghajanyan

Abstract

The rise of large language models has created an opportunity for practical applications of machine learning algorithms in different areas like life science. In this work, we take advantage of the immense learning abilities of large language models and combine that with a training corpus of 110M small molecules to train a model that can predict molecular properties and more. More specifically, we take three publicly available large language models of 125M, 1B and 2B parameter sizes and train them on roughly 40B tokens comprising of molecules in SMILES format and their respective properties. These models demonstrate strong performance in generating molecules with specified properties and predicting new molecular characteristics from limited samples. We introduce a novel optimization algorithm that leverages our language models to optimize molecules for arbitrary properties given limited access to a black box oracle. Our approach combines ideas from genetic algorithms, rejection sampling, and prompt optimization. It achieves state-of-the-art performance on multiple molecular optimization benchmarks, including an 8% improvement on Practical Molecular Optimization compared to previous methods. We publicly release the language models and the dataset.

1 Introduction

Molecular optimization is a cornerstone of drug discovery, involving the complex task of identifying compounds with specific desirable properties. 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 need help with the vast and discrete nature of chemical space [Wu et al., 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]. 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 have often been 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

Foundation Models for Science Workshop,38th Conference on Neural Information Processing Systems (NeurIPS 2024).

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 introduce a new molecule optimization algorithm that unifies concepts from genetic algorithms, rejection sampling, and prompt optimization. This algorithm leverages our trained LLMs to efficiently navigate the vast chemical space, generating molecules with targeted properties.
3. Our approach demonstrates state-of-the-art performance on multiple molecular optimization benchmarks. On the challenging Practical Molecular Optimization (PMO) 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.
4. We illustrate the adaptability of our models through efficient fine-tuning for various molecular property predictions. 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.

2 Related Work

Language Models for Molecular Representation While graph-based representations are common for molecules, string-based representations, particularly Simplified Molecular Input Line Entry System (SMILES) [Weininger, 1988], have gained traction due to their compatibility with language models. This approach leverages the power of pre-trained language models and enables efficient processing of molecular data. Notable examples include ChemFormer [Irwin et al., 2022], MolT5 [Edwards et al., 2022], and BARTSmiles [Chilingaryan et al., 2024], 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, 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]. More recent methods leverage machine learning, particularly deep learning techniques. For instance, variational autoencoders [Kingma and Welling, 2013] have been applied to generate and optimize molecules in latent space, such as [Gómez-Bombarelli et al., 2018] and [Jin et al., 2018]. The GFlowNets [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.

Recurrent Neural Networks in Molecular Design Recurrent neural networks (RNNs) have also been applied to molecular optimization. A notable example is REINVENT [Olivecrona et al., 2017], which uses policy-based reinforcement learning to generate molecules with desired properties. Recent enhancements to REINVENT, such as augmented memory and Beam Enumeration [Guo and Schwaller, 2023b], have further improved its performance. These approaches combine molecular diversity filters, experience replay mechanisms, and substructure filtering to increase sample efficiency in molecular optimization tasks.

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., 2023] developed a

general evolutionary algorithm using language models, optimizing task-specific prompts for various downstream applications. 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.

3 Training Corpus

Molecular Database from PubChem We constructed a comprehensive SQL database using PubChem dumps, encompassing information on molecules, similar molecule pairs, experimental properties, and bioassays. 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, 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 widely used methods.

JSONL Corpus Generation We transformed our database into a corpus of JSONL files, with each molecule represented as a single JSON object. Below is an abbreviated example for aspirin:

```
[WEIGHT]180.16[/WEIGHT][TPSA]63.60[/TPSA][CLOGP]1.31[/CLOGP]
[START_SMILES]CC(=O)OC1=CC=CC=C1C(=O)O[END_SMILES]
[SAS]1.58[/SAS][QED]0.92[/QED]
[SIMILAR]O=C(Oc1ccccc1C(=O)O)c1ccccc1O 0.59[/SIMILAR]
[PROPERTY]Vapor Pressure 2.52X10-5 mm Hg at 25 °C (calc)[/PROPERTY]
```

This representation includes molecular identifiers, computed properties, similarity data, synonyms, experimental properties, and the PubChem compound identifier (CID).

Text Generation Template We developed a template system using paired tags to delimit each property and data point. For instance, a molecule's QED value is represented as `[QED]0.84[/QED]`. To enhance the model's versatility in both property prediction and property-conditioned molecular generation, we randomized the property order and alternated the position of the primary molecule (start vs. in-between other tags) with equal probability.

This carefully curated and structured corpus forms the foundation for training our language models, enabling them to learn complex relationships between molecular structures and properties.

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

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

	QED		SIM		SAS	
	PP	CG	PP	CG	PP	CG
Chemlactica-125M	0.016	0.101 (0.108)	0.046	0.183	0.078	**0.315 (0.379)**
Chemlactica-1.3B	**0.004**	**0.050 (0.050)**	**0.043**	0.167	0.066	0.400 (0.400)
Chemma-2B-2.1B	0.016	0.100 (0.100)	0.049	**0.126**	0.073	0.384 (0.382)
Chemma-2B-39B	**0.004**	0.075 (0.075)	0.046	0.140	**0.037**	0.415 (0.415)

	CLOGP		TPSA		WEIGHT	
	PP	CG	PP	CG	PP	CG
Chemlactica-125M	0.106	0.568 (0.568)	1.322	**5.216 (5.244)**	9.350	30.276 (30.276)
Chemlactica-1.3B	0.100	**0.405 (0.405)**	**0.893**	5.543 (15.640)	3.576	**16.877 (16.877)**
Chemma-2B-2.1B	0.137	1.675 (1.675)	1.638	7.077 (7.077)	8.962	39.695 (41.109)
Chemma-2B-39B	**0.034**	0.461 (0.461)	0.959	6.942 (6.942)	**1.931**	18.933 (20.395)

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].

Tokenization and Sample Preparation 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).

Training Methodology Both Chemma and Chemlactica were trained using the Adam optimizer [Kingma and Ba, 2014] with cross-entropy loss and a causal language modeling objective. We applied dropout only to Chemlactica, maintaining consistency with the original model architectures. Chemma-2B was trained in full bfloat16 for computational efficiency. 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 at Yerevan State University (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.

4.1 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 [END_SMILES] [QED], where M_i 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.

Table 1 presents the results for both Property Prediction (PP) and Conditional Generation (CG) across various properties for our three model variants. For Chemma-2B, we provide evaluations at different

training data volumes, including a compute-controlled run with 2.1B tokens to ensure fair comparison with Chemlactica-125M.

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.

Our generation process incorporates several 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 [Keskar et al., 2019].
- **Undesired Token Suppression:** Employed to ensure the model eventually generates [START_SMILES].

Table 7 provides an ablation study of these sampling components across our three models, demonstrating their individual and combined impacts on generation quality. Surprisingly, the best combinations of hyperparameters coincide for all three models.

These experiments comprehensively show our models' capabilities in predicting molecular properties and generating molecules with specified properties. These are crucial tasks in computational drug discovery and molecular design.

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 constraint set of valid molecules (typically very large), 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 can be viewed as a genetic algorithm where traditional crossover/mutation operations are replaced by language model generation. For S randomly selected molecules from the pool, we generate a new molecule using the prompt:

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

This approach allows for more intelligent exploration of the chemical space compared to traditional mutation operators.

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. This is done using 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 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 at this moment
  $v^{prop} \sim \mathcal{U}(v^{max}, oracle\_max)$
  $suffix \leftarrow$ [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] eos
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 $concat(p, suffix)$

Algorithm 2 presents our complete optimization procedure, which includes 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 language model when progress stagnates. Algorithm 1 details our prompt construction process, which is crucial for effective molecule generation and model fine-tuning.

We employ a dynamic fine-tuning strategy to adapt the language model throughout the optimization process. Fine-tuning is triggered if the best molecule doesn't 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. Detailed methodology and results of our hyperparameter tuning experiments are provided in Appendix A.1.

By combining these elements, our algorithm effectively leverages the power of large language models for molecular optimization, demonstrating 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 setting Gao et al. [2022] propose a 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 the molecular optimization algorithms. To assess the optimization ability and sample efficiency, Gao et al. [2022] put a limit on the number of oracle calls for each task to be 10000 and report the area under the curve (AUC) of the top-10 average property

Algorithm 2 molecular_optimization

Input: P, S, N, K
Initialize an empty $Pool \leftarrow \{\}$
while optim. problem stopping condition **do**
  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
end while

Table 2: 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 $\uparrow \pm$ 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	$\mathbf{2.555 \pm 0.099}$	$\mathbf{17.534 \pm 0.214}$

value versus the number of oracle calls as the performance metric. AUC values are calculated after every 100 oracle call, then combined and normalized to map the $[0, 1]$ range.

Our approach. Using our proposed optimization algorithm we evaluate Chemlactica-125M, Chemlactica-1.3B and Chemma-2B models. The hyperparameters for the optimization algorithm are tuned for each model separately according to the hyperparameter tuning methodology. For this benchmark, we use the bfloat16 data type for the language model's parameters.

Results. Our method performs strongly, surpassing the existing approaches. Our algorithm powered by the smallest Chemlactica-125M model 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 the optimization dynamics, Figures 3-5 illustrate visualizations of the optimization processes for `sitagliptin_mpo` task with different seeds for different models.

Note that, unlike most of the 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.6 we show that such rich prompts can significantly improve the metrics on several PMO tasks.

6.2 Multi-property Optimization with Docking

Problem formulation. This benchmark, initially proposed in the REINVENT paper [Blaschke et al., 2020], evaluates a model's capability to generate viable molecules for practical drug discovery. Specifically, it assesses the model's ability to generate plausible molecules that optimize docking scores (minimize docking energy) against specified protein targets. The benchmark focuses on three targets with extensive real-world applications: the dopamine type 2 receptor (DRD2), MK2-kinase, and acetylcholinesterase. 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.

The primary objective is to maximize the reward function with minimal oracle calls, emphasizing sample efficiency. We quantify this 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 predefined reward threshold. At the same time, generative yield represents the number of unique molecules generated above a reward threshold for a fixed number of oracle calls. To maintain consistency with recent implementations, we adopt the molecular preprocessing, conformational generation, docking parameters, and aggregate reward function from the Beam Enumeration paper, specifically comparing our results with the beam structure 15 methods, which demonstrated superior average-case performance.

Results. We used the exact same hyperparameters as those selected in the PMO experiment. Table 3 presents our approach's performance on this benchmark, simulating real-world drug design scenarios. Chemma-2B consistently achieves the highest performance for the generative yield metric across all evaluated receptors. Conversely, Chemlactica-125M demonstrates superior performance in terms of oracle burden, except for MK2 at oracle burden 1, where Chemma outperforms it. Notably, Chemlactica-1.3B achieved even better yield scores on the DRD2 target. Appendix A.9 shows the set of molecules generated at the beginning and at the end of the optimization trajectory for DRD2 docking.

These results suggest that model size is crucial in balancing exploration and exploitation of the molecular space. Smaller models appear more adept at initial space exploration, while larger models excel in exploiting the reward space. This trade-off between oracle burden and generative yield could have significant implications for applied drug design, particularly when access to oracle functions is limited or costly.

Our findings validate the effectiveness of our approach, demonstrating that our models can leverage pre-training information and selective fine-tuning to optimize complex reward functions, even with limited data unseen during pre-training. Furthermore, the successful transfer of training parameters and sampling strategies from the molecular optimization benchmark to this task underscores our method's flexibility and robustness. This adaptability suggests that our approach could be particularly valuable in scenarios where extensive hyperparameter tuning is impractical or undesirable.

6.3 QED Maximization with Similarity Constrained Molecular Design

Problem formulation. The objective of this optimization problem is to generate a molecule that has a high QED and is similar to some given molecule. More formally, given a molecule M, the objective of the problem 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 performance is measured by 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 and the generated molecule. Additionally, we decreased the maximum number of QED evaluations to 10000, compared to the baselines, which used 50000.

Table 3: Drug discovery case studies via docking function reward optimization. All experiments were 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 parentheses for oracle burden indicate how many unique molecules need to be generated for consideration. The best performance on each task-metric combination is bolded. Note that the hyperparameters of our models are not tuned for this task; instead, we used the best-performing hyperparameters on 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

Table 4: Performance comparison of different algorithms on QED and Similarity constrained molecular optimization problem.

	Success Rate (%) ↑
QMO	92.8
RetMol	94.5
Chemlactica-125M	**99.0**

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 5 times less QED evaluations at maximum. Since the performance of the Chemlactica-125M is very close to perfect, we have not evaluated other models for this task. Table 4 illustrates the performance of different algorithms.

7 Conclusion

This paper presents three language models: Chemlactica-125M, Chemlactica-1.3B, and Chemma-2B. These models were trained on a novel corpus encompassing over 100 million molecules and their properties. We demonstrate the efficacy of these models on multiple tasks in chemistry research, 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 and further research in this area. While our work demonstrates promising results in molecular optimization and related tasks, we acknowledge that it represents an early step in applying language models to chemical research. We hope our contributions will provide a valuable foundation for future work in this domain, potentially enabling new molecular design and analysis approaches.

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.

T. Blaschke, J. Arús-Pous, H. Chen, C. Margreitter, C. Tyrchan, O. Engkvist, K. Papadopoulos, and A. Patronov. Reinvent 2.0: an ai tool for de novo drug design. *Journal of chemical information and modeling*, 60(12):5918–5922, 2020.

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. 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.

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, 2023. URL `https://api.semanticscholar.org/CorpusID:262012566`.

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.

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 J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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.

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.

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

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.

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.

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.

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`.

A Appendix

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 very limited understanding of other biological entities like proteins, which constrains their practical applicability in certain areas of biochemistry and drug discovery. While effective, the optimization algorithms presented in this paper have not been exhaustively tuned, suggesting potential room for improvement. Additionally, our current approach does not account for synthetic accessibility or other practical considerations in drug design, which may limit its immediate applicability in real-world drug discovery pipelines.

Broader Impact

The molecular optimization models 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. Additionally, 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.1 Hyperparameters

Table 5 lists the hyperparameters we used for pretraining the language models.

For supervised fine-tuning we did a grid search over the following hyperparameters: peak learning rate, number of epochs, warmup steps and the amount of Neftune noise. Table 6 shows the best values for all tasks and models. Warmup steps are written as a ratio of the total training steps here.

Table 5: 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

Methodology for Hyperparameter Tuning of the Optimization Algorithm Given the large number of hyperparameters in our optimization algorithm, we adopt a two-step 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.

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. The best-performing configuration is then applied across all benchmarks in our evaluation. Notably, we tune the hyperparameters separately for Chemlactica-125M, Chemlactica-1.3B, and Chemma-2B to account for model-specific optimal settings.

A key hyperparameter, N, which determines the number of molecules generated before updating the pool, is set to 200. We employ vanilla temperature sampling for molecule generation throughout the optimization process. To address the need for generating thousands of unique molecules in many optimization benchmarks, we implement a dynamic temperature scheduling strategy. 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.

Grid search. 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.2 Model Calibration

A.2.1 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]`

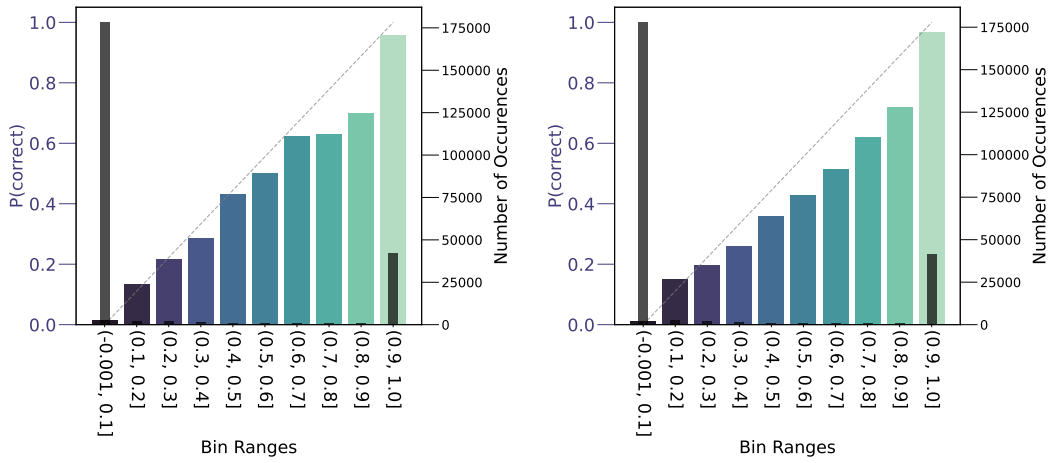
followed by five potential continuations, with only one being correct. This methodology is inspired by the calibration analysis in the GPT-4 technical report [OpenAI, 2023], which highlights calibration as a key indicator of high-quality pretraining.

Table 6: 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	0.0	20	10	5.0e-5	0.4	10	10	2.0e-4	0.0	10	10
HLM	1.0e-4	0.4	10	5	1.0e-5	0.4	10	10	1.0e-4	0.4	10	10
MD1	1.0e-4	0.4	15	0	5.0e-5	0.4	10	10	2.0e-4	0.4	10	0
hPPB	1.0e-4	0.4	10	0	1.0e-5	0.0	10	0	2.0e-4	0.4	10	10
rPPB	2.0e-4	0.0	10	5	5.0e-5	0.0	10	5	2.0e-4	0.4	20	0
Sol	2.0e-4	0.4	15	0	5.0e-5	0.0	20	0	2.0e-4	0.0	15	5
freesolv	2.0e-4	0.0	15	0	5.0e-5	0.0	15	5	2.0e-4	0.4	15	5
esol	5.0e-4	0.4	20	0	1.0e-5	0.0	10	5	2.0e-4	0.0	15	5
lipo	5.0e-4	0.4	10	5	1.0e-5	0.4	10	10	2.0e-4	0.4	10	10

Table 7: 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 prediction task.

			Chemlactica-125M		Chemlactica-1.3B		Chemma-2B	
CoT	rep.	supp.	RMSE (c) ↓	Invalids ↓	RMSE (c) ↓	Invalids ↓	RMSE (c) ↓	Invalids ↓
No	1.0	No	70.02 (70.02)	0/100	15.41 (65.22)	1/100	16.56 (65.58)	1/100
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



(a) Calibration of Chemma-2B.

(b) Calibration of Chemlactica-125M.

Figure 1: Model calibration on synthetic multiple choice question where y=x represents perfect calibration.

For each response, we calculated the model's predicted probability based on 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.

A.2.2 Results

Figures 1a and 1b 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 various molecular prediction and generation tasks.

Table 8: 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	**0.270 ± 0.011**	0.306 ± 0.011	0.533 ± 0.009	0.369 ± 0.000
Chemlactica-1.3B	0.281 ± 0.005	0.356 ± 0.009	0.557 ± 0.021	0.403 ± 0.013
Chemma-2B	0.298 ± 0.014	0.359 ± 0.040	0.563 ± 0.004	0.406 ± 0.012

Table 9: Regression tasks from the ADMET benchmark. All numbers are Pearson correlation ↑.

	HLM	MDR1-MDCK ER	Solubility
MPNN2 (from the original paper)	0.68	**0.78**	**0.59**
Chemlactica-125M	**0.68 ± 0.011**	0.77 ± 0.012	0.57 ± 0.035
Chemlactica-1.3B	**0.68 ± 0.004**	0.77 ± 0.009	0.54 ± 0.043
Chemma-2B	0.67 ± 0.004	**0.78 ± 0.009**	0.53 ± 0.024

	RLM	hPPB	rPPB
MPNN2 (from the original paper)	**0.74**	**0.77**	**0.70**
Chemlactica-125M	0.71 ± 0.004	0.73 ± 0.004	0.60 ± 0.098
Chemlactica-1.3B	0.65 ± 0.004	0.74 ± 0.001	0.62 ± 0.017
Chemma-2B	0.68 ± 0.005	0.75 ± 0.004	0.60 ± 0.030

A.3 Property Prediction

Supervised fine-tuning recipe. We designed and implemented a fine-tuning strategy to evaluate our model's adaptability to novel tasks not present in the initial training corpus. To this end, we fined-tuned our models on 6 tasks introduced by Fang et al. [2023a] and 3 others by MoleculeNet Wu et al. [2018]. Inspired by instruction tuning methodologies, we generated a specialized training corpus formatted as follows:

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

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 $10e-4$ with 3 epochs of warmup and a NEFTune noise [Jain et al., 2023] of 5. However, we observed that our models could significantly benefit from a more rigorous hyperparameter optimization process. Consequently, 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 epoch ratios: [0, 0.4, 1], NEFTune noise : [0.0, 5.0, 10.0]. The results presented in Table 8 and 9 showcase the abilities of our models after the hyperparameter tuning stage. The details of hyperparameters selected per task and model can be found in the Appendix A.1.

Results. Table 8 lists the results for three regression tasks from MoleculeNet [Wu et al., 2018]. Fang et al. [2023b] introduces a new dataset for six ADMET targets. The authors provided training/test split but no validation set. We used a random 20% of the training set as a validation set to pick the best hyperparameters. Table 9 shows the results.

Table 10: 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	$\mathbf{0.951 \pm 0.011}$	0.947 ± 0.012	$\mathbf{0.951 \pm 0.009}$
amlodipine_mpo	0.635 ± 0.035	0.691 ± 0.047	0.761 ± 0.019	$\mathbf{0.772 \pm 0.091}$	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	$\mathbf{0.920 \pm 0.011}$
deco_hop	0.666 ± 0.044	0.658 ± 0.024	0.733 ± 0.109	0.801 ± 0.101	$\mathbf{0.836 \pm 0.117}$	0.831 ± 0.123
drd2	0.945 ± 0.007	0.963 ± 0.006	$\mathbf{0.974 \pm 0.006}$	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	$\mathbf{0.931 \pm 0.014}$
gsk3	0.865 ± 0.043	0.881 ± 0.021	0.881 ± 0.042	0.926 ± 0.022	0.916 ± 0.027	$\mathbf{0.928 \pm 0.021}$
isomers_c7h8n2o2	0.852 ± 0.036	0.853 ± 0.087	0.969 ± 0.003	$\mathbf{0.951 \pm 0.012}$	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	$\mathbf{0.929 \pm 0.012}$	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	$\mathbf{0.891 \pm 0.032}$
median1	0.356 ± 0.009	0.326 ± 0.013	0.379 ± 0.010	0.359 ± 0.060	$\mathbf{0.382 \pm 0.047}$	$\mathbf{0.382 \pm 0.022}$
median2	0.276 ± 0.008	0.291 ± 0.008	0.294 ± 0.007	0.328 ± 0.032	0.329 ± 0.016	$\mathbf{0.366 \pm 0.018}$
mestranol_similarity	0.618 ± 0.048	0.750 ± 0.049	0.708 ± 0.057	0.896 ± 0.064	0.850 ± 0.051	$\mathbf{0.926 \pm 0.023}$
osimertinib_mpo	0.837 ± 0.009	0.855 ± 0.004	0.860 ± 0.008	$\mathbf{0.907 \pm 0.015}$	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	$\mathbf{0.755 \pm 0.066}$	0.711 ± 0.062
qed	0.941 ± 0.000	$\mathbf{0.942 \pm 0.000}$	$\mathbf{0.942 \pm 0.000}$	$\mathbf{0.942 \pm 0.000}$	$\mathbf{0.942 \pm 0.000}$	0.941 ± 0.000
ranolazine_mpo	0.760 ± 0.009	0.801 ± 0.006	0.819 ± 0.018	0.864 ± 0.014	$\mathbf{0.883 \pm 0.017}$	0.868 ± 0.015
scaffold_hop	0.560 ± 0.019	0.567 ± 0.008	0.615 ± 0.100	0.626 ± 0.016	$\mathbf{0.673 \pm 0.080}$	0.669 ± 0.110
sitagliptin_mpo	0.021 ± 0.003	0.284 ± 0.050	0.634 ± 0.039	$\mathbf{0.649 \pm 0.051}$	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	$\mathbf{0.698 \pm 0.121}$
troglitazone_rediscovery	0.441 ± 0.032	0.540 ± 0.048	0.511 ± 0.054	0.734 ± 0.130	0.765 ± 0.138	$\mathbf{0.824 \pm 0.049}$
valsartan_smarts	$\mathbf{0.178 \pm 0.358}$	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	$\mathbf{0.608 \pm 0.055}$
sum	14.196	15.002	16.213	17.170 ± 0.424	17.284 ± 0.284	$\mathbf{17.534 \pm 0.214}$

Table 11: 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

A.4 Detailed Results for Practical Molecular Optimization

Table 10 shows the evaluations of Chemlactica-125M, Chemlactica-1.3B and Gemma-2B, along with other methods on 23 tasks of the PMO benchmark. There is no method that uniformly beats all others on every task. None of our (and many other) methods get non-zero result on `valsartan_smarts`. The reason is that the oracle has a binary multiplier term that is usually equal to zero, so there is no supervision signal for the entire generation process.

A.5 Ablation Study on the Optimization Algorithm

A key component of our proposed optimization algorithm is the fine-tuning step, which is activated 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`. These tasks were chosen to provide a diverse set of challenges and to be representative of the broader benchmark.

Table 11 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 6 through 8. 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.

Table 12: 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 13 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 13: The descriptions of tasks used in the prompts in the extended version of our optimization algorithm. The results are in Table 12. See Section A.6 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]

A.6 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 2 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. We selected four tasks from the PMO benchmark: `jnk3`, `median1`, `sitagliptin_mpo`, and `scaffold_hop`. For these tasks, we modified Algorithm 1 to incorporate relevant known properties into the prompt p between steps 2 and 3.

Table 12 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 13. Notably, no additional properties were added for the `jnk3` task as our models lack specific knowledge about its oracle function.

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's important to note that while this approach showcases the potential of our models, it may not provide a fair comparison with methods that don't have access to such property information.

A.7 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. However, in the context of 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.

Challenges in Batched Generation Molecular optimization pipelines require repeated model calls for generation, followed by oracle function scoring. While batched processing accelerates this process

Table 14: Impact of numerical precision on multi-property optimization with docking task.

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

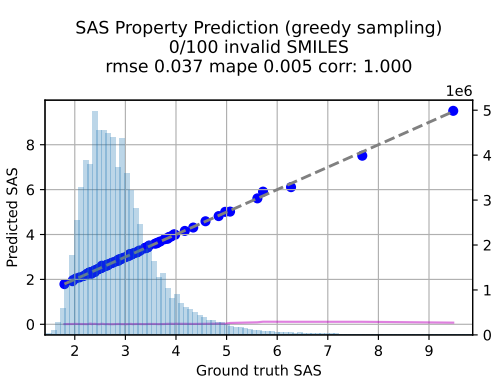
through GPU parallelization, it introduces complications. The necessary padding for batch processing alters matrix sizes, affecting multiply-accumulate operations within the model. These small errors accumulate as they propagate through the model's layers. Lower precision exacerbates these errors, leading 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 used for both additional fine-tuning and identifying similar molecules to guide optimization. However, when lower precision leads to sub-optimal molecule generation, it creates a negative 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 particularly adverse effects of low precision in molecular optimization pipelines.

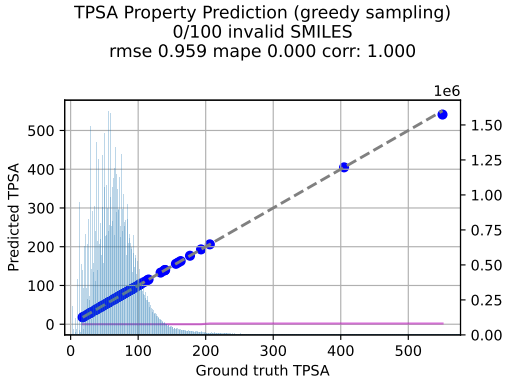
Precision Ablation Study To quantify the impact of numerical precision on the optimization process, we conducted an ablation study comparing 32-bit floating point precision with bfloat16 precision. Table 14 presents the results of this comparison across all drug discovery case studies described in Section 6.2. Despite the potential computational costs, these results demonstrate the critical importance of maintaining higher numerical precision in molecular optimization tasks.

A.8 Visualization of the Model Outputs on Property Prediction and Conditional Generation Tasks

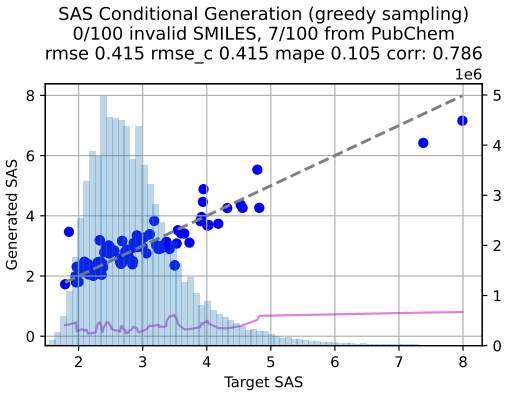
Figures 2e-2e show the performance of Chemma-2B for property prediction and conditional molecular generations tasks. Each dot in the scatter plot corresponds to one molecule. The histogram in the background is the actual distribution of those properties in the database. The purple line shows RMSE error for the given value of the property.



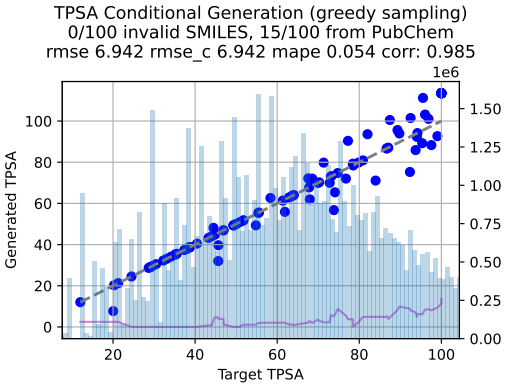
(a) SAS prediction.



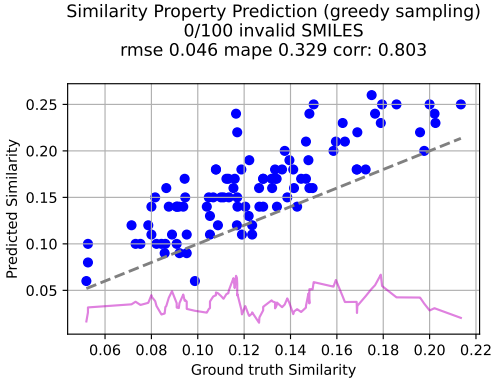
(b) TPSA Prediction.



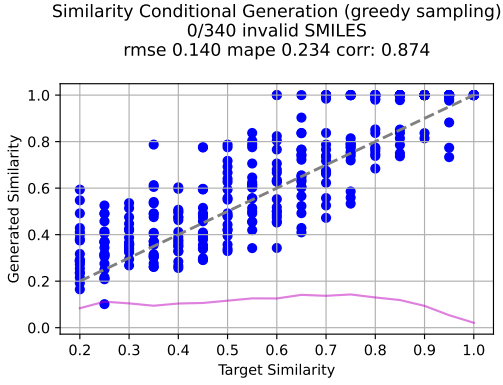
(c) SAS-conditioned generation of molecules.



(d) TPSA-conditioned generation of molecules.



(e) Prediction of similarity between two molecules.



(f) Similarity-conditioned generation of molecules.

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

Figure 3: Optimization process visualization using Chemlactica-125M model for `sitagliptin_mpo` task with four different seeds.

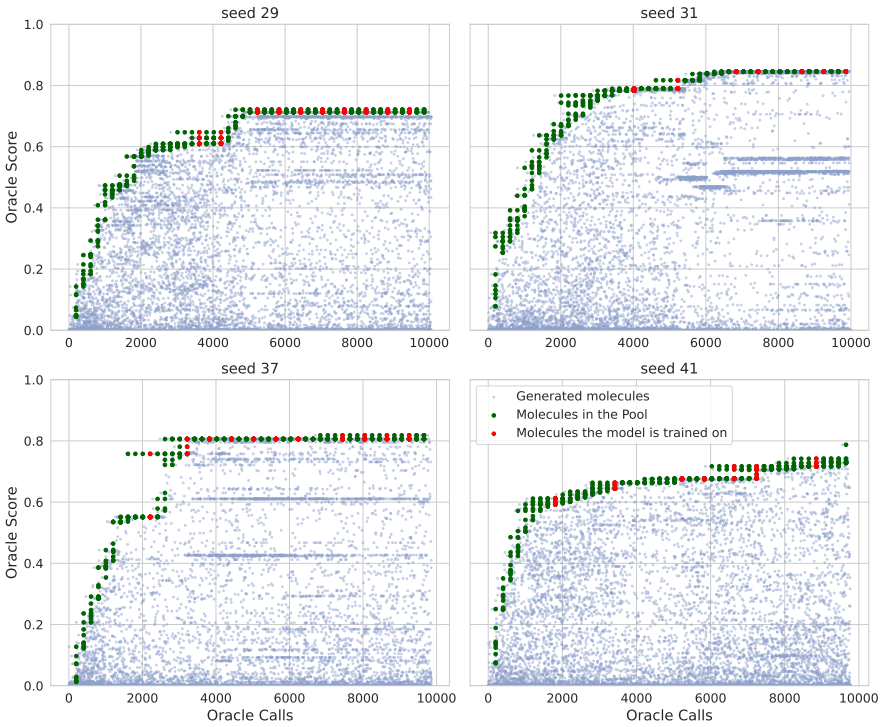


Figure 4: Optimization process visualization using Chemlactica-1.3B model for `sitagliptin_mpo` task with four different seeds.

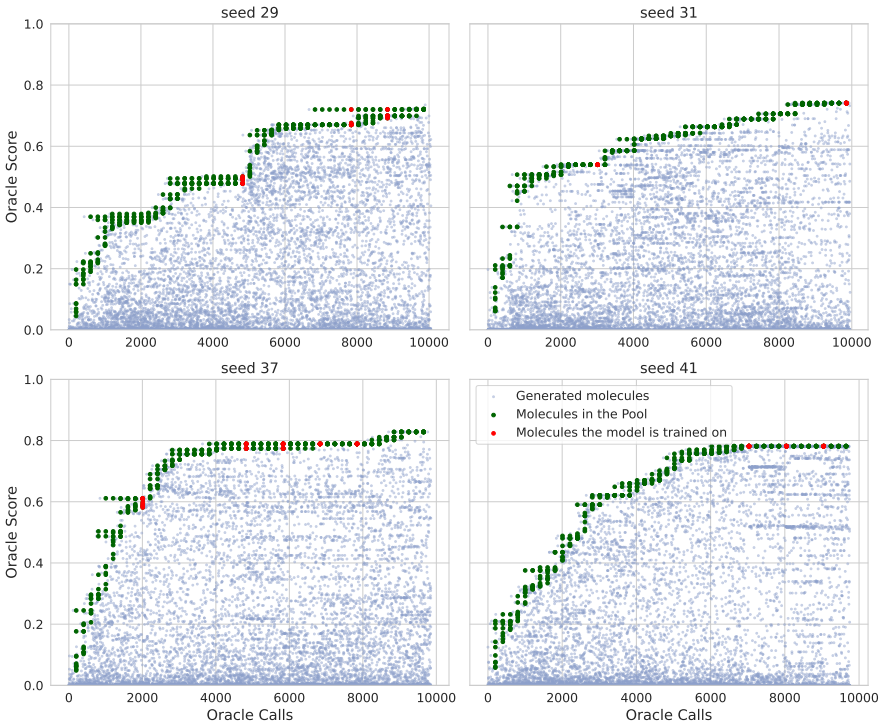


Figure 5: Optimization process visualization using Chemma-2B model for `sitagliptin_mpo` task with four different seeds.

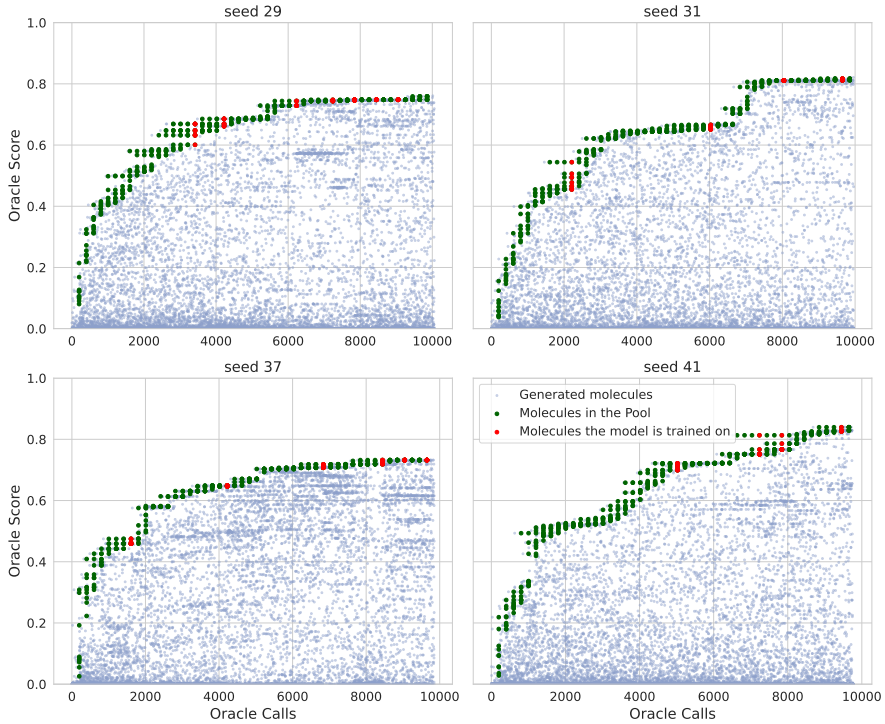


Figure 6: Mean oracle score $\pm$ standard deviation of the generated molecule for Chemlactica-125M.

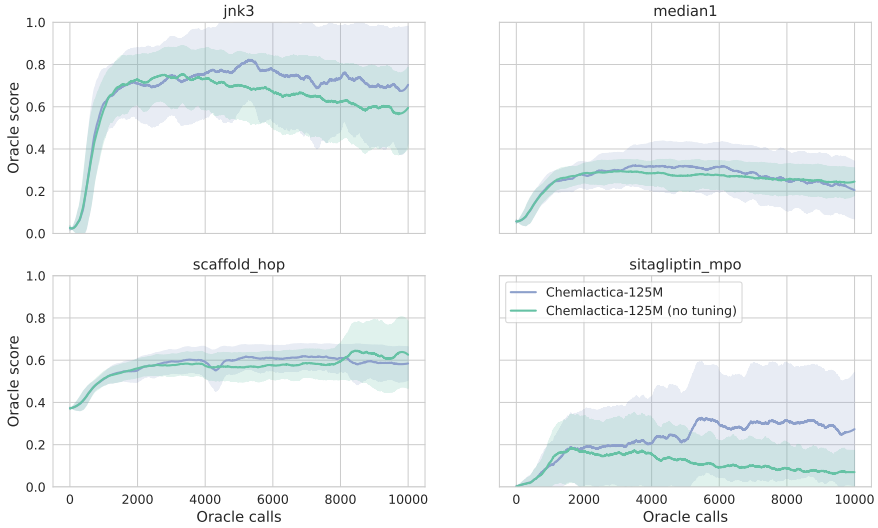


Figure 7: Mean oracle score ± standard deviation of the generated molecule for Chemlactica-1.3B.

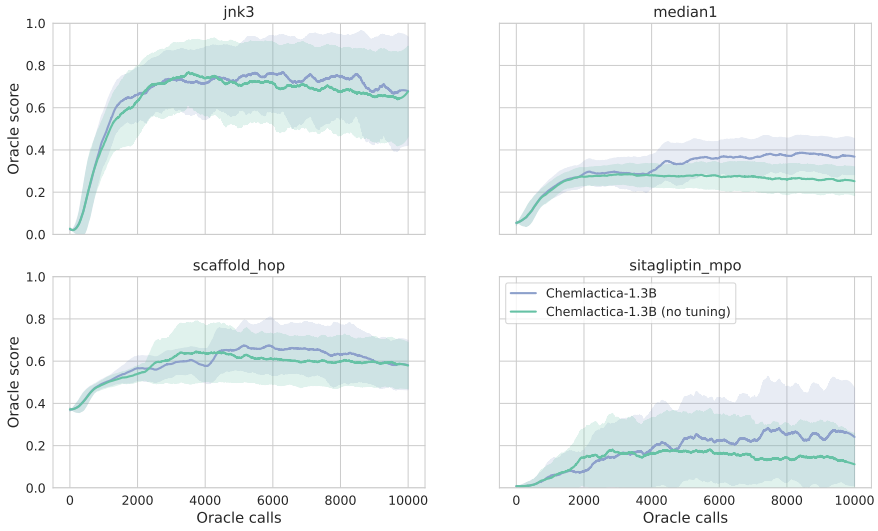
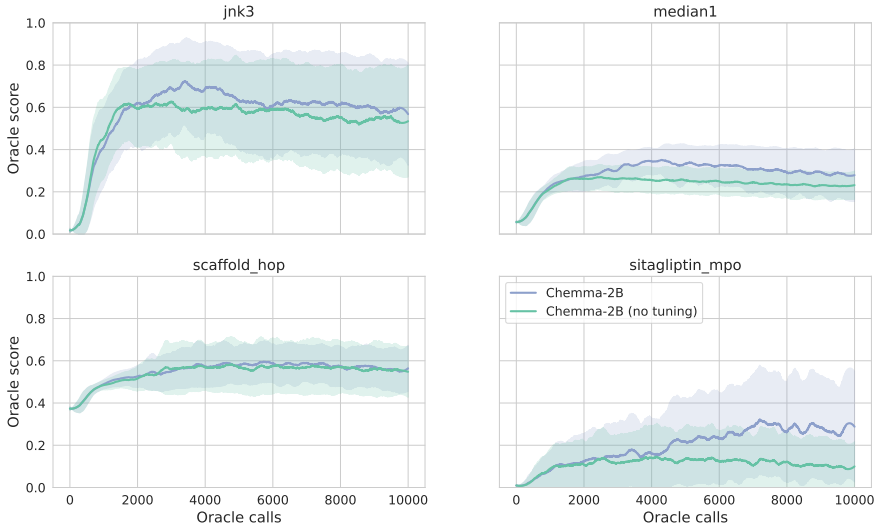
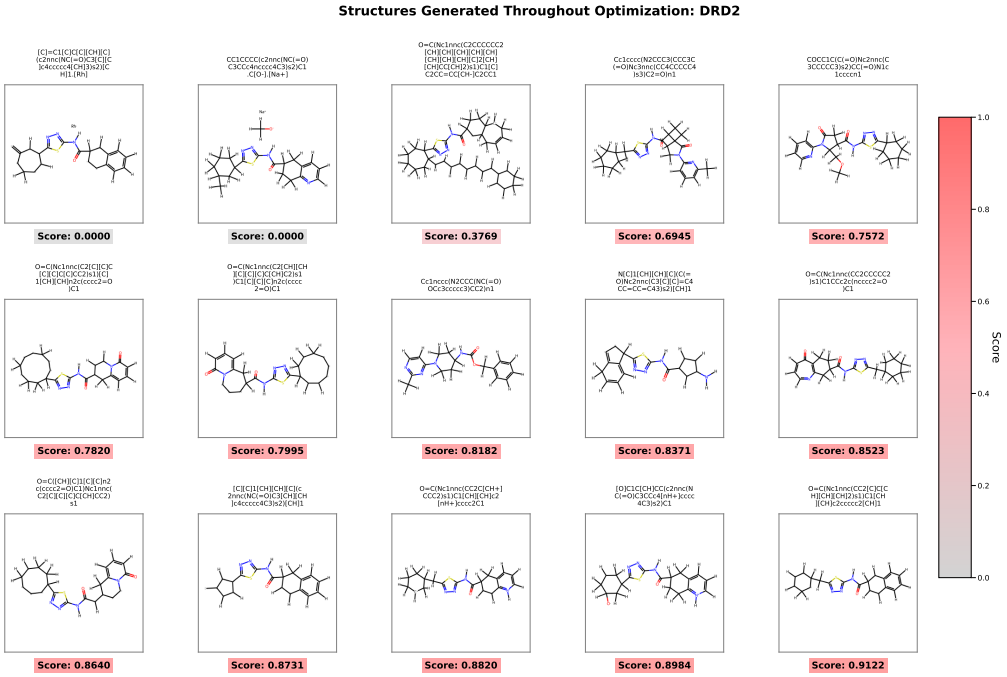


Figure 8: Mean oracle score ± standard deviation of the generated molecule for Chemma-2B.

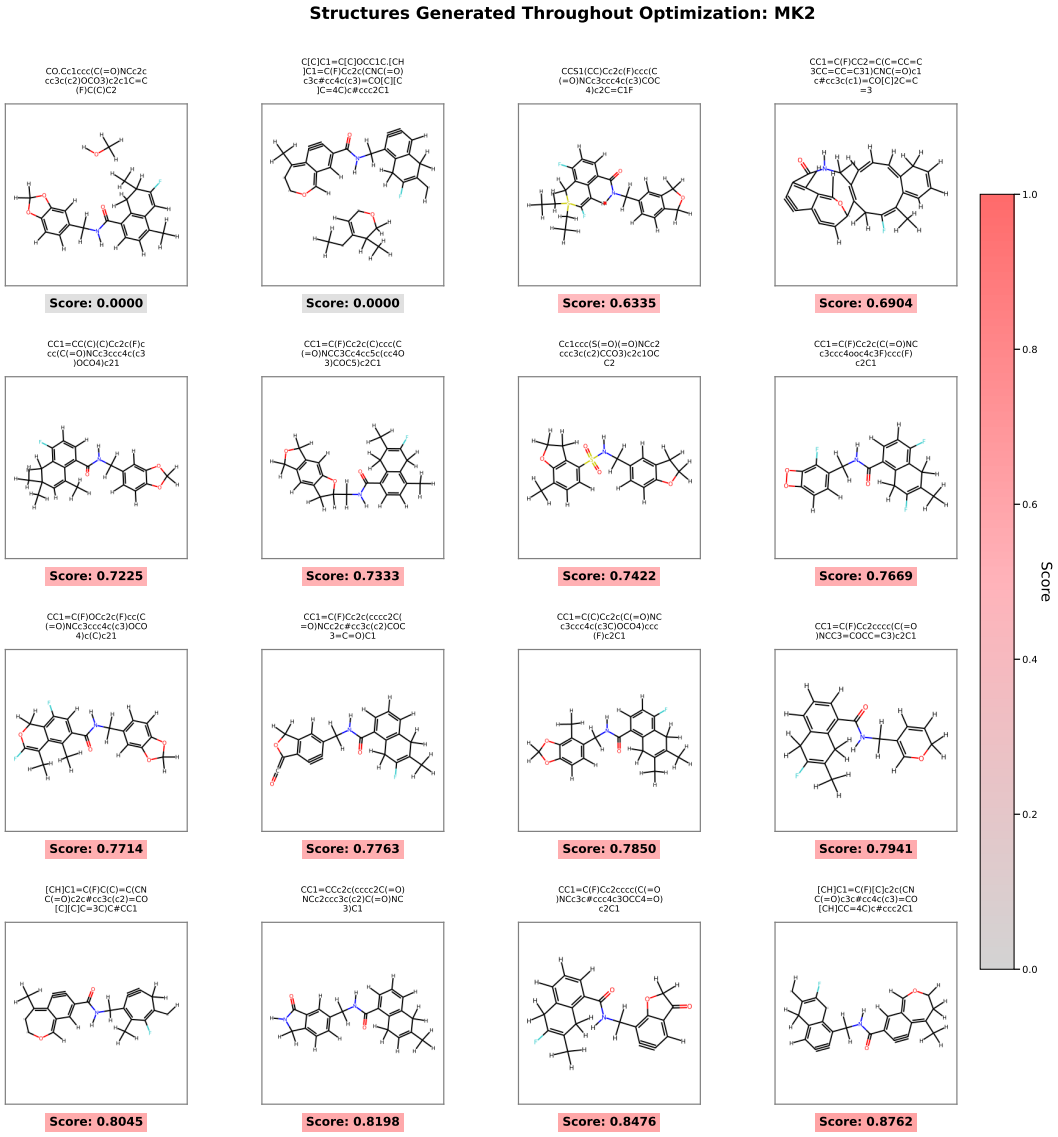


A.9 Generated Molecules in the Docking Experiments

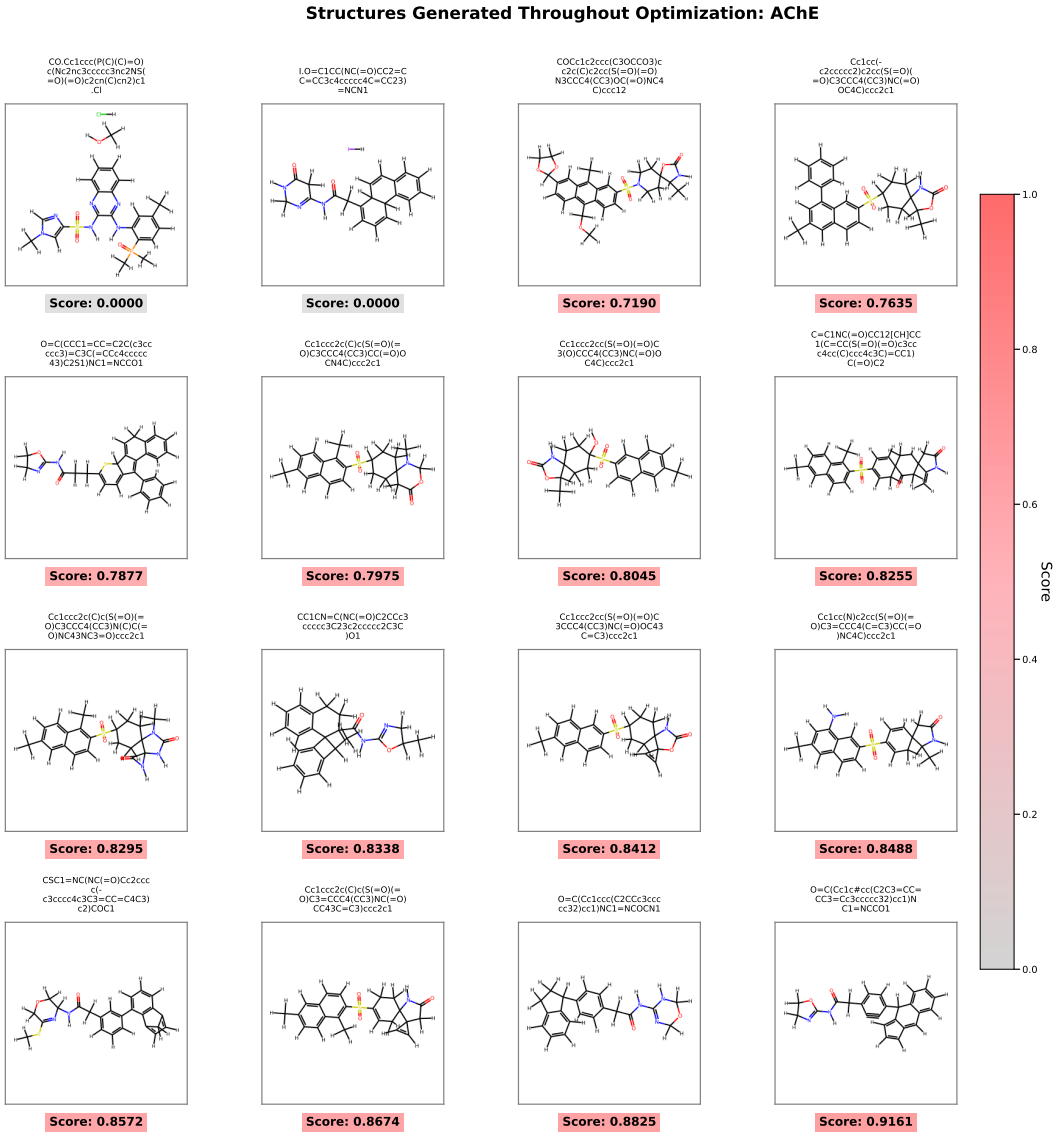
A.9.1 DRD2



A.9.2 MK2


Structures Generated Throughout Optimization: MK2
CO.Cc1ccc(C(=O)NCc2c cc3c(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)=C(O[C][C]C=4C)c#ccc2C1
Score: 0.0000
CCS1(CC)Cc2c(F)ccc(C (=O)NCc3ccc4c(c3)COC 4)c2C=C1F
Score: 0.6335
CC1=C(F)CC2=C(C=CC=C 3CC=CC=C31)CNC(=O)c1 c#cc3c(c1)=C(O[C]2C=C =3
Score: 0.6904
CC1=CC(C)(C)Cc2c(F)c cc(C(=O)NCc3ccc4c(c3)OCO4)c21
Score: 0.7225
CC1=C(F)Cc2c(C)ccc(C (=O)NCC3Cc4cc5c(cc4O 3)COC5)c2C1
Score: 0.7333
Cc1ccc(S(=O)(=O)NCc2 ccc3c(c2)CCO3)c2c1OC C2
Score: 0.7422
CC1=C(F)Cc2c(C(=O)NC c3ccc4ooc4c3F)ccc(F) c2C1
Score: 0.7669
CC1=C(F)OCc2c(F)cc(C (=O)NCc3ccc4c(c3)OCO 4)c(C)c21
Score: 0.7714
CC1=C(F)Cc2c(cccc2C(=O)NCc2c#cc3c(c2)COC 3=C=O)C1
Score: 0.7763
CC1=C(C)Cc2c(C(=O)NC c3ccc4c(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(CN C(=O)c2c#cc3c(c2)=CO [C][C]C=3C)C#CC1
Score: 0.8045
CC1=CCc2c(cccc2C(=O) NCc2ccc3c(c2)C(=O)NC 3)C1
Score: 0.8198
CC1=C(F)Cc2cccc(C(=O)NCc3c#ccc4c3OCC4=O) c2C1
Score: 0.8476
[CH]C1=C(F)[C]c2c(CN C(=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

A.9.3 AChE


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)ON4C)ccc2c1
Score: 0.7975
Cc1ccc2ccc(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)c3ccc4cc(C)ccc4c3C)=CC1)C(=O)C2
Score: 0.8255
Cc1ccc2c(C)c(S(=O)(=O)C3CCC4(CC3)NC(C(=O)NC43NC3=O)ccc2c1
Score: 0.8295
CC1CN=C(NC(=O)C2CCc3cccc3C23c2ccccc2C3CO1
Score: 0.8338
Cc1ccc2ccc(S(=O)(=O)C3CCC4(CC3)NC(=O)OC43C=C3)ccc2c1
Score: 0.8412
Cc1ccc(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(Cc1cccc(C2CCc3cccc32)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