Towards AutoAI: Optimizing a Machine Learning System with Black-box and Differentiable Components

Zhiliang Chen [1 2] **Chuan Sheng Foo** [2 3] **Bryan Kian Hsiang Low** [1]

Abstract

Machine learning (ML) models in the real world typically do not exist in isolation. They are usually part of a complex system (e.g., healthcare systems, self-driving cars) containing multiple ML and *black-box* components. The problem of optimizing such systems, which we refer to as *automated AI* (AutoAI), requires us to *jointly* train all ML components together and presents a significant challenge because the number of system parameters is extremely high and the system has no analytical form. To circumvent this, we introduce a novel algorithm called A-BAD-BO which uses each ML component's local loss as an auxiliary indicator for system performance. A-BAD-BO uses *Bayesian optimization* (BO) to optimize the local loss configuration of a system in a smaller dimensional space and exploits the differentiable structure of ML components to recover optimal system parameters from the optimized configuration. We show A-BAD-BO converges to optimal system parameters by showing that it is *asymptotically no regret*. We use A-BAD-BO to optimize several synthetic and real-world complex systems, including a prompt engineering pipeline for *large language models* containing millions of system parameters. Our results demonstrate that A-BAD-BO yields better system optimality than gradient-driven baselines and is more sample-efficient than pure BO algorithms.

1. Introduction

Modern real-world systems that perform complex tasks in areas such as healthcare, robotics (Karkus et al., 2019), or even large language model (LLM) pipelines (Zhang et al., 2023; Lin et al., 2024), are typically composed of multiple interacting components, only some of which have accessible parameters. To deploy a system effectively, we need to optimize the system performance by *jointly* training system components, a problem which we refer to as *AutoAI*.

Our work here considers a general system comprising both (a) *differentiable ML components* (Bishop & Nasrabadi, 2006; Bruna et al., 2013; Lecun et al., 1998) containing some accessible trainable parameters, and (b) *black-box non-ML components* (i.e., without any analytical form nor accessible parameters) (Hase et al., 2018) that are typically too sophisticated or costly to be modeled and replaced by a differentiable function with high fidelity (e.g., ChatGPT used in our experiments). These components are then arranged based on some structure (e.g., a directed acyclic graph) predefined by a practitioner to accomplish a system-level task. For example, a healthcare system (Fig. 1) contains multiple ML components (enclosed in blue) making predictions on the health of different organs of a patient that are reviewed by two human doctors (black-box components represented by doctor icons). The doctors' assessments are then combined by an aggregation ML model to yield a diabetes risk score. The doctors are considered black-box components as their decisions cannot be characterized in an analytical form, while the models used to predict the organs' health and aggregate the doctors' decisions are differentiable ML components with trainable parameters. The system performance is then evaluated by its accuracy in predicting the diabetes risk score of any patient.

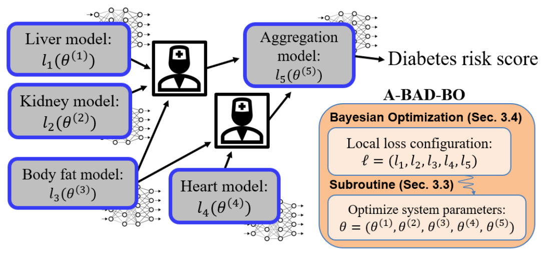

Liver model: $l_1(\theta^{(1)})$
Kidney model: $l_2(\theta^{(2)})$
Body fat model: $l_3(\theta^{(3)})$
Heart model: $l_4(\theta^{(4)})$
Aggregation model: $l_5(\theta^{(5)})$
Diabetes risk score
A-BAD-BO
Bayesian Optimization (Sec. 3.4)
Local loss configuration: $\ell = (l_1, l_2, l_3, l_4, l_5)$
Subroutine (Sec. 3.3)
Optimize system parameters: $\theta = (\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \theta^{(4)}, \theta^{(5)})$

Figure 1. A healthcare system processes a patient's health data with differentiable ML components (enclosed in blue) and black-box components (doctor icons) to produce a diabetes risk score. We propose A-BAD-BO (orange) to optimize the system. The notations are described in Sec. 2.2.

[1]Department of Computer Science, National University of Singapore [2]Institute for Infocomm Research, A*STAR, Singapore [3]Centre for Frontier AI Research, A*STAR, Singapore. Correspondence to: Zhiliang Chen <chenzhiliang@u.nus.edu>.

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

Given *fixed* black-box components and some input data, adjusting the trainable parameters of each ML component would influence the system output directly. So, optimizing the system performance would require *jointly* training all ML components, which is significantly challenging in practice as the system cannot be characterized analytically (due to the presence of black-box components) and has an extremely large number of trainable parameters residing in ML components.

If no black-box components exist, an obvious approach is to view the system as a large neural network and learn its parameters via backpropagation to optimize its performance. Otherwise, the system lacks an analytical form which prevents the use of gradient-based techniques. To circumvent this, a typical industry practice, which we coin as a *local approach*, is to train each differentiable ML component independently with a separate dataset using model-dependent gradient-based techniques (Lecun et al., 1998; Srivastava et al., 2014; Vaswani et al., 2017). For example, a practitioner optimizing the system in Fig. 1 would use a separate dataset to train each ML component for predicting each organ's health independently. However, training every ML component independently does not explicitly consider the system performance nor the black-box components' behavior in the optimization process. So, global system optimality is not guaranteed by locally optimizing its differentiable ML components independently. We have provided rigorous explanations and examples of this phenomenon in App. B.1.

On the other hand, a *global approach* treats the entire system as a black-box function and views the system optimization problem as a *black-box optimization problem* (Jones et al., 1998) w.r.t. system parameters (in the ML components). *Bayesian optimization* (BO) (Garnett, 2023; Dai et al., 2023a) has become a popular framework to optimize such black-box functions in many application domains such as drug design (Pyzer-Knapp, 2018), experimental design (Dai et al., 2023b), hyperparameter tuning (Snoek et al., 2012), among others. Unfortunately, BO performs poorly in high-dimensional problems (Bull, 2011; Frazier, 2018) and ML components in modern systems contain an *extremely large* number of parameters (He et al., 2016; Sanh et al., 2019), invalidating the use of BO to optimize the system parameters directly.

While more advanced variants of BO (Moriconi et al., 2020; Eriksson & Jankowiak, 2021) have been developed to tackle higher-dimensional problems, they have either failed to scale to the dimensionality in our systems here or make assumptions of the underlying intrinsic dimensions (Wang et al., 2016) which may not hold in our setting. Other works have used BO for function networks (Astudillo & Frazier, 2021) but assumed each component can be remodeled differently, whereas our setting involves fixed ML components and only allows adjustment of parameters. Hence, despite incorporating the system performance explicitly into the optimization process, global approaches do not exploit the differentiable structure of ML components and cannot handle the extreme dimensionality of parameters in our systems.

To address the system optimization problem, our work here attains the best of both worlds (i.e., the local and global approaches) by *simultaneously* leveraging the differentiable structure of *local* ML components and incorporating the *global* system performance into the optimization process. To do so, we first identify a novel perspective: The *local loss* of each ML component (w.r.t. a local dataset) serves as an auxiliary indicator of the system performance. This perspective allows us to maintain a global approach and use BO over a much smaller-dimensional space to search for promising *local loss configurations* for the system. Then, we exploit the differentiable structure of ML components to efficiently recover the optimal system parameters from these promising local loss configurations, hence preserving the benefits of local approaches. To our best knowledge, our work in this paper is the first to adopt such a two-pronged approach to optimize a complex ML system with black-box and differentiable ML components. The concrete contributions of our work here can be summarized as follows:

- We introduce AutoAI, a class of problems involving the optimization of complex systems with multiple ML and black-box components (Sec. 2.2).
- We show how the system optimization problem can interestingly be reparameterized as a bilevel optimization problem (Sec. 3.1) capturing the relationship between the local and global system performance. Then, we present a novel **A**lgorithm to optimize a complex ML system with **B**lack-box **A**nd **D**ifferentiable ML components using **BO** called **A-BAD-BO**, which solves this problem efficiently via BO over a lower-dimensional *local loss space* at the outer level to optimize the system's local loss configuration (Sec 3.4). At the inner level, we devise a subroutine (Sec. 3.3) that exploits the differentiable structure of ML components to recover the optimal system parameters from the optimized local loss configuration.
- By incorporating our subroutine's estimation error as observation noise under the BO framework, we provide a theoretical analysis of A-BAD-BO's convergence rate via its *cumulative regret* and show that our iterative use of the global and local approach allows us to eventually converge to the optimal system parameters (Sec. 4).
- We use A-BAD-BO to optimize a variety of synthetic and real-world systems found in AutoAI problems, including a prompt engineering pipeline (Zhou et al., 2022) for LLMs containing millions of trainable parameters, and show that A-BAD-BO achieves better optimality than local approaches and is more sample-efficient than global approaches (Sec. 6).

2. Preliminaries

2.1. Bayesian optimization

Since our algorithm relies partially on BO, we provide an outline of how the conventional BO algorithm can be used to optimize a black-box objective function. Consider an unknown objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$ over the space of inputs $\ell \in \mathbb{R}^n$. The goal is to find the minimizer $\ell^* \triangleq \text{argmin}_\ell f(\ell)$. In each iteration $t = 1, 2, \ldots, T$ of BO, a selected input ℓ_t is queried to obtain a *noisy* observation $\tilde{y}_t \triangleq f(\ell_t) + \epsilon_t$ with a sub-Gaussian noise ϵ_t (e.g., Gaussian or bounded noise) to form the sample $(\ell_t, \tilde{y}_t)$. Consistent with the work of Chowdhury & Gopalan (2017), we model f as a realization of a *Gaussian process* (GP) (Williams & Rasmussen, 2006) that is fully specified by its *prior* mean $\mu(\ell)$ and covariance $\kappa(\ell, \ell')$ for all $\ell, \ell' \in \mathbb{R}^n$ where κ is a *kernel* function chosen to characterize the correlation of the observations between any two inputs ℓ and ℓ'; a common choice is the *squared exponential* (SE) kernel $\kappa(\ell, \ell') \triangleq \exp(-\|\ell - \ell'\|_2^2/(2m^2))$ with a *length-scale* hyperparameter m that can be learned via maximum likelihood estimation. Given a column vector $\boldsymbol{y}_t \triangleq [\tilde{y}_\tau]^\top_{\tau=1,\ldots,t}$ of noisy observations at t previous inputs $\ell_1, \ldots, \ell_t$, the posterior belief of f at any new input ℓ' is a Gaussian distribution with the following *posterior* mean and variance:

$$\begin{aligned} \mu_t(\ell') &\triangleq \kappa_t^\top(\ell')(K_t + \lambda I)^{-1}\boldsymbol{y}_t \\ \sigma_t(\ell') &\triangleq \kappa(\ell', \ell') - \kappa_t^\top(\ell')(K_t + \lambda I)^{-1}\kappa_t(\ell') \end{aligned} \tag{1}$$

where $\kappa_t(\ell') \triangleq [\kappa(\ell', \ell_\tau)]^\top_{\tau=1,\ldots,t}$ is a column vector, $K_t \triangleq [\kappa(\ell_\tau, \ell_{\tau'})]_{\tau,\tau' \in 1,\ldots,t}$ is a $t \times t$ covariance matrix, and $\lambda > 0$ is viewed as a free hyperparameter dependent on the problem setting (Chowdhury & Gopalan, 2017). Using (1), the BO algorithm selects the next input query ℓ_{t+1} by optimizing an *acquisition function*, such as minimizing the *lower confidence bound* (LCB) acquisition function (Srinivas et al., 2010): $\ell_{t+1} = \text{argmin}_\ell\, \mu_t(\ell) - \beta_{t+1}\sigma_t(\ell)$ with an exploration parameter β_{t+1}. The cumulative regret (for T BO iterations w.r.t. a minimization problem) $R_T \triangleq \sum_{t=1}^T [f(\ell_t) - f(\ell^*)]$ is typically used to assess the performance of a BO algorithm theoretically (Chowdhury & Gopalan, 2017; Tay et al., 2023) where $f(\ell^*)$ is the true function minimum. A lower cumulative regret indicates a faster convergence rate of the BO algorithm. In Sec. 3.4, we will show how A-BAD-BO uses BO to search for promising *local loss configurations* in the *local loss space*, concepts of which will be introduced next. We also provide a theoretical analysis of A-BAD-BO's cumulative regret in Sec. 4.

2.2. AutoAI problem setting

In this subsection, we will formally define several notations used in the AutoAI problem setting. We consider a general system with n ML components and at least one black-box component arranged in a directed acyclic graph; we will discuss in App. B.3 special systems not covered by our setting. Let each ML component i contain trainable parameters $\theta^{(i)} \in \mathbb{R}^{d_i}$ for $i = 1, \ldots, n$ where $d_i \in \mathbb{N}$. For conciseness, let $\theta \triangleq (\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n)}) \in \mathbb{R}^d$ denote all model parameters in the system with a potentially extremely high dimension $d = \sum_{i=1}^n d_i$. Since adjusting θ would directly influence the system output, let $F(x, \theta)$ represent the system output w.r.t. some system input x and θ such that it captures the complex interaction between components in a system and has no analytical form. So, given a *fixed* system test dataset $(x_j, z_j)_{j=1,\ldots,N}$, the system loss

$$L(\theta) \triangleq \textstyle\sum_{j=1}^N \text{loss}(F(x_j, \theta), z_j) \tag{2}$$

quantifies how the system performs w.r.t. any pre-defined loss function (e.g., predictive loss of a healthcare system). For simplicity, we assume black-box components are noiseless and hence, $L(\theta)$ is also noiseless (i.e., given a fixed system test dataset and parameters, the system loss is deterministic). This assumption eases our theoretical analyses (Theorems 3.2 and 4.1) but our experiments show that A-BAD-BO can optimize stochastic systems surprisingly well too (Sec. 6.1). We also provide some additional discussions in Sec. 5 related to how theoretical results in our paper could be extended for noisy systems.

Let l_i be a differentiable local loss function for ML component i (e.g., see Fig. 1) trained on a separate local dataset for $i = 1, \ldots, n$. For example, if $g(x', \theta^{(i)})$ represents the ML component i's output w.r.t. some input x' and its model parameters $\theta^{(i)}$ trained on local dataset $(x'_j, z'_j)_{j=1,\ldots,N}$. then the mean squared error $l_i(\theta^{(i)}) \triangleq N^{-1}\sum_{j=1}^N (g(x'_j, \theta^{(i)}) - z'_j)^2$ is a possible local loss function defined over $\theta^{(i)}$. So, l_i is differentiable w.r.t. $\theta^{(i)}$. Using the same example shown in Fig. 1, a liver model's loss function can be defined as its cross-entropy loss over an open source liver health test dataset (Hepatitis, 1988). For conciseness, define the vector function $\ell(\theta) : \mathbb{R}^d \mapsto \mathbb{R}^n$ as

$$\ell(\theta) \triangleq \left(l_1(\theta^{(1)}), l_2(\theta^{(2)}), \ldots, l_n(\theta^{(n)})\right)$$

to map the system parameters θ to a n-dimensional real-valued vector of the local loss of every ML component in the system. From hereon, we refer to this function's output as the system's *local loss configuration* (shown in Fig. 1) which lies in the *local loss space*. Note that the local datasets in which the local loss configuration is obtained from does not have to coincide with the distribution of the system dataset. For example, a liver health dataset or a traffic agent dataset (Helou et al., 2021) used to train a local ML component may not have the same demographics as the place in which a system is deployed in. This is one of the main reasons why simply optimizing the local losses till convergence might not necessarily lead to optimal system performance.

Our goal is to solve the system optimization problem (i.e., AutoAI):

$$\min_\theta L(\theta) \tag{3}$$

by jointly training the system parameters across all ML components. As mentioned in Sec. 1, both local and global approaches cannot solve this problem optimally as L has no analytical form and dimension of θ is extremely large.

3. Introducing A-BAD-BO

A-BAD-BO uses the system's local loss configuration as an auxiliary information source to ease the difficulty of solving problem (3). Intuitively, some relationship exists between $\ell(\theta)$ (i.e., ML components' local losses) and $L(\theta)$ (i.e., system loss) in real-world ML systems: If the local performance of each ML component is good, then the system performance should be good as well, even though this relationship is not necessarily monotonic. We have provided some empirical results in App. B.2 to validate this relationship. In the next subsection, we formalize this relationship and show how A-BAD-BO leverages this relationship to simplify and solve problem (3) efficiently.

3.1. Problem reparameterization

To explicitly capture the relationship between local loss configurations and the system's performance, we show that the system optimization problem (3) can interestingly be reparameterized into a bilevel optimization problem:

Theorem 3.1. *Let $S_{\boldsymbol{\ell}} \triangleq \{\theta \mid \ell(\theta) = \boldsymbol{\ell}\}$. Then, θ^* is a solution of the original system optimization problem, $\min_\theta L(\theta)$, iff $\boldsymbol{\ell}^* = \ell(\theta^*)$ is a solution of the reparameterized problem:*

$$\min_{\boldsymbol{\ell}} \min_{\theta \in S_{\boldsymbol{\ell}}} L(\theta) \tag{4}$$

where $\boldsymbol{\ell} \in \mathbb{R}^n$ is a local loss configuration of the given system with n ML components.

Its proof is in App. C.1 and shows that the solution set of θ^* in the original problem admits a set of $\boldsymbol{\ell}^*$ which is *exactly* the solution set of the reparameterized problem. The reparameterized problem (4) describes how a change in local loss configuration (at outer level) affects the best attainable system performance (at inner level) and presents us with another way to optimize a complex ML system: At the outer level, a local loss configuration $\boldsymbol{\ell}$ is selected in the local loss space. At the inner level, the selected local loss configuration is used to identify the set $S_{\boldsymbol{\ell}}$ of θ, from which the system parameters minimizing the system loss are obtained. So, an optimal local loss configuration $\boldsymbol{\ell}^*$ is one that can recover the optimal system parameters θ^* (for the original problem) at the inner level. However, simply locally optimizing each ML component to convergence does not produce an optimal local loss configuration $\boldsymbol{\ell}^*$ as such a local approach does not guarantee global system optimality, as shown in App. B.1.

Existing bilevel optimization techniques (Sinha et al., 2017) also cannot tackle problem (4) easily because θ at the inner level is still extremely high-dimensional and the relationship between $\boldsymbol{\ell}$ and the minimum of the system loss at the inner level has no analytical form. However, this problem's unique nested structure allows us to novelly combine the local and global approaches to solve it efficiently, as discussed next.

3.2. Brief algorithm description

Briefly, our algorithm, A-BAD-BO, solves the reparameterized problem (4) via an iterative approach (see orange box in Fig. 1). *Firstly*, we introduce a simple yet effective subroutine (Sec. 3.3) which exploits the differentiable structure of ML components (i.e., local approach) to solve the high-dimensional inner problem, $\min_{\theta \in S_{\boldsymbol{\ell}}} L(\theta)$. *Secondly*, we model the relationship between the selected $\boldsymbol{\ell}$ at the outer level and the minimum of the system loss at the inner level using a GP; this design choice will be explained in Sec. 3.4. This allows us to use BO (i.e., global approach) to account for any estimation error incurred by our subroutine and efficiently search for the optimal local loss configuration $\boldsymbol{\ell}^*$ (in a lower-dimensional local loss space) to run our subroutine (Sec. 3.4) on, asymptotically recovering optimal system parameters θ^*.

To illustrate our algorithm more concretely, consider a synthetic healthcare system (i.e., a much simpler variant than that in Fig. 1) with two sequential components represented by a composite function $g_1 \circ g_2$. Here, g_2 is a simple ML component with two trainable parameters $\theta^{(0)}$ and $\theta^{(1)}$, which estimates a patient's kidney health based on the kidney health data. On the other hand, g_1 is a doctor (i.e., black-box component) who looks at g_2's prediction and the patient's other clinical data to estimate the patient's diabetes risk score. Fig. 2 shows the local and system loss landscape for this system based on the local and system test datasets.

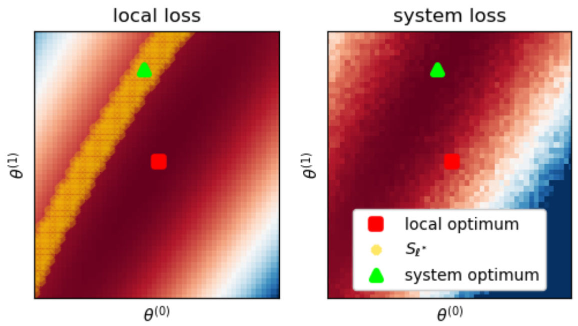


Figure 2. Local and system loss landscape in a simple sequential system. A-BAD-BO uses BO to find the optimal local loss configuration $\boldsymbol{\ell}^*$. The brown region is the set $S_{\boldsymbol{\ell}^*}$ of θ that attains $\boldsymbol{\ell}^*$. Our subroutine recovers the optimal system parameters from $S_{\boldsymbol{\ell}^*}$.

In Fig. 2 (right), directly finding optimal system parameters (green triangle) is challenging: The system loss landscape has no known analytical form since g_1 is black-box. Moreover, performing gradient descent directly on the local loss of g_2 (left) results in convergence to suboptimal system parameters (red square). A-BAD-BO instead uses BO to search for the optimal local loss configuration $\boldsymbol{\ell}^*$ in a lower-dimensional local loss space; in this case, it is 1-dimensional due to only one ML component. The brown region is the set $S_{\boldsymbol{\ell}^*}$ of θ which attains the optimal local loss configuration $\boldsymbol{\ell}^*$. A-BAD-BO then recovers the optimal system parameters from $S_{\boldsymbol{\ell}^*}$ using a subroutine (Sec. 3.3) which exploits the differentiable structure of the ML component g_2.

3.3. Subroutine to solve inner optimization problem

Let $\boldsymbol{\ell} \in \mathbb{R}^n$ be a candidate local loss configuration provided and $S_{\boldsymbol{\ell}} \triangleq \{\theta \mid \ell(\theta) = \boldsymbol{\ell}\}$. We propose a simple yet effective subroutine to estimate the solution of the inner problem:

$$\theta^*_{\boldsymbol{\ell}} \triangleq \operatorname{argmin}_{\theta \in S_{\boldsymbol{\ell}}} L(\theta) \qquad (5)$$

and minimum for the inner problem $y_{\boldsymbol{\ell}} \triangleq L(\theta^*_{\boldsymbol{\ell}})$.

Inspired by previous works on distribution extremum estimation (de Haan, 1981; Lee & Miller, 2022), our subroutine samples system parameters from the set $S_{\boldsymbol{\ell}}$ to estimate $y_{\boldsymbol{\ell}}$. To satisfy the constraint in $S_{\boldsymbol{\ell}}$, we exploit the differentiable structure of ML components: Given a local loss configuration $\boldsymbol{\ell} \in \mathbb{R}^n$, we randomly initialize θ and perform gradient descent on each ML component i's local loss function $l_i(\theta^{(i)})$ w.r.t. $\theta^{(i)}$ using appropriately chosen learning rates (Nesterov, 2013) until the constraint $\ell(\theta) = \boldsymbol{\ell}$ is reached; we will discuss practical ways to reach this constraint to a high degree of precision in Sec. 3.6. Then, this training process is repeated k times (i.e., k is chosen beforehand) to obtain k samples $\theta_1, \theta_2, \ldots, \theta_k$ of system parameters such that every sample satisfies the given constraint. Next, the system loss $L(\theta_i)$ for each sample θ_i of system parameters is evaluated to obtain *an estimator* of $y_{\boldsymbol{\ell}}$:

$$\tilde{y} \triangleq \min\{L(\theta_1), L(\theta_2), \ldots, L(\theta_k)\}\,.$$

In particular, $\tilde{y}$ is positively biased but *consistent*, i.e., $\tilde{y} \to y_{\boldsymbol{\ell}}$ as k increases, as shown in Theorem 3.2. Here, each training phase (across each ML component and sample) is independent and can be parallelized (Li et al., 2020) with sufficient computational budget.

By viewing the process as drawing k random samples from the set $S'_{\boldsymbol{\ell}} \triangleq \{L(\theta) \mid \ell(\theta) = \boldsymbol{\ell}\}$, estimator $\tilde{y}$ is simply the 1st order statistic (Arnold et al., 2008) of the random samples from $S'_{\boldsymbol{\ell}}$. Consequently, we can characterize how well $\tilde{y}$ estimates $y_{\boldsymbol{\ell}}$ depending on the sampling distribution and k. Our experiments show that the sampling distribution of $S'_{\boldsymbol{\ell}}$ generally follows the uniform or truncated exponential distributions, as shown in App. B.4. Hence, the result below shows the theoretical estimation error of our subroutine's estimator of $y_{\boldsymbol{\ell}}$ based on either of these sampling distributions:

Theorem 3.2. *Let $L(\theta_1), L(\theta_2), \ldots, L(\theta_k)$ be k samples drawn randomly from the set $S'_{\boldsymbol{\ell}} = \{L(\theta) \mid \ell(\theta) = \boldsymbol{\ell}\}$. Furthermore, let $S'_{\boldsymbol{\ell}}$ be lower bounded by $y_{\boldsymbol{\ell}}$ and upper bounded by $y_{\boldsymbol{\ell}} + \alpha$ for some $\alpha > 0$. Then, the 1st order statistic estimator $\tilde{y} = \min\{L(\theta_1), L(\theta_2), \ldots, L(\theta_k)\}$ follows a distribution of $y_{\boldsymbol{\ell}} + \epsilon$ with a non-negative random variable ϵ and the following holds:*

- *If each sample $L(\theta_i) \sim U(y_{\boldsymbol{\ell}}, y_{\boldsymbol{\ell}} + \alpha)$ for $i = 1, 2, \ldots, k$, then $\epsilon = \alpha\epsilon'$ with $\epsilon' \sim B(1, k)$ where B is the Beta distribution.*
- *If each sample $L(\theta_i) \sim y_{\boldsymbol{\ell}} + \exp_t(\lambda, \alpha)$ for $i = 1, 2, \ldots, k$ where $\exp_t(\lambda, \alpha)$ is a truncated exponential distribution governed by rate parameter λ and truncated at $\alpha > 0$, then ϵ is a random variable governed by the probability density function*

$$pdf_\epsilon(u) = \frac{\lambda k e^{-\lambda u}}{1 - e^{-\lambda\alpha}} \left(\frac{e^{-\lambda u} - e^{-\lambda\alpha}}{1 - e^{-\lambda\alpha}}\right)^{k-1} \text{ on } u \in [0, \alpha]\,.$$

Its proof is in App. C.2. Theorem 3.2 tells us that the estimation error (i.e., a random variable) of our subroutine depends on (a) the sampling distribution and (b) sampling size k. If the sampling distribution is concentrated nearer to the true minimum $y_{\boldsymbol{\ell}}$ (due to smaller α or difference in distribution type), our estimation error is more likely to be smaller. The estimation error is also likely to be smaller with a larger k. While the size of α depends on the problem setting, it is usually bounded because the system loss is usually bounded (e.g., misclassification rate) and our insight (i.e., the system should perform reasonably well when ML components perform well) from Sec. 3 tells us that α should be small when each element of $\boldsymbol{\ell}$ is relatively small. We have also provided results in App. B.4 to show that α is not large in empirical settings.

The advantage of the 1st order statistic estimator used in our subroutine lies in its implementation simplicity and effectiveness *even if one does not know the underlying sampling distribution*. In App. B.5, we describe an alternative *unbiased estimator* of $y_{\boldsymbol{\ell}}$ and discuss why such alternative estimators cannot recover better-performing system parameters than the 1st order statistic estimator. Our experiments (Sec. 6.1) also show that even though A-BAD-BO uses the subroutine to sample the system loss multiple times, it is still more sample-efficient than the other baselines.

3.4. Solving the outer optimization problem via Bayesian optimization

In the previous subsection, we have presented a subroutine to solve the inner problem. By viewing the inner problem as a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ which takes in a local loss configuration $\boldsymbol{\ell} \in \mathbb{R}^n$ to produce the minimum of the system loss at the inner level (that our subroutine estimates), the outer optimization problem can be succinctly written as

$$\min_{\boldsymbol{\ell}} f(\boldsymbol{\ell}).$$

We propose using BO (as introduced in Sec. 2.1) to solve the outer problem for three reasons. *Firstly*, f is a black-box function with no analytical form, and BO has become a principled and popular framework to optimize black-box functions (Garnett, 2023; Pyzer-Knapp, 2018); unlike original problem (3), which is high-dimensional, objective function f is of a much lower dimension and can be handled by BO effectively. *Secondly*, f here can be shown to be continuous (Folkman & Shapiro, 1967) and so it is appropriate for us to model f as a realization of a GP, allowing us to perform BO over it. *Thirdly*, our subroutine incurs estimation errors (Theorem 3.2) so we cannot solve the inner problem perfectly. Hence, this implies we can only obtain *noisy observations* of $f(\boldsymbol{\ell})$ for a given $\boldsymbol{\ell}$; fortunately, BO handles noisy function observations gracefully (Srinivas et al., 2010; Chowdhury & Gopalan, 2017) during the optimization process, still allowing us to recover optimal system parameters. Therefore, we find BO a suitable choice for solving the outer optimization problem.

3.5. Using A-BAD-BO to optimize complex systems

With the subroutine (local approach) introduced in Sec. 3.3 to solve the inner problem and the use of BO (global approach) in Sec. 3.4 to solve the outer problem, we now present A-BAD-BO (Algorithm 1) which simultaneously uses both approaches to optimize complex ML systems.

Algorithm 1 **A-BAD-BO**

```
1: Input: System parameters $\theta$, ML component loss functions $\ell(\theta)$ and system loss function $L(\theta)$ over fixed datasets, initial observations $\mathcal{D}_0 \triangleq \{(\boldsymbol{\ell}_0, \tilde{y}_0)\}$, SE kernel $\kappa$, sampling size $k$, parameter $\beta_t$ for acquisition step and total number of BO iterations $T$.
2: for $t = 0, 1, \ldots, T$ do
3:     $\boldsymbol{\ell}_{t+1} = \text{argmin}_{\boldsymbol{\ell}}\, \mu_t(\boldsymbol{\ell}) - \beta_t \sigma_t(\boldsymbol{\ell})$ (acquisition step)
4:     Random sample $\sim \{\theta \mid \ell(\theta) = \boldsymbol{\ell}_{t+1}\}$ to obtain $\theta_1, \ldots, \theta_k$.
5:     $\widehat{\theta}^*_{\boldsymbol{\ell}_{t+1}} = \text{argmin}_{\theta \in \{\theta_1, \ldots, \theta_k\}} L(\theta)$
6:     $\tilde{y}_{t+1} = L(\widehat{\theta}^*_{\boldsymbol{\ell}_{t+1}})$
7:     $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\boldsymbol{\ell}_{t+1}, \tilde{y}_{t+1})\}$
8:     Update the GP posterior with $\mathcal{D}_{t+1}$ and $\kappa$ (Sec. 2.1).
9: end for
10: $\widehat{\theta}^* = \text{argmin}_{\theta \in \{\hat{\theta^*}_{\boldsymbol{\ell}_0}, \ldots, \hat{\theta^*}_{\boldsymbol{\ell}_T}\}} L(\theta)$
```

At iteration t, A-BAD-BO uses the LCB acquisition function (Srinivas et al., 2010) on the GP posterior to propose a candidate local loss configuration $\boldsymbol{\ell}_{t+1}$ for the system (line 3); then, we use our subroutine to solve the inner optimization problem w.r.t. $\boldsymbol{\ell}_{t+1}$, yielding a *noisy* observation $\tilde{y}_{t+1} = y_{\boldsymbol{\ell}_{t+1}} + \epsilon_{t+1}$ where $y_{\boldsymbol{\ell}_{t+1}}$ is the minimum for the inner problem and ϵ_{t+1} is the estimation error associated with our subroutine (line 4,5 & 6). We also keep track of $\widehat{\theta}^*{}_{\boldsymbol{\ell}_{t+1}}$, the best system parameter we obtain for this proposed configuration. Next, we include $(\boldsymbol{\ell}_{t+1}, \tilde{y}_{t+1})$ in our historical observations and update the GP posterior (line 7 and 8). After the process repeats for T iterations, we obtain an estimate of the optimal system parameter $\widehat{\theta}^*$ (line 10).

3.6. Practical considerations

There are several ways to improve the effectiveness of A-BAD-BO in practice. *Firstly*, we can reduce the size of the local loss space by removing regions with obvious bad local loss configurations. For example, in a healthcare system, we can remove any configurations where an organ prediction model has less than 30% predictive accuracy over its local dataset because the system is unlikely to perform well with such bad-performing ML components. By doing so, we can improve the effectiveness of BO in solving the outer problem by reducing its search space over the local loss space. *Secondly*, to ensure our BO procedure does not propose unattainable local loss configurations (e.g., negative misclassification rate) for our subroutine, we train each ML component once w.r.t. its local dataset until convergence and use the smallest and largest local loss attained by each ML component (during training) as valid bounds in the local loss space for BO (Ha et al., 2019). *Thirdly*, our subroutine uses gradient descent to attain the constraint $\ell(\theta) = \boldsymbol{\ell}$. In general, it is difficult to use gradient descent (with fixed learning rate) to reach a loss value *precisely* (Qian, 1999). A decaying learning rate (You et al., 2019) and state-of-the-art optimizers in PyTorch (Imambi et al., 2021) allows us to reach this constraint with high precision, yielding empirically good results in our experiments (Sec. 6.1).

4. Theoretical Analysis of A-BAD-BO

A-BAD-BO uses BO to search for optimal local loss configurations in the local loss space. Hence, the classic cumulative regret given in Sec. 2.1, which does not contain observation noise, merely indicates the quality of local loss configuration chosen at each iteration. Equally important is how well we can recover optimal system parameters from a given local loss configuration (i.e., solution quality returned by subroutine over a chosen $\boldsymbol{\ell}$). Thus, we choose to analyze the growth of *attained cumulative regret* $\tilde{R}_T = \sum_{t=1}^{T} |\tilde{y}_t - f(\boldsymbol{\ell}^*)|$ where $\tilde{y}_t = f(\boldsymbol{\ell}_t) + \epsilon_t$ is the output of our subroutine (a noisy observation of $f(\boldsymbol{\ell}_t)$) for chosen $\boldsymbol{\ell}_t$ at each iteration t and $\boldsymbol{\ell}^*$ is the optimal local loss configuration. This regret is more appropriate in our setting because it indicates *both* the quality of local loss configuration found by the BO procedure and the quality of system parameters recovered by our subroutine. We show that if optimization objective f lies in a Reproducing Kernel Hilbert Space (RKHS) with bounded RKHS norm (a standard setting in BO works (Chowdhury

& Gopalan, 2017)), A-BAD-BO with sampling scheme of $k = t$ at each iteration achieves sub-linear attained regret growth, i.e., $\tilde{R}_T/T \to 0$ as $T \to \infty$, implying our algorithm asymptotically converges to the optimal system parameters:

Theorem 4.1. *Let $S'_{\boldsymbol{\ell}}$ be the set defined in Theorem 3.2 and f be the system optimization objective with bounded RKHS norm: $||f||_\kappa = \sqrt{\langle f, f\rangle_\kappa} \leq B$ w.r.t. kernel κ. Then, running A-BAD-BO over f using the IGP-LCB acquisition function (Chowdhury & Gopalan, 2017) and sampling size $k = t$ at BO iteration $t = 1, \ldots, T$ with γ_T as the maximum information gain after T iterations yields the following attained cumulative regret with probability at least $1 - \delta$:*

- $\tilde{R}_T = O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4}) + \frac{\alpha\psi(T+1)}{\sqrt{\delta}}\right)$ *if each random sample from $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width $\alpha > 0$. ψ refers to the digamma function.*
- $\tilde{R}_T = O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4}) + \sqrt{\frac{(1-e^{-\alpha})T}{\delta}}\right)$ *if each random sample from $S'_{\boldsymbol{\ell}}$ is exponentially distributed with rate $\lambda = 1$ and truncated at α with $0 < \alpha \leq 1$.*

The proof is provided in App. C.3 and relies on deriving the upper bound for $|\tilde{y}_t - f(x^*)| = |f(x_t) - f(x^*) + \epsilon_t|$ at each BO iteration using standard techniques for BO with LCB-based acquisition functions (Chowdhury & Gopalan, 2017) and using the expectation of ϵ_t from Theorem 3.2. For the SE kernel, $\gamma_T = O((\log T)^{T+1}) \leq O(\sqrt{T})$ (Srinivas et al., 2010). We see that $\tilde{R}_T$ for both cases is dependent on α: a larger α implies it is more difficult to estimate the solution of the inner problem and so increases the attained cumulative regret. Interestingly, due to differences in the density of distributions, α has a smaller effect on the growth of attained cumulative regret in the exponentially distributed (truncated) case than the uniformly distributed case.

In practice, we may not have the computing budget to use a sampling scheme of $k = t$ (when BO iterations become large, we have to train ML components and query for the system loss many times per iteration). In an extreme case, every system query may be computationally expensive (Snoek et al., 2012). To mitigate this, we can choose a sampling scheme which consumes less computing budget (e.g., fixed k or $k = \sqrt{t}$). In App. C.4, we provide additional analyses on how these sampling schemes would influence our attained cumulative regret. We also investigate the effect of varying sampling size k in our experiments (Sec. 6.2).

5. Extension to noisy ML systems

In Sec. 2.2, we have assumed that black-box components in our system are noiseless and hence the system loss $L(\theta)$ are also noiseless (i.e., deterministic). Some readers may wonder how a noisy system setting would influence the results introduced in this paper. In this section, we briefly elaborate how our paper's results could be extended for noisy systems.

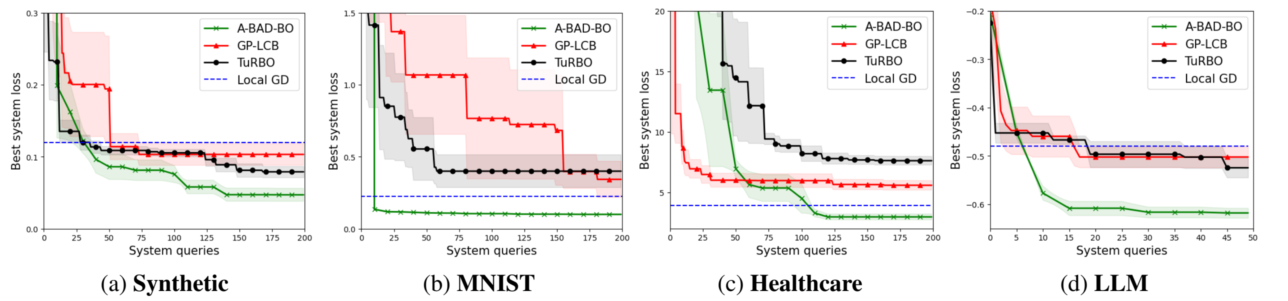
Firstly, we need to define the *expected* system loss as $L(\theta)_{\mathbb{E}} \triangleq \sum_{j=1}^{N} \mathbb{E}(\text{loss}(F(x_j, \theta), z_j))$ (notice the additional expectation term). Following which, our reparameterization Theorem 3.1 still holds true for the expected system loss. However, sample i drawn from $S'_{\boldsymbol{\ell}}$ in our subroutine (Sec. 3.3) will now be observed as $L(\theta_i)_{\mathbb{E}} + \lambda$, where λ is the random variable associated with the system noise. Unfortunately, unlike the noiseless case, each sample observation now may have a complex distribution and we cannot easily write the estimation error distribution of our subroutine's estimator (Theorem 3.2) in closed form. For example, if λ comes from a Gaussian distribution and each sample $L(\theta_i)_{\mathbb{E}}$ (drawn from $S'_{\boldsymbol{\ell}}$) is uniformly distributed, then each sample observation has a complex Bhattacharjee distribution (Bhattacharjee et al., 1963); it is not clear to us, at this time, if our subroutine's estimation error has a closed form w.r.t. this sampling distribution. However, by taking repeated samples for the same set system parameters and taking their average, we can reduce the system noise to zero asymptotically and hence our subroutine is still consistent (i.e., with sufficient samples, we can recover $\theta^*_{\boldsymbol{\ell}}$ for inner problem (5) by reducing the system noise close to zero).

Despite the weaker theoretical results (no closed form noise distribution) for noisy systems, we show in our experiments that A-BAD-BO still outperforms other baselines in optimizing noisy systems, highlighting our algorithm's effectiveness.

6. Experiments and Discussion

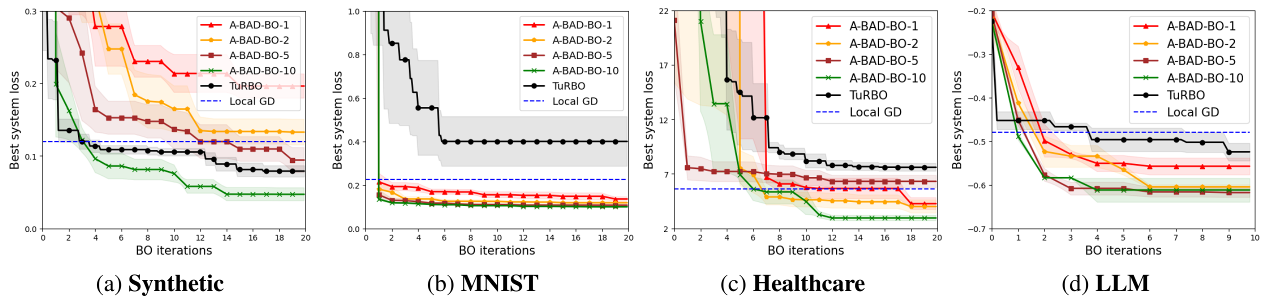
We use A-BAD-BO to optimize various synthetic and real-world complex systems found in AutoAI problems to demonstrate its effectiveness. We also compare the performance of A-BAD-BO with other baselines consisting of local and global approaches. We choose to optimize four systems consisting of multiple ML components and black-box components arranged in a directed acyclic graph. In Sec. 2.2, we have assumed a setting with noiseless systems to ease our theoretical analyses. However, to make our systems more realistic, we use *noisy* black-box components to make each system *stochastic* in our experiments. More details on each system can be found in App. D.1and our code can be found at `https://github.com/chenzhiliang94/A-BAD-BO`.

Our systems are: (a) **Synthetic system** with 9 ML components and 3 black-box components, totaling 26 trainable parameters. (b) **MNIST system** with 2 MNIST digit classifiers (neural networks), 10 ML components and 2 black-box



(a) **Synthetic** (b) **MNIST** (c) **Healthcare** (d) **LLM**

Figure 3. Comparison of **A-BAD-BO**'s convergence with local (**Local GD**) and global (**GP-LCB**, **TuRBO**) approaches (Lower is better).



(a) **Synthetic** (b) **MNIST** (c) **Healthcare** (d) **LLM**

Figure 4. Ablation study on how sampling size k affects **A-BAD-BO**'s convergence rate (Lower is better).

components, totaling 1694 trainable parameters. (c) **Healthcare system** with 4 organ health prediction ML components, 1 aggregation component and 2 black-box components to imitate doctors' assessments in evaluating the prediction from ML components, totaling 198 trainable parameters. (d) An advanced **LLM prompt engineering system** (built with similar principles as Zhou et al. (2022)) which produces good prompts from *noisy examples* while *incurring lower LLM query cost*. In our setting, 50% of input examples are corrupted with noise, a commonly seen phenomenon in real-world text data (Bolding et al., 2023). The LLM system consists of three sequential components: an example-picker ML component to select useful examples from an input list of noisy examples, ChatGPT (OpenAI, 2023), a black-box component, to generate a list of plausible prompts from the selected examples (since fewer examples are used, lower query cost is incurred), and a prompt-filter ML component which performs *prompt valuation* and selects the most appropriate prompt (w.r.t. selected examples) as the system output. Both ML components are DistilBERT (Sanh et al., 2019) models with 66 trainable million parameters each.

We compare A-BAD-BO with local and global approaches. For the local approach, we independently optimize each ML component w.r.t. its local dataset using gradient descent (labeled **Local GD** in our plots). For global approaches, we use vanilla BO with the GP-LCB acquisition function (Srinivas et al., 2010) (labeled **GP-LCB** in our plots) and a more sophisticated trust region BO (labeled **TuRBO** in our plots) introduced by Eriksson et al. (2019) over parameters of all ML components w.r.t. the system loss. For ML components which are large neural networks (e.g., DistilBERT in our LLM system), performing BO over all trainable parameters is impractical due to their extreme dimensionality; so, we train the ML component over its local dataset till convergence and perform BO only on smaller subset of trainable parameters residing in the neural network's output layer.

6.1. Optimality results

We use a sampling size of $k = 5$ for optimizing the LLM system and $k = 10$ for other systems. For comparison fairness, we use the *same number of system queries in all approaches* and plot the best system loss achieved after each system query (Fig. 3). We also use a blue dotted line to indicate the system loss achieved by Local GD after training each ML component (w.r.t. local datasets) till convergence.

A-BAD-BO outperforms local and global approaches. Our results (Fig. 3) show that across all four systems, A-BAD-BO (green line) attains a lower system loss than Local GD (blue dotted line) within 200 system queries; this corroborates our claim that A-BAD-BO achieves better system optimality than local approaches because we explicitly incorporate the system loss into the optimization process (while local approaches do not). Furthermore, A-BAD-BO attains lower system loss than GP-LCB and TuRBO after the same number of system queries, implying that it is *more sample-efficient* as compared to conventional global approaches (i.e., achieving better or same optimality with fewer system queries). This occurs because unlike conventional global approaches, A-BAD-BO uses the system's local loss configuration as an auxiliary information source to reduce the optimization problem dimension while conventional global approaches do not. Furthermore, as mentioned, A-BAD-

BO performs BO over a much lower-dimensional problem as opposed to global approaches in a system setting (due to problem reparameterization), leading to better optimization performance after the same number of system queries. Therefore, our results indicate that A-BAD-BO is superior to both local and global approaches in optimizing complex ML systems.

In addition, we investigated the use of alternative acquisition functions (Wilson et al., 2018) and other early local optimization termination techniques (more details and results can be found in App. D.3 and D.4). In general, we found that LCB acquisition function outperforms the Expected Improvement (EI) acquisition function and A-BAD-BO outperforms alternative optimization approaches related to early stopping.

6.2. Ablation study on sampling size k

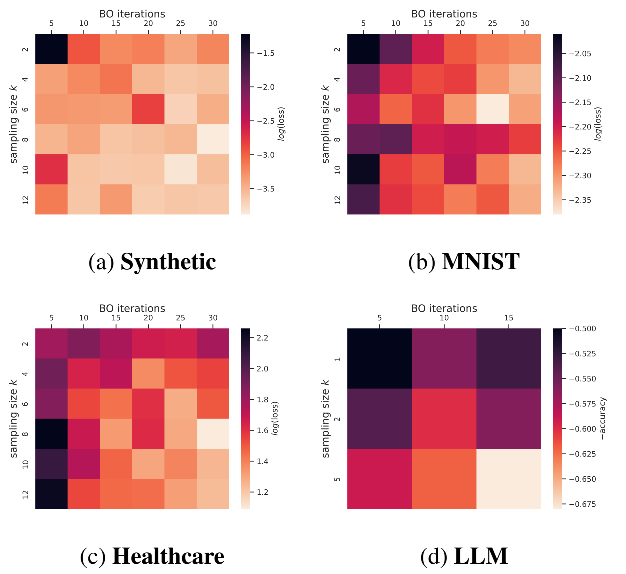
We also conduct ablation studies w.r.t. sampling size k by running A-BAD-BO with a fixed number of BO iterations and varying sampling size k in Fig. 4. Similarly, we compare the resulting performance with local and global approaches (GP-LCB omitted for plot clarity reasons because TuRBO yields similar or better results). For comparison fairness, we plot the convergence of TuRBO after *200 iterations* (since A-BAD-BO uses more system queries per BO iteration).

Larger k improves convergence rate of A-BAD-BO. Our result (Fig. 4) shows that increasing k whilst keeping the total number of BO iterations fixed improves the convergence of A-BAD-BO. This corroborates Theorem 3.2 where our estimation error of the inner problem's minimum decreases with larger k. This leads to less noisy observations at each BO iteration, improving convergence rate of A-BAD-BO.

Small sampling size can still be effective. Interestingly, Fig. 4 also shows that for **MNIST**, **Healthcare** and **LLM** systems, running A-BAD-BO with small sampling size (e.g., $k = 1$) *still* outperforms local and global optimization approaches. This suggests that the system's local loss configuration is a strong source of information and we do not have to estimate the minimum of the inner problem too accurately for A-BAD-BO to yield good performance. From a theoretical perspective, this indicates that in some settings, α is small and hence the estimation error shown in Theorem 3.2 is small despite the small sampling size. As a result, A-BAD-BO's regret bound is lower, allowing it to achieve better performance than other baselines.

6.3. Varying sampling size and BO iterations together

Since the total number of system queries used in A-BAD-BO is kT, we also investigate how varying sampling size k and BO iteration T *together* affects the convergence rate of A-BAD-BO in Fig. 5. The results show that increasing k and T both improves A-BAD-BO's performance. Furthermore, given a *fixed* system query budget (this occurs if each system query is considered computationally expensive (Snoek et al., 2012)), it is better to choose a moderate sampling size and BO iteration as compared to more extreme values (e.g., for a system query budget of $kT = 60$, choosing $k, T = (4, 15)$ yields better performance than $k, T = (2, 30)$ or $(30, 2)$). This observation can be explained theoretically: selecting larger sampling size k reduces the observation error at each BO iteration but allows for fewer BO iterations T; therefore, a tradeoff (as shown in Fig. 5) exists between the choice of k and T.



(a) **Synthetic** (b) **MNIST**

(c) **Healthcare** (d) **LLM**

Figure 5. Increasing k and number of BO iterations *both* improves the performance of A-BAD-BO. Moreover, for a fixed system query budget kT, it is better to choose moderate k and T values.

7. Conclusion

Our paper proposes A-BAD-BO, a novel algorithm to tackle AutoAI problems. A-BAD-BO optimizes complex ML systems with black-box and differentiable ML components by using BO (global approach) and exploiting the differentiable structure of ML components (local approach). We provide theoretical analysis of A-BAD-BO and show that it attains superior performance in our experiments. One limitation of A-BAD-BO is that our problem dimensionality scales with the number of ML components in the system. Even though systems with large number of ML components are rare in reality, we can extend A-BAD-BO to handle systems with extremely large number of ML components by exploiting a system's graphical structure (e.g., decomposing a system into smaller sub-systems in which A-BAD-BO can be applied independently). The optimization procedure can also be extended to federated settings (Lin et al., 2023).

Acknowledgements

This research is supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. The Mens, Manus, and Machina (M3S) is an interdisciplinary research group (IRG) of the Singapore MIT Alliance for Research and Technology (SMART) centre. This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018). Zhiliang Chen is supported by the Institute for Infocomm Research of Agency for Science, Technology and Research (A*STAR).

Impact Statement

Our work presents an algorithm that is used to optimize the performance of complex real-world systems. With the prevalent use of complex systems in the real world (e.g., robotics, healthcare, finance, LLM pipelines), we find our work impactful and important. We do not foresee any obvious negative societal consequences from our work.

References

Andriushchenko, M., D'Angelo, F., Varre, A., and Flammarion, N. Why do we need weight decay in modern deep learning?, 2023.

Arbel, J., Marchal, O., and Nguyen, H. D. On strict subgaussianity, optimal proxy variance and symmetry for bounded random variables, 2019.

Arnold, B. C., Balakrishnan, N., and Nagaraja, H. N. *A first course in order statistics*. SIAM, 2008.

Astudillo, R. and Frazier, P. Bayesian optimization of function networks. In *Proc. NeurIPS*, volume 34, pp. 14463–14475, 2021.

Bhattacharjee, G., Pandit, S., and Mohan, R. Dimensional chains involving rectangular and normal error-distributions. *Technometrics*, 5(3):404–406, 1963.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Bolding, Q., Liao, B., Denis, B. J., Luo, J., and Monz, C. Ask language model to clean your noisy translation data. *arXiv:2310.13469*, 2023.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv:1312.6203*, 2013.

Bull, A. D. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(88):2879–2904, 2011.

Choi, J. and Srivastava, H. Some summation formulas involving harmonic numbers and generalized harmonic numbers. *Mathematical and computer Modelling*, 54 (9-10):2220–2234, 2011.

Chowdhury, S. R. and Gopalan, A. On kernelized multi-armed bandits. In *Proc. ICML*, pp. 844–853. PMLR, 2017.

Dai, Z., Lau, G. K. R., Verma, A., Shu, Y., Low, B. K. H., and Jaillet, P. Quantum Bayesian Optimization. In *Proc. NeurIPS*, 2023a.

Dai, Z., Nguyen, Q. P., Tay, S. S., Urano, D., Leong, R., Low, B. K. H., and Jaillet, P. Batch Bayesian optimization for replicable experimental design. *arXiv:2311.01195*, 2023b.

de Haan, L. Estimation of the minimum of a function using order statistics. *Journal of the American Statistical Association*, 76(374):467–469, 1981.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Eriksson, D. and Jankowiak, M. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Proc. UAI*, pp. 493–503. PMLR, 2021.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local Bayesian optimization. In *Proc. NeurIPS*, volume 32, pp. 5496–5507, 2019.

Folkman, J. and Shapiro, N. On the continuity of the minimum set of a continuous function. *Journal of Mathematical Analysis and Applications*, 17(3), 1967.

Frazier, P. I. A tutorial on Bayesian optimization. *arXiv:1807.02811*, 2018.

Garnett, R. *Bayesian Optimization*. Cambridge Univ. Press, 2023.

Ha, H., Rana, S., Gupta, S., Nguyen, T., Venkatesh, S., et al. Bayesian optimization with unknown search space. *Proc. NeurIPS*, 32, 2019.

Hase, F., Roch, L. M., Kreisbeck, C., and Aspuru-Guzik, A. Phoenics: a bayesian optimizer for chemistry. *ACS central science*, 4(9):1134–1145, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. CVPR*, pp. 770–778, 2016.

Helou, B., Dusi, A., Collin, A., Mehdipour, N., Chen, Z., Lizarazo, C., Belta, C., Wongpiromsarn, T., Tebbens, R. D., and Beijbom, O. The reasonable crowd: Towards evidence-based and interpretable models of driving behavior. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 6708–6715. IEEE, 2021.

Hepatitis. Hepatitis dataset. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C5Q59J.

Honovich, O., Shaham, U., Bowman, S. R., and Levy, O. Instruction induction: From few examples to natural language task descriptions. *arXiv:2205.10782*, 2022.

Imambi, S., Prakash, K. B., and Kanagachidambaresan, G. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pp. 87–104, 2021.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Karkus, P., Ma, X., Hsu, D., Kaelbling, L. P., Lee, W. S., and Lozano-Pérez, T. Differentiable algorithm networks for composable robot learning. *arXiv:1905.11602*, 2019.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, A. and Miller, S. J. Generalizing the german tank problem. *arXiv:2210.15339*, 2022.

Lengagne, S., Ramdani, N., and Fraisse, P. Guaranteed computation of constraints for safe path planning. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 312–317, 2007.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv:2006.15704*, 2020.

Lin, X., Xu, X., Ng, S.-K., Foo, C.-S., and Low, B. K. H. Fair yet asymptotically equal collaborative learning. In *Proc. ICML*, 2023.

Lin, X., Wu, Z., Dai, Z., Hu, W., Shu, Y., Ng, S.-K., Jaillet, P., and Low, B. K. H. Use your INSTINCT: Instruction optimization using neural bandits coupled with transformers. In *Proc. ICML*, 2024.

Marchal, O. and Arbel, J. On the sub-gaussianity of the beta and dirichlet distributions. *arXiv preprint arXiv:1705.00048*, 2017.

Moriconi, R., Deisenroth, M. P., and Sesh Kumar, K. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109:1925–1943, 2020.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2013.

OpenAI. Gpt-4 technical report, 2023.

Penrose, K. W., Nelson, A. G., and Fisher, A. G. Generalized body composition prediction equations for men using simple measurement techniques. *Medicine and Science in Sports and Exercise*, 17:189, 1985.

Pyzer-Knapp, E. O. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2–1, 2018.

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, 2019.

Siddhartha, M. Heart disease dataset, 2020.

Sinha, A., Malo, P., and Deb, K. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Proc. NeurIPS*, 25, 2012.

Soundarapandian, R. and Eswaran. Chronic kidney disease. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5G020.

Spanier, J., Oldham, K. B., and Romer, R. H. An atlas of functions, 1988.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pp. 1015–1022, Madison, WI, USA, 2010. Omnipress.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tay, S. S., Foo, C.-S., Urano, D., Leong, R., and Low, B. K. H. Bayesian optimization with cost-varying variable subsets. In *Proc. NeurIPS*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Proc. NeurIPS*, 30, 2017.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and De Feitas, N. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wilson, J. T., Hutter, F., and Deisenroth, M. P. Maximizing acquisition functions for bayesian optimization, 2018.

You, K., Long, M., Wang, J., and Jordan, M. I. How does learning rate decay help modern neural networks? *arXiv:1908.01878*, 2019.

Zhang, A., Sheng, L., Chen, Y., Li, H., Deng, Y., Wang, X., and Chua, T.-S. On generative agents in recommendation. *arXiv:2310.10108*, 2023.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. Large language models are human-level prompt engineers. *arXiv:2211.01910*, 2022.

A. Notations

Here, we provide a list of important mathematical notations used frequently in our paper. Some of these notations could contain subscripts in our paper to denote the context where they are used (e.g., in BO iterations). We have defined them accordingly in places where these notations are introduced throughout the paper and this table can be used for ease of reference.

Table 1. Important mathematical notations used in this paper.

Notation	Definition
$\theta^{(i)}$	Trainable parameters in ML component i.
θ	Trainable parameters across *all* ML components (also referred to as system parameters).
$l_i(\theta^{(i)})$	Local loss function of ML component i.
$\ell(\theta)$	Local loss vector function representing local loss function of *every* ML component.
$L(\theta)$	System loss w.r.t. θ.
$\boldsymbol{\ell}$	Local loss configuration (a realization of function vector $\ell(\theta)$).
$S_{\boldsymbol{\ell}}$	The set $\{\theta \mid \ell(\theta) = \boldsymbol{\ell}\}$.
$S'_{\boldsymbol{\ell}}$	The set $\{L(\theta) \mid \ell(\theta) = \boldsymbol{\ell}\}$.
$y_{\boldsymbol{\ell}}$	minimum for the inner problem.
$\tilde{y}$	Our subroutine's estimated minimum for the inner problem.
k	Sampling size used by our subroutine.
ϵ	Estimation error of our subroutine. Also treated as *observation noise* in A-BAD-BO under the BO framework.
$f(\boldsymbol{\ell})$	Function representation (with no analytical form) of outer problem. Modeled as a realization of a GP.
$\tilde{R}_T$	Attained cumulative regret after T BO iterations (exact definition found in Sec. 4).

B. Additional Discussions

B.1. Why does local component optimality not imply system optimality?

In this subsection, we will investigate why locally optimizing ML components within a system does not guarantee system optimality. First of all, we provide empirical evidence of this phenomenon in Fig. 7a which shows that beyond a certain point, the system loss actual *increases* when we continue to optimize ML components locally. Then, we investigate why this happens by examining how real-world datasets are generated for complex systems. To begin, consider a two-component system defined by the composite function: $F = g \circ f(\theta, x)$ where $f(\theta, x) = \theta x$ (a ML component with trainable θ) and $g(x) = x^2$ (a black-box component) (Fig. 6). Without loss of generalization, let $\theta^* = 2.1$ be the "true" parameter for the system (this assumption helps us to reason about the labeling process more concretely).

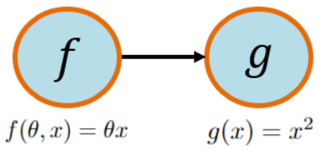


Figure 6. A simple sequential system represented by composite function $F = g \circ f(\theta, x)$.

B.1.1. OPTIMIZATION PERSPECTIVE

We first examine this phenomenon from an optimization perspective. Consider labels generated via a noisy labeling process for N datapoints in a system test dataset. For a given input x_j with $j = 1, \ldots, N$, the label is $g \circ f(\theta^*, x_j) + \epsilon_1$ where ϵ_1 is a random noise variable involved in the labeling process (e.g., human inconsistency). Now, consider another labeling process for the local dataset of f: for the same input x_j the local label is $f(\theta^*, x_j) + \epsilon_2$ where ϵ_2 is another random noise variable. Assuming the mean-squared error (MSE) is used, local optimization (only training ML component w.r.t. local dataset) learns $\hat{\theta}_{\text{local}} = \text{argmin}_\theta \sum_j^N \mathbb{E}_{\epsilon_2}(f(\theta^*, x_j) - f(\theta, x_j) + \epsilon_2)^2$. However system optimization yields $\hat{\theta}_{\text{system}} = \text{argmin}_\theta \sum_j^N \mathbb{E}_{\epsilon_1}(g \circ f(\theta^*, x_j) - g \circ f(\theta, x_j) + \epsilon_1)^2$. In general, $\hat{\theta}_{\text{local}} \neq \hat{\theta}_{\text{system}}$ because ϵ_1 and ϵ_2 may not be identically distributed (this happens because not the same person may have labeled both datasets, or the system and local dataset labeling process are not equally challenging). Furthermore, local optimization of a ML component does not account for how its intermediate output is influenced by later components within the system.

Next, we demonstrate this phenomenon using a simple dataset. Let the system dataset be $\mathcal{X}_L, z_L = \{x_1, z_1 = (3, 30.2), x_2, z_2 = (1, 10.22)\}$ and let the ML component local dataset be $\mathcal{X}_l, z'_l = \{x_1, z'_1 = (3, 5.5), x_2, z'_2 = (1, 3.2)\}$ (These are generated via noisy label generation process based on ground truth θ^*). From these datasets, local optimization (i.e., performing gradient descent on f w.r.t. θ) yields $\hat{\theta}_{\text{local}} \approx 1.97$ while system optimization using $\mathcal{X}_L, y_L$ yields $\hat{\theta}_{\text{system}} \approx 1.85$. This further demonstrates how local component optimality does not necessarily imply system optimality (since they produce different optimization results).

B.1.2. COMPONENT PERTURBATION PERSPECTIVE

In reality, we also observe the phenomenon of *component perturbation* in real-world systems. Roughly speaking, component perturbation refers to the phenomenon that a black-box component in the system deviates from its intended behavior. For example, the motion-planning software in a robotic system may be constantly under feature development under the Agile framework [1] and may not handle all challenging scenarios correctly (e.g., navigating narrow gaps due to difficulty in computing constraints (Lengagne et al., 2007)). In another example, a human doctor playing the role of a black-box component in a healthcare system (Fig. 1) may have some unintended bias when analyzing data. Note that such perturbations only occur as part of the system behavior and **not during the labeling process**. In such cases, *a ML component may need to perform suboptimally to compensate for errors produced by black-box components*.

For example, consider the case where a self-driving car's motion-planning software (black-box component) receives bounding boxes of pedestrians from an upstream object detection module (ML component). The motion-planning software may have purposely enlarged the bounding boxes it receives for safety reasons (the software engineer of the motion-planning module might have done so while the module is constantly under feature development). This enlargement corresponds to *component perturbation* in the motion-planning module. As such, even if the object detection module detects objects' bounding boxes perfectly, the self-driving car behaves too conservatively (i.e., unable to traverse gaps between pedestrians even if there is enough space) because the motion-planning module purposely enlarges any pedestrian bounding boxes. So, if the object detection module purposely produces bounding boxes which are *slightly smaller* than the actual pedestrian (i.e., a suboptimal behavior), we actually *cancel out* the wrongful bounding box enlargement by the black-box component, hence making the system behave optimally. We would like to emphasize that while this example here can be fixed by simply correcting the wrong bounding box enlargement (if we know it exists) directly, real-world systems could contain multiple black-box components with non-obvious perturbations, making direct corrections difficult (we do not know which components have perturbations or if they even exist). As such, we should allow ML components in the system to correct these perturbations automatically, eventually producing optimal system performance.

We can demonstrate this phenomenon mathematically as well. Here, we consider a case with only a single noiseless system datapoint (we remove the index for easier reference): $(z, g \circ f(\theta^*, z))$ and local datapoint (for f): $(z, f(\theta^*, z))$. These datasets reflect what an oracle (e.g., the system owner) expects the system to behave. Now, assume black-box component g is perturbed to g' such that $g'(z) = g(z) + \epsilon$ where ϵ is a fixed positive error term.

Even if one manages to learn θ^* using local optimization, the system produces a suboptimal output of $g' \circ f(\theta^*, z) = g \circ f(\theta^*, z) + \epsilon$, incurring an error deviation of ϵ. This occurs because local optimization does not take into account other components (and perturbations) within the system. However, if we consider the entire system behavior (and perturbations) explicitly during optimization, we can learn better parameters that lead to better system performance. For example, if we are able to find an alternative θ' such that $g \circ f(\theta^*, z) - \epsilon < g \circ f(\theta', z) < g \circ f(\theta^*, z)$, then we can see that a smaller error deviation:

$$\begin{aligned} |g' \circ f(\theta', z) - y| &= |g \circ f'(z) + \epsilon - y| \\ &= |(g \circ f'(z) - g \circ f(z)) + \epsilon| \\ &< \epsilon \end{aligned} \tag{6}$$

can be achieved. As such, despite the positive error term introduced in the black-box component g, we are able to rectify (possibly not perfectly, but better than ignoring it) it by adjusting θ accordingly. In fact, cross-model compensation was noted informally in Karkus et al. (2019): "... allows the modules to adapt to one another and compensate for imperfect models and algorithms ...". Therefore, it is possible that some ML components in real-world complex systems have to

[1] Agile software development comes under the umbrella framework of incremental software development: `https://en.wikipedia.org/wiki/Iterative_and_incremental_development`.

perform suboptimally (purposely) to compensate some errors produced by other components.

B.2. Empirical relationship between ML component local performance and system performance

In our paper, we have used the perspective: If the local performance of each ML component is good, then the system performance should be good as well (although this relationship is neither monotonic nor has an analytical form). While this perspective is intuitive, we provide results from our empirical experiments to support this perspective.

In our experiments, we consider systems which are *stochastic* in nature (caused by noisy black-box components). From Fig. 7, we see that as we train each ML component w.r.t. its local dataset and reduce its local loss (different colored lines), the system loss (**black** line) shows a decreasing trend (fluctuations are caused by system stochasticity). This demonstrates that good ML component performance tend to lead to better system performance (although this relationship is not monotonic).

More importantly, the results show that we *do not* attain the optimal system loss even after each ML component has converged after training (local approach does not guarantee system optimality). In fact, in **Synthetic system**, the system loss actually *increases* as the ML components' local losses decrease beyond a certain level (validating that the relationship is non-monotonic).

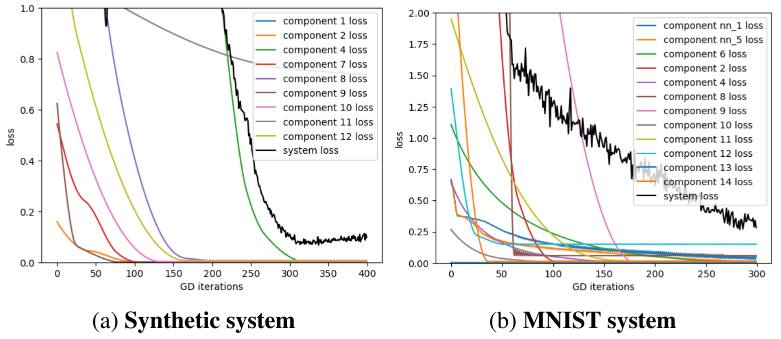


Figure 7. How system loss changes when each ML component's local loss is reduced via gradient descent w.r.t. a local dataset. This demonstrates that when ML components in a system perform well, the system performs well too. However, the increasing system loss at the end of **Synthetic system** demonstrates that local optimality does not imply system optimality.

B.3. Special cases of multi-component ML systems

Our problem setting considers a system with n ML components and one or more black-box components (Sec. 2.2). Under this setting, the system does not have an analytical form and large number of parameters, making optimization difficult (our work focuses on addressing these challenges). Here, we examine some special systems which fall outside of our problem setting and discuss their relevancy in real-world problems.

Systems without black-box components. If a system has no black-box components and is composed entirely of ML components arranged in some specific directed acyclic graphical manner, the entire system has an analytical form (because each ML component has an analytical form). So, one can simply perform gradient descent over the entire system w.r.t. the system loss to jointly train parameters residing in all ML components (just like training a neural network). Interestingly, we note that our algorithm, A-BAD-BO, is still able to work just as fine in this setting. However, we would like to emphasize that many real-world systems typically *do* contain black-box components (Zhang et al., 2023).

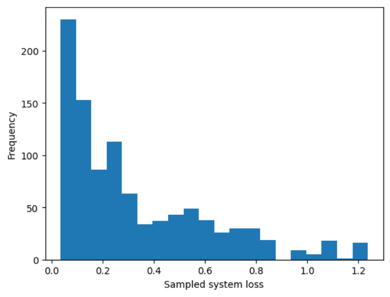
Systems without ML components. In our setting, we assume system parameters only reside in ML components. Hence, a system without ML components is not relevant to our work because we do not have to optimize it. However, in our future work we would like to consider cases where system parameters reside in black-box components as well.

Systems with overwhelming number of black-box components. If a system contains an overwhelming number (e.g., much more than number of ML components) of black-box components which are noisy, then the system may be extremely stochastic. In this case, the positive relationship between each ML component's local performance and the system's performance (as shown in App. 7) may not be strong. However, our algorithm, A-BAD-BO, is still able to work for stochastic systems. In particular, our experimental results (Sec. 3) show that A-BAD-BO is able to handle stochastic systems well (compared to other baselines).

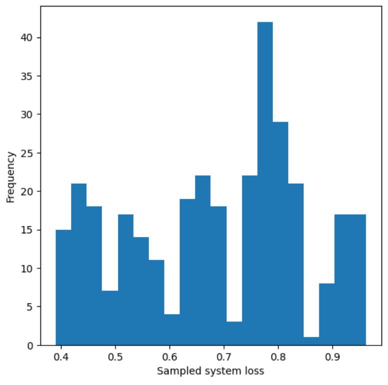
B.4. Empirical sampling distributions

In this subsection, we will provide some empirical observations of the sampling set $S'_{\boldsymbol{\ell}}$ defined in Theorem 3.2 that we encountered in experiments. More details of these experiments can be found in App. D.1.

In general, we observe most sampling distribution of $S'_{\boldsymbol{\ell}}$ in our experiments follows a truncated exponential distribution (Fig. 8a) or approximately a uniform distribution (Fig. 8b). In addition, we also observe that the sampling set in practice is small (more formally, this means α defined in Theorem 3.2 is small). This also implies the error involved in estimating minimum of $S'_{\boldsymbol{\ell}}$ is also relatively small (the exact relationship between this error and α is captured in Theorem 3.2), allowing better convergence rate of A-BAD-BO.



(a) **Truncated exponential**



(b) **Uniform** (approximately)

Figure 8. Examples of sampling distribution of $S'_{\boldsymbol{\ell}}$ encountered during experiments for a particular local loss configuration $\boldsymbol{\ell}$ recommended by BO. For each sampled θ which satisfies $\ell(\theta) = \boldsymbol{\ell}$, we evaluate the system loss. A large sampling size of is used here so we can approximately show the true sampling distribution (in real experiments, we use a much smaller sampling size). **Left** shows a sampling distribution similar to the **truncated exponential distribution**. **Right** shows a sampling distribution similar to a **uniform distribution**.

B.5. Alternative estimator of $y_{\boldsymbol{\ell}}$

In Sec. 3.3, we introduced an estimator for the minimum $y_{\boldsymbol{\ell}}$ of inner problem 5 via $\tilde{y} = \min\{L(\theta_1), L(\theta_2), \ldots, L(\theta_k)\}$. This estimator is positively biased but it is effectiveness even without knowledge of the sampling distribution. Here, we introduce an alternative *unbiased* estimator for $y_{\boldsymbol{\ell}}$ assuming the underlying sampling distribution is a uniform distribution with *unknown* width. Like previous works on estimating distribution extrema (Lee & Miller, 2022), such alternative estimators will produce an estimate slightly smaller than the observed minimum.

Corollary B.1. *Let $L(\theta_1), L(\theta_2), \ldots, L(\theta_k)$ be k samples drawn randomly from the set $S'_{\boldsymbol{\ell}} = \{L(\theta) \mid \ell(\theta) = \boldsymbol{\ell}\}$. Let y_{min} be the minimum of these samples and y_{max} be the maximum of these samples. If sample $L(\theta_i) \sim U(y_{\boldsymbol{\ell}}, y_{\boldsymbol{\ell}} + \alpha)$ for $i = 1, 2, \ldots, k$, then $\tilde{y} = y_{min} - \frac{\alpha}{k+1}$ is an unbiased estimator of $y_{\boldsymbol{\ell}}$. Furthermore, if α is unknown, it can be estimated as $\hat{\alpha} = \frac{k+1}{k-1}(y_{max} - y_{min})$ and $y_{min} - \frac{\hat{\alpha}}{k+1}$ is still an unbiased estimator of $y_{\boldsymbol{\ell}}$.*

Proof. Since sample $L(\theta_i) \sim U(y_{\boldsymbol{\ell}}, y_{\boldsymbol{\ell}} + \alpha)$ for $i = 1, 2, \ldots, k$, we have shown in Theorem 3.2 that $y_{\min} \sim y_{\boldsymbol{\ell}} + \alpha \mathrm{B}(1, k)$. Hence, we can see that

$$\begin{aligned}\mathbb{E}\left(y_{\min} - \frac{\alpha}{k+1}\right) &= y_{\boldsymbol{\ell}} + \alpha \mathbb{E}(\mathrm{B}(1,k)) - \frac{\alpha}{k+1} \\ &= y_{\boldsymbol{\ell}} + \alpha\left(\frac{1}{k+1}\right) - \frac{\alpha}{k+1} \\ &= y_{\boldsymbol{\ell}}\end{aligned} \tag{7}$$

where we have used the fact that $\mathbb{E}(\mathrm{B}(1,k)) = \frac{1}{1+k}$. Hence, $\tilde{y} = y_{\text{min}} - \frac{\alpha}{k+1}$ is an unbiased estimator of $y_{\boldsymbol{\ell}}$.

Next, it is sufficient to prove that $\hat{\alpha}$ is an unbiased estimator of α. First, notice that $\mathbb{E}(y_{\text{max}}) = y_{\boldsymbol{\ell}} + \alpha\mathbb{E}(\mathrm{B}(k,1)) = y_{\boldsymbol{\ell}} + \frac{k\alpha}{k+1}$. Then, we have that

$$\begin{aligned}\mathbb{E}(\hat{\alpha}) &= \frac{k+1}{k-1}(\mathbb{E}(y_{\text{max}}) - \mathbb{E}(y_{\text{min}})) \\ &= \frac{k+1}{k-1}(y_{\boldsymbol{\ell}} + \frac{k\alpha}{k+1} - y_{\boldsymbol{\ell}} - \frac{\alpha}{k+1}) \\ &= \alpha.\end{aligned} \tag{8}$$

Hence, $\tilde{\alpha}$ is an unbiased estimator of α. By linearity of expectation and (7), it follows that $\mathbb{E}(y_{\text{min}} - \frac{\hat{\alpha}}{k+1})$ is also an unbiased estimator of $y_{\boldsymbol{\ell}}$. □

Although this estimator is unbiased, it requires us to know the underlying distribution (in this case, a uniform distribution). For general distribution families, an unbiased estimator may not have a closed form solution and is difficult to construct (in fact, these estimators may result in nonsensical minimum loss estimates such as negative accuracies in our problem setting). Last but not least, such alternative estimators of $y_{\boldsymbol{\ell}}$ are not that useful because they do not recover the system parameters θ^* which practitioners can actually use for their systems. For example, for our alternative estimator we do not know which system parameters actually correspond to our estimate $\tilde{y} = y_{\text{min}} - \frac{\alpha}{k+1}$. Eventually, we can only use the sample which had the smallest observed system loss to retrieve usable system parameters (which is actually just the 1st order statistic estimator used in A-BAD-BO). Hence, we argue that relying on the simple 1st order statistic estimator in Theorem 3.2 as an estimator is a more practical choice.

C. Proofs

C.1. Proof of Theorem 3.1

Theorem 3.1. *Let $S_{\boldsymbol{\ell}} \triangleq \{\theta \mid \ell(\theta) = \boldsymbol{\ell}\}$. Then, θ^* is a solution of the original system optimization problem, $\min_\theta L(\theta)$, iff $\boldsymbol{\ell}^* = \ell(\theta^*)$ is a solution of the reparameterized problem:*

$$\min\nolimits_{\boldsymbol{\ell}} \min\nolimits_{\theta \in S_{\boldsymbol{\ell}}} L(\theta) \tag{4}$$

where $\boldsymbol{\ell} \in \mathbb{R}^n$ is a local loss configuration of the given system with n ML components.

Proof. We prove Theorem 3.1 via two steps. First, we introduce Lemma C.1 to show that we can reparameterize any optimization problem $\min_x f(x)$ (while retaining the solution set *exactly*) under regular assumptions. Second, we show that our problem setting satisfies these assumptions, allowing us to apply the Lemma C.1 directly.

Lemma C.1. *Let $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^n$. Also, consider well-defined functions f over $\mathbb{R}^d \to \mathbb{R}$ and g over $\mathbb{R}^d \to \mathbb{R}^n$. Then x^* is a solution of $\operatorname{argmin}_x f(x)$ if and only if $y^* = g(x^*)$ is a solution of the second optimization problem over domain $\{y \mid \exists x, g(x) = y\}$:*

$$\begin{aligned}\min_y \min_x \quad & f(x) \\ \textit{s.t.} \quad & g(x) = y\end{aligned}$$

Proof. We prove Lemma C.1 by showing that $\iff$ (if and only if implication) holds true w.r.t. both statements. To begin, let $x, y \in \mathbb{R}^n$. Let f and g be well-defined functions over $\mathbb{R}^n \to \mathbb{R}$ and $\mathbb{R}^n \to \mathbb{R}^n$ respectively (the term "well-defined" here implies that $\forall a \in \mathbb{R}^n$, $f(a)$ and $g(a)$ both produce unique function values).

($\Rightarrow$) If x^* is the solution of $\min_x f(x)$, then because g is well-defined, there exists a corresponding $g(x^*) \in \mathbb{R}^n$. We want to show that $y^* = g(x^*)$ is the solution to the following reparameterized optimization problem:

$$\begin{aligned}\min_y \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = y,\end{aligned}$$

To show this, consider any $y' \neq g(x^*)$ such that $S_{y'} = \{x'; g(x') = y'\} \neq \varnothing$. It is clear that $\forall x' \in S_{y'}, f(x') \geq f(x^*)$ because x^* is the global minimizer of f. Thus, $g(x^*)$ must be a solution for the reparameterized optimization problem.

($\Leftarrow$) Conversely, if y^* is the solution for the reparameterized optimization problem. Then consider the set $S_{y^*} = \{x; g(x) = y^*\}$ (This set is non-empty by definition, since y^* is the solution and hence there exists x such that $g(x) = y^*$). Let $x^* = \operatorname{argmin}_{x \in S_{y^*}} f(x)$. By definition that y^* is the solution of the reparameterized problem, $\min_{x \in S_{y^*}} f(x) \leq \min_{x \in S_{y'}} f(x)$ for all $y' \neq y^*$. Therefore, x^* must be a minimizer of f and a solution of $\min_x f(x)$. □

Our reparameterization theorem can be proven fairly straightforwardly by applying the result of Lemma C.1. First, we observe that $\ell(\theta)$ which represents the local loss function of every ML component in the system (Sec. 2.2) and $L(\theta)$ which represents the system loss are clearly well defined functions (i.e., they are valid functions where for any θ, they produce unique loss values). Second, we simply replace the functions L with f and ℓ with g and observe that our reparameterized problem Equation (4) in Theorem 3.1 reduces to the form in Lemma C.1. □

C.2. Proof of Theorem 3.2

Theorem 3.2. *Let $L(\theta_1), L(\theta_2), \ldots, L(\theta_k)$ be k samples drawn randomly from the set $S'_{\boldsymbol{\ell}} = \{L(\theta) \mid \ell(\theta) = \boldsymbol{\ell}\}$. Furthermore, let $S'_{\boldsymbol{\ell}}$ be lower bounded by $y_{\boldsymbol{\ell}}$ and upper bounded by $y_{\boldsymbol{\ell}} + \alpha$ for some $\alpha > 0$. Then, the 1st order statistic estimator $\tilde{y} = \min\{L(\theta_1), L(\theta_2), \ldots, L(\theta_k)\}$ follows a distribution of $y_{\boldsymbol{\ell}} + \epsilon$ with a non-negative random variable ϵ and the following holds:*

- *If each sample $L(\theta_i) \sim U(y_{\boldsymbol{\ell}}, y_{\boldsymbol{\ell}} + \alpha)$ for $i = 1, 2, \ldots, k$, then $\epsilon = \alpha\epsilon'$ with $\epsilon' \sim B(1, k)$ where B is the Beta distribution.*
- *If each sample $L(\theta_i) \sim y_{\boldsymbol{\ell}} + \exp_t(\lambda, \alpha)$ for $i = 1, 2, \ldots, k$ where $\exp_t(\lambda, \alpha)$ is a truncated exponential distribution governed by rate parameter λ and truncated at $\alpha > 0$, then ϵ is a random variable governed by the probability density function*

$$pdf_\epsilon(u) = \frac{\lambda k e^{-\lambda u}}{1 - e^{-\lambda\alpha}} \left(\frac{e^{-\lambda u} - e^{-\lambda\alpha}}{1 - e^{-\lambda\alpha}}\right)^{k-1} \text{ on } u \in [0, \alpha]\,.$$

Proof. Our proof considers a simple probability scenario: Let $X_1, X_2, \ldots, X_k$ be k samples randomly drawn from a sampling distribution and $X_{\text{min}} = \min\{X_1, X_2, \ldots, X_k\}$. This scenario mirrors the setting in Theorem 3.2. Our goal is to derive the distribution of X_{min} and show that it is exactly the same as the distribution of $\tilde{y}$ shown in the Theorem 3.2.

(a) If $X_i \sim U(y_{\boldsymbol{\ell}}, y_{\boldsymbol{\ell}} + \alpha)$, then the *cumulative density function* (CDF) of X_{min} is

$$\begin{aligned}\text{cdf}_{(X_{\text{min}})}(u) &= 1 - \mathbb{P}(X_{\text{min}} \geq u)\\ &= 1 - \mathbb{P}(X_1 \geq u, X_2 \geq u, \ldots, X_k \geq u)\\ &= 1 - \left(1 - \frac{u - y_{\boldsymbol{\ell}}}{\alpha}\right)^k, \quad y_{\boldsymbol{\ell}} \leq u \leq y_{\boldsymbol{\ell}} + \alpha.\end{aligned}$$

and the *probability density function* (PDF) can be computed as

$$\begin{aligned}\text{pdf}_{(X_{\text{min}})}(u) &= \frac{\partial}{\partial u}\left(1 - \left(1 - \frac{u - y_{\boldsymbol{\ell}}}{\alpha}\right)^k\right)\\ &= \frac{k}{\alpha}\left(1 - \frac{u - y_{\boldsymbol{\ell}}}{\alpha}\right)^{k-1}\end{aligned}$$

which is exactly equals to the PDF of a random variable $y_{\boldsymbol{\ell}} + \alpha\epsilon'$ with $\epsilon' \sim \text{B}(1, k)$.

(b) If $X_i \sim \exp_t(\lambda, \alpha)$, we first note that the CDF of any truncated distribution on $[0, \alpha]$ is $\frac{F(u) - F(0)}{F(\alpha) - F(0)}$ where F is the CDF of the original untruncated distribution. Also, we note that for the exponential distribution, $F(u) = 1 - e^{-\lambda u}$. Hence, The CDF of X_{min} is

$$
\begin{aligned}
\mathrm{cdf}_{(X_{\min})}(u) &= 1 - \mathbb{P}(X_{\min} \geq u) \\
&= 1 - \mathbb{P}(X_1 \geq u, X_2 \geq u, \ldots, X_k \geq u) \\
&= 1 - \left(1 - \frac{1 - e^{-\lambda u}}{1 - e^{-\lambda \alpha}}\right)^k, \quad y_{\boldsymbol{\ell}} \leq u \leq y_{\boldsymbol{\ell}} + \alpha.
\end{aligned}
$$

and hence the PDF of $X_{\min}$ can be computed as

$$
\begin{aligned}
\mathrm{pdf}_{(X_{\min})}(u) &= \frac{\partial}{\partial u} F_{(X_{\min})}(u) \\
&= \frac{\lambda k e^{-\lambda u}}{1 - e^{-\lambda \alpha}} \left(\frac{e^{-\lambda u} - e^{-\lambda \alpha}}{1 - e^{-\lambda \alpha}}\right)^{k-1}.
\end{aligned}
$$

□

C.3. Proof of Theorem 4.1

Theorem 4.1. *Let $S'_{\boldsymbol{\ell}}$ be the set defined in Theorem 3.2 and f be the system optimization objective with bounded RKHS norm: $||f||_\kappa = \sqrt{\langle f, f\rangle_\kappa} \leq B$ w.r.t. kernel κ. Then, running A-BAD-BO over f using the IGP-LCB acquisition function (Chowdhury & Gopalan, 2017) and sampling size $k = t$ at BO iteration $t = 1, \ldots, T$ with γ_T as the maximum information gain after T iterations yields the following attained cumulative regret with probability at least $1 - \delta$:*

- $\tilde{R}_T = O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4}) + \frac{\alpha\psi(T+1)}{\sqrt{\delta}}\right)$ *if each random sample from $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width $\alpha > 0$. ψ refers to the digamma function.*
- $\tilde{R}_T = O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4}) + \sqrt{\frac{(1-e^{-\alpha})T}{\delta}}\right)$ *if each random sample from $S'_{\boldsymbol{\ell}}$ is exponentially distributed with rate $\lambda = 1$ and truncated at α with $0 < \alpha \leq 1$.*

Proof. We provide the proof of the sub-linear $\tilde{R}_T$ growth of A-BAD-BO in Theorem 4.1 by establishing upper bounds of $|\mu_t(x) - f(x)|$ and ϵ_t separately at each BO iteration t. To do so, we introduce the following two Lemmas.

Our first Lemma is taken from from known literature on Kernelized Bandits (Chowdhury & Gopalan, 2017) and provides the upper bound on difference between $f(x_t)$ and $\mu_t(x)$ at each BO iteration t.

Lemma C.2. *Let $||f||_\kappa = \sqrt{\langle f, f\rangle_\kappa} \leq B$. Also, assume that the observation noise associated with each BO iteration is R-sub-Gaussian with $R > 0$. Then with probability at least $1 - \delta$, the following holds for BO iteration $t \leq T$:*

$$
|\mu_t(x) - f(x)| \leq \left(B + R\sqrt{2(\gamma_t + 1 + \ln(1/\delta)}\right)\sigma_t(x) \tag{9}
$$

where γ_t is the maximum information gain after t observations and $\mu_t(x), \sigma_t^2(x)$ are mean and variance of posteror distribution of GP defined in Equation 1, with $\lambda = 1 + 2/T$.

Our second Lemma attempts to bound the expectation and variance of ϵ_t, the non-negative observation noise (in our case, it corresponds to the estimation error involved in solving the inner problem) at each BO iteration t. Similiar to Theorem 3.2, we assume ϵ_t belongs to one of two random distribution families.

Lemma C.3. *Let $S'_{\boldsymbol{\ell}}$ and ϵ_t be that defined in Theorem 3.2 with $k = t$. Then*

(a) If $\epsilon_t \sim \alpha B(1, t)$, then $\mathbb{E}(\epsilon_t) = \frac{\alpha}{1+t}$ and $\mathrm{Var}(\epsilon_t) = \frac{\alpha^2 t}{(t+1)^2(t+2)}$.

(b) If ϵ_t is a random variable governed by probability density function $f_{\epsilon_t}(u) = \frac{\lambda t e^{-\lambda u}}{1-e^{-\lambda\alpha}}\left(\frac{e^{-\lambda u} - e^{-\lambda\alpha}}{1-e^{-\lambda\alpha}}\right)^{t-1}$ with $0 < \alpha \leq 1$, $\lambda = 1$ and $u \in [0, \alpha]$, then $\mathbb{E}(\epsilon_t) \leq \frac{4(1-e^{-\alpha})}{t+1}$ and $\mathrm{Var}(\epsilon_t) \leq \mathbb{E}(\epsilon_t)$.

Proof. **(a)** For the case where $\epsilon_t \sim \alpha \mathrm{B}(1,t)$, its expectation and variance are well known results for Beta distributions.

(b) For $f_{\epsilon_t}(u) = \frac{\lambda t e^{-\lambda u}}{1-e^{-\lambda\alpha}}\left(\frac{e^{-\lambda u}-e^{-\lambda\alpha}}{1-e^{-\lambda\alpha}}\right)^{t-1}$ with $0<\alpha<1$, $\lambda=1$ and $u\in[0,\alpha]$, we have

$$\begin{aligned}
\mathbb{E}(\epsilon_t) &= \int_0^\alpha u f_{\epsilon_t}(u)\,du \\
&= \int_0^\alpha \frac{ute^{-u}}{1-e^{-\alpha}}\left(\frac{e^{-u}-e^{-\alpha}}{1-e^{-\alpha}}\right)^{t-1} du \\
&= \frac{t}{(1-e^{-\alpha})^t}\int_0^\alpha ue^{-u}\left(e^{-u}-e^{-\alpha}\right)^{t-1} du \\
&\overset{(1)}{\leq} \frac{t}{(1-e^{-\alpha})^t}\int_0^\alpha u\left(e^{-u}-e^{-\alpha}\right)^{t-1} du \\
&\overset{(2)}{\leq} \frac{t}{(1-e^{-\alpha})^t}\int_0^\alpha u\left(1-\frac{u}{2}-e^{-\alpha}\right)^{t-1} du \\
&\overset{(3)}{=} \frac{4t}{(1-e^{-\alpha})^t}\left(-\frac{u(1-e^{-\alpha}-\frac{u}{2})}{2t}-\frac{(1-e^{-\alpha}-\frac{u}{2})^{t+1}}{t(t+1)}\right)\Bigg|_{u=0}^{u=\alpha} \\
&\leq \frac{4(1-e^{-\alpha})}{t+1}
\end{aligned} \tag{10}$$

where $\overset{(1)}{\leq}$ makes use of the fact that $e^{-\alpha}\leq 1$ for $\alpha>0$, $\overset{(2)}{\leq}$ uses the inequality $e^{-u}\leq 1-\frac{u}{2}$ for $u\in[0,\alpha]$, and $\alpha\leq 1$ and $\overset{(3)}{=}$ is derived via solving the definite integral by parts.

Next, the upper bound of the variance of ϵ_t can be derived by observing that

$$\begin{aligned}
\mathrm{Var}(\epsilon_t) &= \int_0^\alpha u^2 f_{\epsilon_t}(u)\,du \\
&\overset{(1)}{\leq} \alpha\int_0^\alpha u f_{\epsilon_t}(u)\,du \\
&\overset{(2)}{\leq} \int_0^\alpha u f_{\epsilon_t}(u)\,du \\
&= \mathbb{E}(\epsilon_t)
\end{aligned} \tag{11}$$

where $\overset{(1)}{\leq}$ makes use of the fact that ϵ_t lies in $[0,\alpha]$ and $\overset{(2)}{\leq}$ makes use of the fact that $0<\alpha\leq 1$. This completes the proof on the bounds on $\mathbb{E}(\epsilon_t)$ and $\mathrm{Var}(\epsilon_t)$. □

We are now ready to prove Theorem 4.1. First, we observe that x_t at each BO iteration t is chosen via the IGP-LCB acquisition function (i.e., $x_t = \mathrm{argmin}_x\, \mu_{t-1}(x) - \beta_t\sigma_{t-1}(x)$ and $\beta_t = B + R\sqrt{2(\gamma_{t-1}+1+\ln(1/\delta))}$ where the observation noise associated with each BO iteration is R-sub Gaussian). Thus, we can see that at each iteration $t\geq 1$, we have $-\mu_{t-1}(x_t)+\beta_t\sigma_{t-1}(x_t) \geq -\mu_{t-1}(x^*)+\beta_t\sigma_{t-1}(x^*)$. It then follows that for all $t\geq 1$ and with probability at least $1-\delta$,

$$\begin{aligned}
|f(x_t)-f(x^*)| &\overset{(1)}{\leq} f(x_t)-\mu_{t-1}(x^*)-\beta_t\sigma_{t-1}(x^*) \\
&\overset{(2)}{\leq} f(x_t)-\mu_{t-1}(x_t)+\beta_t\sigma_{t-1}(x_t) \\
&\leq \beta_t\sigma_{t-1}(x_t)+|\mu_{t-1}(x_t)-f(x_t)| \\
&\leq 2\beta_t\sigma_{t-1}(x_t)
\end{aligned} \tag{12}$$

where $\overset{(1)}{\leq}$ makes use of Lemma C.2, $\overset{(2)}{\leq}$ holds by the design of the acquisition function where we select x_t over x^* at BO iteration t.

Therefore, it follows that our **attained cumulative regret** can be bounded as

$$
\begin{aligned}
\tilde{R}_T &= \sum_{t=1}^{T} |\tilde{y}_t - f(x^*)| \\
&= \sum_{t=1}^{T} |f(x_t) - f(x^*) + \epsilon_t| \\
&\overset{(1)}{=} \sum_{t=1}^{T} |f(x_t) - f(x^*)| + \sum_{t=1}^{T} \epsilon_t \\
&\overset{(2)}{\leq} 2\beta_T \sum_{t=1}^{T} \sigma_{t-1}(x_t) + \sum_{t=1}^{T} \epsilon_t \\
&\overset{(3)}{=} 2\left(B + R\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}\right) \sum_{t=1}^{T} \sigma_{t-1}(x_t) + \sum_{t=1}^{T} \epsilon_t \\
&\overset{(4)}{\leq} 2\left(B + R\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}\right) \sum_{t=1}^{T} \sigma_{t-1}(x_t) + \sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}} \\
&\overset{(5)}{=} 2\left(B + R\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}\right) O(\sqrt{T\gamma_T}) + \sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}} \\
&= O\left(\sqrt{T}(B\sqrt{\gamma_T} + R\gamma_T)\right) + \sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}} \\
&\overset{(6)}{=} O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4})\right) + \sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}}
\end{aligned}
\tag{13}
$$

where

- $\overset{(1)}{=}$ uses the fact that ϵ_t is non-negative in our problem setting (Theorem 3.2).
- $\overset{(2)}{\leq}$ is derived from Eq. (12).
- $\overset{(3)}{=}$ uses the definition of β_T in IGP-LCB acquisition function (Chowdhury & Gopalan, 2017).
- $\overset{(4)}{\leq}$ uses Chebyshev's inequality over ϵ_t.
- $\overset{(5)}{=}$ uses the fact that $\sum_{t=1}^{T} \sigma_{t-1}(x_t) \leq O(\sqrt{T\gamma_T})$ as shown in **Lemma 4** by Chowdhury & Gopalan (2017).
- $\overset{(6)}{=}$ uses the fact that ϵ_t is bounded on $[0, \alpha]$ and all such bounded random variables are sub-Gaussian with proxy variance $R = \frac{\alpha^2}{4}$ (Arbel et al., 2019). In fact, this variance proxy can be refined further to $\frac{\alpha^2}{4(2+t)}$ for Beta-distributed ϵ_t (Marchal & Arbel, 2017).

Next, we will bound $\sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}}$ using the expectation and variance of ϵ_t defined in Lemma C.3. This bound clearly depends on the distribution of ϵ_t (which is dependent on the distribution of S'_ℓ).

(A) If $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width α, then with probability $1-\delta$ we have

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}(\epsilon_t)+\sum_{t=1}^{T}\sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}} &\overset{(1)}{=} \sum_{t=1}^{T}\frac{\alpha}{1+t}+\sum_{t=1}^{T}\sqrt{\frac{\alpha^2 t}{\delta(1+t)^2(2+t)}}\\ &\le \sum_{t=1}^{T}\frac{\alpha}{1+t}+\sum_{t=1}^{T}\frac{\alpha}{\sqrt{\delta}(1+t)}\\ &\overset{(2)}{=} O\left(\frac{\alpha\psi(T+1)}{\sqrt{\delta}}\right)\end{aligned} \tag{14}$$

where $\overset{(1)}{=}$ makes use of Lemma C.3 (for uniform case) directly and $\overset{(2)}{=}$ makes use of the fact that $\sum_{t=1}^{T}\frac{1}{t}=\psi(T+1)+e$ (Spanier et al., 1988) for digamma function ψ. With the previous results shown in (13), this completes the proof on the attained regret bound of A-BAD-BO in Theorem 4.1 for the case of uniformly distributed $S'_{\boldsymbol{\ell}}$.

(B) If $S'_{\boldsymbol{\ell}}$ is exponentially distributed and truncated at $\alpha>0$ with rate $\lambda=1$, then with probability $1-\delta$ we have

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}(\epsilon_t)+\sum_{t=1}^{T}\sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}} &\overset{(1)}{\le} \sum_{t=1}^{T}\frac{4(1-e^{-\alpha})}{t+1}+\sum_{t=1}^{T}\sqrt{\frac{4(1-e^{-\alpha})}{\delta(t+1)}}\\ &\overset{(2)}{\le} 4(1-e^{-\alpha})(\psi(T+1)+e)+\sum_{t=1}^{T}\sqrt{\frac{4(1-e^{-\alpha})}{\delta(t+1)}}\\ &= 4(1-e^{-\alpha})(\psi(T+1)+e)+\sqrt{\frac{4(1-e^{-\alpha})}{\delta}}\sum_{t=1}^{T}\frac{1}{\sqrt{t+1}}\\ &\overset{(3)}{\le} 4(1-e^{-\alpha})(\psi(T+1)+e)+\sqrt{\frac{4(1-e^{-\alpha})}{\delta}}2\sqrt{T}\\ &= O(\psi(T+1))+O\left(\sqrt{\frac{(1-e^{-\alpha})T}{\delta}}\right)\end{aligned} \tag{15}$$

where $\overset{(1)}{\le}$ makes use of Lemma C.3 (for truncated exponential case) directly, $\overset{(2)}{\le}$ again uses the fact that $\sum_{t=1}^{T}\frac{1}{t}=\psi(T+1)+e$ (Spanier et al., 1988) and $\overset{(3)}{\le}$ uses the fact that $\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\le 2\sqrt{T}$. Since the second term is of a larger growth rate, we can regard the bound as $O\left(\sqrt{\frac{(1-e^{-\alpha})T}{\delta}}\right)$. With the previous results shown in Equation 13, this completes the proof on the attained regret bound of A-BAD-BO in Theorem 4.1 for the case of exponential (truncated) distributed $S'_{\boldsymbol{\ell}}$. □

C.4. Effect of other sampling schemes on regret bounds

In real life settings, it may be computationally expensive to adopt a linearly increasing sampling scheme (e.g., $k=t$) for large BO iterations. In the extreme case, every system query may be computationally expensive and hence, a fixed sampling size (or a sub-linear sampling scheme) can be chosen prior to the algorithm to reduce computational cost. We provide two additional analyses with a sampling scheme of fixed k and $k=\sqrt{t}$ (for each iteration t). Corollary C.4 and C.5 tell us that (a) with a fixed sampling size, A-BAD-BO eventually converges to a certain range of the minimum system loss and (b) with a sampling scheme of $k=\sqrt{t}$, A-BAD-BO still achieves sub-linear regret growth.

Corollary C.4. *Following the conditions laid out in Theorem 4.1 but with fixed sampling size k at each BO iteration $t=1,\ldots,T$, A-BAD-BO yields with probability at least $1-\delta$:*

- $\lim_{T\to\infty}\frac{\tilde{R}_T}{T}\le\frac{\alpha}{1+k}+\alpha\sqrt{\frac{k}{\delta(k+1)^2(k+2)}}$ *if $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width $\alpha>0$.*
- $\lim_{T\to\infty}\frac{\tilde{R}_T}{T}\le\frac{4(1-e^{-\alpha})}{k+1}+2\sqrt{\frac{1-e^{-\alpha}}{\delta(k+1)}}$ *if $S'_{\boldsymbol{\ell}}$ is exponentially distributed and truncated at $0<\alpha\le 1$ and rate $\lambda=1$.*

Proof. First, (13) has shown that $\tilde{R}_T \leq O\left(B\sqrt{T\gamma_T} + \sqrt{T\gamma_T(\gamma_T + 1 + \ln(1/\delta))}\right) + \sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}}$. Hence, $\lim_{T\to\infty} \frac{\tilde{R}_T}{T} = \lim_{T\to\infty} \frac{1}{T}\left(\sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T}\sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}}\right)$ due to the sub-linear growth of $O\left(B\sqrt{T\gamma_T} + \sqrt{T\gamma_T(\gamma_T + 1 + \ln(1/\delta))}\right)$. Furthermore, since sampling size k is fixed, it follows from Lemma C.3 that $\epsilon_t = \epsilon_k$ for every BO iteration t (i.e., the error term is identical at each iteration). Hence, we have

$$\begin{aligned}
\lim_{T\to\infty} \frac{\tilde{R}_T}{T} &\leq \lim_{T\to\infty} \frac{1}{T}\left(\sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T}\sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}}\right) \\
&= \lim_{T\to\infty} \frac{1}{T}\left(T\mathbb{E}(\epsilon_k) + T\sqrt{\frac{\mathrm{Var}(\epsilon_k)}{\delta}}\right) \\
&= \mathbb{E}(\epsilon_k) + \sqrt{\frac{\mathrm{Var}(\epsilon_k)}{\delta}}.
\end{aligned} \tag{16}$$

Finally, we again use Lemma C.3 to see that if $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width $\alpha > 0$, then $\mathbb{E}(\epsilon_k) + \sqrt{\frac{\mathrm{Var}(\epsilon_k)}{\delta}} = \frac{\alpha}{1+k} + \alpha\sqrt{\frac{k}{\delta(k+1)^2(k+2)}}$. If $S'_{\boldsymbol{\ell}}$ is truncated exponentially distributed with $0 < \alpha \leq 1$ and rate $\lambda = 1$, $\mathbb{E}(\epsilon_k) + \sqrt{\frac{\mathrm{Var}(\epsilon_k)}{\delta}} = \frac{4(1-e^{-\alpha})}{k+1} + 2\sqrt{\frac{1-e^{-\alpha}}{\delta(k+1)}}$. This concludes the proof of the regret-growth of A-BAD-BO with fixed sampling size k. □

Therefore, if a practitioner decides to use a fixed sampling size, they would make a decision on what range they want to be within the optimal system loss and pick an appropriate k which satisfies this range using Corollary C.4 (e.g., for the uniform distribution case with $\alpha = 1$, we can pick $k \approx 10$ to get within a 0.1 range of the minimum system loss).

Our next Corollary shows that the attained cumulative regret of A-BAD-BO is still sub-linear w.r.t. a sampling scheme of $k = \sqrt{t}$ at each BO iteration t:

Corollary C.5. *Following the conditions laid out in Theorem 4.1 but with sub-linear sampling scheme of $k = \sqrt{t}$ at each BO iteration $t = 1, \ldots, T$, A-BAD-BO yields with probability at least $1 - \delta$:*

- $\tilde{R}_T = O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4}) + \frac{\alpha\sqrt{T}}{\delta}\right)$ *if $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width $\alpha > 0$.*
- $\tilde{R}_T = O\left(\sqrt{T}(B\sqrt{\gamma_T} + \frac{\alpha^2\gamma_T}{4}) + 4(1-e^{-\alpha})(2\sqrt{T}) + \sqrt{\frac{4(1-e^{-\alpha})}{\delta}} H_T^{0.25}\right)$ *if $S'_{\boldsymbol{\ell}}$ is exponentially distributed and truncated at $0 < \alpha \leq 1$ and rate $\lambda = 1$. $H_T^{0.25}$ refers to the generalized Harmonic number (Choi & Srivastava, 2011).*

Proof. We again use the same results from (13) that $\tilde{R}_T \leq O\left(B\sqrt{T\gamma_T} + \sqrt{T\gamma_T(\gamma_T + 1 + \ln(1/\delta))}\right) + \sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}}$ and prove the bounds for sum of the expectation and variance of ϵ_t until BO iteration T.

(A) If $S'_{\boldsymbol{\ell}}$ is uniformly distributed with width α and we have a sampling scheme of $k = \sqrt{t}$, then we have with probability $1 - \delta$ that

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T}\sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\delta}} &\overset{(1)}{=} \sum_{t=1}^{T} \frac{\alpha}{1+\sqrt{t}} + \sum_{t=1}^{T}\sqrt{\frac{\alpha^2\sqrt{t}}{\delta(1+\sqrt{t})^2(2+\sqrt{t})}} \\
&\leq \sum_{t=1}^{T} \frac{\alpha}{1+\sqrt{t}} + \sum_{t=1}^{T} \frac{\alpha}{\sqrt{\delta}\sqrt{t}} \\
&\overset{(2)}{\leq} 2\alpha\sqrt{T} + \frac{2\alpha\sqrt{T}}{\sqrt{\delta}} \\
&= O\left(\frac{\alpha\sqrt{T}}{\sqrt{\delta}}\right)
\end{aligned} \tag{17}$$

$\overset{(1)}{=}$ makes use of Lemma C.3 (for uniform case) directly and $\overset{(2)}{\leq}$ uses the fact that $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 2\sqrt{T}$.

(B) If $S'_{\boldsymbol{\ell}}$ is exponentially distributed and truncated at $\alpha > 0$ with rate $\lambda = 1$, then with probability $1 - \delta$ we have

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}(\epsilon_t) + \sum_{t=1}^{T} \sqrt{\frac{\text{Var}(\epsilon_t)}{\delta}} &\overset{(1)}{\leq} \sum_{t=1}^{T} \frac{4(1-e^{-\alpha})}{\sqrt{t}+1} + \sum_{t=1}^{T} \sqrt{\frac{4(1-e^{-\alpha})}{\delta(\sqrt{t}+1)}} \\
&\overset{(2)}{\leq} 4(1-e^{-\alpha})(2\sqrt{T}) + \sum_{t=1}^{T} \sqrt{\frac{4(1-e^{-\alpha})}{\delta(\sqrt{t}+1)}} \\
&\overset{(3)}{\leq} 4(1-e^{-\alpha})(2\sqrt{T}) + \sqrt{\frac{4(1-e^{-\alpha})}{\delta}} H_T^{0.25}
\end{aligned} \tag{18}$$

where $H_T^{0.25}$ is the *generalized Harmonic number* (Choi & Srivastava, 2011) and is sub-linear w.r.t. T. $\overset{(1)}{\leq}$ makes use of Lemma C.3 (for truncated exponential case) directly, $\overset{(2)}{\leq}$ again uses the fact that $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 2\sqrt{T}$ and $\overset{(3)}{\leq}$ uses the definition of the generalized Harmonic number (Choi & Srivastava, 2011). □

Corollary C.5 tells us that if we use a sampling scheme of $k = \sqrt{t}$, our attained regret bounds is higher than that for a sampling scheme of $k = t$ shown in Theorem 4.1 (with a smaller sampling size, our subroutine would incur more noise, thus introducing larger *observation noise* in the BO process, increasing cumulative regret growth). However, the advantage is that we may not need as much computational budget since we do not need to perform as many rounds of gradient descent (in our subroutine) per iteration. A practitioner would choose the suitable sampling scheme depending on the problem setting.

D. Additional Experimental Results and Discussions

D.1. Additional details on experimental setup

Our experiments use a squared exponential kernel to model all GPs whenever possible. Also, we use the GP-LCB (Srinivas et al., 2010) acquisition function for A-BAD-BO. In our experimental results shown in Fig. 3 and 4, we used the same data generation seed and repeat our experiments for 5 trials and show the mean system loss achieved after each system query alongside its standard error. We use negative predictive accuracy as the system loss function for LLM system and mean-squared error for the rest of the systems.

In general, real-world systems are typically present in commercial settings and are proprietary. Hence, it is difficult to obtain *synchronized* datasets for each ML components and the system (e.g., liver data and also diabetes risk data for the same patient) in our work. To circumvent this challenge, we adopt the following procedure to generate synchronized local and system datasets in our experiments:

1. We arrange a mixture of ML components and black-box components in topological order. The order dictates how prediction generated by one ML component is passed to the next component.
2. Depending on the system, we define the black-box components' behaviors (e.g., weighted sum or some arbitrary function) but their functional representations are unknown during training time.
3. We generate a local dataset for each ML component. For synthetic ML components, we assume the ground-truth parameters are known and generate the local dataset from them. For real-world systems, we use an open source dataset for each ML component (more information is provided next).
4. If needed, we generate a system dataset by passing through input data (generated randomly or taken from open sources) through the system to generate system-level labels and consequentially a system dataset.

In **MNIST** system, we use the MNIST dataset (Deng, 2012) to train 2 ML components as binary digit classifiers. The other ML components consist of differentiable models.

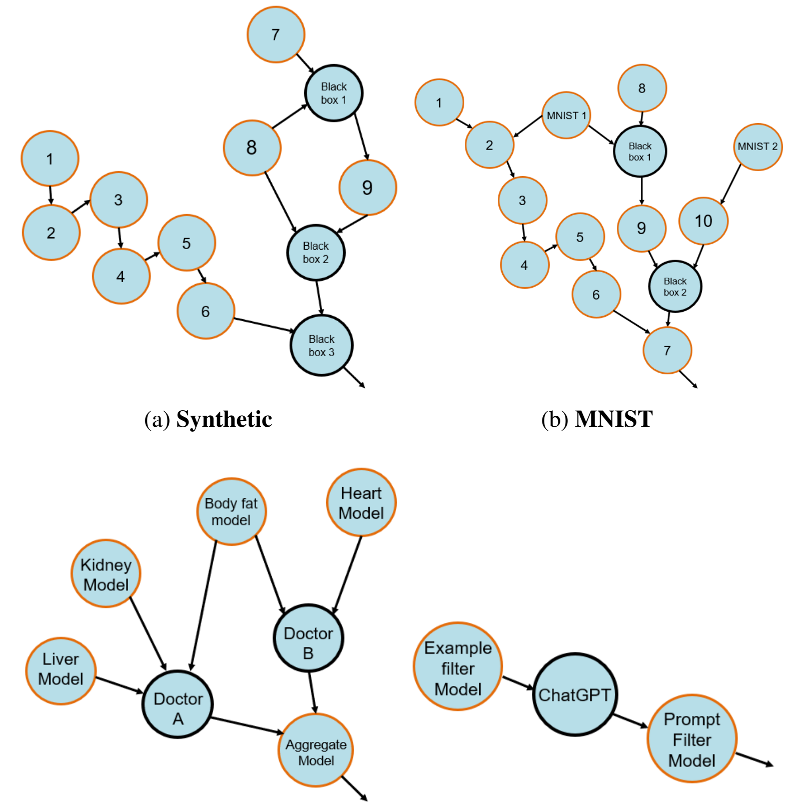
In **Healthcare** system, we use several healthcare models whose local datasets are open source healthcare related datasets; they include a body fat prediction model (Penrose et al., 1985), hepatitis risk prediction model (Hepatitis, 1988), kidney disease prediction model (Soundarapandian & Eswaran, 2015) and health disease prediction model (Siddhartha, 2020). We also place 2 noisy black-box components to imitate human doctors (labeled **Doctor A** and **Doctor B** in Fig. 9c) who make noisy decisions after looking at the model predictions and a final aggregation component which is a simple linear model. A system level dataset is generated by passing a portion of data from local datasets through the system, generating system level labels. This system reflects a possible healthcare system that is used in a real-world setting to provide a diabetes risk score in a hospital setting, therefore assisting doctors in making better medical judgement for a patient.

In **LLM prompt engineering** system, we use a principled approach to generate local and system datasets. First, we generate a single system dataset for prompt engineering by combining available prompt engineering datasets (containing examples and ground-truth prompts) over 20 tasks (Honovich et al., 2022). Our system aims to generate good prompts for a given list of examples (over any task). In addition, we contaminate 50% of examples (by replacing examples with random words). The first ML component is an example filter model whose local dataset contains independent correct example pairs as positively labeled data and incorrect example pairs (e.g., two nonsensical phrases) as negatively labeled data (these can be generated in an unsupervised manner from the prompt engineering dataset by artificially adding wrong examples). After the input examples are filtered (noisy examples are removed), they are passed into ChatGPT (OpenAI, 2023) to generate a list of 5 plausible prompts for these examples [2]. Lastly, a prompt filter model with local dataset consisting of good example-prompt pairs (this can also be generated in an unsupervised manner from the original prompt engineering dataset) is used to evaluate and identify the best prompts from the prompt list. Note that we carefully ensure none of the evaluation examples is present in the local datasets or system datasets.

[2]the ChatGPT model version used in our work is **gpt-3.5-turbo-0301** found in `https://platform.openai.com/docs/models`

D.2. Additional details on system architectures

We also provide more detailed information of the four systems we use in our experiments in Fig. 9. For reproducibility, we provide the exact model type and mathematical function of each component in Table 3.



(a) **Synthetic** (b) **MNIST**

(c) **Healthcare** (d) **LLM Prompt engineering**

Figure 9. System architecture of each experimental system. Nodes highlighted in orange are differentiable ML components while nodes highlighted **black** are black-box components with no closed, mathematical form. ML components with numerical labels are synthetic mathematical models which use synthetic datasets; ML components with alphabetical labels (e.g., Liver Model, MNIST 1 etc.) use real world datasets.

Table 2. Legend for mathematical representation of each ML and black-box component.

Legend	Function representation
$\text{Sin}(\theta_0, \theta_1)$	$\theta_0 \sin(\theta_1 x + 0.5)$
$\text{Poly}(\theta_0, \theta_1)$	$-\theta_0 x + \theta_1 x + e^{\theta_2 x}$
$\text{Exp}(\theta_0, \theta_1)$	$\theta_0 e^{-\theta_1 x}$
$\text{WeightedSum}(\theta_0, \ldots, \theta_m)$	$\theta_0 x_0 + \cdots + \theta_m x_m$
$\text{NoisyWeightedSum}(\theta_0, \ldots, \theta_m, \mu, \sigma)$	$\theta_0 x_0 + \cdots + \theta_m x_m + \epsilon, \epsilon \sim \mathcal{N}(\mu, \sigma^2)$

Table 3. Exact component description of every system used in our experiments. The left column shows the ground truth parameters (wherever possible) for our ML components that we try to learn during optimization. The functions for black-box components are fixed during runtime (no training involved).

	ML components	Black-box components
Synthetic	1. Sin$(0.1, 0.1)$ 2: Poly$(-0.3, 0.3, 0.2)$ 3: Sin$(-0.2, 0.1)$ 4: Exp$(-0.5, 0.5)$ 5: Exp$(-0.2, 0.2)$ 6: Sin$(0.7, -0.5)$ 7: Poly$(0.7, 0.9, -0.5)$ 8: Sin$(0.3, 0.7)$ 9: Exp$(1.1, -0.5)$	Black box 1: NoisyWeightedSum$(1.2, 0.8, 0.3, 0.05)$ Black box 2: NoisyWeightedSum$(0.7, -0.5, 0.3, 0.05)$ Black box 3: NoisyWeightedSum$(0.7, 1.1, 0.3, 0.05)$
MNIST	1: Sin$(-0.4, 0.8)$ 2: WeightedSum$(-0.7, 0.9)$ 3: Poly$(-0.4, 0.8, 0.1)$ 4: Sin$(0.7, 0.9)$ 5: Exp$(0.7, 0.9)$ 6: Poly$(-0.4, 0.8, 0.1)$ 7: Sin$(0.5, 0.8)$ 8: WeightedSum$(0.5, 0.7)$ 9: Exp$(0.9, -0.3)$ 10: Exp$(-0.3, 0.5)$ MNIST 1: CNN (LeCun et al., 1989) MNIST 2: CNN (LeCun et al., 1989)	Black box 1: NoisyWeightedSum$(1.2, 0.8, 0.3, 0.2)$ Black box 2: NoisyWeightedSum$(0.8, 0.8, 0.3, 0.2)$
Healthcare	Liver model: Logistic Regression Kidney model: Logistic Regression Body fat model: Linear Regression Heart Model: Logistic Regression Aggregate Model: WeightedSum$(0.7, 0.3)$	Black box 1: NoisyWeightedSum$(0.7, 0.3, -2.5, 0.5)$ Black box 2: NoisyWeightedSum$(0.4, 0.4, 0.2, -2.5, 0.5)$
LLM	Example filter model: DistilBERT (Sanh et al., 2019) Prompt filter model: DistilBERT (Sanh et al., 2019)	gpt-3.5-turbo-0301 (OpenAI, 2023)

D.3. Additional experimental results with other acquisition functions

Our main results in Fig.3 showed that A-BAD-BO achieves better system performance (e.g. loss, accuracy) across all systems with the Lower Confidence Bound (LCB) (Srinivas et al., 2010) acquisition function. In this section, we also investigated whether using a different acquisition function i.e., Expected Improvement (EI) (Jones et al., 1998), affected the effectiveness of our algorithm.

As shown in Table 4, EI yields similar (at times worse) results as compared to LCB. We believe this occurs because LCB encourages A-BAD-BO to explore different local loss configurations for the system, therefore producing better results. Regardless of the acquisition function, A-BAD-BO is still able to outperform other baselines, further showcasing its effectiveness.

Table 4. Comparison of best system performance achieved with Lower Confidence Bound (LCB) (Srinivas et al., 2010) acquisition function used in A-BAD-BO with the Expected Improvement (EI) acquisition function (Jones et al., 1998). 200 system queries were used across all approaches. The best performing approach is **bolded**.

System	A-BAD-BO (LCB)	A-BAD-BO (EI)	Vanilla BO (EI)	TuRBO (EI)
Synthetic (loss)	0.053±0.081	**0.051±0.033**	0.112±0.043	0.078±0.058
MNIST (loss)	**0.143±0.019**	0.158± 0.007	0.4874±0.099	0.423±0.06
Healthcare (loss)	**3.45±0.333**	5.35±1.14	7.10±0.64	9.09±0.44
LLM (accuracy)	**62.0±5.0**	60.5±6.0	51.3±3.3	53.0±3.0

D.4. Additional experimental results with early local optimization termination

Instead of training each ML component in a system indepedently to convergence (the *local* approach) w.r.t. local data, one might consider training each ML component and terminating the training process early using regularization e.g., weight decay (Andriushchenko et al., 2023). The set of regularization hyper-parameters (one regularization hyper-parameter for each ML component) is then used to control the early termination threshold, and BO can be used to determine the best set of regularization hyper-parameters. In this section, we compare this approach (which we coin as **Early-stopping**) with A-BAD-BO given the same number of system queries.

In particular, at every BO iteration of the **Early-stopping** approach, we train ML components, starting from different initialized component parameters, with a proposed set of regularization hyper-parameters (inputs to the BO problem) until convergence k times (yielding k different set of converged system parameters). Then, we retrieve the set of converged system parameters which yielded the best system loss for that iteration (involving k system queries). The best system loss is treated as the observed output of that BO iteration, and the BO iteration continues.

Table 5. Comparison of best system performance achieved by A-BAD-BO with **Early-stopping** after 200 system queries. The best performing approach is **bolded**.

System	A-BAD-BO ($k = 10$)	Early-stopping ($k = 1$)	Early-stopping ($k = 10$)
Synthetic (loss)	**0.053±0.081**	0.185±0.090	0.0855 ± 0.090
MNIST (loss)	**0.143±0.019**	0.33±0.017	0.23±0.020
Healthcare (loss)	**3.45±0.333**	8.45±2.14	6.55±1.88
LLM (accuracy)	**62.0±5.0**	52.4 ±2.0	55.0±1.5

In our experiments, we used the weight decay coefficient (Andriushchenko et al., 2023) as the regularization hyper-parameter for each ML component. From Table 5, we see that A-BAD-BO outperforms the Early-stopping approach in achieving better system performance after 200 system queries across all 4 systems in our experimental setup.