Informing Acquisition Functions via Foundation Models for Molecular Discovery

Qi Chen[3,4]*, **Fabio Ramos**[8,9], **Alán Aspuru-Guzik**[1,6,7,10], **Florian Shkurti**[1,2,5,6,7]*
[1] University of Toronto (UofT), [2] UofT Data Science Institute,
[3] ELLIS Institute Finland, [4] Aalto University
[5] UofT Robotics Institute, [6] Acceleration Consortium, [7] Vector Institute
[8] NVIDIA Research Seattle Robotics Lab (SRL), [9] University of Sydney
[10] Senior Fellow, Canadian Institute for Advanced Research (CIFAR)

Abstract

Bayesian optimization (BO) is widely used to accelerate molecular discovery by reducing costly oracle evaluations. Foundation models provide a promising source of prior knowledge for BO, but simply using their representations in standard surrogate-based pipelines is often unreliable in low-data regimes with high-dimensional features and vast discrete candidate spaces. Rather than asking whether LLMs are universally useful for molecular BO, we ask how Acquisition Functions (AFs) should be designed to exploit weak, high-dimensional, and partially informative foundation-model priors. We propose *LLM-guided Acquisition Tree* (LLMAT), a surrogate-free BO framework that reformulates likelihood-free acquisition estimation as recursive local acquisition learning. LLMAT trains binary classifiers on LLM representations, where each classifier jointly induces a promising/non-promising partition and defines a local AF within the corresponding region. This produces a hierarchy of localized AFs and enables efficient candidate selection via Monte Carlo Tree Search. To stabilize learning with few observations, LLMAT meta-learns shared classifier parameters and initializations across tree nodes. An optional LLM-guided clustering module further reduces AF evaluation cost by restricting search to statistically promising coarse property clusters. Extensive experiments and ablations demonstrate consistent improvements in scalability, robustness, and sample efficiency for LLM-guided BO in molecular discovery.

1 Introduction

Discovering molecules with desirable properties is crucial for drug design, materials science, and chemical engineering. Since exhaustive evaluation over the vast chemical space [Restrepo, 2022] is infeasible, Bayesian optimization (BO, Frazier [2018], Garnett [2023]) has become a standard tool for reducing the number of costly oracle evaluations, such as DFT simulations [Parr, 1989] and laborious experimental measurements. BO typically relies on a probabilistic surrogate model (*e.g.*, Gaussian Processes (GPs) and Bayesian Neural Networks (BNNs)) to define an Acquisition Function (AF), which selects the next molecule by balancing exploration and exploitation.

Foundation models offer a promising source of prior knowledge for molecular BO, as they encode information from large-scale text and molecular corpora. However, recent studies [Kristiadi et al., 2024] suggest that simply plugging LLM representations into standard BO pipelines does not reliably lead to better molecular optimization: fixed LLM features or Parameter-Efficient Fine-Tuning (PEFT)-enhanced [Hu et al., 2021] neural surrogates can still be hampered by high-dimensional

*The initial work was conducted at the University of Toronto, with subsequent improvements at Aalto University. Correspondence to: qi.chen.inf@aalto.fi, florian@cs.toronto.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

representations, low-data overfitting [Xie et al., 2025], and the cost of repeatedly fitting Bayesian surrogates over large discrete candidate pools. This raises a more fundamental question: not whether foundation-model features are informative, but how AFs should be designed to exploit weak, high-dimensional, and partially informative priors in the low-data regime.

Our central hypothesis is that the bottleneck lies not only in the representation itself, but also in the global nature of conventional AFs. When only a handful of labeled molecules are available, fitting a single global surrogate or AF over high-dimensional foundation-model embeddings can obscure local structure and lead to unstable candidate selection.

To address this, we propose `LLM-guided Acquisition Tree` (LLMAT), illustrated in Fig. 1: a surrogate-free BO framework that turns likelihood-free AF estimation into a recursive local search procedure. Instead of fitting a global Bayesian surrogate, LLMAT learns binary classifiers on top of foundation-model representations. The key novelty is not the use of tree search or likelihood-free BO separately, but the identification of the likelihood-free acquisition classifier as a single object that simultaneously estimates a local acquisition function and induces the next search-space partition. Each classifier therefore serves two coupled roles: partitioning the candidate space into more and less promising regions, and defining a local likelihood-free AF within each induced region.

This formulation is especially beneficial in the low-data regime, where each tree node may contain only a small number of observations. We therefore meta-learn shared classifier and PEFT parameters across nodes, yielding a stable initialization for local AF estimation. We further incorporate an optional LLM-guided clustering module that uses foundation models not as property predictors, but to identify coarse, property-relevant regions of the molecular search space, focusing AF evaluation on statistically promising clusters. The consistency and utility of these clusters across different LLM APIs are exemplified and statistically validated in Sec. A.

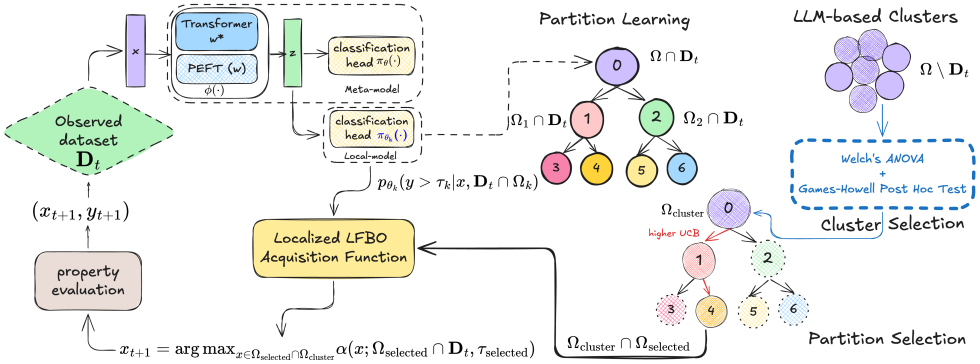


Figure 1: Overview of LLMAT: Binary classifiers on the LLM features recursively partition the candidate space to a tree structure and local AFs are learned from observations in each partition $\Omega_k \cap \mathbf{D}_t$. Promising partition Ω_{selected} is selected via UCT. Unseen candidates are clustered via LLM prompts, and promising clusters Ω_{cluster} are selected via statistical tests. Within $\Omega_{\text{cluster}} \cap \Omega_{\text{selected}}$, the property of the candidate with the highest local AF value is evaluated and added to the dataset.

To summarize, the main contributions of our paper are:

- We introduce a foundation-model-informed AF approach for molecular BO. Unlike standard BO methods that estimate a single global AF, LLMAT recursively learns local AFs over classifier-induced partitions of the molecular search space.
- We show that likelihood-free acquisition classifiers can serve as a unified algorithmic primitive for both density-ratio acquisition estimation and adaptive search-space partitioning. This is the central algorithmic distinction of LLMAT: the same classifier both scores candidates through a local AF and defines the next partition, rather than combining a separately learned partitioning policy with a separately defined acquisition function. The resulting formulation naturally supports fixed foundation-model features and PEFT-based adaptation.
- We stabilize local AF learning in the low-data regime through meta-learned shared classifier and PEFT parameters, enabling reliable use of high-dimensional foundation-model representations.

- We introduce an optional LLM-guided clustering module for scalable AF evaluation and validate LLMAT on six molecular discovery benchmarks, where it improves sample efficiency, robustness, and computational cost over previous LLM-based BO baselines.

2 Related Work

Due to space limitations, a more detailed discussion of related works is provided in Appendix C.

LLMs for Molecule Discovery. Recent studies have explored the potential of LLMs in a variety of chemistry-related tasks, including molecular property prediction [Lu and Zhang, 2022, Christofidellis et al., 2023, Jablonka et al., 2023], property-related molecule optimization [Ramos et al., 2023, Kristiadi et al., 2024], and molecular generation with desired properties [Liu et al., 2024a, Flam-Shepherd and Aspuru-Guzik, 2023, Jablonka et al., 2024]. However, many of these LLM-based approaches [Ramos et al., 2023, Guo et al., 2023] rely heavily on in-context learning (ICL) and prompt engineering. This dependency poses challenges for tasks involving strict numerical objectives, where LLMs often fall short in satisfying precise quantitative constraints or optimizing for specific target values [AI4Science and Quantum, 2023]. To address this limitation, recent efforts have augmented LLMs with optimization frameworks such as BO or Evolutionary Algorithms (EA). For example, Kristiadi et al. [2024] integrate Laplace approximations with BNN for property prediction, while Wang et al. [2024] combine EA to generate and search molecules with desired property.

LLMs for Bayesian Optimization over Molecules. LLMs and chemistry foundation models encode useful priors from text and molecular corpora, but exploiting them for BO remains challenging in low-data regimes. Prior work has used LLMs either through adaptive in-context prompting [Ramos et al., 2023], which is costly and context-limited, or as fixed/PEFT feature extractors with Bayesian neural surrogates [Kristiadi et al., 2024], which require expensive Laplace approximations over large candidate sets. Moreover, a single global surrogate can be unreliable with few observations and high-dimensional LLM embeddings [Wang et al., 2020]. In contrast, LLMAT uses foundation-model priors to directly learn localized acquisition functions without surrogate fitting, and optionally restricts AF evaluation to promising LLM-induced clusters for scalability.

Difference from tree-partitioned black-box optimization. The closest algorithmic line is tree-partitioned black-box optimization [Wang et al., 2020, 2021, Li et al., 2025], which adaptively partitions the search space and uses tree search or partition-level weighting to guide candidate selection. However, these methods typically decouple partition learning from acquisition estimation: partitions are induced by K-means clustering and auxiliary classifiers, while candidates are scored by separate surrogate models. In contrast, LLMAT uses the same likelihood-free acquisition classifier to both estimate a local density-ratio AF and induce the next partition. Thus, LLMAT differs from prior tree-partitioned BO not by merely adding MCTS to LFBO, but by making the LFBO classifier itself the mechanism for both local acquisition estimation and recursive partition learning. This unifies search-space refinement and acquisition learning, turning tree search into a mechanism for recursive local AF learning, which is particularly suitable for LLM-based BO with PEFT-based adaptation.

3 Background

3.1 Bayesian Optimization

Bayesian Optimization (BO) aims to effectively maximize some unknown function $f : \mathcal{X} \to \mathcal{Y}$ over the candidate space $\mathcal{X}$, *i.e.*, find $x^* = \arg\max_{x \in \mathcal{X}} f(x)$ [2], given a dataset $\mathbf{D}_t = \{x_i, y_i\}_{i=1}^t$ of t observations, where x_i is a molecule, $y_i = f(x_i) + \epsilon, \forall i \in [t]$ is a property value, and ϵ is noise conventionally assumed to be a Gaussian with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Due to the intractability of f, a probabilistic surrogate model $\hat{f}$ learned from $\mathbf{D}_t$ is often used to approximate f with consideration of epistemic uncertainty, reflected in the variance of the posterior $p(\hat{f}|\mathbf{D}_t)$. Hence, for any unobserved data (x, y), we have the predictive posterior $p(y|x, \mathbf{D}_t) = \int p(y|\hat{f}(x))p(\hat{f}(x)|\mathbf{D}_t)d\hat{f}(x)$.

Acquisition Function and Expected Utility. To address problems in the sequential decision-making setting, BO methods typically select the next query x_{t+1} by maximizing an acquisition function

[2] It can be extended to minimization without loss of generality.

$\alpha : \mathcal{X} \to \mathbb{R}$, defined as:

$$\alpha(x; \mathbf{D}_t, \tau) := \mathbb{E}_{y \sim p(y|x, \mathbf{D}_t)}[u(y; \tau)], \tag{1}$$

where $x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha(x; \mathbf{D}_t, \tau)$, $u(y; \tau)$ is a chosen utility function and τ is a threshold on y. The selection of τ controls the exploration-exploitation trade-off, and sometimes is set as $\tau = \max_t y_t$, the best observed value so far (the density-ratio form below instead sets τ as a quantile of the observed y). Common choices of the utility function $u(y; \tau)$ satisfying Eq.(1) include: $u^{EI}(y; \tau) = \max(y - \tau, 0)$ for Expected Improvement (EI, Mockus et al. [1978]) that measures how much y exceeds the threshold τ; $u^{PI}(y; \tau) = \mathbf{1}(y - \tau > 0)$, indicating whether y exceeds τ for Probability of Improvement (PI, Kushner [1964]); Many other AFs can be expressed in this form with a more complex utility function, such as Upper Confidence Bound (UCB, Srinivas et al. [2009]), Entropy Search (ES, Hennig and Schuler [2012]), Knowledge Gradient (KG, Frazier et al. [2008]), Thompson Sampling (TS, Thompson [1933]).

Direct Estimation of Acquisition Functions. Typical BO instantiations can be characterized as *indirect*: they first approximate the predictive posterior $p(y \mid x, \mathbf{D}_t)$ via a surrogate model based on Gaussian Processes (GPs) or Bayesian Neural Networks (BNNs), and then compute the acquisition functions via Eq. (1) for a given utility function. We provide an introduction to this in Appendix B.1. However, these methods often face scalability issues due to the high computational cost when combining large, deep models, which limits their flexibility. Our proposed method on the other hand directly models AFs, forgoing the need for surrogate model. Let $l(x) = p(x|y \leq \tau, \mathbf{D}_t)$ and $g(x) = p(x|y > \tau, \mathbf{D}_t)$ be two densities that respectively characterize the data distribution on non-promising and promising region of candidate space. Instead of explicitly modeling the predictive posterior $p(y|x, \mathbf{D}_t)$, Bergstra et al. [2011] and Tiao et al. [2021] directly model the AF as γ-relative density ratio:

$$r_\gamma(x) := \frac{g(x)}{\gamma g(x) + (1 - \gamma) l(x)} = h_\gamma(r_0(x)), \tag{2}$$

where $h_\gamma : u \to (\gamma + u^{-1}(1 - \gamma))^{-1}, u > 0$ is strictly non-decreasing, $r_0(x) = \frac{g(x)}{l(x)}$ is the ordinary density ratio, and $\gamma = p(y > \tau|\mathbf{D}_t)$ indicates τ is set as the $(1 - \gamma)$-quantile of all observed y, i.e., $\tau = F^{-1}(1 - \gamma)$ with F the CDF of y, so that a fraction γ of observations exceed τ. This density ratio has been proven by Song et al. [2022] to be equivalent to PI. Tree Parzen Estimator (TPE, [Bergstra et al., 2011]) estimates the density ratio $r_0(x)$ by separately estimating $l(x)$ and $g(x)$ with tree variant of Kernel Density Estimation (KDE). BORE [Tiao et al., 2021] avoids such indirect estimation by formulating the problem as a binary classification and showing the classifier $\pi_\theta(x) \approx p(c = 1|x, \mathbf{D}_t) = \gamma r_\gamma(x)$, where $c = \mathbf{1}(y > \tau)$ is the class label and $p(c = 1|\mathbf{D}_t) = p(y > \tau|\mathbf{D}_t) = \gamma$. Likelihood-free Bayes Optimization (LFBO, Song et al. [2022]) adopts variational $\phi-$divergence estimation to directly estimate expected utility AFs for complex $p(y|x, \mathbf{D}_t)$ and $u(y; \tau)$, not limited to PI. Then, the likelihood-free AF is defined as:

$$\alpha(x; \mathbf{D}_t, \tau) := \pi_{\theta^*}(x)/(1 - \pi_{\theta^*}(x)), \tag{3}$$

where $\pi_\theta(x) \in (0, 1)$ is the output of a generic binary classifier with parameters θ (it need not equal $p(c = 1|x, \mathbf{D}_t)$, which is only the minimizer of BORE's unweighted cross-entropy), and $\theta^* = \arg\min_\theta \mathcal{L}^\tau_{\mathbf{D}_t}(\theta)$ with the utility-weighted loss

$$\mathcal{L}^\tau_{\mathbf{D}_t}(\theta) := \mathbb{E}_{(x,y) \sim \mathbf{D}_t}[-u(y; \tau) \log \pi_\theta(x) - \log(1 - \pi_\theta(x))].$$

3.2 BO for Molecular Discovery

In molecular discovery, the goal is to identify a novel molecule $x \in \mathcal{X}$ with desirable properties $y \in \mathcal{Y} \subseteq \mathbb{R}$, often requiring exploration of a **vast** and **discrete** search space $\mathcal{X}$ (estimated to contain over 10^{100} unique molecules). Exhaustive property evaluation is infeasible and costly, e.g., when relying on Density Functional Theory (DFT). In practice, we often restrict the search to a smaller candidate subset $\Omega \subset \mathcal{X}$. At each round $t + 1$, given prior observations $\mathbf{D}_t = \{x_i, y_i\}_{i=1}^t \cup \mathbf{D}_0$ with n initial points $\mathbf{D}_0 = \{\tilde{x}_j, \tilde{y}_j\}_{j=1}^n$, a new candidate x_{t+1} is selected from $\Omega \setminus \mathbf{D}_t := \{x_i \in \Omega \mid x_i \notin \text{supp}_X(\mathbf{D}_t)\}$ for evaluation, aiming to find the optimal molecule x^* in as few rounds as possible. This naturally fits the sequential BO framework.

To input molecules into machine learning models, common representations [Griffiths et al., 2023, Janakarajan et al., 2024] include: (1) text-based formats such as SMILES [Weininger, 1988], SELFIES [Krenn et al., 2020], and IUPAC names; (2) feature-based fingerprints such as Morgan [Morgan,

1965] and ECFPs [Rogers and Hahn, 2010]; and (3) graph-based encodings [Duvenaud et al., 2015]. In this work, we focus on text-based representations, which can be directly generated by modern NLP models, including general-purpose LLMs, transformer-based chemical foundation models, and domain-specific LLMs specialized for chemical data.

4 Methodology

Early BO rounds with few observations, high-dimensional molecular features, and large discrete candidate spaces make global surrogate fitting unreliable and full-space AF evaluation costly. LLMAT addresses these challenges by coupling *where to search* with *how to estimate acquisition values*. **(1)** It learns local AFs over classifier-induced partitions, using a meta-learned initialization to stabilize local classifiers in the low-data regime. **(2)** It uses LLM-based clustering and statistical testing to prune the candidate pool to property-relevant regions before AF estimation. Unlike conventional BO, which optimizes a global AF over the full candidate set, and unlike prior MCTS-based BO methods where the tree mainly serves as a search policy over externally defined partitions and surrogate scores, LLMAT makes the tree itself an acquisition-learning structure. Each likelihood-free classifier both estimates a local density-ratio AF and induces the next promising/non-promising partition, thereby coupling candidate-space refinement and acquisition learning in a single objective. This yields a hierarchy of localized AFs traversed by tree search, as illustrated in Fig. 1. A non-rigorous theoretical analysis is given in Appendix I (without a full convergence guarantee): it interprets the node-wise classifiers as density-ratio estimators, shows why recursive local classifiers can outperform a global one, decomposes the error under PI into approximation, estimation and routing terms, and decomposes the cumulative regret.

4.1 Meta-Learning Candidate Partitions and Local AFs

We next describe the training of shared classifiers during tree construction, which simultaneously induce search-space partitions and define local AFs within each partition.

Recursive Candidate Space Partitioning. At any iteration t, we have the currently observed dataset $\mathbf{D}_t$ defined in Sec. 3.1. As shown in Fig. 1, we hierarchically partition the candidate set Ω by combining Monte Carlo Tree Search (MCTS), where each node in the tree corresponds to a subset of Ω. The tree construction begins at the root node (node 0) with $\Omega_0 = \Omega$ being the entire space.

Each node Ω_k is recursively bifurcated into two subsets using a binary classifier π_{θ_k}. Let $\Omega_{2k+1} := \{x \in \Omega_k \mid \pi_{\theta_k}(x) > 0.5\}$ and $\Omega_{2k+2} := \{x \in \Omega_k \mid \pi_{\theta_k}(x) \leq 0.5\}$ be the left and right children respectively, representing the more promising (i.e., more likely to contain high-property candidates) and less promising regions. The tree expansion continues until a predefined maximum depth L or the minimum number of samples in each node is reached, yielding a total maximum of $2^{L+1} - 1$ nodes. To train the binary classifier at each node Ω_k, we use the subset of observed samples within that node, i.e., $\mathbf{D}_t \cap \Omega_k := \{(x_i, y_i) \in \mathbf{D}_t \mid x_i \in \Omega_k\}$, where their class labels can be obtained by directly thresholding y as we described in the following. For unobserved samples in Ω_k, the learned classifier π_{θ_k} assigns them to the left or right child. When estimating the AFs, rather than classifying all samples across the whole tree, we only route the search from the root to the most promising leaf, using classifiers at each node along that path.

Algorithm 1 Meta-learning Tree Partitions and local AFs

Input: Observed dataset $\mathbf{D}_t$, quantile γ, candidate set Ω, tree depth L, classification head θ, PEFT parameters w, learning rate η, SGD steps K, finetuning flag $\mathbb{F}$, minimal leaf sample size m.
Nodes $\mathcal{N} = \{\}$;
$\Omega_0 = \Omega, \mathbf{D}_t \cap \Omega_0 = \mathbf{D}_t, \mathbf{D}_t \cap \Omega_k = \emptyset, \forall k > 0$;
for $k = 0$ **to** $2^{L+1} - 2$ **do**
  if $|\Omega_k \cap \mathbf{D}_t| < m$ or Ω_k not splitable **then**
    continue;
  end if
  $\tau_k = \Phi_k^{-1}(\gamma)$; $\tilde{\theta} = \theta_{\lceil k/2 \rceil - 1}$ if $k > 0$ else $\tilde{\theta} = \theta$;
  Fix w, update $\theta_k = \text{SGD}(\mathcal{L}^{\tau_k}_{\mathbf{D}_t \cap \Omega_k}(\tilde{\theta}, w), K)$;
  $\theta \leftarrow \theta + \eta(\theta_k - \theta)$, $\mathcal{N} \leftarrow \mathcal{N} \cup \{k\}$;
  for (x, y) in $\mathbf{D}_t \cap \Omega_k$ **do**
    if $\pi_{\theta_k}(x) > 0.5$ **then**
      $\mathbf{D}_t \cap \Omega_{2k+1} \leftarrow (\mathbf{D}_t \cap \Omega_{2k+1}) \cup \{(x, y)\}$;
    else
      $\mathbf{D}_t \cap \Omega_{2k+2} \leftarrow (\mathbf{D}_t \cap \Omega_{2k+2}) \cup \{(x, y)\}$;
    end if
  end for
end for
if $\mathbb{F}$ **then**
  Fix θ, update $w = \text{PEFT}(\frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} \mathcal{L}^{\tau_k}_{\mathbf{D}_t \cap \Omega_k}(\theta, w))$;
end if
Output: $\pi_{\theta_k}(x), \alpha(x; \Omega_k \cap \mathbf{D}_t, \tau_k), \forall k \in \mathcal{N}$;

LFBO Classifier Shared for Bifurcating. In LFBO, the AF at each iteration t is estimated directly by learning a (weighted) binary classifier $\pi_\theta(x)$ whose odds estimate the AF: pointwise, the minimizer

of the weighted loss satisfies $\pi^*(x)/(1-\pi^*(x)) = \mathbb{E}[u(y;\tau)|x] = \alpha(x)$, instead of training surrogate models for typical BO algorithms. This classifier can be reused as a natural choice for the aforementioned candidate space partitioning during the construction of the tree.

At each node k, we set $\tau_k = \Phi_k^{-1}(\gamma)$ under the same γ, where $\Phi_k(\tau) := p(y > \tau|\mathbf{D}_t \cap \Omega_k) = 1 - F_k(\tau)$ is the survival function of y in node k and F_k its CDF. Hence $\tau_k = F_k^{-1}(1-\gamma)$ is the $(1-\gamma)$-quantile of the observed y in $\mathbf{D}_t \cap \Omega_k$, i.e., a fraction γ of the samples in the node lie above τ_k. Then, the corresponding binary classifier is $\pi_{\theta_k}(x) = p_{\theta_k}(c = 1|x, \mathbf{D}_t \cap \Omega_k)$ with $c = \mathbf{1}(y > \tau_k)$, where the binary labels c for training the classifier are assigned by thresholding the y for all training samples in node k instead of using the binary clusters in Wang et al. [2020].

Meta-learning Shared Binary Classifiers. Using a shared classification head connected after the transformer feature extractor, we propose a meta-learning approach that works for both fixed features and parameter efficient fine-tuning (PEFT). Since the tree is built recursively, we use the sequential version of Reptile [Nichol et al., 2018], meta-training the classification head. For a maximal L-depth Tree, we store one meta-model θ and $|\mathcal{N}|$ local models $\theta_k, \forall k \in \mathcal{N}$.The detailed algorithm for learning the partitions and AFs at each iteration is presented in Algo. 1.

Algorithm 2 Partition Selection for Refined and Reduced AFs

Input: Dataset $\mathbf{D}_t$, tree depth L, candidates in selected clusters Ω_{cluster}, classifier θ, learning rate η, SGD steps K, quantile γ, UCT hyper-parameter λ, time budget T, all nodes $\mathcal{N}$.
for $t = 0$ **to** T **do**
  Run Algo. 1;
  #Select the search tree path.
  Set $k = 0$, $\tau_k = \Phi_k^{-1}(\gamma)$;
  Path $\mathcal{P} \leftarrow []$;
  while $k \in \mathcal{N}$ **do**
    $\mathcal{P} \leftarrow \mathcal{P} + [k]$;
    Compute partition score for $2k+1$ and $2k+2$;
    $k = \arg\max_{j \in \{2k+1, 2k+2\}} \mathcal{S}_j$;
  end while
  #Backtrack and select local AF up the path.
  $\mathcal{P} \leftarrow \text{Reverse}(\mathcal{P})$;
  selected = 0;
  for each k in $\mathcal{P}$ **do**
    if $|\Omega_k \cap \Omega_{\text{cluster}}| = 0$ **then**
      continue;
    else
      selected = k;
      break;
    end if
  end for
  Sample $x_{t+1} = \underset{x \in \Omega_{\text{selected}} \cap \Omega_{\text{cluster}}}{\arg\max} \alpha(x; \Omega_{\text{selected}} \cap \mathbf{D}_t, \tau_{\text{selected}})$;
  Evaluate $y_{t+1} = f(x_{t+1})$;
  Update $\mathbf{D}_{t+1} \leftarrow \mathbf{D}_t \cup \{(x_{t+1}, y_{t+1})\}$;
end for

4.2 Candidate Selection on Promising Subsets

With the built tree partition and learned local AFs, we can perform refined AF optimization via both partition and cluster selection to identify promising candidates efficiently.

Partition Selection. Given the aforementioned partition rule, a greedy strategy that always selects the left node would exploit the most promising leaf but risks overfitting to suboptimal partitions without sufficient exploration. To balance exploration and exploitation, we select partitions using some score metrics. For example, as in Wang et al. [2020], let $n_k = |\mathbf{D}_t \cap \Omega_k|$ denote the visit count for each node k, and let the node value $v_k = \frac{1}{n_k}\sum_{i=1}^{n_k} y_i$ be the average property value of observed samples at node k. Then, the Upper Confidence Bound for Trees (UCT) score for a node $k \in \mathcal{N}$ is defined as: $\mathcal{S}_k^{\text{UCT}} := v_k + 2\lambda\sqrt{2\log(n_{\lceil \frac{k}{2} \rceil - 1})/n_k}$, where $n_{\lceil \frac{k}{2} \rceil - 1}$ is the number of visits at the parent node of k-th node and λ is a hyper-parameter that controls the exploration-exploitation trade-off. Another possible choice is $\mathcal{S}_k^{\text{Var}} := v_k + 2\lambda\sqrt{\text{var}_k}$, where $\text{var}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}(y_i - v_k)^2$. When $\lambda = 0$, both recover the greedy policy if the classification is accurate enough (so that the left child has the larger v_k). Finally, we select the child node with the highest score from the root to a leaf. If no candidates reach a leaf node, we use a backtracking approach to select the parent node instead. The selected partition is denoted as $\Omega_{selected}$, as shown in Algo. 2.

Clustering for Efficient Estimation of AFs. To reduce the high GPU memory and computation cost of calculating AFs for all candidates at each BO round, we propose a clustering-based approach. Molecular feature representations are precomputed once and grouped into clusters. Candidates within the selected clusters then undergo the partition selection process, and the optimal candidate is determined by comparing AFs only for $\Omega_{\text{cluster}} \cap \Omega_{selected}$.

(1) Cluster Selection via Statistical Test. We propose a statistical filtering approach on $\mathbf{D}_t$ to identify promising clusters for reduced AF estimation. Welch's ANOVA (Appendix B.3) serves

as an omnibus test of equal mean property across clusters, and the Games-Howell post-hoc test (Appendix B.4) compares all pairs of clusters: for each pair that differs significantly at p-value p, the cluster with the lower mean is excluded. To support this, the BO initialization $\mathbf{D}_0$ is constructed by uniformly sampling from each cluster.

(2) LLMs for Property-Related Clustering. Unlike unsupervised methods (e.g., k-means), general LLMs (e.g., ChatGPT) classify molecules into extreme high/high/medium/low/extreme low property groups via tailored prompts (Appendix D), leveraging chemical knowledge from their training data. This yields property-aligned clusters that enhance BO at a cost orders of magnitude lower than ICL-based approaches [Ramos et al., 2023], offering a cost-effective way to exploit knowledge from general LLMs.

5 Experiments

In this section, we experimentally validate the proposed algorithm across multiple datasets and ablation scenarios, demonstrating its ability to accelerate molecular discovery through superior BO performance and increased computational efficiency. All results are reported as the mean over 15 independent runs with different random seeds, with shaded bands showing $\pm$ one standard error of the mean (SE). Each run starts from 10 initial observations and uses a budget of 100 BO rounds. Additional details are provided in Appendix E and F. Our code is publicly available at `https://github.com/livreQ/LLMAT`.

5.1 Setting

Datasets. We employ the well-known multi-modal Levy function [Levy et al., 1982] to generate a synthetic set to demonstrate the effectiveness of the proposed method in achieving more refined AF estimation within localized sub-regions. For molecular data, following Kristiadi et al. [2024], we initialize BO with $n = 10$ points and evaluate our models on six benchmark datasets that capture realistic molecular design challenges across diverse domains: (1) minimizing redox potential for redoxmers (1,407 molecules), (2) minimizing solvation energy for flow battery electrolytes (1,407 molecules) [Agarwal et al., 2021], (3) minimizing docking scores of kinase inhibitors (10,449 molecules) for drug discovery [Graff et al., 2021], (4) maximizing fluorescence oscillator strength for laser materials (10,000 molecules) [Strieth-Kalthoff et al., 2024], (5) maximizing power conversion efficiency (PCE) in photovoltaic materials (10,000 molecules) [Lopez et al., 2016], and (6) maximizing π–π^* transition wavelengths for organic photoswitches (392 molecules) [Griffiths et al., 2022]. Together, these tasks cover a broad spectrum of molecular properties, providing a comprehensive testbed for molecular discovery. For each dataset, the physics-based simulators released by the original authors are used as the ground-truth oracles.

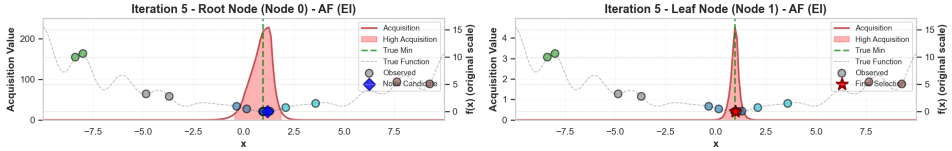


Figure 2: Leaf-node AFs offer more fine-grained candidate suggestions for Levy-1D optimization.

Fixed Features and Foundation Models. To assess whether the findings of Kristiadi et al. [2024] that LLMs benefit molecular BO only when trained on domain specific data hold for our algorithm, we benchmark LLMAT against several baselines on using different features: (i) Morgan fingerprints [Morgan, 1965] as a chemistry specific algorithmic representation, (ii) chemistry foundation models including T5-Chem [Christofidellis et al., 2023] and MolFormer [Ross et al., 2022], and (iii) general LLMs such as T5 [Raffel et al., 2023], GPT-2 (medium and large), and Llama-2-7b [Touvron et al., 2023]. Our comparison includes (1) Bayesian optimization using fixed feature representations extracted from these models, and (2) Parameter-Efficient Fine-Tuning (PEFT) of the above chemistry foundation models.

Evaluation Metrics. We report the trajectory of the best observed property value over time and use the **GAP** metric defined in Jiang et al. [2020]: $\text{GAP}_t := \frac{y_t^* - y_0}{y^* - y_0}$, which normalizes the progression

of the optimal value, where y^* is the global optimum, y_0 is the initial optimum, and y_t^* is the best value observed up to iteration t. However, in molecular discovery, identifying a single optimal molecule is often not the only goal. In practice, it is equally important to discover a *set of high-performing candidates* that scientists can further evaluate, balance multiple criteria, or analyze for structure-property relationships. For this objective, we additionally report the **average regret**: $\text{Regret}_t := \frac{1}{t}\sum_{i=1}^{t} |y^* - y_i|$, which reflects the overall quality of the molecules discovered so far. When aggregating results across datasets, we normalize the average regrets for comparability.

5.2 Performance Analysis

Effectiveness of Localized Search on Synthetic Data. Beyond LLM adaptation and its application to molecular discovery, our method contributes a novel discrete BO algorithm: we introduce shared binary classifiers that jointly serve both tree partitioning and localized AF approximation. To illustrate this benefit, we sample 1,000 examples from the Levy-1D function and observe that the proposed algorithm yields a more refined AF estimate at the selected leaf node than at the root node used by vanilla LFBO [Song et al., 2022]. As shown in Fig. 2, the leaf node exhibits a narrower confidence region and more accurate estimation around the true optimum, and the refinement becomes more pronounced over BO iterations as additional observations accumulate, see Sec. F.6 in the Appendix.

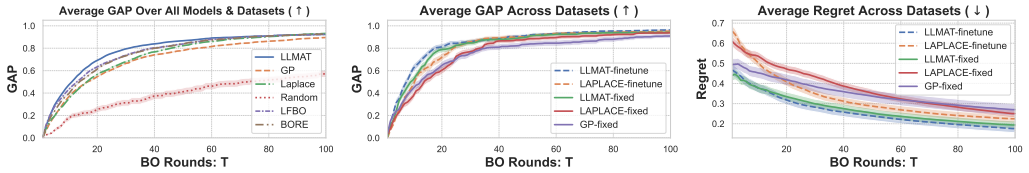


Figure 3: **Left:** Comparison over baselines on fixed features across all models and datasets. **Middle and right:** Average GAP and regret, defined in Sec. 5.1, across datasets for fine-tuned (PEFT on w) and fixed (frozen pretrained weights w^*) T5-Chem models.

Superior Performance of LLMAT on Molecular Data. In the left panel of Fig. 3, we show that the proposed LLMAT algorithm outperforms standard discrete BO baselines when applied to fixed features extracted from the aforementioned models: Gaussian Processes (GP), the Laplace approximation (LAPLACE) [Kristiadi et al., 2024], Likelihood-Free BO (LFBO), BORE [Tiao et al., 2021], and random search (RANDOM), as measured by the average GAP metric across all datasets and models. The middle and right panels of Fig. 3 further compare LLMAT against LAPLACE across all datasets to assess iterative PEFT effectiveness with T5-Chem backbones. MolFormer PEFT results are in Fig. 25. While iterative PEFT enhances BO performance for both approaches, LAPLACE with PEFT still falls short of the performance achieved by our algorithm without PEFT. The radar chart on the left of Fig. 5 summarizes the benefit of fine-tuning relative to a fixed representation, in magnitude and in speed. With $\Delta\text{GAP}_t = \text{GAP}_t^{\text{FT}} - \text{GAP}_t^{\text{Fixed}}$ and $\Delta\text{Regret}_t = \text{Regret}_t^{\text{Fixed}} - \text{Regret}_t^{\text{FT}}$ averaged over runs (positive values favor fine-tuning), it reports six metrics: the peak (maximum over rounds) and final (mean over the last 10 rounds) improvements of GAP and regret, and for each of them a peak-round score that is larger when the maximum improvement is reached earlier (Appendix E). It shows (1) faster convergence in early BO rounds, (2) superior final performance, and (3) greater stability across different foundation models. The PEFT improvements are more substantial for Molformer than T5-Chem, reflecting T5-Chem's larger scale and richer domain knowledge.

Better Exploitation of Priors from Fixed Features. In Fig. 4, we compare LLMAT with maximal tree depth $L = 1$ of 3 nodes against GP, LAPLACE, and RANDOM, using various fixed features mentioned before. We only plot the feature that achieved the best performance for each algorithm; the full plot is deferred to Fig. 10. Fig. 4 shows that LLMAT achieves superior performance compared to all baselines on most datasets, with slightly lower performance than LAPLACE on the Laser dataset. Other baselines exhibit substantial performance drops on several datasets while LLMAT remains consistent. LLMAT, LAPLACE, and GP achieve their best performance on the same feature models for the Redox-mer, Kinase, and Laser datasets (T5-Chem, T5, and MolFormer, respectively). However, the best feature models differ for other datasets. The strong performance of LLMAT with general-purpose LLMs such as T5, GPT2, and Llama-2-7b suggests that they can still provide valuable information, and Morgan fingerprints that do not rely on domain-specific knowledge are also competitive on some datasets. This contrasts with Kristiadi et al. [2024], who argue that LLMs are

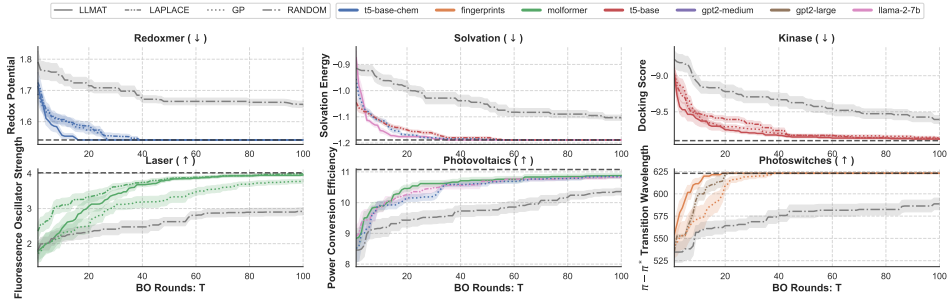


Figure 4: BO with fixed features (best-performing feature per algorithm shown). LLMAT achieves best overall performance, attaining top results on Solvation, Kinase, and Photoswitches using features from non-domain-specific models. The black dashed line marks the true optimum for each dataset.

useful for BO over molecules only when pretrained or finetuned on domain-specific data, highlighting the importance of algorithmic design to fully exploit prior in different models. We also add results with fixed embeddings from a pretrained graph neural network (GIN; Xu et al., 2019, Hu et al., 2020), comparing LLMAT against the baselines on a graph representation in addition to SMILES-based LLM features; LLMAT also achieves the best average Gap AUC there (Appendix F.11).

Improved Computational Efficiency. We also report training time, AF prediction time, and overall runtime per BO round using T5-Chem features in Fig. 20. Our method achieves the shortest training time across all datasets. While prediction time increases slightly for the largest datasets (Kinase, Laser, Photovoltaics), this overhead is offset by GP's poor performance on them. By contrast, LAPLACE incurs the highest overall cost, especially for predicting AFs during PEFT in Fig. 21, making it the least efficient.

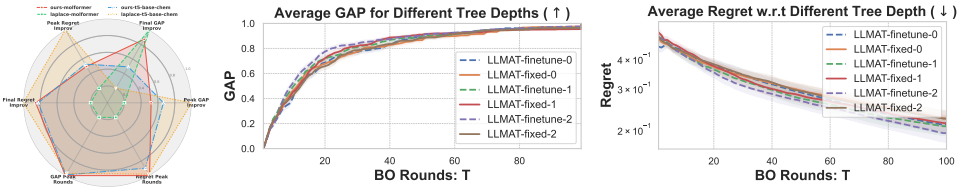


Figure 5: **Left:** Effectiveness of PEFT on T5-Chem and Molformer (radar chart of the improvements of fine-tuning over fixed features; for the peak-round scores, larger means that the maximum improvement is reached earlier, see Appendix E). **Middle and right:** Tree-depth ablation on fixed and fine-tuned MolFormer across datasets, showing the importance of partitioning.

5.3 Ablation Studies

To better understand the behavior of LLMAT, evaluate the effectiveness of its individual modules, and assess its robustness to noise, we conduct comprehensive ablation studies. In addition to average performance over all datasets, per-dataset results are provided in the Appendix.

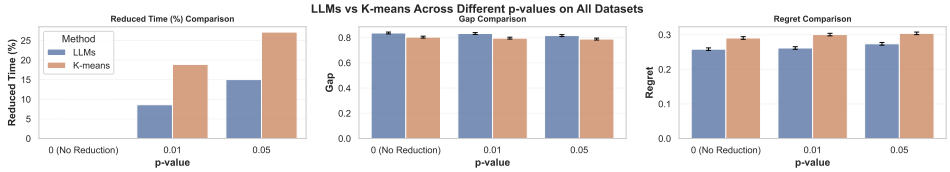


Figure 6: Average reduced prediction time percentage, average GAP, and average regret across six datasets for LLM-based clustering and K-means clustering w.r.t. different p-values.

Tree Depth Ablation. The middle and right plots in Fig. 5 present an ablation on tree depths for LLMAT, which further demonstrates the benefit of refined local AFs. It shows that using only LFBO ($L = 0$) performs worse compared to deeper tree depths. With PEFT, performance peaks at depth 2, demonstrating the benefit of deeper partitions, while with fixed features, performance is highest at depth 1 in Fig. 5. In Appendix F.17, we extend the fixed-feature ablation to $L \in \{0, \ldots, 4\}$ and

report Gap AUC: moderate partitioning improves over $L = 0$ (depth 3 is best, 80.08 vs. 78.12), while deeper trees bring no further gain.

Cluster Selection Ablation. We compare LLM-based clustering with K-means on various features across different p-values, which determine the removal of clusters with significantly poorer property values. Fig. 6 shows that LLM-based clustering achieves better average GAP and regret at p-value = 0, highlighting its superior property-awareness for BO initialization. Increasing p-values filters more clusters during AF estimation, reducing prediction time by 10–25%. While LLM-based clustering removes fewer clusters than K-means at the same p-values, its performance remains stable at 0.01 and degrades only modestly at 0.05. In contrast, K-means is more prone to removing informative clusters, causing larger performance drops. Degradation does not always follow the p-value, as it depends on the property-cluster correlation. Fig. 31 shows LLM clustering removing irrelevant clusters on Redox-mer, reducing computation and improving BO performance simultaneously.

Ablation Study for Different Algorithmic Modules. To provide a clearer understanding of the contribution of each algorithmic component in LLMAT, we conduct a detailed ablation study by selectively removing or adding modules and evaluating their impact on overall performance, as shown in the left subplot of Fig. 7. The results indicate that both MCTS and meta-learning can enhance performance. LLM-based clustering offers a slight improvement, while K-means has a negligible effect. This is expected, as the clustering methods are primarily designed to reduce computational cost rather than boost BO performance.

Sensitivity of Clustering prompts. To evaluate LLMAT's stability with respect to clustering prompts, we generated four similar prompts and compared their cluster labels agreements and cluster distributions in Fig. 9 in the Appendix. The right subplots of Fig. 7 show that the BO performance, measured by the GAP metric, remains largely consistent. This stability can be attributed to the statistical test used during cluster selection, which incorporates observed data to enhance robustness.

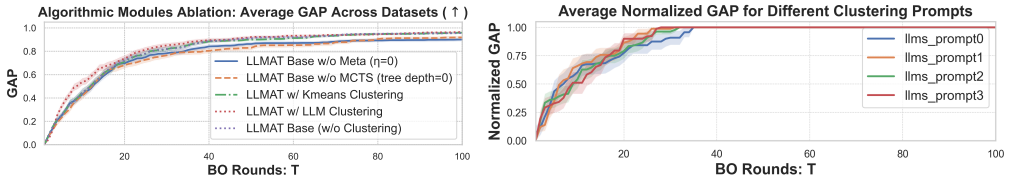


Figure 7: **Left:** Average GAP across six datasets illustrating the impact of individual algorithmic module ablations. **Right:** LLM Clustering Prompt Sensitivity in LLMAT on Redox-mer Data.

Due to space limitations, additional ablation studies on label noise, the quantile γ and the extension to multi-objective optimization are provided in Fig. 37, Fig. 38, and Fig. 41 in the Appendix, respectively.

6 Conclusion

In this paper, we introduced LLMAT, a novel surrogate-free BO algorithm that incorporates LLM and foundation model features to inform the design and effectiveness of acquisition functions in BO. Our method also introduces a learned hierarchical partitioning scheme, using Monte Carlo Tree Search with tree nodes defined by shared binary classifiers to learn local acquisition functions that are informed by LLM priors. By incorporating shared parameters and meta-learning across these binary classifiers, LLMAT achieves better AF generalization in low-data regimes and enables efficient exploration of the vast search spaces of molecules. Extensive evaluations on six chemistry datasets demonstrate that LLMAT consistently outperforms baselines on most datasets, highlighting the value of incorporating general LLMs and domain-specific foundation models with principled algorithmic design for molecular discovery.

Limitations. We focus on BO (for post-hoc verification) rather than candidate generation, so a full comparison with evolutionary and generative methods is left for future work. We also consider only 1D/text-based and 2D molecular representations; since LLMAT operates on generic pretrained representations, incorporating 3D molecular foundation models is conceptually straightforward, but evaluating them with localized BO is future work. Finally, we focus on single-objective BO; a preliminary multi-objective extension is in Fig. 41 (Appendix), while a full treatment with Pareto-front modeling and dedicated acquisition strategies is left for future work.

Acknowledgments and Disclosure of Funding

We appreciate the constructive feedback from the anonymous reviewers and meta-reviewers, and thank Jixiang Qing for proofreading before the submission. The main work was supported by grant number DSI-PDFY3R1P11 from the Data Sciences Institute at the University of Toronto. The refinement of this work continued after Qi Chen joined the ELLIS Institute Finland and Aalto University. Florian Shkurti is supported by the Canada First Research Excellence Fund (CFREF) and would also like to acknowledge funding from the AI2050 Fellowship program by Schmidt Sciences. We thank Compute Canada for computational resources, and acknowledge the computational resources provided by the Aalto Science-IT project through the Triton cluster. **Competing interests.** The authors declare no competing interests.

References

Guillermo Restrepo. Chemical space: limits, evolution and modelling of an object bigger than our universal library. *Digital Discovery*, 1(5):568–585, 2022.

Peter I. Frazier. A Tutorial on Bayesian Optimization, July 2018. URL `http://arxiv.org/abs/1807.02811`. arXiv:1807.02811 [stat].

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

Robert G Parr. Density functional theory of atoms and molecules. In *Horizons of Quantum Chemistry: Proceedings of the Third International Congress of Quantum Chemistry Held at Kyoto, Japan, October 29-November 3, 1979*, pages 5–15. Springer, 1989.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A Sober Look at LLMs for Material Discovery: Are They Actually Good for Bayesian Optimization Over Molecules?, May 2024. URL `http://arxiv.org/abs/2402.05015`. arXiv:2402.05015 [cs].

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, October 2021. URL `http://arxiv.org/abs/2106.09685`. arXiv:2106.09685 [cs].

Yilin Xie, Shiqiang Zhang, Jixiang Qing, Ruth Misener, and Calvin Tsay. BoGrape: Bayesian optimization over graphs with shortest-path encoded. *arXiv preprint arXiv:2503.05642*, 2025.

Jieyu Lu and Yingkai Zhang. Unified Deep Learning Model for Multitask Reaction Predictions with Explanation. *Journal of Chemical Information and Modeling*, 62(6):1376–1387, March 2022. ISSN 1549-960X. doi: 10.1021/acs.jcim.1c01467.

Dimitrios Christofidellis, Giorgio Giannone, Jannis Born, Ole Winther, Teodoro Laino, and Matteo Manica. Unifying Molecular and Textual Representations via Multi-task Language Modelling, May 2023. URL `http://arxiv.org/abs/2301.12586`. arXiv:2301.12586 [cs].

Kevin Maik Jablonka, Alexander Al-Feghali, Shruti Badhwar, Joshua Bocarsly, Stefan Bringuier, Kamal Choudhary, Defne Çirci, Samantha Cox, Matthew Evans, Nicolas Gastellu, Jerome Genzling, María Victoria Gil, Ankur Gupta, Wibe de Jong, Tao Liu, Sauradeep Majumdar, Garrett Merz, Nicolas Moitessier, Lynda Brinson, Beatriz Mouriño, Brenden Pelkie, Mayk Caldas Ramos, Bojana Ranković, Jacob Sanders, Ben Blaiszik, Andrew White, Ian Foster, and Ghezal Ahmad Jan Zia. 14 examples of how LLMs can transform materials science and chemistry: a reflection on a large language model hackathon. *Digital Discovery*, 2(5):1233–1250, 2023. doi: 10.1039/D3DD00113J.

Mayk Caldas Ramos, Shane S. Michtavy, Marc D. Porosoff, and Andrew D. White. Bayesian Optimization of Catalysts With In-context Learning, April 2023. URL `http://arxiv.org/abs/2304.05341`. arXiv:2304.05341 [physics].

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*, 2024a.

Daniel Flam-Shepherd and Alán Aspuru-Guzik. Language models can generate molecules, materials, and protein binding sites directly in three dimensions as XYZ, CIF, and PDB files, May 2023. URL https://arxiv.org/abs/2305.05708v1.

Kevin Maik Jablonka, Philippe Schwaller, Andres Ortega-Guerrero, and Berend Smit. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, 6(2):161–169, February 2024. ISSN 2522-5839. doi: 10.1038/s42256-023-00788-1. URL https://www.nature.com/articles/s42256-023-00788-1. Publisher: Nature Publishing Group.

Taicheng Guo, Kehan Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. What can Large Language Models do in chemistry? A comprehensive benchmark on eight tasks, December 2023. URL http://arxiv.org/abs/2305.18365. arXiv:2305.18365 [cs].

Microsoft Research AI4Science and Microsoft Azure Quantum. The Impact of Large Language Models on Scientific Discovery: a Preliminary Study using GPT-4, November 2023. URL https://arxiv.org/abs/2311.07361v2.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alán Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient Evolutionary Search Over Chemical Space with Large Language Models, July 2024. URL http://arxiv.org/abs/2406.16976. arXiv:2406.16976 [cs].

Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems*, 33:19511–19522, 2020.

Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. Sample-efficient neural architecture search by learning actions for monte carlo tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5503–5515, 2021.

Wenxuan Li, Taiyi Wang, and Eiko Yoneki. Navigating in High-Dimensional Search Space: A Hierarchical Bayesian Optimization Approach, April 2025. URL http://arxiv.org/abs/2410.23148. arXiv:2410.23148 [cs].

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. Toward global optimization, volume 2, chapter bayesian methods for seeking the extremum. 1978.

Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. 1964.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Louis C. Tiao, Aaron Klein, Matthias Seeger, Edwin V. Bonilla, Cedric Archambeau, and Fabio Ramos. BORE: Bayesian Optimization by Density-Ratio Estimation, February 2021. URL http://arxiv.org/abs/2102.09009. arXiv:2102.09009 [cs].

Jiaming Song, Lantao Yu, Willie Neiswanger, and Stefano Ermon. A General Recipe for Likelihood-free Bayesian Optimization, October 2022. URL http://arxiv.org/abs/2206.13035. arXiv:2206.13035 [cs].

Ryan-Rhys Griffiths, Leo Klarner, Henry Moss, Aditya Ravuri, Sang Truong, Yuanqi Du, Samuel Stanton, Gary Tom, Bojana Rankovic, Arian Jamasb, and others. GAUCHE: a library for Gaussian processes in chemistry. *Advances in Neural Information Processing Systems*, 36:76923–76946, 2023.

Nikita Janakarajan, Tim Erdmann, Sarath Swaminathan, Teodoro Laino, and Jannis Born. Language models in molecular discovery. In *Drug Development Supported by Informatics*, pages 121–141. Springer, 2024.

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.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Harry L Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113, 1965.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

ALEJANDRO V Levy, ANTONIO Montalvo, SUSANA Gomez, and A Calderon. Topics in global optimization. In *Numerical Analysis: Proceedings of the Third IIMAS Workshop Held at Cocoyoc, Mexico, January 1981*, pages 18–33. Springer, 1982.

Garvit Agarwal, Hieu A. Doan, Lily A. Robertson, Lu Zhang, and Rajeev S. Assary. Discovery of Energy Storage Molecular Materials Using Quantum Chemistry-Guided Multiobjective Bayesian Optimization. *Chemistry of Materials*, 33(20):8133–8144, October 2021. ISSN 0897-4756. doi: 10.1021/acs.chemmater.1c02040. URL https://doi.org/10.1021/acs.chemmater.1c02040. Publisher: American Chemical Society.

David E. Graff, Eugene I. Shakhnovich, and Connor W. Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chemical Science*, 12(22):7866–7881, June 2021. ISSN 2041-6539. doi: 10.1039/D0SC06805E. URL https://pubs.rsc.org/en/content/articlelanding/2021/sc/d0sc06805e. Publisher: The Royal Society of Chemistry.

Felix Strieth-Kalthoff, Han Hao, Vandana Rathore, Joshua Derasp, Théophile Gaudin, Nicholas H. Angello, Martin Seifrid, Ekaterina Trushina, Mason Guy, Junliang Liu, Xun Tang, Masashi Mamada, Wesley Wang, Tuul Tsagaantsooj, Cyrille Lavigne, Robert Pollice, Tony C. Wu, Kazuhiro Hotta, Leticia Bodo, Shangyu Li, Mohammad Haddadnia, Agnieszka Wołos, Rafał Roszak, Cher Tian Ser, Carlota Bozal-Ginesta, Riley J. Hickman, Jenya Vestfrid, Andrés Aguilar-Granda, Elena L. Klimareva, Ralph C. Sigerson, Wenduan Hou, Daniel Gahler, Slawomir Lach, Adrian Warzybok, Oleg Borodin, Simon Rohrbach, Benjamin Sanchez-Lengeling, Chihaya Adachi, Bartosz A. Grzybowski, Leroy Cronin, Jason E. Hein, Martin D. Burke, and Alán Aspuru-Guzik. Delocalized, asynchronous, closed-loop discovery of organic laser emitters. *Science*, 384(6697): eadk9227, May 2024. doi: 10.1126/science.adk9227. URL https://www.science.org/doi/10.1126/science.adk9227. Publisher: American Association for the Advancement of Science.

Steven A. Lopez, Edward O. Pyzer-Knapp, Gregor N. Simm, Trevor Lutzow, Kewei Li, Laszlo R. Seress, Johannes Hachmann, and Alán Aspuru-Guzik. The Harvard organic photovoltaic dataset. *Scientific Data*, 3(1):160086, September 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.86. URL https://www.nature.com/articles/sdata201686. Publisher: Nature Publishing Group.

Ryan-Rhys Griffiths, Jake L. Greenfield, Aditya R. Thawani, Arian R. Jamasb, Henry B. Moss, Anthony Bourached, Penelope Jones, William McCorkindale, Alexander A. Aldrick, Matthew J. Fuchter, and Alpha A. Lee. Data-driven discovery of molecular photoswitches with multioutput Gaussian processes. *Chemical Science*, 13(45):13541–13551, November 2022. ISSN 2041-6539. doi: 10.1039/D2SC04306H. URL https://pubs.rsc.org/en/content/articlelanding/2022/sc/d2sc04306h. Publisher: The Royal Society of Chemistry.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-Scale Chemical Language Representations Capture Molecular Structure and Properties, December 2022. URL http://arxiv.org/abs/2106.09553. arXiv:2106.09553 [cs].

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, September 2023. URL http://arxiv.org/abs/1910.10683. arXiv:1910.10683 [cs].

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

Shali Jiang, Henry Chai, Javier Gonzalez, and Roman Garnett. BINOCULARS for efficient, nonmyopic sequential experimental design. In *International Conference on Machine Learning*, pages 4794–4803. PMLR, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*, 2020.

Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-Dependent Prompt Evaluation and Optimization with Offline Inverse RL, March 2024. URL http://arxiv.org/abs/2309.06553. arXiv:2309.06553 [cs].

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers, April 2024. URL http://arxiv.org/abs/2309.03409. arXiv:2309.03409 [cs].

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers, February 2024. URL http://arxiv.org/abs/2309.08532. arXiv:2309.08532 [cs].

Elliot Meyerson, Mark J. Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K. Hoover, and Joel Lehman. Language Model Crossover: Variation through Few-Shot Prompting, May 2024. URL http://arxiv.org/abs/2302.12170. arXiv:2302.12170 [cs].

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through Large Models, June 2022. URL http://arxiv.org/abs/2206.08896. arXiv:2206.08896 [cs].

Angelica Chen, David M. Dohan, and David R. So. EvoPrompting: Language Models for Code-Level Neural Architecture Search, November 2023a. URL http://arxiv.org/abs/2302.14838. arXiv:2302.14838 [cs].

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024b.

Michael R Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large language models for hyperparameter optimization. *arXiv preprint arXiv:2312.04528*, 2023.

Qi Chen, Changjian Shui, and Mario Marchand. Generalization bounds for meta-learning: An information-theoretic analysis. *Advances in Neural Information Processing Systems*, 34:25878–25890, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/d9d347f57ae11f34235b4555710547d8-Abstract.html.

Qi Chen, Changjian Shui, Ligong Han, and Mario Marchand. On the stability-plasticity dilemma in continual meta-learning: Theory and algorithm. *Advances in Neural Information Processing Systems*, 36:27414–27468, 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/57587d8d6a7ede0e5302fc22d0878c53-Abstract-Conference.html.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and others. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization, December 2020. URL http://arxiv.org/abs/1910.06403. arXiv:1910.06403 [cs].

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th international conference on learning representations, ICLR 2018-conference track proceedings*, volume 6. International Conference on Representation Learning, 2018.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. 2022.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in neural information processing systems*, 34:20089–20103, 2021.

Rafael Oliveira, Louis Tiao, and Fabio T Ramos. Batch bayesian optimisation via density-ratio estimation with guarantees. *Advances in Neural Information Processing Systems*, 35:29816–29829, 2022.

Appendix

Table of Contents

A General LLMs Capture Relative Property Ordering, Not Absolute Values **18**

B Additional Background **19**
B.1 Indirect Estimation of Acquisition Functions 19
B.2 Parameter-Efficient Finetuning 19
B.3 Welch's ANOVA 20
B.4 Games-Howell Post-hoc Test 20

C Related Works **21**
C.1 LLMs for Optimization. 21
C.2 High-dimensional BO 21

D Clustering Prompts **22**
D.1 Redoxmer 22
D.2 Solvation 22
D.3 Kinase 23
D.4 Photovoltaics 23
D.5 Laser 24
D.6 Photoswitches 24
D.7 Variations in Prompt 25

E Experimental Details **27**
E.1 Foundation Models 27
E.2 BO on Fixed-features 27
E.3 BO with PEFT 28

F Additional Experimental Results **30**
F.1 Fixed-feature results for more foundation models 30
F.2 Fixed and finetuned results for T5-Chem Model 32
F.3 Fixed and finetuned results for Molformer 35
F.4 Clustering results 36
F.5 Ablation on p-values 37
F.6 Toy Data 40
F.7 Ablation of γ 41
F.8 Additional results for Llama-2-7b 42
F.9 Multi-objective Optimization Extension 42
F.10 Ablation over different LLM APIs 43
F.11 Fixed GNN Embeddings (GIN) 43
F.12 Informativeness of Representations 44
F.13 Morgan Fingerprints vs. Foundation-Model Embeddings 44
F.14 Ablation: Unified Classifier vs. Decoupled Partitioning and LFBO 45
F.15 Ablation: UCT vs. Greedy Selection 46
F.16 Meta-learning Dynamics at Deeper Nodes 46
F.17 Tree Depth Ablation 47
F.18 Cost of LLM-based Clustering 47

F.19 Scalability w.r.t. Candidate Pool Size 48

G Broader Impacts **48**

H Declaration of LLMs Usage **48**

I Theoretical Results **49**

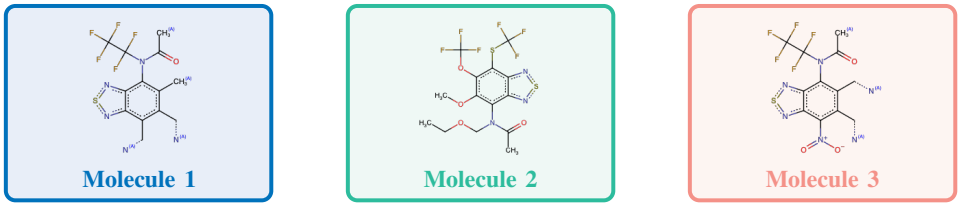
I.1 Density-ratio and divergence interpretation of local classifiers 49

I.2 Why recursive local classifiers can outperform a global model 50

I.3 Approximation–estimation–routing decomposition 50

I.4 Cumulative-regret decomposition 51

A General LLMs Capture Relative Property Ordering, Not Absolute Values



Molecule 1: CC(=O)N(c1c(C)c(C#N)c(C#N)c2nsnc12)C(F)(F)C(F)(F)F
Molecule 2: CCOCN(C(C)=O)c1c(OC)c(OC(F)(F)F)c(SC(F)(F)F)c2nsnc12
Molecule 3: CC(=O)N(c1c(C#N)c(C#N)c([N+](=O)[O-])c2nsnc12)C(F)(F)C(F)(F)F

Prompt: Estimate redox potential for {Molecule 1} {Molecule 2} and {Molecule 3}. Respond with number and unit.
DeepSeek: 2.15 V, 1.42 V, 2.35 V. **GPT-4o:** 1.42 V, 1.10 V, 2.0 V.
Ground Truth: 2.80 V, 1.54 V, 3.38 V.

Prompt: Group {Molecule 1} {Molecule 2} and {Molecule 3} into clusters {0-4, low-high} by redox potential. Respond with a number only.
DeepSeek: 3, 1, 4. **GPT-4o:** 2, 1, 4.

Figure 8: General LLMs provide coarse property ranking of redox potentials rather than accurate numeric value for the three given molecules; responses are color-matched.

Fig. 8 illustrates this on three Redox-mer molecules. When asked to estimate the redox potential directly, both DeepSeek and GPT-4o give inaccurate values: they underestimate all three molecules (e.g., DeepSeek predicts 2.15/1.42/2.35 V and GPT-4o 1.42/1.10/2.0 V, versus the ground truth 2.80/1.54/3.38 V). However, both models order the molecules correctly (Molecule 2 $<$ Molecule 1 $<$ Molecule 3), and when asked to group the molecules into clusters from low to high, they again assign them in this same order (DeepSeek: 1, 3, 4; GPT-4o: 1, 2, 4 for Molecules 2, 1, 3). General LLMs therefore cannot be trusted for absolute property values, but can still provide a coarse property ranking, which is the signal that LLM-based clustering uses.

We further run experiments across multiple LLM APIs to assess whether LLMs yield meaningful property rankings, comparing direct clustering (DC), property prediction (PP), and property-derived clustering (PC).

Table 1: Comparison of LLM APIs across property prediction and clustering metrics.

Model	PP $R^2\uparrow$	DC HunAcc$\uparrow$	DC Spearman$\uparrow$	DC PairAcc$\uparrow$	PC HunAcc$\uparrow$	PC Spearman$\uparrow$	PC PairAcc$\uparrow$
GPT-4o	-6.0016	0.3141	0.4943	0.7732	0.2459	0.1479	0.5795
Gemini-2.5-pro	-87.9568	0.4471	0.8041	**0.9084**	0.2232	0.2569	0.7163
Qwen3-Plus	-0.8572	**0.4961**	**0.8055**	0.8977	**0.3689**	**0.6510**	**0.7845**

Evaluation setup. We evaluate property prediction using $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$, where $R^2 < 0$ indicates performance worse than a mean predictor. Ground-truth clusters are obtained via **quantile-based binning into five equal-frequency groups**. DC outputs labels directly, while for PC, predicted values are mapped to clusters using the same bin edges. We assess ranking via Spearman (rank correlation) and **PairAcc (pairwise ordering accuracy, aligned with BO objectives)**, and clustering via **Hungarian accuracy (HunAcc)**. Meaningful ranking is defined as consistency with ground-truth ordering under these metrics.

This decomposition isolates where LLM clustering contributes in the pipeline.

- While property prediction is unreliable ($R^2 < 0$ for all models), DC achieves strong ranking performance (Spearman up to 0.80, PairAcc up to 0.91), **well above random** ($\approx$0 **and** 0.5).

- Importantly, PC performs substantially worse than DC, indicating that LLMs capture **relative ordering directly rather than through numerical prediction**.
- Overall, these results **show that** although LLMs fail at absolute prediction, they provide reliable ranking signals sufficient for BO, as evidenced by strong PairAcc and consistent BO performance across LLM APIs (Fig. 42).

A related concern is that LLMs may have memorized property values of these public datasets. The negative R^2 of property prediction (Table 1) indicates that the LLMs do not reproduce numerical values, although we cannot rule out that they have seen coarse trends from public data.

B Additional Background

B.1 Indirect Estimation of Acquisition Functions

Based on the definition of the acquisition function in Eq.(1), it is natural to first calculate the predictive posterior $p(y|x, \mathbf{D}_t)$, then estimate the AFs given the selected utility functions. In the following, we introduce two typical surrogate models that are non-parametric and parametric, respectively.

Gaussian Process (GP). A typical choice of a non-parametric surrogate model $\hat{f}(x)$ is a GP with $p(\hat{f}(x)|\mathbf{D}_t) = \mathcal{N}(\mathbf{k}_t^T \mathbf{C}_t^{-1} \mathbf{y}_t, k(x,x) - \mathbf{k}_t^T \mathbf{C}_t^{-1} \mathbf{k}_t)$ being Gaussian given some kernel function $k(\cdot,\cdot)$, where $\mathbf{k}_t[i] = k(x_i, x) \forall i \in [t]$, $\mathbf{K}_t[i,j] = k(x_i, x_j), \forall i,j \in [t]$, and $\mathbf{C}_t = \mathbf{K}_t + \sigma^2 \mathbf{I}_t$. In this case, the predictive posterior is tractable and has the form of $p(y|x, \mathbf{D}_t) = \mathcal{N}(\mathbf{k}_t^T \mathbf{C}_t^{-1} \mathbf{y}_t, k(x,x) + \sigma^2 - \mathbf{k}_t^T \mathbf{C}_t^{-1} \mathbf{k}_t)$. However, the basic GPs have a $\mathcal{O}(t^3)$ computational complexity for training, even sparse GPs still require $\mathcal{O}(t)$, posing difficulty to scale to a large dataset of observations.

Bayesian Neural Network (BNN). If we consider parametric surrogate model $\hat{f}(x) = \text{NN}(\theta, x)$ with $\text{NN}(\theta, \cdot)$ being a predefined Neural Network (NN) of parameter θ, then, the predictive posterior of BNN is $p(y|x, \mathbf{D}_t) = \int p(y|x,\theta) p(\theta|\mathbf{D}_t) d\theta$. The posterior $p(\theta|\mathbf{D}_t)$ is often not analytically tractable. Hence, solutions like Laplace Approximation (LA), Variational Inference (VI) and Monte Carlo Sampling are often adopted to approximate the integral.

B.2 Parameter-Efficient Finetuning

Fine-tuning large pretrained models on downstream tasks often requires updating all model parameters, which can be computationally expensive and memory-intensive. Parameter-efficient fine-tuning (PEFT) methods aim to reduce this overhead by introducing a small set of trainable parameters while keeping the original pretrained model weights frozen.

Among various PEFT methods, *Low-Rank Adaptation* (LoRA) Hu et al. [2021] has gained popularity due to its efficiency and effectiveness. LoRA hypothesizes that the update to the pretrained weight matrices can be approximated by a low-rank decomposition. Formally, consider a pretrained weight matrix $W_0 \in \mathbb{R}^{d \times k}$ in a neural network (e.g., a linear layer or attention projection). Instead of updating W_0 directly, LoRA introduces two trainable matrices $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$, where $r \ll \min(d,k)$ is the rank of the adaptation:

$$W = W_0 + \Delta W = W_0 + BA,$$

where BA is a low-rank matrix capturing task-specific updates. During fine-tuning, W_0 is kept frozen, and only A and B are optimized. For an input $x \in \mathbb{R}^k$, the layer output becomes

$$y = Wx = W_0 x + BAx.$$

To further control the magnitude of updates, a scaling factor α is often introduced:

$$W = W_0 + \frac{\alpha}{r} BA,$$

where the factor $\frac{\alpha}{r}$ ensures the update has a similar scale across different rank settings. This formulation significantly reduces the number of trainable parameters from $d \times k$ to $r(d+k)$ while retaining strong adaptation capability.

B.3 Welch's ANOVA

Suppose we have k clusters, where cluster i has sample size n_i, sample mean $\bar{x}_i$, and sample variance σ_i^2. Define the weights:

$$w_i = \frac{n_i}{\sigma_i^2}, \quad i = 1, \ldots, k,$$

and the weighted mean:

$$\bar{x}_w = \frac{\sum_{i=1}^{k} w_i \bar{x}_i}{\sum_{i=1}^{k} w_i}.$$

The Welch F-statistic is then

$$F = \frac{\sum_{i=1}^{k} w_i (\bar{x}_i - \bar{x}_w)^2}{(k-1)\left[1 + \frac{2(k-2)}{k^2-1} \sum_{i=1}^{k} \frac{(1 - w_i / \sum_j w_j)^2}{n_i - 1}\right]}.$$

The approximate degrees of freedom for the denominator is given by the Satterthwaite approximation:

$$\nu \approx \frac{k^2 - 1}{3 \sum_{i=1}^{k} \frac{(1 - w_i / \sum_j w_j)^2}{n_i - 1}}.$$

Finally, F is compared against an F-distribution with $(k-1, \nu)$ degrees of freedom to compute the p-value.

B.4 Games-Howell Post-hoc Test

Suppose we have k clusters, where cluster i has sample size n_i, sample mean $\bar{x}_i$, and sample variance σ_i^2. For a pair of groups (i, j), the test statistic is

$$t_{ij} = \frac{\bar{x}_i - \bar{x}_j}{\sqrt{\frac{\sigma_i^2}{n_i} + \frac{\sigma_j^2}{n_j}}}.$$

The degrees of freedom for this comparison are approximated using the Welch–Satterthwaite equation:

$$\nu_{ij} = \frac{\left(\frac{\sigma_i^2}{n_i} + \frac{\sigma_j^2}{n_j}\right)^2}{\frac{(\sigma_i^2/n_i)^2}{n_i - 1} + \frac{(\sigma_j^2/n_j)^2}{n_j - 1}}.$$

The p-value is computed from the studentized range distribution with k groups and ν_{ij} degrees of freedom:

$$p_{ij} = 1 - F_q\Big(\sqrt{2}\,|t_{ij}|;\, k, \nu_{ij}\Big),$$

where F_q is the cumulative distribution function of the studentized range distribution. This accounts for multiple comparisons across the k groups, so no additional correction is needed. Our implementation uses the Games-Howell test from the `pingouin` package.

C Related Works

C.1 LLMs for Optimization.

Recent research has increasingly leveraged pretrained Large Language Models (LLMs) as informative priors in optimization tasks across a variety of domains. For instance, LLMs have been used to improve prompt optimization strategies by conditioning on query-dependent representations [Sun et al., 2024, Yang et al., 2024, Guo et al., 2024]. Other works have integrated LLMs into evolutionary algorithms, using them as generative operators to evolve prompts or candidate solutions [Meyerson et al., 2024, Lehman et al., 2022, Chen et al., 2023a]. In the context of Bayesian Optimization (BO), LLMs have also been employed to guide acquisition functions, either for hyperparameter tuning [Liu et al., 2024b, Zhang et al., 2023] or for scientific discovery tasks such as molecular optimization [Kristiadi et al., 2024, Ramos et al., 2023].

C.2 High-dimensional BO

Several approaches have been developed to scale BO to high-dimensional search spaces, including structural assumptions, subspace embedding, variable selection, and local modeling and space partitioning. Wang et al. [2020, 2021] progressively bifurcate the search space into more promising and less promising subspaces, starting with the full candidate space Ω. K-means clustering is applied to observed data (features x and values y) within each space to split the data into "good" and "bad" clusters based on the average values of the clusters. These labeled clusters are then used to train a Support Vector Machine (SVM) classifier, which produces a non-linear decision boundary to define latent actions for bifurcating the search space. Finally, local surrogate models for BO are trained on observed data in the selected subspace via Upper Confidence Bound (UCB). Based on Wang et al. [2020], Li et al. [2025] propose to reweigh the samples in different partitions by their UCB values without constraining sampling to certain partitions. In contrast, we do not separately use SVMs for partitioning and additional models for acquisition function estimation.

We focus on leveraging LLM-derived priors to enhance high-dimensional Bayesian Optimization (BO) for molecular discovery by using LLMs for feature extraction, finetuning and prompting for clustering. Unlike prior work, we introduce a shared neural network classifier that simultaneously partitions the candidate space and estimates acquisition functions at each tree node, effectively amortizing both tasks. To further improve data efficiency, we adopt a meta-learning [Chen et al., 2021, 2023b] approach that amortizes classifier training across nodes, enabling more stable learning in early data-scarce stages. Moreover, while previous methods assume a continuous search space and rely on rejection sampling, they fail in discrete settings where selected regions may contain no valid candidates. We address this limitation via a backtracking strategy. Our method traverses a UCT-guided path through the search tree, estimating local acquisition functions from leaf to parent until valid candidates are found. In the worst case, it returns to the original Likelihood-Free BO (LFBO) approach.

D Clustering Prompts

D.1 Redoxmer

PROMPT | Redoxmer

You are a chemist, please use the molecules provided, group them into five clusters based on their redox potential.

> **Clustering Scale:** extremely low: 0, low: 1, medium: 2, high: 3, extremely high: 4

Analyze the following features for each molecule:

- Number and type of electron-withdrawing groups (e.g., CF_3, NO_2, CN, halogens)
- Number and type of electron-donating groups (e.g., alkyl, methoxy, hydroxyl)
- Positioning of substituents on aromatic rings (meta, para, ortho)
- Presence of sulfur-containing functional groups (e.g., SCF_3, S=O)
- Degree of molecular polarity (based on F, O, or N atoms)

Use these criteria to evaluate the redox potential qualitatively and cluster the molecules accordingly.

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

D.2 Solvation

PROMPT | Solvation

You are a chemist, please use the molecules provided, group them into five clusters based on predicted solvation energy.

> **Clustering Scale:** extremely low: 0, low: 1, medium: 2, high: 3, extremely high: 4

Consider these molecular features:

- Polarity and hydrogen bonding capability
- Molecular size and surface area
- Number and type of charged/ionizable groups
- Hydrophilic/hydrophobic balance

Use these criteria to evaluate the solvation energy qualitatively and cluster the molecules accordingly.

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

D.3 Kinase

PROMPT | Kinase

Act as a computational chemist. You have a list of SMILES strings for molecules, and I want to group them into 5 clusters (0 to 4) where cluster 0 has the lowest predicted kinase docking affinity and cluster 4 has the highest.

> **Clustering Scale:** extremely low: 0, low: 1, medium: 2, high: 3, extremely high: 4

Prioritize these criteria:

- Presence of kinase-binding motifs (e.g., hinge-binding heterocycles, hydrophobic pockets)
- Functional groups (e.g., hydrogen bond donors/acceptors, aromatic rings)
- Molecular weight and polarity (smaller/lipophilic molecules often bind kinases better)
- Similarity to known kinase inhibitors (e.g., ATP analogs, tyrosine kinase inhibitors)

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

D.4 Photovoltaics

PROMPT | Photovoltaics

You are a chemist, please use the molecules provided, group them into five clusters based on predicted photovoltaic conversion efficiency.

> **Clustering Scale:** extremely low: 0, low: 1, medium: 2, high: 3, extremely high: 4

Consider these molecular features:

- Electronic structure indicators (conjugation extent, aromatic systems, electron-rich/deficient regions, push-pull molecular design)
- Molecular architecture (planarity potential, π-system connectivity, structural rigidity, molecular size)
- Light harvesting features (conjugated backbone length, donor-acceptor patterns, chromophore presence, substituent effects)

Use these criteria to evaluate the photovoltaic conversion efficiency qualitatively and cluster the molecules accordingly.

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

D.5 Laser

PROMPT | Laser

Act as a computational chemist with expertise in photophysics. Group these SMILES strings into 5 clusters based on predicted fluorescence oscillator strength (relevant for lasers).

> **Clustering Scale:** very low: 0, low: 1, moderate: 2, high: 3, very high: 4

Consider these factors:

- Conjugation length: Longer conjugation increases oscillator strength
- Aromaticity: Aromatic systems often have strong π-π^* transitions
- Functional groups: Electron-donating/withdrawing groups alter oscillator strength
- Molecular rigidity: Rigid molecules tend to have higher oscillator strength

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

D.6 Photoswitches

PROMPT | Photoswitches

Act as a computational chemist with expertise in photochemistry. You have a list of SMILES strings for organic molecules, and want to group them into 5 clusters based on their predicted π-π^* transition wavelengths.

> **Clustering Scale:**
> 1. Cluster 0: deep UV range (200–300 nm)
> 2. Cluster 1: UV range (300–400 nm)
> 3. Cluster 2: blue/visible range (400–500 nm)
> 4. Cluster 3: green/red/visible range (500–700 nm)
> 5. Cluster 4: near-infrared range (700–1000 nm)

Use your knowledge of molecular structure and photochemistry to analyze the SMILES strings and assign cluster labels. **Consider the following factors:**

- **Conjugation length:** Longer conjugation typically shifts the π-π^* transition to longer wavelengths
- **Aromaticity:** Aromatic systems often have π-π^* transitions in the UV/visible range
- **Functional groups:** Electron-donating or electron-withdrawing groups can alter the HOMO-LUMO gap
- **Molecular planarity:** Planar molecules tend to have stronger π-π^* transitions

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

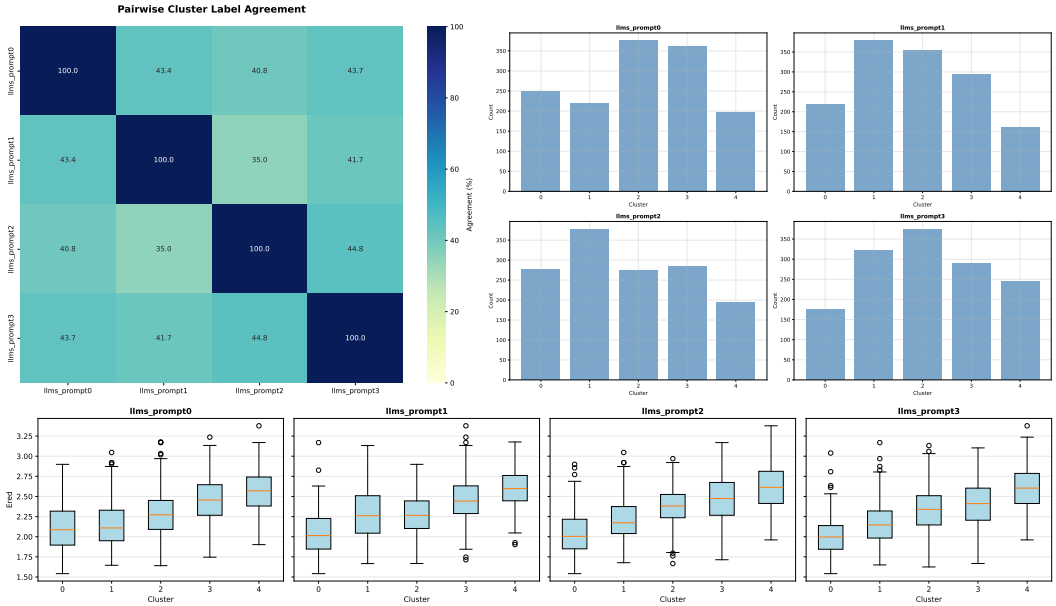


Figure 9: Prompt variation's effect on cluster labels.

D.7 Variations in Prompt

We ask ChatGPT to generate 3 alternative prompts (Prompt1–3; Prompt0 is the original) based on our original prompt, the cluster label disagreements, the label distributions, and the generated prompts are listed below.

PROMPT | Prompt0

You are a chemist, please use the molecules provided, group them into five clusters based on their redox potential (extremely low : 0, low: 1, medium: 2, high: 3, extremely high :4).

Analyze the following features for each molecule:

- Number and type of electron-withdrawing groups (e.g., CF_3, NO_2, CN, halogens).
- Number and type of electron-donating groups (e.g., alkyl, methoxy, hydroxyl).
- Positioning of substituents on aromatic rings (meta, para, ortho).
- Presence of sulfur-containing functional groups (e.g., SCF_3, S=O).
- Degree of molecular polarity (based on F, O, or N atoms).

Use these criteria to evaluate the redox potential qualitatively and cluster the molecules accordingly.

Response Format: Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

PROMPT | Prompt1

Act as a chemist. Using the given molecules, assign each one to a redox-potential cluster: 0 = extremely low, 1 = low, 2 = medium, 3 = high, 4 = extremely high.

Evaluate each molecule based on:

- Electron-withdrawing substituents (CF_3, NO_2, CN, halogens, etc.)
- Electron-donating substituents (alkyl, OMe, OH, etc.)
- Substituent positions on aromatic systems (ortho/meta/para)
- Sulfur-containing functional groups (SCF_3, sulfoxides, etc.)
- Overall polarity from heteroatoms (F/O/N)

Use these qualitative features to estimate redox strength and cluster the molecules.

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

PROMPT | Prompt2

You are an expert computational chemist. For each provided molecule, perform the following steps: Identify electron-withdrawing and electron-donating groups. Determine substituent positions on aromatic cores. Note any sulfur-based functionalities (e.g., SCF_3, S=O). Assess molecular polarity based on heteroatom composition. Infer the molecule's qualitative redox potential from these structural features. Then group all molecules into five redox-potential clusters (0–4) from extremely low to extremely high.

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

PROMPT | Prompt3

As a chemist specializing in structure–property relationships, analyze each molecule and assess its qualitative redox potential. Consider:

- Strength and number of EWG vs EDG
- Aromatic substitution pattern and resonance effects
- Sulfur-functional group contributions
- Polarity arising from heteroatoms

Based on the identified features, explain your reasoning and categorize each molecule into one of five redox classes (0: extremely low $\rightarrow$ 4: extremely high).

> **Response Format:** Respond strictly with the counter and numerical cluster labels only. Do not include any additional text.

Now please cluster the following molecules:

```
{molecules to be inserted here}
```

E Experimental Details

The code for LLMAT and all experiments is available at `https://github.com/livreQ/LLMAT`.

We follow most of the settings in Kristiadi et al. [2024] for using the foundation models and baselines: GP and Laplace, while we refactored their code (repo `https://github.com/wiseodd/lapeft-bayesopt`) for better extension. The LFBO and BORE baselines are LLMAT with only the root classifier (no tree), using the same MLP and $\eta = 0.005$; LFBO uses EI weighting and BORE uses PI weighting. All methods run for 100 BO rounds.

Radar Chart Metrics. The radar chart (Fig. 5, left) summarizes the benefit of fine-tuning the representation relative to a fixed one, considering both the magnitude of the improvement and how early it is achieved. At BO round t, we define $\Delta\mathrm{GAP}_t = \mathrm{GAP}_t^{\mathrm{FT}} - \mathrm{GAP}_t^{\mathrm{Fixed}}$ and $\Delta\mathrm{Regret}_t = \mathrm{Regret}_t^{\mathrm{Fixed}} - \mathrm{Regret}_t^{\mathrm{FT}}$, such that a positive value indicates an advantage of fine-tuning, and denote their means across runs by $\overline{\Delta\mathrm{GAP}}_t$ and $\overline{\Delta\mathrm{Regret}}_t$. The six axes are: *Peak GAP Improv.*, $\max_t \overline{\Delta\mathrm{GAP}}_t$; *Final GAP Improv.*, the mean of $\overline{\Delta\mathrm{GAP}}_t$ over the final 10 BO rounds; *Peak Regret Improv.*, $\max_t \overline{\Delta\mathrm{Regret}}_t$; *Final Regret Improv.*, the mean of $\overline{\Delta\mathrm{Regret}}_t$ over the final 10 BO rounds; *GAP Peak Rounds*, a score of how early the maximum GAP improvement is achieved; and *Regret Peak Rounds*, the analogous score for regret. For the last two, we first find the round of the maximal mean improvement, e.g., $t_{\mathrm{peak}} = \arg\max_t \overline{\Delta\mathrm{GAP}}_t$. Since reaching the maximum improvement earlier indicates better sample efficiency, we transform the raw peak round into a reversed 1–10 score $s_{\mathrm{peak}} = \max\left(1, \min(10,\ 11 - \frac{t_{\mathrm{peak}}}{10})\right)$, so that peaks at rounds 10, 50 and 100 receive scores of 10, 6 and 1, respectively. Hence "larger is better" refers to this transformed score, not to the raw number of BO rounds. For visualization, all axes are further normalized to $[0.2, 1.0]$; the lower bound of 0.2 only keeps the smallest values visible.

Computational Resource The experiments were conducted on multiple server nodes, each equipped with 6 CPUs and a single GPU with 32 GB of memory. To obtain results faster, the experiments were run across different clusters. We ensured that, for each dataset, all algorithms were tested on the same type of CPU and GPU. LLM-based clustering with closed-source APIs costs about $2.59 (GPT-4o), $1.81 (Gemini-2.5-Pro), or $0.47 (Qwen3-Plus) for one clustering pass over all six datasets (Sec. F.18).

E.1 Foundation Models

Features and Prompts for Foundation Model. For the LLM features, we average the last transformer embeddings along the sequence dimension, ignoring padding and EOS tokens. All LLM-related components in this work were implemented using the Hugging Face Transformers library Wolf et al. [2019]. We use the single SMILES string as the prompt input for feature extraction and parameter-efficient finetuning of these foundation models.

E.2 BO on Fixed-features

We use the same batch size 256 for AFs estimation (*i.e.*, prediction) for all the algorithms.

E.2.1 Training Details

GP For GP, we use BoTorch [Balandat et al., 2020], with the Tanimoto kernel from Gauche [Griffiths et al., 2023] for Morgan fingerprints and a Matérn kernel for the other features. The marginal likelihood is optimized using Adam [Kingma and Ba, 2014] with a learning rate of 0.01 for 500 epochs.

Laplace A 2-hidden-layer multilayer perceptron with 50 units per layer is used. The network is optimized with Adam at a learning rate of 1×10^{-3} and weight decay 5×10^{-4} for 500 epochs with batch size 20, using cosine annealing for the learning rate [Loshchilov and Hutter, 2016]. The Laplace approximation is applied post hoc, with prior precision tuned via marginal likelihood for 100 iterations. The Hessian is approximated using a Kronecker structure [Ritter et al., 2018] (a diagonal approximation is used for Llama-2-7b).

LLMAT We build the MCTS trees that satisfy the tree depth and minimal leaf sample size constraints. A node is split only if it holds more than `leaf_sample_size`$= 2$ observations, so nodes deeper in the tree can have very few training points for their classifiers; Appendix F.16 examines training at such deeper nodes. Then we train classifiers that are multilayer perceptrons with one hidden layer of 50 units using ReLU activation. The classifiers are trained with Adam of learning rate 1×10^{-2} and weight decay 5×10^{-4} for 50 epochs with batch size 256. Other hyperparameters like the quantile γ, the λ for UCT, the meta-learning rate η, the threshold for partitioning, the tree depth, and the p-value are summarized in Tab. 2.

Table 2: Hyperparameter configuration of LLMAT with fixed features. Where several values are listed, they are used per feature model: T5-Chem η=0.01, p=0; MolFormer/fingerprints/GIN η=0.005, p=0; T5/GPT-2/Llama η=0.005, p=0.01. `tree_depth`=3 in the code corresponds to L=1 in the paper (3 nodes).

Category	Parameter	Value
General	γ	0.5
	λ	0.5
	η	0.005, 0.01
	tree_depth	3
	p_val	0, 0.01, 0.05
	threshold	0.5
Fix Args	batch_size	256
	Head Parameters:	
	n_epochs	50
	learning rate	1e-2
	batch_size	256
	leaf_sample_size	2

E.3 BO with PEFT

We keep the same LoRA configuration for our algorithm and the Laplace baseline. The batch size for acquisition function estimation was set to 16 for T5-Chem and 32 for Molformer. Same batch size was also used for LoRA training.

LoRA Configuration. We used LoRA with rank 4, applied without bias on the query and value attention weights. The scaling factor α was set to 16, and dropout with probability 0.1 was applied. We followed the implementation from HuggingFace's PEFT library [Mangrulkar et al., 2022].

E.3.1 Training details of Laplace

In Kristiadi et al. [2024], the following setting was applied to PEFT with Laplace Approximation.

LoRA Training. The LoRA parameters and the regression head were jointly trained using AdamW with learning rates of 3×10^{-4} and 1×10^{-3} for the LoRA and regression head weights, respectively (except for the Photoswitch dataset, where they used 3×10^{-3} and 1×10^{-2}). Training was performed for 50 epochs with weight decay 0.01. Subsequently, the regression head was optimized for 100 epochs under the same hyperparameters.

Laplace Approximation. The Laplace approximation was applied to both the LoRA and regression head weights. We used a Kronecker-factored Hessian and optimized the layerwise prior precisions with post hoc marginal likelihood for 200 iterations, following Daxberger et al. [2021].

E.3.2 Training details of our algorithm

Following the fixed-feature setting, we construct MCTS trees subject to tree-depth and minimum leaf-sample constraints, and train MLP classifiers. To enable larger classifier batch sizes, batching is applied after the feature extractor and LoRA layers. Classifiers are trained with Adam (lr 1×10^{-2},

weight decay 5×10^{-4}, batch size 256, 50 epochs). LoRA learning rates are task-specific: 3×10^{-5} (Solvation, Kinase), 5×10^{-5} (Redoxmer, Laser), and 1×10^{-7} (Photovoltaics, Photoswitch). Additional hyperparameters, including γ, λ, η, partitioning threshold, tree depth, and p-value, are listed in Tabs. 2 and 3.

Table 3: Hyperparameter configuration of LLMAT with PEFT.

Category	Parameter	Value
PEFT	batch_size	16, 32
	Head Parameters:	
	n_epochs	50
	learning rate	1e-2
	batch_size	256
	leaf_sample_size	2
	LoRA Parameters:	
	n_epochs	50
	learning rate	3e-5, 5e-5, 1e-7
	batch_size	16, 32

F Additional Experimental Results

Gap AUC. To summarize a whole trajectory in one number, the tables in this appendix report the **Gap AUC**, the area under the GAP curve (Sec. 5.1) over all BO rounds: Gap AUC $:= \sum_{t=1}^{T-1} \frac{1}{2}\big(\mathrm{GAP}_t + \mathrm{GAP}_{t+1}\big)$, computed per run with $T = 100$ rounds (trapezoidal rule), where y_0 is the best value of the initial design and GAP_t is clipped to $[0, 1]$, and then averaged over runs. It lies in $[0, T-1]$, and higher is better.

In this section, we present additional experimental results on best observed values, GAP metrics, regret, and computational costs for each dataset, covering all fixed feature models, finetuned T5-Chem and MolFormer, clustering and p-value ablations, toy data, the Llama-2-7b results, the multi-objective extension, the LLM-API ablation, and further analyses of representations, tree depth, and cost.

F.1 Fixed-feature results for more foundation models

F.1.1 Historical optimums

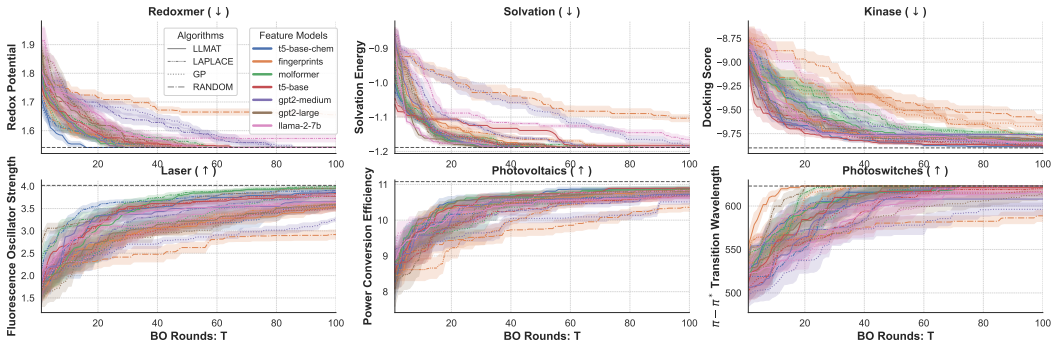


Figure 10: BO process for different datasets on features extracted from various foundation models.

F.1.2 Computation Time

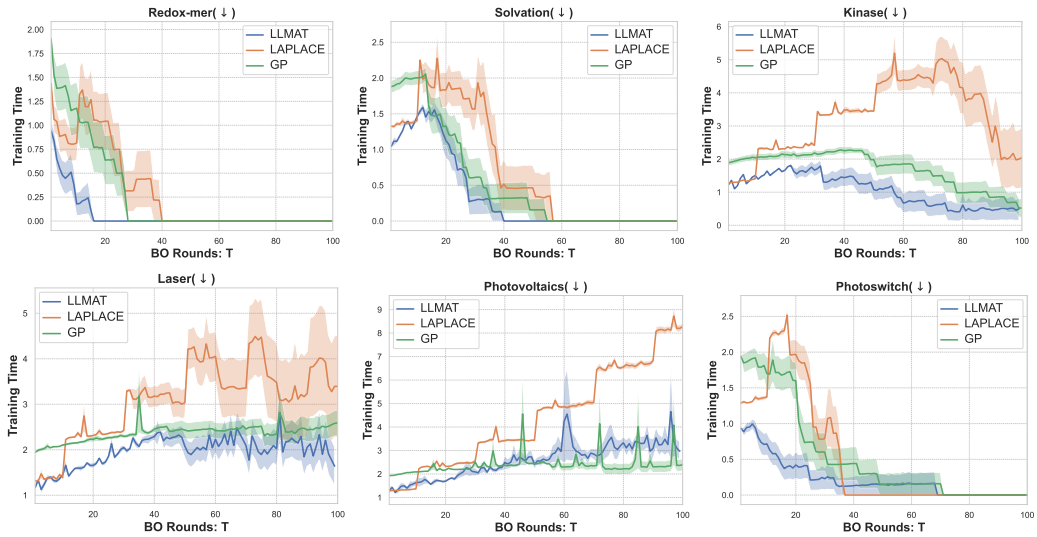


Figure 11: Training time comparison on different chemistry datasets with fixed T5-Chem features.

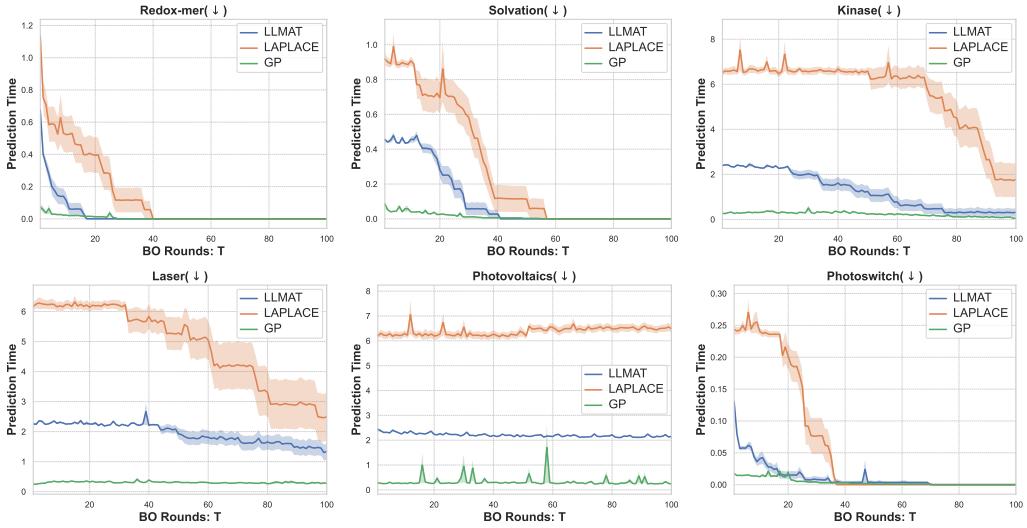


Figure 12: Prediction time comparison on different chemistry datasets with fixed T5-Chem features.

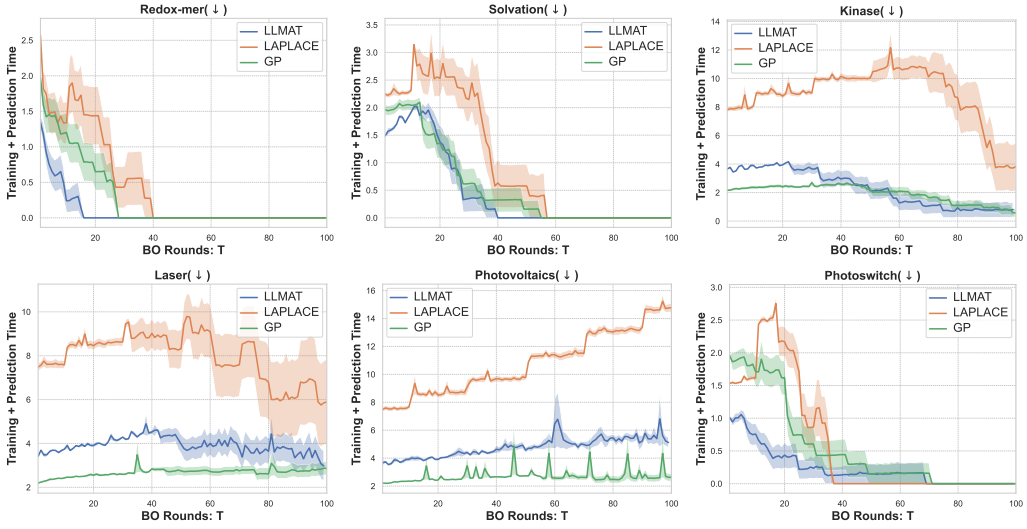


Figure 13: Overall time comparison on different chemistry datasets with fixed T5-Chem features.

F.2 Fixed and finetuned results for T5-Chem Model

F.2.1 Historical optimums

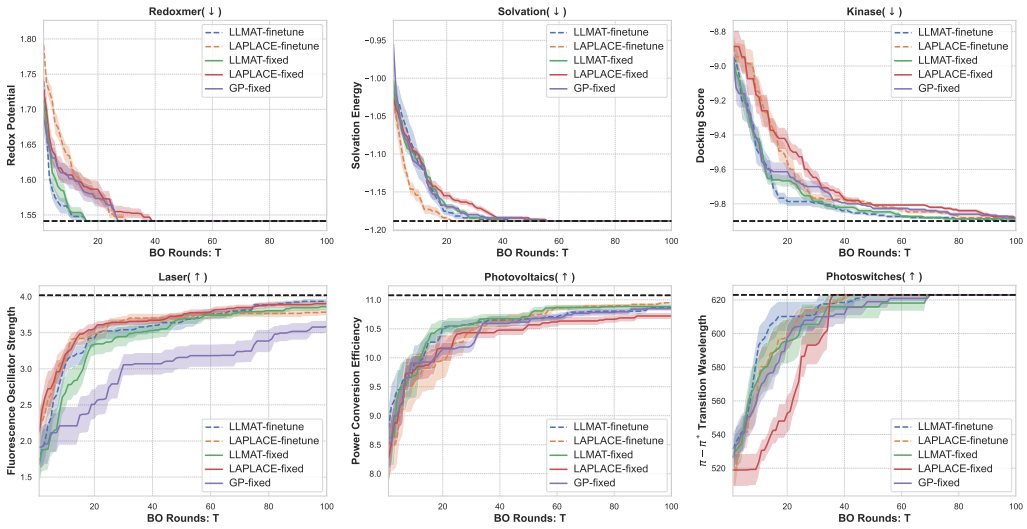


Figure 14: Historical optimums comparison on different chemistry datasets using T5-Chem model.

F.2.2 GAP metric

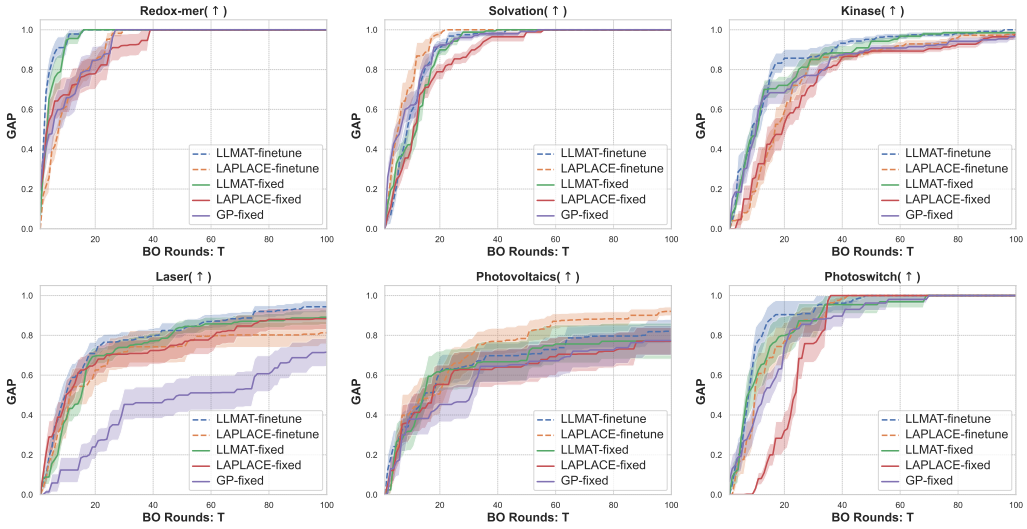


Figure 15: GAP comparison on different chemistry datasets with T5-Chem model.

F.2.3 Average Regret

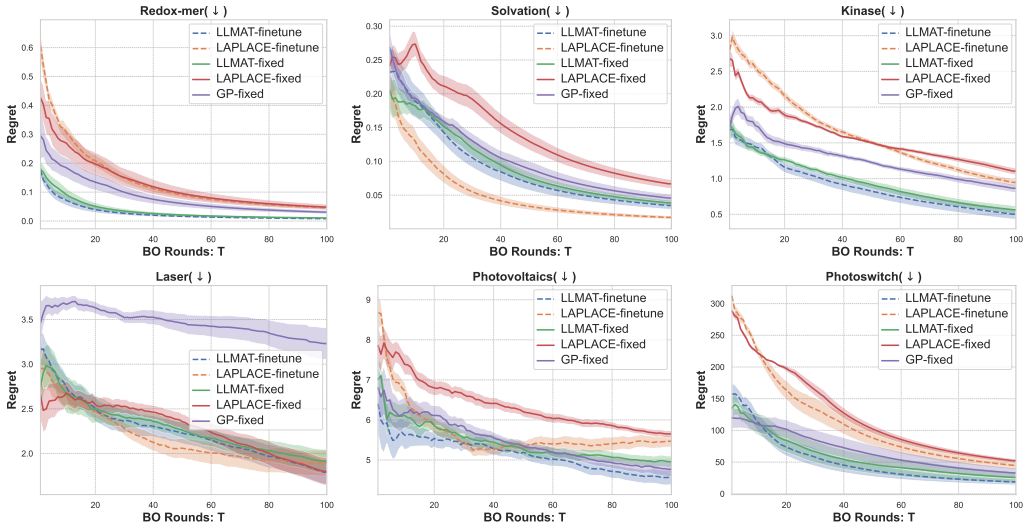


Figure 16: Regrets comparison on different chemistry datasets using T5-Chem model.

F.2.4 Computation time for PEFT

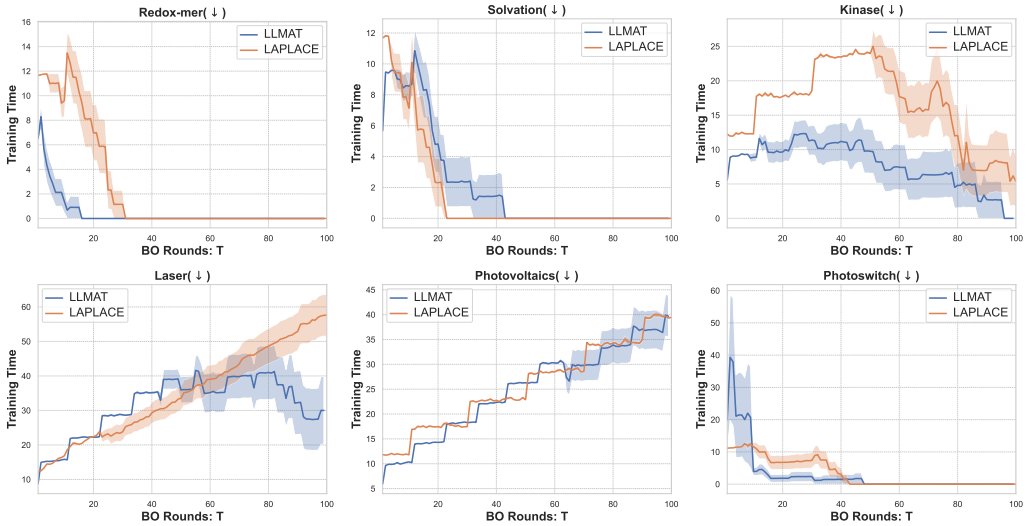


Figure 17: Training time comparison on different chemistry datasets with PEFT on the T5-Chem model.

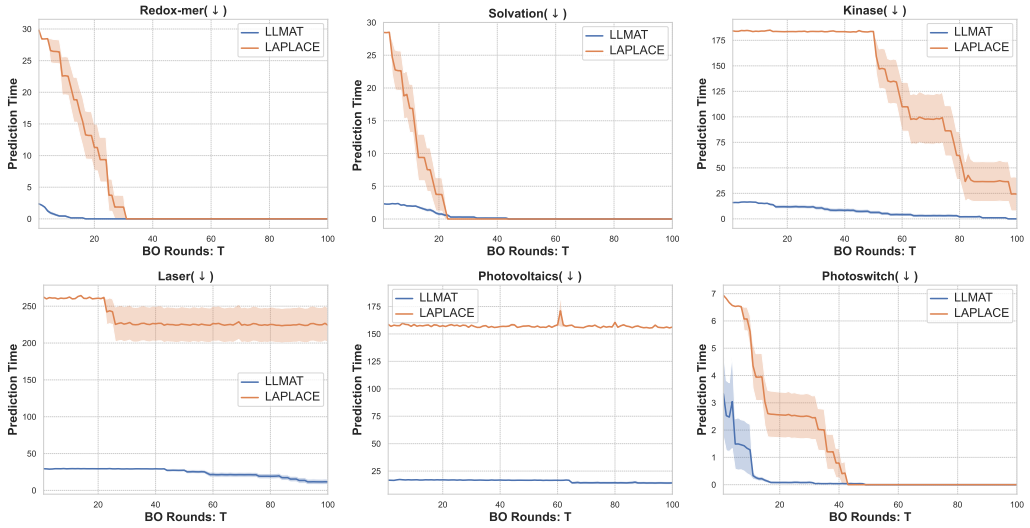


Figure 18: Prediction time comparison on different chemistry datasets with PEFT on the T5-Chem model.

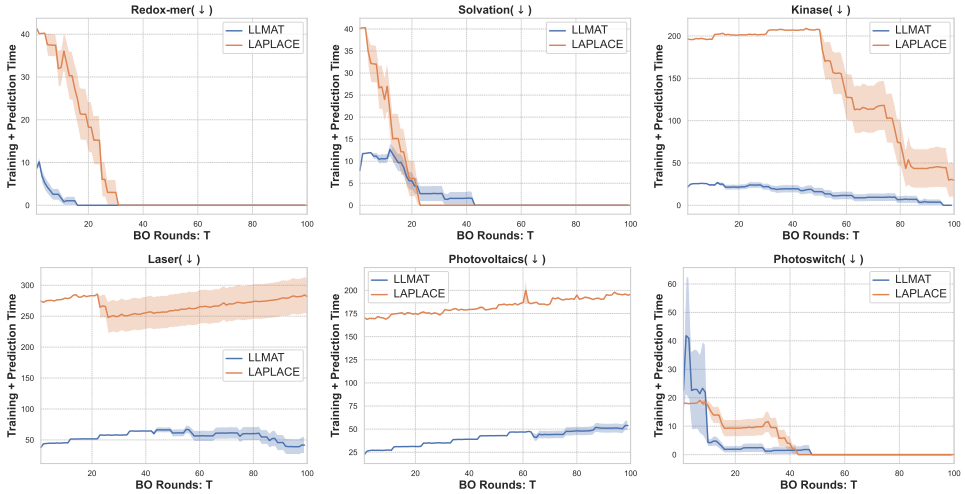


Figure 19: Overall time comparison on different chemistry datasets with PEFT on the T5-Chem model.

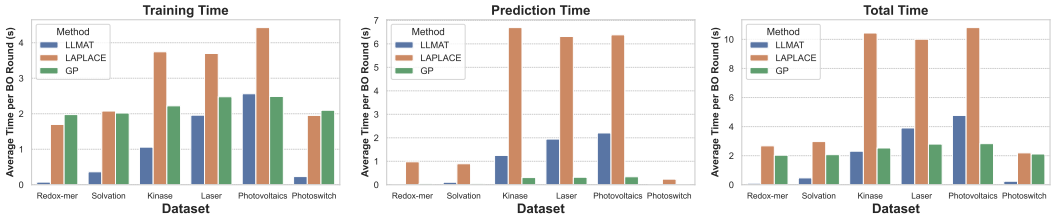


Figure 20: Computation time of different algorithms with fixed T5-Chem features across datasets.

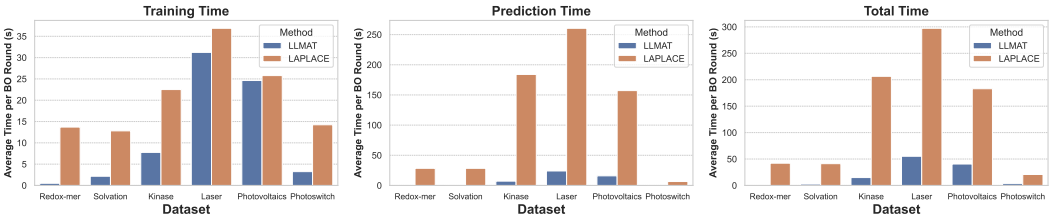


Figure 21: BO time for different datasets on finetuning T5-Chem model with LoRA.

F.3 Fixed and finetuned results for Molformer

F.3.1 Historical optimums

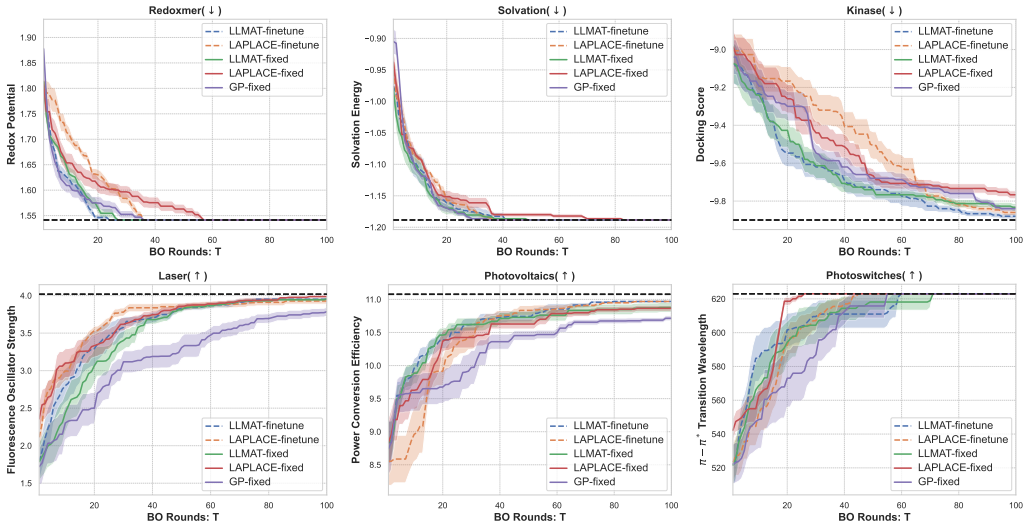


Figure 22: Historical optimums comparison on different chemistry datasets using Molformer.

F.3.2 Average regrets

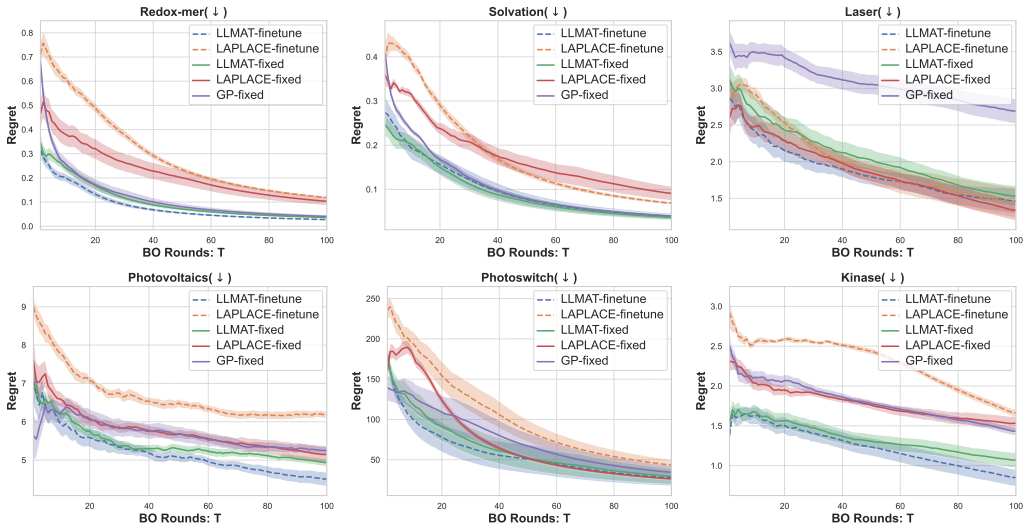


Figure 23: Regrets comparison on different chemistry datasets using Molformer.

F.3.3 GAPs

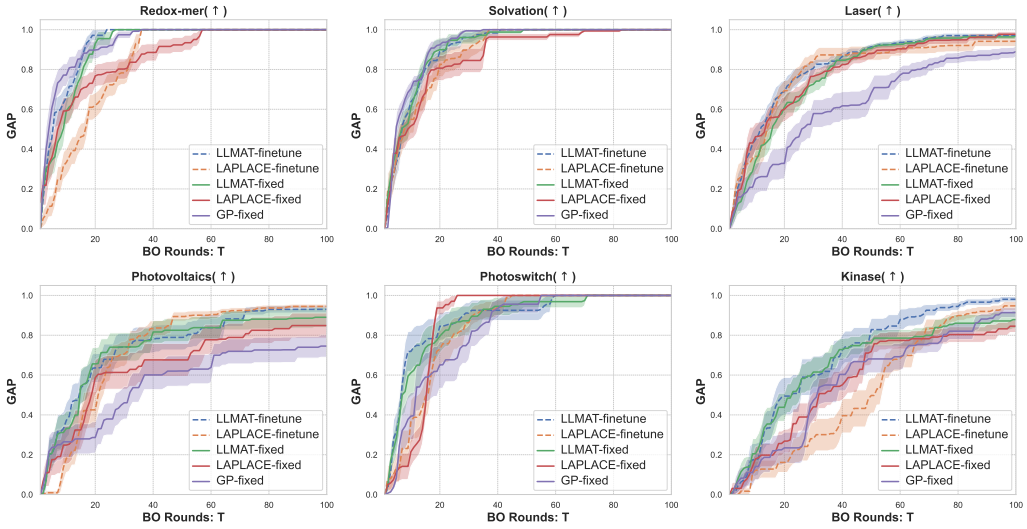


Figure 24: GAPs comparison on different chemistry datasets using Molformer

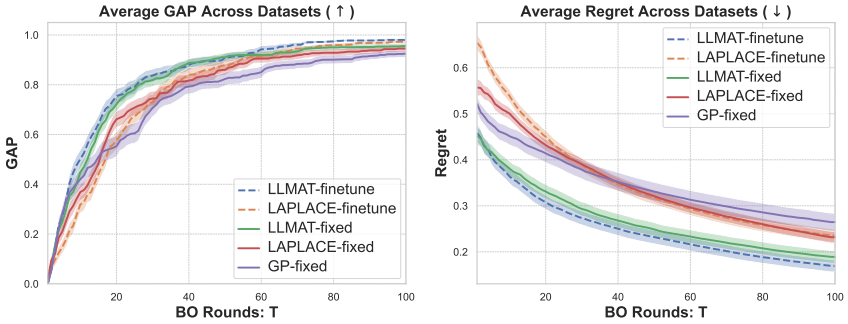


Figure 25: Average GAPs and regrets across all the datasets on finetuning and fixed Molformer.

F.4 Clustering results

F.4.1 K-means clustering

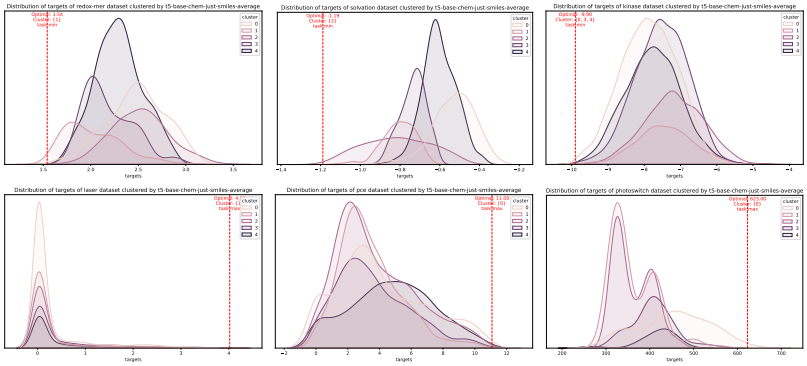


Figure 26: Distribution of property values for K-means clusters on features extracted from T5-Chem.

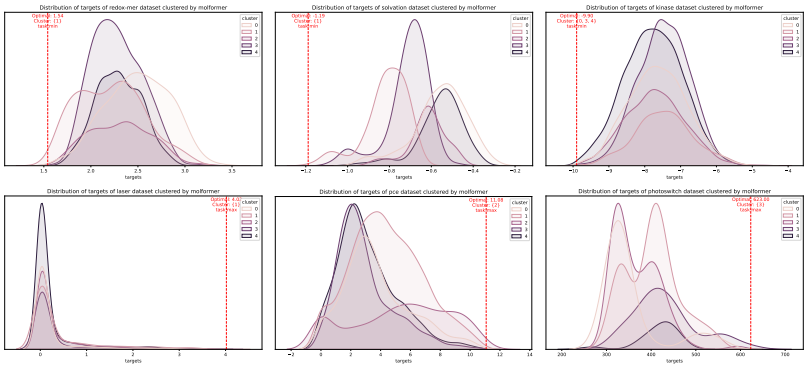


Figure 27: Distribution of property values for K-means clusters on features extracted from Molformer.

F.4.2 LLM-based clustering

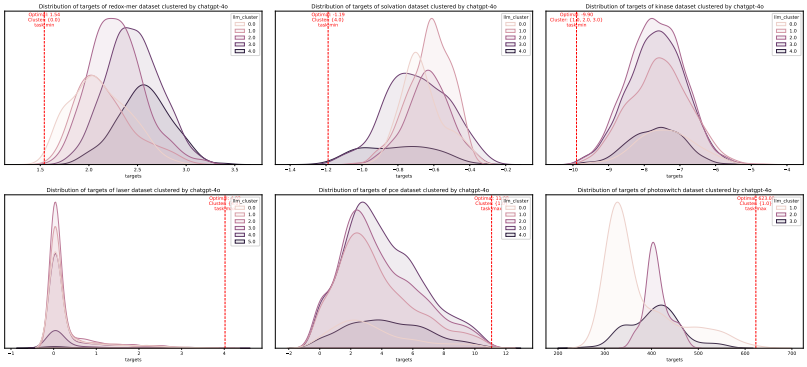


Figure 28: Distribution of property values for GPT-4o-based clustering results.

F.5 Ablation on p-values

In this section, we provide additional details on how varying the p-values impacts our proposed algorithm in terms of the GAP metric, average regret, and prediction time per BO round for each dataset, using both K-means and LLM-based clustering. The results suggest that the LLM-based clustering approach is significantly effective in reducing prediction time while at the same time maintaining (within noise) the GAP and regret performance for Redoxmer and Solvation and improving it for Photovoltaics, reflecting that GPT-4o has informative knowledge on these datasets. This is consistent with its clustering visualization in Fig. 28, where the mean property values for clusters have an obvious difference. K-means clustering is effective for Redoxmer and Solvation in reducing prediction time without degrading the performance.

F.5.1 GAPs' change w.r.t different p-values

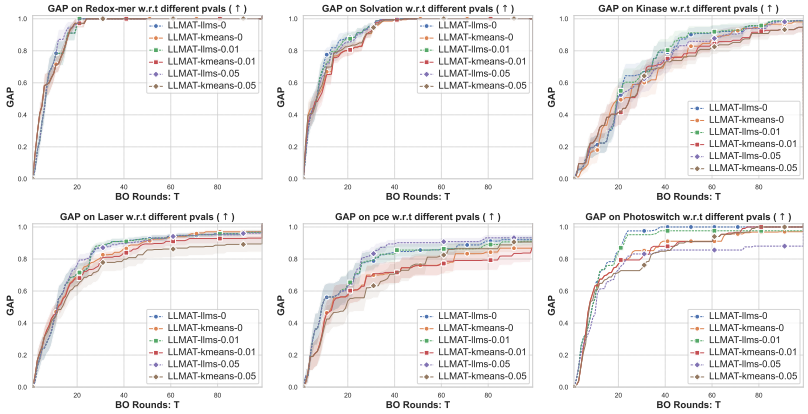


Figure 29: GAPs' change for different clustering approaches and p-values using Molformer.

F.5.2 Regrets change w.r.t different p-values

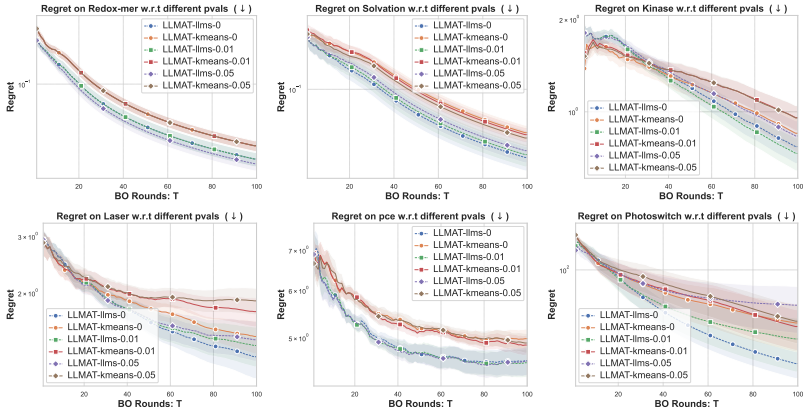


Figure 30: Regrets' change for different clustering approaches and p-values with Molformer.

F.5.3 Effects on reducing prediction time w.r.t different p-values

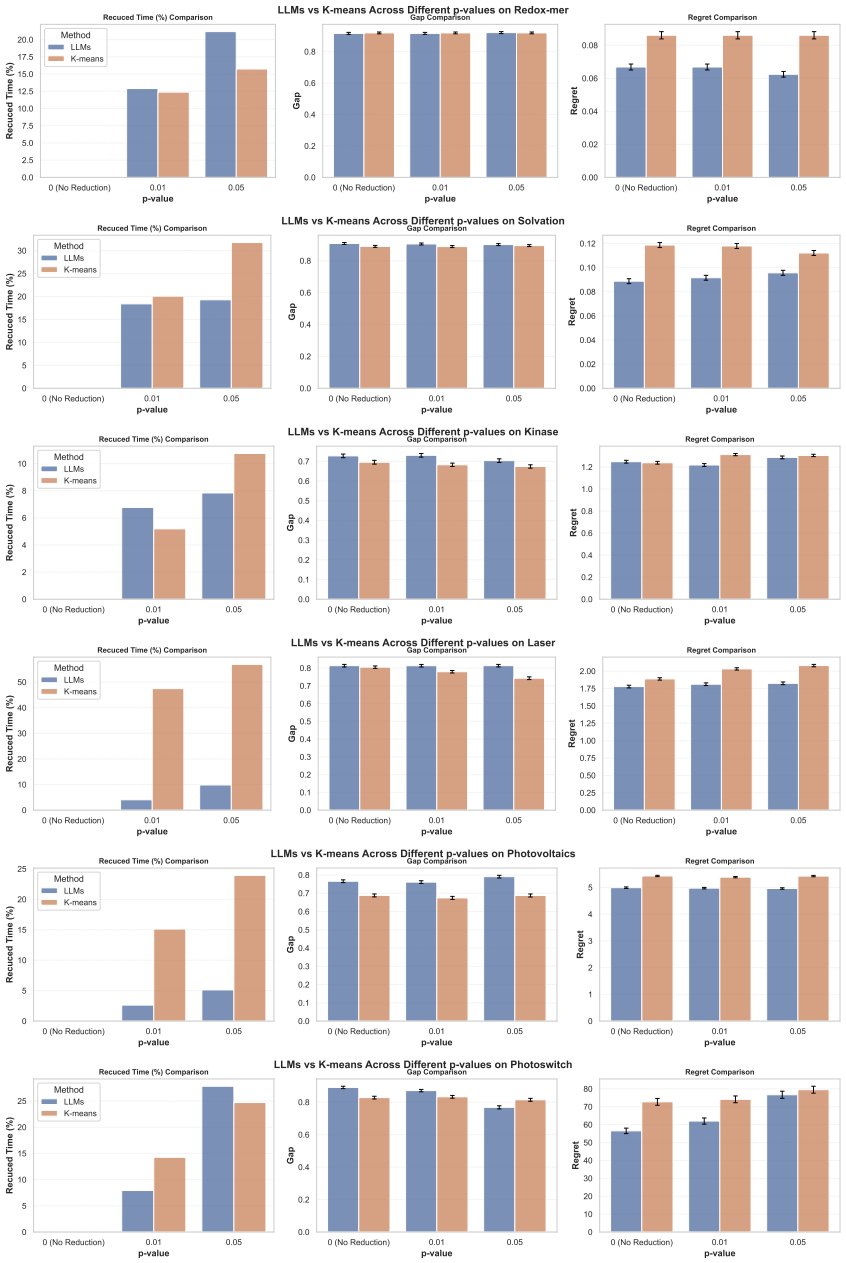


Figure 31: Prediction time reduction percentage, AUC of GAPs, and Regrets across all the datasets for different clustering approaches and p-values.

F.6 Toy Data

F.6.1 Levy-1D Data

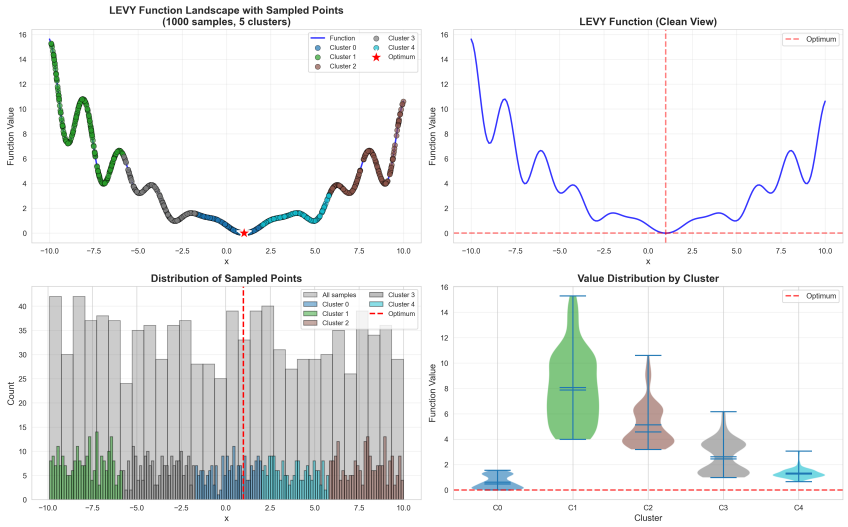


Figure 32: Illustration of 1,000 samples of Levy-1D function with K-means clustering.

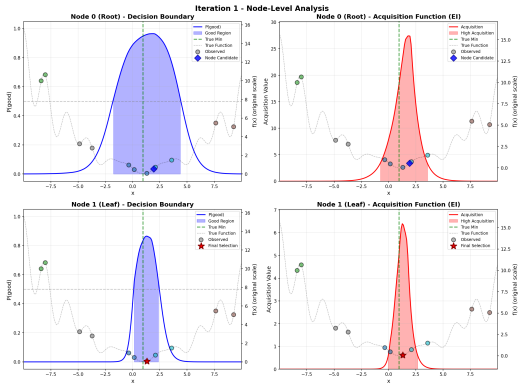


Figure 33: Refined AFs in LLMAT enable more efficient BO (Iteration 1).

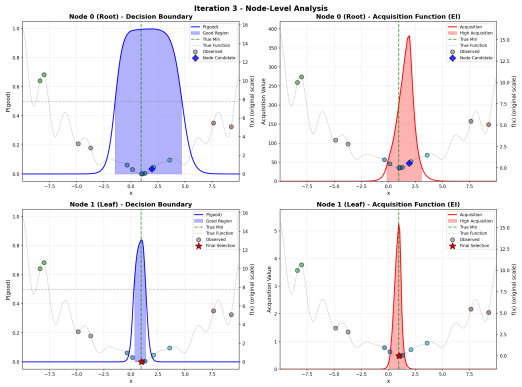


Figure 34: Refined AFs in LLMAT enable more efficient BO (Iteration 3).

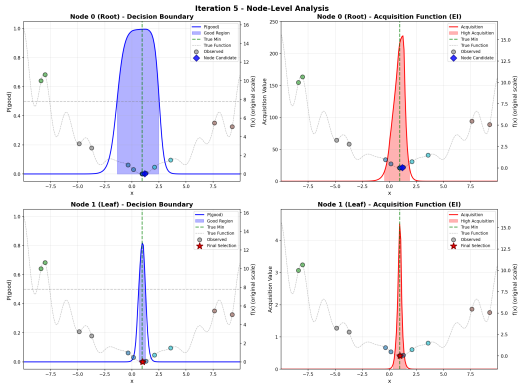


Figure 35: Refined AFs in LLMAT enable more efficient BO (Iteration 5).

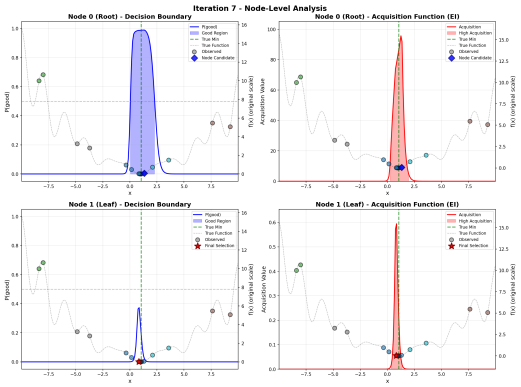


Figure 36: Refined AFs in LLMAT enable more efficient BO (Iteration 7).

F.6.2 Levy-10D Data

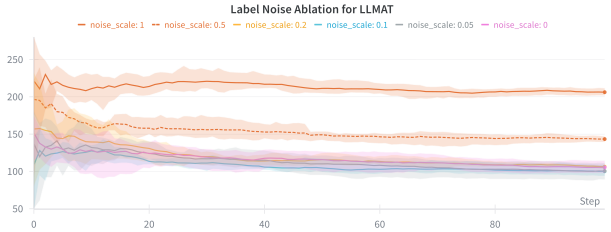


Figure 37: Illustration of label noise's effects on the average regret of LLMAT on the Levy-10D function.

F.7 Ablation of γ

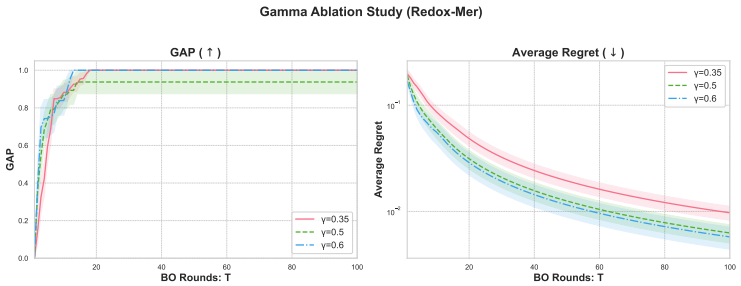


Figure 38: Illustration of LLMAT performance change w.r.t. different γs.

F.8 Additional results for Llama-2-7b

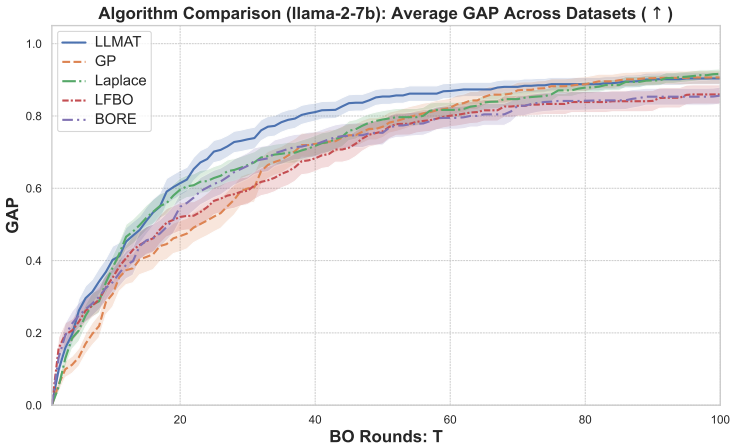


Figure 39: Performance comparison of various algorithms on Llama-2-7b across all datasets.

F.9 Multi-objective Optimization Extension

We consider two objectives on the Redox-mer dataset, redox potential and solvation energy, and compare LLMAT with LAPEFT using the hypervolume and a scalarized objective (Fig. 41). Fig. 40 shows the Pareto fronts obtained with LLM-based and K-means clusters.

Setup. Both objectives (redox potential and solvation energy) are minimized; we negate them so that the problem becomes maximization. LLMAT is extended by scalarization: at every round the objectives of each observation are combined with a linear scalarization with uniform weights $1/k$ ($k = 2$), applied to the raw, unnormalized objective values, and this scalar value replaces y in the original algorithm. The scalar values define the thresholds τ_k and binary labels of the node classifiers, so the tree construction, the local acquisition functions and the UCT-based selection are unchanged, and the candidate with the highest acquisition value is queried. This variant uses fixed features (T5-Chem), 10 initial points, and $T = 100$ rounds. After each query we update the Pareto front of all observations and report its hypervolume, as well as the best scalarized value on the current Pareto front. The hypervolume reference point is inferred with BoTorch's `infer_reference_point` from the Pareto front of the full candidate pool, and the maximal hypervolume is that of this ground-truth front.

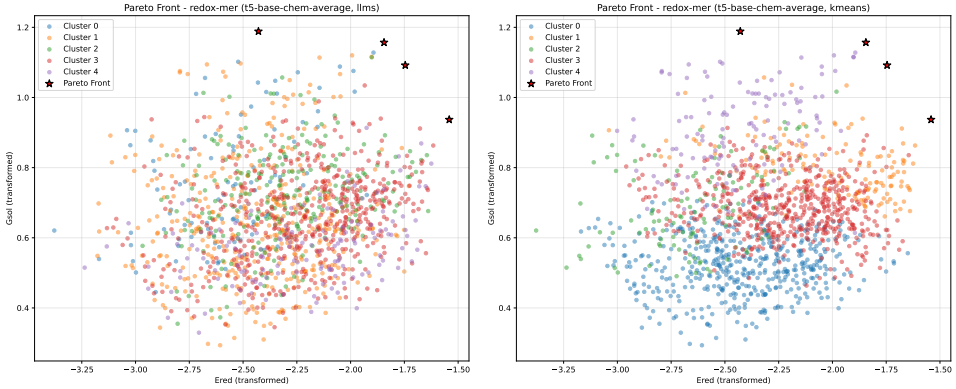


Figure 40: Pareto front with LLM (left) and K-means (right) clusters on Multi-Redox data.

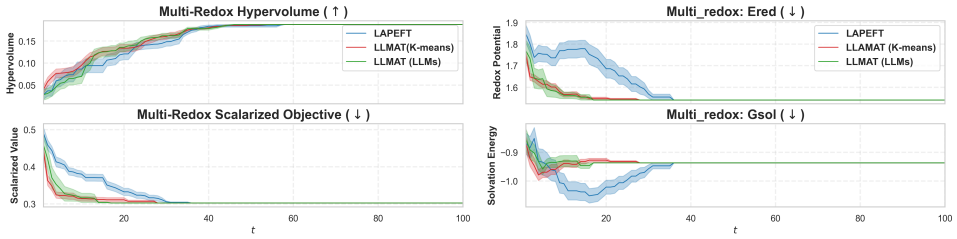


Figure 41: Multi-objective optimization for LAPEFT vs. LLMAT on the Redox-mer dataset: **Upper & Lower Left:** Hypervolume, scalarized objective. **Right:** Redox potential and solvation energy.

F.10 Ablation over different LLM APIs

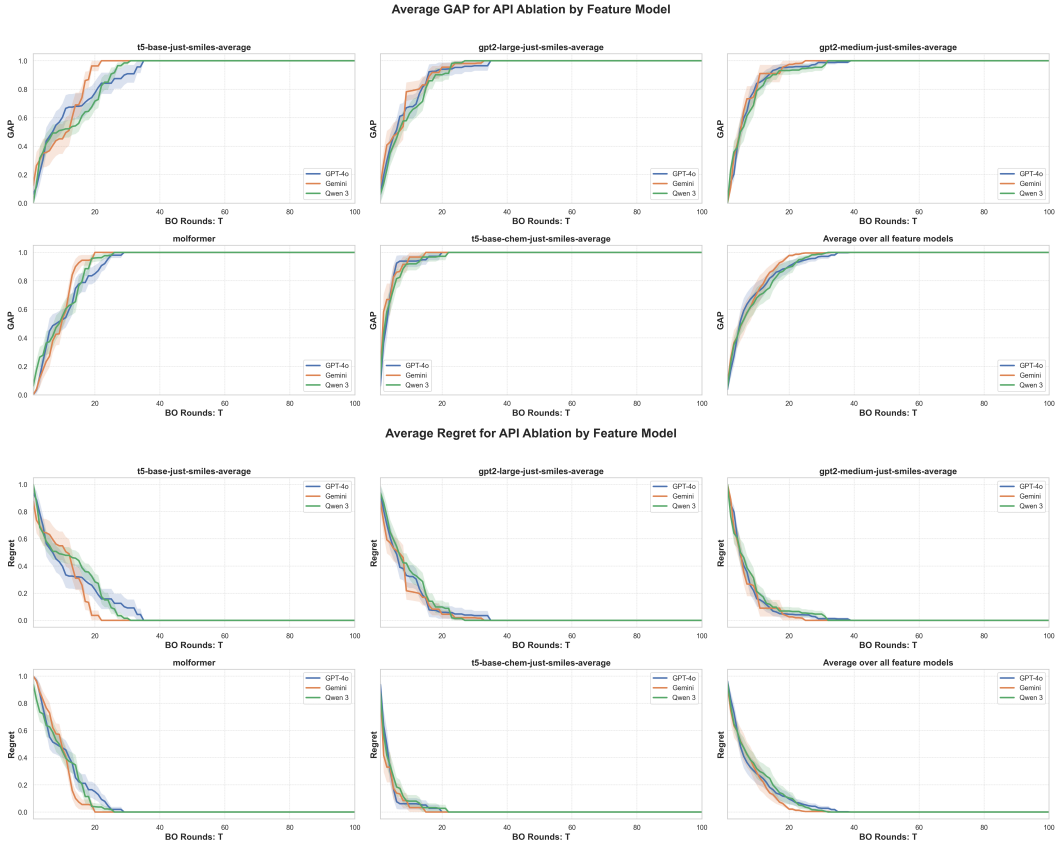


Figure 42: GAP (top) and average regret (bottom) of LLMAT on Redox-mer when LLM-based clustering uses different LLM APIs (GPT-4o, Gemini, Qwen3-Plus), across feature models.

F.11 Fixed GNN Embeddings (GIN)

To verify that LLMAT is not limited to SMILES/text representations, we evaluate it with fixed embeddings from a pretrained GIN [Xu et al., 2019, Hu et al., 2020], without finetuning during BO. We report the AUC of the normalized gap $\mathrm{Gap}_t = \frac{y_t^{\mathrm{best}} - y_0}{y_{\max} - y_0}$ over the whole trajectory (Gap AUC, Sec. F); higher is better. LLMAT achieves the best average Gap AUC (79.11) across the six datasets, with particularly strong gains on Redoxmer, Laser, and Photovoltaics, so its sample-efficiency advantage extends to pretrained graph-based molecular features.

Table 4: Gap AUC (↑) with fixed GIN embeddings (mean ± std). Best per row in bold.

Dataset	LLMAT	GP	Laplace	Random
Redoxmer	**92.3±0.9**	85.4±2.5	75.2±3.1	30.9±6.4
Solvation	83.9±2.8	**85.3±1.1**	76.1±1.5	44.5±4.8
Kinase	77.1±4.7	**78.8±1.8**	67.9±4.4	47.4±6.6
Laser	**74.3±3.4**	47.8±4.8	63.9±3.8	30.2±5.6
Photovoltaics	**68.6±6.4**	55.0±6.6	59.3±6.3	41.1±6.3
Photoswitches	78.5±3.0	**82.4±3.5**	59.2±5.3	34.9±7.5
Average	**79.11**	72.46	66.92	38.14

F.12 Informativeness of Representations

A general-purpose LLM representation is not intrinsically a "weak" prior, so informativeness should be defined with respect to the downstream property. As a task-specific proxy, we fit the same linear probe on each representation and report cross-validated R^2 (Table 5). There is no universal ordering: foundation-model embeddings are generally more predictive on larger, more complex datasets such as Kinase, while Morgan fingerprints perform best on Photoswitches.

Table 5: Linear-probe cross-validated R^2 (↑) for different molecular representations. Best per row in bold.

Dataset	Size	T5-Chem	Fingerprints	MolFormer	T5-Base	GPT2-Med	GPT2-Large	LLaMA-2-7B
Redoxmer	1407	**0.940**	0.931	0.919	0.920	0.892	0.882	0.915
Solvation	1407	**0.856**	0.833	0.845	0.847	0.834	0.841	0.843
Kinase	10449	**0.757**	0.680	0.736	0.708	0.689	0.697	0.744
Laser	10000	0.574	0.598	0.548	0.514	0.566	0.570	**0.629**
PCE	10000	**0.608**	0.595	0.577	0.581	0.549	0.578	0.607
Photoswitch	392	0.441	**0.516**	0.352	0.320	0.303	0.261	0.335

To assess sensitivity to the choice of representation, Table 6 compares each algorithm under its worst-performing representation. LLMAT attains the highest worst-case Gap AUC on five of six datasets and is comparable to Laplace on Laser (56.78 ± 4.96 vs. 57.84 ± 2.72). Averaged over datasets, the worst-case Gap AUC is 72.28 for LLMAT, versus 55.11 for GP and 61.41 for Laplace, so LLMAT is substantially more robust to the representation.

Table 6: Worst-case Gap AUC (↑) over representations; the representation attaining the worst value is in parentheses.

Dataset	GP worst	LLMAT worst	Laplace worst
Redoxmer	70.15±2.94 (GPT2-Med)	**82.09±2.29** (LLaMA-2-7B)	72.86±2.76 (GPT2-Med)
Solvation	56.85±3.37 (GPT2-Med)	**82.72±3.29** (GIN)	66.36±1.38 (LLaMA-2-7B)
Kinase	55.53±4.21 (Fingerprints)	**74.64±3.28** (MolFormer)	50.83±4.07 (Fingerprints)
Laser	42.74±3.10 (GPT2-Med)	56.78±4.96 (Fingerprints)	**57.84±2.72** (Fingerprints)
Photovoltaics	53.90±4.90 (GPT2-Med)	**69.44±3.36** (LLaMA-2-7B)	57.47±3.06 (Fingerprints)
Photoswitches	51.50±6.13 (GPT2-Med)	**68.02±6.22** (LLaMA-2-7B)	63.12±4.10 (GIN)
Average	55.11	**72.28**	61.41

F.13 Morgan Fingerprints vs. Foundation-Model Embeddings

Table 7 summarizes, by visual inspection of the BO trajectories, how foundation-model (LLM) embeddings compare with Morgan fingerprints (FP) for each algorithm. Foundation-model embeddings do not universally outperform Morgan fingerprints, but give clear gains on most datasets. Fingerprints are particularly strong on Photoswitches under LLMAT and GP, consistent with prior observations that standard molecular descriptors can be more than competitive with neural representations on this dataset [Griffiths et al., 2022]. The benefit of foundation-model representations is therefore task- and surrogate-dependent. One possible explanation for the differences across algorithms is that GP and

LLMAT depend directly on the structure induced by the input representation (GP through kernel similarities, LLMAT through classifiers that partition the space into promising and non-promising regions), whereas Laplace applies a learned nonlinear transformation before uncertainty estimation and is thus less sensitive to the original geometry.

Table 7: Qualitative comparison of foundation-model (LLM) embeddings and Morgan fingerprints (FP), from BO trajectories.

Dataset	LLMAT	GP	Laplace
Redoxmer	LLM clearly better	LLM slightly better	Comparable
Solvation	LLM slightly better	LLM slightly better	LLM slightly better
Kinase	LLM better	LLM better	LLM better
Laser	LLM clearly better	LLM better	LLM clearly better
Photovoltaics	LLM better	LLM slightly better	LLM better
Photoswitches	FP clearly better	FP clearly better	LLM slightly better

To quantify these observations, we compute Gap AUC over all BO rounds (Table 8). With the best foundation-model embedding per dataset, LLMAT improves the mean Gap AUC over Morgan fingerprints by 9.7 points and wins on $5/6$ datasets; averaging over all foundation-model representations, the improvement is $+2.6$ points, with the average representation beating fingerprints on $4/6$ datasets. LLMAT also obtains the highest Gap AUC among the three algorithms under all three representation summaries: with Morgan fingerprints it reaches 74.8 (GP 72.0, Laplace 65.3); with the best foundation model per dataset 84.4 (GP 78.7, Laplace 81.2); and averaged over all foundation models 77.4 (GP 68.2, Laplace 72.3). The advantage therefore cannot be attributed solely to more expressive embeddings.

Table 8: Gap AUC with Morgan fingerprints (FP) vs. foundation-model (FM) embeddings. "Best FM" selects the best FM per dataset; "Average FM" averages over all FMs. Δ is the difference to FP.

Algorithm	Morgan FP	Best FM	Δ	Best FM beats FP	Average FM	Δ	Average FM beats FP
LLMAT	74.8	84.4	+9.7	5/6	77.4	+2.6	4/6
GP	72.0	78.7	+6.7	5/6	68.2	-3.8	1/6
Laplace	65.3	81.2	+15.9	6/6	72.3	+7.0	3/6

F.14 Ablation: Unified Classifier vs. Decoupled Partitioning and LFBO

To verify the benefit of coupling partitioning and acquisition in one classifier, we implement a *decoupled* baseline under fixed features. At each node, K-means partitions the samples in the joint (x, y) space and its labels train a routing classifier, while a separate LFBO classifier performs acquisition scoring; features, initialization, BO budget, and the LFBO objective are unchanged. Values in Table 9 are averaged over six fixed feature models per dataset. LLMAT wins on $26/36$ configurations, and among 11 significant paired Wilcoxon comparisons ($p < 0.05$), 10 favor LLMAT. Coupling is thus not uniformly beneficial, but it improves the overall Gap AUC by 2.1 points and is more parsimonious, reusing one classifier for both partitioning and acquisition.

Table 9: Gap AUC (↑) of the unified LLMAT classifier vs. decoupled partitioning + independent LFBO (mean ± std over runs and six fixed feature models). For this paired comparison, GAP is normalized with a baseline y_0 shared by both methods (the worst initial value over all runs of both methods) instead of each run's own y_0, which avoids penalizing runs whose first query already lands near the optimum; values are therefore larger than in tables that use per-run y_0.

Dataset	LLMAT	Decoupled	Gain
Redoxmer	94.6±2.4	89.6±4.4	+5.0
Solvation	92.0±3.2	86.2±4.0	+5.8
Kinase	88.7±3.7	88.2±3.9	+0.4
Laser	82.5±4.8	83.5±2.9	−1.0
Photovoltaics	88.8±3.3	88.2±4.5	+0.6
Photoswitches	88.8±6.0	86.7±5.8	+2.0
Overall	89.2±5.3	87.1±4.5	+2.1

F.15 Ablation: UCT vs. Greedy Selection

The "w/o MCTS" ablation (tree depth 0) removes recursive partitioning, node-local acquisition functions, and UCT-based path selection at once. To isolate the effect of UCT, we add a variant that keeps the learned tree, recursive partitioning, and node-local acquisition functions unchanged, but replaces UCT with greedy selection. Table 10 reports Gap AUC over 15 runs per dataset with MolFormer features. UCT gives a modest improvement over greedy selection. We attribute the small gap to the partitioning mechanism: node classifiers are trained on labels induced by the ranking of property values, so the learned partitions are already aligned with the optimization objective and greedy selection can exploit them effectively, leaving less room for the exploration bonus. UCT nevertheless remains beneficial when partitions are imperfect, by balancing exploration and exploitation.

Table 10: Gap AUC (↑) of LLMAT variants with MolFormer features (15 runs per dataset; mean ± s.e.m. over all 6×15 runs). Gap AUC integrates GAP over rounds $t = 1, \ldots, 100$. "w/o MCTS" uses separate runs of the single-node (LFBO) variant, so it differs slightly from depth 0 in Table 12.

Config	Gap AUC
LLMAT w/ LLM Clustering	**82.36±1.49**
LLMAT w/ K-means Clustering	80.24±1.63
LLMAT Base (w/o Clustering)	80.53±1.58
LLMAT w/ Greedy Selection (UCT→Greedy)	80.29±1.62
LLMAT Base w/o Meta ($\eta = 0$)	79.75±1.76
LLMAT Base w/o MCTS (tree depth $= 0$)	78.39±1.89

F.16 Meta-learning Dynamics at Deeper Nodes

We examine per-node training behavior with and without meta-learning (Table 11). Shallow nodes behave similarly, while meta-learning yields consistently lower loss over much of the training process at deeper, data-scarce nodes, with the clearest improvement at depth 2. At depth 1, where each node holds only about 11 samples, meta-learning reduces the final loss from 0.4269 to 0.4121; at depth 2, with only 7–8 samples per node, the reduction rises to about 14% ($0.3221 \rightarrow 0.2772$). At the root (depth 0) we do not observe an advantage. This supports the intended role of meta-learning: transferring information across nodes is most useful when local adaptation is data-limited, rather than uniformly improving every node.

Table 11: Per-node training statistics with and without meta-learning, by tree depth. n_{pos}/n_{neg}: average numbers of positive/negative labels per node.

Depth	Method	Avg. # samples	Final loss	Avg. n_{pos}	Avg. n_{neg}
0	meta	25.4	0.1871±0.0187	7.9	17.5
0	w/o meta	23.2	0.1439±0.0155	8.5	14.7
1	meta	11.5	0.4121±0.0267	3.4	8.1
1	w/o meta	11.4	0.4269±0.0290	3.5	7.9
2	meta	7.6	0.2772±0.0330	2.3	5.4
2	w/o meta	7.4	0.3221±0.0461	2.2	5.2

F.17 Tree Depth Ablation

With fixed MolFormer features, we vary the maximum depth L from 0 (no partitioning) to 4; the number of nodes is $2^{L+1}-1$, ranging from 1 to 31 (Table 12). Increasing depth does not cause severe degradation: moderate partitioning generally improves performance, and depth 3 improves Gap AUC over depth 0 from 78.12 to 80.08 and reduces Regret@100 from 7.29 to 5.81 (about 20%). Early on, depth 1 achieves the best GAP@25, suggesting shallow partitions help when few observations are available. Going to depth 4 gives no further gain and slightly degrades several metrics, consistent with the trade-off between localization granularity and per-node sample size; the degradation is mild rather than catastrophic. The main cost of depth is computational: runtime grows from 43.9 s (depth 0) to 131.6 s (depth 3) and 171.6 s (depth 4). Differences among depths 1–4 in Gap AUC (78.00–80.08) are small relative to the run-to-run variability (±1.5–1.8), so we do not claim a sharp optimum; a shallow tree ($L = 1$) already attains 79.60 at about half the runtime of depth 3 (69.4 s vs. 131.6 s).

Table 12: Tree-depth ablation with MolFormer features (mean ± s.e.m. over all 6×15 runs). Best per column in bold. Here Gap AUC integrates GAP over rounds $t = 1, \ldots, 99$, so it is about 1 point lower than the $t = 1, \ldots, 100$ integral used in Table 10 for the same configuration; depth 0 uses the depth-0 runs of the tree algorithm rather than the separate LFBO runs of Table 10.

Depth	Gap AUC (↑)	GAP@25	GAP@50	GAP@100	Regret@100 (↓)	Time (s)
0	78.12±1.77	0.7341±0.0334	0.8671±0.0256	**0.9748±0.0055**	7.2938±1.9920	43.9±4.2
1	79.60±1.57	**0.7813±0.0291**	0.9103±0.0163	0.9554±0.0125	6.1819±1.9737	69.4±6.7
2	78.00±1.73	0.7473±0.0334	0.8831±0.0234	0.9630±0.0098	6.9169±2.0112	105.7±10.1
3	**80.08±1.53**	0.7652±0.0318	0.9056±0.0198	0.9725±0.0073	**5.8139±1.7825**	131.6±13.5
4	79.91±1.67	0.7701±0.0314	**0.9129±0.0183**	0.9628±0.0107	6.2854±2.1980	171.6±18.5

F.18 Cost of LLM-based Clustering

One complete clustering pass across the six datasets consumes about 616K input and 102K output tokens (≈718K total), with a summed API latency of about 71 minutes. At current API pricing, the total cost is about $2.59 (GPT-4o), $1.81 (Gemini-2.5-Pro), and $0.47 (Qwen3-Plus). To judge when this overhead is worthwhile, we define the average GAP gain over all BO rounds, in percentage points (pp):

$$\Delta\text{AUC-GAP} = 100 \left[\frac{1}{T} \sum_{t=1}^{T} \text{GAP}_t^{\text{LLM}} - \frac{1}{T} \sum_{t=1}^{T} \text{GAP}_t^{\text{Kmeans}} \right],$$

so $\Delta\text{AUC-GAP} = 2$ means LLMAT with LLM clustering (p=0.01) improves GAP by 2 pp on average across the $T = 100$ rounds over LLMAT with K-means clustering (p=0, no pruning). The largest gains occur on Kinase (+5.77 pp) and PCE (+4.83 pp), while gains are small or negative on Redox-Mer, Solvation, and Photoswitch. The Kinase gain is the largest and not driven by outliers (median +5.08 pp, positive in 9 of 15 seeds), whereas the PCE mean is influenced by a few seeds (median +1.32 pp). With 15 seeds these differences are not statistically significant ($p > 0.3$, paired tests), so they should be read as trends. LLM clustering is thus most justified when it yields a substantial improvement in sample efficiency, particularly when oracle evaluations are expensive; for simpler problems where K-means performs similarly, the overhead may not be warranted.

Table 13: Cost, latency, and benefit of one LLM-clustering pass. N: number of candidates. ΔAUC-GAP compares LLMAT with LLM-based clustering (p=0.01) against LLMAT with K-means clustering (p=0, i.e., no statistical pruning), using MolFormer features, tree depth 3 in code (L=1), and 15 seeds; it therefore reflects both the clustering method and the pruning setting.

Dataset	N	GPT-4o	Gemini	Qwen3	Latency	ΔAUC-GAP (pp)
Redox-Mer	1,407	\$0.103	\$0.076	\$0.018	3.3 min	−0.937
Solvation	1,407	\$0.102	\$0.075	\$0.018	3.3 min	+0.089
Kinase	10,449	\$0.647	\$0.482	\$0.111	22.1 min	+5.773
Laser	10,000	\$1.042	\$0.671	\$0.203	20.8 min	+1.149
PCE	10,000	\$0.670	\$0.485	\$0.118	20.8 min	+4.833
Photoswitch	392	\$0.023	\$0.018	\$0.004	0.8 min	−0.101
Total		\$2.59	\$1.81	\$0.47	71 min	

F.19 Scalability w.r.t. Candidate Pool Size

LLMAT assumes a finite candidate pool and scores all unobserved candidates at the root node in every BO round. Table 14 reports the average per-round runtime (seconds) across T5-Chem and MolFormer features for pool sizes from 392 to about 10K. Prediction cost scales roughly linearly with pool size, as expected from scoring each candidate once at the root: growing the pool from $1{,}407$ to $10{,}000$ ($7.1\times$) increases LLMAT prediction time from 0.097 s to 0.673 s ($6.9\times$) while training time rises only from 0.590 s to 0.925 s, so total per-round time grows from 0.773 s to 1.713 s ($2.2\times$). At the largest pool ($10{,}449$) LLMAT needs about 1.6–1.7 s per round and remains faster overall than GP and Laplace. Scoring a finite pool is shared by pool-based discrete BO methods, and enumerating arbitrarily large chemical spaces remains infeasible; this can be mitigated by pre-clustering or pre-screening to relevant regions, candidate subsampling, batched inference, hierarchical retrieval, and parallel batch BO.

Table 14: Average per-round runtime (s) vs. candidate pool size (T5-Chem and MolFormer features).

Pool size	LLMAT			GP			Laplace		
	Train	Predict	Total	Train	Predict	Total	Train	Predict	Total
392	0.680	0.058	0.816	1.521	0.014	1.548	1.323	0.181	1.516
1,407	0.590	0.097	0.773	1.424	0.035	1.485	1.542	0.597	2.167
10,000	0.925	0.673	1.713	2.091	0.272	2.616	2.836	4.513	7.557
10,449	0.828	0.701	1.636	2.350	0.318	2.944	3.733	6.764	10.801

G Broader Impacts

- **Potential Positive Impacts.** The proposed methods can accelerate molecular discovery by leveraging knowledge from both general LLMs and domain-specific foundation models. They can be combined with generative approaches to propose novel, unseen molecules for drug and material design, offering significant societal benefits: new drugs may save lives, and new materials could contribute to environmental protection.
- **Potential Negative Impacts.** However, the method also carries potential risks. For example, the same capabilities could be misused to create toxic or harmful compounds, and overreliance on AI predictions without rigorous experimental validation could lead to unintended consequences. Responsible use, careful oversight, and appropriate safety measures are therefore critical.

H Declaration of LLMs Usage

We use LLMs as grammar correction tools and for improving written text flow, combined with human proofreading. No LLM was used for the idea or primary text of the paper. However, as this

paper investigates how LLMs can help inform acquisition functions for BO over molecular datasets, we incorporate LLMs as algorithmic modules, where we also prompt ChatGPT to generate proper LLM-clustering prompts. LLM-based coding assistants were also used to help write the plotting code for the figures; the authors reviewed the code and the resulting figures.

I Theoretical Results

A complete regret guarantee for LLMAT would require jointly analyzing adaptive tree construction, node selection, local sample allocation, and acquisition learning. We therefore do not claim a general convergence theorem for the full algorithm. Instead, we first give a density-ratio and divergence interpretation of the node-wise LFBO classifier and show why recursive local classifiers can outperform a global one at the population level. We then provide an approximation–estimation–routing analysis under the PI special case, followed by a cumulative-regret decomposition under the probabilistic least-squares setting of BORE++ [Oliveira et al., 2022].

I.1 Density-ratio and divergence interpretation of local classifiers

We consider the PI case, $u(y;\tau) = \mathbf{1}(y > \tau)$, where the connection is most transparent. For a tree node Ω_k, let $\mathbf{D}_t^k := \mathbf{D}_t \cap \Omega_k = \{(x_i, y_i) \in \mathbf{D}_t \mid x_i \in \Omega_k\}$ denote the observations assigned to this node. Following Sec. 3.1, define

$$g_k(x) = p(x \mid y > \tau_k, \mathbf{D}_t^k), \quad l_k(x) = p(x \mid y \le \tau_k, \mathbf{D}_t^k), \quad \gamma_k = p(y > \tau_k \mid \mathbf{D}_t^k),$$

and let the marginal distribution of all samples in the node be $p_k(x) = p(x \mid \mathbf{D}_t^k) = \gamma_k g_k(x) + (1 - \gamma_k) l_k(x)$. Since $\mathbb{E}_{p_k}[u(y;\tau_k) \log \pi_\theta(x)] = \gamma_k \mathbb{E}_{g_k}[\log \pi_\theta(x)]$, the LFBO loss [Song et al., 2022] in this node becomes

$$\mathcal{L}_{\mathbf{D}_t^k}^{\tau_k}(\theta) = -\gamma_k \, \mathbb{E}_{g_k}\big[\log \pi_\theta(x)\big] - \mathbb{E}_{p_k}\big[\log(1 - \pi_\theta(x))\big]. \tag{4}$$

Proposition I.1 (Density-ratio and divergence interpretation)**.** *Let $w_k = \gamma_k/(1+\gamma_k)$ and $M_k = w_k g_k + (1 - w_k) p_k$. Over all measurable classifiers, the minimizer of Eq. equation 4 is*

$$\pi_k^\star(x) = \frac{\gamma_k g_k(x)}{\gamma_k g_k(x) + p_k(x)}, \qquad \frac{\pi_k^\star(x)}{1 - \pi_k^\star(x)} = \gamma_k \frac{g_k(x)}{p_k(x)} = p(y > \tau_k \mid x, \mathbf{D}_t^k). \tag{5}$$

Moreover, the optimal loss satisfies

$$-\mathcal{L}_{\mathbf{D}_t^k}^{\tau_k}(\theta^\star) = c(\gamma_k) + (1 + \gamma_k) \, \mathrm{JS}_{w_k}(g_k \,\|\, p_k), \tag{6}$$

where $\mathrm{JS}_w(g\|p) := w D_{\mathrm{KL}}(g\|M) + (1-w) D_{\mathrm{KL}}(p\|M)$ is the w-weighted Jensen–Shannon divergence and $c(\gamma_k) = \gamma_k \log \gamma_k - (1+\gamma_k)\log(1+\gamma_k)$ depends only on γ_k.

Proof. Since $(1+\gamma_k)w_k = \gamma_k$ and $(1+\gamma_k)(1-w_k) = 1$, Eq. equation 4 can be rewritten as

$$\mathcal{L}_{\mathbf{D}_t^k}^{\tau_k}(\theta) = -(1+\gamma_k)\Big[w_k \, \mathbb{E}_{g_k} \log \pi_\theta(x) + (1 - w_k) \, \mathbb{E}_{p_k} \log(1 - \pi_\theta(x))\Big].$$

Pointwise maximization of $w_k g_k(x) \log \pi + (1 - w_k) p_k(x) \log(1 - \pi)$ over $\pi \in (0,1)$ gives $\pi_k^\star = w_k g_k / M_k$, which equals Eq. equation 5; its odds equal $\gamma_k g_k / p_k$, which is $p(y > \tau_k \mid x, \mathbf{D}_t^k)$ by Bayes' rule. Substituting $\pi_k^\star = w_k g_k / M_k$ and $1 - \pi_k^\star = (1 - w_k) p_k / M_k$ yields

$$-\frac{\mathcal{L}_{\mathbf{D}_t^k}^{\tau_k}(\theta^\star)}{1+\gamma_k} = w_k \log w_k + (1 - w_k) \log(1 - w_k) + w_k D_{\mathrm{KL}}(g_k \| M_k) + (1 - w_k) D_{\mathrm{KL}}(p_k \| M_k).$$

The last two terms are exactly $\mathrm{JS}_{w_k}(g_k \| p_k)$, and multiplying the first two terms by $1 + \gamma_k$ gives $c(\gamma_k)$. □

Proposition I.1 shows that the node-wise classifier is not an arbitrary discriminator: at the population optimum, **its odds exactly recover the local PI acquisition function**. Equivalently, it measures how strongly promising samples are enriched relative to the distribution of all samples currently contained in the node. Recursive partitioning uses this density-ratio signal to progressively restrict the search to regions with a higher probability of improvement. In the divergence view, the LFBO

classifier distinguishes the **promising-sample distribution** $p(x \mid y > \tau_k, \mathbf{D}_t^k)$ from the **distribution of all samples in the current node** $p(x \mid \mathbf{D}_t^k)$, and its optimal discriminative gain is their weighted Jensen–Shannon divergence. Unlike GANs, LLMAT does not adversarially minimize this divergence; instead, it exploits the corresponding density-ratio signal to identify promising regions and recursively localize acquisition estimation.

I.2 Why recursive local classifiers can outperform a global model

Fix a partition $\{\Omega_k\}_{k=1}^K$ of the candidate space and a target acquisition (i.e., fixed utility and thresholds). Let $P(\Omega_k) := p(x \in \Omega_k)$ denote the probability mass of region Ω_k under the input distribution, and define the population LFBO loss conditioned on that region as

$$\mathcal{L}_{\Omega_k}(S) = -\mathbb{E}_{(x,y)|x\in\Omega_k}\Big[u(y;\tau)f'(S(x)) - f^*\big(f'(S(x))\big)\Big],$$

where f is the convex generator of the variational f-divergence objective in LFBO [Song et al., 2022] and f^* is its convex conjugate.

Proposition I.2 (Localization never increases the best-in-class population loss)**.** *For any model class $\mathcal{F}$,*

$$\sum_{k=1}^{K} P(\Omega_k) \inf_{S_k\in\mathcal{F}} \mathcal{L}_{\Omega_k}(S_k) \;\le\; \inf_{S\in\mathcal{F}} \sum_{k=1}^{K} P(\Omega_k)\mathcal{L}_{\Omega_k}(S). \tag{7}$$

The inequality is strict if and only if no single $S \in \mathcal{F}$ simultaneously attains (or approaches) $\inf_{S_k\in\mathcal{F}} \mathcal{L}_{\Omega_k}(S_k)$ for every k with $P(\Omega_k) > 0$.

Proof. The left-hand side is the value of the same objective when the single function S is replaced by K independently chosen functions $S_1, \ldots, S_K \in \mathcal{F}$. Choosing $S_k = S$ for all k recovers any feasible point of the right-hand side, so the left-hand side is no larger. Equality holds exactly when the region-wise infima can be achieved by a common S. □

The left-hand side of Eq. equation 7 is the population loss of a separate estimator in each region, whereas the right-hand side is that of a single global estimator. Since the Bayes-optimal loss is identical on both sides, Proposition I.2 equivalently states that, for a fixed partition and target acquisition, localization cannot increase the best-in-class excess risk. Note that if $\mathcal{F}$ contains all measurable functions, equality holds, since the LFBO optimum is attained pointwise. The benefit of localization therefore stems from the limited capacity of practical classifiers: when the acquisition landscape is heterogeneous or multimodal, a global model must represent all modes simultaneously, whereas each local model only needs to approximate the acquisition structure within a restricted region. In LLMAT, each node additionally uses its own threshold τ_k, so the local target itself adapts to the region; the comparison above isolates the approximation effect for a fixed target.

This does not imply that deeper localization always improves finite-sample estimation, because smaller regions contain fewer observations and therefore incur higher estimation variance. LLMAT controls this trade-off through the maximal tree depth L and the minimal leaf sample size m (Algo. 1), which is consistent with the tree-depth ablation in Fig. 5. Sec. I.3 makes this approximation–estimation trade-off explicit, together with the additional routing error introduced by partitioning.

I.3 Approximation–estimation–routing decomposition

Let $x^\star \in \arg\max_{x\in\Omega} f(x)$ and $\Delta_f = \max_x f(x) - \min_x f(x) < \infty$. We make the following assumptions.

Assumption I.3. Observations satisfy $y = f(x) + \varepsilon$, where the noise CDF F_ε is strictly increasing.

Assumption I.4. Each node k uses a fixed threshold τ_k, with population PI classifier $\pi_k^\star(x) = p(y > \tau_k \mid x) = 1 - F_\varepsilon(\tau_k - f(x))$. Hence $\pi_k^\star$ and f induce the same ranking within Ω_k. We assume the corresponding inverse link is L_ε-Lipschitz over the relevant probability range.

Assumption I.5. At round t, the tree selects Ω_{k_t} based on the BO history, and $x_{t+1} \in \arg\max_{x\in\Omega_{k_t}} \hat{\pi}_{t,k_t}(x)$.

Assumption I.6 (Low-dimensional local structure)**.** Each local problem has effective dimension d_k, and the adaptive samples assigned to node k sufficiently cover its relevant feature directions. We use the schematic scaling $e_{t,k} = \widetilde{O}\big(\sqrt{d_k/n_{t,k}}\big)$, where $n_{t,k}$ is the number of samples available to node k. This is a structural assumption used to expose the approximation–estimation trade-off, not an i.i.d. generalization claim.

Define the correct-routing event $E_t = \{x^\star \in \Omega_{k_t}\}$, its failure probability $q_t = p(E_t^c)$, and decompose the local classifier error as

$$\big\|\hat{\pi}_{t,k} - \pi_k^\star\big\|_{\infty,\Omega_k} \le a_k + e_{t,k},$$

where a_k and $e_{t,k}$ are the approximation and estimation errors, respectively.

On E_t, since x_{t+1} maximizes $\hat{\pi}_{t,k_t}$, we have $\pi^\star_{k_t}(x^\star) - \pi^\star_{k_t}(x_{t+1}) \le 2(a_{k_t} + e_{t,k_t})$, and therefore $r_{t+1} \le 2L_\varepsilon(a_{k_t} + e_{t,k_t})$. If routing fails, $r_{t+1} \le \Delta_f$. Hence

$$\mathbb{E}[r_{t+1}] \le 2L_\varepsilon(1-q_t)\,\mathbb{E}\left[a_{k_t} + e_{t,k_t} \mid E_t\right] + q_t\Delta_f. \tag{8}$$

For a single global classifier with approximation and estimation errors a_0 and $e_{t,0}$, $\mathbb{E}\big[r^{(0)}_{t+1}\big] \le 2L_\varepsilon\,\mathbb{E}[a_0 + e_{t,0}]$. Therefore, localization has a tighter upper bound whenever

$$(1-q_t)\,\mathbb{E}\left[a_{k_t} + e_{t,k_t} \mid E_t\right] + \frac{\Delta_f}{2L_\varepsilon}\,q_t < \mathbb{E}[a_0 + e_{t,0}]. \tag{9}$$

This gives the basic boundary: localization can reduce approximation and statistical complexity, but it also fragments samples and introduces routing error.

Under Assumption I.6, and similarly writing $e_{t,0} = \widetilde{O}\big(\sqrt{d_0/n_t}\big)$ for the global classifier, the boundary becomes schematically

$$(1-q_t)\,\mathbb{E}\left[a_{k_t} + \sqrt{\frac{d_{k_t}}{n_{t,k_t}}}\;\middle|\; E_t\right] + \frac{\Delta_f}{2L_\varepsilon}\,q_t < a_0 + \sqrt{\frac{d_0}{n_t}}. \tag{10}$$

Ignoring approximation and routing errors, localization improves the estimation term when $d_k/n_{t,k} < d_0/n_t$. Letting $\rho_{t,k} = n_{t,k}/n_t$, this is equivalent to $d_k < \rho_{t,k}\,d_0$. Thus, the reduction in effective dimension must be large enough to compensate for sample fragmentation. Recursive localization is therefore most advantageous when the global problem is heterogeneous or piecewise low-dimensional, so that promising local regions satisfy $d_k \ll d_0$ while still receiving sufficient samples.

I.4 Cumulative-regret decomposition

Because BO observations are adaptively selected, Assumption I.6 should not be interpreted as an ordinary i.i.d. generalization bound. For cumulative regret, we further specialize to the adaptive probabilistic least-squares (PLS)/RKHS setting analyzed in BORE++ [Oliveira et al., 2022].

Assumption I.7. For each visited node k, $\pi_k^\star$ belongs to an RKHS $\mathcal{H}_k$ with bounded norm, and the local binary residuals form a conditionally sub-Gaussian martingale difference sequence. The local query and node-assignment decisions are predictable with respect to the BO filtration.

Applying the adaptive confidence result of Oliveira et al. [2022, Theorem 1] node-wise gives $\big|\hat{\pi}_{t,k}(x) - \pi_k^\star(x)\big| \le \beta_{t,k}\sigma_{t,k}(x)$ with high probability. Following the BORE regret argument [Oliveira et al., 2022, Theorem 2], on correctly routed rounds, $r_{t+1} \le L_\varepsilon\beta_{t,k_t}\big[\sigma_{t,k_t}(x^\star) + \sigma_{t,k_t}(x_{t+1})\big]$. Therefore,

$$R_T \le L_\varepsilon \sum_{t=0}^{T-1} \beta_{t,k_t}\big[\sigma_{t,k_t}(x^\star) + \sigma_{t,k_t}(x_{t+1})\big]\mathbf{1}_{E_t} + \Delta_f \sum_{t=0}^{T-1} \mathbf{1}_{E_t^c}. \tag{11}$$

Thus, sublinear cumulative regret requires both the aggregate local uncertainty and the number of routing failures to be sublinear in T. Low-complexity local models can reduce the former when each promising region receives sufficient samples.

Importantly, LLMAT, like the original BORE algorithm, maximizes $\hat{\pi}$ rather than a classifier UCB. Hence the uncertainty term at $x^\star$ remains, exactly as in BORE [Oliveira et al., 2022, Theorem 2]. BORE++ removes this term by maximizing a classifier UCB, yielding the stronger no-regret result in Oliveira et al. [2022, Theorem 3].

We therefore present this as a **regret decomposition and boundary characterization**, rather than a general no-regret theorem. It identifies three conditions under which recursive localization can be beneficial: **lower local approximation error or effective dimension, sufficient local sampling, and sufficiently accurate routing**.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the scope and contributions of LLMAT, including localized acquisition learning, meta-learned classifiers, optional LLM-guided clustering, and empirical validation on molecular BO benchmarks.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations were discussed in Sec. 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: This paper is not theory-oriented.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The experimental details are presented in Sec.E.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: To support reproducibility, our code is publicly available at `https://github.com/livreQ/LLMAT` (see Sec. E).

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: The experimental details are presented in Sec.E.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: To ensure robustness, performance curves and tables report averages over 15 random seeds, with error bands or $\pm$ values showing variability across runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The details are presented in Sec.E

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses publicly available benchmark datasets and pretrained models, does not involve human subjects or private data, and we are not aware of any violation of the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Broader impacts, including both potential benefits for molecular discovery and potential misuse risks, are discussed in Appendix G

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This work does not release data or models with a high risk of misuse. We only release code for reproducing the proposed method and use publicly available benchmark datasets under their stated terms of use.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the original sources for all benchmark datasets, pretrained models, and software packages used in this work. These assets are publicly available, and our use follows their stated terms of use; license information is provided where available from the original repositories or dataset pages. Llama-2-7b is used under Meta's Llama 2 Community License.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: Our code, with instructions for installation and running, is publicly available at `https://github.com/livreQ/LLMAT`. No new dataset or human-subject asset is introduced; the experiments use publicly available benchmark datasets with citations to the original sources.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: Not related.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

Justification: Not related.

Guidelines:

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

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLM use is described in Appendix H .

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.