LICO: LARGE LANGUAGE MODELS FOR IN-CONTEXT MOLECULAR OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing black-box functions is a fundamental problem in science and engineering. To solve this problem, many approaches learn a surrogate function that estimates the underlying objective from limited historical evaluations. Large Language Models (LLMs), with their strong pattern-matching capabilities via pretraining on vast amounts of data, stand out as a potential candidate for surrogate modeling. However, directly prompting a pretrained language model to produce predictions is not feasible in many scientific domains due to the scarcity of domain-specific data in the pretraining corpora and the challenges of articulating complex problems in natural language. In this work, we introduce LICO, a general-purpose model that extends arbitrary base LLMs for black-box optimization, with a particular application to the molecular domain. To achieve this, we equip the language model with a separate embedding layer and prediction layer, and train the model to perform in-context predictions on a diverse set of functions defined over the domain. Once trained, LICO can generalize to unseen molecule properties simply via in-context prompting. LICO achieves state-of-the-art performance on PMO, a challenging molecular optimization benchmark comprising over 20 objective functions.

1 INTRODUCTION

Black-box optimization (BBO) is the problem of optimizing an unknown, often complex objective function without direct access to its structure or derivatives. This problem is ubiquitous in many science and engineering fields, including material discovery (Hamidieh, 2018), protein engineering (Brookes et al., 2019; Sarkisyan et al., 2016; Angermueller et al., 2020), molecular design (Gaulton et al., 2012), mechanical design (Berkenkamp et al., 2016; Liao et al., 2019), and neural architecture search (Zoph & Le, 2016). Typically, BBO involves an iterative process where each step constructs a surrogate model to approximate the objective function. This model then guides the selection of promising candidates for subsequent evaluation. The main challenge of this approach lies in learning an effective surrogate function that can accurately estimate the objective from limited historical data.

In stark contrast, we have seen impressive generalization abilities of Large Language Models (LLMs) (Brown et al., 2020; Achiam et al., 2023; Bubeck et al., 2023; Team et al., 2023; Touvron et al., 2023a;b; Jiang et al., 2023; 2024) for language-driven reasoning over many kinds of domains. By pretraining on Internet-scale data, LLMs have demonstrated exceptional pattern-matching abilities and generalization from limited observations in both natural language (Brown et al., 2020; Kojima et al., 2022; Wei et al., 2022) and other domains (Lu et al., 2022; Mirchandani et al., 2023; Gruver et al., 2023). This positions LLMs as a promising solution for enhancing surrogate modeling for BBO. Some recent works have indeed shown great potential for using LLMs for solving optimization problems (Yang et al., 2023; Chen et al., 2023; Zhang et al., 2023; Liu et al.). The main idea behind these methods is to frame the optimization problem in natural language, and prompt the language model using previously collected observations to make predictions for new data points (Liu et al.) or to propose better candidates (Yang et al., 2023; Chen et al., 2023; Zhang et al., 2023; Ma et al., 2023; Nie et al., 2023; Meyerson et al., 2023; Lehman et al., 2023; Bradley et al., 2024; Liu et al., 2023a). However, this approach has several limitations. First, performing optimization in the text space requires the problem and solution to be expressed in natural language, thus limiting this approach to selected domains. Second, the scarcity of domain-relevant data in the text corpora used to train language models poses generalization challenges when using these models for general scientific domains such as molecular optimization. Therefore, existing works have only demonstrated the

success of LLMs in neural architecture search (Liu et al.; Chen et al., 2023; Zhang et al., 2023), prompt optimization (Yang et al., 2023), and code generation (Ma et al., 2023; Lehman et al., 2023), corresponding to domains that are well-represented in the training dataset for common language models (Brown et al., 2020; Touvron et al., 2023a; Jiang et al., 2023). Third, relying on verbose textual descriptions for both the problem and its solution imposes practical constraints by inflating the context length and thereby reducing the number of historical observations the model can effectively utilize.

In this work, we propose **L**arge Language Models for **I**n-**C**ontext **O**ptimization (LICO), a general-purpose model that leverages LLMs for black-box optimization, with a particular application to the molecular domain. To generalize a language model to a new scientific domain unseen during pretraining, we equip the model with two embedding layers for embedding the previously collected molecules and their scores, and a prediction head to predict the score of unseen candidates. Intuitively, the embedding layers map the molecules and their scores to the same feature space already learned by the language model, allowing the model to perform in-context learning in this space instead of the raw text space. Unlike previous methods, this approach is applicable to domains that may not be easily described in natural language such as molecular optimization. Moreover, avoiding verbose textual descriptions enables the model to condition on more historical observations, thus scaling better to harder problems that cannot be solved within a few steps.

We train the new layers together with the (frozen) LLM to perform in-context predictions on a family of functions. Specifically, for each function sampled from this family, we condition the model on a set of inputs and their corresponding evaluations, and task the model to predict the function value of the remaining data points. This task mimics surrogate modeling in BBO, where the surrogate model has to iteratively update its estimation of the underlying objective by conditioning on historical data. An ideal function family to train the model should be close to the target objective functions we want to optimize, but also be diverse enough to encourage generalization. Therefore, we propose to combine intrinsic functions and synthetically generated functions for training LICO. Intrinsic functions are inherent properties of the input that are easy to evaluate. In molecular optimization, for example, intrinsic functions include molecular weight, the number of rings, or heavy atom count, which are obtained via simple computation on the molecule. These intrinsic functions are closely related to the actual objective functions we want to optimize such as bioactivities against a target disease. To facilitate generalization outside of the intrinsic functions, we additionally train LICO on synthetic functions defined over the same target domain that are generated by Gaussian Processes. Our empirical evidence shows the importance of learning from both intrinsic and synthetic functions to the performance of the model on downstream tasks. Figure 1 illustrates our approach.

After training, LICO is capable of optimizing a wide range of molecular properties purely via in-context prompting. While the methodology of LICO applies to general scientific domains, in this paper we focus on molecular optimization. This problem plays a pivotal role in advancing drug and material discovery. The complexity of molecular structures and the vastness of the chemical space present unique challenges to black-box optimization algorithms. Moreover, since molecule-relevant data is likely under-represented in the pretraining corpora of existing language models, molecular optimization is a good problem to test the performance and applicability of LICO. We evaluate LICO against the state-of-the-art methods on Practical Molecular Optimization (PMO) (Gao et al., 2022), a challenging molecular optimization benchmark with over 20 objective functions. The experiments show that LICO achieves the best performance and is the highest-ranked method in the benchmark.

2 Problem Statement

Let $f : \mathcal{X} \to \mathbb{R}$ be a real-valued function that operates on a d-dimensional space $\mathcal{X} \subseteq \mathbb{R}^d$. In black-box optimization (BBO), the goal is to find the input $x^\star$ that maximizes f:

$$x^\star \in \arg\max_{x \in \mathcal{X}} f(x), \tag{1}$$

where we do not have direct access to the structure or gradient information of f. In molecular optimization, $\mathcal{X}$ is the space of all possible molecules, and f is a certain property of the molecule we want to optimize over, such as bioactivities against a disease. While f is unknown, we often have access to an unlabeled dataset $\mathcal{D}_\text{u}$ that consists of molecules $x's$ without the corresponding function values $y's$. ZINC (Sterling & Irwin, 2015) is such a dataset with thousands of unlabeled molecules.

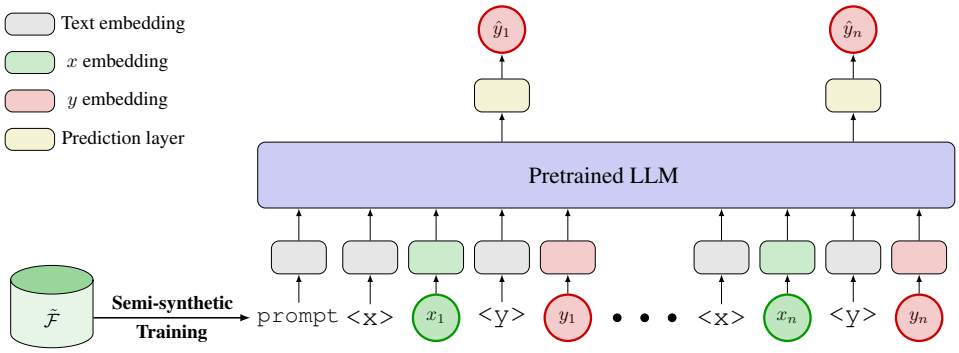

Figure 1: Our proposed approach. We equip a pretrained LLM with an embedding layer for x, an embedding layer for y, and a prediction layer. We train the model on semi-synthetic data to predict y given x and previous (x, y) pairs. We prepend each x with a special token <x> and each y with a special token <y> to guide in-context reasoning.

To solve the optimization in equation 1, we can query f with a limited budget, since evaluation often involves expensive physical experiments. To overcome this challenge, a common BBO approach learns a surrogate model f_θ that approximates the objective f from past observations $\mathcal{D}_{\text{obs}} = \{(x_i, y_i)\}_{i=1}^n$, which starts empty and incrementally expands with new data points $(x, f(x))$ we query at each iteration. Formally, a surrogate model represents a predictive distribution $p_\theta(y \mid x, \mathcal{D}_{\text{obs}})$ of the function value y conditioned on the input x and the evolving observed dataset $\mathcal{D}_{\text{obs}}$. The prediction of this surrogate guides the selection of candidates to balance exploration and exploitation. The newly selected points are added to $\mathcal{D}_{\text{obs}}$, and the process continues.

The success of this approach highly depends on the efficiency of the surrogate model f_θ in estimating f from limited data in $\mathcal{D}_{\text{obs}}$ at each iteration. This resembles few-shot prediction, a setting that Large Language Models (LLMs) have proven to excel in. By pretraining on vast Internet-scale data, LLMs can learn generalizable patterns from limited data, and are capable of adapting to multiple functions at test time simply via in-context prompting (Brown et al., 2020; Mirchandani et al., 2023; Krishnamoorthy et al., 2023a;b). A recent line of works (Yang et al., 2023; Zhang et al., 2023; Chen et al., 2023; Liu et al.) has exploited this ability of LLMs for optimization, but they relied on natural language as the interface, thus lacking generality to scientific domains. In this work, we propose a more general and efficient approach to leveraging LLMs for black-box optimization.

3 Related Work

LLMs for Optimization Recent works have explored the use of LLMs for optimization. The general idea behind these works is to prompt the model with the textual description of the optimization problem and historical evaluations for few-shot reasoning. Yang et al. (2023); Liu et al. (2023a); Zhang et al. (2023); Ma et al. (2023) propose to prompt the language model to directly suggest better candidates to evaluate given the past inputs and their corresponding scores. Meyerson et al. (2023); Lehman et al. (2023); Bradley et al. (2024) integrate LLMs with evolutionary algorithms, and prompt the model to perform crossover and mutation operations based on the population at each optimization step. Liu et al. study the use of LLMs to enhance several components in Bayesian optimization, including warmstarting, surrogate modeling, and candidate generation. Optformer (Chen et al., 2022) proposed to train an LLM specialized for in-context function prediction and optimization.

LLMs for Molecular Optimization Recent works have proposed to leverage LLMs for molecular optimization via prompting (Wang et al., 2024; Liu et al., 2023b; Ramos et al., 2023; Völker et al., 2024), leveraging LLM embeddings (Ranković & Schwaller, 2023), or finetuning on molecular corpora (Guevorguian et al., 2024; Ye et al., 2023; Fang et al., 2023; Kristiadi et al.). MOLLEO (Wang et al., 2024) and ChatDrug (Liu et al., 2023b) are two prominent works in the first direction. MOLLEO proposed to prompt a pretrained LLM to perform crossover and mutation operations in a standard graph genetic algorithm, but its performance largely depends on the prompt format. Similarly, ChatDrug prompts a pretrained LLM for drug editing and requires a retrieval database of molecules to inject domain feedback into the LLM. BoChemian (Ranković & Schwaller, 2023) and Kristiadi

et al. studied the use of LLM embeddings for Bayesian molecular optimization, and Kristiadi et al. additionally explored finetuning an LLM to serve as a surrogate for optimization. Chemlactica/Chemma (Guevorguian et al., 2024) and MOLGEN (Fang et al., 2023) proposed to pretrain and/or finetune LLMs on molecule-related corpora to generate valid molecules, which can serve as a genetic algorithm in molecular optimization. DrugAssist (Ye et al., 2023) creates the MolOpt-Instructions dataset that contains pairs of molecules and their property values to finetune a pretrained LLM that can iteratively propose better molecules after training.

The common approach in existing works has several inherent limitations. First, for general scientific domains, the input x may not be easily described by natural language. Second, even when there is a textual description of the input, for instance, molecules can be represented by SMILES strings (Weininger, 1988), existing prompt-based works require significant prompt optimization to achieve good performance, and the optimal prompt often varies between tasks. Furthermore, from an engineering perspective, naively prompting a language model with verbose textual descriptions of the input x results in an excessively long context, thus reducing the number of examples the model can condition on. For example, an LLM with a maximum context length of 4000 can only utilize up to 100 past observations, assuming the average length of each data point is 40. This practically limits the scalability of this approach to harder problems that require more steps to solve.

LLMs for Non-language Tasks In addition to optimization, several works have studied the extension of pretrained LLMs to non-language domains with two main directions. The first direction considers problems that can be described in natural language, and prompts a pretrained LLM to solve the problem directly in the text space (Mirchandani et al., 2023; Dinh et al., 2022; Gruver et al., 2023; Liu et al., 2024; Sprueill et al., 2024). The second direction tackles more general problems by learning separate encoders for the new domain and aligning it with the embedding space of the pretrained LLM (Lu et al., 2022; Shen et al., 2023; Tsimpoukelli et al., 2021; Li et al., 2022). Our work is closely related to the latter direction. However, as discussed in the following sections, while many of these works completely leave the word space, we find it beneficial to include language instruction while training the new modules.

4 METHOD

We introduce LICO, a methodology for extending arbitrary base LLMs for surrogate modeling in black-box optimization. While the method applies to broad scientific domains, we choose molecular optimization to demonstrate LICO in this paper. We aim to develop a model capable of efficiently adapting to various objective functions after training. To achieve this, we propose a simple extension to existing LLMs and an unsupervised objective using semi-synthetic data to facilitate generalization.

4.1 MODEL ARCHITECTURE

In black-box optimization, a surrogate model f_θ estimates the distribution of the function value y given the input x and past observations $\mathcal{D}_{\text{obs}} = \{(x_i, y_i)\}_{i=1}^n$ the model has collected until the optimization iteration t:

$$p_\theta(y \mid x, x_1, y_1, x_2, y_2, \ldots, x_n, y_n), \tag{2}$$

where x_i and $y_i = f(x_i)$ are drawn from an objective function f. Our goal is to explore LLMs to model p_θ. As discussed earlier, we make no assumptions on the domain $\mathcal{X}$ to be expressed with natural language. To extend a pretrained language model to an arbitrary new domain, we equip the model with 3 new layers – an embedding layer for the inputs $x's$, an embedding layer for the function values $y's$, and a prediction layer for predicting the unknown function value y. Learning separate embedding layers offers several benefits. First, the new embedding layers encode x and y to a shared hidden space obtained by the language model via pretraining, which enables the model to escape the raw text space and perform in-context reasoning in the hidden space instead. Moreover, by embedding each input x to a single hidden vector instead of spanning it over several tokens, we effectively reduce the sequence length and thus allow the model to scale to more conditioning examples.

However, it is challenging for the model to perform this prediction task without any context information about the task. This is because, from the model point of view, embeddings of x and y do not mean anything more than some high-dimensional vectors. In other words, the model does not know what task it should perform and what each token in the embedding sequence represents. To address

this issue, we prepend each sequence with a `task prompt` and prepend each input x with a special token <x> and each function value y with a special token <y>. The `task prompt` instructs the model to perform the task, while the special tokens <x> and <y> inform the model of the position of each input x and the corresponding function value y. In other words, we use a language the model has mastered (natural language) to guide the learning of a new "foreign language" (e.g., molecule). In practice, the `task prompt` is *"Each x is a molecule and each y is the property of the corresponding molecule. Predict y given x."*, whereas <x> and <y> are two single characters *"x"* and *"y"*. Finally, we apply the prediction layer on top of each token <y> to predict the function value given the tokens preceding it. Each prediction consists of a mean and a standard deviation value which will be used for the selection of candidates during optimization. Figure 1 illustrates the architecture of LICO.

It is worth noting that the combination of natural language and domain-specific embeddings is the main distinction between LICO and previous works such as FPT (Lu et al., 2022) which applies pretrained LLMs to *sequence classification* tasks in non-language modalities. FPT also learns new embedding layers for the new domain, but relies entirely on the pretrained self-attention layers to model these embeddings without any language instructions. This distinction stems from the different nature of the tasks we aim to tackle. In sequence classification, the model produces a single prediction for the entire sequence, thus having a good representation of the sequence via self-attention is sufficient. For in-context learning, however, the model must associate each input x with its value y to infer the underlying function f and make predictions for unknown y. A language instruction that specifies where x is and where y is helps the model identify this association and improve its in-context reasoning. Our ablation study in 5.2.1 confirms this utility of retaining language tokens.

4.2 SEMI-SYNTHETIC TRAINING

Our goal is to train LICO on the unlabeled data $\mathcal{D}_\text{u}$ with an unsupervised objective to facilitate efficient generalization to an arbitrary objective function f in the same domain $\mathcal{X}$ after training. Our key insight is that if we train the model to perform the estimation in equation 2 for a wide range of functions, it should adapt to any objective function post-training. While the true function values are unknown before optimization, we can use the unlabeled data $x's$ to generate training data from *other functions*. Assume we have access to a family of functions $\tilde{\mathcal{F}}$ that operate on the same input domain $\mathcal{X}$. For each function $\tilde{f}$ drawn from $\tilde{\mathcal{F}}$, we sample a set of function evaluations $\{(x_i, y_i)\}_{i=1}^n$ and train the model to autoregressively predict y given the input x and preceding (x, y) pairs:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\sum_{i=1}^{n} \log p_\theta(y_i \mid x_i, x_{<i}, y_{<i})\right], \tag{3}$$

in which the expectation is with respect to $\tilde{f} \sim \tilde{\mathcal{F}}$, $x_{1:n} \sim \mathcal{D}_\text{u}$, and $y_{1:n} = \tilde{f}(x_{1:n})$. After training, the estimation in equation 2 can be done purely via in-context prompting, where we condition the model on past observations to make predictions for new data points.

Ideally, the function family $\tilde{\mathcal{F}}$ should be close to the downstream objective f, but also be diverse enough to encourage broad generalization across functions. To achieve this, we propose to train LICO on a mix of *intrinsic* and *synthetic* functions, which we term *semi-synthetic* training. Intrinsic functions are functions that map each input molecule x to an inherent property of x. For example, molecular weight, the number of rings, or heavy atom count are intrinsic properties of the molecule that are known from domain knowledge or can be easily computed using standard tools. These intrinsic properties are closely related to many downstream objective functions. For example, the biological activity of a drug molecule, such as its ability to inhibit a particular enzyme, is often closely related to the molecule's shape or conformation. Therefore, training LICO from these functions encourages the model to learn useful representations of the input x and obtain good prior knowledge about the optimization domain.

However, it is important to note that we are ultimately interested in optimizing other functions outside of the intrinsic function set. Training the model only on a limited set of intrinsic functions may result in overfitting and poor generalization to unseen functions. To diversify the training data, we additionally train the model on synthetically generated functions. A synthetic function family should be easy to sample from and be capable of producing diverse functions. Many such families exist, including Gaussian Processes (GPs), randomly constructed Gaussian Mixture Models, or randomly initialized neural networks. We choose to generate synthetic functions from Gaussian Processes

with a Tanimoto kernel due to its simplicity and efficiency. Tanimoto kernel, also known as the Jaccard coefficient, measures the similarity between two vectors of binary values, a representation that is widely used for many scientific domains such as chemistry, drug discovery, or bioinformatics. Specifically, each synthetic function $\tilde{f}$ is sampled as follows,

$$\tilde{f} \sim \mathcal{GP}(0, \mathcal{K}), \quad \mathcal{K}(x, x') = \frac{x \cdot x'}{||x||^2 + ||x'||^2 - x \cdot x'}, \tag{4}$$

where $\mathcal{K}(x, x')$ is the Tanimoto kernel that measures the similarity between x and x'.

The final family of functions $\tilde{\mathcal{F}}$ used to train LICO is a mixture of intrinsic and synthetic functions with a certain ratio. This design choice is critical to the model's performance. Intuitively, training on both types of functions ensures proximity to the downstream objectives and good coverage of the function space for efficient generalization. The use of intrinsic functions is also the main difference between our work and ExPT (Nguyen et al., 2023), a recent method that studies pure synthetic pretraining for optimization. We hypothesize that while synthetic data is sufficient for ExPT on a few simple tasks, for a more complex domain such as molecular optimization, synthetic training provides too little relevant signal for the model to generalize to downstream objectives. We empirically show the importance of both intrinsic and synthetic functions in the ablation study in section 5.2.2.

4.3 LICO for Black-box Optimization

After training, a single LICO model can be used for optimizing various objective functions within the domain $\mathcal{X}$. Optimization involves an iterative process. At each iteration t, we generate a set of candidates $\{x_i\}_{i=1}^{C}$ using standard crossover and mutation operations for which the model predicts the mean μ_i and standard deviation σ_i conditioned on prior observations $\mathcal{D}_{\text{obs}}$, a dataset of (x, y) pairs collected until t. An acquisition function α then calculates a utility score based on μ_i and σ_i for each candidate, balancing between exploration (favoring high σ) and exploitation (favoring high μ). The top k candidates determined by their utility scores are evaluated using the objective function f. These k candidates and their corresponding evaluations are incorporated into the dataset $\mathcal{D}_{\text{obs}}$, and the cycle repeats. This process terminates once we exhaust the evaluation budget of B. Algorithm 1 summarizes the optimization process and Appendix A.3 outlines the optimization hyperparameters.

5 Experiments

We evaluate LICO on molecular optimization, where the goal is to design new molecules with desired properties such as high chemical stability, low toxicity, or selective inhibition against a target disease. This problem plays a pivotal role in advancing drug and material discovery.

5.1 PMO Benchmark

Benchmark We evaluate LICO on Practical Molecular Optimization (PMO) (Gao et al., 2022), a standard benchmark for molecular optimization with a focus on sample efficiency. We experiment on 21 optimization objectives provided by PMO, including QED Bickerton et al. (2012), DRD2 (Olivecrona et al., 2017), and 19 objective functions from Guacamol (Brown et al., 2019). QED assesses a molecule's drug-likeness by identifying certain "red flags". DRD2 is a machine learning model trained on experimental data to predict bioactivities for specific target diseases. Guacamol objectives emulate drug discovery goals through a multi-property objective (MPO) approach, considering factors like target molecule similarity, molecular weights, and CLogP. All objective values range from 0 to 1, with 1 indicating the best outcome.

Baselines We compare LICO against 5 leading methods in PMO, namely Genetic GFN (Kim et al., 2024), REINVENT (Olivecrona et al., 2017), Augmented Memory (Guo & Schwaller, 2023), Graph GA (Jensen, 2019), and GP BO (Tripp et al., 2021). Genetic GFN employs a GFlowNets (Bengio et al., 2023) model trained to sample molecules proportional to their rewards. REINVENT is a reinforcement learning method that finetunes a pretrained RNN for generating SMILES strings, and Augmented Memory combines REINVENT with data augmentation and experience replay. Graph GA, inspired by evolutionary processes, utilizes crossover and mutation operations to explore the molecule space. GP BO is a Bayesian optimization method that augments Graph GA with a Gaussian

Table 1: The performance of LICO and the baselines on 21 optimization tasks in PMO. Higher score is better. We report the mean and stddev of scores averaged over 5 random seeds. We use **blue** and **violet** to denote the best and second-best method for each task.

Task	GP BO	Graph GA	LICO	REINVENT	Genetic GFN	Augmented Memory
albuterol_similarity	0.636 ± 0.106	0.583 ± 0.065	$\mathbf{0.656 \pm 0.125}$	0.496 ± 0.020	$\mathbf{0.664 \pm 0.054}$	0.557 ± 0.048
amlodipine_mpo	0.519 ± 0.014	0.501 ± 0.016	$\mathbf{0.541 \pm 0.026}$	0.472 ± 0.008	$\mathbf{0.534 \pm 0.019}$	0.489 ± 0.009
celecoxib_rediscovery	0.411 ± 0.046	0.424 ± 0.049	$\mathbf{0.447 \pm 0.073}$	0.370 ± 0.029	$\mathbf{0.447 \pm 0.028}$	0.385 ± 0.027
deco_hop	0.593 ± 0.013	0.581 ± 0.006	$\mathbf{0.596 \pm 0.010}$	0.572 ± 0.006	$\mathbf{0.604 \pm 0.017}$	0.579 ± 0.010
drd2	$\mathbf{0.857 \pm 0.080}$	0.833 ± 0.065	$\mathbf{0.859 \pm 0.066}$	0.775 ± 0.086	0.809 ± 0.045	0.795 ± 0.024
fexofenadine_mpo	$\mathbf{0.707 \pm 0.021}$	0.666 ± 0.009	$\mathbf{0.700 \pm 0.023}$	0.650 ± 0.007	0.682 ± 0.021	0.679 ± 0.021
isomers_c7h8n2o2	0.545 ± 0.158	0.735 ± 0.112	$\mathbf{0.779 \pm 0.099}$	0.725 ± 0.064	$\mathbf{0.738 \pm 0.039}$	0.661 ± 0.039
isomers_c9h10n2o2pf2cl	0.599 ± 0.059	0.630 ± 0.086	$\mathbf{0.672 \pm 0.075}$	0.630 ± 0.032	$\mathbf{0.656 \pm 0.075}$	0.596 ± 0.066
median1	0.213 ± 0.020	0.208 ± 0.015	0.217 ± 0.019	0.205 ± 0.012	$\mathbf{0.219 \pm 0.008}$	$\mathbf{0.219 \pm 0.014}$
median2	$\mathbf{0.203 \pm 0.009}$	0.181 ± 0.009	0.193 ± 0.009	0.188 ± 0.010	$\mathbf{0.204 \pm 0.011}$	0.184 ± 0.010
mestranol_similarity	$\mathbf{0.427 \pm 0.025}$	0.362 ± 0.017	$\mathbf{0.423 \pm 0.016}$	0.379 ± 0.026	0.414 ± 0.022	0.393 ± 0.021
osimertinib_mpo	$\mathbf{0.766 \pm 0.006}$	0.751 ± 0.005	0.759 ± 0.008	0.737 ± 0.007	$\mathbf{0.763 \pm 0.008}$	0.761 ± 0.006
perindopril_mpo	0.458 ± 0.019	0.435 ± 0.016	$\mathbf{0.473 \pm 0.009}$	0.404 ± 0.009	$\mathbf{0.462 \pm 0.033}$	0.422 ± 0.013
qed	0.912 ± 0.010	0.914 ± 0.007	$\mathbf{0.925 \pm 0.005}$	0.921 ± 0.002	$\mathbf{0.928 \pm 0.002}$	0.923 ± 0.002
ranolazine_mpo	$\mathbf{0.701 \pm 0.023}$	0.620 ± 0.014	$\mathbf{0.687 \pm 0.029}$	0.574 ± 0.044	0.623 ± 0.022	0.614 ± 0.033
scaffold_hop	0.478 ± 0.009	0.461 ± 0.008	$\mathbf{0.480 \pm 0.008}$	0.447 ± 0.010	$\mathbf{0.485 \pm 0.015}$	0.460 ± 0.010
sitagliptin_mpo	0.232 ± 0.083	0.229 ± 0.053	$\mathbf{0.315 \pm 0.097}$	$\mathbf{0.261 \pm 0.026}$	0.227 ± 0.041	0.245 ± 0.030
thiothixene_rediscovery	$\mathbf{0.351 \pm 0.039}$	0.322 ± 0.023	0.343 ± 0.035	0.311 ± 0.021	$\mathbf{0.377 \pm 0.015}$	0.336 ± 0.033
troglitazone_rediscovery	$\mathbf{0.313 \pm 0.018}$	0.267 ± 0.015	$\mathbf{0.292 \pm 0.028}$	0.246 ± 0.009	0.277 ± 0.015	0.262 ± 0.012
valsartan_smarts	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zaleplon_mpo	0.392 ± 0.034	0.374 ± 0.024	0.404 ± 0.022	$\mathbf{0.406 \pm 0.017}$	0.400 ± 0.014	$\mathbf{0.415 \pm 0.013}$
Sum of scores (↑)	10.313	10.076	**10.760**	9.772	**10.515**	9.974

Processes surrogate model and UCB acquisition function to guide candidate selection. In addition to PMO baselines, we also compare LICO with GP BO is the most similar to LICO, where the only difference is we use an LLM for surrogate modeling instead of a GP.

LICO training We use ZINC 250K as the unlabeled dataset $\mathcal{D}_{\text{u}}$. ZINC 250K contains around 250000 molecules sampled from the full ZINC database (Sterling & Irwin, 2015) with moderate size and high pharmaceutical relevance and popularity. We adopt 2-radius 2048 bit molecular fingerprints as the input feature of the molecule. To generate training data, we use 47 intrinsic properties of the molecule as the intrinsic functions, which we present in detail in Appendix A.1. We train LICO for 20000 iterations with a batch size of 4, where each data point is a sequence of (x, y) pairs sampled from an intrinsic or synthetic function. The ratio of synthetic data is 0.1. We use Llama-2-7b (Touvron et al., 2023b) as the base LLM, and use LoRA (Hu et al., 2021) for parameter-efficient finetuning.

Optimization details We limit the optimization budget of all methods to 1000 function calls. We report the area under the curve (AUC) of the top-10 average objective value against the number of function calls as the performance metric. AUC metric favors methods that obtain high values with a smaller number of function calls, thus evaluating both optimization capability and sample efficiency. We min-max scale the AUC values to $[0, 1]$. We aggregate the performance for each method across 5 seeds for better reproducibility as suggested by PMO.

Results Table 1 summarizes the performance of the 5 considered methods across 21 optimization tasks. Overall, LICO is the leading method in this benchmark, achieving the highest aggregated score and the lowest mean rank. Specifically, LICO achieves the best performance or the second-best performance in $15/21$ tasks. It is important to note that LICO achieves this impressive result without being explicitly trained on data from downstream objectives. This shows the effectiveness of semi-synthetic training in enabling generalization to a broad range of functions via in-context prompting. TNP performs poorly in this benchmark, despite sharing a similar architecture with LICO. This performance gap highlights the importance of the pattern-matching capabilities LLMs acquire through extensive pretraining, which are crucial for adapting to new domains.

The most closely related method to LICO in this table is GP BO, where the only difference between the two is the surrogate model. This indicates the superiority of LICO compared to GP, a popular surrogate model for black-box optimization. To verify this, we compare the predictive performance of LICO and GP on several objective functions. We do this by first labeling the ZINC unlabeled dataset with the objective functions and randomly choosing a subset of the labeled data points for evaluation. For each task, we vary the number of examples given to each method from 32 to 512, and evaluate their performance on 128 held-out data points. We use negative log-likelihood, mean squared error, and root mean squared calibration error as the evaluation metrics. Figure 2 compares the predictive performance of LICO and GP in 3 objective functions, `median1`, `ranolazine_mpo`, and `troglitazone_rediscovery`. The figure shows that the optimization performance of the method closely aligns with the predictive performance of the surrogate model. In `median1` and

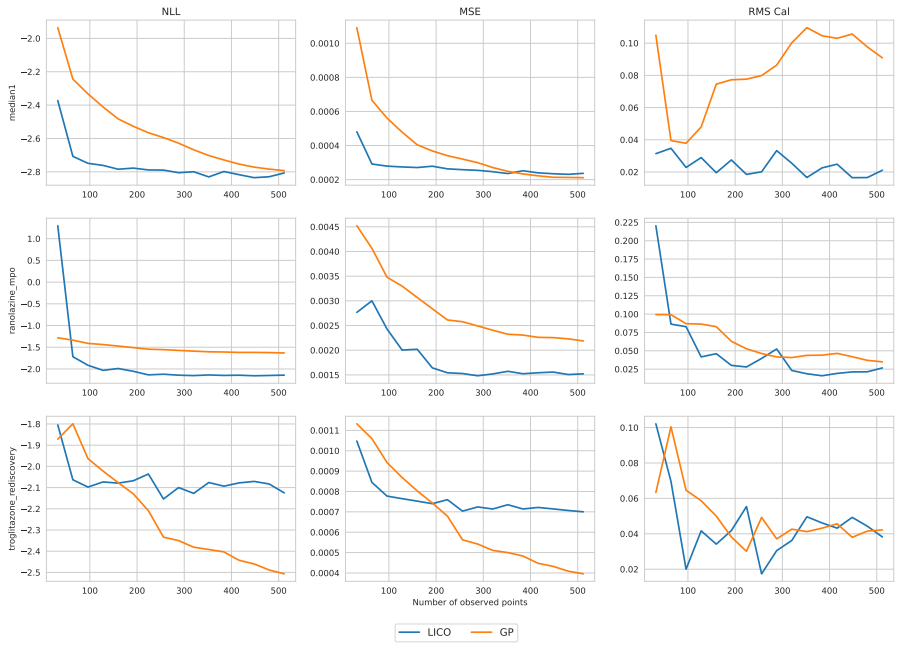

Figure 2: The predictive performance of LICO and GP on 3 objective functions in PMO with different metrics and varying numbers of observations.

Table 2: Performance of LICO on 5 tasks with different language instructions.

Task	albuterol_similarity	amlodipine_mpo	celecoxib_rediscovery	deco_hop	drd2	Sum (↑)
LICO w/o Language	0.615 ± 0.104	0.491 ± 0.018	0.396 ± 0.051	0.585 ± 0.010	0.840 ± 0.063	2.927
LICO w/o Task prompt	0.641 ± 0.107	0.523 ± 0.018	$\mathbf{0.457 \pm 0.041}$	0.595 ± 0.006	0.844 ± 0.105	3.060
LICO	$\mathbf{0.656 \pm 0.125}$	$\mathbf{0.541 \pm 0.026}$	0.447 ± 0.073	$\mathbf{0.596 \pm 0.010}$	$\mathbf{0.859 \pm 0.066}$	**3.099**

Table 3: Performance of LICO on 5 tasks with different ratios of synthetic data.

Task	albuterol_similarity	amlodipine_mpo	celecoxib_rediscovery	deco_hop	drd2	Sum (↑)
LICO Intrinsic	0.598 ± 0.115	0.524 ± 0.029	0.412 ± 0.042	0.585 ± 0.005	0.891 ± 0.032	3.010
LICO 0.1 Synthetic	0.656 ± 0.125	$\mathbf{0.541 \pm 0.026}$	$\mathbf{0.447 \pm 0.073}$	$\mathbf{0.596 \pm 0.010}$	0.859 ± 0.066	**3.099**
LICO 0.5 Synthetic	$\mathbf{0.663 \pm 0.140}$	0.504 ± 0.016	0.402 ± 0.016	0.588 ± 0.006	$\mathbf{0.907 \pm 0.020}$	3.063
LICO Synthetic	0.547 ± 0.080	0.498 ± 0.026	0.404 ± 0.103	0.585 ± 0.003	0.902 ± 0.012	2.936

ranolazine_mpo where LICO outperforms GP in terms of optimization score, the model also achieves lower negative log-likelihood, mean squared error, and calibration error. Similarly, LICO has worse predictive performance in troglitazone_rediscovery where it underperforms GP. This verifies our hypothesis and proves the effectiveness of LICO for surrogate modeling.

We also note that there is a discrepancy between Table 1 and the results reported in PMO. This is due to several reasons. First, we use a smaller optimization budget of 1000 queries compared to 10000 in PMO. We believe 1000 is a more reasonable budget while still allowing optimization methods to achieve meaningful performances. Moreover, we found the GP BO implementation in PMO to be suboptimal, specifically in how it generated the candidate pool. The original implementation applied crossover and mutation to a mix of the best individuals and randomly selected individuals from the last iteration to generate the candidate pool for the current iteration. However, we found that only using the best individuals from the last iteration results in a much better performance. By implementing this change, GP BO becomes a much stronger baseline than Graph GA and REINVENT. To ensure a fair comparison, we performed extensive hyperparameter tuning for each baseline on the first 5 tasks, and used the optimal hyperparameters for the remaining tasks. Appendix B.1 details hyperparameter search for the baselines.

Table 4: Performance of pretrained vs randomly initialized LLMs.

Task	Pretrained LLM	Scratch LLM
albuterol_similarity	$\mathbf{0.656 \pm 0.125}$	0.575 ± 0.064
amlodipine_mpo	$\mathbf{0.541 \pm 0.026}$	0.503 ± 0.029
celecoxib_rediscovery	$\mathbf{0.447 \pm 0.073}$	0.410 ± 0.034
deco_hop	$\mathbf{0.596 \pm 0.010}$	0.583 ± 0.005
drd2	$\mathbf{0.859 \pm 0.066}$	0.827 ± 0.085
Sum	**3.099**	2.898

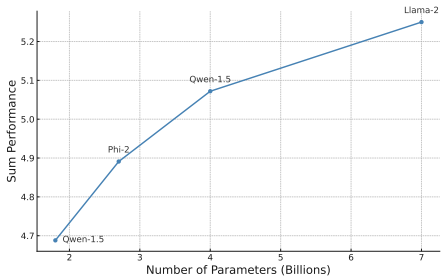

Figure 3: LICO with different LLM sizes.

5.2 Ablation Analysis

We perform various ablation studies to understand the importance of different components and design choices in LICO. For the ablation experiments, we consider the first 5 tasks in Table 1 only. We report the aggregated performance of different models using AUC Top-10 across 5 random seeds.

5.2.1 LICO without language instruction

First, we examine the importance of language instructions to the performance of LICO. We compare 3 variants of LICO: 1) LICO without any language instruction, 2) LICO with special tokens <x> and <y> but without a task prompt, and 3) LICO with both special tokens and the task prompt. Table 2 compares the performance of the 3 variants. LICO performs the best in $4/5$ tasks, followed by LICO without the task prompt. LICO without any language instruction performs the worst, often by a large margin. This result confirms the importance of guiding a pretrained LLM with language instruction when applying the model to in-context reasoning in a completely new domain.

5.2.2 LICO with different synthetic ratios

We investigate the importance of training LICO on both intrinsic and synthetic data. To do this, we gradually increase the ratio of synthetic functions in the training data from 0 (intrinsic-only) to 1 (synthetic-only), and compare the performance of LICO across different ratios. Table 3 shows that LICO with semi-synthetic training performs the best, outperforming both intrinsic-only and synthetic-only data. Training with synthetic data only performs the worst, which is expected when synthetic functions generated by a GP do not include any domain knowledge that is encoded by the intrinsic functions. In other words, synthetic data alone provides too little relevant signal for the model to generalize to unseen downstream objectives. Training with intrinsic functions only, on the other hand, results in quite good performances on most tasks. However, in tasks like albuterol_similarity, semi-synthetic training outperforms this baseline by a large margin. We hypothesize that the underlying objective in albuterol_similarity is far from the intrinsic functions used to train LICO, leading to poor generalization. Finally, training with small (0.1) to moderate (0.5) ratios of synthetic data achieves similarly good performance.

5.2.3 Randomly initialized vs Pretrained LLMs

To understand the importance of using a pretrained LLM, we compare LICO with an autoregressive transformer model of the same size (7B). The transformer architecture is the same as in (Garg et al., 2022), and we train this model to perform in-context learning on the semi-synthetic data from scratch. Table 4 shows the comparison. The scratch model performs much worse than LICO with a pretrained LLM on all tasks despite sharing the same number of parameters. This highlights the importance of the pattern-matching capabilities that LLMs like Llama-2 acquire through extensive language pretraining.

5.2.4 LICO with different LLM sizes

Previous works have shown the favorable scaling laws of Large Language Models where larger models consistently perform better on downstream tasks (Kaplan et al., 2020). In this section, we

investigate the scaling properties of LLMs but in the context of black-box optimization. Specifically, we compare 4 different base LLMs with different sizes – Qwen-1.5 1.8B and 4B (Bai et al., 2023), Phi-2 2.7B (Javaheripi et al., 2023), and Llama-2 7B (Touvron et al., 2023b). We use the same language instructions for all models. We evaluate each model on the first 8 tasks in Table 1 and average the results across 5 random seeds. We report the sum of performance across 8 tasks.

The comparison in Figure 3 shows that the optimization performance scales consistently with the model size, with Llama-2 7B being the best method. This experiment indicates that larger LLMs not only perform better in language tasks but also obtain stronger pattern-matching capabilities that can be transferred to a completely different domain. Given this scaling, we can further improve the current performance of LICO by scaling up the base LLM size.

6 Conclusion and Future Work

We develop LICO, a new method that leverages pretrained Large Language Models for black-box optimization. LICO extends existing LLMs to non-language domains with separate embedding and prediction layers. To enable efficient generalization to various optimization tasks, we train LICO on a diverse set of semi-synthetic functions for few-shot predictions. LICO achieves state-of-the-art performance on PMO, a challenging molecular optimization benchmark with over 20 objective functions. Ablation analyses highlight the importance of incorporating language instruction to guide in-context learning and semi-synthetic training for better generalization.

One limitation of our method is the assumption of an accessible set of intrinsic functions. While this is true for molecular optimization, it may not apply to other scientific domains. In such cases, a better synthetic data generation process incorporating domain knowledge is needed to aid generalization.

Future directions include evaluating LICO in other domains to test its applicability and generality, exploring other prompts that better exploit the capabilities of pretrained LLMs, and using LLMs for other aspects of optimization, such as candidate suggestion or exploration.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. 2020.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *The Journal of Machine Learning Research*, 24(1):10006–10060, 2023.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 491–496. IEEE, 2016.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Herbie Bradley, Honglu Fan, Theodoros Galanos, Ryan Zhou, Daniel Scott, and Joel Lehman. The openelm library: Leveraging progress in language models for novel evolutionary algorithms. *Genetic Programming Theory and Practice XX. Springer*, 2024.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pp. 773–782. PMLR, 2019.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3): 1096–1108, 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Angelica Chen, David M Dohan, and David R So. Evoprompting: Language models for code-level neural architecture search. *arXiv preprint arXiv:2302.14838*, 2023.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35: 11763–11784, 2022.

Yin Fang, Ningyu Zhang, Zhuo Chen, Lingbing Guo, Xiaohui Fan, and Huajun Chen. Domain-agnostic molecular generation with self-feedback. *arXiv preprint arXiv:2301.11259*, 2023.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 35: 21342–21357, 2022.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. *arXiv preprint arXiv:2310.07820*, 2023.

Philipp Guevorguian, Menua Bedrosian, Tigran Fahradyan, Gayane Chilingaryan, Hrant Khachatrian, and Armen Aghajanyan. Small molecule optimization with large language models. *arXiv preprint arXiv:2407.18897*, 2024.

Jeff Guo and Philippe Schwaller. Augmented memory: Capitalizing on experience replay to accelerate de novo molecular design. *arXiv preprint arXiv:2305.16160*, 2023.

Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 2023.

Jan 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.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hyeonah Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. Genetic-guided gflownets: Advancing in practical molecular optimization benchmark. *arXiv preprint arXiv:2402.05961*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Generative pretraining for black-box optimization. In *ICML*, 2023a.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *ICML*, 2023b.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alan Aspuru-Guzik, and Geoff Pleiss. A sober look at llms for material discovery: Are they actually good for bayesian optimization over molecules? In *Forty-first International Conference on Machine Learning*.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pp. 331–366. Springer, 2023.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2488–2494. IEEE, 2019.

Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Large language model for multi-objective evolutionary optimization. *arXiv preprint arXiv:2310.12541*, 2023a.

Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. Chatgpt-powered conversational drug editing using retrieval and domain feedback. *arXiv preprint arXiv:2305.18090*, 2023b.

Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. Conversational drug editing using retrieval and domain feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. In *The Twelfth International Conference on Learning Representations*.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7628–7636, 2022.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Elliot Meyerson, Mark J Nelson, Herbie Bradley, Arash Moradi, Amy K Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting. *arXiv preprint arXiv:2302.12170*, 2023.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *ICML*, 2022.

Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. Expt: Synthetic pretraining for few-shot experimental design. In *NeurIPS*, 2023.

Allen Nie, Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Importance of directional feedback for llm-based optimizers. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

Mayk Caldas Ramos, Shane S Michtavy, Marc D Porosoff, and Andrew D White. Bayesian optimization of catalysts with in-context learning. *arXiv preprint arXiv:2304.05341*, 2023.

Bojana Ranković and Philippe Schwaller. Bochemian: Large language model embeddings for bayesian optimization of chemical reactions. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023.

Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

Junhong Shen, Liam Li, Lucio M Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Cross-modal fine-tuning: Align then refine. *arXiv preprint arXiv:2302.05738*, 2023.

Henry W Sprueill, Carl Edwards, Khushbu Agarwal, Mariefel V Olarte, Udishnu Sanyal, Conrad Johnston, Hongbin Liu, Heng Ji, and Sutanay Choudhury. Chemreasoner: Heuristic search over a large language model's knowledge space using quantum-chemical feedback. *arXiv preprint arXiv:2402.10980*, 2024.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Austin Tripp, Gregor NC Simm, and José Miguel Hernández-Lobato. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*, 2021.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

Christoph Völker, Tehseen Rug, Kevin Maik Jablonka, and Sabine Kruschwitz. Llms can design sustainable concrete–a systematic benchmark. 2024.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, et al. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

David 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.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. Drugassist: A large language model for molecule optimization. *arXiv preprint arXiv:2401.10334*, 2023.

Michael R Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large language models for hyperparameter optimization. *arXiv e-prints*, pp. arXiv–2312, 2023.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A LICO IMPLEMENTATION DETAILS

A.1 MOLECULAR INTRINSIC FUNCTIONS

We utilize 47 intrinsic properties of molecules for pretraining LICO. Table 5 shows the intrinsic properties and their explanation.

Table 5: Inherent Properties of Molecules and their Explanations

Property	**Explanation**
Molecular Weight	Total mass of all atoms in the molecule.
Number of Rotatable Bonds	Bonds that allow free rotation around themselves.
Number of Rings	Count of ring structures in the molecule.
Number of H Donors	Atoms in the molecule that can donate a hydrogen atom.
Number of H Acceptors	Atoms in the molecule capable of accepting a hydrogen atom.
Num Aromatic Rings	Count of rings with a pattern of alternating single and double bonds.
Num Aliphatic Rings	Count of non-aromatic rings in the molecule.
Num Saturated Rings	Rings with single bonds only.
Num Heteroatoms	Atoms other than carbon or hydrogen.
Fraction Csp3	Fraction of carbon atoms bonded with a single pair of electrons.
Heavy Atom Count	Count of all atoms except hydrogen.
Num Valence Electrons	Total number of electrons that can participate in the formation of chemical bonds.
Num Aromatic CarboRings	Aromatic rings composed solely of carbon atoms.
Num Aromatic HeteroRings	Aromatic rings containing at least one heteroatom.
Num Saturated CarboRings	Saturated rings made only of carbon atoms.
Num Saturated HeteroRings	Saturated rings containing at least one heteroatom.
BalabanJ	Topological index to quantify molecular branching.
BertzCT	A measure of structural complexity of the molecule.
Ipc	Information content on the vertex degree.
HallKierAlpha	Valence connectivity index used in molecular shape analysis.
Kappa1	Shape descriptor based on the skeleton of the molecule.
Kappa2	Hydrogen suppressed graph descriptor.
Kappa3	Hydrogen complete graph descriptor.
Chi0	Randić molecular connectivity index.
Chi1	Valence modified Randić molecular connectivity index.
Chi0n	Randić connectivity index normalized.
Chi1n	Valence modified Randić connectivity index normalized.
Chi2n	Second order Randić connectivity index normalized.
Chi3n	Third order Randić connectivity index normalized.
Chi4n	Fourth order Randić connectivity index normalized.
Chi0v	Randić connectivity index for valence electrons.
Chi1v	First order valence molecular connectivity index.
Chi2v	Second order valence molecular connectivity index.
Chi3v	Third order valence molecular connectivity index.
Chi4v	Fourth order valence molecular connectivity index.
Molar Refractivity	Measure of the molecule's polarizability.
AMW	Average molecular weight of all atoms in the molecule.
Max Partial Charge	Maximum partial charge in the molecule.
Min Partial Charge	Minimum partial charge in the molecule.
Max Abs Partial Charge	Maximum absolute value of the partial charges in the molecule.
Min Abs Partial Charge	Minimum absolute value of the partial charges in the molecule.
Labute ASA	Labute's Approximate Surface Area, an estimate of the molecular surface area.
Max EState Index	Maximum electrotopological state index of the atoms in the molecule.
Min EState Index	Minimum electrotopological state index of the atoms in the molecule.
Max Abs EState Index	Maximum absolute value of the electrotopological state indices in the molecule.
Min Abs EState Index	Minimum absolute value of the electrotopological state indices in the molecule.
fr_C_O	Frequency of carbon-oxygen bonds in the molecule.

A.2 TRAINING DETAILS

The x embedding layer, y embedding layer, and prediction layer in LICO are MLPs with a hidden dimension of 1024. We train LICO for 20000 steps with a batch size of 4. For each data point in the batch, we randomly decide whether to sample an intrinsic or a synthetic function, with the probability of choosing synthetic functions being 0.1. Each data point is a sequence of (x, y) pairs with length $n \sim \mathcal{U}[64, 800]$. If the function is an intrinsic function, we uniformly sample a property from Table 5, otherwise sample synthetic data following Equation equation 4.

We use Llama-2-7b (Touvron et al., 2023b) as the base LLM, and use LoRA (Hu et al., 2021) for parameter-efficient finetuning. We use a base learning rate of $5e-4$ with a linear warmup for 1000

steps and a cosine decay for the remaining 19000 steps. We use LoRA with a rank of 16 and α scale of 16.

A.3 BLACK-BOX OPTIMIZATION HYPERPARAMETERS

We use Algorithm 1 to optimize a black-box function with LICO. We initialize the observed dataset $\mathcal{D}_{\text{obs}}$ with a population of 34 molecules sampled randomly from ZINC. At each iteration, we use the best 34 candidates in $\mathcal{D}_{\text{obs}}$ to generate new candidates via crossover and mutation operations, with the mutation rate being 0.01. The candidate pool size C is 100. We predict the mean μ_i and standard deviation σ_i for each candidate x_i in the pool using LICO. We employ a UCB acquisition function to compute the utility score $u_i = \mu_i + \beta\sigma_i$, which balances exploration and exploitation. Following (Gao et al., 2022), we set $\beta = 10^b$, where $b \sim \mathcal{U}[-0.5, 1.5]$. We then pick $k = 15$ candidates with the highest utility scores. We evaluate each selected candidate x_j using the black-box function f, and add the new data point (x_j, y_j) to the observed dataset $\mathcal{D}_{\text{obs}}$. The process continues with the updated observed dataset, and stops when $|\mathcal{D}_{\text{obs}}| = 1000$.

When predicting $\mu_i, \sigma_i = f_\theta(x_i, \mathcal{D}_{\text{obs}})$, we normalize all the $y's$ values in $\mathcal{D}_{\text{obs}}$ to have mean 0 and standard deviation 1. This is to resemble the finetuning data distribution of LICO. We then denormalize μ_i and σ_i to the original space.

A.4 BLACK-BOX OPTIMIZATION WITH LICO

Algorithm 1 outlines the optimization algorithm using LICO as the surrogate model.

Algorithm 1 Black-box optimization with LICO

Require: objective f, LICO model f_θ, budget B, candidate pool size C, acquisition function α, batch size k
  Initialize $\mathcal{D}_{\text{obs}} = \{\}$
  while $|\mathcal{D}_{\text{obs}}| < B$ **do**
    Generate a set of candidates $\{x_i\}_{i=1}^{C}$
    for each candidate x_i **do**
      Predict $\mu_i, \sigma_i = f_\theta(x_i, \mathcal{D}_{\text{obs}})$
      Compute utility score $u_i = \alpha(\mu_i, \sigma_i)$
    end for
    Select k candidates with the highest utility scores
    for each selected candidate x_j **do**
      Evaluate x_j using the actual objective $y_j = f(x_j)$
      Add (x_j, y_j) to the observation dataset $\mathcal{D}_{\text{obs}}$
    end for
  end while

B BASELINE DETAILS

TNP is a transformer-based architecture for in-context learning. We refer to Nguyen & Grover (2022) for more details about TNP. We train a TNP model with 16 attention layers and 2048 hidden dimensions. Other hyperparameters are the same as for LICO. After training, we use TNP for black-box optimization using Algorithm 1 with the same optimization hyperparameters but replace LICO with TNP.

GP BO replaces the LICO surrogate model in Algorithm 1 with a Gaussian Process with a Tanimoto kernel. We optimize the Gaussian Process hyperparameters via maximum likelihood estimation on the initial population sampled from ZINC.

Graph GA is a model-free variant of Algorithm 1. Specifically, at each iteration, Graph GA generates a set of candidates using the same crossover and mutation operations, and directly evaluates and adds them to $\mathcal{D}_{\text{obs}}$, since it does not employ a surrogate model.

REINVENT adopts a policy-based RL approach to finetune a pretrained RNN to generate SMILES strings with high returns. At each optimization iteration, we sample a set of molecules from the

finetuned RNN, evaluate these molecules using the black-box function f, and add the new data points to $\mathcal{D}_{\text{obs}}$. We refer to Gao et al. (2022) for more details of the algorithm and other hyperparameters.

B.1 HYPERPARAMETER TUNING FOR THE BASELINES

To ensure the baselines achieve the best possible performance for the new PMO-1K benchmark, we performed extensive hyperparameter tuning for each baseline on the first 5 tasks with grid search, and used the optimal hyperparameters for the rest of the tasks. Table 6 specifies the grid search for each method.

Table 6: Grid search and optimal hyperparameters for the baseline methods.

Method	Grid Search	Optimal Hyperparameters

GP BO	`population_size ∈ {50, 100, 150, ..., 350}`
	`offspring_size ∈ {50, 100, 150}`
	`kept_offspring_size ∈ {5, 10, ..., 50}`
		`population_size = 50`
		`offspring_size = 100`
		`kept_offspring_size = 15`

Graph GA	`population_size ∈ {50, 100, 150, ..., 350}` `offspring_size ∈ {50, 100, 150}`	`population_size = 50` `offspring_size = 100`
REINVENT	`batch_size ∈ {4, 8, 16, 32, 64}` `experience_replay ∈ {4, 8, 16, 24, 32}`	`batch_size = 16` `experience_replay = 24`

Genetic GFN	`learning_rate ∈ {0.0001, 0.0005}`
	`batch_size ∈ {4, 8, 16, 32, 64}`
	`num_keep ∈ {128, 256, 512, 1024}`
	`offspring_size ∈ {2, 4, 8}`
	`ga_generations ∈ {1, 2}`
		`learning_rate = 0.0001`
		`batch_size = 8`
		`num_keep = 128`
		`offspring_size = 2`
		`ga_generations = 1`

Augmented Memory	`batch_size ∈ {4, 8, 16, 32, 64}` `replay_buffer_size ∈ {50, 100, 150}`	`batch_size = 32` `replay_buffer_size = 100`

C ADDITIONAL RESULTS

C.1 ADDITIONAL METRICS

In addition to AUC Average Top-10, we measure the optimization performance of different methods on AUC Average Top-1 and AUC Average Top-100 for a more comprehensive comparison. Table 7 and 8 show AUC Average Top-1 and AUC Average Top-100 performances, respectively.

D BROADER IMPACT

Our work studies the application of large language models to black-box optimization, particularly in the domain of molecular optimization. This intersection of machine learning and optimization holds significant promise for advancing our understanding of LLMs' capabilities and limitations, and has significant potential in areas like material science and drug discovery. Our main goal is to enhance machine learning and optimization techniques, but it's also important to consider how these advancements might affect society, such as speeding up the development of new medicines and materials.

E COMPUTE RESOURCES

All experiments in this paper are run on a cluster of 4 A6000 GPUs, each with 49GB of memory.

Table 7: The performance of LICO and the baselines on 21 optimization tasks in PMO with AUC Average Top-1 metric. A higher score is better. We report the mean and stddev of scores averaged over 5 random seeds. We use **blue** and **violet** to denote the best and second-best method for each task.

Task	GP BO	Graph GA	LICO	REINVENT	TNP
albuterol_similarity	**0.672 ± 0.109**	0.647 ± 0.080	**0.695 ± 0.150**	0.572 ± 0.026	0.611 ± 0.042
amlodipine_mpo	**0.538 ± 0.016**	0.526 ± 0.017	**0.560 ± 0.026**	0.500 ± 0.016	0.513 ± 0.016
celecoxib_rediscovery	0.434 ± 0.052	0.466 ± 0.062	**0.492 ± 0.079**	0.415 ± 0.031	**0.482 ± 0.067**
deco_hop	**0.598 ± 0.013**	0.590 ± 0.005	**0.603 ± 0.012**	0.585 ± 0.010	0.597 ± 0.002
drd2_current	0.895 ± 0.067	**0.898 ± 0.048**	**0.902 ± 0.055**	0.867 ± 0.077	0.831 ± 0.043
fexofenadine_mpo	**0.728 ± 0.022**	0.691 ± 0.011	**0.719 ± 0.025**	0.696 ± 0.012	0.706 ± 0.014
isomers_c7h8n2o2	0.576 ± 0.154	0.815 ± 0.120	**0.834 ± 0.109**	**0.846 ± 0.070**	0.761 ± 0.145
isomers_c9h10n2o2pf2cl	0.644 ± 0.053	0.708 ± 0.083	**0.714 ± 0.084**	**0.724 ± 0.043**	0.701 ± 0.086
median1	0.235 ± 0.016	0.233 ± 0.018	**0.242 ± 0.020**	0.229 ± 0.015	**0.238 ± 0.015**
median2	**0.212 ± 0.010**	0.193 ± 0.011	0.201 ± 0.009	**0.209 ± 0.013**	0.200 ± 0.018
mestranol_similarity	**0.449 ± 0.028**	0.387 ± 0.020	**0.445 ± 0.014**	0.433 ± 0.034	0.406 ± 0.011
osimertinib_mpo	**0.788 ± 0.008**	0.777 ± 0.008	**0.781 ± 0.007**	0.780 ± 0.009	0.776 ± 0.007
perindopril_mpo	**0.475 ± 0.019**	0.460 ± 0.025	**0.492 ± 0.011**	0.432 ± 0.010	0.457 ± 0.012
qed	0.926 ± 0.011	0.930 ± 0.004	**0.935 ± 0.002**	**0.934 ± 0.003**	0.931 ± 0.001
ranolazine_mpo	**0.729 ± 0.024**	0.684 ± 0.015	**0.711 ± 0.028**	0.657 ± 0.048	0.669 ± 0.032
scaffold_hop	**0.486 ± 0.010**	0.475 ± 0.008	**0.491 ± 0.013**	0.468 ± 0.010	0.484 ± 0.019
sitagliptin_mpo	0.268 ± 0.098	0.281 ± 0.069	**0.363 ± 0.114**	**0.333 ± 0.030**	0.274 ± 0.044
thiothixene_rediscovery	**0.371 ± 0.046**	0.351 ± 0.029	**0.368 ± 0.041**	0.345 ± 0.026	0.332 ± 0.041
troglitazone_rediscovery	**0.329 ± 0.019**	0.289 ± 0.021	**0.309 ± 0.033**	0.276 ± 0.009	0.286 ± 0.012
valsartan_smarts	**0.000 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**0.000 ± 0.000**	0.000 ± 0.000
zaleplon_mpo	0.431 ± 0.031	0.418 ± 0.022	**0.435 ± 0.027**	**0.456 ± 0.020**	0.428 ± 0.022
Sum of scores (↑)	10.784	**10.818**	**11.291**	10.755	10.683
Mean rank (↓)	**2.52**	3.57	**1.62**	3.48	3.75

Table 8: The performance of LICO and the baselines on 21 optimization tasks in PMO with AUC Average Top-100 metric. A higher score is better. We report the mean and stddev of scores averaged over 5 random seeds. We use **blue** and **violet** to denote the best and second-best method for each task.

Task	GP BO	Graph GA	LICO	REINVENT	TNP
albuterol_similarity	**0.548 ± 0.100**	0.470 ± 0.042	**0.563 ± 0.093**	0.395 ± 0.012	0.448 ± 0.028
amlodipine_mpo	**0.458 ± 0.008**	0.422 ± 0.014	**0.486 ± 0.025**	0.407 ± 0.005	0.420 ± 0.013
celecoxib_rediscovery	**0.363 ± 0.040**	0.346 ± 0.036	**0.372 ± 0.070**	0.296 ± 0.024	0.346 ± 0.026
deco_hop	**0.579 ± 0.013**	0.563 ± 0.006	**0.583 ± 0.009**	0.550 ± 0.006	0.568 ± 0.004
drd2_current	**0.741 ± 0.097**	0.605 ± 0.086	**0.725 ± 0.092**	0.615 ± 0.098	0.556 ± 0.095
fexofenadine_mpo	**0.645 ± 0.018**	0.588 ± 0.008	**0.636 ± 0.022**	0.549 ± 0.004	0.599 ± 0.016
isomers_c7h8n2o2	0.300 ± 0.142	**0.535 ± 0.091**	0.450 ± 0.149	**0.511 ± 0.058**	0.492 ± 0.115
isomers_c9h10n2o2pf2cl	**0.474 ± 0.038**	0.441 ± 0.068	**0.535 ± 0.067**	0.445 ± 0.027	0.447 ± 0.049
median1	**0.175 ± 0.022**	0.168 ± 0.013	0.166 ± 0.019	0.162 ± 0.007	**0.170 ± 0.008**
median2	**0.184 ± 0.006**	0.158 ± 0.008	**0.175 ± 0.010**	0.155 ± 0.006	0.162 ± 0.009
mestranol_similarity	**0.379 ± 0.020**	0.311 ± 0.016	**0.361 ± 0.030**	0.302 ± 0.016	0.314 ± 0.003
osimertinib_mpo	**0.706 ± 0.006**	0.667 ± 0.008	**0.694 ± 0.010**	0.623 ± 0.014	0.671 ± 0.006
perindopril_mpo	**0.405 ± 0.019**	0.357 ± 0.012	**0.424 ± 0.007**	0.332 ± 0.011	0.359 ± 0.010
qed	0.853 ± 0.010	0.854 ± 0.011	**0.882 ± 0.007**	**0.874 ± 0.003**	0.857 ± 0.003
ranolazine_mpo	**0.633 ± 0.020**	0.462 ± 0.022	**0.617 ± 0.021**	0.436 ± 0.040	0.468 ± 0.042
scaffold_hop	**0.462 ± 0.006**	0.435 ± 0.008	**0.462 ± 0.006**	0.415 ± 0.009	0.440 ± 0.010
sitagliptin_mpo	0.133 ± 0.062	0.103 ± 0.032	**0.171 ± 0.045**	**0.134 ± 0.016**	0.100 ± 0.023
thiothixene_rediscovery	**0.311 ± 0.030**	0.270 ± 0.015	**0.299 ± 0.026**	0.256 ± 0.015	0.261 ± 0.024
troglitazone_rediscovery	**0.283 ± 0.014**	0.228 ± 0.008	**0.258 ± 0.024**	0.201 ± 0.008	0.230 ± 0.005
valsartan_smarts	**0.000 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**0.000 ± 0.000**	0.000 ± 0.000
zaleplon_mpo	**0.301 ± 0.036**	0.258 ± 0.016	**0.318 ± 0.018**	0.296 ± 0.009	0.257 ± 0.013
Sum of scores (↑)	**8.933**	8.242	**9.175**	7.954	8.167
Mean rank (↓)	**1.95**	3.62	**1.76**	4.14	3.45