A Sober Look at LLMs for Material Discovery: Are They Actually Good for Bayesian Optimization Over Molecules?

Agustinus Kristiadi [1] **Felix Strieth-Kalthoff** [2] **Marta Skreta** [2 1] **Pascal Poupart** [3 1] **Alán Aspuru-Guzik** [2 1] **Geoff Pleiss** [4 1]

https://github.com/wiseodd/lapeft-bayesopt

Abstract

Automation is one of the cornerstones of contemporary material discovery. Bayesian optimization (BO) is an essential part of such workflows, enabling scientists to leverage prior domain knowledge into efficient exploration of a large molecular space. While such prior knowledge can take many forms, there has been significant fanfare around the ancillary scientific knowledge encapsulated in large language models (LLMs). However, existing work thus far has only explored LLMs for heuristic materials searches. Indeed, recent work obtains the uncertainty estimate—an integral part of BO—from point-estimated, *non-Bayesian* LLMs. In this work, we study the question of whether LLMs are actually useful to accelerate principled *Bayesian* optimization in the molecular space. We take a sober, dispassionate stance in answering this question. This is done by carefully (i) viewing LLMs as fixed feature extractors for standard but principled BO surrogate models and by (ii) leveraging parameter-efficient finetuning methods and Bayesian neural networks to obtain the posterior of the LLM surrogate. Our extensive experiments with real-world chemistry problems show that LLMs can be useful for BO over molecules, but only if they have been pretrained or finetuned with domain-specific data.

1. Introduction

Material discovery describes the inherently laborious, iterative process of designing materials candidates, preparing them experimentally, testing their properties, and eventually updating the initial design hypothesis (de Regt, 2020; Greenaway et al., 2023). While human researchers have largely driven this process for the last century, there is demand for more efficient automated methods in the face of pressing societal challenges related to health care, nutrition, or clean energy (Tom et al., 2024). Major challenges associated with the discovery process are the complex and black box-like mapping between a material's structure and its properties, as well as the vastness of the design space (Wang et al., 2023).

Prompt: What is the molecular mass of OS(=O)(=O)O?

Ground truth: 98.079 g/mol

ChatGPT: To calculate the molecular mass of a compound, you need to sum the atomic masses of all the atoms in the molecular formula. [...] Therefore, the molecular mass of OS(=O)(=O)O (sulfuric acid) is approximately 96.07 g/mol.

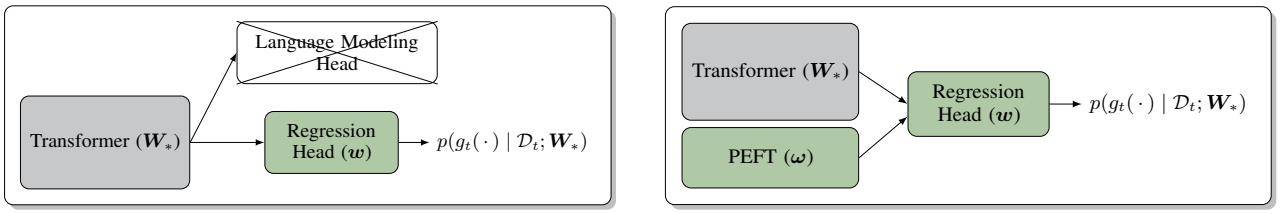
Llama-2-70b: The molecular formula for OS(=O)(=O)O is O3S. [...] Therefore, the molecular mass of OS(=O)(=O)O is 80.07 g/mol.

Figure 1. LLMs seem to "understand" chemistry. However, they often produce completely wrong answers while sounding very convincing. Both APIs were accessed on 2024-01-07.

To address the aforementioned problems, Bayesian optimization (BO; Močkus, 1975) has been increasingly used in chemistry (Griffiths et al., 2023; Hickman et al., 2023). Key components of successful BO include its *priors* (informative priors imply efficient posterior inference with limited data) and its *probabilistic surrogate models*—e.g. via Gaussian processes (Rasmussen & Williams, 2006; Snoek et al., 2012) or Bayesian neural networks (Kim et al., 2022; Li et al., 2024; Kristiadi et al., 2023). The probabilistic formulation of BO is useful since optimizing a black-box function is an inherently uncertain problem. We do not know *a priori* the form of the function and our approximation of it might be imprecise. Probabilistic surrogate models are thus useful to quantify the inherent uncertainty surrounding the optimization landscape, allowing for principled approaches to

[1]Vector Institute [2]University of Toronto [3]University of Waterloo [4]University of British Columbia. Correspondence to: Agustinus Kristiadi <akristiadi@vectorinstitute.ai>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



(a) Fixed-feature LLM surrogate

(b) Adaptive-feature LLM surrogate

Figure 2. The surrogates we consider in this work. "PEFT" refers to parameter efficient finetuning which adds a (proportionally) few trainable weights $\boldsymbol{\omega}$ to the transformer. Grey denotes frozen weights that act as conditioning variables in the posterior over the surrogate g_t. Green denotes weights that are inferred in a Bayesian manner (e.g., to obtain $p(\boldsymbol{w}, \boldsymbol{\omega} \mid \mathcal{D}_t)$) and then marginalized over to obtain the posterior predictive distribution g_t (e.g., $\iint p(g_t(\cdot) \mid \boldsymbol{w}, \boldsymbol{\omega}; \boldsymbol{W}_*)\, p(\boldsymbol{w}, \boldsymbol{\omega} \mid \mathcal{D}_t)\, d\boldsymbol{w}\, d\boldsymbol{\omega}$). Both models are principled Bayesian surrogates, in contrast to the in-context learning frameworks considered by prior works on BO with LLMs (Ramos et al., 2023; Liu et al., 2024).

the exploration-exploitation tradeoff (Garnett, 2023).

Good domain-specific priors, a necessary component for reliable uncertainty estimates, are hard to define analytically. Recent work has thus pursued *implicit priors* often obtained through pretrained feature extractors (Chithrananda et al., 2020; Ross et al., 2022). Large language models (LLMs)—which have become very popular in many domains that are traditionally rather disconnected from natural language processing such as biology (Vig et al., 2021), education (Kasneci et al., 2023), law (Chalkidis et al., 2020), and chemistry (Jablonka et al., 2023a; Guo et al., 2023; Jablonka et al., 2023b, etc.)—are one potential source of pretrained features for BO. On the other hand, recent works have warned that LLMs might not necessarily understand natural language, but simply act as very expensive "stochastic parrots" (Bender et al., 2021); see Figure 1, for example. Nevertheless, due to the apparent capabilities of LLMs, some recent works have leveraged off-the-shelf LLMs such as GPT-4 (OpenAI, 2023) for BO over molecules (Ramos et al., 2023) and hyperparameter tuning (Liu et al., 2024). However, their uncertainty estimates are obtained only through heuristics, such as from the softmax probabilities of the generated answer tokens, coming from point-estimated non-Bayesian LLMs. These non-Bayesian uncertainties thus might not be optimal for the exploration-exploitation tradeoff that is so crucial for BO (Garnett, 2023).

In this work, we take a dispassionate look at LLMs for BO over molecules. We do so by carefully constructing and studying two kinds of surrogate models that are amenable to a principled Bayesian treatment (see Figure 2). First, we treat the LLM as a fixed feature extractor to test whether its pretrained embbeddings are already useful for BO over molecules. Second, we measure to what degree the "stochastic parrot" can be "taught"—via parameter-efficient finetuning methods (PEFT) (e.g., Houlsby et al., 2019; Li & Liang, 2021; Hu et al., 2022) and the Laplace approximation (MacKay, 1992a; Daxberger et al., 2021)—to perform efficient Bayesian exploration in the molecular space.

In sum, our contribution is four-fold:

(a) We study the out-of-the-box usefulness of pretrained LLMs for material discovery by using their last-layer embeddings in BO.

(b) We study whether finetuning through PEFT and then applying approximate Bayesian inference over it is worth the effort in terms of the BO performance.

(c) We provide an easy-to-use software library for principled BO on discrete space with LLMs; see title page.[1]

(d) Through our extensive experiments (8 real-world chemistry problems, 8 recent LLMs—including Llama-2—and non-LLM based features), we provide insights on whether, when, and how "stochastic parrots" can be useful to drive better scientific discovery.

Limitations Our focus in this work is to study LLMs for discrete BO on a predetermined set of molecules, as usually done in real-world chemistry labs (Strieth-Kalthoff et al., 2024, etc.). We leave the study of BO on continuous space with LLM-based Bayesian surrogates as future work. Finally, we focus only on chemistry, although our experiments can also be done for other domains.

2. Preliminaries

Here, we introduce key concepts in Bayesian optimization, Bayesian neural networks, and large language models.

2.1. Bayesian optimization

Suppose $f : \mathcal{X} \to \mathcal{Y}$ is a function that is not analytically tractable and/or very expensive to evaluate. We would like (without loss of generality) to find $\boldsymbol{x}_* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$. For example, we might want to find a new drug $\boldsymbol{x}$ in the

[1]Supplementary experiment code can be found on https://github.com/wiseodd/llm-bayesopt-exps.

Algorithm 1 BO over a pool of molecules.

Input:
Hard-to-evaluate function f; surrogate function g; candidate molecules $\mathcal{D}_{\text{cand}} = \{\boldsymbol{x}_i\}_{i=1}^n$; initial dataset $\mathcal{D}_1 = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^m$; time budget T.

for $t = 1, \ldots, T$ **do**
  Compute posterior pred. $p(g_t \mid \mathcal{D}_t)$ ▷ E.g. via GP or LLA
  $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{D}_{\text{cand}}} \alpha(p(g_t(\boldsymbol{x}) \mid \mathcal{D}_t))$
  Compute $f(\boldsymbol{x}_t)$
  $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\boldsymbol{x}_t, f(\boldsymbol{x}_t))\}$
  $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{cand}} \setminus \{\boldsymbol{x}_t\}$
end for
return $\arg\max_{(\boldsymbol{x}, f(\boldsymbol{x})) \in \mathcal{D}_{T+1}} f(\boldsymbol{x})$

space of all drugs $\mathcal{X}$ that has high efficacy over the population $f(\boldsymbol{x})$. An increasingly common way to approach this problem is to perform Bayesian optimization (BO). The key components of BO are: (i) a *surrogate function* g that tractably approximates f; (ii) a prior belief and a likelihood (and hence a posterior) over g;[2] and (iii) an acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$ that implicitly defines a policy for choosing which $\boldsymbol{x} \in \mathcal{X}$ to evaluate f at. The *expressiveness* of g dictates how accurately we can approximate f; and the *calibration* of the posterior (predictive) distribution $p(g_t \mid \mathcal{D}_t)$ at step t under previous observations $\mathcal{D}_t := \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{t-1}$ dictates where we should explore and where we should exploit in $\mathcal{X}$. This exploration-exploitation balance is the driving force behind the effectiveness of BO in finding the optimum $\boldsymbol{x}_*$ in a reasonable amount of time.

The *de facto* choices of $p(g_t \mid \mathcal{D}_t)$ are Gaussian processes (GPs, Rasmussen & Williams, 2006); although Bayesian neural networks (NNs) have also been increasingly used (Kim et al., 2022; Kristiadi et al., 2023; Li et al., 2024). In the case of GPs, prior knowledge about the function f is injected into g through the choice of prior covariance (also known as the kernel function). For NN-based surrogates, prior knowledge is determined through the choice of architecture (Kim et al., 2022), a weight-space prior (Fortuin et al., 2022), or through the usage of pretrained features (Ranković & Schwaller, 2023). Finally, common choices for α are expected improvement (EI, Jones et al., 1998), upper-confidence bound (UCB, Auer et al., 2002), and Thompson sampling (TS, Thompson, 1933).

2.1.1. BO in chemical space

The discovery of molecular materials represents an optimization problem in a search space of discrete molecules that is estimated to contain at least 10^{100} unique molecules (Restrepo, 2022). At the same time, the practical accessibility of this space is severely limited. To date, only about 10^8 molecules have been reported experimentally, and the synthesis of molecules has unanimously been described as the bottleneck of molecular materials discovery. This applies to autonomous discovery in particular, where the limited robotic action space and the availability of reactants and reagents are additional constraints (Tom et al., 2024).

Experimental discovery campaigns have usually constrained the search space to much smaller sets of accessible, synthesizable molecules. Let $\mathcal{D}_{\text{cand}}$ be such a set of candidate molecules. The BO problem can then be treated as an optimization over a finite discrete set where $\mathcal{X} = \mathcal{D}_{\text{cand}}$—see Algorithm 1. Note that in this case, we do not need to perform continuous optimization (e.g., via SGD) to maximize the acquisition function α—we can simply enumerate all molecules and pick the maximum. While this approach can be expensive when $|\mathcal{D}_{\text{cand}}|$ is large, it is parallelizable and easier than continuous optimization.

Nowadays, contract research organizations offer virtual, synthesizable libraries comprising billions of molecules, and offer synthesis-on-demand services (Enamine Ltd., 2023; Gorgulla et al., 2023). In drug discovery, these libraries serve as the foundation for virtual screening efforts (Shoichet, 2004; Schneider, 2010; Pyzer-Knapp et al., 2015; Lyu et al., 2019), in which the library is sequentially filtered using progressively more costly computational tools, and the remaining candidates are evaluated experimentally. BO has been used for finding optimal candidates in a virtual library, both with simulated (Zhang & Lee, 2019; Korovina et al., 2020; Häse et al., 2021; Hickman et al., 2022; Griffiths et al., 2023) and experimental (Strieth-Kalthoff et al., 2024; Angello et al., 2023) objectives.

2.2. Bayesian neural networks

Let $g : \mathcal{X} \times \Theta \to \mathcal{Y}$ defined by $(\boldsymbol{x}, \boldsymbol{\theta}) \mapsto g_{\boldsymbol{\theta}}(\boldsymbol{x})$ be a neural network (NN). The main premise of Bayesian neural nets (BNNs) is to approximate the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ over the parameters of g via a simpler distribution that encodes uncertainty on $\Theta \subseteq \mathbb{R}^P$. The standard point estimate:

$$\boldsymbol{\theta}_* = \arg\max_{\boldsymbol{\theta} \in \Theta} \underbrace{\log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})}_{= \log p(\boldsymbol{\theta} \mid \mathcal{D}) - \text{const}}, \tag{1}$$

with the log-likelihood loss $\log p(\mathcal{D} \mid \boldsymbol{\theta})$ and a regularizer $\log p(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$ can be seen as a Dirac distribution on Θ. However, it is *not* a BNN since it has zero uncertainty according to any standard metric (variance, entropy, etc.).

2.2.1. Laplace approximations

One of the simplest BNNs is the Laplace approximation (LA, MacKay, 1992b), which has been increasingly used for BO (Kristiadi et al., 2023; Li et al., 2024). Given a (local) maximum $\boldsymbol{\theta}_*$, the LA fits a Gaussian $q(\boldsymbol{\theta} \mid \mathcal{D}) := \mathcal{N}(\boldsymbol{\theta}_*, \boldsymbol{\Sigma}_*)$ centered at $\boldsymbol{\theta}_*$ with covariance given by the inverse-Hessian $\boldsymbol{\Sigma}_* = (-\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \mathcal{D})|_{\boldsymbol{\theta}_*})^{-1}$.

[2] In literature, they are often collapsed into a single notation $p(f \mid \mathcal{D})$. Here, we distinguish f and g for clarity.

A popular instantiation of the LA is the linearized Laplace approximation (LLA Immer et al., 2021), which approximates the Hessian via the Gauss-Newton matrix (Botev et al., 2017) and performs a linearization $g^{\text{lin}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = g_{\boldsymbol{\theta}_*}(\boldsymbol{x}) + \boldsymbol{J}_*(\boldsymbol{x}) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_*)$ of the NN over $\boldsymbol{\theta}$. Here, $\boldsymbol{J}_*(\boldsymbol{x})$ is the Jacobian matrix $(\partial g / \partial \boldsymbol{\theta}|_{\boldsymbol{\theta}_*})$ of the network at $\boldsymbol{\theta}_*$. Note that, the network function $\boldsymbol{x} \mapsto g^{\text{lin}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is still *non*-linear. Crucially, due to the linearity of g over $\boldsymbol{\theta}$ and the Gaussianity of $\boldsymbol{\theta}$, the output/predictive distribution $p(g^{\text{lin}}(\boldsymbol{x}) \mid \mathcal{D}) = \int g^{\text{lin}}_{\boldsymbol{\theta}}(\boldsymbol{x})\, q(\boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta}$ is also Gaussian, given by

$$p(g^{\text{lin}}(\boldsymbol{x}) \mid \mathcal{D}) = \mathcal{N}\left(g_{\boldsymbol{\theta}_*}(\boldsymbol{x}), \boldsymbol{J}_*(\boldsymbol{x}) \boldsymbol{\Sigma} \boldsymbol{J}_*(\boldsymbol{x})^\top\right). \quad (2)$$

In fact, $p(g^{\text{lin}} \mid \mathcal{D})$ is a GP with a mean function given by the NN g and a covariance function that is connected to the empirical neural tangent kernel (Jacot et al., 2018). These facts make the LLA intuitive yet powerful: it adds an uncertainty estimate to the original NN prediction $g_{\boldsymbol{\theta}_*}(\boldsymbol{x})$.[3]

Furthermore, the hyperparameters that dictate the NN prior (which we denote as $\boldsymbol{\gamma}$) can be tuned via the LA's marginal-likelihood approximation (Daxberger et al., 2021):

$$Z(\boldsymbol{\gamma}) = \log p(\boldsymbol{\theta}_* | \mathcal{D}; \boldsymbol{\gamma}) + \tfrac{P}{2} \log 2\pi + \tfrac{1}{2} \log |\boldsymbol{\Sigma}_*(\boldsymbol{\gamma})|\,, \quad (3)$$

where we have made the dependency of the posterior and the Hessian on the hyperparameters $\boldsymbol{\gamma}$ explicit. For example, $\boldsymbol{\gamma}$ could contain the weight decay strength (corresponding to the prior precision of the Gaussian prior on Θ) as well as the noise strength in the likelihood of g.

2.3. Large language models

A crucial component of the recent development in large NNs is the K*-head self-attention* mechanism (Vaswani et al., 2017). Given a length-T sequence of input embeddings of dimension N, say $\boldsymbol{X} \in \mathbb{R}^{T \times N}$, it computes

$$\begin{aligned} \boldsymbol{O} &= [\boldsymbol{H}_1, \dots, \boldsymbol{H}_K] \boldsymbol{W}_o^\top \in \mathbb{R}^{T \times O}, \\ \boldsymbol{H}_i &= s\left(\tfrac{1}{\sqrt{D}} (\boldsymbol{X}\boldsymbol{Q}_i^\top)(\boldsymbol{X}\boldsymbol{K}_i^\top)^\top\right)(\boldsymbol{X}^\top \boldsymbol{V}_i) \in \mathbb{R}^{T \times D}, \end{aligned} \quad (4)$$

where $[\dots]$ is a column-wise stacking operator, taking K-many $T \times D$ matrices to a $T \times KD$ matrix; $\boldsymbol{W}_o \in \mathbb{R}^{O \times KD}$ and $\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i \in \mathbb{R}^{D \times N}$ are linear projectors; and the softmax function $s(\cdot)$ is applied row-wise.

The resulting network architecture, obtained by stacking multiple attention modules along with other layers like residual and normalization layers, is called a *transformer*. When used for language modeling, the resulting model is called a *large language model (LLM)*. The output $\boldsymbol{O}$ of the last transformer module can then be used as a feature for a dense output layer $\mathsf{Head} : \mathbb{R}^O \to \mathbb{R}^C$, taking the row-wise aggregate (e.g. average) of $\boldsymbol{O}$ to a C-dimensional vector, where C is the number of outputs in the problem. For natural language generation, C equals the size of the vocabulary $\mathcal{V}$, e.g. around 32,000 in Touvron et al. (2023a). One can also modularly replace this head so that the LLM can be used for different tasks, e.g. single-output regression where $C = 1$.

2.3.1. Parameter-Efficient Fine-Tuning

Due to their sheer size, the cost of training LLMs from scratch is prohibitively expensive even for relatively small models (Sharir et al., 2020). Thankfully, LLMs are usually trained in a task-agnostic manner and have been shown to be meaningful, generic "priors" for natural-language-related tasks (Brown et al., 2020). One can simply finetune a pretrained LLM to obtain a domain-specific model (Sun et al., 2019). However, standard finetuning, i.e. further optimizing *all* the LLM's parameters, is expensive. Parameter-efficient fine-tuning (PEFT) methods—which add a few additional parameters $\boldsymbol{\omega}$ to the LLM and keep the original LLM parameters frozen—have therefore become standard.

A popular example of PEFT is LoRA (Hu et al., 2022), which uses a bottleneck architecture to introduce additional parameters in a LLM. Let $\boldsymbol{W}_* \in \mathbb{R}^{D \times N}$ be an attention weight matrix. LoRA freezes $\boldsymbol{W}_*$ and augments it into

$$\boldsymbol{W} = \boldsymbol{W}_* + \boldsymbol{B}^\top \boldsymbol{A}; \qquad \boldsymbol{A} \in \mathbb{R}^{Z \times N}, \boldsymbol{B} \in \mathbb{R}^{Z \times D}. \quad (5)$$

If Z is relatively small, the matrices $\boldsymbol{A}, \boldsymbol{B}$ will introduce just a few additional parameters. Note that, many other PEFT methods are also commonly used in practice, e.g., Adapter (Houlsby et al., 2019), Prefix Tuning (Li & Liang, 2021), IA3 (Liu et al., 2022), etc. That is LoRA, is not the only choice for performing PEFT.

3. Experiment Setup

Equipped with the necessary background knowledge from Section 2, we now discuss our experiments to test whether LLMs are good for BO in molecular discovery. We refer the reader to Algorithm 1 for the concrete problem statement.

Datasets We evaluate the models considered (see below) on the following datasets that represent realistic problem sets from molecular materials discovery: (i) minimizing the redox potential (**redoxmer**) and (ii) minimizing the solvation energy (**solvation**) of possible flow battery electrolytes (Agarwal et al., 2021), (iii) minimizing the docking score of **kinase** inhibitors for drug discovery (Graff et al., 2021), (iv) maximizing the fluorescence oscillator strength of **lasers** (Strieth-Kalthoff et al., 2024), (v) maximizing the power conversion efficiency (PCE) of **photovoltaics** materials (Lopez et al., 2016), and (vi) maximizing the π-π^* transition wavelength of organic **photoswitches** (Griffiths et al., 2022). These problems cover a variety of molecular physical properties and therefore represent a diverse set of

[3] Non-Gaussian output distributions can also be constructed, see e.g. (Kristiadi et al., 2022; Bergamin et al., 2023).

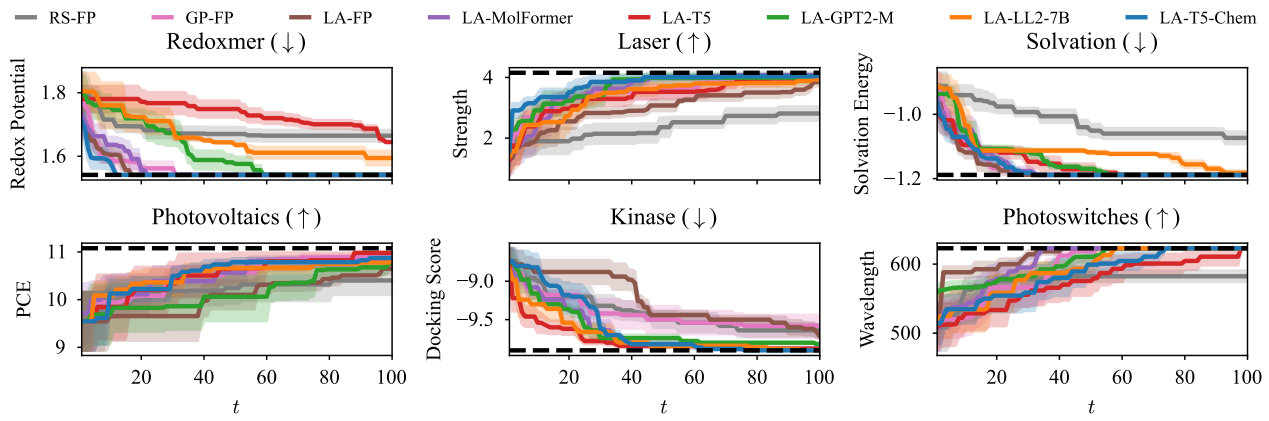


Figure 3. LLM as fixed feature extractors in BO over molecules. See Figure 2(a) for the model schematic.

molecular design tasks. For each virtual library of molecules above, we use physics-inspired simulators proposed by the respective authors as the ground truth functions $f(\boldsymbol{x})$.[4]

Features and LLMs As non-LLM baselines, we use 1024-bit Morgan **fingerprints** (Morgan, 1965) as a chemistry-specific algorithmic vectorization scheme, and the feature vectors from the pretrained **MolFormer** transformer (Ross et al., 2022). Meanwhile, for the general-purpose LLMs, we use various recent architectures of varying sizes: T5-Base (**T5**, Raffel et al., 2020), GPT-2-Medium (**GPT2-M**, Radford et al., 2019), and Llama-2-7b (**LL2-7B**, Touvron et al., 2023a). Finally, we use the work of Christofidellis et al. (2023, **T5-Chem**) to represent domain-specific LLMs.

Prompts For text-based surrogate models, we consider several prompting functions $c(\boldsymbol{x})$ that map molecules $\boldsymbol{x}$ to sentences. They are (i) **just-smiles** which contains just the SMILES (Weininger, 1988) representation of $\boldsymbol{x}$, (ii) **completion** which treats the predicted $f(\boldsymbol{x})$ as a completion to the sentence, (iii) **naive** which asks the LLM for $f(\boldsymbol{x})$, and (iv) **single-number** which augments *naive* with an additional prompt to the LLM to only output numbers. (Details in Appendix A.4.) Unless specified explicitly, the default prompt we use is *just-smiles*.

Evaluation In addition to measuring BO performance via the problem-specific optimum values over time, we use the GAP metric (Jiang et al., 2020) which provides a normalized (i.e., problem-independent) counterpart. The GAP metric is useful since it allows us to compare and aggregate performance across datasets. For multiobjective BO experiments, we measure performance using the standard hypervolume metric (Zitzler, 1999) which computes the volume of the current Pareto front found by the surrogate.

[4]We refer the reader to Appendix A.2 for further details regarding all the datasets we use.

4. How Informative are Pretrained LLMs?

First, we study the out-of-the-box, non-finetuned capability of LLMs for BO. To this end, we treat an LLM as a *fixed* feature extractor: Given a pretrained LLM, we remove its language-modeling head and obtain the function $\phi_{\boldsymbol{W}_*}$, mapping a textual context $c(\boldsymbol{x})$ of a molecule $\boldsymbol{x}$ into its final transformer embedding vector $\phi_{\boldsymbol{W}_*}(c(\boldsymbol{x})) \in \mathbb{R}^H$. We can then apply a standard surrogate model $g_{\boldsymbol{\theta}} : \mathbb{R}^H \to \mathbb{R}$ like GPs or BNNs on $\mathbb{R}^H$. See Figure 2(a) for the illustration and Algorithm 2 in Appendix A for the BO loop.

We use two commonly-used surrogate models over the fixed LLM and non-LLM features: (i) a GP with the Tanimoto and the Matérn kernels for the fingerprints and LLM/MolFormer features, respectively (Griffiths et al., 2023), and (ii) a Laplace-approximated 3-layer ReLU NN with 50 hidden units on each layer, following the finding of Li et al. (2024). The Thompson sampling acquisition function is used in all experiments due to its simplicity and increasing ubiquity in chemistry applications (Hernández-Lobato et al., 2017). Refer to Appendix B.2 for results with the expected improvement acquisition function.

4.1. General or domain-specific LLMs?

We present our first set of results in Figures 3 and 4 (the latter figure summarizes the former). First, LA surrogates are competitive with or better than GP surrogates on the majority of the problems when using fingerprint features. Thus we only consider LA surrogates for LLM features. See also Figure 10 in Appendix B.1 for additional comparisons between the LA and GP which further support our decision in mainly using the LA for the rest of our experiments.

We note that features obtained from general-purpose LLMs (T5, GPT-2-M, and Llama-2-7b) tend to underperform the simple fingerprints baseline. This indicates that although

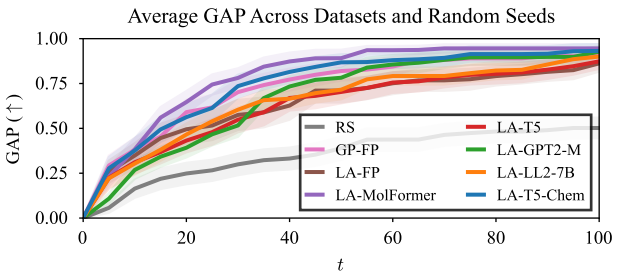


Figure 4. Summarized performance of the results in Figure 3 in terms of the GAP metric. Chemistry-focused features (T5-Chem, MolFormer, and even fingerprints) are better than general-purposed LLM features.

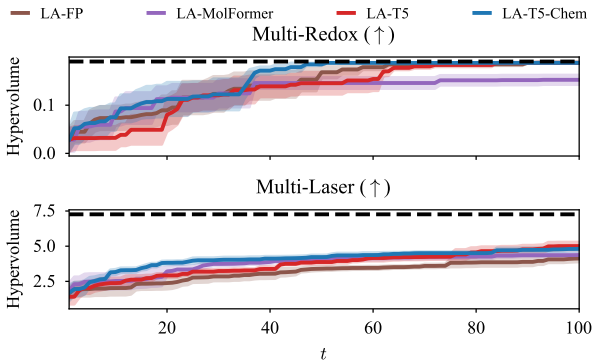


Figure 5. Multiobjective BO performance in terms of the standard hypervolume metric.

general-purpose LLMs seem to "understand" chemistry as illustrated in Figure 1, the *features* encoded by these LLMs are less informative for chemistry-focused BO. While this conclusion holds in our specific problem setup, we note that other studies find LLMs to be useful for more general problems (Gruver et al., 2023; Han et al., 2023).

Meanwhile, chemistry-specific transformer features (T5-Chem, MolFormer), are generally better-suited than the general-purpose LLM features. However, we note that T5-Chem LLM features perform slightly worse on average than the non-LLM MolFormer features. Considering that T5-Chem is larger than MolFormer (220M vs. 44M parameters) and that MolFormer is trained using more chemistry data (100M vs. 33M), this finding may indicate that domain-specific pretraining data matters more than the natural language capability of a transformer model.

> Domain-specific transformers are useful as feature extractors in BO over molecules. They tend to outperform general-purpose LLMs and traditional fingerprint features. However, this may be due to the transformer's capacity and the chemistry-specific pretraining data, not so much its natural language capability.

4.2. Multiobjective optimization

In addition to the single-objective problems in Sections 4 and 5, we perform multiobjective BO experiments by (i) combining both objectives in the flow battery problem above, and (ii) adding an extra maximization objective (electronic gap) to the laser problem. We refer to these problems as **multi-redox** and **multi-laser**, respectively.

To accommodate the additional objectives, we cast the problems as multi-output regression problems—for each $\boldsymbol{x}$, the posterior of $g(\boldsymbol{x})$ is thus a multivariate Gaussian over $\mathbb{R}^C$ where C is the number of the objectives. For the acquisition function, we use the scalarized Thompson sampling (Paria et al., 2020) with a fixed, uniform weighting.

The results, in terms of the standard hypervolume (under the estimated Pareto frontier) evaluation metric (Zitzler, 1999), are provided in Figure 5. We found that T5-Chem performs best overall. When t is small, the non-LLM chemistry-specific transformer MolFormer is better than T5 and LA. However, MolFormer underperforms both LLMs at the latter stages of the optimization. We hypothesize that the smaller model size of MolFormer (see the preceding section) might contribute to this underperformance. In any case, our conclusion here is consistent with the previous section.

4.3. Effects of prompting

Here we test how prompting affects BO performance, comparing the prompts described in Section 3. We present the results for the redoxmer and photoswitches tasks in Figure 6 (see Figure 11 in the appendix for the rest of the problems).

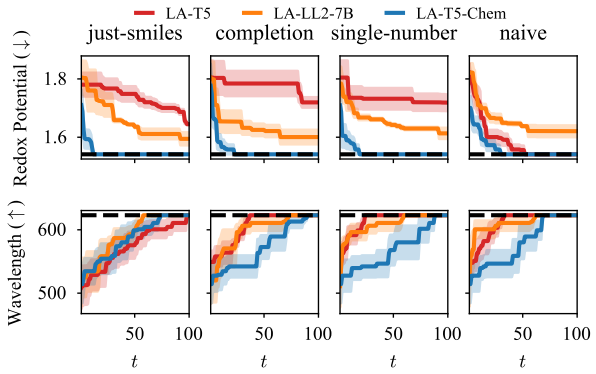


Figure 6. BO results across prompts. **Top:** Redoxmer, **bottom:** Photoswitches. Results for the other datasets are in Appendix A.4.

Prompting does indeed make a difference: unlike general LLMs (T5, Llama-2-7b), the chemistry-specific T5-Chem works best when the prompt is simply the SMILES string

itself. Nevertheless, we note that T5-Chem obtains the best performance in most of the problems considered and across all prompts—see both Figures 6 and 11. Thus the chemistry-specific T5-Chem both yield better BO performance while not requiring prompt engineering.

In Figures 15 and 16 (Appendix B), we show results with IUPAC representation of molecules instead of SMILES. Note that IUPAC strings are closer to natural language than SMILES. E.g., the molecule H_2SO_4 has IUPAC name "sulfuric acid" and SMILES representation "OS(=O)(=O)O". We draw a similar conclusion as in the preceding section that the choice of which representation to use is LLM-dependent. For T5-Chem, SMILES is preferable, consistent with how it was pretrained (Christofidellis et al., 2023).

> Prompting does impact BO performance. It is preferable to stick with a prompt that is close to the one used for pretraining the LLM.

4.4. The case of in-context learning

Finally, we compare Laplace-approximated surrogate models against the recently proposed in-context learning (ICL) optimizer method of Ramos et al. (**BO-LIFT**, 2023). BO-LIFT works purely by prompting chat-based models such as GPT-4 (OpenAI, 2023) and Llama-2-7b (Touvron et al., 2023b, the chat version). See Appendix A.5 for details.

We note that the uncertainty estimates yielded by BO-LIFT are obtained based on the variability in the decoding steps of the LLM. They are thus not Bayesian since they still arise from a point-estimated model. In contrast, all the Bayesian surrogates we consider in this work approximate the posterior distribution over the LLM's weights.

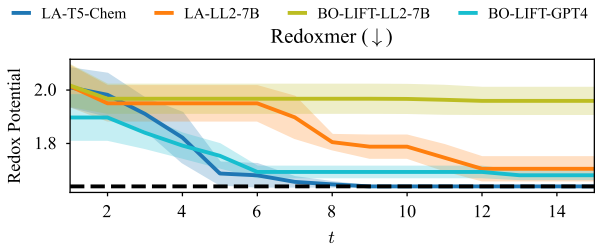


Figure 7. Fixed-feature BO surrogates vs. the in-context-learning optimizer of (Ramos et al., 2023) on the Redoxmer dataset.

We present the result of the subsampled Redoxmer dataset in Figure 7 ($|\mathcal{D}_{\text{cand}}| = 200, |\mathcal{D}_1| = 5, T = 15$ using the notation of Algorithm 1). We subsample due to the cost of the experiment, as discussed below. We find that BO-LIFT is ineffective when combined with Llama-2-7b. Meanwhile, it performs much better with GPT-4, indicating that ICL may require a very large, expensive LLM. Indeed, each optimization run costs between \$12-\$18 USD for GPT-4, totaling to \$75.81 USD over 5 random seeds.

In contrast, using a chemistry-specific, small (200M parameters) T5-Chem as a feature extractor for a principled BO surrogate is better *and* much cheaper. Indeed, T5-Chem can be run on even mid-range consumer-grade GPUs and the LA or GP surrogate's training can be done on CPUs.

> A chemistry-specific LLM combined with a principled Bayesian surrogate is preferable to an ICL optimizer, both in terms of performance and cost.

5. How Useful are Finetuned LLMs?

In the previous section, we have seen that we can use an LLM as a *fixed* feature extractor in a BO loop with standard surrogate models. Here, we answer the question of whether or not treating the whole LLM itself as the surrogate model improves BO. The hypothesis is that we can improve BO performance by performing feature learning—adapting the LLM feature to the problem at hand.

How should we compute the posterior $p(g_t \mid \mathcal{D}_t)$ of a LLM with learned features? Let $g_{\boldsymbol{\theta}} = \mathsf{Head}_{\boldsymbol{w}} \circ \varphi_{\boldsymbol{W}}$ be an LLM feature extractor with weights $\boldsymbol{W}$ composed with a regression head $\mathsf{Head}_{\boldsymbol{w}}$ with weights $\boldsymbol{w}$. Here, $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{w}\}$. Given the dataset $\mathcal{D}_t$ at time t, the seemingly most straightforward way to perform finetuning on $g_{\boldsymbol{\theta}}$ and obtaining its posterior is to perform a Bayesian update on the previous posterior of $\boldsymbol{W}$ using new observations in $\mathcal{D}_t$ (Shwartz-Ziv et al., 2022). However, this requires an existing posterior over $\boldsymbol{W}$ in the first place, which can be *very* costly to obtain.

Recall that a more tractable way to introduce feature learning is to leverage PEFT. However, unlike full finetuning, it does not admit a straightforward Bayesian posterior update on $\boldsymbol{W}$. In particular, how should one incorporate the pretrained LLM weights $\boldsymbol{W}_*$ into the Bayesian inference over the PEFT weights? Here, we generalize the work of Yang et al. (2023)—which applies the LA on LoRA weights—to make it compatible with any general PEFT method.

We treat the original LLM weights $\boldsymbol{W}$ as hyperparameters and perform Bayesian inference only on $\boldsymbol{w}$ and the PEFT weights $\boldsymbol{\omega}$. Specifically, let $g_{\boldsymbol{\theta}} = \mathsf{Head}_{\boldsymbol{w}} \circ \varphi_{\boldsymbol{\omega}, \boldsymbol{W}_*}$ with $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{\omega}\}$ be the new surrogate model, and $\boldsymbol{W}_*$ be the pretrained LLM weights. We define the posterior:[5]

$$p(\boldsymbol{\theta} \mid \mathcal{D}; \boldsymbol{W} = \boldsymbol{W}_*) \propto p(\boldsymbol{\theta}; \boldsymbol{W} = \boldsymbol{W}_*) \times p(\mathcal{D} \mid \boldsymbol{\theta}; \boldsymbol{W} = \boldsymbol{W}_*). \quad (6)$$

[5]The dependence of the prior $p(\boldsymbol{\theta}; \boldsymbol{W} = \boldsymbol{W}_*)$ on $\boldsymbol{W}$ is useful, e.g. for initialization (Li & Liang, 2021).

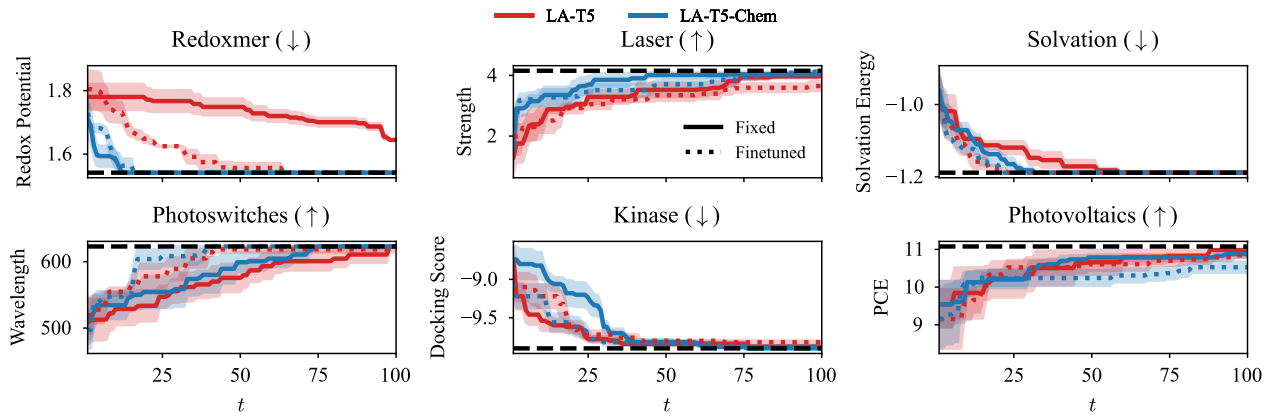


Figure 8. Finetuned LLMs as surrogate models compared against the fixed-feature surrogates from Section 4.

In other words, the pretraining weights $\boldsymbol{W}_*$ act as *conditioning variables* on the PEFT, just like any other hyperparameters. The usual training procedure, i.e. finding the optimal PEFT weights $\boldsymbol{\theta}_*$, can then be seen as MAP estimation under this probabilistic framework. While this formulation is rather obvious after the fact, so far, it has not been clarified in previous work. Indeed, a clear interpretation of Bayesian PEFT in relation to Bayesian updating in full finetuning has been missing, even from the work of Yang et al. (2023).

Notice that any Bayesian method can be used to approximate the posterior in (6). In this work, we use the LA: Given the weight-space LA posterior $p(\boldsymbol{\theta} \mid \mathcal{D}; \boldsymbol{W} = \boldsymbol{W}_*)$ over the PEFT weights, we can obtain the PEFT posterior predictive $p(g_t \mid \mathcal{D}_t; \boldsymbol{W}_*)$ via the LLA (2). This step is tractable since the Jacobian matrix $\boldsymbol{J}_*(\boldsymbol{x})$ is only of size $C \times \widehat{P}$, where C is the number of BO objectives (*much* fewer than the language-modeling head's outputs) and $\widehat{P}$ is the PEFT parameters (*much* fewer than the LLM's parameters; often less than 1% (Hu et al., 2022)).

The formulation of (6) highlights an additional benefit of the LA: since $\boldsymbol{W}$ is now a hyperparameter (i.e. it is part of γ in (3)) one can further optimize it (or a subset of it) via the marginal likelihood. This optimization can potentially improve performance,[6] similar to deep kernel learning (Wilson et al., 2016) in the context of GPs. However, it also makes PEFT training as expensive as full finetuning. We leave optimization of $\boldsymbol{W}$ for future work since our present focus is on the usage of LLMs in BO and not on new methods arising from the probabilistic model in (6).

5.1. Are finetuned LLM surrogates preferable?

Following (Yang et al., 2023), we use LoRA as the PEFT method of choice: For each time t, we reinitialize and train the LoRA weights with MAP estimation (1) using the observed molecules $\mathcal{D}_t$, and then apply the LLA to obtain the posterior predictive distribution $p(g_t \mid \mathcal{D}_t)$. See Figure 2(b) and Algorithm 3 for an illustration and pseudocode, respectively. See also Appendix A.3 for the training details. We compare the resulting finetuned surrogates with their fixed-feature surrogate counterparts from Section 4.

In Figures 8 and 9, we show the finetuning results on T5 and T5-Chem, representing general-purpose and chemistry-specific LLMs, respectively. We found that finetuning is indeed beneficial for both cases—notice that it improves the BO performance in most problems compared to the fixed-feature version.

On the flip side, for some tasks, finetuning does not offer significant improvement over fixed-feature LLMs. Moreover, in one problem (Photovoltaics), we found that finetuning decreases the performance of T5-Chem. We attribute this degradation to the fact that we use the same hyperparameters (learning rate, weight decay, etc. of LoRA's SGD and the LA) on all problems. This setup mirrors common practice: it is standard to simply use the default hyperparameters of a BO algorithm provided by a software package such as BoTorch (Balandat et al., 2020). In any case, it is encouraging to see that finetuning generally works well across most BO problems, even when only considering default hyperparameters.

The cost of each BO iteration t is largely bottlenecked by forward passes over the candidate molecules in $\mathcal{D}_{\text{cand}}$ (Algorithm 1, line 3), and *not* the finetuning and Laplace approximation of the surrogate (see Figure 17 in Appendix B). This is because $|\mathcal{D}_{\text{cand}}|$ can be several orders of magnitude larger than $|\mathcal{D}_t|$ during the BO loop, and forward passes on LLMs are generally expensive. Additionally, the LLA posterior (2) requires computing the Jacobian of the network, amounting to the cost of several backward passes, depend-

[6] Notice the dependence of the likelihood on $\boldsymbol{W}$ in (3).

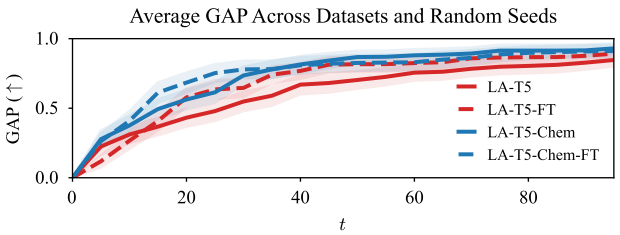


Figure 9. Summarized performance of the finetuning results in Figure 8 in terms of the GAP metric. Finetuning does seem to be beneficial for BO performance.

ing on C. Due to GPU memory limitation, we can only use a minibatch size of 16. We note that fixed-feature surrogates do not have this problem since one can simply cache the LLM's features to be used for all iterations t. Nevertheless, this bottleneck can be alleviated via engineering efforts such as parallelizing the forward passes over $\mathcal{D}_{\text{cand}}$.

> Finetuned LLM surrogates are preferable to their fixed-feature counterparts for both general and domain-specific LLMs. The bottleneck associated with finetuning is forward passes over $\mathcal{D}_{\text{cand}}$ rather than training.

6. Related Work

While LLMs have been leveraged for BO (Ramos et al., 2023; Liu et al., 2024), so far they have only been used in a heuristic manner: uncertainty estimates are obtained from the softmax probabilities outputs of point-estimated (i.e., non-Bayesian) LLMs. Meanwhile, Microsoft Research AI4Science & Microsoft Azure Quantum (2023) study the optimization capability of GPT-4 in a pure prompting scheme without uncertainty estimation of any kind.

The closest work to ours is that of Ranković & Schwaller (2023), who study the usage of text embeddings for BO with GP surrogates. However, the goal of their work differs from ours: We investigate whether the usage of LLMs is justified for BO due to their apparent chemistry question-answering capabilities while (Ranković & Schwaller, 2023) primarily study the usage of text embeddings in BO. Furthermore, they do not study the effect of prompting and finetuning.

The present work can be seen as an extension to the LoRA LA work of Yang et al. (2023). Our work provides a clear probabilistic interpretation of their method, while also generalizing it and using it in the context of BO for chemistry.

Beyond discrete-set optimization, one can cast BO over molecules as a continuous optimization problem with the help of generative models. Gómez-Bombarelli et al. (2018); Tripp et al. (2020); Maus et al. (2022) employ variational autoencoders to construct continuous latent representations of molecules and perform BO in these latent spaces. The surrogate model and the autoencoder can be trained jointly (Stanton et al., 2022; Maus et al., 2022), echoing our finetuning approach in Section 5.1. Unlike these prior works, we focus on studying the role of LLMs in the simpler yet practically relevant setting of discrete-set BO.

7. Conclusion

We have shown that large language models (LLMs) do indeed carry useful information to aid Bayesian optimization (BO) over molecules. However, their usefulness is only apparent when one (i) uses a chemistry-specific LLM or (ii) performs finetuning—preferably both. Indeed, for point (i), even when the recent Llama-2-7b LLM was used as a feature extractor for a BO surrogate or when the state-of-the-art GPT-4 was used in conjunction with in-context learning, the optimization performance was subpar compared to that of a *much* smaller chemistry-focused LLM. We address point (ii) by providing a general way to formulate Bayesian inference for parameter-efficient finetuning (PEFT) methods, which in turn enables principled uncertainty estimation over LLMs with any PEFT method. We find that these principled BO surrogates are effective and yet much cheaper than in-context learning methods since small domain-specific LLMs can be used. We hope that our findings and the accompanying software library can be useful for practitioners and inspire future principled methods around LLMs for scientific discovery, both inside and outside of chemistry. In future work, we aim to characterize the mechanisms underpinning how LLMs induce a capable exploration-exploitation tradeoff.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning for chemistry. We acknowledge the potential dual use of chemistry-specific models (and large language models in general) to search for materials for nefarious purposes. Although the search spaces presented in this work are, to the best of our knowledge, highly unlikely to yield such materials, we recognize the necessity for safeguards in such efforts and encourage open discussions about their development.

Acknowledgments

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. AK thanks Runa Eschenhagen and Alexander Immer for helpful discussions. FSK is a postdoctoral fellow in the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship Program, a program by Schmidt Futures.

References

Agarwal, G., Doan, H. A., Robertson, L. A., Zhang, L., and Assary, R. S. Discovery of energy storage molecular materials using quantum chemistry-guided multiobjective bayesian optimization. *Chemistry of Materials*, 33(20), 2021.

Angello, N., Friday, D., Hwang, C., Yi, S., Cheng, A., Torres-Flores, T., Jira, E., Wang, W., Aspuru-Guzik, A., Burke, M., and et al. Closed-loop transfer enables AI to yield chemical knowledge. *ChemRxiv*, 2023.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47, 2002.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *NeurIPS*, 2020.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *ACM Conference on Fairness, Accountability, and Transparency*, 2021.

Bergamin, F., Moreno-Muñoz, P., Hauberg, S., and Arvanitidis, G. Riemannian Laplace approximations for Bayesian neural networks. In *NeurIPS*, 2023.

Botev, A., Ritter, H., and Barber, D. Practical Gauss-Newton optimisation for deep learning. In *ICML*, 2017.

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

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., and Androutsopoulos, I. LEGAL-BERT: The muppets straight out of law school. In *EMNLP*, 2020.

Chithrananda, S., Grand, G., and Ramsundar, B. ChemBERTa: Large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Christofidellis, D., Giannone, G., Born, J., Winther, O., Laino, T., and Manica, M. Unifying molecular and textual representations via multi-task language modelling. In *ICML*, 2023.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux – effortless Bayesian deep learning. In *NeurIPS*, 2021.

de Regt, H. W. Understanding, values and the aims of science. *Philosophy of Science*, 87(5), 2020.

Enamine Ltd. The Enamine REAL database, 2023. URL https://enamine.net/compound-collections/real-compounds/real-database. Accessed: 2023-11-29.

Fortuin, V., Garriga-Alonso, A., Ober, S. W., Wenzel, F., Rätsch, G., Turner, R. E., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. In *ICLR*, 2022.

Garnett, R. *Bayesian optimization*. Cambridge University Press, 2023.

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

Gorgulla, C., Nigam, A., Koop, M., Çınaroğlu, S. S., Secker, C., Haddadnia, M., Kumar, A., Malets, Y., Hasson, A., Li, M., Tang, M., Levin-Konigsberg, R., Radchenko, D., Kumar, A., Gehev, M., Aquilanti, P.-Y., Gabb, H., Alhossary, A., Wagner, G., Aspuru-Guzik, A., Moroz, Y. S., Fackeldey, K., and Arthanari, H. VirtualFlow 2.0 - the next generation drug discovery platform enabling adaptive screens of 69 billion molecules. *bioRxiv*, 2023.

Graff, D. E., Shakhnovich, E. I., and Coley, C. W. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chemical Science*, 12(22), 2021.

Greenaway, R. L., Jelfs, K. E., Spivey, A. C., and Yaliraki, S. N. From alchemist to AI chemist. *Nature Reviews Chemistry*, 7(8), 2023.

Griffiths, R.-R., Greenfield, J. L., Thawani, A. R., Jamasb, A. R., Moss, H. B., Bourached, A., Jones, P., McCorkindale, W., Aldrick, A. A., Fuchter, M. J., and Lee, A. A. Data-driven discovery of molecular photoswitches with multioutput Gaussian processes. *Chemical Science*, 13 (45), 2022.

Griffiths, R.-R., Klarner, L., Moss, H. B., Ravuri, A., Truong, S., Stanton, S., Tom, G., Rankovic, B., Du, Y., Jamasb, A., Deshwal, A., Schwartz, J., Tripp, A., Kell, G., Frieder, S., Bourached, A., Chan, A., Moss, J., Guo, C., Durholt, J., Chaurasia, S., Strieth-Kalthoff, F., Lee, A. A., Cheng, B., Aspuru-Guzik, A., Schwaller, P., and Tang, J. GAUCHE: A library for Gaussian processes in chemistry. In *NeurIPS*, 2023.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters. In *NeurIPS*, 2023.

Guo, T., Guo, K., Nan, B., Liang, Z., Guo, Z., Chawla, N. V., Wiest, O., and Zhang, X. What can large language models do in chemistry? A comprehensive benchmark on eight tasks. *arXiv preprint arXiv:2305.18365*, 2023.

Han, C., Wang, Z., Zhao, H., and Ji, H. In-context learning of large language models explained as kernel regression. *arXiv preprint arXiv:2305.12766*, 2023.

Hernández-Lobato, J. M., Requeima, J., Pyzer-Knapp, E. O., and Aspuru-Guzik, A. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *ICML*, 2017.

Hickman, R., Parakh, P., Cheng, A., Ai, Q., Schrier, J., Aldeghi, M., and Aspuru-Guzik, A. Olympus, enhanced: Benchmarking mixed-parameter and multi-objective optimization in chemistry and materials science. *ChemRxiv*, 2023.

Hickman, R. J., Aldeghi, M., Häse, F., and Aspuru-Guzik, A. Bayesian optimization with known experimental and design constraints for chemistry applications. *Digital Discovery*, 1(5), 2022.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Häse, F., Aldeghi, M., Hickman, R. J., Roch, L. M., and Aspuru-Guzik, A. Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8(3), 2021.

Immer, A., Korzepa, M., and Bauer, M. Improving predictions of Bayesian neural nets via local linearization. In *AISTATS*, 2021.

Jablonka, K. M., Ai, Q., Al-Feghali, A., Badhwar, S., Bran, J. D., Bringuier, S., Brinson, L. C., Choudhary, K., Circi, D., Cox, S., et al. 14 examples of how LLMs can transform materials science and chemistry: A reflection on a large language model hackathon. *arXiv preprint arXiv:2306.06283*, 2023a.

Jablonka, K. M., Schwaller, P., Ortega-Guerrero, A., and Smit, B. Leveraging large language models for predictive chemistry. *ChemRxiv*, 2023b.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *NIPS*, 2018.

Jiang, S., Chai, H., Gonzalez, J., and Garnett, R. BINOCULARS for efficient, nonmyopic sequential experimental design. In *ICML*, 2020.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of global optimization*, 13, 1998.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al. ChatGPT for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 2023.

Kim, S., Lu, P. Y., Loh, C., Smith, J., Snoek, J., and Soljačić, M. Deep learning for Bayesian optimization of scientific problems with high-dimensional structure. *TMLR*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., and Xing, E. P. ChemBO: Bayesian optimization of small organic molecules with synthesizable recommendations. In *AISTATS*, 2020.

Kristiadi, A., Eschenhagen, R., and Hennig, P. Posterior refinement improves sample efficiency in Bayesian neural networks. In *NeurIPS*, 2022.

Kristiadi, A., Immer, A., Eschenhagen, R., and Fortuin, V. Promises and pitfalls of the linearized Laplace in Bayesian optimization. In *Advances in Approximate Bayesian Inference*, 2023.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Li, Y. L., Rudner, T. G., and Wilson, A. G. A study of Bayesian neural network surrogates for Bayesian optimization. In *ICLR*, 2024.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*, 2022.

Liu, T., Astorga, N., Seedat, N., and van der Schaar, M. Large language models to enhance Bayesian optimization. In *ICLR*, 2024.

Lopez, S. A., Pyzer-Knapp, E. O., Simm, G. N., Lutzow, T., Li, K., Seress, L. R., Hachmann, J., and Aspuru-Guzik, A. The harvard organic photovoltaic dataset. *Scientific Data*, 3(160086), 2016.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Lyu, J., Wang, S., Balius, T. E., Singh, I., Levit, A., Moroz, Y. S., O'Meara, M. J., Che, T., Algaa, E., Tolmachova, K., Tolmachev, A. A., Shoichet, B. K., Roth, B. L., and Irwin, J. J. Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743), 2019.

MacKay, D. J. The evidence framework applied to classification networks. *Neural Computation*, 4(5), 1992a.

MacKay, D. J. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3), 1992b.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Maus, N., Jones, H., Moore, J., Kusner, M. J., Bradshaw, J., and Gardner, J. Local latent space Bayesian optimization over structured inputs. In *NeurIPS*, 2022.

Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using GPT-4. *arXiv preprint arXiv:2311.07361*, 2023.

Močkus, J. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, 1975.

Morgan, H. L. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2), 1965.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Paria, B., Kandasamy, K., and Póczos, B. A flexible framework for multi-objective Bayesian optimization using random scalarizations. In *UAI*, 2020.

Pyzer-Knapp, E. O., Suh, C., Gómez-Bombarelli, R., Aguilera-Iparraguirre, J., and Aspuru-Guzik, A. What is high-throughput virtual screening? A perspective from organic materials discovery. *Annual Review of Materials Research*, 45(1), 2015.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(1), 2020.

Ramos, M. C., Michtavy, S. S., Porosoff, M. D., and White, A. D. Bayesian optimization of catalysts with in-context learning. *arXiv preprint arXiv:2304.05341*, 2023.

Ranković, B. and Schwaller, P. 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.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes in machine learning*. The MIT Press, 2006.

Restrepo, G. Chemical space: limits, evolution and modelling of an object bigger than our universal library. *Digital Discovery*, 1(5), 2022.

Ritter, H., Botev, A., and Barber, D. A scalable Laplace approximation for neural networks. In *ICLR*, 2018.

Ross, J., Belgodere, B., Chenthamarakshan, V., Padhi, I., Mroueh, Y., and Das, P. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12), 2022.

Schneider, G. Virtual screening: An endless staircase? *Nature Reviews Drug Discovery*, 9(4), 2010.

Sharir, O., Peleg, B., and Shoham, Y. The cost of training NLP models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

Shoichet, B. K. Virtual screening of chemical libraries. *Nature*, 432(7019), 2004.

Shwartz-Ziv, R., Goldblum, M., Souri, H., Kapoor, S., Zhu, C., LeCun, Y., and Wilson, A. G. Pre-train your loss: Easy Bayesian transfer learning with informative priors. In *NeurIPS*, 2022.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., and Wilson, A. G. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In *ICML*, 2022.

Strieth-Kalthoff, F., Hao, H., Rathore, V., Derasp, J., Gaudin, T., Angello, N. H., Seifrid, M., Trushina, E., Guy, M., Liu, J., and et al. Delocalized, asynchronous, closed-loop discovery of organic laser emitters. *Science*, 384(6697), 2024.

Sun, C., Huang, L., and Qiu, X. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *NAACL*, 2019.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4), 1933.

Tom, G., Schmid, S. P., Baird, S. G., Cao, Y., Darvish, K., Hao, H., Lo, S., Pablo-Garcia, S., Rajaonson, E. M., Skreta, M., Yoshikawa, N., Corapi, S., Akkoc, G. D., Strieth-Kalthoff, F., Seifrid, M., and Aspuru-Guzik, A. Self-driving laboratories for chemistry and materials sciences. *ChemRxiv*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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

Tripp, A., Daxberger, E., and Hernández-Lobato, J. M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *NeurIPS*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, 2017.

Vig, J., Madani, A., Varshney, L. R., Xiong, C., Socher, R., and Rajani, N. F. BERTology meets biology: Interpreting attention in protein language models. In *ICLR*, 2021.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Katwyk, P. V., Deac, A., Anandkumar, A., Bergen, K., Gomes, C. P., Ho, S., Kohli, P., Lasenby, J., Leskovec, J., Liu, T.-Y., Manrai, A., Marks, D., Ramsundar, B., Song, L., Sun, J., Tang, J., Velickovic, P., Welling, M., Zhang, L., Coley, C. W., Bengio, Y., and Zitnik, M. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972), 2023.

Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1), 1988.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *AISTATS*, 2016.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yang, A. X., Robeyns, M., Wang, X., and Aitchison, L. Bayesian low-rank adaptation for large language models. *arXiv preprint arXiv:2308.13111*, 2023.

Zhang, Y. and Lee, A. A. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical Science*, 10(35), 2019.

Zitzler, E. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Shaker Ithaca, 1999.

Appendices: A Sober Look at LLMs for Material Discovery

A. Additional Details

A.1. Pseudocodes

We present the pseudocode of the BO loop corresponding to Figure 2(a) and Section 4 in Algorithm 2. Meanwhile, the pseudocode corresponding to Figure 2(b) and Section 5 is in Algorithm 3.

Algorithm 2 Using an LLM as a feature extractor in BO.

Input:
Pre-trained LLM feature extractor $\phi_{\boldsymbol{W}_*}$, mapping a context $c(\boldsymbol{x})$ to its last transformer embedding vector $\boldsymbol{h} \in \mathbb{R}^H$; prompting function c mapping a molecule $\boldsymbol{x}$ to a textual prompt; surrogate model $g_{\boldsymbol{\theta}} : \mathbb{R}^H \to \mathbb{R}$; candidate molecules $\mathcal{D}_{\text{cand}} = \{\boldsymbol{x}_i\}_{i=1}^n$; initial dataset $\mathcal{D}_1 = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^m$; time budget T.

for $t = 1, \ldots, T$ **do**
  $\Phi_t = \{(\phi_{\boldsymbol{W}_*}(c(\boldsymbol{x})), f(\boldsymbol{x})) : (\boldsymbol{x}, f(\boldsymbol{x})) \in \mathcal{D}_t\}$
  $p(g_t \mid \mathcal{D}_t) = \texttt{infer}(g_{\boldsymbol{\theta}}, \Phi_t)$ ▷ E.g. via GP or the LLA
  $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{D}_{\text{cand}}} \alpha(p(g_t(c(\boldsymbol{x})) \mid \mathcal{D}_t))$
  $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\boldsymbol{x}_t, f(\boldsymbol{x}_t))\}$
  $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{cand}} \setminus \{\boldsymbol{x}_t\}$
end for
return $\arg\max_{(\boldsymbol{x}, f(\boldsymbol{x})) \in \mathcal{D}_{T+1}} f(\boldsymbol{x})$

Algorithm 3 LLM and PEFT as a surrogate model in BO.

Input:
An LLM with a regression head $g_{\boldsymbol{\theta}}$; prompting function c mapping a molecule $\boldsymbol{x}$ to a textual prompt; candidate molecules $\mathcal{D}_{\text{cand}} = \{\boldsymbol{x}_i\}_{i=1}^n$; initial dataset $\mathcal{D}_1 = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^m$; time budget T.

for $t = 1, \ldots, T$ **do**
  $\mathcal{C}_t = \{(c(\boldsymbol{x}), f(\boldsymbol{x})) : (\boldsymbol{x}, f(\boldsymbol{x})) \in \mathcal{D}_t\}$
  $p(\boldsymbol{\theta} \mid \mathcal{D}_t; \boldsymbol{W} = \boldsymbol{W}_*) = \texttt{LA}(g_{\boldsymbol{\theta}}, \mathcal{C}_t)$ ▷ Consisting of both the training (via SGD) and the Laplace approx. of the PEFT weights
  $p(g_t \mid \mathcal{D}_t) = \texttt{LLA}(p(\boldsymbol{\theta} \mid \mathcal{D}_t; \boldsymbol{W} = \boldsymbol{W}_*))$ ▷ Eq. (2)
  $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{D}_{\text{cand}}} \alpha(p(g_t(c(\boldsymbol{x})) \mid \mathcal{D}_t))$
  $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\boldsymbol{x}_t, f(\boldsymbol{x}_t))\}$
  $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{cand}} \setminus \{\boldsymbol{x}_t\}$
end for
return $\arg\max_{(\boldsymbol{x}, f(\boldsymbol{x})) \in \mathcal{D}_{T+1}} f(\boldsymbol{x})$

A.2. Datasets

Here, we expand the details of the datasets mentioned in the main text:

- **Redoxmer (Agarwal et al., 2021):** 1407 molecules.
- **Solvation (Agarwal et al., 2021):** the same molecules as in Redoxmer.
- **Laser (Strieth-Kalthoff et al., 2024):** 10,000 molecules, subsampled without replacement from the original 182,858 molecules.
- **Photovoltaics (Lopez et al., 2016):** 10,000 molecules, subsampled without replacement from the original 2,320,648 molecules.
- **Kinase (Graff et al., 2021):** 10,449 molecules.
- **Photoswitches (Griffiths et al., 2022):** 392 molecules.
- **Multi-Redox (Agarwal et al., 2021):** 1407 molecules.
- **Multi-Laser (Strieth-Kalthoff et al., 2024):** 10,000 molecules, subsampled without replacement from the original 182,858 molecules.

For all virtual libraries/datasets above, we have a ground-truth label for every molecule. They are obtained (by the corresponding original authors) from physically founded simulations. For instance, quantum chemistry or density functional theory for the materials datasets and molecular dynamics docking for drug discovery. Refer to the respective paper for the details on how the ground-truth labels are computed.

A.3. Training

A.3.1. Fixed-feature surrogates

The following are the training details of the surrogates we used in Section 4. Note that for the LLM features, we average the last transformer embedding of shape `(batch_size, seq_len, embd_dim)` over the second axis, while ignoring padding and `EOS` tokens. We used HuggingFace's `transformers` library (Wolf et al., 2019) for all the LLM-related objects that appeared in this paper.

GP We use BoTorch (Balandat et al., 2020) to construct the surrogate function. The Tanimoto kernel is taken from Gauche (Griffiths et al., 2023). To optimize the marginal likelihood, we use Adam (Kingma & Ba, 2015) with learning rate 0.01 for 500 epochs.

LA Our implementation is based on the `laplace-bayesopt` package `https://github.com/wiseodd/laplace-bayesopt`. The neural net used is a 2-hidden-layer multilayer perceptron with 50 hidden units on each layer along with the tanh activation function for fingerprint features (except for the Photoswitches dataset since we observe a vanishing gradient problem). Otherwise, we use the ReLU activation function. We optimize the network with Adam with learning rate 1×10^{-3} and weight decay 5×10^{-4} for 500 epochs with a batch size of 20. We anneal the learning rate with the cosine annealing scheme (Loshchilov & Hutter, 2017). The Laplace approximation is done *post hoc* and we tune the prior precision with the marginal likelihood for 100 iterations. The Hessian is approximated with a Kronecker structure (Ritter et al., 2018) except for the Llama-2-7b features—for the latter case, we found that the diagonal structure performs better.

A.3.2. Finetuned surrogates

The following are the training details of the surrogates we used in Section 5, i.e. the LA on LoRA weights (Hu et al., 2022; Yang et al., 2023). We jointly trained the LoRA and the regression head with AdamW with batch size 16, learning rate 3×10^{-4} and 1×10^{-3} for the LoRA weights and the regression head's weights respectively (except for Photoswitch where we used 3×10^{-3} and 1×10^{-2} respectively), and weight decay 0.01 for 50 epochs. Then, we further optimized only the regression head with the same hyperparameters for 100 epochs.

We used LoRA with rank 4 without bias on the key and value attention weights, following the original paper, except for GPT2 where we applied LoRA on all attention weights since the GPT2 implementation we used coupled all the attention weights into a single weight matrix. Moreover, the α (scaling) hyperparameter was set to 16. Additionally, we used dropout on LoRA with probability 0.1. See Hu et al. (2022) for the explanation of these hyperparameters. We used HuggingFace's `PEFT` library (Mangrulkar et al., 2022).

The Laplace approximation was done on all LoRA's weights and the head's weights. We used Kronecker-factored Hessian and optimized the layerwise prior precisions (Daxberger et al., 2021) with *post hoc* marginal likelihood for 200 iterations.

A.4. Prompting

We use the following prompts in our experiments (Figures 6 and 11):

- **just-smiles:** ``{smiles_str}''.
- **completion:** ``The estimated {objective_str} of the molecule {smiles_str} is: ''.
- **single-number:** ``Answer with just numbers without any further explanation! What is the estimated {objective_str} of the molecule with the SMILES string {smiles_str}?''.
- **naive:** ``Predict the {objective_str} of the following SMILES: {smiles_str}!''.

The variable `smiles_str` equals the SMILES representation of the molecule at hand, e.g. "OS(=O)(=O)O" for sulfuric acid. The variable `obj_str` has a value the textual description of the problem at hand: "redox potential" for Redoxmer,

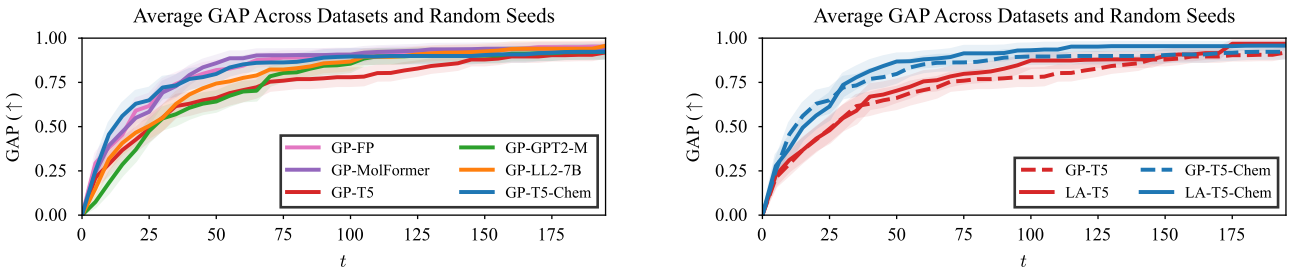


Figure 10. **Left:** Average performance of GP with the Tanimoto kernel on transformer features. **Right:** GP versus LA surrogates in terms.

"solvation energy" for Solvation, "docking score" for Kinase, "fluorescence oscillator strength" for Laser, "Pi-Pi* transition wavelength" for Photoswitches, "power conversion efficiency" for Photovoltaics, "redox potential and solvation energy" for MultiRedox, and "fluorescence oscillator strength and electronic gap" for MultiLaser.

The idea behind this selection of prompts is to cover all the expected inputs of LLMs. For example, GPT-2 and Llama-2 are trained as next-word-prediction models. So, the "completion" prompt might be the most suitable prompt template for them.

A.5. In-context-learning baselines

In Section 4.4, we compared Bayesian surrogates with BO-LIFT, which uses in-context learning in conjunction with non-Bayesian uncertainty estimation (Ramos et al., 2023). To perform prompting, we use their top-k completions template, which selects k previously-seen examples with known properties and appends the SMILES string of the query molecule:

`Q: Given smiles {smiles_str_1}, what is {obj_str}? A:` $\{f(\boldsymbol{x}_1)\}$ `### ... ### Q: Given smiles {smiles_str_k}, what is {obj_str}? A:` $\{f(\boldsymbol{x}_k)\}$ `### Q: Given smiles {smiles_str_query}, what is {obj_str}? A:`

The k-shot examples above are chosen based on the similarity and diversity of their Ada embeddings (part of OpenAI's GPT-4). Then, n completions to the prompt are sampled from the LLM, to compute the mean and variance of $g_t(\boldsymbol{x})$. We use Thompson sampling to make BO-LIFT comparable to the other methods. The values of k and n were set to default values in the BO-LIFT code base, which are both 5.

During our experiment with BO-LIFT (using the authors' code `https://github.com/ur-whitelab/BO-LIFT` to prompt GPT-4), we found that the monetary cost of performing it on the full Redoxmer dataset was already over USD 60 at the third BO iteration $t = 3$. Extrapolating, this means a BO run with $T = 100$ for five different random seeds—as done for other methods considered in this work—will amount to roughly USD 20,000. For this reason, in Section 4.4, we only used a subsampled Redoxmer dataset $|\mathcal{D}_{\text{cand}}| = 200$ for only $T = 15$ BO rounds (repeated over five random seeds). Even in this setup, the cost of BO-LIFT was already USD 75.81.

To circumvent this monetary cost, we also tried ICL using the "chat" version of Llama-2-7b, the most capable free model we could run on a single GPU. However, we also found it difficult to run this model on our entire Redoxamer dataset because it was projected to take approximately 100 hours for $T = 200$ BO rounds. This is another reason why we decided to do the experiment in Section 4.4 with only the subsampled Redoxmer dataset.

All in all, this shows that contrary to popular belief, using principled Bayesian surrogates in a BO over molecules is actually cheaper than using ICL-based non-Bayesian surrogates. As we have shown in the main text, this is because one does not need very large, state-of-the-art LLMs to do BO. One only needs to use small, domain-specific LLMs like that of Christofidellis et al. (2023).

B. Additional Results

In this section, we present additional and/or detailed results to supplement the results in the main text.

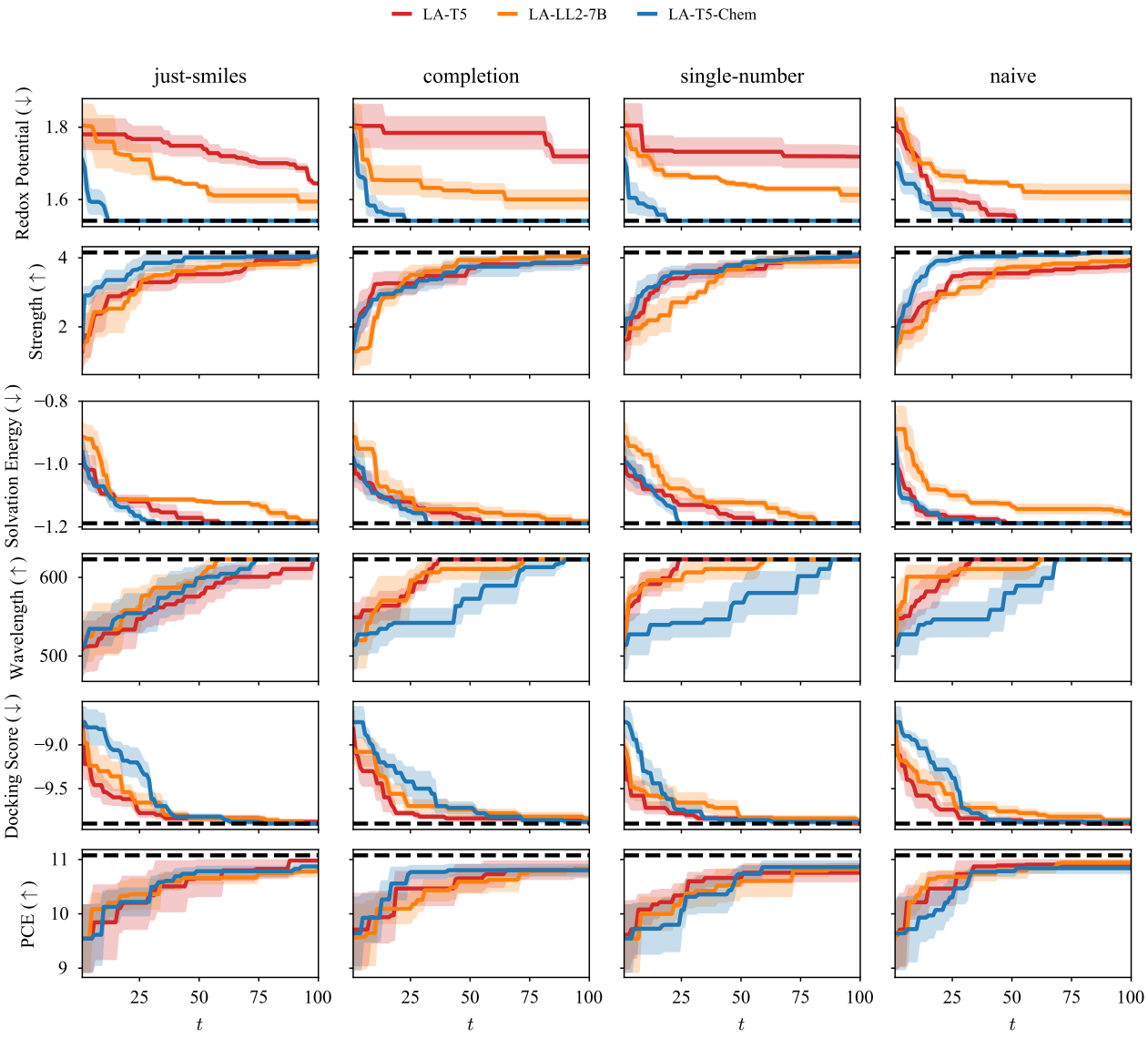
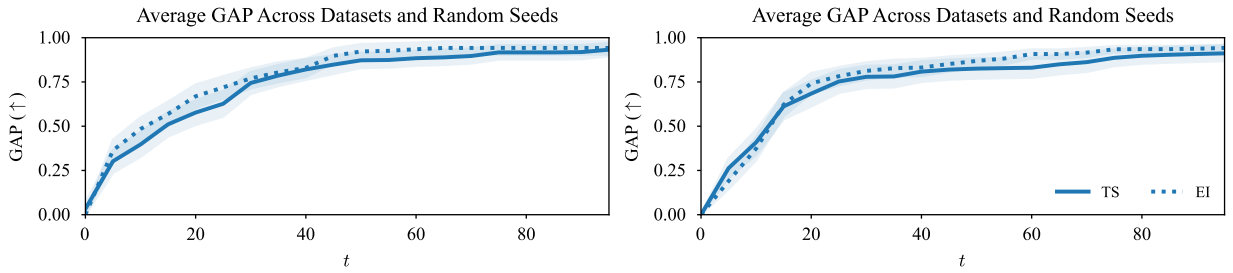


Figure 11. Different prompts.



(a) Fixed

(b) Finetuned

Figure 12. Summarized BO performance with Thompson sampling and expected improvement.

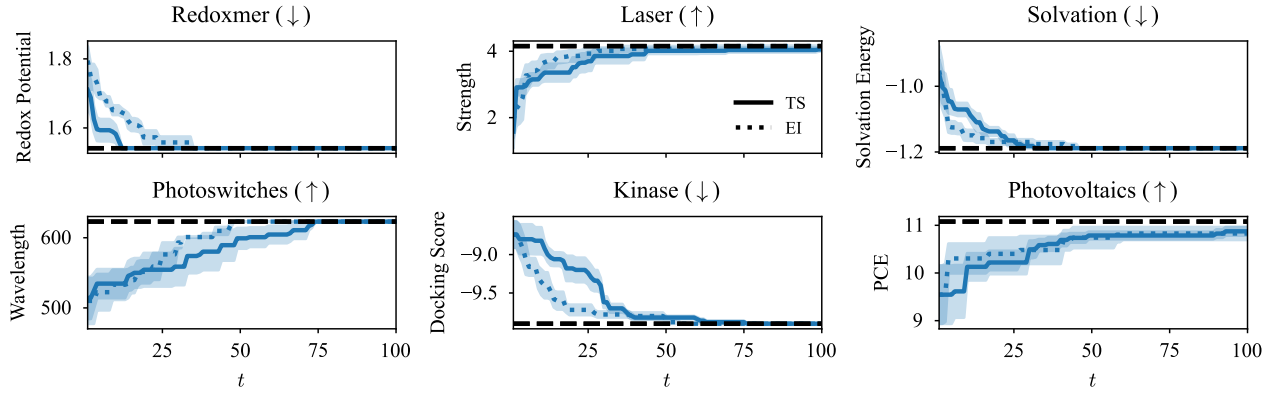


Figure 13. Fixed-feature BO (T5-Chem) with different acquisition functions—Thompson sampling (TS) and expected improvement (EI).

B.1. Laplace vs. GP

We show the performance of the GP (with the Tanimoto kernel) surrogate vis-à-vis LA in Figure 10. Our conclusion is two-fold: (i) We observe a similar ordering as in the main text—chemistry-specific LLMs/transformers are best, although here the performance difference is less pronounced. (ii) The LA surrogate is generally better than the GP one in our tasks.

B.2. Acquisition functions

In Figures 12 to 14 we show the comparison between expected improvement (EI) and Thompson sampling (TS) acquisition functions. We found that, on aggregate, the difference between EI and TS is insignificant (Figure 12, see the error bars). Meanwhile, in some individual cases, their difference can be significant in both directions. So, our suggestion is to do a grid search on a proxy problem whenever possible; otherwise, simply use whichever acquisition function is available.

B.3. Prompts

In Figure 11, we show the effect of the choice of prompt on the BO performance in the fixed-feature (Figure 1(a)) case. We found that prompts do affect BO performance in all datasets considered. Specifically for T5-Chem, the just-smiles prompt template is best. This is consistent with how T5-Chem was finetuned, i.e., using just the SMILES strings as the inputs (Christofidellis et al., 2023). Surprisingly, Llama-2-7b does not gain substantial benefits when different prompt templates are used, even when the "completion" prompt template is used. This indicates that while Llama-2-7b can convincingly answer natural language questions, it does not contain useful information for chemistry (see Figure 1). Finally, we did not see a clear improvement pattern in BO with different prompts using T5. All in all, these findings further support our conclusion that one should use domain-specific LLMs (e.g. T5-Chem) for doing BO.

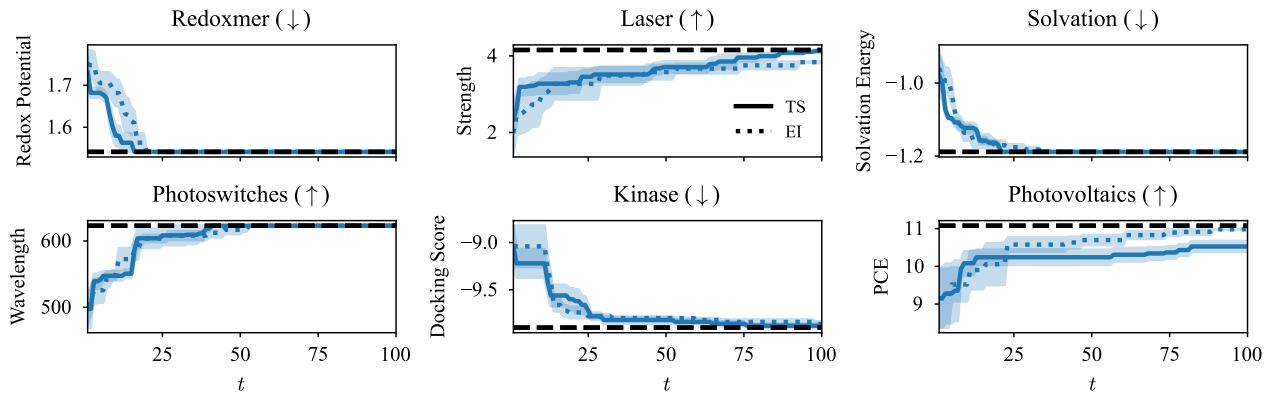


Figure 14. Finetuning T5-Chem with different acquisition functions—Thompson sampling (TS) and expected improvement (EI).

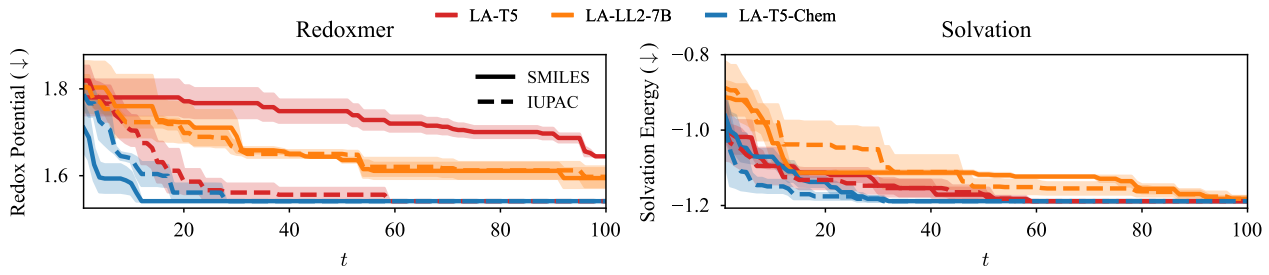


Figure 15. SMILES vs. IUPAC.

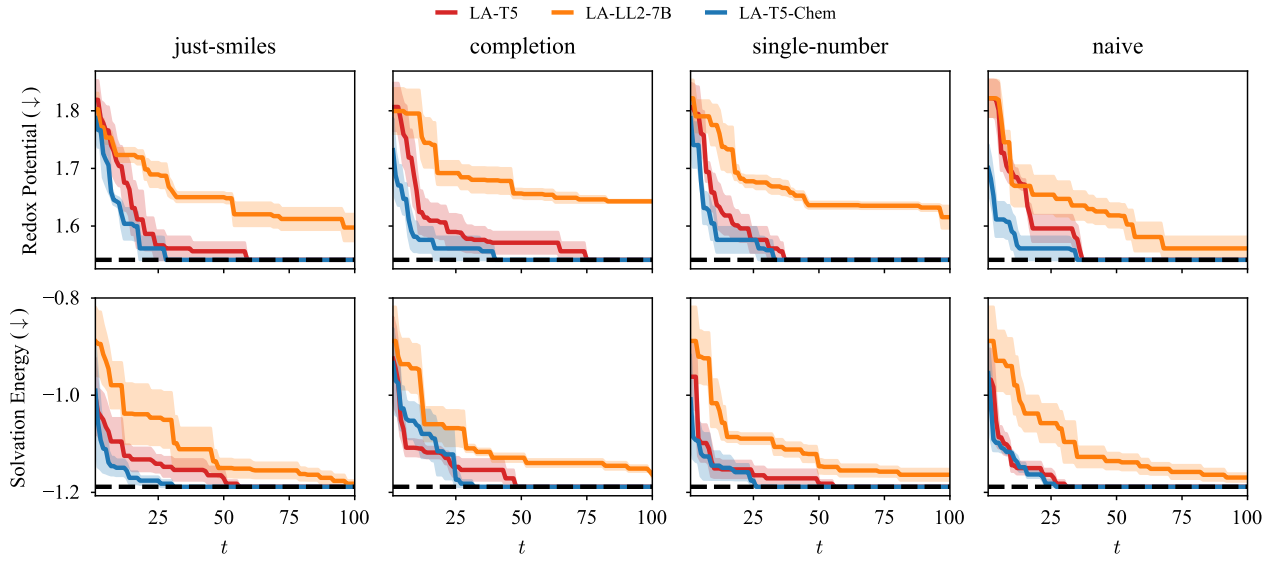


Figure 16. IUPAC with different prompts.

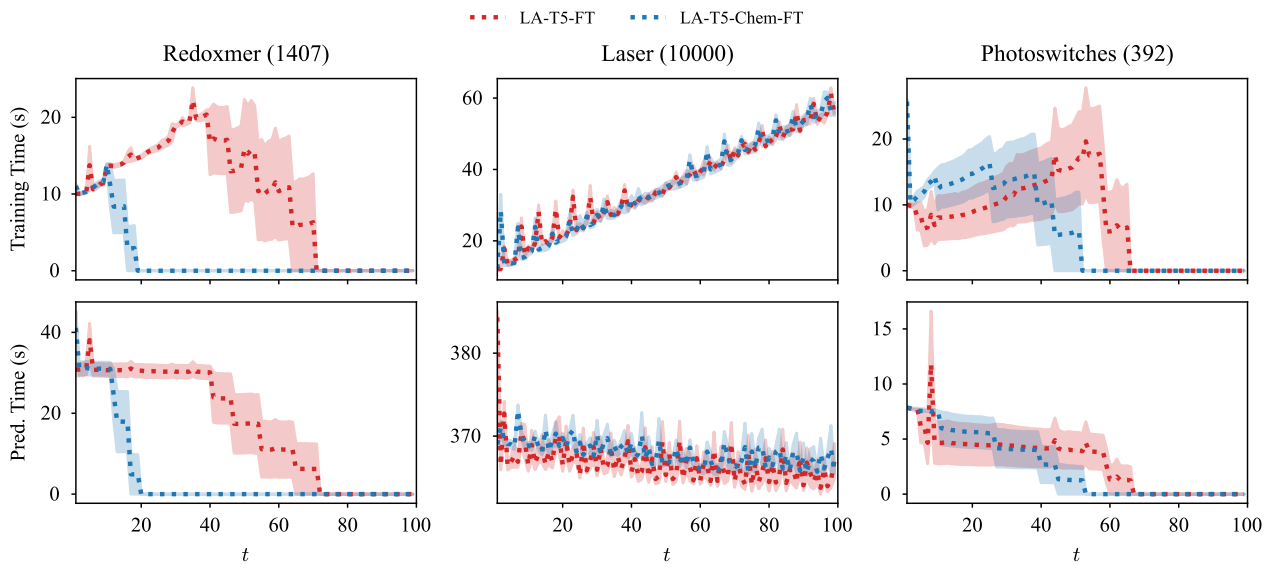


Figure 17. Wall clock time (in seconds) per BO iteration of the finetuned surrogates. Numbers in parentheses are numbers of test points $|\mathcal{D}_{\text{cand}}|$. Notice that the costs are roughly linear in the number of test points, indicating that forward passes over $\mathcal{D}_{\text{cand}}$ take the bulk of the computation. T5-Chem has a lower wall-clock time on average since it tends to terminate faster than T5. E.g., some of the BO runs across random seeds have been done and thus have zero wall-clock time.

B.4. Textual representations

As for which textual representation of the molecule to use, we compare SMILES with IUPAC representations in Figure 15. We found that Llama-2-7b is insensitive to this. Meanwhile, observed significant effects on both T5 and T5-Chem. Thus, we can conclude that it might be worth trying different molecular string representations (IUPAC, SMILES, SELFIES, etc.) when using T5-Chem. Moreover, in Figure 16, we show the impact of prompting when IUPAC representations are used. Our findings are similar to the ones with SMILES. Especially Llama-2-7b is largely insensitive to prompting.

B.5. Computational costs

We present the computational cost of performing finetuning in Figure 17 on datasets of various magnitudes (392, 1407, 10,000). We use a single NVIDIA A40 and NVIDIA RTX6000 GPUs for the laser and the rest of the finetuning experiments, respectively.[7] Since the number of training points is constant across those datasets (equals to t plus 10, the latter is the size of the initial training set $\mathcal{D}_1$), the time needed for each BO iteration is almost exclusively influenced by the prediction phase of the BO iteration (Algorithm 1, line 3). Indeed, we see an almost exact scaling in terms of the size of $\mathcal{D}_{\text{cand}}$. This is encouraging since finetuning/training is *not* the bottleneck, contrary to popular belief.

[7]For fixed-feature computation, we run the BO loop on consumer-level laptop CPU.